UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELÉTRICA
CURSO DE ENGENHARIA ELÉTRICA

PEDRO HENRIQUE SOARES MOREIRA

PROJETO E IMPLEMENTAÇÃO DE FRESA CNC DE BAIXO CUSTO PARA CONFECÇÃO DE TRILHAS DE CIRCUITO IMPRESSO

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2018

PEDRO HENRIQUE SOARES MOREIRA

PROJETO E IMPLEMENTAÇÃO DE FRESA CNC DE BAIXO CUSTO PARA CONFECÇÃO DE TRILHAS DE CIRCUITO IMPRESSO

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso de Engenharia Elétrica do Departamento Acadêmico de Elétrica – DAELE – da Universidade Tecnológica Federal do Paraná – UTFPR, Câmpus Pato Branco, como requisito parcial para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Jean Patric da Costa

PATO BRANCO

2018

TERMO DE APROVAÇÃO

O trabalho de Conclusão de Curso intitulado "**PROJETO E IMPLEMENTAÇÃO DE FRESA CNC DE BAIXO CUSTO PARA CONFECÇÃO DE TRILHAS DE CIRCUITO IMPRESSO",** do aluno **"PEDRO HENRIQUE SOARES MOREIRA"** foi considerado **APROVADO** de acordo com a ata da banca examinadora N° **194** de 2018.

Fizeram parte da banca os professores:

Jean Patric da Costa

Carlos Marcelo de Oliveira Stein

Emerson Giovani Carati

A Ata de Defesa assinada encontra-se na Coordenação do Curso de Engenharia Elétrica

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, que sempre me apoiaram e deram suporte em todas as circunstâncias dessa longa e imprevisível caminhada.

Agradeço aos meus amigos por todos os momentos compartilhados.

Agradeço em especial a minha namorada por tudo e pela constante presença em todos os momentos.

Por fim, agradeço ao meu professor orientador pela oportunidade de propor o próprio tema que vem a agregar em vários aspectos à minha formação.

“Pour ce qui est de l’avenir, il ne s’agit pas de le prévoir, mais de le rendre possible.”

Antoine de Saint-Exupéry

“Em relação ao futuro, não se trata de prevê-lo, mas sim de torná-lo possível.”

Antoine de Saint-Exupéry

RESUMO

MOREIRA, Pedro Henrique Soares. **Projeto e implementação de fresa CNC de baixo custo para confecção de trilhas de circuito impresso**. 2018. 86 f. Monografia (Graduação em Engenharia Elétrica) – Curso de Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Pato Branco, 2018.

Este trabalho de conclusão de curso visa o projeto e desenvolvimento de um sistema de controle numérico computadorizado através de um microcontrolador para implementação em uma máquina fresadora de baixo custo aplicada à confecção de trilhas de circuito impresso. É desenvolvida cada etapa de *software* do processo CNC no microcontrolador, desde à entrada dos dados das trilhas provenientes da etapa CAD/CAM, até à interpretação dos dados e às rotinas de controle do servossistema. Simulações são feitas a fim de validar o sistema desenvolvido. Por fim, a estrutura física da fresa de baixo custo é montada e o sistema CNC desenvolvido é aplicado para a verificação experimental da máquina fresadora.

Palavras-chave: Controle de máquinas e servossistemas. CNC. Microcontrolador. Fresa. Motor de passo.

ABSTRACT

MOREIRA, Pedro Henrique Soares. **Project and implementation of a low-cost CNC milling machine applied to the design of printed circuit boards paths**. 2018. 86 f. Monograph (Degree in Electrical Engineering) – Electrical Engineering course, Federal Technological University of Parana. Pato Branco, 2018.

This work aims the project and development of a numerical control system in a microcontroller for implementation in a low-cost CNC milling machine applied to the design of paths in printed circuit boards. There is developed each software stage of the CNC process in the microcontroller, from the data entry of the paths generated in the CAD/CAM step, going through the interpretation of the data and the creation of the servo system control routine. There are performed simulations in order to evaluate the developed system. At last, the physical structure of the low-cost milling machine is setted up and the developed CNC system is applied for the experimental verification and to validate the CNC milling machine.

Keywords: Servo systems and machine control. CNC. Microcontroller. Milling machine. Stepper motor.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SÍMBOLOS

$e_a(\theta)$	Força contraeletromotriz da fase A
$e_b(\theta)$	Força contraeletromotriz da fase B
i_a	Corrente da fase A
i_b	Corrente da fase B
L_a	Indutância da fase A
L_b	Indutância da fase B
R_a	Resistência da fase A
R_b	Resistência da fase B
v_a	Tensão da fase A
v_b	Tensão da fase B
θ	Ângulo do rotor
K_m	Constante de torque do motor
N_r	Número de dentes num polo do rotor
ω	Velocidade angular
B	Amortecimento rotacional
J	Inércia do rotor
R_m	Relutância de magnetização
T_e	Torque exercido pelo motor
T_d	Torque de detenção do rotor
k	Variável de incrementação do laço *for* para G28
p	Passo da barra de rosca
$fios$	Quantidade de fios da barra de rosca
$Inom$	Corrente nominal ajustada na limitação de corrente
V_{ref}	Tensão do circuito limitador de corrente
R_{sense}	Resistor de limitação de corrente
$fstep$	Frequência de acionamento do motor de passo
vel	Velocidade do motor de passo
n_m	Micro-passos por passo
θ_{step}	Ângulo de passo em *full step*
$periodo$	Período do PWM

$RefX$	Quantidade de passos de retorno à referência do eixo X
$RefY$	Quantidade de passos de retorno à referência do eixo Y
$step$	Quantidade de passos
u	Variável de incrementação do laço *for* para G00 e G01
dx	Distância de movimentação do eixo X
$DX1$	Distância absoluta de movimentação do eixo X
$DX45$	Distância absoluta arredondada de movimentação do eixo X
dy	Distância de movimentação do eixo Y
$DY1$	Distância absoluta de movimentação do eixo Y
$DY45$	Distância absoluta arredondada de movimentação do eixo Y
l	Distância percorrida pela barra de rosca a cada passo do motor
$PosATX$	Posição atual do eixo X
$POSATX$	Posição atual absoluta do eixo X
$PosATY$	Posição atual do eixo Y
$POSATY$	Posição atual absoluta do eixo Y
$counterCWX$	Contador de posição em sentido horário do eixo X
$counterCCWX$	Contador de posição em sentido anti-horário do eixo X
$counterCWY$	Contador de posição em sentido horário do eixo Y
$counterCCWY$	Contador de posição em sentido anti-horário do eixo Y
n	Quantidade de linhas de código G
$Nada$	Variável de indicação
v	Variável de validação das posições dos vetores
EZ	Vetor de entrada do eixo Z
EX	Vetor de entrada do eixo X
EY	Vetor de entrada do eixo Y
P	Vetor do comando G04
$GCODE$	Vetor de entrada dos comandos de código G
A	Constante para interpretação do comando G00
B	Constante para interpretação do comando G01
C	Constante para interpretação do comando G04
D	Constante para interpretação do comando G28

SUMÁRIO

1. INTRODUÇÃO

Desde o início da era industrial, os sistemas e processos fabris vêm evoluindo, tanto em busca de flexibilidade quanto de alta produtividade. Muitas destas evoluções surgiram por conta de necessidade e se tornaram possíveis devido ao avanço da tecnologia. Dentre as principais evoluções no âmbito tecnológico, está a ascensão e o aperfeiçoamento da automatização do ambiente industrial. Grande parte desta automatização diz respeito às máquinas CNC.

Uma máquina CNC é o tipo de máquina que segue os preceitos do comando numérico (CN). O CN foi inicialmente desenvolvido no final da década de 1940 por John T. Parsons em colaboração com o MIT, o Instituto de Tecnologia de Massachusetts a fim de aprimorar a indústria de manufatura no pós-guerra. A principal motivação do estudo foi o limite de precisão que se havia chegado na fabricação de partes de aeronaves por operadores humanos, se tornando inviável a evolução dessa indústria; com isso, buscou-se uma forma de automatização desse processo fabril que, logo em seguida, se estendeu para praticamente todo setor industrial.

O comando numérico é um conceito de controle automático dos movimentos de máquinas pela interpretação direta de instruções codificadas na forma de números e letras, fazendo com que o sistema receba as instruções por meio de entrada própria, compile estas instruções e as transmita em forma de sinais de saída aos componentes da máquina, de modo que esta, sem a intervenção de um operador, realize as operações na sequência programada. (POLI-USP; MARCICANO, 2014).

São inúmeros os tipos e as funções das máquinas CNC, desde as mais simples com poucos eixos e que possuem acionamento em malha aberta até às mais complexas que necessitam de um controle em malha fechada. As máquinas CNC mais comuns no ambiente industrial são o torno mecânico e a fresadora, mas além dessas, pode-se citar as máquinas retificadoras, puncionadeiras, máquinas de eletroerosão a fio, de corte a jato d’água, de corte a laser e diversas outras, como as máquinas de posicionamento preciso. Um sistema CNC é amplamente usado no controle de posição de diferentes máquinas, incluindo robôs, montadoras de chips, manuseadoras de semicondutores bem como em máquinas-ferramenta. (SUH et al, 2008, p. 160).

O que todas essas máquinas têm em comum é um controle baseado num sistema de coordenadas, tendo o tipo de coordenada definido a partir da quantidade de eixos da máquina; no caso da fresa CNC de 3 eixos, o sistema adotado é o cartesiano. Além disso, toda máquina CNC é constituída de uma parte mecânica que se refere à estrutura de suporte da máquina, de uma parte eletroeletrônica e por fim por um sistema computadorizado.

O sistema computadorizado, que se refere à etapa de *software,* é a parte mais importante da máquina e rege todo o seu funcionamento. Essa etapa é composta por algumas unidades de processo, conforme cita Suh et al. (2008, p. 21), “de um ponto de vista funcional, o sistema CNC consiste da unidade IHM (Interface Homem-Máquina), da unidade NCK (*Numerical Control Kernel* ou Núcleo de Controle Numérico) e da unidade de CLP (Controle Lógico Programável)”. Ainda, segundo os autores, “a unidade NCK, sendo o núcleo do sistema CNC, interpreta o código G e executa interpolação, controle de posição [...] baseado no código G interpretado. Por fim, ela controla o servossistema levando a peça de trabalho a ser usinada”.

Portanto, a chamada unidade NCK lida com as principais funções da CNC e é parte essencial da máquina. Por se tratar de uma etapa de *software*, sua criação, modificação e adequação podem ser feitas de maneira flexível e para diversos tipos de máquinas e de configurações de acordo com o que cada aplicação exige.

1.1 OBJETIVO GERAL

O objetivo geral do trabalho é o projeto e implementação de uma CNC desde o desenvolvimento dos quesitos de programação referentes à unidade NCK até à elaboração e integração de suas partes. Assim, interpretando o código G a partir da placa de circuito, a fresa precisa ser capaz de confeccionar as trilhas do circuito projetado.

1.2 OBJETIVOS ESPECÍFICOS

- Adquirir conhecimento sobre o controle de motores de passo aplicados a fresadoras e fazer ambientação com o microcontrolador;
- Propor uma metodologia de projeto para a máquina fresadora em âmbito estrutural e de funcionamento;
- Montagem da parte mecânica e elétrica;
- Projetar uma placa de circuito impresso padrão de teste no *software* Eagle;
- Avaliar os dados exportados pelo *software* Eagle e fazer as adequações necessárias para a utilização do código G proposto;
- Obter os modelos matemáticos dos motores e realizar simulações para investigar o comportamento dinâmico do sistema em malha aberta;
- Implementar o código interpretador desenvolvido no microcontrolador;
- Verificar experimentalmente a máquina fresadora.

2. MÁQUINA FRESADORA CNC DE BAIXO CUSTO

A fresa CNC de baixo custo implementada no trabalho possui sua estrutura baseada na máquina fresadora referenciada, sendo uma fresadora de topologia vertical. Contudo, neste projeto, diversas alterações foram feitas.

A máquina fresadora original possui sua estrutura em madeira MDF de espessuras de $10mm$ e $25mm$. Como forma de maior redução de custos, a fresa implementada no trabalho possui madeiras com espessuras de $6mm$ e $18mm$. A diminuição das espessuras das madeiras não compromete a máquina pois a MDF possui grande resistência, conforme mostra o quadro 1.

Quadro 1 – Propriedades da madeira MDF

Propriedades	Unidade	Valores obtidos com o MDF			
Espessuras	mm	3 - 6	9 - 18	20 - 25	30 - 35
Densidade	Kg/m³	800	750	670	650
Flexão Estática	Kgf/cm²	234	220	190	180
Tração Perpendicular	Kgf/cm²	6,6	5,8	5,6	5,1
Tração Superficial	Kgf/cm²	12,2			

Fonte: Adaptado de Duratex (2015)

É possível observar que para todos os valores de espessuras comerciais, a MDF tem capacidade de lidar com altos valores de tração e flexão, sendo assim propícia para a estrutura da máquina fresadora. As figuras 1 e 2 mostram o conceito da estrutura da máquina fresadora.

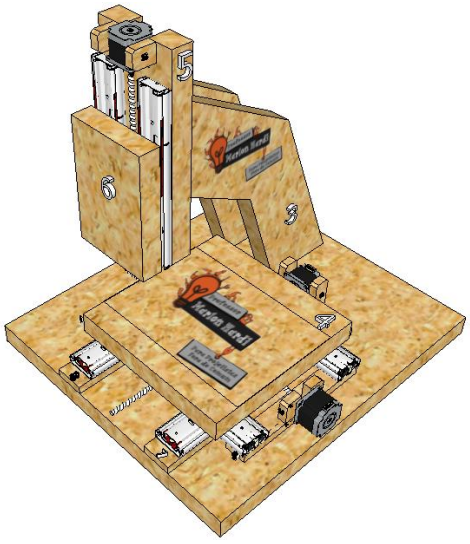


Figura 1 – Estrutura da máquina fresadora CNC
Fonte: Extraída de WALENDORFF (2017)

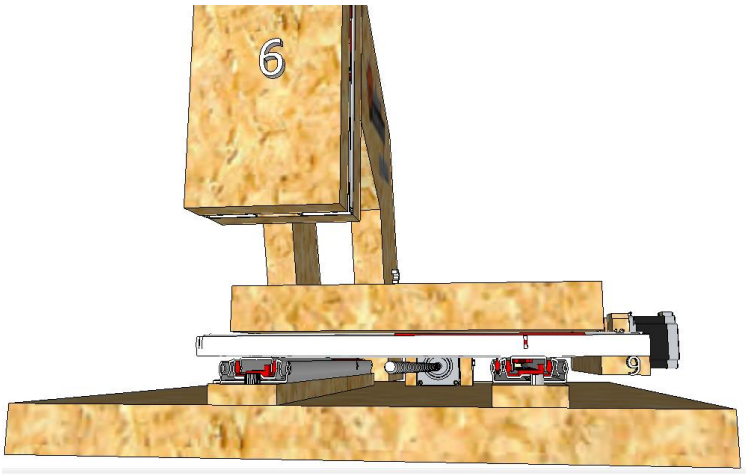


Figura 2 – Estrutura da máquina fresadora CNC, inferior
Fonte: Extraída de WALENDORFF (2017)

Conforme é visto nas figuras 1 e 2, a estrutura é composta por diversas peças de madeira MDF em cortes retos.

O sistema mecânico é composto por trilhos telescópicos *light* modificados nos quais somente uma de suas corrediças é utilizada. Os trilhos são movimentados por meio de barras de rosca do tipo $\frac{5}{16}$" *NC*.

Os motores de passo dos eixos Z e Y, encontrados respectivamente no eixo-árvore de acoplamento do motor CC (peça 6) e no piso da máquina são fixados na madeira e permanecem estacionários durante o funcionamento da CNC. Já o motor de passo do eixo X, embora também fixado na estrutura e sendo visto na figura 16 acoplado à peça de número 9, se movimenta de forma conjunta com os trilhos do eixo Y.

Embora a estrutura em madeira MDF seja referenciada, neste trabalho, diversas alterações foram feitas tanto na estrutura de madeira da máquina quanto nos suportes de cada eixo e tais alterações são apresentadas no capítulo 5.1.

A relação de materiais e custo da CNC é mostrada a seguir.

- 17 peças de madeira MDF espessuras 10mm e 18mm: R$80,00;
- 6 trilhos telescópicos light 25 cm: R$21,00;
- 1 cola instantânea de secagem rápida: R$07,00;
- 1 barra de rosca $\frac{5}{16}$" de 1 metro: R$03,00;
- 7 porcas para barra de $\frac{5}{16}$": R$01,50;

- Espaguete termo retrátil de 9,5mm: R$04,00;
- 2 parafusos 60mm x 4mm: R$00,50;
- 2 parafusos 80mm x 4mm: R$00,50;
- 4 parafusos 40mm x 4mm: R$00,30;
- 14 parafusos 14mm x 4mm: R$03,00;
- 4 parafusos 70mm x 4mm: R$00,50;
- 1 parafuso 60mm x 4mm com 2 porcas: R$00,20;
- 1 conjunto de 20 fresas: R$19,00;
- 1 mini mandril para ferramentas de 0,3 mm a 3,5 mm: R$33,00;
- 2 abraçadeiras metálicas tipo D 50mm diâmetro: R$05,00;
- 2 abraçadeiras metálicas tipo U 30mm diâmetro: R$03,50;
- 3 placas de circuito impresso 10cm x 15cm: R$16,00;
- Conectores KRE, pinos DIL, 2 reguladores lineares de tensão LM78: R$15,00;
- 1 motor CC 24V, 0,3 A, 3150 RPM: R$90,00;
- 3 motores de passo híbridos bipolares NEMA 17, 1,8° por passo, 1,1 kgf.cm, 70 mA: R$240,00;
- 3 *drivers* para motor de passo bipolar 2,5 A, dupla Ponte-H, frequência máx. 250kHz: R$55,00;
- 1 *driver* para motor CC, 43 A, dupla Ponte-H, frequência máx. 25kHz: R$40,00;
- 1 fonte de alimentação 24Vcc, 2,5A: R$29,00;
- 4 *coolers* 12V, 80mm x 80mm: R$40,00.
- 1 LaunchPad microcontrolador TI Delfino TMS320F28377S: R$100,00 reais;

Custo total: R$807,00.

3. FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo apresentar os conceitos mais importantes referentes aos motores de passo que fundamentam o trabalho, mostrando seus principais componentes que definem suas funções no projeto.

3.1 MOTOR DE PASSO

Existem dois tipos de motores que tipicamente são utilizados em máquinas CNC: o motor de passo e o servo motor. Ambos atendem às necessidades de um posicionamento preciso e devem passar por uma etapa de *drive* para poderem ser acionados e ter as informações de direção e rotação trabalhadas. (OVERBY, 2010). Tais motores diferem um pouco em questão de funcionamento, de componentes utilizados e de precisão, ao passo que um ou outro pode ser adotado de acordo com as especificações do projeto. A escolha do motor de passo para o projeto se dá inicialmente por conta do menor custo de implementação, além de quê, o acionamento em malha aberta é suficiente para a aplicação desejada.

O motor de passo é um tipo especial de motor síncrono sem escovas que é projetado para girar um ângulo específico a cada pulso elétrico recebido em sua unidade de controle. Em motores síncronos comuns, os enrolamentos de fase são energizados pela rede trifásica convencional defasada de 120°, criando assim um campo rotacional contínuo. Já nos motores de passo, os polos magnéticos do estator têm em um ângulo finito cada vez que uma nova fase é acionada por uma fonte CC e a fase anterior é desativada. Conceitualmente, o motor de passo só pode receber um pulso e efetuar um passo por vez, e cada passo deve ser do mesmo comprimento.

Ao excitar sequencialmente as fases de um motor de passo, o rotor gira na forma de uma sequência de passos, girando de um ângulo específico a cada passo (FITZGERALD; JR; UMANS, 2003).

São três as topologias básicas desse tipo de máquina, a de relutância variável, a de ímãs permanentes e a híbrida. O motor de passo híbrido é o mais utilizado e é o adotado no projeto pois possui melhorias em relação aos outros dois tipos. O nome híbrido é dado por conta de seu rotor que possui as características dos outros dois

tipos juntas, tendo um rotor polarizado como no tipo de ímãs permanentes e com vários dentes e ranhuras como no de relutância variável. Assim, acrescentam os autores Fitzgerald, Jr. e Umans (2003, p. 390), “esses motores frequentemente combinam uma geometria de relutância variável com ímãs permanentes para produzir aumentos de conjugado e na precisão de posicionamento”.

Os motores de passo híbridos do projeto são classificados como NEMA 17, possuem 2 fases com bobinas bifilares e têm ângulo de passo de 1,8° em acionamento padrão de *full-step* ou, passo completo. A figura 3 mostra a vista frontal de um motor de passo híbrido de 2 fases e 8 polos.

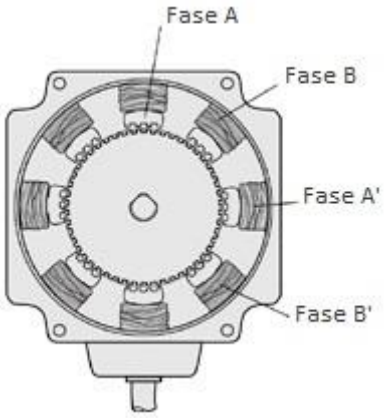


Figura 3 – Vista frontal de um motor de passo híbrido
Fonte: Extraída de Oriental Motor Corp. (2017)

3.1.1 Construção e funcionamento do motor de passo híbrido

O motor de passo híbrido é composto por um estator contínuo com sua estrutura de polos ao longo do comprimento do rotor, ao passo que o rotor possui duas seções idênticas, sendo as duas seções montadas axialmente no mesmo eixo, como pode ser visto pela figura 4.

As duas seções do rotor são caracterizadas por terem suas polaridades norte e sul definidas, que podem ser obtidas através de uma bobina CC estacionária posicionada radialmente na parte central do rotor, entre as duas seções ou, ao invés disso, podem ser obtidas com a utilização de um imã permanente posicionado axialmente entre as duas seções, sendo que esta é a configuração mais adotada em motores de passo híbrido e é a encontrada nos motores do projeto. O rotor possui dentes e ranhuras, e as ranhuras não são bobinadas.

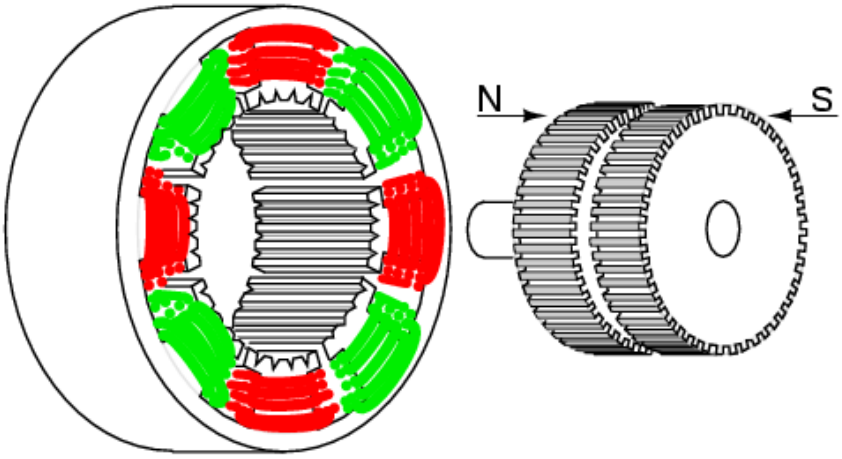


Figura 4 – Estator e rotor de um motor de passo híbrido
Fonte: Adaptada de MORAR, Alexandru (2014, p. 2)

Além disso, as duas seções do rotor são deslocadas entre si por metade de um dente, conforme mostra a figura 5, bem como indica a polaridade do imã permanente axial que define as polaridades de cada seção. Devido ao deslocamento, o rotor tem efetivamente o dobro de intervalos de polos da polaridade oposta. A grande maioria dos motores de passo híbridos NEMA 17 com 200 passos por volta possuem 50 dentes por seção.

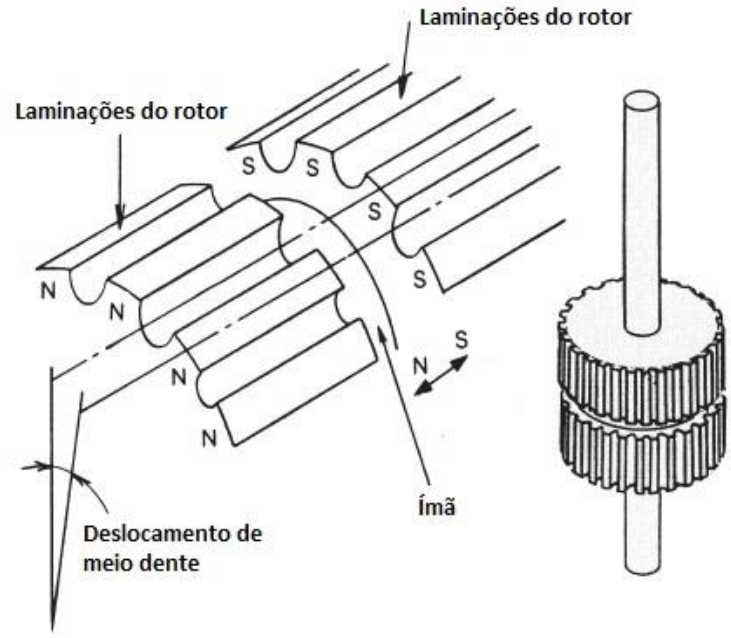


Figura 5 – Detalhes das duas seções rotóricas do motor
Fonte: Extraída de IAASR (2013)

Como os dentes dos polos do estator correspondem aos 50 dentes de cada seção do rotor, exceto para os dentes no espaço entre os polos estatóricos, um dos polos do rotor se alinha com o estator em 50 posições distintas, porém, com a

existência do deslocamento, o rotor se alinha então em 100 posições distintas.

Ainda, como mencionado anteriormente, o motor possui duas fases no estator, sendo que elas são também deslocadas entre si, mas por ¼ de um dente. Com isso, o rotor se move em passos de ¼ de um dente de uma seção do rotor quando as fases são energizadas alternadamente, chegando aos 200 passos necessários para se ter um giro completo, ou seja, possuindo resolução de 1,8° por passo.

O funcionamento do motor de passo híbrido, assim como em suas outras topologias, consiste no acionamento sequencial de suas fases de modo que a fase posterior seja energizada mantendo o rotor girando no mesmo sentido.

Existem dois métodos de sequência de acionamento, onde o primeiro consiste em acionar somente uma fase por vez e, dessa forma, se minimiza o consumo de energia e se reduz o torque disponível. Já o segundo método, implementado no *driver* DRV8825, consiste em acionar ambas as fases do motor por vez, fazendo com que o torque aumente em relação ao método anterior. Tal sequência de acionamento é ilustrada pela figura 6, em referência ao motor da figura 1.

Figura 6 – Sequência de acionamento de fases do motor
Fonte: Extraída de Oriental Motor Corp. (2017)

Como é possível notar pela figura 6, além do acionamento das fases ocorrer de forma sequencial, as fases mudam de polaridade durante o processo. A mudança de polaridade é uma característica da ligação bipolar dos motores. Neste tipo de conexão, as fases do motor não possuem derivação central, em comparação com a conexão unipolar, como mostra a figura 7.

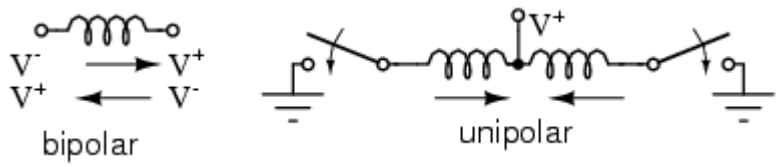


Figura 7 – Ligação bipolar e unipolar das fases de um motor de passo
Fonte: Extraída de All About Circuits (2016)

Dessa forma, a corrente flui pelo enrolamento inteiro em vez de somente metade dele e, com isso, um motor com ligação bipolar produz mais torque do que

outro de mesmo tamanho com ligação unipolar (CONDIT, 2004). Esse aumento de torque é indicado no *datasheet* dos motores de passo do projeto, que mostra um aumento de 30% no torque gerado pelos motores na escolha da ligação bipolar.

A consequência negativa da ligação bipolar é a necessidade de um circuito de controle um tanto mais complexo em relação a um circuito para a ligação unipolar. Como a corrente nesta ligação flui em sentido bidirecional, faz-se necessária a troca de polaridade nos enrolamentos.

Para tal, utiliza-se um circuito de controle conhecido como *H-Bridge* ou, Ponte-H, que é usado justamente para variar a polaridade nas extremidades do enrolamento. Como o motor de passo bipolar possui dois enrolamentos, são requeridas duas Pontes-H para controlar cada motor.

3.1.2 Ponte-H

Conforme descrito, basicamente, uma ponte-H se configura como um circuito que permite que a corrente num enrolamento flua em ambos os sentidos, fazendo com que, dentre outras coisas, o motor possa girar também nos dois sentidos. Existem distintas configurações de circuitos para a ponte-H, contendo mais ou menos transistores, diodos em conexões diversas, dentre outras características.

O circuito comumente utilizado consiste de duas pontes-H compostas por 4 transistores e 4 diodos, como mostra a figura 8.

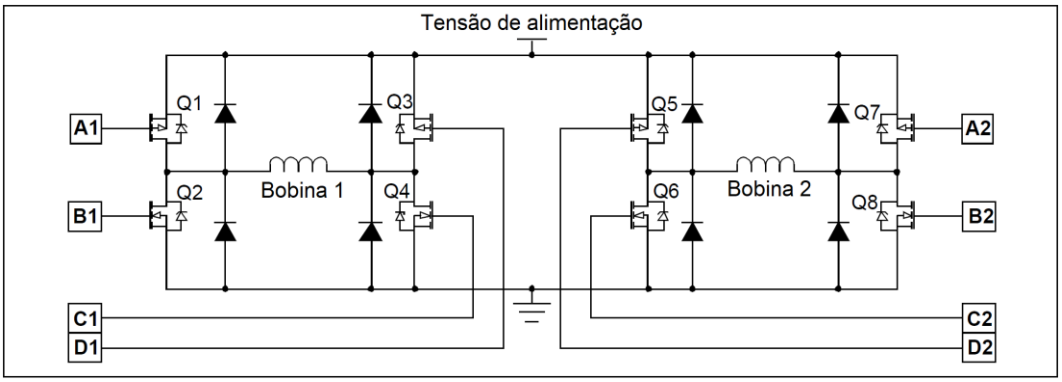


Figura 8 – Pontes-H com 4 chaves MOSFET e 4 diodos
Fonte: Extraída de CONDIT, Reston; JONES, Dr. Douglas W (2004, p. 9)

Tal circuito é o mais utilizado por necessita de somente uma fonte unipolar para funcionar, além de possibilitar que a corrente flua no enrolamento inteiro, promovendo a melhoria de torque já explanada. Nesta topologia de circuito, a corrente

percorre o enrolamento quando os transistores em diagonal são ativados. Portanto, avaliando a figura 8, ao serem ativadas as chaves Q1 e Q4 enquanto Q2 e Q3 permanecem desativadas, a corrente percorre o enrolamento 1 no sentido da esquerda para a direita; de forma complementar, sendo ativados os transistores Q2 e Q3 enquanto Q1 e Q4 permanecem desativados, a corrente percorre o enrolamento no sentido da direita para a esquerda.

Os 4 diodos vistos no circuito são utilizados para proteger os transistores dos picos de tensão causados pelo chaveamento no enrolamento. Como o enrolamento do motor é basicamente um indutor, a corrente não pode variar bruscamente e os diodos em paralelo com os transistores fornecem um caminho para o escoamento da corrente.

3.1.3 Micro-passos

Dada a relativa baixa resolução de passos e a ocorrência natural das vibrações nos motores de passo em *full-step*, normalmente opta-se pelo uso de micro-passos em aplicações onde a elevada precisão é fator chave, como na máquina fresadora CNC.

Como cita Condit (2004, p. 10), o processo de micro-passos “é utilizado para atingir aumentada resolução de passos e transições mais suaves entre passos”, e conclui que “aumenta o desempenho do sistema enquanto limita problemas com ruído e ressonância”.

O processo de micro-passos funciona com o princípio de transferir gradualmente a corrente de uma fase para a outra. Essa dinâmica é obtida pela técnica de modulação por largura de pulso (PWM) sobre a tensão nas fases do motor. O ciclo de tarefa do sinal sobre um enrolamento é diminuído ao passo que o ciclo de tarefa do sinal no enrolamento seguinte é aumentado.

A movimentação desejada de um motor de passo é linear, o que significa que os passos devem ser iguais em tamanho sem aceleração e desaceleração notável do eixo enquanto o motor gira. As implementações de micro-passos procuram chegar o mais próximo possível desse movimento linear (CONDIT, 2004).

Existem dois métodos utilizados para o processo de micro-passos. No primeiro deles as correntes nos enrolamentos se alternam entre se manter constantes e aumentar e diminuir em rampa, já no segundo, denominado Micro-passos seno-cosseno, a corrente adota uma curva próxima da senoidal em ambos os enrolamentos. Este segundo método é o mais comum e implementado no *driver*. A figura 9 mostra o perfil das correntes em ambas fases do motor.

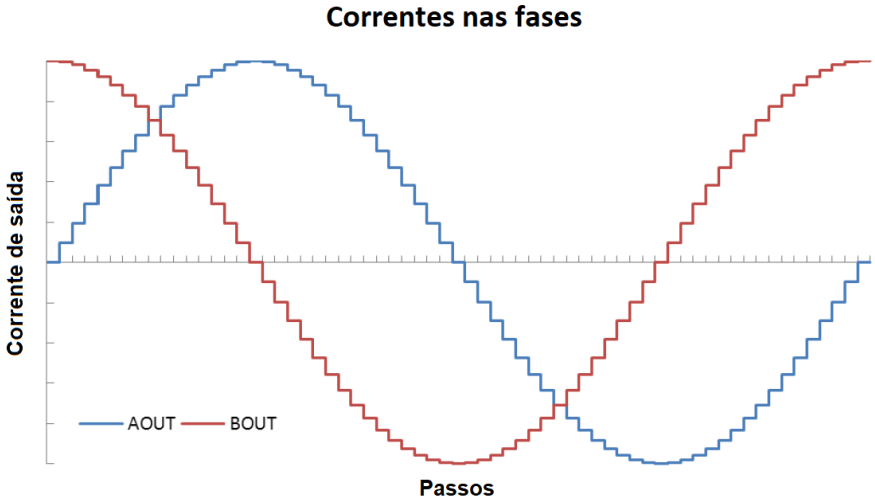


Figura 9 – Formato de onda das correntes em micro-passos
Fonte: Adaptada de DRV8825 *Controller* (2014, p. 2)

O método utilizado ajusta a corrente em cada enrolamento de forma que o torque líquido produzido se mantenha constante. O torque do motor tem formato aproximadamente senoidal. Idealmente, o torque produzido por cada enrolamento é proporcional à corrente naquele enrolamento e varia linearmente.

3.1.4 Modelo matemático do motor de passo híbrido

O modelo matemático do motor de passo híbrido surge inicialmente do circuito equivalente de uma de suas fases, mostrado na figura 10.

Diferentemente de um motor de passo de relutância variável que não possui força contraeletromotriz, o motor de passo híbrido possui uma força contraeletromotriz representada por $e_a(\theta)$ que depende, dentre outros fatores, do seno da posição atual do rotor. Ainda, em comparação com a topologia citada, na versão híbrida a indutância

do enrolamento pode ser considerada independente da posição do rotor devido ao entreferro relativamente grande introduzido pelo ímã permanente.

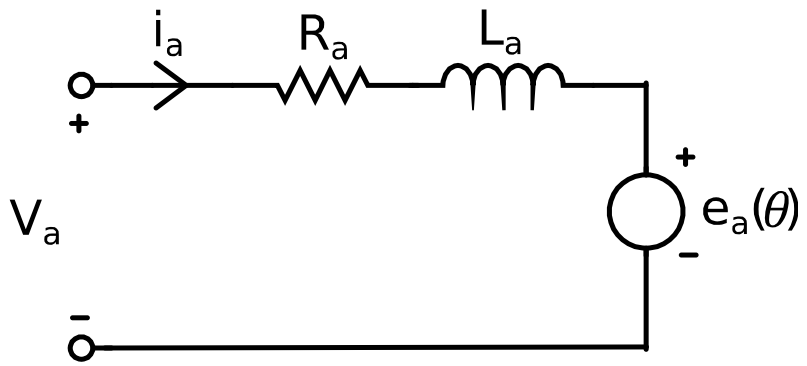


Figura 10 – Circuito equivalente da fase A do motor de passo híbrido
Fonte: autoria própria

A resistência e indutância da fase são representadas respectivamente por R_a e L_a, enquanto v_a e i_a são referentes à tensão e corrente.

As equações que descrevem o comportamento do motor são encontradas através de análises de circuitos elétricos e magnéticos, assim como por considerações construtivas do motor e são apresentadas a seguir.

As correntes nas fases A e B do motor de passo híbrido surgem pela análise do circuito RL, logo, são representadas respectivamente pelas equações 1 e 2.

$$\frac{di_a}{dt} = \frac{\left(v_a - R_a i_a - e_a(\theta)\right)}{L_a} \tag{1}$$

$$\frac{di_b}{dt} = \frac{\left(v_b - R_b i_b - e_b(\theta)\right)}{L_b} \tag{2}$$

A força contraeletromotriz em cada fase é vista pelas equações 3 e 4.

$$e_a(\theta) = -K_m \omega sen(N_r \theta) \tag{3}$$

$$e_b(\theta) = K_m \omega cos(N_r \theta) \tag{4}$$

Como já descrito, essa força que se opõe à magnetização das fases é influenciada pelo seno do ângulo atual do rotor θ, ângulo este que varia conforme o

rotor gira e também pelo número de dentes N_r em cada polo do rotor. Além disso, a velocidade angular ω e a constante de torque do motor K_m também são levadas em consideração.

O torque exercido pelo motor em rotação é igual à soma dos torques resultantes da interação entre as correntes das fases e os fluxos magnéticos criados e depende também do torque de detenção inerente da saliência do rotor. O *holding torque,* sendo o torque necessário para rotacionar o eixo enquanto os enrolamentos estão energizados, pode ser representado então em função da velocidade e aceleração angular pela equação 5, que representa um sistema genérico rotacional com inércia e atrito, mas também pode ser equacionado como mostra a equação 6, proveniente da soma de equações que representam os torques de cada fase do motor com o torque de detenção.

$$T_e = J\frac{d\omega}{dt} + B\omega \tag{5}$$

As constantes J e B são, nessa ordem, a inércia rotórica e o amortecimento rotacional.

$$T_e = -K_m\left(i_a - \frac{e_a(\theta)}{R_m}\right)sen(N_r\theta) + K_m\left(i_b - \frac{e_b(\theta)}{R_m}\right)cos(N_r\theta) - T_d sen(4N_r\theta) \tag{6}$$

Já na equação 6, R_m se refere à relutância de magnetização e T_d ao torque de detenção do motor.

3.1.5 Controle de motores de passo

Motores de passo geralmente são motores de baixa velocidade e potência e utilizados em malha aberta, com raras exceções em aplicações especiais. Nos casos onde as características da carga são conhecidas e não variam muito, um simples controle em malha aberta é suficiente, e componentes tais como transdutores de posição e velocidade, trens de engrenagens, etc., são desnecessários (VENKATARATNAM, 2009). Com isso, os motores de passo não necessitam do uso de *encoders* para sua operação e, assim, podem ser acionados em malha aberta

(OVERBY, 2010). Dessa forma, eles são usados em muitos sistemas de controle, fazendo com que a posição de um eixo ou de outros mecanismos possa ser controlada precisamente (CHAPMAN, 2013).

No acionamento em malha aberta, a posição rotórica não é monitorada a fim de descobrir se o motor de fato executou um passo para um dado comando. Ainda assim, o acionamento em malha aberta é extremamente satisfatório para diversas aplicações, como numa máquina fresadora. É somente preciso garantir que o motor desenvolva o torque necessário para vencer o torque da carga e também que o intervalo de tempo entre os pulsos de entrada seja suficientemente grande para que não ocorra perda de passos. Se estas condições forem cumpridas, o motor não falha em seus passos (VENKATARATNAM, 2009).

Ademais, conforme VENKATARATNAM (2009, p. 20) “uma vez que o motor execute os passos, o erro posicional mesmo para um grande número de passos é limitado a somente uma fração do último passo, e somente se houver algum torque na carga. Por fim, o erro posicional não é cumulativo, só dependendo de um único ângulo de passo, sendo independente do número total de passos do motor”.

Portanto, segundo apontam os autores, o acionamento em malha aberta de um motor de passo de fato é a escolha mais conveniente para a aplicação proposta, não acarretando em perda de desempenho do sistema e nem de qualidade, além de propiciar que a máquina se torne mais barata e tenha seu desenvolvimento menos complexo.

4. CARACTERIZAÇÃO DO PROJETO

Este capítulo tem por objetivo a apresentação das partes constituintes do projeto que prenunciam e baseiam o desenvolvimento do trabalho.

4.1 MÁQUINA FRESADORA

Uma máquina fresadora faz parte de um conceito de máquinas denominadas de *máquinas-ferramenta,* as quais são utilizadas na fabricação de outras máquinas, e por esse motivo também são conhecidas como *mother machines*, máquinas mãe, em português (SUH et al., 2008). Esse tipo de máquina se caracteriza por fabricar variados tipos de peças, normalmente de geometria complexa, podendo ser de inúmeros materiais tais como metais, plásticos, madeiras, entre outros. Ela é um equipamento especializado em cortar a matéria-prima utilizando uma ferramenta de corte chamada de fresa, o que leva a máquina comumente a ser chamada deste mesmo nome.

Existem vários tipos de topologias de máquinas fresadoras, as quais são distinguidas com base em seu perfil estrutural. As máquinas fresadoras comumente encontradas possuem perfil horizontal ou vertical, como mostra a figura 11.

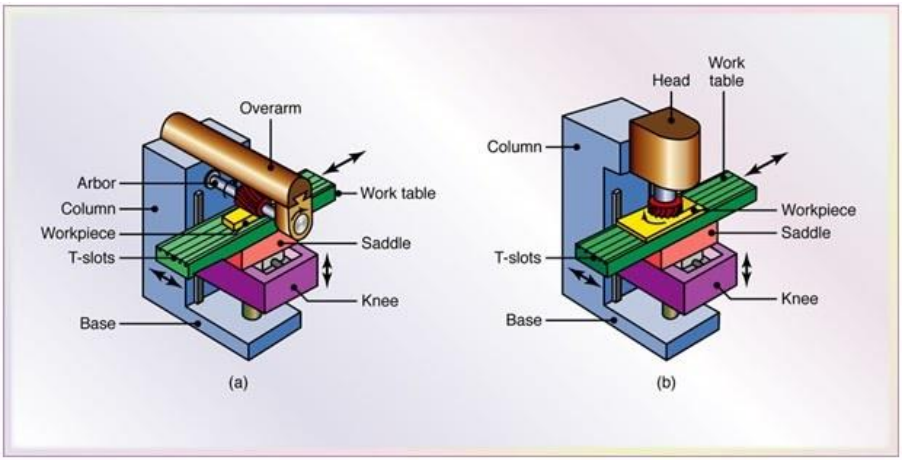


Figura 11 – Máquinas fresadoras convencionais: (a) Perfil horizontal; (b) Perfil vertical
Fonte: Extraída de Machining Handbook (2013)

A fresa CNC aplicada no trabalho possui perfil vertical, como é representado pela figura 9(b). Neste perfil, o eixo-árvore da máquina representado na figura por *head,* cabeça em português, se encontra em direção vertical, estando sempre perpendicular à mesa de trabalho, representada por *work table,* de cor verde.

Esta topologia de máquina fresadora é a mais utilizada quando se necessita do uso de ferramentas de formato delgado, como uma fresa para usinagem de trilhas de circuito impresso, sendo a mais adequada para o fresamento de ranhuras e de furações na peça de trabalho.

Além disso, as máquinas fresadoras se diferenciam por dois tipos possíveis de movimentação durante o processo de usinagem. No primeiro deles, o eixo-árvore da máquina é o que se movimenta nos três eixos cartesianos para usinar a peça, enquanto a mesa de trabalho se mantém fixa. Já no segundo tipo, sendo o aplicado no trabalho, o eixo-árvore possui movimentação somente no eixo Z, enquanto a mesa de trabalho se movimenta nos eixos X e Y.

4.1.1 Especificações da máquina fresadora CNC

Além das especificações de topologia da máquina de acordo com o que se deseja usinar, a fresa CNC é especificada de acordo com sua capacidade de usinagem. O primeiro parâmetro de especificação é o tamanho disponível da mesa de trabalho, a qual possui dimensões de $200mm\ x\ 200mm$, suficiente para usinar uma placa criada na versão livre do *software* Eagle.

Como o *software* CAD/CAM Eagle em sua versão livre utilizada no trabalho se limita a projetos de placas de circuito com dimensão máxima de $100mm\ x\ 80mm$, a área bruta disponível pela mesa de trabalho possibilita a usinagem de qualquer placa projetada. Outros *softwares* também podem ser usados, contanto que o projeto da placa esteja dentro da dimensão máxima.

Entretanto, é preciso garantir também que os eixos da fresa possuam alcance suficiente para tal e, com isso, o alcance de movimentação dos eixos da máquina fresadora também é levado em consideração. A fresa CNC projetada tem capacidade de movimentar os eixos X, Y e Z por, respectivamente, $140mm$, $140mm$ e $35mm$. Com isso, garante-se de fato o alcance necessário para os três eixos.

A troca de ferramentas da CNC é feita de forma manual, sendo possível utilizar ferramentas de 0,3 mm a 3,5 mm.

Para a usinagem adota-se a face da placa de fenolite como o posicionamento zero.

Ademais, uma última especificação relevante da máquina diz respeito à

velocidade de usinagem ou, velocidade de avanço da fresa. A fresa CNC possui duas velocidades de avanço para os eixos, sendo uma a respeito do comando *G00* de velocidade rápida e a outra para o comando *G01* de interpolação linear.

Durante o comando *G00* definiu-se a velocidade nos eixos de $75\ RPM$. Logo, baseando-se no passo da barra de rosca, explanado posteriormente no trabalho, a movimentação rápida possui avanço de $105{,}75\ mm/min$.

Já a velocidade durante a interpolação linear é definida para somente $15\ RPM$ devido às limitações explanadas no decorrer do trabalho, e assim, a velocidade de avanço durante o comando *G01* é de $21{,}15\ mm/min$.

4.2 CONTROLE NUMÉRICO COMPUTADORIZADO

O Controle Numérico Computadorizado possui um processo bem definido e é guiado por um *software* que faz a IHM e por um sistema microcontrolado que desempenha as funções da NCK e executa o controle do servossistema.

A unidade IHM se refere ao *software* CAD/CAM Eagle para o projeto da placa de circuito impresso, que é o primeiro estágio do processo. Já para a criação das funções da NCK é utilizado o *software* Code Composer Studio, sendo o IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado) dos microcontroladores da Texas Instruments onde é feito então todo o processo, como interpretação do código G, a interpolação e o controle do servossistema.

O código G exportado de acordo com a placa projetada necessita de interpretação, já que, como destaca Overby (2010, p. 90), esta não é uma linguagem de programação que exige compilação, ao invés disso, ela precisa ser interpretada pelo controlador utilizado para que possa ser usada.

No *software* Eagle, o projeto da placa de circuito impresso é feito e o Código G é exportado em formato de texto. A partir daí, são feitas adequações no código original exportado utilizando o *software* Excel para que ele possa ser interpretado e processado de maneira correta. Após a adequação do código, ele é avaliado em simulação com o *software* Universal Gcode Sender para se certificar de que os dados da placa projetada estão de acordo com o esperado e, em seguida, o código G é inserido nos vetores pertinentes no código interpretador desenvolvido no microcontrolador.

Como já descrito, o código interpretador criado no IDE do microcontrolador é encarregado de fazer todo o processo da unidade NCK da CNC. A figura 12 mostra um diagrama de todas as etapas necessárias, incluindo as etapas desta unidade.

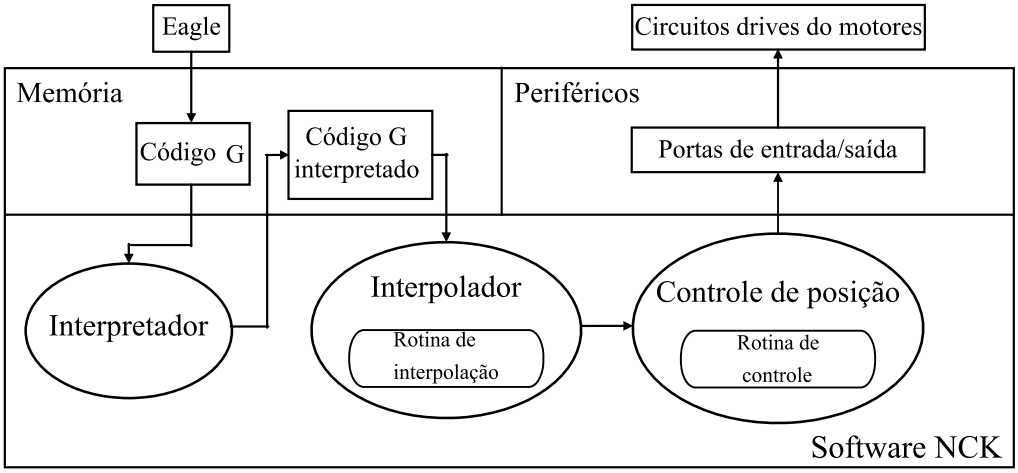


Figura 12 – Diagrama do processo CNC
Fonte: Adaptada de Suh et al. (2008, p. 26)

É possível observar pelo diagrama o seguimento de cada etapa do processo CNC. De início, a placa é projetada no *software* Eagle e o Código G exportado já adequado é inserido na memória do microcontrolador, tendo os dados armazenados nos vetores de comando e coordenadas criados. Esta é a etapa *offline* do processo CNC, ainda sem atuação das funções de processamento da unidade NCK.

Já a partir da NCK os dados da placa projetada inseridos nos vetores são processados a partir da compilação do código interpretador desenvolvido. O módulo interpretador, sendo o primeiro da NCK é responsável por interpretar os comandos de Código G. Para isso, cada posição do vetor de comando de código G denominado *GCODE* é avaliada por meio de uma função onde são feitas comparações com constantes definidas para validar o tipo de ação a ser tomada.

Feito isso, o Código G agora interpretado segue para a rotina de interpolação onde são calculadas as distâncias de deslocamento de cada eixo baseadas na mesma posição correspondente do comando interpretado, representando assim o processamento da linha completa do Código G exportado.

Por fim, a etapa de controle de posição dos motores de passo é responsável por controlar o sentido de giro e o tamanho do trem de pulsos enviado aos circuitos *drives* dos três eixos, tendo como referência os valores obtidos pela etapa de interpolação.

4.3 HARDWARE UTILIZADO E ESQUEMA ELÉTRICO

O hardware correspondente ao microcontrolador se constitui do kit de desenvolvimento LaunchPad do microcontrolador da família C2000 Delfino TMS320F28377S, observado pela figura 13.

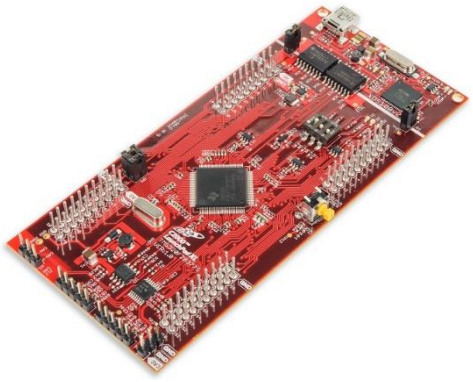


Figura 13 – LaunchPad do microcontrolador Delfino
Fonte: F28377S LaunchPad Development Kit

Este microcontrolador é utilizado no projeto pois, além de possuir todos os módulos necessários ao sistema, ele possui unidade de processamento de ponto flutuante, sendo necessária já que o código G a ser interpretado é composto de coordenadas neste formato. Além disso, o microcontrolador possui referência nesta mesma aplicação, sendo de fato utilizado em máquinas CNC profissionais.

O circuito *driver* utilizado para receber os sinais do microcontrolador e acionar os motores de passo é o DRV8825. O *driver* é escolhido para o projeto pois atende a necessidade de acionar as fases dos motores em ligação bipolar e, além disso, permite a adoção de acionamento em micro-passos, sendo imprescindível para o bom desempenho da máquina fresadora. A figura 14 mostra o circuito *driver* com as conexões necessárias para o acionamento dos motores no modo descrito.

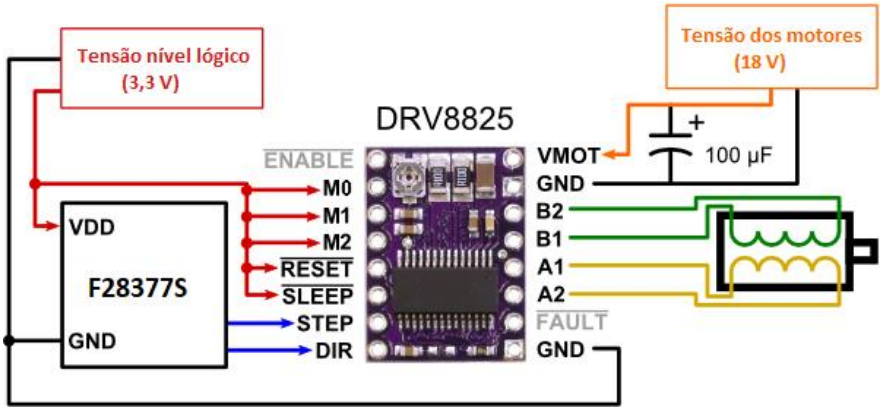


Figura 14 – DRV8825 com as conexões utilizadas
Fonte: Adaptada de DRV8825 Stepper Motor Driver Carrier

A figura 12 ilustra o *driver* de acionamento dos motores de passo, o qual possui dois níveis de tensão de entrada distintos, um para energizar as fases do motor e outro para acionar a lógica de controle de baixa potência.

Para a energização das fases, o *driver* recebe uma tensão de 18V proveniente do regulador de tensão linear LM7818. Este valor de tensão é aplicado pois o *datasheet* dos motores indica a necessidade de pelo menos 16,8V para a energização dos motores em ligação bipolar.

Além disso, nesta ligação, para que os motores de passo exerçam o torque nominal de 1,1 kgf.cm é preciso garantir que suas fases recebam o valor nominal de corrente de 70 mA. Para isso, o *driver* possui um e*ncoder* giratório que regula o nível de tensão do circuito limitador de corrente, sendo possível então definir a corrente necessária.

A corrente de cada fase do motor é expressada pela equação 7.

$$Inom = \frac{V_{ref}}{5 * R_{sense}} \tag{7}$$

Com a equação (7) o valor da tensão V_{ref} é definido de acordo com a corrente nominal $Inom$ que se deseja para o motor. Com o ganho de valor 5 e o resistor de limitação de corrente do *driver*, de 200 mΩ calcula-se a tensão a ser ajustada, chegando ao valor de 70 mV.

A tensão pode ser medida e ajustada através do *encoder* giratório do *driver* DRV8825.

Já os sinais de controle possuem tensões de 3,3V enviados pelo microcontrolador. A entrada *ENABLE negada* do *driver* não recebe ligação pois é conectada internamente por um resistor *pulldown* e, para esta entrada, o nível baixo é o que ativa as saídas e as operações de indexação do *driver.*

As entradas *M0, M1* e *M2* dizem respeito aos modos de acionamento do *driver*, e, estando todas conectadas em nível alto, o modo de 32 micro-passos por passo é selecionado. Já as entradas *RESET* e *SLEEP* ativam, respectivamente, as Pontes-H para cada fase do motor e o *driver,* o retirando do modo de baixo consumo.

Por fim, as entradas *STEP* e *DIR* recebem, nessa ordem, os sinais provenientes do microcontrolador para controlar o trem de pulsos e o sentido de giro

do motor.

Como não é possível utilizar um mesmo *driver* para controlar mais de um motor de passo, são utilizados 3 drivers DRV8825, um para cada motor contendo as mesmas conexões para cada um.

O esquemático do circuito de potência do projeto é ilustrado pela figura 15.

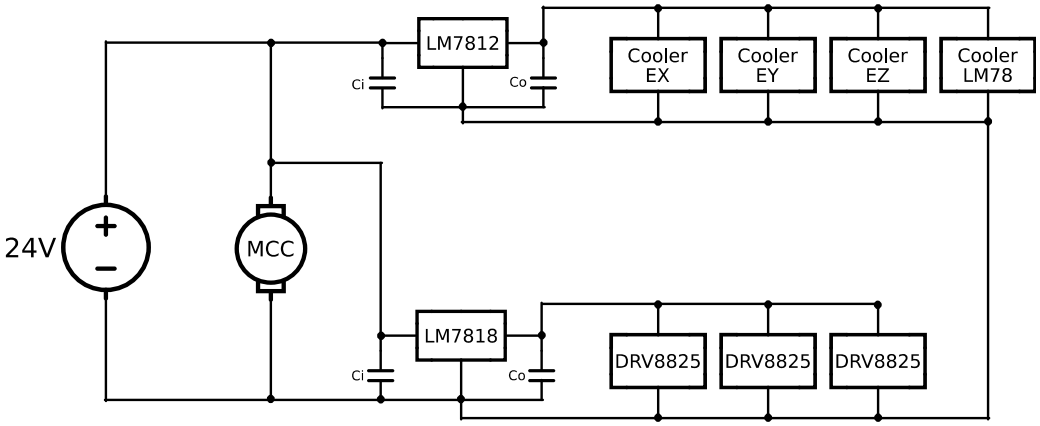


Figura 15 – Esquemático do circuito de potência
Fonte: autoria própria

Observa-se pela figura 15 que o circuito de potência do projeto é alimentado por uma fonte CC de 24V. O motor CC acoplado no eixo-árvore da máquina fresadora CNC tem seu enrolamento alimentado diretamente pela fonte de tensão por meio de seu *driver*, já que o motor CC possui a mesma tensão da fonte. Seria possível acionar o motor mesmo sem seu *driver*, porém, não seria possível controlar sua aceleração e seu acionamento de acordo com a interpretação do código, e por isso o *driver* é empregado.

Para a alimentação dos quatro *coolers* que refrigeram o sistema é utilizado o regulador linear de tensão LM7812 para prover os 12V requeridos. Também, como já citado anteriormente, os *drivers* dos motores de passo recebem a tensão de 18V a partir do segundo regulador linear.

A placa projetada no trabalho se refere ao circuito de potência do sistema sendo composta pelos reguladores lineares e por conectores para receber a tensão da fonte CC e alimentá-los, como será abordado posteriormente.

5. DESENVOLVIMENTO

5.1 MONTAGEM E ADEQUAÇÃO DA ESTRUTURA DA MÁQUINA FRESADORA CNC

Várias adequações do projeto original se fizeram necessárias para a utilização dos componentes do trabalho. Os motores de passo, assim como o motor CC utilizados no trabalho possuem dimensões distintas dos motores do projeto originalmente e, com isso, foram feitas mudanças nas dimensões de peças do projeto no plano de corte das madeiras.

A figura 16 apresenta o conjunto das 17 peças utilizadas na montagem estrutural da máquina.

Figura 16 – Peças MDF constituintes da máquina
Fonte: autoria própria

Para a montagem, diversos itens foram utilizados, tais como: régua, paquímetro, esquadros, cola epóxi, cola de contato, entre outros.

Conforme já mencionado, os trilhos telescópicos sofrem modificações a fim de manter somente uma das corrediças livres. Além de ser suficiente para percorrer toda a área necessária, isso faz com que o sistema se torne mais preciso, diminuindo a folga.

A figura 17 mostra um dos trilhos antes da modificação.

Figura 17 – Trilho telescópico original
Fonte: autoria própria

No trilho padrão observam-se três camadas, correspondentes à base, à primeira e à segunda corrediças. Utilizando um fio metálico, a primeira corrediça é presa ao suporte da base do trilho, sendo visto pela figura 18.

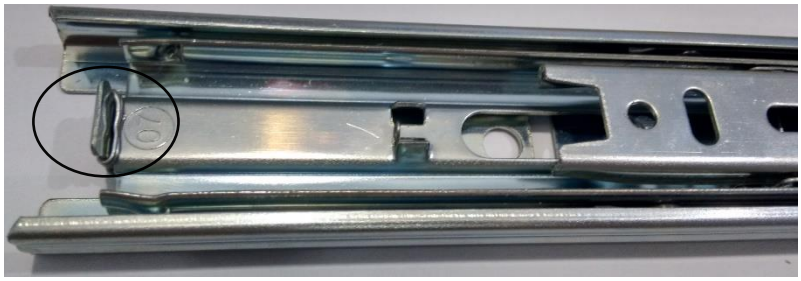


Figura 18 – Trilho telescópico modificado
Fonte: autoria própria

Assim, o conjunto de trilhos dos eixos X e Y são acoplados usando a chamada cola de contato. Como na superfície dos trilhos existem protuberâncias por conta de rebites, quatro peças de MDF da menor espessura são colocadas com o intuito de manter a mesa de trabalho plana. A figura 19 apresenta o conjunto montado.

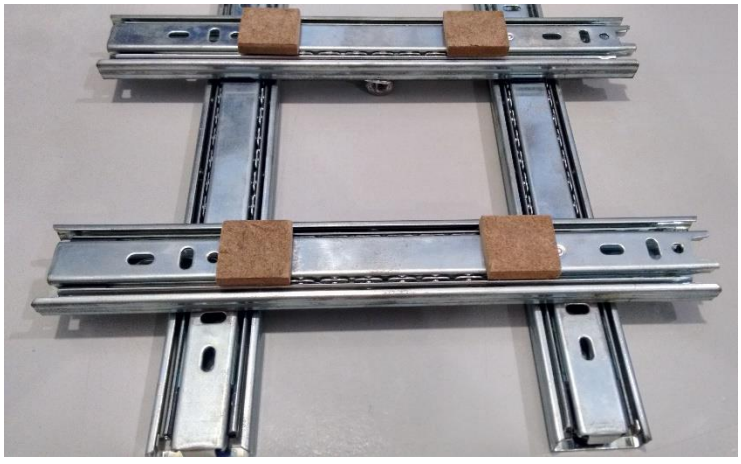


Figura 19 – Trilhos dos eixos X e Y montados
Fonte: autoria própria

A mesa de trabalho é colada nas quatro peças de MDF e serve também como base para o suporte do eixo X. A figura 20 mostra a parte inferior da mesa de trabalho onde é feito o suporte do eixo. Os suportes dos eixos X e Y possuem o mesmo conceito, sendo ambos vistos pela figura 20. Para a estruturação dos suportes são utilizados placa de fenolite, uma dobradiça modificada, parafusos e cola epóxi.

O eixo X, especificamente, possui uma segunda placa de fenolite onde se encontra uma segunda porca envolvida por parafusos. A segunda porca fica livre para movimentação e tem a função de se opor às variações laterais que ocorrem na barra de rosca quando em rotação. Nota-se também pela figura a utilização de graxa na barra de rosca, diminuindo o atrito com as porcas.

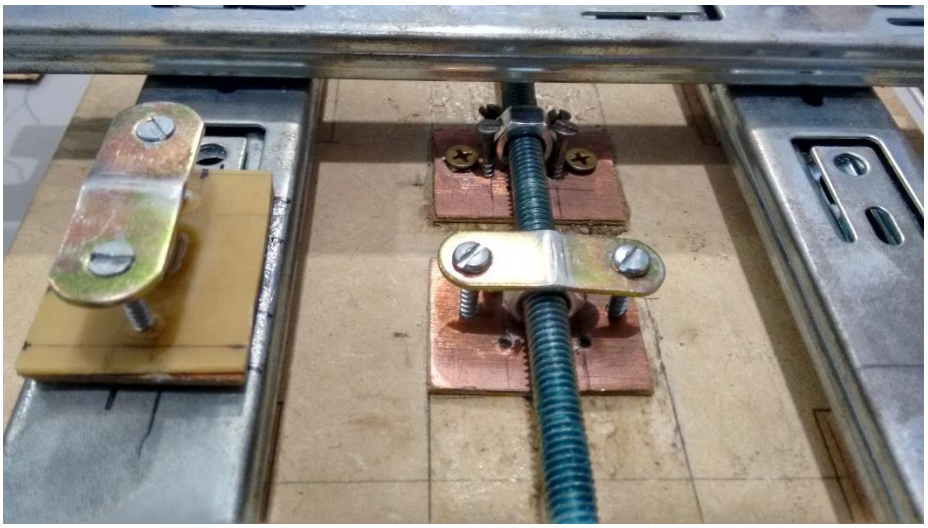


Figura 20 – Suportes dos eixos X e Y
Fonte: autoria própria

O suporte para o eixo Y é mostrado em destaque pela figura 21, sendo composto por três camadas de placa de fenolite, além da dobradiça comum modificada e os parafusos. As três camadas de placa são necessárias pois ajustam a altura da barra de rosca para coincidir com o eixo do motor de passo encontrado no piso da máquina.

As placas de fenolite são unidas e coladas na parte inferior do trilho por meio de cola de contato. Já a porca do eixo é colada na face não cobreada da fenolite utilizando cola epóxi e é pressionada pela dobradiça de forma homogênea por cada parafuso nas laterais. Os parafusos atravessam todas as camadas da placa e se alojam em furos existentes no trilho promovendo firmeza ao suporte.

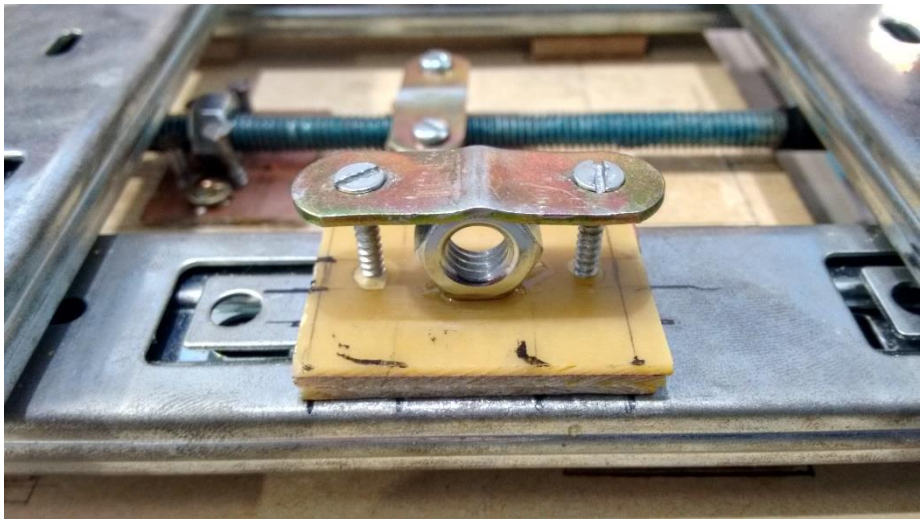


Figura 21 – Suporte do eixo Y
Fonte: autoria própria

Montados os trilhos dos eixos X e Y, o eixo Z é criado de maneira diferente, como aponta a figura 22.

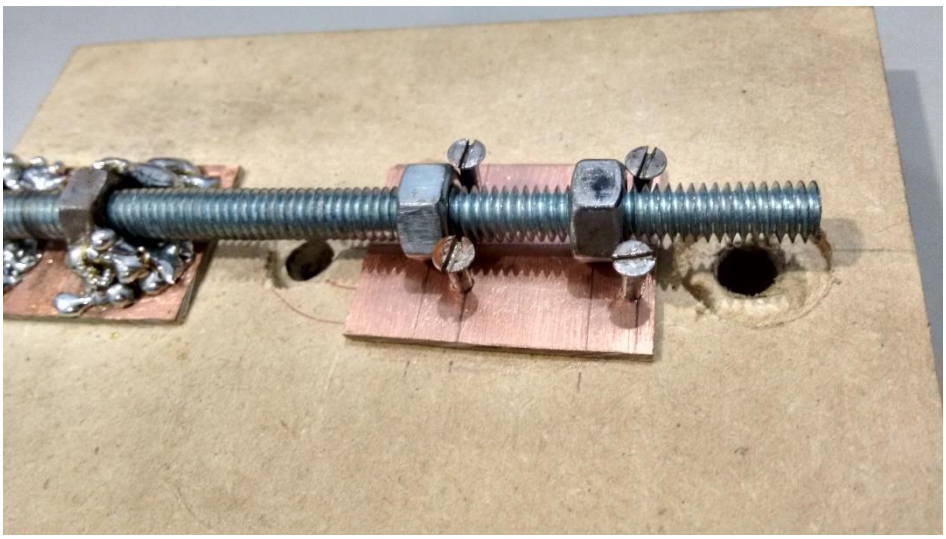


Figura 22 – Suporte do eixo Z
Fonte: autoria própria

Devido à massa relativamente alta que este eixo deve suportar, chegando a mais de 1kg em todo o conjunto, foram utilizadas quatro porcas para o suporte. As porcas são unidas à face cobreada das placas de fenolite por meio de solda, sendo que as duas porcas inferiores do eixo recebem também parafusos. Estes são ajustados de forma a se manter encostados às faces frontais das porcas que, com o eixo montado, se encontram para baixo. Com isso, os parafusos se opõem à força peso exercida auxiliando na robustez do eixo.

A figura 23 mostra o eixo Z finalizado, já acoplado aos trilhos.

Figura 23 – Eixo Z completo
Fonte: autoria própria

Para prender o motor CC ao eixo foram utilizadas duas abraçadeiras metálicas do tipo D, ilustradas pela figura 24.

Figura 24 – Acoplamento do motor CC
Fonte: autoria própria

A altura das abraçadeiras e do motor é definida de forma que a fresa possa chegar à placa de fenolite para o desbaste e furação.

Por fim, chega-se à estrutura final da máquina fresadora CNC observada pela figura 25.

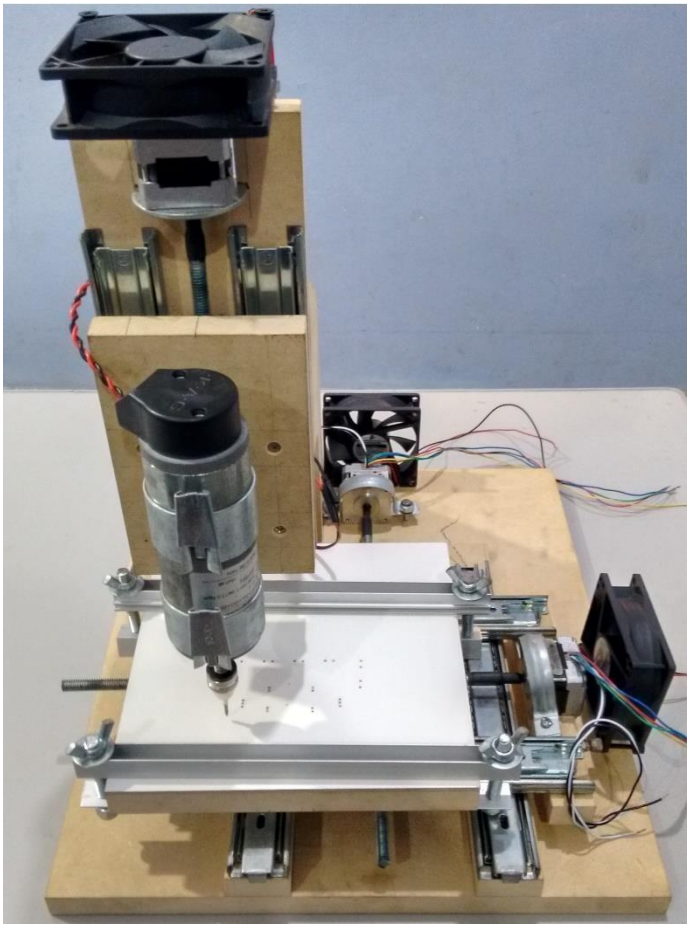


Figura 25 – Máquina fresadora CNC de baixo custo
Fonte: autoria própria

É possível verificar pela figura 26 a utilização de um *cooler* para a refrigeração forçada de cada motor de passo, sendo indispensável para evitar sobreaquecimento. Os motores de passo do projeto não se encontram colados na estrutura da máquina, mas sim pressionados através de abraçadeiras metálicas do tipo U nos eixos X e Y, e através de chapas metálicas no eixo Z.

O abandono da colação dos motores de passo para fixa-los é importante pois permite que sejam retirados com maior facilidade em caso de manutenções. Além disso, a utilização de componentes metálicos para seu suporte contribui na dissipação de calor do motor.

Por fim, para proteger a mesa de trabalho no processo de furação, foi adicionada uma chapa de madeira MDF de $2{,}5mm$ de espessura. As placas a serem usinadas são presas em duas extremidades acima da chapa com o auxílio de suportes de alumínio que são pressionados com um conjunto de quatro parafusos borboletas e peças complementares.

5.2 PROJETO DA PLACA DE CIRCUITO IMPRESSO

Conforme descrito na proposta de trabalho de conclusão de curso, a máquina fresadora CNC deve ser capaz de usinar qualquer placa de circuito impresso dentro das limitações físicas da máquina, sendo este fator, inclusive, o que torna a máquina uma CNC, podendo confeccionar diversos produtos de uma categoria somente com modificações de projeto em *softwares* CAD/CAM, a tornando uma máquina flexível.

Por via de simulação, qualquer placa projetada de acordo com o previsto na proposta atende ao código criado para o controle da CNC. Entretanto, o projeto metódico de uma placa de circuito específica se faz necessário para validar o trabalho pois, são inúmeros os fatores limitadores no projeto de baixo custo, podendo ser citado o próprio sistema mecânico que, mais do que o código de controle e o modo de acionamento dos motores de passo, tem a maior influência na precisão e na qualidade da placa usinada; o tipo da fresa, com seu formato, diâmetro e material, aliado à velocidade disponível no motor CC, possui também grande influência no desbaste da placa, limitando a largura das trilhas e a velocidade de avanço da CNC; por fim, um último fator, mas não menos relevante, é o fato da existência de limitação na área disponível para a criação de placas de circuito no *software* Eagle em sua versão livre, sendo tal limitação correspondente a uma área máxima de $100mm\ x\ 80mm$, influenciando diretamente a quantidade de componentes possíveis para a placa, assim como a largura das trilhas.

Portanto, considerando os atenuantes conhecidos e encontrados durante a elaboração do trabalho, foi projetada uma placa de circuito impresso no *software* Eagle que atendesse às limitações e pudesse validar o trabalho de conclusão de curso tanto de forma simulada quanto implementada.

O sistema de refrigeração forçada para os motores de passo e para os reguladores de tensão se constitui basicamente de 4 *coolers,* sendo um circuito simples e possível de ser aplicado no projeto de placa padrão para o trabalho.

Como forma de aproveitar ao máximo o espaço disponível e a funcionalidade da placa, foram incluídos também outros componentes, como os reguladores lineares de tensão LM78 e seus capacitores, e conectores que, além de receber a alimentação da fonte CC para todo o sistema fornecem tensão para o motor CC e também alimentam a segunda placa composta pelos circuitos *drivers* dos motores de passo.

Foi criada também uma malha de terra envolvendo todo o circuito. Como todos os componentes da placa possuem pelo menos um terminal conectado ao terra, ou *GND*, a criação de uma malha facilita a conexão dos componentes e otimiza a placa como um todo, já que diminui consideravelmente a quantidade de trilhas do circuito, reduzindo assim os dados exportados e permitindo que a placa seja usinada num período de tempo menor.

Através dos diversos recursos para confecção de placas do *software* Eagle, a placa foi projetada como ilustra a figura 26.

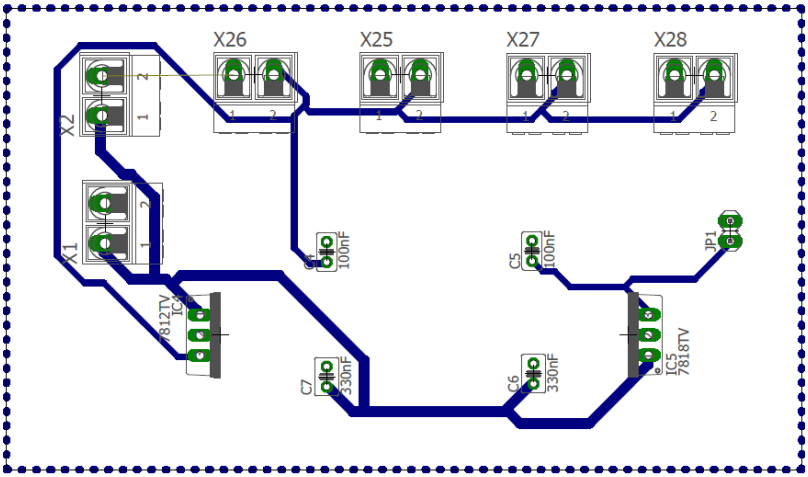


Figura 26 – Placa de circuito projetada
Fonte: autoria própria no *software* Eagle

Primeiramente, à questão de usinagem, a disposição dos componentes na placa foi planejada de como que eles ficassem espaçados o suficiente para que o roteamento ocorresse de maneira satisfatória levando em consideração as limitações já descritas anteriormente.

Considerando o diâmetro máximo da fresa utilizada como sendo de aproximadamente $0{,}8mm$, as distâncias mínimas entre as trilhas, configuradas no parâmetro *clearance* do *software*, foram definidas em $47mil$, valor necessariamente maior do que o agora citado, como mostra a figura 27.

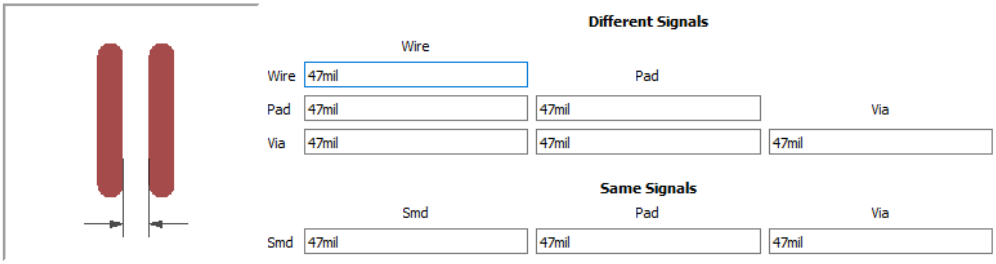


Figura 27 – Distâncias mínimas para a placa projetada
Fonte: Configuração no *software* Eagle

A distância entre as trilhas deve ser sempre maior do que o diâmetro da fresa quando em desbaste pois, como a fresa retira material da placa de forma a manter o cobre para a trilha entre dois caminhos fresados, as trilhas usinadas não podem ter distância menor, senão a fresa desbasta uma área onde estaria projetada a passagem de uma trilha.

Outro ponto importante do projeto da placa diz respeito às larguras das trilhas. Entre outros fatores, a largura de uma trilha deve ser projetada para permitir que a corrente do circuito possa circular normalmente, sem comprometer a capacidade do cobre da placa e o funcionamento do circuito.

Conhecendo os valores de corrente do circuito e levando em conta as relações de larguras de trilhas de acordo com a corrente do circuito, visto no quadro 2, foram definidas as larguras das trilhas para a placa.

Quadro 2 – Relações de largura de trilhas por capacidade de corrente

Largura da trilha	Corrente para cobre = 0,0347 mm	Corrente para cobre = 0,0694 mm
5 mil	500 mA	700 mA
10 mil	800 mA	1,4 A
20 mil	1,4 A	2,2 A
30 mil	1,9 A	3 A
50 mil	2,5 A	4 A
100 mil	4 A	7 A

Fonte: Adaptado de KRUG, Rodrigo (2010, p. 41)

Observa-se pelo quadro 2 que as relações mostradas dependem da espessura de cobre da placa de fenolite. Como a espessura do cobre da placa do projeto é desconhecida, considerou-se como sendo o pior caso, ou seja, sendo menos espessa e possuindo menos condutor. Dessa forma, foi utilizada como parâmetro a coluna de cobre com espessura de $0{,}0347mm$, sendo esta a menor espessura para o cobre de placas de fenolite comercializadas.

Assim, foram definidas as larguras das trilhas para a placa projetada. Como observa-se por seu circuito, nota-se que existem duas larguras distintas de trilhas. Como mencionado anteriormente, as larguras devem permitir com que as trilhas possam suportar as correntes do circuito, todavia, esse não é o único motivador da escolha.

Devido às limitações práticas do trabalho, não é viável o projeto de uma trilha muito fina pois durante o desbaste a fresa pode retirar o seu cobre. Com isso, optou-se como largura mínima o valor de $30mil$, como mostra a figura 28, correspondendo a $0{,}762mm$, o que garante a correta confecção da trilha e suporta em larga escala a necessidade de corrente para as partes do circuito que levam tal largura.

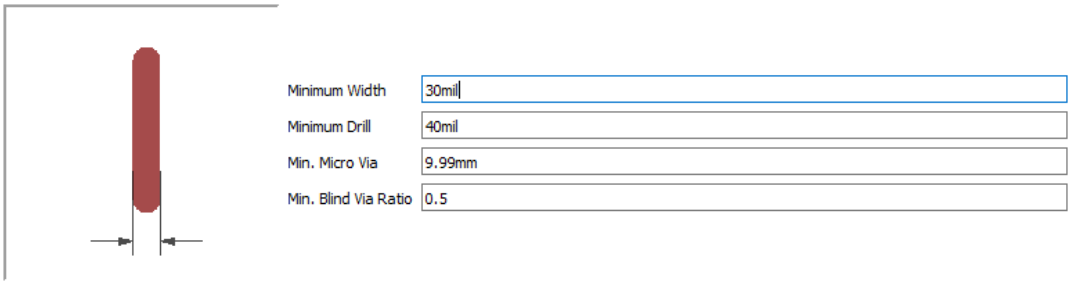


Figura 28 – Largura mínima das trilhas
Fonte: Configuração no *software* Eagle

Já para certa parte do circuito, definiu-se uma nova classe de condutor com maior largura, como mostra a figura 29. O valor escolhido de $50mil$, correspondente a $1{,}27mm$ possui capacidade de corrente de $2{,}5A$ e se faz necessário pois tal parte do circuito pode receber correntes de até $2A$ durante o funcionamento da máquina fresadora CNC.

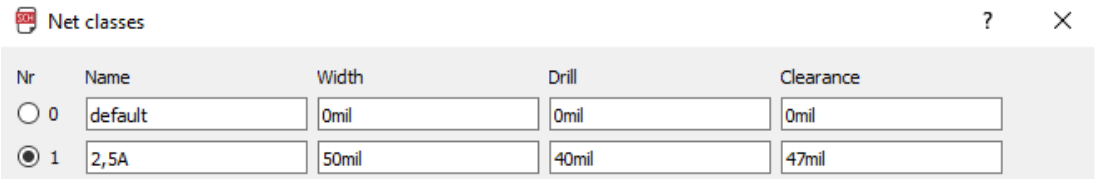


Figura 29 – Largura para a trilha de maior capacidade de corrente
Fonte: Configuração no *software* Eagle

Por fim, como último critério de projeto, a disposição dos componentes foi escolhida de modo a facilitar as conexões com os vários equipamentos já citados aos quais a placa fornece alimentação. Além disso, com maior importância, os CIs LM78 foram espaçados entre si em $56mm$ devido à necessidade de refrigeração forçada por meio de um dos *coolers* do projeto. Tendo o *cooler* dimensões de $80mm\ x\ 80mm$, a partir da distância projetada, ambos componentes podem ser resfriados adequadamente, não comprometendo então seus funcionamentos.

5.3 EXPORTAÇÃO E ADEQUAÇÃO DO CÓDIGO G

A exportação do código G se dá ainda através do *software* Eagle, utilizando a ULP (*User Language Program*), Linguagem de Programação do Usuário, em português *PCB-GCODE*

Esta ULP de licença livre criada por John T. Johnson possui mais de 10 anos de desenvolvimento e contribuição de usuários e é largamente utilizada para exportação de códigos G em projetos de placas de circuito impresso para máquinas CNC através do *software* Eagle. Como um pós-processador CAD/CAM, basicamente, a ULP avalia os dados de coordenadas e as configurações da placa projetada e exporta o código G correspondente.

A ULP oferece diversas opções de exportação de dados, dentre as quais foram utilizadas as mais importantes ao projeto. As figuras 30 e 31 mostram a seleção inicial de opções para o código G exportado.

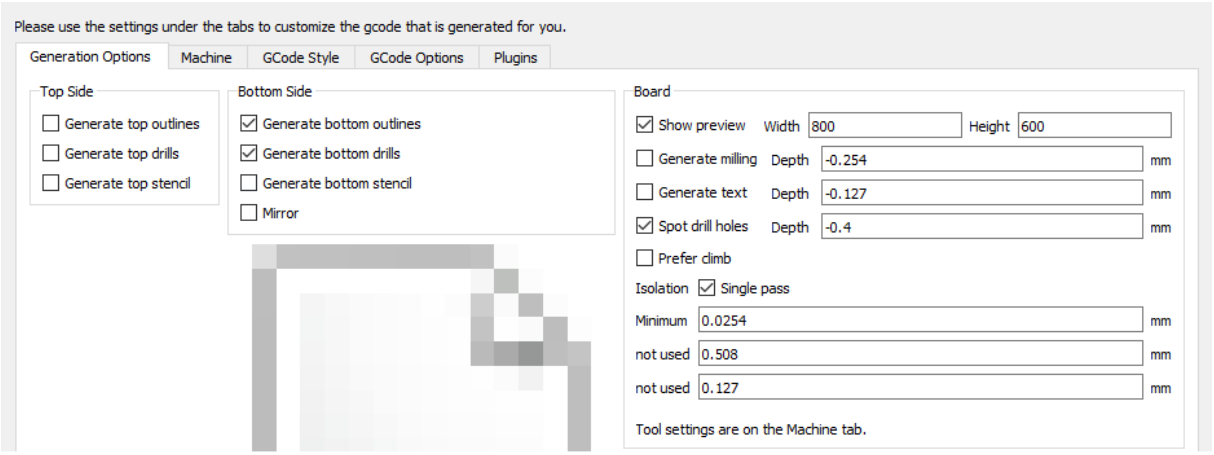


Figura 30 – Opções de geração da placa de circuito
Fonte: ULP PCB-GCODE

A figura 30 apresenta as opções para a geração da placa de circuito projetada. Dentre as variadas opções, a ULP permite a exportação de dados para placas que possuam trilhas em ambas as faces da placa de fenolite, porém, como no projeto da placa somente um lado cobreado é utilizado, foi selecionada a opção de gerar somente os dados das trilhas e furos do lado inferior da placa, ilustrado nas seleções da opção *Bottom Side*.

Já na opção *Board*, *Spot drill holes* e *Single pass* são selecionadas a fim de, respectivamente, marcar inicialmente os furos para os componentes com a ferramenta

fresa antes de furar de fato a placa com a broca e executar somente uma única passagem da fresa pela placa para usinar o circuito impresso.

A primeira opção selecionada é importante pois, ao iniciar os furos primeiramente com a fresa, cria-se uma ranhura no cobre que guia a broca posteriormente na execução final dos furos, evitando com que ela se desvie do ponto correto de furação. Portanto, com a opção escolhida, definiu-se a marcação de furos na coordenada $Z - 0.4$, ou seja, numa profundidade de $-0.4mm$ na placa.

A segunda opção é selecionada pois uma única passagem da fresa é suficiente para desbastar a placa e criar as trilhas para o circuito.

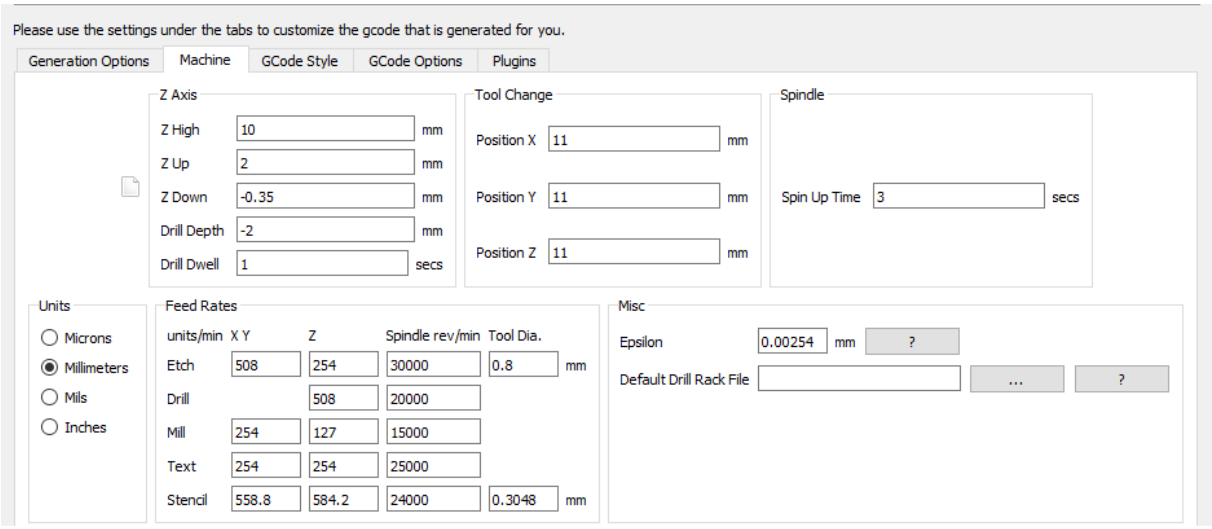


Figura 31 – Opções diversas de usinagem
Fonte: ULP PCB-GCODE

A figura 31 mostra a aba *Machine* que é utilizada para configurar as alturas do eixo Z, no tópico *Z Axis*, a unidade de exportação de coordenadas, em *Units* e o diâmetro da ferramenta fresa em *Tool Dia*. As outras configurações padrões vistas, como velocidade do *spindle*, posição dos três eixos para troca de ferramenta, etc., não são utilizadas, pois não fazem parte do projeto, sendo desconsideradas após a exportação do código G.

A unidade de medida definida para a exportação do código G é o milímetro e o diâmetro máximo da fresa durante a usinagem foi medido em aproximadamente $0.8mm$, sendo então assim ajustado.

As coordenadas configuradas para o eixo Z, sempre em milímetros e em relação à referência zero do plano cartesiano sendo a superfície da placa de fenolite, se referem a etapas diferentes no processo de usinagem da placa. *Z High* diz respeito

à altura máxima deslocada pelo eixo Z para posicionar o mandril de forma a se tornar possível a troca manual da fresa para a broca para iniciar a execução da furação.

Já a configuração *Z Up* se refere à altura que a fresa permanece enquanto ocorre a interpretação do comando G00, ou, movimentação rápida, quando o motor CC se mantém parado e os motores de passo dos eixos X e Y se deslocam para outra coordenada na maior velocidade ajustada a fim de retornar ao processo de fresagem noutro ponto.

De outra maneira, *Z Down*, configurado em $-0.35mm$ é a altura da fresa enquanto ocorre a interpretação do comando G01, ou, interpolação linear. Assim, no momento de desbaste da placa para a criação das trilhas, a fresa se encontra numa profundidade de $-0.35mm$ na fenolite.

Como última opção considerada na aba do eixo Z, *Drill Depth* ajusta a profundidade dos furos na placa. Como a espessura encontrada nas placas possui menos de $2mm$, essa profundidade se mostrou suficiente para que a furação as atravessasse completamente.

Por fim, feitas as configurações, a ULP foi executada para a exportação do código G para a placa. O código G é então exportado em formato *.txt* para que possa ser avaliado e utilizado, como apresenta a figura 32.

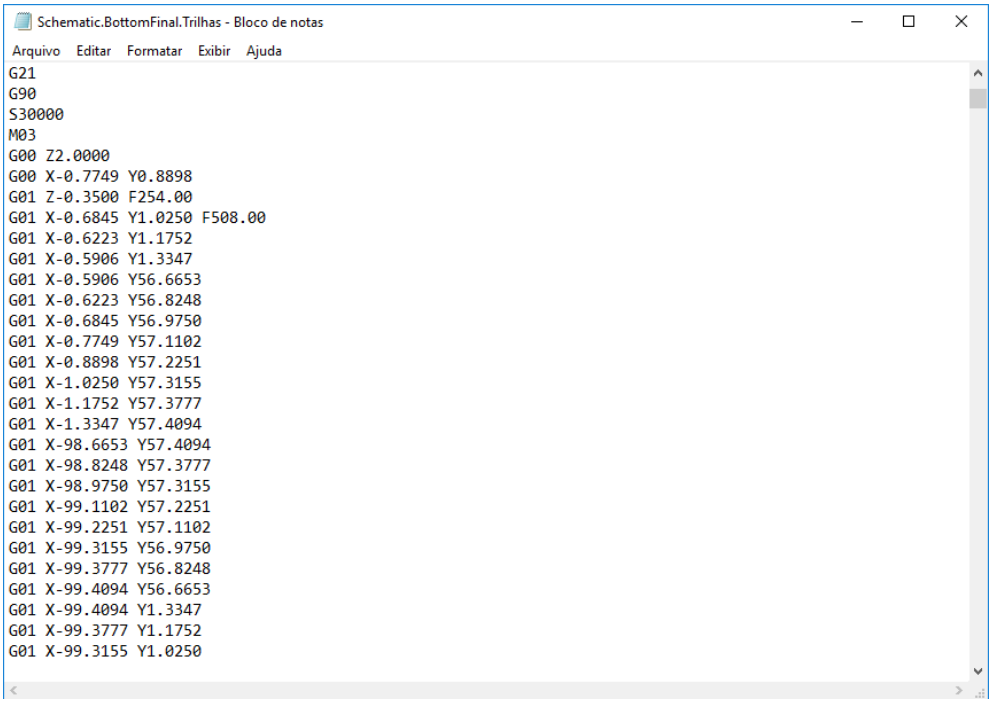


Figura 32 – Código G original exportado
Fonte: ULP PCB-GCODE

Para o código G ser corretamente interpretado pelo código de interpretação e controle criado no Code Composer Studio (CCS), o código G exportado originalmente necessita de adequações de dois tipos.

Primeiramente, é preciso desconsiderar os comandos irrelevantes para o projeto que se encontram na exportação original, como por exemplo *G21*, *G90*, *S3000* e *M03*. Ressalva-se que o comando *F* referente à velocidade de avanço dos eixos também é desconsiderado, pois, na elaboração do código de controle, o avanço dos eixos foi definido de maneira arbitrária baseando-se nas limitações do projeto.

Assim, o primeiro tipo de adequação modifica o código G exportado de modo a ser compatível com o código de interpretação elaborado no trabalho, sendo considerados então somente os comandos mostrados no quadro 3.

Quadro 3 – Códigos G implementados

Código G	Descrição	Parâmetros
G00	Movimentação rápida	Coordenadas (mm)
G01	Interpolação linear	Coordenadas (mm)
G04	Aguardar	Tempo (s)
G28	Retorno à referência	Coordenadas (mm)

Fonte: autoria própria

Já o segundo tipo de adequação leva em consideração o formato de entrada de dados em linguagem C do *software* CCS. Para serem corretamente interpretados, os dados, tanto do tipo *char* quanto do tipo *float* precisam ser compatíveis com a linguagem de programação do *software*.

As seguintes linhas, 33 e 44, extraídas do código do trabalho, encontrado na íntegra no Apêndice A, mostram os formatos requeridos para os dois tipos citados.

```
33 float32 EZ[FLOAT_SIZE] = {2, 2, -0.35,};
44 char *GCODE[] = {"G00", "G00", "G01",};
```

Avaliando a linha 33, observa-se que o vetor do tipo *float* denominado *EZ* e de tamanho *FLOAT_SIZE* requer como entrada de dados valores decimais separados por vírgula (,), o que delimita cada posição do vetor. Além disso, a letra indicativa do eixo, no caso, o eixo Z, não faz parte do tipo definido e é então excluída.

Já na linha 44, observa-se que o vetor do tipo *char* denominado *GCODE* e de tamanho vazio ou, variável de acordo com a quantidade de comandos de código G

inseridos, requer como entrada de dados *strings*, ou seja, um conjunto de caracteres e que devem possuir aspas (“”) e vírgula (,) como delimitação de posições.

Com isso, é preciso executar um segundo tipo de adequação no código exportado originalmente. Dada a grande quantidade de linhas de código G exportadas, chegando a milhares, é inviável a adequação de todo o código de maneira individual.

Logo, foi criado um processo de adequação do código G exportado por meio do *software* Excel e, assim, é possível adequar completamente o código de forma simples e rápida.

Para isso, foi criado o seguinte procedimento de adequação.

i. Copiar o código G original completo;
ii. Colar o código G no Excel usando o assistente de importação de texto;
iii. Com a ferramenta *Localizar e Substituir*, excluir os comandos irrelevantes ao projeto;
iv. Ainda na ferramenta anterior, readequar os comandos utilizados para entradas do tipo *char* e *float*;
v. Fazer adequações finais.

Após a cópia do código completo no procedimento *i.*, com o assistente de importação de textos é possível delimitar o código G para ser modificado com sucesso, conforme é visto pela figura 33.

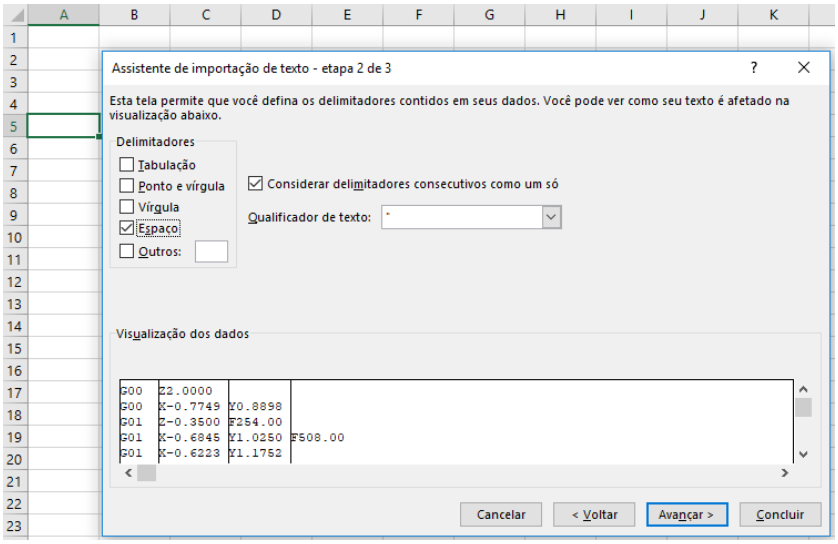


Figura 33 – Etapa ii. do procedimento de adequação
Fonte: *Software* Excel

Com o código G já delimitado, performam-se as etapas *iii.* e *iv.* do procedimento, como mostram as figuras 34 e 35, respectivamente.

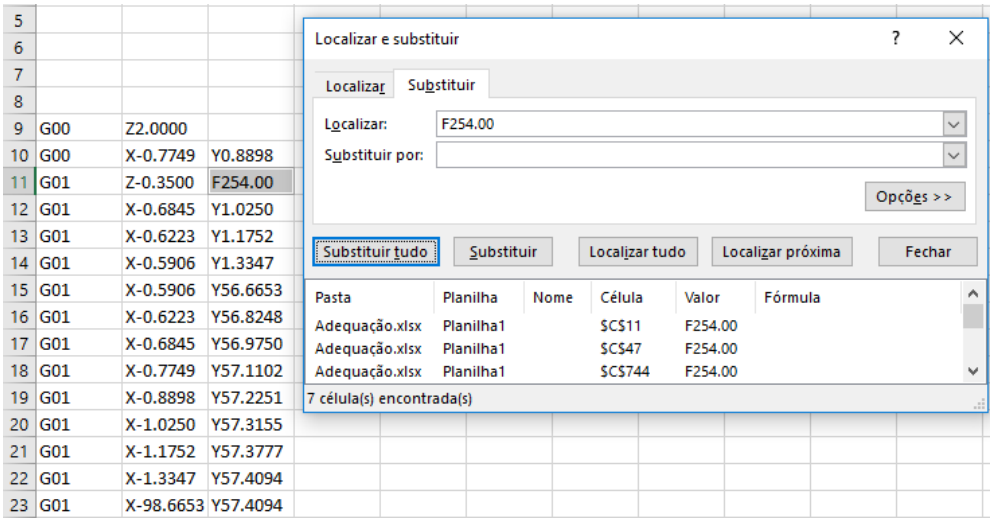


Figura 34 – Etapa iii. do procedimento de adequação
Fonte: *Software* Excel

Conforme ilustra a figura 34, é possível substituir todos os comandos irrelevantes ao projeto de forma conjunta e rápida.

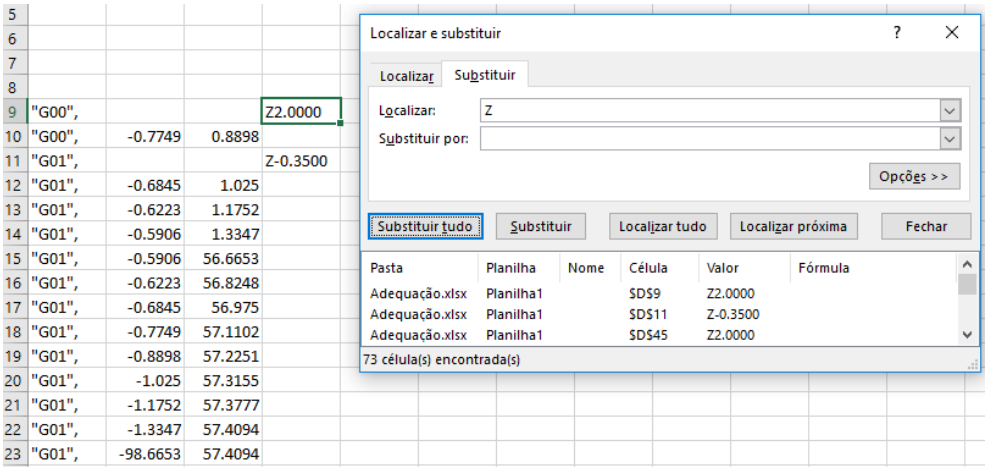


Figura 35 – Etapa iv. do procedimento de adequação
Fonte: *Software* Excel

A figura 35 ilustra a etapa *iv.* de adequação dos comandos e coordenadas para as entradas do tipo *char* e *float* no CCS.

Por fim, após as adequações finais provenientes da etapa *v.*, como por exemplo a inserção de vírgulas como delimitadores de posição para as coordenadas,

sendo feita também de forma conjunta para todas as células, o código G se encontra pronto para ser somente copiado e colado nos vetores criados no código de interpretação no CCS. O procedimento de adequação criado, de sua primeira até sua última etapa leva menos de 2 minutos para a placa projetada.

A figura 36 mostra o código G adequado pronto para ser inserido no código de interpretação.

	A	B	C	D	E	F	G	H	I	J	K
1	GCODE	"G00",	"G00",	"G01",	"G01",	"G01",	"G01",	"G01",	"G01",	"G01",	"G01",
2	EX	0,	-0.7749,	-0.7749,	-0.6845,	-0.6223,	-0.5906,	-0.5906,	-0.6223,	-0.6845,	-0.7749,
3	EY	0,	0.8898,	0.8898,	1.025,	1.1752,	1.3347,	56.6653,	56.8248,	56.975,	57.1102,
4	EZ	2,	2,	-0.35,	-0.35,	-0.35,	-0.35,	-0.35,	-0.35,	-0.35,	-0.35,

Figura 36 – Código G adequado
Fonte: *Software* Excel

Finalizadas a exportação e adequação do código G, a placa pode ser validada através de simulações, como mostra a figura 37.

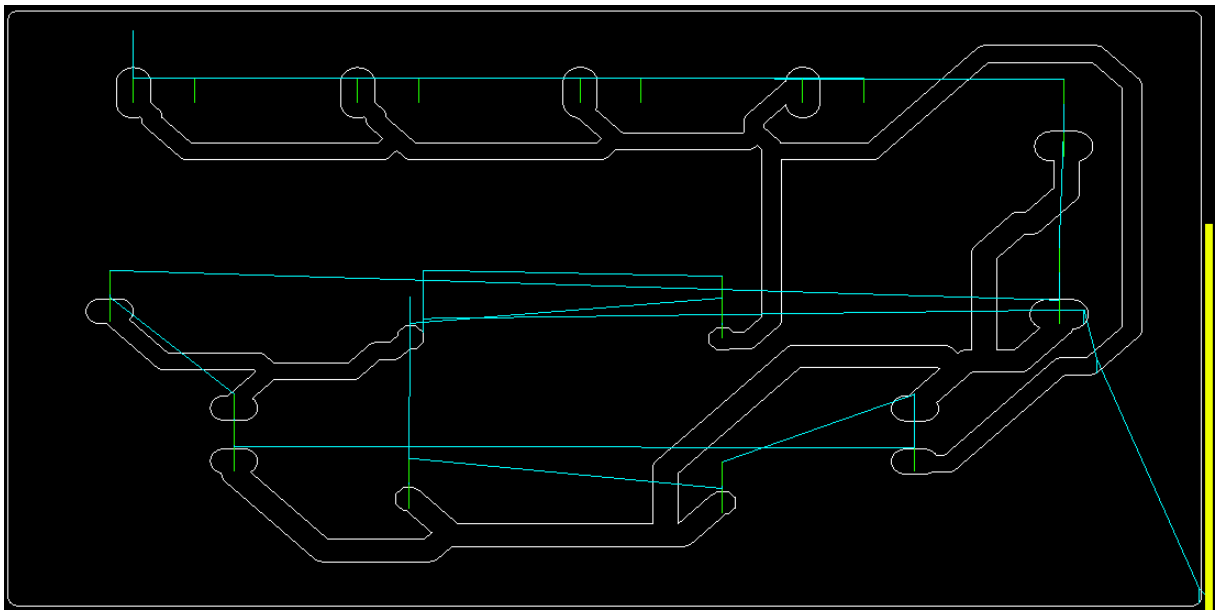


Figura 37 – Placa projetada espelhada simulada
Fonte: Extraída do *software Universal Gcode Sender*

Com a simulação em 3D realizada no *software* livre *Universal Gcode Sender* é possível verificar a disposição das trilhas projetadas e observar os caminhos percorridos pela fresa.

Cada linha tem um significado distinto de acordo com sua cor. O traço amarelo indica a coordenada inicial da fresa, enquanto os traços noutras cores mostram seu comportamento dinâmico. As linhas azuis indicam os caminhos percorridos em sequência pela máquina fresadora durante o comando *G00*. Os traços brancos

caracterizando o desenho da placa projetada mostram os caminhos percorridos pela fresa enquanto o comando *G01* está em vigor. Já as retas verdes apontam as coordenadas nas quais ocorre a furação, mostrando também a profundidade projetada dos furos.

Por fim, é possível comparar a simulação dos dados exportados com a placa projetada espelhada, sendo vista na figura 38.

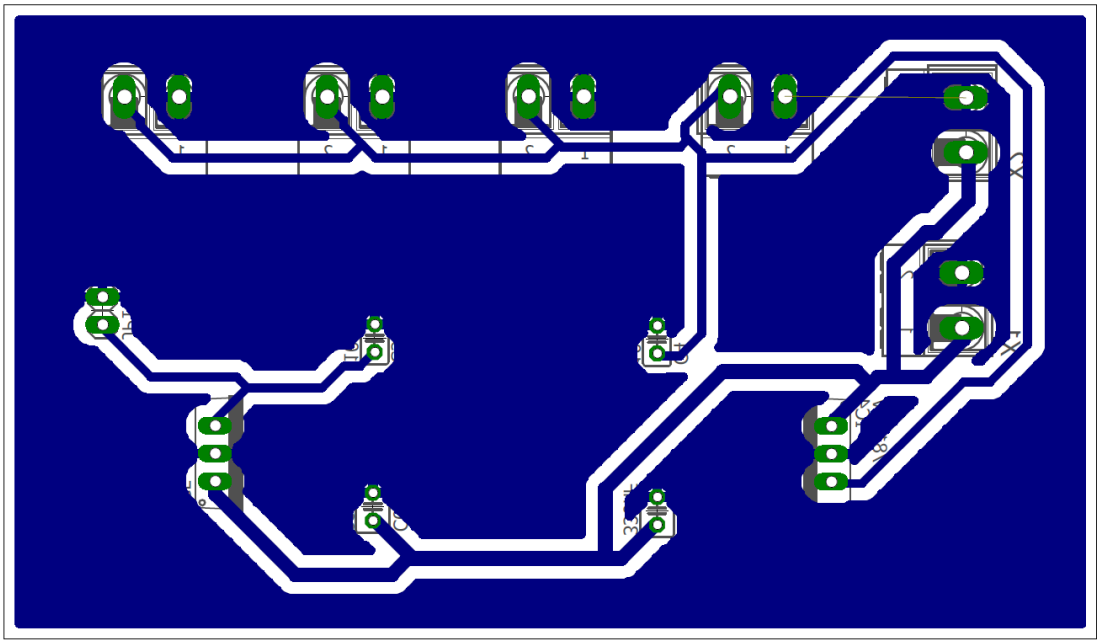


Figura 38 – Placa projetada espelhada
Fonte: autoria própria no *software* Eagle

5.4 DESENVOLVIMENTO DO CÓDIGO INTERPRETADOR

Nesta seção são apresentadas as partes mais importantes do código interpretador desenvolvido, que se encontra na íntegra no Apêndice A do trabalho.

O código de interpretação e controle do servossistema é criado em programação em linguagem C/C++ por meio do *software* Code Composer Studio e tem como base os dados em código G a serem interpretados, além dos componentes constituintes da fresa CNC, como o sistema mecânico e os motores de passo a serem controlados.

As linhas iniciais do código, de 1 a 21, correspondem às bibliotecas, configurações iniciais e variáveis necessárias para o controle do servossistema.

```
1 #include "F28x_Project.h"
2 #include <math.h>
3 #include <string.h>
4
5 //Configurações iniciais
6 void Desabilita_WDT(void); //Desabilita o WDT
7 void Ini_gpio(void); //Inicialização das portas I/O
8 void Ini_pwm(void); //Configuração do ePWM2
9 void Config_cpu(void); //Configuração da CPU
10 interrupt void epwm2_interrupt(void); //Interrupção do ePWM2
11
12 unsigned int periodo = 9687; //Período para PWM de 10kHz
13
14 Uint32 k, u, step, RefX, RefY;
15 float dx, dy, dz, DX1, DX2, DY1, DY2, DZ1, DZ2, DX45, DY45, l = 0.0002204861, /*l: distância (mm) percorrida
16 pela barra de rosca a cada passo em 1/32 microstep. Baseada no tipo da barra de rosca (18 fios por polegada,
17 percorrendo 1,41 mm a cada giro completo da barra)*/
18 PosATX, POSATX, PosATY, POSATY,
19 counterCWX = 0.0, counterCCWX = 0.0, counterCWY = 0.0, counterCCWY = 0.0, counterCWZ = 0.0,
20 counterCCWZ = 0.0;
21 int v, n, Nada;
```

Para o compilador poder receber as informações referentes ao microcontrolador específico utilizado na programação, a primeira linha do código inclui a biblioteca *F28x_Project.h*, a qual, entre outras configurações, define os tipos de dados de entrada para a programação e as estruturas de dados dos periféricos dos microcontroladores F28x, dos quais pertence o microcontrolador C2000 TMS320F28377S utilizado no trabalho através de seu kit de desenvolvimento LaunchPad. Sendo assim, o código programado se torna então compatível com o microcontrolador alvo, podendo ser desenvolvido.

As linhas 2 e 3 do código incluem, respectivamente, bibliotecas para operações matemáticas e *strings* utilizadas durante a programação do código.

Entre as linhas de código 5 e 10 encontram-se as declarações das configurações iniciais dos módulos e periféricos utilizados no projeto, como por exemplo, as portas de entrada e saída GPIO que fazem interface com os circuitos *drives* DRV8825 e os motores de passo e o módulo de PWM utilizado para o acionamento do motor CC, tendo seu período descrito na linha 12.

Já nas linhas de código restantes, de 14 a 21, as variáveis utilizadas no controle do servossistema são criadas. O uso de cada variável é explanado na sequência da descrição do código interpretador.

Para a elaboração do controle da máquina fresadora CNC, é necessário primeiramente conhecer as características do sistema mecânico utilizado que definem

o deslocamento linear a ser inserido no código e o modo de acionamento desejado para os motores de passo.

O sistema mecânico de baixo custo do projeto é composto, dentre outros itens, por barras de rosca do tipo $\frac{5}{16}$" *NC* de 18 fios por polegada. Assim, conhecendo os dados do mecanismo de movimentação dos eixos da máquina, é possível determinar o passo da fresa. A figura 39 ilustra o passo de uma barra dentada.

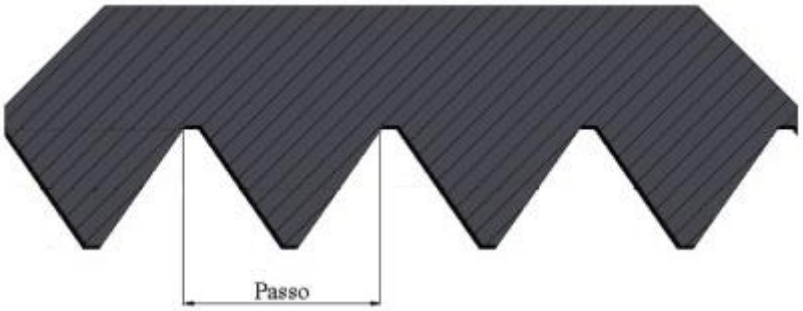


Figura 39 – Passo de barra de rosca
Fonte: Extraída de Microntek Ferramentaria (2018)

O passo de uma barra de rosca é a distância linear percorrida pela rosca quando esta performa um giro completo. O passo da barra depende de suas características construtivas. A barra de rosca utilizada é do tipo *NC*, ou, de rosca grossa e contém 18 fios por polegada, como já indicado. A figura 40 ilustra os fios de uma barra dentada.

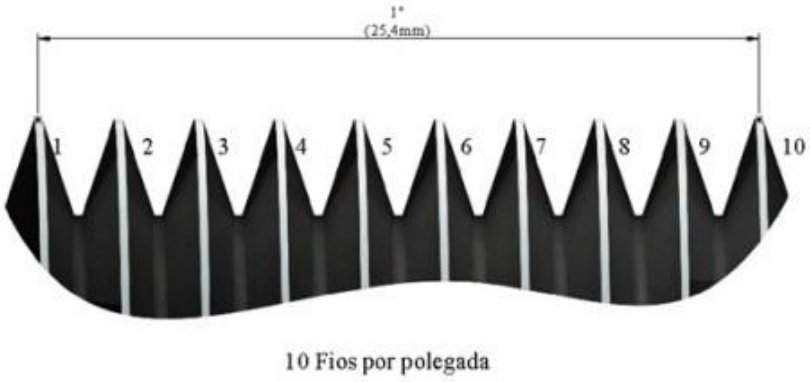


Figura 40 – Fios de barra de rosca
Fonte: Extraída de Microntek Ferramentaria (2018)

O fio de uma barra de rosca corresponde a cada dente da barra e é sempre referenciado a uma polegada de comprimento.

Logo, encontra-se o passo da barra de rosca pela equação 8.

$$p = \frac{25{,}4}{fios} = 1{,}41mm \quad (8)$$

A barra de rosca percorre assim $1{,}41mm$ a cada giro completo, porém, para definir a distância percorrida a cada passo do motor de passo, leva-se em consideração a graduação escolhida para o acionamento dos motores pelo *driver*.

Conforme já explanado, o acionamento em micro-passos dos motores de passo, entre outras melhorias, aumenta a resolução do sistema e faz com que os motores reduzam drasticamente sua trepidação, questão primordial na fresadora CNC que visa a precisão do item produzido. Como forma de se obter a melhor condição disponível, foi escolhida a graduação de passos em $\frac{1}{32}$ micro-passos ou, 32 micro-passos por passo. Isso significa que, para cada passo completo em *full step* de $1{,}8°$ de giro, o motor executará, na verdade, 32 passos.

Portanto, um passo unitário dado pelo motor corresponde a $0{,}05625°$ de giro. Visto que numa volta completa da barra ou, a cada $360°$, desloca-se uma distância de $1{,}41mm$, a cada $0{,}05625°$ desloca-se então $l = 0{,}0002204861mm$.

Tendo as configurações iniciais sido feitas, a etapa seguinte é programada para alocar e receber os dados de código G, sendo vista entre as linhas 23 e 41 do código.

```
23 #define FLOAT_SIZE 6000
24 float32 EZ[FLOAT_SIZE];
25 #pragma DATA_SECTION(EZ, ".COORD_TABLE");
26
27 float32 EX[FLOAT_SIZE];
28 #pragma DATA_SECTION(EX, ".COORD_TABLE");
29
30 float32 EY[FLOAT_SIZE];
31 #pragma DATA_SECTION(EY, ".COORD_TABLE");
32
33 float32 EZ[FLOAT_SIZE] = {};
34
35 float32 EX[FLOAT_SIZE] = {};
36
37 float32 EY[FLOAT_SIZE] = {};
38
39 float32 P[] = {10};
40
41 char *GCODE[] = {}; //Sequência dos comandos das linhas de código G
```

Para que todas as coordenadas dos eixos pudessem ser inseridas no código, foram criados vetores correspondentes a cada eixo da máquina, denominados de *EZ, EX* e *EY*. Definiu-se o tamanho de 6000 posições para cada vetor pois tal valor cobre de maneira considerável a quantidade de coordenadas exportadas para cada eixo, independentemente da placa projetada, pois, como já mencionado, o *software* CAD/CAM utilizado em sua versão livre possui limitação na área das placas projetadas.

Para a alocação dos vetores das coordenadas foi criada na seção de dados não inicializados na memória, no bloco *RAMGS*, uma seção de dados denominada *COORD_TABLE*, sendo vista também pela figura 41.

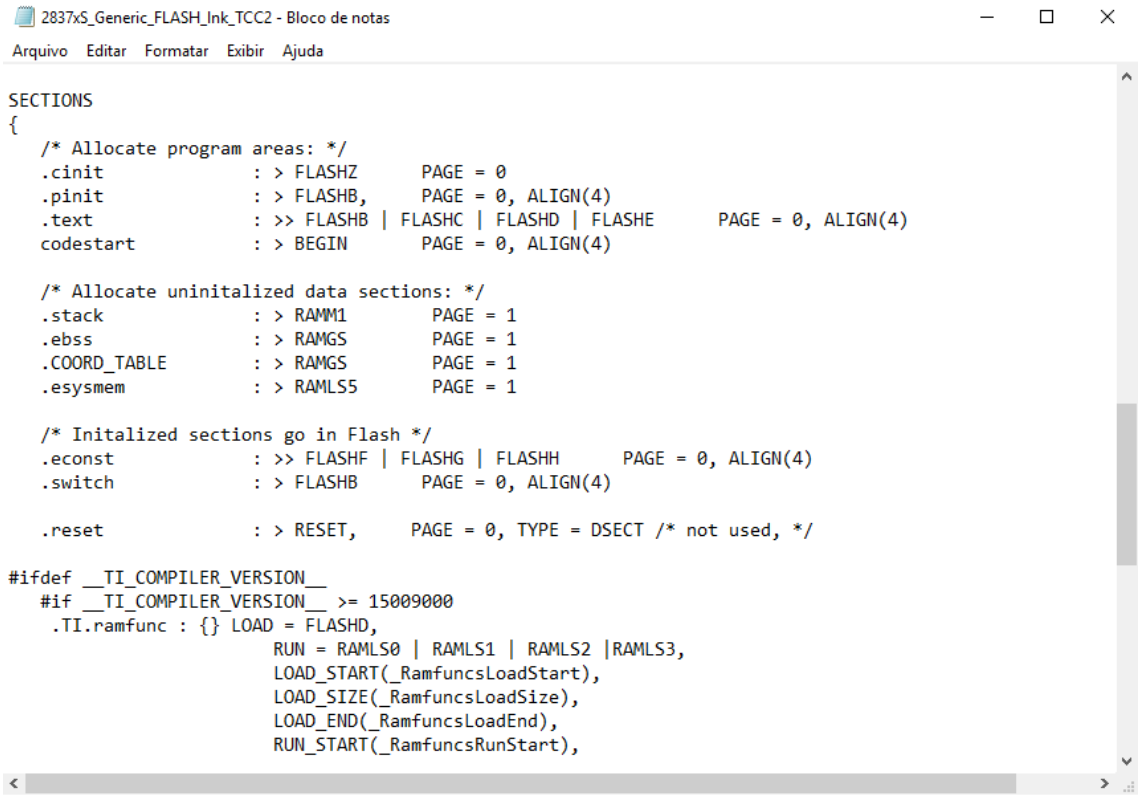


Figura 41 – Arquivo de *link* de memória
Fonte: Configurações no *software* ControlSUITE

A partir daí os vetores prontos para a inserção das coordenadas são definidos. Também são definidos os vetores *P* e *GCODE*. O primeiro citado recebe um valor arbitrário referente a tempo, em segundos, para parada dos motores em ocorrência da interpretação do código *G04.* O segundo recebe todos os comandos de código G referentes à placa projetada.

Nas linhas seguintes de código, já dentro da seção principal de programação *main*, são criadas as constantes de comparação utilizadas para a interpretação do código G.

```
43 void main(void){
44
45 n = (sizeof(GCODE)/sizeof(*GCODE)); //Cálculo da quantidade de linhas/comandos de código G
46
47 //Constantes para interpretação do código G
48 const char * const A[] = {"G00"};
49 const char * const B[] = {"G01"};
50 const char * const C[] = {"G04"};
51 const char * const D[] = {"G28"};
```

As constantes, indo de *A a D* são do tipo *char* constante com ponteiros constantes e apontam os tipos de comando de código G utilizados no projeto, vistos no Quadro 1. Com elas, é possível efetuar comparações com cada posição do vetor *GCODE* para interpretar o tipo de comando a ser executado pela CNC.

Entre essas linhas, nota-se também a exposição da variável *n*. Esta variável é utilizada posteriormente no laço *for* principal do código e como descrito em seu comentário de programação, recebe o valor do cálculo da quantidade de comandos de código G.

As linhas de código de 104 a 119 apresentam o início do laço *for* principal onde é feita a interpretação das *n* linhas de código G.

```
104 //Interpretação das n linhas de código G
105 for (v = 0; v < n; v++){
106
107 //Cálculo das distâncias a serem percorridas (mm)
108 if(v == 0){
109
110 dz = EZ[0] - 0;
111 dx = EX[0] - 0;
112 dy = EY[0] - 0;
113 }
114
115 else{
116
117 dz = EZ[v] - EZ[v-1];
118 dx = EX[v] - EX[v-1];
119 dy = EY[v] - EY[v-1];}
```

Para o cálculo das distâncias a serem percorridas por cada eixo, a primeira estrutura condicional *if* do código avalia a variável *v* que percorre todas as posições do vetor *GCODE* e dos vetores de coordenadas nas posições correspondentes.

Com isso, as variáveis *dz, dx* e *dy* se constituem de equações que definem o deslocamento linear dos eixos, sempre avaliando as coordenadas posteriores em relação às anteriores.

Corroborando com a interpretação do código G, a linha de código 122 mostra a função *strcmp* utilizada para fazer a comparação de cada posição do vetor *GCODE* com as constantes de comparação criadas.

```
121 //G00 = movimentação rápida, sem fresar (motor CC parado), motores de passo em RPM maior
122 if(strcmp (GCODE[v], A[0]) == 0){
```

Ao se verificar a condição verdadeira da comparação, as ações necessárias são programadas. A primeira comparação mostrada diz respeito ao comando *G00*, assim, deve-se manter o motor CC parado e configurar a frequência dos pulsos enviados para os motores de passo em maior valor possível, levando a maiores velocidades de avanço da fresa.

As linhas seguintes mostram o código que mantém o motor CC sem rotação, ajustando o ciclo de tarefa do PWM para zero e desabilitando as GPIO 18 e 19 do microcontrolador que controlam o acionamento do *drive* do motor.

```
124 //Motor CC parado
125 EALLOW;
126 EPwm2Regs.CMPA.bit.CMPA = 0;
127 EDIS;
128 GpioDataRegs.GPACLEAR.bit.GPIO18 = 1;
129 GpioDataRegs.GPACLEAR.bit.GPIO19 = 1;
```

Os pulsos enviados para os motores de passo dependem das distâncias a serem percorridas pelos eixos e também do sinal proveniente do resultado do cálculo. Assim, nas linhas à frente é feita a apuração do cálculo para poder controlar os pulsos corretamente.

```
135 //Eixo Z sempre se move primeiro para determinar altura da ferramenta antes das ações seguintes
136 if(dz > 0){
137 //Sentido horário
138 DZ1 = fabs(dz); //Valor absoluto de dz
139 GpioDataRegs.GPCCLEAR.bit.GPIO69 = 1;
```

```
140 step = DZ1/l; //Quantidade de passos necessários
141 counterCWZ = counterCWZ + DZ1; //Contador das distâncias percorridas (mm) no sentido horário
142 for (u = 0; u < step; u++){
143
144 DELAY_US(117.187); //f = 4,267kHz = 40 RPM
145 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
146 DELAY_US(117.187);
147 GpioDataRegs.GPASET.bit.GPIO11 = 1;
148 }
149 }
```

A verificação do sinal é feita para determinar o sentido de giro dos motores. Em caso de deslocamento positivo, como é condicionado pelo *if* na linha 136, o sentido de giro é definido como horário configurando o sinal enviado ao *driver* através da GPIO69 que controla o sentido de giro do eixo Z.

A variável *DZ1* converte a variável *dz* para valor absoluto pela função *fabs* ou, *float absoluto* fornecida pela biblioteca *math.h*. *DZ1* é diferenciado de *DZ2* com a finalidade de incrementar contadores diferentes, um deles somando às distâncias percorridas no sentido horário e outro no sentido anti-horário. Os contadores são utilizados no código como referência para o comando *G28* que visa o retorno da fresa à origem de usinagem da placa.

Em seguida, pela equação mostrada na linha 140 calcula-se a quantidade de passos necessários para percorrer o deslocamento para o eixo, onde o valor é recebido pela variável *step*. A linha 141 mostra a variável *counterCWZ*, contador das distâncias percorridas especificamente pelo eixo Z em sentido horário. Cada eixo possui seu próprio contador para ambos os sentidos.

Na linha 142 observa-se o laço *for* responsável pela geração de pulsos para os motores de passo. O laço gera um pulso para passo necessário para cumprir a distância calculada. A geração do trem de pulsos é criada a partir da função *DELAY_US* que é configurada com valores em microssegundos de acordo com a frequência/RPM desejada para o acionamento dos motores de passo.

As frequências de acionamento de cada eixo foram escolhidas baseando-se na maior integridade da estrutura de baixo custo da máquina fresadora CNC e também na qualidade de usinagem da placa. A fim de não causar possíveis danos nas estruturas que sustentam as porcas dos eixos da máquina, as frequências foram limitadas. Além disso, devido à relativa baixa rotação do motor CC para a aplicação, de pouco mais de $3150\ RPM$, em conjunto com a ferramenta fresa utilizada, observou-

se que uma frequência de acionamento mais elevada leva a um deslocamento do eixo da ferramenta durante o processo de usinagem, devido de fato à baixa capacidade de desbaste do conjunto motor CC mais a fresa.

Portanto, as frequências são definidas de forma a tornar o sistema estruturalmente mais seguro e proporcionar uma maior qualidade da placa usinada. Como no comando *G00* a fresa não se encontra em contato com a placa para desbaste, frequências mais elevadas podem ser usadas. Já em *G01*, definem-se frequências menores.

A frequência de acionamento dos motores de passo é dada pela equação 9.

$$fstep = \frac{vel * 360 * n_m}{60 * \theta_{step}} \tag{9}$$

Através da variável *vel* determina-se a velocidade desejada para motor, sendo no caso, de 40 RPM. A constantes de valor 360 e 60 referem-se, nessa ordem, ao ângulo dado num giro completo do motor e aos segundos por minuto. A variável n_m recebe o valor de 32 micro-passos por passo e, θ_{step}, o ângulo dado pelo motor a cada passo em *full step*, sendo de $1{,}8$. Logo, chega-se à frequência ajustada de $4{,}267\ kHz$.

Para configurar a função *DELAY_US*, determina-se o período da frequência ajustada, chegando-se então a aproximadamente $234{,}38\ \mu s$. Como a geração de pulsos é feita separadamente para os níveis alto e baixo da onda, o período é dividido por 2 na função para criar ambos os níveis dentro do período total da frequência ajustada e, assim encontra-se o valor de aproximadamente $117{,}187\ \mu s$.

A geração do trem de pulsos através da função *DELAY_US* não compromete o desempenho do motor de passo pois, a única restrição a ser cumprida para o acionamento do motor é o chaveamento dos transistores dentro de seu limite, conforme indica o quadro 4.

Quadro 4 – Limites de acionamento do *driver*

			MIN	MAX	UNIT
1	f_{STEP}	Step frequency		250	kHz
2	$t_{WH(STEP)}$	Pulse duration, STEP high	1.9		µs
3	$t_{WL(STEP)}$	Pulse duration, STEP low	1.9		µs

Fonte: Extraído de DRV8825 *Controller* (2014, p. 7)

Portanto, a uma dada frequência, estando a duração dos pulsos dentro dos limites estabelecidos, aciona-se o motor de passo devidamente. Além disso, as ondas geradas devem possuir o mesmo período entre os níveis alto e baixo, como mostra a figura 42, referenciada aos limites indicados no quadro 4.

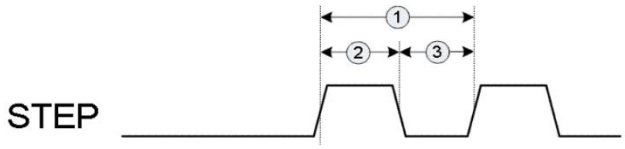


Figura 42 – Formato de onda do trem de pulsos
Fonte: Extraída de DRV8825 *Controller* (2014, p. 7)

Assim, entre as linhas 144 e 147 o trem de pulsos é gerado possuindo a frequência buscada.

O procedimento entre as linhas 136 e 149 para controlar o trem de pulsos enviado para o motor de passo é o mesmo para todos os eixos do sistema, somente mudando o valor da frequência de acionamento conforme os fatores citados anteriormente.

Com isso, a interpretação e controle dos motores de passo para o comando *G00* são efetuados. Para o comando seguinte, o procedimento se repete, com exceção ao acionamento do motor CC e à interpretação das trilhas oblíquas projetadas. As linhas de código a seguir apresentam o procedimento de acionamento do motor CC.

```
245 //G01 = interpolação, ação de fresar (motor CC acionado), motores de passo em RPM menor
246 else if(strcmp (GCODE[v], B[0]) == 0){
247
248 //Motor CC acionado
249 GpioDataRegs.GPASET.bit.GPIO18 = 1;
250 GpioDataRegs.GPASET.bit.GPIO19 = 1;
251 //Motor CC é acionado de forma suave (em 5s, incrementos de 1% de duty cycle)
252 while(EPwm2Regs.CMPA.bit.CMPA < periodo){
253 DELAY_US(0.05e6);
254 EALLOW;
255 EPwm2Regs.CMPA.bit.CMPA = (EPwm2Regs.CMPA.bit.CMPA + periodo*0.01);
256 EDIS;}
```

A primeira ação do interpretador ao se verificar a condição de interpolação é o acionamento do motor CC; neste momento, a ferramenta se encontra ainda fora do plano de contato com a placa de fenolite. O acionamento do motor se dá de forma

suave elevando o ciclo de tarefa do PWM em incrementos de 1%, sendo ajustado em 5 segundos a chegada à velocidade nominal.

O acionamento em rampa do motor CC é necessário devido a alta força de torção gerada num acionamento do tipo degrau, podendo prejudicar o sistema mecânico da máquina.

As trilhas oblíquas projetadas para a placa são padronizadas em $45°$ e, para sua interpretação, alguns passos são necessários. Uma reta com angulação de $45°$ forma um triângulo retângulo com catetos de valores iguais e, nessa situação, os deslocamentos *dx* e *dy* possuem o mesmo valor.

Entretanto, devido aos resultados dos cálculos feitos, os valores se tornam distintos nas últimas casas decimais. Dessa forma, é preciso fazer um condicionamento de valores utilizando a função *floorf* que volta a arredondar os valores permitindo que se tornem exatos e possam ser avaliados. As linhas seguintes mostram a aplicação da função.

```
263 DX1 = fabs(dx);
264 DY1 = fabs(dy);
265 DX45 = floorf(DX1 * 100) / 100; //Arredondamento necessário para a exatidão dos valores dos catetos
266 DY45 = floorf(DY1 * 100) / 100;
```

Por fim, com as distâncias condicionadas, as quatro possiblidades de retas oblíquas nos quadrantes cartesianos são consideradas e o controle dos passos de ambos os eixos é feito, como apresentam as linhas de código seguintes.

```
304 if(DX45 == DY45 && DX45 >= 0.1){
305 step = DX1/l;
306 //Interpolação de retas oblíquas padronizadas em 45°. Análise dos 4 quadrantes cartesianos.
307 if(dx > 0 && dy < 0){
308 GpioDataRegs.GPCSET.bit.GPIO65 = 1;
309 GpioDataRegs.GPCCLEAR.bit.GPIO66 = 1;
310 counterCWX = counterCWX + DX1;
311 counterCCWY = counterCCWY + DY1;
312 for (u = 0; u < step; u++){
313
314 DELAY_US(312.5); //15 RPM
315 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
316 DELAY_US(312.5);
317 GpioDataRegs.GPASET.bit.GPIO3 = 1;
318 DELAY_US(312.5);
319 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
320 DELAY_US(312.5);
321 GpioDataRegs.GPASET.bit.GPIO10 = 1;}}
```

O próximo comando de código G a ser interpretado é o *G04*. Nesta situação, todos os eixos da fresa, bem como o motor CC devem se manter estacionários. Isso deve ser feito para permitir a troca de ferramentas da CNC, onde a fresa é substituída por uma broca para que a furação da placa seja iniciada.

Para sinalização, o LED azul do kit LaunchPad que é conectado à porta GPIO13 é ativado sendo mantido até que a troca de ferramenta seja concluída.

```
447 //G04 = todos os motores parados por tempo arbitrário em segundos
448 else if(strcmp (GCODE[v], C[0]) == 0){
449
450 GpioDataRegs.GPADAT.bit.GPIO13 = 0; //Ativa LED azul
451 //Motor CC parado
452 EALLOW;
453 EPwm2Regs.CMPA.bit.CMPA = 0;
454 EDIS;
455 GpioDataRegs.GPACLEAR.bit.GPIO18 = 1;
456 GpioDataRegs.GPACLEAR.bit.GPIO19 = 1;
457 //Motores de passo parados
458 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
459 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
460 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
461 DELAY_US(P[0]*1e6);
462 asm(" ESTOP0"); //Pausa o debugger para troca de ferramenta
463 GpioDataRegs.GPADAT.bit.GPIO13 = 1; //Desativa LED azul
464 }
```

Por fim, o último comando interpretado é o *G28*. Este comando deve levar a fresa de volta à origem de usinagem da placa e, para isso, os contadores de distâncias percorridas dos eixos X e Y são validados. O eixo Z não requer retorno à referência pois no código G exportado esse eixo já tem sua posição readequada.

As linhas de 480 a 487 mostram o cálculo das coordenadas atuais nas quais os eixos se encontram no final da usinagem.

```
480 //Cálculo das coordenadas atuais (mm)
481 PosATX = counterCWX - counterCCWX;
482 POSATX = fabs(PosATX);
483 RefX = POSATX/l;
484
485 PosATY = counterCWY - counterCCWY;
486 POSATY = fabs(PosATY);
487 RefY = POSATY/l;
```

A variável *PosATX* recebe a coordenada final do eixo X e, após ser condicionada, é usada como parâmetro de referência para definir a quantidade de passos necessários *RefX*. O mesmo ocorre então para o eixo Y.

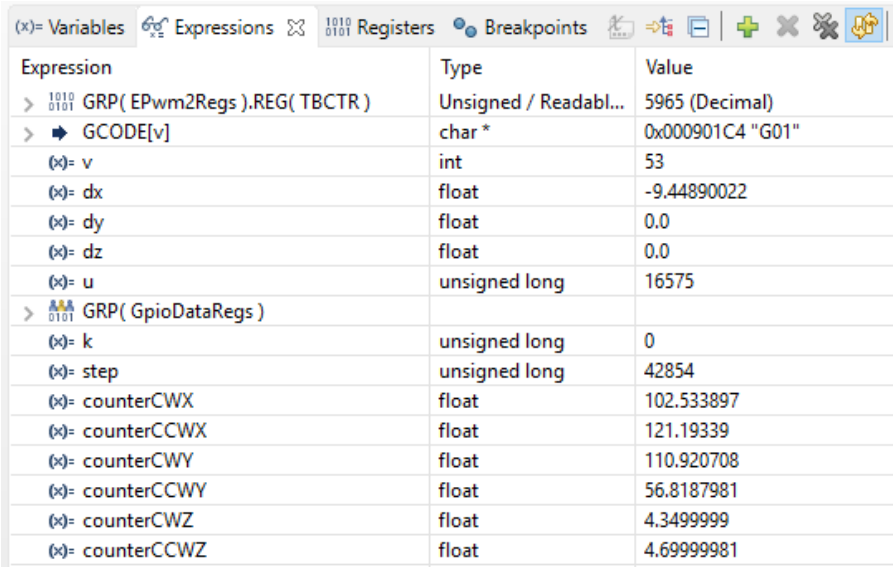
Posteriormente no c+ódigo, o sinal da primeira variável citada é averiguado a fim de determinar qual sentido deve ser ativado para retornar o eixo na posição certa e, assim, é possível levar a máquina à sua posição inicial.

6. RESULTADOS

6.1 VERIFICAÇÃO EXPERIMENTAL DA MÁQUINA FRESADORA CNC

Como primeira instância, a placa projetada no trabalho teve sua verificação de forma simulada a fim de garantir a exportação do código G adequado e sua correta interpretação, porém, para a validação do projeto desenvolvido, a verificação experimental da máquina se faz necessária.

A partir da inserção dos dados no código interpretador, a máquina é verificada experimentalmente. Durante o processo de usinagem é possível acompanhar em tempo real a interpretação dos dados por meio do *software* Code Composer Studio, como mostra a figura 43.



Expression	Type	Value
GRP(EPwm2Regs).REG(TBCTR)	Unsigned / Readabl...	5965 (Decimal)
GCODE[v]	char *	0x000901C4 "G01"
v	int	53
dx	float	-9.44890022
dy	float	0.0
dz	float	0.0
u	unsigned long	16575
GRP(GpioDataRegs)		
k	unsigned long	0
step	unsigned long	42854
counterCWX	float	102.533897
counterCCWX	float	121.19339
counterCWY	float	110.920708
counterCCWY	float	56.8187981
counterCWZ	float	4.3499999
counterCCWZ	float	4.69999981

Figura 43 – Variáveis interpretadas em tempo real
Fonte: Extraída do *software* CCS

Observa-se pela figura 43 algumas das variáveis criadas no código que podem ser avaliadas durante a implementação do projeto.

É possível notar que o código se encontra em processo de interpretação da posição 53 do vetor *GCODE*, o qual, como indicado também, se refere ao comando *G01* e, portanto, a fresa se encontra em desbaste da placa.

A variável *dx* indica que o motor de passo do eixo X se encontra em movimento estando em sentido anti-horário devido ao sinal negativo da distância a ser percorrida. Assim, ao final da posição interpretada, a fresa deve percorrer em torno de $9{,}45mm$ num dado sentido.

Nota-se também pela variável $step$ a quantidade de passos necessários para que o deslocamento seja feito e, pela variável u o número de passos já dados até o momento.

Após a finalização da interpretação das *n* linhas de código G, a placa projetada tem sua usinagem concluída e pode ser averiguada. A figura 44 mostra a placa recém usinada.

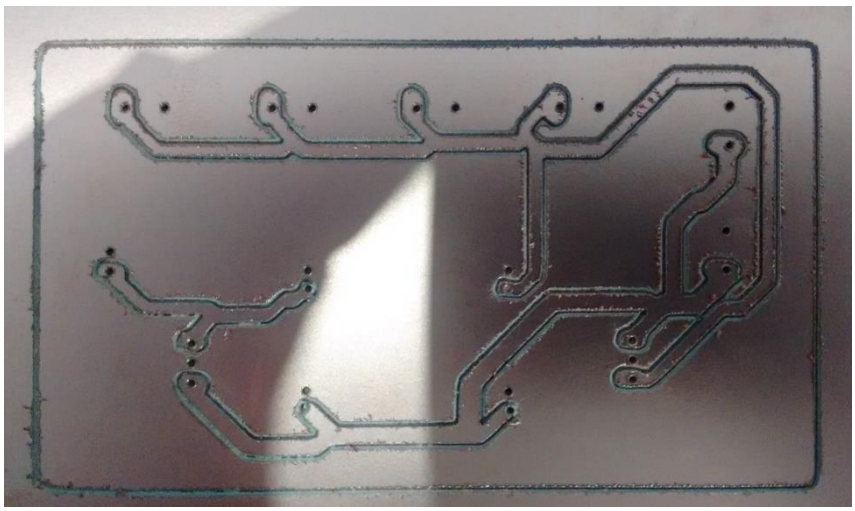


Figura 44 – Placa projetada recém usinada
Fonte: autoria própria

Na placa de circuito impresso recém confeccionada observa-se a existência de cavacos devido ao processo de desbaste. Para o melhor acabamento e funcionalidade do circuito, a placa é acabada utilizando-se uma lixa de papel, chegando ao resultado final, visto na figura 45.

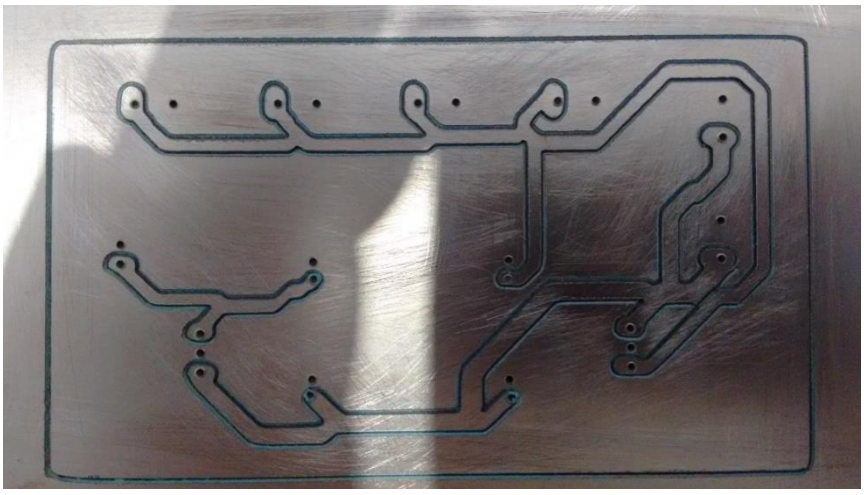


Figura 45 – Placa projetada acabada
Fonte: autoria própria

Feito o acabamento da placa, pode-se avaliar as trilhas confeccionadas para verificar a compatibilidade com a placa projetada. Além da semelhança notável no circuito confeccionado, é possível perceber as diferenças de espessuras que foram projetadas para trilhas distintas. Por fim, após a validação da placa confeccionada também através da verificação de continuidade das trilhas, os componentes são inseridos, conforme mostra a figura 46.

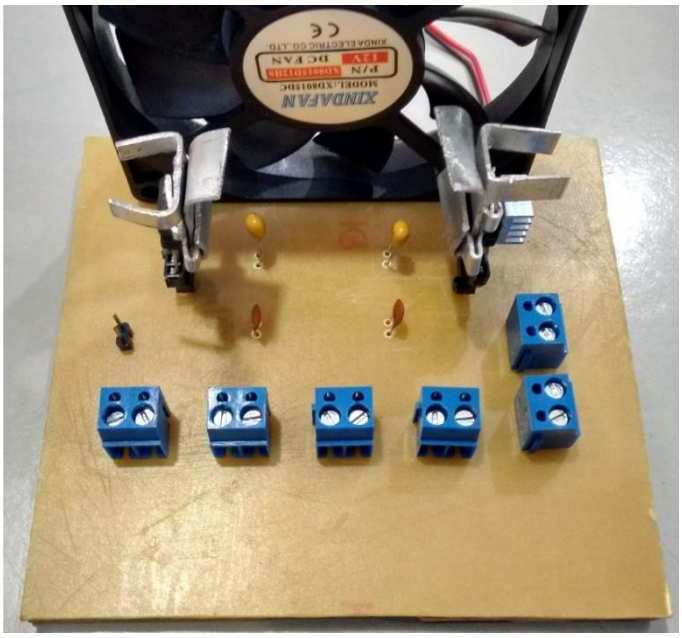


Figura 46 – Placa projetada com componentes
Fonte: autoria própria

7. CONCLUSÃO

A automação de máquinas e processos é fator primordial em todos os setores industriais, assim como em sistemas singulares. Grande parte desta automação diz respeito ao controle numérico computadorizado, sendo utilizado de máquinas fresadoras a robôs industriais.

O motor de passo, considerado tradicionalmente uma máquina especial, tem sido cada vez mais aplicado em diversos sistemas onde o controle de posicionamento é quesito fundamental.

O trabalho de conclusão de curso visou o desenvolvimento de um projeto que contemplasse ambos os temas de forma que suas principais teorias fossem aplicadas. Para tanto, foi criada uma máquina fresadora CNC de baixo custo com a função de confeccionar trilhas de circuito impresso.

Para que o projeto alcançasse os objetivos propostos, foram necessárias diversas etapas. Foram feitos o estudo e montagem da estrutura da máquina fresadora de baixo custo para que o sistema pudesse ser aplicado.

Além disso, foram estudados os conceitos fundamentais a respeito do controle numérico computadorizado e de motores de passo aplicados a controle de posicionamento. Ademais, estudos específicos referentes ao microcontrolador utilizado se fizeram necessários para o desenvolvimento do código interpretador, sendo este o principal componente desenvolvido do projeto.

O código interpretador recebe os dados provenientes da etapa CAD/CAM de projeto da placa de circuito, após serem devidamente condicionados e validados. O código desenvolvido tem como base os principais comandos de código G necessários para a implementação da máquina fresadora, bem como os fatores referentes ao sistema de baixo custo, que foram fundamentais para que o projeto obtivesse êxito.

Para trabalhos futuros, é possível buscar melhorias com relação à etapa de adequação do código G exportado, podendo ser feita juntamente com o código interpretador em linguagem C. Ademais, pode-se trabalhar também na otimização do código interpretador.

REFERÊNCIAS

All About Circuits. **Chapter 13**: Stepper Motors. Disponível em: < https://www.allaboutcircuits.com/textbook/alternating-current/chpt-13/stepper-motors/>. Acesso em: 29 mar. 2018.

CHAPMAN, Stephen J. **Fundamentos de máquinas elétricas**. Porto Alegre: AMGH, 2013.

CONDIT, Reston; JONES, Dr. Douglas W. **Stepping Motor Fundamentals**. Disponível em: <http://ww1.microchip.com/downloads/en/AppNotes/00907a.pdf>. Acesso em: 8 abr. 2018.

Duratex. **Boletim do Marceneiro**. Disponível em: < http://www.guiadomarceneiro.com/index.php?dir=mad_arq&gdm=mdfs>. Acesso em: 18 jan. 2017.

Essel Engenharia. **Cálculos de roscas**. Disponível em: < http://essel.com.br/cursos/material/01/ElementosMaquinas/09elem.pdf>. Acesso em: 25 abr. 2018.

FITZGERALD, Arthur E.; JR, Charles K.; UMANS, Stephen D. **Máquinas Elétricas**: com introdução à eletrônica de potência. USA: McGraw-Hill, 2003.

FPInnovations Forintek. **Understanding CNC Routers**. Disponível em: < http://www.solutionsforwood.ca/_docs/reports/UnderstandingCNCRouters.pdf >. Acesso em: 20 jan. 2018.

IAASR High Integrity Components. **Introduction to Stepper Motors**. Disponível em: <http://www.iaasr.com/introduction-to-stepper-motors/>. Acesso em: 29 mar. 2018.

ISO 6983-1:2009: Automation systems and integration – Numerical control of machines – Program format and definitions of address words – Part 1: Data format for positioning, line motion and contouring control systems. **International Organization for Standardization**, dez. 2009. Disponível em: < https://www.iso.org/standard/34608.html>. Acesso em: fevereiro de 2016

JOHNSON, John T. **PCB-GCODE User's Manual**: v. 3.6.0.4. Disponível em: <http://www.johnjohnson.info/wp-content/uploads/2013/05/pcbgcode.pdf>. Acesso em: 10 mar. 2018.

KRUG, Rodrigo. **Tutorial Software Eagle v. 5.10**. Disponível em: <http://www.placompel.com.br/wa_files/Tutorial_Eagle.pdf>. Acesso em: 7 mar. 2018.

MARCICANO, João Paulo P. Escola Politécnica da Universidade de São Paulo. Introdução à Manufatura Mecânica. **Introdução ao Controle Numérico**. São Paulo, 2014, 14 p. Disponível em: < http://sites.poli.usp.br/d/pmr2202/arquivos/aulas/cnc.pdf>. Acesso em: 07 fev. 2017.

MathWorks. **Stepper Motor**. Disponível em: < https://www.mathworks.com/help/physmod/sps/powersys/ref/steppermotor.html;jsessionid=b911f1a60cd8672060b11a230cbd?s_tid=gn_loc_drop#brbq9w2-6>. Acesso em: 28 abr. 2018.

Microntek Ferramentaria. **Calibrador anel de rosca**. Disponível em: < http://www.microntek.com.br/calibrador-anel-de-rosca/calibrador-anel-de-rosca-whitworth-bs-919.html>. Acesso em: 25 abr. 2018.

MORAR, Alexandru. **The modelling and simulation of bipolar hybrid stepping motor by Matlab/Simulink**. Disponível em: < https://www.sciencedirect.com/science/article/pii/S2212017315000833>. Acesso em: 17 abr. 2018.

Oriental Motor Corp. **2-phase and 5-phase Stepper Motor Comparison**. Disponível em: <http://www.orientalmotor.com/stepper-motors/technology/2-phase-vs-5-phase-stepper-motors.html>. Acesso em: 15 abr. 2018.

OVERBY, Alan. **CNC Machining Handbook**: Building, Programming, and Implementation. USA: McGraw-Hill, 2011.

Pololu. **DRV8825 Stepper Motor Driver Carrier**. Disponível em: < https://www.pololu.com/product/2132>. Acesso em: 17 mai. 2018.

Smithy. **Machining Handbook**. Disponível em: <https://smithy.com/machining-handbook/chapter-3/page/36>. Acesso em: 16 fev. 2018.

SUH, Suk-Hwan; KANG, Seong-Kyoon; CHUNG, Dae-Hyuk; STROUD, Ian. **Theory and Design of CNC Systems**. London: Springer-Verlag, 2008. Disponível em: <http://www.academia.edu/4977293/Theory_and_Design_of_CNC_Systems>. Acesso em: 19 mar. 2018.

Texas Instruments. **DRV8825 Stepper Motor Controller IC**. Disponível em: <http://www.ti.com/lit/ds/symlink/drv8825.pdf>. Acesso em: 12 jan. 2018.

Texas Instruments. **F28377S LaunchPad Development Kit**. Disponível em: < http://www.ti.com/tool/LAUNCHXL-F28377S>. Acesso em: 17 mai. 2018.

VENKATARATNAM, K. **Special Electrical Machines**. USA: CRC, 2009.

WALENDORFF, Marlon N. **Construa sua própria CNC 2.0**. Disponível em: < http://professormarlonnardi.blogspot.com.br/p/construa-sua.html>. Acesso em: 11 jan. 2017.

APÊNDICE A – CÓDIGO INTERPRETADOR

```
#include "F28x_Project.h"
#include <math.h>
#include <string.h>

//Configurações iniciais
void Desabilita_WDT(void); //Desabilita o WDT
void Ini_gpio(void); //Inicialização das portas I/O
void Ini_pwm(void); //Configuração do ePWM2
void Config_cpu(void); //Configuração da CPU
interrupt void epwm2_interrupt(void); //Interrupção do ePWM2

unsigned int periodo = 9687; //Período para PWM de 10kHz

Uint32 k, u, step, RefX, RefY;
float dx, dy, dz, DX1, DX2, DY1, DY2, DZ1, DZ2, DX45, DY45, l = 0.0002204861, /*l: distância (mm) percorrida
pela barra de rosca a cada passo em 1/32 microstep. Baseada no tipo da barra de rosca (18 fios por polegada,
percorrendo 1,41 mm a cada giro completo da barra)*/
PosATX, POSATX, PosATY, POSATY,
counterCWX = 0.0, counterCCWX = 0.0, counterCWY = 0.0, counterCCWY = 0.0, counterCWZ = 0.0,
counterCCWZ = 0.0;
int v, n, Nada;

#define FLOAT_SIZE 6000
float32 EZ[FLOAT_SIZE];
#pragma DATA_SECTION(EZ, ".COORD_TABLE");

float32 EX[FLOAT_SIZE];
#pragma DATA_SECTION(EX, ".COORD_TABLE");

float32 EY[FLOAT_SIZE];
#pragma DATA_SECTION(EY, ".COORD_TABLE");

float32 EZ[FLOAT_SIZE] = {};

float32 EX[FLOAT_SIZE] = {};

float32 EY[FLOAT_SIZE] = {};

float32 P[] = {10};

char *GCODE[] = {}; //Sequência dos comandos das linhas de código G

void main(void){

n = (sizeof(GCODE)/sizeof(*GCODE)); //Cálculo da quantidade de linhas/comandos de código G

//Constantes para interpretação do código G
const char * const A[] = {"G00"};
const char * const B[] = {"G01"};
const char * const C[] = {"G04"};
const char * const D[] = {"G28"};

```

```
53 //Transfere o código para a memória Flash no debug
54 #ifdef _FLASH
55 memcpy(&RamfuncsRunStart, &RamfuncsLoadStart, (size_t)&RamfuncsLoadSize);
56 #endif
57
58 //Inicializa o PLL, WatchDog e clocks periféricos
59 InitSysCtrl();
60
61 DINT; //Desabilita todas as interrupções globais
62
63 //Inicializa registradores de PIE
64 InitPieCtrl();
65
66 IER = 0x0000; //Desabilita todas as interrupções da CPU
67 IFR = 0x0000; //Limpa as flags de todas as interrupções da CPU
68
69 //Inicializa tabela de vetores para ISR
70 InitPieVectTable();
71
72 //Ativa as configurações iniciais
73 Desabilita_WDT();
74 Config_cpu();
75 Ini_gpio();
76 Ini_pwm();
77
78 EINT; //Habilita a interrupção global INTM
79 ERTM; //Habilita a interrupção global em tempo real DBGM
80
81 EALLOW;
82 ClkCfgRegs.PERCLKDIVSEL.bit.EPWMCLKDIV = 1; //Divisão por 2 do PLLSYSCLK = 193,75MHz/2 = 96,875MHz
83 -> clock inicial do ePWM
84 EDIS;
85 EALLOW;
86 CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;
87 //Habilita TBCLK - clock como base de tempo para ePWM
88 CpuSysRegs.PCLKCR2.bit.EPWM2 = 1;
89 //Habilita o clock para o ePWM2
90 EDIS;
91
92 EALLOW;
93 PieVectTable.EPWM2_INT = &epwm2_interrupt; //Registrador para o tratamento da interrupção do ePWM2
94 EDIS;
95
96 IER |= M_INT3; //Habilita interrupções INT3 da CPU (ePWM1-3)
97
98 PieCtrlRegs.PIEIER3.bit.INTx2 = 1; //Habilita ePWM2_INT no PIE: grupo 3, interrupção 2
99 PieCtrlRegs.PIECTRL.bit.ENPIE = 1; //Habilita PIE
100
101 EPwm2Regs.TBCTL.bit.CTRMODE = 0;
102 //Modo de contagem UP
103
104 //Interpretação das n linhas de código G
105 for (v = 0; v < n; v++){
106
107 //Cálculo das distâncias a serem percorridas (mm)
108 if(v == 0){
109
```

```
110 dz = EZ[0] - 0;
111 dx = EX[0] - 0;
112 dy = EY[0] - 0;
113 }
114
115 else{
116
117 dz = EZ[v] - EZ[v-1];
118 dx = EX[v] - EX[v-1];
119 dy = EY[v] - EY[v-1];}
120
121 //G00 = movimentação rápida, sem fresar (motor CC parado), motores de passo em RPM maior
122 if(strcmp (GCODE[v], A[0]) == 0){
123
124 //Motor CC parado
125 EALLOW;
126 EPwm2Regs.CMPA.bit.CMPA = 0;
127 EDIS;
128 GpioDataRegs.GPACLEAR.bit.GPIO18 = 1;
129 GpioDataRegs.GPACLEAR.bit.GPIO19 = 1;
130 //1/32 microstep
131 GpioDataRegs.GPASET.bit.GPIO14 = 1;
132 GpioDataRegs.GPASET.bit.GPIO15 = 1;
133 GpioDataRegs.GPASET.bit.GPIO16 = 1;
134
135 //Eixo Z sempre se move primeiro para determinar altura da ferramenta antes das ações seguintes
136 if(dz > 0){
137 //Sentido horário
138 DZ1 = fabs(dz); //Valor absoluto de dz
139 GpioDataRegs.GPCCLEAR.bit.GPIO69 = 1;
140 step = DZ1/l; //Quantidade de passos necessários
141 counterCWZ = counterCWZ + DZ1; //Contador das distâncias percorridas (mm) no sentido horário
142 for (u = 0; u < step; u++){
143
144 DELAY_US(117.187); //f = 4,267kHz = 40 RPM
145 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
146 DELAY_US(117.187);
147 GpioDataRegs.GPASET.bit.GPIO11 = 1;
148 }
149 }
150
151 else if(dz < 0){
152 //Sentido anti-horário
153 DZ2 = fabs(dz);
154 GpioDataRegs.GPCSET.bit.GPIO69 = 1;
155 step = DZ2/l;
156 counterCCWZ = counterCCWZ + DZ2; //Contador das distâncias percorridas (mm) no sentido anti-horário
157 for (u = 0; u < step; u++){
158
159 DELAY_US(117.187);
160 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
161 DELAY_US(117.187);
162 GpioDataRegs.GPASET.bit.GPIO11 = 1;
163 }
164 }
165
166 else{
```

```
167 //Motor permanece parado
168 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
169 Nada = Nada + 1;
170 }
171
172 if(dx > 0){
173 //Sentido horário
174 DX1 = fabs(dx);
175 GpioDataRegs.GPCSET.bit.GPIO65 = 1;
176 step = DX1/l;
177 counterCWX = counterCWX + DX1;
178 for (u = 0; u < step; u++){
179
180 DELAY_US(62.5); //f = 8kHz = 75 RPM
181 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
182 DELAY_US(62.5);
183 GpioDataRegs.GPASET.bit.GPIO3 = 1;
184 }
185 }
186
187 else if(dx < 0){
188 //Sentido anti-horário
189 DX2 = fabs(dx);
190 GpioDataRegs.GPCCLEAR.bit.GPIO65 = 1;
191 step = DX2/l;
192 counterCCWX = counterCCWX + DX2;
193 for (u = 0; u < step; u++){
194
195 DELAY_US(62.5);
196 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
197 DELAY_US(62.5);
198 GpioDataRegs.GPASET.bit.GPIO3 = 1;
199 }
200 }
201
202 else{
203 //Motor permanece parado
204 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
205 Nada = Nada + 1;
206 }
207
208 if(dy > 0){
209
210 //Sentido horário
211 DY1 = fabs(dy);
212 GpioDataRegs.GPCSET.bit.GPIO66 = 1;
213 step = DY1/l;
214 counterCWY = counterCWY + DY1;
215 for (u = 0; u < step; u++){
216
217 DELAY_US(62.5);
218 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
219 DELAY_US(62.5);
220 GpioDataRegs.GPASET.bit.GPIO10 = 1;
221 }
222 }
223
```

```
224 else if(dy < 0){
225 //Sentido anti-horário
226 DY2 = fabs(dy);
227 GpioDataRegs.GPCCLEAR.bit.GPIO66 = 1;
228 step = DY2/l;
229 counterCCWY = counterCCWY + DY2;
230 for (u = 0; u < step; u++){
231
232 DELAY_US(62.5);
233 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
234 DELAY_US(62.5);
235 GpioDataRegs.GPASET.bit.GPIO10 = 1;
236 }
237 }
238
239 else{
240 //Motor permanece parado
241 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
242 Nada = Nada + 1;
243 }}
244
245 //G01 = interpolação, ação de fresar (motor CC acionado), motores de passo em RPM menor
246 else if(strcmp (GCODE[v], B[0]) == 0){
247
248 //Motor CC acionado
249 GpioDataRegs.GPASET.bit.GPIO18 = 1;
250 GpioDataRegs.GPASET.bit.GPIO19 = 1;
251 //Motor CC é acionado de forma suave (em 5s, incrementos de 1% de duty cycle)
252 while(EPwm2Regs.CMPA.bit.CMPA < periodo){
253 DELAY_US(0.05e6);
254 EALLOW;
255 EPwm2Regs.CMPA.bit.CMPA = (EPwm2Regs.CMPA.bit.CMPA + periodo*0.01);
256 EDIS;}
257
258 //1/32 microstep
259 GpioDataRegs.GPASET.bit.GPIO14 = 1;
260 GpioDataRegs.GPASET.bit.GPIO15 = 1;
261 GpioDataRegs.GPASET.bit.GPIO16 = 1;
262
263 DX1 = fabs(dx);
264 DY1 = fabs(dy);
265 DX45 = floorf(DX1 * 100) / 100; //Arredondamento necessário para a exatidão dos valores dos catetos
266 DY45 = floorf(DY1 * 100) / 100;
267
268 if(dz > 0){
269 //Sentido horário
270 DZ1 = fabs(dz);
271 GpioDataRegs.GPCCLEAR.bit.GPIO69 = 1;
272 step = DZ1/l; //Quantidade de passos necessários
273 counterCWZ = counterCWZ + DZ1;
274 for (u = 0; u < step; u++){
275
276 DELAY_US(312.5); //f = 1,6kHz = 15 RPM
277 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
278 DELAY_US(312.5);
279 GpioDataRegs.GPASET.bit.GPIO11 = 1;
280 }
```

```
281 }
282
283 else if(dz < 0){
284 //Sentido anti-horário
285 DZ2 = fabs(dz);
286 GpioDataRegs.GPCSET.bit.GPIO69 = 1;
287 step = DZ2/l;
288 counterCCWZ = counterCCWZ + DZ2;
289 for (u = 0; u < step; u++){
290
291 DELAY_US(312.5);
292 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
293 DELAY_US(312.5);
294 GpioDataRegs.GPASET.bit.GPIO11 = 1;
295 }
296 }
297
298 else{
299 //Motor permanece parado
300 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
301 Nada = Nada + 1;
302 }
303
304 if(DX45 == DY45 && DX45 >= 0.1){
305 step = DX1/l;
306 //Interpolação de retas oblíquas padronizadas em 45°. Análise dos 4 quadrantes cartesianos.
307 if(dx > 0 && dy < 0){
308 GpioDataRegs.GPCSET.bit.GPIO65 = 1;
309 GpioDataRegs.GPCCLEAR.bit.GPIO66 = 1;
310 counterCWX = counterCWX + DX1;
311 counterCCWY = counterCCWY + DY1;
312 for (u = 0; u < step; u++){
313
314 DELAY_US(312.5); //15 RPM
315 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
316 DELAY_US(312.5);
317 GpioDataRegs.GPASET.bit.GPIO3 = 1;
318 DELAY_US(312.5);
319 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
320 DELAY_US(312.5);
321 GpioDataRegs.GPASET.bit.GPIO10 = 1;
322 }}
323 else if(dx < 0 && dy > 0){
324 GpioDataRegs.GPCCLEAR.bit.GPIO65 = 1;
325 GpioDataRegs.GPCSET.bit.GPIO66 = 1;
326 counterCCWX = counterCCWX + DX1;
327 counterCWY = counterCWY + DY1;
328 for (u = 0; u < step; u++){
329
330 DELAY_US(312.5); //15 RPM
331 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
332 DELAY_US(312.5);
333 GpioDataRegs.GPASET.bit.GPIO3 = 1;
334 DELAY_US(312.5);
335 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
336 DELAY_US(312.5);
337 GpioDataRegs.GPASET.bit.GPIO10 = 1;
```

```
338 }
339 }else if(dx && dy > 0){
340 GpioDataRegs.GPCSET.bit.GPIO65 = 1;
341 GpioDataRegs.GPCSET.bit.GPIO66 = 1;
342 counterCWX = counterCWX + DX1;
343 counterCWY = counterCWY + DY1;
344 for (u = 0; u < step; u++){
345
346 DELAY_US(312.5); //15 RPM
347 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
348 DELAY_US(312.5);
349 GpioDataRegs.GPASET.bit.GPIO3 = 1;
350 DELAY_US(312.5);
351 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
352 DELAY_US(312.5);
353 GpioDataRegs.GPASET.bit.GPIO10 = 1;
354 }
355 }else{
356 GpioDataRegs.GPCCLEAR.bit.GPIO65 = 1;
357 GpioDataRegs.GPCCLEAR.bit.GPIO66 = 1;
358 counterCCWX = counterCCWX + DX1;
359 counterCCWY = counterCCWY + DY1;
360 for (u = 0; u < step; u++){
361
362 DELAY_US(312.5); //15 RPM
363 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
364 DELAY_US(312.5);
365 GpioDataRegs.GPASET.bit.GPIO3 = 1;
366 DELAY_US(312.5);
367 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
368 DELAY_US(312.5);
369 GpioDataRegs.GPASET.bit.GPIO10 = 1;
370 }
371 }
372 }
373 else{
374
375 if(dx > 0){
376 //Sentido horário
377 DX1 = fabs(dx);
378 GpioDataRegs.GPCSET.bit.GPIO65 = 1;
379 step = DX1/l;
380 counterCWX = counterCWX + DX1;
381 for (u = 0; u < step; u++){
382
383 DELAY_US(312.5); //15 RPM
384 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
385 DELAY_US(312.5);
386 GpioDataRegs.GPASET.bit.GPIO3 = 1;
387 }
388 }
389
390 else if(dx < 0){
391 //Sentido anti-horário
392 DX2 = fabs(dx);
393 GpioDataRegs.GPCCLEAR.bit.GPIO65 = 1;
394 step = DX2/l;
```

```
395 counterCCWX = counterCCWX + DX2;
396 for (u = 0; u < step; u++){
397
398 DELAY_US(312.5);
399 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
400 DELAY_US(312.5);
401 GpioDataRegs.GPASET.bit.GPIO3 = 1;
402 }
403 }
404
405 else{
406 //Motor permanece parado
407 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
408 Nada = Nada + 1;
409 }
410
411 if(dy > 0){
412 //Sentido horário
413 DY1 = fabs(dy);
414 GpioDataRegs.GPCSET.bit.GPIO66 = 1;
415 step = DY1/l;
416 counterCWY = counterCWY + DY1;
417 for (u = 0; u < step; u++){
418
419 DELAY_US(312.5);
420 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
421 DELAY_US(312.5);
422 GpioDataRegs.GPASET.bit.GPIO10 = 1;
423 }
424 }
425
426 else if(dy < 0){
427 //Sentido anti-horário
428 DY2 = fabs(dy);
429 GpioDataRegs.GPCCLEAR.bit.GPIO66 = 1;
430 step = DY2/l;
431 counterCCWY = counterCCWY + DY2;
432 for (u = 0; u < step; u++){
433
434 DELAY_US(312.5);
435 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
436 DELAY_US(312.5);
437 GpioDataRegs.GPASET.bit.GPIO10 = 1;
438 }
439 }
440
441 else{
442 //Motor permanece parado
443 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
444 Nada = Nada + 1;
445 }}}
446
447 //G04 = todos os motores parados por tempo arbitrário em segundos
448 else if(strcmp (GCODE[v], C[0]) == 0){
449
450 GpioDataRegs.GPADAT.bit.GPIO13 = 0; //Ativa LED azul
451 //Motor CC parado
```

```
452 EALLOW;
453 EPwm2Regs.CMPA.bit.CMPA = 0;
454 EDIS;
455 GpioDataRegs.GPACLEAR.bit.GPIO18 = 1;
456 GpioDataRegs.GPACLEAR.bit.GPIO19 = 1;
457 //Motores de passo parados
458 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
459 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
460 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
461 DELAY_US(P[0]*1e6);
462 asm(" ESTOP0"); //Pausa o debugger para troca de ferramenta
463 GpioDataRegs.GPADAT.bit.GPIO13 = 1; //Desativa LED azul
464 }
465
466 //G28 = retorno à referência
467 else if(strcmp (GCODE[v], D[0]) == 0){
468
469 //Motor CC parado
470 EALLOW;
471 EPwm2Regs.CMPA.bit.CMPA = 0;
472 EDIS;
473 GpioDataRegs.GPACLEAR.bit.GPIO18 = 1;
474 GpioDataRegs.GPACLEAR.bit.GPIO19 = 1;
475 //1/32 microstep
476 GpioDataRegs.GPASET.bit.GPIO14 = 1;
477 GpioDataRegs.GPASET.bit.GPIO15 = 1;
478 GpioDataRegs.GPASET.bit.GPIO16 = 1;
479
480 //Cálculo das coordenadas atuais (mm)
481 PosATX = counterCWX - counterCCWX;
482 POSATX = fabs(PosATX);
483 RefX = POSATX/l;
484
485 PosATY = counterCWY - counterCCWY;
486 POSATY = fabs(PosATY);
487 RefY = POSATY/l;
488
489 if(PosATX > 0){
490
491 for (k = 0; k < RefX; k++){
492 GpioDataRegs.GPCCLEAR.bit.GPIO65 = 1;
493 DELAY_US(62.5); //75 RPM
494 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
495 DELAY_US(62.5);
496 GpioDataRegs.GPASET.bit.GPIO3 = 1;
497 }
498 }
499
500 else if(PosATX < 0){
501
502 for (k = 0; k < RefX; k++){
503 GpioDataRegs.GPCSET.bit.GPIO65 = 1;
504 DELAY_US(62.5);
505 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
506 DELAY_US(62.5);
507 GpioDataRegs.GPASET.bit.GPIO3 = 1;
508 }
```

```
509 }
510
511 else{
512 //Motor permanece parado
513 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
514 Nada = Nada + 1;
515 }
516
517 if(PosATY > 0){
518
519 for (k = 0; k < RefY; k++){
520 GpioDataRegs.GPCCLEAR.bit.GPIO66 = 1;
521 DELAY_US(62.5);
522 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
523 DELAY_US(62.5);
524 GpioDataRegs.GPASET.bit.GPIO10 = 1;
525 }
526 }
527
528 else if(PosATY < 0){
529
530 for (k = 0; k < RefY; k++){
531 GpioDataRegs.GPCSET.bit.GPIO66 = 1;
532 DELAY_US(62.5);
533 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
534 DELAY_US(62.5);
535 GpioDataRegs.GPASET.bit.GPIO10 = 1;
536 }
537 }
538
539 else{
540 //Motor permanece parado
541 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
542 Nada = Nada + 1;
543 }}
544
545 else{
546 //Motor CC parado
547 EALLOW;
548 EPwm2Regs.CMPA.bit.CMPA = 0;
549 EDIS;
550 GpioDataRegs.GPACLEAR.bit.GPIO18 = 1;
551 GpioDataRegs.GPACLEAR.bit.GPIO19 = 1;
552 //Motores de passo parados
553 GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
554 GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
555 GpioDataRegs.GPACLEAR.bit.GPIO11 = 1;
556 Nada = Nada + 1;
557 }
558 }
559
560 //Para motor CC
561 EALLOW;
562 EPwm2Regs.CMPA.bit.CMPA = 0;
563 EDIS;
564 GpioDataRegs.GPACLEAR.bit.GPIO18 = 1;
565 GpioDataRegs.GPACLEAR.bit.GPIO19 = 1;
```

```
566 //Software breakpoint, para o debugger ao terminar interpretação do código
567 asm(" ESTOP0");
568
569 }
570
571 void Desabilita_WDT(void)
572 {
573 volatile Uint16 temp;
574 EALLOW;
575 temp = WdRegs.WDCR.all & 0x0007;
576 WdRegs.WDCR.all = 0x0068 | temp;
577 EDIS;
578 }
579
580 void Ini_gpio(void)
581 {
582 EALLOW;
583 //Direção das GPIO como saída
584 GpioCtrlRegs.GPADIR.bit.GPIO18 = 1;
585 GpioCtrlRegs.GPADIR.bit.GPIO19 = 1;
586 GpioCtrlRegs.GPADIR.bit.GPIO13 = 1;
587 GpioCtrlRegs.GPADIR.bit.GPIO14 = 1;
588 GpioCtrlRegs.GPADIR.bit.GPIO15 = 1;
589 GpioCtrlRegs.GPADIR.bit.GPIO16 = 1;
590 GpioCtrlRegs.GPCDIR.bit.GPIO65 = 1;
591 GpioCtrlRegs.GPCDIR.bit.GPIO66 = 1;
592 GpioCtrlRegs.GPCDIR.bit.GPIO69 = 1;
593 GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;
594 GpioCtrlRegs.GPADIR.bit.GPIO10 = 1;
595 GpioCtrlRegs.GPADIR.bit.GPIO11 = 1;
596 //Motor CC inicia parado
597 GpioDataRegs.GPACLEAR.bit.GPIO18 = 1;
598 GpioDataRegs.GPACLEAR.bit.GPIO19 = 1;
599 //LED azul desativado
600 GpioDataRegs.GPADAT.bit.GPIO13 = 1;
601
602 //Configura GPIO2 como ePWM2A
603 GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1; //GPIO2 (Pino 80): PWM motor CC
604 //Configura GPIOs subsequentes como GPIO
605 GpioCtrlRegs.GPAMUX1.bit.GPIO13 = 0; //GPIO13 (LED azul)
606 GpioCtrlRegs.GPAMUX2.bit.GPIO18 = 0; //GPIO18 (Pino 76): função R_EN, driver motor CC
607 GpioCtrlRegs.GPAMUX2.bit.GPIO19 = 0; //GPIO19 (Pino 75): função L_EN, driver motor CC
608 GpioCtrlRegs.GPAMUX1.bit.GPIO14 = 0; //GPIO14 (Pino 38): função MODE0, LSB do tipo de passo
609 GpioCtrlRegs.GPAMUX1.bit.GPIO15 = 0; //GPIO15 (Pino 37): função MODE1
610 GpioCtrlRegs.GPAMUX2.bit.GPIO16 = 0; //GPIO16 (Pino 36): função MODE2, MSB do tipo de passo
611 GpioCtrlRegs.GPCMUX1.bit.GPIO65 = 0; //GPIO65 (Pino 47): DIR do eixo X
612 GpioCtrlRegs.GPCMUX1.bit.GPIO66 = 0; //GPIO66 (Pino 50): DIR do eixo Y
613 GpioCtrlRegs.GPCMUX1.bit.GPIO69 = 0; //GPIO69 (Pino 49): DIR do eixo Z
614 GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 0; //GPIO3 (Pino 79): STEP do eixo X
615 GpioCtrlRegs.GPAMUX1.bit.GPIO10 = 0; //GPIO10 (Pino 78): STEP do eixo Y
616 GpioCtrlRegs.GPAMUX1.bit.GPIO11 = 0; //GPIO11 (Pino 77): STEP do eixo Z
617
618 //Desabilita pull-up na GPIO2 (ePWM2A)
619 GpioCtrlRegs.GPAPUD.bit.GPIO2 = 1;
620 //Habilita pull-up nas GPIOs subsequentes
621 GpioCtrlRegs.GPAPUD.bit.GPIO13 = 0;
622 GpioCtrlRegs.GPAPUD.bit.GPIO18 = 0;
```

```
623 GpioCtrlRegs.GPAPUD.bit.GPIO19 = 0;
624 GpioCtrlRegs.GPAPUD.bit.GPIO14 = 0;
625 GpioCtrlRegs.GPAPUD.bit.GPIO15 = 0;
626 GpioCtrlRegs.GPAPUD.bit.GPIO16 = 0;
627 GpioCtrlRegs.GPCPUD.bit.GPIO65 = 0;
628 GpioCtrlRegs.GPCPUD.bit.GPIO66 = 0;
629 GpioCtrlRegs.GPCPUD.bit.GPIO69 = 0;
630 GpioCtrlRegs.GPAPUD.bit.GPIO3 = 0;
631 GpioCtrlRegs.GPAPUD.bit.GPIO10 = 0;
632 GpioCtrlRegs.GPAPUD.bit.GPIO11 = 0;
633 EDIS;
634 }
635
636 void Config_cpu(void)
637 {
638 EALLOW;
639 //Frequência de clock: fPLLSYSCLK = fOSCCLK*(IMULT+FMULT)/PLLSYSCLKDIV = 10M*38,75/2 = 193,75MHz
640 CpuSysRegs.PCLKCR0.bit.CPUTIMER0 = 1;
641 CpuSysRegs.PCLKCR0.bit.CPUTIMER1 = 1;
642 CpuSysRegs.PCLKCR0.bit.CPUTIMER2 = 1;
643 ClkCfgRegs.CLKSRCCTL1.bit.INTOSC2OFF = 0; //Ativa o INTOSC2
644 ClkCfgRegs.CLKSRCCTL1.bit.OSCCLKSRCSEL = 0; //Fonte de clock = INTOSC2
645 ClkCfgRegs.SYSPLLMULT.bit.FMULT = 3; //Fator fracionário de 0,75
646 ClkCfgRegs.SYSPLLMULT.bit.IMULT = 38; //Fator inteiro de 38
647 ClkCfgRegs.SYSCLKDIVSEL.bit.PLLSYSCLKDIV = 1;//Divisão por 2
648 ClkCfgRegs.SYSPLLCTL1.bit.PLLCLKEN = 1; //Bypass no SYSPLL
649 EDIS;
650 }
651
652 void Ini_pwm(void)
653 {
654 EALLOW;
655 DevCfgRegs.SOFTPRES2.bit.EPWM2 = 1;
656 //ePWM2 é resetado
657 DevCfgRegs.SOFTPRES2.bit.EPWM2 = 0;
658 //ePWM2 é liberado do reset
659 EPwm2Regs.TBCTL.bit.CTRMODE = 3;
660 //Desabilita o timer
661 EPwm2Regs.TBPHS.bit.TBPHS = 0;
662 //Carrega zero de defasagem no registrador de fase
663 EPwm2Regs.TBPRD = periodo;
664 //Define o período para o ePWM
665 EPwm2Regs.CMPA.bit.CMPA = 0;
666 //0% de duty cycle inicial
667 EPwm2Regs.TBCTL.bit.PHSEN = 1;
668 //Habilita o carregamento de fase inicial
669 EPwm2Regs.TBCTL.bit.PRDLD = 1;
670 //Carrega imediatamente o período no TBPRD
671 EPwm2Regs.TBCTL.bit.CLKDIV = 0;
672 //f_bclk /1 para ePWM
673 EPwm2Regs.TBCTL.bit.HSPCLKDIV = 0;
674 //Parte da divisão para definir o TBCLK; = 0, /1
675 EPwm2Regs.TBCTR = 0x0000;
676 //Apaga o contador do timer
677 EPwm2Regs.CMPCTL.bit.SHDWAMODE = 0;
678 //O duty cycle é alterado em pontos estratégicos do contador
679 EPwm2Regs.CMPCTL.bit.LOADAMODE = 0;
```

```
680 //O duty cycle é carregado somente quando o contador passa por zero (TBCTR=0X0000)
681 EPwm2Regs.AQCTLA.bit.CAU = 1;
682 //Clear quando o contador chega no CMPA
683 EPwm2Regs.AQCTLA.bit.ZRO = 2;
684 //Seta quando o contador passa por zero
685
686 //Configuração da interrupção de ePWM2 toda vez o contador chegar ao período de contagem
687 EPwm2Regs.ETSEL.bit.SOCAEN = 0;
688 //Desabilita o início de conversão do AD (SOCA)
689 EPwm2Regs.ETSEL.bit.SOCASEL = 4;
690 //Conversão inicia quando TBCTR = 0 ou TBCTR = TBPRD
691 EPwm2Regs.ETPS.bit.SOCAPRD = 1;
692 //Gera pulso no primeiro evento de interrupção
693 EPwm2Regs.ETSEL.bit.INTEN = 0;
694 //Desabilita a interrupção gerada pelo ePWM2
695 EPwm2Regs.ETSEL.bit.INTSEL= 4;
696 //Interrupção quando o contador atinge o período
697 EPwm2Regs.ETPS.bit.INTPRD = 3;
698 //Interrupção a cada ciclo do PWM
699 EDIS;
700 }
701
702 interrupt void epwm2_interrupt(void)
703 {
704 EPwm2Regs.ETCLR.bit.INT = 1;
705 //Apaga a flag de interrupção
706
707 PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
708 //Checa a interrupção para receber novas interrupções
709 }
```