Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

ZÓZIMO PEREIRA GUEDES DA SILVA

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

Campina Grande, Paraíba

Junho de 2016

ZÓZIMO PEREIRA GUEDES DA SILVA

RELÁTÓRIO DE ESTÁGIO SUPERVISIONADO

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de concentração: Controle e Automação

Orientador:

Professor George Acioli Júnior, Dr, Sc.

Campina Grande, Paraíba

Junho de 2016

ZÓZIMO PEREIRA GUEDES DA SILVA

RELÁTÓRIO DE ESTÁGIO SUPERVISIONADO

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de concentração: Controle e Automação

Aprovado em _____ / _____ / _____

_____

Professor Dr. George Acioli Júnior

Universidade Federal de Campina Grande

Orientador

_____

Professor Avaliador

Universidade Federal de Campina Grande

Avaliador

AGRADECIMENTO

Agradeço aos Professor Péricles Rezende Barros e George Acioli Júnior, pela oportunidade de estagiar no LIEC e pelas orientações prestadas não só durante a realização estágio mas durante a maior parte do curso.

Agradeço a minha família, em especial a minha mãe Hélia Pereira, meu pai Iolando Guedes, a minha tia Ivalmira Guedes e a amiga Adail Silva por apoiarem fortemente a realização desse sonho.

E por fim, a todos os colegas do Laboratório de Instrumentação Eletrônica e Controle, em especial a Alequine Batista de Lima, Lucas José da Silva Moreira e Simões Toledo, cuja ajuda foi de grande importância para a realização desse trabalho.

RESUMO

Este relatório apresenta as atividades realizadas pelo aluno Zózimo Pereira Guedes da Silva durante o Estágio Supervisionado no Laboratório de Instrumentação e Eletrônica (LIEC), pertecente ao Departamento de Engenharia Elétrica (DEE) da Universidade Federal de Campina Grande (UFCG) sobre orientação do Professor George Acioli Júnior e supervisão do Professor Péricles Rezende Barros.

Palavras-chave: Braço robótico, AX-12A, dynamixel, MatLab GUI, OPC.

LISTA DE FIGURAS

Figura 01 – LIEC 12

Figura 02 – Característica física e pinagem do AX-12A 14

Figura 03 – Comunicaçao entre controlador e 1 AX-12A 15

Figura 04 – Comunicação entre controladores e vários AX-12A 15

Figura05 – Braço robótico inteligente AX-12A 16

Figura 06 – Dynamixel 17

Figura 07 – Ligação PC-dynamixel-AX-12A 17

Figura 08 – Seleção de modo 18

Figura 09 – Conexão 3P 18

Figura 10 – Conexão de alimentação 18

Figura 11 – Pinagem do dynamixel 19

Figura 12 – Representação de um manipulador com a RVC Tools 22

Figura 13 – Notação Denavit-Hartenberg 23

Figura 14 – Representação do braço robótico inteligente AX-12A com a RVC Tools 24

Figura 15 – Modelo geométrico do braço robótico 26

Figura 16 – Posições alcancáveis pelo braço robótico 27

Figura 17 – Comparação entre o método geométrico e o Fuzzy 28

Figura 18 – Interface gráfica 29

Figura 19 – Ambiente de simulação V-REP 34

Figura 20 – Configuração OPC 37

Figura 21 – Modelo da base 38

Figura 22 – Modelo do ombro 38

Figura 23 – Modelo do cotovelo 38

Figura 24 – Leituras e referência para o servo da base 40

Figura 25 – Leituras e referência para o conjunto de servos do ombro 40

Figura 26 – Leituras e referência para o conjunto de servos do cotovelo 41

LISTA DE TABELAS E QUADROS

TABELAS

Tabela 01 – Especificações do AX-12A 14

Tabela 02 – Resumo da notação DH 23

SUMÁRIO

AGRADECIMENTOS.....IV

RESUMO V

LISTA DE FIGURAS VI

LISTA DE TABELAS E QUADROS VIII

SUMÁRIO 09

1. Introdução 10

1.1 O Plano de Estágio 10

1.1.1 Principais atividades 10

1.1.2 Cronograma 11

2. Local do Estágio 12

3. Atividades Realizadas 13

3.1 Visão Geral do Braço Robótico Inteligente AX-12A 13

3.2 Biblioteca Dynamixel SDK 19

3.3 RVC Tools 21

3.4 Cinemática inversa utilizando geometria e lógica Fuzzy 25

3.5 Elaboração da interface gráfica 28

3.6 V-REP 29

3.7 Comunicação OPC 36

3.8 Modelagem e identificação 37

3.9 Controle 42

4. Melhorias sugeridas 42

5. Conclusão 42

6. Bibliografia 43

1. Introdução

O objetivo deste relatório é apresentar as atividades desenvolvidas e os resultados alcançados pelo estagiário Zózimo Pereira Guedes da Silva, durante o estágio supervisionado no Laboratório de Instrumentação e Controle (LIEC), situado na Universidade Federal de Campina Grande (UFCG). O período de vigência do estágio supervisionado foi de 01/02/2016 até 27/05/2016 totalizando 180h (6 créditos), sendo uma carga horária de 12h semanais, sob a orientação do Professor George Acioli Júnior e supervisão do Professor Péricles Rezende Barros.

1.1 O Plano de Estágio

A elaboração do plano de estágio foi concebida de forma a possibilitar a concepção do Braço Robótico Inteligente AX-12A com fins de identificação e controle de seus motores, requerendo do estagiário a aplicação do conhecimento adquiridos nas diversas disciplinas integralizadas durante a graduação, tendo uma maior aplicação dos temas abordados nas disciplinas de Introdução a Programação, Técnicas de Programação, Eletrônica, Arquitetura de Sistemas Digitais, Controle Analógico e Controle Digital.

1.1.1 Principais atividades

1. Visão geral do Braço Robótico Inteligente AX-12A;
2. Estudo da biblioteca Dynamixel SDK;
3. Estudo e uso da ferramenta RVC Tools;
4. Inclusão da comunicação OPC
5. Cinemática Inversa utilizando geometria e lógica fuzzy;
6. Elaboração da Interface Gráfica;
7. Estudo e uso do software V-REP;
8. Comunicação OPC;
9. Modelagem e identificação dos servos;
10. Implementação dos Controladores P, PI e PID;

1.1.2 Cronograma

Atividades	Fevereiro				Março				Abril				Maio			
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																

2. Local do Estágio

O Laboratório de Instrumentação e Controle (LIEC) localiza-se na Universidade Federal de Campina Grande, campus de Campina Grande e pertence ao Departamento de Engenharia Elétrica (DEE), cujo corpo técnico é formado de Professores Doutores, alunos de graduação e pós-graduação, tendo como objetivo principal desenvolver atividades e projetos ligados a áreas de instrumentação, automação e controle.

Figura 1 – LIEC - Fonte: http://liec.ufcg.edu.br/, acessado em 10/06/2016

Com uma área de aproximadamente 600 m², o LIEC (Figura 1) conta com oito laboratórios de desenvolvimento, duas salas de apoio técnico, sala para apresentação de trabalhos, salas para pós-graduação e professores. Dentre as principais atividades desenvolvidas no laboratório, pode-se destacar as seguintes (LIEC, 2015):

- Laboratório de Aplicações Wireless - desenvolvimento de soluções baseadas em dispositivos móveis para ambientes industriais;
- Laboratório de Automação Industrial - sintonia de controladores PID industriais (mono e multivariável), automação industrial, instrumentação industrial, IHM industrial, avaliação de confiabilidade em malhas de controle;

- Laboratório de Controle e Otimização - projeto e sintonia de PID, modelagem e simulação de processos, sistemas supervisórios;
- Laboratório de Instrumentação Eletrônica - projeto e sintonia de PID;
- Laboratório de Redes Industriais - estudos de técnicas e tecnologias para a comunicação entre dispositivos industriais;
- Laboratório de RFID - desenvolvimento de aplicações baseadas em tecnologia RFID para ambientes industriais;
- Laboratório de Ultrassom - desenvolvimento de sensor de incrustação, desenvolvimento de técnicas de medição de incrustação.

No LIEC, alunos de pós-graduação e de graduação encontram um ambiente propício ao aperfeiçoamento dos conhecimentos teóricos e das habilidades práticas, por intermédio das diversas pesquisas e outras atividades realizadas no mesmo.

3. Atividades Realizadas

3.1 Visão Geral do Braço Robótico Inteligente AX-12A

Antes de descrever a estrutura do Braço Robótico faz-se necessário descrever o atuador da série Dynamixel AX-12A.

AX-12A

O AX-12A trata-se de um atuador Dynamixel inteligente, modular que incorpora um redutor de velocidade, um motor de precisão DC e um circuito de controle com a funcionalidade de rede, tudo em um único pacote. Apesar do seu tamanho compacto, o AX-12A pode produzir um binário elevado e é feito com materiais de elevada qualidade de forma a proporcionar a resistência e capacidade de resistência estrutural necessária para suportar grandes forças externas. Além disso, o AX-12A tem a capacidade de detectar e agir de acordo com as condições internas, como mudanças na temperatura interna ou tensão de alimentação.

Possui um controle de precisão de 1024 níveis com faxa de posicionamento de 300º (i.e.,uma resolução de 0.2930 graus/nível) possibilidade de trabalhar com velocidades de comunicação de até 1Mb.

Tabela 01 – Especificações do AX-12A

	AX-12A	
Peso(g)	55	
Taxa de redução	1/254	
Tensão de entrada	Em 7V	Em 10V
Torque Max (kgf.cm)	12	16.5
Resolução	0.2930º	
Faxa de operação	0 a 300º	
Alimentação	7V ~ 10V(Recomendado de 9.6V)	
Corrente Máxima	900mA	
Temperatura de oper.	-5ºC ~ +85ºC	
Tipo de Protocolo	Serial Half-Duplex Assín. (8bit,1parada,sem paridade)	
ID	254 (0 ~ 254 ID)	
Velocidade de comunicação	7343bps ~ 1Mbps	

Para comunicar-se estabelece uma conexão serial TTL do tipo Half-Duplex, em que o AX-12A identificado por um ID único atua como um servidor que recebe comandos e retorna respostas ao controlador externo (PC). Na figura a seguir. é possível contemplar a geometria do AX-12A assim como a disposição de seus terminais.

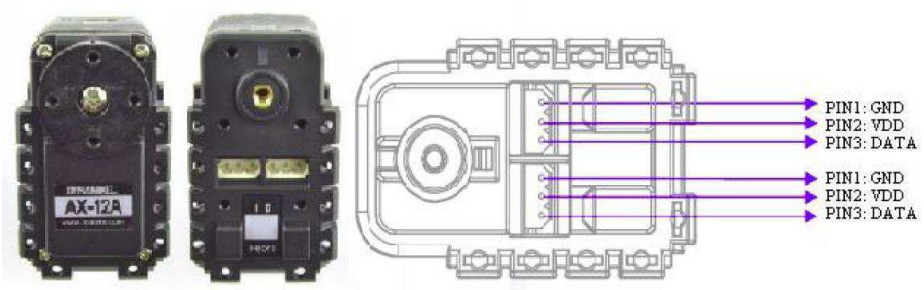


Figura 02 – Característica física e pinagem do AX-12A

Na comunicação o controlador principal comunica-se com os Dynamixel enviando e recebendo pacotes de informação. Existem dois tipos de pacotes que são os “Pacotes de Instrução” (enviados do controlador principal para os Dynamixels) e os “Pacotes de Status” (enviados do Dynamixel para o controlador principal).

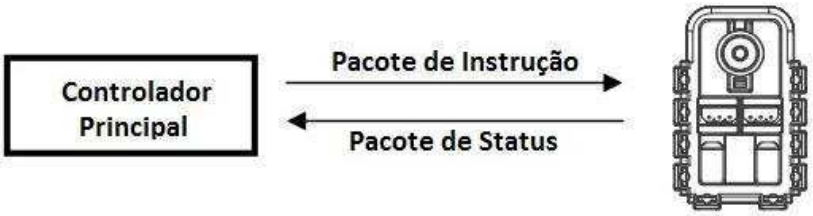


Figura 03 – Comunicação entre controlador e 1 AX-12A.

Para o sistema abaixo, se o controlador principal envia um pacote de instrução com um ID do conjunto N, o Dynamixel com esse valor de ID retorna o respectivo pacote de status e realiza a instrução requerida.

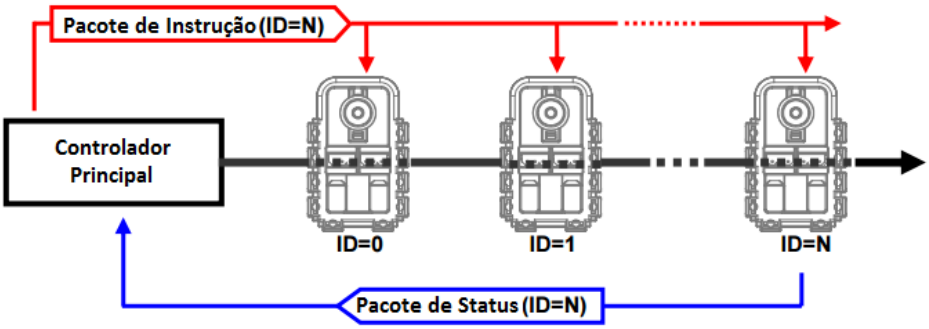


Figura 04 – Comunicação entre o controlador e vários AX-12A.

Se mais de uma unidade do Dynamixel possui o mesmo ID então haverá envio simultâneos de pacote gerando problemas de comunicação.

Braço Robótico Inteligente AX-12A

O braço inteligente foi projetado para ser forte e resistente. Construindo em alumínio galvanizado, detém 7 atuadores do tipo AX-12A de forma a proporcionar 4 graus de liberdade além da garra. Quatro dos AX-12A estão

emparelhados de forma a constituirem as articulações do punho e ombro. As demais articulações possuem apenas um único Ax-12A. Na base existem quatro esferas de aço para a facilitar o trato com cargas mais pesadas e uma articulação ajustável manualmente. A garra possui um prolongamento de maneira a permitir a conexão de câmeras, sensores de pressão, toque ou outros sensores. Há outras garras opcionais disponíveis no mercado fábricadas pela CrustCrawler.

No que tange o comprimento do braço, pode-se dizer que possui 53,78 cm de extensão máxima e garra com abertura máxima de 6.35 cm. Para a fixação de sua base utilizou-se uma estrutura feita em madeira garantindo assim uma boa estabilidade.

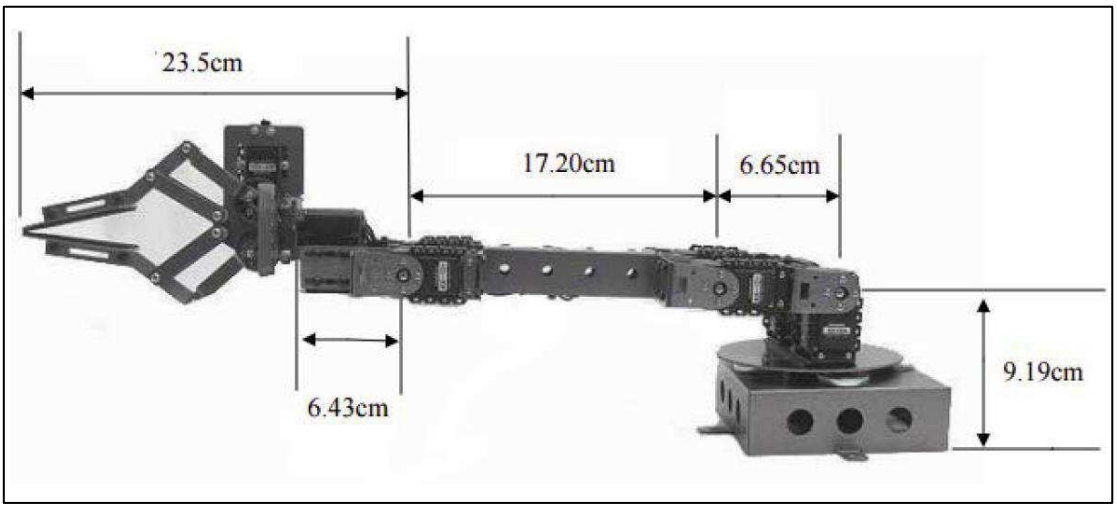


Figura 05 – Braço robótico inteligente AX-12A

USB2Dynamixel

O dispositivo utilizado para controlar os dynamixels presentes no braço robótico foi o USB2Dynamixel a partir da porta USB de um computador. Tal dispositivo possui conexão 3P para dynamixels da série AX assim como 4P para dispositivos das séries DX e RX.

Além disso, o USB2Dynamixel pode ser utilizada para transformar uma porta USB em uma porta serial para um PC sem uma porta serial tal como em

um notebook. As figuras a seguir mostram o aspecto real e como usar o USB2Dynamixel

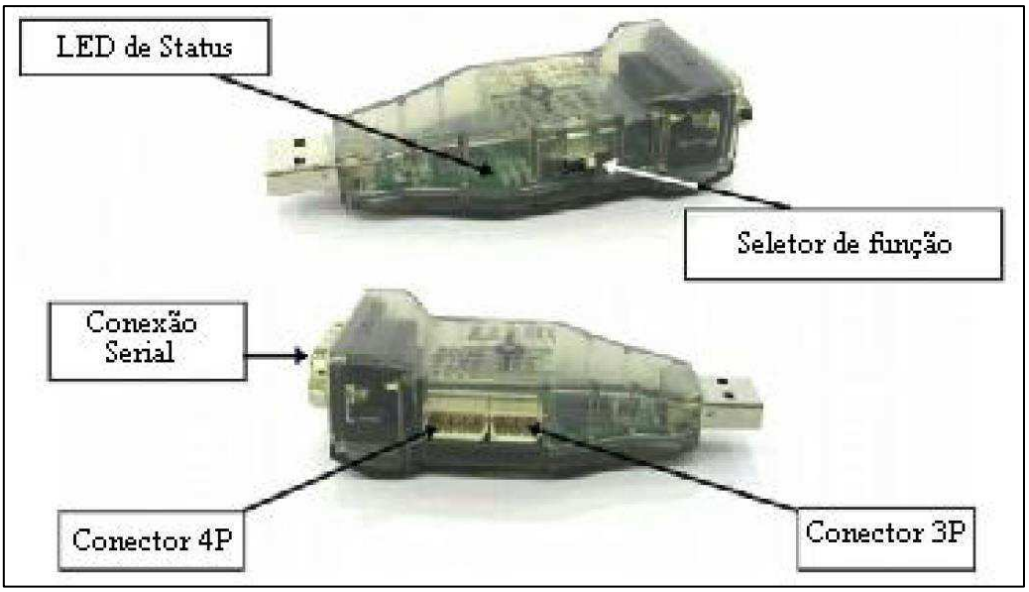


Figura 06 - Dynamixel

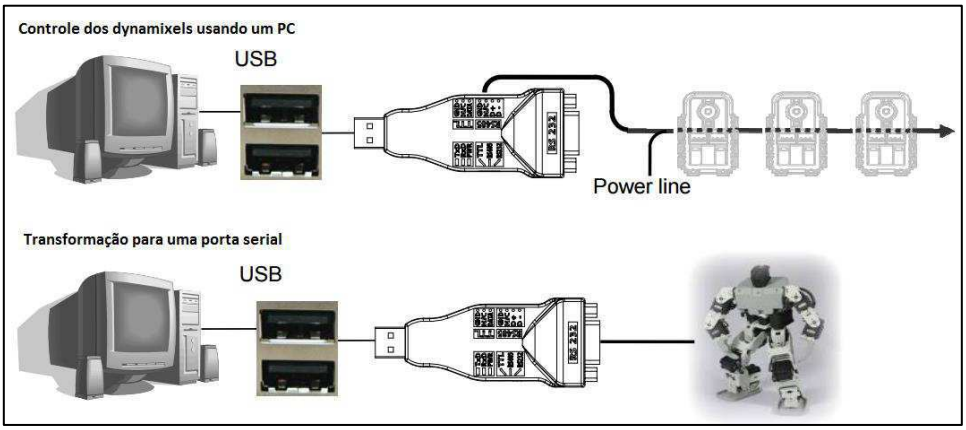


Figura 07 – Ligação PC-dynamixel-AX-12A

Uma vez que os atuadores utilizados foram os da serie AX-12A passaremos a descrição de como foi realizado a conexão entre o PC e os dynamixels utilizando o USB2Dynamixel.

1º Passo - Seleciona-se o modo TTL conforme figura a seguir:

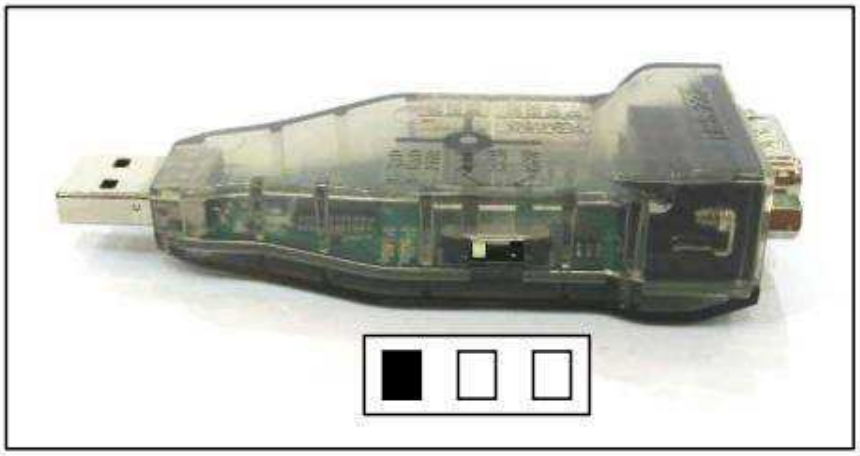


Figura 08 – Seleção de modo

2º Passo – Conecta-se o cabo 3P ao conector 3P do USB2Dynamixel e ao do dynamixel (o AX-12A possui dois conectores, escolhe-se qualquer um dos dois)

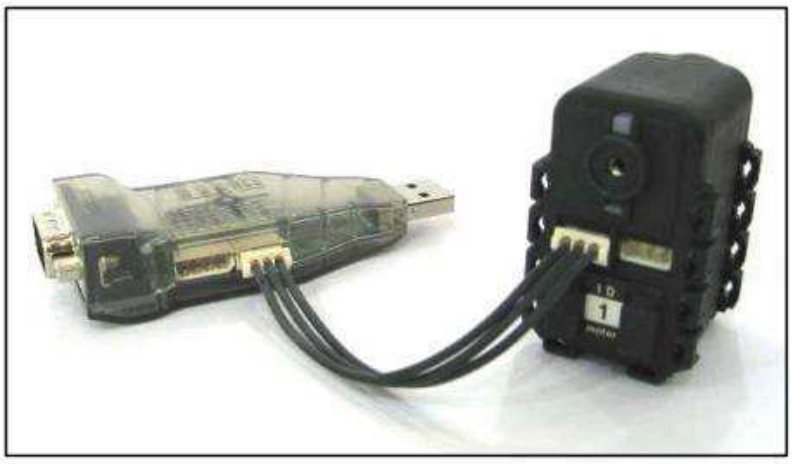


Figura 09 – Conexão 3P

3º Passo – Os dynamixels podem ser conectados em série usando um cabo 3P, conectando a alimentação ao último dynamixel conforme a figura a seguir.

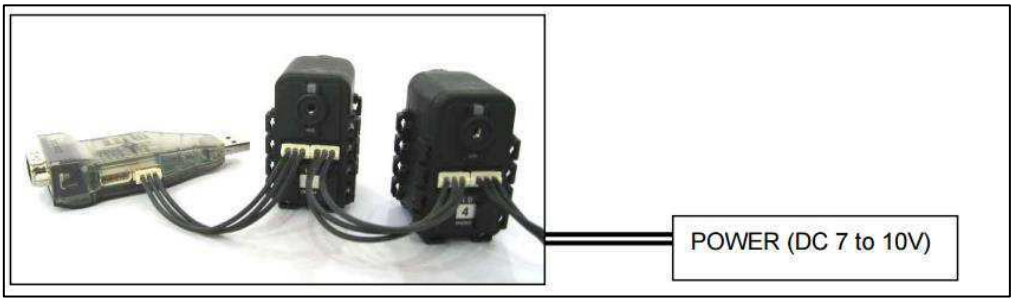
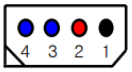
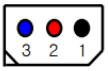


Figura 10 – Conexão da alimentação

A figura a seguir mostra a a pinagem no USB2Dynamixel:

Pin Figure of 4P/3P Cable Connector

4 Pin Cable			3 Pin Cable		
Pin No.	Signal	Pin Figure	Pin No.	Signal	Pin Figure
1	GND		1	GND	
2	NOT Connected		2	NOT Connected	
3	DATA + (RS-485)		3	DATA (TTL)	
4	DATA - (RS-485)				

Pin Figure of Serial Connector

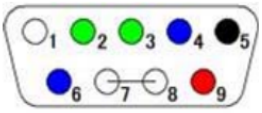
Pin No.	Signal	Pin Figure
1	Data (TTL)	
2	RXD (RS-232)	
3	TXD (RS-232)	
4	D+ (RS-485)	
5	GND	
6	D- (RS-485)	
7	Short with No. 8	
8	Short with No. 7	
9	USB power (5V)	

Figura 11 – Pinagem do dynamixel

As funções ou modos de comunicações propiciadospelo USB2Dynamixel são:

1. Comunicação TTL: dynamixels de 3 pinos tais como os da série AX;

2. Comunicação RS485: dynamixels de 4 pinos tais como os das séries DX, RX e EX;

3. Comunicação RS232: controladores usando cabo serial tais como o CM-5 e CM-510.

3.2 Biblioteca Dynamixel SDK

A fim de realizar o controle utilizou-se a biblioteca SDK, escrita em C, que apesar de seu número limitado de funções facilitou consideravelmente tal controle. O fato dela ser escrita em C e ser padronizada garante a portabilidade e versatilidade frente a diversos controladores em uma mesma aplicação.

Antes de exemplificar o uso é importante estar atento aos passos necessários para a correto uso da biblioteca Dynamixel SDK descritos a seguir:

1. Disponível na página http://www.robotis.us/dynamixel-sdk/
2. Extrair o arquivo dxl_sdk_win32_v1_02.zip
3. Copiar para a pasta do *script*, que fará o controle do braço, os arquivos *dynamixel.dll*, *dynamixel.h*, *ControlTable.m* e *MyDynamixel.m*.

As funções disponíveis na biblioteca são:

Exit() – Fecha a comunicação entre o PC e os dynamixel's;

Init() – Inicia a comunicação entre o PC e os dynamixel's;

addDevice() – Instancia dynamixel;

removeDevice() – Destrói a instancia do dynamixel;

writeAngle() – Altera a posição angular do servo;

setSpeed() – Altera a velocidade do servo;

Para o efetivo controle do braço se fez necessário incluir/elaborar algumas funções dentro do *script MyDynamixel.m* descritas a seguir

readAngle() – Lê a posição angular do servo;

movendoouparado() – Informa se o eixo do servo está em movimento ou não;

Torque() – Ativa ou desativa o torque do servo;

LedOn() – Aciona led do AX-12A;

LedOff() – Desliga led do AX-12A.

Uma vez configurada a biblioteca, elaborou-se um *script* no *MatLab* tendo em vista apresentar as principais funções da biblioteca acréscido de comentários explicativos, dado a seguir:

```
clc %Limpa a Janela de comandos

clear %Limpa o Workspace
```

```
usb2dynamixel_id = 1; %número atribuido ao USB2Dynamixel que enviará e receberá dados

%dos dynamixels

liec = MyDynamixel(dynamixel_id); %instancia o USB2Dynamixel no código e o nomeia como
%liec

liec.portNum = 3; %informa o número da porta ao qual o USB2Dynamixel est_a conectado

liec.baudNum = 1; %informa o valor do baud rate, no caso 1MB

liec.init(); %inicia a comunicação entre o MatLab e o USB2Dynamixel físico

liec.addDevice(1); %Instancia o dynamixel 1 (no caso um AX-12A)

liec.addDevice(7);

liec.setSpeed('id',1,'RPM',10); %configura a velocidade do dynamixel 1 como 10 cujo valor

%máximo é 16

liec.setSpeed('id',7,'RPM',10);

while (1)

ufcg = input('Digite a posição angular desejada em graus: '); %Captura o valor

%da posição angular desejada

if strcmp(ufcg,'dee') %Se ufcg for igual a string "dee"

break; %Encerre o loop

end

liec.writeAngle('id',1,'deg',ufcg); %Gire o dynamixel 01 até a posição "ufcg" graus

liec.writeAngle('id',7,'deg',ufcg); %Gire o dynamixel 07 até a posição "ufcg" graus

end
```

3.3 RVC Tools

Essa toolbox fornece muitas funções frequentemente utilizadas no campo da robótica, incluindo ferramentas envolvendo cinemática, dinâmica e geração de trajetórias. Com ela pode-se para simular e analisar resultados de experimentos feitos com rôbos reais. Escrita em C, a toolbox é baseada em

métodos muitos gerais garantindo assim uma eficiente e versátil representação dos manipuladores. Tais manipuladores correspondem a objetos no MatLab. Os scripts são de fácil entendimento, talvez em virtude da alta eficiência computacional. Possui funções capazes de lidar com vetores, trasnformações homogêneas necessários para representar em um espaço tridiomensional a posição e a orientações dos *link*s.

Disponível no site http://petercorke.com/Home.html a toolbox é de fácil utilização bastante para utilizar suas funções carregar e executar o arquivo *startup_rvc.m* no *script* a ser elaborado. A figura a seguir ilustra um manipulador criado com a RVC Tools.

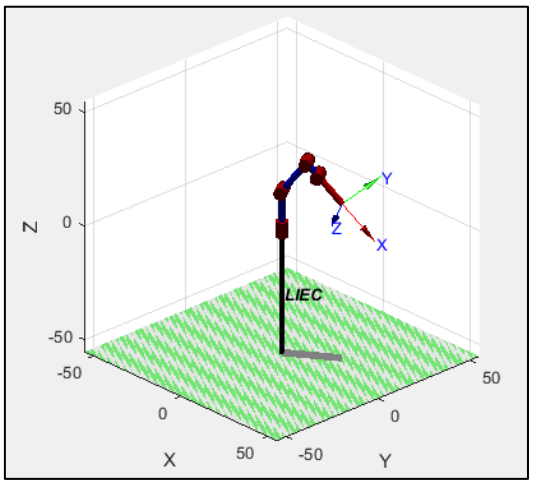


Figura 12 – Representação de um manipulador com a RVC Tools

Antes de exemplificar como implementar tal manipulador com a ferramenta RVC Tools tratar-se-a de elucidar a Notação de Denavit-Hartenberg, comumente abreviada para notação DH.

Notação DH

Um robô manipulador compreende um conjunto de partes, chamadas "*link*s"/ membros unidos por juntas. Cada junta tem um grau de liberdade, seja ela prismática ou rotacional. O movimento da junta altera a posição ou angular dos membros vizinhos. As juntas mais comuns em um manipulador são do tipo de revolução.

O conjunto de juntas de um robô pode ser descrito por uma "string" tal como "RRRRRR" ou "RRRRP", em que o n-ésimo caractere representa o tipo de junta n, seja ela de Revolução ou Prismatica. Uma forma sistemática de descrever a geometria de uma cadeia de membros e juntas foi proposta por

Denavit e Hartenberg em 1955 e hoje denotada por notação de Denavit-Hartenberg.

Para um manipulador com N juntas numeradas de 1 a N, existem N + 1 membros, numerados de 0 até N. O membro / *link* 0 é a base do manipulador e o *link* N (passarei a utilizar o termo *link* ao invés de membro daqui para frente) corresponde a garra ou ferramenta terminal do manipulador.

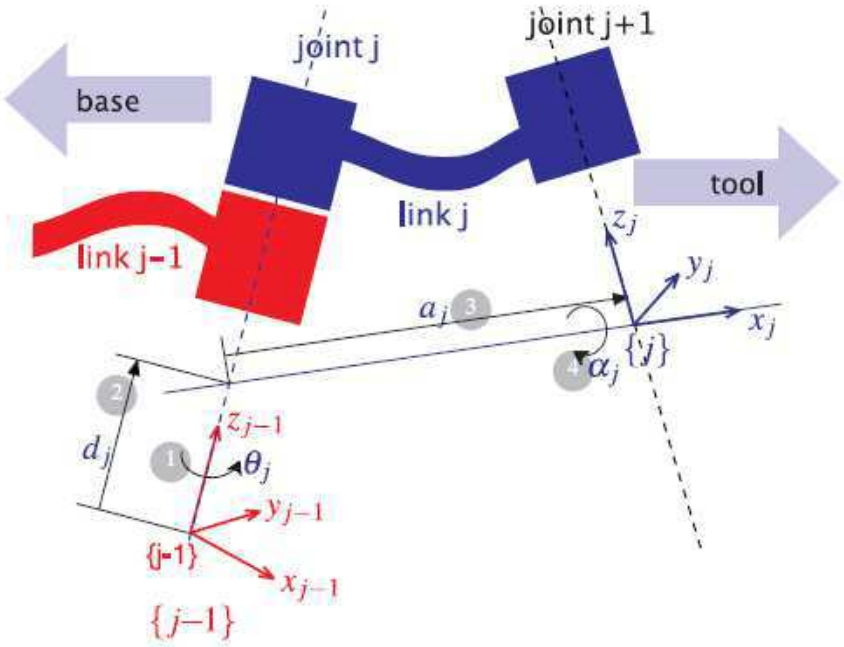


Figura 13 – Notação Denavit-Hartenberg

A junta j conecta o *link j-1* ao *link j* e por isso a junta *j* move o *link j*. Um *link* é considerado como um corpo rígido que define uma relação espacial entre os eixos de duas juntas vizinhas. Um *link* pode ser especificado por dois parâmetros, seu comprimento a_j e sua torçao α_j. Juntas são descritas também por dois parâmetros. O "offset"/compensação d_j é a distância entre a origem do frame de coordenadas da junta *j-1* e o ponto de encontro do prolongamento do eixo na direção do *link* com o frame de coordenadas da junta anterior. O ângulo da junta θ_j é a rotação de um *link* em relação ao outro *link* pertencentes a mesma junta.

Seguindo essa convenção a primeira junta, junta 1, conecta o *link 0* ao *link 1*. O *link 0* é a base do robô. Comumente para o primeiro link $d_1 = \alpha_1 = 0$. Normalmente a base é fixa ao ambiente mais ela pode ser móvel. A seguir é dado um quadro resumo.

Tabela 02 – Resumo da notação DH

Ângulo da junta	θ_j	Ângulo entreos eixos x_{j-1} e x_j sobre o eixo-z	Variável na junta de revolução
Link offset	d_j	A distância da origem do frame *j-1* ao eixo x_j ao longo eixo z_{j-1}	Variável na junta prismática

Comprimento do link	a_j	A distância entre os eixos z_{j-1} e z_j ao longo do eixo x_j; para a intersecção do eixo paralelo a $z_{j-1}xz_j$	Constante
Torção do link	α_j	O ângulo entre os eixos z_{j-1} e z_j sobre o eixo x_j	Constante
Tipo de junta	σ_j	σ=0 para um junta de revolução e σ=1 para uma junta prismática	Constante

Obtido o conhecimento da Notação DH, passaremos a descrever algumas obteu-se os valores dos parâmetros para o Braço cujo código e resultado de sua implementação na ferramenta RVC Tools encontram-se logo a seguir:

```
clear

run('startup_rvc')

L(1) = Link('d', 16.5, 'a', 0, 'alpha', pi/2);
L(2) = Link('d', 0, 'a', 16.7, 'alpha', 0);
L(3) = Link('d', 0, 'a', 7, 'alpha', 0);
L(4) = Link('d', 0, 'a', 15, 'alpha', 0);

BRACO = SerialLink(L, 'name', 'LIEC');

BRACO.plot([0 pi/2 0 0])
```

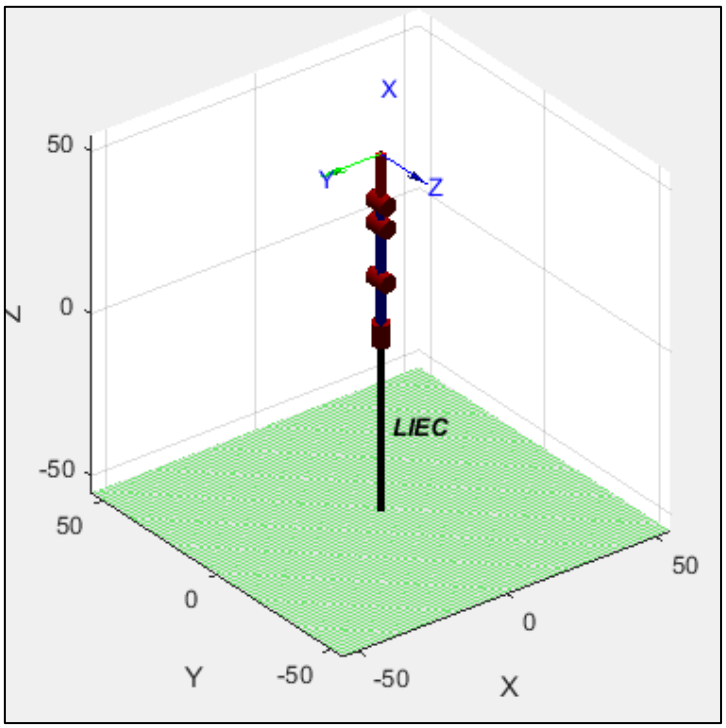


Figura 14 – Representação do braço robótico inteligente AX-12A com a RVC Tools.

3.4 Cinemática inversa utilizando geometria e Lógica Fuzzy

A toolbox RVC Tools possui funções que depende quase que exclusivamente dos parâmetros da notação DH. Todavia aquele que manipula a interface, a ser desenvolvida, dificilmente conhecerá tais parâmetros, restando assim comumente o sistemas de coordenadas cartesianas. Daí a necessidade de buscar métodos que forneçam os parâmtros DH a partir dos valores cartesianos. Na literatura, classifica-se esse tipo de problema como problema de cinemática inversa.

Valeu-se de dois métodos para resolver o problema de cinemática inversa. O primeiro consiste em utilizar a geometria euclidiana ao passo que o segundo usa conceitos da Lógica Fuzzy. É importante ter em mente que o primeiro ratifica a eficácia do segundo método.

Método geométrico

Adotando a articulação dotada de dois atuadores inferior como referência podemos decompor o braço robótico em apenas dois *links* cujos ângulos denotaremos por θ_1 (ângulo de rotação da base), θ_2e θ_3, além disso denotaremos o comprimento de cada um dos *links* como l_1 e l_2. Conforme ilustra a figura abaixo:

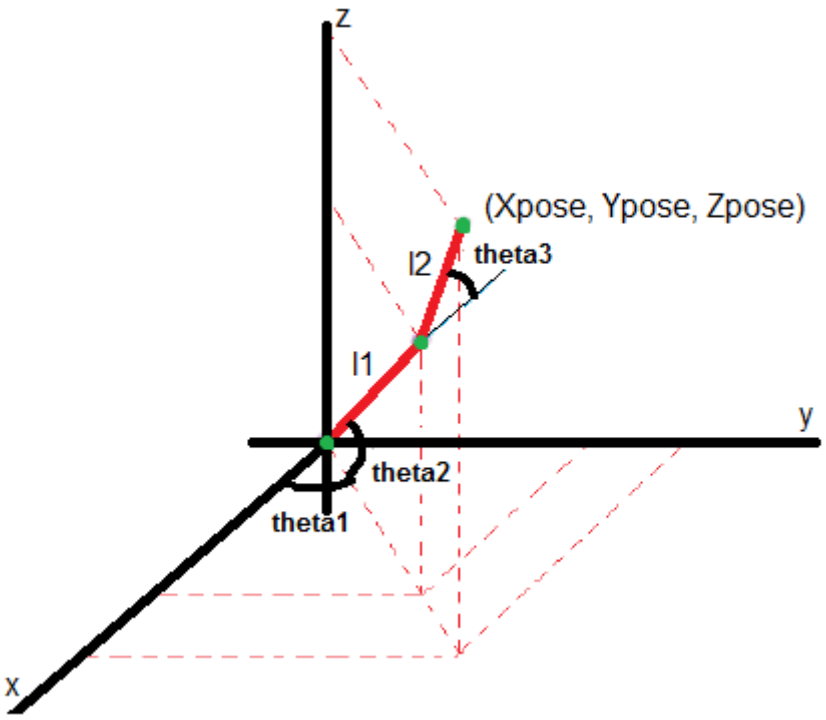


Figura 15 – Modelo geométrico do braço robótico

Pode-se chegar facilmente a partir da aplicação da Lei dos Cossenos as seguintes expressões:

$$\theta_1 = \tan^{-1}\left(\frac{ypose}{xpose}\right)$$

$$\theta_2 = \tan^{-1}\left(\frac{zpose}{\sqrt{xpose^2 + ypose^2}}\right) - \cos^{-1}\left(\frac{xpose^2 + ypose^2 + zpose^2 + {l_1}^2 - {l_2}^2}{2l_1\sqrt{xpose^2 + ypose^2 + zpose^2}}\right)$$

$$\theta_3 = \cos^{-1}\left(\frac{xpose^2 + ypose^2 + zpose^2 - {l_1}^2 - {l_2}^2}{2l_1 l_2}\right)$$

Método usando lógica Fuzzy

Usando lógica fuzzy foi construído um Sistema fuzzy de inferência que deduz a cinemática inversa a partir da cinemática direta. Além disso, a solução fuzzy é facilmente entendível e não requer elevado conhecimento e compreensão da área fuzzy.

As coordenadas (x,y,z) são mapeadas para os ângulos (theta1, theta2,theta2). Esse treinamento, denominado rede ANFIS(Adaptive Neuro-Fuzzy Inference System) é então usado como parte de um sistema de controle maior de forma a controlar o braço. Assim, conhecendo a localização desejada do braço robótica, o sistema de controle usa a rede treinad ANFIS para deduzir as posições angulares das juntas de forma a mover o braço para a localização desejada.

A geração dos dados aplicou-se a posição angular das juntas variando de pequenos incrementos, respeitando as limitações impostas pelos atuadores. , nas equações da cinemática direta dadas a seguir:

$$xpose = (cos\ \theta_1 l_1 + \cos(\theta_1 + \theta_2)\ l_2) sen\ \theta_3$$

$$ypose = \ (cos\ \theta_1 l_1 + \cos(\theta_1 + \theta_2)\ l_2) cos\ \theta_3$$

$$zpose = \mathrm{l}_1 \mathrm{sen}\ (\theta_1) + sen\ (\theta_1 + \theta_2) l_2$$

Cuja visualização dos pontos alcancáveis pelo braço robótico é dada a seguir:

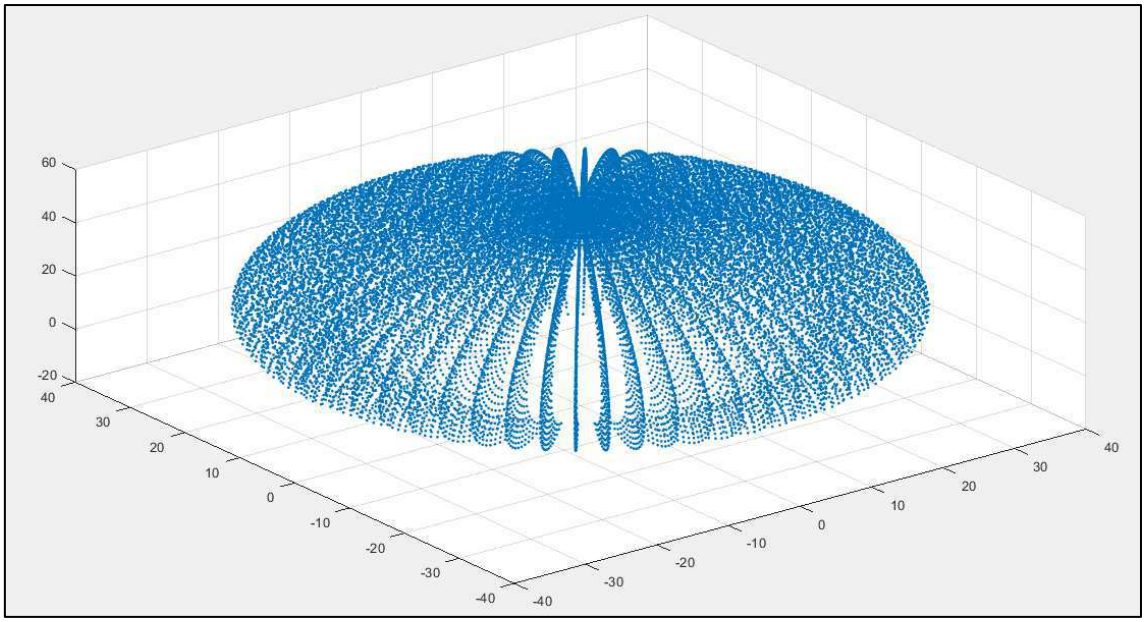


Figura 16 – Posições alcancáveis pelo braço robótico

Comparação entre o Método Geométrico e o Fuzzy

No intuito de validar o método obtido pelo técnica fuzzy elaborou-se um pequeno *script* no MatLab de forma que uma vez selecionados 150 amostras calculou-se o erro obtido da diferença entre o valor obtido geometricamente e o valor obtido via lógica fuzzy.

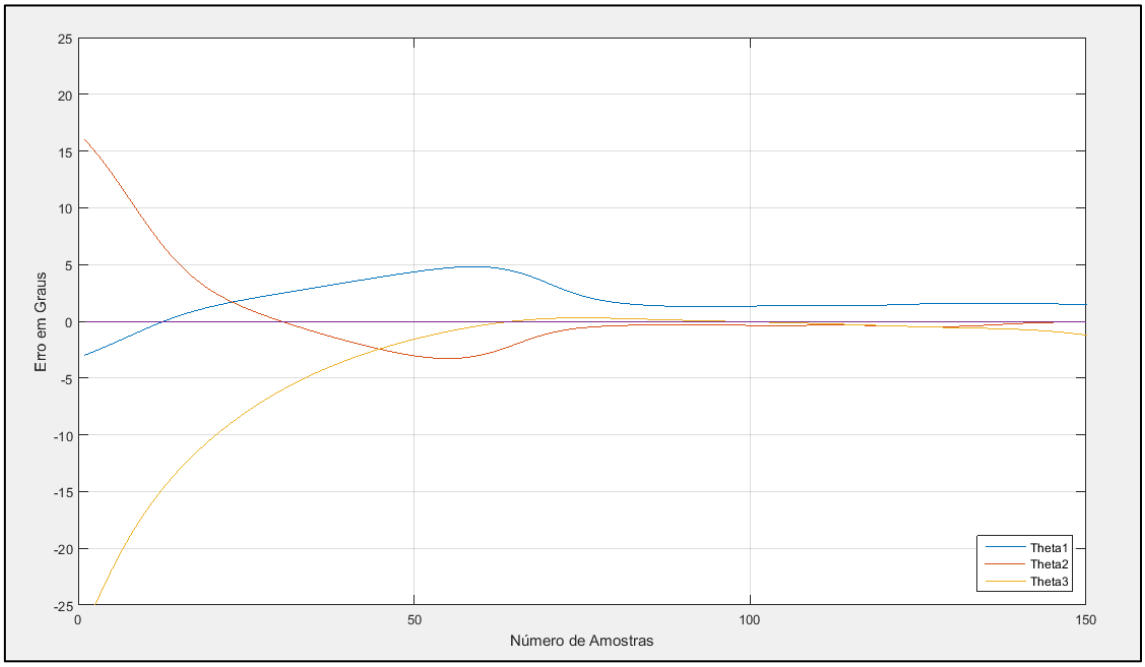


Figura 17 – Comparação entre o método geométrico e o Fuzzy

Como é possível perceber para algumas amostras o erro foi considerável (cerca de 15º) enquanto para outras foi pequeno (cerca de 2,5º). Tal resultado leva a crer que o método fuzzy pode ser utilizado para casos em que a solução geométrica é inviável tendo em vista sua relativa precisão como uma solução provisória.

3.5 Elaboração da Interface Gráfica

No intuito de facilitar os testes com o braço robótico eleborou-se uma interface valendo-se do tipo GUIDE disponível no MatLab, tal interface pode ser visualizada na figura a seguir.

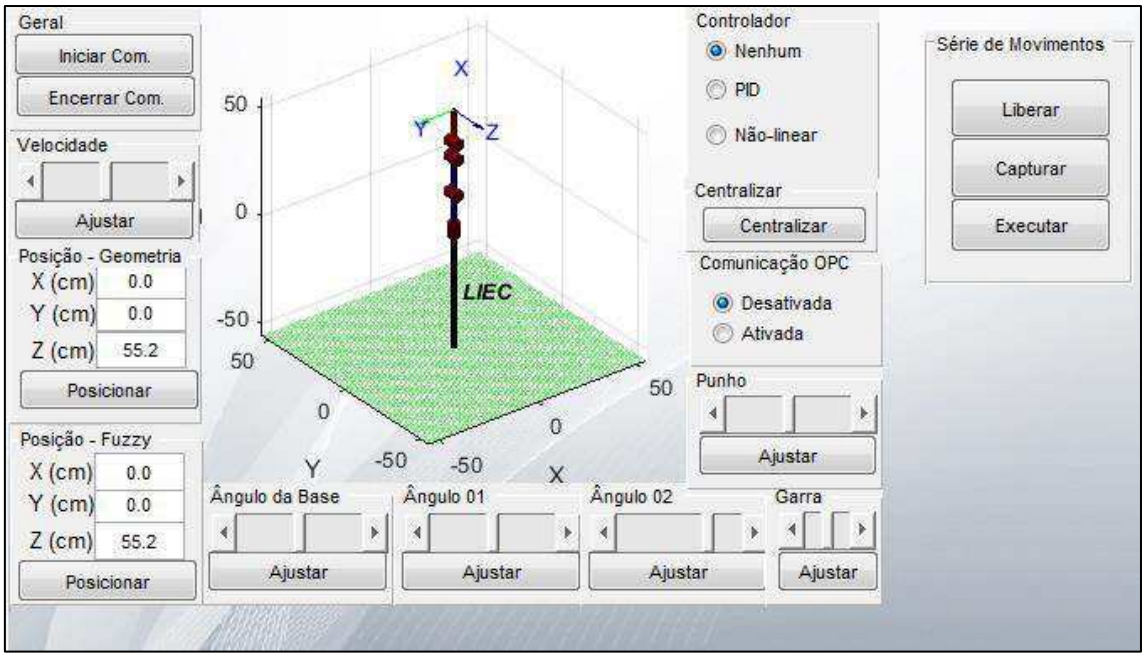


Figura 18 – Interface gráfica

Visando um melhor entendimento da interface haverá uma descrição da implementação e funcionamento dos principais botões a partir daqui.

Botão *"Iniciar Comunicação"*

O botão "*Iniciar Comunicação*" instancia um objeto do tipo MyDynamixel e atribui a ele o nome LIEC. Além disso, ele configura a porta e a velocidade da comunicação serial, inicia a comunicação e carrega os sete atuadores AX-12A do braço robótico. Por fim, exibe uma mensagem informando que a comunicação foi realizada com sucesso A seguir é dado o código que realiza essas operações:

```
dynamixel_id = 7;
LIEC  = MyDynamixel(dynamixel_id);
clc
LIEC.portNum = 3;
LIEC.baudNum = 1;
LIEC.init();
LIEC.addDevice(1);
LIEC.addDevice(2);
LIEC.addDevice(3);
LIEC.addDevice(4);
LIEC.addDevice(5);
LIEC.addDevice(6);
LIEC.addDevice(7);

fprintf('Conexão estabelecida com sucesso\n');
msgbox('Conexão estabelecida com sucesso');
```

Botão *“Encerrar comunicação”*

O botão "*Encerrar Comun.*" libera o conjunto de atuadores carregados no botão "*Iniciar Comun.*" e exibe uma mensagem confirmando que a sessão foi encerrada. A seguir é dado o código que realiza essas operações:

```
LIEC.removeDevice(1);
LIEC.removeDevice(2);
LIEC.removeDevice(3);
LIEC.removeDevice(4);
LIEC.removeDevice(5);
LIEC.removeDevice(6);
LIEC.removeDevice(7);
%LIEC.Exit();
fprintf('Conexão encerrada com sucesso\n');
msgbox('Conexão encerrada com sucesso');
```

Botão “*Ajustar*” da seção Velocidade

Esse botão configura a velocidade do conjunto de atuadores em um valor entre 0 e 16, de forma que todos os AX-12A possuirão a mesma velocidade. A seguir é dado o código que realiza essa operação:

```
aux = get(handles.configvelocidade, 'Value');
velocidade = 6+10*aux;

LIEC.setSpeed('id',1,'RPM',velocidade);
LIEC.setSpeed('id',2,'RPM',velocidade);
LIEC.setSpeed('id',3,'RPM',velocidade);
LIEC.setSpeed('id',4,'RPM',velocidade);
LIEC.setSpeed('id',5,'RPM',velocidade);
LIEC.setSpeed('id',6,'RPM',velocidade);
LIEC.setSpeed('id',7,'RPM',velocidade);
```

Botão “*Posicionar*”

Independentemente do posicionamento ser pelo Método geométrico ou Fuzzy possui a mesma função que é posicionar o braço robótico levando em conta os valores das coordenadas X, Y e Z fornecidas pelo usuário. Além disso o movimento é reproduzido na tela da interface de maneia síncrona. A seguir é dado o código utilizado para implementar essa operação valendo-se do método geométrico.

```
LIEC.Torque(1);

X = str2double(get(handles.xg, 'String'));
Y = str2double(get(handles.yg, 'String'));
Z = str2double(get(handles.zg, 'String'));
Z = Z-16.5;

if Y>0
    theta1 = atan(Y/X);
end
if Y<0
    theta1 = atan(Y/X);
end

if Y == 0
    if X < 0
        theta1 = pi;
    else
        theta1 = 0;
    end
end

if X == 0
    if Y < 0
        theta1 = -pi/2;
    else
        theta1 = pi/2;
    end
end

if (Y==0 && X==0)
    theta1 = 0;
    theta2 = 0;
else
theta2 = atan(Z/sqrt(X^2+Y^2))-acos((X^2+Y^2+Z^2+L1^2-
L2^2)/(2*L1*sqrt(X^2+Y^2+Z^2)));
end

if Z < 0
theta2 = pi/2;
end

theta3 = real(acos((X^2+Y^2+Z^2-L1^2-L2^2)/(2*L1*L2)));

if X == 0 && Y == 0
    theta1 = 0;
    theta2 = pi/2;
end

if(abs(theta2)<0.01)
    theta2 = 0;
end

if(abs(theta3)<0.01)
    theta3 = 0;
end

BRACO.plot([theta1 theta2 theta3 0])
```

```
theta1 = theta1*180/pi;
theta2 = theta2*180/pi;
theta3 = theta3*180/pi;

LIEC.setSpeed('id',1,'RPM',velocidade);
LIEC.setSpeed('id',4,'RPM',velocidade);
LIEC.setSpeed('id',5,'RPM',velocidade);
LIEC.setSpeed('id',2,'RPM',velocidade);
LIEC.setSpeed('id',3,'RPM',velocidade);

LIEC.writeAngle('id',1,'deg',150 + theta1);
while(LIEC.movendoouparado(1))
end

LIEC.writeAngle('id',4,'deg',150 + theta3);
LIEC.writeAngle('id',5,'deg',150 - theta3);
while(LIEC.movendoouparado(5))
end

LIEC.writeAngle('id',2,'deg',150 + 90 - theta2);
LIEC.writeAngle('id',3,'deg',150 - 90 + theta2);
while(LIEC.movendoouparado(3))
end

axes(handles.axes2);
```

Botão "*Centralizar*"

Esse botão fixa os ângulos dos atuadores em 150º de forma a obter a posição com maior coordenada Z. A seguir é dado o trecho do código responsável por executar essa operação.

```
LIEC.Torque(1);

LIEC.setSpeed('id',1,'RPM',velocidade);
LIEC.setSpeed('id',2,'RPM',velocidade);
LIEC.setSpeed('id',3,'RPM',velocidade);
LIEC.setSpeed('id',4,'RPM',velocidade);
LIEC.setSpeed('id',5,'RPM',velocidade);
LIEC.setSpeed('id',6,'RPM',velocidade);

BRACO.plot([0 pi/2 0 0])
LIEC.writeAngle('id',1,'deg',150);
while(LIEC.movendoouparado(1))
end

LIEC.writeAngle('id',4,'deg',150);
LIEC.writeAngle('id',5,'deg',150);
while(LIEC.movendoouparado(5))
end
LIEC.writeAngle('id',2,'deg',150);
LIEC.writeAngle('id',3,'deg',150);
while(LIEC.movendoouparado(3))
end
axes(handles.axes2);
```

Botão “*Liberar*”

Esse botão trata de desativa o torque dos AX-12A de forma que seja possível movimentar o braço livremente sem qualquer torque de resistência proveniente dos motores. Para isso foi preciso criar uma função *Torque* que ativa ou desativa o torque dos atuadores de acordo com a precisão. A seguir é dado o código contido no botão “Liberar”.

```
LIEC.Torque(0);
```

Botão “*Capturar*”

Esse botão captura as posições angulares do AX-12A e armazena em um arquivo de texto.

Botão “*Executar*”

Esse botão executa o conjunto de leituras feitas pelo botão capturar.

3.6 V-REP

Além das simulações feitas com a ferramenta RVC Tools foram feitas algumas simulações com o ambiente de simulação V-REP de fácil integração com o MatLab. Para isso, inicialmente elaborou-se um modelo do braço de forma a atender as características físicas com relativa precisão, conforme ilustra a figura a seguir:

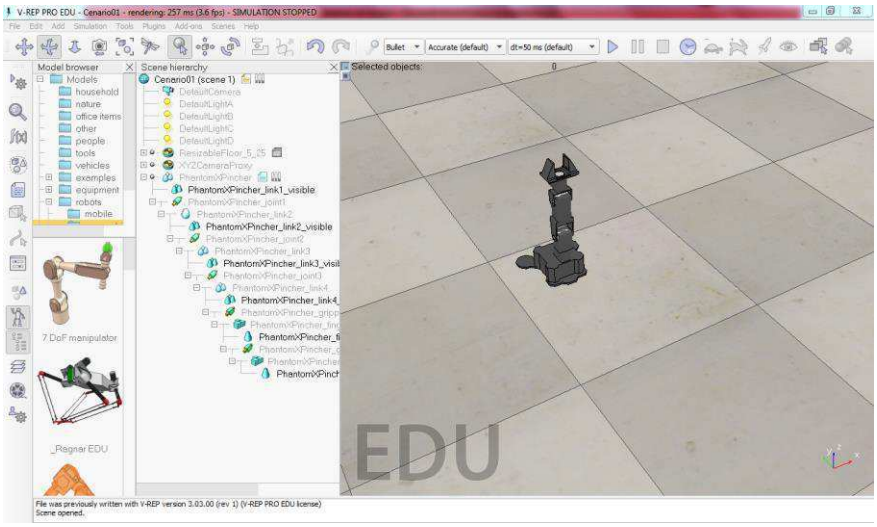


Figura 19 – Ambiente de simulação V-REP

O controle simultâneo do V-REP e do braço robótico real se mostrou efetivo mas deixou a desejar no que tange velocidade de processamento, fato esperado em virtude da memória solicitado pelo processo de comunicação com o V-REP. Mesmo assim, pode-se dizer que o V-REP se aproxima mais da realidade em termos de design e pode muito bem ser integrado no controle do braço robótico usando linguagens mais leves com C, Lua, Python e ROS.

Abaixo um simples código comunicando o MatLab ao V-REP:

```
clear all
clc

vrep = remApi('remoteApi');
vrep.simxFinish(-1);

connectionStatus = 0;

while(connectionStatus == 0)
  clientID = vrep.simxStart('127.0.0.1', 19997, true, true,5000,5);
    pause(0.1);
    if(clientID>-1)
        fprintf('Conexão estabelecida com sucesso');
        connectionStatus = 1;
    else {
        fprintf('Falha na conexão...\nTentando se conectar\n');
    }
    end
end

[res(1),BRACOVREP] = vrep.simxGetObjectHandle(clientID,…
'BRACOVREP', vrep.simx_opmode_oneshot_wait);
while(res(1))
```

```
    [res(1),BRACOVREP] = vrep.simxGetObjectHandle(clientID,…
'BRACOVREP', vrep.simx_opmode_oneshot_wait);
end

[res(2),BRACOVREP_junta1] = vrep.simxGetObjectHandle(clientID,…
'BRACOVREP_joint1', vrep.simx_opmode_oneshot_wait);
while(res(2))
    [res(2),BRACOVREP_junta1] = vrep.simxGetObjectHandle(clientID,…
'BRACOVREP_joint1', vrep.simx_opmode_oneshot_wait);
end

[res(3),BRACOVREP_junta2] = vrep.simxGetObjectHandle(clientID,…
'BRACOVREP_joint2', vrep.simx_opmode_oneshot_wait);
while(res(3))
    [res(3),BRACOVREP_junta2] = vrep.simxGetObjectHandle(clientID,…
'BRACOVREP_joint2', vrep.simx_opmode_oneshot_wait);
end

[res(4),BRACOVREP_junta3] = vrep.simxGetObjectHandle(clientID,…
'BRACOVREP_joint3', vrep.simx_opmode_oneshot_wait);
while(res(1))
[res(4),BRACOVREP_junta3] = vrep.simxGetObjectHandle(clientID,…
'BRACOVREP_joint3', vrep.simx_opmode_oneshot_wait);
end

[res(7)] = vrep.simxStartSimulation(clientID,…
vrep.simx_opmode_oneshot);

[res(6)] = vrep.simxSetJointTargetPosition(clientID,…
BRACOVREP_junta3, pi/2, vrep.simx_opmode_oneshot);
while(res(6))
    [res(6)] = vrep.simxSetJointTargetPosition(clientID,…
BRACOVREP_junta3, pi/2, vrep.simx_opmode_oneshot);
end

pause(1)

[res(6)] = vrep.simxSetJointTargetPosition(clientID,…
BRACOVREP_junta1, 0, vrep.simx_opmode_oneshot);
while(res(6))
    [res(6)] = vrep.simxSetJointTargetPosition(clientID,…
BRACOVREP_junta1, 0, vrep.simx_opmode_oneshot);
end

pause(5);

vrep.simxStopSimulation(clientID, vrep.simx_opmode_oneshot_wait)
fprintf('Fim de simulação\n');
vrep.simxFinish(clientID);
vrep.delete();
```

3.7 Comunicação OPC

Vislumbrando uma futura aplicação instalou-se um servidor OPC de forma que independentemente dos fabricantes dos dispositivos, e consequentemente de seus protocolos, a serem utilizados no controle existirá um único cliente OPC responsável pelo interfaceamento com o braço robótico inteligente AX-12A.

Tomando como pressuposto que o dispositivo a ser instalado pertence a família eZAP91x da HI-Tecnologia partira-se-á para a descrição das configurações necessárias para efetivo comunicação com o MatLab. Além disso, é importante frizar que as descrições a seguir dizem respeito a configuração estabelecida via Ethernet para a informações o caso da comunicação serial vide www.hitecnologia.com.br/. Não é necessário realizar alterações na seção destinada ao Canal de Comunicação (Channel). Por conseguinte descrevemos o menu Equipamento como uma forma de adicionar os dispositivos, no caso em estudo o eZAP, por meio da especificação de seu IP. Por fim temos o Bloco de Dado (Datablock) que corresponde as variáveis a serem lidas/escritas. Para os controladores da HI estão disponíveis três tipos de Memória, Tipo R (valores lógicos), Tipo M (inteiros com sinal: 16 bits) e Tipo D (Ponto flutuante: 32 bits).

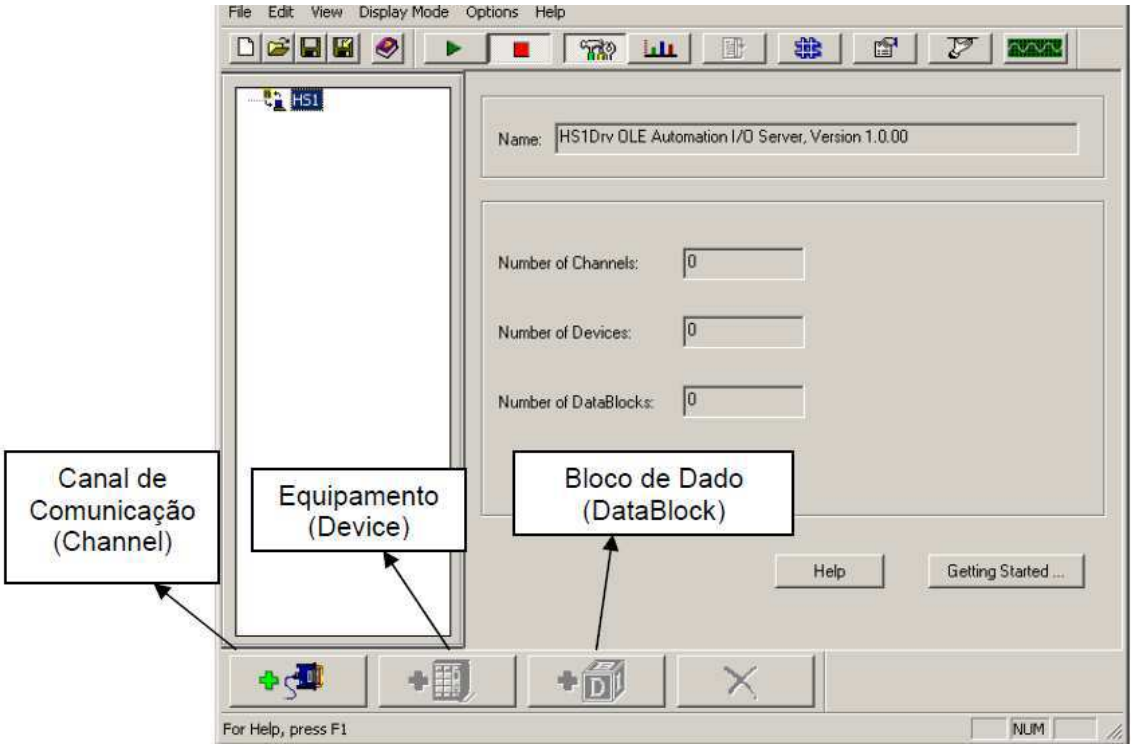


Figura 20 – Configuração OPC

No MatLab, logrou-se pela OPC Toolbox ™ que fornece uma conexão com servidores OPC DA e OPC HDA, dando-lhe acesso a dados históricos do OPC a partir do MATLAB ® e Simulink ® .Ou seja, você pode ler, escrever e registrar dados OPC de dispositivos, tais como sistemas distribuídos de controle, controle de supervisão e sistemas de aquisição de dados e controladores lógicos programáveis, que estejam em conformidade com o padrão OPC Foundation Acesso a Dados (DA). Você pode ler e analisar dados a partir de qualquer historiador de dados que está em conformidade com o padrão OPC Foundation histórico de Acesso a Dados (HDA).

A toolbox inclui blocos no Simulink que permitem modelar o controle de supervisão on-line e realizar o teste de controlador de hardware-in-the-loop.

3.8 Modelagem e Identificação

A modelagem mecânica do braço robótico foi realizada considerando a atuação de um servo por vez. Além disso desprezou-se as vibrações presentes na garra do braço robótico. De forma que os modelos obtidos foram para cada articulação:

Base

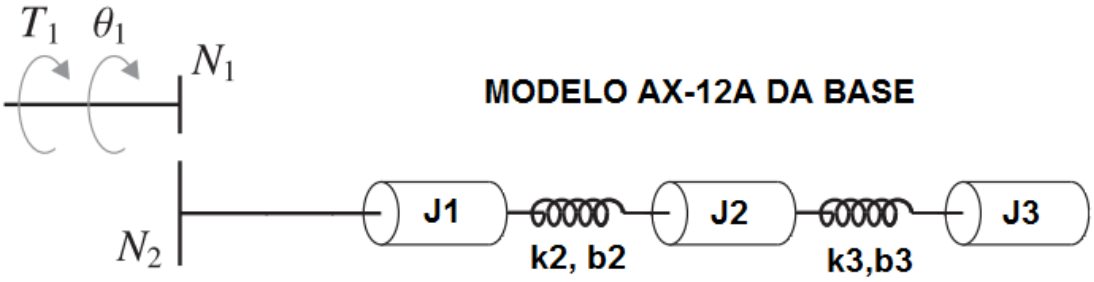


Figura 21 – Modelo da base

Ombro

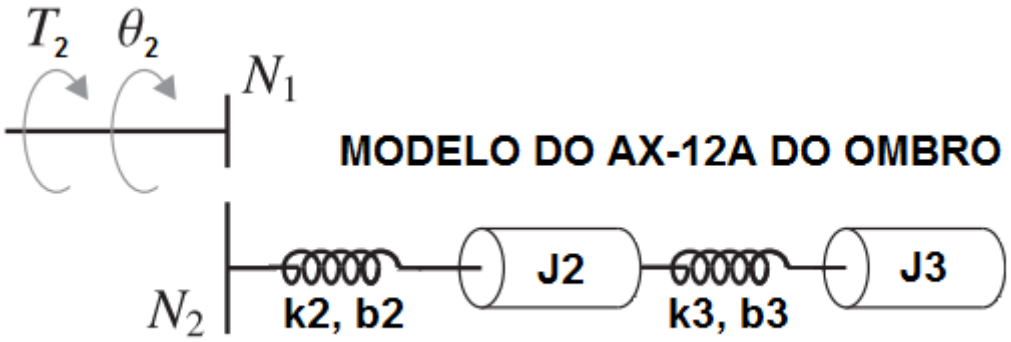


Figura 22 – Modelo do ombro

Cotovelo

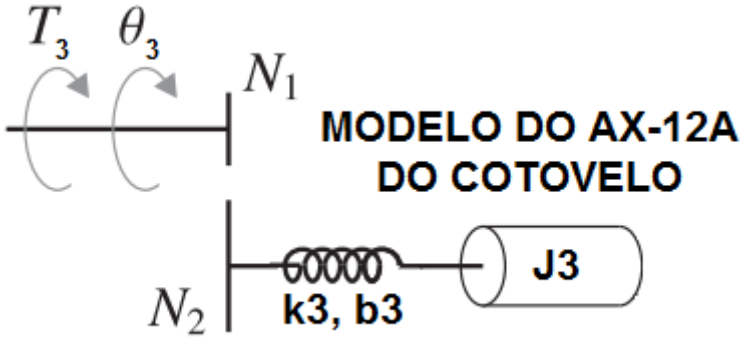


Figura 23 – Modelo da base

T_1 – Torque desenvolvida pelo servo da base do braço robótico;

T_2 – Torque desenvolvida pelo conjunto de servos do ombro do braço robótico;

T_3 – Torque desenvolvida pelo conjunto de servos do cotovelo do braço robótico;

θ_1 – Posição angular do servo da base do braço robótico;

θ_2 – Posição angular do conjunto de servos do ombro do braço robótico;

θ_3 – Posição angular do conjunto de servos do cotovelo do braço robótico;

θ_1 – Posição angular do servo da base do braço robótico;

Demonstrar-se-a o desenvolvimento da função de transferência para o caso do cotovelo como forma de justificar a escolha de modelos de 2ª ordem numa eventual identificação.

$$\theta_3{}' = \frac{N_1}{N_2}\theta_3 \qquad (1)$$

$$T_3{}' = \frac{N_2}{N_1}T_3 \qquad (2)$$

$$T_3{}' - k_3(\theta_3{}' - 0) - b_3\left(\dot{\theta_3}{}' - 0\right) = J_3\ddot{\theta_3}{}' \qquad (3)$$

Substituindo (1) e (2) em (3) obtemos:

$$\frac{N_2}{N_1}T_3 - k_3\left(\frac{N_1}{N_2}\theta_3 - 0\right) - b_3\left(\frac{N_1}{N_2}\dot{\theta}_3 - 0\right) = J_3\frac{N_1}{N_2}\ddot{\theta_3}{}'$$

$$\frac{\theta_3(s)}{T_3(s)} = \frac{\left(\frac{N_2}{N_1}\right)^2}{J_2 s + b_3 s + k_3}$$

É importante frizar que a taxa de redução de cada servo, ou seja, $\frac{N_1}{N_2}$ é igual a $\frac{1}{254}$.

Identificação

No intuíto de realizar a identificação submeteu-se cada articulação a um deslocamento de 70º no sentido positivo (anti-horário) de rotação. E se fez a leitura da posição angular para uma velocidade equivalente a 25% da máxima até o sistema entrar em regime permanente.

Dessa forma para o servo da base obteu-se a seguinte resposta:

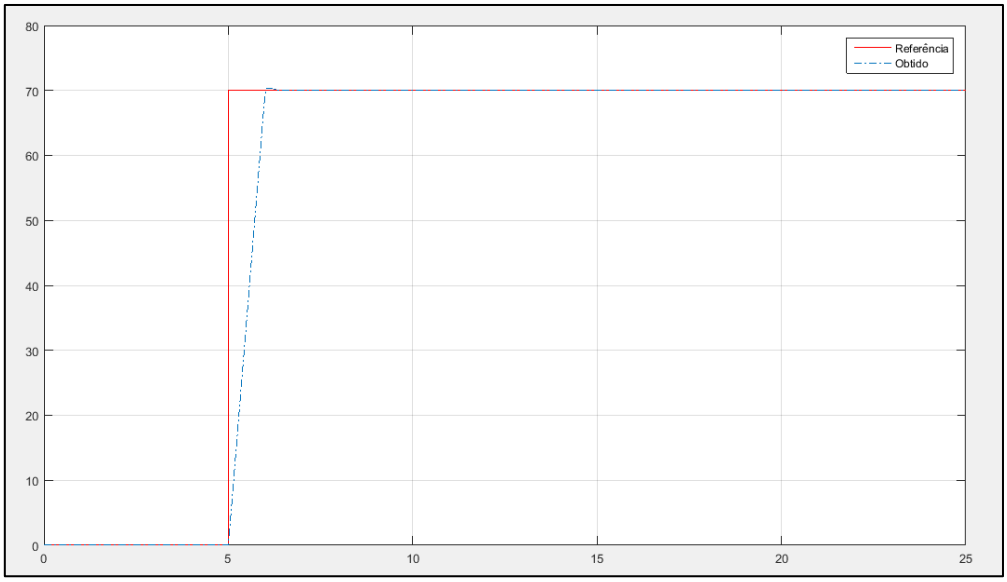


Figura 24 – Leituras e referência para o servo da base

Para o conjunto de servos localizado no ombro observou-se o seguinte comportamento:

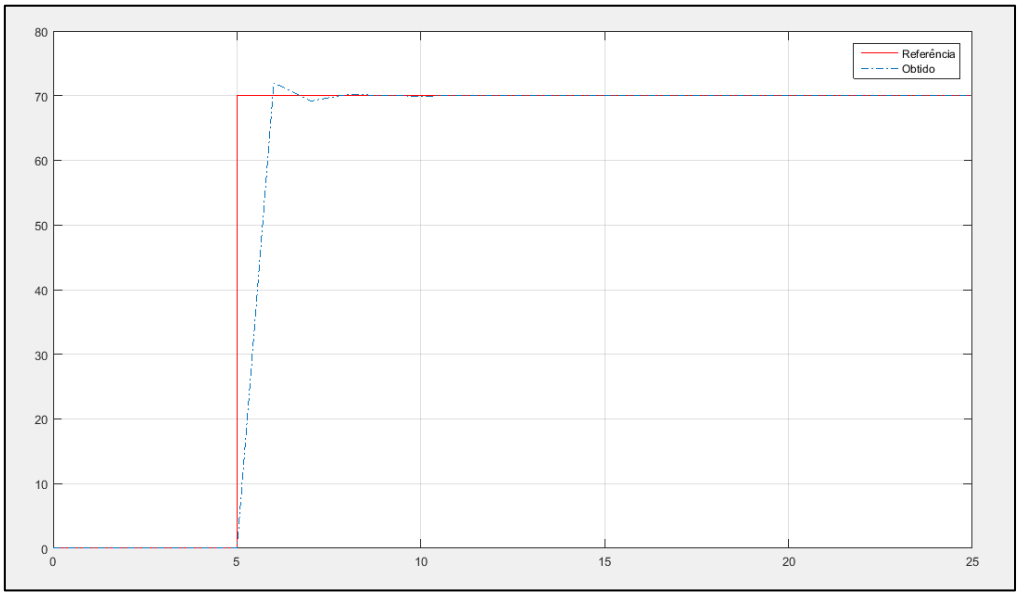


Figura 25 – Leituras e referência para o conjunto de servos do ombro

Como era de se esperar o movimento do ombro possui um carater oscilante em detrimento ao da base, tal oscilação acentua-se na mesma proporção da velocidade de movimento da articulação.

Para o conjunto de servos localizado no cotovelo observou-se o seguinte comportamento:

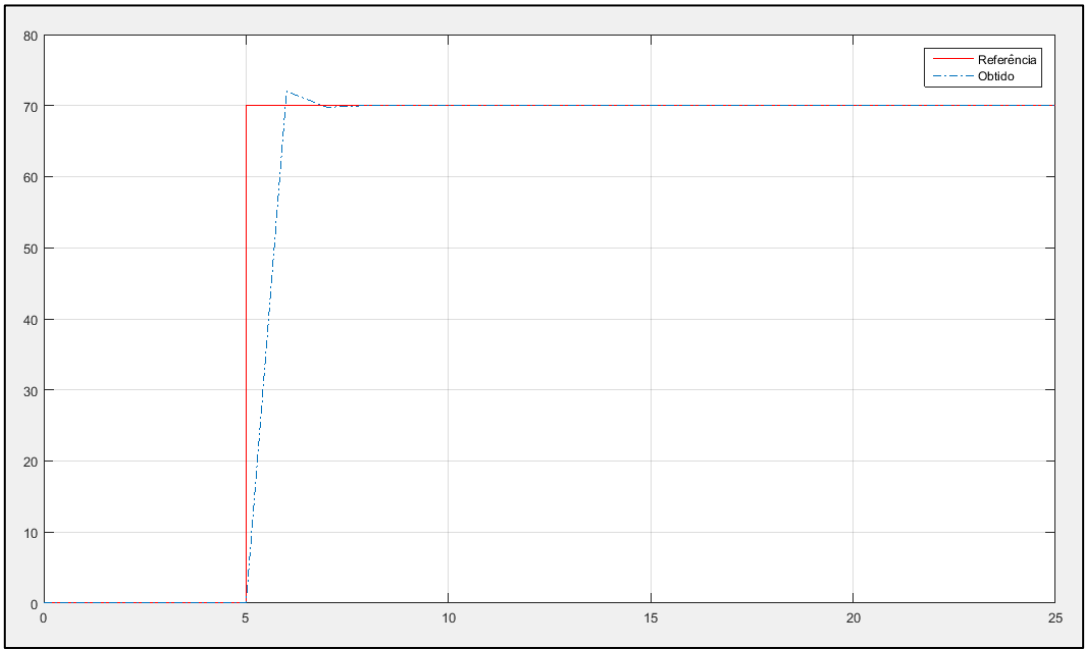


Figura 26 – Leituras e referência para o conjunto de servos do cotovelo

Como era de se esperar a oscilação do cotovelo tomou aspectos intermediários entre a resposta da base e do ombro. Valendo-se da Toolbox Ident do MatLab, chegou-se as seguintes funções de transferência:

$$H_{base}(s) = \frac{18.940}{s^2 + 3.577s + 19.104}$$

$$H_{ombro}(s) = \frac{37.152}{s^2 + 13.072s + 37.930}$$

$$H_{cotovelo}(s) = \frac{60.524}{s^2 + 12.343s + 61.002}$$

3.9 Controle

Infelizmente não foi possível concluir a parte de controle deixando assim como sugestão para atividades futuras.

4. Melhorias Sugeridas

Com base nos testes realizados é possível sugerir algumas melhorias. Tais melhorias são elucidadas nos pontos a seguir:

- Concluir a etapa de controle P, PI e PID;
- Instalar uma câmera o outro objeto de forma que seja possível identificar a posição de objetos;
- Implementar a interface em C++ de forma a maximar a compatibilidade com a biblioteca SDK;

5. Conclusão

O desenvolvimento da interface assim como o estudo da física e controle do braço robótico apresentou um conjunto de desafios, nos quais se fez necessário o conhecimento técnico adquirido em disciplinas outroras cursadas como Introdução a Programação, Técnicas de Programação, Arquitetura de Sistemas Digitais para o desenvolvimento da interface de controle e entendimento do funcionamento dos servos, Controle Analógico para a modelagem física do Braço e Controle Digital para a identificação das articulações.

De forma que uma vez vencidos tais desafios foi possível vislumbrar de forma prática o conhecimento adquirido nas disciplinas do curso supra-citadas.

Por fim, vale destacar o caráter didático do braço robótico, ideal para estudos e implementações relacionadas a área de controle

6. Bibliografia

Dorf, R. C. & Bishop, R. H., 2013. *Sistemas de Controle Modernos.* 12ª ed. Rio de Janeiro: LTC.

LIEC, 2015. *Laboratório de Instrumentação Eletrônica e Controle.* [Online] Available at: http://liec.ufcg.edu.br/[Acesso em 11 05 2016].

Ogata, K., 2010. *Engenharia de Controle Moderno.* 5ª ed. São Paulo: Pearson.

Biblioteca SDK. http://support.robotis.com/en/software/dynamixelsdk.htm /[Acesso em 11 05 2016].

Biblioteca RVC Tools. http://petercorke.com/Robotics_Toolbox.html /[Acesso em 11 05 2016].