Controle e supervisão de um sistema de caldeira simulado

Daniel Alves Farias, Lucas Natanael Leite, Maxela Martins Pontes, Yara Silva Braga

***Resumo*—Este trabalho consiste na aplicação ao dos conhecimentos obtidos na disciplina de Software em Tempo Real,principalmente na observação dos dados adquiridos através dos algoritmos implementados para a solução dos problemas sugeridos no "Controle e supervisão de um sistema de caldeira simulado".Foi disponibilizado pelo professor da disciplina um simulador do sistema de caldeira desenvolvido em Java com interface gráfica, no qual são setados os valores estabelecidos como: temperatura, energia(J/s), fluxo, entre outros parâmetros. Em seguida alguns problemas são implementados em linguagem C, como: tarefas periódicas, armazenamento de valores lidos nos sensores em buffers,verificação e solução de outras condições, etc.Com essas informações geradas foi possível obter dados suficientes para realizar testes comportamentais e visão estatística das medições de tempo real.**

***Index Terms*—Software em tempo real, mutex, monitores, semaforos, tempo de resposta, escalonamento, tarefa periodica,HWM, Hight Water Mark, Tempo de Resposta Maximo, aquecedor eletrico de água,thread.**

I. Introdução

Um sistema de Tempo Real pode ser entendido como um software que gerencia os recursos de um sistema computacional, com o objetivo de garantir com que todos os eventos sejam atendidos dentro de suas restrições de tempo e gerenciados da forma mais eficiente possível.

Uma vez que nesse tipo de sistema se faz utilização de threads, se faz uso de mutexes, que são mecanismos utilizados para implementar exclusão mútua nelas, ou seja, serve para garantir que regiões críticas de código não sejam executadas simultaneamente. Quanto ao uso de monitores, são um conjunto de procedimentos, variáveis e estruturas de dados, todas agrupadas em um módulo especial, onde somente um processo pode estar ativo dentro do monitor em um instante.

Portanto, o trabalho apresentado tem como objetivo a implementação de um sistema de tempo real que faça o controle de uma caldeira, composto por sensores e atuadores, estes últimos responsáveis por ajustar a temperatura da caldeira com valores fornecidos pelo usuário. [1].

II. Metodologia

A princípio, foi disponibilizado um simulador em Java do sistema para a interação com o software em tempo real desenvolvido pela equipe. Foi utilizada a linguagem C para implementar o algoritmo, onde foram feitos os devidos refinamentos nas threads de controle de temperatura, de nível, dentre outras, respeitando os seguintes Requisítos do Sistema definidos pelo professor. São eles:

1. Criar uma tarefa periódica (thread) para o controle de temperatura. O periodo é de 50ms. Use uma implementação com precisão usando as funções clock gettime e clock nanosleep. Ver arquivo exemplo "tarefaperiodica1.c". Esta tarefa deve ter prioridade máxima diante do escalonador do sistema;

2. Criar uma tarefa periódica (thread) para o controle de nıvel da água. O periódo é de 70 ms. Use uma implementação com precisão usando as funções clock gettime e clock nanosleep. Ver arquivo exemplo "tarefaperiodica1.c". Esta tarefa deve ter prioridade máxima diante do escalonador do sistema;

3. Usar os atuadores Ni, Q, Na, Nf nas tarefas de controle;

4. Criar uma tarefa para mostrar informações corrente na tela (Terminal) sobre os sensores Ta, T, Ti, No e H;

5. Criar tarefa de alarme caso o sistema esteja acima de 30 graus.

6. Criar tarefa para alterar através do teclado os valores de referência do nıvel e temperatura;

7. Criar tarefa para armazenar em arquivo os tempos de respostas da tarefa periódica do item 1 em arquivo, através de um buffer duplo produtor/consumidor).

8. Criar tarefa para armazenar a cada segundo os valores dos sensores de nıvel (H) e temperatura (T).

III. Resultados e discussões

Uma vez concluída a implementação, foram feitos os testes e a abtenção dos dados para análise. Foram utilizados uma temperatura de referência de 26 ºC e um nível de 1.5 m, já no simulador foram alterados os valores de Ta para 23 ºC e Ti para 20 ºC, em seguida, quando estava sendo feita a simulação, foram feitas variações em No a cada minuto a partir do início de execução com os respectivos valores: 10 kg/s, 20 kg/s, 35 kg/s, 40 kg/s e 55 kg/s. A cada 10 ms foram capturados os tempos de respostas das tarefas, obtendo ao todo 10 mil amostras, gravadas no arquivo "dados.txt", assim como a obtenção da temperatura e do nível a cada segundo, ambas gravadas nos arquivos "temperatura.txt"e "nivel.txt".

O teste foi efetuado por meio de um código utilizando Mint 20.04.3 LTS com o kernel de tempo real Linux, versão 5.4.74-rt41. O processador da máquina utilizada é um AMD athlon 3000g de 8GB de RAM.

Por fim, foram realizadas a medições, respeitando os seguintes parâmetros pré-definidos pelo professor:

1. Obter o tempo mínimo e médio de resposta da tarefa periódica de controle de temperatura;

Resposta: O tempo mínimo da tarefa periódica de controle de temperatura obtido foi de 0.1890 ms com uma média de 0.5702 ms.

2. Obter o tempo de resposta máximo observado. Apresentar os fatores que podem ter contribuído para caso o deadline da tarefa de controle de temperatura não seja suficiente. Plotar um gráfico, mostrando as medidas realizadas, na ordem dos casos de teste.

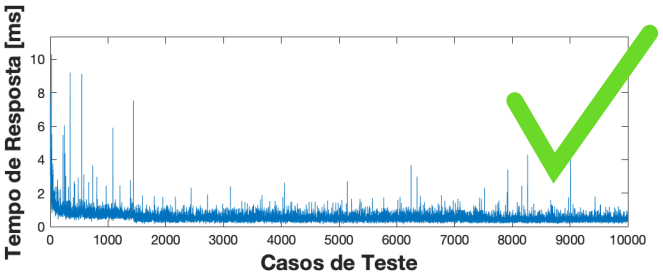

Figura 1: Gráficos de medições realizadas, na ordem dos casos de teste.

Resposta: como observado no gráfico, todas as amostras cumpriram o deadline definido, ficando muito abaixo de 50 ms e obtendo no pior caso um tempo de 10.2840 ms.

3. Plotar um gráfico histograma e verificar a porcentagem de amostras que cumpriram o deadline definido.

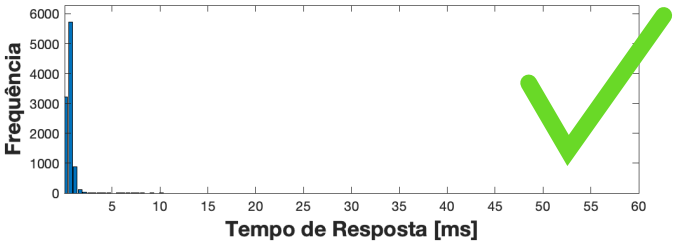

Figura 2: Histograma das medições realizadas.

Resposta: como pode ser observado na Figura 2, todas as amostras cumpriram o deadline definido, ficando abaixo de 1 ms na grande maioria dos casos. Ou seja, pode-se afirmar que 100% das amostras obtiveram êxito.

4. Observar se a tarefa de controle de temperatura tem Fator Skip 20. Plotar um gráfico mostrando apenas as medições onde o deadline foi perdido.

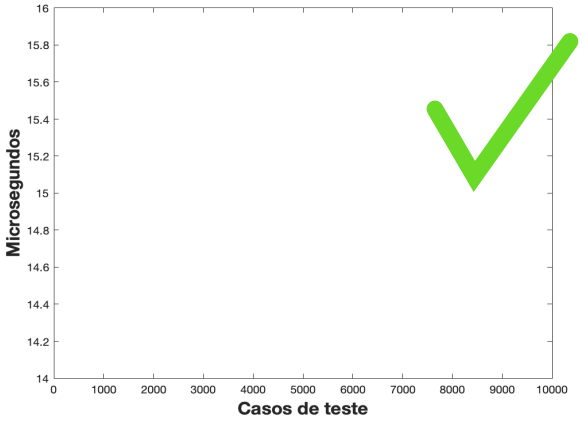

Figura 3: Gráfico mostrando apenas as medições onde o deadline foi perdido.

Respostas: a tarefa de temperatura não tem Fator Skip 20, pois todas as amostras respeitaram o deadline definido.

5. Observar uma janela de 20 ativações consecutivas dos dados coletados das duas tarefas. Identificar o pior momento e com isso determinar o valor de m para (m,20)-firme. Plotar um gráfico para cada tarefa mostrando número de deadlines perdidos em uma janela de 20 ativações.

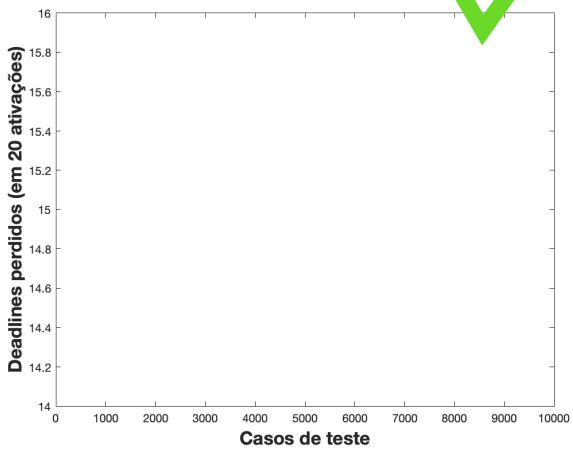

Figura 4: Gráfico mostrando número de deadlines perdidos em janela de 20 ativações.

Respostas: Como todas as amostras cumpriram o deadline definido, então m = 20, o que quer dizer que dentro de uma janela de 20 ativações consecutivas, nenhum deadline foi perdido.

6. Ánalise dos gráficos de temperatura e nível.

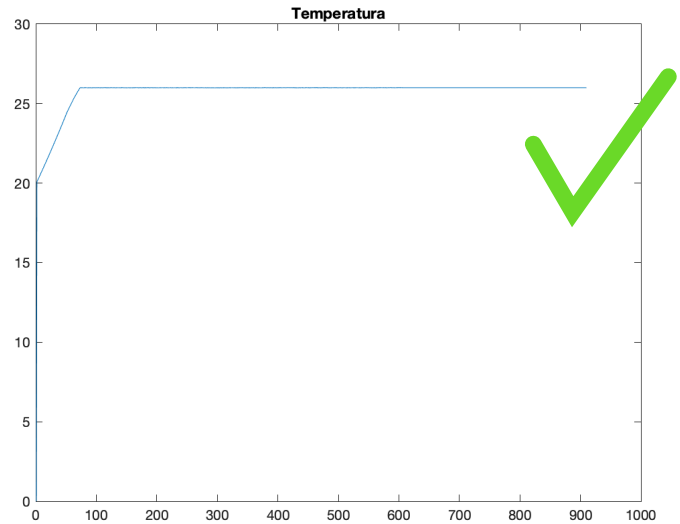

Figura 5: Gráfico mostrando as medições do sensor de temperatura com intervalo de 1 segundo.

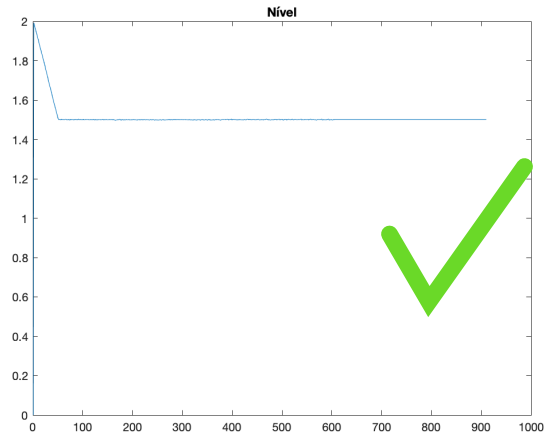

Figura 6: Gráfico mostrando as medições do sensor de nível com intervalo de 1 segundo.

A coleta das 10 mil amostras durou em torno de 8 minutos e 20 segundos, portanto, a cada segundo nesse espaço de tempo foram obtidos cerca de 500 amostras referentes à temperatura e ao nível. No entanto, como a coleta ultrapassou o tempo citado, mais de 600 amostras foram coletas.

Ao analisar ambos os gráficos, é possível ver que nos primeiros momentos de execução, ambas a temperatura e o nível iniciais de 20 ºC e 2 m, respectivamente, passaram a estabilizar rapidamente ao longo do tempo, convergindo para os valores de referência de 26 ºC e 1.5 m.

REFERÊNCIAS

[1] Oliveira, R. S. Fundamentos dos Sistemas de Tempo Real. Original registrado na Biblioteca Nacional. Primeira edição, revisão 3, outubro de 2018. ISBN-13: 9781728694047.

Daniel Alves Farias, estudante do curso de Engenharia da computação na Universidade Federal do Ceará, na cidade de Sobral, no Ceará Campus Mucambinho.

Lucas Natanael Leite, estudante do curso de Engenharia da computação na Universidade Federal do Ceará, na cidade de Sobral, no Ceará, Campus Mucambinho, membro fundador do G.P.I(Grupo de pesquisa e inclusão), atuei como bolsista do P.A.C.C.E.(Programa de aprendizagem cooperativa em células estudantis) durante 3 anos, onde vim a criar o GSEGD(Grupo de Segurança digital), membro fundador do C.A. da Engenharia da Computação onde atuei por 2 anos, atualmente está cursando as últimas disciplinas da faculdade. Atuei durante 6 anos em trabalho voluntário no Rotaract Club de Sobral, Clube filiado ao Rotary, gosto de tecnologias de uma maneira geral, em especial segurança aplicada a IOT, macroeconomia, investimentos e empreendedorismo.

Maxela Martins Pontes, estudante do curso de Engenharia da computação na Universidade Federal do Ceará, na cidade de Sobral,no Ceará Campus Mucambinho,sou formada em Técnica em informática pelo IFCE,atualmente faço parte do grupo de pesquisa TAE (Tecnologias Assistivas e Educacionais).

Yara Silva Braga, estudante do curso de Engenharia de Computação na Universidade Federal do Ceará, na cidade de Sobral-Ce, campus Mucambinho. Atualmente bolsista da Bolsa de Iniciação Acadêmica, atuando em um projeto de Desenvolvimento WEB.

Nota: 10,0
Requisitos do Sistema: 5,0
Medições de tempo real:4,0
Organização do artigo: 1,0