Automação no Processo de Impressão de Placas de Circuito Impresso

1st *Gustavo Cavalcante Linhares*
Universidade de Brasília
16/0007810

2nd *Luiz Eduardo Alves Machado*
Universidade de Brasília
16/0013623

***Resumo* — O presente documento tem como objetivo mostrar todos aspectos da proposta de projeto final relacionado a disciplina de Eletrônica Embarcada. No caso trata-se de uma semi automação no processo de fabricação de placas de circuito impresso (PCI's), tendo como partida tornar a qualidade e o método de fabricação tanto mais fáceis quanto eficientes.**

Keywords—CNC machine, PCI, PCB

I. INTRODUCÃO

A ideia do projeto veio da intenção de estudar o processo de fabricação das PCIs (Placas de Circuito Impresso) . O interesse pelos circuitos aumentou durante o período de estudo sobre eletrônica. Juntando isso com a necessidade de criar um projeto que ajudasse com a prototipação de projetos ao longo da vida acadêmica, surgiu uma ideia que facilitasse a produção (impressão) de uma placa de circuito impresso.

Existem vários tipos de métodos de fabricação caseira de PCI, como por exemplo a transferência térmica [1] ou o método usando tinta fotossensível [2], mas devido à complexidade no processo de fabricação destas placas geralmente, quando feitas em casa, o resultado não é satisfatório. Problemas estes geralmente em relação a qualidade, durabilidade e eficiência das placas decorrentes da falta de equipamentos certos e do seguimento correto dos métodos para a confecção das mesmas.

Então devido os motivos apresentados acima torna-se necessario tornar o método de impressão de PCI mais simples e mais automatizado com passos e com processos bem definidos, mas mesmo assim mantendo o máximo de qualidade e confiabilidade possíveis.

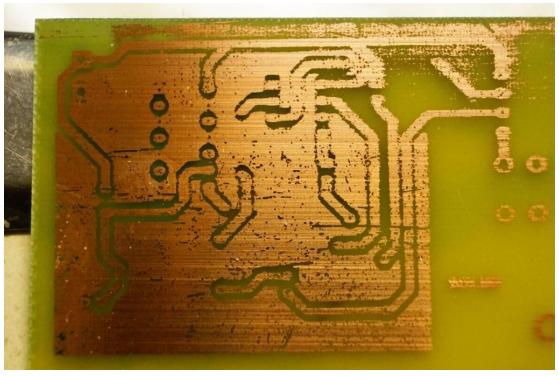

Figura 01 - Placa PCB com acabamento em baixa qualidade

Assim torna-se necessário a existência de um equipamento, que seja acessível para a maioria dos projetistas, e que possa ajudá-los na produção do produto em pequena escala, devido o fato de Máquinas de gravação de placa de circuito [3] prontas ainda estão inviáveis devido seu alto preço.

Figura 02 - Máquina de Gravação de Placas de Circuito

Assim a o projeto trataria da construção de uma máquina de gravação de circuito impresso com os modelos das peças feitos em uma impressora 3D.

II. OBJETIVOS

- Projetar um equipamento que facilite a impressão de PCIs
- Que o equipamento seja capaz de pintar placas de cobre de tamanho 10X10cm
- A capacidade de printar a imagem do circuito na placa através de um interpretador de GCode

III. DESENVOLVIMENTO

RETROSPECTIVA DOS PONTOS DE CONTROLE

O projeto começou com o objetivo de projetar um equipamento que facilitasse a impressão de PCIs utilizando o método de tinta fotossensível, que fosse capaz de pintar placas de cobre do tamanho 10x10cm, com a capacidade de secar a placa e revelar a impressão usando luz ultravioleta após a aplicação da tinta.

O equipamento funcionária de de acordo com vários processos:

- Ao começo da impressão, a placa de circuito deve ser pintada com qualidade, usando a tinta fotossensível, sempre prezando a uniformidade, o que deixa a placa

mais bonita e funcional, e ajuda no resto do processo de fabricação.

- Para continuação do processo de impressão, a tinta fotossensível aplicada a placa deve passar por um processo de secagem, que deve ser rápido e funcional, tentando trazer o equilíbrio entre essas duas exigências.
- Seguindo o processo de impressão, a placa deverá ser exposta a luz negra na presença da impressão, que deve ser previamente colocada pelo usuário, para assim haver uma melhor revelação do modelo na placa.

A ideia inicial do projeto era fazer a estrutura funcionar à base de drivers de DVD, usando uma ponte H. Naquela fase do projeto, o driver do DVD funcionou e o motor andava pra frente e para trás.

Com o decorrer do projeto, foi obtida a estrutura do equipamento (Figura 3), e os drivers de DVD foram trocados por motores de passo que eram controlados por 2 Drivers Motor de Passo EasyDriver A3967, porque esse conjunto tinha mais força no motor

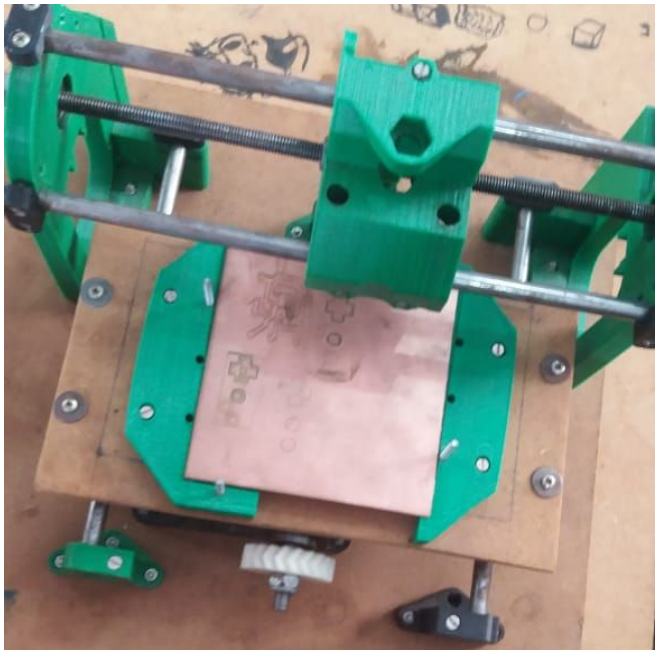

Figura 03 - Estrutura para CNC machine

Porém, ao decorrer do projeto devido a complexidade os objetivos na época vigentes foram mudados, isso devido um diálogo com pessoas mais experientes e também com professor.

Dado o fato do objetivo anterior de pintar a placa exigir muito da calibração e criação de uma máquina muito complexa, foi pensado que com uma CNC seria mais viável a impressão do circuito diretamente por GCode [4].

Assim a construção de uma CNC machine [6] juntamente com todo funcionamento foi o novo foco do projeto.

Na continuação do projeto, percebeu-se que o driver escolhido não estava fornecendo torque suficiente motor de passo rodar, esse fato se deu por falta de informações sobre o modelo do motor de passo usado.

Para continuação do projeto, foi adquirido outro driver para os motores de passo, o Driver Motor de Passo A4988. Mesmo assim o problema com o torque continuou, pois a estrutura era pesada demais e o drive de motor não conseguia fornecer a corrente necessária para o movimento da estrutura.

Como nessa etapa do projeto o tempo se tornou um fator decisivo, foi decidido que o foco do projeto passaria para a parte de software, devido às complicações encontradas com o hardware, assim terminá-lo e corrigi-lo até a data da apresentação do projeto seria a prioridade.

DESCRIÇÃO DO HARDWARE

Para cumprir os requisitos do projeto não serão necessários o uso de sensores, o que traz a parte fundamental e de maior dificuldade do desenvolvimento a calibração e controle dos motores. Lista dos equipamentos de hardware utilizados na prototipagem (Bill of materials):

- MSP430G2ET
- 3 motores de passo
- 3 Drivers Motor de Passo A4988
- Estrutura para CNC Machine
- Fonte externa de 12V-24V

O controle de todo sistema e seus drivers de motor será realizado pelo MSP430. Para a impressão da placa serão necessários 3 motores de passo e 3 Drivers Motor de Passo A4988 , assim cada motor de passo estaria relacionado a um eixo de movimento. O esquemático está representado na figura 04, porem o controlador utilizado será o MSP430

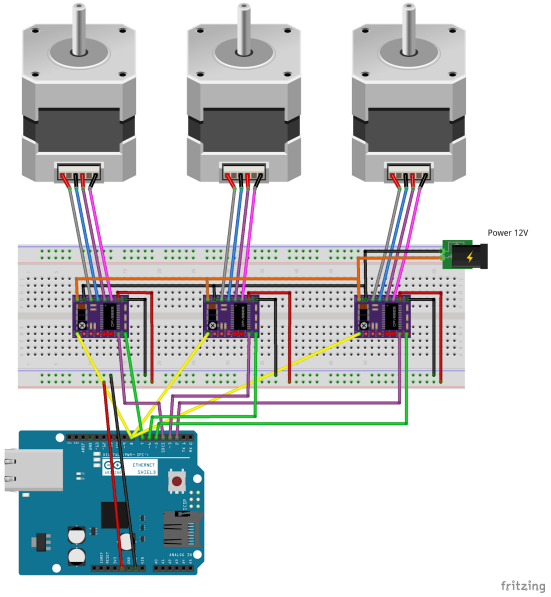

Figura 04 - Esquemático do controle dos motores

Com a seguinte montagem as instruções recebidas pelo controlador seriam decodificadas em movimentos para rotação dos motores conseguindo então concluir a impressão da placa.

As conexões com o drive A4988 e o microcontrolador podem ser melhor visualizadas na figura 05, onde temos as bobinas do motor ligadas nos pinos 2A e 2B e 1A e 1B. A alimentação dos motores no pino VMOT e a do nível lógico do controlador no VDD. Os pinos de Step e dir, são responsáveis pelo controle do número de passos do motor e sua direção respectivamente, sendo nível lógico baixo para sentido anti horário e nível lógico alto para sentido horário, no caso da direção.

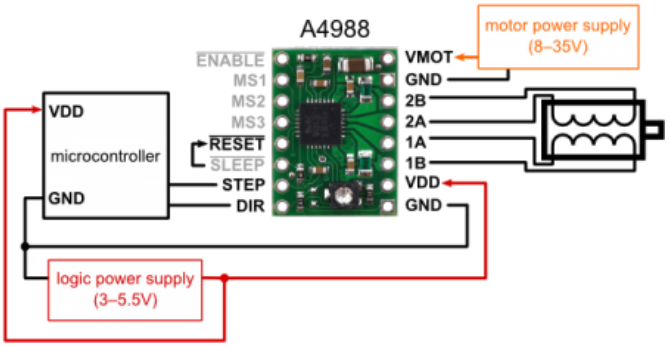

Figura 05 - Esquemático do drive de motor A4988

O step funciona com uma onda quadrada onde cada transição feita é um passo é um passo realizado pelo motor.

PARTES 3D IMPRESSAS

Para o desenvolvimento da estrutura foram necessárias a impressão de várias peças na impressora 3d, a lista de peças e seus definitivos CADs (Computer-aided design) podem ser encontrados na referência [8], juntamente com a lista de rolamentos, parafusos, barras de ferro e demais materiais para a construção da mesma.

Porém para o encaixe dos motores na estrutura foram necessários o desenvolvimento de novas peças (Figura 06/07), que foram projetadas no software CATIA V5 e configuradas para impressão no software Cliever Studio Pro, esse desenvolvimento foi necessário devido uma diferença de tamanho com os motores.

Figura 06 - Impressão dos encaixes das rodas

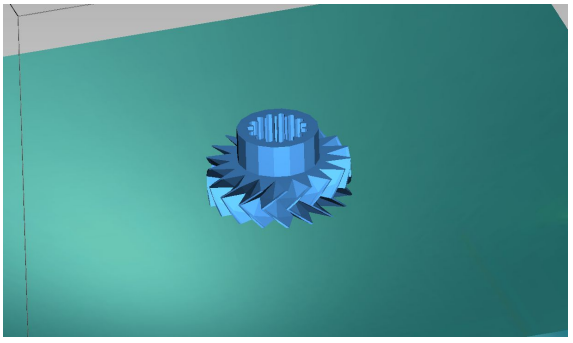

Figura 07 - Modelo 3d construido

DERCRIÇÃO DE SOFTWARE

A plataforma usada para o desenvolvimento dos códigos, nessa etapa do projeto, foi o software chamado Code Composer. Essa IDE tem como objetivo o completo transparência do funcionamento das launchpads da texas instruments, no caso a placa usada foi a MSP430G2ET.

G-CODE

Basicamente o desafio aqui seria montar um interpretador de G-Code. O Código G, do inglês G-Code, é a nome dado à linguagem de programação criada a partir da necessidade de maquinários industriais que faziam uso de sistemas Comando Numérico Computadorizado(CNC).

Tem como função principal instruir a máquina a se mover geometricamente nas três dimensões, x, y e z. É uma linguagem extremamente simples e rudimentar, trata-se de linhas sequenciais de instruções, cada qual responsável por uma tarefa específica e o programa é executado linha por linha até o fim do código. Tem sua maior área de aplicação de uso em máquinas CNC's, porém, com o advento da Impressão 3D, que tem como base a tecnologia CNC, a mesma linguagem é utilizada pelas aplicações relacionadas a Impressão 3D.

O Código G é interpretado por firmwares desenvolvidos para tal função. O firmware converte o Código G em movimentos nos motores e eixos das máquinas.

Então, por suas funcionalidade, e por se adequar aos requisitos do projeto, essa linguagem de programação foi escolhida para nortear o projeto.

Devido o extenso número de comandos existentes no G-Code [7] e muitos deles utilizados somente em grandes e precisos equipamentos da indústria foi optado por fazer somente o suficiente para a impressão das placas já que sua implementação completa foge totalmente do escopo da disciplina.

COMUNICAÇÃO UART

Dada a necessidade de transmissão de dados do computador para o msp430, o protocolo UART foi utilizado para tal finalidade.

A UART funciona da seguinte maneira [9] o pino de transmissão do protocolo envia um pacote de bits que será interpretado bit a bit pelo pino receptor. Cada pacote enviado contém 1 start bit que indica o início da mensagem, 1 ou 2 stop bits para indicar o final da mensagem, 5 a 9 bits de informação e 1 bit de paridade para evitar a recepção de erros.

Então um pacote de bits é enviado de cada vez, assim uma buffer é necessário para guardar os dados até que o envio de uma parcela da informação seja concluída.

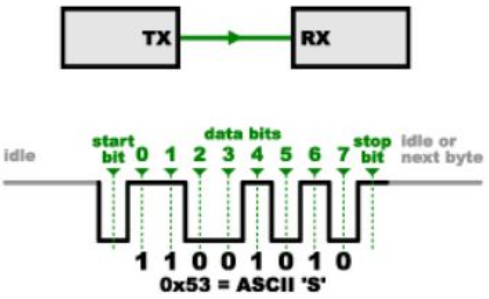

Figura 08 - Protocolo UART

IV. Resultados

O funcionamento do código se dá na realização da recepção de coordenadas e do tipo de movimento dadas pelo G-Code que devem ser transformadas em números de passos e movimento dos motores de cada eixo respectivamente.

Um conjunto de caracteres é enviado ao msp , que quebra essa string em vários números relacionados ao movimento do motor e logo após realizar os movimentos fica a espera de uma nova informação para ser realizada.

Porém não foi-se alcançado o sucesso na parte da tradução das coordenadas em instruções para os motores, apenas o UART e a decodificação dos dados está funcionando de forma satisfatória.

V. Conclusões

Foi percorrido um grande caminho até a conclusão do projeto, onde ocorreram diversos problemas que acarretaram no atraso do andamento do mesmo, devido a dificuldade da necessidade de sincronização de diversos motores, desenvolvimento do GCode e falta de informação sobre partes da estrutura do projeto.

Fatores como a falta de experiência com microcontroladores e de motores de passo também atrasaram significativamente o fluxo do projeto, e mesmo com o avanço com a estrutura, na aquisição dos equipamentos e na familiarização das ferramentas de software sendo alcançadas, todos os problemas no andamento e a mudança de requisitos no meio do projeto fizeram com com o término do projeto fosse inviabilizado.

Contudo, todos os problemas durante o projeto trouxeram um grande aprendizado relacionado a linguagens de programação, parte eletrônica de estruturas e gestão de projeto, de tempo e capacidade de lidar com adversidades. Existiu uma base consolidada nessa parte, foi investido muito tempo no andamento do projeto e mesmo com o término ficando inviável,o aprendizado com todo o projeto foi inegável .

VI. Referências

[1] Transferência de Calor PCB, http://destro.todavia.com.br/tutorial_termico.php

[2] Método de Transferência de Calor, www.youtube.com/watch?v=NtVEvFsT46I

[3] Máquina de Gravura PCB, produto.mercadolivre.com.br/MLB-879960137-mini-gravadora-de-mesa-router-cnc-3-eixos-completa-_JM

[4] Mini CNC machine, www.youtube.com/watch?v=kcz1ygh20c0

[5] Como Fazer Placa de Circuito PCI PCB Profissional, www.youtube.com/watch?v=NtVEvFsT46I\&t=284s

[6] O que é Usinagem CNC, https://www.mecanicaindustrial.com.br/689-o-que-e-usinagem-cnc/

[7] Definição Gcode, https://en.wikipedia.org/wiki/G-code

[8] Documentation for Cyclone PCB Factory, https://github.com/CarlosGS/Cyclone-PCB-Factory/wiki

[9] Comparação entre protocolos de comunicação, https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html

APÊNDICE

```c
#include <msp430g2553.h>
#include <legacymsp430.h>
#include <math.h>
#include <float.h>

// ------------Defines ----------
#define LEDS (BIT0|BIT6)
#define RX BIT1
#define TX BIT2

#define BAUD_9600   0
#define BAUD_19200  1
#define BAUD_38400  2
#define BAUD_56000  3
#define BAUD_115200 4
#define BAUD_128000 5
#define BAUD_256000 6
#define NUM_BAUDS   7

#define MUL 1 // multiplicação de escala

#define MOTORX 0
#define MOTORY 1
#define MOTORZ 2
#define POSF 3 // posicao F
#define POSP 4 // posicao p
#define POSI 5 // posicao i
#define POSJ 6 // posicao j
#define PERIODO 4
#define false 0
#define true 1

#define StepX BIT1
#define DirecaoX BIT2
#define StepY BIT3
#define DirecaoY BIT4
#define StepZ BIT5
#define DirecaoZ BIT7
#define Enable BIT0

// ------------------Funções----------------
int Read_String(char str[], int strlen);
char Receive_Data(void);
void Send_Data(unsigned char c);
void Send_Int(int n);
void Send_String(char str[]);
void Init_UART(unsigned int baud_rate_choice);
void Atraso_ms(volatile unsigned int ms);
int cmp_str(char str1[], char str2[]);
int cmp_char(char str1, char str2);
void processCommand(char str[]);
float get_position(char str1[], int t);
void stepMotor (int Passos, int direcao, int Motor);
void help();
void where();
float stof(char str1[], int position);
int Acha_char(char str1[], char str2);
void Enable_Motor(int a);
void movea(float x,float y);
void headera(int z);
void headerr(int z);
void mover(float x,float y);
void Circle(float x,float y, float i, float j, int dir);
void reset();
float fastsin(float x);

// ------------Variaveis Universais -----------
float x_pos = 0; //current x position
float y_pos = 0; //current y position
float z_pos = 0; //current z position
int feed = 30;
char str[59]; // Buffer do Gcode
int mode_abs = 1;

int main(void)
{
//
    WDTCTL = WDTPW + WDTHOLD;

    BCSCTL1 = CALBC1_1MHZ;
    DCOCTL = CALDCO_1MHZ;

    P1DIR |= LEDS;
    P1OUT |= LEDS;

    Init_UART(BAUD_115200);
    TA1CCR0 = 12500-1;
    TA1CTL = TASSEL_2 + ID_3 + MC_1;

    _BIS_SR(GIE);
    while(1)
    {
        TA1CTL |= TAIE;
        Read_String(str, 59);
        Send_String(str);
        processCommand(str);  // realiza os comandos
    }
    return 0;
}

interrupt(TIMER1_A1_VECTOR) TA1_ISR(void)
{
    P1OUT ^= LEDS;
    TA1CTL &= ~TAIFG;
}

int Read_String(char str[], int strlen)
{
    int i = 0;
    do
    {
        // Salva o caractere recebido
        str[i] = Receive_Data();
        // Ignora o '\r'
        if(str[i]!='\r')
        {
            // Troca '\n' por '\0'
            // e termina a recepção
            if(str[i]=='\n')
            {
                str[i] = '\0';
                break;
            }
            i++;
        }
    } while(i<strlen);

    // Se chegou ao final do vetor,
    // termina-o com '\0'
    if(i==strlen)
    {
        i--;
        str[i] = '\0';
    }
    // Retorna o tamanho da string
    // sem contar o '\0'
    return i;
}

char Receive_Data(void)
{
    while((IFG2&UCA0RXIFG)==0);
    return UCA0RXBUF;
}

void Send_Data(unsigned char c)
{
    while((IFG2&UCA0TXIFG)==0);
    UCA0TXBUF = c;
}
```

```c
void Send_Int(int n)
{
    int casa, dig;
    if(n==0)
    {
        Send_Data('0');
        return;
    }
    if(n<0)
    {
        Send_Data('-');
        n = -n;
    }
    for(casa = 1; casa<=n; casa *= 10);
    casa /= 10;
    while(casa>0)
    {
        dig = (n/casa);
        Send_Data(dig+'0');
        n -= dig*casa;
        casa /= 10;
    }
}

// Envia uma string para o terminal
void Send_String(char str[])
{
    int i;
    for(i=0; str[i]!= '\0'; i++)
        Send_Data(str[i]);
}

// Inicia a UART com o msp430
void Init_UART(unsigned int baud_rate_choice)
{
    unsigned char BRs[NUM_BAUDS] = {104, 52, 26, 17, 8, 7, 3};
    unsigned char MCTLs[NUM_BAUDS] = {UCBRF_0+UCBRS_1,
                                      UCBRF_0+UCBRS_0,
                                      UCBRF_0+UCBRS_0,
                                      UCBRF_0+UCBRS_7,
                                      UCBRF_0+UCBRS_6,
                                      UCBRF_0+UCBRS_7,
                                      UCBRF_0+UCBRS_7};
    if(baud_rate_choice<NUM_BAUDS)
    {
        // Habilita os pinos para transmissao serial UART
        P1SEL2 = P1SEL = RX+TX;
        // Configura a transmissao serial UART com 8 bits de dados,
        // sem paridade, comecando pelo bit menos significativo,
        // e com um bit de STOP
        UCA0CTL0 = 0;
        // Escolhe o SMCLK como clock para a UART
        UCA0CTL1 = UCSSEL_2;
        // Define a baud rate
        UCA0BR0 = BRs[baud_rate_choice];
        UCA0BR1 = 0;
        UCA0MCTL = MCTLs[baud_rate_choice];
    }
}

// Faz um atraso em milisegundos
void Atraso_ms(volatile unsigned int ms)
{

    TACCR0 = 1000-1;
    TACTL = TACLR;
    TACTL = TASSEL_2 + ID_0 + MC_1;
    while(ms--)
    {
        while((TACTL&TAIFG)==0);
        TACTL &= ~TAIFG;
    }
    TACTL = MC_0;
}
```

```c
// Compara duas strings e retorna 1 caso o resultado seja positivo
int cmp_str(char str1[], char str2[])
{
    int i;
    for(i=0; (str1[i]!='\0')&&(str2[i]!='\0'); i++)
    {
        if(str1[i]!=str2[i])
            return 0;
    }
    if(str1[i]!=str2[i])
        return 0;
    else
        return 1;
}

// Compara dois caracteres e retorna 1 caso o resultado seja positivo
int cmp_char(char str1, char str2)
{
    if(str1!=str2)
        return 0;
    else
        return 1;
}

//Parametros da Funcao
// Passos numero a ser andado
//direção 1- Horario // 0-Antihorario
//Motor Escolha do motor pelos defines
void stepMotor (int Passos, int direcao, int Motor){

    int x;
    switch(Motor){
    case 0: // Motor X
      Atraso_ms(5);
      if(direcao==0){
          P1OUT &= ~DirecaoX;
      }
      else{
          P1OUT |= DirecaoX;
      }
      for(x = 0; x < Passos; x++) {
          P1OUT |= StepX;
          Atraso_ms(PERIODO/2);
          P1OUT &= ~StepX;
          Atraso_ms(PERIODO/2);
      }
    break;

    case 1 :// Motor Y
        Atraso_ms(5);
        if(direcao==0){
            P1OUT &= ~DirecaoY;
        }
        else{
            P1OUT |= DirecaoY;
        }
        for(x = 0; x < Passos; x++) {
            P1OUT |= StepY;
            Atraso_ms(PERIODO/2);
            P1OUT &= ~StepY;
            Atraso_ms(PERIODO/2);
        }
    break;
    case 2 :// Motor Z
        Atraso_ms(5);
        if(direcao==0){
            P1OUT &= ~DirecaoZ;
        }
        else{
            P1OUT |= DirecaoZ;
        }
        for(x = 0; x < Passos; x++) {
            P1OUT |= StepZ;
            Atraso_ms(PERIODO/2);
            P1OUT &= ~StepZ;
```

```
          Atraso_ms(PERIODO/2);
        }
    break;
    default: break;
    }
}


//---------------------- Realização dos comandos do motor -------------------------
void processCommand(char str[]){

    unsigned int cmd,i;
    float paralist[7] = {0,0,0,-1,0,0,0};
    int paraexist = 0;

    paralist[0]=get_position(str,MOTORX);
    paralist[1]=get_position(str,MOTORY);
    paralist[2]=get_position(str,MOTORZ);
    paralist[3]=get_position(str,POSF);
    paralist[4]=get_position(str,POSP);
    paralist[4]=(int)paralist[4];
    paralist[5]=get_position(str,POSI);
    paralist[6]=get_position(str,POSJ);
    cmd=0;
    i=0;
    while (cmp_char(str[i+1],' ')!=1){
          if (str[i+1]=='\0'){
              break;
          }
          cmd = cmd*10 + str[i+1] - '0'; // Convertendo em int a string
          i++;
    }

    if(str[0]=='G'){
        switch(cmd){
            case  0: // move in a line
            case  1: // move in a line
                if(mode_abs){

                    movea(paralist[0],paralist[1]);
                    headera(paralist[2]);
                  }else{
                    mover(paralist[0],paralist[1]);
                    headerr(paralist[2]);
                  }
             break;
             case  3:
                  Circle(paralist[0],paralist[1],paralist[5],paralist[6],cmd - 2);
                  break;
             case  4:
                  Atraso_ms(paralist[4]);
                  break;  // wait a while
             case 28:
                    reset();
                    movea(0,0);
                    break;
             case 90:
                    mode_abs = 1;  break;  // absolute mode
             case 91:
                    mode_abs = 0;  break;  // relative mode
             case 92:  // set logical position
                    if(paraexist > 0){
                      if((paraexist & 1) == 1){
                        x_pos = paralist[0];
                      }
                      if((paraexist & 2) == 2){
                        y_pos = paralist[1];
                      }
                      if((paraexist & 4) == 4){
                        z_pos = paralist[2];
                      }
                    }else{
                      x_pos = 0;
                      y_pos = 0;
                      z_pos = 0;
                    }
                    break;
                  default:  break;
        }
    }else if(cmp_char(str[0],'M')==1){
        switch(cmd){
            case 17:  // turns on power to steppers (releases the grip)
                Enable_Motor(1);
              break;
             case 18:  // turns off power to steppers (releases the grip)
                 Enable_Motor(0);
              break;
             case 100:  help();  break;
             case 114:  where();  break;  // prints px, py, fr, and mode.
             default:  break;
           }
        }
    }

void reset(){
  headerr(z_pos);
}

void Circle(float x,float y, float i, float j, int dir){
  if(i == j && j == 0)
    movea(x,y);


  float centx, centy;

  // Centre coordinates are always relative
  centx = i + x_pos/MUL;
  centy = j + y_pos/MUL;
  float angleA, angleB, angle, radius, length, aX, aY, bX, bY;

  aX = (x_pos/MUL - centx);
  aY = (y_pos/MUL - centy);
  bX = (x/MUL - centx);
  bY = (y/MUL - centy);

  if (dir == 0) { // Clockwise
    angleA = atan2(bY, bX);
    angleB = atan2(aY, aX);
  } else { // Counterclockwise
    angleA = atan2(aY, aX);
    angleB = atan2(bY, bX);
  }

  // Make sure angleB is always greater than angleA
  // and if not add 2PI so that it is (this also takes
  // care of the special case of angleA == angleB,
  // ie we want a complete circle)
  if (angleB <= angleA) angleB += 2 * M_PI;
  angle = angleB - angleA;
  if(angle == 0)
    return;

  radius = sqrt(aX * aX + aY * aY);
  length = radius * angle;
  int steps, s, ss;
  steps = (int) ceil(length / 0.1);

  float nx, ny;
  for (s = 1; s <= steps; s++) {
    ss = (dir == 1) ? s : steps - s; // Work backwards for CW
    nx = centx + radius * fastsin(angleA + angle * ((float) ss / steps) + (M_PI/2));
    ny = centy + radius * fastsin(angleA + angle * ((float) ss / steps));
    movea(floor(nx*MUL+0.5), floor(ny*MUL+0.5));

  }
}

float fastsin(float x)
{
  while(x > M_PI)
```

```
  while(x > M_PI)
    x -= 2*M_PI;
  while(x < -M_PI)
    x += 2*M_PI;

  float B = (4 / M_PI);
  float c = -4 / (M_PI * M_PI);

  //return 0;
  return B * x + c * x * ((x < 0) ? -x : x);
}
void movea(float x,float y){
  int dx = (int)(x - x_pos);
  int dy = (int)(y - y_pos);
  mover(dx,dy);
}

void headera(int z){
  int dz = z_pos - z;
  headerr(dz);
}

void headerr(int z){
    stepMotor(z,0,MOTORZ);
    z_pos -= z;
}

void mover(float x,float y){
  int dirx;
  int diry;
  int i;
  dirx= 1;
  diry= 1;
  if(x == y){
    for(i = 0; i < (int)x; i++){
      stepMotor(dirx,0,MOTORX);
      x_pos += dirx;
      stepMotor(diry,1,MOTORY);
      y_pos += diry;
    }
  }else if(x > y){
    float acc = 0;
    int flag = false;
    for(i = 0; i < (int)x; i++){
      stepMotor(dirx,0,MOTORX);
      x_pos += dirx;
      if(flag){
        stepMotor(diry,1,MOTORY);
        y_pos += diry;
        flag = false;
      }
      acc += y / x;
      if(acc > 0.5){
        flag = true;
        acc--;
      }
    }
  }else{
    float acc = 0;
    int flag = false;
    for(i = 0; i < (int)y; i++){
      stepMotor(diry,1,MOTORY);
      y_pos += diry;
      if(flag){
        stepMotor(dirx,0,MOTORX);
        x_pos += dirx;
        flag = false;
      }
      acc += x / y;
      if(acc > 0.5){
        flag = true;
        acc--;
      }
    }
  }
}
```

```
void Enable_Motor(int a){
    if(a==0){
       P2OUT &= ~Enable;
    }
    else{
       P2OUT |= Enable;
    }

}

float get_position(char str1[], int t){
    int position=0;
    float resultado=0;
    switch(t){
        case 0:
            position=Acha_char(str1,'X');
            break;
        case 1:
            position=Acha_char(str1,'Y');
            break;
        case 2:
            position=Acha_char(str1,'Z');
            break;
        case 3:
            position=Acha_char(str1,'F');
            break;
        case 4:
            position=Acha_char(str1,'P');
            break;
        case 5:
            position=Acha_char(str1,'I');
            break;
        case 6:
            position=Acha_char(str1,'J');
            break;
        default:  break;
    }
    resultado = stof(str1,position);
    return resultado;
}

float stof(char str1[], int position){
    float result= 0.0f;
    int dotpos = 0;
    int len = 8;
    int n;
    if (position==0){
        return result;
    }
    for (n = position; n <= position+7 ; n++)
    {
      if (cmp_char(str1[n],'.'))
      {
        dotpos = position+len - n  - 1;
      }
      else
      {
        result = result * 10.0f + (str1[n]-'0');
      }
    }
    while ( dotpos--)
    {
      result /= 10.0f;
    }
    return result;

}

int Acha_char(char str1[], char str2){
    int i, flag;
    flag=0;
    for(i=0; (str1[i]!='\0'); i++)
    {
        if(cmp_char(str1[i],str2)==1){
            flag=i;
```

```
602             flag++;
603             break;
604         }
605     }
606     return flag;
607
608 }
609
610 void where() {
611     Send_String("\n\rCurrent State:\n\r");
612     Send_String("X:");
613     Send_Int(x_pos);
614     Send_String("\n\rY:");
615     Send_Int(y_pos);
616     Send_String("\n\rZ:");
617     Send_Int(z_pos);
618     Send_String("\n\rFeed Rate:");
619     Send_Int(feed);
620 }
621
622 void help() {
623     Send_String("\n\rCommands:\n\r");
624     Send_String("G00 [X(steps)] [Y(steps)] [Z(steps)] [F(feedrate)]; - linear move\n\r");
625     Send_String("G01 [X(steps)] [Y(steps)] [Z(steps)] [F(feedrate)]; - linear move\n\r");
626     Send_String("G04 P[seconds]; - delay\n\r");
627     Send_String("G90; - absolute mode\n\r");
628     Send_String("G91; - relative mode\n\r");
629     Send_String("G92 [X(steps)] [Y(steps)] [Z(steps)]; - change logical position\n\r");
630     Send_String("M18; - disable motors\n\r");
631     Send_String("M100; - this help message\n\r");
632     Send_String("M114; - report position and feedrate\n\r");
633 }
634
635
```