unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

GABRIEL SEIJI FONTANA

ESTUDO PARA A IMPLEMENTAÇÃO DE UM SISTEMA DE DESBASTE DE COBRE PARA MANUFATURA DE PLACAS DE CIRCUITO IMPRESSO

Guaratinguetá
2011

GABRIEL SEIJI FONTANA

ESTUDO PARA A IMPLEMENTAÇÃO DE UM SISTEMA DE DESBASTE DE COBRE PARA MANUFATURA DE PLACAS DE CIRCUITO IMPRESSO

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Márcio Abud Marcelino

Guaratinguetá
2011

Fontana, Gabriel Seiji
F679s Sistema de desbaste de cobre para manufatura de placas de circuito impresso / Gabriel Seiji Fontana – Guaratinguetá : [s.n], 2011.
102 f : il.
Bibliografia: f. 63

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2011.
Orientador: Prof. Dr. Márcio Abud Marcelino

1. Microcontroladores I. Título

CDU 621.381

unesp

UNIVERSIDADE ESTADUAL PAULISTA
Faculdade de Engenharia do Campus de Guaratinguetá

SISTEMA DE DEASBASTE DE COBRE PARA MANUFATURA DE PLACAS DE CIRCUITO IMPRESSO

GABRIEL SEIJI FONTANA

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE **GRADUADO EM ENGENHARIA ELÉTRICA**

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Prof. Dr. Samuel Euzédice de Lucena
Coordenador

BANCA EXAMINADORA:

Prof. Dr. MÁRCIO ABUD MARCELINO
Orientador/UNESP-FEG

Prof. Dr. TEÓFILO MIGUEL DE SOUZA
UNESP-FEG

Prof. Msc. FERNANDO RIBEIRO FILADELFO
UNESP-FEG

Dezembro de 2011

Reservo a este momento de conquistas a meus amigos, minha família e minha namorada.

AGRADECIMENTOS

Agradeço à minha família, José Carlos, Tereza e Karen, pela ajuda, esforço, união e alegria por todos os anos de minha vida.

À minha namorada, Tyfany, por ter mudado a minha vida em tantos aspectos, me completanto pessoal e profissionalmente.

Agradeço ao pessoal da república Masmorra que vivenciaram a minha formação, trazendo grandes amizades, momentos de descontração e amadurecimento.

E por fim agradeço a Deus por tudo que já me ocorreu na vida.

“As dificuldades são como as montanhas, elas só se aplainam quando avançamos sobre elas.”

Provérbio Japonês

Fontana, G. S. Estudo para implementação de um sistema de desbaste de cobre para manufatura de placas de circuito impresso. 2011. 102 f. Trabalho de Graduação em Engenharia Elétrica – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2011.

RESUMO

Este trabalho apresenta o estudo sobre o desenvolvimento de um sistema de manutafura de placas de circuito impresso por meio do desbaste do cobre. Uma vantagem deste sistema é a substituição de processos químicos pelo processo físico apresentado uma solução sustentável. O trabalho utiliza programas que gerarão o código G necessário para estabelecer as coordenadas em que ocorrerá o desbaste para a formação das trilhas. Após a obtenção do código, o mesmo será transformado em passos que serão enviados pela porta serial para o microcontrolador e todo o controle da comunicação serial será por *software*. Após a obtenção das informações o microcontrolador irá executar o movimento dos motores de passo por meio dos *drivers*, ponte H, o microcontrolador também aciona o motor de *spindle*, responsável pela rotação da ferramenta de desbaste, por meio de um *driver* fotoacoplador juntamente com um TRIAC.

PALAVRAS-CHAVE: microcontrolador, comunicação serial, *drivers*, motor de passo.

Fontana, G. S. Study for implementation of a copper milling system for the manufacture of printed circuit boards. 2011. 102 f. Electrical Engineering Graduation Work – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2011.

ABSTRACT

This paper presents a study on the development of a manufacturing system of printed circuit boards through copper milling. An advantage of this system is the replacement of chemical processes by physical process presenting a sustainable solution. The paper uses programs that will generate G-code needed to establish the coordinates where the milling forms the tracks. After obtaining the code, it will be transformed into steps that will be sent through the serial port to the microcontroller and the serial communication control will be in software. After obtaining information the microcontroller will execute the movement of the stepper motors through their drivers, H-bridge, the microcontroller also drives the spindle motor responsible for rotating the grinding tool, using a driver with a optocoupler and TRIAC.

KEY-WORDS: microcontroller, serial communication, drivers, step motor.

LISTA DE FIGURAS

Figura 1 - Circuito esquemático 18
Figura 2 - Placa de Circuito Impresso 18
Figura 3 - Área de corte superior 19
Figura 4 - Área de corte inferior 19
Figura 5 - Parte do Código G 20
Figura 6 - 30° por passo. 23
Figura 7 - Sequência de Polarização 23
Figura 8 - Sequência com dupla polarização 24
Figura 9 - Curva de Torque por Velocidade 27
Figura 10 - Datasheet de um Motor de Passo 28
Figura 11 - Modelo do Motor de Passo 29
Figura 12 - Corrente em função do tempo no enrolamento 29
Figura 13 - Resistência em Série 30
Figura 14 - Comparação utilizando resistência em série 31
Figura 15 - método com duas fontes de tensão 32
Figura 16 - Comparação dos Métodos Vistos 33
Figura 17 - Forma de Onda da corrente e Tensão pelo método Chopper/PWM 34
Figura 18 - Pinagem do PIC16F877A 35
Figura 19 - Características do PIC 36
Figura 20 - Ponte H 40
Figura 21 - Curva Id por Vgs 41
Figura 22 - Curva Id x Vds para diferentes valores de Vgs 42
Figura 23 - Fluxo do programa em VB 44
Figura 24 - Tela Inicial do Programa 45
Figura 25 - Seleção da Operação 46

Figura 26 - Programa com o código G carregado 46
Figura 27 - Fluxo do tratamento dos comandos 47
Figura 28 – Subfluxo do tratamento dos comandos G00 e G01 48
Figura 29 - Subrotina do comando M02 50
Figura 30 - Fluxo main do PIC 51
Figura 31 - fluxo da função Ex_x 52
Figura 32 - fluxo da função ex_A 55
Figura 33 - IRF 630 datasheet 56
Figura 34 - IRF 630 gráfico Id x Vds em função de Vgs 57
Figura 35 - Datasheet IRF 9520 58
Figura 36 - *High Side* e *Low Side* da ponte H 58
Figura 37 - Driver do *Spindle* 60

SUMÁRIO

1 INTRODUÇÃO 14

1.1 Objetivo 14

1.2 Motivação 14

1.3 Discussão do problema 14

1.4 Metodologia utilizada e descrição dos capítulos 15

2 CAD E CAM 16

3 MOTORE DE PASSO 21

3.1 Escolha do motor de passo com relação ao tipo da aplicação. 24

3.1.1 Relutância variável x Imã permanente ou Hibrido 24

3.1.2 Unipolar x Bipolar 25

3.1.3 Hibrido x Imã permanente 25

3.2 Características funcionais dos motores 26

3.2.1 *Step Size* 26

3.2.2 Torque 26

3.2.3 Longevidade 27

3.3 Limitação de Corrente 28

3.3.1 Modelo de Motor de passo 28

3.3.2 Resistência em Série 30

3.3.3 Duas fontes de tensão 31

3.3.4 Limitar a corrente utilizando controle Chopper 33

4 Microcontrolador e Programação em C 35

5 COMUNICAÇÃO SERIAL 37

5.1 Controle do fluxo de dados 38

6 VISUAL BASIC E PROGRAMAÇÃO ORIENTADA A OBJETOS 39

7 DRIVER DE ACIONAMENTO DE MOTORES 40

7.1 Ponte H 40

7.2 Driver do Motor *Spindle* 43

8 DESENVOLVIMENTO 44

9 CONCLUSÃO 61

ANEXO A – Programação do microcontrolador em C 63

ANEXO B - Programa Form1 em Visual Basic 85

Anexo C - Programa Form1 em Visual Basic 100

Anexo D – Circuito do PIC com o driver do eixo X 101

1 INTRODUÇÃO

1.1 Objetivo

O estudo sobre o desenvolvimento de um sistema que objetiva a manufatura de placas de circuito impresso de simples ou dupla face com dimensões de placas e de largura de trilhas limitadas. Tal sistema será constituído de: movimentos com três graus de liberdade correspondendo aos eixos x, y e z; e ferramentas de fresa para o desbaste do cobre e a broca para a realização de furos cilíndricos. O estudo também engloba como será feita a transmissão da informação, via leitura óptica de um leiaute impresso ou, se possível, por meio de software e transferência de dados de um leiaute eletrônico. Eventuais adaptações na idéia inicial podem ser estudadas e implementadas.

1.2 Motivação

A motivação deste trabalho é a crescente preocupação com o meio ambiente, visando a sustentabilidade, pois, este método de desenvolvimento de placas de circuito impresso é uma alternativa para o método de corrosão de cobre. Com este sistema é possível desenvolver uma placa de circuito impresso sem a necessidade do banho químico, procedimento que gera um descarte de produtos químicos na natureza. Não tendo que buscar serviço em uma empresa especializada em métodos de corrosão, evitanto todo o processo burocrático, minimizando custos e tempo.

1.3 Discussão do problema

Atualmente o mundo está em uma fase de grande crescimento global, porém a necessidade por processos e medidas sustentáveis já estão começando a ser praticadas por grandes empresas. Grandes empresas fabricantes de PCB utilizam processos químicos para fabricação de PCB´s em larga escala de forma rápida para atendero mercado aquecido de eletrônicos. Existem alguns fornecedores de placas de circuito impresso que possuem máquinas *PCB(printed circuit boards) milling* principalmente como forma de estudo para

implementação em larga escala, porém ainda sem viabilidade. A grande utilidade das PCB *milling* é a sua utilização em pequena escala e na utilização em protótipos.

1.4 Metodologia utilizada e descrição dos capítulos

O trabalho está estruturado da seguinte forma:

O capítulo dois, CAD e CAM, apresenta o que é CAD e CAM e como ele é utilizado para a obtenção do código G.

O capítulo três, Motor de passo, apresenta as principais características dos motores de passo e explica sobre a utilização dos mesmos usando a limitação de corrente.

o capítulo quatro, Microcontrolador e Programação em C, tem-se breve introdução sobre o microcontrolador utilizado e sobre a programação em C.

O capítulo cinco, Comunicação Serial, apresenta uma breve introdução teórica sobre a comunicação serial, como é feito o controle do fluxo de informações e também sobre a emulação de portas seriais.

O capítulo seis, Visual Basic e programação orientada a objetos, o capítulo menciona o que é uma programação orientada a objetos e a sua aplicação no trabalho.

O capítulo sete, Driver de acionamento de motores, o capítulo descreve sobre a ponte H e sobre o driver do motor *spindle* utilizado.

O capítulo oito, Desenvolvimento, apresenta todo o desenvolvimento do trabalho, o tratamento do código G, a comunicação serial e o acionamento dos motores.

O capítulo nove, Conclusão, a conclusão apresenta alguns comentários sobre o sistema e também possibilidades de melhorias.

2 CAD E CAM

CAD é a sigla para *Computer-aided design*, como a sigla sugere, é a utilização de computação para a elaborar desenhos ou modelos. Neste caso o software de CAD ajuda o usuário com ferramentas para desenhar o circuito que será impresso na placa de cobre. Hoje no mercado existem inúmeros softwares que fazem estes desenhos, o software utilizado neste trabalho é o EAGLE, *Easily Applicable Graphic Layout Editor*, uma ferramenta de fácil utilização para desenhar os leiautes de placas de circuito impresso.

O EAGLE possui três modulos que facilitam o desenho do circuito: *Schematic*, *Layout* e *Autorouter*. O módulo *Schematic* faz o desenho esquemático do circuito impresso e verifica a integridade das conexões elétricas e a correlação do circuito esquemático com o leiaute da placa por meio de modelos pré-configurados. O módulo *Layout* define o modelo e a posição dos componentes eletrônicos no leiaute. O módulo *Autorouter* possui um algoritmo que desenha as trilhas dos circuitos, as trilhas são desenhadas de acordo com as especificações dadas como a largura mínima e máxima e também o número de camadas da placa de circuito impresso (CADSOFT, 2011).

Além destas utilidades o software também possui um módulo ULP, *User Language Program*. Com este módulo é possível programar em linguagem C integrado com os dados provenientes do EAGLE, com essa combinação inúmeras funções podem ser criadas, e uma delas é a criação de um CAM (*Computer-Aided Manufacturing*).

CAM é um software computacional que controla máquinas e ferramentas para a manufatura de peças, ajudando a aumentar o processo de produção e também para dar maior precisão na confecção das peças, além de minimizar desperdícios de matéria prima e consumo de energia. Tradicionalmente, CAM é considerado como um software que utiliza modelos de CAD para gerar o código G que é um código numérico computacional utilizado para controlar máquinas CNC (*Computer Numerical Control*).

Este trabalho terá auxílio de uma ULP que utiliza a placa desenhada no EAGLE para gerar o código G. O quadro 1, a seguir, mostra os comandos do código que serão utilizados

no trabalho. Apesar da ULP gerar mais comandos, esses comandos são o mínimo necessário para que a máquina CNC consiga executar o trabalho.

Quadro1:

G01	Especifica os pontos de começo e fim para as coordenadas de corte, por meio de interpolações lineares
M03	Inicia a rotação do motor *spindle*, que realiza o corte ou o furo.
M05	Para a rotação do motor *spindle*
M02	Código de finalização de Programa.

Para melhor entendimento de como o software e a ULP funcionam serão mostradas nas figuras 1, 2, 3, 4 e 5, respectivamente, um exemplo de um esquemático, a sua placa de circuito impresso, a parte inferior da camada de cobre a ser fresada, a parte superior da camada de cobre a ser fresada e parte do código G gerado pela ULP. O esquemático e a placa de circuito impresso mostrados são parte de um exemplo do software EAGLE.

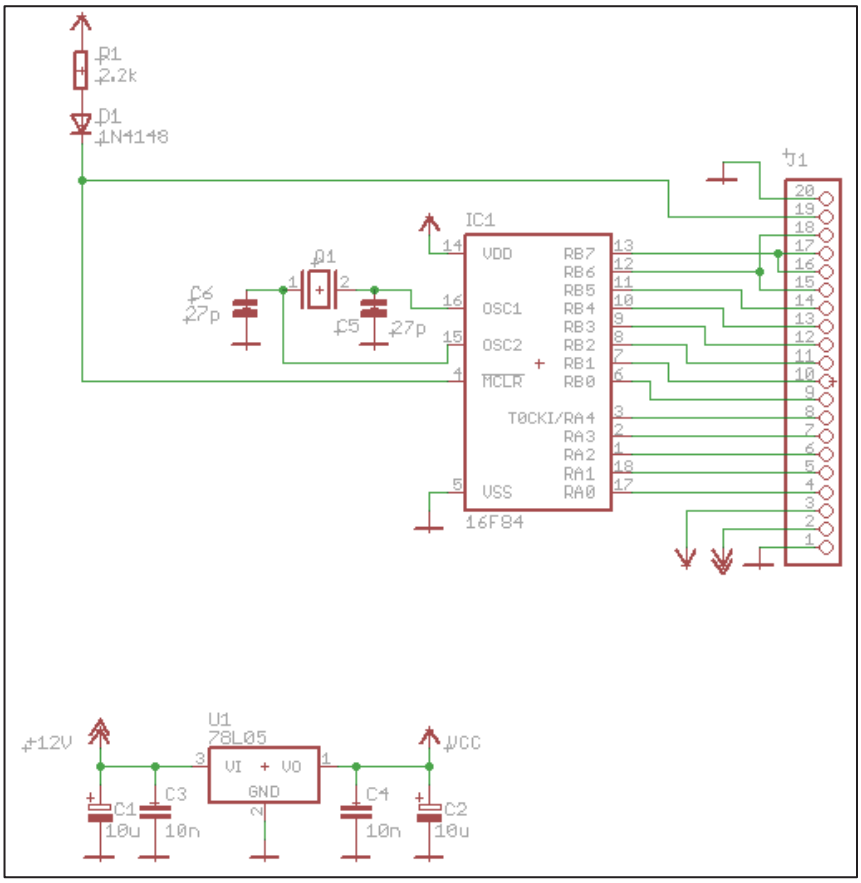


Figura 1 - Circuito esquemático (CADSOFT,2011)

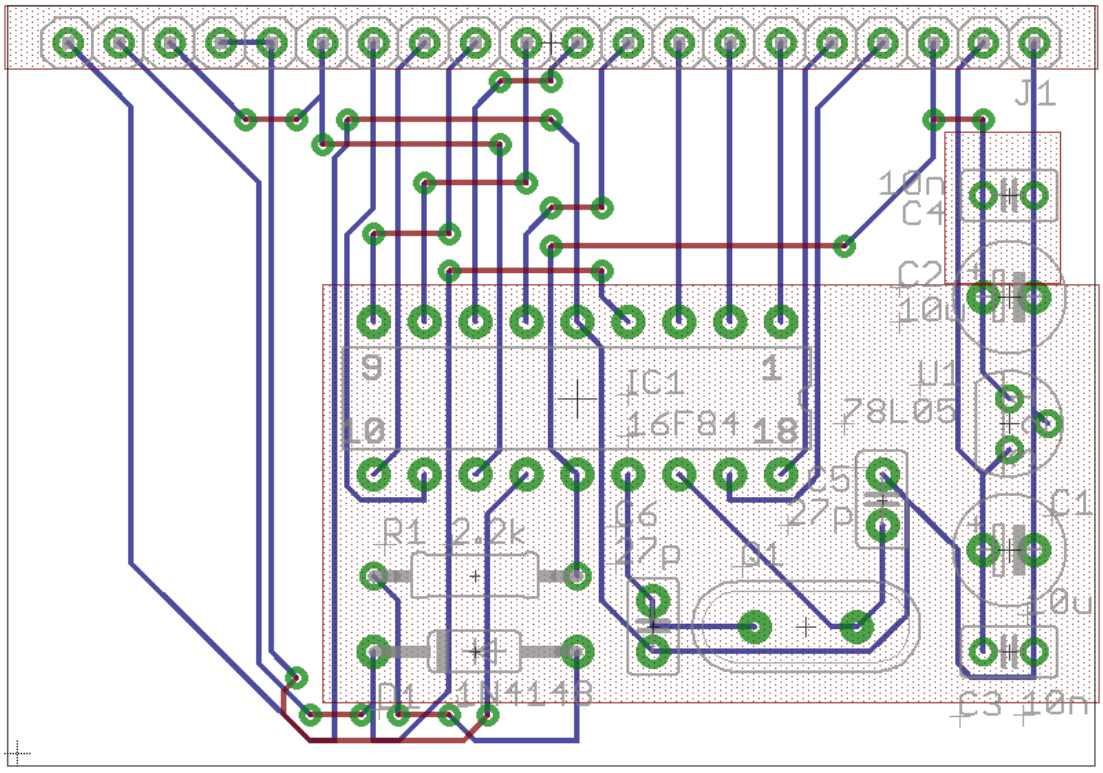


Figura 2 - Placa de Circuito Impresso (CADSOFT,2011)

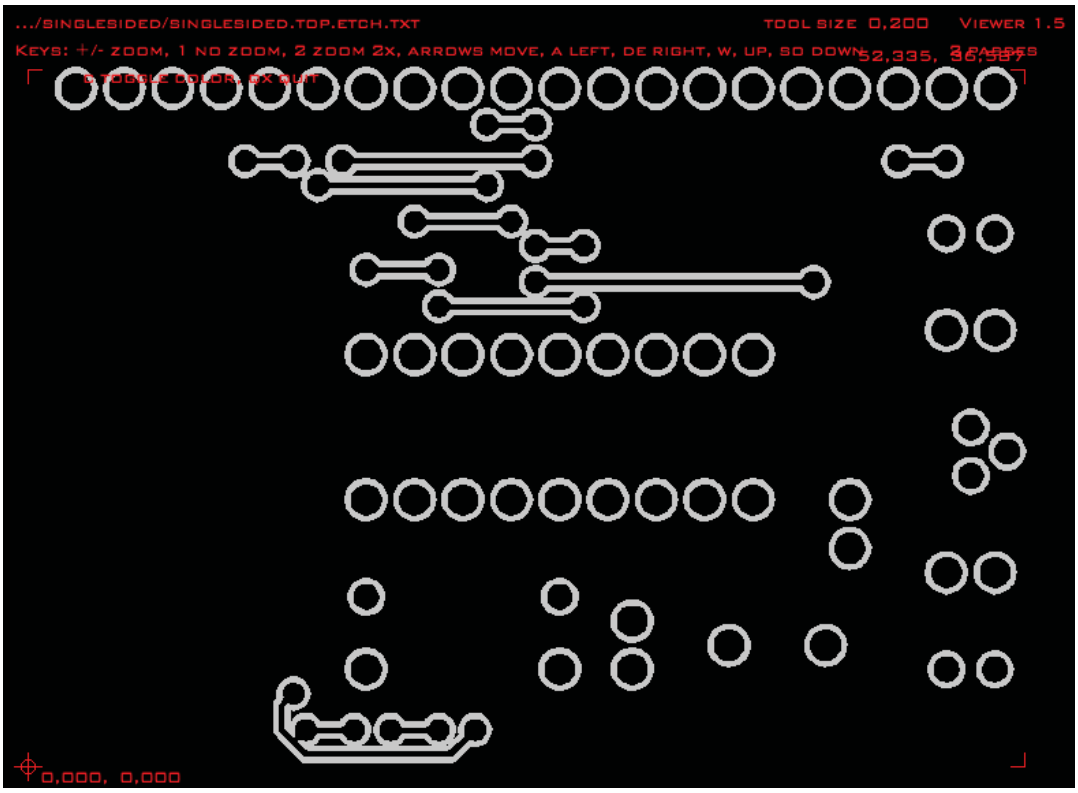


Figura 3 - Área de corte superior

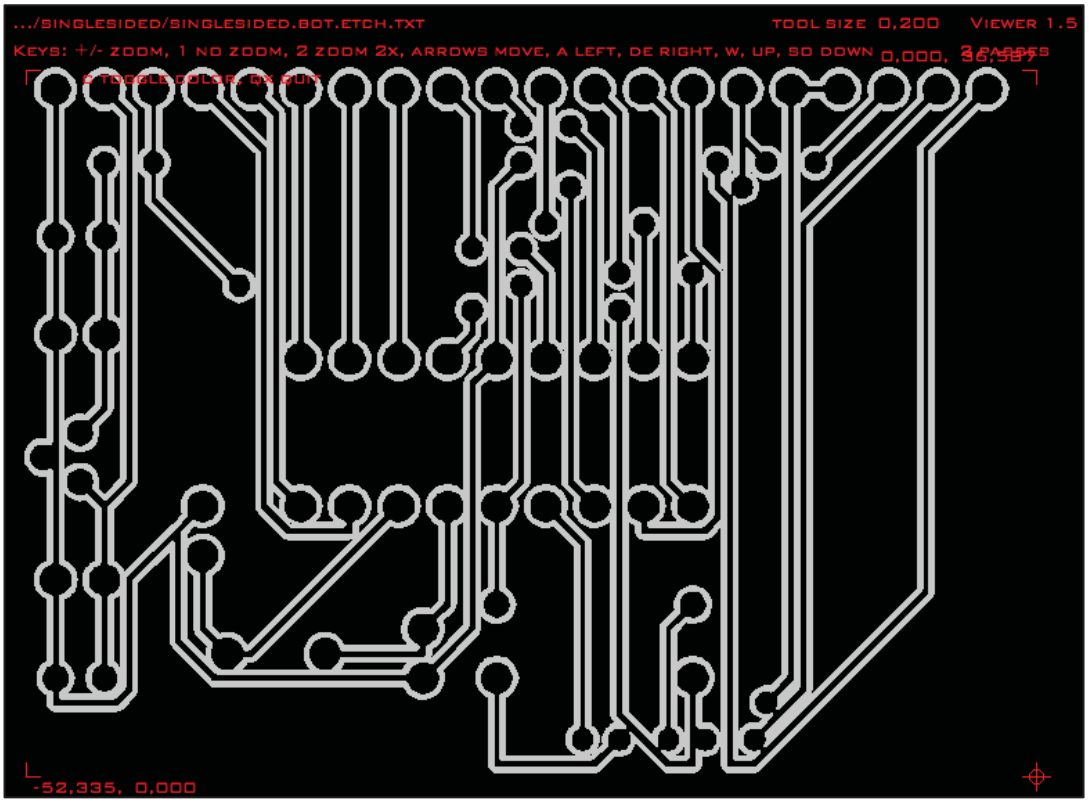


Figura 4 - Área de corte inferior

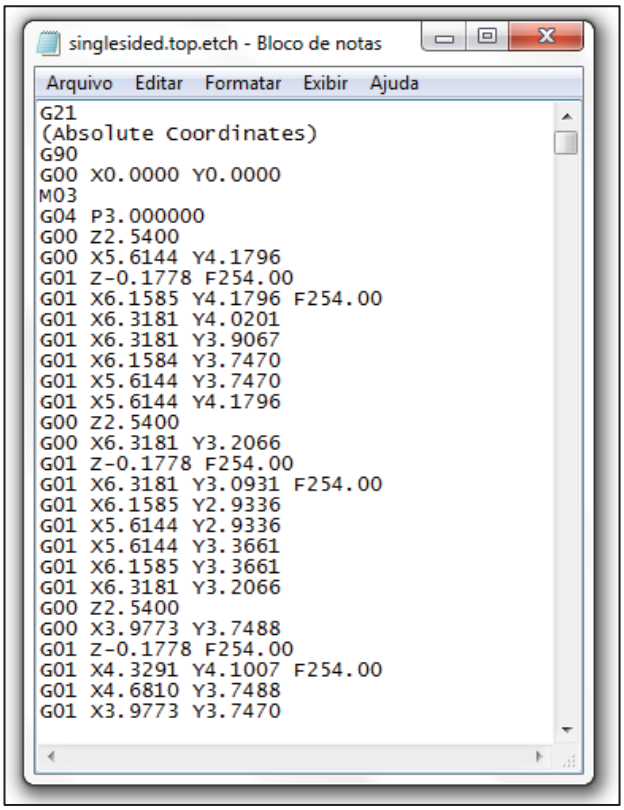


Figura 5 - Parte do Código G

A figura 2 mostra o circuito impresso com trilhas em duas cores, azul e verde, neste caso a cor verde é referente às trilhas na superfície superior da placa de circuito impresso, e a azul na superfície inferior. As figuras 3 e 4 mostram as superfícies superior e inferior da placa de circuito impresso, inicialmente, deve-se ter em mente que uma placa de circuito impresso virgem possui cobre em toda a sua superfície, as figuras mostram em branco as áreas que deverão sofrer o desbaste do cobre, as áreas em branco fazem um contorno nas trilhas da figura 2. Nas figuras 3 e 4 também é possível ver o referencial usado, o tamanho da placa em militros e o raio da ferramenta, *tool size*, também em milímetros.

A figura 5 mostra um trecho do código G derivado da ULP que por meio de interpolação calcula as coordenadas da trajetória onde ocorrerá o desbaste.

Esta é apenas uma introdução teórica de como funciona o *software* EAGLE e a ULP, no próximo capítulo também será apresentada uma introdução teórica sobre o motor de passo, o principal componente de atuação no sistema, no qual as coordenadas apresentadas anteriormente deverão ser transformadas em movimento. (JOHNSON,2009)

3 MOTOR DE PASSO

Neste capítulo a teoria sobre os motores de passo será apresentada, primeiramente falando sobre os tipos de motores de passo com suas respectivas vantagens, algumas características físicas relevantes para a escolha dependendo da aplicação. Também será apresentado um modelo e figuras para entender o porque de se utilizar uma tensão de alimentação maior fazendo o controle da corrente máxima nos enrolamentos.

O motor de passo é um motor elétrico sem buchas que divide a rotação em passos. A grande vantagem deste motor é controlar a posição do motor sem a necessidade de um mecanismo de *feedback*, ou seja, apartir de um controle por malha aberta. Diferentemente de motores de corrente contínua, que rotacionam quando é aplicada tensão nos terminais, os motores de passo possuem pólos eletromagnéticos no estator que se alinham magneticamente com o rotor, quando um pólo é energizado o rotor gira de modo a se alinhar com o pólo, nesta nova posição o rotor está deslocado do próximo pólo, de modo que, se o próximo pólo é energizado o rotor gira novamente para se alinhar com o novo pólo, para cada movimento dá-se o nome de passo, portanto, o número de pólos do motor define a quantidade de passos necessários para uma volta.

Outras características do motor de passo estão descritas a seguir: (JONES, 2004)

- Utilizam tensão contínua.
- O torque é inversamente proporcional à velocidade.
- A curva de torque pode ser otimizada usando um limitador de corrente aumentando a tensão de alimentação
- Os motores apresentam mais vibração por causa dos passos discretos, a vibração pode fazer com que em algumas velocidades o motor perca torque ou direção. Isto porque a cada passo o motor tende a se alinhar com o campo magnético do pólo o que produz um ricocheteio até a posição alinhada com uma frequência ressonante, se a frequência do passo se alinhar com a frequência ressonante, então, a vibração irá aumentar e o motor ficará fora de sincronismo perdendo o controle. Este efeito pode ser mitigado acelerando rapidamente o motor pelo range de velocidades que produzem a ressonância ou por meio de um *micro-stepping driver*.

Como dito anteriormente uma das vantagens do motor de passo é o controle que se pode obter por meio de uma malha aberta, mas para isto deve ser feito um estudo sobre como a carga se comporta e varia. Se a carga é muito alta ou a variação durante a operação é muito alta, é provável que o motor perca passos. Portanto é necessário realizar um estudo para analisar se é viável utilizar um mecanismo de servo motor em malha fechada com alto custo e de pequenas dimensões, ou utilizar um motor de passo de grandes dimensões porém mais com custo reduzido.

Atualmente existem quatro tipos principais de motores de passo:

1. Imã permanente
2. Híbrido
3. Relutância variável
4. Lavet

Motores de passo utilizam imã permanente no rotor e operam de acordo com a atração e repulsão dos imãs no rotor e o campo eletromagnético do estator. Os motores de relutância variável utilizam um rotor com núcleo de ferro com o princípio de que a mínima relutância ocorre na menor distância entre o rotor e o estator, e com isso ocorre o alinhamento com os pólos eletromagnéticos do estator. Os motores Híbridos utilizam uma combinação das duas técnicas para melhorar a eficiência, com altas potências e baixo encapsulamento. O tipo Lavet é utilizado na construção de relógios eletromecânicos, a sua estrutura é diferente de um motor de passo usual, ele somente rotaciona em uma direção e é uma combinação de estator energizado e de estator *currentless*, diminuindo o consumo de energia (Yeadon,2001).

Os motores de passo eletromagnéticos podem ser construídos com dois tipos diferentes de enrolamentos, o bipolar e o unipolar.

O enrolamento unipolar, mostrado na figura 6 a seguir, possui um enrolamento com center tap em cada fase, cada seção deste enrolamento é acionado para cada direção do campo magnético. Neste arranjo o pólo magnético pode ser revertido sem alterar a direção da corrente, a comutação do circuito pode ser feita de modo simples, com um transistor para cada enrolamento.

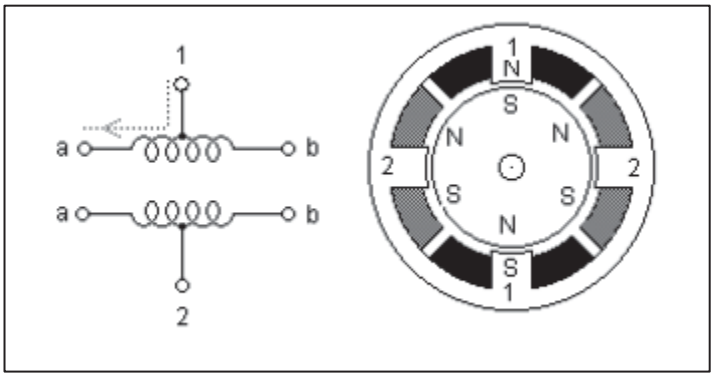


Figura 6 - 30° por passo.(CONDIT; JONES, 2004)

Para rotacionar o motor no sentido horário é necessário energizar o enrolamento conforme descrito na sequência da figura 7 a seguir.

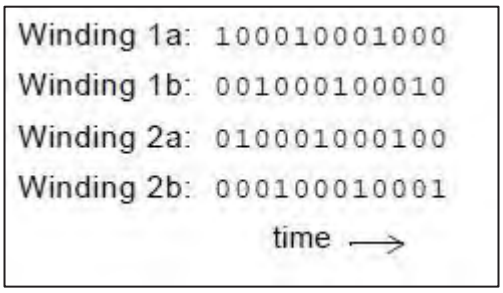


Figura 7 - Sequência de Polarização (CONDIT; JONES, 2004)

Nesta sequência somente metade de um enrolamento é energizado por vez, é possível acionar também o polo oposto aumentando o torque como na sequência da figura 8 a seguir:

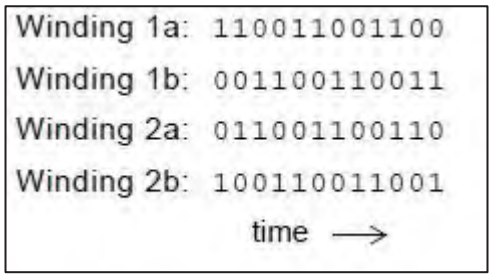


Figura 8 - Sequência com dupla polarização (CONDIT; JONES, 2004)

O enrolamento bipolar possuem um enrolamento por fase, a corrente em um enrolamento precisa ser revertida para reverter o pólo magnético, por isto, o circuito para o seu acionamento é mais complicado, tipicamente uma ponte H. Nesta topologia o enrolamento é energizado por inteiro, portanto, ele possui um torque maior. Este tipo de motor pode ser ligado de diversas maneiras descritas a seguir:

- Unipolar.
- Bipolar Série, possui alta indutância e menos corrente por enrolamento.
- Bipolar Paralela, possui maior corrente e menor indutância.
- Bipolar com um único enrolamento por fase. Este método utiliza somente metade dos enrolamentos que reduz o torque a baixas velocidades e reduz a corrente necessária.

3.1 Escolha do motor de passo com relação ao tipo da aplicação.

Primeiramente é necessário entender e escolher o tipo de motor que vai ser utilizado, esta escolha influenciará no torque necessário no sistema, na complexidade do controlador e também nas características físicas do motor.

3.1.1 Relutância variável x Imã permanente ou Hibrido

Motores de Relutância variável se beneficiam da simplicidade da sua modelagem, Esses motores não precisam de complexos rotores eletromagnéticos e geralmente são mais robustos. Como todos os motores, o torque decai com o aumento da velocidade, porém essa

redução é menos acentuada nos motores de relutância variável. Alguns modelos conseguem uma velocidade de 10.000 passos por segundo, enquanto os melhores motores de imã permanente ou híbrido conseguem 5.000 passos por segundo, e os mais usuais conseguem 1.000 passos por segundo. O decaimento do torque conforme o aumento da velocidade é menor possibilitando o uso do motor sem a necessidade de uma caixa de engrenagem para aumento do torque, entretanto, os motores com relutância variável apresentam mais barulho e vibração com relação aos motores de imã permanente ou híbridos. Os motores de imã permanente e híbrido sapresentam uma resistência quando são girados mecanicamente sem que nenhum enrolamento esteja energizado, esta resistência ao giro é chamada de torque residual, o que pode ocasionar problemas na desaceleração do motor, este problema pode ser solucionado utilizando controles de *microstep* possibilitando o posicionamento em uma fração de passo evitando folgas entre os passos (CONDIT; JONES, 2004).

3.1.2 Unipolar x Bipolar

Motores de Imã permanente ou híbrido podem possuir enrolamento unipolar ou bipolar. A escolha entre esses dois enrolamentos é devido à simplicidade do *driver* e da taxa de potência relacionada com o peso do motor. Motores bipolares possuem aproximadamente 30% mais torque do que o equivalente unipolar de mesmo volume, a razão para isto é que apenas metade do enrolamento é energizado por vez, enquanto que o bipolar utiliza todo o enrolamento. Porém o uso de um enrolamento bipolar necessita de um circuito de controle com maior complexidade.

3.1.3 Hibrido x Imã permanente

A principal diferença entre estes dois tipos é o custo e a resolução, geralmente o driver e a forma de conectar os enrolamentos são aplicáveis a ambos. Os motores de Imã permanente possuem custo reduzido devido a sua confecção, os motores hibridos e de relutância variável são construídos usando núcleos de camadas de lâminas aumentando o custo do motor. A resolução dos motores com mais de 50 pólos são raros, então, a resolução normalmente é de 30° a 3.6°, motores híbridos podem facilmente chegar a 1.8° com a

construção de finos dentes no rotor. Os motores hibridos também sofrem com problemas de vibração como os motores de relutância variável, porém com menos intensidade, geralmente os hibridos podem chegar a 5000 passos por segundo, maiores velocidades fazem com que o torque diminua, perdendo assim a sua utilidade.

3.2 Características funcionais dos motores

Após selecionar o tipo de motor ainda deverão ser considerados inúmeras características como o torque, o ambiente de operação, a longevidade, o tamanho físico, o tamanho do passo e a rotação máxima (Yeadon,2001).

3.2.1 *Step Size*

É uma das características mais importantes definida pela resolução necessária da aplicação, os passos mais comuns de motores de imã permanente são de 7.5 e 3.6 graus, correspondendo a 48 e 100 passos respectivamente. Motores híbridos possuem normalmente 3.6 a 0.9 graus, 100 a 400 passos por revolução. Alguns motores de passos são vendidos juntamente a redutores o que produz passos menores e aumenta o torque, porém diminuem a máxima rotação.

3.2.2 Torque

O torque é uma característica crítica, existem diferentes tipos de torque com as seguintes definições descritas a seguir:

Holding torque – é o torque requerido para rotacionar o motor enquanto um enrolamento estiver energizado.

Pull-in-torque – é o torque necessário para que o motor acelere de uma posição estática sem perder movimento.

Pull-out-torque – é a carga que o motor consegue mover em velocidade de operação.

Detent torque – é o torque necessário para rotacionar o motor enquanto o enrolamento não estiver energizado.

Dependendo do fabricante nem todos esses torques são definidos no *datasheet*. Esses torques são importantes para determinar se o motor irá funcionar corretamente, mover e não mover quando necessário pois se o motor não funcionar corretamente, o controlador irá perder a posição de referência, quando em malha aberta. O torque fornecido pelo motor depende do momento de inércia da carga acoplada ao eixo, portanto alguns fornecedores oferecem uma curva de torque como na figura:

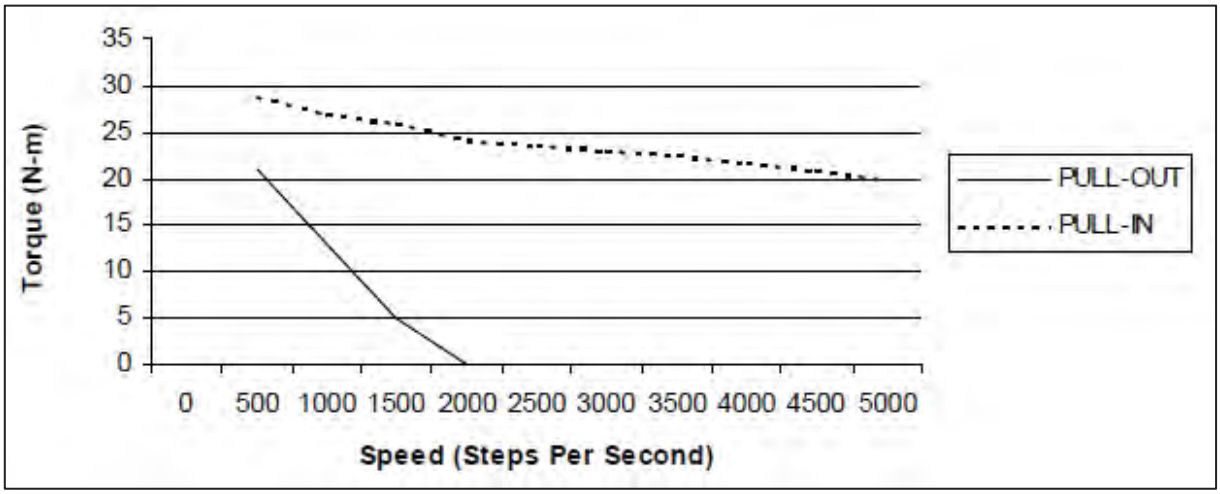


Figura 9 - Curva de Torque por Velocidade(CONDIT; JONES, 2004).

3.2.3 Longevidade

A longevidade de um motor é uma característica importante em algumas aplicações portanto é necessário saber quanto tempo o motor precisa trabalhar adequadamente, qual o ambiente de trabalho que o motor estará submetido, a temperatura de trabalho do motor, se a operação é contínua ou intermitente. Os motores de passo são mais robustos que os motores que utilizam escovas, que se desgastam com o tempo, muitas das vezes outros componentes do sistema vão se desgastar mais rapidamente do que o motor. Portanto, é

importante escolher o tipo de motor e fazer todas as considerações para que não ocorra problemas na sua utilização.

A seguir na figura 10 um exemplo simples de uma especificação de motor de passo:

Step Angle	Voltage	Current	Resistance	Inductance	Holding Torque	Rotor Inertia	Number of Leads	Weight
Deg	V	A/Phase	Ohm/Phase	mH/Phase	kg-cm	$g\text{-}cm^2$	LEAD	kg
1.8	1.8	4.5	0.4	0.96	16	570	6	1.4

Figura 10 - Datasheet de um Motor de Passo (CONDIT; JONES, 2004).

3.3 Limitação de Corrente

Um motor de passo eventualmente é energizado com correntes maiores do que as especificadas. Motores com altos torques precisam ser alimentados com tensões superiores para que o motor atinja seu máximo desempenho, aumentar a tensão alimentada por um motor significa aumentar a quantidade de corrente nos enrolamentos do motor, e consequentemente, aumentar a resposta de torque e velocidade do motor.

3.3.1 Modelo de Motor de passo

Para melhor compreender a necessidade de se alimentar motores de passo com correntes superiores da especificada é necessário entender como a corrente se comporta no enrolamento do motor, a modelagem é mostrada na figura 11 a seguir:

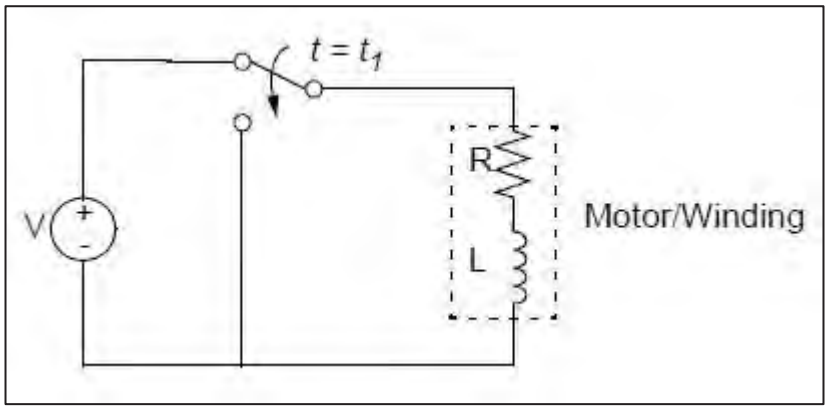


Figura 11 - Modelo do Motor de Passo (CONDIT; JONES, 2004).

A modelagem do motor de passo é simples consistindo em uma resistência em paralelo com uma indutância, a seguir será mostrada a figura 12 do comportamento da corrente em função do tempo.

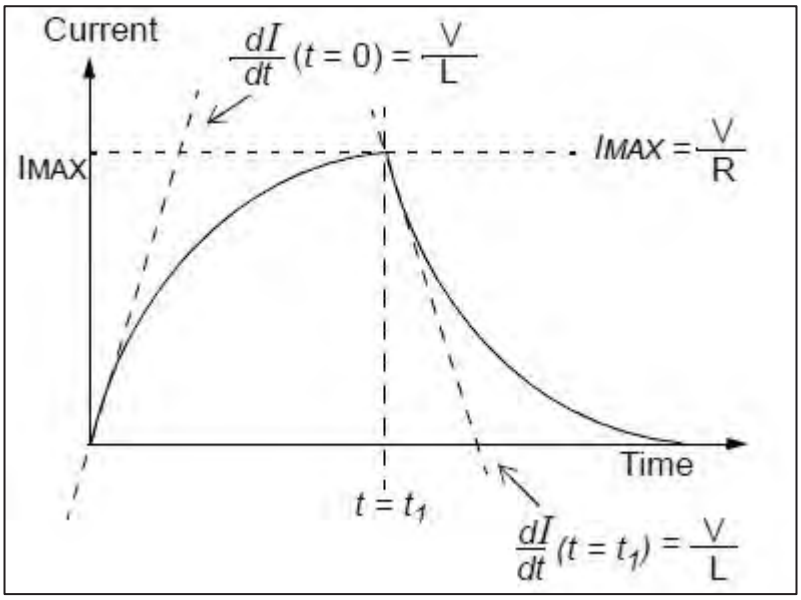


Figura 12 - Corrente em função do tempo no enrolamento (CONDIT; JONES, 2004).

Pode-se notar que a corrente demora um tempo $t = t_1$ para alcançar o seu valor máximo como na expressão (3.1), devido à indutância do motor com a equação (3.2) a seguir.

$$I_{MAX} = \frac{V}{R} \tag{3.1}$$

$$I_{(t)} = \left(\frac{V}{R}\right) \times \left(1 - e^{-t \times \frac{R}{L}}\right) \quad (3.2)$$

Este tempo de alguns milissegundos para alcançar o valor máximo da corrente pode comprometer todo o sistema, o torque pode não ser o suficiente, ou, a velocidade não poderá ser elevada acarretando perda de passos. Para contornar este problema algumas medidas podem ser tomadas

3.3.2 Resistência em Série

O primeiro método para aumentar a tensão de alimentação, sem ultrapassar o limite da corrente máxima é colocando um resistor em série com o enrolamento do motor utilizando uma tensão de alimentação maior, a figura a seguir mostra um exemplo:

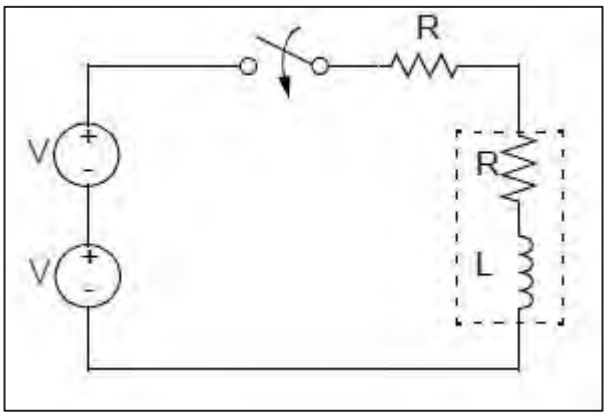


Figura 13 - Resistência em Série (CONDIT; JONES, 2004).

Nesta figura a tensão de alimentação agora é $2V$ e a resistência total é $2R$. Utilizando a resistência adicional em série a equação da corrente em função do tempo (3.3) é dada a seguir.

$$I_{(t)} = \left(\frac{2V}{2R}\right) \times \left(1 - e^{-t \times \frac{2R}{L}}\right) \quad (3.3)$$

Nota-se que a corrente máxima é $\left(\frac{2V}{2R}\right)$ ou seja a corrente total ainda será $\left(\frac{V}{R}\right)$ não ultrapassando o limite. A porção exponencial da equação é diferente, apresentando um ganho no tempo necessário para atingir a corrente máxima, a figura 14 a seguir demonstra tal efeito:

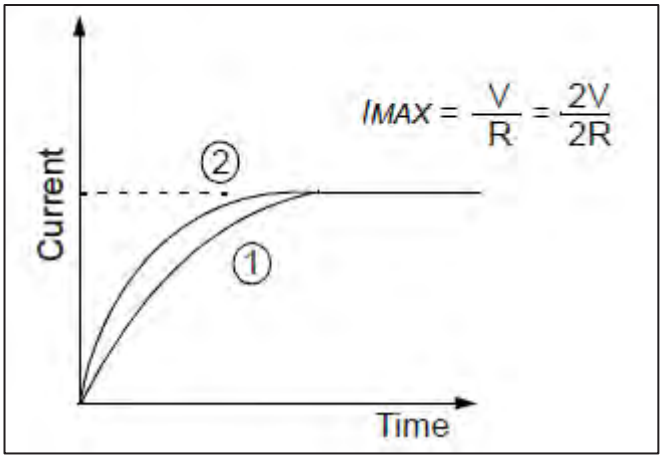


Figura 14 - Comparação utilizando resistência em série (CONDIT; JONES, 2004).

Vários problemas surgem desta solução, o primeiro é a utilização de um resistor de potência de valor igual ao do enrolamento do motor, pegando-se o exemplo do motor da figura 10 a resistência do enrolamento é extremamente baixa, 0.4Ω. Para um resistor desta magnitude dissipar uma corrente elevada, a capacidade de dissipação do resistor deverá ser muito elevada, e consequentemente o seu tamanho, inviabilizando o sistema. Outro problema é a eficiência do circuito, com a introdução da resistência em série a eficiência cai pela metade e ainda faz com que a temperatura aumente, tudo isto, devido à dissipação no resistor.

3.3.3 Duas fontes de tensão

Uma outra forma para resolver o problema é eliminar a resistência em série, mas, continuar com o aumento da tensão, portanto, utiliza-se a técnica de duas fontes de tensão como na figura a seguir:

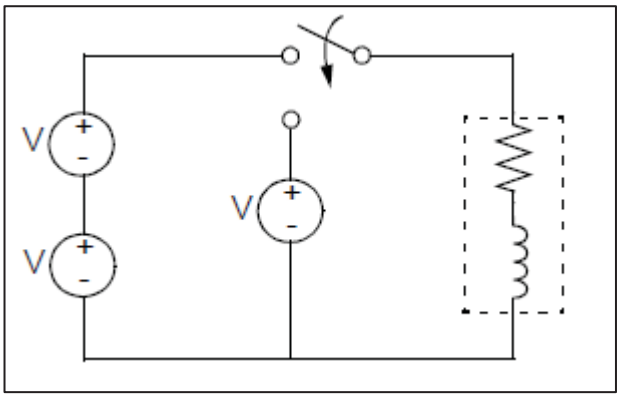


Figura 15 - método com duas fontes de tensão (CONDIT; JONES, 2004).

O circuito mostrado utiliza uma chave que comuta entre duas fontes e uma fonte em série. É necessário realizar a comutação para evitar que o motor trabalhe com a corrente maior do que a especificada. A comutação é feita por meio de uma malha aberta ou fechada. Utilizando a malha aberta o tempo para comutação é calculado através da equação (3.4), portanto, utilizando um circuito temporizador atuando como chave, por malha fechada um resistor de baixa resistência fornece o *feedback* necessário para o circuito de controle atuar e realizar a comutação.

$$I_{(t)} = \left(\frac{2V}{R}\right) \times \left(1 - e^{-t \times \frac{R}{L}}\right) \qquad (3.4)$$

A figura 16 a seguir mostra uma comparação dos acionamentos vistos até agora:

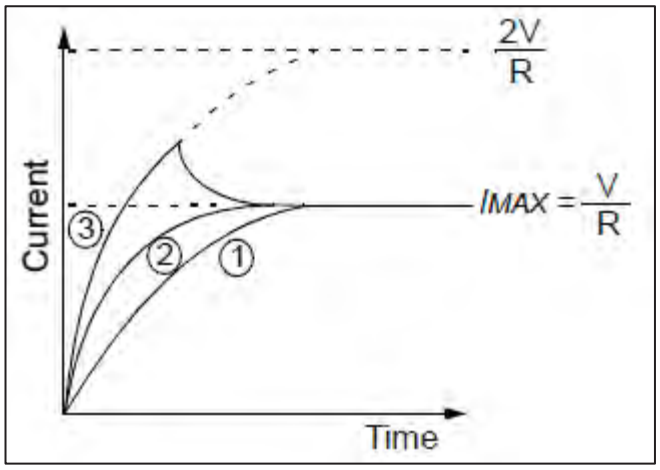


Figura 16 - Comparação dos Métodos Vistos (CONDIT; JONES, 2004).

A curva número 1 corresponde ao motor com alimentação nominal, a curva número 2 corresponde à resistência em série e a curva número 3 utilizando duas fontes.

Apartir da figura 16 nota-se que o método utilizando duas fontes é a melhor opção, mas ele também apresenta desvantagens. Uma das desvantagens é a utilização de duas fontes, trabalhar com duas fontes de tensão apresenta maior custo para o sistema. Um outro problema é o *overshoot* que a curva apresenta, este problema depende da solução adotada pelo circuito de controle, podendo ser insignificante devido ao curto tempo de duração, porém, como a velocidade de acionamento dos motores pode ser muito elevada esta corrente de overshoot pode não ser desprezível acarretando problemas quanto a longevidade do motor.

3.3.4 Limitar a corrente utilizando controle Chopper

A melhor solução praticada atualmente é a utilização de um circuito Chopper para limitar a corrente. A palavra *chopper* proveniente da língua inglesa significa cortar ou fatiar, portanto, a ideia desta solução é cortar a tensão de alimentação quando a corrente máxima é atingida. Um sinal PWM (*Pulse Width Modulation*) é utilizado para criar uma tensão e corrente perto da nominal após a corrente máxima ser atingida. A figura 17 a seguir mostra a forma de onda da corrente e tensão.

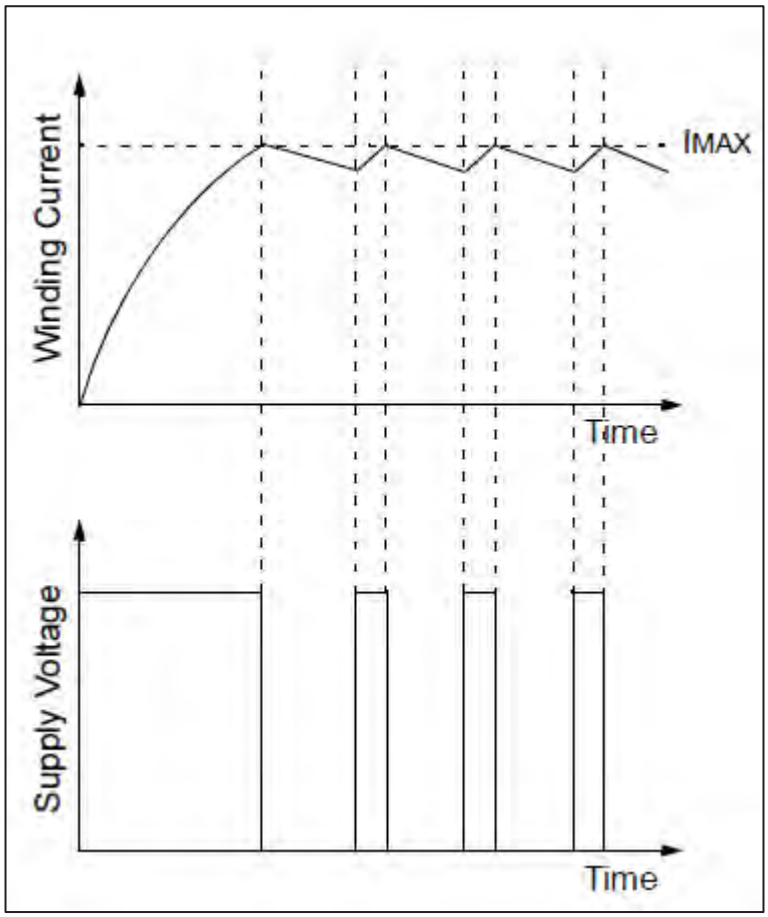


Figura 17 - Forma de Onda da corrente e Tensão pelo método Chopper/PWM (CONDIT; JONES, 2004).

Nota-se da figura 17 que no início o circuito chopper não deve atuar, ele atua somente depois de um determinado tempo, quando a corrente atinge o limite máximo. A ideia do circuito de controle é equivalente ao mencionado no método das duas fontes. A solução por malha aberta deve ser muito bem calculada e o circuito muito bem projetado, o que não é uma tarefa simples, muitas das vezes devido à indutância ser mal documentada.

No capítulo anterior foi apresentado o funcionamento do software EAGLE para a criação do código G, e neste capítulo foi apresentado o funcionamento de um motor de passo. O próximo capítulo terá uma breve introdução sobre o microcontrolador e como foi feita a programação em linguagem C.

4 Microcontrolador e Programação em C

Este capítulo será uma breve introdução sobre o microcontrolador PIC (*Peripheral Interface Controller*) utilizado e sobre a programação em C.

Para a programação no microcontrolador PIC em linguagem C é necessário utilizar um programa adicional que realiza a transformação da linguagem C em *assembler* para depois ser transformada em linguagem hexadecimal e binária, esta que será utilizada pelo microcontrolador para execução das funções. O *software* utilizado é o CCS (*Custom Computer Services*) especializado em desenvolvimento com microncontroladores da Microchip PIC. Além da linguagem de programação C, o CCS possui bibliotecas com funções pré-definidas que auxiliam o desenvolvimento de projetos com microcontroladores.

O microcontrolador a ser utilizado é o PIC16F877A. A escolha por este microcontrolador se deve pela quantidade de portas I/O, no máximo 33, pelo protocolo de comunicação USART(*Universal asynchronous receiver/transmiter*) e pelos módulos PWM, que apesar de não utilizados, foram considerados para futuros trabalhos.

As figuras 18 e 19 a seguir mostram respectivamente, a pinagem do microcontrolador e algumas características.

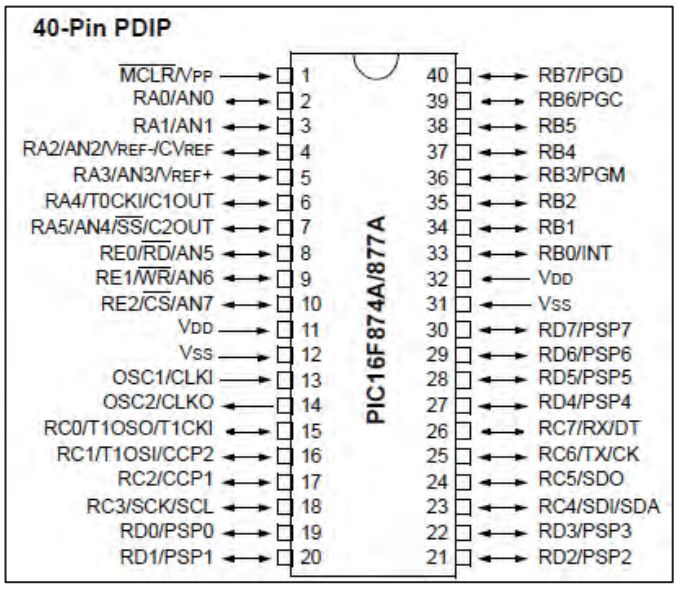


Figura 18 - Pinagem do PIC16F877A

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM),
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (VREF) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

Figura 19 - Características do PIC

O papel microcontrolador neste trabalho é acionar os motores de acordo com as informações provenientes do computador por meio da porta serial, o próximo capítulo uma terá uma introdução teórica sobre a comunicação serial, responsável pela comunicação do computador com o PIC.

5 COMUNICAÇÃO SERIAL

Neste capítulo uma breve introdução teórica sobre a comunicação serial, como é feito o controle do fluxo de informações e também sobre a emulação de portas seriais.

A comunicação serial é o processo de envio de dados de forma sequencial bit a bit, por meio de um canal de comunicação, este tipo de comunicação é usualmente comparada com a comunicação em paralelo, na qual vários bits são enviados simultaneamente. Apesar de interfaces como a Ethernet, FireWire e USB enviarem dados de forma serial, o termo comunicação serial será usado para referenciar o hardware *RS-232 standard*, ainda muito utilizado em aplicações industriais de automação de sistemas.

Apesar de parecer que uma comunicação paralela pode transmitir dados de forma mais rápida que a serial, ela não transmite. Isto acontece porque os sistemas que utilizam a comunicação serial utilizam uma frequência de *clock* mais rápida. Um fator que permite que a comunicação serial possua uma frequência maior de *clock* é a utilização de uma comunicação assíncrona, o que anula o fenômeno do *clock skew.* Este fenômero acontece quando se utiliza uma comunicação síncrona de alta velocidade, na qual o sinal de *clock* gerado não é o mesmo entre dois componentes distintos, normalmente ocasionado porque em um dos caminhos o sinal caminha mais lentamente. Outros fatores são a isolação e a diminuição do *Crosstalk*, a isolação pode ser maior devido ao reduzido número de condutores, o mesmo ocorre com o *Crosstalk*, interferências entre os condutores, que chega a ser desprezível (ZELENOVSKY; MENDONÇA, 2006)

Algumas configurações são requeridas para a comunicação serial como: a velocidade, o número de bits por *caracter*, paridade e número de *stop bits* por caracter.

A velocidade da comunicação é geralmente dada em bauds, símbolos por segundo e que particularmente neste tipo de comunicação é o mesmo que bits por segundo, lembrando que nem todos os bits são dados, alguns são de controle como o bit de paridade e *stop bit*. Alguns valores comuns de velocidade são: 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600 and 115200 bit/s.

Data bits é a quantidade de *bits* em cada caracter, o número de *data bits* mais utilizado é 8 o que representa ao código ASCII

Paridade é um método de correção de erros na transmissão, quando este método é utilizado na comunicação serial um bit é transmitido a cada transmissão de um caracter, de modo que a quantidade de 1´s em cada caracter, incluíndo o bit de paridade, é sempre par ou ímpar. Se um *byte* é recebido com a paridade errada, então o dado foi corrompido e então resolvido por uma instrução no protocolo usado.

Stop bits são os bits enviados no final da transmissão de um caracter para detectar e o fim e resincronizar o envio dos caracteres.

5.1 Controle do fluxo de dados

Em algumas aplicações é necessário que a transmissão pare e continue, por exemplo em uma impressora, que não possua um sistema de memória interna, é necessário que a transmissão seja interrompida enquanto o sistema executa uma ação e que a transmissão retome quando necessário, este tipo de controle de fluxo é chamado de *handshake*. Algumas aplicações utilizam o *hardware flow control* este tipo de controle utiliza os sinais provenientes do hardware para o controle como os sinais RTS/CTS (*Request to Send / Clear to Send*) ou DTR/DSR (*Data Terminal Ready / Data Set Ready*) do padrão RS-232. Outras aplicações utilizam *software flow control*, este tipo de controle utiliza o software para controlar o fluxo de informações, normalmente se utiliza caracteres especiais como XON/XOFF (ZELENOVSKY; MENDONÇA, 2006).

Virtual serial port é uma emulação de uma porta serial padrão. Esta porta é criada por software habilitando portas seriais extras em um sistema operacional sem a necessidade de adicionar *hardware*. A virtualização de portas é particularmente em aplicações de *Bluetooth* com módulos de GPS(*Global Position System*), também são amplamente utilizadas na conexão com outras portas seriais por uma rede LAN (*Local Area Network*) e Internet, no caso deste trabalho serão utilizadas portas virtuais para efeito de simulação.

6 VISUAL BASIC E PROGRAMAÇÃO ORIENTADA A OBJETOS

Este capítulo menciona o que é uma programação orientada a objetos e como a linguagem de programação é útil neste trabalho.

Programação orientada a objetos é um tipo de paradigma de programação utilizando objetos, objetos são estruturas de dados constituído de variáveis e funções juntamente com suas interações com outros objetos.

Visual Basic é uma linguagem de programação que utiliza direcionamento por eventos e uma IDE (*Integrated development environment)* da Microsoft para utilizar o modelo de programação COM (*Component* Object Model), usado para criar objetos e ativar processos internos de comunicação entre os objetos. A programação em VB (Visual Basic) foi projetada para um fácil entendimento e uso, devido a isto e à facilidade com que se pode utilizar os recursos do computador, como a porta serial, por meio do Windows, o programa foi desenvolvido nesta linguagem de programação.

O programa em VB neste trabalho seleciona a porta serial conectada com o PIC e também o arquivo contendo o código G, e então transforma as linhas de comando em informações necessárias para que o PIC faça o acionamento os motores de acordo com as coordenadas do código. O próximo capítulo explica como é feito este acionamento dos motores.

7 DRIVER DE ACIONAMENTO DE MOTORES

7.1 Ponte H

A ponte H é um circuito eletrônico que aplica tensão a uma carga em uma dada direção. O termo ponte H é devido à representação gráfica do circuito, consistindo de quatro chaves conforme a figura 20, abrindo as chaves A e D a tensão de alimentação é aplicada na carga, abrindo as chaves B e C a tensão aplicada é reversa. Em um motor DC a inversão da tensão inverte o movimento do motor, em um motor de passo.

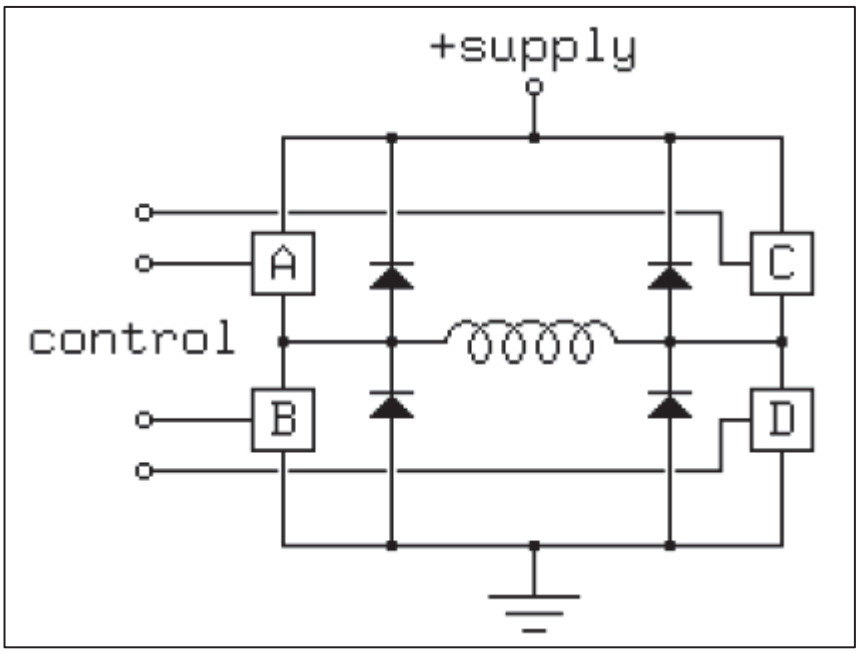


Figura 20 - Ponte H (JONES, 2004)

Apesar da ponte H poder ser encontrada em forma de circuitos integrados para elevadas correntes, é necessário utilizar componentes discretos. A seleção dos componentes são basicamente os transistores que atuarão como chaves. Assumindo a utilização de transistores MOSFET (*metal–oxide–semiconductor field-effect transistor*) para este trabalho, será discutido como as saídas do microcontrolador fazem com que os transistores operem como chaves.

Um transistor possui basicamente três regiões de operação: região de corte, região de saturação e região linear. Na região de corte o transistor está operando como uma chave aberta, a corrente que passa pelos terminais S (*Source*) e D(*Dreno*) é praticamente nula. Na região de saturação o transistor opera como uma chave fechada, a resistência entre os terminais S e D é baixa, a resistência é especificada nos *datasheets* como rds(on). A figura 21 a seguir apresenta as curvas de Id x Vgs para os MOSFET´s tipo P (*p-channel enhancement*) e tipo N (*n-channel enhancement*).

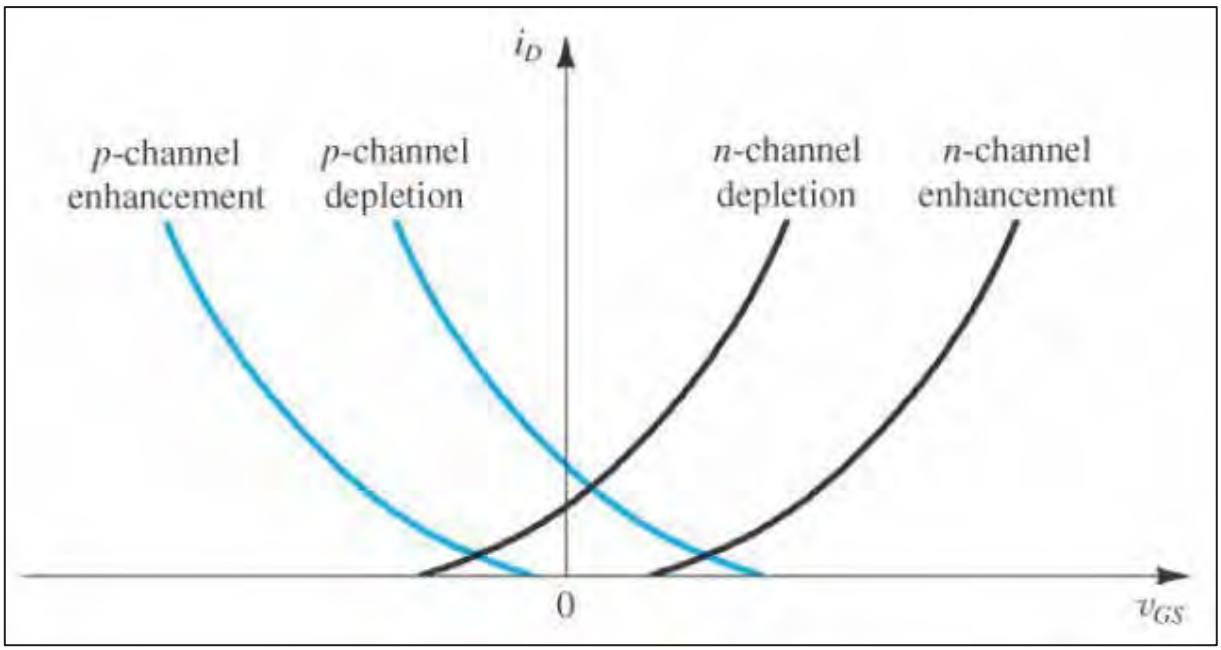


Figura 21 - Curva Id por Vgs(SEDRA; SMITH, 2007)

Como pode ser observado pela figura 21 e de acordo com (SEDRA; SMITH, 2007) o MOSFET tipo N possui um Vth, tensão de *threshold*, maior do que zero, e a condição para a condução (saturação) é Vgs > Vth, conforme a figura 22 a seguir é possível observar a transição da região ativa (*triode*) para a região de saturação. A diferença de operação entre o MOSFET tipo N com o tipo P é que a tensão de *threshold* Vth < 0, e a condição para condução Vgs < Vth.

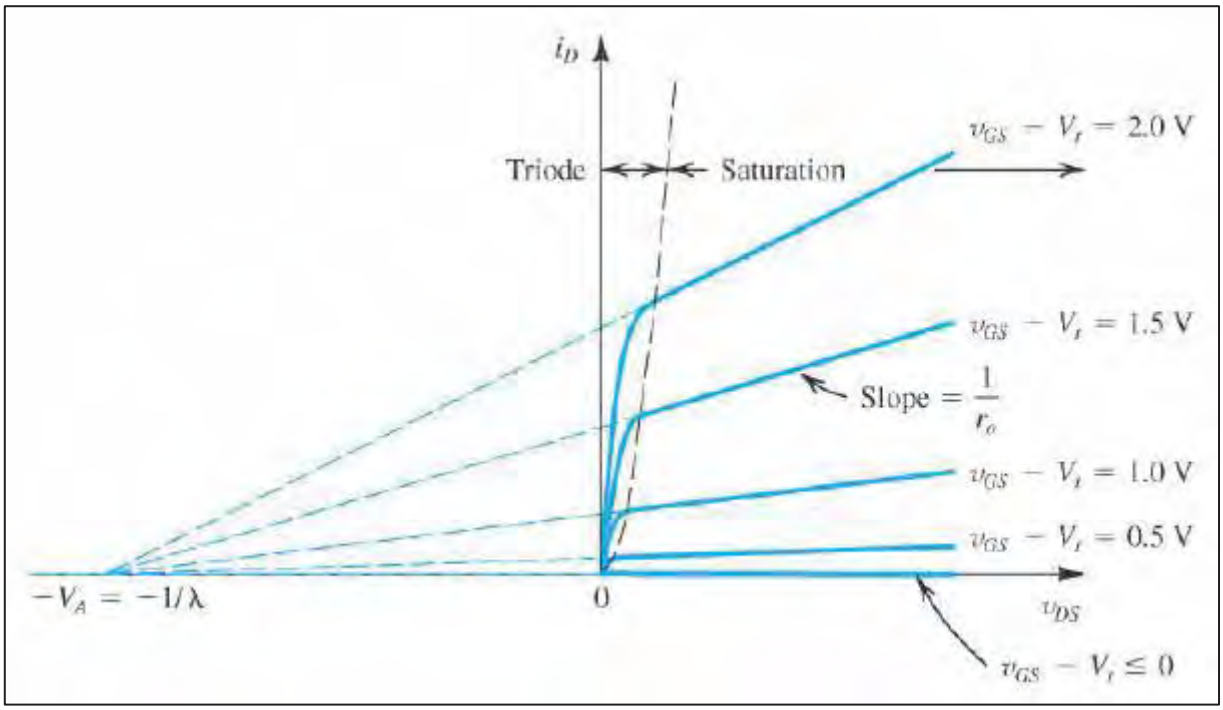


Figura 22 - Curva Id x Vds para diferentes valores de Vgs (SEDRA; SMITH, 2007)

Devido a propriedades construtivas o MOSFET tipo N possui valores menores de rds(on), por isto seria conveniente usar quatro MOSFET´S tipo N, porém uma situação desfavorável acontece com os transistores do lado superior, chaves A e C da figura 20, um MOSFET tipo N no lado superior tem o seu terminal D(*Drain*) conectado à fonte de tensão e o terminal S(*Source*) conectado no enrolamento do motor, quando o terminal G(*Gate*) possui sinal lógico 1 ou +5V ou sinal lógico 0 ou 0V, o transistor está em corte, porque a tensão Vgs será menor ou igual a zero. Portanto o transistor não entrará no estado de saturação apenas alterando os níveis lógicos do microcontrolador, uma vez que o transistor entra no estado de condução a tensão em D, tensão de alimentação, será praticamente igual à tensão em S, apenas a pequena diferença na queda de tensão em rds(on), logo a tensão em G deveria ser maior do que a tensão de alimentação para Vgs > Vth, condição de saturação. Para que a tensão em G seja maior do que a tensão da fonte de alimentação é necessário utilizar um capacitor de *bootstrap* capaz de elevar a tensão, note que se a tensão de alimentação do motor for muito elevada é capaz que o terminal G do transistor não suporte o nível de tensão, assim, é necessário utilizar um capacitor de *bootstrap* A utilização deste capacitor pode gerar problemas com o tempo de carga e descarga, então para aplicações com altas

frequências é recomendável utilizar um MOSFET tipo P, apesar dos MOSFET tipo N possuírem menor rds(on).

7.2 Driver do Motor *Spindle*

O motor *spindle* é responsável pela rotação da fresa que realiza o desbaste do cobre, ele é um motor universal que pode ser alimentado com tensão AC, 127/220V. Para fazer o acionamento deste motor por meio do microcontrolador foi utilizado um opto-isolador com driver para TRIAC, este componente utiliza um LED (Light emissor diode) no circuito de controle. Este LED está acoplado com isolação elétrica a um fototransistor, de modo que se existe corrente passando pelo LED, este fará com que o fototransistor se abra. Por meio do fototransistor o TRIAC irá entrar em regime de operação e será mantido aberto enquanto existir uma corrente mínima no Gate do TRIAC. O TRIAC sairá da região de operação quando o fototransistor entrar em corte, e este entrará em corte quando o circuito do LED estiver em aberto.

A operação destes *drivers* será melhor explicada no próximo capítulo, o qual mostrará todo o desenvolvimento do trabalho proposto.

8 DESENVOLVIMENTO

Neste capítulo o desenvolvimento do trabalho será demonstrado, o tratamento do código G por meio do programa em Visual Basic, o envio dos comandos via serial para o PIC por meio do controle de fluxo por software e o acionamento dos motores pelo PIC.

Primeiramente será exibido o fluxograma da programação, figura 23 a seguir, feita em VB e cada bloco será minusciosamente explicado.

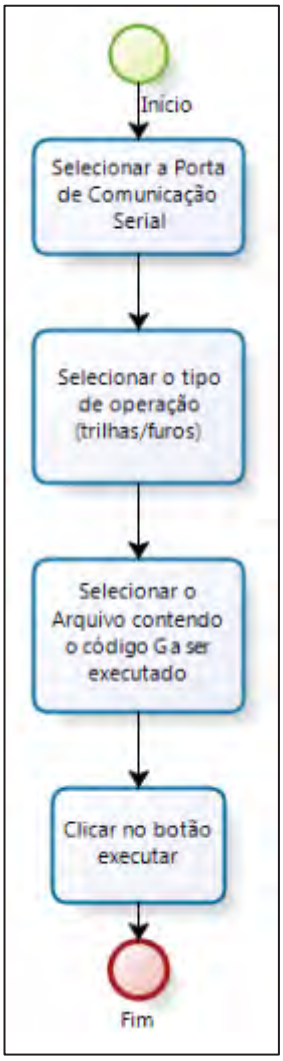


Figura 23 - Fluxo do programa em VB

O programa é bem simples, a seguir a figura 24 mostra a tela principal do programa dando uma visão de como são executadas as atividades do fluxo.

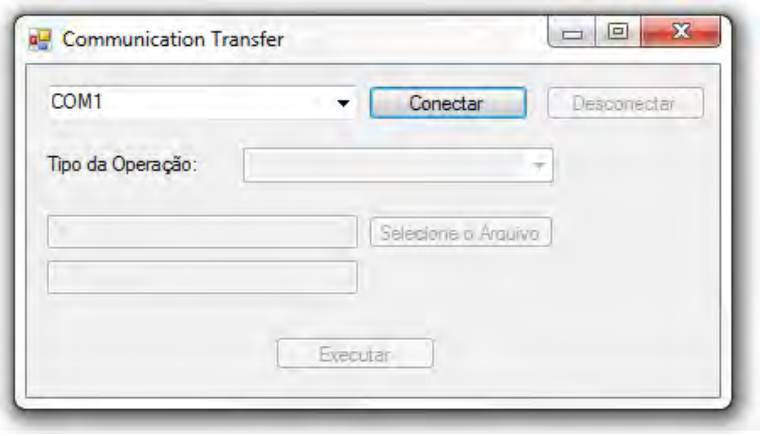


Figura 24 - Tela Inicial do Programa

Como descrito no fluxo a porta de comunicação serial é escolhida e então pressiona-se o botão Conectar. Ao clicar no botão o programa executa um evento que utiliza o valor selecionado no *Combobox* e tenta abrir a porta serial correspondente, se a porta foi aberta com sucesso, ou seja ela está disponível para que programas possam utilizá-la, o programa continua a sua execução habilitando a escolha da operação como na figura 25 a seguir.

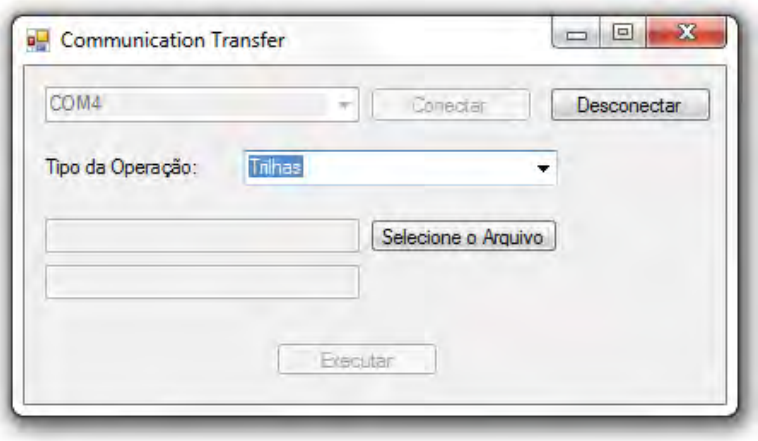


Figura 25 - Seleção da Operação

Após selecionar a operação é necessário carregar o arquivo txt que possui o código G. A figura 26 a seguir mostra a tela do programa após carregar o arquivo. Nota-se que nesta figura o programa mostra o nome do arquivo carregado e também o número de linhas de comandos, no caso o sistema executará 15299 comandos, sendo que a maioria é de movimento. Ao clicar no botão executar o envio das informações pela porta serial terá início, para a comunicação serial é mostrado um novo fluxo na figura 27.

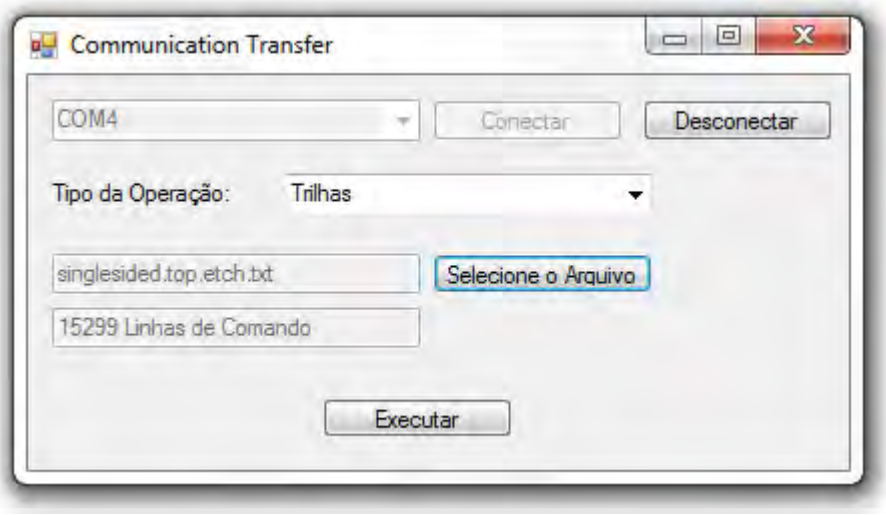


Figura 26 - Programa com o código G carregado

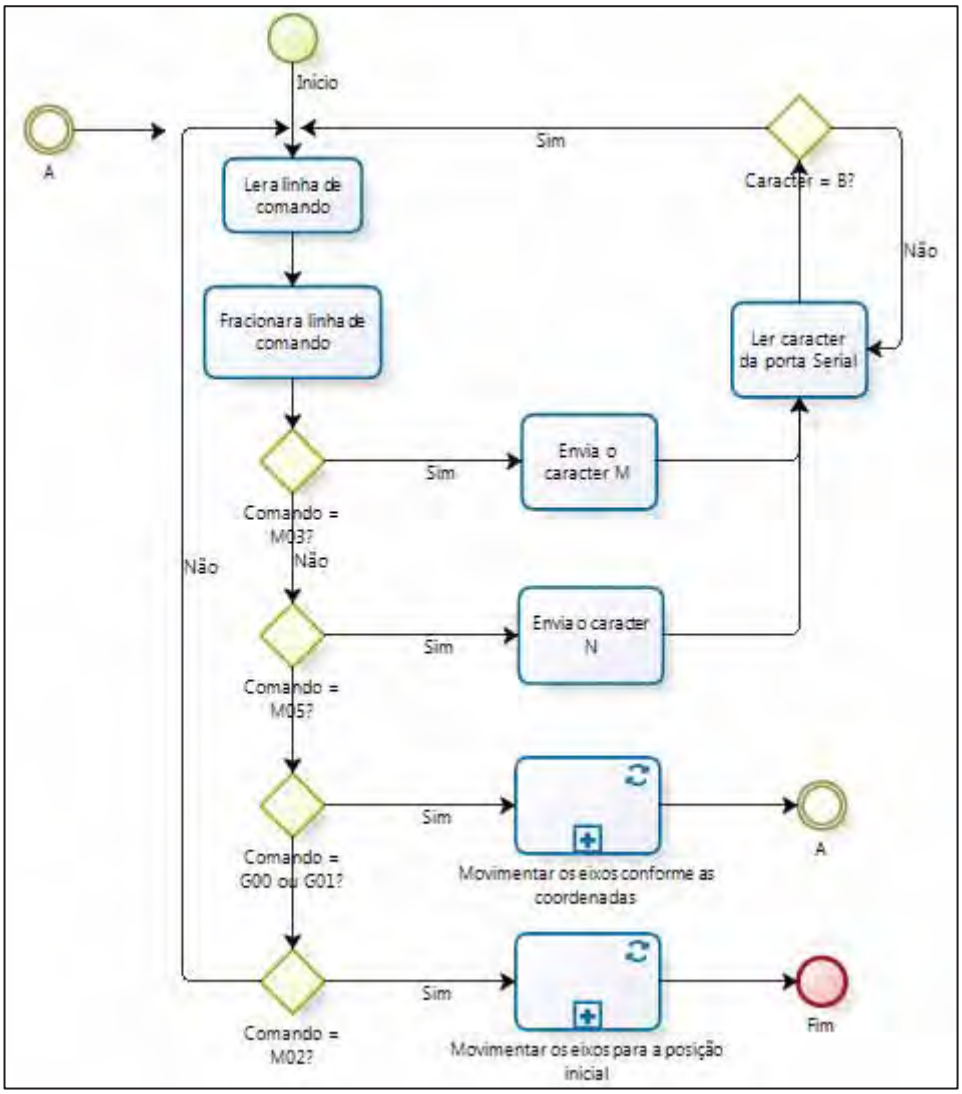


Figura 27 - Fluxo do tratamento dos comandos

O programa lê a linha do comando e a fraciona separando o código das demais variáveis. O comando M03 do código G envia o caracter M que será entendido pelo microcontrolador para ligar o motor spindle, o comando M05 envia o caracter N que desligará o motor Spindle, a próxima linha do código G só será lida e executada após o microcontrolador enviar o caracter B. O envio do caracter B indica que o microcontrolador executou a função corretamente e está pronto para receber um novo comando.

Os comandos G00 ou G01 e o comando M02 possuem um tratamento diferente pois envolvem o controle dos motores de passo, portanto, serão tratados nos fluxos das figuras 28 e 29.

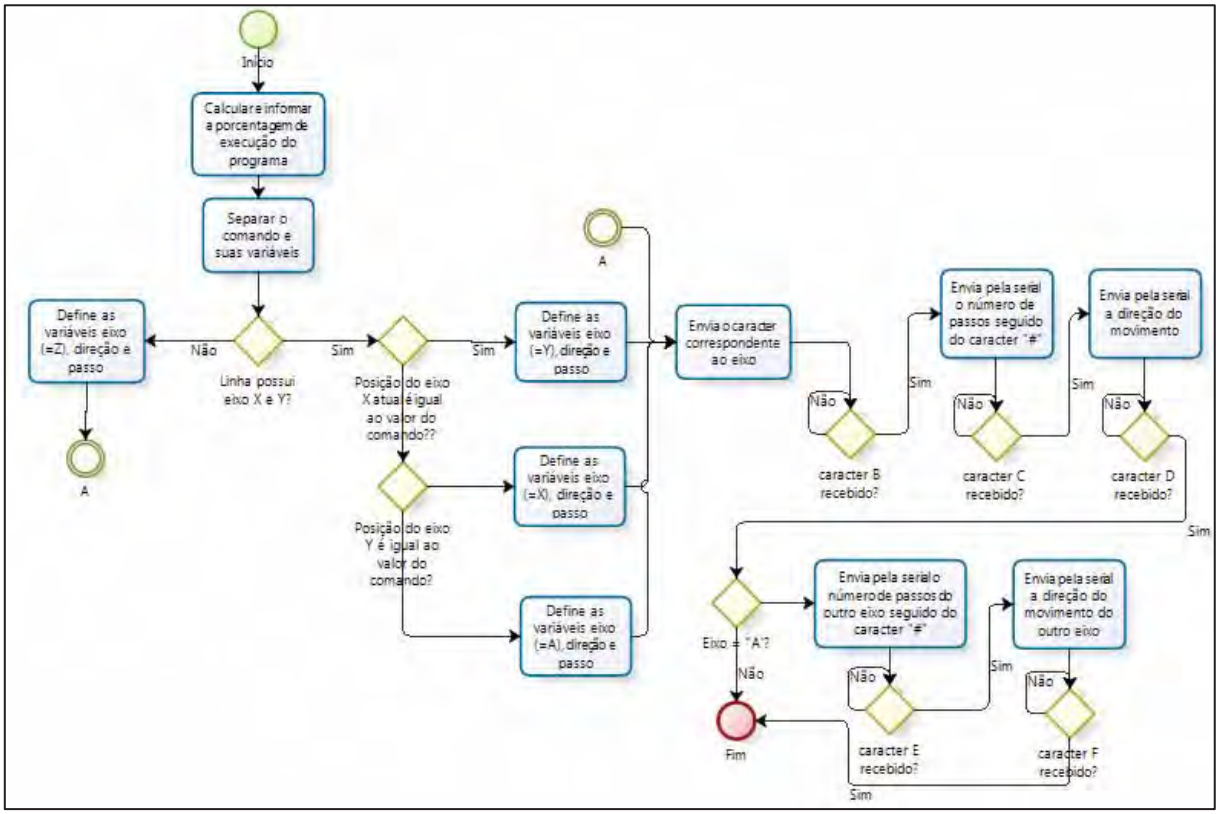


Figura 28 – Subfluxo do tratamento dos comandos G00 e G01

A figura 28 mostra o subfluxo para o envio das informações no caso dos comandos G00 e G01, o programa inicialmente calcula e mostra na tela do computador a porcentagem de execução em que o programa está através da expressão 8.1 a seguir.

$$\%\ execução = \frac{número\ do\ comando\ atual}{número\ total\ de\ comandos} \qquad (8.1)$$

Após o programa separar o comando e suas variáveis, verifica-se os eixos que estão na linha de comando, de acordo com o eixo o programa seguirá um caminho, definido no fluxo. Se a movimentação for em apenas um eixo, o fluxo envia o caracter correspondente ao eixo e espera receber o caracter B, após isto é enviado o número de passos seguido do caracter #

para que o microcontrolador saiba que o envio das informações terminou. O número de passos é calculado apartir do ângulo do passo e também do passo da rosca usada no eixo. No caso o motor de passo possui ângulo de passo 1.8° e a rosca possui um passo de 4mm, portanto utilizando a equação 13.1 a seguir é possível saber quantos milimetros corresponde a um passo.

$$passo(mm) = \frac{passo\ da\ rosca * \hat{a}ngulo\ do\ passo}{360} \quad (8.2)$$

$$passo(mm) = \frac{4 * 1.8}{360} = 0.02mm \quad (8.3)$$

Como as coordenadas são absolutas, é necessário calcular a diferença entre a última coordenada, ambas dadas em mm, e dividir pelo valor do passo em mm para saber quantos passos são necessários até o ponto desejado, é neste ponto que ocorre o arredondamento e através deste define-se a resolução do sistema. Esta diferença pode ser negativa indicando que o movimento deve ser no sentido contrário, para isto o programa analisa se o valor do passo é negativo, se for ele calcula o módulo e define a variável de direção como N, direção oposta.

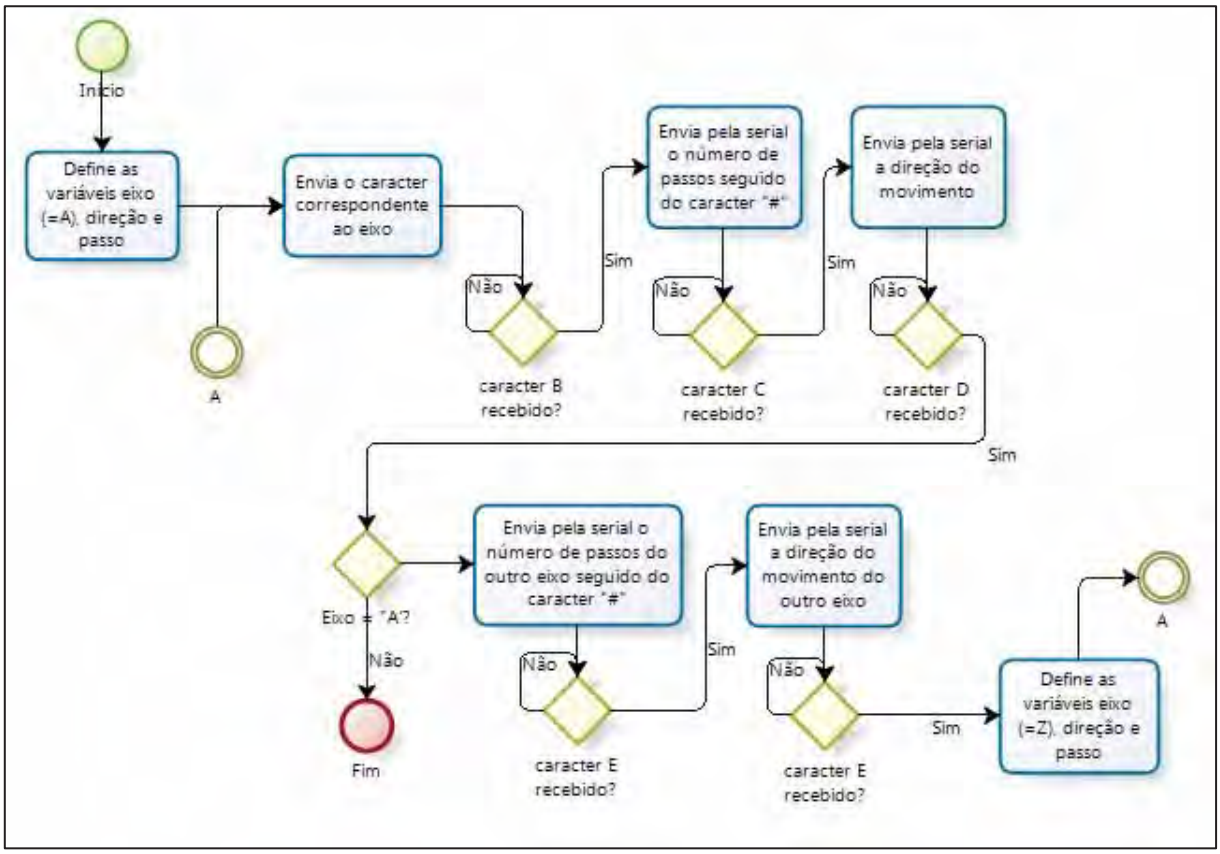


Figura 29 - Subrotina do comando M02

A subrotina do comando M02 serve para indicar que o programa terminou, e portanto é executada no final, neste comando os motores de passo rotacionam de modo a posicionar o sistema na origem, portanto, os comandos G00 X0.0000 Y0.0000 e G00 Z2.5400 são executados.

Todos os comandos e sinais de controle de fluxo enviados pelo programa em VB devem ser entendidos pelo microcontrolador, portanto será discutida a programação em C utilizada pelo microcontrolador, primeiramente será mostrado um fluxo do tratamento das informações recebidas pelo fluxo principal da programação (*main)*, a figura 30 a seguir mostra o fluxo.

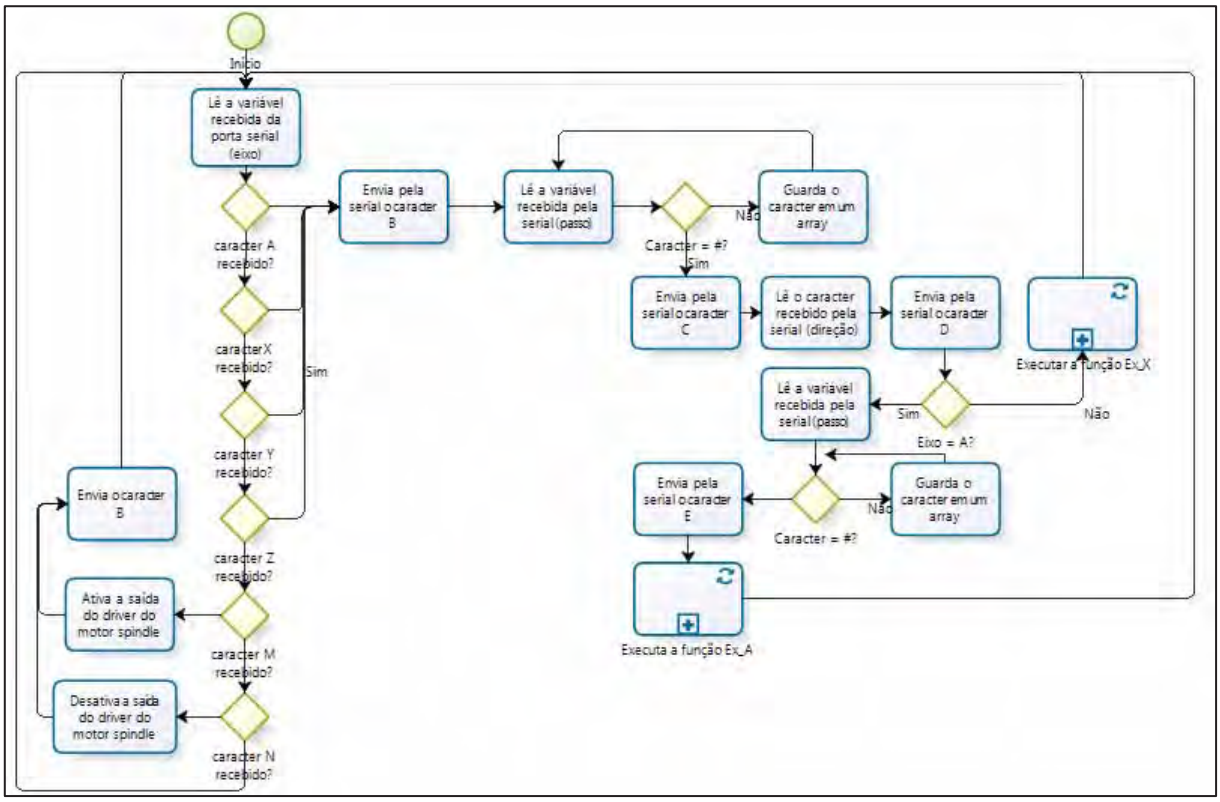


Figura 30 - Fluxo main do PIC

A programação do PIC deve estar de acordo com a programação em VB, uma vez que dados são transferidos de uma para outra pela comunicação serial. Inicialmente o PIC fica fazendo uma varredura na porta serial aguardando o recebimento de um caracter, este caracter é exatamente o caracter do eixo, o caracter A significa que o movimento acontece nos dois eixos, os caracteres X, Y e Z são movimentos nos eixos respectivos, o caracter M coloca a saída correspondente ao driver do motor spindle em nível lógico alto, e o caracter N coloca a saída em nível lógico baixo. As duas funções do fluxo da figura 30, Ex_X e Ex_A executam a movimentação do motor de passo de acordo com as variáveis recebidas. A figura 31 e 32 a seguir mostram o fluxo dessas funções.

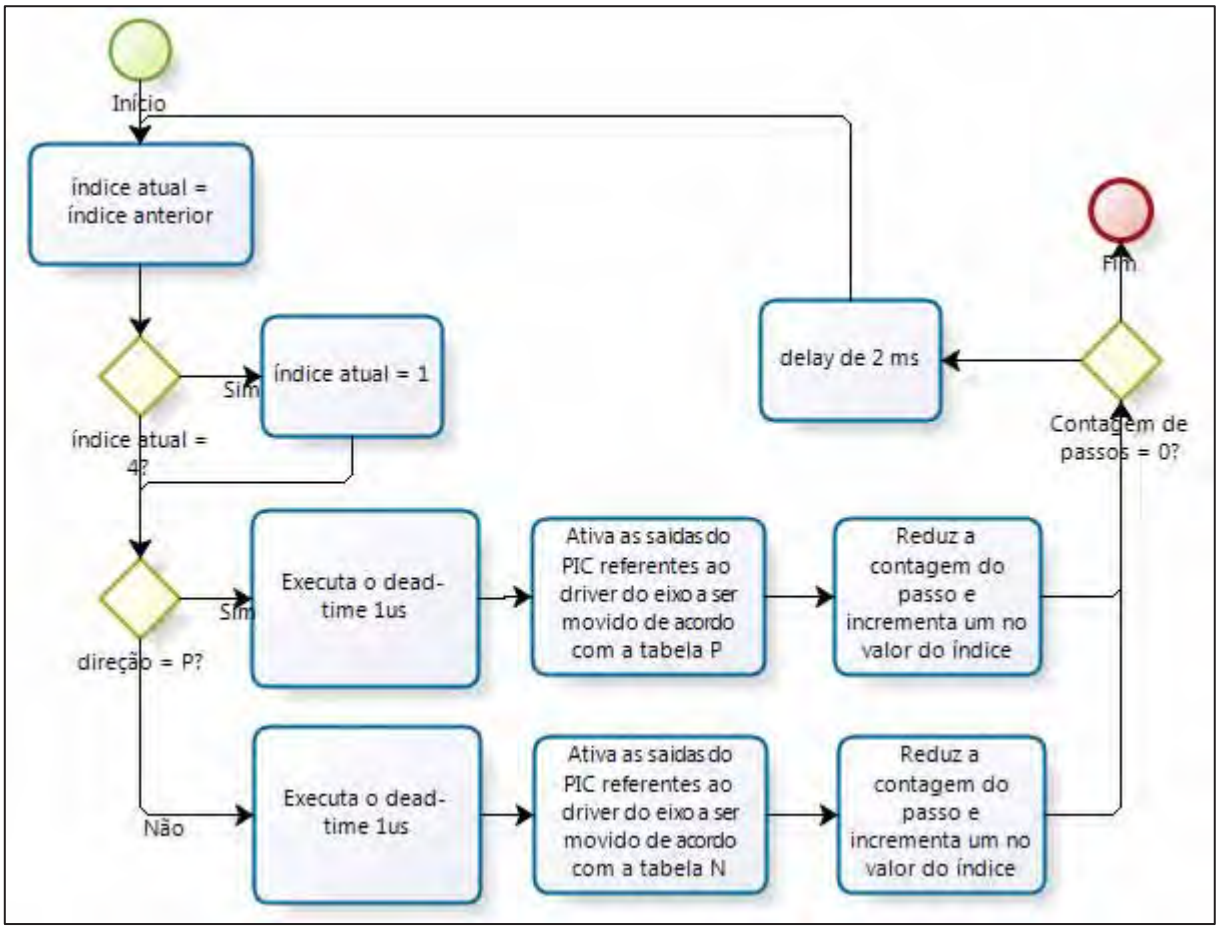


Figura 31 - fluxo da função Ex_x

A função Ex_x utiliza uma tabela com o valor índice indicando em que estado de energização dos enrolamentos o motor dará sequência para o próximo passo, a tabela 8.1 a seguir mostra a sequência P, movimento do motor no sentido positivo. Para movimentos no sentido negativo a sequência a ser seguida é a da tabela N, mostrada na tabela 8.2 a seguir. No caso do eixo Z duas tabelas são usadas simultaneamente, por causa da saída do driver utilizar duas portas do microcontrolador, porta A e E. As tabela 8.3 mostra a sequência positiva do eixo Z e a tabela 8.4 mostra a sequência negativa do eixo Z.

Tabela 8.1 – Sequência Positiva

Índice da sequência	Sequência Binária do bit mais significativo ao menos significativo
1	10011001
2	10010110
3	01100110
4	01101001

Tabela 8.2 - Sequência Negativa

Índice da sequência	Sequência Binária do bit mais significativo ao menos significativo
1	01101001
2	01100110
3	10010110
4	10011001

Tabela 8.3 - Sequência Positiva do eixo Z

Índice da sequência	Sequência Binária do portA, os 6 bits menos significativos	Sequência Binária do portE, os 2 bits mais significativos
1	011001	010
2	010110	010
3	100110	001
4	101001	001

Tabela 8.4 - Sequência Negativa do eixo Z

Índice da sequência	Sequência Binária do portA, os 6 bits menos significativos	Sequência Binária do portE, os 2 bits mais significativos
1	101001	001
2	100110	001
3	010110	010
4	011001	010

Portanto o valor das tabelas serão os valores da saída do microcontrolador responsáveis pelo acionamento do driver, mais especificamente dos transistores, mais adiante será possível compreender melhor a relação entre as saídas do microcontrolador com o driver a ser acionado. Então, a função utiliza a tabela e os índices para acionarem o motor de passo, antes de cada sequência a ser acionada o microcontrolador executa o *dead-time*, que realiza o desligamento dos enrolamentos energizados e espera 1µ, tempo suficiente para que os transistores mudem de estado. Os valores de td(*on*) (*Turn-on delay time*), tr(*Rise time*), td(*off*) (*Turn-off delay time*) e tf(*Fall time*) estão na faixa de 50ns, assegurando que o tempo é suficiente. O *dead-time* é necessário para que não ocorra um pico de corrente ocasionado pela condução simultânea dos transistores do lado de alta e de baixa, o *dead-time* asseguda que, ou o transistor do lado de alta, ou o do lado de baixa esteja cortado.

Após o passo ser realizado, o valor do índice da sequência é incrementado e é decrescido de 1 o valor da contagem de passos, a posição do último índice utilizado é salvo para que em futuros passos neste eixo o microcontrolador utilize como referência este ultimo índice para que não haja quebra na sequência. E então, verifica-se se a contagem dos passos chegou a zero, senão o microcontrolador executa um delay de 2ms e executa o próximo passo. O delay de 2ms é utilizado para uma velocidade de 500 passos por segundo, como na equação 8.4 a seguir.

$$velocidade = \frac{passos}{segundo} = \frac{1}{2 * 10^{-3}} = 500\ \frac{passos}{segundo} \qquad (8.4)$$

Como dito anteriormente a figura 32 a seguir mostra o fluxo da função ex_A:

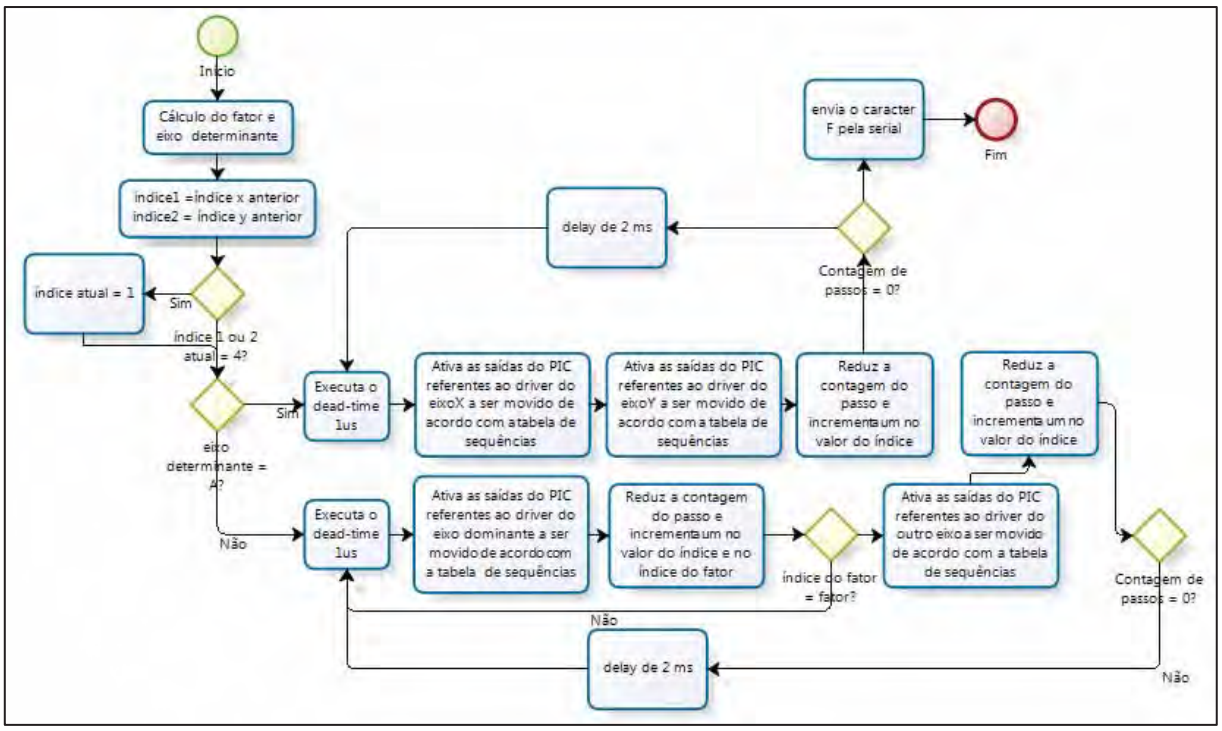


Figura 32 - fluxo da função ex_A

A função ex_A executa o movimento do passo quando os eixos X e Y precisam ser locomovidos simultaneamente, neste caso será necessário calcular o fator e o eixo determinante. O eixo determinante é o eixo que possui a maior quantidade de passos, e o fator é o número de passos que o eixo determinante precisa executar antes de um passo no outro eixo, o fator é calculado utilizando a quantidade de passos do eixo dominante divido pela quantidade de passos do outro eixo, isto é feito porque duas coordenadas representam uma reta que nem sempre é vertical, horizontal, possui 45° ou 135°. Portanto, utiliza-se o fator e um índice do fator indicando quantos passos já foram dados no eixo dominante e se é a vez de executar um passo no outro eixo.

Neste ponto do desenvolvimento foi mostrado como é feita a obtenção do código G, como ele é tratado e transmitido pela serial, e como o recebimento da informação é transformado em saídas do microcontrolador, porém estas saídas não foram claramente definidas, portanto agora serão definidas as saídas do microcontrolador e o seu funcionamento juntamente com os drivers dos motores de passo e *spindle*.

O anexo D mostra como o PIC está ligado ao driver do motor de passo que movimenta o eixo X, utilizando todas as saídas da *portB* do PIC, cada saída está ligada a um transistor da ponte H. Como o motor de passo bipolar possui dois enrolamentos então são necessários 4 transistores por enrolamento, portanto no total são necessários 8 transistores, totalizando as 8 saídas da *portB*. A figura 33 a seguir mostra parte da ponte H para melhor entendimento do circuito.

O driver da ponte H é composta do lado de alta, *high side*, e do lado de baixa, *low side*. O lado de alta é composto pelo transistor MOSFET tipo-P IRF9520 e pelo transistor BC547, o lado de baixa é composto pelo transistor MOSFET tipo-N IRF630. As figuras 33 e 34 e 35 a seguir mostram parte do datasheet dos transistores.

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Drain to Source Breakdown Voltage	BV_{DSS}	I_D = 250µA, V_{GS} = 0V (Figure 10)	200	-	-	V
Gate Threshold Voltage	$V_{GS(TH)}$	V_{GS} = V_{DS}, I_D = 250µA	2	-	4	V
Zero Gate Voltage Drain Current	I_{DSS}	V_{DS} = Rated BV_{DSS}, V_{GS} = 0V	-	-	25	µA
		V_{DS} = 0.8 x Rated BV_{DSS}, V_{GS} = 0V, T_J = 125°C	-	-	250	µA
On-State Drain Current (Note 2)	$I_{D(ON)}$	V_{DS} > $I_{D(ON)}$ x $r_{DS(ON)MAX}$, V_{GS} = 10V	9	-	-	A
Gate to Source Leakage Current	I_{GSS}	V_{GS} = ±20V	-	-	±100	nA
Drain to Source On Resistance (Note 2)	$r_{DS(ON)}$	I_D = 5A, V_{GS} = 10V (Figure 8, 9)	-	0.25	0.4	Ω
Forward Transconductance (Note 2)	g_{fs}	V_{DS} > $I_{D(ON)}$ x $r_{DS(ON)MAX}$, I_D = 5A (Figure 12)	3	4.8	-	S
Turn-On Delay Time	$t_{d(ON)}$	V_{DD} = 90V, I_D = 9A, R_{GS} = 9.1Ω, V_{GS} = 10V R_L = 9.6Ω MOSFET Switching Times are Essentially Independent of Operating Temperature	-	-	30	ns
Rise Time	t_r		-	-	50	ns
Turn-Off Delay Time	$t_{d(OFF)}$		-	-	50	ns
Fall Time	t_f		-	-	40	ns

Figura 33 - IRF 630 datasheet

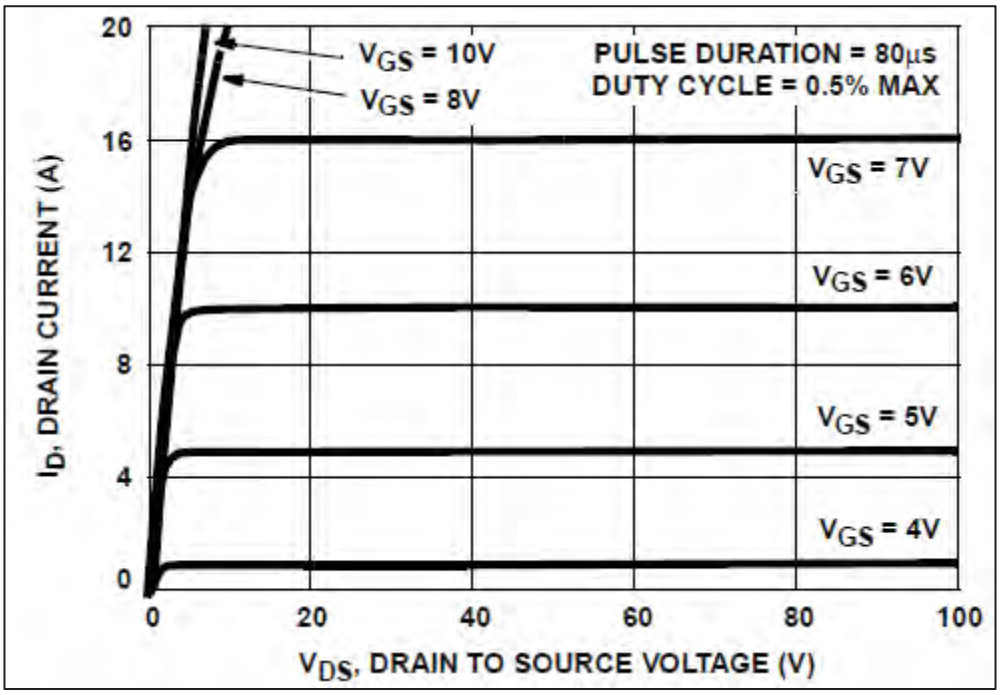


Figura 34 - IRF 630 gráfico Id x Vds em função de Vgs

A figura 34 mostra o valor de Vgs(th) e rds(on), importantes para entender como o driver funciona. Como visto no capítulo 7 o valor de Vgs é o valor apartir do qual o transistor passa a conduzir, na figura 35 é possível notar os valores de Vgs necessários para que o transistor passe para o estado de saturação. O valor típido de rds(on) é 0.25Ω e o máximo 0.4Ω, supondo que o valor de rds(on) é 0.4Ω uma corrente de 5A passando pelo transistor significa uma queda de tensão Vds de 2V, esta queda de corrente impede que o motor de passo utilize o seu máximo desempenho, tal queda pode ser recompensada pelo aumento da tensão na fonte de alimentação, mas muitas vezes o valor que a fonte de tensão deve possuir é inviável, sendo necessários resistores, que dissipam potência e diminuem o rendimento do sistema, ou circuitos adicionais que aumentam a complexidade do sistema. Uma solução interessante é utilizar um sistema chopper que utiliza uma tensão elevada capaz de atingir a corrente máxima em um tempo reduzido compensando as quedas de tensão nos transistores. Como citado anteriormente este sistema é difícil de ser dimensionado justamente por causa das quedas e da má documentação dos motores de passo.

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Drain to Source Breakdown Voltage	BV_{DSS}	I_D = -250µA, V_{GS} = 0V (Figure 10)	-100	-	-	V
Gate Threshold Voltage	$V_{GS(TH)}$	V_{GS} = V_{DS}, I_D = -250µA	-2	-	-4	V
Zero Gate Voltage Drain Current	I_{DSS}	V_{DS} = Rated BV_{DSS}, V_{GS} = 0V	-	-	-25	µA
		V_{DS} = 0.8 x Rated BV_{DSS}, V_{GS} = 0V T_C = 125°C	-	-	-250	µA
On-State Drain Current (Note 2)	$I_{D(ON)}$	V_{DS} > $I_{D(ON)}$ x $r_{DS(ON)\ MAX}$, V_{GS} = -10V	-6	-	-	A
Gate to Source Leakage Current	I_{GSS}	V_{GS} = ±20V	-	-	±100	nA
Drain to Source On Resistance (Note 2)	$r_{DS(ON)}$	I_D = -3.5A, V_{GS} = -10V (Figures 8, 9)	-	0.500	0.600	Ω
Forward Transconductance (Note 2)	g_{fs}	V_{DS} > $I_{D(ON)}$ x $r_{DS(ON)MAX}$, I_D = -3.5A (Figure 12)	0.9	2	-	S
Turn-On Delay Time	$t_{d(ON)}$	V_{DD} = 0.5 x Rated BV_{DSS}, I_D ≈ -6.0A, R_G = 50Ω , R_L = 7.7Ω€ for V_{DSS} = 50Ω MOSFET Switching Times are Essentially Independent of Operating Temperature	-	25	50	ns
Rise Time	t_r		-	50	100	ns
Turn-Off Delay Time	$t_{d(OFF)}$		-	50	100	ns
Fall Time	t_f		-	50	100	ns

Figura 35 - Datasheet IRF 9520

Igualmente ao transistor IRF630 o transistor IRF9520 tipo P possui rds(on) máximo de 0.6Ω, os transistores do tipo P apresentam valores mais elevados de rds(on) apresentando também maior queda de tensão e dissipação de energia, com uma corrente de 5A o transistor apresenta queda de 3A e dissipa 5.4W. Na figura 36 a seguir será mostrado parte do circuito da ponte H.

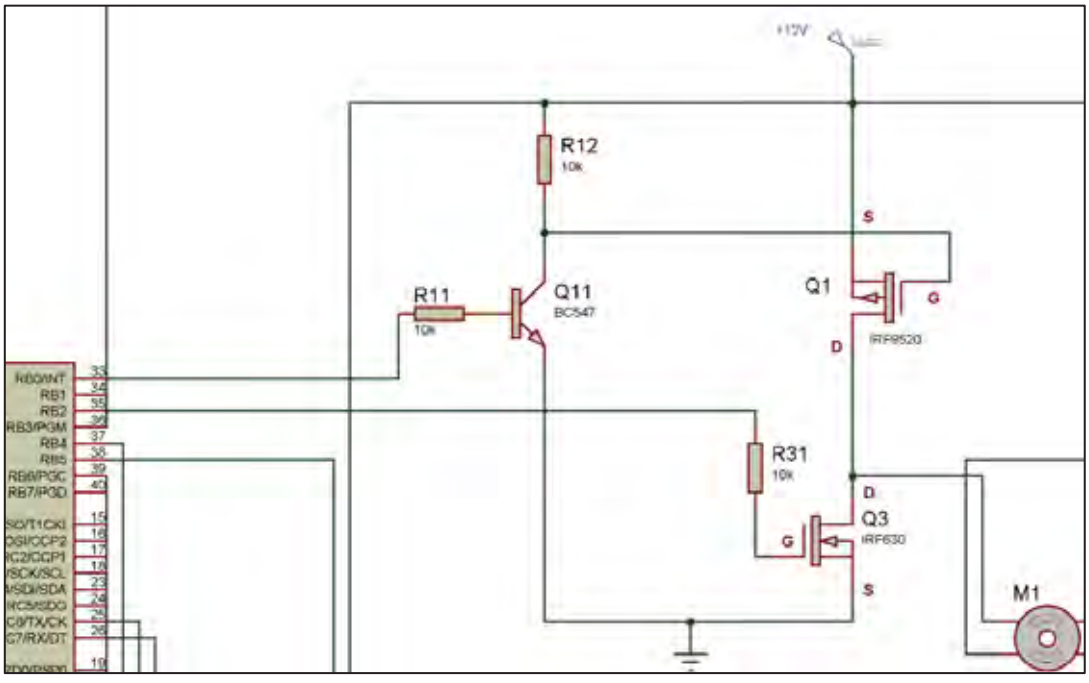


Figura 36 - *High Side* **e** *Low Side* **da ponte H**

Este circuito mostra como o PIC está ligado com os transistores. O lado de alta possui o transistor BC547 que aumenta o nível da tensão do sinal de gate, quando a saída do PIC é 0V o transistor BC547 está cortado fazendo com que a tensão no gate seja 12V, portanto a tensão Vgs é dada pela equação 8.5 a seguir. Quando a saída do PIC é 5V o transistor BC547 está saturado fazendo com que a tensão no gate seja praticamente 0V, com isso a tensão Vgs é dada na equação 8.6 a seguir.

$$V_{GS} = V_G - V_S = 12 - 12 = 0V \quad (8.5)$$

$$V_{GS} = V_G - V_S = 0 - 12 = -12V \quad (8.6)$$

Conforme visto na figura 35 Vgs(th) é de no máximo -4V, deste jeito o transistor IRF9520 está cortado quando o nível lógico de saída é 0 e saturado quando o nível lógico de saída é 1.

Para o lado de baixa da ponte H o transistor utilizado é o IRF630 e o terminal G está diretamente ligado no PIC e o terminal S na referência 0V. Portanto quando o nível lógico da saída é 1 a tensão Vgs é 5V, e quando o nível lógico é 0 a tensão é 0V, conforme a figura 33 o transistor está saturado com a tensão de 5V e cortado com 0V, e da figura 34 e tensão Vgs de 5V a corrente é limitada a um pouco mais de 4A.

Nota-se que não foram mostrados os diodos volantes em paralelo com os transistores, IRF630 e IRF9520, mas eles são obrigatórios sempre que cargas indutivas são utilizadas. Os diodos volantes dissipam a energia remanescente nos enrolamentos para não danificar o transistor.

É mostrado o driver do motor Spindle com o circuito na figura 37.

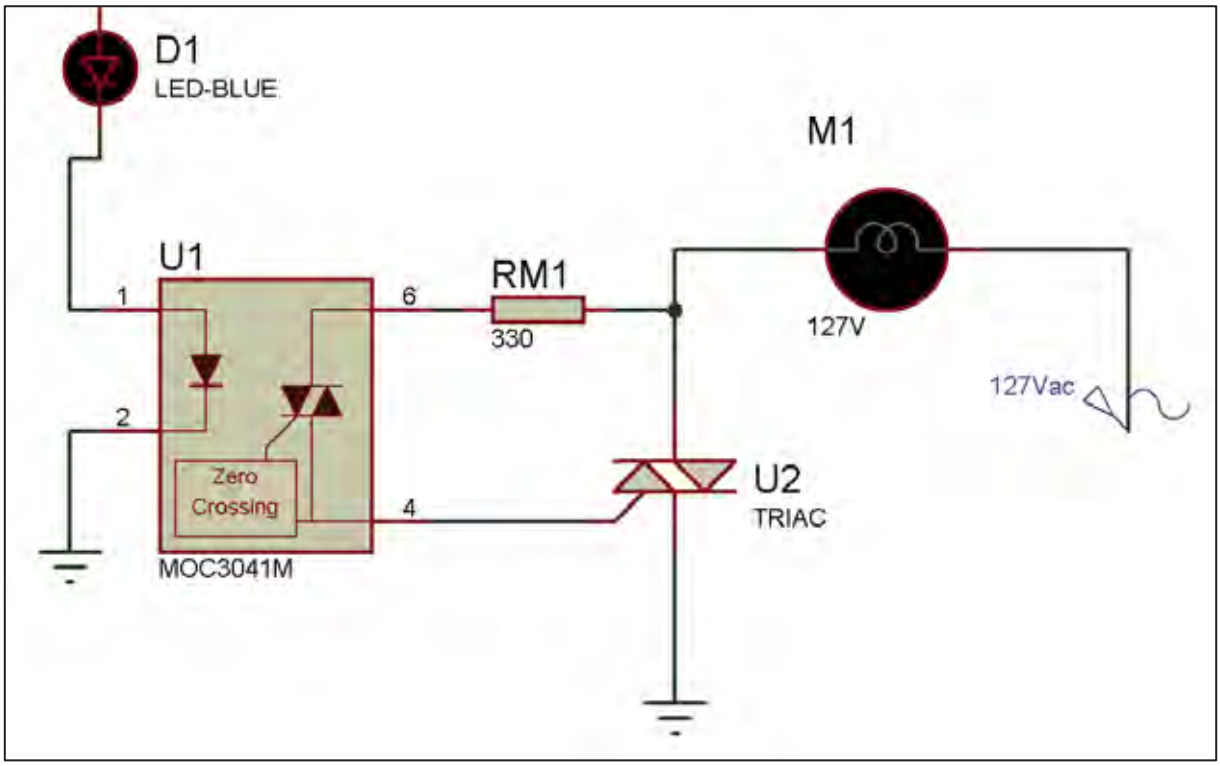


Figura 37 - Driver do *Spindle*

A figura 37 mostra o driver do motor *spindle* (M1) utilizando um TRIAC e um fotoacoplador e driver do TRIAC. O fotoacoplador utilizado é o MOC3041M composto por uma parte emissora e uma parte emissora e outra detectora. O fotoacoplador foi projetado para simplificar o acionamento do TRIAC utilizando fontes AC. O seu funcionamento é simples, a parte emissora composta por um LED é acionada apartir da saída do PIC, o seu acionamento é detectado pelo lado que funciona como uma chave, e portanto, abrindo o circuito fazendo com que a corrente acione o terminal de *gate* do TRIAC e estabelaça a condução por ele. Quando a emissão de luz do LED é interrompida, a saída do PIC muda para nível lógico 0, a corrente de manutenção no *gate* fica praticamente zero, fazendo com que o TRIAC pare de conduzir.

9 CONCLUSÃO

Por meio da execução dos programas em Visual Basic e da programação no microcontrolador em C, juntamente com o auxílio do programa *Hyper Terminal* do *Windows*, foi possível testar a comunicação serial dos programas. Com a utilização do software ISIS – Proteus foi possível testar a atuação do microcontrolador e drivers de acordo com as saídas do microcontrolador e comunicação serial.

A programação, a comunicação serial e a funcionalidade dos drivers foram comprovadas, porém, o *software* de simulação não é capaz de visualizar e reproduzir erros como a perda de passos ou deficiência no torque durante a execução. Para isto o ideal seria construir e montar o sistema para avaliar como o driver atua de acordo com a velocidade dos motores e a potência necessária.

Não foi desenvolvido um driver que utilize um chopper como limitador de corrente para os motores de passo, conforme a quantidade de dados a serem processados pode ser necessário mais do que um microcontrolador para executar as funções de passo e, também, gerar o tempo até a corrente máxima para que o chopper possa atuar mantendo a corrente.

Como o objetivo do trabalho era implementar um sistema capaz de utilizar um código G para a manufatura de placas de circuito impresso, não foi possível criar um sistema otimizado, portanto, todas as partes de seu projeto podem ser melhorados. Alguns exemplos são:

- O código G gerado utiliza diversos comandos, como a troca de ferramentas, é possível implementar um novo sistema que utilize outros códigos melhorando a utilidade do sistema.
- Implementar um motor de passo com chopper para limitar a corrente e otimizar o motor de passo
- Verificar a possibilidade de criar um circuito de controle de motor individual para cada eixo, podendo assim mover mais de um eixo simultaneamente.
- Otimizar os códigos de comunicação serial e a ao invés de utilizar a porta serial, utilizar a comunicação USB.

REFERÊNCIAS

CADSOFT., Eagle Manual. Disponível em: < http://www.cadsoftusa.com/> Acesso em: nov. 2011.

Condit R.; Jones, D.W.; Microchip Technology Inc., Stepping Motors Fundamentals. 2004. Disponível em < http://www.divms.uiowa.edu/~jones/step/an907a.pdf> Acesso em: dez. 2011.

Deital, H.M.; Deitel, P.J.; Nieto, T.R., Visual basic 6 how to program, New Jersey: Prentice Hall, 1999.

Johnson, J., PCB-GCODE. 2009. Disponível em: <http://pcbgcode.org> Acesso em: ago. 2011.

Jones D.W. Control of Stepping Motors. The university of Iowa Department of Computer Science. 2004. Disponível em <http://www.divms.uiowa.edu/~jones/step/> Acesso em: dez. 2011.

Jordan, Howard E., Energy-efficient electric motors and their applications; New York; Plenum Press; 1994.

R. Boylestad;L. Nashelsky ., Dispositivos Eletrônicos e Teoria de circuitos, Pearson Prentice Hall, 8ª edicão, 2004.

Sedra A.; Smith K., Microeletrônica. 5ª edição. Pearson Prentice Hall, 2007

Yeadon, W. H.; Yeadon A. W., Handbook of small electric motors. 1 ed. NewYork: McGraw-Hill Professional, 2001. seções 5.2.10, 10.8, 10.9 e 10.10.

ZELENOVSKY R.; MENDONÇA A., PC: um Guia Prático de Hardware e Interfaceamento, MZ Editora Ltda, 4ª Edição, 2006

ANEXO A – Programação do microcontrolador em C

```
#include <16F877A.h>
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES LP                       //Low power osc < 200 khz
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES WRT_50%                  //Lower half of Program Memory is Write Protected

#use delay(clock=8000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

byte const posicao_pos[4] = {0b10011001,
                             0b10010110,
                             0b01100110,
                             0b01101001};
```

```
byte const posicao_neg[4] = {0b01101001,
                             0b01100110,
                             0b10010110,
                             0b10011001};

byte const porta_pos[4] = {0b011001,
                           0b010110,
                           0b100110,
                           0b101001};

byte const porta_neg[4] = {0b101001,
                           0b100110,
                           0b010110,
                           0b011001};

byte const porte_pos[4] = {0b010,
                           0b010,
                           0b001,
                           0b001};

byte const porte_neg[4] = {0b001,
                           0b001,
                           0b010,
                           0b010};
```

```
int xpos;
int ypos;
int zpos;

void ex_x(int passo1,char dir1, char eixo)
{
  int step1;
  int indice1;

  step1 = passo1;

  if (eixo =='X')
  {
    indice1 = xpos;
    Do
    {
      If (indice1 == 4)
      {
        indice1 = 0;
      }
      if (dir1 == 'P')
      {
        output_B(0x00);
        delay_us(1);
        output_B(posicao_pos[indice1]);
        indice1++;
```

```
      step1--;
    }
    Else
    {
      output_B(0x00);
      delay_us(1);
      output_B(posicao_neg[indice1]);
      indice1++;
      step1--;
    }
    delay_ms(2);
    putc('P');

  }while (step1!=0);
  xpos = indice1;
}
if (eixo =='Y')
{
  indice1 = ypos;
  Do
  {
    If (indice1 == 4)
    {
      indice1 = 0;
    }
    if (dir1 == 'P')
```

```
    {
      output_D(0x00);
      delay_us(1);
      output_D(posicao_pos[indice1]);
      indice1++;
      step1--;
    }
    Else
    {
      output_D(0x00);
      delay_us(1);
      output_D(posicao_neg[indice1]);
      indice1++;
      step1--;
    }
    delay_ms(2);
    putc('P');
  }while (step1!=0);
  ypos = indice1;
}
if (eixo =='Z')
{
  indice1 = zpos;
  Do
  {
    If (indice1 == 4)
```

```
    {
      indice1 = 0;
    }
    if (dir1 == 'P')
    {
      output_B(0x00);
      delay_us(1);
      output_A(porta_pos[indice1]);
      output_E(porte_pos[indice1]);
      indice1++;
      step1--;
    }
    Else
    {
      output_B(0x00);
      delay_us(1);
      output_A(porta_neg[indice1]);
      output_E(porte_neg[indice1]);
      indice1++;
      step1--;
    }
    delay_ms(2);
    putc('P');
  }while (step1!=0);
  zpos = indice1;
}
```

```
}

void ex_a(int passo1,char dir1, int passo2, char dir2)
{

  int step1;
  int step2;

  int indice1;
  int indice2;
  int fator;
  int fator_count;
  char determinante;

  If (passo1 == passo2)
  {
    determinante ='A';
    fator = 1;
  }
  Else
  {
    If (passo1 > passo2)
    {
      If (passo2 != 0)
      {
        determinante ='X';
```

```
        fator = passo1/passo2;
      }
      Else
      {
        fator = -1;
      }
    }
    Else
    {
      If (passo1 != 0)
      {
        determinante ='Y';
        fator = passo2/passo1;
      }
      Else
      {
        fator = -1;
      }
    }
}

putc (determinante);
indice1 = xpos;
indice2 = ypos;

step1 = passo1;
```

```
step2 = passo2;

If (determinante =='A')
{

  Do
  {
    If (indice1 == 4)
    {
      indice1 = 0;
    }
    If (indice2 == 4)
    {
      indice2 = 0;
    }

    if (dir1 == 'P')
    {
      output_B(0x00);
      delay_us(1);
      output_B(posicao_pos[indice1]);
      indice1++;
      step1--;
    }
    Else
    {
```

```
    output_B(0x00);
    delay_us(1);
    output_B(posicao_neg[indice1]);
    indice1++;
    step1--;
  }

  if (dir2 == 'P')
  {
    output_D(0x00);
    delay_us(1);
    output_D(posicao_pos[indice2]);
    indice2++;
    step2--;
  }
  Else
  {
    output_D(0x00);
    delay_us(1);
    output_D(posicao_neg[indice2]);
    indice2++;
    step2--;
  }
  delay_ms(2);
  putc('P');
}while (step1!=0);
```

```
  xpos = indice1;
  ypos = indice2;
}

If (determinante =='X')
{
  fator_count=0;
  Do
  {
    If (indice1 == 4)
    {
      indice1 = 0;
    }
    If (indice2 == 4)
    {
      indice2 = 0;
    }
    if (dir1 == 'P')
    {
      output_B(0x00);
      delay_us(1);
      output_B(posicao_pos[indice1]);
      indice1++;
      fator_count++;
      step1--;
      putc('P');
```

```
}
Else
{
  output_B(0x00);
  delay_us(1);
  output_B(posicao_neg[indice1]);
  indice1++;
  fator_count++;
  step1--;
}
If (fator_count == fator)
{
  if (dir2 == 'P')
  {
    If (step2 != 0)
    {
      output_D(0x00);
      delay_us(1);
      output_D(posicao_pos[indice2]);
      indice2++;
      fator_count=0;
      step2--;
      putc('L');
    }
  }
  Else
```

```
        {
          If (step2 != 0)
          {
            output_D(0x00);
            delay_us(1);
            output_D(posicao_neg[indice2]);
            indice2++;
            fator_count=0;
            step2--;
          }
        }
        delay_ms(2);
      }
    }while (step1!=0);
    xpos = indice1;
    ypos = indice2;
}

If (determinante =='Y')
{
    fator_count=0;
    Do
    {
      If (indice1 == 4)
      {
        indice1 = 0;
```

```
}
If (indice2 == 4)
{
  indice2 = 0;
}
if (dir2 == 'P')
{
  output_D(0x00);
  delay_us(1);
  output_D(posicao_pos[indice2]);
  indice2++;
  fator_count++;
  step2--;
  putc('L');
}
Else
{
  output_D(0x00);
  delay_us(1);
  output_D(posicao_neg[indice2]);
  indice2++;
  fator_count++;
  step2--;
}
If (fator_count == fator)
{
```

```
if (dir1 == 'P')
{
  If (step1 != 0)
  {
    output_B(0x00);
    delay_us(1);
    output_B(posicao_pos[indice1]);
    indice1++;
    fator_count=0;
    step1--;
    putc('P');
  }
}
Else
{
  If (step1 != 0)
  {
    output_B(0x00);
    delay_us(1);
    output_B(posicao_neg[indice1]);
    indice1++;
    fator_count=0;
    step1--;
  }
}
delay_ms(2);
```

```
        }
      }while (step2!=0);
      xpos = indice1;
      ypos = indice2;
    }
}



void main()
{

  char eixo;     // Define variável caracter
  char c;
  char passo1[10] = {};
  char passo2[10] = {};
  int passos1;
  int passos2;
  int counter;
  int n;
  char dir1;
  char dir2;

  xpos = 0;
  ypos = 0;
  zpos = 0;
```

```
while (true)
{
  eixo = getch();
    switch (eixo)
    {

      case 'A':
        putc ('B');
        counter = -1;
       Do
         {
         c = getch();
         If (c != '#')
           {
           counter ++;
           passo1[counter] = c;
           }
         }
        While (c != '#');
        putc ('C');
        c = getch();
        dir1 = c;
        putc ('D');
        counter = -1;
        Do
```

```
    {
    c = getch();
    If (c != '#')
      {
      counter ++;
      passo2[counter] = c;
      }
    }
  While (c != '#');
  putc ('E');
  c = getch();
  dir2 = c;
  passos1 = atoi(passo1);
  passos2 = atoi(passo2);
  ex_a(passos1,dir1,passos2,dir2);
  putc ('F');
  for(n = 0; n <= 11; n++)
  {
    passo1[n] = 0;
    passo2[n] = 0;
  }
  break;

case 'X':
  putc ('B');
  counter = -1;
```

```
  Do
    {
    c = getch();
    If (c != '#')
      {
      counter ++;
      passo1[counter] = c;
      }
    }
  While (c != '#');
  passos1 = atoi(passo1);
  putc ('C');
  c = getch();
  dir1 = c;
  ex_x(passos1,dir1,eixo);
  putc ('D');
  for(n = 0; n <= 11; n++)
  {
    passo1[n] = 0;
  }
  break;

case 'Y':
  putc ('B');
  counter = -1;
  Do
```

```
    {
    c = getch();
    If (c != '#')
      {
      counter ++;
      passo1[counter] = c;
      }

    }
  While (c != '#');
  passos1 = atoi(passo1);
  putc ('C');
  c = getch();
  dir1 = c;
  ex_x(passos1,dir1,eixo);
  putc ('D');
  for(n = 0; n <= 11; n++)
  {
    passo1[n] = 0;
  }
  break;

case 'Z':
  putc ('B');
  counter = -1;
  Do
```

```
    {
    c = getch();
    If (c != '#')
      {
      counter ++;
      passo1[counter] = c;
      }
    }
  While (c != '#');
  passos1 = atoi(passo1);
  putc ('C');
  c = getch();
  dir1 = c;
  ex_x(passos1,dir1,eixo);
  putc ('D');
  for(n = 0; n <= 11; n++)
  {
    passo1[n] = 0;
  }
  break;

case 'M':
  output_high(PIN_C0);
  putc ('B');
  break;
```

```
        case 'N':
          output_low(PIN_C0);
          putc ('B');
          break;
        default:
      }
  }
}
```

ANEXO B - **Programa Form1 em Visual Basic**

```
Public Class Form1
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        If SerialPort1.IsOpen = True Then
            SerialPort1.Close()
            Conectar.Enabled = True
            ComboBox1.Enabled = True
            Button4.Enabled = False
            Button2.Enabled = False
            Button6.Enabled = False
            ComboBox2.Enabled = False
            ComboBox2.Text = ""
            TextBox2.Text = ""
            TextBox1.Text = ""
        End If
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Conectar.Click
        SerialPort1.PortName = ComboBox1.Text
        Try
            SerialPort1.Open()
            Button2.Enabled = True
            ComboBox2.Enabled = True
            Conectar.Enabled = False
            ComboBox1.Enabled = False
        Catch ex As Exception
            MsgBox(Err.Description)
        End Try
    End Sub

    Private Sub SerialPort1_DataReceived(ByVal sender As Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived

    End Sub

    Private Sub Button3_Click(ByVal sender As Object, ByVal e As System.EventArgs)
        If SerialPort1.IsOpen = True Then
            SerialPort1.Write(TextBox2.Text)
        Else
            MsgBox("Serial Port is Closed")
        End If
    End Sub
```

```
    Private Sub Button4_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Button4.Click
        Dim openFileDialog1 As New OpenFileDialog()
        Dim FileString() As String
        Dim CCount As Integer

        openFileDialog1.InitialDirectory = "c:\"
        openFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"
        openFileDialog1.FilterIndex = 1
        openFileDialog1.RestoreDirectory = True

        CCount = 0
        If openFileDialog1.ShowDialog() = DialogResult.OK Then
            TextBox1.Text = openFileDialog1.SafeFileName()
            Label2.Text = openFileDialog1.FileName

            FileString = IO.File.ReadAllLines(openFileDialog1.FileName)
            For Each line In FileString
                If Mid(line, 1, 3) = "G00" Or Mid(line, 1, 3) = "G01" Then
                    'TextBox2.Text = openFileDialog1.FileName
                    CCount = CCount + 1
                End If
            Next
            TextBox2.Text = CCount & " Linhas de Comando"
            Label3.Text = CCount
            If CCount = 0 Then
                Button6.Enabled = False
            Else
                Button6.Enabled = True
            End If
        End If
    End Sub
    Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        If SerialPort1.IsOpen = True Then
            SerialPort1.Write(TextBox5.Text)
        Else
            MsgBox("Serial Port is Closed")
        End If
    End Sub
    Private Sub ComboBox2_TextChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ComboBox2.TextChanged
        If ComboBox2.Text = "Trilhas" Or ComboBox2.Text = "Furos" Then
            Button4.Enabled = True
        Else
            Button4.Enabled = False
            TextBox2.Text = ""
```

```
        TextBox1.Text = ""
        Button6.Enabled = False
    End If
End Sub

Private Sub Button6_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Button6.Click
    Dim Resultado
    Dim FileString() As String
    Dim FileName As String
    Dim porc As Double
    Dim contador As Integer

    Dim ssplit1()
    Dim ssplit2()
    Dim eixo
    Dim passos1
    Dim passos2
    Dim aux1
    Dim aux2
    Dim lastX
    Dim lastY
    Dim lastZ
    Dim recv
    Dim dir1
    Dim dir2

    ReDim ssplit2(20)

    lastX = "X0.000"
    lastY = "Y0.000"
    lastZ = "Z0.000"
    contador = 0

    porc = CInt(Label3.Text) / 100

    Resultado = MsgBox("Deseja executar os comandos do arquivo: " & TextBox1.Text,
MsgBoxStyle.YesNo, "Executar")
    If Resultado = vbYes Then
        FileName = Label2.Text
        Form2.Show()
        Me.Hide()
        FileString = IO.File.ReadAllLines(FileName)
        For Each line In FileString

            If Mid(line, 1, 3) = "M03" Then
                SerialPort1.Write("M")
```

```
            Do
               recv = SerialPort1.ReadChar()
               recv = Convert.ToChar(recv)
               If recv = "B" Then
                  Exit Do
               End If
            Loop While (1)
         End If
         If Mid(line, 1, 3) = "M05" Then
            SerialPort1.Write("N")
            Do
               recv = SerialPort1.ReadChar()
               recv = Convert.ToChar(recv)
               If recv = "B" Then
                  Exit Do
               End If
            Loop While (1)
         End If
         If Mid(line, 1, 3) = "M02" Then
            ssplit1 = Split("G00 X0.0000 Y0.0000")
            eixo = "A"
            aux1 = Replace(ssplit1(1), ".", ",")
            aux2 = Replace(lastX, ".", ",")
            passos1 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
            lastX = aux1
            aux1 = Replace(ssplit1(2), ".", ",")
            aux2 = Replace(lastY, ".", ",")
            passos2 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
            lastY = aux1
            ssplit2 = ssplit1
            ''''''''''''''
            If passos1 < 0 Then
               System.Math.Abs(passos1)
               dir1 = "N"
            Else
               dir1 = "P"
            End If
            If passos2 < 0 Then
               System.Math.Abs(passos2)
               dir2 = "N"
            Else
               dir2 = "P"
            End If

            SerialPort1.Write(eixo)
```

```
            Do
                recv = SerialPort1.ReadChar()
                recv = Convert.ToChar(recv)
                If recv = "B" Then
                    SerialPort1.Write(passos1 & "#")
                    Exit Do
                End If
            Loop While (1)

            Do
                recv = SerialPort1.ReadChar()
                recv = Convert.ToChar(recv)
                If recv = "C" Then
                    SerialPort1.Write(dir1)
                    Exit Do
                End If
            Loop While (1)
            Do
                recv = SerialPort1.ReadChar()
                recv = Convert.ToChar(recv)
                If recv = "D" Then
                    SerialPort1.Write(passos2 & "#")
                    Exit Do
                End If
            Loop While (1)
            Do
                recv = SerialPort1.ReadChar()
                recv = Convert.ToChar(recv)
                If recv = "E" Then
                    SerialPort1.Write(dir2)
                    Exit Do
                End If
            Loop While (1)
            Do
                recv = SerialPort1.ReadChar()
                recv = Convert.ToChar(recv)
                If recv = "F" Then
                    Exit Do
                End If
            Loop While (1)
            ''''''''''''
            ssplit1 = Split("G00 Z2.5400")
            eixo = "Z"
            aux1 = Replace(ssplit1(1), ".", ",")
            aux2 = Replace(lastZ, ".", ",")
            passos1 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
```

```
    If passos1 < 0 Then
        System.Math.Abs(passos1)
        dir1 = "N"
    Else
        dir1 = "P"
    End If
    lastZ = aux1
    ssplit2 = ssplit1
    SerialPort1.Write(eixo)
    Do
        recv = SerialPort1.ReadChar()
        recv = Convert.ToChar(recv)
        If recv = "B" Then
            SerialPort1.Write(passos1 & "#")
            Exit Do
        End If
    Loop While (1)

    Do
        recv = SerialPort1.ReadChar()
        recv = Convert.ToChar(recv)
        If recv = "C" Then
            SerialPort1.Write(dir1)
            Exit Do
        End If
    Loop While (1)
    Do
        recv = SerialPort1.ReadChar()
        recv = Convert.ToChar(recv)
        If recv = "D" Then
            Exit Do
        End If
    Loop While (1)
End If


If Mid(line, 1, 3) = "G00" Or Mid(line, 1, 3) = "G01" Then
    contador = contador + 1
    Form2.TextBox1.Text = (contador / porc) & "%"
    ssplit1 = Split(line)
    If contador = 1 Then
        If Mid(ssplit1(1), 1, 1) = "X" And Mid(ssplit1(2), 1, 1) = "Y" Then
            eixo = "A"

            aux1 = Replace(ssplit1(1), ".", ",")
            aux2 = Replace(ssplit1(2), ".", ",")
            passos1 = Decimal.Round(Convert.ToDouble(Mid(aux1, 2)) / 0.02)
```

```
passos2 = Decimal.Round(Convert.ToDouble(Mid(aux2, 2)) / 0.02)
If passos1 < 0 Then
   System.Math.Abs(passos1)
   dir1 = "N"
Else
   dir1 = "P"
End If
If passos2 < 0 Then
   System.Math.Abs(passos2)
   dir2 = "N"
Else
   dir2 = "P"
End If
lastX = aux1
lastY = aux2
ssplit2 = ssplit1

SerialPort1.Write(eixo)
Do
   recv = SerialPort1.ReadChar()
   recv = Convert.ToChar(recv)
   If recv = "B" Then
      SerialPort1.Write(passos1 & "#")
      Exit Do
   End If
Loop While (1)

Do
   recv = SerialPort1.ReadChar()
   recv = Convert.ToChar(recv)
   If recv = "C" Then
      SerialPort1.Write(dir1)
      Exit Do
   End If
Loop While (1)
Do
   recv = SerialPort1.ReadChar()
   recv = Convert.ToChar(recv)
   If recv = "D" Then
      SerialPort1.Write(passos2 & "#")
      Exit Do
   End If
Loop While (1)
Do
   recv = SerialPort1.ReadChar()
   recv = Convert.ToChar(recv)
   If recv = "E" Then
```

```
            SerialPort1.Write(dir2)
            Exit Do
        End If
    Loop While (1)
    Do
        recv = SerialPort1.ReadChar()
        recv = Convert.ToChar(recv)
        If recv = "F" Then
            Exit Do
        End If
    Loop While (1)
Else

    eixo = "Z"
    aux1 = Replace(ssplit1(1), ".", ",")
    passos1 = Decimal.Round(Convert.ToDouble(Mid(aux1, 2)) / 0.02)
    If passos1 < 0 Then
        System.Math.Abs(passos1)
        dir1 = "N"
    Else
        dir1 = "P"
    End If
    lastZ = aux1
    ssplit2 = ssplit1

    SerialPort1.Write(eixo)
    Do
        recv = SerialPort1.ReadChar()
        recv = Convert.ToChar(recv)
        If recv = "B" Then
            SerialPort1.Write(passos1 & "#")
            Exit Do
        End If
    Loop While (1)

    Do
        recv = SerialPort1.ReadChar()
        recv = Convert.ToChar(recv)
        If recv = "C" Then
            SerialPort1.Write(dir1)
            Exit Do
        End If
    Loop While (1)
    Do
        recv = SerialPort1.ReadChar()
        recv = Convert.ToChar(recv)
        If recv = "D" Then
```

```
                    Exit Do
                End If
            Loop While (1)
        End If

    Else
        If Mid(ssplit1(1), 1, 1) = Mid(ssplit2(1), 1, 1) Then
            If ssplit1(1) = ssplit2(1) Then
                eixo = "Y"
                aux1 = Replace(ssplit1(2), ".", ",")
                aux2 = Replace(lastY, ".", ",")
                passos1 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
                If passos1 < 0 Then
                    System.Math.Abs(passos1)
                    dir1 = "N"
                Else
                    dir1 = "P"
                End If
                ssplit2 = ssplit1
                lastY = aux1
                SerialPort1.Write(eixo)
                Do
                    recv = SerialPort1.ReadChar()
                    recv = Convert.ToChar(recv)
                    If recv = "B" Then
                        SerialPort1.Write(passos1 & "#")
                        Exit Do
                    End If
                Loop While (1)

                Do
                    recv = SerialPort1.ReadChar()
                    recv = Convert.ToChar(recv)
                    If recv = "C" Then
                        SerialPort1.Write(dir1)
                        Exit Do
                    End If
                Loop While (1)
                Do
                    recv = SerialPort1.ReadChar()
                    recv = Convert.ToChar(recv)
                    If recv = "D" Then
                        Exit Do
                    End If
                Loop While (1)
            Else
```

```
                If Mid(ssplit1(1), 1, 1) = "X" And Mid(ssplit1(2), 1, 1) = "Y" Then
                    If ssplit1(2) = ssplit2(2) Then
                        eixo = "X"
                        aux1 = Replace(ssplit1(1), ".", ",")
                        aux2 = Replace(lastX, ".", ",")
                        passos1 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
                        If passos1 < 0 Then
                            System.Math.Abs(passos1)
                            dir1 = "N"
                        Else
                            dir1 = "P"
                        End If
                        ssplit2 = ssplit1
                        lastX = aux1
                        SerialPort1.Write(eixo)
                        Do
                            recv = SerialPort1.ReadChar()
                            recv = Convert.ToChar(recv)
                            If recv = "B" Then
                                SerialPort1.Write(passos1 & "#")
                                Exit Do
                            End If
                        Loop While (1)

                        Do
                            recv = SerialPort1.ReadChar()
                            recv = Convert.ToChar(recv)
                            If recv = "C" Then
                                SerialPort1.Write(dir1)
                                Exit Do
                            End If
                        Loop While (1)
                        Do
                            recv = SerialPort1.ReadChar()
                            recv = Convert.ToChar(recv)
                            If recv = "D" Then
                                Exit Do
                            End If
                        Loop While (1)
                    Else
                        eixo = "A"
                        aux1 = Replace(ssplit1(1), ".", ",")
                        aux2 = Replace(lastX, ".", ",")
                        passos1 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
                        lastX = aux1
```

```
                aux1 = Replace(ssplit1(2), ".", ",")
                aux2 = Replace(lastY, ".", ",")
                passos2 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
                lastY = aux1
                ssplit2 = ssplit1
                ''''''''''''''''''
                If passos1 < 0 Then
                   System.Math.Abs(passos1)
                   dir1 = "N"
                Else
                   dir1 = "P"
                End If
                If passos2 < 0 Then
                   System.Math.Abs(passos2)
                   dir2 = "N"
                Else
                   dir2 = "P"
                End If

                SerialPort1.Write(eixo)
                Do
                   recv = SerialPort1.ReadChar()
                   recv = Convert.ToChar(recv)
                   If recv = "B" Then
                      SerialPort1.Write(passos1 & "#")
                      Exit Do
                   End If
                Loop While (1)

                Do
                   recv = SerialPort1.ReadChar()
                   recv = Convert.ToChar(recv)
                   If recv = "C" Then
                      SerialPort1.Write(dir1)
                      Exit Do
                   End If
                Loop While (1)
                Do
                   recv = SerialPort1.ReadChar()
                   recv = Convert.ToChar(recv)
                   If recv = "D" Then
                      SerialPort1.Write(passos2 & "#")
                      Exit Do
                   End If
                Loop While (1)
                Do
```

```
                    recv = SerialPort1.ReadChar()
                    recv = Convert.ToChar(recv)
                    If recv = "E" Then
                        SerialPort1.Write(dir2)
                        Exit Do
                    End If
                Loop While (1)
                Do
                    recv = SerialPort1.ReadChar()
                    recv = Convert.ToChar(recv)
                    If recv = "F" Then
                        Exit Do
                    End If
                Loop While (1)
                ''''''''''
            End If
        Else
            eixo = "Z"
            aux1 = Replace(ssplit1(1), ".", ",")
            aux2 = Replace(lastZ, ".", ",")
            passos1 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
            If passos1 < 0 Then
                System.Math.Abs(passos1)
                dir1 = "N"
            Else
                dir1 = "P"
            End If
            lastZ = aux1
            ssplit2 = ssplit1
            SerialPort1.Write(eixo)
            Do
                recv = SerialPort1.ReadChar()
                recv = Convert.ToChar(recv)
                If recv = "B" Then
                    SerialPort1.Write(passos1 & "#")
                    Exit Do
                End If
            Loop While (1)

            Do
                recv = SerialPort1.ReadChar()
                recv = Convert.ToChar(recv)
                If recv = "C" Then
                    SerialPort1.Write(dir1)
                    Exit Do
                End If
```

```
                    Loop While (1)
                    Do
                        recv = SerialPort1.ReadChar()
                        recv = Convert.ToChar(recv)
                        If recv = "D" Then
                            Exit Do
                        End If
                    Loop While (1)
                End If
            End If
        Else
            If Mid(ssplit1(1), 1, 1) = "X" And Mid(ssplit1(2), 1, 1) = "Y" Then
                eixo = "A"
                aux1 = Replace(ssplit1(1), ".", ",")
                aux2 = Replace(lastX, ".", ",")
                passos1 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
                lastX = aux1
                aux1 = Replace(ssplit1(2), ".", ",")
                aux2 = Replace(lastY, ".", ",")
                passos2 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
                lastY = aux1
                ssplit2 = ssplit1
                ''''''''''''''
                If passos1 < 0 Then
                    System.Math.Abs(passos1)
                    dir1 = "N"
                Else
                    dir1 = "P"
                End If
                If passos2 < 0 Then
                    System.Math.Abs(passos2)
                    dir2 = "N"
                Else
                    dir2 = "P"
                End If

                SerialPort1.Write(eixo)
                Do
                    recv = SerialPort1.ReadChar()
                    recv = Convert.ToChar(recv)
                    If recv = "B" Then
                        SerialPort1.Write(passos1 & "#")
                        Exit Do
                    End If
                Loop While (1)
```

```
                    Do
                        recv = SerialPort1.ReadChar()
                        recv = Convert.ToChar(recv)
                        If recv = "C" Then
                            SerialPort1.Write(dir1)
                            Exit Do
                        End If
                    Loop While (1)
                    Do
                        recv = SerialPort1.ReadChar()
                        recv = Convert.ToChar(recv)
                        If recv = "D" Then
                            SerialPort1.Write(passos2 & "#")
                            Exit Do
                        End If
                    Loop While (1)
                    Do
                        recv = SerialPort1.ReadChar()
                        recv = Convert.ToChar(recv)
                        If recv = "E" Then
                            SerialPort1.Write(dir2)
                            Exit Do
                        End If
                    Loop While (1)
                    Do
                        recv = SerialPort1.ReadChar()
                        recv = Convert.ToChar(recv)
                        If recv = "F" Then
                            Exit Do
                        End If
                    Loop While (1)
                    ''''''''''''
                Else
                    eixo = "Z"
                    aux1 = Replace(ssplit1(1), ".", ",")
                    aux2 = Replace(lastZ, ".", ",")
                    passos1 = Decimal.Round((Convert.ToDouble(Mid(aux1, 2)) -
Convert.ToDouble(Mid(aux2, 2))) / 0.02)
                    If passos1 < 0 Then
                        System.Math.Abs(passos1)
                        dir1 = "N"
                    Else
                        dir1 = "P"
                    End If
                    lastZ = aux1
                    ssplit2 = ssplit1
```

```
                    SerialPort1.Write(eixo)
                    Do
                        recv = SerialPort1.ReadChar()
                        recv = Convert.ToChar(recv)
                        If recv = "B" Then
                            SerialPort1.Write(passos1 & "#")
                            Exit Do
                        End If
                    Loop While (1)

                    Do
                        recv = SerialPort1.ReadChar()
                        recv = Convert.ToChar(recv)
                        If recv = "C" Then
                            SerialPort1.Write(dir1)
                            Exit Do
                        End If
                    Loop While (1)
                    Do
                        recv = SerialPort1.ReadChar()
                        recv = Convert.ToChar(recv)
                        If recv = "D" Then
                            Exit Do
                        End If
                    Loop While (1)
                End If
            End If
        End If
    End If
Next
MsgBox("O trabalho foi concluído")
    End If
  End Sub
End Class
```

Anexo C - Programa Form2 em Visual Basic

```
Public Class Form2

   Private Sub Form2_FormClosing(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
      Try
         Select Case MessageBox.Show("O processo será cancelado e o trabalho perdido,
deseja continuar?", _
           "Aviso", MessageBoxButtons.YesNo, MessageBoxIcon.Question)
            Case MsgBoxResult.Yes
               Form1.Show()
            Case MsgBoxResult.No
               e.Cancel = True
         End Select
      Catch ee As Exception
         Throw ee
      End Try
   End Sub

End Class
```

Anexo D – Circuito do PIC com o driver do eixo X

