UNIVERSIDADE DE SÃO PAULO
Escola de Engenharia de São Carlos

KIT DIDÁTICO PARA O APRENDIZADO DE CONTROLADORES EM TEMPO REAL UTILIZANDO MATLAB E TOOLBOX REALTIME WORKSHOP

Daniel Vicentini Guimarães

São Carlos, SP

DANIEL VICENTINI GUIMARÃES

KIT DIDÁTICO PARA O APRENDIZADO DE CONTROLADORES EM TEMPO REAL UTILIZANDO MATLAB E TOOLBOX REALTIME WORKSHOP

Trabalho de Conclusão de Curso
apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Eletrônica

ORIENTADOR: Manoel Luís de Aguiar

São Carlos
2011

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

Guimarães, Daniel Vicentini.
G963k Kit didático para aprendizado de controladores em tempo real utilizando *Matlab e toolbox Real Time Workshop*. / Daniel Vicentini Guimarães ; orientador Manuel Luís de Aguiar -- São Carlos, 2011.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2011.

1. *Matlab-RTW*. 2. Controle digital. 3. Motor CC. 4. *Matlab*. 5. *Simulink*. I. Titulo.

FOLHA DE APROVAÇÃO

Nome: Daniel Vicentini Guimarães

Título: "Kit Didático para o Aprendizado de Controladores em Tempo Real Utilizando Matlab e Toolbox Realtime Workshop"

Trabalho de Conclusão de Curso defendido e aprovado em 21/11/2011,

com NOTA 9,0 (NOVE, ZERO), pela comissão julgadora:

Prof. Dr. Rodrigo Andrade Ramos - EESC/USP

MSc. Eduardo Sylvestre Lopes de Oliveira - EESC/USP

Prof. Associado Homero Schiabel
Coordenador da CoC-Engenharia Elétrica
EESC/USP

Dedicatória

Dedico este trabalho aos meus pais, Gaspar Salvador e Heleda, que estavam sempre presentes nestes cinco anos de formação em minhas atitudes de amor e dedicação pelo que faço. Obrigado pelo apoio de minhas decisões. Pelos abraços e beijos que de longe, sentia todos os dias.

Ao meu irmão Pedro Guimarães, por aturar as crises de saudades de meus pais.

Ao meu amigo e companheiro de república Marcelo. Obrigado pelas risadas e por todos os momentos que passamos nesses anos em São Carlos no apartamento 21 do Saint Pièrre.

Aos meus tios, Eliana e Christiano, pelos conselhos.

Aos meus amigos de Santos, Isadora Franco, Maria Beatriz, Pedro Ramos, Roberta Batalha e Déborah Branco. Amigos que dividem minhas lágrimas de alegria e de tristeza.

Agradecimentos

Agradeço ao Prof. Manoel Luís de Aguiar pelas manhãs e tardes que ficamos discutindo o projeto e pelas ajudas na realização dos experimentos.

Aos técnicos de laboratório Alessandro Rodrigo Locatti e César Domingues por abrirem o laboratório todos os dias em que me era disponível.

Ao meu amigo Luciano por sempre me motivar no trabalho.

Aos amigos Gil e Vanessa que estiveram presentes nos momentos mais felizes de minha graduação.

Aos meus veteranos Rodolpho Maciel, Daniel Pelichek e Lucas Capra por me apresentarem a secretaria acadêmica da engenharia elétrica, SA-SEL, e o integra elétrica.

À SA-SEL, a chapa 29 de agosto e a COIE.

Aos meus amigos da eletrônica 07 e amigos de jornadas de estudos Biel, Tots, Gaspar, Castanha, Tocantins, Timão, Débora, Maga, Letícia, Anna e Bahia.

E a todos que me ajudaram direta ou indiretamente.

Sumário

Lista de figuras

Resumo

Este trabalho tem por objetivo a criação de um *kit* didático para futuramente ser utilizado em disciplinas práticas do Laboratório de Controle de Sistemas e Controle Digital, com vistas a auxiliar no aprendizado de controladores digitais em tempo real bem como introduzir a ferramenta *Real-Time Workshop* para Matlab aos alunos. Primeiramente foi desenvolvido um *hardware* para realizar a comunicação com o *software* além de acionar motor CC. Em seguida criou-se e testou-se um conjunto de controladores utilizando a ferramenta *Real-Time Workshop* do Matlab com a finalidade de verificar o desempenho dos controladores no *kit*. O Matlab *Real-Time Workshop* se mostrou uma ótima ferramenta para a realização de controle em tempo real e o hardware desenvolvido apresentou problemas como ruído na aquisição do sinal e atraso no sinal enviado para o motor, no entanto foi possível a realização do controle. É apresentado no trabalho como configurar o Matlab Real-Time workshop em forma *tutorial*.

Palavras chaves: Matlab-RTW; controle digital; motor CC; Matlab; Simulink.

Abstract

The following thesis presents the creation of an educational kit for future use in practical disciplines of the Laboratory of Control Systems and Digital Control, with the objective to assist the learning of digital controllers in real time and to introduce the Real Time Workshop Tool for Matlab to students. First was developed a hardware in order to communicate with the software as well as a DC motor. Then was set up and tested controllers using the Real-Time Workshop tool, from Matlab, in order to verify the performance of the controllers in the kit. Matlab Real-Time Workshop proved to be a great tool for performing real-time control and the hardware that was developed, showed problems like noise in the signal acquisition and delayed signal sent to the motor, however it was possible to perform control. It is also presented in this work how to setting up the Matlab Real-Time Workshop in tutorial form.

Keywords: Matlab-RTW; digital control; DC motor; Matlab; Simulink.

Capítulo 1:

INTRODUÇÃO

Aplicar um controlador em tempo real não é uma tarefa fácil. Além de ser uma atividade complexa existem inúmeras tarefas a serem realizadas previamente para construção de um controlador. Em vista disso, *softwares* como o Labview e o Matlab são extremamente importantes para a realização dessas atividades.

Na Escola de Engenharia de São Carlos (**EESC**), algumas disciplinas como SEL0328 Laboratório de Controle de Sistemas ou SEL0359 Controle Digital têm como um de seus objetivos a aplicação de controladores em tempo real utilizando *softwares* que existem na atualidade. Ambas as disciplinas utilizam Labview para aquisição dos dados e o Matlab, com seu *toolbox* de controle de sistemas, para os cálculos que definem a planta do sistema bem como a definição dos controladores.

Neste projeto de conclusão de curso foi desenvolvido um *kit* didático para utilização do Matlab Real-Time Workshop (RTW), outra ferramenta da atualidade que pode ser utilizada para aquisição dos dados laboratoriais bem como controlar o sistema em tempo real. Essa ferramenta, diferente do Labview, oferece ao usuário a possibilidade de trabalhar em freqüências de amostragem superiores a 1kHz e portanto pode abranger sistemas que apresentam uma resposta mais rápida. Além disso, operando juntamente com o SIMULINK, é integrada ao Matlab. Assim, não há necessidade de importar dados de outro programa, pois todas as informações já podem ser incluídas diretamente em seu ambiente de trabalho - *workspace*. Além disso, possui a vantagem de não haver a necessidade de transformar as funções de transferência em equações de diferenças.

No projeto criou-se um *hardware* para a comunicação entre um motor de corrente contínua e o RTW. Depois de pronto, aplicaram-se alguns tipos de controladores no *kit* para testar a interação do *software* com o *hardware* e simulações utilizando o Matlab foram feitas

para testar a confiabilidade do *hardware* e principalmente desempenho dos controladores propostos.

A intenção do *kit* é ser utilizado em disciplinas dos cursos de controle para ajudar no aprendizado da teoria e de técnicas de controle bem como introduzir o RTW, ferramenta poderosa do Matlab, aos alunos de graduação.

Este trabalho de conclusão de curso (TCC) está relacionado a outros TCC's, são os do Newton, A. C. [1] e do Corder, R. M. [2]. Além disso, as fontes bibliográficas foram obtidas através da pesquisa de como configurar o Matlab [3] [4], estudo das características dos componentes utilizado para a criação do hardware [5] [6] e para a realização do controle e o equacionamento de alguns problemas utilizou-se livros e apostilas com teoria de controle, controle digital e fundamentos de análise de circuito elétrico [7] [8] [9] [10].

A monografia divide-se em cinco capítulos. No Capítulo 2, seguinte a esta introdução, apresenta-se as características do Matlab RTW e suas configurações básicas em modo tutorial, inspirados na monografia de Newton, A. C. [1], com algumas melhorias observadas na realização deste projeto. No Capítulo 3 está descrito o hardware do kit, seus componentes e suas características. O Capítulo 4 mostra os tipos de controladores que foram aplicados e seus resultados. Por fim, conclui-se o trabalho no Capítulo 5.

Capítulo 2:

MATLAB REAL-TIME WORKSHOP

2.1.Visão Geral

As disciplinas práticas do Laboratório de Controle de Sistemas e Controle Digital, usualmente trabalham com dois softwares: Labview e Matlab. Com o auxílio do Labview é feita a aquisição de dados de um sistema real e, os quais depois de serem obtidos e armazenados em arquivos são importados no ambiente do Matlab para o cálculo, verificação, comparação e/ou geração da ação de controle para o sistema. Nem sempre é fácil importar os dados entre os programas e estes operam os dados de formas diferentes. Além disso, o Labview tem uma freqüência de amostragem limitada de 1kHz [1]. Utilizando o Matlab RTW o usuário soluciona os dois problemas. Além disso não é necessário transformar a função de transferência a uma equação de diferenças.

O RTW é um *toolbox* do Matlab, extensão do Simulink. Fazendo um modelo em diagrama de blocos, gera-se e compilam-se programas executáveis baseados em linguagem C de programação [2][3]. Com uma placa de entrada e saída de dados, o *toolbox* ainda pode enviar e receber sinais em tempo real fazendo assim possível a realização de um controlador de forma simples e eficiente.

Portanto é uma ferramenta totalmente integrada com o Matlab e fácil de ser utilizada.[2]

2.2.Configuração do Matlab RTW

Primeiramente deve-se ter certeza se a versão do Matlab utilizada suporta o RTW. A versão utilizada no projeto foi a 2008b. Para o correto funcionamento do RTW é necessário que o processador dê prioridades às instruções dadas pelo Matlab, para isso é necessária a instalação do *Real-Time kernel*. Deve-se digitar na tela de comando do Matlab *rtwintgt–install* e prosseguir com a instalação, pressionando e*nter*. Se a instalação for bem sucedida, aparecerá na tela a mensagem "The Real-Time Windows Target kernel has been successfully installed". Para verificar se a instalação foi realizada pode-se digitar *rtwho* e aparecerá na tela informações referente à versão instalada bem como o *driver* da placa de I/O que foi reconhecido e instalado juntamente [4]. Para o Windows vista e o Windows 7 é necessário desabilitar alguns níveis de

segurança. Como se utilizou o Windows XP não houve problemas em relação a segurança do sistema.

Com a instalação do *kernel* pode-se iniciar o processo de configuração do ambiente *Simulink*. Quando se cria um novo modelo, deve-se optar pelo modo *external* como indicado na Figura 1. Feito isso, o usuário deve configurar o *Matlab RTW* para o tipo de aplicação que ele utilizará. Para tal deve-se entrar no menu de configuração dos parâmetros de simulação (menu *Simulation*, *Configuration* ou *Ctrl+E*).

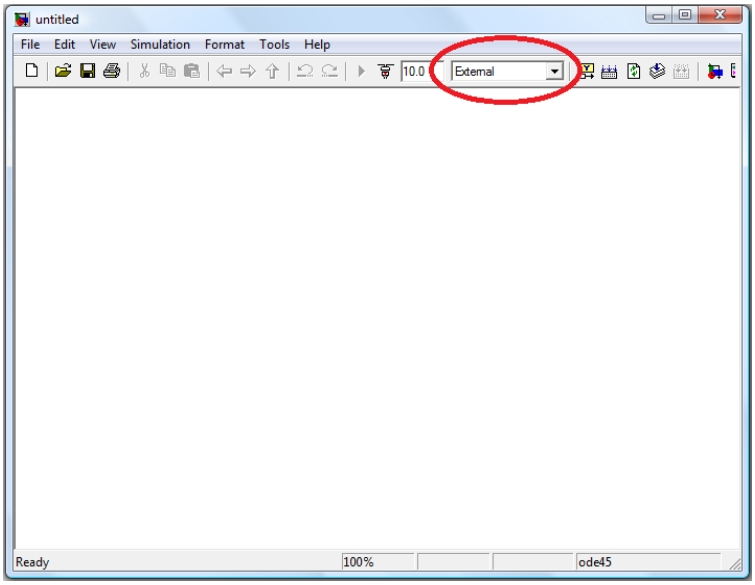


Figura 1 - Configuração do Simulink para modo External.

No menu de configuração de parâmetros, primeiramente deve-se configurar o tipo de hardware e o sistema de instruções que o RTW vai operar. Clica-se em *Hardware Implementation* e ajustam-se as variáveis como indicado na Figura 2. Nos campos "*device vendor*" e "*device type*" seleciona-se *Generic* e *32-bit x86 compatible* respectivamente. Deixa-se a opção *none* selecionada.

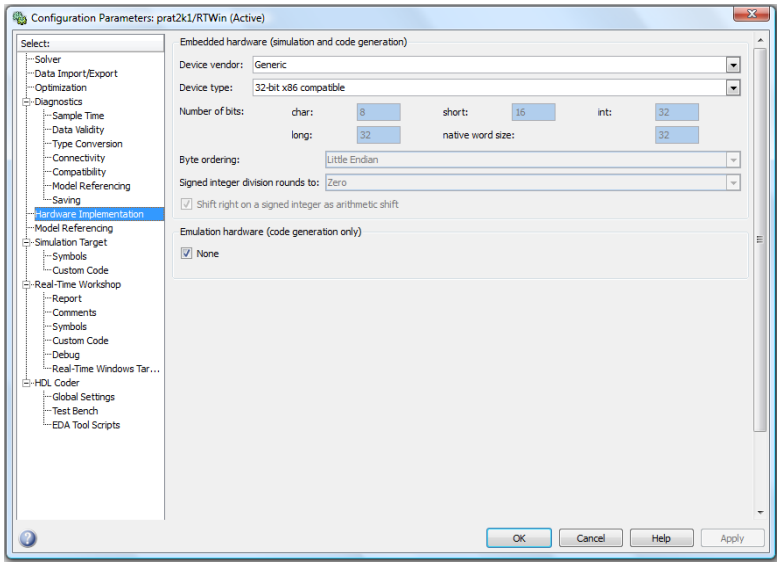


Figura 2 - Configurações de parâmetros do RTW, Hardware Implementation.

O próximo passo é a configuração do *target*. Com ele é possível optar por diversos tipos de geração e compilação de código. Cada um está direcionado a utilização do usuário. Por exemplo, podem-se fazer programas com opções dinâmicas de memória ou outros que serão executados através de pacotes de funções ou simulação de Monte Carlo salvando cada passo em um arquivo de texto. Na aplicação dirigida ao projeto do kit será utilizado *Real-Time Windows Target* que habilita o computador para operar o *Simulink* em tempo real [4]. Para isso, no mesmo *menu* de configurações de parâmetros deve-se acessar o submenu *Real-Time workshop* e configura-se os campos da maneira disposta pela Figura 3.

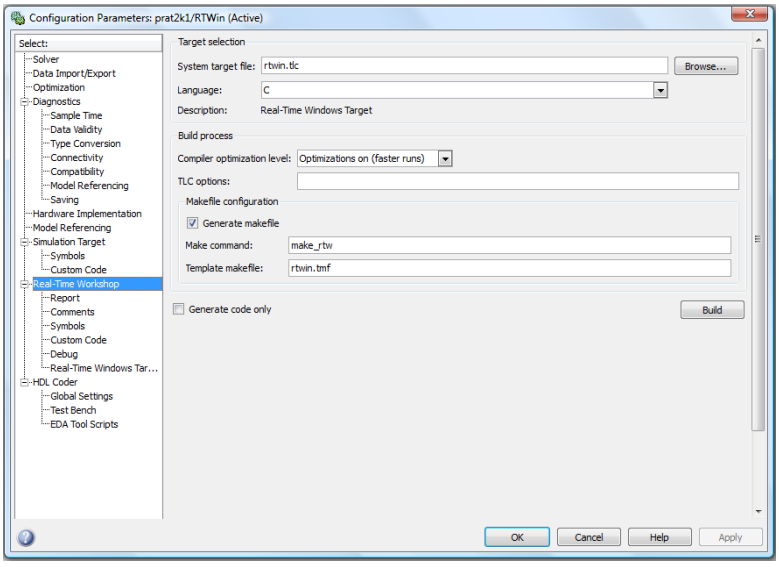


Figura 3 - Configurações de parâmetros do RTW, target.

O padrão da ferramenta é trabalhar sempre com os últimos 1000 pontos amostrados. Para alguns casos, esse número de pontos não é suficiente. No projeto em questão configurou-se para 3000 pontos. Para fazer a devida modificação, ainda no menu de configuração de

parâmetros, no submenu *Data Import/Export*, o usuário modifica o campo "*Limit data points to last"* para o número de pontos escolhidos, Figura 4. Modificado todos os parâmetros necessários do RTW, confirma-se as modificações do menu com o botão *Apply* e *Ok*, depois deve-se entrar no menu *External Mode Control Painel* (menu *Tools*, *External Mode Control Panel*). Abrirá a janela da Figura 5. Clica-se em *Signal & Triggering* e abrirá a janela da Figura 6. Nessa janela pode-se fazer a alteração do valor padrão de pontos de 1000 para o escolhido pelo usuário.

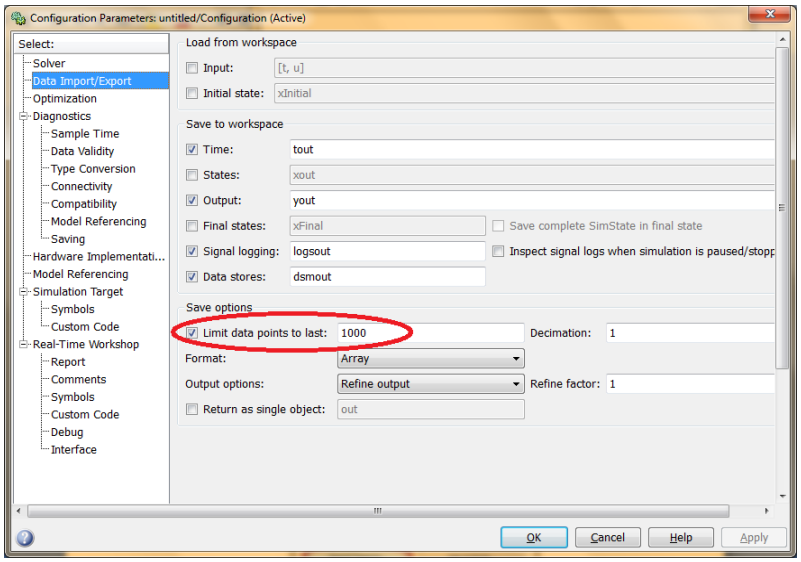


Figura 4 - Configurações de parâmetros do RTW, data import/export.

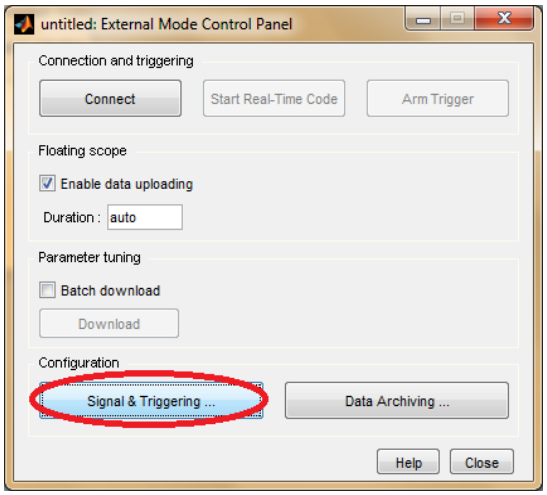


Figura 5 - External Mode Control Panel

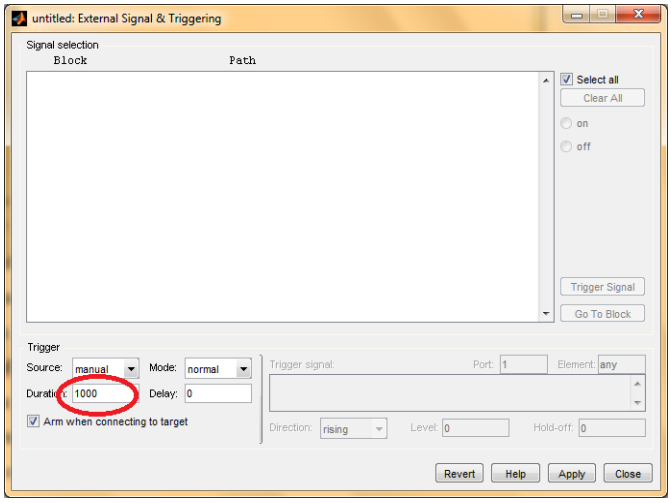


Figura 6 - External Signal & Triggering, número de pontos

Também é importante saber configurar as portas de entrada e saída bem como os *timers* e os contadores da placa. Na biblioteca do Simulink, nos blocos do RTW, observa-se a subdivisão do *Real-Time Windows Target***,** nele encontra-se todos os blocos referentes a comunicação com a placa de entrada e saída de dados. Na Figura 7 estão dispostas as propriedades do bloco de uma entrada analógica (*analog input*). Observa-se que a placa *National Instruments AT-MO-16E-10* está selecionada – placa utilizada no projeto. Caso não haja nenhuma para selecionar, deve-se instalar uma nova clicando no botão *new board*. No menu ainda pode-se escolher por exemplo o tempo de amostragem, o canal de entrada da porta e os limites da entrada. Todos os blocos, de entrada e de saída, devem apresentar o mesmo valor de freqüência de amostragem para não haver erros na compilação dos arquivos.

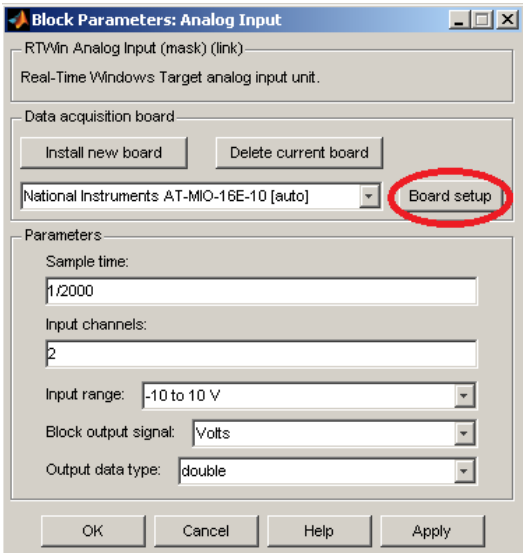


Figura 7 - Configuração de entrada analógica (analog input)

Ao entrar no menu *board setup*, sinalizado na Figura 7, o usuário pode modificar algumas propriedades da placa. Na Figura 8, configurações da placa, observa-se que as quatro primeiras portas digitais estão configuradas como saídas, as quatro últimas estão como entradas, o *timer 1* está configurado para contador e o *timer 2* como gerador de freqüência para criação de um sinal com modulação de largura de pulso (do inglês *pulse-width modlation*, PWM). É importante notar que a conexão A/D está configurada para o padrão *NRSE* evitando erros futuros.

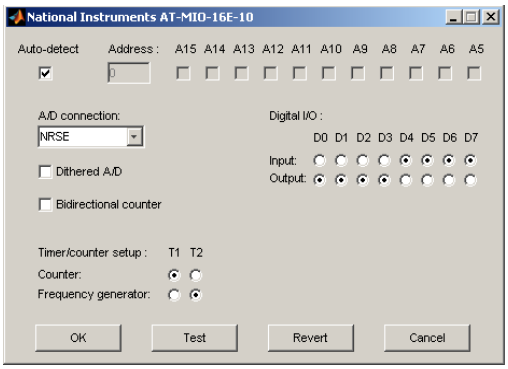


Figura 8 - Configurações da placa, board setup

Depois de tudo configurado pode-se montar um diagrama de blocos do sistema. Com o diagrama de blocos pronto para funcionar, o usuário deve montar os arquivos para o funcionamento do código através do comando *build* (menu *Tools*, *Real-Time Workshop*, *build* ou *Ctrl+B*). Agora o usuário pode começar o controle clicando no botão *connect to target* e depois clicar no botão *start real-time code***,** Figura 9**,** que ativará o código montado pelo comando *build*.

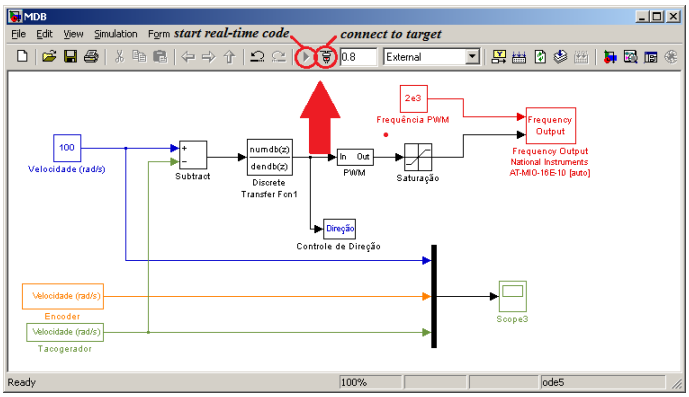


Figura 9 - Iniciar controle

2.3.Observações e dificuldades

Para criar alguns controladores, é interessante que o tempo de amostragem na porta de entrada seja diferente da saída. A ferramenta não aceita esse tipo de configuração. Caso o usuário programe as portas com diferentes freqüências de amostragem, ao compilar o programa haverá erro. Enquanto não selecionar um mesmo tempo de amostragem para todas as portas o erro persistirá.

Depois que o usuário conecta com o alvo (*connect to target*) e pressiona o botão para iniciar o código (*start real-time code*) por vezes aparece um erro inesperado que pode inclusive reiniciar o computador sem aviso prévio. Portanto é importantíssimo que o usuário sempre salve seu trabalho antes de iniciar o programa.

Capítulo 3:

DESCRIÇÃO DO *HARDWARE*

O *Hardware* do projeto foi criado para estabelecer a comunicação entre a placa de comunicação e aquisição de dados com o módulo, para gerar potência para o motor CC além de fazer o condicionamento de sinais (corrente, tacogerador, encoder e isoladores ópticos).

Pode-se observar no projeto, disposto no anexo desta monografia, que este contém três pontos de referência de tensão "Terra" distintos. São eles:

(i) "Terra" digital

Esta referência está relacionada à placa de aquisição de dados. Os componentes envolvidos com essa referência são aqueles que geram os comandos eletrônicos ao circuito de eletrônica de potência, determinando a direção de rotação, a potência do motor bem como faz sua contagem de giro.

(ii) "Terra" do circuito de potência

Referente ao módulo de eletrônica de potência do projeto. Todos os sinais enviados ao L298, a ponte H que controla a potência do motor, estão nessa referência.

(iii) "Terra" da bateria

Referência da bateria que fornece corrente necessária ao funcionamento do motor.

Separar os terras é necessário para a segurança da placa de aquisição de dados. Estando os subsistemas, potência e eletrônica, isolados, evita-se danificar a placa instalada no computador, elemento mais caro e delicado do projeto.

Neste capítulo estão descritos os elementos que foram utilizados para a construção do *hardware* bem como suas principais características.

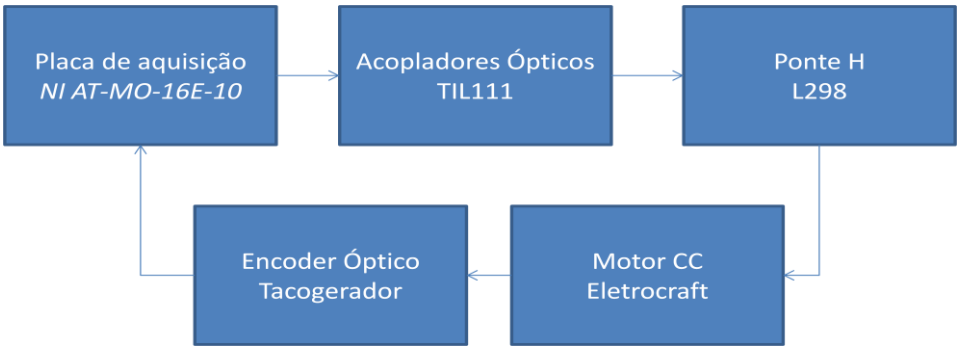


Figura 10 - Esquemático do Hardware

3.1. A placa de aquisição de sinais

Para estabelecer a comunicação com o *software* Matlab, está instalada no computador a placa de entrada e saída de dados da *National Instruments*, *NI AT-MO-16E-10*. Entre suas principais características esta possui:

- Entrada analógica limitada de ±10V;
- 8 canais de entrada ou saída digitais;
- Duas entradas configuráveis como contador ou timer de 24 bits.

No projeto utiliza-se uma saída digital, um *timer*, uma saída de +5V, um contador e a referência digital *DGND* da placa.

A saída digital utilizada tem como objetivo fornecer o sentido de rotação do motor. Pode acionar o motor no sentido horário (+5V) ou anti-horário (0V). A saída está ligada diretamente a um acoplador óptico e retorna ao *DGND*, contido na própria placa de dados.

O *timer* comanda a potência fornecida ao motor, operando como um PWM, que será abordado na seção 3.2 referente a potência. Liga-se o sinal a um acoplador óptico fechando o circuito na referência digital.

A saída de +5V é utilizada no *encoder* óptico, que gera a cada volta 1024 pulsos. Esses pulsos por sua vez vão para o contador da placa.

3.2. Acoplador óptico

Como acopladores ópticos, utilizaram-se o circuito integrado (**CI**) TIL111, Figura 11.

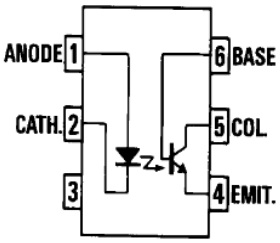


Figura 11 - Esquemático Til111 [5]

O componente é usado para isolar opticamente os circuitos de eletrônica com o de potência para assim proteger a placa de aquisição de dados. Para toda entrada ou saída de dados que se comunica com o circuito de potência, utilizou-se um TIL111.

O circuito integrado funciona como uma chave. Sempre que existe uma corrente no intervalo de 15mA < I < 30mA no foto-diodo, este emite luz que excita a base do transistor [5]. Assim, funcionando como uma chave, o transistor libera corrente ao circuito de controle de potência.

Todos os TIL111 estão voltados com o diodo para a placa de aquisição e transmissão de sinal e o transistor para o circuito de potência. Ambos foram polarizados para o modo ligado com uma corrente de diodo de 16mA e uma corrente de emissor do transistor de 1mA, assim atendendo respectivamente o funcionamento do fotodiodo e a necessidade do circuito de potência. O diodo foi aterrado no "terra" digital da placa de aquisição de dados enquanto o transistor foi aterrado com circuito de potência.

É importante notar que um deles atua como uma chave porta inversora de sinal. A medida é necessária para a criação de sinais complementares para comandar a direção do motor. O estudo deste sinal será mais detalhado na seção 3.2.

3.3.Circuito de potência e o L298

Para realizar o controle do motor usando PWM, não basta o sinal da placa de aquisição de dados. A corrente necessária para o motor não é suficiente. Além da falta de corrente, ainda é necessária a inversão de polaridade sobre o motor a fim de garantir os dois sentidos de rotação.

Para solucionar esses dois problemas utilizou-se um circuito em ponte H, disposto na Figura 12.

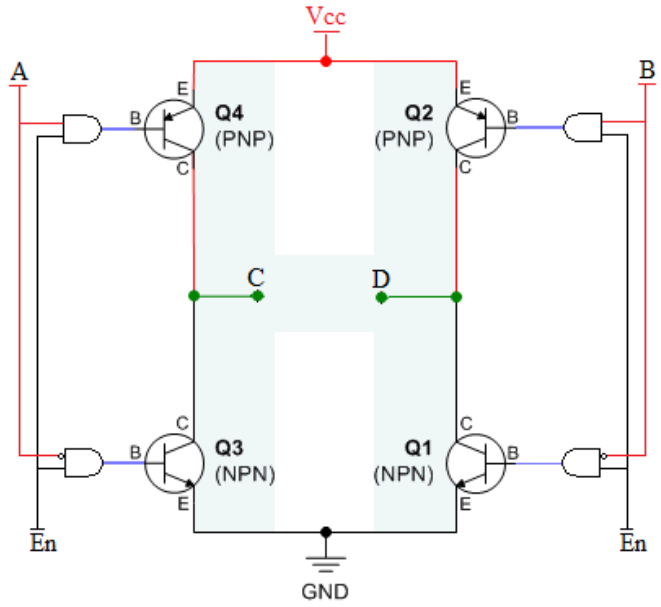


Figura 12 – Circuito ponte H

Este circuito funciona de acordo com a tabela da verdade disposta na tabela 1.

Tabela 1 - Tabela verdade

En	A	B	C	D
1	0	0	0	0
1	0	1	0	Vs
1	1	0	Vs	0
1	1	1	Roda-livre	
0	X	X	Roda-livre	

No caso em que A=B, C e D estão em curto. Por outro lado, quando $A = \bar{B}$, C ou D recebem Vs. Com isso, pode-se ligar os pólos do motor nos terminais C e D. Assim, quando En=0, o motor está em roda-livre e quando $A = \bar{B}$ o motor gira em sentido horário ou anti-horário, dependendo dos valores adotados para A e B. Perceba que a situação $A = B$ nunca ocorre devido a construção do circuito eletrônico.

Quando o motor opera em roda-livre, o próprio gera uma tensão que poderia danificar a ponte H. Portanto, é importantíssimo adicionar diodos *schottkys* para proteção do sistema como disposto na Figura 13.

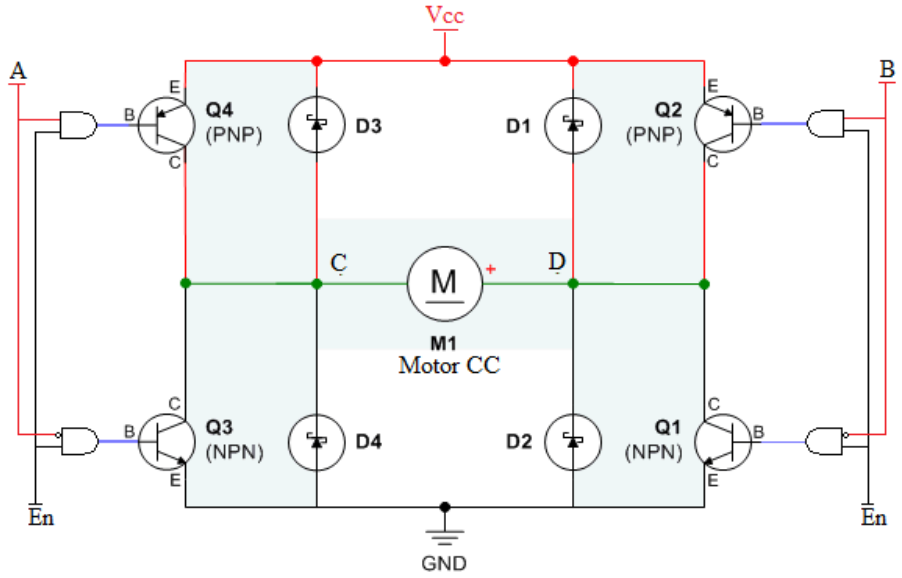


Figura 13 - Ponte H com motor e diodos schottkys

Dessa forma, para o circuito fornecer potência ao motor, deve conter os sinais A e B complementares um ao outro e o *En* = 1. Com isso, os sinais A e B ditam a direção ao motor enquanto o *En* limita sua potência.

Portanto faz-se dois sinais digitais na placa, um de sentido de rotação do motor e outro de PWM. O sentido de direção liga-se na lógica nos sinais A e B (sendo que $B = \bar{A}$) enquanto o sinal de PWM é ligado ao sinal *En*.

O circuito integrado L298, da *STMicroelectronics*, possui duas pontes H como mostra a Figura 14. Para fornecer a corrente necessária ao motor, e não saturar o sistema, as pontes do **CI** foram ligadas em paralelo. Gera-se o sentido de rotação utilizando-se do sinal fornecido pelo TIL111 e seu complementar, também gerado por um acoplador óptico.

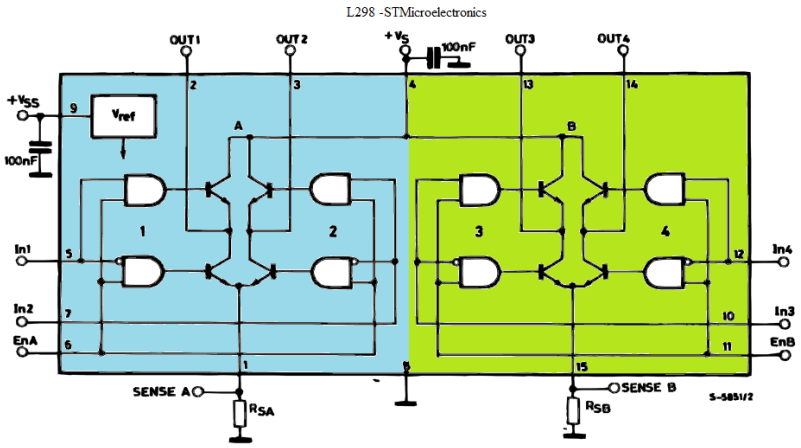


Figura 14 - Esquemático do circuito L298 [6]

Para limitar a potência do motor, e para realizar, portanto, o controle, conecta-se o sinal do TIL111 proveniente do PWM ao sinal de *enable* do L298. Dessa forma consegue-se controlar a tensão média aplicada no motor, como exemplificado na Figura 15.

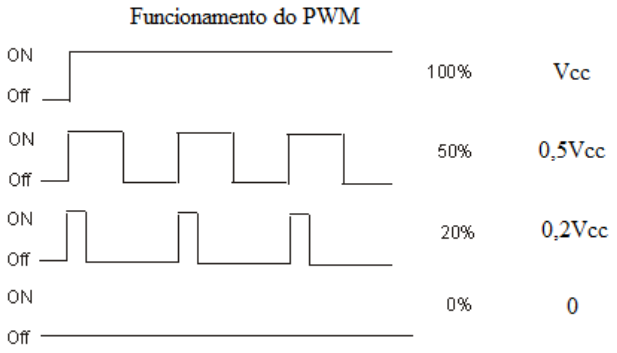


Figura 15 - Funcionamento do PWM

No circuito de potência usou-se uma tensão +5V no CI L298 e para a polarização dos transistores da porta inversora e dos acopladores ópticos. Para gerar essa tensão usou-se um limitador de tensão de +5V, o CI LM7805.

Para o *feedback* do circuito de potência utilizou-se um *encoder* óptico ligado ao motor. A cada volta completada do motor o *encoder* gera 1024 pulsos. Esses pulsos passam por um acoplador óptico e são contados pela placa de aquisição de dados. Com um simples cálculo é possível verificar a velocidade em que o motor se encontra.

Todas as referências do circuito de potência foram ligadas no terra da fonte que alimenta o motor, assim protegendo o circuito eletrônico.

3.4.Motor CC

No laboratório de controle do departamento de engenharia elétrica estão disponíveis servomotores de ímã permanente de dois fabricantes: Faulhaber e Eletro-Craft. No projeto em questão foi utilizado o motor da Eletro-Craft, um servo motor de ímã permanente que opera no máximo com 60V.

Realizaram-se alguns testes em bancada:

(i) Influencia do L298 no motor

(ii) Atuação do PWM

3.4.1. Influência do L298 no motor

Fizeram-se dois testes para se aferir a influência do L298 no acionamento do motor.

Primeiramente, chaveou-se manualmente o motor utilizando uma bateria de carro e obteve-se a resposta degrau da Figura 16. Pode-se notar que no início do sinal há uma insignificante oscilação na tensão de entrada devido a um pico de corrente solicitado a fonte de alimentação.

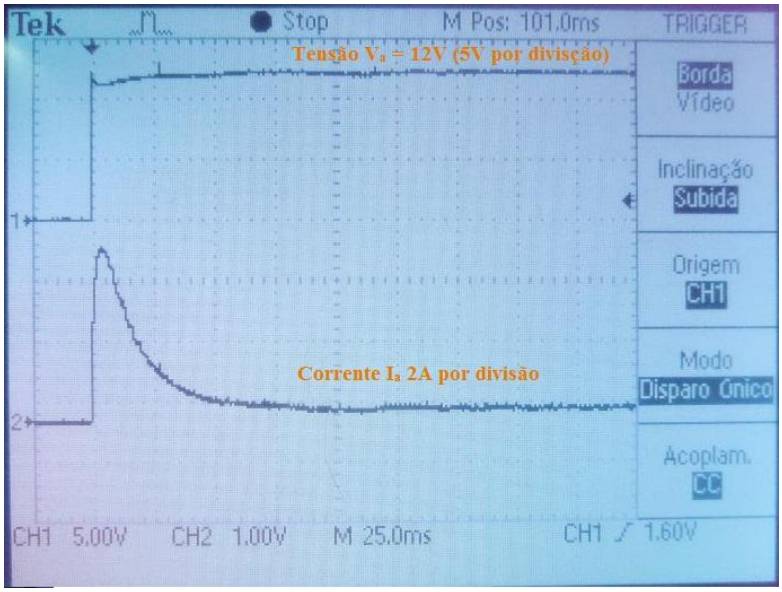


Figura 16 - Motor sem a utilização do L298

A Figura 17 mostra a curva da tensão de entrada do motor utilizando o L298. Para simular o degrau, colocou-se um PWM com 100%. Observa-se que nos primeiros 5ms aparece uma não linearidade devido a um atraso do sinal. Isso ocorre devido ao sistema lógico de controle do CI. Outro fator observado é a queda de tensão que existe no L298. O circuito de potência, alimentado com 12V, perde aproximadamente 2V nos transistores BJT's da ponte H.

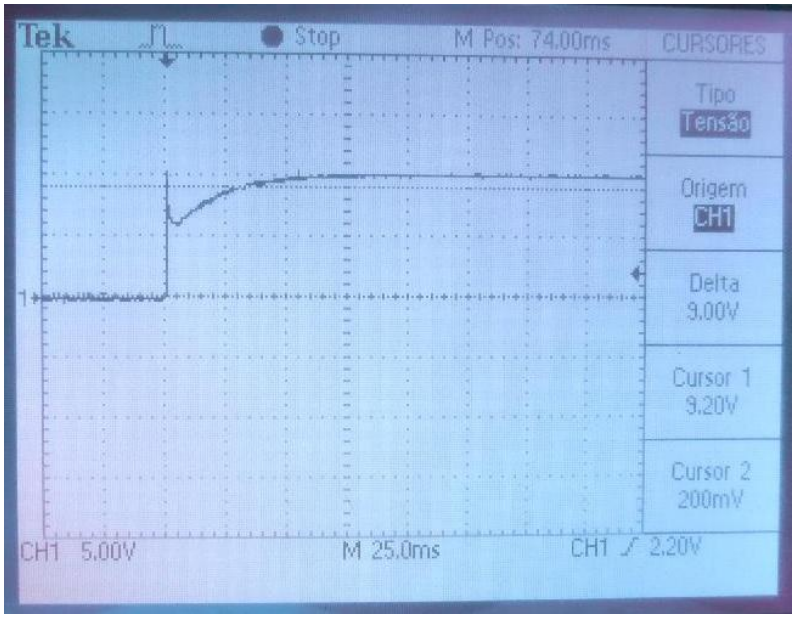


Figura 17 - Motor com a utilização do L298

3.4.2 Teste de PWM

No teste de PWM, Figura 18, percebeu-se outra característica do motor. Quando o PWM está em T_{off}, o motor gira em roda-livre e gera-se tensão em sua armadura diferente de zero. Esse é o motivo da elevação da curva para o T_{off} em que se esperava uma tensão nula.

Figura 18 - Atuação de PWM no motor CC

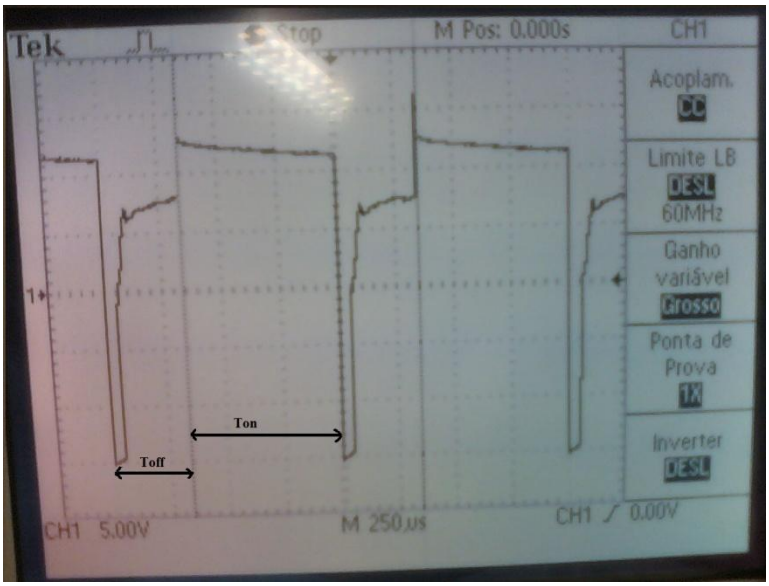


Figura 19 - Atuação de PWM no motor CC, com zoom

3.5. Alguns problemas e dificuldades encontradas

3.5.1. Montagem do Protoboard

Para fazer a montagem no circuito no *protoboard*, o encapsulamento Multiwat 15 V do L298 [6] apresentou pinos frágeis e com uma numeração não convencional. Teve-se que ter muito cuidado para sua insersão no circuito bem como, para não haver confusão na contagem dos pinos, escolheu-se fios de cores semelhantes para as portas que haveria necessidade em ligar em paralelo. Observações que parecem óbvias, mas no processo de montagem são extremamente importantes seguir.

A folha de dados do acoplador óptico, TIL111, apresenta uma corrente típica para o diodo foto emissor de 5mA. Primeiramente o diodo foi polarizado com essa corrente. Não havendo chaveamento do transistor, aumentou-se a corrente para a faixa indicada na seção 3.4. Com isso, o funcionamento da chave foi corrigido.

3.5.2.Aquisição da velocidade

Para o projeto estavam disponíveis dois tipos de motores, e para cada motor havia um tacogerador para a verificação da velocidade. Quando o motor é alimentado de 0 a 12V a tensão do tacogerador do motor Eletrocraft varia de 0 a 20V e para o motor da Faulhaber, de 0 a 3V. A placa de aquisição de sinal suporta até 10V, portanto em um dos casos havia a necessidade de atenuar o sinal e no outro ampliar para uma melhor visualização do resultado. Sendo assim, haveria a necessidade da criação de dois tipos de condicionadores de sinal.

Um dos fatores mais agravantes para a aquisição analógica do sinal que quantiza a velocidade é a segurança da placa de aquisição de dados. Pesquisaram-se alguns tipos de **CI's** para esse fim e o que estava disponível para ser utilizado - proteção por isolação galvânica **ISO122 -** haveria a necessidade de outra fonte de alimentação, simétrica, além da fonte utilizada pelo motor. Dessa forma, não havendo disponível em todas as bancadas, seria inviável a utilização deste componente no *hardware* desenvolvido.

Mesmo não havendo a segurança embutida no *hardware*, esses condicionamentos foram implementados em vistas de que há como fazer uma segurança externa ao *hardware* usando-se os módulos 5B41 disponíveis no laboratório

Deve-se fazer a observação que um *jumper* seleciona qual condicionamento de sinal será utilizado. Deve-se escolher se deseja amplificar ou atenuar o sinal.

Capítulo 4:

CONTROLADORES DIGITAIS EM TEMPO REAL - INTRODUÇÃO TEÓRICA, SIMULAÇÃO E RESULTADOS

4.1.Introdução teórica

Controlar um sistema é algo natural. Encontramos em diversas partes da natureza mecanismos de controle e pode-se observar que a maior parte deles envolve um mecanismo de realimentação. Observa-se, por exemplo, num simples ato de abrir uma porta: aproxima-se da porta, através da visão sabe-se que está perto, através do tato toca-se a maçaneta, gira-se, e através do som e do tato novamente sabe-se que pode de uma vez por todas abri-la.

Para o homem, os sentidos - visão, audição, tato, paladar e olfato - são ótimos sensores de *feedback* para suas ações. Tratando-se de sistemas eletromecânicos, que é o caso do projeto, é necessário a implementação de sensores e atuadores para que assim o sistema consiga ter percepção e reação, como um mecanismo natural.

No projeto com controle de velocidade rotacional do motor CC usou-se para *feedback* do sistema um *encoder* óptico. A cada volta do motor o *encoder* emite 1024 pulsos e sabendo a quantidade de pulsos emitidos em um segundo é possível obter a freqüência que o motor está girando, e portanto, sua velocidade. Para realizar esse cálculo o *encoder* foi ligado no contador da placa de aquisição de sinais que amostra o sinal analógico.

O sinal do *encoder*, agora binário, passa por um algoritmo de controle (como por exemplo, somatórias sucessivas, derivação ou até mesmo multiplicação por constantes) retornando ao sistema à saída do sistema. Para transformar o sinal digital para analógico, é necessário fazer uma transformação inversa da amostragem.

Dado que o sistema faz uma amostragem do sinal, o processamento numérico dos dados, e depois uma conversão para o sinal operar no sistema analógico o projeto em questão é um controlador digital. Na Figura 20 pode-se observar como a informação é processada para realizar-se um controle digital. [7]

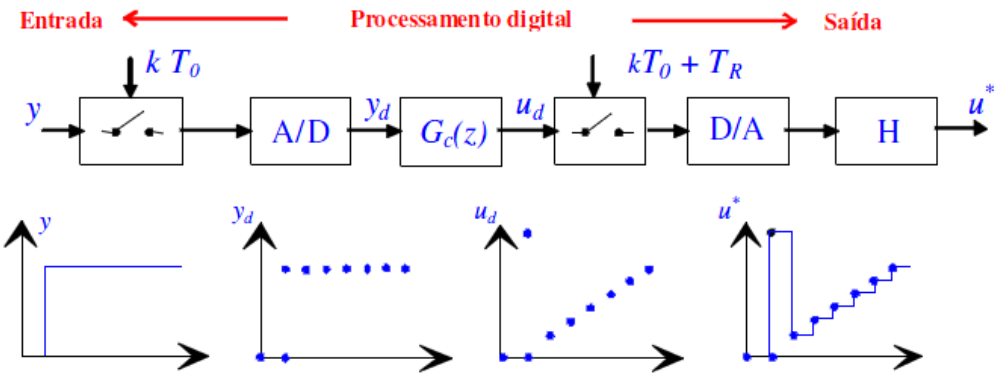


Figura 20 - Processamento de informação para controle digital [7]

De uma forma geral, o sistema discutido anteriormente pode ser representado pelo diagrama de blocos da Figura 21, um esquema geral para uma malha de controle digital com realimentação.

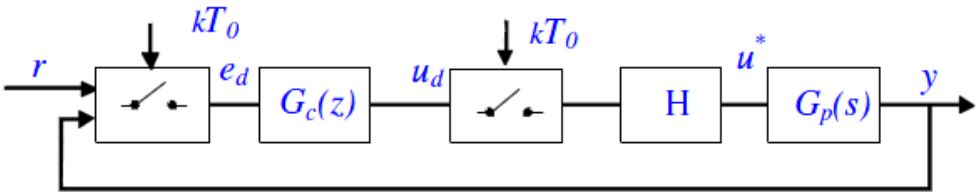


Figura 21 - Diagrama de blocos de controle digital [7]

Usaram-se três tipos de controladores no *kit* didático desenvolvido:

- PID discreto
- Realimentação por Espaço de Estados
- Controlador Dead-Beat.

Nas próximas seções serão feitas breves discussões a respeito dos controladores e como foram aplicados no *kit* didático projetado.

4.1.1.Controlador PID discreto

O controlador proporcional, integrativo e derivativo, assim chamado *PID*,recebe este nome justamente pois carrega as três operações em sua função de transferência, como vista na equação 1, representação continua de um controlador *PID*. [7]

$$u(t) = K\left[e(t) + \frac{1}{T_1}\int e(\tau)d\tau + T_D\frac{d}{dt}e(t)\right] \quad (1)$$

A popularidade do controlador se dá ao fato de apresentar um desempenho robusto sobre uma grande faixa de condições operacionais e em parte à sua simplicidade de configuração. [8]

Admitindo-se um tempo de amostragem pequeno, e utilizando o método retangular de integração, o controlador pode ser aproximado através da equação 2 [7]. Nessa equação observa-se que o termo proporcional atua diretamente no erro atual, o termo integrativo sobre a soma dos erros atual e passados e o termo derivativo na taxa de variação do erro.

$$u(k) = K\left[\, e(k) + \frac{T_0}{T_1}\sum_{v=0}^{k-1} e(v) + \frac{T_D}{T_0}\left(e(k) - e(k-1)\right)\right] \tag{2}$$

Da equação, chega-se a forma recursiva dada pela equação 3.

$$u(k) - \, u(k-1) = q_0 e(k) + \, q_1 e(k-1) + \, q_2 e(k-2) \tag{3}$$

Com os coeficientes dados por:

$$q_0 = K\left(1 + \frac{T_D}{T_0}\right) \tag{4}$$

$$q_1 = -K\left(1 + 2\frac{T_D}{T_0} - \frac{T_0}{T_1}\right) \tag{5}$$

$$q_2 = K\frac{T_D}{T_0} \tag{6}$$

Assim, tem-se a função de transferência discreta para o controlador dado pela equação 7.

$$G_R(z) = G_{PID}(z) = \frac{U(z)}{E(z)} = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}} \tag{7}$$

Se fosse adotado o método trapezoidal de aproximação, a formulação encontrada seria diferente [7].

Para o projeto em questão, o controlador PID-Digital desejado foi investigado e projetado diretamente no domínio da variável "z" com auxílio da discretização da planta (motor CC) e da ferramenta RLTOOL do Matlab.

Por vezes, por limitação física do problema, o fator derivativo não é realizável. A situação será discutida na seção 4.3.1 na qual são obtidos os parâmetros PID do controlador.

4.1.2. Controlador *Dead-Beat*

Os controladores *Dead-Beat* são utilizados quase que exclusivamente de forma discreta. Visa forçar a resposta do sistema ao valor da entrada com um atraso puro, dessa forma garantindo assim uma resposta extremamente rápida ao sistema.[7]

Deseja-se que:

$$\frac{Y(z)}{r(z)} = z^{-m} \tag{8}$$

Para um processo de ordem "m", depois de "m" períodos de amostragem, o mesmo terá um resposta igual a entrada.

Este tipo de controlador tem a função de transferência típica como sendo:

$$G_r(z) = \frac{p_1 z^{m-1} + p_2 z^{m-2} + \ldots + p_m}{z^m} \tag{9}$$

Em que:

$$p_i = b_i q_0 \; ; \; i = 1,2,\ldots,m \; ; \; q_0 = \frac{1}{\sum_{i=1}^{m} b_i} \tag{10}$$

Nas relações das equações 10, tem-se que b_i são os coeficientes do denominador da função de transferência discretizada do sistema a ser controlado com i variando de 1 até "m".

4.1.3. Controle por realimentação de espaço de estados

Dado um processo por variáveis de espaço de estado discreto V.E., tal que,

$$\begin{cases} x(k+1) = \boldsymbol{F}x(k) + \boldsymbol{H}u(k) \\ y(k) = \boldsymbol{c}x(k) \end{cases} \tag{11}$$

Com o diagrama de blocos representado pela Figura 22.

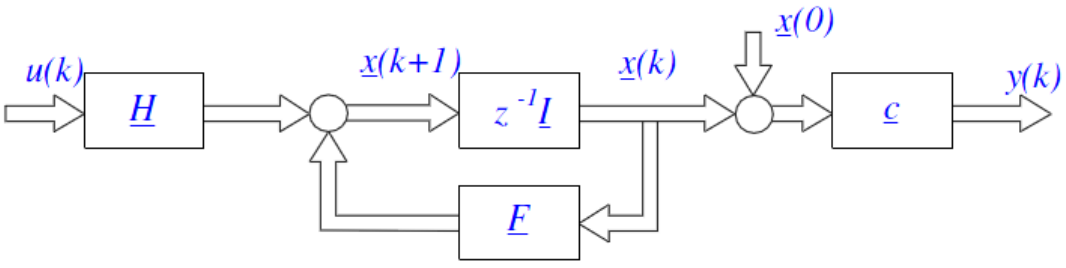


Figura 22 - Diagrama de blocos, variáveis de estados discretas [7]

Deseja-se criar uma realimentação dos estados $\underline{x}(k)$ do processo que faça com que a saída *y(k)* atenda algum critério do projeto. Para isso, cria-se uma matriz $\underline{K}$, que produza uma combinação linear dos estados de $\underline{x}(k)$, fornecendo assim para a entrada um sinal de realimentação. Tal modelo está disposto na Figura 23.

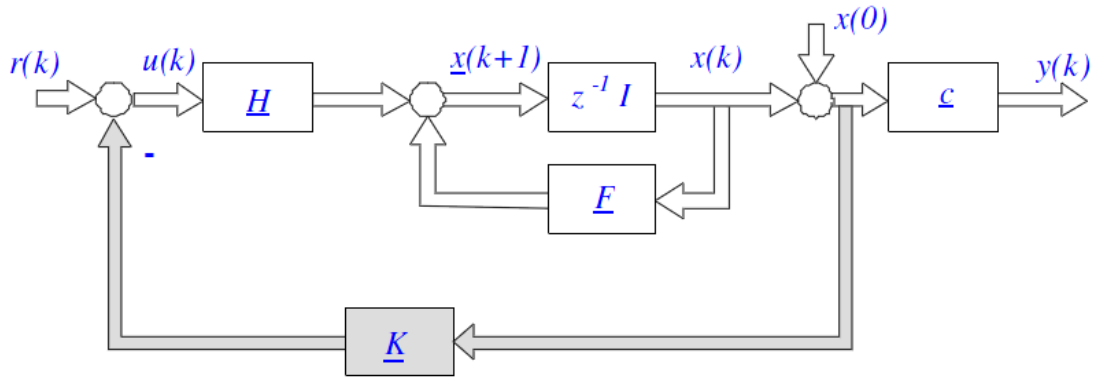


Figura 23 - Diagrama de blocos, variáveis de estado com realimentação K [7]

Com o auxílio do Matlab, usando o comando place com as matrizes $\underline{H}$, $\underline{F}$ e o conjunto de pólos desejados obtêm-se a Matriz $\underline{K}$ de realimentação facilmente.

4.2.Obtenção da Planta do Sistema

O motor utilizado na prática foi o Eletrocraft, e pode ser representado através do diagrama de blocos representado na Figura 24. [9]

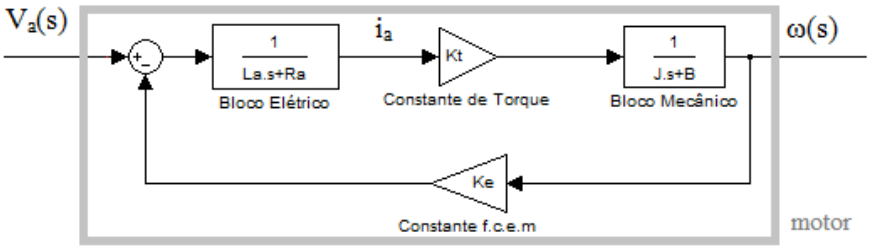


Figura 24 - Esquemático do Motor CC

Observa-se no esquemático que se aplica uma tensão $\boldsymbol{V_a}$ na entrada e o motor, representado pelo retângulo cinza, a planta, responde com uma velocidade $\boldsymbol{\omega}$ em sua saída.

Dentro de seu funcionamento há quatro blocos. Para utilizar o controle PID e *Dead-Beat* não é necessária a modelagem interna do sistema, porém, para utilizar o controlador de estados, é necessário calcular a representação por espaço de estados do sistema. E para isso, é necessário modelar os blocos internos do motor.

Dessa forma para o cálculo dos controladores e para controlar o sistema deve-se modelar:

- A planta do motor (bloco cinza da Figura 24);
- Os blocos internos, elétrico e mecânico.

Na seção 4.2.1 será disposto como é feita a modelagem da planta do motor e na seção 4.2.2 como foram obtidas os blocos internos, elétrico e mecânico.

Para a obtenção de todos os blocos usou-se a ferramenta *ident* do Matlab. Na ferramenta, basta indicar o vetor de entrada, o vetor de saída, o número de raízes e a ordem do sistema, feito isso, o Matlab processa os dados e retorna a sua função transferência.

Na Figura 25 está disposta a bancada e todos os equipamentos utilizados no projeto. Não é possível a visualização da bateria de carro pois esta está disposta abaixo da bancada.

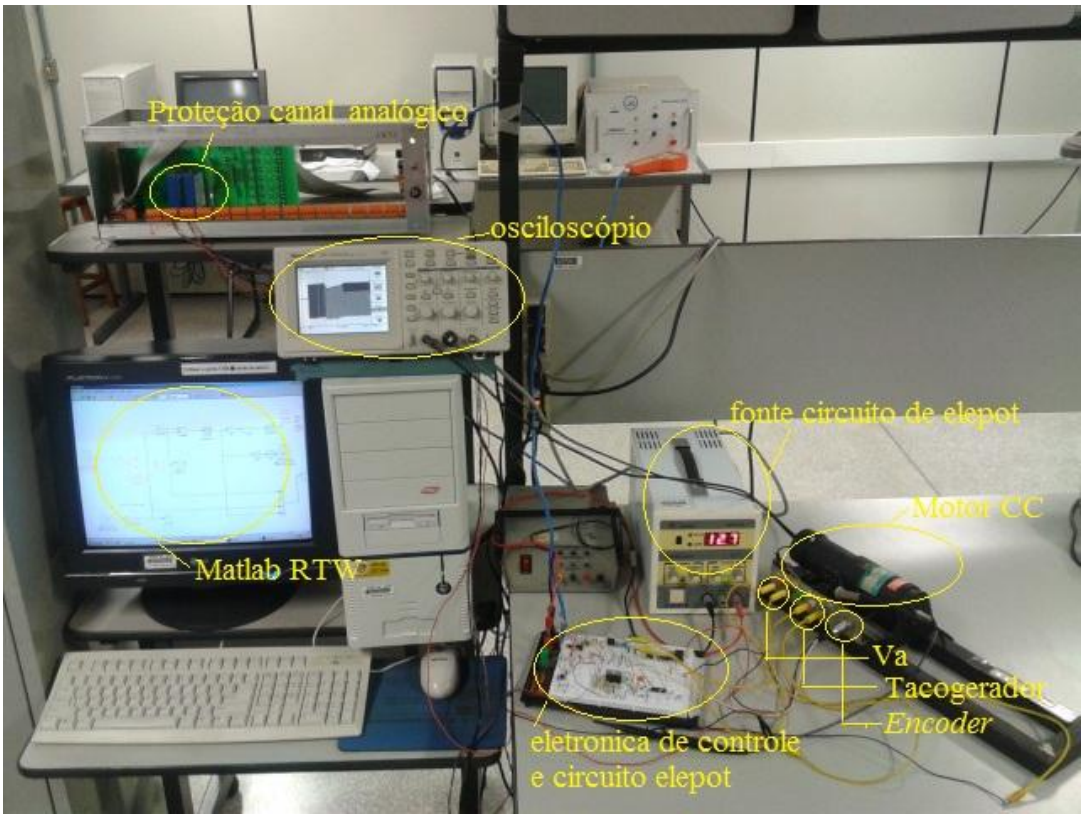


Figura 25 - Bancada

4.2.1.Obtenção da modelagem planta do motor.

Com o auxílio do Matlab, modelou-se o sistema da Figura 26.

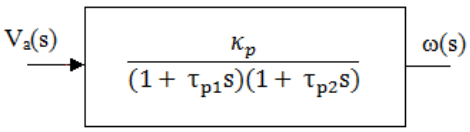


Figura 26 - Bloco Planta

Para a realização das medidas chaveou-se manualmente o motor CC com uma fonte de alimentação e os dados de velocidade foram obtidos através do Matlab Real-Time workshop através do *enconder* óptico. Para o cálculo da velocidade, o algoritmo verifica o número de pulsos que o motor realiza num tempo de amostragem T_o e a partir disso calcula-se a velocidade em radianos por segundo multiplicando-se essa diferença pela constante $2\pi T_o^{-1}/1024$ – pois a cada volta, portanto 2π, há 1024 pulsos. Observa-se esse algoritmo com na Figura 26, diagrama de blocos utilizado no Matlab RTW.

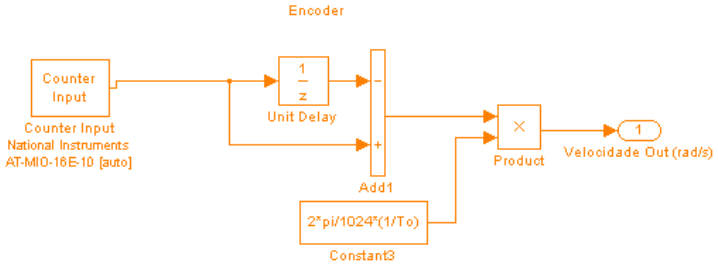


Figura 27 - Esquemático para cálculo de velocidade via simulink utilizando encoder óptico

Aferidos os dados da velocidade, para a eliminação do ruído utilizou-se o comando *filtfilt* do matlab utilizando os coeficiente de um filtro de Butterworth - ordem 5 com $\omega_n = 0{,}1$. Caso utilizasse somente o filtro de Butterworth, apareceria na imagem um atraso indesejado. Na Figura 28 podem-se observar os dados antes e depois de se utilizar o filtro.

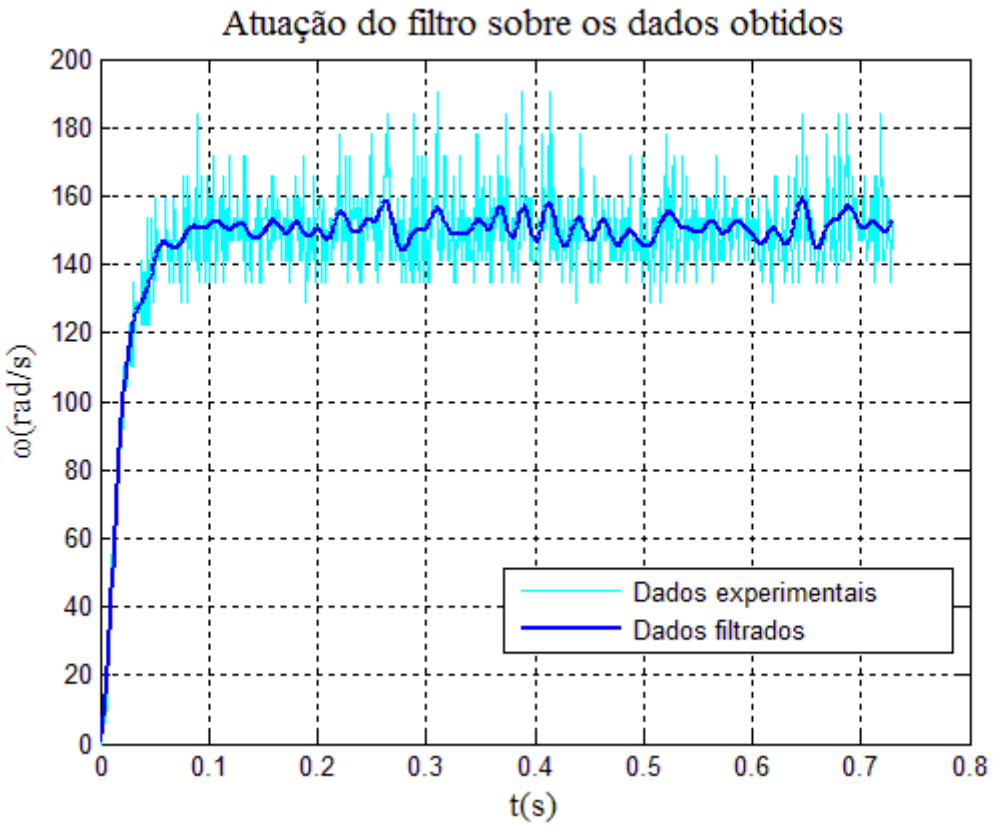


Figura 28 - Atuação do filtro de Butterwoth sobre os dados obtidos

Com os valores filtrados, usou-se a ferramenta *ident* para obtenção da função de transferência. Repetiu-se o procedimento quatro vezes: duas rotacionando o motor no sentido horário, e duas no anti-horário. Obteve-se portanto quatro funções de transferências, muito próximas para ambos os sentidos. Fez-se uma média aritimética de todas e obteve-se como resultado a planta da equação 12. Com o comando *zpk* chegou-se no modelo "zero pólos e ganho" da equação 13.

$$G_{motor} = \frac{12{,}5827}{(1+0{,}0151s)(1+0{,}0032s)} \tag{12}$$

$$\overset{zpk}{\Longrightarrow} G_{motor} = \frac{260403{,}5596}{(s+312.5)(s+66.23)} \tag{13}$$

O modelo obtido é comparado com os valores aferidos através do gráfico da Figura 29.

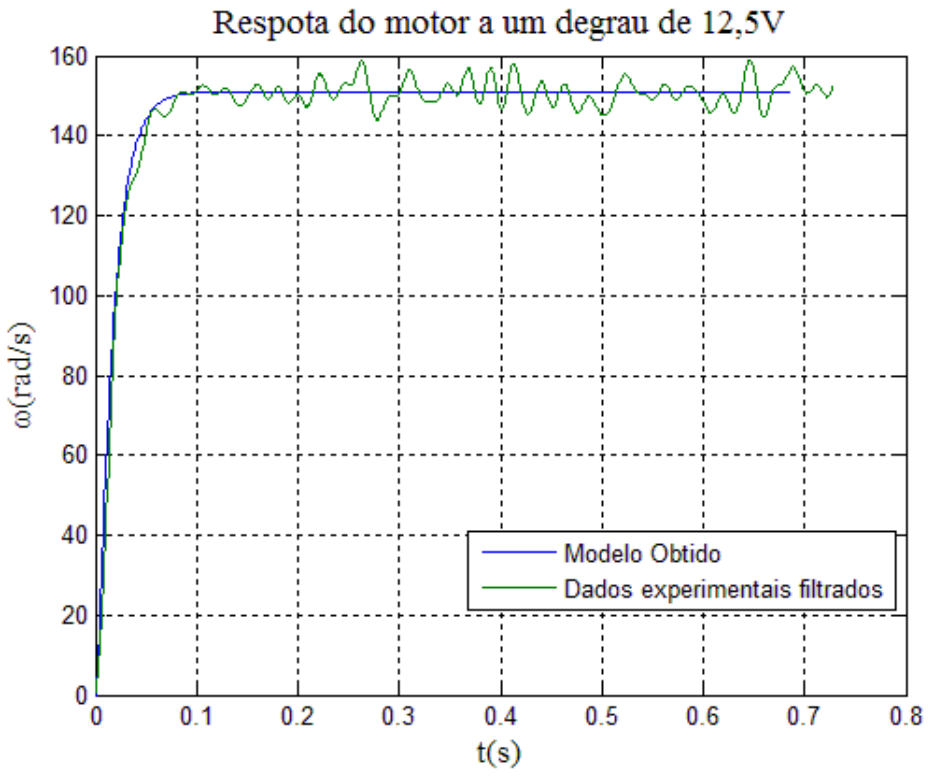


Figura 29 - Comparação planta modelada com seus valores obtidos

Assim, conclui-se que o modelo obtido é muito próximo do real.

4.2.2. Obtenção dos blocos internos do motor

Com o auxilio do *Matlab*, modelou-se os dois blocos da Figura 30.

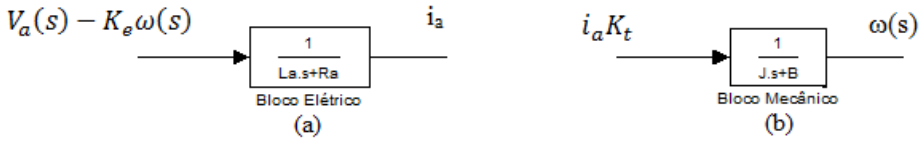


Figura 30 - Blocos internos do motor CC, (a) bloco elétrico e (b) bloco mecânico.

Nos dois blocos da Figura 30, observam-se duas constantes $\mathbf{K_e}$ e $\mathbf{K_t}$ que não se conhece o valor e que devem ser calculadas. Com elas, tem-se os dados de entrada e saída do blocos e finalmente pode-se usar a ferramenta do Matlab que calcula a função de transferência do sistema, *ident*. É importante notar que as entradas dos blocos não são degraus e sim as expressões $\mathbf{V_a(s) - K_e\,\omega(s)}$ e $\mathbf{i_aK_t}$ para os blocos elétrico e mecânico respectivamente.

Na modelagem do motor, se conhece a equação 13 [9]:

$$v_a = R_a i_a(t) + K_e\omega(t) + L_a\frac{d}{dt}i(t) \tag{14}$$

Para uma corrente constante tem-se que $\frac{d}{dt}i(t) = 0$ e portanto,

$$v_a = R_a i_a(t) + K_e\omega(t) \tag{15}$$

Dessa forma tem-se que,

$$K_e = \frac{V_a - R_a I_a}{\omega} \tag{16}$$

Portanto, através da equação 16 verifica-se como se pode calcular $\mathbf{K_e}$. Encontra-se outro problema, o valor de $\mathbf{R_a}$. Para obter o valor de $\mathbf{R_a}$, aplica-se uma tensão pequena no motor. Com este ainda parado, com o auxílio do multímetro verifica-se a corrente e calcula-se pela lei de ohm [10] sua resistência. Com $\mathbf{R_a}$ obtido**,** é possível aplicar uma tensão na armadura $\mathbf{V_a}$ no motor, mede-se a corrente $\mathbf{I_a}$ e com o valor da velocidade em rad/s nesse instante, **ω,** é possível calcular o valor de $\mathbf{K_e}$.

Obtem-se assim:

$$R_a = 2{,}2\Omega\ ;\ K_e = 0{,}0849 \tag{17}$$

Pelas propriedades físicas do motor [7], no sistema internacional de unidades tem-se que $K_t = K_e = 0{,}0849$.

Com estes cálculos agora é possível determinar as entradas do bloco elétrico e mecânico do sistema.

A velocidade ω(t) é obtida através do *encoder* como na seção 1.6, mas para a obtenção da corrente foi necessária a entrada analógica da placa de dados. Utilizou-se o diagrama de blocos da Figura 31 no Matlab RTW para a aquisição do sinal.

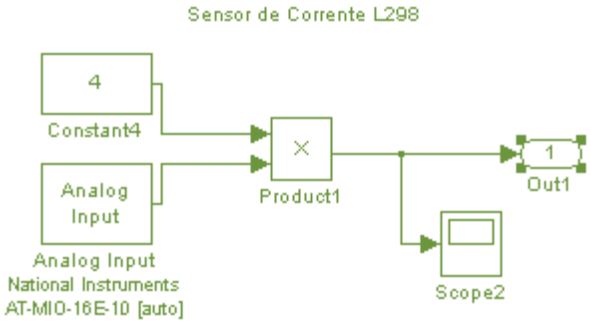


Figura 31 - Sensor de corrente

De um divisor de tensão, o sinal passar por um bloco de proteção óptico que atenua o sinal pela metade e por fim segue para a entrada analógica da placa. A constante multiplicativa da Figura 31 é uma compensação referente a essas divisões.

Dessa forma, tem-se todos os sinais necessários para utilização do *ident* no matlab. Da mesma forma que no caso do cálculo da planta, foram feitas 4 medidas, e com a média delas obtiveram-se os resultados:

- Bloco Elétrico

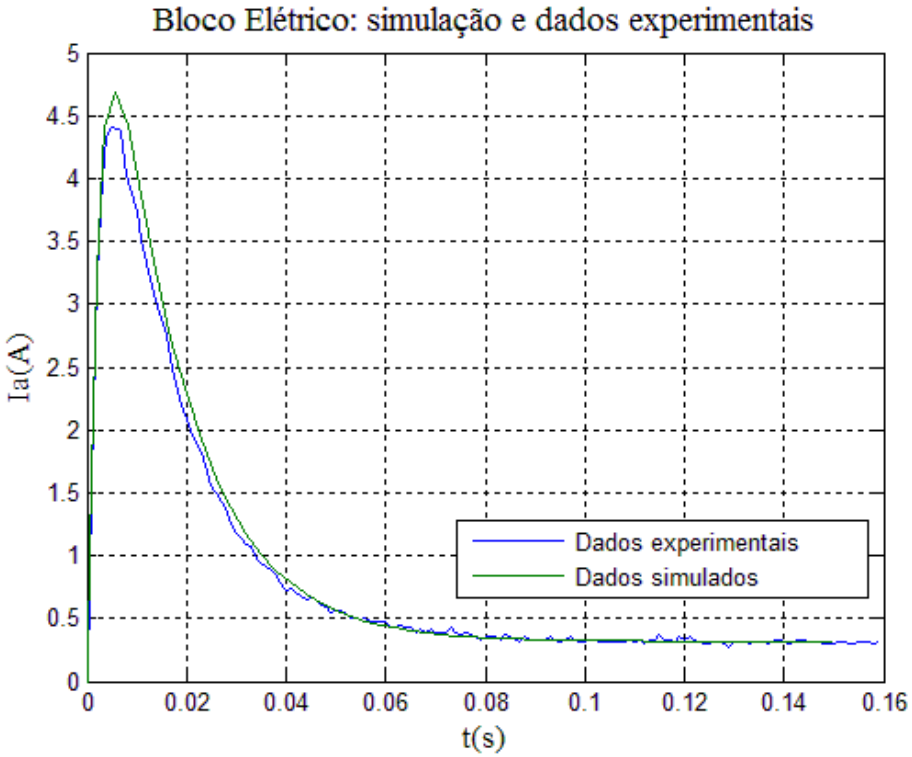


Figura 32 - Bloco elétrico, comparação dos dados simulados com os aferidos

$$G_{eletrico} = \frac{0{,}4141}{1+0{,}0022s} \quad (18)$$

- Bloco Mecânico

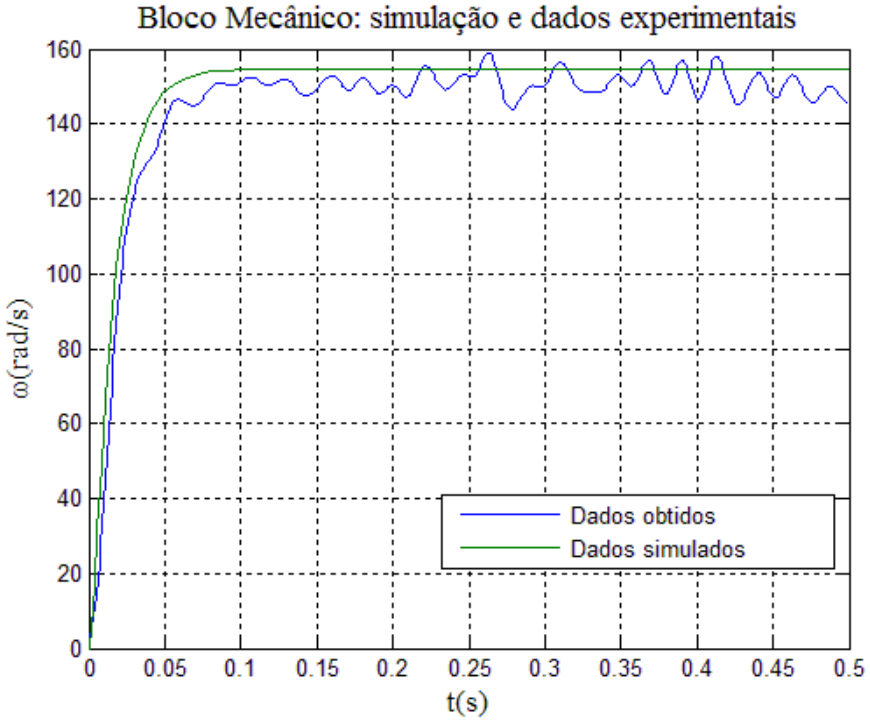


Figura 33 - Bloco mecânico, comparação dos dados simulados com os aferidos

$$G_{mecânico} = \frac{5730}{1+0{,}3068s} \quad (19)$$

Os dados simulados foram obtidos através da simulação feita no Simulink disposta na Figura 34.

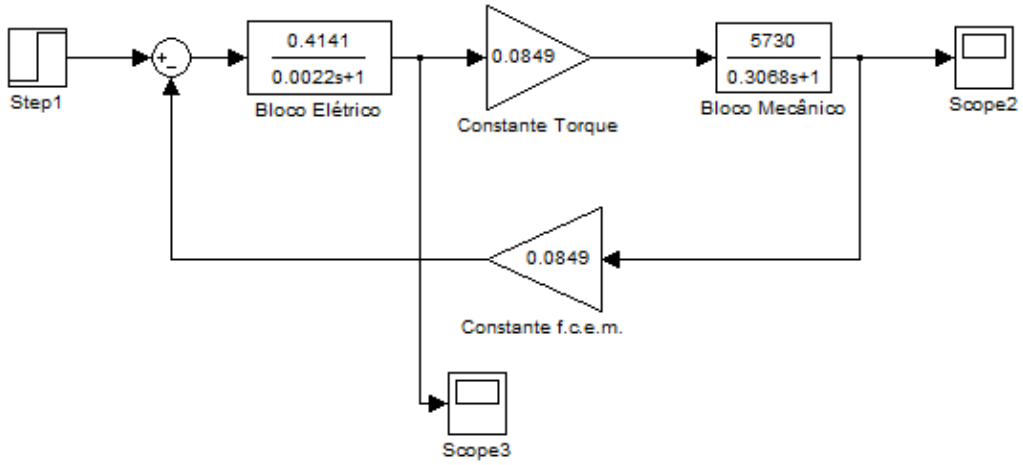


Figura 34 - Esquemático usado em simulink para a simulação

A função de transferência do processo é dada pela equação

$$G_{motor2} = \frac{201.5}{0.000675\,s^2 + 0.309\,s + 18.1} \quad (20)$$

$$\overset{zpk}{\Longrightarrow} G_{motor2} = \frac{298462.3173}{(s+388.8)\,(s+68.98)} \quad (21)$$

Comparando a equação 13 com a equação 21 percebe-se que os pólos estão muito próximos. Por outro lado, o ganho das duas equações apresentam uma diferença significativa. Essa diferença pode ser justificada pelo valor de K_e que foi encontrado através de dados adquiridos no laboratório, portanto, podendo trazer erros de precisão.

Com as funções de transferências dispostas na equação 18 e equação 19, fez-se uma comparação polinomial com as funções de transferência da Figura 30 (a) e Figura 30 (b) respectivamente. Com isso, obtiveram-se os valores dos parâmetros elétricos e mecânicos do motor, dispostos na tabela 2.

Tabela 2 - Parametros internos do motor Eletrocraft obtidos experimentalmente

Parâmetros	**Valores**
Ra (Ω)	2,2
La(H)	3,09e-3
J (Nms²/rad)	53,54e-6
Ke (Vs/rad)	0,0849
B (Nms/rad)	176,7e-6

4.3.Projeto em controle

Um dos grandes problemas enfrentados na realização do controle do motor foi a escolha do valor de referência ou *setpoint*. Com a modelagem disposta na seção 4.2.1, vê-se que a velocidade máxima atingida pelo motor com a tensão disposta a ele de 12,5V é de aproximadamente 150rad/s. Por vezes, o controle demanda uma velocidade maior do que essa, assim saturando a fonte de alimentação. Dessa forma, teve-se que escolher uma velocidade relativamente baixa referente à máxima do motor evitando, portanto, o problema com a saturação.

4.3.1.Controlador PID

Para a realização do controlador *PID*, primeiramente fez-se o controlador proporcional e depois partiu-se para o modelo proporcional, integrativo derivativo.

(i) Controlador Proporcional

Com o auxílio do simulink, primeiramente montou-se uma simulação do sistema em malha fechada para se obter um valor de K_p, constante proporcional, que aumentava a velocidade de resposta do motor e ao mesmo tempo minimizava o erro da saída. Utilizou-se o diagrama de blocos da Figura 35 para se obter o melhor valor da constante.

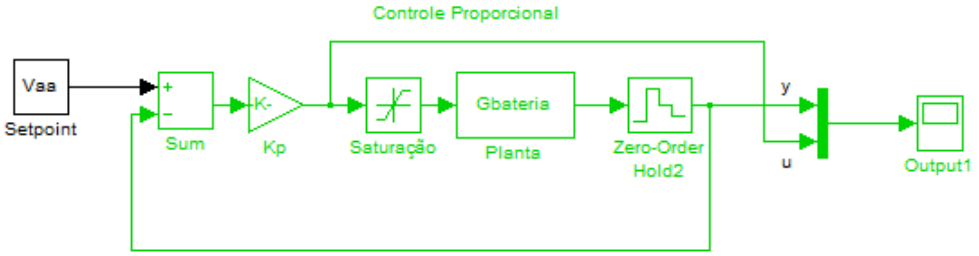


Figura 35 – Controlador proporcional, diagrama de blocos.

Na Figura 36, observa-se como a saída **y** do sistema é alterada devido a variação do ganho proporcional K_p. Percebe-se que com o aumento do ganho, diminui-se o erro à velocidade de referência da simulação - ω=60rad/s – bem como há o aparecimento de um sobresinal. Nota-se também que não há valor de K_p em que zere o erro em relação ao setpoint.

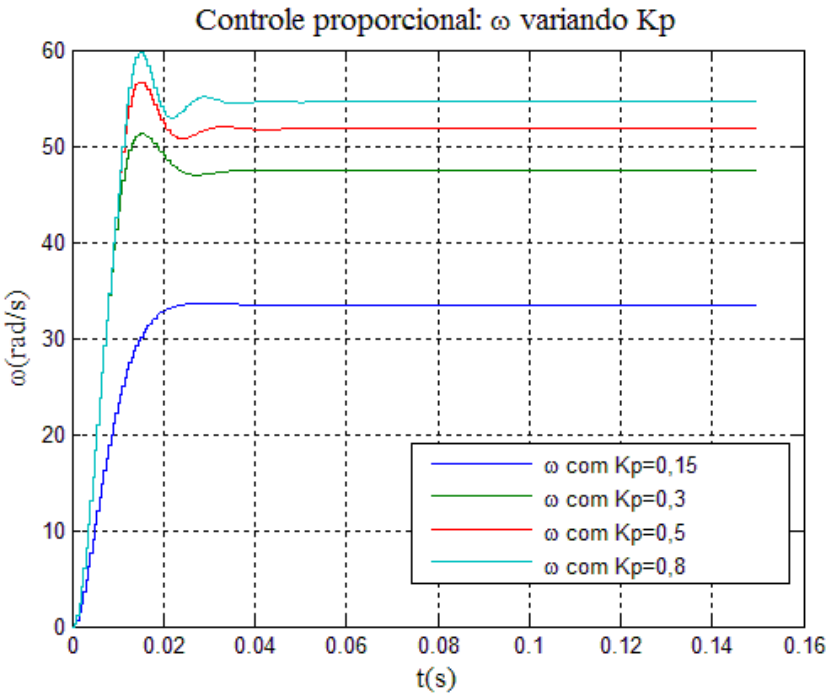


Figura 36-Controlador proporcional, variação de Kp

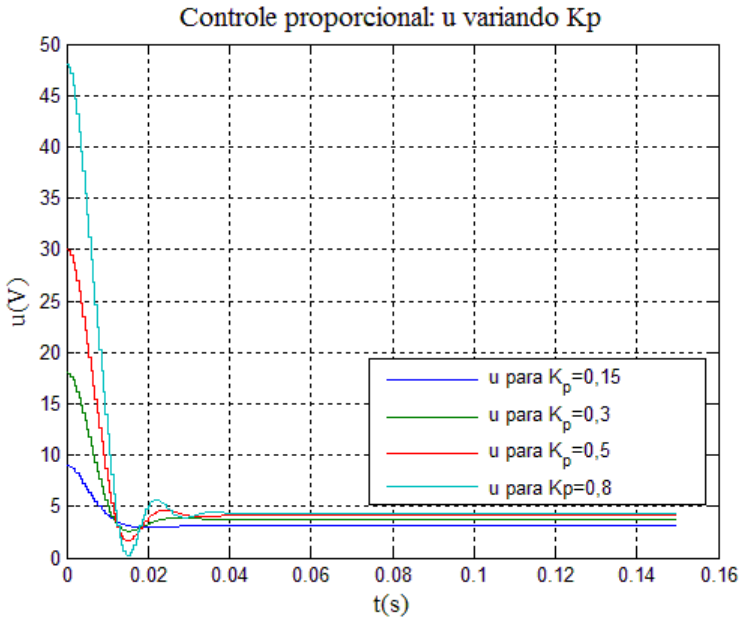


Figura 37 - Controlador proporcional, u variando com Kp

Por outro lado, com o aumento de K_p, a saída controlada **u,** Figura 37, aumenta. Para u>10V, há uma saturação no PWM e não há a atuação de controle. Portanto, em todos os K_p simulados, com exceção do K_p=0,15 há uma saturação.

Com a análise feita a partir da simulação, fez-se utilizando o Matlab RTW o esquemático da Figura 38 para a realização do controle proporcional em tempo real.

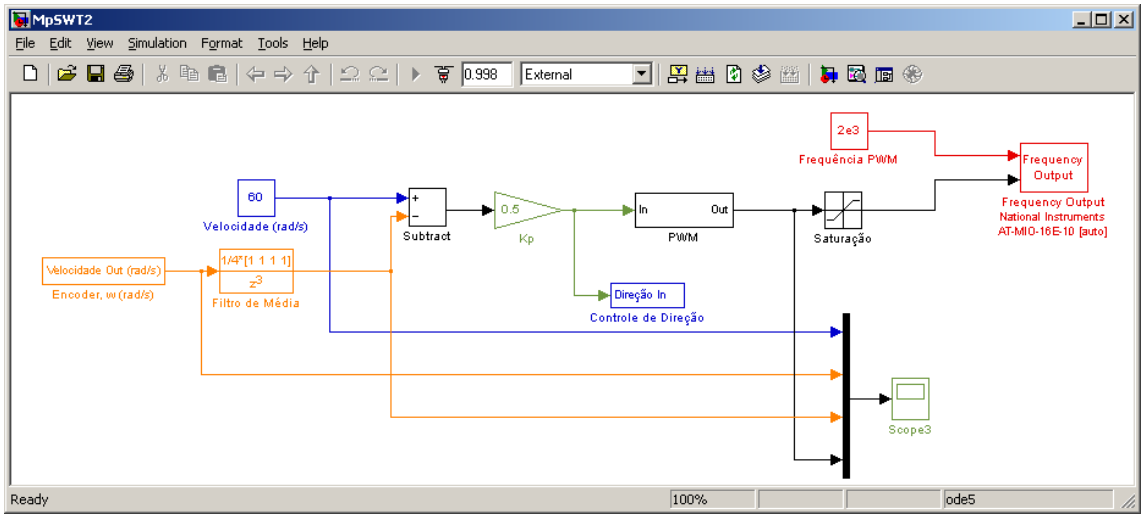


Figura 38 - Controle proporcional digital em tempo real

O bloco *Filtro de Média* opera como um filtro que elimina ruído do sinal. Os blocos *Velocidade Out*, *PWM*, *Direção In*, observados na Figura 3 estão dispostos respectimente na Figura 28, Figura 39 e Figura 40.

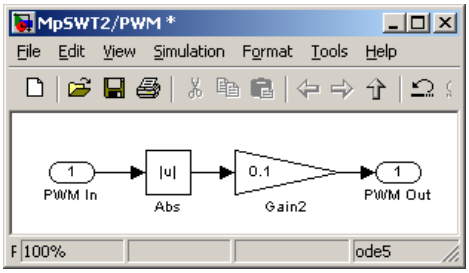


Figura 39 - Diagrama de blocos do PWM no Simulink

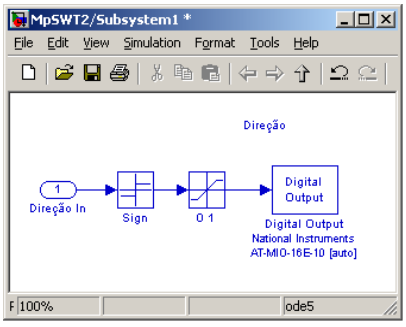


Figura 40 - Diagrama de blocos do sentido de rotação do motor no Simulink

No bloco de PWM, Figura 39, nota-se o fator de condicionamento com ganho 0.1. Este transforma 0V a 10V em 0% a 100%, sinal necessário ao PWM.

Os dados obtidos estão dispostos e comparados com os simulados no gráficos das figuras 41, 42, 43 e 44.

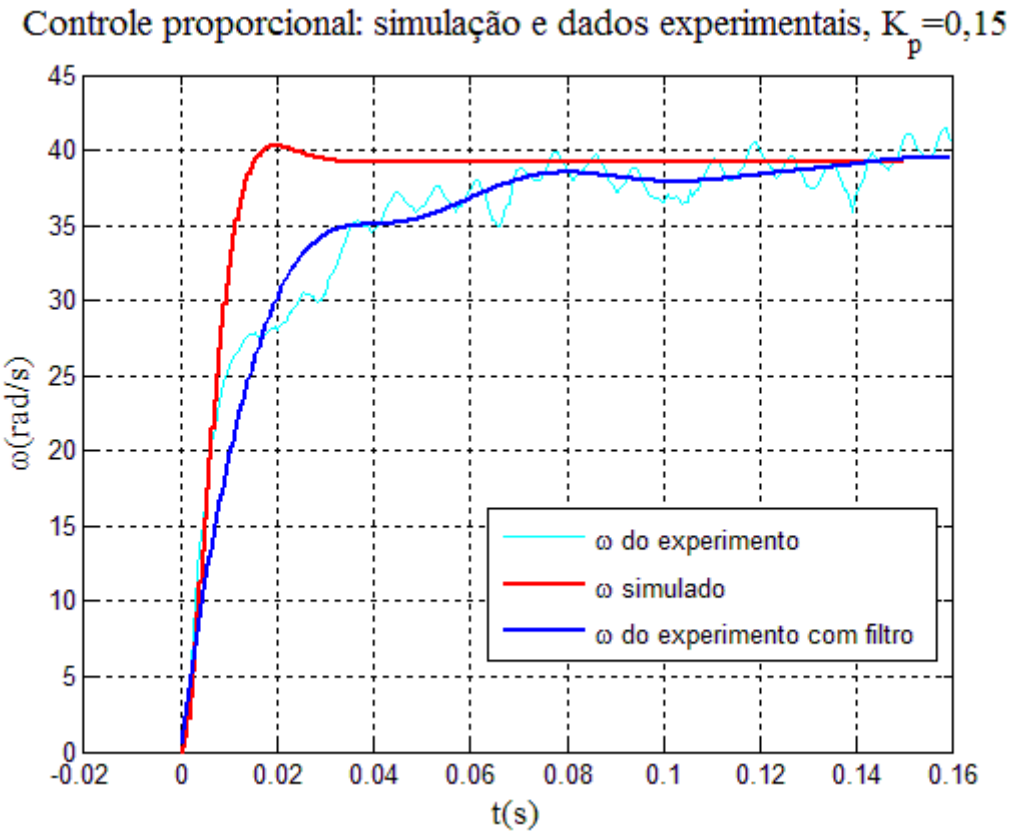


Figura 41 - Controlador proporcional, simulação comparada com experimento para Kp=0,1

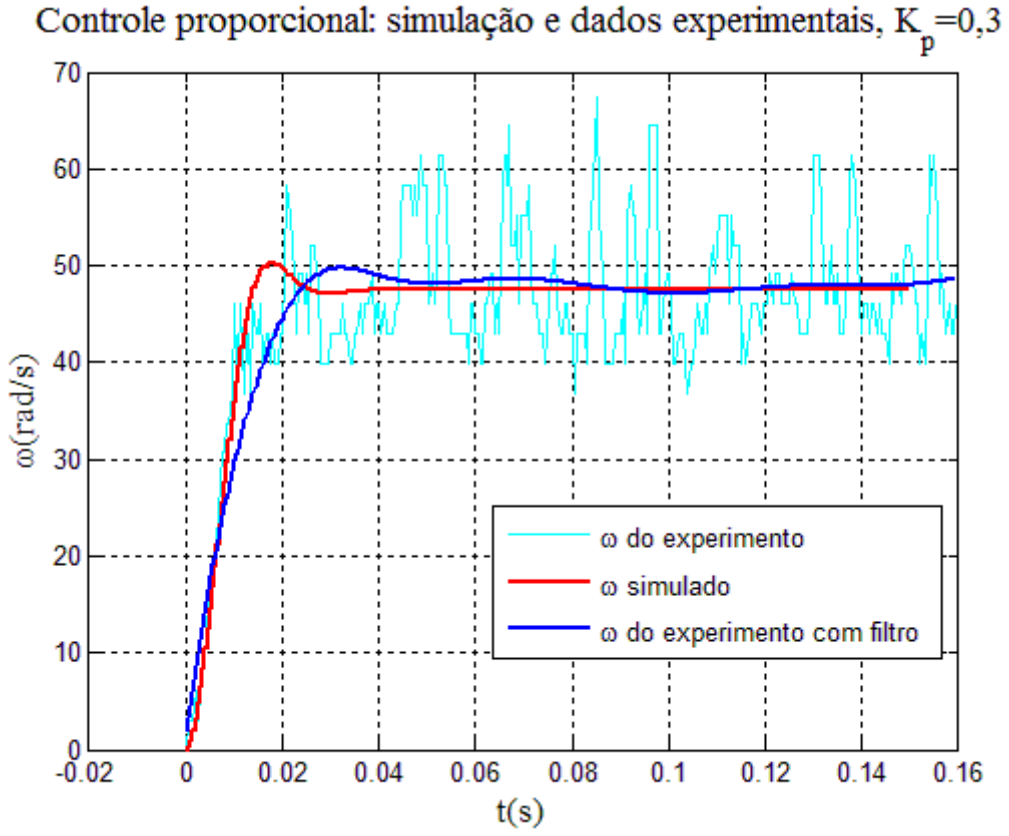


Figura 42 - Controlador proporcional, simulação comparada com experimento para Kp=0,3

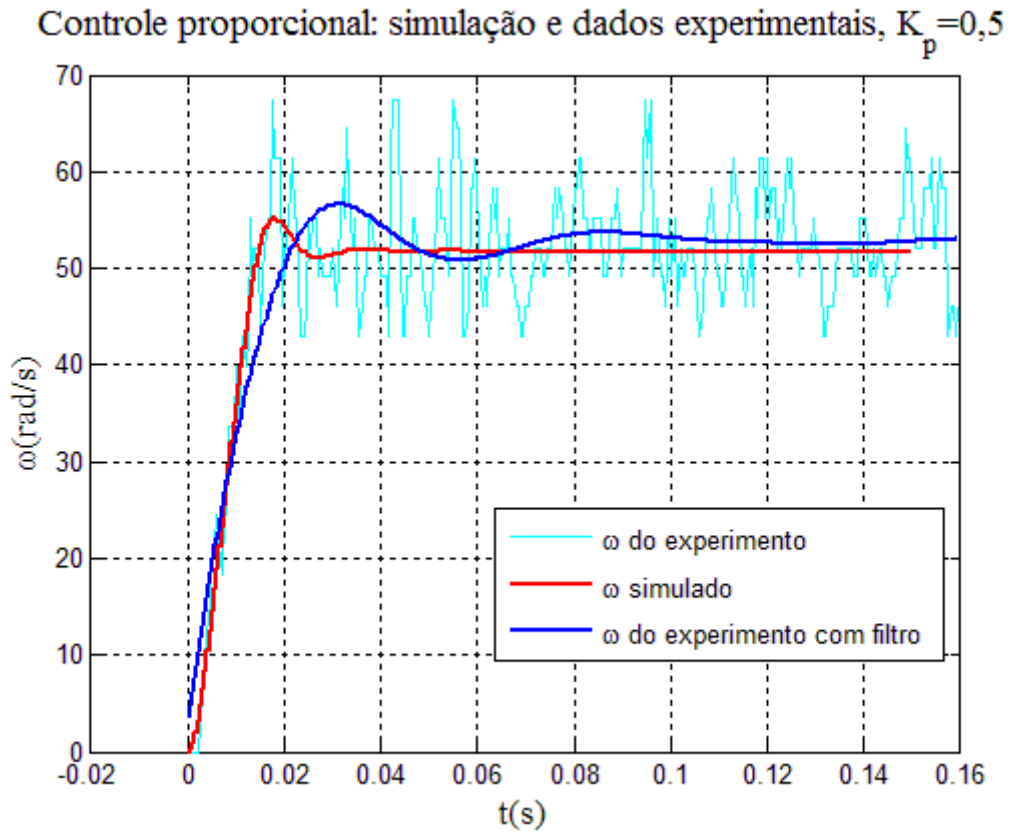


Figura 43 - Controlador proporcional, simulação comparada com experimento para Kp=0,5

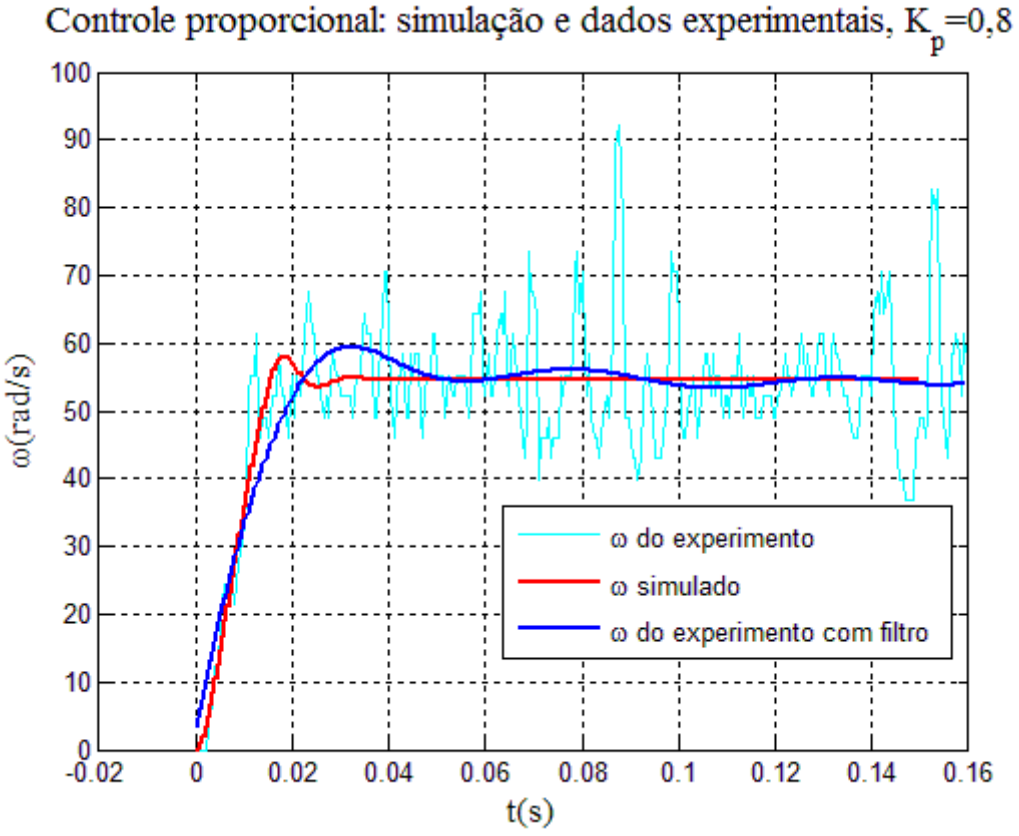
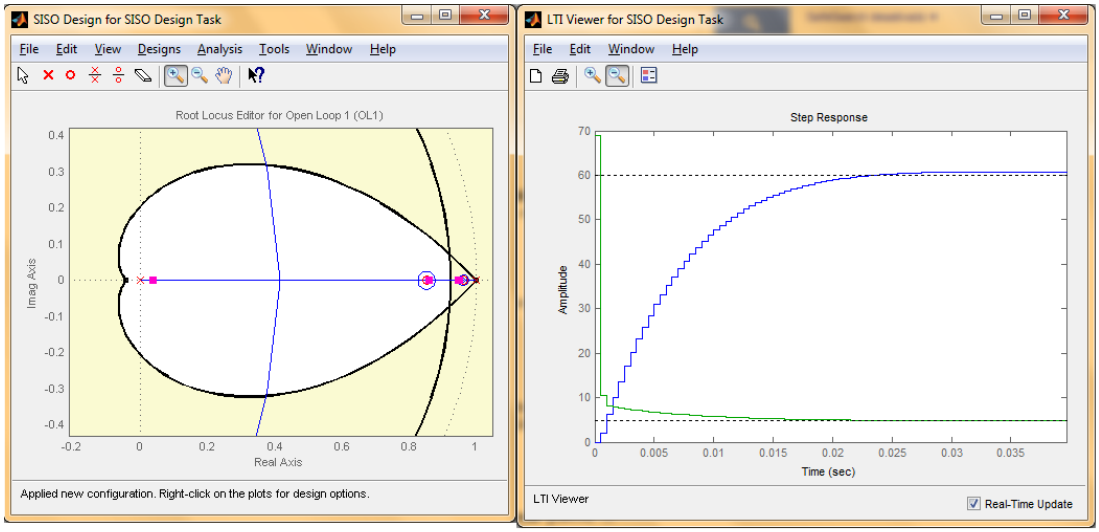


Figura 44 - Controlador proporcional, simulação comparada com experimento para Kp=0,8

Os dados obtidos, figuras 41, 42, 43 e 44, foram compatíveis com as expectativas da simulação. Pode-se observar que quanto maior o valor de K_p, menor o erro em regime. Constatam-se as ondulações e o aparecimento de sobresinal. Porém, pode-se notar que o sinal controlado estabiliza-se após algum tempo depois da simulação. Isso ocorre, pois o sinal de controle é um pouco atrasado devido a influência do L298, discutida na seção 3.2.1.

(ii) Controlador PID

Como constatado, o controlador proporcional não zerou o erro de regime. Assim, se construiu o controlador PID para que houvesse um cancelamento desse erro. Para construir este controlador utilizou-se a ferramenta do Matlab *rltool*. Observam-se na Figura 45 os gráficos gerados pela ferramenta rltool do matlab.



(a) (b)

Figura 45 - *rltool* mostrando **(a)** os pólos e as raízes do compensador e **(b)** a resposta ao degrau **y** em relação a **r** e **u** em relação a **r** para um controlador PID

Para o projeto do controlador levou-se em consideração os pólos do sistema com ganho proporcional K_p=0,1 ([0.9102-0.0372i;0.9102 + 0.0372i]). Na Figura 46 (b) vê-se que a saída controlada u, com os pólos já alocados nos desejados, está saturada no instante inicial em u=70V aproximadamente e essa saturação impede o correto funcionamento do controlador.

Em vista disso, tentou-se a construção do controlador PI, pois exige menos energia do que PID. Utilizando o *rltool*, Figura 46, obteve-se a equação 22 para o controlador com T_o=1/2000s.

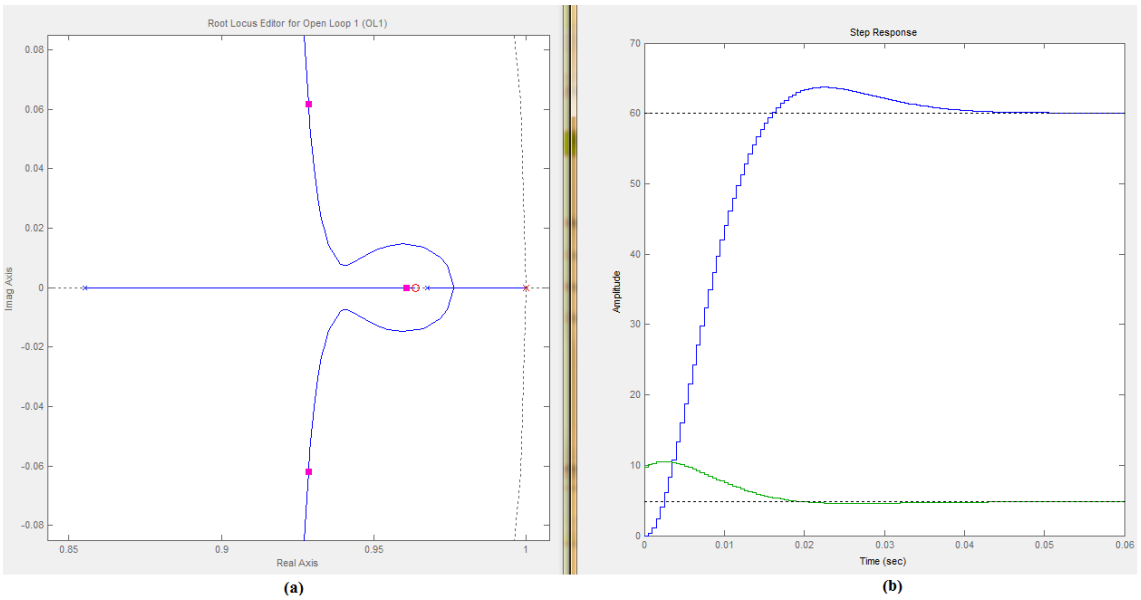


Figura 46 - Controlador PI, rltool mostrando (a) os polos e as raízes do compensador e (b) a resposta ao degrau **y** em relação a **r** e a resposta **u** em relação a **r** para um

$$C = \frac{0.16271\,(z-0.9637)}{(z-1)} \qquad (22)$$

Com o compensador realizou-se as simulações da forma disposta na Figura 47.

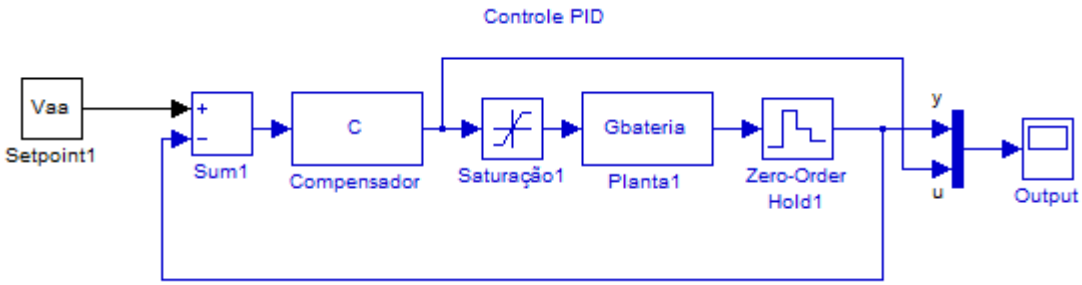


Figura 47 – Diagrama de blocos, controlador PID

A resposta da simulação é a curva mostrada no gráfico da 48.

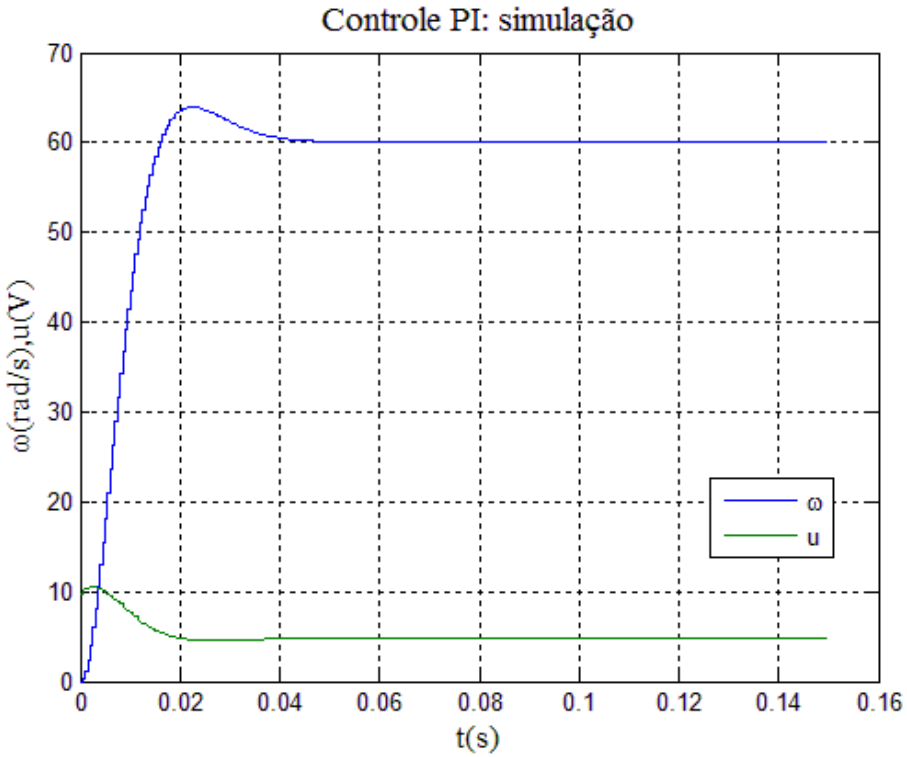


Figura 48 - Simulação controlador PID

Com esses resultados montou-se o diagrama de bloco utilizando o Matlab RTW, Figura 49, e tiveram-se os resultados comparados na Figura 50.

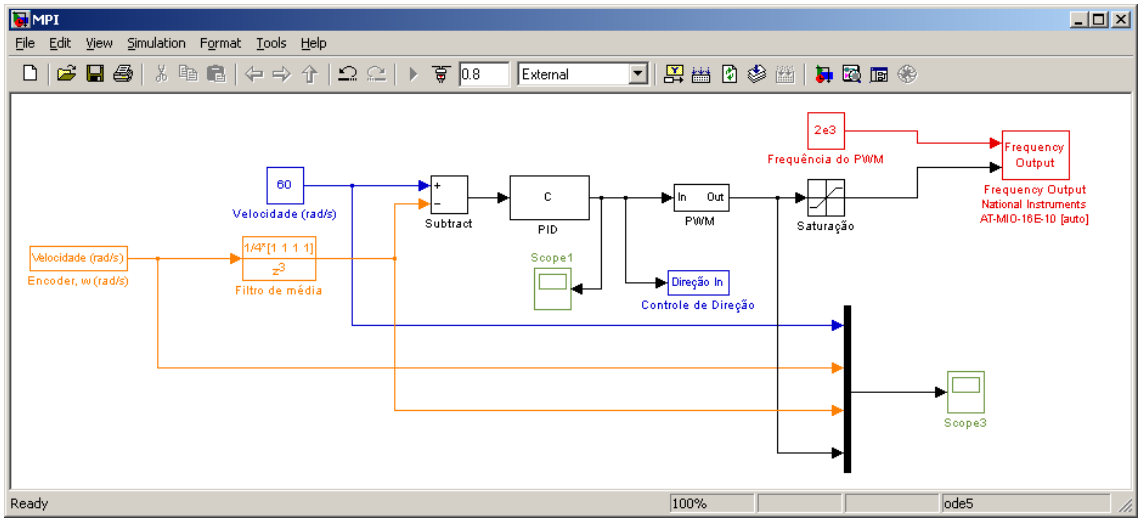


Figura 49 - Controle PID utilizando o Matlab RTW

Os blocos *Filtro de Média*, *Velocidade Out*, *PWM* e *Direção In* são os mesmos utilizados no caso do controlador proporcional.

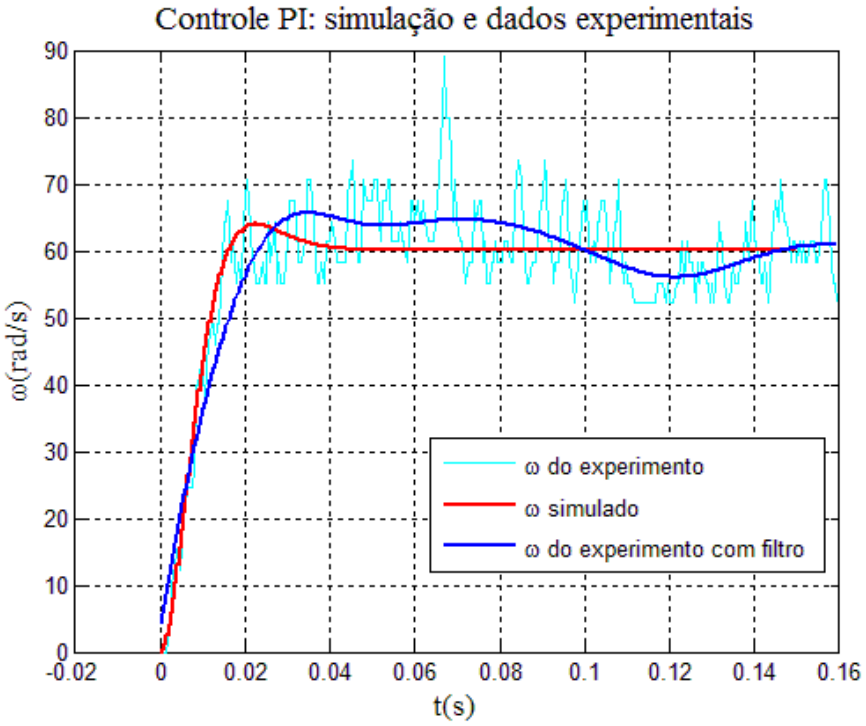


Figura 50 - Controle PI, comparação entre simulação e valores experimentais

Pode-se observar que o controlador PI apresentou um atraso em realação ao sistema simulado e um erro em relação à referência. Esses erros são reflexos dos problemas discutidos na seção 3.2.1, no que se refere à influência do L298 no sinal do motor.

4.3.2.Controlador Dead-Beat

Para obter o controlador Dead-Beat, é importante notar que:

- Os parâmetros do controlador dependem somente da FT da planta utilizada discretizada;
- Sua primeira ação de controle, $u(0)=q_o$, é inversamente proporcional a soma dos coeficientes b_i da planta discretizada.

- Para se ter $\mathbf{b_i}$'s relativamente grandes há de se escolher um período de amostragem T_o o maior possível.

Dessa forma, a escolha de T_o para o controlador é extremamente importante.

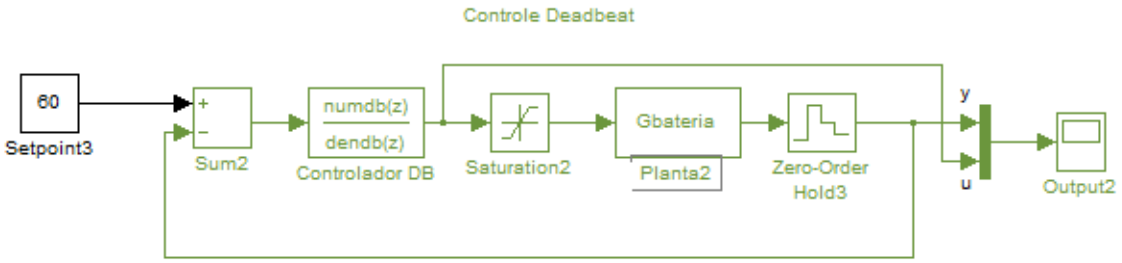


Figura 51 - Simulação, controlador Dead-Beat

Para as simulações, Figura 51, adotou-se T_o=10ms e T_o=25ms. Para cada tempo de amostragem apresentaram as funções de transferência dadas pela equação 20 e equação 21.

$$G_{DB1,To=10ms} = \frac{0{,}1716-0{,}0961z^{-1}+0{,}0039z^{-2}}{1-0{,}7720z^{-1}-0{,}2280z^{-2}} \quad (23)$$

$$G_{DB2,To=25ms} = \frac{0{,}0983-0{,}0188z^{-1}+0{,}0000z^{-2}}{1-0.9370z^{-1}-0.0630z^{-2}} \quad (24)$$

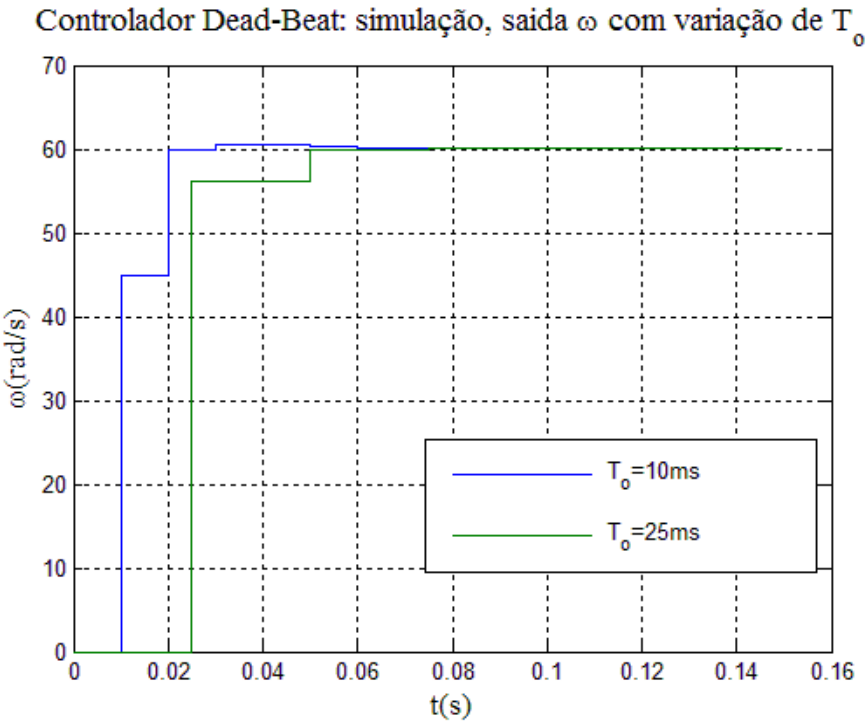


Figura 52 - Controlador Dead-Beat, simulação com com To=10ms

Na simulação, pode-se observar que a saída **y**, Figura 52, para T_o=10ms alcança e estabiliza-se no setpoint de forma mais rápida do que T_o=25ms. Por outro lado, a saída controlada **u** para o caso de T_o=10ms apresenta uma saída controlada **u**, Figura 53, maior do que 10V, portanto saturada enquanto para T_o=25ms essa saturação não ocorre.

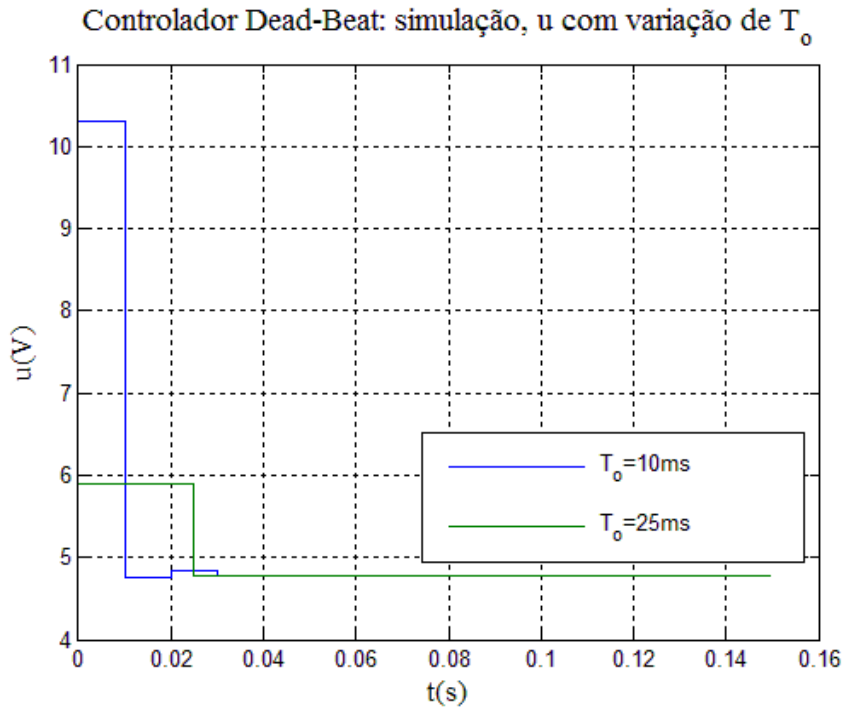


Figura 53 - Controlador Dead-Beat, saída **u** simulada

Fez-se o esquemático da Figura 54 e usando o Matlab RTW realizou-se o controle em tempo real.

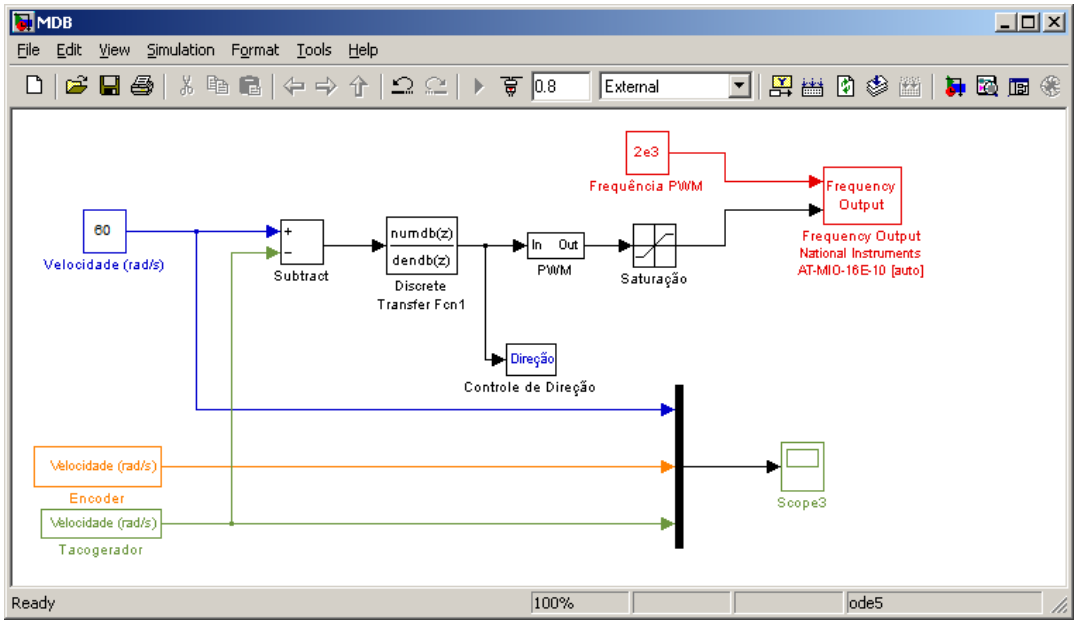


Figura 54 - Controlador Dead-Beat, esquemático RTW

O bloco *Discrete Transfer Fcn1* é o controlador Dead-Beat da equação 20 e os blocos *Controle de Direção, PWM, Encoder* foram devidamente explicados na seção 3.2.1. Os resultados obtidos podem ser observados nos gráficos da Figura 55 e Figura 56.

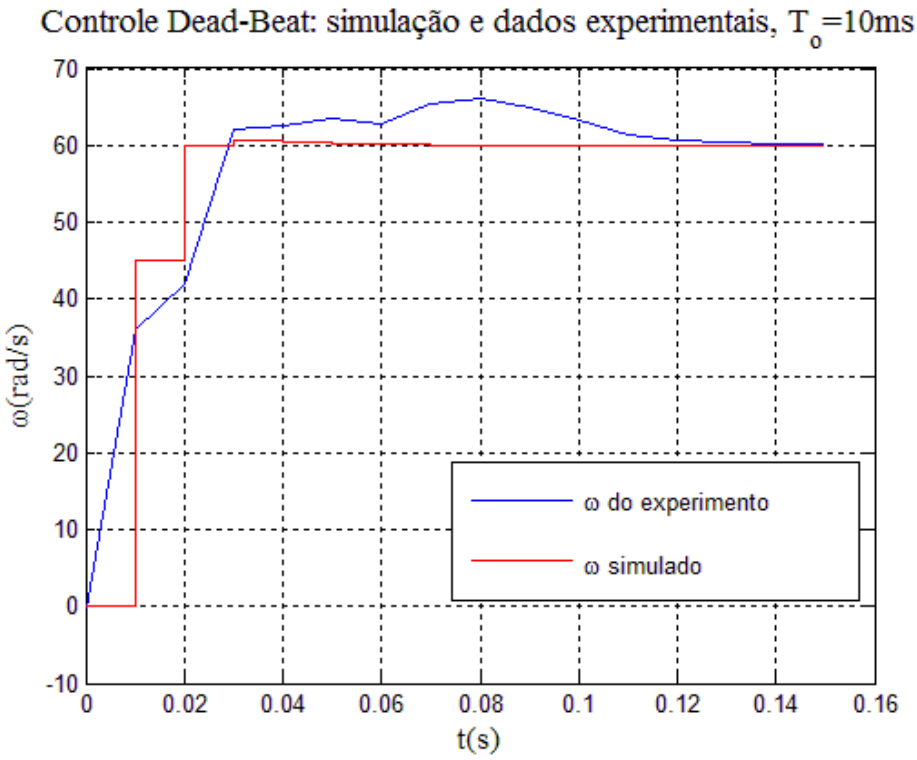


Figura 55 -Controlador Dead-Beat, simulação e experimento para To=10ms

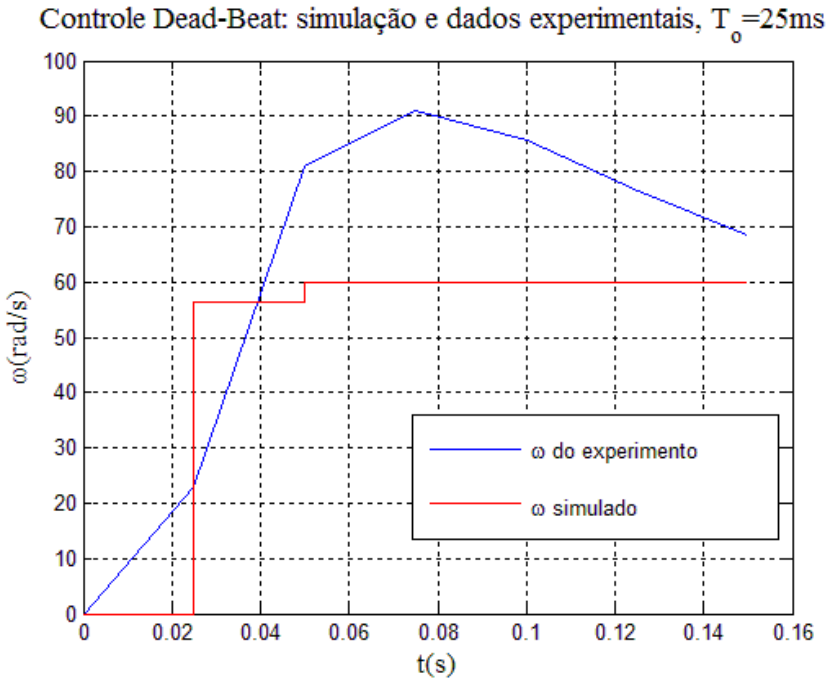


Figura 56 - Controlador Dead-Beat, simulação e experimento para To=25ms

Para um controlador Dead-Beat, em uma planta de segunda ordem, o sistema estabiliza-se na segunda amostragem. Esse fato é facilmente observado nos gráficos referente a simulações. No caso do experimento o controlador Dead-Beat não foi funcional, Figura 55 Figura 56. Ainda, com T_o grande o ruído foi amostrado, dessa forma explica-se o *overshoot* da Figura 56 que na verdade pode ter sido o ruído amostrado.

No primeiro caso, T_o=10ms, não houve a ação de controle no primeiro período de amostragem devido à saturação, já discutida e evidenciada pela Figura 17. No segundo caso, a causa mais significativa do problema é o atraso do sinal que faz com que nos 10ms iniciais, a potência fornecida é aproximadamente 60% da desejada. Na Figura 57 mostra-se o problema de saturação e o atraso.

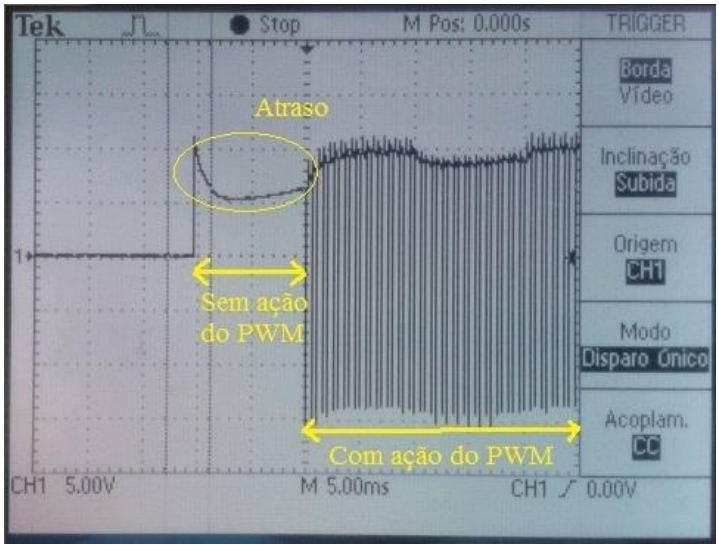


Figura 57 - Controlador Dead-Beat, problemas devido ao atraso e à saturação

4.3.3.Controlador por Espaço de Estados

Para realimentar o sistema por espaço de estado deve-se encontrar as matrizes de espaço de estado discretas. Para isso devem-se obter primeiramente as matrizes no espaço contínuo. Consideram-se o sistema de equação 22.

$$\begin{cases} v_a(t) = K_e\omega(t) + L_a\frac{d}{dt}i_a(t) & (I) \\ K_t i_a = J\frac{d}{dt}\omega(t) + B\omega(t) + F & (II) \end{cases} \tag{25}$$

Sendo u a tensão de entrada v_a [V], os espaços de estados x_1, velocidade [rad/s] e x_2, a corrente [A] que passa pelo motor e fazendo F=0 chega-se nas equações 23 e 24.

$$\overset{(II)}{\Longrightarrow} \dot{x_1} = -\frac{B}{J}x_1 + \frac{K_t}{J}x_2 \tag{26}$$

$$\overset{(I)}{\Rightarrow} \dot{x_2} = -\frac{K_e}{L_a}x_1 - \frac{R_a}{L_a}x_2 + \frac{1}{L_a}u \tag{27}$$

Utilizando o modelo de espaço de estados

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{28}$$

Tem-se que:

$$A = \begin{bmatrix} -\frac{B}{J} & \frac{K_t}{J} \\ -\frac{K_e}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} \tag{29}$$

$$B = \begin{bmatrix} 0 \\ \frac{1}{L_a} \end{bmatrix} \tag{30}$$

Com os parâmetros obtidos na seção 4.2.2, tem-se as matrizes de espaço de estados contínuas:

$$A = \begin{bmatrix} -0{,}0033 & 1{,}5856 \\ -0{,}0139 & -0{,}7106 \end{bmatrix}\text{k} \tag{31}$$

$$B = \begin{bmatrix} 0 \\ 163{,}9344 \end{bmatrix} \tag{32}$$

$$C = [1\ 0] \tag{33}$$

$$D = [0] \tag{34}$$

Obtidas as matrizes contínuas faz-se a discretização das matrizes **A** e **B** e obtem-se as matrizes **F** e **G** discretizadas com T_o=1/2000s.

$$F = \begin{bmatrix} 0{,}9958 & 0{,}7242 \\ -0{,}006356 & 0{,}8326 \end{bmatrix}\text{k} \tag{31}$$

$$G = \begin{bmatrix} 0.03059 \\ 0.07493 \end{bmatrix} \tag{32}$$

Com o espaço definido, se obtém a matriz de realimentação de estado **K** através do comando *place* do Matlab. Para realizar a realimentação de estado, considerou os novos pólos do sistema no mesmo lugar dos pólos dominantes de um sistema com controlador proporcional de ganho K_p=0,1 ([0.9102 -0.0372i ; 0.9102 + 0.0372i]). Com isso encontra-se **K**.

$$K = [0.0894\ 0.1166] \tag{35}$$

Fez-se a simulação do controlador, como disposto na Figura 58.

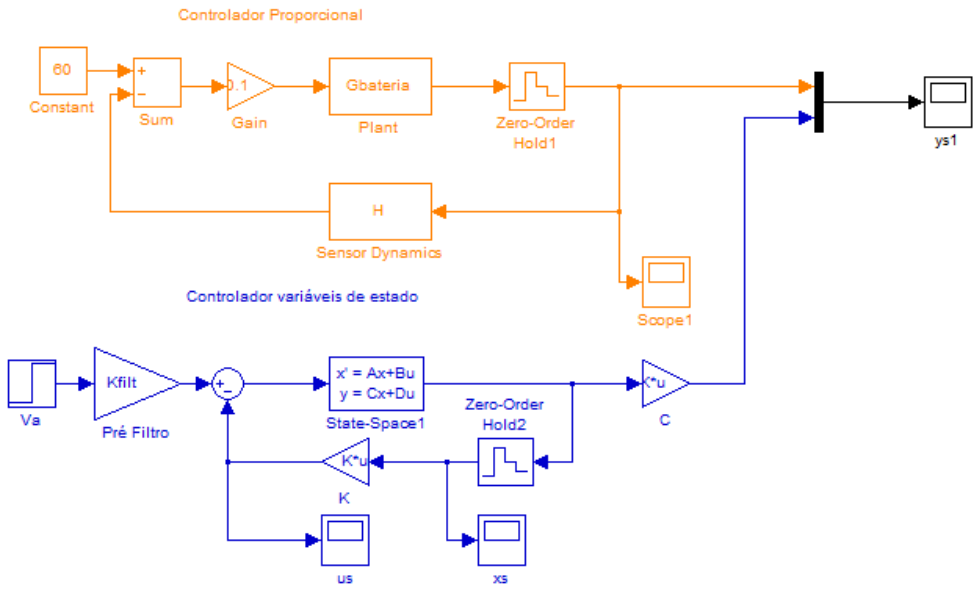


Figura 58 - Controlador por variáveis de estado, simulação

Primeiramente, fez-se o pré-filtro, do controlador por variáveis de estado como unitário e obteve-se um resultado com a mesma dinâmica do controlador proporcional de ganho K_p=0,1, porém com o *setpoint* muito maior. Dessa forma, cria-se o um pré-filtro de ganho K_{filt}=0,100 para realizar o ajuste. Na Figura 59 estão dispostos os quatro casos, a saída **y** para o controlador proporcional de ganho K_p, a saída **y** para a realimentação por espaço de estados com o pré-filtro unitário, a saída **y** com o pré-filtro K_{filt}=0,1 e a saída **y** com o pré-filtro de K_{filt}=0,2.

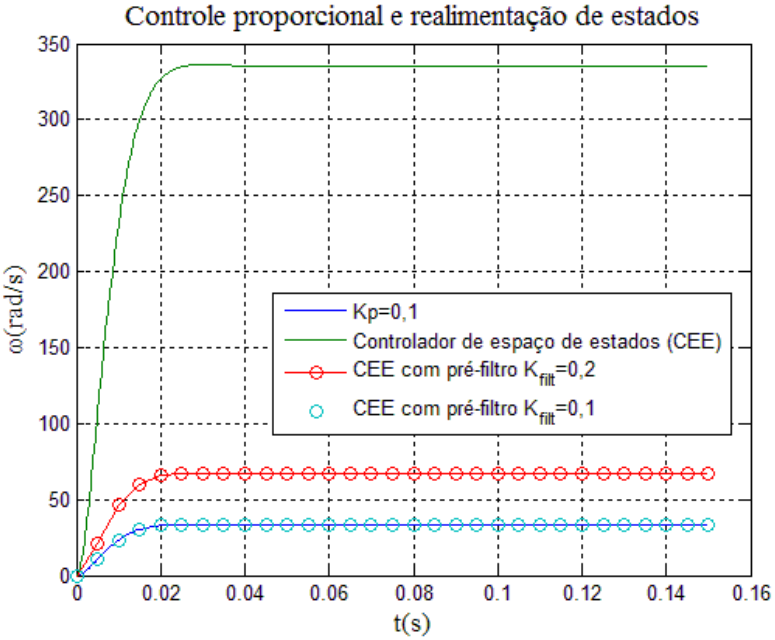


Figura 59 - Controlador por realimentação de estados e controlador proporcinoal

Com as simulações realizadas, fez-se o controle utilizando o Matlab RTW, Figura 60. Obtiveram-se os resultados apresentados na Figura 61.

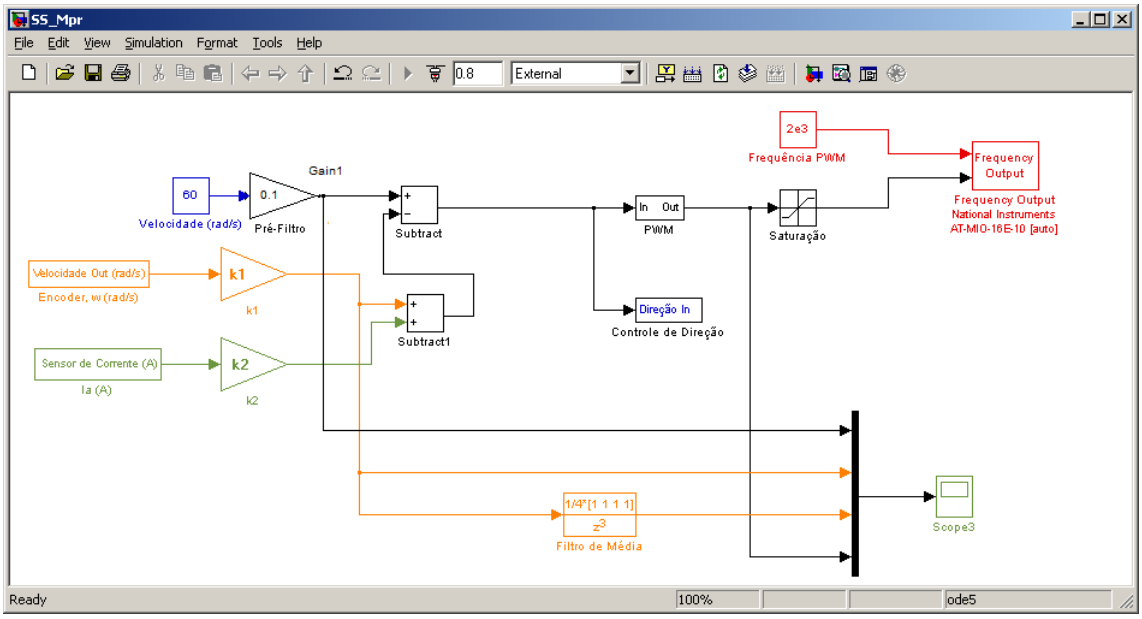


Figura 60 - Realimentação por espaço de estados, controle realizado por Matlab RTW

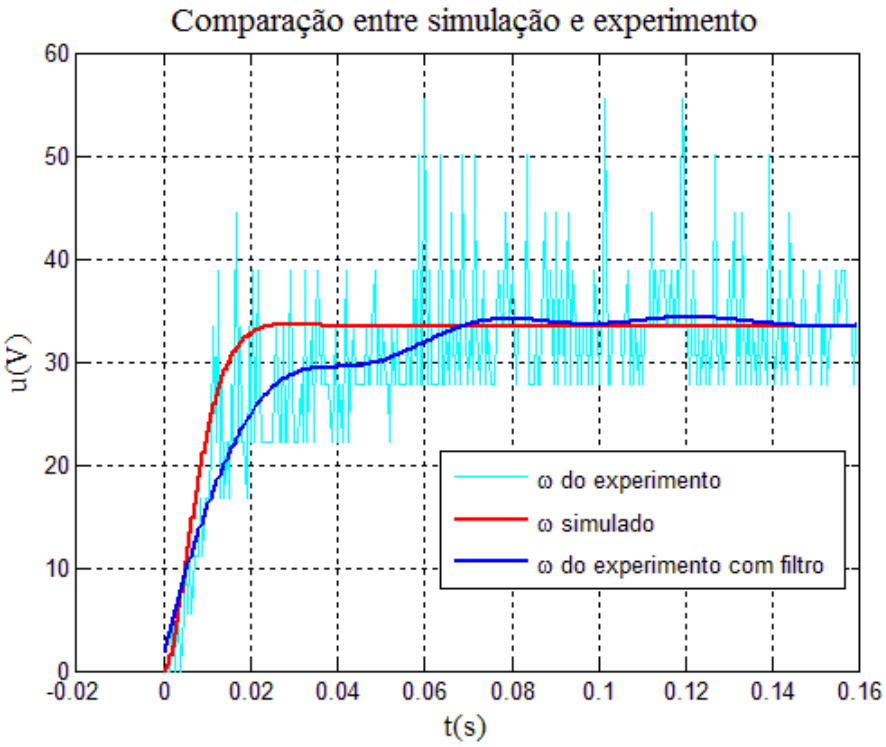
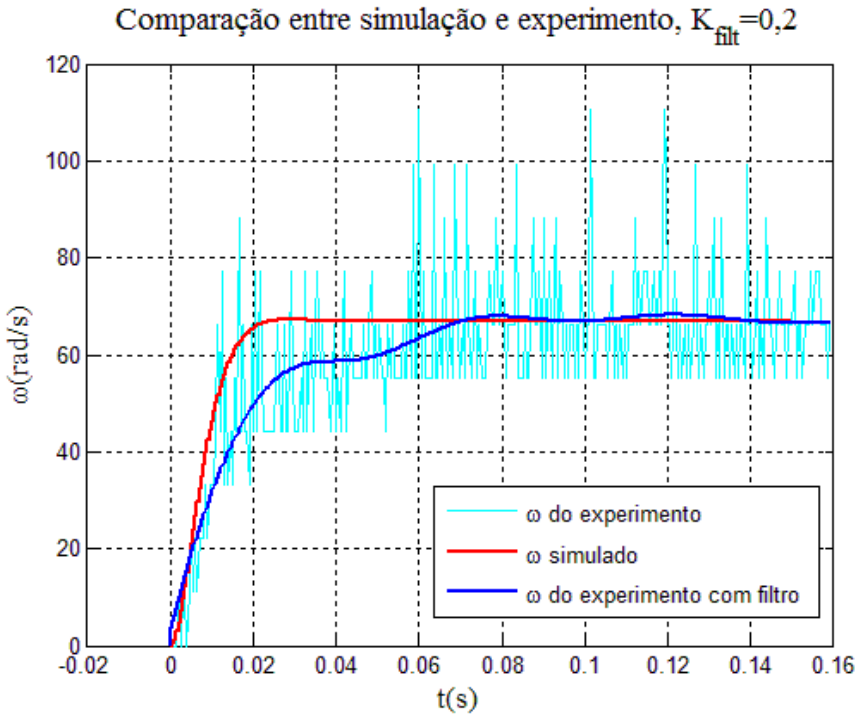


Figura 61 - Realimentação por espaço de estado, comparação de dados experimentais com simulados para pré-filtro Kfilt = 0,1

Comparando os resultados pode-se perceber que a curva do experimento está atrasada em relação à curva simulada e apresenta um erro de regime para um pré-filtro de K_{filt} = 0,1.

Para fazer essa correção no erro de regime, modificou-se o pré-filtro para K_{filt} = 0,2. Obteve-se o resultado da Figura 62.



.

Figura 62- Realimentação por espaço de estado, comparação de dados experimentais com simulados para pré-filtro Kfilt = 0,2

Percebe-se que ao invés de 30 rad/s, a curva para K_{filt}=0,2 estabilizou-se em 60 rad/s, dessa forma evitando o erro em regime.

Capítulo 5:

CONCLUSÃO

O trabalho realizado investigou a versatilidade do Toolbox RTW do Simulink na execução de tarefas em tempo real através de experimentos de controle digital aplicado a um motor CC.

Com a integração do Matlab RTW ao workspace do Matlab, obter a planta do sistema e realizar o cálculo dos controladores se mostrou um processo rápido e prático. Além disso o controle em tempo real utilizando o Matlab RTW foi funcional e se mostrou simples graças à interface em diagrama de blocos que o *Simulink* apresenta.

O *kit* didático, constituído da placa de interface, sensores, atuador L298N e o Motor CC, apresentaram alguns problemas tanto na transmissão quanto na recepção do sinal. Na transmissão, no L298, ocorre uma queda de tensão devido aos transistores BJT e um atraso do sinal devido sua lógica de acionamento. Na recepção, os dados se mostraram ruidosos tanto na aquisição pelo *encoder* quanto pelo tacômetro.

Os controladores realizados sofreram as conseqüências do atraso do sinal causado pelo L298. O controlador proporcional, o PI e o de realimentação por espaço de estados, apesar do atraso, funcionaram bem. Por outro lado, o controlador *Dead-Beat* não foi realizável, pois o atraso causado pelo L298 é significativo em seu funcionamento.

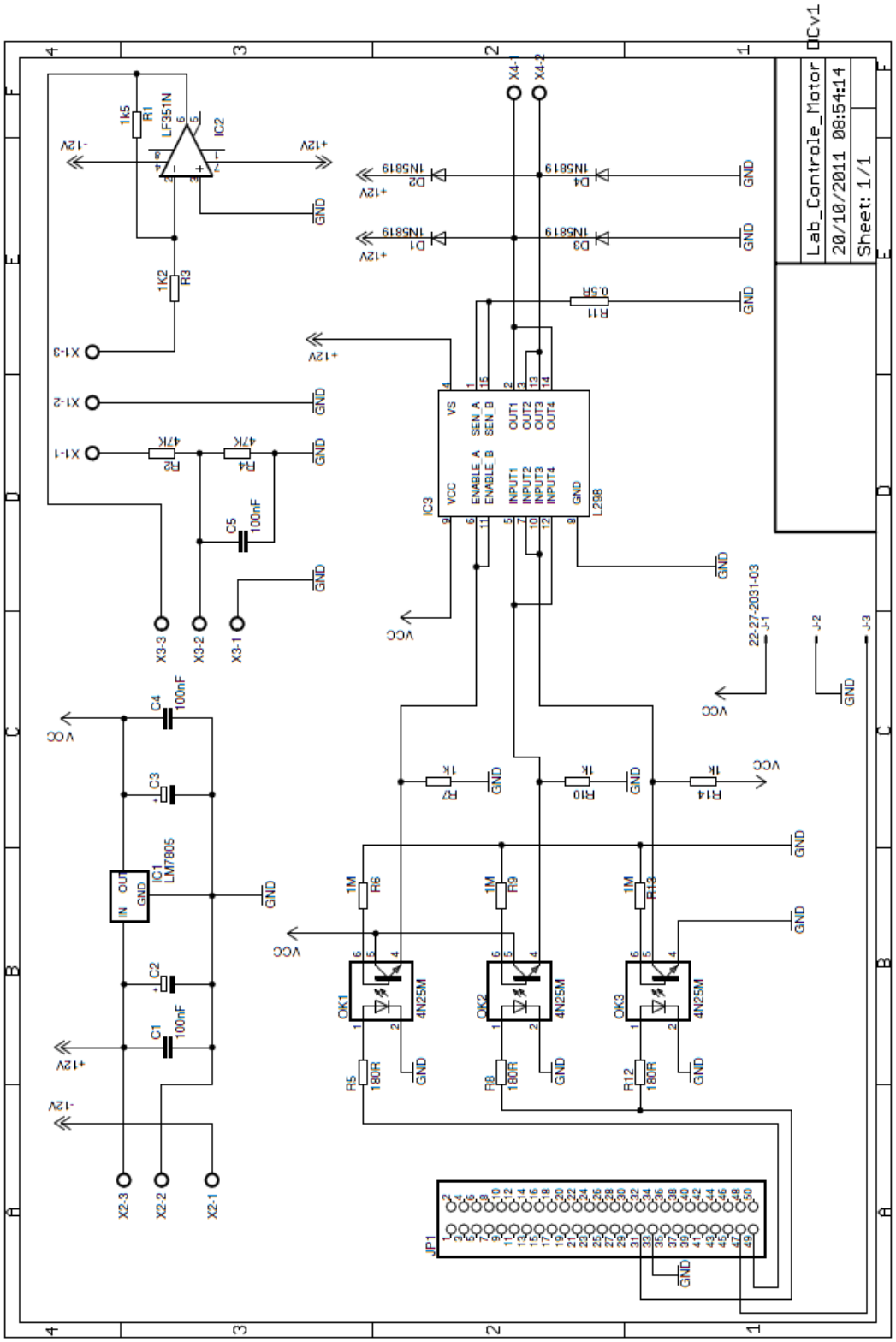
Para projetos futuros podem-se realizar melhorias no que diz respeito à transmissão, optar por alguma alternativa de ponte H, assim diminuindo a queda da potência do circuito interno do CI e não atrasando o sinal de forma significativa. Em relação à recepção, pode ser criado filtros capacitivos para que assim diminua o ruído do sinal enviado ao Matlab.

Outros projetos que podem ser encadeados é a realização de controladores usando o bloco *S-function* do Matlab ou até mesmo realizar técnicas de controle avançadas como por exemplo técnicas de controle robusto.

Com tudo o que foi apresentado, pode-se afirmar que o *kit* didático, apesar de algumas falhas, mostrou resultados satisfatórios para ser aproveitado nas disciplinas práticas do Laboratório de Controle de Sistemas e Controle Digital.

Anexos

1.Esqeuemático do projeto



Referências Bibliográficas

[1] A. C. M. NEWTON, Experimentos de controle digital em tempo real com auxílio do Matlab e Simulink. Monografia de Projeto de Conclusão de curso, São Carlos 2009.

[2] R. M. Corder, Controle em Tempo Real via Matlab/Simulink. Monografia de Projeto de Conclusão de curso, São Carlos 2009.

[3] Real-Time Windows Target - Run Simulink models on a PC in real time http://www.mathworks.com/products/rtwt/, acesso em novembro de 2011

[4] Real-Time Windows Target User's Guide, disponível em http://dali.feld.cvut.cz/ucebna/matlab/pdf_doc/rtw/rtwin_ug.pdf, acesso em novembro de 2011

[5] Fairchild Semiconductor Corporation. Datasheet TIL111, phototransistor optoisolator. Publicação eletrônica, 2000.

[6] STMicroelectronics. Datasheet L298, Dual Full Bridge Driver. Publicação eletrônica, 2000.

[7] Departamento de Engenharia Elétrica da Escola de Engenharia de São Carlos – USP. Disciplinas Online, SEL359, Script de aulas, PDF Documents.

[8] R. C. Dorf e R. H. Bishop. Sistemas de Controle Modernos. LTC editora, 2001.

[9] V. A. Oliveira, M. L. Aguiar, J. B. Vargas, Sistemas de Controle, Aulas de laboratório, EESC-USP 2005

[10] David E. Johnson, Fundamentos De Analise De Circuitos Eletricos, Ltc Editora, 2001