ESCOLA ESTADUAL DE ENSINO PROFISSIONAL ESTRELA
CURSO TÉCNICO DE INFORMÁTICA EM REDES DE COMPUTADORES

Frederico Schardong

Controle Numérico Computadorizado: Construindo o Hardware e Software de uma Impressora

Estrela
2009

Frederico Schardong

Controle Numérico Computadorizado: Construindo o Hardware e Software de uma Impressora

Projeto de conclusão apresentado ao Curso Técnico de Informática em Redes de Computadores, da EEEPE, para a obtenção do título de Técnico de Informática em Redes de Computadores.
Orientador: Prof. Clei Delco Acosta, Me.

Estrela
2009

AGRADECIMENTOS

Existem situações na vida em que é fundamental poder contar com o apoio e a ajuda de algumas pessoas.

Para a realização deste trabalho de conclusão, pude contar com várias. E a essas pessoas prestarei, através de poucas palavras, os mais sinceros agradecimentos:

Ao meu pai, Valnir Schardong que, através do seu vasto conhecimento em eletrônica e automação industrial construiu o hardware de minha implementação e me apoiou psicológicamente e financeiramente. Sem ele este trabalho não sairia do papel.

Ao professor Clei Acosta, orientador deste trabalho, pelo seu conhecimento e grande vontade de me ajudar.

Ao amigo Luciano Metz, que emprestou um *joystick*.

Ao time de desenvolvimento do toolkit GTK+ e da biblioteca ALLEGRO, por terem criado essas maravilhosas ferramentas de desenvolvimento.

RESUMO

Neste trabalho foi construído o *hardware* e *software* de um Controle Numérico Computadorizado (CNC) que desenha em papel com uma caneta esferográfica. O *hardware* foi fabricado com partes de outras impressoras e sucatas eletrônicas. O *software* consiste em dois programas, sendo o primeiro uma aplicação de computador, escrito em C e interface gráfica em GTK+. Ele permite ao operador múltiplos modos de interação com o CNC. Ele pode controlar os dois eixos com um joystick, criar um desenho em um editor gráfico, e também selecionar imagens do seu computador para impressão. No último caso, o aplicativo permite ao operador visualizar como a imagem, em duas cores (preto e branco), ficará quando impressa. O segundo programa controla o microcontrolador do CNC, que operacionaliza os comandos de alto nível recebidos do programa de computador. Toda a comunicação com o programa de computador acontece através de um protocolo de rede criado como parte do trabalho. Na impressão de imagens, devido ao limite de memória, a imagem é enviada em partes durante sua impressão. O microcontrolador controla os motores de passo (eixos X e Y) e o motor de corrente contínua (eixo Z), conforme interpreta os comandos ou imagem recebida.

Palavras-chave: Controle Numérico Computadorizado. Automação. Mecatrônica. Impressora. Microcontrolador. PIC. C. GTK.

ABSTRACT

In this work, the hardware and software of a Computer Numerical Control (CNC) that draws on paper with a ballpoint pen were built. The hardware was made with parts from other printers and electronic scrap. The software consists of two programs, the first being a computer application, written in C and with a graphical interface in GTK+. It allows the operator to interact with the CNC in multiple ways. He can control the two axes with a joystick, create a drawing in a graphical editor, and also select images from his computer. In the last case, a viewer allows the operator to see how the image, in two colors (black and white), will look when printed. The second program controls the CNC's microcontroller, which operates the high-level commands received from the computer program. All communication with the computer program occurs through a network protocol created as part of the work. When printing images, due to memory limits, the image is sent in parts during printing. The microcontroller controls the stepper motors (X and Y axes) and the direct current motor (Z axis), according to how it interprets the commands or image received.

Keywords: Computer Numerical Control. Automation. Mechatronics. Printer. Microcontroller. PIC. C. GTK.

NOTA DE REVISÃO

O presente manuscrito foi revisado e amplamente reescrito a partir do capítulo 4 em relação ao manuscrito original. A revisão aconteceu no final de 2024. O objetivo foi incluir descrições mais precisas, imagens ilustrativas, referências e melhorar a formatação, que estava ausente na versão original. Essa revisão busca aprimorar a compreensão dos conceitos abordados e melhor apresentar os resultados obtidos. Diferenças no estilo de escrita entre os capítulos é consequência da diferença de 15 anos entre o manuscrito original e este.

LISTA DE FIGURAS

LISTA DE SIGLAS

CI	Circuíto Integrado
CLP	Controladores Lógicos Programáveis
CN	Controle Numérico
CNC	Controle Numérico Computadorizado
DIP	Dual In-Line Package
EEPROM	Electrically Erasable Programmable Read-Only Memory
GIMP	GNU Image Manipulation Program
GNU GLP	GNU General Public Licenses
GTK+	GIMP Toolkit
PeD	Pesquisa e Desenvolvimento
PWM	Pulse-Width Modulation
RAD	Rapid Application Development
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory
USART	Universal Synchronous Asynchronous Receiver Transmitter

SUMÁRIO

1 INTRODUÇÃO

Desde a Revolução Industrial, com a invenção das primeiras máquinas, o homem vem notando como elas facilitam o trabalho. As primeiras máquinas for criadas para transformar fios de lã para tecelagem. Uma dessas máquinas era capaz de substituir o trabalho de múltiplas pessoas, e eram capazes de trabalhar sem parar (TEIXEIRA, 1997).

O surgimento dessas indústrias ocorreu ao redor de córregos e rios pois eles forneciam energia na forma de movimento para as máquinas funcionarem. Com o passar do tempo surgiram as máquinas a vapor, assim possibilitando que essas indústrias têxteis se instalassem em locais sem córregos de água (TEIXEIRA, 1997).

Alguns anos depois foi inventada uma técnica nova para a confecção de tapetes. Tábuas com saliências corriam sob pinos de metais de forma que quando os pinos caíssem alavancas eram puxadas e a máquina mudava a forma que um certo ponto era costurado. Nascia assim o código binário com a invenção da máquina de tear de Joseph Marie Jacquard (COSTA, 2008).

Cerca de 150 anos depois da primeira Revolução Industrial, Henry Ford criou, além da montadora Ford, o conceito Fordismo. A principal característica do conceito é a linha de montagem, onde uma esteira de montagem, onde uma esteira corre com a matéria-prima e os funcionários/máquinas fazem fazem o que deve ser feito para chegar ao produto final. Taylor, estudando o fordismo, acrescentou a característica que em cada ponto da linha de montagem devem haver pessoas especializadas em sua sua área de trabalho, com horários específicos para tudo e serem treinadas (FERREIRA, 1993).

Com o surgimento de máquinas cada vez melhores, a grande quantidade de trabalhadores nas linhas de montagens foi sendo substituída por máquinas que precisavam ser operadas por menos pessoas. Para as indústrias isso foi muito bom, pois elas não precisavam pagar tantos funcionários, só os que operavam as máquinas. Surgia aqui o conceito de máquinas Controle Numérico (CN). Com mais dinheiro em caixa, as empresas passaram a investir mais em pesquisa para criar novas tecnologias e, consequentemente, criar novas e melhores máquinas (TAVARES, 2009).

Com o avanço da tecnologia as máquinas deixaram de ser operadas por pessoas e passaram a funciona por conta própria. Na verdade, elas operam seguindo os passos de um *software* que diz o que, como e onde deve ser feito. Surge aqui o conceito Controle Numérico Computadorizado (CNC), onde a máquina opera sozinha, sem precisar ser comandada por alguém (TAVARES, 2009).

Essas máquinas também foram um grande avanço para a indústria, pois ela deixou de gastar com funcionários que operavam as antigas CN e passaram a investir mais dinheiro em Pesquisa e Desenvolvimento (PeD) para sucessivamente produzirem máquinas automatizadas.

2 OBJETIVOS GERAIS E ESPECÍFICOS

2.1 OBJETIVO GERAL

Este trabalho tem como objetivo geral criar o *hardware* e o *software* de um CNC que desenha em papél com uma caneta esferográfica.

2.2 OBJETIVOS ESPECÍFICOS

Este trabalho tem como objetivos específicos:

- Construir um CNC com partes de impressoras.
- Codificar um microcontrolador para controlar o CNC e se comunicar com o computador.
- Codificar uma aplicação para computador, com interface gráfica utilizando a biblioteca GTK+, que se comunique com o CNC.

3 AUTOMAÇÃO INDUSTRIAL

Neste capítulo é abordada a automação industrial, seus conceitos, e também a programação nesta área.

3.1 INTRODUÇÃO E CONCEITOS

Automação industrial é o conceito de aplicar técnicas para aumentar a produtividade, aumentar a qualidade do produto, diminuir a poluição, melhorar as condições de segurança do produto ou do seu processo de fabricação. Podemos dizer que a automação industrial fica um nível acima da mecanização. Mecanização ocorre quando funcionários são auxiliados por máquinas, diferentemente da automação, onde as máquinas os substituem (ROSáRIO, 2009).

Na automação industrial os CNCs são largamente utilizados, mas não são os únicos. Existem também os Controladores Lógicos Programáveis (CLP). Estes dispositivos são centrais de processamento industriais, e neles são controladas máquinas complexas através de várias entradas digitais e analógicas. Sensores instalados nas máquinas a serem controladas enviam a informação ao CLP e ele processa essa informação e toma sua respectiva ação (ROSáRIO, 2009).

A aplicação da automação industrial é visto por muitos apenas como robótica. Porém, como falado antes, automação industrial consiste em melhorar processos produtivos, em que nem sempre a robótica é a peça principal. Nas indústrias farmacêuticas, petroquímicas e químicas a utilização de robôs é de mínima importância comparado com uma CLP, que pode controlar sensores de temperatura, pressão, pH, concentração de substâncias, etc.

3.2 CONTROLE NUMÉRICO COMPUTADORIZADO

O CN pode ser definido como uma máquina, dispositivo ou até ferramenta que ajuda o trabalho de alguma forma. Seja reduzindo o esforço humano ou até fazendo o que o homem não consegue por suas limitações corporais, como atingir alta profundidade, ir ao espaço, voar e andar a altas velocidades (TAVARES, 2009). Já o CNC faz o que o CN faz, apenas sem a intervenção humana. Pode-se dar como exemplo de CNC: robôs, impressoras, digitalizadores e centros de usinagem. Centros de usinagens são máquinas que podem cortar, polir, furar em vários eixos. Outra categirua de CNC largamente utilizado são os CNCs que furam placas de circuito impresso, e montam e soldam circuitos nessas placas (TAVARES, 2009).

Os CNCs são de grande importância na automação industrial. São eles que produzem vários produtos dos quais necessitamos para viver. Por exemplo, as placas mães dos computadores, e a montagem dos carros. São os CNCs de centros de usinagem que produzem praticamente todas as peças dos carros, motos e caminhões. São CNCs que facilitam o trabalho humano, tornam os produtos mais baratos, e iguais.

Os CNCs possuem uma central de comando (microcontrolador ou microprocessador), sensores, atuadores e alguma forma de comunicação. A seguir discutimos alguns destes assuntos.

3.2.1 Microcontrolador e Microprocessador

A principal diferença entre um microcontrolador e um microprocessador são seus objetivos. O microprocessador visa processar informações, já o microcontrolador além de processar informações, controla outros dispositivos. Por exemplo, controla entradas digitais e analógicas e controla interrupções externas (FRANCISCO, 2008).

O microprocessador sempre precisa estar acompanhado de outras peças, tais como memória, *chipsets* e componentes para receber e enviar dados. O microcontrolador já vem com todos esses componentes dentro dele, memória Random Access Memory (RAM) e ROM, entradas e saídas analógicas e digitais, e interfaceamento para comunicação, tal como USB, I2C e também controladores de LCD. Os microcontroladores operam em frequências entre KHz e MHz. Já os microprocessadores podem alcançar até vários GHz (FRANCISCO, 2008).

3.2.2 Comunicação

Comunicação serial é o ato de enviar a informação sequencialmente por uma mesma linha de transmissão, diferentemente da comunicação paralela, onde a informação trafega por várias linhas de transmissão em simultâneo.

O protocolo RS-232 é um protocolo amplamente utilizado para padronizar a comunicação serial entre dois dispositivos que desejam trocar informações. No protocolo RS-232 os sinais variam entre -12 e +12 V, sendo que a lógica é invertida, ou seja, quando a voltagem -12 é recebida, é considerado como nível lógico alto, ou seja, o bit 1. Para notebooks a voltagem da porta serial pode variar entre -3,3 e + 3,3 V (HIRAKAWA, 2005).

Existem dois modos de transmissão, o modo assíncrono permite enviar e receber informações a qualquer momento, a informação é transmitida e quem está recebendo é encarregado de detectar isso. No modo síncrono existe um bit de paridade, um bit de início de mensagem e um ou dois bits de fim de mensagem (HIRAKAWA, 2005).

Para se comunicar utilizando o modo síncrono, a informação precisa ser encapsulada. O primeiro bit a ser emitido é o bit de partida, ele serve para alertar a quem vai receber a informação que a transmissão irá começar. Após o bit de partida, vem a informação, ela pode variar de 5 a 8 bits, logo após é enviado, ou não, um bit de paridade (controle de consistência da informação), e por último um ou dois bits que informam o fim do pacote.

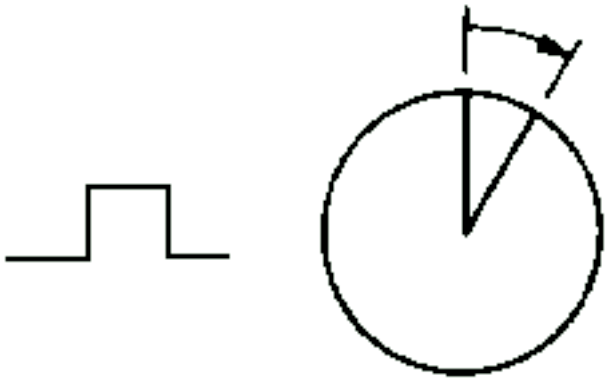
O modo assíncrono possui uma taxa de transferência mais alta que o modo síncrono, porém, é mais ineficiente, pois não há bit de paridade, bits de início, ou fim, com isso a informação é facilmente corrompida. Entretanto, no modo síncrono há uma segurança maior quanto a consistência das informações transmitidas.

3.2.3 Motor de Passo

Uma categoria de atuador amplamente utilizado em CNCs é o motor de passso. O motor de passo é um dispositivo que transforma a informação digital enviada a ele em movimento. A principal diferença entre um motor de corrente contínua e um motor de passo é que ele precisa ser alimentado constantemente com a informação do que ele deve fazer, ao contrário dos motores de corrente contínua que giram somente aplicando uma tensão (MEZENGA, 2000).

A cada pulso recebido, o motor de passo faz um incremento rotativo (passo), conforme mostrado na Figura 1. Cada passo é só uma parte de uma rotação completa. Vários pulsos devem ser enviados para atingir uma desejada rotação do eixo do motor. A precisão de um motor de passo vem da quantidade de paços necessários para completar uma volta completa de eixo. Por exemplo, um motor que necessita de 36 passos para completar uma volta no eixo. Ou seja, 360º/36, resultará em 10º de rotação por passo recebido. Quanto menos graus por passo, maior será a precisão do motor (MEZENGA, 2000).

Figura 1 – Um pulso se transforma em incremento rotativo.



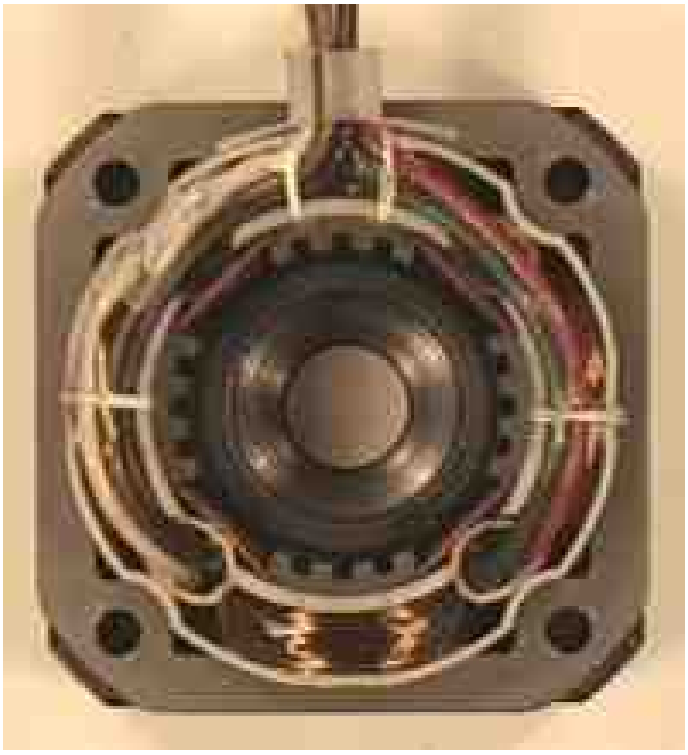
Fonte: (MEZENGA, 2000)

Existem várias categorias de motores de passo, o mais comum é o motor de passo unipolar de quatro fases. Este motor de passo é composto por um rotor (eixo) que está acoplado a um ímã permanente (GEDEA, 2005). Em volta dele existem quatro bobinas, como pode ser visto na Figura 2.

Conforme a corrente que passa pelas bobinas, é formado um campo magnético. Este campo magnético alinha o rotor, movimentando o eixo. Combinando a lógica de acionamento e a frequência dos pulsos, é possível obter a rotação e a direção desejada. A Figura 3 ilustrativa que o eixo do motor só possui um pólo magnetizado, e a cada pulso enviado ao motor, ele gira em 90º.

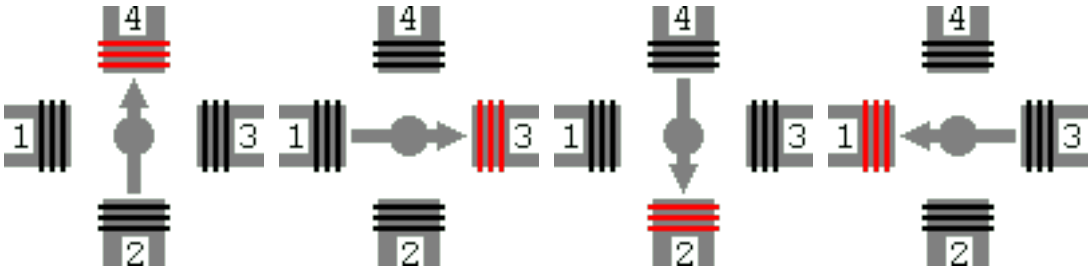
Outra opção de acionamento do motor de passo é o micro passo. Este método consiste em acionar duas bobinas em simultâneo, assim o rotor se alinha no meio das duas bobinas ligadas. Isso dobra a precisão do motor (GEDEA, 2005).

Figura 2 – Interior de um motor unipolar de quatro fases de ímã permanente.



Fonte: (GEDEA, 2005)

Figura 3 – Bobinas eletrizadas alinhando o rotor.



(a) Centralized. (b) Centralized. (c) Centralized. (d) Centralized.

Fonte: (GEDEA, 2005).

3.3 PROGRAMAÇÃO NA AUTOMAÇÃO INDUSTRIAL

3.3.1 Linguagens de Programação

Tradicionalmente microcontroladores e microprocessadores são programados nas linguagens: Basic, C, Visual Basic ou Assembly, por serem linguagens de baixo nível, por isso trabalham diretamente com o *hardware*, proporcionando melhor desempenho (ROSáRIO, 2009).

3.3.2 Interface Gráfica

Em 1995 Spencer Kimball e Peter Mattis começaram a desenvolver o GNU Image Manipulation Program (GIMP), inicialmente usavam o *toolkit* Motif para construir sua interface gráfica, criado em 1980 para o UNIX. Porém, esse *toolkit* era pago, e decidiram criar um *toolkit* próprio, inicialmente ele fazia parte do código-fonte do GIMP, mas decidiram que seria melhor torna-lo um *toolkit* separado do projeto GIMP. Nasceu ali o GIMP Toolkit (GTK+)(GTK+, 2007).

O GTK+ é um *toolkit* baseado em *widgets*. Um *widget* é um componente de baixo nível, uma estrutura que contem uma janela, tamanho, atributos que dizem se ele é visível, editável, sensível, etc. Um *widget* pode ser uma janela, pode ser um botão, uma entrada de texto, um menu, e outras coisas. Os *widgets* também disparam eventos. Os eventos são disparados por elementos e ações do usuário. Estes eventos são conectados a funções e elas tratam os eventos (GTK+, 2007).

Para programar em GTK+ é preciso conhecer os *widgets*, seus atributos, e saber trabalhar com seus eventos. Eventos são ações disparadas pelo sistema quando um determinado *widget* está sobre uma condição pré-determinada. Esta condição pode ser um clique, ser minimizado, ser movido, ser deletado, ser criado, aparecer, sumir e outros.

No GTK+ é possível criar e organizar os *widgets* apenas com comandos, porém isso é trabalhoso. Para otimizar o processo de construção de widgets, foi criado o Glade, ele é um Rapid Application Development (RAD), faz parte do projeto GNOME e está sobre a licença GNU General Public Licenses (GNU GLP). O Glade gera arquivos XML que através da biblioteca `libglade` é possível atribuir os *widgets* criados nele aos *widgets* do programa. No Glade é possível atribuir aos *widgets* as funções desejadas que devem ser executadas quando certos eventos forem disparados.

3.3.3 Sistema Embarcado

Um sistema embarcado é composto por *hardware* e *software* que possuem uma finalidade específica. Ao contrário de um computador, que pode sofrer várias alterações, troca de sistema operacional, entre outros, isso não acontece em um sistema embarcado. Sistemas embarcados, na sua maioria, não aceitam ser reprogramados e se espera que durem vários anos sem apresentar nenhuma falha, pois a memória Read-Only Memory (ROM) de seu microprocessador ou microcontrolador nunca é alterada (ROSáRIO, 2009).

Um sistema embarcado normalmente não possui uma interface de usuário muito complexa, pois isso representa custos adicionais ao projeto. A melhor maneira de fornecer uma interface completa é utilizar acesso remoto, isso pode acontecer com programas específicos como o utilizado neste trabalho, ou gerando páginas html, como nos modens e switchs.

4 CONSTRUÇÃO DE UM CNC PARA IMPRESSÃO

Neste capítulo são apresentados o *hardware* e o *software* construídos para atingir os objetivos do trabalho.

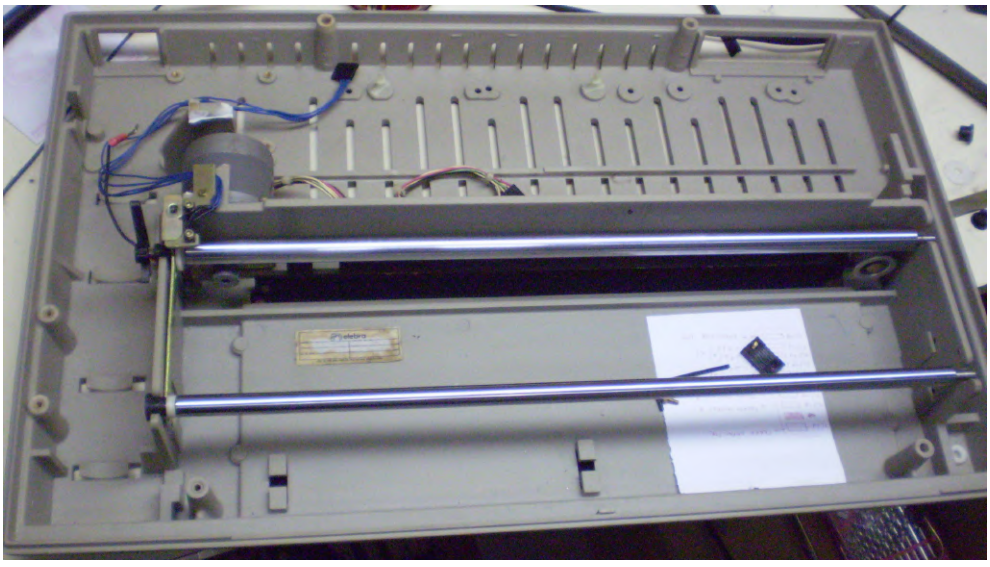
4.1 HARDWARE

Foi construído um CNC de três eixos (X, Y e Z), capaz de desenhar em duas dimensões (eixos X e Y) através do acionamento de uma caneta esferográfica (eixo Z).

O *hardware* do CNC construído é, na sua maioria, reaproveitado de outros dispositivos. Os principais materiais usados na construção do CNC vêm de uma impressora matricial da marca elebra (eixo X do CNC), e de uma impressora jato de tinta, da marca HP (eixo Y do CNC). Os motores utilizados no projeto são de passo e destas impressoras. O mecanismo utilizado para subir ou descer uma caneta esferográfica é uma bandeja de abertura de um leitor de CD (eixo Z do CNC).

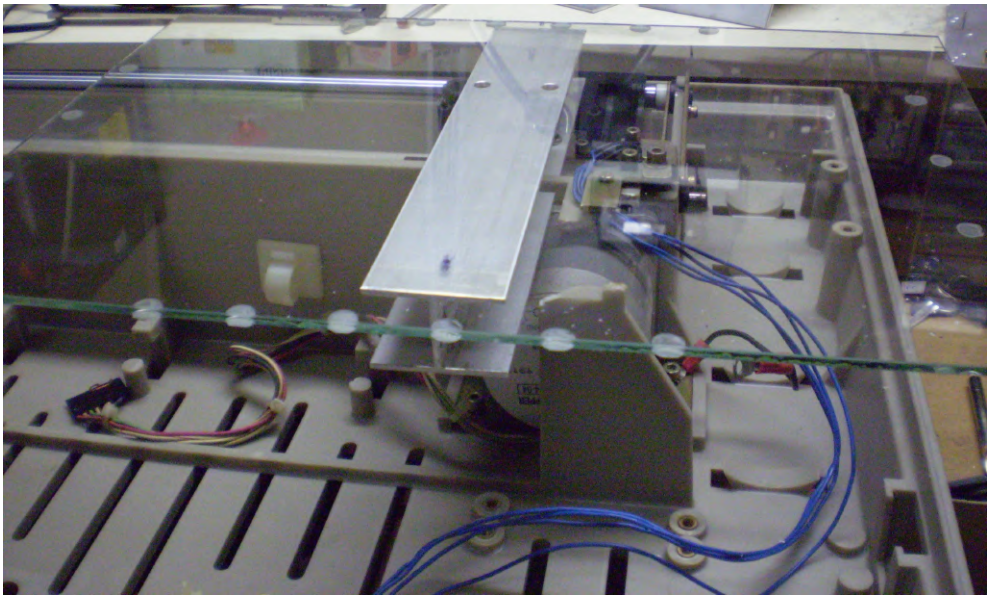
A Figura 4 mostra a parte inferior do CNC. A base é feita de uma liga de plástico duro, de forma que o carro de impressão orignal pode ser alterado para suportar uma barra de alumínio e um vidro de dimensões com poucos centímetros a mais que uma folha A4. A Figura 5 mostra o vidro preso a barra de alumínio, que está presa no carro de impressão do eixo X.

Figura 4 – Parte inferior da impressora matricial utilizada como base do CNC e eixo X.



Fonte: O autor.

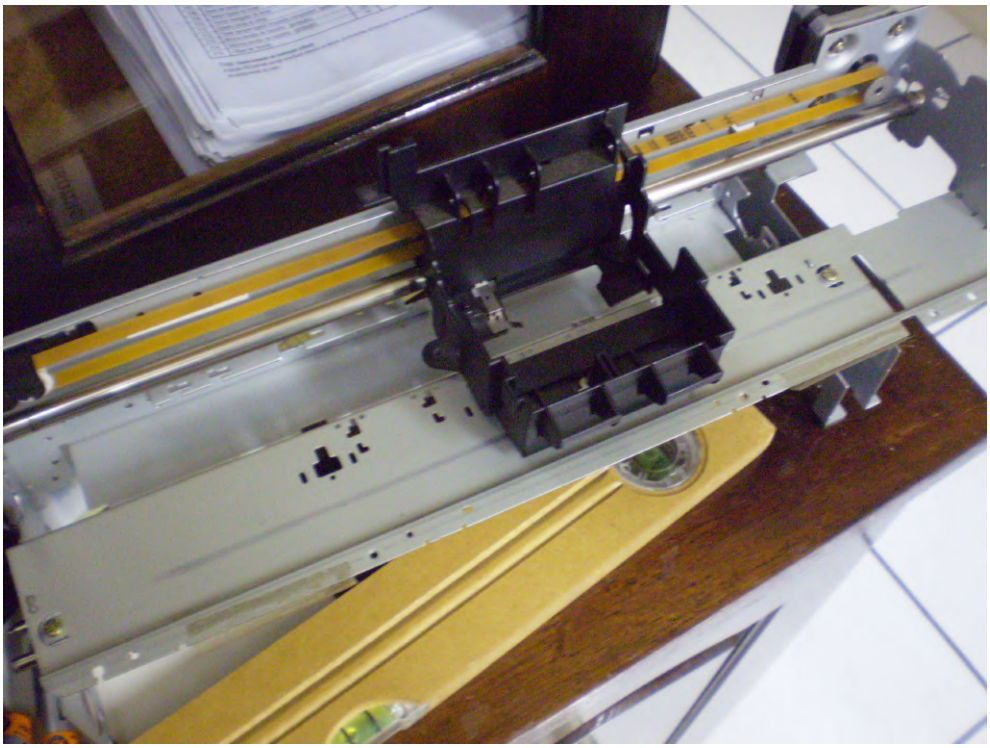
Figura 5 – Vidro maior que folha A4, preso a uma barra de alumúnio presa ao carro de impressão do eixo X.



Fonte: O autor.

A parte superior do CNC, mostrada na Figura 6 veio de uma impressora jato de tinta HP. Sua estrutura é feita de alumínio. Do carro de impressão foi removida a estrutura original que comportava o cartucho de tinta. Neste carro de impressão...

Figura 6 – Parte da impressora jato de tinta utilizada como eixo Y do CNC.

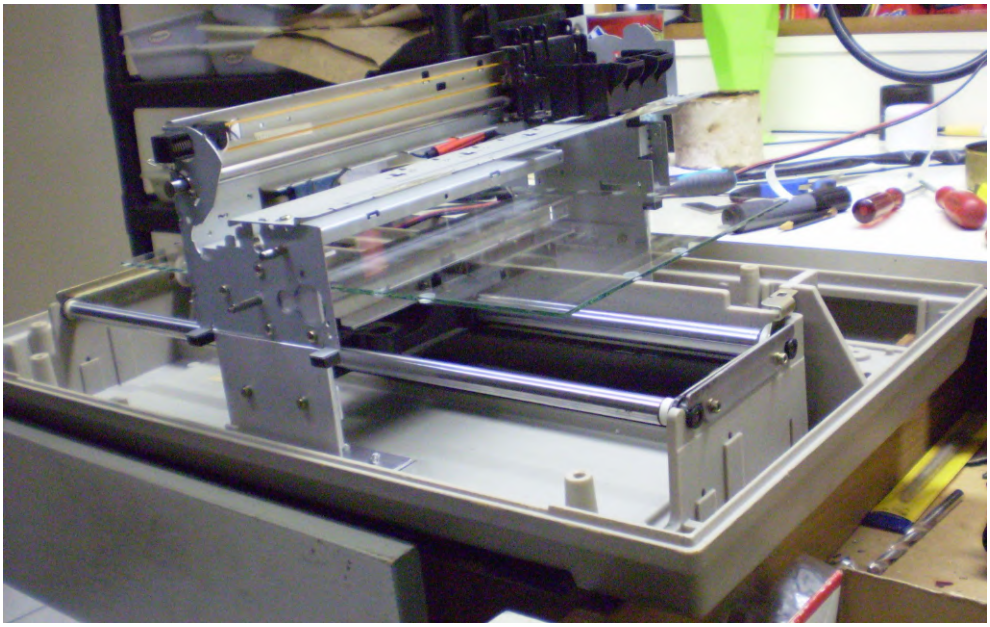


Fonte: O autor.

A parte superior do CNC (eixo Y) é presa à parte inferior (eixo X) através de dois suportes de alumínio, um em cada lado. A Figura 7 mostra a parte superior presa à parte

inferior através do suporte de alumínio de um dos lados.

Figura 7 – Parte superior (eixo Y) presa à parte inferior (eixo X).



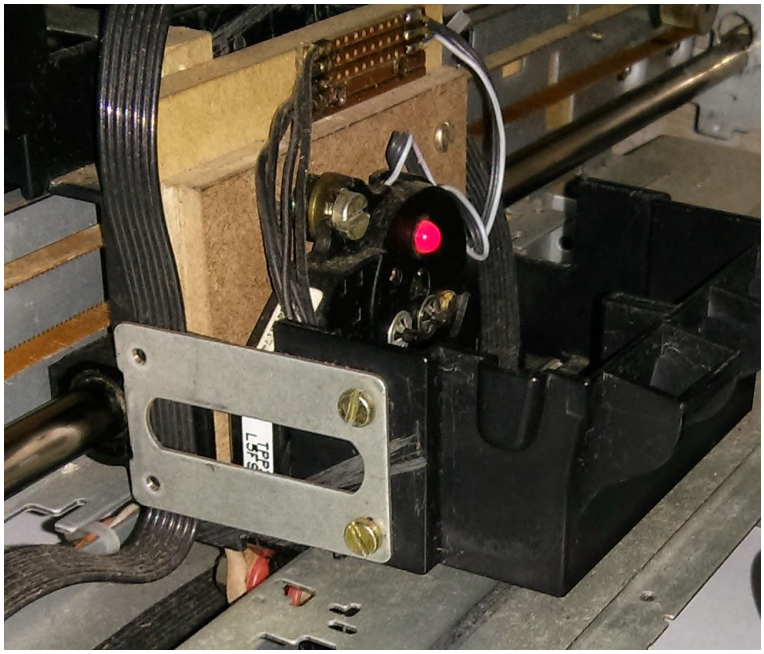
Fonte: O autor.

O eixo Z do CNC foi criado utilizando a bandeja de abrir e fechar de um leitor de CD. Nela foi acoplada uma caneta esferográfica. A Figura 8 mostra este mecanismo em ação, enquanto a Figura 9 mostra este mecanismo acoplado ao carro do eixo Y.

Figura 8 – Eixo Z em atuação.

Fonte: O autor.

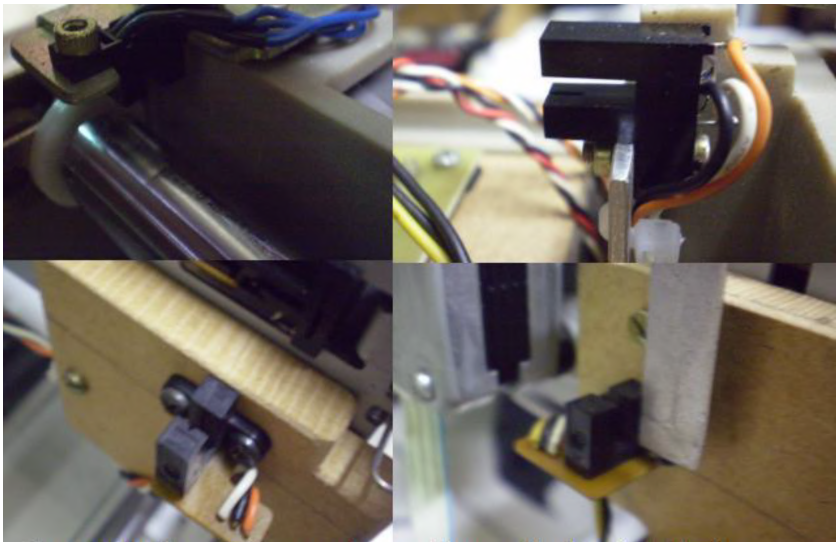
Figura 9 – Montagem do eixo Z no eixo Y.



Fonte: O autor.

Todos os eixos possuem sensores de final de curso. Entretanto, para o eixo Z os sensores atuam diretamente no motor de corrente contínua, que controla o subir e descer da caneta esferográfica. Já os eixos X e Y possuem dois sensores de final de curso cada, conforme mostrado na Figura 10. São sensores de presença foto-acopladores na posição limite de cada carro de impressão. O sensor de presença do eixo X é original da impressora matricial, já o sensor do eixo Y não existia na impressora original. Os sensores são construídos com um emissor de luz e logo a sua frente um receptor de luz. Quando um objeto interrompe a luz, a voltagem de saída do sensor é alterada.

Figura 10 – Sensores foto-acopladores de final de curso dos eixos X e Y.

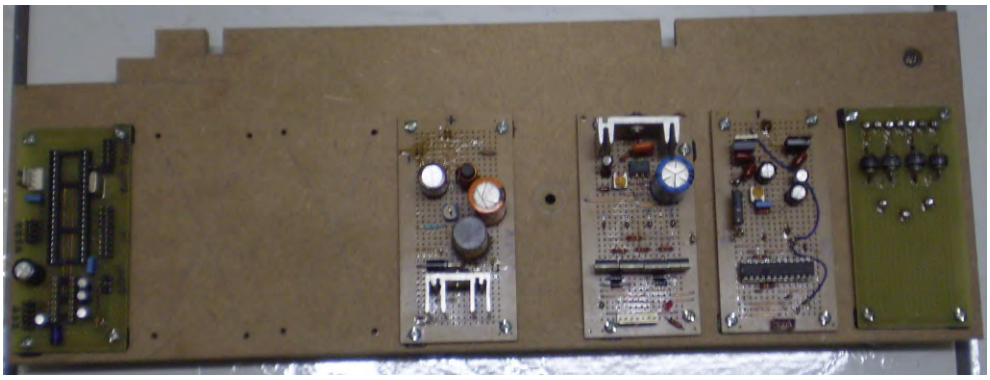


Fonte: O autor.

Os sensores são utilizados para o microcontrolador do CNC saber quando cada carro de impressão chega no limite do seu curso. Se os sensores não fossem usados, a correia que liga o carro de impressão ao motor seria rompida, pois, quando ele chegar no fim do curso o motor continuaria girando, até arrebentá-la. Outra razão pela qual os sensores nos eixos X e Y são necessários é, a partir do seu acionamento, poder calcular a posição exata do carro de impressão de cada eixo. Com isso, é possível posicioná-los para desenhar a imagem recebida do computador.

Para realizar o controle dos sensores, motores dos eixos e comunicação com o computador, foram criados módulos em placas virgens. A Figura 11 mostra a organização de alguns destes módulos, com o CNC ainda em construção.

Figura 11 – Módulos de *hardware* em construção.



Fonte: O autor.

Uma das placas administra o sinal recebido dos sensores, estabilizando e mandando para o microcontrolador que controla os motores. Esta etapa de estabilização é necessária, pois se o feixe de luz estiver parcialmente bloqueado, uma voltagem menor que a máxima e maior que a mínima aparecerá na saída do sensor. Como o microcontrolador só aceita estados lógicos de 5 e 0 V, um circuito com 4 portas lógicas AND foi desenvolvido, de modo que um pino está sempre ligado a 5 V e o outro à saída do sensor, a porta AND só emitirá o nível lógico alto quando a voltagem recebida do sensor for 0 V, e nível lógico baixo quando receber 5 V.

Quando o sensor está bloqueado, é emitido ao microcontrolador o bit 0 e quando está desbloqueado é emitido 1, para fins de facilitar a programação do microcontrolador, foi usado um conversor lógico na saída da porta AND, de forma que quando algum sensor for bloqueado, o microcontrolador receberá o bit 1 e quando nada estiver bloqueando-o ele receberá o bit 0.

4.1.1 Motores

Como os motores dos eixos X e Y são reaproveitados de impressoras diferentes, deu-se a necessidade de criar módulos específicos para cada um, pois eles não possuem as mesmas características elétricas.

O motor de passo do eixo X do CNC é um motor quadrifásico, que trabalha com 6 V e 1 A, mas segundo os dados do seu fabricante sua tensão máxima de trabalho é 1,2 A. No CNC manteu-se as características originais de acionamento deste motor, pois apresentou bons resultados nos testes de velocidade, força e ruído.

O motor do eixo Y, segundo os dados do fabricante e de acordo com sua utilização original, trabalha com até 42V e 0,5A, tendo como pico de corrente 0,75 A. Após submetido a vários testes de velocidade, força e ruído, foi estabelecido como voltagem e tensão ideais para o motor de passo da parte superior, respectivamente 12V e 0,5A.

A diferença de tensão e corrente entre os motores acontece devido ao motor do eixo X originalmente carregava muito mais peso que o motor do eixo Y, pois as impressoras modernas são muito mais leves, com cartuchos de tintas pequenos e leves. Quanto mais peso o motor tem que puxar ou empurrar, mais amperes serão consumidos. O carro de impressão da impressora matricial levava consigo a esfera de metal que continha as letras e números que podiam ser impressos, e o carro de impressão da impressora jato de tinta, levava apenas um leve cartucho de tinta e o cabeçote de impressão (dispositivo que larga a tinta no papel).

Como o microcontrolador do CNC tem saída digital de 5V e no máximo 0,1 A, foi usado um Circuíto Integrado (CI) para controlar o motor do eixo X. Esse CI veio de outra impressora, ele foi desenvolvido especialmente para controlar motores de passo bifásicos. Fornecemos a ele a voltagem de acionamento do motor, a amperagem máxima que pode ser consumida, e uma sequência lógica de acionamento própria do CI (ele transforma para a sequência real), se a amperagem máxima for atingida, ele desliga as bobinas do motor por segurança.

Para o motor inferior, não foi encontrado nenhum CI para controlá-lo, por isso, foi construído uma base de transístores, de forma que quando o microcontrolador enviar o sinal lógico, os transístores enviarão para o motor os 6 V. Foi implementado também, um controle de amperagem para este motor, mas após vários testes demostrou-se inútil a presença do mesmo.

O controle de amperagem só é necessário quando o motor girar muito devagar, pois as bobinas ficam ligadas muito tempo e vão cada vez consumindo mais e mais energia. Esse caso se aplica a este trabalho, mas o microcontrolador foi programado de tal forma que mandará o sinal para o motor inferior girar, e após alguns milisegundos ele é desligado. Isso não causa problemas, pois os ímãs do motor irão segurá-lo no lugar, impedindo qualquer deslocamento do carro de impressão, e consequentemente imprecisão do CNC.

4.1.2 Microcontrolador

O microcontrolador `16F877A`, mostrado na Figura 12 fabricado pela Microchip Technology, foi utilizado para controlar o CNC. Ele controla os motores, recebe informações dos sensores e se comunica com o computador. Este microcontrolador possui uma arquitetura de 8 *bits*. Sua memória *flash* (área destinada a armazenar o programa) tem capacidade para armazenar até 8 KB, memória RAM de 368 *bytes* e memória Electrically Erasable Programmable Read-Only Memory (EEPROM) de 256 *bytes*.

Figura 12 – Microcontrolador 16F877A.

Fonte: (Custom Computer Services, Inc., 2007).

O 16F877A, assim como todos os outros da família PIC, são programados originalmente em assembly, mas existem compiladores que transformam programas escritos na linaguagem de programação C para a linguagem de máquina. Existem também compiladores para a linguagem Basic.

Segundo a Microchip, o PIC 16F877A pode operar com até 20 Mhz de *clock*, mas nesse trabalho foi usado um clock de 4 MHz, pois não havia necessidade de mais. Existem somente 35 instruções Reduced Instruction Set Computer (RISC) para controlá-lo.

Como a maioria dos PICs, o 16F877A pode operar de 2 V a 5 V. Seu encapsulamento é do tipo Dual In-Line Package (DIP) é uma categoria de encapsulamento de CI. Suas principais características são o invólucro plástico ou metálico e duas fileiras de pinos em lados opostos do CI de 40 pinos. O 16F877A possui 33 pinos que podem ser configurados como entrada ou saída digital, possui 8 entradas analógicas-digitais (portas que aceitam várias voltagens e são internamente convertidas a valores decimais), 2 saídas com suporte a Pulse-Width Modulation (PWM), modulação por largura de pulso, usado para gerar voltagens entre 0 V e 5 V. Ele também tem um Universal Synchronous Asynchronous Receiver Transmitter (USART), protocolo de comunicação serial RS-232.

A memória EEPROM do microcontrolador usado, pode ser lida, apagada e gravada 1 milhão de vezes. A informação gravada dura até 40 anos sem perder sua integridade.

4.1.3 Comunicação entre o Microcontrolador e o Computador

O microcontrolador 16F877A precisa se comunicar com o computador, pois é do computador a imagem a ser impressa ou instruções diretas para onde mover os motores, no caso do uso de *joystick*. A forma adotada de implementar essa comunicação entre o microcontrolador e computador foi a comunicação serial, pois ela compromete somente dois pinos do microcontrolador, um para transmitir e outro para receber.

O microcontrolador usado, assim como todos os outros de sua família, vem com suporte ao protocolo RS-232, mas no formato TTL. O formato TTL define apenas que o bit 0 é 0

volts e o bit 1 é 5 Volts. Como o computador trabalha com -12V e 12V em sua porta RS-232, o CI MAX-232 da empresa Texas Componentes foi utilizado. Ele faz a conversão do modo TTL para TIA/EIA-232-F, ou seja, os -12V e +12V do protocolo RS-232 definidos pelo TIA/EIA. Os detalhes sobre os protocolos de comunicação criados em cima do RS-232 serão explicados na próxima seção.

4.2 SOFTWARE

4.2.1 Aplicativo de Computador

O aplicativo para computador, também chamado de programa servidor, aplicação *desktop* ou apenas aplicativo ou sistema, foi desenvolvido na linguagem de programação C. Ele usa, principalmente, as bibliotecas GTK+ e ALLEGRO, para fornecer uma interface gráfica amigável, manipular imagens e interagir com *joystick*. O aplicativo não é multiplataforma, pois faz uso de funções via interpretador de comandos bash únicas de sistemas Linux, bem como usa endereçamento da porta serial e *joystick* específico de sistemas Linux. Para compilar o programa, foi usado o compilador GCC na versão 4.3.3. A versão do GTK+ foi 2.16.1, do ALLEGRO foi a versão 4.2.2. A Figura 13 mostra a janela principal do sistema, onde o usuário pode escolher uma das três opções de interação.

Figura 13 – Janela principal do sistema.

Fonte: O autor.

As três principais funcionalidades do sistema são: (i) controlar os movimentos do CNC via *joystick*; (ii) desenhar em um editor gráfico para então imprimir no CNC; e (iii) carregar uma imagem do computador e então configurá-la e imprimir no CNC. Essas funcionalidades são descritas a seguir.

4.2.1.1 Controlar com Joystick

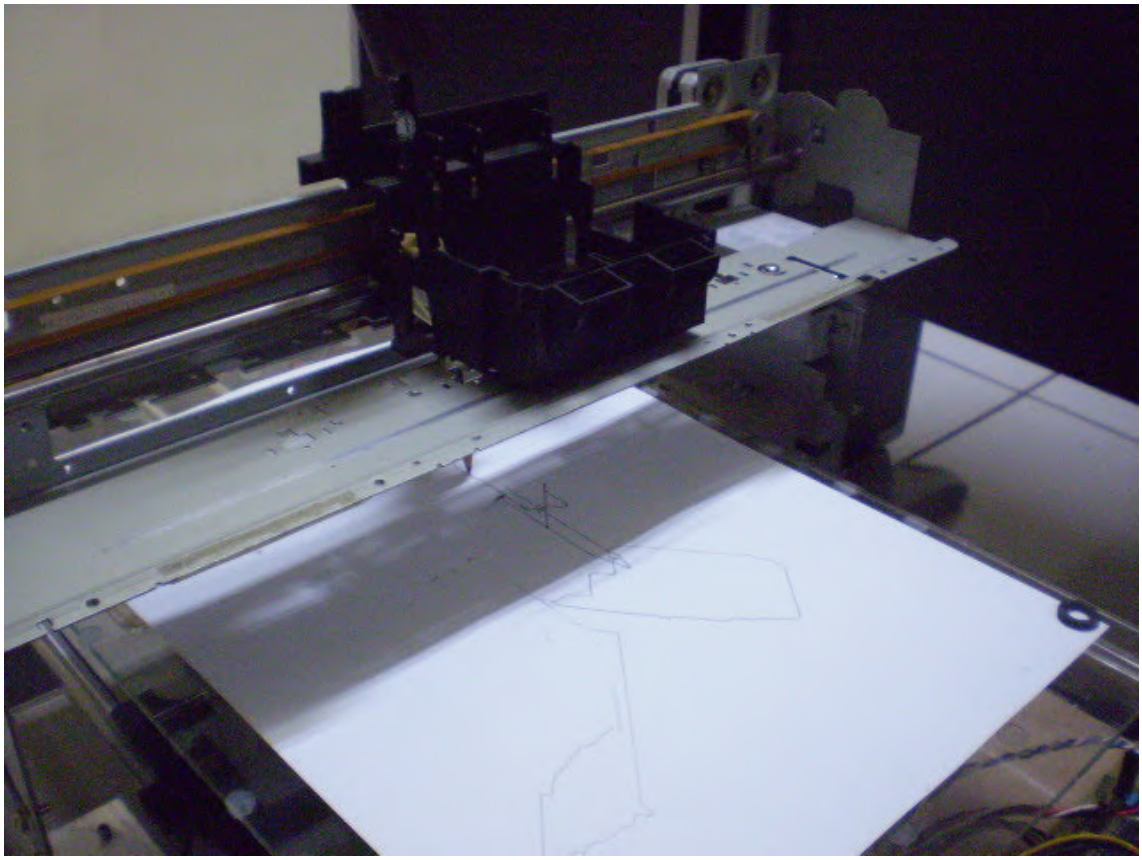
Para detectar o *joystick* e o movimento dos eixos e pressionar dos botões foi utilizada a biblioteca ALLEGRO. A biblioteca ALLEGRO foi criada inicialmente para a plataforma Atari ST, após a sua extinção obteve sua maturidade com o compilador DJGPP. Atualmente é mantida por voluntários (Allegro, 2009). Seu código é aberto e está licenciado sobre a giftware, essa licença diz que você pode fazer o que quiser com a ALLEGRO, porém deve contribuir para a comunidade ALLEGRO, seja reportando um erro, criando funcionalidades, liberando o código-

fonte para os outros aprenderem, fazer melhorias na biblioteca, ou apenas citando a ALLEGRO nos créditos.

A ALLEGRO captura o valor do eixo X e Y do *joystick* e o estado lógico dos botões, através destes valores, o programa *desktop* manda para o microcontrolador 3 valores: (i) o primeiro é a direção para onde o *joystick* se moveu (norte, nordeste, sul, sudoeste, sudeste, leste, etc...); (ii) o segundo é um valor que varia de 0 a 9, onde quanto maior é o número, mais longe do centro do *joystick* está o eixo X do manche; e (iii) um número de 0 a 9 com o mesmo objetivo, porém indicando para o eixo Y.

Com isso temos uma ferramenta de desenho a mão livre que da ao usuário a opção de mover os motores com diversas velocidades e em diversas direções. A Figura 14 mostra o CNC imprimindo um desenho sendo feito em tempo real com este modo do sistema.

Figura 14 – CNC imprimindo um desenho através de *joystick*.



Fonte: O autor.

4.2.1.2 Desenho a Mão Livre

Uma funcionalidade desenvolvida para o aplicativo *desktop* é a ferramenta de desenho. A Figura 15 mostra essa ferramenta. Ela consiste em uma janela para o usuário desenhar usando mouse, e uma janela de controles. Os controles permitem alterar a altura e largura do pincél de desenho. Também existe o pincel do tipo borracha, que apaga pontos do desenho. Um exemplo

de desenho é mostrado na Figura 15. Note que não é possível alterar a cor do desenho. Isso acontece, pois, o CNC só consegue imprimir uma única cor.

Figura 15 – Ferramenta de desenho criada no aplicativo.

Fonte: O autor.

A lógica foi construída pelo autor e usa o componente `GtkDrawingArea` ocupando toda a janela de desenho. Esta é uma estrutura do GTK+ que possibilita representar desenhos. Quando o usuário clica nela, um evento `button_press_event` é disparado. Ele é conectado a uma função do programa através da função `g_signal_connect()`. Esta função recebe como parâmetros em qual *widget* o evento aconteceu e a estrutura `GdkEventButton`, que nela consta as coordenadas X, Y de onde o clique aconteceu no *widget* que disparou o evento. O código que implementa a tratativa deste evento é apresentado no Algoritmo 4.1.

```
gboolean button_pressed(GtkWidget *a, GdkEventButton *event){
    gint x = event->x, y = event->y, x1, y1, i, tamH, tamV;
    gint x2, y2, xp, yp, xa, ya;
    gchar *modo = gtk_combo_box_get_active_text(GTK_COMBO_BOX(comboBox));

    if(g_strcmp0(modo, "Pincel") == 0){
        for(x1 = (0-posH); x1 < posH; x1++){
            for(y1 = (0-posV); y1 < posV; y1++){
                gdk_draw_point(
                    desenho->drawingArea->window,
                    desenho->drawingArea->style->fg_gc[
                        GTK_WIDGET_STATE(desenho->drawingArea)],
                    x+x1,
                    y+y1);
            }
        }

        g_ptr_array_add(desenho->parray, GINT_TO_POINTER(x));
        g_ptr_array_add(desenho->parray, GINT_TO_POINTER(y));
        g_ptr_array_add(desenho->parray, GINT_TO_POINTER(posH));
        g_ptr_array_add(desenho->parray, GINT_TO_POINTER(posV));
    }

    if(g_strcmp0(modo, "Borracha") == 0){
        for (x1 = 0; x1 < desenho->parray->len; x1 = x1 + 4){
            xp = GPOINTER_TO_INT(desenho->parray->pdata[x1]);
            yp = GPOINTER_TO_INT(desenho->parray->pdata[x1+1]);
            xa = GPOINTER_TO_INT(desenho->parray->pdata[x1+2]);
            ya = GPOINTER_TO_INT(desenho->parray->pdata[x1+3]);

            if (abs(xp-x) <= xa && abs(yp-y) <= ya){
                g_ptr_array_remove_index(desenho->parray, x1);
                g_ptr_array_remove_index(desenho->parray, x1);
                g_ptr_array_remove_index(desenho->parray, x1);
                g_ptr_array_remove_index(desenho->parray, x1);
            }
        }

        gtk_widget_hide(desenho->drawingArea);
        gtk_widget_show(desenho->drawingArea);
    }

    return TRUE;
}
```

Algoritmo 4.1 – Apaga ou desenha usando o retângulo com as dimensões escolhidas pelo usuário.

As variáveis locais x e y recebem as posições event->x e event->y de onde o mouse foi clicado dentro do *widget*. Como a janela de opções do desenho fornece um seletor de modo, os dois valores possíveis são ‘‘Pincel’’ ou ‘‘Borracha’’. A variável modo recebe o que o usuário selecionou e logo após isto a função g_strcmp0 compara se o conteúdo da variável modo é igual a *string* ‘‘Pincel’’, se for igual então a função retorna 0.

Logo após há dois laços de repetição cujo objetivo é desenhar um quadrilátero, a largura e a altura são definidas pelo usuário na janela de opções e estes valores são armazenados nas variáveis posH e posV. A função gdk_draw_point é responsável por desenhar um ponto no GtkDrawingArea. Os laços são feitos de forma que vários pontos sejam desenhados, formando a figura com as dimensões escolhidas pelo usuário.

Um *array* grava a informação de cada figura desenhada no GtkDrawingArea. Neste *array*, são gravados quatro valores: x, y, a largura e altura do quadrilátero desenhado.

Se a opção do usuário não for o pincel, e sim a borracha a primeira condição não será atendida, só a segunda. Nela, há um laço que percorre todo o *array* onde estão gravadas as posições em que há algum quadrilátero sendo feita uma análise para testar se um dos quadriláteros coincide com a posição x, y de onde o mouse foi pressionado. Se existir então o quadrilátero é removido do *array*. Após a remoção do quadrilátero, o *widget* de desenho é escondido e mostrado, para disparar outro evento. Este evento é conectado a uma função que desenha no GtkDrawingArea todos os quadriláteros registrados no *array*, isto precisa ser feito, pois, por padrão quando ele perde o foco do usuário tudo nele é apagado.

Após o usuário fazer um desenho e clicar no botão gravar, uma rotina de comunicação com o CNC é criada e após estabelecer comunicação, a rotina de gravação de imagem é chamada. Essa rotina é a mesma utilizada na impressão de imagens existentes, descrita a seguir.

4.2.1.3 Imprimir Imagem

Por fim, o último modo de interagir com o CNC é através da impressão de imagens existentes no computador de quem está usando o sistema.

Ao escolher uma imagem através do diálogo do sistema operacional, uma janela mostrando a imagem é apresentada, como mostrado na Figura 16. Existem três abas de visualização. A primeira e padrão mostra a imagem sem alterações. A segunda aba mostra a imagem em tons de cinza, como mostrado na Figura 17.

Figura 16 – Visualização de imagem do computador.

Fonte: O autor.

Figura 17 – Visualização de imagem em tons de cinza.

Fonte: O autor.

O Algoritmo 4.2 mostra como a imagem é convertida para tons de cinza, usando a biblioteca ALLEGRO para obter os tons de vermelho, verde e azul de cada *pixel*.

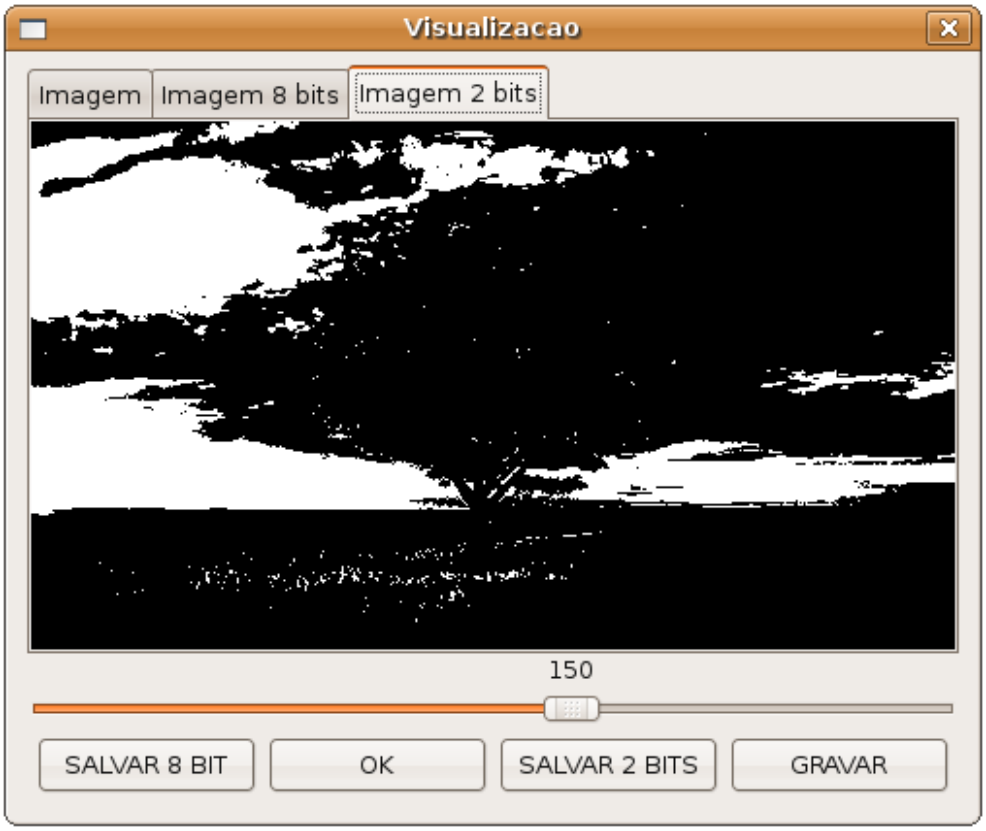
```
imagem = load_bitmap(".temp1.bmp", NULL);
imagemGray = create_bitmap(imagem->w, imagem->h);
gint matrizPixel[imagem->w][imagem->h];

for(x = 0; x < imagem->w; x++){
    for(y = 0; y < imagem->h; y++){
        matrizPixel[x][y] = getpixel(imagem, x, y);
        tempPixel = 0.299 * getr(matrizPixel[x][y]) +
                    0.587 * getg(matrizPixel[x][y]) +
                    0.114 * getb(matrizPixel[x][y]);
        matrizPixel[x][y] = makecol(tempPixel, tempPixel, tempPixel);
        putpixel(imagemGray, x, y, matrizPixel[x][y]);
    }
}

save_bitmap(".temp.bmp", imagemGray, NULL);
```

Algoritmo 4.2 – Converte uma imagem para tons de cinza.

O algoritmo inicia carregando a imagem escolhida pelo usuário, armazenada com o nome ‘‘`.temp1.bmp`’’ por um comando oriundo da biblioteca `gdk-pixbuf`, integrante da GTK+. Ela é usada somente para dar ao programa *desktop* o suporte às imagens nos formatos jpeg e png. A variável `imagem` recebe a imagem selecionada pelo usuário, logo após, a variável `imagemGray` recebe uma imagem em branco com a altura e largura da imagem original. Na próxima linha é declarada uma matriz bidimensional que armazena variáveis do tipo inteiro. O tipo `gint` é igual ao `int`, apenas foi definido como `gint` pelo GTK+ para seguir o seu padrão. Essa matriz possui as mesmas dimensões da imagem carregada pelo usuário, ela é posteriormente enviada ao microcontrolador para gravar a imagem no papel. Os dois laços de repetição percorrem toda a imagem original e através das funções `getr()`, `getb()`, `getg()`, é extraído de cada *pixel*, respectivamente, a quantidade do vermelho, azul e verde, as cores primárias (RGB). Após estes valores serem extraídos, é calculado 29% de vermelho, 58% de azul e 11% de verde somados para obter o píxel em tons de cinza (8 *bits*). Então a função `makecool()`, junta-os em um *pixel* e esse *pixel* é atribuído a posição atual do laço à variável `imagemGray` e à matriz. Por fim, obtem-se uma imagem em tons de cinza obtida a partir da transformação da imagem carregada pelo usuário, e ela é salva temporamente como ‘‘`.temp.bmp`’’.

A última aba permite ao usuário ver como a imagem em tons de cinza será gravada no papel. A Figura 18 mostra essa aba. Essa transformação é necessária dado que o CNC possui apenas uma caneta esferográfica. Ou seja, um *pixel* pode ser preenchido apenas por uma única cor. O usuário pode, através do controle deslizante horizontal na parte inferior da janela, escolher a partir de qual tom de cinza um *pixel* deve ser considerado como preenchido. A Figura 19 mostra como a imagem a ser impressa pelo CNC muda ao alterar o controle deslizante horizontal de 150 para 100. Os valores possíveis são os números inteiros de 0 a 255.

Figura 18 – Visualização de imagem em tons de cinza transformada em imagem com duas cores.



Fonte: O autor.

Figura 19 – Visualização de imagem em duas cores com parâmetro de transformação distinto.

Fonte: O autor.

Por fim, ao pressionar o botão `Gravar`, a rotina de comunicação com o CNC utilizando a portal serial e o protocolo RS-232 é iniciada. Caso o CNC esteja conectado, ele irá imediatamente imprimir a imagem conforme foi personalizada pelo usuário.

4.2.2 Sistema do Microcontrolador

O sistema criado em C para orquestrar o microcontrolador implementa funções de abstração para controlar os três eixos do CNC. Essas funções consideram os valores lidos dos sensores para garantir que os motores não ultrapassem os carros de impressão para além de seus limites físicos. Além destas abstrações, foram criados dois protocolos de comunicação em cima do RS-232. O primeiro é usado no modo de desenho livre com o *joystick*.

Esse protocolo transporta a direção em que os motores dos eixos X e Y devem girar, e a respectiva intensidade, sendo um inteiro de 0 a 9. O microcontrolador recebe estes valores e com isso coordena os motores para girarem para a direção orientada, numa certa velocidade. Os motores podem girar em velocidades diferentes, pois a velocidade de cada um independe do outro. Os valores da velocidade dos motores são transformados em uma frequência. Esta frequência é obtida somando à intensidade recebida o valor 1 e multiplicando-se o valor por 30. Isso possibilita uma variação de 30 Hertz até 300 Hertz em cada motor. Por fim, uma mensagem de finalização pode ser enviada pelo computador, indicando que o CNC deve retornar para o modo inicial, ou seja, aguardando a instrução de qual será o próximo modo de uso.

4.2.2.1 Rotina de Gravação de Imagem

A rotina de gravação de imagem inicia recebendo do computador a altura e a largura da imagem. Após isso, dois laços de repetição percorrem a imagem. Ao percorrer a imagem, o CNC recebe 50 pacotes de 8 *bits* cada, representando 400 *pixels* da imagem. Após receber 400 *pixels*, o CNC os imprime, então volta para o modo de transferência da imagem. A escolha de um *array* com 50 posicões de 8 *bits* não é arbitrária, é um limite imposto pelo microcontrolador.

Os dois laços de repetição são iguais no computador e no microcontrolador. Ou seja, não existe um protocolo de pergunta e resposta, onde o microcontrolador solicita os próximos *pixels* ao computador. Ambos estão sincronizados e devem finalizar toda a imagem. Não é possível pausar a impressão. Após receber os 400 *bits*, representando 400 *pixels* da imagem, o microcontrolador solicita que o computador espere até que o microcontrolador avise que está pronto para receber os próximos 400 *pixels*. Após solicitar a pausa, o CNC irá verificar cada bit e se algum bit for verdadeiro, a rotina de posicionar os motores e imprimir será iniciada.

Essa rotina recebe a posição X e Y em que um *pixel* deve ser colorido com a caneta esferográfica. O CNC então posiciona os motores dos eixos X e Y na exata posição do *pixel*, então aciona o eixo Z para baixar a caneta até encostar no papel. O motor do eixo Y sempre desloca um píxel para o lado para garantir que o papel foi manchado de tinta. Apenas baixar a

caneta, tocar o papel, e levantar a caneta não é o suficiente para que a tinta da caneta escreva no papel. Cada *pixel* da imagem é representada por um passo do motor de passo.

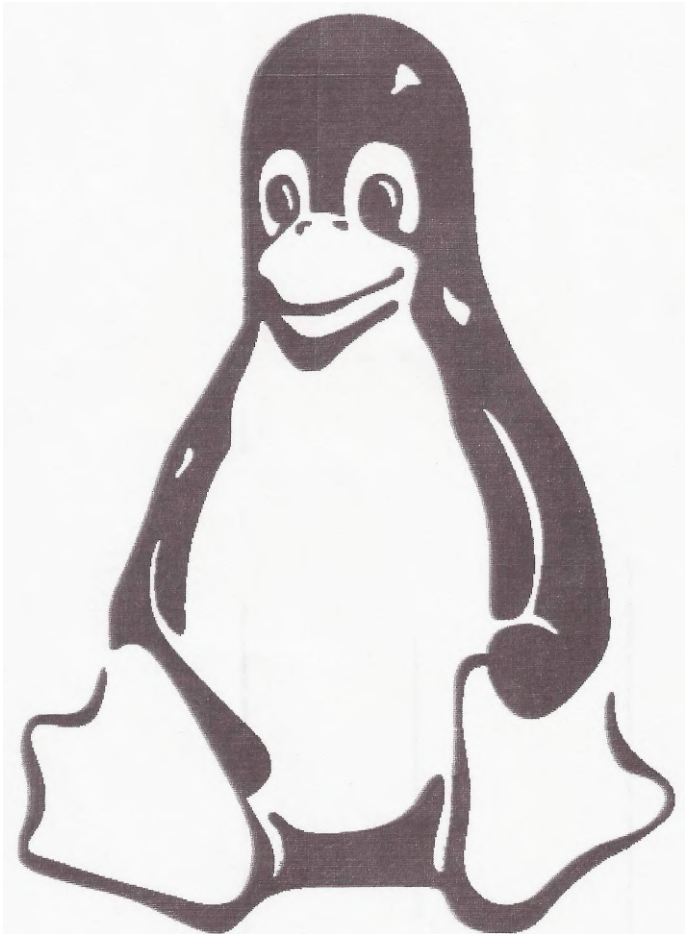
Como o movimento do eixo Y é o que, de fato, escreve no papel, uma otimização foi realizada. Em linhas pares o carro do eixo Y move-se da esquerda para a direita, enquanto em linhas ímpares o carro move-se da direita para a esquerda. As linhas da imagem são numeradas de 0 a 1599. Assim não se aguarda o posicionamento do carro de impressão do eixo Y para cada linha da imagem. Essa otimização obriga que o envio dos *pixels* obedeça este ordenamento.

Após todo o *array* de 400 *bits* ter sido percorrido e os respectivos *pixels* desenhados, o firmware envia uma mensagem para o programa *desktop* enviar os próximos 400 *pixels*, e assim sucessivamente até percorrer todos os *pixels* da imagem e gravá-la no papel.

4.3 RESULTADOS

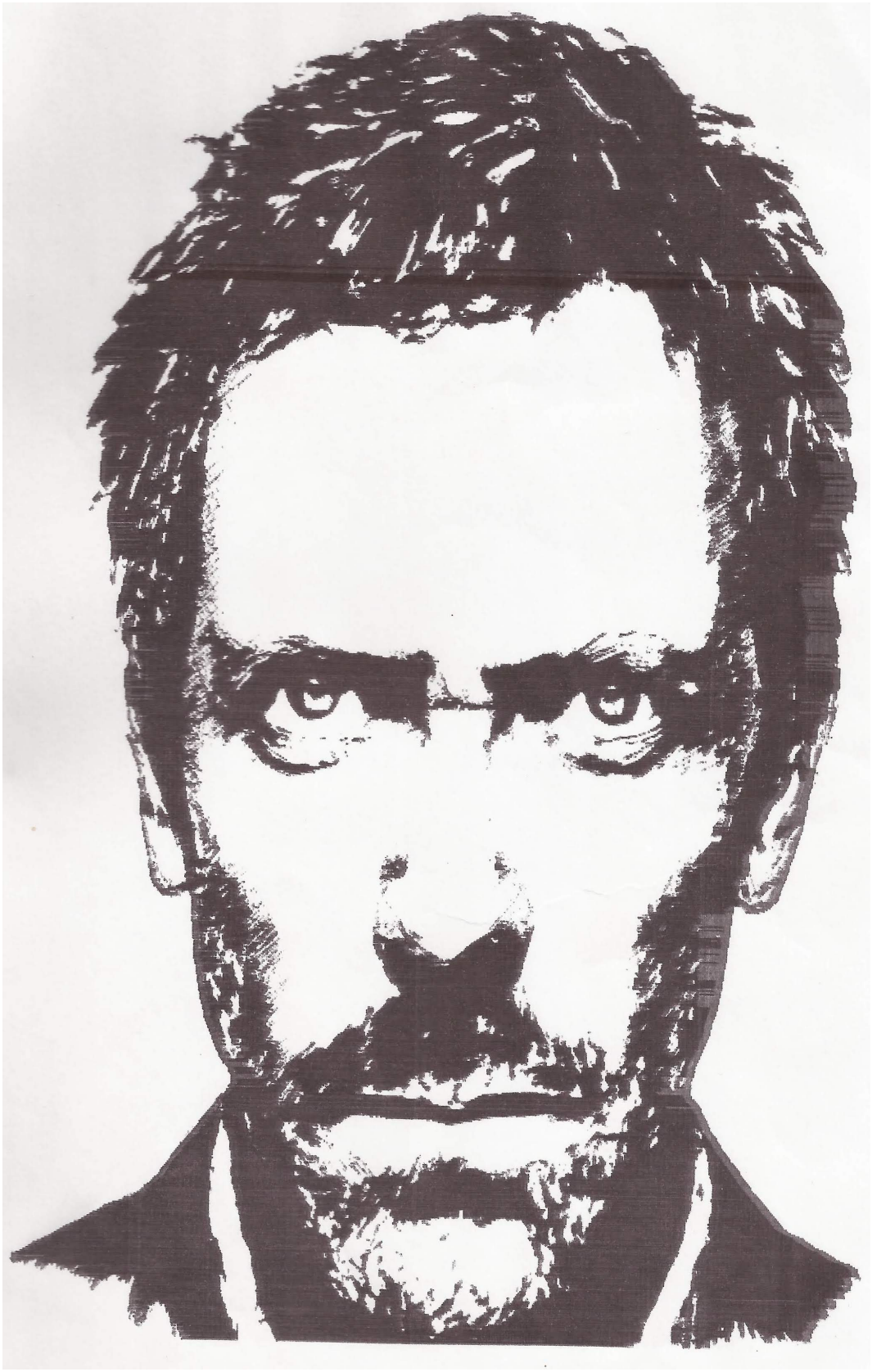
As Figuras 20 e 21 mostram duas figuras impressas com o CNC. Elas foram escaneadas e recortadas para otimizar o espaço. A Figura 22 não foi recortada. Ela mostra o tamanho real de um desenho que ocupa toda a altura e largura suportados pelo CNC.

Figura 20 – Tux, o mascote oficial do *kernel* Linux.



Fonte: O autor.

Figura 21 – Ator James Laurie.



Fonte: O autor.

Figura 22 – Folha A4 escaneada com impressão de imagem de autor desconhecido.

Fonte: O autor.

4.3.1 Limitações

Apesar de atingir o objetivo principal e os objetivos secundários propostos, o *hardware* e *software* do CNC possuem limitações. As principais limitações são apresentadas a seguir.

- *Limite de resolução para impressões de 1200 por 1600 pixeis*. Essa limitação é resultado da conexão direta dos motores de passo dos eixos X e Y aos carros de cada eixo. Caso caixas de redução fossem utilizadas, resoluções maiores seriam suportadas. Consequentemente, imagens com mais detalhes poderiam ser impressas.

- *Só é possível imprimir com uma única cor*. Adicionar mais mecanismos ao eixo Z com canetas esferográficas de outras cores não é desafiador. O maior problema está em garantir o alinhamento das canetas.

- *Os pixeis verticais não tem o mesmo tamanho que os pixeis horizontais*. É uma consequência da utilização de motores de passo com quantidades de bobinas distintas. Outros motores devem ser usados para corrigir essa limitação.

- *Alimentação de folhas A4 é um processo manual*. O mecanismo de entrada e saída de folhas A4 é uma das partes mecânicas mais complexas de uma impressora. Projetar e construir um mecanismo preciso para o CNC é um desafio em aberto.

- *O tempo de impressão pode ser de várias horas*. Subir e descer a caneta esferográfica leva o mesmo tempo que ejetar e introduzir a bandeja de leitor de CD. Desta forma, quanto mais detalhes uma imagem tiver, além de seu tamanho, mais tempo levará para ser impressa.

- *Não é possível pausar e retomar uma impressão*. De todas as limitações, esta talvez seja a de mais fácil endereçamento.

- *Não é possível desenhar apenas 1 pixel*. Cada *pixel* é transformado em dois *pixels* pelo CNC devida a incapacidade da caneta esferográfica apenas encostar no papel e pintar o *pixel*.

- *É necessário pausar a impressão para atualizar o buffer da imagem sendo impressa*. Como o microcontrolador não suporta *threads*, não é possível enviar a imagem e imprimí-la em simultâneo. Uma forma de amenizar as pausas durante a impressão é aumentando a velocidade do RS-232.

5 CONCLUSÃO

O desenvolvimento do *hardware* e *software* para o CNC apresentado neste trabalho demonstrou ser possível reutilizar componentes de impressoras descartadas para criar um sistema funcional e inovador. A abordagem adotada permitiu superar limitações de custo, viabilizando a construção de um equipamento capaz de desenhar em papel com precisão e eficiência.

Durante o projeto, foram enfrentados desafios técnicos que exigiram soluções criativas, como a integração de motores com características distintas e a implementação de uma comunicação eficiente entre o microcontrolador e o computador. A utilização de um protocolo de comunicação desenvolvido especificamente para este trabalho foi essencial para assegurar a confiabilidade do sistema.

No campo do *software*, a aplicação *desktop* desenvolvida ofereceu múltiplas possibilidades de interação com o CNC, desde o controle por *joystick* até a impressão de imagens. Além disso, a inclusão de ferramentas como a conversão de imagens para tons de cinza e a manipulação gráfica contribuiu para ampliar a usabilidade do sistema.

Os resultados obtidos comprovam a viabilidade do projeto, destacando o potencial da automação baseada em CNCs para promover inovação tecnológica de baixo custo. No entanto, o sistema apresenta limitações, como a impossibilidade de imprimir em múltiplas cores. As limitações abrem caminho para trabalhos futuros.

REFERÊNCIAS

Allegro. **Allegro**. 2009. Online. Disponível em https://web.archive.org/web/20090123105632/https://liballeg.org/.

COSTA, E. B. L. d. **O invento de Jacquard e os computadores: alguns aspectos das origens da programação no século XIX**. 2008.

Custom Computer Services, Inc. **PIC16F877A**. 2007. Online. Disponível em https://web.archive.org/web/20071109010341/https://www.ccsinfo.com/product_info.php?products_id=877areprog.

FERREIRA, C. G. **O fordismo, sua crise e o caso brasileiro**. [S.l.], 1993. Disponível em https://www.cedeplar.ufmg.br/pesquisas/td/TD%2065.pdf.

FRANCISCO, A. M. S. **A história e as diferenças entre um microcontrolador e um microprocessador**. 2008. Online. Disponível em https://web.archive.org/web/20081021103432/http://amsfrancisco.planetaclix.pt/download/PICs/Micros_historia.pdf.

GEDEA. **Controle de Motores de Passo Unipolares de Quatro Fases de Ímã Permanente (5 ou 6 fios)**. 2005. Online. Disponível em https://web.archive.org/web/20090212181636/http://br.geocities.com/gedaepage/Doc/MP_5fios.htm.

GTK+. **The GTK+ Project**. 2007. Online. Disponível em https://web.archive.org/web/20090220073302/http://gtk.org/overview.html.

HIRAKAWA, A. R. **Interfaces Seriais**. 2005. Online. Disponível em http://sites.poli.usp.br/d/pcs2529/index_arquivos/serial.pdf.

MEZENGA, B. **Motores de Passo**. 2000. Online. Disponível em https://web.archive.org/web/20090130050203/http://www.geocities.com/CollegePark/Dorm/ 8863/motordepasso.htm.

ROSáRIO, J. M. **Automação industrial**. [S.l.]: Barauna, 2009. ISBN 978-8579230004.

TAVARES, J. M. **Introdução ao Controlo Numérico Computorizado - I Conceitos Gerais**. [S.l.], 2009.

TEIXEIRA, F. M. P. **Revolução Industrial**. [S.l.]: Ática, 1997. 1–38 p. (O Cotidiano da História). ISBN 8508036434.