LEANDRO SOUZA GASPAR

PLATAFORMA PARA MEDIÇÃO DE VELOCIDADE EM MOTOR CC SEM O USO DE SENSOR ACOPLADO AO EIXO

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: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos
2011

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

Gaspar, Leandro Souza.
G153a Plataforma para medição de velocidade em motor CC sem o uso de sensor acoplado ao eixo. / Leandro Souza Gaspar ; orientador Evandro Luis Linhari Rodrigues -- 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. Motor de corrente contínua. 2. Sensor de velocidade. 3. Controle de motores. 4. IHM. 5. Aquisição. 6. Força eletromotriz. I. Titulo.

FOLHA DE APROVAÇÃO

Nome: Leandro Souza Gaspar

Título: "Plataforma para Medição de Velocidade em Motor CC Sem o Uso de Sensor"

Trabalho de Conclusão de Curso defendido e aprovado em 30 / 11 / 2011,

com NOTA 85 (oito , cinco), pela comissão julgadora:

Prof. Dr. Dennis Brandão - EESC/USP

Prof. Dr. Maximiliam Luppe - EESC/USP

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

Dedico este trabalho a todos que contribuíram para sua concretização, de forma direta e indireta, seja com ideias ou mesmo uma conversa informal. Professores, técnicos, funcionários, alunos, amigos, namorada, pais e familiares. Pequenos gestos somados fazem a diferença.

Agradecimentos

Aos antigos grupos de robótica USPdroids do ICMC-USP e GEAR da EESC-USP, atual grupo unificado WARTHOG ROBOTICS, que ofereceram toda a estrutura necessária para meu crescimento profissional durante minha graduação e também pelo apoio durante este projeto.

Aos Professores por toda a dedicação e apoio.

Ao apoio de minha família durante todos esses anos e em especial durante minha graduação.

Resumo

O presente trabalho aborda o desenvolvimento do método da força eletromotriz (F.E.M) para mensurar a velocidade de um motor corrente continua de imã permanente e com escovas sem a utilização de sensores acoplados ao seu eixo, evitando assim a utilização de um equipamento mecânico junto ao motor, que adicionaria volume e peso ao sistema, sujeito a desgastes físicos e que aumentam o custo final do projeto. Juntamente com o estudo deste método foi desenvolvido um sistema composto por uma plataforma de aquisição e controle, *software* para controle da plataforma, e uma interface de usuário utilizando o MATLAB para comunicação com a plataforma, desta forma o processo de aquisição de dados do motor fica facilitada. O Sistema é capaz de realizar o teste da resposta ao degrau para o motor CC, encontrar a constante de velocidade para o motor, dado este necessário para o uso do método da F.E.M, traçar gráficos comparativos para análise do método e também monitorar as velocidades medidas do motor pelo método da F.E.M e pelo encoder em tempo real. Todo o estudo e desenvolvimento do sistema foram pensados para que seja possível sua reprodução, assim toda a plataforma foi montada com componentes comuns.

Palavras-Chave: Motor de Corrente Contínua; Sensor de Velocidade; Controle de Motores; IHM; Aquisição; Força Eletromotriz.

Abstract

This work discusses the development of electromotive force (EMF) method to measure the speed of a permanent magnet direct current motor with brushes without the use of sensors coupled to its axis, thus avoiding the use of a mechanical equipment next to the motor, which would add weight and volume to the system, subject to physical damage and increasing the final cost of the project. With the study of this method was developed a system composed of a platform for acquisition and control, software to control the platform, and a user interface using MATLAB for communication with the platform, thus the data acquisition process of the motor is facilitated. The system is able to perform the step response test for the DC motor, to find the speed constant for the DC motor, this is necessary for the electromotive force (EMF) method, to plot graphs for comparative analysis of the sensorless method and with the encoder in real time. All the study and development of the system were designed to allow its reproduction, so the entire platform was mounted with common components, and MATLAB was chosen because it is a common tool for engineers, thus avoiding the use of proprietary equipment.

Keywords: Direct current motor; Speed sensor; Motor control; HMI; Acquisition; Electromotive force.

Lista de Figuras

Figura 1 – (a) Estator, (b) Rotor, (c) Comutador e (d). [3] 3
Figura 2 - Esquema eletromecânico do motor CC 4
Figura 3 - Sinal do encoder 5
Figura 4 - Ponte H genérica 6
Figura 5 - Encoder Óptico. [3] 7
Figura 6 - Sinal típico de um encoder 8
Figura 7 - Motor 2224 com encoder acoplado 8
Figura 8 - Sistema *Hardware*-*Software*-IHM 10
Figura 9 - Sinal nos terminais do motor 11
Figura 10 - Relação tensão gerada por velocidade do motor 2224 13
Figura 11 - Diagrama da Plataforma de Aquisição e Controle 15
Figura 12 - Sinais QEA e QEB do encoder 20
Figura 13 - Software de Simulação 21
Figura 14 - Fluxograma do programa do microcontrolador 22
Figura 15 - Interface Gráfica 23
Figura 16 - Curva da Relação Velocidade por F.E.M 26
Figura 17 - Curva da F.E.M pelo Tempo 27
Figura 18 - Curva Característica do Motor 27
Figura 19 - IHM no teste "Continuos" 28
Figura 20 - Acoplamento dos motores 29
Figura 21 - Curva da Relação Velocidade por F.E.M para o motor da Action 30
Figura 22 - Curva da Relação Velocidade por FEM para o motor Action com filtro 30
Figura 23 - Curva da F.E.M pelo Tempo para o motor da Action 31
Figura 24 - Curva característica para o motor da Action 31

Lista de Tabelas

Tabela 1 - Acionamento da Ponte-H 6
Tabela 2 – Relação da tensão gerada por RPM do motor 2224 12
Tabela 3- "BAUD RATES" para a serial [5]..... 18

Lista de abreviações e siglas

A/D	*Conversor Analógico para Digital*
AMPOP	*Amplificador Operacional*
ASCII	*American Standard Code for Information Interchange*
CA	*Corrente Alternada*
CC	*Corrente Contínua*
F.C.E.M	*Força Contra Eletromotriz*
F.E.M	*Força Eletromotriz*
FIFO	*First In, First Out*
IDE	*Integrated Development Environment*
IHM	*Interface Homem-Máquina*
PWM	*Pulse-Width Modulation*
RPM	*Rotações por Minuto*
USB	*Universal Serial Bus*

Sumário

Resumo ix
Abstract xi
Lista de Figuras xiii
Lista de Tabelas xv
Lista de abreviações e siglas xvii
1. **Introdução** 1
1.1. Motivação 2
1.2. Objetivos 2
1.3. Organização 2
2. **Fundamentos teóricos** 3
2.1. Motores CC com imã permanente e com escovas 3
2.1.1. Funcionamento do motor CC 3
2.1.2. Modelo do motor de corrente contínua 3
2.2. Acionamento para motores CC 5
2.3. Métodos para medição de velocidade em motores CC 6
2.3.1. Medição de velocidade para motores CC utilizando sensores 6
2.3.2. Medição de velocidade para motores CC sem o uso de sensores 8
3. **Material e método** 10
3.1. Estudo prático de um motor CC 10
3.2. Visão geral do Sistema 13
3.3. Plataforma de aquisição e controle 14
3.4. Ambiente de desenvolvimento 17
3.5. *Software* de simulação 21
3.6. Interface Homem-Máquina (IHM) 23
4. **Resultados** 26
5. **Conclusões** 32
Referências Bibliográficas 33
Apêndice A –Eletrônica da Plataforma de Aquisição e Controle 34
Apêndice B – Programa do Microcontrolador 35
Apêndice C – Programa da IHM 40
ANEXO A – Folha de dados do Motor 2224-006SR 45
ANEXO B – Folha de dados do Motor PM080-AU-S 46

1. Introdução

O progresso tecnológico e a automatização de vários processos que antes eram realizados manualmente provocaram um aumento no uso de pequenos motores, como por exemplo, na simples tarefa de abrir e fechar a janela de um carro, que antes era realizada manualmente e agora é realizado pela ação de um apertar de botão que aciona um motor elétrico. Nas aplicações onde é primordial a precisão tanto em velocidade quanto em posicionamento e não exige eficiência nem alta potência, está sendo dominada pelo emprego de motores de corrente continua (MCC). Este domínio vem em parte pela grande vantagem que apresentam os motores CC na simplicidade de seu controle. [1]

Na maioria das aplicações onde é exigido o controle de velocidade dos motores CC é necessário um sistema de controle em malha fechada sendo utilizado algum tipo de sensor para leitura da velocidade real do motor, como por exemplo, os *encoders* ópticos, tacogeradores e sensores de efeito hall. O uso desses sensores apresentam algumas desvantagens como maior volume ao motor, encarecimento do sistema e aumento do risco de falhas já que necessitam de um acoplamento mecânico junto ao motor.

Outra desvantagem encontrada no uso de sensores de velocidade está na dificuldade de se encontrar no mercado brasileiro motores que já possuam esses sensores incorporados, sendo necessária sua importação, encarecendo ainda mais o projeto.

Neste trabalho será apresentado um estudo geral de alguns métodos para medição de velocidade em motores CC sem o uso de sensores, o estudo teórico do método da força eletromotriz (F.E.M), o desenvolvimento de um sistema computacional com interface homem – máquina (IHM) para o estudo prático do método da F.E.M e também, generalizando, será indicada as possibilidades do uso deste sistema na caracterização de motores CC e finalizando será apresentado discussões praticas das vantagens e desvantagens do método em questão.

1.1. Motivação

Este projeto surgiu com o intuído de resolver um problema encontrado pelo grupo de Robótica Móvel *Uspdroids*, atualmente conhecido por *Warthog Robotics* que está vinculado ao LAR – Laboratório de Aprendizado de Robôs, da Universidade de São Paulo, Campus de São Carlos. Com o objetivo de tornar a robótica móvel mais acessível, e possibilitar ao grupo a construção de robôs móveis a um custo reduzido, surgiu a necessidade de se trocar os motores importados que eram utilizados por possuírem sensores de velocidade por motores de fabricação brasileira que tinham a desvantagem de não possuírem esses sensores, desta maneira foram propostos os objetivos deste trabalho.

1.2. Objetivos

Uma das dificuldades encontradas na construção de pequenos robôs móveis com controle de velocidade é o alto custo dos motores que possuem encoders para a medição de velocidade. Considerando isso como tema relevante, este trabalho propõe estudar e implementar uma plataforma capaz de realizar a leitura da velocidade de um motor corrente contínua de imã permanente e com escovas sem a utilização de sensores acoplados a esses motores.

1.3. Organização

Este trabalho está organizado da seguinte forma:

O Capítulo 2 apresenta conteúdos teóricos necessários para o entendimento do projeto.
O Capítulo 3 contém todo o desenvolvimento do projeto com a descrição dos materiais utilizados, assim como da metodologia adotada durante o projeto.
O Capítulo 4 apresenta os resultados dos testes realizados com a plataforma.
O Capítulo 5 apresenta as conclusões pertinentes a este trabalho.

2. Fundamentos teóricos

Neste capítulo serão apresentados conteúdos teóricos necessários para o entendimento deste projeto.

2.1. Motores CC com imã permanente e com escovas

Quando se tratando de motores de corrente contínua de baixa potência, da ordem de alguns amperes, e que são utilizados para propósitos gerais, os mais utilizados são os motores de corrente contínua com imã permanente e com escovas.

2.1.1. Funcionamento do motor CC

Este tipo de motor é composto de duas estruturas magnéticas, estator (imã permanente) (Figura 1a), e rotor (enrolamento/bobinas de armadura) (Figura 1b.) O rotor é um eletroímã constituído de um núcleo de ferro com enrolamentos em sua superfície que são alimentados por um sistema mecânico de comutação (Figura 1c). Esse sistema é formado por um comutador, solidário ao eixo do rotor, que possui uma superfície cilíndrica com diversas lâminas às quais são conectados os enrolamentos do rotor; e por escovas fixas (Figura 1d), que exercem pressão sobre o comutador e que são ligadas aos terminais de alimentação. O propósito do comutador é o de inverter a corrente na fase de rotação apropriada de forma a que o conjugado desenvolvido seja sempre na mesma direção. [2]

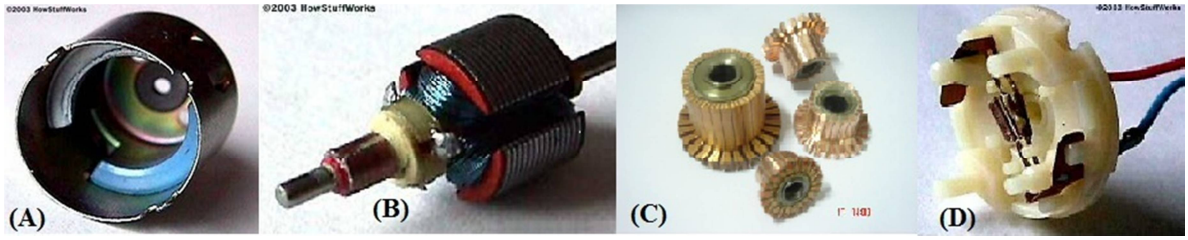


Figura 1 – (a) Estator, (b) Rotor, (c) Comutador e (d). [3]

2.1.2. Modelo do motor de corrente contínua

O motor de corrente contínua pode ser descrito matematicamente por meio de seus parâmetros elétricos e mecânicos.

Primeiramente temos o conjunto de espiras que constitui a armadura. Estas espiras caracterizam uma resistência elétrica de armadura denominada R_a e também uma indutância de armadura denominada L_a. A tensão induzida nas espiras pode ser deduzida como sendo direta-

mente proporcional à velocidade de rotação *ω(t)*, desde que $\vec{B}$ (fluxo magnético), *l* (comprimento do fio) e *d* (largura da espira) sejam constantes. Assim temos a Equação (1).

$$e(t) = K_e \times \omega(t) \tag{1}$$

Onde K_e é denominado de tensão ou constante de força contra eletromotriz (fcem). Também, pelo fato de B, l, d e Θ serem constantes, obtém-se que o torque desenvolvido no motor de corrente contínua é diretamente proporcional à intensidade da corrente que circula na armadura, como em (2).

$$T_e = K_t \times i_a(t) \tag{2}$$

Onde K_t é denominado constante de torque e i_a(t) é a corrente instantânea de armadura. Desde que o torque T_e e a tensão induzida $e(t)$ dependem de $\vec{B}, l, d, \Theta$ no sistema internacional de unidades o valor de K_e é numericamente igual ao valor de K_t. A estrutura que compõe a armadura representa uma massa em rotação, portanto, define-se para ela: momento de inercia J, coeficiente de atrito viscoso (proporcional a *ω)* B, e o coeficiente de atrito estático denotado F. O esquema eletromecânico do motor CC pode ser visto na Figura 2.

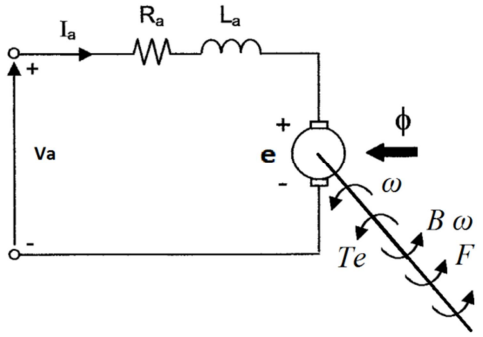


Figura 2 - Esquema eletromecânico do motor CC

Através do diagrama eletromecânico do motor de corrente contínua, obtém se o seguinte equacionamento dinâmico do motor de corrente contínua. [4]

$$v_a(t) = R_a i_a(t) + L_a \frac{d}{dt} i_a(t) + K_e \omega(t) \tag{3}$$

$$T_e(t) = K_t i_a(t) \tag{4}$$

$$T_e(t) = J\frac{d}{dt}\,\omega(t) + B\omega(t) + F \tag{5}$$

2.2. Acionamento para motores CC

Utilizando a técnica do PWM (*Pulse-Width Modulation*) podemos controlar a potência dos motores CC, através de um sinal digital. Configurando uma frequência de pulso alta o suficiente para que o motor não responda a ela, o motor irá se comportar de forma diretamente proporcional à tensão média do PWM. Um critério de escolha da frequência de acionamento f_a é fazer com que a indutância L_a do motor resulte em uma grande impedância nesta frequência, desta forma temos a equação (**6**).

$$2\pi f_a L_a \gg R_a \tag{6}$$

A tensão média do PWM é dada por (7).

$$V_m = \frac{tp}{T} \times V_{pulso} \tag{7}$$

Onde:

V_m – Tensão média

tp – Duração do pulso em nível alto

T – Período da onda

V_{pulso} – Tensão máxima do pulso

Na Figura 3, observa-se a frequência constante e a possível variação da largura de pulso do sinal PWM.

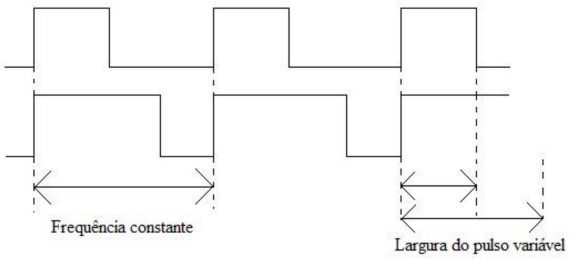


Figura 3 - Sinal do encoder

Como o sinal PWM é gerado a partir de um microprocessador e este possuir baixa capacidade de corrente em suas portas, e ainda por termos a necessidade de inversão de sentido de rotação do motor utilizamos uma ponte-H para realizar a interface entre microprocessador e o motor.

Uma ponte-H genérica pode ser vista na Figura 4.

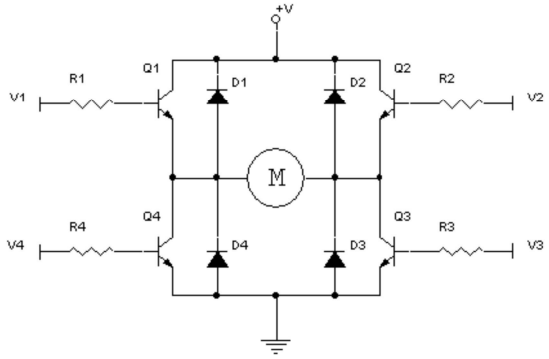


Figura 4 - Ponte H genérica

Seu funcionamento segue conforme Tabela 1.

Tabela 1 - Acionamento da Ponte-H

V1	V2	V3	V4	Resultado
H	L	H	L	Q1 e Q3 fechados, corrente da esquerda para a direita.
L	H	L	H	Q2, Q4 fechados, corrente da direita para a esquerda.
L	L	L	L	Motor desligado.

2.3. Métodos para medição de velocidade em motores CC

2.3.1. Medição de velocidade para motores CC utilizando sensores

Uma das formas de se medir a velocidade rotacional de um motor é utilizar algum dispositivo que transforme diretamente esta grandeza mecânica em uma grandeza elétrica. O dispositivo mais clássico que realiza esta operação é o tacogerador.

- **Tacogerador**

O tacogerador possui ímãs permanentes no estator, com a função de produzir um campo magnético. No rotor bobinado é gerada uma tensão contínua de amplitude proporcional à rotação e de polaridade que depende do sentido de giro da máquina elétrica a qual ele está acoplado.

- **Encoder**

O encoder é um dispositivo que gera um pulso para um determinado incremento de rotação do eixo ao qual está acoplado (encoder rotativo). Embora seja mais utilizado no controle de posição, o encoder também é utilizado para medir velocidade, uma vez que medindo o ângulo total percorrida (através da contagem dos pulsos na saída do encoder) e o tempo necessário para este ângulo ser percorrida, consegue-se calcular a velocidade. Há vários tipos em uso: magnético, contato, resistivo, e óptico.

O principio de funcionamento básico do encoder óptico pode ser observado na Figura 5.

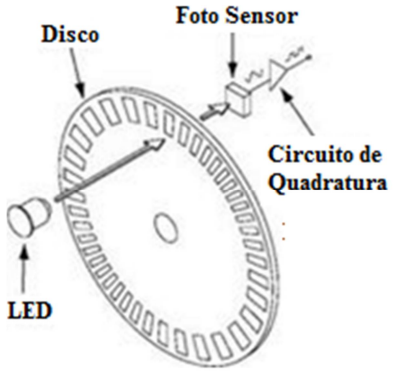


Figura 5 - Encoder Óptico. [3]

Onde um disco perfurado é fixado ao eixo do motor, quando esse disco entra em movimento, essas perfurações interrompem, periodicamente, um feixe óptico que é detectado por um foto sensor gerando os pulsos.

Um sinal típico deste tipo de encoder pode ser visto na Figura 6, onde se observa que a velocidade do motor está aumentando já que a frequência dos pulsos aumenta.

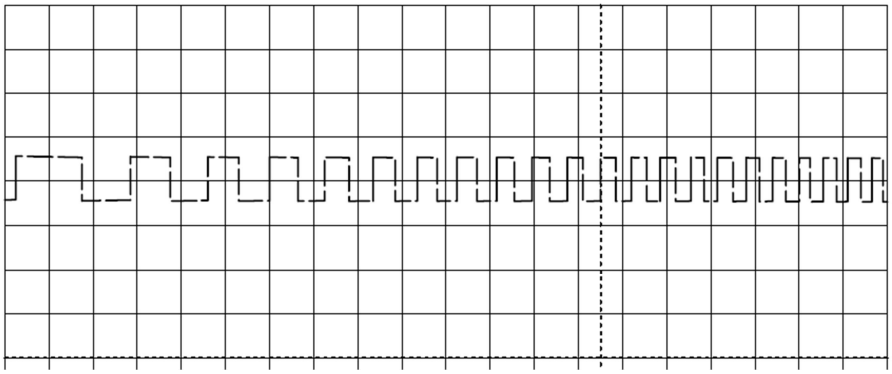


Figura 6 - Sinal típico de um encoder

A Equação (8) nos mostra a relação entre o período dos pulsos e a velocidade do motor.

$$n = \frac{60}{T \times \mathrm{P}} \tag{8}$$

Onde:

T – Período dos pulsos

n – Velocidade em RPM

P – Quantidade de pulsos por volta do encoder

O motor 2224 utilizado neste projeto possui um encoder magnético de 64 pulsos por volta acoplado ao seu eixo, como pode ser observado na Figura 7.

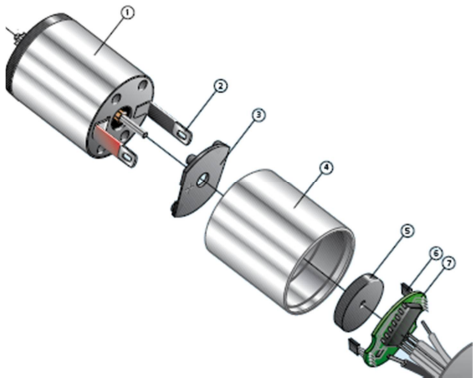


Figura 7 - Motor 2224 com encoder acoplado

2.3.2. Medição de velocidade para motores CC sem o uso de sensores

Além dos dispositivos utilizados para medição de velocidade, mencionados no item acima, existem outras técnicas que dispensam o uso de sensores, ou seja, dispensam o uso de dispositivos mecânicos externos ao motor.

- **Leitura de "Ripple"**

A corrente em um motor CC é composta por uma componente contínua, referente ao torque, sobre a qual se superpõe uma componente alternada conhecida como *ripple*. As características da componente *ripple* dependem da velocidade de giro e dos parâmetros do motor. [1]
A componente ripple da corrente do motor é decorrente do chaveamento das bobinas do rotor através do comutador do motor CC, assim a frequência do *ripple* é proporcional à velocidade do rotor como indicado na Equação (9).

$$f = \frac{2pK_d n}{60\eta} \tag{9}$$

Onde p é o número de pares de polo, K_d é o número de seguimentos do comutador, n é a velocidade em RPM e η é o maior denominador comum de 2p e K.

- **Leitura da força eletromotriz (F.E.M)**

Este método parte do principio de que um motor de corrente contínua pode também ser um gerador elétrico de corrente contínua, desta maneira observando a equação (3), colocando o motor para operar como gerador elétrico, ou seja, fornecendo $\omega(t)$, e com o gerador operando sem carga i_a=0, encontramos a Equação (10).

$$v_a(t) = K_e\omega(t) = e(t) \tag{10}$$

Indicando que a tensão gerada nos terminais do motor/gerador é igual à força eletromotriz, que por sua vez é diretamente proporcional à velocidade angular do rotor.

Portanto, para a leitura da velocidade, basta desligar o motor periodicamente durante um pequeno intervalo de tempo, suficiente para que ocorra o transitório entre o comportamento de motor para gerador elétrico, e então realizar a aquisição do valor da tensão gerada v_a que dividida por K_e nos fornece o valor da velocidade.

Este é o método escolhido para ser desenvolvido neste trabalho, assim maiores detalhe serão apresentados no capítulo seguinte.

3. Material e método

Para atingir o objetivo final de se desenvolver uma plataforma de leitura de velocidade em motores CC sem o uso de sensores foi desenvolvido um Sistema *Hardware-Software*-IHM, Figura 8, para que fosse possível automatizar o processo de aquisição da curva velocidade por força eletromotriz para qualquer motor CC e testar, em tempo real, o funcionamento do método.

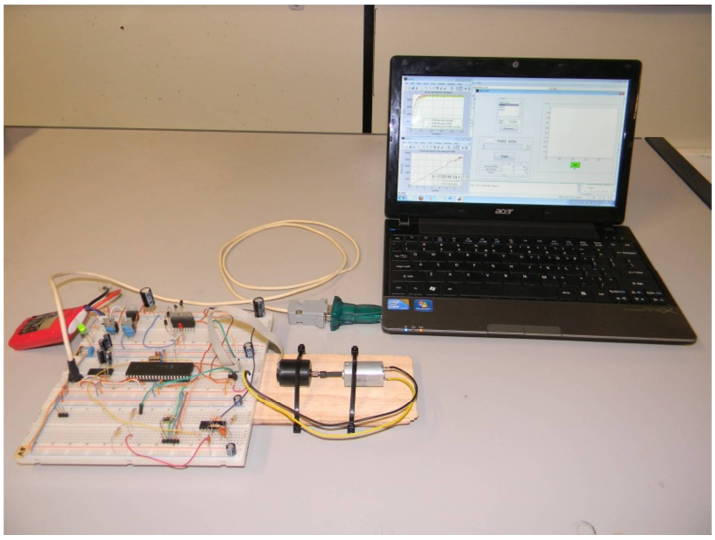


Figura 8 - Sistema *Hardware-Software*-IHM

Posteriormente foi proposta a generalização deste Sistema para um sistema capaz de realizar a caracterização de motores CC, oferecendo a resposta transitória do motor à entrada degrau.

Inicialmente foi realizado um estudo prático e especifico em um motor do fabricante Faulhaber, quanto as suas características funcionando como gerador elétrico. O principal dado obtido neste estudo é o tempo necessário para que ocorra a transição do estado motor para o estado gerador do motor CC. Com base nessas informações foi projetado o *hardware* necessário para automatizar o processo de aquisição da F.E.M, a aquisição do sinal do encoder provendo a velocidade real do motor em RPM, e a comunicação serial para comunicação com o microcomputador, onde serão processados os dados através de uma interface gráfica para o usuário requisitar ações e recolher dados do motor.

3.1. Estudo prático de um motor CC

Todo o estudo inicial deste trabalho é baseado em apenas um motor CC, no Capítulo Resultados o leitor poderá observar a generalização para outros motores CC.

O motor CC escolhido para estudo é o 2224-006SR do fabricante Faulhaber. Este motor foi escolhido por já possuir acoplado ao seu eixo um encoder incremental magnético de 64 pulsos por volta, sendo ideal para a realização do estudo proposto. É um motor de 6V de tensão nominal e 4,55W de potência máxima. Sua folha de dados pode ser visto no Anexo A.

Utilizando um gerador de função foi gerado uma onda quadrada de aproximadamente *60 Hz* com seu *duty cycle* negativo em 85%. Ligando este sinal a uma ponte H (L298) acionou-se o motor para que fosse possível observar a tensão gerada pelo motor quando este está desligado, ou seja, quando este está girando por inercia e se comportando como um gerador elétrico. O sinal observado nos terminais do motor pode ser visto na Figura 9.

Em (A) pode ser visto um período completo da onda quadrada, com semiciclo positivo de 5 Volts de pico. Observando o estado gerador, quando a onda quadrada está em seu semiciclo negativo, temos uma componente CC decorrente da força eletromotriz com tensão máxima de 3 Volts, é este o nível CC que estamos interessados em mensurar. O sinal alternado que aparece sobreposto ao nível CC da força eletromotriz é decorrente do chaveamento das bobinas do rotor, ou seja, esta ondulação é introduzida pelo comutador. Para o método aqui proposto, que é o método da F.E.M, esse sinal alternado é indesejado, já que adiciona ruído ao nível CC que desejamos mensurar. Já para o método do *ripple* é justamente esta ondulação o sinal desejado, sendo sua frequência diretamente proporcional à velocidade do rotor, dependendo do número de seguimentos que o comutador tenha, situação análoga à explicitada no item 2.3.2.

Em (B) pode ser visto mais detalhadamente o momento do chaveamento de 5 para 0 Volts na tensão de alimentação do motor, sendo possível observar o comportamento do transitório do estado motor para gerador, este transitório tem um tempo aproximado de 30µs. O tempo do transitório é importante para que seja definido o momento em que será realizada a leitura da tensão gerada, evitando a leitura de valores de tensão do transitório.

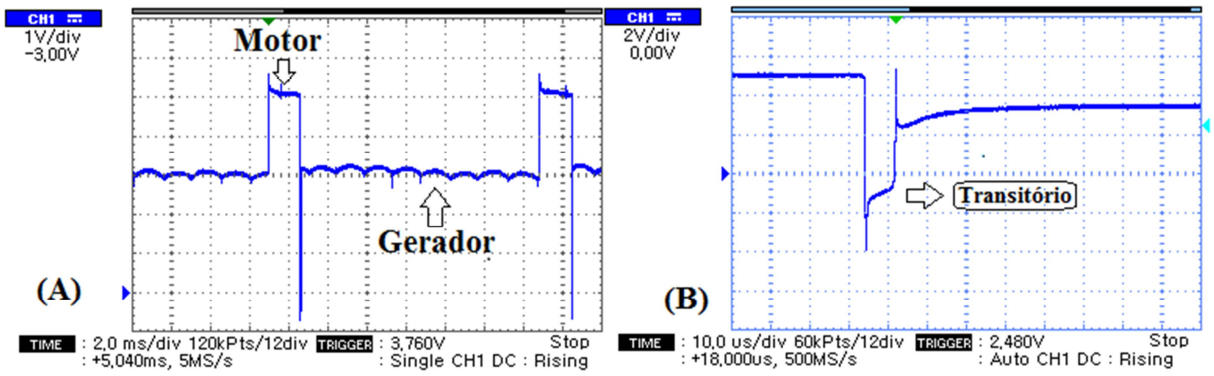


Figura 9 - Sinal nos terminais do motor

Com o intuito de traçar a curva de velocidade do rotor pela tensão gerada foi ligado no canal 2 do osciloscópio o sinal proveniente do encoder incrementar, desta maneira foi possível fazer a leitura da frequência do sinal do encoder, assim como a tensão gerada no estado gerador. Variando o *duty cicle* da onda quadrada e fazendo as medidas obteve-se a Tabela 2.

Tabela 2 – Relação da tensão gerada por RPM do motor 2224

f(Hz)	*n(RPM)*	$V_a(V)$
1000	937,5	0,70
2000	1875	1,44
3000	2812,5	2,10
4000	3750	2,82
5000	4687,5	3,53
6000	5625	4,20
7000	6562,5	4,85
8000	7500	5,70
9000	8437,5	6,30

A Equação (8) foi utilizada para fazer a conversão da frequência do sinal do encoder para velocidade em RPM.

Utilizando o *Software* MATLAB, e a partir da Tabela 2, foram plotados os pontos experimentais da relação tensão gerada por velocidade do motor 2224 (Figura 10). Utilizando a função `polyfit` do MATLAB é possível encontrar os termos da função da reta, equação (11), que melhor representa os pontos plotados no gráfico, onde n é a velocidade em RPM, V_a a tensão gerada pelo rotor, a_1 o coeficiente angular da reta e a_0 o coeficiente linear da reta.

$$n = a_1 V_a + a_0 \tag{11}$$

Utilizando a função `polyval` do MATLAB é possível encontrar os valores numéricos da reta (11), desta maneira, plotando esses valores temos a reta aproximada no gráfico da Figura 10.

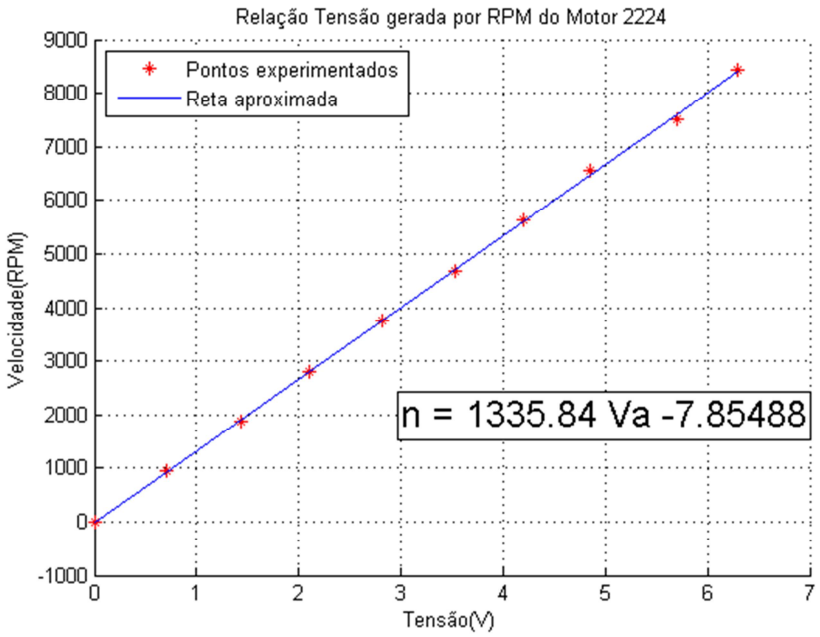


Figura 10 - Relação tensão gerada por velocidade do motor 2224

O termo a_1 da reta é chamado de K_n (constante de velocidade do motor) que é o inverso de K_e (constante de força contra eletromotriz) da equação (1). Conforme o *datasheet* do motor 2224, Anexo A, a constante K_n é de 1380 RPM/V muito próximo do valor encontrado experimentalmente de 1335 RPM/V, indicando que a metodologia utilizada é satisfatória para a leitura da força eletromotriz.

3.2. Visão geral do Sistema

Como já foi explicitado na introdução deste Capítulo o Sistema proposto é composto por hardware de interface com o motor CC, programa do microcontrolador para controle do hardware e interface gráfica do usuário. O funcionamento geral do Sistema *Hardware-Software*-IHM ocorre da seguinte maneira:

- Primeiramente o programa do microcontrolador inicializa todas variáveis e periféricos necessários, feito isso, o algorítimo fica em estado de espera, aguardando algum pedido da IHM via porta serial.

- Na IHM o usuário deve realizar a conexão da porta serial, configurando qual a porta e qual a velocidade. Estando o sistema conectado o usuário pode requisitar do microcontrolador dois tipos de teste com o motor CC.

- O primeiro teste possível, chamado de *Step*, consiste em ligar o motor CC durante meio segundo (500ms) realizando uma amostragem da velocidade via encoder e da F.E.M do motor a cada 5ms. O microcontrolador transmite então esses dados para a IHM onde são montados três gráficos, um da relação velocidade por tensão gerada, outro da tensão gerada pelo tempo, e o ultimo gráfico contendo a velocidade mensurada pelo encoder e a velocidade mensurada através da F.E.M pelo tempo.

- O segundo teste possível, chamado de *Continuos*, consiste em ligar o motor continuamente até que seja realizada uma requisição pelo usuário para que seja desligado. Desta forma, a IHM recebe continuamente amostras da velocidade proveniente do encoder e da velocidade proveniente da F.E.M a cada 100ms, plotando um gráfico em tempo real com essas velocidades.

3.3. Plataforma de aquisição e controle

A plataforma de aquisição e controle consiste no *hardware* necessário para realizar a aquisição de dados dos sensores (encoder e A/D), processamento destes dados e a comunicação com o microcomputador que incorpora a interface do usuário.

A fim de atender um dos objetivos deste projeto que é diminuir o custo final de um sistema de controle para motores CC, a plataforma de aquisição e controle foi projetada, construída e programada durante este trabalho, ficando o usuário independente de qualquer tecnologia reservada ou proprietária, podendo substituir facilmente qualquer componente do sistema ou mesmo transpor os códigos para outros microcontroladores.

Uma visão geral da plataforma de aquisição e controle pode ser vista na Figura 11. O projeto completo do *hardware* encontra-se no Apêndice A.

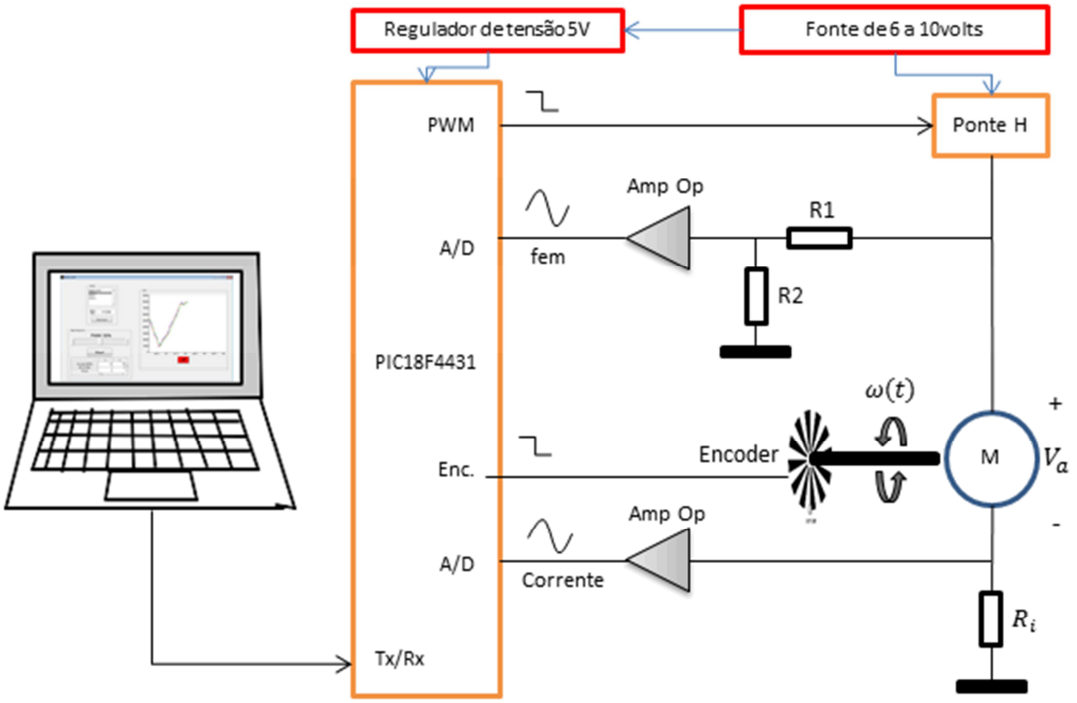


Figura 11 - Diagrama da Plataforma de Aquisição e Controle

Microcontrolador

Dentre todo o leque de opções disponíveis no mercado de microcontroladores algumas opções foram escolhidas para estudo e futura escolha de qual a opção que melhor se enquadra na aplicação. Em ordem crescente de custo e complexidade os microcontroladores estudados são: PIC18F, DsPIC33F, e a Família ARM7 da Atmel. Atendendo a exigência de um projeto de baixo custo optou-se pelo PIC18F4431 do fabricante Microchip [5]. Este é um microcontrolador projetado para aplicações onde envolva o controle de motores, oferecendo uma gama de periféricos necessários para este projeto, como:

- **"14-Bit Power Control PWM Module":** Modulo para geração de sinal PWM com resolução de 14-Bit.
- **"Motion Feedback Module w/ Quadrature Encoder Interface" :** Modulo para leitura da frequência do sinal gerado por um encoder incremental.
- **"High-Speed, 200 ksps 10-Bit A/D Converter":** Conversor Analógico/Digital de alta velocidade com resolução de 10-Bit.
- **"Enhanced USART module":** Modulo para comunicação serial.
- **"40 MHz Max Speed":** Velocidade máxima do clock de 40 MHz.

- **"8 x 8 Hardware Multiplier":** Multiplicador implementado em hardware capaz de realizar multiplicações de 8-bit em um único ciclo de máquina, reduzindo de 6,9us para 100ns o tempo de uma multiplicação, para um clock de 40 MHz.

Aquisição da força eletromotriz

Para a aquisição da tensão gerada pelo motor quando em estado gerador, ou seja, quando o PWM está desligado, adicionou-se um divisor por 2 de tensão ao terminal positivo do motor, fazendo $R_1 = R_2 = 100\ K\Omega$ e então se adicionou um *buffer* (amplificador operacional com ganho 1) para que o A/D do microcontrolador não carregasse o circuito anterior.

O conversor Analógico/Digital do microcontrolador foi programado para utilizar como tensão de referência sua própria alimentação, neste caso, 5 Volts. Isto se fez porque a alimentação do microcontrolador é proveniente de um regulador de tensão, sendo esta uma alimentação mais estável quanto a flutuações, garantindo uma tensão de referência confiável. Desta forma a tensão máxima admitida pelo sistema para a leitura da F.E.M é de 10 Volts já que a tensão V_a é dividida por 2 no divisor resistivo. Esta configuração para o conversor A/D também proporciona liberdade na tensão de alimentação do motor podendo ser de no mínimo 6 Volts, fixado pelo regulador de tensão de 5 Volts, e de no máximo 10 volts para que a F.E.M também não ultrapasse esse valor, que é o máximo aceito pelo conversor A/D nesse caso.

Aquisição do sinal do *encoder*

Para a aquisição do sinal do encoder foi utilizado o periférico *"Quadrature Encoder Interface"* que o microcontrolador PIC18F4431 possui. Com este periférico dispensou-se qualquer outro dispositivo de interface, sendo, portanto o encoder ligado diretamente no microcontrolador. Para isto, são utilizados dois fios, QEA e QEB que fornecem dois sinais de quadratura defasados de 45^o graus, necessários para a possibilidade de detecção de sentido de rotação do motor.

O funcionamento de um encoder incremental foi explicado no item 2.3.1.

Ponte H

O componente utilizado para a ponte H foi o L298. Este componente possui duas pontes H em um mesmo encapsulamento com capacidade de acionar cargas de 50 Volts com até 2A em modo continuo e 3A para correntes de pico. Esses valores são para cada um dos dois canais, se

utilizados em modo paralelo, os valores de corrente dobram, podendo chavear correntes de até 4A.

Com esta ponte é possível, além de chavear o motor com o sinal do PWM, inverter seu sentido de rotação. Porém esta função não foi implementada por não ser necessária ao projeto.

3.4. Ambiente de desenvolvimento

Para o desenvolvimento do programa para o microcontrolador PIC18F4431 foi utilizado o Ambiente de Desenvolvimento Integrado MPLAB do próprio fabricante do microcontrolador, a MICROCHIP. Este é um *software* distribuído gratuitamente pelo fabricante.

Segundo o fabricante [5]

> [...] MPLAB Integrated Development Environment (IDE) é um conjunto de ferramentas livres integrado para o desenvolvimento de aplicações embarcadas utilizando o PIC da Microchip® e também os dsPIC® microcontroladores. MPLAB IDE é executado como um aplicativo de 32 bits em MS Windows® e inclui uma série de componentes de *software livre* para desenvolvimento de aplicações rápidas e para depuração. MPLAB IDE também serve como uma interface de usuário unificada e gráfica para *softwares* Microchip adicionais e *software* de terceiros e ferramentas de desenvolvimento de *hardware* [...].

Para a programação do microcontrolador utilizando a linguagem de programação C foi utilizado o software adicional MPLAB C18 C Compiler [6] que deve ser instaldo de maneira conjunta com o MPLAB (IDE).

Descrisão do Programa de Controle

Com todos os requisitos de projeto em mente, o método a ser desenvolvido e o material necessário como descrito no item 3, foi projetado o programa de controle para o microcontrolodor como pode ser visto no fluxograma da Figura 14.

No Apêndice B encontra-se o programa escrito em linguagem C utilizando o MPLAB (IDE). A seguir serão explanados os principais pontos do programa de controle.

Inicialização

Este projeto exige do microcontrolador o uso de vários periféricos. Para que eles possam ser utilizados o programa deve inicializá-los, configurando todos os seus registradores de acordo com as funcionalidades requeridas. Os cinco periféricos que devem ser inicializados são: porta serial, *timer*, modulo de controle PWM, interface de quadratura do encoder e o conversor A/D.

Porta Serial

Para a comunicação serial foi escolhida a velocidade de 115,2Kbps na configuração assincrona. O *Baudrate* da comunicação serial deve ser configurado utilizando os registradores SPBRG e SPBRGH. Conforme Tabela 3 retirada da folha de dados do microcontrolador PIC18F4431 e considerando que $F_{OSC} = 40MHz$ temos o valor 86 para o registrador SPBRG.

Tabela 3- "BAUD RATES" para a serial [5].

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	33332	0.300	0.00	16665	0.300	0.00	8332	0.300	-0.01	6665
1.2	1.200	0.00	8332	1.200	0.02	4165	1.200	0.02	2082	1.200	-0.04	1665
2.4	2.400	0.02	4165	2.400	0.02	2082	2.402	0.06	1040	2.400	-0.04	832
9.6	9.606	0.06	1040	9.596	-0.03	520	9.615	0.16	259	9.615	-0.16	207
19.2	19.193	-0.03	520	19.231	0.16	259	19.231	0.16	129	19.230	-0.16	103
57.6	57.803	0.35	172	57.471	-0.22	86	58.140	0.94	42	57.142	0.79	34
115.2	114.943	-0.22	86	116.279	0.94	42	113.636	-1.36	21	117.647	-2.12	16

Os registradores TXSTA e RCSTA também devem ser configurados. Maiores detalhes podem ser encontrados na folha de dados do microcontrolador.

A comunicação serial entre microcontrolador (plataforma de aquisição e controle) e microcomputador (IHM) ocorre de maneira bidirecional, com a IHM realizando pedidos e a plataforma transmitindo dados.

A mensagem de pedido, transmitido pela IHM e recebido pelo microcontrolador, é composta por quatro caracteres o primero podendo ser ‘S’ou ‘C’ indicando dois tipos de rotinas distintas. O segundo caracter é escrito como um inteiro sem sinal, ou seja, é um valor de 0 a 255, porém serão usados apenas valores de 0 a 100 para indicar a porcentagem do sinal PWM

requerida pelo usuário. Os caracteres seguintes não estão sendo usados, foram adicionados para serem usados em futuras aplicações.

A mensagem de transmissão de dados, transmitida do microcontrolador para a IHM, é formada por caracteres ASCII. A cada pacote transmitido forma-se uma linha contendo três colunas, sendo cada uma das colunas um dado diferente, a primeira coluna contém a velocidade em RPM mensurada pelo encoder, a segunda coluna contém o do conversor A/D para a F.E.M e a terceira coluna pode conter o *timer* do microcontrolador ou o valor do conversor A/D para a corrente do motor.

Timer

O timer do microcontrolador foi utilizado para que fosse fixado o tempo de amostragem dos valores de velocidade dos sensores. Desta forma a cada iteração do programa existe uma rotina de espera para que o timer atinja o tempo desejado. Os tempos escolhidos para as amostragens são de 5ms para a funcão *Step* e de 100ms para a função *Continuos*.

Modulo de controle PWM

Para a configuração do PWM são utilizados os registradores PTCON0, PWMCON0 e PTCON1 onde são configurados quesitos como canais, portas, postscale e prescale. Os registradores PTPERH e PTPERL são resposáveis pela frequência do PWM, desta maneira, seguindo recomendaçõs da folha de dados do microcontrolador PIC18F4431, foi utilizado o valor 01FFh para estes registradores, assim a frequência do PWM será de 19,5KHz, com precisão de 11bits para o "duty cicle" do PWM, com isso os registradores PDC0L e PDC0H podem ser carregados com valores de 0 a 2047 para a escolha de 0 a 100% do "duty cicle" do sinal PWM.

Interface de quadratura do encoder

Para este periférico são três registradores os responsáveis por sua configuração, QEICON, CAP1CON e T5CON. Deve ser selecionado o modo como o periodo do sinal de quadratura do encoder será lido, como o encoder utilizado neste projeto é um encoder de baixa resolução, no caso 64 pulsos por volta, foi escolhido o modo que oferece a maior resolução possivel. Este modo realiza a contagem de tempo entre a borda de subida de um canal e a borda de descida do outro canal, como indicado na Figura 12.

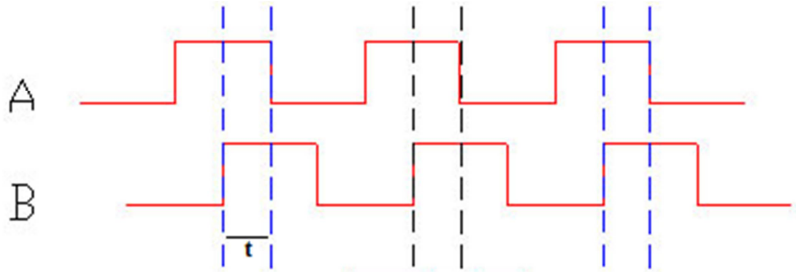


Figura 12 - Sinais QEA e QEB do encoder

Para uma velocidade de 10000rpm, utilizando a equação (8), o período de um dos sinais do encoder é de 93,75 µs, como o tempo entre as bordas de subida de um canal e a de descida do outro canal é um quarto do período de um sinal, temos um tempo de 23,43 µs, sendo este o tempo minimo que o *timer* do encoder, neste caso o *timer 5*, deve ser capaz de mensurar para a leitura de velocidades de até 10000rpm.

Para satisfazer a necessidade acima o timer 5 foi configurado com um *prescale* de 1:8, sendo o *clock* de 10MHz e o *timer* de 16-bit, o tempo minimo medido pelo timer é de 0,8 µs, mais do que suficiente para leituras de até 10000rpm. Porém este tempo minimo para o timer não pode ser muito grande para que se tenha uma precisão de leitura suficiente.

O tempo máximo que o timer 5 pode contar é de 52,4 ms o que corresponde a aproximadamente 4 rpm, sendo esta a velocidade mínima possível de ser lida.

Conversor A/D

Para o periférico conversor anlógico-digital são vários os registradores para configuração. As configurações principais são a escolha dos canais A/D, quais serão as tensões de referência, configurações referentes à memoria FIFO para registrar os valores lidos, *clock* de conversão, entre outros.

Para realizar a leitura da F.E.M o programa realiza a seguinte rotina. Primeiramente o motor é desligado, fazendo o sinal do PWM ir a zero; uma rotina de *delay* é adicionada para esperar o transitório do motor, no caso do motor 2224 o transitório médio é de 30µs, para o delay foi utilizado 300µs para garantia de qualidade no sinal lido. Após este *delay* inicia-se a conversão, o tempo de conversão é variável, desta maneira utiliza-se a função While( ) para espera da conversão. Realizada a conversão o motor é acionado novamente.

O tempo médio que o motor deve permanecer desligado para que ocorra a leitura da F.E.M é a soma do tempo do transitório mais o tempo para que ocorra a conversão A/D, para este projeto utilizando o motor 2224 este tempo é de no máximo 350µs.

3.5. *Software* de simulação

Durante o desenvolvimento do algoritimo para o microcontrolador foi utilizado o *Software* ISIS-PROTEUS v7.8 SP2 onde foi montado todo o hardware incluindo o microcontrolador. Neste Software é possivel rodar o programa no microcontrolador, possibilitando testes rápidos e eficazes. Posteriormente foi utilizado o Software *Free Virtual Serial Ports* capaz de criar um duto de dados entre duas portas seriais virtuais, desta maneira foi possivel comunicar o hardware simulado no ISIS com o MATLAB para testes com a IHM. Na Figura 13 está a imagem do software ISIS com o esquemático do hardware testado.

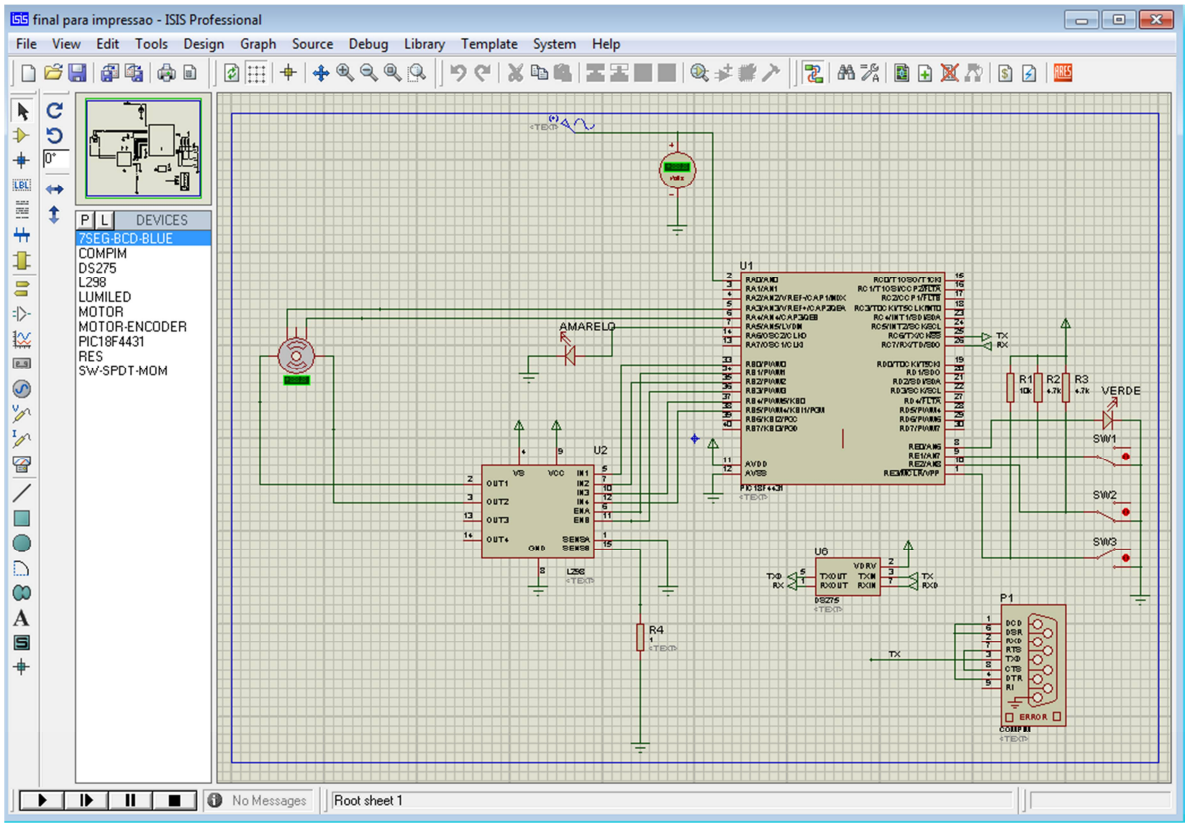


Figura 13 - Software de Simulação

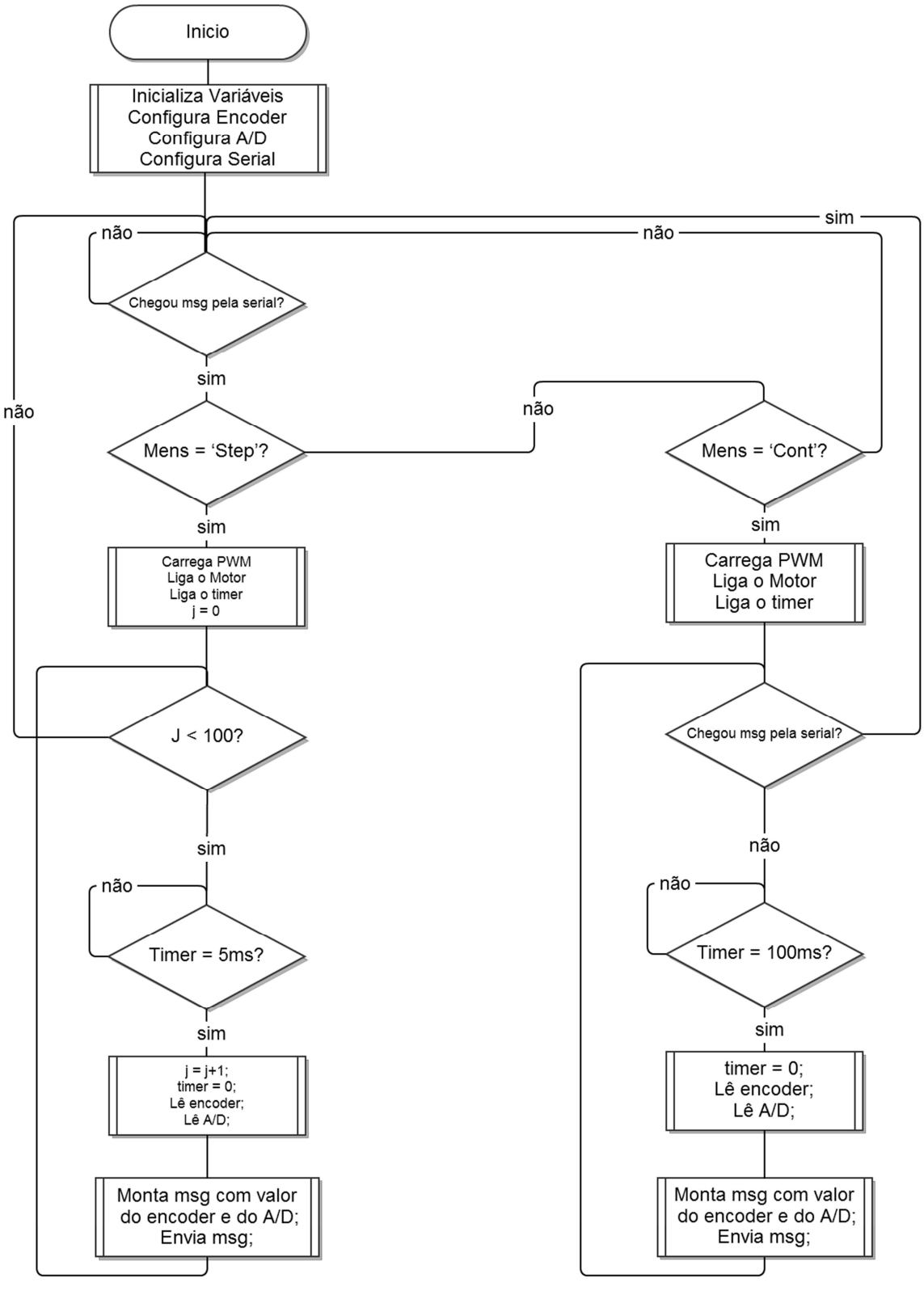


Figura 14 - Fluxograma do programa do microcontrolador

3.6. Interface Homem-Máquina (IHM)

Com o intuito de facilitar todo o processo de ação de comando e consecutiva aquisição de dados pela plataforma de aquisição e controle, foi criada uma interface para que o usuário possa realizar algumas ações de forma rápida e eficiente.

Desenvolvimento da IHM

O MATLAB possui uma ferramenta muito prática para o desenvolvimento de interfaces gráficas. Com o utilitário GUI "Graphical User Interface" é possível montar rapidamente uma interface, selecionando e arrastando as componentes desejadas, como botões, caixa de texto, campos de seleção e até mesmo gráficos. Este utilitário pode ser acessado através do comando *GUIDE* na janela de comandos do MATLAB. Na Figura 15 está a interface montada para este projeto. Os seguintes componentes foram utilizados para compor a interface: *Push Button, ListBox, Edit Text, Slider, Table e Axes*.

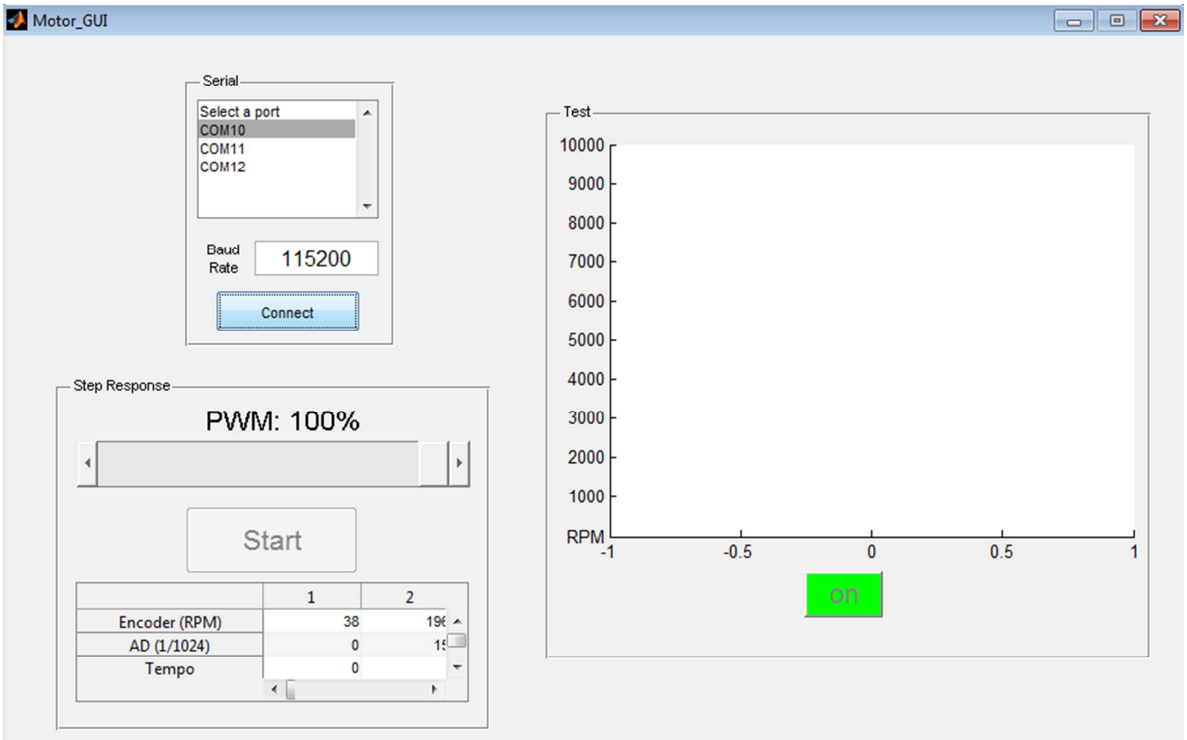


Figura 15 - Interface Gráfica

Funcionamento da IHM

Tendo a interface montada com os componentes necessários, quando o usuário clicar em *Run Figure* na *Toolbar* do GUIDE, será gerado o código em linguagem do MATLAB da

interface gráfica montada. Este código é composto por uma parte que o usuário não deve editar, referente à construção da interface, e outra parte onde se encontram todos os *Callbacks* dos componentes da interface.

Callback é uma função que é chamada/executada quando o usuário executa alguma ação na interface gráfica, desta forma, quando um botão é pressionado o *Callback* referente a este botão é chamado e então é executada a ação programada para este botão.

O botão *Connect*

A primeira ação que o usuário deve executar para colocar o sistema em funcionamento é selecionar em qual porta serial está conectada a plataforma de aquisição e controle através da *ListBox* que lista todas as portas seriais disponíveis no microcomputador, deve também entrar com o valor do *Baud Rate* da comunicação serial e por fim clicar no botão *Connect* para que a porta serial seja aberta ficando disponível para troca de dados.

Step Response

Estando o sistema conectado, os botões Start e On/Off ficam liberados para seleção. O botão Start aciona a ação chamada aqui de *Step*, ação referente à *"Mens=Step"* no fluxograma da Figura 14. Esta ação roda o programa que pode ser visto a seguir:

```
PWM=get(handles.slider1, 'Value');

fwrite(handles.serConn, 's');
fwrite(handles.serConn, PWM, 'uint8');
fwrite(handles.serConn, 'xx');

data1=fscanf(handles.serConn,'%u', [3 100]);

encoder=data1(1,:);
AD=data1(2,:);
time=(0.005)*data1(3,:);%time em 5ms
RPM_AD=(AD*2*(5/1023)*1340);%2224

..Continua
```

A primeira linha do código realiza a leitura do *Slider* referente ao valor do PWM escolhido pelo usuário na interface gráfica. As três linhas seguintes possuem a função *fwrite( )*, função esta que escreve na porta serial, neste caso ela inicia escrevendo a letra S seguida do valor do PWM e então duas letras X. A letra S é indicativa de função *Step*, e as letras XX apenas preenchem um espaço vazio no pacote da mensagem. Este espaço vazio foi deixado para que o software fique flexível, possibilitando futuras modificações.

A função *fscanf( )* é responsável por ler o conteúdo do *buffer* da serial do microcomputador, um de seus parâmetros é [3 100] indicando que deve ser lido do *buffer* uma matriz de 3 colunas e 100 linhas. As linhas seguintes do código mostram como esta matriz 3x100 foi separada. O valor da velocidade em RPM proveniente do encoder encontra-se na primeira coluna. O valor da F.E.M proveniente do conversor A/D está na segunda coluna da matriz.

O valor transmitido pelo microcontrolador e lido pela IHM para o valor da F.E.M está em seu estado não "tratado", ou seja, este é um valor entre 0 e 1024, escala referente as tensões de referência 0 e 5 Volts do conversor A/D. Esta variável ainda deve ser multiplicada por 2 já que existe uma divisão por 2 no valor da F.E.M, e então multiplicada por K_n (constante de velocidade do motor), que no exemplo acima se usou o valor 1340, assim o resultado desta operação é o valor da velocidade em RPM proveniente da F.E.M. Estas operações podem ser colocadas no processamento do microcontrolador , mas foi escolhido fazê-las no MATLAB para que o usuário tenha a possibilidades de adaptá-las de maneira rápida.

As linhas seguintes deste código podem ser consultadas no **Apêndice C** onde se encontra todo o código da IHM. As linhas seguintes são responsáveis pela geração dos três gráficos propostos, um da relação velocidade por tensão gerada, outro da tensão gerada pelo tempo, e o ultimo gráfico contendo a velocidade mensurada pelo encoder e a velocidade mensurada através da F.E.M pelo tempo. O comando utilizado para gerar estes gráficos é o comando *plot( )* com algumas funções auxiliares para adicionar legendas e cores aos gráficos. Neste ponto do programa o usuário tem total liberdade de alterar os tipo, os dados, e a maneira como estes gráficos serão gerados de acordo com suas necessidades.

Test *Continuos*

O botão *On* aciona a ação chamada aqui de *Continuos*, ação referente à *"Mens=Cont"* no fluxograma da Figura 14. Este Callback funciona de maneira idêntica ao *Step* com a diferença de que os dados são lidos da serial em tempo real e o gráfico gerado é atualizado também em tempo real. Esta função é executada até que o usuário aperte a tecla On/Off novamente.

4. Resultados

Realizando o teste *Step* para o motor **2224-006SR** da Faulhaber que já possui *encoder* acoplado ao seu eixo. Encontramos o gráfico da relação velocidade por F.E.M, Figura 16. Pela equação resultante para a reta aproximada temos que o coeficiente angular da reta, que também corresponde à constante de velocidade do motor (K_n) é igual a *1374 rpm/V,* valor próximo ao esperado para este motor que é de *1380 rpm/V* conforme folha de dados do Anexo A.

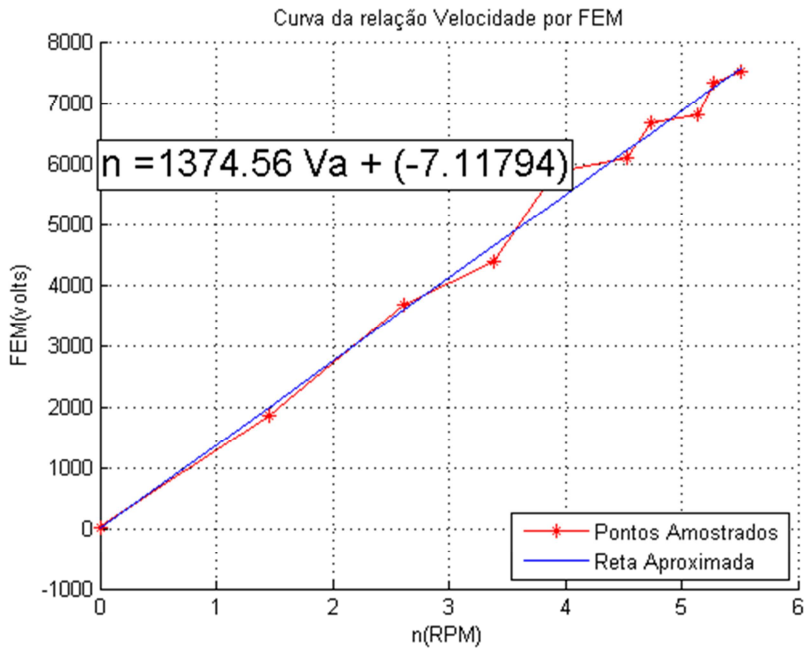


Figura 16 - Curva da Relação Velocidade por F.E.M

O segundo gráfico produzido pelo sistema é a curva da F.E.M pelo tempo, Figura 17. Como pode ser observada, a tensão máxima desta curva é de 6 Volts correspondendo à velocidade máxima do motor 2224 quando alimentado com uma tensão de 6 Volts. O ruído sobreposto ao sinal CC devido ao comutador para este motor não é muito significativo, assim não foi adicionado nenhum tipo de filtro para este sinal.

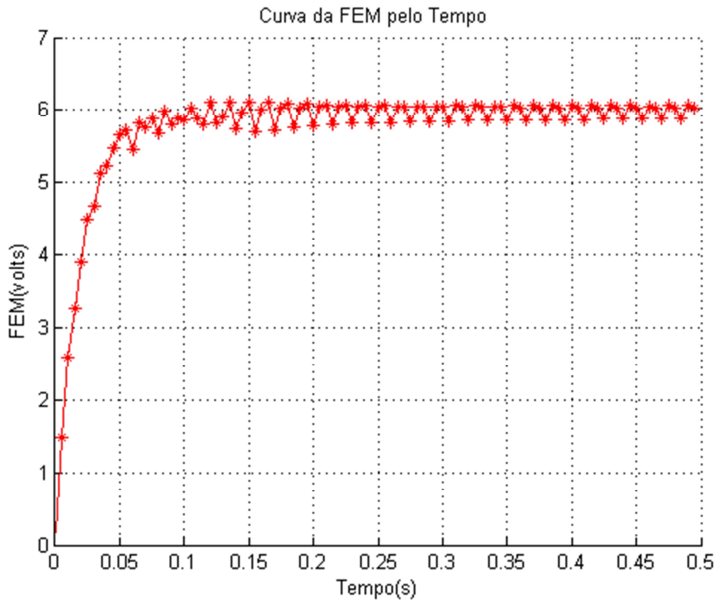


Figura 17 - Curva da F.E.M pelo Tempo

O terceiro gráfico produzido pelo sistema é a curva característica do motor contendo dois traços, um referente à velocidade em RPM lida por meio do encoder e o outro referente à velocidade em RPM lida por meio da F.E.M, (Figura 18). Como pode ser observada no gráfico, a velocidade lida pela F.E.M é muito próxima da velocidade lida pelo encoder indicando que o sistema está operando satisfatoriamente. Não foi calculado o erro médio para este gráfico já que os dois sinais são ruidosos, não existindo, portanto, um valor exato de referência para a velocidade do motor.

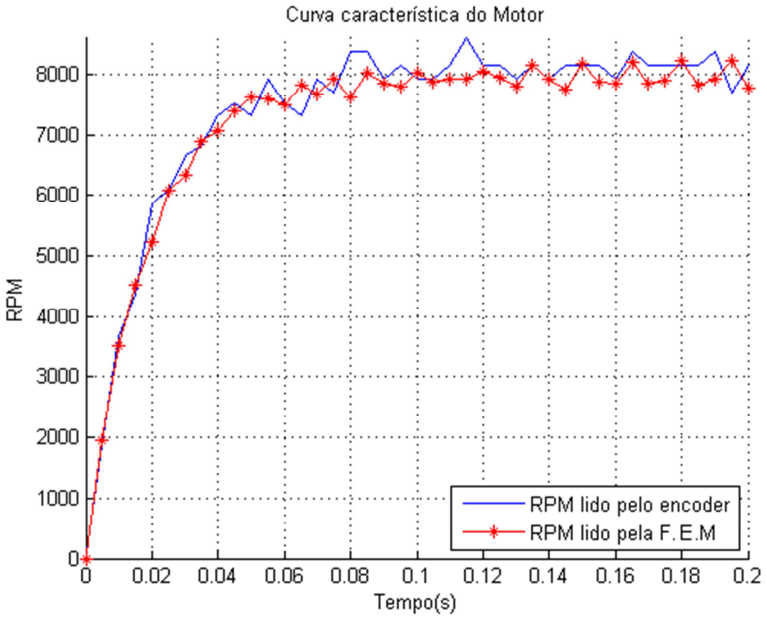


Figura 18 - Curva Característica do Motor

Realizando o teste *Continuos* para o motor 2224-006SR obteve-se a Figura 19. Como pode ser observado no gráfico da IHM ocorre uma variação no sinal, isto porque foi adicionado um torque variado ao motor para que pudesse ser visto o comportamento da velocidade. O gráfico possui três traços distintos, o de cor azul refere-se à velocidade em RPM mensurada pelo encoder, o de cor vermelha à velocidade mensurada pela F.E.M e o de cor verde é referente a velocidade mensurada pela F.E.M através de um filtro de média móvel.

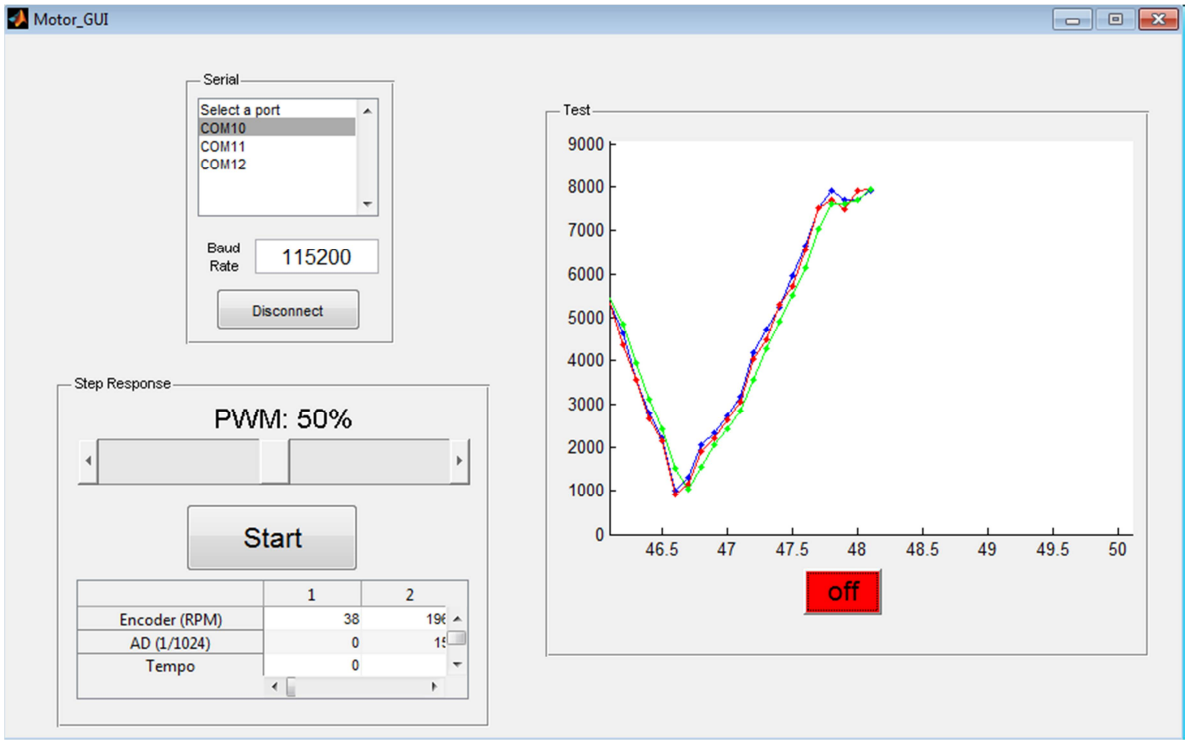


Figura 19 - IHM no teste "Continuos"

Os testes foram realizados também com o motor **PM080-AU-S** da Action, Anexo B, que é um motor CC de baixo custo de fabricação brasileira. Para possibilitar a leitura da velocidade real do motor fez-se a montagem da Figura 20. Acoplando o eixo do motor PM080 ao eixo do motor 2224 foi possível utilizar o encoder do motor 2224.

Figura 20 - Acoplamento dos motores

Na Figura 21 pode ser observado o gráfico da relação velocidade por F.E.M. O valor encontrado para a constante de velocidade do motor (K_n) é igual a 2058 RPM/V, porém com os testes realizados foi constatado que este valor não corresponde ao valor correto e que observando o gráfico da Figura 23 percebe-se que o valor da F.E.M é muito ruidoso para este motor. Para diminuir a influência do ruído adicionado pelo comutador foi projetado um filtro de média móvel para este sinal. Obteve-se o melhor resultado para o filtro de média móvel utilizando a média das três ultimas amostras do sinal, com isso o valor encontrado para a constante K_n foi de 2160 RPM/V, Figura 22.

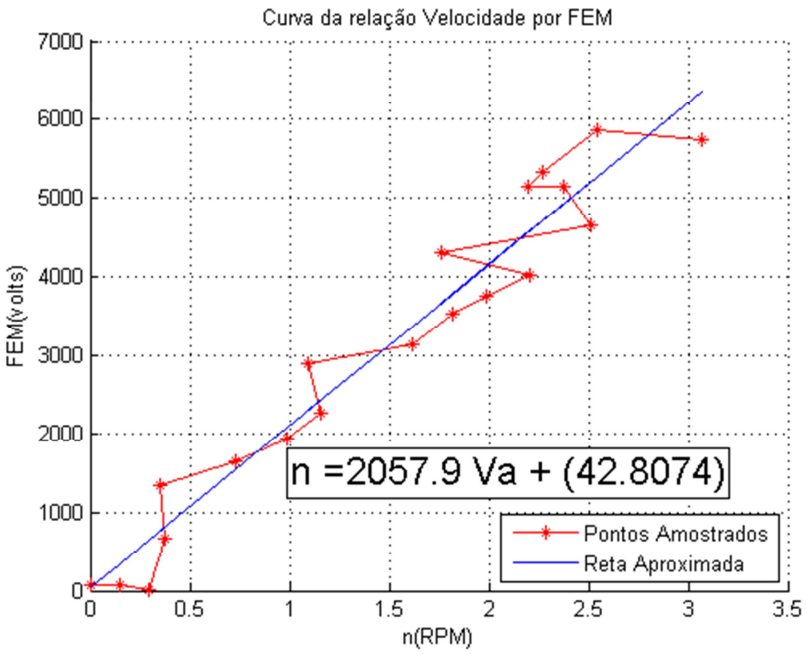


Figura 21 - Curva da Relação Velocidade por F.E.M para o motor da Action

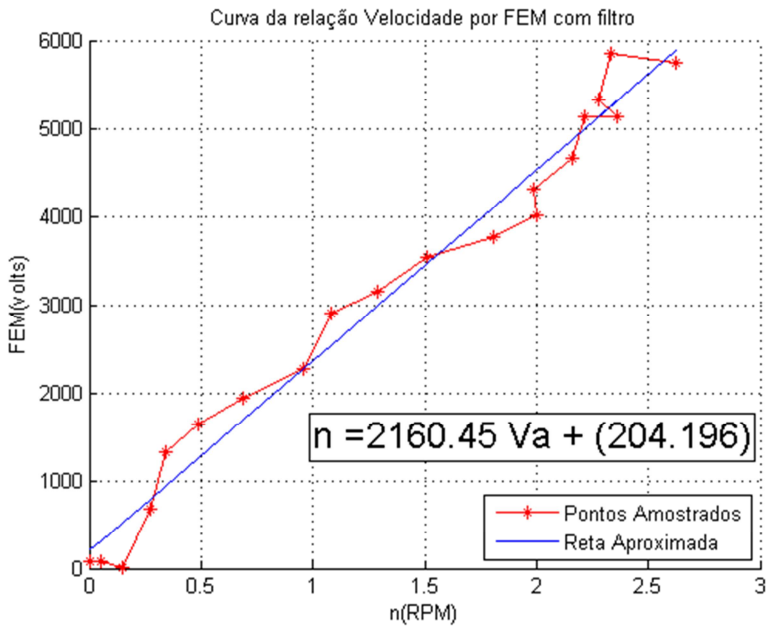


Figura 22 - Curva da Relação Velocidade por FEM para o motor Action com filtro

Utilizando $K_n = 2160\ rpm/V$ obteve-se o gráfico da Figura 24 para a curva característica do motor PM080-AU-S da Action, no gráfico podem ser observados três diferentes traços, velocidade mensurada pelo encoder, mensurada pela F.E.M e a velocidade mensurada pela F.E.M com o filtro de média móvel das três ultimas amostras. O valor nominal de K_n para este motor não é disponibilizado pelo fabricante, desta maneira seu valor foi determinado por meio de testes.

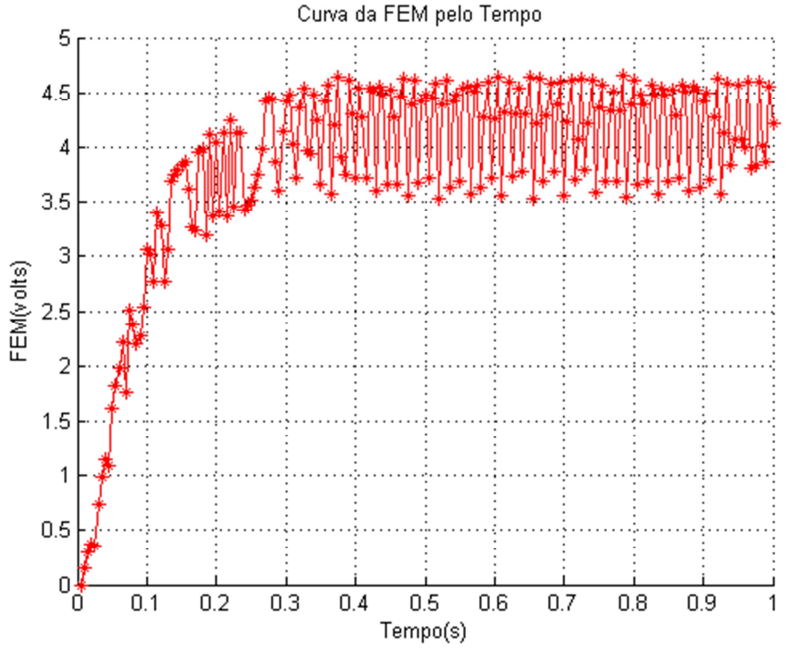


Figura 23 - Curva da F.E.M pelo Tempo para o motor da Action

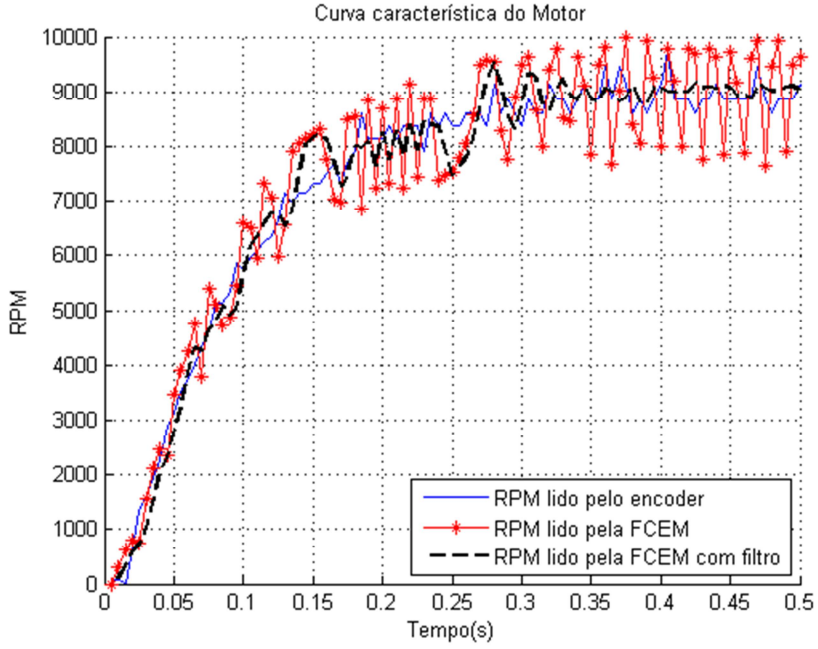


Figura 24 - Curva característica para o motor da Action

5. Conclusões

Este projeto possibilitou a integração de várias tecnologias e conceitos adquiridos durante o curso de Engenharia Elétrica da Escola de Engenharia de São Carlos, como teorias nas áreas de Microcontroladores, Programação C, Controle, Instrumentação, Eletrônica de Potência, entre outros.

O objetivo inicial deste projeto era apenas estudar e implementar uma metodologia de leitura de velocidade em motores CC, porém o trabalho foi além do estudo da metodologia, foi desenvolvido todo um sistema *Hardware – Software* – IHM de baixo custo que possibilita futuros estudos nas áreas de controle, filtros, aquisição de sinais, e controles gerais por meio de interface com o computador. Todas as técnicas aqui utilizadas fazem uso de componentes e programas comuns a estudantes de Engenharia Elétrica, possibilitando fácil entendimento e fácil acesso aos materiais necessários para construção deste Sistema. Um exemplo é o uso do MATLAB para a implementação da IHM, *software* este muito estudado e utilizado por estudantes de Engenharia Elétrica.

O Sistema mostrou-se satisfatório dentro do esperado; Pode-se concluir que para motores CC de melhor qualidade, ou seja, com perdas resistivas internas menores, com um número maior de polos, e um comutador de maior qualidade, que a adição de ruído é consideravelmente menor comparado com motores CC de baixa qualidade. Porém, com a introdução do filtro, verificou-se que mesmo para motores de menor qualidade é possível obter um bom sinal para leitura da velocidade para aplicações onde não é exigida grande precisão.

As limitações do sistema estão relacionadas com a precisão de leitura dos sinais e também com a frequência de amostragem. A primeira limitação encontrada está no fato do encoder utilizado possuir apenas 64 pulsos por volta de resolução, limitando a taxa de amostragem da velocidade. A comunicação serial utilizada também adiciona limitações ao sistema, sua taxa de transferência é baixa quando comparada a sistemas que utilizam a porta USB para comunicação, desta forma a frequência de amostragem dos sinais fica limitada à velocidade com que essas amostragens podem ser transmitidas à IHM. A terceira limitação está no microcontrolador utilizado que possui frequência máxima de operação de 40MHz.

Para a continuação deste projeto são sugeridas implementações de algumas ferramentas junto à IHM no MATLAB, como: adicionar ao teste *Step* (resposta ao degrau) a automatização do processo de obtenção dos parâmetros do motor através da curva característica, encontrando a função transferência do motor CC; Implementar um sistema de sintonia automática para controladores do tipo PID para motores CC de maneira interativa utilizando a IHM e as ferramentas do MATLAB.

Referências Bibliográficas

[1] E. V. Sánchez e J. G. Gil, "Análisis de tres métodos para la detección de la velocidad de giro de un motor DC, sin el empleo de sensores externos," *REE,* pp. 72-79, 2009.

[2] SIEMENS, MOTORES DE CORRENTE CONTÍNUA/Guia rápido para uma especificação precisa, 2006.

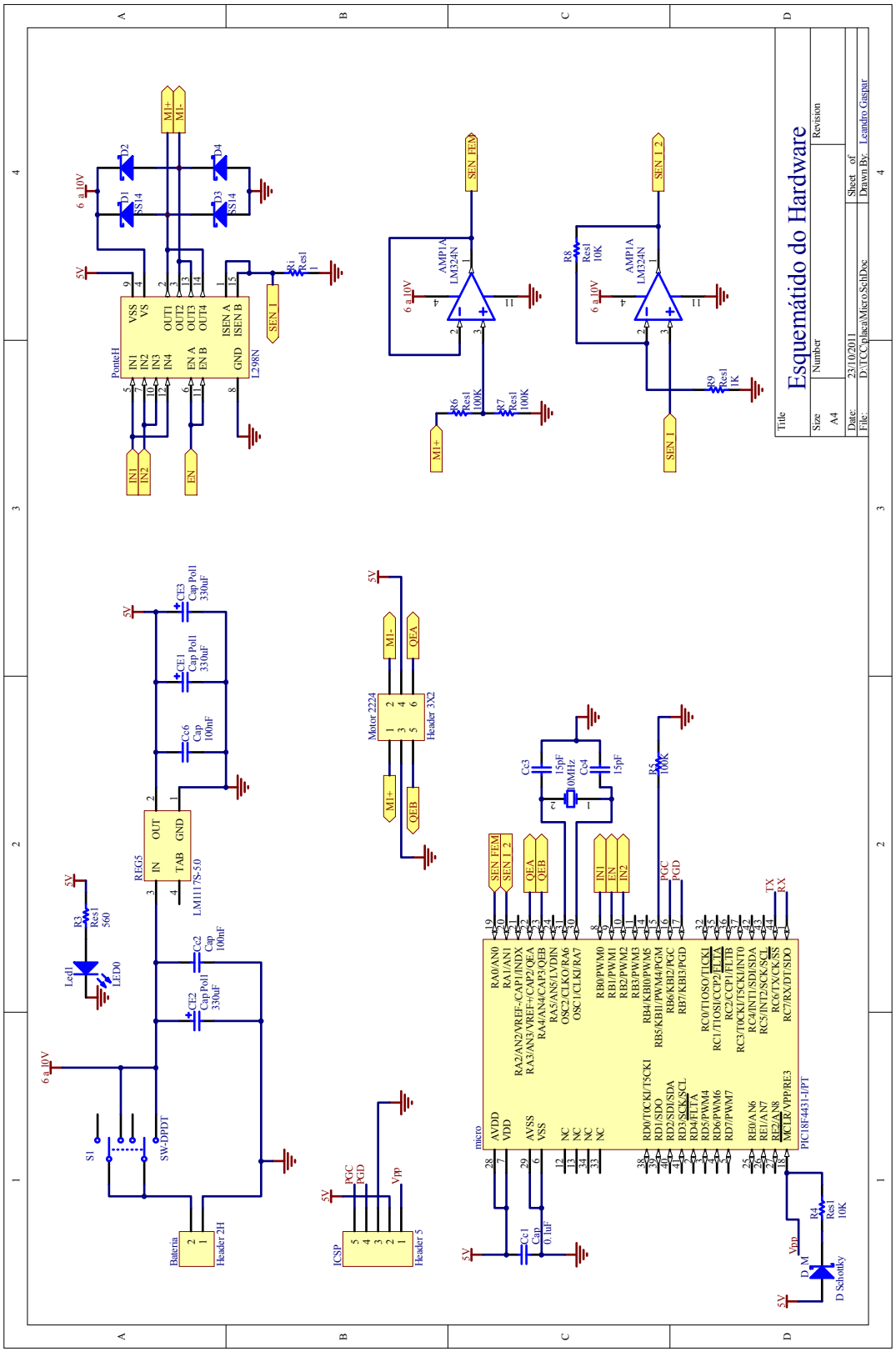
[3] "How stuff works," [Online]. Available: http://www.howstuffworks.com. [Acesso em 01 11 2011].

[4] V. A. Oliveira, M. L. Aguiar e J. B. Vargas, Sistemas de Controle, São Carlos: EESC-USP, 2005.

[5] "http://www.microchip.com/," [Online].

[6] Microchip Technology Inc., *MPLAB® C18 C COMPILER GETTING STARTED,* 2005.

Apêndice A –Eletrônica da Plataforma de Aquisição e Controle



Apêndice B – Programa do Microcontrolador

```
include <p18f4431.h>
#include <adc.h>
#include <delays.h>
#include <timers.h>
#include <stdio.h>
#include <stdlib.h>

#pragma config OSC = HSPLL
#pragma config WDTEN = OFF
#pragma config PWRTEN = ON

//Macros
#define     IN1                     PORTBbits.RB0
#define     IN2                     PORTBbits.RB2
#define     IN3                     PORTBbits.RB4
#define     IN4                     PORTBbits.RB5
#define     AMARELO PORTAbits.RA5
#define VERDE           PORTEbits.RE0
#define     PWM_ON_OFF          PTCON1bits.PTEN

struct sent_motor
{
            unsigned M1:1;
            unsigned M2:1;
            unsigned dados_ready:1;
            unsigned M1_zero:1;
            unsigned M2_zero:1;
};

typedef union
{
            struct sent_motor bits;
            unsigned char data;
}sent_motor;

volatile sent_motor sent_motorbits;
unsigned char data[4];
int i=0;

void high_isr (void);
void inicializacao (void);
void config_encoder (void);
void config_analog (void);
unsigned int get_encoder(void);
void print(unsigned int x1, unsigned int x2, unsigned int y);

////////Interrupçoes//////
#pragma   code high_vector=0x08
void interrupt_at_high_vector (void)
{
            _asm GOTO high_isr  _endasm
}
#pragma   code

///////Sub-rotinas de interrupcao//////
#pragma   interrupt high_isr
void high_isr (void)
{
//INTCONbits.GIE=0;//turn off the interrupt
INTCONbits.PEIE=0;

            if(PIR3bits.IC2QEIF==1)//Encoder1
            {
                        PIR3bits.IC2QEIF=0;
                        sent_motorbits.bits.M1=PORTCbits.RC1;
                        sent_motorbits.bits.M1_zero=0;
            }

            if(PIR1bits.CCP1IF==1)//Encoder
```

```
		{
			PIR1bits.CCP1IF=0;
			sent_motorbits.bits.M2=PORTAbits.RA4;
			TMR1H=0;//A escrita no Low desencadeia a escrita nos dois registradores, portanto o low deve ser escrito depois.
			TMR1L=0x05;//correcao do tempo de atendimento da interrupcao
		}

		if(PIR1bits.RCIF==1)//Serial
		{
			data[i]=RCREG;
			if(i==3)sent_motorbits.bits.dados_ready=1;
			i++;
		}

		if(INTCONbits.TMR0IF==1);//Timer0
		{
			INTCONbits.TMR0IF=0;
			sent_motorbits.bits.M1_zero=1;
		}

//INTCONbits.GIE=1;
INTCONbits.PEIE=1;

}
//-----------------------------------------------------//
void main (void)
{
unsigned int encoder_vel_0=0;
unsigned int AD3=0,AD2=0,AD1=0,AD0=0,time=0,PWM=0,j=0;
unsigned int stop=0;
float M;

AMARELO=0;

inicializacao();
config_encoder();
config_analog();

INTCONbits.GIE=1; //Global Interrupt Enable
INTCONbits.PEIE=1; //Peripheral Interrupt Enable

//------------------------loop-------------------------//

while(1)
{
if(sent_motorbits.bits.dados_ready)
		{
					i=0;
					sent_motorbits.bits.dados_ready=0;
					if(data[0]=='s')//Step
					{

						PWM=data[1];
						PWM=PWM*20; //para PWM em %
						PDC0L=PWM; //de 0 a 2000
						PDC0H=PWM/0x100;

						IN1=1;
						IN2=0;
						PWM_ON_OFF=1;

						WriteTimer0(65341);// (5ms)/25.6us = 195.3 => 65536-195 = 65341
						sent_motorbits.bits.M1_zero=1;

						for(j=0; j<100; j++)
						{
						while(sent_motorbits.bits.M1_zero==0);
						WriteTimer0(65341);//5ms
						sent_motorbits.bits.M1_zero=0;
```

```
		//////Encoder/////
		encoder_vel_0=get_encoder();

		///// A/D ////
		PWM_ON_OFF=0;
		Delay100TCYx(30);//Para o 2224!
		//Delay100TCYx(100);//(x=> x*10us)Para o Action!
		ADCON0bits.GO_DONE=1; //inicia aquisicao
		while(ADCON0bits.GO_DONE);
		PWM_ON_OFF=1;
		AD0 = ADRESH;
		AD0 <<= 8;
		AD0 |= ADRESL;

		/////Imprime/////
		print(encoder_vel_0, AD0, time);
		time+=1;
		}//for

printf("\n");
PWM_ON_OFF=0;
time=0;
}

if(data[0]=='c')//continuos
{
		PWM=data[1];
		PWM=PWM*20; //para PWM em %
		PDC0L=PWM; //de 0 a 2000
		PDC0H=PWM/0x100;

		IN1=1;
		IN2=0;
		PWM_ON_OFF=1;

		WriteTimer0(61630);// 100ms/25.6us = 3906 => 65536-3906=61630
		sent_motorbits.bits.M1_zero=1;

		while(stop==0)
		{
		if(sent_motorbits.bits.dados_ready)
			{
			i=0;
			sent_motorbits.bits.dados_ready=0;
			stop=1;
			}

		while(sent_motorbits.bits.M1_zero==0);//IF do timer
		WriteTimer0(61630);
		sent_motorbits.bits.M1_zero=0;

		//////Encoder/////
		encoder_vel_0=get_encoder();

		///// A/D ////
		PWM_ON_OFF=0;
		Delay100TCYx(30);//Para o 2224!
		//Delay100TCYx(100);//(x=> x*10us)Para o Action!
		ADCON0bits.GO_DONE=1; //inicia aquisicao
		while(ADCON0bits.GO_DONE);
		PWM_ON_OFF=1;
		AD0 = ADRESH;
		AD0 <<= 8;
		AD0 |= ADRESL;

		/////Imprime/////
		print(encoder_vel_0, AD0, time);
		printf("\n");
		time+=1;
		}//for
```

```
                                PWM_ON_OFF=0;
                                time=0;
                                stop=0;
                                }

        }//if(dados.ready)
}//while
}//main

void inicializacao (void)
{
//Porta A
        ANSEL0 = 0b00000001; //[1] - analog
        ANSEL1 = 0;
        TRISA = 0b00011101;
        PORTA = 0;
//Porta B
        TRISB=0;
        PORTB=0;
//Porta C
        TRISC=0b10000000;
        PORTC=0;
//Porta D
        TRISD=0b00000000;
        PORTD=0;
//Porta E
        TRISE=0b00000000;
        PORTE=0;
//Serial
        BAUDCTLbits.BRG16=1;
        SPBRG=0x56;             //Baudrate para 40mhz em 16bits
        SPBRGH=0x00;                                    //115200 0x56
        TXSTA=0x24;                                             //57600    0xAC
        RCSTA=0b10010000;                       //19200    0x208
        PIE1bits.RCIE=1;                        //38400 0x0103
        TRISCbits.TRISC6=0;
        TRISCbits.TRISC7=1;
//Time
        OpenTimer0( TIMER_INT_ON &
        T0_16BIT &
        T0_SOURCE_INT &
        T0_PS_1_256 );
//PWM
        PTCON0=0b00000000; //[7-4]postscale,[3-2]prescale,[1-0]Mode
        PTPERL=0xFF;
        PTPERH=0x01;
        PWMCON0=0b00011111; //[7]0, [6-4]ports io, [3-0]output pair //PWM1 e PWM3 output
        PDC0L=0; //inicializa todos os registradores de PWM
        PDC0H=0;
        PDC1L=0;
        PDC1H=0;
        PTCON1=0b10000000;//[7]pwm on, [6]count direction. [5-0]not used
}

void config_encoder (void)
{
        QEICON=0b00010100; //4x and 1:1
        CAP1CON=0b01000010; //every rise start the counter
        T5CON=0b11011101; //Encoder 64 - prescale(1:8)
}

void config_analog (void)
{
        ADCON0=0b00000001; //[7-6]0,[5]Cont ou single conversion,[6]
        ADCON1=0b00000000;
        ADCON2=0b10111010;
        ADCON3=0b00000000;
        ADCHS= 0b00000000; //AN0-A
}

unsigned int get_encoder(void)
```

```
{
        unsigned int encoder;
        encoder = VELRH;
        encoder <<= 8;
        encoder |= VELRL;
        encoder= 292968.75/encoder;
        return encoder;
}

void print(unsigned int x1, unsigned int x2, unsigned int y)
{
        printf("%u ", x1);//u% for unsigned
        printf("%u ", x2);
        printf("%u", y);
        printf("\r");
}
```

Apêndice C – Programa da IHM

```
function varargout = Motor_GUI(varargin)
%   Author: Leandro S. Gaspar
%   Version: 1.0  |  Date: 17.10.2011

% Motor_GUI M-file for Motor_GUI.fig

% Last Modified by GUIDE v2.5 17-Oct-2011 12:08:55

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Motor_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Motor_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Motor_GUI is made visible.
function Motor_GUI_OpeningFcn(hObject, eventdata, handles, varargin)

serialPorts = instrhwinfo('serial');
nPorts = length(serialPorts.SerialPorts);
set(handles.portList, 'String', ...
    [{'Select a port'} ; serialPorts.SerialPorts ]);
set(handles.portList, 'Value', 2);

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Motor_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Motor_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in portList.
function portList_Callback(hObject, eventdata, handles)
% hObject    handle to portList (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns portList contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from portList


% --- Executes during object creation, after setting all properties.
function portList_CreateFcn(hObject, eventdata, handles)
% hObject    handle to portList (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function history_box_Callback(hObject, eventdata, handles)
% hObject    handle to history_box (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of history_box as text
%        str2double(get(hObject,'String')) returns contents of history_box as a double


function baudRateText_Callback(hObject, eventdata, handles)
% hObject    handle to baudRateText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of baudRateText as text
%        str2double(get(hObject,'String')) returns contents of baudRateText as a double


% --- Executes during object creation, after setting all properties.
function baudRateText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to baudRateText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in connectButton.
function connectButton_Callback(hObject, eventdata, handles)
if strcmp(get(hObject,'String'),'Connect') % currently disconnected
    serPortn = get(handles.portList, 'Value');
    if serPortn == 1
        errordlg('Select valid COM port');
    else
        serList = get(handles.portList,'String');
        serPort = serList{serPortn};
        serConn = serial(serPort, 'TimeOut', 1, ...
            'BaudRate', str2num(get(handles.baudRateText, 'String')),...
            'InputBufferSize', 32768);

        try
            fopen(serConn);
            handles.serConn = serConn;

            % enable Tx text field and Rx button
            set(handles.Start, 'Enable', 'On');
            set(handles.on, 'Enable', 'On');

            set(hObject, 'String','Disconnect')
        catch e
            errordlg(e.message);
        end

    end
else
    set(handles.Start, 'Enable', 'Off');
    set(handles.on, 'Enable', 'Off');

    set(hObject, 'String','Connect')
    fclose(handles.serConn);
end
guidata(hObject, handles);
```

```
% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if isfield(handles, 'serConn')
    fclose(handles.serConn);
end
% Hint: delete(hObject) closes the figure
delete(hObject);


% --- Executes on button press in Start.
function Start_Callback(hObject, eventdata, handles)
% hObject    handle to Start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%fprintf(handles.serConn, 'vaaa');

PWM=get(handles.slider1, 'Value');

fwrite(handles.serConn, 's');
fwrite(handles.serConn, PWM, 'uint8');
fwrite(handles.serConn, 'xx');

data1=fscanf(handles.serConn,'%u', [3 100]);

encoder=data1(1,:);
AD=data1(2,:);
time=(0.005)*data1(3,:);%time em 5ms
RPM_AD=(AD*2*(5/1023)*1340);%2224

%%
figure(1);
hold on;
plot(time,encoder,'b-');
plot(time,RPM_AD,'r-*');
%plot(time,filter(ones(1,3)/3, 1, RPM_AD),'g-o');
title('Curva característica do Motor');
axis([0 0.2 0 Inf])
xlabel('Tempo(s)');
ylabel('RPM');
legend('RPM lido pelo encoder','RPM lido pela FCEM','RPM lido pela FCEM com filtro',4);
grid
hold off
%%
figure(2)
hold on
plot(time,AD*2*(5/1023),'r-*');
plot(time,AD*2*(5/1023),'r-*');
title('Curva da FEM pelo Tempo ');
xlabel('Tempo(s)');
ylabel('FEM(volts)');
grid
hold off
%%
figure(3)
hold on

e=AD*2*(5/1023);

p = polyfit(e(1:10),encoder(1:10),1);%Polinomio da reta (y = a1 x + a0)
RPMa = polyval(p,e);%Valores numéricos da curva de ajuste
plot(e(1:10),encoder(1:10),'r-*');
plot(e(1:10),RPMa(1:10),'b-');
text(0,6000,['n =',num2str(p(1),6),' Va +
(',num2str(p(2),6),')'],'FontSize',18,'BackgroundColor',[1 1 1],...
             'EdgeColor','black');
title('Curva da relação Velocidade por FEM');
xlabel('n(RPM)');
ylabel('FEM(volts)');
legend('Pontos Amostrados','Reta Aproximada',4);
grid
hold off
%%
set(handles.tabela,'data',data1);
```

```
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
valor=get(handles.slider1, 'Value');
str=['PWM: ',num2str(valor),'%'];
set(handles.text4,'String', str);


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in on.
function on_Callback(hObject, eventdata, handles)
% hObject    handle to on (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if strcmp(get(hObject,'String'),'on') %ligar motor

   PWM=get(handles.slider1, 'Value');
   fwrite(handles.serConn, 'c');
   fwrite(handles.serConn, PWM, 'uint8');
   fwrite(handles.serConn, 'xx');

   RPM1 = 0;
   RPM2 = 0;
   RPM3 = 0;
   time = 0;
   count = 1;

   RPM3_1=0;
   RPM3_0=0;

 hold on;

 plotHandle1 = plot(handles.axes1,time,RPM1,'Marker','.','LineWidth',1,'Color','blue');
 plotHandle2 = plot(handles.axes1,time,RPM2,'Marker','.','LineWidth',1,'Color','red');
 plotHandle3 = plot(handles.axes1,time,RPM3,'Marker','.','LineWidth',1,'Color','green');

 xlim(handles.axes1,[min(time) max(time+1)]);
 ylim(handles.axes1,[0 10000]);

set(handles.on,'UserData',1);
set(hObject, 'String','off');
set(hObject, 'BackgroundColor','red');

else
    strcmp(get(hObject,'String'),'off');
    fwrite(handles.serConn, 'xxxx');% deliga o motor
    set(handles.on,'UserData',0);% muda a flag
    set(hObject, 'String','on');% muda o nome do pushbutton
    set(hObject, 'BackgroundColor','green');
end

 while (get(handles.on,'UserData') ==1)% testa a flag

    data2=fscanf(handles.serConn,'%u', [3 1]);
    encoder=data2(1,:);
    AD=data2(2,:);
    t=(0.1)* data2(3,:);
    RPM_AD =(AD*2*(5/1023)*1340);%2224
```

```
    time(count) = t;
    RPM1(count) = encoder;
    RPM2(count) = RPM_AD;
    %RPM3(count) = 3*(filter(ones(1,3)/3, 1, RPM_AD));

    RPM3_0=RPM_AD;
    RPM3(count)=(RPM3_1+RPM3_0)/2;
    RPM3_1=RPM3_0;

    set(plotHandle1,'YData',RPM1,'XData',time);
    set(plotHandle2,'YData',RPM2,'XData',time);
    set(plotHandle3,'YData',RPM3,'XData',time);

    xlim(handles.axes1,[max(time-2) max(time+2)]);
    ylim(handles.axes1,[0 max(RPM2+1000)]);

    pause(0.01);
    count = count +1;
    drawnow
 end

    if(handles.serConn.BytesAvailable > 0)
        fread(handles.serConn,handles.serConn.BytesAvailable);% clear the InputBuffer-
Size
    end

    cla(handles.axes1,'reset');
    guidata(hObject, handles);
```

ANEXO A – Folha de dados do Motor 2224-006SR

FAULHABER

DC-Motors

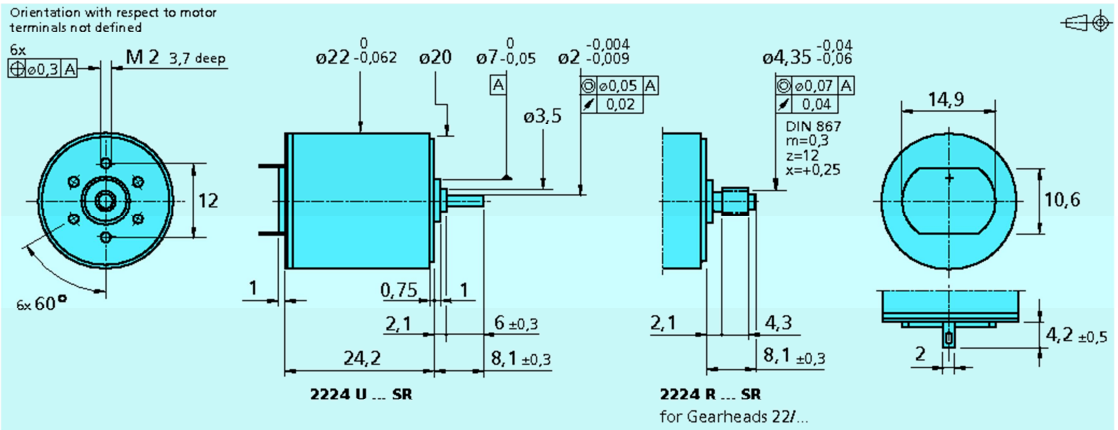
5 mNm

Precious Metal Commutation

For combination with
Gearheads: 20/1, 22E, 22/2, 22/5, 22/6, 23/1, 38/3
Encoders: IE2 – 16 ... 512, HE2-16

Series 2224 ... SR

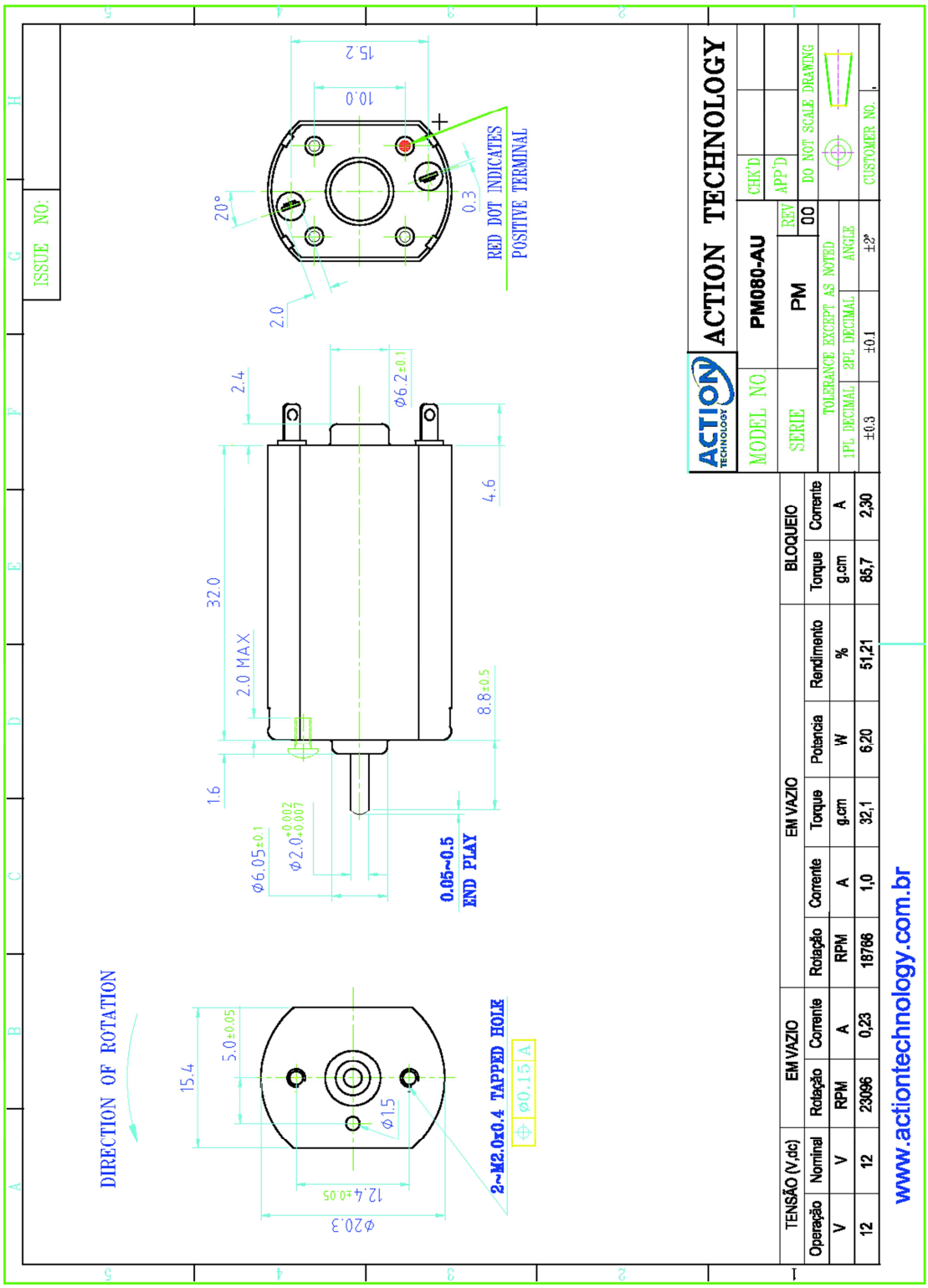
		2224 U	003 SR	006 SR	012 SR	018 SR	024 SR	036 SR	
1	Nominal voltage	U_N	3	6	12	18	24	36	Volt
2	Terminal resistance	R	0,56	1,94	8,71	17,50	36,30	91,40	Ω
3	Output power	$P_{2\ max.}$	3,92	4,55	4,05	4,54	3,88	3,46	W
4	Efficiency	$\eta_{max.}$	80	82	82	82	81	80	%
5	No-load speed	n_o	8 100	8 200	7 800	8 100	7 800	7 800	rpm
6	No-load current (with shaft ø 2,0 mm)	I_o	0,066	0,029	0,014	0,010	0,007	0,005	A
7	Stall torque	M_H	18,5	21,2	19,8	21,4	19,0	16,9	mNm
8	Friction torque	M_R	0,23	0,2	0,2	0,21	0,2	0,22	mNm
9	Speed constant	k_n	2 730	1 380	657	454	328	219	rpm/V
10	Back-EMF constant	k_E	0,366	0,725	1,520	2,200	3,040	4,560	mV/rpm
11	Torque constant	k_M	3,49	6,92	14,50	21,00	29,10	43,50	mNm/A
12	Current constant	k_I	0,286	0,144	0,069	0,048	0,034	0,023	A/mNm
13	Slope of n-M curve	Δn/ΔM	438	387	394	379	411	462	rpm/mNm
14	Rotor inductance	L	11	45	200	450	800	1 800	µH
15	Mechanical time constant	τ_m	11	11	11	11	11	11	ms
16	Rotor inertia	J	2,4	2,7	2,7	2,8	2,6	2,3	gcm²
17	Angular acceleration	$\alpha_{max.}$	77	78	74	77	74	74	$\cdot 10^3$rad/s²
18	Thermal resistance	R_{th1} / R_{th2}	5 / 20						K/W
19	Thermal time constant	τ_{w1} / τ_{w2}	6,8 / 440						s
20	Operating temperature range:								
	– motor		– 30 ... + 85 (optional – 55 ... + 125)						°C
	– rotor, max. permissible		+ 125						°C
21	Shaft bearings		sintered bronze sleeves (standard)		ball bearings (optional)		ball bearings, preloaded (optional)		
22	Shaft load max.:								
	– with shaft diameter		2,0		2,0		2,0		mm
	– radial at 3 000 rpm (3 mm from bearing)		1,5		8		8		N
	– axial at 3 000 rpm		0,2		0,8		0,8		N
	– axial at standstill		20		10		10		N
23	Shaft play:								
	– radial	≤	0,03		0,015		0,015		mm
	– axial	≤	0,2		0,2		0		mm
24	Housing material		steel, black coated						
25	Weight		46						g
26	Direction of rotation		clockwise, viewed from the front face						
Recommended values - mathematically independent of each other									
27	Speed up to	$n_{e\ max.}$	8 000	8 000	8 000	8 000	8 000	8 000	rpm
28	Torque up to	$M_{e\ max.}$	5	5	5	5	5	5	mNm
29	Current up to (thermal limits)	$I_{e\ max.}$	2,200	1,200	0,570	0,400	0,280	0,180	A



Specification subject to change without notice JAN-2008

MicroMo Electronics, Inc. · 14881 Evergreen Ave. · Clearwater FL, 33762-3008 · Phone: (800) 807-9166 Fax: (727) 572-7763 · info@micromo.com · www.micromo.com

ANEXO B – Folha de dados do Motor PM080-AU-S



TENSÃO (V,dc)		EM VAZIO			EM VAZIO					BLOQUEIO		
Operação	Nominal	Rotação	Corrente	Rotação	Corrente	Torque	Potencia	Rendimento	Torque	Corrente
V	V	RPM	A	RPM	A	g.cm	W	%	g.cm	A
12	12	23096	0,23	18786	1,0	32,1	6,20	51,21	85,7	2,30