

TRABALHO SEMESTRAL

SISTEMAS OPERACIONAIS E TEMPO REAL

Bruno Gabriel Flores Sampaio

Eduardo ~~

Vitor Fernandes ~~

Versão digital

Santa Maria - RS

2021

Sumário

[Resumo 3](#_Toc66491691)

[Introdução 4](#_Toc66491692)

[Objetivo 5](#_Toc66491693)

[Sensor 6](#_Toc66491694)

[Controlador 6](#_Toc66491695)

[Atuador 6](#_Toc66491696)

[Desenvolvimento 6](#_Toc66491697)

[Conclusão 6](#_Toc66491698)

[Bibliografia 6](#_Toc66491699)

Figuras

[Figura 1: Modelo esquemático do trabalho 3](#_Toc66491168)

[Figura 2: Estrutura genérica de um sistema operacional, separado em kernel e interface de usuário 4](#_Toc66491169)

Tabelas

**Nenhuma entrada de índice de ilustrações foi encontrada.**

# 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 à 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.

Autor “tecnoblog/o-que-e-um-sistema-operacional”

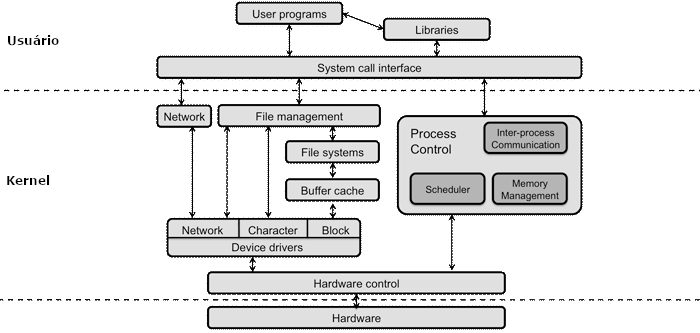


Figura 1: Estrutura genérica de um sistema operacional, separado em kernel e interface de usuário

Apesar de que um SO lide com todo o gerenciamento de processos, muitas vezes, esses processos possuem um pré-determinismo temporal, onde determinada saída de dados possui data e hora certas para serem processados, caso contrário, podem comprometer todo o processo final e nesses casos um SO deve ser dotado de uma característica especial, o Tempo Real (*Real Time* – RT).

Para processos críticos e que possuam restrições temporais grandes, um sistema operacional precisa garantir que tais processos respeitem seus tempos de processamento, podendo citar de exemplo sistemas como:

1 – Sistemas de radar aeroespaciais que detectam as posições e rotas de uma aeronave para evita colisões.

2 – Sistema do Rover que opera de forma autônoma em Marte.

3 – Inputs do teclado em um computador convencional que se não tratado a tempo, pode causar a sensação de travamento.

No primeiro caso, temos um sistema crítico, ao qual a falha do sistema pode causar morte de pessoas, sendo classificado como “*Hard Real-Time Systems*”. No segundo caso, apesar de termos um caso que não põem em risco a vida de nenhuma pessoa, gera danos irreparáveis ao processo e a perda de muitos bilhões de reais, levando a mesma classificação. Ao contrário do terceiro caso, onde nenhum das duas situações se encaixa, sendo classificado como “*Soft Real-Time Systems*”

Para um sistema ser classificado como RT, precisa-se que o mesmo possua algum tipo de pre-determinismo temporal, não necessariamente exigindo uma resposta imediata ao problema que deve solucionar, podendo muitas vezes ser até “demorado” em comparação a outros sistemas, desde que ele nunca perca o seu tempo de resposta já determinado pelo processo.

# 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 a cerca dos resultados obtidos nele.

Como *Hardware* será utilizado:

2 Arduinos (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.

# Objetivo

Tendo como o objetivo no trabalho, demonstrar o que foi aprendido em sala de aula a cerca 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 2.

Autor - Próprio autor

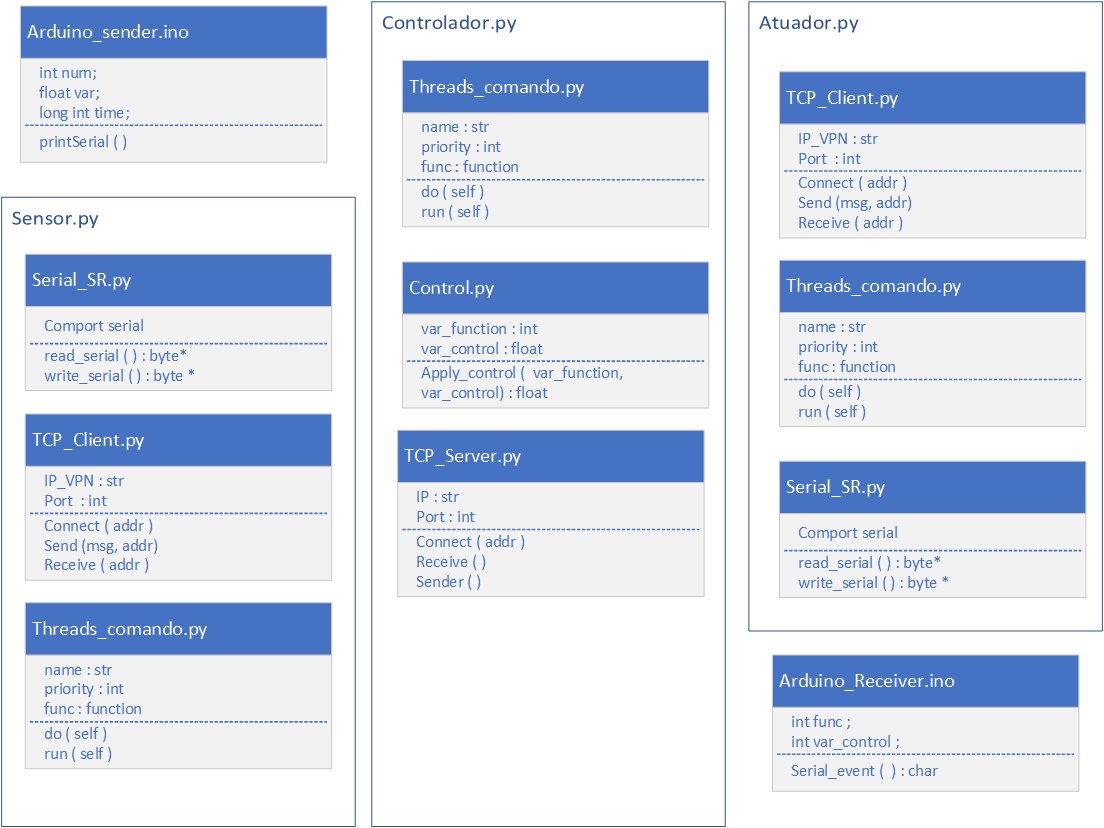


Figura 2: Modelo esquemático do trabalho

Autor - Próprio autor

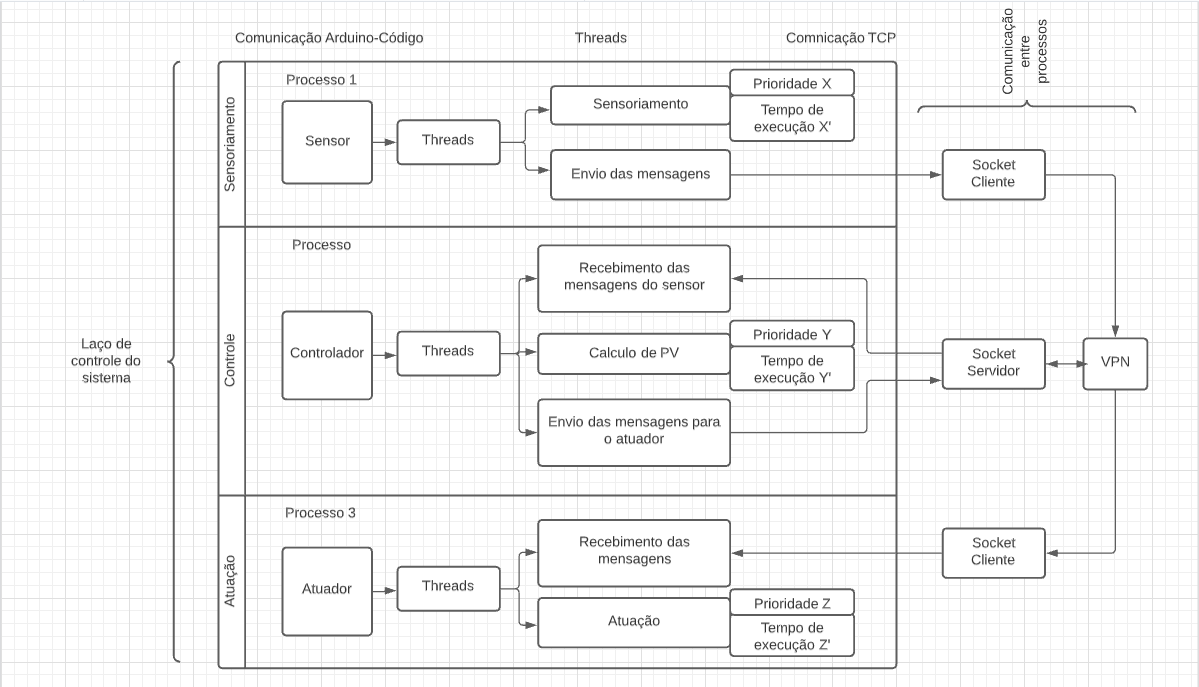


Figura 3- Fluxograma

# Desenvolvimento

## Código Sensor

O Código sensor pode ser dividido em duas partes:

### Comunicação

Como o próprio nome sugere, o Código sensor estabelece a comunicação com o servidor como um Cliente para o envio de dados e também estabelece a comunicação com o Esp8266 para a obtenção dos parâmetros gerados.

### Obtenção de dados.

Por falta de componentes a disposição, optou-se por executar um código que gerasse de forma aleatória parâmetros. Estes parâmetros são gerados de forma randômica pelo esp8266 e enviados como dito na seção 1.1.

O valor gerado de forma randômica estão entre 0 e 180, simulando os graus de atuação de um servo atuador. Tais valores são elevados ao quadrado e enviados via serial para o script em Python responsável pela comunicação Cliente/Servidor.

## Controlador

O controlador é o responsável pelo tráfego de informações, atuando como servidor 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.

Os valores de ângulos recebidos pelo sensor possuem um

## Código Atuador

O Código atuador, assim como o Código Sensor, pode ser dividido em duas partes:

### Comunicação

Como o próprio nome sugere, o Código Atuador estabelece a comunicação com o servidor como um Cliente, e também estabelece a comunicação com o Arduino X.

Atuação.

O atuador como dito anteriormente, é um servo motor e o microcontrolador Arduino X é responsável por manipular o servo motor de acordo com as informações passadas.

# Conclusão

Asuhiasijsa

# Bibliografia

[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.