DESENVOLVIMENTO DE SOFTWARE E CONTROLE PARA IMPLEMENTAÇÃO NA *NOODLE MACHINE*

Software Development and control for Implementation in Noodle Machine

Moraes, Augusto R.
Centro Universitário de Jaguariúna

COSTA, Lucas Ferrari de Carvalho
Centro Universitário de Jaguariúna

RESUMO: Com o crescimento e a popularização das *Vending Machines* no Brasil, acarretou-se um aumento significativo no ramo destas máquinas que ofereçam outros tipos de alimentos. Graças a isso houve uma intensificação, por parte do setor alimentício, para o desenvolvimento de meios que atendam esta nova demanda.

O presente artigo se trata do desenvolvimento de Software, o qual agrega valor em suas definições de funcionabilidade e caracteriza o nível tecnológico do sistema, apresenta-se desde os cálculos de capacidade e necessidade do uso dos recursos elétricos, eletrônicos e eletromecânicos para processamento de sinais até a sua composição final da estruturação do código implementado no embarcado (Arduino) para controle de todos os processos presentes na Noodles Machine.

Palavras-chave: Software; Desenvolvimento; Funcionabilidade; Controle.

Abstract: Considering the growth of vending machines in Brazil, the improvements and expansion of this sector are eminent. Due to this expansion, linked to the food business, the market needs to encompass these demands. However, to build these types of equipment, a pre-project must have been designed to be followed, seeking the best methods of development, and beyond that, ensure the users' safety.

The present article deals with the development of Software, which adds value in its definitions of functionality and characterizes the technological level of the system, it presents itself from the calculations of capacity and the need to use electrical, electronic, and electromechanical resources for signal processing until its final composition of the structuring of the code implemented in the embedded (Arduino) to control all the processes present in the Noodles Machine.

Keywords: Software; Development; functionality; Control.

INTRODUÇÃO

Devido a necessidade que os seres humanos têm de otimização do tempo em dias atuais, traz-se à tona a indispensabilidade de métodos mais rápidos e prático para a realização do preparo de alimentos. É notório que a redução destas e outras atividades ao longo do dia, pode proporcionar ao indivíduo um melhor aproveitamento do tempo para a realização de tarefas que necessitam de uma maior demanda.

Refletindo sobre esta crescente necessidade, que pode ser notada no dia a dia, é imprescindível que haja novas técnicas de preparo de alimentos. Impulsionado pelo aumento da utilização de *Vending Machine* por todo o Brasil, onde “A conveniência das *vending machines* movimentou, em 2014, R$1 bilhão de reais e a previsão de crescimento do segmento para 2015 é de 12%.” (Surek et al., 2016, p. 38), nota-se que o crescimento do uso destes tipos de equipamentos acontece por se tratarem meios práticos e rápidos para obtenção de alimento, porém estas em sua grande parte fornecem apenas os lanches mais comuns.

Avaliando estes pontos, percebe-se que a aplicação dos meios tecnológicos utilizado por *Vending Machine* são os métodos mais atrativos para as pessoas que buscam agilidade; portanto novas opções de produtos podem agregar ainda mais o crescimento deste tipo de negócio no país. Logo buscando uma ideia que se encaixa nestes pontos foi pensado em uma máquina de preparo autônoma de macarrão instantâneo, uma saída encontrada para estrategicamente trazer um novo modelo de negócio baseado

em preparo rápido e vendas ágeis de um produto popularmente consumido no mundo todo.

A *Noodle Machine* projetada para fornecer em até quatro sabores do produto, tem por finalidade para atingir os gostos mais diversificados, também conta com um sistema de dois reservatórios de água interligados entre si, fornecendo uma alimentação constante de água, garantindo assim uma rapidez do preparo. A implementação de um sistema de controle, responsável por manter a temperatura da água constante para o preparo e mais um sistema para realizar o controle da quantidade de líquido injetado no alimento, tem por objetivo fazer o cozimento completo do macarrão atingindo as condições necessárias para o consumo.

OBJETIVO

O objetivo principal deste, consiste no estudo, elaboração dos modelamentos de controle, lógicas de funcionamento e desenvolvimento de programas em linguagem C para Microcontroladores, com o propósito de atribuir valor característico de funcionabilidade através da aplicação de *Software* na *Noodles Machine*. Em suma, para atingir esse objetivo os métodos de desenvolvimento para Software utilizados foram extraídos exclusivamente de conceitos de metodologia Ágil e *Lean*, visando o mapeamento e simplificação dos processos, identificação de oportunidades para melhoria e alterações centralizadas durante a fase inicial do projeto, garantindo assim, maior robustez e eficiência em sua conclusão.

DESENVOLVIMENTO

Para a realização do desenvolvimento geral utiliza-se da ferramenta de *Lean*: Ciclo PDCA, a qual consiste em seguir ordens metódicas sequências dos grupos de atividades divididos nas etapas de: Planejamento, Execução, Verificação e Ação de feedback. Juntamente com o clico PDCA para amplificar os resultados é aplicado a metodologia *Scrum* com foco nos requisitos do projeto e melhorias em versões subsequentes.

Para realizar o controle de Software do presente projeto faz-se necessário fazer o uso de dois controladores separados, justificados

tecnicamente pela necessidade de processamento simultâneo de instruções e pela garantia do suporte para melhorias com o uso de entradas ou saídas adicionais. Optou-se pela utilização do microcontrolador Arduino Mega 2560 R3 e o Controlador Rex C100, pois ambos atendem as necessidades de processamento e possuem entradas, saídas digitais e analógicas de dados em sua composição de *Hardware*.

Para o uso dos controladores citados foi realizado o devido dimensionamento de entradas e saídas. Em relação a isso, para o Arduino Mega 2560 R3 faz-se necessário a utilização de trinta conexões de entradas ou saídas de dados digitais, atende-se a essa necessidade com o modelo escolhido, pois ele possui o máximo de cinquenta e quatro entradas ou saídas digitais e seis conexões de entradas analógicas.

O referido Arduino Mega 2560 R3 é o responsável pelo controle especificamente dos processos de preparo sequenciais, consequentemente, a interação direta com os processamentos de dados da escolha do usuário. Este por sua vez, controla e processa os dados dos seguintes componentes apresentados na **Tabela 1**, onde cada um dos componentes é respectivamente abreviado com um código de abreviação para relação das características de nomenclatura técnica e descrição.

Tabela 1 - Referências, descrições e códigos de abreviação (Arduino Mega).

Abreviação	Descrição	Tipo de dado	Entrada ou Saída
B4	Botão início do processo	Digital	IN
B5	Botão escolha do produto	Digital	IN
B3	Botão desliga	Digital	IN
SF	Sensor de Fluxo	Digital (pwm)	IN
SUS	Sensor Ultrassônico	Digital	IN
SNO	Sensor de Nível- Tanque 2/ Nível Operacional	Digital	IN
SIR	Sensor de obstáculo IR	Digital	IN
BT_um	Módulo relé Arduino - Bomba Tanque 1	Digital	OUT
BT_dois	Ponte H - Bomba Tanque 2	Digital	OUT
TCS	Sensor TCS3200	Digital	IN/OUT
MP	Motor de Passo (Easy Driver)	Digital (pwm)	OUT
PH	Ponte H (Motor DC- Trava Elétrica)	Digital	OUT
LA	LED Nível Alto Tanque 1	Digital	OUT
LM	LED Nível Médio Tanque 1	Digital	OUT
LB	LED Nível Baixo Tanque 1	Digital	OUT
LCD	Display LCD 16X2	Digital	OUT
VSO	Válvula solenoide 5v	Digital	OUT

Fonte: Desenvolvido pelo autor.

Outrossim, o Controlador Rex C100 será responsável pelo controle

especificamente da temperatura da água no reservatório 2. Este por sua vez, será o responsável por enviar e receber os sinais da malha de controle através da disposição das configurações de parâmetros do PID.

A priori para desenvolvimento da lógica geral de interfaceamento com o usuário se faz necessário definir o Fluxo de funcionamento geral, através do uso da ferramenta de lean: *Flowchart,* representado na **Figura 1**.

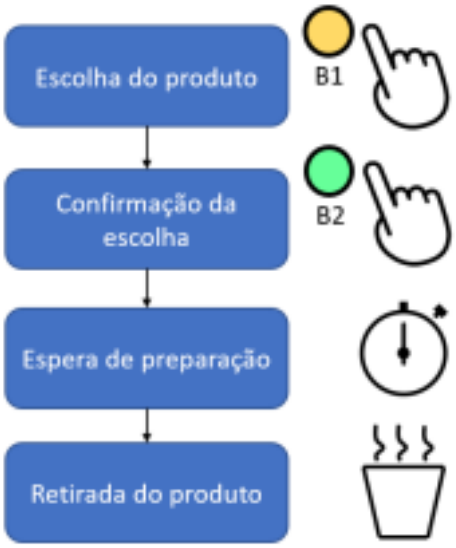


Figura 1: Flowchart: Fluxo do processo simplificado.

Fonte: Desenvolvido pelo autor.

Desse modo, possibilitou-se o exercício das lógicas de funcionamento subsequentes para realização do controle de Software do projeto, em suma essas que são definidas em formatos de fluxograma, esquemáticos visuais e tabelas verdade com o fito de auxiliar, direcionar e ser sobrescrevido posteriormente em instruções em linguagem C na plataforma Arduino IDE.

A princípio para o Arduino, que efetua o controle dos processos anteriormente descritos, neste é atribuído a lógica do fluxograma funcional geral disponibilizado na **Figura 2**, são definidas e enumeradas as “rotinas” de cada passo transitório do programa, sendo realizada essa separação para compactar as etapas mais complexas melhorando o entendimento do funcionamento geral do *Software* incorporado na máquina. As rotinas são amplamente empregadas nos desenvolvimentos de *softwares* mais complexos e robustos, temos o seu uso amplificado desse método de programação em

linguagens de alto nível, devido a tratativa em seu conjunto de funções com o fito de simplificar, clarificar o entendimento e organizar o código.

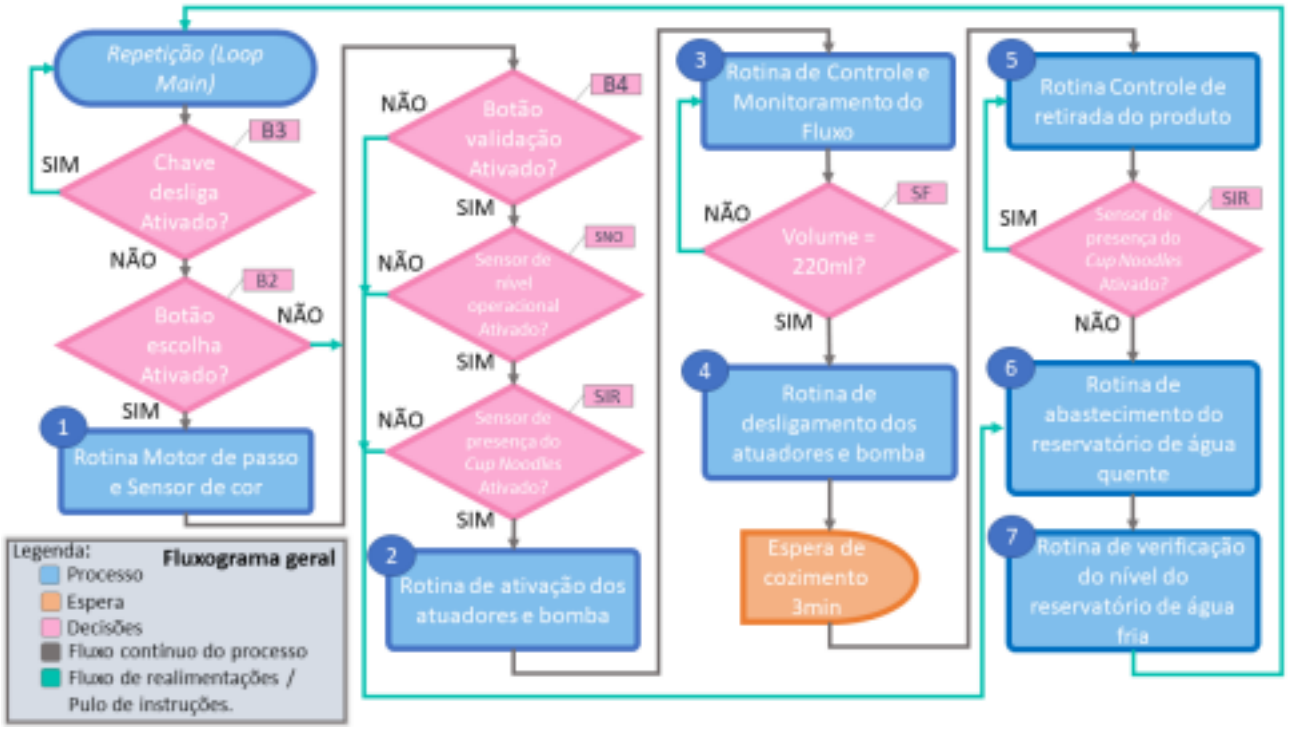


Figura 2: Fluxograma geral funcional dos processos atribuídos ao Arduino. **Fonte**: Desenvolvido pelo autor.

Uma outra boa prática adotada no fluxograma geral, foi a identificação do fluxo de realimentações e os momentos de pulo de instruções, devido a condições não satisfeitas em tomadas de decisões por uma cor destacada, assim facilitando a sua visualização.

A seguir, são tratadas todas as rotinas individualmente com as suas respectivas explicações ao que tange o desenvolvimento. A primeira rotina a ser evidenciada na **Figura 3** com seu fluxograma interno é a rotina responsável pelo controle de opções do usuário.

Essa rotina é executada toda vez que usuário pressionar o botão de seleção (B5), de forma que, executa o movimento de ¼ de volta em relação ao sistema de rotação do carrossel de opções, chama a função referente ao sensor de cor (TCS) para identificação dos níveis de RGB e faz comparações classificando especificamente as cores identificadas por opções de sabores, mostrando no *display* para o usuário.

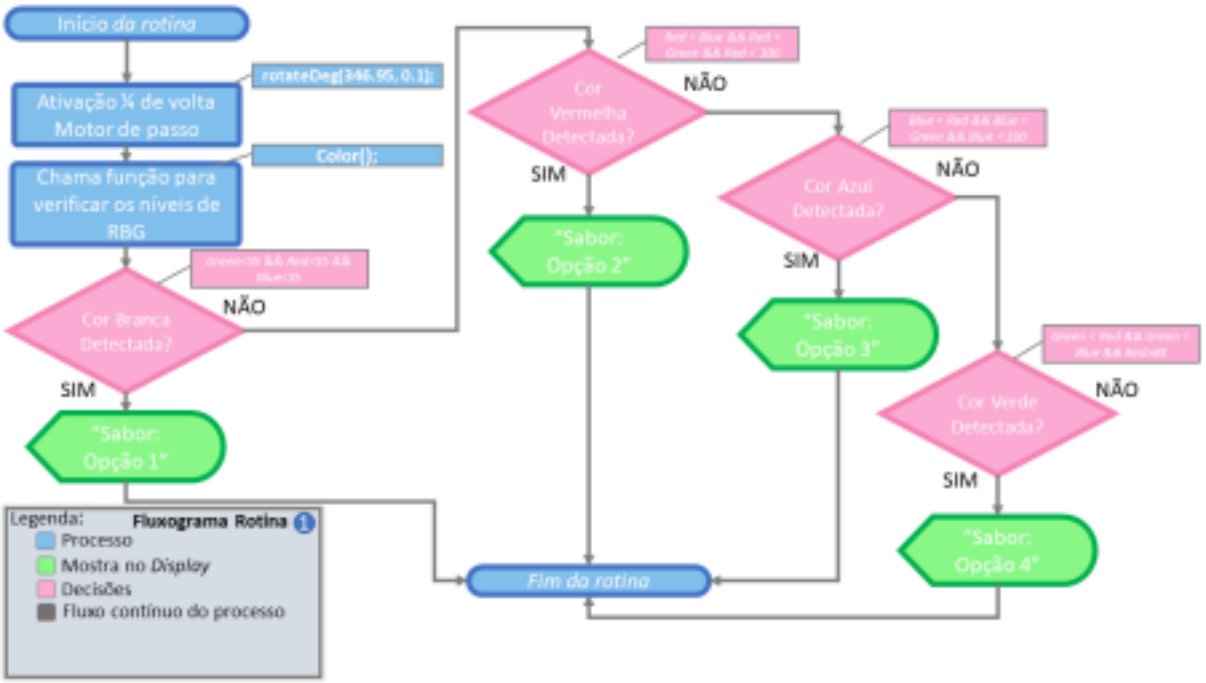


Figura 3: Fluxograma referente a rotina 1 dos processos atribuídos ao Arduino.
Fonte: Desenvolvido pelo autor.

Sobre o motor de passo, para calcular com precisão a quantidade de graus de rotação para executar ¼ da volta em relação ao carrossel é preciso aplicar a regra de relações de transmissão de engrenagem utilizando os valores de 185 dentes para a engrenagem maior e 45 para a engrenagem menor.

$$\textit{Relação de engrenagem} = {}^{185}/_{45} = 4{,}1111111.$$

$$\textit{Relação de engrenagem para ¼ de volta} = {}^{4{,}1111111}/_{4} = 1{,}02777777$$

$$\textit{Graus necessários para ¼ de volta em relação ao carrossel} = {}^{360º}/_{1.0277} = 350º.$$

Conforme no cálculo descrito o valor inicial utilizado como parâmetro na função *rotateDeg (350,0.1)*, assim sendo, o primeiro parâmetro os graus de movimento do motor de passo em relação ao eixo e o segundo definindo a velocidade, o segundo parâmetro escolhido com o valor de 0,1 é explicado pela necessidade de um alto torque devido a inercia do movimento, compensando assim a menor velocidade em favor de um consumo menor de corrente devido ao alto torque requisitado.

O controle de cores é composto por uma função chamada de *color ()*, essa por sua vez é responsável por enviar sinais de saída em combinação para

determinar a leitura dividida entre os 64 fotodiodos, habilitando 16 fotodiodos por vez referente a respectiva leitura da cor *RGB*, conforme mostrado na **Tabela 2.**

Tabela 2 - Tabela verdade para lógica controle do sensor TCS3200.

Pino sensor TCS (S2)	Pino sensor TCS (S3)	Fotodiodo
LOW	*LOW*	VERMELHO/ *RED*
LOW	*HIGH*	AZUL/ *BLUE*
HIGH	*LOW*	SEM FILTRO
HIGH	*HIGH*	VERDE/ *GREEN*
Referência Arduino:	TCS_C = S2	TCS_D = S3
HIGH =NÍVEL ALTO	*LOW* =NÍVEL BAIXO	

Fonte: Desenvolvido pelo autor.

A leitura do sinal é capturada pela sua saída *(OUT)* definida como "TCS_E", essa leitura é realizada entre cada condição de fotodiodo conforme anteriormente mostrado armazenando o valor lido de forma *pulseIn,* isto é, retorna à duração do pulso em microssegundos, essa duração é atribuída a uma variável que armazena como valor numérico inteiro a duração de acordo com a quantidade de pulsos em nível alto, dessa forma é possível extrair os níveis das três variáveis respectivamente *red, blue* e *green* para serem usados em comparações numéricas para cada cor. A **Figura 4** mostra o trecho do código da função referente a explicação acima.

```
//********Funções para indentificações dos niveis RBG
void color()
{
  //Rotina que le o valor das cores
  digitalWrite(TCS_C, LOW);
  digitalWrite(TCS_D, LOW);
  //count OUT, pRed, RED
  red = pulseIn(TCS_E, digitalRead(TCS_E) == HIGH ? LOW : HIGH); //armazena pulsos
  digitalWrite(TCS_D, HIGH);//troca condição
  //count OUT, pBLUE, BLUE
  blue = pulseIn(TCS_E, digitalRead(TCS_E) == HIGH ? LOW : HIGH); //armazena pulsos
  digitalWrite(TCS_C, HIGH); //troca condição
  //count OUT, pGreen, GREEN
  green = pulseIn(TCS_E, digitalRead(TCS_E) == HIGH ? LOW : HIGH); //armazena pulsos
}
```

Figura 4: Trecho retirado do código que contém a função *color().* **Fonte**: Desenvolvido pelo autor.

Por fim, com os valores extraídos de *RGB* é realizada as comparações de valores de acordo com os valores numéricos de *RGB* com cada cor ilustrado no fluxograma da **Figura 3.** Serão reconhecidas as cores azul, vermelha, verde e branca, cada cor é endereçada fisicamente com um produto de sabor

diferente, como exemplificado na **Figura 5**.

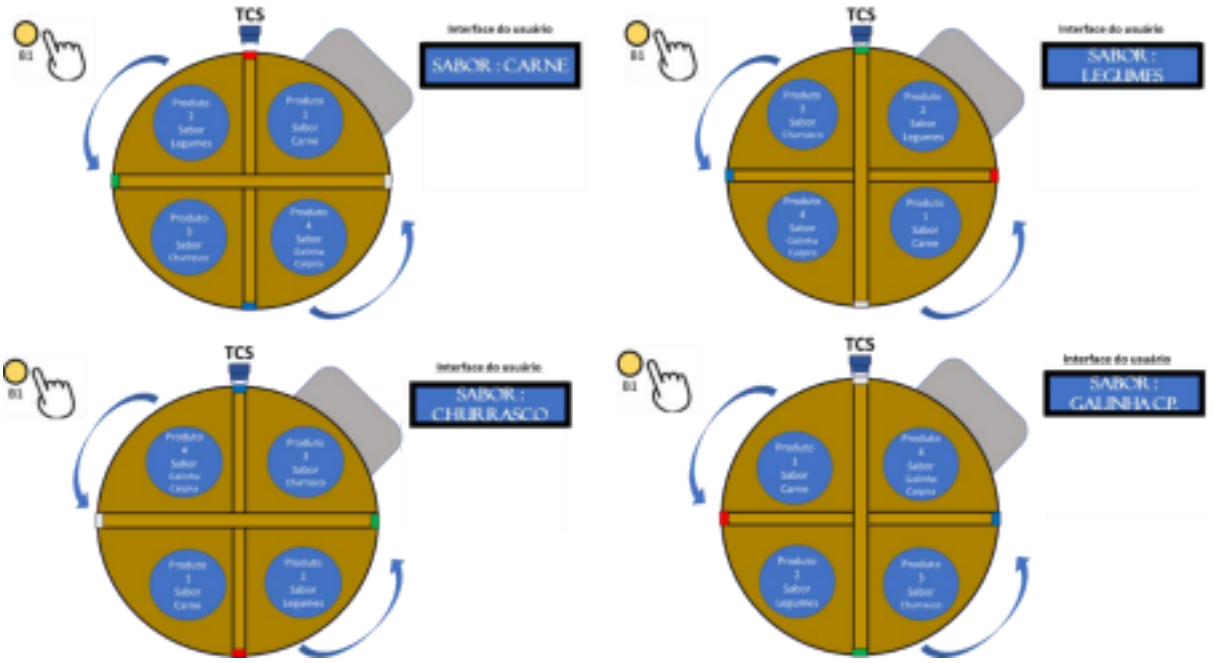


Figura 5: Esquemático de funcionamento do sensor TCS3200 no modelo físico.

Fonte: Desenvolvido pelo autor.

A próxima rotina, sendo a segunda, após a validação do usuário, compondo assim a primeira parte do início do processo de preparo do produto representado pela **Figura 6**, está é responsável pela configuração de ativação dos atuadores e bomba de injeção do líquido quente. É definido tempos de esperas mínimos entre as operações para evitar picos de corrente entre os atuadores garantindo assim, uma melhor performance ao custo de um tempo baixo adicionado a operação.

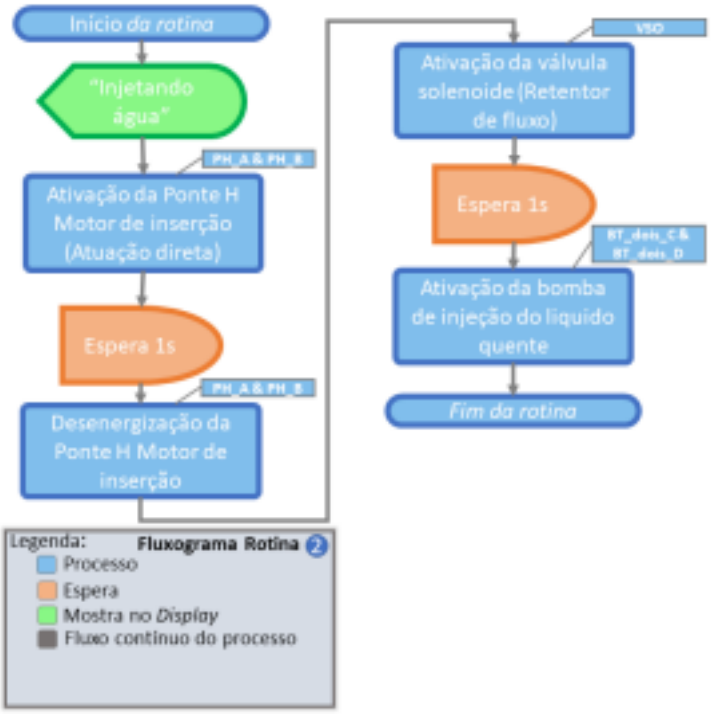


Figura 6: Fluxograma referente a rotina 2 dos processos atribuídos ao Arduino.
Fonte: Desenvolvido pelo autor.

Como uma outra boa prática de programação, se faz necessário fornecer ao display uma mensagem em específico do processo para identificar esse passo e os próximos, pois se faz eficiente em testes como um *breakpoint* de *log* para o programador, podendo ser mantido para mostrar o passo a passo ao usuário caso haja interesse dessa parte em saber o estado do processo de preparo. Essa prática perdura e poderá ser observada ao longo do artigo.

Em sequência, a próxima rotina a ser executada é a rotina 3, é a rotina essencial para o controle do processo de injeção do líquido ilustrado no fluxograma da **Figura 7**, pois é responsável por controlar a quantidade do volume injetado no Cup Noodles, definindo assim, o tempo necessário de permanência da ativação dos atuadores e bomba.

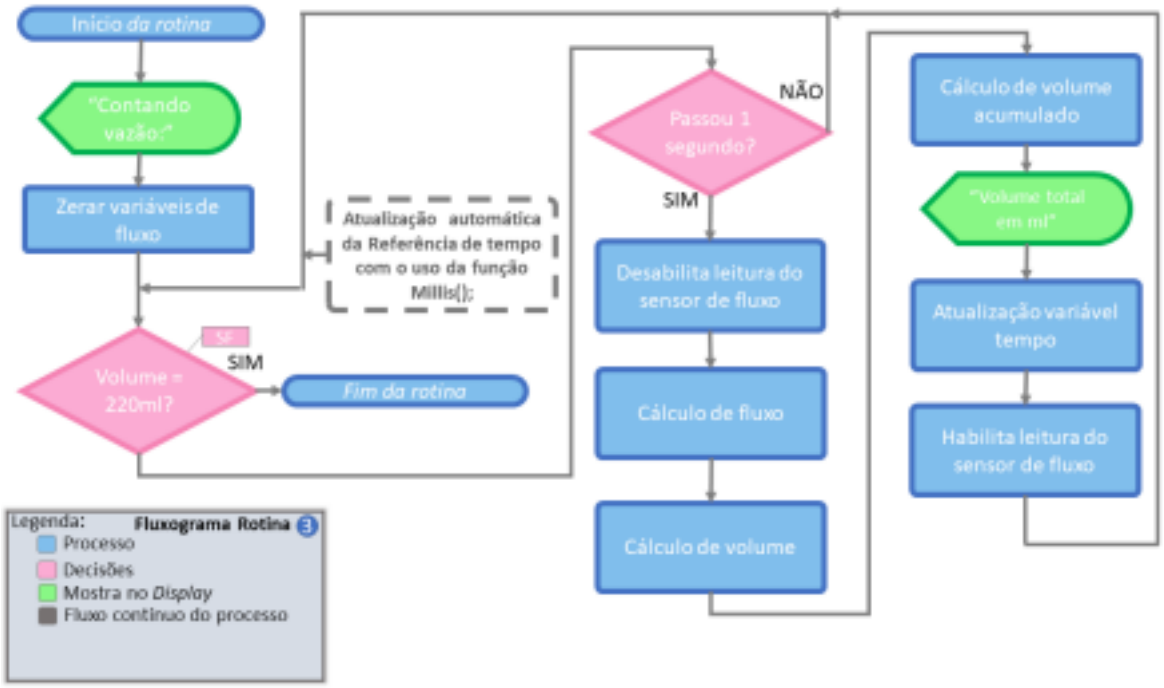


Figura 7: Fluxograma referente a rotina 3 dos processos atribuídos ao Arduino.

Fonte: Desenvolvido pelo autor.

Desse modo, o início da rotina é marcado por zerar as variáveis de controle de fluxo, sendo essas as seguintes variáveis com suas respectivas funções: fluxo, armazena o valor o valor do cálculo de fluxo; volume, armazena o volume em litros passado pelo sensor; volume_total, armazena o volume acumulado em litros passado pelo sensor para cada segundo de leitura; volume_total_ml, armazena o volume total acumulado em ml; contador, armazena os pulsos do sensor de fluxo; tempo_antes, armazena o valor de dentro da verificação de passagem do tempo de um segundo.

Após esse passo, as variáveis entram em uma repetição até que a variável volume_total obtenha o valor de 220ml, dentro dessa repetição existe uma verificação por segundo descrita com a seguinte lógica “millis () - tempo_antes> 1000”, está por sua vez, utiliza a função millis () do Arduino, que retorna o número de milissegundos passados desde que o programa iniciou (*unsigned long*). Caso a condição seja satisfeita é desligada a interrupção de leitura do sensor de fluxo (SF) e executado os cálculos de conversão dos sinais lidos pelo sensor.

Exemplificando o tempo de *millis* com o valor de “4000” e considerando a variável tempo_antes igual a zero, simulando que o Arduino esteja ligado

durante esse tempo em especifico a comparação pode ser ilustrada da seguinte forma abaixo:

$$Condição\ de\ entrada\ = 4000 - 0 > 1000$$

Logo, após essa verificação de início do programa a condição será satisfeita e dentro da condição tem-se o desligamento da interrupção responsável pela leitura do sensor de fluxo e os cálculos necessários de conversão conforme ilustrado na **Figura 8**.

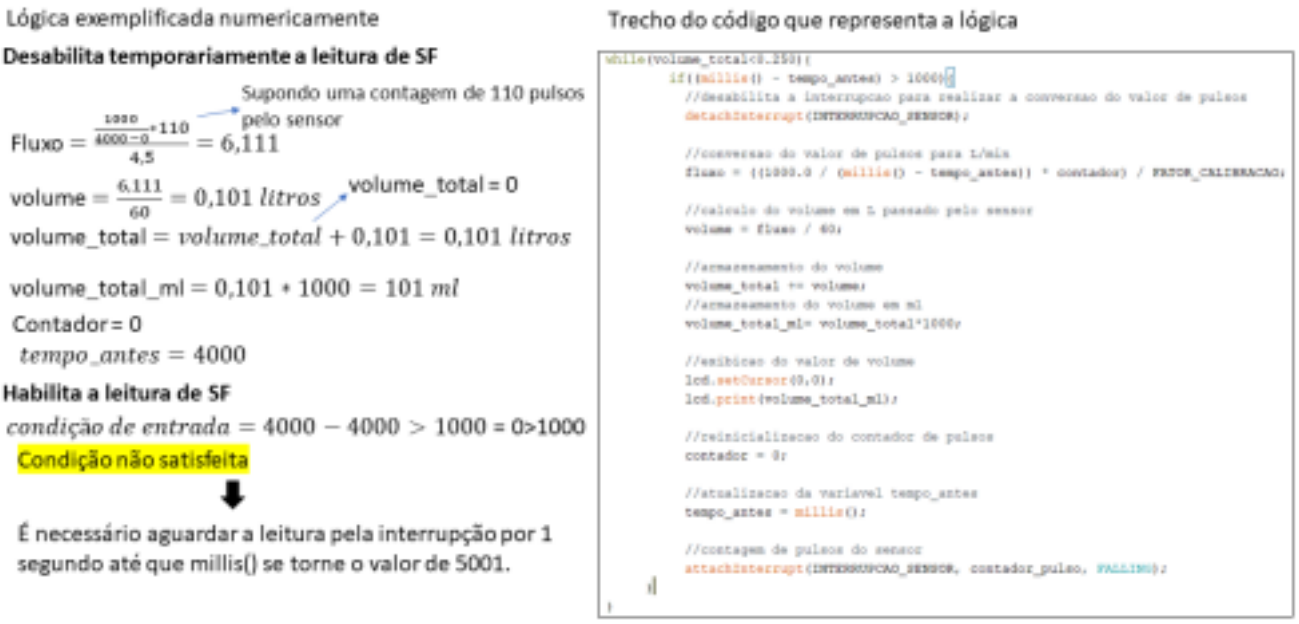


Figura 8: Fluxograma referente a rotina 3 dos processos atribuídos ao Arduino.

Fonte: Desenvolvido pelo autor.

A seguir, tem-se a rotina responsável pelo desligamento dos atuadores e bomba de injeção de líquido quente mostrada na **Figura 9**, essa rotina contém uma lógica parecida com a rotina de ativação desses mesmos itens, exceto pelo seu sequenciamento de operações, valores de tempos de espera, bem como também no caso da ativação do Motor de inserção que antecedendo sua desenergização é feito uma ativação por ação reversa para garantir o seu retorno em seu estado inicial.

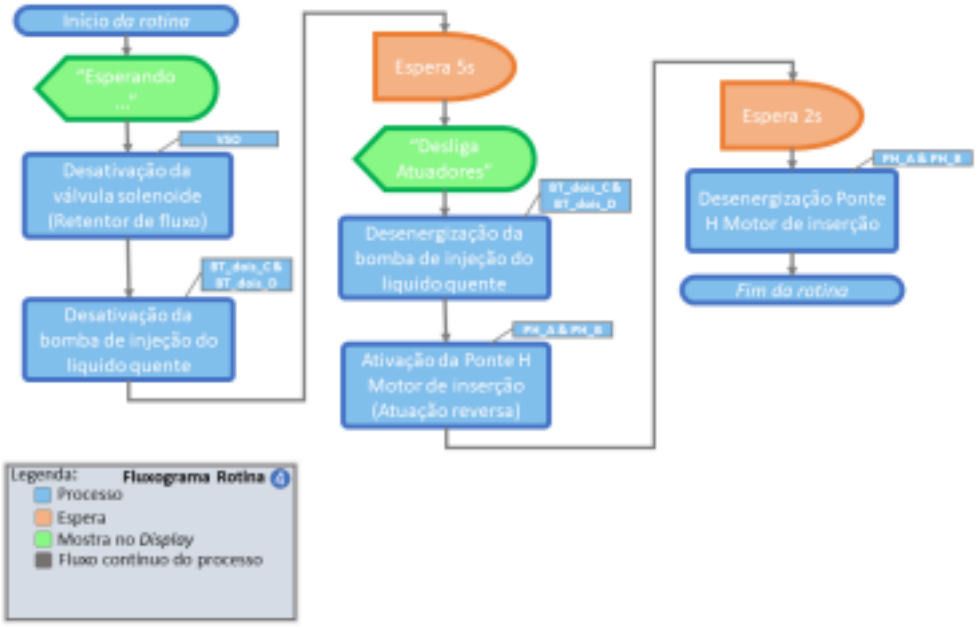


Figura 9: Fluxograma referente a rotina 4 dos processos atribuídos ao Arduino. **Fonte**: Desenvolvido pelo autor.

Em continuação, para realizar o controle de retirada do produto ilustrado na **Figura 10**, impedindo a sobreposição do processo de preparo faz-se necessário o uso de um intertravamento usando o sensor de presença do Cup Noodles, a rotina somente é finalizada quando é efetuada a retirada do produto, habilitando um novo processo sob as condições de validação necessárias mostradas no fluxograma geral da **Figura 2**.

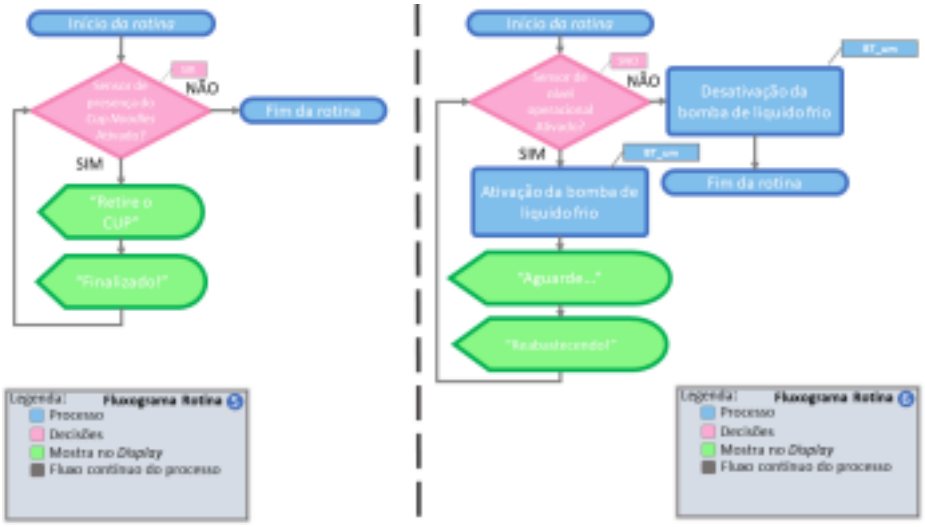


Figura 10: Fluxograma referente a rotina 5 e 6 dos processos atribuídos ao Arduino.

Fonte: Desenvolvido pelo autor.

Ainda sobre a **Figura 10**, também é ilustrado o fluxograma da rotina 6, rotina que não fica sob influência de condições, é repetida ciclicamente pelo

programa principal, essa rotina é responsável pelo abastecimento do reservatório de água quente, ativando o bombeamento do líquido do reservatório de água quente para, por fim, sempre manter o nível operacional do reservatório alvo. Todavia o sistema repetirá esse clico de abastecimento sempre que o sistema executar um preparo e não haverá riscos de iniciar o preparo sem água, já que o mesmo sinal do sensor de nível operacional é um requisito para iniciar o processo de preparo do produto.

Assim como a rotina 6 a rotina 7 tem a mesma tratativa de execução, com dependência apenas do programa principal para sua repetição, essa rotina é responsável por informar o nível do reservatório de água fria conforme a mostra na imagem da **Figura 11**, para assim, o administrador da máquina ter visibilidade da necessidade de reabastecimento.

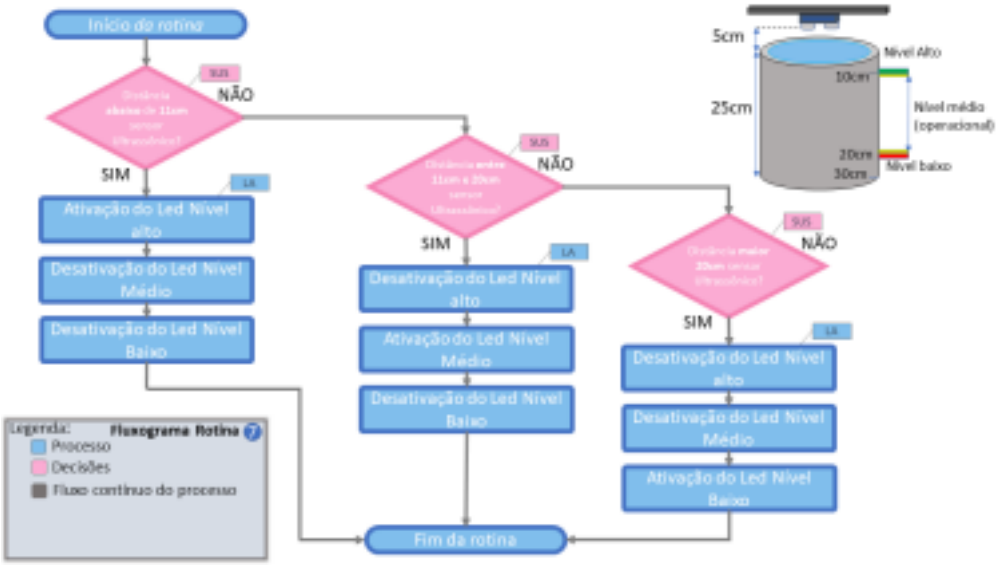


Figura 11: Fluxograma referente a rotina 7 dos processos atribuídos ao Arduino.

Fonte: Desenvolvido pelo autor.

Continuando, para a última rotina que compõem os processos incorporados pelo Arduino, faz-se necessário o uso de um sensor ultrassônico para medir a distância atual do líquido em relação ao sensor, obtendo assim, uma relação proporcional do nível do tanque, levando em consideração a diferença da distância do sensor em relação ao tanque. Define-se o nível médio (operacional) os valores entre os níveis alto e baixo, considerando assim, a necessidade de abastecimento apenas quando atingir o nível baixo, nesse

momento ainda deve existir minimamente 943ml conforme cálculo da **Figura 12** de água no reservatório garantindo o preparo anterior de 220ml até atingir esse determinado nível.

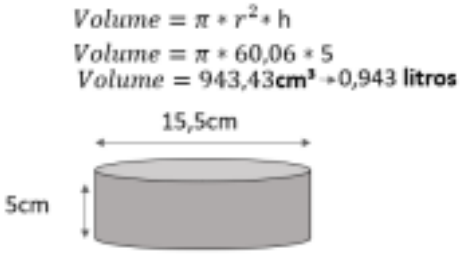


Figura 12: Cálculo de volume para nível baixo.

Fonte: Desenvolvido pelo autor.

Por fim, no **Anexo A** e **Anexo B** se encontra o código completo, o qual foi comentado e indentado para melhor organização.

Anexo A – Código em “. txt” (Arduíno 1).

https://drive.google.com/drive/folders/1nKiyZUIaZsiedV2NGtPS6_IkkYv8ACkM?usp=sharing

Fonte: Desenvolvido pelo autor

Anexo B – Código em “. Ino” (Arduíno 1).

https://drive.google.com/drive/folders/1E11LnaokH8NNCll8RLD_2kDd0V01bB4e?usp=sharing

Fonte: Desenvolvido pelo autor

Entretanto, para o REX C-100, para desenvolver o controle de temperatura do segundo reservatório, responsável por manter o líquido aquecido pronto para preparo, faz se necessário o processo de desenvolvimento e sintonia da malha de controle, sendo uma malha fechada de 1ºordem responsável pelo controle desse reservatório.

A princípio, como primeiro passo para o modelamento do sistema, faz-se necessário realizar a criação do esquemático da planta, para isso primeiro ocorre a definição dos instrumentos necessários a serem utilizados na malha, estes são um controlador PID REX C-100, relé de estado sólido 20ª, resistência

modelo mergulhão 500w e sensor termopar tipo K 400Cº. Após a definição e criação dos elementos representativos da malha o sistema é interligado conforme a norma ISA 5.1, foram criadas duas plantas do sistema, uma versão visual do sistema e uma versão completamente de acordo com a norma ISA 5.1, conforme a **Figura 13**.

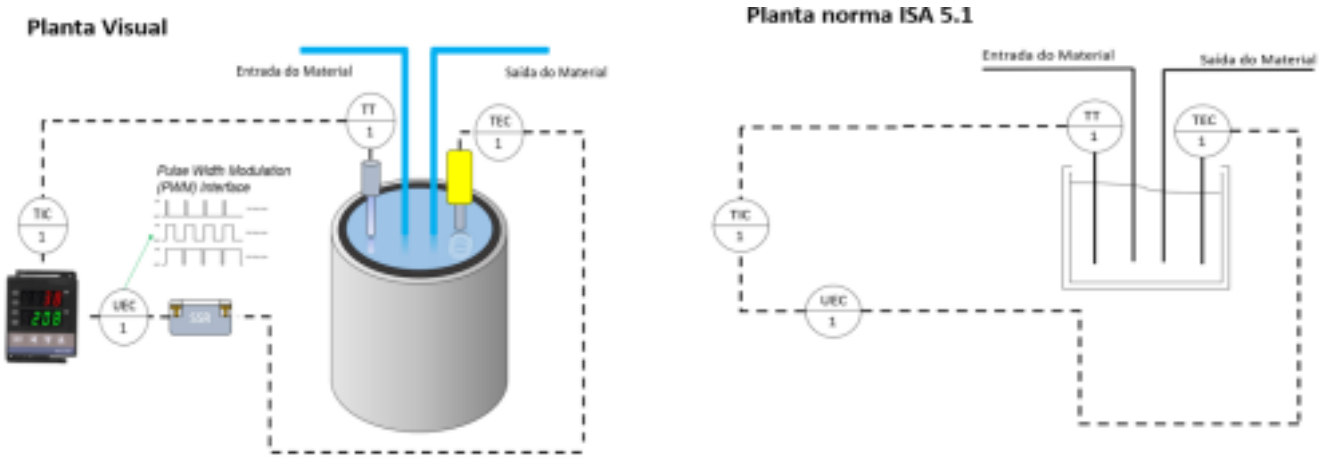


Figura 13: Planta do sistema de aquecimento do reservatório 2.
Fonte: Desenvolvido pelo autor.

De acordo com o modelo de planta desenvolvido é possível realizar a identificação das variáveis do processo e os seus distúrbios, conforme realizados na **Tabela 3**. A identificação das variáveis é uma etapa de suma importância, pois é a partir dessa identificação que é possível entender corretamente o funcionamento da planta melhorando a assertividade no desenvolvimento da malha de controle.

Tabela 3: Identificação das variáveis e distúrbio.

Variável	**Função**
MV (Variável manipulada)	Temperatura da resistência.
PV (Variável do processo)	Temperatura do líquido – Água (reservatório).
Distúrbio	Variação de temperatura ambiente, vazão de entrada e saída.

Fonte: Desenvolvido pelo autor.

A partir dessas informações, é estabelecida a malha de controle para o processo conforme a **Figura 14**. Através de um teste prático **Figura 15**,

utilizando a curva de resposta ao *set point* faz-se possível a descoberta por meio de cálculos analíticos sobre o gráfico, a definição da malha de controle e sua sintonia conforme disponibilizado detalhadamente no **Anexo C,** assim sendo possível a estabilização do valor da água no reservatório em 80ºC.

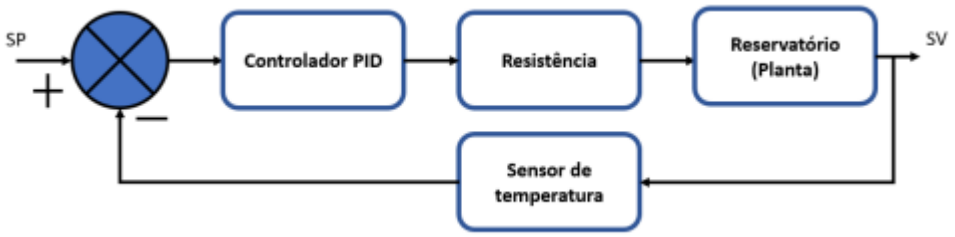
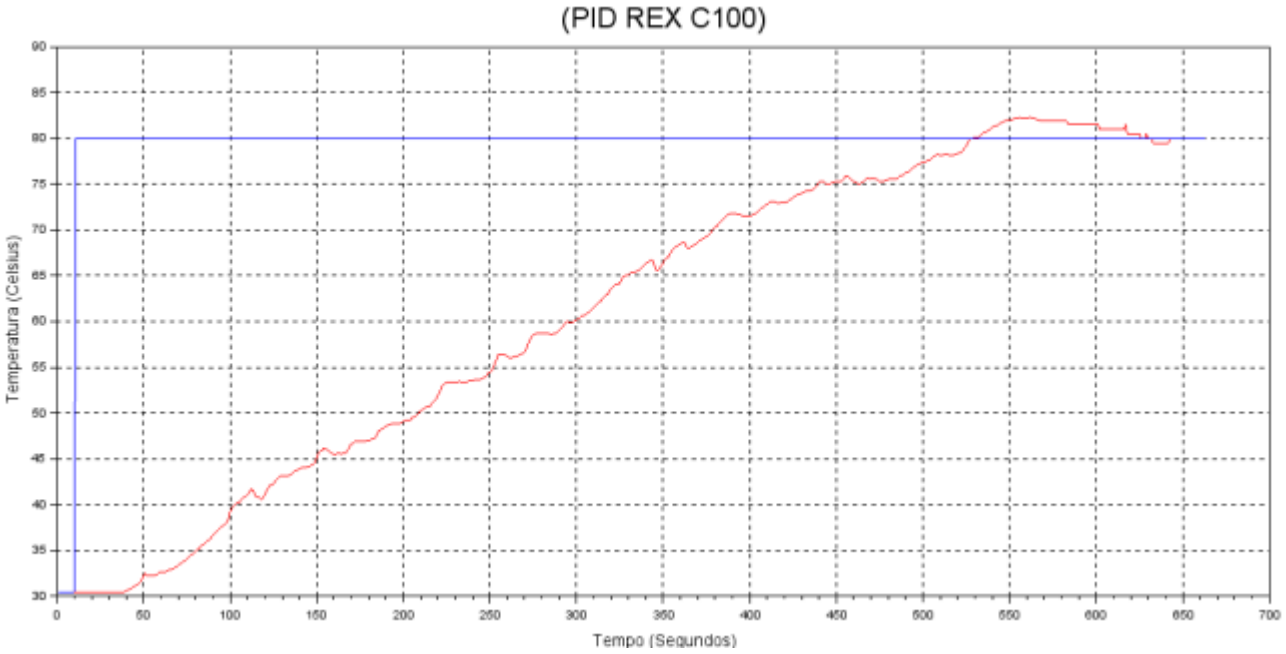


Figura 14: Malha de controle do processo de aquecimento.



Fonte: Desenvolvido pelo autor.

Figura 15: Curva de resposta ao *set point*, teste prático de aquecimento da resistência em função do tempo com distúrbios simulados.

Fonte: Desenvolvido pelo autor.

Com os parâmetros descobertos também é possível redesenhar a malha geral **Figura 16** de controle com as informações das variáveis completas, restando assim, a sua sintonia para chegar nos valores de PID.

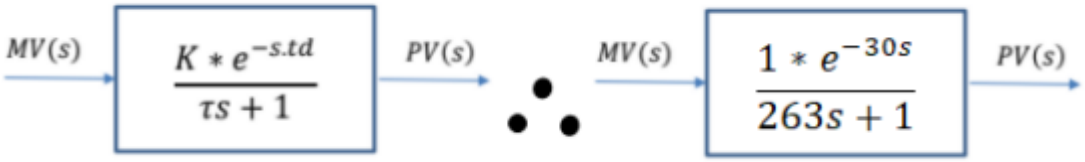


Figura 16: Malha geral de controle do processo de aquecimento.
Fonte: Desenvolvido pelo autor.

Anexo C – Sintonia e modelamento detalhado do PID.

https://drive.google.com/file/d/1me1UoVaRrbJdd0ZXae6cfpc6rm3Kf-t/view?usp=sharing

Fonte: Desenvolvido pelo autor

Após a sintonia, faz-se necessário efetuar a alteração dos valores de PID no REX C-100, com os seguintes parâmetros da **Tabela 4.**

Tabela 4: Definição de parâmetros PID.

Variável	Valor
P (Ação proporcional)	**12** (Arredondamento do valor de 11,7 - devido a limitação decimal do PID)
I (Ação Integrativa)	**80**
D (Ação Derivativa)	**20**

Fonte: Desenvolvido pelo autor.

RESULTADOS

Acerca do microcontrolador Arduino, a implementação do programa final no modelo físico foi realizada com êxito. Também foram obtidas informações importantes para ajustes finos, como foi no caso do controle do motor de passo, os valores previamente calculados sofreram ajuste de 3,1º em relação aos graus de rotação, o seu valor foi alterado de *rotateDeg (350,0.1)* para *rotateDeg (346,95.1)* apresentando uma resposta extremamente satisfatória.

Entretanto, outro ajuste importante foi adicionar a comunicação entre REX-C100 e Arduino, por meio da função alarme do REX-C100 indicando que

a temperatura desejada foi atingida, para permitir o funcionamento das demais rotinas apenas quando a temperatura estiver adequada na faixa de aproximadamente 80ºC. Além disso foi adicionado ao processo um comando para desativação da resistência em situação dos níveis de água abaixo dela, evitando superaquecimento e adicionando vida útil ao equipamento, entretanto, houve a inserção na rotina do preparo de um abastecimento padrão após a inserção da água, para ajudar na manutenção da água no reservatório, isso pode ser mais bem compreendido na **Figura 17**.

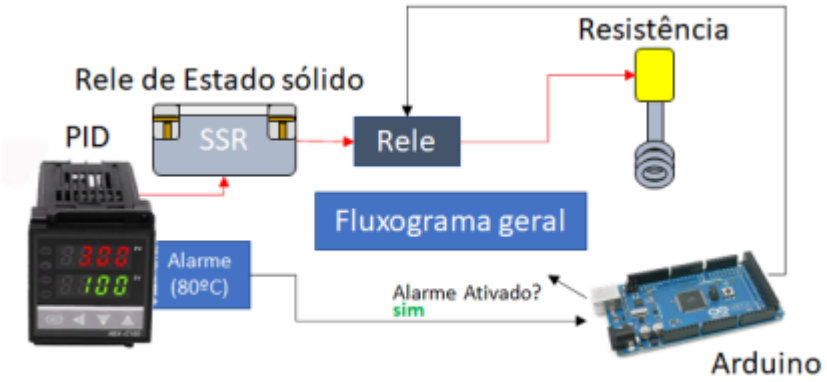


Figura 17: Esquema visual de controle e comunicação do PID e Arduino. **Fonte**: Desenvolvido pelo autor.

Por fim, como última alteração necessária para garantir o pleno funcionamento, o uso do sensor ultrassônico foi substituído por um circuito eletrônico, isso devido à grande variação dos valores na leitura do sensor ultrassônico causado pelo reflexo do reservatório de água fria, impedindo o funcionamento conforme programado no microcontrolador.

Atendendo a todos os requisitos funcionais, incluindo o PID (modelamento e sintonia) funcionaram corretamente, garantindo com sucesso a implementação de *Software* no aparato. Pode ser observado os *logs* de cada passo a passo dos processos de escolha e preparação na **Figura 18.**

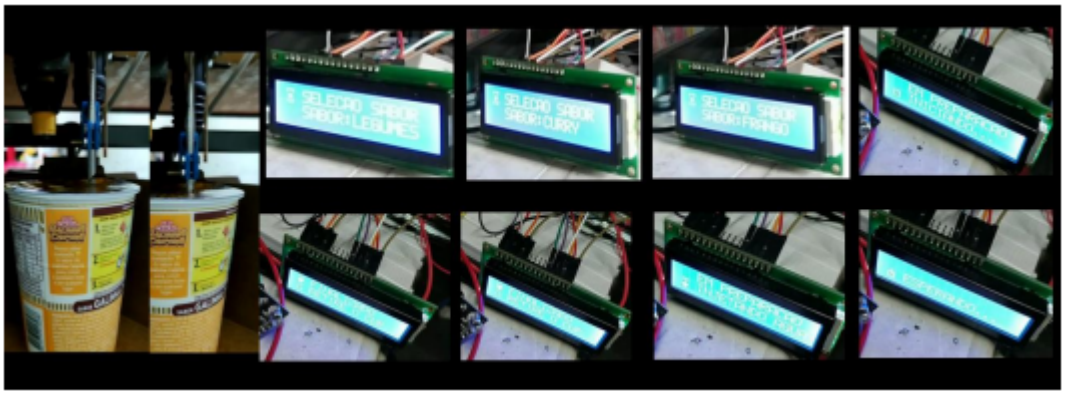


Figura 18: Fotos dos logs no display de cada passo do processo. **Fonte**: Desenvolvido pelo autor.

CONSIDERAÇÕES FINAIS

Contudo, pode-se notar que ao longo da realização do projeto houve pontos chaves que otimizaram os resultados, desde o planejamento de materiais até mesmo a interface de *Sotware* a ser utilizada. Os fluxogramas e a aplicação do pseudocódigo no início do projeto também foram ferramentas excelentes que ajudaram a prever erros, evitaram retrabalho e facilitaram as mudanças necessárias no código. Desde a fase inicial até a última versão do código foram feitas várias revisões devido a necessidade de melhorar o projeto e corrigir problemas encontrados durante a sua execução.

A integração do código subdivido em rotinas também foi um ponto crucial para agilizar a fase de testes e desenvolvimento, todas as rotinas foram testadas e individualmente tratadas para integração no código principal do programa, de forma que, para cada adição de código houvesse um teste em específico para verificação das funcionalidades conjuntas.

REFERÊNCIAS BIBLIOGRÁFICAS

SUREK, A. C, et all. Vending machines, uma análise do Mercado brasileiro. **Revista FAE,** Curitiba, Edição Especial, v. 1, p. 27-45, 2016.

PALHAIS, C. B. C. **Prototipagem: uma abordagem ao processo de desenvolvimento de um produto**. Tese (MESTRADO EM DESIGN DE EQUIPAMENTO COM ESPECIALIZAÇÃO EM DESIGNE DE PRODUTO) – Faculdade de belas-artes, Universidade de Lisboa, Lisboa, p. 29-34, 2015.

MATHIAS, I. M. **Computação: Algoritmos e Programação I**. Curso de

Licenciatura em Computação, Universidade Estadual de Ponta Grossa, Ponta Grossa. p. 1-177, 2017.

SOBRE O AUTOR

Augusto Rodrigues de Moraes

Graduando em Engenharia de Controle e Automação pelo Centro Universitário de Jaguariúna.

Assistente de Engenharia no setor de Engenharia Industrial pela FLEXTRONICS DO BRASIL LTDA.

E-mail para contato: gusto-moraes@hotmail.com