Universidade Tecnológica Federal do Paraná – Toledo
Engenharia da Computação – COENC

Sistemas Embarcados

Introdução aos sistemas de tempo real

Tiago Piovesan Vendruscolo

Definição

Diferente do que o termo “tempo real” pode sugerir, sistemas de tempo real não significam que a resposta deva ser imediata, no entanto, ela deve possuir previsibilidade e respeitar os requisitos de tempo do projeto.

Sistemas de tempo real

- Sistemas Operacionais de Uso Geral
  - *Executam a tarefa usando o tempo necessário*
- Sistemas Operacionais de Tempo Real
  - *Executam a tarefa no tempo disponível*

Sistemas de tempo real

- Sistemas de tempo real são compostos por eventos que possuem prazo de processamento fixo e deve ser respeitado sempre que possível.
- O conceito da velocidade em tempo real está diretamente ligado à sua aplicação:
  - *Um sistema de freios ABS deve ser capaz de responder sempre na ordem de µs (de acordo com a especificação do projeto).*
  - *No entanto, a ordem para subir ou baixar o vidro de uma janela do carro, pode demorar dezenas ou centenas de ms, sem que isso cause problemas ou incômodo ao usuário.*

Sistemas de tempo real

- Sistemas de tempo real podem ser classificados em dois grupos principais:
  - *Soft real-time*: Sistemas em que o cumprimento das restrições temporais são desejáveis, porém, *não obrigatórias*. O não cumprimento do requisito de tempo não causará danos ao sistema e/ou usuário, porém, afetará a qualidade do serviço, incluindo inconsistências no funcionamento. Ex: Vidro elétrico do carro, sistemas de aquisição de dados, caixa automático, etc.
  - *Hard real-time*: Sistemas onde o cumprimento das restrições de tempo são *obrigatórias*, caso contrário, poderá trazer danos severos ao sistema e/ou usuário. Ex: Sistemas de airbag, freios ABS, controle de estabilidade, sistemas de controle de um míssil, etc.
  - ➢ Geralmente os sistemas em tempo real utilizam uma combinação entre *soft* e *hard real-time*.

Sistemas de tempo real

- Terceira opção (pouco utilizada)
  - *Firm real-time: o não atendimento dos requisitos de tempo são toleradas desde que não seja frequente, no entanto, elas degradam a qualidade do serviço. Os resultados obtidos após o deadline são descartados.*

Sistemas de tempo real – Background / Foreground

- Formas de programação
  - *Super loop – Sistema foreground / background, também conhecido como super laço.*
  - *Máquinas de estados*
  - *Kernel RTOS*

Sistemas de tempo real – Background / Foreground

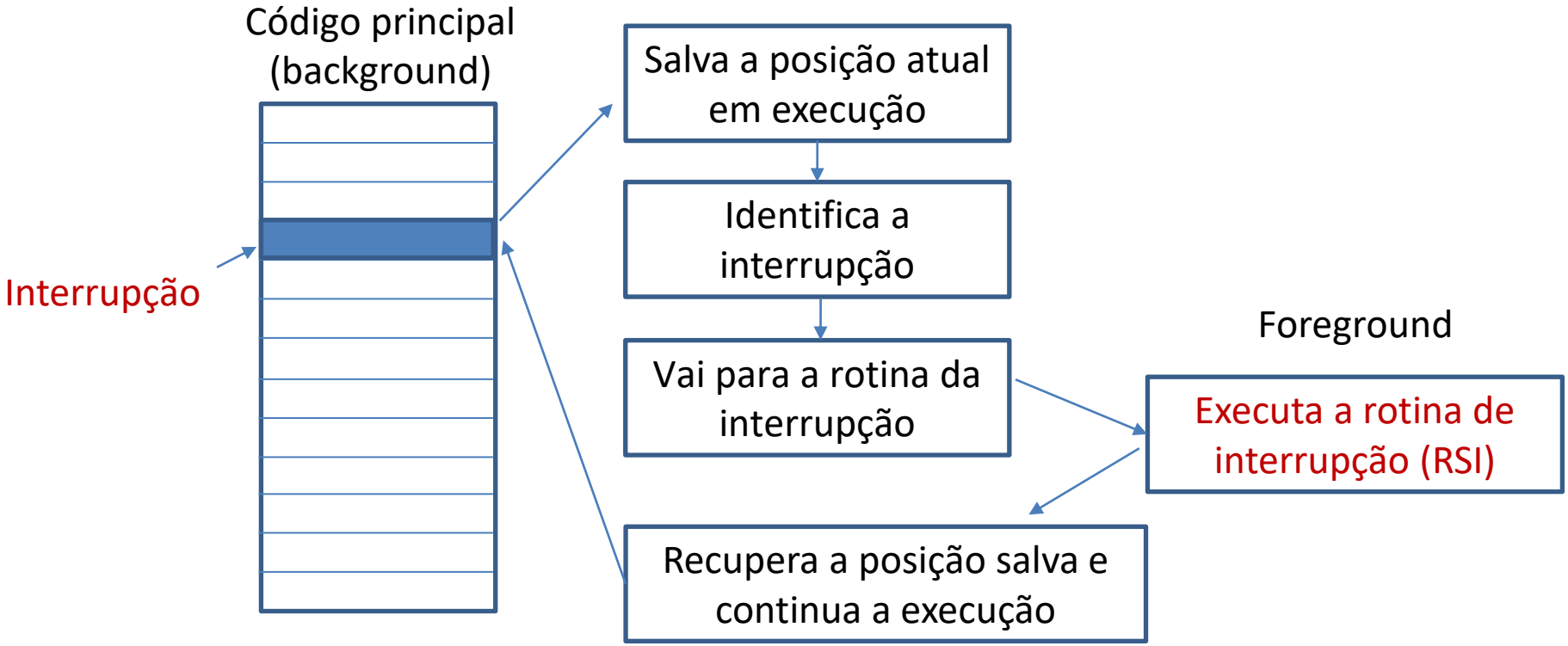
- Diferente da programação convencional, onde o software costuma ser sequencial, em sistemas embarcados são desenvolvidos softwares que respondam a eventos.
  - *A forma mais simples de desenvolver esse tipo de código, é utilizando o sistema foreground / background, também conhecido como super laço.*

➢ *Background* é um laço infinito que executa determinadas tarefas por meio da chamada de funções, sempre que a condição de execução dessas tarefas ocorrer. Também é conhecido como nível de tarefas.

➢ *Foreground* é quando o software "pula" para a execução de rotinas de serviço de interrupção (RSI). Também conhecido como nível de interrupções.

Sistemas de tempo real – Background / Foreground

- Interrupções são utilizadas quando desejamos que um evento externo ao processamento atual tenha prioridade a esse processamento.
- Quando o processador percebe a interrupção, ele salva a posição do processamento atual, executa a rotina de interrupção e então continua o processamento anterior seguindo do ponto que tinha parado.

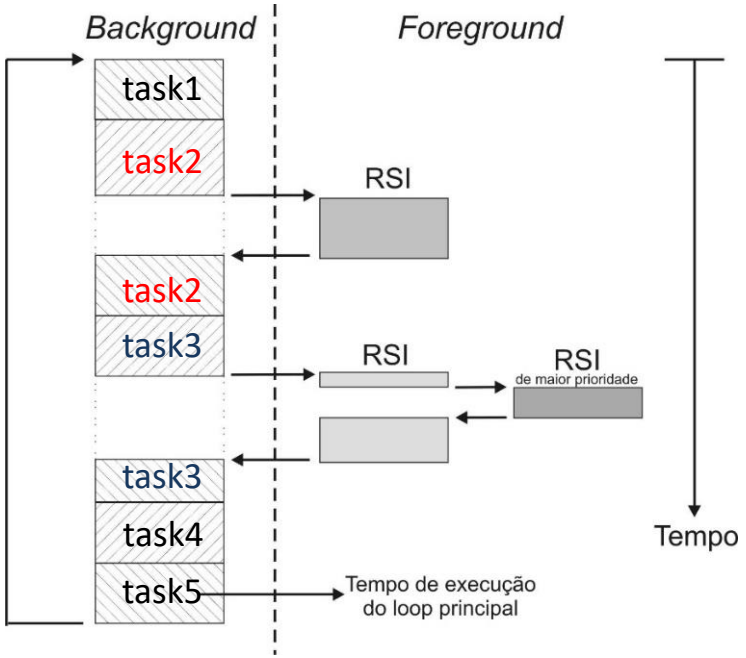


Sistemas de tempo real – Background / Foreground

- Rotinas de serviço de interrupção (RSI) são utilizadas para tratar eventos assíncronos.
  - *Eventos assíncronos são imprevisíveis e podem ocorrer múltiplas vezes. Esses eventos não estão relacionados à execução sequencial do software corrente.*
  - *O tempo necessário para a execução das informações é chamado de "Tempo de Resposta de tarefa".*
- Eventos síncronos são previsíveis e ocorrem na execução sequencial do software corrente.
  - *O excesso ou mau uso de RSI acaba dificultando a previsibilidade dos eventos síncronos.*

Sistemas de tempo real – Background / Foreground

- Eventos RSI são prioritários, e dentre eles, pode-se estabelecer níveis de prioridades.
- Eventos críticos são executados pelas RSIs, para assegurar os requisitos de tempo necessários.
  - *Os resultados obtidos pela RSI podem ser processadas pela própria RSI (mas deve ser evitado) quanto no background.*
  - *Deve-se garantir que o código da RSI de um determinado evento seja concluído antes que ocorra uma nova interrupção do mesmo evento.*



Fonte: Adaptado de (DENARDIN, G. W.; BARRIQUELLO, C. H. 2019)

Sistemas de tempo real – Background / Foreground

- Uma forma de desenvolver sistemas em super laço, é utilizando o agendamento de tarefas, que pode ser realizado a partir do teste de flags.

Sistemas de tempo real – Background / Foreground

- Exemplo genérico de sistema Background / Foreground com flags:

- Escolher o tempo de cozimento no micro-ondas.

```
int main(void)
{
    iniciar_hardware();
    while(1)
    {
      if (display == 1){
         display = 0;
         atualiza_display();
      }

      if (teclado == 1){
         teclado = 0;
         trata_teclado();
         display = 1
      }

    }
}
```

BACKGROUND

```
void TECLA_RSI(1)
{
        limpa_interrupção;
        Le_tecla ();
        teclado = 1;
}
```

FOREGROUND

E se acontece um atraso dentro da interrupção?

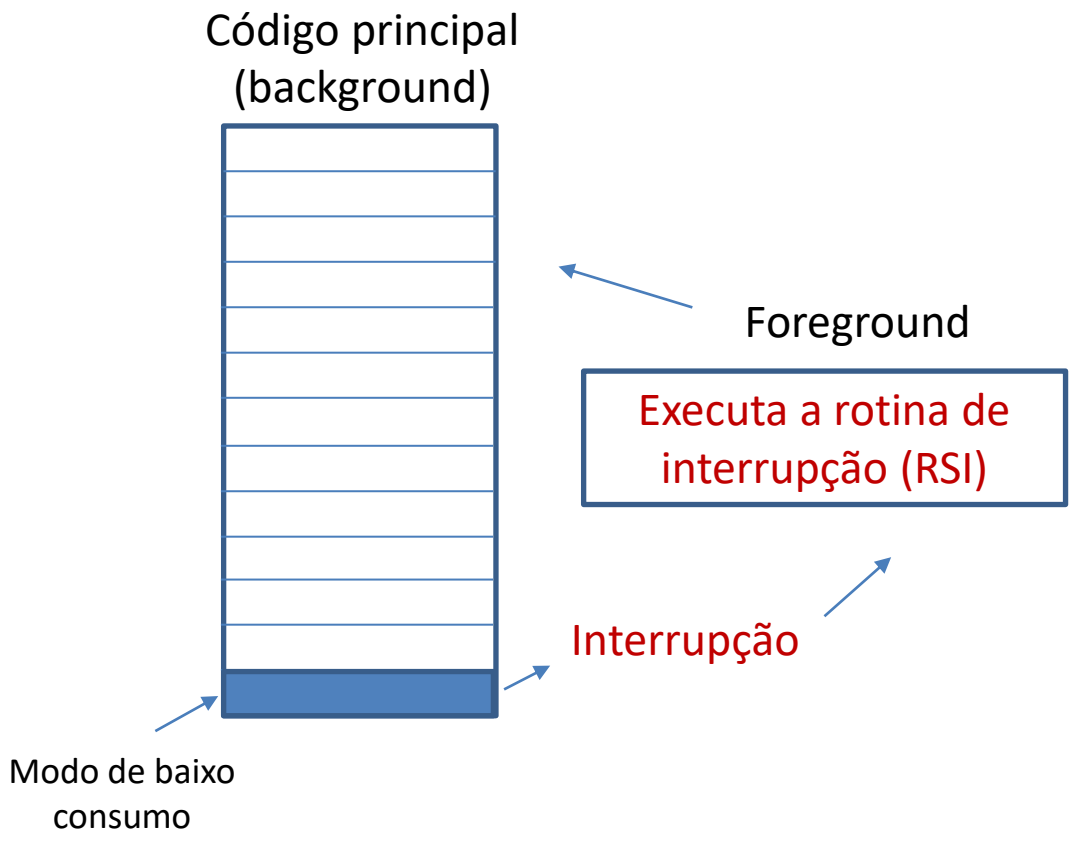
Sistemas de tempo real – Background / Foreground

- Sistemas super laço com sincronia de tempo. Ex: Supomos que precisamos executar 3 funções que são executadas em um tempo desprezível para a contagem de tempo.
  - *Controle_50ms*
  - *Controle_100ms*
  - *Controle_200ms.*
- Nesse caso, não é utilizado interrupções.
- Como fazer se o tempo de execução da função for alto?
- A contagem de tempo pode ser feita por Timer, contagem de clock, etc.

```
int flag = 0;
while(1)
        {
            switch(flag)
                {
                case 0:
                    controle_50ms();
                    controle_100ms();
                    controle_200ms();
                    flag = 1;
                    break;

                case 1:
                    controle_50ms();
                    flag = 2;
                    break;

                case 2:
                    controle_50ms();
                    controle_100ms();
                    flag = 3;
                    break;

                case 3:
                    controle_50ms();
                    flag = 0;
                    break;
                }
        Delay_ms(50);
        }
```

Sistemas de tempo real – Background / Foreground

- Uma forma de otimizar o consumo de energia é colocar o processador em modo de baixo consumo ao final do processamento. Quando uma interrupção acontecer, ele sairá do modo de baixo consumo, executará a RSI e a mesma acionará o background pela chamada de função.




```
while(1)
   {
     if (display == 1){
         display = 0;
         atualiza_display();
     }

     if (teclado == 1){
         teclado = 0;
         trata_teclado();
         display = 1
     }
     if (flags == 0){
         modo_baixo_consumo();
     }
   }
```

Sistemas de tempo real – Background / Foreground

- Vantagens:
  - *Mais rápido e simples de desenvolver.*
  - *Consome poucos recursos do sistema comparado aos sistemas operacionais. Não é necessário conhecimento de API`s de um sistema operacional específico.*
  - *É a solução mais apropriada para projetos pequenos e com restrições de tempo modestas (soft).*

Sistemas de tempo real – Background / Foreground

- Desvantagens:
  - *Não aconselhado para sistemas hard real-time, pois é muito difícil garantir a execução de tempo.*
  - *Todo o código em background tem a mesma prioridade.*
  - *O tempo de execução não é determinístico em sistemas foreground/background.*
  - *Se uma função demorar mais que o esperado, todo o sistema sofrerá atraso.*
  - *Mais difícil adicionar novas funcionalidades.*
    - *Alterações no código afetarão o tempo de execução total do laço.*
  - *É mais difícil coordenar o código quando desenvolvido em grupo.*

- Segundo *Stallings* e *Tanembaum*, existem dois modos distintos de conceituar um SO:
  - *Visão Top-Down: Pela perspectiva do usuário, faz uma abstração do hardware, fazendo o papel de intermediário entre o software (aplicativo) e os componentes físicos que compõe o hardware.*
  - *Visão Bottom-up: é um gerenciador de recursos, controla quais tarefas podem ser executadas, e quais recursos de hardware podem ser utilizados.*

- Um sistema operacional possui as seguintes funções:
  - *Gerenciamento de tempo e recursos de CPU;*
  - *Gerenciamento de processos;*
  - *Gerenciamento de memória;*
  - *Gerenciamento de periféricos;*
  - *Sistemas de arquivos;*
  - *Controle de entrada e saída e dados, protocolos, etc.*

- Exemplos de sistemas embarcados que utilizam sistemas operacionais:
  - *Automóveis;*
  - *Smart TV;*
  - *Roteadores/modems;*
  - *Dispositivos IoT;*
  - *Celulares;*
  - *Etc.*

- Motivações
  - *O sistema foreground / background torna-se limitado com o aumento de complexidade*
  - *Maior facilidade para padronizar e reutilizar os códigos, principalmente sendo desenvolvido em equipe*
  - *Controlar diferentes prioridades entre tarefas*
  - *Facilidade em atualizar/acrescentar funcionalidades, sem prejudicar o sistemas como um todo*
  - *Oferecem uma camada de abstração do hardware*
  - *Etc.*

Sistemas Operacionais de tempo real

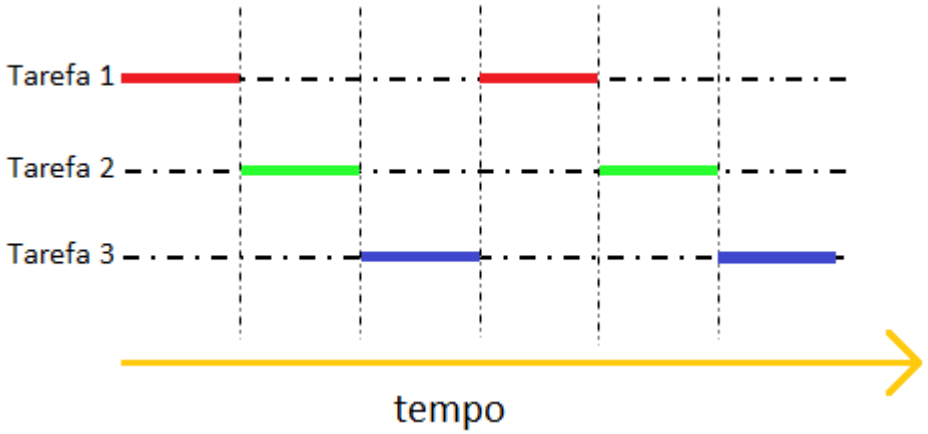
- Normalmente são chamados de RTOS (Real time operating system).
- Possuem recursos para manter o determinismo desejado em sistemas de tempo real.
- O núcleo (kernel) é responsável pelo gerenciamento de: Memória, tempo, tarefas e recursos.
- Fica a cargo de desenvolvedor dividir o sistema em tarefas e definir as prioridades de acordo com os requisitos de tempo real de cada uma delas.

- Multitarefa
  - Em muitos casos, é comum a necessidade de execução de várias tarefas paralelamente.
  - Processadores possuem um número limitado de núcleos.
  - A programação concorrente possibilita que várias atividades possam ser executadas “ao mesmo tempo”.
    - Na realidade, o processador só consegue executar uma atividade ou parte da atividade a cada instante, no entanto, é possível “multiplexar” diversas (pedaços de) tarefas, dando a impressão que exista um processamento paralelo (pseudoparalelismo).

Sistemas Operacionais de tempo real multitarefa

- Multitarefa – Desafios
  - *Múltiplos núcleos de processamento*
    - Sincronização de tarefas: 2 ou mais tarefas que devem manter sincronia.
  - *Um núcleo de processamento*
    - Preempção de tarefas: organizar e decidir qual tarefa será executada.
  - *Compartilhamento de recursos*
    - Processador
    - Periféricos: Evitar que duas tarefas tentem utilizar o mesmo periférico ao mesmo tempo – Conflito.
    - Regiões de memória – variáveis – dados corrompidos.

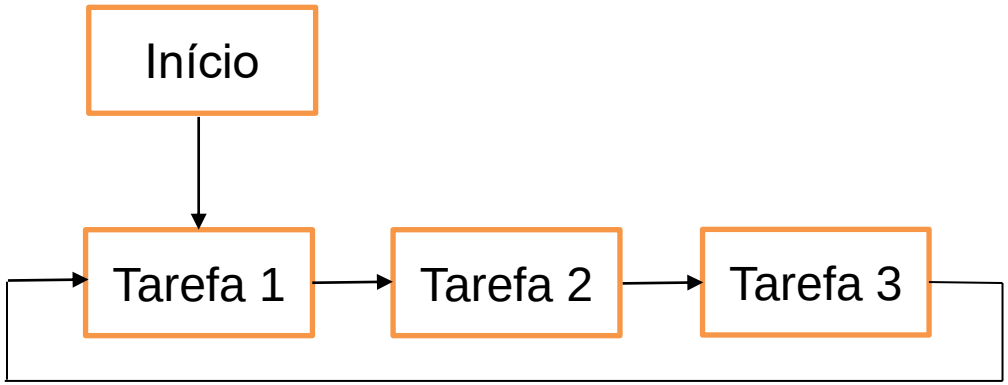
Sistemas Operacionais de tempo real multitarefa



- Uma tarefa executa uma ação através de uma sequência de instruções.
- Pode-se dizer que uma tarefa é um programa simples, que pensa possuir o processador exclusivo para si.
- Sistemas capazes de gerenciar múltiplas tarefas “ao mesmo tempo”, são denominados sistemas multitarefas, e caso façam essa execução respeitando rigorosamente os requisitos de tempo, são denominados sistemas multitarefas de tempo real.
- O processador é multiplexado entre várias tarefas.

Sistemas Operacionais de tempo real multitarefa

- Um sistema embarcado de tempo real multitarefa pode ser desenvolvido de duas formas:
  - Sem RTOS (bare-metal)
    - Sistema utiliza execução cíclica utilizando ou não interrupções.
    - Sistema de controle muito complexo para sistemas grandes.

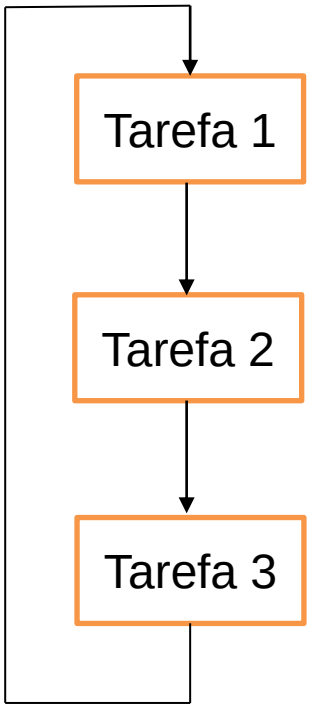



```
int main(void)
{
    iniciar_hardware();
    while(1)
    {
       tarefa1();
       tarefa2();
       tarefa3();
       .
       .
       tarefa_n();
    }
}
```

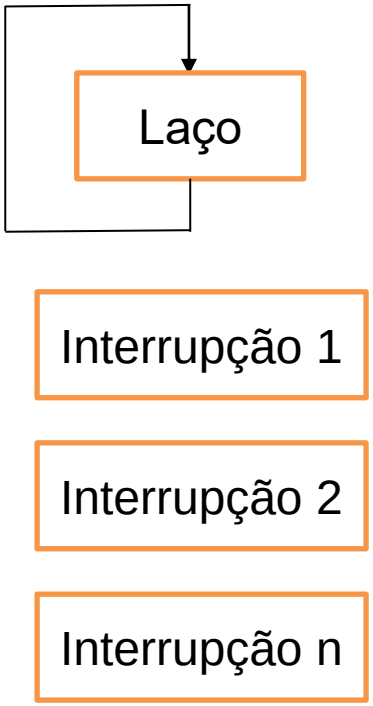
  - Utilizando RTOS
    - Utilização de algoritmos de escalonamento.
    - Há separação entre atividade e controle de atividade (priorização).

Sistemas Operacionais de tempo real multitarefa

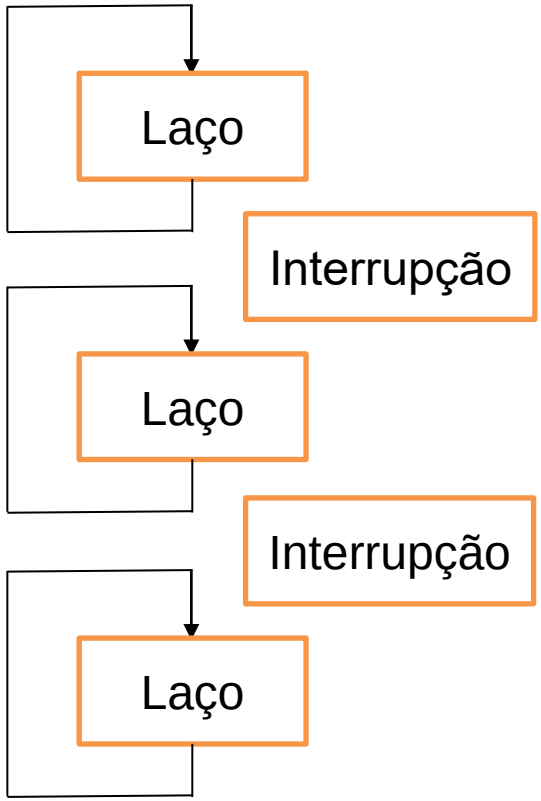
Sequencial



Laço + Interrupções



RTOS (multitarefas)



- Periféricos (timers, A/D...)
- Eventos externos (sensores...)
- Comunicação

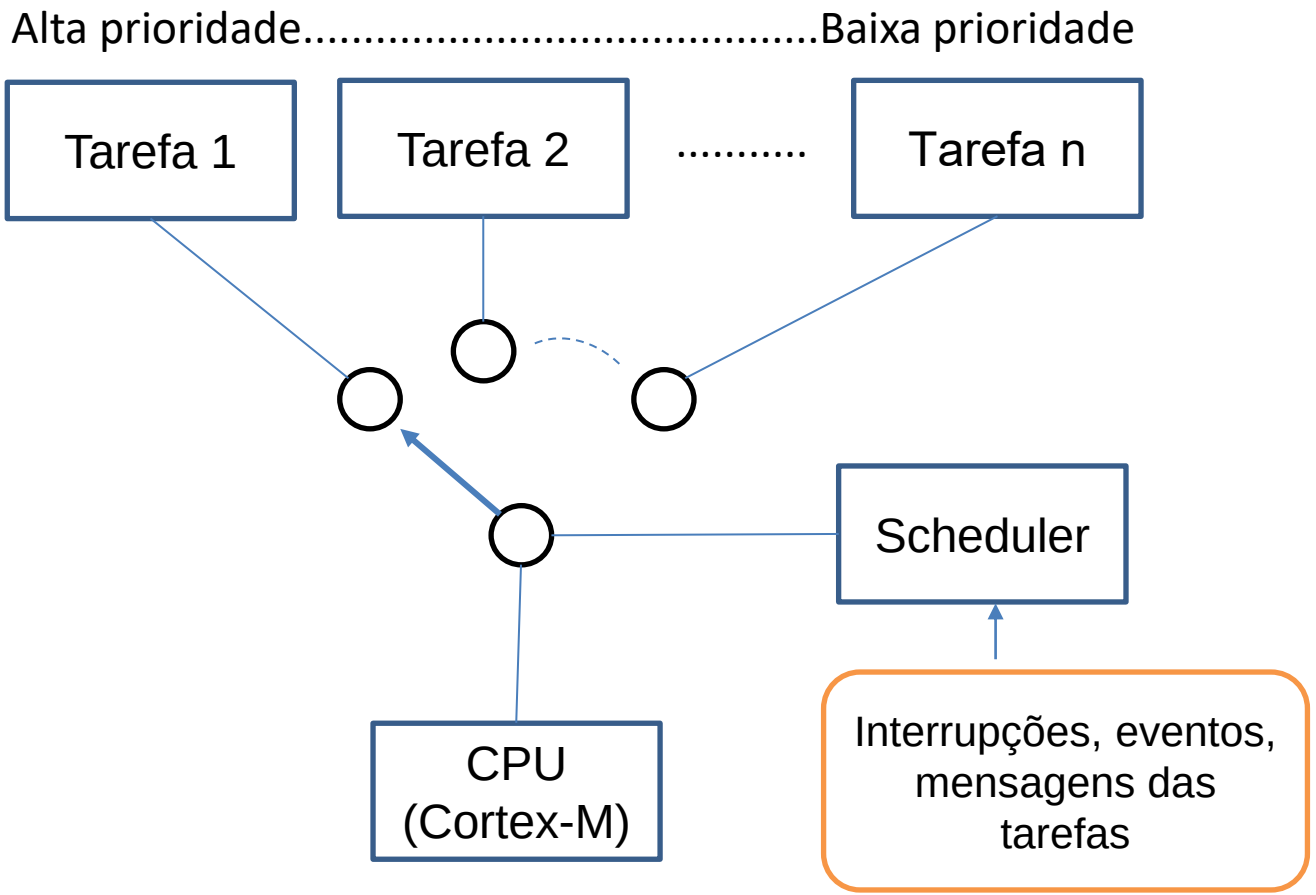
Sistemas Operacionais de tempo real multitarefa

- Diferenças entre RTOS e Sistemas operacionais de propósito geral. Características clássicas de um RTOS:
  - Menos suscetíveis à falha, facilidade de implementação de controle de falhas (watchdog timer).
  - Possui portabilidade entre hardwares com plataformas diferentes.
  - Escalonamento customizáveis.
  - Possui determinismo.

Sistemas Operacionais de tempo real multitarefa

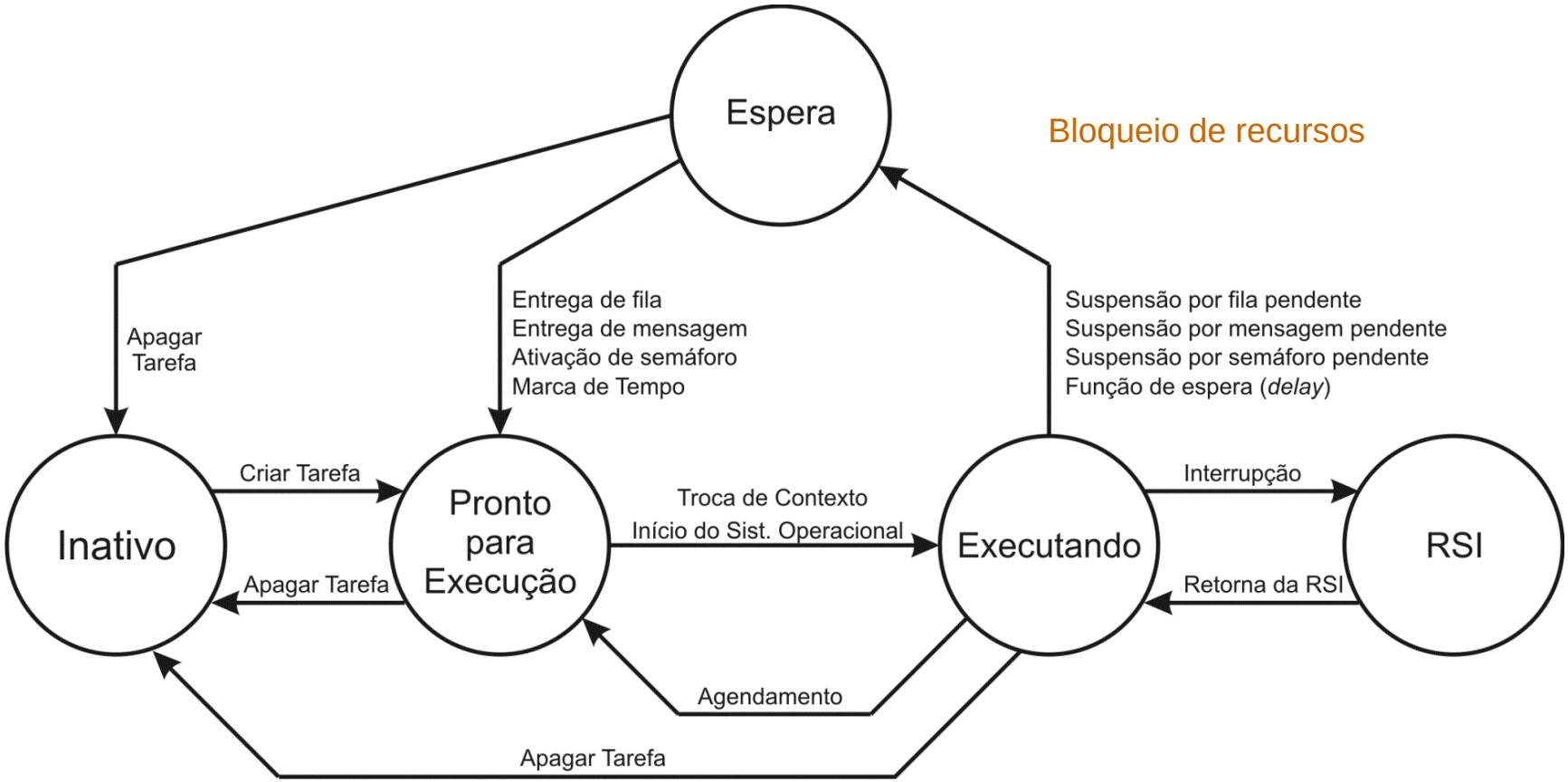
- Normalmente, o hardware de um RTOS deve gerar uma interrupção periódica, conhecida como *marca de tempo*, ou *timer tick*. A partir dessa marca de tempo é realizado o agendamento de tarefas e outras funções básicas do sistema. No entanto, alguns sistemas podem operar utilizando eventos assíncronos, não sendo necessária a marca de tempo.
- Um RTOS normalmente suporta as seguintes funções:
  - Multitarefas
    - Criar, ativar, desativar e deletar tarefas
    - Configurar prioridade de tarefas
    - Agendamento de tarefas
  - Sincronia e comunicação entre tarefas
  - Gerenciamento de memória e de tempo

Sistemas Operacionais de tempo real multitarefa



Sistemas Operacionais de tempo real multitarefa

- As tarefas normalmente se apresentam como laços infinitos que podem estar em um dos cinco estados:



Fonte: (DENARDIN, G. W.; BARRIQUELLO, C. H. 2019)

- Em sistemas operacionais tradicionais os termos mais utilizados são processos e threads, em referência a estruturas de concorrência.
- *Processo: agrupamento de recursos relacionados – memória, threads (uma ou mais), etc.*
- *Threads: são o fluxo de execução do processo, escalonadas pelo sistema operacional. Possuem contexto e pilha (stack). As threads compartilham o espaço de memória ocupado pelo processo, arquivos abertos, etc.*

Contexto: *conjunto de informações usados que deve ser salvo para permitir uma interrupção da thread de forma que ela continue sendo executada a partir do mesmo ponto no futuro.*

- Em um RTOS cada tarefa possui um contexto com pilha própria e as threads são executadas pela prioridade/time slicing possuindo contexto compartilhado entre as corrotinas.
- O sistema operacional controla as tarefas, que utilizam os serviços de sistema para se comunicar.
- Threads e corrotinas são controladas diretamente pelo desenvolvedor.

- *Em RTOS normalmente não se utilizam processos devido à complexidade, e sim, apenas tarefas e threads, sendo que as tarefas assumem o papel das threads.*

- Corrotinas: São similares às tarefas, possuem prioridades, estados e a função é executada quando a corrotina assume o controle do processador.
  - Não possui pilha da tarefa, ou seja, não possui contexto próprio. Dessa forma, a corrotina pode ter um valor armazenado em um registrador quando for interrompida, no entanto, o valor desse registrador pode não ser mais o mesmo quando ela voltar a ser executada.
    - Dessa forma o escalonamento de corrotinas deve ser cooperativo, ou seja, a liberação do processador deve ser feita pela própria corrotina.

- Recursos
  - Os principais recursos que devem ser gerenciados são: processador, memória e os periféricos do sistemas:
    - Dispositivos de E/S, teclado, conversor A/D, comunicação, etc.
  - São chamados de recursos compartilhados quando podem ser utilizados por mais de uma tarefa, no entanto, cada tarefa sempre deve possuir acesso exclusivo (exclusão mútua) ao recurso para evitar conflitos e corrupção de dados.
    - Ex. Acesso simultâneo de um protocolo de comunicação serial, onde os dados poderão entrar em conflito e não resultarem na informação desejada.

- Núcleo (Kernel)
  - Módulo central de todo o sistema operacional. Responsável pelo gerenciamento de todos os recursos disponíveis.
    - Criação e eliminação de tarefas
    - Sincronia e comunicação entre as tarefas
    - Processo de troca de tarefas.
  - O núcleo adiciona *overhead* ao sistema
    - Custo computacional para a troca de contexto, para a troca de tarefas
    - Espaço de código para a implementação de seus serviços
    - Normalmente um SO ocupa entre 1 a 5% da CPU

- Núcleo (Kernel)
  - Os núcleos dos RTOS podem ter gerenciamento preemptivos ou não-preemptivos (cooperativos)

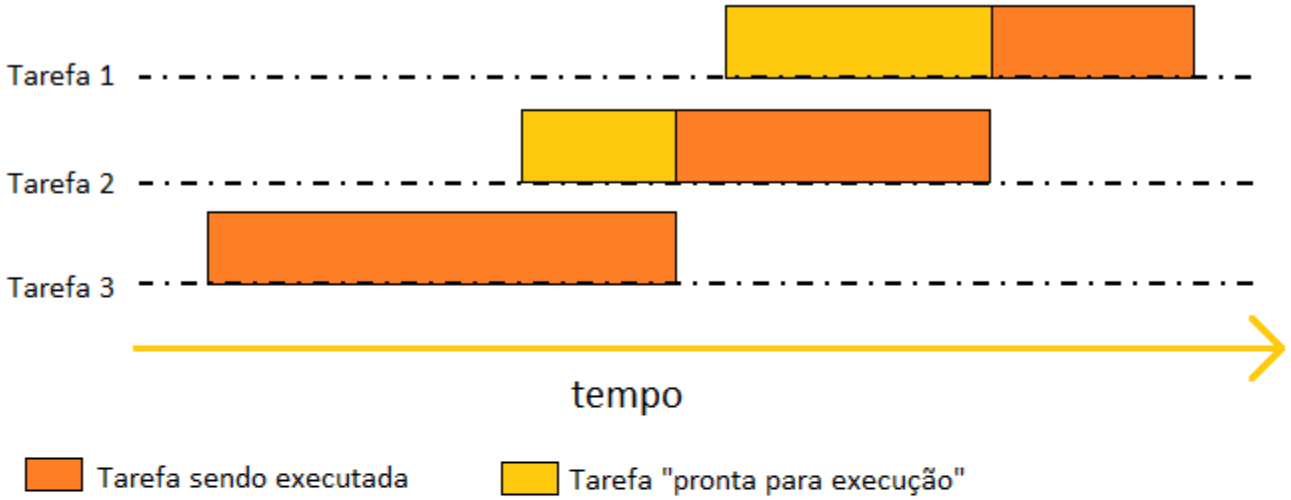
Sistemas Operacionais de tempo real multitarefa

- Núcleo não-preemptivo (cooperativos)
  - *Requer que a tarefa termine e libere o uso da CPU, para então iniciar a execução da próxima tarefa da fila.*
  - *Eventos assíncronos continuam sendo tratados pelas rotinas de interrupção.*
  - *Caso uma interrupção de maior prioridade seja acionada, a rotina de interrupção entra na lista de tarefas prontas para execução.*
  - Menor preocupação com recursos compartilhados.
  - *Permite a utilização de funções não-reentrantes.*
    - *Função não-reentrante é uma função que não pode ser utilizada por mais de uma tarefa, nem permite a possibilidade de perda de dados.*

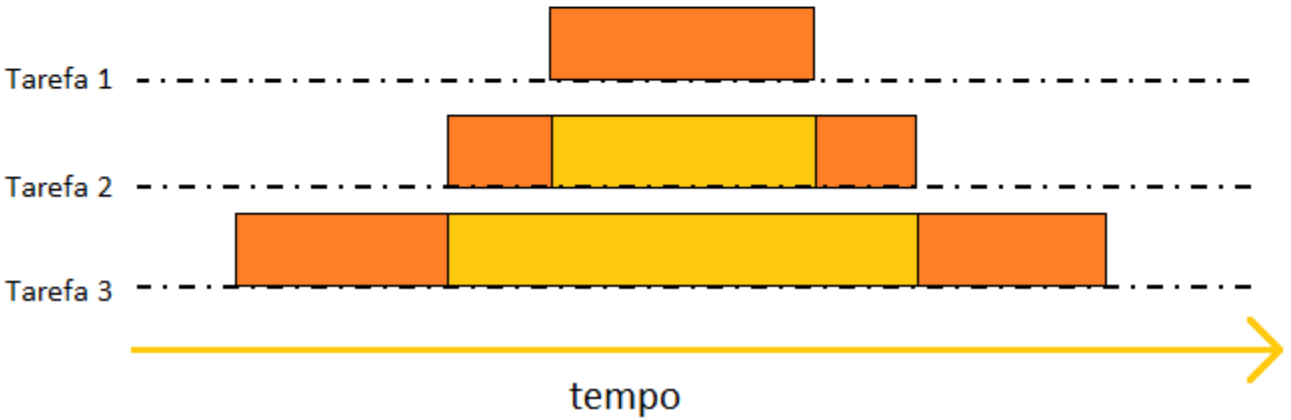
Sistemas Operacionais de tempo real multitarefa

- Prioridades: tarefa 1 > tarefa 2 > tarefa 3

Núcleo não-preemptivo Cooperativo



Núcleo preemptivo



Sistemas Operacionais de tempo real multitarefa

- O núcleo *não-preemptivo é utilizado quando é importante a conclusão das tarefas em andamento. No entanto, é possível que a tarefa desista do controle da CPU antes do término.*
  - *Um tarefa de alta prioridade poderá ter que esperar um longo período para ser executada, porém continua sendo determinístico.*
  - *O tempo de resposta das tarefas não depende de todo o laço, é dado apenas pelo tempo da maior tarefa (apresenta um desempenho melhor que o super-loop).*

Sistemas Operacionais de tempo real multitarefa

- *O* núcleo *preemptivo é utilizado quando a resposta do sistema a um determinado evento é importante.*
  - *Uma tarefa de maior prioridade será executada no menor tempo possível.*
  - *O tempo de execução das tarefas de maior prioridade é determinístico.*
  - *A tarefa também pode ser interrompida se o tempo destinado para a execução tiver expirado.*

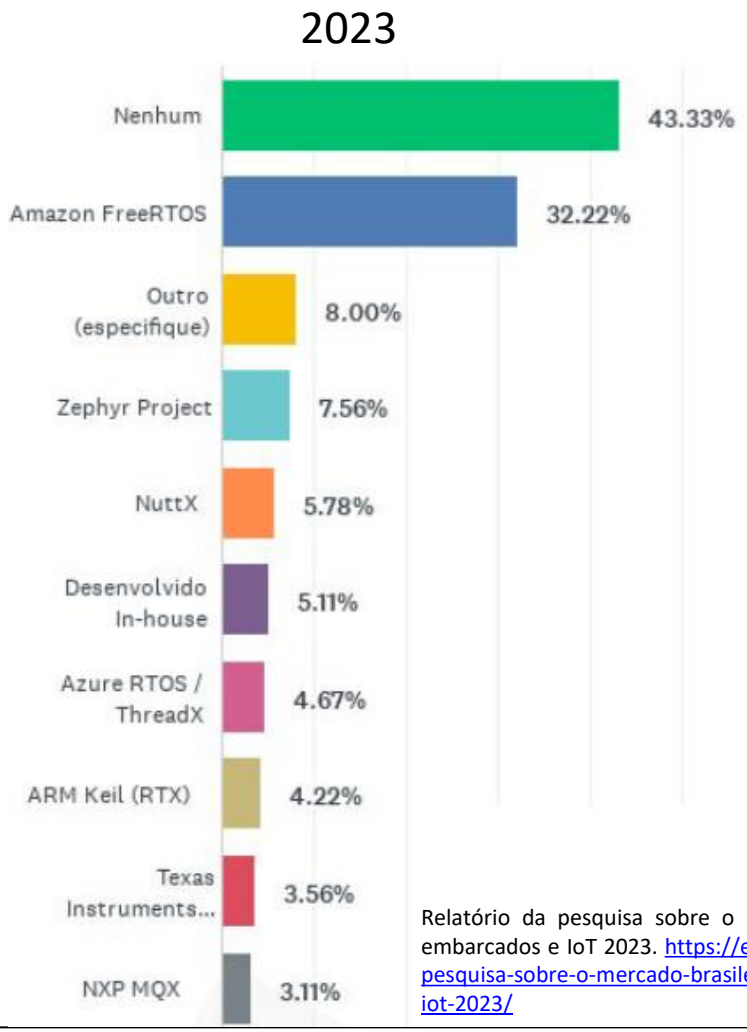
- Estrutura no FreeRTOS
  - *Tarefas: Como criar, deletar, suspender, reiniciar, utilizar múltiplas instâncias e selecionar núcleo de processamento.*
  - *Filas (Queue) com e sem interrupção.*
    - Enviar valores entre tarefas e entre tarefas e interrupções.
  - *Semáforos: Binários, contadores e MUTEX*
    - Proteger/evitar conflito de recursos.
  - *Software Timer*
  - *Event Groups*
  - *Task Notifications*

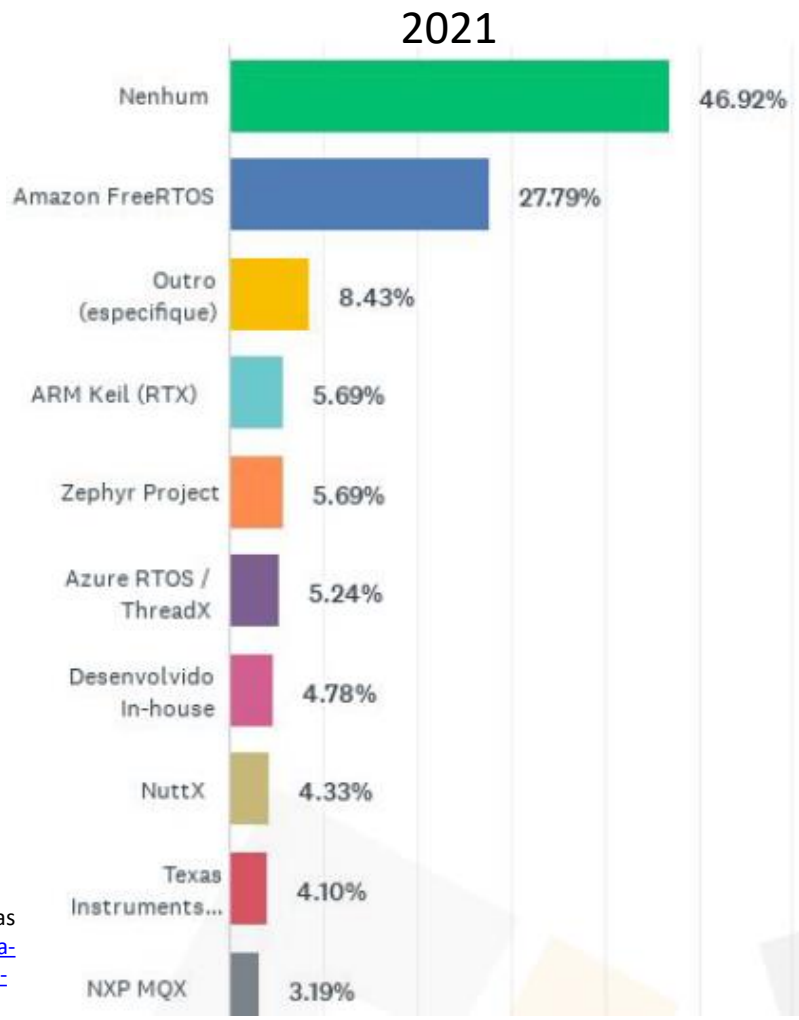
Sistemas Operacionais de tempo real multitarefa

- *Alguns exemplos de RTOS*
  - *Zephyr – Fundação Linux – Gratuito*
  - *µC/OS – Micrium, Inc., Silicon Labs – Atualmente Gratuito.*
  - *mbedOS – Gratuito para ARM*
  - *BRTOS – Brasileiro – Gratuito*
  - *FREERTOS - Gratuito*

Mercado nacional

Por favor, selecione todos os sistemas operacionais de tempo real (RTOS) que você está usando atualmente nos seus sistemas embarcados.





Relatório da pesquisa sobre o mercado brasileiro de sistemas embarcados e IoT 2023. https://embarcados.com.br/relatorio-da-pesquisa-sobre-o-mercado-brasileiro-de-sistemas-embarcados-e-iot-2023/

Sistemas Operacionais de tempo real multitarefa

- *Quais benefícios teremos utilizando um RTOS?*
  - *Sistema pronto e "livre" de falhas*
  - *Modularidade/portabilidade*
  - *Administração do tempo facilitada*
  - *Melhor controle de periféricos*
  - *Desenvolvimento em equipe*
  - *Aproveitamento de tempo ocioso da CPU*
    - *Melhor gerenciamento de energia*
  - *Facilidade no tratamento de interrupções*
  - *Alta capacidade/facilidade na manutenção e implementação de novas funções/recursos no projeto.*

Referências

- DENARDIN, G. W.; BARRIQUELLO, C. H. Sistemas Operacionais de Tempo Real e sua Aplicação em Sistemas Embarcados. 1ª edição, São Paulo, Blucher, 2019.
- DENARDIN, G. W. Notas de aula de sistemas embarcados. UTFPR.
- STALLINGS, William. Operating systems: internals and design principles. 5.ed. Upper Saddle River: Pearson Prentice Hall. 2004.
- TANENBAUM, Andrew. Sistemas operacionais modernos. Rio de Janeiro: LTC. 1999.
- https://sites.google.com/site/profsuzano/sistemas-operacionais
- BACURAU, R.M. Notas de aulas de projeto de sistemas embarcados. UNICAMP, 2020. Disponível em: https://sites.google.com/site/rodrigobacurau/cursos-2020-1/es670---projeto-de-sistemas-embarcados
- OFUCHI, C. Y. Programação Concorrente e CMSIS RTOS, Notas de Aula EL68E Sistemas Embarcados, UFTPR. Disponível em: http://paginapessoal.utfpr.edu.br/ofuchi/sistemas-embarcados-el68e/aula-6-programacao-concorrente-rtos/6_prog_conc_RTOS.pdf/view