Guilherme Menezes Costa

Controle de diâmetro na produção de filamento para impressoras 3D

Brasil

2015

Guilherme Menezes Costa

Controle de diâmetro na produção de filamento para impressoras 3D

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Mecânica, Computação e Controle.

Centro Federal de Educação Tecnológica de Minas Geras – CEFET-MG

Departamento de Engenharia Mecatrônica

Engenharia Mecatrônica

Orientador: Daniel Alves Costa

Brasil

2015

Guilherme Menezes Costa
Controle de diâmetro na produção de filamento para impressoras 3D/ Guilherme Menezes Costa. – Brasil, 2015-
74 p. : il. (algumas color.) ; 30 cm.

Orientador: Daniel Alves Costa

TCC (Graduação) – Centro Federal de Educação Tecnológica de Minas Geras – CEFET-MG
Departamento de Engenharia Mecatrônica
Engenharia Mecatrônica, 2015.

1. Filamentos. 2. Impressora 3D. 3. Controle de diâmetro. 4. Extrusão. 5. Visão Computacional. I. Daniel Alves Costa. II. Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG. III. Departamento de Engenharia Mecatrônica. IV. Graduação.

Guilherme Menezes Costa

Controle de diâmetro na produção de filamento para impressoras 3D

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Mecânica, Computação e Controle.

Trabalho aprovado. Brasil, 09 de junho de 2015:

Daniel Alves Costa
Orientador

Professor
Luiz Cláudio Oliveira

Professor
Wagner Custódio de Oliveira

Brasil
2015

Este trabalho é dedicado à Larissa Maia, pessoa cuja qual me acompanhou em tantos momentos difíceis e que hoje é parte integrante e indispensável da minha vida e aos meus pais cujo o apoio e incentivo se mostraram essenciais ao longo do curso.

Agradecimentos

Gostaria de primeiramente agradecer a minha namorada e companheira de uma vida Larissa Maia Carvalho. Absolutamente nada disso seria possível sem o seu apoio incondicional, amor e suporte. Essa conquista nada mais é que apenas um fruto de algo plantado e regado com muito amor ao longo de muitos anos.

Aos meus pais pelo amor, carinho, suporte e fé que depositaram em mim ao longo de toda essa jornada.

Ao meu orientador Daniel Alves Costa por toda a dedicação e empenho na realização deste projeto no pouco tempo que lhe coube.

Ao CEFET-MG por fornecer dentro de suas limitações o melhor ensino possível bem como o apoio necessário para que projetos antes julgados como ambiciosos demais, saíssem do papel.

A todos que fizeram parte direta ou indiretamente dessa conquista.

"Quem avança confiante na direção
de seus sonhos e se empenha em
viver a vida que sonhou para si,
encontra um sucesso inesperado
em seu dia-a-dia"
(Henry Ford)

Resumo

Este trabalho tem como objetivo principal produzir filamentos para impressoras 3D com um diâmetro nominal de 3 mm e com uma tolerância dimensional de aproximadamente 0,05 mm, utilizando técnicas de controle com um sensoriamento de alta precisão e de baixo custo. Dentre os processos de manufatura atualmente disponíveis, escolheu-se utilizar a extrusão devido a facilidade, simplicidade, baixo custo e compatibilidade com as características do produto final. Visa-se implementar duas malhas de controle, sendo uma para realizar o controle do diâmetro do filamento através da ação de um sistema de puxamento ao final da linha de produção e outra para realizar o controle da temperatura do tubo de extrusão principal que fará o aquecimento do material. Estes parâmetros afetam a qualidade do filamento produzido. Ao final do trabalho são apresentados os resultados bem como é realizada uma breve discussão de como parâmetros cruciais do processo como temperatura e velocidade de extrusão influenciam na qualidade do filamento produzido.

Palavras-chave: Filamentos. Impressora 3D. Controle de diâmetro. Extrusão. Visão computacional.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PWM	*Pulse Width Modulation*
PC	*Personal computer*
IHM	Interface homem-máquina
ABS	Acrilonitrila Butadieno Estireno
MABS	Metil Acrilonitrila Butadieno Estireno
USB	*Universal Serial Bus*
PRBS	*Pseudorandom binary sequence*
ARX	Auto regressivo com entradas exógenas

Lista de símbolos

β	Letra grega Beta
Ω	Letra grega Omega. Símbolo para resistência elétrica

Sumário

1 Introdução

A tecnologia de impressão 3D se torna cada dia mais popular. A possibilidade de produzir peças de forma rápida e eficiente mesmo com geometrias complexas no conforto de casa é realmente atraente. Porém, um dos fatores ainda limitantes dessa tecnologia é o preço dos componentes da impressora e do filamento que a alimenta. Um quilograma de filamento para impressoras 3D de qualidade pode chegar a custar um décimo do valor da própria impressora.

Vê-se então que é preciso popularizar o processo de produção desse tipo de filamento, não apenas da impressora em si. Este trabalho busca portanto, realizar o projeto, modelagem, simulação e fabricação de uma extrusora de termoplásticos de baixo custo, porém de qualidade equiparável ou mesmo superior a dos filamentos atualmente disponíveis no mercado brasileiro. Para tanto, o trabalho fornecerá um estudo detalhado sobre o processo de extrusão empregado especificamente para a produção deste.

1.1 Definição do problema

O grande problema encontrado na produção de filamentos para impressora 3D é o controle de seu diâmetro. As impressoras atualmente disponíveis requerem que o diâmetro seja constante ao longo de todo o filamento, uma vez que caso este seja maior do que o informado, haverá deposição de material em excesso, causando defeitos dimensionais na peça e, caso este seja menor, haverá falta de material, ocasionando lacunas na estrutura da peça fabricada. Desta forma, um controle preciso do diâmetro de saída do filamento se torna essencial para a produção de peças com qualidade e alta precisão dimensional.

1.2 Motivação

A motivação em realizar este trabalho de pesquisa vem da possibilidade de preencher uma lacuna do mercado brasileiro. A crescente demanda pela nova tecnologia de impressão 3D leva também a uma elevada demanda por este tipo de produto. Dado que atualmente existem poucas empresas que fornecem esse tipo específico de filamento, a oportunidade de mercado se torna evidente.

1.3 Objetivo geral

Planeja-se produzir filamentos a partir de termoplásticos com a qualidade necessária para utilização em impressoras 3D.

1.4 Objetivos específicos

Deseja-se obter filamentos com diâmetro nominal de 3 mm (real de 2.85 mm) e tolerância dimensional de 0,05 mm. Para tanto, foi projetada e fabricada uma extrusora de termoplásticos de pequeno porte que possibilite a extrusão do filamento especificado. Para atingir as especificações, será necessário um rígido controle de temperatura e da velocidade de extrusão do processo. O sistema contará então com duas malhas de controle para monitorar e ajustar essas variáveis de acordo com o especificado.

1.5 Escopo do trabalho

No capítulo dois faz-se uma breve introdução aos temas relacionados ao processo de estereolitografia e assuntos relacionados à produção de filamentos para impressoras 3D, ao processo de extrusão, ao método de visão computacional utilizado para realizar o sensoreamento do projeto e as técnicas de controle empregadas para realizar o controle das variáveis de interesse do processo de extrusão.

No capítulo três é descrita a metodologia utilizada para o desenvolvimento da máquina extrusora de filamentos. É descrito em detalhes como foram projetadas as malhas de controle, e os programas que são parte integrante do projeto.

O capítulo quatro apresenta os resultados obtidos com a máquina. Além disso, uma pequena discussão acerca dos mesmos é feita de forma a qualificar os resultados obtidos.

As conclusões bem como algumas sugestões para trabalhos futuros são apresentadas no capítulo cinco.

As referências bem como o apêndice e os anexos estão presentes no final do trabalho.

2 Revisão da literatura

2.1 O filamento para impressora 3D

Pouca literatura pode ser encontrada sobre o tema uma vez que a produção de filamentos para impressoras 3D é um assunto relativamente novo e de âmbito industrial. Assim, este capítulo será divido entre a escolha dos materiais utilizados na construção de uma extrusora e uma breve discussão teórica sobre a influência do erro dimensional no diâmetro do filamento no traço de impressão.

A escolha do material foi baseada no coeficiente térmico dos materiais. Materiais com alta condutividade térmica tendem a espalhar o calor gerado pelo sistema de aquecimento da extrusora, interferindo de forma indesejável no processo. Dessa forma, é de interesse do ponto de vista de concentração de calor que o tubo principal seja feito de um material que possua uma baixa condutividade térmica. A tabela 2.1 mostra alguns coeficientes a mérito de comparação.

Analisando os dados, optou-se por fabricar o barril principal de aço reduzindo o gradiente térmico ao longo de todo o barril, evitando assim que o material plástico seja derretido antes da hora, o que poderia provocar um entupimento do barril extrusor.

Para a rosca extrusora, o principal fator a ser considerado é a resistência às forças a que esta será submetida e a facilidade de usinagem. Por essa razão, optou-se por utilizar aço carbono ABNT 1020 na fabricação da rosca extrusora por se tratar de um material de baixo custo, fácil usinagem e com boa resistência a esforços mecânicos.

Quanto ao filamento, o parâmetro de interesse é o diâmetro do mesmo. Segundo (BOTFEEDER, 2015), a tolerância dimensional geralmente aceita vai de 0,05 mm à 0,07 mm. Dessa forma, deseja-se que a tolerância dimensional do filamento produzido seja de aproximadamente 0,05 mm. A figura 1 ilustra a influência do erro no diâmetro na espessura do traço.

Pela figura 1 é possível ver que:

$$A_f = K \cdot \Delta A_f$$

$$K = \frac{\frac{\pi}{4} \cdot (\Delta D + D)^2}{\frac{\pi}{4} \cdot D^2}$$

$$K = \frac{D^2 + 2\Delta D \cdot D + \Delta D^2}{D^2} \tag{2.1}$$

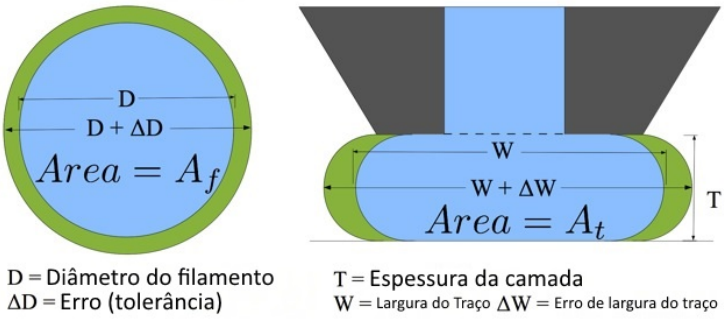

Figura 1: Modelagem do erro no filamento e no traço impresso (adaptado de PROTOPARADIGM,2014).

Sabendo que ΔD é a variação de diâmetro do filamento e pode ser expressa como:

$$\Delta D = P \cdot D \tag{2.2}$$

Onde P é a variação do filamento nominal expressa em forma de porcentagem. Substituindo 2.1 em 2.1 e desconsiderando o último termo por ser muito pequeno em comparação aos termos da equação temos:

$$K = 1 + 2P \tag{2.3}$$

$$A_f = (1 + 2P) \cdot \Delta A_f \tag{2.4}$$

A equação 2.4 nos mostra que o erro na área do filamento será de duas vezes o erro em porcentagem do filamento. Dessa forma, se considerarmos T da figura 1 que é a espessura do traço como sendo zero, temos que a área de impressão será a área do filamento expandida por um fator Q. Sendo assim temos:

$$A_t = Q \cdot A_f$$

Vemos então que, uma vez que a relação de área do filamento e do traço é linear, a largura do traço de impressão será alterada na mesma proporção. Dessa forma, podemos concluir que uma variação de, por exemplo, 5% no filamento irá causar uma alteração de 10% na área de impressão.

Além disso, Inúmeros erros podem ser induzidos na impressora ao se utilizar filamentos com alta tolerância dimensional. A figura 2 mostra um exemplo de extrusora utilizada em impressoras 3D. É possível ver como o filamento é comprimido de forma a forçar sua passagem através do orifício de saída, realizando assim a extrusão do filamento.

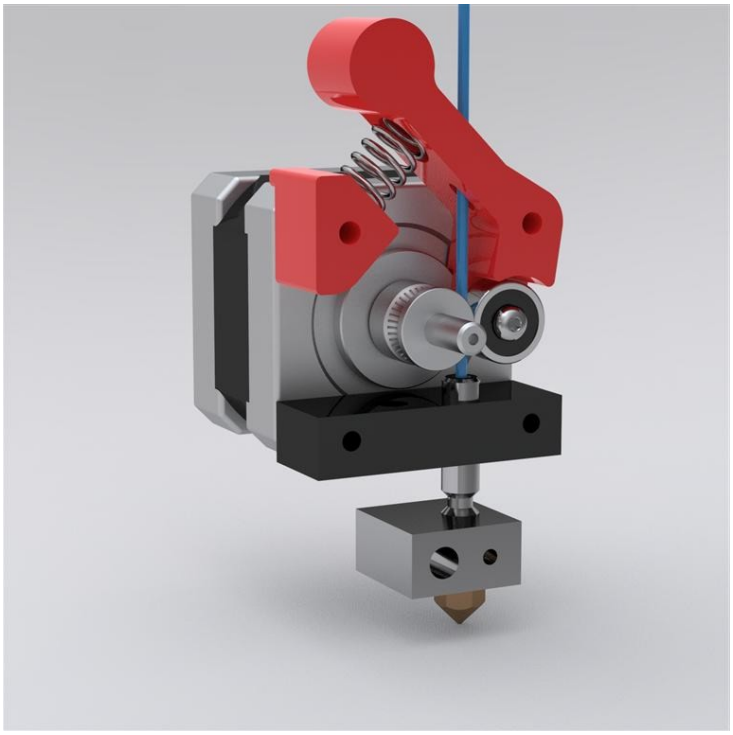

Figura 2: Exemplo de uma extrusora de impressora 3D (MAKERSLIDE.COM, 2015).

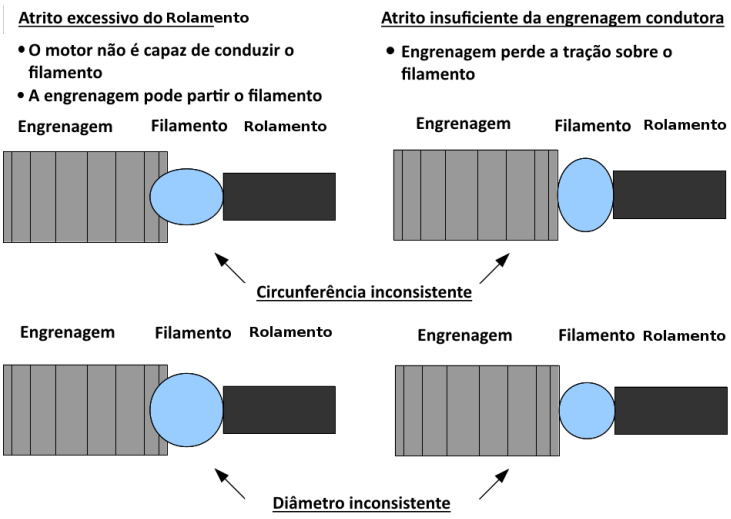

Figura 3: Influência da variância do filamento no drive de impressão (adaptado de PROTOPARADIGM, 2014).

Um filamento inconsistente em seu diâmetro causa uma variação na pressão exercida sobre o filamento. Tal variação pode causar diversos erros na impressão e devem ser minimizadas. A figura 3 mostra alguns dos efeitos causados por filamentos inconsistentes.

É possível observar que pequenas variações na circunferência e no diâmetro podem então levar ao travamento da extrusora da impressora. Desta forma, é imprescindível que o filamento possua tais características consistentes ao longo de todo o fio para que o alimentador da impressora possa trabalhar de forma correta e evitar falhas como mostra a

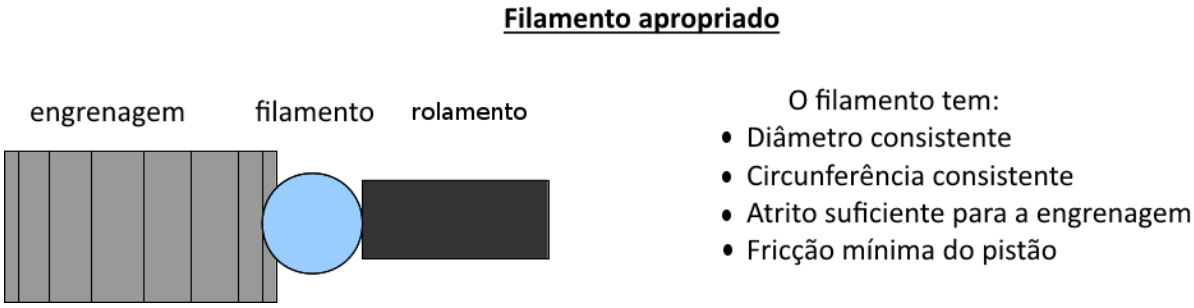

Figura 4: Características de um filamento de boa qualidade (adaptado de PROTOPARADIGM, 2014)

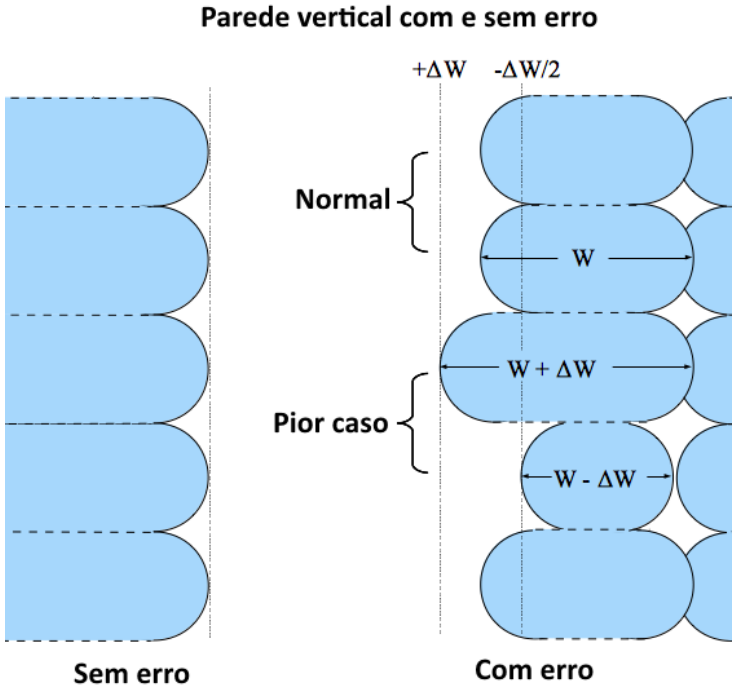

Figura 5: Modelagem do erro no filamento e no traço impresso (adaptado de PROTOPARADIGM,2014).

figura 4.

Em processos estereolitográficos a qualidade do produto impresso deve ser sempre uma prioridade. Além do travamento da impressora, filamentos com uma tolerância dimensional muito grande podem influenciar diretamente na qualidade da impressão conforme ilustra a figura 5.

Pela figura 5, é fácil perceber que diâmetros inconsistentes geram camadas irregulares. Caso o erro seja suficientemente grande, a estrutura do produto impresso poderá ser comprometida, gerando assim um produto com dimensões diferentes do que se planeja obter.

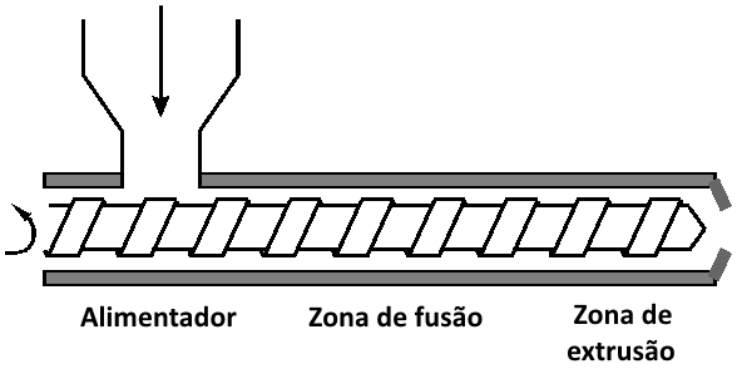

Figura 6: O processo de extrusão (adaptado de POLYMERPROCESSING.COM,2015).

2.2 O processo de extrusão

Um processo de extrusão consiste forçar a matéria prima ao longo de uma matriz de forma a moldar o material conforme desejado. A figura 6 ilustra o processo.

No caso da extrusão de filamentos, o plástico é forçado contra um orifício de diâmetro maior que o desejado para o filamento final. Dentro do tubo, o plástico se encontra em alta pressão e temperatura. Segundo (HAROLD JR.; WAGNER, 2005) após deixar a extrusora, o plástico sofre o que é chamado de *die swell*, um inchamento devido à variação de pressão e temperatura. Motivo pelo qual é impossível controlar o diâmetro final do filamento pela escolha do bico extrusor.

Para controlar o diâmetro, é necessário passar o filamento por um sistema de *puller*, que tem como objetivo puxar o filamento de forma que a vazão de saída de filamento do *puller* seja maior do que a de saída da extrusora, reduzindo assim a secção transversal do filamento. Para tanto é necessário algum sensor que reconheça o diâmetro do filamento logo após este deixar a extrusora.

Os sensores atualmente disponíveis no mercado são geralmente muito caros, o que acaba inviabilizando economicamente pequenas produções. O presente trabalho tem como proposta a utilização de uma *webcam* convencional para o reconhecimento do diâmetro do filamento através do emprego de algoritmos de visão computacional disponíveis na biblioteca OpenCV.

2.3 Visão computacional

A visão computacional é um subcampo da inteligência artificial. Seu propósito, segundo (MARENGONI; STRINGHINI, 2009), é emular a visão humana onde temos como entrada uma imagem e a partir desta é gerada uma saída, uma interpretação da imagem

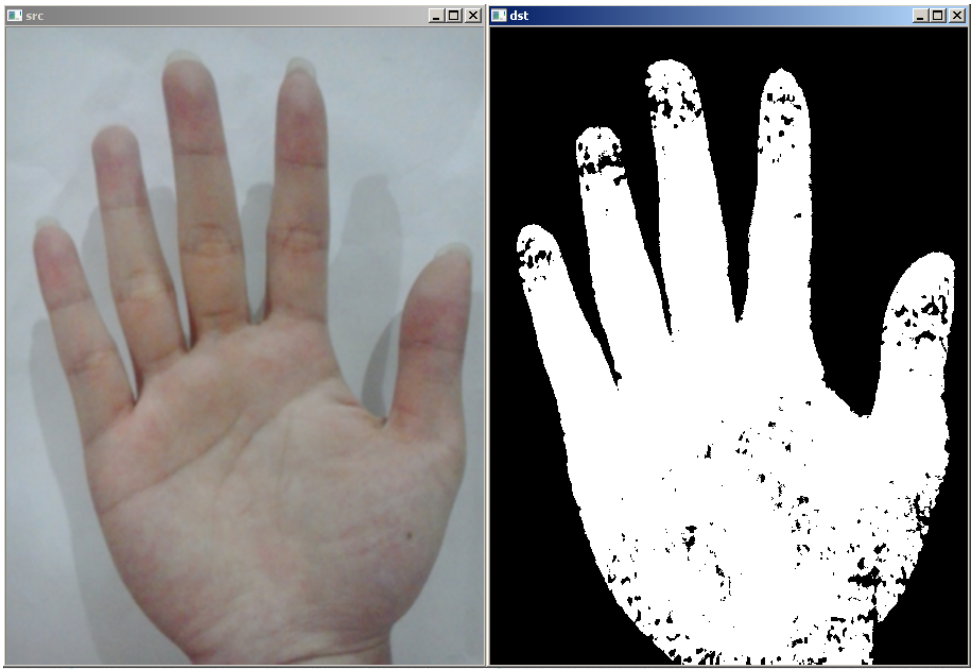

Figura 7: Exemplo de segmentação por *threshold*.

na forma de dados ou qualquer outro tipo de informação que seja desejável. A partir do processamento de imagens é possível extrair medidas como larguras, diâmetros e cores ou mesmo informações mais complexas como placas de carros e reconhecimento de faces.

A OpenCV é uma biblioteca multiplataforma que pode ser utilizada em C++, C, Python e Java. Desenvolvida pela Intel, é uma biblioteca de código aberto e que possui mais de 500 funções de processamento, segmentação, calibração e filtragem de imagens. A OpenCV torna o desenvolvimento de algoritmos de processamento de imagens uma tarefa rápida até mesmo para desenvolvedores inexperientes no campo de visão computacional.

Segundo (OPENCV, 2015a), dentre todos os métodos de segmentação de imagens disponíveis na biblioteca OpenCV, o *threshold* é o mais simples mas é também muito eficiente. O método consiste em diferenciar os *pixels* de interesse de todos os outros presentes numa imagem. Seu algoritmo realiza uma comparação com a intensidade de cada pixel com base numa "fronteira"estipulada pelo desenvolvedor. Um exemplo dessa segmentação é vista na figura 7.

A segmentação por *threshold* pode ser descrita pela equação 2.5.

$$dst(x,y) = \begin{cases} maxVal & \text{se } src(x,y) > threshold \\ 0 & \text{se o contrário} \end{cases} \tag{2.5}$$

Uma imagem fonte $src(x,y)$ cuja intensidade dos pixels segue uma distribuição típica pode ser representada como no gráfico da figura 8.

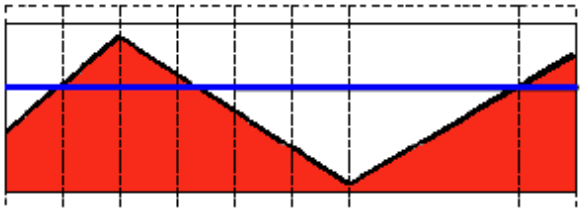

Figura 8: Representação da intensidade dos pixels da imagem fonte (OPENCV, 2015a).

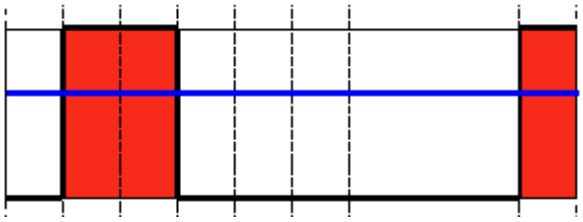

Figura 9: Representação da intensidade dos pixels da imagem fonte após o processo de *threshold* (OPENCV, 2015a).

A linha azul representa a "fronteira"ou *threshold* de intensidade a ser considerada na segmentação. Ao aplicar a equação 2.5, a imagem é segmentada de acordo com essa fronteira e a distribuição de intensidade dos *pixels* pode então ser exemplificada pela figura 9.

Após a segmentação, tem-se uma matriz de *pixels* binária onde cada *pixel* abaixo da fronteira é igualado a zero (o que representa a cor preta na imagem) e os *pixels* acima da fronteira são igualados a 255 (o que representa a cor branca na imagem). Finalmente, é possível percorrer a matriz de forma a obter qualquer informação que seja de interesse. Percorrendo os *pixels*, é possível realizar uma conversão do tipo *pixels*-milímetros obtendo-se assim medidas com significados reais para a medição de grandezas como larguras, áreas e diâmetros.

2.4 O controle

Atualmente, os controladores mais utilizados são os do tipo PID e suas variações. Seu projeto é de relativa facilidade bem como sua implementação. Ocorre que a teoria dos controladores PID foi desenvolvida para controlar sistemas contínuos no tempo. Em sistemas digitais, a taxa de amostragem é de importância fundamental no projeto de tais controladores, podendo até mesmo levar um sistema estável à instabilidade caso essa taxa seja alterada. Segundo (LAGES, 2010), por definição, a expressão do controlador PID contínuo é dada por:

$$u(t) = K_P\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_D \frac{d}{dt} e(t) \tag{2.6}$$

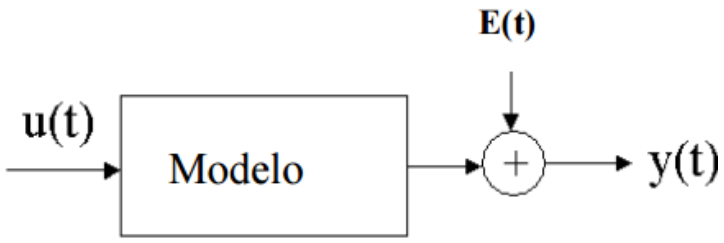

Figura 10: Topologia da malha de controle de diâmetro (BRANDOLT, 2002).

É possível ver que tal implementação não considera a taxa de amostragem. Entretanto, uma vez que o código dos controladores do presente trabalho serão executados pelo Arduino, que é um dispositivo digital, faz-se necessária a adaptação da equação 2.6 para o domínio discreto. Ainda segundo (LAGES, 2010) tal adaptação pode ser feita utilizando a aproximação *backward differences* de forma a obter o que é chamado de controlador PID ótimo. A equação 2.7 mostra tal implementação.

$$u(k) = K_P\, e(k) + K_i T \sum_{i=0}^{k} e(i) + K_D \frac{1}{T}(e(k) - e(k-1)) \tag{2.7}$$

O problema então se resume a obter os valores de K_P K_i e K_D de forma que o desempenho seja o melhor possível. Isso é feito através de técnicas de projeto de controladores como *root locus* ou lugar das raízes, análise de entrada em degrau, PRBS *pseudorandom binary sequence*, dentre outros. No presente projeto foi utilizado majoritariamente o método da locação de zeros e polos pelo lugar das raízes.

Foi utilizada também a aproximação ARX[1] para a estimação do modelo que descreve o comportamento do diâmetro do filamento para uma entrada em degrau. Segundo (AGUIRRE, 2007), um modelo ARX é dado por:

$$y(k) = a1y(k-1) + \cdots + a_{n_y} y(k-n_y) + b_1 u(k-1) + \cdots + b_{n_u} u(k-n_u) \tag{2.8}$$

A topologia desse modelo pode ser vista na figura 10

Segundo (BRANDOLT, 2002), o modelo também pode ser descrito como:

$$y(t) = G(q)u(t) + H(q)e(t) \tag{2.9}$$

[1] auto-regressivo com entrada exógena

Sendo que:

$$G(q) = \sum_{k=1}^{\infty} g(k)q^{-k} \quad \text{e} \quad H(q) = 1 + \sum_{k=1}^{\infty} h(k)q^{-k} \tag{2.10}$$

Na equação 2.10 G(q) e H(q) são os parâmetros a serem estimados de forma a modelar o sistema satisfatoriamente. Tal estimação pode ser feita usando diferentes métodos. Entretanto, neste trabalho será utilizada apenas a estimação pelo método dos mínimos quadrados. Esse método define os parâmetros da equação 2.8 de forma que a equação resultante seja o melhor ajuste para o conjunto de dados fornecido.

3 Metodologia

Foi desenvolvida uma pequena extrusora para a produção do filamento e realização dos testes necessários para a validação do controle e do sensoreamento. A planta pode ser vista na figura 11.

Após a extrusão, o filamento passa por uma *webcam* que lê o diâmetro através do emprego de métodos de visão computacional. Após a leitura, o filamento segue em direção ao puxador para que este ajuste o diâmetro do filamento conforme desejado.

A comunicação é feita inteiramente através do protocolo Serial que é emulado tanto pelo Qt Creator quanto pelo arduino através da porta USB do computador.

O projeto foi dividido em quatro grandes áreas sendo elas o construção mecânica da planta, a eletrônica/elétrica de acionamento, a programação e o controle.

3.1 Montagem mecânica

A montagem mecânica foi baseada no projeto original de (LYMAN; MULIER, 2014). A tabela 1 descreve os itens utilizados para realizar a construção da planta.

Além desses materiais, toda a estrutura de sustentação da extrusora foi impressa em impressora 3D. A montagem das peças foi feita de forma a minimizar os efeitos do

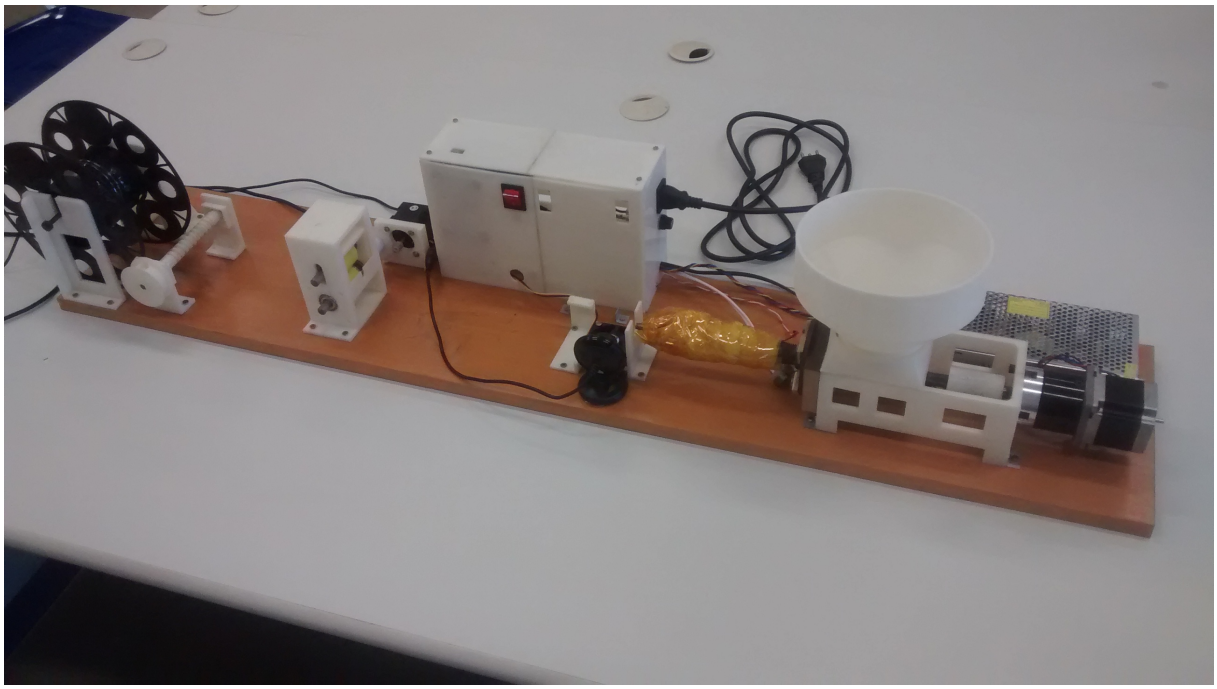

Figura 11: A extrusora.

Quantidade	Descrição
1	Tubo de aço negro de 1/2"x 4"
1	Broca de aço rápido BOSCH 5/8 x 17"x
1	Bico de mangueira de latão de 1/2"x 1/4"NPT
1	Acoplador 1/2"x 4"
1	Flange de aço negro de 1/2"
1	Bloco de madeira para isolamento térmico
2	Rolo de uretano
4	Rolamentos R6ZZ de 3/8"
1	Rolamento axial 1/2"
1	Eixo de aço 3/8"
1	Lã de vidro para isolamento térmico

Tabela 1: Lista de peças mecânicas da extrusora.

processo como atrito do plástico no tubo extrusor através da centralização da broca com o tubo extrusor por exemplo. Foram adicionadas arruelas em todos os parafusos passantes em peças impressas para diminuir a folga devida a compressão da peça, uma vez que por serem feitas de plástico, a estrutura pode ser deformada devido às altas forças decorrentes do processo de extrusão.

O isolamento térmico no tubo extrusor também é de enorme importância para o processo. O isolamento contribui para a eficiência energética da máquina além de ajudar a manter a temperatura constante e mais insensível as condições do ambiente de trabalho da extrusora.

É importante frisar que das peças citadas, a broca foi a única a sofrer algum tipo alteração. A broca foi cortada a uma distância de 267 mm da base hexagonal para poder ser alojada dentro do tubo extrusor.

3.2 Eletrônica e elétrica

O projeto elétrico também foi baseado no projeto de (LYMAN; MULIER, 2014). Os materiais são descritos na tabela a seguir.

A maior parte dos materiais descrito encontra-se alojados dentro da caixa de eletrônicos. Uma visão mais detalhada da caixa pode ser vista na figura 12.

A fiação, por se tratar de ligações elétricas simples, não está no escopo deste trabalho e assim não será coberta.

A potência no aquecedor é controlada conforme uma frequência de pulsos de chaveamento (PWM). Para realizar esse chaveamento, foi utilizado um relé de estado sólido que, por se tratar de um chaveamento eletrônico, consegue chavear uma potência relativamente alta em alta frequência desde que seja respeitados os limites do componente,

Quantidade	Descrição
1	Sensor de temperatura NTC de 100KΩ
1	Aquecedor tubular 25mmx30mm 110V 300W
1	Arduino MEGA 2560
1	Shield para arduino RAMPS 1.4
2	Drivers para motor de passo DRV8825
1	Relé de estado sólido
1	Fusível 15A
1	Fonte bivolt de 12V
1	WebCam 640x480px
1	Botão liga/desliga
1	Motor de passo 57STH56 NEMA 23 com reduçao de 15:1
1	Motor de passo NEMA 17 com redução de 4:1
1	Cooler para resfriamento da caixa de eletrônicos
1	Cooler para resfriamento do filamento

Tabela 2: Lista de peças elétricas e eletrônicas.

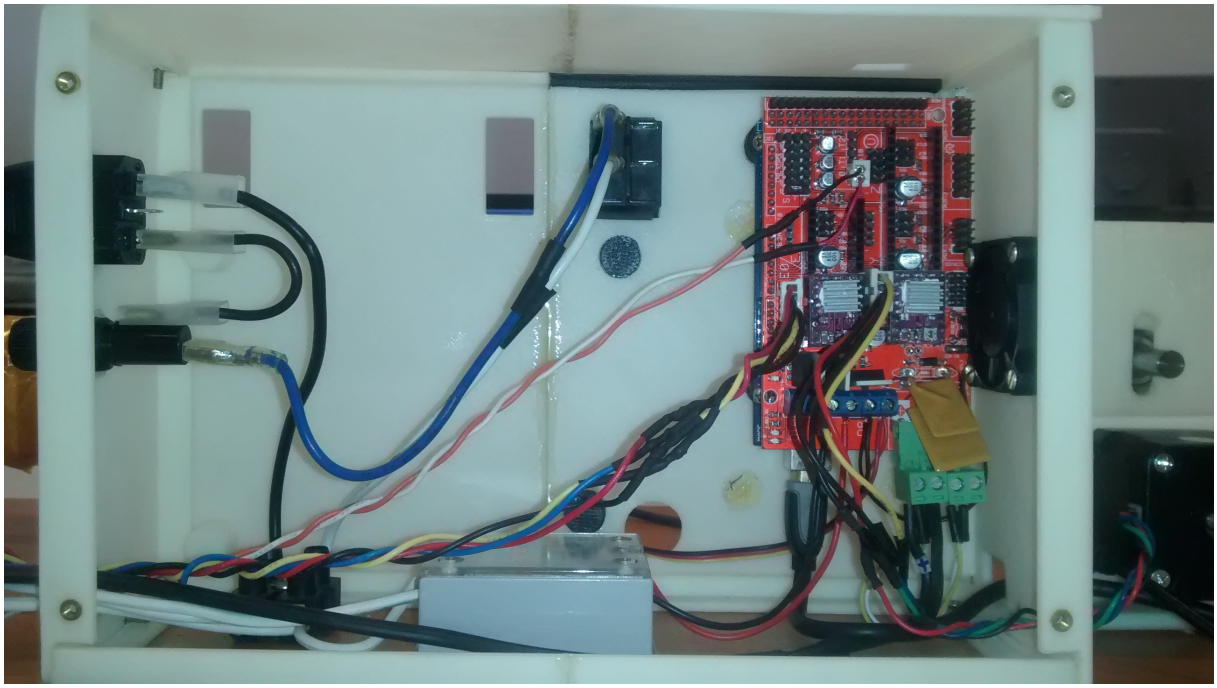

Figura 12: A caixa de eletrônicos.

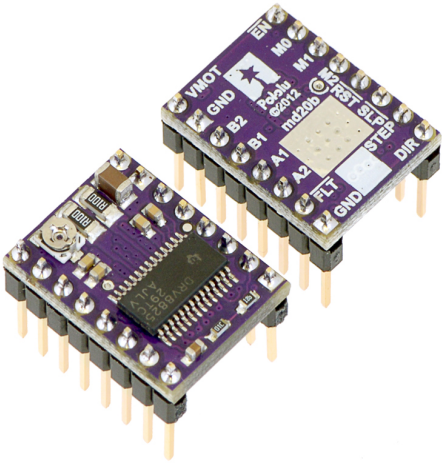

Figura 13: O *driver* de motor de passo DRV8825. (POLOLU, 2015)

que no caso do relé utilizado, é de 10ms para a estabilização em nivel lógico alto e baixo. Assim, a programação da frequência do PWM originada a partir do controlador projetado para a malha de controle de temperatura, foi feita de forma a nunca exceder tais limites.

3.2.1 Os motores de passo

A extrusora conta com dois motores de passo. Um para realizar a extrusão do filamento e um para realizar o controle do diâmetro através do puxador. Os motores de passo são facilmente controlados utilizando um *driver* de motor de passo como o que foi utlizado nesse projeto, um DRV8825 da Pololu que realiza o controle de corrente além de fornecer circuitos de proteção e seleção de micropasso, e o *shield* para arduino RAMPS 1.4 que cuida de toda alimentação e fornece todas as ligações, restando ao programador apenas ajustar o potênciometro presente no *driver* e programar a frequência de $STEP$ que será a velocidade de rotação dos motores. O driver pode ser visto na figura 13.

Segundo (POLOLU, 2015), a corrente máxima em um motor de passo controlado pelo DRV8825 pode ser ajustada conforme a seguinte equação:

$$I_{max} = V_{ref} \cdot 2 \tag{3.1}$$

Onde V_{ref} é a tensão medida em cima do potênciomentro com a referência no GND do driver. No caso do motor extrusor a corrente foi experimentalmente ajustada para 2A. Já para o motor do puxador, a corrente também foi experimentalmente ajustada para 1.2A.

3.2.2 O sensor de temperatura

Foi utilizado um sensor de temperatura do tipo NTC da Musor como o da figura 14 para realizar a amostragem da temperatura atual do barril de extrusão. Uma vez que esse sensor não é linear, se faz necessária a conversão da tensão lida segundo a equação 3.2.

Figura 14: O sensor de temperatura NTC da Musor. (MUSOR, 2014)

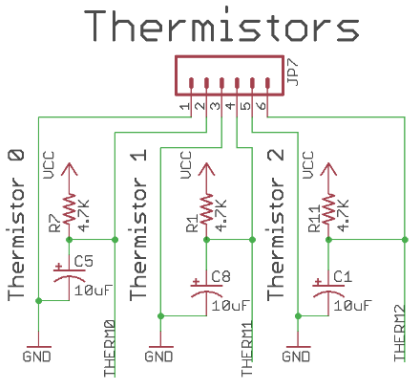

Figura 15: O circuito do termistor. (REPRAP, 2014b)

Este sensor é geralmente utilizado como uma das resistências de um divisor de tensão. Como foi dito, o presente trabalho utiliza a placa RAMPS 1.4 da RepRap para facilitar as conexões. O diagrama elétrico das conexões de sensores da placa pode ser visto na figura 15 Assim, é necessário saber o valor da resistência em série com o sensor.

$$T_{atual} = \frac{1}{\ln \frac{R}{R_0} \cdot \frac{1}{\beta} + \frac{1}{(T_0+273,15)}} - 273,15 \tag{3.2}$$

Onde T_0, R e R_0 são a temperatura nominal, a resistência em série e a resistência nominal do sensor e β é um parâmetro característico de cada sensor fornecido pelo fabricante. A temperatura fornecida pela equação 3.2 é fornecida em graus Celsius.

3.3 Programação

A programação da extrusora pode ser isolada em dois programas principais: o programa que é executado no computador fazendo a interface homem-maquina e enviando

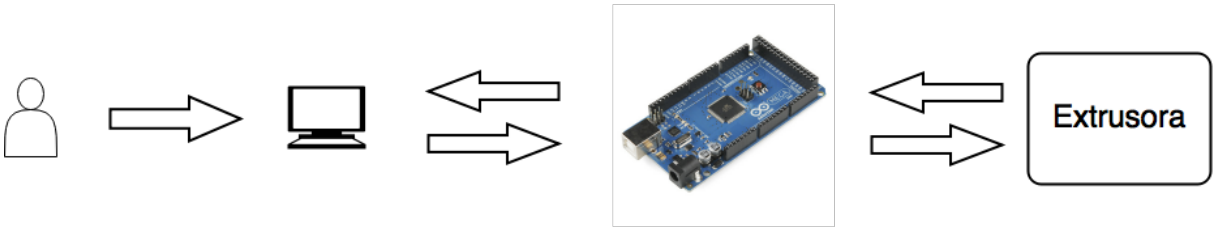

Figura 16: Topologia de interação dos programas.

os dados da câmera para o Arduino e o programa que é executado no Arduino que realiza os acionamentos assim como é responsável por executar os *loops* de controle do sistema. A figura 16 mostra a topologia de interação do usuário e dos programas com a extrusora.

É possível perceber que todo o controle e acionamento da extrusora é feito exclusivamente a partir de um computador que comanda toda a operação. Fato que diferencia a extrusora desenvolvida ao longo desse projeto das extrusoras convencionais, uma vez que o operador não precisa sequer estar na mesma sala que a extrusora. Tal fato se mostra conveniente para processos de extrusão de termoplásticos, uma vez que gases tóxicos podem ser liberados ao longo do processo. O usuário pode então deixar a extrusora em algum local arejado por exemplo e realizar todo o acionamento e monitoramento a distância de outra sala caso isso seja de interesse.

3.3.1 A interface homem-máquina

A interface do programa foi desenvolvida no software Qt Creator utilizando a linguagem C++. O programa foi desenvolvido para servir como uma interface homem-máquina de forma a permitir um acionamento rápido de todos os recursos da extrusora. A figura 17 mostra uma imagem do programa.

Para um melhor entendimento de todos os métodos disponíveis para o usuário, foi desenvolvido um diagrama UML contendo as principais classes do programa e suas relações. O diagrama pode ser visto na figura 18.

3.3.1.1 O painel de testes

O painel de testes foi criado para um completo controle sob todas as funções da extrusora. Em quanto no painel principal o usuário fica restrito ao acompanhamento dos estados da máquina e o acionamento de funções básicas como o início da produção, botões de emergência e conectividade, o painel de testes foi desenvolvido para dar total liberdade ao operador caso seja necessário algum acionamento não usual como inversão do sentido de giro e controle manual de velocidade dos motores. Caso a máquina apresente algum problema e seja necessário algum tipo de teste, o operador fica livre para realizar qualquer tipo de operação podendo até mesmo desabilitar o dispositivo de segurança que impede

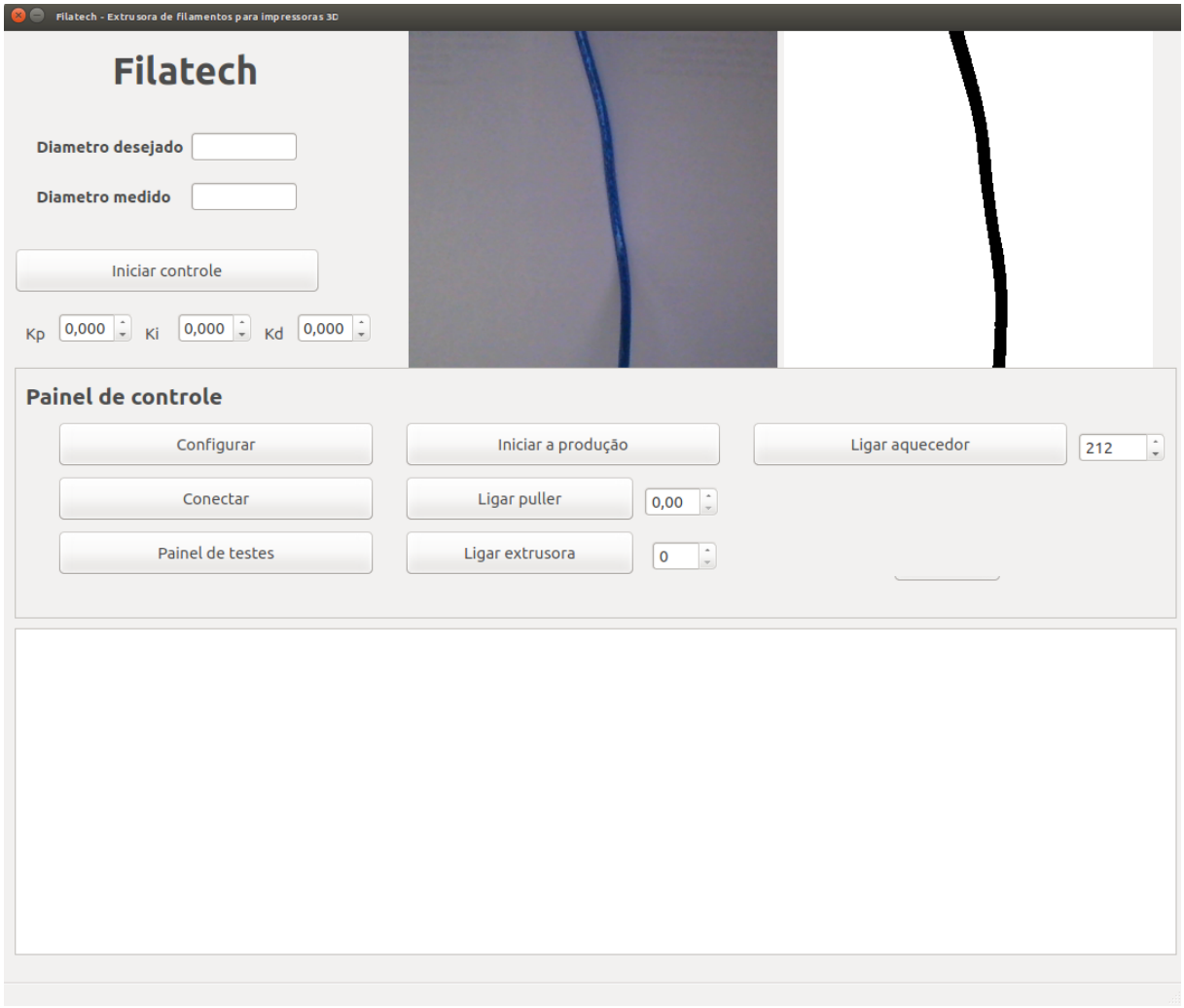

Figura 17: O programa feito utilizando o QtCreator.

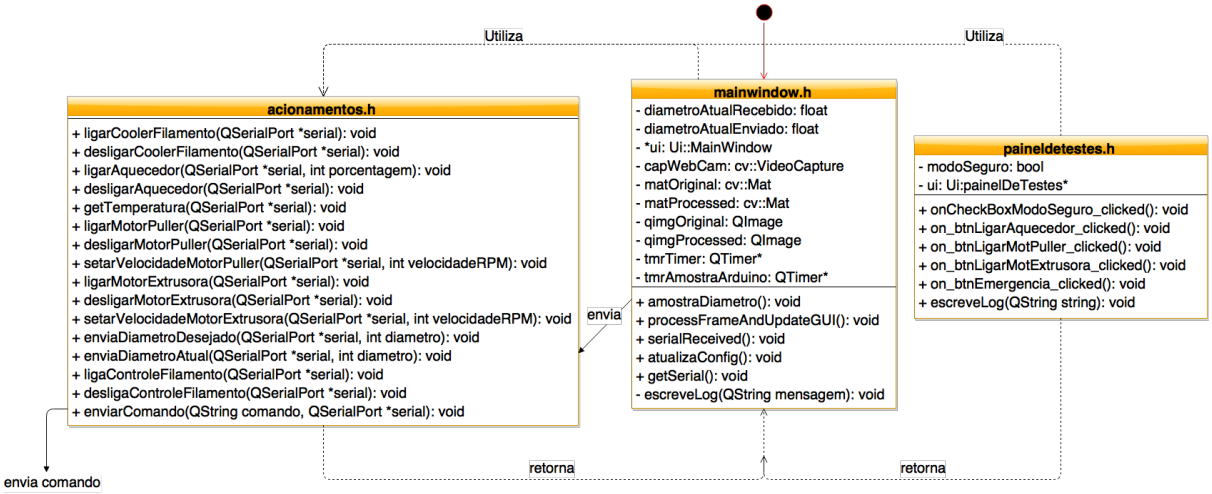

Figura 18: Diagrama UML das principais classes do programa.

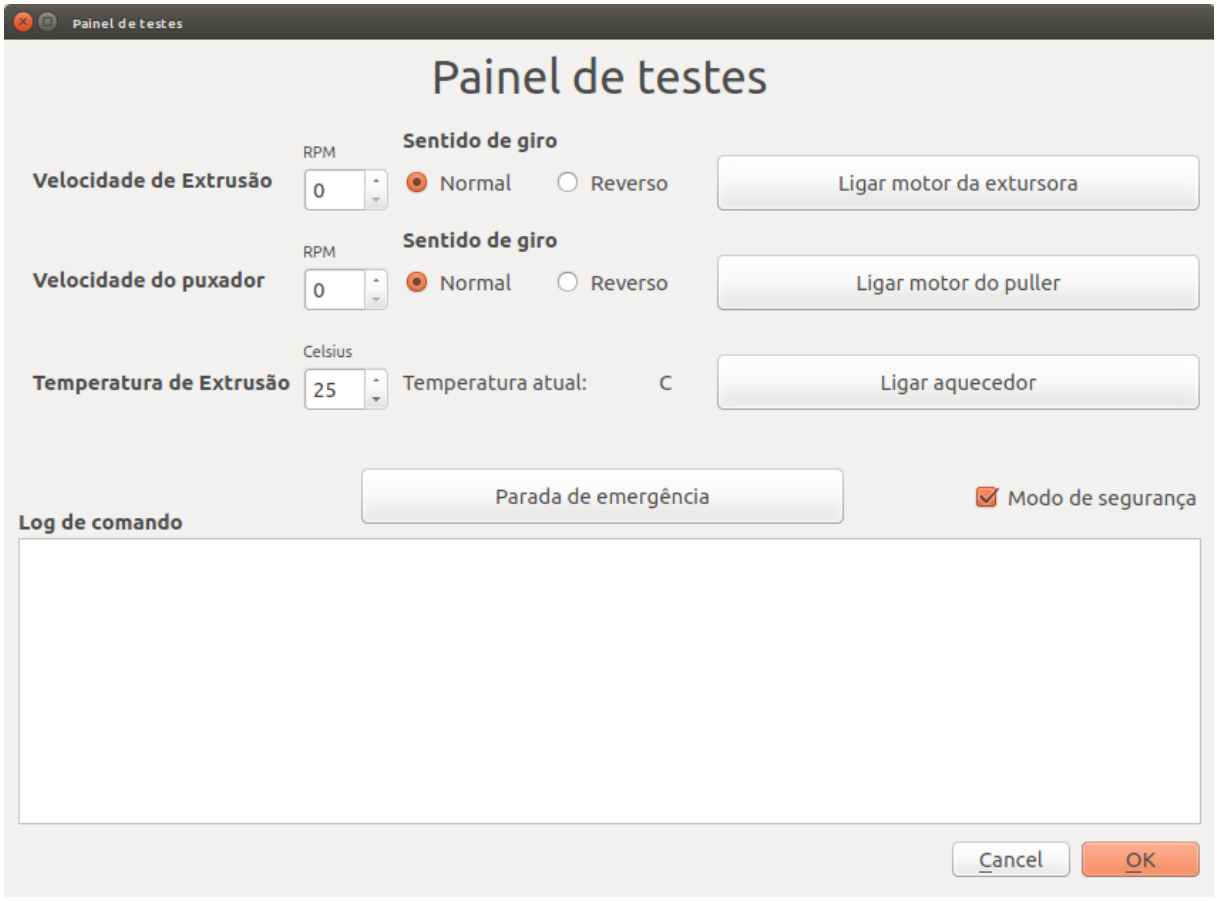

Figura 19: O painel de testes.

que o motor da extrusora gire caso a temperatura do barril de extrusão esteja abaixo do permitido, o que pode causar danos à maquina. Essa temperatura foi experimentalmente *setada* em 175^oC uma vez que notou-se que nessa temperatura o motor da extrusora começa a perder passos devido à força necessária para a extrusão ser acima de sua capacidade delimitada pelo controlador de corrente do driver. O painel de testes pode ser visto na figura 19.

3.3.2 O Arduino

O programa do Arduino foi desenvolvido em C++ e como dito anteriormente, é responsável por receber os comandos de acionamento do PC e executar os *loops* de controle agindo como um cliente do programa executado a partir do PC. O fluxograma pode ser visto na figura 20.

Além dos acionamentos o arduino também é responsável por executar o *loop* de controle de temperatura. A figura de código 3.1 mostra o método que realiza o controle de temperatura.

No código 3.1 é feita a implementação de um controlador PI digital. Logo após a

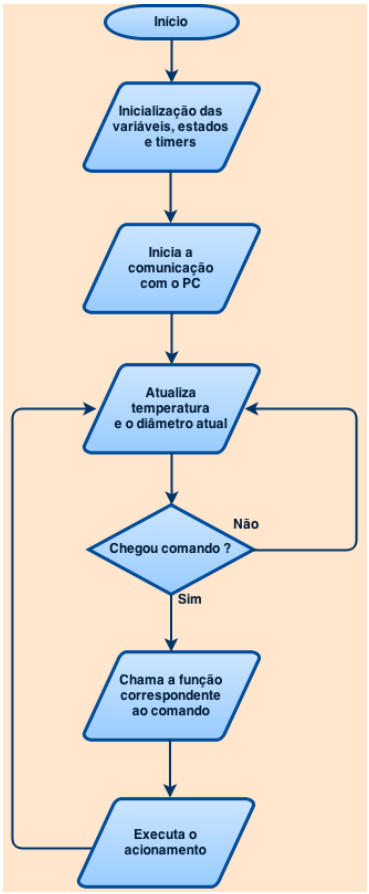

Figura 20: Fluxograma do programa executado no Arduino.

inicialização das variáveis é feito o calculo do erro atual de temperatura. Sendo u o sinal de controle, a implementação do controlador segue a implementação do controlador PID ótimo baseado no método de *backward differences* como já foi explicado anteriormente. O tempo de amostragem é uma constante definida no começo do código que contém o valor do tempo de amostragem em segundos. Para o caso do controle de temperatura, esse tempo é de um segundo.

Depois de recalcular o valor do sinal de controle, é feita uma verificação de saturação do mesmo. Uma vez que o sinal de controle é o valor em porcentagem do PWM a ser utilizado no acionamento do aquecedor tubular, este valor não pode ser nem menor que zero nem maior que cem. Caso alguma dessas condições ocorra, o controlador é saturado e o calculo da integral não é feito. Essa técnica, conhecida como *anti-windup*, consiste em manter o valor da integral do erro em valores aceitáveis caso o sinal de controle seja muito diferente do esperado. Isso é importante principalmente em sistemas lentos como os de temperatura pois caso o calculo da integral continue sendo feito, o sistema poderá ser facilmente levado à instabilidade uma vez que ao atingir valores próximos da referência, o termo integrador estará muito grande para que o controle possa ser executado da maneira prevista. Após esses cálculos, a variável que contem o erro anterior a que contem o valor

```
1  void controlaTemperatura() {
2
3    float kp = 2.4;
4    float ki = 0.008;
5    float u = 0;
6    static float integral = 0;
7
8    float erroAtual = temperaturaDesejada - temperaturaAtual;
9
10 // Implementacao do controlador PI digital
11   u = kp * erroAtual +
12                 ki * (integral + erroAtual) * TEMPODEAMOSTRAGEM;
13
14 // Implementacao do anti-windup
15   if (u > 100) {
16       u = 100;
17   }
18   else if (u < 0) {
19       u = 0;
20   }
21   else {
22       integral = integral + erroAtual;
23   }
24
25 // Atualizacao do erro
26   erroAnterior = erroAtual;
27
28 // Atualizacao do valor de pwm a ser utilizado na planta
29   pwmVal = (int)u;
30 }
```

Código 3.1: Método de controle de temperatura

do PWM a ser utilizado são finalmente atualizadas.

O método de implementação do controle de diâmetro segue a mesma estrutura do controlador de temperatura e pode ser visto na figura de código 3.2.

A diferença entre os dois controladores está apenas na saturação do sinal de controle. Em quanto o controlador de temperatura realiza a atualização do valor da integral apenas se a saturação do sinal de controle não tiver ocorrido, o controlador de diâmetro o faz a cada iteração do sinal de controle. Isso se deve ao fato de que neste caso, a saturação do sinal de controle do integrador dificilmente ocorreria em condições de operação normais e mesmo que isso ocorra momentaneamente devido a algum erro no reconhecimento do diâmetro atual do filamento, por se tratar de um sistema relativamente rápido, o calculo de integral do erro não seria drasticamente diferente do esperado.

```
1  void controlaDiametro() {
2
3    float u = 0;
4    float kpDiam = -0.07;
5    float kiDiam = -0.004;
6    float erroAtual = diametroDesejado - diametroAtual;
7
8    u = kpDiam * erroAtual + kiDiam * (integralDiam + erroAtual)
9            * TEMPODEAMOSTRAGEMDIAMETRO;
10
11   if (u > 15) {
12       u = 15;
13   }
14   else if (u < 1) {
15       u = 1;
16   }
17   integralDiam = integralDiam + erroAtual
18                  * TEMPODEAMOSTRAGEMDIAMETRO;
19
20   erroAnteriorDiam = erroAtual;
21   RPMpuller = (float)u;
22   atualizaRpmPuller();
23 }
```

Código 3.2: Método de controle de temperatura

É importante frisar que ambos os *loops* de controle são executados a partir do estouro de um *Timer* exclusivo para cada malha para garantir o tempo de amostragem fixo.

Para realizar os acionamentos, foi criado um protocolo baseado em linguagem G. A figura de código 3.3 mostra os acionamentos permitidos e seus respectivos códigos.

No código 3.3 é possível ver também os códigos que realizam também a alteração de parâmetros como as velocidades dos motores e dos valores de diâmetro atuais. Foi criado um padrão de comunicação onde os parâmetros devem ser separados do código por um espaço e conter sempre três dígitos representados pelas letras X nos códigos G32, G42, G70 e G71.

3.3.3 A visão computacional

A utilização da visão computacional vem como uma solução de sensoriamento de baixo custo. Através dela é possível realizar a segmentação da imagem de forma a isolar o filamento do fundo de uma imagem, realizar a contagem dos *pixels* e sua conversão em milímetros para obter então a medida do diâmetro do filamento.

```
G20  // Liga o aquecedor
G21  // Desliga o aquecedor
G30  // Liga o motor da extrusora
G31  // Desliga o motor da extrusora
G32 XXX // Seta a velocidade do motor da extrusora
G40  // Liga o motor do puller
G41  // Desliga o motor do puller
G42 XXX // Seta a velocidade do motor do puller
G50  // Liga os ventiladores
G51  // Desliga os ventiladores
G70 XXX // Seta o diametro desejado do filamento
G71 XXX // Seta o diametro atual do filamento
G72  // Liga o controle de diametro do filamento
G73  // Desliga o controle de diametro do filamento
```

Código 3.3: Lista de códigos de acionamentos

O código 3.4 mostra a implementação do método de segmentação de imagem utilizando visão computacional. O código é parte integrante do programa que realiza a interface homem-máquina.

Primeiramente é utilizado o método *cvtColor* para converter a imagem adquirida da câmera para uma escala de cinza. Será a partir desta escala de cinza que será executado o método de segmentação para o reconhecimento do filamento produzido. Logo após a conversão, é utilizado o método *blur* que, segundo (OPENCV, 2015b) tem como objetivo realizar uma filtragem na imagem fazendo com que o resultado da segmentação seja mais eficiente ao facilitar a detecção das bordas da imagem.

Logo após o *blur* é utilizado o método *threshold*. Dentre os tipos de filtragem disponíveis para o *threshold*, escolheu-se utilizar o método binário já explicado anteriormente devido à sua simplicidade.

Logo após a segmentação, é implementado o código que percorre toda a matriz de intensidade de *pixels* em busca do diâmetro do filamento. Isso é feito buscando as mudanças de intensidades. O algoritmo guarda os *pixels* onde houve transições de intensidade (do branco para o preto ou do preto para o branco) e pela diferença de posição desses *pixels* consegue obter a quantidade de *pixels* que representa o diâmetro do filamento. Com isso em mãos, é finalmente feita a conversão de *pixels* para milímetros na linha 29 do código. A equação da reta que representa a conversão foi obtida experimentalmente ao dispor sob o mesmo plano do filamento um objeto de tamanho conhecido. Realizando a contagem de pixels que esse objeto ocupava na tela, é então possível obter a reta que realiza a relação entre *pixels* e milímetros.

```
1  // Convertendo a imagem para uma escala de cinza e
2  // realizando uma filtragem
3  cvtColor( src, src_gray, COLOR_BGR2GRAY );
4  blur( src_gray, src_gray, Size(3,3) );
5
6  // Detectando as bordas do filamento usando o threshold
7  threshold(src_gray,threshold_output,thresh,255,THRESH_BINARY);
8
9  for(int i=0; i<IMAGEHEIGHT;i++){
10         for(int j=1; j<IMAGEWIDTH;j++){
11                 if(threshold_output.at<uchar>(j,i) !=
12                  threshold_output.at<uchar>(j-1,i)){
13                     vetor.append(j);
14                 }
15         }
16             if(vetor.size() == 2){
17                 diametro+= (vetor.at(1) - vetor.at(0));
18                 count++;
19         }
20         vetor.clear();
21 }
22 if(diametro != 0 && count != 0){
23     diametro /= count;
24 }else{
25     diametro = 0;
26 }
27
28 // Realiza a conversao de pixels para milimetros
29    float diametro_f=(float)diametro*2.81/131;
```

Código 3.4: Implementação do *threshold*

3.4 Controle

Para o projeto dos controladores foram utilizados o *software* MATLAB e seu pacote *Simulink*. No projeto da extrusora foi necessário o controle de duas variáveis independentes do processo: a temperatura do barril de extrusão e o diâmetro do filamento.

3.4.1 O controlador de temperatura

O controlador foi obtido a partir da resposta ao degrau do sistema. Uma potência de 50% de sua capacidade total foi fornecida ao aquecedor tubular e os dados foram adquiridos utilizando o próprio Arduino que realiza os acionamentos com um tempo de amostragem fixo de 1 segundo. O resultado pode ser visto na figura 21.

Nesta figura, é possível ver que o modelo da planta pode ser estimado a partir de um sistema de primeira ordem. Entretanto, experimentalmente notou-se que o sistema

Figura 21: Resposta do Sistema.

térmico possui um atraso no tempo da ordem de 28 segundos. Dessa forma, o sistema real será aproximado por um sistema de primeira ordem com atraso no tempo do tipo da equação 3.3.

$$H(s) = \frac{K}{\tau s + 1} e^{-ts} \tag{3.3}$$

Assim, a partir dos dados medidos e do gráfico de resposta ao degrau, tem-se:

$$T_0 = 24^oC \quad T_F = 239^oC \quad \Delta T = 215^oC \quad \Delta U = 50\% \quad t_0 = 36s \quad t_{63\%} = 513s$$

Onde T_0 é a temperatura inicial, T_F é a temperatura final, ΔT é a variação da temperatura, ΔU é a variação da entrada no sistema, t_0 é o tempo em qual ocorreu a primeira variação significativa no sistema e $t_{63\%}$ é o tempo necessário para que o a planta atinja 63% de sua variação total. Com esses dados, é possível então achar o ganho DC e a constante de tempo do sistema.

$$K = G_{DC} = \frac{\Delta T}{\Delta U} = \frac{215}{50} = 4,3\,(^{oC}/_{\%PWM}) \tag{3.4}$$

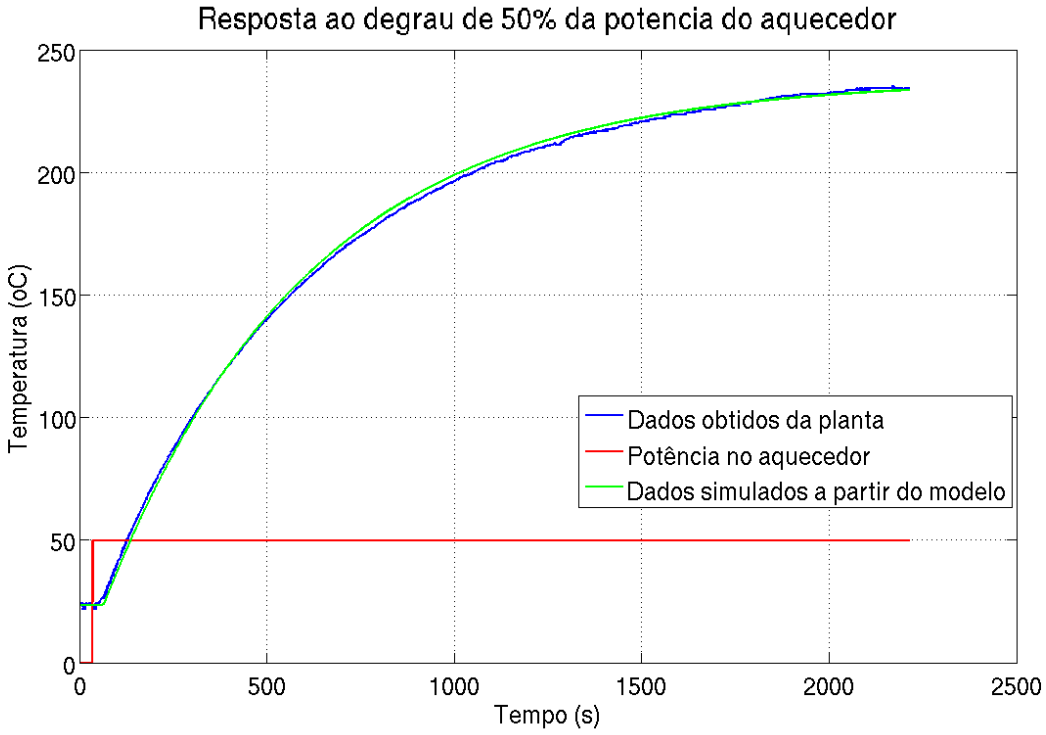

Figura 22: Comparação do sistema real com o modelo aproximado.

Já para a constante de tempo do sistema tem-se:

$$\tau = t_{63\%} - t_0 = 549s \tag{3.5}$$

Substituindo as equações 3.4 e 3.5 em 3.3 tem-se o modelo do sistema real aproximado por um sistema de primeira ordem acrescido de um atraso dado por:

$$H(s) = \frac{4,3}{549s + 1} \cdot e^{-28s} \tag{3.6}$$

Utilizando a equação 3.6 é possível comparar a proximidade do sistema estimado com o real. Isso é feito na figura 22.

É possível ver que o modelo simulado responde com bastante fidelidade ao sistema real tornando-o aceitável para o projeto e simulação de um controlador para a planta.

Para o projeto do controlador de temperatura foi utilizado o método do lugas das raízes. Para tanto foi utilizado a ferramenta *sisotool* do MATLAB que permite uma rápida análise de sistemas do tipo SISO (*Single Input Single Output*) facilitando assim o projeto de controladores para tais sistemas. Ao fornecer como parâmetro a equação 3.6, é possível alocar polos e zeros de forma a obter o melhor controlador possível para a planta. O resultado das alocações pode ser visto na figura 23.

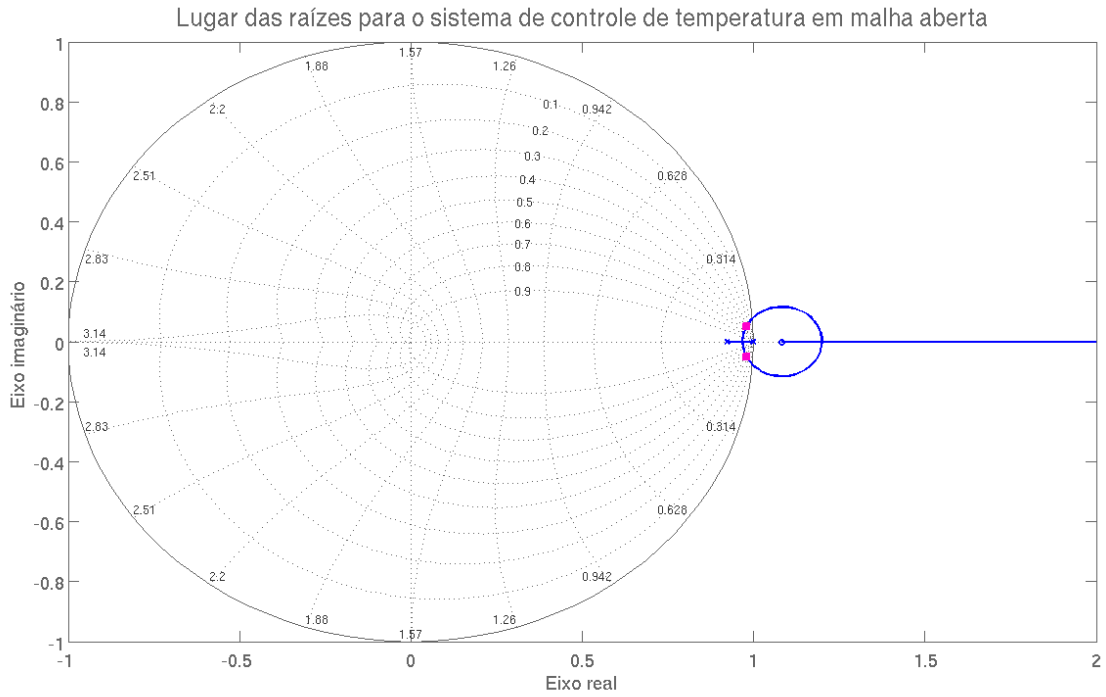

Figura 23: Projeto do controlador através do método do lugar das raízes.

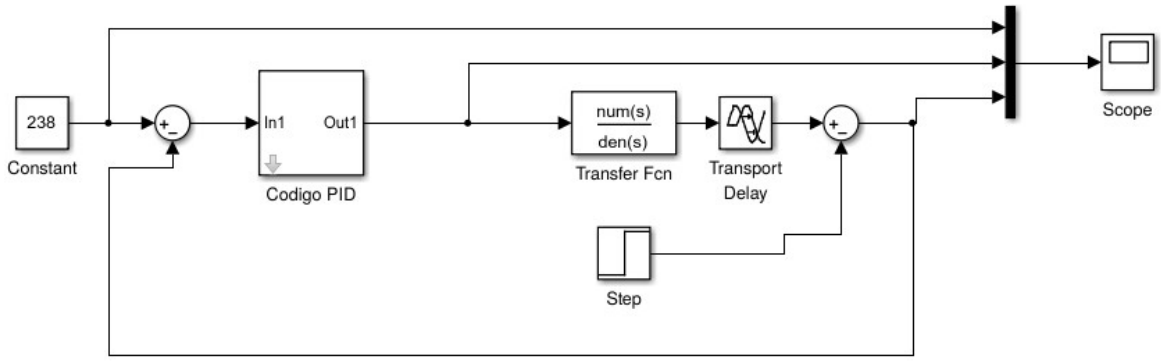

Figura 24: Topologia da malha de controle de temperatura.

Após a alocação dos polos, o controlador resultante é exportado para o *workspace* do MATLAB e adaptado para a equação de *backward differences*. O Controlador PI resultante é apresentado na equação 3.7.

$$C(z) = \frac{2,4z - 2.392}{z - 1} \tag{3.7}$$

Uma vez que o controlador foi projetado, é necessário então simular a resposta da planta sob ação do mesmo. Para tanto, foi simulado no *Simulink* a malha de controle da figura 24.

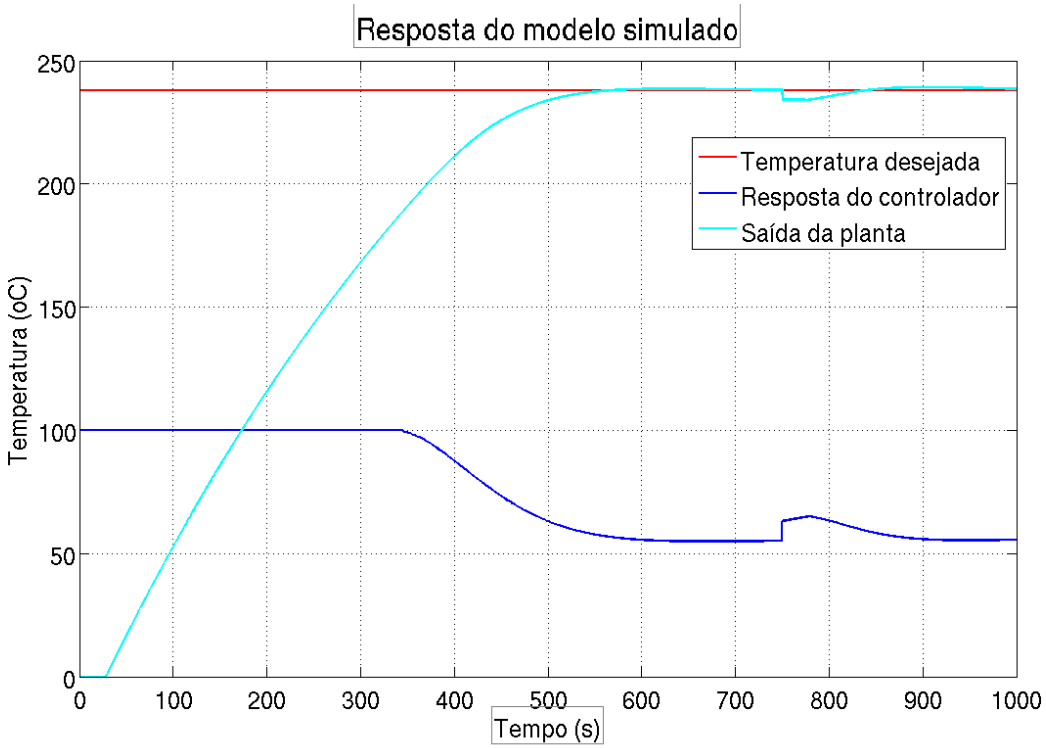

Figura 25: Resposta ao degrau no controle de temperatura.

Segundo (HAROLD JR.; WAGNER, 2005), cada plástico possui uma temperatura indicada de extrusão. Uma vez os autores não fornecem dados para o MABS, foi utilizada a temperatura de extrusão indicada para o ABS, uma vez que estes possuem propriedades físicas bastante parecidas. Assim, a simulação foi feita para uma temperatura desejada de 238^oC. Utilizando o modelo obtido anteriormente foi possível simular a resposta do controlador projetado para o sistema. A resposta pode ser vista na figura 25.

A figura 25 mostra como através da ação do controlador a planta manteve-se estável após atingir a temperatura desejada. Foi simulada também uma perturbação no sistema depois de esse ter atingido a estabilidade. A perturbação é simulada por um degrau externo em $t = 750s$. É possível ver que mesmo com a perturbação, o controlador consegue levar o sistema à estabilidade com um erro em regime permanente nulo, validando assim o controlador projetado.

3.4.2 O controlador de diâmetro

O controlador de diâmetro foi projetado seguindo uma estrutura um pouco diferente do controlador de temperatura. O levantamento do modelo da planta foi feito utilizando um sinal PRBS (*Pseudorandom binary sequence*) ao invés da aproximação por um sistema de primeira ordem. O resultado pode ser visto na figura 26.

A figura 26 mostra entretanto que o sistema é ligeiramente variável no tempo. As

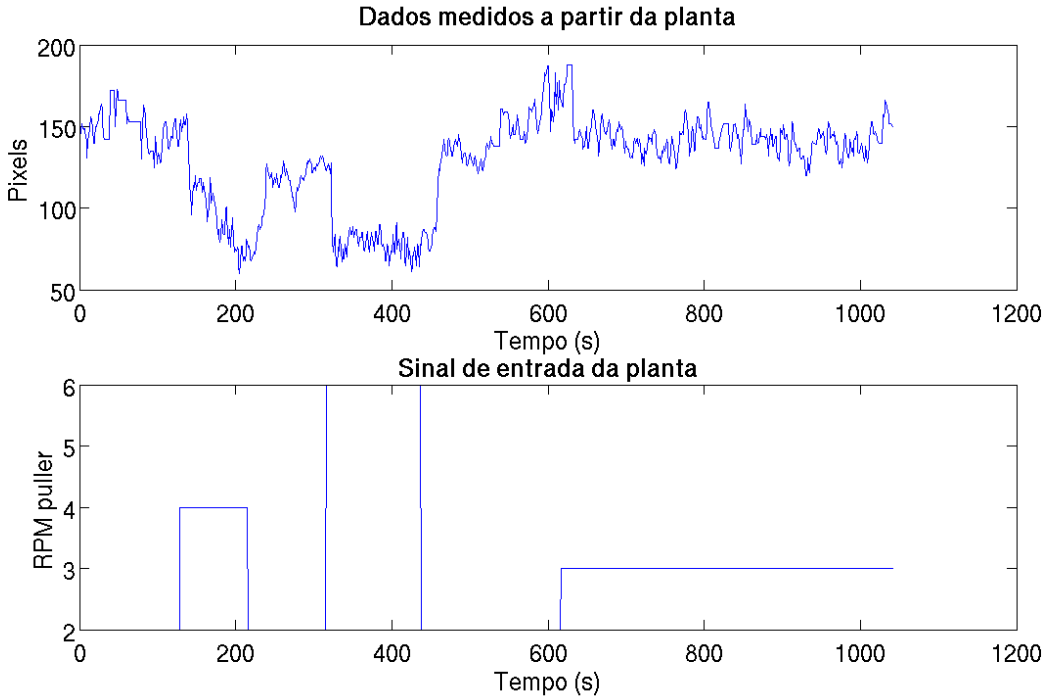

Figura 26: Resposta da planta a um sinal do tipo PRBS.

técnicas até então utilizadas no presente no projeto não conseguem controlar tais tipos de sistemas. Foram então propostas pequenas alterações na estrutura mecânica do projeto, como a forma de fixação da câmera, de forma a minimizar a influência de pequenos defeitos que acabavam por causar a variância no sistema e esta foi então desprezada sem grandes prejuízos ao sistema de controle.

Com os dados da resposta, é possível então realizar a identificação do sistema. Para tanto foi utilizado o método ARX do MATLAB que estima os parâmetros de um modelo ARX do tipo da equação 2.8 utilizando o método de mínimos quadrados. É desejado que a ordem do numerador e do denominador seja de ordem 2, além disso, experimentalmente verificou-se um atraso no sensoreamento de 3 segundos. Uma vez que o tempo de amostragem da câmera é de 1 segundo, o modelo foi feito levando em consideração um atraso de 3 tempos de amostragem. A figura 27 mostra uma comparação da saída real do sistema e do modelo simulado para uma mesma entrada.

É possível ver que o modelo simulado se assemelha bastante a saída do sistema real, sendo prejudicado apenas pela já citada variância no tempo. Entretanto, o modelo ainda é bastante fiel ao sistema real e será utilizado para o projeto do controlador. O controlador então foi desenvolvido com base no modelo gerado utilizando o método do lugar das raízes. A figura 28 mostra o lugar das raízes para o sistema de controle de diâmetro.

É importante notar que em quanto o controlador de temperatura foi desenvolvido usando técnicas de controle contínuo e apenas sua implementação foi dada no domínio

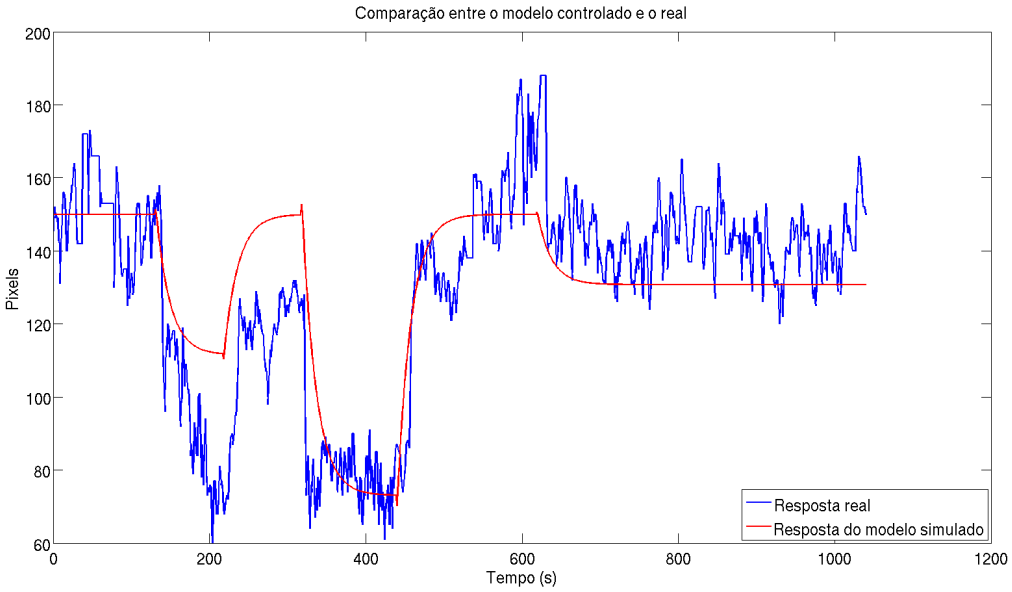

Figura 27: Comparação do modelo estimado e do sistema real.

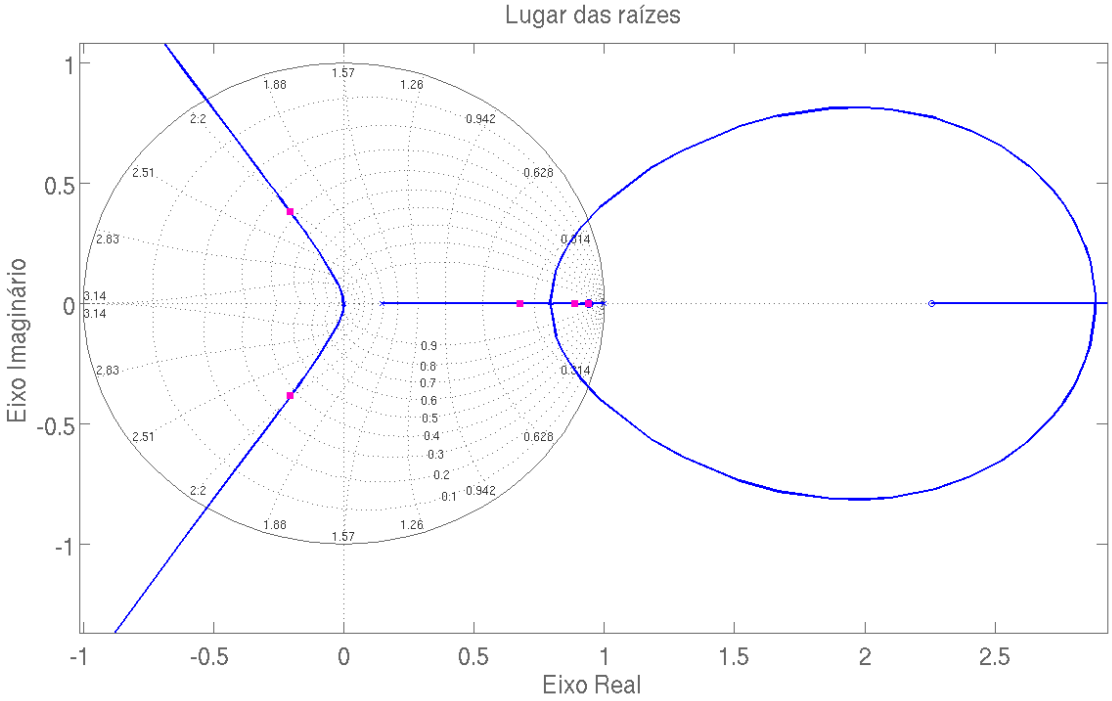

Figura 28: Lugar das raízes do modelo simulado.

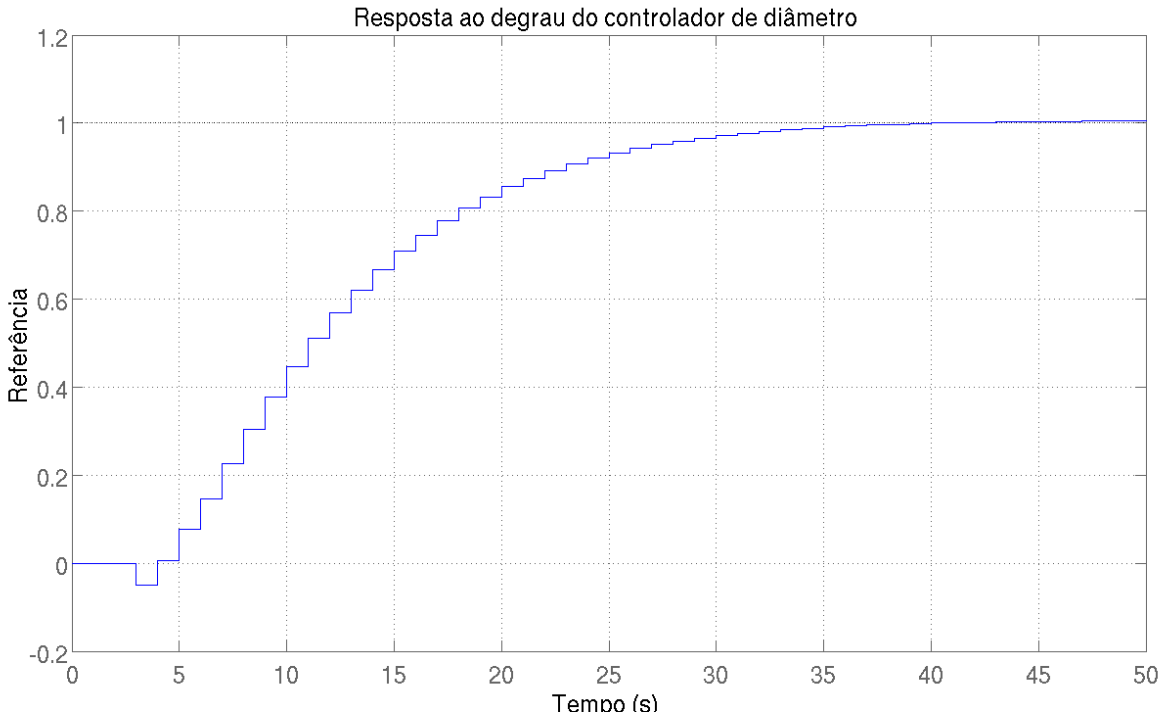

Figura 29: Resposta do sistema a uma entrada em degrau.

discreto do tempo, o método ARX do MATLAB só pode ser utilizado para sistemas discretos no tempo. Sendo assim, a análise do lugar das raízes é feita também no domínio discreto.

Após a alocação dos polos, o controlador de diâmetro resultante foi um PI e é representado pela equação 3.8.

$$C(z) = \frac{-0,07z + 0,066}{z - 1} \tag{3.8}$$

O sinal negativo se deve à ação inversa do controlador. Uma vez que o diâmetro medido seja maior que o desejado, o controlador precisa aumentar a ação de controle e não diminuir.

Após a obtenção do controlador, foi feita uma simulação da ação do controlador sobre a planta. A topologia da malha de controle pode ser vista na figura 30.

A resposta do sistema de controle pode ser vista na figura 31. É possível perceber como o controlador consegue retornar o diâmetro para o valor de referência de maneira satisfatória, validando assim o controlador para esta malha de controle.

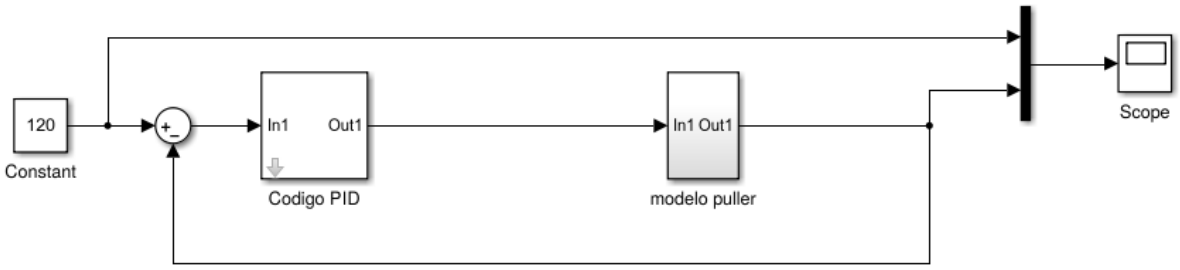

Figura 30: Topologia da malha de controle de diâmetro.

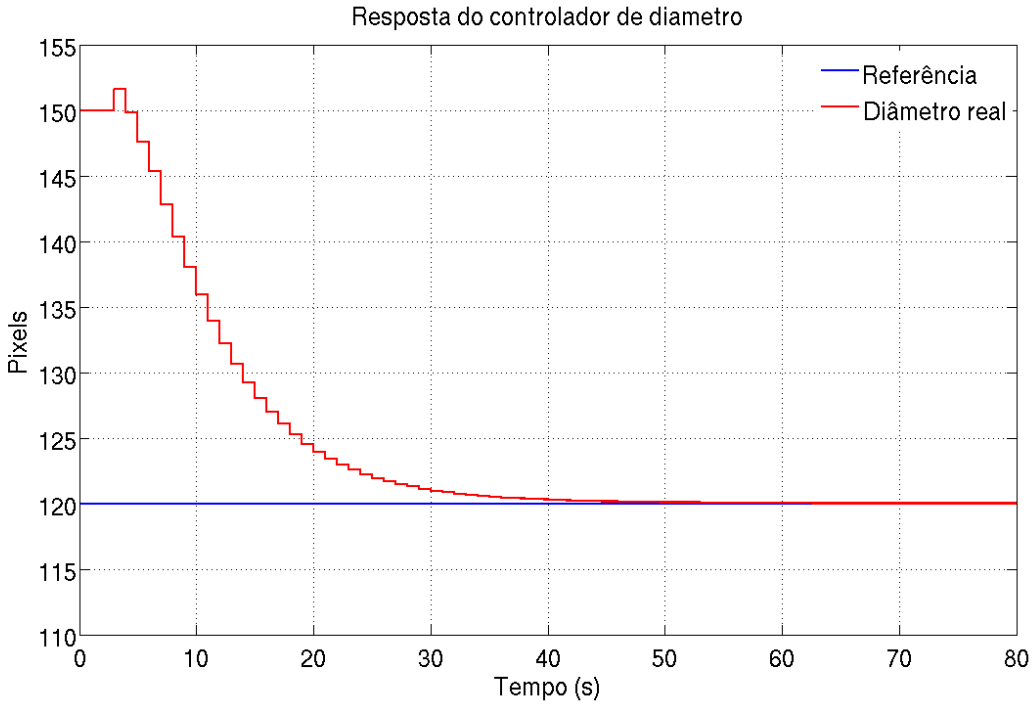

Figura 31: Resposta ao degrau no controle de diâmetro.

3.5 O processo

Para exemplificar uma produção de filamentos para impressora 3D foi escolhido o termoplástico MABS por ter características químicas e físicas bastante semelhantes ao ABS que normalmente é utilizado como material para filamentos. Entretanto, o MABS possui a vantagem de ser parcialmente translúcido, facilitando a análise de qualidade do material. O MABS escolhido já se encontra tingido de cor rubi, não sendo então necessária a adição de colorantes do tipo *masterbatch*, um famoso pigmento vendido em forma de granulado de forma a facilitar a adição deste ao processo de extrusão, facilitando assim o tingimento do material. O MABS pode ser visto na figura 32.

O processo de extrusão do plástico se resume a duas etapas: A preparação do material e a extrusão em si.

Figura 32: O MABS.

3.5.1 A preparação do material

A preparação do material é de suma importância para o processo, uma vez que esta é responsável pela padronização da produção e minimização de defeitos ocasionados por fatores externos. Dentre esses fatores, o mais preocupante é a umidade.

O MABS é um termoplástico altamente higroscópico, fazendo com que a umidade seja um fator determinante na qualidade do produto final. Caso as condições de armazenamento do plástico sejam inadequadas e o material seja exposto a altas taxas de umidade, tal umidade penetrará no plástico com relativa facilidade e, uma vez na câmara de extrusão, ao ser submetido à altas temperaturas, essa umidade tende a sair do plástico causando bolhas no mesmo. Além de causar graves defeitos estruturais nas peças impressas devido a falta de material ocasionada pela ocupação de espaço da bolha, o controle de diâmetro se torna impraticável, uma vez que com a variação de temperatura ao longo do processo, essas bolhas tendem a variar de tamanho ou até mesmo estourar, murchando o filamento naquele ponto. A imagem 33 mostra um filamento produzido sem a preparação do plástico contendo alto teor de umidade.

Vê-se então que tais fatores precisam ser controlados para garantir a qualidade do produto. Segundo (TEXCO, 2015) a preparação do MABS deve ser feita aquecendo o material a ser utilizado à uma temperatura de 80^oC por um período de 2 a 4 horas de forma a garantir a completa remoção da umidade.

Para isso, foi utilizado o laboratório de Materiais do CEFET-MG Campus Divinópolis. O laboratório é equipado com um forno mufla para pequenas preparações. O forno

Figura 33: Filamento produzido com material com alta taxa de umidade.

Figura 34: Forno utilizado para a preparação da matéria prima.

utilizado pode ser visto na figura 34.

Através da utilização do forno para o pré-aquecimento do plástico foi possível eliminar de forma satisfatória a umidade do filamento conforme pode ser visto na figura 35.

É possível notar claramente a diminuição da variância de diâmetro do filamento

Figura 35: Filamento produzido com material com baixa taxa de umidade.

sem que qualquer modificação no sistema de controle ou de extrusão tenha sido feita. É, entretanto, necessário mencionar que a preparação do plástico não é permanente. Processos de extrusão podem ser demorados e caso o plástico fique novamente exposto a umidade, uma nova preparação deve ser feita antes que esse possa ser de fato utilizado no processo.

3.5.2 A extrusão

Após a preparação do plástico, o material é adicionado ao bocal da extrusora para que o processo possa ter início. Apesar das recomendações do fabricante (TEXCO, 2015) e de (HAROLD JR.; WAGNER, 2005), notou-se experimentalmente que ao reduzir a temperatura recomendada de 238^oC para 210^oC, era possível ajustar o diâmetro do filamento com maior facilidade. Uma provável razão para isso é a de que em sistemas industriais, o filamento é resfriado de maneira mais eficiente através de tanques de água como o que pode ser visto na figura 36.

Dessa forma, é provável que o *cooler* utilizado para realizar o resfriamento do filamento não esteja conseguindo resfriar o filamento de forma satisfatória. Tal problema pode ser contornado de inúmeras formas como a diminuição da velocidade de extrusão, aumento da distância entre o *puller* e o bico da extrusora, diminuição da temperatura de extrusão ou troca do sistema de resfriamento. Foi decidido que abaixar a temperatura de extrusão seria a melhor opção para que a velocidade de produção do filamento e a estrutura já montada não fossem alteradas.

Além das principais variáveis do processo já citadas anteriormente, existem outras

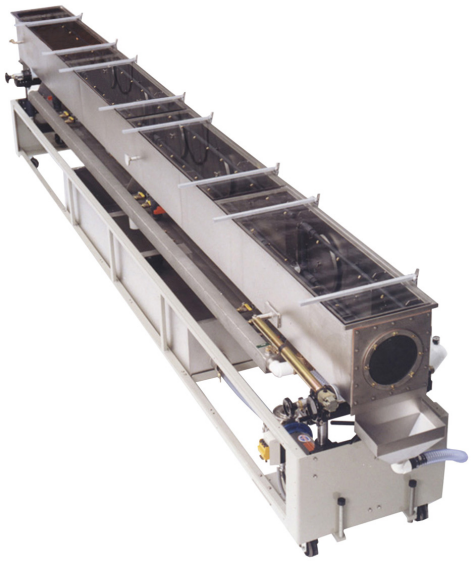

Figura 36: Tanque de resfriamento (CONAIRGROUP, 2014).

variáveis de importância secundária para o controle do diâmetro do filamento como a velocidade de extrusão dada pela rotação do motor de extrusão. Experimentalmente observou-se que uma velocidade de 10rpm é suficiente para garantir uma boa vazão de material pelo tubo ao mesmo tempo em que ainda é possível garantir que o plástico chegue ao sistema de *puller* frio o suficiente para que possa ser comprimido sem que haja alteração de seu diâmetro em qualquer direção.

Outro fator interessante é a razão entre a velocidade do *puller* e a velocidade de extrusão. Observou-se que quanto maior essa razão, melhor se torna a qualidade do filamento. Na prática o que acontece é uma maior uniformização do filamento, ficando mais fácil de garantir uma baixa variância de diâmetro, além de um melhor acabamento superficial.

4 Resultados e discussões

4.1 Tolerância dimensional

Para validar a tolerância dimensional do filamento foi proposta a seguinte metodologia:

- Os testes foram feitos utilizando um comprimento padronizado de filamento de 280mm.
- Foi feita uma medição no diâmetro a cada 5cm, totalizando assim 56 medições.
- O teste foi feito para uma referência de pixels de 120px.
- Todas as medidas foram realizadas com o auxílio de um micrômetro de resolução 0,01mm.

O resultado das medições pode ser visto na figura 37.

A figura 37 foi gerada pelo *software* MINITAB. É possível notar que para uma referência de 120 *pixels*, o filamento produzido tem um diâmetro médio de 2,5852mm. É possível notar que foi obtido um desvio padrão de 0,097. Uma vez que o desvio padrão é uma medida de quanto os dados estão dispersos em torno de uma média, e que quanto mais próximo de zero esse medida é melhor o resultado, é possível notar que o filamento possui uma desvio muito pequeno em torno da média.

Através do intervalo de confiança é possível analisar a qualidade por uma outra ótica. Segundo os dados, em 95% das vezes é possível afirmar que a medida do filamento se encontra entre 2,5592mm e 2,6112mm. Ou seja, a variação de diâmetro do filamento será de no máximo $\pm 0,029$mm. Dessa forma é possível afirmar que o objetivo do trabalho que era conseguir uma tolerância do diâmetro de no máximo $\pm 0,05$mm foi atingido.

Entretanto, alguns pontos devem ser considerados. Apesar do sucesso obtido no controle do diâmetro, a referência de 120 *pixels* utilizada neste trabalho não foi adequada ao objetivo que seria de produzir um filamento de 2.85mm de diâmetro. Uma melhor conversão de *pixels*-diâmetro deve então ser realizada. Outro ponto é a grande variância do diâmetro mesmo que de forma ocasional. Grandes variações do diâmetro podem emperrar a impressora. Essas variações devem obrigatoriamente ser corrigidas para garantir que tal situação jamais ocorra durante o seu uso.

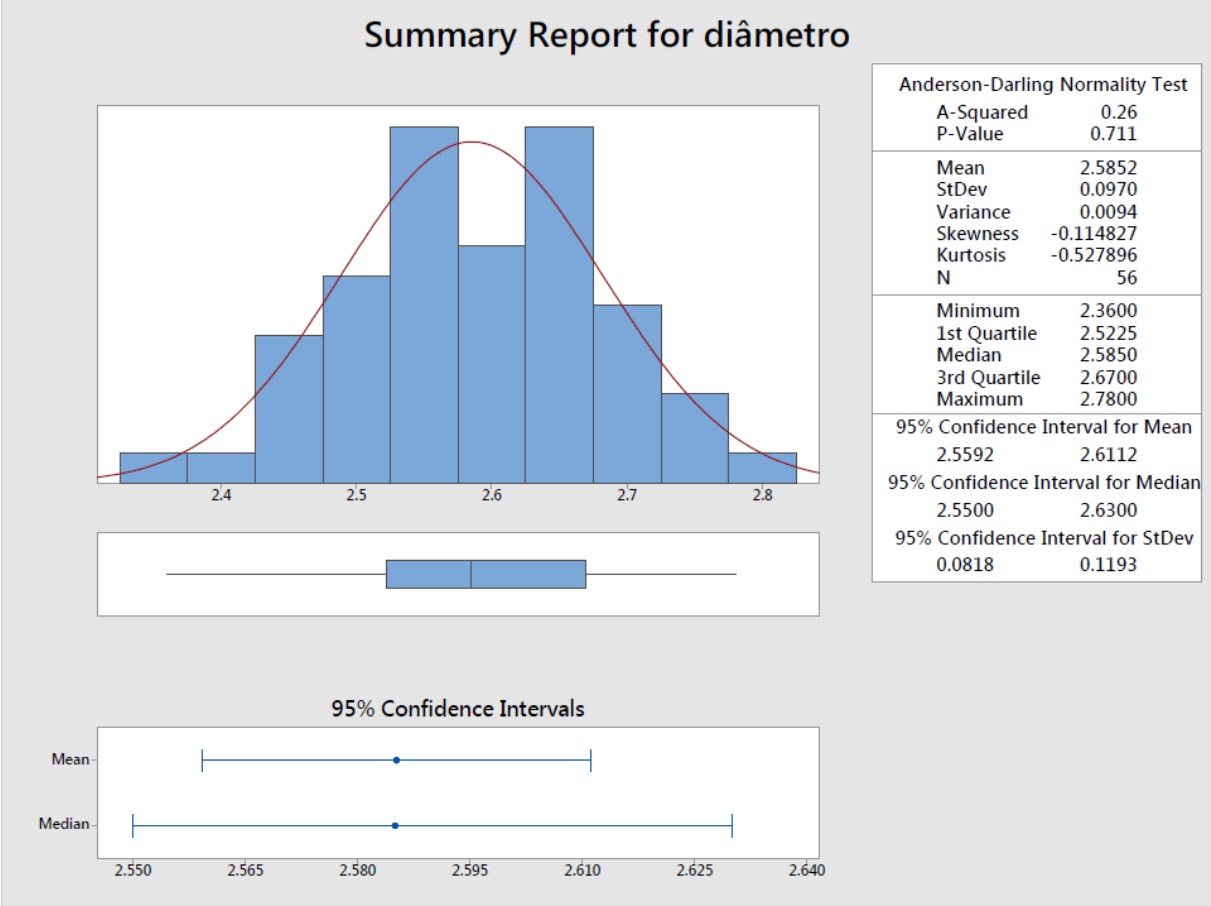

Figura 37: Resultado das medições do filamento produzido.

4.2 Qualidade final

O filamento produzido mostrou ter qualidade aceitável para utilização em impressoras 3D. Foi mostrado que é possível produzir filamentos de boa qualidade utilizando uma máquina extrusora de baixo custo através do emprego da visão computacional como uma alternativa de sensoreamento de baixo custo e malhas de controle simples. A imagem 38 mostra uma peça impressa com o filamento produzido.

É possível notar a ausência de qualquer defeito grave na peça bem como a consistência de suas camadas. A ausência de bolhas na superfície da peça é um indicativo de que a umidade foi de fato mantida em um padrão aceitável de forma que esta não afetou a estrutura ou o acabamento superficial da peça.

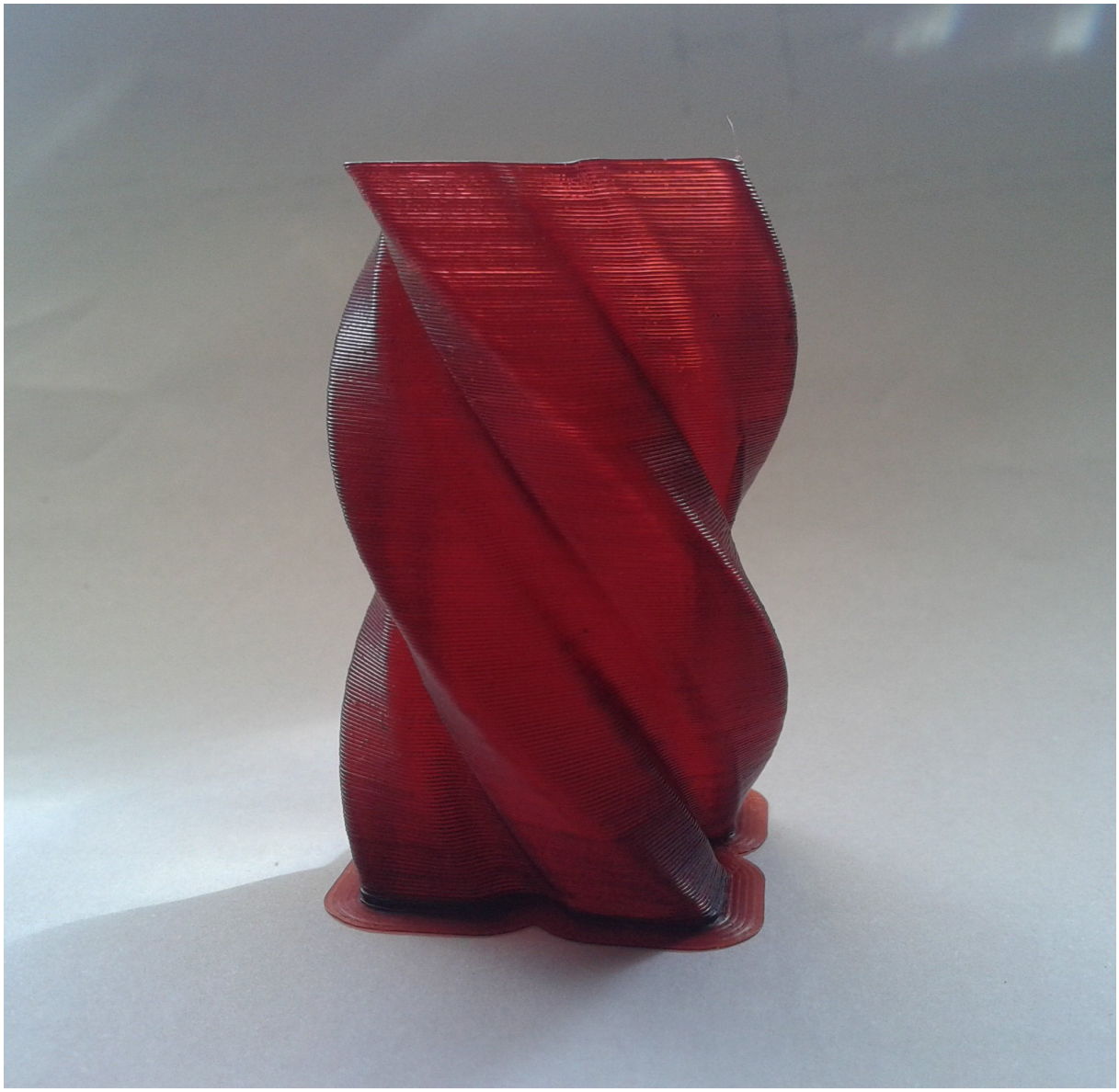

Figura 38: Peça impressa utilizando o filamento produzido.

5 Conclusões e sugestões para trabalhos futuros

É possível concluir que o objetivo do trabalho foi atingido de forma satisfatória. Foi produzido filamento de qualidade industrial com equipamentos e sensoreamento de baixo custo. A visão computacional se mostrou uma ótima alternativa a sistemas de medição de custo mais elevado e aliada a sistemas de controle é de fato muito precisa no controle de processos. Assim, foi comprovada a precisão do filamento de forma a garantir a viabilidade de seu uso em uma impressora 3D. Entretanto, pequenos problemas como uma ocasional grande variação do diâmetro do filamento deve ser corrigida de forma a viabilizar o uso comercial do filamento.

No que tange à sugestões para trabalhos futuros, é interessante que haja um sistema de fácil remoção do plástico restante de operações passadas que acabam ficando alojados dentro da base da extrusora. É importante lembrar da influência da umidade no plástico e caso esse restante de matéria prima fique sujeita a umidade em quanto não é utilizada, a qualidade do filamento produzido pode diminuir. Além disso, outro ponto interessante é o monitoramento da pressão no tubo extrusor. O monitoramento dessa variável é de grande importância para processos de extrusão mas que devido ao tempo e recursos limitado, não foi monitorada ao longo deste trabalho. um sistema de pré-aquecimento integrado à maquina seria também uma melhoria indispensável. Tal sistema possibilitaria um melhor aproveitamento do plástico além de facilitar bastante seu tratamento prévio do plástico para que este possa ser utilizado na extrusora, além de ser possível projetá-lo utilizando materiais de baixo custo.

Referências

AGUIRRE, L. A. *Introdução à Identificação de Sistemas - Técnicas Lineares e nã-Lineares Aplicadas a Sistemas Reais*. [s.n.], 2007. Disponível em: <https://books.google.com.br/books?id=f9IwE7Ph0fYC>. Citado na página 24.

BOTFEEDER. *3D Printer Filament Buyer's guide*. 2015. Disponível em: <https://www.botfeeder.ca/3d-printer-filament-buyers-guide>. Citado na página 17.

BRANDOLT, H. G. *Simulação de escoamento em dutos por caracterização de eventos*. 2002. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/83601/189337.pdf?sequence=1>. Citado 2 vezes nas páginas 8 e 24.

CONAIRGROUP. 2014. Disponível em: <http://www.conairgroup.com/assets/Extrusion/Images/HighRes/MCB-Cooling-Tank.jpg>. Citado 2 vezes nas páginas 9 e 50.

HAROLD JR., G.; WAGNER, J. R. J. *Extrusion: The Definitive Processing Guide and Handbook*. 1. ed. [S.l.]: Willian Andrew, 2005. Citado 3 vezes nas páginas 21, 42 e 49.

LAGES, W. F. Controladores pid. 2010. Citado 2 vezes nas páginas 23 e 24.

LYMAN, H.; MULIER, F. *Lyman / MulierFilament Extruder v5*. 2014. Disponível em: <http://www.thingiverse.com/thing:380987>. Citado 2 vezes nas páginas 26 e 27.

MAKERSLIDE.COM. 2015. Disponível em: <http://store.makerslide.com/images/qd_fix_right.jpg>. Citado 2 vezes nas páginas 8 e 19.

MARENGONI, M.; STRINGHINI, D. *Tutorial: Introdução à Visão Computacional usando OpenCV*. 2009. Disponível em: <http://seer.ufrgs.br/rita/article/viewFile/rita_v16_n1_p125/7289>. Citado na página 21.

MUSOR. 2014. Disponível em: <http://www.aliexpress.com/snapshot/6450420270.html?orderId=65453277021461>. Citado 2 vezes nas páginas 8 e 30.

OPENCV. *Basic Thresholding Operations*. 2015. Disponível em: <http://docs.opencv.org/doc/tutorials/imgproc/threshold/threshold.html>. Citado 3 vezes nas páginas 8, 22 e 23.

OPENCV. *Smoothing Images*. 2015. Disponível em: <http://docs.opencv.org/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html>. Citado na página 37.

POLOLU. *DRV8825 Stepper Motor Driver Carrier, High Current*. 2015. Disponível em: <https://www.pololu.com/product/2133>. Citado 2 vezes nas páginas 8 e 29.

REPRAP. *File:Arduinomega1-4connectors.png*. 2014. Disponível em: <http://www.reprap.org/wiki/File:Arduinomega1-4connectors.png>. Citado 2 vezes nas páginas 9 e 73.

REPRAP. *File:RAMPS1.4schematic.png*. 2014. Disponível em: <http://www.reprap.org/wiki/File:RAMPS1.4schematic.png>. Citado 4 vezes nas páginas 8, 9, 30 e 74.

TEXCO. *Starex BF-0677HF datasheet*. 2015. Disponível em: <http://catalog.ides.com/Datasheet.aspx?I=54824&U=1&CULTURE=en-US&E=148062>. Citado 2 vezes nas páginas 47 e 49.

Apêndices

APÊNDICE A – Código do arduino

```
#include <TimerFive.h>
#include <TimerThree.h>
#include <TimerFour.h>
#include <TimerOne.h>
#include <stdlib.h>
#include "funcoes.h"

/*
* ====================== FILATECH ======================
 * Descricao: Programa de controle e acionamento da extrusora
 * Autor: Guilherme Menezes Costa
 * Data: 19/04/2015
 * ================================
 * ============ TIMERS ============
 * ================================
 * Timer 1-> Loop de temperatura
 * Timer 3-> Extrusora
 * Timer 4-> Puller
 * Timer 5-> Atualiza temperatura atual
 * ================================
 * ===== CODIGO DE ACIONAMENTOS =====
 * ================================
 * G20 - Liga o aquecedor
 * G21 - Desliga o aquecedor
 * G30 - Liga o motor da extrusora
 * G31 - Desliga o motor da extrusora
 * G32 - Seta a velocidade do motor da extrusora
 * G40 - Liga o motor do puller
 * G41 - Desliga o motor do puller
 * G42 - Seta a velocidade do motor do puller
 * G50 - Liga os ventiladores
 * G51 - Desliga os ventiladores
 * G70 - Seta o diametro desejado do filamento
 * G71 - Seta o diametro atual do filamento
 * G72 - Liga o controle de dimetro do filamento
```

```
 * G73 - Desliga o controle de dimetro do filamento
 * ========================================================
 * ========================================================
 */

// ================= DEFINICOES E INICIALIZACOES ======================
#define MAXARRAY 20
#define PINHEATER 9
#define PINSENSORTEMPERATURA 13
#define PINCOOLER 8
#define PINEXTRUSORAENABLE 24
#define PINEXTRUSORASTEP 26
#define PINEXTRUSORADIR 28
#define PINPULLERENABLE 38
#define PINPULLERSTEP A0
#define PINPULLERDIR A1
#define STEPRESPULLER 32
#define STEPSPORREVPULLER 1200 // Reducao de 15/90 = 1/6 (200*6)
#define STEPRESEXTRUSORA 16
#define STEPSPORREVEXTRUSORA 3000// Reducao de 1/15 (200*15)
#define TEMPODEAMOSTRAGEM 1 // Em segundos
#define TEMPOPWM 200000
#define TEMPODEAMOSTRAGEMDIAMETRO 1

// Valor do termistor na temperatura nominal
#define TERMISTORNOMINAL 100000
// Temp. nominal descrita no Manual
#define TEMPERATURENOMINAL 25
// Beta do nosso Termistor
#define BCOEFFICIENT 3950
// valor do resistor em série
#define SERIESRESISTOR 4800

float temperaturaAtual;
float temperaturaDesejada;
float temperaturaAmostrada;
float diametroDesejado = 0;
float diametroAtual = 0;
float erroAtual;
```

```
float erroAnterior = 0;
float erroAnteriorDiam = 0;
char inData[MAXARRAY]; // Allocate some space for the string
char inParameter[MAXARRAY]; // Array para parametros
char inChar = -1; // Where to store the character read
byte index = 0; // Index into array; where to store the character
String st = "";
boolean motorExtrusoraLigado = false;
boolean motorPullerLigado = false;
float RPMextrusora = 60;
float RPMpuller = 60;
double t = 0.0000000;
float pwmVal = 0;
float pullerRpmVal = 0;
bool controleFilamento = false;
static float integralDiam = 0;

float kpDiam = -0.07;
float kiDiam = -0.004;
float kdDiam = 0;

// ==================== INICIO SETUP ================================
void setup() {

  Serial.begin(115200);

  pinMode(PINHEATER, OUTPUT);
  pinMode(PINCOOLER, OUTPUT);
  pinMode(PINEXTRUSORAENABLE, OUTPUT);
  pinMode(PINEXTRUSORASTEP, OUTPUT);
  pinMode(PINEXTRUSORADIR, OUTPUT);
  pinMode(PINPULLERENABLE, OUTPUT);
  pinMode(PINPULLERSTEP, OUTPUT);
  pinMode(PINPULLERDIR, OUTPUT);
  pinMode(PINSENSORTEMPERATURA, INPUT);

  digitalWrite(PINHEATER, LOW);
  digitalWrite(PINCOOLER, HIGH);
  digitalWrite(PINEXTRUSORAENABLE, HIGH);
```

```
  digitalWrite(PINPULLERENABLE, HIGH);
  digitalWrite(PINPULLERDIR,HIGH); // Seta a direcao de giro do puller

  Timer4.initialize(10000); //Timer do motor da Extrusora
  Timer5.initialize(1000000);
  Timer5.attachInterrupt(amostraTemperatura);
  Timer1.initialize(1000000); // Timer aquisicao e controle da temperatura
}

// ============== Inicio do programa principal =====================
void loop()
{
  temperaturaAmostrada = getTemperatura();
  while (Serial.available() > 0) // Don't read unless
    // there you know there is data
  {
    if (index < MAXARRAY - 1) // One less than the size of the array
    {
      inChar = Serial.read(); // Read a character
      inData[index] = inChar; // Store it
      index++; // Increment where to write next
      inData[index] = '\0'; // Null terminate the string
    }

  }
  // Atualiza o valor da string de comando
  st = inData;

  // Bloco de comparacao de comandos recebidos pela porta serial
  if (Comp("G20") == true) {
    temperaturaDesejada = getParametro(st);
    Timer1.attachInterrupt(loopOn);
  }
  if (Comp("G21") == true) {
    Serial.write("Desligando o aquecedor\n");
    Timer1.detachInterrupt();
    digitalWrite(PINHEATER, LOW);
  }
  if (Comp("G22") == true) {
```

```
    float temp = getTemperatura();
    String str;
    if (temp < 100) {
      str = "G22,0";
    }
    else {
      str = "G22,";
    }

    char charVal[10];
    dtostrf(temp, 4, 2, charVal);
    for (int i = 0; i < sizeof(charVal); i++)
    {
      str += charVal[i];
    }
  }
  if (Comp("G30") == true) {
    motorExtrusoraLigado = true;
    Serial.write("Ligando o motor da extrusora\n");
    digitalWrite(PINEXTRUSORAENABLE, LOW);
  }
  if (Comp("G31") == true) {
    Serial.write("Desligando o motor da extrusora\n");
    motorExtrusoraLigado = false;
    Timer3.detachInterrupt();
    digitalWrite(PINEXTRUSORAENABLE, HIGH);
  }
  if (Comp("G32") == true) {

    RPMextrusora = getParametro(st);
    t =  60.0 / RPMextrusora;
    Serial.print("valor: ");
    Serial.println(t, 7);
    t = t / STEPSPORREVEXTRUSORA;
    Serial.print("valor: ");
    Serial.println(t, 7);
    t = t / STEPRESEXTRUSORA;
    Serial.print("valor: ");
    Serial.println(t, 7);
```

```
  t = t * 1000000; // Passando para microsegundos
  t = t / 2;
  // É necessario dividir por dois para que o
  // tempo de step completo (UP and DOWN ) seja o especificado

  Serial.print("valor de t: ");
  Serial.println(t, 7);
  Timer3.initialize(1000000); // Inicializa o timer da extrusora
  Timer3.attachInterrupt(acionaMotExtrusora);
  Timer3.setPeriod(t);

  motorExtrusoraLigado = true;
  Serial.write("Setando a velocidade do motor da extrusora\n");
  Serial.println(RPMextrusora);
}
if (Comp("G40") == true) {
  motorPullerLigado = true;
  Serial.write("Ligando o motor do puller\n");
  digitalWrite(PINPULLERENABLE, LOW);
}
if (Comp("G41") == true) {
  Serial.write("Desligando o motor do puller\n");
  motorPullerLigado = false;
  Timer4.detachInterrupt();
  digitalWrite(PINPULLERENABLE, HIGH);
}
if (Comp("G42") == true) {

  RPMpuller = getParametro(st);
  t =  60.0 / RPMpuller;
  Serial.print("valor: ");
  Serial.println(t, 7);
  t = t / STEPSPORREVPULLER;
  Serial.print("valor: ");
  Serial.println(t, 7);
  t = t / STEPRESPULLER;
  Serial.print("valor: ");
  Serial.println(t, 7);
  t = t * 1000000; // Passando para microsegundos
```

```
  t = t / 2;

  // E necessario dividir por dois para que o
  // tempo de step completo (UP and DOWN ) seja o especificado

  Serial.print("valor de t: ");
  Serial.println(t, 7);
  Timer4.initialize(1000000); // Inicializa o timer do puller
  Timer4.attachInterrupt(acionaMotPuller);
  Timer4.setPeriod(t);

  motorPullerLigado = true;
  Serial.write("Setando a velocidade do motor da extrusora\n");
  Serial.println(RPMpuller);
}
if (Comp("G50") == true) {
  Serial.write("Ligando o ventilador\n");
  digitalWrite(PINCOOLER, HIGH);
}

if (Comp("G51") == true) {
  Serial.write("Desligando o ventilador\n");
  digitalWrite(PINCOOLER, LOW);
}

if (Comp("G70") == true) {
  diametroDesejado = getParametro(st);
}

if (Comp("G71") == true) {
  diametroAtual = getParametro(st);
  Serial.print(diametroAtual);
  Serial.print("+");
}

if (Comp("G72") == true) {
  Serial.write("Ligando o controle do filamento\n");
  controleFilamento = true;
  motorPullerLigado = true;
```

```
    digitalWrite(PINPULLERENABLE, LOW);
  }

  if (Comp("G73") == true) {
    Serial.write("Desligando o controle do filamento\n");
    controleFilamento = false;
    motorPullerLigado = false;
    integralDiam = 0;
    digitalWrite(PINPULLERENABLE, HIGH);
  }

  if (Comp("G77") == true) {
            kpDiam = ((float)-getParametro(st))/1000;
    Serial.print("KpVal: ");
    Serial.print(-kpDiam);
    Serial.print("-");
  }

  if (Comp("G78") == true) {
    kiDiam = ((float)-getParametro(st))/1000;
    Serial.print("KiVal: ");
    Serial.print(-kiDiam);
    Serial.print("-");
  }

  if (Comp("G79") == true) {
    kdDiam = ((float)-getParametro(st))/1000;
    Serial.print("KdVal: ");
    Serial.print(-kdDiam);
    Serial.print("-");
  }
  // Fim do bloco de comandos
  // Limpa o array que recebe a porta serial para
  // que possa ser escrito novamente
  limpaArray(inData);

  // Delay para estabilidade
  delay(10);
}
```

```
// ============== FIM DO PROGRAMA PRINCIPAL =============

// ================================================
// ============== BLOCO DE FUNCOES ================
// ================================================

// ======= FUNCAO QUE IMPLEMENTA O CONTROLE DE TEMPERATURA ============
void controlaTemperatura() {

  float kp = 2.4;
  float ki = 0.008;
  float kd = 0;
  float u = 0;
  static float integral = 0;

  float erroAtual = temperaturaDesejada - temperaturaAtual;
  float derivada = (erroAnterior - erroAtual) / TEMPODEAMOSTRAGEM;

    u = kp * erroAtual + kd * derivada + ki * (integral + erroAtual)
     * TEMPODEAMOSTRAGEM;
    if (u > 100) {
      u = 100;
    }
    else if (u < 0) {
      u = 0;
    }
    else {
      integral = integral + erroAtual;
    }

  erroAnterior = erroAtual;
  pwmVal = (int)u;
}

// ===== FUNCAO QUE IMPLEMENTA O CONTROLE DE DIAMETRO DO FILAMENTO =====
void controlaDiametro() {

  float u = 4;
```

```
  float erroAtual = diametroDesejado - diametroAtual;
  float derivada = (erroAnteriorDiam - erroAtual)
   / TEMPODEAMOSTRAGEMDIAMETRO;

    u = kpDiam * erroAtual + kdDiam * derivada + kiDiam
    * (integralDiam + erroAtual) * TEMPODEAMOSTRAGEMDIAMETRO;

    Serial.print("uAnt: ");
    Serial.print(u);
    if (u > 15) {
      u = 15;
    }
    else if (u < 1) {
      u = 1;
    }
    integralDiam = integralDiam + erroAtual*TEMPODEAMOSTRAGEMDIAMETRO;

  Serial.print("u: ");
  Serial.println(u);
  erroAnteriorDiam = erroAtual;
  RPMpuller = (float)u;
  atualizaRpmPuller();
}

void atualizaRpmPuller(){
    t =  60.0 / RPMpuller;
    t = t / STEPSPORREVPULLER;
    t = t / STEPRESPULLER;
    t = t * 1000000; // Passando para microsegundos
    t = t / 2;

// E necessario dividir por dois para que
// o tempo de step completo (UP and DOWN )
// seja o especificado

    if (!motorPullerLigado){
      Timer4.initialize(1000000); // Inicializa o timer do puller
      Timer4.attachInterrupt(acionaMotPuller);
      Timer4.setPeriod(t);
```

```
      motorPullerLigado = true;
    }else{
      Timer4.setPeriod(t);
    }
}

// ================== FUNCOES DE LOOP DO PWM ======================
void loopOn() {
  if (pwmVal > 5.08) {
    digitalWrite(PINHEATER, HIGH);
    float aux = pwmVal;
    aux = aux * TEMPOPWM / 100;
    long int resultado = aux;

    Timer1.setPeriod(resultado); //Setando periodo p/ prox interrupcao
  }
  else {
    Timer1.setPeriod(10010); // Tempo minimo de acionamento do rele SS
  }
  Timer1.detachInterrupt();
  Timer1.attachInterrupt(loopOff);
}

void loopOff() {
  if (pwmVal < 94.91) {
    digitalWrite(PINHEATER, LOW);
    float aux = 100 - pwmVal;
    aux = aux * TEMPOPWM / 100;
    long int resultado = aux;

    Timer1.setPeriod(resultado); //Setando periodo p/ interrupcao anterior
  }
  else {
    Timer1.setPeriod(10010); // Tempo minimo de acionamento do rele SS
  }
  Timer1.detachInterrupt();
  Timer1.attachInterrupt(loopOn);
}
// ==================== FIM DO BLOCO DE PWM =======================
```

```
void acionaMotPuller() {
  if (motorPullerLigado == true) {
    digitalWrite(PINPULLERSTEP, digitalRead(PINPULLERSTEP) ^ 1);
  }
}

void acionaMotExtrusora() {
  if (motorExtrusoraLigado == true) {
    digitalWrite(PINEXTRUSORASTEP, digitalRead(PINEXTRUSORASTEP) ^ 1);
  }
}

// Funcao que compara uma string a outra e retorna verdadeiro se for
// Esta funcao compara apenas os 3 primeiros caracteres
// Comparando assim apenas o codigo principal recebido
boolean Comp(char* This) {

  if (st.substring(0, 3).equals(This)  == true) {
    for (int i = 0; i < MAXARRAY - 1; i++) {
      inData[i] = 0;
    }
    index = 0;
    return (true);
  }
  else {
    return (false);
  }
}

// Pega o parametro que acompanha o codigo
// Ex: G32 030 -> 030
int getParametro(String string) {

  String subString;
  int arrayParametro[3];
  int valParametro = 0;
  subString = string.substring(4, 7);
```

```
  for (int i = 0; i < 3; i++) {
    arrayParametro[i] = int(subString.charAt(i)) - 48; // ASCII -> dec.
  }
  valParametro =  arrayParametro[0] * 100 + arrayParametro[1]
   * 10 + arrayParametro[2];

  return valParametro;
}

// Funcao que pega a temperatura do sensor
float getTemperatura() {

  float media = 0;
  int samples = 10;
  float temperatura;

  for (int i = 0; i < samples; i++) {
    media += analogRead(PINSENSORTEMPERATURA);
  }
  media /= samples;
  media = 1023 / media - 1;
  media = SERIESRESISTOR / media;

  //Faz o cálculo pela fórmula do Fator Beta
  temperatura = media / TERMISTORNOMINAL;     // (R/Ro)
  temperatura = log(temperatura); // ln(R/Ro)
  temperatura /= BCOEFFICIENT;                   // 1/B * ln(R/Ro)
  temperatura += 1.0 / (TEMPERATURENOMINAL + 273.15); // + (1/To)
  temperatura = 1.0 / temperatura;                 // Inverte o valor
  temperatura -= 273.15;                         // Converte para Celsius
  return temperatura;
}

void amostraTemperatura() {
  temperaturaAtual = temperaturaAmostrada;
  controlaTemperatura();
    controlaDiametro();

  Serial.print(temperaturaAtual);
```

```
  Serial.print("-");
}

// Funcao que limpa o array que recebe o serial
void limpaArray(char *array) {

  for (int i = 0; i < MAXARRAY - 1; i++) {
    array[i] = 0;
  }
  index = 0;
}

//========== FIM DO BLOCO DE FUNCOES ================
```

Anexos

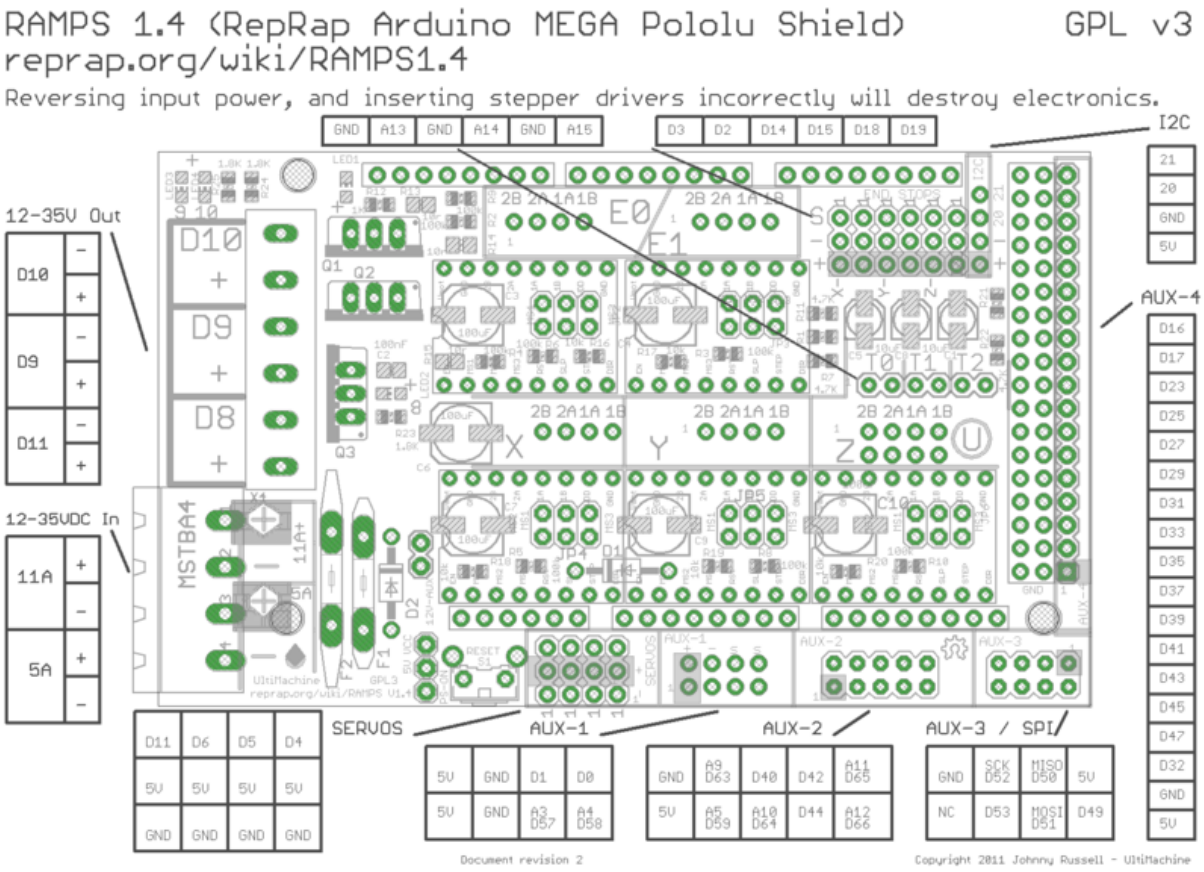

Figura 39: Diagrama da placa Ramps (REPRAP, 2014a)

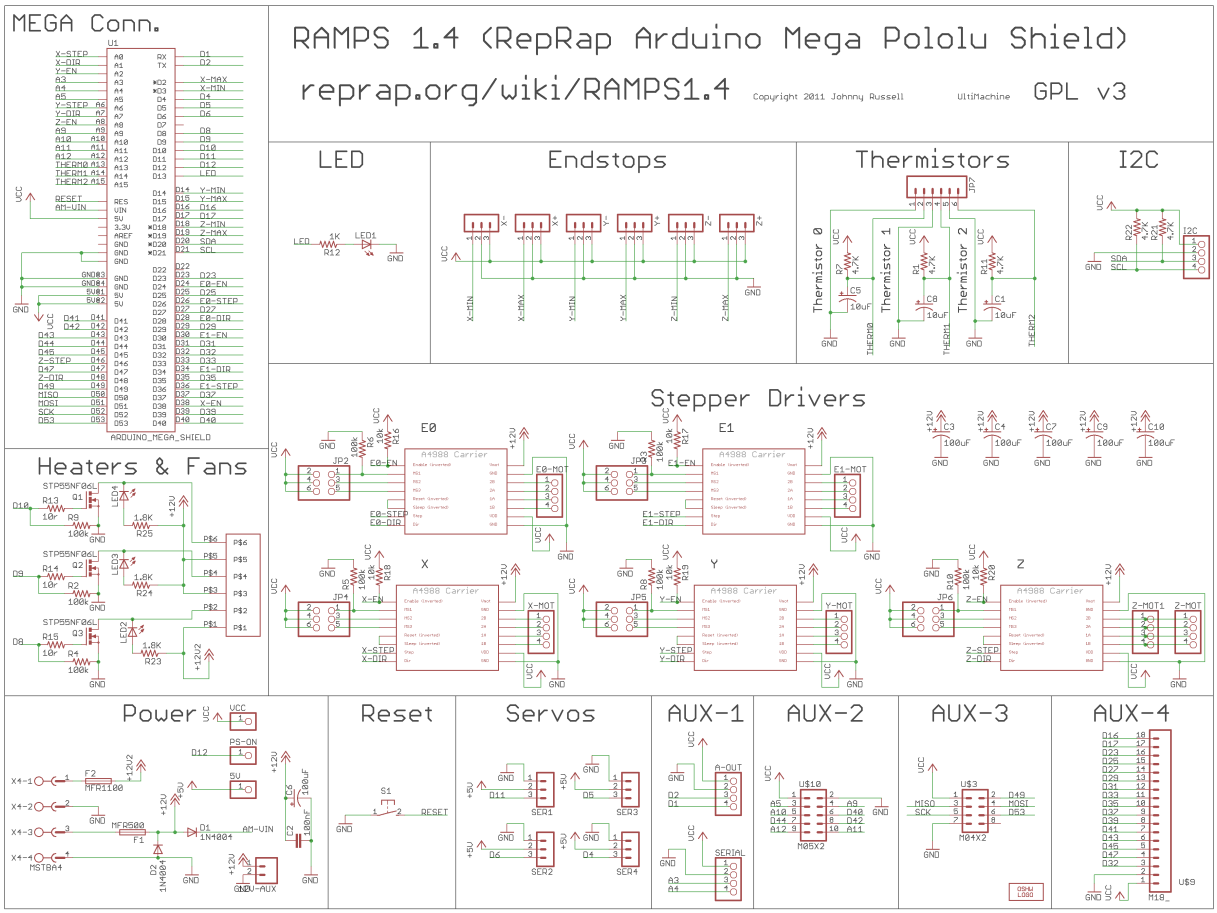

Figura 40: Esquemático da placa Ramps(REPRAP, 2014b)