Universidade Estadual de Campinas

Faculdade de Engenharia Mecânica

Projeto final

ES670 – Projeto de Sistemas Embarcados

Igor Barros Teixeira 217947

Matheus Santos Sano 222370

CAMPINAS, junho de 2022

1. Resumo

Este projeto consiste no desenvolvimento de um sistema embarcado utilizando o kit de desenvolvimento FRDM KL25Z acoplado a uma placa de periféricos MCLAB2. Os principais objetivos do sistema implementado são controlar a temperatura do aquecedor da placa de periféricos, implementar uma comunicação remota via UART e uma comunicação por meio da interface local.

Visando tais objetivos, foi implementada uma máquina de estados cujos estados representam as ações que o usuário pode escolher para controlar a temperatura. Esta máquina de estados é uma parte muito importante do projeto, pois é por meio dela que é possível obter e alterar os parâmetros importantes para controlar a temperatura, dentre eles estão a velocidade do cooler, a temperatura desejada do aquecedor (set point) e os ganhos do controlador PID.

Ademais, foi implementada uma comunicação serial via protocolo UART para que o usuário possa se comunicar com o sistema e configurar os parâmetros que são pertinentes para controlar a temperatura do aquecedor. O terminal serial utilizado foi o Putty. Além disso, foi implementada uma comunicação por interface local, na qual o usuário interage com o sistema por meio de botões. Neste caso, é possível trocar as informações apresentadas pela placa LCD apertando um dos botões, podendo visualizar todos os parâmetros pertinentes para o controle de temperatura. É possível também alterar seus valores por meio dos botões, em que o botão 3 serve para diminuir o valor e o botão 4 serve para aumentá-lo. Vale ressaltar que a comunicação por interface local também foi implementada por meio de uma máquina de estados presente no arquivo “screenControl.c”.

Para que a temperatura fosse controlada corretamente o sistema possui um controlador PID cujos ganhos são setados inicialmente com um valor fixo para que o sistema possa aquecer o mais rápido possível e possuir um overshoot máximo de 2°C. Entretanto, o usuário pode alterar seus ganhos pela comunicação remota e pela interface local.

Por meio de uma interrupção o sistema atualiza o controlador com a temperatura atual e o set point, e obtém as informações importantes do cooler e do aquecedor, sendo elas a velocidade do cooler em RPM, a média das últimas 5 temperaturas do aquecedor.

O sistema também possui funcionalidades interessantes. Caso o usuário coloque um set point superior à temperatura máxima de 90°C, o sistema automaticamente irá desligar o aquecedor, zerando seu duty cycle, e ligando o cooler no máximo. Uma outra funcionalidade é o uso do RGB LED do FRDM-KL25Z, que altera a sua cor com base no duty cycle do aquecedor:

DUTY CYCLE AQUECEDOR (DC)	RGB LED
DC < 25%	VERDE
25% < DC < 75%	AZUL
DC > 75%	VERMELHO

2. Documentação do sistema

Em primeiro lugar, deve-se ter os requisitos de projeto para que seja possível toda a implementação de hardware e software embarcado. Os requisitos são fornecidos pelo cliente e ajustados de acordo com a fase de análise juntamente com o responsável, esses requisitos podem ser alterados de acordo com o custo para cumprir cada um deles. Abaixo uma tabela com os requisitos funcionais e não funcionais do projeto.

Requisitos funcionais	
R1	Manter a temperatura no valor especificado pelo usuário
R2	Uso de cooler com velocidade fixa definida pela usuário
R3	Possuir interface local
R4	Possuir interface UART
R5	Exibir temperatura atual
R6	LED mudar de cor em relação ao duty cycle do aquecedor:
	verde se DC < 25%, azul se 25% < DC < 75%, vermelho se DC > 75%
Requisitos não funcionais	
	Temperatura máxima de trabalho de 90°C
	Possuir conexão USB
	Alimentação 12Vcc
	Aquecer o mais rápido possível
	Overshoot máximo de 2°C

Após definidos os requisitos, é o momento de detalhar o hardware necessário para cumprir com margem de segurança tudo que é pedido. Nesta fase é importante ter em mente a robustez necessária de acordo com local de instalação do sistema, tempo de uso, condições ambientes, infraestrutura necessária, dentre outros. Abaixo um diagrama de blocos indicando quantidades e quais hardwares serão utilizados.

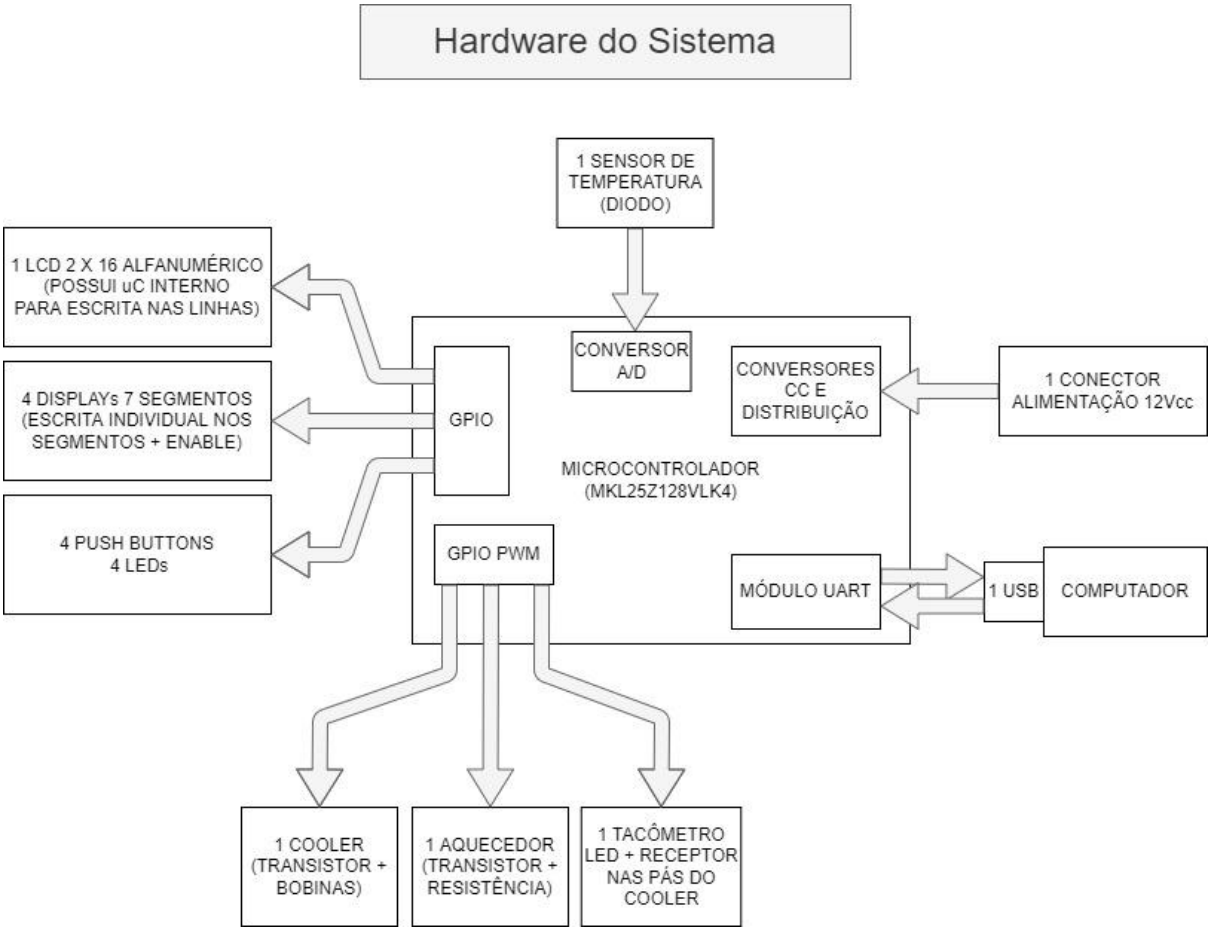

Figura 01: Esquemático do hardware que compõe o sistema.

Tendo o hardware definido, deve-se planejar o escopo modular das ações que o uC fará. Esta visão macro ajudará na organização de tarefas e interfaces entre códigos. Abaixo estão os diagramas utilizados como base para idealizar o projeto global e trabalhar nos softwares embarcados.

Os dois primeiros diagramas mostram um sistema baseado em máquina de estado para uso das telas/modos de configuração local do sistema e para comunicação entre computador e uC via UART. O terceiro diagrama mostra o ciclo de controle da temperatura do aquecedor, ele ocorre periodicamente utilizando interrupções ativadas pelo temporizador de baixo consumo.

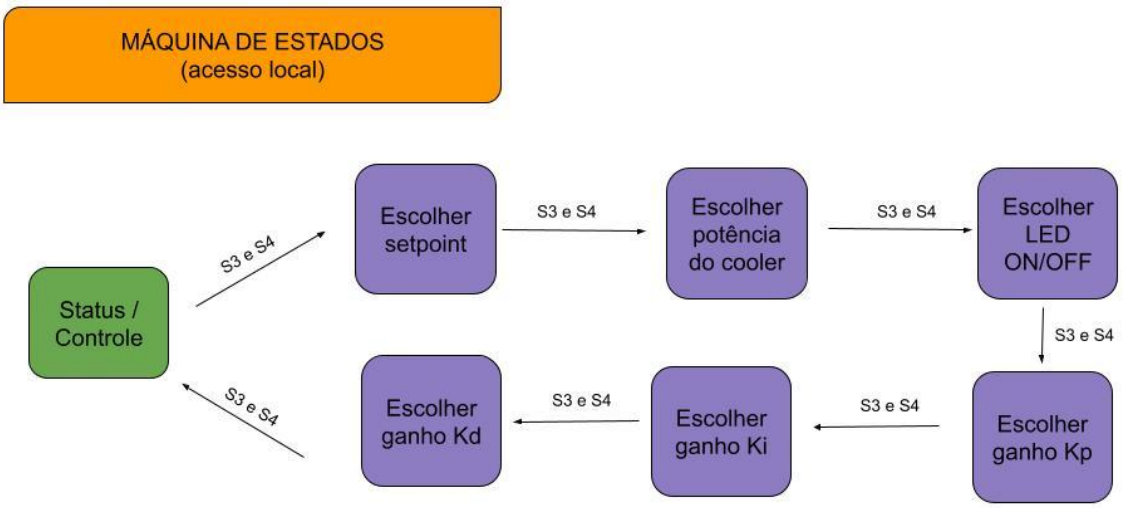

Figura 02: Máquina de estados do acesso local.

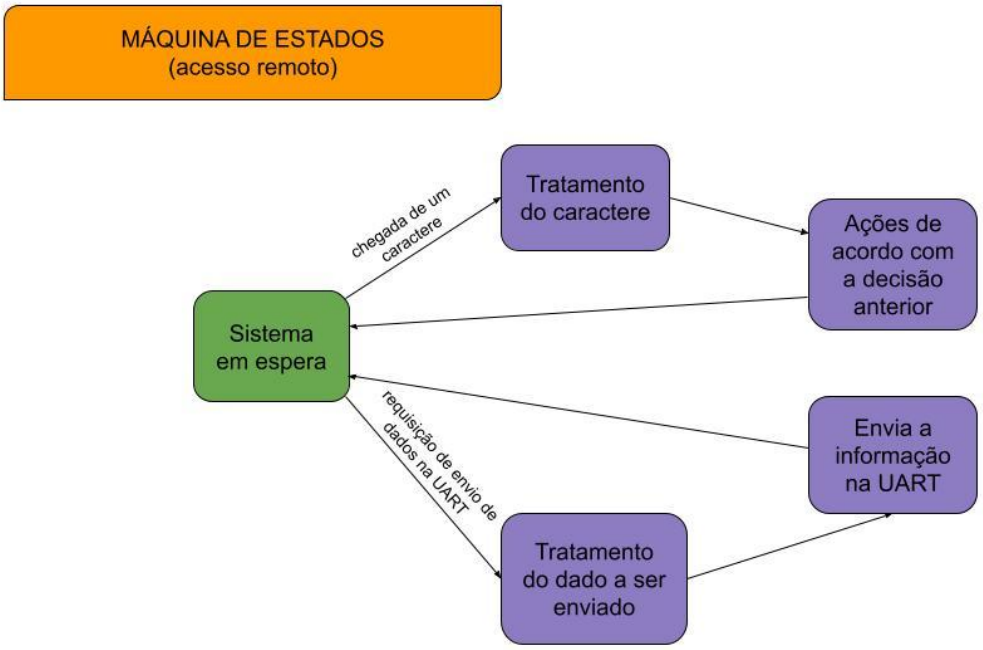

Figura 03: Máquina de estados do acesso remoto.

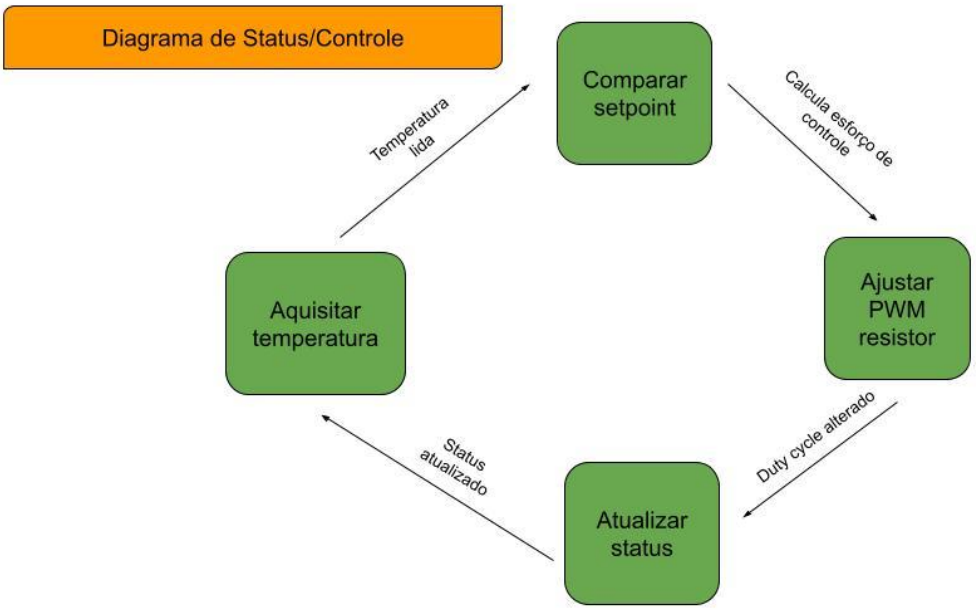

Figura 04: Diagrama de funcionamento do módulo de controle de temperatura.

Tendo o funcionamento global do sistema, é possível arquitetar a sequência de funcionamento do software, colocando um pouco mais de detalhes das atividades que serão realizadas e os módulos de software envolvidos. Na próxima página o fluxograma do software do uC.

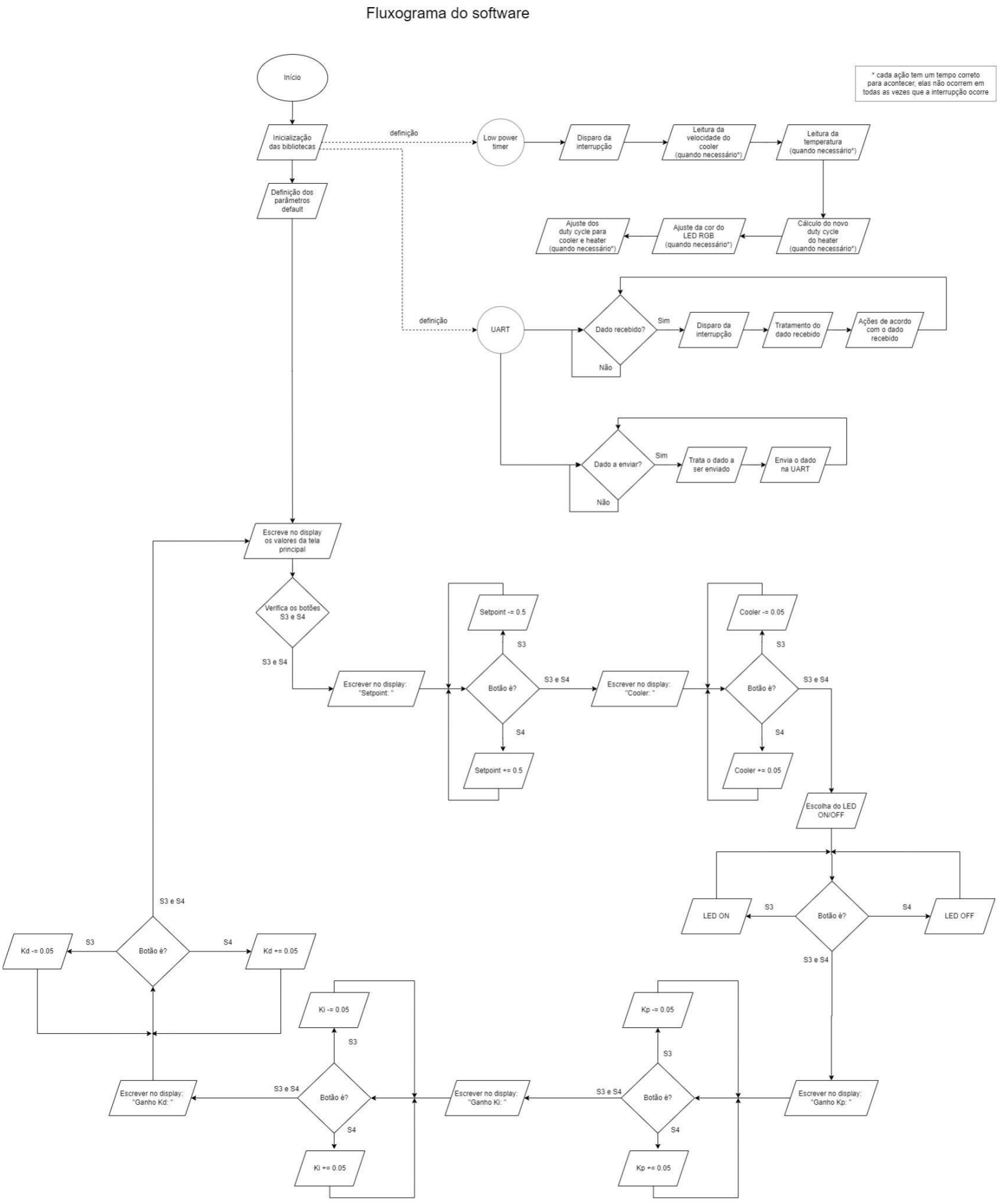

Figura 05: Fluxograma do software.

3. Procedimento de sintonização do controlador

Para a sintonização do controlador foram utilizados dois métodos: pelo método de Ziegler-Nichols e por tentativa e erro. O controlador foi implementado para controlar a temperatura do aquecedor para que ela se aproxime do set point (valor de temperatura desejado) o mais rápido possível, com a menor oscilação possível e com o menor overshoot. Vale salientar que o cooler foi usado para gerar um distúrbio no aquecedor.

O primeiro método aplicado para sintonizar o controlador foi por tentativa e erro, alterando os valores dos ganhos do PID e avaliando seus efeitos na variação de temperatura do aquecedor. A primeira coisa feita foi aumentar o ganho proporcional K_p até a temperatura do aquecedor variar com uma pequena oscilação. Durante a definição do valor do K_p, foi visto que o aquecedor não conseguia atingir o set point, quando a temperatura desejada era muito elevada. Para que tal valor fosse atingido, foi aumentado o ganho integral K_i, eliminando as oscilações geradas pelo ganho proporcional.

Ao determinar os valores de K_p e K_i , ainda foi necessário ajustar ambos os ganhos para que a temperatura do aquecedor possa atingir o set point com menos oscilação possível. Feito isso, o aquecedor apresentava um overshoot muito alto e para diminuí-lo, foi necessário aumentar o ganho derivativo K_d . Vale salientar que o K_d é muito sensível, então uma pequena diferença no K_d foi suficiente para conseguir um sistema mais rápido e com um overshoot menor.

Após sintonizar o controlador PID por tentativa e erro, o sistema funcionou muito bem, apresentando um crescimento e uma diminuição linear de temperatura, atingindo o set point com um overshoot menor do que 1°C. Vale ressaltar que o bom funcionamento deste controlador é oriundo também da implementação de uma função que calcula a média das 5 últimas temperaturas obtidas pelo ADC. O resultado desta média é o valor que o controlador compara com o set point. Assim, nosso projeto trabalha com a média das temperaturas, evitando possíveis erros de medição causados por falha eletrônica da placa. Portanto, os valores dos ganhos obtidos por meio de tentativa e erro foram:

$$K_p = 12,21 \qquad K_i = 0,08 \qquad K_d = 0,7$$

Contudo, quando o set point é superior à 45°C, o controlador continua aumentando a temperatura de forma linear e sem oscilações, porém apresenta um overshoot superior a 2°C e tem uma grande queda de temperatura após atingir o set point. Essa grande variação ocorre por conta da placa apresentar saltos de temperatura quando está maior do que 45°C. Como o valor apresentado no LCD é a temperatura média das 5 últimas medições, é visto apenas uma queda linear de temperatura pelo LCD. Porém, é o valor médio que está diminuindo linearmente após uma queda brusca de temperatura.

Quando o controlador percebe esta queda brusca, ele coloca o duty cycle do aquecedor para 100% mesmo que a temperatura atual esteja próxima do set point. Dessa forma, o controlador não consegue evitar um overshoot menor do que 2°C.

O outro método utilizado para sintonizar o controlador PID foi o método de Ziegler-Nichols, especificamente o método da curva de reação. Para isso o sistema foi implementado em malha aberta, ou seja, o controlador não controla mais o aquecedor, sendo este ajustado com um duty cycle fixo de 50%. O cooler foi usado como distúrbio com um duty cycle também de 50%. Assim, com o sistema em malha aberta, foram obtidos vários valores de temperatura em um determinado tempo. Após a temperatura do aquecedor se estabilizar, foi plotado um gráfico da temperatura em função do tempo.

Para obter o tempo de cada medição de temperatura, foi implementado um código na interrupção que, a cada 100 ms, o valor da temperatura atual era printado na tela da comunicação serial. Assim, foi obtida a temperatura do aquecedor a cada 100 ms, podendo enfim plotar o gráfico a seguir:

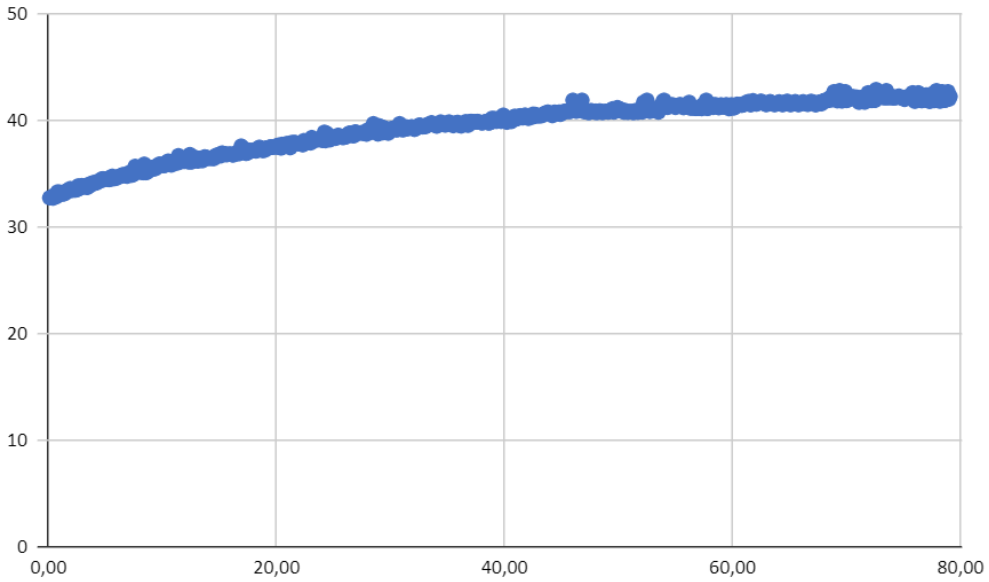

Gráfico 01: Temperatura do aquecedor em função do tempo.

O método de Ziegler-Nichols consiste em traçar uma reta no ponto de maior derivada de tal curva (Gráfico 01) e, a partir desta curva, obter os valores de L, K e T, como apresentado a seguir:

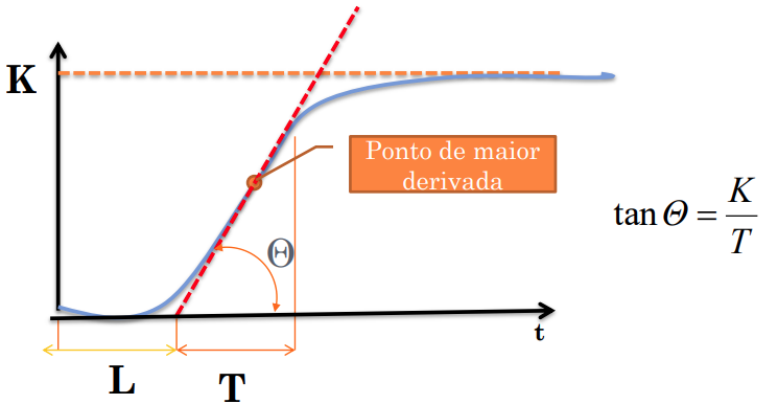

Figura 06: Esquemático de como obter os valores de K, L e T a partir da curva Temperatura x Tempo.

O valor de K é a diferença entre a temperatura mínima medida e a temperatura do aquecedor na estabilidade:

$$K = 10,06$$

Traçando uma reta no ponto de maior derivada, é possível obter os parâmetros L e T:

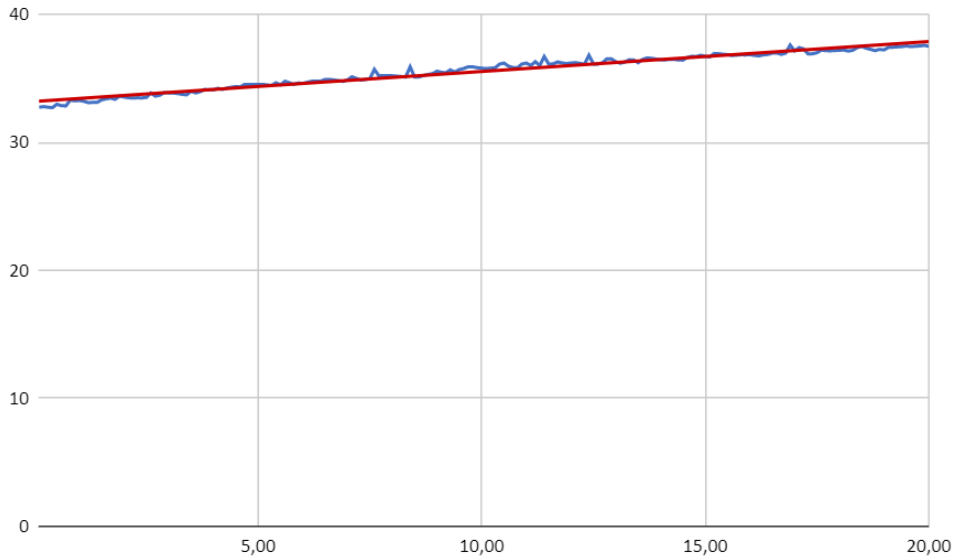

Gráfico 02: Intervalo da curva da Temperatura x Tempo entre 0 segundos e 20 segundos (em azul). Linha de tendência deste intervalo da curva (em vermelho).

Os valores de L e T são obtidos a seguir:

$$L = 0,13$$

$$tg(\theta) = \frac{K}{T} \quad \Leftrightarrow \quad T = \frac{K}{tg(\theta)} \quad \Rightarrow \quad T = \frac{10,09}{0,32} = 31,4375$$

Com os valores dos parâmetros K, L e T, é possível obter os ganhos K_p , K_i e K_d do controlador PID por meio dos cálculos a seguir:

$$K_p = 1,2 \cdot \frac{T}{L} = 1,2 \cdot \frac{31,4375}{0,13}$$

$$K_i = 2 \cdot L = 2 \cdot 0,13$$

$$K_d = 0,5 \cdot L = 0,5 \cdot 0,13$$

$K_p = 290,19$	$K_i = 0,26$	$K_d = 0,065$

Por meio destes ganhos, o controlador não funcionou corretamente, apresentando altas oscilações. Infelizmente, após muitas tentativas, todos os ganhos calculados foram insuficientes para sintonizar um controlador que funcione atingindo 100% dos requisitos.

4. Manual de utilização

A utilização do sistema implementado pode ser feita de duas formas: por meio da comunicação remota via UART e pela interface local via botões e display.

A comunicação remota via UART consiste na comunicação por meio do terminal serial Putty e da máquina de estados implementada. Para iniciar um novo comando por meio da serial é necessário digitar #. Em seguida, o usuário pode escolher determinadas opções, digitando uma das letras a seguir:

LETRA	OPÇÃO
h	Habilitar interface local
d	Desabilitar interface local
g	Obter determinado parâmetro (GET)
s	Alterar valor de determinado parâmetro (SET)

Caso a opção GET seja escolhida, o usuário tem as possíveis possibilidades:

LETRA	OPÇÃO
t	Obter valor da temperatura e do duty cycle do aquecedor
c	Obter valor da velocidade do cooler (RPM) e do duty cycle do cooler

Caso a opção SET seja escolhida, o usuário tem as possíveis opções:

LETRA	OPÇÃO
t	Alterar a temperatura do aquecedor
c	Alterar o duty cycle do cooler
p	Alterar o ganho proporcional (Kp)
i	Alterar o ganho integral (Ki)
d	Alterar o ganho derivativo (Kd)

Vale ressaltar que o comando é finalizado quando o usuário digita qualquer outra letra que não seja compatível com uma das opções disponíveis. Ademais, para que um comando seja concluído, o usuário deve teclar ponto e vírgula (;). Para facilitar a visualização deste funcionamento é apresentado na próxima página um diagrama da máquina de estados:

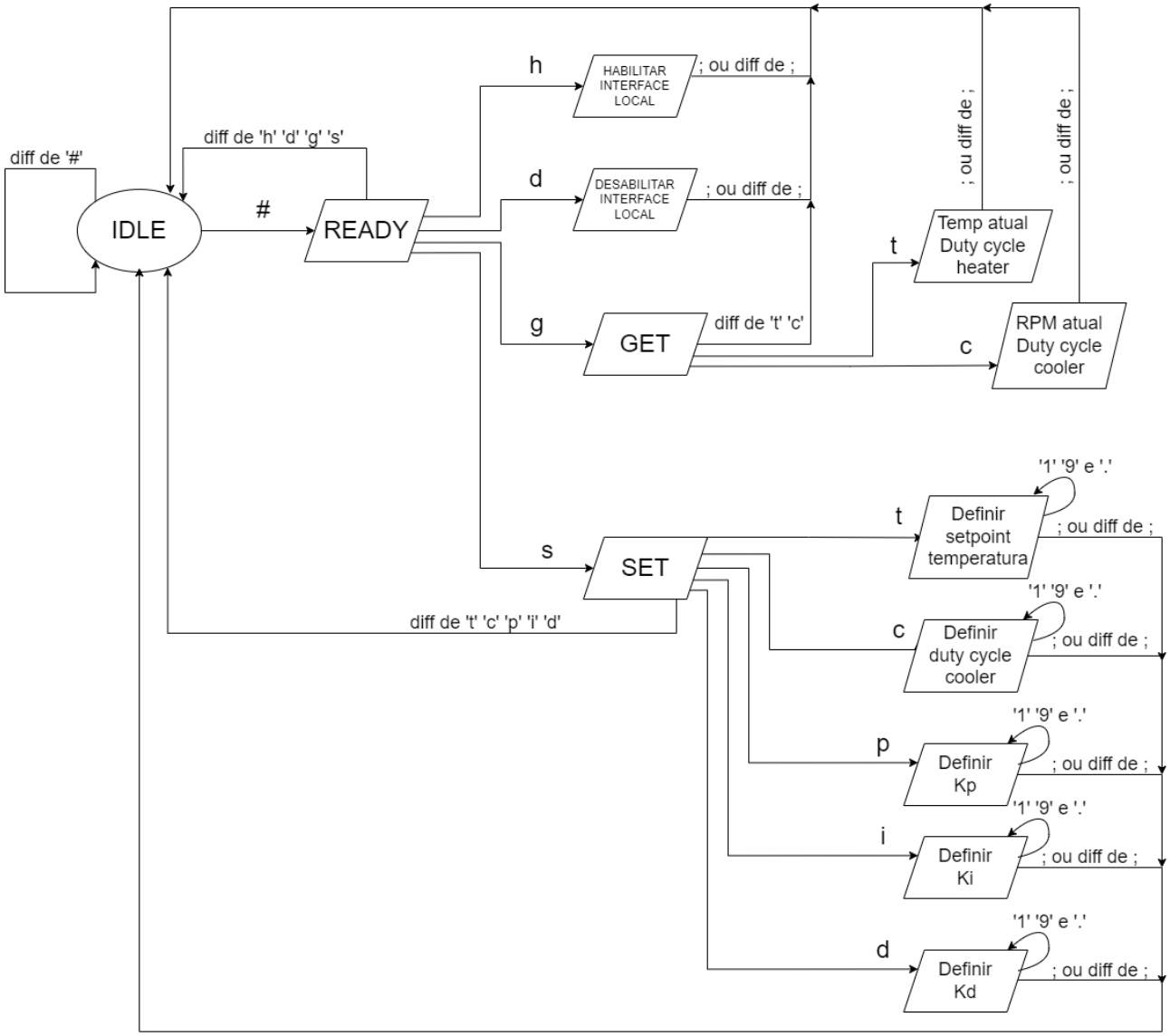

Figura 07: Diagrama da máquina de estados implementada para uso na comunicação remota via UART.

A máquina de estados considera que caracteres maiúsculos são opções inválidas e cancela o comando caso receba um deles.

A outra forma de interação com o sistema é por meio da interface local via botões e display. Como pode ser visto na Figura 02, esta interface local pode ser habilitada e desabilitada por meio da comunicação via UART. Vale ressaltar que para usar a interface local pela primeira vez, o usuário deve habilitá-la pela comunicação serial.

Existem diferentes telas na interface local do sistema, para todas elas a linha de baixo mostra o setpoint de temperatura e a temperatura atual lida e a linha de cima muda de acordo com a tela corrente. Para transitar entre as telas basta pressionar os botões S3 e S4 até que a tela mude. Caso mantenha eles pressionados, as telas ficam sendo alteradas até que eles sejam soltos.

A tela inicial mostra o status do duty cycle do aquecedor e do cooler em porcentagem, nada é possível de ser alterado nesta tela. Em todas as outras existe um parâmetro que pode ser configurado pressionando o botão S3 para decrementar ou S4 para incrementar. Os parâmetros possuem valores mínimos e máximos, o próprio sistema limita esses parâmetros para que não atinjam valores inadequados via interface local. As variáveis alteradas via interface local são atualizadas nos parâmetros de funcionamento em tempo real e permanecem salvas enquanto o sistema estiver ligado. Sempre que o sistema é reiniciado ele admite valores default para todos os parâmetros, ou seja, alterações via interface local são perdidas caso o microcontrolador seja desligado.

Abaixo uma imagem indicando a máquina de estados da interface local descrita:

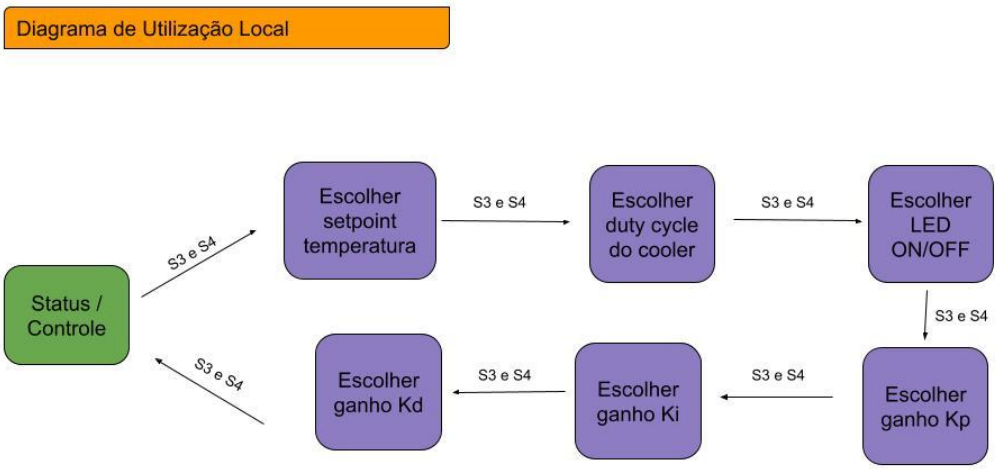

Figura 08: Diagrama da máquina de estados implementada para uso na transição de telas e alteração de parâmetros.

5. Problemas identificados e não resolvidos

Um dos principais problemas identificados foi a sintonização do controlador. Foram realizados vários testes para obter os ganhos por meio do método de Ziegler-Nichols, plotando diferentes gráficos e adquirindo diferentes valores destes gráficos. Infelizmente, não foi possível adquirir um resultado satisfatório.

Apesar de a sintonização do controlador por meio de tentativa e erro ter sido eficaz, o controlador não foi capaz de controlar a temperatura do aquecedor com temperaturas próximas e maiores do que 45°C, graças aos saltos de temperatura. Mesmo tentando corrigir esses saltos, não foi possível garantir um controlador eficaz para altas temperaturas.

Um outro problema que foi notado nos testes feitos após o projeto pronto é que na tela de configuração de Kp na interface local ocorre um problema na **exibição** do valor de ganho quando ele é menor que 0,1. Quando o valor do ganho é 0,05 por exemplo, ele é exibido como 0,5 mesmo que na realidade ele seja 0,05 (pela interface serial e no controlador PID o valor está sendo enviado corretamente). Este problema ocorre por conta da conversão de float para char, a parte fracionária do número (,05) é interpretada como 5 pelo microcontrolador. Ao invés de enviar ‘0’ ‘.’ ‘0’ ‘5’ ele envia ‘0’ ‘.’ ‘5’, causando o erro visual citado. Este erro aconteceu em outros momentos durante os testes iniciais da criação das telas e foi resolvido fazendo um tratamento diferente da conversão float para char quando o primeiro dígito após a vírgula é 0. Infelizmente não foi visto que este erro também ocorre com o Kp então por conta do tempo hábil não foi possível replicar a mesma modificação já feita em outro momento para ele.

6. Autoria dos códigos fornecidos

Parte do código deste projeto não foi implementado pela dupla, e sim fornecido pelo professor Rodrigo Bacurau. Esta parte do projeto que não é de autoria da dupla é apontada a seguir:

→ “adc.c” e “adc.h”: foram pouco modificados, sendo a sua maior parte fornecida pelo professor;

→ “board.h”: parte do código foi fornecido pelo professor. Algumas máscaras foram adicionadas pela dupla;

→ “fsl_debug_console.c”: este arquivo foi fornecido integralmente pelo professor e não foi alterado pela dupla;

→ “lcd.c” e “lcd.h”: partes destes arquivos foram fornecidos pelo professor e houve grande modificação destes arquivos por parte da dupla;

→ “ledrgb.c” e “ledrgb.h”: parte do código fornecido pelo professor e pouco alterado pela dupla;

→ “lptmr.c” e “lptmr.h”: os códigos foram integralmente fornecidos pelo professor;

→ “lut_adc_3v3.c” e “lut_adc_3v3.h”: os códigos foram integralmente fornecidos pelo professor;

→ “main.c”: parte do código foi fornecido pelo professor, havendo muita alteração do código por parte da dupla;

→ “mcg.c” e “mcg.h”: códigos integralmente fornecidos pelo professor;

→ “pid.c” e “pid.h”: partes dos códigos fornecidos pelo professor, havendo alteração por parte da dupla;

→ “print_scan.c” e “print_scan.h”: os códigos foram integralmente fornecidos pelo professor;

→ “UART.c” e “UART.h”: maior parte dos códigos foi fornecida pelo professor, sendo pouco alterado pela dupla;

→ “util.c” e “util.h”: parte dos códigos foi fornecida pelo professor, sendo alterado pela dupla;