

Alequine Batista de Lima

## **Relatório de Estágio Supervisionado**

Campina Grande, Brasil

24 de fevereiro de 2017

Alequine Batista de Lima

## **Relatório de Estágio Supervisionado**

Relatório de Estágio Supervisionado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Orientador: George Acióli Júnior, D. Sc.

Campina Grande, Brasil

24 de fevereiro de 2017

---

Alequine Batista de Lima

Relatório de Estágio Supervisionado/ Alequine Batista de Lima. – Campina Grande, Brasil, 24 de fevereiro de 2017-

37 p. : il. ; 30 cm.

Orientador: George Acióli Júnior, D. Sc.

Relatório de Estágio Supervisionado – Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE , 24 de fevereiro de 2017.

---

Alequine Batista de Lima

## **Relatório de Estágio Supervisionado**

Relatório de Estágio Supervisionado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Trabalho aprovado em \_\_\_\_ de fevereiro de 2017.

---

**George Acióli Júnior, D. Sc.**  
Orientador

---

**Péricles Rezende Barros, PhD.**  
Convidado

Campina Grande, Brasil  
24 de fevereiro de 2017

*Dedico este trabalho a Romildo Cassiano de Lima, amado pai, e a Maria Jerusa Batista de Lima, amada mãe.*

# Agradecimentos

Primeiramente, aos mais importantes nesta jornada. Agradeço a meus pais, Romildo e Maria Jerusa, por terem sempre incentivado a procurar o conhecimento, pelas oportunidades oferecidas e pela manutenção do saber desde a minha infância.

Agradeço aos tios e tias que sempre me deram suporte e, em especial, a Luciene, que assim como meus pais sempre incentivou e ajudou, quando possível.

A minha namorada Laryssa, por todo amor, confiança, estímulo e companheirismo.

Aos amigos e colegas de curso, pelas ajudas nas dificuldades encontradas, compartilhamento de materiais de estudos e por ajudarem na manutenção da tranquilidade e paciência, no decorrer deste curso.

Aos meus professores os quais contribuíram para minha formação como aluno. Em especial ao professor Péricles Rezende Barros, pelas oportunidades oferecidas, por me permitir crescer dentro do curso e por ser meu supervisor e ao professor George Acíoli Júnior, por aceitar orientar-me neste estágio supervisionado.

*"It is not knowledge, but the act of learning, not the possession of but the act of getting there, which grants the greatest enjoyment."*

*Carl Friedrich Gauss*

# Resumo

Este relatório apresenta as atividades realizadas pelo aluno Alequine Batista de Lima durante o Estágio Supervisionado no Laboratório de Instrumentação Eletrônica e Controle (LIEC), pertencente ao Departamento de Engenharia Elétrica (DEE) da Universidade Federal de Campina Grande (UFCG), sob orientação do Professor George Acioli Júnior e supervisão do Professor Péricles Rezende Barros. O principal objetivo deste trabalho é apresentar um conjunto de estudos relacionados a uma plataforma robótica de cunho didático, abordando temas recorrentes na graduação, como identificação de sistemas e controle realimentado. Além disso, estudos associados à robótica foram realizados. Estudou-se também aspectos da comunicação OPC (OLE for Process Control), com o intuito de realizar uma integração entre softwares desenvolvidos para a plataforma e o software MATLAB®, onde controladores PI (proporcional-integral) foram implementados.

**Palavras-chaves:** Plataforma robótica; OPC; Controladores;



# Lista de ilustrações

Figura 1 – Faixada das instalações físicas do Laboratório de Instrumentação Eletrônica e Controle . . . . .	3
Figura 2 – Arquitetura representativa de uma rede OPC. . . . .	6
Figura 3 – Ilustração das variáveis de ângulos e de coordenadas . . . . .	8
Figura 4 – Ilustração das variáveis de ângulos e de coordenadas em um posicionamento alternativo . . . . .	8
Figura 5 – Plataforma instalada no LIEC . . . . .	10
Figura 6 – Distribuição dos servos de acordo com capacidade de Torque e Velocidade	11
Figura 7 – Componentes eletrônicos do servo AX-12A . . . . .	12
Figura 8 – Caixa de engrenagens redutoras do servo AX-12A . . . . .	13
Figura 9 – Potenciômetro Murata SV01 utilizado como sensor de posição angular do servo AX-12A . . . . .	13
Figura 10 – Característica física do AX-12A. . . . .	14
Figura 11 – Comunicação entre controlador e 1 AX-12A. . . . .	14
Figura 12 – Comunicação entre o controlador e vários AX-12A. . . . .	15
Figura 13 – Braço robótico inteligente AX-12A. . . . .	15
Figura 14 – Dispositivo USB2Dynamixel, utilizando na conversão entre comunicação serial e USB . . . . .	16
Figura 15 – Ligação entre o servo AX-12A e o computador através do USB2Dynamixel	16
Figura 16 – Seleção de modo de conversão . . . . .	17
Figura 17 – Conexão do tipo 3P. . . . .	17
Figura 18 – Conexão da alimentação. . . . .	17
Figura 19 – Diagrama de conexões do Dynamixel para os tipos de conectores 3P, 4P e Serial . . . . .	18
Figura 20 – Disposição dos eixos na plataforma robótica didática . . . . .	20
Figura 21 – Interface gráfica elaborada em C# . . . . .	21
Figura 22 – Onda quadrada aplicada (azul) ao servo da base e resposta obtida (vermelho) . . . . .	23
Figura 23 – Onda quadrada aplicada (azul) ao servo do ombro e resposta obtida (vermelho) . . . . .	23
Figura 24 – Onda quadrada aplicada (azul) ao servo do cotovelo e resposta obtida (vermelho) . . . . .	24
Figura 25 – Onda quadrada aplicada (azul) ao servo do punho e resposta obtida (vermelho) . . . . .	24
Figura 26 – Onda quadrada aplicada (azul) ao servo da garra e resposta obtida (vermelho) . . . . .	25

Figura 27 – Janela do cliente OPC da interface C# . . . . .	26
Figura 28 – Criando um novo modelo e localização dos blocos OPC do Simulink . . . . .	27
Figura 29 – Adicionando um servidor . . . . .	28
Figura 30 – Selecionando um servidor . . . . .	28
Figura 31 – Adicionando uma malha . . . . .	28
Figura 32 – Adicionando uma variável a ser lida através do bloco <i>OPC Read</i> . . . . .	29
Figura 33 – Adicionando uma variável para escrita através do bloco <i>OPC Write</i> . . . . .	29
Figura 34 – Diagrama de blocos para implementação do controlador PI por eixo. . . . .	30
Figura 35 – Servo do eixo base sem controlador . . . . .	31
Figura 36 – Servo do eixo base com controlador . . . . .	31
Figura 37 – Servos do eixo ombro sem controlador . . . . .	32
Figura 38 – Servos do eixo ombro com controlador . . . . .	32
Figura 39 – Servos do eixo cotovelo sem controlador . . . . .	33
Figura 40 – Servos do eixo cotovelo com controlador . . . . .	33
Figura 41 – Servo do eixo punho sem controlador . . . . .	34
Figura 42 – Servo do eixo punho com controlador . . . . .	34
Figura 43 – Servo do eixo mão sem controlador . . . . .	35
Figura 44 – Servo do eixo mão com controlador . . . . .	35

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>1</b>
<b>2</b>	<b>LABORATÓRIO</b>	<b>3</b>
<b>3</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b>	<b>5</b>
<b>3.1</b>	<b>Padrão de Comunicação OPC</b>	<b>5</b>
3.1.1	Funcionamento	5
<b>3.2</b>	<b>Cinemática Direta e Inversa</b>	<b>6</b>
3.2.1	Cinemática Direta	7
3.2.2	Cinemática Inversa	7
<b>4</b>	<b>A PLATAFORMA</b>	<b>10</b>
<b>4.1</b>	<b>Recursos de Hardware</b>	<b>10</b>
4.1.1	Servos Inteligentes	10
4.1.2	Braço Robótico Inteligente AX-12A	15
4.1.3	USB2Dynamixel	16
<b>4.2</b>	<b>Recursos de Software</b>	<b>18</b>
4.2.1	Biblioteca Dynamixel SDK	18
<b>5</b>	<b>ATIVIDADES REALIZADAS</b>	<b>21</b>
<b>5.1</b>	<b>Elaboração da interface gráfica em C#</b>	<b>21</b>
5.1.1	Primeira e Segunda “Groupbox”	21
5.1.2	Terceira e Quarta “Groupbox”	22
<b>5.2</b>	<b>Identificação</b>	<b>22</b>
5.2.1	Através da interface em C#	22
<b>5.3</b>	<b>Comunicação OPC</b>	<b>26</b>
<b>5.4</b>	<b>Implementação do Controle</b>	<b>30</b>
5.4.1	Eixo Base	31
5.4.2	Eixo Ombro	32
5.4.3	Eixo Cotovelo	33
5.4.4	Eixo Punho	34
5.4.5	Eixo Mão	35
<b>6</b>	<b>CONSIDERAÇÕES FINAIS</b>	<b>36</b>
	<b>REFERÊNCIAS</b>	<b>37</b>

# 1 Introdução

O estágio supervisionado, cujas atividades são descritas neste relatório, teve duração de 180 horas (06 créditos) e foi realizado no Laboratório de Instrumentação Eletrônica e Controle (LIEC), durante o período de 01 de dezembro de 2016 até 03 de fevereiro de 2017, sob a supervisão do professor e engenheiro eletricista Péricles Rezende de Barros e orientação do professor George Acióli Júnior.

O estágio supervisionado tem como objetivo o cumprimento das exigências da disciplina Estágio Curricular, integrante da grade curricular do Curso de Engenharia Elétrica da Universidade Federal de Campina Grande. Essa disciplina é indispensável para a formação profissional, visto que consolida os conhecimentos adquiridos durante o curso de forma prática.

Nesse estágio foram realizadas atividades referentes implementação de controle e de comunicação OPC em servos inteligentes, bem como elaboração de guias e tutoriais para o uso de uma plataforma robótica didática em conjunto como MATLAB®.

O plano de estágio fora pensado de modo a conceber um conjunto de documentos que permitissem introduzir conhecimento a cerca de robótica, identificação de sistemas e comunicação OPC. Utilizando-se da plataforma robótica didática, baseada em servos AX-12A, fora possível realizar aplicação dos conhecimentos adquiridos nas diversas disciplinas integralizadas durante a graduação, tendo uma maior aplicação dos temas abordados nas disciplinas como:

- Introdução a Programação;
- Técnicas de Programação;
- Eletrônica;
- Arquitetura de Sistemas Digitais;
- Controle Analógico;
- Controle Digital;
- Instrumentação Eletrônica;
- Sistemas de Aquisição de Dados e Interface;

Por sua vez, as principais atividades desenvolvidas envolvendo a plataforma robótica didática foram:

- 
- Descrição completa dos aspectos físicos da plataforma robótica;
  - Estudo e implementação apropriada funções disponíveis em Dynamixel SDK;
  - Estudo e implementação da comunicação OPC;
  - Estudo e implementação da cinemática inversa utilizando o método geométrico;
  - Identificação dos eixos utilizando *System Identification Toolbox* MATLAB®;
  - Implementação de controladores PI (proporcional-integral) para cada um dos eixos utilizando comunicação OPC integrada ao *Simulink* MATLAB®;

## 2 Laboratório

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 de pesquisa e extensão ligados a áreas de instrumentação eletrônica, automação e controle.

Entre os anos de 2016 e 2017, o mesmo passara por reformas para melhoria da infraestrutura disponível na realização de pesquisa e extensão e atualmente encontra-se com a fachada tal qual na figura (1). Além disso, a aquisição de recursos de software modernos e atualizados é recorrente bem como treinamento constante de pessoal.



Figura 1 – Faixada das instalações físicas do Laboratório de Instrumentação Eletrônica e Controle

Com uma área de aproximadamente  $600 m^2$ , o LIEC conta com sete laboratórios de desenvolvimento, duas salas de apoio técnico, sala para apresentação de trabalhos, salas para pós-graduação e professores. Antes da expansão, o LIEC contava com os seguintes laboratórios disponíveis:

- Laboratório de Aplicações Wireless: são desenvolvidas soluções baseadas em dispositivos móveis para ambientes industriais;

- Laboratório de Automação Industrial: aborda trabalhos com sintonia de controladores PID industriais (Mono e Multivariável), automação industrial, instrumentação industrial, IHM industrial e avaliação de confiabilidade em malhas de controle;
- Laboratório de Controle e Otimização: produz projetos de sintonia de controladores PID e modelagem e simulação de processos e sistemas supervisórios;
- Laboratório de Instrumentação eletrônica: são desenvolvidos projetos de sintonia de PID;
- Laboratório de Redes Industriais: permite o estudo 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 UltraSom: trabalha-se com o desenvolvimento de sensor de incrustação e desenvolvimento de técnicas de medição de incrustação.

No LIEC, alunos de pós-graduação e de graduação são bem vindos a iniciar pesquisas e projetos, tendo a disposição uma infraestrutura robusta, corpo de professores bem capacitados, técnicos com experiência e alunos com forte espírito de trabalho em grupo.

## 3 Fundamentação Teórica

### 3.1 Padrão de Comunicação OPC

Fato é que as diferenças de padrões de comunicação entre os dispositivos de diferentes fabricantes sempre fora um gargalo no desenvolvimento e na implementação destes no setor industrial. Uma empresa para adotar um determinado produto, deveria portanto adotar todos os dispositivos necessários desta mesma marca de modo a compatibilizar a comunicação (IWANITZ; LANGE, 2001).

Em 1995, desenvolvedores de software e consumidores finais reuniram-se com o objetivo de resolver os problemas de compatibilização de dispositivos que se comunicavam por padrões distintos. Surgira então a OPC Foundation, onde OPC é a sigla para *OLE for Process Control* e OLE significa *Object Linking and Embedding*, que em tradução livre significa *Vinculação e Incorporação de Objetos*.

Quando foi lançado pela primeira vez em 1996, tinha como objetivo oferecer um padrão para que protocolos específicos de diferentes marcas de CLP (Controladores Lógicos Programáveis) conseguissem realizar leitura e gravação em solicitações específicas do dispositivo.

#### 3.1.1 Funcionamento

O funcionamento do OPC é tipicamente como uma relação cliente-servidor onde um ou mais servidores fornecem dados para uma ou mais aplicações cliente, Basicamente, uma aplicação cliente, pede um determinado dado ao servidor OPC que lhe atende e retorna com a informação.

O servidor OPC é dividido em três partes:

- Server - onde são executadas as interfaces entre as aplicações e onde ocorre o controle de eventos e alarmes;
- Group - fica em uma camada superior, é onde os itens são organizados e onde ocorre o controle de atualização dos valores;
- Item - representa uma variável específica do sistema, além do valor da variável, possui informações sobre a qualidade da informação;

Os servidores são softwares que fornecem dados no padrão OPC e o computador é o hardware convergente e que disponibiliza os dados.



Dizemos que os *clientes OPC* serão aqueles que receberão os dados e podem estar em quaisquer computadores conectados à rede do servidor OPC.

Na figura (2) temos um exemplo da arquitetura de uma rede OPC.

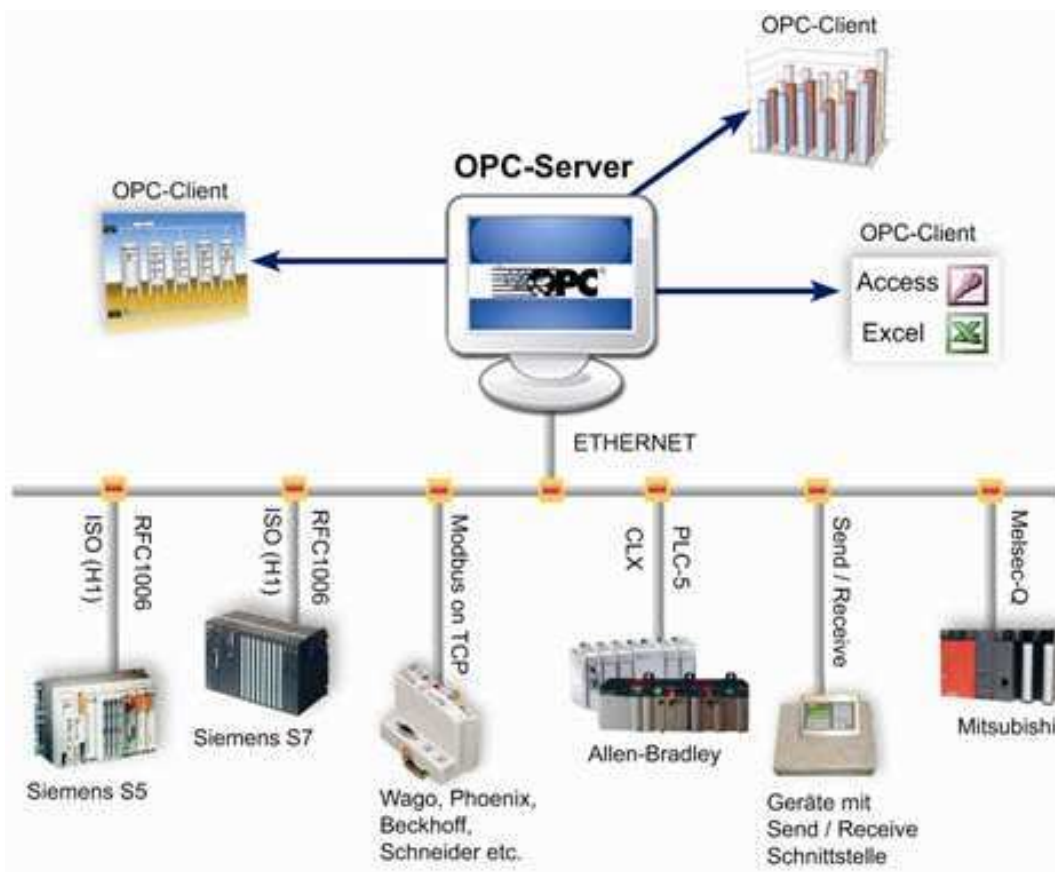


Figura 2 – Arquitetura representativa de uma rede OPC.

## 3.2 Cinemática Direta e Inversa

Quando sabemos todos os ângulos das juntas, facilmente determinamos o ponto de interesse. Entretanto, um dos problemas ao trabalhar com robótica é que dado uma posição de interesse, como saber quais os ângulos corretos devem ser atribuídos a cada uma das juntas? Este problema é chamado de cinemática inversa. No método geométrico, utiliza-se a geometria euclidiana na determinação dos ângulos a serem atingidos a partir de uma coordenada de referência para o ponto de interesse.

### 3.2.1 Cinemática Direta

Na figura (3) é possível observar que a decomposição do vetor  $\vec{L}_1$  no plano XY e no eixo z dos leva as equações (3.1) e (3.2)

$$\vec{L}_1 = \text{sen}\phi.\cos(\theta_1 L_1)\widehat{a}_x \quad (3.1)$$

$$\vec{L}_2 = \text{sen}\phi.\cos(\theta_1 + \theta_2)L_2\widehat{a}_x + \cos\phi.\cos(\theta_1 + \theta_2)L_2\widehat{a}_y + \text{sen}(\theta_1 + \theta_2)L_2\widehat{a}_z \quad (3.2)$$

Que nos levam as equações (3.3), (3.4) e (3.5):

$$X_p = \text{sen}\phi.(L_1\cos\theta_1 + \cos(\theta_1 + \theta_2)L_2) \quad (3.3)$$

$$Y_p = \cos\phi.(L_1\cos\theta_1 + \cos(\theta_1 + \theta_2)L_2) \quad (3.4)$$

$$X_p = L_1\text{sen}\theta_1 + \text{sen}(\theta_1 + \theta_2)L_2 \quad (3.5)$$

A cinemática direta não representa um grande problema matemático, uma vez que sabendo os ângulos, facilmente podemos calcular quais coordenadas finais do sistema.

### 3.2.2 Cinemática Inversa

Uma vez que trata-se de um dispositivo físico, com dimensões, é possível, através de uma análise geométrica, relacionar os pontos de interesse com os ângulos de cada junta. Para tanto, observe a figura (3).

Através da figura, é possível deduzir o conjunto de equações (3.6) (3.7) (3.8) utilizando-se dos conceitos de geometria analítica e álgebra vetorial:

$$\theta_1 = \tan^{-1}\left(\frac{\sqrt{Y^2 + X^2}}{Z}\right) - \cos^{-1}\left(\frac{Y^2 + X^2 + Z^2 + L_1^2 - L_2^2}{2L_1\sqrt{Y^2 + Z^2 + X^2}}\right) \quad (3.6)$$

$$\theta_2 = \cos^{-1}\left(\frac{Y^2 + X^2 + Z^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \quad (3.7)$$

$$\Phi = \tan^{-1}\left(\frac{X}{Y}\right) \quad (3.8)$$

Verifica-se na própria interface e, de forma intuitiva que outra disposição do manipulador quase sempre é possível de ser realizada, levando o sistema para o mesmo ponto

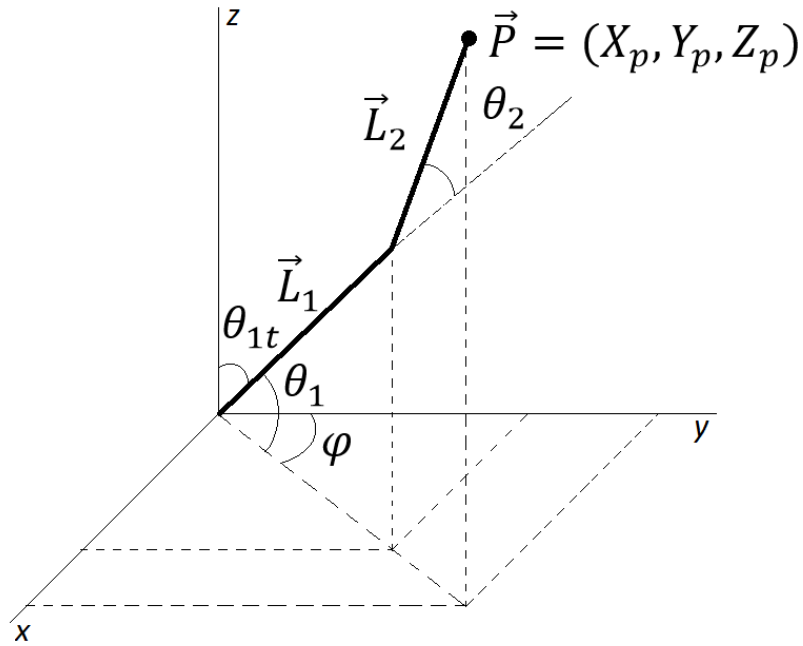


Figura 3 – Ilustração das variáveis de ângulos e de coordenadas

de interesse. Isto é, temos um sistema de múltiplas soluções, mas que apresenta um equacionamento insuficiente para representá-lo. Para tanto, observe a figura (4):

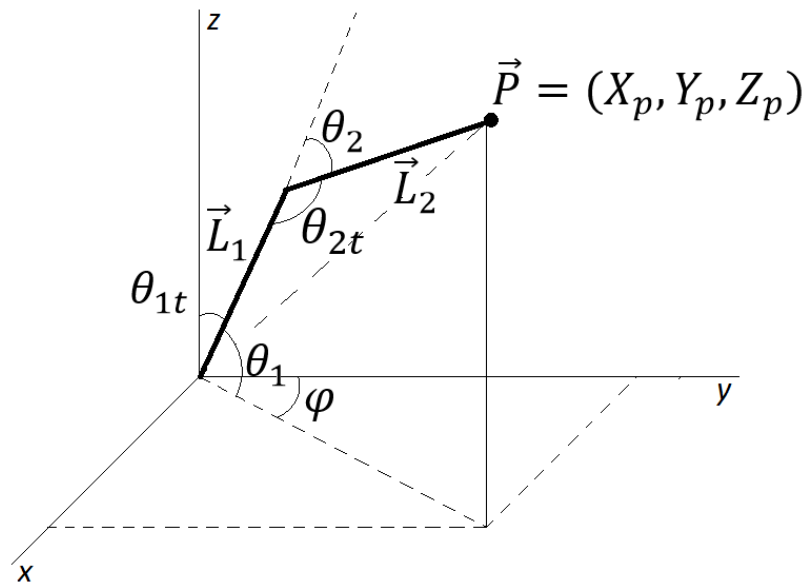


Figura 4 – Ilustração das variáveis de ângulos e de coordenadas em um posicionamento alternativo

A partir da figura (4) é então possível, em conjunto com as equações já encontradas, definir um novo conjunto de equações capaz de representar o sistema com uma melhor

fidelidade, como as equações a seguir (3.9) (3.10) (3.11):

$$\theta_1 = \tan^{-1} \left( \frac{\sqrt{Y^2 + X^2}}{Z} \right) \mp \cos^{-1} \left( \frac{Y^2 + X^2 + Z^2 + L_1^2 - L_2^2}{2L_1 \sqrt{Y^2 + Z^2 + X^2}} \right) \quad (3.9)$$

$$\theta_2 = \pm \cos^{-1} \left( \frac{Y^2 + X^2 + Z^2 - L_1^2 - L_2^2}{2L_1 L_2} \right) \quad (3.10)$$

$$\Phi = \tan^{-1} \left( \frac{X}{Y} \right) \quad (3.11)$$

## 4 A Plataforma

A plataforma robótica didática que está instalada fora adquirida pelo LIEC em 2013 quando ocorrera uma movimentação do coordenador do laboratório em desenvolver uma abordagem à robótica para o curso de engenharia elétrica da UFCG.



Figura 5 – Plataforma instalada no LIEC

Fato é que a plataforma adquirida, apesar de limitações quanto a aplicações industriais, apresenta vantagens para introduzir o aluno no estudo da robótica. Estas vantagens estão associadas ao hardware adotado e também aos recursos de software disponíveis, descritos a seguir.

### 4.1 Recursos de Hardware

#### 4.1.1 Servos Inteligentes

A empresa sul-coreana ROBOTIS produz uma série de servos denominados inteligentes para o público. Estes relacionam-se devido a capacidade de torque e de velocidade, como pode ser visto na figura (6):

O AX-12A (servo selecionado) trata-se de um atuador Dynamixel inteligente e 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 dispositivo. Apesar do seu

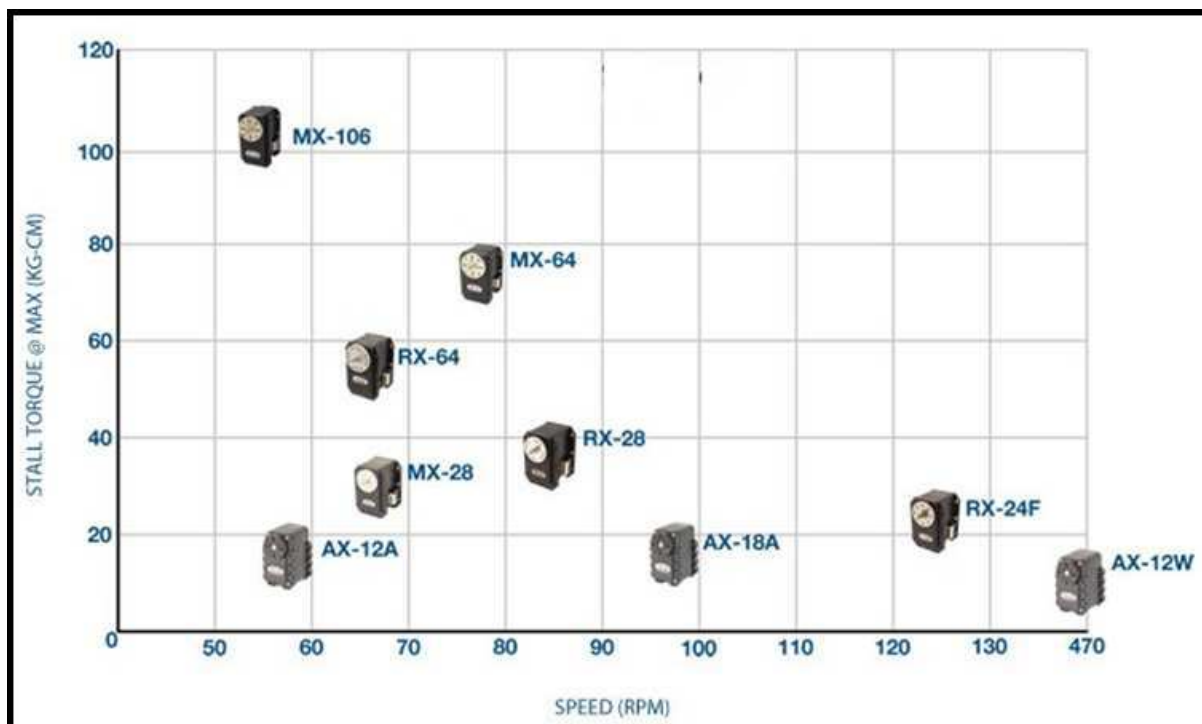


Figura 6 – Distribuição dos servos de acordo com capacidade de Torque e Velocidade

tamanho compacto, o AX-12A pode desenvolver um torque elevado comparado a outros do mesmo segmento e é feito com materiais de elevada qualidade de forma a proporcionar a resistência e robustez estrutural necessária para suportar grandes forças externas para seu porte. Além disso, o AX-12A tem a capacidade de detectar e agir a condições internas, como mudanças na temperatura interna ou tensão de alimentação.

Possui ainda um controle primitivo de precisão de 1024 níveis com faixa de posicionamento de  $300^\circ$  (i.e., uma resolução de  $0.2930^\circ/\text{nível}$ ) possibilidade de trabalhar com velocidades de comunicação de até 1Mbps. O preço em relação aos demais dispositivos listados na figura (6) fora um diferencial na escolha.

Na tabela (1) é possível observar as especificações técnicas mais básicas do servo adotado, sendo estas essenciais para correta utilização do dispositivo.

Internamente, é possível visualizar a presença de sensores específicos e estudá-los. Na figura (7), observa-se a estrutura eletrônica do servo-mecânismo.

Com uma inspeção rápida, observam-se partes do Servo Motor:

- Circuito de Controle – responsável pelo monitoramento do potenciômetro e acionamento do motor visando obter uma posição pré-determinada;
- Potenciômetro – ligado ao eixo de saída do servo, monitora a posição do mesmo;
- Motor – movimenta as engrenagens e o eixo principal do servo Ax-12A;

Tabela 1 – Especificações do AX-12A

AX-12	
Peso(g)	55
Taxa de redução	1/254
Tensão (V)	7 10
Torque Max. (kgF.cm)	12 16.5
Resolução (°)	0,2930
Faixa de operação (°)	0 a 300
Alimentação (V)	7 10 (Recomandado de 9,6)
Corrente Máxima (mA)	900
Temperatura de operação (°)	-5 +85
Tipo de Protocolo	Serial <i>Half-Duplex Assin.</i> (8 bit, 1parada, sem paridade)
ID	254 (0 254 ID)
Velocidade de Comun.	7343 bps 1 Mbps



Figura 7 – Componentes eletrônicos do servo AX-12A

- Engrenagens – reduzem a rotação do motor, transferem mais torque ao eixo principal de saída e movimentam o potenciômetro junto com o eixo como visto na figura (8);
- Caixa do Servo – caixa para acondicionar as diversas partes do servo;

Dentre os sensores observados, o mais utilizado nos testes fora o sensor de posição, tratando-se apenas de um potenciômetro de nome MURATA SV01, visualizado na figura (9). O sensor de posição é na verdade um potenciômetro rotativo de  $10\Omega$  que baseia-se no princípio da variação da resistência conectada mecanicamente ao eixo, onde a diferença de potencial nos terminais do potenciômetro é proporcional ao ângulo de rotação de um rotor, onde o ângulo de rotação pode ser facilmente encontrado pela medição da tensão.

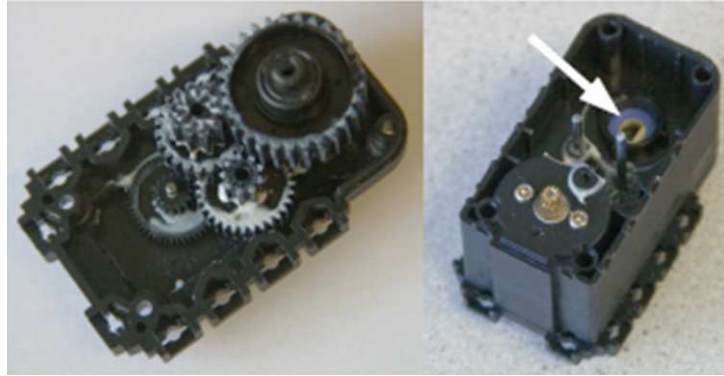


Figura 8 – Caixa de engrenagens redutoras do servo AX-12A

As aplicações vão desde controladores de joystick até sistemas de controle de velocidade, de articulação de robôs e de medição de inclinações.

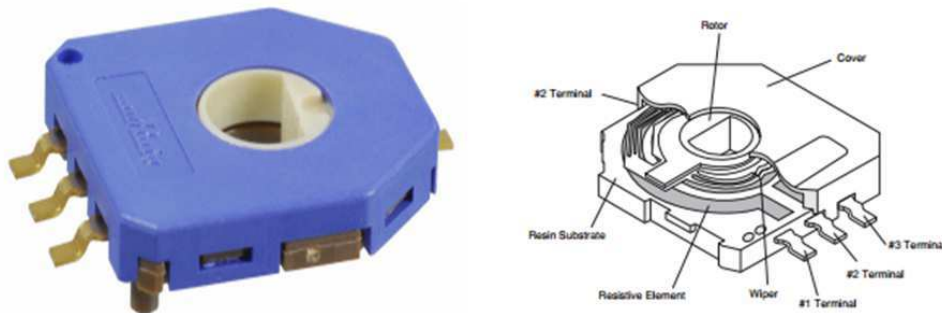


Figura 9 – Potenciômetro Murata SV01 utilizado como sensor de posição angular do servo AX-12A

Embora o sensor de posição angular possa variar de  $0^\circ$  –  $320^\circ$  linearmente, a fabricante do servo resolveu limitar para a faixa de  $0^\circ$  –  $300^\circ$ . A leitura do sensor vai para o conversor A/D do microcontrolador interno, onde pode-se obter tanto a velocidade como a aceleração por diferenciação. A utilização de capacitores na saída de sensores deste tipo tipicamente suaviza a resposta obtida.

O servo-mecânismo inteligente AX-12A ainda detém um conjunto de memórias EEPROM e RAM, que são responsáveis por armazenar os valores lidos e guardar parâmetros de controle, como podemos ver nas tabelas a seguir.

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 (10), é possível contemplar a geometria do AX-12A assim como a disposição de seus terminais.

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



Tabela 2 – Posições da memória EEPROM e suas respectivas faixas de valores

Endereço	Item	Endereço	Item
0x03	ID	0x18	Torque Ativo
0x04	Baud Rate	0x19	LED
0x05	Tempo de Atraso	0x1A	Margem Horária
0x06	Limite Angular Horário	0x1B	Margem Anti-horária
0x08	Limite Angular Anti-horário	0x1C	Margem Horária 2
0x0B	Limite Superior de Temperatura	0x1D	Margem Anti-horária 2
0x0C	Limite Inferior de Tensão	0x1E	Destino
0x0D	Limite Superior de Tensão	0x20	Velocidade Medida
0x0E	Torque Máximo	0x22	Limite de Torque
0x10	Status de Retorno	0x2C	Instrução Registrada
0x11	Led de Alarme	0x2F	Trava
0x12	Desligamento do Alarme	0x30	Soco

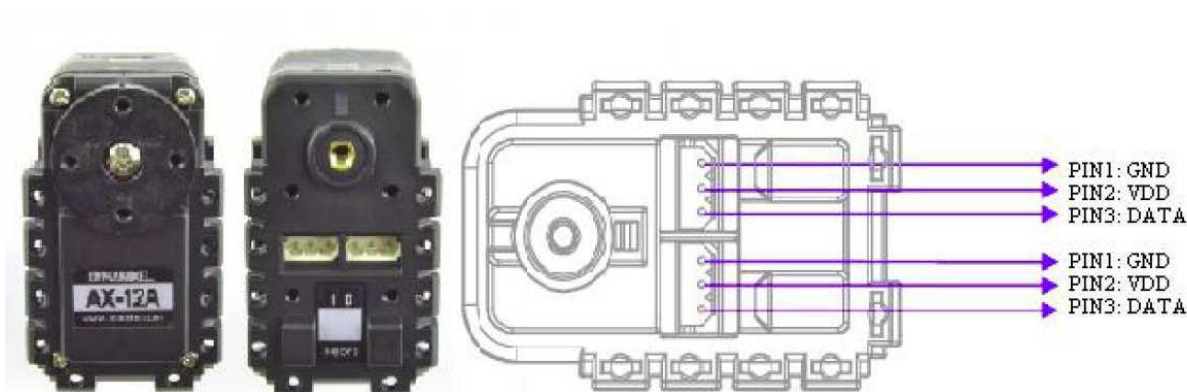


Figura 10 – Característica física do AX-12A.

Status” (enviados do Dynamixel para o controlador principal), como pode ser observado na figura (11).

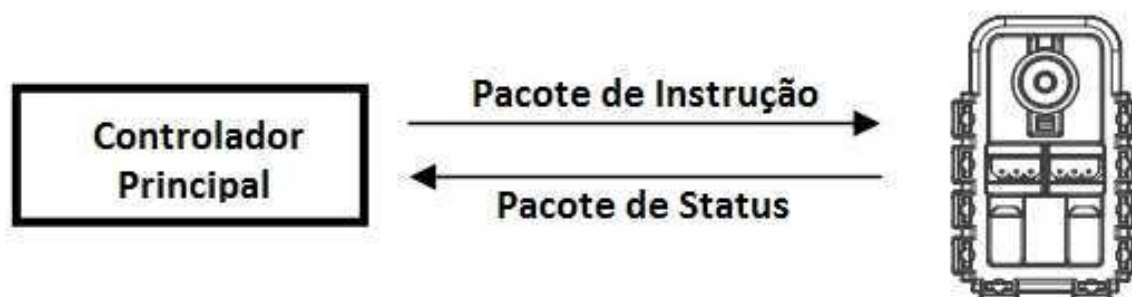


Figura 11 – Comunicação entre controlador e 1 AX-12A.

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.

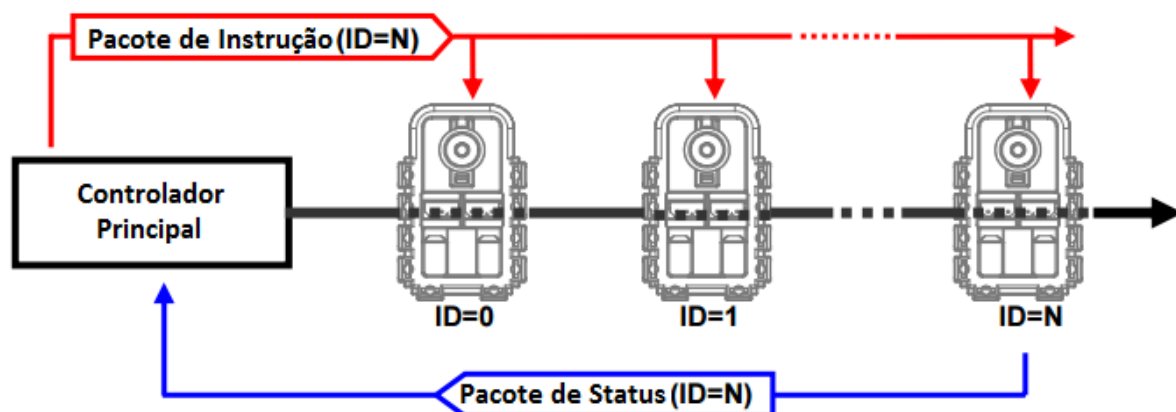


Figura 12 – Comunicação entre o controlador e vários AX-12A.

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

#### 4.1.2 Braço Robótico Inteligente AX-12A

O braço inteligente foi projetado para ser forte e resistente. Construído em alumínio galvanizado, detém 7 atuadores do tipo AX-12A de forma a proporcionar 4 graus de liberdade além da garra e tem imagem original do fabricante como sendo a figura (13).



Figura 13 – Braço robótico inteligente AX-12A.

Quatro dos AX-12A estão emparelhados de forma a constituírem as articulações do punho e ombro. As demais articulações possuem apenas um único Ax-12A. Na base existem quatro esferas de aço para 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 fabricadas 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.

### 4.1.3 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 (14) e (15), mostram o aspecto real e como usar o USB2Dynamixel.

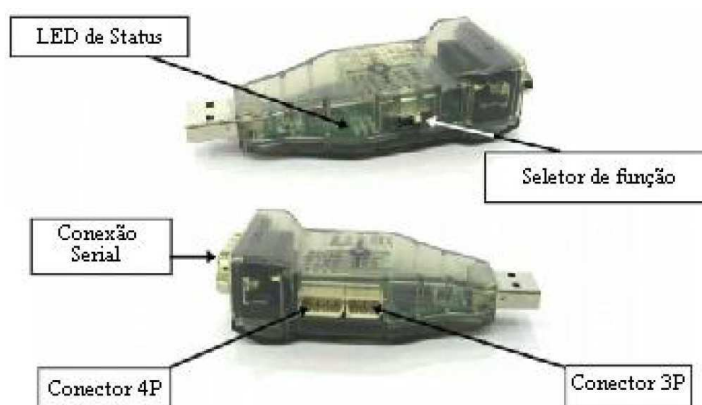


Figura 14 – Dispositivo USB2Dynamixel, utilizando na conversão entre comunicação serial e USB

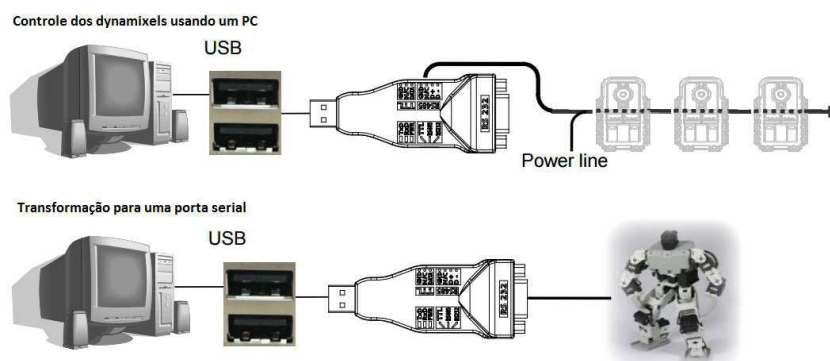


Figura 15 – Ligação entre o servo AX-12A e o computador através do USB2Dynamixel

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, respectivamente às figuras (16), (17) e (18).

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

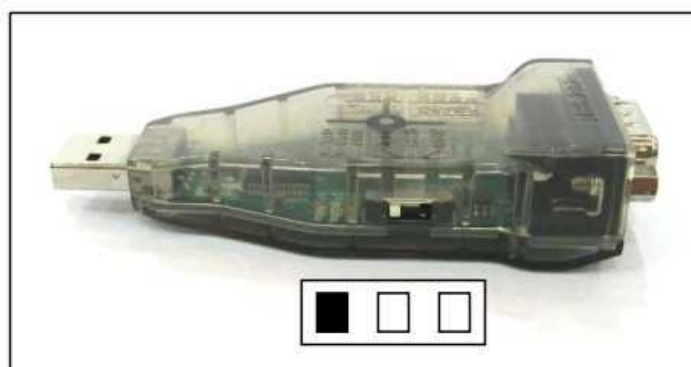


Figura 16 – Seleção de modo de conversão

2ª - 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 17 – Conexão do tipo 3P.

3ª - Os dynamixels podem ser conectados em série usando um cabo 3P, conectando a alimentação ao último dynamixel conforme a figura a seguir:




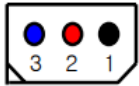
Figura 18 – Conexão da alimentação.

As funções ou modos de comunicações propiciados pelo USB2Dynamixel são:

- Comunicação TTL: dynamixels de 3 pinos tais como os da série AX;
- Comunicação RS485: dynamixels de 4 pinos tais como os das séries DX, RX e EX;
- Comunicação RS232: controladores usando cabo serial tais como o CM-5 e CM-510.

A figura (19) mostra a pinagem do 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

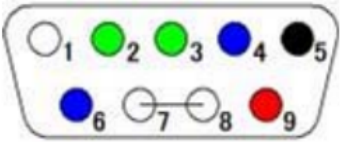
Pin No.	Signal	Pin Figure
1	Data (TTL)	
2	RXD (RS-232)	
3	TXD (RS-232)	
4	D+ (RS-485)	
5	GND	
6	D- (RS-485)	
7	Short with No. 8	
8	Short with No. 7	
9	USB power (5V)	

Figura 19 – Diagrama de conexões do Dynamixel para os tipos de conectores 3P, 4P e Serial

## 4.2 Recursos de Software

### 4.2.1 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. Neste caso específico, fora utilizada a linguagem C# para elaboração de uma interface gráfica.

O correto uso da biblioteca Dynamixel SDK ([ROBOTIS...](#)) é descrito 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 projeto, que fará o controle do braço, os arquivos `dynamixel.dll`, `dynamixel.h`.

As funções disponíveis na biblioteca são:

- `dxl_terminate()` - Fecha a comunicação entre o PC e os dynamixel's;
- `dxl_initialize(int port, int baud)` - Inicia a comunicação entre o PC e os dynamixel's;
- `dxl_get_result()` - Verifica o último pacote recebido;
- `dxl_ping(int id)` - Verifica a existência de um servo com o ID sugerido;
- `dxl_read_byte(int id, int address)` - Solicita do servo com ID igual a "id" o byte localizado na posição de memória "address";
- `dxl_write_byte(int id, int address, int value)` - Solicita do servo com ID igual a "id" a escrita do valor "value" no byte localizado na posição de memória "address";
- `dxl_read_word(int id, int address)` - Solicita do servo com ID igual a "id" a palavra de 2 bytes localizada na posição de memória "address";
- `dxl_write_word(int id, int address)` - Solicita do servo com ID igual a "id" a escrita da palavra de valor "value" de 2 bytes, localizado na posição de memória "address";

Uma vez configurada a biblioteca, e devidamente instalada no computador, foram utilizados encapsulamentos a partir de classes de modo que os servos foram dispostos por eixos, como na tabela (3).

Tabela 3 – Organização dos servos de acordo com o eixo do robô

Eixo	ID dos servos
Base	1
Ombro	2 & 3
Cotovelo	4 & 5
Punho	6
Mão	7

A disposição dos eixos, assim como descritos na tabela (3) pode ser observada na figura (20).

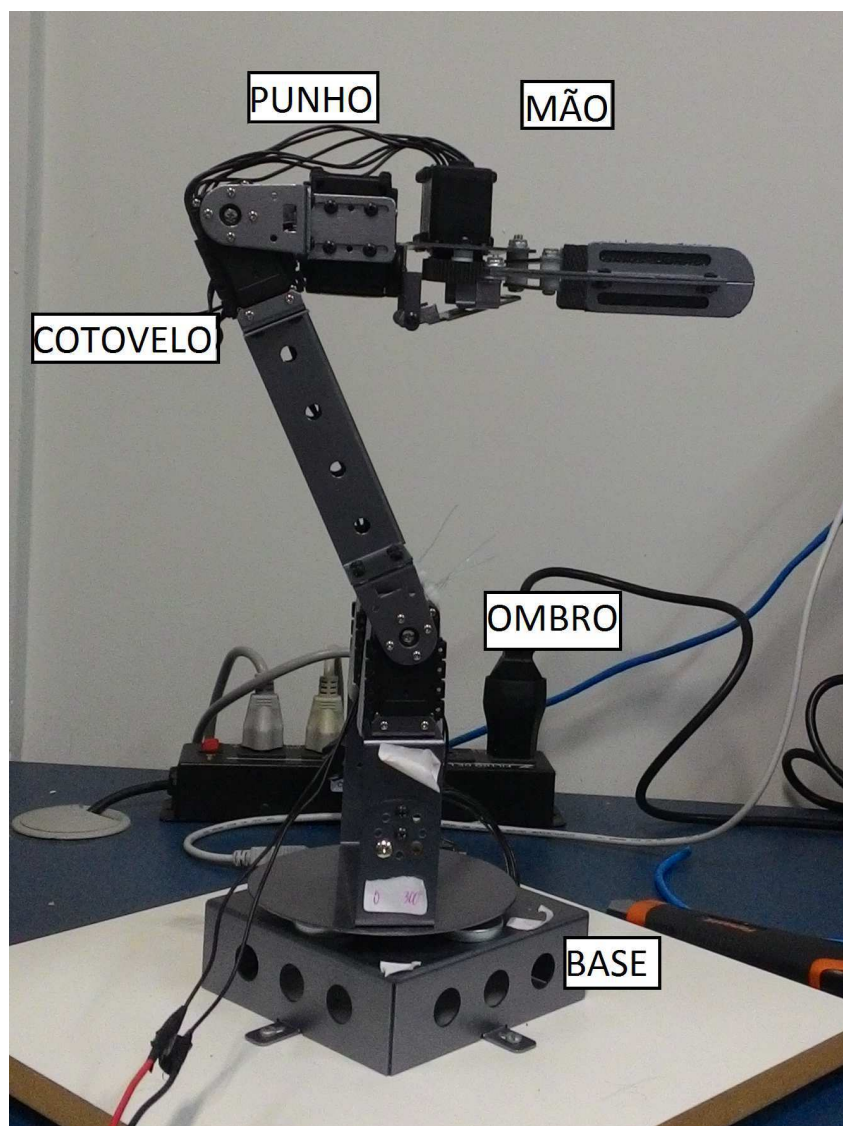


Figura 20 – Disposição dos eixos na plataforma robótica didática

## 5 Atividades Realizadas

### 5.1 Elaboração da interface gráfica em C#

Para a execução dos ensaios, fora elaborado um programa utilizando a linguagem de programação C#, visando aprimorar e tornar mais eficaz a utilização dos servos. Diferentemente da montagem, que permite controlar todos os motores por meio de apenas um fio, optou-se por realizar-se as classes relacionadas a cada eixo de forma individual. Isto se mostrou eficiente para facilitar a construção do programa e posteriormente, permitirá a construção de um controle particular para cada eixo.

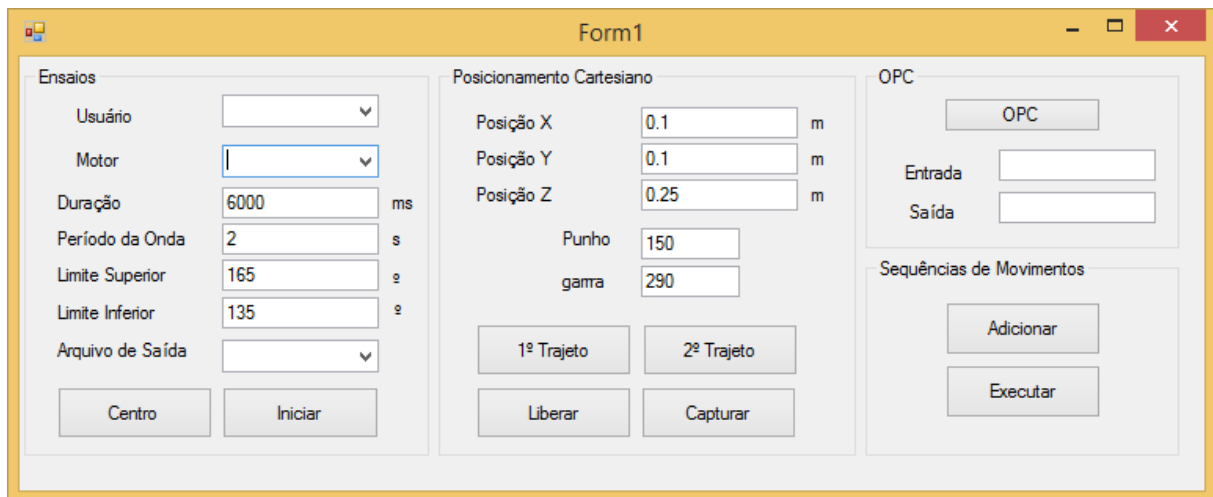


Figura 21 – Interface gráfica elaborada em C#

#### 5.1.1 Primeira e Segunda “GroupBox”

Na primeira GroupBox, intitulada “Ensaio”, é possível selecionar qual eixo excitar utilizando uma onda quadrada. Além disso, é gerado um arquivo de saída, que pode ser exportado como “.m” , para o MATLAB, ou como “.txt”.

Na segunda GroupBox, Posicionamento Cartesiano, temos a implementação das equações de modelagem do robô, tais quais vistas em (ACIOLI; LIMA; BARROS, 2016), permitindo posicionar o braço robótico nas posições desejadas de acordo com coordenadas cartesianas no que diz respeito as juntas e ao ponto de interesse. É notável também a presença das opções por trajeto, uma vez que é possível obter o mesmo posicionamento para o ponto de interesse utilizando trajetos diferentes.

Ainda na GroupBox “Posicionamento Cartesiano”, temos os botões Liberar e Capturar que permitem, respectivamente: Liberar os motores do torque a ser imprimido, e



capturar a posição atual do ponto de interesse, em coordenadas cartesianas, com os respectivos ângulos do motor de punho e do motor da mão robótica.

### 5.1.2 Terceira e Quarta “GroupBox”

Por sua vez, a opção OPC, na GroupBox de mesmo nome, permite utilizar o servidor OPC instalado no computador para gerar diferentes sinais, como senoide, onda quadrada, degrau, RPBS, etc. ou ainda conectar-se com o MATLAB®. Por fim, temos a opção de gerar sequencias desejadas. Com o auxílio do botão liberar, posicionamos o robô na posição de interesse e salvamos em um arquivo “.pos”. E continuamos a adicionar diversas posições, até que o usuário ache suficiente. Desse modo, ao pressionar Executar, o robô reexecutará todos os passos salvos no arquivo.

Os arquivos gerados a partir da execução de um ensaio eram do formato a seguir:

```

k1c          133.69          ];
clear all   133.69          Time = [
close all   133.98          0
dados = [   133.69          19
133.98      133.69          31
133.98      133.69          52
133.98      133.69          61
133.98      133.69          77
133.98      133.69          98
133.98      133.69          108
133.98      ];           124
133.98      Setpoints = [ 145
134.56      135           155
134.85      165           171
134.85      165           186
134.85      165           206
135.14      165           217
135.72      165           237
136.3       165           258
136.59      165           265
136.88      165           285
137.46      165           296
138.04      165           311
138.62      165           332
138.91      165
139.49      165

```

## 5.2 Identificação

### 5.2.1 Através da interface em C#

Em 2008, Arno Mensink realizou uma série de ensaios envolvendo os servos AX-12a. Estes ensaios incluíram ensaio de posição, velocidade e torque, com ou sem a presença de uma carga. Visando elaborar um modelo de representação para estes servos, com objetivo de posteriormente produzir um controle em malha fechada, ensaios baseados no trabalho de Mensink foram reproduzidos, de modo a observar o comportamento dos servos quando submetidos a diferentes situações, como atuação conjunta com outro servo de mesmo eixo, ou ainda à operação quando submetido a altas cargas.

Por sua vez, nos ensaios realizados, utilizou-se um sinal de onda quadrada com 3 ciclos, para cada motor. Esse sinal oscilava entre  $\pm 10\%$  em torno de uma posição central. Para cada eixo estudado, observa-se o seguinte comportamento, nas figuras (22), (23), (24), (25) e (26):

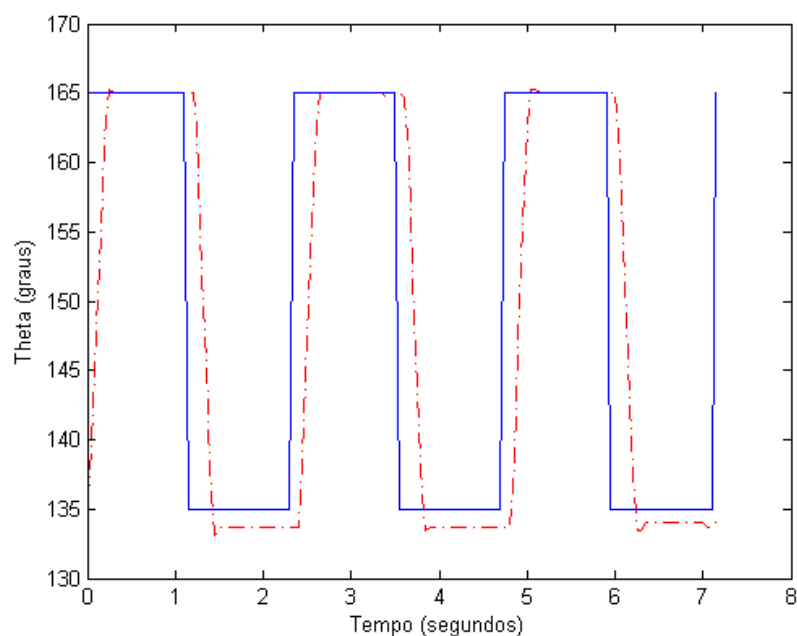


Figura 22 – Onda quadrada aplicada (azul) ao servo da base e resposta obtida (vermelho)

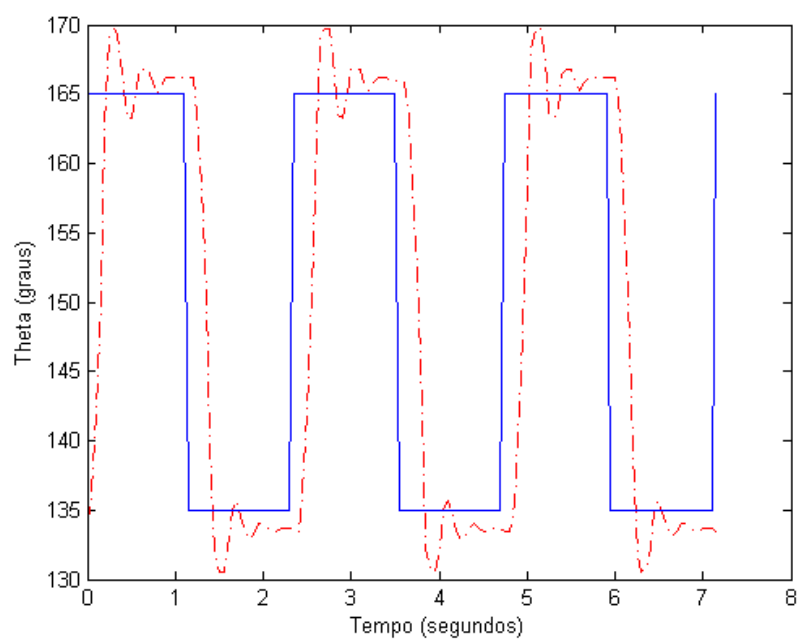


Figura 23 – Onda quadrada aplicada (azul) ao servo do ombro e resposta obtida (vermelho)

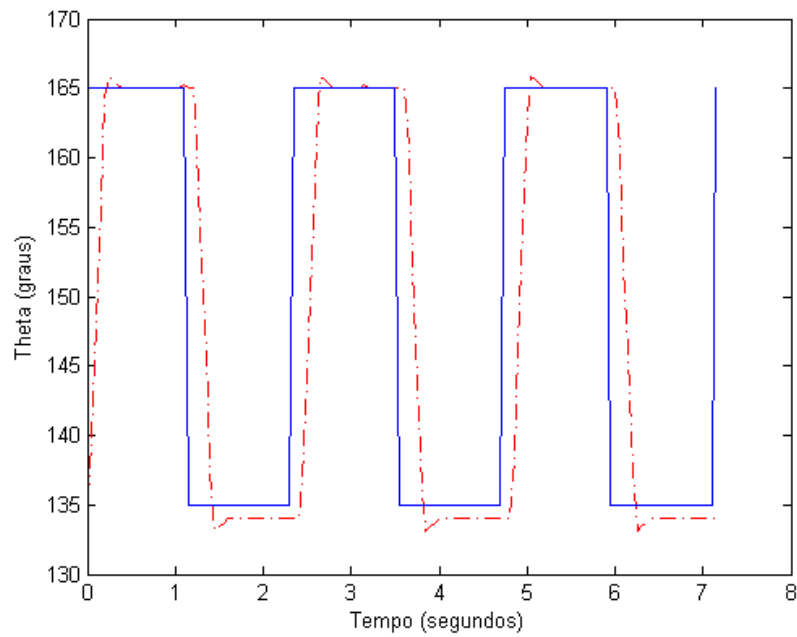


Figura 24 – Onda quadrada aplicada (azul) ao servo do cotovelo e resposta obtida (vermelho)

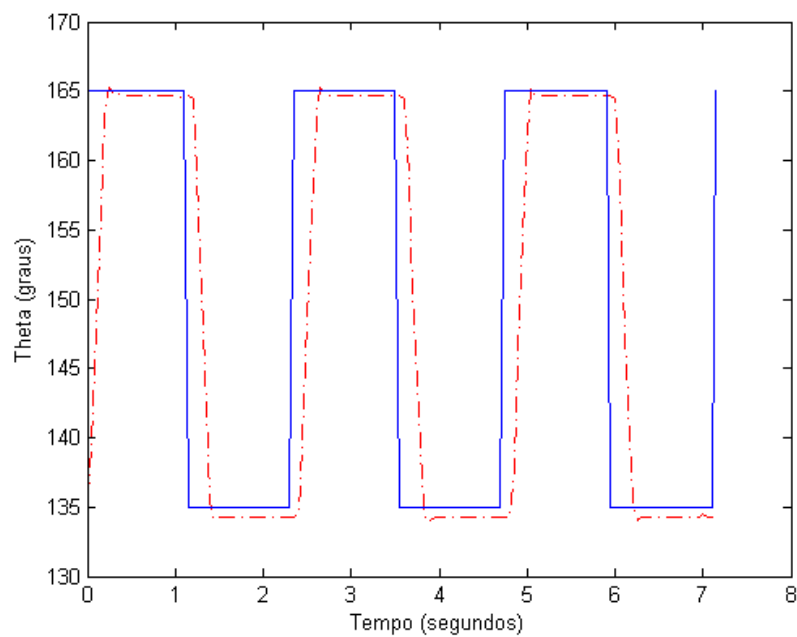


Figura 25 – Onda quadrada aplicada (azul) ao servo do punho e resposta obtida (vermelho)

Para o caso mais preocupante, o ombro, devido as fortes oscilações, decidiu-se realizar uma sequência de experimentos, para a determinação de uma função de transferência que represente o comportamento, a partir do uso da ferramenta *System Identification Toolbox* do MATLAB®. Obtiveram-se as mais convenientes para o estudo as funções de

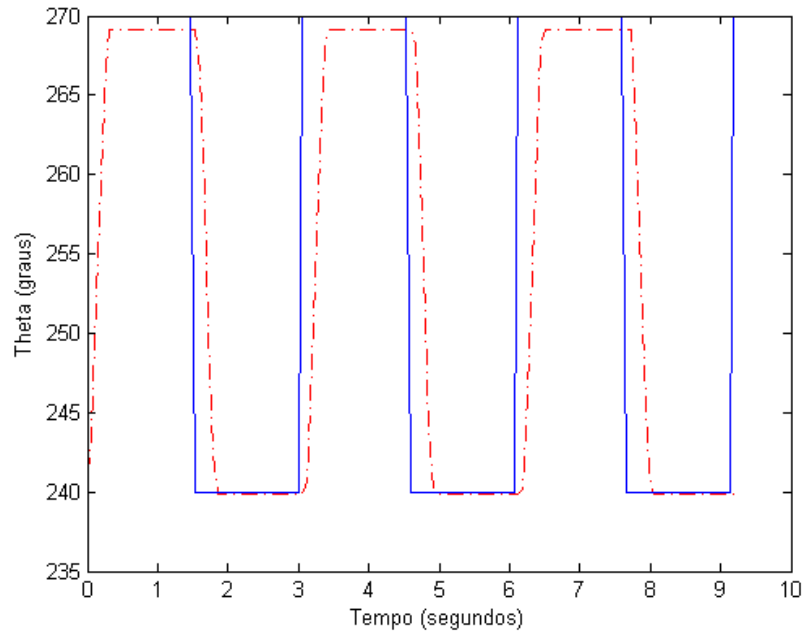


Figura 26 – Onda quadrada aplicada (azul) ao servo da garra e resposta obtida (vermelho) transferência. As funções de transferência obtidas foram (5.1), (5.2), (5.3), (5.4) e (5.5):

$$H_b(s) = \frac{15.16}{s^2 + 6.05s + 15.23} \quad (5.1)$$

$$H_o(s) = \frac{31.95}{s^2 + 7.072s + 31.93} \quad (5.2)$$

$$H_c(s) = \frac{62.93}{s^2 + 11.17s + 63.1} \quad (5.3)$$

$$H_p(s) = \frac{26.93}{s^2 + 7.753s + 27.03} \quad (5.4)$$

$$H_m(s) = \frac{64.04}{s^2 + 25.2s + 64.27} \quad (5.5)$$

Os ajustes obtidos foram respectivamente, de 92.27%, 88.77%, 95.93%, 93.77%, 80.7%. Observa-se ainda, uma necessidade de controladores PI para eliminar erro de regime permanente e obter melhor precisão.

## 5.3 Comunicação OPC

Vislumbrando a utilização do software Simulink MATLAB®, 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.

No MatLab, logrou-se pela OPC *Toolbox*<sup>TM</sup> 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, é possível realizar leituras e escritas, bem como registrar dados OPC de dispositivos, 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*.

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*. Para utilizar a comunicação OPC em conjunto com o Simulink MATLAB®, seguem os passos efetuados durante o estágio:

Na interface C#:

- 1 Na interface construindo em C#, certificar-se que a caixa *OPC On* está marcada e pressionar o botão OPC;
- 2 Na janela recém aberta - denominada OPCClientForm - pressione *Browse Ser.* e em seguida selecione o servidor *LIEC.1000Malhas.V1* e pressione o botão *Connect*. Deve-se observar o aparecimento de 1000 malhas na groupbox Itens OPC, tal qual na figura (27).

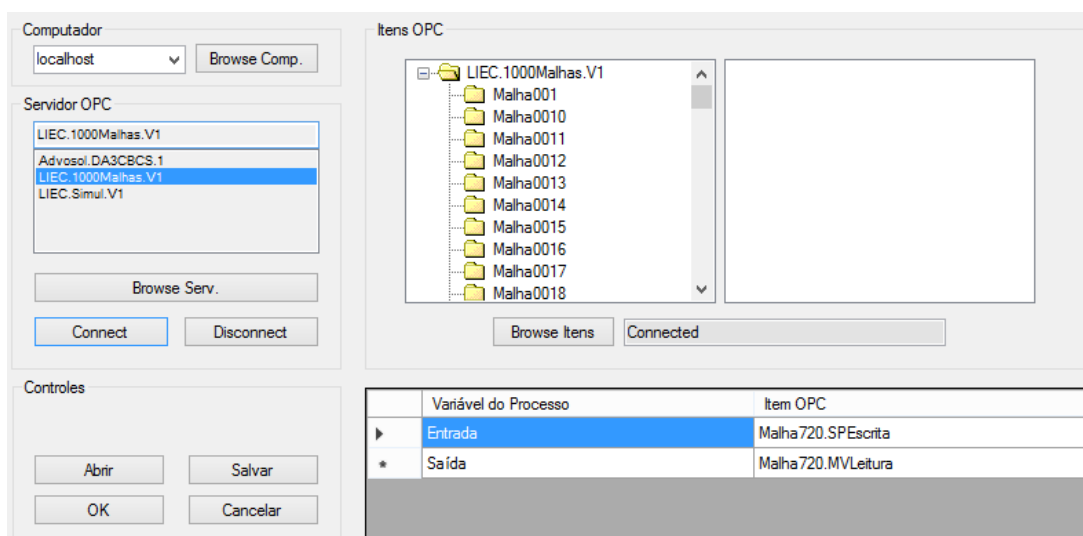


Figura 27 – Janela do cliente OPC da interface C#

- 3 Selecione a malha de interesse e arraste as variáveis de escrita e leitura para os respectivos locais. Neste caso específico, fora selecionada a malha de nome Malha720 e as variáveis de entrada e saída foram respectivamente *Malha720.SPEscrita* e *Malha720.MVLeitura*;
- 4 Pressione *Salvar* para armazenar as escolhas de malha e variáveis e em seguida pressione o botão *OK*;

No ambiente MATLAB ®:

- 1 Na tela inicial do software MATLAB, inserir na linha de comando *simulink* para iniciar o software Simulink;
- 2 Na tela aberta do simulink, pressione para criar um novo modelo, como destacado na figura (28) e em seguida localize a guia de nome *OPC Toolbox* para verificar os blocos disponíveis para a aplicação;

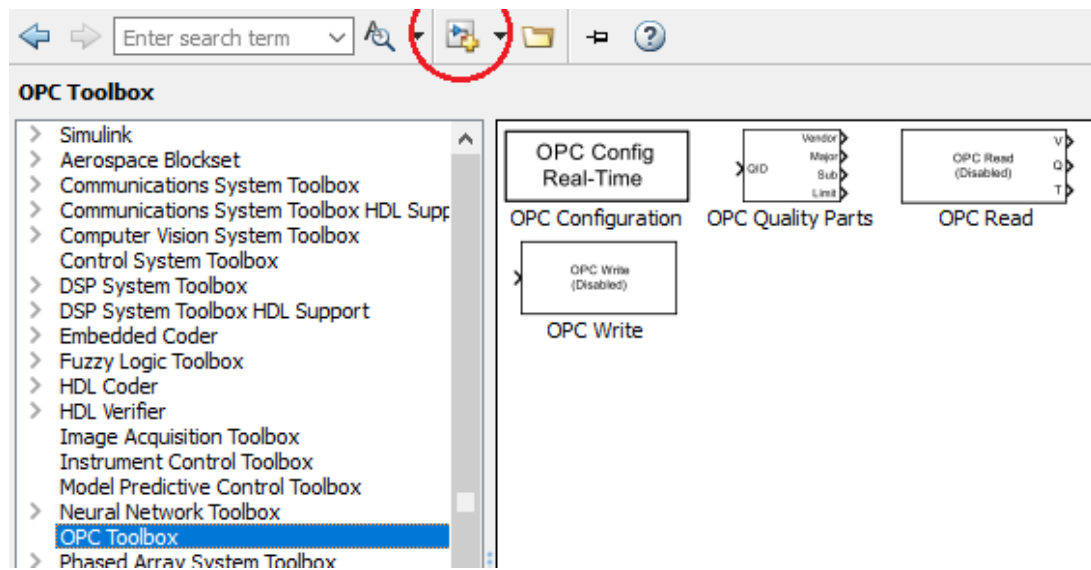


Figura 28 – Criando um novo modelo e localização dos blocos OPC do Simulink

- 3 Arraste os blocos *OPC Configuration*, *OPC Read* e *OPC Write* para o modelo em branco;
- 4 Pressione duas vezes sobre o bloco *OPC Configuration* e garanta que a caixa *Enable pseudo real-time simulation* está habilitada;
- 5 Pressione o botão *Configure OPC Clients ...*;
- 6 Na janela aberta como na figura (29), pressione o botão *Add...* para adicionar uma malha de interesse;

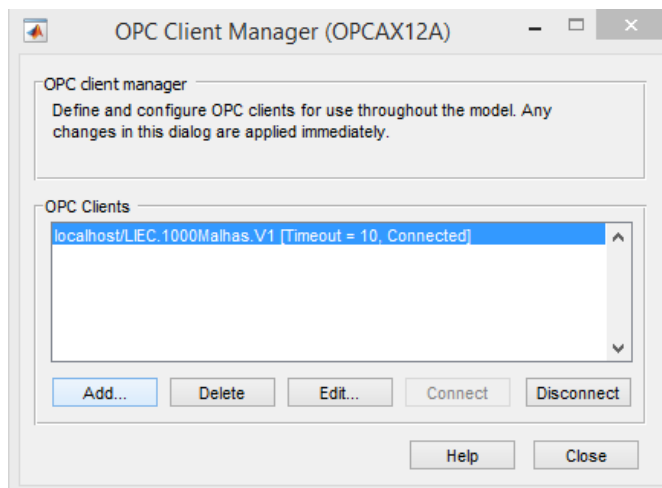


Figura 29 – Adicionando um servidor

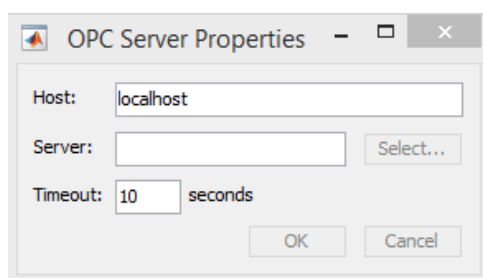


Figura 30 – Selecionando um servidor

- 7 Na janela aberta, como na figura (30), selecione o mesmo servidor de nome *Liec.1000 Malhas.V1* e pressione *OK*;
- 8 Após a configuração do primeiro bloco, pressione duas vezes sobre o bloco *OPC Read* e na groupbox *Itens IDs* selecione *Add Items...* e adicione a variável a ser lida, assim como na figura (31);

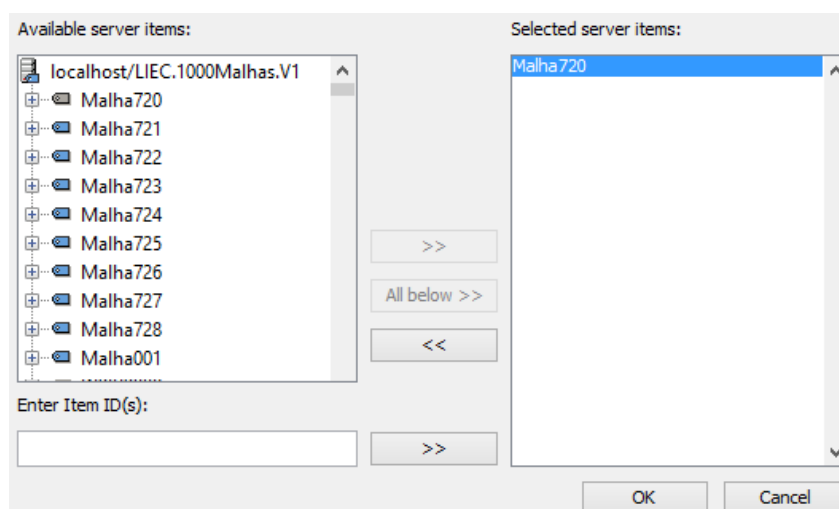


Figura 31 – Adicionando uma malha

Certifique-se ainda que o tempo de amostragem é compatível com o dispositivo em estudo. Neste caso, a variável é *Malha720.MVLeitura*, como na figura (32) e o tempo de amostragem é 16 milissegundos;

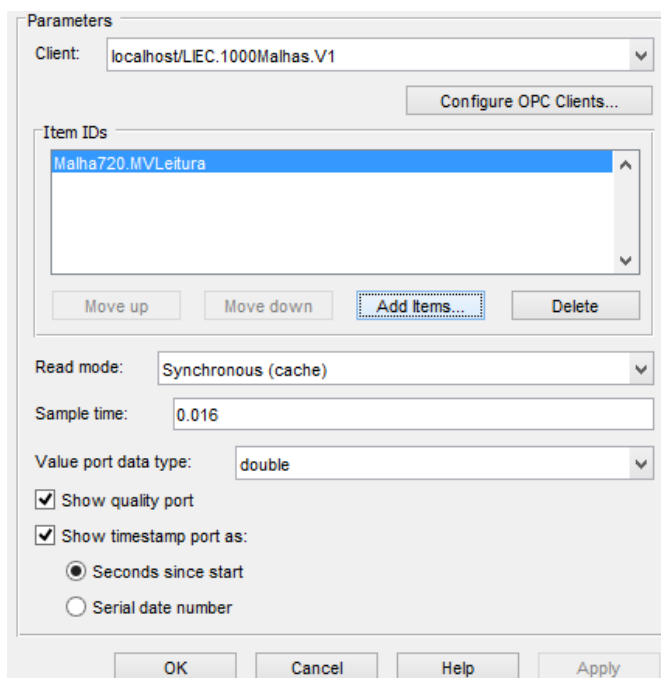


Figura 32 – Adicionando uma variável a ser lida através do bloco *OPC Read*

9 Analogamente ao caso de leitura, no bloco *OPC Write*, pressione-o duas vezes. Na groupbox *Items IDs* selecione *AddItems...* e adicione a variável a ser escrita. Neste caso esta variável é *Malha720.SPEscrita* como na figura (33).

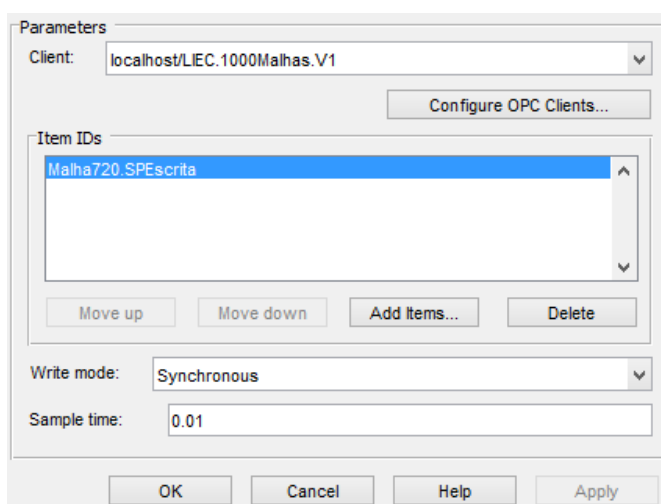


Figura 33 – Adicionando uma variável para escrita através do bloco *OPC Write*



## 5.4 Implementação do Controle

Para implementação do controlador, fora aproveitada a estrutura OPC já construída, e adicionado os blocos nativos do Simulink MATLAB ® apropriados. Foram estes:

- Continuous:
  - PID Controller;
- Math Operations:
  - Sum;
- Signal Routing:
  - Bus Creator;
- Sinks:
  - Scope;
- Sources:
  - Constant;
  - Step;

Montando-se então, a estrutura de fluxo de dados da figura(34).

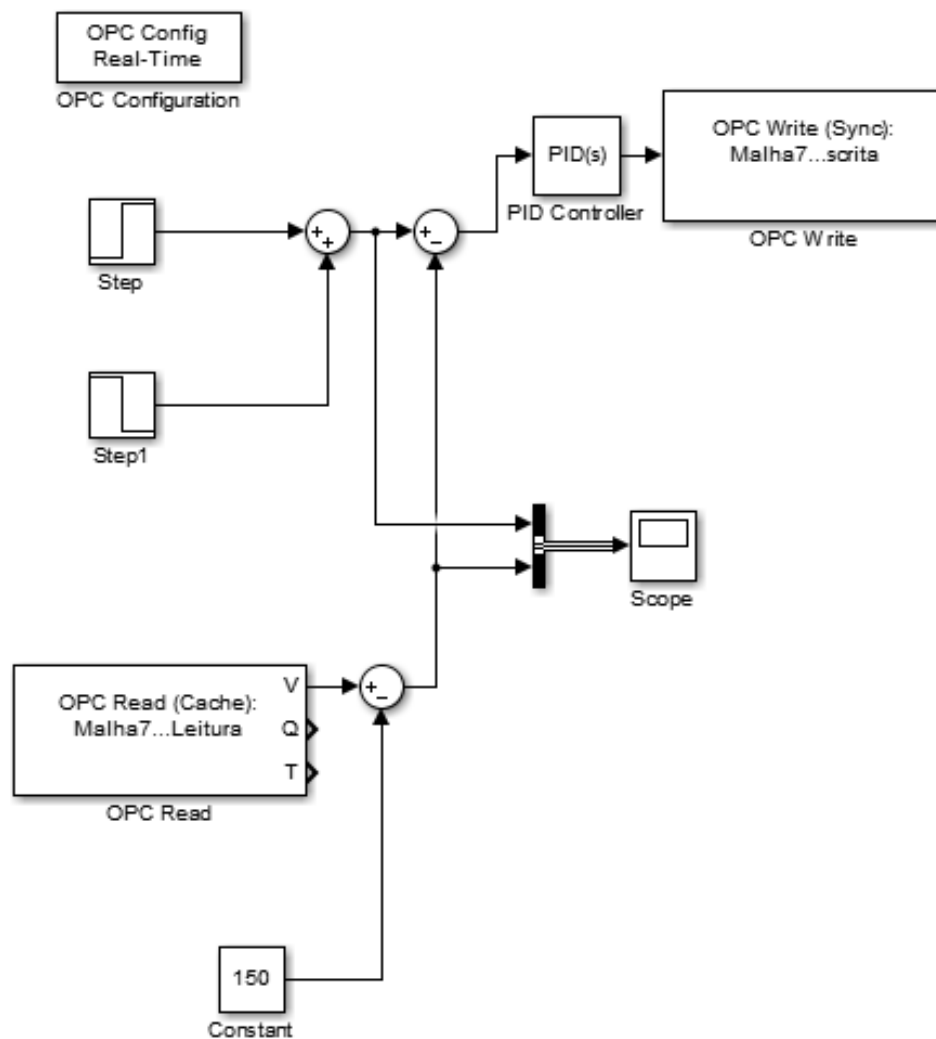


Figura 34 – Diagrama de blocos para implementação do controlador PI por eixo.

### 5.4.1 Eixo Base

Para o eixo denominado base, verificou-se a necessidade de um controlador uma vez que são recorrentes as situações onde há erro de regime. Como podemos ver na figura (35), existe um pico nos sinais recebidos que não condiz com o fenômeno físico e sim ao desgaste do sensor do servo em questão.

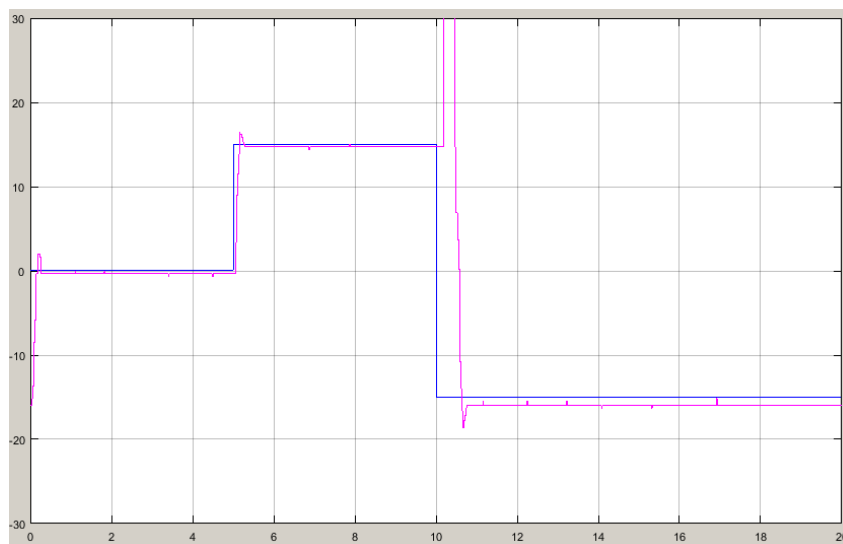


Figura 35 – Servo do eixo base sem controlador

Priorizando um baixo erro de regime uma resposta sobreamortecida, foi projetado um controlador PI de ganhos  $K_p = 0.3394$  e  $K_i = 1.484$ , que garantisse erro de regime nulo e poucas ou nenhuma oscilações, nos levando à resposta da figura (36).

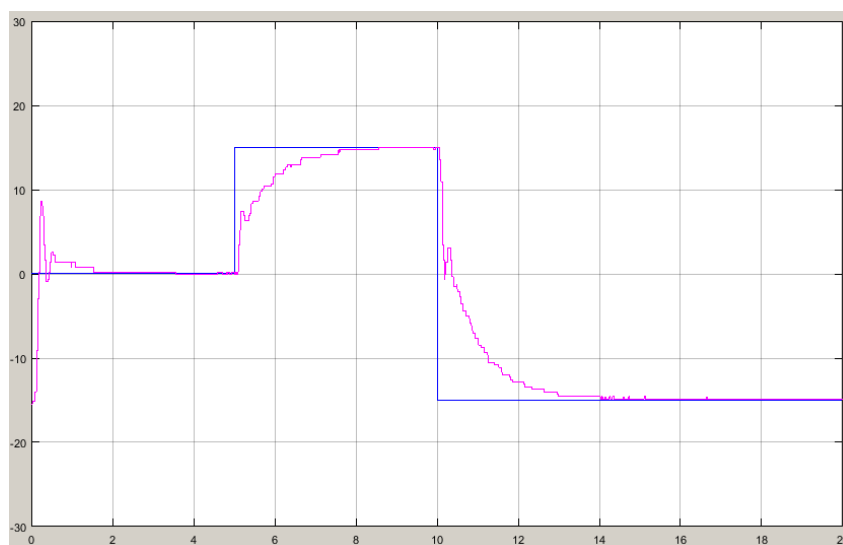


Figura 36 – Servo do eixo base com controlador

### 5.4.2 Eixo Ombro

Para o eixo denominado ombro, a necessidade de um controlador estava associada ao controle das fortes oscilações, que podem ser observadas na figura (37).

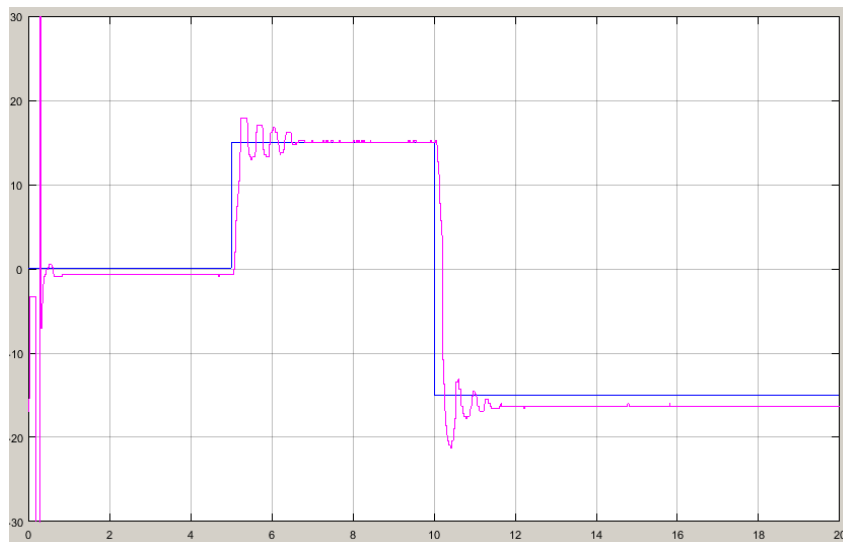


Figura 37 – Servos do eixo ombro sem controlador

Mais uma vez, fora priorizado o erro de regime e excepcionalmente a atenuação das oscilações, ainda que o tempo para atingir o valor de referência fosse elevado. Desse modo, foram calculados os ganhos  $K_p = 0.1323$  e  $K_i = 1.587$  para um controlador PI, nos levando à resposta da figura (38).

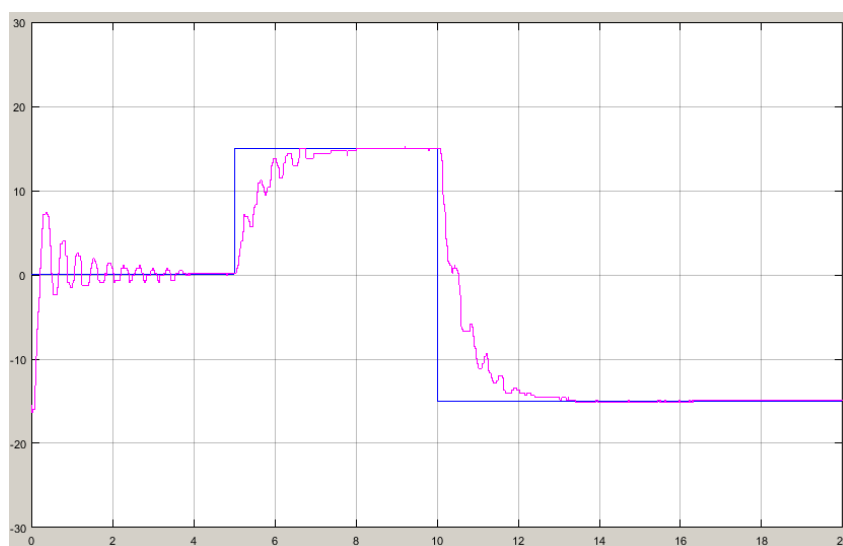


Figura 38 – Servos do eixo ombro com controlador

### 5.4.3 Eixo Cotovelo

Para o eixo denominado cotovelo, a necessidade controlador estava associada ao erro de regime uma vez que as oscilações tem baixa amplitude e rapidamente cessam, que podemos ser observar na figura (39).

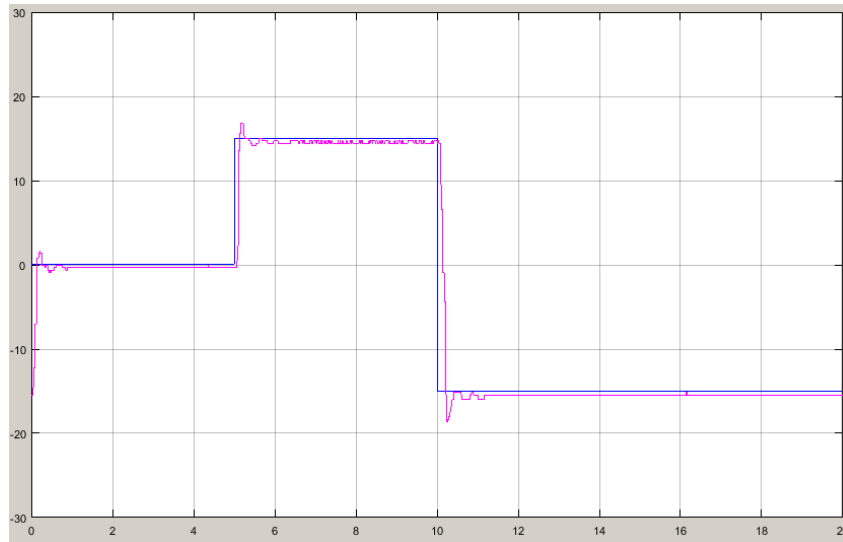


Figura 39 – Servos do eixo cotovelo sem controlador

Ainda que apresentasse uma resposta mais *amena* do ponto de vista do erro e das oscilações, implementou-se um controlador PI com parâmetros  $K_p = 0.1692$  e  $K_i = 2.255$  com o intuito de anular o erro de regime, como podemos ver na figura (40). Apesar de oscilações de amplitude ainda menor, considerou-se que o controlador fora efetivo.

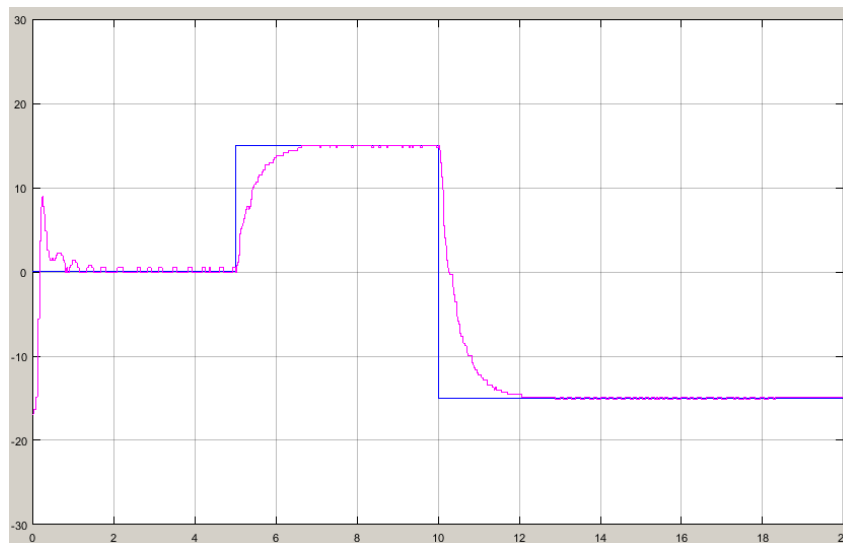


Figura 40 – Servos do eixo cotovelo com controlador

#### 5.4.4 Eixo Punho

Para o eixo punho, temos um diferencial. Trata-se, entre os demais eixos, o que sobre menos carga durante a operação. Dessa forma, é também o que apresente menos oscilações e menor erro de regime, como podemos ver na figura (41).

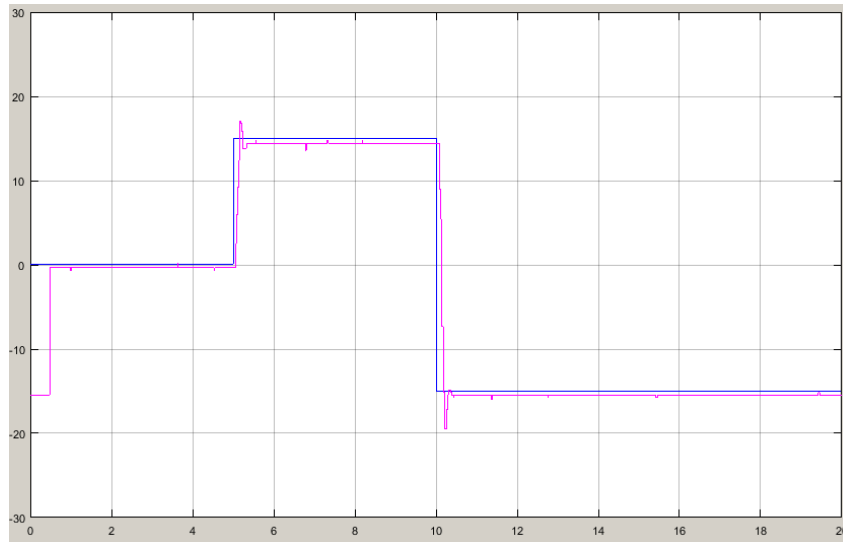


Figura 41 – Servo do eixo punho sem controlador

Ainda que tivesse uma resposta satisfatória, desejou-se melhorar os resultados obtidos, e fora implementado um controlador PI com parâmetros  $K_p = 0.4806$  e  $K_i = 2.084$ , que proporcionou uma resposta de erro de regime nulo e poucas oscilações como na figura (42).

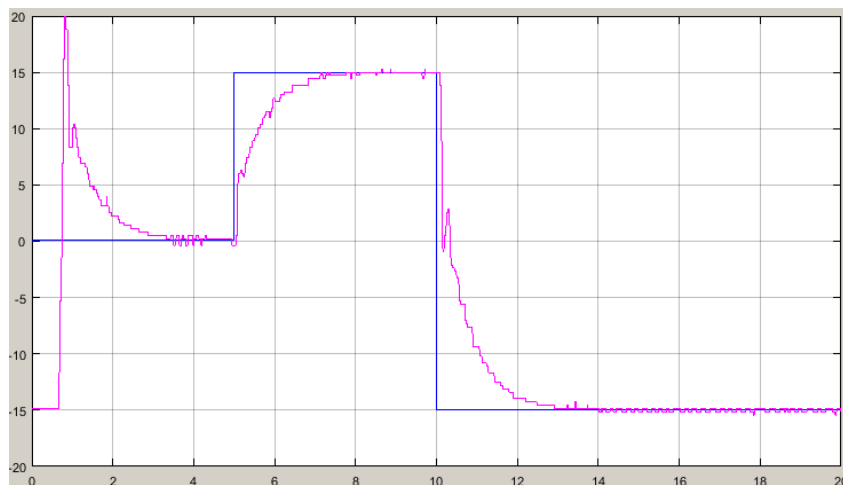


Figura 42 – Servo do eixo punho com controlador

### 5.4.5 Eixo Mão

Para o eixo mão, temos outra situação problemática. Apesar do pequeno erro de regime, este ainda existe e sinaliza ao projetista que um controlador PI é necessário, como pode-se ver na figura (43).

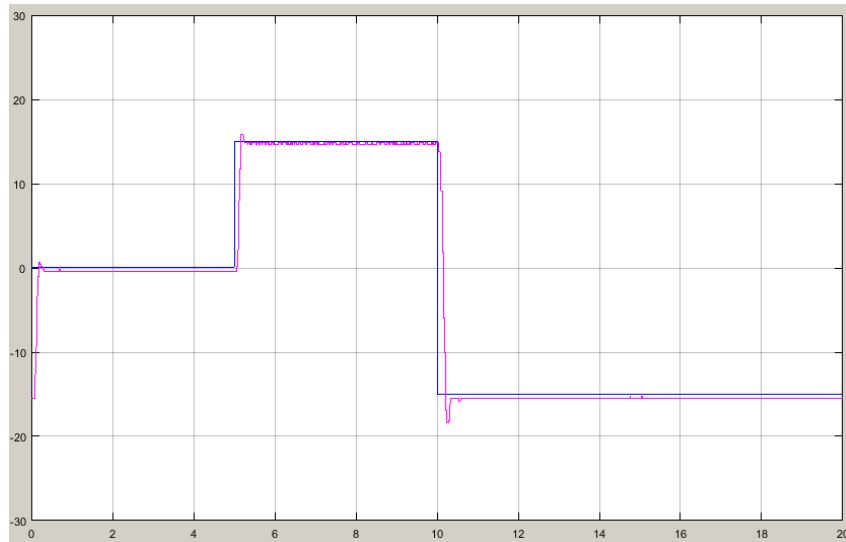


Figura 43 – Servo do eixo mão sem controlador

Entretanto, para os controladores projetados, a resposta apresentara uma dinâmica tanto quanto errática, com níveis de oscilações estranhos para um dispositivo com resposta sobreamortecida. Os ganhos do controlador usado para obter a resposta da figura (44) foram  $K_p = 0.7234$  e  $K_i = 1.553$ .

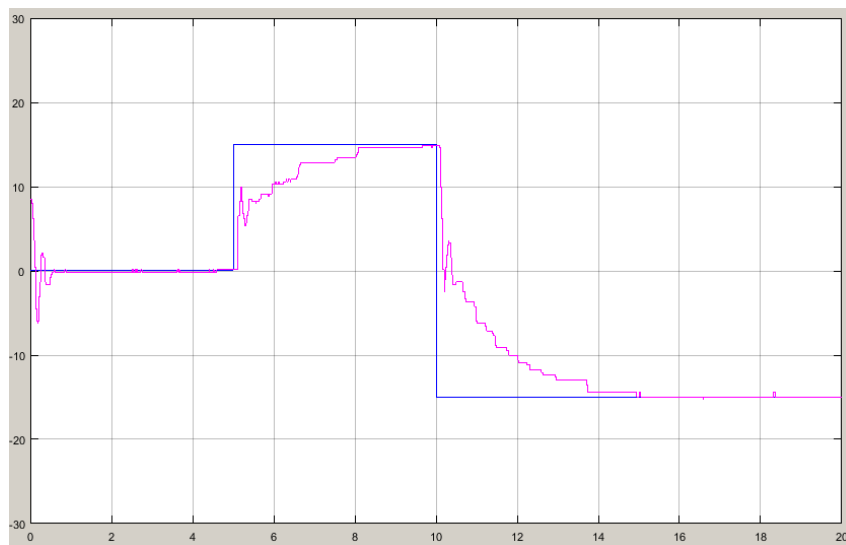


Figura 44 – Servo do eixo mão com controlador

## 6 Considerações Finais

Neste trabalho, intencionou-se apresentar as atividades realizadas durante a disciplina de Estágio Supervisionado (180 horas) - esta que é exigida para a conclusão do curso de Engenharia Elétrica da Universidade Federal de Campina Grande. O programa de estágio fora fundamental para desenvolvimento das habilidades práticas do aluno uma vez que permitiu utilizar os conhecimentos adquiridos em disciplinas anteriores em um projeto real e com objetivos.

Além de ter permitido ao aluno aplicar os conhecimentos adquiridos de Identificação de sistemas, introduzidos na disciplina de *Controle Digital*, a pesquisa a cerca dos servos e sua composição - no que diz respeito à eletrônica - pode ser fortemente relacionada com as disciplinas de *Eletrônica*, *Instrumentação Eletrônica* e *Sistemas de Aquisição de Dados e Interface*. Esta última também teve grande contribuição na elaboração da interface em C# voltada para coleta de dados.

Quanto ao projeto dos controladores, pode ser fortemente relacionado com a já citada disciplina de *Controle Digital* e também com a disciplina de *Controle Analógico*, enquanto o estudo das tabelas de controle do servo AX-12A remontam à disciplina de *Arquitetura de Sistemas Digitais*.

A utilização da comunicação OPC por sua vez, fora uma contribuição especial fornecida pelo Laboratório de Instrumentação Eletrônica e Controle, uma vez que é um tema pouco explorado durante a graduação e fora introduzido durante treinamentos realizados periodicamente, fato este que também se aplica ao estudo de controladores ótimos.

Por fim, o estudo da robótica atrelada ao manipulador exigiu um estudo da física e controle do braço robótico, apresentando um conjunto de desafios, nos quais se fez necessário dedicação. O estudo abre portas para o futuro, uma vez a robótica é uma tecnologia em constante ascensão. Destaca-se ainda a utilização da plataforma robótica como meio de ensino para técnicas de identificação e implementação de controladores, por meio dos guias e tutoriais produzidos.

# Referências

ACIOLI, G.; LIMA, A. B. de; BARROS, P. R. Identificação e modelagem de plataforma robótica didática. *Congresso Brasileiro de Automática*, 2016. ISSN 1474-6670. Citado na página 21.

IWANITZ, F.; LANGE, J. Ole for process control – fundamentals, implementation and application. *Hüthig Verlag Heidelberg*, 2001. Citado na página 5.

LIEC Laboratório de Instrumentação Eletrônica e Controle. <<http://liec.ufcg.edu.br>>. Accessed: 19-02-2017. Citado na página 3.

ROBOTIS Biblioteca SDK. <<http://support.robotis.com/en/software/dynamixel/sdk.htm>>. Accessed: 19-02-2017. Citado na página 19.

---