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

**Nilteomar Nascimento Gabay**

# DESENVOLVIMENTO DO PROJETO MECATRÔNICO E CONTROLE DE UM ROBÔ DE PÊNDULO

Manaus
2015

**Nilteomar Nascimento Gabay**

# DESENVOLVIMENTO DO PROJETO MECATRÔNICO E CONTROLE DE UM ROBÔ DE PÊNDULO

Trabalho de Conclusão de Curso submetido à Coordenação do curso de Engenharia de Controle e Automação da Universidade do Estado do Amazonas como parte dos requisítos necessários para a obtenção do grau de Engenheiro.

Orientador Dr. Israel Mazaira Morales

Manaus
2015

**Nilteomar Nascimento Gabay**

# DESENVOLVIMENTO DO PROJETO MECATRÔNICO E CONTROLE DE UM ROBÔ DE PÊNDULO

Trabalho de Conclusão de Curso submetido à Coordenação do curso de Engenharia de Controle e Automação da Universidade do Estado do Amazonas como parte dos requisítos necessários para a obtenção do grau de Engenheiro.

Aprovado em de de 2015.

BANCA EXAMINADORA

---

**Dr. Israel Mazaira Morales**
Orientador

---

**Walter Andres Valenzuela, Dr.**
Presidente da banca

---

**Almir Kimura Junior**
Convidado 1

---

**Israel Francisco Benitez Pina**
Convidado 2

Manaus
2015

# Agradecimentos

Ao Senhor Jesus Cristo, por ter me dado saúde, força e entendimento para conseguir os resultados necessários à conclusão deste trabalho. Aos meus pais, Nilteomar e Rosenilse, por suas orações e intercessões e por terem me dado a educação, os valores e o suporte necessário não somente para concluir esta etapa de meus estudos, como também as já encerradas e as vindouras. Da mesma forma ao meu irmão Nikmar, que deu (mesmo sem saber) um grande apoio para a finalização deste curso. Às minhas tias e tios da família Nascimento, bem como minha avó Margarida "Xexé"Nascimento, por terem me encorajado a seguir a carreira acadêmica em uma instituição de ensino superior pública. A Linda Moreira, por estar ao meu lado em momentos difíceis e decisivos, me compreendendo e me apoiando nestas situações. Ao meu primo Alzimar Leda e ao meu tio Ivan Andrade, pelas lições de eletrônica analógica, que muito ajudaram no tocante à parte mais técnica deste trabalho. Ao professor Doutor Israel Mazaira Morales, grande educador e exemplo em nossa Instituição, que esteve ao meu lado em momentos decisivos, compreendendo as dificuldades encontradas, sempre com uma palavra de conhecimento, de ânimo, transmitindo dedicação e atenção em suas orientações. Ao grupo Cicari e toda a equipe do Laboratório Multiusuário, na pessoa do Profº Msc. Moisés Bastos e da Profª Dra. Ana Carolina, e em especial ao meu amigo, Engº Luiz Cordovil, que muito colaborou com sua experiência acadêmica e com palavras de ânimo. Á minha turma de Controle e Automação, que muito me ajudaram não só na elaboração deste trabalho como durante todos os períodos do curso. À Universidade do Estado do Amazonas, em especial à Coordenação de Controle e Automação e a todo seu corpo Docente, pela oportunidade de adquirir mais conhecimento e qualificação profissional. Deus é bom.

*"...porque Ele é bom,*
*e a sua misericórdia dura para sempre."*
*(Salmos 136;1b)*

# Resumo

O *Pendubot* é um sistema mecatrônico que consiste em um robô planar de dois graus de liberdade subatuado, utilizado em renomados centros de robótica para fins de ensino e pesquisa. Em virtude de não haver ainda um robô deste gênero na Instituição, descreve-se neste trabalho o projeto de desenvolvimento de um robô Pendubot com intuito de servir de base para trabalhos futuros focados na construção do sistema em questão. Servomotor e servoamplificador, interface de comunicação entre o servoamplificador e o computador que executa o controle do sistema, transdutores de posição angular para mensurar os ângulos das juntas e o algoritmo que efetuará o controle do sistema em ferramenta computacional de simulação, foram testados individualmente, visando a utilização futura no robô. A modelagem matemática bem como os controladores de *swing* e balanço foram sintonizados e simulações com os parâmetros obtidos foram executadas para verificação da resposta do sistema.

**Palavras-chaves**: Pendubot. Projeto. Controle. Hardware.

# Abstract

Pendubot is a mechatronic system consisting of a robot planar underactuated two degrees of freedom used in robotics renowned centers for the purposes of teaching and research. Because not even be a robot of its kind in the institution, describes this work the development project of a robot based Pendubot serving of order for further work focused on the construction of the system in question. Servo motor and servo amplifier, communication interface between the servo amplifier and the computer running the system control angular position transducers to measure joint angles and the algorithm that will make the system control in computational simulation tool were tested individually in order the future use of the robot. Mathematical modeling as well as swing and balance controllers were tuned and simulations with the parameters obtained were performed to check the system's response.



**Key-words**:Pendubot. Design. Control. Hardware.

# Lista de ilustrações

# Lista de tabelas

# Sumário

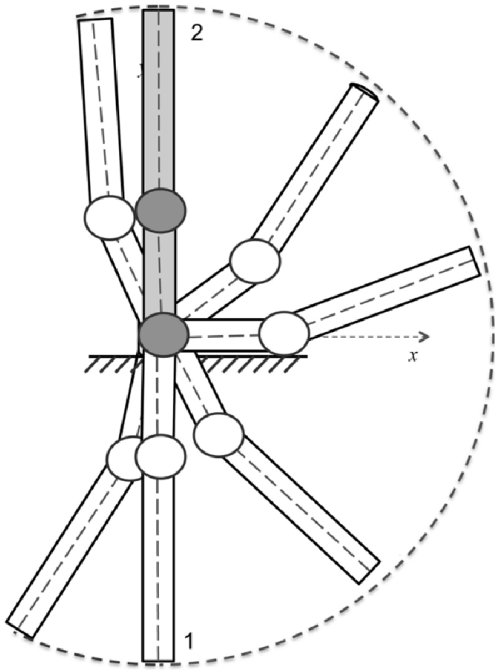
# 1 INTRODUÇÃO

O Robô de Pêndulo (*Pendubot*) abordado neste trabalho, é constituído de dois elos conectados a duas juntas, na qual uma junta possui atuação, enquanto a outra junta pode mover-se livremente, sem atuação. Por isso, diz-se que trata-se de um sistema subatuado, pois o número de graus de liberdade, a saber dois, é maior que a quantidade de atuadores no sistema, sendo um sistema não-linear (AURELIE et al., 2006).

Por ser montado verticalmente, o *Pendubot* só possui uma posição de equlíbrio estável, a qual está demarcada pelo algarismo 1 na figura 1. Esta posição de equilíbrio estável (também chamada *down-down*) é alcançada naturalmente quando o sistema não está sujeito a torques de qualquer natureza, exceto o da gravidade.

O objetivo do controle do *Pendubot* é levá-lo a uma posição de equilíbrio instável, como por exemplo, a demarcada com o algarismo 2 na figura 1. Esta posição de equilíbrio também é chamada de *up-up*.

Figura 1 – Posições de equilíbrio do *Pendubot*: *down-down* (1) e *up-up* (2).




Fonte: http://brzu.net/06cvg, adaptado pelo autor.

Trata-se de um sistema utilizado para fins didáticos, podendo ser um instrumento motivacional para o estudo de Teorias de Controle, particularmente Controle Não-Linear, utilizando técnicas de Controle já consagradas para atingir o objetivo, podendo ser também um instrumento para pesquisa e desenvolvimento de novos métodos e até junções de métodos de Controle (Controles Híbridos) (SHEPPARD, 1998).

## 1.1 Formulação do Problema

É fato que a prática aliada a teoria na metodologia pedagógica, fornece a possibilidade de solidificar os conhecimento obtidos pelos alunos, permitindo a verificação de fatos e fenômenos aprendidos teoricamente por meio de um experimento ou na análise de um sistema como um todo.

A partir desta linha de raciocínio e sendo um robô utilizado em grandes centros de estudos, como por exemplo na Universidade de Illinois, para fins de pesquisa e desenvolvimento, o *Pendubot* é uma interessante ferramenta didática. Entretanto, ainda não há um robô deste gênero na Instituição a qual o autor faz parte atualmente. Também não há nenhum estudo sobre o projeto de um *Pendubot* utilizando tecnologias disponíveis na Instituição por parte de acadêmicos da referida Instituição.

## 1.2 Justificativa

Sendo um sistema não-linear, como muitos dos sistemas do mundo real, o *Pendubot* pode ser aproximado para um sistema linear, seguindo algumas restrições. É com esse caráter não-linear que se comportam muitos sistemas no campo da Engenharia (SEMAN et al., 2010), sistemas estes que podem ser estudados e soluções de problemas podem ser propostas com base nestes estudos.

Por outro lado, o desenvolvimento do projeto de um *Pendubot*, possibilitaria uma futura implementação do mesmo na Instituição, sendo uma ferramenta a mais para efetuar exposições didáticas, podendo ser objeto de estudo de determinadas componentes curriculares, como por exemplo Controle Não-Linear e até mesmo utilizado para pesquisa e desenvolvimento com foco em diferentes metodologias de controle, como Controles Híbridos.

## 1.3 Motivação

São poucos os recursos didáticos de baixo custo disponíveis para estudos de controle não-linear e até teorias de Controle, de um modo geral. Por conta disso, a alternativa de muitos estudantes de Engenharia de Controle e Automação e Mecatrônica, é utilizar ferramentas computacionais de simulação. Contudo, é muito difícil modelar um sistema que seja idêntico ao sistema real, havendo pequenas divergências quando se comparam os resultados apresentados por simulações do modelo com resultados amostrados de um sistema real (DORF; BISHOP, 2009).

Levando em consideração esses poucos recursos, propõe-se o desenvolvimento do projeto mecatrônico e controle de um robô de pêndulo (*Pendubot*). Chama-se atenção ao fato de que o *Pendubot* não é um sistema de baixo custo, pois possui componentes de custo bastante

elevado. Todavia, os principais componentes para confecção de um robô deste gênero estavam disponíveis na Instituição, como por exemplo servomotores e servoamplificadores, sendo que alguns não haviam sido testados ainda. Outros componentes haviam sido adquiridos recentemente, como conversores tensão-frequência.

Foi então feito um levantamento e verificou-se que juntando estes componentes, e possivelmente alguns elementos auxiliares, poderia ser apresentada uma proposta de construção e controle de um *Pendubot*.

## 1.4 Objetivos

Com intuito de expor melhor os objetivos do presente Trabalho, foi feita uma divisão entre objetivo geral e objetivos específicos, explicados a seguir.

### 1.4.1 Objetivo geral

Desenvolver o projeto mecatrônico e o controle de um robô de pêndulo (*Pendubot*) de modo que este robô possa ser implementado futuramente. Este objetivo geral foi dividido em dois objetivos menores, mas não ainda específicos, os quais foram denominados "macroetapas", que são o Gerenciamento de *Hardware* e o Controle do Sistema.

Dentro da macroetapa Gerenciamento de *Hardware,* existem duas etapas menores, a etapa de *hardware* da primeira junta e *hardware* da segunda junta, que são compostas de objetivos específicos. Na macroetapa Controle do Sistema há apenas objetivos específicos.

### 1.4.2 Objetivos específicos

Na etapa *hardware* da primeira junta, tem-se os seguintes objetivos específicos:

- Efetuar ligação elétrica do servomotor utilizado para atuação da primeira junta;
- Estabelecer a comunicação do servomotor com o computador que gerencia o algoritmo executado para controle do sistema;
- Entender os modos de funcionamento do servomotor;
- Selecionar componentes auxiliares para atuação da primeira junta, se necessário;
- Definir uma interface de comunicação entre o Servomotor e o computador e;
- Desenvolver um algoritmo para controle da primeira junta, para validação do funcionamento.

Os seguintes objetivos específicos compõem a etapa *hardware* da segunda junta:

- Propor um sensor que mensure a posição angular da segunda junta;

- Análisar a proposta de sensoriamento apresentada no item anterior e;
- Propor soluções quanto a problemas relacionados a segunda junta.

Na última etapa, o Controle do Sistema, os objetivos específicos são:

- Escolher os parâmetros mecânicos dos elos, tais como material, dimensões e formato;
- Modelar o sistema matematicamente, utilizando para tal a mecânica de Lagrange;
- Efetuar a linearização da primeira junta para implementação do controle de *swing*;
- Efetuar a linerização do sistema por série de Taylor, para modelagem do controlador de realimentação de estados completo e;
- Simular o modelo obtido visando validar a proposta de controle.

## 1.5 Metodologia

Para o trabalho em questão deve ser criada uma aplicação (algoritmo) na ferramenta computacional MATLAB® que permita testar a modelagem matemática efetuada no decorrer deste Trabalho, a qual deve ser obtida utilizando os conceitos básicos de posicionamento de manipuladores robóticos, os conceitos e equações da Mecânica Lagrangeana, o método de linearização por série de Taylor, a técnica do projeto de controladores por realimentação de variável de estado completo (*full state feedback controller*, do inglês) pelo método de alocação de polos utilizando a variante Regulador Linear Quadrático (LQR) e o próprio MATLAB® para efetuar os cálculos necessários.

Para atingir o objetivo desejado, será realizada pesquisa sobre o funcionamento de manipuladores robóticos, bem como seus principais componentes e algoritmos de funcionamento. Testes individuais serão realizados no *hardware*, os quais serão descritos detalhadamente no capítulo referente a Materiais e Métodos, assim como os passos para funcionamento destes componentes.

O projeto será desenvolvido baseando-se em estudos bibliográficos, publicações *online* e de experimentos práticos sobre as tecnologias utilizadas nos testes do *hardware,* verificando e implementando, onde possível, obtendo resultados práticos, como por exemplo a construção da primeira junta do P*endubot* utilizando o servomotor da *Mitsubishi*® HC-RFS503S e seus periféricos e elementos adicionais, utilizando o *software Simulink*® para efetuar o controle e a plataforma Arduino para comunicação entre o conjunto Servo e o computador onde é executado o algoritmo de controle, a fim de constatar a possibilidade de construção e o controle da proposta em trabalhos futuros.

## 1.6 Organização do Trabalho

A estrutura desta monografia está da seguinte forma:

No capítulo 1 é apresentada uma breve Introdução com o intuito de situar o leitor no contexto do Trabalho bem como permitir uma visão geral do que será abordado.

Trabalhos relacionados com objetivos semelhantes ao abordado nesta monografia são apresentados no capítulo 2. Algumas semelhanças e diferenças entre os trabalhos são destadas de forma geral.

Os principais conhecimentos e tecnologias necessários à execução deste projeto, na visão do autor, foram listados no Referêncial Teórico, título do capítulo 3.

O capítulo 4 aborda o desenvolvimento do Trabalho em duas macrofases, uma denominada Gerenciamento de *Hardware,* na qual toda a parte técnica relacionada a *Hardware* (materiais) necessária ao desenvolvimento do projeto são abordadas, e a outra denominada Controle do Sistema, que abrange a teoria e a matemática (métodos) do sistema, onde são discutidas as questões de modelagem e controle do robô.

Os Resultados Obtidos e Discussões são apresentados no capítulo 5, no qual são abordados e comparados os resultados bem como são apresentadas algumas propostas de melhorias e trabalhos futuros. Posteriormente uma conclusão é apresentada, sintetizando o Trabalho.

Posteriormente, o apêndice A apresenta os principais algoritmos utilizados pelo autor para obter os resultados matemáticos e gráficos e o apêndice B contém algumas fotografias do teste realizado para validar o funcionamento da primeira junta do *Pendubot*.

# 2 TRABALHOS RELACIONADOS

Alguns trabalhos foram tomados como base para a realização deste Projeto, os quais são abordados em linhas gerais nas seções a seguir.

## 2.1 Projeto Mecânico e Controle do Pendubot

Sendo um dos primeiros trabalhos a serem realizados sobre o *Pendubot* (abreviatura de *Pendulum* e *Robot*), *Mechanical Design And Control of the Pendubot*, do autor Daniel J. Block (BLOCK, 1996), aborda os pontos principais para a construção deste robô, que é baseado em outro robô chamado *Acrobot.* Nesta tese, a modelagem matemática bem como o Controle deste robô são abordados.

O *hardware* utilizado por Block é explanado superficialmente e a construção tende a ficar um tanto subentendida por parte do leitor. Algumas especificações do robô, como por exemplo as massas dos elos do *Pendubot* não foram citadas. Os apêndices, nos quais estão presentes as principais informações referentes ao trabalho, como a identificação da fricção nas juntas, os arquivos de simulações e os algoritmos de cálculo, não foram publicados.

Três métodos diferentes são apresentados para a parametrização (identificação) dos coeficientes de atrito nas juntas, a saber modelo sólido em Desenho Assistido por Computador (CAD), método de equação de energia e método de otimização.

O presente trabalho teve a tese citada como a principal referência no sentido de se basear para o desenvolvimento do projeto mecatrônico e controle do *Pendubot* apresentado nesta monografia. Entretanto, por não haver um modelo físico do robô disponível na Instituição a qual pertence o autor, os atritos nas juntas não foram considerados. Os controles de balanço e *swing* foram baseados nesta tese, bem como as comparações dos resultados finais.

## 2.2 Controle de Swing do Pendubot

Neste artigo (*Swing Up Control of the Pendubot*, título original), Sheppard (1998) efetua uma abordagem semelhante a feita por Block em sua tese. O controle de balanço também foi considerado, apesar de o controle enfatizado ser o de *swing*, conforme informa o título do artigo. O controle de *swing* consiste em levar o robô de sua condição de equilíbrio estável (inercialmente parado sem qualquer torque de natureza interna) até a posição de equilíbrio instável no ponto superior, $180^{o}$ em relação à posição de equilíbrio estável. Já o controle de balanço está relacionado à rejeição de distúrbios no sistema.

Outra difereça é que Sheppard utiliza-se da estimação de estados (observador linear) e Filtros de Kalman, estimando os estados do controlador. Pelo que ficou subentendido,

os parâmetros de especificação do *Pendubot* de Sheppard foram baseados em um robô feito por uma de suas referências, por isso ele não realizou a identificação do sistema, mas apenas citou estes resultados.

Esta monografia se baseou no *hardware* apresentado por este autor em seu artigo, isto é, nos requisitos mínimos do *hardware* necessário à modelagem de uma proposta de robô deste gênero.

## 2.3 Controle de laboratório de um modelo de Pendubot.

Utilizando uma abordagem um pouco diferente da usada pelos autores citados anteriormente, Pavol Seman (SEMAN et al., 2010), principal autor deste artigo, em *Control of Laboratory Model of Pendubot* (título original), utilizou apenas o controle de balanço do *Pendubot*, o qual está relacionado à rejeição de distúrbios. Para isso, modelou o sistema com base na mecânica Lagrangeana e obteve um modelo linear no espaço de estados. Depois, utilizou-se do MATLAB® para efetuar os cálculos referentes ao controlador linear quadrático (LQR). E por fim realizou experimentos com o modelo obtido diretamente no robô.

Outra diferença é que Seman conseguiu uma forma de mensurar a fricção entre as juntas do *Pendubot*, forma essa que não foi explicada em seu trabalho, mas apenas informado os valores numéricos de tal grandeza.

Um ponto em que o presente Trabalho se baseia no trabalho de Seman, é no tocante a utilização de um servomotor e um servoamplificador de uso industrial, a saber um conjunto da *Mitsubishi® Electric Corporation*, sendo o servoamplificador de modelo MR-J2S-40A, por coincidência da mesma família do servoamplificador abordado neste trabalho (MR-J2S-500A). Entretanto, para aquisição das medidas de ângulos das juntas, medidas essas efetuados pelos transdutores de posição angular (*encoders*) presentes em cada junta, foi utilizada uma placa de aquisição de dados do fabricante Humosotf® compatível com o ambiente de simulação utilizado pelo autor, a saber o *Simulink®*.

## 2.4 Relatório Final sobre o Pendubot

Este relatório foi desenvolvido por uma equipe de cinco pessoas, provavelmente em caráter de projeto final de determinado Curso. Aurelie et al. (2006), em *Final report of Pendubot* (título original), apresenta uma modelagem matemática sólida e bem desenvolvida na qual o autor desta monografia se baseou para a modelagem utilizada no presente trabalho. Uma tática interessante utilizada no relatório em questão, foi a verificação gráfica de comportamento do sistema real de ambas as juntas caindo (*falling* foi o termo empregado originalmente) livremente bem como o sistema teórico (simulado) na mesma condição.

As respostas foram bem parecidas, sendo considerado o modelo matemático uma boa aproximação do sistema real.

Os cálculos referentes à linearização do sistema foram mostrados em parte, sendo apresentados ao final os resultados obtidos. Uma técnica chamada rede segura (*safety net*, originalmente) foi criada para ações de emergência, caso o processo se tornasse perigoso de certa forma ou o usuário desejasse a parada do sistema. Apesar de se tratar de um relatório final, não foram encontrados resultados referentes a resultados experimentais no sistema real ou mesmo simulações.

Como já relatado anteriormente, este relatório foi utilizado como base pelo autor desta monografia principalmente por causa da modelagem matemática envolvida e também pelo fato de que estes autores passaram por problemas com relação a taxa de recepção das medidas dos *encoders* (*sample time*), apesar de terem utilizado uma placa de aquisição de dados, componente também utilizado nos trabalhos de Block, Sheppard e Seman.

# 3 REFERENCIAL TEÓRICO

Para o desenvolvimento deste Trabalho, necessitou-se de uma pesquisa bibliográfica abrangente sobre teorias relacionadas à execução do desenvolvimento do projeto, como Mecânica Lagrangeana, método de linearização de Taylor, técnicas de alocação de pólos para implementação de controladores no espaço de estados, conceitos básicos sobre manipuladores robóticos e conhecimentos relacionados à ferramenta computacional MATLAB® para simular a aplicação.

Se fez necessário também adquirir conhecimentos sobre as tecnologias e dispositivos prováveis à aplicação das teorias estudadas, tais como servomotor e servoamplificador, transdutor de posicionamento angular (comumente chamado *encoder*), microcontrolador, circuito integrado inversor de sinal, conversor tensão/frequência e conhecimentos relacionados à ferramenta computacional *Simulink*® para efetuar testes e validação de partes do Projeto.

## 3.1 Mecânica de Lagrange

Em algumas aplicações da mecânica, as leis de Newton se tornam muito complicadas de serem aplicadas a sistemas dinâmicos. Um método geral e mais sofisticado para descrever equações de movimento de sistemas dinâmicos é a mecânica de Lagrange (KELLY; DAVILA; LORíA, 2005). Sem considerar o formalismo matemático por trás dos resultados, os quais podem ser consultados pelo leitor em Soldovieri (2013), temos que a função de Lagrange ou o chamado Lagrangeano de um sistema, é dada pela diferença entre a energia cinética $K$ e a energia potencial $U$ do sistema com respeito a uma variável e sua primeira derivada em relação ao tempo. Matematicamente, tem-se:

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{U}(q) \tag{3.1}$$

A aplicação do Lagrangeano em manipuladores robóticos, como é o caso deste Trabalho, é matematicamente representada pela equação de Euler-Lagrange (CRAIG, 1989), que para sistemas não conservativos é dada por:

$$\frac{d}{dt}\left[\frac{\mathfrak{d}\mathcal{L}(q_i, \dot{q}_i)}{\mathfrak{d}\dot{q}_i}\right] - \frac{\mathfrak{d}\mathcal{L}(q_i, \dot{q}_i)}{\mathfrak{d}q_i} = \tau_i \tag{3.2}$$

Onde $\tau$ é está relacionado às forças externas e torques com relação à i-ésima variável.

## 3.2 Método de linearização por série de Taylor

A maioria dos sistemas dinâmicos, sejam mecânicos, elétricos, químicos, biológicos ou de qualquer outra área, possuem comportamento não-linear, o que é notório à medida em

que os valores das variáveis crescem indefinidamente (DORF; BISHOP, 2009). Entrentanto, desde que hajam pequenas variações em torno de um ponto de funcionamento, um sistema pode ser linearizado para uma dada faixa de valores.

Uma das técnicas utilizadas para obter uma aproximação linear ou efetuar uma linearização é a expansão em série de Taylor (GUIDORIZZI, 2008), a qual é dada matematicamente, para polinômios até a terceira ordem por:

$$f(x) = f(x_0) + f^{'}(x_0)(x - x_0) + \frac{f^{"}(x_0)}{2}(x - x_0)^2 + \frac{f^{"'}(\bar{x})}{3!}(x - x_0)^3 \tag{3.3}$$

onde $x_0$ é o ponto em torno de onde pretende-se efetuar a linearização.

## 3.3 Técnica de alocação de polos

Uma técnica utilizada para projetar controladores no espaço de estados é a alocação de pólos, que tem como resultado a matriz de ganho $K$ da realimentação de estados para a alocação de pólos (DORF; BISHOP, 2009). De modo geral, para um sistema definido por $\dot{x} = Ax + Bu$, com o sinal de controle dado por $u = -Kx$, que possui todas as variáveis de estado mensuráveis e disponíveis para retroação, pode ser projetado um controlador por realimentação de variável de estado completo (*full state feedback controller*, do inglês), desde que os estados sejam completamente controláveis (OGATA, 2003). A matriz de controlabilidade é dada por:

$$M = [B \vdots AB \vdots \cdots \vdots A^{n-1}B] \tag{3.4}$$

Onde $A$ e $B$ são as matrizes no espaço de estados do sistema.$W$ é a matriz formada pelos $n$ elementos da matriz dada por $|sI - A| = s^n + a_1 s^{n-1} + \cdots + a_{n-1}s + a_n$, sendo definida como:

$$W = \begin{bmatrix} a_{n-1} & a_{n-2} & \cdots & a_1 & 1 \\ a_{n-2} & a_{n-3} & \cdots & 1 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ a_1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{3.5}$$

A matriz de transformação e estados na forma canônica $T$, é dada por:

$$T = MW \tag{3.6}$$

As características do sistema podem ser definidas escrevendo-se os polos desejados $\mu_n$, onde pode-se definir até o n-ésimo pólo por meio da equação:

$$(s - \mu_1)(s - \mu_2)\cdots(s - \mu_n) = s^n + \alpha_1 s^{n-1} + \cdots \alpha_{n-1}s + \alpha_n \tag{3.7}$$

onde deseja-se identificar os coeficientes $\alpha$ até o n-ésimo termo. Então a matriz de ganho de retroação $K$ pode ser definida como:

$$K = [\alpha_n - a_n \vdots \alpha_{n-1} - a_{n-1} \vdots \ldots \vdots \alpha_2 - a_2 \vdots \alpha_1 - a_1] T^{-1} \tag{3.8}$$

Uma outra abordagem para a alocação de polos é o Regulador Linear Quadrático (LQR), cuja definição depende da função custo, dada por:

$$J = \frac{1}{2} \int (x^T Q x + u^T R u) dt \tag{3.9}$$

onde Q e R são matrizes diagonais de peso que devem ser definidas pelo projetista com a restrição de possuir apenas valores maiores que zero (DAS et al., 2012). Após definir as matrizes Q e R, é necessário resolver a Equação Algébrica de Riccati (RAHMAN; ALI, 2013) dada por $A^T P + PA + Q = PBR^{-1}B^T P$, para encontrar o valor da matriz simétrica $P$ (nxn) e calcular a matriz $K$, dada por:

$$K = R^{-1} B^T P \tag{3.10}$$

## 3.4 Manipuladores robóticos

Uma das características mais relevantes dos manipuladores robóticos são os graus de liberdade que este possui. Os graus de liberdade determinam de modo geral, a quantidade de movimentos independentes que um braço robótico pode efetuar (GROOVER, 2011). Manipuladores planares de 2 graus de liberdade possuem duas juntas de rotação e dois elos, podendo ter ou não um extremo, comumente chamado garra.

Para descrever um manipulador planar com duas rotações, em que os elos tenham comprimento $a_1$ e $a_2$, pode-se descrever as coordenadas cartesianas do extremo, sabendo-se o ângulo que há entre o eixo $x$ ($\theta_1$) e o elo e o ângulo que há entre a linha do elo $a_1$ e o elo $a_2$ ($\theta_2$) (KELLY; DAVILA; LORíA, 2005). A Figura 2 mostra os aspectos geométricos para posicionamento do extremo de um manipulador.

As equações que descrevem tais posições são:

$$\triangle x = \triangle\theta_1 \left| a_1 sen\theta_1 + a_2 sen(\theta_1 + \theta_2) \right| + \triangle\theta_2 \left| a_2 sen(\theta_1 + \theta_2) \right| \tag{3.11}$$

e

$$\triangle y = \triangle\theta_1 \left| a_1 cos\theta_1 + a_2 cos(\theta_1 + \theta_2) \right| + \triangle\theta_2 \left| a_2 cos(\theta_1 + \theta_2) \right| \tag{3.12}$$

É possível obter-se os ângulos $\theta_1$ e $\theta_2$ sabendo-se a posição em que se encontra o extremo. Este método é chamado cinemática inversa (KELLY; DAVILA; LORíA, 2005) e é matematicamente dado pelas equações a seguir, contudo não gera solução única:

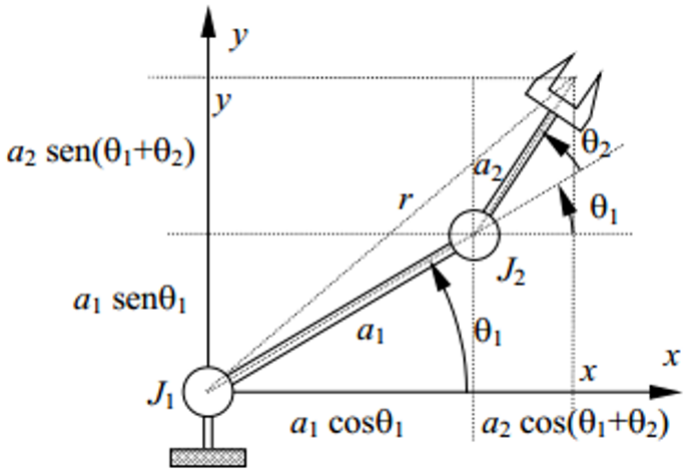
$$\theta_2 = atan2\left(\frac{\sqrt{1-\frac{(x_{p+}^2 y_p^2 - a_1^2 - a_2^2)^2}{(2a_1a_2)^2}}}{\frac{x_{p+}^2 y_p^2 - a_1^2 - a_2^2}{2a_1a_2}}\right) \tag{3.13}$$

e

$$\theta_1 = atan2(\frac{y_p}{x_p}) - atan2(\frac{a_2 sen\theta_2}{a_1 + a_2 cos\theta_2}) \tag{3.14}$$

Onde $x_p$ e $y_p$ são os pontos desejados.

Figura 2 – Aspectos geométricos para posicionamento do extremo do manipulador.

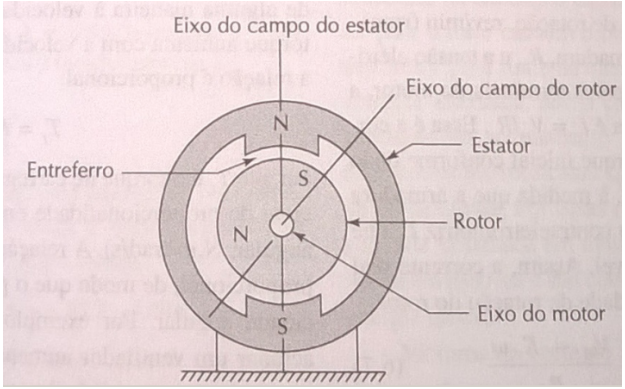



Fonte: (KELLY; DAVILA; LORíA, 2005).

## 3.5 Servomotor

Motores elétricos rotativos são uma grande classe de atuadores que convertem um sinal, geralmente de natureza elétrica, em energia mecânica, especificamente, a rotação do eixo ou rotor do motor elétrico. Uma subclasse de motores elétricos são os motores de corrente contínua ou motores CC.

Figura 3 – Partes básicas de um motor elétrico.



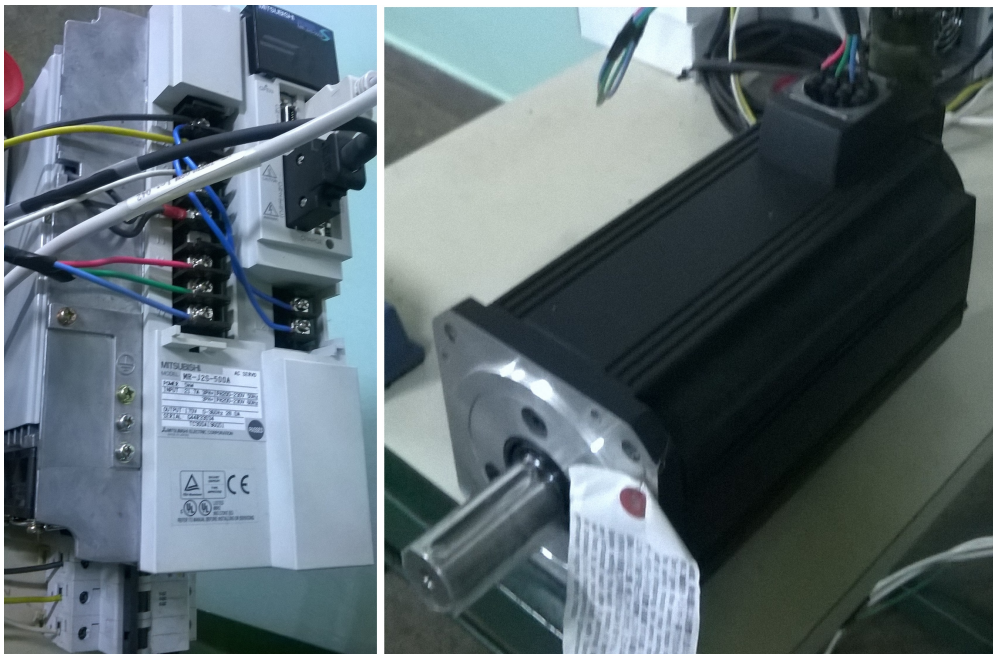

Fonte: (GROOVER, 2011).

Motores CC são alimentados por corrente e tensão elétrica constantes. Estas grandezas elétricas criam um campo magnético por meio de um componente chamado comutador, que altera continuamente a polaridade relativa entre o rotor e estator, produzindo assim um torque que gira o eixo do motor initerruptamente (GROOVER, 2011). A Figura 3 mostra as partes descritas de um motor CC.

Uma classe especial de motores CC são os servomotores. O termo servomotor indica que existe uma malha de realimentação que é usada para alcançar uma velocidade pré-definida, assim um servomotor deve necessariamente, ter um sistema de potência e outro de controle para poder desempenhar suas funções (GOUVEIA; ANDRADE; BALTAZAR, 2013).

Para controlar a posição, a velocidade ou mesmo o torque de um servomotor, normalmente existe um controlador chamado de servoamplificador que realiza o controle de diversos parâmetros do sistema. Para efetuar o sensoriamento de posições angulares, existe um transdutor de posição angular, que pode ser um codificador óptico, comumente chamado *encoder,* ou um transformador de giro, comumente chamado *resolver.*

Figura 4 – Servomotor HC-RFS503S e Servoamplificador MR-J2S500A.



Fonte: Autor.

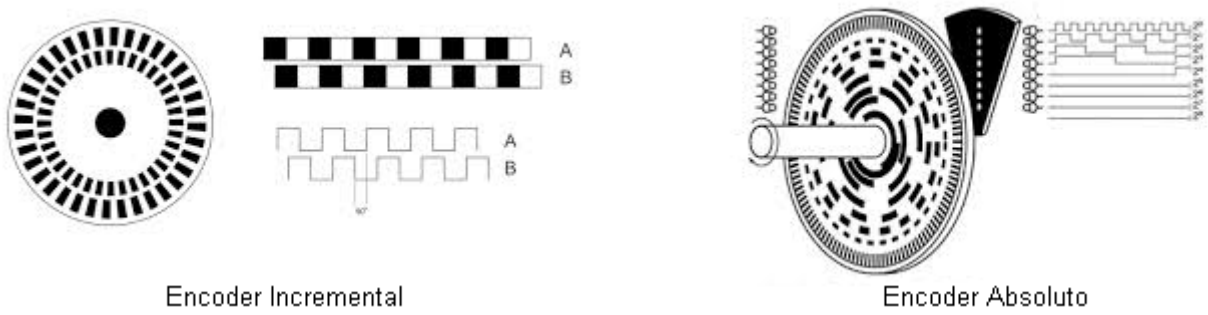
### 3.5.1 Servoamplificador

O controle das funções de um servomotor é feito por um equipamento chamado servoamplificador, também chamado de *servodriver*, que normalmente vem recomendado pelo fabricante de cada servomotor, pois necessita ser compatível com o servomotor que estará controlando. Um conjunto com servoamplificador e seu servomotor recomendado pelo fabricante, são ilustrados na Figura 4.

### 3.5.2 Transdutor de posição angular (Encoder)

O transdutor de posição angular, também conhecido como *encoder* consiste em um conjunto emissor-receptor ótico acoplado ao servomotor. Este conjunto é normalmente composto por um diodo emissor de luz (*Led*) que emite um sinal luminoso e por um fotodiodo ou fototransistor, que recebe esse sinal luminoso decodificado. Esta decodificação é feita por um disco que está acoplado ao eixo girante do servomotor, podendo este disco possuir apenas uma ou duas trilhas de informações paralelas e levemente defasadas, caracterizando um *encoder* incremental ou várias trilhas defasadas seguindo um código binário, caracterizando um *encoder* do tipo absoluto. A Figura a seguir exibe os tipos de *encoders* citados anteriormente.

Figura 5 – Esquema simplificado de *encoders* do tipo incremental e absoluto.



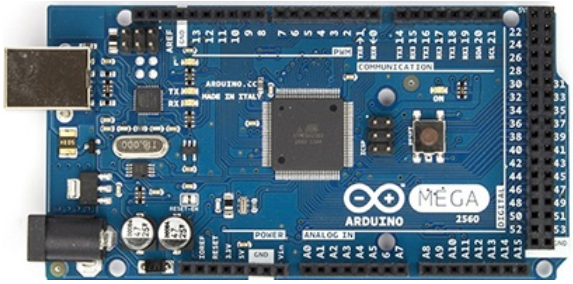

Fonte: Autor.

## 3.6 Microcontroladores

Muito utilizado atualmente (KURNIAWAN, 2013a), o microcontrolador é uma opção de utilização em campo por se tratar de uma solução que pode resolver tarefas específicas, ao ser programada com linguagem de programação distinta. Possui uma memória para guardar a aplicação feita pelo usuário, a chamada memória de programa ou ROM (*Read Only Memory* - Memória Somente de Leitura) e uma memória para guardar as informações temporárias, a memória de dados ou RAM (*Randon Acess Memory* - Memória de Acesso Aleatório) (NICOLOSI, 2002). Quanto à linguagem de programação, pode variar de acordo com o fabricante.

Quanto ao invólucro, alguns microcontroladores necessitam de uma placa com os componentes necessários ao seu funcionamento, tais como fonte de alimentação, gerador de pulsos (*clock*), locais para encaixar as entradas e saídas físicas do componente, dentre outros, já que o microcontrolador em si é apenas um chip encapsulado. Contudo, em outros casos, o microcontrolador já vem com o *hardware* preparado fisicamente para uso, como é o caso da plataforma Arduino, equipada com o microcontrolador Atmel® AVR (TIMMIS, 2011). A Figura 6 ilustra uma placa exemplo de plataforma, chamada Arduino Mega 2560.

Figura 6 – Plataforma Arduino Mega 2560 com Microcontrolador Atmega 1280.

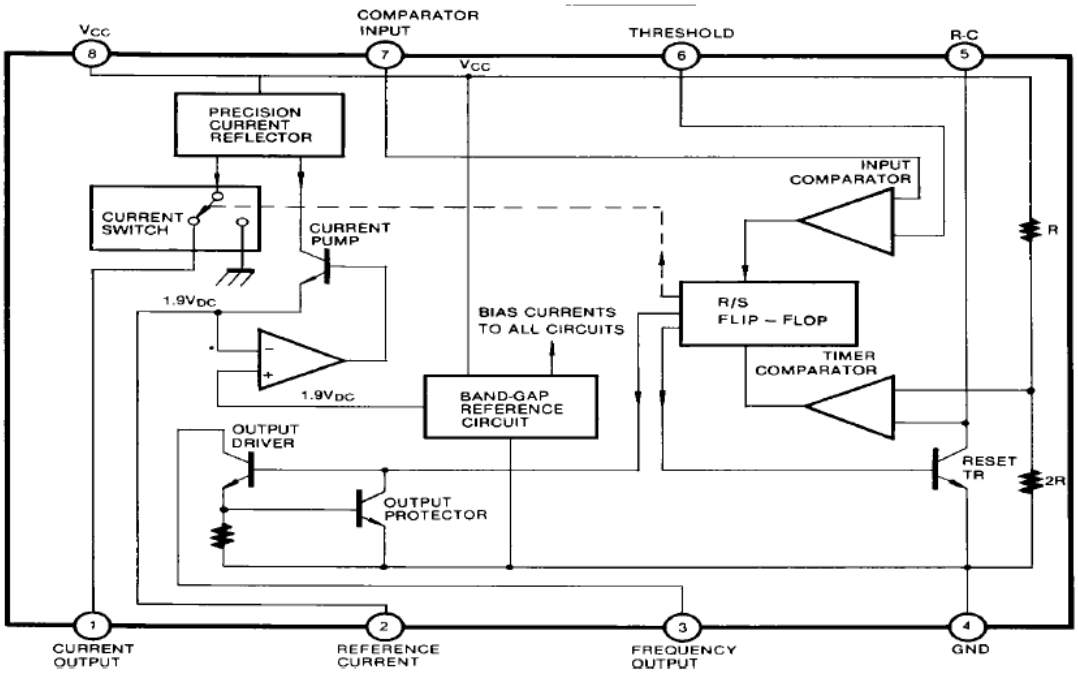



Fonte: Autor.

## 3.7 Conversores tensão-frequência.

Em algumas aplicações, é necessário ter um gerador de pulsos que pode ser controlado por uma variação de tensão ou corrente (BRAGA, 2015b). Alguns circuitos integrados podem realizar esta função com o auxílio de alguns componentes, como é o caso do LM331, que pode gerar frequências de até 100Khz com uma amplitude fixa (FAIRCHILD, 2001). O princípio de funcionamento deste equipamento é que, dado um valor de tensão na entrada do circuito, haja uma conversão para um valor de frequência na saída e para variações lineares da tensão de entrada, haja uma variação linear também na saída. São definidos também os limites de tensão e frequência no manual do fabricante (*datasheet*) do componente, a qual o leitor pode consultar em Fairchild (2001). A Figura a seguir mostra o diagrama de blocos simplificado de um circuito integrado LM331.

Figura 7 – Diagrama de blocos simplificado de um circuito integrado LM331.

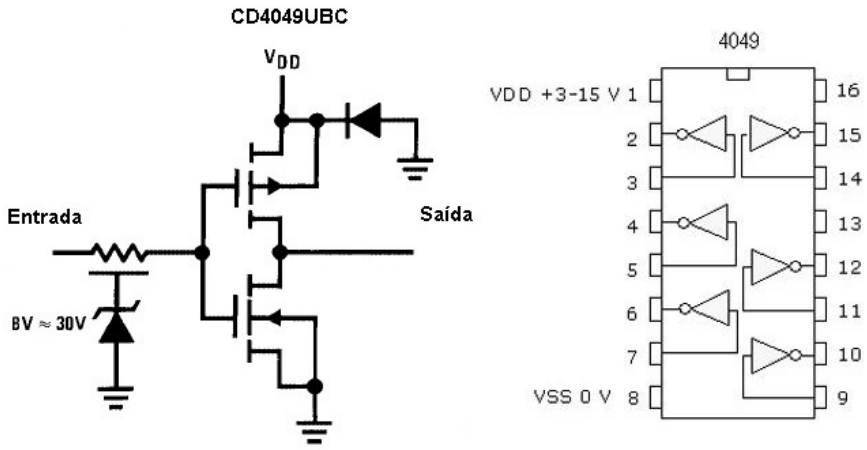



Fonte: (FAIRCHILD, 2001).

## 3.8 Circuito integrado inversor de sinal

Há aplicações em que é necessário inverter determinado sinal digital. Nestas ocasiões é interessante utilizar um amplificador operacional como inversor de sinal (BOYLESTAD; NASHELSKY, 2004) ou mesmo um circuito inversor de sinal que efetue esta função (BRAGA, 2015a), como por exemplo um CI (circuito integrado) 7404 da família TTL ou um CI 4049 da família CMOS. As características e modo de funcionamento de cada CI podem ser encontradas em seus respectivos manuais (*datasheets*). Um esquema com a pinagem do CI 4049 é mostrado na Figura a seguir.

Figura 8 – Esquema de um circuito inversor CMOS 4049.




Fonte:(FAIRCHILD, 2002).

## 3.9 Ferramenta computacional de simulação MATLAB/Simulink®

O *software* MATLAB® (*Matrix Laboratory* - Laboratório de Matrizes) é uma ferramenta computacional que permite efetuar cálculos matemáticos, modelagens e simulações das aplicações do usuário. Permite ao usuário escrever em linguagem de alto nível, possuindo diversas bibliotecas para muitas aplicações específicas, como aplicações de eletrônica analógica, eletrônica digital, cálculo diferencial e sistemas de controle (MATHWORKS, 2015). Possui também compatibilidade com vários tipos de *hardware*, desde microcontroladores até placas de aquisição de dados.

Ao usuário está disponível ainda, desde que este instale também esta ferramenta, um pacote de simulação chamado *Simulink*®, que permite ao usuário escrever sua aplicação em formato de diagrama de blocos, o que possibilita ao programador uma visão mais geral do sistema.

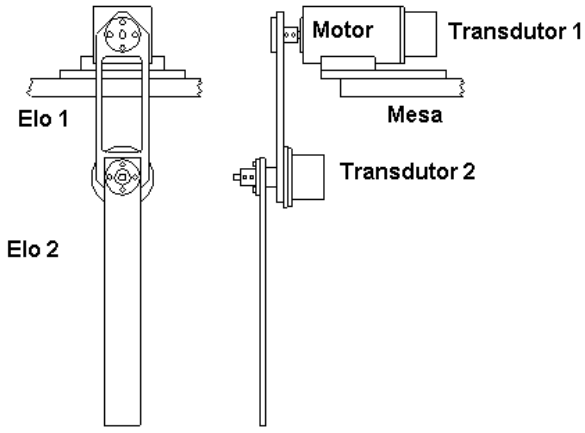
# 4 MATERIAIS E MÉTODOS

Neste capítulo estão descritas as etapas realizadas para alcançar os objetivos específicos, pontualmente e de forma detalhada, até a obtenção do objetivo geral, utilizando a metodologia descrita anteriomente para alcançar tais resultados.

## 4.1 Proposta do desenvolvimento do projeto mecatrônico e controle de um Pendubot

A proposta deste projeto é desenvolver o projeto mecatrônico, isto é a parte mecânica, com a definição de uma proposta de estrutura mecânica dos elos e juntas, e modelagem matemática, tomando por base os parâmetros adotados, como também a parte eletrônica, dimensionando e selecionando os componentes eletro-eletrônicos necessários ao funcionamento do sistema que irá compor o *Pendubot*. A Figura 9 retrata os itens básicos do *Pendubot* criado por Daniel J. Block.

Figura 9 – Itens básicos do *Pendubot*.




Fonte: <http://brzu.net/06cvi.>

Como o *Pendubot* precisa de um atuador em sua primeira junta, propõe-se a atuação utilizando um servomotor industrial, que possa ser controlado por meio de um *software* no qual seja desenvolvido o algoritmo de controle. O controle deve ser projetado com base na modelagem matemática efetuada, a qual por conseguinte está baseada nos estudos e conhecimentos referentes aos itens 3.1, 3.2, 3.3 e 3.4. A comunicação entre o sistema servo (servomotor e servoamplificador) e o computador que executa o algoritmo de controle também deve ser definida, levando em conta os parâmetros de controle. Quanto à segunda junta, um transdutor de posição angular (*encoder*) deve ser proposto para mensurar as medidas dos ângulos desta junta. A comunicação entre esta segunda junta e o computador que executa o algoritmo de controle precisa de uma solução eficaz, para que não haja

perda de dados no tráfego das informações, o que impossibilitaria a eficácia do controle, levando a instabilidade do sistema.

Para melhor aproveitamento e foco nas tarefas, o trabalho foi dividido em duas macroetapas, a saber o Gerenciamento de *Hardware* e o Controle do Sistema.

A macroetapa de Gerenciamento de *Hardware* compreende as etapas de funcionamento da primeira e da segunda junta do *Pendubot*, que são as duas etapas principais, as quais envolvem muitos itens e atividades. As principais atividade dentro da etapa primeira junta são:

- Ligação elétrica do Servomotor;
- Comunicação do servomotor com o computador;
- Modos de funcionamento do servomotor;
- Componentes auxiliares para atuação da primeira junta;
- Interface de comunicação entre o servomotor e o computador e;
- Desenvolvimento do algoritmo de controle.

Quanto à segunda junta, temos que as principais atividades dentro desta etapa são:

- Proposta de transdutor de posição angular (*encoder*);
- Análise da proposta de sensoriamento e;
- Proposta de solução à problemas relacionados à segunda junta.

Com relação à macroetapa Controle do Sistema, esta compreende as seguintes etapas:

- Escolha dos parâmetros mecânicos dos elos, tais como massa, comprimento e formato dos elos;
- Modelagem matemática do sistema;
- Linerização por série de Taylor;
- Modelagem do controlador e;
- Simulação dos resultados obtidos.

Nas seções seguintes, será descrita a metodologia aplicada para resolução de cada etapa, bem como as tarefas inerentes a cada atividade, detacando os resultados e citando algumas propostas que não tiveram resultado satisfatório, visando melhor desempenho em trabalhos futuros relacionados.

## 4.2 Hardware da primeira junta do Pendubot

Serão abordadas nas subsecções a seguir, as etapas de Configuração, ajustes e funcionamento do *hardware* necessário ao funcionamento da primeira junta do *Pendubot*, tais como ligação e testes do sistema servo, escolha dos componentes para funcionamento da parte eletrônica do projeto, escolha da interface de comunicação do servo com o computador, entre outros.

### 4.2.1 Ligação elétrica do servoamplificador

O primeiro passo para o teste da primeira junta, foi a ligação elétrica do servoamplificador e do servomotor, haja visto que o conjunto servo que estava disponível para utilização, fornecido pela Instituição de Ensino, ainda não havia sido utilizado. Trata-se de um servomotor da *Mitsubishi*® modelo HC-KFS73, cujo servoamplificador é o de modelo MR-J2S-70B, que podem ser visualizados na Figura 10.

Figura 10 – Primeiro servomotor testado no projeto - HC-KFS73


Fonte: Autor.

Foram feitas todas as conexões elétricas, Conforme indicado no manual do fabricante, as quais o leitor pode Conferir em Mitsubishi (2004), a saber, a ligação da alimentação trifásica do servoamplificador a um disjuntor trifásico ligado à rede elétrica de 220V, a conexão do *encoder* do servomotor ao servoamplificador e a ligação trifásica de potência do servomotor ao servoamplificador. Como tudo estava Conforme o manual do fabricante, o sistema foi energizado.

Com o servoamplificador ligado, foi exibida uma mensagem no *display* do mesmo, indicando um problema. Ao Conferir o significado da mensagem no manual do fabricante, indicava um problema relacionado ao *encoder* do equipamento. Tentou-se então efetuar a troca do cabo para sanar o problema, contudo esta ação não foi suficiente, indicando problema no transdutor acoplado ao servomotor. Por este motivo foi necessário adquirir outro sistema servo.

Havia a disposição um outro conjunto servo, do mesmo fabricante, da mesma série, mas de família diferente, a saber, o servomotor de modelo HC-RFS503K e o servoamplificador cujo modelo era o MR-J2S-500A. A Tabela 1 descreve as principais características deste sistema. Para maiores informações, consultar a referência Mitsubishi (2007).

Tabela 1 – Principais especificações do servoamplificador MR-J2S-500A.

| | |
|---|---|
| Alimentação - Tensão/Frequência | 3 fases de 200 a 230VAC, 50/60Hz |
| Freio Dinâmico | Acoplado-Interno |
| Max. Freq. pulsos entrada (kpps) | 500 (diferencial), 200 (pulsos coletor aberto) |
| Controle de velocidade (rpm) | Com. analógico: 2000; Com. interno: 5000. |
| Potência de máx. de saída | 5kW |
| Modos de controle | Posição (P), Velocidade (S) e Torque (T) |
| Modos de posição híbridos | P/S, P/T, S/T |
| Software de Configuração | MR Configurator - MRZ-JW3 SETUP151E |
| Saída do Servoamplificador | 170V 0-360Hz, 28.0A |
| Qtd. Parâmetros de Controle | 84 |
| Resolução do *Encoder* | 131.072 pulsos/revolução |

Fonte: (MITSUBISHI, 2007).

Este conjunto servo também ainda não havia sido utilizado, contudo o servoamplificador já estava conectado eletricamente e era possível ligá-lo. Entretanto uma dificuldade foi encotrada: o conector de potência trifásica do servomotor não estava disponível e por isso não era possível ligá-lo. Como não foi encontrado este conector, a solução foi desmontar mecanicamente o local de encaixe do conector, retirar a solda com muito cuidado da fiação de potência do servomotor, mantendo a organização e depois ressoldar as pontas dos fios para não haver perdas na transmissão de potência. As pontas ressoldadas foram fixadas em 4 polos de um conector tipo *nybloc* de um lado, e do outro lado foi conectada a fiação que transmite os comandos de potência vindos do servoamplificador. A Figura 11 exibe, à esquerda, uma vista superior do conector antigo, destacado em amarelo, e à direita após a operação descrita, a solução com o conector *nybloc*.

Depois de implementada a solução do conector, o servoamplificador foi ligado e exibiu a seguinte mensagem: *AL. 16.* Esta mensagem aparece porque, por segurança, o servomotor possui uma chave de emergência normalmente aberta (NA) e se não há um comando externo desabilitando esta opção, o alarme de emergência, *AL. 16,* é ativado. Para desativar este alarme, algumas conexões precisam ser feitas na porta CN1B, uma das portas de controle do servoamplificador.

O servoamplificador em questão possui 4 portas para entrada/saída de dados: *CN1A, CN1B,CN2 e CN3.* Cada porta possui as seguintes funcionalidades:

- CN1A: comunicação e recepção de comandos externos referentes ao modo posição, sensoriamento (pistas do *Encoder*) e alimentação;

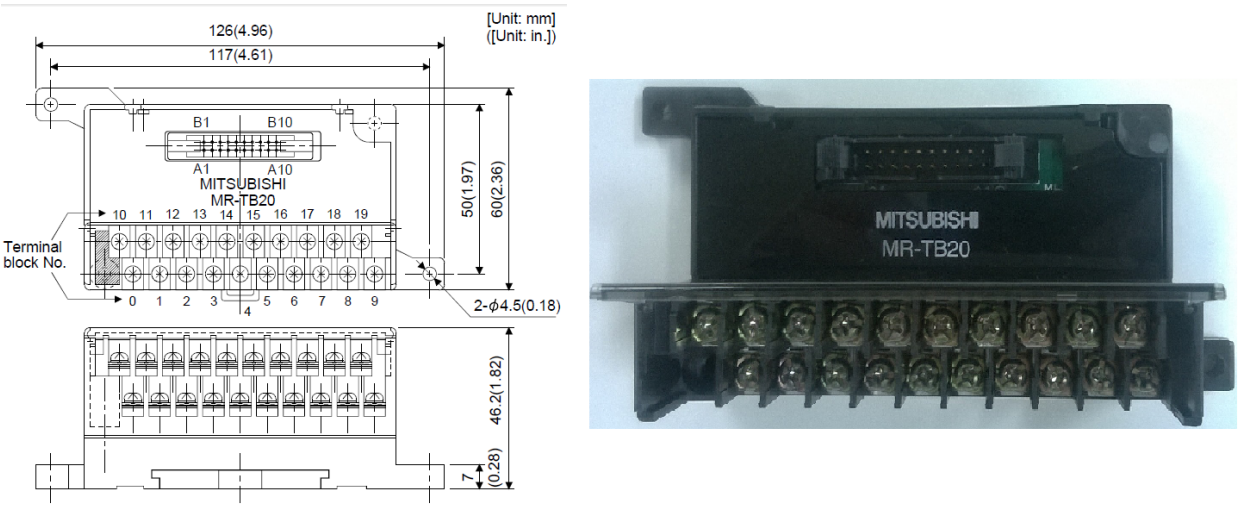
- CN1B: comunicação e recepção de comandos externos referentes aos modos velocidade e torque, alimentação, segurança e habilitação;
- CN2: comunicação com o *encoder* e;
- CN3: comunicação com computador ou RS-232C/RS-422 com outros servoamplificadores, caso esteja em rede.

Figura 11 – Solução de conexão de potência do sistema servo.

Fonte: Autor.

Para facilitar as conexões, existe um adaptador que pode ser ligado ao cabo que é conectado nas entradas CN1A e CN1B chamado MR-TB20, cujo desenho técnico e uma fotografia estão ilustradas na Figura 12.

Figura 12 – Adaptador para as portas CN1A e CN1B - MR-TB20.




Fonte: Autor.

Este conector pode ser ligado às portas CN1A e CN1B, cada um com sua pinagem, Conforme se pode verificar em Mitsubishi (2007). Dois destes pinos (EMG - emergência e SG - Comum das entradas digitais) precisam estar conectados, bem como os pinos VDD (fonte de alimentação interna) e COM (entrada da fonte de alimentação digital). Feito este procedimento, a mensagem de alarme sumiu e o servomotor estava pronto para o primeiro teste: Comunicação do servomotor com o computador.

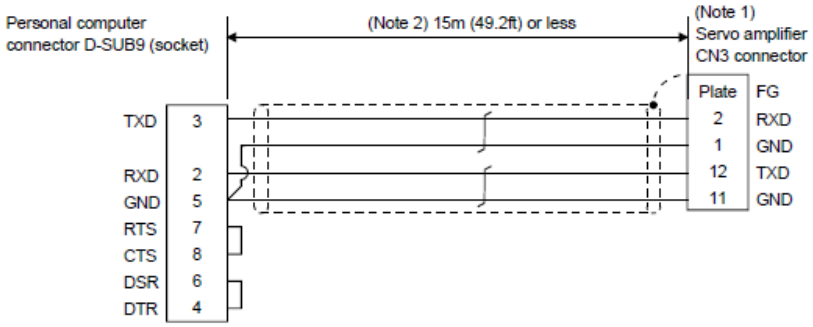
### 4.2.2 Comunicação do servoamplificador com o computador.

O manual do fabricante recomenda um *software* chamado *MR-Configurator* no qual se pode realizar comunicação com os sistemas servo, testes dos modos de funcionamento, Configuração dos parâmetros do servoamplificador, criação de algoritmos por meio da linguagem de programação disponível, entre outros. Entretanto, este *software* é proprietário e de protocolo fechado, não permitindo acesso à sua estrutura interna, por exemplo para saber o que se passa na execução de um algoritmo criado no próprio *software*, mas para efeito de teste do conjunto, o *software* foi utilizado.

Após a instalação do *software* em um computador, tentou-se estabelecer comunicação com o servoamplificador, entretanto foi encontrado um problema: o servoamplificar não respondia à comunicação. Como a porta serial do computador ainda não havia sido utilizada, havia suspeita de que ela pudesse estar com problemas.

Foi realizado então um teste simples na porta serial: um conector D-Sub9 foi encaixado na porta serial, estando curto-circuitados os pinos 2 e 3 - *TX* e *RX*, respectivamente, e utilizando o programa *Hyper Terminal* do *Windows*® XP, foi constatado que a porta serial não tinha problema. A próxima hipótese era de que o cabo RS-232C para CN3 (a porta de comunicação) estava com defeito e não funcionava. O manual do *software MR-Configurator* (MITSUBISHI, 2014) possui o esquemático do cabo em questão e baseado neste esquema, que pode ser observado na Figura 13, foi feito um cabo de comunicação em caráter de teste, utilizando um terminal D-Sub9.

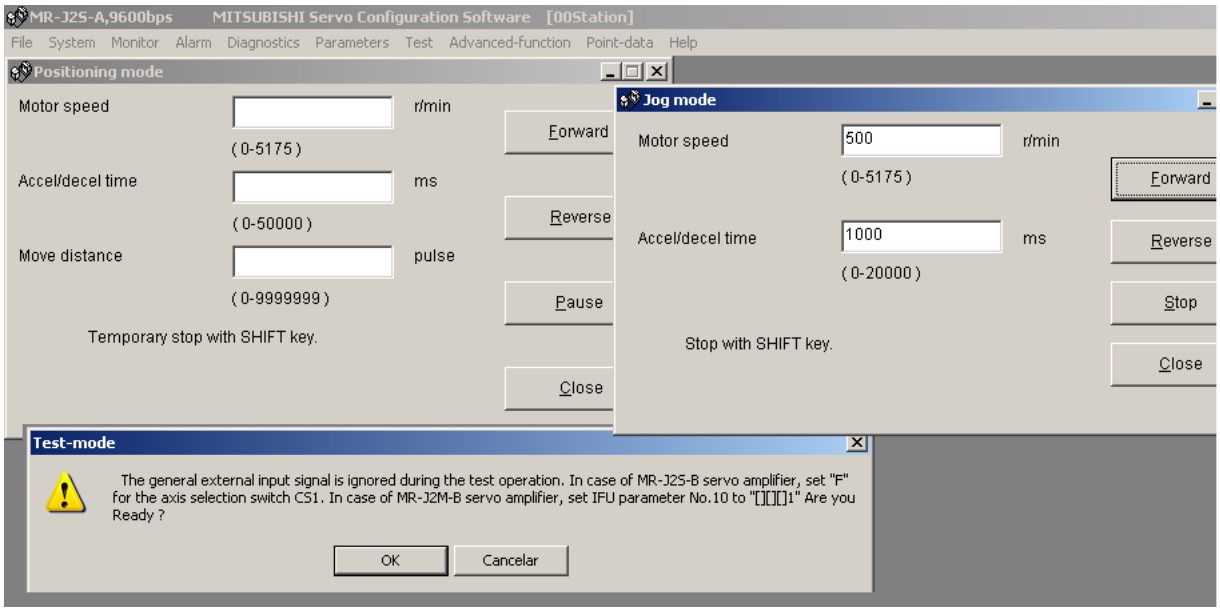
Figura 13 – Esquemático do cabo de comunicação servo-computador (RS-232C).




Fonte: (MITSUBISHI, 2014).

Tentou-se novamente estabelecer conexão com o servoamplificador e desta houve resposta: o cabo antigo estava realmente defeituoso. Com a conexão estabelecida, pode-se efetuar alguns testes como por exemplo os testes com o modo velocidade e com o modo posicão. A Figura 14 mostra duas janelas do programa *MR-Configurator,* a saber a janela do modo posição (*positioning mode*, à esquerda) e a janela do modo velocidade (*jog mode*, à direita).

Figura 14 – Captura de tela do software *MR-Configurator* - Modo teste de posicionamento e velocidade.




Fonte: Autor.

Como é possível notar na Figura 14, ao testar o conjunto servo o usuário pode definir a velocidade do eixo do servo em rpm (rotações por minuto), o tempo de aceleração em milissegundos e também um valor de deslocamento angular em pulsos. O valor de deslocamento angular é dado em pulsos por conta da forma como funciona o modo posição, baseado em pulsos. Como a resolução do *encoder* é de 131.072 pulsos por revolução, se este valor for colocado no *dropdown move distance* (mover a distância de), o eixo do servomotor dará uma volta e assim proporcionalmente.

Tabela 2 – Comandos e descrições dos comandos da linguagem de programação do *MR-Configurator*.

| Comando | Descrição |
|---|---|
| SPN | Configura velocidade do servo. |
| STC | Configura aceleração do servo. |
| MOV | Configura quantidade de pulsos para posicionamento. |
| SYNC | Habilita algumas entradas digitais do servo. |
| TIM | Tem a função de um *delay* (retardo). |
| TIMES | Tem a função de uma estrutura de repetição. |
| STOP | Para a execução do programa. |

Fonte: (MITSUBISHI, 2014).

Quanto à linguagem de programação do *MR-Configurator,* esta é bastante limitada possuindo ao todo sete comandos. Se porventura fosse tentado efetuar a programação do *Pendubot* utilizando esta linguagem, não seria possível fazê-lo pois, primeiro que a linguagem não possui um comando para eventos externos (nesta versão) e segundo que não há como implementar um controlador dentro desta linguagem, indispensável ao êxito

do projeto. Uma tabela com os nomes dos comandos e uma breve descrição destes é apresentada na Tabela 2. Mais detalhes podem ser consultados em Mitsubishi (2014).

Como os testes realizados indicaram o perfeito funcionamento do conjunto servo mas não atenderam aos requisitos necessários para utilização, outro modo de comando foi buscado, mas primeiro foi necessário entender como trabalhavam os modos de funcionamento do servoamplificador.

### 4.2.3 Modos de funcionamento do servoamplificador.

A utilização do *MR-Configurator* é considerada um modo de teste do servoamplificador. Para utilizar os 3 modos do servoamplificador, é necessário conhecer um pouco do *hardware* e dos parâmetros de Configuração do servoamplificador.

O servoamplificador possui 3 modos de funcionamento:

- Modo posição: neste modo, os pinos PP e NP da porta CN1A trabalham recebendo pulsos, como explicado na secção anterior, de um equipamento externo, por exemplo os equipamentos FX-10GM ou o AD75Px (A1SD75Px), que são módulos auxiliares da *Mitsubishi*® e específicos para esta função. Para utilizar este modo, o parâmetro 0, que está relacionado aos modos do servo, precisar ser Configurado como xxx0. A chave referente a SON (servo ON) precisa estar fechada e então o equipamento pode enviar pulsos ao servomotor. É o mais complicado de se trabalhar, em algumas condições.

- Modo velocidade: para habilitar este modo, o parâmetro 0 precisar ser Configurado como xxx2. Para utilizá-lo, basta aplicar um valor de tensão contínua entre -10 e +10V no pino VC da porta CN1B, e o servo manterá uma velocidade constante proporcional à esta tensão;

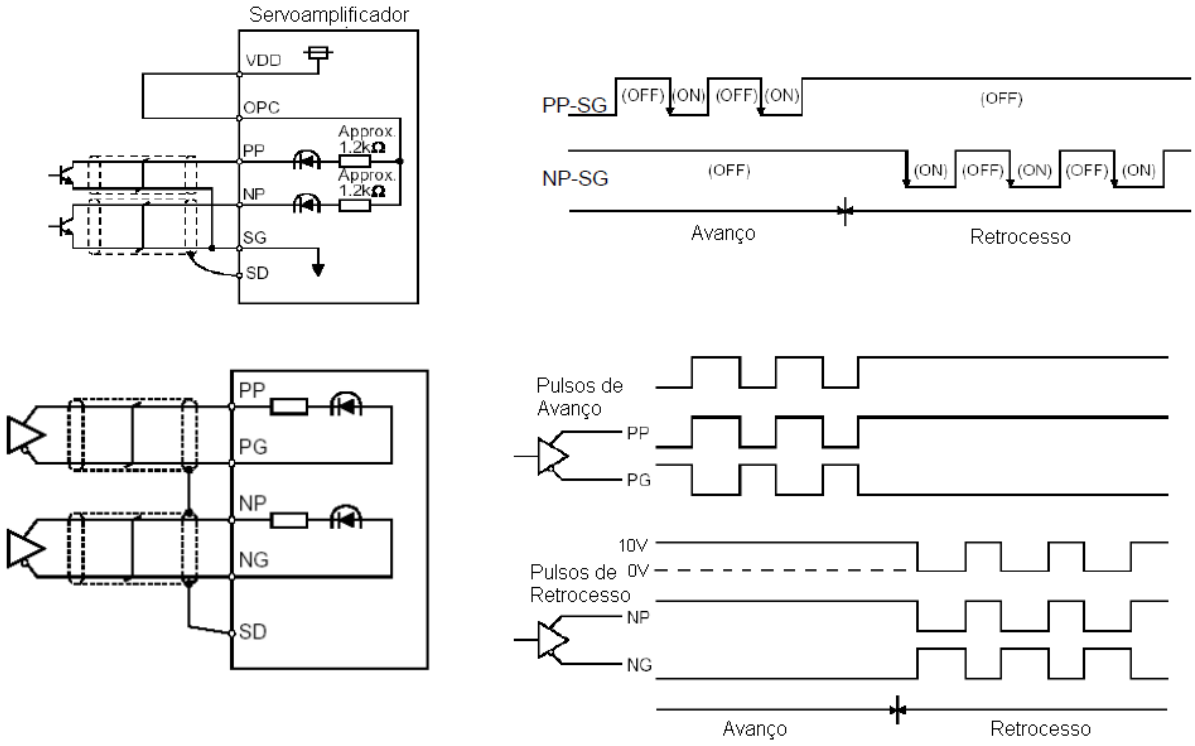
- Modo torque: para trabalhar com este modo, o parâmetro 0 precisar ser Configurado como xxx4. Para utilizá-lo, basta aplicar um valor de tensão contínua entre -10 e +10V no pino TLA da porta CN1B, e o servo manterá um torque constante proporcional à esta tensão.

É possível utilizar modos híbridos como posição/velocidade, velocidade/torque, etc, mas este não é o escopo do trabalho e o leitor pode consultar Mitsubishi (2007) para mais informações. Neste Trabalho, será utilizado apenas o modo posição.

Para utilizar o modo posição pela primeira vez, teve-se a ideia de mandar pulsos utilizando um gerador de funções, Configurado com pulsos de onda quadrada e frequência variável. O gerador utilizado foi o de modelo MFG-4221 20MHz DDS do fabricante *Minipa*® e para visualizar as formas de onda, foi utilizado um osciloscópio digital, do mesmo fabricante, modelo *Multi Variable bandwidth* MVBDSO. E efetuou-se o procedimento já descrito para habilitar o modo posição.

Há um detalhe importante quanto ao modo posição, mostrado na Figura 15, que é o modo de envio do trem de pulsos, a saber, em forma coletor aberto (na parte superior da Figura 15) e em modo diferencial (na parte inferior da Figura 15).

Figura 15 – Formas de envio de trem de pulso no modo posição.




Fonte: (MITSUBISHI, 2007), adaptado pelo autor.

Optou-se por utilizar o modo diferencial por este receber frequências de até 500KHz, enquanto que no modo coletor aberto, a frequência máxima é de 200KHz.

Na Figura 15, toma-se atenção ao fato de que a entrada PG deve receber sinal simétrico ao que recebe a entrada PP (o mesmo vale para NP e NG). Isto foi feito por meio do gerador de funções, defasando os sinais dos canais do mesmo em 180º. Depois, aplicando uma frequência de 1KHz, o servomotor atuou. O mesmo teste foi realizado com NP e NG e o servo atuou girando em sentido oposto.

A frequência do gerador de funções foi variada até seu valor máximo (500KHz), frequência em que o eixo do servomotor girava com muita velocidade. Um tacômetro não estava disponível para aferir o valor precisamente, mas pode-se dizer que a velocidade máxima estava entre 2000 e 2500 rpm, valor este comparando com resultados obtidos com o *MR-Configurator*.

O gerador de funções seria uma opção interessante de controle de frequência para atuar a primeira junta do *Pendubot* se não fosse o fato de que não há como controlar o valor de frequência do mesmo senão mecanicamente, ou seja, girando o *knob* do equipamento ou clicando nos botões relativos à mudança de frequência. Por isso, era necessário outro modo de atuar o sistema servo.

### 4.2.4 Componentes auxiliares para atuação da primeira junta.

Devido ao fato de o servoamplificador trabalhar com pulsos, precisou-se de um modo de atuar o *Pendubot*, no qual uma grandeza elétrica como tensão ou corrente (que são as formas como o computador que executaria a aplicação pode se comunicar), pudesse atuar em um componente que poderia após receber estes sinais, variar a frequência do trem de pulsos de saída. Este componente já foi citado neste trabalho e trata-se do abordado na secção 3.7: o conversor tensão/frequência.

A melhor opção seria utilizar o componente LM331, da *Fairchild Semiconductor*®, pois este gera frequências de até 100KHz (FAIRCHILD, 2001). Contudo, este necessita de um circuito eletrônico relativamente trabalhoso, o que exigiria bastante tempo para ser costruído. Outra opção seria utilizar um dispositivo semelhante e comercial, já pronto para utilização, isto é, com o circuito montado, pronto para uso. O componente que se encaixou nestas especificações foi o conversor tensão/frequência da Unidigital® modelo UD-C450. Este componente gera frequências de até 50KHz para uma entrada de +10V com um erro associado muito pequeno.

Com o auxílio do osciloscópio digital citado na secção 4.2.3, foi levantada a curva, supostamente linear segundo o fabricante em UNIDIGITAL (2013). Então, utilizando uma fonte de tensão analógica e variando-se a tensão de entrada do conversor com passos de 1V no intervalo de 0 a +10V, as frequências obtidas na saída do conversor foram organizadas na Tabela 3.

Tabela 3 – Verificação das tensões *versus* frequência do conversor UD-C450 e resposta do *encoder* para 1080 pulsos/revolução.

| Tensão (V) | Frequência (KHz) | Freq. *Encoder* (Hz) |
|---|---|---|
| 1,00 | 4,95 | 11 |
| 2,00 | 10,1 | 21 |
| 3,00 | 15,2 | 32 |
| 4,00 | 19,9 | 42 |
| 5,00 | 25,3 | 52 |
| 6,00 | 30,5 | 63 |
| 7,00 | 35,6 | 73 |
| 8,00 | 40,8 | 83 |
| 9,00 | 45,6 | 94 |
| 10,00 | 50,05 | 103 |

Fonte: Autor.

Da Tabela 3, pode-se analisar que a relação tensão-frequência pode ser considerada linear pois o erro associado é muito pequeno (menor que 0,5%), sendo aceitável para a aplicação. A coluna Freq. *Encoder* (Hz) será discutida mais adiante.

Um elemento importante ainda resta para completar a atuação da primeira junta. Para relembrar, basta rever a Figura 15: o inversor de sinal. Alguns circuitos integrados podem

ser utilizados para esta função, como é o caso de amplificadores operacionais e inversores de sinal.

Para utilizar o amplificador operacional, precisaríamos de uma fonte simétrica, que normalmente é obtida juntando duas fontes normais, conectando-se o positivo de uma fonte ao negativo da outra fonte (BOYLESTAD; NASHELSKY, 2004). Entretanto, esta seria uma forma de subutilização das fontes, em relação à utilização de um circuito inversor de sinal, como será abordado adiante.

Um circuito inversor de sinal, como um TTL (*Transistor-Transistor Logic*, Lógica Transistor-Transistor) 7404, pode ser usado para esta aplicação. Um circuito integrado (CI) inversor, possui a estrutura interna como ilustrado na Figura 8, onde é possível notar que, independente do CI utilizado, há uma entrada e uma saída e entre elas há um amplificador em modo inversor. Para o CI 7404, a amplitude máxima de tensão na saída é de 5V bem como o valor máximo de entrada, segundo a folha de especificações do fabricante, suficiente para interpretação do servoamplificador segundo Mitsubishi (2007). Então foram efetuados testes para analisar a eficiência deste CI.

Foram feitas as ligações elétricas pertinentes e depois realizado o teste utilizando a fonte de alimentação de tensão contínua ligada a entrada do conversor V/F e a saída deste dividida em duas: uma indo para o CI 7404 e sendo invertida e posteriormente indo para o pino PP e a outra indo para o pino PG, sem sofrer inversão. Quando foi enviado 5V ao conversor, nada aconteceu. Então com o auxílio de um multímetro, foram medidas as tensões no pino PP com e sem o fio portador do sinal invertido. Percebeu-se que quando a chave SON era ligada, o servoamplificador gerava um *offset* que tendia a comprimir a tensão do trem de pulsos invertido, reduzindo-o a cerca de 2,7V, insuficiente para interpretação dos pulsos por parte do servoamplificador. Como não era possível elevar o nível de amplitude de tensão na saída do CI, foi necessário escolher outro modelo, com nível de amplitude de tensão de saída maior.

A família CMOS (que significa Semicondutor Complementar de Metal-Óxido) consegue produzir tensões de até +15V em sua saída (FAIRCHILD, 2002) e recebe na entrada tensões de até +18V, necessitando apenas de uma alimentação positiva (VDD) e de um potencial neutro (GND ou VSS), como é possível verificar na Figura 8. O servoamplificador possui um pino de alimentação chamado *P15R*, que fornece uma tensão de +15V, eliminando a necessidade de fonte de alimentação externa e também 3 tipos de terra (ou massa) a saber, SG (comum das entradas digitais), LG (comum das entradas de controle) e SD (Massa). Portanto pode-se alimentar este CI com as tensões do próprio servoamplicador, sem necessidade de fonte, como é necessário no amplificador operacional.

Foram feitas todas as ligações procedendo da mesma forma que para o CI 7404. Depois, foi ligada a fonte de tensão que alimenta o conversor V/F em 5V. O resultado foi o eixo girando, recebendo pulsos de 25KHz: estava pronta a atuação da primeira junta.

### 4.2.5 Interface de comunicação entre o servoamplificador e o computador.

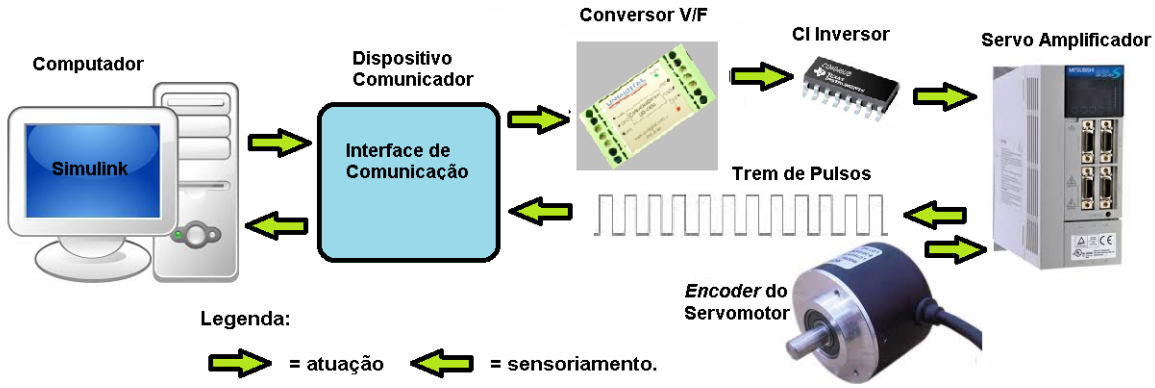
Com a atuação da primeira junta pronta, o próximo passo foi verificar uma forma de comunicar o servoamplificador ao computador onde o algoritmo de controle seria executado, e como o *software MR-Configurator* não permite comunicação com outro *software* para analisar os estados das variáveis do servoamplificador (protocolo fechado), era necessário um dispositivo que permitisse receber os dados de sensoriamento, que são os pulsos do *encoder*, enviando este sinal ao computador, o qual processaria estas informações e enviaria a tensão de atuação ao conversor V/F por meio deste dispositivo. A Figura 16 ilustra o processo descrito. Este dispositivo pode ser um microcontrolador ou precisamente uma plataforma, que neste trabalho foram utilizadas duas em caráter de teste, o *Raspberry® Pi B* e posteriormente o *Arduino Mega 2560.* A seguir é discutida a experiência obtida com cada um.

#### 4.2.5.1 Experiências obtidas com o Raspberry® Pi modelo B

O *Raspberry® Pi B* é um microcomputador de bolso, do tamanho de um cartão de crédito (UPTON; HALFACREE, 2012). Ele possui um sistema operacional que é uma versão de Linux, que pode ser o *Raspbian* (utilizada neste Trabalho), a qual o usuário precisa instalar, como em todo computador. O procedimento para instalar este sistema operacional pode ser consultado em Upton e Halfacree (2012) e Kurniawan (2013b). Depois, conectando-se alguns periféricos, como *mouse*, teclado e monitor, pode-se trabalhar normalmente como em um computador *desktop* ou *notebook.* A Figura 17 mostra uma visão geral de alguns itens do *Raspberry® Pi* modelo B.

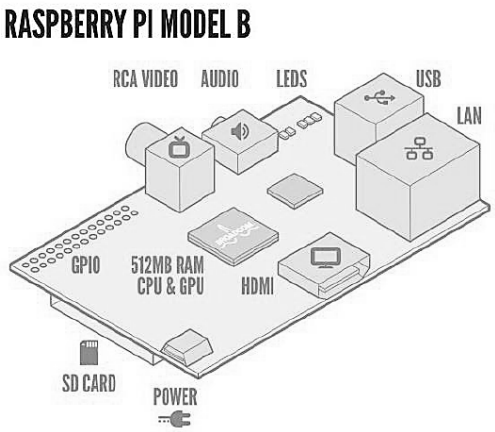
O *Raspberry® Pi B* possui uma característica muito útil à esta aplicação que é o fato de possuir uma biblioteca específica no *Simulink®*, que facilita a programação, possuindo blocos de leitura das GPIOs (*General Purpose Inputs and outputs* - entradas e saídas de propósito geral), escrita nas GPIOs, entre outros. Para mais informações, consultar a referência Kurniawan (2013b).

Figura 16 – Diagrama ilustrativo do processo atuação-sensoriamento com componentes.



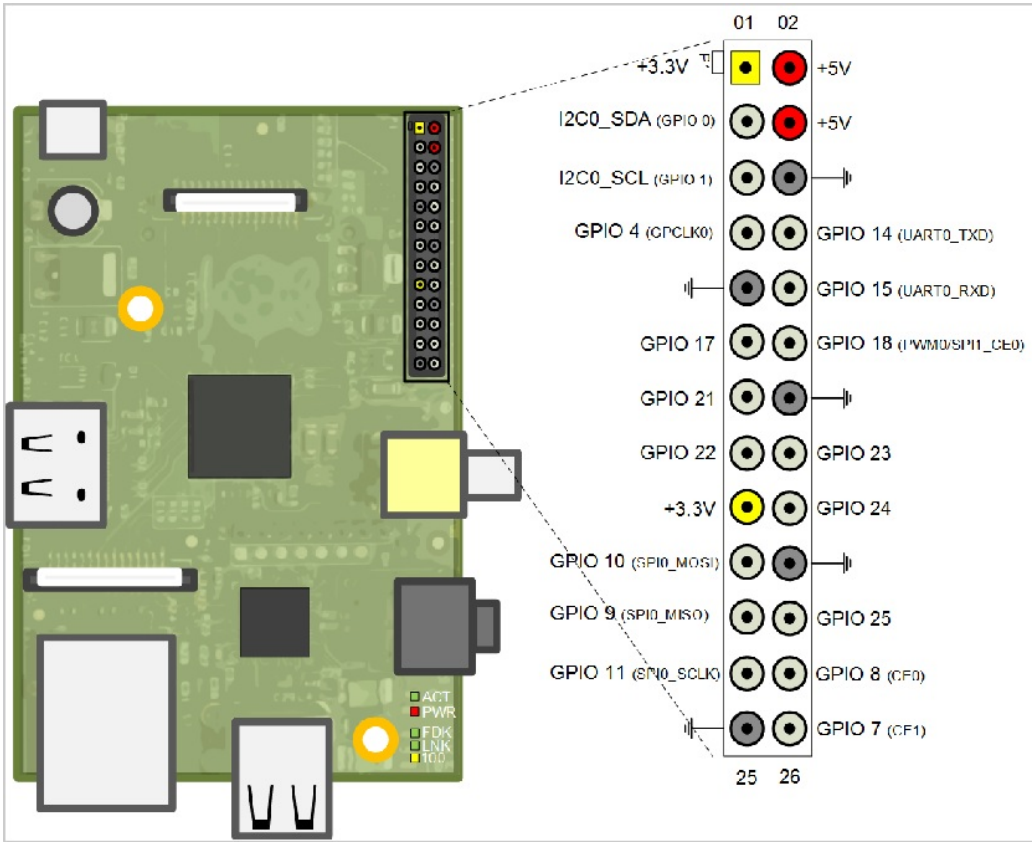

Fonte: Autor.

Figura 17 – *Raspberry® Pi* modelo B: visão geral.




Fonte: (UPTON; HALFACREE, 2012).

Figura 18 – Pinagem do *Raspberry® Pi* Modelo B: GPIOs e outros pinos.




Fonte: Autor, obtida no Help do Simulink®.

As entradas e saídas do *Raspberry® Pi B* são as GPIOs, as quais o usuário endereça na própria aplicação, estas podendo fornecer até 3,3V de tensão contínua e correntes de até algumas dezenas de mA. Os mesmos valores podem ser recebidos por meio destas portas. A Figura 18 exibe as GPIOs do *Raspberry®* Pi B.

Como não se sabia qual o valor de corrente fluia dos pinos LA e LB (respectivamente as pistas A e B do *Encoder*, localizados no terminais 6 e 7 da porta CN1A) de acordo com

a velocidade e resolução do *encoder* acoplado ao eixo do servomotor, foram coletados os valores de corrente com o auxílio de um multímetro digital, utilizando como atuação o *software MR-Configurator*. Estes valores podem ser consultados na Tabela 4, e como é possível comparar, mesmo a velocidades de cerca de 4000rpm, a corrente é bem menor que a máxima especificada para as entradas de GPIO, o que não deve acarretar nenhum dano ao *Raspberry®*.

Tabela 4 – Verificação das correntes *versus* velocidade do eixo do servomotor com o *encoder* Configurado para 1080 pulsos/revolução.

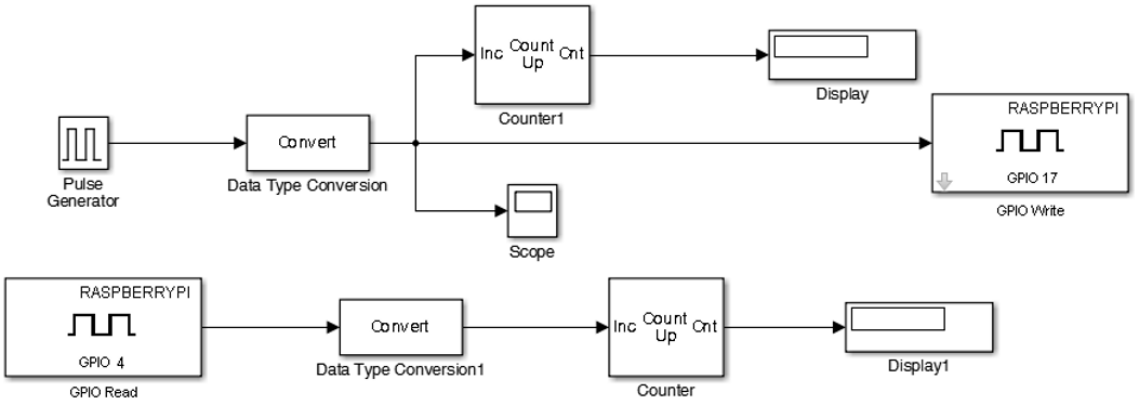
| Corrente ($\mu$A) | Velocidade (Rpm) |
|---|---|
| 46,6 | 4000 |
| 24,3 | 2000 |
| 12,4 | 1000 |
| 6,2 | 500 |
| 2,5 | 200 |
| 0,3 | 100 |
| 0,0 | 50 |

Fonte: Autor.

O *Raspberry® Pi B* é um dispositivo que trabalha com algumas linguagens de programação como *Python*, C e no caso do *Simulink®*, pode-se programar com a linguagem nativa deste software, o diagrama de blocos. Então, no *Simulink®*, foi testado o algoritmo ilustrado na Figura 19, que tem por objetivo criar um valor de trem de pulsos com o bloco *pulse generator*, converter este dado para um valor lógico (o *Raspberry® Pi B* não possui entradas analógicas, apenas digitais) e este sinal, incrementa um contador que possui um visor (*display*) conectado na sua saída. Este sinal é enviado através de um bloco de escrita GPIO, o qual é curto-circuitado externamente a outra porta GPIO que também converte este dado para um valor lógico, incrementa um contador, que também possui um visor (*display*) conectado na sua saída. Ao final da execução do algoritmo, os dois visores devem ter valores iguais ou bem próximos.

Para trens de pulsos gerados no próprio *Simulink®*, como por exemplo com o bloco *pulse generator* (gerador de pulsos), como foi o caso, o *Raspberry® Pi B* responde bem até cerca de 1500Hz. Entretanto, para captar frequências externas, muitos dados são perdidos. A suspeita é que, devido a linguagem de blocos do *Simulink®* precisar ser debugada para linguagem nativa do *Raspberry®*, há necessidade de um consumo maior de processamento do próprio *Raspberry®*. Se para estas circunstâncias o sistema tem esse comportamento, que dirá quando for executado em tempo real, que é o 'pior caso' e que é necessário quando se utiliza a versão 2014a do MATLAB®. Outra suspeita era quanto à Configuração do computador onde o programa estava embarcado, que possuia apenas 4Gb de memória RAM e um processador de 1,66GHz.

Figura 19 – Algoritmo de teste do controle da primeira junta para o *Raspberry*® Pi B.




Fonte: Autor.

Alguns testes foram realizados neste 'pior caso'. Os resultados mostraram que está fora de cogitação a utilização nesta Configuração, pois de 1080 pulsos (uma rotação) emitidos pelo *encoder* do servomotor, apenas média de 42 foram captados (os testes foram realizados com uma variação do algoritmo exibido na Figura 19, diferindo apenas no fato de que o gerador de pulsos não era mais o bloco *pulse generator* e sim o próprio *encoder* do servomotor, girando a velocidades baixas de 50rpm, mensuradas pelo *software MR-Configurator*). Por conta destes resultados, foi testada outra plataforma, o Arduino Mega 2560.

#### 4.2.5.2 Experiências obtidas com o Arduino Mega 2560.

A plataforma Arduino possui um *software* chamado *Arduino*, que neste trabalho foi utilizada a versão 1.6.3. Neste programa, foi feito um algoritmo para verificar até quantos KHz de frequência a plataforma (*hardware*) Arduino conseguia gerar. Este algoritmo é ilustrado na Figura 20.

Os resultados obtidos foram bons, cerca de 20KHz, e posteriormente, foram retirados os comandos que inserem atrasos no sistema (*delay(10);*) e obteve-se resultados de cerca de 35KHz, satisfatório quanto à aplicação realizada. Por isso, optou-se por realizar uma aplicação no *Simulink*® para esta plataforma também.

Neste ponto, pode-se explicar um assunto que ficou em aberto na tabela 3. Por padrão, a resolução do *encoder* do servomotor é Configurada como 4000 pulsos/revolução por meio do parâmetro 27 (ENR - *Encoder Output Pulses* ou Pulsos de Saída do Encoder), contudo apenas $\frac{1}{4}$ deste valor é utilizado por questão de construção do fabricante. O leitor pode consultar Mitsubishi (2007) para mais detalhes. Contudo, quando a resolução do *encoder* é alterada, tem-se que ter em mente um compromisso precisão *versus* volume de dados, pois uma vez que se está trabalhando com *encoder* do tipo incremental (para mais detalhes consultar a subsecção 3.5.2), quanto mais pulsos por revolução se recebe, maior é a precisão do sistema, entretanto maior também é quantidade de pulsos para serem

contados e se o *hardware* (ou mesmo o *software*) utilizado para efetuar estas contagens possuir limitação na recepção de frequências, é certo que muitos pulsos serão perdidos, podendo o sistema não responder da forma como foi projetado ou mesmo tornar-se instável, pois não se pode controlar o que não se pode mensurar (DORF; BISHOP, 2009). Assim, para este trabalho Configurou-se a resolução do *encoder* do servomotor para um valor de 3 pulsos por grau, o que corresponde a 1080 pulsos por revolução (4320 pulsos por revolução no servoamplificador), de forma intuitiva, por se acreditar ser um valor equilibrado no compromisso precisão *versus* volume de dados pela deficiência do Arduino Mega 2560 em receber altas frequências. Com relação aos resultados registrados na tabela 3, verifica-se que a frequência máxima obtida foi de 103Hz, muito inferior ao valor máximo de 20KHz obtido por meio do algoritmo ilustrado na Figura 20, entretanto vale ressaltar que o diagrama de blocos a ser embarcado no Arduino, terá que ser debugado para outra linguagem, necessitando de um tempo de execução maior que no algoritmo ilustrado na Figura 20, qualquer que seja o diagrama de blocos, o que deve reduzir em muito a capacidade de recepção de frequências, uma vez que o problema de envio de frequências já foi resolvido com a tecnologia do conversor tensão-frequência.

Figura 20 – Algoritmo desenvolvido para testes de envio de frequências.

```
int ledPin1=13;
void setup() {
// put your setup code here, to run once:
pinMode(ledPin1, OUTPUT);
}
void loop() {
// put your main code here, to run repeatedly:
digitalWrite(ledPin1, HIGH);
delay(10);
digitalWrite(ledPin1, LOW);
delay(10);
}
```

Fonte: Autor.

As plataformas Arduino possuem bibliotecas no *Simulink*®, assim como o *Raspberry*®, tendo diferença significativa de *hardware*. A Figura 21 exibe a organização da pinagem do Arduino Mega 2560.

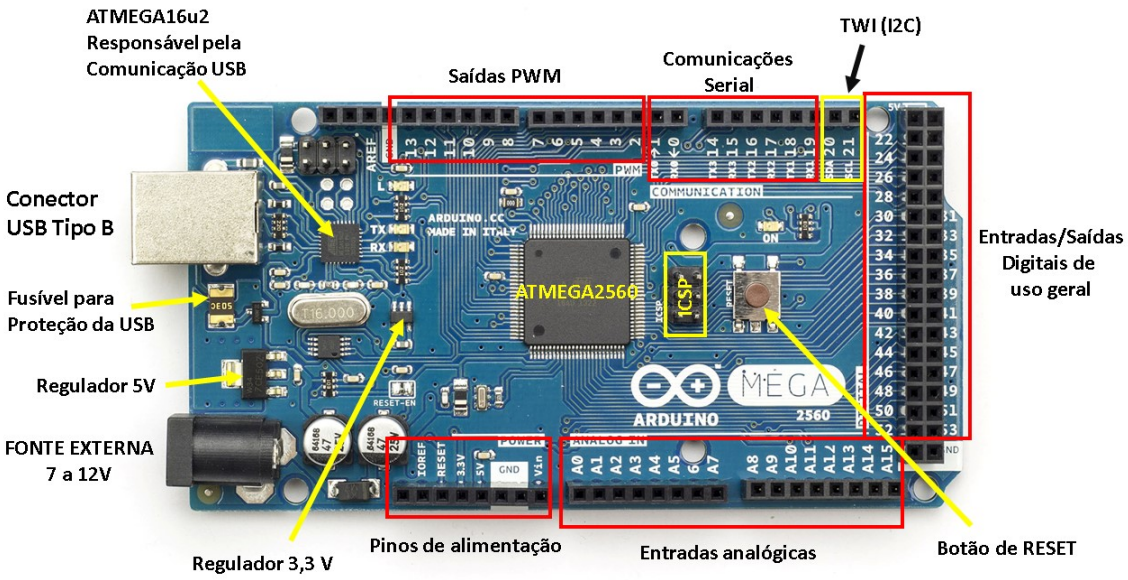
Como se pode observar na Figura 21, o Arduino dispõe de entradas analógicas e saídas PWM (*Pulse Width Modulation* - modulação por largura de pulso), as quais o *Raspberry*® Pi B não possui. Em nossa aplicação utilizaremos as saídas PWM, pois alterando a largura do pulso, podemos alterar o valor médio ($V_{RMS}$) da amplitude da tensão e variar a frequência da tensão gerada na saída do conversor tensão-frequência, variando assim a velocidade do eixo do servomotor, o que pode ser constatado por meio da equação a seguir:

$$V_{RMS} = \sqrt{V_p^2 \frac{t_p}{T} + V_n^2 (1 - \frac{t_p}{T})} \tag{4.1}$$

Onde:

- $V_p$ - valor de pico superior de tensão;
- $t_p$ - tempo de nível alto da onda (*duty cycle*);
- $T$ - período da onda e;
- $V_n$ - valor de pico inferior de tensão.

Figura 21 – Organização dos pinos do Arduino Mega 2560.




Fonte: <http://brzu.net/06f0c.>

Com a principal funcionalidade do Arduino Mega 2560 comprovada, a saber, a geração/recepção de frequências para o sistema servo, pôde-se iniciar a construção de um algoritmo que pudesse atuar e sensoriar a primeira junta do *Pendubot*, a fim de validar a proposta de utilizar este sistema como a primeira junta de atuação do *Pendubot*.

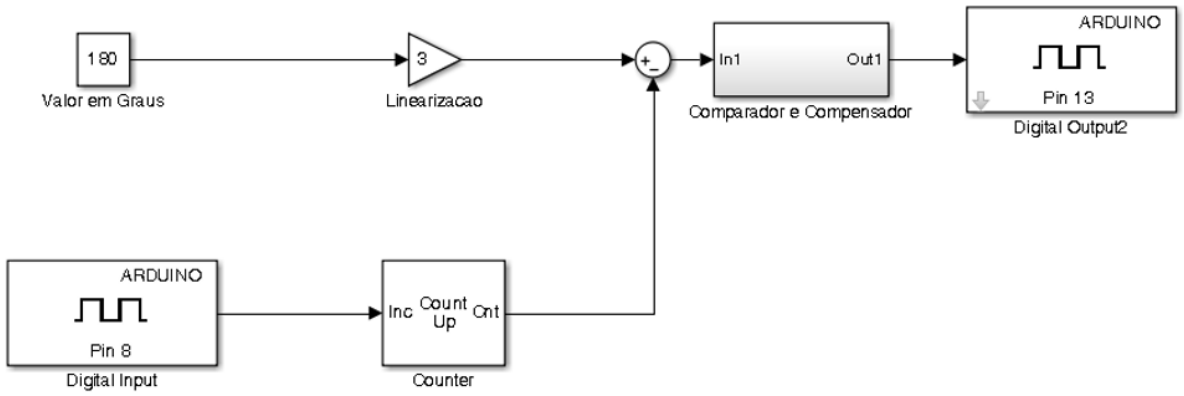
Então, foram levantados os requisitos para a construção deste algoritmo:

1. Permitir que o usuário ajuste um *setpoint* (referência) de valor de posição angular a qual o eixo do servomotor deverá atingir;
2. Receber os pulsos vindos do *encoder*;
3. Armazenar os pulsos do *encoder* para comparação com o valor de referência (*setpoint*) *e;*

4. Atuar no conversor V/F até que ele chegue à posição angular setada.

Levando em conta os requisitos, foi criada uma aplicação no *Simulink*®, a qual é exibida na Figura 22.

Figura 22 – Algoritmo de teste do controle da primeira junta para o Arduino Mega 2560.




Fonte: Autor.

Os resultados obtidos com este algoritmo foram satisfatórios e serão discutidos no capítulo 5. Algumas fotos da montagem física do sistema são ilustradas no Apêndice B.

## 4.3 Hardware da segunda junta do Pendubot

Esta secção relata as experiências obtidas na primeira junta do *Pendubot* que podem ser aplicadas à segunda junta deste robô. Serão discutidas algumas tecnologias inerentes à segunda junta e feitas propostas ao desenvolvimento desta.

### 4.3.1 Sensoriamento da segunda junta

Como a segunda junta do *Pendubot* não possui qualquer atuação (SHEPPARD, 1998), é necessário um elemento que mensure a posição angular desta junta para que seja obtido o resultado final, a saber levar o *Pendubot* à posição de equilíbrio de instável no ponto superior.

Antes de abordar sobre o dispositivo que efetuará estas medições, vejamos mais detalhes sobre o *Pendubot* de Block (1996).

#### 4.3.1.1 Posições de equilíbrio do Pendubot

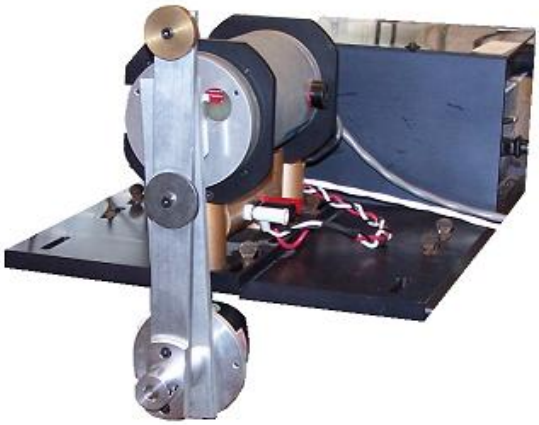
Fazendo uma análise por equilíbrio, o *Pendubot* possui algumas posições de equilíbrio, das quais 3 regiões são bem definidas: uma com equilíbrio estável e duas com equilíbrio instável. A Figura 23 exibe uma destas posições de equilíbrio, a posição de equilíbrio instável superior.

Figura 23 – Posição de equilíbrio instável superior do Pendubot - posição alta.

Fonte: <http://coecsl.ece.illinois.edu/pages/pendubot.html.>

Na Figura 1, também tem-se a ilustração destas posições. Na Figura 1, a posição 1 (também chamada *down-down*, segundo Aurelie et al. (2006) ou posição baixa) refere-se á posição na qual o sistema tende ao equilíbrio estável, quando está livre de torques internos ou externos, exceto o gerado pela força da gravidade. Já a posição 2 (também chamada *up-up,* segundo Aurelie et al. (2006) ou posição alta), é a posição de equilíbrio instável do ponto superior, na qual livre de distúrbios externos e mesmo livre de torque interno ao sistema, o sistema tende a ficar equilibrado nesta Configuração. Contudo, sendo instável, distúrbios externos podem levá-lo à posição de equilíbrio estável. A outra posição de equilíbrio instável é a exibida na Figura 24.

Figura 24 – Posição de equilíbrio instável do Pendubot - posição média.



Fonte: <http://coecsl.ece.illinois.edu/pages/pendubot.html.>

Na Figura 24, pode-se notar que qualquer perturbação no segundo elo, estando o sistema sujeito a distúrbios externos, podem levá-lo à posição de equilíbrio estável, desde

que esteja livre de torque interno ao sistema.

Outras posições de equilíbrio podem ser obtidas considerando o torque da primeira junta, contudo não serão abordadas neste Trabalho.

Durante a execução do movimento para chegar ao *setpoint*, a velocidade angular do segundo elo do Pendubot pode assumir valores muito altos, dependendo do *setpoint* ou mesmo em testes de desempenho do sistema, como o leitor pode Conferir em Sande (2009). Por isso é necessário que o dispositivo que irá receber as informações de medida de posição angular, esteja apto a trabalhar com recepção de altas frequências.

### 4.3.2 Propostas de dispositivos de sensoriamento da posição angular do segundo elo

A seguir, são discutidas algumas propostas para efetuar o sensoriamento da posição angular do segundo elo. São analisados os pontos fortes e fracos de cada proposta, bem como a possibilidade de implementação de cada uma.

#### 4.3.2.1 Potenciômetro Multivolta

A primeira ideia para contornar o problema de recepção de altas frequências, foi a utilização de um potenciômetro multivoltas, como o mostrado na Figura 25.

Figura 25 – Potenciômetro multivoltas.

Fonte: <http://brzu.net/06f1y.>

A ideia de se utilizar um dispositivo deste gênero é que, com a variação da posição angular da haste do potenciômetro, há uma variação linear no valor da resistência elétrica aferida nos terminais do componente. Assim, quando houvesse um deslocamento angular do segundo elo em relação ao primeiro elo, este valor de resistência elétrica poderia ser convertido em uma medida de deslocamento angular por um algoritmo no próprio computador no qual o controle estivesse sendo executado. Quanto à parte mecânica, não haveria tanto problema no acoplamento pois, soluções como a utilização de rolamentos para acoplar os elos e acondicionar o potenciômetro poderiam ser adotadas.

Entretanto, há um problema de ordem eletromecânica na utilização deste componente. Segundo o *datasheet* (folha de especificações) do fabricante Sharma (2014), potenciômetros

multivolta possuem uma quantidade limitada de voltas, comumente 10 voltas. Assim, poderia até ser que uma condição de contorno do sistema fosse atuar dentro de uma faixa de utilização de, por exemplo, 5 voltas para cada lado. Todavia, percebe-se que é uma condição perigosa ao analisar casos como o de Sande (2009). Por este motivo, esta ideia fica num segundo plano.

Outra ideia seria utilizar um *encoder* absoluto para mensurar o valor do deslocamento angular da segunda junta.

#### 4.3.2.2 Encoder Absoluto

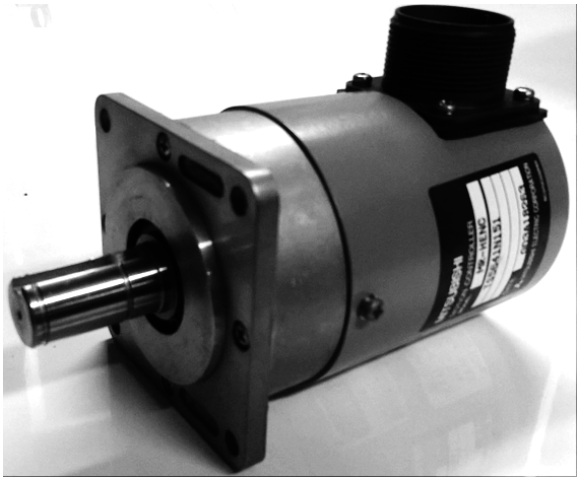
A princípio a melhor solução para resolver o problema de sensoriamento da posição angular do segundo elo, o *encoder* absoluto trás algumas vantagens em relação ao potenciômetro multivolta e também ao *encoder* incremental, como por exemplo não possuir limite de rotação e entregar um valor de posição angular já decodificado e indicando a direção do movimento, além de possuir uma espécie de *memória* que armazena estes valores, que no caso de alguns é um controlador, um equipamento que permite efetuar inclusive certos ajustes nos parâmentros do transdutor.

O grande problema do *encoder* absoluto é o seu custo de aquisição, que segundo algumas pesquisas efetuadas pelo autor deste Trabalho, chega a ser até 185% maior que o de um *encoder* incremental. Como a Instituição não possuia nenhum equipamento deste tipo até a Confecção deste Trabalho, nenhum teste foi efetuado com este tipo de componente.

#### 4.3.2.3 Encoder Incremental

Como já citado, o problema de se trabalhar com *encoders* do tipo incremental é contar a quantidade de pulsos que este envia a altas frequências. Em alguns caso, pode-se diminuir a relação precisão *versus* volume de dados, como abordado na subsecção 4.2.5.2, entretanto em outros casos, tem-se apenas uma taxa fixa de pulsos por revolução no equipamento.

Figura 26 – *Encoder* do tipo incremental com taxa fixa de pulsos por revolução.



Fonte: Autor.

Um *encoder* do tipo incremental com taxa fixa de pulsos por revolução é o ilustrado na Figura 26.

Este equipamento estava disponível na Instituição até o tempo em que este Trabalho estava sendo escrito. As principais caracterísitcas do dispositivos são listadas na Tabela 5, a seguir.

Tabela 5 – Principais características do *encoder* do tipo incremental abordado.

| Item | Especificações |
|---|---|
| Modelo | MR-HENC |
| Resolução | 16384pulsos/rev |
| Velocidade Max. | 4300rpm |
| Aceleração Max. | $40.000 rad/s^2$ |
| Consumo de corrente | 0.15A |
| Método de Comunicação | Driver diferencial/RS-422a |
| Conector do Cabo | MR-JHSCBLxM-H |

Fonte: (MITSUBISHI, 2005).

Como se pode perceber na tabela 5, a resolução do *encoder* citado é bem alta (aproximadamente 45 pulsos/revolução). No caso deste trabalho, o problema com a resolução do *encoder* está relacionada à taxa de recepção do dispositivo que efetua a interface com o computador que gerencia o algoritmo do processo de acordo com a velocidade angular que desenvolve o segundo elo. Assim, quanto maior a velocidade deste, maior é a frequência dos pulsos e menor tem que ser o período para recepção destes no dispositivo, a chamada taxa de amostragem ou *sample time.*

Como já discutido na subsecção 4.2.5.2, o Arduino perde muitas informações quando recebe frequências maiores que 12KHz. Assim, necessitaria-se de um dispositivo que efetuasse esta leitura com um *sample time* muito pequeno.

Figura 27 – Placa de aquisição de dados: opção para colher dados da segunda junta.


Fonte: <http://brzu.net/06f38.>

Nos trabalhos de Block (1996), Sheppard (1998) e Seman et al. (2010), é utilizado um componente chamado placa de aquisição de dados ou simplesmente placa de entrada/saída. Este componente é conectado diretamente no computador que gerencia a aplicação e tem compatibilidade com muitos programas utilizados para aplicações na área de controle e mecatrônica, como por exemplo o MATLAB®. Uma placa deste tipo do fabricante *National Instruments*® é exibida na Figura 27.

Foi feita então uma pesquisa para fins de levantamento de preços, compatibilidade e disponibilidade pelo autor. Os preços pesquisados ficaram na faixa de R$220,00, para as placas que possuiam compatibilidade com o MATLAB®, as quais o usuário pode consultar os fabricantes e modelos compatíveis no site da *MathWorks®*. Quanto ao item disponibilidade, não foi encontrado no Brasil, em pesquisas feitas via internet, uma opção que possuísse a compatibilidade desejada. Uma das opções encontradas foi a ACL-8111, cujas principais características eram:

- 8 canais de entrada/saída;
- Resolução de 12 bits A/D;
- Taxa de amostragem máxima de 50kS/s (*scans* por segundo);
- Compatível com *Windows*® 2000/NT/XP/9x;
- Permite programação em VisualBasic/Dephi, Turbo C e Borland C.

Há modelos muito mais novos e com maiores velocidades do que o citado, que podem ser utilizados como solução para o problema enfrentado nesta seção.

Pelo fato de ter-se trabalhado com o *encoder* do servomotor na primeira junta, que era também do tipo incremental, não foram efetuados testes com a proposta de *encoder* da segunda junta.

## 4.4 Controle do Sistema

Após verificada a realizabilidade da primeira e da segunda junta do *Pendubot*, foi iniciada a última etapa do projeto, a etapa de Controle do Sistema. Esta macroetapa compreende algumas etapas gerais que são as partes relevantes ao projeto mecânico do robô, a modelagem matemática do sistema, bem como dos controladores do mesmo, e para validar estes estudos com a simulação do projeto na ferramenta computacional MATLAB®.

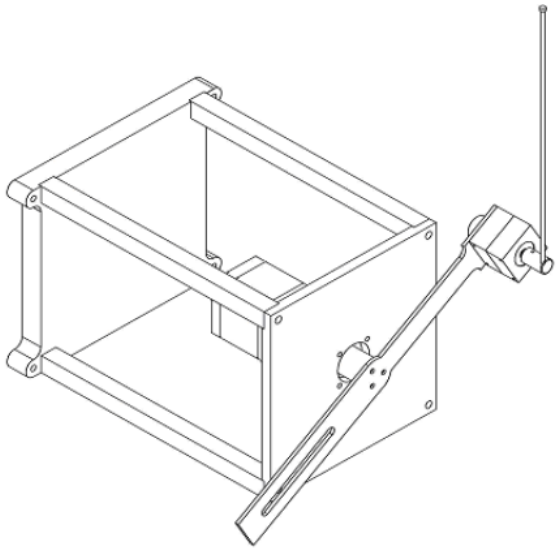
### 4.4.1 Escolha dos parâmetros mecânicos dos elos

Os parâmetros dos elos foram escolhidos com base no *hardware* que já foi proposto e também de alguns trabalhos consultados, como os de Block (1996), Sheppard (1998) e Seman et al. (2010).

#### 4.4.1.1 Proposta de elo para a Primeira junta

Como o servomotor proposto é do tipo industrial e possui potência de saída elevada (5kW) e fornece até 39,7 N.m (MITSUBISHI, 2008), é muito provável que não teremos problemas com falta de potência ou superaquecimento, por isso não serão utilizados *designs* como das Figuras 9 ou 28.

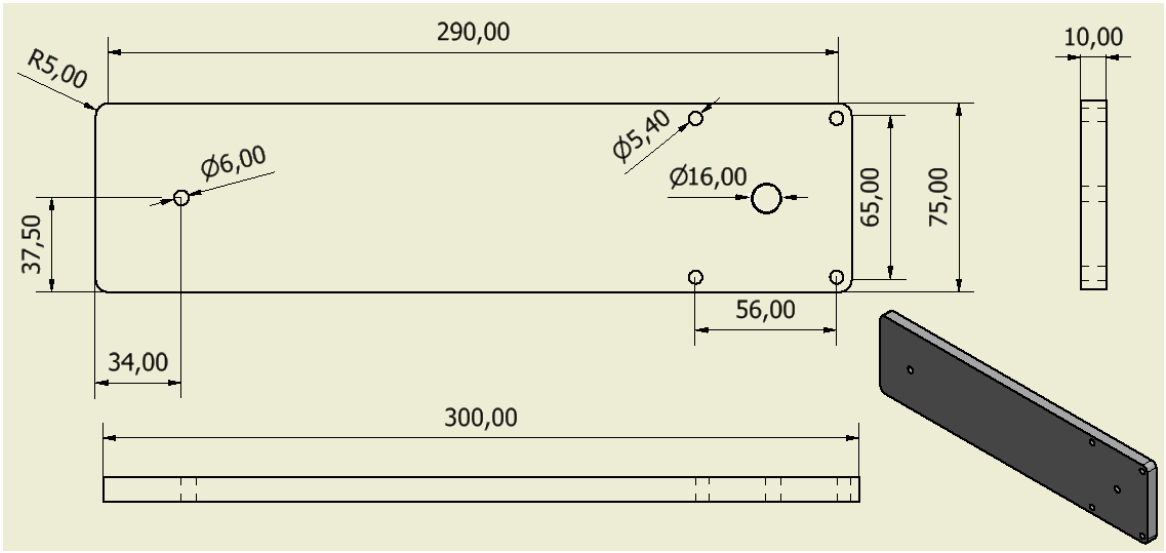
Figura 28 – Solução para ajudar no impulso do atuador da primeira junta.



Fonte: (SEMAN et al., 2010).

No caso da Figura 9, foi retirado material do primeiro elo, a fim de dimuinuir a massa deste, Conferindo uma potência relativa maior ao atuador. De modo parecido, na Figura 28, o elo é alongado para Conferir um torque auxiliar à primeira junta.

Figura 29 – Proposta de elo para a primeira junta.



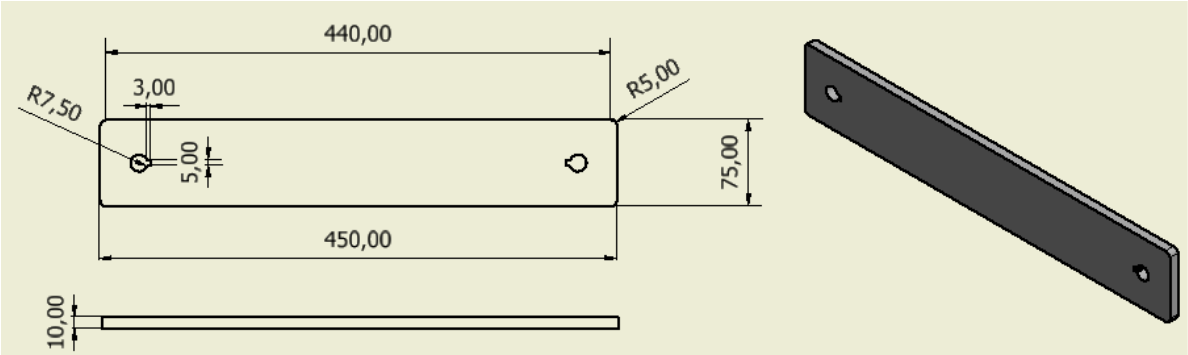

Fonte: Autor.

Na Figura 29, é ilustrada a proposta de elo para a primeira junta. O furo à esquerda é para rosca M6, mesma medida da rosca no eixo do servomotor, podendo o elo ser fixado ao eixo utilizando apenas um parafuso de mesma medida de rosca. Na outra extremidade, há quatro furos nos quais é possível fixar o *encoder* citado no item 4.3.2.3 com o auxílio de porcas, preferencialmente autotravantes, a fim de não folgarem com a utilização do sistema e nem interferir no torque da primeira junta. O furo central serve para passar o eixo do *encoder* no qual deve ser fixado o segundo elo. Salienta-se o posicionamento simétrico dos furos transversalmente e longitudinalmente. Recomenda-se que a peça seja de um metal leve, o qual pode ser alumínio como em Block (1996) e tenha espessura entre 8 e 12mm.

#### 4.4.1.2 Proposta de elo para a Segunda junta

Considerando questões geométricas (simetria), a Figura 30 exibe uma proposta de elo para a segunda junta. Neste caso não é interessante que o elo tenha redução de massa, até porque a inércia ajuda na estabilidade, inclusive trabalhos como o de Aurelie et al. (2006) utiliza-se de balanços metálicos para o acréscimo de massa neste elo. Novamente, recomenda-se que a peça seja de um metal leve e tenha uma espessura entre 8 e 12mm.

Figura 30 – Proposta de elo para a segunda junta.

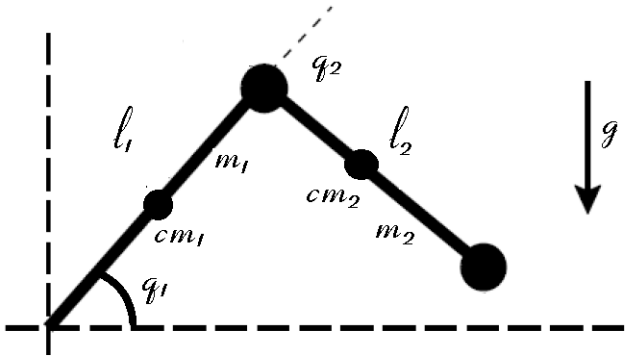



Fonte: Autor.

### 4.4.2 Modelagem matemática do sistema

Tendo uma proposta para os elos do robô, pode-se iniciar a modelagem do sistema. Sendo o *Pendubot* um sistema cujas energias dominantes são de natureza cinética e potencial, a mecânica de Lagrange pode ser aplicada em sua modelagem. A Figura 31, exibe o *Pendubot* em um instante qualquer da trajetória *swing* até o ponto de equilíbrio instável na posição superior (*up-up*).

Figura 31 – O *Pendubot* em um instante qualquer da trajetória *swing* - Figura base para a modelagem matemática.




Fonte: Autor.

O desenvolvimento das equações matemáticas referentes ao modelo dinâmico, isto é, a aplicação direta das equações de energia cinética e potencial ao modelo, bem como a organização dos seus valores na forma de matrizes, podem ser consultados na referência Aurelie et al. (2006) da página 11 à 16 e/ou Shahab (2007), da página 2 à 3, pois a análise presente neste Trabalho tem por base os resultados descritos nestas obras. As matrizes de Inércia ($D(q)$), de forças de Coriólis e centrípeta ($C(q,\dot{q})$) e de forças gravitacionais ($G(q)$) são mostradas nas equações a seguir.

$$D(q) = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}, \tag{4.2}$$

Sendo:

$$d_{11} = m_1 cm_1^2 + m_2(l_1^2 + cm_2^2 + 2l_1 cm_2 cos(q_2)) + I_1 + I_2$$

$$d_{12} = d_{21} = m_2(cm_2^2 + l_1 cm_2 cos(q_2)) + I_2$$

$$d_{22} = m_2 cm_2^2 + I_2$$

$$C(q,\dot{q}) = \begin{bmatrix} (-m_2 l_1 cm_2 sen(q_2)\dot{q}_2 & (\dot{q}_1 + \dot{q}_2)(-m_2 l_1 cm_2 sen(q_2)) \\ (-m_2 l_1 cm_2 sen(q_2))\dot{q}_1 & 0 \end{bmatrix} \tag{4.3}$$

$$g(q) = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{4.4}$$

Sendo:

$$g_1 = (m_1 cm_1 + m_2 l_1)gcos(q_1) + m_2 cm_2 gcos(q_1 + q_2)$$

$$g_2 = m_2 cm_2 gcos(q_1 + q_2)$$

Onde:

$m_1$ - massa do elo 1;

$l_1$- comprimento do elo 1;

$cm_1$ - distância do centro massa do elo 1 até a junta 1;

$I_1$ - momento de inércia do elo 1 sobre seu centróide;

$m_2$ - massa do elo 2;

$l_2$- comprimento do elo 2;

$cm_2$ - distância do centro massa do elo 2 até a junta 2;

$I_2$ - momento de inércia do elo 2 sobre seu centróide;

$g$ - aceleração gravitacional.

Na análise feita neste trabalho, foram desconsideradas as fricções presentes nas juntas. Para análise considerando estas componentes, o leitor deve consultar as referências Block (1996), Sheppard (1998), Seman et al. (2010) e/ou Aurelie et al. (2006)

As matrizes encontradas devem ser organizadas Conforme a equação a seguir:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{4.5}$$

onde o vetor $\tau$ está relacionado às forças e torques. Sendo a matriz $D(q)$ inversível, pode-se reorganizar a equação 4.5 como na equação 4.6, a seguir.

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = D(q)^{-1}\tau - D(q)^{-1}C(q,\dot{q})\dot{q} - D(q)^{-1}g(q) \tag{4.6}$$

Na qual os vetores de estados são dadas por:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} q_1 \\ \dot{q_1} \\ q_2 \\ \dot{q_2} \end{bmatrix} \tag{4.7}$$

e

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \ddot{q_1} \\ \dot{q_2} \\ \ddot{q_2} \end{bmatrix} \tag{4.8}$$

Com a modelagem matemática pronta, pode-se então partir para o controle do sistema.

### 4.4.3 Controle do Sistema

Para tornar o sistema mais robusto, o controle do *Pendubot* é dividido em duas etapas:

- Controle de *swing*: este controle atua na trajetória desde o ponto de inércia (posição de equilíbrio estável ou *down-down*) até próximo ao ponto de referência da posição superior (posição de equilíbrio instável superior ou *up-up*) e;
- Controle de balanço: este controle atua em regiões próximas à região que compreende o *setpoint* (referência), com o objetivo de ajudar a estabilizar o sistema, além permitir a rejeição de pequenos distúrbios (AURELIE et al., 2006).

A seguir, será descrito como proceder para efetuar o controle do sistema, efetuando-se a linearização do mesmo: o Controle de *swing*.

#### 4.4.3.1 Controle de swing

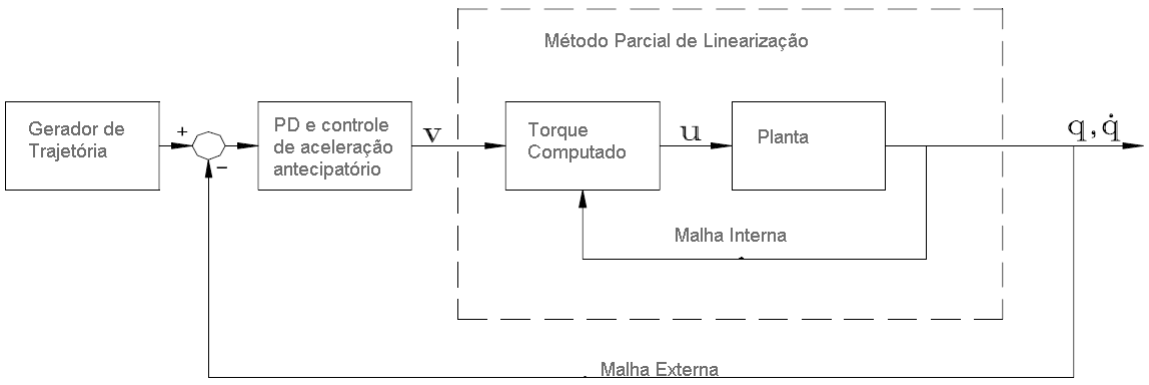
Desenvolvendo a equação 4.5, temos que as equações para as duas juntas do *Pendubot* são:

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{11}\dot{q}_1 + c_{12}\dot{q}_2 + d_1 = \tau_1 \tag{4.9}$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{21}\dot{q}_1 + g_2 = 0 \tag{4.10}$$

Como não se tem atuação na segunda junta do *Pendubot*, é necessário realizar uma linearização do sistema, a qual para este projeto, foi escolhido o método de linearização por realimentação parcial, utilizado em Sheppard (1998), no qual o intuito geral é linearizar uma variável em função de outra variável, assim diz-se que há linearização de um grau de liberdade.

Figura 32 – Diagrama de blocos de controle por linearização por realimentação parcial.




Fonte: (BLOCK, 1996), adaptado pelo autor.

Essa linearização permite que seja projetado um controlador de malha fechada que vai dirigir a trajetória de um destes graus de liberdade. Para o *Pendubot*, a junta que pode ser linearizada é a primeira, que é a junta que possui atuação. Dessa forma, podemos obter o

comportamento da segunda junta dada uma excitação na primeira junta. Na Figura 32, é exibido um diagrama de blocos que exibe o controle por linearização por realimentação parcial.

Escrevendo $\ddot{q}_2$ em função de $\ddot{q}_1$ na equação 4.10, tem-se:

$$\ddot{q}_2 = \frac{-d_{21}\ddot{q}_1 - c_{21}\dot{q}_1 - g_1}{d_{22}} \tag{4.11}$$

Por substituição na equação 4.9, tem-se que:

$$\bar{d}_{11}\dot{q}_1 + \bar{c}_{11}\dot{q}_1 + \bar{c}_{12}\dot{q}_2 + \bar{g}_1 = \tau_1 \tag{4.12}$$

onde:

$$\bar{d}_{11} = d_{11} - \frac{d_{12}d_{21}}{d_{22}}$$

$$\bar{c}_{11} = c_{11} - \frac{d_{12}c_{21}}{d_{22}}$$

$$\bar{c}_{12} = c_{12}$$

$$\bar{g}_1 = g_1 - \frac{d_{12}g_2}{d_{22}}$$

Na Figura 32, o controle de malha interna que lineariza a variável $q_1$ pode ser definido como:

$$\tau_1 = \bar{d}_{11}v_1 + \bar{c}_{11}\dot{q}_1 + \bar{c}_{12}\dot{q}_2 + \bar{g}_1 \tag{4.13}$$

Resultando no sistema:

$$\ddot{q}_1 = v_1 \tag{4.14}$$

e

$$d_{22}\ddot{q}_2 + c_{21}\dot{q}_1 + g_2 = -d_{21}v_1 \tag{4.15}$$

Dessa forma, a equação 4.14 é linear e a equação 4.15 é considerada linear com relação a $v_1$. Para este projeto, foi considerado um PD com controle de aceleração antecipatório, cuja equação é dada por:

$$v_1 = \ddot{q}_1^d + K_d(\dot{q}_1^d - \dot{q}_1) + K_p(q_1^d - q_1) \tag{4.16}$$

Foi utilizado um controlador PD porque, segundo Block (1996), quando utilizado no sistema real, o controlador PID trabalha relativamente bem, entretanto amplifica ruídos

numéricos. A linearização por realimentação parcial precisa da realimentação da posição de ambas juntas, e leva em conta os efeitos não lineares da ligação, fato esse que cria um controle muito mais "limpo" em comparação com um controle PID, que deve rejeitar os efeitos do primeiro e do segundo elo.

#### 4.4.3.2 Controle de balanço

No controle de balanço, efetua-se a linearização do sistema em torno do ponto de equilíbrio em questão. Aplicando a equação 3.3 ao *Pendubot*, temos que a forma geral desta é:

$$f_a(x,u) = f_a(x_r,u_r) + \frac{\partial f}{\partial x}|_{x_r u_r}(x - x_r) + \frac{\partial f}{\partial u}|_{x_r u_r}(u - u_r) \tag{4.17}$$

Na equação descrita anteriormente, $f_a(x_r, u_r)$ deve ser sempre *zero* pois apenas os pontos de equilíbrio são interessantes à aplicação. A função $f$ está relacionada aos estados citados na equação 4.8 e o termo relacionado a $u$ refere-se aos componentes de torque ($\tau$) na equação 4.6.

Para calcular a matriz $A$ e a matriz $B$, deve-se calcular as derivadas parciais da equação 4.18 e também da equação 4.19 e após isso, aplicar no(s) ponto(s) de equilíbrio descritos a seguir.

$$A = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix}, \tag{4.18}$$

Sendo:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \ddot{q}_1 \\ \dot{q}_2 \\ \ddot{q}_2 \end{bmatrix}$$

e

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix}$$

$$B = \frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \\ \frac{\partial f_3}{\partial u} \\ \frac{\partial f_4}{\partial u} \end{bmatrix}, \tag{4.19}$$

Lembrando que $\ddot{q}_1$ e $\ddot{q}_2$ são descritos pela equação 4.6.

Como o objetivo aqui é linearizar em torno de um ponto de operação, no qual este ponto é definido como ponto de equilíbrio instável superior (*up-up*), tem-se que as condições de equilíbrio são $q_1 = \frac{\pi}{2}$, $q_2 = 0$, $\dot{q}_1 = 0$ e $\dot{q}_2 = 0$, que são as condições necessárias para que o *Pendubot* esteja em equilíbrio instável no ponto superior ou *up-up*.

Após efetuar o cálculos das derivadas parciais e a substituição das condições de equilíbrio, são encontradas as matrizes $A$ e $B$. Sabendo-se então as matrizes $A$ e $B$, pode-se projetar o controlador por realimentação de variáveis de estado completo, o qual é definido com base nas matrizes R e Q, as quais o usuário pode definir de acordo com testes ou experiência. Pode-se ainda definir valores de peso para estas matrizes Conforme a subsecção 3.3. Aqui foram adotados os pesos 1, e como se quer trabalhar nos ganhos de $q_1$ e $q_2$, apenas os termos 11 e 33 da matriz Q têm estes ganhos.

Assim, as matrizes R e Q são:

$$R = [1]$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

O último passo que é o cálculo do controlador, pode ser feito por meio da resolução da equação 3.10 ou no MATLAB®, utilizando o comando "*lqrd*", o qual recebe como parâmetros as matrizes $A$, $B$, $Q$, $R$ e um parâmetro de taxa de amostragem ($Ts$), o qual o usuário pode Configurar Conforme a necessidade do projeto do controlador (ganho ótimo).

Neste Trabalho, as matrizes $A$ e $B$ foram calculadas por meio de algoritmos (*script m file*) e foi simulado o controle de balanço com os resultados obtidos. O controle de *swing* também foi simulado, sendo os parâmetros escolhidos por meio da sintonia fina do controlador, o qual será discutido no capítulo referente aos Resultados Obtidos. Também serão abordadas as simulações realizadas.

# 5 RESULTADOS OBTIDOS E DISCUSSÕES

Neste capítulo serão abordados os principais resultados obtidos, referentes à primeira junta e a simulação com os resultados da modelagem matemática do sistema. Comparações com outros trabalhos e possibilidades de melhorias também serão abordadas.

## 5.1 Primeira junta

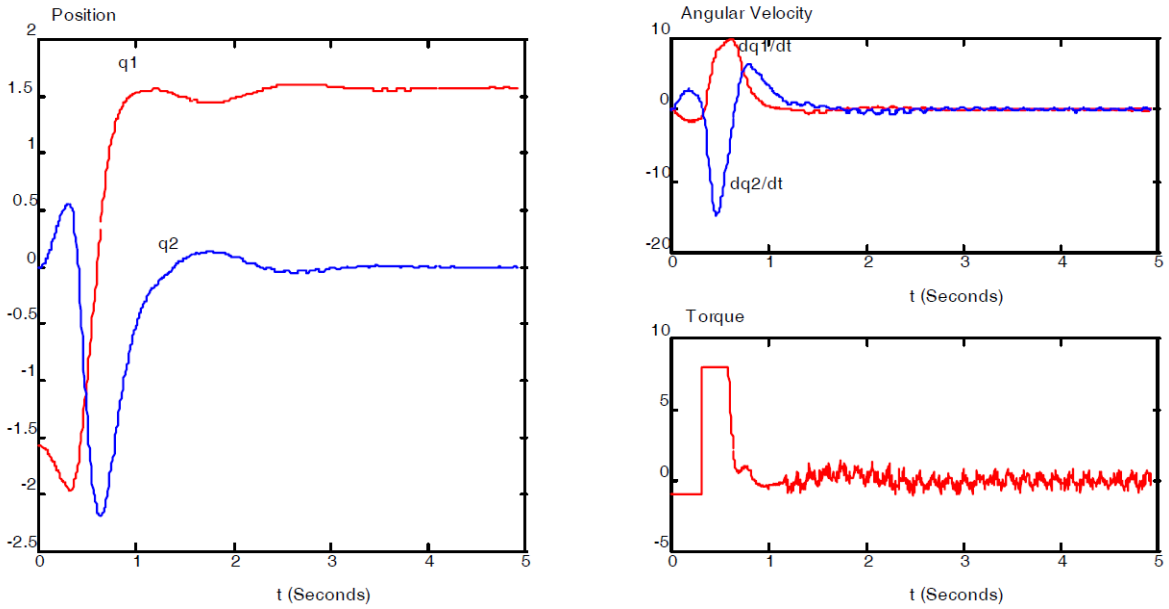
Como exibido na Figura 22, foi criado um algoritmo em linguagem de diagrama de blocos no *Simulink*® para validar o funcionamento da primeira junta do *Pendubot*.

Na aplicação, o usuário pode definir um valor de *setpoint* (referência) em graus, entre 0 e 360º. O bloco chamado comparador e compensador, recebe o sinal vindo do somador que efetua a aritmética entre o valor de referência e o valor de pulsos acumulado no contador e atua conforme a necessidade no conversor tensão-frequência (por meio da porta 13 do Arduino, no caso da Figura 22). Conforme o valor de posição do *encoder* incrementa o contador, tem-se uma diminuição no sinal recebido pelo bloco compensador e comparador, o que ocorre até a chegada no valor de referência.

Este algoritmo simples, valida a proposta de aplicação dos componentes utilizados neste teste como possibilidade para atuação da primeira junta do *Pendubot*. No quesito servomotor, não há problemas com relação a falta de torque, pois o servomotor possui torque suficiente para ser utilizado em aplicações de alta demanda de torque, como por exemplo movimento de engrenagens motrizes de máquinas industriais (MITSUBISHI, 2007). Também não apresenta problemas com relação a superaquecimento, como no caso de Block (1996) onde o motor CC utilizado na aplicação não possuia torque elevado o suficiente para cumprir a trajetória do ponto *down-down* até *up-up* diretamente. A solução encontrada foi fornecer uma quantidade de energia cinética à segunda junta primeiramente no sentido oposto ao movimento, para então inverter o sentido de torque e chegar em uma zona próxima a referência. Este resultado pode ser verificado na Figura 33, onde é possível visualizar o torque do motor no sentido inverso e a saturação de torque no nível positivo. É possível notar ainda que ruídos estão presentes quando o sistema está na região de regime permanente. Para comparação, a Figura 34 mostra o resultado simulado obtido por Block.

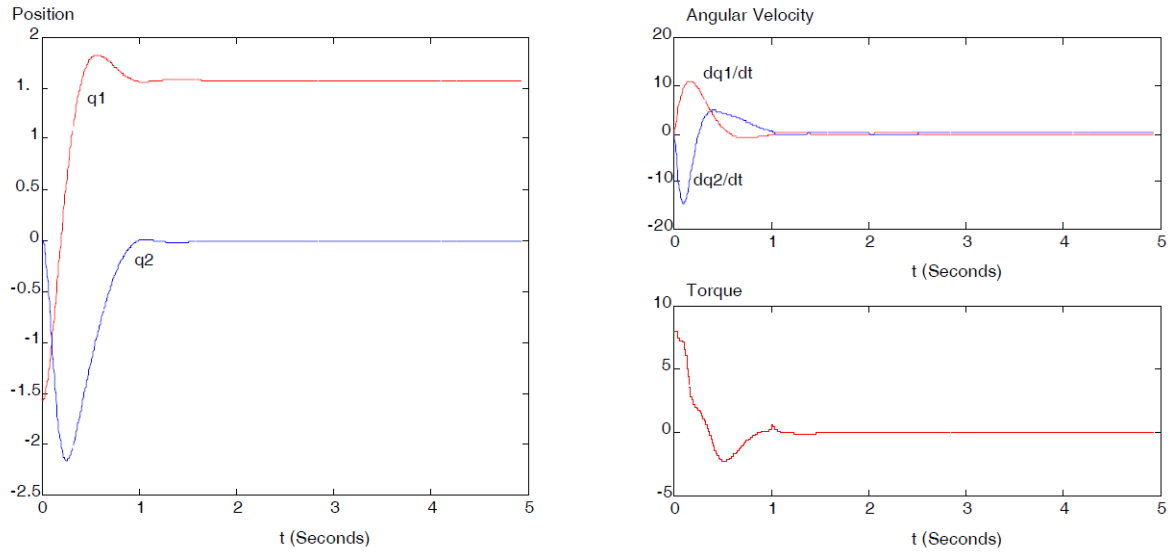
Algumas melhorias podem ser aplicadas ao *hardware* proposto para a primeira junta, como é o caso do trem de pulsos enviado ao MR-TB20. Neste trabalho, utilizou-se como tensão de excitação no conversor tensão-frequência, a tensão de 5V enviada pelos pinos da plataforma Arduino. Entretanto, pode-se aplicar até 10V, segundo UNIDIGITAL (2013) neste conversor e obter frequências de até 50KHz, dobrando a velocidade obtida. Outra opção seria utilizar o CI LM331 (FAIRCHILD, 2001) e obter frequências de até 100KHz, prodizindo uma velocidade quatro vezes maior que a apresentada na subsecção 4.2.4.

Figura 33 – Resultado experimental obtido por Daniel J. Block.




Fonte: (BLOCK, 1996).

Figura 34 – Resultado simulado obtido por Daniel J. Block.




Fonte: (BLOCK, 1996).

Uma outra opção, seria a utilização dos modos torque e velocidade, que necessitam apenas de um valor analógico de tensão para referência em vez de um trem de pulsos. Neste caso, teria que ser feito novo estudo para verificar os modos de controle do *Pendubot*, isto é, a relação existente entre a saída do algoritmo no *software* de controle e o controle no servoamplificador propriamente dito. Os modos híbridos (citados na secção 4.2.3) também podem ser uma opção de uso.

## 5.2 Simulação do modo swing utilizando o MATLAB®.

Para as simulações do modo *swing* e modo balanço, foram usadas as medidas dos elos ilustrados nas Figuras 29 e 30, bem como os seguintes parâmetros:

- Massa do primeiro elo: $m_1 = 0,49Kg$;
- Massa do segundo elo: $m_2 = 0,76Kg$;
- Centro de massa do primeiro elo: $cm_1 = 0,15m$;
- Centro de massa do segundo elo: $cm_2 = 0,227m$;
- Momento de inércia do primeiro elo: $I_1 = 0,0037Kg * m^2$;
- Momento de inércia do segundo elo: $I_2 = 0,0128Kg * m^2$;

Obs: os valores das massas foram calculados considerando que o material das juntas fosse alumínio, com densidade aproximada $d = 2697kg/m^3$, largura de cada elo $w = 75mm$ e espessura $e = 10mm$ (valores que geram as massas citadas, considerando os dimensionais escolhidos) e os centros de massa foram obtidos considerando a simetria dos elos.

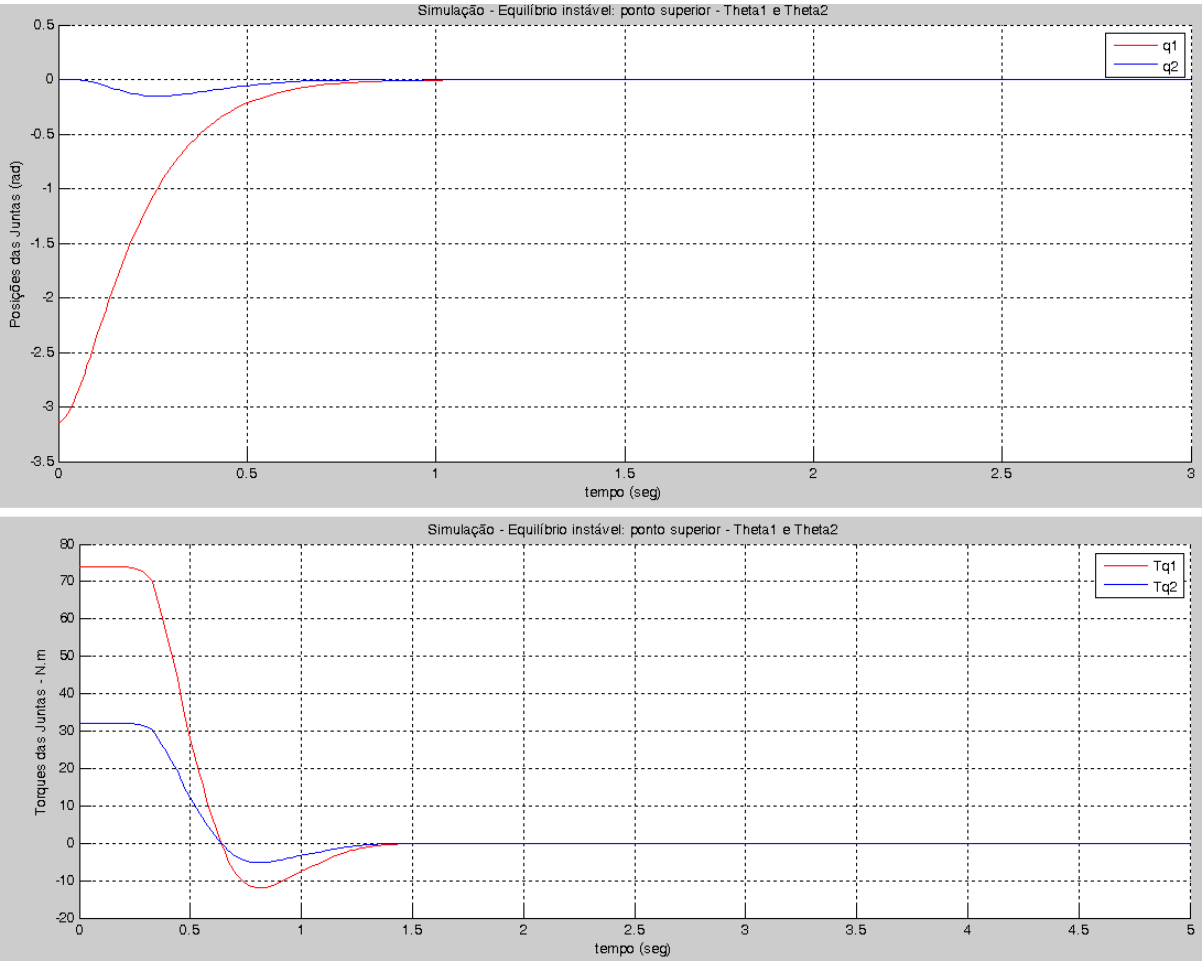
Então, com os resultados das equações 4.6, 4.14, 4.15 e 4.16, foi possível construir uma aplicação (*script m file*) para o modo *swing* no MATLAB®, baseada nos algoritmos desenvolvidos por Shahab (2007), simulando a resposta das duas juntas do *Pendubot* de forma gráfica e também de forma animada (vídeo), na qual é possível visualizar a movimentação das juntas do ponto *down-down* até o ponto *up-up*. A resposta gráfica é ilustrada na Figura 35, na qual são exibidas as posições angulares das duas juntas com o decorrer do tempo e também o torque em ambas as juntas.

Os parâmetros de controle citados na Figura 35 foram os que geraram o melhor resultado. Uma resposta um tanto oscilatória, também é exibida como resultado na Figura 36, bem como seus parâmetros de controle.

Comparando as respostas das Figuras 35 e 36, percebe-se que o conjunto de valores do controlador da Figura 36 torna o sistema bem mais oscilatório e gera um atraso no tempo de subida, bem como leva um tempo bem maior que no primeiro caso para se acomodar.

Agora, comparando a Figura 35 com a Figura 33, é possível perceber que o sistema simulado apresenta uma resposta mais 'suave', com um tempo de subida menor e um tempo de acomodação também menor. Esta mesma comparação vale para a resposta simulada de Block. Vale ressaltar entretanto que, neste trabalho não foi considerada a questão do atrito nas juntas, a qual nos trabalhos de Sheppard (1998) e Block (1996), foram colhidas informações do próprio sistema, isto é, do *Pendubot* real, e utilizado algoritmo de otimização nos valores. Foi feita ainda a parametrização de certos valores no modelo dinâmico e depois, incluídas nestes parâmetros, os componentes coletados do sistema considerando o atrito, como uma espécie de identificação. Sheppard (1998) teve resultados

Figura 35 – Resultados obtidos na simulação do modo *swing* com parâmetros do controlador PD $K_p = 98,7$ e $K_d = 24,1$.




Fonte: Autor.

próximos ao de Block e Seman não chegou a utlizar o modo *swing* mas apenas o modo balanço (SEMAN et al., 2010).

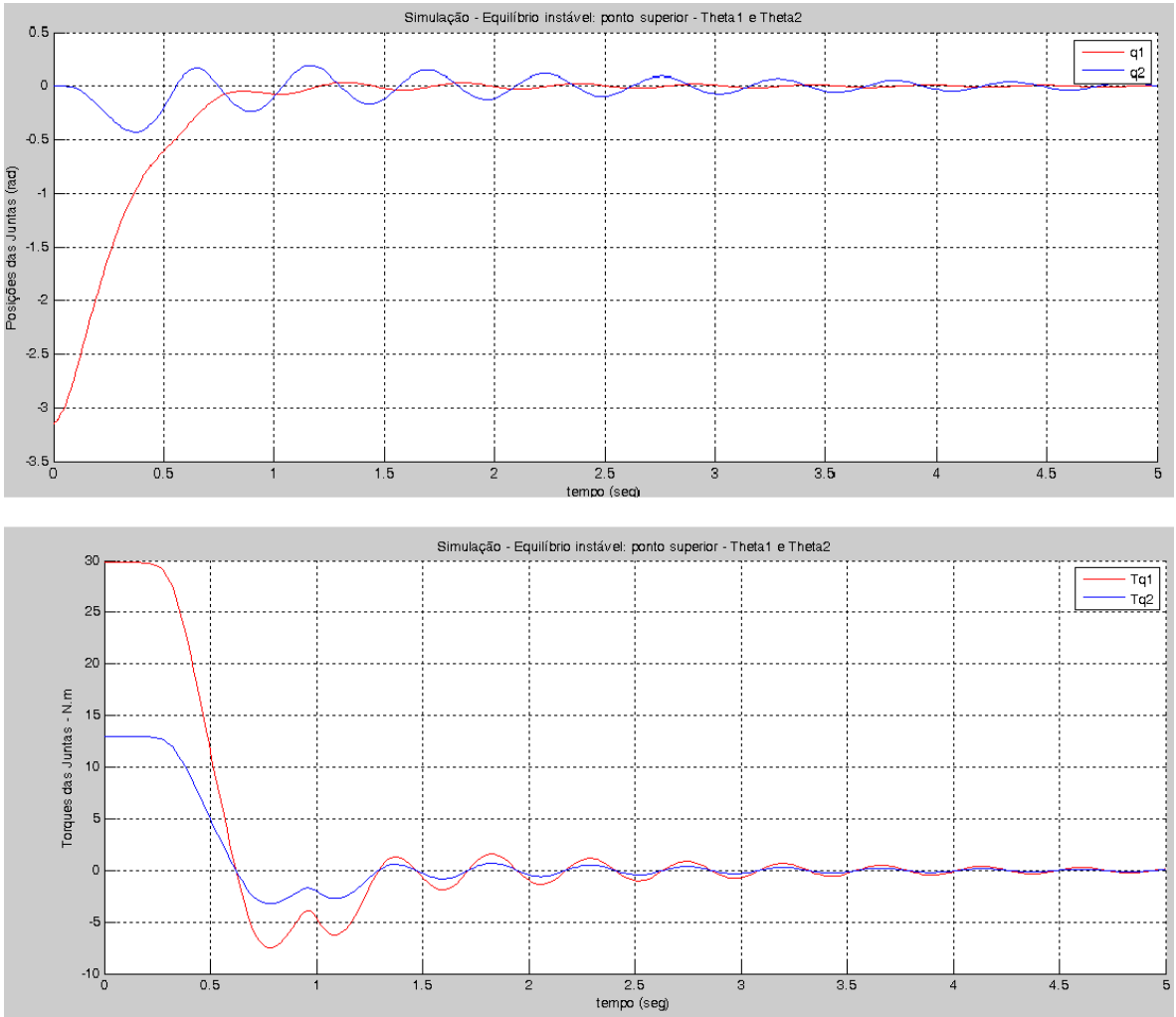
Como forma de visualizar fisicamente o exibido na Figura 35, a saber as posições do *Pendubot* com o decorrer do tempo, são exibidos na Figura 37 alguns instantâneos do sistema. No apêndice A, o leitor pode consultar os algoritmos (*script m file*) utilizados para construir as simulações cujos resultados são os obtidos nas Figuras 35, 36 e 37.

## 5.3 Simulação do modo balanço utilizando o Simulink®.

Utilizando as equações 4.18 e 4.19, o comando "*lqrd*" e os valores das especificações escolhidas para o *Pendubot*, os seguintes valores para as matrizes foram encontrados:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 40,3121 & 0 & -53,3635 & 0 \\ 0 & 0 & 0 & 1 \\ -47,8904 & 0 & 139,0445 & 0 \end{bmatrix}$$

Figura 36 – Resultados obtidos na simulação do modo *swing* com parâmetros do controlador PD $K_p = 51,7$ e $K_d = 19,1$.

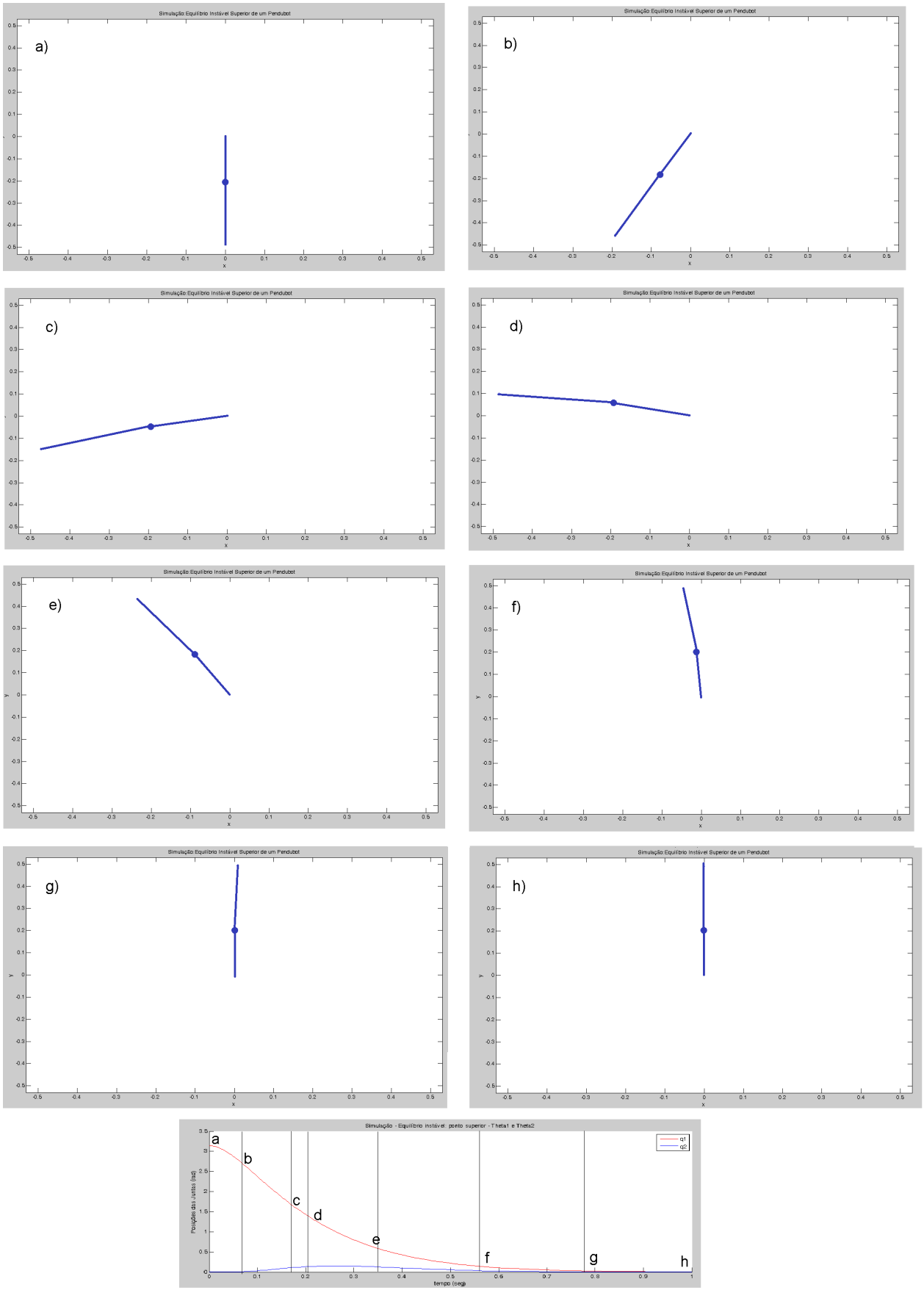



Fonte: Autor.

$$B = \begin{bmatrix} 0 \\ 31,6716 \\ 0 \\ -63,2025 \end{bmatrix}$$

$$K = \begin{bmatrix} -18,1913 & -4,681 & -18,1426 & -2,8713 \end{bmatrix}$$

Figura 37 – Instantâneos da simulação do *Pendubot* em modo *swing*.



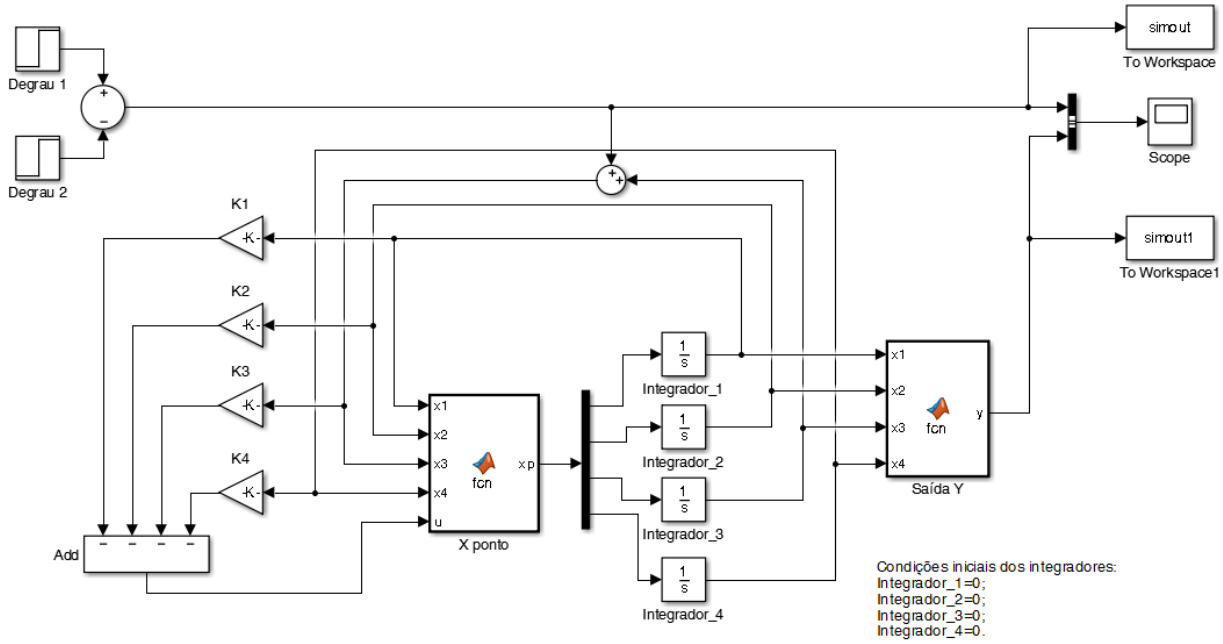

Fonte: Autor.

Como as variáveis que se quer analisar o comportamento são $q_1$ e $q_2$, a matriz $C$ que está relacionada a saída $y$ é dada por $y = Cx$, sendo:

$$C = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

O algoritmo para cálculo das matrizes $A$, $B$ e $K$ pode ser consultados no Apêndice A. Então foi feito um algoritmo em *Simulink* ® para a simulação de resposta do sistema aos distúrbios, sendo o diagrama de blocos exibido na Figura 38, a seguir.

Figura 38 – Diagrama de blocos para simulação do modo balanço do *Pendubot*.



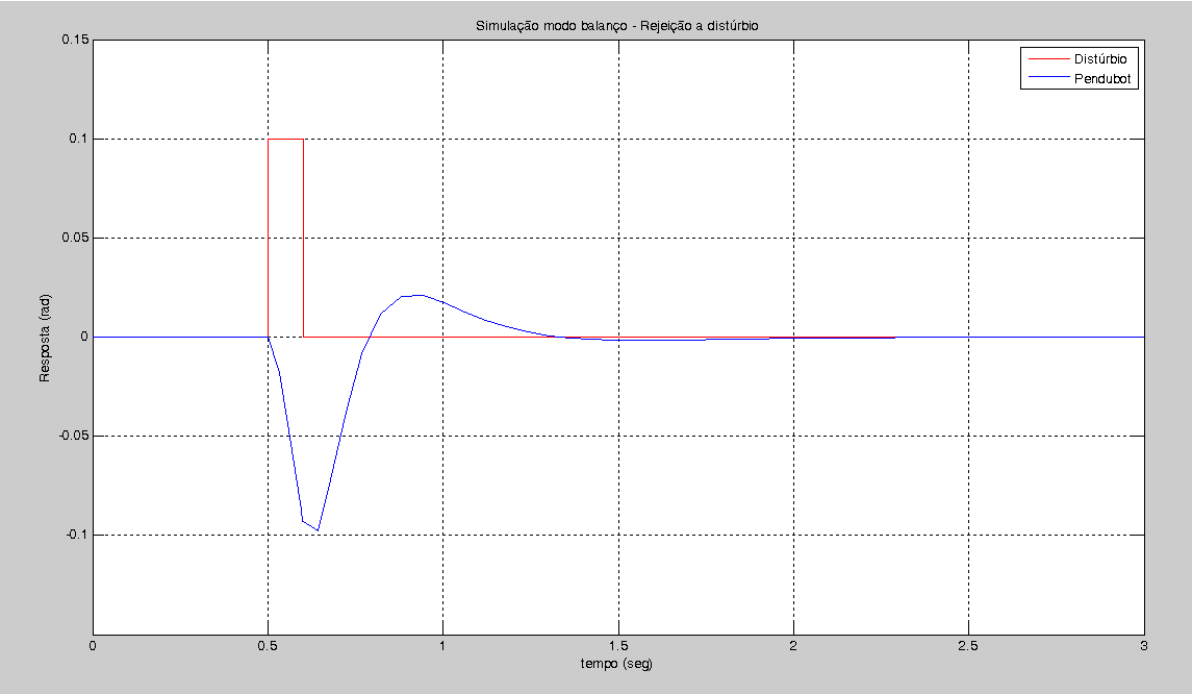

Fonte: Autor.

Como ilustrado na Figura 38, as condições dos integradores são as condições de repouso no ponto de equilíbrio instável superior ou *up-up*. O bloco rotulado como *x ponto* contém a equação $\dot{x} = Ax + Bu$, o bloco rotulado como y contém a equação $y = Cx$ e os blocos rotulados como k1, k2, k3 e k4, contém cada valor individualmente da matriz $K$, o controlador de realimentação de estados completo. Na parte superior esquerda desta Figura, há dois degraus em simetria acionados em 0,5 e 0,6s, respectivamente, gerando assim um pulso de amplitude 0,1 e de largura 0,1, simulando um distúrbio na segunda junta do *Pendubot*. Como é possível verificar na Figura 39, o *Pendubot* consegue rejeitar estes distúrbio voltando a posição de equilíbrio.

## 5.4 Sugestões de trabalhos futuros

Para melhoria da proposta atual, algumas sugestões são apresentadas nesta secção, bem como trabalhos a serem realizados futuramente.

Figura 39 – Rejeição a distúrbios do modo balanço.




Fonte: Autor.

1. Melhorias no acionamento da primeira junta: como comentado no item 5.1, a primeira junta possui limitações de velocidade, por conta da alimentação do conversor tensão-frequência que está fixado em 5V, máxima tensão enviada pelos pinos do Arduino. Um amplificador baseado em tecnologia MOSFET (transistor de efeito de campo metal óxido semicondutor) pode ser utilizado para construir um conversor de tensão contínua elevador de 5V para 10V. Isto dobraria a frequência do trem de pulsos do pulsos, dobrando a velocidade do eixo do servomotor. Um circuito baseado no LM331 também poderia ser construído, permitindo alcançar frequências de até 100Khz, aumentando a frequência por um fator de quatro vezes. Outra opção seria a utilização dos modos torque e velocidade para o controle do servomotor ou ainda a utilização de modos híbridos;

2. Melhorias na recepção de pulsos dos *encoders*: como abordado no item 4.2.5.2, o Arduino possui limitação quanto a recepção de frequências. Para resolver este problema, poderia ser utilizada uma placa de aquisição de dados que possui taxa de recepção bem maior que a atual proposta, opção essa explicada com mais detalhes no item 4.3.2.3.;

3. Construção e montagem mecânica do *Pendubot*: a confecção dos elos e o acoplamento destes aos eixo do servomotor bem como ao eixo do *encoder* na segunda junta, permitiriam realizar esta etapa;

4. Testes e obtenção de resultados experimentais do sistema: com as melhorias de

acionamento da primeira junta, uma recepção de pulsos dos *encoders* a taxas mais altas e tendo o sistema mecanicamente montado, os primeiros testes do *Pendubot* podem ser efetuados. A partir disso, podem ser relacionadas as grandezas *virtuais*, isto é, os comandos executados no algoritmo de controle do sistema com as grandezas *reais*, como tensões, torques e correntes, para finalmente obter os primeiros experimentais de controle do *Pendubot*;

5. Equilíbrio instável em diferentes posições: o intuito neste Trabalho foi estabilizar o *Pendubot* no ponto de equlíbrio instável superior. Entretanto, há outros pontos de equlíbrio instável do *Pendubot*, como por exemplo o ponto exibido na Figura 24;

6. Utilizaçao de técnicas de Controle diferenciadas: utilização de outras técnicas de Controle, como Controle híbrido ou Controle Preditivo para estabilização do Pendubot em pontos a determinar.

Outros pontos podem ser otimizados, pelo fato de o Trabalho apresentado se tratar de uma proposta e sendo uma ferramenta didática, voltada a fins acadêmicos, o *Pendubot* permite a obtenção de conhecimento de múltiplas áreas com o foco no equilíbrio instável.

# 6 CONCLUSÕES

Os resultados obtidos no decorrer deste trabalho, expuseram a possibilidade de efetuar o Controle do robô de pêndulo (*Pendubot*), utilizando ferramentas computacionais consagradas como o MATLAB® da *MathWork*® e o ambiente de simulação *Simulink*®. Com servoamplificador, servomotor e seus periféricos, além de outros dispositivos como circuitos integrados (CI's) específicos, conversores de tensão-frequência, interface de comunicação entre o sistema servo e o computador que executa o algoritmo de controle do sistema e transdutores que permitam mensurar a posição angular das juntas do robô, são alguns componentes do *hardware* necessário ao funcionamento da proposta. No tocante ao Controle, foi desenvolvida a modelagem matemática do sistema, bem como técnicas específicas para obtenção da proposta de controle, como a linearização e a técnica por alocação de polos, sendo efetuada a sintonia dos controladores tanto do modo *swing* quanto do modo balanço. Considerações de melhorias foram apresentadas bem como uma visão geral para a montagem e posterior funcionamento do sistema fisicamente.

# Referências


AURELIE, D. et al. *Final REPORT Pendubot*. [S.l.], 2006. 12, 18, 46, 52, 53, 54, 55

BLOCK, D. J. *Mechanical Design and Control of the Pendubot*. Dissertação (Mestrado) — University of Illinois, 1996. 17, 45, 50, 52, 54, 55, 56, 59, 60, 61

BOYLESTAD, R. L.; NASHELSKY, L. *Dispositivos eletrônicos e teoria de circuitos*. [S.l.]: Pearson Prentice Hall, 2004. v. 8. 27, 38

BRAGA, N. *7404 - TTL (IP268)*. 2015. Disponível em: <http://www.newtoncbraga.com.br/index.php/ideias-dicas-e-informacoes-uteis/45-circuitos-integrados-ttl/2822-ip268>. 27

BRAGA, N. *Como funciona o LM331 - conversor tensão-frequência de precisão (ART851)*. 2015. Disponível em: <http://www.newtoncbraga.com.br/index.php/como-funciona/6857-como-funciona-o-lm331-conversor-tensao-frequencia-de-precisao-art851>. 26

CRAIG, J. J. *Introduction to robotics: mechanics and control*. [S.l.]: Addison Wesley Longman, 1989. 20

DAS, S. et al. Lqr based improved discrete pid controller design via optimum selection of weighting matrices using fractional order integral performance index. *Applied Mathematical Modelling*, v. 37, p. 4253–4268, 2012. 22

DORF, R. C.; BISHOP, R. H. *Sistemas de Controle Modernos*. [S.l.]: LTC, 2009. 13, 21, 43

FAIRCHILD, S. *LM331 V-F Converter*. 1. ed. [S.l.], 2001. 26, 37, 59

FAIRCHILD, S. *Datasheet CD4049UBC, CD4050BC, Hex Inverting Buffer, Hex Non-Inverting Buffer*. [S.l.], 2002. 27, 38

GOUVEIA, D. C. D.; ANDRADE, F.; BALTAZAR, P. M. *ESTUDO E SIMULAÇÃO DE CONTROLE DE TORQUE EM SISTEMAS SERVOACIONADOS*. Dissertação (Mestrado) — UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ, 2013. 24

GROOVER, M. *Automação Industrial e Sistemas de Manufatura*. [S.l.]: Pearson Prentice Hall, 2011. 22, 23, 24

GUIDORIZZI, H. L. *Um Curso de Cálculo Vol. 1*. [S.l.]: LTC, 2008. 21

KELLY, R.; DAVILA, V. S.; LORíA, A. *Control of Robot Manipulators in Joint Space*. [S.l.]: Springer-Verlag London, 2005. 20, 22, 23

KURNIAWAN, A. *Getting Started with Matlab Simulink and Arduino*. [S.l.]: Agus Kurniawan, 2013. 25

KURNIAWAN, A. *Getting Started with Matlab Simulink and Raspberry Pi*. [S.l.]: Agus Kurniawan, 2013. 39

MATHWORKS. *The Language of Technical Computing*. 2015. Disponível em: <http://www.mathworks.com/products/matlab/index-b.html>. 27

MITSUBISHI, C. E. *General-Purpose AC Servo MELSERVO J2-Super Series SSCNET Compatible MODEL MR-J2S- B - SERVO AMPLIFIER INSTRUCTION MANUAL.* [S.l.], 2004. 30

MITSUBISHI, C. E. *Motion Controller Q series - Q173CPU(N) and Q172CPU(N).* [S.l.], 2005. 49

MITSUBISHI, C. E. *General-Purpose AC Servo MELSERVO J2-Super Series General-Purpose Interface MODEL MR-J2S- A - SERVO AMPLIFIER INSTRUCTION MANUAL.* [S.l.], 2007. 31, 32, 35, 36, 38, 42, 59

MITSUBISHI, C. E. *General-Purpose AC Servo Melservo MODEL Servo Motor - INSTRUCTION MANUAL.* [S.l.], 2008. 51

MITSUBISHI, C. E. *General-Purpose AC Servo MELSERVO Servo Configuration Software MODEL MRZJW3- SETUP161E INSTALLATION GUIDE.* [S.l.], 2014. 33, 34, 35

NICOLOSI, D. E. C. *Micrrocontrolador 8051 Detalhado.* [S.l.]: Erica, 2002. 25

OGATA, K. *Engenharia de Controle Moderno.* [S.l.]: PEARSON PRENTICE HALL, 2003. 21

RAHMAN, A.; ALI, S. M. Design and analysis of a quadratic optimal control system for a type one plant model. *International Journal of Research in Engineering & Technology*, v. 1, p. 177–186, 2013. 22

SANDE, C. *Pendubot.* 2009. Disponível em: <https://www.youtube.com/watch?v=Wz1t7McSnUg>. 47, 48

SEMAN, P. et al. Control of laboratory model of pendubot. In: *International Conference CYBERNETICS AND INFORMATICS.* [S.l.: s.n.], 2010. 13, 18, 50, 51, 54, 62

SHAHAB, M. *2DOF Robotic Manipulator - Control Design & Simulation.* [S.l.], 2007. 53, 61

SHARMA, P. *WXD3590 Precision Wirewound Potentiometer.* [S.l.], 2014. 47

SHEPPARD, P. Swing Up Control of the Pendubot. 1998. 12, 17, 45, 50, 54, 55, 61

SOLDOVIERI, T. *Introduccion a La Mecanica de Lagrange y Hamilton.* [S.l.]: La Universidad del Zulia, 2013. 20

TIMMIS, H. *Practical Arduino Engineering.* [S.l.]: Apress, 2011. 25

UNIDIGITAL. *Folha de Dados do Conversor Tensão-frequênciaUD-C450.* [S.l.], 2013. 37, 59

UPTON, E.; HALFACREE, G. *Raspberry Pi® User Guide.* [S.l.]: Jonh Wiley & Sons Ltd, 2012. 39, 40

# APÊNDICE A – Algoritmos (scripts m file) utilizados.

Alguns algoritmos foram utilizados para realizar os cálculos ao longo do Trabalho bem como gerar os gráficos. Tais algoritmos são listados nas próximas páginas.Alguns possuem continuação (parte 1, parte 2, etc.), entrentanto formam ao todo um mesmo código.

**Algoritmo A.1** Algoritmo para cálculo das matrizes A e B no espaço de estados (parte 1).

```
%%Algoritmo para cálculo da matriz A - Pendubot de Gabay,2015
%
clear, clc
%
%declaração das variáveis simbólicas e matrizes
%
syms q1 q2 q1p q2p teta1 teta2 teta3 teta4 teta5 tau1
Q=[q1 ;q2]
Qp=[q1p ;q2p]
D_q=[teta1+teta2+2*teta3*cos(q2) teta2+teta3*cos(q2);
teta2+teta3*cos(q2) teta2]
C_q_qp=[-teta3*sin(q2)*q2p -teta3*sin(q2)*(q1p+q2p);
-teta3*sin(q2)*q1p 0]
G_q=[teta4*9.81*cos(q1)+teta5*9.81*cos(q1+q2);
teta5*9.81*cos(q1+q2)]
Tau=[tau1;0];
%
%Modelo matemático
%
Qpp=(inv(D_q))*(Tau-(C_q_qp*Qp)-G_q)
%
%Declaração das funções simbólicas e derivadas
%
f1=symfun(Qpp(1), [q1, q2, q1p, q2p, teta1, teta2,
teta3, teta4, teta5, tau1])
df1=diff(f1,q1)
f2=symfun(Qpp(1), [q1, q2, q1p, q2p, teta1, teta2,
teta3, teta4, teta5, tau1])
df2=diff(f2,q2)
f3=symfun(Qpp(2), [q1, q2, q1p, q2p, teta1, teta2,
teta3, teta4, teta5, tau1])
df3=diff(f3,q1)
f4=symfun(Qpp(2), [q1, q2, q1p, q2p, teta1, teta2,
teta3, teta4, teta5, tau1])
df4=diff(f4,q2)
```

**Algoritmo A.2** Algoritmo para cálculo das matrizes A e B no espaço de estados (parte 2).

```
%
%Declaração das variáveis simbólicas de especificação
%do Pendubot
%
syms m1 m2 cm1 cm2 l1 l2 I1 I2
Teta1=symfun((m1*cm1^2)+(m2*l1^2)+I1, [m1, cm1, m2, l1, I1])
Teta2=symfun((m2*cm2^2)+I2, [m2, cm2, I2])
Teta3=symfun(m2*l1*cm2, [m2, l1, cm2])
Teta4=symfun((m1*cm1)+(m2*l1), [m1, cm1, m2, l1])
Teta5=symfun(m2*cm2, [m2, cm2])
%
%Atribuição das especificação do Pendubot
%
m1=0.49;                %massa do elo 1
m2=0.76;                %massa do elo 2
l1=0.3;                 %comprimento do elo 1
l2=0.45;                %comprimento do elo 2
cm1=0.15;               %centro de massa do elo 1
cm2=0.227;              %centro de massa do elo 2
I1=(1/12)*m1*l1^2;      %momento de inércia elo 1
I2=(1/12)*m2*l2^2;      %momento de inércia elo 2
%
tau1=10;                %torque na junta 1
q1=pi/2;                %ângulo da junta 1
q2=0;                   %ângulo da junta 2
q1p=0;                  %dq1/dt
q2p=0;                  %dq2/dt
%
%aplicação dos pontos atribuídos às
%parametrizações
%
teta1=Teta1(m1, cm1, m2, l1, I1)
teta2=Teta2(m2, cm2, I2)
teta3=Teta3(m2, l1, cm2)
teta4=Teta4(m1, cm1, m2, l1)
teta5=Teta5(m2, cm2)
%
%aplicação dos pontos atribuídos à função
%derivada
%
A21=df1(q1, q2, q1p, q2p, teta1, teta2, teta3,
teta4, teta5, tau1);
A23=df2(q1, q2, q1p, q2p, teta1, teta2, teta3,
teta4, teta5, tau1);
A41=df3(q1, q2, q1p, q2p, teta1, teta2, teta3,
```

**Algoritmo A.3** Algoritmo para cálculo das matrizes A e B no espaço de estados (parte 3).

```
teta4, teta5, tau1);
A43=df4(q1, q2, q1p, q2p, teta1, teta2, teta3,
teta4, teta5, tau1);
%
%Matriz A
%
display('Matriz A')
A=[0 1 0 0; A21 0 A23 0; 0 0 0 1; A41 0 A43 0]
%_____________________________________________________________
%
%%Algoritmo para cálculo da matriz B - Pendubot de Gabay,2015
%
syms q1 q2 q1p q2p teta1 teta2 teta3 teta4 teta5
Q=[q1 ;q2];
D_q=[teta1+teta2+2*teta3*cos(q2) teta2+teta3*cos(q2);
teta2+teta3*cos(q2) teta2];
%
%Modelo matemático
%
G=(inv(D_q)) %como u=Tau para este caso,
%a derivada é a própria (D_q)^-1
%
%Declaração das funções simbólicas
%
g1=symfun(G(1), [q2,teta1, teta2, teta3, teta4, teta5]);
g2=symfun(G(2), [q2,teta1, teta2, teta3, teta4, teta5]);
%
%
%Declaração das variáveis simbólicas de especificação
%do Pendubot
%
syms m1 m2 cm1 cm2 l1 l2 I1 I2
Teta1=symfun((m1*cm1^2)+(m2*l1^2)+I1, [m1, cm1, m2, l1, I1])
Teta2=symfun((m2*cm2^2)+I2, [m2, cm2, I2])
Teta3=symfun(m2*l1*cm2, [m2, l1, cm2])
Teta4=symfun((m1*cm1)+(m2*l1), [m1, cm1, m2, l1])
Teta5=symfun(m2*cm2, [m2, cm2])
%
%Atribuição das especificação do Pendubot
%
m1=0.49;                %massa do elo 1
m2=0.76;                %massa do elo 2
l1=0.3;                 %comprimento do elo 1
l2=0.45;                %comprimento do elo 2
cm1=0.15;               %centro de massa do elo 1
```

**Algoritmo A.4** Algoritmo para cálculo das matrizes A e B no espaço de estados (parte 4).

```
cm2=0.227;              %centro de massa do elo 2
I1=(1/12)*m1*l1^2;      %momento de inércia elo 1
I2=(1/12)*m2*l2^2;      %momento de inércia elo 2
%
tau1=10;                %torque na junta 1
q1=pi/2;                %ângulo da junta 1
q2=0;                   %ângulo da junta 2
q1p=0;                  %dq1/dt
q2p=0;                  %dq2/dt
%
%aplicação dos pontos atribuídos às parametrizações
% teta1=Teta1(m1, cm1, m2, l1, I1)
teta2=Teta2(m2, cm2, I2)
teta3=Teta3(m2, l1, cm2)
teta4=Teta4(m1, cm1, m2, l1)
teta5=Teta5(m2, cm2)
%
%aplicação dos pontos atribuídos
%
b1=g1(q2,teta1, teta2, teta3, teta4, teta5);
b2=g2(q2,teta1, teta2, teta3, teta4, teta5);
%
%Matriz B
%
display('Matriz B')
B=[0; b1; 0; b2]
```

**Algoritmo A.5** Algoritmo para cálculo do controlador de realimentação de estados completo.

```
%as matrizes A e B consideradas aqui, foram as
%matrizes calculadas por meio do algoritmo anterior.
A=[0 1 0 0;40.3121 0 -53.3635 0;
0 0 0 1; -47.8904 0 139.0445 0]
B=[0;31.6716;0;-63.2025]
Q=[1 0 0 0; 0 0 0 0; 0 0 1 0; 0 0 0 0];
R=[1]
Ts=0.005;
[Kd,S,e] = lqrd(A,B,Q,R,Ts)
```

**Algoritmo A.6** Algoritmo para gerar a Figura 39 deste trabalho.

```
%Colar este script na Command Window
%depois de simular o Pendubot de
%Gabay no Simulink
plot(simout,'r')
hold on
plot(simout1)
grid
axis([0 3 -0.15 0.15]);
title('Simulação modo balanço - Rejeição a distúrbio')
xlabel('tempo (seg)')
ylabel('Resposta (rad)')
leg1 = legend('Distúrbio','Pendubot');
```

**Algoritmo A.7** Algoritmo (função) auxiliar para efetuar os cálculos do *swing* do *Pendubot*. (Parte 1).

```
%este algoritmo não deve ser compilado, mas
%deve estar na mesma pasta em que estiverem
%os outros dois algoritmos descritos a seguir.
%
function xdot = r2dof(t,x,ths,spec,Kpid)
xdot=zeros(8,1);
%
% set-points ou referências
%
th1s=ths(1);
th2s=ths(2);
%
% Especificações do Robô
%
M1=spec(3);
M2=spec(4);
CM1=spec(5);
CM2=spec(6);
L1=spec(1);
L2=spec(2);
g=9.8;
%
% Matriz de Inércia
%
b11=(M1*CM1^2)+M2*(L1^2+CM2^2+2*L1*CM2*cos(x(4)))+
((1/12)*M1*L1^2)+((1/12)*M2*L2^2);
b12=M2*(CM2^2+L1*CM2*cos(x(4)))+((1/12)*M2*L2^2);
b21=M2*(CM2^2+L1*CM2*cos(x(4)))+((1/12)*M2*L2^2);
b22=M2*CM2^2+((1/12)*M2*L2^2);
Bq=[b11 b12;b21 b22];
qponto=[x(5);x(6)];
%
% Matriz C (Coriólis e Centrípeta)
%
c11=(-M2*L1*CM2*sin(x(4)))*(x(6));
c12=(-M2*L1*CM2*sin(x(4)))*(x(5)+x(6));
c21=-(-M2*L1*CM2*sin(x(4)))*x(5);
c22=0;
Cq=[c11 c12;c21 c22];
%
%Matriz de Gravidade
%
g1=((M1*CM1+M2*L1)*g*sin(x(3)))+(M2*CM2*g*sin(x(3)+x(4)));
g2=M2*g*CM2*sin(x(3)+x(4));
```

**Algoritmo A.8** Algoritmo (função) auxiliar para efetuar os cálculos do *swing* do *Pendubot*. (Parte 2).

```
Gq=[g1;g2];
%
% Controle PID % Parâmetros do PID para theta 1
%
Kp1=Kpid(1);
Kd1=Kpid(2);
Ki1=Kpid(3);
% Parâmetros do PID para theta 2
Kp2=Kpid(4);    %%    A segunda junta
Kd2=Kpid(5);    %%       não possui
Ki2=Kpid(6);    %%         atuação
%
%Entrada de controle dissociada
%
f1=Kp1*(th1s-x(3))-Kd1*x(5)+Ki1*(x(1));
f2=0;
Ftorques=[f1;0];
F=Bq*Ftorques;
%entrada real para o sistema
%Estados do Sistema
xdot(1)=(th1s-x(3));   %estado fictício de integração de theta1
xdot(2)=(th2s-x(4));   %estado fictício de integração de theta2
xdot(3)=x(5);          %theta1-ponto
xdot(4)=x(6);          %theta2-ponto
q2dot=inv(Bq)*((-Cq*(qponto))-Gq+F);
xdot(5)=q2dot(1);      %theta1-2pontos
xdot(6)=q2dot(2);      %theta1-2potnos
%função de entrada de controle para saída do algoritmo
xdot(7)=F(1);
xdot(8)=F(2);
end
```

**Algoritmo A.9** Algoritmo de especificações e controle do *Pendubot* para *swing* (parte 1)

```
close all
clear all
clc
%
%Inicialização
%
th_int=[-pi 0]; %posição (0;-0,75)  % posição inicial
%(ângulo entre vertical e juntas)
ths=[0 0];      %posição (0;0,75)   % set-points
x0=[0 0 th_int 0 0 0 0];  %vetor de estados iniciais
Ts=[0 3];                 %tempo
%
% Especificações do Robô
%
L1=0.3;      %comprimento do elo 1
L2=0.45;     %comprimento do elo 2
M1=0.49;     %massa do elo 1 - considerando 8mm de espessura e
M2=0.76;     %massa do elo 2 - densidade do aluminio de 2697 kg/m3
CM1=0.15;    %Centro de massa do elo 1
CM2=0.227;   %Centro de massa do elo 2
spec=[L1 L2 M1 M2 CM1 CM2];
%
%Parâmetros do controlador PD (ou PID)
% Parâmetros do PID para theta 1
Kp1=98.7;      %resposta satisfatória kp1=98.7 e kd1=24.1
Kd1=24.1;      %resposta interessante oscilatória 51.7 e 19.1
Ki1=0;
%
%Parâmetros do PID para theta 2
Kp2=0;          %% a segunda junta
Kd2=0;          %%    não possui
Ki2=0;          %%      atuação
Kpid=[Kp1 Kd1 Ki1 Kp2 Kd2 Ki2];
%
%Resolução por meio de ODE
%
[T,X] = ode45(@(t,x) r2dof(t,x,ths,spec,Kpid),Ts,x0);
%
% Saída
%
th1=X(:,3);   %Forma de onda de theta1
th2=X(:,4);   %Forma de onda de theta2
%
%Cálculo das entradas de torque
%dos estados 7 e 8 da ODE
```

**Algoritmo A.10** Algoritmo de especificações e controle do *Pendubot* para *swing* (parte 2).

```
F1=diff(X(:,7))./diff(T);
F2=diff(X(:,8))./diff(T);
tt=0:(T(end)/(length(F1)-1)):T(end);
%posições x e y
x1=L1.*sin(th1);                          % X1
y1=L1.*cos(th1);                          % Y1
x2=L1.*sin(th1)+L2.*sin(th1+th2);         % X2
y2=L1.*cos(th1)+L2.*cos(th1+th2);         % Y2
%
%plotagem das posições de theta 1 e 2
%
hold on
grid
plot(T,th1-ths(1),'r')
plot(T,th2-ths(2),'b')
title('Simulação - Equilíbrio instável: ponto
superior - Theta1 e Theta2')
ylabel('Posições das Juntas (rad)')
xlabel('tempo (seg)')
leg1 = legend('q1','q2');
hold off
%
%plotagem dos torques de theta 1 e 2
%
figure,
hold on,
grid
plot(tt,F1, 'r')
plot(tt,F2, 'b')
title('Simulação - Equilíbrio instável: ponto
superior - Theta1 e Theta2')
ylabel('Torques das Juntas - N.m')
xlabel('tempo (seg)')
leg2 = legend('Tq1','Tq2');
```

**Algoritmo A.11** Algoritmo para gerar a animação exibida na Figura 37 deste Trabalho.

```
%ConFiguração de velocidade do frame
d=2;
j=1:d:length(T);
%
%gerador de imagens em 2D
%
figure
for i=1:length(j)-1
hold off
plot([x1(j(i)) x2(j(i))],[y1(j(i)) y2(j(i))],
'o',[0 x1(j(i))],[0 y1(j(i))],'k',
[x1(j(i)) x2(j(i))],[y1(j(i)) y2(j(i))],'k')
title('Simulação:Equilíbrio Instável Superior
de um Pendubot')
xlabel('x'),
ylabel('y')
axis([-0.8 0.8 -0.8 0.8]);
%grid
hold
on MM(i)=getframe(gcf);
end drawnow;
```

# APÊNDICE B – Teste de atuação da primeira junta - Fotos do sistema real.

Apresenta-se neste Apêndice algumas fotos que ilustram o teste de atuação da primeira junta do *Pendubot*, citada no item 4.2.5.2, onde está ilustrado o algoritmo de teste que controla o sistema exibido a seguir.

Figura 40 – Vista superior do teste de atuação da primeira junta.

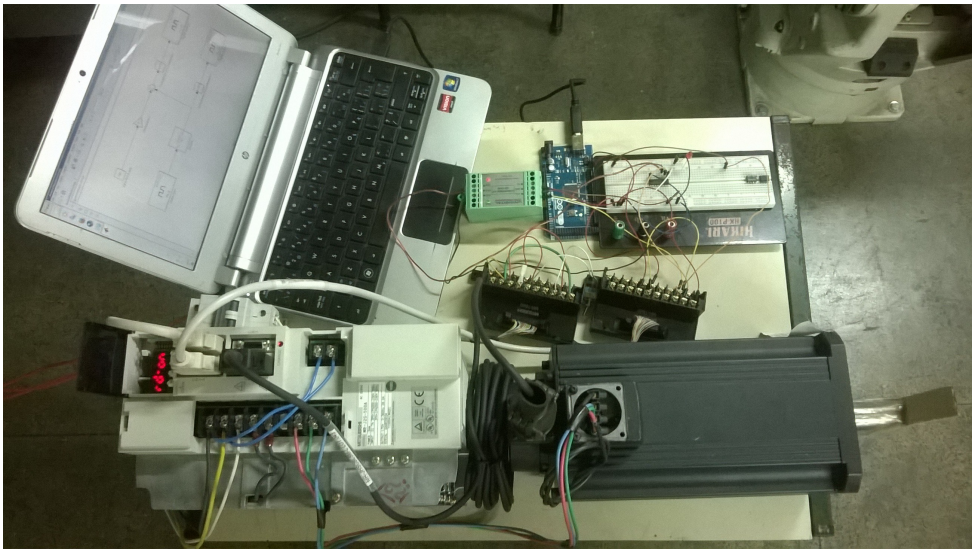


Figura 41 – Vista frontal do teste de atuação da primeira junta.