UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADORES

DESENVOLVIMENTO DE UMA PLACA CONTROLADORA E DA INTERFACE DE CONTROLE E COMUNICAÇÃO PARA IMPRESSORAS 3D

TRABALHO DE CONCLUSÃO DE CURSO

Diones de Vargas Dutra

Santa Maria, RS, Brasil

2013

DESENVOLVIMENTO DE UMA PLACA CONTROLADORA E DA INTERFACE DE CONTROLE E COMUNICAÇÃO PARA IMPRESSORAS 3D

Diones de Vargas Dutra

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Redes de Computadores da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Tecnólogo em Redes de Computadores

Orientador: Profº . Ms. Walter Priesnitz Filho

Santa Maria, RS, Brasil

2013

Universidade Federal de Santa Maria
Colégio Técnico Industrial de Santa Maria
Curso Superior de Tecnologia em Redes de Computadores

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Conclusão de Curso

DESENVOLVIMENTO DE UMA PLACA CONTROLADORA E DA INTERFACE DE CONTROLE E COMUNICAÇÃO PARA IMPRESSORAS 3D

elaborado por
Diones de Vargas Dutra

como requisito parcial para obtenção do grau de
Tecnólogo em Redes de Computadores

COMISSÃO EXAMINADORA:

Walter Priesnitz Filho, Ms.
(Presidente/Orientador)

Claiton Colvero, Dr. (UFSM)

Murilo Cervi, Dr. (UFSM)

Santa Maria, 25 de Janeiro de 2013.

AGRADECIMENTOS

Universidade Federal de Santa Maria – pela qualidade do ensino público e gratuito;
Carla Juliana Biesdorf – minha namorada, pela calma e compreensão da minha ausência neste período conturbado de conclusão de curso;
Walter Priesnitz Filho – ao meu orientador, pelo apoio e ajuda neste trabalho;
Comunidade do software livre – sem as boas ideias deles não teria concluído este trabalho;

A todos aqueles que, de alguma forma, contribuíram para a realização deste trabalho, e não estão nominalmente citados.

EPIGRAFE

“Se você é um carpinteiro e está fazendo um belo armário de gavetas, você não vai usar um pedaço de compensado na parte de trás porque as pessoas não o enxergarão, pois ele estará virado para a parede. Você sabe que está lá e, então, usará um pedaço de madeira bonito ali. Para você dormir bem à noite, a qualidade deve ser levada até o fim.”

— STEVE JOBS

RESUMO

Trabalho de Conclusão de Curso
Curso Superior de Tecnologia em Redes de Computadores
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UMA PLACA CONTROLADORA E DA INTERFACE DE CONTROLE E COMUNICAÇÃO PARA IMPRESSORAS 3D

AUTOR: DIONES DE VARGAS DUTRA
ORIENTADOR: WALTER PRIESNITZ FILHO
Local da Defesa e Data: Santa Maria, 25 de Janeiro de 2013.

Uma nova tendência de tecnologia está surgindo, as impressoras 3d, porem com um custo bastante elevado ainda, então neste trabalho foi desenvolvido uma maneira mais simples de criar uma impressora de código aberto, visando remover componentes complexos da confecção do hardware e optar por componentes mais baratos.

Também será o foco deste trabalho a abstração da impressora para quem tiver o interesse de melhorá-la, pois uma quantidade significativa de dados do processamento que nas demais impressoras é no próprio dispositivo, será transferido para o computador, visto que o usuário que utilize uma impressora.

Para uma impressão já existe a necessite de um computador para gerar as imagens ou adquiri-las, para leigos em micro informática, este projeto pode ajudar no desenvolvimento e melhorias na programação de impressoras 3d, diminuindo seu custo e melhorando seu desempenho, pois pode utilizar algoritmos de maior complexidade com um aumento de poder computacionais.

Palavras-chave: Impressora 3D. Pinguino. Python. Pynguino.

ABSTRACT

Undergraduate Final Work
Course of Technology in Computer Networks
Federal University of Santa Maria

DEVELOPMENT OF A PLATE PARENT AND INTERFACE CONTROL AND COMMUNICATION FOR PRINTERS 3D
AUTHOR: DIONES DE VARGAS DUTRA
ADVISOR: WALTER PRIESNITZ FILHO
Defense Place and Date: Santa Maria, March 25st , 2013.

A new trend is emerging technology, 3D printers, but with one fairly high cost yet, so this work was developed a simpler way create a printer open source, aiming to remove components of the complex manufacturing of hardware and opt for cheaper components.

It will also be the focus of this work to abstraction printer for those who have interest to improve it, because a significant amount of data processing than in other printers is the device itself, is transferred to the computer, since the user using a printer.

For an impression exists a need for a computer to generate images or acquire them, to lay in micro computing, this project can help develop and improvements in programming 3d printers, reducing cost and improving performance, because it can use more complex algorithms with a power computing.

Key words: Printer 3D. Pinguino. Pynguino. Python.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

STL	Stereolithography
RP	*Rapid prototyping*(prototipagem rápida)
FDM	*Fused Deposition Modeling*(Modelagem por fusão e deposição)
SLS	*Selective Laser Sintering*(sinterização seletiva a laser)
CNC	Controle Numérico Computadorizado
PID	Proporcional Integral e Derivado
PIC	*Peripheral Interface Controller* (Controle de interface periférica)
USB	*Universal Serial Bus*(barramento de serial universal)
IDE	*integrated development environment*(ambiente de desenvolvimento integrado)
PCB	*printed circuit board*(placa de circuito impresso)
PWM	Pulse width modulation(Modulação de largura de pulso)
PVC	Polyvinyl chloride
CDC	*communications device class*(comunicações classe de dispositivo)
CSV	*comma separated values*(valores separados por vírgula)
CPU	*central processing unit*(unidade central de processamento)
PLA	Polylactic acid
SLA	Stereolithography

SUMÁRIO

INTRODUÇÃO

O presente trabalho demonstra o desenvolvimento de uma interface e placa para controle de uma impressora 3d, bem como todos critérios para seu funcionamento, como eletrônica, software e a comunicação com um computador.

Objetivo geral

O desenvolvimento de um protocolo de comunicação para impressoras 3d, visando a criação da comunicação com um hardware simplificado, diminuindo a complexidade da construção, e transferir todo processamento da impressora para um computador, aumentando toda capacidade computacional e flexibilidade na programação.

Objetivo específico

• Promover a troca de dados entre dois equipamentos, computador e placa mãe da impressora.

• Verificar a ocorrência de falha na transmissão dos dados padrao2

• Desenvolver um software que interprete código G, código atualmente usado em impressoras 3d.

• Melhorar a linguagem de programação para que o código possa ser disponibilizado e aberto.

• Desenvolver uma placa mãe para a impressora que suporte comunicação com um computador pessoal através da porta USB.

Estrutura do trabalho

No capítulo 2 é apresentado um breve histórico das impressoras 3d, e para onde está evoluindo as impressas de código aberto e tipos diferentes de impressoras. No capítulo 3 é descrito o problema da implementação. No capitulo 4 o projeto começa a ser criado, sera demonstrado problemas e soluções. No capítulo 5 são demonstrado os resultados obtidos e no capítulo 6 uma conclusão do trabalho.

1 REVISÃO TEÓRICA

A Impressão em 3D, segundo Celana (2009), é uma das técnicas de prototipagem rápida cuja origem se baseia em duas técnicas: a topografia e a foto escultura. A primeira foi um método desenvolvido por Blanther no final do século XIX para a construção de mapas que apresentassem o relevo, e consiste na impressão de uma série de discos de areia contendo as curvas de nível das cartas topográficas.

Já no início da década de 70, Matsubara (da Mitsubishi Motors) propôs um processo fotográfico. Regiões de uma camada de foto polímero recoberta por pó de grafite ou areia eram endurecidas após a exposição à luz, e mais tarde as outras partes eram retiradas com a utilização de um solvente. Verificou-se que essa técnica poderia ser empregada para reproduzir as superfícies de fabricação complexa, em função da operação da máquina.

A técnica da foto escultura nasceu no século XIX com o propósito de criar cópias exatas tridimensionais de objetos. Frenchman Francois Willème realizava o posicionamento de 24 câmeras fotográficas igualmente distribuídas em torno de um objeto colocado no centro de uma sala circular, sendo todas as câmeras acionadas simultaneamente. Com o contorno gerado por cada foto, um artista esculpia cada posição em sua respectiva referencia, em um cilindro para formar o objeto.

De modo a reduzir o trabalho de escultura, desenvolveu-se uma técnica que utilizava uma luz graduada para expor uma gelatina fotossensível, que se expande proporcionalmente ao contato com a água. Essas pesquisas originaram as técnicas atualmente empregadas na RP, obtendo êxito comercial a partir do lançamento da SLA-1, pela *3D Systems* em 1987, conforme descrito por Sachs (Sachs ,1990). A empresa desenvolveu e patenteou o processo de estereolitografia, como também desenvolveu o formato STL, utilizado até os dias de hoje na indústria. Nos anos seguintes, outras empresas de RP surgiram vagarosamente, comercializando suas próprias versões de estereolitografia. Em 1991, a Stratasys inovou ao inserir no mercado uma nova tecnologia: modelagem por fusão e deposição (FDM – *Fused Deposition Modeling*). Em seguida, a DTM Corporation introduziu a sinterização seletiva a laser (SLS –*Selective Laser Sintering*), processo em que o calor de um laser é utilizado para fundir metais pulverizados,como o titaneo, e vários outros foram aparecendo mais tarde.

Anunciado em 2010(CARNETT, 2010), Bre Pettis e e outros apresentaram o *kit CNC CupCake*: uma impressora 3D capaz de gerar qualquer objeto de menos de quatro centímetros em um lado usando dois tipos de plástico. O objetivo da empresa é fazer de fabricação caseira, barata e comum. O código da configuração é aberto, podendo ser copiado ou mesmo

modificado. Ainda em 2010, segundo Dillow (Dillow, 2010), a tendência da impressora é gerar itens feitos de areia e vidro, visto que novos produtos químicos ajudam na liga das camadas e novos mecanismos tornam as camadas mais precisas. Vasos, luminárias e outros itens com formas delicadas, impossíveis de serem impressas antes, poderão ser realizadas.

Para esse tipo de impressão será utilizado o pó de vidro, que é inserido em um forno portado na impressora que faz o vidro fundir a um formato desejado. Objetos de argila poderão ser gerados com precisão de até 0,02 cm de espessura, porque uma válvula de alívio de pressão lixa o excesso. O inventor Enrico Dini gostaria de imprimir edifícios, pois seu fabricante , a Dshape, pode criar estruturas de pedra de até 16 por 16 por 10 metros. Esse tipo de impressão permitirá formas e curvas que são difíceis e caras, com a construção de concreto convencional. Dini pretende trabalhar com arquitetos em tijolos enormes.

Em setembro de 2012 a empresa MakerBot anunciou(ALBANESIUS, 2012) o lançamento da impressora *Makerbot Replicator 2 Desktop 3D*, dando início à quarta geração de impressoras 3D da empresa. A impressora é projetada para engenheiros, pesquisadores, profissionais criativos, ou qualquer um que gosta de fazer as coisas. O objetivo é impressão de qualidade profissional, e para isso houve uma atualização de software para a impressão ficar mais rápida e consistente. O volume da área de construção é trinta e sete por cento maior do que a MakerBot original, permitindo que mais peças possam ser impressas simultaneamente. A altura de cada camada de material é duas vezes e meia mais fina do que a altura da camada da MakerBot anterior, lançada no início de 2012. A impressora fornece camada com resolução de cem mícron e um volume de construção de quatrocentos e dez polegadas cúbicas, garantindo uma impressão de qualidade, sem a necessidade de ser lixada ou ter tratamentos pós-produção. A impressora já está disponível através do site da empresa, distribuidores e loja principal por 2.199 dólares, e é entregue ao cliente totalmente montada.

O sítio da MakerBot afirmava que o projeto de montar a impressora demorava em média dois dias para que duas pessoas pudessem concretizá-lo. O processo de montagem era simples e as instruções constavam nas páginas do site *wiki* da empresa.

A parte mais complicada de fazer um kit de impressora 3D é a codificação de software que lhe diz o que fazer. Além de poder montar seu próprio objeto, Makerbot.com possui exemplos de objetos prontos para serem impressos e o sítio Thingaverse.com hospeda milhares de modelos 3D que o usuário pode baixar e usar.

Segundo Carnett (Carnett,2010) , o topo da impressora é a placa controladora de temperatura, pois o plástico sólido entra no topo da impressora e a extrusora move-se

derretendo o plástico, e imprimindo um objeto em três dimensões. Para o usuário, algumas habilidades em codificação são exigidas, dependendo do software usado. Partes diferentes do desenho requerem diferentes temperaturas e velocidades, e isso pode causar obstrução dos bicos de impressão, exigindo limpeza manual.

2 ANÁLISE DO PROJETO

O problema se divide em duas partes bem distintas, uma é o controle de um robô do tipo cartesiano e a outra é a o controle da temperatura que derrete o plástico para a extrusão.

2.1 Robô cartesiano

Um robô é um manipulador programável, multi funcional, projetado para movimentar materiais, peças, ferramentas ou dispositivos especiais, usando movimentos variados programados para a execução de diferentes tarefas(RIVIN, 1988). Um robô convencional é uma estrutura formada por um conjunto de elos ligados por articulações, sendo os movimentos possibilitados pelos elos sucessivos e formado pelos seguintes elementos:

• Base - base do manipulador.

• Elos - elementos que fornecem robustez e rigidez.

• Juntas - proporcionam movimento entre elos

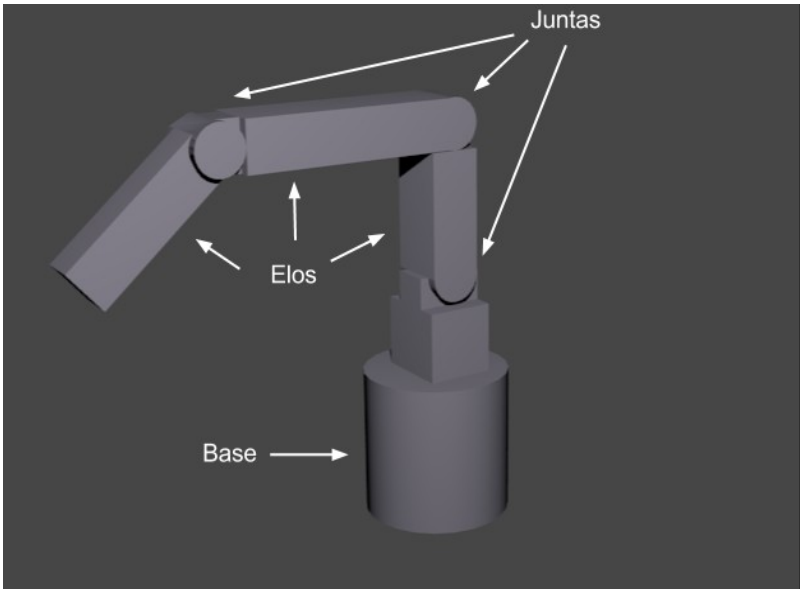

Figura 1: Disposição dos itens de um robô

Fonte: Autor

Uma junta prismática é uma variação das demais juntas, com seu deslocamento linear ao invés de um deslocamento angular. Este trabalho apresenta um robô cartesiano com 3 graus de liberdade, com 3 juntas prismáticas para fornecer seus movimentos.

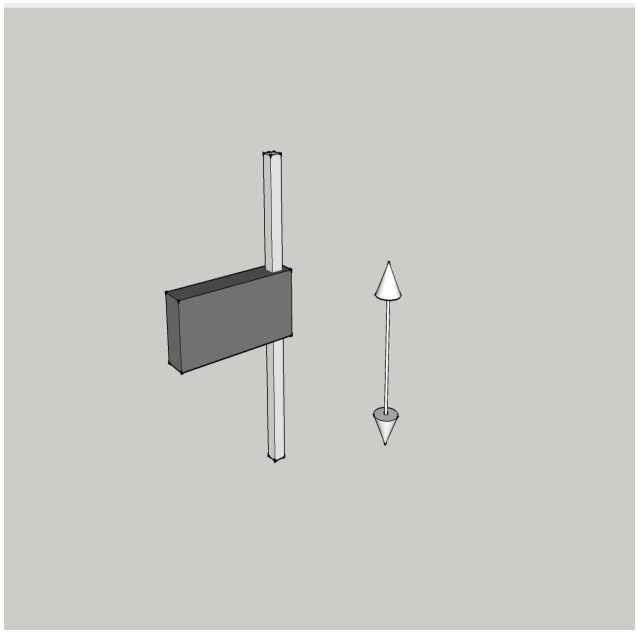

Figura 2: Junta prismática

Fonte: Autor

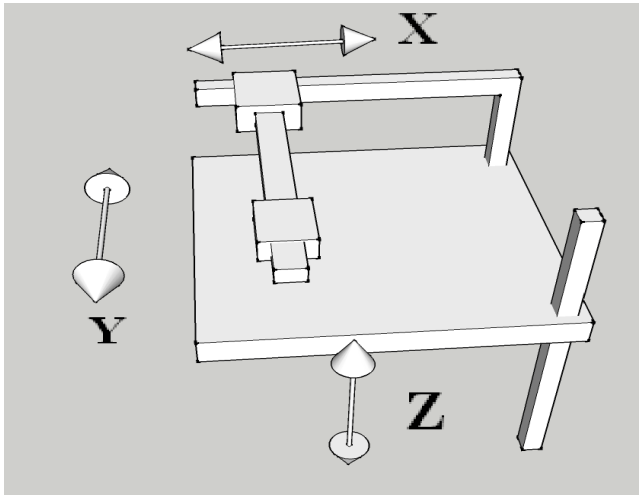

Figura 3: Robô cartesiano com 3 graus

Fonte: Autor

2.2 Atuadores

Atuadores são os componentes responsáveis por transmitir força a um robô, e podem ter seu deslocamento tanto linear quanto angular. Como exemplo de atuadores podem-se citar motores e pistões pneumáticos.

2.3 Temperatura

Para que o plástico escorra pelo bico, um ponto de temperatura especifico deve ser atingido para cada tipo de material. Esse ponto de temperatura para o derretimento deve ser controlado por software.

2.3.1 Controle PID

O controle Proporcional, Integral e Diferencial (PID) é uma técnica que consiste em corrigir o erro e atenuar o valor de uma variável de controle, baseando-se valor desejado e o no atual.

Quadro 1: Descrição do processo PID

Proporcional(P)	Correção proporcional ao erro	A correção a ser aplicada ao processo cresce na proporção que cresce o erro entre o valor atual e o desejado.
Integral(I)	Correção proporcional ao produto TEMPO X ERRO	Pequenos erros que se prologam por muito tempo devem ser corrigidos de forma intensa.
Derivado(D)	Correção proporcional à variação do erro	Se o erro está variando muito rápido,esta taxa de variação deve ser reduzida para evitar oscilações.

2.3.2 Extrusor

O extrusor é composto por um atuador, que é responsável por movimentar o filamento de plástico para o bico aquecido onde será derretido. O bico aquecido é composto por resistores, que são a fonte de calor para o derretimento do material, e um bico metálico com um pequeno furo por onde escorre o plástico.

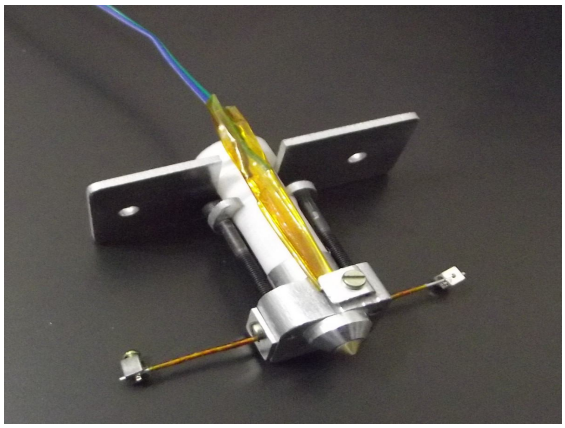

Figura 4: Bico para derreter o plástico

Fonte: Movtech Impressora 3d

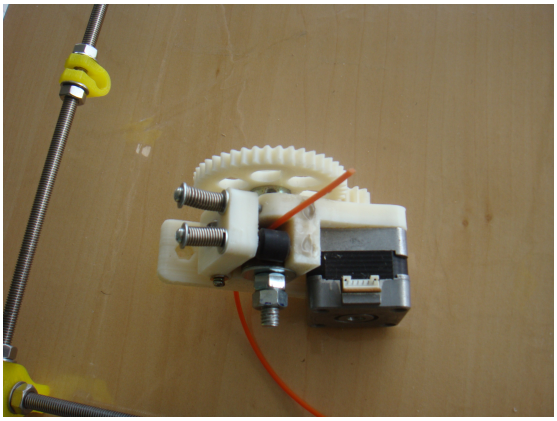

Figura 5: Atuador para levar o plástico ao bico

Fonte: REPRAP

Com o problema interpretado, como a dificuldade de controle da temperatura e o uso de atuadores para mover a impressora, partiu-se para a busca de soluções.

3 DESENVOLVIMENTO DA SOLUÇÃO ENCONTRADA

Muitas soluções podem ser empregadas para resolver este trabalho, mas foi buscada a simplicidade, menor número de componentes e o custo reduzido, que a linha de desenvolvimento encontrada seguiu.

3.1 Construção do hardware

Para um programador acostumado a desenvolver em plataformas *desktop*, é um grande desafio atingir o mundo real, e em mecanismos que estejam fora do computador pessoal. Para isso foi empregado o Pinguino(PINGUINO, 2012), que é uma plataforma pronta e já testada e bem consolidada. Trata-se de uma plataforma de código e hardware aberto, com uma comunidade nova e com grande expansão, por se tratar de um hardware muito simples e de fácil construção. A placa Pinguino possui um microcontrolador PIC 18f2550, fabricado pela Microchip(MICROCHIP, 2012), e possuindo uma porta USB nativa, importante para comunicação com computadores, Devido a ausência de uma porta serial ou porta paralela nos computadores mais modernos e na maioria dos notebooks. Segue algumas características do PIC usado, retiradas do *datasheet*. São apresentadas no quadro 2.

Quadro 2: Descrição retirada do *datasheet*

Device	Flash(bytes)	SRAM	EEPROM	I/O	A/D	CCP	SPI	I2C(Master)	USART	Timers(8/16 bit)
PIC18F2550	32K	2048	256	24	10	2	1	1	1	1/3

• Compatível com USB V2.0

• Velocidade baixa (1,5 Mb / s) e velocidade máxima (12 Mb / s)

• Suporta interrompição, transferências isócronos e Bulk

• Suporta até 32 terminais (16 bidirecional)

• 1-Kbyte RAM acesso dupla para USB

• On-chip transceptor USB com on-chip regulador de tensão

• Interface para off-chip transceptor USB

• Transmissão de porta paralela (SPP) para transferências de Transmissão USB (dispositivos 40/44-pin apenas)

Baseado neste projeto e no mesmo microcontrolador, foi desenvolvido uma placa para realizar a comunicação do computador com o periférico. O *bootloader* do Pinguino, um arquivo com extensão *hex* deve ser transferido para o PIC, pois ao adquiri-lo o microcontrolador vem sem nenhum programa na memória, é preciso gravá-lo através de um dispositivo especifico. Quando este arquivo é enviado ao microcontrolador, este se encarrega de montar um dispositivo USB com capacidade de escrever em seus próprios endereços de memória.

Com o *bootloader* em execução pode-se regravar os códigos do microcontrolador através de sua própria porta USB sem a necessidade de um dispositivo pra isto. Para gravar o *bootloader* no PIC foi usado o gravador PIC KIT2, com uma versão mais amigável e gráfica para Windows. Este também possui uma versão para Linux, porém por linha de comando.

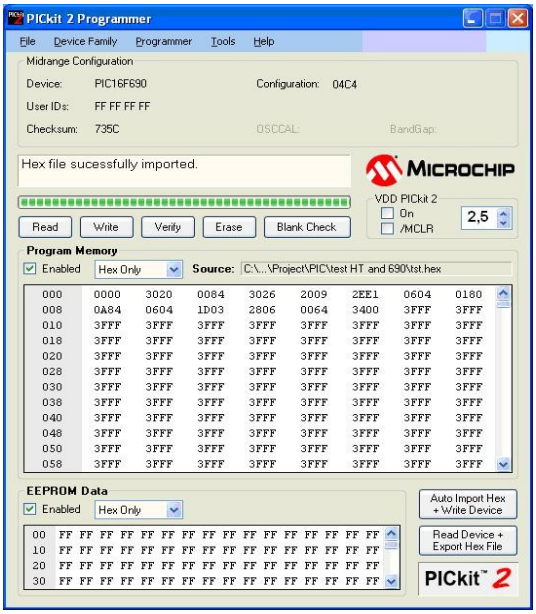

Figura 6: Programa para gravar o microcontrolador

Fonte:Autor

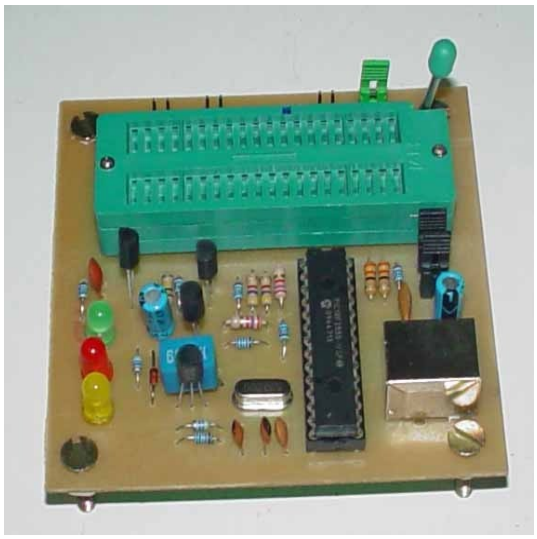

Figura 7: Hardware para gravar o microcontrolador

Fonte:m-azrul.blogspot.com

Após gravar o PIC18f2550, foi criada a placa com os componentes necessários para que este funcione, como cristal, capacitores, resistores, entre outros. Utilizando a ferramenta Fritzing(FRITZING, 2012), de código aberto,com uma interface amigável e com vários modos de edição, é possível desenhar a placa direto na *protoboard* ou gerar o esquemático em uma placa de circuito impresso profissional de forma automática. Dois protótipos foram criados na placa tipo matriz de contato, tanto para o Pinguino, quanto para o restante que faz parte da eletrônica de maior potência, que farão a ligação com os motores e resistores do extrusor.

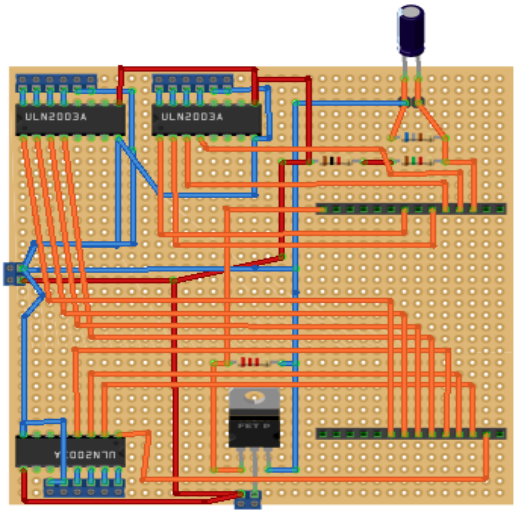

Figura 8: Placa para controle dos motores e resistores de aquecimento

Fonte:Autor

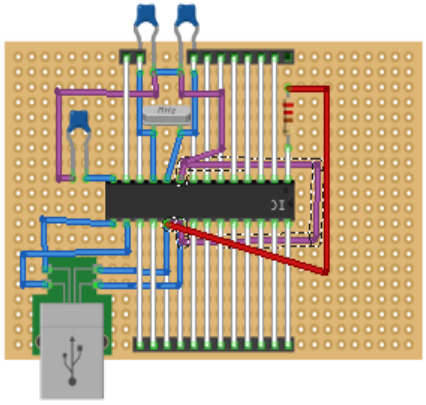

Figura 9: Placa com o microcontrolador

Fonte;autor

Após desenhar as placas, e ter uma simulação de como o protótipo ficará, o software gera de forma quase automática o esquemático para ajudar na documentação do projeto. Com algum ajuste na posição dos componentes na tela, o restante é gerado pelo programa, o que ajuda muito iniciantes na construção de projetos que envolvam eletrônica.

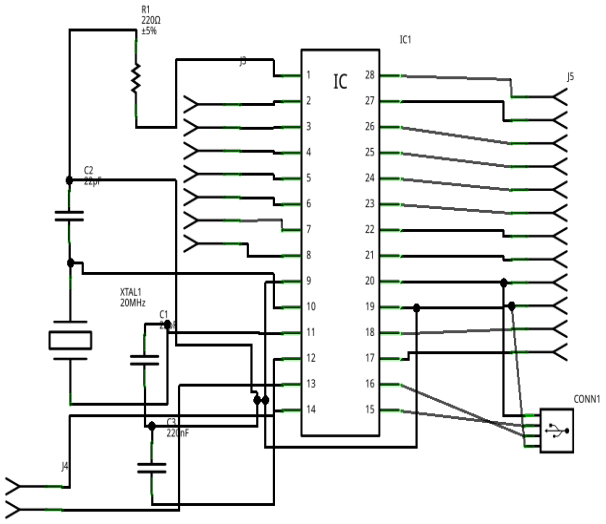

Figura 10: Esquemático da placa do microcontrolador

Fonte:Autor

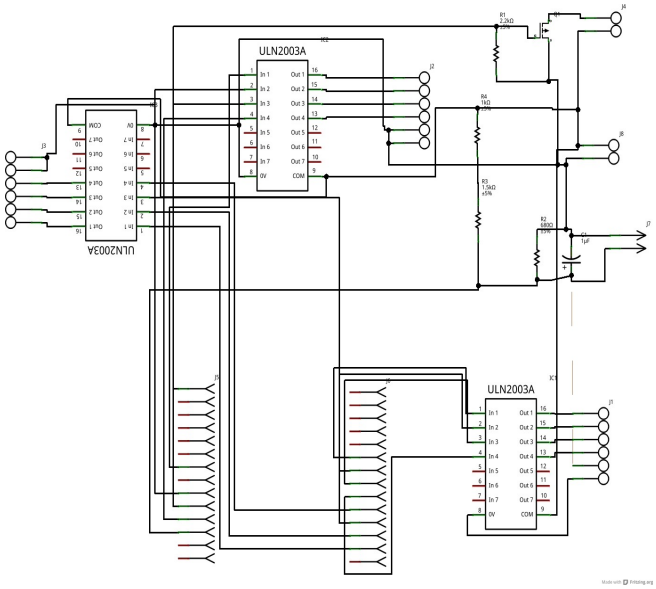

Figura 11: Esquemático da placa do circuito de controle

Fonte:Autor

Para um trabalho pode-se exigir a confecção de uma placa de circuito impresso, ou até mesmo para projetos maiores, a construção em longa escala exige que seja confeccionado algo mais prático de ser construído. No caso deste trabalho, um pequeno protótipo, fica mais fácil de se alterar componentes na placa de matriz do que em uma PCB (*Printed Circuit Board*), no caso de uma alteração na PCB seria necessário a confecção de uma nova placa.

Pela facilidade da construção de placas no fritzing foram criados placas para este projeto, mas não foram utilizadas, para criá-las bastou um *click* e o software se encarregou de gerá-la com todas as trilhas corrigidas e com o melhor caminho. Com alguns conhecimentos sobre solda de componentes eletrônicos, foi implementada a placa proposta.

Inicialmente não foram efetuados testes de conexões, vários componentes foram queimados na pressa de concluir esta etapa, e devido a atrasos na aquisição de novos componentes, teve que ser adotada uma política de sempre testar as ligações antes de realizá-las. Um suporte para circuitos integrados foi utilizado, para evitar que o microcontrolador fosse danificado por superaquecimento. Com o *bootloader* do Pinguino, não há necessidade de remover o microcontrolador para enviar um novo programa para o mesmo, pois este já tem a capacidade de regravá-lo sem a necessidade do dispositivo de gravação, mas se o Pinguino for perdido, deve-se regravar do modo convencional.

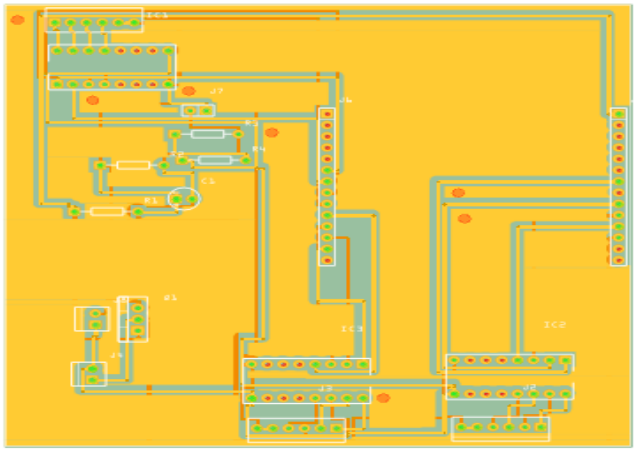

Figura 12: Placa para controle dos motores e resistores de aquecimento

Fonte:Autor

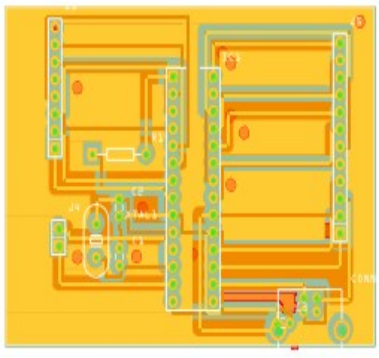

Figura 13: Placa com o microcontrolador

Fonte:Autor

Uma estrutura de teste também foi criada, baseada em impressora de código e software aberto, como REPRAP(REPRAP, 2012) e MAKERBOT(MAKERBOT, 2012). Foi criada uma impressora também de código aberto, com a proposta de apresentar uma forma simples, conhecida e que não intimidasse ao olhar. Então uma forma geométrica, um cubo, porém devido a algumas cinemáticas a forma foi um pouco alterada. O material escolhido foram canos de PVC, pelo baixo custo e rápida fabricação, e realmente se mostrou útil sua construção.

Houveram diferença entre testes realizados somente com a placa e a placa na estrutura e com ruídos causados pelos componentes adicionados, bem como diferentes situações. E o projeto que foi desenvolvido terá que atender a uma situação real, não apenas a uma situação hipotética de comunicação por simulação ou sem as intemperes de interferências de diferentes componentes. Níveis mais baixos de erro por exemplo exigiriam um protocolo mais simples, bem como uma taxa baixa de comunicação e com muitos erros exigiriam um protocolo mais complexo e que consiga regenerar a mensagem original. Após o término da construção da estrutura, que após a fase de projeto, foi rápido e de fácil construção, aproximadamente algumas horas, o que demorou um pouco mais, foi fixar os motores. Então foi criado a primeira comunicação para controle dos motores.

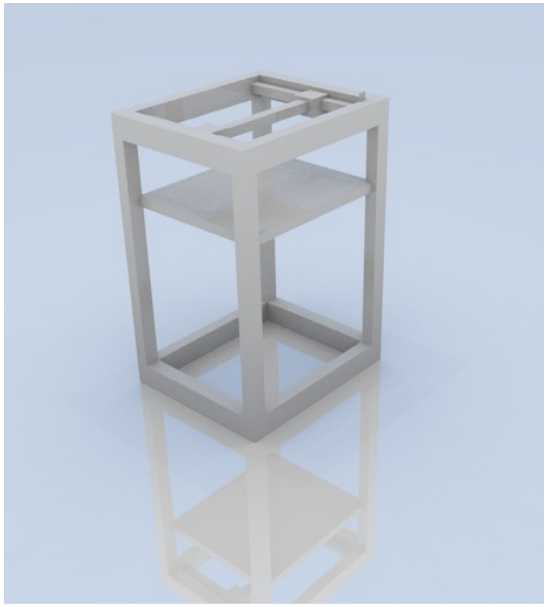

Figura 14: Esboço da estrutura

Fonte:Autor

Figura 15: Primeira estrutura de teste

Fonte:Autor

3.2 Desenvolvimento do software

A linguagem de programação inicial foi o JAVA(JAVA, 2012), mas devido a problemas em acessar a porta serial, devido ao java ser uma linguagem de alto nível e estar rodando em uma máquina virtual. Assim não tem acesso ao hardware, para isso existem bibliotecas para esse acesso, cada biblioteca acessa um tipo e versão de sistema operacional, depois de algumas atualizações no sistema operacional utilizado, um Linux Ubuntu 11.04(UBUNTU, 2012), a biblioteca parou de funcionar.

Com pouca documentação e a dependência de procurar a versão correta da biblioteca,esta linguagem foi abandonada, porém uma boa parte do trabalho já tinha sido feito, como as telas gráficas do usuário e um interpretador de código.

Uma segunda abordagem foi utilizar Python(PYTHON, 2012), uma linguagem de *script* e de alto nível, com módulos prontos para realizar toda comunicação serial. Esta mostrou-se uma boa solução por já tratar da versão do sistema operacional e ter uma comunidade de desenvolvedores bem ativa para corrigir erros, assim como muitos fóruns e tutoriais para resolver problemas. Em pouco tempo foi migrado o código de Java para Python, apenas com o interpretador funcionando. Então foi criado o protocolo que proverá a comunicação com a impressora, baseado em protocolos de redes industriais, vistos no decorrer do curso, como o Zigbee(ZIGBEE, 2012), que é simples, e apresenta boa funcionalidade. Foi usado um *start byte* e *checksum* para verificação de erros.

Quadro 3: Estrutura de um pacote

Start	Pacote de Dados	*Checksum*

Uma vez que as trocas de mensagem são de pouca variação quanto ao tamanho do pacote, um identificador define tanto o tipo de pacote, quanto o seu tamanho.

Existem apenas três tipos de pacotes a serem enviados: um para mover os eixos, incluindo o eixo pertencente a extrusora, outra para recebimento de mensagem *ack* ou *nack* e por ultimo para controle da temperatura.

Quadro 4: Tamanho e função de um pacote

ID	Tamanho do pacote	Função
0x01	5 bytes	mover eixos
0x02	2 bytes	informação da transmissão de dados
0x04	3 bytes	recebimento e envio de temperatura

Com a base já formada para a comunicação, a estrutura física praticamente pronta e a eletrônica completamente formada, foram escritos *scripts* para teste, e com o Pinguino instalado no PIC, um programa foi criado para realizar a comunicação de teste.

Nos exemplos da IDE de programação do Pinguino existem bons modelos para emular uma porta serial através da USB, utilizando um *driver* CDC(*Communications Device Class*) (CDC, 2012). Com este drive a USB se comporta como uma porta serial, executando a comunicação do dispositivo, sendo praticamente todo gerenciamento realizado no computador host, troca de dados são implementados pelo protocolo CDC, são abstraídas e removidas na aplicação que é executada em ambas as partes da conexão. Para a aplicação uma porta comum serial é criada, e o sistema operacional juntamente com o *driver* se encarrega de criar uma porta virtual e controlar seu tráfego de dados.

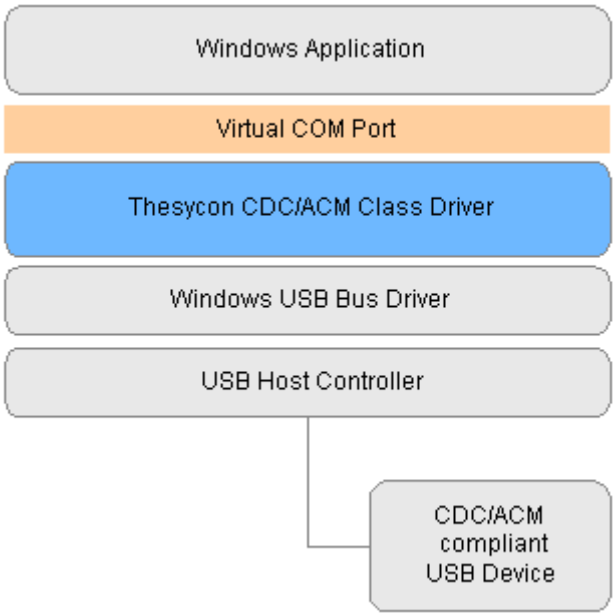

Figura 16: Pilha do protocolo CDC

Fonte:www.thesycon.de

Com o *bootloader* instalado no Pinguino, basta programar na IDE o programa de teste e fazer uso de uma função que utilize o CDC para que o *driver* seja carregado. O compilador

identifica a necessidade e o adiciona ao código, com toda implementação do *driver* e configurações necessárias para que o dispositivo seja reconhecido pelo sistema operacional.

Dentre eles o PID(*Product* ID) e o VID(*Vendor* ID), que identificam o produto e fabricante, são números únicos e devem ser comprados, porém não há necessidade.

A Microship, empresa que fabrica os PICs, comprou alguns números e os disponibiliza gratuitamente para que possam ser utilizados em projetos. Se o projeto for de grande porte, é necessário a compra de PID e VID, caso contrário o dispositivo aparece para o sistema operacional pertencendo à Microship.

A velocidade da USB é estabelecida de forma física, utilizando a elevação de tensão em um dos pinos para uma taxa de transmissão alta o pino D+, que é chamado de modo de alta velocidade(480 Mb/s)(USB, 2012), deve elevar a tensão com um resistor ligado em 3.3 volts. E para uma taxa de transmissão baixa, modo de baixa velocidade(1.5Mb/s), um resistor deve ser ligado no pino D-. Sem tensão em ambos os terminas, não há dispositivo conectado.

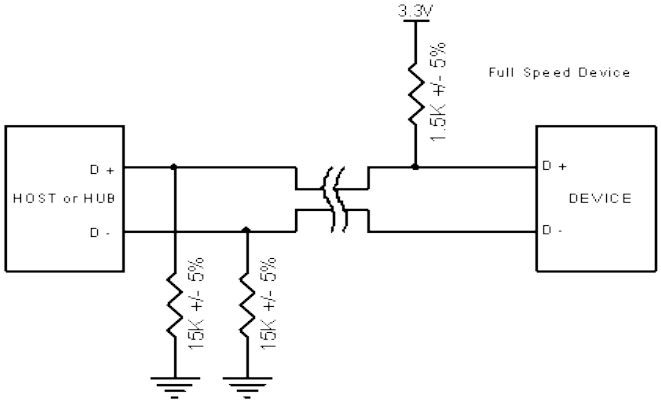

Figura 17: USB alta velocidade

Fonte:hw-server.com

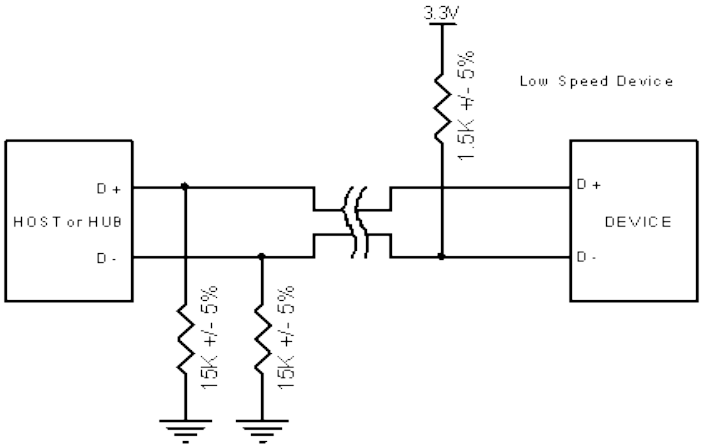

Figura 18: USB baixa velocidade

Fonte:hw-server.com

Para USB 2.0 existe um terceiro modo de operação, onde o dispositivo possui a alta velocidade e ao ser conectado, negocia com o *host* o modo de máxima velocidade(12 Mb/s).

O microcontrolador utilizado suporta a baixa velocidade e a máxima velocidade. Ajustados automaticamente por um resistor programável, através da velocidade ajustada para o *clock*, uma velocidade de 24 MHz é baixa velocidade e 48 MHz é máxima velocidade, no Pinguino é utilizado a máxima velocidade. Com um exemplo de comunicação USB CDC para o Pinguino e um *script* em Python para testar a comunicação, para facilitar a elaboração, todo processo foi dividido em estados, o fluxograma abaixo foi criado e os estados foram transcritos para o Pinguino.

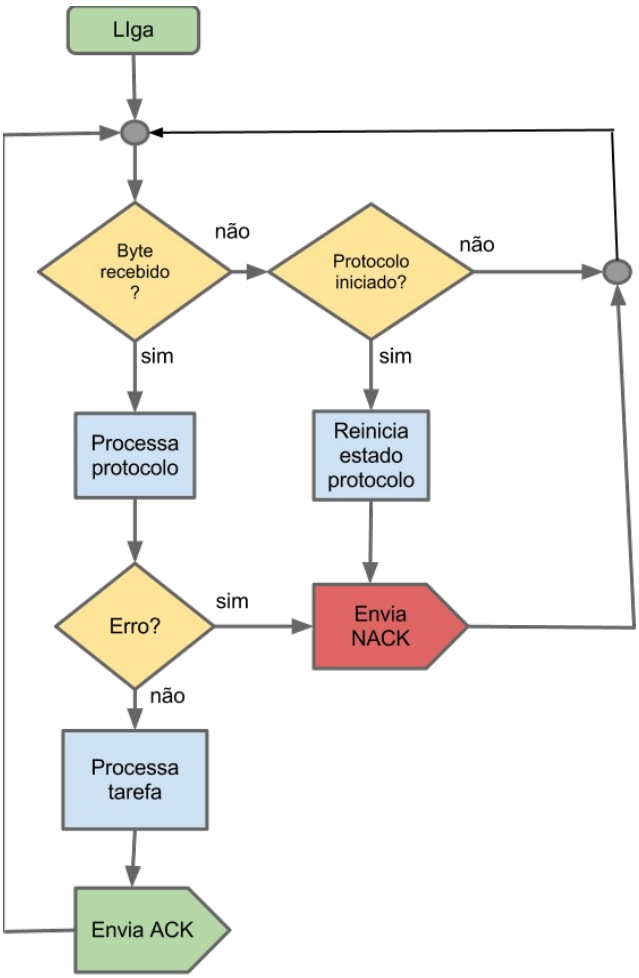

Figura 19: Fluxograma

Fonte: Autor

O programa fica sempre verificando a chega de um byte na serial emulada pela USB. Se um byte chegar, o protocolo é processado, montando um *array* de bytes até que o tamanho seja atingido. O tamanho é delimitado pelo segundo byte recebido, que informa o ID e este por sua vez se refere a um tamanho específico de pacote. Após atingir o tamanho correto, é verificada a ocorrência de erro através do algoritmo de verificação *checksum*. Se não houve erro, um *ack* é enviado e se houver erro é enviado um *nack*. Existe uma outra situação em que o programa está em estado de recebimento de dados para resolver, e os bytes param de chegar.

Provavelmente um byte foi perdido e o programa aguarda sua chegada. Então o algorítimo detecta e envia um *nack* e o computador reinicia o pacote, pois um *nack* faz com que ocorra esta ação.

Para realizar a comunicação no *script* em Python, um módulo para comunicação serial apenas é definido e a porta com a velocidade de transmissão de dados, um vetor contendo todo pacote é enviado a serial, e se o que for retornado um *ack*, é registrado um acerto e no caso de um *nack* o erro é registrado. Para obter os resultados da taxa de erro da comunicação a saída do código foi gravada em um arquivo de texto csv. Python sendo uma linguagem bem mais intuitiva do que o C utilizado no Pinguino, foi muito menos trabalhado o protocolo, pois já utiliza um forma melhorada para tratar da comunicação. Como por exemplo o tamanho do pacote, basta dizer quantos bytes serão recebidos, que um vetor é criado e o módulo controla a chegada até atingir o tamanho, se o tamanho não é atingido em um tempo máximo estipulado pelo programador ou não, um erro ocorre.

```
ser=serial.Serial('/dev/ttyACM0',115200)
vetor=[0x7e,0x1,0x1,0x2,0x1,0x1,6]
erros=0
acertos=0
    for j in range(10000):
        for i in vetor:
            ser.write(chr(i))
            x=ser.read(4);
            if ord(x[2])==2:
```

```
                acertos+=1
        else:
                erros+=1
    time.sleep(.02)
```

Acima um trecho de código que realiza toda a comunicação, nos teste realizados foram transmitidos dez mil pacotes, e o processo foi repetido dez vezes. Sendo obtido uma taxa média de apenas 9 erros para cada dez mil pacotes, que é considerada uma taxa muito boa de comunicação.

Quadro 5: Pacote para movimento

Start	ID	X	Y	Z	E	*Checksum*
0x7e	0x01	0x01	0x01	0x01	0x02	0x06

Quadro 6: Pacote de ack

Start	ID	*ack*	*Checksun*
0x7e	0x02	0x01	0x03

Quadro 7: Pacote de nack

Start	ID	*nack*	*Checksun*
0x7e	0x02	0x02	0x04

O *startbyte* é responsável pelo início da comunicação, o programa fica aguardando sua chegada e só então inicia o protocolo. O ID se refere a um pacote de movimento de motores com tamanho de 5 bytes. Sabendo desta informação o protocolo fica em um laço recebendo os bytes e adicionando-os em um vetor. Ao termino do recebimento, é realizado o cálculo de *checksum* e se for igual ao byte de conferência, sétimo byte, o pacote é verificado como íntegro, as funções devem ser executadas. E um pacote de *ack* deve ser enviado para o computador, para que a próxima mensagem ser enviada. Caso ocorra um erro, então um *nack* é enviado e uma retransmissão ocorre.

Um byte 0x01 nos eixos referentes ao X,Y,Z e E(extrusor) indica retorno de um passo dos motores, o 0x02 indica que o motor deve ficar parado e o 0x03 indica um passo de avanço do motor. Segundo Gonçalves(BRITES, 2008), motores de passo são dispositivos eletro mecânicos que convertem pulsos elétricos em movimento mecânico que geram variação angular discreta, e sua velocidade varia conforme a frequência dos pulsos.

Os motores de passo possuem quatro bobinas e cada uma ligada em um pino do Pinguino. Com a ajuda de um circuito integrado ULN2003A para aumentar a corrente de saída, devido ao fato do Pinguino suportar apenas 20 mA por saída e o motor exigir aproximadamente 150 mA, para o controle contendo cada pino de saída correspondente a uma bobina. Por exemplo:

u8 motorx[4]={15,14,16,13};

Como visto no vetor acima o pino 15 do Pinguino está ligado na primeira bobina do motor o pino 14 na segunda e assim por diante. Para fazer com que o motor gire em um sentido basta alternar pulsos com a sequência das bobinas e para inverter a rotação inverte-se a sequência das bobinas. O controle que foi feito de maneira simples, percorrendo-se o vetor em um sentido ou em outro e uma variável controla em que passo o motor está, para saber a próxima bobina a ser ligada. Um motor de passo pode ser controlado de forma simples, pois segue uma lógica digital de controle, possui um alto torque mesmo em rotações muito baixas, boa precisão de posicionamento, e praticamente nenhuma manutenção.

Quadro 8: Funcionamento de um motor de passo

nº passo	**Bobina 1 pino 15**	**Bobina 2 pino 14**	**Bobina 3 pino 16**	**Bobina 4 pino 13**
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Este capítulo abordou a lógica que será seguida para resolver o problema proposto, assim como as soluções de hardware e software a serem seguidas no decorrer do trabalho.

4 RESULTADOS

Este capítulo trata da coleta de dados para validar os algorítimos e hardwares empregados até o momento, demonstrando a viabilidade do uso de cada ferramenta.

Após muitos testes de validação, o algoritmo de controle e o protocolo de comunicação entre o dispositivo e o computador se mostraram eficientes e capazes de realizar o controle de movimento da impressora.

Com o primeiro problema resolvido, partiu-se para o segundo, o controle de temperatura para o derretimento do plástico. Esse controle deve evitar oscilações na temperatura, uma vez que diferenças muito grandes de temperatura podem danificar o plástico, tornando-o opaco e quebradiço. Para isso um simples controle liga e desliga não funciona adequadamente, devido à ineficiência de utilizar toda potência ou desligá-la por completo.

Para resolver esta questão foi escolhido um controle PID, não sendo a única alternativa, mas como outras impressoras de código aberto utilizam, é uma boa prática optar por esta alternativa, bem como pelo tempo de realização do trabalho ser curto. Testes com outras formas de controle seriam inviáveis. Houve uma alteração no fluxograma, em cada ciclo de programa é processado o controle de PID. Para que o PID funcione é preciso que o algoritmo tenha uma entrada analógica e uma saída também analógica.

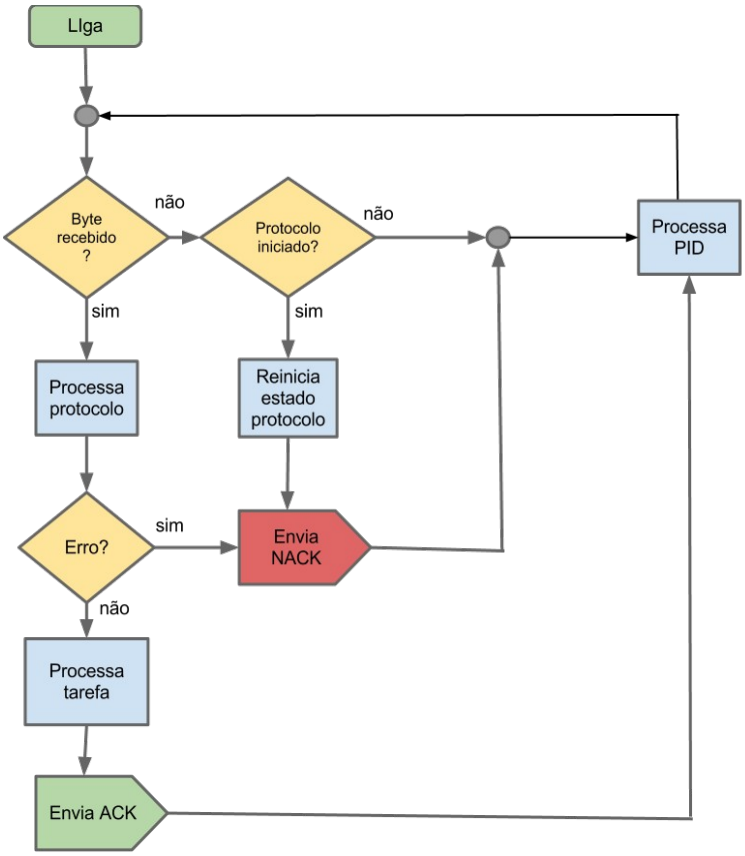

Figura 20: Fluxograma com PID

Fonte:Autor

Para ler a entrada analógica, a tensão é dividida em 10 bits, que resulta em 1023 posições, sua tensão limite de leitura é de cinco volts, cada bit varia por aumento de 4,88mV, ou seja 5/1023, isto limita a leitura de pequenas variações e também para tensões muito baixas. A porta analógica é utilizada para ler a temperatura através de um NTC(*Negative Temperature Coefficient*), que varia sua resistência conforme a temperatura aumenta, sua resistência diminui. Foi utilizado um NTC de 10kohms e para corrigir a pouca variação da resistência com o aumento da temperatura.

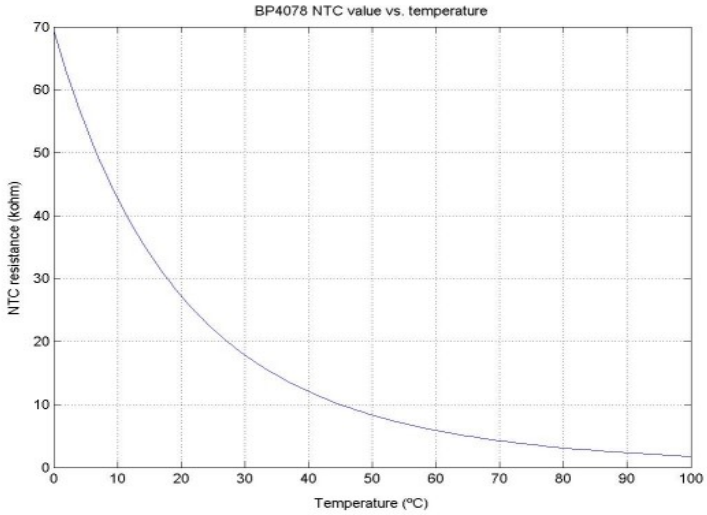

Figura 21: Curva do NTC

Fonte:REPRAP

Foi utilizado o

esquema adotado em impressoras REPRAP, onde um divisor de tensão é criado para suavizar um pouco a variação. E com a ajuda de um *script* em Python, é criado pontos para determinar a temperatura, pois um outro problema é a variação não ser linear, exigindo cálculos mais complexos e envolvendo logaritmos, por se tratar de um processador de apenas 8 bit, seria um uso muito grande de memória e processamento, por ser um sistema embarcado, estes recursos são escassos.

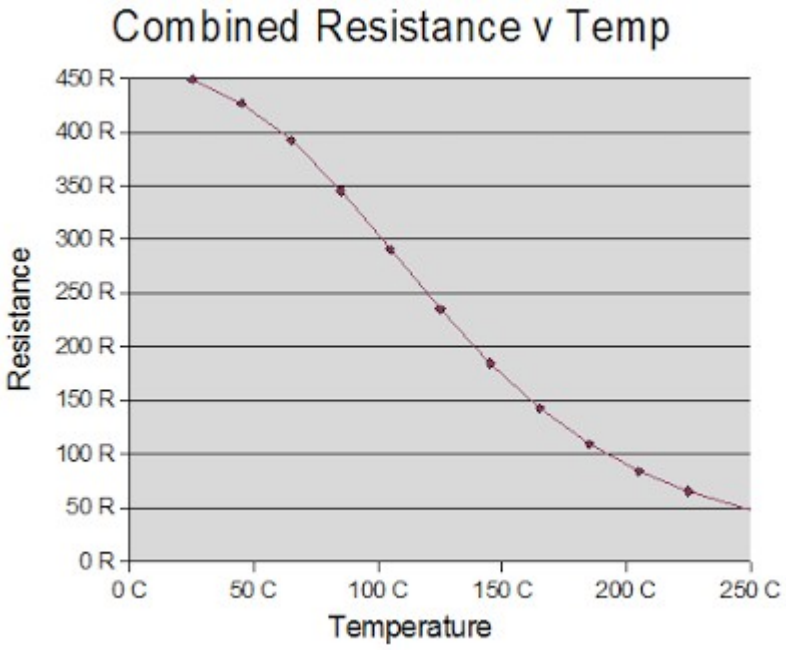

Figura 22: Curva com a combinação de componente

Fonte:REPRAP

O esquema adotado é para um NTC de 10kohms, com um resistor r1 de 680 ohms, r2 de 1,6kohms, sendo este não encontrado na faixa comercial. Teve de ser colocado um resistor de 1,5 kohms e mais um de 100 ohms em série, pela composição do NTC ser apenas um material que varia sua resistência com a temperatura e por características do material de construção do mesmo, durante a leitura ocorre oscilações no valor, para corrigi-lo é adicionado um capacitor em paralelo com o NTC pra estabilizar as variações, o capacitor utilizado foi de 10 micro Faraday.

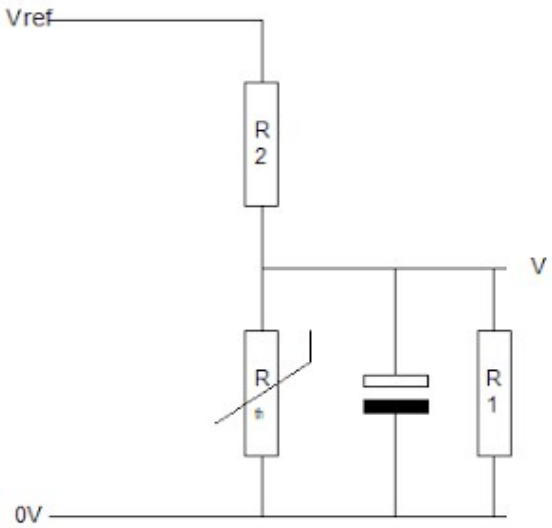

Figura 23: Esquemático com os componentes

Fonte:REPRAP

A saída de tensão analógica do PID é feita através de uma saída de chaveamento PWM, onde um pulso de onda quadrada é aplicado(HIRZEL, 2012) com uma frequência aproximada de 500 Hz. A variação de tensão depende do tempo em que a onda fica em ciclo ativo em relação ao ciclo que não está ativo, sendo uma relação direta.

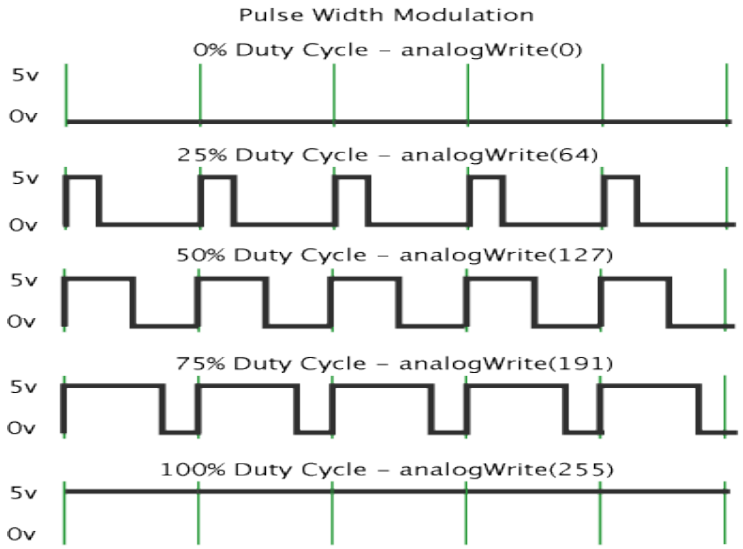

Figura 24: Saída PWM

Fonte:arduino.com

Se o ciclo for de 50% ativo, temos metade da tensão da fonte, no caso do exemplo, 2,5 volts, se tivermos 25%, 1,25 volts e assim por diante. Para que o Pinguino suporte toda potência de aquecimento da resistência foi colocado um Mosfet IRLZ44N, com um *driver* interno para acionamento, não necessitando da construção de um, e diminuindo o número de componentes do projeto. Com a faixa de acionamento a partir de 3 volts, os resistores utilizados para o aquecimento são de 5,6 ohms ligados em 12 volts, sendo assim com uma corrente de i=v/r, i=12/5,6, i=2,14 amperes, sendo duas resistências, corrente total é de 4,28 amperes, o Mosfet suporta até 45 amperes e com uma resistência interna de 0,022 ohms, não houve aquecimento significativo durante o funcionamento, e sem a necessidade de colocar um dissipador de calor para o componente.

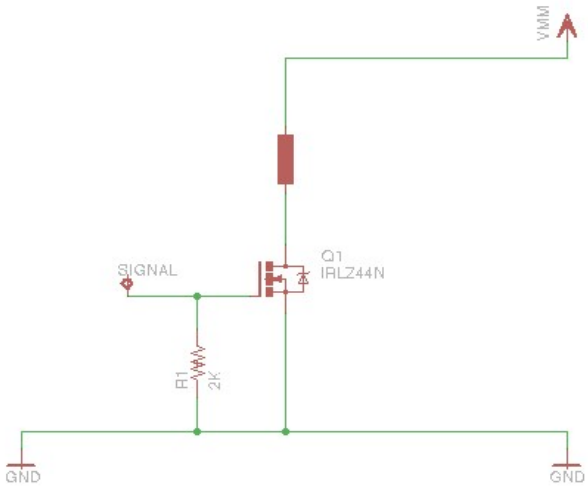

Figura 25: Esquemático utilizando o Mosfet

Fonte:REPRAP

O algoritmo para controle de PID foi baseado no Yarf(AGTEN, 2012), juntando meu código de controle e mais o PID, a memória de programa foi para aproximadamente 85% da CPU, em outros casos que já havia testado, com um uso grande da memória o microcontrolador começa a apresentar comportamentos imprevisto, com o *upload* do novo código a comunicação começou a falhar e apenas 10% dos pacotes começaram a chegar sem erros, tornando a comunicação inviável. O código do PID foi desligado, e apenas as linhas continuaram no programa e o erro persistiu, comprovando que foi o uso de memória que causou o erro na transmissão, com um teste de 92% da memória utilizada, o microcontrolador não conseguiu carregar o programa, ficou reiniciando o código toda vez que saia do modo de *bootloader*, podendo ser um erro do compilador. Foram testadas outras versões do compilador, tentativas de alterar linhas e diminuir o código, e todas as tentativas sem sucesso.

Durante as pesquisas para achar uma solução para o problema, foi encontrado o projeto Pynguino(ÁLVAREZ, 2012), que também utilizando o projeto Pinguino, foi criado um protocolo de comunicação com a placa, que realiza praticamente todas as funções do PIC pelos comandos do protocolo, muito semelhante ao que estava criando, porém com todas as funções do Pinguino. Com uma troca de mensagem em texto e sem a confirmação da entrega, e com a mesma emulação de serial pela USB, porém o projeto Pynguino possui a facilidade de um módulo feito Python e para ser utilizado em *scripts* do próprio Python, que realizam toda a comunicação com a placa. E de forma simples abstrai a placa, cria-se um objeto Pinguino, que para o programador se parece com uma ação comum a programação, acesso a

pinos, entradas e saídas do Pinguino, com a diferença de tudo estar rodando dentro do Python e sendo programado nesta linguagem, o modulo que se encarrega de enviar os comandos para placa executar.

```
Pinguino=PinguinoProcessing()
Pinguino.Conect('/dev/ttyAcm0')
led1=1
led2=8
Pinguino.pinMode(led1,'output')
Pinguino.pinMode(led2,'output')
tempo=.5
while True:
    time.sleep(tempo)
    Pinguino.digitalWrite(led1,"high")
    Pinguino.digitalWrite(led2,"low")
    time.sleep(tempo)
    Pinguino.digitalWrite(led1,"low")
    Pinguino.digitalWrite(led2,"high")
```

Com o exemplo acima, identifica-se o uso do modulo em relação a abstração da comunicação, o programador nem percebe que existe o protocolo e toda a comunicação com a placa, pois este código, ao contrario da IDE não é enviado a placa, ele é executado totalmente no computador, e apenas os comando de escrita alta ou baixa nos pinos é enviada. em pouco tempo todo o código que foi desenvolvido inicialmente para o Pinguino foi convertido para Python. Devido a facilidade de tratar as variáveis, somente uma alteração no módulo foi necessária, pois não havia a confirmação da chegada dos pacotes de escrita em um pino. Isto é essencial para o trabalho, para isso bastou editar o módulo, devido o fato do Python ser uma linguagem bem limpa e intuitiva, não necessita de muita documentação, o código fica bem claro para quem tem um conhecimento da linguagem. Então foi acrescentada uma saída no código do Pinguino, caso o pacote de gravação em um pino chegue corretamente um 'OK' é retornado, e no módulo se chegar um 'OK' na gravação de um pino, ele é repassado para

quem solicitou. Assim no código que for criado, saberei se houve uma gravação ou não. Com o algoritmo de PID funcionando corretamente, foi executado alguns testes de calibração dos ganhos.

4.1 Discussão dos Resultados

Esta sessão apresenta as discussões dos resultados obtidos nos testes realizados com o controle de temperatura, utilizando o controle de PID.

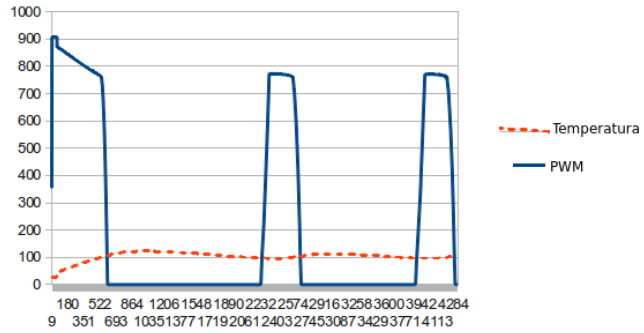

Figura 26: Controle PID com ganhos muito elevados

Fonte:Autor

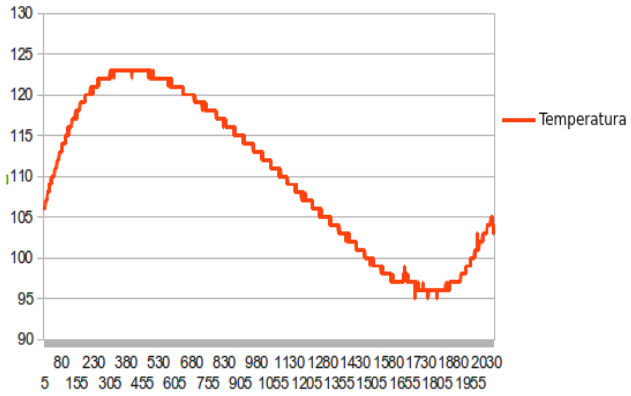

Figura 27: Detalhe da variação de temperatura

Fonte:Autor

Como visto na figura acima, sendo em linha continua a potência da resistência aplicada pelo PWM e em trasejado a temperatura do bico extrusor. Com os ganhos muito elevados o controle se comporta como um liga desliga. E na imagem a esquerda, percebe-se com mais clareza que existe uma variação muito grande da temperatura, mesmo depois da temperatura chegar a 100 graus, que neste caso foi o valor setado, e quando o valor é atingido a resistência é desligada. Porém a potência era muito elevada e ela continua trocando calor com o material até atingir a estabilidade e voltar a descer. chegando até a uma elevação de aproximadamente 25 graus acima do valor desejado. Isto prejudica muito o derretimento do plástico. Os valores foram então diminuídos, e o tempo entre as execuções do código foram aumentadas, para que o controle integral não cresça muito, pois ele é um somatório do erro, se o tempo for curto ele irá aumentar muito rápido, o tempo escolhido foi 800 milissegundos, anteriormente era de 100 milissegundos. Com esta nova calibração o algorítimo acertou a relação de potência com a temperatura até atingir um equilíbrio.

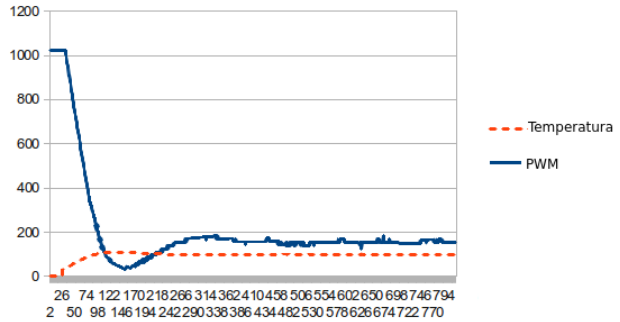

Figura 28: Controle PID com ganhos corrigidos

Fonte :Autor

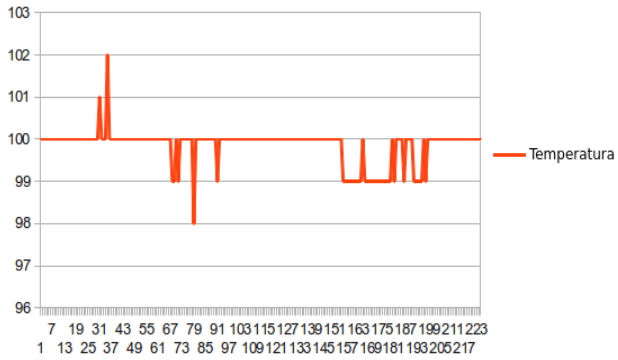

Figura 29: Detalhe da variação de temperatura do teste final

Fonte; Autor

Em linha continua percebe-se o comportamento da potência para que a temperatura se estabilize em 100 graus da melhor maneira possível, sem ocasionar muitas oscilações no valor com o decorrer do processo. Uma boa vantagem desse controle, além de ser estável, o desgaste do Mosfet é menor, pois a potência é bem menor que passa através dele, com um aquecimento baixo há um aumentando da vida do componente. A ligação desses algorítimos de controle com as impressoras de código aberto atuais ocorre pelo código gerado, que é inserido nas impressoras, e as mesmas interpretam e fazem a impressão. Este projeto se diferencia dos demais, devido ao fato de praticamente todo processamento ser realizado no computador, e não na impressora. Um programa feito em Python interpreta código G, que é o usado por todas impressoras 3D, e este programa se encarrega de trabalhar com as demais partes criadas neste trabalho. O código G informa o deslocamento que deve ocorrer nos eixos x,y e z, bem como nesse um eixo adicional encarregado da saída do plástico, o da extrusora, que foi denominado sendo o eixo E. A temperatura também é informada nas primeiras linhas do código G.

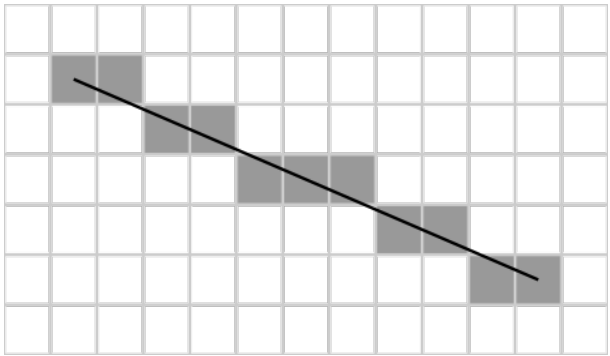

Figura 30: Algorítimo Bresenham

Fonte:Henrik Abelsson, Abelsson

Para o deslocamento dos eixos, foi usado um algorítimo Bresenham, que interpreta o deslocamento de uma linha em qualquer direção e qualquer ângulo, para que os motores de passo saibam seu caminho passo a passo.

O algorítimo de Bresenham informa cada passo que de ser dado no eixo x e y para criar a linha na matriz, usando esta função, foi utilizado para realizar o deslocamento dos motores, passando para o restante do código cada passo que dever ser feito nos eixo no decorrer do deslocamento para chegar ao destino, e basta decidir se vai ligar ou não a extrusora, o deslocamento do eixo z sempre é realizado de forma única, sem mais outros deslocamentos.

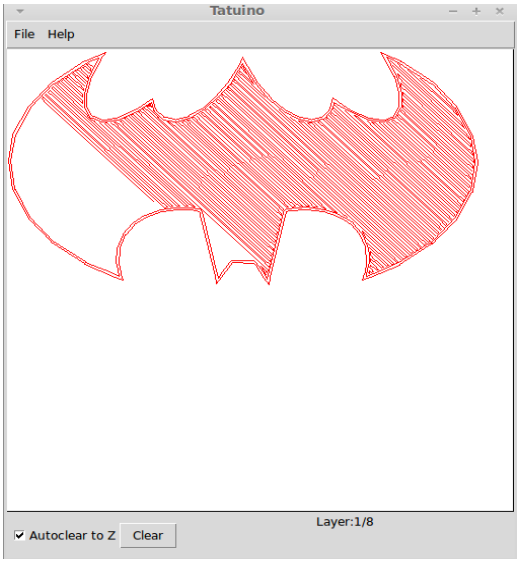

Figura 31: Tela do programa criado

Fonte:Autor

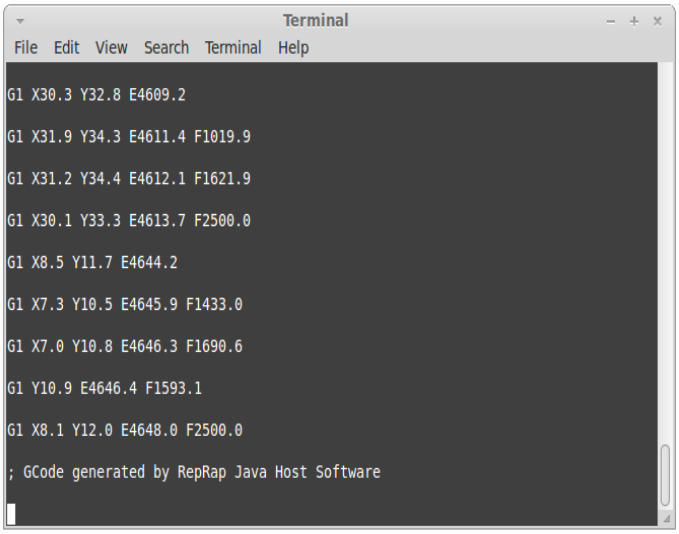

Figura 32: Detalhe do código G sendo interpretado

Fonte: Autor

O arquivo com o código G, tendo o comando iniciando na linha com G1, informa um deslocamento, e os valores x e y o quanto deve ser movido em milímetros, este código é gerado pelo mesmo software usado em outras impressoras 3D, contendo uma boa portabilidade de uma grande quantidade de arquivos já criados para outras impressoras de outros usuários. O deslocamento dos motores foi implementado no código, a impressora se moveu sem erros, o teste foi feito com uma caneta no lugar da extrusora, e a impressora fez o mesmo desenho que foi gerado no computador, só que apenas em dois planos, x e y, mostrando a viabilidade do uso deste trabalho, quanto a temperatura foi apenas testado o controle, e o aquecimento do plástico, bem como uma extrusão com inserção manual do PLA(plástico utilizado), o algorítimo de controle de temperatura para 165 graus célsius, também funcionou.

5 CONCLUSÃO

Com este trabalho foram testados muitos conceitos de comunicação de dados e eletrônica, pois a tarefa de comunicação de dados está muito ligada ao meio físico. Também é muito importante o uso de ferramentas livres, pois em muitos casos elas disseminam-se muito mais que as proprietárias e conseguem uma boa usabilidade pelos usuários que as desenvolvem e até mesmo aos usuários comuns, que podem criar novas funcionalidades, e resolver problemas. Sair do conforto do ambiente de programação controlado dentro de um computador não foi fácil pois erros surgiram e novos conhecimentos foram necessários. Apesar disso, como há base teórica e boa assimilação do conteúdo a solução foi pesquisar e adaptar-se aos novos conceitos. Afinal, o mercado de trabalho exige sempre a adaptação e busca por novas tecnologias.

REFERÊNCIAS

AGTEN, P. **YARF**. Acessado em 30 de Novenbro de 2012, http://code.google.com/p/yarf/source/browse/src/temperature/pid.c.

ALBANESIUS, C. **MakerBot Unveils Replicator 2 3D Printer.** Acessado em 14 de Novenbro de 2012, http://www.pcmag.com/article2/0,2817,2409928,00.asp.

BRITES, F. G. Motor de Passo. **PETELA**, Niterói,RJ,Brasil, p.3, 2008.

CARNETT, J. B. Making the MakerBot: for less than $1,000, the makerbot kit provides nearly everything you need for your very own 3d plastic printer. we find out what it takes to build and use one. **Popular Science**, EUA, ESTADOS UNIDOS, v.277.1, p.82, 2010.

CDC. **Application Note AN1164**, USB CDC Class. Acessado em 27 de Novembro de 2012, http://ww1.microchip.com/downloads/en/AppNotes/01164a.pdf.

CELANI, G. Digitalização tridimensional de objetos:um estudo de caso. **SIGraDi 2009 sp**, São paulo, SP, BRASIL, v.13, p.309–311, 2009.

DILLOW, C. Print a product: nextgen machines create objects out new materials. **Popular Science**, EUA, ESTADOS UNIDOS, v.277.3, p.30, 2010.

FRITZING. Acessado em 22 de Novenbro de 2012, http://fritzing.org.

HIRZEL, T. **PWM**. Acessado em 30 de Novembro de 2012, http://arduino.cc/en/Tutorial/PWM.

JAVA. Acessado em 25 de Novenbro de 2012, http://www.java.com/.

MAKERBOT. Acessado em 25 de Novenbro de 2012, http://www.makerbot.com/.

MICROCHIP. Acessado em 20 de Novenbro de 2012, http://www.microchip.com.

PINGUINO. Open Hardware Electronics Prototyping Platform Open Source Integrated Development Environment (IDE). Acessado em 20 de Novembro de 2012, www.pinguino.cc.

PYTHON. Acessado em 27 de Novenbro de 2012, http://www.python.org/.

REPRAP. Acessado em 22 de Novenbro de 2012, http://www.reprap.org/.

RIVIN, E. **Concepts of Programming Languages**. 1.ed. New York, NY, USA: McGraw-Hill Inc, 1988.

SACHS, E. Three-Dimensional Printing: rapid tooling and prototypes directly from a cad model. **Annals of the ClRP**, PARIS, FRANÇA, v.39, p.201–204, 1990.

UBUNTU. Acessado em 27 de Novenbro de 2012, http://www.ubuntu.com/.

USB. Acessado em 27 de Novenbro de 2012, http://www.usb.org/developers/packaging/.

ZIGBEE. Acessado em 27 de Novenbro de 2012, http://www.xbeestore.com.br/.

ÁLVAREZ, Y. N. C. Pynguino. Acessado em 30 de Novenbro de 2012, http://code.google.com/p/pinno-processing/wiki/Pynguino.