Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Engenharia Elétrica

Trabalho de Conclusão de Curso

Construção de um simulador virtual para o braço robótico ARM-7220-4

Pedro Augusto Silva Andrade

João Monlevade, MG
2024

Pedro Augusto Silva Andrade

Construção de um simulador virtual para o braço robótico ARM-7220-4

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Ouro Preto como parte dos requisitos para obtenção do Título de Bacharel em Engenharia Elétrica pelo Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto.

Orientador: Dr. Márcio Feliciano Braga

Coorientador: Dra. Wendy Yadira Eras Herrera

Universidade Federal de Ouro Preto
João Monlevade
2024

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

A553c Andrade, Pedro Augusto Silva.
Construção de um simulador virtual para o braço robótico ARM-7220-4. [manuscrito] / Pedro Augusto Silva Andrade. - 2024.
61 f.: il.: color., tab.. + Equações.

Orientador: Prof. Dr. Marcio Feliciano Braga.
Coorientadora: Profa. Dra. Wendy Yadira Eras Herrera.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Engenharia Elétrica .

1. Automação. 2. Cinemática. 3. Engenharia Elétrica. 4. Manipuladores (Mecanismo). 5. MATLAB (Programa de computador). 6. Robótica. I. Braga, Marcio Feliciano. II. Herrera, Wendy Yadira Eras. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 681.5

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE ENGENHARIA ELETRICA

FOLHA DE APROVAÇÃO

Pedro Augusto Silva Andrade

Construção de um simulador virtual para o braço robótico ARM-7220-4

Monografia apresentada ao Curso de Engenharia Elétrica da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de bacharel em Engenharia Elétrica

Aprovada em 25 de junho de 2024.

Membros da banca

Dr. Márcio Feliciano Braga — Orientador — Universidade Federal de Ouro Preto
Dra. Wendy Yadira Eras Herrera — Coorientadora — Universidade Federal de Ouro Preto
MSc. Anny Verly — Convidada — Universidade Federal de Ouro Preto
Dr. Rodrigo Augusto Ricco — Convidado — Universidade Federal de Ouro Preto

Márcio Feliciano Braga, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 25/02/2024

Documento assinado eletronicamente por **Marcio Feliciano Braga**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 25/06/2024, às 20:54, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0726286** e o código CRC **9E2EAB3B**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0818 - www.ufop.br

Dedico este trabalho a Deus e minha família.

Agradecimentos

Em primeiro lugar, gostaria de expressar minha sincera gratidão a Deus. Sua orientação e força foram constantes ao longo deste esforço. A fé que tenho em Ti, mesmo nos meus momentos de fraqueza, tem sido uma fonte de inspiração e motivação para perseverar mesmo diante de desafios.

A minha família maravilhosa, seu apoio e carinho são a base de minhas realizações. Minha determinação de me sair bem neste curso foi fortalecida por seus sacrifícios, paciência e fé em meus talentos. Sou muito grato por ter vocês ao meu lado, pois seu apoio e presença fizeram uma diferença grande.

Agradeço a minha incrível esposa Danieli, por sua amizade, companheirismo e compreensão constante, que foi a minha força por trás de minhas conquistas. Com a sua presença, minha jornada se encheu de alegria e propósito.

Aos meus amigos, obrigado por me apoiarem nos bons e maus momentos. Suas risadas e momentos compartilhados forneceram o equilíbrio necessário para minhas atividades acadêmicas. O apoio de vocês tem sido um lembrete de que o sucesso é mais doce quando comemorado com os entes queridos.

Por último, sou profundamente grato aos meus dedicados orientadores. Suas experiências moldaram o meu aprendizado. A paciência e dedicação para transmitir conhecimento foram fundamentais para o meu crescimento e sinto-me honrado por ter a oportunidade de aprender com vocês, muito obrigado.

"Tudo que existe é obra da graça de Deus."
Paulo Junior

Resumo

Os manipuladores robóticos são braços mecânicos adaptáveis compostos por juntas e elos interconectados, feitos para se mover como mãos humanas e realizar trabalhos com precisão. Eles são usados em linhas de montagem, procedimentos cirúrgicos e até missões de exploração espacial. Este trabalho apresenta uma exploração do manipulador ARM-7220-4, disposto de 5 juntas rotativas. O objetivo principal está na simulação do manipulador, por meio do *software* Matlab®, utilizando a cinemática direta, cuja descrição é realizada por meio dos parâmetros de Denavit-Hartenberg. Por conseguinte, se utilizou o *software* SOLIDWORKS®, para a realização do modelo do manipulador, junto ao *software* Matlab®, com finalidade de efetuar uma simulação realista. Os testes indicaram que o simulador virtual apresentou resultados satisfatórios com o objetivo principal, isto é, emular o comportamento do manipulador ARM-7220-4 de modo fidedigno.

Palavras-chave: Manipulador Robótico, ARM-7220-4, Cinemática Direta, Matlab.

Abstract

Robotic manipulators are adaptable mechanical arms made up of interconnected joints and links, made to move like human hands and perform work with precision. They are used in assembly lines, surgical procedures and even space exploration missions. Therefore, this work presents a comprehensive exploration of the ARM-7220-4 manipulator, arranged with 5 rotating joints. The main objective is to simulate the manipulator, using Matlab® software, using direct kinematics, whose description is carried out using Denavit-Hartenberg parameters. Therefore, SOLIDWORKS® software was used to create the manipulator model, together with Matlab® software, in order to carry out a more visually real simulation. The tests indicated that the virtual simulator presented results compatible with the main objective, that is, to reliably emulate the behavior of the ARM-7220-4 manipulator.

Keywords: Robotic Manipulator, ARM-7220-4, Direct Kinematics, Matlab, SOLIDWORKS.

Lista de ilustrações

Lista de tabelas

Lista de siglas e abreviaturas

DH	Denavit–Hartenberg
PPP	manipulador cartesiano
RPP	manipulador cilíndrico
RRP	manipulador esférico
RRR	manipulador articulado

Lista de símbolos

θ_i	ângulo da junta
α_i	ângulo do elo
c_i	$cos(\theta_i)$
d_i	deslocamento do elo
$\hat{K}$	eixo de rotação
Δx	orientação e posição final do manipulador em relação à base na coordenada x
Δy	orientação e posição final do manipulador em relação à base na coordenada y
Δz	orientação e posição final do manipulador em relação à base na coordenada z
q_x	posição do efetuador final ao longo do eixo x
q_y	posição do efetuador final ao longo do eixo y
q_z	posição do efetuador final ao longo do eixo z
A_i	matriz de transformação homogênea do elo i
T_0^n	matriz de transformação do elo n em relação a base
T	matriz de transformação homogênea generalizada
$D_Q(q)$	matriz de translação
$R_K(\theta)$	operador rotacional
s_i	$sen(\theta_i)$
a_i	tamanho do elo
$\vec{Q}$	vetor de translação

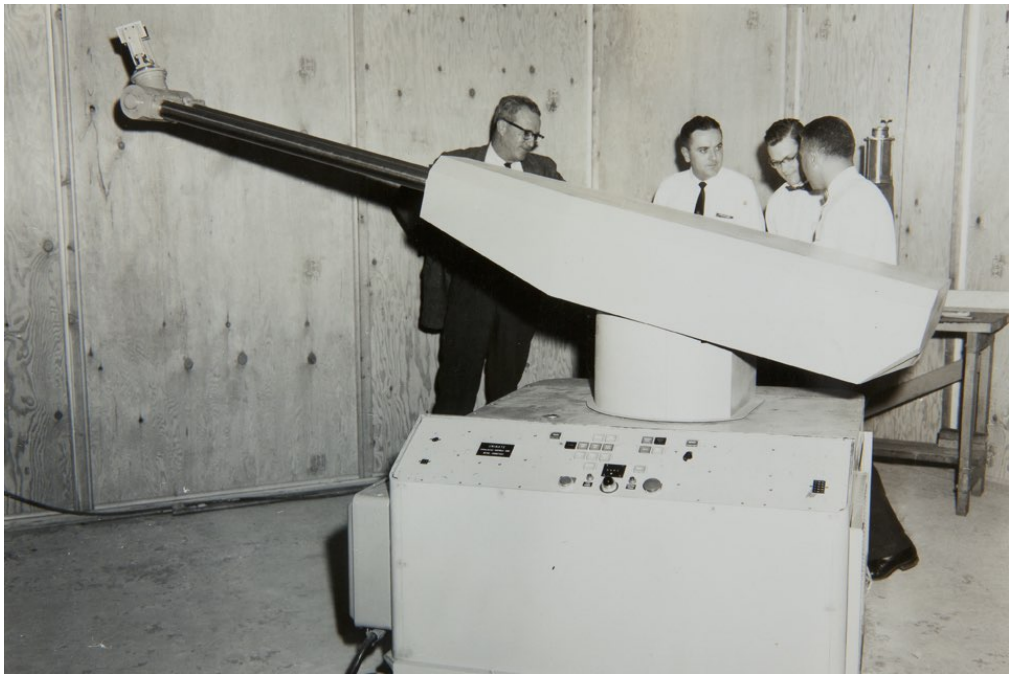
Sumário

1 Introdução

Com sua peça *Robôs universais de Rossum*, o dramaturgo tcheco Karel Čapek popularizou o termo "robô" em 1921. *Rabota*, que significa "trabalho obrigatório", e *robotnik*, que significa "servo", são combinados para formar o termo (MATARIC, 2014), o qual é usado para caracterizar computadores automatizados programados que podem realizar atividades sem envolvimento humano (CAMPOS, 2018).

O inovador robô programável conhecido como "*Unimate*" foi desenvolvido por Joseph Engelberger e George Devol em 1954, marcando uma virada significativa no desenvolvimento da robótica industrial. A ideia inovadora de um "robô programável" foi desenvolvida por George Devol no mesmo ano, e Joseph Engelberger foi uma pessoa essencial em sua promoção e comercialização (SILVA, 2010). O *Unimate* era um braço robótico com propulsão hidráulica que mostrava talentos excepcionais ao manusear materiais e soldar ao longo de uma linha de produção. A Figura 1 mostra um exemplo do *Unimate* em um formato visual. Quando o robô foi colocado em uma fábrica de veículos da Ford em 1961, ele teve sua primeira aplicação em ambiente industrial.

Figura 1 – Braço Robótico — *Unimate*.



Fonte: Kawasaki Robotics.

Destaca-se ainda que a demanda por robôs industriais vem se expandindo gradualmente desde 2010, à medida que as empresas usam cada vez mais a automação. Nos dias atuais, mesmo os processos de baixo volume podem ser efetivamente automatizados, graças aos avanços realizados em robôs industriais ao longo do tempo. Isso, porque a

utilização em processos envolvendo por exemplo, soldagem, corte, linha de montagem, embalagem, paletização, etc., aumenta o lucro das empresas (ROBOTS, 2019).

As ferramentas mecânicas gerenciadas por *softwares*, com funções especializadas para uma variedade de procedimentos automatizados são conhecidas como manipuladores robóticos. Adicionalmente, os sensores podem ser utilizados por manipuladores robóticos para auxiliar no posicionamento e movimentação de seus componentes em diversos cenários previamente determinados (LOPES, 2002).

Existem múltiplas tecnologias disponíveis para o ensino de robótica empregando simuladores, atendendo a uma variedade de níveis de habilidade e objetivos educacionais. Algumas opções de simuladores são, por exemplo, Gazebo (KOENIG; HOWARD, 2004), Webots (WEBOTS, 1998) e CoppeliaSim (ROHMER; SINGH; FREESE, 2013). O preço desses programas varia, alguns oferecem versões gratuitas para fins educacionais e outros exigem licenças para determinados recursos ou uso comercial. Além disso, a maioria desses simuladores inclui modificação por meio de códigos, permitindo que os educadores personalizem o ambiente de simulação de acordo com suas próprias metas e objetivos educacionais.

O objetivo principal deste trabalho envolve desenvolver um simulador virtual, projetado em ambiente Matlab®, para o braço robótico ARM-7220-4, presente no Laboratório de Controle e Automação da UFOP. O objetivo do manipulador é auxiliar no ensino de conceitos de robótica de manipuladores. Em adição, objetiva-se realizar a integração do Matlab® e SOLIDWORKS®, com intuito de aproveitar as vantagens de projetar a construção física do manipulador e a validação de algoritmos de controle, planejamento de trajetória e simulação do comportamento do manipulador em diferentes cenários.

O objetivo final da simulação de um manipulador robótico com o modelo cinemático é prever e compreender com precisão o movimento e o comportamento do robô dentro de seu ambiente operacional. Além disso, por utilizar representações matemáticas da geometria, articulações e restrições do robô, permite-se realizar simulações precisas dos movimentos do manipulador, desde tarefas simples como alcançar e agarrar até trajetórias complexas em três dimensões. Por fim, simular um manipulador robótico com o modelo cinemático permite ultrapassar os limites da automação, revelando novos níveis de eficiência, variedade e invenção na busca de um futuro mais inteligente e conectado.

1.1 Justificativa

A relevância de estudar e desenvolver manipuladores robóticos se dá pelo fato de que eles podem aumentar significativamente a eficiência e a produtividade em processos industriais, reduzindo custos e melhorando a qualidade dos produtos. Além disso, os robôs podem realizar tarefas que seriam perigosas ou impraticáveis para seres humanos. Destaca-se ainda, a necessidade de desenvolver ferramentas de controles acurados para

manipuladores robóticos com o intuito de evitar acidades graves. Um caso marcante da falta de segurança ocorreu em 2015 na Alemanha, em que um robô industrial esmagou um homem, que acabou vindo a óbito (WELT, 2015).

Ao realizar trabalhos sobre manipuladores robóticos, é possível explorar novas tecnologias e soluções para aprimorar a precisão, a velocidade e a flexibilidade desses sistemas, bem como melhorar sua interação com humanos. Também, é possível investigar questões relacionadas à ética e responsabilidade social na utilização de robôs em diversas áreas.

Diante dessa demanda, é imprescindível que estudantes e profissionais da área tenham acesso a manipuladores robóticos. No entanto, devido a diversos empecilhos, como o alto custo de aquisição, riscos de acidentes e manutenção de braços robóticos, o desenvolvimento de um simulador virtual torna-se importante, pois permitirá aos interessados aprofundar seu conhecimento na área e realizar testes consecutivos e em menor tempo.

1.2 Objetivos

O objetivo principal deste trabalho é desenvolver um simulador virtual do manipulador robótico ARM-7220-4 composto por 5 juntas de revolução. Além do mais, pretende-se incluir, no simulador, visualizadores gráficos necessários para modelagem, controle e representação do movimento do robô manipulador.

1.2.1 Objetivos Específicos

- Empregar métodos clássicos da literatura (SPONG; HUTCHINSON; VIDYASAGAR, 2020);
- Obtenção da modelagem cinemática do manipulador;
- Simular a trajetória do manipulador robótico tridimensional;
- Estudo das ferramentas a serem utilizadas para a simulação, sendo o Matlab® e sua junção ao SOLIDWORKS®.

1.3 Estrutura do Trabalho

Os cinco capítulos, que compõem a estrutura deste trabalho, contêm os seguintes assuntos. No Capítulo 1, é feito um breve levantamento histórico sobre a robótica e sua crescente utilização no meio industrial. No Capítulo 2, apresentam-se conceitos fundamentais da robótica, como a descrição da classificação dos manipuladores. Também, são relatados os métodos de posição e orientação (SPONG; HUTCHINSON; VIDYASAGAR, 2020). No Capítulo 3, é apresentada a estrutura e a definição da modelagem do

ARM-7220-4 em relação à cinemática. No Capítulo 4, apresentam-se os resultados obtidos, por meio de simulações, via cinemática direta. Por fim, no Capítulo 5, apresentam-se as considerações finais e as propostas de continuação deste trabalho.

2 Revisão Bibliográfica

2.1 Introdução

Neste capítulo, apresenta-se uma sucinta aplicação dos manipuladores robóticos. Também serão detalhados os aspectos de classificação, elementos, tipos de juntas e a descrição do seu movimento. Na Figura 2, pode-se verificar um exemplo de um manipulador robótico, na qual alguns dos conceitos citados são indicados.

Figura 2 – Robô KUKA 500 FORTEC.

Fonte: Spong, Hutchinson e Vidyasagar (2020).

2.2 Conceitos

Manipulação é uma habilidade avançada que separa os robôs de outros sistemas computadorizados e automatizados. A capacidade robótica de interagir fisicamente e modificar ambientes, isto é, manipular, abre margem para várias aplicações. Robôs não podem sentir e usar o sentido de toque como seres humanos. Por isso, sensores de força/torque são necessários para compensar essa falta da habilidade motora (BRASIL, 2022).

Algumas aplicações típicas dos robôs industriais são: fundição, pintura, soldagem, montagem, movimentação de cargas, inspeção de produtos, reconhecimento de imagens e realização de testes (BRASIL, 2022). Com a migração para a indústria 4.0, a ideia é que a robótica esteja cada vez mais presente na linha de montagem e fora dela (GAMERO, 2018).

De acordo com (ABREU, 2002), uma das áreas que se alarga à utilização de robôs diz respeito a operações de inspeção e teste. Trata-se de operações de controle de qualidade que envolvem a verificação de componentes, subprodutos ou produtos analisando a sua

conformidade com determinados critérios previamente fixados. Nas modernas linhas de montagem de automóveis, os manipuladores robóticos desempenham um papel crucial em diversas tarefas, como soldagem, pintura e montagem. Por exemplo, braços robóticos equipados com ferramentas de soldagem são usados para unir com precisão componentes metálicos, garantindo integridade estrutural e consistência de qualidade no produto final.

2.3 Classificação dos Manipuladores

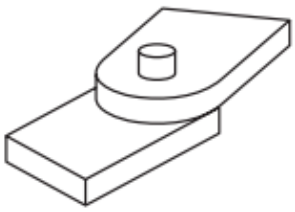
Os manipuladores robóticos são dispostos em uma estrutura que permite o movimento controlado de seus braços mecânicos, articulados em juntas para realizar tarefas específicas. A disposição dos manipuladores altera conforme a aplicação desejada e pode incluir diferentes configurações de braços, juntas, garras, sensores e dispositivos de controle.

2.3.1 Especificação das Juntas

Existem vários tipos de juntas em um manipulador robótico, incluindo:

- **Rotacional ou Revoluta:** Permite que o braço robótico gire em torno de um eixo, como uma articulação do cotovelo ou do punho. A geometria da junta rotacional pode ser verificada na Figura 3.

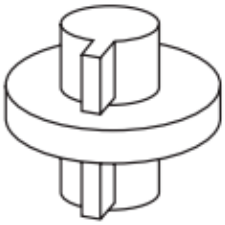
Figura 3 – Junta rotacional.



Fonte: Craig (2012).

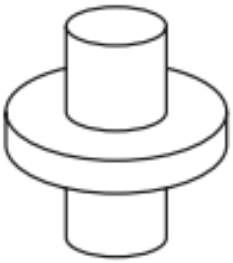
- **Prismática ou Linear:** Permite que o braço do robô se mova linearmente em uma direção, como um elevador ou uma gaveta. A geometria da junta prismática é ilustrada na Figura 4.
- **Cilíndrica:** Permite que o braço do robô se mova em uma direção linear combinada com uma rotação em torno de um eixo, como um pistão. Na Figura 5, indica-se a geometria da junta cilíndrica.
- **Esférica:** Permite que o braço do robô se mova em todas as direções, como uma junta do ombro. A geometria da junta esférica é assinalada na Figura 6.

Figura 4 – Junta prismática.



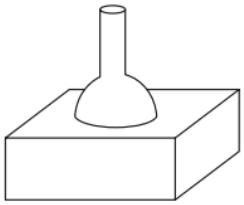
Fonte: Craig (2012).

Figura 5 – Junta cilíndrica.



Fonte: Craig (2012).

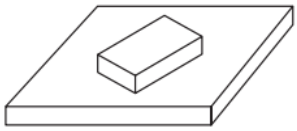
Figura 6 – Junta esférica.



Fonte: Craig (2012).

- **Planar:** Permite que o braço do robô se mova em um plano, como uma junta de movimento lateral e vertical em um joystick. A geometria da junta planar pode ser verificada na Figura 7.

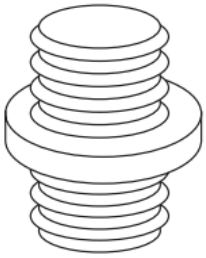
Figura 7 – Junta planar.



Fonte: Craig (2012).

- **Parafuso:** Composta por um parafuso e uma porca, onde o parafuso é girado para avançar ou recuar a porca em uma direção linear ao longo do eixo do é assinalada na Figura 8.

Figura 8 – Junta parafuso.



Fonte: Craig (2012).

2.3.2 Estruturas

Com as juntas referenciadas anteriormente, pode-se citar as classes mais comuns dos robôs a seguir.

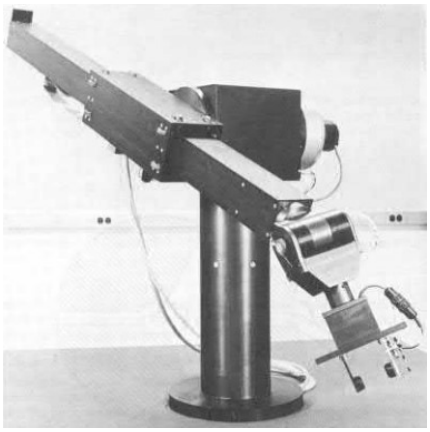
- **Articulado (RRR)**
  O manipulador articulado (RRR) é um tipo de robô industrial que possui três juntas rotativas (ou articulações) que permitem a movimentação em três eixos: X, Y e Z. O robô é chamado de "RRR" porque cada junta é do mesmo tipo, rotacional. A Figura 9 ilustra um exemplo de robô articulado.

Figura 9 – Robô Kuka 500.

Fonte: Industry (2023).

- **Esférico (RRP)**
  O manipulador esférico (RRP) é um tipo de robô industrial que possui três juntas, sendo duas juntas rotativas (R) e uma junta prismática (P). As juntas rotativas permitem a movimentação em torno do eixo de rotação, enquanto a junta prismática permite a movimentação linear em um dos eixos. Esse tipo de manipulador é chamado de "RRP" porque a junta prismática é flanqueada por duas juntas rotativas. Um exemplo de robô esférico é mostrado na Figura 10.

Figura 10 – Robô Stanford Arm.



Fonte: Spong, Hutchinson e Vidyasagar (2020).

- **SCARA (RRP)**
  O manipulador SCARA (do inglês, *Selective Compliance Assembly Robot Arm*) é um tipo de robô industrial que possui dois braços articulados, sendo um braço vertical e um braço horizontal. A configuração articulada do SCARA permite que ele se mova em dois eixos cartesianos, X e Y, e também rotacione em torno do eixo vertical, Z. Na Figura 11, apresenta-se um exemplo de robô SCARA.

Figura 11 – Robô Omron.

Fonte: Industry (2023).

- **Cilíndrico (RPP)**
  Um manipulador cilíndrico (RPP) é um tipo de robô industrial composto por uma estrutura cilíndrica com três graus de liberdade. A sigla "RPP" refere-se à configuração dos seus três elos, que são Rotativo-Prismático-Prismático, como exemplificação na Figura 12.

- **Cartesiano (PPP)**
  Um manipulador cartesiano (PPP) é um tipo de robô industrial que consiste em

Figura 12 – Robô Omron.

Fonte: Industry (2023).

uma estrutura cartesiana com três graus de liberdade, em que cada elo é prismático. A sigla "PPP" refere-se à configuração dos seus três elos, sendo todos prismáticos. Cada elo é orientado em um eixo diferente, permitindo movimentos precisos em três direções cartesianas: X, Y e Z. A Figura 13 ilustra um exemplo de robô cartesiano.

Figura 13 – Robô Wittmann Battenfeld.

Fonte: Industry (2023).

- **Paralelo**
  Um robô industrial conhecido como manipulador paralelo move seu efetuador final usando um sistema de ligação paralela de vários braços. Os robôs paralelos apresentam uma série de elos conectados em paralelo ao efetuador do robô, em oposição aos robôs convencionais, que empregam uma estrutura serial de elos conectados um após o outro. Eles podem ser organizados de várias maneiras, incluindo hexápodes, polvos e outros arranjos que usam um número par ou ímpar de elos. Uma ilustração de um robô paralelo pode ser vista na Figura 14.

2.3.3 Efetuadores

O componente do manipulador robótico que está em contato direto com o mundo exterior e é responsável por realizar uma tarefa desejada para o robô é o *end-effector*,

Figura 14 – Robô Panasonic.

Fonte: Industry (2023).

também conhecido como efetuador final. Um instrumento, como uma pinça, uma furadeira, uma pistola de pulverização ou um sensor como uma câmera ou um medidor de distância, podem ser usados para isso. O tipo de seletor usado é determinado pelo tipo de trabalho que o robô deve realizar. Por exemplo, um robô de linha de produção pode ter um efetuador construído para pegar itens e colocá-los em uma caixa.

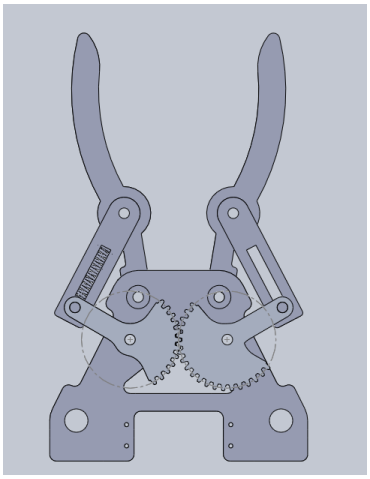
O efetuador é conectado ao resto do manipulador por meio de uma interface mecânica, geralmente, na extremidade do braço do robô. Em resumo, o efetuador é a parte do manipulador robótico em contato direto com o ambiente externo e é responsável por realizar a tarefa desejada pelo robô. Pode ser projetado para ser intercambiável para permitir a reconfiguração do robô para diferentes tarefas. A seguir, pode-se verificar alguns tipos de efetuadores.

- **Mecânico**
  A pinça mecânica é uma das variedades mais fundamentais de efetuadores. Agem como dedos mecânicos para agarrar. Um controlador do robô ativa as mandíbulas para fazer os movimentos necessários para agarrar o objeto (MAHENDRA, 2023). Na Figura 15 é possível verificar o um modelo de um efetuador mecânico.

- **Magnético**
  As menores peças de trabalho podem ser manuseadas usando uma pinça magnética. Para capturar objetos de metal, as pinças magnéticas empregam uma superfície magnética. Esse tipo de pinça geralmente não possui dedos e mandíbulas e, em vez disso, depende de superfícies magnéticas planas para segurar objetos. Nas indústrias onde chapas metálicas e peças de automóveis são transportadas ao longo de uma linha de montagem, pinças magnéticas são usadas com frequência (ROBOTS, 2021). Na Figura 16 é possível verificar o efetuador magnético.

- **Vácuo**
  Quando há um diferencial de pressão suficiente entre a atmosfera e o vácuo, ou

Figura 15 – Efetuador Mecânico.



Fonte: Serrato-Sosa (2016).

Figura 16 – Efetuador Magnético.

Fonte: Robots (2021).

pressão negativa, as pinças de vácuo podem levantar, segurar, mover e realizar outras atividades. Isso só é possível se o lado plano e grande de uma peça permitir que uma pinça a vácuo crie uma diferença de pressão suficientemente significativa. Como possuem grandes laterais planas, essas peças são mais adequadas para o uso de pinças a vácuo. As pinças a vácuo podem não ser apropriadas para manusear peças muito pesadas, pois seria necessário gerar uma grande quantidade de pressão negativa (AUTOMATION, 2019). Na Figura 17, ilustra-se o emprego de um efetuador a vácuo.

- **Pneumático**
  Nas linhas de produção, as garras pneumáticas são efetuadores finais robóticos cruciais para o manuseio de materiais. Ao rejeitar itens de um transportador por uma estação de controle de qualidade, essas garras seguram, direcionam e posicionam pe-

Figura 17 – Efetuador a vácuo.

Fonte: AUTOMATION (2019).

ças e outros objetos para processamento, montagem com outras peças ou rejeição. Eles são frequentemente utilizados em ambientes industriais, particularmente em máquinas de trabalho automatizadas, linhas de produção e montagem, máquinas de tendência ligadas à fabricação controlada, partes periféricas da fábrica e logística, bem como em operações militares automatizadas (UKO, 2022). Um exemplo de efetuador pneumático é indicado na Figura 18.

Figura 18 – Efetuador pneumático.

Fonte: Uko (2022).

- **Elétrico**
  A geração mais recente de pinças elétricas permite obter um maior retorno de dados do que as pinças pneumáticas, e eles podem igualar ou até superar seus predecessores movidos a ar em termos de força de preensão e velocidade. É possível ter um controle consideravelmente maior sobre as forças de preensão, velocidades, etc. Como ele vem com sensores de posição integrados, sensores eextras, magnéticos ou de proximidade, seus respectivos conjuntos de cabos e blocos de distribuição não são necessários,

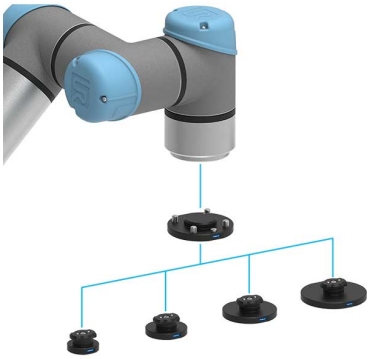
nem são uma despesa adicional (TOMORROW, 2018). Na Figura 19, mostra-se um exemplo de um efetuador elétrico.

Figura 19 – Efetuador elétrico.

Fonte: Uko (2022).

- **Adesiva**
  Sem a necessidade de uma fonte de energia externa, a peça de trabalho pode ser agarrada e segura. Proporciona ao usuário dois benefícios diferentes: menos requisitos de instalação e comissionamento, e menores custos de energia devido à operação da pinça adesiva não requer ar comprimido, vácuo ou corrente (ROBOTS, 2023). Um exemplo de efetuador adesivo é apresentado na Figura 20.

Figura 20 – Efetuador adesivo.



Fonte: Robots (2023).

2.3.4 Graus de Liberdade

O número de graus de liberdade (GDL, ou DoF, em inglês, *Degree of Freedom*) que um manipulador possui é o número de variáveis de posição independentes que teriam de ser especificadas para se localizarem todas as peças do mecanismo. Esse é um termo geral usado para qualquer mecanismo. Por exemplo, uma concatenação de quatro barras

tem apenas um grau de liberdade (embora haja três membros móveis). No caso dos robôs industriais típicos, como o manipulador é quase sempre uma cadeia cinemática aberta e, como a posição de cada junta costuma ser definida com uma única variável, o número de juntas é igual ao de graus de liberdade (CRAIG, 2012).

2.3.5 Espaço de Trabalho

O espaço de trabalho de um manipulador é todo o volume variado enquanto o manipulador executa todos os movimentos possíveis. O espaço de trabalho é limitado pela geometria do manipulador, bem como por restrições mecânicas nas juntas. Por exemplo, um manipulador de revolução, pode ser restrito a nada menos que 360° de movimento total. Em geral, o espaço de trabalho é dividido em um espaço de trabalho acessível e um espaço de trabalho móvel. O conjunto total de pontos atingíveis pelo manipulador constitui a área de trabalho, enquanto os pontos atingíveis são os que o manipulador pode alcançar com o auxílio da direção do efetuador final (SPONG; HUTCHINSON; VIDYASAGAR, 2020).

2.4 Transformações

De acordo com Craig (2012), as transformações são um conceito fundamental na robótica e na automação industrial. Elas são usadas para descrever a posição e orientação de um objeto no espaço tridimensional. As transformações são uma forma de descrever como um braço robótico é posicionado e orientado em relação a um sistema de coordenadas global, quando se trata de manipuladores robóticos. Na robótica, muitas técnicas transformacionais, como transformações homogêneas, rotacionais e translacionais, são empregadas.

2.4.1 Translação

O deslocamento de um sistema ou objeto em três dimensões é representado por uma transformação de translação. As coordenadas x, y e z da translação são representadas pelos três primeiros componentes da última coluna de uma matriz 4×4. A matriz de translação é dada por

$$D_Q(q) = \begin{bmatrix} 1 & 0 & 0 & q_x \\ 0 & 1 & 0 & q_y \\ 0 & 0 & 1 & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.1}$$

sendo q_x, q_y, q_z componentes do vetor de translação $\vec{Q} \in \mathbb{R}$.

2.4.2 Rotação

A orientação de um sistema ou objeto em três dimensões é representada por uma transformação de rotação. Os quatro eixos de rotação são representados pelos elementos das três primeiras colunas de uma matriz 4×4. O operador rotacional $R_{\hat{K}}(\theta)$ realiza uma rotação de ângulo θ em torno do eixo de direção $\hat{K}$.

Para uma matriz que rotaciona em relação ao eixo z, por exemplo, segue a seguinte estrutura matricial

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\text{sen}\,\theta & 0 & 0 \\ \text{sen}\,\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

sendo $R_z(\theta) \in \mathbb{R}$.

2.4.3 Transformação Homogênea

A transformação homogênea é usada para caracterizar a localização e a orientação da ferramenta em relação à base do robô em um manipulador robótico. A orientação e a localização do sistema de coordenadas do efetuador final em referência ao sistema de coordenadas de base são descritas pela matriz de rotação e matriz de translação que compõem a transformação.

A questão fundamental da união dessas duas matrizes é que elas possuem dimensões diferentes. Para resolver esse problema, uma transformação de dimensão é utilizada, na qual a matriz de rotação recebe mais alguns componentes, convertendo-a em uma matriz quadrada 4×4, sem alterar como os outros coeficientes são calculados.

Uma matriz de transformação homogênea generalizada é dada por

$$T = \begin{bmatrix} \overbrace{\begin{matrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{22} & r_{33} \end{matrix}}^{\text{Rotação}} & \overbrace{\begin{matrix} \Delta x \\ \Delta y \\ \Delta z \end{matrix}}^{\text{Translação}} \\ \underbrace{\begin{matrix} 0 & 0 & 0 \end{matrix}}_{\text{Perspectiva}} & \underbrace{\begin{matrix} 1 \end{matrix}}_{\text{Fator de escala}} \end{bmatrix}, \tag{2.3}$$

sendo $T \in \mathbb{R}$, na qual se indica cada um de seus componentes. Juntamente com os componentes de translação e rotação, (2.3) também mostra os componentes de perspectiva e fator de escala, que em robótica são assumidos como tendo valores de $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ e $\begin{bmatrix} 1 \end{bmatrix}$, respectivamente.

2.5 Cinemática

A cinemática dos manipuladores robóticos é o estudo do movimento dos robôs manipuladores, que envolve a determinação das posições, velocidades e acelerações dos elos e do efetuador final. A cinemática aborda dois problemas: a cinemática direta e inversa (SPONG; HUTCHINSON; VIDYASAGAR, 2020). Neste trabalho, o enfoque principal está no estudo da cinemática direta.

2.5.1 Cinemática Direta

A cinemática direta é uma técnica utilizada na robótica para determinar a posição e orientação do efetuador final do manipulador com base nas posições dos elos do robô. Em outras palavras, ela permite calcular a trajetória do efetuador final do robô, dadas as posições de cada um dos seus elos.

A convenção de Denavit–Hartenberg (DH) é adotada, pois ela e a cinemática direta estão intimamente relacionadas, porque os parâmetros são usados para construir as matrizes de transformação necessárias para cálculos de cinemática direta. Essa convenção fornece uma maneira sistemática de atribuir quadros de coordenadas aos elos do manipulador e derivar as matrizes de transformação que representam o movimento relativo entre elos adjacentes (CRAIG, 2012).

2.5.1.1 Convenção de *Denavit-Hartenberg*

Um conjunto de regras é fornecido pela convenção de DH para especificar a posição relativa e a orientação entre os elos subsequentes. Torna-se necessário desenvolver um método geral sistemático para definir a posição relativa e a orientação de dois elos consecutivos. Pode-se identificar os dois sistemas de coordenadas conectados aos dois elos e calcular as transformações de coordenadas entre eles (SICILIANO, 2009).

Desse modo, é realizado um conjunto de regras fornecido por DH para especificar a posição relativa e a orientação entre os elos subsequentes, vista na Figura 21 para melhor compreensão.

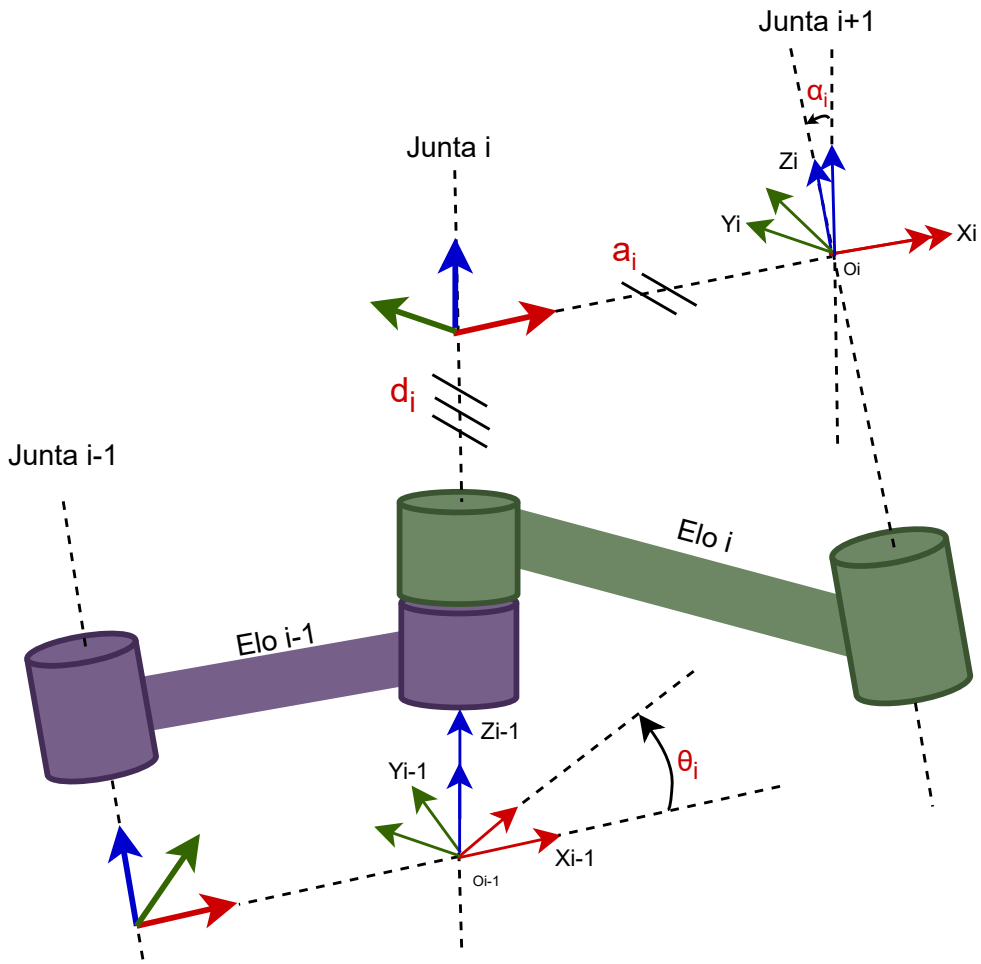
Conforme Spong, Hutchinson e Vidyasagar (2020), as regras são:

Passo 1: Localizar os eixos das articulações z_0, z_1, até z_{i-1}.

Passo 2: Estabelecer o sistema de coordenadas da base, definindo a origem em qualquer lugar no eixo z_0. Os eixo x_0 e y_0 são escolhidos arbitrariamente.

Passo 3: Encontrar o ponto O_i, sendo a origem do sistema i, onde a normal comum dos eixos z_i e z_{i-1} intercepta o eixo z_i. Encontre o ponto O_i na interseção se os eixos z_i e z_{i-1} se cruzarem. Encontre o ponto O_i na junta i se os eixos z_i e z_{i-1} forem paralelos.

Figura 21 – Parâmetros de DH.



Fonte: Imagem adaptada de Spong, Hutchinson e Vidyasagar (2020).

Passo 4: Estabelecer o eixo x_i ao longo da normal comum dos eixos z_{i-1} e z_i, começando no ponto O_i. O eixo x_i é normal para ambos em qualquer direção devido a como os eixos z_i e z_{i-1} se cruzam.

Passo 5: Estabelecer o eixo y_i para completar o sistema de coordenadas, segundo a regra da mão direita, disposta em Spong, Hutchinson e Vidyasagar (2020).

Desse modo, é necessário repetir os passos 3 a 5 para $i = 1,...,n-1$.

Passo 6: Estabelecer o sistema de coordenadas do efetuador, que é $O_n - x_n - y_n - z_n$. Embora a origem desse sistema seja selecionada arbitrariamente, ela é normalmente escolhida como o centro da garra ou outro ponto do efetuador. Desde que o eixo x_n seja perpendicular ao eixo x_{n-1}, os eixos do sistema podem ser definidos de qualquer forma.

Passo 7: Fazer uma tabela referenciando cada um dos ligamentos ou articulações com as características do DH.

- a_i: distância ao longo de x_i da interseção dos eixos x_i e z_{i-1} até o_i.
- d_i: coordenada de $O_{i'}$ ao longo de z_{i-1}. Se a junta i é prismática, d_i é variável.
- α_i: ângulo entre os eixos z_{i-1} e z_i sobre o eixo x_i.
- θ_i: ângulo entre os eixos x_{i-1} e x_i em torno do eixo z_{i-1}. Se a junta i é de revolução, θ_i é variável.

Passo 8: Criar as matrizes de transformação homogêneas, usando os parâmetros de DH, dada por

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}, \tag{2.4}$$

em que Rot_{z,θ_i}, $Trans_{z,d_i}$, $Trans_{x,a_i}$ e Rot_{x,α_i} representam, respectivamente, as matrizes de rotação em z, matriz de translação em z, matriz de translação em x e, por fim, matriz de rotação em x. Reescrevendo (2.4), tem-se

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.5}$$

ou ainda,

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.6}$$

na qual c_{θ_i}, s_{θ_i}, c_{α_i}, s_{α_i} representa $cos(\theta_i)$, $sen(\theta_i)$, $cos(\alpha_i)$ e $sen(\alpha_i)$, respectivamente. Note ainda que a divisão indicada na matriz (2.3) em relação à rotação, translação, perspectiva e fator de escala pode ser aplicada à matriz (2.6).

Passo 9: Obter a matriz de transformação homogênea $T_0^n = A_1 \cdots A_n$ que relaciona a posição e orientação do efetuador em relação ao sistema da base.

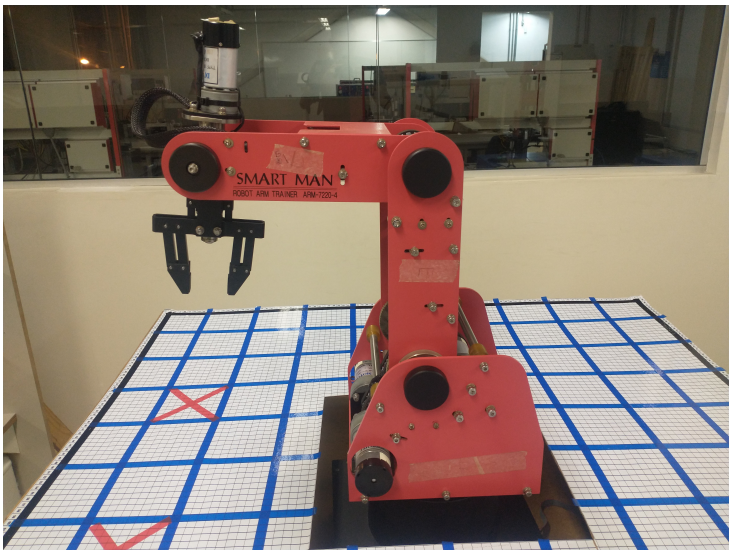
3 Descrição do manipulador ARM-7220-4

Neste capítulo serão apresentados os aspectos construtivos do manipulador robótico RRR, ou um braço com 6 graus de liberdade e articulações giratórias. Adicionalmente, é fornecida a modelagem cinemática, bem como os materiais compostos no manipulador e suas respectivas finalidades.

3.1 Introdução

O Smart Man Robot Arm Trainer ARM-7220-4 é um manipulador robótico fabricado pela *Eshed Robotec*, empresa especializada em soluções avançadas de automação. Trata-se de um braço robótico versátil e possui várias aplicações industriais, incluindo tarefas de montagem, *pick-and-place* e manuseio de materiais. O manipulador presente no Laboratório de Controle e Automação, da Universidade Federal de Ouro Preto, pode ser visto na Figura 22.

Figura 22 – Braço Robótico ARM-7220-4.



Fonte: Do autor.

3.2 Aspectos físicos

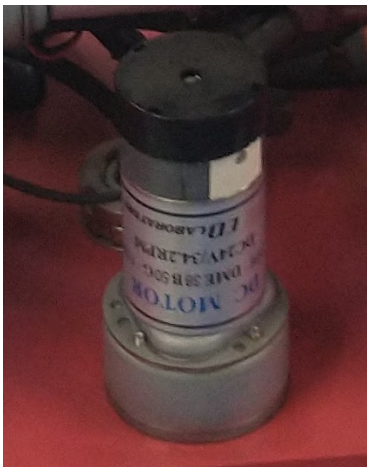
3.2.1 Base

O manipulador robótico começa com uma base robusta que fornece estabilidade e serve como base para todo o sistema. A base é geralmente fixada no chão ou em uma estrutura de suporte.

3.2.2 Motor

Os motores DC são utilizados nas articulações do braço robótico educacional SMART MAN. Pode-se visualizar o modelo do motor na Figura 23.

Figura 23 – Motor DC DME38B50G-115.



Fonte: Do autor.

Cada um dos motores possui uma caixa de engrenagens e neles há um transdutor rotativo chamado codificador que converte um movimento angular em uma série de pulsos digitais, referido como um servomotor, quando as ações de velocidade e movimento são transmitidas por um sinal codificado eletronicamente. A Tabela 1 apresenta a especificação dos motores do manipulador.

Tabela 1 – Parâmetros do Motor DC DME38B50G-115.

Tensão nominal	$24V$
Velocidade nominal	$65rpm$
Relação de engrenagens	72
Número de pulsos por rotação (PPR)	48

Os servomotores geralmente consistem em 4 componentes básicos:

1. **Motor de corrente contínua (DC):** Esse gerador de torque confere mobilidade ao servo. O motor gira em sua velocidade máxima quando um potencial é aplicado a ambos os seus terminais. O sentido de rotação é invertido se a tensão ligada aos seus dois terminais também for invertida.

2. **Engrenagens de redução:** Um trem de engrenagens que reduz a alta velocidade de rotação do motor para aumentar sua capacidade de torque.

3. **Sensor de deslocamento:** Na robótica, os codificadores são frequentemente empregados como sensores, embora outras opções incluam potenciômetros, tacômetros

e resolvedores. Os potenciômetros são pouco atraentes e inconvenientes devido à sua natureza analógica, sensibilidade à temperatura, precisão e não linearidade. O tacômetro não é prático devido ao seu tamanho e peso, resolução insuficiente e perda de participação de mercado para a tecnologia óptica usada em codificadores.

4. **Circuito de controle:** Uma técnica de controle de posição baseada em realimentação é realizada por uma placa eletrônica no motor. O circuito compara o sinal de entrada de referência de posição pretendido com a medição de posição atual do codificador para conseguir isso. O motor é movido na direção necessária para diminuir o erro, reduzindo a diferença entre a posição atual e a pretendida.

3.2.3 Microinterruptor (*Micro Switch*)

Cada uma das 5 juntas tem um microinterruptor. Quando o microinterruptor de cada junta é pressionado, o robô assume uma posição conhecida como *home*. Esse é o ponto de referência operacional do robô. Sempre que o sistema é ligado, o robô é direcionado para essa configuração.

3.2.4 Estrutura

O sistema é baseado em 6 graus de liberdade, na Figura 24, pode-se observar as medidas, as quais são sumarizadas na Tabela 2, e seus elementos. Com uma garra conectada, as juntas são rotacionais, e os limites da restrição rotacional mecânica das juntas são descritos na Tabela 3. Lembrando que todos os ângulos são medidos a partir da direção positiva do eixo X.

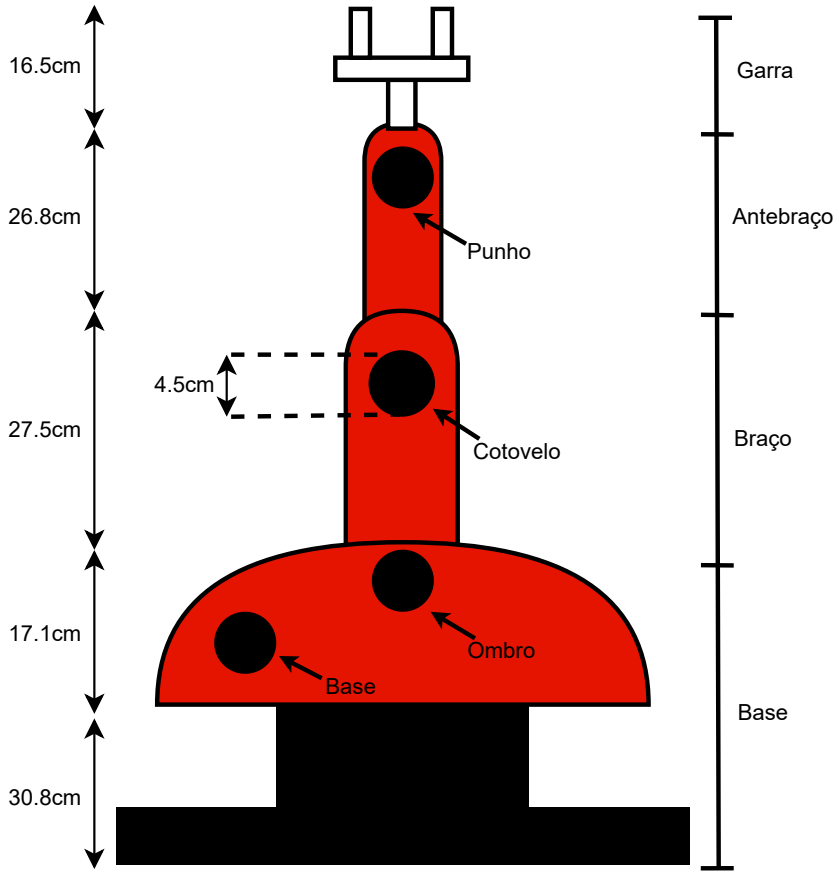
Tabela 2 – Comprimentos dos elos do braço robótico ARM-7220-4.

Parâmetros	**Valores**
L_1	17.1 cm
L_2	27.5 cm
L_3	26.8 cm
L_4	16.5 cm

Tabela 3 – Restrição rotacional do ARM-7220-4.

Parâmetros	**Valores**
Rotação da base	310°
Movimento angular do ombro	+130° / -35°
Movimento angular do cotovelo	±130°
Movimento angular do punho	±130°
Rotação do punho	360°

Figura 24 – Estrutura do manipulador.



Fonte: Do autor.

3.2.5 Espaço de Trabalho

O espaço, ou envelope, de trabalho do robô é determinado pelo comprimento dos elos e o grau de rotação da junta. O tamanho e a faixa de operação do ARM-7220-4 são mostrados na Figura 25. Normalmente, a base do robô é presa a uma superfície de trabalho fixa. No entanto, pode ser conectado a uma base móvel, proporcionando uma faixa de trabalho mais ampla. A altura máxima do robô (medida como a distância entre os degraus da base) é de 1.040 mm e sua distância máxima de guinada é de 610 mm.

3.3 Modelagem do manipulador ARM-7220-4

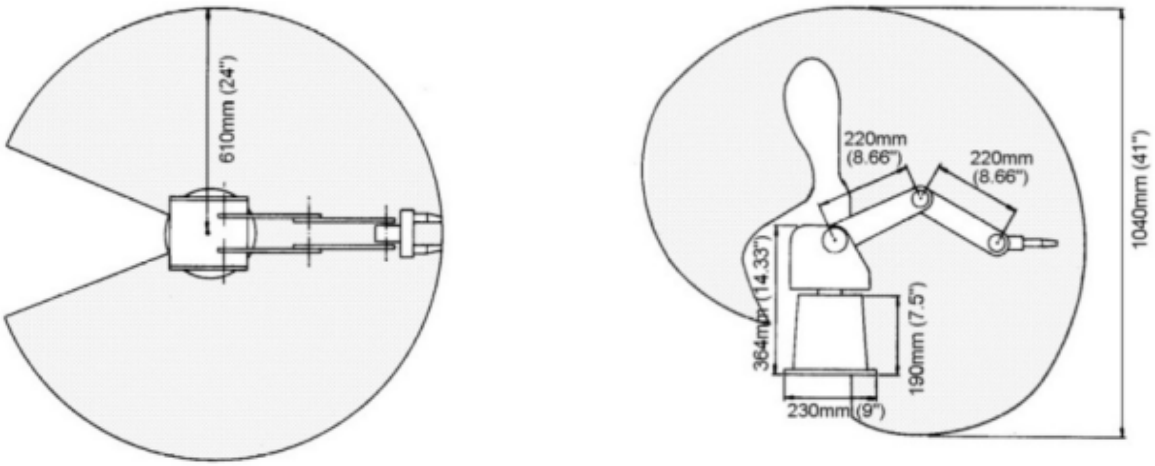
3.3.1 Cinemática Direta

Na Figura 24, é possível verificar os comprimentos dos elos. Como definido na subseção 2.5.1, foi obtido o sistema de coordenadas, demonstrado na Figura 26.

Com os sistemas de coordenadas e os parâmetros de DH, listados na Tabela 4, obtêm-se as matrizes de transformação homogênea por meio de (2.6) para cada junta i,

Figura 25 – Espaço de trabalho do manipulador ARM-7220-4.

(a) Vista superior. (b) Vista lateral.



Fonte: Kumar e Chand (2015)

Tabela 4 – Parâmetros do manipulador.

Junta i	a_i	α_i	d_i	θ_i
1	0	$-90°$	L_1	θ_1
2	L_2	0	0	θ_2
3	L_3	0	0	θ_3
4	0	$-90°$	0	θ_4
5	0	0	L_4	θ_5

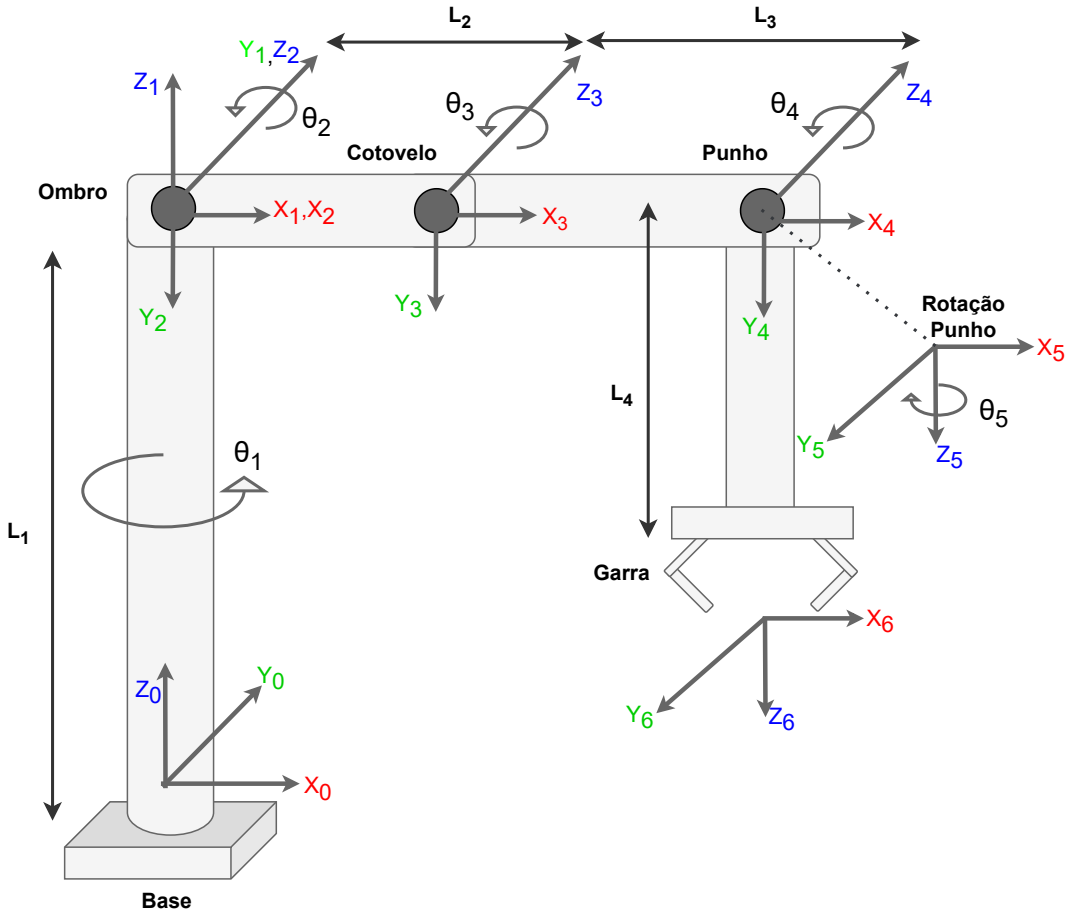
com $i = 1,2,\ldots,5$, as quais são dadas por

$$A_1 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1}cos(-90°) & s_{\theta_1}\,sen(-90°) & 0 \\ s_{\theta_1} & c_{\theta_1}cos(-90°) & -c_{\theta_1}sen(-90°) & 0 \\ 0 & sen(-90°) & cos(-90°) & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{\theta_1} & 0 & -s_{\theta_1} & 0 \\ s_{\theta_1} & 0 & c_{\theta_1} & 0 \\ 0 & -1 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$A_2 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2}cos(0°) & s_{\theta_2}sen(0°) & L_2c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2}cos(0°) & -c_{\theta_2}sen(0°) & L_2s_{\theta_2} \\ 0 & sen(0°) & cos(0°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & L_2c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & L_2s_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$A_3 = \begin{bmatrix} c_{\theta_3} & -s_{\theta_3}cos(0°) & s_{\theta_3}sen(0°) & L_3c_{\theta_3} \\ s_{\theta_3} & c_{\theta_3}cos(0°) & -c_{\theta_3}sen(0°) & L_3s_{\theta_3} \\ 0 & sen(0°) & cos(0°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & L_3c_{\theta_3} \\ s_{\theta_3} & c_{\theta_3} & 0 & L_3s_{\theta_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Figura 26 – Sistema de coordenadas.



Fonte: Do autor.

$$A_4 = \begin{bmatrix} c_{\theta_4} & -s_{\theta_4}cos(-90°) & s_{\theta_4}sen(-90°) & 0 \\ s_{\theta_4} & c_{\theta_4}cos(-90°) & -c_{\theta_4}sen(-90°) & 0 \\ 0 & sen(-90°) & cos(-90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{\theta_4} & 0 & -s_{\theta_4} & 0 \\ s_{\theta_4} & 0 & c_{\theta_4} & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$A_5 = \begin{bmatrix} c_{\theta_5} & -s_{\theta_5}cos(0°) & s_{\theta_5}sen(0°) & 0 \\ s_{\theta_5} & c_{\theta_5}cos(0°) & -c_{\theta_5}sen(0°) & 0 \\ 0 & sen(0°) & cos(0°) & L_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{\theta_5} & -s_{\theta_5} & 0 & 0 \\ s_{\theta_5} & c_{\theta_5} & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

A matriz de transformação homogênea T_0^5 final, obtida a partir do produto das matrizes (3.1) a (3.5), é dada por

$$T_0^5 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \Delta x \\ r_{21} & r_{22} & r_{23} & \Delta y \\ r_{31} & r_{32} & r_{33} & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

em que

$$\begin{aligned}
r_{11} &= s_1 s_5 + c_5[c_4(c_1c_2c_3 - c_1s_2s_3) - s_4(c_1c_2s_3 + c_1c_3s_2)],\\
r_{12} &= c_5 s_1 - s_5[c_4(c_1c_2c_3 - c_1s_2s_3) - s_4(c_1c_2s_3 + c_1c_3s_2)],\\
r_{13} &= -c_4(c_1c_2s_3 + c_1c_3s_2) - s_4(c_1c_2c_3 - c_1s_2s_3),\\
r_{21} &= c_5[c_4(c_2c_3s_1 - s_1s_2s_3) - s_4(c_2s_1s_3 + c_3s_1s_2)] - c_1s_5,\\
r_{22} &= -c_1c_5 - s_5[c_4(c_2c_3s_1 - s_1s_2s_3) - s_4(c_2s_1s_3 + c_3s_1s_2)],\\
r_{23} &= -c_4(c_2s_1s_3 + c_3s_1s_2) - s_4(c_2c_3s_1 - s_1s_2s_3),\\
r_{31} &= -c_5(c_4(c_2s_3 + c_3s_2) + s_4(c_2c_3 - s_2s_3)),\\
r_{32} &= s_5[c_4(c_2s_3 + c_3s_2) + s_4(c_2c_3 - s_2s_3)],\\
r_{33} &= s_4(c_2s_3 + c_3s_2) - c_4(c_2c_3 - s_2s_3),\\
\Delta x &= L_2c_1c_2 - L_4[c_4(c_1c_2s_3 + c_1c_3s_2) + s_4(c_1c_2c_3 - c_1s_2s_3)] + L_3c_1c_2c_3 - L_3c_1s_2s_3,\\
\Delta y &= L_2c_2s_1 - L_4[c_4(c_2s_1s_3 + c_3s_1s_2) + s_4(c_2c_3s_1 - s_1s_2s_3)] - L_3s_1s_2s_3 + L_3c_2c_3s_1,\\
\Delta z &= L_1 - L_2s_2 - L_4[c_4(c_2c_3 - s_2s_3) - s_4(c_2s_3 + c_3s_2)] - L_3c_2s_3 - L_3c_3s_2.
\end{aligned}$$

3.4 Modelagem virtual

3.4.1 SolidWorks

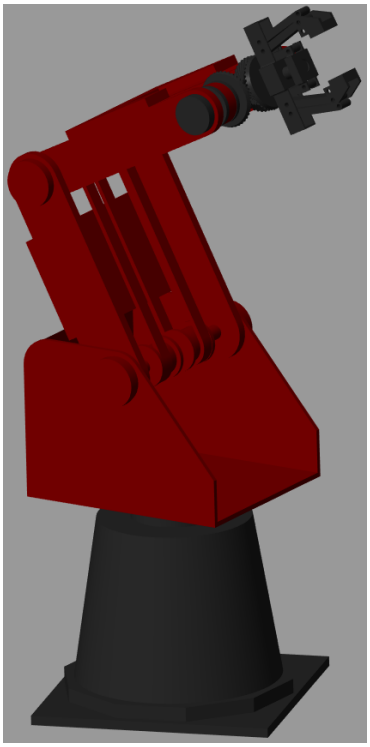
SOLIDWORKS® é um *software* de projeto auxiliado por computador (CAD) e engenharia auxiliada por computador (CAE) desenvolvido pela *Dassault Systèmes*. É amplamente utilizado na área de engenharia mecânica para projetar modelos 3D, simulações e gerar desenhos de engenharia (SAM; ARRIFIN; BUNIYAMIN, 2012).

Empregou-se o SOLIDWORKS® para realizar a modelagem do manipulador robótico ARM-7220-4, que pode ser visto na Figura 27. Com o desenho realizado, foi utilizado o *toolbox Simscape Multibody* para a importação ao *Simulink*. Desta forma, é gerado um arquivo com extensão *xml* para a execução no Matlab®.

3.4.2 Simulink

Para o controle da simulação do braço robótico, empregou-se o Simulink. Na Figura 28, mostra-se o ambiente simulacional. Para adicionar o arquivo do SOLIDWORKS® no Matlab®, deve-se inserir o comando *smimport*(“*nome_do_arquivo.xml*”) no *Command Window*. Caso se inicie a simulação como descrito na Figura 28, o manipulador

Figura 27 – Manipulador robótico.



Fonte: Do autor.

robótico irá se movimentar de maneira aleatória. Desse forma, foram realizadas alterações no diagrama, como a adição de alguns blocos, conforme a Figura 29, repetindo para os outros blocos, para garantir que a simulação sempre comece a partir de uma posição predefinida.

Os blocos adicionados são demostrados na Figura 30, o bloco *Slider Gain* permite o movimento do manipulador robótico, por meio de um *slider*. O bloco *Saturation* exerce a função de limitar o deslocamento dos elos, para se mover de forma mais real.

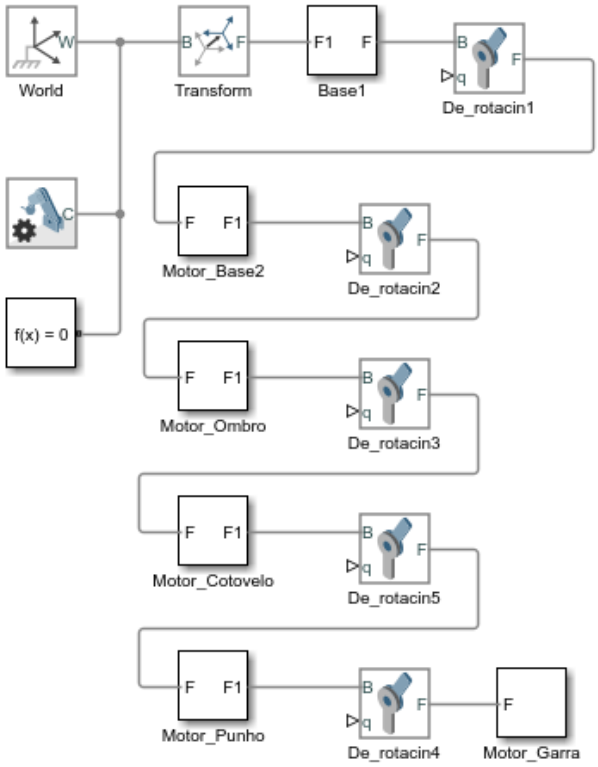
Os demais blocos visualizados em Figura 29, estão descritos abaixo.

- *World*: dá acesso ao "sistema de coordenadas" (quadro) do manipulador, também conhecido como quadro de solo, sendo um quadro de coordenadas imóvel, ortogonal sendo predefinido em qualquer modelo mecânico. A estrutura mundial é a base de todas as redes de estruturas em um modelo mecânico.

- *Mechanism Configuration*: um conjunto independente de blocos interligados com características mecânicas e de simulação é fornecido pelo bloco. Os parâmetros delta de linearização e gravidade são usados para calcular derivadas. Somente o mecanismo ao qual o bloco está fixado é afetado por essas características.

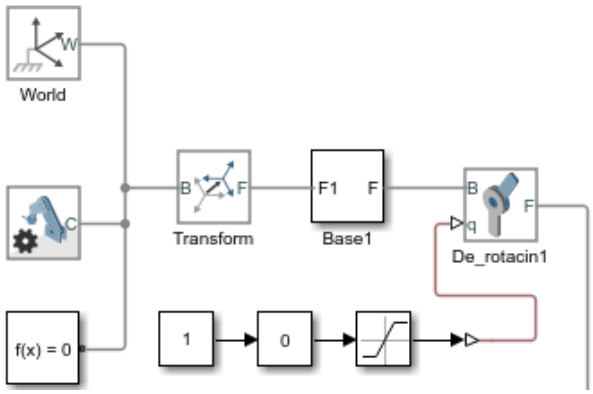
- *Solver Configuration*: O bloco especifica as configurações do solucionador exigidas pelo modelo para iniciar a simulação.

Figura 28 – Modelagem base.



Fonte: *Simulink*.

Figura 29 – Diagrama simulado ajustado.



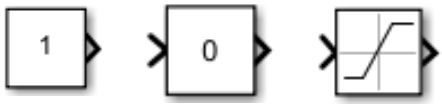
Fonte: *Simulink*.

- *Transform*: explica uma transição 3D fixa e rígida que ocorre entre dois quadros. Os componentes translacionais e rotacionais da transformação são especificados por dois componentes distintos. Pode-se misturar e combinar diferentes translações e rotações à vontade.

  As portas dos quadros B e F representam os quadros base e seguidor, respectivamente. A transformação no quadro base representa a orientação do eixo e a origem do quadro seguidor.

Figura 30 – Blocos inseridos.

(a) *Constant*. (b) *Slider Gain*. (c) *Saturation*.



Fonte: *Simulink*.

- *Base* 1: corpo importado de um arquivo .xml da peça 3D, desenvolvido em *SolidWorks*. Este bloco é o mesmo do Motor_Base2, Motor_Ombro, Motor_Cotovelo, Motor_Punho e Motor_Garra.
- *De _ rotacin*: retrata a ação de uma junta de revolução entre dois sistemas de coordenadas. Uma primitiva de revolução representa o único grau de liberdade rotacional para esta junta.

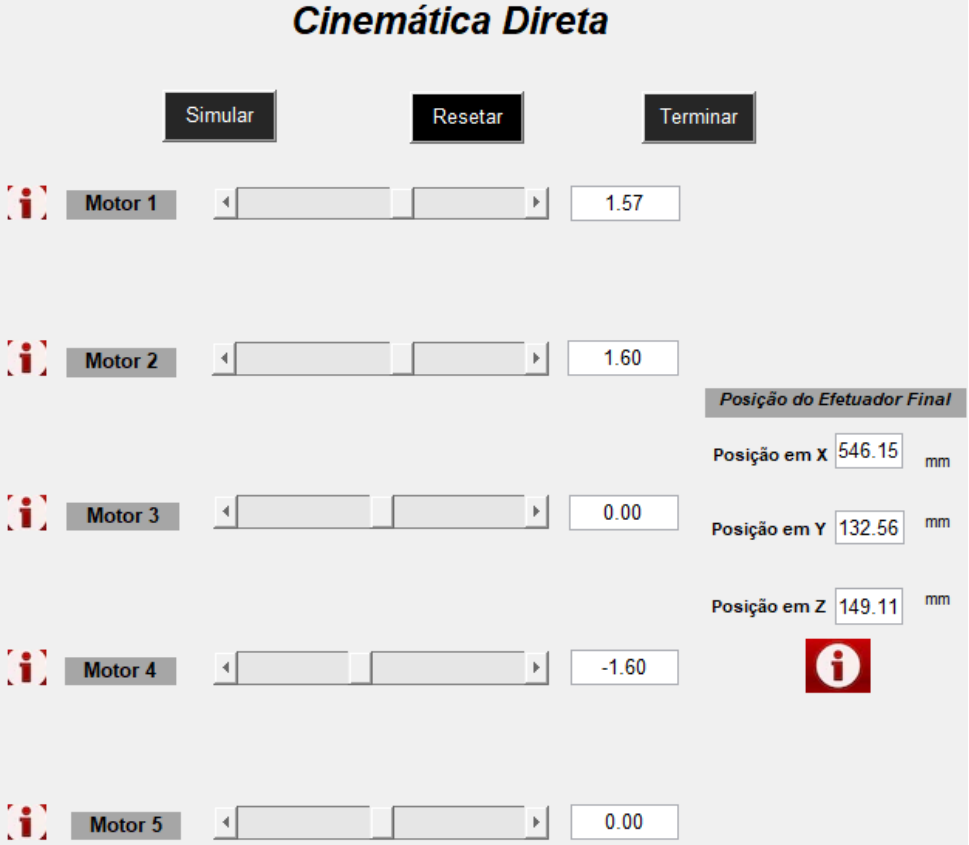
3.4.3 Interfaces

As interfaces gráficas de usuário, do inglês, *Graphical User Interfaces (GUIs)*, comumente chamadas de aplicativos, oferecem um meio fácil de controlar o programa. Elas permitem que os usuários interajam com o programa por meio de ações de apontar e clicar, eliminando a necessidade de os indivíduos aprenderem uma linguagem de programação ou inserir comandos para operar o aplicativo (MATHWORKS, 2024).

A Figura 31 mostra a interface criada. Para a realização do controle e projeto, é necessário implementar o código de programação para a definição do comportamento do manipulador e estrutura da *GUIDE*.

O campo de interação ao usuário possui módulo e botões. Em relação ao módulo, a interação ocorre por meio dos *Sliders*, possibilitando a alteração dos valores das juntas do manipulador robótico dados por Motor 1 a Motor 5 que correspondem, respectivamente, a junta da base, ombro, cotovelo, punho e rotação do punho, cujos valores são dados em radianos. Após os movimentos, é possível verificar a posição final do efetuador nas coordenadas x, y e z. Os botões estão relacionados para iniciar a simulação, reiniciar as posições do manipulador, finalizar a simulação e de fornecer informações para o usuário.

Figura 31 – Interface.



Fonte: *Matlab®*.

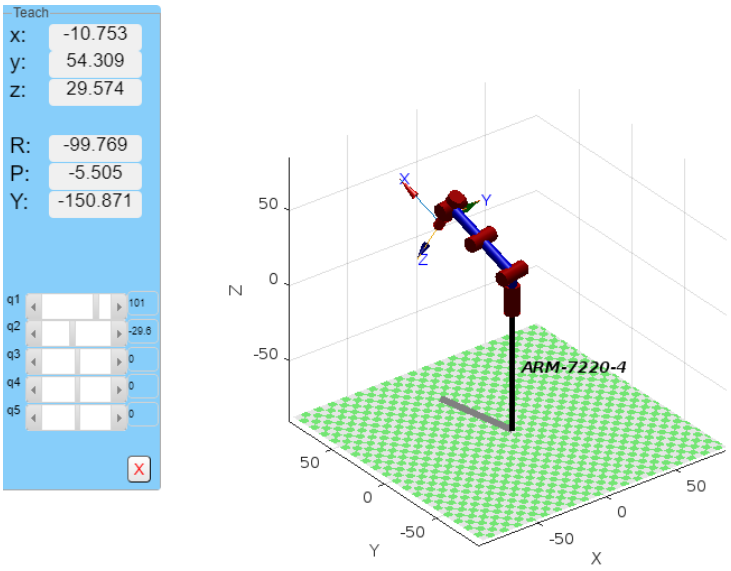
4 Resultados

Neste capítulo, os resultados serão apresentados e discutidos. A implementação da modelagem do manipulador robótico será realizada no *software Matlab®*, utilizando a ferramenta *Robotics Toolbox* (CORKE, 2017), com visualização 3D, por meio da cinemática, foi a principal ênfase do estudo. Também será efetuada a simulação com a inclusão do *software SolidWorks®* ao *Matlab®*, e comparando os resultados.

4.1 Análise da convenção DH

Nesta etapa, foi verificada se a posição do manipulador seria conforme mostrado na Figura 26, tanto quanto à posição dos elos, quanto à orientação das juntas. É possível verificar na Figura 32 que a junta i_2, que rotaciona o ombro, está conforme a Figura 26, ao apresentar um ângulo de $-29.6°$ no elo L_2. O código desenvolvido no Matlab® para realizar a simulação da Figura 32 é mostrado no Apêndice A. O resultado atendeu as expectativas, pois as juntas i_1, que rotaciona a base, e i_2 não dispõe da mesma função.

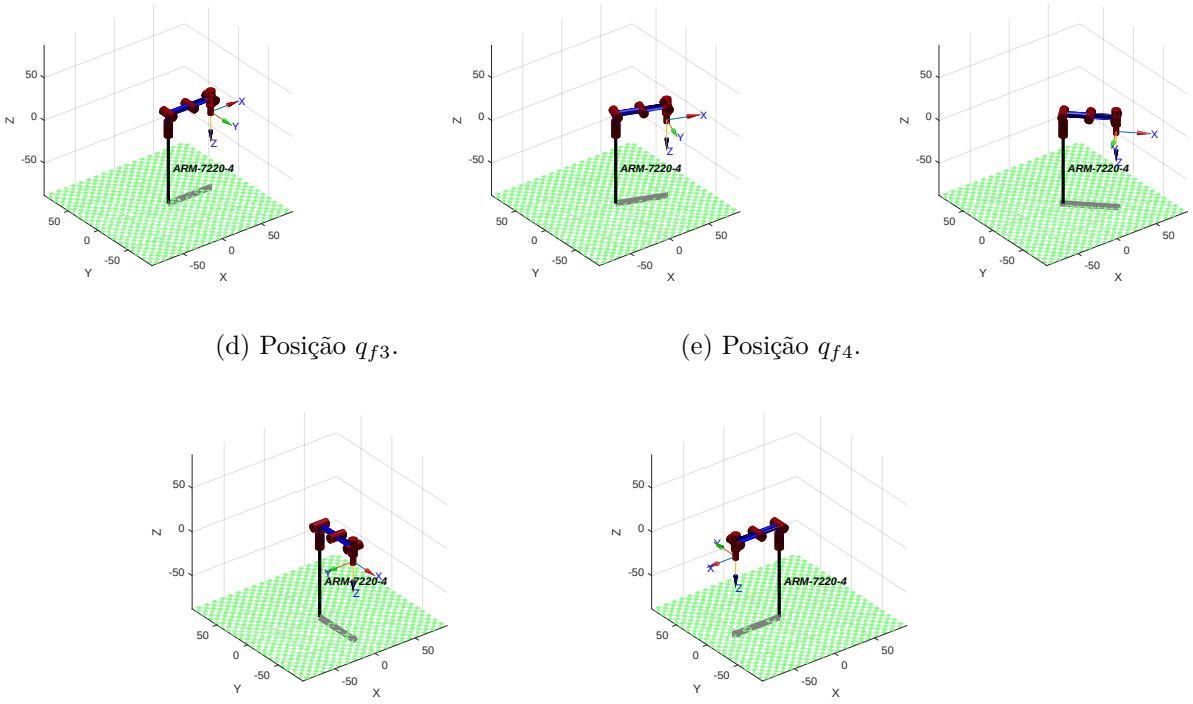
Figura 32 – Visualização do manipulador em 3D por meio dos parâmetros da Tabela 4.



Fonte: Do autor.

Figura 33 – Posição inicial e final do manipulador após os ajustes da tabela de DH — Experimento 1.

(a) Posição q_{f0}. (b) Posição q_{f1}. (c) Posição q_{f2}.



(d) Posição q_{f3}. (e) Posição q_{f4}.

Fonte: Do autor.

4.2 Experimento 1

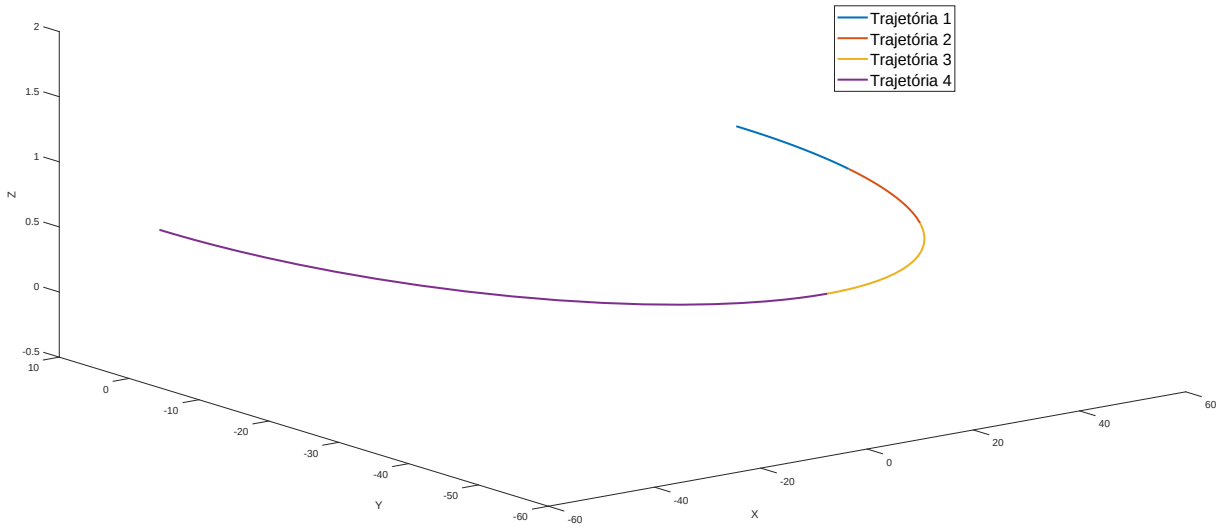
Para o experimento, foram definidos 4 movimentos, começando na posição inicial q_{f0}, até a posição final q_{f4}. As posições escolhidas foram

$$\begin{aligned} q_{f0} &= [0, 0, 0, 0, 0], \\ q_{f1} &= [-\pi/9, 0, 0, 0, 0], \\ q_{f2} &= [-\pi/4, 0, 0, 0, 0], \\ q_{f3} &= [-\pi/2, 0, 0, 0, 0], \\ q_{f4} &= [-\pi, 0, 0, 0, 0]. \end{aligned} \tag{4.1}$$

Desse modo, somente a junta i_1 será rotacionada. Os ângulos são dados em radianos. Pode-se visualizar a posição inicial q_{f0} a final q_{f4} do manipulador pela Figura 33. É possível verificar a trajetória por meio da Figura 34. O percurso do efetuador final corresponde ao desejado, pois apenas a junta q_1 está sendo rotacionada, em torno de z. É possível analisar as coordenadas das juntas, por meio da Figura 35, sendo a junta q_1, a única a ser deslocada.

Na Figura 36 pode-se visualizar a posição do efetuador final em centímetros, durante a trajetória. Observa-se que a coordenada z está no valor de 0,6 e não terá variação,

Figura 34 – Trajetória total do manipulador com as trajetórias q_{f_i} separadas por linha contínua — Experimento 1.



Fonte: Do autor.

pois a rotação é em torno de z. Os valores de x e y correspondem, pois x começa próximo a 54,3, indo até $-54{,}3$. Enquanto y começa em 0, passando em $-54{,}3$, a $-\pi/2$, e finalizando próximo de 0. O código desenvolvido no Matlab® para realizar a simulação da Figura 32 é mostrado no Apêndice B.

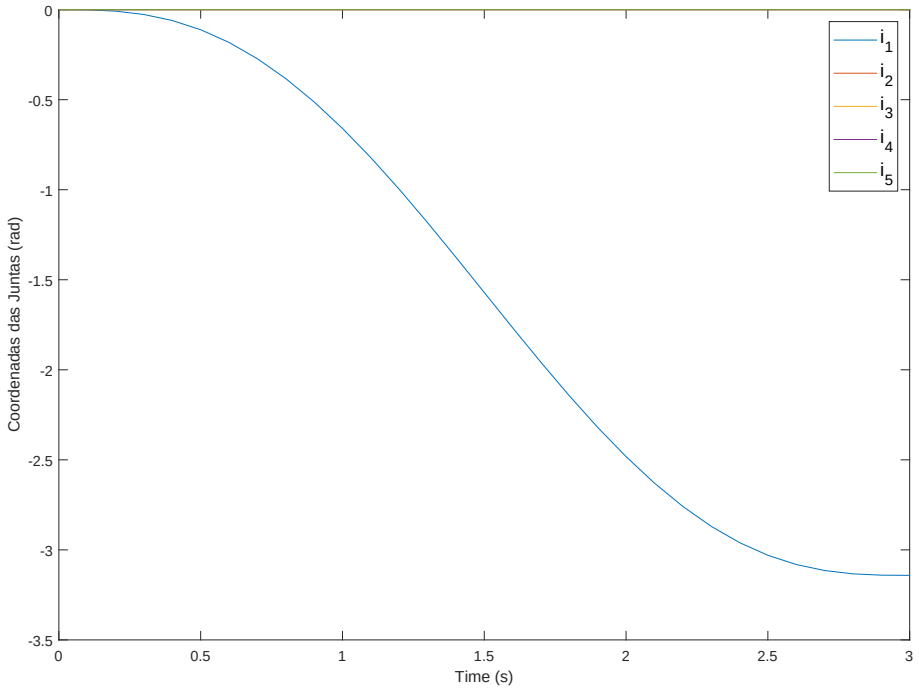
Os valores da matriz de transformação homogênea da posição q_{f_0}, q_{f_1}, q_{f_2}, q_{f_3} e q_{f_4} são dadas por

$$T_{q_{f_0}} = \begin{bmatrix} 1 & 0 & 0 & 54{,}3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0{,}6 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.2}$$

$$T_{q_{f_1}} = \begin{bmatrix} 0{,}9397 & -0{,}3420 & 0 & -51{,}0253 \\ -0{,}3420 & -0{,}9397 & 0 & -18{,}5717 \\ 0 & 0 & -1 & 0{,}6 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.3}$$

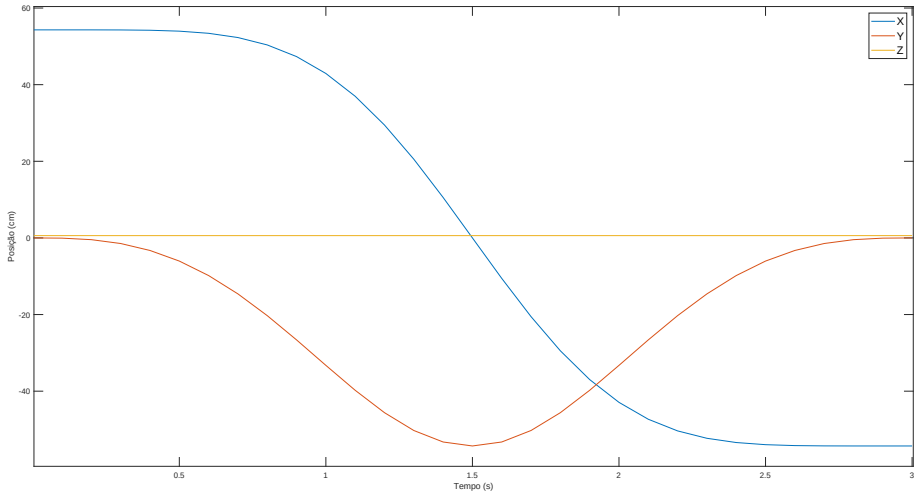
$$T_{q_{f_2}} = \begin{bmatrix} 0{,}7071 & -0{,}7071 & 0 & 38{,}3959 \\ -0{,}7071 & -0{,}7071 & 0 & -38{,}3959 \\ 0 & 0 & -1 & 0{,}6 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.4}$$

Figura 35 – Coordenadas das juntas i_1 a i_5, durante a trajetória q_{f_0} a q_{f_4} — Experimento 1.



Fonte: Do autor.

Figura 36 – Posição do efetuador final nas coordenadas x, y e z — Experimento 1.



Fonte: Do autor.

$$T_{q_{f_3}} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & -54{,}3 \\ 0 & 0 & -1 & 0{,}6 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.5}$$

$$T_{q_{f4}} = \begin{bmatrix} -1 & 0 & 0 & -54{,}3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0{,}6 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.6}$$

Após os ajustes das posições, o Experimento 1 apresentou respostas satisfatórias, pois as juntas estão se movimentando conforme o que foi apresentando na Figura 24. Desse modo, as trajetórias, coordenadas das juntas e a posição do efetuador final cumpriram com o previsto.

4.3 Experimento 2

Neste experimento, foi possível verificar a simulação do manipulador de forma mais realista, por meio da junção do *SolidWorks®* e *Matlab®*. O código desenvolvido no *Matlab®* para realizar a simulação deste experimento é mostrado no Apêndice C.

Após a execução do programa e pressionando o botão *Simular* na *GUIDE*, mostra-se o manipulador em sua posição predefinida ao importar do *SolidWorks*, vista na Figura 37, via ferramenta *Mechanics Explorer*. Tal ferramenta permite visualizar e explorar modelos multicorpos, e a posição inicial está pré-definida ao ser importada pelo toolbox *Simscape Multibody*. As coordenadas do efetuador final podem ser visualizadas na Tabela 5.

Tabela 5 – Posição predefinida do efetuador final nas coordenadas x, y e z — Visualização da GUIDE.

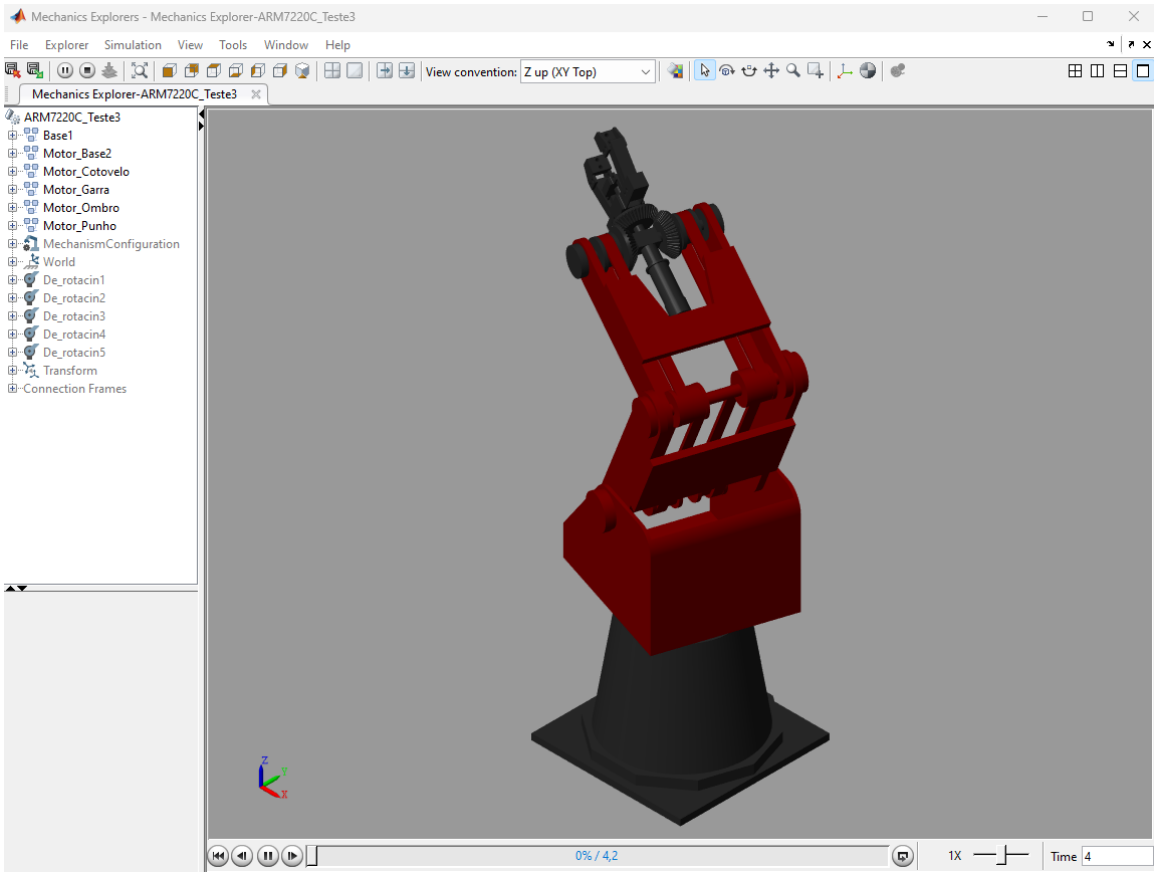
Coordenadas	Posição (mm)
x	550,00
y	117,55
z	172,00

Foi realizada a tentativa para deixar o manipulador disposto da mesma forma, que na Figura 33, da posição inicial até a final, que podem ser visualizadas na Figura 38. Para a posição inicial q_{f0}, as coordenadas das juntas e do efetuador final podem ser visualizadas na Tabela 6 e Tabela 7, respectivamente.

Como os próximos movimentos serão somente da rotação da junta da base (Motor 1), as juntas do Motor 2 ao Motor 5 irão permanecer com os mesmos valores. Desse modo, têm-se os dados para as juntas nas posições da base, como indicado nas Tabelas 8 e 9.

Com os resultados expressos, é notório que tanto os ângulos das juntas, quanto os valores das coordenadas do efetuador final foram bem diferentes ao do Experimento 1. Na Tabela 10, é possível verificar uma comparação em relação à posição q_{f4}, constatando uma diferença evidente.

Figura 37 – Posição predefinida do manipulador.



Fonte: Do autor.

Tabela 6 – Ângulo das juntas na posição inicial q_{f0} — Visualização da GUIDE.

Junta i	Valores (rad)	Valores (graus)
Base (Motor 1)	1,57	89,95°
Ombro (Motor 2)	1,60	91,67°
Cotovelo (Motor 3)	0,00	0,00°
Movimento angular do punho (Motor 4)	-1,60	−91,67°
Rotação do punho (Motor 5)	0,00	0,00°

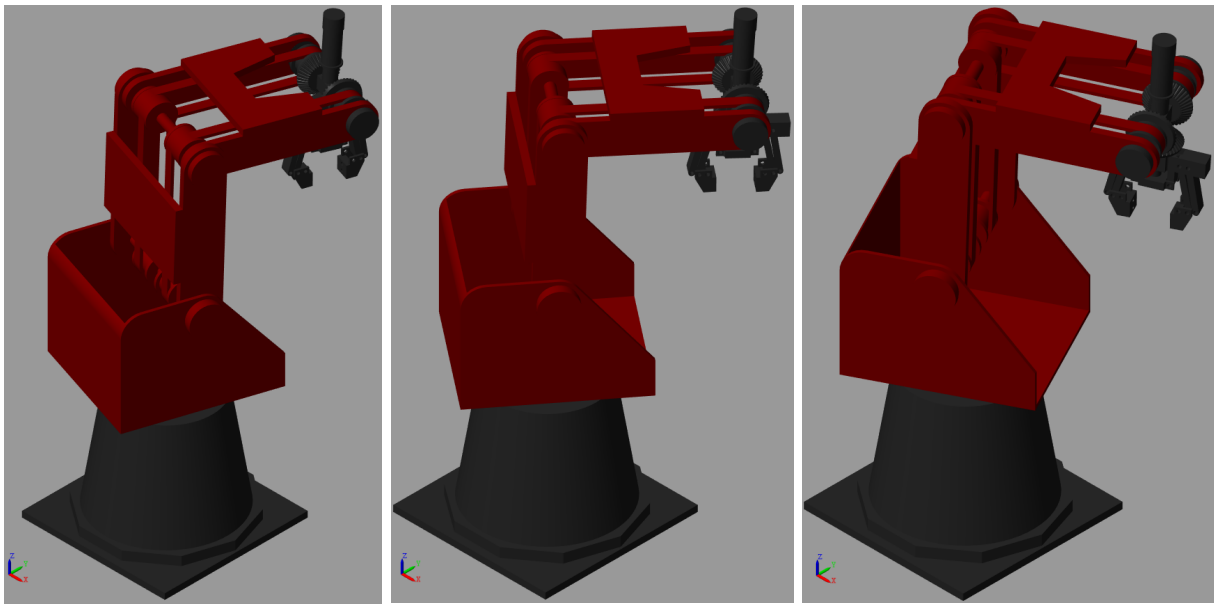
Tabela 7 – Posição inicial q_{f0} do efetuador final nas coordenadas x, y e z — Visualização da GUIDE.

Coordenadas	Posição (mm)
x	546,15
y	132,56
z	149,11

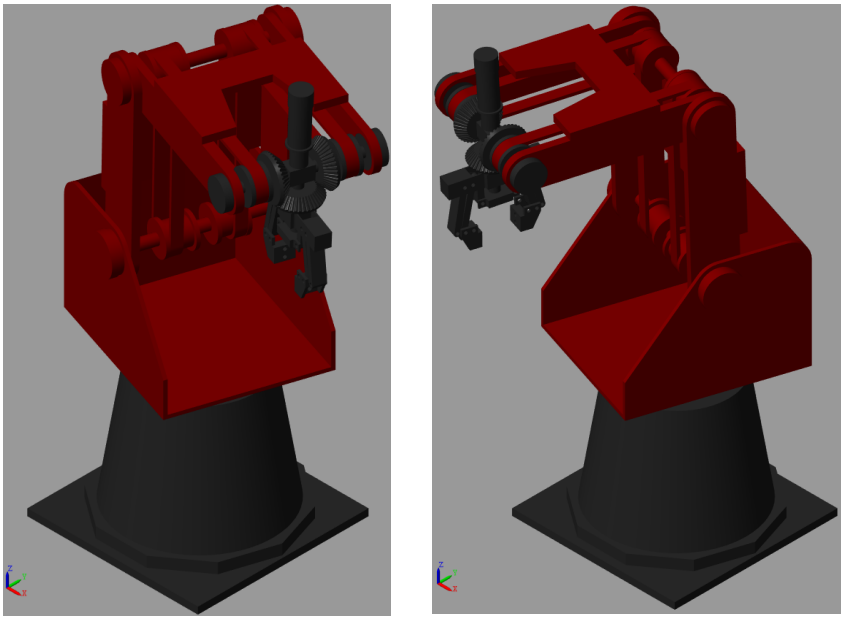
Desta maneira, o Experimento 2 produziu resultados que foram pertinentes, de forma visual em relação ao Experimento 1. A simulação do Experimento 2 atingiu seus objetivos a cerca da interface criada, e sua relação aos movimentos do manipulador e mostrou que pode funcionar, mas também identificou áreas que precisam de refinamento e

Figura 38 – Posições do manipulador robótico — Experimento 2.

(a) Posição q_{f0}. (b) Posição q_{f1}. (c) Posição q_{f2}.



(d) Posição q_{f3}. (e) Posição q_{f4}.



Fonte: Do autor.

Tabela 8 – Ângulo das juntas nas posições da Base (Motor 1) — Visualização da GUIDE.

Posição da junta	Valores (rad)	Valores (graus)
q_{f1}	1,92	110,00°
q_{f2}	2,35	134,64°
q_{f3}	3,14	179,90°
q_{f4}	4,71	269,86°

Tabela 9 – Posições do efetuador final nas coordenadas x, y e z — Visualização da GUIDE.

Posições da Base (Motor 1)	Coordenadas	Posição (mm)
q_{f1}	x	545,33
	y	135,86
	z	149,11
q_{f2}	x	544,27
	y	140,01
	z	149,11
q_{f3}	x	542,31
	y	147,44
	z	149,11
q_{f4}	x	538,05
	y	162,29
	z	149,11

Tabela 10 – Comparação das coordenadas do efetuador e ângulos da junta na posição q_{f4} nos Experimentos 1 e 2.

Posição do Efetuador		**Experimento 1**	**Experimento 2**
Coordenadas	x	$-54300\ mm$	$538{,}05\ mm$
	y	$0\ mm$	$162{,}29\ mm$
	z	$600\ mm$	$149{,}11\ mm$
Ângulo da junta da Base (Motor 1)		$-180°$	269,86°

otimização, como, por exemplo, a realização da mudança de coordenadas do Experimento 2, para garantir que tanto o Experimento 1 e 2 comecem do mesmo ponto, de modo que os valores da posição do efetuador final sejam similares.

5 Conclusão e Trabalhos Futuros

Este trabalho apresentou a modelagem e simulação do braço robótico ARM-7220-4, de cinco juntas revolutas. A modelagem cinemática do manipulador foi desenvolvida por meio da convenção de Denavit-Hartenberg, que permitiu obter a posição e orientação do efetuador final, utilizando uma matriz de transformação homogênea. Com essa implementação, foi possível obter uma representação gráfica da posição do ARM-7220-4, desenvolvida mediante uma animação para a trajetória gerada, que , no primeiro momento, a simulação foi realizada apenas em ambiente Matlab®.

Para uma visualização do manipulador mais realista, foi utilizado o *software* SOLIDWORKS® junto ao Matlab®. Entretanto, a simulação apresentou divergências em relação às coordenadas das juntas e posição do efetuador final nos movimentos realizados do robô, quando comparados aos obtidos por meio da ferramenta *Robotics Toolbox*. O motivo é que o sistema de coordenadas iniciado no SOLIDWORKS® possivelmente é diferente do sistema utilizado no *Robotics Toolbox*, o que explica os problemas vistos.

Mesmo diante aos desafios e dificuldades, a disposição do manipulador, em relação a sua representação física, obteve um resultado satisfatório. A integração dos dois programas para projetar e analisar manipuladores robóticos, apresenta vantagens notáveis em termos de amplos recursos de simulação e processos de projeto otimizados. Contudo, para aumentar a eficácia da integração e superar quaisquer obstáculos, são necessárias preparação e organização.

Em síntese, a utilização de simuladores virtuais para treinamento e instrução de manipuladores robóticos é um desenvolvimento notável no campo da educação em robótica. Com a ajuda de simuladores, profissionais e estudantes podem mergulhar totalmente em situações do mundo real sem incorrer nos custos e riscos dos testes presenciais. Eles fornecem uma plataforma que combina compreensão teórica com aplicação prática. Com simuladores, os usuários podem testar e obter um conhecimento mais profundo de sistemas robóticos complexos, enquanto aprimoram suas habilidades em um ambiente simulado.

5.1 Sugestões de Trabalhos Futuros

Como etapas futuras, destacam-se alguns pontos a serem considerados para a complementação deste trabalho:

- Ajuste das condições iniciais na simulação via SOLIDWORKS®;
- Avaliação de outros programas para a simulação, por exemplo, *CoppeliaSim* ;
- Implementação da cinemática inversa e o modelo dinâmico;

- Estudo dos gêmeos digitais.

Referências

ABREU, P. *Aplicações industriais de robôs*. Dissertação (Mestrado em Automação, Instrumentação e Controle) — Universidade de Porto, 2002.

AUTOMATION, A. F. A. *Is a Vacuum Gripper Right for Your Collaborative Robot Application?* 2019. <https://www.automate.org/blogs/is-a-vacuum-gripper-right-for-your-collaborative-robot-application>. Acesso em 10/08/2023.

BRASIL, U. R. *ROBÔS MANIPULADORES: FUNÇÃO, PRINCIPAIS USOS E VANTAGENS*. 2022. <https://www.universal-robots.com/br/blog/robÃ‡s-manipuladores-funcÃčo-principais-usos-e-vantagens/>. Acesso em 11/03/2023.

CAMPOS, L. F. A. D. A. *INTELIGÊNCIA ARTIFICIAL E INSTRUMENTALIZAÇÃO DIGITAL NO ENSINO: A SEMIFORMAÇÃO NA ERA DA AUTOMATIZAÇÃO COMPUTACIONAL*. Dissertação (Mestrado) — UNIVERSIDADE ESTADUAL PAULISTA, 2018.

CORKE, P. I. *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*. Second. [S.l.]: Springer, 2017. ISBN 978-3-319-54412-0.

CRAIG, J. J. *Robótica*. 3ª. ed. [S.l.]: São Paulo: Pearson Education do Brasil, 2012.

GAMERO, I. *Robôs Industriais: tudo o que você precisa saber!* 2018. <https://pollux.com.br/blog/robos-industriais-tudo-o-que-voce-precisa-saber/>. Acesso em 11/03/2023.

INDUSTRY, D. *Que robô industrial escolher?* 2023. <https://guide.directindustry.com/pt/que-robo-industrial-escolher/>. Acesso em 12/03/2023.

KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. [S.l.: s.n.], 2004. v. 3, p. 2149–2154.

KUMAR, R. R.; CHAND, P. Inverse kinematics solution for trajectory tracking using artificial neural networks for Scorbot ER-4U. In: *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*. [S.l.: s.n.], 2015. p. 364–369. doi: 10.1109/ICARA.2015.7081175.

LOPES, A. M. *Modelação cinemática e dinâmica de manipuladores de estrutura em série*. Dissertação (Mestrado em Automação, Instrumentação e Controle) — Universidade de Porto, 2002.

MAHENDRA, S. *What is an End Effector In Robotics?* 2023. <https://www.aiplusinfo.com/blog/what-is-an-end-effector-in-robotics/>. Acesso em 26/07/2023.

MATARIC, M. J. *Introdução à Robótica*. 1ª. ed. [S.l.]: Editora Unesp, 2014.

MATHWORKS. *MATLAB GUI*. 2024. <https://www.mathworks.com/discovery/matlab-gui.html>. Acesso em 20/01/2024.

ROBOTS, U. *Industrial Robots in Manufacturing*. 2019. <https://www.universal-robots.com/in/blog/industrial-robots-in-manufacturing/>. Acesso em 09/03/2023.

ROBOTS, U. *Magnetic Grippers For Manufacturers*. 2021. <https://www.universal-robots.com/blog/magnetic-grippers-for-manufacturers/>. Acesso em 10/08/2023.

ROBOTS, U. *Adhesive gripper ADHESO*. 2023. <https://www.universal-robots.com/plus/products/schunk/adhesive-gripper-adheso/>. Acesso em 10/08/2023.

ROHMER, E.; SINGH, S. P. N.; FREESE, M. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2013. Www.coppeliarobotics.com.

SAM, R.; ARRIFIN, K.; BUNIYAMIN, N. Simulation of pick and place robotics system using solidworks softmotion. In: *2012 International Conference on System Engineering and Technology (ICSET)*. [S.l.: s.n.], 2012. p. 1–6. doi: 10.1109/ICSEngT.2012.6339325.

SERRATO-SOSA, A. *Gripper Design*. 2016. <https://wikis.utexas.edu/display/RMD/Gripper+Design>. Acesso em 10/08/2023.

SICILIANO, B. *Robotics: Modelling, Planning and Control*. 1ª. ed. [S.l.]: Springer, 2009.

SILVA, L. F. P. C. e. *Utilização de manipuladores em ambientes não estruturados*. Dissertação (Mestrado), 2010.

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. *Robot Modeling and Control*. 2ª. ed. [S.l.]: Wiley, 2020.

TOMORROW, R. *Electric Grippers*. 2018. <https://www.roboticstomorrow.com/article/2018/04/electric-grippers/11628/>. Acesso em 10/08/2023.

UKO, E. *Fundamentos das garras pneumáticas para aplicações industriais*. 2022. <https://www.digikey.com.br/pt/articles/fundamentals-of-pneumatic-grippers-for-industrial-applications>. Acesso em 10/08/2023.

WEBOTS. *http://www.cyberbotics.com*. 1998. Open-source Mobile Robot Simulation Software. Disponível em: <http://www.cyberbotics.com>.

WELT. *Roboter zerquetscht Arbeiter bei Volkswagen*. 2015. <https://www.welt.de/vermischtes/article143392085/Roboter-zerquetscht-Arbeiter-bei-Volkswagen.html>. Acesso em 17/10/2023.

A Análise da convenção de DH

A seguir, encontra-se o código implementado no Matlab, em que se desenvolve a etapa da visualização do manipulador, a partir da Tabela 4.

```
clc
clearvars
close all

% Inicializacao Toolbox Robotics
startup_rvc

%Tamanho dos elos
L1 = 17.1;
L2 = 27.5;
L3 = 26.8;
L4 = 16.5;

% Criacao dos elos com os parametros de Denavit-Hartenberg
% theta = angulo da junta (rad)
% d = deslocamento do elo (m)
% a = comprimento do elo (m)
% alpha = torcao do elo (rad)
% sigma = tipo de junta (0: rotativa ou 1: prismatica)

theta1 = 0;
theta2 = 0;
theta3 = 0;
theta4 = 0;
theta5 = 0;

%% Etapa de verificacao - Tabela 3

%L(x) = Link([theta d a alpha sigma])
L(1) = Link([theta1 L1 0 -pi/2 0]);
L(2) = Link([theta2 0 L2 0 0]);
L(3) = Link([theta3 0 L3 0 0]);
L(4) = Link([theta4 0 0 -pi/2 0]);
L(5) = Link([theta5 L4 0 0 0]);

robo = SerialLink(L, 'name', 'ARM-7220-4');

figure(2)
view([-90 -90 90])
robo.teach
```

B Experimento 1

A seguir, encontra-se o código implementado no Matlab, em que se desenvolve a etapa da visualização da trajetória do manipulador, juntamente as coordenadas das juntas e a posição do efetuador final.

```
clc
clearvars
close all

% Inicializacao Toolbox Robotics
startup_rvc

%Tamanho dos elos
L1 = 17.1;
L2 = 27.5;
L3 = 26.8;
L4 = 16.5;

% Criacao dos elos com os parametros de Denavit-Hartenberg
% theta = angulo da junta (rad)
% d = deslocamento do elo (m)
% a = comprimento do elo (m)
% alpha = torcao do elo (rad)
% sigma = tipo de junta (0: rotativa ou 1: prismatica)

theta1 = 0;
theta2 = 0;
theta3 = 0;
theta4 = 0;
theta5 = 0;

%% Etapa de teste - Tabela 4

%L(x) = Link([theta d a alpha sigma])
L(1) = Link([theta1 L1 0 -pi/2 0]);
L(2) = Link([theta2 0 L2 0 0]);
L(3) = Link([theta3 0 L3 0 0]);
L(4) = Link([theta4 0 0 -pi/2 0]);
L(5) = Link([theta5 L4 0 0 0]);

robo = SerialLink(L, 'name', 'ARM-7220-4');

%Angulos pre-definidos para movimentos para obter os pontos
qf0 = [0 0 0 0 0];
```

```
qf1 = [-pi/9 0 0 0 0];
qf2 = [-pi/4 0 0 0 0];
qf3 = [-pi/2 0 0 0 0];
qf4 = [-pi 0 0 0 0];

q1 = [qf0; qf1; qf2; qf3; qf4];

figure()

for i=1:1:4

t = 0:0.1:3; %Vetor de tempo t variando de 0 a 3 em incrementos de 0,1.
q = jtraj(q1(i,:), q1(i+1,:), t);%Gera uma trajetoria de espa o de
    junta da configuracao
% de junta inicial qf0 at  a configuracao de junta de destino
% qf1 em um intervalo de tempo especificado t.

Tr = fkine(robo,q); %Calcula as matrizes de transformacao (T) que
% representam a pose do efetuador final a cada
% intervalo de tempo da trajetoria.

%Loop
%Em seguida, o codigo percorre cada etapa de tempo na matriz de
%tempo t e extrai os componentes translacionais (coordenadas x, y e z)
%da pose do efetuador final da matriz de transformacao correspondente T.

for i=1:1:length(t) %Interacao de 1 at  o comprimento do vetor t
          T = Tr(i); %Esta linha atribui a i-esima matriz de
              transformacao T da matriz Tr a variavel T.
          trs = transl(T); %Esta linha extrai o componente translacional
              da matriz
             %de transformacao T e o atribui a variavel trs.
          xx(i) = trs(1); %Esta linha armazena a coordenada x do vetor
              translacional
            %na i-esima posicao da matriz xx.
          yy(i) = trs(2); %Esta linha armazena a coordenada y do vetor
              translacional
           %na i-esima posicao da matriz yy.
          zz(i) = trs(3); %Esta linha armazena a coordenada z do vetor
              translacional
          %na i-esima posicao da matriz zz.

end


plot3(xx,yy,zz,'LineWidth',2,'XDataSource','X','YDataSource','Y','
    ZDataSource','Z');%Plota a trajetoria 3D do efetuador final
```

```
80 %usando as coordenadas x, y e z armazenadas nas matrizes xx, yy e zz,
81 %respectivamente. Cada trajetoria e uma cor diferente.
82 plot(robo,q);%Plota o manipulador robotico no espa o articular para a
       trajetoria q.
83 hold on
84
85 end
86
87 %% Coordenadas das Juntas
88
89 figure()
90 for i=1:1:4
91
92 t = 0:0.1:3; %Vetor de tempo t variando de 0 a 3 em incrementos de 0,1.
93 q = jtraj(q1(i,:), q1(i+1,:), t);%Gera uma trajetria de espa o de junta
       da configuracao
94 %%de junta inicial qf0 ate a configuracao de junta de destino
95 %%qf1 em um intervalo de tempo especificado t.
96 plot(t,q(:,i));
97 hold on
98 end
99
100
101 figure()
102 %%Posicao do efetuador final - Ira funcionar somente quando a unica
       rotacao for da junta 1, de 0 a -pi.
103
104 for i=1:1:4
105
106 t = 0:0.1:3; %Vetor de tempo t variando de 0 a 3 em incrementos de 0,1.
107 q = jtraj(qf0, qf4, t);%Gera uma trajetoria de espa o de junta da
       configuracao
108 %de junta inicial qf0 ate a configuracao de junta de destino
109 %qf1 em um intervalo de tempo especificado t.
110
111 Tr = fkine(robo,q); %Calcula as matrizes de transformacao (T) que
112 %representam a pose do efetuador final a cada
113 %intervalo de tempo da trajetoria.
114
115 p = Tr.transl;
116 plot(t,p);
117 hold on
118
119 end
```

C Experimento 2

A seguir, encontra-se o código implementado no Matlab, em que se desenvolve a etapa da visualização da trajetória do manipulador junto ao *SolidWorks*. Em seguida, é possível visualizar o código da função Cinemática Direta.

```
function varargout = Experimento2(varargin)
% EXPERIMENTO2 MATLAB code for Experimento2.fig
%      EXPERIMENTO2, by itself, creates a new EXPERIMENTO2 or raises the existing
%      singleton*.
%
%      H = EXPERIMENTO2 returns the handle to a new EXPERIMENTO2 or the handle to
%      the existing singleton*.
%
%      EXPERIMENTO2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in EXPERIMENTO2.M with the given input arguments.
%
%      EXPERIMENTO2('Property','Value',...) creates a new EXPERIMENTO2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Teste_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Teste_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Teste

% Last Modified by GUIDE v2.5 18-Dec-2023 20:56:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Experimento2_OpeningFcn, ...
```

```
32                    'gui_OutputFcn',  @Experimento2_OutputFcn, ...
33                    'gui_LayoutFcn',  [] , ...
34                    'gui_Callback',   []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 % End initialization code - DO NOT EDIT
45
46
47
48 % --- Executes just before Teste is made visible.
49 function Experimento2_OpeningFcn(hObject, eventdata, handles, varargin)
50 % This function has no output args, see OutputFcn.
51 % hObject    handle to figure
52 % eventdata  reserved - to be defined in a future version of MATLAB
53 % handles    structure with handles and user data (see GUIDATA)
54 % varargin   command line arguments to Teste (see VARARGIN)
55
56 % Choose default command line output for Teste
57 handles.output = hObject;
58
59 % Update handles structure
60 guidata(hObject, handles);
61
62 % UIWAIT makes Teste wait for user response (see UIRESUME)
63 % uiwait(handles.figure1);
64
65
66 % --- Outputs from this function are returned to the command line.
67 function varargout = Experimento2_OutputFcn(hObject, eventdata, handles)
68 % varargout  cell array for returning output args (see VARARGOUT);
69 % hObject    handle to figure
70 % eventdata  reserved - to be defined in a future version of MATLAB
71 % handles    structure with handles and user data (see GUIDATA)
72
73 % Get default command line output from handles structure
74 varargout{1} = handles.output;
75
76
77 % --- Executes on slider movement.
78 function slider1_Callback(hObject, eventdata, handles)
```

```
79  % hObject    handle to slider1 (see GCBO)
80  % eventdata  reserved - to be defined in a future version of MATLAB
81  % handles    structure with handles and user data (see GUIDATA)
82
83  % Hints: get(hObject,'Value') returns position of slider
84  %        get(hObject,'Min') and get(hObject,'Max') to determine range of
        slider
85  ModelName='ARM7220C_Experimento2';
86  global var;
87
88  t1=get(handles.slider1,'value');
89  t2=get(handles.slider2,'value');
90  t3=get(handles.slider3,'value');
91  t4=get(handles.slider4,'value');
92  t5=get(handles.slider5,'value');
93  %----------------------------------------
94  set(handles.edit1,'string',num2str(t1,'%.2f'));
95  set(handles.edit2,'string',num2str(t2,'%.2f'));
96  set(handles.edit3,'string',num2str(t3,'%.2f'));
97  set(handles.edit4,'string',num2str(t4,'%.2f'));
98  set(handles.edit5,'string',num2str(t5,'%.2f'));
99  %----------------------------------------
100 set_param([ModelName '/Slider Gain'],'Gain',num2str(t1));
101 set_param([ModelName '/Slider Gain1'],'Gain',num2str(t2));
102 set_param([ModelName '/Slider Gain2'],'Gain',num2str(t3));
103 set_param([ModelName '/Slider Gain3'],'Gain',num2str(t4));
104 set_param([ModelName '/Slider Gain4'],'Gain',num2str(t5));
105
106 % A1=[cosd(t1) 0 -sind(t1) 0;sind(t1) 0 cosd(t1) 0;0 -1 1 172;0 0 0 1];
107 % A2=[cosd(t2) -sind(t2) 0 280*cosd(t2); sind(t2) cosd(t2) 0 280*sind(t2
        );0 0 1 0;0 0 0 1];
108 % A3=[cosd(t3) -sind(t3) 0 270*cosd(t3); sind(t3) cosd(t3) 0 270*sind(t2
        );0 0 1 0;0 0 0 1];
109 % A4=[cosd(t4) 0 -sind(t4) 0;sind(t4) 0 cosd(t4) 0;0 -1 1 0;0 0 0 1];
110 % A5=[cosd(t5) -sind(t5) 0 0;sind(t5) cosd(t5) 0 0;0 0 1 117.55;0 0 0
        1];
111 % T= A1*A2*A3*A4;
112 % Px=T(1,4);
113 % Py=T(2,4);
114 % Pz=T(3,4);
115
116 [Px,Py,Pz]= Cinematica_Direta(t1,t2,t3,t4,t5);
117
118 set(handles.edit8,'string',num2str(Px,'%.2f'));
119 set(handles.edit9,'string',num2str(Py,'%.2f'));
120 set(handles.edit10,'string',num2str(Pz,'%.2f'));
121
```

```
122 % --- Executes during object creation, after setting all properties.
123 function slider1_CreateFcn(hObject, eventdata, handles)
124 % hObject    handle to slider1 (see GCBO)
125 % eventdata  reserved - to be defined in a future version of MATLAB
126 % handles    empty - handles not created until after all CreateFcns
        called
127
128 % Hint: slider controls usually have a light gray background.
129 if isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
130     set(hObject,'BackgroundColor',[.9 .9 .9]);
131 end
132
133 % --- Executes on slider movement.
134 function slider2_Callback(hObject, eventdata, handles)
135 % hObject    handle to slider2 (see GCBO)
136 % eventdata  reserved - to be defined in a future version of MATLAB
137 % handles    structure with handles and user data (see GUIDATA)
138
139 % Hints: get(hObject,'Value') returns position of slider
140 %        get(hObject,'Min') and get(hObject,'Max') to determine range of
         slider
141 ModelName='ARM7220C_Experimento2';
142
143 t1=get(handles.slider1,'value');
144 t2=get(handles.slider2,'value');
145 t3=get(handles.slider3,'value');
146 t4=get(handles.slider4,'value');
147 t5=get(handles.slider5,'value');
148 %----------------------------------------
149 set(handles.edit1,'string',num2str(t1,'%.2f'));
150 set(handles.edit2,'string',num2str(t2,'%.2f'));
151 set(handles.edit3,'string',num2str(t3,'%.2f'));
152 set(handles.edit4,'string',num2str(t4,'%.2f'));
153 set(handles.edit5,'string',num2str(t5,'%.2f'));
154 %----------------------------------------
155 set_param([ModelName '/Slider Gain'],'Gain',num2str(t1));
156 set_param([ModelName '/Slider Gain1'],'Gain',num2str(t2));
157 set_param([ModelName '/Slider Gain2'],'Gain',num2str(t3));
158 set_param([ModelName '/Slider Gain3'],'Gain',num2str(t4));
159 set_param([ModelName '/Slider Gain4'],'Gain',num2str(t5));
160
161 % A1=[cosd(t1) 0 -sind(t1) 0;sind(t1) 0 cosd(t1) 0;0 -1 1 172;0 0 0 1];
162 % A2=[cosd(t2) -sind(t2) 0 280*cosd(t2); sind(t2) cosd(t2) 0 280*sind(t2
        );0 0 1 0;0 0 0 1];
163 % A3=[cosd(t3) -sind(t3) 0 270*cosd(t3); sind(t3) cosd(t3) 0 270*sind(t2
        );0 0 1 0;0 0 0 1];
```

```
% A4=[cosd(t4) 0 -sind(t4) 0;sind(t4) 0 cosd(t4) 0;0 -1 1 0;0 0 0 1];
% A5=[cosd(t5) -sind(t5) 0 0;sind(t5) cosd(t5) 0 0;0 0 1 117.55;0 0 0
    1];
% T= A1*A2*A3*A4;
% Px=T(1,4);
% Py=T(2,4);
% Pz=T(3,4);

[Px,Py,Pz]=Cinematica_Direta(t1,t2,t3,t4,t5);

set(handles.edit8,'string',num2str(Px,'%.2f'));
set(handles.edit9,'string',num2str(Py,'%.2f'));
set(handles.edit10,'string',num2str(Pz,'%.2f'));

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
     slider
ModelName='ARM7220C_Experimento2';

t1=get(handles.slider1,'value');
t2=get(handles.slider2,'value');
t3=get(handles.slider3,'value');
t4=get(handles.slider4,'value');
t5=get(handles.slider5,'value');
%---------------------------------------
set(handles.edit1,'string',num2str(t1,'%.2f'));
set(handles.edit2,'string',num2str(t2,'%.2f'));
```

```
207 set(handles.edit3,'string',num2str(t3,'%.2f'));
208 set(handles.edit4,'string',num2str(t4,'%.2f'));
209 set(handles.edit5,'string',num2str(t5,'%.2f'));
210 %----------------------------------------
211 set_param([ModelName '/Slider Gain'],'Gain',num2str(t1));
212 set_param([ModelName '/Slider Gain1'],'Gain',num2str(t2));
213 set_param([ModelName '/Slider Gain2'],'Gain',num2str(t3));
214 set_param([ModelName '/Slider Gain3'],'Gain',num2str(t4));
215 set_param([ModelName '/Slider Gain4'],'Gain',num2str(t5));
216
217 % A1=[cosd(t1) 0 -sind(t1) 0;sind(t1) 0 cosd(t1) 0;0 -1 1 172;0 0 0 1];
218 % A2=[cosd(t2) -sind(t2) 0 280*cosd(t2); sind(t2) cosd(t2) 0 280*sind(t2
    );0 0 1 0;0 0 0 1];
219 % A3=[cosd(t3) -sind(t3) 0 270*cosd(t3); sind(t3) cosd(t3) 0 270*sind(t2
    );0 0 1 0;0 0 0 1];
220 % A4=[cosd(t4) 0 -sind(t4) 0;sind(t4) 0 cosd(t4) 0;0 -1 1 0;0 0 0 1];
221 % A5=[cosd(t5) -sind(t5) 0 0;sind(t5) cosd(t5) 0 0;0 0 1 117.55;0 0 0
    1];
222 % T= A1*A2*A3*A4;
223 % Px=T(1,4);
224 % Py=T(2,4);
225 % Pz=T(3,4);
226
227 [Px,Py,Pz]=Cinematica_Direta(t1,t2,t3,t4,t5);
228
229 set(handles.edit8,'string',num2str(Px,'%.2f'));
230 set(handles.edit9,'string',num2str(Py,'%.2f'));
231 set(handles.edit10,'string',num2str(Pz,'%.2f'));
232 % --- Executes during object creation, after setting all properties.
233 function slider3_CreateFcn(hObject, eventdata, handles)
234 % hObject    handle to slider3 (see GCBO)
235 % eventdata  reserved - to be defined in a future version of MATLAB
236 % handles    empty - handles not created until after all CreateFcns
    called
237
238 % Hint: slider controls usually have a light gray background.
239 if isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
240     set(hObject,'BackgroundColor',[.9 .9 .9]);
241 end
242
243
244 % --- Executes on slider movement.
245 function slider4_Callback(hObject, eventdata, handles)
246 % hObject    handle to slider4 (see GCBO)
247 % eventdata  reserved - to be defined in a future version of MATLAB
248 % handles    structure with handles and user data (see GUIDATA)
```

```
249
250 % Hints: get(hObject,'Value') returns position of slider
251 %        get(hObject,'Min') and get(hObject,'Max') to determine range of
        slider
252 ModelName='ARM7220C_Experimento2';
253
254 t1=get(handles.slider1,'value');
255 t2=get(handles.slider2,'value');
256 t3=get(handles.slider3,'value');
257 t4=get(handles.slider4,'value');
258 t5=get(handles.slider5,'value');
259 %----------------------------------------
260 set(handles.edit1,'string',num2str(t1,'%.2f'));
261 set(handles.edit2,'string',num2str(t2,'%.2f'));
262 set(handles.edit3,'string',num2str(t3,'%.2f'));
263 set(handles.edit4,'string',num2str(t4,'%.2f'));
264 set(handles.edit5,'string',num2str(t5,'%.2f'));
265 %----------------------------------------
266 set_param([ModelName '/Slider Gain'],'Gain',num2str(t1));
267 set_param([ModelName '/Slider Gain1'],'Gain',num2str(t2));
268 set_param([ModelName '/Slider Gain2'],'Gain',num2str(t3));
269 set_param([ModelName '/Slider Gain3'],'Gain',num2str(t4));
270 set_param([ModelName '/Slider Gain4'],'Gain',num2str(t5));
271
272 % A1=[cosd(t1) 0 -sind(t1) 0;sind(t1) 0 cosd(t1) 0;0 -1 1 172;0 0 0 1];
273 % A2=[cosd(t2) -sind(t2) 0 280*cosd(t2); sind(t2) cosd(t2) 0 280*sind(t2
        );0 0 1 0;0 0 0 1];
274 % A3=[cosd(t3) -sind(t3) 0 270*cosd(t3); sind(t3) cosd(t3) 0 270*sind(t2
        );0 0 1 0;0 0 0 1];
275 % A4=[cosd(t4) 0 -sind(t4) 0;sind(t4) 0 cosd(t4) 0;0 -1 1 0;0 0 0 1];
276 % A5=[cosd(t5) -sind(t5) 0 0;sind(t5) cosd(t5) 0 0;0 0 1 117.55;0 0 0
        1];
277 % T= A1*A2*A3*A4;
278 % Px=T(1,4);
279 % Py=T(2,4);
280 % Pz=T(3,4);
281
282 [Px,Py,Pz]=Cinematica_Direta(t1,t2,t3,t4,t5);
283
284 set(handles.edit8,'string',num2str(Px,'%.2f'));
285 set(handles.edit9,'string',num2str(Py,'%.2f'));
286 set(handles.edit10,'string',num2str(Pz,'%.2f'));
287
288 % --- Executes during object creation, after setting all properties.
289 function slider4_CreateFcn(hObject, eventdata, handles)
290 % hObject    handle to slider4 (see GCBO)
291 % eventdata  reserved - to be defined in a future version of MATLAB
```

```
292 % handles    empty - handles not created until after all CreateFcns
    called
293
294 % Hint: slider controls usually have a light gray background.
295 if isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
296     set(hObject,'BackgroundColor',[.9 .9 .9]);
297 end
298
299
300 % --- Executes on slider movement.
301 function slider5_Callback(hObject, eventdata, handles)
302 % hObject    handle to slider5 (see GCBO)
303 % eventdata  reserved - to be defined in a future version of MATLAB
304 % handles    structure with handles and user data (see GUIDATA)
305
306 % Hints: get(hObject,'Value') returns position of slider
307 %        get(hObject,'Min') and get(hObject,'Max') to determine range of
     slider
308 ModelName='ARM7220C_Experimento2';
309
310 t1=get(handles.slider1,'value');
311 t2=get(handles.slider2,'value');
312 t3=get(handles.slider3,'value');
313 t4=get(handles.slider4,'value');
314 t5=get(handles.slider5,'value');
315 %----------------------------------------
316 set(handles.edit1,'string',num2str(t1,'%.2f'));
317 set(handles.edit2,'string',num2str(t2,'%.2f'));
318 set(handles.edit3,'string',num2str(t3,'%.2f'));
319 set(handles.edit4,'string',num2str(t4,'%.2f'));
320 set(handles.edit5,'string',num2str(t5,'%.2f'));
321 %----------------------------------------
322 set_param([ModelName '/Slider Gain'],'Gain',num2str(t1));
323 set_param([ModelName '/Slider Gain1'],'Gain',num2str(t2));
324 set_param([ModelName '/Slider Gain2'],'Gain',num2str(t3));
325 set_param([ModelName '/Slider Gain3'],'Gain',num2str(t4));
326 set_param([ModelName '/Slider Gain4'],'Gain',num2str(t5));
327
328 % A1=[cosd(t1) 0 -sind(t1) 0;sind(t1) 0 cosd(t1) 0;0 -1 1 172;0 0 0 1];
329 % A2=[cosd(t2) -sind(t2) 0 280*cosd(t2); sind(t2) cosd(t2) 0 280*sind(t2
    );0 0 1 0;0 0 0 1];
330 % A3=[cosd(t3) -sind(t3) 0 270*cosd(t3); sind(t3) cosd(t3) 0 270*sind(t2
    );0 0 1 0;0 0 0 1];
331 % A4=[cosd(t4) 0 -sind(t4) 0;sind(t4) 0 cosd(t4) 0;0 -1 1 0;0 0 0 1];
332 % A5=[cosd(t5) -sind(t5) 0 0;sind(t5) cosd(t5) 0 0;0 0 1 117.55;0 0 0
    1];
```

```
% T= A1*A2*A3*A4;
% Px=T(1,4);
% Py=T(2,4);
% Pz=T(3,4);

[Px,Py,Pz]= Cinematica_Direta(t1,t2,t3,t4,t5);

set(handles.edit8,'string',num2str(Px,'%.2f'));
set(handles.edit9,'string',num2str(Py,'%.2f'));
set(handles.edit10,'string',num2str(Pz,'%.2f'));

% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as
    a double
set(handles.edit1,'String', sprintf('%.2f'));

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
```

```
375 end
376
377
378 function edit5_Callback(hObject, eventdata, handles)
379 % hObject    handle to edit5 (see GCBO)
380 % eventdata  reserved - to be defined in a future version of MATLAB
381 % handles    structure with handles and user data (see GUIDATA)
382
383 % Hints: get(hObject,'String') returns contents of edit5 as text
384 %        str2double(get(hObject,'String')) returns contents of edit5 as
        a double
385
386
387 % --- Executes during object creation, after setting all properties.
388 function edit5_CreateFcn(hObject, eventdata, handles)
389 % hObject    handle to edit5 (see GCBO)
390 % eventdata  reserved - to be defined in a future version of MATLAB
391 % handles    empty - handles not created until after all CreateFcns
        called
392
393 % Hint: edit controls usually have a white background on Windows.
394 %       See ISPC and COMPUTER.
395 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
396     set(hObject,'BackgroundColor','white');
397 end
398
399
400
401 function edit4_Callback(hObject, eventdata, handles)
402 % hObject    handle to edit4 (see GCBO)
403 % eventdata  reserved - to be defined in a future version of MATLAB
404 % handles    structure with handles and user data (see GUIDATA)
405
406 % Hints: get(hObject,'String') returns contents of edit4 as text
407 %        str2double(get(hObject,'String')) returns contents of edit4 as
        a double
408
409
410 % --- Executes during object creation, after setting all properties.
411 function edit4_CreateFcn(hObject, eventdata, handles)
412 % hObject    handle to edit4 (see GCBO)
413 % eventdata  reserved - to be defined in a future version of MATLAB
414 % handles    empty - handles not created until after all CreateFcns
        called
415
416 % Hint: edit controls usually have a white background on Windows.
```

```
417 %       See ISPC and COMPUTER.
418 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
419     set(hObject,'BackgroundColor','white');
420 end
421
422
423 function edit3_Callback(hObject, eventdata, handles)
424 % hObject    handle to edit3 (see GCBO)
425 % eventdata  reserved - to be defined in a future version of MATLAB
426 % handles    structure with handles and user data (see GUIDATA)
427
428 % Hints: get(hObject,'String') returns contents of edit3 as text
429 %        str2double(get(hObject,'String')) returns contents of edit3 as
    a double
430
431
432 % --- Executes during object creation, after setting all properties.
433 function edit3_CreateFcn(hObject, eventdata, handles)
434 % hObject    handle to edit3 (see GCBO)
435 % eventdata  reserved - to be defined in a future version of MATLAB
436 % handles    empty - handles not created until after all CreateFcns
    called
437
438 % Hint: edit controls usually have a white background on Windows.
439 %       See ISPC and COMPUTER.
440 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
441     set(hObject,'BackgroundColor','white');
442 end
443
444
445
446 function edit2_Callback(hObject, eventdata, handles)
447 % hObject    handle to edit2 (see GCBO)
448 % eventdata  reserved - to be defined in a future version of MATLAB
449 % handles    structure with handles and user data (see GUIDATA)
450
451 % Hints: get(hObject,'String') returns contents of edit2 as text
452 %        str2double(get(hObject,'String')) returns contents of edit2 as
    a double
453
454
455 % --- Executes during object creation, after setting all properties.
456 function edit2_CreateFcn(hObject, eventdata, handles)
457 % hObject    handle to edit2 (see GCBO)
458 % eventdata  reserved - to be defined in a future version of MATLAB
```

```
459 % handles    empty - handles not created until after all CreateFcns
    called
460
461 % Hint: edit controls usually have a white background on Windows.
462 %       See ISPC and COMPUTER.
463 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
464     set(hObject,'BackgroundColor','white');
465 end
466
467
468 function edit8_Callback(hObject, eventdata, handles)
469 % hObject    handle to edit8 (see GCBO)
470 % eventdata  reserved - to be defined in a future version of MATLAB
471 % handles    structure with handles and user data (see GUIDATA)
472
473 % Hints: get(hObject,'String') returns contents of edit8 as text
474 %        str2double(get(hObject,'String')) returns contents of edit8 as
    a double
475
476 % --- Executes during object creation, after setting all properties.
477 function edit8_CreateFcn(hObject, eventdata, handles)
478 % hObject    handle to edit8 (see GCBO)
479 % eventdata  reserved - to be defined in a future version of MATLAB
480 % handles    empty - handles not created until after all CreateFcns
    called
481
482 % Hint: edit controls usually have a white background on Windows.
483 %       See ISPC and COMPUTER.
484 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
485     set(hObject,'BackgroundColor','white');
486 end
487
488
489 function edit9_Callback(hObject, eventdata, handles)
490 % hObject    handle to edit9 (see GCBO)
491 % eventdata  reserved - to be defined in a future version of MATLAB
492 % handles    structure with handles and user data (see GUIDATA)
493
494 % Hints: get(hObject,'String') returns contents of edit9 as text
495 %        str2double(get(hObject,'String')) returns contents of edit9 as
    a double
496
497 % --- Executes during object creation, after setting all properties.
498 function edit9_CreateFcn(hObject, eventdata, handles)
499 % hObject    handle to edit9 (see GCBO)
```

```
500 % eventdata  reserved - to be defined in a future version of MATLAB
501 % handles    empty - handles not created until after all CreateFcns
    called
502
503 % Hint: edit controls usually have a white background on Windows.
504 %       See ISPC and COMPUTER.
505 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
506     set(hObject,'BackgroundColor','white');
507 end
508
509
510 function edit10_Callback(hObject, eventdata, handles)
511 % hObject    handle to edit10 (see GCBO)
512 % eventdata  reserved - to be defined in a future version of MATLAB
513 % handles    structure with handles and user data (see GUIDATA)
514
515 % Hints: get(hObject,'String') returns contents of edit10 as text
516 %        str2double(get(hObject,'String')) returns contents of edit10 as
     a double
517
518
519 % --- Executes during object creation, after setting all properties.
520 function edit10_CreateFcn(hObject, eventdata, handles)
521 % hObject    handle to edit10 (see GCBO)
522 % eventdata  reserved - to be defined in a future version of MATLAB
523 % handles    empty - handles not created until after all CreateFcns
    called
524
525 % Hint: edit controls usually have a white background on Windows.
526 %       See ISPC and COMPUTER.
527 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
528     set(hObject,'BackgroundColor','white');
529 end
530
531 % --- Executes on button press in pushbutton1.
532 function pushbutton1_Callback(hObject, eventdata, handles)
533 % hObject    handle to pushbutton1 (see GCBO)
534 % eventdata  reserved - to be defined in a future version of MATLAB
535 % handles    structure with handles and user data (see GUIDATA)
536 ModelName='ARM7220C_Experimento2';
537 open_system(ModelName);
538 set_param(ModelName,'BlockReduction','off');
539 set_param(ModelName,'StopTime','inf');
540 set_param(ModelName,'simulationMode','normal');
541 set_param(ModelName,'StartFcn','1');
```

```
542 set_param(ModelName, 'SimulationCommand', 'start');
543
544 % --- Executes on button press in pushbutton3.
545 function pushbutton3_Callback(hObject, eventdata, handles)
546 % hObject    handle to pushbutton3 (see GCBO)
547 % eventdata  reserved - to be defined in a future version of MATLAB
548 % handles    structure with handles and user data (see GUIDATA)
549 t1 = 0;
550 t2 = 0;
551 t3 = 0;
552 t4 = 0;
553 t5 = 0;
554
555 [Px,Py,Pz] = Cinematica_Direta(t1,t2,t3,t4,t5);
556 ModelName='ARM7220C_Experimento2';
557
558 set_param([ModelName '/Slider Gain'],'Gain',num2str(t1));
559 set_param([ModelName '/Slider Gain1'],'Gain',num2str(t2));
560 set_param([ModelName '/Slider Gain2'],'Gain',num2str(t3));
561 set_param([ModelName '/Slider Gain3'],'Gain',num2str(t4));
562 set_param([ModelName '/Slider Gain4'],'Gain',num2str(t5));
563 %-------------------------------------
564 set(handles.slider1,'value',t1);
565 set(handles.slider2,'value',t2);
566 set(handles.slider3,'value',t3);
567 set(handles.slider4,'value',t4);
568 set(handles.slider5,'value',t5);
569 %-----------------------------------------
570 set(handles.edit1,'string',num2str(0));
571 set(handles.edit2,'string',num2str(0));
572 set(handles.edit3,'string',num2str(0));
573 set(handles.edit4,'string',num2str(0));
574 set(handles.edit5,'string',num2str(0));
575 set(handles.edit8,'string',num2str(Px,'%.2f'));
576 set(handles.edit9,'string',num2str(Py,'%.2f'));
577 set(handles.edit10,'string',num2str(Pz,'%.2f'));
578
579
580 % --- Executes on button press in pushbutton5.
581 function pushbutton5_Callback(hObject, eventdata, handles)
582 % hObject    handle to pushbutton5 (see GCBO)
583 % eventdata  reserved - to be defined in a future version of MATLAB
584 % handles    structure with handles and user data (see GUIDATA)
585 close;
586
587 % --- Executes on button press in pushbutton7.
588 function pushbutton7_Callback(hObject, eventdata, handles)
```

```
589 % hObject    handle to pushbutton7 (see GCBO)
590 % eventdata  reserved - to be defined in a future version of MATLAB
591 % handles    structure with handles and user data (see GUIDATA)
592 msgbox('Motor 1 est  relacionado a rota  o da base do rob  ARM-7220C
        .','Informa  o','help');
593
594
595 % --- Executes on button press in pushbutton8.
596 function pushbutton8_Callback(hObject, eventdata, handles)
597 % hObject    handle to pushbutton8 (see GCBO)
598 % eventdata  reserved - to be defined in a future version of MATLAB
599 % handles    structure with handles and user data (see GUIDATA)
600 msgbox("Motor 2 est  relacionado a rota  o do ombro do rob  ARM-7220
        C.","Informa  o","help");
601
602
603 % --- Executes on button press in pushbutton9.
604 function pushbutton9_Callback(hObject, eventdata, handles)
605 % hObject    handle to pushbutton9 (see GCBO)
606 % eventdata  reserved - to be defined in a future version of MATLAB
607 % handles    structure with handles and user data (see GUIDATA)
608 msgbox("Motor 3 est  relacionado a rota  o do cotovelo do rob  ARM
        -7220C.","Informa  o","help");
609
610
611 % --- Executes on button press in pushbutton10.
612 function pushbutton10_Callback(hObject, eventdata, handles)
613 % hObject    handle to pushbutton10 (see GCBO)
614 % eventdata  reserved - to be defined in a future version of MATLAB
615 % handles    structure with handles and user data (see GUIDATA)
616 msgbox("Motor 4 est  relacionado ao movimento angular do punho do rob
         ARM-7220C.","Informa  o","help");
617
618
619 % --- Executes on button press in pushbutton11.
620 function pushbutton11_Callback(hObject, eventdata, handles)
621 % hObject    handle to pushbutton11 (see GCBO)
622 % eventdata  reserved - to be defined in a future version of MATLAB
623 % handles    structure with handles and user data (see GUIDATA)
624 msgbox("Motor 5 est  relacionado a rota  o do punho do rob  ARM-7220
        C.","Informa  o","help");
625
626
627 % --- Executes on button press in pushbutton12.
628 function pushbutton12_Callback(hObject, eventdata, handles)
629 % hObject    handle to pushbutton12 (see GCBO)
630 % eventdata  reserved - to be defined in a future version of MATLAB
```

```
631 % handles    structure with handles and user data (see GUIDATA)
632 %msgbox("As posi  es s o refenciadas a posi  o do efetuador final (
        garra),
633 %em rela  o a um sistema de coordenadas x, y e z","Informa  o","help
        ");
634 msgbox("Os valores de x, y e z referem-se    posi  o do efetuador
        final (garra) em rela  o ao sistema de coordenadas da base.","
        Informa  o","help");
```

```
1  function [Px, Py, Pz]= Cinematica_Direta(t1,t2,t3,t4,t5)
2
3  A1=[cosd(t1) 0 -sind(t1) 0;sind(t1) 0 cosd(t1) 0;0 -1 1 172;0 0 0 1];
4  A2=[cosd(t2) -sind(t2) 0 280*cosd(t2); sind(t2) cosd(t2) 0 280*sind(t2)
       ;0 0 1 0;0 0 0 1];
5  A3=[cosd(t3) -sind(t3) 0 270*cosd(t3); sind(t3) cosd(t3) 0 270*sind(t2)
       ;0 0 1 0;0 0 0 1];
6  A4=[cosd(t4) 0 -sind(t4) 0;sind(t4) 0 cosd(t4) 0;0 -1 1 0;0 0 0 1];
7  A5=[cosd(t5) -sind(t5) 0 0;sind(t5) cosd(t5) 0 0;0 0 1 117.55;0 0 0 1];
8
9  T= A1*A2*A3*A4*A5;
10
11 Px = T(1,4);
12 Py = T(2,4);
13 Pz = T(3,4);
14 % p= T*[0;0;0;1];
15 %
16 % Px= p(2);
17 % Py= p(3);
18 % Pz= p(1);
19 end
```

TERMO DE RESPONSABILIDADE

O texto do trabalho de conclusão de curso intitulado "Construção de um simulador virtual para o braço robótico ARM-7220-4" é de minha inteira responsabilidade. Declaro que não há utilização indevida de texto, material fotográfico ou qualquer outro material pertencente a terceiros sem a devida citação ou consentimento dos referidos autores.

João Monlevade, 27 de junho de 2024 .

Pedro Augusto Silva Andrade
Nome completo do(a) aluno(a)