Universidade Federal do Estado do Rio de Janeiro
Centro de Ciências Exatas e Tecnologia
Escola de Informática Aplicada

IMPRESSÃO 3D

Vinícius Martins Lino

Orientador
Prof. Leonardo Luiz Alencastro Rocha

Rio de Janeiro, RJ – Brasil
Janeiro de 2014

IMPRESSÃO 3D

Vinícius Martins Lino

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Prof. Leonardo Luiz Alencastro Rocha (UNIRIO)

_____

Profa. Morganna Carmem Diniz UNIRIO)

_____

Prof. Luiz Amâncio Machado de Sousa Junior (UNIRIO)

Rio de Janeiro, RJ – Brasil.

Agradecimentos

Janeiro de 2014

Aos meus pais, Vera Lúcia Gomes Martins e Francisco José Saraiva Lino, por todo apoio, suporte e confiança que sempre me deram e continuam oferecendo para que eu consiga vencer todos os obstáculos e por sempre estarem ao meu lado em todos os momentos da minha vida.

À toda minha família pelo incentivo, educação, paciência, amor e alegrias nos bons momentos, mas também pelo conforto em todos os momentos difíceis da minha vida.

Aos meus amigos em geral que sempre me proporcionam boas lembranças e entendem quando não posso comparecer aos compromissos por conta de eventuais provas e/ou trabalhos da universidade.

Aos meus amigos da universidade pelos momentos de descontração dentro e fora da UNIRIO, bem como nos momentos de aflição em épocas de provas e trabalhos.

Aos professores do corpo docente do BSI pelos conhecimentos adquiridos e, especialmente, ao meu orientador, o professor Leonardo Rocha, pela atenção e transmissão de conhecimentos.

À Igor Campos, por desenvolver a biblioteca CustomStepper, fundamental para o desenvolvimento do projeto.

Finalmente, mas não menos importante, à Larry Page e Sergey Brin, por desenvolverem o Google. Esse texto é tão meu quanto de vocês.

RESUMO

O presente trabalho propõe o desenvolvimento do projeto e início do desenvolvimento de uma impressora 3D funcional e barata, simples o bastante para ser montada em casa por alguém com um mínimo de conhecimento.

Este projeto, inicialmente pretende esclarecer todas as etapas de uma impressão 3D, desde o desenvolvimento do modelo virtual, passando por sua conversão em formato STL, sua conversão em G-code, a implementação de um protótipo capaz de imprimir em duas dimensões e, finalmente, o modelo físico impresso em 2D através da interpretação do G-code pelo controlador da impressora.

Palavras-chave: impressão 3D, STL, G-code, Controlador de impressora.

ABSTRACT

This paper proposes the inicial development of a 3D printer project from scratch in a way that it become an affordable and easy to assemble kit to anyone with minimal knowledge.

This project, inicially intends to explain every step of 3D printing, from the development of virtual model, covering its conversion to the STL extension, the STL conversion to G-code, the implementation of a prototype capable of printing in two dimensions and, finally, the physical 2D model printed trough the controller.

Keywords: 3D printing, STL, G-code, Host software.

Introdução

- **1.1 Motivação**

Nos últimos anos, a impressão 3D deixou de ser uma ideia fantasiosa para se tornar uma realidade em rápido desenvolvimento. Desde o surgimento dos primeiros protótipos, a atenção do mercado se voltou a essa nova tecnologia, que tem um vasto potencial de aplicação, seja produzindo peças para sua própria replicação (REPRAP.ORG, 2013), objetos pequenos e decorativos, peças de substituição, até a criação de próteses mecânicas de baixo custo (1). As possibilidades são vastas o bastante, de modo a justificar qualquer projeto que envolva o conceito.

A ideia de montar uma impressora 3D de modelo próprio surgiu como um projeto pessoal do professor Leonardo Rocha. Ao citar em sala de aula que tinha tal plano e também que pretendia ser orientador de alguém no começo de 2014, o professor despertou meu interesse.

Num primeiro momento o projeto não seria necessariamente uma impressora 3D, mas algo envolvendo o uso de Arduino. No entanto, como o mercado de impressoras 3D está em expansão e existem diversos projetos open-source, o desenvolvimento de uma variante nacional parece uma ideia boa para diminuir custos da aquisição de modelos e tornar algo acessível a todos. Todavia, um projeto desse tamanho parece inviável de ser desenvolvido sem uma equipe em tão pouco tempo e, portanto, este trabalho fica como passo inicial no processo.

- **1.2 Objetivos**

O objetivo principal deste trabalho é de esclarecer de maneira suscinta o funcionamento e as etapas de uma impressão 3D, bem como as tecnologias envolvidas, suas complicações e facilidades, dar início ao desenvolvimento de um modelo próprio e definir um TO-DO do que deverá ser implementado futuramente.

- **1.3 Organização do texto**

Este trabalho está dividido em cinco capítulos, incluindo esta introdução. O segundo e terceiro capítulos descrevem a solução proposta, incluindo as tecnologias envolvidas.
O capítulo 4 aborda o projeto técnico da solução, os problemas encontrados e as tecnologias utilizadas, bem como as integrações entre elas e a arquitetura da solução.
O quinto capítulo é o de conclusão, citando as contribuições realizadas, as limitações do presente projeto e os espaços para trabalhos futuros.

2 Modelagem 3D – Do conceito à realização

Uma impressora 3D envolve diversos conceitos que serão abordados nesse capítulo. A seção a seguir descreve brevemente o processo por trás da impressão, desde o desenvolvimento do modelo virtual até a impressão em si.

- **2.1 Tecnologias**

  - **2.1.1 Modelagem 3D**

    É basicamente a utilização de formas geométricas simples. São feitas através de softwares especializados. Alguns exemplos são: SketchUp, 3Ds Mas, Blender, Cinema 4D, Maya, ZBrush, OpenSCAD, entre outros. Existem padrões de mercado utilizados que fazem com que um arquivo (CAD – Design Auxiliado por Computador) criado por uma ferramenta possa ser lido e alterado em outra. Os modelos são compostos de polígonos, vértices e arestas que geram uma malha complexa em 3D e acabam por gerar a forma o mais precisa possível do objeto desejado.

  - **2.1.2 Computer Numerical Control (CNC)**

    Controle numérico (**NC**) é a automação de ferramentas que são operadas por comandos precisamente programados em um controlador sem intervenção manual direta de um operador. Grande parte dos comandos numéricos hoje em dia é feita por computadores (**CNC**), na qual a máquina é inteiramente automatizada. (WIKIPEDIA, 2014)

    Em sistemas CNC modernos, a geração produto final é altamente automatizada utilizando programas de Design Auxiliado por Computador (CAD).

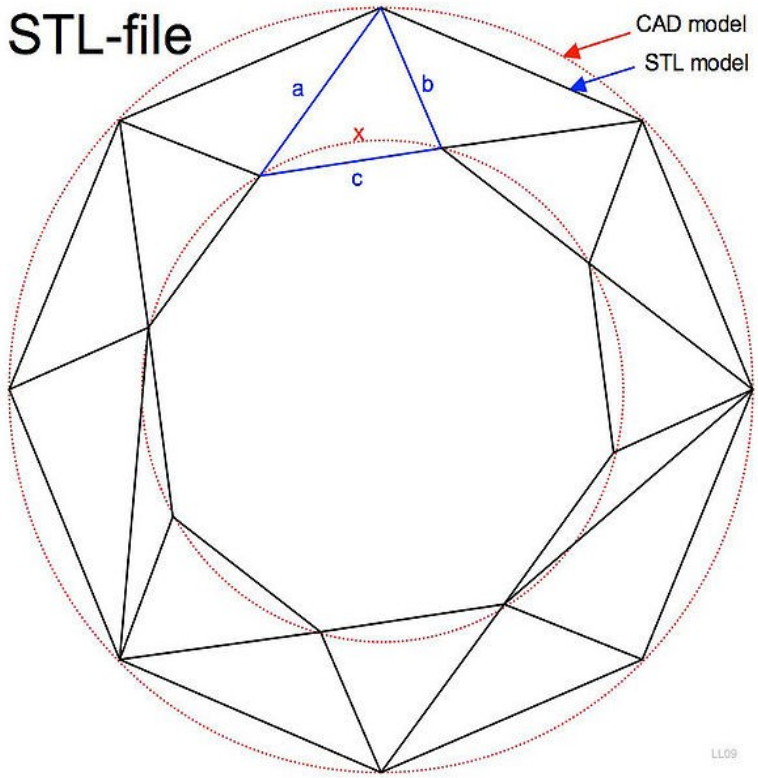
  - **2.1.3 Ferramenta CAM (Computer Aided Manufacturing)**

    São softwares que auxiliam na conversão de arquivos CAD para um formato(STL) mais bem recebido por máquinas CNC, em especial pela impressora 3D. Como foi descoberto ao longo do projeto, a maior parte das ferramentas que gera um modelo 3D já tem funcionalidades para exportar um arquivo no formato STL,

eliminando a necessidade de ferramentas externas.

- **2.1.4 Formato STL**

O STL (StereoLithography (2)) é um formato de arquivos criado pela empresa 3D Systems. O formato é suportado por vários softwares diferentes e é muito usado para prototipação rápida e para manufaturação assistida por computador. Ao contrário de um formato de modelo 3D padrão, o STL só descreve a superfície geométrica do objeto, sem se preocupar com cor, textura ou outros atributos. A superfície é convertida em uma malha de triângulos, de modo a simplificar formas mais complexas como um círculo (figura 1) ou até mesmo um cubo . Dessa maneira, o objeto fica significativamente mais simples e reduz substancialmente o tempo e custo de impressão.



(Figura 1 - Comparação de um modelo CAD padrão e um modelo STL)

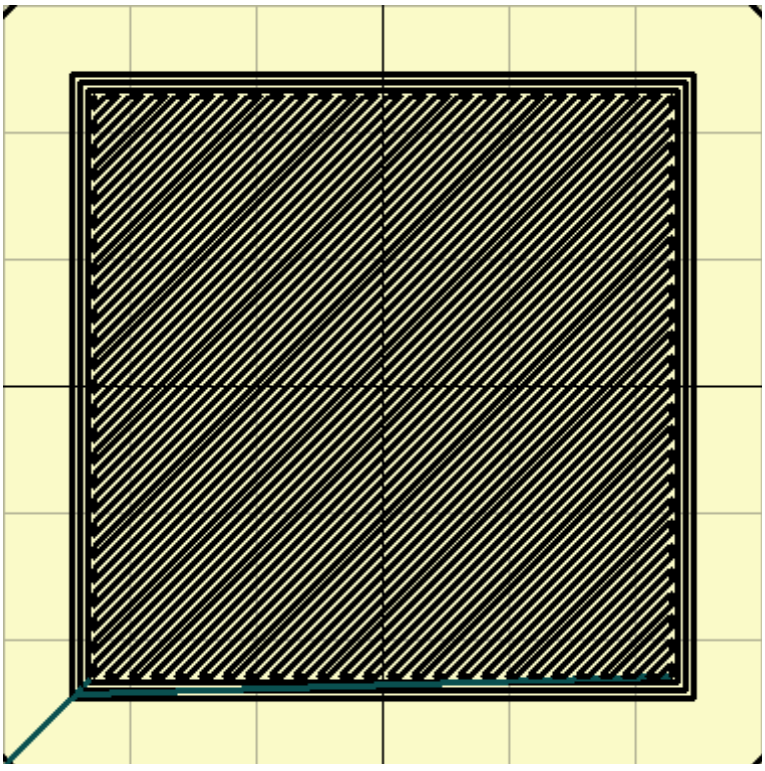
A figura 1 mostra a diferença entre o modelo CAD e um STL, em que o STL trabalha as formas sempre em triângulos, aproximando ao máximo do original CAD.

- **2.1.5 Fatiador (Slicing Software)**

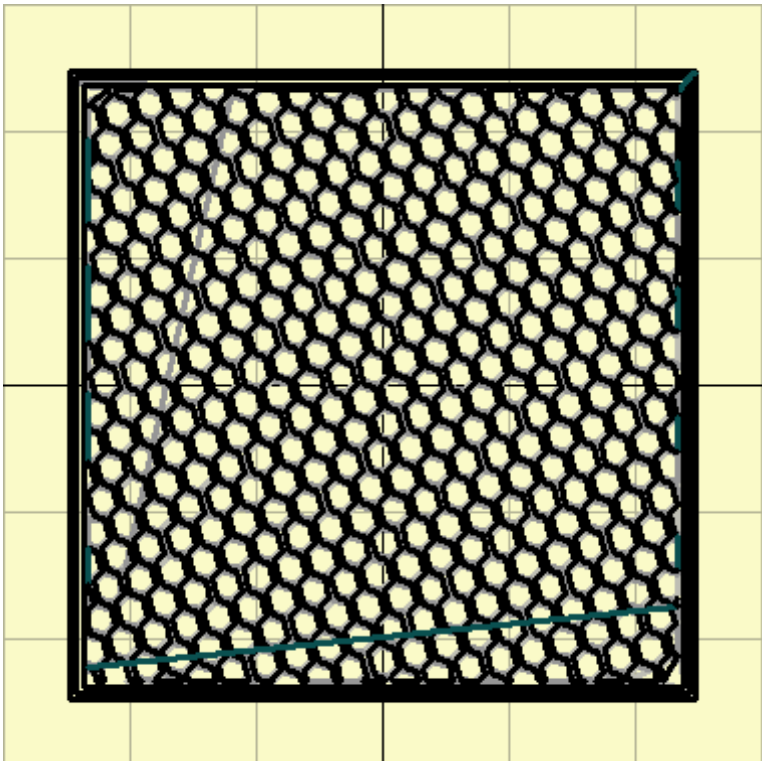
É o software responsável por converter o arquivo STL em G-Code. O programa divide a malha do modelo 3D em fatias, depois analisa as interseções entre cada fatia e usa a informação pra determinar o deslocamento que o extrusor (parte mecânica responsável por aquecer e guiar o depósito de material) da impressora terá de realizar, onde depositar o material, e calcula quanto do filamento será necessário para a distância corrida. O fatiador também preenche o modelo, que originalmente é oco, com uma estrutura não maciça, de modo que ele não colapse sob seu próprio peso. A estrutura mais comum encontrada nos estudos foi a de favo de mel (honeycomb).

- **2.1.6 Interpretador**

É o software responsável por ler o G-Code. Ele pega o código gerado pelo Fatiador, lê linha por linha e diz ao controlador o que fazer com cada uma. A forma como isso é feito não ficou clara dentro do período de conclusão desse texto. O Interpretador serve também como interface entre o usuário e a impressora, oferecendo ferramentas como um visualizador com função de exibir camada por camada da futura impressão baseado na leitura do código, controle de temperatura e dimensões da impressora e edição manual do G-Code.



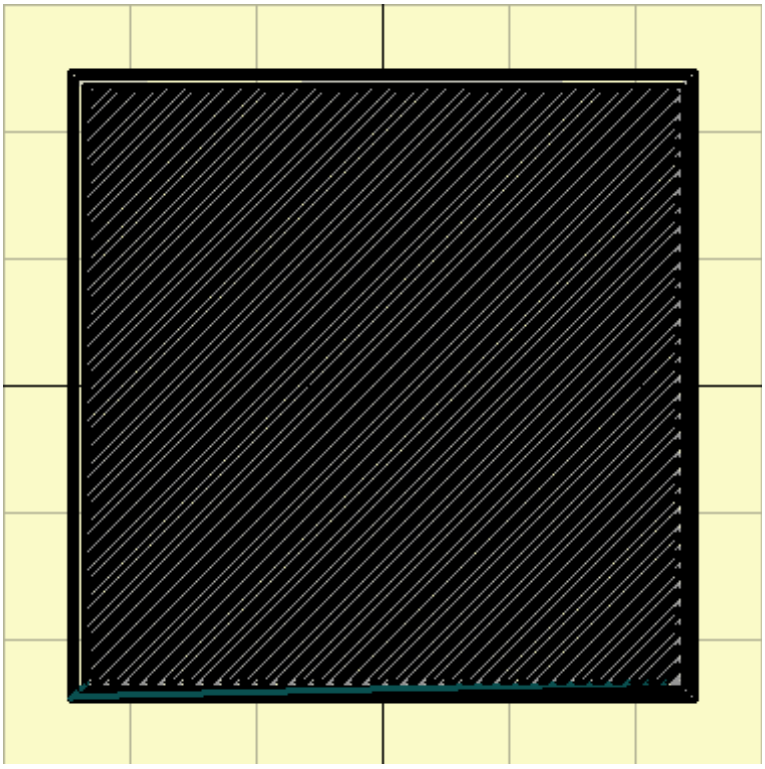
(Figura 2 - Primeira camada impressa do cubo)



A figura 2 mostra como fica a primeira camada da impressão de um cubo, vista de cima. A impressora desenha o contorno e o preenchimento sem nunca deixar de depositar material.

(Figura 3 - Camada intermediária a ser impressa)

A figura 3 exemplifica bem o conceito de estrutura honeycomb ao mostrar uma camada intermediária da impressão.



(Figura 4 - Camada final a ser impressa)

A figura 4, bastante semelhante à 2, demonstra a última camada impressa que deixa de fazer o padrão de honeycomb e preenche inteiramente o lado do cubo.

- **2.1.7 Controlador**

A parte eletrônica responsável pela execução e implementação das diretrizes dadas pelo Interpretador. O Controlador utiliza um firmware para transformar as diretrizes que o Interpretador envia a ele e realizar enfim a impressão. Basicamente, ele possui um interpretador G-code incorporado. No projeto proposto, o controlador utilizado será um Arduino UNO, mas existem diversos modelos no mercado capazes de exercer a função.

- **2.1.8 RepRap**

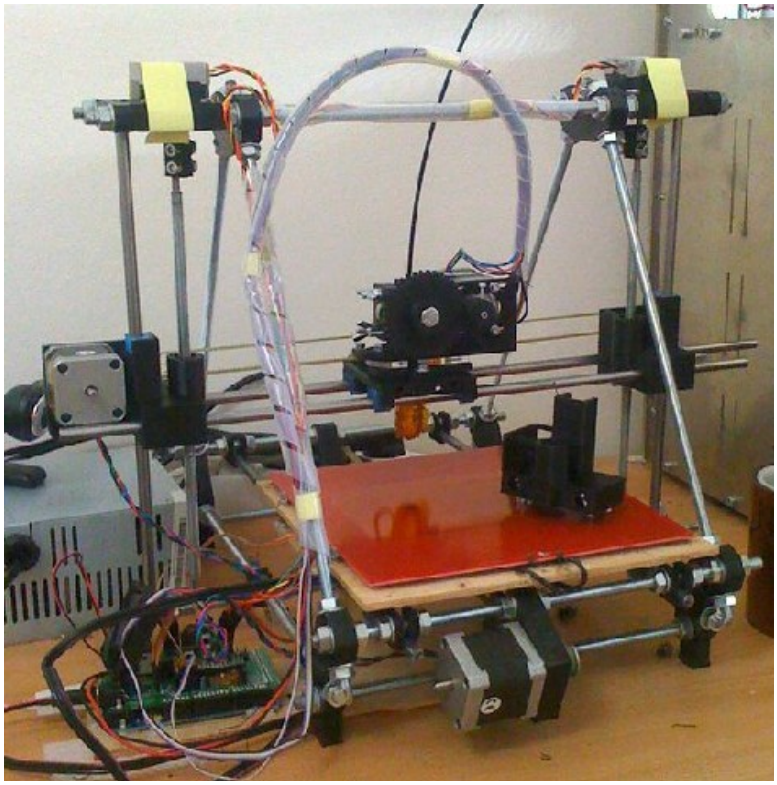
"RepRap é o primeiro projeto de ferramenta de manufatura auto replicável da história da humanidade.". (3)

RedRap é um projeto open source de impressora 3D capaz de imprimir objetos plásticos. Como muitas das partes da RepRap são plásticas e ela imprime essas partes, a RepRap se replica ao fazer um kit dela mesma – um kit que qualquer um pode montar tendo tempo e material para tal. Isso também significa que se você tem uma RepRap, pode realizar o reparo da sua ou até mesmo imprimir uma nova para alguém sem grandes problemas.

O objetivo da RepRap é justamente o de criar máquinas auto-replicantes, tornando-as fácilmente adquiríveis para beneficiar a todos.

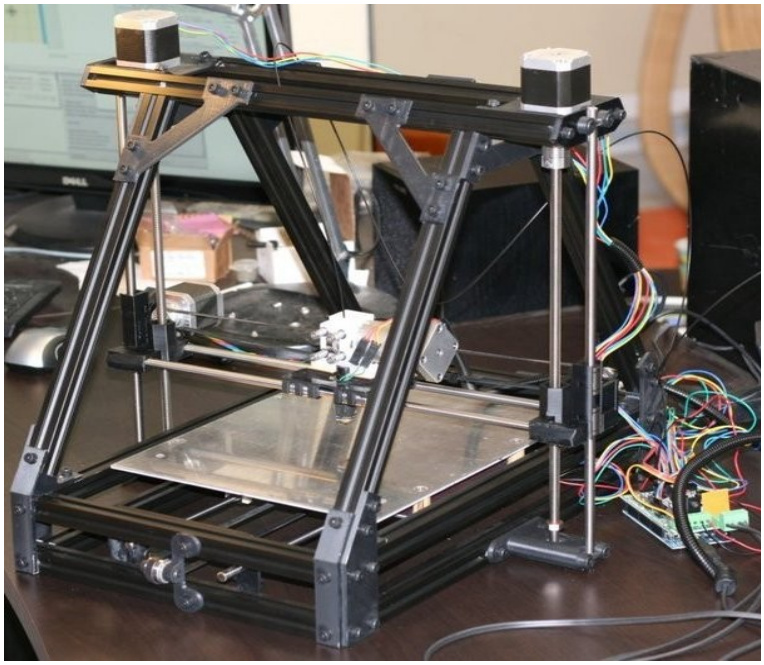
Existem diversos modelos da plataforma, cada um desenvolvido para abordar uma necessidade diferente. Abaixo há uma lista com alguns dos modelos gratuitos mais comuns e suas motivações:

Prusa Mendel - RepRap completa, focada em baixo custo e facilidade de abastecimento.



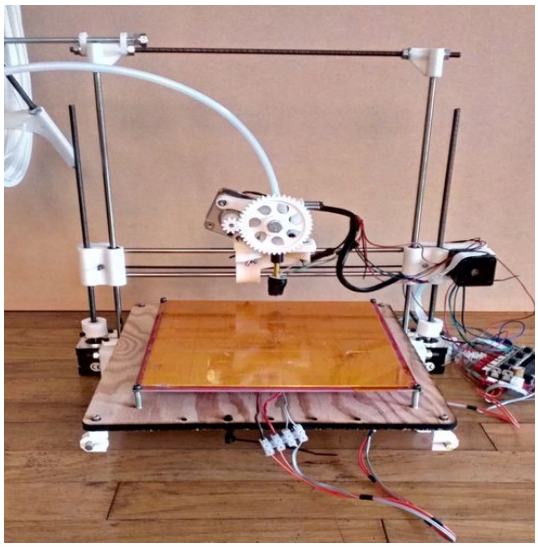
(Figura 5 – Prusa Mendel)

MendelMax - RepRap completa, focada em robustez estrutural e facilidade de montagem.



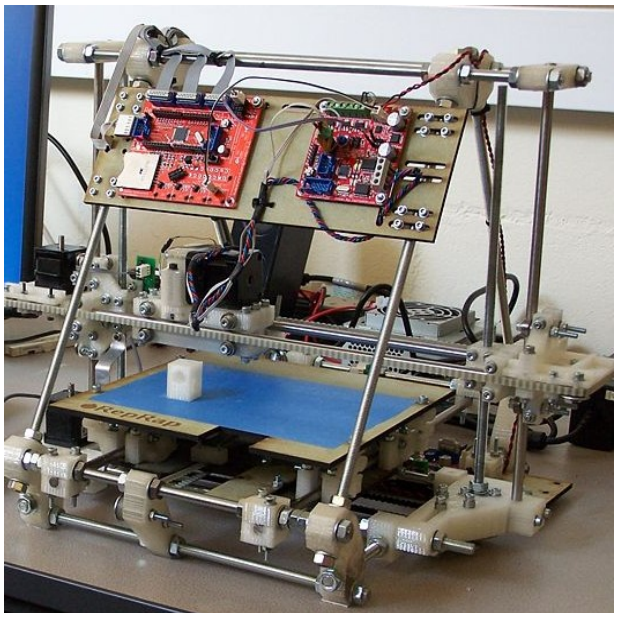
(Figura 6 – MendelMax)

Wallace - RepRap compacta desenhada para reduzir a quantidade de partes e minimizar a complexidade da montagem.



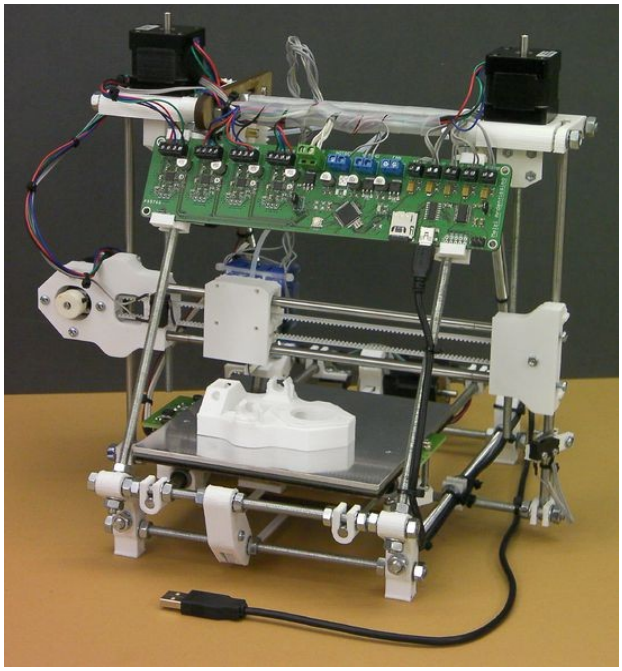
(Figura 7 – Wallace)

Mendel Original - O primeiro desenho da Mendel: completamente funcional, mas mais complicada que variantes posteriores.



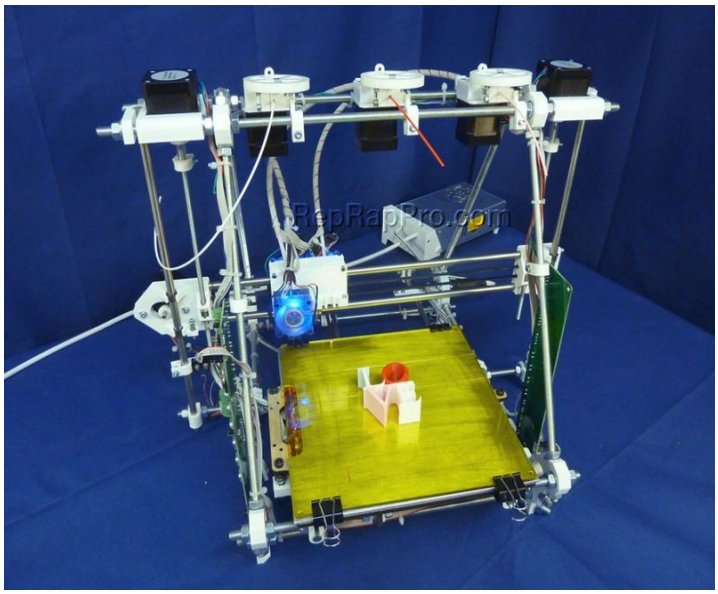
(Figura 8 – Mendel Original)

RepRapPro Huxley - RepRap compacta, desenhada para reduzir a quantidade de peças e adicionar portabilidade eu projeto.



(Figura 9 – RepRapPro Huxley)

ReepRap Tricolour Mendel – Desenvolvida para imprimir com mais de 1 material e cores



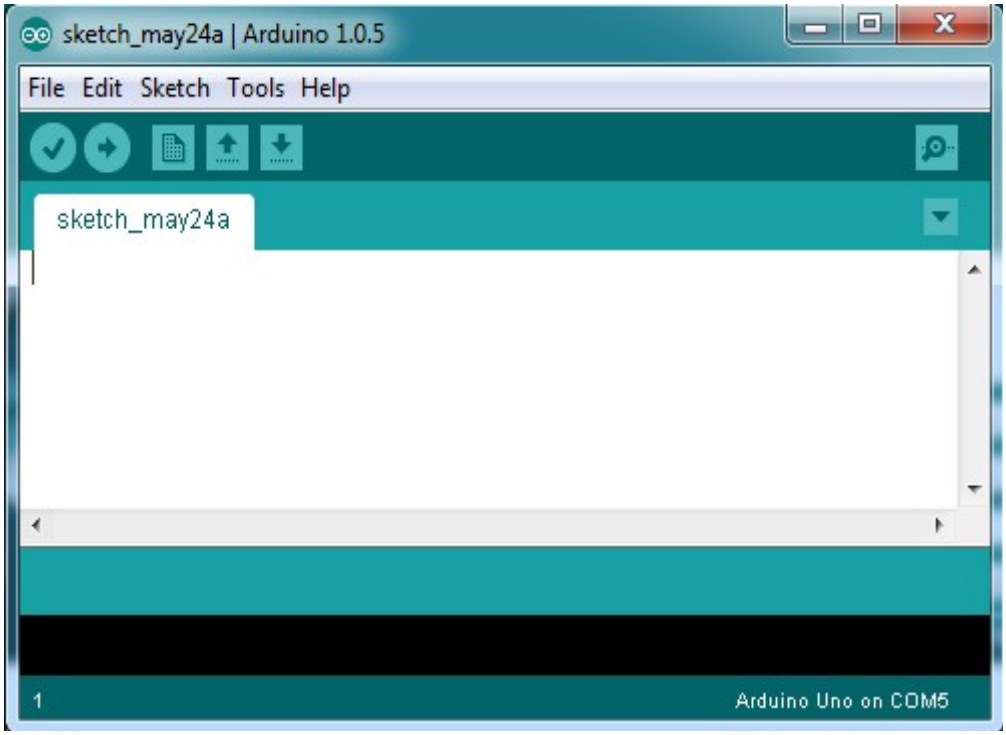
(Figura 10 – RepRap Tricolour Mendel)

3 Tecnologias Utilizadas

- **3.1 Arduino**

"Arduino é uma plataforma de prototipação 'open-source' baseada em hardwares e softwares fáceis de usar e flexíveis. É direcionada a artistas, designers, hobbistas e qualquer um interessado em objetos ou ambientes interativos." (ref3)

O kit básico utilizado nesse projeto é o UNO, composto de um microcontrolador com saídas para 14 pinos digitais, 6 analógicos, conexão usb, botão de reset, alimentado por 5 volts, uma protoboard e alguns componentes simples e fáceis de repor, como potenciômetros, LEDs, sensores de iluminação, botões, cabos para fazer a fiação, entre outros. A plataforma contém também uma IDE própria para desenvolvimento e envio dos códigos da máquina para o controlador baseada em c++ que utiliza uma porta usb na máquina para se comunicar por uma entrada serial no controlador.



(Figura 11 – Kit Arduino UNO)

O site oficial da plataforma possui uma comunidade de desenvolvedores na forma de um fórum organizado por áreas de interesse e/ou região onde projetos são discutidos e soluções a diversos problemas são apresentadas. Diversas bibliotecas para necessidades específicas e recorrentes são disponibilizadas gratuitamente por lá.

O site possui ainda uma área dedicada a ensinar o básico da plataforma, o Arduino Wiki(Playground) que é composta de uma coleção de tutoriais e instruções criadas pelos próprios usuários do Arduino.

(Figura 12 – Kit Arduino UNO)

- **3.2 G-Code**

Dentro dos casos estudados, o G-code foi identificado como a principal linguagem de programação voltada para controle numérico (CNC). Basicamente é onde são definidas as coordenadas para que as máquinas saibam o que fazer, fisicamente.

Máquinas de cortar, montadoras e impressoras 3D são algumas das máquinas que utilizam G-code para guiar seus movimentos.

O código é estruturado de forma que cada linha possua alguns poucos comandos que são executados em sequência, cada comando é composto de um identificador do tipo de comando, seguido do identificador do comando específico.

A tabela abaixo explica alguns tipos de comando que os modelos RepRap utilizam. Os números nos campos estão representados por nnn. Números podem ser inteiros, ou um decimal, dependendo do contexto. Por exemplo a coordenada X pode ser um inteiro (X175) ou um double (X17.62), enquanto selecionar o extrusor 2,76

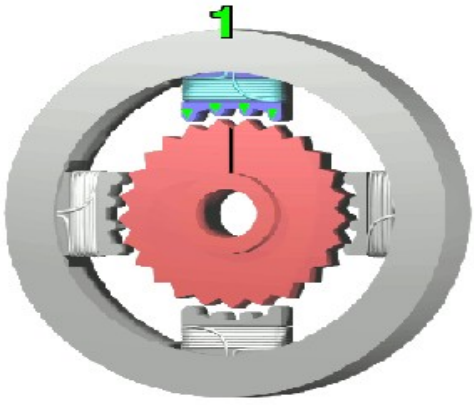
não faria sentido.

Letra	Significado
Gnnn	Comando G-Code padrão, como mover até determinado ponto.
Tnnn	Seleciona ferramenta nnn. No caso, o extrusor.
Xnnn	Uma coordenada X, geralmente de destino.
Ynnn	Uma coordenada Y, geralmente de destino.
Znnn	Uma coordenada Z, geralmente de destino.
Fnnn	Taxa de alimentação em mm por minuto.
Ennn	Comprimento do filamento a ser extrudado.
Nnnn	Número da linha. Usado para requisitar repetição de transmissão em caso de erro de comunicação.
*nnn	Checksum. Usado para checar erros de comunicação.

Tab.1 - Fonte: http://reprap.org/wiki/G_code

- **3.3 Motor de passo**

Um motor de passo é um motor elétrico de precisão. Ele tem a rotação controlada por uma série de campos eletromagnéticos que são ativados e desativados através de impulsos eletrônicos. Eles possuem um número fixo de polos magnéticos que determinam o número de passos por revolução. Os motores de passo mais comuns possuem de 3 a 72 passos/revolução, significando que são necessários entre 3 e 72 passos para completar uma volta.



(Figura 13 – Motor de passo com 1 polo ativo e 3 inativos)

Controladores mais avançados de motores de passo utilizam modulação por largura de pulso pra realizar micropassos, aumentando significativamente a quantidade de passos por revolução e sacrificando um pouco de velocidade.

O controle computadorizado de motores de passo é uma das formas mais versáteis de sistemas de posicionamento. (4)

- **3.3.1 Modelo utilizado**

O motor escolhido para o projeto foi o modelo 28BYJ-48. Modelo que possui uma biblioteca específica dedicada do Arduino.

Especificações:

Fases	4
Ângulo do passo	5.625°
Distância do passo	NULL
Perímetro do Motor	28mm
Resistência da fase	300Ω
Torque	29.4mN.m
Inércia do rotor	NULL
Cabos	5
Peso	50.00g
Modelo	IM120723014

(Figura 14 – Motor de passo 28BYJ-48)

Como o projeto trata de algo que precisa de precisão milimétrica ou maior, precisávamos de um motor que trabalhasse com micro passos (meio passo) e graças a um mecanismo interno, o 28BYJ-48 atinge aproximadamente 4075(quatro mil e setenta e cinco) saltos de posição dentro de 1 volta completa do motor, o que o deixa com a precisão necessária para a impressora.

Existem 3 padrões de comando que podem ser enviados ao motor, cada um com uma característica específica:

com1	com2	com3
0001	0011	0001
0010	0110	0011
0100	1100	0010
1000	1001	0110
		0100
		1100
		1000
		1001

O com1 é rápido mas menos potente, o com2 é rápido e potente, o com3 é 2 vezes mais lento que os outros, mas tão potente quando o 2 e mais preciso que os

anteriores.

4 Implementação

Em um primeiro momento, foi necessário estudar o passo a passo das etapas a que o objeto a ser impresso era submetido para entender a fundo o funcionamento de uma impresora 3D de modo que o projeto não se tornasse só mais uma implementação de um modelo já existente no mercado. Então, o trabalho foi dividido em partes, que contemplam:

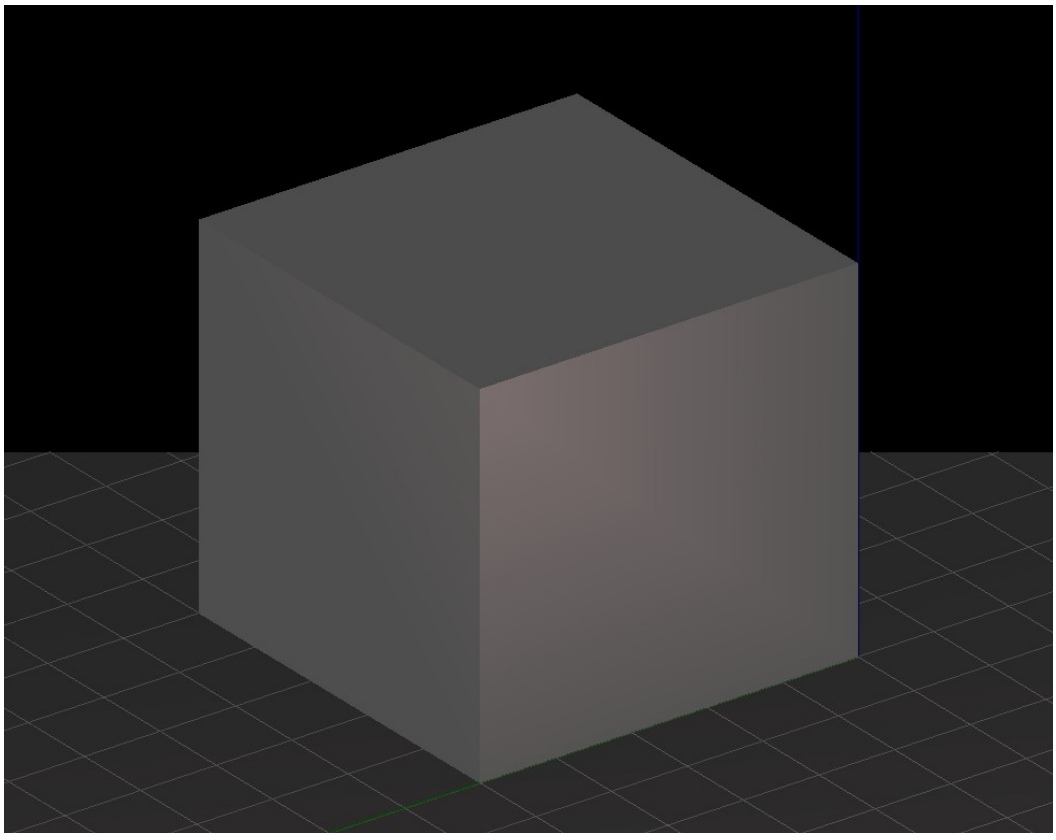
1. Desenvolvimento de um modelo 3D básico (CAD).
2. Conversão do modelo 3D em um formato de fácil conversão para código (STL).
3. Conversão do arquivo STL para G-code.
4. Estudo do Gcode.
5. Desenvolvimento do firmware do controlador.
6. Desenvolvimento do modelo físico de 1 esteira (eixo X).
7. Desenvolvimento do restante do modelo físico.

O projeto presente pretende chegar ao ponto do desenvolvimento em que uma esteira para o eixo X e uma para o Y estejam funcionais, de modo que uma impressão em 2D seja viável.

ETAPAS

1. Desenvolvimento de um modelo 3D básico (CAD).

Um modelo 3D CAD (Computer Aided Design) é um formato de arquivo que usa polígonos para descrever figuras geométricas e é a base para a impressão 3D. Dessa forma, o primeiro passo em entender o processo de impressão, seria desenvolver um modelo 3D simples. O caso escolhido a ser usado como base de estudo foi um cubo de dimensões 50x50x50 mm.



(Figura 15 – Modelo 3D gerado em OpenSCAD)

Para tal, foi usado o programa OpenSCAD, um software open source que usa código para definir as dimensões e forma de seus modelos. Existem diversos programas que trabalham de maneira mais manual e menos lógica, de maneira que formas mais avançadas podem ser feitas mais facilmente e com menos cálculos, mas para o objeto de estudo proposto, o software utilizado foi suficiente.

O código gerador do modelo acima é simples e não toma mais que uma linha:

- cube(size = 50, center = false);

O programa disponibiliza uma série de dados do modelo programado, tamanho dos lados, quantidade de vértices, arestas, entre outros que podem vir a ser relevantes em projetos mais complexos.

Feito isso, o passo seguinte é exportar o arquivo no formato desejado(STL) para a impressão de modo que a malha não seja complexa demais e não tome exponencialmente mais tempo e recursos do que deveria sendo reproduzida.

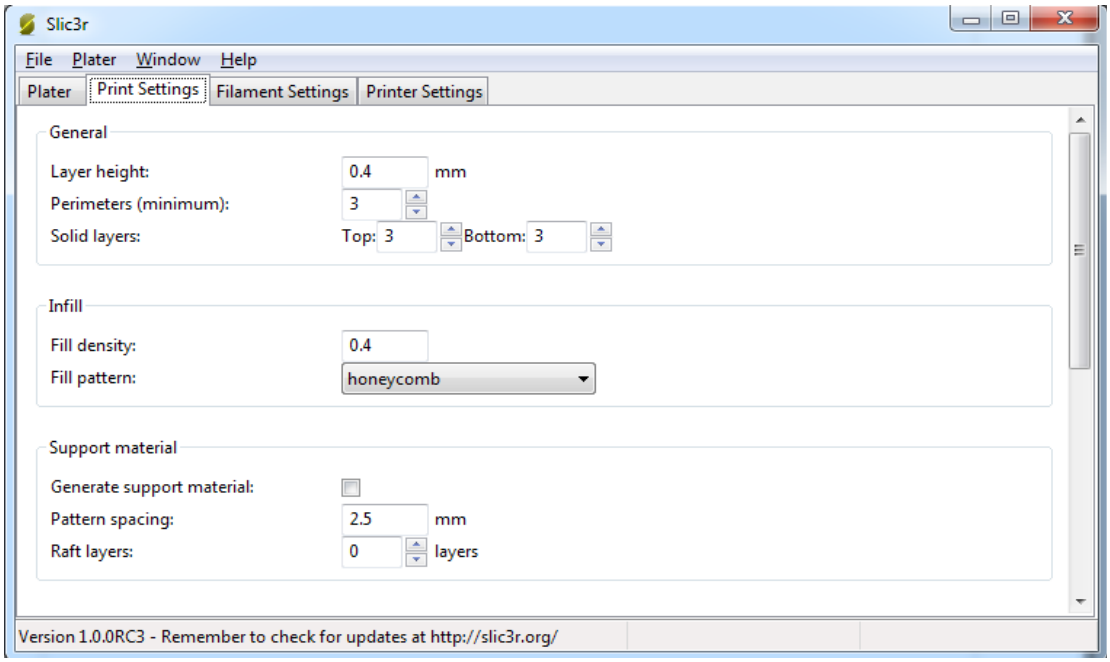
2. Conversão do modelo 3D em um formato(STL) de fácil conversão para código .

Felizmente, a maioria dos softwares de modelagem 3D já suportam o formato STL e conseguem gerar um arquivo direto, sem precisar salvar em seu formato nativo antes, eliminando assim o trabalho do ítem **2** de converter o arquivo com alguma ferramenta externa.

3. Conversão do arquivo STL para G-code.

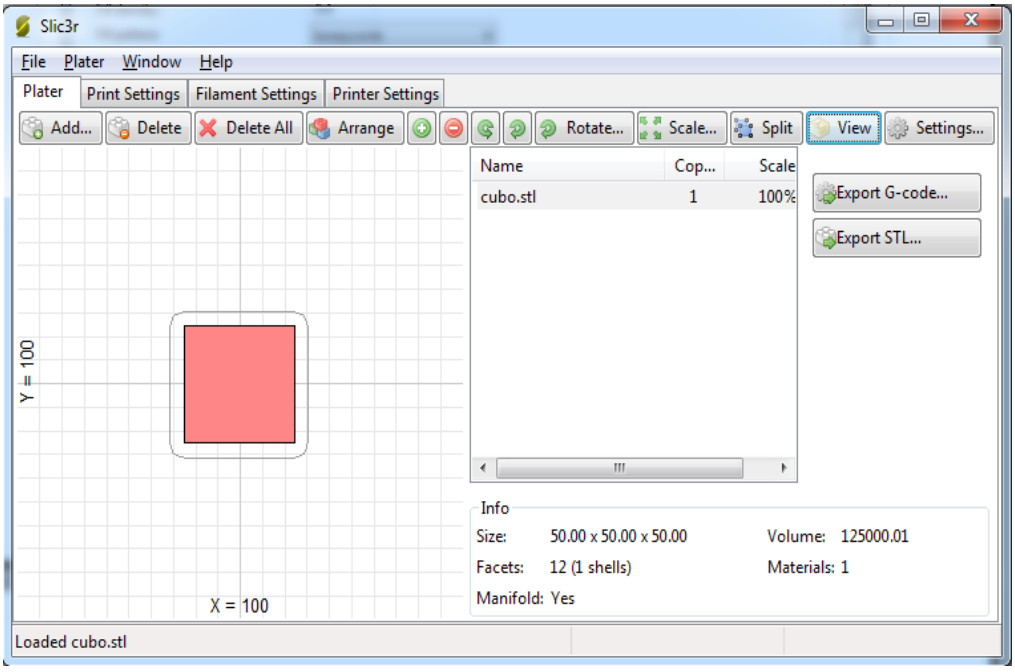
Com o arquivo STL em mãos, o próximo passo era entender o que era e como funcionava o G-code, e como a conversão de um modelo 3D para um conjunto de coordenadas era feito.

O software utilizado nessa solução foi o Slic3r. Um programa da categoria Fatiador (explicada no capítulo anterior), que pede inputs como o perímetro do filamento, a altura de cada camada impressa, a densidade do filamento e alguns outros dados que serão usados na geração do G-code.



(Figura 16 – Exemplo de tela de configuração da impressão no Slic3r)

Ao abrir o arquivo STL no Slic3r, ele calcula os dados do modelo e se prepara para gerar o código de acordo com as preferências definidas na interface, gera uma visualização lateral do modelo e dá opções como vizualizar o modelo em 3D, escalar para um tamanho apropriado, desenvolver uma estrutura (em forma de colméia) interna não maciça de modo a reduzir custos de impressão (tempo/material) sem comprometer a sustentação do modelo ou a parte que nos interessa no experimento, exportar o G-code.



(Figura 17 – Exemplo de tela de configuração da impressão no Slic3r)

4. Estudo do G-Code.

Para explicar melhor a estrutura do G-Code, será exposto abaixo um breve pedaço de código comentado:

; layer_height = 0.4 ; perimeters = 3 ; top_solid_layers = 3 ; bottom_solid_layers = 3 ; fill_density = 0.4 ; perimeter_speed = 30 ; infill_speed = 60 ; travel_speed = 130 ; nozzle_diameter = 0.5 ; filament_diameter = 3 ; extrusion_multiplier = 1 ; perimeters extrusion width = 0.50mm ; infill extrusion width = 0.53mm ; solid infill extrusion width = 0.53mm ; top infill extrusion width = 0.53mm ; first layer extrusion width = 1.00mm	O caractere “;” indica o começo de uma linha comentada. As linhas inteiramente comentadas são documentação da configuração usada pelo Slic3r para gerar o G-code.
G21 ;	Define que as unidades serão tratadas em milímetros.
M107 ;	Algum parâmetro específico.
M104 S205 ;	Define a temperatura.
G28 ;	Ordena que todos os eixos sejam setados em sua posição inicial.
G1 Z5 F5000 ;	Levanta a caneta e define a taxa de alimentação em mm/min.
M109 S205 ;	Espera a temperatura desejada ser atingida.
G90 ;	Determina que as coordenadas serão absolutas.
G1 F1800.000 ;	Altera a taxa de alimentação.
G1 Z0.500 F7800.000 ;	Define que a próxima altura do eixo Z é 0.5 mm e a taxa de alimentação ;do extrusor aumente gradativamente de 1800 para 7800 mm/min.

G1 X70.130 Y70.130 F7800.000 ;	Define próximas posições dos eixos X e Y e mantém o a alimentação em 7800 mm/min.
G1 X71.170 Y69.270 F540.000 ;	Define próximas posições dos eixos X e Y e cai a taxa de alimentação de 7800 para 540 gradativamente.
G1 X72.360 Y68.630 ;	Define próximas posições dos eixos X e Y
G1 X73.660 Y68.240 ;	

A parte relevante inicialmente para o estudo é a em negrito pois é a que trata da movimentação dos eixos, o que foi a primeira questão escolhida a ser tratada no desenvolvimento do firmware e do hardware da solução. Foram identificados alguns padrões que, nesse projeto, foram tomadas como convenções do código.

- Como pode-se ver, o código funciona com precião em centézimos de milímetros e é executado linha por linha, só passando ao comando seguinte quando as diretrizes da linha anterior tiverem sido concluídas.
- Comandos diferentes numa mesma linha são executados em paralelo, ou seja, eles escalonam o tempo de execução de maneira a progredirem juntos do começo ao fim.
- Se existir a necessidade de se mover um eixo independentemente dos outros, seu comando precisa estar sozinho em uma linha.
- Um mesmo eixo não recebe duas ordens na mesma linha de comando. Dessa forma, não há possibilidade de um mesmo eixo tentar ir para 2 posições diferentes ao mesmo tempo.
- A ordem seguinte só será executada quando todos os eixos terminarem de realizar sua movimentação, de modo que mesmo que o eixo X tenha chegado a seu destino, por exemplo, ele não poderá ir ao ponto seguinte antes que o Y e Z tenham terminado de se movimentar, caso haja algo a ser realizado por eles.

5. Desenvolvimento do firmware do controlador.

Como dito no tópico anterior, o primeiro ponto atacado pelo firmware foi a movimentação dos eixos. Para tal, foi necessário montar uma esteira com uma rosca de 8 mm, um motor de passo 28byj-48, 2 rolamentos, 1 controlador Arduino UNO e a interface do motor com o controlador. Está sendo utilizada a plataforma nativa do Arduino e o editor de texto Notepad++ no desenvolvimento do código, assim como uma biblioteca customizável (encontrada nos foruns do próprio Arduino e devidamente creditada na seção

de agradecimentos) feita para o modelo específico do motor utilizado.

Inicialmente havia a necessidade de determinar a precisão do motor, sua proporção de mm/giro ou mm/passo. Logo foi verificado que marcar o deslocamento em passos não seria ideal pois são mais de 4(quatro) mil passos para uma volta completa do motor, o que seria mais complexo do que o pretendido para a fórmula de deslocamento desejada. Já a relação de mm/giro se mostrou interessante, tendo em vista que 1 volta completa apresentou deslocamento de 1,25 mm, que fazendo poucas contas geraram um resultado de 0,8 giros/mm. Um valor muito mais interessante para nossos cálculos.

O próximo passo para o projeto foi definir como seria enviado o comando ao controlador, tendo em vista que não seria usado um interpretador existente e que não foi descoberto ainda como ele envia o código ao controlador. Foi decidido então que um arquivo de texto com o G-code seria enviado como Input pela porta serial, e o próprio controlador leria e interpretaria o código, através do seu firmware.

Tendo recebido com sucesso o arquivo, o firmware inicialmente correrá o arquivo buscando os índices do array de bytes que contém X, Y ou Z pegando o conteúdo seguinte até o primeiro espaço em branco ou ';' (indicando o fim do comando e começo do seguinte) e armazenará o valor em uma variável que representa o destino do eixo em questão. Esse valor, como visto anteriormente, é a posição absoluta final do eixo que, quando subtraída da posição atual, vai determinar o deslocamento necessário a ser feito. Esse valor então vai ser jogado na fórmula:

R = 0,8*D*P;

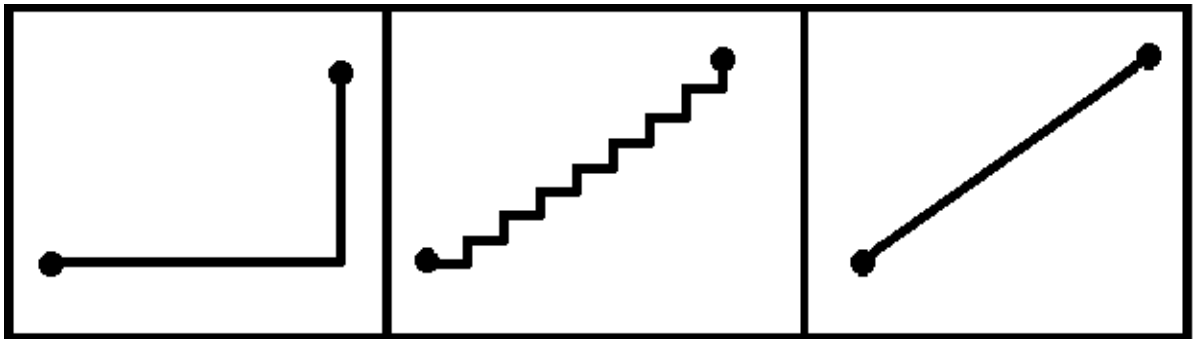
Onde D é o deslocamento em milímetros, P é a precisão desejada (no caso atual foi definida em 1 mm) e R a quantidade de rotações necessárias para tal.

Ex: se o eixo X se encontra na posição 170.000 e precisa ir à 193.800, D assume o valor de 193.8-170 = 23.8 mm. Ao jogarmos na fórmula, recebemos a quantidade de voltas do motor necessárias para atingir o ponto de destino = 19.04 giros.

Informação pontual: o motor gira (no momento) a uma velocidade máxima de 16 rpm devido a limitações da biblioteca utilizada.

No entanto, esse deslocamento não vai ser feito todo de uma vez. Como existe a possibilidade de X não ser o único eixo a sofrer alteração em uma linha de comando, temos que dividir esse resultado em partes menores e escalonar com os deslocamentos concorrentes para diminuir a propagação de erro, tendo em vista que se X correr todo seu trajeto antes de Y, ao invés de uma reta, ligando o ponto inicial do extrusor ao final, teríamos 2 retas formando um L, proveniente do fato de Y não ter sido alterado por toda a execução

do X e vice versa.



(Figura 18 – Execução consecutiva (zoom) VS execução concorrente (zoom) VS execução concorrente vista à distância)

O comando é quebrado levando em conta qual dos eixos a se moverem vai se deslocar mais, pega-se o maior deslocamento, divide pela precisão desejada (no caso em milímetros) e o resultado é arredondado pra cima e usado como o número de vezes que o comando mover 1 mm vai ser executado nessa linha do G-Code.

Exemplo:

x = 180.000 e y = 20.000

'x' e 'y' são as posições iniciais do eixo ao receber o comando de movimentação do G-code:

G1 X192.000 Y150.000

Temos então que os deslocamentos serão de 12 mm pro X e 130 mm pro Y.

Assim temos 130 (o maior) / 1 (precisão em mm) = 130 ciclos de movimentação pros 2 eixos.

A seguir é calculada a distância a ser percorrida por cada eixo em cada um desses 130 ciclos. No caso do eixo Y, temos 130/130 = 1 mm por ciclo. No caso do X, temos 20/130 mm por ciclo.

Jogando esses valores na fórmula de deslocamento, ficamos com:

D [X,Y] = {'20/130', '1'} mm

R[X,Y] = 0,8*D;

R[X] = 0,8 * 20/130 = 0,12 voltas;

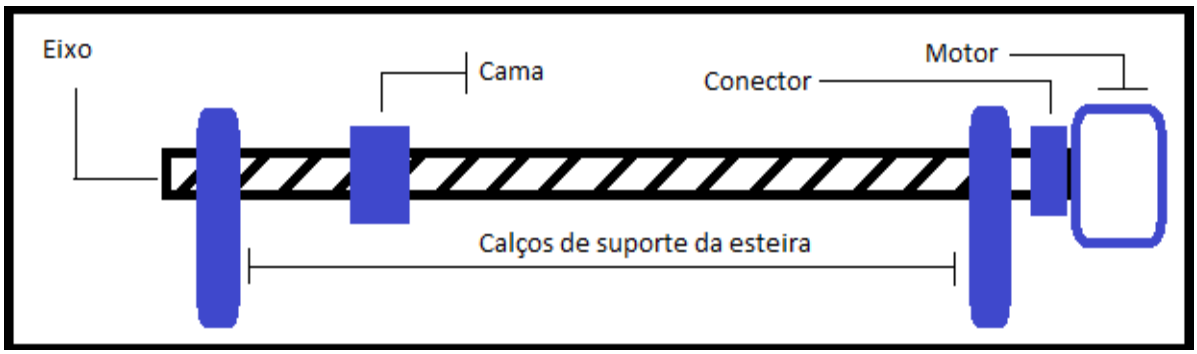
R[Y] = 0,8 * 1 = 0,8 voltas;

Assim, temos a necesidade de dar 0,12 rotações do eixo X e 0,8 rorações do Y 130 vezes para atingir os deslocamentos de 12 e 130 mm dessa linha de comando.

Resolvida essa problemática, o passo seguinte passa a ser o controle de temperatura do extrusor, seguido da alimentação.

6. Desenvolvimento do modelo físico de 1 esteira (eixo X).

Para os testes iniciais, foi desenvolvido um protótipo de esteira composto por 1 motor de passo 28byj-48, 1 controlador Arduino UNO, 1 interface entre o motor e o controlador, uma rosca de 8 mm, 2 rolamentos, 1 suporte para o motor, 2 calços para os rolamentos e 1 conector flexível para o eixo do motor e a rosca, 2 porcas para medir o deslocamento.



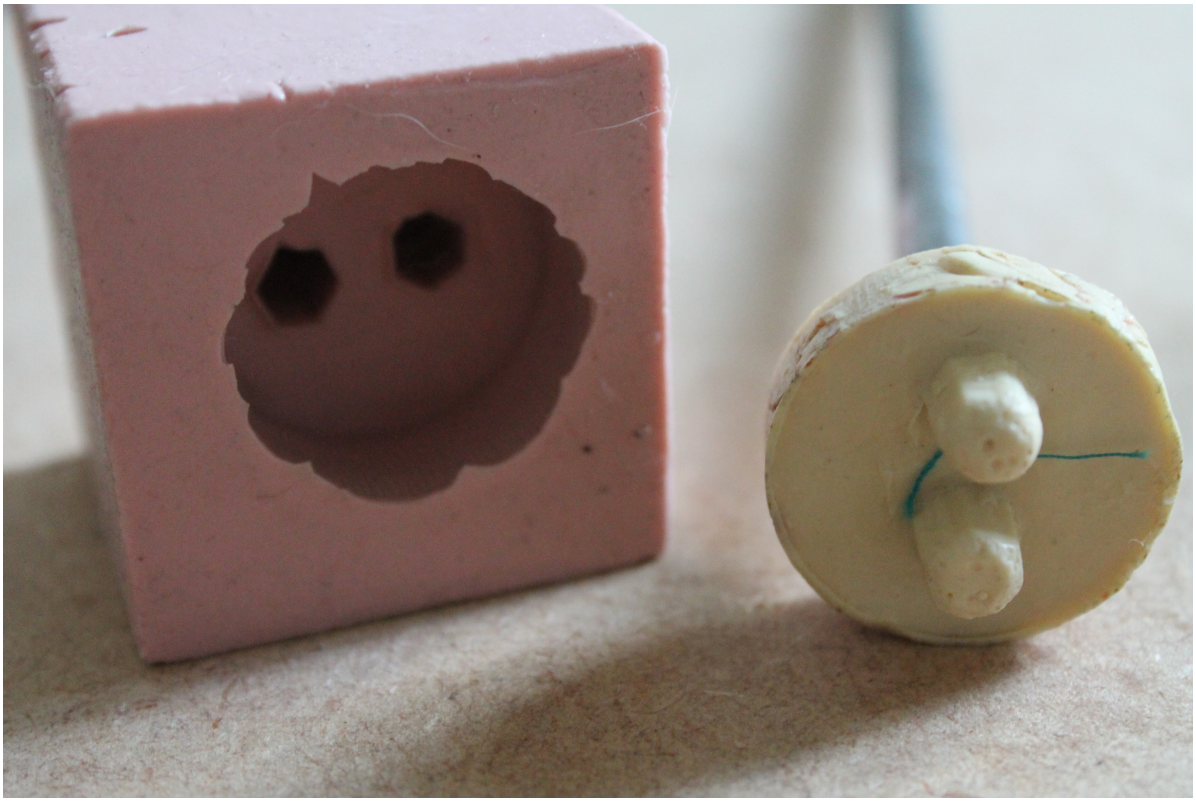
(Figura 10 – Esquema da esteira protótipo)

Com exceção do suporte e dos calços, tudo foi encomendado pronto. Estas peças foram feitas à mão, com massa de modelar, silicone e resina. A seguir, algumas fotos do processo de desenvolvimento tanto das peças quanto da esteira em si, incluíndo algumas tentatívas fracassadas de modelos.

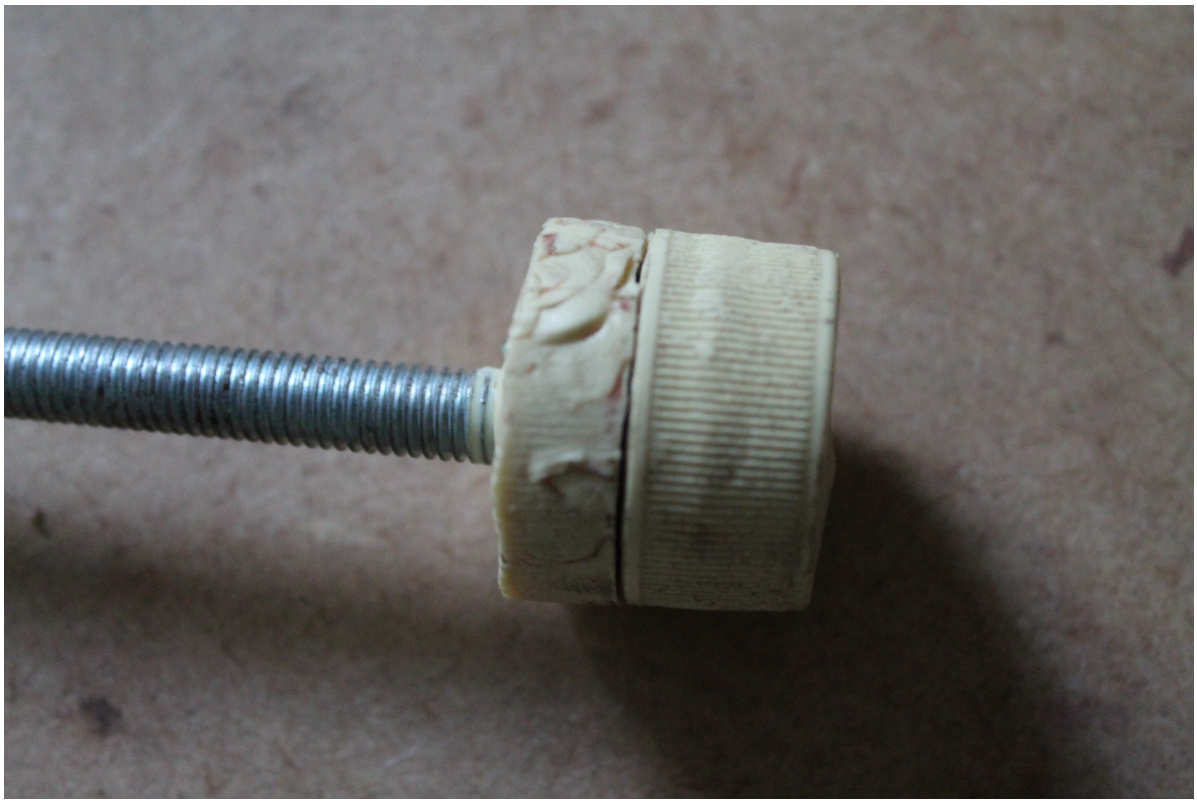
Num primeiro momento, a ideia era fazer o máximo de peças o possível manualmente, tendo em vista que um dos principais intuitos de uma impressora 3D open source é ser facilmente replicável e pouco custosa monetariamente.

Dessa forma, foi definido que os calços, suportes de motor e conectores seriam todos feitos em resina, material de fácil aquisição, resistente e rápido de extrair uma peça.

No entanto, após 2 tentativas diferentes de fazer o conector, concluiu-se que ele precisava de mais precisão do que era possível conseguir sem equipamentos próprios para sua manufatura, então os protótipos foram deixados de lado e substituídos por conectores flexíveis encomendados.

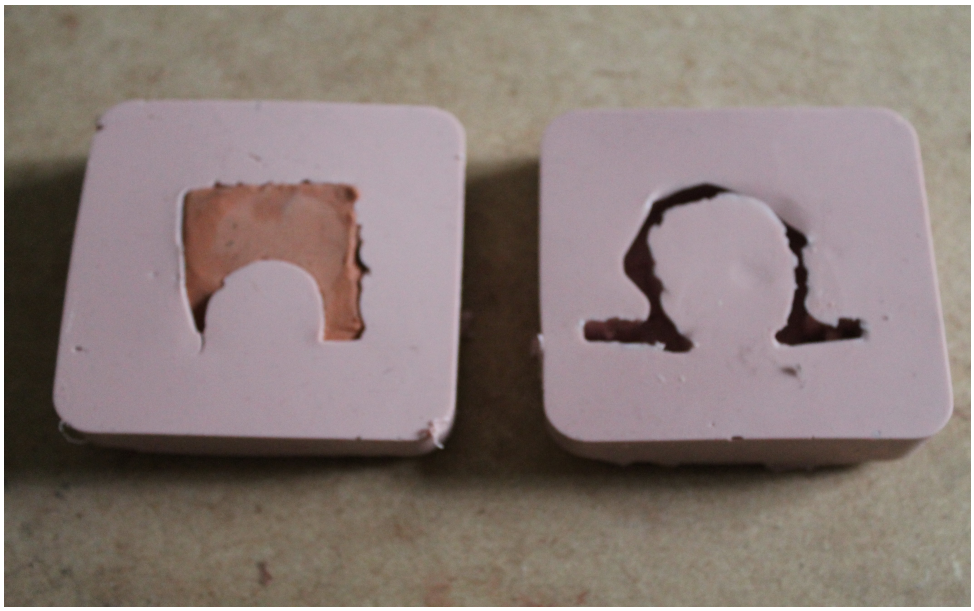


(Figura 11 – Protótipo abandonado de conector & molde)



(Figura 12 – Protótipo abandonado de conector – Abandonado por ser impreciso demais)

Apesar do fracasso dos conectores, os moldes do calço e do suporte geraram peças satisfatórias. Foram feitos inicialmente 4 calços e 2 suportes. 2 calços e 1 suporte para cada esteira dos eixos X e Y.

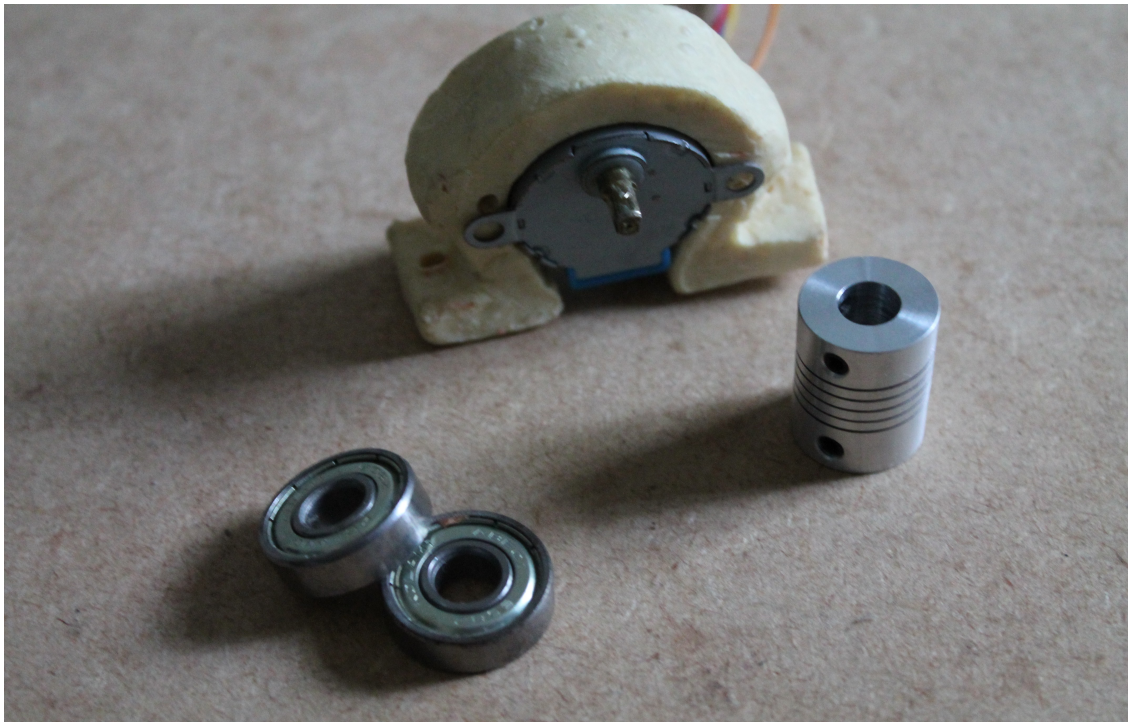


(Figura 13 – Moldes do calço para rolamento e suporte para motor)

Houve um pequeno acidente no primeiro suporte. A resina não catalisou bem e ele ficou imperfeito, com excesso de material e bolhas de ar. No entanto, isso não inutilizou a peça, tendo em vista que só influenciou visualmente em um acabamento mal feito. Sua funcionalidade se manteve e, portanto foi aproveitada.

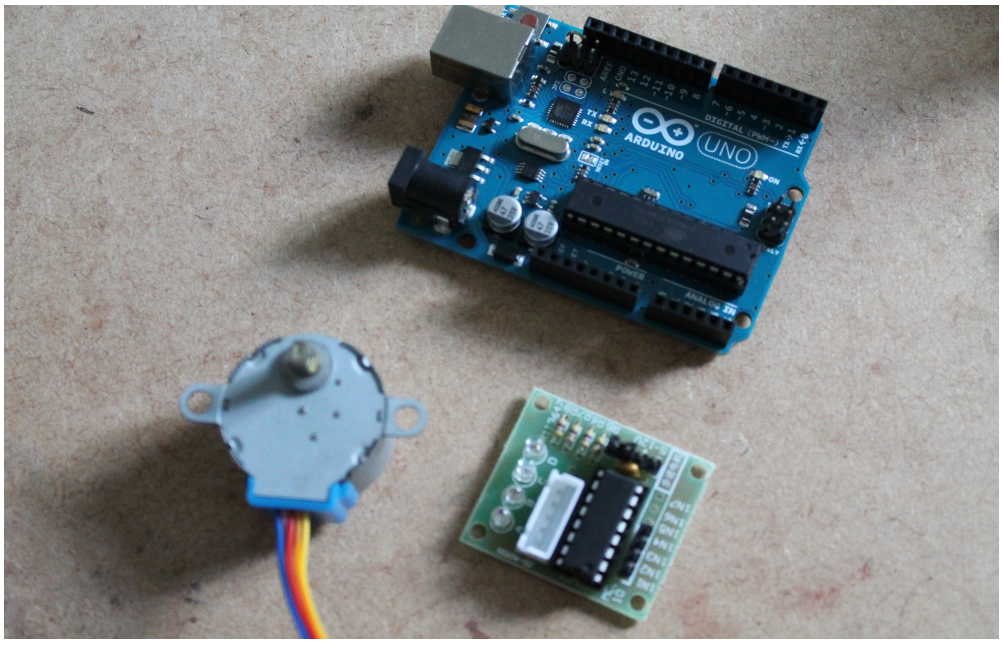
(Figura 14 – Moldes do suporte para motor e suporte (defeituoso, mas válido para o teste))

ps: foto a ser atualizada com uma peça não defeituosa.

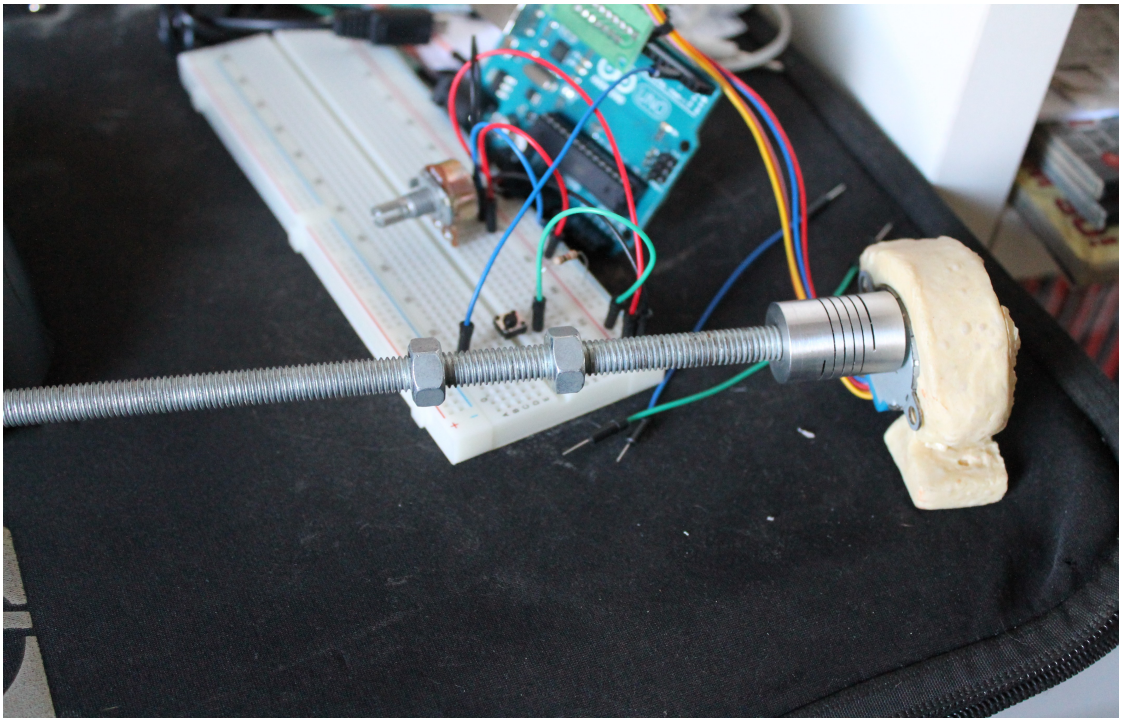


(Figura 15 – Molde acoplado ao suporte, conector flexível e rolamentos)

O motor escolhido foi do modelo 28byj-48 de aproximadamente 4075 passos por rotação. O motor é alimentado por uma carga de 5 a 12 V e recebe sinal de 4 pinos digitais ligados ao Arduino, que determinam a velocidade e sentido da rotação através de seus pulsos.



(Figura 16 – Motor, Arduino e interface)



(Figura 16 – Esteira montada e funcional (Foto a ser atualizada com imagem integral dela))

4 Conclusão

4.1 Contribuições

O presente trabalho implementou uma solução inicial para o desenvolvimento de uma impressora 3D com base em integrações de diversas tecnologias. A solução desenvolvida até então encaminha o processo de desenvolvimento de modo que os próximos passos sejam determinados de maneira intuitiva. A seção de bibliografia disponibiliza os principais links relevantes para o bom andamento do projeto.

4.2 Limitações

Por tratar-se de uma solução inicial, a implementação atualmente disponível, resultante deste trabalho, apresenta limitações, dentre as quais podemos destacar:

- Os motores de eixos diferentes não trabalham em paralelo, tendo que escalonar o tempo de execução de cada comando para diminuir a propagação de erro na impressão.

- O sistema ainda não recebe inputs pelo serial, só pelo console. O que impossibilida o envio do arquivo contendo o G-Code ao controlador.

- O terceiro eixo não foi implementado ainda, o que limita a impressão atual a 1 camada só.

- O design do modelo físico não terminou de ser implementado devido a carência de testes e da falta de implementação do último eixo.

- A solução desenvolvida para o erro de precisão da impressão não pôde ser testada e validada inteiramente devido à carência de um modelo físico completo e inteiramente funcional.

- Não existe nada do hardware ou do firmware referente ao extrusor ou à alimentação da impressora implementado ainda.

- Diferente de alguns projetos vistos, a presente solução não trabalha com uma cama aquecida para evitar que o material sofra distorção ao resfriar rápido demais.

4.3 Trabalhos Futuros

Algumas possibilidades para evolução do estágio alcançado por este trabalho são listadas a seguir:

- A implementação do terceiro eixo.
- A implementação do extrusor.
- A implementação do alimentador.
- A implementação de uma cama aquecida.
- Definição do design final da impressora.
- Implementação do envio do G-Code por serial para o controlador.
- Implementar uma versão que funcione em sistemas operacionais diferentes do Windows.

4.4 Considerações Finais

Com relação à proposta de solução para o iníciu do desenvolvimento de uma impressora 3D de modelo próprio, pode-se concluir que o resultado final é satisfatório, embora existam diversos pontos de evolução necessários para que sua função final seja alcançada. Entretanto, o desenvolvimento deste trabalho possibilitou a sedimentação de diversos conhecimentos tanto na área de desenvolvimento e projeto de sistemas como na integração entre físico de digital. Além disso, as diversas etapas pré impressão foram fundamentais para desmistificar o processo pelo qual um modelo 3D impresso passa.

Referências Bibliográficas

- 1 - Impressão de próteses

  <http://edition.cnn.com/2014/03/19/opinion/3d-print-arm-daniel/>

- 2 - Introdução ao conceito de STL

  <http://www.3dsystems.com/quickparts/learning-center/what-is-stl-file>

- 3 - Definição de RepRap

  <http://reprap.org/wiki/RepRap>

- 4 - Definição de motor de passo

  <http://pt.wikipedia.org/wiki/Motor_de_passo>

- Foruns RepRap.

  <http://forums.reprap.org/>

- RepRap Wiki.

  <http://reprap.org/wiki/Main_Page>

- Tutorial de conversão de CAD para STL

  <http://www.instructables.com/id/3D-Printing-your-Zbrush-Sculpts/?lang=pt >

- Manual de montagem da reprap Wallace

  <http://reprap.org/wiki/Wallace_Build_Manual>

- OpenSCAD (modelador 3D para programadores)

  <http://www.openscad.org/>

- github com o código para partes impressas da reprap Wallace

  <https://github.com/whosawhatsis/RepRap-Wallace>

- Partes impressas do reprap Wallace

<http://www.thingiverse.com/thing:14208>

- Programa Controlador da impressora

  <http://www.repetier.com/documentation/repetier-host/>

- Outro controlador para a aimpressora

  <http://reprap.org/wiki/RepSnapper_Manual:Introduction>

- Apresentação do universo reprap

  <http://reprapbook.appspot.com/#d0e20>