Projetos de Sistemas Embarcados - ES670

Engenharia de Controle e Automação (ECA)
Faculdade de Engenharia Mecânica (FEM)
Universidade Estadual de Campinas (UNICAMP)

Relatório Final

Controlador de Temperatura

Kit de desenvolvimento FRDM KL25 + Placa MCLAB2

ALUNOS

Giácomo Antonio Dollevedo (RA: 173029)

Gustavo Lino Fernandes (RA: 174167)

PROFESSOR

Prof. Rodrigo Moreira Bacurau

Campinas
Julho de 2020

1. Resumo

Neste projeto foi desenvolvida a implementação em software de um Sistema capaz de controlar temperatura, utilizando o microcontrolador Freedom - FRDM KL25Z da *NXP Semiconductors*, em conjunto com kit McLab2 disponível no laboratório. Durante o projeto, houve a implementação de códigos para realizar a interface do microcontrolador com os periféricos do kit: uma resistência de aquecimento, um sensor de temperatura, uma ventoinha, um encoder, displays de 7 segmentos, display LCD, botões e LEDs. Além disso, também foram desenvolvidos códigos para testar cada módulo em separado. Todo o projeto se encontra documentado em seu repositório no GitHub.

O sistema em questão foi construído para controlar a temperatura de uma resistência do kit, fazendo uso de um controlador PID implementado digitalmente. Ele é capaz de operar até uma temperatura máxima de 90ºC, permitindo a configuração da temperatura alvo ao usuário. Esta configuração pode ser realizada tanto por interface local (botões do kit), quanto serial (RS232).

O projeto permite a fácil identificação visual do estado do sistema pelo usuário, fazendo uso dos LEDs do kit para indicar se o sistema está em modo de configuração, ou se está sendo controlado. Os mesmos LEDs permitem que o usuário diferencie se a temperatura atual está abaixo, acima ou igual a esperada.

O sistema também foi projetado para que a leitura de temperatura atual também possa ser realizada através dos displays de 7 segmentos. Já o display LCD sempre exibe a temperatura alvo e também o estado do sistema.

2. Introdução

Como forma de aprofundar o conhecimento de sistemas embarcados e microcontrolador, utilizando o Kit de desenvolvimento FRDM KL25Z em conjunto com a Placa MCLAB2 disponibilizada, foi elaborado um sistema de controle de temperatura. Diversas são as aplicações de tal sistema, seja em plantas industriais, conforto residencial ou até mesmo em ambientes hospitalares.

Será apresentado neste documento, boa parte da maneira que o sistema foi idealizado e a documentação que permitiu a organização dos códigos,como os diagramas UML, o manual de orientação para utilização, técnica esperada para o controle de temperatura aplicado, a interface eletrônica de controle e a interação com o usuário, que pode ser um usuário final (apenas utilização) ou um usuário técnico (análise e configuração).

Faz parte integral deste trabalho, os códigos desenvolvidos, de forma modularizada para o funcionamento adequado do sistema, que devem ser encontrado no mesmo diretório digital que este documento foi obtido. Deve-se ainda ressaltar que o desenvolvimento deste projeto enfrentou cenários inesperados, os quais impediram o acesso físico ao hardware propriamente dito do Kit de desenvolvimento, desta forma alguns erros que só poderiam se manifestar com a execução prática e em bancada, podem ficar evidenciados no momento em que a implementação completa se realize.

Uma seção dedicada para os problemas identificados, e como foi possível (ou não) solucioná-los foi separada ao final para reconhecer adequadamente as limitações e dificuldades superadas.

3. Documentação do Sistema

A seguir, será descrita a documentação do sistema projetado. Todas as imagens referentes aos diagramas do sistema (modelagem e UML) estão disponíveis no repositório do GitHub em melhor resolução.

3.1 Requisitos do Projeto

Os requisitos do projeto podem ser separados em requisitos funcionais (aqueles que definem o funcionamento do sistema de forma completa e concisa), e em não funcionais (se referem às restrições do projeto que afetam uma ou mais de suas funcionalidades).

3.1.1 Requisitos Funcionais

- Manter a temperatura de um resistor em um valor estabelecido pelo usuário
- Permitir a configuração através de uma interface local
- Exibir a temperatura atual para o usuário
- Exibir a temperatura alvo para o usuário
- Permitir a interface do usuário via serial RS232
- Permitir o Resfriamento Ativo (Cooler)
- Permitir que o sistema seja configurado enquanto opera, sem necessidade de reiniciá-lo

3.1.2 Requisitos Não Funcionais

- Temperatura máxima de operação igual a 90ºC
- Overshoot Máximo de 1ºC
- Aquecimento mais rápido possível
- Permitir a fácil identificação visual do estado do sistema
- Permitir ao usuário acesso aos parâmetros do controlador

3.2 Diagrama de casos de uso

Basicamente, os cenários que o usuário poderá interagir com o sistema está descrito no Diagrama de Casos de Uso da figura 1. Pode resumir dois cenários principais de uso:

- Checar Parâmetros: O indivíduo que atua no sistema poderá escolher se os parâmetros serão checados através do display de 7 segmentos (que estará sempre disponível), o qual o único parâmetro permitido de exibição é a temperatura atual. Também haverá a opção de checagem através da UART, onde além da temperatura, poderá ser consultados os valores de duty cycle do aquecedor e do ventilador.
- Alterar Parâmetros: O usuário do sistema poderá atuar diretamente na alteração dos parâmetros, através dos botões será permitido somente a alteração da temperatura

alvo, entretanto com acesso ao conector RS-232 o usuário poderá utilizá-lo para além de alterar a temperatura alvo, também será possível setar novos valores para os ganhos Kp, Ki e Kd do controlador PID, assim como alterar manualmente o duty cycle do aquecedor e do ventilador.

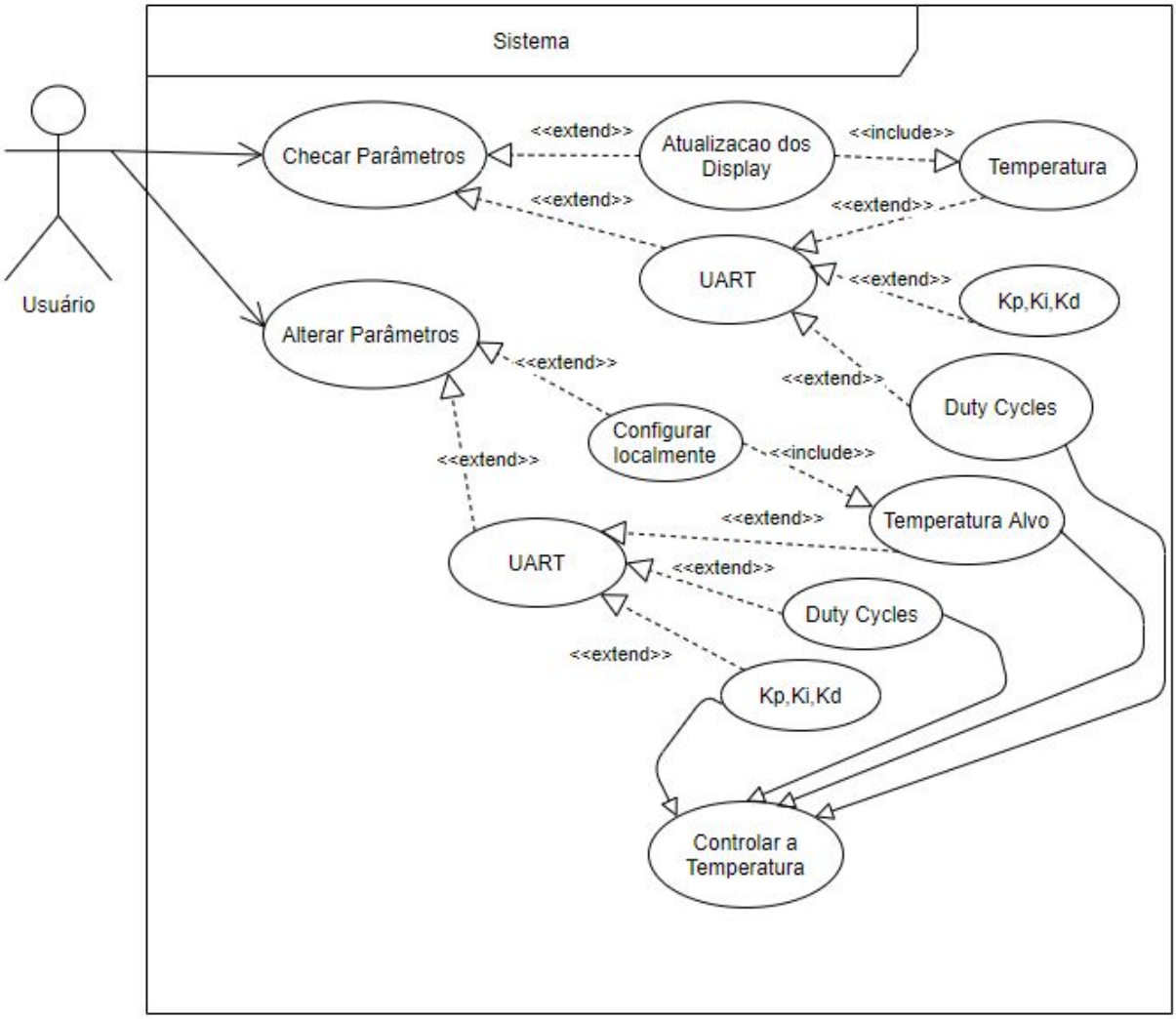


Figura 1 - Diagrama de casos de uso do Sistema de Controle de Temperatura

3.2.1 Diagrama de Cenários

Para detalhar um pouco melhor como cada caso de uso foi idealizado, o diagrama de cenários foi construído para as principais operações realizadas pelo sistema contidas na Figura 1.

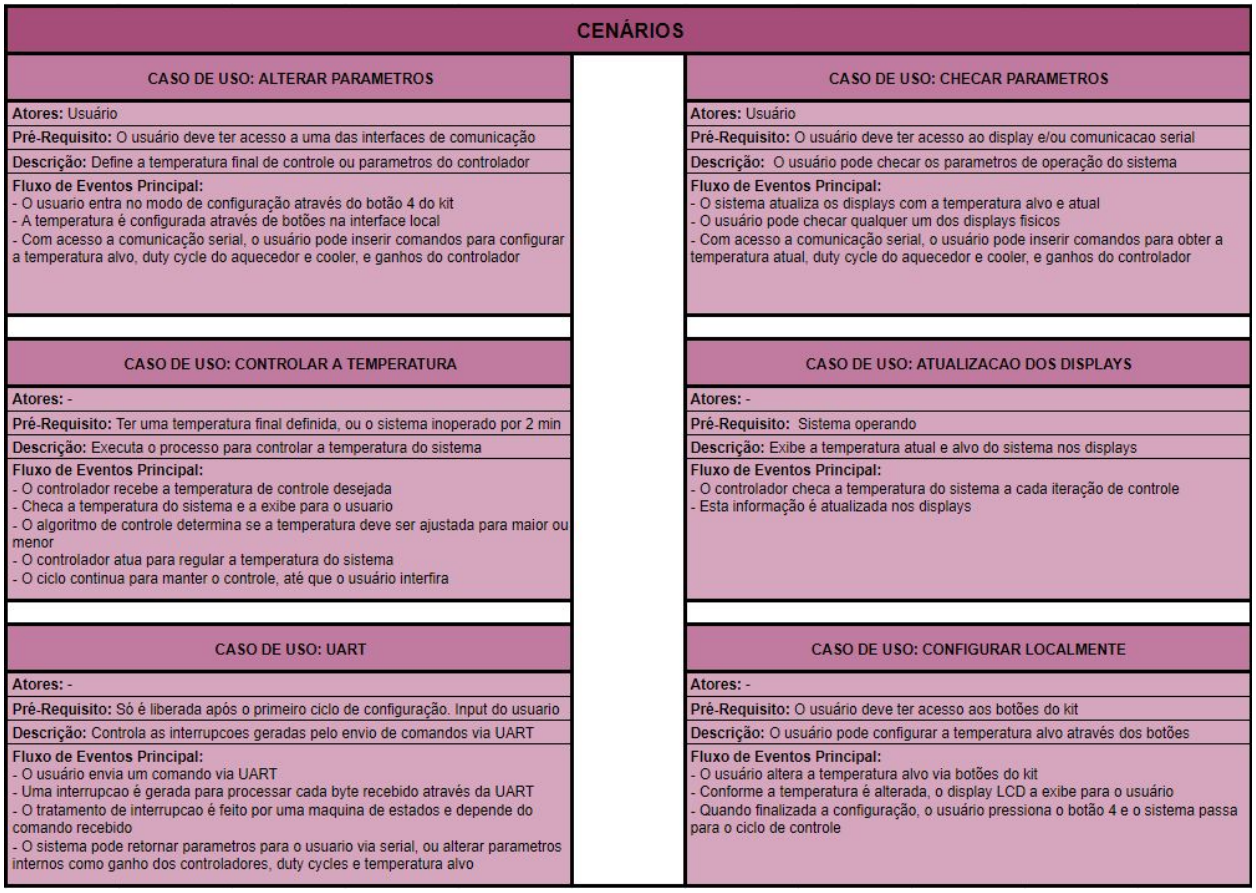


Figura 2 - Diagrama de cenários do Sistema de Controle de Temperatura

3.3 Diagrama de blocos

O diagrama de blocos do sistema traz informações sobre o relacionamento entre seus componentes físicos (hardware). Com ele, podemos observar como as principais peças do sistema interagem entre si, em termos de hardware de entrada e de saída, bem como alguns componentes internos do próprio microcontrolador. Assim, as principais relações para o funcionamento de nosso sistema está representada pelo diagrama a seguir:

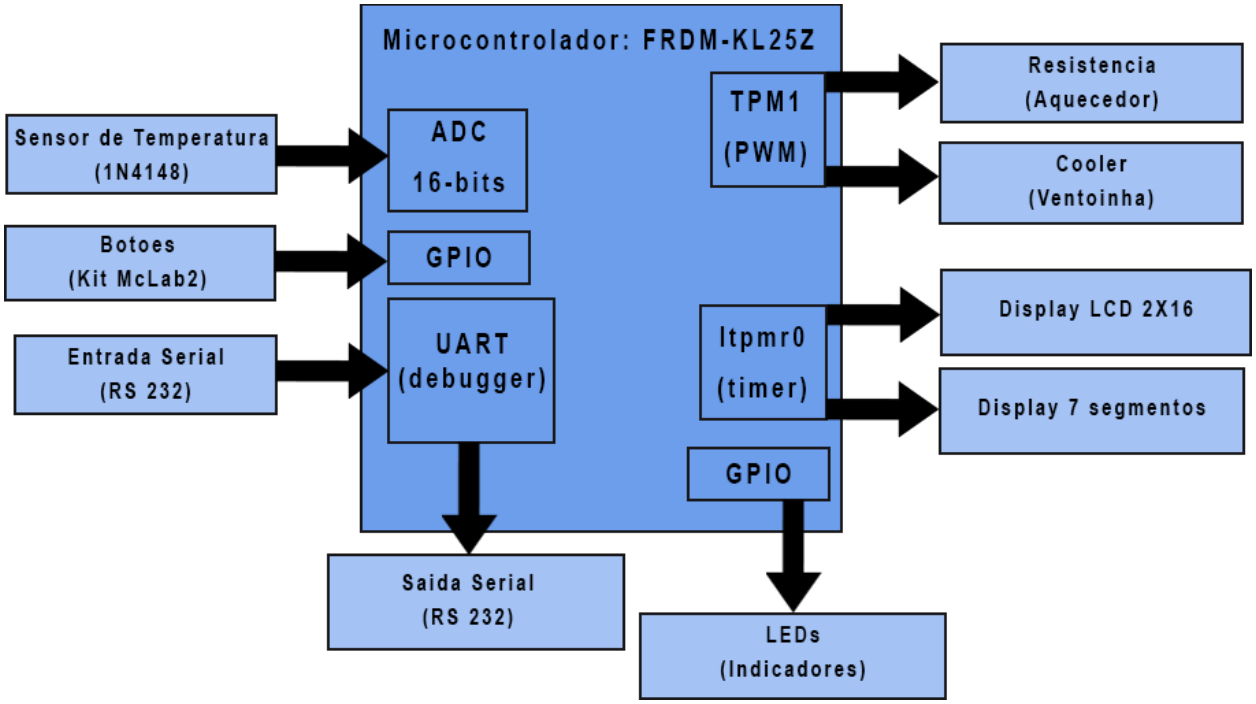


Figura 3 - Diagrama de Blocos do Sistema de Controle de Temperatura

3.4 Fluxograma

A descrição básica da lógica do fluxo de funcionamento do projeto, está descrito na figura 4, através do fluxograma. De início, ao ser ligado o sistema e dado seu start é executada uma rotina pré programada que o coloca em estado de configuração, em seguida será aguardado a inserção do dígito referente a primeiro dezena da temperatura alvo, e caso o usuário não faça isso em até 2 minutos, uma rotina também pré programada irá determinar a temperatura alvo padrão de 30ºC e iniciará o estado de controle.

Inserido o primeiro dígito da temperatura, o usuário terá a atualizar do valor selecionado no LCD e deve pressionar OK para confirmar o dígito e seguir para a inserção do dígito referente a unidade, que funcionará da mesma forma, exceto que no momento em que OK for pressionado, a temperatura alvo será definida e o sistema entrará no estado de controle.

Uma vez no estado de controle, o procedimento do controlador PID estará constantemente atualizando os valores referente ao duty cycle do heater e do cooler até que uma solicitação de configuração seja realizada. A solicitação de configuração será dada com o pressionamento do botão “Reset” da interface de comando, neste caso o sistema retornará para o estado que tinha assim que foi ligado.

Outra opção que é reconhecida como uma solicitação de configuração é causada pela comunicação UART do sistema, neste caso o procedimento de tratamento dessa solicitação será executado e após finalizado o sistema voltará ao seu ciclo de controle, até que seu estado mude.

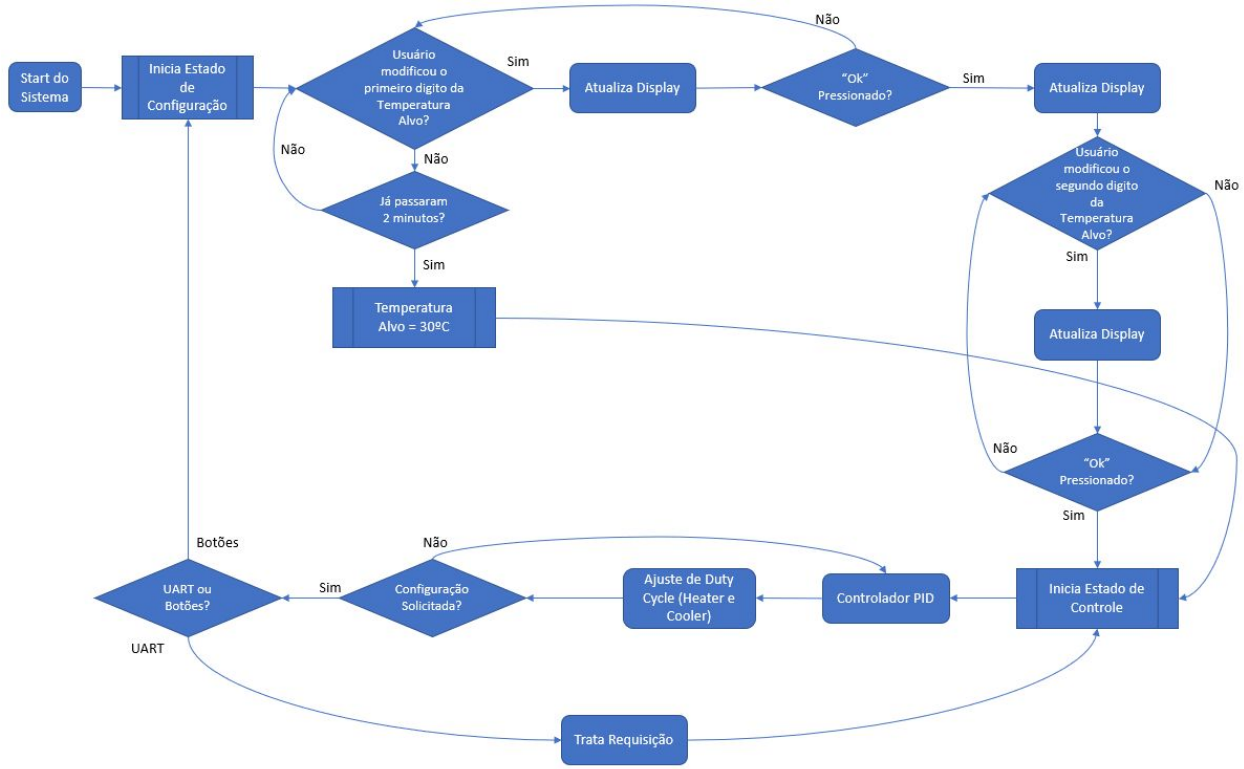


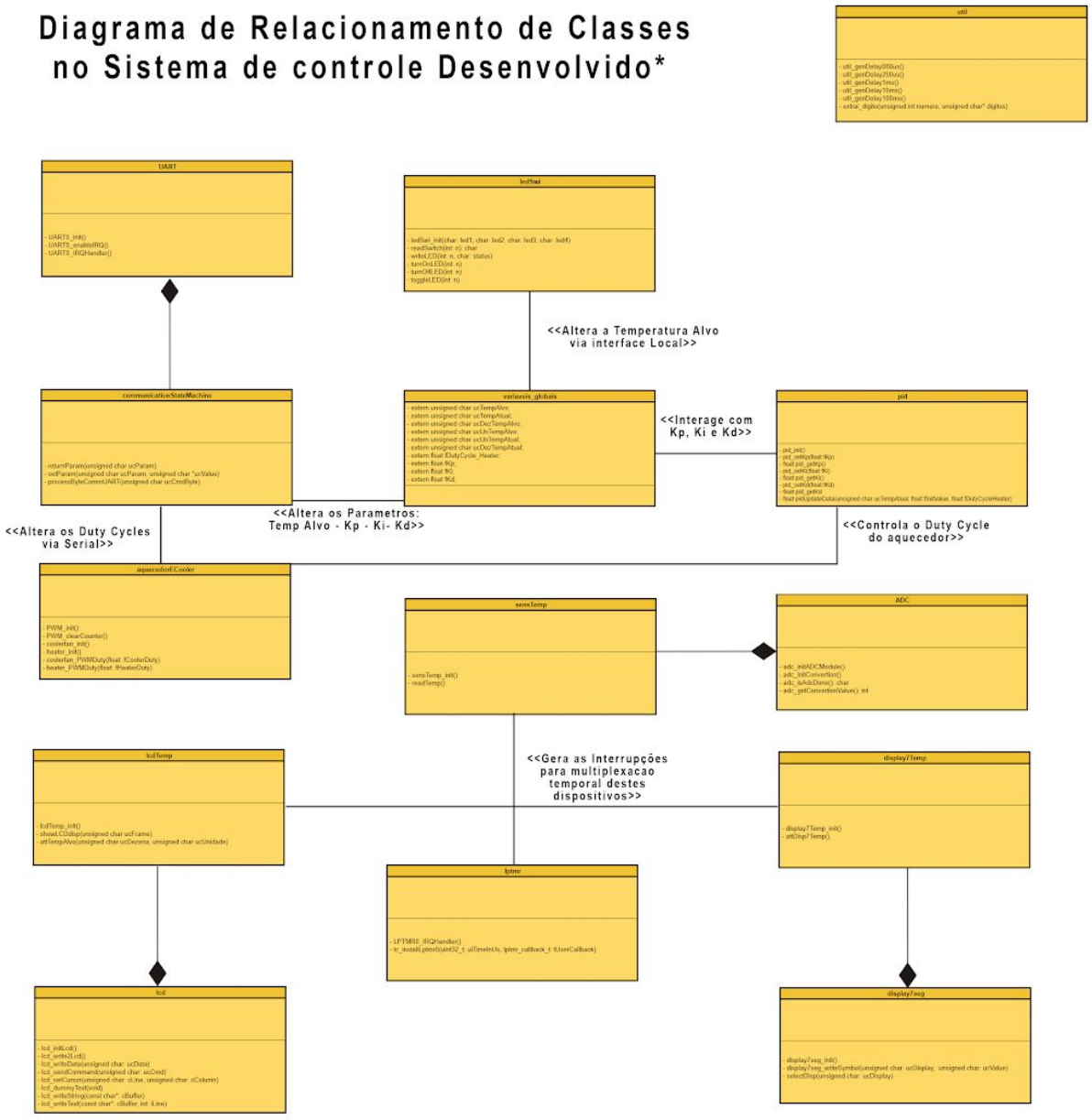
Figura 4 - Fluxograma do Sistema de Controle de Temperatura

3.5 Diagramas UML

3.5.1 Diagrama de Classes

A relação interna do sistema (em termos de código) pode ser representada por um diagrama de classes. Durante o projeto, várias “classes” foram desenvolvidas que facilitaram a integração das funcionalidades do microcontrolador com as interfaces disponíveis ao usuário, assim como as que se relacionam internamente para o funcionamento do sistema. Este diagrama pode ser encontrado na página seguinte. Uma versão em melhor resolução pode ser encontrada no repositório do GitHub associado a este projeto.

É notável que a forma com que este diagrama foi construído pode gerar confusão sobre o funcionamento do sistema. Isto será abordado e discutido na seção 5 deste relatório.



*A classe “util” é utilizada para gerar delays necessários durante o funcionamento do sistema, e por isso não está ligada diretamente a nenhuma outra classe.

Demais classes como “fsl_debug_console”, “lut_adc_3v3” e “print_scan” não estão representadas por não se relacionarem diretamente com o sistema.
A classe “tacometro”, apesar de implementada, não foi utilizada nesta versão do sistema.

Figura 5 - Diagrama de Classes do Sistema de Controle de Temperatura

3.5.2 Diagrama de Sequência

Na página seguinte, está reservado o espaço para figura 6, que representa o diagrama de sequência elaborado para a descrição do sistema. Podemos observar, que enquanto o sistema estiver ligado, o display de 7 segmentos estará constantemente (a cada 4ms) sendo atualizado para permitir, ao olho humano, a visualização da última temperatura lida pelo sensor.

Deve-se notar que o controlador PID só começa a atuar no sistema quando o mesmo sai do estado de configuração inicial, seja pela inserção dos dados de temperatura através dos botões ou após ter se passado mais de 2min sem a inserção do primeiro dígito. O tempo de vida para o aquecedor e para o ventilador também tem início junto ao tempo de vida do controlador.

O loop do estado de controle é quebrado uma vez que uma solicitação de configuração é realizada, como visto no fluxograma já descrito, há duas alternativas para essa solicitação: através da UART ou do botão “Reset” da interface de comando. Finalizada essa etapa, o sistema retorna para o estado inicial e fica neste loop global enquanto estiver ligado.

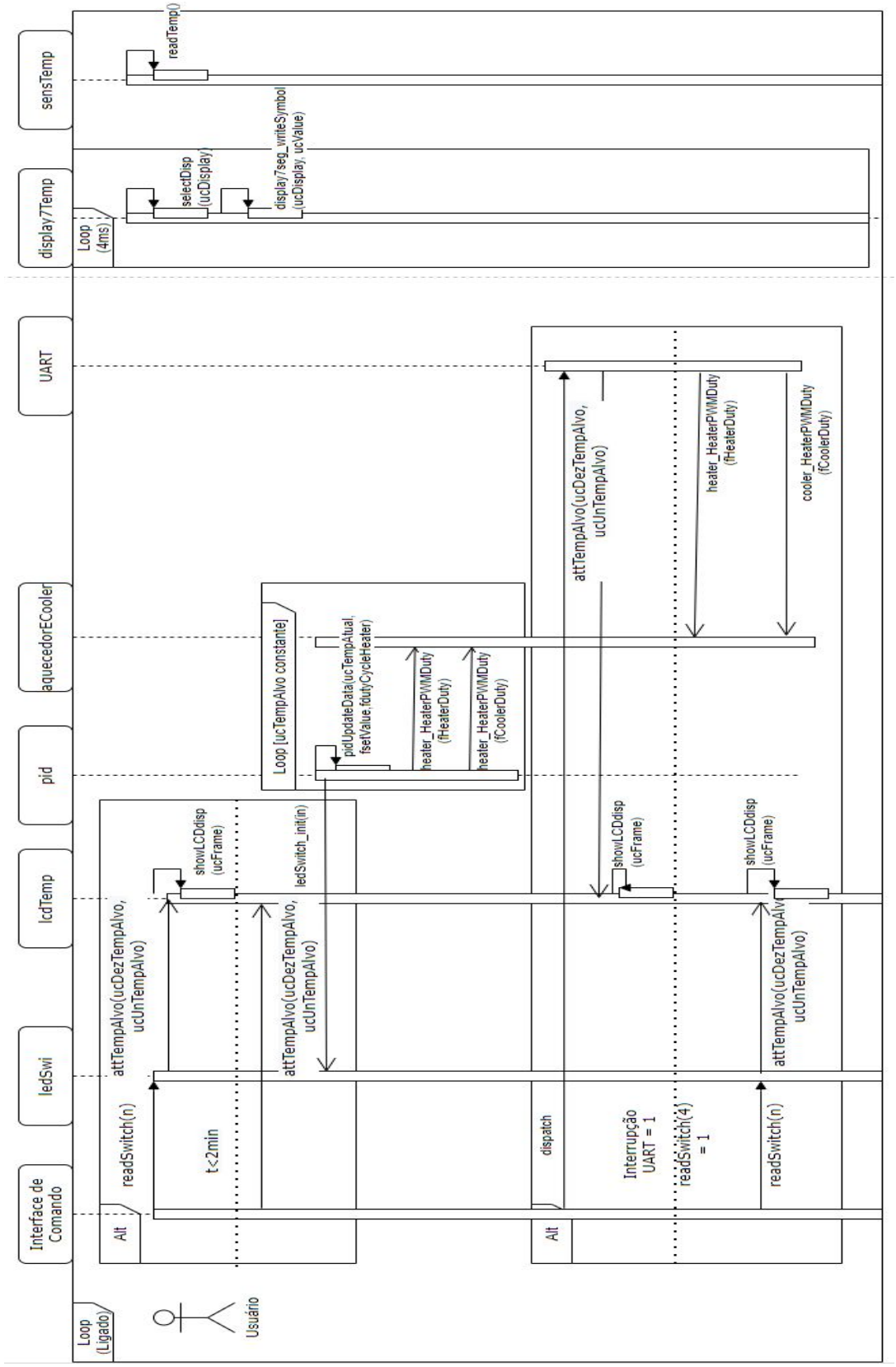


Figura 6 - Diagrama de Sequência

3.5.3 Máquina de Estados

Como a operação do sistema como um todo não permite a tradução direta para o formato de “máquina de estados”, algumas simplificações foram feitas, de forma a permitir esta representação. Para isso, serão apresentadas três diagramas distintos, um representando o sistema geral, e outros dois representando subestados deste.

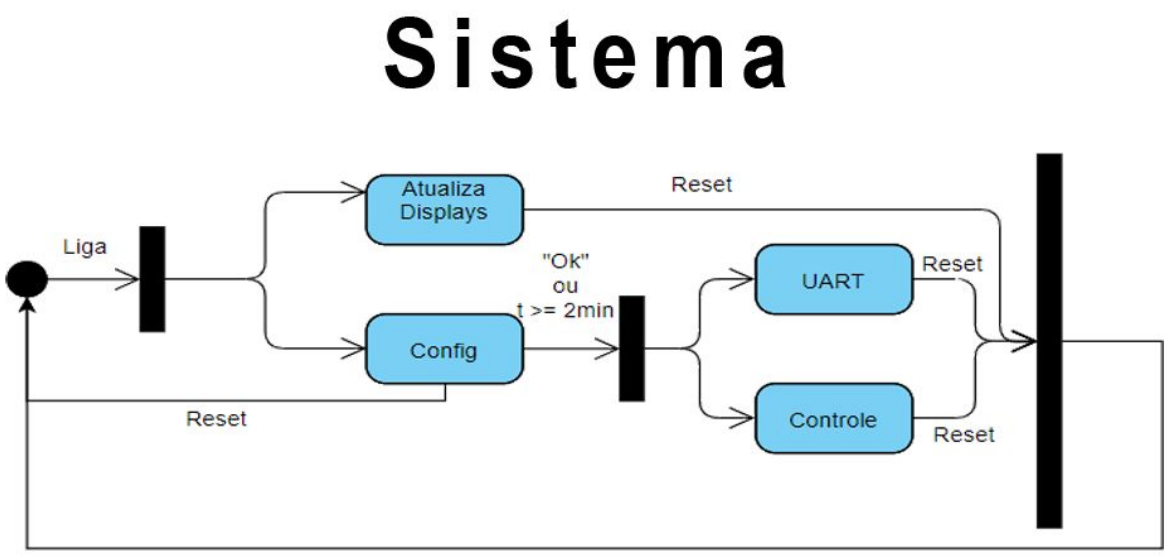


Figura 7 - Máquina de Estados do Sistema de Controle de Temperatura

Note que as “barras de bifurcação” indicam a operação em paralelo de dois estados. Isto representa operações que ocorrem simultaneamente no sistema. Por exemplo, os displays LCD e de 7 segmentos estão sempre sendo atualizados a partir de que o sistema é ligado. Além disso, o tratamento de interrupções gerada pela comunicação serial UART (que é habilitado após o estado de configuração) também ocorre em paralelo, já que só é necessário a partir de uma interrupção externa.

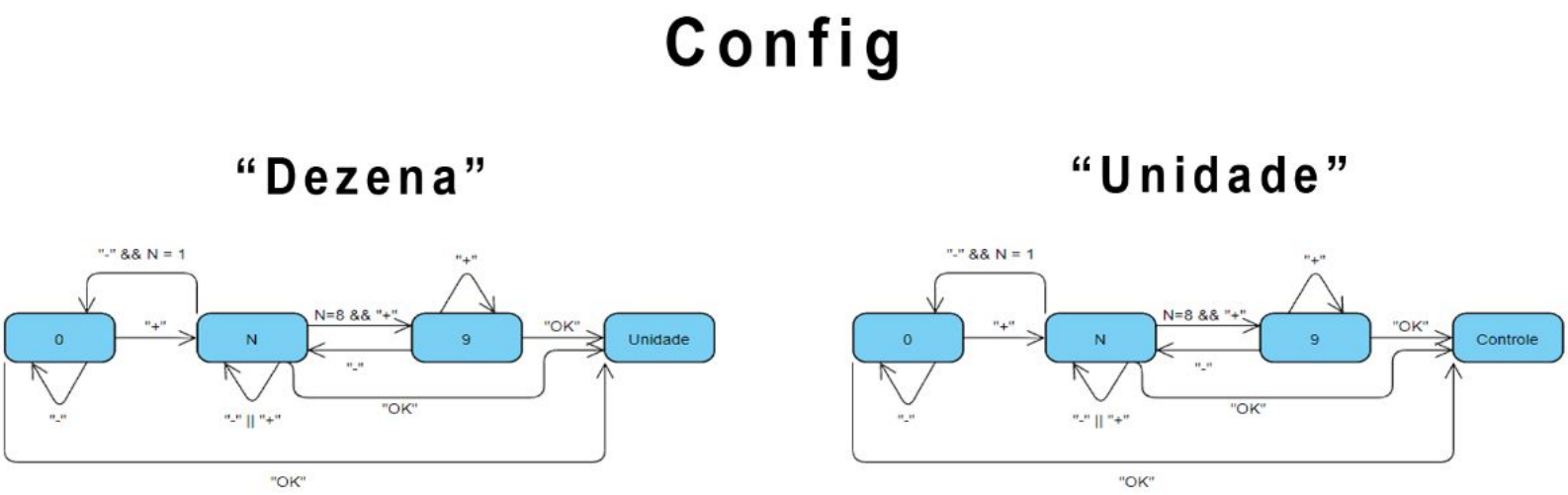


Figura 8 - Subestado “Config” do Sistema

Este subestado representa a configuração via interface local do sistema. Este passo realmente é tratado como uma máquina de estados idêntica a apresentada acima, dividido em estados para configuração de “Dezena” e “Unidade” da temperatura alvo.

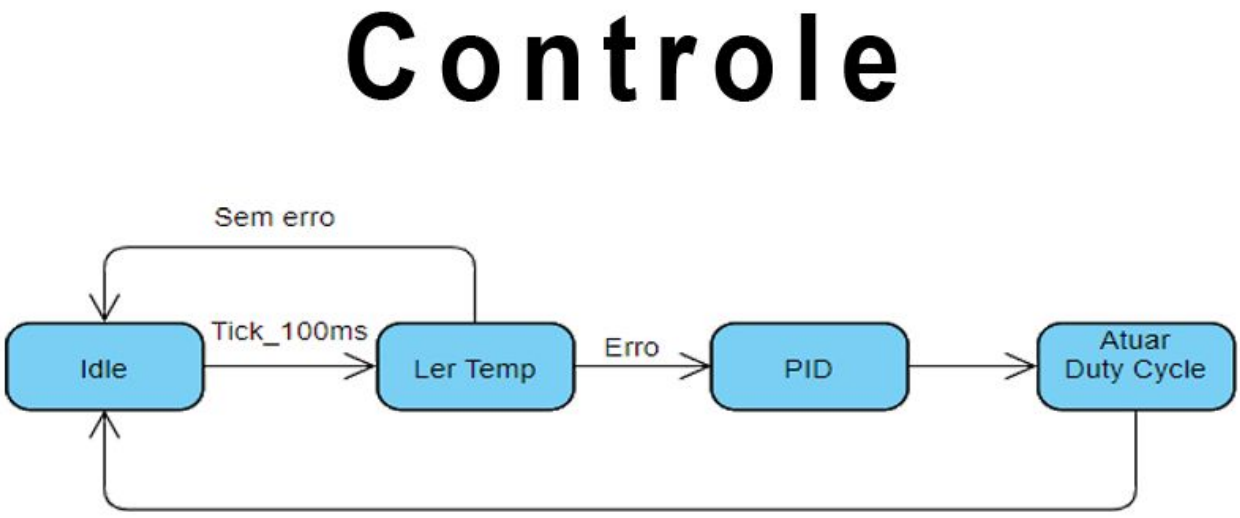


Figura 9 - Subestado “Controle” do Sistema

Este subestado representa o ciclo de controle de temperatura do sistema.

4. Manual de Utilização

O sistema pode ser operado através de duas interfaces distintas:

- Via interface local, pelos botões do kit, a qual é mais limitada, destinada ao usuário final;
- Via interface Serial RS232, a qual permite um poder de manipulação maior sobre as configurações e obtenção de parâmetros do sistema, destinada para testes e debug do sistema.

4.1 Interface Local (Operação via botões)

Ao iniciar o sistema, o primeiro LED do kit deve acender, indicando que está habilitada a configuração de temperatura alvo para o usuário através dos botões. O display de 7 segmentos também é ligado e irá exibir, a todo momento, as leituras de temperatura da resistência.

Também devem ser exibidas as seguintes mensagens no display LCD:

Linha 1: "Configure a Temp";
Linha 2: "Temp Alvo: 00C".

Com isso, os botões 2, 3 e 4 podem ser utilizados para configurar a temperatura alvo que o sistema deve atingir e controlar. Estes botões realizam as operações de aumentar, diminuir e confirmar, respectivamente, conforme mostra a Figura 11.

Primeiramente, o usuário deve configurar a dezena da temperatura alvo, utilizando os botões 2 e 3. Ao confirmar (botão 4), a mesma configuração será realizada para a unidade da temperatura alvo, utilizando os botões 2 e 3.

Ao confirmar novamente, através do botão 4, o sistema passa a controlar a temperatura da resistência para atingir o valor inserido pelo usuário, e a seguinte mensagem deve aparecer no display LCD:

Linha 1: "UART HABILITADO";
Linha 2: "Temp Alvo: xxC",

onde “xx” é a temperatura inserida pelo usuário.

O LED 1 deve se apagar, indicando que o sistema agora está atuando para controlar a temperatura, e que a configuração via botões foi desligada. Os LEDs 2 e 3 indicarão se o sistema está atuando para aumentar a temperatura (ligando o LED 2), diminuir a temperatura (ligando o LED 3 e o cooler) ou se a temperatura está estável (ligando os LEDs 2 e 3).

Neste momento, a interface Serial é habilitada para configurações adicionais. O botão 4 se mantém habilitado para que o usuário possa retornar ao passo de configuração do sistema.

Nota: Se uma temperatura alvo não for definida em uma janela de 2 minutos, uma temperatura padrão de 30 ºC é *setada* e o sistema passa para o controle desta temperatura.

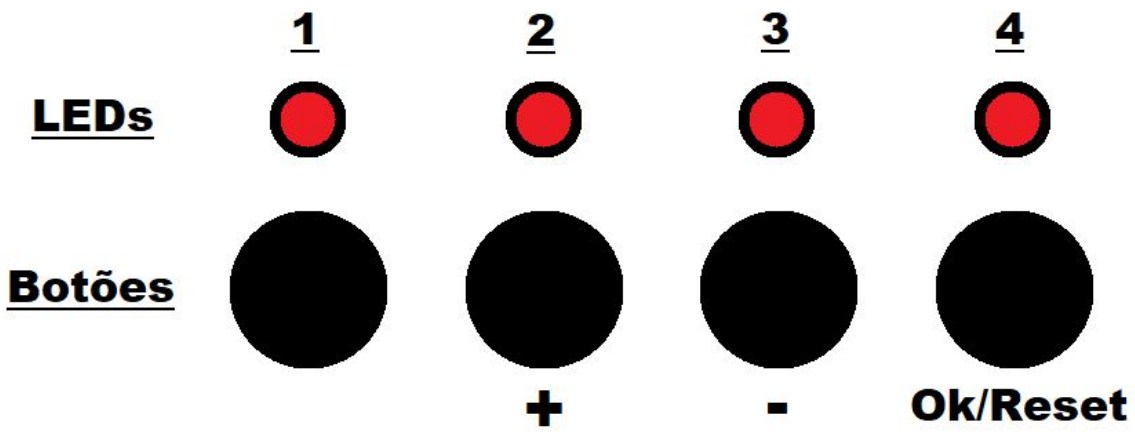


Figura 11 - Ilustração dos botões e LEDs para configuração

4.2 Interface Digital (Operação via Serial RS232)

A operação via Serial só fica disponível quando o sistema passa para o primeiro ciclo de controle, ou seja, após se passar 2 minutos na tela de configuração, ou quando o usuário pressionar o botão 4 (“Ok”) após definir a unidade da temperatura alvo. Quando isso ocorre, o display LCD mostrará, em sua primeira linha, a seguinte mensagem:

“UART HABILITADO”

A partir deste momento, o usuário pode realizar um conjunto de operações sobre o sistema através de uma interface serial conectada ao Kit, via o conector RS232. Esta comunicação pode ser feita, por exemplo, através do terminal Serial PuTTY. Todas as operações devem seguir um padrão específico para que o controlador seja capaz de interpretar corretamente o comando. Estas operações, junto ao padrão, seguem o seguinte formato:

“Comando” | Operação

Comandos do tipo “get”:
"#gt;" Pedido de valor de temperatura atual

"#gc;" Pedido do Duty Cycle do Cooler

"#gh;" Pedido do Duty Cycle do Aquecedor

Comandos do tipo “set”:
"#sb<N>;" Configuração dos botões/leds do kit onde N é qualquer número de até 7 bytes. Para cada dígito de N = 1, o dispositivo é configurada como LED. Se N = 0, é configurado como botão. Qualquer dígito diferente de 0 ou 1 é ignorado. Qualquer entrada com mais de 4 dígitos é ignorada.

"#st<N>;" Configuração Temperatura Máxima desejada para controle, onde <N> é qualquer número de até 7 bytes.

"#sc<N>;" Configuração duty cycle do cooler, onde <N> é qualquer número de até de 7 bytes.

"#sh<N>;" Configuração duty cycle do heater, onde <N> é qualquer número de até de 7 bytes.

"#sp<N>;" Configuração do ganho proporcional do controlador PID, onde <N> é qualquer número de até 7 bytes.

"#si<N>;" Configuração do ganho integrativo do controlador PID, onde <N> é qualquer número de até 7 bytes.

"#sd<N>;" Configuração do ganho derivativo do controlador PID, onde <N> é qualquer número de até 7 bytes.

Respostas:
Dado os comandos do tipo “get”, o controlador retornará uma resposta no formato:

“#a<p>;" Onde “<p>” pode ser o valor de temperatura atual, Duty Cycle do Cooler ou do Aquecedor.

Note que todos os comandos e respostas seguem o padrão de começar com o caractere “#” e terminar com o caractere “;”.

5. Controlador PID

Uma vez que não foi descrito o modelo do sistema de aquecimento e obtenção da temperatura, não possuímos a sua função de transferência, desta forma faz-se necessário analisar como é data a saída do sistema, uma resposta ao degrau, de forma a estimar o controlador para a planta, baseando-se no método de sintonia de controladores desenvolvido Ziegler–Nichols, para malhar aberta.

A fim de obter a resposta ao degrau do sistema projetado, com o aquecedor inicialmente configurado a partir de um duty cycle muito próximo ao zero, mudando bruscamente o seu valor para o máximo permitido (50%), e observando como a temperatura de saída responde a essa simulação de degrau.

Para a configuração dos parâmetros de ganho, o usuário responsável pela tarefa deverá pressionar o botão 4 da interface do microcontrolador, com o projeto “PID” carregado no microcontrolador. A partir deste momento, se dará o início do envio da resposta ao degrau simulado, através da porta de comunicação estabelecida pela UART, com 1250 amostras. Estes valores devem ser coletados do terminal de comunicação, tratados e inseridos no *Matlab* para obter a curva que será usada para aplicação do método de Ziegler–Nichols.

A seguir, será descrito resumidamente como é dado este procedimento, como não houve a possibilidade de durante a execução deste projeto, ter contato direto com o hardware do microcontrolador e sua respectiva interface, iremos simular toda a obtenção dos dados e o tratamento dos mesmo para a aplicação do método.

Um dos pré requisitados para a implementação do método adotado é que a curva de resposta limitada no tempo, que deve ter o formato de “S”, portanto é a primeira assunção realizada do que deverá ser observado em bancada, caso essa realidade não se manifesta outros métodos devem ser estudados. Foi utilizado a função de transferência de um sistema já pré conhecido[1] que possui esse tipo de resposta ao degrau, que usaremos como Base de Simulação, da forma:

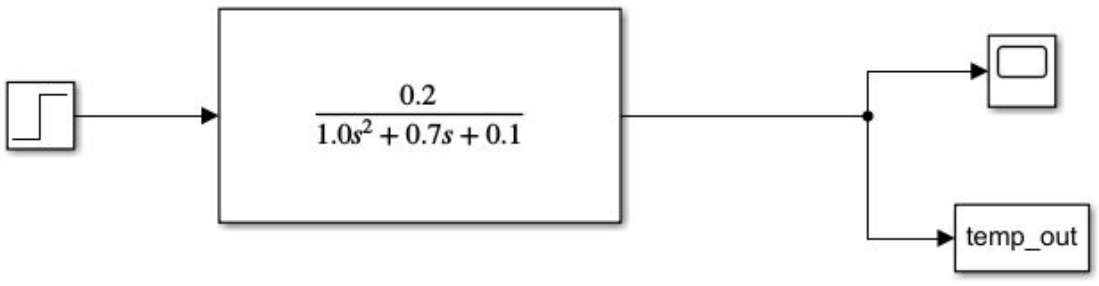


Figura 12 - Sistema (Base de Simulação)

A curva, esperada na prática, e que pode ser observada no bloco Scope do Simulink é apresentada na figura 13, abaixo:

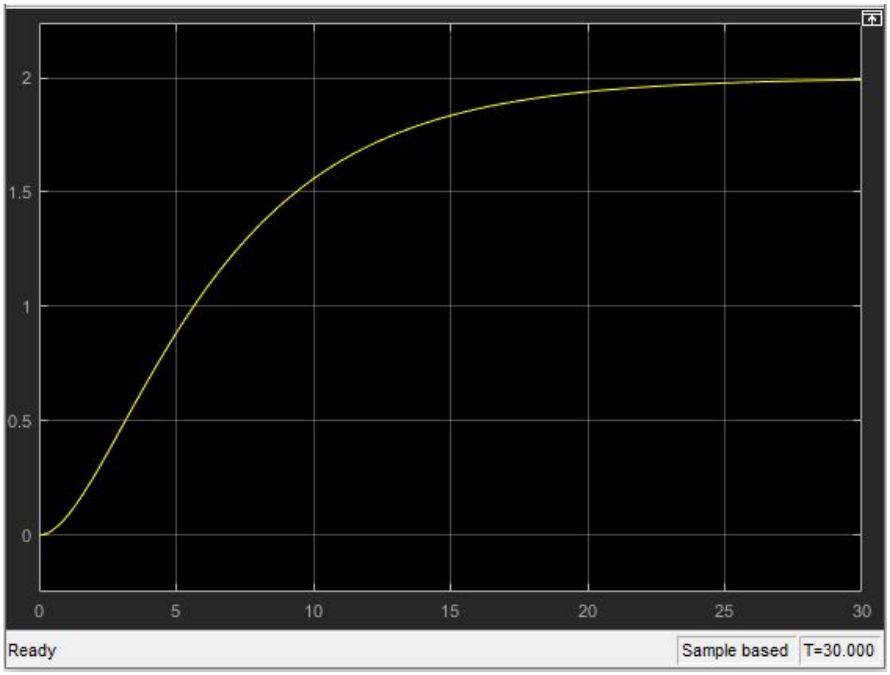


Figura 13 - Curva Formato S (Base de Simulação)

De acordo com a figura 12, a saída para o workspace “temp_out” representa os dados que seriam obtidos pela UART. Com o sinal construído e pronto para ser analisado, iremos encontrar os valores K, L e T que serão usados para a estimativa inicial dos ganhos para o controlador. Tais valores, são definidos como:

- L: Intervalo de tempo definido entre t = 0 e onde o prolongamento da derivada traçada no ponto de inflexão da curva cruza o eixo X.
- K: Valor de estabilização da resposta ao degrau.
- T: Intervalo de tempo definido entre o ponto que a derivada traçada no ponto de inflexão cruza o ponto zero e quando a mesma reta cruza a linha horizontal definida por K.

Visualmente, para o nosso sinal de análise, temos a representação destes parâmetros descritos na figura 14:

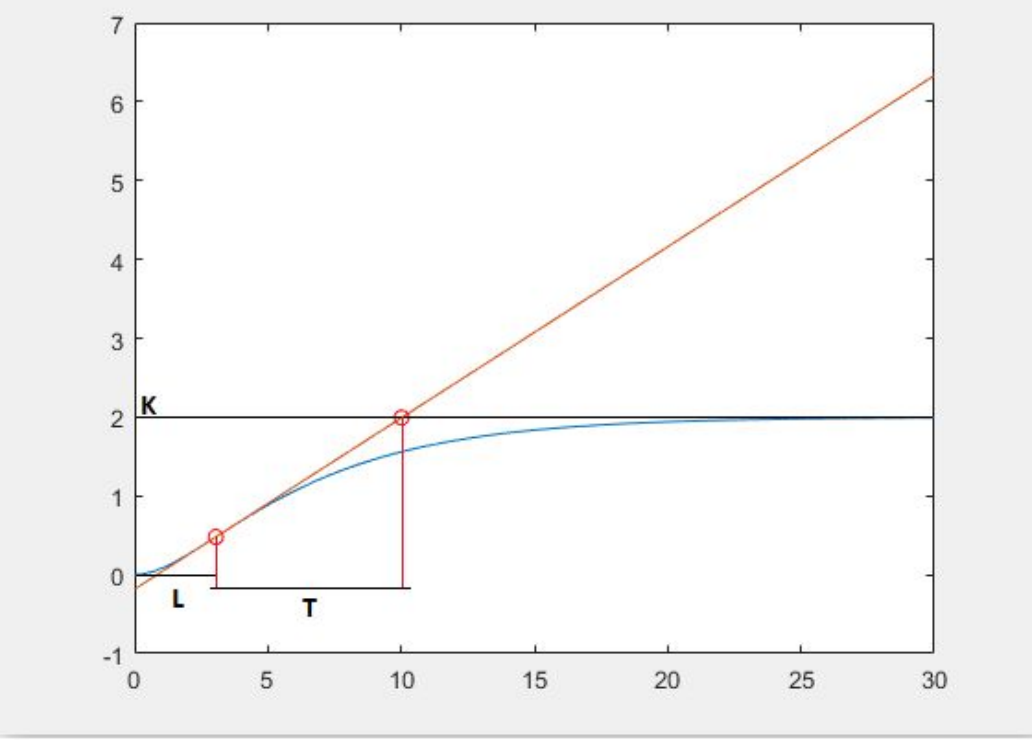


Figura 14 - Obtenção dos parametros de ZN (Base de Simulação)

Para não limitar a análise à inspeção visual, o seguinte script foi elaborado para determinação dos valores K,L e T

```
%%Encontra a tangente da resposta do sistema ao degrau

t = temp_out.time;
y = temp_out.data;
%derivada numérica do vetor de pontos e ponto de inflexão
yp = gradient(y,t);

[m, p] = max(yp);
%equação da linha tangente ao ponto de inflexão

tang = yp(p)*(t-t(p))+y(p);
plot(t,y)
hold on
plot(t, tang)
plot(t(p),y(p),'ro')
hold on
%determinação dos parametros de ZN
L_pos = knnsearch(tang, 0);
L = t(L_pos);
K = max(y);
T_pos = knnsearch(tang, K);
T = t(T_pos) - t(L_pos);

plot(t(T_pos),tang(T_pos),'ro')
```

Figura 15 - Script para análise da resposta

Assim, obtemos K = 1.9917, L= 0.8400 e T = 9.1800. De forma geral, para este método, recomenda-se que a relação de L por T esteja próxima do intervalo $0.1 \leq L/T \leq 0.3$, que é outro ponto de assunção de que os valores obtidos na bancada atenderia, para o sinal base utilizado, temos que $L/T = 0.0915$, o que podemos assumir como aceitável.

Para o controlador PID, temos que $Kp = 1.2 \cdot T/(K \cdot L)$, $Ki = Kp/(2L)$ e $Kd = Kp \cdot L/2$,para a configuração em paralelo do controlador. Assim, com a planta em malha fechada com o controlador descrito, o sinal de saída obtido está representado pela figura 16:

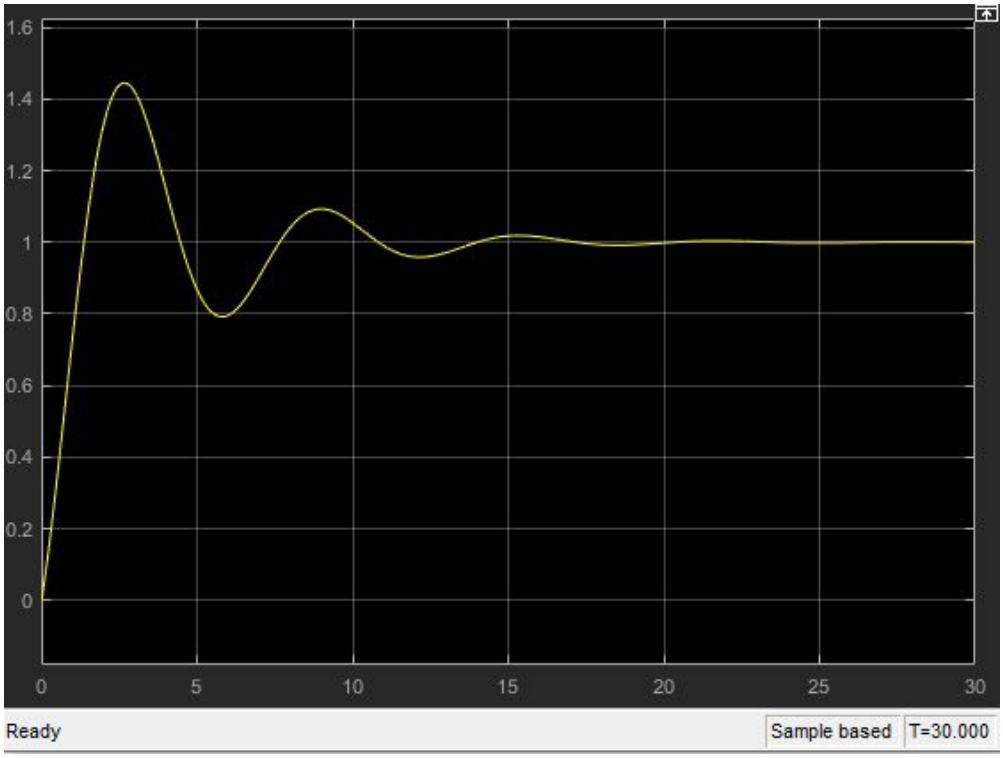


Figura 16 - Resposta controlada (Base de Simulação)

Pode-se observar que apesar de apresentar um overshooting da ordem de 50% , e uma oscilação considerável antes da acomodação, o sistema seguiu a entrada com uma eficiência que merece destaque quando comparado ao sistema não controlado. Aplicando um método heurístico, pode-se buscar melhorar a resposta diminuindo aos poucos o valor de Kp e analisando se a resposta se aproxima do esperado.

6. Problemas Identificados

Nesta seção, serão discutidas as dificuldades encontradas pelos alunos ao longo do projeto, bem como os problemas encontrados pelo Professor nos códigos que contém a implementação dos diferentes módulos. Os mesmos se encontram em seus respectivos *branches* no repositório do Github para referência.

6.1 Problemas identificados pelo Professor

6.1.1 Resolvidos

Aula 6 (ledSwi):

- Adequação do padrão de codificação 7 e 8 do projeto.
- Utilização de variáveis parâmetros do tipo char ao invés do tipo booleana.

Aula 8 (lcd):

- Adequação das constantes utilizadas no board.h (unificado ao final do projeto) para respeitarem o padrão 10 de codificação do projeto.
- No comando "GPIOC_PDOR |= LCD_RS_CMD;" poderia ser considerado valores anteriores, resolvido o problema zerando a saída da porta C responsável pelo pino RS ao entrar na função lcd_write2Lcd, foi inserido na board.h a constante LCD_RS_WAITING
- Alocação de memória para as strings utilizadas na função lcd_writeText.

Aula 10 (display7seg):

- Declaração das constantes utilizadas para configuração do display no arquivo board.h.
- Função tc_installLptmr0 não era chamada, foi inserida junto à seção de inicialização da placa.
- Período da interrupção adequado para milissegundos.

Aula 12 (aquecedorEcooler):

- Alguns pinos do registrador “TPM1_SC” não era colocado em LOW devido ao uso da operação “OU”. O mesmo acontecia com "TPM1_C0SC”. A máscara utilizada para *setar* os pinos foi refeita, e a operação lógica “E” foi utilizada;
- O valor dos registradores “TPM1_MOD” e “TPM1_CnV” não foram alterados. O problema descrito pelo Professor não foi encontrado;

Aula 14 (tacometro):

- O registrador para liberar o clock para a porta E foi trocado para o “SIM_SCGC5” (antes era o “SIM_SCGC6”);
- A liberação de clock para o “TPM0” no registrador "TPM0_SC” foi feita com a máscara correta.
- A transformação da contagem de pulsos em RPM na função “tachometer_readSensor” foi consertada.

Aulas 16 e 18 (UART):

- O tipo do segundo parâmetro da função “setParam” foi alterado de “unsigned char” para “unsigned char*”;
- Foi implementada alterações de diferentes parâmetros dentro da máquina de estados, como temperatura alvo, ganhos do controlador e duty cycle do aquecedor e cooler.

Aula 20 (ADC):

- O codigo foi adequado para atender a regra 6 do padrão de codificação;
- A função “extrai_digito” foi adequada;
- O loop “while(0 != adc_isAdcDone())” foi arrumado em sua posterior utilização na classe “sensTemp”;

6.1.2 Não Resolvidos

Um dos problemas encontrados pelo professor que não foi resolvido é a da adição do “projeto todo” em cada laboratório. Devido a uma falha de interpretação dos alunos, apenas os *branches* do “PID” e do “Projeto Final” contém o projeto do KDS completo.

De maneira geral, problemas relacionados à função “main” em alguns laboratórios não foram diretamente solucionados. Como a refatoração destes arquivos não seria colocada em seus respectivos *branches*, o feedback foi levado em consideração apenas para a elaboração dos novos laboratórios e do projeto final.

6.2 Problemas identificados pelos alunos

6.2.1 Resolvidos

Durante a elaboração do projeto final, surgiram dúvidas quanto a alteração de uma mesma variável por diferentes módulos. Por exemplo, a temperatura alvo deveria ser alterada na função “main” pelos botões, mas também é possível que o usuário a altere via comunicação Serial. Para que não houvesse a necessidade da refatoração destes códigos (externos a “main”) para que incluíssem valores de retorno em suas funções, um conjunto de variáveis globais foi criado. Este conjunto pode ser encontrado no arquivo “variaveis_globais”, onde

estas são declaradas com o prefixo “extern”, tornando-se visíveis e alteráveis para os módulos que incluem esta classe. Outra possível solução para este problema seria a criação de funções de “get” e “set” para estas variáveis, porém não foi implementada.

6.2.2 Não Resolvidos

Um dos problemas identificados mas não resolvidos pelos alunos é a falta de clareza no diagrama de classes. Talvez com a implementação de classes mais específicas ao contexto do projeto faria com que a elaboração do diagrama de classes fosse mais clara e suas relações menos confusas.

Outro problema que não houve possibilidade de ter sido resolvido é a garantia do overshoot de 1ºC e aquecimento mais rápido possível. Apesar da sintonização do controlador ter sido discutida, não há garantia de que estes requisitos serão alcançados sem a apuração com o sistema real.

7. Autoria e Reconhecimento

À parte dos códigos fornecidos pelo próprio professor, não houve códigos copiados ou baseados em implementações de terceiros. Houveram, no entanto, consultas para revisar sintaxe e operação de funções padrão de C, por exemplo, foram feitas em fóruns online como “Stack Overflow”, “Tutorialspoint” e “Geeksforgeeks”. Os códigos fornecidos pelo Professor incluem:

- “adc.c” e “adc.h”
- “lut_adc_3v3.c” e “lut_adc_3v3.h”
- “fsl_debug_console.c”
- “lptmr.c” e “ltpmr.h”
- “print_scan.c” e “print_scan.h”
- “UART.c” e “UART.h”
- “util.c” e “util.h” -- (Houve modificação pelos alunos nestes arquivos)
- “pid.c” e “pid.h” -- (Houve modificação pelos alunos nestes arquivos)

Além destes códigos, a máquina de estados que implementa o protocolo de comunicação serial em “communicationStateMachine” foi baseada no código apresentado pelo Professor durante a aula 17.

8. Referências Bibliográficas

[1] - Castaño S., “ZIEGLER e NICHOLS Malha Aberta [Controle PID] Parte 1 - Disponível em: https://www.youtube.com/watch?v=wmyQJnTpOIs

[2] - Ogata K., “*Modern control engineering*” - 5ed, 2011

[3] - Freescale Semiconductor, *KL25 Sub-Family Reference Manual* - Rev. 3, 2012