

**UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

**CONTROLE DE TEMPERATURA UTILIZANDO
SISTEMA OPERACIONAL DE TEMPO REAL**

**RELATÓRIO DA DISCIPLINA DE PROJETO DE SISTEMAS
EMBARCADOS**

Prof. Carlos Henrique Barriquello

**ARTHUR DAMASCENO
MATEUS CHEROBINI PICCININ**

Santa Maria, RS, Brasil

2021

INTRODUÇÃO

Este relatório tem como objetivo implementar um controle de temperatura com ação “Split Ranged” de uma planta teste, utilizando um sistema operacional de tempo real. Este é implementado em C embarcado através da placa de desenvolvimento STM32F103C8T6 e deve executar três tarefas na operação do sistema.

Para esta aplicação são utilizados conceitos da disciplina de Projeto de Sistemas Embarcados, além de conhecimentos sobre eletrônica e controle. O projeto em questão, contempla desde a montagem e ensaio da planta e obtenção da sua função de transferência até a implementação do controle por meio do sistema operacional de tempo real, parte esta que será apresentada com mais destaque neste relatório.

DESENVOLVIMENTO TEÓRICO

Neste capítulo são apresentados tópicos acerca do sistema a ser controlado, dos materiais utilizados para o desenvolvimento do sistema e conceitos de sistemas operacionais de tempo real.

1.1 Planta

O sistema a ser controlado neste projeto é constituído de um resistor de potência ligado a uma fonte de 12V, do qual deseja-se controlar a temperatura. Para isso serão utilizados dois atuadores, um MOSFET, responsável pelo chaveamento do elemento aquecedor, e uma ventoinha para resfriamento do componente.

1.2 Sensoriamento

A leitura da variável que se deseja controlar será realizada por meio da utilização de um termopar do Tipo K, este será utilizado em conjunto com um modulo MAX6675 que realiza a compensação de junta fria e também o condicionamento do sinal, enviando um sinal digital com protocolo baseado no padrão SPI para o microcontrolador.

1.3 Microcontrolador

O processamento dos dados do termopar e a implementação do sistema operacional se dá através da utilização do microcontrolador STM32F103C8T6 em sua versão placa de desenvolvimento, conhecida como “Bluepill”.

Os principais periféricos incluídos consistem em:

- 4 portas GPIO de 16bits
- 3 USART
- 2 controladores I2C
- 2 controladores SPI
- 2 x ADC
- 4 temporizadores

- 1 controlador CAN
- 1 gerador CRC
- CPU ARM CortexM3 com clock máximo de 72MHz.

1.4 Sistemas Operacionais de Tempo Real

São ditos Sistemas de Tempo Real os sistemas que além de executarem as tarefas de processamento e controle de informações, também possuem a característica de apresentar respostas em um tempo hábil, para que o sistema não entre em um estado inconsistente ou inválido.

Dessa forma, o sistema de tempo real é o software que gerencia os recursos de um sistema computacional, e tem o objetivo de assegurar com que todos os eventos sejam atendidos dentro de seu tempo estabelecido, e gerenciados da forma mais eficiente possível.

1.4.1 Tarefas

Tarefas são pequenos trechos de programa, estes apresentam responsabilidades específicas. Cada tarefa possui uma prioridade, que deve ser atribuída de acordo com sua importância. É a partir desta prioridade que busca-se garantir com que eventos com restrições de tempo possam ser executados de forma correta.

Como exemplos de tarefas, pode-se citar o tratamento de protocolos de comunicação, leitura de sensores, acionamento de leds indicativos e escrita em displays por exemplo.

1.4.2 Escalonador

O escalonador de tarefas é um serviço de gerenciamento. É responsável por definir quando uma tarefa será realizada, de forma a otimizar o uso do processador e garantir o cumprimento dos prazos de cada tarefa. Os processos de definição de execuções pode ser chamado de políticas de escalonamentos, estas definem critérios ou regras para a ordenação das tarefas de tempo real.

Os escalonadores utilizando dessas políticas produzem escalas que, se realizáveis, garantem o cumprimento das restrições temporais impostas as tarefas de tempo real. Uma escala é dita ótima se o arranjo das tarefas é o melhor possível no atendimentos dos prazos temporais.

1.4.3 Fila

Uma fila funciona como um espaço para armazenamento de dados no formato FIFO (First In First Out), esta permite a sincronização de tarefas e também é utilizada para comunicação entre tarefas onde é necessário o envio de valores.

1.4.4 Atraso

As funções de atraso são mecanismo de temporização do sistemas operacionais, assim como timers. Geralmente atrasos são utilizados para bloquear uma determinada rotina ou tarefa por um tempo determinado.

CAPÍTULO 2 DESENVOLVIMENTO PRÁTICO

Neste capítulo, discorreu-se sobre os procedimentos realizados no desenvolvimento do projeto, onde realizou-se a montagem do hardware e posterior levantamento da função de transferência da planta, seguido pelo projeto do controlador e sua implementação juntamente com o sistema operacional no microcontrolador utilizado.

2.1 Circuito de aquisição

Inicialmente, realizou-se a montagem da planta (vide figura 1) para obtenção da resposta em malha aberta do sistema. Neste ensaio, utilizou-se um sinal com duty cycle de 25% para o chaveamento do MOSFET, de forma a permitir o aquecimento do resistor livremente, a aferição dos dados de temperatura foi realizada utilizando um microcontrolador Arduino e os valores salvos em um arquivo de texto.

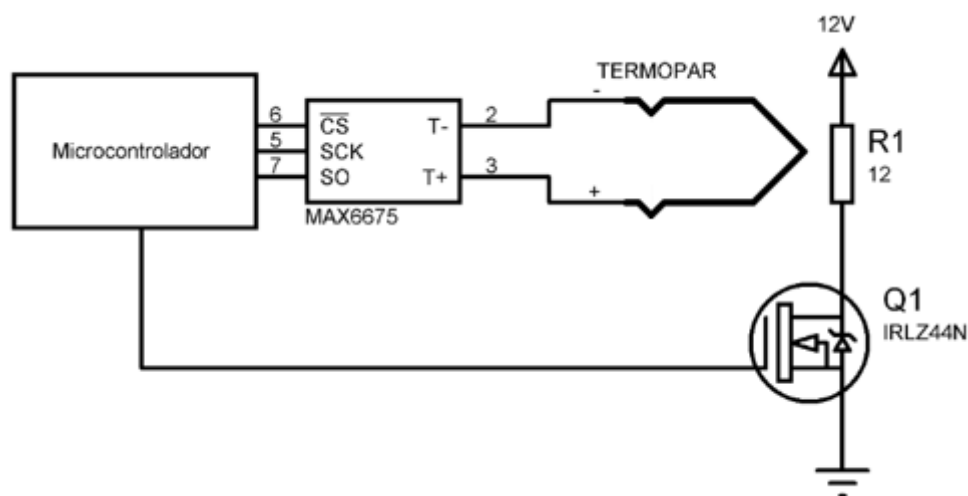


Figura 1 - Esquemático do sistema de aquisição

2.2 Função de Transferência

A partir do ensaio realizado obteve-se a resposta ao degrau do sistema em malha aberta, onde pode-se observar que o modelo adotado trata-se de uma função de primeira ordem (vide Figura 2). Dos dados obtidos foi possível obter a função de transferência do modelo, esta é descrita pela Equação 1.

$$G(s) = \frac{89}{450.5s + 1} \quad (1)$$

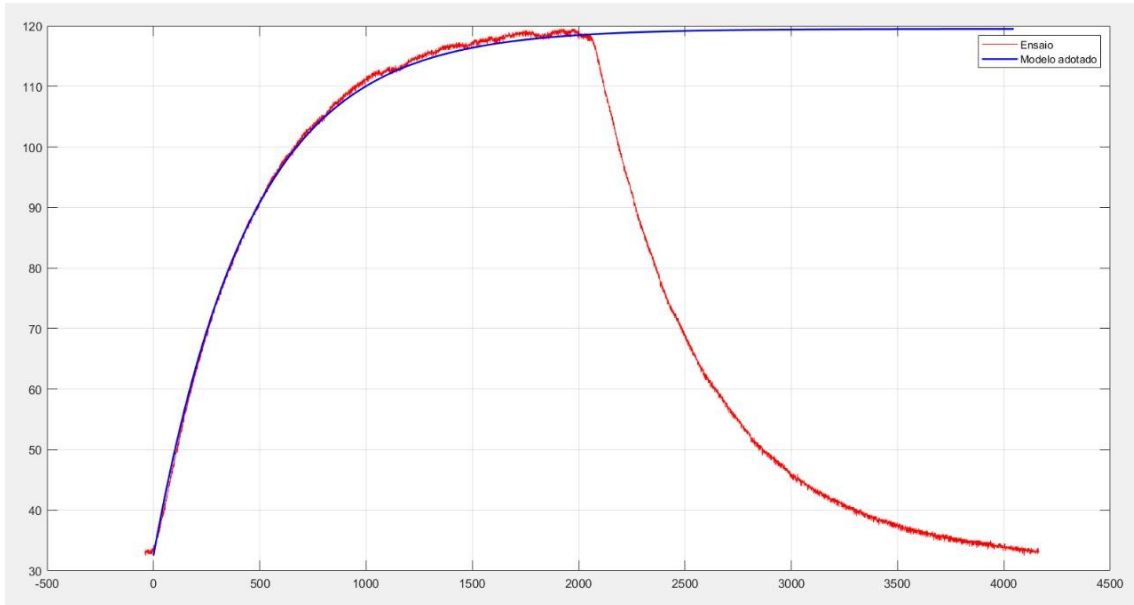


Figura 2 – Resultado do ensaio x modelo adotado

2.3 Projeto do Controlador

No projeto do controlador, optou-se pela utilização de um controlador PI para o aquecimento, ou proporcional-integral. O projeto foi realizado com auxílio do software Matlab, no qual é possível observar as respostas ao degrau em malha fechada e também a resposta em frequência do sistema compensado de maneira dinâmica, conforme altera-se o projeto do controlador.

Para o atuador responsável pelo arrefecimento, utilizou-se um controle ON-OFF que age sobre a ventoinha utilizada no sistema.

2.4 Implementação do Controlador Digital

Definido o controlador, realizou-se a digitalização do mesmo e a separação paralela em equações de diferenças para utilização no microcontrolador. Ainda adicionou-se um

filtro anti-windup para atenuar a ação do integrador quando ocorre a saturação do atuador, diminuindo assim as oscilações do sistema e evitando instabilidade.

2.5 Filtro FIR

Implementou-se um Filtro Digital FIR (Finite Impulse Response) na leitura dos valores de temperatura na frequência de 1,5Hz, o equivalente a 60% da frequência de Nyquist. Este tem como finalidade diminuir as ocorrências de falsas leituras na leitura da variável.

2.6 Implementação do Sistema Operacional

Inicialmente foi realizado a inserção das bibliotecas necessárias para a aplicação do sistema, seguido das declarações iniciais, como constantes e macros. Dentre estas constantes encontram-se os coeficientes do controlador implementado.

Na sequência foi realizado a inicialização dos periféricos e a criação das tarefas na pilha do escalonador. Este que possui um algoritmo de escalonamento baseado em prioridade, tendo a maior prioridade maior preferência de execução sempre.

Foram criadas quatro tarefas no sistema, sendo elas:

- Temp_Task
- Filter_Task
- Control_Task
- Display_Task

A tarefa *TempTask* é responsável por ler a temperatura do termopar, esta foi configurada com prioridade 3 e um atraso de 200ms, ou seja, a taxa de amostragem do sinal de temperatura é de 5Hz. Assim que a tarefa executa a leitura, o valor é enviado para a fila denominada *tempQueue*.

Assim que a fila *tempQueue* recebe a variável, esta envia o valor lido para a tarefa *Filter_Task*, definida com prioridade 2. Nesta tarefa, é realizado a filtragem dos valores recebidos, o valor filtrado é enviado a fila denominada *filteredTempQueue* após serem recebidos cinco valores.

O valor recebido pela fila *filteredTempQueue* é enviado para a tarefa *Control_Task*, que possui prioridade 4. Nesta tarefa são recebidos os valores de referência, lido a partir de um potenciômetro e o valor de temperatura lido pelo termopar. A partir destes é calculado a ação de controle. Além disso são acionados por esta tarefa os atuadores, onde, caso o erro seja positivo é acionado o MOSFET e caso seja negativo a ventoinha é acionada.

Por fim, a tarefa *Display_Task* é responsável por atualizar as variáveis em um mostrador gráfico através da comunicação UART com prioridade 1. Esta é implementada com um atraso de 1000ms, de forma a ocorrer a atualização do display com uma frequência de 1Hz.

O fluxograma da Figura 3 exemplifica de forma visual o processo descrito acima.

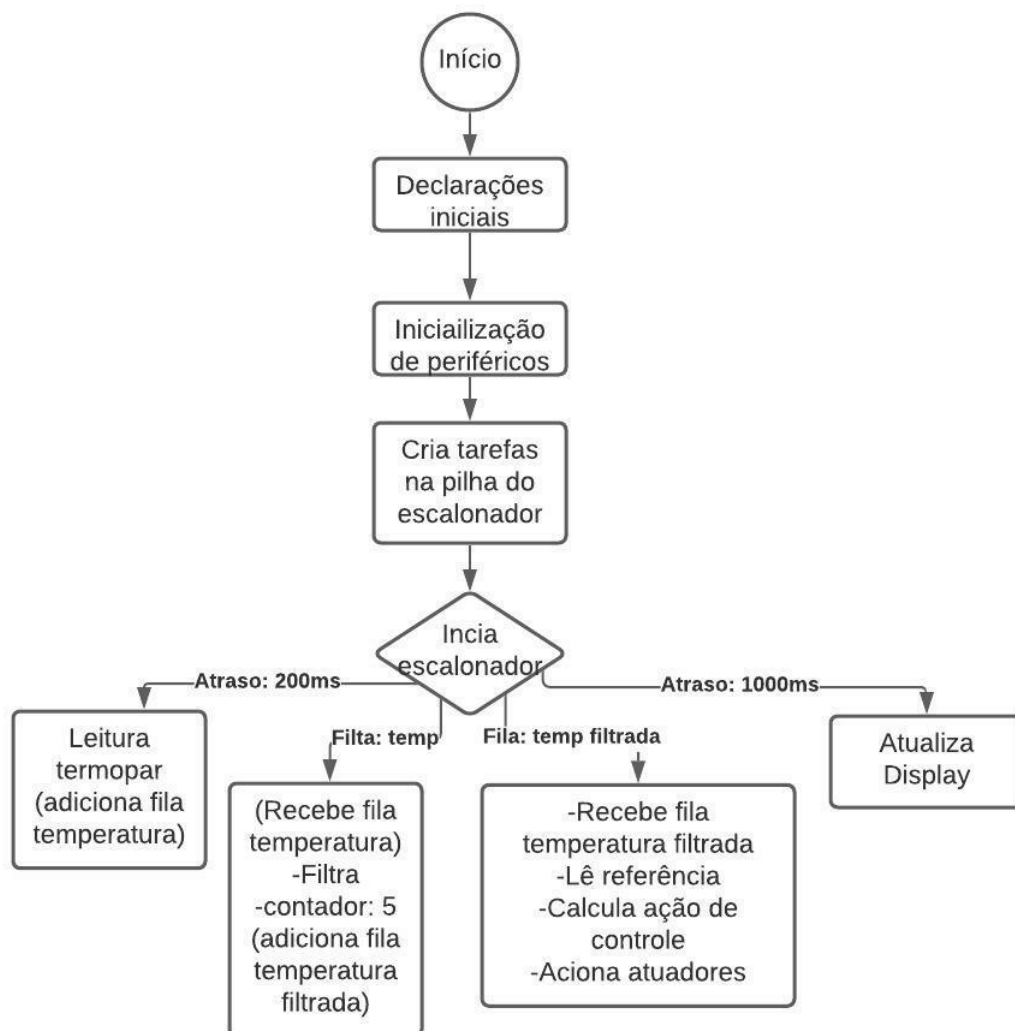


Figura 3 - Fluxograma do sistema implementado

2.7 Esquemático do sistema

A instalação física do sistema completo, contendo a planta, o microcontrolador, sensores, atuadores e mostradores gráficos segue o esquemático descrito pela Figura 4, onde são exibidos os componentes do sistema, bem como as ligações entre os mesmos.

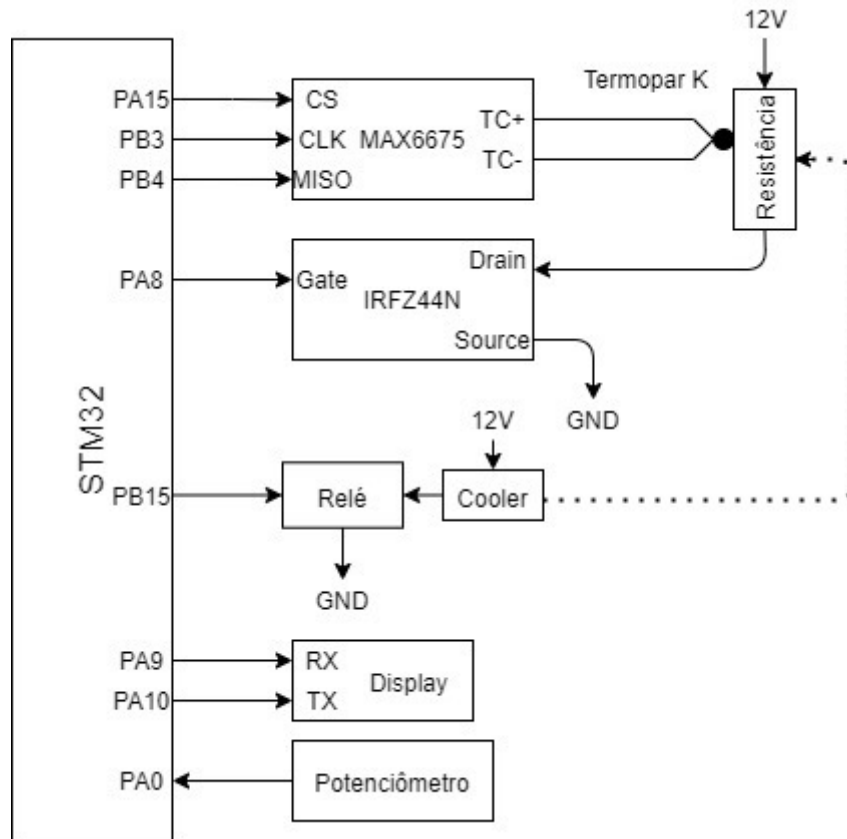


Figura 4 - Esquemático do sistema completo

CAPÍTULO 3 RESULTADOS

Neste capítulo são apresentados os resultados obtidos, tanto para o sistema de controle, quanto para o sistema operacional de tempo real.

3.1 Análise do controlador

O controlador projetado apresentou uma melhor resposta ao degrau em comparação ao sistema em malha aberta, através da Figura 5 é possível observar que o sistema em malha fechada apresenta um leve sobressinal e uma melhora no tempo de assentamento.

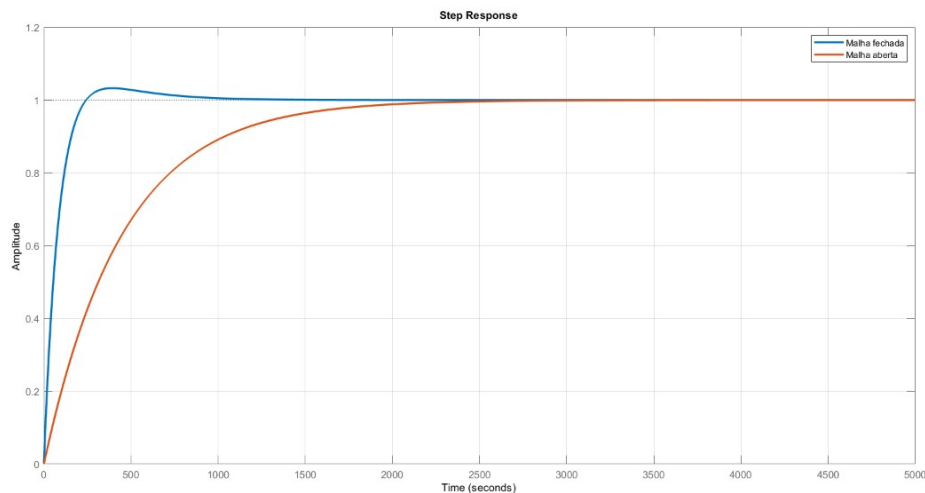


Figura 5 - Resposta ao degrau

Como citado anteriormente, o sistema de controle opera com dois atuadores, um atuador de aquecimento, e outro de resfriamento. O sistema de arrefecimento foi projetado para ser acionado quando o erro for maior em modulo do que o valor de 15°C. O comportamento do sistema em função do erro é descrito pela Figura 6.

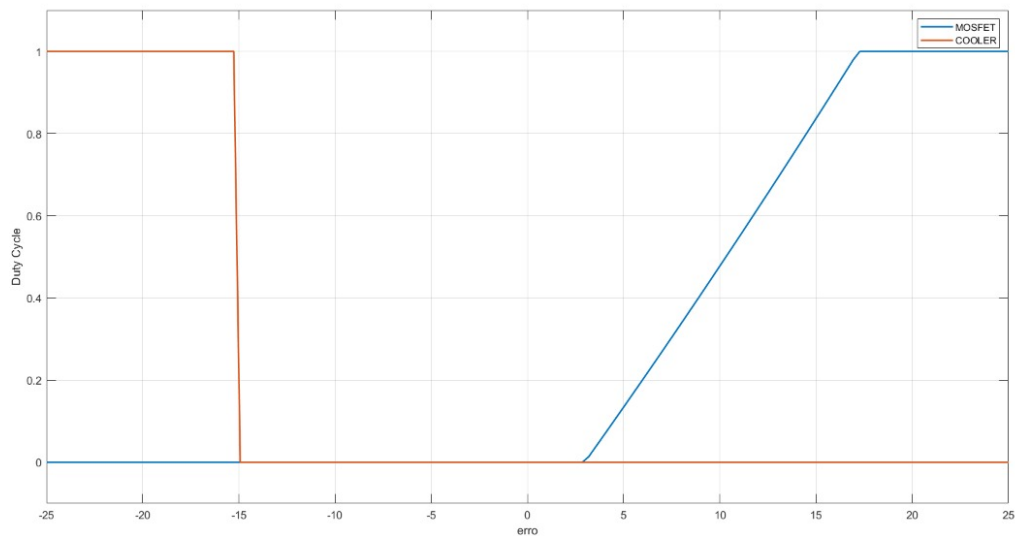


Figura 6 - Resposta do sistema em função do erro

3.2 Análise do sistema operacional

Utilizando o *software* SEGGER SystemView e sua biblioteca condizente para o ARM Cortex-M3 foi possível salvar indicadores de eventos do sistema operacional de tempo real na memória do microcontrolador, os quais foram exportados para análise no visualizador (vide Figura 7).

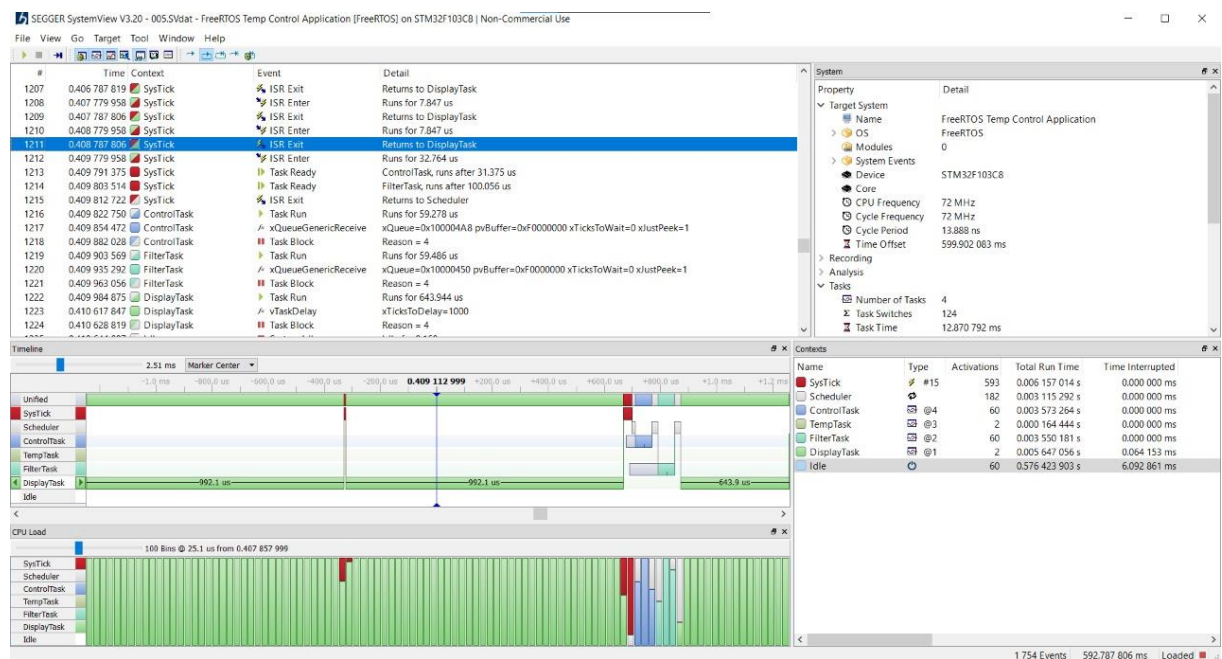


Figura 7 - Visão Geral do visualizador

O *software* permite a análise de um sistema operacional em diversos âmbitos de operação. Como ilustra a Figura 8, pode-se observar os contextos que compõem o sistema, sendo quatro tarefas criadas para o sistema, o *scheduler*, o *systick*, e uma tarefa denominada *Idle* que é acionada quando nenhuma outra tarefa está sendo executada.















Name	Type	Activations	Total Run Time	Time Interrupted
 SysTick	 #15	593	0.006 157 014 s	0.000 000 ms
 Scheduler		182	0.003 115 292 s	0.000 000 ms
 ControlTask	 @4	60	0.003 573 264 s	0.000 000 ms
 TempTask	 @3	2	0.000 164 444 s	0.000 000 ms
 FilterTask	 @2	60	0.003 550 181 s	0.000 000 ms
 DisplayTask	 @1	2	0.005 647 056 s	0.064 153 ms
 Idle		60	0.576 423 903 s	6.092 861 ms

Figura 8 - Contextos do sistema

A visualização dos contextos pode ser realizada em forma gráfica em linha de tempo, conforme Figura 9, e também em forma textual conforme Figura 10.

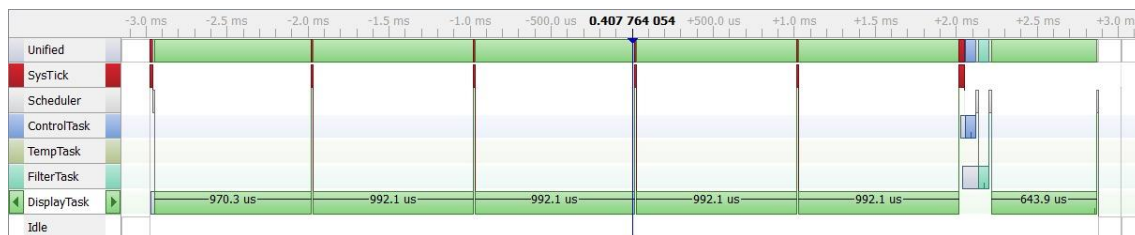


Figura 9- Linha de tempo















































#	Time	Context	Event	Detail
1202	0.404 799 583	 SysTick	 ISR Exit	Returns to Scheduler
1203	0.404 809 611	 DisplayTask	 Task Run	Runs for 5.003 ms
1204	0.405 779 958	 SysTick	 ISR Enter	Runs for 7.847 us
1205	0.405 787 806	 SysTick	 ISR Exit	Returns to DisplayTask
1206	0.406 779 972	 SysTick	 ISR Enter	Runs for 7.847 us
1207	0.406 787 819	 SysTick	 ISR Exit	Returns to DisplayTask
1208	0.407 779 958	 SysTick	 ISR Enter	Runs for 7.847 us
1209	0.407 787 806	 SysTick	 ISR Exit	Returns to DisplayTask
1210	0.408 779 958	 SysTick	 ISR Enter	Runs for 7.847 us
1211	0.408 787 806	 SysTick	 ISR Exit	Returns to DisplayTask
1212	0.409 779 958	 SysTick	 ISR Enter	Runs for 32.764 us
1213	0.409 791 375	 SysTick	 Task Ready	ControlTask, runs after 31.375 us
1214	0.409 803 514	 SysTick	 Task Ready	FilterTask, runs after 100.056 us
1215	0.409 812 722	 SysTick	 ISR Exit	Returns to Scheduler
1216	0.409 822 750	 ControlTask	 Task Run	Runs for 59.278 us
1217	0.409 854 472	 ControlTask	 xQueueGenericReceive	xQueue=0x100004A8 pvBuffer=0xF0000000 xTicksToWait=0 xJustPeek=1
1218	0.409 882 028	 ControlTask	 Task Block	Reason = 4
1219	0.409 903 569	 FilterTask	 Task Run	Runs for 59.486 us
1220	0.409 935 292	 FilterTask	 xQueueGenericReceive	xQueue=0x10000450 pvBuffer=0xF0000000 xTicksToWait=0 xJustPeek=1
1221	0.409 963 056	 FilterTask	 Task Block	Reason = 4
1222	0.409 984 875	 DisplayTask	 Task Run	Runs for 643.944 us
1223	0.410 617 847	 DisplayTask	 vTaskDelay	xTicksToDelay=1000
1224	0.410 628 819	 DisplayTask	 Task Block	Reason = 4
1225	0.410 644 097	Idle	System Idle	Idle for 9.168 ms

Figura 10 – Visualização textual

Na Figura 9 pode-se observar que ocorre preempção da tarefa `Display_Task`, esta é interrompida para execução das tarefas `Control_Task` e `Filter_Task`, observa-se que a tarefa `Display_Task` é a tarefa mais pesada da aplicação, por este motivo possui a menor prioridade, de forma a não atrapalhar a execução. Ainda, na Figura 10 observa-se de forma mais detalhada o funcionamento do sistema facilitando a análise de maneira mais precisa o tempo de duração de cada tarefa

Uma análise gráfica com intervalo de tempo menor é ilustrada na Figura 11, nesta observa-se outro ciclo de execução de tarefas. A primeira tarefa executada pelo sistema é a `Control_Task` que apresenta prioridade 4, esta tenta ler o valor contido na fila *filteredTempQueue* mas como essa não possui nenhum valor a tarefa logo libera o recurso do sistema. Na sequência é executada a tarefa `Temp_Task` de prioridade 3, e em seguida a tarefa `Filter_Task` de prioridade 2. Além disso, após a execução da `Filter_Task` neste corte da linha de tempo ocorre a aquisição do quinto valor de temperatura, o que faz com que seja inserido o valor filtrado na fila *filteredTempQueue* e a execução da tarefa `Control_Task` é acionada novamente devido a disponibilidade de valores atualizados na fila.

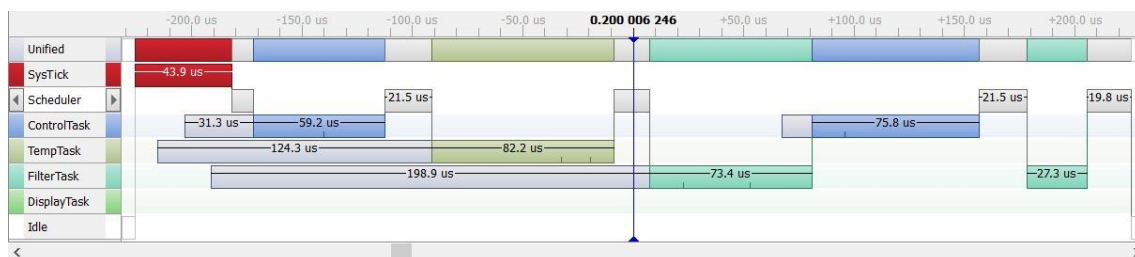


Figura 11 - Análise em intervalo menor

A partir da análise apresentada, pode-se concluir que o escalonamento do sistema implementado é dito realizável e cumpre os requisitos funcionais propostos.

3.3 Implementação pratica do sistema

Realizou-se pôr fim a montagem do sistema, onde pode-se observar o comportamento do sistema de maneira prática. Foi utilizada uma placa de prototipagem (protoboard) para montagem da maior parte do circuito, onde estão incluso o microcontrolador, o CI MAX 6675 e o potenciômetro. Por outro lado, a resistência a ser aquecida e o MOSFET foram inseridos em uma placa de circuito impresso separada para

evitar danos a placa de prototipagem. Além disso, utilizou-se um display TFT para exibição dos valores de referência, valor atual e porcentagem da razão cíclica.

O sistema descrito acima é apresentado pela Figura 12 onde:

1. Microcontrolador
2. Display
3. Placa com resistor, MOSFET e Termopar atrelado
4. Ventoinha
5. Relé para acionamento da ventoinha
6. Potenciômetro para ajuste da referência.

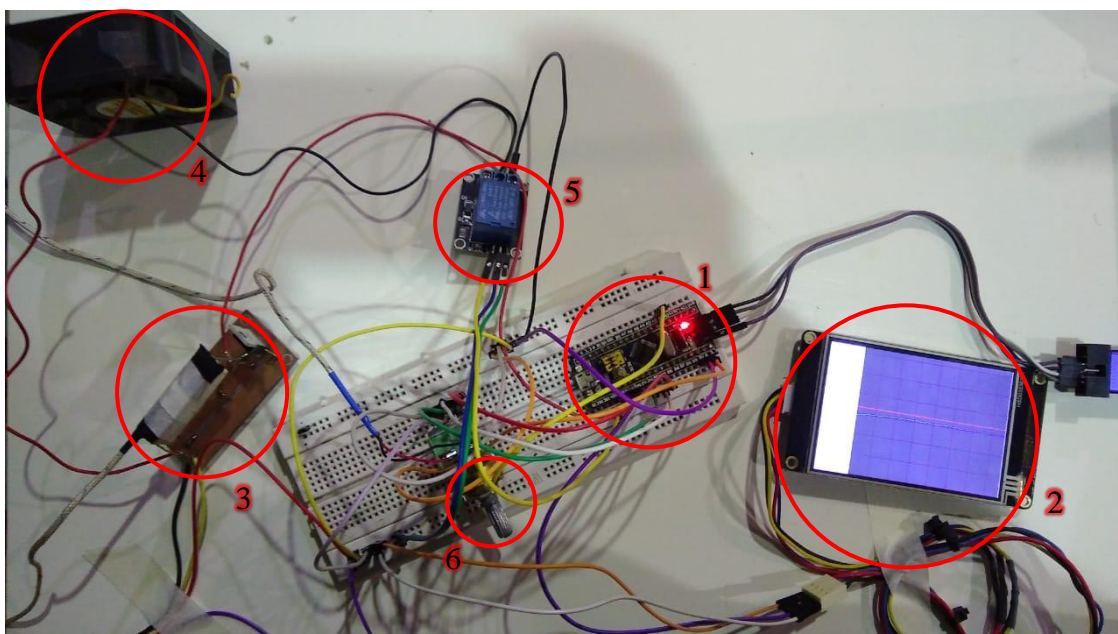


Figura 12 - Montagem do sistema

A Figura 13 ainda ilustra o *layout* de exibição das variáveis na tela do display.

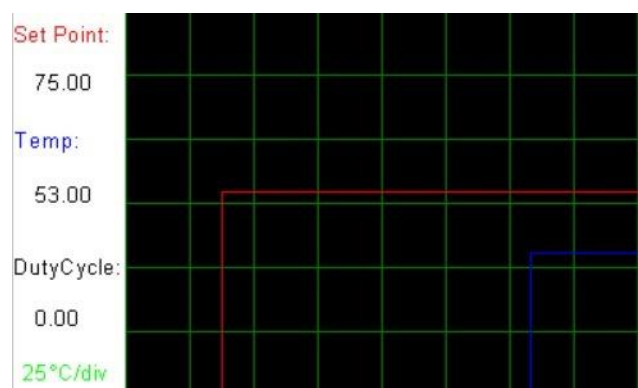


Figura 13 - Tela do display

CONCLUSÃO

Este trabalho contempla a implementação de um controlador digital de temperatura utilizando a interface de sistema operacional em tempo real, permitindo o determinismo temporal de execução de suas tarefas. A utilização de sistemas operacionais torna prática a execução e concorrência de tarefas paralelas além da portabilidade de código e praticidade na programação em equipe.

O sistema descrito apresentou resultados condizentes com o objetivo proposto, onde observou-se através do *software* SEGGER SystemView a execução das tarefas em sincronia com suas prioridades e períodos definidos previamente, reafirmando que o escalonamento empregado se mostra realizável.

Para acesso ao repositório de códigos do sistema embarcado, segue o link: <https://github.com/arthurdamasceno/Controle-de-temperatura>