UNIVERSIDADE FEDERAL DO CEARÁ – CAMPUS SOBRAL
CURSO DE ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA: SOFTWARE EM TEMPO REAL
PROFESSOR: REUBER REGIS DE MELO

TRABALHO Nº 01
CONTROLE E SUPERVISÃO DE UM SISTEMA DE CALDEIRA SIMULADO

GIDEÃO LEVI DE OLIVEIRA FROTA	**428557**
KLAYVER XIMENES CARMO	**427651**
SAMUEL GOMES RIBEIRO	**422005**
VITOR CESAR OLIVEIRA GOMES ARRUDA	**428229**

SOBRAL – CE

2021.2

SUMÁRIO

1. RESUMO

Esse trabalho consiste em aplicar os conhecimentos adquiridos na disciplina de Software em Tempo Real, com foco na observação dos dados adquiridos através dos algoritmos implementados para a solução dos problemas, desta forma, foi trabalhado com o problema sugerido "Controle e supervisão de um sistema de caldeira simulado".

O professor da disciplina disponibilizou um simulador do sistema de caldeira desenvolvido em Java com interface gráfica (aprimorada e melhorada pelo professor), em que são setados os valores estabelecidos como: fluxo da água de saída do recipiente (Kg/segundo) "No", temperatura da água que entra no recipiente (graus celsius) "Ti", temperatura do ar ambiente em volta do recipiente (grau celsius) "Ta", entre outros parâmetros. Em seguida alguns problemas são implementados em linguagem C, como: tarefas periódicas de controle, armazenamento de valores lidos dos sensores utilizando buffers duplos, verificação e solução de outras condições, etc.

Dessa forma, com essas informações geradas foi possível obter dados suficientes para realizar controles, testes comportamentais e visão estatística das medições de tempo real.

2. INTRODUÇÃO

Softwares em tempo real possuem uma grande importância nos dias de hoje. Graças a eles foi possível tornar realidade a existência de ferramentas como freios automáticos para carros, controles de temperatura em usinas nucleares, entre outros. Esse tipo de software interage diretamente com o hardware e executa seu papel (normalmente de controle) em resposta aos eventos. Eles são normalmente escritos em linguagens com recursos de baixo nível de programação e tem os requisitos temporais e as threads como mecanismo fundamental.

Por utilizarem threads, é essencial que se tenham formas de evitar leituras/escritas concorrentes no programa. Para isso é utilizado principalmente os mutexes e monitores, uma vez que esses têm o poder de travar os recursos do sistema assim evitando, por exemplo, que dois ou mais processos escrevam valores em alguma variável, o que poderia causar problemas, alguns sendo até críticos ao sistema, no local onde o software está instalado.

O trabalho apresentado tenta simular uma situação onde se pode instalar um software de tempo real: o controle de temperatura e nível em uma caldeira, utilizando-se dos conhecimentos obtidos durante todo o curso da disciplina.

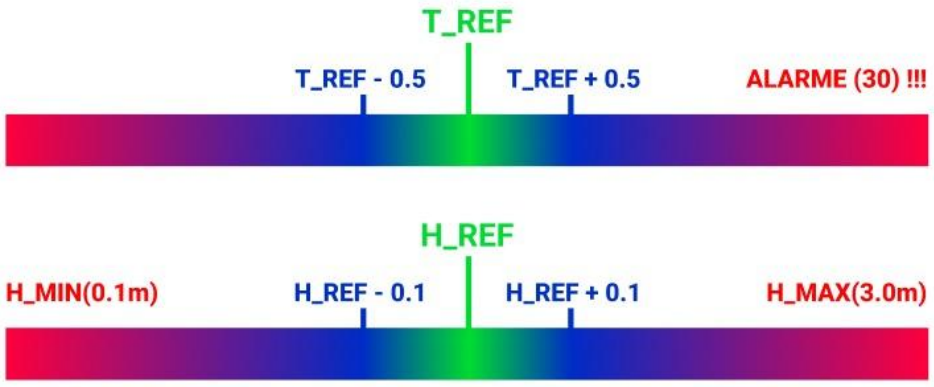
3. METODOLOGIA

O código deve fazer o controle da temperatura e do nível da caldeira, e são preciso de algumas lógicas para fazer esse controle em sincronia. Em um dos requisitos na descrição do trabalho foi pedido que seja possível que o usuário digite um valor de referência para a temperatura e para o nível. Nos testes feitos, foram colocados como valores de referência: 20ºC para a temperatura e 2m para o nível. Para não gerar conflitos (pegar valores antigos ou misturá-los), foi feita uma alocação diferente para cada valor, então cada um tem um monitor para realizar o gerenciamento. Eles são colocados em arquivos diferentes (um para o nível e outro para temperatura).

Após isso é criada uma conexão via *socket* (apresentada nas aulas), e depois a threads são criadas. Suas funções são: mostrar os status, ler o sensor, alarme (que será acionado em tela, quando a temperatura chegar em 30º), controlar temperatura e o nível e gravar o tempo de resposta da temperatura e do nível.

Para entender melhor a lógica do código, foi feito um arquivo em forma de pseudocódigo, chamado **flowControl.md**. É apresentado todos os cenários possíveis que podem acontecer durante a simulação: temperaturas baixas, médias e altas, além de níveis baixos, médios e altos.

Para os valores médios de temperatura e nível foram feitos valores de transição para delimitar qual é o escopo do valor médio de temperatura e nível. Os valores de referência da temperatura são maiores que os do nível, definido então como 0.5 o valor de transição. Logo, de acordo com o valor de referência de entrada, o mesmo terá um intervalo de 0.5 a menos ou a mais, e estando nesse intervalo, ainda continuará com uma temperatura média. O mesmo acontece para os valores de nível, que, por sua vez, têm valores menores do que a temperatura, então o valor de transição foi definido com o valor de 0.1.A imagem abaixo apresenta o funcionamento entre os valores de referência (em verde) e os valores de transição (em azul).



O controle da temperatura é atualizado a cada 50ms, e o do nível é atualizado a cada 70ms. Todos os valores (temperatura da água, do ambiente, da água que entra, e fluxo de saída e nível de água) são mostrados em tela, além da temperatura e o nível de

referência, que foram definidos no início do programa pelo usuário. Os valores são calculados antes do controle ser feito, o tempo de resposta é calculado e armazenado através de um buffer duplo. É feito um arquivo com o nome **dados_sensores.txt** para guardar os tempos de respostas da thread de controle da temperatura.

Obs.: Foi feito uma thread no repositório do projeto para armazenar também os tempos de resposta do controle de nível, pois foi preciso para fazer análise do mesmo, sendo criado um arquivo com nome **dados_nivel.txt.**

Foi criado um arquivo **Makefile**, que é uma maneira de executar os arquivos no Linux de forma mais fácil. Ele contém flags, bibliotecas, monitores e hosts necessários para rodar os códigos. O arquivo contém comandos para compilar e executar os arquivos, além de limpar a tela para melhor visualização.

É importante frisar que é recomendável executar os arquivos **Makefile** (simulador e controle) via **WSL** ou **Linux**, pois via Windows a conexão das portas não acontecia.

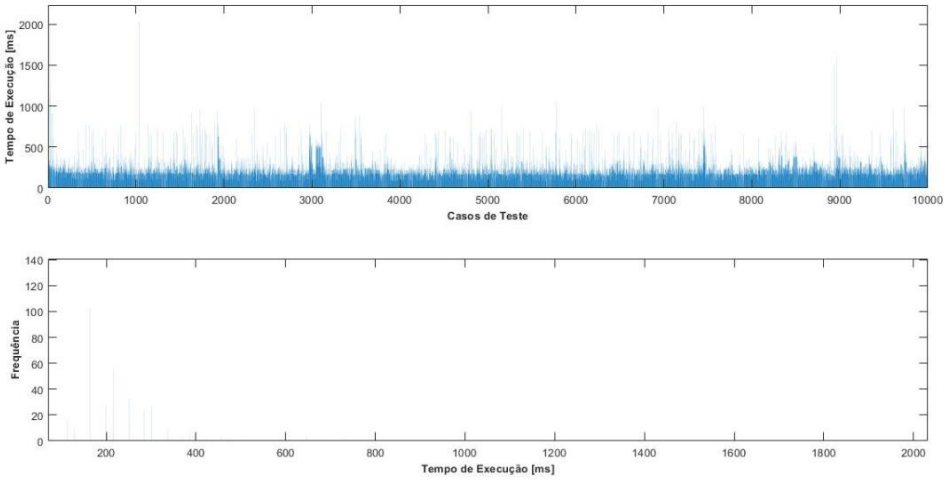
4. RESULTADOS E DISCUSSÕES

Obs: os códigos feitos para as testagens foram feitos em matlab (arquivos .m), porém como pode ser que o professor não tenha, no arquivo do trabalho os códigos foram colocados em formato de texto (txt). Os gráficos colocados neste relatório foram apenas analisados e utilizados os dados armazenados da temperatura, porém também é gerado no código um arquivo do armazenamento do nível para caso seja necessário uma plotagem e análise dos gráficos.

O tópico 1 pede os valores mínimo e médio do tempo de resposta da tarefa periódica com número de amostras igual a 10.000, obtendo os seguintes resultados: Tempo de Resposta Mínimo = 70 e o Tempo de Resposta Médio = 228.0998.

A Figura 1 expressa o tópico da parte 1 e 2, a 2 sendo o tempo de resposta máximo observado, também conhecido como HWM, dos mesmos 10.000 valores obtidos. Tempo de Resposta Máximo(HWM) = 2032. Não são suficientes. Foi executado em um desktop com sistema operacional Windows por meio do WSL de propósito geral, não voltado para sistemas de tempo real.

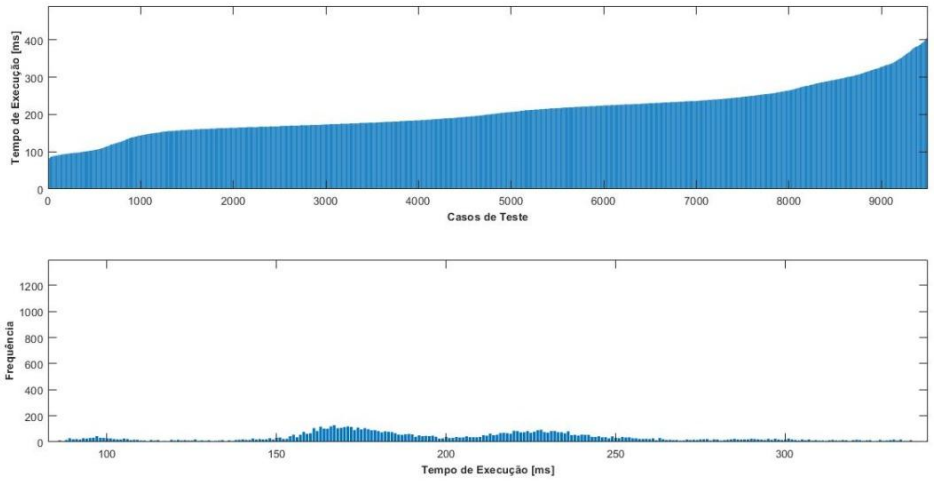
Figura 1: Gráfico com as medições realizadas, na ordem dos casos de teste



Fonte: próprio autor

A Figura 2 expressa o tópico 3 que é o tempo de resposta máximo observado, também conhecido como HWM, utilizando 95% do número das amostras, pois essa porcentagem foi suficiente para cumprir o deadline. Tempo de Resposta Máximo(HWM) = 409.

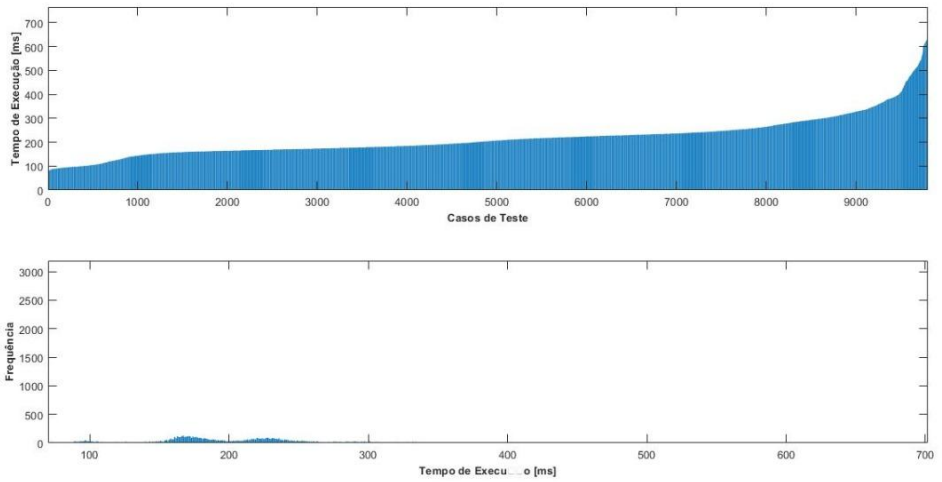
Figura 2: Histograma das medições realizadas



Fonte: próprio autor

A Figura 3 expressa o tópico 3 que é o tempo de resposta máximo observado, também conhecido como HWM, porém usando 98% do número de amostra, pois assim como o teste anterior de 95% esse valor também é suficiente para cumprir o deadline. Tempo de Resposta Máximo(HWM) = 638.

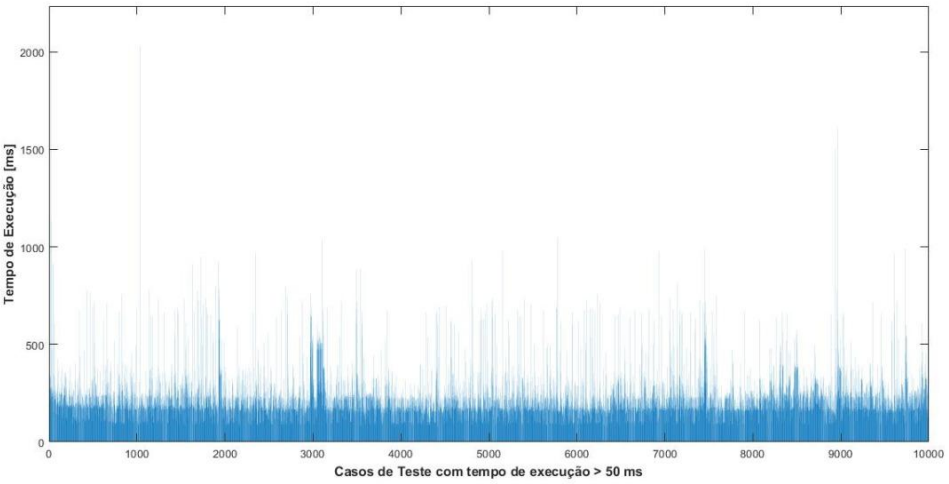
Figura 3: Histograma das medições realizadas



Fonte: próprio autor

A Figura 4 expressa o tópico 4 que é as medições onde o deadline foi perdido, mostrando assim os testes que tem um tempo de execução maior que 50 ms.

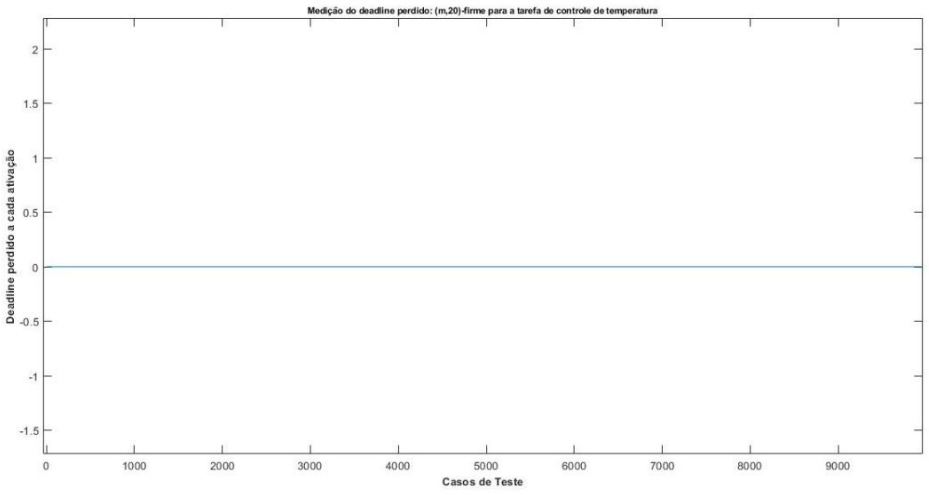
Figura 4: Gráfico mostrando apenas as medições onde o deadline foi perdido



Fonte: próprio autor

A Figura 5 expressa o tópico 5 número de deadlines perdidos em intervalo de 20 ativações, porém mesmo fazendo mudanças no código feito para mostrar o gráfico da melhor disposição possível ainda assim não ficou como desejado. Logo as análises e conclusões do mesmo foram inconclusivas.

Figura 5: Gráfico mostrando número de deadlines perdidos em janela de 20 ativações.



Fonte: próprio autor

Obs.: Todas as imagens, códigos fonte e dados utilizados dos sensores de temperatura e nível estão no repositório com diretório `~/str-trabalho01/Graficos`.