RELATÓRIO FINAL
Plotter para Placas de Circuito Impresso

Hachid Habib Cury

Programa de Engenharia Eletrônica
Faculdade do Gama
Universidade de Brasília – UnB
Brasília, Brasil
hachidcury@gmail.com

Larissa Aidê Araújo Rocha

Programa de Engenharia Eletrônica
Faculdade do Gama
Universidade de Brasília – UnB
Brasília, Brasil
larissa.aide1@gmail.com

***Resumo*— O projeto visa desenvolver um sistema que realiza a impressão meticulosa do layout de circuitos em placas de fenolite ou de fibra de vidro para a criação de placas de circuito impresso, a distância com uso de uma raspberry PI como server .**

Palavras-chave— raspberry pi; PCB; impressão; PCI;

I. JUSTIFICATIVA

Quando se trata de produtos eletrônicos principalmente os que fazem uso de circuitos integrados, as placas de circuito impresso (PCIs) são indispensáveis, elas desenvolvem um grande papel para o desenvolvimento mais compacto e seguro de circuitos eletrônicos.

O circuito impresso consiste em uma base isolante como fenolite ou fibra de vidro que são cobertas por uma fina película de cobre, constituindo as trilhas condutoras, revestidas por ligas à base de ouro, níquel, estanho chumbo, ou verniz orgânico (OSP), entre outras, que representam o circuito onde serão soldados e interligados os componentes eletrônicos.

Atualmente, as placas de circuito impresso podem ser fabricadas de uma forma mais técnica (em indústrias por máquinas) como também artesanalmente (por *makers* de PCIs em sua própria casa). Dessa forma, nosso projeto está focado em desenvolver um sistema que atenda na produção de placas produzidas artesanalmente.

Apesar de podermos produzir nossa própria PCI em casa, existe uma parte crucial no processo de fabricação que requer bastante habilidade e cuidado, que é a parte de passar o layout do circuito para a placa. Na maior parte dos casos, qualquer erro nessa etapa da fabricação da PCI pode acarretar na perda total do circuito.

Comumente, as formas mais utilizadas pelos *makers* na etapa de desenho são: desenhar diretamente na placa com
uma caneta de tinta resistente, que requer bastante habilidade e cuidado ou por meio da fototransferência que requer vários processos e um longo tempo de espera até que o desenho seja completamente transferido para a placa.

Dessa forma nosso projeto está focado em otimizar o tempo de usuários que tenham a necessidade frequente de fabricar placas de circuito impresso com uma qualidade e exatidão do layout do circuito, por meio de um plotter de caneta.

Existem vários plotters de caneta no mercado destinados ao uso de desenhos técnicos e precisos, porém pouco são fabricados com objetivo de fabricação de PCB. Durante um levantamento bibliográfico dos maquinários de mercado não conseguimos encontrar nenhum que estivesse disponível para venda com esse intuito, e muito menos um que possua uma interface que ofereça ao usuário a possibilidade de visualizar e modificar seu projeto antes da impressão. O preço encontrado para máquina XY Plotter no mercado nacional está na faixa de 1500 a 3000 como mostrado na figura 1.

Figura 1.1 Plotter XY retirado do Mercado Livre

II. OBJETIVOS

O objetivo geral deste projeto é fabricar uma mesa linear XY que produzirá o desenho do layout do circuito sobre a própria placa de fenolite, deixando-a pronta para a etapa de corrosão. Será utilizado uma raspberry pi com um sistema operacional embarcado para construir um maquinário completo que permita ao usuário mandar seu arquivo via telegram e acompanhar sua impressão.

III. REQUISITOS

Para o correto funcionamento será necessário a utilização de uma raspberry pi 3 para enviar os arquivos de impressão e dar comandos para um microcontrolador que fará o controle dos motores de passo do nosso maquinário. Também será necessário a construção de uma mesa CNC que trabalhe em três dimensões e utilize motores de passo.

O microcontrolador utilizado para fazer o controle dos motores será o ESP32 pois há uma maior precisão ao controlar os motores que serão utilizados para fazer o desenho.

Nosso produto contará com uma interface que fará a interação humano-máquina via wi-fi, assim por meio do e-mail ou telegram, o plotter entrará em contato com o usuário que poderá validar a impressão e acompanhar por meio de uma câmera se sua placa está pronta ou não.

Além disso o Plotter terá a capacidade de verificar se o usuário colocou a placa na máquina, se sim ele realizará a impressão, caso contrário ele entrará em contato com o usuário para que o mesmo posicione a placa.

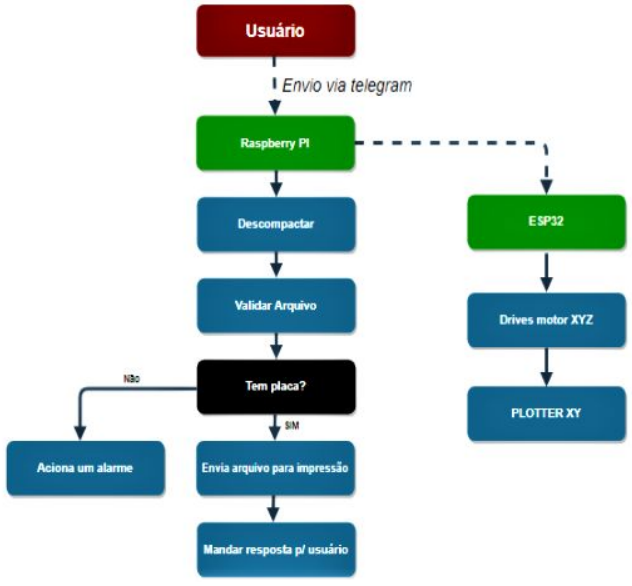

Figura 3.1 Diagrama simplificado do funcionamento do protótipo

IV. BENEFÍCIOS

Por meio do Plotter para Placa de Circuito Impresso, o usuário do produto poderá obter o desenho do layout do circuito na placa de uma forma mais precisa e diligente, acarretando assim a minimização de erros na etapa de desenho. Tudo isso a distância, não necessitando então que o usuário fique presente durante todo o processo de impressão e no final sendo notificando quando o mesmo terminar

V. MATERIAIS UTILIZADOS

A fim de obter um melhor planejamento da parte física do projeto, o mesmo foi dividido em duas partes: *parte mecânica* e *parte eletrônica*.

5.1 Parte Mecânica

- Eixo retificado (Guia Linear):
  2x 400mm
  2x 300mm
  2x 70mm
- 10 Rolamentos lineares lm8uu
- 9 Rolamentos 604zz (4x12x4mm)
- 2 Metros de correia GT2 20 dentes
- 2 Polias GT2 20 dentes
- 1 Servo MG996R
- 4 Abraçadeiras de nylon
- Parafusos
  5 Parafusos M4x20mm
  10 Parafusos M3x8mm
  8 Parafusos M3x16mm
  11 Parafusos M3x30mm
  2 Barras roscadas 7/16pol de 420mm
- Porcas
  8 porcas 7/16pol
  7 porcas M4
  23 porcas M3
- Peças auxiliares mesa CNC (Impressas na impressora 3D)

5.2 Parte Eletrônica

- 1 Raspberry Pi3
- 1 Esp32 Dev
- 1 Webcam
- Sensor de Luminosidade LDR 5mm
- 3 Drivers DRV8825
- 1 Regulador de tensão stepdown
- 2 Motores de Passo Nema 17HS
- 1 Micro Servo Motor Tower
- Jumpers para conexão
- 1 botão switch
- 3 botões push
- 3 resistores 10k ohm

- Capacitor 220 µF 25V
- Protoboard
- Fonte de Computador de 200W

VI. DESCRIÇÃO DE HARDWARE

A montagem da parte física foi feita da seguinte maneira:

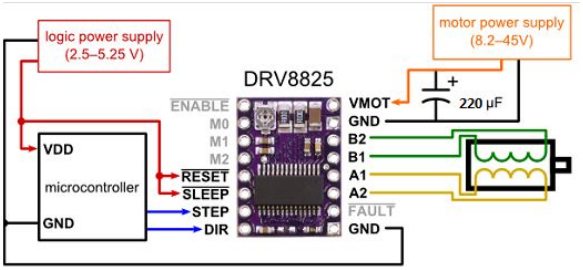

Figura 6.1 Conexão detalhada do Drive DRV8825 ao motor e ao microcontrolador.

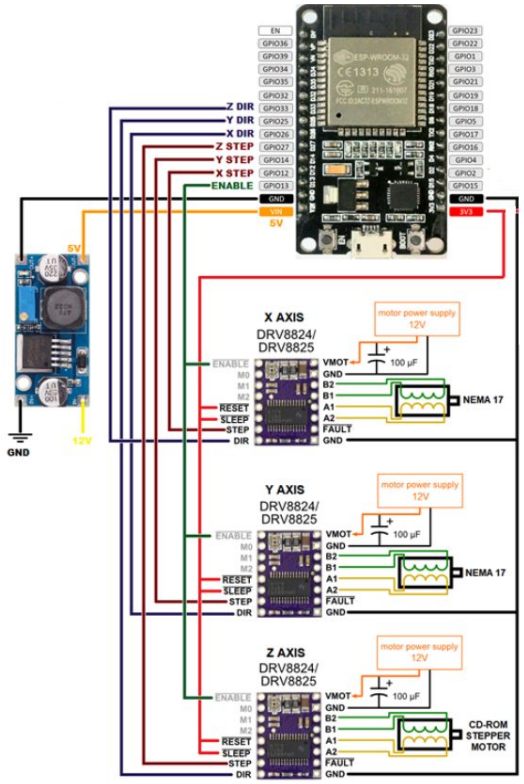

Figura 6.2: Conexão de cada Drive DRV8825 na Esp32

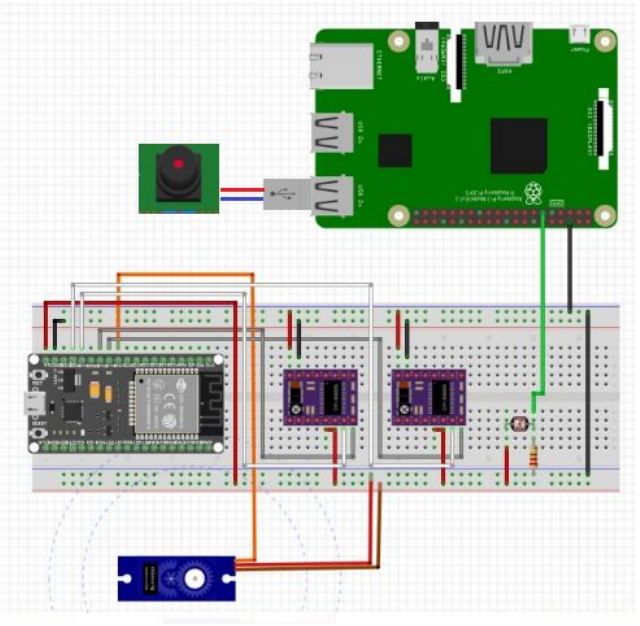

Figura 6.3 Conexões feitas a Raspberry Pi

VII. ESTRUTURA FÍSICA DO PROJETO

Para que a Plotter consiga realizar os desenhos é necessário o uso de 3 motores, 2 Motores de passo para determinar as direções da caneta (motor da direita e motor da esquerda) e um outro Micro Servo Motor para determinar quando a caneta irá para baixo desenhando na placa ou para cima quando não é mais necessário que a caneta desenhe na placa.

As peças foram projetadas para serem impressas na impressora 3D, onde o seu layout foi baseado em peças presentes em máquinas de CNC. O protótipo foi pensado para ser compacto, leve e fácil para transporte.

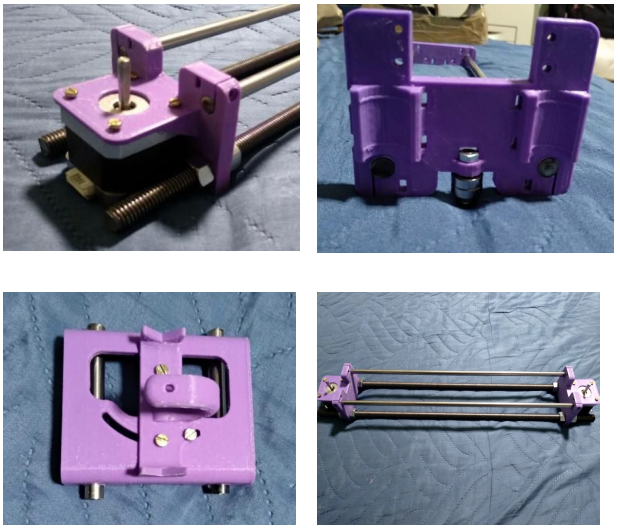

Figura 7.1 Estrutura Parcial do Projeto

Por fim, com todas as peças impressas, foi possível finalizar a montagem da Plotter, a figura abaixo mostra como ficou a estrutura.

Figura 7.2 Estrutura Final do Projeto

VIII. DESCRIÇÃO DE SOFTWARE

Instalou-se na Raspberry Pi o sistema operacional Raspbian que é uma distribuição baseada no sistema operacional Debian versão *Stretch* com interface gráfica, baseada no ARM hard-float, sendo um porte da arquitetura Wheezy, otimizada para o conjunto de instruções ARMv6. Para facilitar a instalação utilizou-se o instalador NOOBS.

Após a instalação do sistema operacional na Raspberry, os códigos foram feitos separadamente a fim de testar alguns dos requisitos do projeto.

a) Código comunicação via Telegram com a Raspberry Pi

O Telegram é um serviço de mensagens instantâneas multi-plataforma totalmente grátis (e open-source!), isso significa que pode ser usado no Linux. Além disso, outro recurso bastante importante para o nosso projeto fornecido pelo Telegram Bot API é a capacidade de criar um bot, um bot nada mais é que um programa que atua em uma conta de usuário no Telegram que não é operada por um humano, mas sim, por um robô.

Figura 8.1: Integração Raspberry e Telegram

Essa interface HTTP, libera ao desenvolvedor uma chave única de autenticação (chamada *token*) toda vez que um bot é criado, é por meio desse *token* que é possível desenvolver um código para obter uma interação chat com o usuário. Dessa forma, usaremos o bot da Figura 7.2 para desenvolver o código de comunicação do usuário com o protótipo.

Figura 8.2: Bot criado para o projeto

Por não atender os requisitos do projeto e por ser bastante complexa ao que se refere no download de arquivos, a biblioteca *tgbot.h* utilizada em C++ foi substituída pela biblioteca *ShellBot* baseada na linguagem Shell script, que é uma linguagem de script usada em vários sistemas operativos (operacionais), com diferentes dialetos, dependendo do interpretador de comandos utilizado.

Dessa forma, essa parte do projeto foi desenvolvida na linguagem Shell script utilizando o interpretador de comandos bash, possibilitando a criação de uma espécie de menu, que facilita a interação do produto com o usuário, além disso, foi possível fazer o download de documentos em .zip (documento esse enviado pelo usuário que contém os arquivos do layout de placa de circuito impresso).

As principais funções e comandos utilizados no código foram:

- Declaração da biblioteca usada e também a especificação do Token do Bot:

```
# Importando API
source ShellBot.sh

# Token do bot
bot_token = '<token bot>'

# Exemplo trabalhando com o modo de retorno do tipo:
value ShellBot.init --token "<token bot>" --monitor
ShellBot.username
```

- Função utilizada para mandar a mensagem do Bot

para o usuário:

ShellBot.sendMessage --chat_id "${message_chat_id[$id]}" --text "

No código, a função ShellBot.sendMenssage foi associada a alguns comandos que o usuário pode enviar ao Bot, assim a partir de cada um desses comandos o bot é capaz de enviar uma mensagem específica. Os comandos são:

- */start*: Quando o usuário envia esse comando o Bot envia uma mensagem de iniciação dando boas vindas e algumas instruções do funcionamento do Bot.
- */plotter*: Ao enviar esse comando o usuário informa se deseja realmente fazer uma impressão, assim o código retorna uma mensagem solicitando o arquivo para impressão e pede para que o mesmo posicione a placa na impressora.
- */ready:* Ao enviar esse comando o usuário confirma a impressão do arquivo. É dentro dele que a Raspberry Pi faz uma verificação se a placa de fenolite foi ou não posicionada na impressora a partir de um sensor de luminosidade LDR.
- */photo:* Quando o usuário envia esse comando, a mensagem retorna uma foto do processo de impressão.
- */help:* Ao enviar esse comando, o código retorna uma mensagem contendo algumas informações sobre a Plotter e o formato de impressão do arquivo.

- Função utilizada para fazer o download do documento .zip:

ShellBot.downloadFile --file_path "$out" --dir $HOME

- Função utilizada para enviar a foto do processo de impressão para o usuário:

ShellBot.sendPhoto --chat_id ${message_chat_id[$id]} --photo '@/home/pi/testebot/photo/foto.jpg'

b) Código Interação Raspberry Pi com a Esp32

No projeto precisamos que cada traço do desenho seja minimamente preciso. Apesar da Raspberry conseguir controlar bem o motor, por ser um computador e não um microcontrolador esse controle não é tão apurado assim.

Dessa forma o controle dos motores serão feitos por meio da Esp32, que é dotado de um poderoso microcontrolador de 32 bits com WiFi integrado e possui integração com Arduino IDE tendo assim suporte a muitas bibliotecas compatíveis.

A comunicação serial foi adotada para fazer o processo de enviar dados bit a bit entre a Raspberry e a Esp32, por ser mais simples e mais fácil de ser implementada do que a comunicação SPI. A Esp32 receberá os arquivos de impressão da Raspberry, que foram transmitidos anteriormente via telegram, mas para que possa haver essa comunicação, foi necessário criar quatro funções, uma para verificar se chegou um arquivo .ZIP, uma para descompacta-lo, outra para excluir o arquivo .ZIP e por fim uma para limpar tudo que estava contido nele depois do descompactamento.

As funções a seguir estão contidas em Anexos com título de Código da Raspberry para tratar os arquivos recebidos

- *Obter_arq(char linha[100])*: é uma função que lê todos os arquivos .ZIP das pasta "/testebot" e joga todos arquivos em um arquivo txt, após isso ele abre esse arquivo, lê a primeira linha (linha que contém o título do primeiro arquivo .ZIP que será impresso), após isso ele exclui o arquivo txt criado e retorna o título na variável de char que foi passado.
- *Descompactar(char Linha[100])*: utiliza a variável que contém o título do arquivo ZIP para descompactá-lo e enviá-lo para a pasta "arquivodescompactado".
- *RemoverZip(char Linha[100]):* utiliza a variável que contém o título do arquivo ZIP para excluí-lo.
- *Remover(void)*: remove a pasta "arquivodescompactado" com todos os arquivos que foram descompactados e logo após cria novamente a pasta "arquivodescompactado".

c) Código Esp 32 para controlar o Motor de Passo

Os motores de passo permitem um posicionamento preciso com facilidade, acoplando ao DRV8825 que conseguimos realizar a divisão dos micro passos do motor para que alcance movimentos muito pequenos e ainda mais precisos, que são fundamentais na montagem de CNCs e na montagem do nosso protótipo.

O código para aferir o comportamento do motor e do drive foi realizado na plataforma IDE do arduino, como o código é bastante simples nenhuma biblioteca foi utilizada.

Para isso foi definido as constantes que utilizaremos para indicar os pinos STEP e DIR do driver, onde STEP funcionará como um PWM e DIR indicará a direção em que o motor deverá girar, 1 para a esquerda e 0 para a direita. Assim, com os pinos dos motores definidos, foi possível criar funções que determinam o movimento dos dois motores responsáveis pela direção.

- *frente()*; Para que a Plotter faça um movimento para a frente com a caneta é necessário que o DIR do motor da direita esteja em nível lógico baixo e o DIR da esquerda em nível lógico alto.
- *tras();* Para que a Plotter faça um movimento para trás com a caneta é necessário que o DIR do motor da direita esteja em nível lógico alto e o DIR da esquerda em nível lógico baixo.
- *direita();* Para que a Plotter faça um movimento para a direita com a caneta é necessário que o DIR dos dois motores estejam em nível lógico baixo.

- *esquerda();* Para que a Plotter faça um movimento para a direita com a caneta é necessário que o DIR dos dois motores estejam em nível lógico alto.

Do mesmo modo, foram criadas funções para o terceiro motor, responsável pelo movimento de levantar e abaixar a caneta.

- *Levanta();* Para levantar a caneta o DIR do motor C deve está em alto.
- *Escreve();* Para que a Plotter abaixe a caneta e comece desenhar na placa o DIR do motor C deve está em alto.

Com as funções criadas foi possível fazer testes capazes de aferir o tamanho da área total que a Plotter é capaz de desenhar, que correspondente a uma área de 29,5cm x 19 cm - 1480 passos x 957 passos, como também a sua diagonal que equivale a 1762.5 passos - 35 cm. Após o êxito dos testes foram criadas 3 funções de desenhos geométricos no código que são plotados após o usuário dá o comando via telegram.

- *atotal*(); Função que fornece o desenho da área total em que a plotter pode desenhar. 29,5cm x 19 cm - 1480 passos x 957 passos
- *quad*(); Função que fornece o desenho de um quadrado 9.5cm x 9.5cm - 480 passos x 480 passos
- *tri*(); Função que fornece o desenho de um triângulo.

Apesar dos comandos acima serem eficientes para desenhos com poucos traços, ainda sim não são suficientes para desenhar um layout mais complexo de PCB. Dessa forma, utilizou-se a firmware GRBL, que é um excelente interpretador de g-code, com capacidade de processar junto dos tradicionais eixos, o controle de um spindle e flúido de usinagem. O código GRBL desenvolvido para a ESP32 utilizado no projeto pode ser acessado por meio do Github da Bibliografia [20].

IX. CONSIDERAÇÕES FINAIS

De maneira geral o projeto atendeu todos os requisitos esperados, mostrando um bom resultado e precisão na impressão do layout. O firmware GRBL foi o ponto chave para o resultado final, pois permitiu fazer um melhor controle dos motores pela Esp32.

O custo final do projeto se comparado com as EleksMaker disponíveis no mercado se mostrou um pouco mais acessível, visto que a Plotter aqui projetada faz o uso de uma Raspberry Pi, dessa forma por mais que possa haver produtos semelhantes no mercado para realizações de desenhos em papéis, nenhum deles oferece o benefício direto de se fazer a impressão em uma PCB, como também realizar e acompanhar essa impressão remotamente via Telegram.

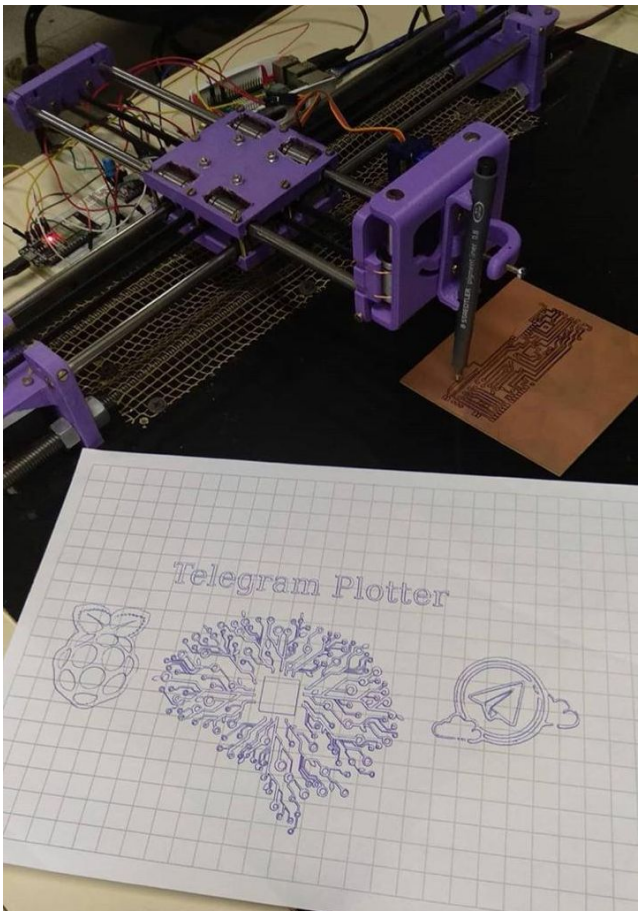

Figura 8.3: Resultado Final da Impressão

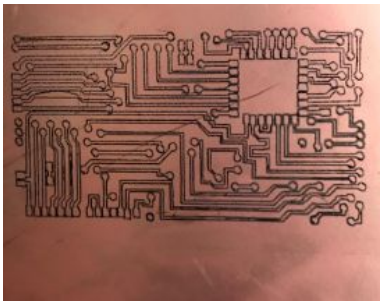

Figura 8.4: Resultado Final da Impressão

REFERÊNCIAS

[1] Como fazer suas propiras PCBs https://blog.fazedores.com/como-fazer-suas-proprias-pcbs-placas-de-circuito-impresso/ <Acesso em: 23/03/2019>

[2] Como Placas de Cicuito Impresso são produzidas https://www.tecmundo.com.br/como-e-feito/18501-como-as-placas-de-circuito-impresso-sao-produzidas.htm <Acesso em: 26/03/2019>

[3] Placas de Circuito Impresso http://www.newtoncbraga.com.br/index.php/almanaque-tecn

ologico/205-p/7551-placas-de-circuito-impresso-alm345<Acesso em: 27/03/2019>

[4] Conceitos Fundamnetais Sobre Placas De Circuito Impresso http://www.eletrica.ufpr.br/mehl/te232/textos/PCI_Conceitos_fundamentais.pdf <Acesso em: 27/03/2019>

[5] Plotter a venda: anúncio retirado do mercado livre https://produto.mercadolivre.com.br/MLB-1190535869-kit-de-robotica-makeblock-xy-plotter-robot-kit-_JM <Acesso em: 27 /03/2019>

[6] Primeira impressora 3D controlada por Raspberry Pi do mundo https://3dprint.com/16060/raspberry-pi-3d-printer/ <Acesso em: 25/03/2019>

[7] X-Y Plotter com Arduino https://www.instructables.com/id/X-Y-Plotter-1/ <Acesso em: 25/03/2019>

[8] ARDUINO por mim mesmo CNC (Plotter) https://www.instructables.com/id/ARDUINO-by-Myself-Mini-CNC-Plotter/ <Acesso em: 25/03/2019>

[9] DIY Mini Maquina de CNC https://www.instructables.com/id/DIY-MINI-CNC-DRAWING-MACHINE/ <Acesso em: 25/03/2019>

[10] Projeto de braço robô desenhista é sucesso de financiamento coletivo https://www.tecmundo.com.br/kickstarter/114031-projeto-braco-robo-desenhista-sucesso-financiamento-coletivo.htm <Acesso em: 25/03/2019>

[11] Raspberry Pi + Thermal Printer https://taylorhokanson.com/2017/02/27/raspberry-pi-thermal-printer/ <Acesso em: 25/03/2019>

[12] Utilização da Raspberry PI em processamento de imagens de satélite http://pequenoscomputadores.blogspot.com/2012/12/utilizacao-da-raspberry-pi-em.html <Acesso em: 01/05/2019>

[13] RASPBERRY PI SERVIDOR DE WEBCAM https://roboott.wordpress.com/2016/01/07/raspberry-pi-servidor-de-webcam/ <Acesso em: 01/05/2019>

[14] Framework Python para a API Telegram Bo thttps://telepot.readthedocs.io/en/latest/ <Acesso em: 01/05/2019>

[15] “IOT feito fácil”: Brincando com o ESP32 no Arduino IDE https://mjrobot.org/2017/09/26/iot-feito-facil-brincando-com-o-esp32-no-arduino-ide/ <Acesso em: 01/05/2019>

[16] DRV8825: Como Utilizar Com Arduino e Motor de Passo http://blog.baudaeletronica.com.br/drv8825-com-arduino-e-motor-de-passo/ <Acesso em: 02/05/2019>

[17] Tutorial Arduino Drv8825 Motor de Passo https://jgamblog.wordpress.com/2016/09/24/tutorial-arduino-drv8825-motor-de-passo/ <Acesso em: 02/05/2019>

[18] FAZENDO UM BOT PARA TELEGRAM EM PYTHON https://juliarizza.wordpress.com/2016/08/06/fazendo-um-bot-para-telegram-em-python/ <Acesso em: 02/05/2019>

[19] Biblioteca Shell Bot. <https://github.com/shellscriptx/ShellBot/wiki> Acesso em: 03/06/2019

[20] Firmware GRBL para ESP32 <https://github.com/bdring/Grbl_Esp32> Acesso em: 15/06/2019

IX. Anexo

Código da Raspberry para comunicação com o telegram

```
#!/bin/bash

#
# Importando API
source ShellBot.sh

# Token do bot
bot_token='xxxxxxxxxxxxxxxxxxxxxxxxx''

# Exemplo trabalhando com o modo de retorno do tipo: value
ShellBot.init --token
"xxxxxxxxxxxxxxxxxxxxxxxxx" --monitor
ShellBot.username

while :
do
    # Obtem as atualizações
    ShellBot.getUpdates --limit 100 --offset
$(ShellBot.OffsetNext) --timeout 30

    # Lista o índice das atualizações
    for id in $(ShellBot.ListUpdates)
    do
    # Inicio thread
    (
        # Obtendo o id da foto com maior
resolução.

photo_id=${message_photo_file_id[$id]##*|}

documento_id=${message_document_file_id[$id]##
*|}

        # Executa os métodos se o arquivo
enviado é uma imagem.
        if [[ $photo_id ]]; then
            # Capturando as informações da
foto.
            out=$(ShellBot.getFile --file_id
$photo_id)

            # Extraindo o caminho absoluto do
retorno do método.
            out=${out##*|}

            # Baixando o arquivo para o home
do usuário.
            ShellBot.downloadFile --file_path
"$out" --dir $HOME/testebot
        fi
        if [[ $documento_id ]]; then
            # Capturando as informações do
documento.
            out=$(ShellBot.getFile --file_id
$documento_id)

            # Extraindo o caminho absoluto do
retorno do método.
            out=${out##*|}

            # Baixando o arquivo para o home
do usuário.
            ShellBot.downloadFile --file_path
"$out" --dir $HOME/testebot
            ShellBot.sendMessage --chat_id
"${message_chat_id[$id]}" --text "Envie um
comando /read para imprimir este documento."

        fi
    ) & # Utilize a thread se deseja que o bot
responda a várias requisições simultâneas.
    done
    case ${message_text[$id]} in
                        '/start') # comando teste

ShellBot.sendMessage --chat_id
"${message_chat_id[$id]}" --text "Bem vindo ao
Bot da PCB Plotter, eu fui criado para te auxiliar
no seu processo de impressão!\n\nAqui estão alguns
comandos que vão te ajudar:\n\n/plotter -
Confirma que deseja fazer a impressão\n\n/photo -
Você pode obter uma foto de como anda o processo
de impressão sempre que desejar.\n\n/help - Caso
tenha alguma dúvida de como funciona a Plotter
esse comando utilize esse comando.
                        "
                        ;;

                        '/plotter') # comando teste
                            ShellBot.sendMessage
--chat_id "${message_chat_id[$id]}" --text "Para
que a Plotter possa fazer a impressão, você deverá
primeiramente posicionar a sua placa no centro da
impressora (Tampando o sensor de identificação da
Placa).\nFeito isso, basta apenas enviar o seu
aquivo (exclusivamente no formato .zip) aqui mesmo
no telegram."
                        ;;

                        '/read') # comando teste
                            ShellBot.sendMessage
--chat_id "${message_chat_id[$id]}" --text "Seu
arquivo está sendo imprimido."
                        ;;

                        '/photo') # comando teste
                            ShellBot.sendMessage
--chat_id "${message_chat_id[$id]}" --text
"Tirando
```

```
fswebcam -r 1280x720 --no-banner
/home/pi/testebot/photo/fotinha.jpg
                                # Envia o álbum contendo
as mídias.
                                ShellBot.sendPhoto
--chat_id ${message_chat_id[$id]} --photo
'@/home/pi/testebot/photo/fotinha.jpg'
                            ;;

                            '/help') # comando teste
                                ShellBot.sendMessage
--chat_id "${message_chat_id[$id]}" --text "Esse
comando foi feito para sancionar as principais
duvidas dos usuários da PCB Plotter.\n\n- A
Plotter é uma impressora de circuito impresso, que
usufrui do firmware GRBL, dessa forma dentro do
arquivo .zip enviado pelo telegram, deverão conter
os arquivos de layout da placa adquiridos pelo
software de sua escolha (Protheus, Eagle...),
lembrando que tais arquivos deverão está no
formato aceito pelo GRBL, para que não dê erro
na hora da impressão.\n\n-A Plotter contém um
sesnor que identifica se a placa de impressão foi
posicionada no centro da impressora, caso o usuário
não posicione a placa, a Plotter não executará a
impressão e enviará uma mensagem ao usuário.\n"
                            ;;
                        esac

done
```

Código da Raspberry para tratar os arquivos recebidos

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void Descompactar(char Linha[100]){
   char cmd[200];
   sprintf(cmd, "unzip -d
/home/pi/testebot/arquivodescompactado
/home/pi/testebot/%s ", Linha);
   system(cmd);
}

void Remover(void){
system("rm -rf/home/pi/testebot/arquivodescomp
actado; mkdir/home/pi/testebot/arquivodescompa
ctado");
}
void RemoverZip(char Linha[100]){
   char cmd[200];
   sprintf(cmd, "rm %s", Linha);
   system(cmd);
}

void Obter_arq(char linha[100]){
   FILE *arq;
   char *result;

    //Escreve o nome dos arquivos .zip que estão na
pasta

    system("ls /home/pi/testebot | grep .zip >>
ArqTeste.txt");


    // Abre um arquivo TEXTO para LEITURA

    arq = fopen("ArqTeste.txt", "r");

    if (arq == NULL)  // Se houve erro na
abertura

    {

      printf("Problemas na abertura do
arquivo\n");

      return;

    }

    result = fgets(linha, 50, arq);



    fclose(arq);

    system("rm ArqTeste.txt");

}


int main()

{
 char Linha[100];

 Obter_arq(Linha);
 Descompactar(Linha);
 RemoverZip(Linha);
 Remover();

 return 0;
}
```

Código da Raspberry para o Sensor de luminosidade LDR

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
delayt = .1
value = 0 # this variable will be used to store the ldr value
ldr = 4
led = 17
GPIO.setup(led, GPIO.OUT) # as led is an output device so that's why we set it to output.
GPIO.output(led, False) # keep led off by default
def rc_time (ldr):
    count = 0

    #Output on the pin for
    GPIO.setup(ldr, GPIO.OUT)
    GPIO.output(ldr, False)
    time.sleep(delayt)

    #Change the pin back to input
    GPIO.setup(ldr, GPIO.IN)

    #Count until the pin goes high
    while (GPIO.input(ldr) == 0):
        count += 1

    return count


#Catch when script is interrupted, cleanup correctly
try:
    # Main loop
    while True:
        print("Ldr Value:")
        value = rc_time(ldr)
        print(value)
        if ( value <= 10000 ):
                print("Lights are ON")
                GPIO.output(led, True)
        if (value > 10000):
                print("Lights are OFF")
                GPIO.output(led, False)
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
```

Código do Microcontrolador

```
// define números de pinos do driver no arduino
const int E_stepPin = 12;
const int E_dirPin = 26;

// define números de pinos do driver no arduino
```

```
const int D_stepPin = 14;
const int D_dirPin = 25;

// define números de pinos do driver no arduino
const int C_stepPin = 27;
const int C_dirPin = 33;

// define números de pinos do driver no arduino
const int Sel1 = 35;
const int Sel2 = 34;
const int imprimindo = 32;

int estado1;
int estado2;

void frente(void)
{
  digitalWrite(D_dirPin,LOW); // Horario
  digitalWrite(E_dirPin, HIGH); //AntiHorário

  digitalWrite(E_stepPin,HIGH);
  digitalWrite(D_stepPin,HIGH);
  delay(10);
  digitalWrite(E_stepPin,LOW);
  digitalWrite(D_stepPin,LOW);
  delay(10);
}
void tras(void)
{
  digitalWrite(E_dirPin,LOW); // Horario
  digitalWrite(D_dirPin, HIGH); //AntiHorário

  digitalWrite(E_stepPin,HIGH);
  digitalWrite(D_stepPin,HIGH);
  delay(10);
  digitalWrite(E_stepPin,LOW);
  digitalWrite(D_stepPin,LOW);
  delay(10);
}
void direita(void){

  digitalWrite(D_dirPin,LOW); // Horario
  digitalWrite(E_dirPin, LOW); //Horário

  digitalWrite(E_stepPin,HIGH);
  digitalWrite(D_stepPin,HIGH);
  delay(10);
  digitalWrite(E_stepPin,LOW);
  digitalWrite(D_stepPin,LOW);
  delay(10);
  }

  void esquerda(void){

  digitalWrite(D_dirPin,HIGH); // AntiHorario
  digitalWrite(E_dirPin, HIGH); //AntiHorário

  digitalWrite(E_stepPin,HIGH);
  digitalWrite(D_stepPin,HIGH);
  delay(10);
  digitalWrite(E_stepPin,LOW);
  digitalWrite(D_stepPin,LOW);
  delay(10);
  }
  void F_Direita(void){

  digitalWrite(D_dirPin,LOW); // Horario

  digitalWrite(D_stepPin,HIGH);
  delay(10);
  digitalWrite(D_stepPin,LOW);
  delay(10);
  }
  void T_Esquerda(void){

  digitalWrite(D_dirPin,HIGH); // AntiHorario

  digitalWrite(D_stepPin,HIGH);
  delay(10);
  digitalWrite(D_stepPin,LOW);
  delay(10);
  }
  void F_Esquerda(void){

  digitalWrite(E_dirPin,HIGH); // AntiHorario

  digitalWrite(E_stepPin,HIGH);
  delay(10);
  digitalWrite(E_stepPin,LOW);
  delay(10);
  }
  void T_Direita(void){

  digitalWrite(E_dirPin,LOW); // Horario

  digitalWrite(E_stepPin,HIGH);
  delay(10);
  digitalWrite(E_stepPin,LOW);
  delay(10);
  }

  void Levanta(void){
  digitalWrite(C_dirPin, HIGH);

  digitalWrite(C_stepPin,HIGH);
  delay(10);
  digitalWrite(C_stepPin,LOW);

  delay(10);
  }
    void Escreve(void){
  digitalWrite(C_dirPin, LOW);
```

```
    digitalWrite(C_stepPin,HIGH);
    delay(10);
    digitalWrite(C_stepPin,LOW);
    delay(10);

  }
   void areatotal(void){
    delay(500);
    for(int x = 0; x < 1480; x++) {
     direita();
   }
    for(int x = 0; x < 957; x++) {
     frente();
    }
    for(int x = 0; x < 1480; x++) {
     esquerda();
    }
    for(int x = 0; x < 957; x++) {
     tras();
    }
     digitalWrite(imprimindo, LOW);
   }

   void quadrado(void){
    delay(500);
    for(int x = 0; x < 35; x++) {
     Levanta();
    }
    for(int x = 0; x < 500; x++) {
     direita();
    }
    for(int x = 0; x < 239; x++) {
     frente();
    }

    for(int x = 0; x < 35; x++) {
     Escreve();
    }
    for(int x = 0; x < 480; x++) {
     direita();
    }
    for(int x = 0; x < 480; x++) {
     frente();
    }
    for(int x = 0; x < 480; x++) {
     esquerda();
    }
    for(int x = 0; x < 480; x++) {
     tras();
    }
    for(int x = 0; x < 35; x++) {
     Levanta();
    }
 for(int x = 0; x < 239; x++) {
tras();
    }
    for(int x = 0; x < 500; x++) {
```

```
    esquerda();
   }
   for(int x = 0; x < 35; x++) {
    Escreve();
   }
    digitalWrite(imprimindo, LOW);
  }
   void  triangulo(void){
   delay(500);
   for(int x = 0; x < 35; x++) {
    Levanta();
   }
   for(int x = 0; x < 500; x++) {
    direita();
   }
   for(int x = 0; x < 239; x++) {
    frente();
   }

   for(int x = 0; x < 35; x++) {
    Escreve();
   }
   for(int x = 0; x < 480; x++) {
    direita();
   }
   for(int x = 0; x < 235; x++) {
    frente();
    esquerda();
   }
   for(int x = 0; x < 235; x++) {
    tras();
    esquerda();
   }
   for(int x = 0; x < 35; x++) {
    Levanta();
   }
   for(int x = 0; x < 239; x++) {
   tras();
   }
   for(int x = 0; x < 500; x++) {
   esquerda();
   }
for(int x = 0; x < 35; x++) {
    Escreve();
   }
    digitalWrite(imprimindo, LOW);
  }
void setup () {
// Define os dois pinos como Saídas
pinMode (E_stepPin, OUTPUT);
pinMode (E_dirPin, OUTPUT);
pinMode (D_stepPin, OUTPUT);
pinMode (D_dirPin, OUTPUT);
pinMode (C_stepPin, OUTPUT);
pinMode (C_dirPin, OUTPUT);
pinMode (Sel1, INPUT_PULLUP);
pinMode (Sel2, INPUT_PULLUP);
pinMode (imprimindo, OUTPUT);
}

void loop() {
 int val=0;
```

```
 digitalWrite(imprimindo, LOW);
 estado1=digitalRead(Sel1);
 estado2=digitalRead(Sel2);

 if(estado1==LOW && estado2==LOW ){
   val=3;
 }else if(estado1==HIGH && estado2==LOW){
   val=1;
 }else if(estado1==LOW && estado2==HIGH){
   val=2;
 }else{
   val=0;
 }

 switch (val) {
 case 0:
   digitalWrite(imprimindo, LOW);
   break;
 case 1:
    digitalWrite(imprimindo, HIGH);
    areatotal();
   break;
 case 2:
   digitalWrite(imprimindo, HIGH);
   quadrado();
   break;
 case 3:
   digitalWrite(imprimindo, HIGH);
   triangulo();
   break;
}

}
```