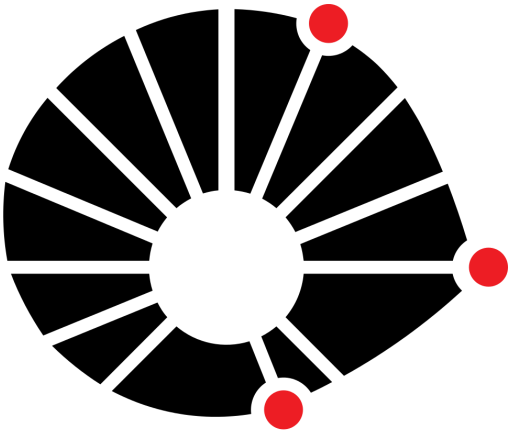
ES 670

Projeto de Sistemas Embarcados

Controlador de Temperatura



UNICAMP

Caio Villela - 168342
Hebert Wandick - 174335

Resumo

A proposta para o projeto fora o desenvolvimento de um sistema embarcado de controle de temperatura, a partir do uso do kit de desenvolvimento FRDM KL25 acoplado à placa de periféricos MCLAB2.

Através do sistema desenvolvido pelo grupo, é possível setar uma temperatura entre ambiente e 77°C, com baixo overshoot e rápido aquecimento. Optou-se pela utilização de um *cooler* operando a velocidade constante, de forma a facilitar o resfriamento do sistema.

A interface local, composta de um LCD e botões, pode ser utilizada para mostrar e setar a temperatura (*set point*) e ganhos *Kp, Ki e Kd* do controlador PID implementado. Além disso, ele mostra também a temperatura atual do protótipo, como lida pelo sensor.

Outros periféricos empregados para o desenvolvimento foram comunicação UART, utilizada para definir o *setpoint* de temperatura, habilitar e desabilitar a interface local e para realizar a o controle dos duty cycle do aquecedor e cooler, assim como da temperatura atual. Empregou-se também conversores AD para o controle da temperatura.

Para a correta utilização do projeto desenvolvido, favor atentar-se aos comandos explicados nas próximas seções do relatório.

Documentação do Sistema

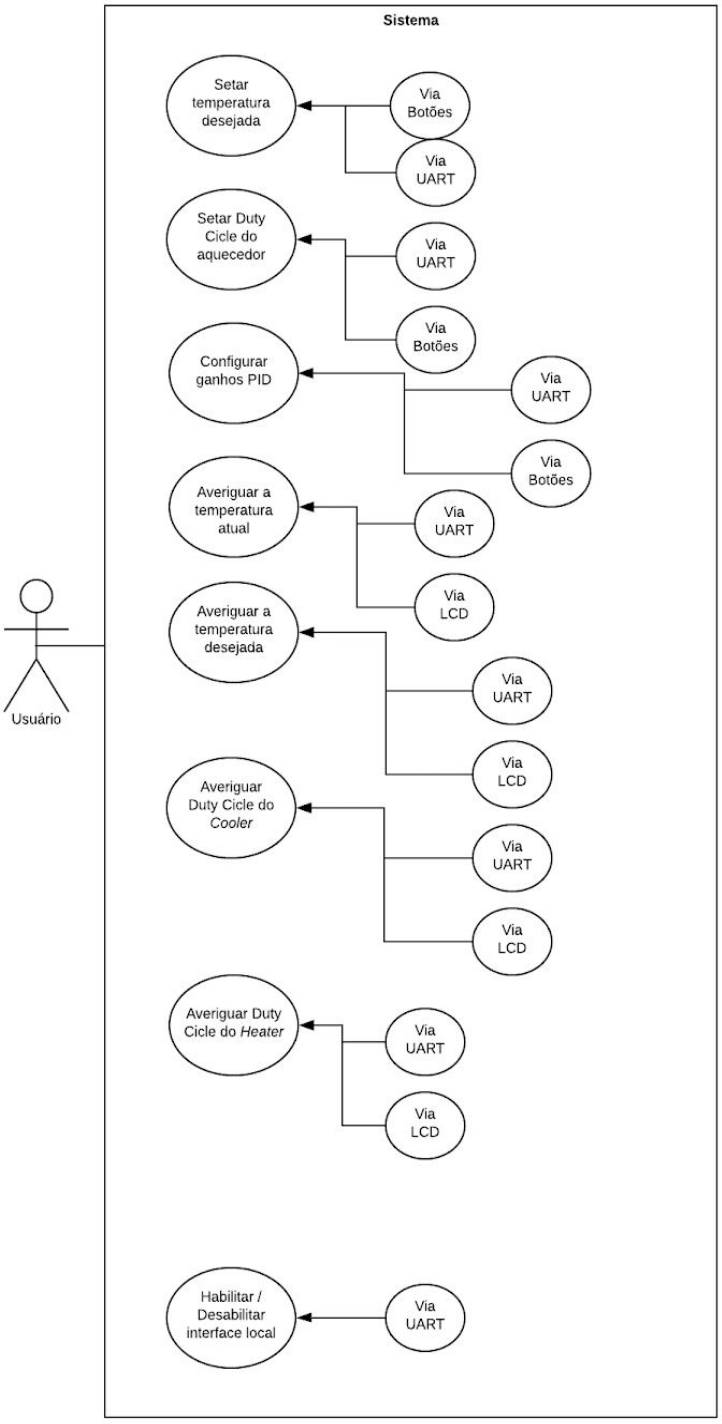
A tabela de requisitos apresenta o que o sistema em questão se propõe a fazer. Isso inclui funcionalidades explícitas (requisitos funcionais) e funcionalidades ligadas à sua implementação (requisitos não funcionais).

O diagrama de casos de uso, bem como os cenários, mostram a relação entre o usuário e o sistema. Nos diagramas de cenário foram descritos dois procedimentos que podem vir a ser úteis para o primeiro contato com o sistema. Como os casos de aplicação são muitos, recomenda-se utilizar esses cenários para um primeiro contato com o sistema, e que se busque demais comandos na tabela anexada à máquina de estados.

Tabela de Requisitos

Requesitos Funcionais	Requesitos Não Funcionais
O sistema deverá permitir controle por comandos seriais em uma interface UART	O display de informações se dará via display LCD 16x2
O sistema deverá exibir a temperatura atual	O sistema configurará a temperatura até 90 graus
O *overshoot* máximo deve ser de 1°C.	A interface local será manipulada via botões
O sistema deverá oferecer uma interface local para operação	Se utilizará controle PID para manutenção da temperatura
O sistema deverá exibir diversos parâmetros quando requisitados	Se utilizará um diodo de silício como sensor de temperatura, bem como um tacômetro para medir a rotação do *cooler*
O duty cycle do Cooler deverá ser setado manualmente	O Heater deverá ter um Duty Cycle máximo de 50%

Casos de Uso e Cenário



Caso de Uso: Regular PID (via botões)
Ator: Usuário
Pré-requisito: -
Fluxo de Eventos Principal: **1.** Aparelho em modo standby (estado IDLE) **2.** Usuário deve apertar o botão 1 do kit **3.** O "goal" de temperatura será mostrado na tela, e deve ser configurado utilizando-se os botões 2 e 3 para aumentá-lo ou subtraí-lo. Confirmar mudança com o botão 1. Os valores de temperatura atual começarão a ser enviados via UART a uma frequência de 10 Hz **4.** Os ganhos PID serão mostrados na tela (um por vez) , e devem ser configurado utilizando-se os botões 2 e 3 para aumentá-los ou subtraí-los. Confirmar mudança com o botão 1. **5.** O duty cycle do cooler será mostrado na tela, e deve ser configurado utilizando-se os botões 2 e 3 para aumentá-lo ou subtraí-lo. Confirmar mudança com o botão 1. **6.** Aparelho volta ao estado standby. **7.** Fim de caso de uso.

Caso de Uso: Realizar leitura do Duty Cycle do Aquecedor (via UART)
Ator: Usuário
Pré-requisito: -
Fluxo de Eventos Principal: **1.** Aparelho em modo standby (estado IDLE) **2.** O usuário deverá pressionar '**#**' (passando ao estado READY) **3.** O usuário deverá pressionar '**g**' (passando ao estado GET) **4.** O usuário deverá pressionar '**a** (passando o parâmetro Duty Cycle do aquecedor ao estado) **5.** O usuário deverá pressionar '**;**' , finalizando a operação. Ele receberá então o valor do parâmetro Duty Cycle via UART **6.** Aparelho volta ao estado standby. **7.** Fim de caso de uso.

Fazer uso da tabela anexa à Máquina de Estados para compreensão de todos os comandos sequenciais.

Máquina de Estados

A Máquina de Estados do sistema dita o seu comportamento a depender dos comandos enviados para ele e do estado em que ela se encontra. É importante atentar-se aos comandos descritos na tabela acima para a correta operação do sistema, as cores representam qual informação está disponível no lcd para aquele estado.

Máquina de Estados :

	Temperatura atual	Duty Cycle Cooler	Duty Cycle Aquecedor	Habilitar/Desabilitar Botões	Ki	Kp	Kd	Temperatura desejada
get	**#gt;**	**#gc;**	**#ga;**	**#gb;**	**#gi;**	**#gp;**	**#gd;**	**#gs;**
set	-	**#sc<V>, v == float**	-	**#sb<v>;** , v == {0, 1}	**#si<v>;** ,v == float	**#sp<v>;** ,v == float	**#sd<v>;** ,v == float	**#ss<v>;** , v == float
answer	**#a<v>,** v == float	**#a<v>**, v == float	**#a<v>**, v == float	-	**#a<v>**, v == float	**#a<v>**, v == float	**#a<v>**, v == float	**#a<v>**, v == float

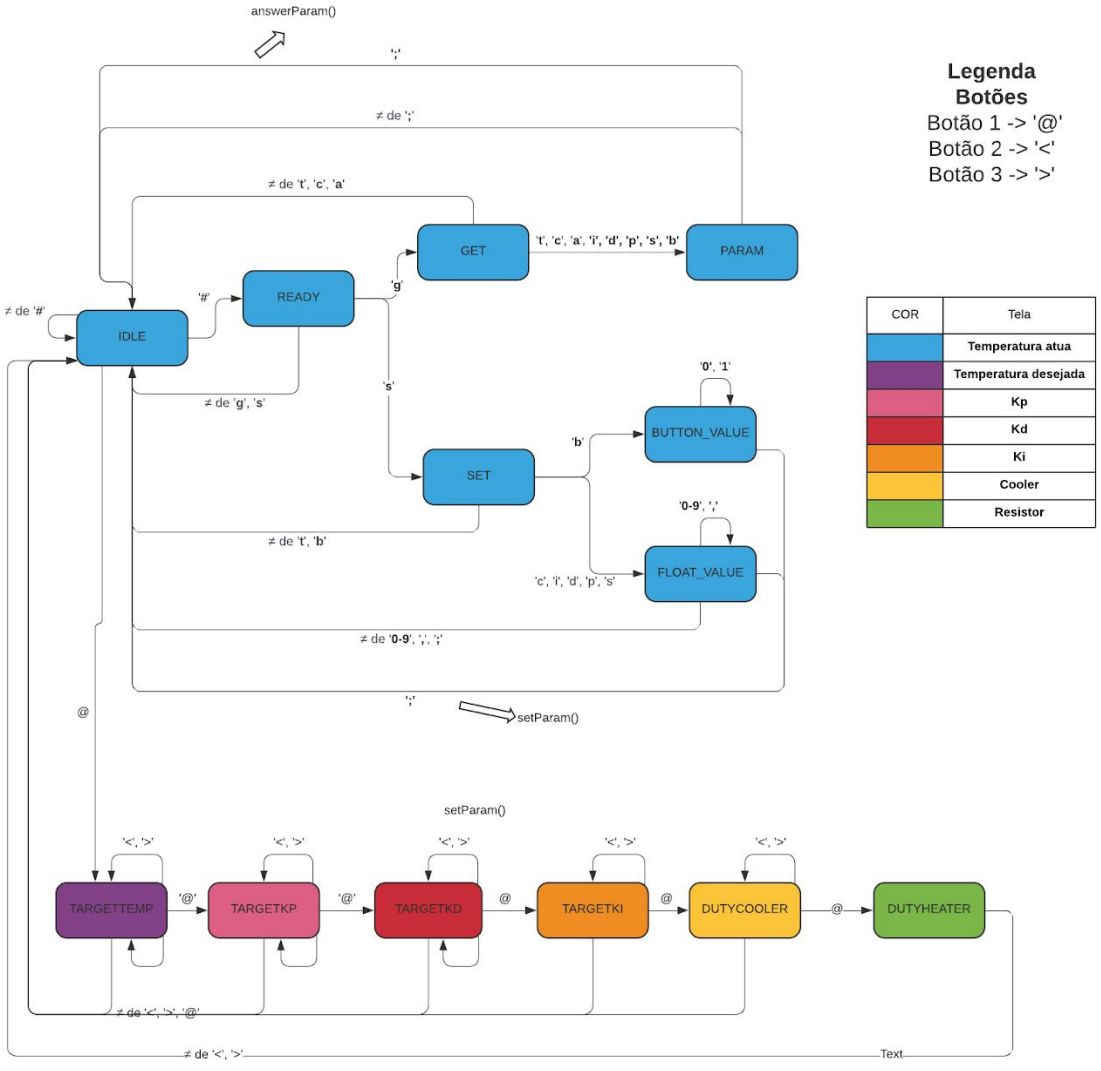


Diagrama de fluxo:

O diagrama de fluxo acaba ficando praticamente redundante uma vez que o diagrama da máquina de estado representa fielmente todas as etapas e fluxos de entrada e saída de informação presente no equipamento.

Fluxograma:

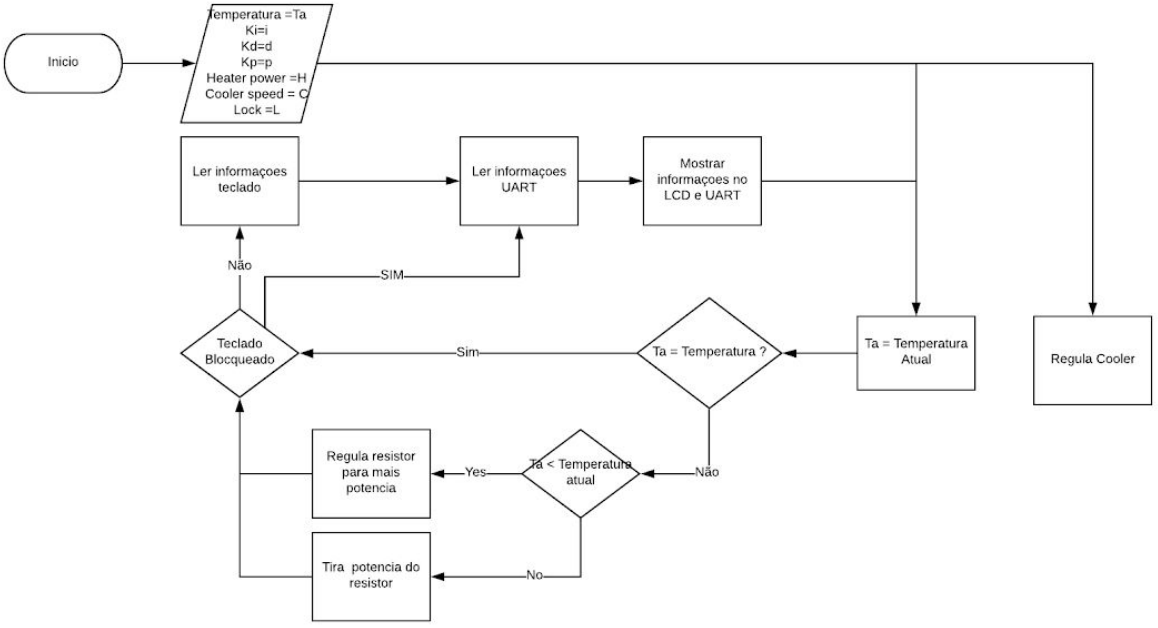


Diagrama de Sequência:

O diagrama de Sequência mostra a colaboração entre diferentes objetos e atores do sistema, dando ênfase à ordenação temporal em que as mensagens são trocadas.

Caio Simão Villela - RA 168342
Hebert Wandick Parreira - RA 174335

Diagrama de Sequência

caso de uso: realizar a leitura do duty cyle do resistor via UART

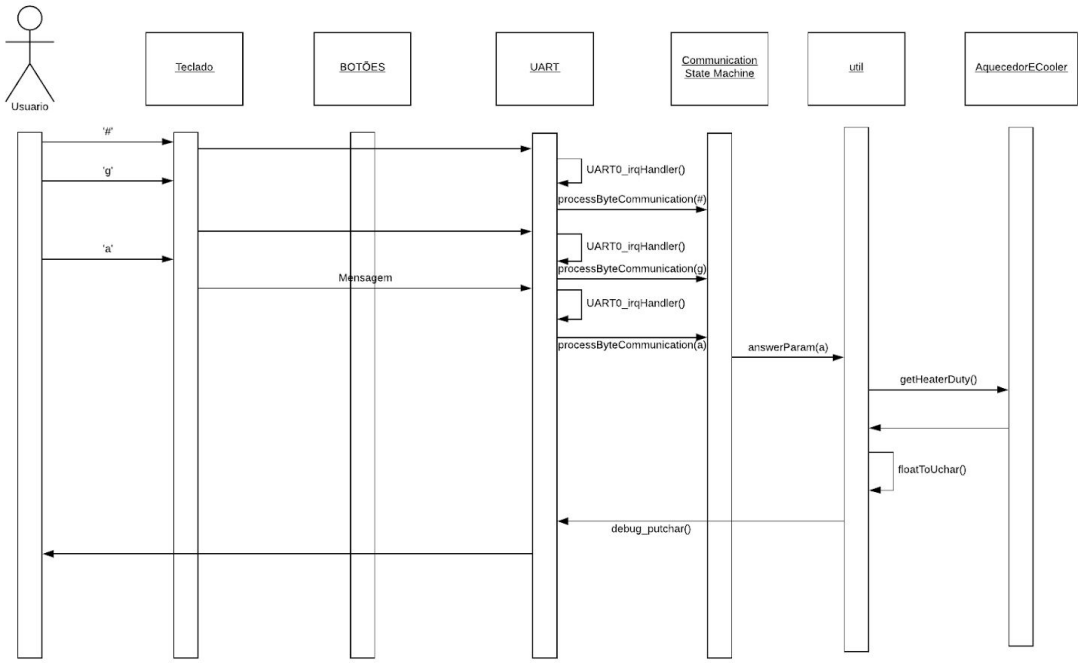
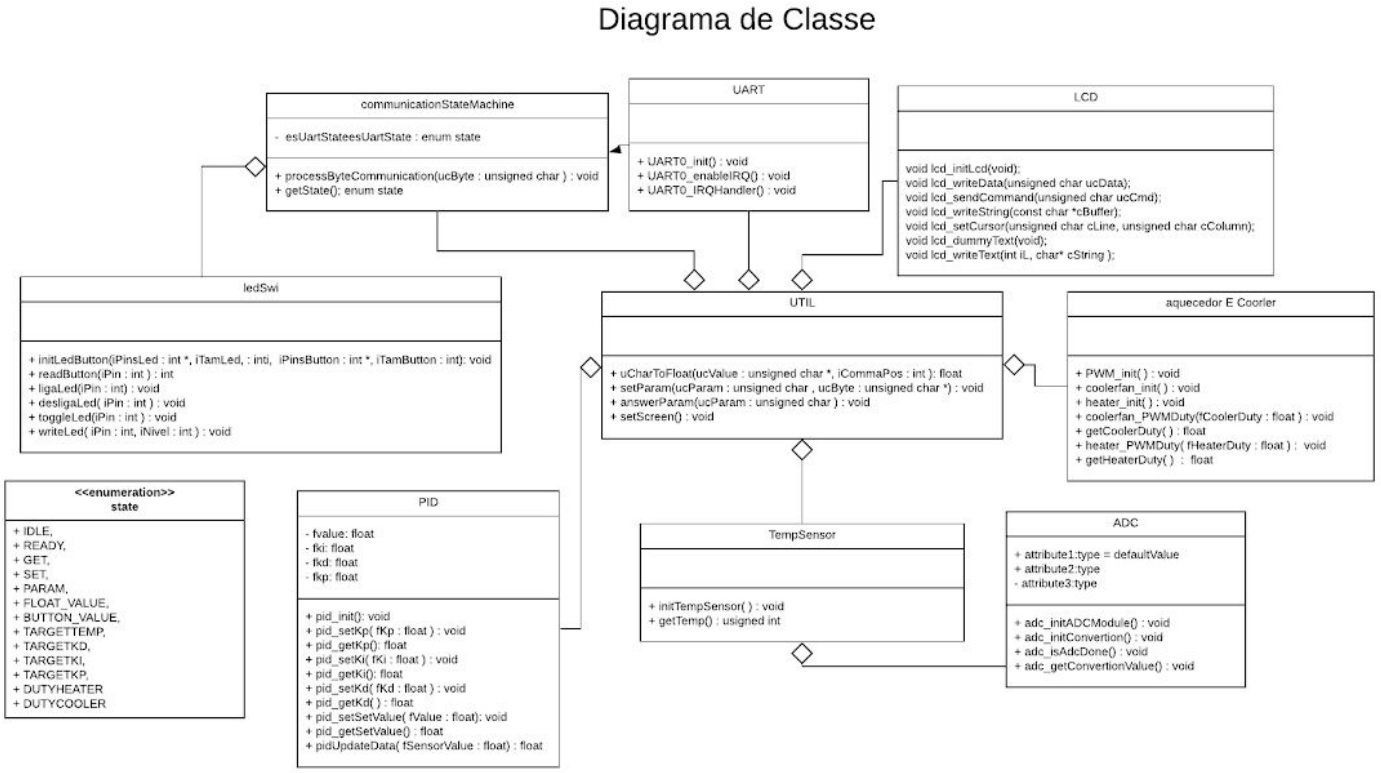


Diagrama de classes :

O diagrama de estados mostra as relações entre cada módulo do sistema, mostrando suas interfaces e sua conexões.



Manual de Utilização

Notas:

- A máquina não possui memória, portanto o estado de início de operação sempre será os seguintes;
    - Temperatura alvo = 20ºC
    - Constante de integração no controle PID Ki=0;
    - Constante de derivativa no controle PID Kd=0;
    - Constante de proporção no controle PID Kp=0;
    - Com o teclado destravado

- O aquecedor nunca ultrapassa 50% de utilização (duty cycle <=50%) por questões de segurança;
- Caso o equipamento esteja sendo configurado via interface local (botões) e haja uma tentativa de configuração via UART, o comando não funcionará, e a interface local voltará a mostrar a temperatura atual;
- O teclado só poderá ser travado/destravado via interface UART;
- O controlador PID controla apenas o resistor, o cooler deve ter sua velocidade setada manualmente;
- A UART espera receber um número float no formato decimal com 3 casas antes da vírgula e duas após, para facilitar a utilização em terminais seriais simples (sem acesso a caracteres especiais, como é o caso da IDE do arduino), porém o retorno de float via UART é feito mandando byte a byte do float;

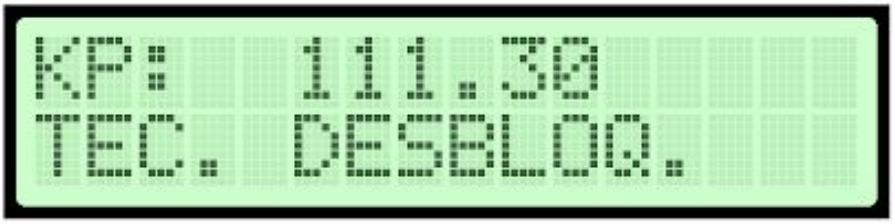
Utilização na Interface Local

Ao iniciar a operação do equipamento ele se apresenta da seguinte forma:

Para que o operador possa, configurar o equipamento ele deve apertar a tecla de “SET”, o que levará a seguinte tela:

Nesta tela o operador poderia aumentar ou diminuir a temperatura a ser controlada pelo equipamento cada clique nas respectivas teclas(UP/DOW) levará a um aumento de 1ºC(um grau) na temperatura desejada;

Ao apertar SET novamente o usuário será levado a seguinte tela:



Nesta tela o operador poderia aumentar ou diminuir o valor da constante KP cada clique nas respectivas teclas levará a um aumento ou diminuição de de 2(dois)

Ao apertar SET novamente o usuário será levado a seguinte tela:

Nesta tela o operador poderia aumentar ou diminuir o valor da constante KD cada clique nas respectivas teclas levará a um aumento ou diminuição de de 2(dois);

Ao apertar SET novamente o usuário será levado a seguinte tela:

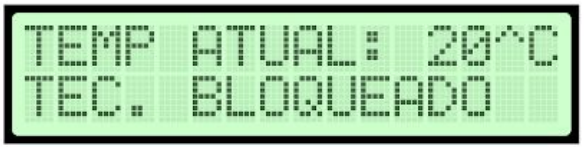
Nesta tela o operador poderia aumentar ou diminuir o valor da constante KI cada clique nas respectivas teclas levará a um aumento ou diminuição de de 2(dois)

Ao apertar SET novamente o usuário será levado a seguinte tela:

Nela o usuário poderá configurar a velocidade do FAN aumentando ou diminuído em 1% a velocidade;
Ao apertar SET novamente o usuário será levado a seguinte tela:

Nela o usuário só poderá ver a potência atual do resistor, já que esse controle é feito pelo controle PID embutido no sistema;
Ao apertar a tecla SET o sistema retornará ao início do ciclo

As teclas para aumentar e abaixar os parâmetros de operação podem ser travadas, porém as tecla de SET continuará operando permitindo a visualização de todos os parâmetros do equipamento, quando elas estiverem travadas a mensagem de “TEC. BLOC” aparecerá na segunda linha da tela LCD, comos nos dois exemplos a seguir.



Quando eles estiverem operacionais

Utilização na UART

Para utilizar a interface UART o usuário deve conectar seu computador através de sua conexão serial, ou através de um conversor USB-Serial.
No computador o usuário pode utilizar qualquer simples programa para comunicação serial que preferir, como o putty a interface serial do Arduino IDE etc ...

- A interface UART manda de segundo em segundo a temperatura atual;

A comunicação acontece através do envio de códigos específicos para o equipamento, abaixo a uma tabela contendo todos os comandos disponíveis;

AS informações disponíveis para controle e checagem O comandos disponíveis são os seguintes

Informação	GET	SET	Tipo da saída/ entrada	Descrição
Temperatura atual	✔	❌	Float	Mostra a temperatura medida no sensor de temperatura em tempo real em ºC
Exemplo	#gt;			
Duty Cycle Cooler	✔	✔ #sc<V>;	Float	Mostra a velocidade em porcentagem da velocidade máxima de (0~100%)
Exemplo	#gc;	#sc075,40;		
Duty Cycle Aquecedor	✔	❌	Float	Mostra a porcentagem de potência dissipada no cooler (0%50%)
Exemplo	#ga			
Habilitar/Desabilitar Botões	❌	✔	Byte	Mostra se o equipamento pode ser configurado via interface local ou não, 0 para desabilitar, 1 para habilitar;
Exemplo		#sb1;		
Ki	✔	✔ #si<V>;	Float	Responsável pela parte de integração do controlador PID embutido no equipamento
Exemplo	#gi;	#si100.00;		
Kp	✔ #gp;	✔ #sp<V>;	Float	Responsável pela parte proporcional do controlador PID embutido no equipamento
Exemplo	#gp;	#sp123.40;		
Kd	✔ #gd;	✔ #sd<V>;	Float	Para responsável pela parte derivativa do controlador PID embutido no equipamento
Exemplo	#gd;	#sd080.30;		
Temperatura	✔ #gs;	✔ #ss<V>;	Float	É a temperatura que se deseja manter o sistema

desejada				
Exemplo	#gs;	#ss070.30;		

Problemas identificados e não resolvidos

Visto que não pudemos fazer uso do kit McLab2 disponibilizado na universidade, tivemos alguns problemas em identificar problemas não resolvidos. No entanto há algumas funcionalidades que, justamente não terem sido testadas, merecem alguma atenção neste tópico.

O overshoot máximo deve ser de 1°C e o sistema deve aquecer o mais rápido possível.

O controle PID do sistema foi implementado corretamente, assim como o método de configuração dos ganhos, no entanto não houve como verificar o overshoot nem a resposta do sistema ao controle.

A temperatura aparece em valores decimais

Devido à implementação da *lookup table,* que retorna os valores em inteiros, não obtivemos sucesso em garantir precisão decimal do sensor de temperatura.

Autoria dos Códigos Fornecidos

Os arquivos, adc.(h / c), lut_adc_3v3.(c / h) tem os seguintes autores, **dloubach, julioalvesMS, lagoAF e rbacurau.**

Os arquivos, aquecedorECooler.(c / h), board.h,lcd.(c / h), ledSwi.(h / c), main.c e TempSensor.(c/h) tem os seguintes autores, **Caio Villela, Hebert Wandick;**

Os arquivos, fsl_debug_console.c, print_scan.(c / h) :foram feitos pela própria **Freescale Semiconductor**
Os arquivos lptmr.(c /h),util.(c/h) foram feitos diretamente pelo professor somente **dloubach**

Por fim UART.(c/ h) from feitos por **dloubach e rbacurau**