UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA DE CONTROLE E AUTOMAÇÃO
TCC II

JOSÉ RUDINEY DE SOUSA MIRANDA JÚNIOR

# Implementação de um robô cartesiano três eixos controlado via MatLab

Manaus - Am

2024

JOSÉ RUDINEY DE SOUSA MIRANDA JÚNIOR

# Implementação de um robô cartesiano três eixos controlado via MatLab

Trabalho de Conclusão de Curso apresentado no curso de Engenharia de Controle e Automação na Universidade do Estado do Amazonas

Orientador: Dr. Israel Mazaira Morales

Manaus - Am

2024

**RESUMO**

Este trabalho apresenta o desenvolvimento de um robô cartesiano XYZ controlado via MATLAB, com o objetivo de demonstrar sua viabilidade e eficácia no contexto da automação e robótica. O projeto foi dividido em três etapas principais: design mecânico, estruturação dos movimentos no CLP e CPU Motion, e elaboração de scripts no MATLAB para controle e comunicação. O design mecânico foi cuidadosamente planejado e executado, utilizando materiais disponíveis no laboratório para minimizar custos e otimizar recursos. A estruturação dos movimentos no CLP e na CPU Motion envolveu a configuração precisa dos motores e o desenvolvimento de lógicas de controle para garantir a coordenação eficiente dos movimentos nos eixos X, Y e Z. Foram elaborados scripts no MATLAB para comunicação com o CLP, permitindo o envio de comandos precisos para o robô em tempo real. Os resultados dos testes de funcionamento demonstraram um desempenho satisfatório do robô em cumprir as tarefas propostas, dentro dos limites especificados. Embora algumas limitações tenham sido identificadas durante os testes, como perda de precisão em altas velocidades de deslocamento, os resultados obtidos corroboram a viabilidade e a eficácia do sistema de controle desenvolvido. Sugestões para trabalhos futuros incluem aprimoramentos mecânicos para reduzir a inércia do sistema e melhorar a precisão dos movimentos, desenvolvimento de algoritmos avançados de controle e exploração de aplicações específicas em diferentes setores industriais. Este estudo representa um passo importante na jornada em direção a sistemas robóticos mais avançados e eficientes, fornecendo insights valiosos para futuras pesquisas e desenvolvimentos na área de automação e robótica.

**ABSTRACT**

This paper presents the development of a XYZ Cartesian robot controlled via MATLAB, aiming to demonstrate its feasibility and effectiveness in the context of automation and robotics. The project was divided into three main stages: mechanical design, structuring of movements in the PLC and CPU Motion, and development of scripts in MATLAB for control and communication. The mechanical design was carefully planned and executed, using materials available in the laboratory to minimize costs and optimize resources. The structuring of movements in the PLC and CPU Motion involved the precise configuration of the motors and the development of control logic to ensure efficient coordination of movements in the X, Y, and Z axes. MATLAB scripts were developed for communication with the PLC, allowing the sending of precise commands to the robot in real-time. The results of the operational tests demonstrated satisfactory performance of the robot in fulfilling the proposed tasks within the specified limits. Although some limitations were identified during the tests, such as loss of precision at high displacement speeds, the results obtained corroborate the feasibility and effectiveness of the developed control system. Suggestions for future work include mechanical enhancements to reduce system inertia and improve movement accuracy, development of advanced control algorithms, and exploration of specific applications in different industrial sectors. This study represents an important step in the journey towards more advanced and efficient robotic systems, providing valuable insights for future research and developments in the field of automation and robotics.

**SUMÁRIO**

# 1 INTRODUCAO

A automação e a robótica têm desempenhado um papel cada vez mais importante na otimização de processos industriais e na busca por soluções eficientes em diversas áreas. Nesse contexto, o desenvolvimento de sistemas robóticos capazes de realizar movimentos precisos e coordenados em três dimensões torna-se fundamental para atender às demandas crescentes por eficiência e qualidade. Este trabalho propõe a implementação e análise de um robô cartesiano XYZ controlado via MATLAB, visando demonstrar sua viabilidade e eficácia em aplicações práticas.

O design de pesquisa adotado foi experimental, com o objetivo de testar a eficiência de um protocolo de comunicação específico para controlar o movimento do robô cartesiano XYZ desenvolvido. As variáveis independentes consistem na implementação do protocolo de comunicação, enquanto as variáveis dependentes incluem o desempenho do robô em termos de precisão e velocidade de resposta. Para garantir a validade dos resultados, foram controladas diversas condições de teste, incluindo diferentes cenários e condições operacionais.

Na seção seguinte, serão apresentados os fundamentos teóricos relacionados à robótica, aos controladores lógicos programáveis (CLPs), à linguagem Ladder, aos sistemas cartesianos de três eixos, aos servo motores e ao protocolo TCP/IP. Essa revisão da literatura oferece uma base sólida para compreender os conceitos e tecnologias subjacentes ao desenvolvimento do robô cartesiano XYZ.

Posteriormente, serão descritos detalhadamente os materiais e métodos utilizados no desenvolvimento do robô, incluindo o design mecânico, a estruturação dos movimentos no CLP e na CPU Motion, além da elaboração de scripts no MATLAB para comunicação e controle dos movimentos. Essa seção fornece uma visão abrangente do processo de implementação do sistema robótico, destacando os passos e procedimentos adotados.

Os resultados obtidos durante o desenvolvimento do robô cartesiano XYZ serão discutidos em relação aos objetivos propostos, destacando as conquistas alcançadas, os desafios enfrentados e as possíveis melhorias para trabalhos futuros. Além disso, são apresentadas considerações finais sobre o estudo, ressaltando sua importância e contribuição para o avanço da automação e da robótica.

Por fim, serão sugeridos trabalhos futuros para explorar novas oportunidades de pesquisa e aprimoramento do robô cartesiano XYZ, visando maximizar seu potencial em diferentes contextos industriais e acadêmicos.

# 1 TEMA

Implementação de um robô cartesiano três eixos controlado via MatLab.

# 2 OBJETIVO GERAL

Desenvolver um sistema cartesiano de três eixos capaz de ser movimentado remotamente pelo MatLab, para uso acadêmico no Laboratório de Robótica.

# 3 OBJETIVOS ESPECÍFICOS.

- Projetar e construir um sistema cartesiano de três eixos (XYZ) ;
- Programar o CLP para controlar os movimentos do sistema cartesiano de três eixos;
- Desenvolver uma interface de comunicação entre o CLP e o software MatLab para movimentar o sistema em tempo real;
- Realizar testes e simulações para avaliar o desempenho do sistema cartesiano de três eixos e da interface com o MatLab;
- Analisar os resultados obtidos e discutir as limitações e possibilidades de utilização do sistema em estudos de controle de movimentos em três eixos.

# 4 DESIGN DE PESQUISA

O design de pesquisa utilizado foi o experimental, com o objetivo de testar a viabilidade e a eficácia de um protocolo de comunicação capaz de controlar o movimento do robô cartesiano XYZ desenvolvido.

## 4.1 Variáveis:

- Variável independente: implementação do protocolo de comunicação.
- Variável dependente: Desempenho do robô (precisão, velocidade de resposta).
- Variáveis controladas: Condições de teste (cenários e condições).

## 4.2 Procedimento:

- Implementação do protocolo de comunicação no robô.
- Elaboração de um plano de testes abrangente.
- Execução dos testes em diferentes cenários e condições.

- Coleta e análise dos dados.
- Interpretação dos resultados e formulação de conclusões.

4.3 Resultados Esperados:

- Demonstrar a viabilidade e a eficácia do robô cartesiano XYZ controlado via MatLab.
- Identificar áreas para aprimoramento do sistema.
- Contribuir para o avanço da automação e da robótica.

# 5 REFERENCIAL TEORICO

## 5.1 Robótica

Segundo MATARIC (2014), um robô é um sistema autônomo que existe no mundo físico, pode sentir o seu ambiente e pode agir sobre ele para alcançar alguns objetivos. (“Aula 01 – Robótica – Site do Professor Carlos Fernandes”) Robôs são mecanismos capazes de receber instruções e informações e com base nisto tomar ações autônomas, podemos dizer que robôs são sistemas autônomos que podem sentir e interagir com o seu ambiente.

**Figura 1** – Unimate, primeiro robô.

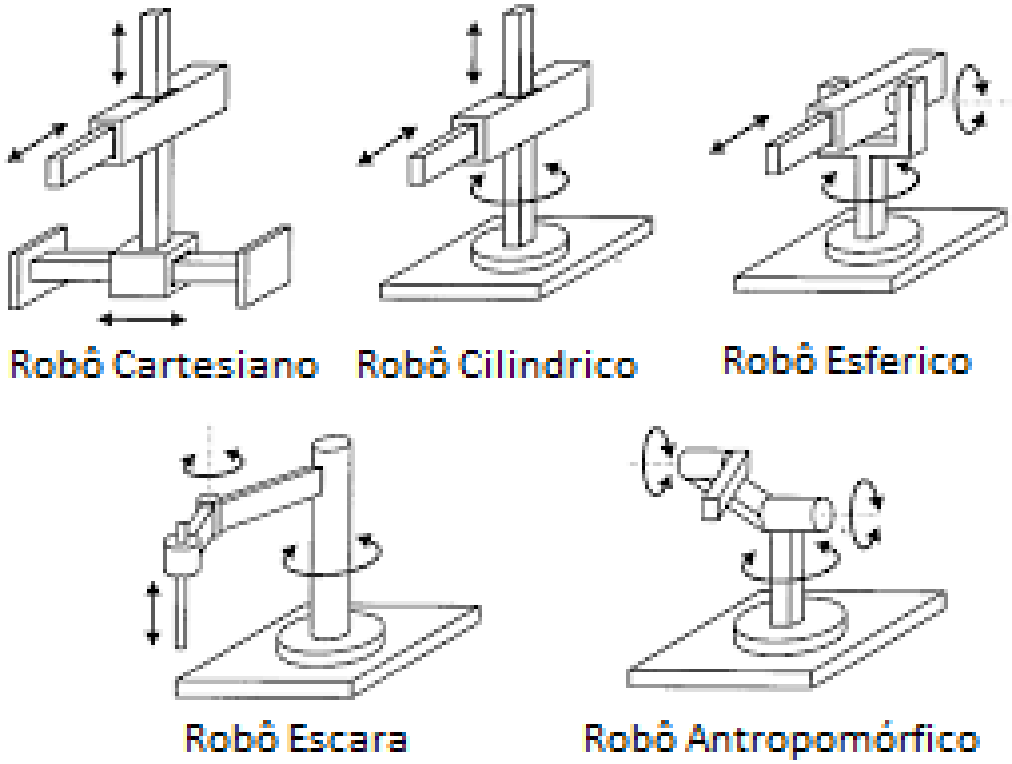
Fonte: Kawasaki Robotics

Com base nos requisitos do projeto desejado um robô deve apresentar algumas características específicas como: capacidade de se mover, de reconhecer o ambiente, de

manipular objetos, de interagir com o meio, de se comunicar, transmitir dados máquina-máquina, raciocinar, realizar atividades gerenciadas remotamente, de forma semiautônoma ou autônoma e capacidade de adequar o comportamento em razão ao ambiente que esteja inserido (FORNI,2017).

Existem diferentes tipos de robôs como mostrado na Figura 02 abaixo, cada um com suas características e funções específicas. quanto mobilidade da base, se são móveis ou fixos, quanto a estrutura cinemática, se são seriais ou paralelos, quanto aos seus graus de liberdade, ao seu tipo de acionamento, podendo ser elétrico, pneumático e hidráulico e ao seu espaço de trabalho (MATARIC, 2014).

**Figura 2** – Tipos de robô




Fonte: (BARRIENTOS et al, 1997).

- Robô Cartesiano

São robôs que possuem 3 juntas prismáticas, sendo assim o seu movimento coincidem com um sistema cartesiano em 3 eixos de translação.

- Robô Cilíndrico

São robôs que possuem dois movimentos de translação e uma rotação, fazendo com o robô possua duas juntas prismáticas e uma junta de revolução

- Robô Esférico.

Robô que os eixos formas um sistema de coordenada polar por possuir uma junta prismática e duas juntas de revolução

- Robô Scara

  Possui duas juntas de rotações paralelas e uma junta prismática perpendicular.

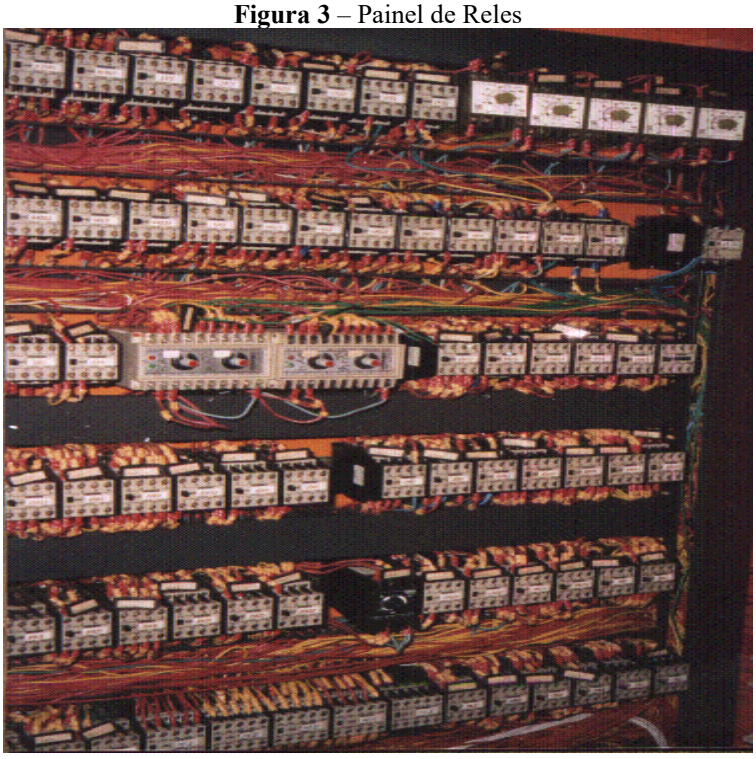
- Robô Antropomórfico

  Robô com um grande grau de liberdade, possuindo pelo menos 3 juntas de rotação,

## 5.2 Controladores Lógicos Programáveis (CLPs)

Os controladores lógicos programáveis (CLPs) são hoje a tecnologia de controle de processos industriais mais amplamente utilizada (PETRUZELLA, 2014). Eles foram desenvolvidos para substituir sistemas de controle mais antigos, que eram baseados em relés eletromecânicos e que eram muito limitados em termos de funcionalidade e flexibilidade como ilustrado na Figura 03.

**Figura 3** – Painel de Reles



Fonte: Internet

Segundo PETRUZELLA(2014):

> Ele é basicamente um computador digital projetado para uso no controle de máquinas, mas diferentemente de um computador pessoal, ele foi projetado para funcionar em um ambiente industrial e é equipado com interfaces especiais de entrada/saída e uma linguagem de programação de controle.

Os componentes básicos de um CLP incluem uma unidade central de processamento (CPU), memória para armazenar o programa, entradas e saídas digitais para se comunicar com sensores e atuadores, e interfaces de comunicação para se comunicar com outros dispositivos, como computadores ou outros CLPs como mostrado na figura 04.

**Figura 4** – Controlador Lógico Programável (CLP)


Fonte: Delta Eletronics

A vantagem dos CLPs é que eles oferecem uma solução de controle flexível e programável para uma ampla variedade de aplicações industriais. Eles são capazes de controlar processos complexos com velocidade e precisão, e podem ser facilmente reprogramados para atender às necessidades de diferentes aplicações.

## 5.3 Linguagem Ladder

Linguagem Ladder (também conhecida como LD, do inglês Ladder Diagram) é uma linguagem de programação de CLPs amplamente utilizada na automação industrial. Mesmo tendo sido a primeira linguagem destinada especificamente à programação de CLPs, a linguagem Ladder mantém-se ainda como a mais utilizada, estando presente em praticamente todos os CLPs disponíveis no mercado (GEORGINI, 2000).

Os símbolos na linguagem Ladder são desenhados como se fossem diagramas elétricos, com linhas verticais e horizontais que representam as entradas, saídas e os elementos lógicos do circuito. As entradas podem ser sensores, botões ou interruptores, enquanto as saídas podem ser motores, solenoides ou luzes, por exemplo. Os elementos lógicos são os blocos de instruções, que podem ser operações matemáticas, comparações, temporizadores, contadores, entre outros.

A linguagem Ladder é considerada fácil de aprender e programar, devido à sua semelhança com diagramas elétricos, o que facilita a interpretação do programa. Além disso, a linguagem é padronizada pela norma IEC 61131-3, o que garante a sua universalidade e compatibilidade com diversos fabricantes de CLPs.

Embora a linguagem Ladder seja eficiente para programação de CLPs, ela apresenta algumas limitações, como a falta de recursos avançados de programação, como orientação a objeto e manipulação de strings. Além disso, a programação em Ladder pode ser limitada por sua natureza gráfica, tornando-se confusa e difícil de manter em projetos complexos.

Apesar de suas limitações, a linguagem Ladder continua sendo uma das principais linguagens de programação de CLPs na indústria, devido à sua simplicidade, padronização e eficiência na programação de circuitos elétricos.

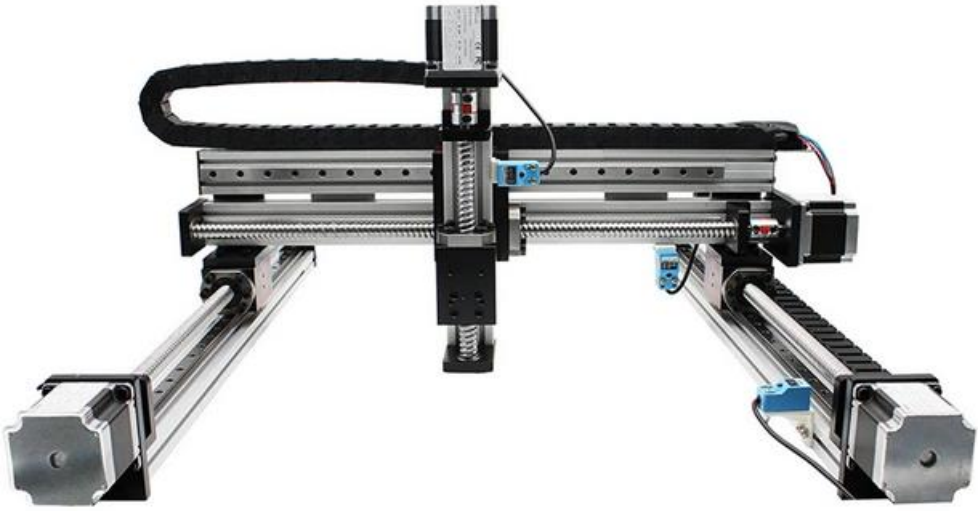
## 5.4 Sistemas cartesiano de três eixos.

Sistemas cartesianos de três eixos são dispositivos mecânicos usados para movimentar objetos em três dimensões (x, y, z) em relação a um sistema de coordenadas cartesianas. Eles consistem em três eixos: o eixo X, que se move horizontalmente de um lado para o outro, o eixo Y, que se move horizontalmente para frente e para trás, e o eixo Z, que se move verticalmente para cima e para baixo.

Esses sistemas são amplamente utilizados em aplicações industriais, como em máquinas CNC (Controle Numérico Computadorizado), em que a precisão é fundamental. Eles também são usados em aplicações de montagem, onde é necessário movimentar peças em diferentes posições, bem como em sistemas de inspeção, medição e posicionamento de objetos.

Os componentes básicos de um sistema cartesiano de três eixos incluem motores, guias lineares, fusos de esferas, correias e polias, além de uma estrutura que suporta e movimenta os componentes, como podemos ver na figura 05. Esses componentes são controlados por um software que envia sinais de controle para os motores para movimentar a plataforma nos três eixos.

**Figura 5 -** Sistema Cartesiano 3 eixos



Fonte: Internet

## 5.5 Servo Motores

São dispositivos eletromecânicos que convertem sinais elétricos em movimento mecânico, com alta precisão e controle, possuem a capacidade de manter uma posição fixa e realizar movimentos com grande precisão e velocidade. Esses equipamentos possuem um sensor que identifica a posição e velocidade do eixo e os envias para o controlador formando assim um sistema de retroalimentação.

Os servos motores são compostos por três componentes principais: o motor, o sensor e o controlador. O motor é responsável por transformar a energia elétrica em energia mecânica, movimentando o eixo do servo motor. O sensor é utilizado para enviar informações sobre a posição e velocidade do eixo do servo motor para o controlador. Já o controlador é responsável por receber as informações do sensor e enviar sinais elétricos para o motor, de forma a manter a posição e velocidade desejadas.

Em seguida é mostrado um exemplo de servo motor e servo drive (Figura 06).

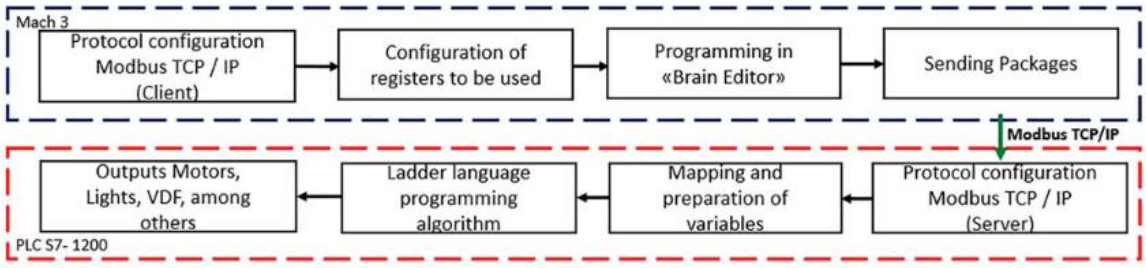
**Figura 6 -** Servo Motor, servo driver

Fonte: Internet

## 5.6 Protocolo TCP/IP

A comunicação eficaz entre os sistemas de controle é essencial para garantir o desempenho preciso e coordenado de máquinas CNC. Como destacado por Meza et al. (2018), a adaptação do software Mach3 a uma rede Modbus TCP/IP, juntamente com o CLP S7-1200, demonstrou ser uma solução viável para controlar uma fresadora CNC de três eixos. Essa arquitetura de comunicação permitiu resultados de alta precisão, tanto nos movimentos independentes dos eixos quanto nos movimentos simultâneos em três dimensões, contribuindo assim para o avanço da tecnologia de controle de movimento em máquinas CNC. O diagrama de bloco da implementação é mostrado na figura 07.

**Figura 7 –** Diagrama de blocos do procedimento de implementação




Fonte: Meza et al., (2018).

## 5.7 MELSEC COMMUNICATION PROTOCOL

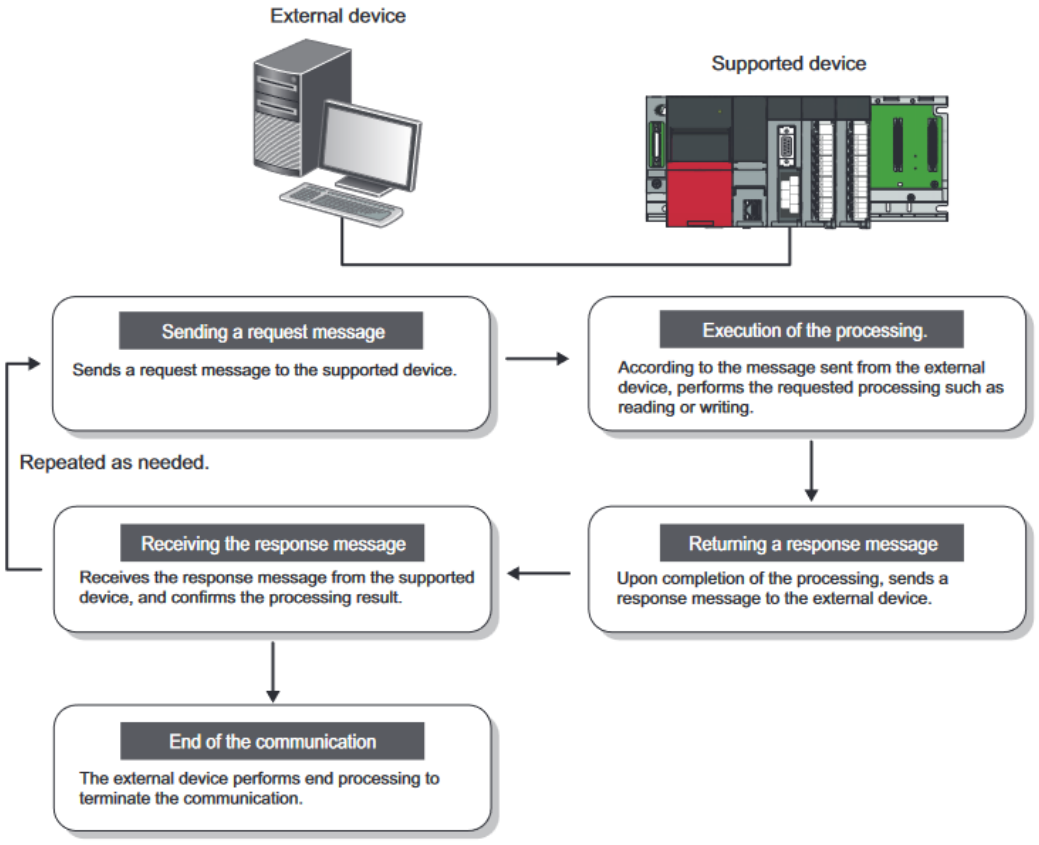
Segundo o MELSEC Communication Protocol Reference Manual da Mitsubishi, O protocolo de comunicação MELSEC (abreviado como MC protocol) é um protocolo de

comunicação para controladores programáveis MELSEC usado ao acessar um controlador programável de um dispositivo externo via C24 ou E71.

O protocolo de comunicação MELSEC (MC protocol) é responsável por facilitar a comunicação entre dispositivos externos e o controlador programável MELSEC. Quando uma mensagem de solicitação é enviada pelo dispositivo externo, ela é processada no módulo de CPU por meio de dispositivos compatíveis com o protocolo MC.

Após o processamento da mensagem no módulo de CPU, o resultado do processamento é então enviado de volta ao dispositivo externo como uma mensagem de resposta. Este processo de troca de mensagens permite uma comunicação bidirecional entre o controlador programável e dispositivos externos, garantindo uma interação eficiente e coordenada no sistema. Este fluxo de mensagens é ilustrado a seguir na figura 8

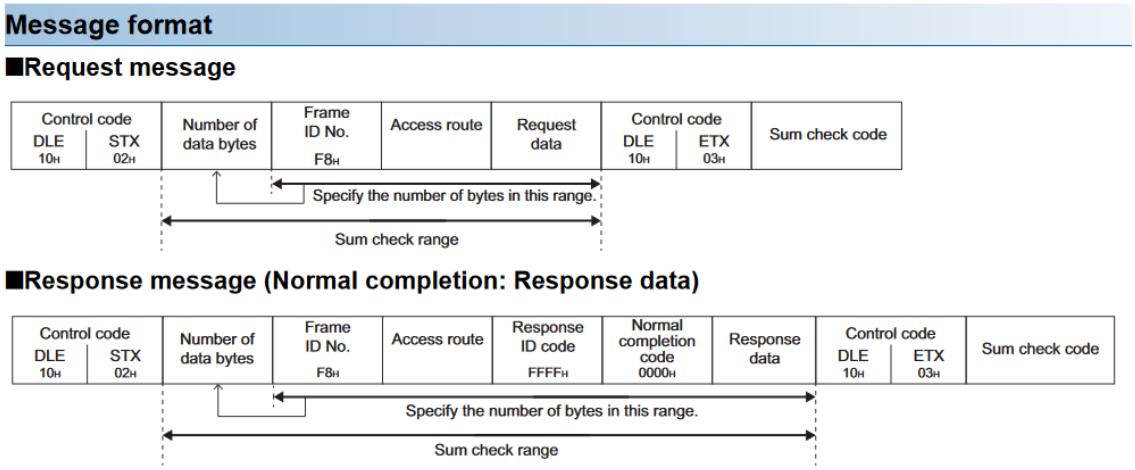
**Figura 8 -** Fluxo de mensagem no MC Protocol




Fonte: Mitsubishi.

O formato das mensagens são determinados pelo manual conforme a figura 9.

**Figura 9** - Formato das mensagens do MC Protocol




Fonte: Mitsubishi.

## 6 MATERIAIS E METODOS

O desenvolvimento do robô cartesiano foi estrategicamente dividido em três etapas cruciais para atingir o objetivo proposto: o design mecânico, a estruturação dos movimentos na CPU Motion e CLP, e a elaboração de scripts no MATLAB para a comunicação e controle dos movimentos do robô.

### 6.1 Design Mecânico:

Na fase inicial, uma análise criteriosa dos materiais disponíveis foi conduzida para a montagem do robô. A lista de materiais incluiu perfis de alumínio, base MDF, guias lineares, atuadores verticais e lineares da SMC, além de motores, servo drivers, CPUs e expansões de entrada/saída, descritos na tabela abaixo

Tabela 01 – Materiais mecânicos utilizados para montagem do sistema cartesiano.

| MATERIAL | QUANTIDADE |
|---|---|
| Perfil80x80x1200 | 4 |
| Perfil80x80x570 | 4 |
| Perfil80x80x570 | 4 |
| Chapa 830x600x10 | 1 |

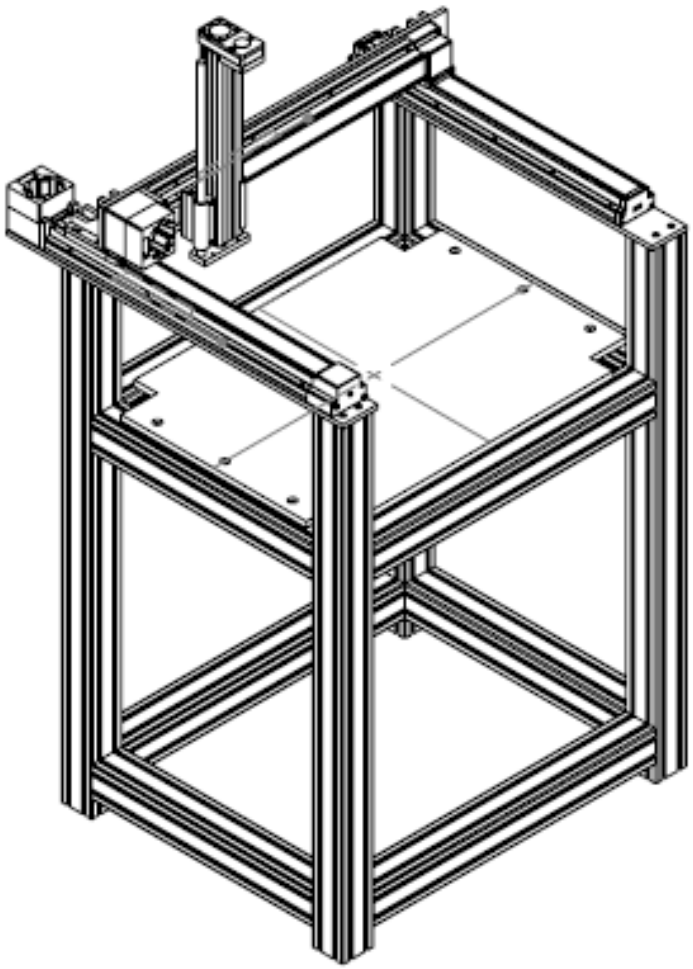
| | |
|---|---|
| Chapa 90x870x10 | 2 |
| Chapa 95x895x10 | 1 |
| Acoplamento XY | 2 |
| Motor HG-KR43 K 400W 3000r/min | 3 |

Fonte: Autor, 2023

. Essa escolha cuidadosa visou otimizar recursos, aproveitando materiais já disponíveis no laboratório e evitando gastos desnecessários.

A utilização do software SOLIDWorks para criar o desenho 3D do robô permitiu aproveitar os perfis já cortados no laboratório, minimizando a necessidade de novas peças e reduzindo custos. Após a conclusão do modelo 3D mostrado na figura 10, a documentação 2D em anexo foi elaborada para uma possível alteração posterior de projeto.

**Figura 10** - Modelo 3D do robô cartesiano



Fonte: Autor, 2023.

A montagem do equipamento foi realizada com precisão como evidenciado na figura 11, utilizando ferramentas adequadas, parafusos, porcas e cantoneiras para fixar os perfis metálicos. O resultado foi uma estrutura robusta e confiável, demonstrando uma abordagem prática e eficiente no desenvolvimento mecânico do robô.

**Figura 11 - Sistema Cartesiano em Montagem parcial**

**Fonte: Autor, 2023.**

## 6.2 Estruturação dos Movimentos no CLP e na CPU Motion:

A seguir, apresenta-se uma lista dos materiais utilizados no processo de estruturação dos movimentos no CLP e na CPU Motion, juntamente com suas respectivas descrições:

Tabela 2: Materiais Elétricos Utilizados

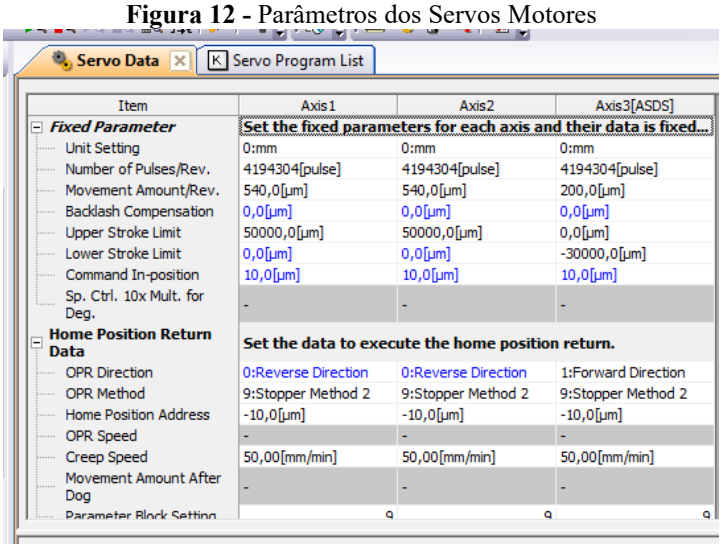
| MATERIAL | DESCRIÇÃO |
|---|---|
| CLP | Q03UDECPU |
| CPU | Q172DSCPU |
| SERVO DRIVER | MR-JJ4W3-44B |
| Cartão de Entrada | QX10 |
| Cartão de Saída | QY10 |

Fonte: Autor, 2023

Esses materiais foram selecionados com base nas especificações técnicas necessárias para garantir um funcionamento eficiente e coordenado do robô durante o processo de movimentação.

Nesta etapa crucial, a configuração dos motores no servo driver foi iniciada com a definição dos valores de resolução e a conversão precisa dos valores de rotação para deslocamento em milímetros nos atuadores lineares(figura 12). Este procedimento é essencial para garantir uma correspondência precisa entre os comandos do sistema e os movimentos físicos do robô.

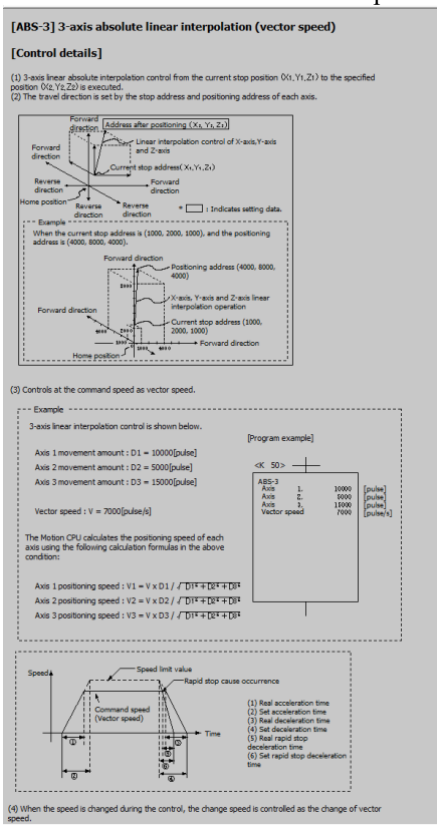
**Figura 12 -** Parâmetros dos Servos Motores




**Fonte: MR Developer Mitsubishi, 2023**

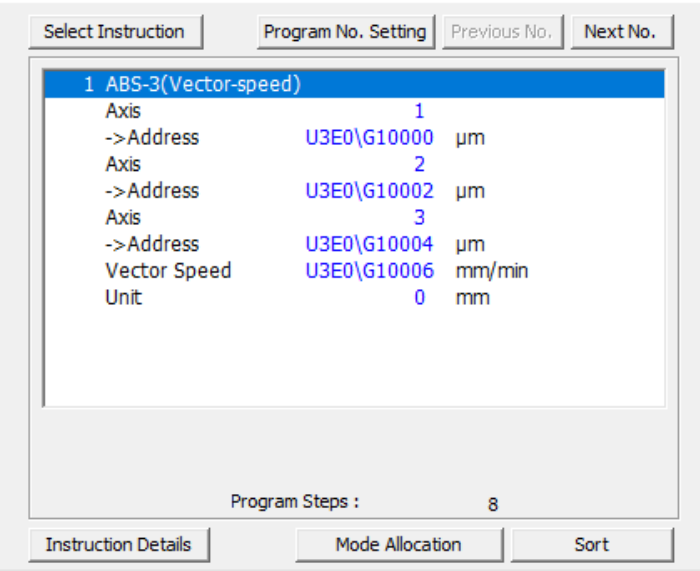
Após essa etapa, foi criada uma função padrão de movimentação dos servos na CPU Motion (figura 14), utilizando o software MR DEVELOPER da Mitsubishi. Nesse processo, foi selecionado o perfil de movimento linear interpolado absoluto (figura 13), que se mostrou mais adequado à aplicação por já realizar os cálculos de velocidade entre os eixos diretamente na CPU MOTION, assegurando uma operação eficiente e coordenada do robô.

**Figura 13 –** Perfil de movimento linear interpolado absoluto




**Fonte: MR Developer Mitsubishi, 2023**

**Figura 14** – Servo Program com os endereços dos valores a serem lidos.

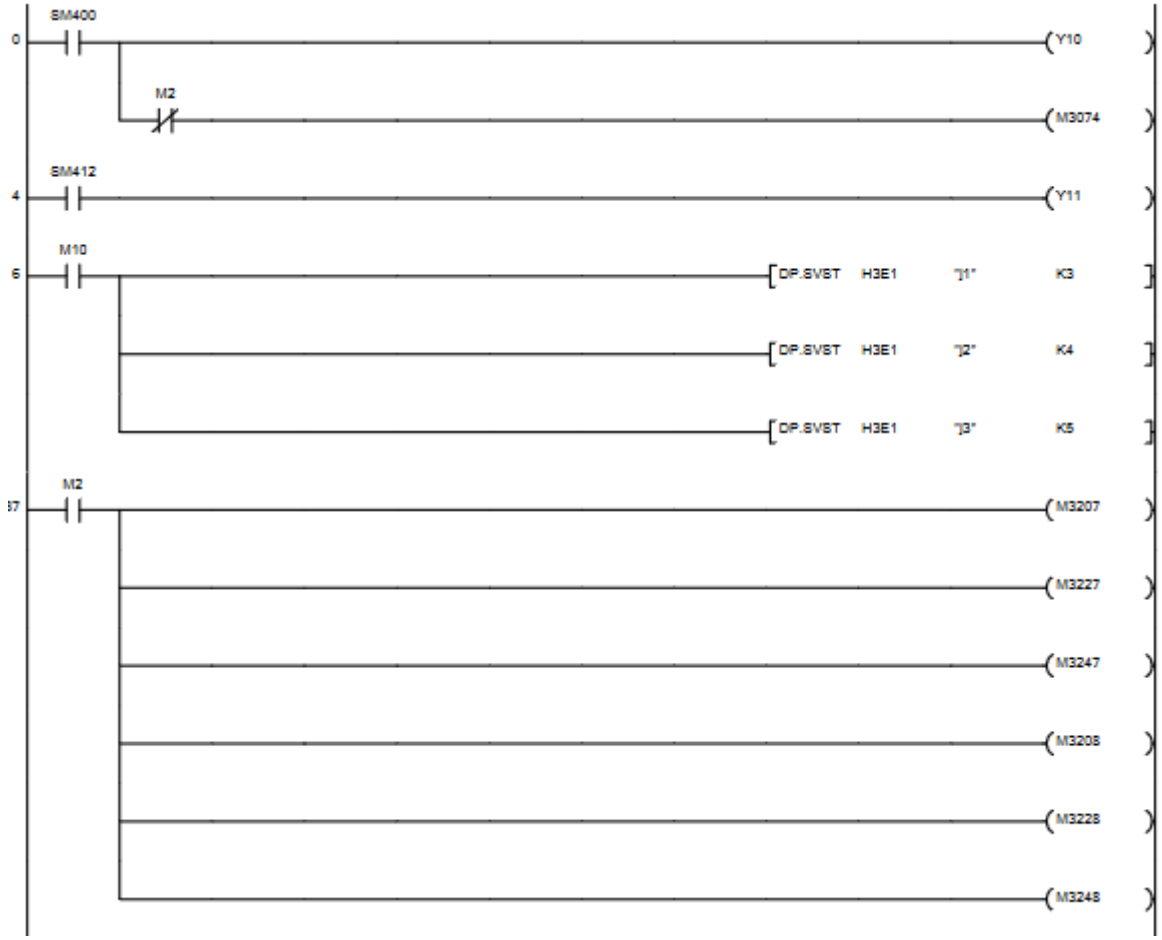



Fonte: MR Developer Mitsubishi, 2023.

Dentro do CLP, foram desenvolvidos dois arquivos em ladder para a rotina do robô. No primeiro arquivo, uma lógica foi elaborada para realizar o reset dos servo drivers ao acionar a memória “M2”, removendo-os de possíveis erros e reiniciando a comunicação. Adicionalmente, foram inseridas linhas de código para executar a operação de HOME dos servos ao acionar a

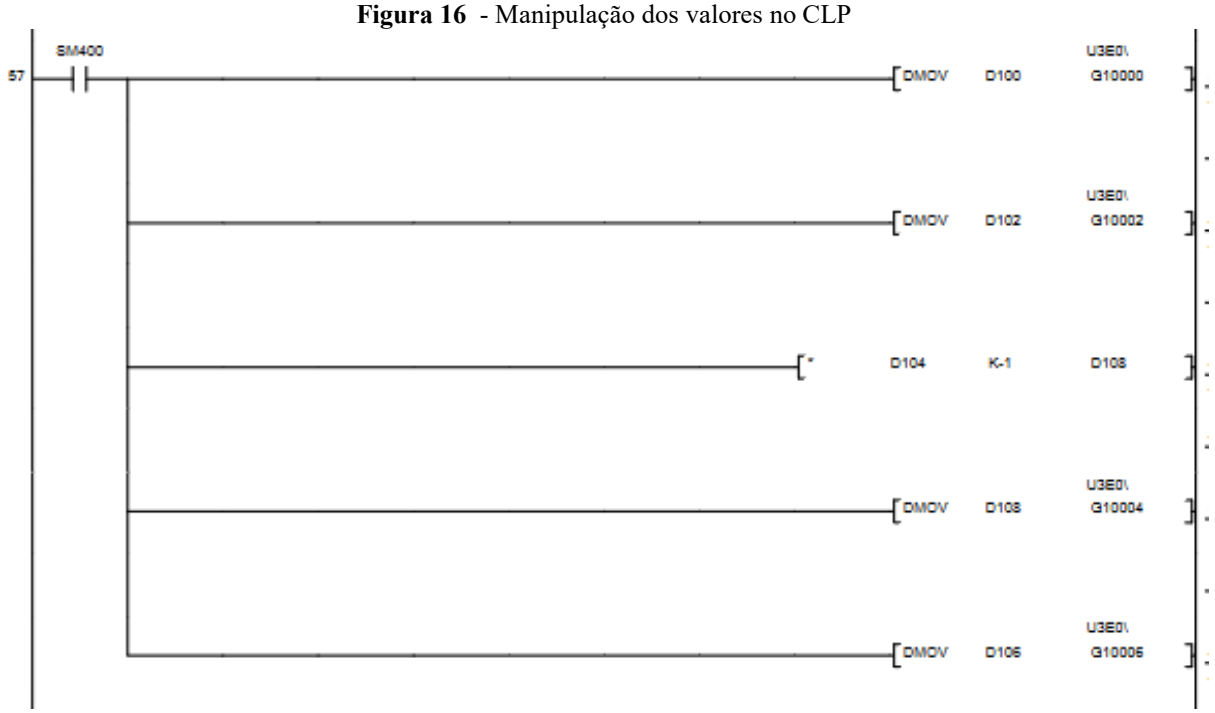
memória “M10”, estabelecendo assim a posição inicial dos motores, como mostrado na figura 15.

**Figura 15** - Lógica Ladder RESET e HOME




Ainda no primeiro arquivo, uma lógica foi desenvolvida para facilitar a comunicação com o MATLAB e processar os dados enviados pelo MATLAB para o CLP. Os dados de movimentação são escritos em 4 registradores do CLP que usando o relé especial SM400 (sempre ligado quando o CLP está em modo RUN), os valores dos 4 registradores são movidos para os parâmetros do programa de movimentação da CPU MOTION descrita anteriormente, os valores do eixo 3, responsável pelo movimento em vertical “Z”, passa por um tratamento no CLP modificando o valor de positivo para negativo como mostrado na figura 16.

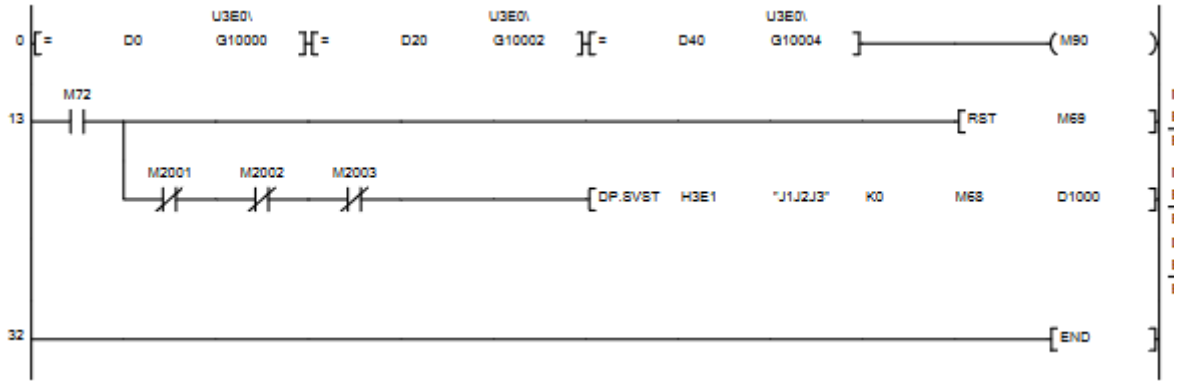
**Figura 16** - Manipulação dos valores no CLP




Fonte: Autor, 2023.

No segundo arquivo foi adicionado duas linhas de códigos figura 17, na primeira uma memória (m90) é acionada sempre que os valores dos encoders dos motores são iguais aos valores nos parâmetros do programa de movimentação que foi mostrado na figura 14, na segunda linha temos ativação do programa de movimentação através de uma memória (m72), através da função DP.SVST, que tem como parâmetros o a porta da CPU Motion (H3E1), os eixos que serão atualizados no caso J1J2J3, o índice do programa na CPU motion e uma memoria de ativação quando a movimentação termina M68.

**Figura 17** - Ladder da função de movimentação




Fonte: Autor, 2023

## 6.3 Elaboração de Scripts no MATLAB para a Comunicação e Controle:

O último desafio enfrentado consistiu na implementação da comunicação entre o MATLAB e o CLP, utilizando o MELSEC Communication Protocol (MC Protocol) da mitsubishi, que é o protolo

Para iniciar, o CLP foi configurado para receber e enviar informações externas utilizando o protocolo MC protocol. A porta 2500 do CLP foi configurada para o protocolo TCP, a configuração de comunicação foi alterada para código binário para facilitar a conversão dos valores no MATLAB.

Posteriormente, desenvolveu-se o script READ_BIT para leitura de memórias do CLP que é mostrado na figura 18. Inicialmente, estabelece-se uma conexão TCP/IP com o dispositivo, utilizando o endereço IP e a porta especificados. Em seguida, cria-se uma estrutura de comunicação contendo informações cruciais, como cabeçalho, comprimento, comando e endereço do dispositivo. O quadro é, então, enviado ao CLP através da conexão TCP/IP. Após um breve intervalo para permitir a resposta do dispositivo, os dados recebidos são analisados para determinar o status da operação. Se o último byte da resposta indicar que a operação foi bem-sucedida, a função retornará verdadeiro; caso contrário, retornará falso. Por fim, a conexão TCP/IP é encerrada e o status é retornado como saída da função.

**Figura 18 -**Código Fonte script READ_BIT

```
function [status] = READ_BIT(a)
    myTCPConnection = tcpclient('192.168.1.38', 2502);
    % Define the frame structure
    frame.header = [hex2dec('50'), 0, 0, hex2dec('ff'),
hex2dec('ff'), 3, 0];
    frame.length = [hex2dec('0c'), 0];
    frame.timer = [hex2dec('20'), 0];
    frame.command = [1, 4];
    frame.sub_command = [1, 0];
    frame.start_addr = [a, 0, 0];
    frame.device = hex2dec('90');
    frame.points = [1, 0];
    frame.length  =  [length(uint8(struct2array(frame)))-
9, 0];
    write(myTCPConnection, uint8(struct2array(frame)));
    pause(0.2);
    data = read(myTCPConnection, 12);
    if data(12)== 16
        status = true
        clear myTCPConnection
        return
    else
        status = false
        clear myTCPConnection
        return
```

Fonte: Autor, 2023.

O próximo script implementado foi a função ON_BIT, projetada para ativar um bit específico no CLP conectado via TCP/IP, evidenciado na figura 19. Esta função aceita um argumento de entrada, representando o endereço do bit a ser ativado no CLP. Inicialmente, estabelece-se uma conexão TCP/IP com o dispositivo, utilizando o endereço IP e a porta especificados. Em seguida, constrói-se uma estrutura de comunicação contendo informações essenciais, como cabeçalho, comprimento, comando e endereço do dispositivo, além de especificar a ativação do bit. Posteriormente, o quadro é enviado ao CLP através da conexão TCP/IP. Após o envio do quadro, a função exibe uma mensagem indicando a ativação do bit e, então, encerra a conexão TCP/IP.

**Figura 19** – Código fonte script ON_BIT

```
function ON_BIT(a)
    myTCPConnection = tcpclient('192.168.1.38', 2502);
    frameMW.header = [80, 0, 0, 255, 255, 3, 0];
    frameMW.length = [14, 0];
    frameMW.timer = [32, 0];
    frameMW.command = [1, 20];
    frameMW.sub_command = [0, 0];
    frameMW.start_addr = [a, 0, 0];
    frameMW.device =  hex2dec('90');
    frameMW.points = [1, 0];
    frameMW.w_data = [1, 0];
    frameMW.length  =  [length(uint8(struct2array(frameMW)))-
9, 0];
    write(myTCPConnection, uint8(struct2array(frameMW)));
    display "ONBIT"
    clear myTCPConnection
end
```

Fonte: Autor, 2023.

Na sequência, desenvolveu-se a função OFF_BIT demonstrado na figura 20, destinada a desativar um bit específico no CLP. Assim como nos scripts anteriores, esta função aceita um argumento de entrada representando o endereço do bit que será desativado no dispositivo remoto. Estabelece-se uma conexão TCP/IP com o dispositivo, utilizando o endereço IP e a porta especificados. Em seguida, constrói-se uma estrutura de comunicação semelhante às anteriores, porém especificando a desativação do bit. Posteriormente, o quadro é enviado ao dispositivo remoto através da conexão TCP/IP.

**Figura 20** – Código fonte script OFF_BIT

```
      function OFF_BIT(a)
           myTCPConnection = tcpclient('192.168.1.38', 2502);
          frameMW.header = [80, 0, 0, 255, 255, 3, 0];
          frameMW.length = [14, 0];
          frameMW.timer = [32, 0];
          frameMW.command = [1, 20];
          frameMW.sub_command = [0, 0];
          frameMW.start_addr = [a, 0, 0];
          frameMW.device =  hex2dec('90');
          frameMW.points = [1, 0];
          frameMW.w_data = [0, 0];
          frameMW.length  = [length(uint8(struct2array(frameMW)))-
9, 0];
          write(myTCPConnection, uint8(struct2array(frameMW)));
          display "OFFBIT"
          clear myTCPConnection
      end
```

Fonte: Autor, 2023.

Outro script desenvolvido é a função Send_REG, projetada para enviar um valor para um registrador específico no CLP descrito na figura 21. Esta função aceita dois argumentos de entrada, a e b, representando o endereço do registrador e o valor a ser enviado, respectivamente. Inicialmente, estabelece-se uma conexão TCP/IP com o dispositivo, utilizando o endereço IP e a porta especificados. Em seguida, constrói-se uma estrutura de comunicação contendo informações essenciais, como cabeçalho, comprimento, comando e endereço do dispositivo, bem como o valor a ser enviado para o registrador especificado. Após a construção do quadro de comunicação, o script envia o quadro ao dispositivo remoto através da conexão TCP/IP utilizando a função write. Finalmente, a conexão TCP/IP é encerrada.

**Figura 21** – Código fonte script Send_REG

```
      function Send_REG(a,b)
          myTCPConnection = tcpclient('192.168.1.38', 2502);
          frame1.header = [80, 0, 0, 255, 255, 3, 0];
          frame1.length = [14, 0];
          frame1.timer = [32, 0];
          frame1.command = [1, 20];
          frame1.sub_command = [0, 0];
          frame1.start_addr = [a, 0, 0];
          frame1.device =  hex2dec('a8');
          frame1.points = [1, 0];
          frame1.w_data = [b,1];
          frame1.length                                          =
[length(uint8(struct2array(frame1)))-9, 0];
          write(myTCPConnection, uint8(struct2array(frame1)));
          clear myTCPConnection
        end
```

Fonte: Autor, 2023.

Finalmente, foi criado um script no MATLAB para enviar as informações necessárias ao robô executar um movimento, mostrado na figura 22. Essa função foi denominada de "MOVE", exigindo quatro parâmetros: “b, c e d”, que correspondem às coordenadas X, Y e Z, respectivamente. O parâmetro “e” representa a velocidade do movimento.

Antes de iniciar o movimento, a função verifica se as coordenadas especificadas estão dentro dos limites físicos do robô. Se alguma coordenada estiver fora desses limites, a função exibirá "Ponto Inválido" e encerrará a execução.

Em seguida, verifica-se se a velocidade especificada está dentro do intervalo permitido. Se a velocidade estiver fora desse intervalo, a função exibirá "Velocidade Inválida" e encerrará a execução.

Caso as coordenadas e a velocidade estejam dentro dos limites permitidos, a função iniciará o movimento do robô. Primeiramente, convertem-se as coordenadas e a velocidade para o formato adequado para comunicação com o CLP. Em seguida, estabelece-se uma conexão TCP/IP com o CLP.

Após estabelecer a conexão, são montados e enviados três quadros de comunicação para o CLP:

- O primeiro quadro configura os registradores de posição dos eixos XYX para a coordenada ‘b,c,d’ e a velocidade na variável ‘e’.
- O segundo quadro ativa o bit de início de movimento no CLP.
- O terceiro quadro é utilizado para verificar se o movimento foi concluído.

Após o envio do quadro de verificação, a função entra em um loop de espera, onde verifica-se periodicamente se o movimento foi concluído. Assim que o movimento for concluído, a função encerra a conexão e retorna.

**Figura 22** – Código fonte script MOVE2

```
      function MOVE2(b,c,d,e)
          if or(or((b > 500), (b < 0)), or(or((c < 0),(c > 500)), or((d < 0), (d
> 300))))
              display("Ponto Invalido")
              return
          end
          if or(e<1,e>500)
              display    ("Velocidade Invalida")
              return
          end
          display("Movendo para")
          disp([b,c,d]);
          b=b*100+400;
          c=c*100+400;
          d=d*100+300;
```

```
            e=e*100;
            b = [rem(b,256), floor(b/256)];
            c = [rem(c,256), floor(c/256)];
            d = [rem(d,256), floor(d/256)];
            e = [rem(e,256), floor(e/256)];

            myTCPConnection = tcpclient('192.168.1.38', 2502);

            frame1.header = [80, 0, 0, 255, 255, 3, 0];
            frame1.length = [14, 0];
            frame1.timer = [32, 0];
            frame1.command = [1, 20];
            frame1.sub_command = [0, 0];
            frame1.start_addr = [100, 0, 0];
            frame1.device =  hex2dec('a8');
            frame1.points = [8, 0];
            frame1.w_data                                                    =
[c(1),c(2),0,0,b(1),b(2),0,0,d(1),d(2),0,0,e(1),e(2),0,0];
            frame1.length = [length(uint8(struct2array(frame1)))-9, 0];

            write(myTCPConnection, uint8(struct2array(frame1)));

            frameMW.header = [80, 0, 0, 255, 255, 3, 0];
            frameMW.length = [14, 0];
            frameMW.timer = [32, 0];
            frameMW.command = [1, 20];
            frameMW.sub_command = [0, 0];
            frameMW.start_addr = [72, 0, 0];
            frameMW.device =  hex2dec('90');
            frameMW.points = [1, 0];
            frameMW.w_data = [1, 0];
            frameMW.length = [length(uint8(struct2array(frameMW)))-9, 0];

            write(myTCPConnection, uint8(struct2array(frameMW)));

            pause(0.1)
            frameMW.w_data = [0, 0];

            write(myTCPConnection, uint8(struct2array(frameMW)));

            frame.header = [hex2dec('50'), 0, 0, hex2dec('ff'), hex2dec('ff'), 3,
0];
            frame.length = [hex2dec('0c'), 0];
            frame.timer = [hex2dec('20'), 0];
            frame.command = [1, 4];
            frame.sub_command = [1, 0];
            frame.start_addr = [90, 0, 0];
            frame.device = hex2dec('90');
            frame.points = [1, 0];
            frame.length = [length(uint8(struct2array(frame)))-9, 0];

            while true
                myTCPConnection = tcpclient('192.168.1.38', 2502);
                write(myTCPConnection, uint8(struct2array(frame)));
                pause(0.05);
                data = read(myTCPConnection, 12);
                if data(12)== 16
                    clear myTCPConnection
                    return
                end
                clear myTCPConnection
            end
        end
```

Fonte: Autor, 2023.

Por fim foram criadas mais duas funções, a função HOME_Robot e RESET_Robot, mostrado na figura 23, para movimentar os eixos até suas posições de origem e para resetar os motores do sistema respectivamente.

**Figura 23 -** Código fonte das funções HOME e RESET

```
function HOME_Robot
    ON_BIT(10)
    pause(1)
    OFF_BIT(10)
end

function RESET_Robot
    ON_BIT(2)
    pause(1)
    OFF_BIT(2)
end
```

Fonte: Autor, 2023.

# 7 RESULTADOS E DISCUSSÃO

Nesta seção, são apresentados os resultados obtidos durante o desenvolvimento do robô cartesiano XYZ controlado via MATLAB. Os resultados são discutidos em relação aos objetivos propostos, destacando as conquistas alcançadas, os desafios enfrentados e as possíveis melhorias para trabalhos futuros.

## 7.1 Implementação do Robô Cartesiano

O robô cartesiano foi implementado conforme as especificações de design e requisitos técnicos estabelecidos, como evidenciado na figura 24. Durante o processo de desenvolvimento, foram consideradas as limitações físicas e técnicas dos materiais e componentes utilizados. A estrutura mecânica do robô foi montada com sucesso, garantindo uma base sólida e confiável para os movimentos controlados.

**Figura 24**– Robô XYZ montado

Fonte: Autor, 2024

## 7.2 Controle e Comunicação

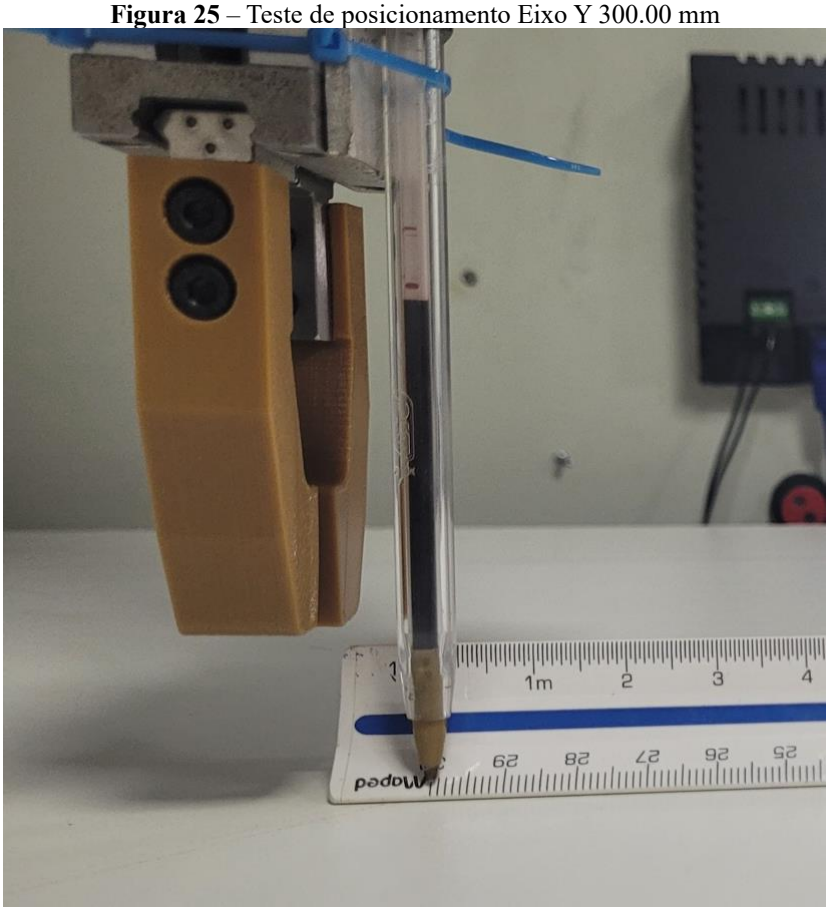
Os scripts desenvolvidos para controle e comunicação do robô foram eficazes na coordenação dos movimentos nos eixos X, Y e Z. A comunicação entre o MATLAB e o Controlador Lógico Programável (CLP) foi consistente, permitindo o envio de comandos precisos para o robô, com o tempo de resposta aceitável e sem perda de dados nos pacotes enviados.

## 7.3 Testes de Funcionamento

Foram realizados testes para validar a operação do robô. Os resultados demonstraram um desempenho satisfatório em cumprir as tarefas propostas, dentro dos limites especificados.

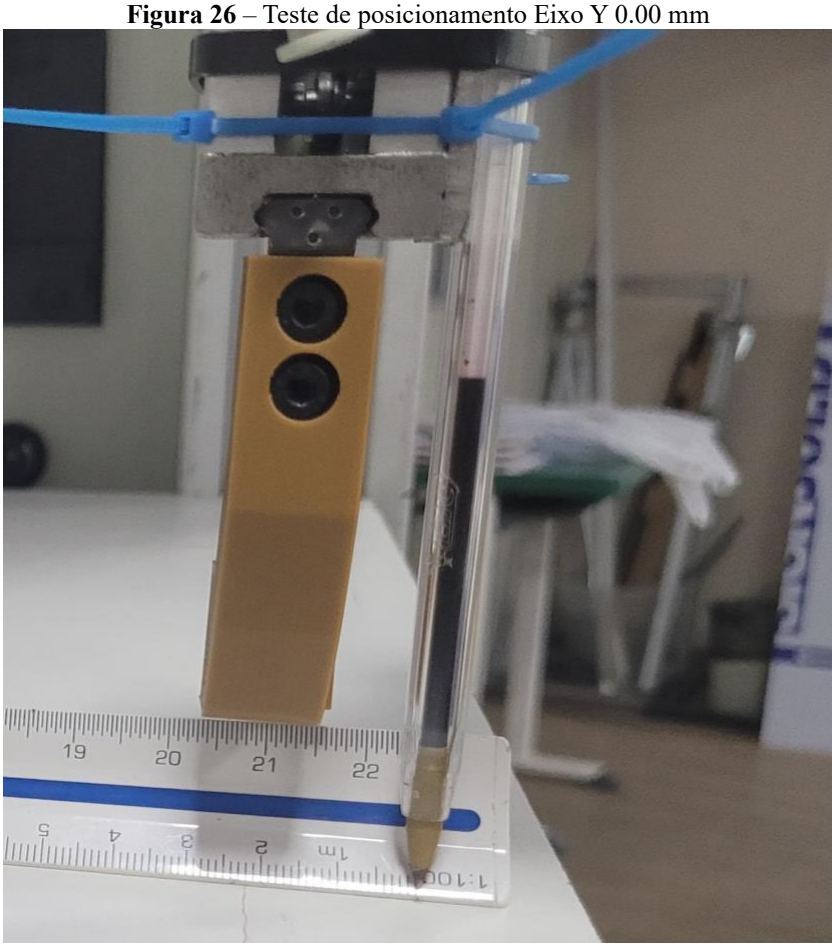
Nas figuras 25 e 26 podemos ver os testes de posicionamento do eixo Y, nas figuras 27, 28 e 29 os teste de posicionamento do eixo Z e por fim nas figuras 30,31 e 32 os teste de posicionamento do eixo X, para o teste foi utilizado um escalimetro de 300mm na escala 1:100 para comparamos as posições relativas e saber se ao incrementar a quantidade de movimento o sistema real executaria o determinado valor.

**Figura 25** – Teste de posicionamento Eixo Y 300.00 mm



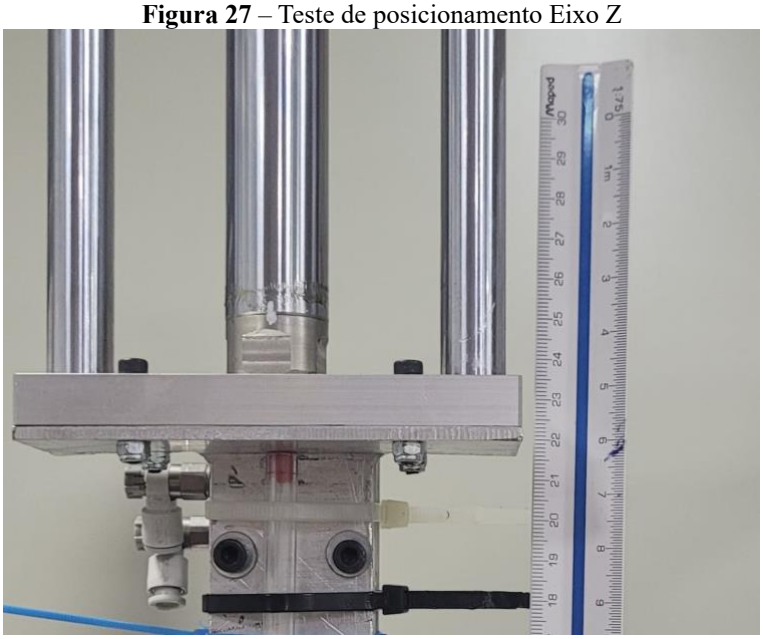

Fonte: Autor, 2024.

**Figura 26** – Teste de posicionamento Eixo Y 0.00 mm



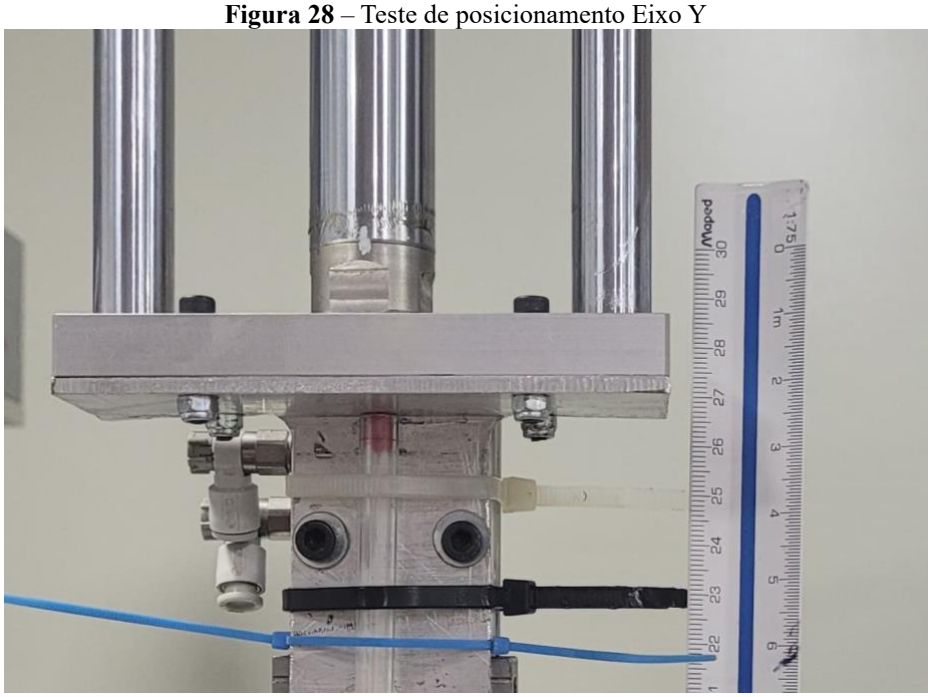

Fonte: Autor, 2024

**Figura 27** – Teste de posicionamento Eixo Z



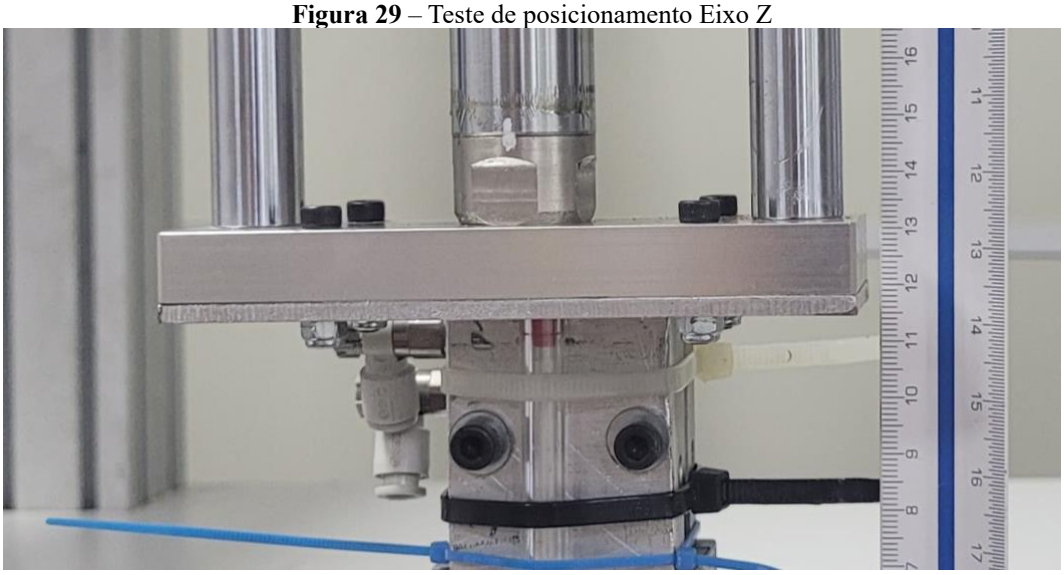
Fonte: Autor, 2024.

**Figura 28** – Teste de posicionamento Eixo Y



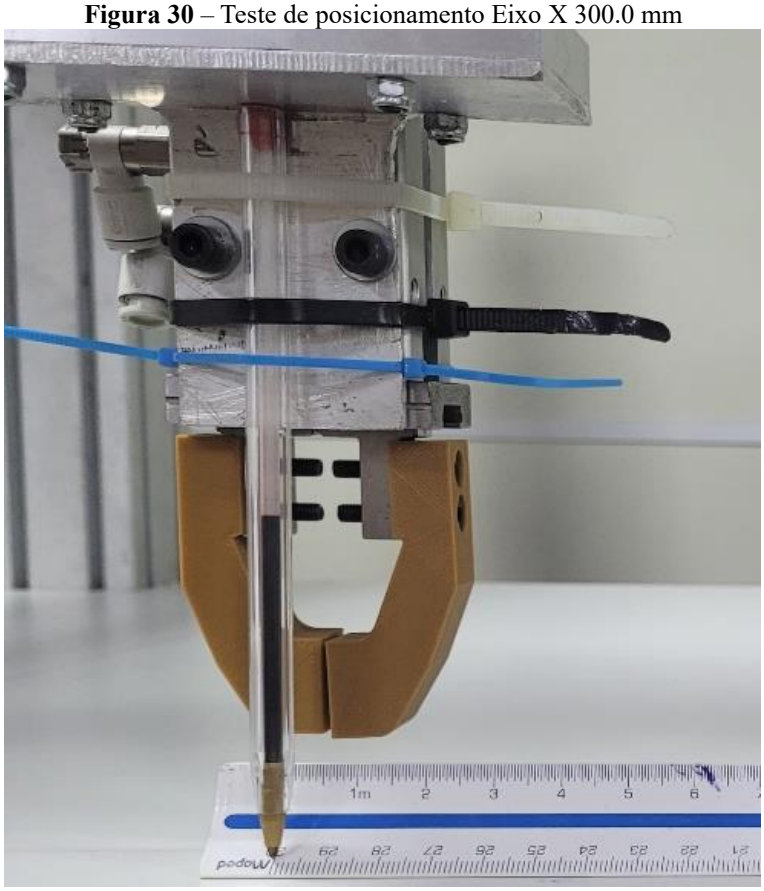
Fonte: Autor, 2024.

**Figura 29** – Teste de posicionamento Eixo Z



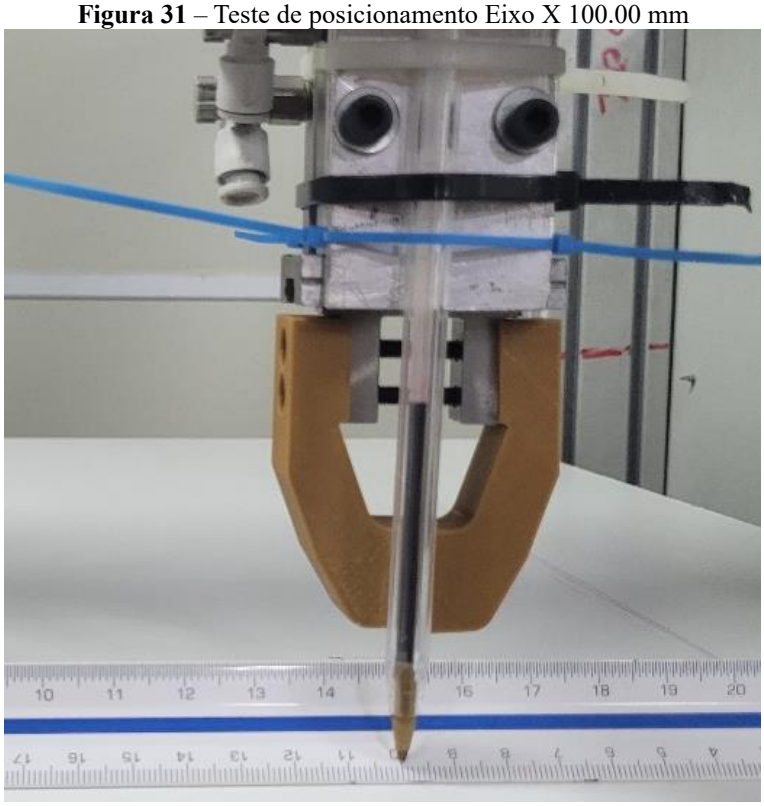
Fonte: Autor, 2024

**Figura 30** – Teste de posicionamento Eixo X 300.0 mm



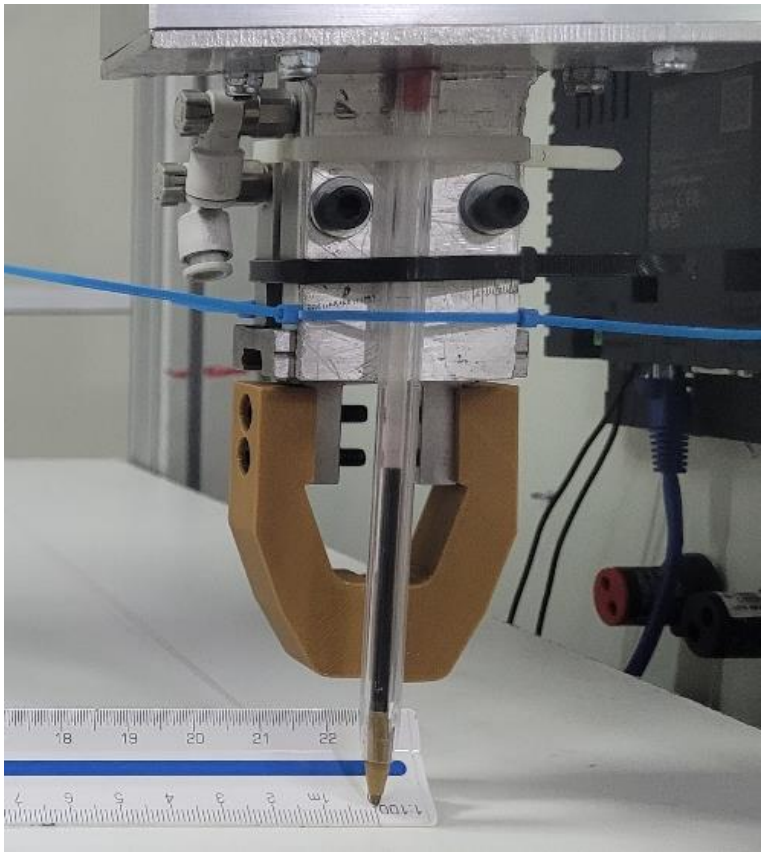
Fonte: Autor, 2024.

**Figura 31** – Teste de posicionamento Eixo X 100.00 mm



Fonte: Autor, 2024.

**Figura 32 -** Teste de posicionamento Eixo X 0.00 mm



Fonte: Autor, 2024.

## 7.4 Análise de Resultados

A análise dos resultados revelou que o robô cartesiano foi capaz de realizar os movimentos planejados com precisão e consistência. As medições realizadas durante os testes confirmaram a conformidade do robô com as especificações de projeto. No entanto, foram identificadas algumas áreas que podem ser aprimoradas, principalmente no que diz respeito ao desenvolvimento mecânico do dispositivo.

## 7.5 Discussão

Durante os testes de funcionamento, observamos uma perda de precisão no posicionamento do eixo Y ao movimentar-se em velocidades superiores a 250 unidades. Esse fenômeno foi atribuído à inércia do sistema, especialmente relacionada ao guia linear que não possui motor. Notou-se uma espécie de "overshoot" durante esses movimentos, indicando a necessidade de melhorias futuras. Uma possível abordagem para otimizar esse aspecto seria substituir o guia linear atual por um atuador linear com motor, o qual poderia ser sincronizado com os componentes existentes no projeto. Essa modificação tem o potencial de reduzir

significativamente os efeitos da inércia e melhorar a precisão do posicionamento do eixo Y em velocidades mais altas.

Os resultados obtidos corroboram a viabilidade e a eficácia do sistema de controle desenvolvido para o robô cartesiano XYZ. A integração entre o MATLAB e o CLP permitiu uma interação fluida e robusta, possibilitando o controle preciso do robô em tempo real. As limitações identificadas durante os testes fornecem insights valiosos para futuras melhorias e refinamentos no sistema. Além disso, os resultados destacam o potencial de aplicação do robô em uma variedade de contextos industriais e acadêmicos, contribuindo para avanços significativos na automação e na robótica.

# 8 CONSIDERAÇÕES FINAIS

O desenvolvimento do robô cartesiano XYZ controlado via MATLAB representou uma etapa significativa na busca por conhecimento em automação e robótica. Ao longo deste estudo, foram realizados esforços significativos para projetar, implementar e testar um sistema capaz de controlar o movimento preciso em três dimensões, com o objetivo de demonstrar sua viabilidade e eficácia.

Os resultados obtidos revelaram uma implementação bem-sucedida do robô cartesiano, com a estrutura mecânica montada de forma precisa e confiável, proporcionando uma base sólida para os movimentos controlados nos eixos X, Y e Z. Os testes de funcionamento demonstraram um desempenho satisfatório do robô em cumprir as tarefas propostas, dentro dos limites especificados.

Além disso, os resultados obtidos destacam a importância da integração entre o MATLAB e o Controlador Lógico Programável (CLP), que permitiu uma interação fluida e robusta, possibilitando o controle preciso do robô em tempo real. Essa colaboração entre diferentes plataformas e tecnologias é essencial para impulsionar avanços significativos na automação e na robótica, abrindo novas possibilidades de aplicação em uma variedade de contextos industriais e acadêmicos.

Em suma, este estudo representa um passo importante na jornada em direção a sistemas robóticos mais avançados e eficientes. Os insights obtidos durante este processo fornecem uma base sólida para futuras pesquisas e desenvolvimentos.

# 9 . TRABALHOS FUTUROS

Apesar dos resultados promissores obtidos durante o desenvolvimento do robô cartesiano XYZ controlado via MATLAB, existem várias oportunidades para pesquisas e melhorias adicionais. Abaixo estão algumas sugestões para trabalhos futuros:

## 9.1 Aprimoramento Mecânico:

- Investigações mais detalhadas sobre o sistema mecânico do robô, incluindo análises de vibração e estabilidade, podem ajudar a identificar e corrigir possíveis fontes de imprecisão nos movimentos, como o fenômeno de "overshoot" observado no eixo Y.
- Explorar diferentes materiais e métodos de fabricação para otimizar a estrutura do robô, visando aumentar a rigidez e reduzir o peso, o que pode contribuir para melhorias no desempenho geral do sistema.

## 9.2 Controle e Algoritmos:

- Desenvolver algoritmos avançados de controle de movimento para minimizar os efeitos da inércia e melhorar a precisão em altas velocidades de deslocamento, especialmente no eixo Y.
- Explorar técnicas de controle adaptativo que possam ajustar dinamicamente os parâmetros do sistema em resposta a mudanças nas condições operacionais, garantindo um desempenho otimizado em diferentes cenários.

## 9.3 Interface do Usuário e Visualização:

- Desenvolver uma interface de usuário mais intuitiva e amigável para facilitar a programação e operação do robô, permitindo que usuários com diferentes níveis de experiência possam interagir com o sistema de forma eficaz.
- Implementar ferramentas de visualização 3D para acompanhar e analisar os movimentos do robô em tempo real, proporcionando uma compreensão mais clara do comportamento do sistema e facilitando a depuração de possíveis problemas.

## 9.4 Aplicações Específicas:

- Explorar aplicações específicas para o robô cartesiano XYZ em diferentes setores industriais, como fabricação, automação de processos, montagem de componentes

eletrônicos, entre outros, identificando oportunidades para otimização e personalização do sistema de acordo com as necessidades do usuário final.

Essas são apenas algumas das muitas direções que podem ser exploradas em trabalhos futuros relacionados ao desenvolvimento e aprimoramento do robô cartesiano XYZ.

## 10 REFERÊNCIAS

BHALERAO, Abhiraj et al. Pick and Place Robotic ARM using PLC. **Int. J. Eng. Res. Technol**, v. 8, n. 08, p. 667-670, 2019.

FORNI, Andrea. **ROBÔS - A NOVA ERA: Vivendo, trabalhando e investindo nasociedade robótica do futuro.** ("Robôs - A Nova Era. Vivendo, trabalhando e investindo na sociedade ...") [S. l.: s. n.], 2017.

GEORGINI, Marcelo. **Automação Aplicada - PLCs**. São Paulo: Érica, 2000.

KULA, Ahmed; GEORGIEVA, Anna; RUZHEKOV, Georgi. Modelling and Control of Rectification Column. In: **2022 22nd International Symposium on Electrical Apparatus and Technologies (SIELA)**. ("2022 22nd International Symposium on Electrical Apparatus and ...") IEEE, 2022. p. 1-4.

LI, Xueru; MENG, Fanwen; ZHENG, Xuan. Automatic Control System of Sluice Based on PLC, MCGS and MODBUS Communication. (In: **2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC)**. ("2021 7th Annual International Conference on Network and Information ...") IEEE, 2021. p. 716-720.

LOZANO, Jesús et al. Sistema de monitorización y control de un robot cartesiano basado en PLC. In: **XXXVII Jornadas de Automática**. Comité Español de Automática, 2016. p. 1030-1036.

MATARIĆ, Maja J. **Introdução à robótica**. Editora Blucher, 2014.

Meza, Gonzalo; Carpio, Christian del; Vinces, Nikolai; Klusmann, Mirko CONTROL of a three-axis CNC machine using PLC S7 1200 with the Mach3 software adapted to a Modbus TCP/IP network. **IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)** , [*S. l.*], p. 1-4, 8 ago. 2018.

MITSUBISHI. **MELSEC Communication Protocol Reference Manual**. [*S. l.*: *s. n.*], 2023. SH-080008.

PETRUZELLA, Frank D. **Controladores lógicos programáveis**. AMGH Editora, 2014.