



Universidade Estadual de Campinas

Faculdade de Engenharia Mecânica

ES670 Projeto de sistemas embarcados

Prof Dr. Rodrigo Moreira Bacurau

Projeto do Controlador de temperatura utilizando sistemas embarcados



Nome:

Vitor Akio Isawa 178369

João Vitor Mendes 237881

Campinas
23 de Julho de 2024

Sumário

Sumário.....	1
1. Introdução.....	3
2. Documentação do sistema.....	4
Requisitos do projeto.....	4
Diagrama de Blocos.....	5
Principais componentes de Hardware.....	6
Diagrama de Camadas.....	6
Diagrama de Classes.....	7
Fluxogramas.....	7
3. Procedimento de sintonização do controlador.....	9
4. Manual de utilização.....	17
5. Problemas identificados e não resolvidos.....	18
6. Autoria dos códigos fornecidos.....	19

1. Introdução

Este relatório irá conduzir a criação de um sistema embarcado de controle de temperatura preciso e eficiente, utilizando um microcontrolador STM32 e o software CubeMX. Este projeto guiará passo a passo o processo de desenvolvimento, desde a concepção até a implementação final.

A jornada começa com a criação do código do sistema, utilizando a poderosa plataforma CubeMX. Através de uma interface amigável, será configurado os periféricos do microcontrolador STM32, definindo as funcionalidades e recursos que darão vida ao projeto.

Em seguida, embarcaremos na delicada tarefa de integrar o código com os periféricos do microcontrolador. Cada periférico, como sensores de temperatura, cooler, controle de temperatura e interfaces de comunicação, será cuidadosamente integrado ao sistema, garantindo uma sincronia perfeita de funcionalidade.

Interface UART:

A interface UART será sua chave para o controle remoto. Através de comandos seriais enviados por um computador, você poderá ajustar as configurações do sistema, monitorar a temperatura em tempo real e até mesmo controlar o cooler de forma remota.

A Comunicação com o Cooler:

O cooler, um componente crucial do sistema, será controlado manualmente ou automaticamente. O código será responsável por determinar a velocidade ideal de rotação do cooler, otimizando o processo de aquecimento e resfriamento.

Monitoramento Constante:

A temperatura do resistor será monitorada incessantemente, utilizando sensores precisos e confiáveis. O código interpretará os dados dos sensores, fornecendo informações precisas sobre a temperatura atual e permitindo ajustes finos no processo de controle.

Interface Local: Uma Janela para o Controle:

Uma interface local, intuitiva, será desenvolvida para que você possa interagir diretamente com o sistema segundo o manual de instruções descrito no relatório. Através de botões, displays e outros elementos, você poderá definir a temperatura desejada, monitorar a temperatura em tempo real e ajustar as configurações do sistema.

2. Documentação do sistema

Requisitos do projeto

Em primeiro momento será realizado um levantamento prévio dos requisitos do projeto, diferenciando-os em funcionais e não funcionais. Para definir as respectivas diferenças, utilizamos o seguinte conceito:

“Requisitos funcionais são aqueles que definem as funcionalidades do sistema de forma completa e concisa.[...]. Requisitos não funcionais se referem às restrições do projeto que afetam uma ou mais de suas funcionalidades, tais como restrições de tempo, utilização de recursos, consumo de energia, custo, ambiente de operação, confiabilidade, desempenho, usabilidade, testabilidade, segurança, escalabilidade etc. ”

Essa etapa será realizada de forma iterativa, ou seja, será atualizada ao longo do desenvolvimento dos dados apresentados no projeto, a fim de que seja coberta uma parte maior das necessidades demandadas pelo mesmo. Então, a tabela com os requisitos “finais” do projeto é dada a seguir:

Requisitos Funcionais	Requisitos não funcionais
O sistema deverá possuir uma interface local que utilize os botões, leds, display lcd e teclado matricial para interação com o usuário	Interface local deverá mostrar no LCD a temperatura, setpoint, potencia do cooler e heater no modo inicial
Interface local de exibição de dados do sistema, seja na função de exibição do controlador ou seja no modo de configuração.	Interface local deverá mostrar um menu de configuração no qual é possível ver e configurar os parâmetros como ganhos do controlador PID (K_p , K_i e K_d), temperatura de referência, potência no heater e no cooler, modo com controle ativo e desativo na interface local.
Existir um modo manual e um automático para o controle de temperatura	O teclado matricial deverá ser utilizado para inserção de valores no menu de configuração
O sistema deverá manter a temperatura do resistor do kit em um valor determinado pelo usuário	O buzzer deverá apitar quando algum valor for alterado via interface local
A interface local deverá mostrar as variáveis de ambiente pelo LCD.	O sistema deve aquecer o mais rápido possível
O sistema deverá permitir o controle por comando seriais em uma interface UART	Temperatura máxima 90°C
O usuário poderá inserir o valor estipulado para a	Será utilizado um controle PID para a

temperatura no sistema	potência do resistor
O cooler deverá ser utilizado	O modo de configuração deve ser acessado pelos pushbuttons na interface local. Para entrar no modo de configuração, deverá ser apertado durante três segundos o pushbuttons "enter"
Deve ser possível monitorar a velocidade de rotação do cooler	Os Leds deverão mostrar o status atual da temperatura em relação ao setpoint, ou seja, mostrará se está acima ou abaixo do setpoint
O sistema deverá exibir a temperatura atual	Uso dos botões para controlar a interface local
	O Tacômetro será utilizado para a medição de velocidade do Cooler
	O controle de temperatura se trata de um PID e utilizará um sinal de PWM para controlar o <i>heater</i> e o cooler para casos de temperatura acima da temperatura de referência.
	O overshoot máximo deve ser de 2°C.
	O sistema deve aquecer o mais rápido possível.
	O Cooler deverá ser acionado para garantir que o overshoot seja o menor possível

Tabela 2.1: Requisitos do projeto

 Tabela de Requisitos

Diagrama de Blocos

Desenvolvemos um diagrama de blocos para representar o sistema, foram identificados os componentes essenciais e os módulos utilizados do microcontrolador. Este diagrama proporciona uma visão mais clara da arquitetura do sistema, destacando a interconexão entre os diferentes elementos. Os componentes do sistema, desde sensores até Leds, estão posicionados e conectados aos respectivos pinos do microcontrolador. Este diagrama serve como um guia visual abrangente para o desenvolvimento e implementação bem-sucedidos do sistema proposto.

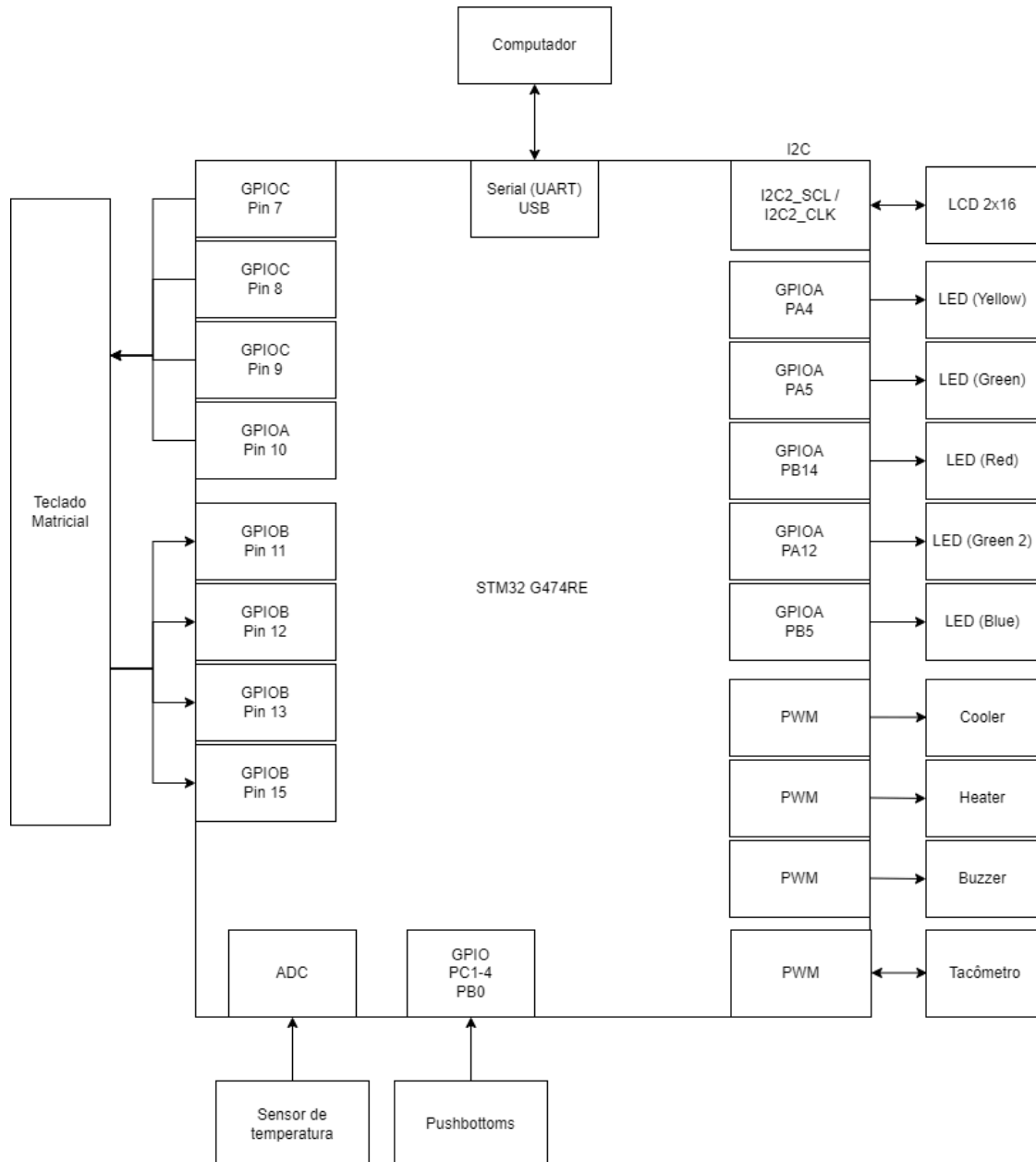


Figura 2.1: Diagrama de Blocos

[Link] [Diagrama de Blocos](#)

Principais componentes de Hardware

A seguir será apresentado a relação de componentes principais de hardware, segundo os equipamentos já instalados no kit e seus periféricos.

Principais Componentes de Hardware

Diagrama de Camadas

Após realizar a descrição dos componentes iremos representar o sistema em um diagrama de camadas, que é uma forma visual de se analisar visando uma arquitetura em camadas, seguindo uma hierarquia de processos e relacioná-las através das chamadas de funções (APIs). Esse modelo tem uma vantagem na análise das funções sob uma implementação orientada a objetos, podendo ser mais interpretativa e visual conforme a necessidade de análise. A estrutura que projetamos para o nosso sistema possui o seguinte layout:

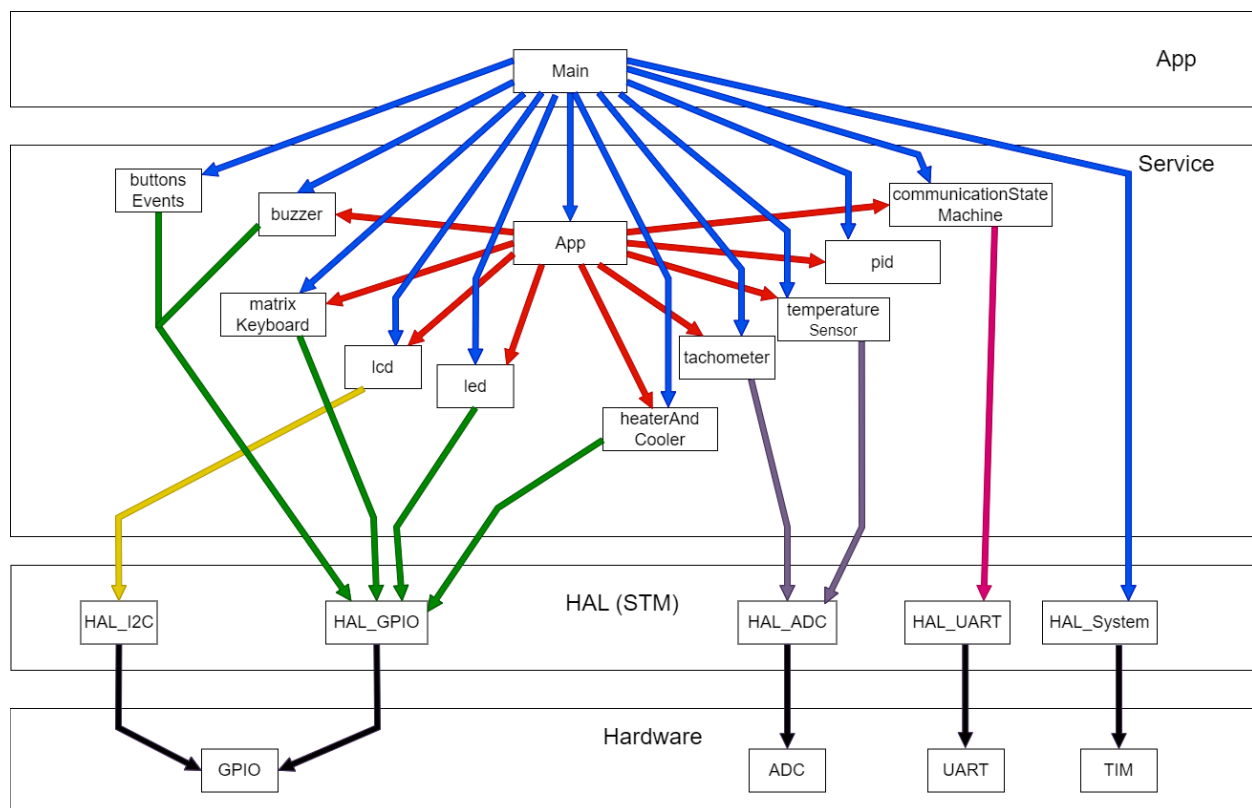


Figura 2.2: Diagrama de Camadas

Diagrama de Classes

Tendo o diagrama de camadas em mãos, vamos então representar o diagrama de classes, que é definido por uma representação visual que escreve a estrutura estática de um sistema de software, organizando suas classes em diferentes camadas ou níveis de abstração. As classes são representadas por um diagrama UML, contendo o nome da classe, seus atributos e seus métodos. Esse tipo de diagrama é utilizado para compreender a organização do sistema e sua distribuição de funcionalidades entre as camadas, facilitando o processo de design, implementação e manutenção de software. Portanto foram feitos alguns diagramas de classe representando algumas das partes do código, esses podem ser vistos a seguir.

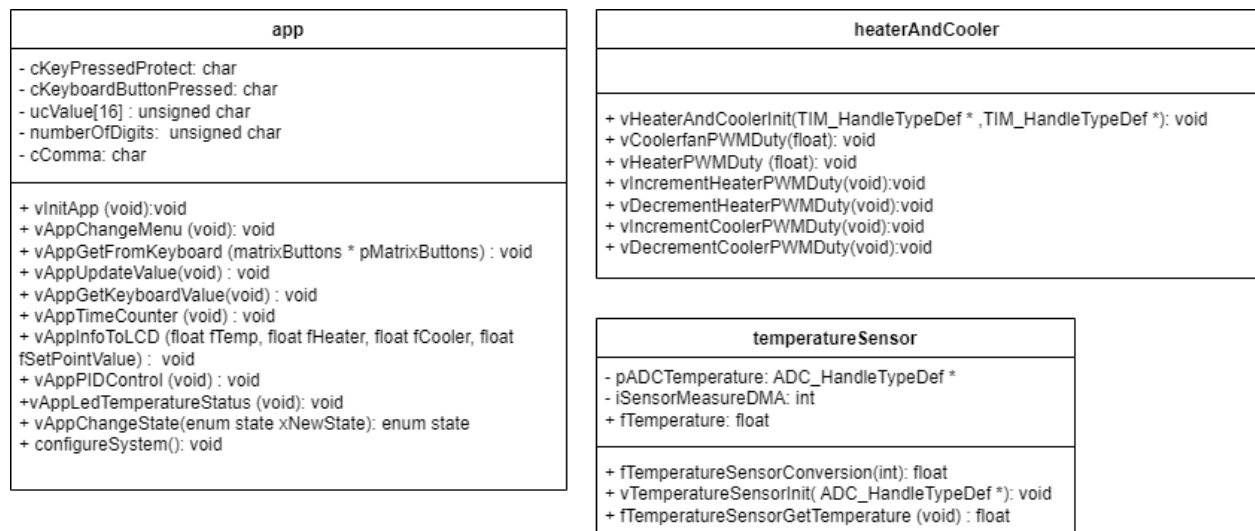


Figura 2.3: Diagrama de Classes

Fluxogramas

Conforme foram detalhadas as funções, construímos os fluxogramas para o funcionamento do sistema. Em primeiro instante foi estruturado um fluxo para os comandos da main, inicializando o sistema e os recursos necessários para a execução e então iremos exemplificar algumas das funções que serão implementadas no nosso controlador. Esta última parte será melhor desenvolvida ao longo da criação das rotinas e recursos das funções, conforme a codificação do controlador foi sendo estruturada.

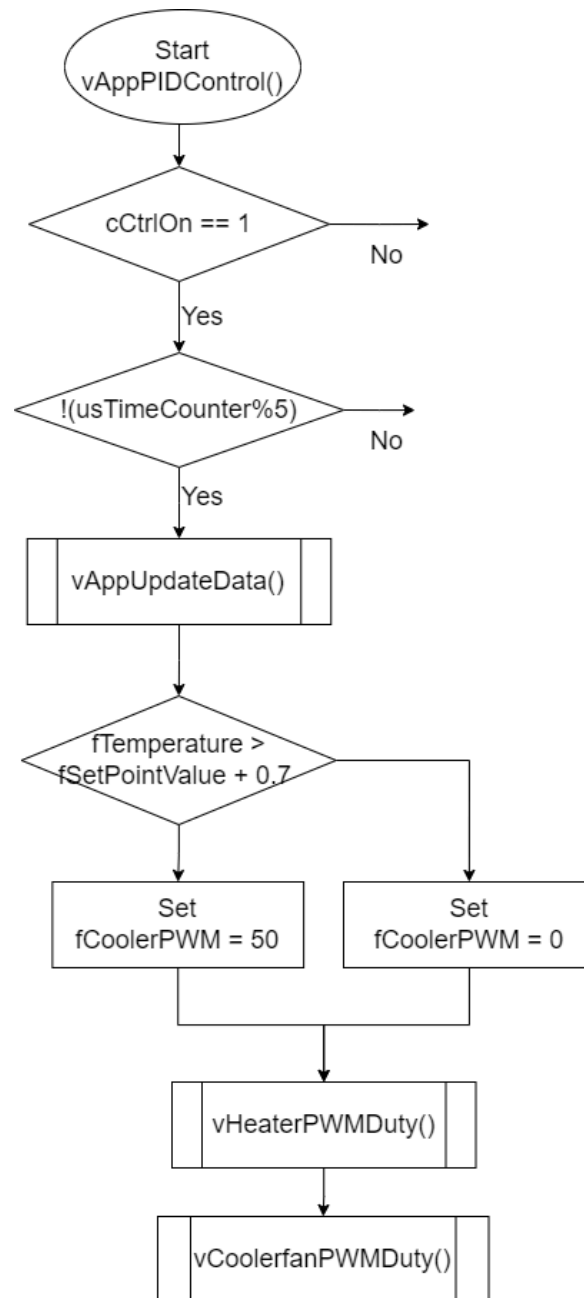


Figura 2.4: Fluxograma da função vAppPIDControl

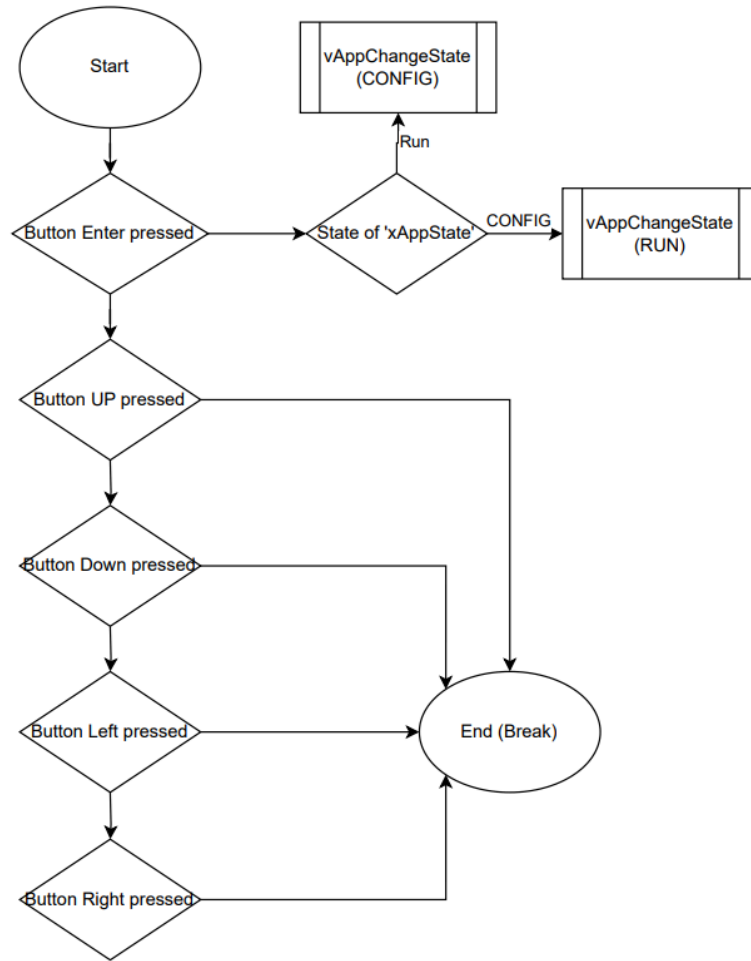


Figura 2.5: Fluxograma da função `vButtonsEventCallback3sPressedEvent`

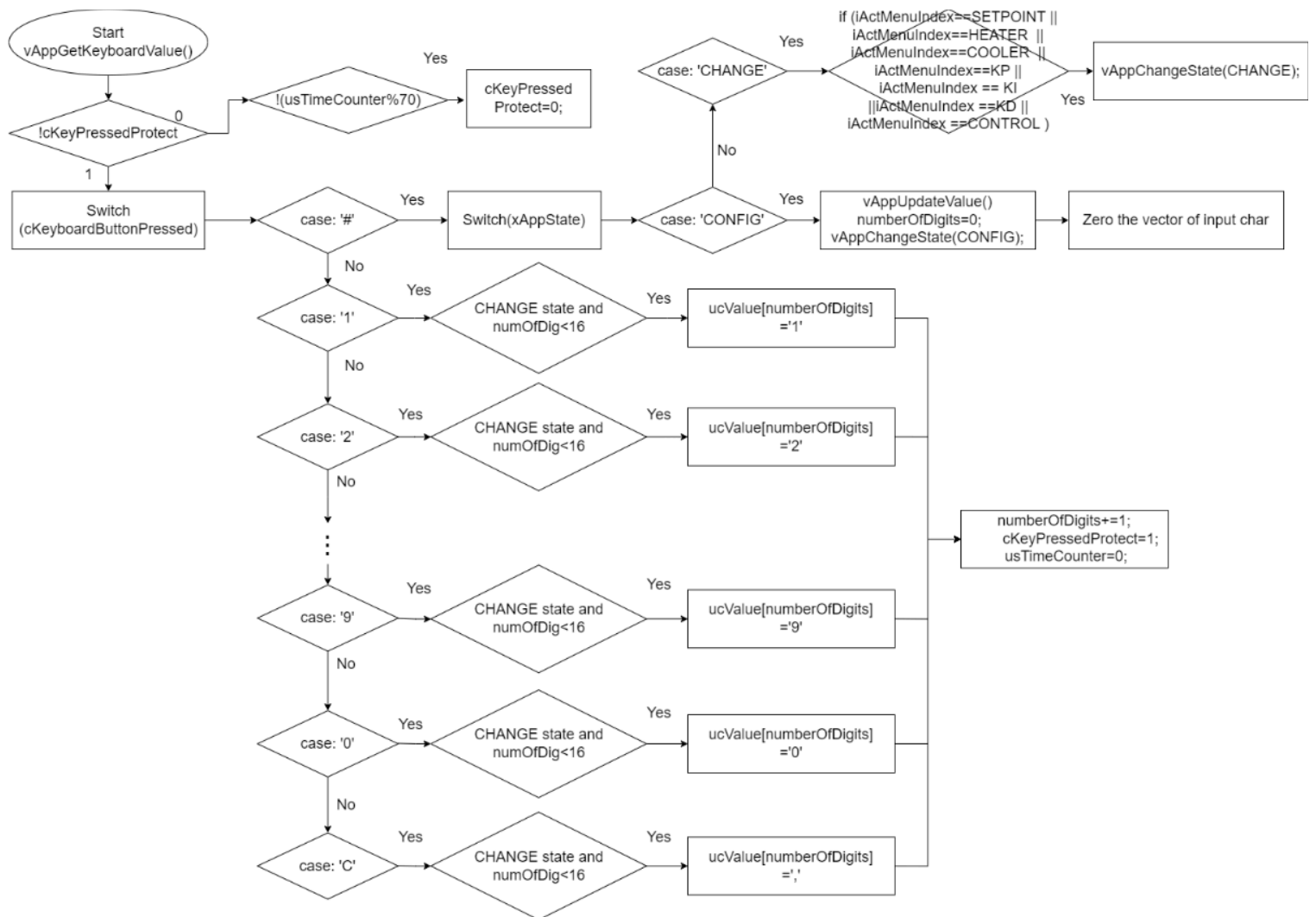


Figura 2.6: Fluxograma da função vAppGetKeyboardValue()

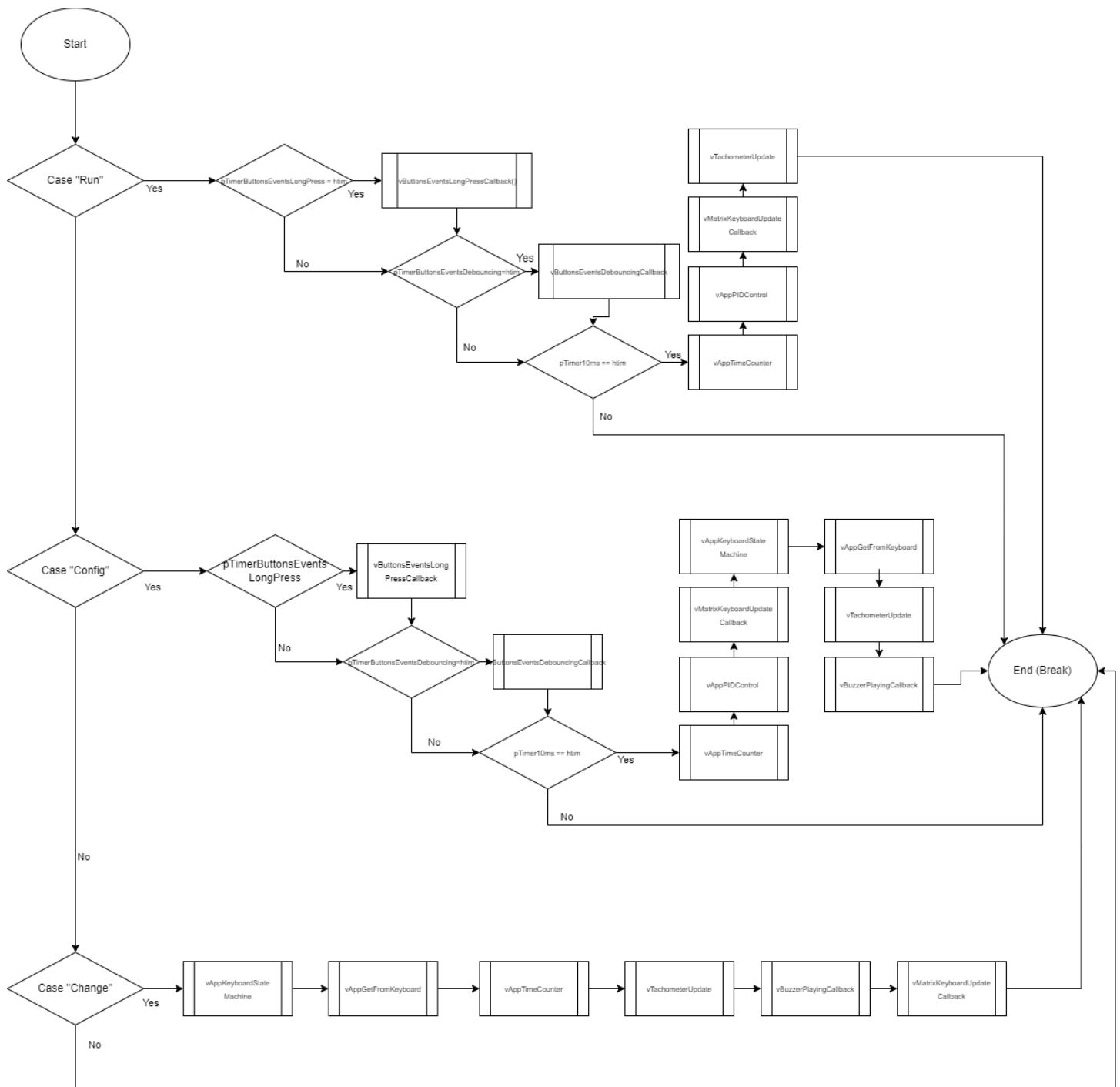
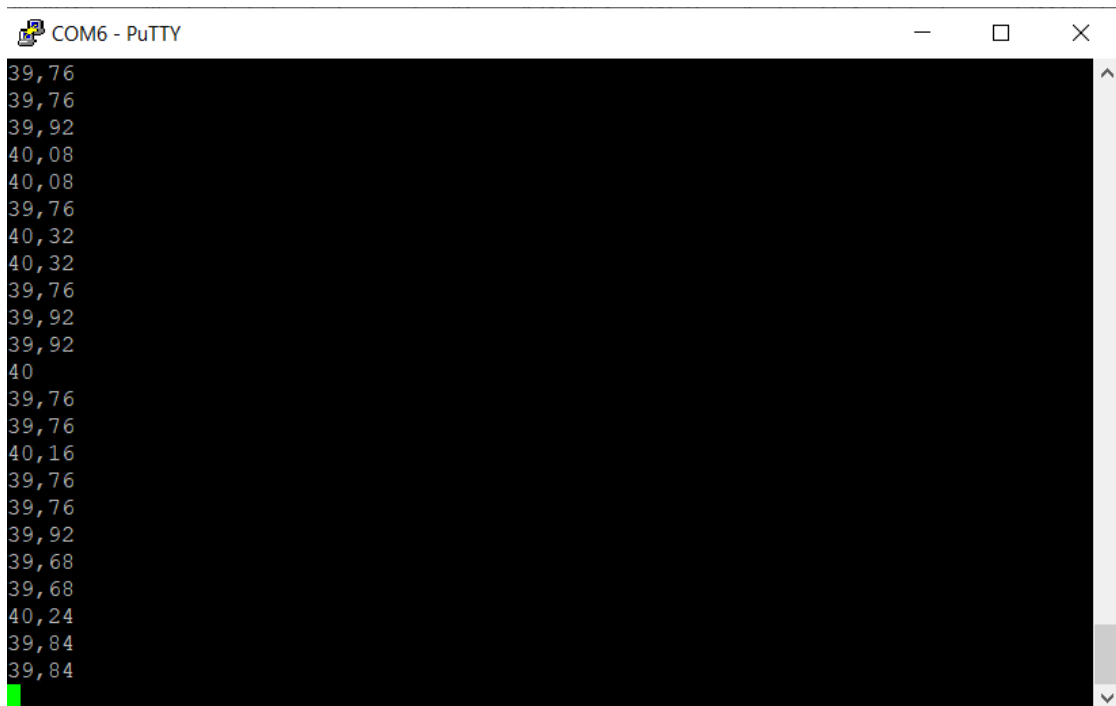


Figura 2.7: Fluxograma da função HAL_TIM_PeriodElapsedCallback [1]

3. Procedimento de sintonização do controlador

Inicialmente foi utilizado o procedimento de Ziegler Nichols com o método de resposta ao degrau em malha aberta.

Para isso foi feito um código que retornasse a temperatura medida pelo sensor via UART na qual era mostrada de meio em meio segundo no terminal do *putty*.



```
COM6 - PuTTY
39,76
39,76
39,92
40,08
40,08
39,76
40,32
40,32
39,76
39,92
39,92
40
39,76
39,76
40,16
39,76
39,76
39,92
39,68
39,68
40,24
39,84
39,84
```

Figura 3.1: Terminal do *putty*.

A partir disso, foi desligado o *heater* e o sistema foi resfriado até a menor temperatura, 27°C. Enfim, foi realizado um step de tensão na *heater* até sua tensão máxima, 3,3V e foi dado um tempo para que o sistema esquentasse até a temperatura máxima a fim de caracterizar o sistema.

Foi obtida a seguinte curva de temperatura:

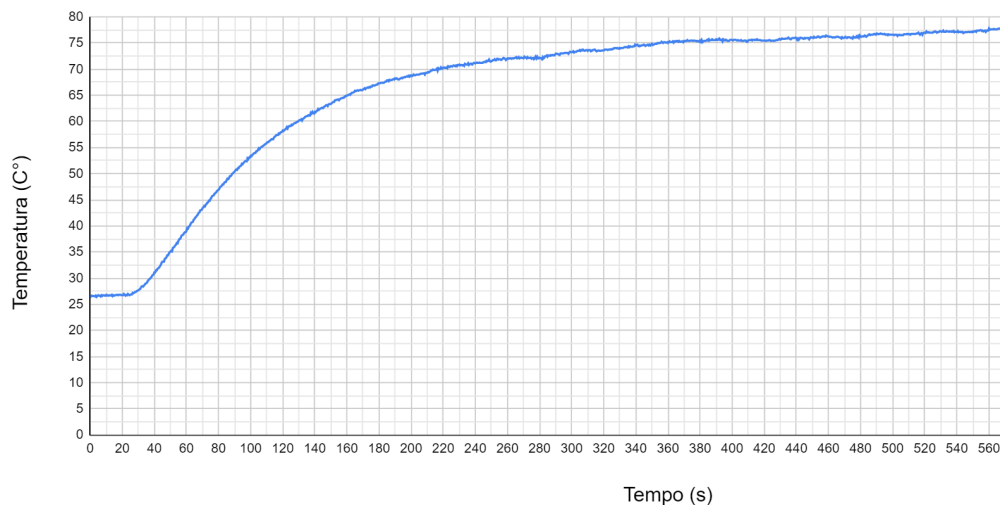


Figura 3.2: Ziegler Nichols - Resposta ao degrau

A partir dessa resposta foram estipulados os valores necessários para a definição dos valores de K_p , K_i e K_d , ou seja, foram definidos, L , R e K .

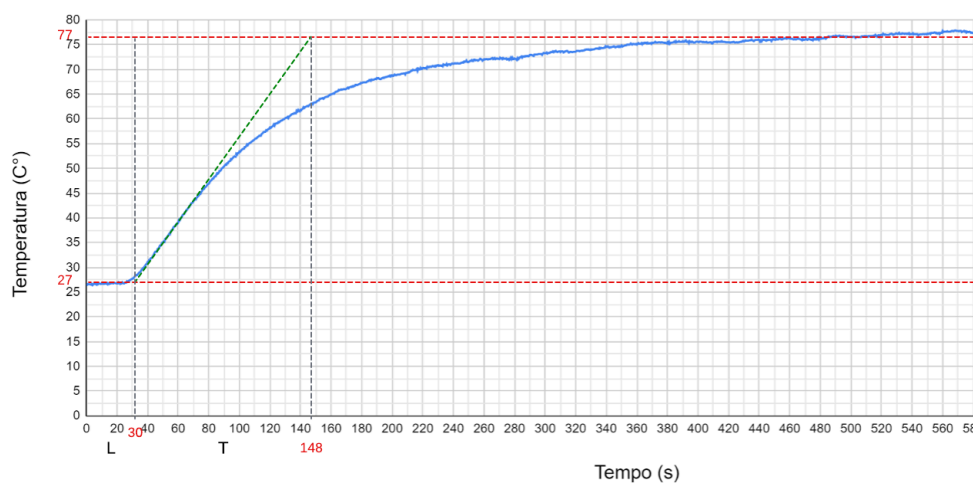


Figura 3.3: Ziegler Nichols - Análise da resposta ao degrau

Então, pelo gráfico é facilmente visível os valores de $L = 30$ e $T = 118$, e para K , foi calculado a variação de temperatura e dividido pelo valor do degrau de tensão, 3,3V. Logo $K = (77 - 27) \div 3,33 \Rightarrow K = 15,15$. Também foi definido o valor de $a = K * L \div T$. A partir disso foram calculados os respectivos valores de K_p , K_i e K_d , baseado na tabela a seguir.

Controller	Kp	Ti	Td
P	1/a	-	-
PI	0.9/a	3L	-
PID	1.2/a	2L	L/2

Tabela 3.1: Ziegler Nichols - Fórmulas para o método da resposta ao degrau

$$Kp = 1,2 \div a = 4,62$$

$$Ti = 2 * L = 2 * 30 = 60 \Rightarrow Ki = 0,077$$

$$Td = L \div 2 = 15 \Rightarrow Kd = 0,31$$

Porém, foi visto que o sistema não respondia muito bem em regime permanente além de ter um overshoot relativamente alto, além disso em regime permanente o PWM do Heater variava de 0 a 100 muito rápido, mostrando que a saída do controlador estava saturando.

Dessa forma o PID foi alterado de forma empírica a partir de alguns experimentos.

Inicialmente foi feita a tentativa de se definir o Kp, zerando Ki e Kd, foram feitos uma bateria de testes com a temperatura saindo de 30°C com a referência em 50°C.

Foram testados Kp =2, Kp = 3 e Kp =5 e foi visto inicialmente que o tempo para cruzar a referência era muito similar. Logo a escolha foi feita baseada no overshoot.

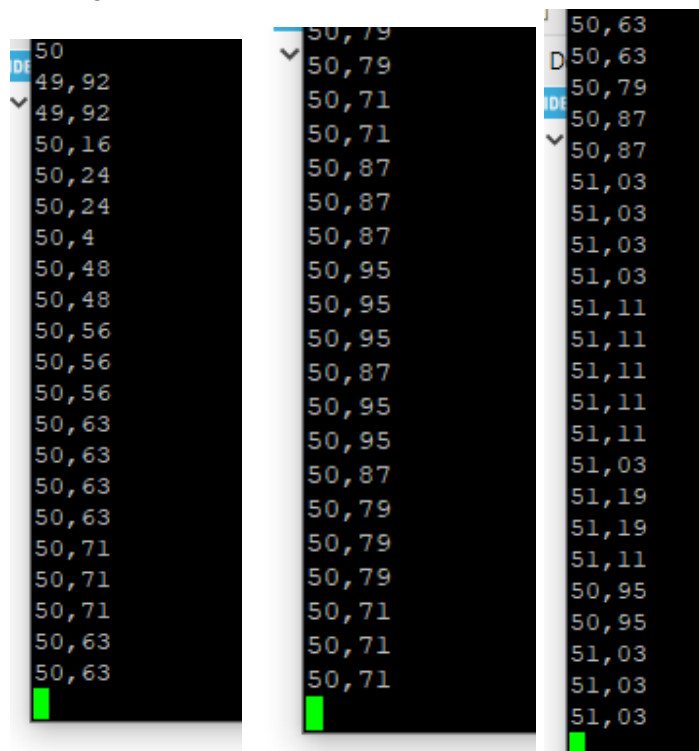


Figura 3.4: Ziegler Nichols - Valores obtidos para o método

Como Kp=2 apresentou menor overshoot, foi utilizado esse valor de Kp.

Posteriormente foi definido o K_i , para isso foi fixado o K_p no valor achado anteriormente ($K_p=2$) e foram feitos diversos testes variando o K_i para 0,2; 0,5; 0,7; 0,9 e 2. Como parâmetro de escolha foi definido o overshoot e o tempo de acomodação considerando o tempo após a temperatura cruzar a referência pela primeira vez.}

Nos gráficos a seguir o eixo vertical representa a temperatura medida e o eixo horizontal o tempo.

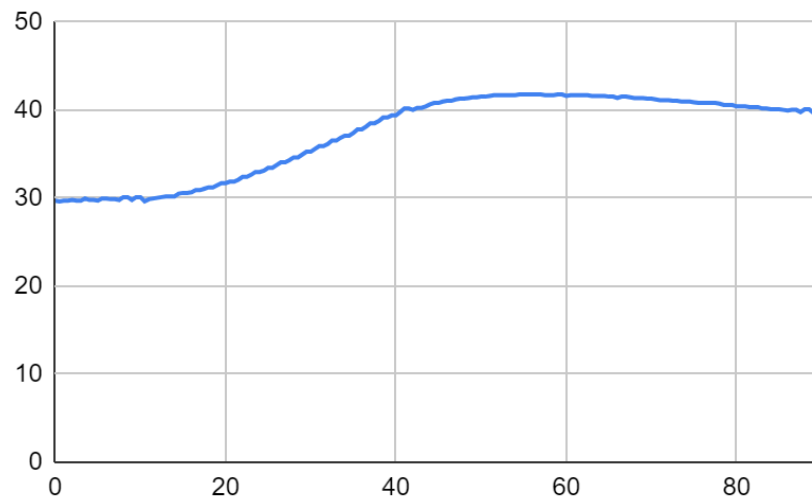


Figura 3.5: Temperatura x Tempo. $K_i = 0,2$; $T_s = 42s$; $Overhoot = 1,75^\circ C$

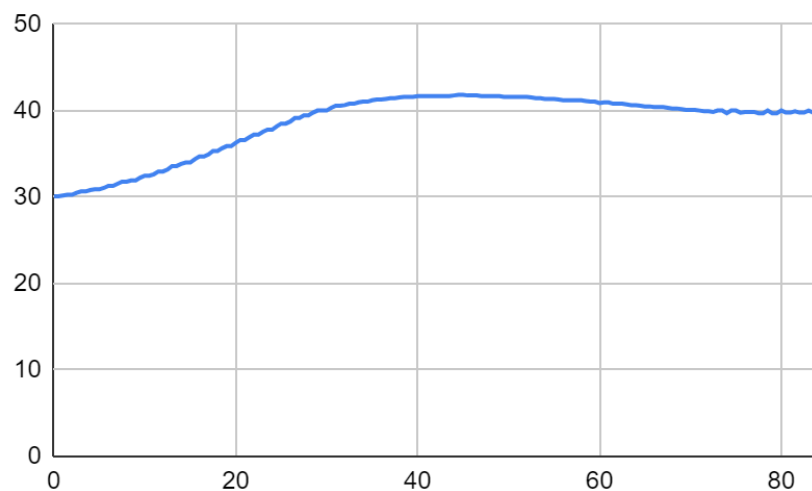


Figura 3.6: Temperatura x Tempo. $K_i = 0,5$; $T_s = 44s$; $Overhoot = 1,86^\circ C$

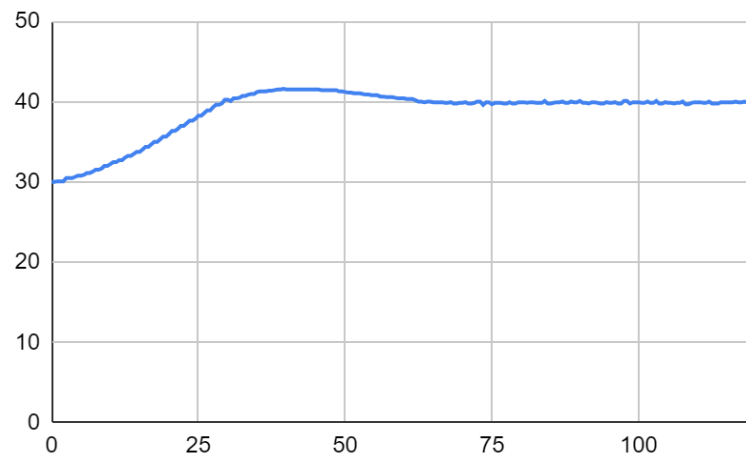


Figura 3.7: Temperatura x Tempo. $K_i = 0,7$; $T_s = 32s$; *Overshoot* = 1,59°C

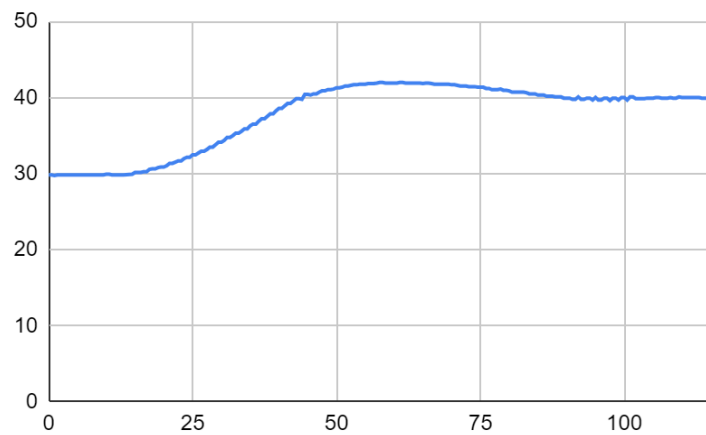


Figura 3.8: Temperatura x Tempo. $K_i = 0,9$; $T_s = 46s$; *Overshoot* = 1,98°C

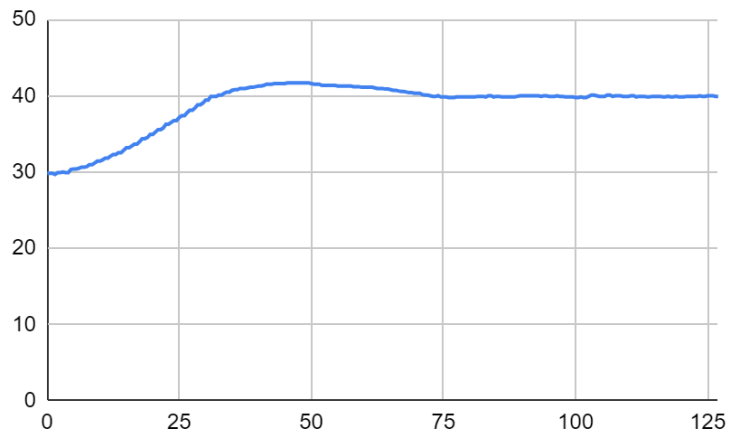


Figura 3.9: Temperatura x Tempo. $K_i = 2$; $T_s = 41s$; $Overhoot = 1,67^\circ C$

Então o K_i foi definido como 0,7.

Finalmente, para definição do K_d , foi utilizado $K_p = 2$ e $K_i = 0,7$ e então variado os valores de K_d entre 0,05; 0,1; 1 e 3.

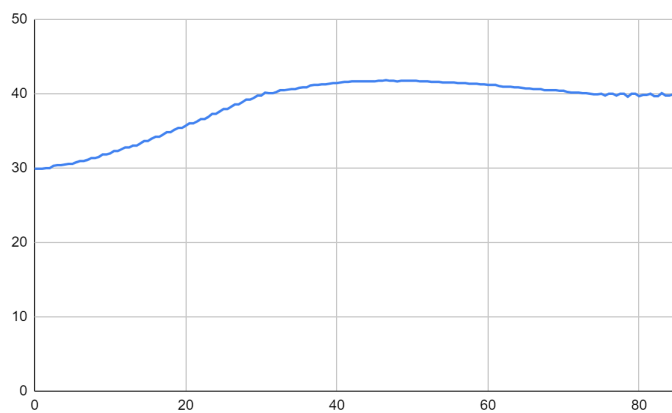


Figura 3.10: Temperatura x Tempo. $K_d = 0,05$; $T_s = 43s$; $Overhoot = 1,75^\circ C$

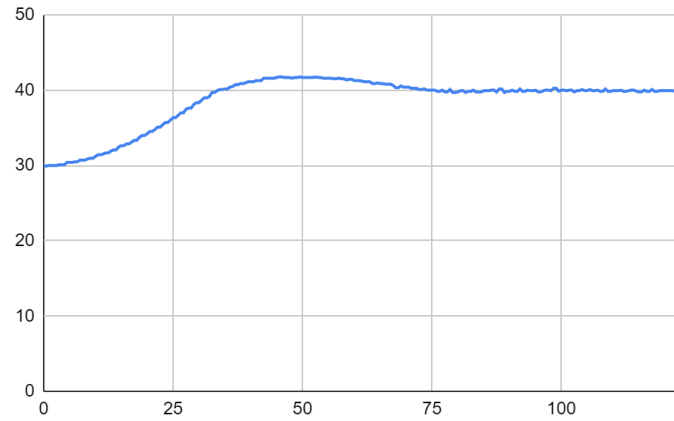


Figura 3.11: Temperatura x Tempo. $K_d = 0,1$; $T_s = 45s$; $Overhoot = 1,67^\circ C$

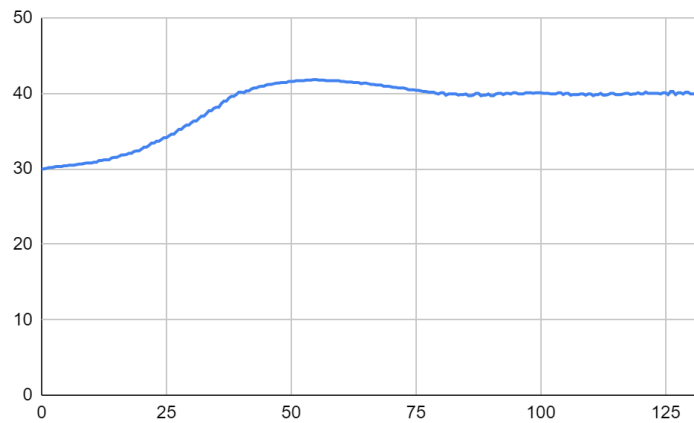


Figura 3.12: Temperatura x Tempo. $K_d = 1$; $T_s = 42s$; $Overhoot = 1,83^\circ C$

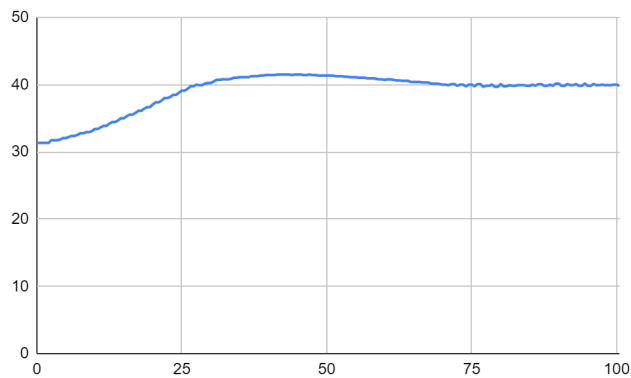


Figura 3.13: Temperatura x Tempo. $K_d = 3$; $T_s = 43s$; $Overhoot = 1,51^\circ C$

Ao final dos testes foi visto que o uso do ganho K_d não era muito vantajoso, já que o controlador PI com ganhos $K_p = 2$ e $K_i = 0,7$ era mais eficiente e melhor que o PID, já que tinha

menor overshoot e menor tempo de acomodação.

Portanto, chegou se a um controlador PID com ganhos $K_p = 2$, $K_i = 0,7$, $K_d = 0$. Porém, ao mudar de equipamento (placa) foi visto que esse controlador não mantinha a temperatura no valor de referência de forma satisfatória, de modo geral ele ficava abaixo desse valor. Além disso, com a implementação da energização do cooler para diminuir o overshoot, foi necessário reprojeter os ganhos do PID.

Com um processo semelhante, porém mais empírico, foi visto que o controlador ideal seria com ganhos: $K_p = 2,7$; $K_i = 0,9$; $K_d = 0$.

4. Manual de utilização

Foi realizada a concepção de um manual de instruções para operar o controlador, com o objetivo de facilitar a utilização e configuração do controlador de temperatura, seja utilizando a interface local ou através do computador.

 Manual do proprietário para o Controlador de Temperatura B5

5. Problemas identificados e não resolvidos

- Primeiro caractere a ser enviado por UART não é reconhecido
- Pulso duplo ao apertar os push buttons mesmo com o tratamento de Debouncing (fenômeno visto em aula);
- Às vezes o LCD se inicia com lixo na tela, é necessário remover o cabo USB e colocá-lo para voltar a funcionar;
- Cálculo da velocidade do cooler feito de forma errada.

6. Autoria dos códigos fornecidos

Não foi necessário utilizar nenhum código dos colegas, visto que ao longo dos experimentos foi possível finalizar ou compreender o funcionamento geral do sistema. Porém tivemos um único ponto que não conseguimos resolver que foi a contabilização correta da velocidade do cooler através do tacômetro, e como tentamos solucionar o erro até os últimos minutos antes da entrega, não tivemos a oportunidade ou intuito de utilizar algum código externo, mesmo que fosse a solução para nosso controlador.