ES670 Projeto de Sistemas Embarcados
Projeto Final

June 23, 2024

Nome	**RA**
Isabelle Miki Ikuno	173336
Pedro Henrique Limeira da Cruz	215663

Contents

1 Resumo

O sistema desenvolvido é um controlador de temperatura. Os principais componentes presentes são um microcontrolador STM32, um aquecedor, um cooler, um sensor de temperatura, um tacômetro, uma tela LCD, um buzzer, botões e um teclado matricial.

O objetivo desse dispositivo é aquecer o resistor e mantê-lo em uma temperatura desejada, atribuída pelo usuário. Além disso, é possível ajustar as constantes do controlador PID (ganho proporcional, integral e derivativo) para observar o comportamento do controlador em relação a cada componente ou tentar atribuir valores que resultem em outro comportamento desejado. Outro parâmetro que também pode ser especificado é a frequência e o período de acionamento do buzzer. Por fim, o sistema permite que o usuário desligue o aquecimento rapidamente e acionando o cooler na rotação máxima.

A atribuição e visualização dos valores dos parâmetros citados podem ser feitas pela interface local (usando botões, teclado matricial e display) e pelo protocolo de comunicação remota (via UART). É importante ressaltar que cada interface não abrange a configuração de todos os parâmetros, a seção 4 - Manual de Utilização demonstra a funcionalidade de cada interface, incluindo suas respectivas configurações.

O sistema também possui um alerta sonoro quando a atribuição de um valor é feita corretamente. Se um valor acima do limite esperado for inserido (também especificado na seção 4), será exibido um alerta visual.

Por fim, essa documentação também contém uma tabela de requisitos, diagrama de camadas, diagrama de classes, diagrama de blocos, fluxograma de algumas principais funções do sistema, máquina de estados da interface UART e local, os componentes principais e suas características e a metodologia utilizada para o cálculo do controlador.

2 Documentação do Sistema

2.1 Tabela de Requisitos

Para especificar e desenvolver um produto, é necessário inicialmente termos uma tabela de requisitos. Tais requisitos podem ser funcionais (que definem a funcionalidade do sistema) e não funcionais (restrições do projeto que afetam uma ou mais de suas funcionalidades). As figuras 1, 2 e 3 mostram os requisitos do sistema desenvolvido.

1. Temp System

1.1 Heat System

ReqID	Requirement Description	Rationale
HEAT001	O sistema de aquecimento deverá ser capaz de capturar a temperatura [REAL_TEMP] do aquecedor, em uma frequência de 10Hz	-
HEAT002	O valor [TEMP_TARGET] não deve assumir valores maiores que 90° C	Considerando que estamos lidando com um componente montado em uma PCB, altas temperaturas podem danificar sistemas próximos
HEAT003	O sistema de aquecimento não deverá demorar mais que 2min para atingir o valor [TEMP_TARGET]	-
HEAT004	O sistema de aquecimento não deverá ultrapassaro valor de [TEMP_TARGET] por mais de 2°C	-
HEAT005	O valor [REAL_TEMP] não deve assumir valores maiores que 90° C	Considerando que estamos lidando com um componente montado em uma PCB, altas temperaturas podem danificar sistemas próximos

1.2 Cooling System

ReqID	Requirement Description	Rationale
COOL001	O sistema de resfriamento deverá ser capaz de capturar a velocidade real de rotação [REAL_FAN_SPEED] do Cooler de resfriamento	-
COOL002	O valor [REAL_FAN_SPEED] não deverá assumir valores acima de 4500RPM	Limitação física do cooler

Figure 1: Requisitos do sistema de temperatura.

2. HMI System

ReqID	Requirement Description	Rationale
HMI001	O sistema deverá fornecer uma interface local para operação	-
HMI002	O sistema deverá ser capaz de mostrar [TEMP_TARGET] no Display LCD	-
HMI003	O sistema deverá ser capaz de mostrar [REAL_TEMP] no Display LCD	-
HMI004	O sistema deverá atualizar o valor [REAL_TEMP], quando em amostra no display LCD, a uma taxa de 2Hz	-
HMI005	O sistema deverá ser capaz de mostrar [REAL_FAN_SPEED] no Display LCD	-
HMI006	O sistema deverá atualizar o valor [REAL_FAN_SPEED], quando em amostra no display LCD, a uma taxa de 2Hz	-
HMI007	O sistema deverá ser capaz de mostrar [PID_KP] no Display LCD	-
HMI008	O sistema deverá ser capaz de mostrar [PID_KI] no Display LCD	-
HMI009	O sistema deverá ser capaz de mostrar [PID_KD] no Display LCD	-
HMI010	O sistema deverá ser capaz de ler [KEYBOARD_INPUT] de um teclado matricial	-
HMI011	O sistema deverá ser capaz de identificar o pressionamento de botões embarcados [UP_BTN, DOWN_BTN, LEFT_BTN, RIGHT_BTN, MIDDLE_BTN]	-
HMI012	O sistema deverá ser capaz de alterar o valor [TEMP_TARGET] localmente	-
HMI013	O sistema deverá ser capaz de alterar o valor [PID_KP] localmente	-
HMI014	O sistema deverá ser capaz de alterar o valor [PID_KI] localmente	-
HMI015	O sistema deverá ser capaz de alterar o valor [PID_KD] localmente	-
HMI016	O sistema deverá ser capaz de desabilitar e abilitar o sistema de aquecimento localmente	-
HMI017	O sistema deverá ser capaz de avisar de maneira sonora o usuário	-
HMI018	O sistema sonoro deve estar na faixa de frequência de 0 e 9999,99Hz	-
HMI019	O sistema sonoro deve ser capaz de ser acionado em um intervalo de tempo de 0 a 9999,99ms	-
HMI020	O sistema deve enviar uma mensagem sonora de confirmação quando um valor for recebido com sucesso	-

Figure 2: Requisitos do sistema de interface homem máquina.

3. Comunication System

ReqID	Requirement Description	Rationale
COM001	O sistema deve ser capaz de enviar o valor [REAL_TEMP] via UART	-
COM002	O sistema, para o envio do valor [REAL_TEMP] via UART, seguir o a formatação: Valor em formato de String, com três casas decimais, separadas por "\r\n"	-
COM003	O sistema deve ser capaz de enviar o valor [PID_KP] via UART	-
COM004	O sistema, para o envio do valor [PID_KP] via UART, seguir o a formatação: Valor em formato de String, com três casas decimais, separadas por "\r\n"	-
COM005	O sistema deve ser capaz de enviar o valor [PID_KI] via UART	-
COM006	O sistema, para o envio do valor [PID_KI] via UART, seguir o a formatação: Valor em formato de String, com três casas decimais, separadas por "\r\n"	-
COM007	O sistema deve ser capaz de enviar o valor [PID_KD] via UART	-
COM008	O sistema, para o envio do valor [PID_KD] via UART, seguir o a formatação: Valor em formato de String, com três casas decimais, separadas por "\r\n"	-
COM009	O sistema deve ser capaz de enviar o valor [REAL_FAN_SPEED] via UART	-
COM010	O sistema, para o envio do valor [REAL_FAN_SPEED] via UART, seguir o a formatação: Valor em formato de String, com três casas decimais, separadas por "\r\n"	-
COM011	O sistema deve ser capaz de enviar o valor [BUZZER_FREQUENCY] via UART	-
COM012	O sistema, para o envio do valor [BUZZER_FREQUENCY] via UART, seguir o a formatação: Valor em formato de String, com três casas decimais, separadas por "\r\n"	-
COM013	O sistema deve ser capaz de enviar o valor [BUZZER_PERIOD] via UART	-
COM014	O sistema, para o envio do valor [BUZZER_PERIOD] via UART, seguir o a formatação: Valor em formato de String, com três casas decimais, separadas por "\r\n"	-
COM015	O sistema deve ser capaz de receber o valor [TEMP_TARGET] via UART	-
COM016	O sistema deve ser capaz de receber o valor [PID_KP] via UART	-
COM017	O sistema deve ser capaz de receber o valor [PID_KI] via UART	-
COM018	O sistema deve ser capaz de receber o valor [PID_KD] via UART	-
COM019	O sistema deve ser capaz de receber o valor [BUZZER_FREQUENCY] via UART	-
COM020	O sistema deve ser capaz de receber o valor [BUZZER_PERIOD] via UART	-
COM021	O sistema deve ser capaz de receber o valor [BUZZER_STATUS] via UART	-
COM022	O sistema deve ser capaz de acionar o buzzer após receber o valor [BUZZER_STATUS] corretamente com a frequência [BUZZER_FREQUENCY] e por um período [BUZZER_PERIOD]	-
COM023	O sistema deve enviar uma mensagem sonora de confirmação quando um valor for recebido com sucesso	-

Figure 3: Requisitos do sistema de comunicação.

2.2 Componentes de Hardware

Os principais componentes do sistema de controle de temperatura desenvolvido e suas principais características estão demonstradas na figura 4.

STM32G474

Clock máximo	170MHz
Memória Programada	512KBytes
Memória RAM	96KBytes
Conversor A/D	12bits de resolução, 2.66MHz de clock
Timers	2x32bits, 2x16bits
Encapsulamento	64 pinos
Alimentação	1.71 V to 3.6 V
Custo	154,81 reais

Buzzer

Alimentação	12V
Corrente máxima	12mA
Temperatura	-20~+70C
Dimensões	Diametro: 13.8mm, Altura: 7.5mm
Sound Pressure Level	85 dB
Custo	2,23 reais

Fotodiodo - Tacômetro

Área sensível a radiação	0.17mm^2
Temperatura	-40 a 80C
Tensão coletor emissor máxima	32V
Corrente máxima do coletor	50mA
Potência total máxima	90mW
Dimensão	2.2x2x3.45mm
Custo	5,18 reais

Sensor de Temp Analógico

Alimentação	4 a 30V
Corrente Máxima	114 µA
Ganho	10 mV/C
Faixa de Leitura	-55 a 150C
Dimensões	2x1x1cm
Custo	14,90 reais

LEDs

Alimentação	1,8V
Potencia máxima	60 mW
Temperatura	-20 a +80C
Dimensões	Diâmetro: 5mm, Altura: 7.8mm
Custo	3 reais

Display LCD

Alimentação	5 V
Controlador	HD44780
Interface	4 ou 8 pinos de dados
Quantidade de caracteres	2 linhas, 16 colunas
Comunicação	I2C
Dimensões	80x36x12
Custo	24 reais

Ventilador

Alimentação	12V
Corrente	0.09A
Dimensões	40x40x10mm
Interface	3 pinos
Controle de velocidade	3000 rpm
Custo	32,21 reais

Teclado Matricial

Alimentação	5V
Corrente máxima	100mA
Interface	8 pinos
Quantidade teclas	16 teclas
Temperatura	0 a 50C
Dimensões	6.9x7,6cm
Custo	8,46 reais

Aquecedor

Resistência	68R
Potência	2 W
Tensão máxima	350 V
Dimensão	5x12mm
Custo	0,05 reais

Push Button

Alimentação	12 V
Corrente máxima	0,5 A
Dimensões	6x6x5mm
Custo	0,20 reais

Figure 4: Pricipais componentes e suas características.

2.3 Diagrama de Blocos

O diagrama de blocos é uma das ferramentas de representação de hardware de sistemas embarcados, em que o fluxo de informações entre os componentes são representados por setas. A figura 5 mostra este diagrama para o sistema de controle de temperatura desenvolvido.

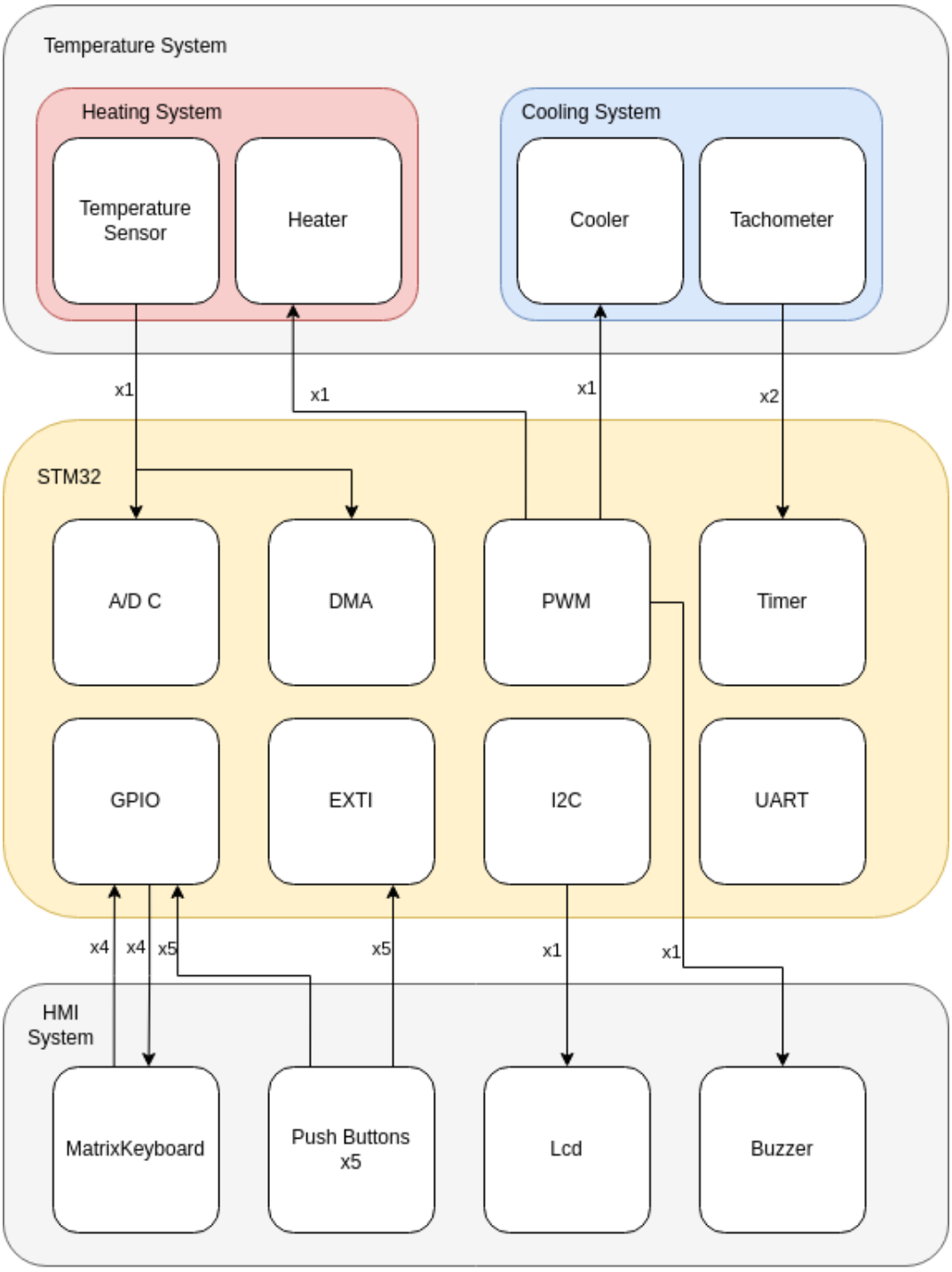

Figure 5: Diagrama de Blocos.

2.4 Diagrama de Camadas

O diagrama de camadas é utilizado para modelar e representar a arquitetura de software de um sistema em diversos níveis de abstração. A figura 6 mostra este diagrama para o sistema de controle de temperatura desenvolvido.

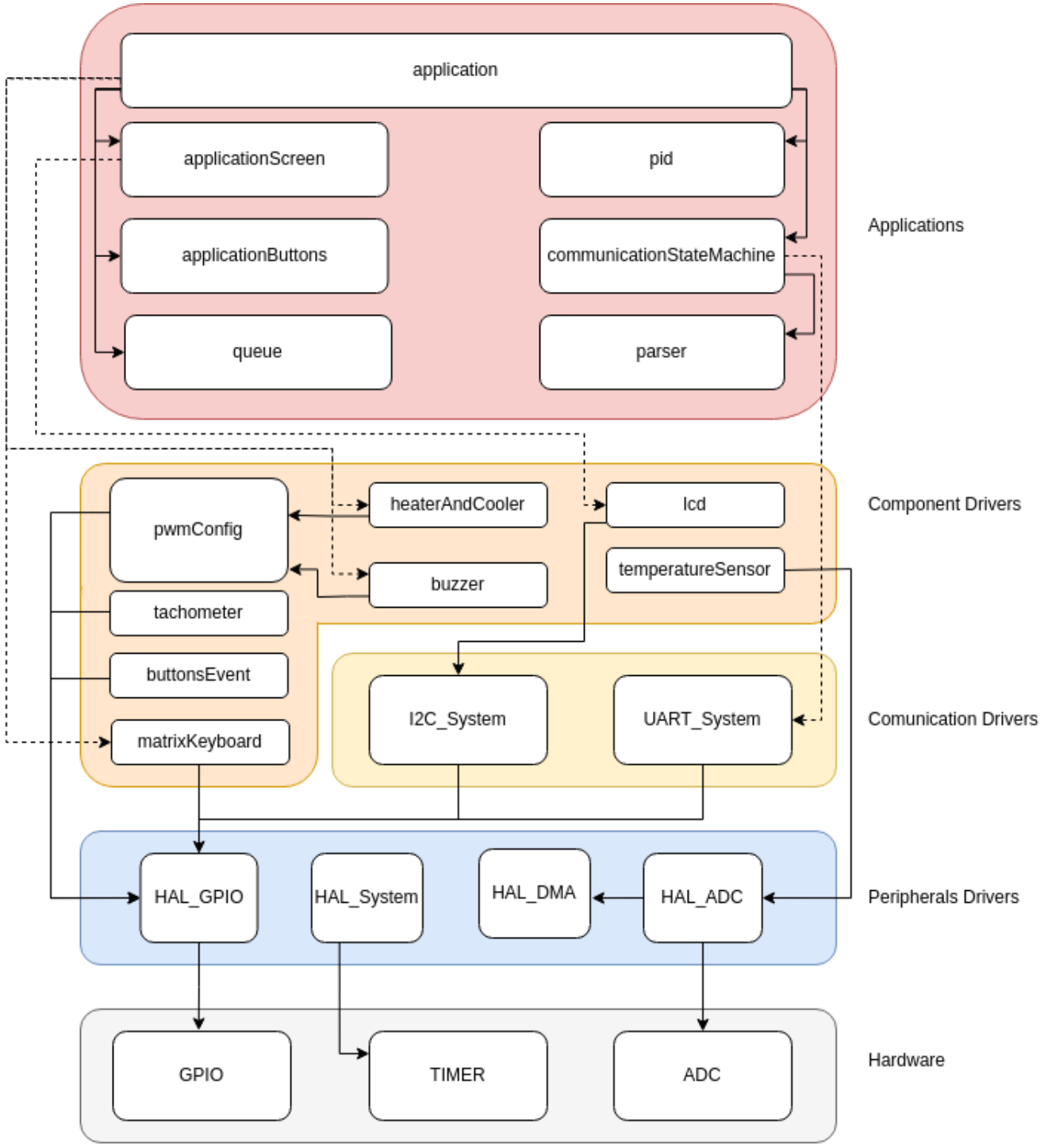

Figure 6: Diagrama de Blocos.

2.5 Diagrama de Classes

O diagrama de classes, assim como o de camadas, também é utilizado para a modelagem de software. Este diagrama representa a estrutura de um sistema, mostrando as classes, seus atributos, métodos e os relacionamentos entre elas. No nosso sistema, as classes são representadas por um arquivo fonte (.c e .h) em que seus atributos serão variáveis e structs e seus métodos serão as funções que manipulam o seu estado.

A figura 7 mostra este diagrama para parte do sistema desenvolvido.

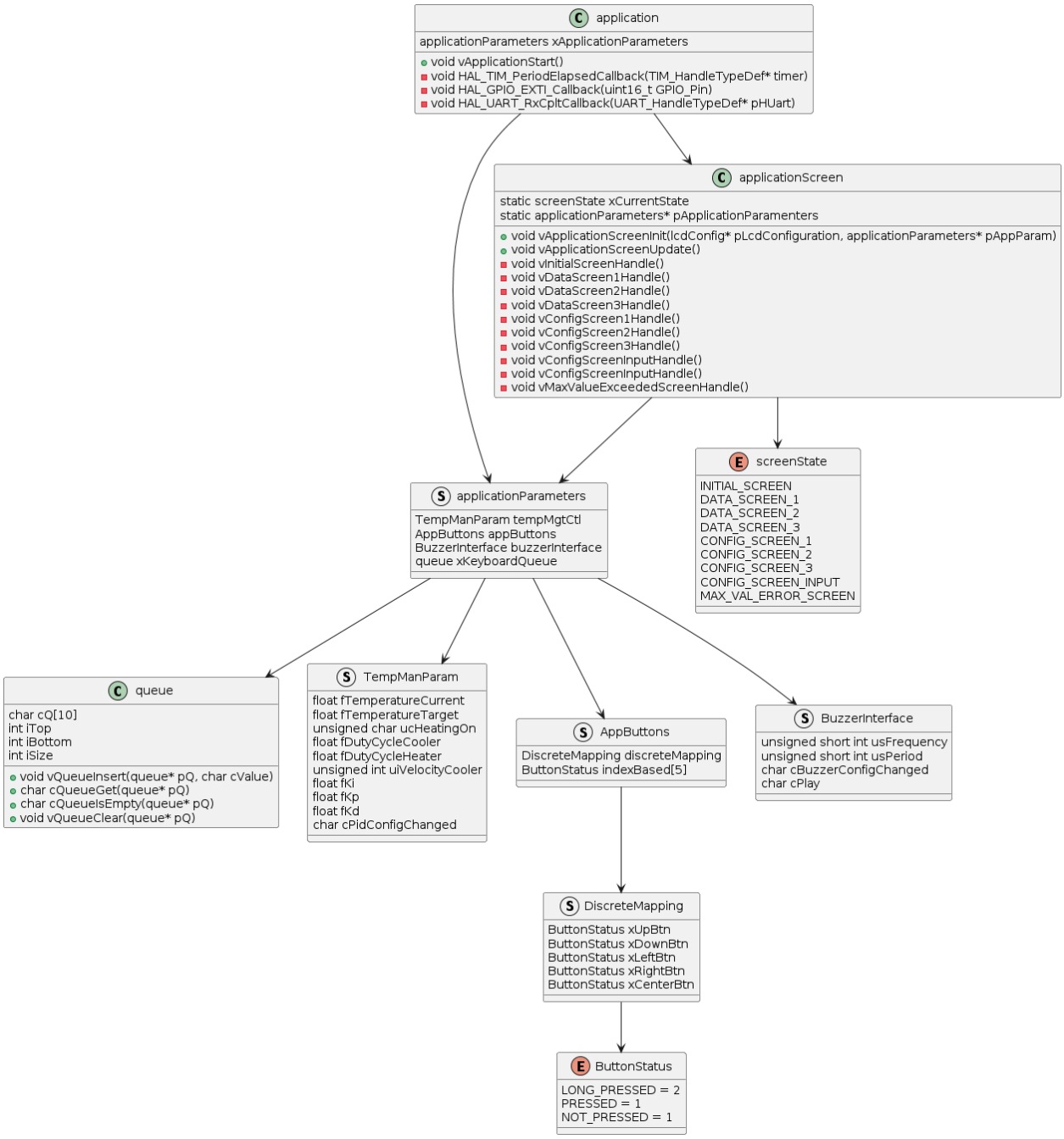

Figure 7: Diagrama de Classes Parcial

2.6 Fluxograma das Principais Funções

Os fluxogramas auxiliam no entendimento de como funcionam os algoritmos. As figuras 8 e 9 mostram dois desses fluxogramas do nosso sistema de controle de temperatura. O primeiro fluxograma descreve o arquivo application.c, em que implementa a aplicação do sistema de controle de temperatura e o segundo descreve o arquivo matrixKeyboard.c, em que implementa as funções do teclado matricial.

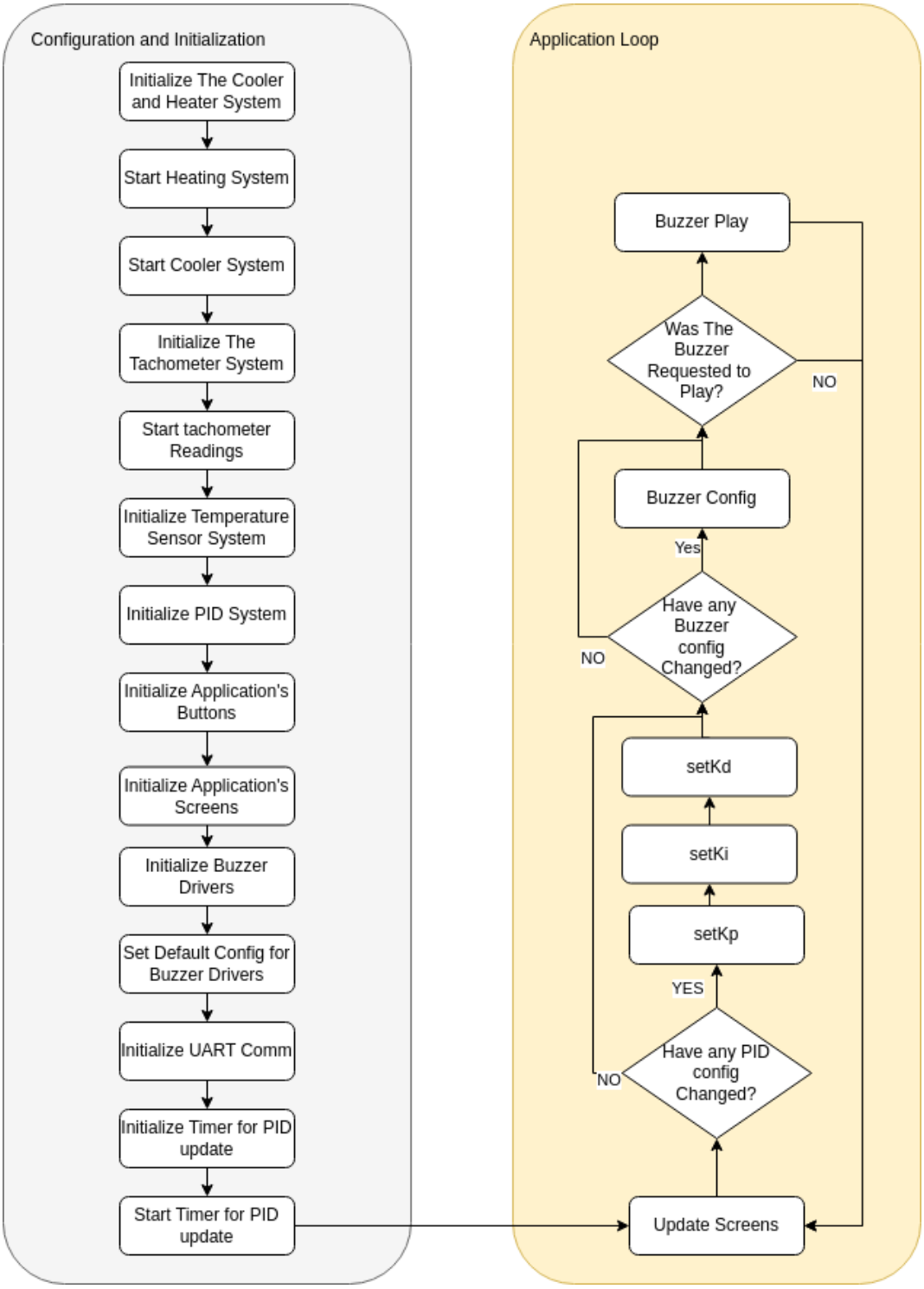

Figure 8: Fluxograma - Aplicação do sistema.

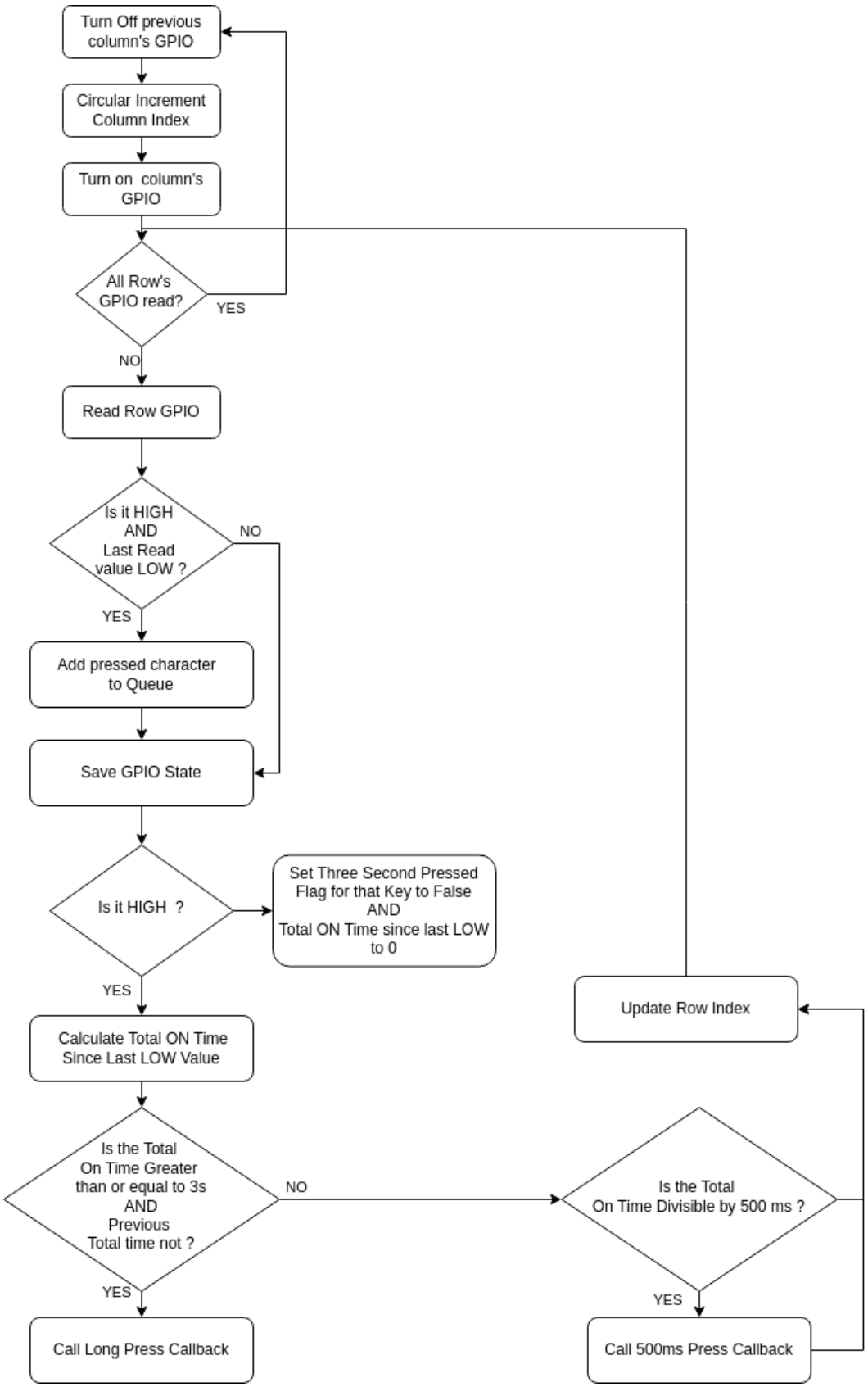

Figure 9: Fluxograma - Teclado matricial.

2.7 Comunicação entre o Sistema e o Computador

O protocolo de comunicação remota é feita via UART. Nele é possível visualizar e atribuir os valores em parâmtros definidos.

Sua máquina de estados é mostrada na figura 10, sendo que dentro dos estados “Param” e “Value”, se recebessem o caractere “;”, iriam retornar ou atribuir o valor pedido, respectivamente.

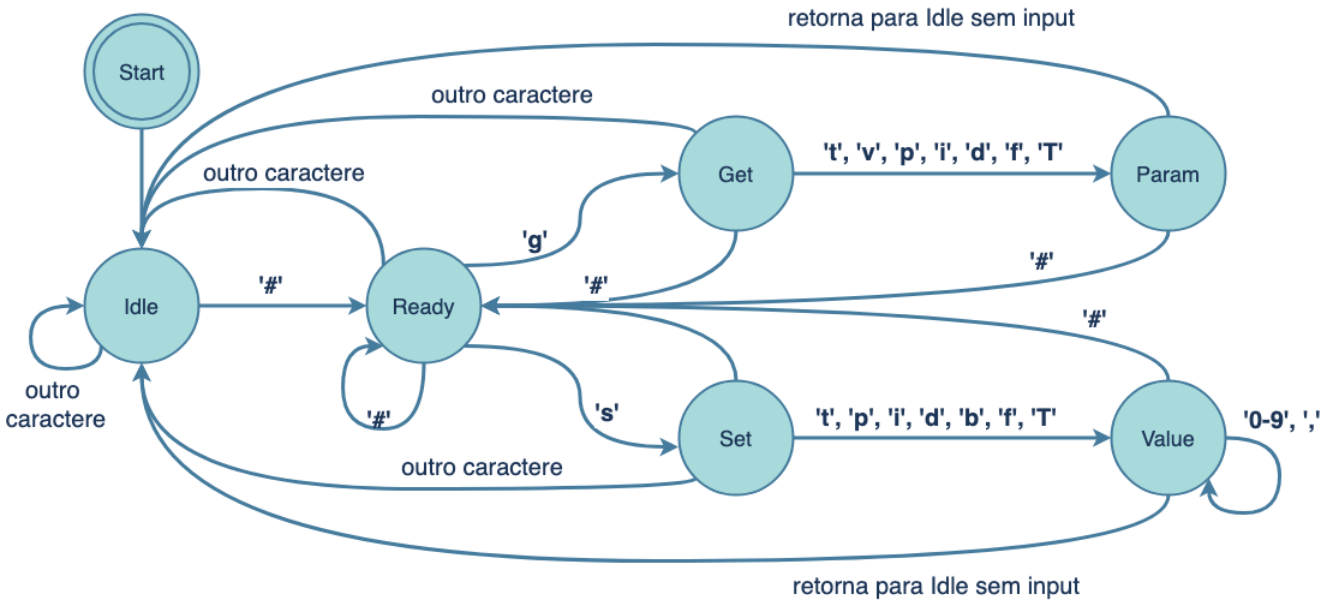

Figure 10: Pricipais componentes e suas características.

Os comandos seguem o seguinte padrão:

$$\#<Ordem><Parametro><Valor, N/A>;$$

Todo comando deve inicializar com ‘#’, seguido pela ordem get (‘g’) ou set (‘s’), então pelo parâmetro desejado: Para a ordem get:

- Temperatura atual (‘t’), tipo float
- Velocidade do cooler (‘v’), tipo unsigned int
- Ganho proporcional do controlador PID (‘p’), tipo float
- Ganho derivativo do controlador PID (‘d’), tipo float
- Ganho integrativo do controlador PID (‘i’), tipo float
- Frequência do buzzer (‘f’), tipo unsigned short int
- Período do buzzer (‘T’), tipo unsigned short int

Para a ordem set:

- Temperatura desejada (‘t’), tipo float
- Ganho proporcional do controlador PID (‘p’), tipo float
- Ganho derivativo do controlador PID (‘d’), tipo float
- Ganho integrativo do controlador PID (‘i’), tipo float
- Frequência do buzzer (‘f’), tipo unsigned short int

- Período do buzzer ('T'), tipo unsigned short int
- Status do buzzer ('b'), tipo char

Para os comandos set, é seguido pelo valor desejado para atribuir o parâmetro. O tamanho é de 0-7 caracteres e os valores reais devem ser separados por vírgula. Já para os comandos get, não deve ter nada nesta posição. Além disso, todos os comandos devem terminar com ';'.

Deve ser ressaltado que se o valor for corretamente atribuido aparecerá uma mensagem "[CONFIG SYS] Value set SUCCESSFUL" e o buzzer será ativado por um breve período, T, definido.

Inclusive, se o valor da temperatura desejada for maior que $90,00°C$, irá aparecer uma mensagem de erro: "[CONFIG SYS] ERROR: Max Value Exceeded" e não será atribuido nada ao parâmetro temperatura. A mesma coisa acontece quando os valores Kp, Kd, Ki forem maiores que 99,99 e o valor de status do buzzer for diferente de 1, para esses casos, aparecerá a mensagem "[CONFIG SYS] ERROR: Val. Outside Range".

Além disso, se o valor do status do buzzer for atribuido como 1, o buzzer apenas irá apitar por 500ms por padrão.

Por fim, é válido ressaltar os limites dos valores dos parâmetro para retornar e atribuir respectivamente:

- Temperatura atual: $0,000 \sim 90,000(°C)$
- Velocidade do cooler: $0 \sim 4500$ (RPM)
- Temperatura desejada: $0,000 \sim 90,000$ (°C)
- Ganho proporcional do controlador: $0,000 \sim 99,99$
- Ganho derivativo do controlador: $0,000 \sim 99,99$
- Ganho integrativo do controlador: $0,000 \sim 99,99$
- Frequência do buzzer: $0,000 \sim 9999,99$ (Hz)
- Período do buzzer: $0,000 \sim 9999,99$ (ms)
- Status do buzzer: 1

3 Procedimento Sintonisação do Controlador

O controlador utilizado foi PID, e para tal é necessário a sintonização (também chamado de "Tunning") dos ganhos do controlador, sendo eles o Kp (representa o ganho proporcional ao erro), Ki (representa o ganho proporcional à integral do erro) e Kd (representa o ganho proporcional à derivada do erro).

A fim de simular um workflow comumente utilizado em problemas de controle foi utilizado o processo de identificação de sistemas, onde aplicamos um input conhecido à nossa planta (no nosso caso uma tensão média de 3.3V de saída da porta[1] PWM) e medimos a resposta no tempo do nosso sistema (no nosso caso a temperatura medida pelo sensor analógico). Onde obtivemos a seguinte *resposta ao degrau*:

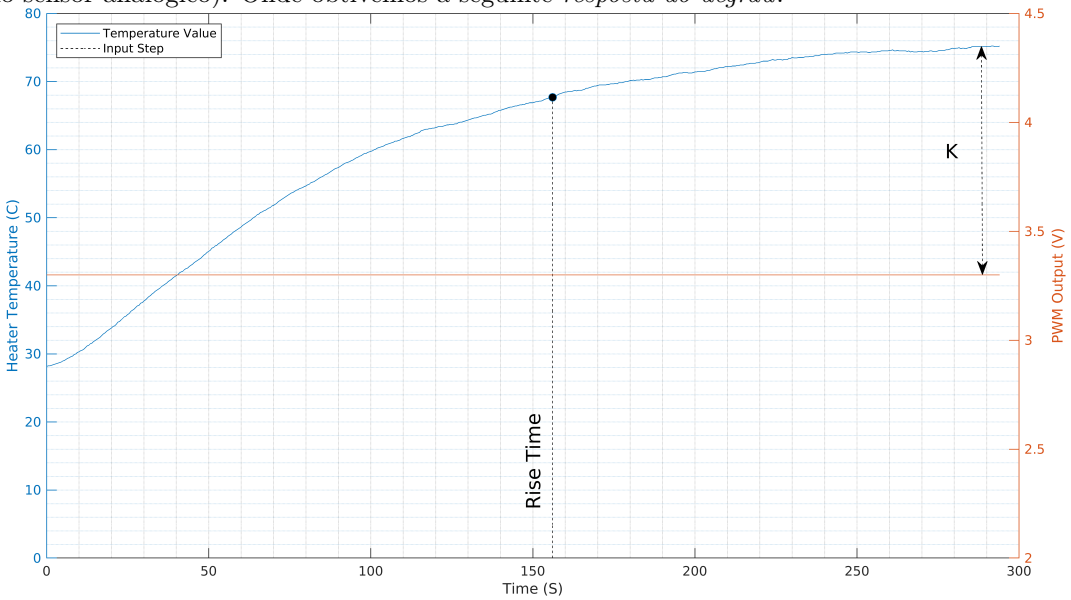

Figure 11: Resposta ao Grau do Sistema

Após aplicarmos as funções de identificação de sistemas do MatLab (que utiliza informações como o ganho K e o *Rise Time*, explicitados no gráfico), considerando o sistema como sendo de segunda ordem[2], obtivemos a seguinte função de transferência:

$$G(S) = \frac{23.97s + 0.3492}{s^2 + 1.428s + 0.01472} \tag{1}$$

Com isso, fomos capazes de fazer o procedimento de tunning dos ganhos do controlador baseado em modelo (model based control system design) utilizando as ferramentas do simulink, obtendo os seguintes ganhos que satisfazem os requisitos de overshoot e a seguinte resposta a um teste feito (figura 12).

$$K_p = 0.18$$
$$K_i = 2.34$$
$$K_d = 0.05$$

[1] Essa escolha foi arbitrária, mas em retrospecto o mais adequado a ter sido feito era a entrada ser de *Duty Cycle* = 1, o que teria evitado a necessidade de uma conversão no código, mas devido a natureza do controlador a escolha feita também funciona de forma adequada.

[2] Uma possível melhoria seria a aproximação da planta para um sistema de segunda ordem com *Dead Time*, mas devido à problemas técnicos e considerando que o resultado obtido já cumpria com todos os requisitos, tal melhoria será considerada somente para futuras versões do sistema.

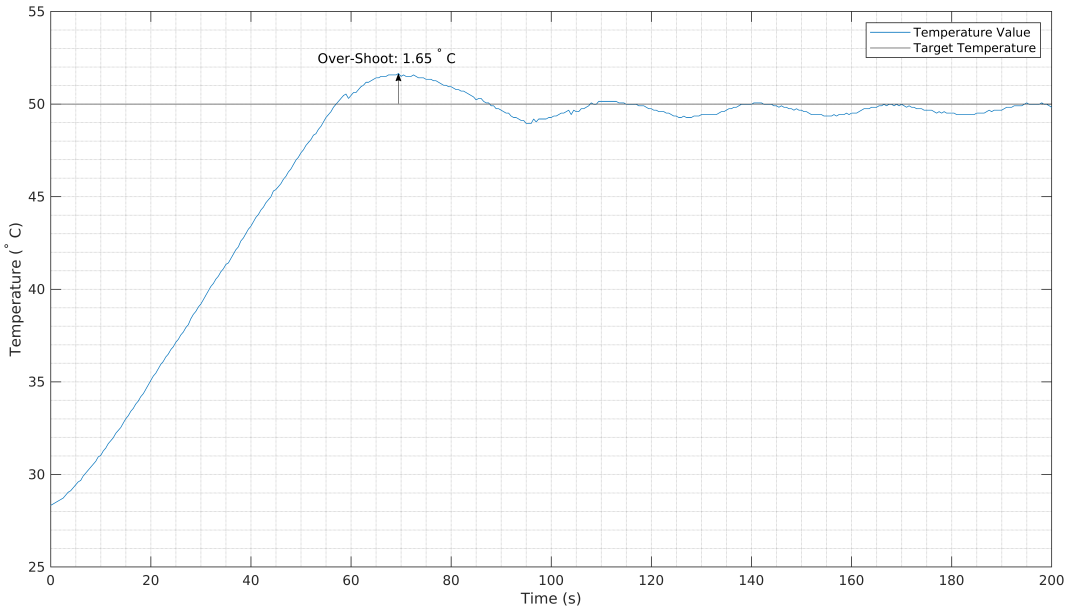

Figure 12: Resposta a Teste

Podemos observar que na resposta teste, o overshoot está abaixo do limite máximo especificado de $2°C$ acima da temperatura desejada. Além disso, podemos observar que ele chega a esse valor desejado de $50°C$ em menos de 1 minuto, sendo tal resposta rápida considerando que o sistema de aquecimento tem uma inércia maior.

Por fim, podemos observar que a resposta tende a se estabilizar em $50°C$, novamente sem ultrapassar os $\pm 2°C$ da temperatura desejada.

Como conclusão, temos que o controle atende os requisitos pedidos.

4 Manual de Utilização

O cotrolador de temperatura é composto principalmente por um aquecedor, um cooler, um sensor de temperatura, um tacômetro, uma tela LCD, um buzzer, botões e um teclado matricial. Há duas interfaces (local e remoto) em que ele pode ser configurado.

4.1 Interface Local

Essa interface é composta pelos botões UP e DOWN, teclado matricial, buzzer e tela LCD. Nele é possível:
Visualizar:

- Temperatura Atual ($^{\circ}C$)
- Temperatura Desejada ($^{\circ}C$)
- Velocidade do Cooler (RPM)
- Ganho proporcional do controlador PID (Kp)
- Ganho derivativo do controlador PID (Kd)
- Ganho integral do controlador PID (Ki)

Configurar:

- Temperatura Desejada ($0 < t < 90^{\circ}C$)
- Ganho proporcional do controlador PID ($00,00 < Kp < 99,99$)
- Ganho derivativo do controlador PID ($00,00 < Kd < 99,99$)
- Ganho integral do controlador PID ($00,00 < Ki < 99,99$)
- Status do controlador (Heater ON e Heater OFF), em que o Heater ON o sistema mantêm a temperatura próximo da desejada e tem o cooler desligado e o Heater OFF liga o cooler e desliga o aquecedor.

Por padrão, o sistema comça com os seguintes valores:

- Temperatura Desejada: $40^{\circ}C$
- Kp: 0.18
- Ki: 2.34
- Kp: 0.05
- Frequência do Buzzer: $1000Hz$
- Período de Ativação do Buzzer: $500ms$

A figura 13 mostra o diagrama da máquina de estados demonstrando todos os possíveis caminhos de visualização e configuração. Vale ressaltar que:

- Quando o valor é configurado com sucesso, o buzzer apita momentaneamente.
- BTN_CENTER = PRESS, BTN_UP = PRESS e BTN_DOWN = PRESS significa apertar o botão de centro, de cima e de baixo, respectivamente, e logo soltar
- BTN_CENTER = LONG_PRESS significa apertar o botão central por 3 segundos.
- MATRIX_KEY refere-se ao teclado matricial

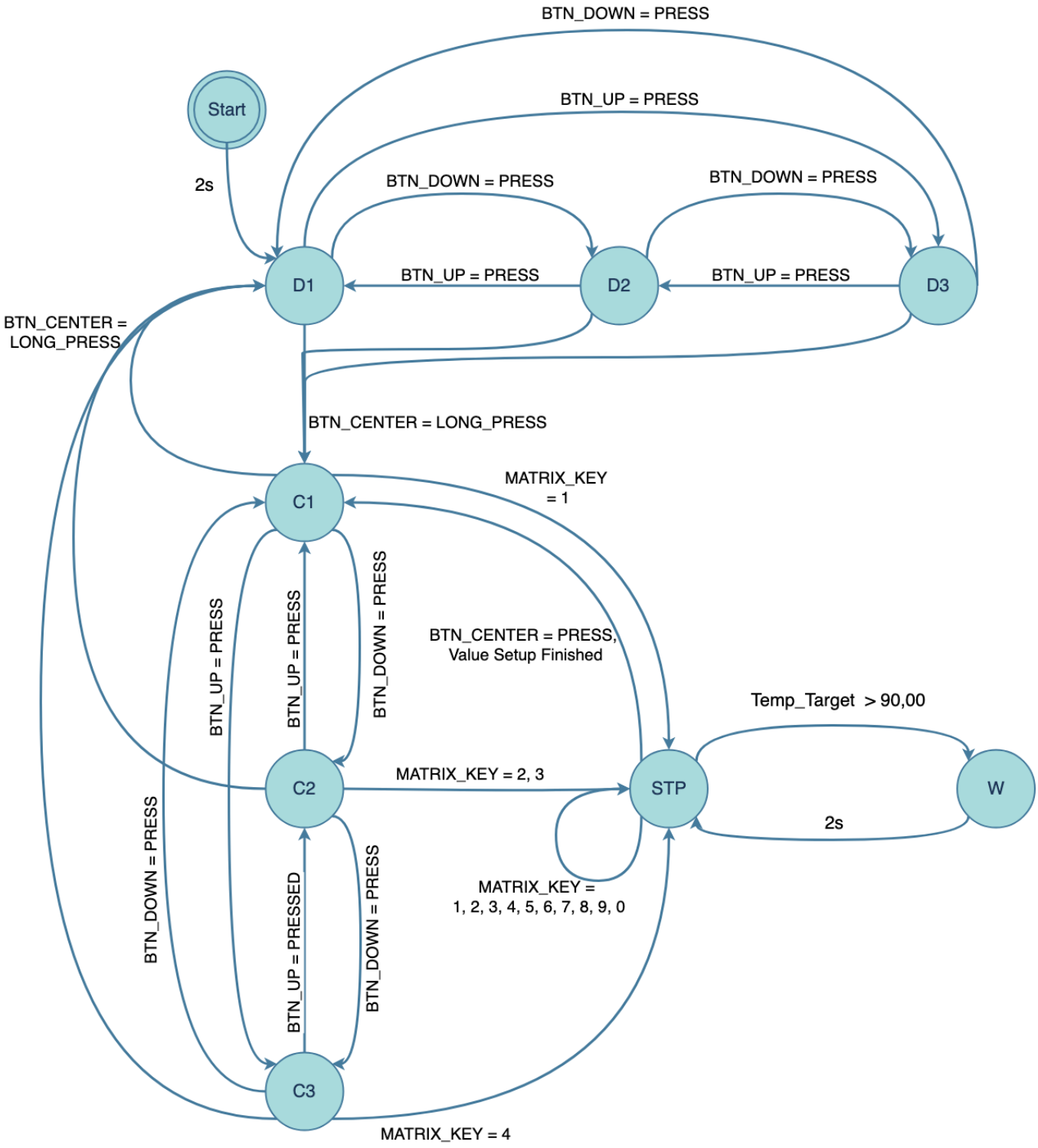

Figure 13: Máquina de Estados da Interface Local.

Estado	Descrição
Start (Tela inicial)	Tela LCD com texto "Controlador de Temperatura"
D1 (Tela de Display 1)	Tela LCD mostra os valores da temperatura atual (Temp. At:) e a temperatura desejada (Temp. Ds:)
D2 (Tela de Display 2)	Tela LCD mostra o valor da velocidade atual do cooler (Vel. C:)
D3 (Tela de Display 3)	Tela LCD mostra os valores de ganho do controlador PID proporcional (Kp:), derivativo (Kd:) e integrativo (Ki:)
C1 (Tela de Configuração 1)	Tela LCD mostra o título "**Configuracao**" e o parâmetro temperatura desejada (1-Temp. Desejada) para configurar
C2 (Tela de Configuração 2)	Tela LCD mostra os parâmetro Kp (2-Kp) e Ki (3-Ki) para configurar
C3 (Tela de Configuração 3)	Tela LCD mostra os parâmetro Kd (4-Kd) e Status do sistema (5-Heater ON/5-Heater OFF) para configurar
STP (Tela de Setup)	A depender de qual parâmetro foi selecionado nas telas de configuração, aparecerá o nome do parâmetro acima (exemplo: "Temp. Desejada") e então abaixo o espaço para adicionar os valores do teclado matricial ("__,__"). Conforme for adicionando os números, o mesmo irá aparecer na tela (exemplo: "1_,__")
W (Tela de Aviso)	Tela LCD mostra o aviso "Max Ultrapasssado Verifique Manual" por 2 segundos. Isso ocorrerá se no campo de setup da temperatura desejada for colocar um valor maior que $90,00°C$

Table 1: Descrição dos Estados da Interface Local.

4.2 Protocolo de Comunicação Remota

Essa interface é composta pela comunicação UART. Nele é possível visualizar e atribuir os valores em parâmtros definidos.

Os comandos seguem o seguinte padrão:

$$\#<Ordem><Parametro><Valor, N/A>;$$

Todo comando deve inicializar com '#', seguido pela ordem get ('g') ou set ('s'), então pelo parâmetro desejado:
Para a ordem get:

- Temperatura atual ('t')
- Velocidade do cooler ('v')
- Ganho proporcional do controlador PID ('p')
- Ganho derivativo do controlador PID ('d')
- Ganho integrativo do controlador PID ('i')
- Frequência do buzzer ('f')
- Período do buzzer ('T')

Para a ordem set:

- Temperatura desejada ('t')
- Ganho proporcional do controlador PID ('p')
- Ganho derivativo do controlador PID ('d')
- Ganho integrativo do controlador PID ('i')
- Frequência do buzzer ('f')
- Período do buzzer ('T')
- Status do buzzer ('b')

Para os comandos set, é seguido pelo valor desejado para atribuir o parâmetro. O tamanho é de 0-7 caracteres e os valores reais devem ser separados por vírgula. Já para os comandos get, não deve ter nada nesta posição. Além disso, todos os comandos devem terminar com ';'.

Deve ser ressaltado que se o valor for corretamente atribuido aparecerá uma mensagem "[CONFIG SYS] Value set SUCCESSFUL" e o buzzer será ativado por um breve período, T, definido.

Inclusive, se o valor da temperatura desejada for maior que $90,00°C$, irá aparecer uma mensagem de erro: "[CONFIG SYS] ERROR: Max Value Exceeded" e não será atribuido nada ao parâmetro temperatura. A mesma coisa acontece quando os valores Kp, Kd, Ki forem maiores que 99,99 e o valor de status do buzzer for diferente de 1, para esses casos, aparecerá a mensagem "[CONFIG SYS] ERROR: Val. Outside Range".

Além disso, se o valor do status do buzzer for atribuido como 1, o buzzer apenas irá apitar por 500ms por padrão.

Por fim, é válido ressaltar os limites dos valores dos parâmetro para retornar e atribuir:

- Temperatura atual: $0,000 \sim 90,000(°C)$
- Temperatura desejada: $0,000 \sim 90,000$ (°C)
- Ganho proporcional do controlador: $0,000 \sim 99,99$
- Ganho derivativo do controlador: $0,000 \sim 99,99$

- Ganho integrativo do controlador: $0,000 \sim 99,99$
- Frequência do buzzer: $0,000 \sim 9999,99$ (Hz)
- Período do buzzer: $0,000 \sim 9999,99$ (ms)
- Status do buzzer: 1

5 Problemas Identificados e Não Resolvidos

No desenvolvimento do projeto final alguns problemas foram identificados e não puderam ser resolvidos:

- **Clear Screen:** O comando CMD_CLEAR do código fonte ldc.h dado não funcionou. Para contornar esse problema foi criado um outro define CLEAR_SCREEN() no arquivo applicationScreen.c:

```
#define CLEAR_SCREEN() vLcdSetCursor(0,0);\
                       vLcdWriteString("                ");\
                       vLcdSetCursor(1,0);\
                       vLcdWriteString("                ")
```

  Nesse código, o clear screen funciona colocando o caractere ' ' em todas as posições.

- **Display LCD:** Algumas vezes ao compilar o código, a tela LCD inicia com caracteres estranhos, sendo necessário, ou recompilar novamente o código, ou tirar a fonte de alimentação do sistema e colocá-la de volta para, enfim, funcionar corretamente.

- **Resolução dos valores atribuidos:** Este não é um erro, mas vale ressaltar que pela representação digital de números racionais pelo sistema ser feita por ponto flutuante, temos que ao atribuir valores aos parâmetros, o dado salvado tem uma resolução de ± 0.01 em alguns casos onde o número resulta em uma dízima na sua representação de ponto flutuante.

6 Autoria dos Códigos Fornecidos

Neste projeto não houve partes copiadas ou baseadas em códigos desenvolvidos por terceiros e nem houve colaboração que este grupo tenha feito com outros grupos.