UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Trabalho de Conclusão de Curso

Proposta de Criação de um Serviço *Online* para Impressão Remota de Objetos 3D

Autores: Mateus Plez Ricciardi 7987442

Victor Luis Hatsumura Jácomo 8006992

Orientador: Evandro Luis Linhari Rodrigues

2017

MATEUS PLEZ RICCIARDI e VICTOR LUIZ HATSUMURA JÁCOMO

Proposta de Criação de um Serviço *Online* para Impressão Remota de Objetos 3D

Trabalho de Conclusão de Curso apresentado

à Escola de Engenharia de São Carlos, da

Universidade de São Paulo

Curso de Engenharia Elétrica

ORIENTADOR: Evandro L. L. Rodrigues

São Carlos

2017

Hatsumura Jácomo, Victor Luiz
H365p Proposta de criação de um serviço online para impressão remota de objetos 3D / Victor Luiz Hatsumura Jácomo, Mateus Plez Ricciardi; orientador Evandro Luís Linhari Rodrigues; coorientador José Carlos de Melo Vieira Júnior. São Carlos, 2017.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2017.

1. Arquivo .stl. 2. Impressora 3D. 3. Python. 4. Site. 5. Raspberry Pi. 6. GNU/Linux. 7. Linux Embarcado.. I. Título.

FOLHA DE APROVAÇÃO

Nome: Mateus Plez Ricciardi

Título: "Proposta de criação de um serviço online para impressão remota de objetos 3D"

Trabalho de Conclusão de Curso defendido e aprovado em 27 / 06 / 2017,

com NOTA 6,7 (seis , sete), pela Comissão Julgadora:

Prof. Associado Evandro Luis Linhari Rodrigues - Orientador - SEL/EESC/USP

Prof. Associado Ivan Nunes da Silva - SEL/EESC/USP

Prof. Associado Antonio Freitas Rentes - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado José Carlos de Melo Vieira Júnior

FOLHA DE APROVAÇÃO

Nome: Victor Hatsumura Jácomo

Título: "Proposta de criação de um serviço online para impressão remota de objetos 3D"

Trabalho de Conclusão de Curso defendido e aprovado em 27/06/2017,

com NOTA 6,7 (seis, sete), pela Comissão Julgadora:

Prof. Associado Evandro Luis Linhari Rodrigues - Orientador - SEL/EESC/USP

Prof. Associado Ivan Nunes da Silva - SEL/EESC/USP

Prof. Associado Antonio Freitas Rentes - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado José Carlos de Melo Vieira Júnior

Resumo

Este trabalho consiste no desenvolvimento e resultados do trabalho de conclusão de curso dos alunos de graduação Mateus Plez Ricciardi e Victor Luiz Hatsumura Jácomo. As vendas de impressoras 3D para uso doméstico se expandiram a partir do vencimento de suas principais patentes. Porém, por ser um mercado novo, ainda é muito fragmentado, o que dificulta o acesso a essa tecnologia para leigos. Neste cenário, este projeto visa a criação de um serviço *online* no qual clientes sem conhecimento técnico ou que não possuam impressoras próprias, possam enviar objetos tridimensionais no formato *.stl*, para impressão remota, com análise e envio dos resultados para a impressora através de códigos próprios. Para servir o *site* e processar as impressões enviadas foi utilizada a linguagem de programação *Python*, num ambiente com *GNU/Linux* embarcado rodando em uma *Raspberry Pi 2 Model B*.

Palavras-Chave: Arquivo *.stl*, Impressora 3D, *Python*, *Site*, *Raspberry Pi*, *GNU/Linux*, *Linux* Embarcado.

Abstract

This paper contains details about the development process and final results of the end-of-course project by undergraduate students Mateus Plez Ricciardi and Victor Luiz Hatsumura Jácomo. Sales of 3D printers for personal use have grown due to the expiration of its main patents. However, the market is still quite fragmented for being on its early days, which makes it more difficult for newcomers to have access to this technology. In this scenario, this project focuses on creating an online service where clients without technical knowledge or who do not own 3D printers themselves, are able to send their own three-dimensional objects with .stl extension to be printed remotely, with analysis and dispatch of results performed by our own code. The Python programming langue was used to host the website and process the prints, within a GNU/Linux environment running on a Raspberry Pi 2 Model B.

Keywords: .stl files, 3D printer, Python, Website, Raspberry Pi, Linux, Embedded Linux.

Lista de Figuras

Sumário

Capítulo 1

Introdução

O crescimento do mercado de impressoras 3D, devido à redução dos custos de produção e manutenção, fez com que se expandisse a quantidade de empresas e instituições interessadas nesse tipo de tecnologia. Com essa expansão, se tornará comum a produção em massa de produtos personalizados.

As áreas da indústria que mais se beneficiam com o avanço dessa tecnologia são a bioengenharia, na qual, por exemplo, ocorre a produção de próteses e órteses personalizadas para deficientes físicos, amputados, com perdas ósseas, entre outros; e a engenharia aerospacial, pois o envio de objetos para o espaço é caro, o que poderia ser evitado com a produção sob demanda dos mesmos já no espaço - neste caso, o custo se reduziria ao envio do material utilizado para as impressões.

Devido ao crescimento exponencial do uso doméstico de impressoras 3D, há diversos modelos e softwares disputando o mercado. A ideia do projeto é a criação de um serviço *web* de interface simples e amigável que facilite a impressão de objetos, via uma impressora 3D do tipo FDM.

O controle da impressora e a análise dos arquivos de impressão foram realizados com a linguagem de programação *Python* e suas ferramentas. A mesma linguagem também foi utilizada na criação do serviço *web* com interface entre o usuário e produto.

1.1 Objetivos

Este trabalho teve como objetivo a criação de um serviço online que permite a impressão de objetos tridimensionais a longa distância. O *site* possui uma interface amigável que visa facilitar todos os processos de impressão para o cliente, e um código próprio de interpretação de um arquivo de objetos 3D.

A linguagem de programação utilizada é o *Python*, pela sua flexibilidade e simplicidade. Foi utilizada uma placa *Raspberry Pi 2* como processador de informação para impressão e servidor para o *site*.

1.2 Justificativas

Apesar da redução dos custos de compra e manutenção de uma impressora 3D, ainda falta muito para que o equipamento se torne tão comum quanto computadores e celulares são atualmente, o que torna este um mercado com grande potencial para ser explorado.

Como o mercado é novo e em expansão, o mesmo ainda é muito fragmentado, o que faz com que o usuário comum tenha dificuldades em assimilar todas as plataformas. Desta forma, um serviço online que abstrai as complexidades técnicas poderia simplificar muito o processo para os usuários. Além da facilidade, outra utilidade seria o uso para aqueles que não possuem uma impressora 3D, uma vez que estas ainda possuem preço elevado apesar do barateamento.

1.3 Organização do Trabalho

Este trabalho está distribuído em 5 capítulos, incluindo esta introdução, dispostos conforme a descrição que segue:

Capítulo 2: Descreve o embasamento teórico sobre as impressoras 3D e os métodos de análise dos objetos 3D.

Capítulo 3: Discorre sobre os materiais e os métodos utilizados para a confecção do trabalho.

Capítulo 4: Apresenta os resultados parciais do trabalho até o momento da confecção da monografia.

Capítulo 5: Conclui sobre tudo que foi produzido e aprendido durante a confecção do trabalho até o presente momento.

Anexos: Discorre sobre alguns métodos de otimização de uso de material na impressão 3D, e apresenta os códigos usados para a manipulação do objeto 3D e manejo do site.

Capítulo 2

Embasamento Teórico

2.1 Estado da Arte

Nesta seção são apresentadas as mais novas tecnologias dentre as áreas estudadas para o trabalho.

2.1.1 Impressoras 3D

A impressão 3D evolui cada vez mais rapidamente, devido ao barateamento de sua produção e quedas de patentes.

Atualmente, o mercado de impressoras para consumidores comuns cresceu mais que os outros mercados[1]. Estima-se que o mercado total de impressoras 3D no mundo em 2022 poderá valer mais de 30 bilhões de dólares[2].

Indústrias como a civil, médica e aeroespacial se beneficiarão muito com esse desenvolvimento. Na área de engenharia civil, já existem casas inteiras impressas com esta tecnologia, casas improvisadas para casos de emergência, pontes de aço para pedestres sobre rios, impressoras de argila, entre outras[3].

Na área aeroespacial, estão sendo testadas impressoras que possam imprimir tijolos feitos de regolito, que é um tipo de solo bem fino e solto que faz parte do solo lunar e marciano, através de solos terrestres parecidos. O processo consiste na coleta do material e endurecimento do mesmo apenas com a luz solar concentrada [4]. Este método faria com que a construção de bases interplanetárias ou lunares se tornem extremamente baratas, pois seria necessário somente o envio da impressora e do coletor de material. Além de materiais para construção de bases espaciais, para economizar o peso enviado para o espaço, os astronautas poderiam imprimir qualquer ferramenta necessária sob medida e somente quando necessário, dando oportunidade de reciclagem dos materiais dos objetos após o uso.

Na área médica, o investimento atual ainda é baixo, mas a tendência é crescer cada vez mais. Por

enquanto, as impressoras 3D fazem modelos personalizados de próteses e órteses de todos os tipos, como membros e ossos. Mesmo com o investimento ainda baixo, existem atualmente pesquisas para impressão de tecido humano e até mesmo de órgãos completos, como um fígado[5].

2.1.2 *Site*

Desde a abertura da *Internet* para o público comum, a quantidade de sites na mesma cresceu exponencialmente. Atualmente é relativamente fácil criar seu próprio endereço, com muitos serviços que oferecem ferramentas para a criação e hospedagem.

2.2 Impressoras 3D

2.2.1 Breve Introdução Histórica

As primeiras impressoras de manufaturação aditiva foram criadas no início da década de 1980, sendo a primeira construída em 1981 por Hideo Kodama[6] que consistia num método de endurecimento de polímeros plásticos por luz.

Já em 1983, Chuck Hull registrou sua patente para sistemas baseados em estereografia, o primeiro a ser feito por camadas curadas por luz ultravioleta[7].

Durante este tempo, existiu também um tipo de impressão no qual eram pintadas várias camadas de folhas de papel, que depois eram alinhadas e postas em ordem e passado um produto químico. Este produto remove o papel que não possui a tinta, colando e endurecendo o resto. Este método, no entanto, era caro, falho e só era útil para objetos maciços.

Em 2009 as patentes para a FDM (*Fused Deposition Modeling*), o tipo mais comum de impressora 3D, expiraram, o que possibilitou a criação de modelos mais baratos. O preço foi reduzido de 200 mil dólares a cerca de 2 mil dólares[8].

2.2.2 Tipos de Impressão

Existem vários tipos de impressão 3D, cada um com suas vantagens, desvantagens, aplicações e materiais usados. Abaixo serão tratados alguns dos tipos mais comuns de impressão e seus materiais.

FDM (*Fused Deposition Modeling*)

Tipo mais comum e barato de impressão 3D que consiste no aquecimento de um filamento de um material pela cabeça extrusora que, em seguida, é depositado fazendo um trajeto determinado, camada por camada, resfriando automaticamente na cama da impressora, fornecendo a sua base para

sustentação, como exemplificado na figura 2.1.

Utiliza uma grande variedade de materiais como o plástico ABS, PLA, *nylon* e até mesmo bronze, carbono e madeira. Suas impressões têm uma aparência estratificada, como pode ser conferido na figura 2.2.

É uma opção para projetos de baixo custo, não sendo recomendado para designs muito arrojados devido ao seu modo de impressão que necessita de suportes e preenchimentos que aguentem a sustentação do mesmo.

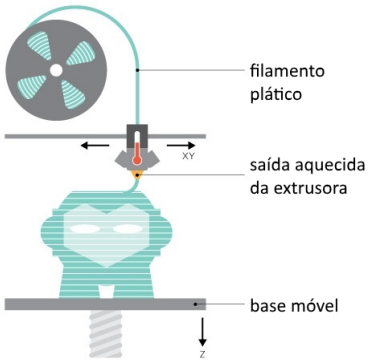


Figura 2.1: Esquema de uma impressora tipo FDM[8]

Figura 2.2: Objeto criado por uma impressora tipo FDM[9]

SLA & DLP (*Stereolithography and Digital Lighting Processing*)

Durante a impressão, uma plataforma é imersa no líquido do polímero e um *laser* (SLA) ou um projetor (DLP) mapeia cada camada do objeto e imprime-o, solidificando a resina, como mostrado na figura 2.3. A plataforma é levantada permitindo que uma nova camada de resina se forme embaixo da

camada recém criada. O ciclo é repetido.

Este método é limitado a resinas e é ótimo para a criação de objetos pequenos e detalhados, por possuir mais suavidade, como mostra a figura 2.4. Não é recomendado para objetos grandes ou que fiquem muito expostos a radiações ultravioleta excessivas.

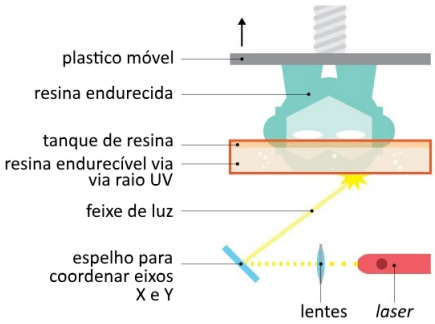


Figura 2.3: Esquema de uma impressora tipo SLA[8]

Figura 2.4: Objeto criado por uma impressora tipo SLA e DLP[10]

SLS (*Selective Laser Sintering*)

Este método usa *lasers* para o derretimento e a solidificação de camadas de pó do material até formar o objeto.

Ele funciona com duas plataformas: uma que irá conter o objeto a ser impresso, e outra com o pó para a formação do mesmo. A plataforma com o pó sobe um pouco e um rolo é passado distribuindo o mesmo sobre a outra plataforma que então recebe os *lasers* que solidificarão o material. Essa plataforma desce um pouco enquanto a outra sobe, e o rolo distribui novamente mais material sobre a primeira, e o processo de impressão se repete até o término do objeto, como exemplificado na figura 2.5.

Este método é muito aplicado na indústria, porém alguns modelos para usuários comuns já come-

çaram a surgir. Vários tipos de materiais podem ser utilizados, principalmente plásticos como *nylon* e poliestireno.

Sua grande vantagem é que pode ser usado para qualquer formato de objetos, pois o pó que não foi derretido serve como sustentação do mesmo, conforme visto na figura 2.6. Sua desvantagem está na demora do esfriamento do objeto.

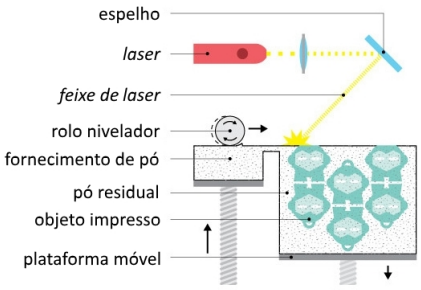


Figura 2.5: Exemplo de impressora tipo SLS[8]

Figura 2.6: Objeto criado por uma impressora tipo SLS[11]

***Material Jetting* (*PolyJet e MultiJet Modeling*)**

Parecido com o método de impressão 2D por jato de tinta, consiste na solta de material líquido num caminho determinado que é instantaneamente solidificado com *lasers* de radiação ultravioleta, processo que é repetido camada por camada. A imagem 2.7 ilustra o processo.

Possui um resultado final muito preciso e tem a capacidade de operar com diversas cores e tipos de materiais ao mesmo tempo, então consegue estruturas complexas que precisam de formas muito mais precisas e menos danificantes para o objeto, pois as sustentações podem ser feitas de outro material que usa um solvente diferente para ser retirado. Um objeto com materiais de cores diferentes, resultado desse tipo de impressão, pode ser conferido em 2.8.

Seu uso é mais industrial devido ao seu preço. Consegue usar uma variedade muito grande de

materiais diferentes, como plásticos opacos e transparentes, plásticos com propriedades semelhantes e borrachas, polipropileno. Tem uma precisão de 16 micrômetros por camada.

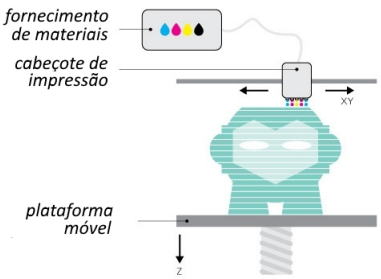


Figura 2.7: Exemplo de impressora tipo *Polyjet*[8]

Figura 2.8: Objeto criado por uma impressora tipo *Polyjet*[12]

Binder Jetting

Similar ao SLS, porém em vez de *laser* para a solidificação do material é usado um agente agregador que une o pó, como visto na figura 2.9. O processo das plataformas é o mesmo, a diferença é que depois de retirados os objetos necessitam ser colados por uma cola adesiva que dará força ao mesmo.

Mais usado na indústria, feito na maioria das vezes com areia colorida. Mais barato que o SLS e usado mais para modelos arquitetônicos ou esculturas, devido ao seu resultado menos suave, como pode ser observado na figura 2.10. Da mesma forma o material não colado serve como sustentação.

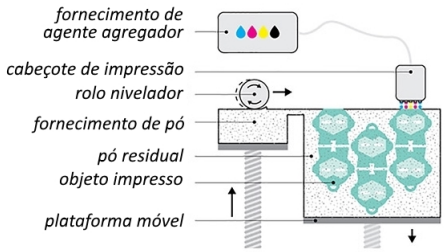


Figura 2.9: Exemplo de impressora tipo *binder*[8]

Figura 2.10: Objeto criado por uma impressora tipo *binder*[13]

***Metal Printing* (*Selective Laser Melting* e *Electron Beam Melting*)**

Estes são os dois tipos mais comuns de impressão 3D utilizando metais. Funcionam do mesmo modo que o SLS e o *Binder Jetting*, porém com um *laser* muito mais potente no modelo *Selective Laser Melting* (SLM), ou um raio de elétrons no modelo *Electron Beam Melting* (EBM). As plataformas funcionam do mesmo modo, porém diferente dos outros este necessita estruturas de sustentação devido ao peso do objeto criado e também para a transmissão do calor da estrutura para fora. As duas formas de impressão podem ser vistas na figura 2.11.

Os dois métodos são feitos, o primeiro com nível baixo de oxigênio e o segundo no vácuo, para impedir oxidação e estresse dos metais devido ao aquecimento. São de uso industrial e utilizam uma gama grande de metais como aço, alumínio, titânio, cromo-cobalto e níquel.

Muito utilizado nas indústrias aeroespaciais, aviônica, automotiva e de saúde. Seu resultado é muito preciso e suave, como visto na figura 2.12.

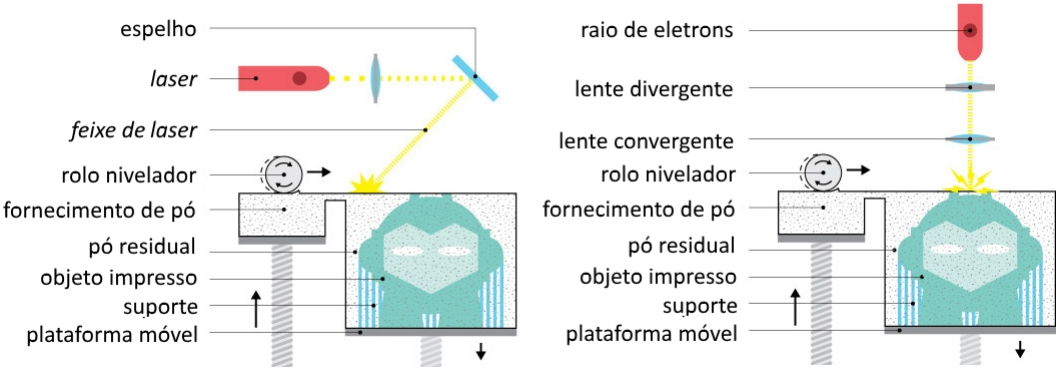


Figura 2.11: Exemplo de impressora tipo SLM e EBM[8]

Figura 2.12: Objeto criado por uma impressora tipo SLM[14]

2.2.3 Cuidados com a Impressão Utilizando Material ABS

O plástico ABS é o mais utilizado para impressões 3D do tipo FDM e o mais barato. Ele possui alta resistência e um ponto de fusão relativamente baixo, com uma transição para um estado vítreo em aproximadamente 105°C, e uma temperatura ideal de extrusão entre 170°C e 180°C. A *bed* (cama ou plataforma) da impressão deve estar com uma temperatura desejável de aproximadamente 70°C. Ele é utilizado, por exemplo, nas peças de *LEGO*.

Para a alimentação da impressora, o tipo de plástico ABS utilizado está no formato de filamentos, que são enrolados em uma bobina, como mostrado na figura 2.13, junto com peças de um brinquedo do mesmo material.

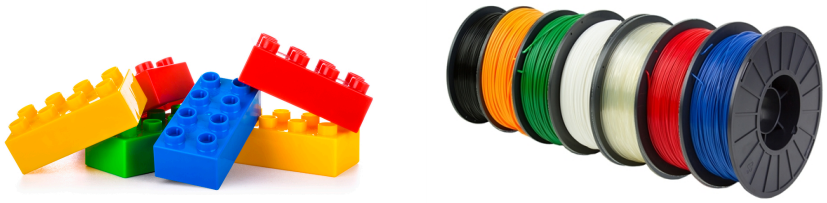


Figura 2.13: Peças de LEGO[15] e filamentos de cores variadas[16]

Uma temperatura muito baixa na extrusão pode acarretar o entupimento do canal e o material irá se solidificar muito rapidamente, então o mesmo não conseguirá colar na plataforma. Já uma temperatura muito alta poderá queimar o plástico, liberando gases tóxicos e, dependendo de sua intensidade, poderá entrar em combustão. Mesmo que não ocorra nenhum derretimento, as camadas ainda poderão ser irregulares, dado que a alta temperatura do plástico da camada acima pode interferir na de baixo.

A temperatura da cama também deve estar dentro de sua faixa de operação para o material. Uma intensidade menor pode resultar no material envergar. Já uma temperatura muito alta pode causar o derretimento do material, estragando a impressão.

Como na maioria das impressões por FDM, alguns cuidados devem ser tomados, como diminuir a exposição de circulação de ar na hora da impressão para impedir envergamentos do objeto, devido às mudanças de temperatura.

Além disso, para ajudar na fixação do objeto e impedir envergamentos, para o material ABS, pode-se utilizar, desde que a cama esteja limpa, *spray* fixador de cabelo ou então fitas adesivas como a *blue-tape* ou a *kapton-tape* na plataforma, sendo que a última possui uma resistência maior para altas temperaturas.

2.3 Formato *.STL*

Sigla para *"Standard Triangle Language"* (linguagem triangular padrão), o formato *.stl* é um formato CAD de descrição de objetos tridimensionais[17]. Normalmente, o formato não apresenta atributos como cor ou textura, apenas diversos triângulos que formam o objeto, apesar de existirem variações.

As informações armazenadas de cada triângulo são o vetor normal e seus vértices. Existem dois modelos de produção do arquivo: ASCII e binário. Como o modelo binário é mais compacto, é o mais utilizado. Um exemplo de um objeto 3D com suas triângulos pode ser observado em 2.14.

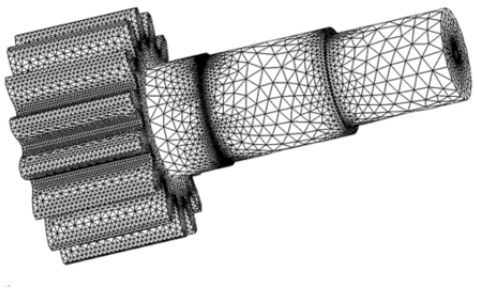


Figura 2.14: Objeto *.stl* com seus triângulos de Formação[18].

2.4 Geometria Analítica

Geometria Analítica é o estudo de geometria a partir de um sistema de coordenadas na qual são utilizados métodos algébricos para resolver problemas geométricos. Como toda a impressão pode ser descrito como um objeto contido em um sistema de coordenadas cartesianas, é imprescindível este campo matemático.

O objeto tridimensional apresenta apenas os dados de diversos triângulos que compõem a superfície. Para a construção do objeto com base no método de impressão FDM, é necessário o cálculo do corte do objeto para as diversas camadas, e para a produção de preenchimento e casca.

Cada camada do corte é criada com base na interseção entre o plano da camada e cada triângulo que há na superfície do objeto. Já o processo de preenchimento e casca, uma área deve ser coberta por diversas linhas internas, que podem ser criadas pela interseção entre retas e os segmentos que formam o triângulo.

2.4.1 Interseção de Reta e Segmento

A interseção de uma reta com um segmento pode ser um conjunto vazio, um ponto ou um segmento, devido à inclinação e posição.

A reta é um objeto geométrico de uma dimensão, de valor infinito.[19]

Para o cálculo de qualquer ponto contido em uma reta, é necessário um ponto definido l_0, e um vetor l multiplicado por uma constante d, formando a equação (2.1).

$$p = d.l + l_0 \tag{2.1}$$

O segmento é uma parte definida da reta.[19] Os resultados possíveis são limitados para um ponto p_1 inicial e um p_2 final. A relação descrita em sua forma matemática pode ser vista na equação (2.2)

$$p = d.l + l_0, \quad p_1 \leq p \leq p_2 \tag{2.2}$$

Se há interseção, existe pelo menos um p igual. A análise do conjunto resultado da interseção é feita separando as condições em paralela e inclinada. Se forem paralelas entre si e tiver pelo menos um ponto em comum, todo o segmento de reta é a resposta. Caso haja uma inclinação existe apenas um ponto possível. Se esse ponto estiver fora do limite (p_1, p_2), o resultado é um conjunto vazio. Se oposto, existe um ponto em comum p.

2.4.2 Interseção de Segmento de Reta e Plano

Como no tópico anterior, a interseção de um segmento de reta em um plano pode resultar em um ponto, um segmento ou conjunto vazio, dependendo da inclinação e posição dos dois elementos.

O plano é um objeto infinito de duas dimensões[19]. Para o cálculo de um ponto qualquer pertencente ao plano, é necessário um ponto previamente conhecido p_0 e um vetor normal n ao plano. Subtraindo-se p_0 de p é gerado um vetor. Caso este vetor seja perpendicular a n, o ponto pertence ao plano. Uma forma de se descobrir essa relação é via a multiplicação de elementos desses vetores. Caso seja zero, é perpendicular, visível matematicamente na equação (2.4).

$$(d.l + l_0 - p_0).n = 0, \quad p_1 \leq d.l + l_0 \leq p_2 \tag{2.3}$$

Caso o plano e o segmento não sejam paralelos, é possível determinar igualando os p. Se o resultado estiver entre l_1 e l_2, há interseção de um ponto. Se não, o resultado é um conjunto vazio.

$$(p - p_0).n = 0 \tag{2.4}$$

Para o caso paralelo, se o plano e segmento forem coincidente, o resultado será o próprio segmento, posto que ele estará inteiramente contido no plano. Caso não sejam coincidentes, não possuem nenhum ponto em comum.

2.4.3 Interseção de um triângulo e o plano

Um caso específico, mas necessário para o problema proposto. Decompondo o triângulo em segmentos, é realizado a interseção entre os planos. Se o triângulo for paralelo, existem duas possíveis soluções: o conjunto vazio, se for externo ao plano, ou o próprio triângulo, se for coincidente. A figura 2.15 demonstra a condição de que o triângulo é coincidente com o plano.

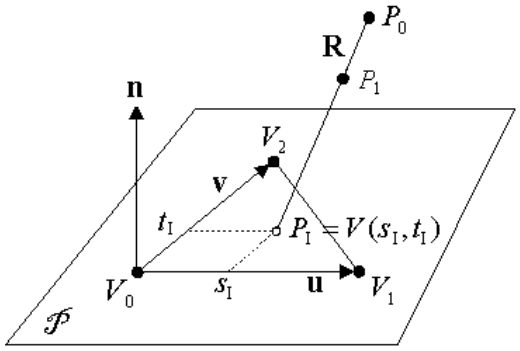


Figura 2.15: Intersecção de um triângulo com um plano cujas normais são paralelas[20]

Se o triângulo for inclinado em relação aos planos, existem três soluções possíveis: conjunto vazio, um ponto ou um segmento. O vazio acontece se todos os vértices estiverem acima ou abaixo do plano. A figura 2.16 mostra as três possíveis soluções, e como ocorrem.

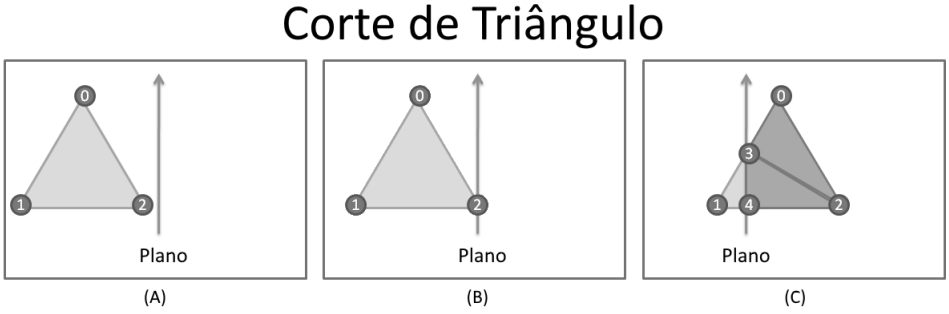


Figura 2.16: (A) Não há interseção. (B) Há interseção e há um ponto em comum. (C) Há interseção e há um segmento de reta em comum[21]

2.5 Teoria de Grafos

Grafos são representações de objetos ou caminhos por seus vértices e arestas. Um exemplo pode ser conferido na figura 2.17. O conjunto dos vértices apresenta as informações relativas aos pontos, podendo ser apenas uma ordem numérica, ou coordenadas. Já as arestas representam as ligações entre pontos contidos no conjunto dos vértices, ligadas em pares.

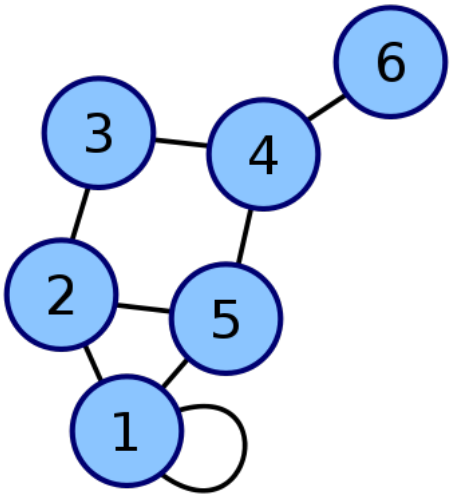


Figura 2.17: Representação gráfica de um grafo. [22]

Conjunto de vértices(V) e arestas(E) do grafo contido na figura 2.17 pode ser vistos nas equações (2.5).

$$V = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \qquad E = \begin{pmatrix} (1,1) \\ (1,2) \\ (1,5) \\ (2,3) \\ (2,5) \\ (3,4) \\ (4,5) \\ (4,6) \end{pmatrix} \tag{2.5}$$

2.5.1 Matriz de Adjacência

A matriz de adjacência é uma representação matricial das arestas contidas em um grafo. Suas ligações podem ser direcionadas e não direcionadas. As primeiras são as em que as ligações independem do ponto escolhido. Identificando os dois pontos como A e B, é possível realizar o caminho A para B, e B para A. Já o não direcionado, não é possível o retorno, apenas o caminho determinado do ponto A ao B é permitido.

A matriz apresenta como propriedade o fato de ser quadrada e, no caso de todas as ligações serem não direcionadas, simetria em relação à diagonal principal. A representação matricial da figura 2.17 pode ser visto em (2.6).

$$\begin{pmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.6}$$

2.5.2 Método de Busca

Existem diversas formas de busca de soluções numa estrutura em grafo.

O método de busca cega é aquele cujos caminhos apresentam o mesmo peso de escolha. Existem duas formas de se realizar: busca em profundidade e em largura. Devido a circunstância do problema que utilizou este algoritmo, foi escolhida a busca em profundidade.

Busca em Profundidade

Busca em Profundidade ou *Depth-First Search* é a forma mais simples de busca de um nó ou ponto final F via um nó ou ponto inicial I definido. A partir de I, são escolhidos nós consecutivos, até que se alcance o nó desejado, ou encontre um final. Caso o último aconteça, retrocede-se até um nó conectado a um nó filho não explorado e realiza-se o mesmo procedimento. Um exemplo de busca em profundidade pode ser visto na figura 2.18.

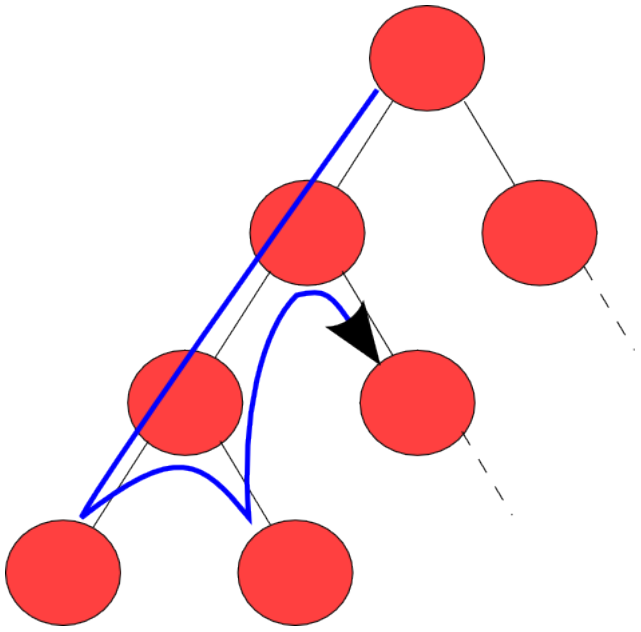


Figura 2.18: Demonstração da execução do algoritmo de busca em profundidade[23].

Há dois problemas visíveis nesse procedimento, a necessidade de evitar *loops*, cujos caminhos seriam potencialmente infinitos, e o fato de que não ser um processo rápido já que não há critério de escolha de caminho.

Busca em Largura

A Busca em Largura ou *Breadth-First Search* consiste em começar a busca pelo nó raiz e então depois explorar os nós vizinhos. Se não houver nenhum, passa para a próxima linha nos nós inexplorados até encontrar o resultado desejado. Esse processo de busca pode ser conferido na figura 2.19.

Este processo tem um problema de redundância de nós visitados, então para garantir que isso não ocorra, os nós já visitados são marcados, e seus vizinhos são enfileirados, então somente os nós nas filas são verificados.

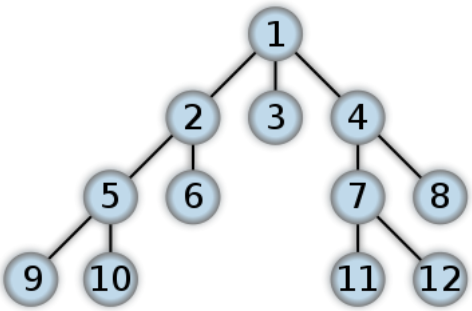


Figura 2.19: Demonstração da ordem de execução do algoritmo de busca em largura[24].

2.6 Algoritmo de Bresenham

Desenvolvido por Jack Elton Bresenham em 1962 para o computador IBM 1401[25], é um código de rasterização, isto é, de produção de linha. Apesar de gerar erro de amostragem, é uma forma rápida de produção para sistemas matriciais. Um exemplo de como este algoritmo preencheria uma matriz de pontos dado uma reta, pode ser conferido na figura 2.20.

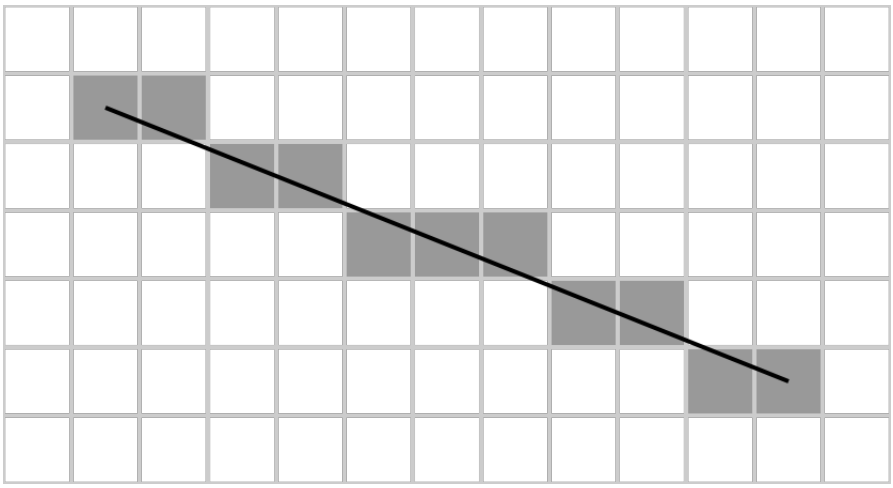


Figura 2.20: Ilustração da aplicação do Algoritmo de Bresenham[26].

2.7 Processamento do Objeto 3D

2.7.1 *Slicing* -Corte de um objeto tridimensional em planos

Para o corte, são utilizados diversos planos perpendiculares ao eixo Z, todos espaçados em relação a metade do passo da impressora. O formato *.stl* armazena suas informações em triângulos, assim, são decompostos em segmentos. Inicialmente é testado se os pontos em relação ao z estão dentro do erro, caso contrário, não há necessidade de todo o processo de interseção. Após o processo, as matrizes com os pontos são transformados em uma matriz de adjacência. Apesar de todas as informações estarem contidas nos itens anteriores, a adjacência revela todas as ligações de um determinado ponto, o que facilita o processamento.

2.7.2 Contorno de Perímetro

A forma encontrada de criar o perímetro é uma busca cega modificada utilizando a matriz de adjacência. Diferente da busca tradicional, que tem uma definição de fim, esta o fim é não haver conexões não utilizadas. O algoritmo utilizado foi simples: verifica os números e cada linha. Se for 0 continua, caso contrário, vai para o ponto definido por esse bit, apaga-se o valor deste ponto e de seu simétrico, e continua até que todos os pontos da matriz serem zero, um exemplo de ser visto em (2.7) e (2.8).

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & \underline{\mathbf{0}} & 1 & 0 \\ \underline{\mathbf{0}} & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & \underline{\mathbf{0}} & 1 \\ 1 & \underline{\mathbf{0}} & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & \underline{\mathbf{0}} & 0 \\ 0 & 0 & 0 & 1 \\ \underline{\mathbf{0}} & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{2.7}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{\mathbf{0}} \\ 0 & 0 & 0 & 1 \\ 0 & \underline{\mathbf{0}} & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{\mathbf{0}} \\ 0 & 0 & \underline{\mathbf{0}} & 0 \end{pmatrix} \tag{2.8}$$

Os caminhos encontrados a partir das relações (2.7) e (2.8) respectivamente foram:

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \tag{2.9}$$

$$2 \rightarrow 4 \rightarrow 3 \tag{2.10}$$

O caminho mais eficiente seria o maior caminho possível, já que evitaria tempo de locomoção desnecessária da extrusora, isto é, tempo em que a extrusora não derrete plastico e apenas se locomove. Os caminhos gerados em (2.9) e (2.10) não são os mais eficientes possíveis, mas há a certeza de que todas conexões foram preenchidas. O processo é extremamente demorado, como se checa toda a linha da matriz, e não apenas a segunda metade.

Um método mais rápido, porém que gera caminhos mais curtos, e portanto, piores, é pelo emprego dos elementos que estão acima ou abaixo da diagonal principal. É desempenhado um código semelhante ao que usa a matriz inteira, porém muito menos custoso, já que é necessário apenas a leitura de metade da matriz. Abaixo um exemplo:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & \underline{\mathbf{0}} & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & \underline{\mathbf{0}} & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & \underline{\mathbf{0}} \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{2.11}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & \underline{\mathbf{0}} & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{2.12}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & \underline{\mathbf{0}} \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{2.13}$$

São gerados respectivamente via os processos (2.11), (2.12) e (2.13), os caminhos:

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \tag{2.14}$$

$$1 \rightarrow 3 \tag{2.15}$$

$$2 \rightarrow 4 \tag{2.16}$$

2.7.3 *Infill* - Criação de Área

Há dois preenchimentos: Casca externa e preenchimento interno. A diferença entre o preenchimento e a casca é a quantidade de segmentos que os preenchem, da qual a casca é mais densamente preenchida, pois é o acabamento externo do objeto.

Para a criação do preenchimento, são estudadas as ligações encontradas na matriz de adjacência. Como a continuidade não importa, apenas meia matriz pode ser analisada. Após o processo, cada contorno é analisado separadamente e feito o preenchimento de linha, como descrito na imagem 2.21.

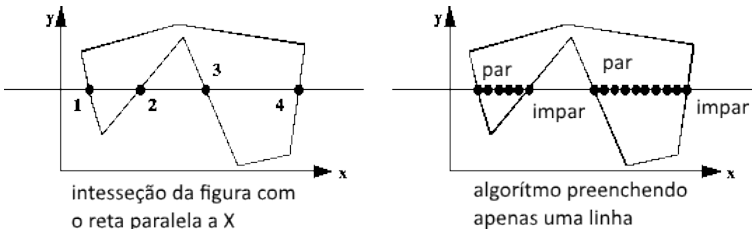


Figura 2.21: Método de preenchimento com base em ordem numérica de pontos[27]

O preenchimento é feito de forma que cada linha tenha a direção contrária da anterior, para minimização do tempo de locomoção da extrusora, semelhante a visível na figura 2.22.

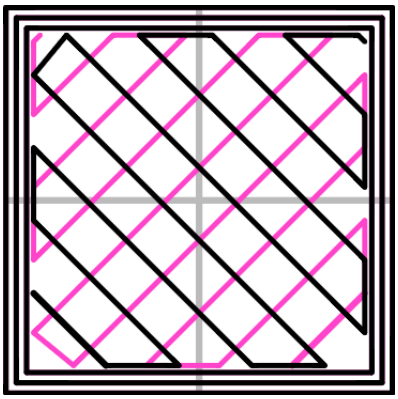


Figura 2.22: *Infill* utilizado pelo programa *Slic3r*. O código feito é semelhante, apesar de que há variação dos ângulos de impressão [28]

Capítulo 3

Materiais e Métodos

O esquema da figura 3.1 apresenta os principais componentes do trabalho, e como os mesmos interagem entre si.

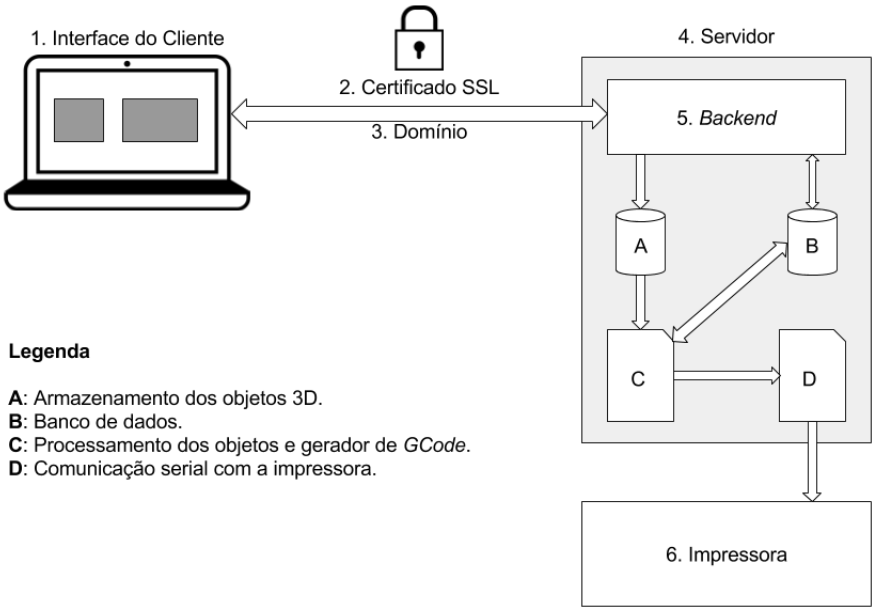


Figura 3.1: Diagrama representativo do projeto

3.1 Materiais

3.1.1 Placa *Raspberry Pi 2*

Para o servidor do trabalho, item 4 no diagrama da figura 3.1, foi escolhida a utilização de uma *Raspberry Pi 2 Model B*. Ela faz parte do leque de produtos da organização *Raspberry Pi* que produz computadores em placa única acessíveis e relativamente potentes.[29]

O modelo escolhido possui basicamente 4 saídas USB, 1 saída de som *Jack* 3,5mm, 1 saída de vídeo HDMI, 40 pinos GPIO, 1 porta *Ethernet*, *900MHz quad-core ARM Cortex-A7 CPU*, 1GB de memória RAM e entrada para cartão *micro SD*, núcleo gráfico *VideoCore IV 3D*.[30]

Para mais informações sobre *Raspberry Pi* acesse o *link*: `https://www.raspberrypi.org/` . Uma foto contendo a vista superior de uma *Raspberry Pi2 Model B* pode ser conferida na figura 3.2.

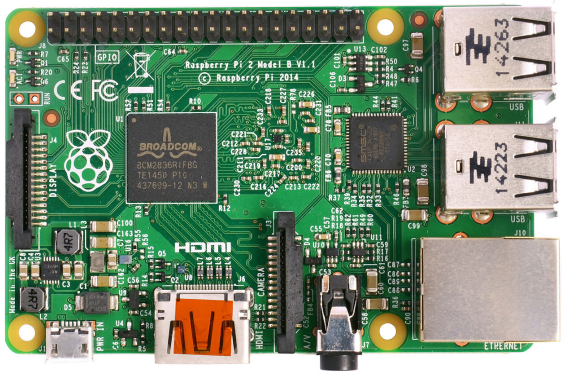


Figura 3.2: Foto de uma *Raspberry Pi 2 Model B*.[29]

3.1.2 Sistema Operacional *Raspbian*

O sistema operacional utilizado foi o *Raspbian*, que é um GNU/Linux baseado em *Debian* e compatível com a *Raspberry Pi*, mantido pela empresa homônima. É um sistema robusto e estável com grande documentação e comunidade ativa. O *link* para *download*: `https://www.raspberrypi.org/downloads/raspbian/` .

3.1.3 Impressora *CNC Brasil*

É uma impressora 3D da empresa brasileira *CNC Brasil* de um modelo já antigo, sem dados nas documentações da empresa fornecida. Ela possui uma cabeça extrusora, uma cama com aquecimento automático e uma entrada USB, com seu sistema de controle feito por um *Arduino* interno com módulos de potência. A foto da mesma pode ser vista na figura 3.3, e está representada no diagrama da figura 3.1 como item de número 6.

Ela suporta o material ABS que é um plástico barato e resistente, de temperatura de fusão relativamente baixa e com alto nível de reciclabilidade.

Figura 3.3: Foto da impressora CNC Brasil do laboratório LAVISIM.

3.1.4 Firmware de Impressora *Marlin*

Marlin é um *firmware* para impressoras 3D de código aberto e gratuito, sendo o mesmo utilizado pela impressora CNC Brasil. Possui suporte para várias impressoras baseadas em *Arduino*.[31]

Seu *GitHub* é público e pode ser acessado neste *link*: `https://github.com/MarlinFirmware/Marlin` .

3.1.5 Software de Impressão *MatterControl*

MatterControl é um aplicativo que permite ao usuário comum enviar objetos 3D em diversos formatos, e nele se faz o fatiamento e controla a impressora, usando para calibrá-la, ajustar a temperatura e o preenchimento. Foi utilizado inicialmente para controle da impressora.

Link para acesso: `http://www.mattercontrol.com/` .

3.1.6 Software de Modelamento 3D *Blender*

Blender é uma ferramenta gratuita e de código aberto para criação de modelos 3D, podendo ser usado para a criação inclusive de jogos e filmes. Disponibiliza uma API em *Python*. Foi utilizado para a criação de um *preview* da impressão no *site*.

Site para *download* e mais informações: `https://www.blender.org/` .

3.1.7 Linguagem de Programação *Python*

Python é uma linguagem de alto nível de distribuição gratuita, e que possui enorme documentação e bibliotecas com uma comunidade extremamente ativa.

Para o tratamento de objetos 3D, com o armazenamento dos mesmos representado na figura 3.1 pelo item A, foram utilizadas as bibliotecas *numpy-stl*, para modificação de arquivo de formato *.stl*; *mecode*, para geração e envio serial de G-Code, representados pelos itens C e D, no mesmo diagrama. Além de bibliotecas de base: *numpy*, *math*, *serial* e *time*.

Para a criação do *site* foi usada o *framework Flask*, o qual oferece funções que facilitam a renderização, obtenção e envio de dados, representado pelo item 5 no diagrama. *Python* também foi usado para a comunicação com o banco de dados, sendo o último indicado como o item B.

Para o servidor *web Python* foi usado o *Gunicorn* em conjunto com *Nginx*, utilizado como *proxy*. Um serviço recorrente (*cronjob*) verifica a cada minuto se o servidor está sendo executado, e reinicia o mesmo caso não esteja. Todos esses componentes também fazem parte do item 5 no diagrama.

3.1.8 *Framework fronted Materialize*

Materialize é um *framework frontend* para *sites* baseado no *Material Design* do *Google*, possuindo documentação e suporte. Foi utilizado para a composição do sistema de interface do cliente, representado pelo item 1 no diagrama da figura 3.1.

Pode ser acessado pelo *link*: `http://materializecss.com/` .

3.1.9 Site de Armazenamento *GitHub*

Todo o código produzido foi armazenado no *GitHub*.

GitHub é uma plataforma de criação e compartilhamento de projetos para programadores que oferece várias vantagens como um histórico de todas as mudanças enviadas, criação de *branches* para o projeto, ver o que cada pessoa do time contribuiu para o mesmo e suas mudanças. Possui duas versões, gratuita e paga, sendo que a primeira concede um número limitado de projetos privados para serem criados.

Ela também possibilita a fácil atualização e integração do código com todos os envolvidos através de simples comandos. Acesse pelo *link*: `https://github.com/` .

3.1.10 Projeto de Segurança *Let's Encrypt*

Let's Encrypt é um projeto da *Linux Foundation* que provê certificados de segurança gratuitos, abertos e automatizados. O projeto não tem fins lucrativos e conta com apoio de diversas entidades, empresas e organizações como a *Google*, *Mozilla Foundation*, e *Facebook*.

Ele garante que a comunicação entre o cliente e o servidor, representados, respectivamente, pelos itens 1 e 4 do diagrama na figura 3.1, esteja criptografada, sendo representado pelo item 2 no mesmo

diagrama.

Pode ser acessado pelo *link*:`https://letsencrypt.org/` .

3.1.11 Servidor DNS *Dot tk*

Dot tk é um serviço de *DNS* gratuito no qual pode-se reservar um domínio por um determinado período de tempo, sendo possível realizar renovações. Foi utilizado para criar o endereço `asap3d.tk` , representado então pelo item 3 do diagrama na figura 3.1. Pode ser conferido através do *link*:`http://www.dot.tk/pt/index.html?lang=pt` .

3.1.12 Linguagem de Programação G-Code

G-Code[32] é a forma de envio de informação utilizada por diferentes *firmwares* no mercado, como *Marlin*, *RepRap*, entre outros, com algumas diferenças de comandos entre versões e marcas. Em sua maioria por uma letra, representando a ação, seguida de um número, disponibilizando um valor. A tabela 3.2 demonstra as palavras aceitas. Um conjunto dos comandos pode ser visto na tabela 3.1, com uma breve explicação.

Tabela 3.1: Lista de comandos G-Code

Comando	**Significado**
G0 e G1	Comandos de controle de posição e Extrusora. G0 é mais rápida que G1
G28	Move para a Origem. Também realiza um processo de calibragem
G90	Define a posição atual como a origem
M84	Desliga o modo de espera da impressora, utilizado no fim da impressão
M104	Define a temperatura da extrusora
M105	Revela a temperatura da extrusora
M109	Define a temperatura da extrusora e espera
M140	Define a temperatura da *Bed*
M149	Define a unidade de temperatura. C para Celsius e K para Kelvin
M190	Espera um temperatura da *Bed* ser alcançada

Tabela 3.2: Lista de possíveis comandos G-Code

Letra	**Significado**
Gnnn	Comando padrão G-Code, como o movimento para um ponto
Mnnn	Comandos definidos para RepRap, como ligar ventiladores para resfriamento
Tnnn	Seleção de ferramentas nnn, normalmente associadas com o bico da(s) extrusora(s)
Snnn	Comando de parâmetros, como tempo em segundos, temperatura e tensão do motor
Pnnn	Comando de parâmetros, como o tempo (ms), o valor proporcional (Kp) de sintonização PID
Xnnn	Cordenada X, usualmente para o movimento. Aceita número fracionário
Ynnn	Cordenada Y, usualmente para o movimento. Aceita número fracionário
Znnn	Cordenada Z, usualmente para o movimento. Aceita número fracionário
U,V,W	Eixos de coordenada adicionais (RepRapFirmware)
Innn	Parâmetro de movimento sentido X em Arco, o valor integrativo (Ki) de sintonização PID
Jnnn	Parâmetro de movimento sentido Y em Arco
Dnnn	Parâmetro utilizado pelo diâmetro, o valor derivativo (Kd) de sintonização PID
Hnnn	Parâmetro ligado ao aquecimento
Fnnn	Avanço em mm/s. (Velocidade de movimento do cabeçote de impressão)
Rnnn	Parâmetro utilizado para temperatura
Qnnn	Parâmetro não utilizado atualmente
Ennn	Comprimento do filamento consumido da extrusora.
Nnnn	Número de linhas. Para solicitação de repetição de transmissão, caso haja erro de comunicação.
*nnn	Soma de verificação. Utilizado para checar erros de comunicação.

3.2 Métodos

Os códigos completos podem ser acessados em `https://github.com/Plezito/TCC_fuji_plez` . A seguir seguem os métodos utilizados para a criação do servidor *web*, *design* do *site*, banco de dados e envio, edição e transformações dos arquivos enviados, e as configurações necessárias para a utilização da *Raspberry Pi 2*.

3.2.1 *Web Server* em *Python*

O servidor do *site* que permite cadastros de usuários e envio de arquivos no formato *.stl* para impressão, foi desenvolvido em *Python* utilizando o *framework Flask*.

O *framework* utilizado tem como objetivo ser simples de configurar e usar, sem prejuízos à perfor-

mance e funcionalidade.

A organização dos arquivos que compõem o *site* é composta pelo arquivo *controller.py*, que é onde todas as rotas (isto é, caminhos ou URL válidas) são definidas, juntamente com as ações a serem realizadas. Foram criados pacotes utilitários (*helpers*) para prover funções básicas utilizadas diversas vezes durante o projeto, o que melhora muito a organização, legibilidade e manutenção do código.

As ações associadas a cada rota são responsáveis por obter dados do banco de dados, validá-los, armazená-los quando necessário, e então produzir um conteúdo em *HTML* que será exibido ao usuário, ou então redirecionar para outra rota que o faça.

A estrutura básica de uma rota é como segue:

```
@app.route( '/path', methods = [ 'GET', 'POST']) # Define o caminho e métodos.
@login_required   # Decora a rota para restringi-la a usuários logados.
# Outras decorações podem ser utilizadas, como a @requires_admin
def path():
  do_some_work()   # Realiza as ações necessárias para a rota.
  return render_template( 'some_file.html') # Retorna um arquivo HTML.
```

Além do arquivo *controller.py*, o arquivo *dbconnector.py* também apresenta funções muito importantes ao funcionamento do site. Através dele, são realizadas operações no banco de dados. Todas as funções definidas neste arquivos possuem proteção para que elas só sejam executadas casa o usuário tentando executá-las tenha um nível de acesso suficiente para tal. Esta camada de proteção é somada à camada já provida em *controller.py*, onde cada rota pode definir um nível mínimo de acesso para estar ou não disponível.

As senhas dos usuários são armazenadas no banco de dados de forma criptografada e com a utilização de *salts* (“temperos”), de modo que mesmo senhas idênticas tenham representações distintas quando armazenadas.

Os arquivos HTML do projeto estão no diretório *templates*. O arquivo *base.html* possui a estrutura básica de todo o site. Todos os outros arquivos estendem o arquivo base, garantindo, deste modo, uma aparência consistente em todas as páginas. Os arquivos em *JavaScript* e *CSS* do projeto, bem como imagens, estão disponíveis no diretório *static*.

3.2.2 Servidor *Python (Gunicorn e Nginx)*

Durante o desenvolvimento, é possível utilizar um servidor simples provido pelo próprio *framework Flask* utilizado para codificação. Para iniciar este servidor, base executar em um terminal:

```
python controller.py
```

Este servidor, no entanto, não é próprio para uso em ambientes de produção.

Para executar o código implementado em *Python*, foi utilizado o *Gunicorn* e, para servir de *proxy* entre o *Gunicorn* e o mundo externo, foi utilizado o *web proxy Nginx*. Com a utilização dos mesmos, é possível definir diversas configurações de segurança e desempenho. O certificado *SSL*, por exemplo, que criptografa toda a comunicação realizada entre o servidor e os usuários, é definido no *Nginx*.

3.2.3 Banco de Dados

Para poder guardar e acompanhar melhor os dados dos usuários, do *chat* e das impressões, foram criadas tabelas num banco de dados.

Foi escolhido o servidor *MySQL* para criar e atualizar as tabelas criadas. Foram então criados os seguintes arquivos com as definições das tabelas.

user_db.sql

Tabela que armazena dados sobre os usuários do sistema.

```
drop database if exists asap3d;
create database asap3d;

use asap3d;

CREATE TABLE users
(

FirstName  varchar(255)  NOT NULL,
LastName  varchar(255)  NOT NULL,
UserEmail  varchar(255)  NOT NULL  UNIQUE,
UserName  varchar(255)  PRIMARY KEY,
UserPass  varchar(255)  NOT NULL,
Gender  ENUM('male', 'female', 'other') NOT NULL,
Birthdate  DATE NOT NULL,
Status  ENUM('active', 'inactive', 'banned', 'admin') NOT NULL

);
```

print_db.sql

Tabela que armazena dados sobre impressões, sejam elas pendentes, processadas ou concluídas.

```
use asap3d;

CREATE TABLE prints
(

PrintId  int AUTO_INCREMENT PRIMARY KEY,
UserName  varchar(255)  NOT NULL,
CreationTimestamp    TIMESTAMP NOT NULL  DEFAULT "1970-01-01 00:00:00",
LastUpdateTimestamp    TIMESTAMP NOT NULL
DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
FileName  varchar(255)  NOT NULL,
FileSize  int NOT NULL,
Status  ENUM('received', 'processed', 'queue', 'printing', 'finished', 'canceled')
NOT NULL,
PrintingTime   int,
Material  real,
Divisions  int,
Cost  real,
FOREIGN KEY (UserName)   REFERENCES users(UserName)    ON DELETE CASCADE

);
```

chat_db.sql

Tabela que armazena dados dos *chats* entre os usuários e os administradores.

```
use asap3d;

create table chat(

MessageId  int AUTO_INCREMENT PRIMARY KEY,
UserNameFrom   varchar(255)  NOT NULL,
UserNameTo  varchar(255)  NOT NULL,
```

```
Message  varchar(1023)  NOT NULL,
MessageTimestamp   TIMESTAMP NOT NULL  DEFAULT "1970-01-01 00:00:00",
FOREIGN KEY (UserNameFrom)   REFERENCES users(UserName)   ON DELETE CASCADE

);
```

3.2.4 Configuração do *Crontab*

Foi usado a ferramenta *crontab* para a verificação do *status* do serviço, fazendo-o iniciar ao carregar o *Linux* ou reiniciando-o caso necessário.

Foi criado o seguinte arquivo do tipo *.sh* como modelo do *script* que é executado periodicamente pelo *crontab*.

```
#!/bin/bash

# To install this, run
# sudo crontab -u www-data -e
# and add an entry like
# */5 * * * * /caminho/para/este/arquivo

GUNICORN=/path/to/gunicorn

PID_FILE=/tmp/asap3d.pid

SOCKET_FILE=/path/to/controller.sock

APP_HOME=/path/to/site

# Do not edit below this line.

# If pid file exists
if [ -f $PID_FILE ] ;   then
  # And process is alive
  if kill -0  $(cat $PID_FILE  ) ; then
    echo  "ASAP3D is already running"
```

```
    exit 0
  fi
fi

# Otherwise, asap3d is not running and we start it.
cd $APP_HOME
$GUNICORN --daemon --workers 1 --pid $PID_FILE --bind unix:$SOCKET_FILE -m 007 wsgi:app
if [ -f $SOCKET_FILE ] ;   then
  chown pi:www-data $SOCKET_FILE
fi
echo "ASAP3D has been started"
```

Depois deve-se usar o seguinte comando.

sudo crontab -l

Para editar a lista de *cronjobs*.

sudo crontab -u www-data -e

Esse comando irá abrir o editor de texto padrão, então nele deve-se colocar.

** * * * * /bin/execute/this/script.sh*

Em que cada asterisco, ou estrela (*), significa respectivamente, minutos (de 0 a 59), horas (de 0 a 23), dias do mês (de 1 a 31), mês (de 1 a 12) e dia da semana (de 0 a 6 sendo que '0' significa domingo). Designam o tempo entre verificações.

3.2.5 *Preview* utilizando *Blender*

Na página do site onde são realizados os envios dos arquivos para serem impressos, foi destinada uma área onde seriam exibidas imagens ilustrativas dos objetos recebidos.

Para este fim, foi utilizado o *software* de modelagem 3D *Blender*, que oferece uma API completa em *Python*. É possível executar o *Blender* com um *script* em *Python* que processa o arquivo *.stl* (escala, reposiciona, define materiais e iluminação), e renderiza a cena com ótima qualidade. Este processo é executado toda vez que um novo arquivo é enviado ao servidor, e leva alguns segundos para ser concluído. O resultado pode ser conferido na figura 3.4.

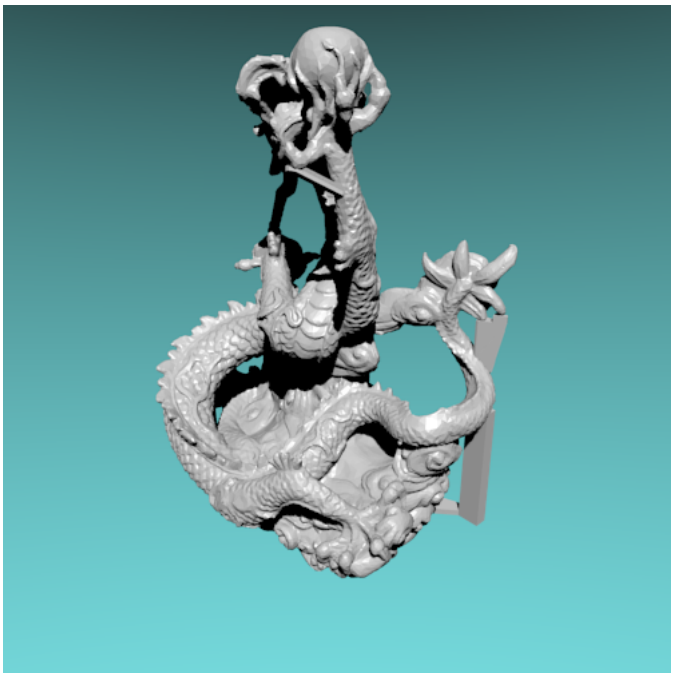


Figura 3.4: *Preview* obtido com a utilização do *Blender*.

3.2.6 *Slicing* e Matriz de Adjacência

O formato *.stl* descreve um objeto tridimensional como um conjunto de triângulos. Se a altura máxima e mínima contêm a altura do plano, então o triângulo é decomposto em 3 linhas e é analisado separadamente a interseção.

Se a linha interceptar o plano, retorna um ponto; se a linha for paralela e coincidente ou dentro de um erro aceitável ligada ao passo, retorna 2 pontos. Se for paralela e fora deste erro, retorna vazio.

Todos os pontos são armazenados em duas matrizes: uma que armazena pontos, e outra que armazena os tipos ligações. Em sequência, diminui-se redundância. Se apresentar 0 ou 1 ponto único é descartado, não é útil para a impressão; se 2 pontos únicos, é descrito como um segmento; e se 3 ou 4 pontos únicos, é descrito como um triângulo ou quadrilátero de casca.

Após analisar todos os triângulos, todos os pontos possíveis com suas ligações armazenadas, as ligações são convertidas para para uma matriz de adjacência e os pontos são reduzidos para apenas os pontos únicos. O código utilizado encontra-se como anexo.

3.2.7 Algoritmo de Bresenham

O algorítimo de Bresenham é uma versão que calcula nos 4 quadrantes matemáticos, e que a linha produzida parte do ponto inicial ao ponto final, diferente do código tradicional, utilizado para imagens, que a direção não importa. O código Bresenham, apenas para o primeiro quadrante é:

```
def bresenham(x1,y1,x2,y2,z,g,plastico,t_plastico,t):
    a = y2-y1
    b = x1-x2
    V = 2*a+b
    x = x1
    y = y1
    while x<=x2:
        plastico += t_plastico
        l = 'G1 X' + str(x) +  " Y" + str(y) +  " Z" + str(z) +  " E" + str(plastico)
        f.write(l +  '\n')
        x = x+1
        if V<=0:
            V = V+2*a
        else:
            v = V+2*(a+b)
            y = y+1
    return plastico
```

Para o cálculo dos demais quadrantes, multiplicam-se duas variáveis "sx"e "sy", de valor unitário com sinal, relativos ao quadrante para o calculo. São multiplicados as variáveis "x"e "y"para garantirem a direção, e no resultado, para garantir fidelidade aos pontos final e inicial. Também é necessário conferir se há variação em "x", já que este código funciona até que "x1"seja igual a "x2". O resultado que é descrito como uma *string*, com os valores de "x", "y", "z"e "e", este ultimo relativo a extrusora.

3.2.8 Algoritmo de produção de linha vetorial

O algorítimo de Bresenham apresenta a desvantagem de desenhar a linha em formato matricial, o que gera diversas linhas que se movem apenas no eixo x ou apenas no eixo y. Como a impressora apresenta um movimento vetorial para os eixos X e Y, foi modificado o código para suavizar o efeito serrilhado. O código utilizado, apenas para o primeiro quadrante é:

```
def linha_simples(xa1,ya1,xa2,ya2,za,g,plastico,t_plastico,t):
    xa = xa1
    ya = ya1
    while xa <= xa2:
        m = (xa2-xa1)/(ya2-ya1)
```

```
        xa = xa1
        ya = ya1
        while ya <= ya2:
            plastico += t_plastico
            l = 'G1 X' + str(s2*xa) +  " Y" + str(s2*s1*ya) +  " E" + str(plastico)
            print 'Sending: ' + l + ' \n',
            f.write(l +  '\n')
            ya = ya+t
            xa = xa+t*m
    return plastico
```

A diferença entre esse código e o de Bresenham é que X e Y são incrementados simultaneamente. Para os demais quadrantes, utiliza-se as variáveis "sx"e "sy"da mesma forma que o código de Bresenham. O resultado é uma *string* contendo o *G-code* do movimento e gasto do material relativo ao movimento.

3.2.9 Manutenção da Impressora

Sempre que a impressora for ligada deve ser feita a nivelação da plataforma principal que é sustentada por dois parafusos sem fim em dois motores nas laterais e dois trilhos na parte traseira da mesma, como não possui sustentação na parte frontal a plataforma tende a desnivelar neste lado. Através de um *software* com rotinas de comando é possível realizar esse trabalho, no qual a cabeça extrusora encontra três pontos da mesa para formar um plano e faz a ajuste grosso e fino do movimento dos motores, utilizando uma folha de papel sulfite comum é subida a plataforma até o mesmo ficar preso, e então é soltado, e assim é feito para três níveis de ajuste em três posições diferentes, caso haja algum desnivelamento deve-se usar os parafusos para o nivelamento.

Pelo modelo ser antigo a impressora possui a sua câmara de impressão totalmente exposta ao ambiente nas laterais e na parte superior, o que aumenta o envergamento dos objetos devido ao fluxo de ar, então recomenda-se o uso de estruturas, como vidro, ou até mesmo papelão, para tampar essas aberturas. Recomenda-se também a não utilização de ventiladores ou ar-condicionados durante a impressão, que podem acarretar em variações na temperatura.

A impressora é alimentada para impressão por um sistema que empurra o filamento de plástico, localizado na parte traseira. Se o fio se romper e a cabeça extrusora estiver entupida, ou sua temperatura muito baixa, a partir de um certo momento, o mecanismo não será capaz de continuar empurrando o filamento. Por isso deve-se observar sempre a integridade do fio e a temperatura. Caso o mesmo

tenha se rompido, é necessário somente que se introduza a outra ponta em seguida.

Se o material não colar e envergar, o mesmo continuará sendo carregado pela extrusora, que pode ocasionar na queima do plástico e estragar toda a impressão. Se o material derreter e ficar muito tempo preso na extrusora, pode ocorrer entupimento, então um agente dissolvente deve ser usado para a limpeza da mesma, como acetona.

Antes de cada impressão deve ser feita a limpeza do vidro da cama, com álcool ou acetona, para remover restos do plástico e de partículas de poeira.

3.2.10 Serial em python

O arquivo foi criado puramente para testar a comunicação serial com a impressora, baseado no código do *stream.py* do *grbl*. Foi utilizado um arquivo *.txt*, arquivo contendo G-code, gerado pelos caminhos encontrados. Importando "serial", pode-se associar um valor s com os seguintes dados: a porta USB ('/dev/ttyACM0') e velocidade de informação, *Baud Rate*, de 115200 bald/s, e o processo de iniciação serial.

O comando utilizado para esse feito foi:

```
s = serial.Serial(  '/dev/ttyACM0',115200)
s.write( "\r\n\r\n")
time.sleep(2)   #espera para iniciação
s.flushInput()   #inicia o texto de incialização serial
```

Para impressão, foi lida cada linha do arquivo *.txt* e enviada para a impressora. Após isso, é esperado uma resposta do destinatário a para o envio da próxima linha. Utilizou-se o seguinte *loop*:

```
for line in f:
^^Il = line.strp()
    print  'Enviando: ' + l,
    s.write(l +  '\n') #envio do bloco de G-code
    grbl_out = s.readline()    #espera da resposta da impressora
    print  ' : '
```

Capítulo 4

Resultados

Todos os códigos do trabalho (incluindo os que não foram incluídos neste documento) podem ser encontrados no *GitHub* através do *link*: `https://github.com/Plezito/TCC_fuji_plez` .

4.1 *Site*

Para o projeto foi criado um *site*, acessível pelo *link* `asap3d.tk` , no qual as pessoas podem criar usuários e gerenciar sua conta, e então enviar seus arquivos *.stl* contendo o objeto 3D desejado para a impressão. Depois de recebido o *site* fornecerá uma pré-visualização do objeto para o usuário. O arquivo então entra numa fila e automaticamente é posto para impressão se for o mais recente. Os administradores podem cancelar a impressão a qualquer momento pelo próprio *site*.

Foram criados bancos de dados para armazenar as informações dos usuários e das impressões como visto no capítulo de métodos.

4.1.1 *Front-end*

Página Inicial

Podemos notar na página inicial detalhes como os textos e *links*, além do *header* e *footer* que são padrões para todas as outras páginas. Sem *login*, o primeiro exibe os *links* para o *GitHub*, a página 'Sobre' e a de *'Login'*. O segundo sempre irá mostrar os *links* para os *sites* relevantes da escola, para os termos de uso, além de um *e-mail* para contato.

Na figura 4.1, que representa a página inicial, pode-se notar informações sobre o projeto e seu estado atual, além de *links* para o cadastro e *login*.

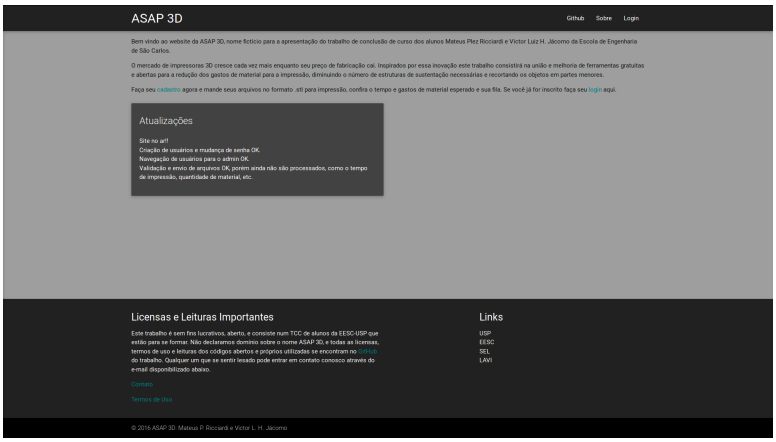


Figura 4.1: Página inicial

Página 'Sobre'

Vários cartões na página que mostram informações sobre os alunos, o professor e algumas ferramentas utilizadas, como pode ser observado na figura 4.2.

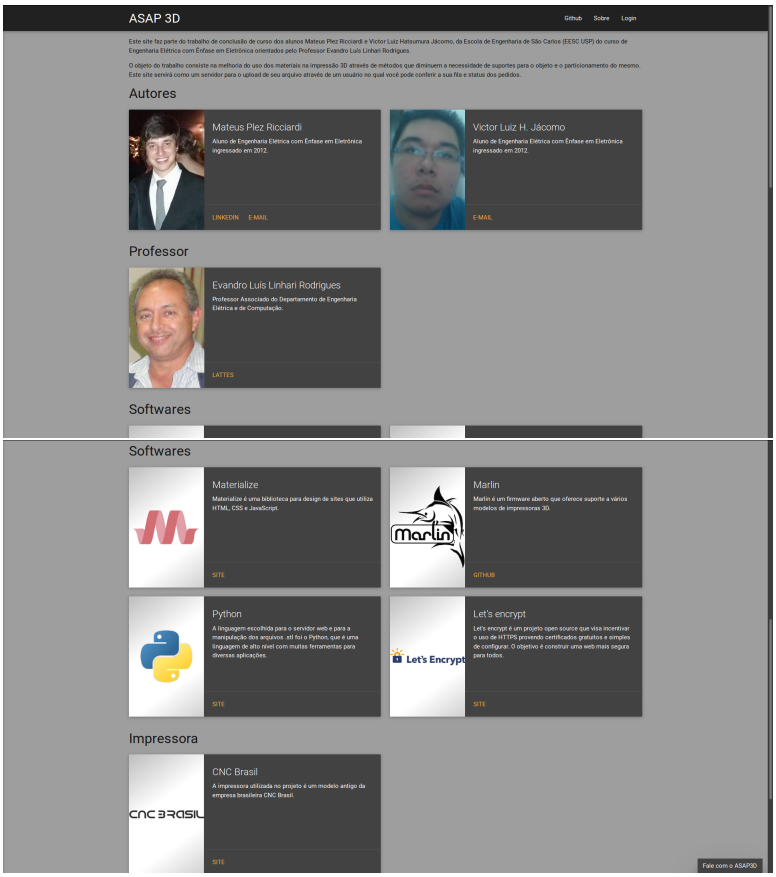


Figura 4.2: Página 'Sobre'.

Página de Registro

Um cartão com um formulário de registro com informações básicas necessárias para a inscrição, com *link* para os termos de uso da página. Só pode terminar o cadastro caso concorde com os termos. Sua aparência pode ser vista em 4.3.

O cadastro do usuário é salvo em uma tabela, com a sua senha protegida por um método de encriptação.

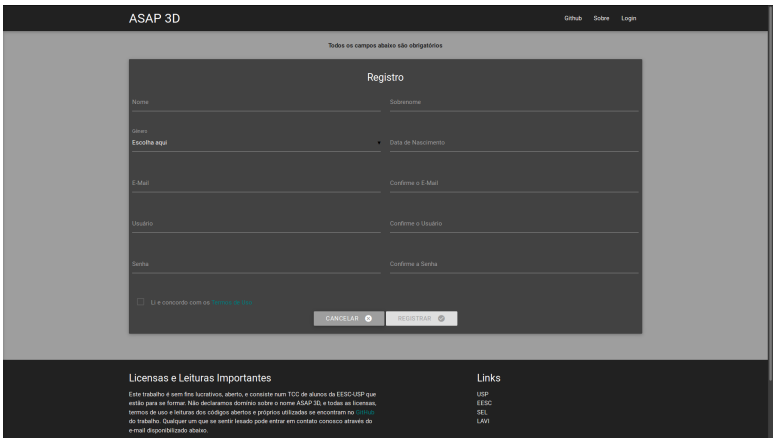


Figura 4.3: Página de registro

Página de *Login*

Um cartão com um formulário para acessar as áreas de *print* e administração caso tenha permissão. Podendo também ser feito o *login* e registro pelo *e-mail* do *Google*, além de um *link* para a recuperação da senha. Depois do *login*, se for válido, o *header* mudará e passará a possuir as opções de seu perfil e administração, caso tenha permissão. A página pode ser conferida na figura 4.4.

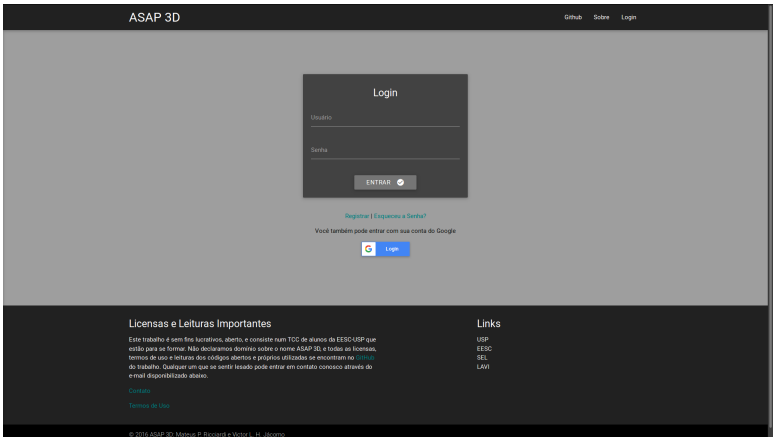


Figura 4.4: Página de *login*

Página de Impressão

Página contendo uma área de envio de arquivos do tipo *'.stl'*, no qual o limite de tamanho de envio é de 50MB, no topo. Abaixo possui um cartão com os *previews* dos arquivos enviados e selecionados, podendo, através do botão, mudar o *status* da sua impressão para 'cancelado'.

Ao lado do cartão de *previews*, caso o usuário possua algum item com o *status 'printing'*, uma janela se abrirá ao lado com acesso a câmera. Como por motivos técnicos, a câmera do laboratório não pode ser acessada, então foi utilizada a câmera local de um dos alunos para demonstrar.

Abaixo do cartão temos a fila e histórico de suas impressões com *links* para as imagens de *preview* das mesmas com os *status* e informações de cada uma.

Na figura 4.5 pode-se observar no topo a página de impressão sem a câmera, ou seja, sem nenhuma impressão acontecendo no momento, e abaixo está a página com um objeto imprimindo, o que liga a câmera para observação.

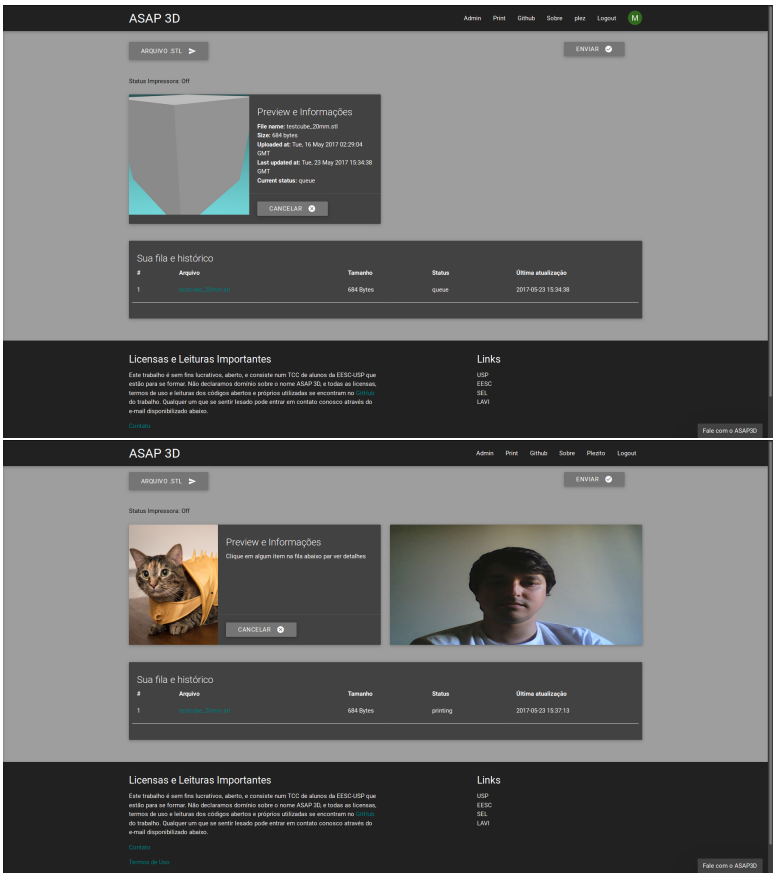


Figura 4.5: Página de impressão

Página de Administração

Se o usuário possuir permissão de administrador essa página se tornará acessível e nela são encontradas todas as informações de todos os usuários cadastrados, e suas respectivas permissões.

O primeiro cartão possui as impressões no banco de dados, com seus *status* e informações, podendo o primeiro ser alterado através dos botões.

Já o segundo cartão possui os usuários no banco de dados, com *status*, ou nível de acesso, e ao lado, os botões para mudança de nível ou de bloqueio e exclusão podem ser acessados.

Na figura 4.6 pode ser conferida a página de administração, no topo a primeira parte da mesma, e logo abaixo a segunda parte.

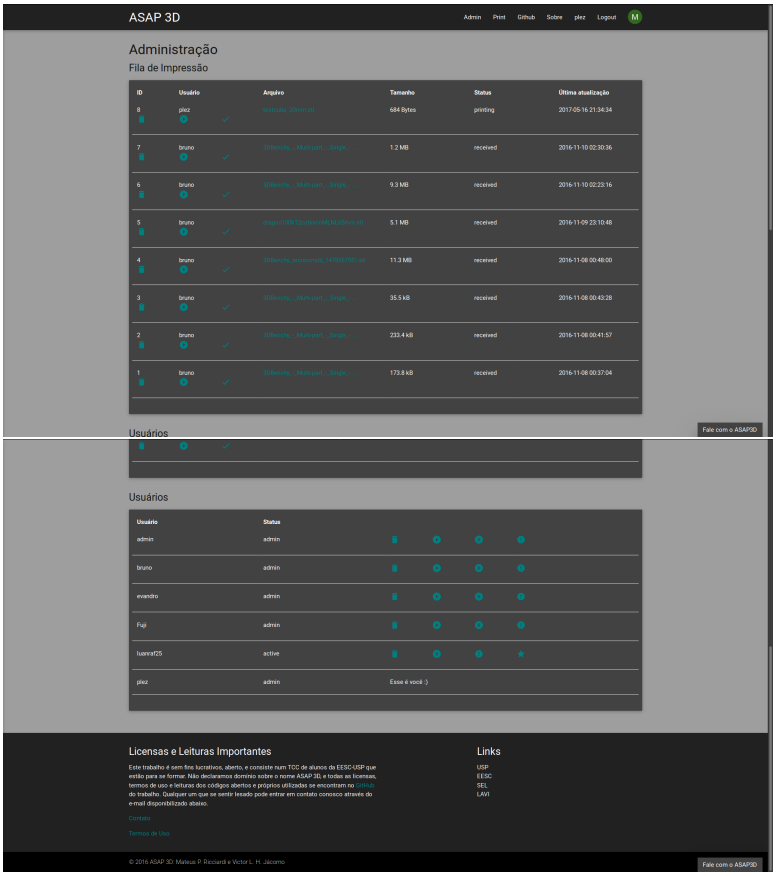


Figura 4.6: Página de administração

Página de Troca de Senha

Nesta página, caso esteja logado, será necessário somente confirmar a sua data de nascimento para realizar a troca de senha, se não estiver logado será necessário acrescentar todos os dados pedidos. Esta página permite que os usuários recuperem suas senhas, caso seja necessário. Ela pode ser conferida na figura 4.7.

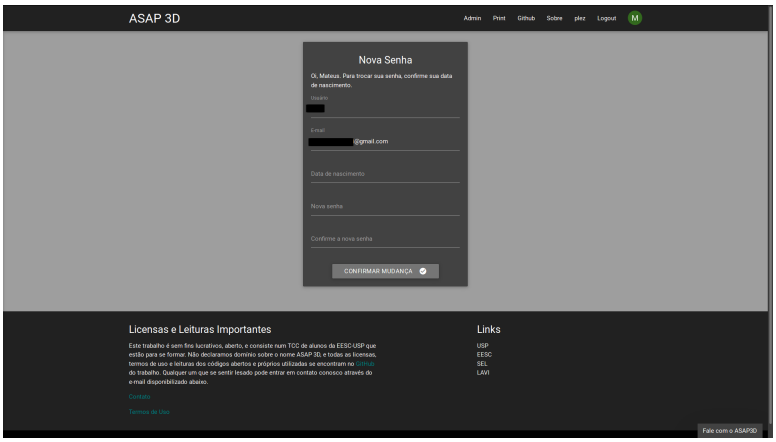


Figura 4.7: Página de troca de senha

Chat

Este recurso permite a conversa dos usuários com os administradores e entre os próprios administradores. A caixa de diálogo pode ser observada na figura 4.8.

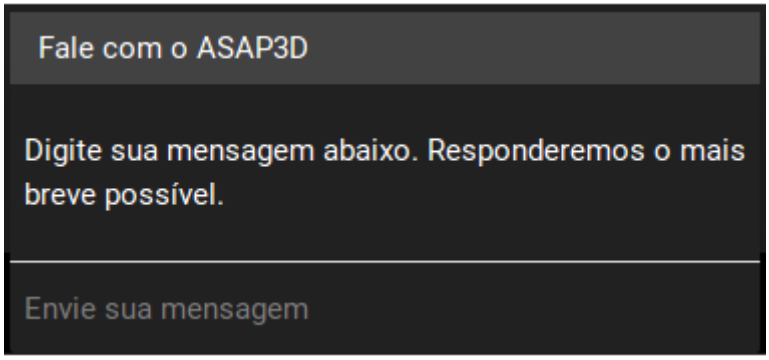


Figura 4.8: Chat exemplo de um administrador mandando mensagem para o usuário 'plez'

4.2 Comunicação Serial e Impressão

4.2.1 Código de impressão

Foi possível criar um código que retira as informações do um arquivo *.stl*, realiza o corte do objeto e criar um arquivo *.txt* capaz de armazenar os arquivos g-code gerados. A transmissão do código para impressora foi possível, apesar que não foi feito um código de verificação de condições, apenas para ver se há alguma resposta da impressora, o que pode gerar problemas quando recebidas mensagem de erro.

Para demonstrar a criação de linhas, tem-se os códigos de matriz de adjacência da equação (4.2) e da matriz contendo os pontos da equação (matrizdepontos). A partir destas duas matrizes, retiradas na

confecção da primeira camada da pirâmide, foi feito os três processos possíveis: A criação do Contorno, a criação do contorno com preenchimento simples e a criação do contorno com preenchimento mais sólido.

Matriz de Pontos sem redundância:

$$\begin{pmatrix} -125. & 105.63160706 \\ -125. & 148.05801392 \\ -82.57359314 & 105.63160706 \\ -82.57359314 & 148.05801392 \end{pmatrix} \tag{4.1}$$

Matriz de Adjacência

$$\begin{pmatrix} 0. & 1. & 1. & 1. \\ 1. & 0. & 0. & 1. \\ 1. & 0. & 0. & 1. \\ 1. & 1. & 1. & 0. \end{pmatrix} \tag{4.2}$$

Caminho gerado:

$$\begin{pmatrix} 1 & 2 & 4 & 1 & 3 & 4 & 0 & 0 & 0 \end{pmatrix} \tag{4.3}$$

Com o caminho gerado em (4.3), foi possível gerar a figura 4.2.1.

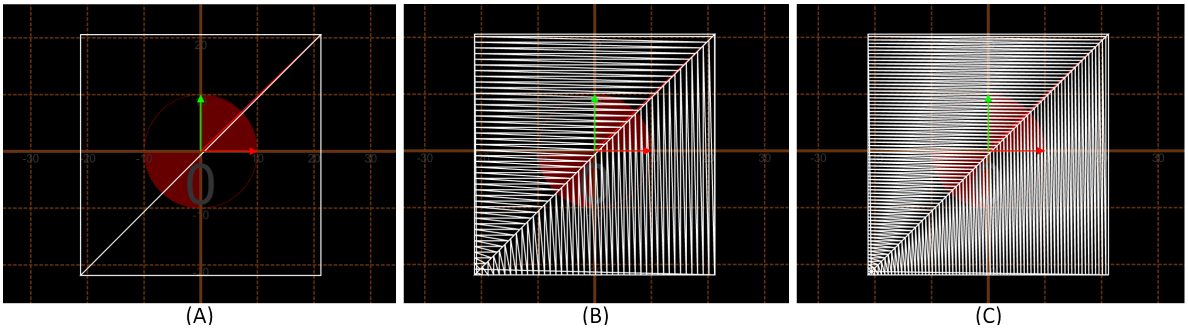


Figura 4.9: Imagens formada da base da pirâmide. (A) Sem *infill*. (B) Com *infill* para preenchimento. (C) Com *infill* para casca

O código teste foi com base em uma pirâmide, pela simplicidade geométrica e um iglu, pela sua característica circular. O iglu pode ser visto na figura . Esta representação apresenta apenas a área externa, visto que a imagem seria ainda menos detalhada se tivesse o preenchimento. Para facilitar a visão do processo de preenchimento, será demonstrado a pirâmide com fatias espaçadas, visível na figura . Nota-se uma diferença entre o preenchimento da base com as demais fatias. Isto ocorre pois o preenchimento é diferente entre o casca e *infill*, sendo a casca mais densamente preenchida, como já demonstrado na figura .

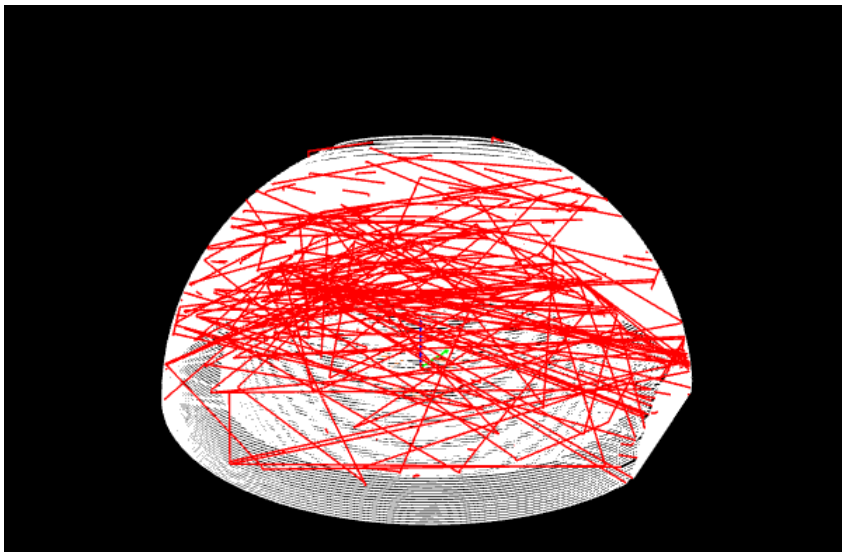


Figura 4.10: Imagens formadas do Iglu

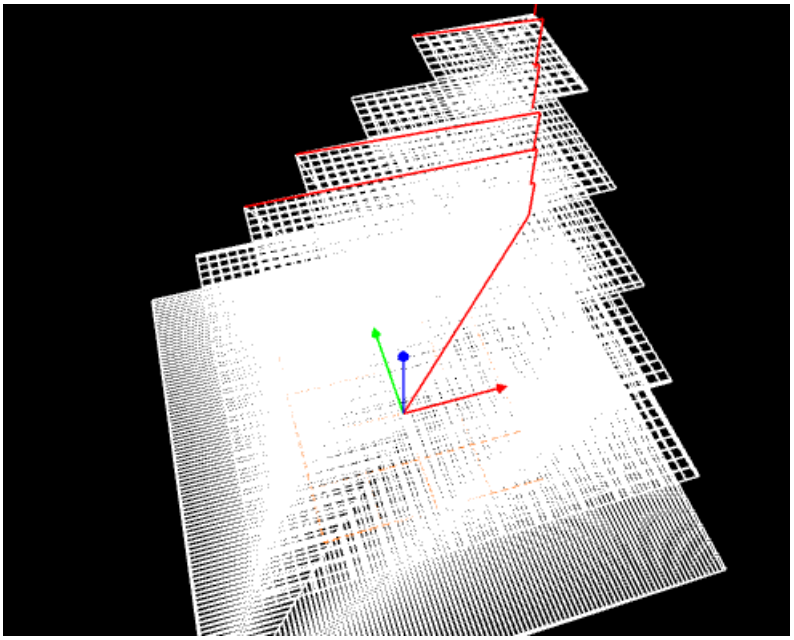


Figura 4.11: Camadas da pirâmide. Nota-se que há diferenciação entre a Casca (base da figura) e as demais camadas.

4.2.2 Código de transmissão

O código de transmissão foi feito em *Python*, utilizando a biblioteca *serial*. Após a todo o processo anterior ser realizado, é formado um arquivo formato *.txt*. O código de transmissão consiste unicamente no envio deste *.txt* para a impressora. O código analisa apenas se a impressora responde, o que pode causar problemas se o correr um erro.

No teste final, foi possível a criação de um canal de comunicação e envio de informações ligadas a movimento e controle de temperatura.

Capítulo 5

Conclusão

Com o desenvolvimento do projeto, foi visto a grande complexidade do mesmo. Todavia, foi possível o desenvolvimento do site e do algoritmo de interpretação de arquivos *.stl*.

O *site* apresenta uma interface minimalista e intuitiva, e com um sistema de comunicação para suporte. Em relação ao código de geração, foi possível a interpretação das figuras tridimensionais em um domínio bidimensional, o que potencialmente aumenta a velocidade de processamento, apesar de ter sido usada a linguagem de programação *Python*, que por ser interpretada, é mais lenta em comparação a linguagens compiladas, como, por exemplo, *C*, *C++* ou *Java*. Vale ressaltar que foi possível a criação de preenchimento e diferenciação entre exterior e interior.

A comunicação com a impressora foi possível através de um *script* em *python*. O sistema envia informações em G-code e espera a resposta da impressora.

Referências

[1] Mercado de impressoras 3d para consumidores comuns cresce. https://3dprint.com/128648/global-3d-printer-market-up/ , Acesso em: 30 de Novembro de 2016.

[2] Mercado de impressoras 3d em 2022. http://www.marketsandmarkets.com/PressReleases/3d-printing.asp , Acesso em: 30 de Novembro de 2016.

[3] Impressões 3d na engenharia civil. https://3dprintingindustry.com/news/top-10-3d-printed-construction-innovations-83578/ , Acesso em: 30 de Novembro de 2016.

[4] Esa 3d prints sturdy bricks from sunlight and moon dust. https://3dprint.com/173267/3d-printed-bricks-sunlight-moon/ , Acesso em: 26 de Maio de 2017.

[5] Impressões 3d na engenharia biomédica. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4189697/ , Acesso em: 30 de Novembro de 2016.

[6] Hideo Kodama. Automatic method for fabricating a three-dimensional plastic model with photo-hardening polymer, 1981.

[7] Chuck hull primeira impressão estereográfica. http://www.pcmag.com/slideshow_viewer/0,3253,l=293816&a=289174&po=1,00.asp , Acesso em: 30 de Novembro de 2016.

[8] Imagem modificada de histórico impressão 3d e seus tipos. https://www.3dhubs.com/what-is-3d-printing , Acesso em: 30 de Novembro de 2016.

[9] Rapid prototyping. http://www.eastbrancheng.com/rapid-prototype/ , Acesso em: 30 de Maio de 2017.

[10] 3d printing technology comparison: Sla vs. dlp. https://formlabs.com/blog/3d-printing-technology-comparison-sla-dlp/ , Acesso em: 30 de Maio de 2017.

[11] Here come the cheaper sls 3d printers, but they will still cost you. https://gigaom.com/2014/08/19/here-come-the-cheaper-sls-3d-printers-but-they-will-still-cost-you/ , Acesso em: 30 de Maio de 2017.

[12] Stratasys makes 3d-printed objects pop. http://www.technewsworld.com/story/79862.html , Acesso em: 30 de Maio de 2017.

[13] Additive elements develops 1003d printing. http://www.3ders.org/articles/20160427-additive-elements-develops-eco-friendly-material-system-for-binder-jetting-3d-printing.html , Acesso em: 30 de Maio de 2017.

[14] Tiny metal 3d printing overview: Mica freeform vs. slm. https://3dprintingindustry.com/news/metal-3d-printing-mica-freeform-39146/ , Acesso em: 30 de Maio de 2017.

[15] What is abs plastic? http://www.iosp.org/what-is-abs-plastic/ , Acesso em: 30 de Maio de 2017.

[16] https://cheap3dfilaments.com , Acesso em: 30 de Maio de 2017.

[17] K. F.; Lim C. S. Chua, C. K; Leong. Rapid prototyping: Principles and applications, segunda edição. World Scientific, 2003.

[18] http://cdn.shopify.com/s/files/1/0518/7929/products/Emendo_logo_2_de6ad830-1dc8-4788-995d-cda997332369_large.png?v=1415654665 , Acesso em: 30 de Maio de 2017.

[19] I. Boulos, P.; de Carmargo. Geometria analítica: Um tratamento vetorial. In: MAKROM Books do Brasil Editora Ltda.

[20] D Sunday. Intersection of rays and triangles. http://geomalgorithms.com/a06-_intersect-2.html/ , Acesso em: 20 de Maio de 2017.

[21] RAVEH GONEN'S BLOG.

[22] User:Chris-martin. Small undirected graph. https://commons.wikimedia.org/wiki/File:6n-graph2.svg , Acesso em: 20 de Maio de 2017.

[23] User:Baruneju. Ricerca in profondità. https://commons.wikimedia.org/wiki/File:Depth_first_search.svg , Acesso em: 20 de Maio de 2017.

[24] User:Alexander Drichel. Order in which the nodes get expanded. https://upload.wikimedia.org/wikipedia/commons/5/5d/Breadth-First-Search-Algorithm.gif , Acesso em: 25 de Maio de 2017.

[25] P. E. Black. National institute of standards and technology. https://xlinux.nist.gov/dads/HTML/bresenham.html , Acesso em: 20 de Abril de 2017.

[26] User:Crotalus horridus. A diagram illustrating bresenham's line algorithm. https://en.wikipedia.org/wiki/File:Bresenham.svg , Acesso em: 20 de Maio de 2017.

[27] Computer Science at Brown University.

[28] Slic3r manual. http://manual.slic3r.org/expert-mode/images/infill_rectilinear.png , Acesso em: 20 de Maio de 2017.

[29] Raspberry pi. https://www.raspberrypi.org/ , Acesso em: 30 de Novembro de 2016.

[30] Raspberry pi model b. https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ , Acesso em: 30 de Novembro de 2016.

[31] Marlin firmware. http://reprap.org/wiki/Marlin , Acesso em: 30 de Novembro de 2016.

[32] Reprap. http://reprap.org/ , Acesso em: 30 de Novembro de 2016.

[33] *3D MATTER* em associação com *Arts et Métiers Paristech*.

[34]

[35] W. Wang; C. Zanni; L. Kobbelt. Histórico impressão 3d e seus tipos. https://www.graphics.rwth-aachen.de/media/papers/EG_Printing_Segmentaton.pdf , Acesso em: 30 de Novembro de 2016.

[36] F. ROGER and P. KRAWCZAK. 3d-printing of thermoplastic structures by fdm using heterogeneous infill and multi-materials: An integrated design-advanced manufacturing approach for factories of the future. In: CONGRÈS FRANÇAIS DE MÉCANIQUE, 22., 2015. Lyon.

Anexo 1: Métodos de Otimização

Existem várias formas de analisar a qualidade de uma impressão: peso, custo, resíduo, velocidade e resistência. É impraticável a otimização de todas as áreas ao mesmo tempo, sendo inevitável a análise com base na relevância de cada uma dessas características.

Otimização de Espessura de camadas de Plástico

A partir dos estudos realizados pela *3D MATTER*, em associação *Arts et Métiers ParisTech*, foram obtidas analises de resistência, custo, velocidade e qualidade em relação a altura da camada, a partir de diferentes espessuras e porcentagens de preenchimento de camadas [33]. O resultado dos estudos pode ser conferido na tabela 1. A tabela demonstra as melhores opções para os parâmetros desejados individualmente ou relacionados, marcando-se "x"caso a opção desejada seja cumprida.

REQUISITOS				OPÇÕES	
Resistência	Qualidade	Baixo Custo	Velocidade	Preenchimento (%)	Altura por Camada
✖				100	0.25
	✖			10	0.1
		✖		10	0.1
			✖	10	0.3
✖	✖			90	0.15
✖		✖		70	0.2
✖			✖	90	0.3
	✖	✖		10	0.1
	✖		✖	10	0.15
		✖	✖	10	0.3
✖	✖	✖		80	0.15
✖	✖		✖	90	0.2
✖		✖	✖	70	0.3
	✖	✖	✖	10	0.15
✖	✖	✖	✖	70	0.2

Figura 1: Teste de força, custo, velocidade e qualidade geral, em função da porcentagem e espessura do material utilizado [33]

.

Também foram analisadas para diferentes padrões de camadas, todos com o mesmo tipo de material, espessura e porcentagem de preenchimento. Os padrões testados foram linear, diagonal, hexagonal, estrela marroquina e preenchimento com gatos (*Catfill*). Pelos testes do estudo, os três padrões mais fortes são linear, diagonal e hexagonal.

Para uma análise de 10%, linear apresentou 10% mais forte que os outros dois. O teste não é conclusivo pra outras porcentagens de plástico, visto que outro teste realizado com apenas linear e diagonal a 100% de preenchimento, o diagonal apresentou-se 10% mais forte.

Não houve diferença significativa no teste de elasticidade para os três principais materiais. Suportam aproximadamente 2% antes de quebrarem.

Apesar de a pesquisa ter sido feita com o material PLA, é possível extrair diversos pontos de interesse, como comportamento para diferentes espessuras e porcentagem de preenchimento, que podem ser de grande ajuda.

Segmentação

O Packmerger ou Segmentação [34] é um método mais rápido, econômico e com baixo resíduo. Este método possui o intuito de diminuir a necessidade vigas de sustentação em todo interior, subdividindo o objeto em partes imprimíveis.

Inicialmente, é necessário que a figura apresente apenas a sua superfície. Não é desejado que haja vigas de sustentação internas e externas. É possível criar um algorítimo que conserte este problema, mas como nos outros métodos que serão descritos, será considerada uma figura com apenas área superficial. A partir deste objeto, é feito um processo de ampliação da espessura da casca, para que permita sustentação por si própria.

A segmentação é realizada por *Seeds*, células de área aleatoriamente escolhidas, que são expandidas por células próximas até que se forme o *"cluster"*, um tamanho de volume pré-definido. O processo continua de novas *seeds* até que toda a área do objeto seja ocupada por *"clusters"*.

O processo une esses *"clusters"* em função da proximidade para evitar uma quantidade excessivamente grande de partes. Então se preciso o algoritmo analisa a necessidade pinos para o encaixe das partes.

A figura 2 mostra um diagrama dos processos para esse tipo de otimização.

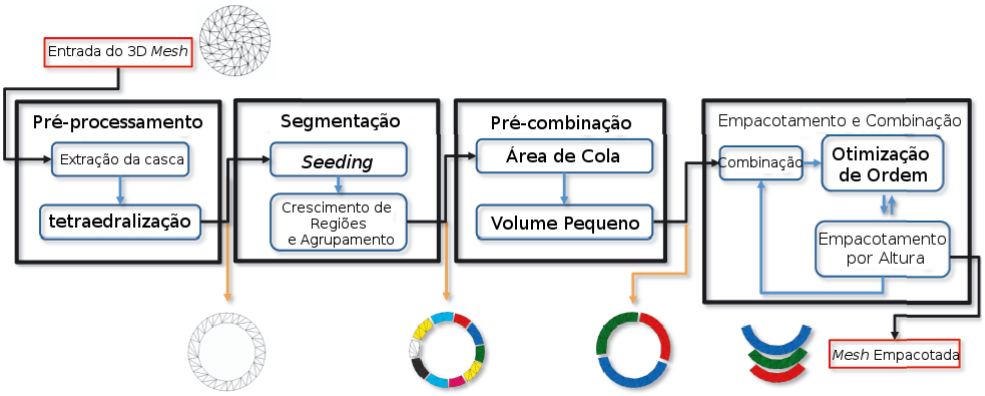


Figura 2: Processo de fortalecimento da casca, segmentação e corte[34]

Além disso, foi acrescentado posteriormente ao algoritmo, um método de rotação desses *"clusters"* para que veja qual deles terá uma menor necessidade de suporte, aumentando ainda mais a economia de material, porém é perdido tempo de processamento.

Outro trabalho que envolve a segmentação de objetos 3D, envolve em achar partes dos mesmos que já sejam perpendiculares em relação à direção de impressão, otimizando o processo, pois quanto mais vertical, menos estruturas são necessárias para a sua sustentação [35]. O resultado desse tipo de segmentação pode ser conferido em 3.

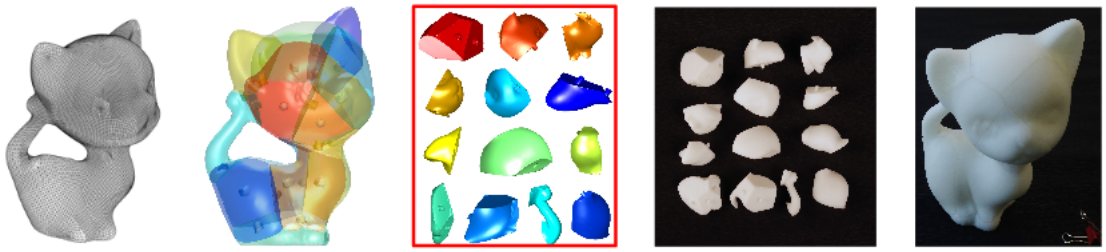


Figura 3: Processo de segmentação de um objeto 3D utilizando o método de perpendicularidade [35]

Preenchimento Heterogêneo

O objetivo desta otimização é minimizar o peso total com a distribuição de material, de forma que maximize a rigidez. No exemplo utilizado, visto na figura 4, foi construída uma estrutura de vigas a partir do *software Comsol Multiphysics*©[36]. Após o cálculo do estresse sofrido em cada região, foi utilizado 60% de preenchimento para as regiões de alto estresse, devido a característica de peso-rigidez dessa porcentagem, e 20% para as regiões de menos necessidades estruturais.

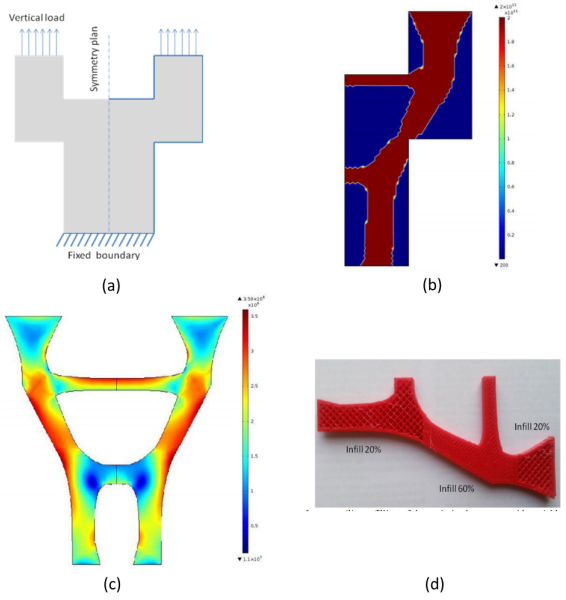
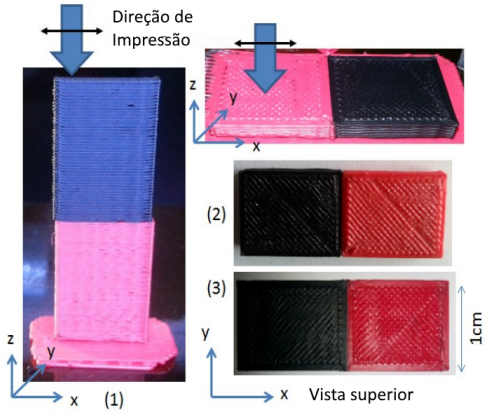


Figura 4: (a)Condição inicial e de contorno do objeto; (b)Estrutura interna com formato otimizado; (c)Estesse em pascal da estrutura interna; (d)Resultado impresso. [36].

A direção dos preenchimentos numa impressão também é importante, pois dependendo da direção das forças aplicadas no objeto, pode ocasionar no mesmo em se quebrar ou desmanchar. Por isso geralmente é feito a impressão de cada camada com direções diferentes. Sendo que o mais horizontal-

mente possível a impressão, melhor, de acordo com o mesmo trabalho. Na figura 5 mostra exemplos de impressões em determinadas direções e suas consequências com relação à força da estrutura criada.



	Dois materiais			Um material	
Configuração da Amostra	1	2	3	ABS preto	ABS vermelho
Posição durante Impressão	vertical	horizontal	horizontal	vertical	horizontal
Resistência à tração	3,2	frágil	6,24	10,55	23

Figura 5: Direções de impressão e os resultados dos testes de resistência. [36].

Anexo 2: Código de processamento e geração de GCODE

```
import numpy as np
import sys
from stl import mesh
from numpy import size
from mecode import G
import time
import mecode.printer
import serial

def env_serial(s):
    f = open( 'in.nc','r')
    for line  in f:
        l = line.strip()
            print 'Sending: ' + l,
            s.write(l +   '\n')
            s.flushInput()
            grbl_out = s.readline()
            print ' : ' + grbl_out.strip()
    f.close()
    s.close()

def bresenham(x1,y1,x2,y2,z,g,plastico,t_plastico,t):
    s2 = 1
    if x2<x1:
```

```
        s2 = -1
        y2 = -y2
        x2 = -x2
        y1 = -y1
        x1 = -x1
    s1 = 1
    if y2<y1:
        y1 = -y1
        y2 = -y2
        s1 = -1
    if abs(y2-y1)<abs(x2-x1):
        a = y2-y1
        b = x1-x2
        V = t*(2*a+b)
        x = x1
        y = y1
        while x<=x2:
            plastico = plastico + t_plastico
            l = 'G1 X' + str(s2*x) +  " Y" + str(s2*s1*y) +  " Z" + str(z) +  " E" + str(plasti
            print 'Sending: ' + l + ' \n',
            f.write(l +  '\n')
            x = x+t
            if V<=0:
                V = V+2*a*t
            else:
                V = V+2*(a+b)*t
                y = y+t
        else:
                a = x2-x1
                b = y1-y2
                V = (2*a+b)*t
                x = x1
                y = y1
                while y<=y2:
```

```
            plastico = plastico + t_plastico
            l = 'G1 X' + str(s2*x) +  " Y" + str(s2*s1*y) +  " Z" + str(z) +  " E" + str(plasti
            print 'Sending: ' + l + ' \n',
            f.write(l +  '\n')
            y = y+t
            if V<=0:
                V = V+(2*a)*t
            else:
                V = V+2*(a+b)*t
                x = x+t
    return plastico

def linha_simples(xa1,ya1,xa2,ya2,za,g,plastico,t_plastico,t):
    s2 = 1
    if xa2<xa1:
        s2 = -1
        ya2 = -ya2
        xa2 = -xa2
        ya1 = -ya1
        xa1 = -xa1
    s1 = 1
    if ya2<ya1:
        ya1 = -ya1
        ya2 = -ya2
        s1 = -1
    if ya2-ya1 == 0:
        xa = xa1
        ya = ya1
        while xa <= xa2:
            plastico = plastico + t_plastico
            l = 'G1 X' + str(s2*xa) +  " Y" + str(s2*s1*ya1) +  " E" + str(plastico)
            print 'Sending: ' + l + ' \n',
            f.write(l +  '\n')
            xa = xa+t
```

```
        else:
            m = (xa2-xa1)/(ya2-ya1)
            xa = xa1*1
            ya = ya1*1
            while ya <= ya2:
                plastico = plastico + t_plastico
                l = 'G1 X' + str(s2*xa) +  " Y" + str(s2*s1*ya) +  " E" + str(plastico)
                print 'Sending: ' + l + ' \n',
                f.write(l +  '\n')
                ya = ya+t
                xa = xa+t*m
    return plastico

def infill_sem(pontos,matriz_ad,z,plastico,t_plastico,t):
    v = np.array([])
    p1 = np.array([])
    p2 = np.array([])
    cont = 1
    for n in range (0,len(matriz_ad)):
        for m in range (n+1,len(matriz_ad)):
            if matriz_ad[m][n] == 1:
                tmp_v = pontos[n] - pontos[m]
                tmp_p1 = pontos[n]
                tmp_p2 = pontos[m]
                v.resize(cont,2)
                p1.resize(cont,2)
                p2.resize(cont,2)
                v[cont-1] = tmp_v
                p1[cont-1] = tmp_p1
                p2[cont-1] = tmp_p2
                cont+=1
    return v,p1,p2

def intersec_vertical_ponto(v,p1,p2,n,x):
```

```
    vetor = [0,1]
    determinante = np.linalg.det ([v[n],vetor])
    minimo = np.amin([p1[n][0],p2[n][0]])
    maximo = np.amax([p1[n][0],p2[n][0]])
    if determinante != 0   and maximo>=x  and minimo<=x:
        y = (v[n][1]/v[n][0])*(x-p1[n][0])+p1[n][1]
        return np.array([x,y])
    return np.array([])

def preenchimento_vertical(v,p1,p2,pontos,z,plastico,t_plastico,t,passos):
    count = 0
    pontos_vertical = np.array([])
    maximox = np.amax(pontos)
    minimox = np.amin(pontos)
    xx = minimox
    while xx <= maximox:
        for n in range (0,size(p1)/2):
            ipvertical = intersec_vertical_ponto(v,p1,p2,n,xx)
            if size(ipvertical) > 0:
                pontos_vertical.resize(count+1,1)
                pontos_vertical[count] = ipvertical[1]
                count += 1
        yy = np.sort(np.unique(pontos_vertical))
        if size(yy) > 1:
            plastico = linha_simples(xx,yy[0],xx,yy[1],z,g,plastico,t_plastico,t)
            plastico = plastico + t_plastico
            l = 'G1 X' + str(xx) +  " Y" + str(yy[1]) +  " E" + str(plastico)
            print 'Sending: ' + l + ' \n',
            f.write(l +  '\n')
        pontos_vertical = np.array([])
        count = 0
        xx = xx + passos

def intersec_horizontal_ponto(v,p1,p2,n,x):
```

```
    vetor = [0,1]
    determinante = np.linalg.det ([v[n],vetor])
    minimo = np.amin([p1[n][0],p2[n][0]])
    maximo = np.amax([p1[n][0],p2[n][0]])
    if determinante != 0   and maximo>=x  and minimo<=x:
        y = (v[n][1]/v[n][0])*(x-p1[n][0])+p1[n][1]
        return np.array([x,y])
    return np.array([])

def preenchimento_horizontal(v,p1,p2,pontos,z,plastico,t_plastico,t,passos):
    count = 0
    pontos_vertical = np.array([])
    maximox = np.amax(pontos)
    minimox = np.amin(pontos)
    x = minimox
    lado = 1
    while x <= maximox:
        for n in range (0,size(p1)/2):
            pvertical = intersec_horizontal_ponto(v,p1,p2,n,x)
            if size(ipvertical) > 0:
                pontos_vertical.resize(count+1,1)
                pontos_vertical[count] = ipvertical[1]
                count += 1
        y = np.sort(np.unique(pontos_vertical))
        if size(y) > 1:
            plastico = linha_simples(y[0],x,y[1],x,z,g,plastico,t_plastico,t)
            plastico = plastico + t_plastico
            l = 'G1 X' + str(y[1]) +  " Y" + str(x) +  " E" + str(plastico)
            print 'Sending: ' + l + ' \n',
            f.write(l +  '\n')
        pontos_vertical = np.array([])
        count = 0
        x+=passos
```

```
def interseccao(my_mesh, z, i,esp,casca):
    pontos_plano = np.array([])
    [x1,y1,z1,x2,y2,z2,x3,y3,z3] = (my_mesh.points[i])
    pontos = np.array([[x1,y1,z1], [x2,y2,z2], [x3,y3,z3]])
    maximo = max([z1,z2,z3])
    minimo = min([z1,z2,z3])
    count = 0
    if z<minimo  or z>maximo:
        return pontos_plano,casca
    for n in range (0,3):
        lo = pontos[n]
        dl = pontos[(n+1)\%3]
        l = dl - lo
        po = np.array([0,0,z])
        pl = np.subtract(po,lo)
        denominador = np.dot([0,0,1],l)
        numerador = np.dot([0,0,1],pl)
        if maximo <= z+esp   and minimo >= z-esp:
            pontos_plano.resize(count+2,3)
            pontos_plano[count] = lo
            pontos_plano[count+1] = dl
            count = count + 2
            casca = 0.4
        if denominador != 0:
            d = numerador/denominador
            if d>=0  and d<=1:
                tmp = d*l + lo
                pontos_plano.resize(count+1,3)
                pontos_plano[count] = tmp
                count = count + 1
    return pontos_plano,casca

def triang_linha(pontos_plano):
    p = np.array([])
```

```
if not list(pontos_plano):
    return []
if size(pontos_plano/3)>=12:
    a1 = pontos_plano[0]
    b1 = pontos_plano[1]
    c1 = pontos_plano[2]
    d1 = pontos_plano[3]
    n = 0
    m = 0
    g = 0
    while (a1[0]==b1[0]   and a1[1]==b1[1]):
                    b1 = pontos_plano[n]
                    n=n+1
                    if n == size(pontos_plano)/3:
                            return []
            while (a1[0]==c1[0]   and a1[1]==c1[1])   or (b1[0]==c1[0]   and b1[1] == c1[1]):
                    c1 = pontos_plano[m]
                    m=m+1
                    if m == size(pontos_plano)/3:
                            p.resize(2,3)
                            p[0] = a1
                            p[1] = b1
                            return p
    while (a1[0]==d1[0]   and a1[1]==d1[1])   or (b1[0]==d1[0]   and b1[1]==d1[1])   or (c1[0]==
        d1 = pontos_plano[g]
        g = g+1
        if g == size(pontos_plano)/3:
            p.resize(3,3)
                p[0] = a1
                p[1] = b1
                p[2] = c1
                return p
    p.resize(4,3)
    p[0] = a1
```

```
    p[1] = b1
    p[2] = c1
    p[3] = d1
if size(pontos_plano/3)>=9     and size(pontos_plano/3)<=12:
    a1 = pontos_plano[0]
    b1 = pontos_plano[1]
    c1 = pontos_plano[2]
    n = 0
    m = 0
    while (a1[0]== b1[0]   and a1[1]==b1[1]):
        b1 = pontos_plano[n]
        n=n+1
        if n == size(pontos_plano)/3:
            return []
    while (a1[0]==c1[0]   and a1[1]==c1[1])   or (b1[0]==c1[0]   and b1[1] == c1[1]):
        c1 = pontos_plano[m]
        m=m+1
        if m == size(pontos_plano)/3:
            p.resize(2,3)
            p[0] = a1
            p[1] = b1
            return p
    p.resize(3,3)
    p[0] = a1
    p[1] = b1
    p[2] = c1
    return p
elif size(pontos_plano) <= 3:
    return []
else:
    a1 = pontos_plano[0]
    b1 = pontos_plano[1]
    if (a1[0]==b1[0]   and a1[1]==b1[1]):
        return []
```

```
    return pontos_plano


def armazenamento(x,tmp,count,t1):
    x1 = np.array([])
    t = np.array([])
    if size(b) == 0:
        return x,count,t1
    if size(b) == 9:
        x1.resize(count+3,3)
        t.resize(count+3,1)
        for n in range (0,count):
            x1[n] = x[n]
            t[n] = t1[n]
        x1[count]   = tmp[0]
        x1[count+1] = tmp[1]
        x1[count+2] = tmp[2]
        t[count] = 2
        t[count+1] = 0
        t[count+2] = -2
        count = count + 3
        return x1,count,t
    elif size(b) == 6:
        x1.resize(count+2,3)
        t.resize(count+2,1)
            for n in range (0,count):
            x1[n] = x[n]
            t[n] = t1[n]
            x1[count] =   tmp[0]
            x1[count+1] = tmp[1]
            t[count] = 1
            t[count+1] = -1
            count = count + 2
            return x1,count,t
    else:
```

```
        return x,count,t1


def pontos(x,t):
    a = x[:,[0,1]]
    b = np.vstack({tuple(row)     for row in a})
    tmp1 = np.array([0,0])
    tmp2 = np.array([0,0])
    c = np.eye((b.size/2))
    for n in range (0,a.size/2):
        for m in range (0, b.size/2):
            if a[n][0] == b[m][0]     and a[n][1] == b[m][1]:
                if   t[n] == 1:
                    tmp1 = np.array([a[n+1][0],a[n+1][1]])
                    for k in range (0,b.size/2):
                    if tmp1[0] == b[k][0]    and tmp1[1] == b[k][1]:
                        c[m][k] = 1
                elif t[n] == -1:
                    tmp1 = np.array([a[n-1][0],a[n-1][1]])
                        for k in range (0,b.size/2):
                            if tmp1[0] == b[k][0]    and tmp1[1] == b[k][1]:
                                c[m][k] = 1
                elif t[n] == 2:
                    tmp1 = np.array([a[n+1][0],a[n+1][1]])
                    tmp2 = np.array([a[n+2][0],a[n+2][1]])
                    for k in range (0,b.size/2):
                        if tmp1[0] == b[k][0]    and tmp1[1] == b[k][1]:
                            c[m][k] = 1
                        if tmp2[0] == b[k][0]    and tmp2[1] == b[k][1]:
                            c[m][k] = 1
                elif t[n] == -2:
                    tmp1 = np.array([a[n-1][0],a[n-1][1]])
                    tmp2 = np.array([a[n-2][0],a[n-2][1]])
                    for k in range (0,b.size/2):
                        if tmp1[0] == b[k][0]    and tmp1[1] == b[k][1]:
```

```
                            c[m][k] = 1
                        if tmp2[0] == b[k][0]    and tmp2[1] == b[k][1]:
                            c[m][k] = 1
                elif t[n] == 0:
                    tmp1 = np.array([a[n+1][0],a[n-1][1]])
                    tmp2 = np.array([a[n-1][0],a[n-1][1]])
                    for k in range (0,b.size/2):
                        if tmp1[0] == b[k][0]    and tmp1[1] == b[k][1]:
                            c[m][k] = 1
                        if tmp2[0] == b[k][0]    and tmp2[1] == b[k][1]:
                            c[m][k] = 1
                else:
                    print "erro de conexao perdida"
    c = c - np.eye((b.size/2))
    return b,c,tmp1

def erro_red(x,z):
    for n in range(0,x.size/3):
        for m in range(n,x.size/3):
            if (abs(x[n][0]-x[m][0])<z     and abs(x[n][1]-x[m][1])<z):
                x[m][0] = x[n][0]
                x[m][1] = x[n][1]
    return x


def caminho(ligacoes):
    tamanho = len(ligacoes)
    count = 0
    path = np.array([])
    for m in range(0, tamanho):
        for n in range (0, tamanho):
            if ligacoes[m][n] == 1:
                count = count + 1
                count2 = 2
```

```
                path.resize(count, 2*tamanho+1)
                path[count-1][0] = m+1
                path[count-1][1] = n+1
                ligacoes[m][n] = 0
                ligacoes[n][m] = 0
                j = n
                i = 0
                while(sum(ligacoes[j])!=0):
                    if ligacoes[j][i] == 0:
                        i = i + 1
                    else:
                        path[count-1][count2] = i+1
                        ligacoes[j][i] = 0
                        ligacoes[i][j] = 0
                        count2 = count2 + 1
                        j = i
                        i = 0
    return path

fi = open( 'resultados.txt','w')
sysargv1 = int(sys.argv[1])
my_mesh = mesh.Mesh.from_file(    'thirdcube.stl')
count = 0
minimo = my_mesh.min_
maximo = my_mesh.max_
redimensionar = 1
my_mesh.x -= (maximo[0]+minimo[0])/2
my_mesh.y -= (maximo[1]+minimo[1])/2
my_mesh.points = redimensionar*my_mesh.points
x = np.array([])
t = np.array([])
erro = 1
erro2 = 5
j = 1
```

```
w = 0
passo_m = 1
f = open( 'in.txt','w')
g = 0
espaco_fill = 1
plastico = 0.0
t_plastico = 0.1
f.write( 'G28\nG90\nM149 [C]\nM140 S70\nM104 S200\n')
m = minimo[2]*redimensionar
while m <= maximo[2]*redimensionar:
    z = m
    casca = 1
    for n in range (0,size(my_mesh.points)/9):
        a,casca = interseccao(my_mesh, m, n, erro,casca)
        b = triang_linha(a)
        x,count,t = armazenamento(x,b,count,t)
    fi.write(str(list(x)) + str(list(t)))
    if list(x):
        cnt = x[:,[0,1]]
        x = erro_red(x,erro)
        ordem,c,tmp1 = pontos(x,t)
        c1 = c*1
        fi.write(str(list(ordem))+str(list(c)))
        a = caminho(c1)
        fi.write(str(list(a)))
        np.savetxt( 'caminho.txt',a,delimiter=  ',')
        np.savetxt( 'ordem.txt',ordem,delimiter=  ',')
        l = 'G0 Z' + str(z)
        print 'Sending: ' + l,
        f.write(l +  '\n')
        for i in range (0,len(a)):
            w = 0
            j = 0
            while w!=1:
```

```
            if a[i][j] != 0:
                p1 = ordem[a[i][j] - 1]
                p2 = ordem[a[i][j+1] - 1]
                l = 'G0 X' + str(p1[0]) +  " Y" + str(p1[1])
                print 'Sending: ' + l + ' \n',
                f.write(l +  '\n')
                plastico = linha_simples(p1[0],p1[1],p2[0],p2[1], z,g,plastico,t_plastico,erro2)
                plastico += t_plastico
                l = 'G1 X' + str(p2[0]) +  " Y" + str(p2[1]) +  " E" + str(plastico)
                print 'Sending: ' + l + ' \n',
                f.write(l +  '\n')
                j=j+1
            elif i!= len(a)-1:
                p1 = ordem[a[i][j-1]-1]
                p2 = ordem[a[i+1][0]-1]
                l = 'G0 X' + str(p1[0]) +  " Y" + str(p1[1]) +  " Z" + str(z)
                print 'Sending: ' + l + ' \n',
                f.write(l +  '\n')
                plastico = linha_simples(p1[0],p1[1],p2[0],p2[1],z,g,plastico,t_plastico,erro2)
                w = 1
            else:
                w = 1
v,p1,p2 = infill_sem(ordem,c*1,z,plastico,t_plastico,erro2)
preenchimento_vertical(v,p1,p2,ordem,z,plastico,t_plastico,erro2,espaco_fill*casca)
tmp1 = 1*ordem[:,0]
tmp2 = 1*ordem[:,1]
ordem[:,1] = tmp1
ordem[:,0] = tmp2
v,p1,p2 = infill_sem(ordem,c*1,z,plastico,t_plastico,erro2)
preenchimento_horizontal(v,p1,p2,ordem,z,plastico,t_plastico,erro2,espaco_fill*casca)
tmp1 = 1*ordem[:,0]
tmp2 = 1*ordem[:,1]
ordem[:,1] = tmp1
ordem[:,0] = tmp2
```

```
m = m+passo_m
x = []
count = 0
t = []
ordem = []
c = []
tmp1 = []
a = []
```

Anexo 3: Servidor *Web*

```
import os
import sys
import io
import cv2
import subprocess
from datetime import datetime
from time import sleep
import uuid

from flask import Flask, request, render_template, session, flash, abort, send_file
from flask import send_from_directory, jsonify, redirect
from flask import url_for

from asap3d_helpers.auth.auth_helper import AuthException, verify_user_by_google_token
from asap3d_helpers.auth.auth_helper import admin_required, verify_user, save_new_user, load_
from asap3d_helpers.auth.auth_helper import login_required, get_safe_password, skip_if_logged
from asap3d_helpers.form.form_helper import get_form_email, get_form_username, get_form_passw
from asap3d_helpers.form.form_helper import get_form_chat_message, get_form_username_to
from asap3d_helpers.form.form_helper import get_form_file, FormException
from asap3d_helpers.form.form_helper import get_form_gender, get_form_name, get_form_surname
from config import UPLOAD_PATH
from db.dbconnector import update_user_password, get_all_users, insert_print, get_all_prints, get_p
from db.dbconnector import insert_new_chat_message, get_user_messages
from db.dbconnector import update_user_status, update_print_status

from stream import DummyProcessor
```

```
DEBUG = None
DEBUG_IMAGE = None

processor = None
lock_code = None

app = Flask(__name__)
app.secret_key = os.getenv(    'SESSION_SECRET_KEY', 'DEBUG_KEY')
app.config[  'MAX_CONTENT_LENGTH'] = 50 * 1024 * 1024    # 50 Mb limit
app.config[  'UPLOAD_FOLDER'] = UPLOAD_PATH


def get_lock_code():
    _code = str(uuid.uuid4())[0:4].upper()
    print "Code: " + _code
    return _code


def lock_processor(Processor):
    global processor, lock_code
    if processor:
        return False
    if DEBUG:
        processor =   'DEBUGGING'
    else:
        processor = Processor()
        processor.daemon = True
        processor.start()
        sleep(3)
        if not processor.is_ready():
            processor = None
            return False
    lock_code = get_lock_code()
```

```
    return True


@app.before_request
def before_request():
    return load_current_user()


@app.route( '/admin')
@admin_required
def admin():
    return render_template( 'admin.html', all_users=get_all_users(), all_prints=get_all_prints(all_u


@app.route( '/terms')
def terms():
    return render_template( 'terms.html')


@app.route( '/')
def index():
    return render_template( 'index.html')


@app.route( '/login', methods=[ "POST", "GET"])
@skip_if_logged
def login():
    if request.method != "POST":
        return render_template( 'login.html', title= 'ASAP 3D Login')

    try:
        user = get_form_username()
        password = get_form_password()
        username = verify_user(user, password)
```

```
        session[ 'is_logged'] = True
        session[ 'username'] = username
        return redirect( '/')
    except (FormException, AuthException)      as e:
        return render_template(  'login.html', title= 'ASAP 3D Login', login_error=e.message)


@app.route(  '/google_login', methods=[ "POST"])
@skip_if_logged
def google_login():
    id_token = request.form.get(    'id_token', '')
    try:
        username, picture = verify_user_by_google_token(id_token)
        session[ 'is_logged'] = True
        session[ 'username'] = username
        session[ 'picture'] = picture
        return jsonify(success=True)
    except AuthException   as e:
        return jsonify(success=False, message=e.message)


@app.route(  '/signup', methods=[ "POST", "GET"])
@skip_if_logged
def signup():
    if request.method !=    "POST":
        return render_template(  'signup.html', title= 'ASAP 3D Signup')


    try:
        name = get_form_name()
        surname = get_form_surname()
        gender = get_form_gender()
        birth_date = get_form_birth_date()
        email = get_form_email(confirmation=True)
        username = get_form_username(confirmation=True)
```

```
        password = get_form_password(confirmation=True)

        save_new_user(name, surname, email, username, password, gender, birth_date)

        return redirect( '/login')
    except (FormException, AuthException)      as e:
        return render_template(  'signup.html', title= 'ASAP 3D Signup', error_code=e.message


@app.route(  '/print', methods=[  'GET', 'POST'])
@login_required
def _print():
    is_printing = False
    is_streaming = False
    all_prints = get_all_prints()
    for printer  in all_prints:
        if printer[ 'Status'] == 'printing':
            is_printing = True
    if is_printing   and lock_processor(DummyProcessor):
        is_streaming = True
    return render_template(   'print.html', all_prints=all_prints, is_printing=is_printing, is_streami


@app.route(  '/frame')
def frame():
    global processor
    if processor  is None:
        return redirect( '/stream')
    else:
        if DEBUG:
            frame = DEBUG_IMAGE
        else:
            frame = processor.get_frame()
        if frame  is None:
```

```
            return abort(404)
        jpg = cv2.imencode('.jpg', frame)[1]
        return send_file(io.BytesIO(jpg), mimetype='image/jpeg', cache_timeout=1)


@app.route('/stream')
def stream():
    global processor
    if processor is None:
        if lock_processor(DummyProcessor):
            return render_template('stream.html', title='Simple Stream', code=lock_code)
        else:
            return render_template('error.html', title='Ops =/')
    else:
        return render_template('busy.html', title='Camera is busy')


@app.route('/end', methods=['GET', 'POST'])
def end():
    global processor
    if processor is None:
        return redirect('/')
    else:
        if request.method == 'POST':
            user_code = request.form.get('code', '').upper()
            empty = request.form.get('empty', False)
            if user_code == lock_code:
                if not DEBUG:
                    processor.finish()
                processor = None
                if empty:
                    return ''
                return render_template('end.html', title='Session ended', invalid=False, s
            else:
```

```
                return render_template( 'end.html', title= 'Invalid code', invalid=True, succ
        return render_template( 'end.html', title= 'End session', form=True, success=False)


@app.route( '/upload', methods=[ 'POST'])
@login_required
def upload_file():
    try:
        received_file, filename = get_form_file()
        file_full_path = os.path.join(app.config[ 'UPLOAD_FOLDER'], filename)
        received_file.save(file_full_path)
        file_size = os.path.getsize(file_full_path)
        insert_print({
            'FileName': filename,
            'CreationTimestamp': datetime.utcnow(),
            'Status': 'received',
            'FileSize': file_size,
        })
        with open( '/dev/null') as sink:
            subprocess.Popen([ 'blender', '-b', '--python', 'preview_generator.py', '-
                              stdout=sink, stderr=subprocess.STDOUT)
        flash( 'Arquivo enviado com sucesso!')
        return redirect( '/print')
    except FormException as e:
        flash(e.message)
        return redirect( '/print')


@app.route( '/about')
def about():
    return render_template( 'about.html')


@app.route( '/logout')
```

```
def logout():
    session.clear()
    return redirect( '/login')


@app.route( '/change', methods=[ "POST", "GET"])
def change():
    if request.method !=   "POST":
        return render_template(  'change.html')

    try:
        username = get_form_username()
        email = get_form_email()
        birth_date = get_form_birth_date()
        new_password = get_form_password(confirmation=True)

        valid_user = load_user(username)
        if valid_user[ 'UserEmail'] != email  or valid_user[ 'Birthdate'] != birth_date:
            raise AuthException(  'INVALID_ENTRY')

        new_password = get_safe_password(new_password)
        success = update_user_password(username, new_password)
        if not success:
            raise AuthException(  'INSERT_ERROR')
        return redirect( '/logout')
    except (FormException, AuthException)     as e:
        return render_template(  'change.html', error_code=e.message)


@app.route( '/update/<string:action>/<string:username>', methods=[ 'GET'])
@admin_required
def update(action, username):
    actions = dict(ban=  'banned', activate= 'active', deactivate= 'inactive', make_admin= 'a
    if action  in actions:
```

```
        status = actions[action]
        success = update_user_status(username, status)
        if success:
            flash(username +   ' atualizado para status ' + status)
        else:
            flash( 'Falha ao atualizar status de ' + username +  ' para ' + status)
    return redirect( '/admin')


@app.route( '/update_print/<string:action>/<string:printid>', methods=[ 'GET'])
@admin_required
def update_print(action, printid):
    actions = dict(receive=  'received', printer= 'printing', cancel= 'canceled', process= 'p
    if action  in actions:
        status = actions[action]
        success = update_print_status(printid, status, ignore_username=True)
        if success:
            flash(printid +  ' atualizado para status ' + status)
        else:
            flash( 'Falha ao atualizar status de ' + printid +  ' para ' + status)
    return redirect( '/admin')


@app.route( '/update_print_user/<string:action>/<string:printid>', methods=[ 'GET'])
@login_required
def update_print_user(action, printid):
    actions = dict(cancel=  'canceled')
    if action  in actions:
        status = actions[action]
        success = update_print_status(printid, status)
        if success:
            flash(printid +  ' atualizado para status ' + status)
        else:
            flash( 'Falha ao atualizar status de ' + printid +  ' para ' + status)
```

```
    return redirect( '/print')


@app.route( '/json/print/<int:print_id>', methods=[ 'GET'])
@login_required
def json_get_print(print_id):
    full_print = get_print(print_id)
    if full_print  is None:
        return jsonify(error=True)
    preview_name = full_print[   'FileName'][:-4] +  '.png'
    if os.path.exists( 'static/previews/' + preview_name):
        full_print[ 'PreviewUrl'] = url_for( 'static', filename= 'previews/' + full_print[
    return jsonify(full_print)


@app.route( '/printer/<string:action>/<string:order>', methods=[ 'GET'])
@admin_required
def printer():
    return redirect( '/admin')


@app.route( '/chat/<string:action>', methods=[ 'POST'])
@login_required
def send_new_message(action):
    if action ==  'send':
        message = get_form_chat_message()
        success = insert_new_chat_message(message, datetime.utcnow())
        return jsonify(success=success)
    if action ==  'get':
        messages = get_user_messages()
        return jsonify(messages=messages)
    return abort(404)
```

```
@app.route( '/chat_admin/<string:action>', methods=[ 'POST'])
@admin_required
def admin_send_new_message(action):
    try:
        if action == 'send':
            message = get_form_chat_message()
            username_to = get_form_username_to()
            success = insert_new_chat_message(message, datetime.utcnow(), username_to=username_to)
            return jsonify(success=success)
        if action == 'get':
            username = get_form_username_to()
            messages = get_user_messages(username=username)
            return jsonify(messages=messages)
    except FormException as e:
        return jsonify(error=True, code=e.message)
    return abort(404)


@app.route( '/fonts/<path:name>')
def fonts(name):
    return send_from_directory( 'static/fonts/', name)


if __name__ == "__main__":
    DEBUG = len(sys.argv) > 1 and sys.argv[1] == '-d'
    if DEBUG:
        DEBUG_IMAGE = cv2.imread( 'frame.jpg')
    app.run(host= '0.0.0.0', threaded=True, debug=True)
```

Anexo 4: Conexão com o Banco de Dados

```
import re


from mysql import connector
from config import get_config  as _get_config
from flask import g

# In ascending access level order
ACCESS_LEVELS = [  'public', 'user', 'admin']



class Context(object):
    def __init__(self, cnx, access_level, current_username):
        self.cnx = cnx
        self.access_level = access_level
        self.current_username = current_username



def inject_context(min_level=    'public'):
    def _provides_context(f):
        def wrapper(*args, **kwargs):
            access_level =   'public'
            current_username = None
            if g and g.get( 'user', None):
                current_username = g.user[    'UserName']
                if g.user[ 'Status'] ==  'admin':
                    access_level =   'admin'
```

```
                else:
                    access_level =   'user'

            if ACCESS_LEVELS.index(access_level) < ACCESS_LEVELS.index(min_level):
                raise Exception("INSUFFICIENT_DB_PERMISSION")

            config = _get_config(access_level)
            cnx = connector.connect(**config)
            context = Context(cnx, access_level, current_username)
            if access_level ==   'admin':
                return f(context, *args, **kwargs)
            return f(context, *args)
        return wrapper
    return _provides_context


def _get_short_file_name(filename):
    match = re.match(  r'^(.+)_.+_.+.stl$', filename)
    if match:
        groups = match.groups()
        return str(groups[0]) +   '.stl'
    return ''


def _make_user_dict(raw_user):
    return {
        'FirstName': raw_user[0],
        'LastName': raw_user[1],
        'UserEmail': raw_user[2],
        'UserName': raw_user[3],
        'UserPass': raw_user[4],
        'Gender': raw_user[5],
        'Birthdate': raw_user[6],
        'Status': raw_user[7]
```

```
    }


def _make_print_dict(raw_print):
    return {
        'PrintId': raw_print[0],
        'UserName': raw_print[1],
        'CreationTimestamp': raw_print[2],
        'LastUpdateTimestamp': raw_print[3],
        'FileName': raw_print[4],
        'ShortFileName': _get_short_file_name(raw_print[4]),
        'Status': raw_print[5],
        'FileSize': raw_print[6],
    }


def _make_full_print_dict(raw_print):
    return {
        'PrintId': raw_print[0],
        'UserName': raw_print[1],
        'CreationTimestamp': raw_print[2],
        'LastUpdateTimestamp': raw_print[3],
        'FileName': raw_print[4],
        'ShortFileName': _get_short_file_name(raw_print[4]),
        'Status': raw_print[5],
        'FileSize': raw_print[6],
        'Material': raw_print[7],
        'Cost': raw_print[8],
        'Divisions': raw_print[9],
    }


def _make_chat_dict(raw_chat):
    return {
```

```
        'MessageId': raw_chat[0],
        'UserNameFrom': raw_chat[1],
        'UserNameTo': raw_chat[2],
        'MessageTimestamp': raw_chat[3],
        'Message': raw_chat[4],
    }


@inject_context()
def insert_user(context, user):
    cnx = context.cnx
    query = ( "INSERT INTO users(FirstName, LastName, UserEmail, UserName, UserPas
              "values(%(FirstName)s, %(LastName)s, %(UserEmail)s, %(UserName)s, "
              "%(UserPass)s, %(Gender)s, %(Birthdate)s, %(Status)s)")

    cursor = cnx.cursor()
    success = True
    try:
        cursor.execute(query, user)
        cnx.commit()
    except connector.Error:
        success = False

    cursor.close()
    cnx.close()

    return success


@inject_context()
def get_user(context, user_name):
    cnx = context.cnx
    query = ( "SELECT FirstName, LastName, UserEmail, UserName, UserPass, Gender,
              "FROM users WHERE UserName = %(UserName)s LIMIT 1")
```

```
    cursor = cnx.cursor()
    cursor.execute(query, {    'UserName': user_name})
    raw_user = cursor.fetchone()

    cursor.close()
    cnx.close()

    if raw_user  is not None:
        return _make_user_dict(raw_user)
    return None


@inject_context()
def get_user_by_email(context, user_email):
    cnx = context.cnx
    query = ( "SELECT FirstName, LastName, UserEmail, UserName, UserPass, Gender,
              "FROM users WHERE UserEmail = %(UserEmail)s LIMIT 1")

    cursor = cnx.cursor()
    cursor.execute(query, {    'UserEmail': user_email})
    raw_user = cursor.fetchone()

    cursor.close()
    cnx.close()

    if raw_user  is not None:
        return _make_user_dict(raw_user)
    return None


@inject_context(min_level=    'admin')
def get_all_users(context):
    cnx = context.cnx
```

```
    query = ( "SELECT FirstName, LastName, UserEmail, UserName, UserPass, Gender, 
              "FROM users order by UserName")

    cursor = cnx.cursor()
    cursor.execute(query)

    users = []
    for raw_user  in cursor:
        user = _make_user_dict(raw_user)
        del user[ 'UserPass']
        users.append(user)
    cursor.close()
    cnx.close()
    return users


@inject_context(min_level=    'admin')
def update_user_status(context, user_name, new_status):
    cnx = context.cnx
    query = ( "UPDATE users SET Status = %(NewStatus)s "
              "WHERE UserName = %(UserName)s LIMIT 1")

    cursor = cnx.cursor()
    try:
        cursor.execute(query, {   'UserName': user_name,   'NewStatus': new_status})
        cnx.commit()
        success = cursor.rowcount == 1
    except connector.Error:
        success = False
    cursor.close()
    cnx.close()

    return success
```

```
@inject_context()
def update_user_password(context, user_name, new_password):
    cnx = context.cnx
    query = ( "UPDATE users SET UserPass = %(NewPassword)s "
              "WHERE UserName = %(UserName)s LIMIT 1")

    cursor = cnx.cursor()
    try:
        cursor.execute(query, {   'UserName': user_name,   'NewPassword': new_password})
        cnx.commit()
        success = cursor.rowcount == 1
    except connector.Error:
        success = False
    cursor.close()
    cnx.close()

    return success


@inject_context(min_level=    'user')
def insert_print(context, file_print):
    cnx = context.cnx
    file_print[  'UserName'] = context.current_username
    query = ( "INSERT INTO prints("
              "UserName, CreationTimestamp, FileName, Status, FileSize) "
              "values("
              "%(UserName)s, %(CreationTimestamp)s, %(FileName)s, %(Status)s, %(Fi

    cursor = cnx.cursor()
    try:
        cursor.execute(query, file_print)
        cnx.commit()
        success = True
```

```
    except connector.Error   as e:
        print e
        success = False

    cursor.close()
    cnx.close()

    return success


@inject_context(min_level=    'user')
def update_print_status(context, print_id, new_status, ignore_username=False):
    cnx = context.cnx
    query_admin = (  "UPDATE prints SET Status = %(NewStatus)s "
             "WHERE PrintId = %(PrintId)s LIMIT 1")
    query_user = (  "UPDATE prints SET Status = %(NewStatus)s "
             "WHERE PrintId = %(PrintId)s and WHERE UserName = %(UserName)s LIMIT

    query = query_admin    if ignore_username    else query_user
    cursor = cnx.cursor()
    try:
        cursor.execute(query, {    'PrintId': print_id,   'NewStatus': new_status,   'UserName':
        cnx.commit()
        success = cursor.rowcount == 1
    except connector.Error:
        success = False
    cursor.close()
    cnx.close()

    return success


@inject_context(min_level=    'user')
def get_print(context, print_id):
```

```
    cnx = context.cnx
    query = ( "SELECT PrintId, UserName, CreationTimestamp, LastUpdateTimestamp, F
              "FROM prints WHERE PrintId = %(PrintId)s and UserName = %(UserName)s

    cursor = cnx.cursor()
    cursor.execute(query, {    'PrintId': print_id,   'UserName': context.current_username})
    raw_print = cursor.fetchone()

    cursor.close()
    cnx.close()

    if raw_print  is not None:
        return _make_print_dict(raw_print)
    return None


@inject_context(min_level=    'user')
def get_all_prints(context, all_users=False):
    cnx = context.cnx
    query = ( "SELECT PrintId, UserName, CreationTimestamp, LastUpdateTimestamp, F
              "FROM prints " + ("WHERE UserName = %(UserName)s " if not all_users  els
              "ORDER BY LastUpdateTimestamp DESC limit 10")

    cursor = cnx.cursor()
    cursor.execute(query, {    'UserName': context.current_username})

    prints = []
    for raw_print  in cursor:
        prints.append(_make_print_dict(raw_print))
    cursor.close()
    cnx.close()
    return prints
```

```
@inject_context(min_level=    'user')
def insert_new_chat_message(context, message, timestamp, username_to=          'group:admin'):
    cnx = context.cnx
    query = ( "INSERT INTO chat("
              "UserNameFrom, UserNameTo, MessageTimestamp, Message) values("
              "%(UserNameFrom)s, %(UserNameTo)s, %(MessageTimestamp)s, %(Message)s

    cursor = cnx.cursor()
    try:
        cursor.execute(query,
                       {'UserNameFrom': context.current_username,
                        'UserNameTo': username_to,
                        'MessageTimestamp': timestamp,
                        'Message': message})
        cnx.commit()
        success = cursor.rowcount == 1
    except connector.Error:
        success = False

    cursor.close()
    cnx.close()
    return success


@inject_context(min_level=    'user')
def get_user_messages(context, username=None):
    username = context.current_username      if username  is None  else username
    cnx = context.cnx
    query = ( "SELECT MessageId, UserNameFrom, UserNameTo, MessageTimestamp, Messa
              "FROM chat WHERE UserNameTo = %(UserName)s or UserNameFrom = %(UserN
              "ORDER BY MessageTimestamp DESC limit 30")

    cursor = cnx.cursor()
    cursor.execute(query, {    'UserName': username})
```

```
    chats = []
    for raw_chat  in cursor:
        chats.append(_make_chat_dict(raw_chat))
    cursor.close()
    cnx.close()
    return chats


def manager_get_all_prints(status):
    query = ( "SELECT PrintId, UserName, CreationTimestamp, LastUpdateTimestamp, F
              "FROM prints WHERE Status = %(Status)s "
              "ORDER BY LastUpdateTimestamp ASC")

    config = _get_config(   'admin')
    cnx = connector.connect(**config)
    cursor = cnx.cursor()
    cursor.execute(query, {    'Status': status})

    prints = []
    for raw_print  in cursor:
        prints.append(_make_full_print_dict(raw_print))
    cursor.close()
    cnx.close()
    return prints


def manager_update_print(p):
    config = _get_config(   'admin')
    cnx = connector.connect(**config)
    query = (
            "UPDATE prints SET "
            "Status = %(Status)s, "
            "Material = %(Material)s, "
```

```
            "Divisions = %(Divisions)s, "
            "Cost = %(Cost)s "
            "WHERE PrintId = %(PrintId)s LIMIT 1")

    cursor = cnx.cursor()
    try:
        cursor.execute(query, p)
        cnx.commit()
        success = cursor.rowcount == 1
    except connector.Error:
        success = False
    cursor.close()
    cnx.close()

    return success
```

Anexo 5: Códigos de teste

Teste de criação e rotação de arquivo stl

```
# Create 3 faces of a cube
data = numpy.zeros(6, dtype=mesh.Mesh.dtype)

# Top of the cube
data['vectors'][0] = numpy.array([[0, 1, 1],
                                  [1, 0, 1],
                                  [0, 0, 1]])
data['vectors'][1] = numpy.array([[1, 0, 1],
                                  [0, 1, 1],
                                  [1, 1, 1]])
# Right face
data['vectors'][2] = numpy.array([[1, 0, 0],
                                  [1, 0, 1],
                                  [1, 1, 0]])
data['vectors'][3] = numpy.array([[1, 1, 1],
                                  [1, 0, 1],
                                  [1, 1, 0]])
# Left face
data['vectors'][4] = numpy.array([[0, 0, 0],
                                  [1, 0, 0],
                                  [1, 0, 1]])
data['vectors'][5] = numpy.array([[0, 0, 0],
                                  [0, 0, 1],
                                  [1, 0, 1]])
```

```
# Since the cube faces are from 0 to 1 we can move it to the middle by
# substracting .5
data['vectors'] -= .5

cube_back = mesh.Mesh(data.copy())
cube_front = mesh.Mesh(data.copy())

# Rotate 90 degrees over the X axis followed by the Y axis followed by the
# X axis
cube_back.rotate([0.5, 0.0, 0.0], math.radians(90))
cube_back.rotate([0.0, 0.5, 0.0], math.radians(90))
cube_back.rotate([0.5, 0.0, 0.0], math.radians(90))

cube = mesh.Mesh(numpy.concatenate([
    cube_back.data.copy(),
    cube_front.data.copy(),
]))

# Optionally render the rotated cube faces
from matplotlib import pyplot
from mpl_toolkits import mplot3d

# Create a new plot
figure = pyplot.figure()
axes = mplot3d.Axes3D(figure)

# Render the cube
axes.add_collection3d(mplot3d.art3d.Poly3DCollection(cube.vectors))

# Auto scale to the mesh size
scale = cube_back.points.flatten(-1)
axes.auto_scale_xyz(scale, scale, scale)
```

```
# Show the plot to the screen
pyplot.show()
```

Teste de Geração de G-Code e comunicação serial

```
from mecode import G
g = G(printer_port = 2560, baudrate = 115200, extrude = False)
g.move(10, 10)    # move 10mm in x and 10mm in y
g.arc(x=10, y=5, radius=20, direction=     'CCW')  # counterclockwise arc with a radius of
g.meander(5, 10, spacing=1)      # trace a rectangle meander with 1mm spacing between pa
g.abs_move(x=1, y=1)     # move the tool head to position (1, 1)
g.home()   # move the tool head to the origin (0, 0)
```

Teste Serial com base grbl

```
#!/usr/bin/env python
"""\
Simple g-code streaming script for grbl
"""

import serial
import time

# Open grbl serial port
s = serial.Serial(  '/dev/ttyACM0',115200)

# Open g-code file
f = open( 'Codigo-GCode.nc','r');

# Wake up grbl
s.write( "\r\n\r\n")
time.sleep(2)     # Wait for grbl to initialize
s.flushInput()    # Flush startup text in serial input

# Stream g-code to grbl
```

```
for line in f:
    l = line.strip()   # Strip all EOL characters for streaming
    print 'Sending: ' + l,
    s.write(l +  '\n') # Send g-code block to grbl
    grbl_out = s.readline()    # Wait for grbl response with carriage return
    print ' : ' + grbl_out.strip()

# Wait here until grbl is finished to close serial port and file.
raw_input( "  Press <Enter> to exit and disable grbl.")

# Close file and serial port
f.close()
s.close()
```