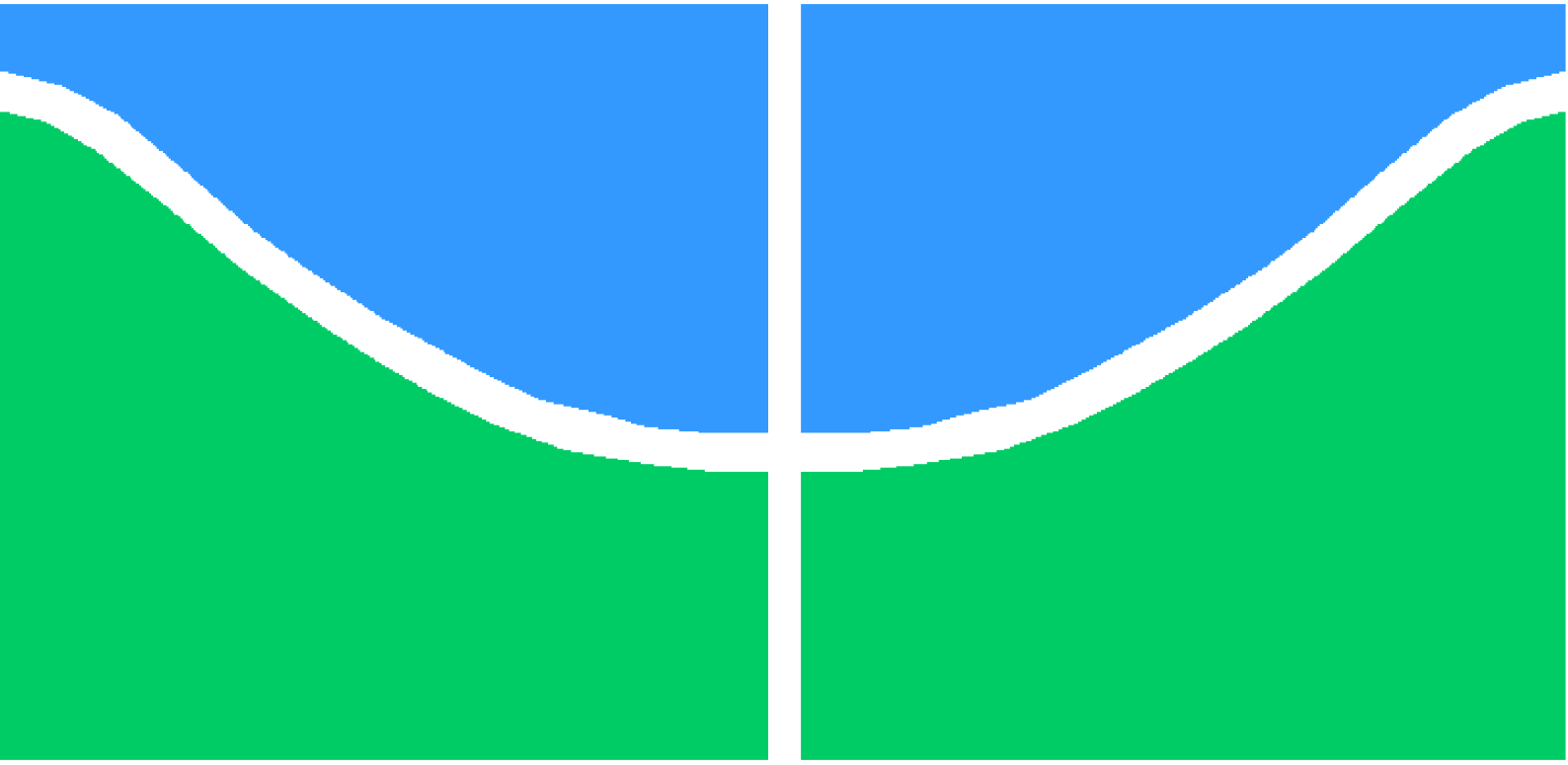

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia Eletrônica

Desenvolvimento de máquina semi-automatizada de fabricação de cerveja artesanal

Autor: Natália Santos Mendes

Orientador: Prof. Dr. Diogo Caetano Garcia

Brasília, DF

2018

Natália Santos Mendes

Desenvolvimento de máquina semi-automatizada de fabricação de cerveja artesanal

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Diogo Caetano Garcia

Brasília, DF

2018

Natália Santos Mendes
Desenvolvimento de máquina semi-automatizada de fabricação de cerveja artesanal/ Natália Santos Mendes. – Brasília, DF, 2018-
68 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Diogo Caetano Garcia

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. Automação. 2. Cerveja. I. Prof. Dr. Diogo Caetano Garcia. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Desenvolvimento de máquina semi-automatizada de fabricação de cerveja artesanal

CDU 02:141:005.6

Natália Santos Mendes

Desenvolvimento de máquina semi-automatizada de fabricação de cerveja artesanal

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Brasília, DF, 2018:

Prof. Dr. Diogo Caetano Garcia
Orientador

Prof. Dr. Tiago Alves
Convidado 1

Prof. Dr. Renan Utida Ferreira
Convidado 2

Brasília, DF
2018

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado segurança, inteligência e paciência durante toda minha graduação.

Aos meus pais, Ângela e David, que não mediram esforços para me oferecer o melhor durante toda minha vida acadêmica. Sempre me apoiaram nas mais diversas decisões, além de todo carinho oferecido diariamente. Foi graças ao meu pai que a estrutura da máquina ganhou vida. Obrigada por todo cuidado e amor!

Aos meu irmãos, Viviane e Rodrigo, que de forma sempre bem-humorada, me apoiaram e elogiaram meus feitos na área de engenharia. Obrigada por todos os momentos de risadas!

Aos meus amigos, os que me acompanharam desde o ínicio do curso e aos que conheci recentemente, que demonstraram companheirismo, amizade, momentos de diversão e risadas nas mais diversas situações. Aos que conheço desde o colégio e também os amigos de infância. Obrigada por todos momentos inesquecíveis!

Ao meu namorado, Felipe Duerno, que além de namorado, é meu coorientador, professor, melhor amigo e companheiro. Obrigada por ser o melhor namorado que alguém poderia ter!

Aos meus professores, desde a época do ensino médio até a faculdade. Os professores não só ensinam como também inspiram o melhor de seus alunos. Obrigada por toda paciência e todo aprendizado!

E por fim, agradeço à todos que, de algum forma, me ajudaram em alguma época dessa longa e maravilhosa jornada!

"Falou Daniel, dizendo: Seja bendito o nome de Deus de eternidade a eternidade, porque dele são a sabedoria e a força; E ele muda os tempos e as estações; ele remove os reis e estabelece os reis; ele dá sabedoria aos sábios e conhecimento aos entendidos. Ele revela o profundo e o escondido; conhece o que está em trevas, e com ele mora a luz."
(Bíblia Sagrada, Daniel 2, 20-22)

Resumo

O interesse em produzir cerveja artesanal em casa ganha mais adeptos com o passar do tempo. Porém, muitos desistem após poucas tentativas pelo trabalho e tempo que esse processo requer. Dessa forma, este trabalho propõe a criação de uma máquina semi-automatizada de fabricação de cerveja artesanal, que tem como objetivo otimizar esse processo para o usuário, trazendo o mesmo resultado quando feito todo manualmente, mas com a praticidade da automação. O protótipo é composto por duas panelas monitoradas por sensores de temperatura, com compartimentos exclusivos para os lúpulos e troca de líquido entre as duas panelas. Todo o processo é controlado pelo microcomputador Raspberry Pi. Com uma interface simples para o usuário, a ideia é de automatizar a maior parte do processo de brassagem - composto pelas etapas de mosturação, fervura, resfriamento e fermentação - otimizando assim o tempo do consumidor.

Palavras-chaves: cerveja, automação, brassagem.

Abstract

The interest in producing homemade beer gains more fans with the passage of time. However, many give up after a few attempts because of the work and time that this process requires. With that in mind, this work proposes the criation of an semi-automated brewing machine; which aims to optimize this process for the user, bringing the same result when done all manually, but with the practicality of the automation. The prototype consists of two pans monitored by temperature sensors, with exclusive compartments for the hops and exchange of liquid between the two pans. The entire process is controlled by a Raspberry Pi microcomputer. With a simple user interface, the idea is to automate most of the brewing process - consisting of the steps of blasting, boiling, cooling and fermentation, optimizing consumer's time.

Key-words: beer, automation, brewing.

Lista de ilustrações

Sumário

1 Introdução

Atualmente, a automação é um processo usual na vida das pessoas. Por sua popularidade, muitas das vezes as pessoas não notam a autonomia de diversos objetos que as cercam, direta ou indiretamente. Por definição, automação é o "sistema pelo qual os mecanismos controlam seu próprio funcionamento, quase sem interferência do homem" (HOLANDA, 2002). Ou seja, a ideia de automatizar algo é que esse algo não só não necessite mais da intervenção humana durante sua atuação, como também possa tomar suas próprias decisões, além de trazer confiabilidade a processos repetitivos.

Um de seus muitos benefícios é a liberdade que essa tecnologia pode trazer ao homem: as máquinas, ao controlarem e realizarem sozinhas processos antes considerados longos e demorados, trazem conforto ao ser humano e melhor aproveitamento de seu tempo. Por outro lado, em contexto industrial, traz preocupação aos trabalhadores, já que muitas vezes a máquina pode substituir o trabalho humano, deixando muitos desempregados. Já em outra visão, essa evolução da automação traz qualificação profissional aos empregados, visto que os mesmos precisam se especializar e buscar maior competência para o desenvolvimento de suas atividades. (SILVA, 2000)

A automação pode ser aplicada em grandes e pequenas escalas, desde a substituição do trabalho humano em grandes indústrias ou usinas, até em pequenos eletrodomésticos. O objetivo de automatizar algo é facilitar e/ou realizar o trabalho humano, trazendo assim mais facilidade ao usuário. Unindo esse pensamento ao aumento da popularidade de produtores caseiros de cerveja artesanal (NACHEL, 2008), propõe-se uma pequena máquina semi-automatizada de fabricação de cerveja. Ao automatizar o processo, o produtor dessa bebida deverá apenas se preocupar no início e final do processo, podendo assim desfrutar do seu tempo enquanto a maior parte da brassagem é realizada pela máquina.

A maior motivação para a execução desse trabalho foi de que, ao estudar o processo de fabricação de cerveja artesanal, percebeu-se que apesar de ser um hobby para muitos, há muito trabalho manual. Ele pode ser considerado cansativo pois o produtor deve sempre ficar próximo à cerveja, adicionando os ingredientes na hora correta, verificando se a temperatura está certa e também o tempo que já passou, trocando o líquido de uma panela para outra entre processos, etc. Por isso, automatizar esse processo seria uma forma de facilitar o trabalho do produtor que não está interessado em participar ativamente do processo, ou seja, que apenas quer saborear a sua cerveja customizada.

Esse processo de brassagem que será automatizado pode ser dividido em duas etapas: mosturação e fervura.

Na mosturação, a água é misturada com o malte e aquecida a diversas tempera-

turas em períodos determinados, convertendo o amido do malte em açúcar. Cada tipo de cerveja tem temperaturas e períodos específicos, por isso é importante escolher uma receita. Depois dessa fase, separa-se o malte do líquido, obtendo-se apenas o mosto doce. Posteriormente, esse mosto é elevado à temperatura de fervura por mais um período específico; nessa etapa, o lúpulo é acrescentado. Novamente, dependendo da receita, pode-se adicionar mais de um tipo de lúpulo e também por mais de uma vez.

Em seguida, deve-se resfriar o líquido. Algumas técnicas são deixar esfriar naturalmente ou passar por serpentinas inseridas em gelo. O importante desse processo é filtrar mais uma vez o mosto e oxigenar o líquido para facilitar o processo inicial de fermentação. Por fim, a levedura é colocada na cerveja já resfriada para iniciar o processo de fermentação. O líquido então é colocado em um recipiente fechado por um AirLock, que é um dispositivo que permite a saída do gás carbônico e proíbe a entrada do oxigênio.

Se for da vontade do consumidor, depois coloca-se essa cerveja em garrafas fechadas para o gás carbônico gaseificar o líquido e finalizar totalmente o processo de produção.

Como visto, são diversos processos para a produção de cerveja, além do controle de várias variáveis. Para isso, serão utilizados componentes eletrônicos, tais como sensores, relés e mini-bombas. Dessa forma, a maior parte dos processos será automatizado, ficando a cargo do usuário colocar os ingredientes no início do processo e, ao final, adicionar o fermento e colocar o líquido em refrigeração adequada.

Esse texto está separado em cinco partes: histórico e processo de fabricação de cerveja, funcionamento da máquina confeccionada, explicação da parte de hardware e software, resultados encontrados e conclusão.

2 Fundamentação Teórica

O processo de fabricação de cerveja requer várias etapas com detalhes específicos em cada uma delas. Nesse capítulo serão apresentados o histórico da cerveja, bem como o processo de fabricação.

2.1 Histórico

A história da cerveja começa há cerca de 10000 a.C. com um povo chamado curdo, na região de Curdistão, onde hoje estão a Turquia e o Iraque. Acredita-se que eles descobriram acidentalmente o processo da brassagem e a transformação do malte (SMITH, 2014).

Os primeiros nômades que cultivaram a cevada e o trigo foram os sumérios. Arqueólogos encontraram tábuas de argila contendo diversas receitas, em que a cerveja era aromatizada com tâmaras e mel. Em 2000 a.C., os babilônios subjugaram os sumérios, e então começaram a produzir cerveja em maior escala. O rei Hammurabi, inclusive, categorizou vinte tipos de cerveja, algumas usando apenas a cevada e outras com outros grãos, tais como trigo.

Os egípcios foram outro povo conhecido pela produção de cerveja, desde 3000 a.C. Seus aromatizantes eram ervas, gengibre ou açafrão. Os momentos da produção foram inclusive eternizados em arte, como pode ser visto na Figura 1. Tanto para os egípcios quanto para os babilônios, a cerveja tinha um significado mais especial: era uma oferta para os deuses para uma melhor vida após a morte (SMITH, 2014).

Figura 1 – Egípcios produzindo cerveja. Ao lado esquerdo é possível ver o processo de brassagem e ao lado direito o processo de envasamento. Fonte: (SMITH, 2014)

Os primeiros a produzirem em escala comercial na Europa foram os monges de

Saint Gallen, datada em 829 a.C. Eram produzidos três tipos de cerveja: celia, uma mais forte servida para os membros de alto escalão dos mosteiros; a cervisa, que era servida para os monges; e outra mais fraca para os visitantes e mendigos. Essa bebida era preferível do que a água, visto que o processo de fervura deixava-a menos contaminada.

Nessa época, os processos de brassagem não eram vistos do ponto de vista biólogico, mas sim como um milagre, principalmente a etapa da fermentação. Foi também nessa época que descobriram o lúpulo, que ajudava na conservação da cerveja. Ironicamente, o lúpulo, conhecido por dar o amargor desejado à cerveja, na época não era considerado saboroso.

Desde então, a bebida foi tornando-se mais popular. Ao espalhar-se na Europa, principalmente no século 16 quando foi criada na Alemanha a *Reinheitsgebot*, ou Lei de Pureza da Cerveja, a produção de cerveja não parou de crescer chegando então até as Américas e se espalhando pelo mundo.

2.2 Processo de fabricação de cerveja

O processo de fabricação começa com a escolha do malte. Essa escolha geralmente depende da receita que será utilizada, ou seja, o tipo da cerveja que será produzida. Geralmente usa-se a cevada, mas também pode-se usar trigo, centeio, aveia, sorgo, milho, entre outros. Ao selecionar a semente para colheita, ela é imersa na água para hidratar, trocando a água pelo menos uma vez. Depois disso, a semente é drenada e germinada em um ambiente úmido, com cuidado para as raízes das plantas não ficarem juntas (para isso, é necessário afofar a terra constamente). É nesse processo que as enzimas e açúcares se acumulam na parte externa do grão.

Os grãos são classificados em três tipos:

- **Malte verde:** para esse malte, o grão é colhido prematuramente. É rico em enzimas e é de fácil fermentação. Apesar do processo de secagem dos grãos ser em baixa temperatura e consequentemente deixar o processo mais barato, esse malte não é muito utilizado, pois pode trazer sabores indesejáveis na cerveja.

- **Malte branco:** Assim como o malte verde, ele é cozinhado em baixas temperaturas e a taxa de enzimas é considerada relativamente alta.

- **Malte escuro:** Os maltes com cores mais fortes são cozidos com temperaturas altas, e a quantidade de enzimas sobreviventes é baixa.

A escolha do tipo de grão do malte tem grande influência no produto final, como a cor, quantidade de carboidratos e a amargura da cerveja.

Geralmente, ao comprar o malte em lojas especializadas em cervejaria, já se compra o malte pronto para utilização, sendo necessário apenas a escolha do tipo.

Outro fator importante é a água, já que é o líquido usado na produção da cerveja. Em números, a cerveja é composta de 91%-98% de água (BRIGGS et al., 2004). Sais como magnésio ou cálcio influenciam no sabor do produto final, sendo necessários vários tratamentos antes de utilizar a água para o processo de brassagem. A fama do sabor da cerveja de muitos países advém das propriedades da água do local.

Para uma cerveja mais industrializada, são feitos cerca de três tratamentos diferentes para a água: processos de tratamento de pH (colocando soluções ácidas e alcalinas na água), tratamentos aeróbicos (colocando micro-organismos no líquido, separando-os em água, dióxido de carbono e iodo, removendo assim demanda bioquímica de oxigênio ou sólidos em suspensão (SYSTEMS, 2017)) e tratamentos anaeróbicos (coloca-se micro-organismos anaeróbicos que em uma taxa lenta transformam o líquido em água, dióxido de carbono e metano).

Devido aos processos de tratamento já realizados no saneamento nacional de água do Brasil, o fator a ser considerado na qualidade da água será apenas o do pH do líquido, que deve ser de 6.5-10 unidades de pH (BRIGGS et al., 2004). Porém, isso também depende do tipo de cerveja que será feita.

Com esses dois ingredientes, é possível começar o processo de mosturação. Ele é basicamente a mistura do malte com a água em certas temperaturas por tempos e temperaturas determinados pela receita, transformando o amido (aproximadamente 58% do malte (BRIGGS et al., 2004)) em açúcar simples (mono-, di-, e tri-sacarídeos), o que transforma a mistura no chamado mosto doce.

O processo de mosturação começa com o malte misturado em água quente. Isso se dá porque a extração das enzimas do malte em água fria é de cerca de 15-22% enquanto na água quente é de 75-83%. Dessa forma, na água quente obtém-se cerca de 53-68% mais mistura entre o malte e o líquido e consequentemente fermenta-se entre 75% a 85% a mais que na água fria (BRIGGS et al., 2004). Porém, a temperatura deve ser controlada pois as enzimas do malte ficam inativas em temperaturas muito altas.

Um dos fatores que podem ajudar a determinar as temperaturas médias e o tempo necessário para produção de cerveja é o nitrogênio permanentemente solúvel. Ele mostra quanto o malte foi modificado, ou seja, quanto maior esse número, melhor o processo de mosturação (NOONAN, 2017).

Cada receita tem sua própria temperatura de operação, que gira em torno de 50°C a 70°C, dependendo do tipo de mosturação. É nessa etapa que se transforma o amido em açúcar, de acordo com um complexo processo químico que não será detalhado nesse trabalho, pois foge ao escopo do mesmo.

Terminado o processo de mosturação, deve-se filtrar o malte do líquido. Existem diversos tipos de filtragem, tais como o *Brewing in a Bag* (BIAB) que é composto basicamente de um recipiente e um saco. Depois da primeira etapa do processo de brassagem, deve-se separar o malte do mosto. Para isso, basta retirar o saco e continuar o processo de brassagem (BHAGWAN, 2010).

Um processo de filtragem alternativo ao BIAB, que é o mais popular entre produtores de cerveja caseira, é instalar uma mangueira com pequenos furos no fundo da panela, e ao final do processo, abrir essa torneira, colocando todo o mosto em outra panela e deixando o malte na panela antiga. Uma variação dessa técnica é apenas colocar filtros redondos ou retangulares em lugares estratégicos na panela.

Depois da filtragem, inicia-se o processo de fervura, quando se inclui outro importante ingrediente na produção de cerveja: o lúpulo. O lúpulo é derivado de uma planta da família das Canabidácea. Ele é vendido inteiro, geralmente triturado em *pelets* ou em extrato. Sua função é dar aroma e/ou amargor à cerveja. Ele é colocado durante esse processo de fervura, e pode-se colocar mais de um tipo de lúpulo em uma receita.

Uma das consequências do processo de fervura, que dura de uma a duas horas, é que o líquido perde de 7%-10% de volume nessa etapa, devido a evaporação da água, além de mudar de cor (BRIGGS et al., 2004). Caramelos também podem ser adicionados para ajustar a cor.

O processo seguinte é o de refrigeração. Geralmente, resfria-se até chegar à temperatura ambiente. Essa etapa é importante pois no processo de fervura, o lúpulo se quebra em pequenos pedaços, deixando o mosto com alguns resíduos sólidos. Então, ao refrigerar, esses fragmentos se separam do mosto, tornando-se mais fácil a filtragem. Além disso, graças a esse processo, o mosto é oxigenado, facilitando o início da etapa da fermentação.

O último processo é o de fermentação. O fermento para cerveja (também chamado de levedura) é composto por pequenas células que, do ponto de vista químico, convertem o açúcar presente no mosto em álcool etanol, dióxido de carbono e calor (que é controlado pela refrigeração da cerveja). Do ponto de vista estrutural, na primeira parte da fermentação, o mosto filtrado deve ser colocado em um bacia fechada, meio fechada ou aberta, dependendo do fermento. Geralmente, ela é deixada meio aberta, com o uso do Airlock, que nada mais é que um pequeno cano em formato de S com água no seu interior que não deixa o oxigênio entrar para bacia porém permite o dióxido de carbono sair. Isso é necessário pois o oxigênio só é interessante no início do processo da fermentação, dado que posteriormente o processo feito pela levedura é anaeróbico. Já a saída do dióxido de carbono se faz necessária pois como a quantidade produzida é grande, se esse gás for colocado em ambientes fechados, existe o risco de explosão.

Cada receita tem um tipo de fermento com uma duração de tempo específica para

ficar nesse estado. Graças ao calor gerado na conversão, é importante deixar a cerveja durante todo o processo em temperatura controlada. Depois desse tempo estabelecido, deve-se engarrafar dessa vez com a garrafa toda fechada, para o dióxido de carbono gerado gaseificar a cerveja. Para iniciar a segunda etapa do processo de fermentação, utiliza-se a técnica do *priming*, que consiste em colocar um pouco de açúcar na cerveja para a conversão do açúcar em dióxido de carbono e álcool ser mais rápida. A quantidade é entre 4-7g/L (SILVA, 2015).

Esse processo está também explicado em forma de fluxograma, que pode ser visto na Figura 2. Os pontos em que contém um asterisco significam que esse detalhe depende do tipo de receita de cerveja escolhida.

Figura 2 – Fluxograma explicando o processo genérico de fabricação de cerveja.

Os tipos de cerveja mais populares são a Lager, em seguida Pale Ale e posteriormente a Porter e a Stout (MIRON; BROWN, 2006). A diferença entre esses tipos de cerveja são os ingredientes (malte, lúpulo e fermento) e o tempo de produção de cada uma delas. As cervejas do tipo Larger, por exemplo, tem tempo de fervura de uma hora. Já a Pale Ale e a Porter podem levar cerca de uma hora e meia. Esse dado das preferências de cervejas foi levado em consideração para a escolha dos tipos de cerveja que a máquina poderá fazer.

3 Metodologia

Nesse capítulo serão apresentados os materiais e métodos utilizados, bem como o modelo proposto para o protótipo confeccionado.

3.1 Materiais e métodos

Os materiais e softwares utilizados para confecção do trabalho foram:

- Computador DELL Inspiron 14 série 5000, com processador Intel Core i5, memória RAM de 8GB e disco rígido de 1TB, com sistema operacional Ubuntu 16.04;
- Raspberry Pi 3 B+, com sistema operacional Raspbian Stretch Lite, em um cartão de memória de 4GB;
- Software para criação e teste dos circuitos Proteus 8.1 Professional;
- Software para criação e teste dos circuitos Fritzing;
- Software para edição de códigos Gedit 3.18.3;
- Compilador gcc 5.4.0;
- Ferramenta para versionamento de códigos e texto Git e Github.

Os encontros com o professor orientador foram realizados quinzenalmente, com reuniões de aproximadamente trinta minutos na faculdade.

3.2 Modelo proposto para a máquina

Para decidir o modelo final proposto para essa máquina de fabricação de cerveja, foram feitos diversos modelos até chegar em um que conservasse as propriedades e caracteristícas desse processo. Foram levados em consideração a facilidade para manusear os utensílios, o tamanho final, as especificações no processo para não causar os chamados *off-flavors*, que são os defeitos, seja de sabor ou aroma, encontrados na cerveja, e também o custo final.

Dessa forma, o modelo proposto, nomeado por MakeBeer, é de uma máquina feita em aço, que comporte duas panelas uma ao lado da outra, monitoradas por um sensor de temperatura infravermelho, aquecidas por uma resistência potente, controladas por um microcomputador Raspberry Pi. As panelas terão filtros em seus fundos, e o

líquido será movido entre elas por mini bombas. A interface com o usuário será uma tela LCD, com botões para serem escolhidas as opções disponíveis. Acima de uma das panelas, haverá compartimentos para colocar o lúpulo, que será controlado por motores DC automaticamente.

Cada componente eletrônico foi pensado para atender todas as necessidades de fabricação da cerveja. Os sensores de temperatura foram colocados porque cada etapa do processo deve ter uma temperatura específica; portanto, essa variável deve ser medida e controlada. O aquecimento foi feito por resistências para controlar automaticamente quando ligá-las ou desligá-las. Os lúpulos devem ser colocados em tempos específicos na fervura, pois se colocados no começo alteram o amargor, e se for no final, o aroma. Logo, os motores DC foram pensados para liberar o compartimento no momento correto. As mini-bombas foram colocadas para separar o mosto doce do malte, na etapa da fervura; e também para transportar a água quente para panela do malte, na mosturação.

As seguintes etapas serão realizadas a cada processo de produção de cerveja, melhor demonstradas no fluxograma da Figura 3:

1. O usuário colocará o malte em uma das panelas e a água em outra. No compartimento específico, colocará o lúpulo.
2. Selecionará o tipo de cerveja que deseja ser produzida, através de instruções dadas pela tela LCD. A máquiná inicializará o processo.
3. Após a água ter aquecido, a bomba a transportará para a panela que contém o malte. Lá, o líquido passará pela etapa de mosturação.
4. Depois de o mosto doce ficar pronto, a bomba transportará novamente o líquido para segunda panela, passando o líquido por um filtro para separar o mosto do malte.
5. A mistura então entrará no processo de fervura, ficando nesse processo por um tempo estabelecido previamente.
6. A máquina será desligada.
7. O usuário esperará o líquido esfriar até a temperatura ambiente naturalmente.
8. O usuário deve realizar manualmente os demais processos: colocar a cerveja no balde fermentador juntamente com a levedura para começar o processo de primeira fermentação em um ambiente com temperatura controlada. Posteriormente, tranferir para garrafas para o segundo processo de fermentação (cujo objetivo é a gaseificação).
9. A cerveja estará pronta para consumo.

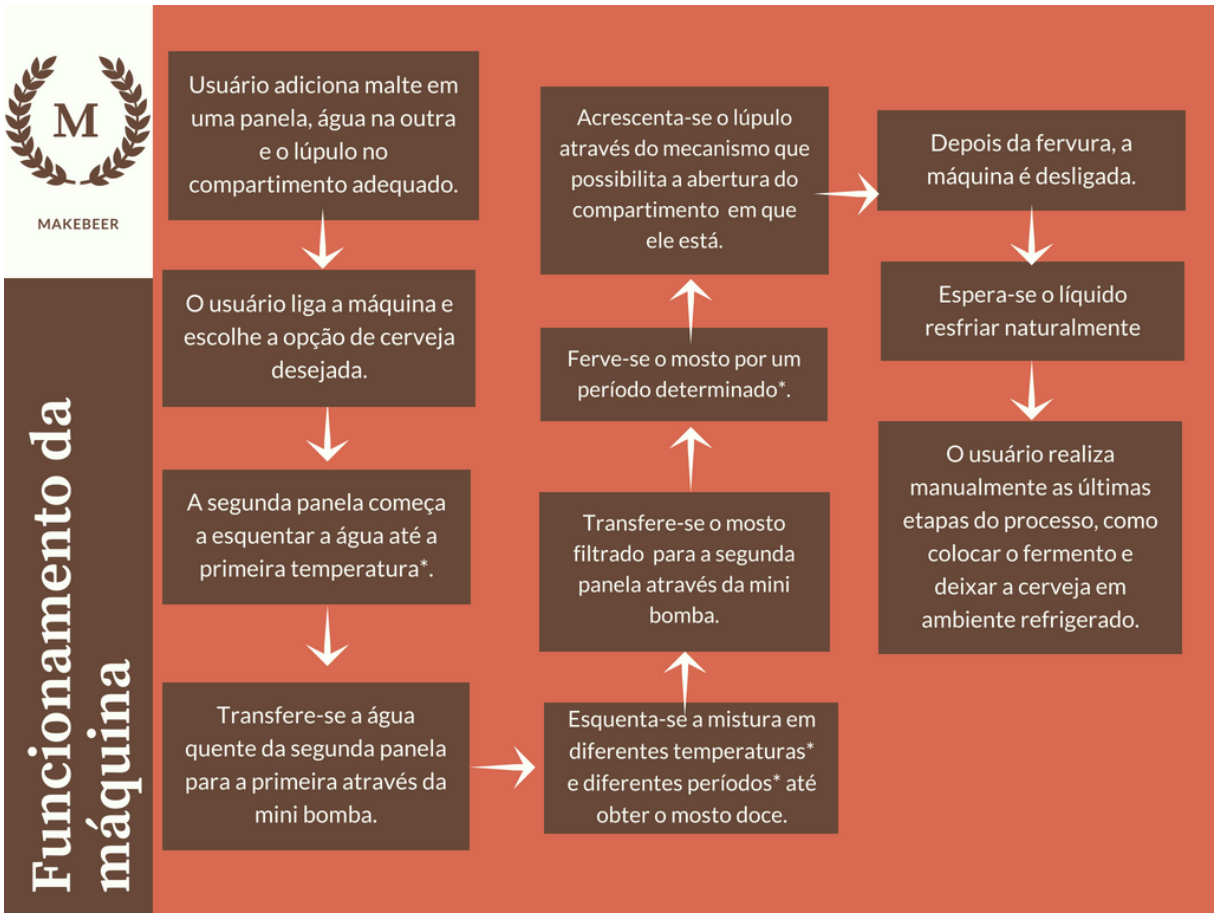

Figura 3 – Fluxograma contendo as etapas de funcionamento da máquina.

4 Estrutura

Para escolher o material de que a estrutura da máquina será feita, pensou-se em utilizar o acrílico, vidro temperado, aço inox e aço 1020. O acrílico é resistente e de fácil manuseio, porém começa a amolecer com temperaturas acima de 80°C. Já o vidro temperado, apesar de ser mais resistente que o vidro comum, ainda não é suficientemente resistente e pode quebrar com facilidade, dependendo do local em que é atingido. O aço inox tem propriedades parecidas com o aço 1020, porém é mais díficil de ser encontrado. As chapas de aço, apesar de possuir alta ductilidade, baixa dureza, resistência à altas temperaturas (ATKINS; JONES, 2009), e de ser um material de fácil manuseio, tem um custo muito elevado.

Por isso, foi escolhido a cantoneira de aço furado, que tem menor custo e atende as necessidades do projeto, pelo fato do seu material ser de aço.

A estrutura possui 1 metro de comprimento, 31 cm de largura e 47 cm de altura. As panelas usadas possuem 20 cm de diâmetro, e tem capacidade para até 6L.

A refrigeração será feita naturalmente, ou seja, o usuário pode deixar o líquido em algum ambiente com ventilação para acelerar o processo, ou deixá-lo durante a noite para garantir que, ao colocar o fermento, a temperatura do líquido esteja na temperatura ambiente.

A estrutura feita para máquina pode ser vista nas Figuras 4 e 5. As panelas são numeradas por 1 e 2 da esquerda para direita. Na primeira panela, encontra-se o filtro necessário para separar o mosto doce do malte. Esse filtro foi reutilizado de uma peneira comum. Há também outro filtro menor, como pode ser visto na Figura 6. Esse filtro foi colocado para impedir que pequenos pedaços do malte entrassem pela mini-bomba, evitando assim que a mesma pudesse entupir. As duas panelas contém um suporte para os sensores de temperatura, que ficam bem próximos às mesmas.

Já na segunda panela, existe o mecanismo para colocar o lúpulo na hora correta no momento da fervura. Para isso, foi preso um fio de nylon no eixo de um motor. A outra ponta desse fio foi presa em um mecanismo que serve de apoio para a abertura do compartimento que contém o lúpulo. Assim, quando o motor é ligado, ele puxa o fio que, por sua vez, puxa o apoio da abertura, abrindo o compartimento e liberando o lúpulo. Foram feitos dois compartimentos diferentes para liberar o lúpulo em momentos diferentes. Logo, foram usados dois motores. Todos esses detalhes podem ser vistos nas Figuras 7 e 8.

Na Figura 9, é possível ver o suporte feito para prender as mini-bombas. Toda sua

Figura 4 – Estrutura completa da máquina.

fiação, juntamente com a fiação dos motores e das resistências, foram presas em partes da estrutura.

Para os circuitos, foi feita uma caixa de tamanho 30cm por 30cm, com encaixes para tela LCD e três botões, e também espaços para entrada de fios e de ar, que pode ser vista nas Figuras 10, 11 e 12. Dessa forma, todos os circuitos e a Raspberry Pi ficam concentrados em um só lugar. As únicas partes visíveis para o usuário são a tela LCD e os botões, para escolha da cerveja desejada.

Figura 5 – Detalhes da estrutura, com foco na segunda panela.

Figura 6 – Dois filtros foram utilizados na primeira panela: um para grandes pedaços de malte e outro para menores, evitando o entupimento da bomba.

Figura 7 – Mecanismo usado para controle da queda do lúpulo.

Figura 8 – Os dois compartimentos para os lúpulos.

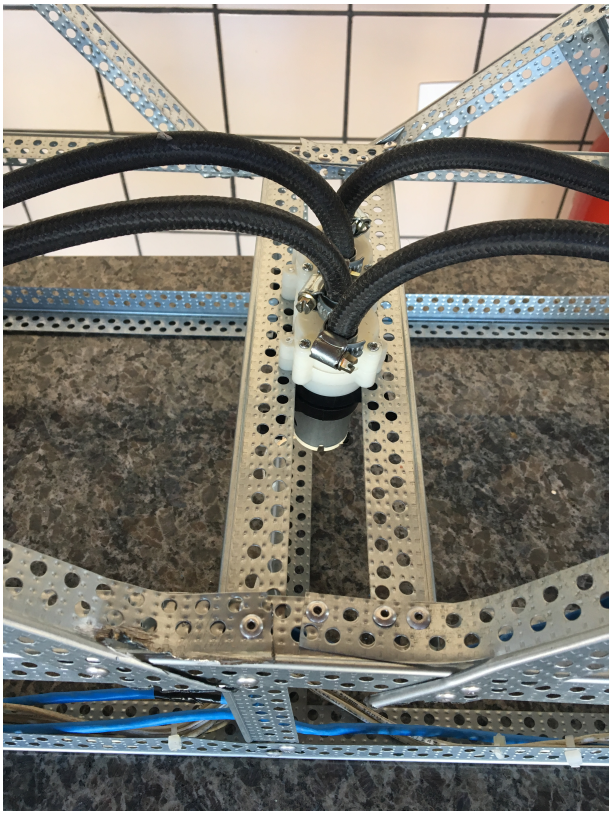

Figura 9 – Suporte e mini-bombas utilizadas no projeto.

Figura 10 – Parte da frente da caixa contendo os circuitos eletrônicos.

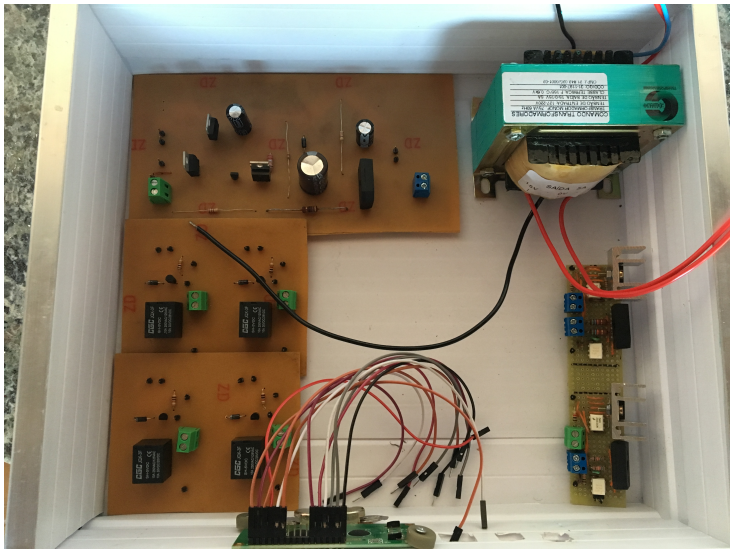

Figura 11 – Parte de dentro da caixa contendo os circuitos eletrônicos, sem a fiação nem a Raspberry.

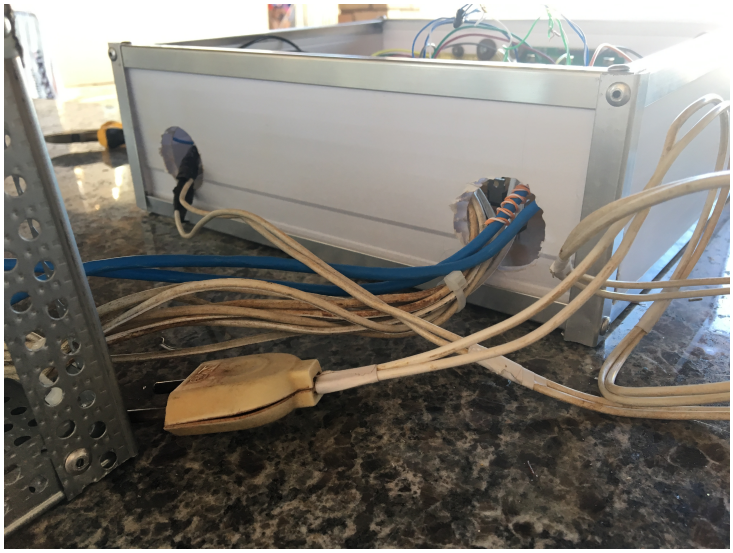

Figura 12 – Parte detrás da caixa contendo os circuitos eletrônicos.

5 Hardware

Os principais circuitos que compõem a máquina são a fonte de alimentação, o controle das mini-bombas por relés e o controle das resistências, realizado através de um TRIAC. O intuito da fonte de alimentação é, através da rede elétrica, fornecer tensão e corrente necessárias para os outros circuitos. Já o circuito usando relés serve para ligar e desligar as mini-bombas e motores quando necessário. E, por fim, o circuito da resistência tem como objetivo controlar a quantidade de corrente fornecida à carga, controlando assim a temperatura das panelas.

Nesse capítulo serão mostrados as explicações, cálculos e esquemáticos de cada circuito da máquina.

5.1 Fonte de alimentação

5.1.1 Potência e consumo

A fonte de alimentação foi projetada para alimentar todos os componentes do circuito: RaspberryPi, coolers, mini-bombas, motores e alguns circuitos integrados. Para dimensionar o transformador, foi levado em conta o momento em que mais componentes estariam conectados. A relação de tensão e corrente em cada componente pode ser vista na Tabela 1. Todos os dados foram retirados dos *datasheets* de cada componente.

Tabela 1 – Relação entre componente, tensão e corrente.

Componente	**Tensão (V)**	**Corrente (A)**
Coolers	12	0.12
Mini-bomba	12	0.7
Raspberry Pi	5	2.5
Motores	6	0.07
CI	5	0.01

O momento mais crítico seria com os dois coolers ligados, uma mini bomba, a Raspberry Pi e os circuitos integrados, totalizando aproximadamente 3,45A. O transformador adquirido tem como característica de 15V a 30V de tensão e 5A de corrente, valores suficientes para alimentar todos os componentes.

Sabendo disso, é possível calcular a potência do aparelho somando a potência consumida pelos circuitos e pelas resistências. Para calcular a potência, sabe-se da relação entre tensão e corrente, dada por

$$P = V * I \tag{5.1}$$

onde P é a potência em Watts, V a tensão em Volts e I a corrente em Amperes.

Aplicando essa equação para todos os circuitos, tem-se:

$$P = ((12 * 0.12) * 2) + (12 * 0.7) + (5 * 2.5) + (5 * 0.01) = 23,83W \tag{5.2}$$

Juntamente com a potência da resistência, tem-se:

$$P = 700W + 23,83W = 723,83W \tag{5.3}$$

Logo, para calcular o consumo de energia elétrica, basta multiplicar a potência pela quantidade de horas ligado e dividi-lo por 1000, obtendo-se:

$$Consumo = \frac{723,83W * 3,5h}{1000} = 2,53kWh/uso \tag{5.4}$$

5.1.2 Esquemático eletrônico

O circuito projetado é de uma fonte simples, composta por uma ponte de diodos (BR1) e capacitores (C2) para retificação. Além disso, foram colocados transistores (Q1 e Q2) para certificar a corrente necessária. Por fim, foram colocados reguladores de tensão (U1 e U2) de 6V e 15V para alimentar todos os pontos necessários dos outros circuitos. Inicialmente esse circuito foi simulado, como visto na Figura 13, feito desenho para placa de impressão, visto na Figura 14 e por fim, passado para placa, como mostrado na Figura 15.

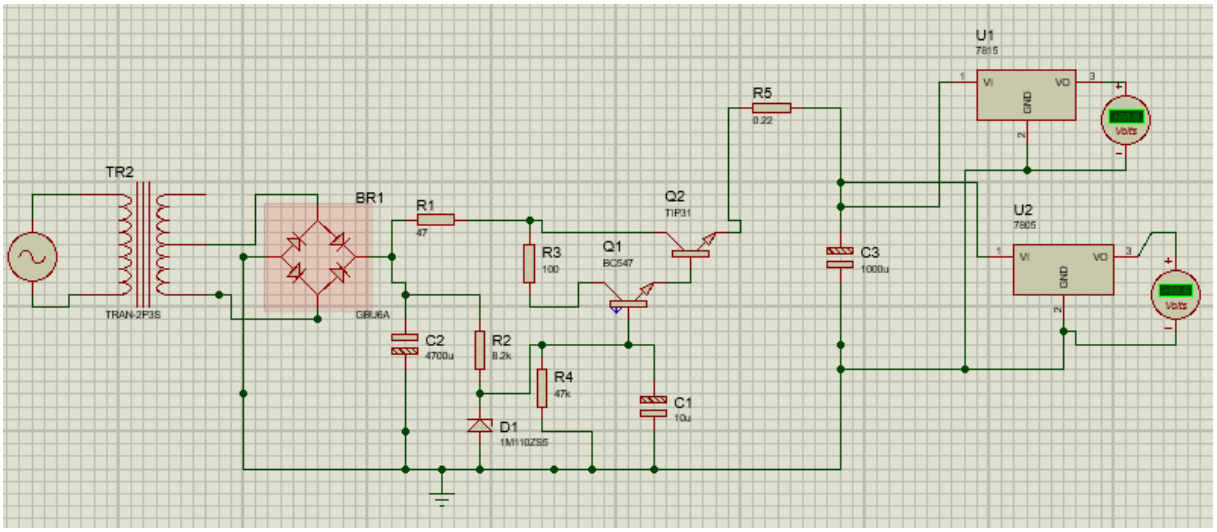

Figura 13 – Esquemático do circuito de alimentação.

- Componentes utilizados para confecção do circuito da fonte:
  - Ponte retificadora KBU-808;

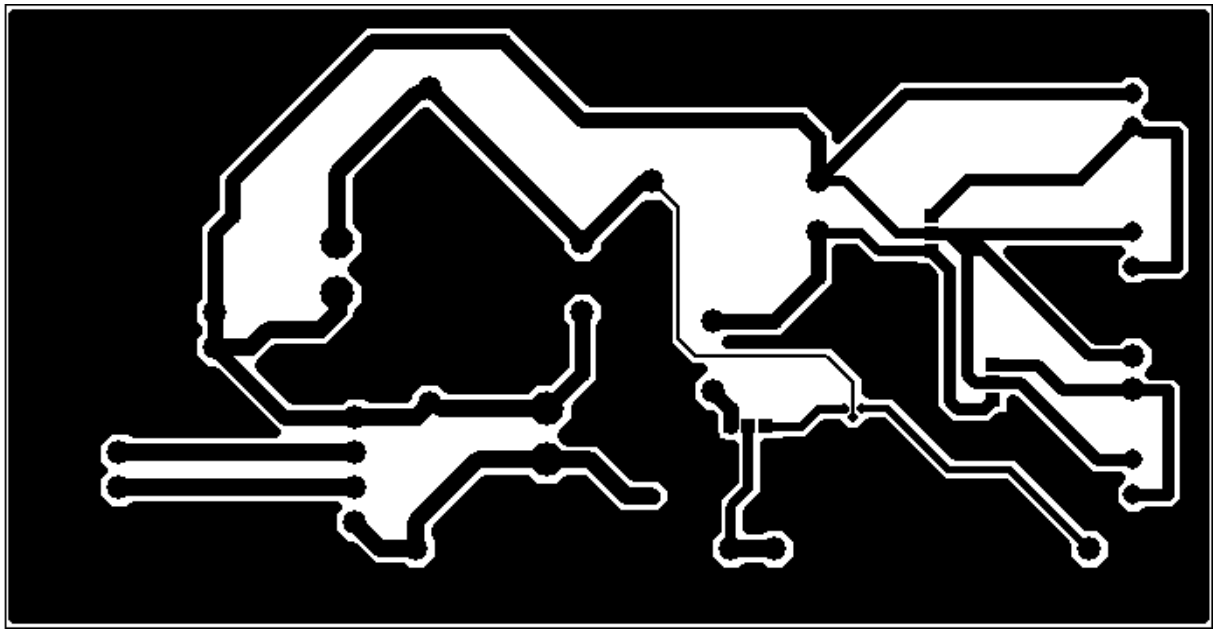

Figura 14 – Desenho da placa de circuito impresso.

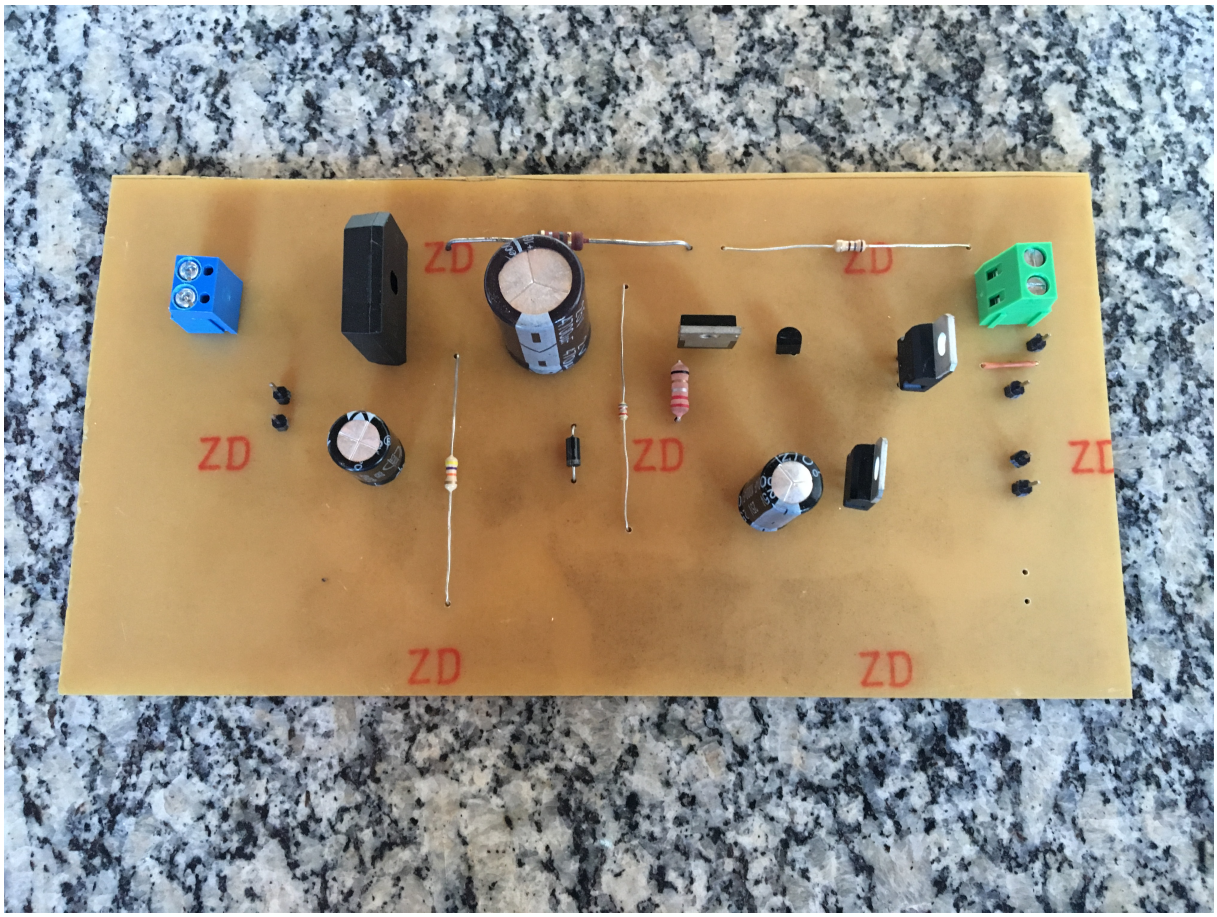

Figura 15 – Circuito da fonte de alimentação.

– Capacitor eletrolítico 4700uF 50V;

– Resistências de 47 Ω 2W;

- Resistências de 100 Ω, 8.2 kΩ, 47 kΩ e 0.22 Ω, todas de 1/4W;
- Diodo 1N4007;
- Capacitores eletrolíticos de 10uF e 1000uF, ambos de 63V;
- Dois transistores NPN BC547 e TIP31C;
- Reguladores de tensão 7815 e 7806.

5.2 Controle das resistências

O controle da corrente dada às resistências foi feito digitalmente através do controle PID (Proporcional, Integrativo, Derivativo), que será explicado posteriormente. O circuito de controle detecta o momento em que a onda da corrente alternada chega a zero e em seguida, há outro circuito que zera parte dessa onda para assim controlar a quantidade de corrente.

5.2.1 Detecção de zero

O principal objetivo de detectar quando a onda de corrente alternada vinda da tomada chega no ponto zero é de sincronizar o dimmer e assim a Raspberry Pi poder controlar quando disparar o TRIAC. Para isso, usa-se o optacoplador 4N35, que transmite para a Raspberry quando o ponto zero da onda é alcançado. Esse optacoplador é composto por um LED e um fototransistor. Quando a onda AC chega ao ponto zero, o LED ativa o fototransitor que conduz a tensão de 5V para o pino da Raspberry. Ela, por sua vez, gera uma interrupção no código que, depois de alguns cálculos, aciona o TRIAC no momento correto.

5.2.2 Controle do TRIAC

A segunda parte do circuito controla a corrente dada às cargas, que são resistências de 220V e 700W, uma em cada panela. O CI MOC3022 tem como objetivo disparar o *gate* do TRIAC quando necessário, além de proteger a Raspberry Pi contra descargas elétricas. O TRIAC funciona como uma chave eletrônica, que permite passar a corrente alternada quando disparado. Logo, quando detectado o ponto zero, dependendo da quantidade de corrente que deve-se passar, aguarda-se um tempo para disparar o TRIAC. Dessa forma, apenas parte da onda é transmitida para carga, como é possível ver na Figura 16. Logo, para fazer esse controle, usou-se digitalmente o controle de processos PID, e no TRIAC o PWM (Pulse Width Modulation), que é o controle da largura de pulso transmitido ao TRIAC para controle da potência.

Pensando nisso, foi elaborado um circuito de controle para cada uma das resistências, já que elas teriam controles independentes. Cada um conta com o circuito de

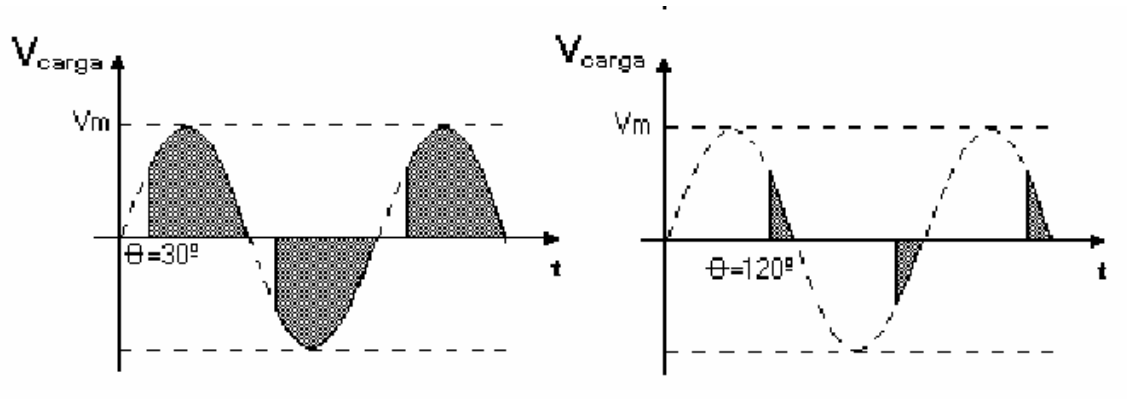

Figura 16 – Funcionamento da onda AC utilizando o TRIAC. O controle pode ser dado por angulação da onda ou por tempo após atingir o ponto zero. Fonte: (LIMA, 2011)

detecção de zero e do controle do TRIAC. O esquemático pode ser visto na Figura 17. Na Figura 18, é possível ver a foto do circuito pronto.

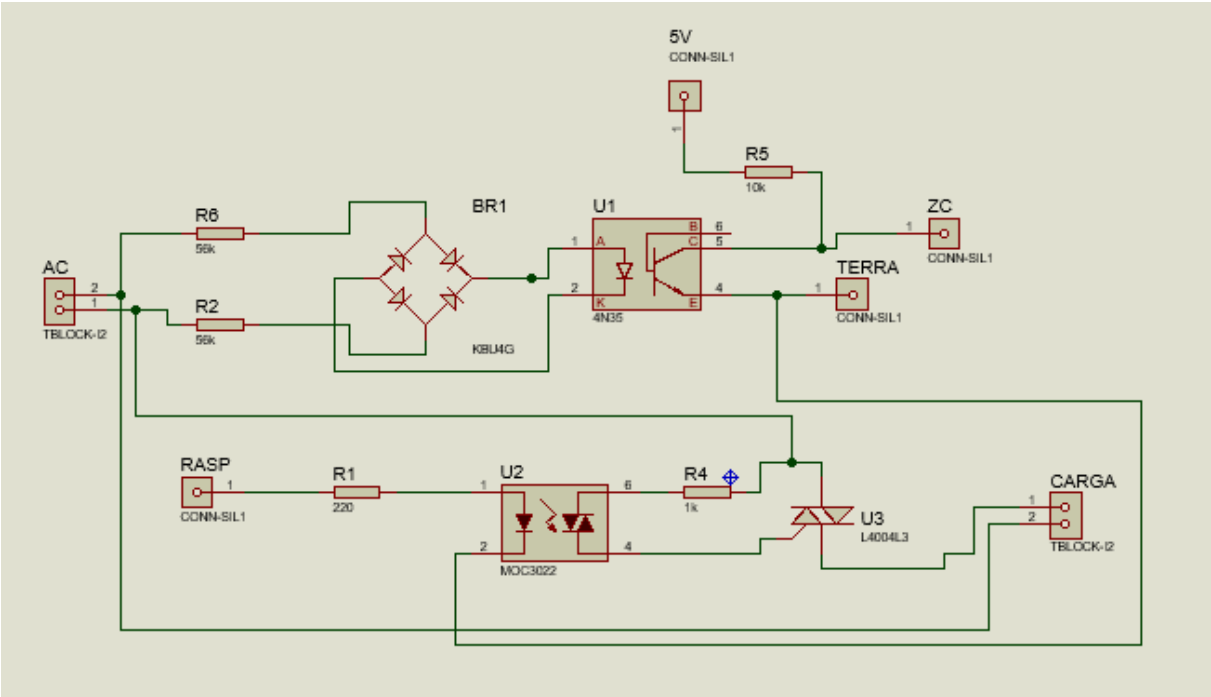

Figura 17 – Esquemático do circuito de controle das resistências.

- Componentes utilizados para a confecção dos circuitos das resistências:
  - Circuito integrado MOC3022 e 4N35;
  - Ponte retificadora KBU-808;
  - 2 resistências de 56 kΩ de 2W;
  - Resistências de 220 Ω, 1 kΩ, 10 kΩ, todas de 1/4W;
  - TRIAC BT136 com dissipador de calor;

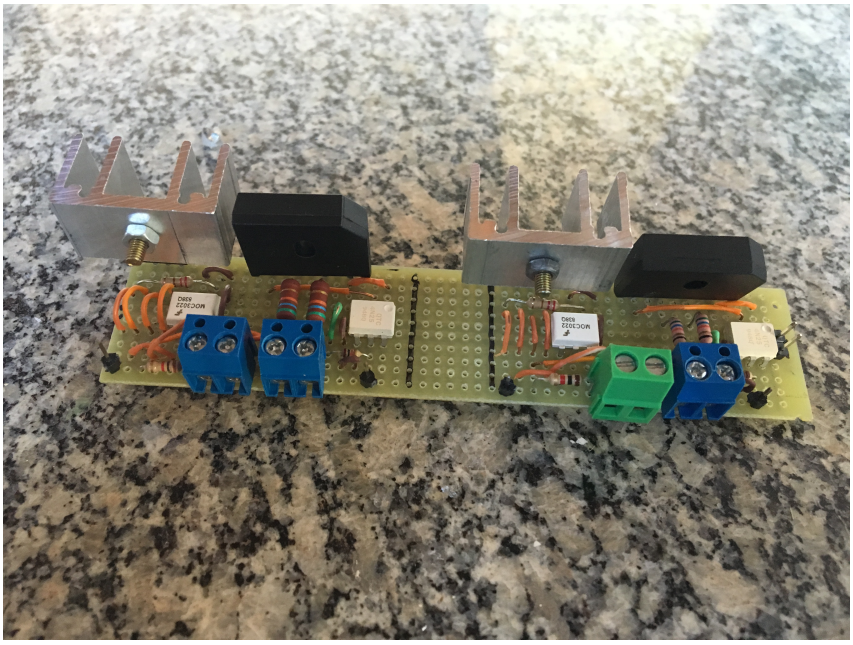

Figura 18 – Circuito de controle das resistências.

5.3 Sensor de temperatura

Uma das variáveis mais importantes para se obter é a temperatura, pois é através dela que se controla a resistência - que é a fonte de calor - controlando assim as etapas do processo de fabricação da cerveja. Para isso, foi utilizado o sensor de temperatura infravermelho MLX90614.

Esse sensor pode ser alimentado por 3V ou 5V, dependendo do modelo. O intervalo de temperatura que ele consegue medir é de -40°C a 125°C para temperaturas ambientes e -70° a 380°C para temperaturas em objetos específicos (MELEXIS, 2006). Sua principal característica é de que não é necessário o sensor encostar no objeto, visto que se mede a temperatura através da radiação infravermelha. Essa radição funciona da seguinte forma: o objeto que deseja-se medir a temperatura emite uma energia que é captada pelo sensor. Nele, encontram-se detectores fotosensíveis que convertem essa energia para sinal elétrico, que posteriormente é convertido em valores de temperatura. Essa relação de energia e temperatura baseia-se na Lei de Stefan-Boltzmann (PESSOA; SPINOLA, 2014), dada por:

$$W = \varepsilon K T^4, \tag{5.5}$$

onde W é a energia radiante total emitida por um corpo negro por unidade de superfície, ε a emissividade do corpo, K é a constante de Stefan-Boltzmann com valor de $5,6697 * 10^{-8} \frac{W}{m^2 K^4}$, e T a temperatura.

Além disso, o sensor conta com duas saídas com diferentes tipos de protocolo: I2C

e SMBus. Os dois, em geral, são compatíveis. Suas mudanças estão em velocidade do clock, tensão e corrente. Por ser mais comum, será utilizado o I2C, que é um protocolo de comunicação do tipo mestre/escravo com dois fios: o SDA (dados seriais, que transportam endereços, controles e dados) e SCL (clock serial, que sincroniza o transmissor e o receptor durante a transferência). Esse procotolo permite diversos escravos, sendo necessário apenas especificar o endereço do dispositivo escravo desejado (LIMA; VILLACA, 2012).

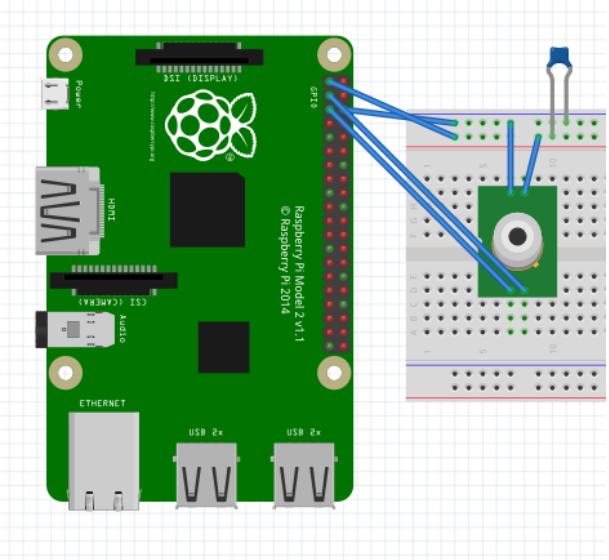

Figura 19 – Esquemático das ligações realizadas entre Raspberry Pi e o sensor de temperatura.

As ligações são simples: liga-se a conexão SDA e a conexão SCL do protocolo I2C do sensor nos pinos 3 e 5 da Raspberry Pi, liga-se o de alimentação (Vdd) no pino 3,3V e o Vss ligado ao terra, com um capacitor de 0,1uF entre o Vdd e Vss. Esse sensor de temperatura trabalhará em conjunto com o controle PID para regular a temperatura da panela.

5.4 Controle das mini bombas e motores DC

Para passar o líquido entre as duas panelas - inicialmente para passar a água quente para panela do malte, depois o líquido da panela do malte retornando à primeira panela para o processo de fervura - serão usadas duas mini bombas. Elas suportam temperaturas satisfatórias para o projeto (80°C), são pequenas, fornecem uma vazão de 1,5 a 2L por minuto, são alimentadas por 12V e necessitam de 0,7A de corrente.

Essa bomba foi escolhida pelo seu leve peso e pela rápida vazão - como serão produzidos 5L de cerveja, no máximo em 4 minutos todo o líquido será transportado.

O controle dessas mini-bombas e dos motores DC que acionam os compartimentos dos lúpulos é feito através de relés. As mini-bombas são alimentadas por 12V e os motores

por 6V. Logo, a lógica do circuito é, quando o relé não está sendo alimentado, ele é conectado ao pino NF (normalmente fechado), portanto não transfere tensão para o motor e o mesmo não liga. Ao dar o comando pela Raspberry, o transistor BC548 funciona como uma chave e começa a conduzir a alimentação de 5V, ativando o relé. Dessa forma, ele começa a conduzir a tensão do pino NA (normalmente aberto) e conduz 12V à bomba, ligando-a durante o tempo desejado.

É importante ressaltar que, como cada bomba tem acionamento independente, cada motor tem seu circuito de controle individual, totalizando quatro circuitos utilizando o relé.

Na Figura 20, é possível ver o esquemático do circuito, na Figura 21 o desenho da placa de circuito impresso e na Figura 22 o circuito pronto.

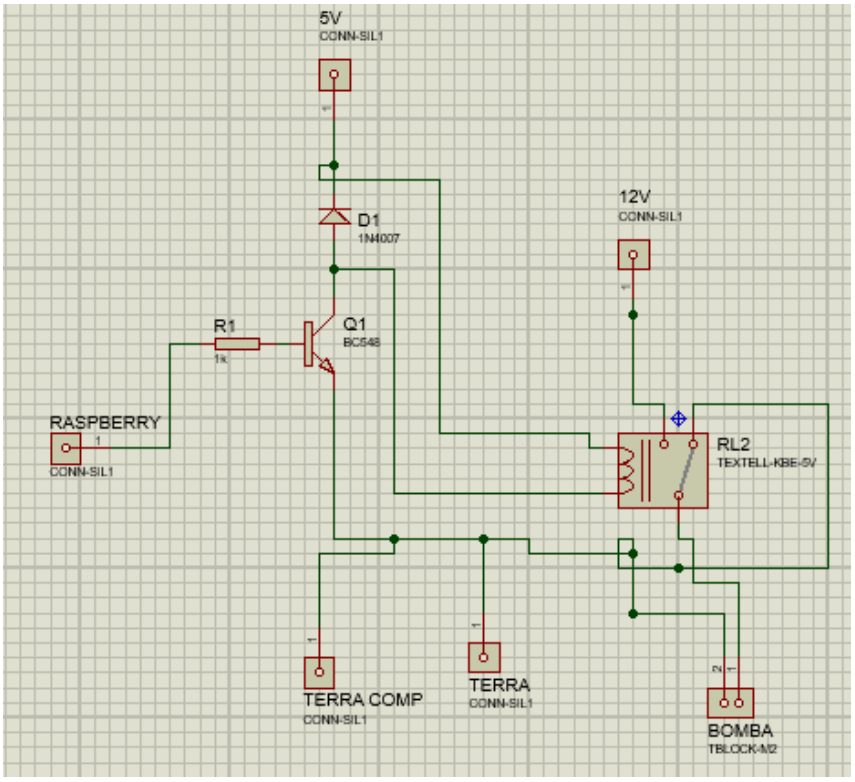

Figura 20 – Esquemático do circuito de controle das mini-bombas.

- Componentes utilizados para a confecção do circuito de controle das mini-bombas e motores:
  - Relé 5V;
  - Transistor NPN BC548;
  - Resistência de 1 kΩ 1/4W;
  - Diodo 1N4007;

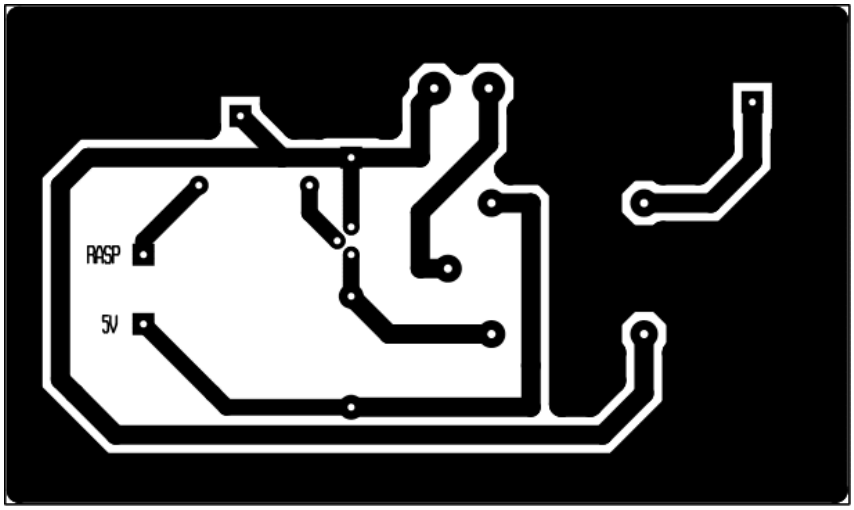

Figura 21 – Desenho da placa de circuito impresso do controle das mini-bombas.

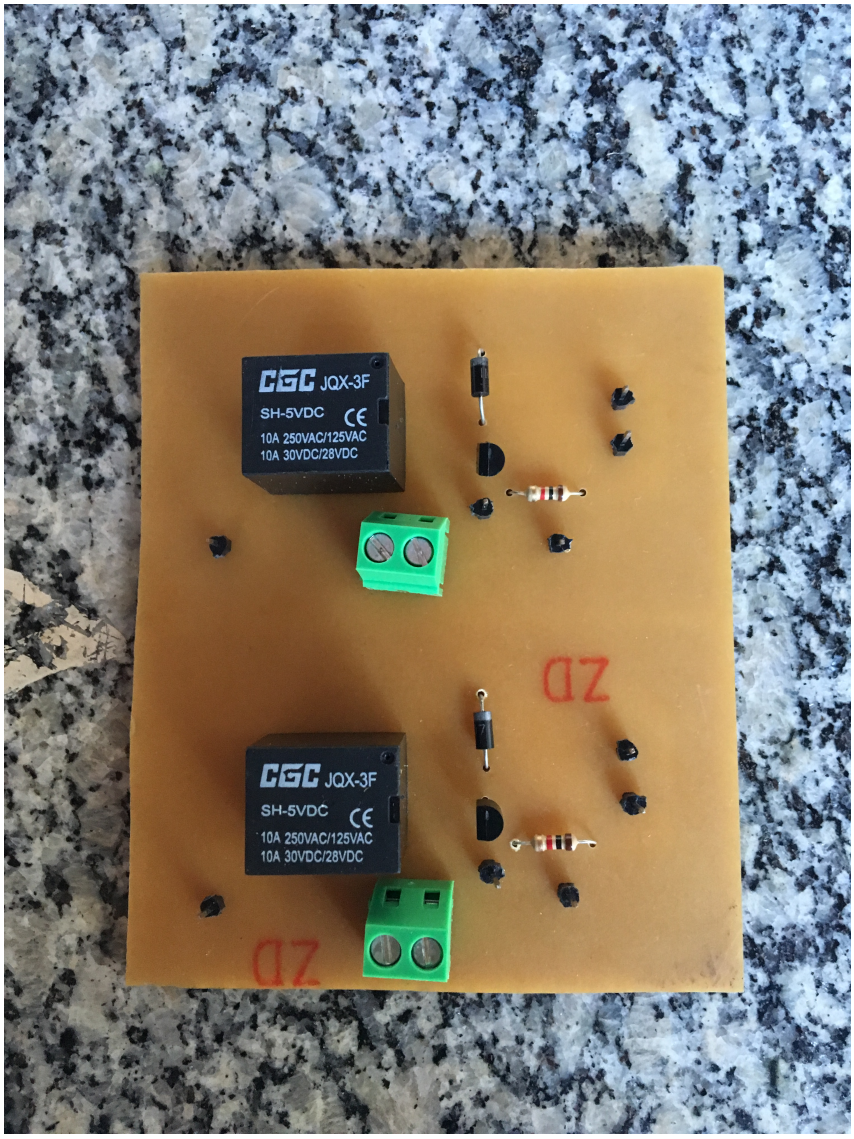

Figura 22 – Circuito de controle das mini bombas.

6 Software

Este capítulo tem como objetivo explicar a lógica utilizada no código para controle da máquina. Todo o código encontra-se no apêndice.

O fluxograma de explicação do código pode ser visto na Figura 23.

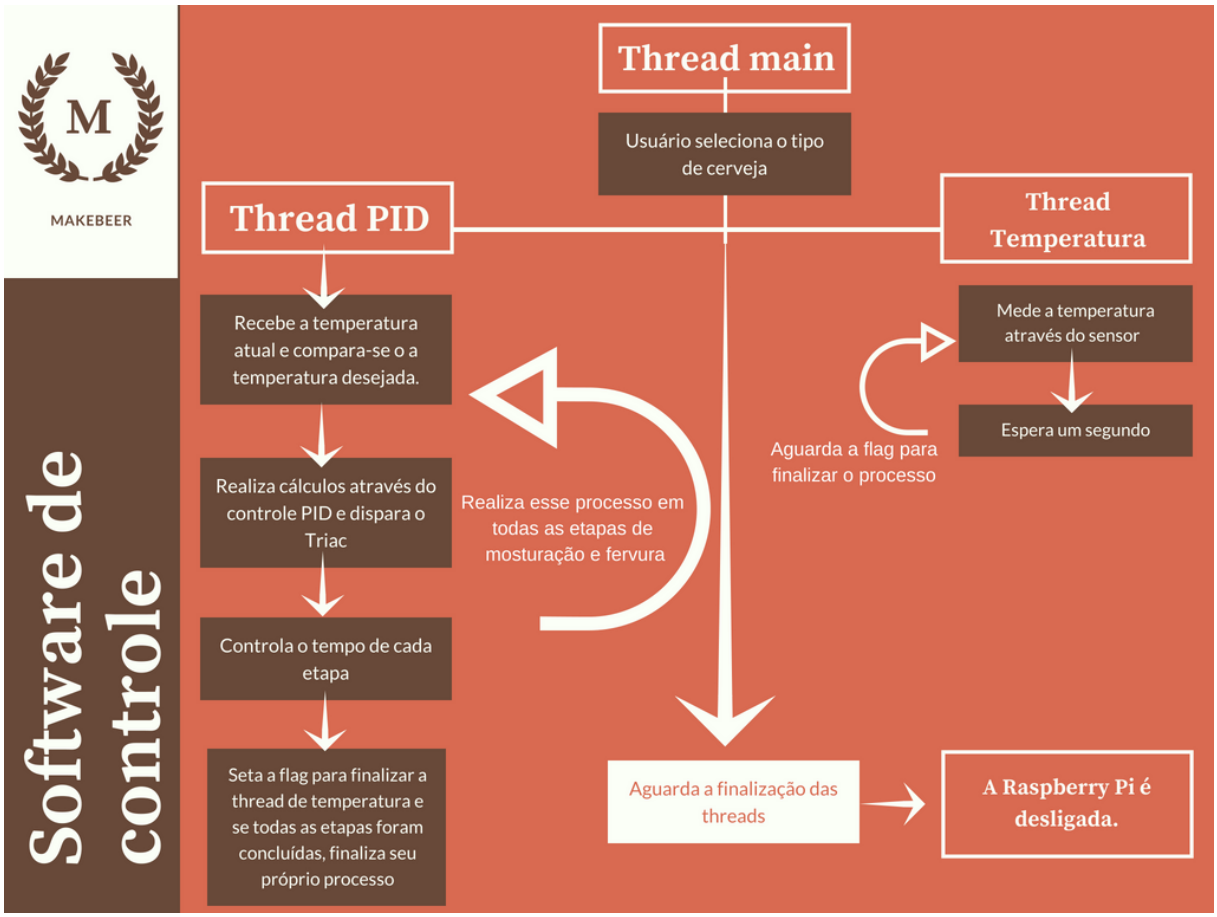

Figura 23 – Fluxograma contendo a lógica do código utilizado para controle.

O software contém três *threads*: a principal, a de PID e a de temperatura. Ao inicializar a Raspberry, primeiro há a opção de escolha da cerveja dependendo do botão pressionado pelo usuário. Depois disso, essas duas *threads* são criadas.

A *thread* da temperatura tem como objetivo calcular a temperatura medida pelo sensor infravermelho a cada um segundo. Para isso, utiliza-se a biblioteca bcm2835 para fazer a comunicação i2C, já que esse é o protocolo utilizado pelo sensor. Para conseguir obter os dados dos dois sensores, foi necessário modificar o endereço de um deles. Dessa forma, primeiro estabelece-se uma comunicação com um dos sensores (via i2C), obtém-se a temperatura, fecha a comunicação, inicia-se outra com o outro escravo, obtém-se a temperatura e fecha-se a comunicação novamente.

Como retirado do *datasheet* do sensor, os dados vem em 10 bits e em Kelvin, portanto o código também é responsável por ler esses bits e transformá-los em graus Celsius.

A *thread* do PID tem como objetivo calcular quando deve-se disparar o TRIAC para obter a quantidade de corrente necessária da onda AC na carga. Os cálculos feitos foram para obter justamente o tempo necessário de espera antes de ativar o componente. Sabe-se que a corrente alternada vinda da tomada de uma residência no Brasil tem frequência de 60Hz. Sabendo-se que o período é o inverso da frequência, tem-se:

$$T = \frac{1}{f} = \frac{1}{60} = 16,6ms \tag{6.1}$$

onde T é o período e f a frequência.

Como para o TRIAC é necessário analisar só metade da onda, divide-se esse período por 2, obtendo-se 8,33 ms. Para controlar esse tempo com PID, esse tempo foi dividido em 128 passos (quantidade suficiente para se realizar um controle preciso dos TRIACs).

$$T_{passo} = \frac{8,33ms}{128} = 65us/passo. \tag{6.2}$$

Dessa forma, dependendo da diferença da temperatura atual com a temperatura desejada, poderia-se multiplicar 65us pela quantidade de passos depois dos cálculos do PID, para obter o tempo de espera antes do disparo do TRIAC. Por exemplo, se não deseja-se que passe corrente para carga, passariam-se 128 passos, para o software esperar 8,32ms depois que a onda atingesse zero para ativar o TRIAC. Como ela já estaria no final, muito pouco seria passado para carga. Mas, por outro lado, se fossem passados 7 passos, esperaria-se 455us para ativar o TRIAC e a maior parte da onda seria transmitida à carga.

Nessa parte de software, foi criada uma função para calcular o PID. Essa sigla significa Proporcional-Integral-Derivativa, que é um sistema em malha fechada que utiliza um retorno obtido de qualquer váriável, como a temperatura, e mantém um ponto de ajuste. Esse ponto de ajuste, conhecido como *setpoint*, é comparado com o valor da variável atual, e a diferença entre os dois é chamado de erro, que deve ser minimizada pelo processo (LAMB, 2015). A saída é o atuador que controlará o sensor responsável pela mudança do processo. Esse processo pode ser visto na Figura 24.

O erro atual está relacionado a parte proporcional do controle, e é uma espécie de compensador direto ao erro encontrado. Já o valor integrativo pode ser considerado como um acúmulo de erros anteriores e o derivativo uma predição de erros futuros. (LAMB, 2015)

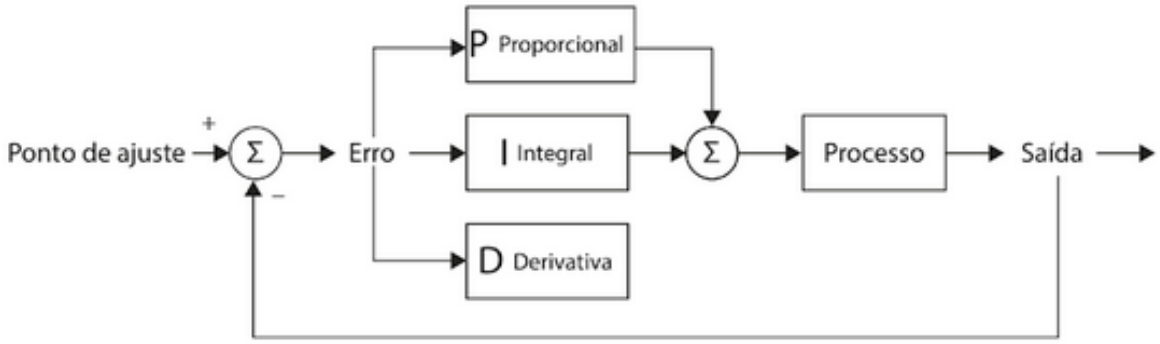

Figura 24 – Diagrama de blocos PID. Fonte: (LAMB, 2015)

Matematicamente, essas relações podem ser dadas pela expressão encontrada na Equação 6.3.

$$u\left(t\right) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}, \tag{6.3}$$

onde *u(t)* é o sinal de saída; K_p, K_i e K_d os ganhos proporcional, integrativo e derivativo; e o erro e τ o tempo de integração.

Os ganhos proporcional, integrativo e derivativo usados foram descobertos de forma empírica. Os valores usados foram 0.4, 0.5 e 0.01, respectivamente.

No código foi feito uma função que fazia o cálculo descrito na Equação 6.3. Essa função é chamada sempre pela *thread* PID, enquanto algum processo da cerveja está ativo. A quantidade de passos, descrita anteriormente, é calculada pela quantidade de passos atual menos a saída dos cálculos do PID.

Por fim, a *thread* principal tem como objetivo determinar os tempos de cada processo, dependendo do tipo de cerveja escolhida, criar as outras duas *threads* e esperar até que elas finalizem para, ao final, desligar a Raspberry por um comando *System*.

7 Resultados

7.1 Custos

Para a soma dos custos, foi levado em conta todos os gastos referentes à máquina, incluindo equipamentos que foram usados em testes e protótipos, e pode ser visto na Tabela 2. Os itens não contabilizados se referenciam a objetos eletrônicos, tais como capacitores e resistências; e todos os materiais utilizados na estrutura. Não há contagem deles pois são muito específicos e essa é uma tabela resumida.

Tabela 2 – Custos

Item	Quantidade	Valor unitário (R$)	Valor total (R$)
Panela	2	31,20	62,40
Mini-bombas	2	33,00	66,00
Raspberry Pi	1	150,00	150,00
Balde fermentador de cerveja	1	89,81	89,81
Transformador	1	99,90	99,90
Componentes eletrônicos em geral	-*	-	394,92
Materiais da estrutura	-*	-	207,00
Outros	-*	-	23,00
Total	-	-	1093,03

* Itens não contabilizados.

7.2 Testes individuais

Nessa seção serão explicados os testes realizados em cada circuito. Só serão explicados aqui os testes realizados depois da confecção da placa de circuito impresso, e não os testes iniciais realizados em *protoboard*.

7.2.1 Fonte

Para teste da fonte, foi conectado a tensão AC ao transformador e em seguida, à fonte. Para verificar as tensões nas saídas, utilizou-se um multímetro. Além disso, verificou-se a temperatura dos componentes enquanto utilizava-se essa fonte, principalmente em momentos em que a carga estava conectada e em uso.

O transformador e o circuito conseguiram atender as necessidades de corrente e tensão dos componentes utilizados em outros circuitos, sem sequer ser necessário refrigeração ou dissipadores de calor.

7.2.2 Controle das mini-bombas e motores

O controle tanto das mini-bombas quanto dos motores foi feito da mesma forma: através de um circuito em que o comando da Raspberry acionava o relé e permitia a tensão necessária ir para a carga durante o tempo estabelecido por código. Portanto, para teste individual, conectou-se as cargas na saída de cada um dos quatro circuitos, alimentou-as com as tensões vindas da fonte e executou-se o código que pode ser visto no apêndice B.

Como resultado, obteve-se o armamento e depois desarmamento do relé, ligando e desligando as cargas de acordo com o comando da Raspberry, obedecendo o tempo estabelecido.

7.2.3 Controle das resistências

Foram feitos testes do circuito de controle das resistências inicialmente apontando o sensor de temperatura diretamente para a resistência e posteriormente na panela pronta da estrutura. O teste foi feito acompanhando os valores da temperatura mostrados pela tela do computador, que estava conectado via SSH na Raspberry.

Dessa forma, notou-se que, ao alcançar a temperatura desejada, o controle da Raspberry atuava de forma que impedia a passagem de corrente para a resistência. Porém, como essa resistência já estava quente e ela leva um tempo para perder calor, a temperatura ultrapassava o limite até esfriar e estabilizar novamente. O código utilizado para teste pode ser visto no apêndice C.

O gráfico da Figura 25 ilustra os valores obtidos para aquecer a água da panela até 52ºC. Esse valor foi escolhido pois era a primeira temperatura que deveria ser atingida na receita de uma American IPA. Para atingir essa temperatura, com 5 litros de água na panela, levou-se cerca de 26 minutos.

7.3 Resultados finais

Com todos os circuitos funcionando individualmente, passou-se para etapa de integração. Foi feito o código para controle geral da máquina, que pode ser visto no apêndice A. O código foi compilado sem nenhum erro. Fez-se a ligação dos componentes como mostrado na Figura 26. Dessa forma, foram utilizados 26 pinos, no total de 40 disponíveis na Raspberry Pi 3 B+.

Então, ligou-se todos os pinos necessários e executou-se o código na Raspberry. Após cerca de 10 segundos executando o código, a Raspberry queimou e não ligou novamente.

Depois, foi conectada outra Raspberry Pi Model B, contendo 20 pinos, para novos

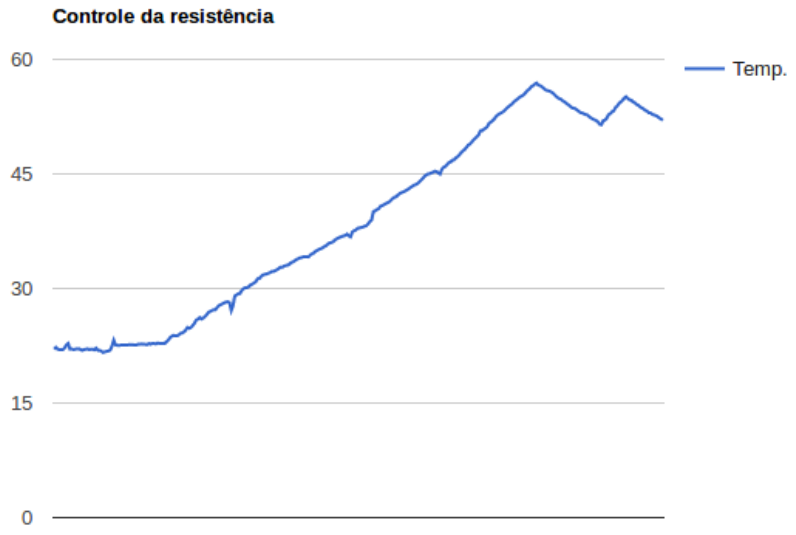

Figura 25 – Gráfico mostrando o comportamento do controle da resistência.

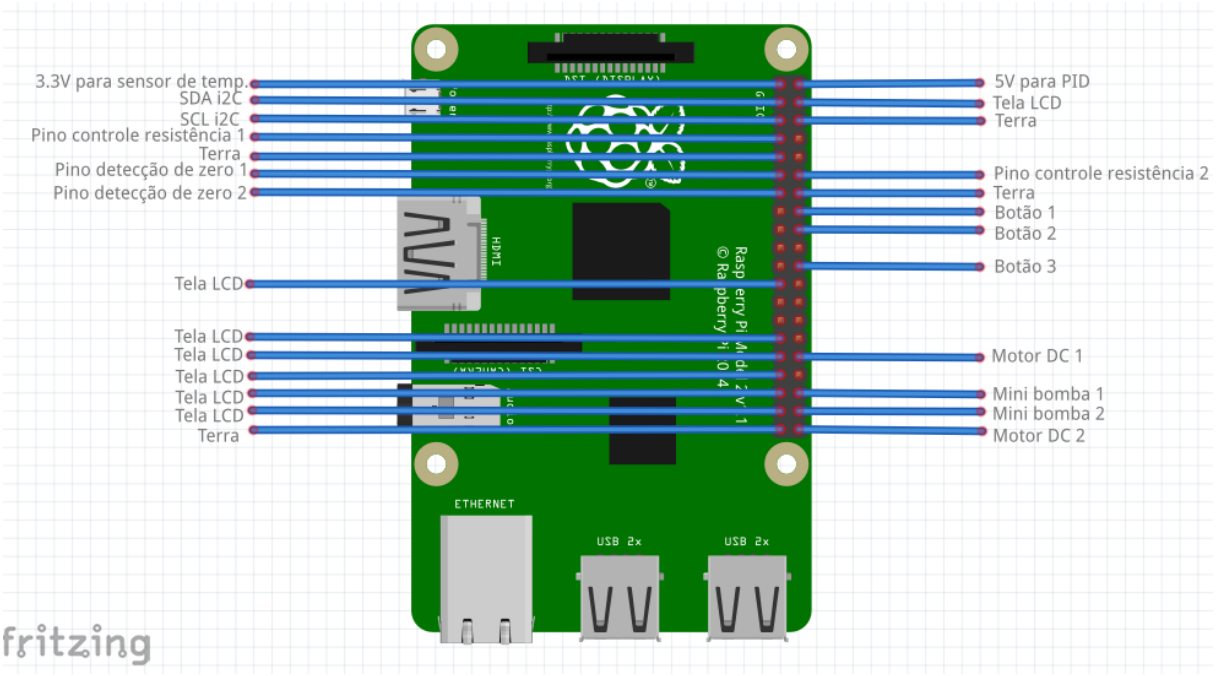

Figura 26 – Esquemático com todas as ligações realizadas na Raspberry.

testes. Dessa vez, foram retirados pinos "desnecessários"para o funcionamento da máquina, como a tela LCD e os botões de escolha, liberando então 12 pinos. Porém, como essa Raspberry possuia menos pinos, nem foi necessário ligar todos os circuitos para o microcomputador esquentar e desligar por segurança. Para evitar a queima, os testes gerais foram cessados.

Ao testar individualmente cada circuito, foram usados por vez poucos pinos da Raspberry Pi. Dessa forma, nunca ultrapassou-se o limite de corrente disponibilizado

pelos pinos do microcomputador; então todos os testes obtiveram sucesso. Entretanto, ao realizar o primeiro teste de integração, onde foram usados 26 pinos dos 40 disponibilizados, a Raspberry Pi queimou. O provável motivo para a queima foi de uso exagerado dos pinos. Apesar do cuidado em separar os circuitos de potência dos pinos da Raspberry justamente para não forçar a corrente, a grande quantidade de pinos usados utilizou mais que os 50mA oferecidos pelo microcomputador.

8 Conclusão

A paixão pela produção de cerveja artesanal faz amantes dessa bebida produzirem grandes quantidades de cerveja por hobby, e nos últimos tempos essa ideia tem se popularizado ainda mais. Porém, o processo de fabricação de cerveja pode ser cansativo: deve-se ficar frequentemente de olho nas temperaturas, nos horários corretos para colocar outros ingredientes necessários e também no tempo total de cada etapa da produção. Dessa forma, o cervejeiro deve ficar sempre perto da preparação e, como o processo pode levar até 5h, ao final do dia, o cansaço pode predominar.

Dessa forma, pensou-se em uma forma de automatizar boa parte do processo. Assim, o usuário pode apreciar a cerveja feita caseiramente, mas sem o trabalho e cansaço causados pelo produção manual. Nesse contexto, surge a MakeBeer. Uma máquina que tem como objetivo automatizar as partes mais cansativas do processo de fabricação de cerveja, garantindo temperaturas e tempos corretos em todas as etapas do processo.

Depois de estudos e diferentes ideias para o modelo dessa máquina automatizada, levando em consideração fatores adversos para o sabor final da cerveja e preço final do produto, chegou-se ao conceito de uma máquina composta por duas panelas, uma para mosturação e outra para fermentação. Essas panelas ficam uma ao lado da outra. O líquido entre as duas é passado, quando necessário, através de mini-bombas. Quando necessário, através de comandos do microcomputador, os lúpulos são inseridos na fervura no tempo exato conforme uma receita. A resistência esquenta na temperatura correta o líquido, garantindo assim mais exatidão ao processo.

Para esse feito, foram utilizados circuitos eletrônicos: fonte, controle dos motores e controle da resistência. A fonte, composta por um transformador de 30V e 5A, serve para alimentar o restante dos circuitos utilizados, gerando saídas de 6V e 15V. O controle dos motores é dado através de relés que, ao comando do microcomputador, conectam a carga à sua tensão necessária para funcionamento (seja 12V para mini-bombas ou 6V para os motores DC), causando o acionamento dos mesmos por um período determinado previamente. E por fim, o controle das resistências é feito através do controle PID em software, e em hardware por controle do disparo do triac. Tudo isso controlado pelo microcomputador Raspberry Pi.

Todos os testes individuais foram realizados com sucesso. Ao realizar o primeiro teste de integração, ou seja, com todos os circuitos e componentes conectados, a Raspberry Pi não suportou a quantidade de utilização dos pinos e queimou. Dessa forma, após mais estudos e nova tentativa, percebeu-se que esse microcomputador não aguentaria a tensão e corrente necessários para o projeto, apesar de todas as proteções feitas. Assim, concluiu-se

que, apesar do funcionamento individual dos circuitos, o projeto em geral não era viável utilizando esse microcomputador.

Dessa forma, ao continuar esse projeto no futuro, as melhorias que podem ser feitas são a troca de microcontrolador ou então a utilização de duas Raspberrys com comunicação, menor complexidade no código, maiores circuitos de proteção, mais formas de dissipação de calor e menor quantidade de pinos utilizados.

Referências

ATKINS, P. W.; JONES, L. *Princípios de Química: Questionando a Vida Moderna e o Meio Ambiente*. [S.l.]: Editora Bookman, 2009. 1048 p. Citado na página 29.

BHAGWAN. *The Basics of BIAB*. 2010. Disponível em: <http://www.biabrewer.info/viewtopic.php?f=25&t=194>. Citado na página 22.

BRIGGS, D. et al. *Brewing: Science and practice*. [S.l.]: Woodhead Publishing, 2004. 863 p. Citado 2 vezes nas páginas 21 e 22.

HOLANDA, A. B. de. *Dicionário Aurélio*. [S.l.]: Editora Positivo, 2002. 2272 p. Citado na página 17.

LAMB, F. *Automação Industrial na Prática*. [S.l.]: AMGH Editora, 2015. 376 p. Citado 3 vezes nas páginas 13, 46 e 47.

LIMA, C. B. de; VILLACA, M. V. M. *AVR e Arduino: Técnicas de projeto*. [S.l.]: Edição dos Autores, 2012. 632 p. Citado na página 41.

LIMA, G. E. *El Triac: electrónica, ciencia y tecnología*. [S.l.: s.n.], 2011. 22 p. Citado 2 vezes nas páginas 13 e 39.

MELEXIS. *Infra Red Thermometer in TO-39*. Estados Unidos, 2006. 35 p. Citado na página 40.

MIRON, A.; BROWN, D. R. *The Professional Bar & Beverage Manager's Handbook: How to Open and Operate a Financially Successful Bar, Tavern and Nightclub*. [S.l.]: Atlantic Publishing Company, 2006. 554 p. Citado na página 23.

NACHEL, M. *Homebrewing For Dummies*. [S.l.]: John Wiley & Sons, 2008. 432 p. Citado na página 17.

NOONAN, G. *Understanding Malt Analysis Sheets – How to Become Fluent in Malt Analysis Interpretation*. 2017. Disponível em: <https://www.morebeer.com/brewingtechniques/bmg/noonan.html>. Citado na página 21.

PESSOA, M. S. de P.; SPINOLA, M. de M. *Introdução à automação para engenharia e gestão*. [S.l.]: Elsevier - Campus, 2014. 352 p. Citado na página 40.

SILVA, D. *Como fazer a Carbonatação da Cerveja Artesanal*. 2015. Disponível em: <http://www.condadodacerveja.com.br/como-fazer-a-carbonatacao-da-cerveja-artesanal/>. Citado na página 23.

SILVA, S. F. Automação industrial via internet: Uma abordagem de software voltada à pequena empresa. Centro Universitário do Triângulo - Unit, 2000. Citado na página 17.

SMITH, G. D. *Beer, A Global History*. [S.l.]: Reaktion Books Ltd, 2014. 100 p. Citado 2 vezes nas páginas 13 e 19.

SYSTEMS, A. *Tratamento aeróbico*. 2017. Disponível em: <https://www.adisystemsinc.com/pt/tecnologias-1/tratamento-aerobio>. Citado na página 21.

Apêndices

APÊNDICE A – Código geral

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <unistd.h>
#include <errno.h>
#include <pthread.h>
#include <wiringPi.h>
#include <bcm2835.h>
#include <signal.h>
#include <lcd.h>

// Variaveis globais
double temp_panela1, temp_panela2;
int dimming1 = 128, dimming2 = 128;
double integral;
double pre_erro;
volatile int mark = 0;
int setpoint[4], tempo[6], unica_vez, limite, n, i, end = 1;
int fervura = 100;
int lcd;

#define ZERO_PIN1 0
#define AC_LOAD1 7
#define ZERO_PIN2 2
#define AC_LOAD2 1
#define AVG 1
#define LCD_RS  25               //Pino de register select
#define LCD_E   24               //Pino de Enable
#define LCD_D4  23               //Pino de dados 4
#define LCD_D5  22               //Pino de dados 5
#define LCD_D6  21               //Pino de dados 6
#define LCD_D7  14               //Pino de dados 7
#define BUT1 4                   //Primeira opcao de cerveja
#define BUT2 5                   //Segunda opcao de cerveja
#define BUT3 6                   //Terceira opcao de cerveja
#define BOMB1 27
#define BOMB2 28
#define MOTOR1 26
#define MOTOR2 29



// Prototipos
void trigger(int sig);
double calcular_temp();
void dimmer(int setpoint, int mode);
double calcular_pid(double dt, double max, double min, double Kp, double Kd, double Ki, double
    setpoint, double pv);
void zero_crossing1(void);
void zero_crossing2(void);
void *funcao_temperatura(void* param);
void *funcao_pid(void* param);

int main(void)
{
    // inicializa a wiringPi
    if (wiringPiSetup () < 0)
    {
        fprintf (stderr, "Erro ao inicializar a wiringPi: %s\n", strerror (errno));
        return 1;
    }

    pinMode(ZERO_PIN1, INPUT);
    pinMode(ZERO_PIN2, INPUT);
    pinMode(AC_LOAD1, OUTPUT);
    pinMode(AC_LOAD2, OUTPUT);
    pinMode(BUT1, INPUT);
    pinMode(BUT2, INPUT);
    pinMode(BUT3, INPUT);
    pullUpDnControl(BUT1, PUD_DOWN);
    pullUpDnControl(BUT2, PUD_DOWN);
    pullUpDnControl(BUT3, PUD_DOWN);
```

```
73      pinMode(BOMB1, OUTPUT);
74      pinMode(BOMB2, OUTPUT);
75      pinMode(MOTOR1, OUTPUT);
76      pinMode(MOTOR2, OUTPUT);
77      digitalWrite(AC_LOAD1, LOW);
78      digitalWrite(AC_LOAD2, LOW);
79      digitalWrite(BOMB1, HIGH);
80      digitalWrite(BOMB2, HIGH);
81      digitalWrite(MOTOR1, HIGH);
82      digitalWrite(MOTOR2, HIGH);
83      dimming1 = 128;
84      dimming2 = 128;
85      unica_vez = 0;
86      limite = 0;
87      n = 0;
88
89      lcd = lcdInit (2, 16, 4, LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7, 0, 0, 0, 0);
90
91      lcdPuts(lcd, "Bem vindo a MakeBeer!");
92      sleep(4);
93      lcdClear(lcd);
94      lcdPuts(lcd, "Selecione o tipo de cerveja desejada:");
95      sleep(2);
96      lcdClear(lcd);
97      while (digitalRead(BUT1) == 0 && digitalRead(BUT2) == 0 && digitalRead(BUT3) == 0)
98      {
99          lcdPuts(lcd, "1 - Lager Pilsen");
100         sleep(2);
101         lcdClear(lcd);
102         lcdPuts(lcd, "2 - American IPA");
103         sleep(2);
104         lcdClear(lcd);
105         lcdPuts(lcd, "3 - Porter");
106         sleep(2);
107         lcdClear(lcd);
108     }
109     if (digitalRead(BUT1) == 1)
110     {
111         setpoint[0] = 47;
112         setpoint[1] = 66;
113         setpoint[2] = 72;
114         setpoint[4] = 78;
115         tempo[0] = 60*60;
116         tempo[1] = 20*60;
117         tempo[2] = 10*60;
118         tempo[3] = 1;
119         tempo[4] = 45*60;
120         tempo[5] = 15*60;
121     }
122     if (digitalRead(BUT2) == 1)
123     {
124         setpoint[0] = 72;
125         setpoint[1] = 62;
126         setpoint[2] = 72;
127         setpoint[4] = 78;
128         tempo[0] = 60*60;
129         tempo[1] = 10*60;
130         tempo[2] = 5*60;
131         tempo[3] = 1;
132         tempo[4] = 75*60;
133         tempo[5] = 15*60;
134     }
135     if (digitalRead(BUT3) == 1)
136     {
137         setpoint[0] = 50;
138         setpoint[1] = 65;
139         setpoint[2] = 70;
140         setpoint[4] = 70;
141         tempo[0] = 60*60;
142         tempo[1] = 15*60;
143         tempo[2] = 15*60;
144         tempo[3] = 30*60;
145         tempo[4] = 50*60;
146         tempo[5] = 10*60;
147     }
148
149     // relaciona o trigger do pino ZERO_PIN1 a funcao zero_crossing para a primeira resistencia.
150     if (wiringPiISR (ZERO_PIN1, INT_EDGE_RISING, &zero_crossing1) < 0)
151     {
152         fprintf (stderr, "Erro ao inicializar a ISR1: %s\n", strerror (errno));
153         return 1;
```

```
154     }
155      // relaciona o trigger do pino ZERO_PIN2 a funcao zero_crossing para a segunda resistencia.
156     if (wiringPiISR (ZERO_PIN2, INT_EDGE_RISING, &zero_crossing2) < 0)
157     {
158         fprintf (stderr, "Erro ao inicializar a ISR2: %s\n", strerror (errno));
159         return 1;
160     }
161
162     pthread_t temp;
163     pthread_t pid;
164     pthread_create(&temp, NULL, &funcao_temperatura, NULL); // Cria a thread do sensor de temperatura
165     pthread_create(&pid, NULL, &funcao_pid, NULL); // Cria a thread do controle pid
166
167
168     pthread_join(temp, NULL); // Espera as threads finalizarem
169     pthread_join(pid, NULL);
170     system("sudo poweroff"); // Desliga a Raspberry
171
172 }
173
174 void trigger(int sig) // Funcaoo relacionada ao alarme e consequentemente ao timer
175 {
176         mark = 1;
177 }
178
179 double calcular_temp() // Faz os calculos de temperatura
180 {
181         char buf[6];
182         char i, reg = 7;
183         double temp=0, calc=0, atemp;
184         double temperatura_obj;
185
186         for(i=0;i<AVG;i++)
187         {
188                 bcm2835_i2c_write (&reg, 1);
189                 bcm2835_i2c_read_register_rs(&reg,&buf[0],3);
190                 temp = (double) (((buf[1]) << 8) + buf[0]);
191                 temp = (temp * 0.02)-0.01;
192                 temp = temp - 273.15;
193                 calc+=temp;
194                 sleep(1);
195         }
196
197         temperatura_obj=calc/AVG;
198         if (unica_vez == 0 && temperatura_obj > setpoint[0]) // Para parar de esquentar quando a
                panela chegar a primeira temperatura desejada
199         {
200                 limite = 1;
201                 unica_vez = 1;
202         }
203         return temperatura_obj;
204 }
205
206 void dimmer(int setpoint, int mode)
207 {
208         if (mode == 1) // Controla o dimming da primeira resistencia
209         {
210                 double inc = calcular_pid(0.1, 110, -110, 0.1, 0.01, 0.5, setpoint, temp_panela1); //
                      Calcula o PID
211                 dimming1 -= inc;
212                 if (dimming1 < 7)
213                         dimming1 = 7;
214                 if (dimming1 > 128)
215                         dimming1 = 128;
216         }
217         if (mode == 2) // Controle o dimming da segunda resistencia
218         {
219                 double inc = calcular_pid(0.1, 110, -110, 0.1, 0.01, 0.5, setpoint, temp_panela2); //
                      Calcula o PID
220                 dimming2 -= inc;
221                 if (dimming2 < 7)
222                         dimming2 = 7;
223                 if (dimming2 > 128)
224                         dimming2 = 128;
225         }
226 }
227
228 double calcular_pid(double dt, double max, double min, double Kp, double Kd, double Ki, double
      setpoint, double pv) // Calcula o PID
229 {
230         if (i == 0)
```

```
231         {
232                 integral = 0;
233                 pre_erro = 0;
234                 i = 1;
235         }
236         double erro = setpoint - pv;
237         double Pout = Kp * erro;
238         integral += erro * dt;
239         double Iout = Ki * integral;
240         double derivada = (erro - pre_erro) / dt;
241         double Dout = Kd * derivada;
242         double output = Pout + Iout + Dout;
243         if (output > max)
244                 output = max;
245         if (output < min)
246                 output = min;
247 
248         pre_erro = erro;
249         return output;
250 }
251 
252 void zero_crossing1(void) // Recebe o momento que chega a zero e dispara o trigger da primeira
        resistencia
253 {
254   // Calculo do angulo de disparo: 1 onda completa de 60Hz = 1/60 = 16,6ms
255   // Chega ao ponto zero em: (60Hz)-> 8,3ms (1/2 ciclo)
256   // 8,3ms=8300us
257   // (8300us - 8.33us) / 128 = 65 (Aprox)
258 
259   if (dimming1 >= 128)
260         return;
261   //printf("Dimming: %d\n", dimming);
262   int dimtime1 = (65*dimming1);    // Para 60Hz => 65
263   usleep(dimtime1);    // Espera o tempo dimtime para disparar o TRIAC
264   digitalWrite(AC_LOAD1, HIGH);   // Dispara o TRIAC
265   usleep(100);       // Delay do disparo do TRIAC
266   digitalWrite(AC_LOAD1, LOW);    // Desliga o TRIAC
267 }
268 
269 void zero_crossing2(void) // Recebe o momento que chega a zero e dispara o trigger da segunda
        resistencia
270 {
271   if (dimming2 >= 128)
272         return;
273   //printf("Dimming: %d\n", dimming);
274   int dimtime2 = (65*dimming2);    // Para 60Hz => 65
275   usleep(dimtime2);    // Espera o tempo dimtime para disparar o TRIAC
276   digitalWrite(AC_LOAD2, HIGH);   // Dispara o TRIAC
277   usleep(100);       // Delay do disparo do TRIAC
278   digitalWrite(AC_LOAD2, LOW);    // Desliga o TRIAC
279 }
280 
281 void *funcao_temperatura(void* param) // Thread da temperatura
282 {
283         bcm2835_init();
284         bcm2835_i2c_begin();
285         bcm2835_i2c_set_baudrate(25000);
286 
287         while (end)
288         {
289                 bcm2835_i2c_begin();
290                 bcm2835_i2c_setSlaveAddress(0x5a);
291                 temp_panela1 = calcular_temp();
292                 sleep(1);
293                 bcm2835_i2c_end();
294                 bcm2835_i2c_begin();
295                 bcm2835_i2c_setSlaveAddress(0x5b);
296                 temp_panela2 = calcular_temp();
297                 sleep(1);
298                 bcm2835_i2c_end();
299 
300                 printf("Temperatura do objeto 1 = %04.2f\n", temp_panela1);
301                 printf("Temperatura do objeto 2 = %04.2f\n", temp_panela2);
302         }
303 
304         return NULL;
305 }
306 
307 void *funcao_pid(void* param)
308 {
309         signal(SIGALRM, trigger);
```

```
310         dimming1 = 128;
311         dimming2 = 128;
312         i = 0;
313         while(limite != 1) // Esquenta a panela ate chegar a primeira temperatura desejada. Nao tem
                tempo definido
314         {
315                 dimmer(setpoint[0], 2);
316                 dimming1 = 128;
317         }
318         alarm(60*3.5); // Liga a bomba para transferir o conteudo da panela 2 para a panela 1
319         while(!mark)
320                 digitalWrite(BOMB1, LOW);
321         digitalWrite(BOMB1, HIGH);
322         mark = 0;
323         i = 0;
324         alarm(tempo[0]);
325         while(!mark) // Esquenta ate o primeiro tempo necessario
326         {
327                 dimmer(setpoint[1], 1);
328                 dimming2 = 128;
329         }
330         mark = 0;
331         i = 0;
332         alarm(tempo[1]); // Esquenta ate o segundo tempo necessario
333         while(!mark)
334         {
335                 dimmer(setpoint[2], 1);
336                 dimming2 = 128;
337         }
338         mark = 0;
339         i = 0;
340         alarm(tempo[2]); // Esquenta ate o terceiro tempo necessario
341         while(!mark)
342         {
343                 dimmer(setpoint[3], 1);
344                 dimming2 = 128;
345         }
346         mark = 0;
347         alarm(60*3.5); // Liga a bomba para transferir o conteudo da panela 1 para a panela 2
348         while(!mark)
349                 digitalWrite(BOMB2, LOW);
350         digitalWrite(BOMB2, HIGH);
351         mark = 0;
352         i = 0;
353         alarm(tempo[3]); // Ferve ate o tempo para colocar o primeiro lupulo
354         while(!mark)
355         {
356                 dimmer(fervura, 2);
357                 dimming1 = 128;
358         }
359         mark = 0;
360         digitalWrite(MOTOR1, LOW); // Libera o lupulo
361         sleep(1);
362         digitalWrite(MOTOR1, HIGH);
363         mark = 0;
364         i = 0;
365         alarm(tempo[4]); // Ferve ate o tempo para colocar o segundo lupulo
366         while(!mark)
367         {
368                 dimmer(fervura, 2);
369                 dimming1 = 128;
370         }
371         digitalWrite(MOTOR2, LOW); // Libera o lupulo
372         sleep(1);
373         digitalWrite(MOTOR2, HIGH);
374         mark = 0;
375         i = 0;
376         alarm(tempo[5]); // Ferve o tempo restante
377         while(!mark)
378         {
379                 dimmer(fervura, 2);
380                 dimming1 = 128;
381         }
382         mark = 0;
383         alarm(30); // Mensagem de processo finalizado
384         while(!mark)
385         {
386                 lcdPuts(lcd, "Processo finalizado! Deixe a cerveja esfriar");
387                 sleep(4);
388                 lcdClear(lcd);
389                 lcdPuts(lcd, "e depois adicione o fermento.");
```

```
390                     sleep(4);
391                     lcdClear(lcd);
392             }
393             end = 0; // finaliza a thread de temperatura
394
395             return NULL;
396     }
```

APÊNDICE B – Código dos motores

```
1  #include <wiringPi.h>
2
3  #define PINO_MOTOR 0
4  #define TEMPO 1
5
6  int main (void)
7  {
8    wiringPiSetup();
9    pinMode (PINO_MOTOR, OUTPUT);
10   while(1)
11   {
12     digitalWrite(PINO_MOTOR, HIGH);
13     sleep(TEMPO);
14     digitalWrite(PINO_MOTOR, LOW);
15     sleep(TEMPO);
16   }
17   return 0;
18 }
```

APÊNDICE C – Código das resistências

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <wiringPi.h>
#include <bcm2835.h>
#include <pthread.h>

// Variaveis globais
double skytemp;
int dimming = 128;
double integral;
double pre_error;

#define ZERO_PIN 0
#define AC_LOAD 2
#define AVG 1
#define tempteste 35

double calcular_pid (double dt, double max, double min, double Kp, double Kd, double Ki, double
    setpoint, double pv)
{
    int i = 0;
    if (i == 0)
    {
        integral = 0;
        pre_error = 0;
        i = 1;
    }
    double error = setpoint - pv;
    double Pout = Kp * error;
    integral += error * dt;
    double Iout = Ki * integral;
    double derivative = (error - pre_error) / dt;
    double Dout = Kd * derivative;
    double output = Pout + Iout + Dout;
    if (output > max)
        output = max;
    if (output < min)
        output = min;

    pre_error = error;
    return output;
}

/*
   Calculo do angulo de disparo: 1 onda completa de 60Hz = 1/60 = 16,6ms
   Chega ao ponto zero em: (60Hz)-> 8,3ms (1/2 ciclo)
   8,3ms=8300us
   (8300us - 8.33us) / 128 = 65 (Aprox)
*/
void zero_crossing(void)
{
    if (dimming >= 128)
        return;

    //printf("Dimming: %d\n", dimming);
    int dimtime = (65*dimming);    // Para 60Hz => 65
    usleep(dimtime);    // Espera o tempo dimtime para disparar o TRIAC
    digitalWrite(AC_LOAD, HIGH);   // Dispara o TRIAC
    usleep(100);       // Delay do disparo do TRIAC
    digitalWrite(AC_LOAD, LOW);    // Desliga o TRIAC
}

void *funcao_temperatura(void* param)
{
    char buf[6];
    char i, reg;
    double temp=0, calc=0, atemp;
    bcm2835_init();
    bcm2835_i2c_begin();
    bcm2835_i2c_set_baudrate(25000);
    bcm2835_i2c_setSlaveAddress(0x5a);
```

```
73
74
75      while (1)
76      {
77          calc=0;
78          reg=7;
79
80          for(i=0;i<AVG;i++)
81          {
82              bcm2835_i2c_begin();
83              bcm2835_i2c_write (&reg, 1);
84              bcm2835_i2c_read_register_rs(&reg,&buf[0],3);
85              temp = (double) (((buf[1]) << 8) + buf[0]);
86              temp = (temp * 0.02)-0.01;
87              temp = temp - 273.15;
88              calc+=temp;
89              sleep(1);
90          }
91
92          skytemp=calc/AVG;
93          printf("Temperatura do objeto = %04.2f (d: %d)\n", skytemp, dimming);
94      }
95
96      return NULL;
97  }
98
99  void *funcao_pid(void* param)
100 {
101     skytemp = 0;
102     dimming = 128;
103     while(1)
104     {
105         double inc = calcular_pid(0.1, 110, -110, 0.4, 0.01, 0.5, tempteste, skytemp);
106         dimming -= inc;
107         //printf("Inc: %f\n", inc);
108         //printf("Skytemp: %f\n", skytemp);
109         if (dimming < 7)
110             dimming = 7;
111         if (dimming > 128)
112             dimming = 128;
113         //printf("Dimming: %d\n", dimming);
114     }
115     return NULL;
116 }
117
118 int main(void)
119 {
120     // inicializa a wiringPi
121     if (wiringPiSetup () < 0)
122     {
123         fprintf (stderr, "Erro ao inicializar a wiringPi: %s\n", strerror (errno));
124         return 1;
125     }
126     pinMode(0, INPUT);
127     pinMode(AC_LOAD, OUTPUT);
128     digitalWrite(AC_LOAD, LOW);     // Desliga o TRIAC
129     dimming = 128;
130     integral = 0;
131     pre_error = 0;
132
133     // relaciona o trigger do pino 17 a funcao zero_crossing.
134     if (wiringPiISR (ZERO_PIN, INT_EDGE_RISING, &zero_crossing) < 0)
135     {
136         fprintf (stderr, "Erro ao inicializar a ISR: %s\n", strerror (errno));
137         return 1;
138     }
139
140     pthread_t temp;
141     pthread_t pid;
142     pthread_create(&temp, NULL, &funcao_temperatura, NULL); // Cria a thread do sensor de temperatura
143     pthread_create(&pid, NULL, &funcao_pid, NULL); // Cria a thread do controle pid
144
145     while(1);
146 }
```