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

PROJETO SEMESTRAL V

RELATÓRIO DA DISCIPLINA DE SISTEMAS OPERACIONAIS E TEMPO REAL
Prof. Frederico M. Schaf

Vitor Fernandes, Bruno Sampaio e Eduardo Goulart

Santa Maria, RS, Brasil

2021

SUMÁRIO

LISTA DE FIGURAS

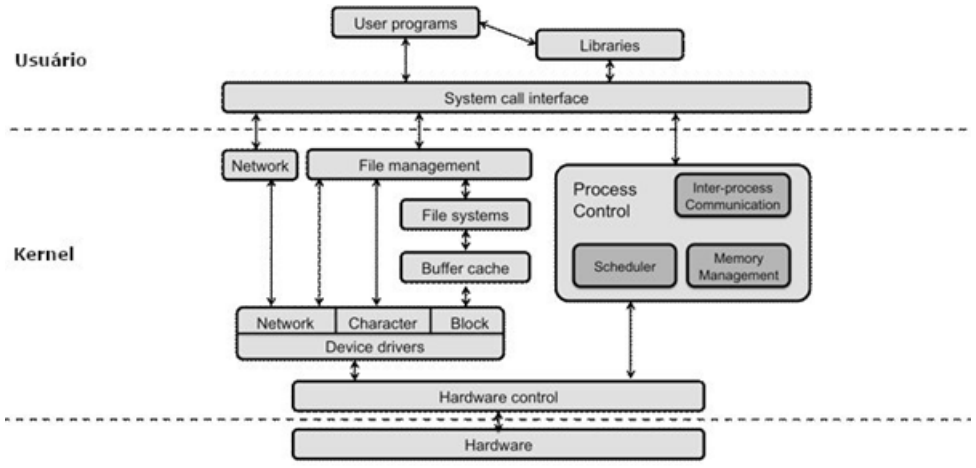
1 RESUMO

Quando usamos um computador desktop, notebooks, telefones celulares, smartphones ou qualquer outro eletrônico cotidiano como videogames ou até mesmo carros com computador a bordo, estamos utilizando um produto capaz de rodas múltiplos processos paralelamente e de forma ordenada, executando tarefas muitas vezes complexas. Essa capacidade dos eletrônicos de rodar paralelamente processos complexos sem erros é dada graças ao seu Sistema operacional (SO).

Partindo do conceito de Sistema Operacional sendo "Um conjunto de softwares cuja função é administrar e gerenciar os recursos de um sistema, desde componentes de hardware e sistemas de arquivos a programas de terceiros, estabelecendo a interface entre o computador e o usuário.". (TECNOBLOG, 2021) Dessa forma, um SO é uma forma de abstração entre o hardware e software, garantindo que usuário e máquina consigam trabalhar em conjunto sem que haja conflitos.

Para que haja essa segurança, os processos são divididos em Processos de Usuário e Processos de Kernel. O primeiro diz respeito às funções que um usuário ordinário pode fazer dentro do SO e o outro diz respeito a funções que somente "super" usuários ou o próprio SO pode assumir, sendo nessa camada onde as funções de Hardware podem ser acessadas diretamente. A Figura 1 ilustra essa ideia de divisão de processos Usuário / Kernel e o Hardware sendo a parte física do sistema.

Figura 1.1 – Estrutura genérica de um sistema operacional, separado em kernel e interface de usuário



Fonte: Autor "tecnoblog/o-que-e-um-sistema-operacional"

2 INTRODUÇÃO

Tendo um conhecimento prévio do que é um Sistema Operacional e Tempo Real (SOTR) apresentado em aula, estimasse que seremos capazes de criar ou replicar um sistema ao qual consiga adotar todas as principais características de um SOTR. Como forma de solucionar o desafio proposto como Trabalho semestral da disciplina de Sistemas Operacionais e Tempo Real, pelo curso de Engenharia de controle e automação na UFSM (Universidade Federal de Santa Maria), iremos criar um sistema genérico e de propósitos didáticos para fazermos a Monitoração/Controle tempo real com auxilio computacional do sistema e a partir dele, fazer uma análise temporal de resposta do sistema e discutir acerca dos resultados obtidos nele.

Como Hardware será utilizado:

- 2 Arduino (Não importa o modelo, qualquer Microcontrolador genérico assume o papel);
- 3 Computadores com acesso à Internet, interligados por uma rede virtual sob um VPN;
- 1 Potenciômetro para controle;
- 1 Servo atuador para ser controlado.

Como Softwares será utilizado:

- Linguagem de programação C/C++ (Wiring) para programação dos Arduinos através da interface de desenvolvimento integrada (IDE) disponibilizada por Arduino.cc (ARDUINO, sd);
- Linguagem de programação Python para a criação dos sistemas (PYTHON, sd);
- Ambiente de programação genérico.

Tendo as definições de projeto especificadas e enquadradas na Possibilidade 5 de propostas de trabalhos semestrais apresenta em 20/01/2021, seguiremos o projeto com o compromisso dado a cima.

3 OBJETIVO

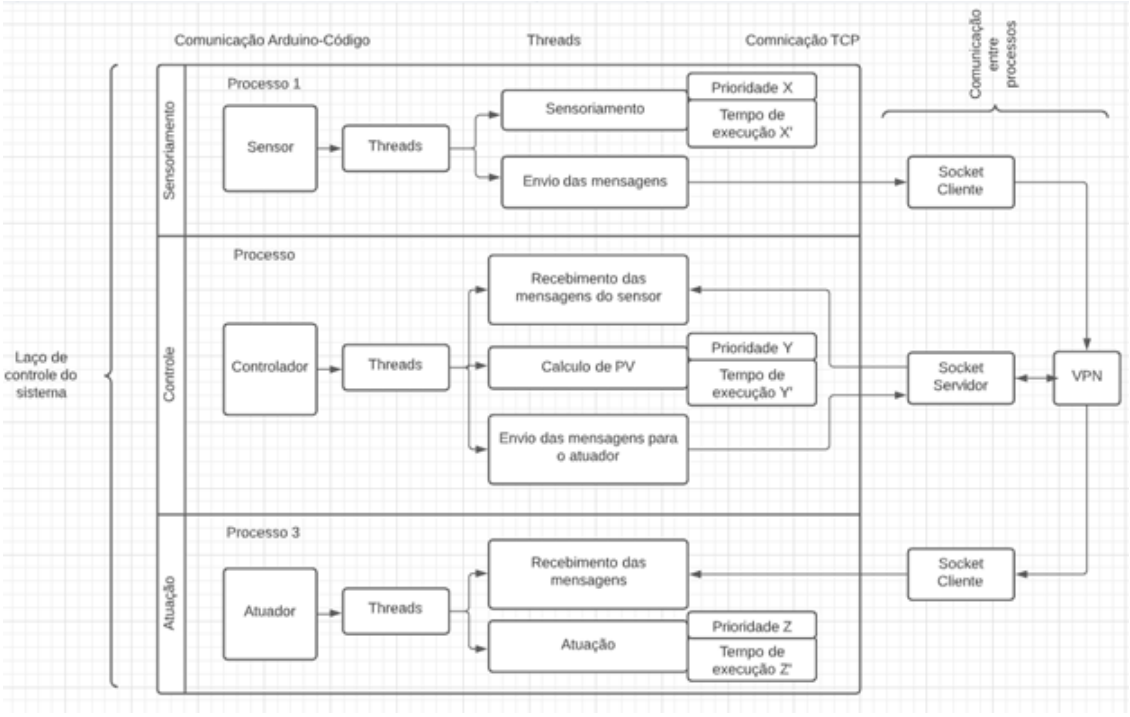
Tendo como o objetivo no trabalho, demonstrar o que foi aprendido em sala de aula acerca dos conceitos de SOTR, propomos a criação de um sistema que adote alguns dos conceitos de pré-determinismo temporal e que seja capaz de responder a esse determinismo sem atrasos ou falhas.

Como o objetivo é puramente didático, simulamos uma situação que poderia ser adotada em uma ocasião real a fim de facilitar o desenvolvimento do projeto, para isso foram idealizados 3 sistemas que serão executados de forma independente e simultaneamente.

Pensando que em uma situação real, existem múltiplos processos sendo executados paralelamente dentro do chão de fábrica e que muitas vezes, a troca de informação entre os processos seja feita através de uma rede virtual privada, adotamos o mesmo sistema e foi distribuído um sistema para cada integrante do grupo ao qual cada um se conectaria remotamente através de um VPN (Virtual Private Network) remoto, uma vez que no âmbito de pandemia, não podemos nos reunir para a execução do projeto.

Os 3 sistemas foram classificados de acordo com a função de cada um e seguindo a ideia de sensor -> controlador -> atuador, usamos o mesmo conceito como ilustrado na Figura 3.1.

Figura 3.1 – Modelo esquemático do trabalho - lucidchart



Fonte: Autores

4 DESENVOLVIMENTO

Todos os códigos dos arquivos foram disponibilizados na plataforma do GitHub contidos nas referências.Como pode ser visto no fluxograma da imagem 3.1, o presente projeto utiliza Threads nos três processos comentados no Objetivo, devido a necessidade de ter um fluxo de programas executados em paralelo.

4.1 Código Sensor

O Código sensor pode ser dividido em duas partes, como mostrado no fluxograma da Figura 3.1 na seção Sensor. A primeira trata da aquisição dos dados de medição que simulam um processo real e a segunda trata dos envios desses dados para um servidor, ficando dividido como Sensoriamento e Envio de mensagens:

4.1.1 Sensoriamento

O sensoriamento desenvolvido no projeto, simula um processo real, onde um sensor faz suas medições e retorna um valor 4bytes. Por falta de componentes a disposição, optou-se por executar um código que gerasse de forma aleatória esses parâmetros. Tais parâmetros são gerados de forma pseudo-randômica pelo esp8266 e enviados como dito na seção 4.1.2.

O valor gerado de forma randômica esta entre 0 e 180 elevado ao quadrado, simulando os graus de atuação de um servo atuador. Os valores elevados ao quadrado, sugerem que os dados precisam ser tratados pelo servidor antes de serem atuados.

Os dados são aquisitados usando uma conexão Serial de 9600 bps entre o esp8266 e um computador que roda um script em python, com uma periodicidade de leitura do sensor de 500ms.

4.1.2 Comunicação

O Código sensor estabelece a comunicação com o Servidor como um Cliente para o envio de dados. Tal comunicação ocorre toda vez que um novo dado é recebido via Serial do Esp8266 e roda independente da aquisição do Serial.

4.2 Controlador

O controlador é o responsável pelo tráfego de informações, atuando como servidor TCP do processo. Os dois clientes do servidor seria o Esp8266 atuando como sensor e o Arduino Nano sendo responsável pelo servo atuador (Atuador). Além disso, o controlador é responsável pela manipulação dos dados obtidos pelo sensor, ou seja, existe uma função responsável por efetuar cálculos com os parâmetros obtidos pelo sensor, após os cálculos serem feitos ângulos são enviados ao atuador via a mesma conexão TCP.

Os valores de ângulos recebidos pelo sensor possuem 5 bytes sendo um deles um byte de identificação padrão 'S' e envia os dados periodicamente a cada 500ms idealmente.

Para o atuador fazer a aquisição dos valores do sensor, o cliente envia um byte especial 'A' que ao chegar no servidor, é entendido como a solicitação de mensagem e logo é respondido os 4 bytes de valor do sensor. Nesse modelo de troca de mensagens, o atuador solicita mensagens a todo instante e sempre que há um novo valor a ser enviado, o servidor os envia.

Para conseguir acompanhar ambos envios sem haver perda de sincronia entre as comunicações, o servidor roda em paralelo ambas trocas de mensagens Recebimento/Envio, através de Threads.

4.3 Código Atuador

O Código atuador, assim como o Código Sensor, pode ser dividido em duas partes. A primeira delas é a parte da comunicação, onde envia e recebe valores que serão usados para atuação via conexão TCP e a segunda é a conexão do script em pyhton para o arduino Nano via conexão serial, sendo a primeira identificada no Fluxograma 3.1 como Recebimento das mensagens e a segunga como Atuação, ambas rodam em paralelo e independentes uma das outras:

4.3.1 Recebimento das mensagens

O código do Atuador estabelece 2 conexões interprocessos, sendo a primeira com o servidor do sistema como um Cliente, onde faz a solicitação e recebimento dos dados dos sensores usados para a atuação e a segunda conexão estabelecida é a comunicação com o Arduino Nano via serial, onde os dados de atuação são enviados para o Arduino Nano para serem atuados.

4.3.2 Atuação

Foi usado um servo motor/atuador conectado em um Arduino Nano. Os dados são recebidos no arduino via serial em 4 bytes, ao qual é manipulado e transformado em um número inteiro que será a informação responsável por manipular o servo motor.

Assim que o arduino processa as informações e manipula o servo motor, ele responde na serial o valor que recebeu e atuou, para ser conferido via script se esta tudo certo.

4.4 Código fonte

Os códigos usados e todas modificações do projeto que foram realizadas podem ser encontradas na pltaforma GitHub, um ambiente para trabalhos de software compartilhado e pode ser acessadas pelo link $https://github.com/iOsnaaente/SistemasOperacionais_TempoReal$ (GITHUB, sd).

5 CONCLUSÃO

Para fazer os testes, visando que cada um dos membros do grupo ficou encarregado por um dos scripts do sistema e não estávamos na mesma rede local, usamos uma VPN (Virtual Privite Network) Hamachi (HAMACHI, sd) que cria uma rede virtual. Utilizou-se o software Wireshark (WIRESHARK, sd) para efetuar a aquisição dos dados para validação do projeto. Foram efetuadas 147 medições de comunicação, a figura 5.1 é um exemplo dos dados coletados.

Figura 5.1 – Exemplo Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	25.59.115.113	25.114.157.253	TCP	60	52443 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=65136 Len=1
2	0.039748	25.114.157.253	25.59.115.113	TCP	54	1234 → 52443 [ACK] Seq=1 Ack=2 Win=64788 Len=0
3	0.081914	25.94.9.254	25.114.157.253	TCP	60	55416 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=65472 Len=5
4	0.122235	25.114.157.253	25.94.9.254	TCP	54	1234 → 55416 [ACK] Seq=1 Ack=6 Win=65277 Len=0
5	0.389609	25.59.115.113	25.114.157.253	TCP	60	52443 → 1234 [PSH, ACK] Seq=2 Ack=1 Win=65136 Len=1
6	0.394047	25.114.157.253	25.59.115.113	TCP	58	1234 → 52443 [PSH, ACK] Seq=1 Ack=3 Win=64787 Len=4
7	0.481704	25.59.115.113	25.114.157.253	TCP	60	52443 → 1234 [ACK] Seq=3 Ack=5 Win=65132 Len=0
8	0.614078	25.94.9.254	25.114.157.253	TCP	60	55416 → 1234 [PSH, ACK] Seq=6 Ack=1 Win=65472 Len=5
9	0.649472	25.59.115.113	239.255.255.250	SSDP	535	NOTIFY * HTTP/1.1
10	0.653261	25.114.157.253	25.94.9.254	TCP	54	1234 → 55416 [ACK] Seq=1 Ack=11 Win=65272 Len=0
11	0.655690	fe80::141e:c36:d76e…	ff02::c	SSDP	562	NOTIFY * HTTP/1.1
12	0.701208	25.59.115.113	25.114.157.253	TCP	60	52443 → 1234 [PSH, ACK] Seq=3 Ack=5 Win=65132 Len=1
13	0.704206	25.114.157.253	25.59.115.113	TCP	58	1234 → 52443 [PSH, ACK] Seq=5 Ack=4 Win=64786 Len=4

Fonte: Autores

Os dados foram tratados com um algorítimo e obteve-se os valores do tempo decorrido da mensagem ser enviada pelo Código Sensor e ser recebida no Código Atuador. Os dados do Wireshark foram manipulados e se obteve a Tabela 5.1.

Foram feitas as seguintes manipulações:

- Obtenção do Tempo sensor (Ts) e do Tempo Atuador (Ta);
- Obtenção do ΔTempo (Ta - Ts);
- Obtenção do ΔTempo máximo;
- Obtenção do ΔTempo mínimo.

Tendo os dados de tempo do envio do sensor e recebimento do atuador, pode-se calcular o tempo de atraso entre uma solicitação e outra e o tempo de transito das mensagens sendo esse o principal fator medido no sistema, pois para haver um pré-determinismo temporal, é necessário que esse tempo seja respeitado rigorosamente para o sistema ganhar o nome de Sistema de Tempo Real. Na tabela 5.1 podemos ver os dados coletados da troca de mensagens entre os sistemas iniciais (coleta dos dados no sensor) e finais (atuação do sistema):

Tabela 5.1 – Tempos Wireshark

Tempo Sensor	**Tempo Atuador**	**Δ Tempo**
0.081914	0.394047	0.312133
0.614078	0.704206	0.090127
1.135127	1.384053	0.248925
1.649793	1.675822	0.026029
2.164445	2.435237	0.270791
2.710066	2.752023	0.041957
3.203905	3.547358	0.343453
3.728383	3.813027	0.084643
4.251557	4.455151	0.203593
4.824809	5.085352	0.260543
5.310707	5.386255	0.075548
6.397668	6.405369	0.007700
6.925579	6.956572	0.030993
7.386371	7.716844	0.330472
7.982457	8.377651	0.395193
8.439792	8.718914	0.279121
8.964660	9.040580	0.075919
$\mathbf{\Delta Tempo_{máx}}$	$\mathbf{\Delta Tempo_{min}}$	$\mathbf{\Delta Tempo_{médio}}$
0.395193	0.007700	0.1810088

Fonte: Autores

Onde,

$\Delta Tempo =$ Tempo sensor $-$ Tempo Atuador;

$\Delta Tempo_{\text{médio}} =$ media aritmética de $\Delta Tempo$;

$\Delta Tempo_{\text{máx}} =$ maior valor de $\Delta Tempo$;

$\Delta Tempo_{\text{min}} =$ menor valor de $\Delta Tempo$;

A partir da tabela 5.1 pode-se concluir que há um respeito da periodicidade no sistema, mesmo com altas flutuações nas medições $\Delta T_{\text{máx}}$ e ΔT_{mim} causadas pelo VPN utilizado, mas que mesmo assim, alcançam valores lineares de tempo de resposta. Estimando-se que o sistema possua uma periodicidade de atuação de 1 segundo, podemos validar o projeto dentro dos requisitos de Tempo Real.

Ao utilizar uma VPN, estamos criando uma rede virtual e automaticamente aumentamos o tempo de tráfego das mensagens TCP trocadas, dessa forma, se o mesmo sistema estiver rodando dentro de uma rede local, tais valores de atraso cairiam drasticamente, trazendo ainda mais usabilidade ao sistema. Como forma didática e com fins de apresentação, optou-se por fazer as medições dos resultados através do VPN, para que todos os integrantes do grupo possam interagir entre si de forma remota e por isso aceita-se ceder esse atraso.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] TECNOBLOG, O que é um sistema operacional? - https://tecnoblog.net/303055/o-que-e-um-sistema-operacional/ - Acesso em 13/03/2021 as 00:45.

[2] ARDUINO, Home Page Arduino - https://www.arduino.cc/ , sd – Acesso em 13/03/2021 as 01:10.

[3] PYHTON, Home Page Python - https://www.python.org/ , sd – Acesso em 13/03/2021 as 01:21.

[4] HAMACHI, LogmeIn - https://www.vpn.net/ , sd – Acesso em 13/03/2021 as 13:52.

[5] WIRESHARK, Home Page - https://www.wireshark.org/ , sd – Acesso em 13/03/2021 as 15:02.

[6] GITHUB, Home Page - https://github.com/ , sd – Acesso em 13/03/2021 as 15:33.