UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

TRABALHO DE CONCLUSÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Desenvolvimento de uma plataforma de testes de controladores para os dois primeiros graus de liberdade de um robô SCARA

Autor: Rafael Marquetto Vargas

Orientador: Prof. Dr. Mário Roland Sobczyk Sobrinho

Porto Alegre, 19 de junho de 2015

Sumário

Agradecimentos

Agradeço ao Dr. Carlos Sarmanho Junior por ter me apresentado e ensinado sobre o mundo da robótica durante os vários semestres em que trabalhamos juntos. Obrigado por ter incentivado o meu interesse nessa área respondendo pacientemente as minhas dúvidas mesmo quando meu conhecimento era muito superficial.

Agradeço ao Prof. Dr. Mário Roland Sobczyk Sobrinho pela orientação durante o TCC mas agradeço principalmente pela parceria e a amizade durante os semestres em que tivemos contato durante a minha graduação.

Agradeço ao Prof. Dr. Rafael Comparsi Laranja pelo apoio moral e financeiro durante o início do projeto deste robô há dois anos atrás e por sempre estimular os alunos a pensarem fora da caixa fornecendo orientação e materiais.

Agradeço ao LAMECC pelo fornecimento do espaço e ferramentas para o desenvolvimento do projeto e especialmente a todos os integrantes do laboratório que tive a oportunidade de trabalhar junto durante minha graduação. Grande parte do conhecimento necessário para a execução deste trabalho e para a conclusão da minha graduação foram obtidos através da experiência que vivi neste laboratório.

Resumo

Este trabalho consiste no desenvolvimento de uma plataforma de testes para os dois primeiros graus de liberdade de um manipulador SCARA. A plataforma tem o propósito de ser uma ferramenta de aprendizado para os alunos do curso de Controle e Automação. Através dela os alunos poderão aprender sobre robótica testando diferentes tipos de algoritmos de controle. O hardware necessário para implementação de diferentes tipos de controladores foi desenvolvido e um controlador foi testado. Foi executada a modelagem completa do sistema, desde motores até a dinâmica dos manipuladores. A mesma foi devidamente validada de forma experimental, de modo que o comportamento dos algoritmos pode ser simulado e avaliado antes da implementação.

Abstract

This report presents the development of a test platform for the first two degrees of freedom of a SCARA manipulator. The platform has the purpose of being a learning tool for students of Control and Automation engineering. Through this platform, students will be able to learn about robotics by testing different types of control algorithms. The hardware required to implement different types of controllers was developed and a controller was tested. The complete modeling of the system, from actuators to manipulator dynamics was done and properly validated experimentally. Through this model, the behavior of the developed algorithms can be tested before their implementation in the real system.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

R_a	Resistência de armadura (Ω)
L_a	Indutância de armadura (H)
K_t	Constante de Torque (N.m/A)
J_m	Inércia do motor (kg.m^2)
B_m	Atrito viscoso do motor (kg.m^2/s)
K_e	Constante de força contra-eletromotriz (V.s/rad)
I_a	Corrente de armadura (A)
V_a	Tensão de armadura (V)
E_a	Tensão contra-eletromotriz (V)
θ	Posição angular (rad)
$\dot{\theta}$	Velocidade angular (rad/s)
$\ddot{\theta}$	Aceleração angular (rad/s^2)
T_m	Torque do motor (N.m)
T_c	Torque da carga (N.m)
m_i	Massa do elo i (kg)
l_i	Comprimento do elo i (m)
B_{li}	Coeficiente de atrito viscoso do elo i (kg.m^2/s)
θ_i	Posição do elo i (rad)
r_i	Vetor do centro de massa do elo i em relação ao sistema de coordenadas generalizadas do elo (m)
$\overline{I_i}$	Tensor de inércia do elo i (kg.m^2)
$\overline{D}(\boldsymbol{q})$	Matriz de inércia do manipulador (kg.m^2)
$\overline{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$	Matriz de forças de Coriolis e centrípetas do manipulador (kg.m^2/s)
$\overline{B}$	Matriz de atrito viscoso do manipulador (kg.m^2/s)
$\boldsymbol{q}$	Vetor de coordenadas generalizadas do manipulador (rad)
$\boldsymbol{\tau}$	Vetor de torques do manipulador (N.m)
(x, y)	Coordenadas do efetuador no plano cartesiano (m)
P	Ganho proporcional (V/°)
D	Ganho Derivativo (V.s/°)
N	Coeficiente de filtro para altas frequências (s^-1)

Lista de Abreviaturas e Siglas

SCARA	Selective Compliance Assembly Robot Arm
UFRGS	Universidade Federal do Rio Grande do Sul
CT	*Computed Torque control*
DFA	*Design For Assembly*
PID	Proporcional-Integral-Derivativo
DAQ	*Data Acquisition System*
CC	Corrente Contínua
BJT	*Bipolar Junction Transistor*
PWM	*Pulse Width Modulation*
RPM	Rotações Por Minuto
PD	Proporcional-Derivativo
A/D	Analógico-Digital

1 Introdução

No presente trabalho foi desenvolvida uma plataforma de testes de controladores para os dois primeiros graus de liberdade de um robô SCARA. Essa plataforma começou a ser desenvolvida no semestre 2013/1 para a disciplina de Projeto II no Departamento de Engenharia Mecânica da UFRGS. Até o início desde trabalho a parte mecânica dos primeiros dois graus de liberdade estava finalizada e os motores e sensores para os dois primeiros graus de liberdade estavam devidamente posicionados e fixados.

1.1 Motivação

Durante o curso de Engenharia de Controle e Automação na UFRGS, os alunos são expostos a dois tipos de conhecimentos relativos a sistemas robóticos. Por um lado, diversas disciplinas abordam os aspectos teóricos do controle de sistemas dinâmicos em geral, onde manipuladores robóticos são vistos como um caso particular. No extremo oposto, há somente uma disciplina obrigatória que propicia a oportunidade de lidar com robôs industriais, onde os mesmos são tratados como "caixas-pretas" a serem programadas para executar tarefas variadas de posicionamento, tais como manipulação de ferramentas e paletização. Desta forma, é difícil relacionar os conteúdos da área de controle com o funcionamento de um robô real, pois os aparelhos comerciais disponíveis no Laboratório de Robótica não permitem acesso aos detalhes relativos à implementação dos seus algoritmos de controle. Assim, a principal motivação deste trabalho é a necessidade de desenvolver uma "ponte" entre estes dois aspectos extremos do funcionamento de um robô, permitindo que os estudantes do curso tenham a chance de desenvolver diferentes abordagens de controle para manipuladores robóticos e testá-las em um sistema real, livre das restrições associadas a um equipamento comercial, de modo a "visualizar" o efeito de suas escolhas de projeto sobre o desempenho do robô.

1.2 Objetivos

O objetivo geral do trabalho é desenvolver uma plataforma robusta e de fácil programação que possa ser utilizada pelos alunos da UFRGS para testes de controladores para os dois primeiros graus de liberdade de um robô SCARA. Para isso, é necessária a solução de diversos problemas de hardware e a implementação de algumas ferramentas de software que auxiliarão na tarefa de testar os controladores desenvolvidos pelos alunos. A seguir pode ser observada uma lista de objetivos específicos que são necessários para a concretização do objetivo geral:

1.2.1 Objetivo 1: Desenvolvimento de um hardware de acionamento e aquisição de dados para o manipulador.

Pesquisa das soluções de hardware encontradas na literatura, comparação com o caso de estudo e determinação das técnicas e componentes a serem utilizados no projeto. Os principais elementos desse hardware são os sensores das juntas e os *drivers* de potência dos motores.

A solução deve ser robusta, confiável e de simples utilização de forma que o usuário não tenha que se preocupar com o hardware, apenas com a programação do controlador.

1.2.2 Objetivo 2: Desenvolvimento de uma interface de comunicação flexível e intuitiva entre o hardware da plataforma e o software de controle.

O usuário deve ser capaz de ter acesso fácil aos dados dos sensores e às variáveis de controle do robô. A comunicação idealmente deve ser de fácil entendimento de forma que se futuramente sejam necessários modificações no hardware da plataforma, seja possível adaptar a comunicação de forma simples.

1.2.3 Objetivo 3: Desenvolver um modelo e dinâmico do manipulador para criar uma ferramenta de simulação para os controladores.

Através da modelagem mecânica e elétrica do manipulador, é possível obter um modelo dinâmico do comportamento do robô. Este modelo foi utilizado para desenvolver um ambiente de simulação em que os controladores desenvolvidos podem ser testados primeiramente em simulação antes de serem implementados no sistema real.

1.3 Organização do trabalho

Este trabalho é organizado da seguinte forma: durante a revisão bibliográfica serão apresentados os conteúdos necessários para o desenvolvimento do modelo do manipulador e um estudo sobre o estado da arte de manipuladores robóticos do tipo SCARA e sobre os possíveis métodos de comunicação a serem utilizados na plataforma.

Na seção de descrição do sistema serão apresentados os métodos e ferramentas escolhidos para realizar as tarefas necessárias para a realização do projeto assim como as justificativas das escolhas do projeto.

Finalmente, durante os resultados e discussões, o modelo proposto será utilizado para prever o comportamento do robô durante uma tarefa de seguimento de trajetória. A fim de verificar a validade do modelo em questão, os resultados previstos nas simulações serão comparados com os de um experimento equivalente, executado pelo robô propriamente dito.

2 Revisão Bibliográfica

2.1 Manipulador robótico SCARA

A estrutura mecânica de um manipulador robótico consiste em uma sequência de corpos rígidos chamados elos, interconectados por articulações chamadas juntas. Essas articulações entre dois elos são criadas por meio de dois tipos de juntas, prismática ou rotacional. A mobilidade de um manipulador robótico é gerada pela presença das juntas, cada junta proporciona um grau de liberdade para o manipulador (Siciliano, et al., 2009).

O manipulador robótico desenvolvido neste projeto é do tipo SCARA e pode ser observado na Figura 2-1. O robô SCARA (selective compliance assembly robot arm) foi inventado em 1978 por Hiroshi Makino na Universidade de Yamanashi, Japão. O manipulador de 4 graus de liberdade e de baixo custo foi a ferramenta perfeita para a realização de montagens de peças pequenas. Sistemas de montagem flexível baseados no robô SCARA em conjunto com o design de produtos compatíveis com esse sistema de montagem contribuíram para a produção em massa de eletrônicos e bens de consumo (Siciliano, et al., 2008). O espaço de trabalho reduzido comparado com outras configurações tradicionais que pode ser observado na Figura 2-1, não prejudica o propósito principal do manipulador que é montagem de peças uma vez que as mesmas usualmente podem ser posicionadas dentro desse espaço de trabalho.

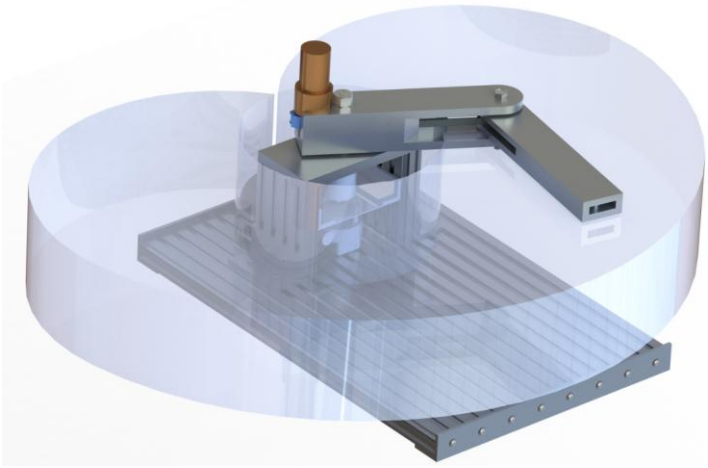


Figura 2-1: Manipulador SCARA e seu espaço de trabalho.

A geometria do manipulador SCARA é composta por duas juntas rotacionais e uma prismática de forma que todos os eixos de movimento sejam paralelos (Siciliano, et al., 2009). Pode-se também encontrar uma quarta junta, rotacional, ao final do manipulador que garante a orientação do efetuador. Neste trabalho foram utilizadas apenas as duas primeiras juntas rotacionais, gerando um manipulador planar de dois graus de liberdade.

2.2 Modelagem Cinemática do manipulador

2.2.1 Cinemática inversa

Para o planejamento de trajetória do manipulador é necessário o conhecimento da cinemática inversa do manipulador. Após decididas as coordenadas no espaço cartesiano desejadas para a trajetória, usa-se a cinemática inversa para calcular estas mesmas trajetórias no espaço de coordenadas generalizadas do manipulador. Desta forma é

possível, a partir de uma trajetória no espaço de coordenadas cartesianas, gerar duas trajetórias no espaço de coordenadas generalizadas, uma para cada junta do robô.

As equações de cinemática inversa apresentadas foram baseadas em Spong, et al., 2004. Para o desenvolvimento destas equações foi utilizada uma projeção dos elos no plano xy, desta forma pode-se obter uma relação geométrica entre as variáveis de junta e a posição dos elos no plano cartesiano. Os parâmetros das equações referentes aos dois graus de liberdade do manipulador são apresentados na Figura 2-2.

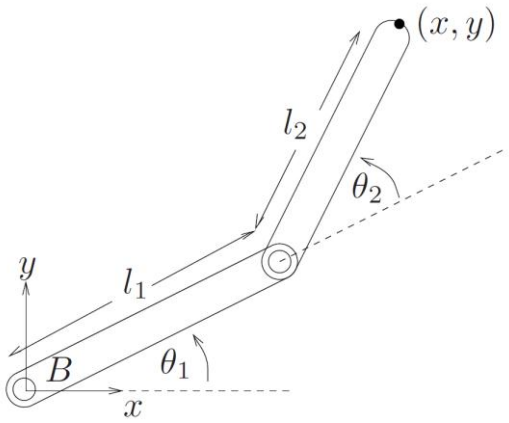


Figura 2-2: Manipulador planar de dois graus de liberdade e seus parâmetros (Murray, et al., 1994).

As coordenadas x e y são as posições desejadas no espaço cartesiano enquanto θ_1 e θ_2 são as coordenadas generalizadas necessárias para que o manipulador atinja essa posição. Os parâmetros l_1 e l_2 são os comprimentos dos elos têm valor 0,213 metros e 0,27 metros respectivamente. A partir da Figura 2-2, geometricamente podem ser determinadas as equações (1), (2) e (3) que definem a cinemática inversa do manipulador:

$$\theta_2 = \tan^{-1}\left(\cos\theta_2 \ , \pm\sqrt{1-\cos^2\theta_2}\right) \tag{1}$$

onde

$$\cos\theta_2 \ = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2} \tag{2}$$

$$\theta_1 = atan2(y, x) + \text{atan2}\big((l_2 \operatorname{sen}\theta_2), (l_1 + l_2\cos\theta_2\ )\big)\,. \tag{3}$$

2.2.2 Cinemática direta

A cinemática direta do manipulador foi utilizada durante os experimentos para que a partir dos valores das coordenadas generalizadas medidas pelos sensores, fosse possível calcular a posição (x, y) no plano de coordenadas cartesianas. Para calcular a cinemática direta deste manipulador planar de dois graus de liberdade utilizou-se a convenção de Denavit-Hartenberg (Hartenberg, et al., 1964) e a partir dessa convenção, a Tabela 2-1 foi obtida.

Tabela 2-1: Parâmetros de Denavit-Hartenberg.

$Link$	a_i	α_i	d_i	θ_i
1	l_1	0	0	θ_1^*
2	l_2	0	0	θ_2^*

Os parâmetros de Denavit-Hartenberg são utilizados para gerar a matriz homogênea de transformação das coordenadas do efetuador em relação ao *frame* da base. Segundo (Spong, et al., 2004), esta configuração de parâmetros gera a equação (4):

$$T_2^0 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & a_1\cos\theta_1 + a_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & a_1\sin\theta_1 + a_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (4)$$

As primeiras 3 linhas e 3 colunas da matriz representam a orientação do efetuador enquanto os 3 primeiros elementos da quarta coluna representam a posição do efetuador no espaço cartesiano. Neste caso não existe efetuador, então essas coordenadas são referentes ao ponto em que o elo 2 termina.

2.3 Modelagem dinâmica do manipulador

Neste trabalho, o modelo dinâmico foi baseado no desenvolvido em Murray, et al., 1994, para um manipulador planar de 2 graus de liberdade. A diferença é que foi adicionada uma parcela de torque referente ao atrito viscoso das juntas. Os parâmetros geométricos do modelo podem ser observados na Figura 2-3.

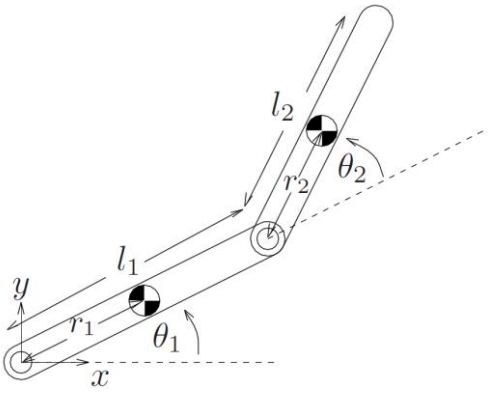


Figura 2-3: Manipulador planar de 2 DOF (Murray, et al., 1994).

Cada elo possui massa m_i, comprimento l_i, centro de massa $r_i = [r_{xi}, r_{yi}, r_{zi}]$, atrito viscoso B_{li}, e um tensor de inércia

$$\overline{I_i} = \begin{bmatrix} I_{xxi} & I_{xyi} & I_{xzi} \\ I_{yxi} & I_{yyi} & I_{yzi} \\ I_{zxi} & I_{zyi} & I_{zzi} \end{bmatrix}$$

relativo a um sistema de coordenadas posicionado no centro de massa do elo i e alinhado com o sistema de coordenadas do elo. Os parâmetros l_1 e l_2 são conhecidos e têm valor

0,213 metros e 0,27 metros respectivamente. Conhecendo o restante dos parâmetros citados anteriormente, tem-se o modelo dinâmico para este manipulador sendo:

$$\overline{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \overline{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \overline{B}\dot{\boldsymbol{q}} = \boldsymbol{\tau}\,. \tag{5}$$

A matriz $D(\boldsymbol{q})$ é a matriz de inércia do manipulador e tem dimensão 2x2, $C(\boldsymbol{q},\dot{\boldsymbol{q}})$ é a matriz de forças de Coriolis e centrípetas e tem dimensão 2x2, B é a matriz de atrito viscoso e tem dimensão 2x2, $\boldsymbol{q}$ é o vetor de coordenadas generalizadas com dimensão 2x1 e $\boldsymbol{\tau}$ é o vetor de torques relativos a cada grau de liberdade com dimensão 2x1. Baseado em (Murray, et al., 1994), tem-se que a dinâmica deste manipulador é regida pelas equações (6), (7), (8) e (9):

$$\begin{bmatrix} \alpha + 2\beta\cos\theta_2 & \delta + \beta\cos\theta_2 \\ \delta + \beta\cos\theta_2 & \delta \end{bmatrix}\begin{bmatrix}\ddot{\theta}_1 \\ \ddot{\theta}_2\end{bmatrix} + \begin{bmatrix} -\beta\,\mathrm{sen}\,\theta_2\,\dot{\theta}_2 & -\beta\,\mathrm{sen}\,\theta_2\,(\dot{\theta}_1+\dot{\theta}_2) \\ \beta\,\mathrm{sen}\,\theta_2\,\dot{\theta}_1 & 0 \end{bmatrix}\begin{bmatrix}\dot{\theta}_1 \\ \dot{\theta}_2\end{bmatrix} + \begin{bmatrix} B_{l1} & 0 \\ 0 & B_{l2}\end{bmatrix}\begin{bmatrix}\dot{\theta}_1 \\ \dot{\theta}_2\end{bmatrix} = \begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix}, \tag{6}$$

onde

$$\alpha = I_{zz1} + I_{zz2} + m_1 r_1^2 + m_2(l_1^2 + r_2^2)\,, \tag{7}$$

$$\beta = m_2 l_1 r_2\,, \tag{8}$$

$$\delta = I_{zz2} + m_2 r_2^2\,. \tag{9}$$

Nota-se que este modelo não apresenta nenhuma componente gravitacional porque nenhumas das duas juntas avaliadas sofre ação direta da força gravitacional assim como os parâmetros deste modelo que são constantes. O modelo dinâmico completo de um manipulador SCARA de 4 graus de liberdade pode ser encontrado também em (Murray, et al., 1994).

2.4 Instrumentação e controle

Foi realizado um estudo sobre controle e instrumentação de manipuladores SCARA com o intuito de se decidir que tipos de sensores são adequados para a implementação e teste dos controladores mais utilizados. Para se criar uma plataforma de testes completa e flexível, ela deve ser capaz de disponibilizar os dados necessários para os diferentes tipos de testes que serão feitos no manipulador.

Um dos métodos mais simples em questão de instrumentação é apresentado em Moreno–Valenzuelaa, 2010, no qual é apresentado um tipo de controlador PI que utiliza apenas realimentação de velocidade dos dois primeiros graus de liberdade de um manipulador SCARA. Para a implementação desse tipo de controlador é suficiente o uso de um apenas um sensor de posição em cada junta. O mesmo vale para controladores PID com realimentação de posição ou velocidade, provavelmente a primeira escolha de um estudante ao usar uma plataforma de testes de controladores flexível.

Controle de posição é adequado para tarefas como transferência de materiais ou tarefas em que o manipulador não interage significativamente com o ambiente de

trabalho. Porém, tarefas como montagem, polimento ou rebarbamento que envolvem contato com o ambiente são usualmente melhor desempenhadas com sistemas de controle da força de interação com o mesmo. Em uma estrutura rígida como a de um robô, um pequeno erro de posição pode gerar forças de interação excessivas e com consequências desastrosas, como a quebra ou a deformação dos objetos manipulados ou do próprio manipulador (Spong, et al., 2004).

Para propósitos de controle da interação de forças entre o efetuador e o ambiente, usualmente a técnica que gera melhores resultados é a utilização de sensores de força-torque de 6 eixos posicionados no pulso do manipulador (Spong, et al., 2004). Em Puente, et al., 1991, um sensor com essa configuração é usado em um robô SCARA para a compensação de erros de repetitividade do robô ou de posicionamento de peças frágeis em sistemas de montagem. Quando ocorre um erro de posicionamento na montagem dos elementos, uma perturbação de força devido ao choque das peças é gerada e através da medição do sensor, o controlador é capaz de corrigir o posicionamento na direção correta.

Outro controlador com realimentação de torque foi apresentado em Morel, et al., 1999. Neste artigo um sensor de força/torque de seis eixos montado embaixo da base de um manipulador PUMA 550 é utilizado para compensar o atrito nas juntas. Através das medidas do sensor, é possível calcular os torques dinâmicos aplicados nos elos do manipulador, essas medidas não sofrem influência do atrito das juntas uma vez que as única forças transferidas para a base são referentes à inércia do manipulador e à gravidade. Os torques estimados nas juntas são usados em um controlador de torque que virtualmente elimina os efeitos de atrito e da gravidade, gerando grande precisão no controle de pequenos movimentos em baixas velocidades.

As estratégias apresentadas nos dois parágrafos anteriores utilizam sensores de força/torque como meio de compensar os torques gerados pela inércia das massas, pela força da gravidade ou por forças de contato. A utilização desse tipo de sensor pode ser a escolha mais intuitiva para controle de força/torque pois a grandeza medida é a própria a ser controlada, mas esta não é necessariamente a melhor escolha para todos os casos. Sensores de força/torque na maioria dos casos são compostos por extensômetros, estes dispositivos possuem desvantagens como difícil instalação e calibração, alta fragilidade, grande sensibilidade a ruídos e alto custo (Tsetserukou, et al., 2005).

Uma alternativa para o uso de sensores de torque em sistemas de acionamento elétrico é o uso de sensores de corrente que comparados com os anteriores se mostram mais baratos e de simples instalação e calibração. O uso de sensores de corrente para medição de torque é possível quando se conhece o modelo do motor utilizado, por exemplo, em motores CC o torque é diretamente proporcional à corrente de armadura do motor.

Esta estratégia é utilizada em Fateh, 2008, no qual o autor, através de uma técnica de *feedback linearization*, usa a medição da corrente para linearizar efeitos de torque no manipulador uma vez que a corrente do motor contém todos os efeitos dinâmicos do manipulador que são transferidos para o motor. Essa estratégia de controle utilizando apenas sensores de posição e corrente foi capaz de cancelar todos os efeitos dinâmicos e de carga no manipulador, inclusive perturbações de torque. Um aspecto interessante neste controlador é que ele não necessita de nenhum modelo dinâmico ou parâmetro do robô, somente o modelo dos motores, outro interessante fato é que o controle é feito junta a

junta de forma que os graus de liberdade não são acoplados entre si no laço de controle diferentemente da técnica de torque calculado, esse fato possibilita uma baixa quantidade de cálculos no loop de controle. Estes dois fatos são importantes pois permitem que o algoritmo seja rapidamente calculado, logo o sistema terá uma alta taxa de amostragem, facilitando o controle em tempo real (Fateh, 2008).

Finalmente, o último método estudado é utilizado em diversos estudos e se chama controle por torque calculado. Esta é uma técnica de compensação direta (*tool*) baseada no modelo de dinâmica do manipulador. Esse método calcula os torques necessários para que o robô execute uma trajetória desejada considerando todos os efeitos de inércia e da gravidade inclusos no modelo e faz o controlador seguir essa referência de torque. Controladores independentes são adicionados às juntas para compensar diferenças entre o modelo utilizado e o sistema real (Khosla, et al., 1987).

Para que seja possível o seguimento de uma referência de torque utilizando atuadores de natureza elétrica é necessário o uso de outro sistema de controle que aplique a tensão necessária no motor para a geração do torque necessário. A referência de torque gerada pode ser diretamente transformada em uma referência de corrente utilizando-se o modelo do motor. Em (Keiser, 2013) foi utilizado um controlador PID para fazer esse sistema de controle em que o controlador tem uma entrada de erro de corrente e gera um sinal de tensão para ser aplicado no motor e corrigir o erro de corrente.

2.5 Interface e programação

O software mais utilizado para simulações de sistemas dinâmicos durante o curso de engenharia de Controle e Automação é a plataforma MATLAB/Simulink devido à facilidade de flexibilidade de programação. No software MATLAB pode-se programar em código (*scripts*) ou em diagrama de blocos com o auxílio da ferramenta Simulink. Nele existem comandos/blocos que tornam simples as tarefas de multiplicar matrizes, operar funções de transferência ou gerar gráficos; ferramentas fundamentais para controle e robótica.

Além das ferramentas básicas do software citadas, no MATLAB pode-se usar módulos específicos chamados *toolkits* que possuem funções prontas otimizadas para tarefas específicas. Um *toolkit* de interesse para este projeto se chama *Robotics Toolbox*, criada pelo professor australiano Peter Corke e compartilhado gratuitamente na sua página na internet (http://www.petercorke.com/Robotics_Toolbox.html). Essa *toolbox* possui diversas funções para simulação de robôs móveis e para manipuladores robóticos. Manipuladores robóticos arbitrários podem ser criados e a *toolbox* possui funções para cálculo da cinemática direta, inversa, dinâmica do manipulador e planejamento de trajetória (Corke, 2011).

MATLAB é uma ferramenta eficiente para design e análise de sistemas de controle mas para aplicações em tempo real sendo parte de um sistema de automação o software se torna uma ferramenta mais complicada de se trabalhar. As razões para isso são por exemplo: limitadas opções de hardware compatível e complicada programação de interface gráfica, o que faz o software não ser uma ferramenta de automação muito útil (Tekin, 2010).

Uma ferramenta de automação que os alunos da Engenharia de Controle e Automação têm acesso é o software LabVIEW. O LabVIEW possui uma fácil programação de interface gráfica e uma simples e eficiente programação de aquisição de dados, funções interessantes para a execução deste projeto e que não são encontradas no MATLAB. Outra vantagem de usar LabVIEW é a grande variedade de hardware da National Instruments compatíveis com o software. Em consequência destas vantagens e desvantagens dos softwares, algumas vezes o LabVIEW é utilizado no lugar ou em conjunto com o MATLAB para o desenvolvimento de sistemas de automação (Tekin, 2010).

Uma das possíveis formas de se trabalhar com estes dois softwares ao mesmo tempo é utilizando um *toolkit* do LabVIEW chamado *Simulation Interface Toolkit*. O *Simulation Interface Toolkit* é capaz de integrar o LabVIEW com o MATLAB em tempo real de uma forma que nos permite desenvolver e testar sistemas de controle usando modelos desenvolvidos no ambiente de simulação do Simulink. O *Simulation Interface Toolkit* automaticamente gera o código de LabVIEW que comunica os dois softwares resultando em uma interface flexível e fácil de utilizar (Xuejun, et al., 2007).

Outra opção para a aquisição de dados com o MATLAB é a utilização da plataforma Arduino como DAQ (*Data Acquisition System*). Arduino é uma plataforma de hardware *opensource* que vem crescendo nos últimos anos e se estabelecendo como uma valiosa tecnologia para estudantes de engenharia em função de seu baixo custo, fácil programação e da grande quantidade de conhecimento disponibilizada na internet pela comunidade de usuários de Arduino.

A ferramenta que possibilita a interface simplificada entre a plataforma Arduino e o MATLAB é um hardware support package para Arduino disponibilizado pela Mathworks, produtora do MATLAB/Simulink. Esse package disponibiliza blocos no Simulink que possibilitam a leitura e escrita nas portas da placa Arduino, desta forma pode-se criar sistemas de controle ou automação através do Arduino programando em Simulink e utilizando as diversas funcionalidades fornecidas por esse programa.

Existem duas maneiras de rodar o programa com o Arduino support package. Uma delas é baseada em um servidor rodando dentro do Arduino que recebe comandos do MATLAB através da porta serial, e depois de executar esses comandos retorna uma resposta se solicitado. Isso permite a visualização dos dados adquiridos pelo Arduino de forma instantânea no Simulink. A outra maneira consiste em converter o modelo de Simulink em um código que roda diretamente no Arduino, desta forma, o microcontrolador pode ser desconectado do computador e continuar rodando a aplicação independentemente (Barber, et al., 2013).

3 Descrição do Sistema

Os elementos básicos necessários para o desenvolvimento da plataforma de testes são: o sistema mecânico, os atuadores, os *drivers* de potência, os sensores, um sistema de aquisição de dados e um sistema de controle. A forma em que estes elementos do sistema se conectam é apresentada na Figura 3-1. Ao início deste trabalho, tarefas como a construção do sistema mecânico, o acoplamento dos atuadores e o posicionamento dos sensores de posição já haviam sido concluídas. Ao fim deste trabalho, o usuário irá encontrar o sistema completo no qual caberá a ele apenas editar o sistema de controle, fazer os testes desejados e analisar os resultados.

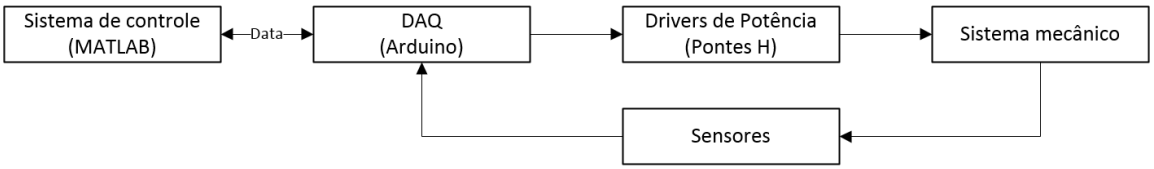


Figura 3-1: Sistema de controle desenvolvido.

3.1 Sistema mecânico, atuadores e sensores

O sistema mecânico consiste dos dois primeiros graus de liberdade de um manipulador do tipo SCARA. Cada grau de liberdade é atuado por um motor CC acoplado a uma redução, e possui um sensor de posição acoplado ao seu eixo.

Os sensores presentes atualmente no manipulador são potenciômetros multivolta de 10 voltas e 10 kΩ, esses sensores fornecem informação de posição da junta através de um sinal de tensão proporcional à posição da junta. Para este trabalho esses sensores serão mantidos pois são sensores de fácil utilização e o foco da aplicação não é precisão no momento.

Através do estudo feito na revisão bibliográfica, foi decidido que além dos sensores de posição seria necessária a utilização de sensores de corrente. Entre as técnicas estudadas, apenas as mais simples não utilizam algum tipo de realimentação de torque ou corrente. A utilização de sensores de corrente possibilitará a implementação de técnicas de controle mais avançadas na plataforma, uma vez que a realimentação de torque se torna possível através de um sensor de corrente quando se conhece o modelo do motor utilizado. A presença de sensores de corrente também foi importante para a modelagem dos motores, uma vez que a corrente em regime estacionário foi um parâmetro importante.

O sensor de corrente escolhido se chama Acs712 de 5A, ele será colocado em série com cada motor de forma a medir a corrente que alimenta o mesmo. A sensibilidade deste sensor é de 185 mV/A sendo que para uma corrente 0A a saída é de 2,5 V e o valor varia em torno deste número dependendo o sentido da corrente (Allegro MicroSystems, 2013). Estes sensores serão importantes também para a modelagem experimental dos motores.

3.2 *Drivers* de potência

Para acionar os motores é necessário um *driver* de potência que tem como entrada um sinal de baixa potência gerado pelo sistema de controle e como sinal de saída, um sinal de alta potência que alimenta o motor da junta. Para alimentação de motores CC o que se é

utilizado é um circuito de ponte H (Rohm Semiconductor, 2009). Esse circuito usualmente possui quatro entradas que quando acionadas em combinações corretas fazem o motor girar no sentido desejado. Entretanto, esta configuração permite uma combinação de acionamento dos transistores que gera um curto circuito na fonte. Neste projeto, o objetivo é criar uma plataforma de uso fácil para os alunos da universidade, então, foi feita uma adaptação no circuito de ponte H clássico para que não seja possível o acionamento errado em que ocorre curto circuito da fonte. Nesta ponte H adaptada, só existem duas entradas e nenhuma combinação delas gera curto circuito como pode ser observado no circuito da Figura 3-2. Isso é possível pelo fato de que diferentemente da topologia de ponte H comum, esta ponte H possui em cada entrada um transistor BJT polarizando 2 MOSFETs ao mesmo tempo. A lógica dos MOSFETs de canal P e de canal N são invertidas, logo, os MOSFETs nunca serão polarizados ao mesmo tempo por um mesmo sinal lógico impedindo um curto circuito na fonte.

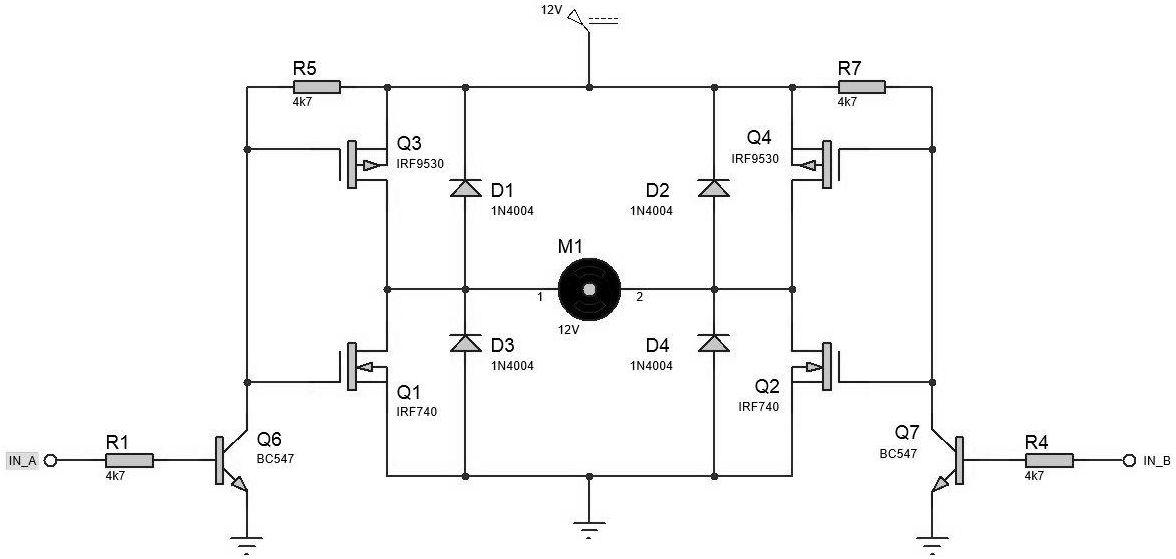


Figura 3-2: Ponte H desenvolvida para a plataforma.

Este circuito foi fabricado e testado e apresentou desempenhou a sua função corretamente. A tensão aplicada no motor foi de 11,8V para uma alimentação de 12V, enquanto o esperado de acordo com as simulações era de 10,8V.

3.3 Sistema de controle

Um algoritmo de controle foi desenvolvido através do MATLAB, incluindo aquisição de dados, sistema de controle e geração de trajetória. Esse algoritmo é capaz de executar todas as tarefas necessárias para fazer o manipulador seguir uma trajetória e é editável, de forma que os próximos usuários possam utilizá-lo para experimentos diferentes do realizado neste trabalho.

3.4 Hardware de aquisição de dados

Em função da facilidade de programação do LabVIEW e da disponibilidade de recursos da universidade, o hardware escolhido inicialmente para aquisição de dados foi a placa USB-6009 da National Instruments. Foi testada a aquisição de dados com essa placa e o envio destes dados para o Simulink através do *Simulation Interface Toolkit*, porém alguns problemas determinaram a ineficiência deste método para o projeto em questão.

A *Simulation Interface Toolkit* parou de ser atualizada em 2012, a última versão disponível é a do LabVIEW 2012 e é compatível apenas com MATLAB 2011b ou anteriores. Isto foi um problema pois a *Robotics Toolbox* utilizada não funcionou corretamente nesta versão do MATLAB assim como em algumas anteriores. Tentou-se utilizar a *toolbox* de robótica do LabVIEW mas ela se provou muito menos eficiente do que a do MATLAB devido ao fato de que a documentação de utilização era menos detalhada do que a do MATLAB e os programas exemplo disponíveis eram mais abstratos e de difícil compreensão. Outro motivo que desfavoreceu o uso da placa da National Instruments é o fato de que ela não possui saídas PWM, apenas saídas analógicas de baixa potência, para gerar o PWM que alimenta a ponte H teria que ser usado uma placa Arduino ou então um circuito externo de condicionamento.

Tendo em mente que a placa USB-6009 não seria uma solução para o problema, foi testada a aquisição de dados com uma placa Arduino comunicando diretamente com o Simulink. Embora o Arduino não consiga taxas de amostragem tão altas quanto as obtidas com placas USB-6009, as taxas obtidas foram consideradas compatíveis com o projeto. O *support package* do Arduino se provou uma ferramenta simples e intuitiva de se utilizar, muito mais fácil de usar do que a *Simulation Interface Toolkit*. Com esse *package* têm-se à disposição blocos que nos dão acesso direto aos pinos do Arduino, esses blocos podem ser observados na Figura 3-3.

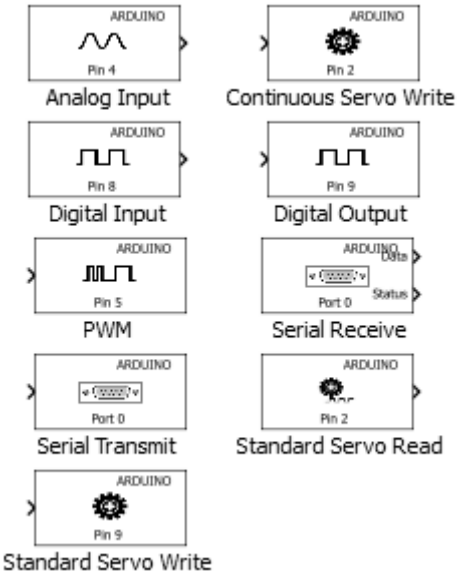


Figura 3-3: Blocos do *support package* de Arduino no Simulink.

A placa Arduino escolhida para o projeto é a Arduino Mega 2560 R3 pois ela é um dos modelos que suporta o modo online de simulação do *support package*, com placas mais simples como o Arduino Uno, não é possível mostrar os dados instantaneamente na tela do Simulink, apenas é possível gerar um código do programa no Simulink e gravá-lo no Arduino para executá-lo independentemente do computador. Através dessa placa foi feita a aquisição de dados dos sensores do robô através das entradas analógicas e foi feita a atuação sobre os motores através das saídas PWM. Desta forma, foi possível que todo o algoritmo de controle do robô tenha sido feito no Simulink.

4 Modelagem do Sistema

4.1 Modelagem dos motores

4.1.1 Modelagem matemática

Para este manipulador, foram utilizados dois motores CC de 12V. Esses motores são de ímã permanente, isso significa que o não possuem enrolamento de campo, apenas de armadura pois o campo é gerado pelos ímãs. O modelo matemático de um motor CC de ímã permanente controlado por tensão de armadura é descrito em (Spong, et al., 2004) pelas equações (10) a (13).

$$V_a(t) = R_a I_a(t) + L_a \dot{I}_a(t) + E_a(t) \tag{10}$$

$$E_a(t) = K_e \dot{\theta}(t) \tag{11}$$

$$T_m(t) = K_t I_a(t) \tag{12}$$

$$T_m(t) = T_c(t) + B_m \dot{\theta}(t) + J_m \ddot{\theta}(t) \tag{13}$$

O significado dos parâmetros do modelo são descritos na Tabela 4-1.

Tabela 4-1: Parâmetros do modelo do motor CC

Parâmetros	Significado
R_a	Resistência de armadura (Ω)
L_a	Indutância de armadura (H)
K_t	Constante de Torque (N.m/A)
J_m	Inércia do motor (kg.m^2)
B_m	Atrito viscoso do motor (kg.m^2/s)
K_e	Constante de força contra-eletromotriz (V.s/rad)
I_a	Corrente de armadura (A)
V_a	Tensão de armadura (V)
E_a	Tensão contra-eletromotriz (V)
$\dot{\theta}$	Velocidade angular (rad/s)
T_m	Torque do motor (N.m)
T_c	Torque da carga (N.m)

Conhecendo o modelo do motor foi possível descobrir o valor de todos os parâmetros por meio de medições e experimentos que serão apresentados a seguir.

4.1.2 Identificação dos motores

Inicialmente procurou-se utilizar catálogos presentes em trabalhos passados sobre o robô, entretanto, após a comparação dos parâmetros do catálogo com os dados dos primeiros experimentos, percebeu-se que os catálogos não poderiam estar corretos. Então testou-se o parâmetro mais simples, a velocidade (RPM) do motor à 12V e percebeu-se que

nenhum dos motores tinha a velocidade dos catálogos. Tentou-se então achar os catálogos corretos dos motores. Um dos motores possuía a informação do modelo na carcaça porém não foi encontrado nenhum catálogo do modelo específico tanto no site do fabricante quanto em outras ferramentas de busca da internet, somente foram encontrados catálogos de modelos semelhantes que possuíam velocidade diferente, por isso não tinham utilidade. O outro motor não possui nenhum tipo de identificação na carcaça tornando impossível a tarefa de encontrar um catálogo que proporcionasse garantia de exatidão.

Tendo em vista que não se tem nenhuma fonte confiável de informação para esses motores, resolveu-se fazer a modelagem completa dos motores de forma experimental. A premissa básica necessária para que o modelo apresentado represente corretamente o comportamento desses motores é o fato de serem motores CC com ímã permanente. Para ter certeza disso, os motores foram abertos e foram observadas características físicas e de construção como a presença de ímãs em volta do enrolamento do rotor, a presença de escovas e comutadores. Tais características confirmaram que estes são motores CC de ímã permanente.

4.1.3 Parâmetros R_a e L_a

Os parâmetros R_a e L_a dos dois motores foram inicialmente medidos por uma ponte RLC no Laboratório de Instrumentação do Departamento de Engenharia Elétrica da UFRGS através dos terminais dos cabos que alimentam os motores. Esses valores foram utilizados nas primeiras simulações, porém, resultados controversos do modelo geraram suspeitas de que esses valores estavam incorretos. Valores que deveriam ser constantes estavam variando demasiadamente dependendo do ponto de operação.

Os parâmetros foram medidos novamente e apresentaram valores diferentes. Após alguns testes e especulações resolveu-se abrir novamente os motores e observar o funcionamento interno. Analisando o posicionamento das escovas e o formato dos comutadores, percebeu-se que dependendo da posição em que o rotor estava, cada escova encostava em um ou mais de um comutador ao mesmo tempo e isso modificada a medição feita a partir dos terminais de alimentação dos motores. Após esta conclusão, realizou-se uma nova medição com o mesmo instrumento utilizado inicialmente, entretanto, desta vez se mediu os parâmetros diretamente nas escovas do rotor. Essa medição gerou valores que apresentaram resultados compatíveis nas simulações eliminando as anormalidades encontradas anteriormente.

4.1.4 Parâmetros K_t, K_e e B_m

Para determinação dos demais parâmetros um experimento foi desenvolvido no qual o motor se move entre duas posições sob uma tensão de entrada constante. O arquivo que implementa este teste se chama *experimento_motor.slx*. Este e todos os outros códigos citados no texto podem ser encontrados no apêndice A.

Através da função PWM da toolbox do Arduino, o usuário é capaz de escolher qualquer tensão entre 0V e 12V para ser aplicada no experimento. Esse arquivo salva em vetores no *workspace* do MATLAB, as variáveis de posição, velocidade, corrente e tensão de armadura adquiridos durante o experimento. Então é utilizado o comando *save('motor1')* para salvar

todas as variáveis do *workspace* em um arquivo. Com esses dados é possível verificar os seguintes valores em regime permanente: V_a, $\dot{\theta}$, I_a.

Considerando que em regime permanente as derivadas da corrente e da velocidade são nulas, tem-se que a equação (10) se torna:

$$V_a = R_a I_a + E_a \,. \tag{14}$$

Logo, substituindo (11) em (14) tem-se:

$$K_e = \frac{V_a - R_a * I_a}{\dot{\theta}} \,. \tag{15}$$

Desta forma calcula-se K_e, agora K_t será calculado. Utilizando a lei da conservação de energia, faz-se a aproximação de que a potência mecânica consumida pelo motor é igual à potência elétrica entregue pela armadura, então conclui-se que:

$$T_m \dot{\theta} = E_a I_a \,. \tag{16}$$

Substituindo (3) e (5) em (7) tem-se que:

$$K_t = K_e \,. \tag{17}$$

Agora, sabendo que T_c é nulo pelo fato de que os ensaios são feitos sem carga, tem-se que a equação (13) em regime permanente fica:

$$T_m(t) = B_m \dot{\theta} \,. \tag{18}$$

Substituindo (3) em (9), tem-se:

$$B_m = \frac{K_t I_a(t)}{\dot{\theta}} \,. \tag{19}$$

Utilizando o procedimento descrito, calculam-se os parâmetros K_t, K_e e B_m para os dois motores. Inicialmente foi escolhido o ponto de operação de 6V para fazer a modelagem. No ensaio feito, foram avaliados os valores de tensão de entrada (V_a), corrente em regime permanente (I_a) e velocidade angular em regime permanente ($\dot{\theta}$). Conhecendo o valor de R_a que foi medido anteriormente, usam-se as equações (15) e (17) para calcular o valor das constantes K_t e K_e. Com o uso da equação (19) calcula-se B_m. Agora resta somente o cálculo de J_m.

OBS: O sinal de corrente adquirido nos ensaios é extremamente ruidoso devido ao sinal PWM aplicado nos motores, para descobrir o valor dele em regime permanente foi utilizado um filtro de média móvel de 50 amostras para cada lado. A comparação entre sinal original e filtrado é observada na Figura 4-1.

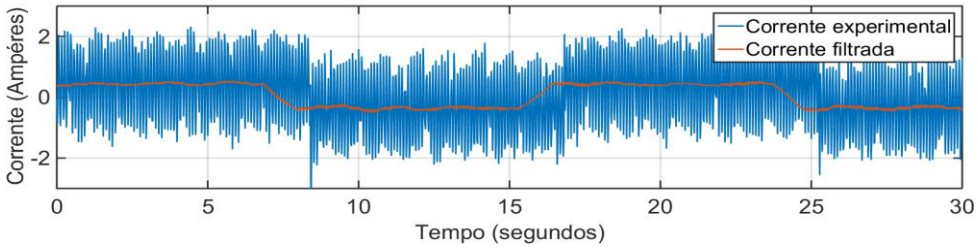


Figura 4-1: Comparação entre sinais de corrente original e filtrado.

4.1.5 Parâmetro J_m

O procedimento apresentado anteriormente não é capaz de calcular o parâmetro J_m mas através da ferramenta de identificação de sistemas do MATLAB, foi comprovado que, neste experimento, esse valor tem influência desprezível na dinâmica do modelo de posição e pode ser considerado zero.

Segundo o modelo apresentado nas equações (10) a (13) tem-se que a função de transferência de terceira ordem da posição do rotor em relação à tensão de entrada é:

$$\frac{\theta(t)}{V_a(s)} = \frac{K_t}{L_a J_m s^3 + (L_a B_m + R_a J_m)s^2 + (R_a B_m + K_e K_t)s}. \tag{20}$$

Quando J_m e L_a são valores muito pequenos, os dois respectivos polos se posicionam muito distantes da origem, o que caracteriza uma dinâmica muito rápida. Quanto o transiente da dinâmica destes dois polos acaba, o que resta é apenas o integrador, caracterizando uma função de transferência de primeira ordem. Substituindo $J_m = 0$ e $L_a = 0$ na equação (20) tem-se a equação (21) como sendo o modelo de primeira ordem:

$$\frac{\theta(t)}{V_a(s)} = \frac{\frac{K_t}{R_a B_m + K_e K_t}}{s}. \tag{21}$$

Neste experimento dois fatores dificultam a manifestação destes polos: primeiro, a taxa de amostragem é de 0,021 segundos, variações muito rápidas geradas por esses polos podem não ser percebidas; segundo, a resolução do deslocamento para os sensores utilizados não é muito alta e é fortemente corrompida por ruído.

Para comprovar que J_m assim como L_a são tão pequenos que não exercem influência no experimento, a *toolbox* de identificação de sistemas do MATLAB foi utilizada para fazer a identificação do sistema utilizando dois modelos, um de terceira ordem e outro de primeira ordem (integrador).

Os dados de um dos experimentos feitos foram importados na *toolbox* de identificação e foram configurados dois tipos de modelos a terem seus parâmetros identificados: um modelo com 1 polo e nenhum zero e um modelo com 3 polos e nenhum zero. A saída do modelo de primeira ordem identificado obteve 96.3% de eficácia quando comparado com a saída dos dados do experimento enquanto o modelo de terceira ordem identificado obteve 97.23% de eficácia. Isso é um indicativo que, neste caso, um modelo de primeira ordem é praticamente tão eficaz quanto um de terceira ordem.

O polo identificado para o modelo de primeira ordem tem valor -0,0026 s^{-1} e os polos identificados para o modelo de terceira ordem tem valores -160,7 s^{-1}, -20.6 s^{-1} e -0,018 s^{-1}. Os polos identificados de valor -160,7 s^{-1} e -20,6 s^{-1} no modelo de terceira ordem possuem constantes de tempo de 0,0062 segundos e 0,0485 segundos respectivamente. Sabendo que o tempo de amostragem do experimento é 0,021 segundos, conclui-se que a dinâmica desses polos é tão rápida que não se manifesta significativamente e não pode ser medida com confiabilidade neste experimento.

4.1.6 Parâmetros identificados

Os valores encontrados para os dois motores operando com 6V como tensão de entrada podem ser observados na Tabela 4-2.

Tabela 4-2: Parâmetros identificados para os motores para operação em 6V.

Parâmetros	Motor 1	Motor 2
R_a	1,03 Ω	1,26 Ω
L_a	0,007533 H	0,00970 H
K_t	0,7759 N.m/A	1,2677 N.m/A
J_m	0 kg.m^2	0 kg.m^2
B_m	0,2449 kg.m^2/s	0,1043 kg.m^2/s
K_e	0,7759 V.s/rad	1,2677 V.s/rad

Para saber se este modelo aproximado é válido para operações com outras tensões diferentes de 6V, foram calculados também os parâmetros K_t, K_e e B_m baseados em outros ensaios de 2V, 4V, 8V, 10V e 12V. Os parâmetros J_m, R_a e L_a permaneceram inalterados. Os valores de K_t, K_e e B_m encontrados para os diferentes ensaios são apresentados na Figura 4-2.

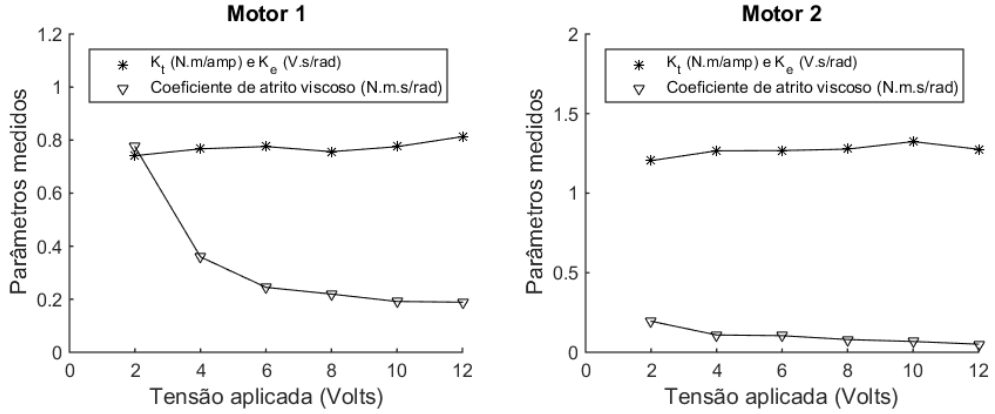


Figura 4-2: Valores de Ke, Kt e Bm encontrados para ensaios de tensão de entrada variada.

Pode-se perceber que para ambos os motores, os valores das constantes dos mesmos tem pouca variação durante os diferentes ensaios. Entretanto, o valor do atrito viscoso não se mostra constante em todos os ensaios, principalmente no caso do motor 1.

Isso se deve ao fato de que quando a tensão de entrada é baixa, a velocidade do rotor é baixa e nesta faixa de operação o atrito seco se manifesta (Suzuki, 2010). Nosso modelo não considera o atrito seco mas ele está presente no experimento, ele se manifesta nos ensaios de baixa velocidade na forma de um amortecimento extra e o processo de identificação acaba incluindo esse efeito no coeficiente de atrito viscoso. Este é o motivo de existir uma variação no parâmetro de atrito viscoso sendo que esse parâmetro não deveria variar.

4.1.7 Validação dos modelos dos motores

Através dos programas *modelagemmotor1.m* e *modelagemmotor2.m* pode-se simular os modelos e compará-los com os dados dos experimentos utilizando como entrada do modelo a mesma entrada utilizada no experimento. Nas figuras Figura 4-3 e na Figura 4-4 podem ser observadas as comparações das respostas do modelo obtido e do experimento real para os ensaios de 6V.

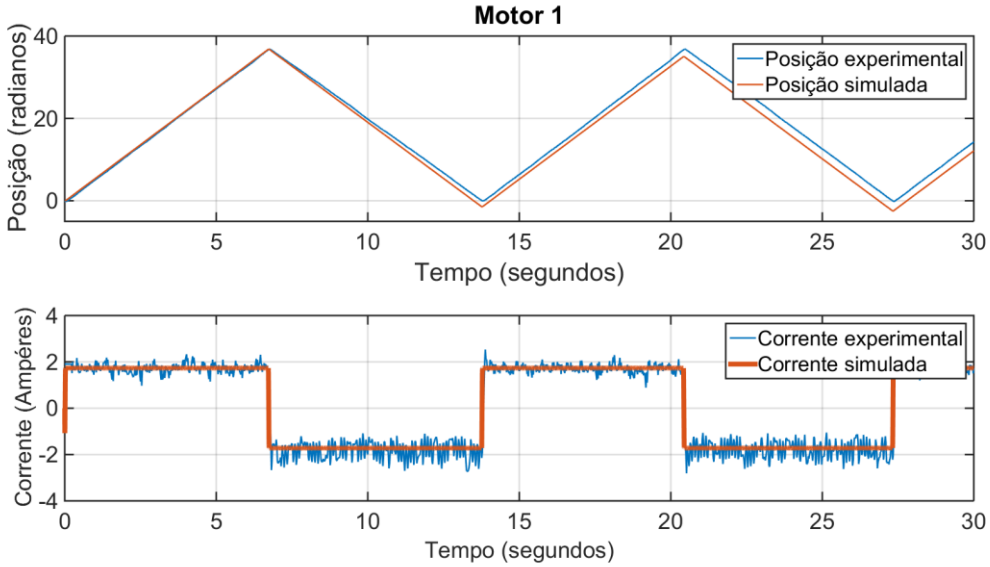


Figura 4-3: Resposta do motor 1 comparada com a do modelo.

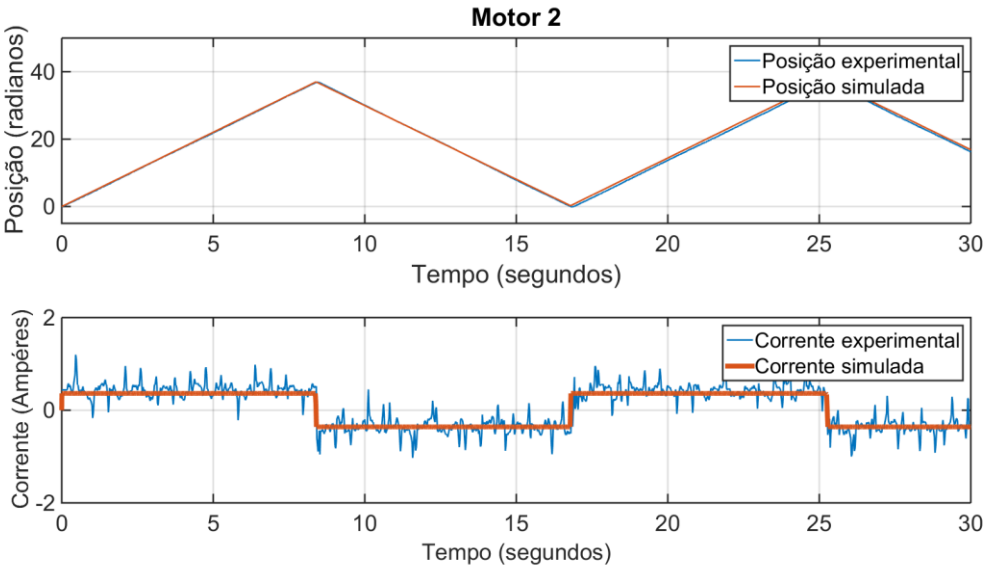


Figura 4-4: Resposta do motor 2 comparada com a do modelo.

Pode-se perceber que para o motor 1 durante rampa de subida na posição, o modelo se comporta igual ao experimento, porém, na rampa de descida o modelo tem velocidade maior do que a do experimento. Isso acontece porque o motor 1, devido a alguma característica de construção desconhecida, não se comporta da mesma forma quando gira em sentidos diferentes. Todos os modelos foram feitos baseados na velocidade positiva dos ensaios, por isso, é esperado que exista uma diferença maior entre modelo e experimento quando o motor gira com velocidade negativa. Felizmente o motor 2 não apresenta essa característica e se comporta de forma semelhante quando gira em ambos os sentidos.

Para avaliar a eficácia do modelo aproximado em 6V operando em outras faixas de operação, foi feito um experimento através de uma versão modificada do programa *experimento_motor.slx*. Neste experimento, a tensão de entrada no motor varia de 0 a 12V na forma de uma senoide porém trocando de sinal sempre que o motor atinge uma das duas posições determinadas. O motor necessita ficar variando entre duas posições porque o sensor potenciométrico possui um limite de 10 voltas. O resultado obtido para os dois motores pode ser observado nas figuras Figura 4-5 e Figura 4-6.

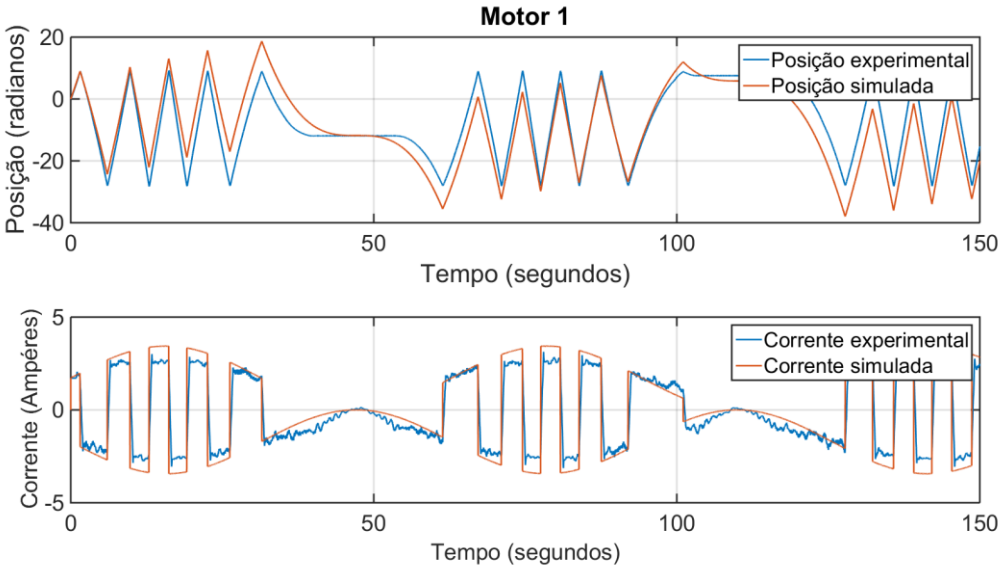


Figura 4-5: Simulação com tensão de entrada variante para o motor 1.

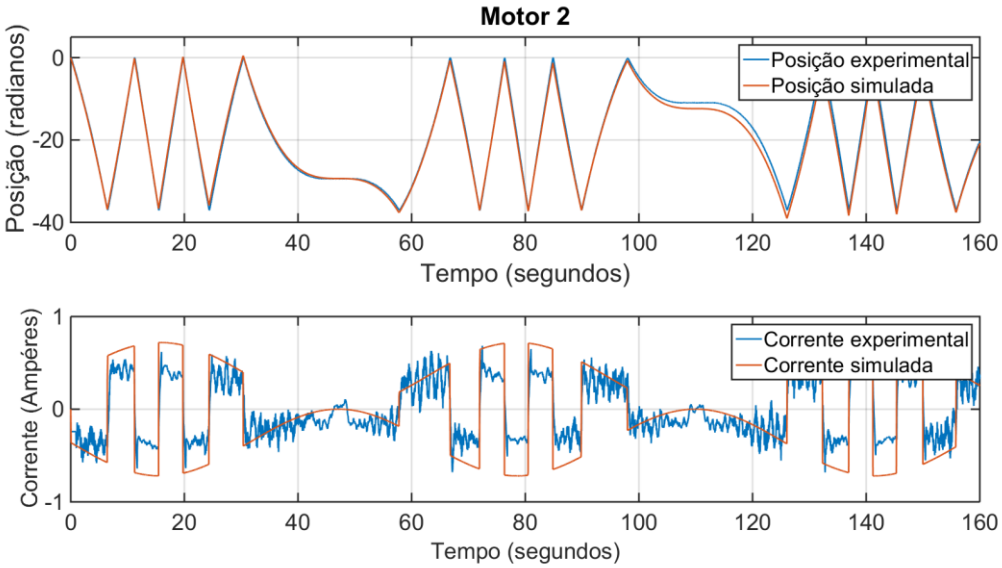


Figura 4-6: Simulação com tensão de entrada variante para o motor 2.

Observa-se uma considerável diferença entre modelo e experimento para o motor 1 em alguns momentos. Isso acontece devido à característica mencionada anteriormente desse motor, que faz o mesmo se comportar de forma diferente dependendo o sentido de rotação. Já o motor 2 apresenta um modelo que segue fielmente a posição obtida no experimento. Em relação à corrente, nota-se que para valores pequenos (tensão baixa) e para valores altos (tensão alta) a mesma apresenta um erro maior do que para valores intermediários. Isso é compreensível uma vez que o modelo que está sendo utilizado nesta simulação foi obtido com um ensaio de 6V, logo, é mais fiel à planta real quando a mesma opera nessa faixa de tensão.

4.2 Dinâmica do manipulador: cálculo dos parâmetros

O modelo dinâmico deste manipulador foi apresentado na seção 2.3. Agora serão calculados e identificados todos os parâmetros apresentados nesse modelo de forma que se torne possível a simulação da dinâmica do manipulador. Todas as fórmulas de momento de inércia apresentadas foram retiradas de (Murray, et al., 1994).

4.2.1 Cálculo dos parâmetros de forma analítica

Para calcular o momento de inércia e o centro de massa dos elos de forma analítica foram feitas algumas aproximações. O elo 1 foi aproximado para um cilindro (motor 2) e uma barra retangular. O elo 2 foi aproximado para apenas uma barra retangular. As aproximações podem ser observadas na Figura 4-7.

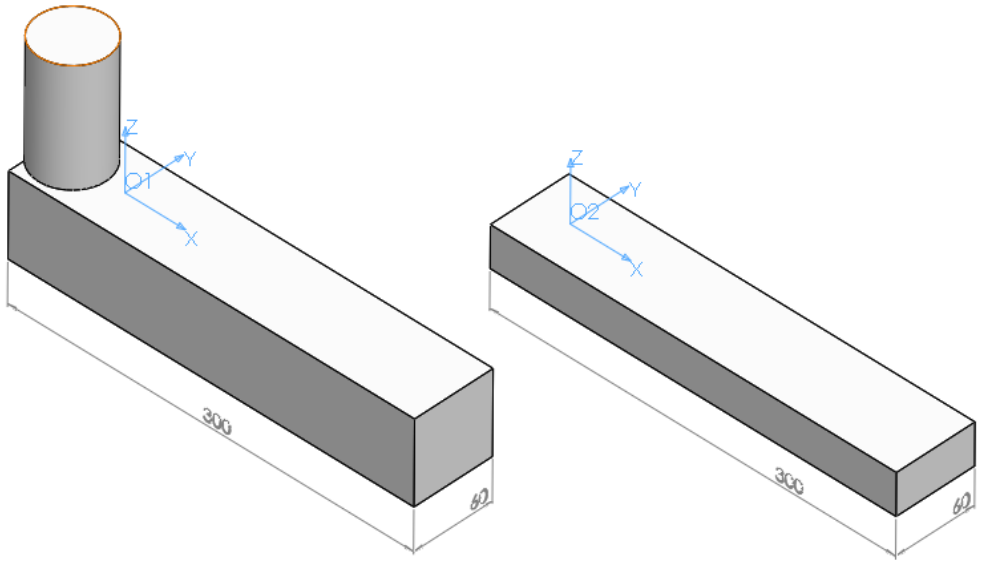


Figura 4-7: Modelos aproximados dos elos para cálculo analítico (medidas em mm).

Os sistemas de coordenadas O1 e O2 apresentados na Figura 4-7 são os eixos de rotação dos elos. Eles serão usados para referenciar as posições dos centros de massa. Para simplificar os cálculos, os centros de massa serão calculados apenas nas dimensões x e y pois a posição z não influencia no momento de inércia desejado.

O centro de massa do elo 1 é uma combinação do centro de massa do cilindro e do da barra. Sendo esses dois objetos simétricos, seus centros de massa se posicionam no seu centro geométrico. Esses valores referenciados a O1 são representados pelas equações (22) e (23):

$$r_{cilindro} = [x_c \quad y_c] = [-0{,}0393 \quad 0]\, m\,, \tag{22}$$

$$r_{barra} = [x_b \quad y_b] = [0{,}0930 \quad 0]\, m\,. \tag{23}$$

As equações (24) e (25) apresentam as massas dos elos medidas com uma balança:

$$m_{cilindro} = 0{,}604\ kg\,, \tag{24}$$

$$m_{barra} = 1{,}522\ kg\ . \tag{25}$$

Logo, pode-se calcular o centro de massa do conjunto dos dois objetos através das equações (26), (27) e (28):

$$r_{x1} = \frac{0{,}604 * (-0{,}0393) + 1{,}522 * 0{,}0930}{0{,}604 + 1{,}522} = 0{,}0554\ , \tag{26}$$

$$r_{y1} = \frac{0{,}604 * 0 + 1{,}522 * 0}{0{,}604 + 1{,}522} = 0\ , \tag{27}$$

$$r_1 = [0{,}0554 \quad 0]. \tag{28}$$

Para calcular o momento de inércia, tem-se as equações (29) e (30):

$$I_{cilindro} = \frac{m_c R^2}{2}\ , \tag{29}$$

$$I_{barra} = \frac{m_{barra}}{12}(W_1^2 + L_1^2)\ . \tag{30}$$

Onde R é o raio do cilindro e tem valor 0,02532 metros, W_1 é a largura da barra e tem valor 0,06 metros e L_1 é o comprimento da barra e tem valor 0,3 metros. Esses valores de momento de inércia são referentes ao eixo de rotação que passa pelo centro de massa de cada peça e tem direção paralela a z. Para calcular o momento de inércia do conjunto em relação sistema de coordenadas alinhado com O1 e posicionado no centro de massa deve-se que utilizar o teorema dos eixos paralelos para calcular cada momento de inércia separadamente e depois somá-los. As equações (31) e (32) representam os momentos de inércia de cada objeto já com a parcela do teorema de eixos paralelos:

$$I^*_{cilindro} = \frac{m_c R^2}{2} + m_c d_c^2\ , \tag{31}$$

$$I^*_{barra} = \frac{m_{barra}}{12}(W_1^2 + L_1^2) + m_b d_b^2\ . \tag{32}$$

Onde d_c e d_b são as distâncias entre os eixos de rotação das peças em relação ao centro de massa do conjunto $r_x - x_c$ e $r_x - x_b$ respectivamente. Somando as duas parcelas tem-se a equação (33) que representa o momento de inércia total do elo 1:

$$I_{zz1} = \frac{m_c R^2}{2} + m_c d_c^2 + \frac{m_{barra}}{12}(W_1^2 + L_1^2) + m_b d_b^2 = 0{,}01963025\,\frac{kg}{m^2}\,. \tag{33}$$

Para o elo 2 o procedimento é mais simples pelo fato de ser apenas um objeto e ser simétrico. O centro de massa é o centro geométrico, então tem-se que a equação (34) apresenta o centro de massa calculado:

$$r_2 = [r_{x2} \quad r_{y2}] = [0{,}12 \quad 0]\,. \tag{34}$$

O momento de inércia da barra em relação ao eixo z de um sistema de coordenadas alinhado com O2 e posicionado no centro de massa é dado pela equação (35):

$$I_{zz2} = \frac{m_2}{12}(W_2^2 + L_2^2)\,. \tag{35}$$

Onde m_2 é a massa do elo 2 e tem valor 1,011 kg, W_2 e L_2 tem mesmo significado e valor que W_1 e L_1. Substituindo esses valores na equação (35), temos a equação (36):

$$I_{zz2} = 0{,}00789199\,\frac{kg}{m^2} \tag{36}$$

A comparação deste modelo com os valores experimentais é apresentada na seção 4.2.3 nas figuras Figura 4-10 e Figura 4-11.

4.2.2 Cálculo dos parâmetros via software

Para o cálculo destes parâmetros em cada elo, foi utilizado o software SolidWorks no qual um modelo sólido aproximado de cada elo foi criado. O elo 1 foi aproximado para 3 peças principais: o motor 2, um elo superior, e um elo inferior. O elo 2 foi aproximado com duas peças: um elo inferior e um superior. Partes menores como cabos, parafusos, polias e cabos foram ignorados. Para as peças de alumínio foi utilizada a densidade padrão do alumínio 1060 e para o motor, a densidade foi calculada baseada no volume calculado pelo SolidWorks e pelo valor obtido com a pesagem em uma balança de precisão. O modelo sólido dos elos pode ser observado na Figura 4-8.

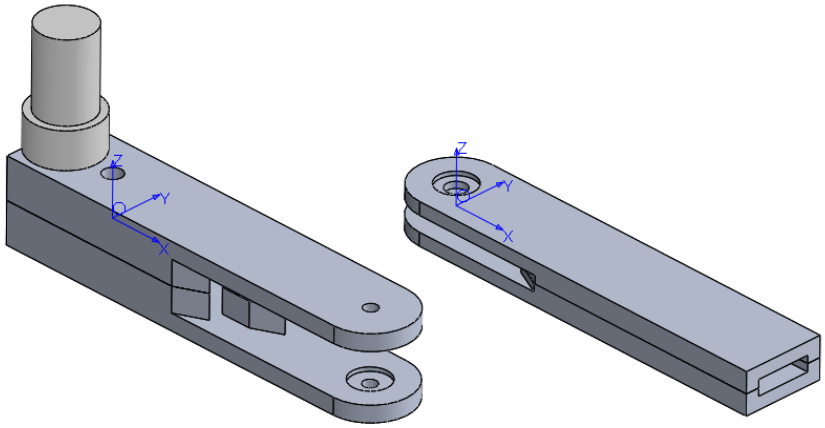


Figura 4-8: Modelos sólidos dos elos com seus sistemas de coordenadas auxiliares, elo 1 à esquerda e elo 2 à direita.

Foram definidos sistemas de coordenadas auxiliares equivalentes aos eixos de rotação dos elos e através da ferramenta *Mass Properties* foram calculados os valores de massa, centro de massa e tensor de inércia apresentados na Tabela 4-3. O valor do atrito viscoso foi encontrado experimentalmente de acordo com o procedimento apresentado a seguir.

Tabela 4-3: Parâmetros necessários para simulação dos 2 elos do manipulador.

Parâmetros	Elo 1	Elo 2
Massa (kg)	$m_1 = 2{,}12669573$	$m_2 = 1{,}01179436$
Comprimento (m)	$l_1 = 0{,}213$	$l_2 = 0{,}270$
Centro de massa (m)	$r_{x1} = 0{,}04197199$ $r_{y1} = 0{,}00000000$ $r_{z1} = 0{,}01758825$	$r_{x2} = 0{,}13135414$ $r_{y2} = 0{,}00000000$ $r_{z2} = -0{,}00001150$
Tensor de inércia (kg*m^2)	$^{I_1} = \begin{bmatrix} 0{,}00342902 & 0{,}00000000 & -0{,}00302056 \\ 0{,}00000000 & 0{,}01834578 & 0{,}00000000 \\ -0{,}00302056 & 0{,}00000000 & 0{,}01610542 \end{bmatrix}$	$^{I_2} = \begin{bmatrix} 0{,}00043250 & 0{,}00000000 & 0{,}00000159 \\ 0{,}00000000 & 0{,}00696966 & 0{,}00000002 \\ 0{,}00000159 & 0{,}00000002 & 0{,}00720124 \end{bmatrix}$
Atrito viscoso (N.m.s/rad)	Modelo analítico: $B_{l1} = 0{,}0430$ Modelo do SW: $B_{l1} = 0{,}0340$	Modelo analítico: $B_{l1} = 0{,}0300$ Modelo do SW: $B_{l1} = 0{,}0320$

4.2.3 Validação do modelo do manipulador

Para simular a dinâmica desses elos foi utilizada a *Robotics Toolbox* v. 9.10 com o MATLAB 2014b x64. Foi utilizado o bloco de dinâmica direta de um manipulador serial que, segundo (Corke, 2011), implementa a dinâmica na forma:

$$\ddot{\boldsymbol{q}} = \overline{D}^{-1}(\boldsymbol{q})\left(\boldsymbol{\tau} - \overline{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \overline{B}\dot{\boldsymbol{q}}\right);$$

através de um algoritmo de integração e considerando as condições iniciais nulas, o bloco é capaz de simular a dinâmica do manipulador devido à uma entrada de torque. Essa equação é derivada da dinâmica modelada no início da seção 2.3.

Foi criado um programa no MATLAB/Simulink (*link1.m* e *link2.m*) que cria um manipulador de apenas um elo através da função *SerialLink()* e posiciona esse elo de forma que seu eixo de rotação seja perpendicular ao eixo da aceleração gravitacional, então o programa simula o giro livre desse elo a partir de uma posição inicial de 90° em relação ao eixo gravitacional. A única força que faz o elo se mover nesta simulação é a força gravitacional. Inicialmente foram usados os parâmetros geométricos calculados pelo SolidWorks e atrito viscoso de valor zero. Uma imagem da animação gerada pela simulação pode ser observada na Figura 4-9.

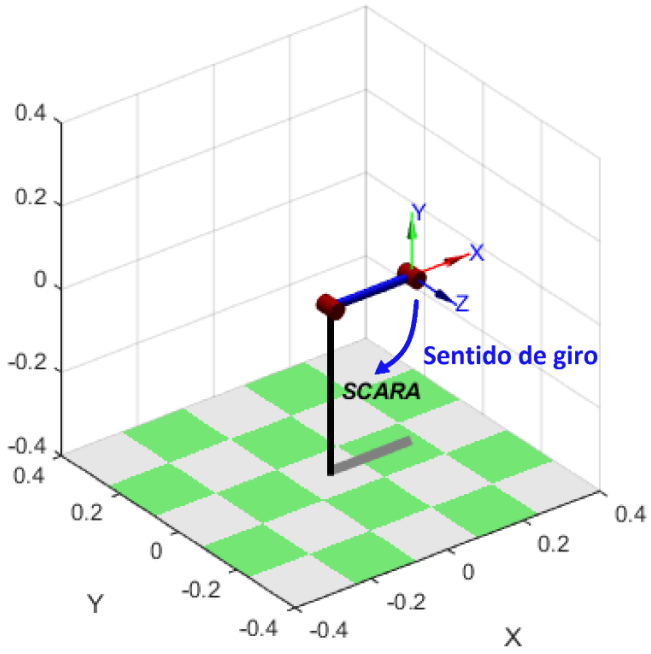


Figura 4-9: Imagem da simulação da queda livre do elo.

Tendo os resultados da simulação, foi preparado um experimento no qual os elos foram posicionados da mesma forma utilizada na simulação. Seus eixos de rotação foram fixados com uma morsa e devidamente alinhados com o auxílio de uma régua de nível. Foram adquiridos os dados do giro livre com o abandono em 90° e somente foram utilizados os ensaios em que a posição final foi 0° (medido com a régua de nível). Devido ao atrito seco, nem em todos os ensaios a posição final era exatamente alinhada com o eixo da gravidade.

Através da comparação dos resultados do experimento e dos resultados da simulação, o parâmetro B_l foi descoberto. Os valores de B_l escolhidos foram os que proporcionaram o mesmo amortecimento para a simulação e para o experimento. Esses valores de atrito viscoso podem ser observados na Tabela 4-3. A comparação entre simulação dos modelos com os valores experimentais são apresentados nas figuras Figura 4-10 e Figura 4-11.

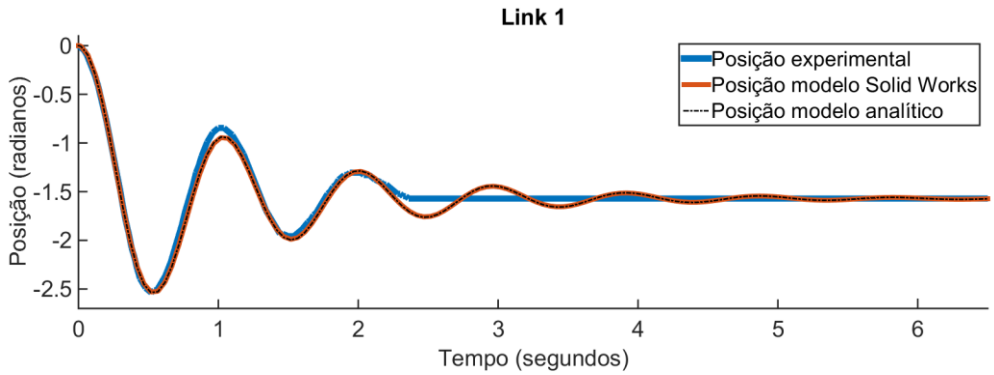


Figura 4-10: Comparação entre modelo e experimento para elo 1.

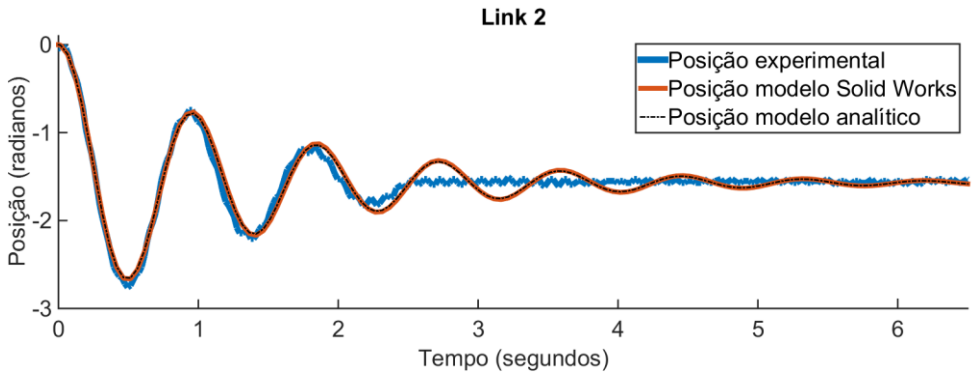


Figura 4-11: Comparação entre modelo e experimento para elo 2.

A maior diferença entre experimento e simulação que ocorre aproximadamente a partir de 2 segundos, isso acontece devido ao atrito seco existente no experimento que impossibilita a oscilação de pequena amplitude. Quando a oscilação é muito pequena, a força gravitacional não é maior do que a força de atrito seco e o elo para de se mover. A *robotics toolbox* disponibiliza a entrada de parâmetros de atrito seco porém foi decidido não fazer essa modelagem neste trabalho.

Outra característica presente na comparação é o fato de que o modelo analítico se comportou praticamente de forma igual ao modelo do SolidWorks neste experimento. Embora os parâmetros de centro de massa e momento de inércia calculados em cada método sejam diferentes como pode ser observado na Tabela 4-3, isso só mudou o amortecimento do sistema, a frequência de oscilação continuou igual. Neste caso, o modelo analítico funcionou tão bem quanto o do SolidWorks, mas o resultado sobre outros experimentos/simulações é incerto. Portanto, para simulações futuras, o modelo do SolidWorks será utilizado pelo fato de ser um modelo mais fiel à realidade e ter uma quantidade de detalhes e precisão maior.

4.3 Geração de trajetória

Para geração da trajetória, novamente foi utilizada a *Robotics Toolbox*. Através do script *trajetoria.m*, a trajetória necessária para que o manipulador "desenhasse" um quadrado de 20 cm de lado no ar foi criada.

A *toolbox* possui uma série de comandos que possibilitam entre outras coisas, a geração da trajetória do manipulador. O primeiro comando utilizado é chamado *transl()* que cria uma matriz homogênea de transformação referente à uma translação de tamanho arbitrário. Foram criadas quatro matrizes com esse comando, uma para cada canto do quadrado utilizando as coordenadas cartesianas dos cantos como entrada. Tendo essas matrizes foi utilizado o comando *ctraj()* para gerar trajetórias em linha reta que ligam esses quatro pontos. Esse comando divide a trajetória em um número desejado de pontos e cria uma matriz de transformação homogênea para cada ponto. No caso do comando *ctraj()* a trajetória gerada é retilínea no espaço de coordenadas cartesiano e possui as três primeiras derivadas contínuas, característica que garante uma trajetória suave (Corke, 2011).

Tendo a trajetória no espaço cartesiano calculada, usa-se a cinemática inversa do manipulador apresentada na seção 2.2.1 para calcular as trajetórias correspondentes no espaço de coordenadas generalizadas. Para isso, o comando *SerialLink()* é utilizado para

criar um objeto referente ao manipulador utilizando todos os parâmetros calculados nas seções 4.1 e 4.2. Através do método *ikine()* do objeto do manipulador, aplica-se a cinemática inversa do manipulador em cada ponto da trajetória criando duas novas trajetórias, uma para cada junta. Para criar a trajetória do manipulador no espaço de coordenadas cartesianas, foi utilizado o método *fkine()* que utiliza a cinemática direta do manipulador calculada na seção 2.2.2 para gerar essa trajetória a partir das trajetórias de coordenadas generalizadas. Todas essas trajetórias serão apresentadas na seção de resultados.

5 Resultados

O experimento final realizado neste trabalho foi o seguimento de trajetória realizado tanto pelo manipulador real quanto em simulação. Nesta seção os resultados serão apresentados.

5.1 Experimento

Através do arquivo *trajetoria.m* a trajetória de um quadrado no espaço cartesiano de lado 20 cm é criada assim como as trajetórias respectivas para as coordenadas generalizadas. Essas últimas serão utilizadas como referência para os controladores das juntas. A posição inicial do quadrado é o ponto (0,1m 0,2m) do espaço cartesiano. No experimento, um controlador PD com realimentação de posição foi implementado para cada junta. A escolha de um controlador PD se deve ao fato de que o sistema em si já é um integrador (observado na equação (21)), logo, não há necessidade de um controlador integrador. O ganho derivativo será responsável por acelerar a resposta caso o ganho proporcional não seja suficiente.

Decidiu-se primeiramente sintonizar o controlador manualmente por motivos de segurança. Os ganhos foram sendo aumentados gradativamente sempre com cuidado para não instabilizar o sistema e arriscar a segurança do equipamento ou do usuário. Depois de algumas tentativas, percebeu-se que a zona morta presente nos motores estava atrapalhando muito o desempenho do controlador, então se utilizou uma compensação de zona morta na qual a saída mínima do motor 1 é 2,11 V e para o motor 2, 0,23 V. Esses valores foram escolhidos da mesma forma que os ganhos dos controladores.

Após algumas tentativas, concluiu-se que o melhor desempenho possível para o este sistema sem que o mesmo se torne instável é obtido com os ganhos apresentados na Tabela 5-1.

Tabela 5-1: Ganhos dos controladores do experimento

Ganhos	PD do motor 1	PD do motor 2
P (V/°)	0,9412	0,0235
D (V.s/°)	0,5647	0,0038

O controlador PD implementado pelo bloco do Simulink utilizado é apresentado na equação (37):

$$U(s) = P + D\frac{N}{1 + N\frac{1}{s}}, \tag{37}$$

onde U é o sinal de controle, P é o ganho proporcional, D é o ganho derivativo e N é o coeficiente de filtro para altas frequências que foi mantido com o valor padrão de 100 s^{-1} para os dois controladores.

O desempenho experimental obtido com esses controladores pode ser observado nas figuras Figura 5-1 e Figura 5-2. Na Figura 5-1 observa-se as referências de trajetórias em coordenadas generalizadas assim como a resposta do sistema em malha fechada tentado

seguir essa referência. Na Figura 5-2, é apresentada a trajetória desejada e a obtida pelo sistema em coordenadas cartesianas.

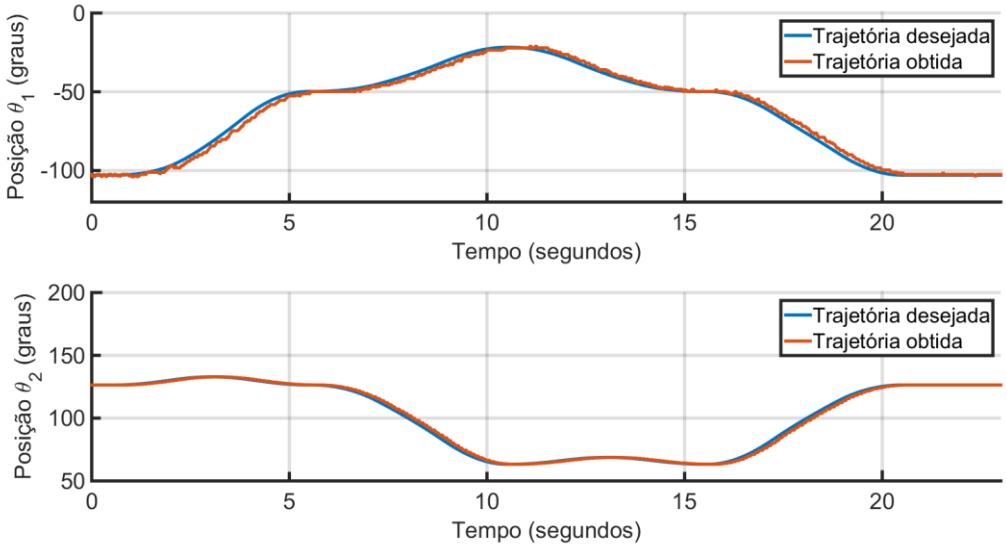


Figura 5-1: Referências de trajetória e resposta experimental em coordenadas generalizadas.

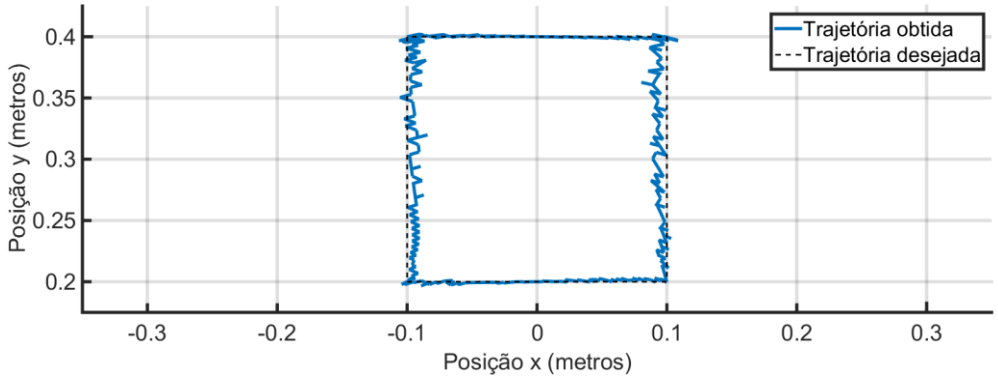


Figura 5-2: Referência de trajetória e resposta experimental em coordenadas cartesianas.

Percebe-se que o maior erro apresentado na trajetória cartesiana encontra-se nas retas verticais. Isso acontece devido à trepidação causada pelo compensador de zona morta aplicado ao primeiro motor. Percebe-se que o seguimento de trajetória da junta 1 não é tão eficiente quanto o da junta 2, em alguns momentos se percebe que a trajetória obtida é levemente "ruidosa". Este erro apresentado na trajetória da junta 1 não se manifesta nas trajetórias horizontais do plano cartesiano por causa da geometria do manipulador. Neste caso, a trajetória retilínea está "tangenciando" o raio de giro da junta 1, desta forma por mais que a junta 1 erre, esse erro se manifesta na forma de um movimento para frente e para trás em cima da própria trajetória retilínea. Isso não acontece no caso das trajetórias verticais, quando a junta 1 erra neste caso, a posição do efetuador vai de um lado para o outro de forma perpendicular à trajetória retilínea, causando o grande erro observado. Se em vez de ser a junta 1, fosse a junta 2 que tivesse o maior erro no seguimento de trajetória, seria observado um erro maior nas trajetórias horizontais.

5.2 Simulação

A mesma trajetória aplicada ao sistema real, foi usada para fazer uma simulação. Os modelos dos motores utilizados são os aproximados para o ponto de operação de 2 V, esses modelos foram escolhidos devido ao fato de que essa trajetória trabalha principalmente nessa faixa de tensão. Através do arquivo *meu_modelo.slx*, o modelo dos motores foi acoplado ao o modelo do manipulador e os mesmos controladores utilizados no experimento foram utilizados para fechar a malha. Cada controlador possui saturação de 12V (limite da fonte). As figuras Figura 5-3 e Figura 5-4 apresentam o desempenho obtido nesta simulação.

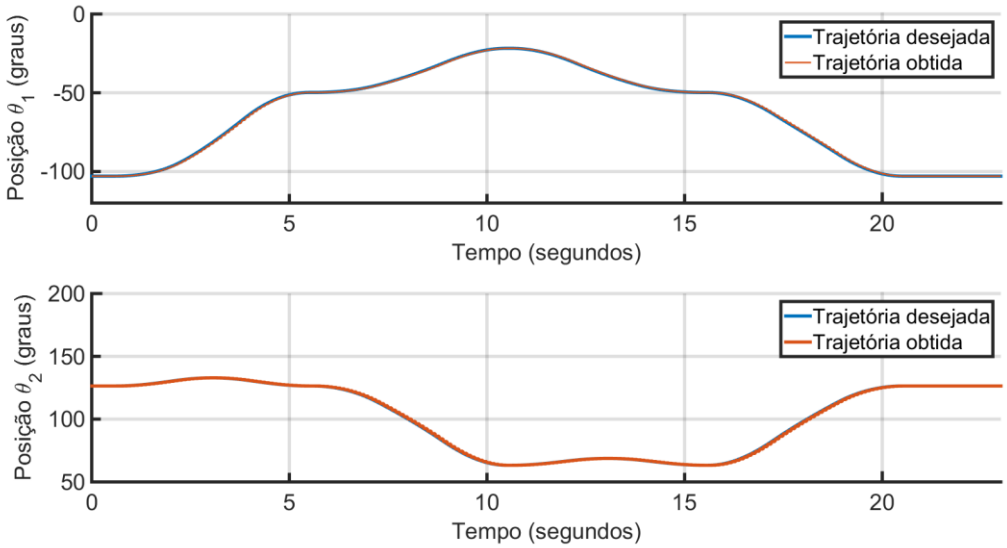


Figura 5-3: Referências de trajetória e resposta simulada em coordenadas generalizadas.

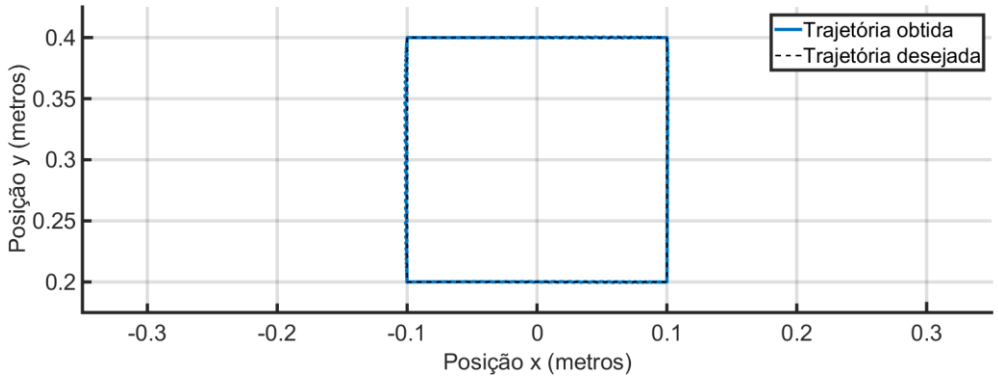


Figura 5-4: Referência de trajetória e resposta simulada em coordenadas cartesianas.

O mesmo controlador se provou mais eficiente na simulação, fato que era esperado sabendo que a simulação não conta com a intervenção de problemas práticos como atrito seco, ruído de medição e baixa resolução de aquisição de dados e de atuação. Um fato interessante é que mesmo em condições ideais, a trajetória cartesiana simulada apresentou o mesmo problema nas retas verticais porém com amplitude menor. Isso indica que além do problema da zona morta no experimento, outro fator ajudou a gerar este erro.

Aparentemente, devido à configuração geométrica do manipulador, este tipo de trajetória amplifica mais os erros nas trajetórias verticais das juntas do que as trajetórias horizontais.

6 Conclusões e Trabalhos Futuros

6.1 Conclusões gerais

Ao fim deste trabalho conclui-se que os objetivos apresentados na introdução foram alcançados, embora alguns problemas inesperados tenham surgido durante o processo de desenvolvimento. A seguir, cada objetivo e respectivo resultado serão apresentados e explicados.

O objetivo 1, que se referia ao desenvolvimento de hardware de acionamento e aquisição de dados para a plataforma, foi cumprido e apresentou os resultados esperados. Em relação às pontes H, a preocupação inicial era que a corrente dos motores não fosse suportada pelo circuito e alguma falha ocorresse, entretanto, a escolha de se fazer as conexões entre componentes com cabos e não trilhas de placa de circuito impresso se provou válida pois mesmo depois de todos os testes executados, os circuitos não apresentam sinais de deterioração. Quanto ao sistema de aquisição de dados, funcionou corretamente, os sensores foram capazes de fornecer os dados de posição e corrente como esperado. Entretanto, constatou-se que a combinação dos sensores potenciométricos de posição com o conversor A/D de 10 bits do Arduino fornecem dados de posição com uma confiabilidade reduzida em relação a um sensor digital (*encoder* óptico). Isso acontece pelo fato de que os sensores potenciométricos utilizados possuem 10 voltas e os limites dos elos do manipulador fazem com que seja utilizada apenas uma parte da faixa de operação total dos sensores. Além disso, o sinal é corrompido por ruído elétrico, ambos problemas não encontrados em sensores do tipo *encoder*. Um problema semelhante ocorreu com os sensores de corrente. Devido ao fato de o seu sinal de saída variar entre uma pequena faixa de valores, apenas uma pequena faixa do conversor A/D foi utilizada para adquirir esse sinal, fato que tornou a resolução do dado de corrente menor do que seria caso se utilizasse uma faixa maior do conversor A/D.

Quanto ao objetivo 2, pode-se considerar que ele foi alcançado, porém existem limitações na solução obtida. O objetivo se refere ao desenvolvimento de uma interface de comunicação flexível e intuitiva para o hardware da plataforma. Foi possível desenvolver esta interface de comunicação através da Arduino *toolbox* para MATLAB e a mesma é de fácil entendimento e de fácil adaptação para outras configurações de hardware. A Figura 6-1 apresenta o programa de Simulink utilizado para os experimentos finais que é a representação dessa interface de comunicação. Este programa controla os dois elos do manipulador fazendo com que os mesmos sigam uma referência que vem do *workspace* do MATLAB através de controladores PD, implementados pelo bloco “PID Controller”. Como pode-se observar na Figura 6-1, para se utilizar outro controlador, ou outra estratégia de controle, a simples substituição do bloco PID por outro(s) bloco(s) desejado(s) deveria ser suficiente. Essa facilidade de substituição dos blocos torna o sistema flexível e adaptável.

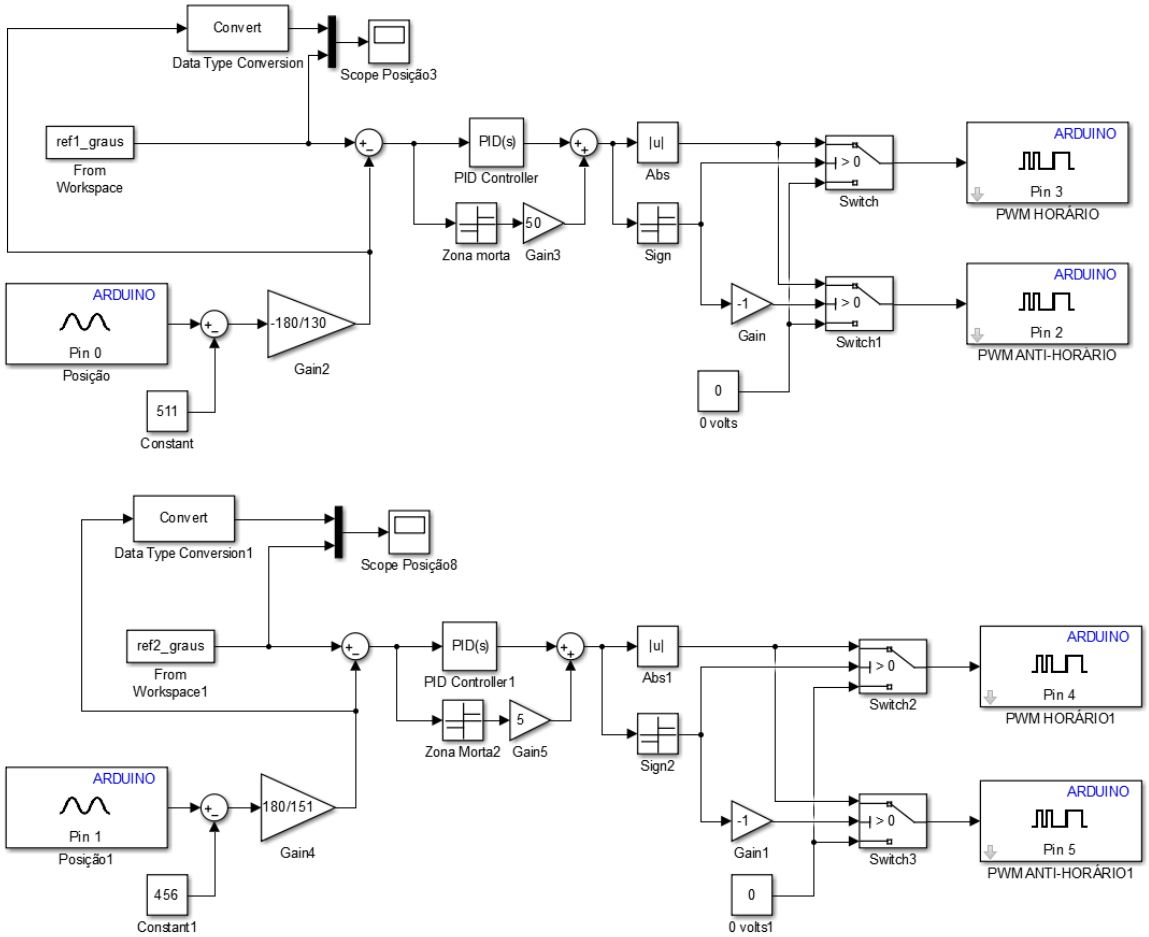


Figura 6-1-Sistema de controle e interface de comunicação desenvolvida.

Entretanto, um problema não esperado ocorreu. A Arduino *toolbox* compila o programa de Simulink desenvolvido e usa a memória do Arduino para armazená-lo de forma que o programa usado tenha um limite de tamanho determinado pelo tamanho da memória do Arduino. O programa apresentado na Figura 6-1 representa exatamente o limite do tamanho da memória do Arduino Mega utilizado. Na tentativa de fazer adaptações ou de colocar mais blocos neste programa, o Simulink alertou que não seria possível porque a memória do Arduino não seria suficiente. Este fato torna o objetivo 2 não completamente alcançado uma vez que, mesmo que a interface seja flexível e facilmente editável, ela possui uma limitação que praticamente impede a implementação de qualquer controlador que exija mais blocos do que este programa inicial.

Já o objetivo 3 obteve um resultado melhor do que o objetivo 2. A modelagem do sistema completo (atuadores e elos do manipulador) foi completada e devidamente validada. Os experimentos e medições realizadas foram suficientes para determinação de todos os parâmetros do sistema e o modelo desenvolvido em Simulink proporcionou resultados coerentes com os resultados obtidos experimentalmente. Desta forma, os futuros usuários da plataforma poderão testar diferentes tipos de controladores em simulação antes de implementá-los na prática. O modelo também poderá ser utilizado para o estudo de robótica, uma vez que o usuário pode testar e aprender sobre geração de trajetória, cinemática e algoritmos de controle através do uso dos programas desenvolvidos nesse trabalho.

6.2 Trabalhos Futuros

- Os problemas de resolução e de memória apresentados nas conclusões poderiam ser resolvidos, ou pelo menos atenuados, com a utilização de um Arduino DUE (também suportado pela *toolbox*) no lugar do Arduino Mega utilizado nesse projeto. O Arduino DUE possui recursos como maior memória e maior velocidade de processamento que seriam úteis na melhoria do projeto.

- Os sensores de posição potenciométricos poderiam ser substituídos por *encoders* digitais. Isso solucionaria o problema de ruído e resolução limitada do conversor A/D.

- Poderia ser feita uma modelagem não linear dos motores uma vez que o atrito se provou um parâmetro não constante. Um modelo não linear aumentaria a confiabilidade dos resultados das simulações.

- Para aumentar a resolução da medição de corrente, poderia ser desenvolvido um circuito condicionador de sinal. Esse circuito poderia aumentar a faixa de operação utilizada do conversor A/D aumentando a resolução.

- Futuramente podem ser construídos os dois graus de liberdade restantes para completar o manipulador SCARA. Este trabalho pode ser utilizado como base para o desenvolvimento do modelo e dos algoritmos de controle do manipulador completo.

7 Referências

Allegro MicroSystems, LLC. 2013. *Fully Integrated, Hall Effect-Based Linear Current Sensor IC with 2.1 kVRMS Isolation and a Low-Resistance Current Conductor.* 2013.

Barber, R., Horra, M. e Crespo, J. 2013. Control Practices using Simulink with Arduino as Low Cost Hardware. *International Journal of Advanced Robotic Systems.* 2013.

Corke, Peter. 2011. *Robotics, Vision and Control.* s.l. : Springer, 2011.

Fateh, Mohammad Mehdi. 2008. On the Voltage-Based Control of Robot Manipulators. *International Journal of Control, Automation, and Systems.* Vol 6, 2008.

Hartenberg, Richard S. e Denavit, Jaques. 1964. *Kinematic Synthesis of Linkages.* s.l. : McGraw-Hill Book Company, 1964.

Keiser, Benjamin. 2013. *Torque Control of a KUKA youBot Arm.* University of Zurich : Master Thesis, Robotics and Perception Group, 2013.

Khosla, Pradeep e Kanade, Takeo. 1987. *Real-Time Implementation and Evaluation of Computed-Torque Scheme.* 1987.

Morel, Guillaume, Iagnemma, Karl e Dubowsky, Steven. 1999. The precise control of manipulators with high joint-friction using. 1999.

Moreno–Valenzuelaa, J. 2010. Analysis and experimental evaluation of a robust saturated PI velocity controller for robot manipulators. 2010.

Murray, Richard M., Li, Zexiang e Sastry, S. Shankar. 1994. *A Mathematical Introduction to Robotic Manipulation.* 1994.

Puente, E.A., Balaguer, C. e Barrientos, A. 1991. Force—torque sensor-based strategy for precise assembly using a SCARA robot. *Robotics and Autonomous Systems .* 1991, Vol. 8.

Rohm Semiconductor. 2009. *Controlling DC Brush Motors with H-bridge Driver ICs.* 2009.

Siciliano, Bruno e Khatib, Oussama. 2008. *Handbook of Robotics.* s.l. : Springer, 2008.

Siciliano, Bruno, et al. 2009. *Robotics: Modelling, Planning and Control.* s.l. : Springer, 2009.

Spong, Mark W., Hutchinson, Seth e Vidyasagar, M. 2004. *Robot Dynamics and Control.* 2004.

Suzuki, Ricardo Murad. 2010. *Controle baseado em linearização por realimentação de estados aplicado a um servoposicionador pneumático.* 2010.

Tekin, Raziye. 2010. MATLAB and Lab VIEW in Modeling, Analysis and Real Time Control of a Motion Control System. *2010 8th IEEE International Conference on Control and Automation.* 2010.

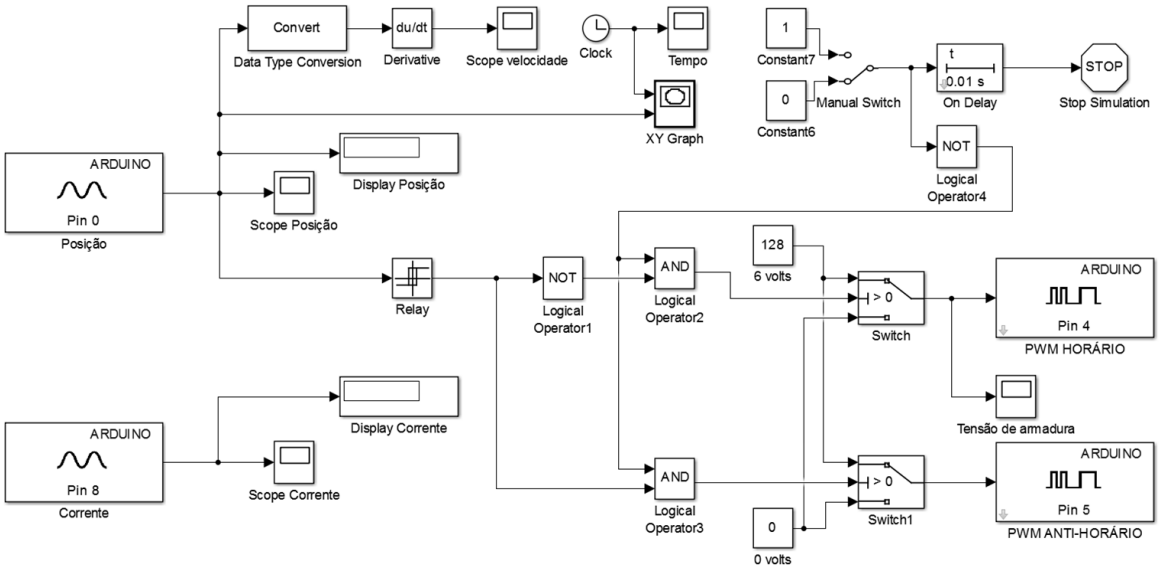
Tsetserukou, Dzmitry, et al. 2005. Optical Torque Sensors for Local Impedance Control. 2005.

Xuejun, Xiang, Xia Ping, Yang Sheng e Ping, Liu. 2007. Real-time Digital Simulation of Control System with LabVIEW Simulation Interface Toolkit. *Proceedings of the 26th Chinese Control Conference.* 2007.

8 APÊNDICE A – Scripts e diagramas de blocos

No apêndice serão apresentados os diagramas de blocos e scripts de MATLAB citados no texto.

8.1 experimento_motor.slx



8.2 modelagemmotor1.m:

```
%% Modelagem motor 1

close all
load('05_06_motor1_6V.mat')
v=6;

inicio=1040;
pwm=pwm(inicio:end,:);
theta=theta(inicio:end,:);
corrente=corrente(inicio:end,:);

tempoinicial = pwm(1,1);

pwm(:,1) = pwm(:,1) - tempoinicial;
corrente(:,1) = corrente(:,1) - tempoinicial;
theta(:,1) = theta(:,1) - tempoinicial;

tsim=theta(end,1);

Va=6
Ia=1.72
w=5.45

La1 = 0.007533;        %H
Ra1 = 1.03;          % ohms
La=La1;
Ra=Ra1;
Ea=Va-Ra*Ia
```

```
Ke=(Ea)/w
Tm=Ea*Ia/w

Bm1=Tm/w
Kt1=Tm/Ia

% La1 = 0.007533;        %H
% Ra1 = 0.8874;          % ohms
%
% Kt1 = 0.8209;            % N.m/amp
% Bm1  = 0.2591;
Ke1 = Kt1;           %volt/(rad/s)
% Jm1 = 0.0035;
Jm1 = 0;

La = La1;
Ra = Ra1;
Kt = Kt1;
Jm = Jm1;
Bm  = Bm1;
Ke = Ke1;

savefile = 'm1';
save(savefile, 'La1', 'Ra1', 'Kt1', 'Jm1', 'Bm1', 'Ke1');

%% Plot dos dados

sim('motormodelagem.slx');

figure('Position', [100, 100, 1400, 600]);
f=28;  %font size
lw=2;  %line width

% Posição experimental
theta(:,2) = theta(:,2) - theta(1,2); % zerando a posicao inicial
theta(:,2) = theta(:,2)*2*pi*10/1024;
subplot(2,1,1);
plot(theta(:,1),theta(:,2),'LineWidth',lw); % plota posição experimental
hold on;
axis([0 30 -5 40])

%Posição simulada

plot(posicaosimulada(:,1),posicaosimulada(:,2),'LineWidth',lw);
legend('Posição experimental','Posição simulada');
title('Motor 1');
ylabel('Posição (radianos)');
xlabel('Tempo (segundos)');
set(gca,'fontsize',f,'LineWidth',lw)
grid on

%Corrente experimental
pontos=4;       %filtro para a corrente
corrente(:,2) = movavg(corrente(:,2),pontos,pontos,0);   % 185 mV / A
corrente(:,2)=(-corrente(:,2)+512)*(5000/(1024*185));
corrente(1:10,2)=corrente(10,2);

subplot(2,1,2);
plot(corrente(:,1),corrente(:,2),'LineWidth',lw);
hold on;
```

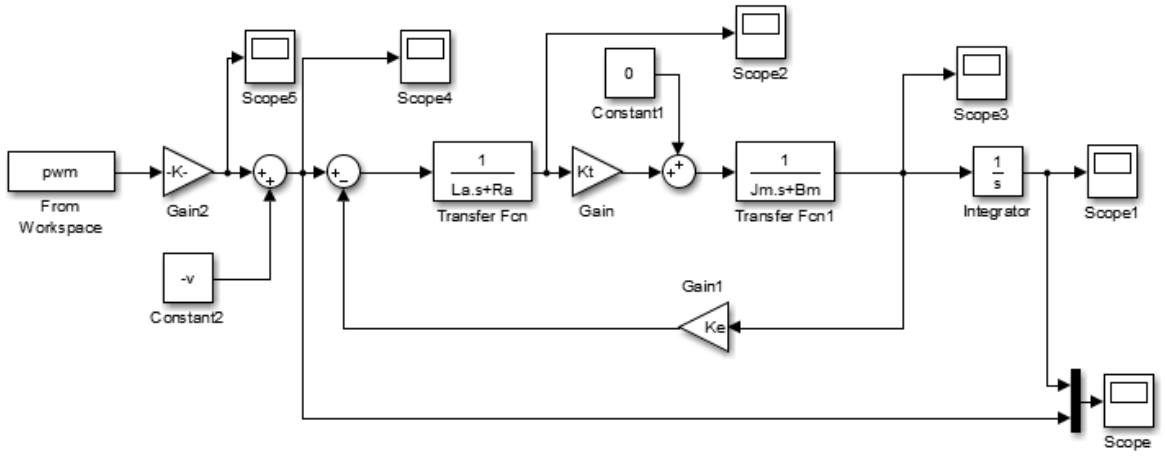
```
%corrente simulada
plot(correntesimulada(:,1),correntesimulada(:,2),'LineWidth',3*lw);
handle=legend('Corrente experimental','Corrente simulada');
% set(handle,'FontSize',f);
ylabel('Corrente (Ampéres)','FontSize',f);
xlabel('Tempo (segundos)','FontSize',f);
set(gca,'fontsize',f,'LineWidth',lw)
grid on
axis([0 30 -4 4])
```

8.3 modelagemmotor2.m:

```
%% Modelagem motor 2

close all
load('05_06_motor2_6V.mat')
v=6;

inicio=435;
pwm=pwm(inicio:end,:);
theta=theta(inicio:end,:);
corrente=corrente(inicio:end,:);

tempoinicial = pwm(1,1);

pwm(:,1) = pwm(:,1) - tempoinicial;
corrente(:,1) = corrente(:,1) - tempoinicial;
theta(:,1) = theta(:,1) - tempoinicial;

tsim=theta(end,1);

Va=6
Ia=0.36
w=4.375

La2 = 0.0097       %H
Ra2 = 1.26         % ohms
La = La2;       %H
Ra = Ra2;         % ohms

Ea=Va-Ra*Ia
Ke=(Ea)/w
Tm=Ea*Ia/w

Bm2=Tm/w
Kt2=Tm/Ia

% Kt1 = 0.6959;           % N.m/amp
% %Kt1 = 1.3039;           % N.m/amp
%
% Bm1  =0.0573;
% %Bm1 = 0.1073;

Ke2 = Kt2;          %volt/(rad/s)
% Jm1 = 0.005;
```

```
Jm2 = 0.00;

La = La2;
Ra = Ra2;
Kt = Kt2;
Jm = Jm2;
Bm  = Bm2;
Ke = Ke2;

savefile = 'm2';
save(savefile, 'La2', 'Ra2', 'Kt2', 'Jm2', 'Bm2', 'Ke2');

%% Plot dos dados

sim('motormodelagem.slx');

figure('Position', [100, 100, 1400, 600]);
f=28;  %font size
lw=2;  %line width

% Posição experimental
theta(:,2) = theta(:,2) - theta(1,2); % zerando a posicao inicial
theta(:,2) = theta(:,2)*2*pi*10/1024;
subplot(2,1,1);
plot(theta(:,1),theta(:,2),'LineWidth',lw); % plota posição experimental
hold on;
axis([0 30 -5 50])

%Posição simulada

plot(posicaosimulada(:,1),posicaosimulada(:,2),'LineWidth',lw);
legend('Posição experimental','Posição simulada');
title('Motor 2');
ylabel('Posição (radianos)');
xlabel('Tempo (segundos)');
set(gca,'fontsize',f,'LineWidth',lw)
grid on

%Corrente experimental
pontos=50;       %filtro para a corrente
corrente(:,2) = movavg(corrente(:,2),pontos,pontos,0);   % 185 mV / A
corrente(:,2)=(-corrente(:,2)+512)*(5000/(1024*185));
corrente(1:10,2)=corrente(10,2);

subplot(2,1,2);
plot(corrente(:,1),corrente(:,2),'LineWidth',lw);
hold on;

%corrente simulada
plot(correntesimulada(:,1),correntesimulada(:,2),'LineWidth',2*lw);
handle=legend('Corrente experimental','Corrente simulada');
% set(handle,'FontSize',f);
ylabel('Corrente (Ampéres)','FontSize',f);
xlabel('Tempo (segundos)','FontSize',f);
set(gca,'fontsize',f,'LineWidth',lw)
grid on
axis([0 30 -2 2])
```

8.4 motormodelagem.slx



8.5 link1.m

```
close all
clear L
%            theta    d        a     alpha
L(1) = Link([  0      0        0           pi/2], 'modified');
L(2) = Link([  0      0        0.213       0], 'modified');


L(1).m = 2.12669573;
L(2).m = 0.00000000;


%              rx           ry           rz
L(1).r = [0.04197199   0.00000000   0.01758825];
L(2).r = [0.00000000   0.00000000   0.00000000];


correcao=0;

%              Ixx          Iyy          Izz          Ixy          Iyz
Ixz
L(1).I = [0.00408691   0.02275016   0.01610542   0.00000000   0.00000000
-0.00145060];
L(2).I = [0.00000000   0.00000000   0.00000000   0.00000000   0.00000000
0.00000000];


L(1).Jm =  200e-100;
L(2).Jm =  200e-100;

L(1).B =  330e-4;
L(2).B =  0;

L(1).G =  1;
L(2).G =  1;
```

```
qz = [0 0];

scara_link = SerialLink(L, 'name', 'SCARA', 'manufacturer', 'Unimation',
'comment', 'AK&B');


sim('link');

load('01_06_motor1_quedalivre.mat')

zero=169;
noventa=234;
escala=(pi/2)/(zero-noventa);
t1=19.12;

figure('Position', [100, 100, 2000, 600]);
hold on

posicaoquedalivre(:,1)=posicaoquedalivre(:,1)-t1;
posicaoquedalivre(:,2)=escala*(posicaoquedalivre(:,2)-zero);
plot(posicaoquedalivre(:,1),posicaoquedalivre(:,2),'LineWidth',8);
axis([0 6.5 -2.7 0.1]);
f=28;
plot(posicaoquedalivresimulada(:,1),posicaoquedalivresimulada(:,2),'LineW
idth',6);

L(1).r = [0.0554   0.00000000   0.0];
L(1).I = [0   0   0.01963025   0   0  0];
L(1).B =  430e-4;
scara_link = SerialLink(L, 'name', 'SCARA', 'manufacturer', 'Unimation',
'comment', 'AK&B');
sim('link');
plot(posicaoquedalivresimulada(:,1),posicaoquedalivresimulada(:,2),'k-
.','LineWidth',2);

title('Link 1','FontSize',f)
handle=legend('Posição experimental','Posição modelo Solid
Works','Posição modelo analítico','FontSize',f);

ylabel('Posição (radianos)','FontSize',f);
xlabel('Tempo (segundos)','FontSize',f);
set(gca,'fontsize',f,'LineWidth',2)
clear L
```

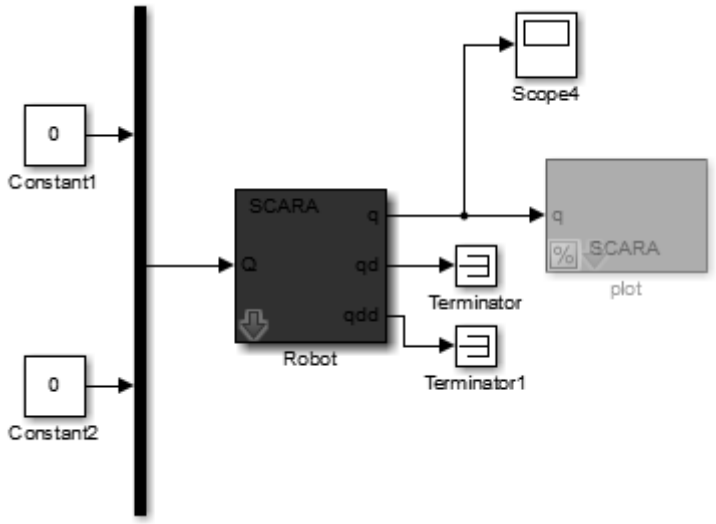
8.6 link2.m

```
close all
clear L
%            theta    d        a    alpha
L(1) = Link([  0      0        0           pi/2], 'modified');
L(2) = Link([  0      0        0.270       0], 'modified');

L(2).m = 0.00000000;
L(1).m = 1.01179436;


%            rx           ry           rz
```

```
L(1).r = [0.13135414   0.00000000  -0.00001150];
L(2).r = [0.00000000   0.00000000   0.00000000];


Izz=1.01179436*(0.3^2+0.06^2)/12
%             Ixx          Iyy          Izz          Ixy          Iyz
Ixz
L(2).I = [0.00000000   0.00000000   0.00000000   0.00000000   0.00000000
0.00000000];
L(1).I = [0.00043250   0.02442707   0.00720124   0.00000000   0.00000002
0.00000006];

L(1).Jm =  200e-100;
L(2).Jm =  200e-100;

L(1).B =  320e-4;
L(2).B =  0;

L(1).G =  1;
L(2).G =  1;

qz = [0 0];

scara_link = SerialLink(L, 'name', 'SCARA', 'manufacturer', 'Unimation',
'comment', 'AK&B');

load('m1');
load('m2');
sim('link');

load('01_06_motor2_quedalivre2.mat')
zero=502;
noventa=576;
escala=(pi/2)/(zero-noventa);
t1=203.125;
figure('Position', [100, 100, 2000, 600]);
hold on
posicaoquedalivre(:,1)=posicaoquedalivre(:,1)-t1;
posicaoquedalivre(:,2)=escala*(posicaoquedalivre(:,2)-zero);


plot(posicaoquedalivre(:,1),posicaoquedalivre(:,2),'LineWidth',8);
axis([0 6.5 -3 0.1]);
f=28;
plot(posicaoquedalivresimulada(:,1),posicaoquedalivresimulada(:,2),'LineW
idth',6);

L(1).I = [0.0   0.0   0.00789199   0.00000000   0.000000   0.0000000];
L(1).r = [0.12   0.00000000  0];
L(1).B =  300e-4;
scara_link = SerialLink(L, 'name', 'SCARA', 'manufacturer', 'Unimation',
'comment', 'AK&B');
sim('link');
plot(posicaoquedalivresimulada(:,1),posicaoquedalivresimulada(:,2),'k-
.','LineWidth',2);

title('Link 2','FontSize',f)
handle=legend('Posição experimental','Posição modelo Solid
Works','Posição modelo analítico','FontSize',f);
```

```
set(handle,'FontSize',f);
ylabel('Posição (radianos)','FontSize',f);
xlabel('Tempo (segundos)','FontSize',f);
set(gca,'fontsize',f,'LineWidth',2)

clear L
```

8.7 link.slx



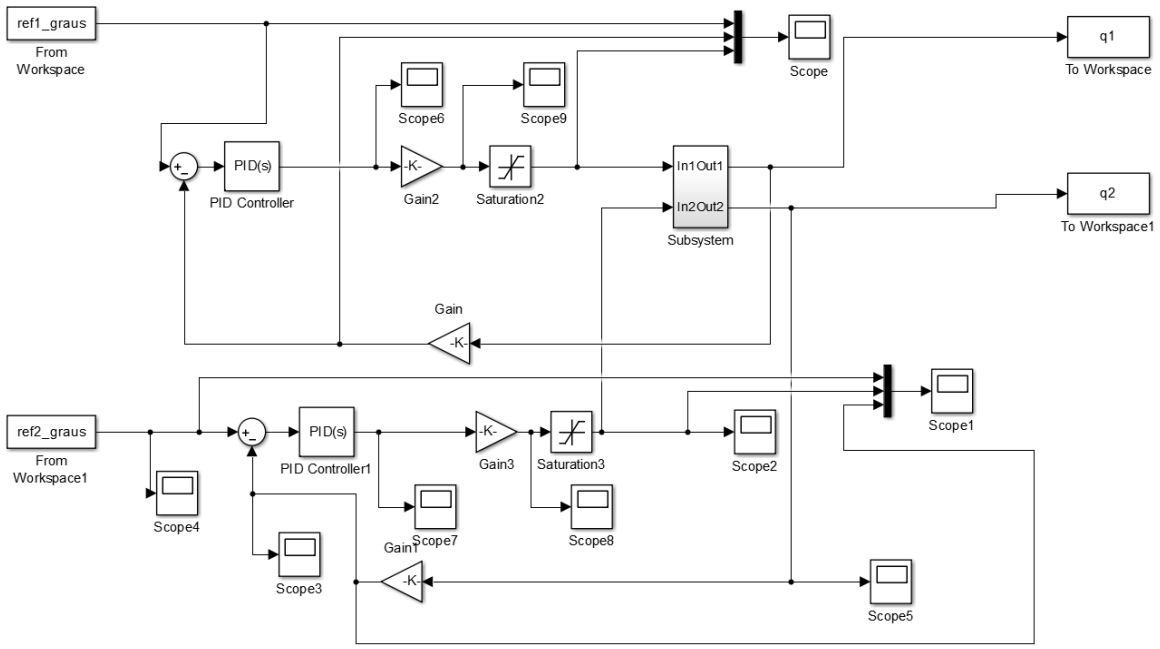
8.8 trajetoria.m

```
qz=[-1.7590    1.7932         0]
M = [1 1 1 0 0 0]

p1=transl(0.200, -0.100, 0)
p2=transl(0.200, 0.100, 0)
p3=transl(0.400, 0.100, 0)
p4=transl(0.400, -0.100, 0)

T0 = ctraj(p1, p1, 25);
T1 = ctraj(p1, p2, 50);
T2 = ctraj(p2, p3, 50);
T3 = ctraj(p3, p4, 50);
T4 = ctraj(p4, p1, 50);
T5 = ctraj(p1, p1, 25);

T = cat(3,T0,T1,T2,T3,T4,T5)

x=squeeze(T(1, 4, :));
y=squeeze(T(2, 4, :));

qr=scara_robot.ikine(T,qz,M)
t=[0.1:0.1:25]'
ref1=[t qr(:,1)]
ref2=[t qr(:,2)]


ref1_graus=ref1

ref1_graus(:,2)=ref1(:,2)*180/pi

ref2_graus=ref2
```

```
ref2_graus(:,2)=ref2(:,2)*180/pi

% scara_robot.plot(qr)
load('ensaioperfeito.mat')
f=28
lw=4

hold on

theta3=theta3(40:end,:)
theta8=theta8(40:end,:)
theta3(:,1)=theta3(:,1)-2
theta8(:,1)=theta8(:,1)-2

n=size(theta3(:,2))

qr_simulado=[theta3(:,2)*pi/180 theta8(:,2)*pi/180 zeros(n(1),1)]

T_simulado = scara_robot.fkine(qr_simulado);
x_simulado=squeeze(T_simulado(1, 4, :));
y_simulado=squeeze(T_simulado(2, 4, :));
plot(y_simulado,x_simulado,'linewidth',lw)
axis([-0.35 0.35 0.1 0.5])
plot(y,x,'k--','linewidth',lw/2)
handle=legend('Trajetória obtida','Trajetória desejada');
% set(handle,'FontSize',f);
ylabel('Posição y (metros)','FontSize',f);
xlabel('Posição x (metros)','FontSize',f);
set(gca,'fontsize',f,'LineWidth',lw)
grid on

figure
ref1_graus(:,1)=ref1_graus(:,1)-2;

subplot(2,1,1);
hold on
plot(ref1_graus(:,1),ref1_graus(:,2),'linewidth',lw)

plot(theta3(:,1),theta3(:,2),'linewidth',lw)
handle=legend('Trajetória desejada','Trajetória obtida');
% set(handle,'FontSize',f);
ylabel('Posição \theta_1 (graus)','FontSize',f);
xlabel('Tempo (segundos)','FontSize',f);
set(gca,'fontsize',f,'LineWidth',lw)
axis([0 23 -120 0])
grid on

ref2_graus(:,1)=ref2_graus(:,1)-2;



subplot(2,1,2);hold on
plot(ref2_graus(:,1),ref2_graus(:,2),'linewidth',lw)
plot(theta8(:,1),theta8(:,2),'linewidth',lw)

handle=legend('Trajetória desejada','Trajetória obtida');
% set(handle,'FontSize',f);
```

```
ylabel('Posição \theta_2 (graus)','FontSize',f);
xlabel('Tempo (segundos)','FontSize',f);
set(gca,'fontsize',f,'LineWidth',lw)
axis([0 23 50 200])
grid on
```

8.9 meu_modelo.slx



Subsystem: