UNIVERSIDADE DE LUANDA

# INSTIC

INSTITUTO DE TECNOLOGIAS DE INFORMAÇÃO ECOMUNICAÇÃO

PROJECTO DE PESQUISA

**Curso:** Engenharia Informática
**Turma:**5º ano A

**Título:** Sistemas Operacionais de Tempo Real: Aspectos Funcionais. Tarefas e Threads. Interrupções

Trabalho De Sistema Em Tempo Real

Professor:

Luanda, 2023

UNIVERSIDADE DE LUANDA

**INSTIC**

INSTITUTO DE TECNOLOGIAS DE INFORMAÇÃO ECOMUNICAÇÃO

PROJECTO DE PESQUISA

**Curso:** Engenharia Informática
**Turma:**5º ano A

**Título:** Sistemas Operacionais de Tempo Real: Aspectos Funcionais. Tarefas e Threads. Interrupções


## Integrantes do Grupo 6:

Joel Malamba – 2037
Leonel Diogo – 2050


Trabalho De Sistema Em Tempo Real

Professor

Luanda, 2023

# Índice

# 1. Introdução

Um sistema operacional em tempo real (RTOS) é um SO leve usado para fácil multitarefa e integração de tarefa em recurso e projetos restritos de tempo, que normalmente é o caso de sistemas embarcados.

## 1.1. Objetivo

### 1.1.1. Objetivo Geral

O objetivo grela deste presenta trabalho é de apresentar os conceitos relevantes sobre os Sistemas Operacionais Em Tempo Real (RTOS).

### 1.1.2. Objetivos Específicos

- Abordar sobre os aspectos funcionais dos RTOS
- Abordar sobre as Tarefas(Threads)
- Aplicações do RTOS
- Vantagens e desvantagens dos RTOS

# 2. Sistema Operacionais Em Tempo Real

Um Sistema Operativo em Tempo Real (RTOS da sigla Real Time Operating System) é um sistema operacional/operativo destinado à execução de múltiplas tarefas onde o tempo de resposta a um evento (externo ou interno) é pré-definido. Esse tempo de resposta é chamado de prazo da tarefa e a perda de um prazo, isto é, o não cumprimento de uma tarefa dentro do prazo esperado, caracteriza uma falha do sistema.

Outra característica dos sistemas de tempo real é a sua interação com o meio ao redor. Os STR (Sistemas de Tempo Real) têm que reagir, dentro de um prazo pré-definido, a um estímulo do meio. Por exemplo, em um hospital, o sistema que monitora os batimentos cardíacos de um paciente deve avisar os médicos caso haja alteração nos batimentos. Outro aspecto importante dos STR é a previsibilidade. O sistema é considerado previsível quando podemos antecipar seu comportamento independentemente de falhas, sobrecargas e variações de hardware.

Um RTOS não tem que ter necessariamente um elevado débito nas saídas, ou um elevado número de saídas, no entanto, tem que garantir que certas tarefas sejam executadas em um determinado intervalo de tempo. Um RTOS é mais eficaz e é mais valorizado pela forma previsível e rápida na resposta a um evento, do que pela quantidade de dados que processa.

### 2.1.1. Aspectos funcionais

Existem diferentes tipos de funcionalidades básicas de um RTOS que seguem

- Agendamento de Tarefas;
- Rotina de interrupção do relógio do sistema;
- Comportamento determinístico;
- Sincronização e Mensagens.

### 2.1.2. Agendamento de Tarefas

O RTOS usa um algoritmo de agendamento para determinar a ordem de execução das tarefas. Existem diferentes algoritmos de agendamento, como FIFO (First-In-First-Out), Round Robin, Prioridade, entre outros. O agendador atribui prioridades às tarefas com base em sua importância e requisitos temporais e decide qual tarefa deve ser executada em determinado momento.

### 2.1.3. Comportamento Determinístico

O comportamento determinístico é fundamental para garantir que as tarefas críticas sejam concluídas dentro dos prazos estabelecidos, evitando atrasos e variações inesperadas. É uma característica essencial para aplicações em tempo real, onde a consistência e a previsibilidade são requisitos-chave.

### 2.1.4. Rotina de interrupção do relógio do sistema

Um RTOS lida com interrupções de hardware e eventos externos de forma eficiente. Quando

ocorre uma interrupção, o RTOS pode pausar temporariamente a tarefa em execução para atender à interrupção e, em seguida, retomar a tarefa no ponto em que parou. Isso permite uma resposta rápida a eventos externos e ajuda a garantir a execução pontual das tarefas críticas.

### 2.1.5. Sincronização e Mensagens

A sincronização e as mensagens fornecem a comunicação entre a tarefa de um sistema para outro sistema e os serviços de mensagens estão seguindo. Para sincronizar as atividades internas é usado o flag de evento e para enviar as mensagens de texto que podemos usar na caixa de correio, canais e filas de mensagens. Nas áreas de dados comuns, os semáforos são usados.

- Semáforos;
- Bandeiras de eventos;
- Caixas de correio;
- Tubos;
- Filas de mensagens.

## 3. Tarefas(Threads)

As tarefas (também poderia ser chamada de processo/threads): São funções independentes que executam em loops infinitos, em geral cada um responsável por uma característica. As tarefas se executam de forma independente do seu próprio tempo (isolação temporal) e pilha de memória (isolação espacial).

**Isolação espacial**: A isolação espacial entre threads pode ser garantida com o uso de uma unidade de hardware de proteção de memória (MPU), que restringe a região de memória acessível e aciona exceções de falha na violação de acesso. Normalmente, os periféricos internos são memórias mapeadas, logo uma MPU pode ser usada para restringir o acesso a periféricos também.

### 3.1. Estados Entre Tarefas

As tarefas podem estar em estados diferentes:

- **Bloqueado:** O thread está aguardando um evento (por exemplo, tempo limite de atraso, disponibilidade de dados/recursos)
- **Pronto:** O thread está pronto para executar na CPU, mas não está executando porque a CPU está em uso por outra tarefa
- **Execução:** O thread está atribuído para ser executada na CPU

### 3.2. Agendador

Os agendadores em RTOS controlam qual tarefa a ser executada na CPU, com disponibilidade de diferentes algoritmos de agendamento. Normalmente, eles são:

- **Preemptivo:** A execução de thread pode ser interrompida, caso outro thread com prioridade mais alta estiver pronto;
- **Cooperativo:** A troca de tarefas só acontecerá se a tarefa em execução atual desistir por conta própria

O **agendamento preemptivo** permite que threads de prioridade mais alta interrompam um thread inferior, a fim de cumprir as restrições de tempo real, mas isso gera um custo de mais sobrecarga na troca de contexto.
No **agendamento cooperativo**, a decisão de interromper uma tarefa e permitir que outra comece a ser executada é deixada para as próprias tarefas. Em vez de ser controlado pelo sistema operacional, o agendamento é baseado na cooperação das tarefas. Cada tarefa é responsável por voluntariamente liberar o controle do processador quando apropriado, permitindo que outras tarefas tenham a chance de serem executadas.

### 3.3. Comunicação Entre Threads(ITC)

Normalmente, vários threads precisarão compartilhar informações ou eventos uns com os outros. A maneira mais simples de compartilhar é ler/escrever variáveis globais compartilhadas de forma direta na RAM, mas isto é indesejável devido ao risco de corromper dados provocados por uma condição de corrida. Uma maneira melhor é ler/escrever variáveis estáticas com escopo de arquivo acessíveis por funções setter e getter, e as condições de corrida podem ser evitadas ao desabilitar interrupções ou usar um objeto de exclusão mútua (mutex) interno à função setter/getter. A maneira mais clara é usar objetos RTOS thread-safe como fila de mensagens para passar informações entre threads.

Além do compartilhamento de informações, os objetos RTOS também são capazes de sincronizar a execução de tarefas, porque elas podem ser bloqueadas para aguardar pela disponibilidade de objetos RTOS. A maioria dos RTOS possuem objetos como:

- **Fila de mensagens**
  - Fila do “primeiro a entrar, primeiro a sair” (FIFO) para passar dados;
  - Os dados podem ser enviados por cópia ou por referência (ponteiro);
  - Usado para enviar dados entre tarefas ou entre interrupção e tarefa.
- **Semáforo**
  - Pode ser tratado como um contador de referência para registrar disponibilidade de um recurso particular;
  - Pode ser um semáforo binário ou de contagem;
  - Usado para proteger o uso de recursos ou sincronizar a execução de tarefas.
- **Mutex**
  - Semelhante ao semáforo binário, geralmente usado para proteger o uso de um único recurso (exclusão mútua);
  - O mutex FreeRTOS vem com um mecanismo de herança de prioridade, para evitar o problema de inversão de prioridade (condição quando a tarefa de alta prioridade termina esperando pela tarefa de baixa prioridade).
- **Caixa de correio**
  - Local de armazenamento simples para compartilhar uma variável única;
  - Pode ser considerada com uma fila de elemento único;
- **Grupo de evento**
  - Grupo de condições (disponibilidade de semáforo, fila, sinalizador de evento, etc.)

- A tarefa pode ser bloqueada e pode esperar que uma condição de combinação específica seja atendida;

## 4. Porque usar RTOS

- Organização
- Modularização
- Pilhas de comunicação e drivers

### 4.1. Organização

Os aplicativos sempre podem ser escritos de uma forma “bare-metal”, mas conforme a complexidade do código aumenta, ter algum tipo de estrutura ajudará no gerenciamento de diferentes partes do aplicativo, mantendo-os separados. Além disso, com uma forma estruturada de desenvolvimento e uma linguagem de design familiar, um novo membro da equipe pode entender o código e começar a contribuir com mais rapidez. Padrões de projeto semelhantes nos permitem desenvolver aplicativos em diferentes microcontroladores e até mesmo em um RTOS diferente.

### 4.2. Modularização

Dividir e conquistar. Ao separar recursos em tarefas diferentes, novos recursos podem ser adicionados facilmente sem quebrar outros recursos; desde que o novo recurso não sobrecarregue recursos compartilhados como a CPU e periféricos. Normalmente, o desenvolvimento sem RTOS estará em um grande loop infinito, onde todos os recursos fazem parte do loop. Uma mudança em qualquer recurso dentro do loop terá um impacto em outros recursos, tornando o software difícil de modificar e manter.

### 4.3. Pilhas De Comunicação e Drivers

Muitos drivers extras ou pilhas como TCP/IP, USB, pilhas BLE e bibliotecas de gráficos são desenvolvidos/portados para RTOSs existentes. Um desenvolvedor de aplicativo pode se concentrar em uma camada de aplicativo do software e reduzir, de forma significativa, o tempo de lançamento no mercado.

## 5. Tipos de RTOS

Podemos classificar os sistemas operacionais de tempo real em basicamente dois tipos:

- **Críticos**: os prazos na execução das tarefas são vitais e não podem ser descumpridos. Os sistemas que controlam os meios de transporte, como carros, trens e aviões, e os sinalizadores de controle de tráfego, como semáforos, são exemplos onde o não cumprimento dos prazos podem provocar acidentes fatais, e, por conseguinte, os sistemas embutidos de tempo real deles são críticos.

- **Não-críticos**: o tempo é fundamental, mas uma perda de prazo é tolerável. O Sistema Operacional de um aparelho de DVD ou de um console de vídeo game é não-crítico, pois o não cumprimento de uma tarefa em resposta a um evento em um determinado intervalo de tempo não provoca danos irreversíveis. Ao contrário dos sistemas

críticos, esses sistemas normalmente trabalham com um grande volume de dados.

Aqui estão alguns dos exemplos dos tipos mais comuns de RTOS:

**1. Hard Real-Time OS:** esse tipo de RTOS é projetado para aplicações em que a falha no cumprimento dos prazos é inaceitável. As tarefas críticas têm prazos rígidos que devem ser atendidos, e o sistema operacional garante que essas tarefas sejam executadas dentro desses prazos, mesmo que isso signifique adiar ou interromper tarefas de menor prioridade.

**2. Soft Real-Time OS:** nesse tipo de RTOS, o cumprimento dos prazos é importante, mas existem certas margens de tolerância. O sistema operacional prioriza as tarefas de acordo com suas prioridades, mas pode permitir que algumas tarefas de menor prioridade sejam adiadas ou interrompidas caso necessário, para atender aos requisitos das tarefas mais críticas.

**3. Firm Real-Time OS:** esse tipo de RTOS combina características de sistemas hard e soft real-time. Ele oferece garantias de tempo de execução para a maioria das tarefas, mas pode permitir algumas violações ocasionais dos prazos, desde que a maioria das tarefas seja concluída dentro dos limites de tempo especificados.

**4. Single-Tasking OS:** esses sistemas operacionais de tempo real permitem a execução de apenas uma tarefa em um determinado momento. Eles são adequados para aplicações simples e de tempo real com apenas uma única tarefa crítica.

**5. Multi-Tasking OS:** esses RTOS permitem a execução simultânea de várias tarefas, atribuindo prioridades a cada uma delas. Eles são projetados para lidar com cenários mais complexos, em que múltiplas tarefas precisam ser executadas de forma concorrente, enquanto garantem o cumprimento dos prazos e a previsibilidade do sistema.

É importante ressaltar que esses tipos de RTOS não são mutuamente exclusivos, e muitos sistemas operacionais de tempo real podem apresentar características de mais de um tipo, dependendo das necessidades e requisitos específicos da aplicação em que são utilizados.

## 6. Aplicações de um RTOS

Um RTOS é um sistema simples e leve usado para sistemas limitados ou simples, como dispositivos embarcados. Isso os torna perfeitos para **aplicações** como:

- Controle industrial.
- Comutação de telefone.
- Controle de voo.
- Simulações em tempo real.
- Aplicações militares.
- Aparelhos.
- Dispositivos eletrônicos básicos de consumo.

## 7. Vantagens e Desvantagens

Os sistemas operacionais de tempo real (RTOS) possuem vantagens e desvantagens que devem ser consideradas ao escolher essa abordagem para um determinado sistema. Aqui estão algumas das vantagens e desvantagens dos RTOS:

### 7.1. Vantagens dos RTOS

**1. Comportamento Determinístico:** os RTOS oferecem um comportamento determinístico, ou seja, a execução das tarefas é previsível e repetível, garantindo a capacidade de cumprir prazos rígidos e requisitos de tempo real.

**2. Gerenciamento de Tarefas:** os RTOS fornecem recursos avançados para o gerenciamento de tarefas, como agendamento preemptivo, prioridades de tarefas e mecanismos de sincronização, permitindo uma execução eficiente e otimizada.

**3. Tempo de Resposta Rápido:** os RTOS são projetados para oferecer um tempo de resposta rápido a eventos externos e interrupções, garantindo que tarefas críticas sejam executadas dentro de prazos específicos.

**4. Modularidade:** os RTOS permitem a divisão de um sistema complexo em tarefas menores e independentes, facilitando o desenvolvimento, a manutenção e a depuração do software.

**5. Reusabilidade:** os RTOS são projetados para serem reutilizáveis, o que significa que componentes e funcionalidades específicas de um RTOS podem ser aproveitados em diferentes projetos e sistemas.

### 7.2. Desvantagens dos RTOS

**1. Complexidade:** O desenvolvimento de sistemas com RTOS pode ser complexo devido à necessidade de lidar com prioridades, sincronização, gerenciamento de tarefas e outros aspectos relacionados. Isso pode exigir habilidades avançadas de programação e compreensão detalhada do RTOS em uso.

**2. Overhead:** O uso de um RTOS geralmente implica em algum overhead (custo adicional) em termos de uso de memória, consumo de CPU e outros recursos do sistema. Isso ocorre devido às funcionalidades e serviços fornecidos pelo RTOS para garantir a execução correta das tarefas.

**3. Configuração e Customização:** A configuração e customização de um RTOS para atender às necessidades específicas de um sistema pode exigir tempo e esforço adicionais. Algumas vezes, pode ser necessário ajustar e adaptar o RTOS para melhor atender aos requisitos e restrições do sistema.

**4. Custo:** alguns RTOS comerciais podem ter um custo associado ao uso de licenças e suporte técnico. Isso pode ser um fator a ser considerado, especialmente em projetos com recursos financeiros limitados.

## 8. Conclusão

Os sistemas operacionais de tempo real (RTOS) oferecem benefícios como comportamento determinístico, gerenciamento eficiente de tarefas, tempo de resposta rápido, modularidade e reusabilidade. No entanto, eles também apresentam desafios, como complexidade, overhead de recursos, configuração/customização e possíveis custos associados. A escolha de usar um RTOS depende dos requisitos do sistema. Para aplicações em tempo real com prazos rígidos, os RTOS são altamente recomendados, enquanto sistemas operacionais de propósito geral podem ser preferíveis para sistemas menos críticos ou com recursos limitados.

## Referências


- https://blog.eletrogate.com/rtos-sistema-operacional-de-tempo-real/
- https://materialpublic.imd.ufrn.br/curso/disciplina/2/16/13/8