Centro Federal De Educação Tecnológica de Minas Gerais
Departamento de Engenharia Elétrica

Guilherme Diniz Silva

PROJETO E DESENVOLVIMENTO DE SISTEMA DE CONTROLE DE TEMPERATURA DO PROCESSO DE BRASSAGEM

Belo Horizonte
06/07/2017

Guilherme Diniz Silva

PROJETO E DESENVOLVIMENTO DE SISTEMA DE CONTROLE DE TEMPERATURA DO PROCESSO DE BRASSAGEM

Trabalho de Conclusão apresentado ao Departamento do Curso de Graduação em Engenharia Elétrica do Centro Federal de Educação Tecnológica de Minas Gerais, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador: Patrick Mendes dos Santos
Centro Federal de Educação Tecnológica de Minas Gerais

Belo Horizonte
Departamento de Engenharia Elétrica
2017

“Há várias maneiras de ser entendido, ser claro é uma delas. ”

Bernard Shaw

Agradecimentos

A Deus, por ter me dado saúde e força para superar as dificuldades.

Aos meus pais e minha família, pelo apoio e encorajamento incondicional.

À minha namorada, pelo incentivo constante, pela força e apoio emocional, e pelas broncas também.

Aos meus amigos, pelas colaborações e incentivo a continuar seguindo em frente (mesmo que as vezes eu quisesse desistir).

Ao meu orientador Patrick Mendes dos Santos, pelo suporte e incentivo e pelo seu espírito entusiasmado.

A todos os professores que colaboraram para a elaboração deste trabalho, em especial aos professores Giovani Guimarães Rodrigues, Everthon de Souza Oliveira e Márcio Expedito Guzzo.

Ao CEFET-MG e seu corpo docente, que trabalham continuamente para contribuir para um horizonte superior para mim e centenas de outros alunos.

E a todos que direta ou indiretamente fizeram parte da minha formação, muito obrigado.

Resumo

Brassagem é um processo cervejeiro que consiste em cozinhar grãos de malte tratados em água para obter, através da dissolução, extratos para a preparação da bebida. A maneira como é feito o cozimento afeta o sabor da cerveja, o que retrata a importância do controle minucioso da temperatura ao longo do processo.

O projeto consistiu no projeto e análise de um sistema de controle da temperatura do processo de brassagem, baseado no sistema térmico formado entre a água, o malte e a panela. A Função de Transferência do circuito foi obtida, assim como os parâmetros do sistema e do controlador. A simulação de uma placa de controle foi elaborada para executar a aquisição do sinal de um sensor de temperatura e fazer o tratamento do mesmo por um microcontrolador. Também foi estudado o melhor método de acionamento do atuador da planta. A determinação da instrumentação necessária foi feita através de uma revisão bibliográfica e estudos acerca do sistema térmico específico e das mínimas necessidades de atuação inerentes ao processo.

PALAVRAS-CHAVE: Brassagem, mosto, cerveja, sistema térmico, Arduino, DS18B20, PID, temperatura, controle de sistemas lineares.

Abstract

Mashing is a brewing method consisting on cooking malt grains in water to obtain, through dissolution, extracts for beer preparation. The cooking method affects the beer flavor, wich justify the importance of the meticulous temperature control along the process.

This work consists on a design and analysis of a mashing temperature control, based on the thermal system formed by water, malt and the pot. The Transfer Function of the thermal circuit is obtained, as well as the parameters of the system and the controller. A simulation of a control circuit was elaborated to execute the temperature acquisition and to treat it's signal by a microcontrolador. It was also studied the best plant heating resistor. The additional instrumentation needed was obtained through bibliographical review and studies about the specific thermal system and the minimum needs of performance inherent in the process.

KEYWORDS: Mashing, wort, beer, PID, thermal system, Arduino, DS18B20, PID, temperature, linear systems control.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

°C	Celsius
K	Kelvin
t	Tempo
ΔT	Variação de temperatura
min	Minuto
dQ	Taxa de variação de calor
dt	Variação do tempo
qk	Taxa de transferência de calor por condução (W)
dT/dx	Gradiente de temperatura na direção x (K/m)
k_c	Condutividade térmica (W/mk)
A	Área da superfície (m^2)
q_c	Taxa de transferência de calor por convecção (W)
k_{fluido}	Condutividade térmica (W/mK)
h_c	Coeficiente médio de transferência de calor por convecção sobre uma área A (W/m^2k)
~~h_c~~	H_c médio
C	Capacitância térmica
c	Calor específico (J/kgK)
m	Massa (kg)
R	Resistência térmica (W/K)
L	Comprimento (m)
i	Corrente elétrica (A)
ΔV	Diferença de potencial ou tensão (V)
$\partial E/\partial t$	Taxa de armazenamento de energia (W)
ρ	Resistividade do material (Ω.m)
P	Potência ativa

Ni	Níquel
Cr	Cromo
Fe	Ferro

Lista de Abreviações

RISC	Reduced Instruction Set Computer
PID	Controlador proporcional, integral e derivativo

Capítulo 1

Introdução

1.1. Contextualização

O controle de processos desempenha, cada vez mais, um papel de avanço tecnológico na engenharia. O objetivo do controle e de um processo é torná-lo mais confiável e simplificar sua operação. Nas grandes indústrias a integração da engenharia de controle nos processos já se tornou indispensável. Porém a automação de processos ainda não é muito difundida em pequenas. Apesar disso a automação superou resistências dos investidores devido às inúmeras vantagens que trouxe. Além da capacidade de controlar o processo, as empresas podem ter melhor utilização do tempo e diminuírem os custos de fabricação.

Inserido nesse contexto estão os produtores artesanais de cerveja. O mercado brasileiro de cerveja passa, atualmente, por um processo de aumento na produção artesanal de cerveja, associado ao desejo dos consumidores de adquirir bebidas de qualidade, sem adicionais e cuidadosamente preparadas [1] . Com o aumento da renda consequente da situação econômica favorável em meados de 2007, houve um aumento das viagens para o exterior (que permitiu consumidores conhecerem novos sabores de cervejas), e crescimento da população com acesso à internet, que pode entrar em contato com o mercado cervejeiro em um patamar mundial. Além destes fatores, se somam a popularização dos meios de fabricação de cerveja e a maior facilidade de aquisição da matéria-prima. O aumento de estabelecimentos especializados na venda de materiais básicos para a produção incentivou cervejeiros a aderirem ao mercado no Brasil.

Nesse cenário surgiu então o problema da automação do processo cervejeiro. Este processo é complexo, caracterizado por várias etapas, cada uma delas com características de produção específicas. A fermentação, por exemplo, exige um controle do tempo e da quantidade de enzimas e amido consumidos, que podem ser controlados a partir da quantidade de CO_2 produzido pelas leveduras. Ou mesmo a brassagem que tem as rampas de temperatura que devem ser aplicadas durante um determinado espaço de tempo, proporcionais ao calor, sendo o objeto de estudo desse trabalho. Assim, esses exemplos ilustram como a automação

do processo faz com que as exigências de cada receita sejam atendidas, economiza tempo, reduz perdas de lotes de cerveja e permite mais flexibilidade ao produtor de cerveja, pois requer menos mão-de-obra.

Por isso o mercado de automação da fabricação de cerveja artesanal se mostra um mercado de oportunidades, e aos poucos vem crescendo, para atender as demandas dos empreendedores. Já existem disponíveis controladores de temperatura voltados à demanda de controle de temperatura da brassagem. Muitos controladores como o CAD-T-326-CSSRBZ-1 da empresa Baed, ou AGEON G101 apresentam, no entanto, a solução de controle ON-OFF, não atendendo o requisito ideal de controle. Outros controladores, com lógica PID, como NOVUS N480D-RP ou o CDP48I22P-101-RL também da Baed, são muito caros para um microempreendedor.

Dessa forma se justifica a execução de um controlador PID que possa atuar nas rampas de brassagem de forma rápida e precisa, simples e barata.

1.2. Objetivos do trabalho

O tema do trabalho vigente é o controle de temperatura de cozimento de malte, processo conhecido como brassagem na fabricação de cerveja. Esse controle tem se tornado popular justamente devido à crescente produção artesanal de cerveja. O controle deve ser preciso por estar relacionado a processos bioquímicos que exigem faixas específicas de temperatura e de variação da mesma.

O objetivo deste trabalho é desenvolver o controle de temperatura do processo de brassagem, e adequar esse sistema às condições exigidas pelo processo. Uma vez projetado o sistema, configurar os sinais de rampas de temperatura, requeridas na fabricação de cerveja. As rampas dependem do tipo de cerveja, e assim, das exigências da receita, sendo definidas como critérios básicos.

Como meta final, está o projeto de uma planta de controle do processo de cozimento da mistura de malte e água e uma posterior simulação de uma placa eletrônica que permita a implementação prática deste controle, estabelecendo uma configuração mínima de controle para o sistema. Isso exigirá um estudo do sistema de aquisição de dados através de sensores térmicos, e posteriores simulações para avaliação da robustez do projeto.

1.3. Metodologia do trabalho

A implementação de um controle da temperatura da brassagem requer, primeiramente, um entendimento completo das etapas iniciais do processo de produção de cerveja, com ênfase na relevância da temperatura ao longo desse início. É necessária também uma definição das técnicas de controle necessárias ao projeto do controlador. Por sim, é preciso fazer um estudo dos elementos componentes da planta de controle, como sensor, atuador e circuitos adicionais. Assim os estudos gerais e uma aprofundada revisão bibliográfica do processo de controle da brassagem são parte do trabalho.

Mirando os objetivos descritos acima as atividades foram divididas em:

1. Revisão bibliográfica do processo de brassagem: como se dá o cozimento; detalhes sobre o aumento de temperatura; temperatura inicial e final; e efeitos do processo no sabor da cerveja;
2. Revisão bibliográfica do controle de temperatura, considerando aspectos importantes da teoria de controle;
3. Estudo e projeto dos principais componentes associados ao controle de temperatura (acionador, microcontrolador, sensor), tomando como direcionadores: facilidade de aquisição e implementação, capacidade de processamento e funções associadas;
4. Modelagem de um sistema de controle adequando esse modelo aos parâmetros essenciais do projeto;
5. Projeto e simulação de uma placa de eletrônica contendo funções mínimas que possibilitem o controle;
6. Adequação da placa às condições específicas do processo de brassagem;
7. Desenvolvimento e programação de um sistema de controle no microcontrolador escolhido.

1.4. Organização do texto

O trabalho está organizado em 6 capítulos, incluindo este capítulo introdutório, com a justificativa ao trabalho.

No Capítulo 2 apresenta-se uma breve descrição dos principais ingredientes e etapas no processo de fabricação de cerveja, essenciais para a contextualização e entendimento do trabalho.

O Capítulo 3 apresenta uma breve revisão bibliográfica dos sistemas de controle e também alguns conceitos sistemas térmicos necessários ao projeto. É feita uma apresentação dos conteúdos necessários para o projeto como técnicas de modelagem matemática, teoria de controle PID e uma descrição do controle realizado por computador.

No Capítulo 4 é abordada a descrição dos principais circuitos e elementos da planta. Descreve-se o ebulidor que age como atuador, o circuito de medição, o circuito de potência responsável por acionar o ebulidor e a interface de controle. São também descritas as ferramentas computacionais necessárias para a modelagem e projeto.

O Capítulo 5 trata da modelagem matemática analítica e experimental do processo de segunda ordem. Apresentam-se os dados coletados com os respectivos modelos obtidos para cada conjunto de dados. São apresentados métodos de implementação do processo de medição de temperatura, as rotinas de captura de temperatura e os resultados das leituras para dois diferentes casos. Além disso, é realizado uma descrição da geração de *setpoint* a partir de uma função de contagem de tempo e as formas de implementação dessa função. Por fim, é realizado projeto de controle proporcional, integral e derivativo. Em seguida é apresentado o método de malha aberta de Ziegler-Nichols para a sintonia do controlador PID, em comparação com um ajuste com auxílio da ferramenta *rltool* do Matlab® e finalmente o controlador no domínio discreto.

No Capítulo 6 são apresentadas as conclusões obtidas neste trabalho e sugestões para trabalhos futuros.

Capítulo 2

Ingredientes e Processos da fabricação de Cerveja

A cerveja é uma bebida alcoólica formada a partir de água, cereais maltados (como cevada ou trigo), lúpulo e leveduras. De acordo com o artigo 36 do Decreto n °6.871, de 4 de junho de 2009, que regulamenta a Lei n ° 8.918 “cerveja é a bebida obtida pela fermentação alcoólica do mosto cervejeiro oriundo do malte de cevada e água potável, por ação da levedura, com adição de lúpulo”.

Preliminarmente é preciso entender do processo de produção de cerveja para que se tenha uma visão dos estudos necessários para todo o modelamento matemático e projeto de um controlador. A revisão sobre a brassagem determina os critérios básicos de projeto a serem seguidos, a fim de atingir o objetivo.

2.1. PRINCIPAIS COMPONENTES

2.1.1. Água

A água é um importante componente da fabricação de cerveja, pois consiste em 90% do produto final dela [2]. Sendo assim, a água para fabricação da bebida deve ser potável, sem impurezas ou microrganismos, e não deve possuir aromas ou sabores característicos. As duas principais dimensões características a serem controladas, para um produto final de qualidade, são a dureza da água e seu pH.

2.1.2. Malte

Malte é o grão de cereal no início da germinação, interrompida pelo processo chamado de “maltagem”. Na maltagem os grãos de um determinado cereal (cevada, trigo, milho ou arroz) são imersos em água para que se inicie sua germinação. Quando os grãos estão prestes a desabrochar possuem grandes quantidades de enzimas, proteínas e amido que serão processados pela levedura na etapa da fermentação. Assim, nesta fase são aquecidos em

fornos, de forma controlada, interrompendo a ação das enzimas e o processo de germinação [3], mantendo esses substratos.

2.1.3. Lúpulo

Lúpulo é uma planta, originária de climas frios. O lúpulo é um tipo de conservante natural, sendo esta a razão de ser utilizado nas cervejas ao longo dos séculos. Além disso, o lúpulo possui substâncias responsáveis pelo aroma e sabor amargo característico, servindo como uma espécie de "tempero" que diferencia cada cerveja.

2.1.4. Levedura

Leveduras são fungos unicelulares. Seu nome vem da palavra latina "*levare*" que significa crescer, pois o processo de fermentação de pães provocado pela levedura libera gás carbônico, fazendo as massas de alimentos aumentarem em volume. No caso da cerveja estes fungos consomem o açúcar do mosto, e como consequências de reações, liberam, além do gás carbônico, álcool.

2.1.5. Mosto

Mosto não é uma matéria prima em si, mas um substrato do cozimento do malte. É o líquido ligeiramente viscoso, com cor e consistências parecidas com mel, encaminhado para a fermentação. É obtido a partir do cozimento do malte no processo de brassagem.

2.2. PRINCIPAIS ETAPAS NA PRODUÇÃO DE CERVEJA

2.2.1. Moagem

Consiste em moer os grãos de malte de cevada, para expor a parte interna do grão à ação das enzimas. Nessa etapa utilizam-se moedores culinários pois se deve ter o cuidado de atender o critério de moagem.

2.2.2. Mosturação ou brassagem

É o cozimento do malte para a obtenção do mosto. Esse processo pode ser feito de diversas formas e possui patamares de temperatura de cozimento diferentes, dependendo do tipo de receita a ser produzida. Como objeto de nosso estudo, o processo será detalhado posteriormente ainda neste Capítulo.

2.2.3. Drenagem

Consiste na separação do mosto, das sobras de casca e bagaço dos grãos de malte. A drenagem é feita através de recirculação e lavagem. Na etapa de recirculação a mistura de mosto e bagaço é depositada em uma panela de fundo falso (como uma peneira), onde o liquido escoa pelo fundo e a parte solida fica contida acima do fundo [2]. Depois, na etapa de lavagem, o bagaço é jogado em uma panela com água para extrair o resto de possíveis açúcares ainda contidos no bagaço.

2.2.4. Fervura

Consiste basicamente em ferver o mosto para sua esterilização e caramelização de alguns açúcares presentes. Além disso o lúpulo deve ser adicionado nesta etapa (no início ou no final, dependendo do tipo de lúpulo a ser adicionado).

2.2.5. Resfriamento

Após a fervura, o mosto deve ser resfriado rapidamente para atingir a temperatura ideal de ação das leveduras, para impedir contaminações posteriores, e para impulsionar a decantação de algumas substâncias restantes.

2.2.6. Aeração

Fase em que se adiciona oxigênio ao mosto. O oxigênio é usado pela levedura na etapa de fermentação. Normalmente é feita simplesmente derramando lentamente o mosto em um tanque [4].

2.2.7. Fermentação

Etapa onde ocorre a transformação química de carboidratos em dióxido de carbono e de álcoois ou ácidos orgânicos por meio da ação das leveduras. Esta etapa também é responsável por fornecer o aroma característico à cerveja, e eliminar compostos inapropriados à mistura.

2.2.8. Maturação

Maturação é considerada como o término do processo de fermentação. Após a fermentação a cerveja é armazenada em temperaturas próximas a 0°C, na qual o processo de fermentação ainda ocorre. Essa fermentação secundária promove uma redução na concentração de compostos indesejáveis à cerveja. Como consequência da redução desses compostos, a cerveja adquiriu seu aroma específico.

2.2.9. Envase

Envase é o armazenamento, ou "engarrafamento", da cerveja ao fim de todo o processo de produção. Essa etapa também é associada à qualidade da cerveja, pois durante a transferência desta para as garrafas pode ocorrer a oxidação do líquido, adquirindo um aroma indesejado.

2.3. BRASSAGEM – DETALHAMENTO

Brassagem ou mosturação é a etapa do processo cervejeiro onde ocorre o cozimento do malte imerso em água quente. Esta etapa varia de acordo com o tipo de receita de cerveja, pois cada receita exige uma configuração de patamares diferentes. A brassagem tem como objetivo converter o amido do malte em açúcar e algumas proteínas em aminoácidos (que serão consumidos pelas leveduras na fermentação). Para proporcionar essas quebras moleculares é necessário ativar enzimas que foram inibidas na maltagem, para que sua ação transforme o amido em moléculas de açúcares menores, e quebre moléculas de proteínas em aminoácidos, substâncias consumíveis pelas leveduras. Essa ativação é obtida aumentando sua temperatura até o patamar correspondente de atuação de cada tipo específico de enzima. A quebra de proteínas em aminoácidos é particularmente importante no processo de produção de cerveja. As proteínas de alto peso molecular contribuem para a geração de espuma, mas deixam a produção de cerveja instável, não se diluem na cerveja, não contribuindo para o

corpo ou sabor da mesma e ainda exigindo um processo mais cuidadoso de filtração e podem deixar a cerveja turva. As de baixo peso molecular ou mesmo os aminoácidos são essenciais na etapa de fermentação, mas não contribuem com a espuma.

A etapa de brassagem é executada após a moagem dos grãos do cereal componente da cerveja. O processo se inicia colocando o malte em uma panela com água, para que os grãos se hidratem criando um ambiente propício de ação das enzimas. Porém o malte possui, além do amido, outros açúcares e proteínas diferentes, exigindo ação catalizadora de enzimas também diferentes, cada uma com sua temperatura ideal de trabalho. Por isso o processo de brassagem pode ser executado em diversas temperaturas. A Figura 2.1 exibe um gráfico com as diferentes faixas de temperatura, em graus Celsius, de ação das enzimas contidas do malte.

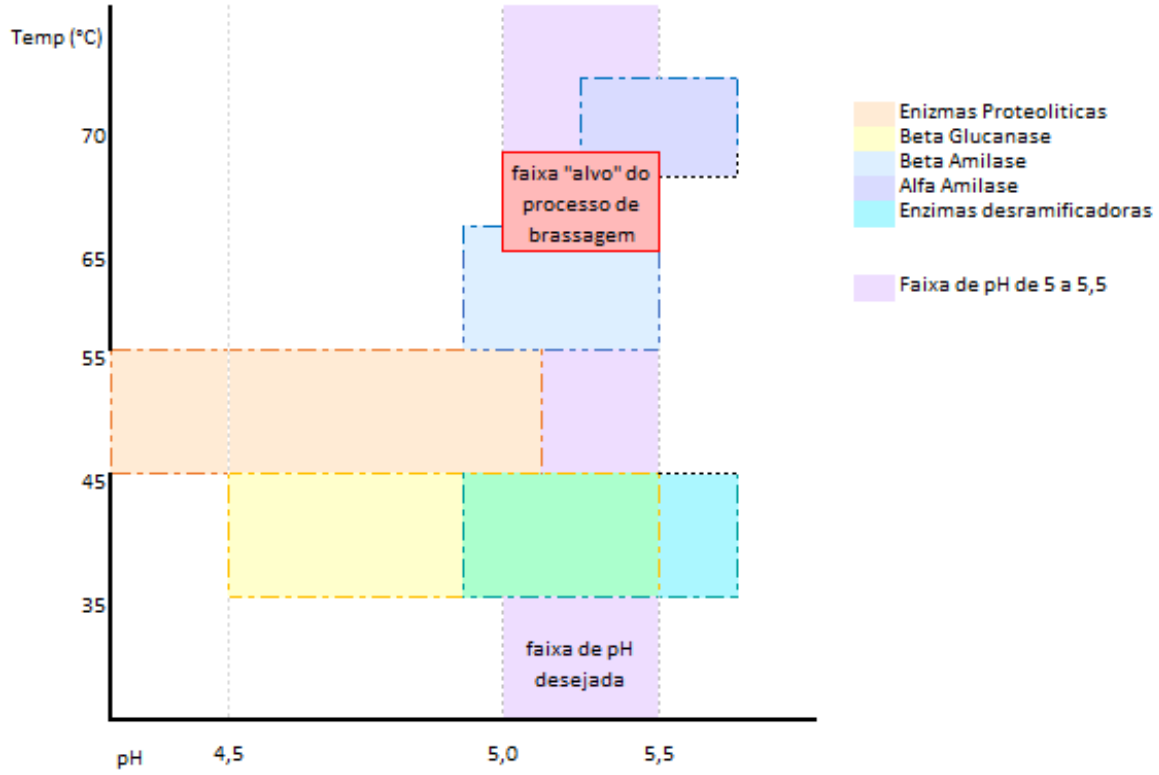


Figura 2.1 – Faixas de ação enzimática na brassagem. Fonte: [5]

Existem dois tipos principais de brassagem: simples e a escalonada.

A brassagem simples consiste na deposição dos grãos em água quente visando obter uma única faixa de temperatura da mistura (entre 65,5°C e 70°C) [5]. Naturalmente a temperatura da água deve ser maior que a temperatura final, prevendo sua queda, devido à adição do malte.

A escalonada é a variação gradual na temperatura da mistura ao longo do processo, para favorecer a ação de enzimas com faixas de atuação diferentes. Para cada temperatura, deixa-se atuar um tempo que depende do objetivo final no sabor da cerveja.

2.3.1. Infusão

A infusão consiste na adição de malte à água já fervida, para que possa se atingir a temperatura visada. Geralmente uma porção de água proporcional à de malte moído deve estar entre 5,5C e 8,5°C acima da temperatura que se deseja fazer a brassagem [5].

Para que o fluxo de calor da água não se dissipe, é recomendado que o processo seja executado dentro de um recipiente isolante térmico.

2.3.2. Aquecimento do recipiente

O método consiste em depositar o malte moído junto à água em uma panela, e executar a brassagem através do simples aquecimento do recipiente, com uma fonte externa de calor. Esse método, como vantagem, permite a execução de várias rampas de temperatura com maior exatidão (o método da decocção também permite a execução de várias rampas de temperatura, mas é menos preciso).

A brassagem por aquecimento do recipiente exige constante agitação para que o conteúdo se aqueça uniformemente, e não haja queima do malte.

2.3.3. Decocção

Outra forma de se realizar a brassagem é através da decocção. Esse processo é bastante tradicional, sendo muito executado antes da invenção dos termômetros, pois permite um aumento da temperatura do mosto de maneira controlada. É feito retirando parte da mistura de malte com água, e o aquecendo através de um dos métodos anteriores. Depois o mosto resultante é colocado novamente junto ao malte moído, elevando sua temperatura [5].

É claramente um processo mais complexo, mas tem a vantagem de ferver parte do malte às temperaturas acima da temperatura de trabalho da brassagem. Isso permite que categorias diferentes de enzimas sejam ativadas, processando moléculas de açúcar diferentes, além de permitir uma coagulação das proteínas dando à cerveja um aspecto mais limpo (é o caso de alguns tipos de cervejas Pilsen que passam por uma decocção tripla, diferente das cervejas de

trigo comumente tendo uma decocção simples ou outro tipo de processo de brassagem. Por isso a cerveja de trigo tem um aspecto mais pardo).

2.3.4. Projeto

O projeto tem como foco a receita de uma "Blonde Belge". O objetivo específico do trabalho é executar o controle da temperatura de cozimento do malte através das rampas de temperatura, em um processo de brassagem por aquecimento do recipiente, para atender a receita de produção da cerveja. A receita prevê, como critérios, que será usado 4,90kg de malte em grãos, somente; típica água potável filtrada característica da rede de abastecimento da Belo Horizonte [6]; e temperatura ambiente de 25°C (e consequentemente 25°C de temperatura inicial dos grãos e da água no início do processo). Por fim, tomando como base as referências sobre produção de cerveja, e boas práticas já consolidadas no meio, assume-se para as rampas uma taxa de elevação de temperatura de 1°C por minuto.

A primeira etapa do processo da brassagem será o aquecimento da mistura de malte e água, de maneira linear, até que a mistura atinja 50°C. Considerando a taxa de 1°C/min temos:

$$t = \Delta T \times \left(\frac{dT}{dt}\right) \tag{2.1}$$

$$t = (50℃ - 25℃) \times \frac{min}{1℃} = 25min$$

Onde:

t	=	Tempo (min)
ΔT	=	Variação de Temperatura (°C)

A segunda etapa consiste na conservação da temperatura do sistema em 50,0°C durante 30 minutos, para que haja um descanso proteico do malte. De acordo com a Figura 2.1, o aquecimento à 50°C ativa as enzimas Proteolíticas, havendo quebras de grandes moléculas de proteínas em proteínas menores ou mesmo aminoácidos.

A terceira etapa, na segunda rampa, elevar a temperatura do sistema de 50°C para 65°C, mantendo a taxa de elevação de temperatura do critério inicial de projeto de 1°C/min. A quarta etapa consiste novamente em manter a temperatura estável em 65°C durante 50 minutos, para favorecer ação das enzimas beta-amilase (57°-66°). Na quinta etapa uma nova rampa é aplicada, elevando o sistema de 65°C para 68°C. A sexta etapa é o controle da

temperatura em 68°C durante 30 minutos, para favorecer a ação das alfa-amilase (68°-73°). As amilases são enzimas que fragmentam polissacarídeos como o amido ou o glicogénio (grandes cadeias moleculares) em moléculas de maltose e dextrinas (pequenas cadeias moleculares).

A sétima etapa é a elevação de temperatura de 68°C para 75°C. O cozimento em temperaturas de 60°C e 70°C age sobre os açúcares fermentáveis em cadeias maiores. A variação dos tempos nos descansos de 60° e 70° C permite ajustar os perfis de açúcares fermentáveis, variando entre uma cerveja mais doce e encorpada ou uma mais seca e suave.

Por fim, na oitava etapa, deve-se manter a temperatura em 75°C por 10 minutos para a saturação do efeito das enzimas e término do processo. O processo é ilustrado graficamente na Figura 2.2:

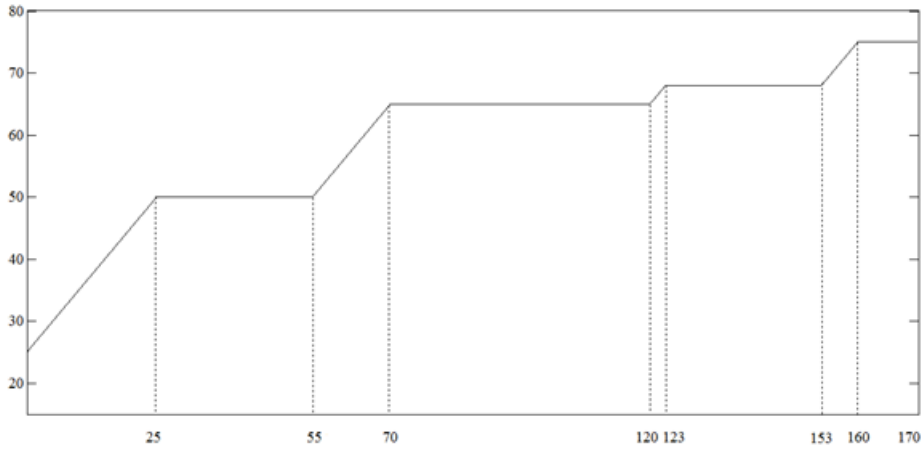


Figura 2.2 – Rampas Padrões Temperatura – Blonde Belge.

Após o estudo aprofundado da brassagem e da análise das enzimas ativadas em cada faixa de temperatura, pode-se perceber a necessidade de controle do processo uma vez que as proteínas e os tipos diferentes de amido requerem temperaturas diferentes, de maneira exata e controlada.

Capítulo 3

Termodinâmica e Sistemas de Controle

O objetivo do Capítulo é estudar os princípios básicos de termodinâmica e as técnicas de modelagem de Função de Transferência necessários para obter um modelo matemático adequado do sistema em estudo. A partir do entendimento do sistema térmico, revisar as técnicas de controle necessárias para o comando da planta.

3.1. INTRODUÇÃO DE SISTEMAS TÉRMICOS

Em um sistema onde haja variação gradativa de temperatura haverá transferência de energia sob a forma de calor (ou fluxo de calor) [7].

A transferência de calor de um sistema a outro acontece de três formas: condução, convecção e radiação. No primeiro caso, quando uma fonte de calor é colocada em contato com um condutor de calor (seja sólido, fluido ou gasoso) a temperatura das outras extremidades do condutor aumenta, até que as perdas de calor para o ambiente se tornem tais que compensem o ganho de calor. Em nível atômico, as partículas em contato com a fonte térmica têm maior energia cinética. Através do choque "mecânico" de umas com as outras a temperatura é transferida de uma extremidade à outra do corpo. O calor em metais é conduzido mais rapidamente pois os metais tem elétrons livres [7].

Quando uma quantidade *dQ* de calor é transferida através de um corpo em um instante de tempo *dt*, a taxa de transferência de calor é dada por $q_k = dQ/dt$. À essa grandeza é dada o nome de taxa de transferência de calor ou taxa de fluxo de calor q_k. Para a condução de calor através de um meio homogêneo a taxa de transferência de calor por condução é dada pela Equação (3.1)

$$q_k = -k_c A \frac{dT}{dx} \tag{3.1}$$

Onde:

q_k = Taxa de energia transferida por condução (W)
dT/dx = Diferença ou gradiente de temperatura (K/m)
k_c = Constante de condutividade térmica (W/mK)

A = Área da superfície ortogonal ao fluxo de calor (m^2)

O sinal negativo ilustra que o fluxo de calor q_k deve ir da temperatura mais alta para a mais baixa [8]. A condutividade térmica k_c se altera com a temperatura, e é utilizada para definir se um determinado material se comporta como isolante ou condutor térmico.

A transferência de calor por convecção é composta de dois processos simultâneos. O primeiro, chamado de difusão, é o processo de condução, citado anteriormente. O segundo, advecção, se dá pelo movimento macroscópico de parcelas do fluido, provocado pela mudança de densidade ou por algum método forçado (fica naturalmente, implícito que a convecção acontece apenas em fluidos) [8]. Esse tipo de transferência de calor depende da diferença de densidade, da viscosidade e da velocidade do fluido. Na convecção natural, a diferença de densidade em pontos distintos do fluido são consequência da diferença entre a temperatura na superfície de contato com a fonte de calor e um ponto mais distante. Por esse motivo a transferência de calor depende indiretamente da variação de temperatura em pontos diversos do fluido. Existem métodos complexos para esse cálculo, mas para fins de simplificação essas dependências podem ser representadas pelo coeficiente médio $\overline{h_c}$ de transferência de calor por convecção sobre uma área *A.* Sendo assim taxa de transferência de calor por convecção pode ser dada pela Equação (3.2) retirada de [9]:

$$q_c = h_c\, A\, \Delta T \tag{3.2}$$

Onde:

h_c = Coeficiente médio de transferência de calor por convecção sobre uma área A (W/m^2K)

Valores usuais de h_c para a água e ar, que serão utilizados no trabalho, são exibidos na Tabela 3.1:

Tabela 3.1 – Coeficientes de transferência de calor por convecção (Fonte: [9])

Substancia	h_c (W/m^2K)
Água – convecção natural	20 - 1000
Ar	6 – 30

A avaliação do coeficiente de transferência de calor por convecção é de difícil obtenção. Por isso em aplicações de engenharia é utilizado um valor de coeficiente médio $\overline{h_c}$.

3.1.1. Capacitância Térmica

Capacitância (ou capacidade) térmica é a taxa de variação da quantidade de calor absorvido por um corpo *Q*, pela variação de temperatura *dT* do mesmo. Dessa forma, a capacitância térmica *C* é uma propriedade que busca relacionar calor com temperatura, e, pela definição, *C* é dada pela Equação (3.3):

$$C = \frac{Q}{\Delta T} \tag{3.3}$$

Onde:

C	=	Capacitância térmica J/(K)
ΔQ	=	Variação da quantidade de calor (J)
ΔT	=	Variação de temperatura (K)

Supondo que um corpo A submetido a uma fonte de quantidade de calor absorve *dQ*, e tem sua temperatura modificada em dT_1. Um segundo corpo B, que recebe a mesma quantidade de calor *dQ*, pode apresentar outra variação dT_2 de temperatura, considerando dT_1 diferente de dT_2 ($dT_1 \neq dT_2$). Quanto menor a capacitância térmica de um corpo, maior a sua variação de temperatura para uma mesma quantidade de fluxo de calor. Analogamente, quanto maior a capacitância térmica, maior o fluxo de calor, se fixada a variação de temperatura [10].

A capacitância térmica é uma característica do objeto, devido à dependência da massa do mesmo. Por isso, a relação da capacitância pela massa do objeto caracteriza o calor específico, como descrito na Equação (3.4):

$$c = \frac{C}{m} \tag{3.4}$$

Onde:

c	=	Calor específico (J/kgK)
m	=	Massa do objeto (kg)

3.1.2. Resistência Térmica

Resistência térmica é a propriedade dos materiais de contrapor a condução de calor e é definida como a razão entre a variação da diferença de temperatura, e a variação do fluxo de calor. Materiais como metais, de alta capacidade de condução de calor, possuem baixa

resistividade térmica. A equação da resistividade térmica em relação à transferência de calor por condução é apresentada na Equação (3.5):

$$R = \frac{d(\Delta T)}{dq} \tag{3.5}$$

Em que:

R = Resistência térmica (K/W)
ΔT = Variação de temperatura (K)
dq = Taxa de calor (W)

Substituindo a equação (3.1) em (3.5), temos a resistência *R* em função dos coeficientes de condução. Uma vez que os coeficientes são próximos de constantes, para cada tipo de material, de acordo com a Tabela 3.2, temos a Equação (3.6):

$$R_{condução} = \frac{L}{Ak_c} \tag{3.6}$$

Em que:

L = Dimensão do condutor de calor, paralelo ao fluxo de calor (m)

Tabela 3.2 - Valores de Condutividade Térmica e Calor Específico

Fluido	Temperatura T(K)	Condutividade Térmica k (W/mK)	Calor Específico c (J/kgK)
Água	300,00	0,6089	4183,00
	350,00	0,6622	4193,00
	400,00	0,6848	4262,00
Ar	300,00	0,0240	1013,00
	350,00		1019,00

Sólido	Temperatura T(K)	Condutividade Térmica k (W/mK)	Calor Específico c (J/kgK)
Alumínio	300,00	237,0000	910,00
	573,20	268,0000	
Cobre	291,20	384,1000	390,00
	373,20	379,9000	393,56
Aço Inox	373,00	15,9000	500,00
	423,00	21,0000	

A resistividade *R* em função dos coeficientes de convecção é exibida na Equação 3.7:

$$R_{convecção} = \frac{1}{\overline{h_c}A} \tag{3.7}$$

O conceito de resistência térmica é aplicável na área de controle pela equivalência entre grandezas físicas térmicas e elétricas. Em situações onde o fluxo de calor é transmitido através de mais de um meio (condução e convecção, por exemplo), a resistência total é a soma das resistências individuais relativas a cada tipo de transferência de calor. Esse método de resistências deve ser usado assumindo que o meio condutor não armazena energia térmica, absorvendo parte do calor emitido para o ambiente. Caso o condutor, de fato, adquira uma parcela desse calor, deve-se adicionar à resistência, uma capacitância térmica ao modelo.

Nesse modelo análogo de sistemas térmicos e elétricos o fluxo de corrente i equivale ao potencial de tensão ΔV dividido por um valor de resistividade R_e. O fluxo de calor é igual ao potencial de temperatura ΔT dividido pelo valor de resistividade térmica.

3.1.3. Equacionamento de um Sistema Térmico

De acordo com a primeira Lei da Termodinâmica, considerando que o sistema é fechado (não havendo, desta forma, entrada ou saída de massa) e que não realiza trabalho, todo calor fornecido ao sistema deve ser igual ao calor absorvido por este, menos as perdas. Considerando o sistema estudado no trabalho como exemplo, temos o balanço de fluxo de calor do sistema ilustrado na Figura 3.1:

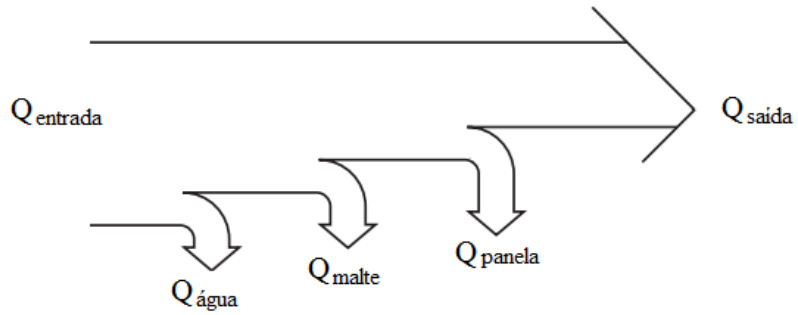


Figura 3.1 – Balanço de fluxo de calor

$$\begin{pmatrix} variação\ na\ energia \\ interna\ do\ corpo \end{pmatrix} = \begin{pmatrix} fluxo\ liquido\ de \\ calor\ do\ corpo \end{pmatrix}$$

Temos, pela Equação (3.8), da Primeira Lei da Termodinâmica:

$$(q_{in} - q_{out}) = \frac{\partial E}{\partial t} \tag{3.8}$$

Onde:

$q_{in} - q_{out}$ = Taxa líquida de transferência de calor dentro do volume de controle (W)

$\partial E/\partial t$ = Taxa de armazenamento de energia dentro do volume de controle (W)

O estudo de sistemas térmicos, o balanço de fluxo de calor da Figura 3.1 e a Equação (3.8) fornecem a teoria necessária para uma dedução analítica da Função de Transferência do sistema, capaz de traduzir matematicamente o comportamento físico do sistema na natureza.

A ordem de um sistema é um parâmetro importante na caracterização deste. No domínio do tempo, a ordem do sistema se refere a derivada de maior ordem da variável controlada na equação diferencial que descreve este sistema. Já no domínio da frequência a ordem do sistema é a ordem da equação característica em *s* (sendo *s* apenas uma simplificação da notação de uma variável complexa em Laplace). Sistemas térmicos podem ser representados por equações de primeira ordem no domínio de Laplace com aceitável grau de fidelidade [11]. Porém, a capacitância e a resistência térmicas são, em geral, distribuídas ao longo do sistema. No entanto, equacionamentos mais simples envolvem simplificações como a consideração de que os parâmetros do sistema são concentrados.

3.2. MÉTODOS DE OBTENÇÃO DA FUNÇÃO DE TRANSFERÊNCIA

3.2.1. Método Analítico

O método analítico sugere uma teoria formal, traduz o sistema em equações e modelos matemáticos, obtém os resultados a partir desses modelos e finalmente, os compara com resultados empíricos. Esse método dedutivo oferece uma base teórica para o desenvolvimento de modelos a partir do próprio conhecimento do funcionamento físico do sistema. O método analítico pode provar modelos matemáticos a partir de conceitos, gerando um projeto mais robusto e confiável.

O método analítico consiste, geralmente, de quatro passos: modelagem, montagem das equações matemáticas, análise e projeto. Os dois primeiros passos estão fortemente relacionados. Se usada uma matemática simples, o modelo escolhido deverá ser correspondentemente simples. Já com uma matemática mais sofisticada, o modelo será mais complexo e realista. A modelagem é o passo mais crítico no projeto analítico. Se um sistema físico for modelado incorretamente, os estudos subsequentes não terão utilidade alguma. Uma vez escolhido um modelo, o restante do projeto analítico é essencialmente um problema matemático.

3.2.2. Método Empírico

O objetivo da modelagem matemática é desenvolver e implementar modelos matemáticos de sistemas reais. A identificação de sistemas estuda técnicas alternativas da modelagem matemática. Uma das características dessas técnicas é que pouco ou nenhum conhecimento prévio do sistema é necessário e, consequentemente, tais métodos são também referidos como modelagem ou identificação caixa preta ou modelagem empírica.

O objetivo dos modelos empíricos é descrever as relações de entrada e saída entre as variáveis de entrada e de saída. Nesse caso, o tipo de modelo, as técnicas usadas e os requisitos necessários são bastante distintos dos correspondentes na modelagem pela natureza do processo pois se considera o sistema como uma "caixa preta", com determinadas entradas e saídas. Nesta situação se realiza um conjunto de experimentos que proporcionam pares de medidas de entradas e saídas durante a evolução do sistema até o estado estacionário, a partir dos quais se determina o modelo do sistema.

O método empírico é usualmente aplicado na modelagem de ordem baixa (principalmente primeira e segunda ordem). E dentro do método empírico tem-se duas abordagens principais: a estimativa por Mínimos Quadrados e os métodos de Curva de Reação. Neste trabalho trataremos do segundo conjunto, sendo esses métodos vantajosos por serem rápidos e de simples execução, baseado em engenharias heurísticas.

Os métodos de Curva de Reação podem ser aplicados da seguinte forma: para um sistema em malha aberta, aplica-se uma perturbação degrau na entrada do processo, a partir do estado de referência estável. Registrar a saída do processo até que um novo estado estacionário seja atingido, e, a partir dos dados coletados, obtém-se a curva do processo. Após esta etapa, é necessário analisar a semelhança da curva da grandeza de saída com um perfil de resposta desejado (por exemplo, de primeira ordem) e, assim, pode-se calcular o ganho e os parâmetros temporais da Função de Transferência a partir de métodos como o de Ziegler–Nichols ou de Smith.

3.3. TÉCNICAS DE CONTROLE RELEVANTES

3.4. Controle PID

A maioria dos sistemas simples tem suas variáveis de entrada assumindo apenas dois valores distintos: zero ou o valor nominal. Pode-se controlar alguns sistemas sem muita

precisão a partir da manipulação destes dois estados. Porém, na maioria das vezes esse tipo de controle, chamado controle *ON/OFF*, gera oscilações indesejadas na grandeza de saída. O controle PID é uma técnica mais adequada de controle, pois permite a seleção de diversos valores das variáveis controladas, e por consequência, um controle mais preciso.

Esse tipo de controle leva em conta a comparação entre um sinal de referência (ou *setpoint*) com outro sinal, medido, que representa o estado da variável de saída em um determinado instante de tempo. Como resultado dessa comparação temos o sinal de erro.

O controle PID é a junção de três ações distintas de controle: a proporcional, a integral e a derivativa. No controle PID, o termo proporcional consiste no valor do erro multiplicado por uma constante de proporcionalidade. O termo integral é o somatório do erro, multiplicado pela variação no tempo. E o termo derivativo é a variação do erro também multiplicado por uma constante de tempo.

No controle PID cada termo atua de uma forma específica sobre o sistema. O controlador proporcional é um amplificador com ganho ajustável K. O aumento do ganho K, diminui o erro de regime permanente. Em geral, o aumento do ganho proporcional torna o sistema mais oscilatório, podendo instabilizá-lo, de forma que o ganho proporcional melhora a resposta em regime permanente mas piora o transitório, sendo bastante limitado. A ação integral do controlador atua na variável de controle zerando o erro permanente, pois aumenta o tipo do sistema. Essa ação é utilizada quando temos resposta em regime insatisfatória, ou seja, o sistema não chega no valor desejado em seu estado permanente. A ação derivativa atua sobre a taxa de variação do erro, sendo utilizada quando temos resposta transitória insatisfatória. O termo derivativo introduz um efeito de antecipação no sistema, fazendo com que o mesmo reaja não somente à magnitude do sinal de erro, mas também à sua tendência para o instante futuro, antecipando uma ação corretiva. Mas a ação derivativa tem a desvantagem de amplificar sinais de ruído o que pode causar saturação nos atuadores do sistema.

O sinal de controle gerado pelo PID é genericamente modelado pela Equação (3.9) no domínio do tempo, ou pela Equação (3.10) no domínio da frequência [12]:

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right) \tag{3.9}$$

Onde:

u(t) = Sinal de entrada
e(t) = Sinal de erro

K_p = Constante proporcional
T_i = Tempo de ação integral
T_d = Tempo de ação derivativa

$$U(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s\right)\mathrm{E(s)} \tag{3.10}$$

Onde:

U(s) = Sinal de entrada em Laplace
E(s) = Sinal de erro em Laplace

Projetar um controlador PID, então consiste em ajustar as constantes que definem o controlador. Método de Sintonia de Ziegler-Nichols de Malha Fechada se utiliza do seguinte algoritmo para essa sintonização:

- Determinar K_C crítico, considerando-se apenas o ganho proporcional, isto é, fazendo T_d igual a zero e T_i igual a infinito.
- Com este método pretende-se obter no máximo 25% de *overshoot*.
- Determinar a frequência de oscilação.
- Determinar o período crítico.

A partir destas determinações utiliza-se a tabela Tabela 3.3 para determinar os parâmetros PID:

Tabela 3.3 – Constantes PID

Tipo de Controlador	K_P	T_I	T_D
P	$0.5K_c$		0
PI	$0.45K_c$	$1/1.2\ P_c$	0
PID	$0.6K_c$	$0.5P_c$	$0.125P_c$

3.5. Sistemas Digitais

Os sistemas digitais processam informações discretas, no domínio z. Os elementos de controle, ao invés de amplificadores e circuitos integradores e diferenciadores, são microcontroladores e computadores. A Função de Transferência é implementada por um algoritmo. Porém, seu correto funcionamento não depende apenas deste algoritmo implementado, mas da rapidez e precisão do sensor, do atuador e dos conversores A/D e D/A. As estabilidades relativa e absoluta dependem agora também do período de amostragem do

sistema. Os controladores digitais apresentam maior flexibilidade na alteração dos parâmetros do sistema (pois se altera apenas o algoritmo ao invés do hardware), tem maior imunidade a ruídos e com um único controlador digital é possível implementar a função de um ou mais controladores analógicos.

As grandezas encontradas na natureza são grandezas continuas ou analógicas, logo é necessário condicionar esses sinais para serem processados por sistemas digitais. A conversão D/A é, em geral, simples e realizada quase instantaneamente. A conversão A/D, por outro lado, é feita em 2 passos, e não é instantânea. É, em geral, composta por um bloco amostrador e um segurador. Um sinal é chamado de contínuo no tempo ou sinal analógico se o mesmo é definido para todo instante de tempo, como exemplo um sinal ondulatório na Figura 3.2 (a). Um sinal variante no tempo pode ser amostrado com um intervalo de tempo “T", formando uma sequência de valores discretos. Aplicando esta sequência discreta num sistema dinâmico contínuo, teremos uma resposta que será definida apenas nos instantes de amostragem, como ilustrado na Figura 3.2(b). Um sinal digital a ser aplicado em um sistema analógico reconstruído a partir de um sinal amostrado pode ser visto na Figura 3.2 (c):

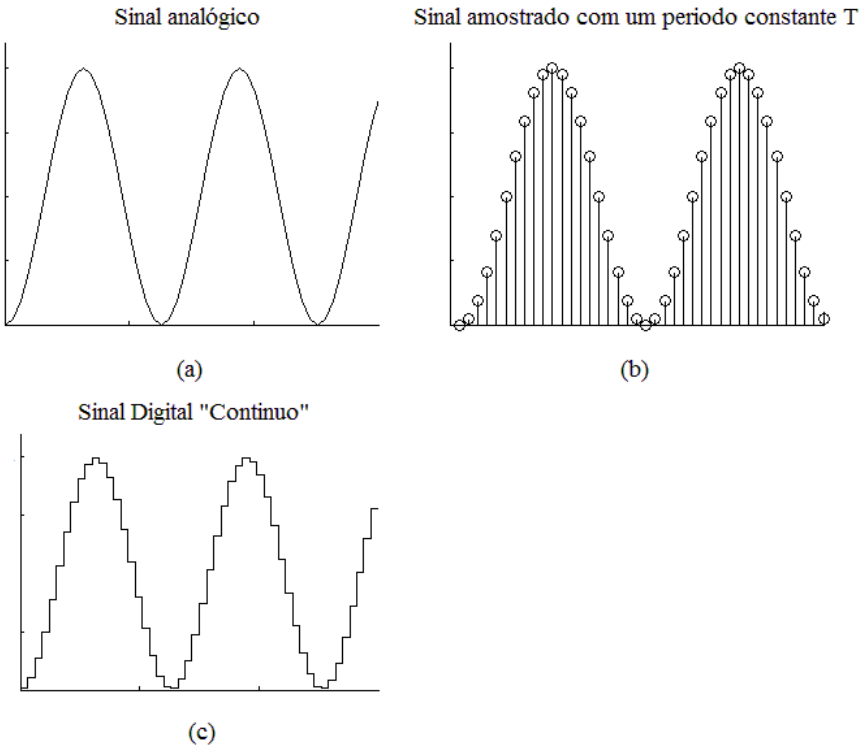


Figura 3.2 – (a) Sinal analógico. (b) Sinal amostrado com periodo constante T. (c) Sinal digital “contínuo”.

Para um controle digital é necessário converter sinais analógicos em sinais digitais, tendo em vista que se utiliza um computador para efetuar tal controle. Um sinal digital é

definido em instantes de tempo discretos. Estes instantes de tempo nos quais o sinal aparece são chamados instantes de amostragem, e o intervalo entre dois instantes de amostragem consecutivos é denominado período de amostragem.

O sinal amostrado pode ser definido pela Equação (3.11):

$$u^*(t) = u(t)\delta(t) = \sum_{k=0}^{+\infty} u(kT)\delta(t - kT) \tag{3.11}$$

Onde:

$u(t)$	=	Sinal de entrada
$\delta(t)$	=	Sinal impulso ou delta dirac
T	=	Período de amostragem

De acordo com o Teorema da Amostragem, o tempo de amostragem T_{sample} tem que ser pelo menos duas vezes maior que todas as frequências presentes no espectro do sinal original. Do contrário, ocorrerá o efeito de *aliasing*, que consiste na perda de informação dada pela baixa frequência de amostragem. O sinal de erro é convertido em um sinal discreto pelo conversor A/D. Esta conversão é realizada seguindo um período de amostragem pré-determinado de modo a garantir a correta amostragem do sinal analógico.

Capítulo 4

Circuitos do Projeto e Ferramentas

O objetivo deste Capítulo é descrever os circuitos componentes do sistema de controle de temperatura da brassagem. Para isso é necessário ter uma visão geral do sistema. A planta é ilustrada na Figura 4.1, pelo diagrama de blocos onde cada bloco é detalhado adiante:

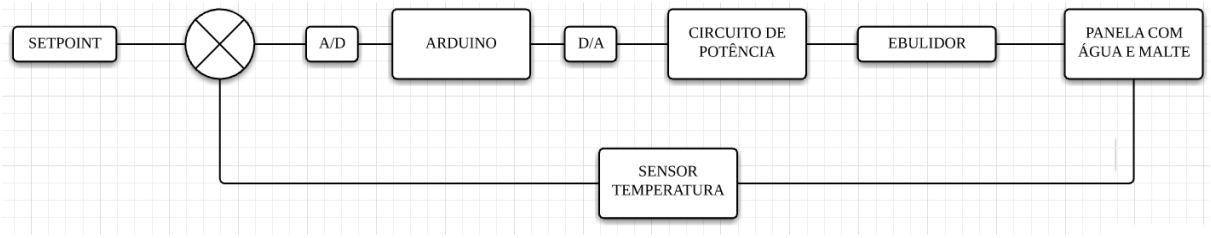


Figura 4.1 – Diagrama de blocos do projeto sob estudo.

Como estudado no Capítulo 3, o *setpoint* é definido de maneira computacional através de um algoritmo que implemente as rampas de brassagem descritas na Seção 2.3.4 do Capítulo 2. A medição de temperatura é feita a partir de um sensor e o sinal de erro calculado internamente no controlador digital a partir de um algoritmo que compare os sinais. Em seguida a aplicação do controle PID discreto e o cálculo do sinal de saída são feitos também computacionalmente, apenas para aplicar em um circuito de potência que acione o atuador que irá aquecer a panela.

4.1. FONTE TÉRMICA

A primeira etapa da determinação da fonte térmica foi avaliar os principais sistemas de aquecimento de uma grande quantidade de água contida na panela, objeto de estudo do trabalho. Os dois tipos principais são o aquecimento da água são o aquecimento por resistor elétrico de calefação e o controle da vazão do gás que serve de combustível para a chama no fundo da panela através de uma válvula.

4.1.1. Válvula de gás X Resistência de aquecimento

O sistema de controle de chama valvulado tem a vantagem de ser direto e não exigir uma adaptação da panela à aplicação. Porém ao se trabalhar com gás, uma série de riscos devem ser considerados. Primeiramente é preciso garantir que não há nenhum tipo de vazamento, pois este pode causar explosões. Analogamente, deve-se ter muito cuidado ao trabalhar com sistemas elétricos em um sistema alimentado por gás, evitando bobinas e indutores pois estes podem apresentar algum tipo de faiscamento por descarga estática.

O sistema de aquecimento por resistor de calefação de alta potência é mais caro e exige uma grande quantidade de energia elétrica da rede de alimentação. Além disso, alguns tipos de resistores tem a desvantagem de não serem facilmente acoplados à panela. Porém, é um método de aquecimento mais robusto, simples de ser implementado e controlado, pois a variável de controle (sinal elétrico) já é o próprio sinal amplificado que aciona o resistor. É um método mais seguro, evitando que erros de controle e ou projeto possam gerar impactos mais graves.

4.1.2. Resistência de aquecimento

Uma das formas mais comuns de aquecimento é a utilização de resistências com alta capacidade de dissipação de potência. Esse método é utilizado tanto em residências, nos chuveiros, como em processos industriais, em aquecimento de líquidos ou mesmo gases.

Basicamente um resistor é um dispositivo que dificulta a passagem de corrente elétrica por ele. Ao aquecimento do resistor devido à passagem de cargas elétricas se dá o nome de efeito Joule. Esse efeito é explicado em um nível atômico pelos choques internos causados por elétrons nos átomos do material.

A resistência efetiva de um resistor é definida pela Equação (4.1):

$$R_e = \frac{P}{I^2} \tag{4.1}$$

Onde:

R_e = Resistência do condutor (Ω)
P = Potência ativa aplicada ao resistor (W)
I = Corrente elétrica através do resistor (A)

O valor de resistência de um resistor é dado pela Equação (4.2) que demonstra a dependência que a resistência tem do comprimento do material e, inversamente, de sua área:

$$R = \rho \frac{L}{A} \tag{4.2}$$

Onde:

R_e = Resistência do condutor (Ω)
ρ = Resistividade do material (Ωm)
L = Comprimento do material (m)
A = Área da seção transversal (m^2)

Todos os resistores têm propriedades básicas que os caracterizam, e são analisadas em suas especificações:

- Valor nominal da resistência – é o valor de resistência em ohms (Ω) obtido sobre condições nominais de tensão e corrente.
- Tolerância – é uma faixa de variação aceitável do valor de resistência que o resistor pode apresentar.
- Linearidade – é a propriedade que permite a aplicação da Lei de Ohm no estudo da resistência, mantendo a relação proporcional entre tensão e corrente.
- Potência de dissipação nominal – é a potência máxima a que o resistor pode ser submetido sem ser danificado.
- Densidade – é a razão entre a potência de dissipação e o volume do resistor. Um resistor de alta densidade pode ser muito pequeno e dissipar a mesma potência que um resistor com grande volume e superfície de contato, no entanto com baixa densidade. A diferença entre os dois tipos de resistores está principalmente no espalhamento da potência dissipada.

Assim, dentro da classificação de aplicações de resistores (sensores, resistores eletrônicos, de precisão, etc.) estão os ebulidores de calefação ou aquecimento, como ilustrado na Figura 4.2. Esses ebulidores são caracterizados por terem alta capacidade de dissipação de potência. Os resistores de aquecimento possuem uma grande superfície de contato, derivado de sua densidade baixa. Dessa forma, conseguem aquecimento maior, mais uniforme, além de permitir a passagem de uma corrente maior. São, em geral, projetados para

resistir a altas pressões, devido à grande aplicação em tanques e tubos pressurizados. A conexão desses resistores é feita através de conexões de roscas.

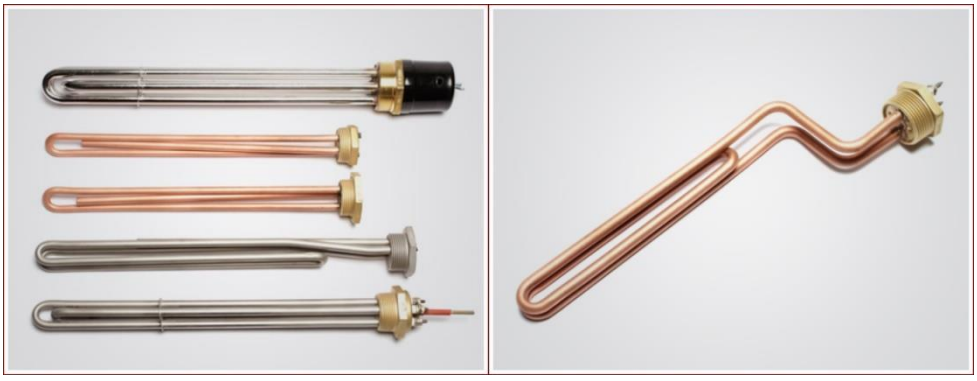


Figura 4.2 – Resistores de aquecimento de diversos tipos de liga. Fonte: SMS Resistências Elétricas

Outro aspecto importante desse tipo de resistor é o material que o compõe. Normalmente são ligas metálicas caracterizadas por alta resistividade, como a de níquel, cromo e ferro em proporções altas de ferro (12Ni + 12Cr + 76Fe). Mas, de acordo com a literatura específica de cerveja, são encontrados em aço inox para aplicações culinárias. A determinação da liga metálica deve levar em conta sua relutância à corrosão. Por esse motivo, os ebulidores possuem camadas superficiais de óxido que desaceleram o efeito do oxigênio ou líquidos corrosivos.

Como o projeto visa uma aplicação culinária, é de extrema importância a não corrosão e os cuidados com a deterioração do ebulidor, preferindo-se, dessa forma, ligas em aço inox à ligas de chumbo, pois são mais resistentes e confiáveis [13].

Para a especificação dos resistores de aquecimento deve-se levar em conta a potência de dissipação, a tensão de alimentação do ebulidor, o diâmetro e o comprimento linear do resistor, fixação e ligação em parafusos ou rabichos, o tipo e densidade da liga, esse último importante para avaliar a aplicação na culinária.

4.1.3. Cálculo de potência e escolha do ebulidor

O objetivo do projeto é fazer com que o sistema tenha um comportamento de temperatura que siga as rampas de brassagem à uma taxa de aquecimento de 1°C/min. Para a determinação da potência do ebulidor necessária para manter essa taxa, utiliza-se as relações do Capítulo 3. Substituindo a Equação (3.4), que define calor específico, na Equação (3.3), e rearranjando os termos, temos a seguinte relação:

$$\Delta Q = m \times c \times \Delta T \tag{4.3}$$

A potência utilizada será a quantidade de calor necessária para o aquecimento da mistura de água e malte na panela, durante todo o processo de brassagem. Com o tempo de duração determinados pelas próprias rampas, e a quantidade de calor também conhecida, a potência térmica será dada por:

$$P = \frac{\Delta Q}{t} = m \times c \times \frac{\Delta T}{t} = \ m \times c \times \frac{1℃}{min} \tag{4.4}$$

Substituindo na Equação (4.4) os dados de entrada de massa (4,09 kg de malte e 10 kg de água) e o calor específico da mistura, e substituindo graus Celsius por Kelvin, temos, então:

$$P = \left[\left(10kg \ \times \ 4186{,}6 \ \frac{J}{kgK}\right) + \left(4{,}09kg \ \times 1549 \frac{J}{kgK}\right)\right] \times \frac{1K}{min}$$

$$P = 803{,}36 \ W$$

Está é, então, a potência ativa mínima necessária para permitir que o ebulidor atue no sistema aquecendo-o à taxa desejada. A tensão de alimentação do ebulidor será determinada em função da instalação da rede. Como a produção é artesanal, leva-se em conta a tensão de alimentação residencial padrão de $127V_{RMS}$.

A determinação da resistividade do elemento leva em conta também os parâmetros de seção transversal e comprimento do ebulidor, como elucidado na Equação (4.2). Os dois fatores são dinâmicos, e, assim como o conjunto de ligação e fixação, devem levar em conta as dimensões da panela e as condições construtivas do sistema onde o ebulidor será inserido. A medida que se determina o comprimento, pode-se estimar a seção transversal.

Além da potência, foi necessário avaliar a linearidade do resistor, pois como elemento atuador, essa característica determina o desempenho do controle do sistema. Um resistor de característica não linear exige um tratamento muito mais complexo de atuação para que seja possível aplicar a potência adequada para manter a taxa de variação de temperatura exigida. Uma curva de linearidade do ebulidor foi obtida para determinar o valor de sua resistência elétrica, com o auxílio de um varistor, excitando o ebulidor com diferentes níveis de tensão *V* e obtendo seu equivalente de corrente *i*. Foram feitos três ensaios para as temperaturas de 50ºC, 60ºC e68ºC. A curva é exibida na Figura 4.3. O gráfico da Figura 4.3 apresenta um leve

deslocamento das curvas para as três diferentes temperaturas, apenas para facilitar a visualização adequada de cada curva. Na pratica, todas estariam sobrepostas.

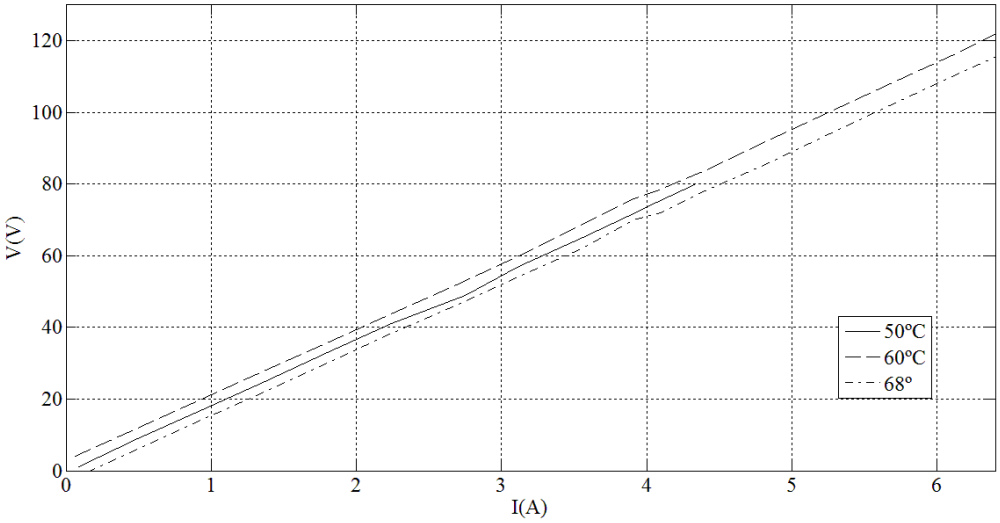


Figura 4.3 – Curva linearidade do ebulidor.

O ebulidor apresenta um valor médio de resistência de 18,34 Ω. A resistência térmica do ebulidor é de 0,0095°C/W.

4.2. CIRCUITO DE POTÊNCIA

A aplicação completa de potência envolve primeiramente um circuito detector de passagem por zero, cuja função é o sincronismo da tensão da rede à entrada do Arduino. O objetivo desse circuito é detectar quando a onda senoidal atinge o valor zero, através de um pulso elétrico na saída, que serve para acionar uma função de interrupção do Arduino. Esse circuito é muito utilizado para monitoramento da rede elétrica e para acionamentos de potência, como é o caso deste trabalho.

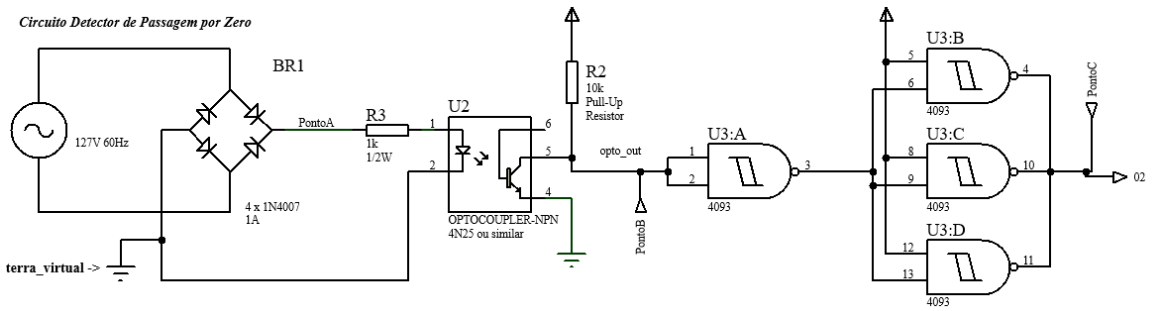


Figura 4.4 – Circuito Detector de Passagem por Zero

O circuito, da Figura 4.4 é composto primeiramente por uma ponte retificadora com o objetivo de retificar a onda vinda da rede. O sinal de saída da ponte retificadora pode ser visto na Figura 4.5, ilustrado pela onda senoidal em amarelo, de $127V_{rms}$ de amplitude e uma frequência de 120Hz (pois a onda é retificada). Em seguida, um optoacoplador para condicionar o sinal de $127V_{rms}$ para um nível de tensão aplicável à porta de entrada digital do Arduino. A função do optoacoplador é garantir segurança ao circuito visto que ele proporciona isolamento elétrico entre as partes do circuito, funcionando à base de excitação luminosa. Em todo momento em que houver uma tensão acima de zero, haverá emissão de luz do LED interno do optoacoplador, e o fototransistor satura, forçando um nível baixo de tensão na saída. Quando a tensão se aproxima de zero, o LED é desligado, cortando o fototransistor, e através de um resistor de *pull-up*, um nível alto é aplicado à saída. O sinal na saída conectada ao resistor de *pull-up* é ilustrado na Figura 4.5 pelo pulso em azul, onde podemos ver claramente que o pulso é emitido quando a senóide chega a zero. O pulso ocorre a cada período da onda senoidal, ou seja, 8,333ms.

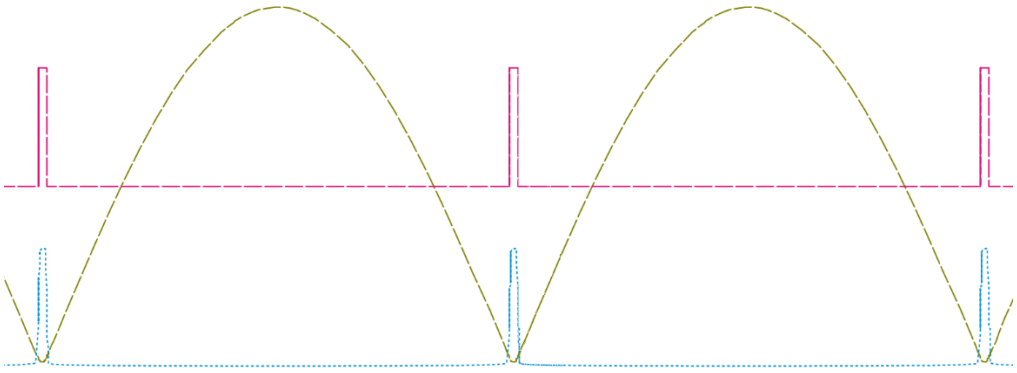


Figura 4.5 – Etapas do condicionamento de sinal do Circuito de Passagem por Zero: Amarelo – onda retificada. Azul – pulso de onda ligado a saida do resistor *pull-up*. Rosa – Pulso de onda corrigido através de um circuito de *buffer*.

Porém o pulso da saída do detector é distorcido e não ideal para a aplicação na entrada de microcontroladores, o que justifica a adição da etapa de correção e ajuste do pulso. Esse ajuste é feito no bloco do circuito através do circuito de *Schmitt Trigger*, com o CI CMOS 4093, e é exibido na Figura 4.5 pela onda em rosa. O circuito possui quatro portas lógicas NAND, onde a primeira é ligada como um inversor o que garante um nível de saída contrário ao de entrada. As outras três portas lógicas funcionam como uma espécie de amplificador, pois estão ligadas a uma entrada de 5V. O circuito funciona como uma espécie de *buffer*, garantindo um estado estável do pulso aplicado na entrada.

Em um segundo estágio temos o circuito de potência propriamente dito, onde o princípio de funcionamento deste circuito é o controle do ângulo de condução do TRIAC. A função do circuito de potência é aplicar a corrente adequada para uma carga de modo a fornecer potência necessária. No sistema em estudo, é possível manter a temperatura interna da panela próxima ao valor de temperatura desejado, a partir da potência injetada no ebulidor. Aplicando os pulsos de acionamento ao *trigger* em diversos pontos do sinal senoidal da rede de energia é possível aplicar a uma carga potências diferentes.

O TRIAC é uma chave eletrônica de estado sólido. A ação de controle, da saída do Arduino, é enviada ao TRIAC, que tem a função de chavear a corrente elétrica que alimentará o ebulidor promovendo ou não o aumento da temperatura da mistura. Assim, se o disparo for feito no início do semiciclo, a corrente pode ser conduzida para a carga e ela receberá maior potência. No entanto, se o disparo for feito no final do semiciclo, apenas uma pequena parcela da corrente será transferida para a carga que operará com potência reduzida. Esse circuito é ilustrado na Figura 4.6:

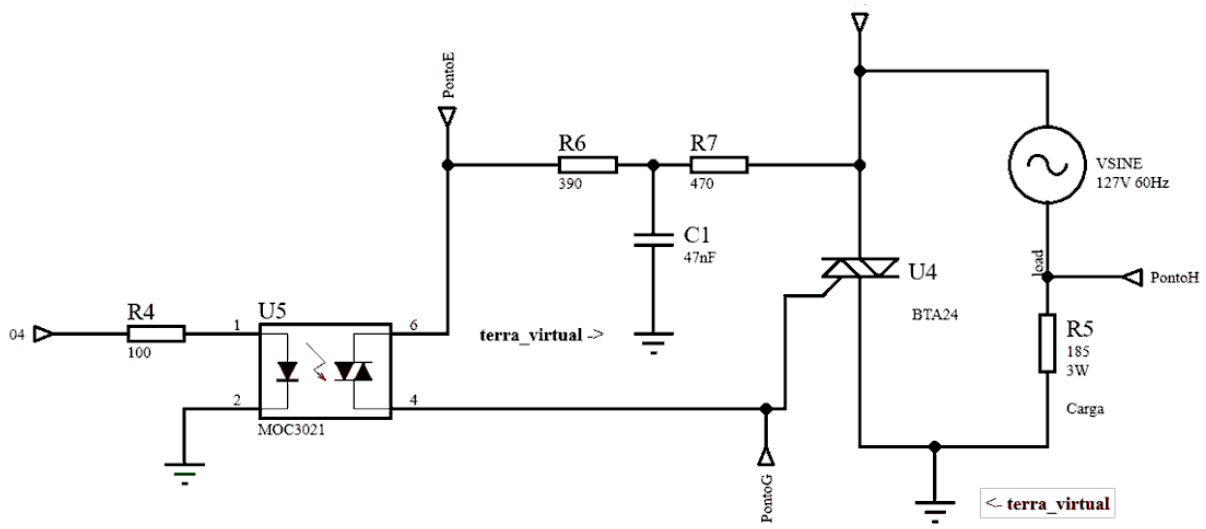


Figura 4.6 – Circuito de acionamento do TRIAC por ângulo de disparo

A saída do Arduino apresenta um pulso de acionamento, e esta saída é ligada a um FOTOTRIAC através de um resistor de polarização. A vantagem do dispositivo MOC3021 sobre fototransistores é a possibilidade de um sinal de maior intensidade na saída, sem a necessidade de várias etapas de condicionamento de sinal e amplificação para acionamento dos dispositivos. De forma a evitar um disparo falso do TRIAC, a tensão no anodo não deve aumentar rapidamente, acima da taxa de variação de tensão aceita pelo dispositivo. Para evitar que essas tensões transientes causem efeitos de chaveamento, utiliza-se um circuito amortecedor ou *RC Snubber* [14]. Se um transiente de chaveamento de alta velocidade

aparecer na tensão de alimentação do circuito, a taxa de elevação da tensão será reduzida no anodo devido ao efeito da dinâmica temporal do circuito RC.

O funcionamento do circuito de acionamento é ilustrado na Figura 4.7, que mostra quantidade de corrente aplicada à carga em função do ângulo de disparo do TRIAC, para diferentes ângulos, como 45°, 90°, 135° ou 172° (próximo do ciclo completo de 180°). Pode-se observar que a onda é seccionada antes do disparo. Para um ângulo α = 0, a potência aplicada à carga é máxima, enquanto para α = 180°, a potência é zero.

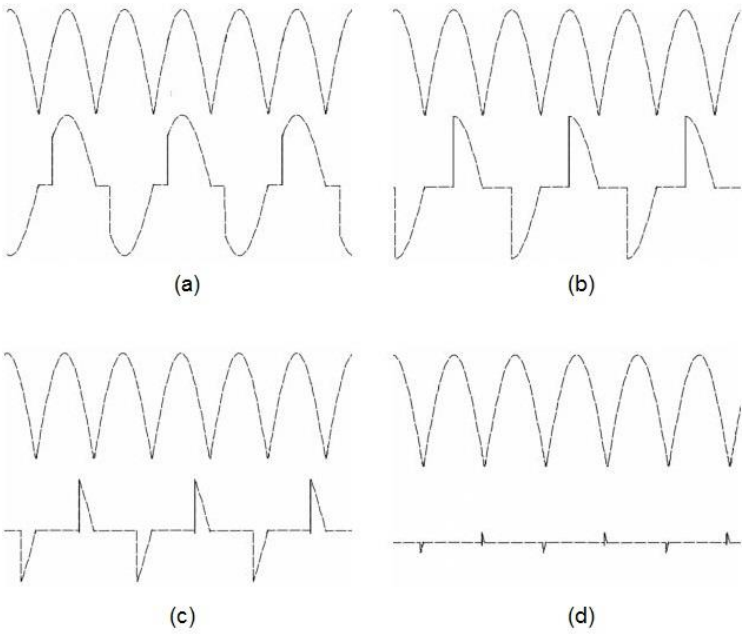


Figura 4.7 – Comparação entre a onda retificada e o acionamento do TRIAC a angulos de: (a) 45°. (b) 90°. (c) 135°. (d) 172°.

As equações dos valores de tensão e corrente eficazes e potência média de saída para cargas resistivas são dadas por:

$$P = \frac{{V_{pico}}^2}{2\pi R}\left(\pi - \alpha + \frac{sen\ 2\alpha}{2}\right) \tag{4.5}$$

Onde:

P	=	Potência aplicada a carga (W)
V_{pico}	=	Tensão de pico da rede (V)
R	=	Valor de resistência da carga puramente resistiva (Ω)
α	=	Ângulo de disparo (radianos)

4.4. TRANSDUTORES TÉRMICOS

Transdutores são componentes que sofrem variação de uma ou mais propriedades relacionadas a uma grandeza física em função da exposição à outra. À medida que se tem uma variação da grandeza elétrica em função de outra grandeza física, pode-se enviar um sinal elétrico para uma placa de aquisição de dados e posteriormente trabalhar com estes dados. O objetivo é estudar alguns componentes capazes de alterar suas propriedades elétricas em função da sua exposição a diferentes temperaturas, de forma a responderem a essas diferenças enviando respostas ao sistema.

4.4.1. Circuitos integrados inteligentes

São transdutores que contêm, além do sensor, entradas lógicas programáveis. Esses CI's possuem funcionalidades que permitem executar uma ampla quantidade de tarefas. Um exemplo básico é o DS18B20 da empresa *Dallas Semiconductor*. Esse CI é de fácil conexão com placas microcontroladoras, e não necessita de tratamento externo de dados, uma vez que possui internamente um conversor A/D e memória para os dados de temperatura captados.

4.4.2. Protocolo *One-Wire*

É um protocolo de comunicação serial que permite que um dispositivo se conecte com outro através de uma única conexão. O *one-wire* pode ser composto de apenas dois cabos: o de transmissão de dados e o de conexão para terra. Para permitir a comunicação através dessa conexão, os dispositivos que trabalham com esse protocolo vêm com um capacitor acoplado que armazena energia quando ligado a um sistema e alimenta o dispositivo no momento da transmissão dos dados seriais.

Essa interface de comunicação apresenta uma série de vantagens:

- Requer apenas um fio de comunicação;
- Utiliza a alimentação parasita, eliminando muitas vezes a necessidade de alimentação externa;
- Comandos podem ser enviados pelo barramento para um grande conjunto de dispositivos conectados a este;
- Baratos;

Cada dispositivo *one-wire* tem um código identificador exclusivo que o permite comunicar individualmente com dispositivos de controle. Dessa maneira o protocolo

consegue distinguir cada dispositivo conectado em um barramento. Essa característica também tem a vantagem de permitir a comunicação e o controle particular de cada dispositivo, mas traz a desvantagem de exigir uma programação complexa para um sistema composto por muitos dispositivos.

4.4.3. Sensor de circuito integrado DS18B20

O sensor DS18B20 é um tipo de sensor digital que envia os dados de temperatura através do protocolo *one-wire*. Todo dispositivo que utiliza a interface *one-wire* necessita de um pino ou barramento de entrada/saída bidirecional. Felizmente isso é quase padrão nos microcontroladores atuais.

O sensor DS18B20 possui uma memória interna onde armazena os dados de temperatura convertidos em informação digital binária. O sensor trabalha em uma faixa de temperatura que pode variar entre -55°C e 125°C, com uma precisão de 2°C de variação. Caso o sensor trabalhe entre -10°C e 85°C sua precisão aumenta para uma 0,5°C de variação. Pode ser alimentado com uma tensão entre 3V e 5,5V, a uma corrente que varia entre 1mA a 1,5mA. Outra característica fundamental do DS18B20 é que seu tempo máximo de aquisição de dados é de 750ms, caso seja exigida uma precisão de 0,0625°C. Para precisões menores o tempo pode ser de até 93,75ms.

O DS18B20 aceita alimentação a partir de uma fonte externa, sendo a forma mais fácil para garantir seu correto funcionamento. A alimentação parasita exige um resistor de *pull-up* para permitir a carga do capacitor interno enquanto a comunicação serial não está sendo feita. Porém o método de conexão parasita envolve duas principais desvantagens: dependendo do resistor de *pull-up* a tensão que alimenta o sensor pode cair, causando um reset no dispositivo; durante a ação de alimentação pelo *pull-up* nenhuma outra comunicação pode ser feita a partir do barramento, pois este está sendo utilizado como fonte.

Os circuitos integrados são os transdutores com maior linearidade e baixo custo. Porém apresentam auto aquecimento e tem resposta lenta. Essa última desvantagem, no entanto, pode ser desconsiderada, pois se avaliarmos o sistema foco deste trabalho, temos uma dinâmica temporal de minutos, o que dispensa respostas mais rápidas do que a ordem de grandeza de segundos.

O CI inteligente é, desta forma, a opção mais viável para o projeto dadas as necessidades de linearidade do mesmo, baixo custo e, principalmente, as facilidades de implementação em um sistema microcontrolador. Como contribuição adicional,

especificamente o DS18B20 é utilizado na produção de cerveja, de acordo com a comunidade de produtores artesanais.

4.5. ARDUINO

Arduino é uma plataforma eletrônica, que permite o controle de sistemas interativos de baixo custo. É reconhecido por sua facilidade de implementação e por ter todo seu material *Opensource*, o que permite um vasto conteúdo de desenvolvimento acessível a todos.

A plataforma é composta essencialmente de duas partes: O Hardware e o Software. O microcontrolador é caracterizado pela sua arquitetura. No caso do Arduino, sua arquitetura é a AVR, uma modificação da arquitetura Harvard, desenvolvida pela empresa Atmel. A arquitetura Harvard veio priorizar a velocidade de trabalho do microcontrolador, e se caracteriza principalmente por ter dois conjuntos de memória separados: um dedicado a instruções e outro dedicado a dados. A arquitetura AVR também tem como característica serem os chamados RISC (*Reduced Instruction Set Computer*), onde se caracterizam por ter um conjunto reduzido de processamento e instruções que podem executar.

Para fins de objetividade e considerando o que será aplicado ao projeto, iremos focar especificamente na placa Arduino UNO.

4.5.1. Alimentação da placa Arduino UNO

A placa do Arduino UNO pode ser alimentada por uma conexão USB ou uma fonte de alimentação de tensão contínua, como exibido na Figura 4.8 como “Alimentação Externa” e “Conector USB”. Os valores de tensão da fonte devem ter entre 6V a 20V, porém abaixo de 7V ou acima de 12V o Arduino UNO pode apresentar um comportamento instável. O ideal é manter uma alimentação entre 7V e 12V. O Arduino possui um regulador de tensão interno, NCP1117, logo na entrada do pino de alimentação, auxiliando na conservação de uma tensão estável.

O Arduino UNO possui também um circuito comutador, caso seja alimentado por uma fonte contínua e esteja conectado ao USB simultaneamente. Neste caso, a placa prioriza a alimentação da fonte contínua e recebe da entrada USB apenas dados. O Arduino não possui um dispositivo de acionamento/desligamento. Para desligar o microcontrolador basta apenas retirar o pino da fonte de alimentação da entrada da placa.

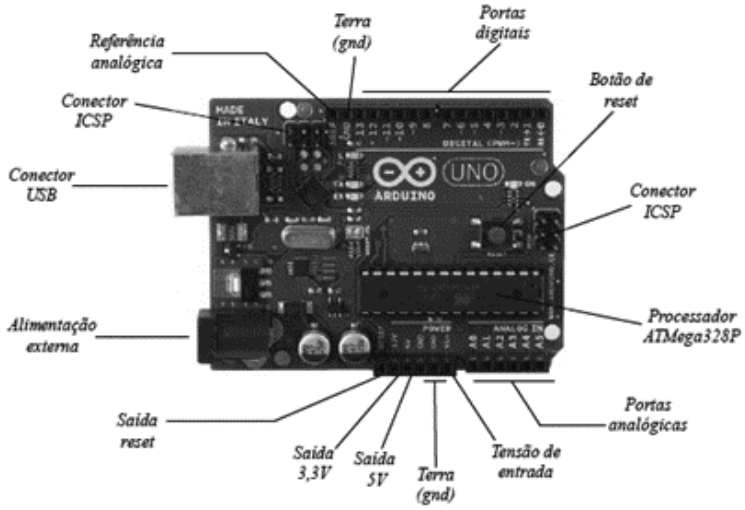


Figura 4.8- Layout microprocessador Arduino UNO.

4.5.2. Conectores

O Arduino UNO possui três grupos de conectores: conectores de alimentação, conectores analógicos de entrada e conectores digitais de entrada/saída. Na Figura 4.8, estão descritos como “Portas analógicas”, “Portas digitais”, “Tensão de Entrada” e as saídas. A Tabela 4.1 exibe os principais pinos do Arduino e descreve suas respectivas funções:

Tabela 4.1 - Principais Pinos do Arduino UNO e suas funções

PINO	FUNÇÃO
AREF - Analog Reference	Tensão de referência para entradas analógicas.
Digital GND	Terra digital
Digital I/O (2-13)	Pinos de entrada e saída digitais, que operam em 5V e podem receber ou fornecer um máximo de 40mA de corrente. Os pinos 2 e 3 também podem ser usados para gerar interrupções.
Serial OUT (TX) e Serial IN (RX)	Usados para transmitir e receber dados seriais TTL.
Analog In (0-5)	Entradas analógicas, que fornecem uma resolução de 10 bits (1024 valores distintos); medem por padrão valores de 0 a 5V.
PWM	Os pinos 3,5,6,9,10 e 11 podem ser usados como saída PWM de 8 bits.
SPI	Os pinos 10.11.12 e 13 suportam comunicação SPI com o uso da biblioteca apropriada.
Reset	Reseta o microcontrolador quando em nível baixo.
Voltage In	Pode-se fornecer tensão elétrica através desse pino ou acessá-la caso se use uma fonte de alimentação externa.
5 Volt Power Pin	Saída regulada de 5V
3.3 Volt Power Pin	Saída regulada de 3.3V

Os pinos digitais 3,5,6,9,10 e 11 podem ser usados como saídas PWM de 8 bits. Esses são identificados pelo sinal de “~” que antecede a numeração do pino. A frequência do sinal PWM na maioria dos pinos é de aproximadamente 490 Hz, exceto para os pinos 5 e 6 que têm uma frequência de aproximadamente 980 Hz.

Os microprocessadores da arquitetura AVR tem uma memória EEPROM interna para armazenamento de instruções permanentes relacionadas à ligação e acionamento do dispositivo. Essa memória é caracterizada por manter os dados mesmo quando o microprocessador é desligado, além de não ser uma memória acessível em condições usuais, só sendo possível através de dispositivos auxiliares externos, para fins de aplicações especiais. Memória flash é um tipo especial de memória do tipo EEPROM, que, apesar de ser não-volátil, possui a particularidade de permitir que múltiplos endereços sejam escritos através de um conjunto de instruções internas do microcontrolador.

O microcontrolador da plataforma Arduino UNO, exibido na Figura 4.8, o ATmega328P, possui algumas subdivisões, e por isso, serão tratadas o mais comum deles: o Arduino UNO. As principais características do Arduino UNO são:

- Microcontrolador de 28 pinos
- 32Kb de memória de programação FLASH de palavra de 8kb
- Clock de 16MHz

4.6. SOFTWARES E FERRAMENTAS COMPUTACIONAIS

4.6.1. IDE Arduino

O software que controla os pinos e ações do Arduino pode ser desenvolvido na Arduino IDE (*Integrated Development Environment*), que é a interface de desenvolvimento bastante simples e didática, como é possível ver na Figura 4.9. A interface exige que seja configurada a porta de comunicação serial e o tipo de Arduino para qual se destina o programa. O código Arduino tem o formato ".ino" e o código compilado tem o formato ".hex" que é diretamente carregado no microcontrolador. Em todo código existe o procedimento de *setup()*, onde é feito o ajuste das configurações de portas de entrada, saída e interrupções, e o procedimento *loop()* que é um laço infinito que executa a rotina principal. A linguagem de programação utilizada é o C++, com pequenas modificações.

Além das bibliotecas nativas do Arduino, o projeto envolve as bibliotecas *OneWire.h* e a *DallasTemperature.h* que possuem um conjunto de rotinas de inicialização, configuração e manipulação dos dispositivos *one-wire*, como o sensor DS18B20.

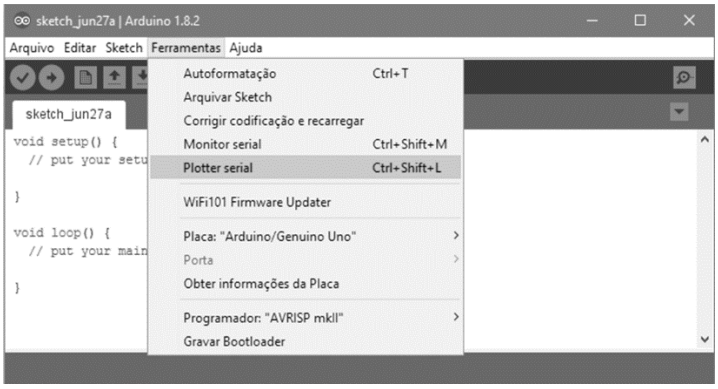


Figura 4.9 - IDE do Arduino. Ferramentas Monitor e Plotter serial.

O Arduino Uno e vários outros modelos de Arduino conseguem se comunicar com o PC através da porta USB. Esta comunicação se dá de forma serial, e pode ser acessada pelo Monitor Serial na IDE do Arduino, que permite enviar e receber dados. Além do Monitor Serial a IDE do Arduino possui outra ferramenta de visualização de dados: o Plotter Serial, que possibilita a impressão dos dados em forma gráfica. A Figura 4.9 exibe como é possível selecionar cada uma das ferramentas após o carregamento do programa na placa.

4.6.2. Proteus® e pacote Arduino

O Proteus® é um software de simulação que oferece facilidades para desenhos esquemáticos, simulação SPICE e layout de circuito. É uma ferramenta útil para a simulação e a construção de circuitos elétricos onde estão disponíveis vários componentes, incluindo microcontroladores, com a possibilidade de observar o funcionamento de forma virtual. A simulação do Arduino no Proteus® exigiu a instalação de uma biblioteca popular da internet: o Simulino. Este ambiente é um dos primeiros projetos de simulação do Arduino no Proteus®. Com a biblioteca é possível simular diversas instâncias do Simulino no mesmo projeto, sem erro de duplicidade. Além disso, os componentes permitem o upload do código ".hex" em linguagem de Arduino de maneira simples.

Capítulo 5

Resultados

Esta etapa do trabalho tem por objetivo demonstrar os métodos de obtenção da Função de Transferência, o projeto e a análise do controlador, a validação do controlador e sua avaliação para as exigências do trabalho, além do funcionamento dos circuitos auxiliares.

5.1. DESENVOLVIMENTO DA FUNÇÃO DE TRANSFERÊNCIA DA PLANTA

5.1.1. Função de Transferência a partir do método analítico

A finalidade do trabalho é controlar a temperatura do sistema de brassagem. O sistema de brassagem em questão consiste essencialmente de uma mistura composta por 10 kg de água e 4,09 kg de grãos de malte, uma panela cilíndrica de alumínio de 25 cm de altura e 28 cm de diâmetro, aquecidos por um ebulidor acoplado internamente ao fundo da panela e conectado à uma fonte de corrente externa, servindo de forte térmica. Além da montagem básica do sistema de brassagem temos a placa de controle, o sensor para aquisição de temperatura, o circuito de potência e o ebulidor que age como atuador do sistema.

Inicialmente o sistema deverá ser preparado e aquecido à temperatura base de 50°C. A Figura 5.1 ilustra o sistema térmico:

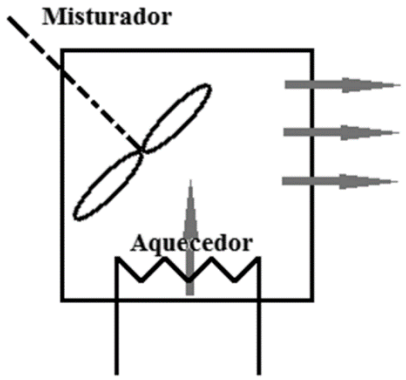


Figura 5.1 – Representação do sistema térmico composto por uma panela com água e malte.

À medida em que a mistura de água e malte começar a esquentar o sistema de controle deverá agir de forma a manter a temperatura desejada estável.

A variável de entrada da planta é a taxa de fluxo de calor q (W) do ebulidor, controlada pela corrente que passa por este. A variável a ser controlada é a temperatura da mistura de água e malte. E a taxa de aquecimento especificada pelo projeto é de 1°C/min. Para o sistema de controle completo, o *setpoint* será determinado como temperatura em °C, assim como o sinal medido pelo sensor. Dessa forma o sinal de erro também será um sinal de temperatura.

Algumas hipóteses de simplificação são consideradas. Para a análise do fluxo transitório de calor do sistema, supõe-se que a temperatura depende apenas do tempo, e é uniforme em toda extensão do sistema. Essa suposição é ilustrada sob a forma de um misturador, como na Figura 5.1. Além disso, a transferência de calor por convecção é representada por um valor adequado de coeficiente de transferência de calor por convecção h_c. Outra simplificação é a de que a mistura de água e malte e a panela são isotérmicas. O sistema é fechado, considerando a panela de alumínio como o volume de controle, pois não haverá adição posterior de água após o início do processo, nem saída de matéria, uma vez que a água não atinge sua temperatura de ebulição.

A taxa de transferência de calor no ebulidor pode ser obtida a partir da Equação (3.8), considerando o calor de entrada como a potência P gerada no ebulidor. Como pode-se observar na Equação (5.1), toda potência elétrica de entrada do resistor é convertida em calor. No caso particular do resistor, não foi considerada a energia armazenada no mesmo:

$$q_{in} = q_{out} \tag{5.1}$$

$$i^2 R = \left[\overline{h_{c\_a}} A_r (T_r - T_a)\right]$$

Onde:

R	=	Resistência (linear na faixa de temperatura) do resistor de aquecimento (Ω)
i	=	Corrente elétrica que percorre o resistor (A)
$\overline{h_{c\_a}}$	=	Coeficiente médio de transferência de calor da água (W/m^{2}K)
A_r	=	Área externa do resistor, ou a área de contado do resistor com a água (m^2)
T_r	=	Temperatura do resistor (K)
T_a	=	Temperatura da água (K)

O calor de entrada é a potência elétrica gerada, e o calor de saída é, então o calor de entrada menos o armazenado no resistor. Mas este calor de saída do resistor é o próprio calor

de entrada na água. Aplicando a mesma dedução, em que substituímos os valores térmicos na Equação (3.8), temos, considerando a água um sistema fechado, a Equação (5.2):

$$q_{in\_a} - q_{out\_a} = \frac{\partial E_a}{\partial t} \tag{5.2}$$

$$\left[\overline{h_{c\_a}} A_r (T_r - T_a)\right] - \left[\overline{h_{c\_a}} A_p (T_a - T_p)\right] = (C_a)\frac{\partial T_a}{\partial t}$$

De maneira análoga temos a Equação (5.3), relacionando o calor fornecido pela água com o calor armazenado pela panela com as perdas para o ambiente:

$$q_{in\_p} - q_{out\_p} = \frac{\partial E_p}{\partial t} \tag{5.3}$$

$$\left[\overline{h_{c\_a}} A_p (T_a - T_p)\right] - \left[\overline{h_{c\_amb}} A_p (T_p - T_{amb})\right] = (C_p)\frac{\partial T_p}{\partial t}$$

A partir das equações e do estudo sobre a equivalência entre modelos térmicos e elétricos, podemos chegar à um circuito térmico que representa o sistema em questão. Esse circuito é exibido na Figura 5.2:

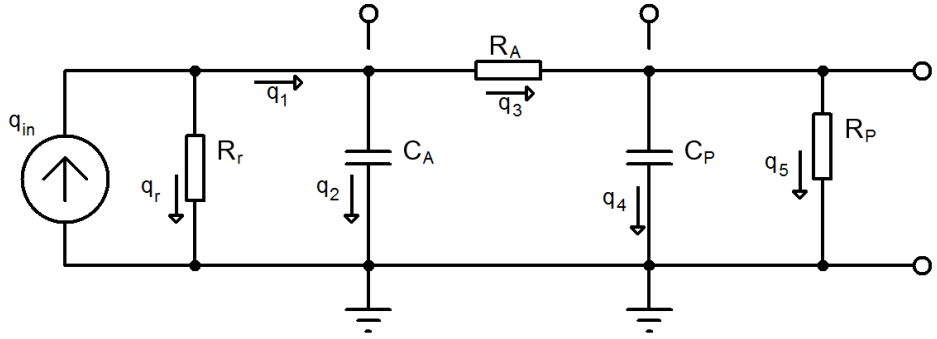


Figura 5.2 – Circuito elétrico equivalente ao circuito térmico.

Onde:

q_{in} = Taxa de fluxo de calor de entrada

R_r = Resistência ao fluxo de calor transmitido do resistor para a água. $\frac{1}{\overline{h_{Água}} A_{Resistor}}$

q_r = Taxa de fluxo de calor que aquece o resistor

q_1 = Taxa de fluxo de calor que atravessa o sistema

C_A = Calor armazenado na água. $c_{Água} V_{Água} \rho_{Água}$

q_2 = Modelo de taxa de fluxo de calor que percorre a capacitância da água. Equivale à energia armazenada.

R_2 = Resistência ao fluxo de calor transmitido da água para a panela. $\frac{1}{\overline{h_{Água}} A_{panela}}$

q_3 = Taxa de fluxo de calor que atravessa a panela

C_P = Calor armazenado na panela. $c_{panela}V_{panela}\rho_{panela}$

q_4 = Modelo de taxa de fluxo de calor que percorre a capacitância da panela.

R_P = Resistência ao fluxo de calor transmitido da panela para o ambiente. $\frac{1}{\overline{h_{Ar}}A_{panela}}$

q_5 = Modelo de Taxa de fluxo de calor que vai para o ambiente.

Desenvolvendo as Equações de (5.1) a (5.3) e analisando o circuito ilustrado na Figura 5.2 chegamos à Função de Transferência do sistema térmico estudado, no domínio de Laplace, na Equação (5.4):

$$\frac{T_{out}(s)}{Q_{in}(s)} = \frac{{}^{1}/_{C_A R_A C_P}}{\left[s^2 + \left(\frac{1}{C_P R_P} + \frac{1}{R_A C_P} + \frac{1}{C_A R_A} + \frac{1}{C_A R_R}\right)s + \left(\frac{R_R + R_A + R_P}{R_R C_A R_A C_P R_P}\right)\right]} \tag{5.4}$$

A partir das equações previamente desenvolvidas pode-se montar o diagrama de blocos do sistema representado na Figura 5.3:

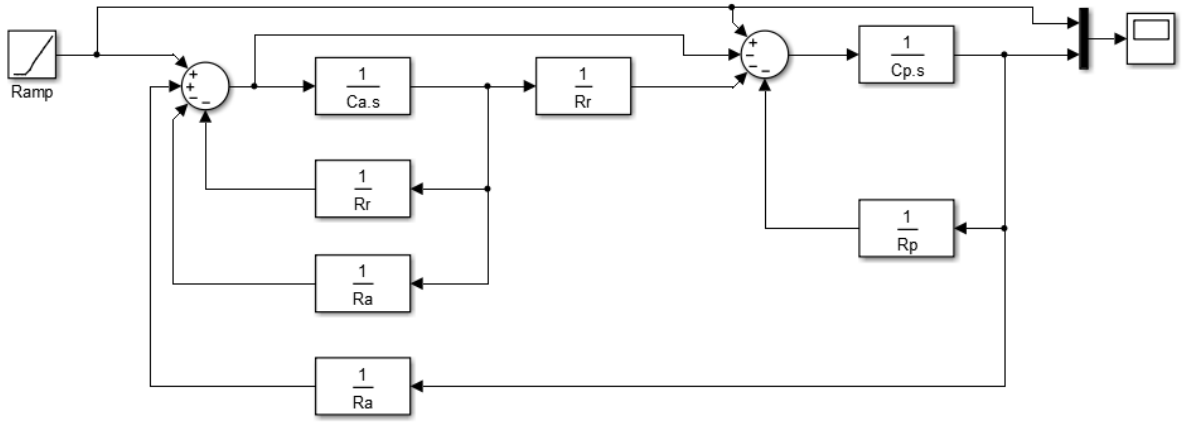


Figura 5.3 – Diagrama de Blocos no Simulink.

A partir do modelo matemático, do projeto do Simulink no Matlab®, e dos parâmetros de resistência e capacitância exibidos na Tabela 5.1, pode-se gerar a resposta do sistema à uma excitação em degrau, exibida na Figura 5.4:

Tabela 5.1 – Parâmetros da Função de Transferência

$C_{ÁGUA}$	$4.114 \cdot 10^{+07}$
C_{PANELA}	$7.923 \cdot 10^{+03}$
$R_{RESISTOR}$	$9.533 \cdot 10^{-03}$
$R_{ÁGUA}$	$7.831 \cdot 10^{-02}$
R_{PANELA}	$2.176 \cdot 10^{-01}$

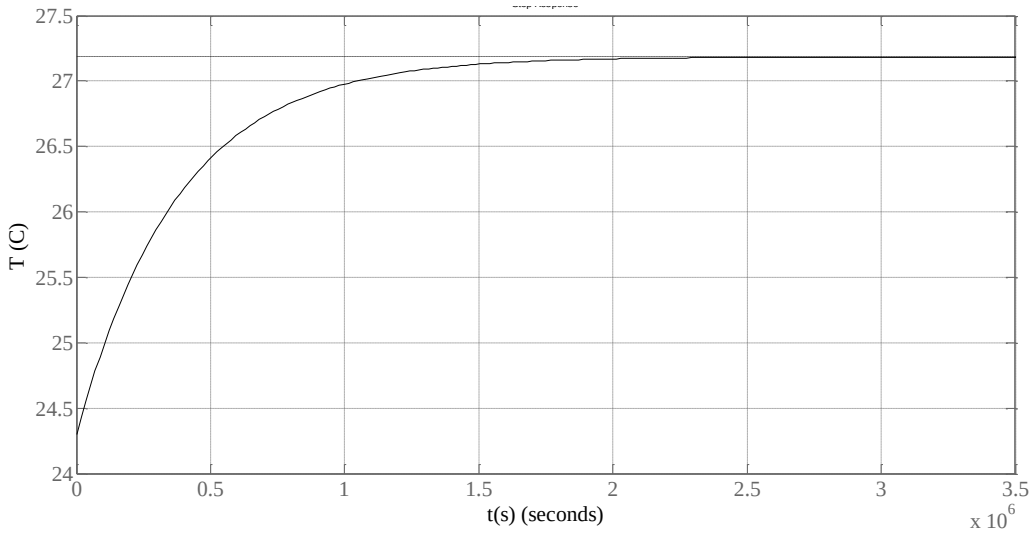


Figura 5.4 – Função de Transferência calculada: resposta ao degrau.

Os valores traduzidos pela resposta do sistema, em malha aberta, à uma excitação à um sinal de entrada em degrau, não representaram o comportamento esperado ou mesmo real do modelo. Apesar da forma da resposta corresponder ao esperado, a dinâmica temporal do sistema calculado foi muito mais lenta do que o sistema apresenta, levando a crer que a Função de Transferência calculada não representou adequadamente o sistema térmico prático.

5.1.2. Validação a partir do modelo empírico

Para obter a Função de Transferência do sistema descrito anteriormente foram executados ensaios de perturbações em degrau da grandeza de entrada, taxa de fluxo de calor, controlada a partir do monitoramento da potência elétrica de entrada. Assim, para a aplicação do método empírico, foram injetadas potências de entrada de diferentes valores, e coletados os dados de variação de temperatura, ao longo do tempo, em segundos. Os dados são apresentados na Figura 5.5:

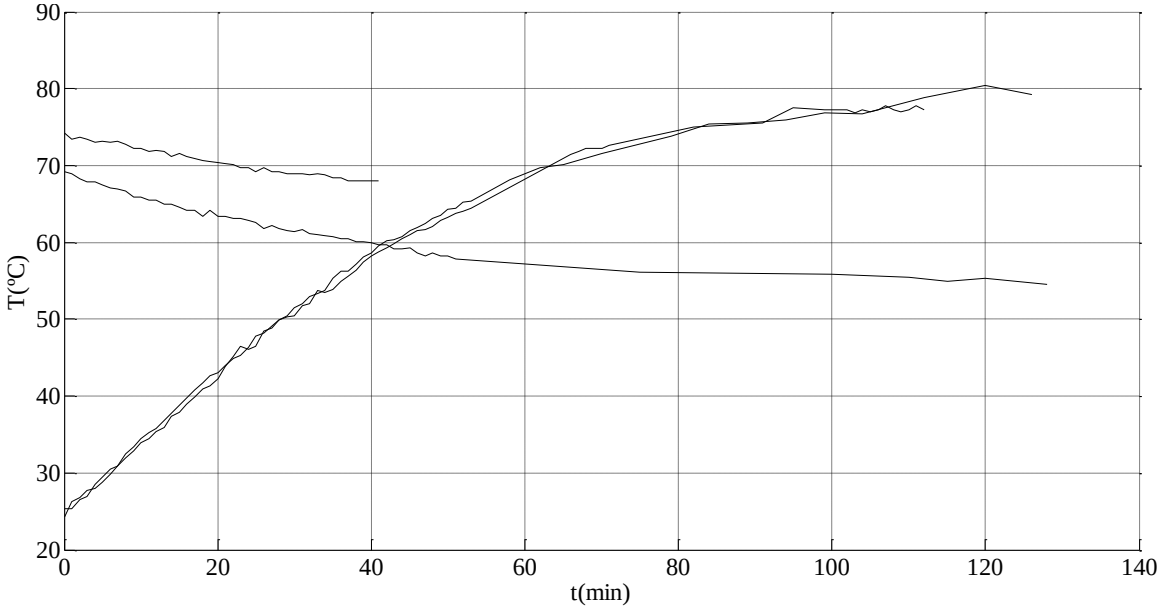


Figura 5.5 – Dados coletados de Temperatura (°C) x tempo (s).

De posse dos dados, foram aplicados dois métodos de ajuste da curva de resposta baseado na resposta do processo ao degrau. Em ambos o foco consiste na identificação dos parâmetros de ganho *K*, a constante de tempo τ e um tempo morto ϑ. Para a aproximação de Ziegler–Nichols consideramos uma Função de Transferência genérica como na Equação (5.5):

$$G(s) = \frac{K \times e^{-\vartheta s}}{\tau s + 1} \tag{5.5}$$

Onde:

K = Ganho do processo
τ = Constante de tempo do processo
q_r = Constante de tempo morto do processo

Determina-se o tempo morto ϑ a partir da interseção da tangente à curva de resposta em degrau, com o valor inicial da variável de saída, enquanto é aproximado para 63,2% do valor de acomodação do processo.

O método de Smith pode ser aplicado à primeira e segunda ordem. Considerando a aproximação de segunda ordem, a Função de Transferência genérica é ilustrada na Equação (5.6):

$$G(s) = \frac{K \times e^{-\vartheta s}}{\tau^2 s^2 + 2\xi\tau s + 1} \tag{5.6}$$

Este segundo método de aproximação envolve a determinação de vários parâmetros, o que torna o processo mais difícil. No entanto, a Figura 5.6 justifica a aproximação de Smith

em relação a de Ziegler–Nichols, uma vez que a primeira é muito mais fiel aos dados que a segunda.

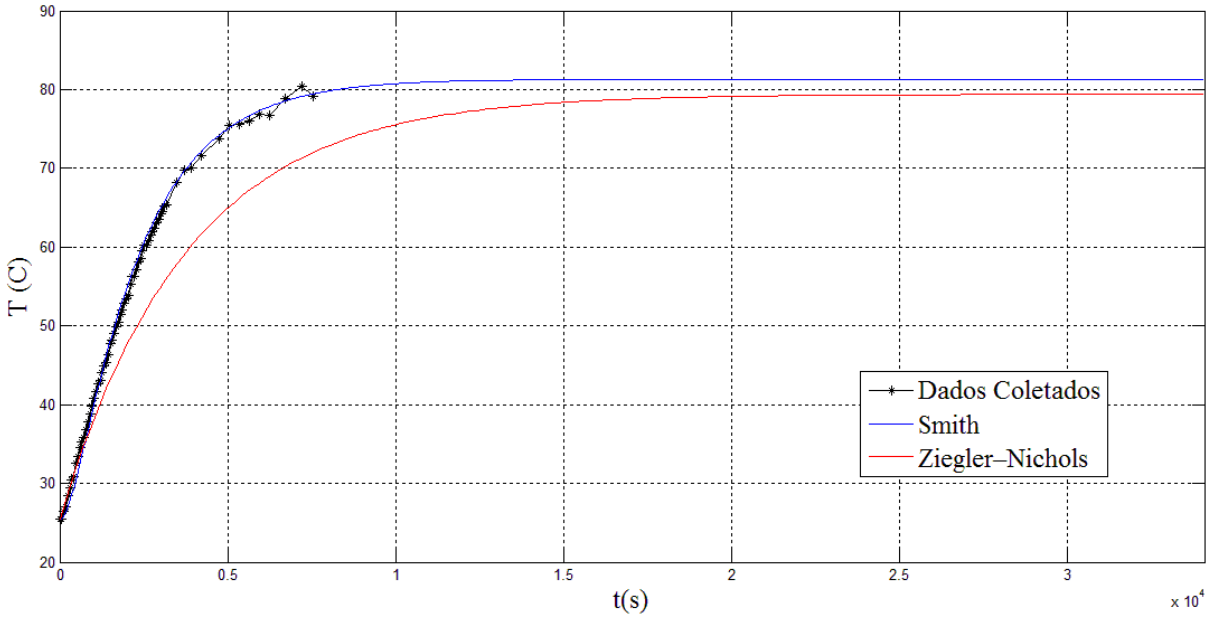


Figura 5.6 – Ajuste da Função de Transferência: Ziegler-Nichols e Smith.

Os dados coletados não apresentam tempo morto, de forma que essa parcela foi desconsiderada. Assim, a aproximação final da Função de Transferência é a Equação (5.7)

$$\frac{T_{out}(s)}{Q_{in}(s)} = \frac{8{,}379 \cdot 10^{-8}}{s^2 + 0{,}003s + 1{,}207 \cdot 10^{-6}} \tag{5.7}$$

Os dados da Função de Transferência obtida empiricamente se mostraram mais fiéis ao comportamento do sistema do que o método analítico. Vale considerar que devido à proximidade da resposta da Função de Transferência com os dados coletados pelos ensaios, bem como o fato de terem sido feitos quatro ensaios onde três deles foram validados, o modelo foi considerado satisfatório. Como uma análise complementar, a resposta do sistema se assemelha a uma resposta de segunda ordem superamortecida, o que condiz com as análises teóricas, sendo também uma típica resposta de um sistema térmico genérico. Outra consideração importante a se fazer é que o modelo foi ajustado para ser o mais fiel aos dados justamente na faixa de 50°C a 75°C, intervalo onde ocorrerá o processo de brassagem.

5.2. Comportamento e dinâmica do sensor

5.2.1. Simulação e implementação do sensor DS18B20

O objetivo da análise do sensor é determinar sua Função de Transferência, seu efeito na resposta do sistema e seu tempo de amostragem de temperatura.

Para se determinar a Função de Transferência do sensor temos que chegar em uma relação de tensão/temperatura gerada pela medição. Para a análise do erro de offset foi feita uma coleta de dados com o sensor DS18B20 e comparado com dados coletados por um termômetro de bulbo. O DS18B20, por ter vasta aplicação voltada para Arduino, não apresentou erro de offset visível. Porém o teste foi feito com apenas um sensor, e ao longo da literatura existem registros de sensores defeituosos que exigem essa validação inicial.

O DS18B20 pode então ser modelado por um ganho unitário e um amostrador, pois a discretização dos dados já ocorre internamente, fazendo com que a função do conversor A/D seja feita logo na etapa de medição.

O circuito de medição de temperatura é apresentado na Figura 5.7, onde mostra o sensor conectado à porta digital 7 do Arduino. O terminal 01 é ligado ao Monitor Display do Arduino. Um resistor de *pull-up* é necessário para evitar que em instantes variados o barramento de comunicação do sensor fique flutuante, sem sinal lógico definido. Como todo dispositivo *one-wire*, este permite dois tipos de ligação. A ligação de alimentação externa foi escolhida arbitrariamente, mas o circuito funciona igualmente na configuração parasita.

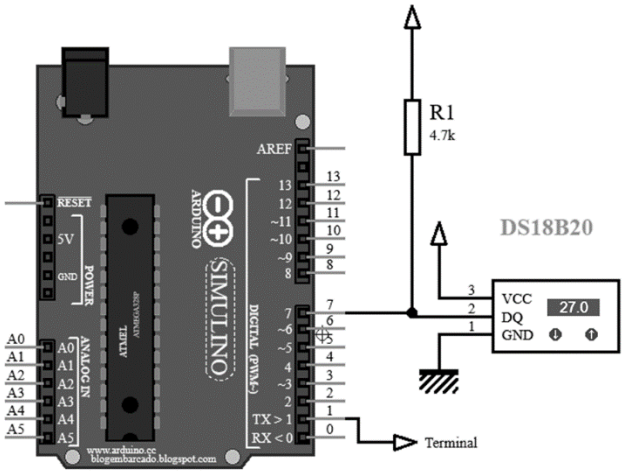


Figura 5.7 – Circuito de medição de temperatura: conexão entre Arduino e sensor DS18B20.

O código de aquisição de dados do sensor DS18B20 é apresentado no Anexo da página 79. A partir da simulação, os dados são exibidos no Monitor Serial, como mostrado na Figura 5.8:

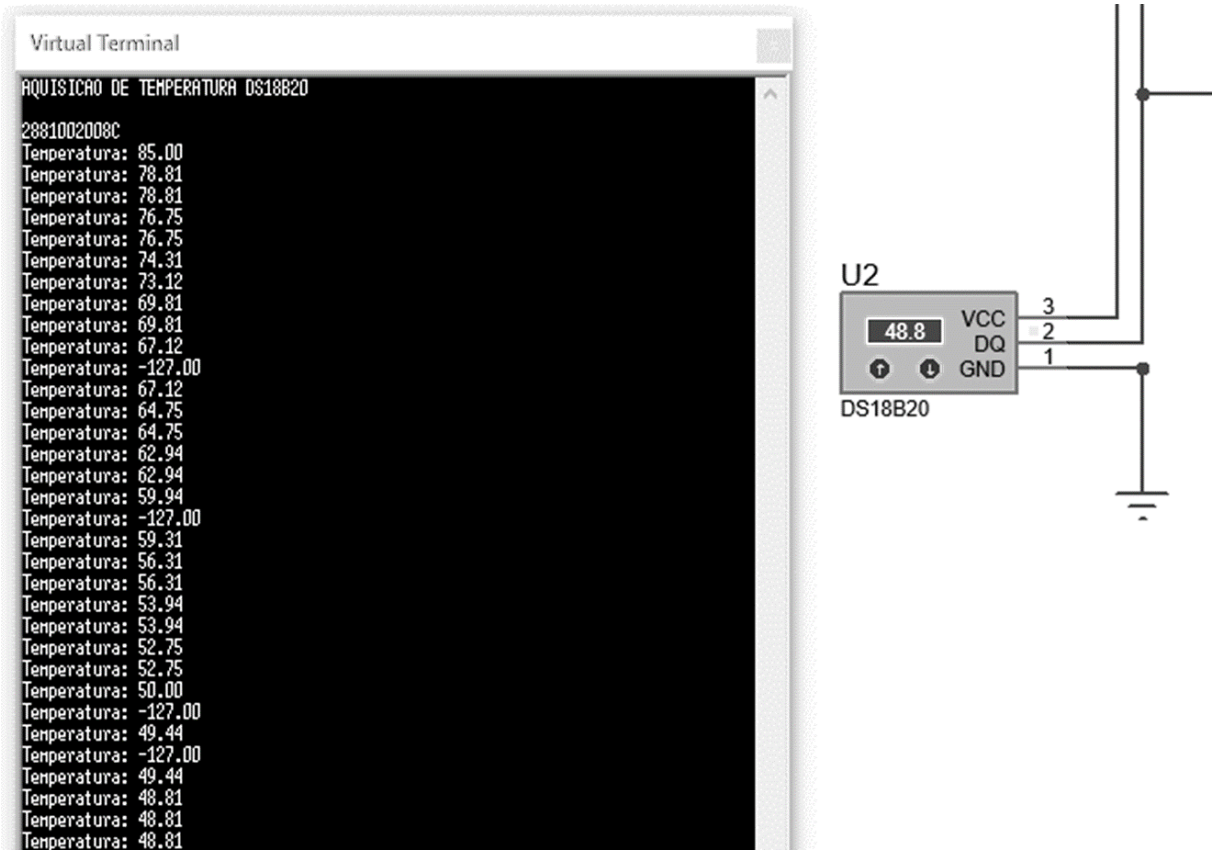


Figura 5.8 – Dados de medição de temperatura a partir da simulação do circuito de medição.

É possível avaliar que a implementação recebe adequadamente o endereço MAC do sensor, e capta os valores de temperatura do sensor. O tempo de aquisição dos dados de temperatura na faixa de precisão determinada é de 750ms. Em alguns momentos o sensor indicou "-127" que é um sinal de erro, devido a questões de simulação. Na Figura 5.9 é possível ver que o log de simulação do Proteus® indicou um problema relacionado ao sensor. À medida que o indicador de uso de CPU durante a simulação aumenta, o número de leituras erradas também cresce.

Figura 5.9 – Log de simulação do Proteus® – mensagem de erro de leitura.

A validação da simulação foi feita na prática através de testes de captação de temperatura com o algoritmo implementado. A Figura 5.10 exibe a montagem do circuito exibido na Figura 5.7, na ligação normal, onde a implementação é composta pela montagem prática do circuito e um recipiente com água quente a aproximadamente 70°C, de forma a avaliar o funcionamento do sensor na faixa de temperatura desejada para a brassagem:

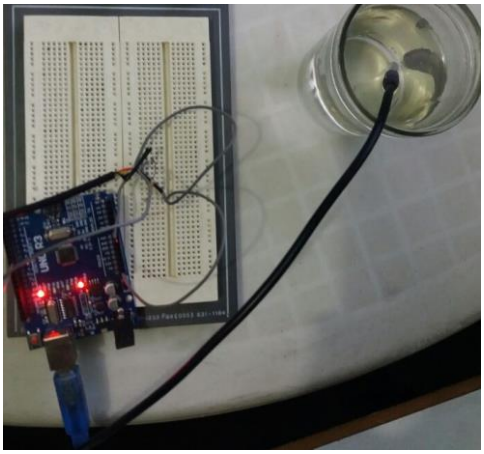


Figura 5.10 – Implementação prática do circuito de medição de temperatura.

A partir da implementação e da comunicação com o PC é possível ver que os resultados práticos são mais consistentes na obtenção de temperatura, a partir do Monitor serial, exibido na Figura 5.11. É possível observar que os erros de detecção de temperatura a partir do sensor não ocorrem no circuito prático.

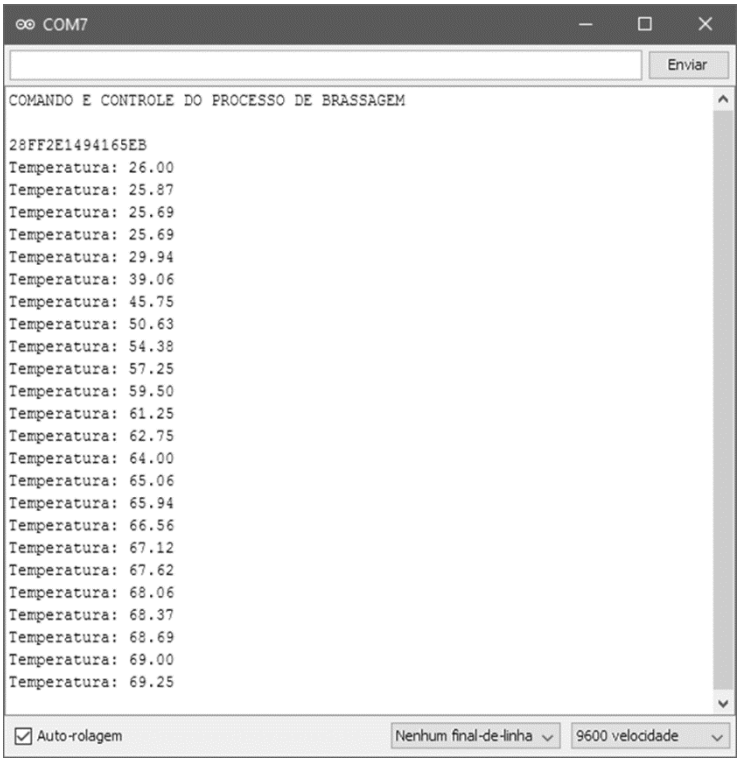


Figura 5.11 – Dados de medição de temperatura a partir da implementação prática do circuito.

Para a construção do gráfico, o tempo de impressão de cada temperatura lida pelo sensor foi alterado para 10ms, de forma a permitir um número maior de pontos, mantendo o tempo de execução do programa abaixo do tempo de medição de temperatura do sensor, de 750ms, como exibido na Figura 5.12.

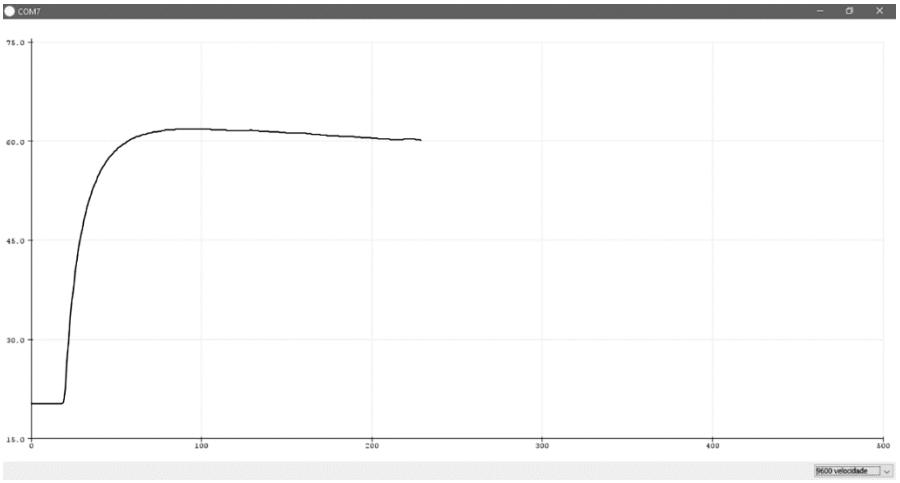


Figura 5.12 – Gráfico obtido dos dados de temperatura pela implementação prática.

5.2.2. Simulação do sensor do modulo *Heated Oven* do Proteus®

Apesar da satisfatória implementação do circuito de medição com o sensor DS18B20, o módulo referente a este sensor não possui uma forma de entrada que permita medir uma temperatura simulada no Proteus®. Devido a limitações do pacote do componente, uma forma alternativa de simulação foi aplicada, para tornar possível a simulação do sistema de brassagem no Proteus®.

Para a simulação do sistema térmico no Proteus® e a captação da temperatura do sensor em função da variação da temperatura do sistema foi utilizado o módulo de forno da biblioteca Proteus®, que permite a simulação de um sistema térmico genérico. O componente OV1, da Figura 5.13(a), é um forno, aquecido a partir de um ebulidor ligado diretamente à rede elétrica. A saída do forno é conectada ao voltímetro, para medir a temperatura de aquecimento atual. Na Figura 5.13(b), são exibidas as propriedades de configuração do forno. O módulo do forno possui um coeficiente de tensão por temperatura (V/°C) configurável, que permite ajustar a proporção de tensão de saída em função da temperatura. Esse coeficiente foi ajustado para 1V/°C de forma que o voltímetro mede exatamente a temperatura de saída do forno.

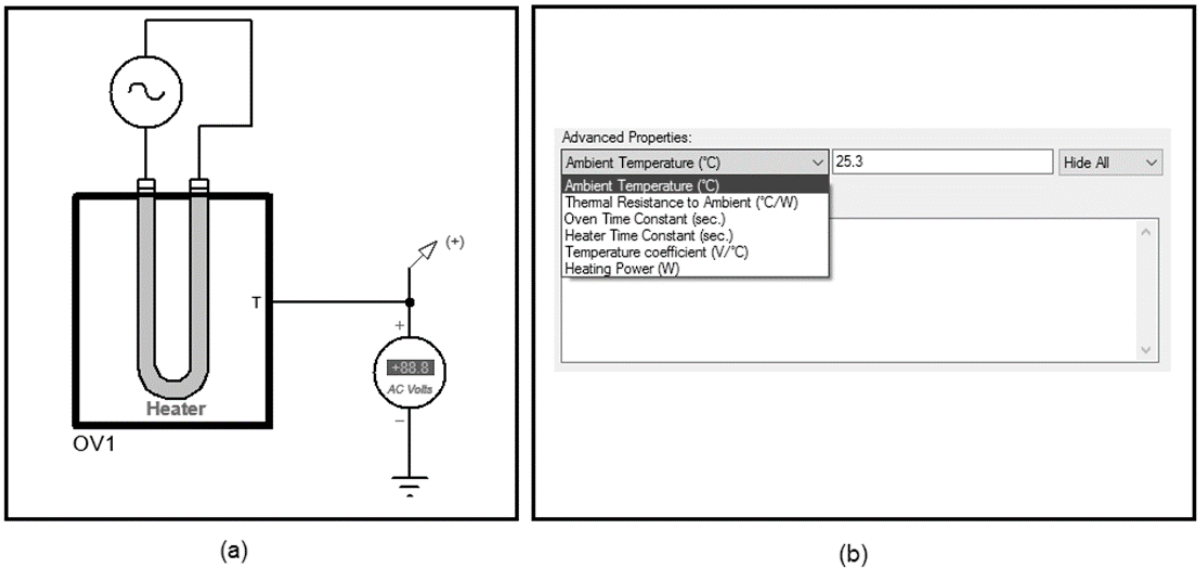
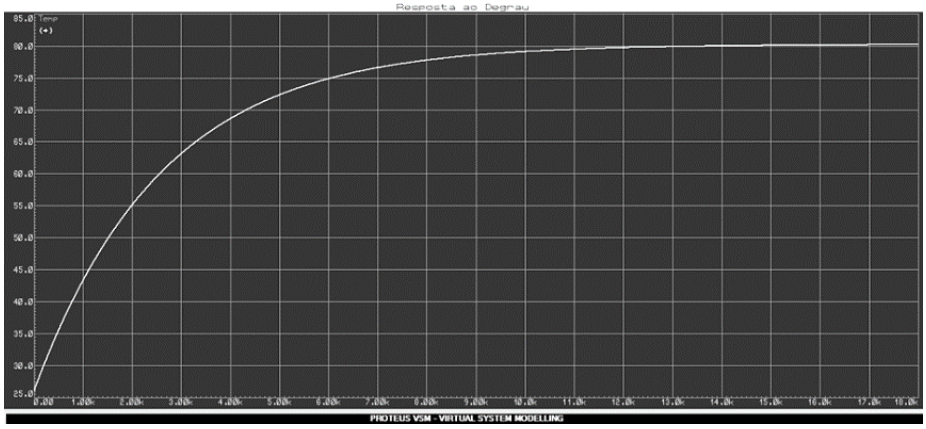


Figura 5.13 – Modulo Forno do Proteus®: (a) Diagrama esquemático do forno. (b) Propriedades de configuração do módulo.

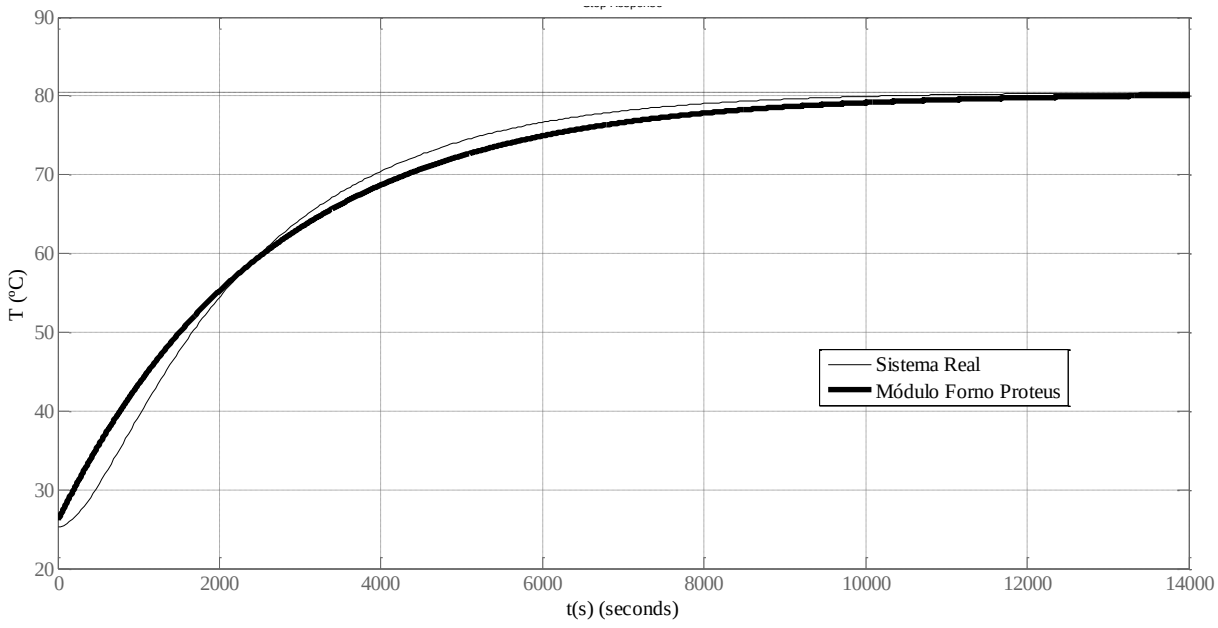
Para configurar o forno, foram utilizados os dados obtidos nos ensaios descritos e os dados obtidos no Matlab®. A temperatura ambiente foi definida como 25,3°C. A resistência térmica para o ambiente foi obtida com a aplicação da Equação (3.5), com o valor de

0,0691K/W. A dinâmica temporal do ebulidor foi considerada cerca de 100 vezes menor do que a dinâmica do sistema térmico, de forma a não afetar o sistema. A potência de entrada foi considerada a de 795,825W, que é a potência medida do terceiro ensaio, para a temperatura em regime permanente de 80,3°C. A resposta de temperatura pelo tempo do sistema foi configurada a partir da Função de Transferência desenvolvida pelo método empírico.

A resposta do forno a um degrau, exibida na Figura 5.14, demonstra o funcionamento da simulação em relação à constante de tempo configurada.



(a)



(b)

Figura 5.14 – Resposta do módulo Forno Proteus® ao degrau: (a) Simulação da resposta no Proteus®. (b) Dados exportados para o Matlab®.

A Figura 5.14 (a) corresponde à simulação da resposta do forno no Proteus®, que permite uma visualização qualitativa da resposta. Após a simulação, os dados foram

exportados para comparação com a Função de Transferência do sistema real. De acordo com a Figura 5.14 (b), os dados da simulação estão bastante próximos da resposta do sistema real, onde a maior discrepância se encontra na faixa de temperatura de 25°C a 50°C, o que não influi na precisão do controle pois, como dito, a principal faixa de atuação do processo de brassagem é de 50°C a 75°C.

Para a implementação do módulo em um circuito com o Arduino foi necessário condicionar o sinal com um divisor de tensão, para ajustar a tensão de 0 a 80,3V para 0 a 5V.

No código do Arduino a leitura de temperatura do módulo foi feita a partir da porta analógica, ajustada para a faixa de temperatura multiplicando-se pelo fator 85/1024, pois a leitura da porta analógica tem 10bits de resolução, apresentando valores de 0 a 1023, e foi configurado para uma faixa de valores de 0 a 85°C, pois o sistema estabiliza por volta de 80,3°C, mas uma pequena margem de segurança foi adicionada, para que não seja aplicado mais do que 5V na porta do Arduino, gerando erro de simulação.

5.3. Função Timer para implementação do *setpoint*

O *setpoint* do projeto não é um sinal estático degrau, mas sim um conjunto de degraus e rampas. Para a implementação das rampas no Arduino a maneira mais simples foi ajustar o sistema para pequenos degraus, que se incrementam a cada instante de tempo, desde que esse tempo seja muito menor que a dinâmica temporal do sistema. Para isso foi selecionado um intervalo de tempo de 15s, onde o *setpoint* é incrementado de 0,25°C, mantendo a taxa de 1°C/min. Foi criada uma função de atualização do *setpoint* para cada faixa de temperatura da brassagem. A Figura 5.15 mostra a atualização do valor monitorado pelo Plotter serial do Arduino:

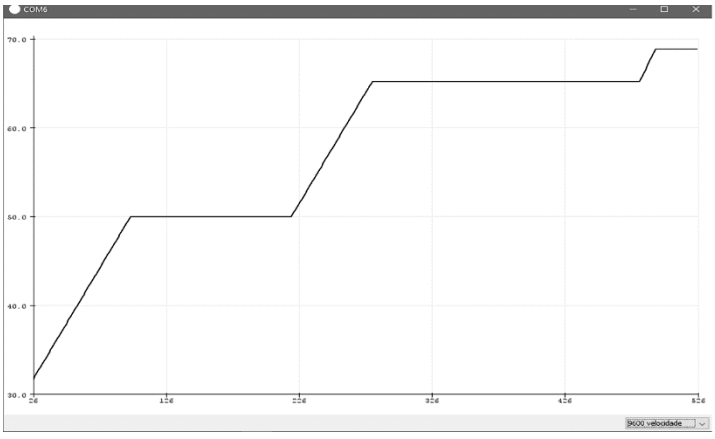


Figura 5.15 – Dados da rotina de atualização do *setpoint* implementada no Arduino.

Para a atualização do valor de *setpoint* foi necessário implementar uma rotina de tempo que não parasse a execução do código, como é o caso da função *delay()* do Arduino. Foram avaliadas duas opções: o uso da função *millis()* e a implementação de uma rotina de interrupção por estouro de timer.

A função *millis()* retorna o número de milissegundos do momento em que o Arduino inicia a execução do código. Esse número estoura após aproximadamente 50 dias, o que ultrapassa o necessário para o projeto.

A função por estouro de *timer* exige inicialmente uma configuração dos registradores do processador do Arduino. Os registradores são TCCR2A (registrador de controle do timer 2, em operação normal), TCCR2B (configuração de escala de divisão de frequência, que permite contar valores maiores, operando em modo *prescaler* de 1/1024 valores), TCNT2 (é o registrador contador, em operação normal) e o TIMSK2 (registrador de interrupção por timer, que habilita a interrupção por *overflow* do timer). O ciclo de máquina do Arduino é 1/16Mhz = 62,5ns (cada instrução leva 62,5ns para ser executada), pois o Arduino trabalha na própria frequência de *clock*. Inicializando o contador de *timer0* para 100 ao início de cada interrupção, o programa conta de 255-100=155 vezes. O estouro é dado pela Equação (5.8)

$$Estouro = timer0 \times prescaler \times ciclo\ de\ maquina \tag{5.8}$$

$$Estouro = (255_{máximo} - 100) \times 1024 \times 62{,}5ns = 9{,}984ms$$

O objetivo da inicialização do contador de *timer0* é obter um valor mais próximo de um múltiplo de 10, o que facilita o tratamento da variável. Porém após a implementação da rotina de tempo a partir do estouro de *timer*, foi constatado que a função apresenta um erro em relação ao tempo real. Isso foi associado ao erro de 0,016ms a cada interrupção, que ao longo de alguns minutos acumula um tempo de erro considerável. A implementação da rotina de tempo a partir da função *millis()* não apresentou erro, e se provou de mais simples implementação.

5.4. Sintonia do controlador

No caso do projeto do sistema em malha fechada, é possível aproximar o efeito do circuito de potência e do atuador como linear, modelados apenas por um ganho proporcional. Essa aproximação é sustentada pelo efeito linear do ebulidor e da relação da Equação (3.5). A

partir dessa relação de linearidade, concluiu-se que a resistência térmica é de 14,4695(°C/W). A relação de potência em função da tensão é exibida na Figura 5.16:

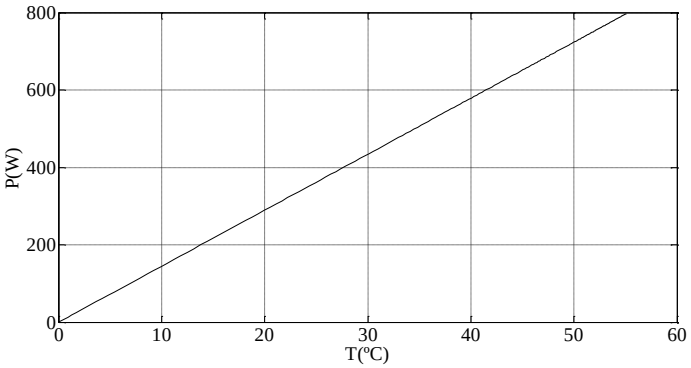


Figura 5.16 – Temperatura x Potência.

Com as considerações de linearidade, é possível montar a função de transferência em malha fechada do sistema de brassagem, tomando como base a Equação (3.10). Com o método de Ziegler-Nichols em malha fechada e a partir da simulação do sistema na ferramenta Simulink, exibido na Figura 5.17, foi possível fazer um ajuste otimizado para um controlador K_p = 3,35, K_i = 0.0015, K_d = 550. Com a ferramenta *Rltool* do Matlab® também foi possível projetar experimentalmente um controlador a partir da manipulação dos polos e zeros do sistema no gráfico de Lugar das Raízes, onde se obteve Kp = 490, Ki = 1 e Kd = 50.000, sendo estes, porém, ganhos exagerados, o que demonstra a inviabilidade do controlador.

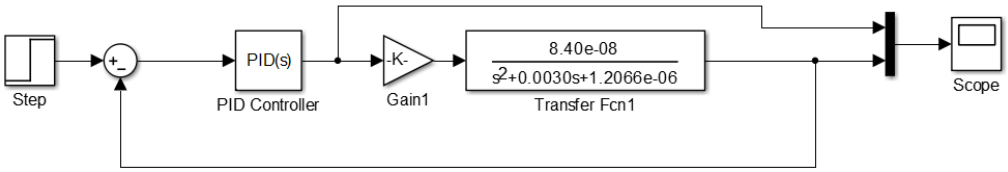


Figura 5.17 – Diagrama de blocos do planta completa.

. Para o controle ajustado pelo método Ziegler-Nichols, a resposta do sistema apresentou um comportamento mais rápido do que a resposta em malha aberta, mas um pequeno sobressinal, efeito indesejado na brassagem. A Figura 5.18 exibe a resposta do sistema à um sinal em degrau em malha aberta, e em malha fechada demonstrando o efeito do controlador. A adição do termo derivativo não traz grandes melhorias na velocidade de subida, mas tem uma pequena atuação no amortecimento do sobressinal, de forma que a adição de uma ação derivativa ao controlador não é essencial, mas pode melhorar o desempenho do sistema.

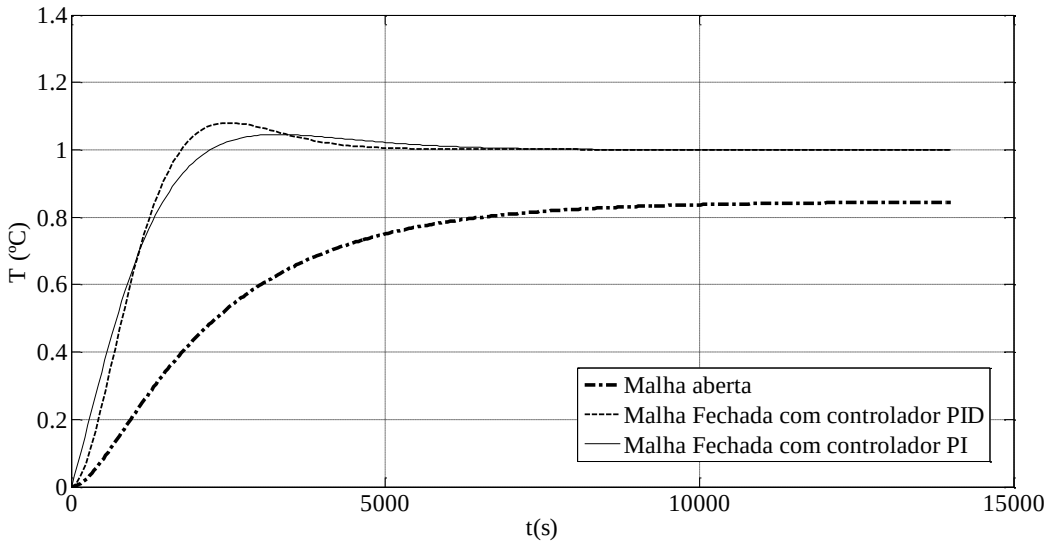


Figura 5.18 – Resposta a um sinal degrau do sistema em malha aberta e em malha fechada com controlador *Rltool* e Ziegler-Nichols.

O mesmo sistema submetido a uma entrada em rampa é exibido na Figura 5.19. O controlador consegue manter o sinal ajustado na mesma taxa de subida, porém exibe um erro de estado estacionário.

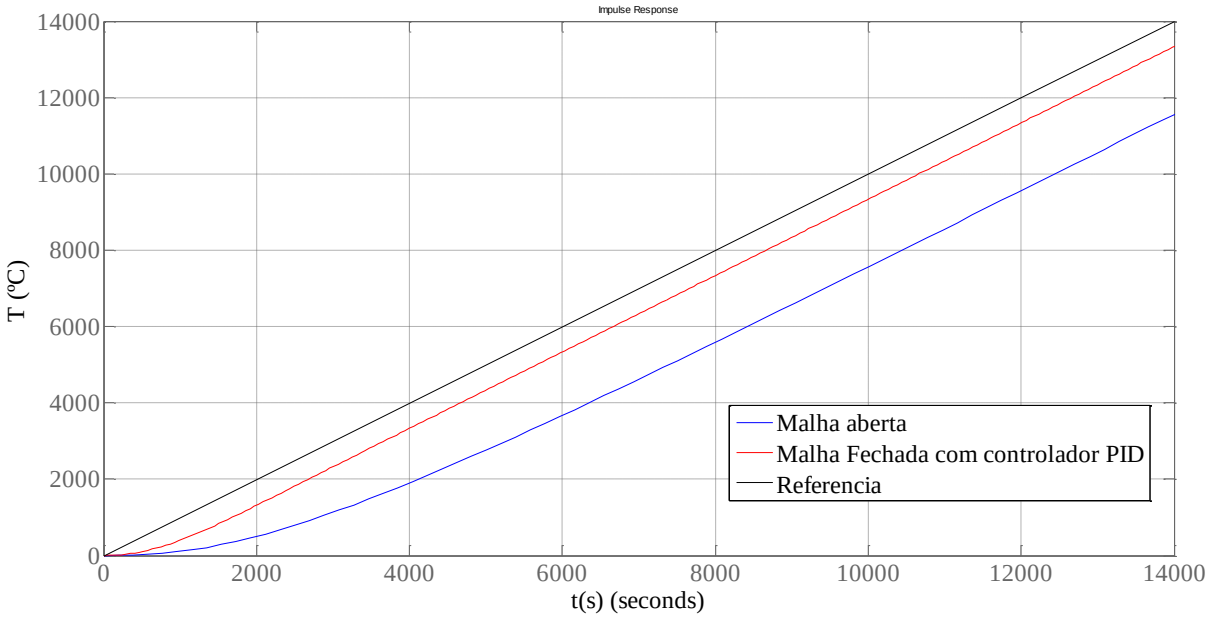


Figura 5.19 – Resposta dos controladores a um sinal rampa.

5.5. Controlador discreto

A partir da simulação do sistema contínuo o primeiro passo para determinar o sistema discreto é a determinação do período de amostragem. Considerando o Teorema da amostragem, foi avaliada a dinâmica temporal da panela com água, cujo tempo de subida é de

aproximadamente 80 minutos. A definição computacional do *setpoint* atualiza seu valor a cada 15 segundos. Já o sensor mede um novo valor de temperatura a cada 750 milissegundos. Considerando o tempo de aquisição de temperatura do sensor como o menor, o tempo de amostragem deve ser menor que a metade deste valor, ou seja, menor que 375 milissegundos. Foi adotado um tempo de 250 milissegundos, considerado satisfatório.

A partir da Equação (5.7), a Função de Transferência em malha fechada discreta é indicada abaixo, para um tempo de amostragem T_1 de 0.25s na Equação (5.9):

$$\frac{T_{out}(z)}{Q_{in}(z)} = \frac{(1{,}892 \cdot 10^{-8})z^2 + (3{,}783 \cdot 10^{-8})z + (1{,}892 \cdot 10^{-8})}{z^2 - (1{,}999)z + 0.9993} \tag{5.9}$$

A forma de discretização utilizada foi através da "Transformação Bilinear de Tustin". Esse tipo de transformação, no entanto, pode apresentar uma distorção, em função da frequência, chamada de *warping*. É o caso do sistema, que como apresentado na Figura 5.20, apresenta esta distorção:

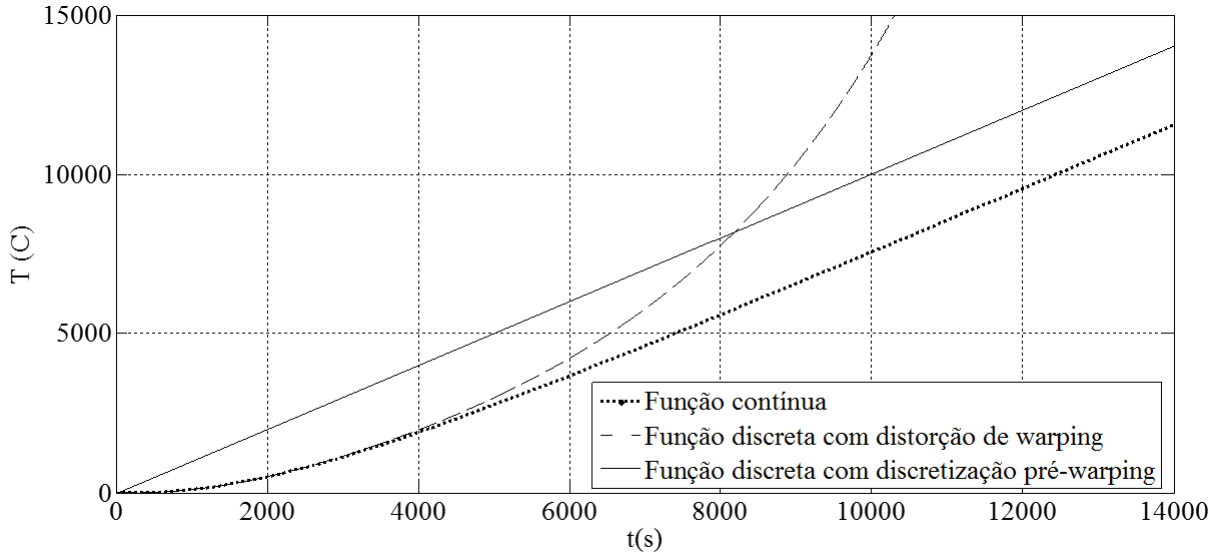


Figura 5.20 – Resposta a rampa da Função de Transferência para os casos: contínuo discreto sem correção e discreto com correção de *prewarping*.

A Função de Transferência discreta com correção é exibida na Equação (5.10):

$$\frac{T_{out}(z)}{Q_{in}(z)} = \frac{(1.094 \cdot 10^{-4})z^2 + (2.189 \cdot 10^{-4})z + (1.0940 \cdot 10^{-4})}{z^2 - (1.943)z + 0.9438} \tag{5.10}$$

A partir do sistema discreto, e após a validação, pode-se analisar o efeito do controlador aplicado às rampas de brassagem, que é o objetivo do trabalho. Com o sinal de referência

aplicado à entrada do sistema como *setpoint*, pode-se analisar a resposta do sistema controlado a partir da Figura 5.21. O sistema apresenta um grande erro estacionário na resposta à rampa. Além disso era esperado que o sistema não conseguisse acompanhar a inclinação do *setpoint* na região da rampa devido à diferença entre a potência disponível (795,825W) e a potência mínima calculada para uma taxa de 1°C/min, calculada no Item 4.1.3 do Capítulo 4 (803,36W). Porém a diferença de 7,5W é relativamente pequena em relação à ordem de grandeza da potência fornecida. Além disso durante as medições, foi aplicada uma tensão de 121,5V_{RMS} ao ebulidor. Em condições ideais a tensão aplicada é de 127V_{RMS}, de forma que a taxa de potência fornecida pode atingir o necessário.

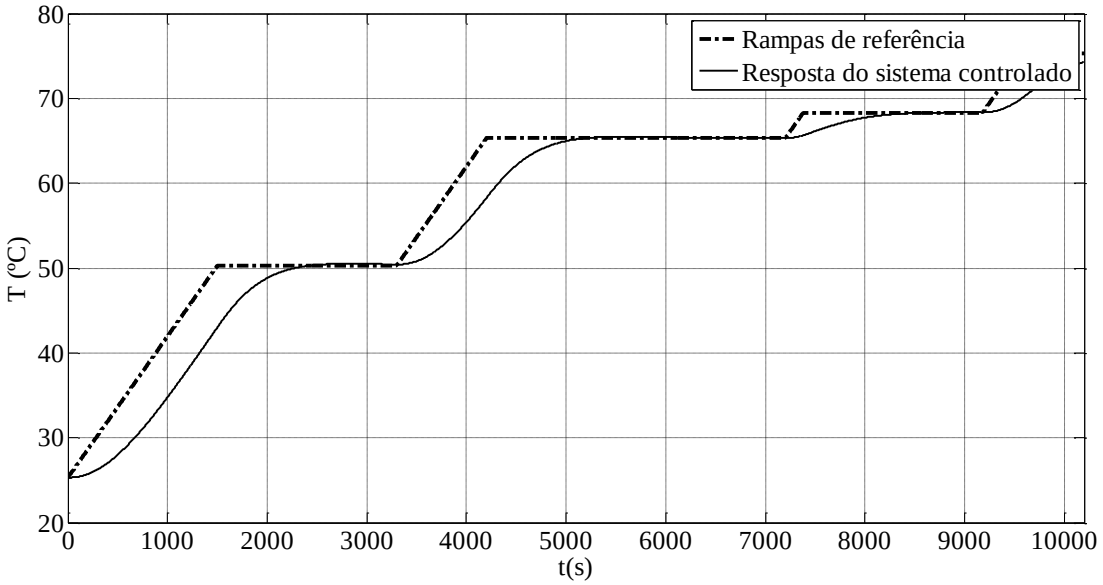


Figura 5.21 – Comparação entre o *setpoint* e o sinal controlado.

5.6. Validação do projeto

Neste momento aplica-se o PID projetado a fim de se validar o circuito final da planta. O funcionamento global do circuito segue o seguinte processo: inicialmente ocorre a medição de temperatura do sistema. Simultaneamente o *setpoint* é atualizado a cada 15 segundos. A partir de então é calculado o sinal de erro e o mesmo é inserido à entrada do PID que determina o tempo de acionamento do TRIAC. O TRIAC injeta potência no ebulidor, que aquece o sistema. O circuito completo é exibido na Figura 5.22:

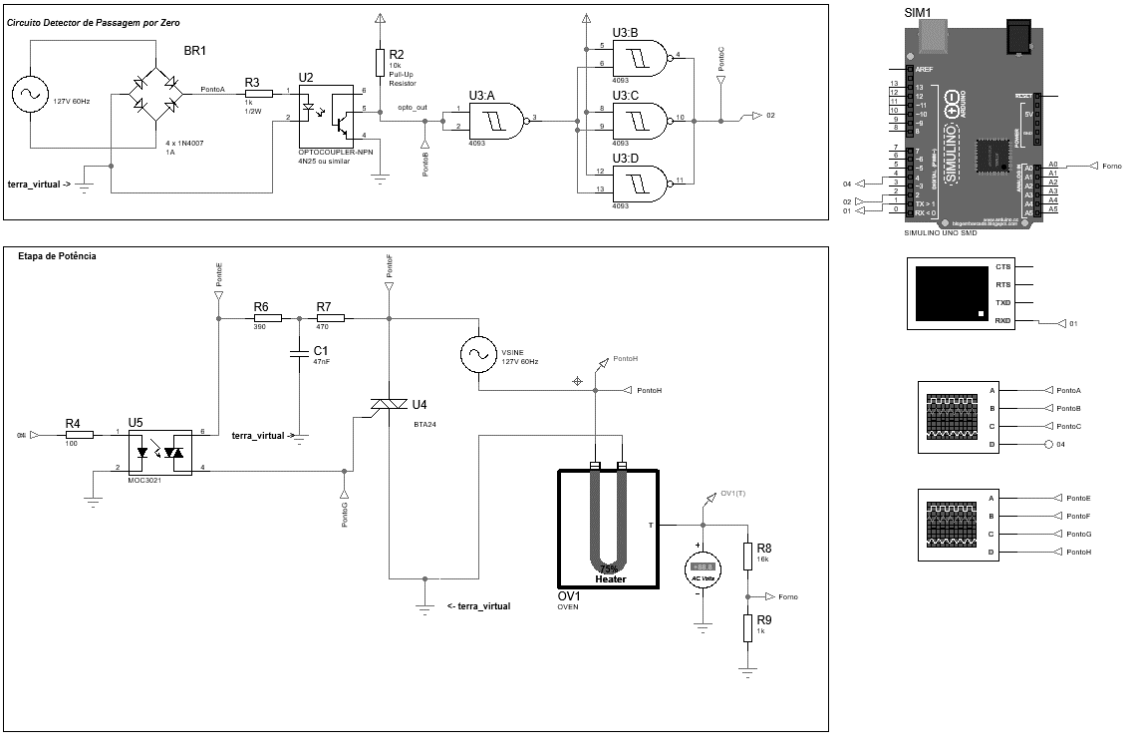


Figura 5.22 – Circuito final.

Na rotina do Arduino o tempo de acionamento do TRIAC é calculado em função da potência necessária a partir do PID. Para que o pulso de acionamento do TRIAC não entre em uma região anterior a um cliclo de onda (atuando no ciclo anterior), foram determinados tempos mínimos e máximos de acionamento, de 300ms e 8200ms, respectivamente. A Equação (4.5) exibe a relação não linear entre a potência e o tempo de acionamento, sendo este de difícil cálculo, justamente devido à dependência do termo senoidal. A solução encontrada foi determinar uma função polinomial que se aproximasse do comportamento da função Tempo x Potência. Com o auxílio do Matlab® se chegou à uma aproximação cúbica, exibida na Figura 5.23.

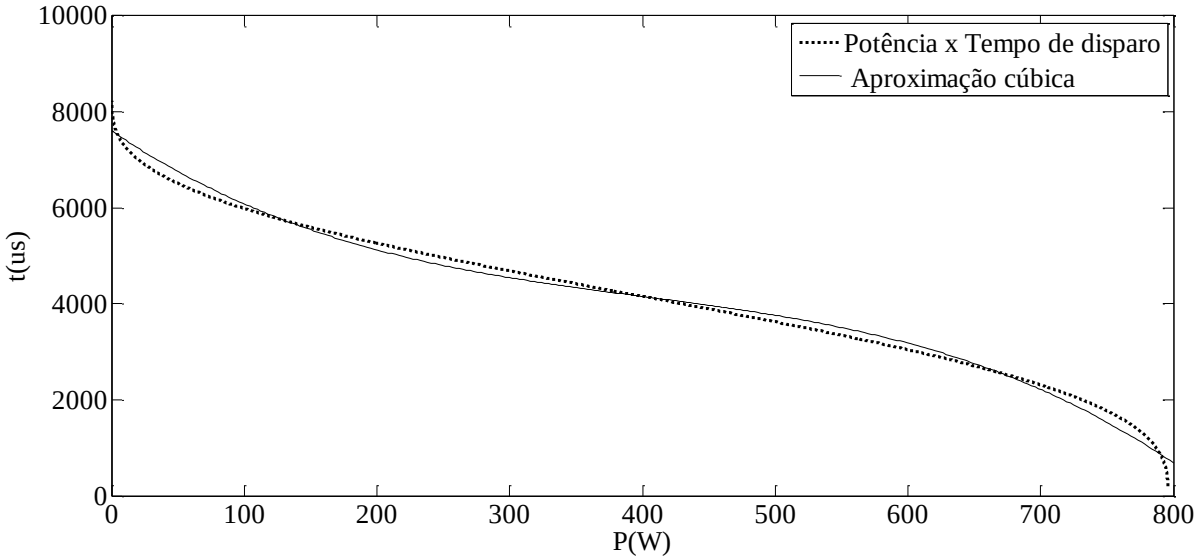


Figura 5.23 – Linearização da curva de relação Potência x Tempo de disparo.

Para avaliar o funcionamento do controlador definiu-se o *setpoint* para os patamares de temperatura desejados de 50°C, 68°C e 75°C, e depois o *setpoint* como as rampas do sistema de brassagem. É válido lembrar que a simulação do sistema não leva em conta ruídos e comportamentos inesperados, como ocorreria no sistema real. O resultado dos três primeiros ensaios para *setpoint* como valor único definido é exibido na Figura 5.24:

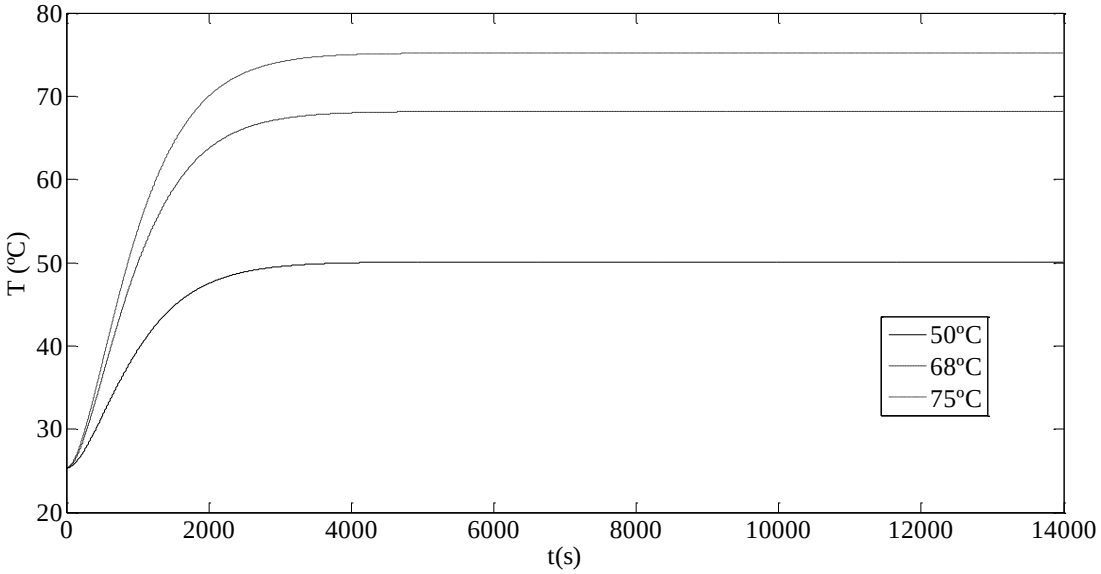


Figura 5.24 – Curvas geradas no Proteus®.

Para os testes em degraus, o sistema consegue atingir o patamar de temperatura desejado, mostrando o funcionamento desejado da parcela PI do controle PID. Durante a implementação do algoritmo, o termo derivativo apresentou erro de cálculo e não foi aplicado. Seu efeito, no entanto, não é essencial, como avaliado a partir da Figura 5.18. O sistema, no

entanto, não apresentou sobressinal visível, devido a aproximações do módulo do Forno. O sistema apresentou, no entanto, um pequeno erro em regime permanente, de aproximadamente 0,3ºC, o que não era esperado devido ao termo integral do controle.

Por fim, a curva de resposta do sistema às rampas de brassagem é exibida na Figura 5.25, onde podemos ter uma avaliação qualitativa de que o sistema segue as curvas com certa fidelidade. Devido a limitações computacionais o sistema não pôde ser simulado em tempo real, sendo necessário diminuir sua dinâmica de tempo na ordem de 10 vezes. Porém, ainda assim é possível constatar que o sistema aplica o controle de forma adequada, se comportando como previsto no projeto do controlador.

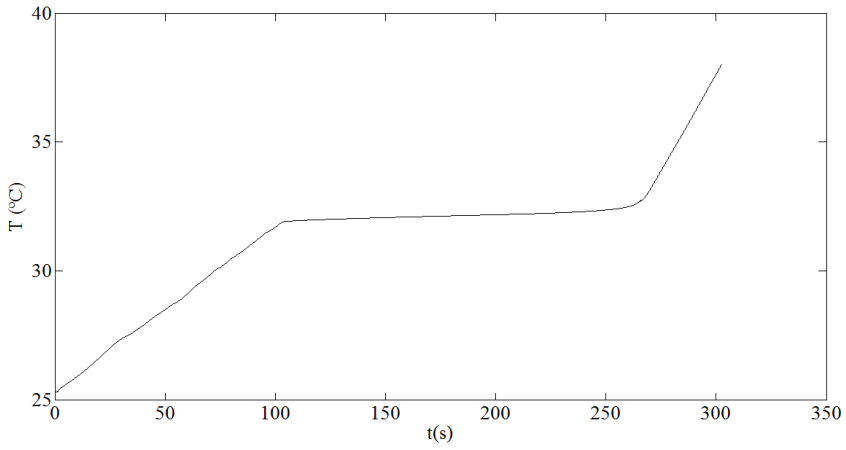


Figura 5.25 – Resposta do sistema final às rampas de brassagem.

Capítulo 6

Conclusão

O trabalho tratou do controle de aquecimento de uma panela com mistura de água e malte de forma específica, focando os ensaios e simulações em parâmetros próprios do sistema.

O trabalho evidenciou a necessidade de um atuador que consiga transmitir mais potência para o sistema, para que se pudesse obter um melhor desempenho. A partir dos cálculos de potência mínima necessária para uma taxa de aquecimento de 1°C/min, foi constatado que o atuador está muito próximo desta potência mínima. O ideal seria que o atuador tivesse um valor maior de potência nominal, apresentando uma maior folga. O sensor DS18B20 apresentou precisão suficiente para a aplicação, e se mostrou de fácil implementação por permitir uma modelagem matemática simples e fornecer a temperatura em Celsius de maneira direta, exigindo apenas tratamento no algoritmo. O Arduino também se mostrou um controlador poderoso, rápido em relação às outras dinâmicas de tempo do sistema e também de fácil programação. O circuito de potência foi o mais complexo exigindo uma eletrônica mais complexa, maior tratamento computacional a partir de interrupção e a necessidade da linearização da relação de Potência pelo Tempo. Todo o trabalho dependeu do software de simulação Proteus® para a demonstração do funcionamento dos circuitos de medição, acionamento e do controle PID. Essa foi a maior limitação do projeto, pois o software exige muito esforço computacional, limitando testes de desempenho. Aliada a essa limitação temos a própria dinâmica temporal do sistema que, por ser de dezenas de minutos, limita a quantidade de testes que podem ser feitos, exigindo alterações nos parâmetros de dinâmica para que se consiga obter dados da simulação.

Em relação à modelagem pode-se constatar que o aumento da ordem do sistema aumentou a complexidade de forma que o método analítico não traduziu adequadamente a realidade, demonstrando que aumentar a ordem do sistema não traz benefícios. A partir do método empírico pôde-se obter um modelo mais adequado.

A execução do trabalho exigiu uma série de aproximações, tanto na modelagem quanto na aplicação dos circuitos. Em cada caso foi avaliado o efeito das respectivas aproximações

no projeto. As aproximações de conversão de valores analógicos para digitais do sensor, aproximações de tempo do Arduino e discretização do controlador não tiveram efeito significativo, sendo improváveis origens de erro no sistema. Porém, as considerações de temperatura uniforme ao longo da panela, de parâmetros concentrados e da não adição de massa ao sistema; da linearização dos limites mínimo e máximo do tempo de acionamento do TRIAC, podem gerar os erros de comportamento do sistema em relação as simulações.

Após a definição adequada da Função de Transferência, os três métodos de sintonia de controladores permitiram uma melhor avaliação do controlador em termos práticos, pois o objetivo do trabalho transcende softwares específicos de controle, visando também a aplicação prática do controlador projetado.

O controle do sistema a entradas que se comportam como degraus apresentou erros pequenos em relação aos respectivos sinais de referência. Isso não era esperado devido à ação do termo integrativo ao controle PI, o que demonstra que o algoritmo implementado, apesar de executar o controle do sistema, ainda pode ser aprimorado. O controlador discreto teve um bom desempenho em relação ao controlador contínuo após a correção de *prewarping*, principalmente devido ao baixo tempo de amostragem em relação ao sistema.

Finalmente, o algoritmo de implementação do PID no microcontrolador não teve a performance esperada. O termo derivativo apresentou um estouro de variável que inviabilizou sua aplicação, limitando ao controle PI. E em relação ao controle em estado estacionário, este demostrou um erro de 0,3ºC, que pode ser considerado pequeno, mas não se justifica em relação à existência do termo integrativo.

As principais melhorias ao trabalho e possibilidade de trabalhos futuros se encontram:

- Na possibilidade de consolidar um modelo matemático que seja fiel ao sistema real;
- No desenvolvimento de um algoritmo mais preciso de implementação do controle PID, visto que esse foi um ponto de falha de desempenho do projeto;
- Na construção real do modelo, que permita a validação das simulações e do projeto desenvolvido neste trabalho.

Além disso, como contribuições secundárias estão:

- Implementação de um shield (placas de circuito que podem ser conectadas ao Arduino, adicionando funcionalidades) de tempo, com um módulo de relógio de alta precisão;

– Implementação de vários sensores em partes diferentes da panela na construção do modelo.

Apêndice A

Receita para o estilo Blond Belge

Blonde Belge - Teste Temperatura

Belgian Blond Ale (18 A)

Type: All Grain
Batch Size: 19,00 l
Boil Size: 24,36 l
Boil Time: 60 min
End of Boil Vol: 22,36 l
Final Bottling Vol: 16,00 l
Fermentation: Ale, 3 est., Blonde

Date: 12 Sep 2015
Brewer: Saturno Brewery
Asst Brewer:
Equipment: Saturno 41 - 40 Litros
Efficiency: 70,00 %
Est Mash Efficiency: 79,2 %
Taste Rating: 30,0

Taste Notes:

Ingredients

Amt	Name	Type	#	%/IBU
4,40 kg	Pilsen Argentino Cargill (4,0 EBC)	Grain	1	84,6 %
0,40 kg	Wheat Malt, Bel (3,9 EBC)	Grain	2	7,7 %
0,10 kg	Munich Malt - 20L (39,4 EBC)	Grain	3	1,9 %
20,00 g	Saaz [3,75 %] - Boil 50,0 min	Hop	4	8,4 IBUs
20,00 g	Styrian Goldings [5,40 %] - Boil 50,0 min	Hop	5	12,1 IBUs
6,00 g	Saaz [3,75 %] - Boil 10,0 min	Hop	6	1,0 IBUs
6,00 g	Styrian Goldings [5,40 %] - Boil 10,0 min	Hop	7	1,4 IBUs
2,0 pkg	SafBrew Ale (DCL/Fermentis #S-33) [23,66 ml]	Yeast	8	-
0,30 kg	Cane (Beet) Sugar (0,0 EBC)	Sugar	9	5,8 %

Gravity, Alcohol Content and Color

Est Original Gravity: 1,062 SG
Est Final Gravity: 1,011 SG
Estimated Alcohol by Vol: 6,8 %
Bitterness: 22,9 IBUs
Est Color: 8,3 EBC

Measured Original Gravity: 1,061 SG
Measured Final Gravity: 0,000 SG
Actual Alcohol by Vol: 0,0 %
Calories: 0,0 kcal/l

Mash Profile

Mash Name: Rampa Padrao Teste Temperatura
Sparge Water: 16,49 l
Sparge Temperature: 75,0 C
Adjust Temp for Equipment: FALSE

Total Grain Weight: 5,20 kg
Grain Temperature: 25,0 C
Tun Temperature: 25,0 C
Mash PH: 5,20

Mash Steps

Name	Description	Step Temperature	Step Time
Protein Rest	Add 18,28 l of water at 52,5 C	50,0 C	30 min
Sacarificação Beta-Amilase	Heat to 65,0 C over 15 min	65,0 C	50 min
Sacarificação Alfa Amilase	Heat to 68,0 C over 3 min	68,0 C	30 min
Mash Out	Heat to 75,0 C over 7 min	75,0 C	10 min

Sparge: Fly sparge with 16,49 l water at 75,0 C

Mash Notes: Two step profile with a protein rest for mashes with unmodified grains or adjuncts. Temperature mash for use when mashing in a brew pot over a heat source such as the stove. Use heat to maintain desired temperature during the mash.

Carbonation and Storage

Carbonation Type: Bottle
Pressure/Weight: 85,63 g
Keg/Bottling Temperature: 21,1 C
Fermentation: Ale, 3 est., Blonde

Volumes of CO2: 2,3
Carbonation Used: Bottle with 85,63 g Table Sugar
Age for: 7,00 days
Storage Temperature: 18,0 C

Notes

Created with BeerSmith

Apêndice B

Dedução analítica detalhada da Função de Transferência

Temos, pela Equação (3.8), da Primeira Lei da Termodinâmica:

$$(q_{in} - q_{out}) = \frac{\partial E}{\partial t} \tag{B.1}$$

Do resistor para a água:

$$(q_{in} - q_{out}) = \frac{\partial E}{\partial t}$$

$$-q_{out\_r} + i^2 R = \frac{\partial E_r}{\partial t}$$

$$-\left[\overline{\boldsymbol{h_{c\_a}}}\boldsymbol{A_r}(\boldsymbol{T_r} - \boldsymbol{T_a})\right] + \boldsymbol{i^2 R} = (\boldsymbol{C_r})\frac{\boldsymbol{dT_r}}{\boldsymbol{dt}} \tag{B.2}$$

Da água para a panela:

$$(q_{in} - q_{out}) = \frac{\partial E}{\partial t}$$

$$q_{in\_a} - q_{out\_a} = \frac{\partial E_a}{\partial t}$$

$$\left[\overline{\boldsymbol{h_{c\_a}}}\boldsymbol{A_r}(\boldsymbol{T_r} - \boldsymbol{T_a})\right] - \left[\overline{\boldsymbol{h_{c\_a}}}\boldsymbol{A_p}(\boldsymbol{T_a} - \boldsymbol{T_p})\right] = (\boldsymbol{C_a})\frac{\boldsymbol{\partial T_a}}{\boldsymbol{\partial t}} \tag{B.3}$$

Da panela para o ar – desprezando a capacitância térmica da panela:

$$\left[\overline{\boldsymbol{h_{c\_agua}}}\boldsymbol{A_{panela}}(\boldsymbol{T_{água}} - \boldsymbol{T_{panela}})\right] = \left[\overline{\boldsymbol{h_{c\_ar}}}\boldsymbol{A_{panela}}(\boldsymbol{T_{panela}} - \boldsymbol{T_{ar}})\right] \tag{B.4}$$

A partir do circuito equivalente da Figura 5.2, são feitas análises de malhas e de nós para a dedução analítica da Função de Transferência. Pelas analise de malhas:

Malha 01:

$$e_{in} = v_{c1}$$

Malha 02:

$$v_{c1} - R_1 i_3 - v_{c2} = 0$$

Malha 03

$$v_{c2} - R_2 i_5 - e_{out} = 0$$

Pela análise de nós:

$$i_5 = \frac{e_{out}}{R_3}$$

$$i_4 = C_2 \frac{dv_{c2}}{dt}$$

$$i_3 = i_4 + i_5 = C_2 \frac{dv_{c2}}{dt} + \frac{e_{out}}{R_3}$$

$$i_2 = C_1 \frac{dv_{c1}}{dt} = C_1 \frac{de_{in}}{dt}$$

$$i_1 = i_2 + i_3 = C_1 \frac{de_{in}}{dt} + C_2 \frac{dv_{c2}}{dt} + \frac{e_{out}}{R_3}$$

Assim, tendo em mãos as equações do circuito, podemos substituir da seguinte forma:

(1) $$e_{in} = v_{c1}$$

(2) $$v_{c1} - R_1 i_3 - v_{c2} = 0$$

(3) Substituindo 1 em 2 $$e_{in} - R_1 i_3 - v_{c2} = 0$$

(4) $$v_{c2} - R_2 i_5 - e_{out} = 0$$
$$v_{c2} = R_2 i_5 + e_{out}$$

(5) Substituindo 4 em 3 $$e_{in} - R_1 i_3 - R_2 i_5 - e_{out} = 0$$

Agora, aplicando as substituições de equações de corrente obtidas na equação (5):

(1) $$e_{in} - R_1 i_3 - R_2 i_5 - e_{out} = 0$$

(2) $$e_{in} - R_1 \left(C_2 \frac{dv_{c2}}{dt} + \frac{e_{out}}{R_3} \right) - R_2 \left(\frac{e_{out}}{R_3} \right) - e_{out} = 0$$

(3) $$e_{in} - \left(R_1 C_2 \frac{dv_{c2}}{dt} \right) - \left(\frac{R_1}{R_3} e_{out} \right) - \left(\frac{R_2}{R_3} e_{out} \right) - e_{out} = 0$$

(4) $$e_{in} - \left(R_1 C_2 \frac{dv_{c2}}{dt}\right) = \left(\frac{R_1}{R_3} e_{out}\right) + \left(\frac{R_2}{R_3} e_{out}\right) + e_{out}$$

(5) $$e_{in} - \left(R_1 C_2 \frac{dv_{c2}}{dt}\right) = e_{out}\left[\left(\frac{R_1}{R_3}\right) + \left(\frac{R_2}{R_3}\right) + 1\right]$$

(6) $$e_{in} - \left[R_1 C_2 \frac{d}{dt}(R_2 i_5 + e_{out})\right] = e_{out}\left(\frac{R_1 + R_2 + R_3}{R_3}\right)$$

(7) $$e_{in} - \left(R_1 C_2 \frac{dR_2 i_5}{dt} + R_1 C_2 \frac{de_{out}}{dt}\right) = e_{out}\left(\frac{R_1 + R_2 + R_3}{R_3}\right)$$

(8) $$e_{in} - \left(R_1 C_2 R_2 \frac{d}{dt}\frac{e_{out}}{R_3} + R_1 C_2 \frac{de_{out}}{dt}\right) = e_{out}\left(\frac{R_1 + R_2 + R_3}{R_3}\right)$$

(9) $$e_{in} - \left(\frac{R_1 C_2 R_2}{R_3}\frac{de_{out}}{dt} + R_1 C_2 \frac{de_{out}}{dt}\right) = e_{out}\left(\frac{R_1 + R_2 + R_3}{R_3}\right)$$

(10) $$e_{in} - \left(\frac{R_1 C_2 R_2}{R_3}\frac{de_{out}}{dt}\right) - \left(R_1 C_2 \frac{de_{out}}{dt}\right) = e_{out}\left(\frac{R_1 + R_2 + R_3}{R_3}\right)$$

(11) $$e_{in} = e_{out}\left(\frac{R_1 + R_2 + R_3}{R_3}\right) + \left(\frac{R_1 C_2 R_2}{R_3} + R_1 C_2\right)\frac{de_{out}}{dt}$$

(12) $$e_{in} = \left(\frac{R_1 + R_2 + R_3}{R_3}\right) e_{out} + \left(\frac{R_1 C_2 R_2 + R_1 C_2 R_3}{R_3}\right)\frac{de_{out}}{dt}$$

Aplicando a Transformada de Laplace

(13) $$E_{in}(s) = \left(\frac{R_1 + R_2 + R_3}{R_3}\right) E_{out}(s) + \left(\frac{R_1 C_2 R_2 + R_1 C_2 R_3}{R_3}\right) s E_{out}(s)$$

(14) $$E_{in}(s) = E_{out}(s)\left[\left(\frac{R_1 + R_2 + R_3}{R_3}\right) + \left(\frac{R_1 C_2 R_2 + R_1 C_2 R_3}{R_3}\right)\boldsymbol{s}\right]$$

(15) $$E_{in}(s) = E_{out}(s)\left[\frac{R_1 + R_2 + R_3 + (R_1 C_2 R_2 + R_1 C_2 R_3)\boldsymbol{s}}{R_3}\right]$$

(16) $$\frac{E_{out}(s)}{E_{in}(s)} = \frac{R_3}{(R_1 C_2 R_2 + R_1 C_2 R_3)\boldsymbol{s} + (R_1 + R_2 + R_3)}$$

(17) $$\frac{E_{out}(s)}{E_{in}(s)} = \frac{\frac{R_3}{(R_1 C_2 R_2 + R_1 C_2 R_3)}}{\boldsymbol{s} + \frac{(R_1 + R_2 + R_3)}{(R_1 C_2 R_2 + R_1 C_2 R_3)}}$$

Apêndice C

Códigos de Programação

Nesta seção são exibidos os códigos desenvolvidos e utilizados ao longo do trabalho

C-1 Código de implementação do sensor DS18B20 em Arduino

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define BARRAMENTO_PINO 7   //entrada de dados do sensor
#define TEMP_PRECISAO 9     //constante que define a precisao do sensor de acordo com o datasheet
float temp=0;
byte addr[8];
OneWire oneWire(BARRAMENTO_PINO);   //inicializa um objeto One-Wire
DallasTemperature sensors(&oneWire); //Faz com que o objeto herde caracteristicas da classe One-Wire

void setup() {
            byte i=0;
            Serial.begin(9600);            //inicializa o monitor Display
            Serial.println("COMANDO E CONTROLE DO PROCESSO DE BRASSAGEM");
            Serial.println();

            sensors.begin();
           //-----------Esse algoritmo guarda o MAC ADDRESS do sensor em uma variavel
            delay(1000);

            //recupera o endereço dos sensores e armazena em "addr"
            if(!oneWire.search(addr)){
              return;
            }


            oneWire.reset();//sinal de reset
            oneWire.select(addr);//envia o endereço do sensor
           //-----------Esse algoritmo imprime o MAC ADDRESS na tela
            for ( i = 0; i < 8; i++) {
               Serial.print(addr[i],HEX);
            }
            Serial.println();
}
void loop() {
                    sensors.requestTemperatures();      //commando que manda um sinal para os sensors ligados no barramento
                                                        //para que estes iniciem a transmissão de dados de temperatura
                    temp = sensors.getTempC(addr);    //capta a temperature do dispositivo com o endereco guardado na variavel
                    Serial.print("Temperatura: ");
                    Serial.println(temp);
                    delay(1000);
}
```

C-2 Código de implementação do modulo Forno Proteus® e comparação com o DS18B20

```
//------------------------------ Header ---------------------
#include <OneWire.h>
#include <DallasTemperature.h>
#define PINO_DS18B20 7
#define PINO_FORNO 0
#define TEMP_PRECISAO 9

float temp1 = 0,
    temp2 = 0,
    temp2_Celsius = 0;
OneWire oneWire(PINO_DS18B20);
DallasTemperature sensors(&oneWire);

//##--------------------------- Codigo ----------------------

void setup() {
        byte i = 0;
        Serial.begin(9600);
        Serial.println("PROGRAMA FINAL TCC - PROJETO DE CONTROLE DE BRASSAGEM");
        Serial.println();
                sensors.begin();
        delay(1000);
        byte addr[8];
        //recupera o endereço dos sensores e armazena em "addr"
        if (!oneWire.search(addr)) {
          return;
        }
        oneWire.reset();//sinal de reset
        oneWire.select(addr);//envia o endereço do sensor
        for ( i = 0; i < 8; i++) {
          Serial.print(addr[i], HEX);
        }
        Serial.println();
}

void loop() {
  // chama sensors.requestTemperatures() para emitir uma solicitação de temperatura
  // global a todos os dispositivos no barramento
  sensors.requestTemperatures();
  temp1 = sensors.getTempCByIndex(0);  //Caso o outro comando não de certo pois não está identificando os endereços
  temp2 = analogRead(PINO_FORNO);
  temp2_Celsius = (((temp2 / 1023) * 85));

  Serial.print("  Temp Forno: ");
  Serial.print(temp2_Celsius);
  Serial.print("      Temp DS18B20: ");
```

```
  Serial.println(temp1);
  delay(500);
}
```

C-3 Rotina de atualização do *setpoint*

```
void setpointUpdate(){
        if(currentTime>0 && currentTime<=15000){            //rampa 01
              setpoint = setpoint + 0.25;
         }
        if(currentTime>15000 && currentTime<=33000){  //degrau 01
             //setpoint = setpoint;
         }
        if(currentTime>33000 && currentTime<=42000){  //rampa 02
             setpoint = setpoint+0.25;
         }
        if(currentTime>42000 && currentTime<=72000){  //degrau 02
              //setpoint = setpoint;
         }
        if(currentTime>72000 && currentTime<=73800){ //rampa 03
              setpoint = setpoint+0.3;
         }
        if(currentTime>73800 && currentTime<=91800){  //degrau 03
             //setpoint = setpoint;
         }
        if(currentTime>91800 && currentTime<=96000){  //rampa 04
             setpoint = setpoint+0.25;
         }
        if(currentTime>96000 && currentTime<=102000){  //degrau 04
              //setpoint = setpoint;
         }
        if(currentTime>102000){  //degrau 04
              setpoint = 25;
              Serial.println("FIM DE PROGRAMA");
         }
}
```

C-4 Rotina de tempo para atualização do *setpoint*

Função *millis()*:

```
long previousTime = 0;
unsigned long currentTime=0;
long interval = 1000;
```

```
void setup() {
  Serial.begin(9600);

}

void loop()
```

```
{
 currentTime = millis();
 if(currentTime - previousTime >= interval) {
  previousTime = currentTime;
  setpointUpdate()
 }
}
```

Interrupção por *timer0*:

```
// --- Variáveis Globais ---
int counter = 0x00;                //variável auxiliar de contagem
unsigned int counter2 = 0x00;
float setpoint = 0;
// --- Rotina de Interrupção ---
ISR(TIMER2_OVF_vect){
        TCNT2=100;              // Reinicializa o registrador do Timer2
                //registrador de 8 bits (0-255) que contem
                //valor de contagem do ATMega.
        counter++;        // variavel auxiliar que permitirar criar
                // multiplos de 10ms
        counter2++;
        //como o registrador estoura a cada 10ms ->  10ms*100 = 1seg

        if(counter2>0 && counter2<=1500){ //rampa
                if(counter == 100){ // counter igual a '100'?
                  counter = 0x00;  //reinicia counter
                  setpointUpdate();
                 }
        }
}
// --- Configurações Iniciais ---
void setup(){
          TCCR2A = 0x00;      //Timer operando em modo normal
          TCCR2B = 0x07;   //Prescaler 1:1024
          TCNT2  = 100;    //10 ms overflow again
          TIMSK2 = 0x01;   //Habilita interrupção do Timer2
}
// --- Loop Infinito ---
void loop()
{
}
```

C-5 Rotina completa do sistema

```
################## DECLARAÇÃO DE VARIAVEIS ################################################
#define PINO_FORNO A0
#define DETEC_0 2
#define GATE   4
#define S1 15000
#define S2 33000
#define S3 42000
#define S4 72000
#define S5 73800
#define S6 91800
#define S7 96000
#define S8 102000


unsigned long currentTime=0;  //Utilizadas na função timer
```

```
unsigned long previousTime=0; //Utilizadas na função timer


int interval = 1500;          //define o intervalo de tempo em que o setpoint será atualizado. No caso 15s

float tempoDisparo = 0,       //Variável para o tempo de disparo do TRIAC
      temp = 0,               //Variavel qeu recebe a temperatura
      erro = 0,               //erro = setpoint - a temperatura do sistema
      temp_bits = 0,          //recebe a temperatura na porta analógica de 0 - 1024
      setpoint =0;

float P=0, I=0, D=0, pid=0;           //acumulam o erro*Kp, erro*Ki..
float kP=3.35, kI=0.0015, kD=550,     //Kp = 3,35, Ki = 0.0015, Kd = 550
      deltaTempo = 0,                 //Variavel que acumula o termo diferencial dt. Na integral multiplica e na derivada divide
      tempAnterior,
      potencia=0;

//##############################################                                                    CODIGO
##################################################

void interrupt()    //funcao executada a cada pulso de passagem por 0. Aciona um sinal
                    //de disparo do GATE e depois de 100ms desliga o pulso
{
   delayMicroseconds(int(tempoDisparo));
   digitalWrite(GATE, HIGH);
   delayMicroseconds(100);
   digitalWrite(GATE, LOW);
}
void setpointUpdate(){    // funçao que atualiza o sinal de rampa. Essa funcao é executada a cada 15s
          if(currentTime>0 && currentTime<=S1){              //rampa 01
                    setpoint = setpoint + 0.25;
           }
          if(currentTime>S1 && currentTime<=S2){  //degrau 01
                    setpoint = 50;
           }
          if(currentTime>S2 && currentTime<=S3){  //rampa 02
                    setpoint = setpoint+0.25;
           }
          if(currentTime>S3 && currentTime<=S4){  //degrau 02
                    setpoint = 65;
           }
          if(currentTime>S4 && currentTime<=S5){ //rampa 03
                    setpoint = setpoint+0.3;
           }
          if(currentTime>S5 && currentTime<=S6){  //degrau 03
                    setpoint = 68;
           }
          if(currentTime>S6 && currentTime<=S7){  //rampa 04
                    setpoint = setpoint+0.25;
           }
```

```
        if(currentTime>S7 && currentTime<=S8){  //degrau 04
              setpoint = 75;
         }
        if(currentTime>S8){  //degrau 04
              setpoint = 25;
         }
}

void setup() {
      Serial.begin(9600);
      Serial.println("COMANDO E CONTROLE DO PROCESSO DE BRASSAGEM - CONTROLE PID");
      Serial.println();
                  //inicializa o setpoint para que a rampa nao comece de 25°C, mas da temperatura ambiente do local
      temp_bits = analogRead(PINO_FORNO);
      temp = ((temp_bits / 1023) * 85);
      setpoint = temp;
      Serial.print("Inicializando Setpoint --- ");
      Serial.print("Setpoint = ");
      Serial.println(setpoint);
      Serial.println();

      pinMode(DETEC_0, INPUT);      //entrada INT0 - essa é a entrada do pulso detector de passagem por 0. Toda vez que a senoide passar por 0
                  //esse pino recebe um pulso e ativa a interrupcao.
      pinMode(GATE, OUTPUT);     //saída digital 4. Saida do pulso acionador do gate do TRIAC, que acontece dependendo do angulo de disparo.
      digitalWrite(GATE, LOW);  //apenas inicializa a saida com um sinal baixo para evitar disparos indesejados
      attachInterrupt(0, interrupt, RISING); //ativa a interrupcao por sinal de passagem por 0
}

void loop() {

      deltaTempo = millis() - currentTime; //calcula a variacao do tempo desde a ultima execucao desta linha ou seja calcula a variacao do tempo delta t. Transforma em segundos
      currentTime = millis();
      if (currentTime - previousTime >= interval) {
            previousTime = currentTime; //menos 2 e uma taxa de correção de erro acumulativo
            //setpointUpdate();
            setpoint=50;
      }
      temp_bits = analogRead(PINO_FORNO);
      temp = ((temp_bits / 1023)* 85);
      //temp = ((analogRead(PINO_FORNO);/ 1023)* 85);  reduz o numero de variaveis e de interacoes do microcontrolador


      //----------   PID  ------------------
      erro = setpoint - temp;
      //Proporcional
      P = erro * kP;
      //Integral
```

```
      I += (erro*kI) * (deltaTempo/1000.0);

//      //Derivada
//      //D = ((tempAnterior - temp)*kD *1000)/deltaTempo;
//      D = (kD *1000)/deltaTempo;
//      tempAnterior = temp;
//      //PID completo
      pid = P+I;

      potencia = pid*14.4695;
      if(potencia<795.75 && potencia>=774.18){
          tempoDisparo = ((-50.934)*potencia +40985)*2;
       }
      if(potencia<774.18 && potencia>=706.07){
          tempoDisparo = ((-12.956)*potencia +11458)*2;
       }
      if(potencia<706.07 && potencia>=389.83){
          tempoDisparo = ((-6.1479)*potencia +6631)*2;
       }
      if(potencia<389.83 && potencia>=66.605){
           tempoDisparo = ((-6.3534)*potencia +6694)*2;
       }
      if(potencia<66.605 && potencia>=8.8551){
           tempoDisparo = ((-17.464)*potencia +7382.3)*2;
       }
      if(potencia<8.8551 && potencia>=0){
           tempoDisparo = ((-96.613)*potencia +8020.3)*2;
       }

      if(tempoDisparo < 300){
            tempoDisparo = 300;
      }
      if(tempoDisparo > 16533){
            tempoDisparo = 16533;
      }
 }
```

C-6 Rotina em Matlab® – Função de Transferência analítica e empírica

```
clear all
close all
clc

% %LINEARIDADE DO RESISTOR
%
% % NO ensaio 1 foi estabilizada uma temperatura constante, e variada
% % a tensão atraves de um varivolt. A corrente variou pela lei de Ohm, e o
% % resultado representa a curva de resistencia
%
% % No ensaio 2 a tensão e a corrente foram mantidas constantes. Foram
% % medidos os dados de temperatura variando com o tempo.
%
% % %% Para T =50
% V1 = [1917253341.14957657375 80];
% I1 = [0.09 0.49 0.94 1.37 1.80 2.24 2.75 3.12 3.56 3.97 4.07 4.34];
% figure(1);grid;
% plot(I1,V1)
% hold on;
% set(gca, 'FontName', 'Times New Roman','FontSize',18);
% title('Curva de linearidade do Ebulidor - Tensão V(V) x Corrente I(A)','FontSize',18,'FontName', 'Times New Roman');
```

```
%
xlabel('I(A)','FontSize',18,'FontName',
'Times New Roman'); % x-axis label
%
ylabel('V(V)','FontSize',18,'FontName',
'Times New Roman');  % y-axis label
% axis([0 6.4 0 130]);
%
% % Para T =60
% V2 = [1917253341.1495765737581
89 97 105 113 119.4];
% I2 = [0.061 0.494 0.934 1.375 1.81
2.256 2.69 3.12 3.526 3.91 4.070 4.41
4.83 5.25 5.68 6.1 6.42];
% plot(I2,V2+3,'r')
% xlabel('I(A)') % x-axis label
% ylabel('V(V)') % y-axis label
%
% % Para T =68-69
% V3 = [1917253341.1495765737581
89 97 105 113 119.4];
% I3 = [0.055 0.491 0.929 1.366 1.80
2.23 2.675 3.115 3.55 3.91 4.09 4.40
4.84 5.26 5.67 6.10 6.45];
% plot(I3,V3-3,'g'); grid;
% leg1=legend('50°C','60°C','68°');
% xlabel('I(A)') % x-axis label
% ylabel('V(V)') % y-axis label
% R1 = V1./I1;
% M1 = median(R1)
% R2 = V2./I2;
% M2 = median(R2)
% R3 = V3./I3;
% M3 = median(R3)


%% DEGRAUS DE TEMPERATURA

%o objetivo dessa segunda etapa não
foi a avaliação do resistor mas a
%validação do modelo da FT. Assim o
objetivo foi obter a resposta do
%sistema à um sinal degrau. Não
utilizamos a rampa pois isso exigiria 4
%variaveis não fixas, o que tornaria o
experimento impraticavel. Com a
%resposta ao degrau podemos chegar
ao comportamento do sistema e mesmo
%obter a dinamica temporal do mesmo.

%fixamos a tensão e a corrente,
obtendo apenas os dados de
Temperatura e
%tempo.

%V=80 e I=4.376 P=350.08W
Tamb=24.3C
Temp1 = [74.2 73.4 73.7 73.4 73 73.1
73 73.1 72.7 72.3 72.3 71.9 72.0 71.8
71.2 71.6 71.2 70.9 70.7 70.5 70.35
70.2 70.1 69.7 69.7 69.2 69.7 69.2 69.2
68.9 68.9 68.9 68.8 69.0 68.8 68.4 68.4
68.0 68.0 68.0 68.0 68.0];
t1_1 = [0 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41];
t1 = 60.*t1_1;

figure(2)
hold on;grid on;
plot(t1_1,Temp1,'k');
set(gca,'FontName', 'Times New
Roman','FontSize',18);
title('Dados coletados de Temperatura
(°C) x tempo (s)');
xlabel('t(min)') % x-axis label
ylabel('T(°C)') % y-axis label


% V=50 e I=2.74
Temp2 = [69.2 69 68.3 67.9 67.9 67.5
67.1 66.9 66.7 65.9 65.9 65.5 65.5 65
65 64.6 64.2 64.2 63.4 64.2 63.4 63.4
63.1 63.1 62.8 62.6 61.8 62.2 61.8 61.5
61.4 61.7 61.2 61. 60.9 60.8 60.5 60.5
60.1 60.1 59.9 59.7 59.7 59.2 59.2 59.3
58.6 58.2 58.6 58.2 58.2 57.9 56.1 55.9
55.4 54.9 55.3 54.5];
t2_2 = [0 1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47 48 49
50 51 75 100 110 115 120 128];
t2 = 60.*t2_2;
plot(t2_2,Temp2,'k')

% V=119,2 e I=6,43
Temp3 = [24.3 26.3 26.8 27.7 28 28.8
29.9 30.9 31.9 32.9 33.9 34.5 35.4 35.9
37.4 37.9 38.9 39.9 40.9 41.3 42.3 43.9
45.1 46.5 46.1 46.5 48.4 48.9 49.9 50.3
50.4 51.8 52 53.7 53.5 53.9 55 55.6
56.4 57.4 58.2 58.8 59.3 59.8 60.5 61
61.5 61.7 62.1 62.9 63.3 63.8 64 64.5
71.5 71.9 72.2 72.2 72.2 72.6 75 75.5
77.5 77.3 77.3 76.9 77.3 77 77.3 77.8
77.3 77 77.3 77.8 77.3];
t3_3 = [0 1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47 48 49
50 51 52 53 66 67 68 69 70 71 82 91
95 99 102 103 104 105 106 107 108
109 110 111 112];
t3 = 60*t3_3;
plot(t3_3,Temp3,'k')

% V=121,5 e I=6,55
Temp4 = [25.4 25.4 26.5 27 28.5 29.5
30.5 30.9 32.5 33.4 34.5 35.3 35.8 36.8
37.8 38.8 39.8 40.8 41.7 42.7 43.1 44
44.9 45.3 46.3 47.8 48.2 49.1 49.9 50.5
51.5 52 52.9 53.4 53.8 55.3 56.3 56.3
57.2 58.1 58.6 59.5 60.2 60.3 60.8 61.5
61.9 62.4 63.1 63.5 64.3 64.5 65.2 65.4
68.2 69.7 70.1 71.6 73.8 75.4 75.6 76.0
76.85 76.7 78.8 80.4 79.2];
t4_4 = [0 1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47 48 49
50 51 52 53 58 62 65 70 79 84 89 94
99 104 112 120 126];
t4 = 60*t4_4;
plot(t4_4,Temp4,'k')


%%
x=0:1:6720;
load('y2.mat');
load('y3.mat');
load('y4.mat');
figure(3)
plot(x,y2,'-c',x,y3,'c',x,y4,'c');grid;
%plot(x,y2,'-
c',x,y3,'b','LineWidth',1.5);grid;
hold on;
```

```
set(gca,'FontName', 'Times New
Roman','FontSize',18);
title('Interpolacao e Aproximacao');
xlabel('t(min)') % x-axis label
ylabel('T(°C)') % y-axis label
%plot(x,y4);grid;

%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
tamanho_vetor_saida = length(y2);

dy = 55.68-68.82;
M = -137;
K2 = dy/M;

y_tempo1 = y2(1)+dy*0.632;
y_tempo2 = y2(1)+dy*0.283;

%%%%% Entrar com os valores de
tempo referentes a y_tempo
tau_a = 2117;
tau_b = 825;
tau2 = 1.5*(tau_a-tau_b);
td2 = tau_a-tau2;

fprintf('K = %0.3d\n', K2);
fprintf('td = %0.3d\n', td2);
fprintf('tau  = %0.3d\n', tau2);

num = [0 K2];
den = [tau2 1];
FT2 = tf(num,den)
step((M*FT2)+68.82,'y');
set(gca,'FontName', 'Times New
Roman','FontSize',18);
title('Interpolacao e Aproximacao');
xlabel('t(min)') % x-axis label
ylabel('T(°C)') % y-axis label
%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
tamanho_vetor_saida = length(y3);

dy = y3(tamanho_vetor_saida)-y3(1);
M = 700;
K3 = dy/M;

y_tempo1 = y3(1)+dy*0.632;
y_tempo2 = y3(1)+dy*0.283;

%%%%% Entrar com os valores de
tempo referentes a y_tempo
tau_a = 2378;
tau_b = 929;
tau3 = 1.5*(tau_a-tau_b);
td3 = tau_a-tau3;

fprintf('K = %0.3d\n', K3);
fprintf('td = %0.3d\n', td3);
fprintf('tau  = %0.3d\n', tau3);

num = [0 K3];
den = [tau3 1];
FT3 = tf(num,den)
num = [0 1.1459e-07];
den = [1 0.00301 1.4286e-06];
FT32 = tf(num,den)
%step((M*FT3)+24.3,'y',(M*FT32)+2
4.3,'b');

%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
tamanho_vetor_saida = length(y4);

dy = y4(tamanho_vetor_saida)-y4(1);
M = 795.825;
K4 = dy/M;

y_tempo1 = y4(1)+dy*0.632;
y_tempo2 = y4(1)+dy*0.283;
%%
%%%%% Entrar com os valores de
tempo referentes a y_tempo
tau_a = 2420;
tau_b = 901;
tau4 = 1.5*(tau_a-tau_b);
td4 = tau_a-tau4;

fprintf('K = %0.3d\n', K4);
fprintf('td = %0.3d\n', td4);
fprintf('tau  = %0.3d\n', tau4);
num = [0 K4];
den = [tau4 1];
FT4 = tf(num,den)
%%
num = [0 0.06957];
den = [2618.2 0.985];
FT43 = tf(num,den)

num = [0 0.0706];
den = [2579 1];
FT44 = tf(num,den)
%[g2,tg2]=step((M*FT43)+25.4);

num = [0 8.4790e-08];
den = [1 0.0030 1.2066e-06];
FT42 = tf(num,den)
[g1,tg1]=step((M*FT42)+25.4);
step((M*FT42)+25.4,'k',(M*FT43)+25.
4,'r',(M*FT44)+25.4,'y');
step(((-137)*FT42)+68.82,'k',((-
137)*FT43)+68.82,'r');
set(gca,'FontName', 'Times New
Roman','FontSize',18);
title('Interpolacao e Aproximacao');
xlabel('t(min)') % x-axis label
ylabel('T(°C)') % y-axis label

%plot(tg1,g1,'b',tg2,g2,'r'); grid on;
figure(4)
step((M*FT43)+25.4,(M*FT44)+25.4);
hold on

clear all
close all
clc

degrau = tf(1,[1 0]);
rampa = tf(1,[1 0 0]);
Ts = 0.25;
rampad = c2d(rampa,Ts,'Tustin');
%rampad = tf([Ts 0],[1 -2 1],Ts);

%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%
num = [0 8.37e-08];
```

```
den = [1 0.0030 1.2066e-06];
FT = tf(num,den);

K = 14.4695;

num = [50000 490 1];
den = [1 0];
C1 = tf(num,den);
C2 = pid(3.35,0.0015,550);

G=feedback(K*FT,1);
G1 = feedback(C1*K*FT,1);
G2 = feedback(C2*K*FT,1);

%opt = c2dOptions('Method','tustin','PrewarpFrequency',3.4);
% [ft,t]=step((K*FT),14000);
% plot(t,ft,'-.k','LineWidth',3); hold on; grid on;
% [c,t]=step(G1,14000);
% plot(t,c,'--k','LineWidth',2); hold on;
% [c,t]=step(G3,14000);
% plot(t,c,'k')
% [r,t1]=impulse((G3)*rampa);
% plot(t1,r,'*r','LineWidth',0.5); hold on; grid on;
% [ft,t2]=impulse((K*FT)*rampa);
% plot(t2,ft,'-.k','LineWidth',2); hold on; grid on;
% [c,t3]=impulse((G3)*rampa);
% plot(t3,c,'k');
%axis([0 14000 0 1.2]);
% set(gca, 'FontName', 'Times New Roman','FontSize',20);
%
xlabel('t(s)','FontSize',20,'FontName', 'Times New Roman')
% ylabel('T (°C)','FontSize',20,'FontName', 'Times New Roman');
% legend('Malha aberta','Malha Fechada com controlador PID','Malha Fechada com controlador PI')
%% DISCRETIZAÇÃO DOS SISTEMAS
F = K*FT;
opt = c2dOptions('Method','tustin','PrewarpFrequency',12.5);
FTd = c2d(F,Ts,opt);
Cd1 = c2d(C1,Ts,opt);
%with Kp = 3, Ki = 0.0017, Kd = 600, Tf = 0.00236, Ts = 0.3
Cd2 = c2d(C2,Ts,opt);


% FTd = c2d(F,Ts,'Tustin');
% Cd1 = c2d(C1,Ts,'Tustin');
% %with Kp = 3, Ki = 0.0017, Kd = 600, Tf = 0.00236, Ts = 0.3
% Cd2 = c2d(C2,Ts,'Tustin');
%
Gd=feedback(FTd,1);
Gd1 = feedback(Cd1*FTd,1);
Gd2 = feedback(Cd2*FTd,1);
%
%
% [r,t1]=impulse(rampa*K*FT,14000);
% [f,t2]=impulse(rampad*FTd,14000);
% figure(1)
% plot(t1,r,'-.k','LineWidth',2.5); hold on
% plot(t2,f,'--k','LineWidth',0.5); grid on;hold on
%
% opt = c2dOptions('Method','tustin','PrewarpFrequency',12.5);
% FTd = c2d(F,Ts,opt);
% Cd1 = c2d(C1,Ts,opt);
% %with Kp = 3, Ki = 0.0017, Kd = 600, Tf = 0.00236, Ts = 0.3
% Cd2 = c2d(C2,Ts,opt);
%
% [f,t2]=impulse(rampad*FTd,14000);
% plot(t2,f,'k'); grid on;hold on
% axis([0 14000 0 15000]);
% set(gca, 'FontName', 'Times New Roman','FontSize',20);
%
xlabel('t(s)','FontSize',20,'FontName', 'Times New Roman')
% ylabel('T (C)','FontSize',20,'FontName', 'Times New Roman');
% legend('Função contínua','Função discreta com distorção de warping','Função discreta com discretização prewarping')
%%
figure(1)
subplot(2,2,1)
step(K*FT,'b',G,'r'); grid on;
set(gca, 'FontName', 'Times New Roman','FontSize',14);
xlabel('t(s)','FontSize',14,'FontName', 'Times New Roman')
ylabel('T (C)','FontSize',14,'FontName', 'Times New Roman');

subplot(2,2,2)
step(K*FT,'b',G1,'g',G2,'k'); grid on;
set(gca, 'FontName', 'Times New Roman','FontSize',14);
xlabel('t(s)','FontSize',14,'FontName', 'Times New Roman')
ylabel('T (C)','FontSize',14,'FontName', 'Times New Roman');


subplot(2,2,3)
impulse(rampa*K*FT,'b',rampa*G,'r',rampa,'c'); grid on;
set(gca, 'FontName', 'Times New Roman','FontSize',14);
xlabel('t(s)','FontSize',14,'FontName', 'Times New Roman')
ylabel('T (C)','FontSize',14,'FontName', 'Times New Roman');

subplot(2,2,4)
impulse(rampa*K*FT,'b',rampa*G1,'g',rampa*G2,'k',rampa,'r'); grid on;
set(gca, 'FontName', 'Times New Roman','FontSize',14);
xlabel('t(s)','FontSize',14,'FontName', 'Times New Roman')
ylabel('T (C)','FontSize',14,'FontName', 'Times New Roman');


%%
figure(2)
subplot(2,2,1)
step(FTd,'b',Gd,'r'); grid on;
set(gca, 'FontName', 'Times New Roman','FontSize',14);
xlabel('t(s)','FontSize',14,'FontName', 'Times New Roman')
```

```
ylabel('T (C)','FontSize',14,'FontName',
'Times New Roman');

subplot(2,2,2)
step(FTd,'b',Gd1,'g',Gd2,'k'); grid on;
set(gca, 'FontName', 'Times New
Roman','FontSize',14);
xlabel('t(s)','FontSize',14,'FontName',
'Times New Roman')
ylabel('T (C)','FontSize',14,'FontName',
'Times New Roman');



subplot(2,2,3)
impulse(rampad*FTd,'b',rampad*Gd,'r',
rampad,'c'); grid on;
set(gca, 'FontName', 'Times New
Roman','FontSize',14);
xlabel('t(s)','FontSize',14,'FontName',
'Times New Roman')
ylabel('T (C)','FontSize',14,'FontName',
'Times New Roman');



subplot(2,2,4)
impulse(rampad*FTd,'b',rampad*Gd1,'
g',rampad*Gd2,'k'); grid on;
set(gca, 'FontName', 'Times New
Roman','FontSize',14);
xlabel('t(s)','FontSize',14,'FontName',
'Times New Roman')
ylabel('T (C)','FontSize',14,'FontName',
'Times New Roman');
xlim([0 14000]);

clear all
close all
clc
%%
rampa = tf(1,[1 0 0]);
t1 = linspace(0,1500,6000);
t2 = linspace(1500, 3300,7200);
t3 = linspace(3300, 4200,3600);
t4 = linspace(4200,7200,12000);
t5 = linspace(7200,7380,720);
t6 = linspace(7380,9180,7200);
t7 = linspace(9180,9600,1680);
t8 = linspace(9600,10200,2400);

y1 = linspace(0,25,6000);
y2 = linspace(25,25,7200);
y3 = linspace(25,40,3600);
y4 = linspace(40,40,12000);
y5 = linspace(40,43,720);
y6 = linspace(43,43,7200);
y7 = linspace(43,50,1680);
y8 = linspace(50,50,2400);



% t1 = linspace(0,1500,1500);
% t2 = linspace(1500, 3300,1800);
% t3 = linspace(3300, 4200,900);
% t4 = linspace(4200,7200,3000);
% t5 = linspace(7200,7380,180);
% t6 = linspace(7380,9180,1800);
% t7 = linspace(9180,9600,420);
% t8 = linspace(9600,10200,600);
%
% y1 = linspace(25,50,1500);
% y2 = linspace(50,50,1800);
% y3 = linspace(50,65,900);
% y4 = linspace(65,65,3000);
% y5 = linspace(65,68,180);
% y6 = linspace(68,68,1800);
% y7 = linspace(68,75,420);
% y8 = linspace(75,75,600);

% plot(t1,y1); grid on;
% hold on
% plot(t2,y2);
% plot(t3,y3);
% plot(t4,y4);
% plot(t5,y5);
% plot(t6,y6);
% plot(t7,y7);
% plot(t8,y8);

t = horzcat(t1,t2,t3,t4,t5,t6,t7,t8);
y = horzcat(y1,y2,y3,y4,y5,y6,y7,y8);
plot(t,y+25.3,'-.k','LineWidth',3);    grid
on;

hold on;
%%
% num = [1];
% den = [0.1 1];
% FT = tf(num,den);
% t= linspace(0,11700,11700);
% [g,T]=lsim(FT,y,t);
% plot(T,g,'b'); grid on;
num = [0 8.40e-08];
den = [1 0.0030 1.2066e-06];
FT = tf(num,den);

C1 = pid(3.35,0.0015,550);
K = 15.4695;
G=feedback(K*FT,1);
G1 = feedback(C1*K*FT,1);

F = K*FT;
Ts = 0.25;
opt =
c2dOptions('Method','tustin','PrewarpFr
equency',8);
FTd = c2d(F,Ts,'Tustin');
Cd1 = c2d(C1,Ts,'Tustin');

Gd=feedback(FTd,1);
Gd1 = feedback(Cd1*FTd,1);


%%

% t = linspace(0,10200,40800);
% [g,T]=lsim(G1,y,t);
% %T(40801)=[];
% plot(T,g+25.3,'g'); grid on; hold on;
% axis([0 10220 15 80]);
% set(gca, 'FontName', 'Times New
Roman','FontSize',20);
%
xlabel('t(s)','FontSize',20,'FontName',
'Times New Roman')
% ylabel('T
(°C)','FontSize',20,'FontName', 'Times
New Roman');
% legend('Rampas de
referência','Resposta do sistema
controlado')

for i =1:1
   pause(0.5)
      opt =
c2dOptions('Method','tustin','PrewarpFr
equency',8);
      FTd = c2d(F,Ts,opt);
      Cd1 = c2d(C1,Ts,opt);

      Gd=feedback(FTd,1);
      Gd1 = feedback(Cd1*FTd,1);
```

```
t = linspace(0,10200,40800);
[g,T]=lsim(Gd1,y,t);
T(40801)=[];
plot(T,g+25.3,'k','LineWidth',2);
grid on; hold on;
axis([0 10200 20 80]);
set(gca, 'FontName', 'Times New Roman','FontSize',20);

xlabel('t(s)','FontSize',20,'FontName', 'Times New Roman')
ylabel('T (°C)','FontSize',20,'FontName', 'Times New Roman');
legend('Rampas de referência','Resposta do sistema controlado')



end
%%
figure(2)
%impulse(796*FT*rampa,'--k',850*FT*rampa,'-.k',1000*FT*rampa,'--b',2000*FT*rampa,'-.b',G*rampa,'g',G1*rampa,'r')
%step((2000*FT)+25.3,(140*G1)+25.3)
impulse(796*FT*rampa,'--k',55*G1*rampa,'k',2000*FT*rampa,'--b',140*G1*rampa,'b')
clc
clear all
close all

%%
format shortg;
tempo_min = 2e-04;
tempo_max = 8.23e-03;
delta_tempo = tempo_max-tempo_min;
LSB = delta_tempo/1024;

Vpico = 171.8269;
R = 18.54962;

angle = (tempo_micro*pi)/(1000000/120);

%%
Potencia = ((Vpico^2)/(2*pi*R)) * (pi-angle+(sin(2.*angle)./2));


% for i=1:149
%     t(i) = tempo_micro(i);
%     p(i) = Potencia(i);
%
% end

plot(Potencia,tempo_micro,'k','LineWidth',0.5); hold on
%plot(p,t,'r','LineWidth',1); hold on

p1 = linspace(0.0115,8.8551,114);
tempo_1 = -96.613*p1 + 8020.3;    %t = 1024 - 910
p2 = linspace(8.8551,66.605,130);
tempo_2 = -17.464*p2 + 7382.3;    %t = 910 - 780
p3 = linspace(66.605,389.83,268);
tempo_3 = -6.3534*p3 + 6694 ;     %t = 780 - 512
p4 = linspace(389.83,706.07,249);
tempo_4 = -6.1479*p4 + 6631;      %t = 512 - 263
                                  %P(t) = 389,83 - 706,1
p5 = linspace(706.07,774.18,114);
tempo_5 =  -12.956*p5 +11458      %t = 263 - 149
                                  %P(t) = 706,07 - 774,18

p6 = linspace(774.18,795.75,149);
tempo_6 =  -50.934*p6 +40985 ;    %t = 149 - 1

plot(p1,tempo_1,'r'); hold on
plot(p2,tempo_2,'r'); hold on
plot(p3,tempo_3,'r'); hold on
plot(p4,tempo_4,'r'); hold on
plot(p5,tempo_5,'r'); hold on
plot(p6,tempo_6,'r'); hold on;
set(gca, 'FontName', 'Times New Roman','FontSize',20);
xlabel('P(W)','FontSize',20,'FontName', 'Times New Roman')
ylabel('t(us)','FontSize',20,'FontName', 'Times New Roman');
legend('Potência x Tempo de disparo','Curvas linearizadas ')

clc
clear all
close all
%% RESISTENCIA VALORES

P = input ('POTÊNCIA P(W): ');
V = input ('TENSÃO V(V): ');
comp = input ('COMPRIMENTO DO RESISTOR (cm): ');
d = input ('DIAMETRO DO EBULIDOR (cm): ');
fprintf('\nDigite o tipo do material\n1 - aluminio\n2 - inox 304|304L|316|316L|321\n3 - inox 410|420\n4 - cobre\n5 - outro\n');
material = input ('Material: ');


%%
%conversao para metros
comprimento = comp/100;
diametro = d/100;

Rr = (V^2)/P;
i = P/V;
area = comprimento*(pi*diametro);
volume = comprimento*(pi*(diametro^2)/4);

calor_especifico=0;
densidade = 0; %kg/m3

if material==1
   calor_especifico=910;
   densidade = 2700;
elseif material==2
   calor_especifico=502.416;
   densidade=8000;
elseif material==3
   calor_especifico=460.548;
   densidade=7800;
elseif material==4
   calor_especifico=393.56;
   densidade=8890;
else
```

```
   c=input ('CALOR ESPECIFICO: ');
   calor_especifico=
(c*4.186798)*1000; %conversao de
cal/g*C para J/kg*K
   densi=input ('DENSIDADE: ');
   densidade = densi*1000;
%conversao de g/cm3 para kg/m3
end


%% IMPRIMINDO OS VALORES

fprintf('POTÊNCIA P(W):
%.2f\t\t\t\t\t',P);
fprintf('TENSÃO V(V): %.2f\n',V);
fprintf('RESISTÊNCIA (Ohm):
%.2f\t\t\t\t\t',Rr);
fprintf('CORRENTE (A): %.2f\n',i);
 #
fprintf('COMPRIMENTO(m):
%.3f\t\t\t\t\t',comprimento);
fprintf('AREA DE CONTATO(m2):
%f\n',area);
fprintf('DIAMETRO DO CONDUTOR
DO EBULIDOR (m): %f\n',diametro);
fprintf('CALOR ESPECIFICO (J/kgK):
%d\t\t\t',calor_especifico);
fprintf('DENSIDADE (kg/m3):
%d\n',densidade);

%% Tempo para aquecimento da água

c_especifico_agua = 4186;%(J/kgK);
c_especifico_malte = 1549;%(J/kgK);
m_agua = 10;%quantidade de agua a
ser aquecida no projeto
m_malte = 4.0984;
Temp_delta = input ('Quantos graus
voce quer aquecer? '); %diferenca de
temperatura para ferver a agua (100-
25)
Q = ((c_especifico_agua*m_agua) +
(c_especifico_malte*m_malte))*Temp
_delta;
t= Q/P;
%conferir com a tabela de tempo de
fervura "electric-heat"
fprintf('TEMPO (s): %.2f\n',t);

%% A potencia para manter a taxa de
1°C/min
P = ((c_especifico_agua*m_agua) +
(c_especifico_malte*m_malte))*(1/60);
```

Figura Anexo C.1

(C.1)

Referências Bibliográficas

[1] Ambev, "Apresentação Institucional - Conferência BTG," em *Relatório de Apresentação Institucional - Conferência BTG - Fevereiro*, São Paulo, 2012, p. 34.

[2] L. Botto, "Curso de Produção de Cerveja Artesanal Caseira," p. 162, 2011.

[3] M. R. Reinold, "Reações enzimáticas e físico-químicas que ocorrem durante a malteação da cevada.," 2003. [Online]. Available: http://www.cervesia.com.br/malte/26-reacoes-enzimaticas-e-fisico-quimicas-que-ocorrem-durante-a-malteacao-da-cevada.html. [Acesso em 25 Setembro 2015].

[4] A. Fontes, "Cursos sobre Cerveja - Módulo III - Produção Artesanal," p. 36, julho 2011.

[5] J. J. Palmer, How To Brew - Everything you need to know to brew beer right the first time, NATL Book Network, 2006, p. 347.

[6] COPASA, "Pesquisa de Qualidade da Água - Portaria 2914," Belo Horizonte, 2016.

[7] A. Chaves, Física Básica (Gravitação, Fluidos, Ondas, Termodinâmica), Rio de janeiro: Editora LTC, 2007.

[8] F. Kreith e M. S. Bohn, Principios da Transferencia de Calor, 6ª edição ed., São Paulo: Pioneira Thomson Learning, 2003.

[9] F. P. Incropera e D. P. DeWitt, Fundamentos de Transferência de Calor e de Massa, 6ª ed., Editora LTC, 2011.

[10] D. Halliday e R. Resnick, Fundamentos de Física 2 - Gravitação, Ondas, Termodinâmica, Rio de Janeiro: LTC Editora, 2009.

[11] K. Ogata, Engenharia de Controle Moderno, Rio de Janeiro: Editora LTC, 2000.

[12] R. C. Dorf e R. H. Bishop, Sistemas de Controle Modernos, Rio de Janeiro: Editora LTC, 2001.

[13] J. C. Finkbeiner, "Resistências Elétricas - quase tudo que você quer saber," *Home Brew Talk Brasil*, p. 7, Abril 2013.

[14] A. P. Malvino, Eletrônica: Volume 1, São Paulo: Editora MAKRON Books do Brasil, 1995.

[15] D. W. Clarke, "Sensor, Actuator and Loop Validation," *IEEE Control Systems Magazine,* vol. 15, no. 4, pp. 39-45, August 1995.

[16] M. R. Reinold, "Água: a base para uma boa cerveja," *Beer Life,* p. 4, Março 2009.

[17] M. Q. Barboza, "O negócio milionário das cervejas artesanais," 16 Agosto 2013. [Online]. Available: http://www.istoe.com.br/reportagens/319458_O+%20NEGOCIO+MILIONARIO+DAS+CERVEJAS+ARTESANAIS. [Acesso em 25 Setembro 2015].

[18] R. Daniels, DESIGNING GREAT BEERS - The Ultimate Guide to Brewing Classic Beer Styles, Brewers Publications, 2000, p. 404.

[19] J. H. Giacomin, "Estudo de Viabilidade Econômico-Financeira de uma Microcervejaria no Estado de Santa Catarina," *Universidade Federal de Santa Catarina,* p. 89, 2008.

[20] F. L. d. S. Lima, "Como montar uma microcervejaria," *Idéia de Negócios,* p. 29, 2012.

[21] R. Mosher, Radical Brewing: Recipes, Tales and World-Altering Meditations in a Glass, Brewers Publications, 2004, p. 363.

[22] D. M. J. d. C. Bonfim, Sensores de Temperatura., Rio de Janeiro, 2005.

[23] R. Bird, Fenómenos de transporte = Transport Phenomena, 2a ed., Limusa Wiley, 2006.

[24] P. M. d. Santos, "Blonde Belge - Teste Temperatura," BeerSmith, Belo Horizonte, 2016.

[25] Measurement Specialties, "Resistance Temperature Detectors (RTDs)," Andover, 2014.

[26] H. D. Young e R. A. Freedman, Física II (Termodinâmica e Ondas), São Paulo: Editora Addison Wesley, 2003.