Descrição geral de FreeRTOS

Sistemas Embutidos e de Tempo Real

Seminário de 02 de maio de 2023

| André Pereira | Beatriz Aguiar | João Marinho | Tiago Silva |

RTOS

Sistema operativo desenhado para ser utilizado em real-time applications.

Sistema otimizado para casos de utilização específicos fornecendo um ambiente de execução confiável e determinístico.

Escalonamento e execução determinísticos de tarefas, assegurando o cumprimento das restrições temporais de **critical-time tasks**.

Escolha popular para uma vasta gama de aplicações.

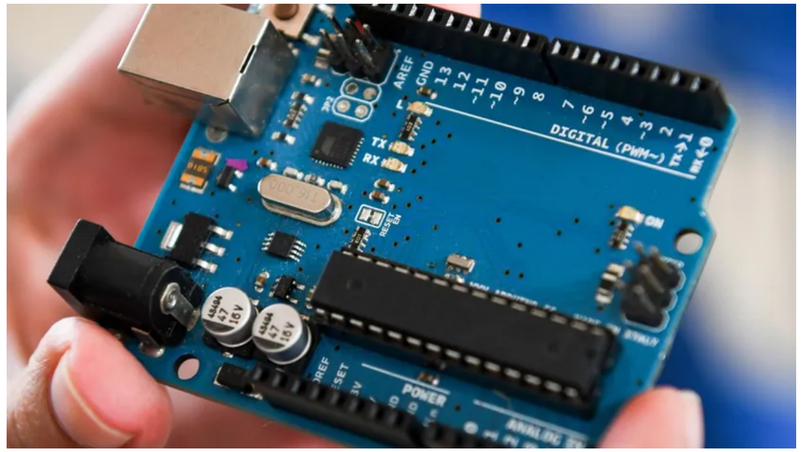
Free RTOS

Open-source real time operating system.

- Maioritariamente desenvolvido em C
- Desenhado para ser pequeno o suficiente para correr em microcontroladores utilizados em sistemas embutidos, que exercem uma determinada tarefa específica e dedicada.

Microcontroladores

Dispositivos pequenos e com recursos limitados.



Estes dispositivos incorporam:

- Um ou mais CPUs
- Memória read-only (ROM ou FLASH)
- Random access memory (RAM)
- Timers, counters, input/output ports...

Devido às restrições de tamanho e à natureza dedicada da aplicação final, raramente possibilitam a implementação completa de um RTOS.

FreeRTOS fornece:

- Escalonamento em tempo real
- Comunicação entre tarefas
- Temporização
- Primitivas de sincronização
- Conjunto extenso de componentes middleware como:
  TCP/IP stack, file systems, USB stack

FreeRTOS possui uma pegada de memória e overheads bastante pequenos.
Kernel configurável.

História

2003

Desenvolvimento

Projeto desenvolvido por Richard Barry como um "side project", com o intuito de possuir um conjunto básico de features e uma fácil utilização.

2003 → 2017

Administração

A empresa Real Time Engineers Ltd acarregou-se posteriormente da administração do produto.

2017

Administração

A administração do projeto foi passada para a AWS.

Atualmente...

Popularidade

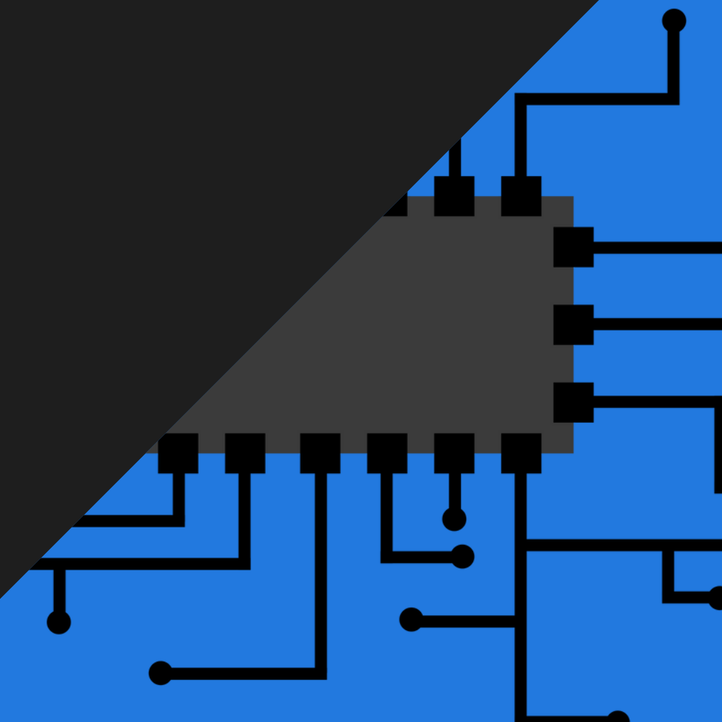
Este projeto continuou a crescer ao longo dos anos, tornando-se um dos principais Real Time Operating Systems (RTOS).

Popularidade do FreeRTOS

- Espaço bastante pequeno
  - Facilmente integrado em dispositivos relativamente pequenos
- Portablidade
- Rico conjunto de features
- Licença MIT
  - Atrativo para evitar vendor lock-in
- Suporte da comunidade:
  - Facilita a obtenção de ajuda, recursos e documentação

  https://github.com/FreeRTOS/FreeRTOS

  https://www.freertos.org/



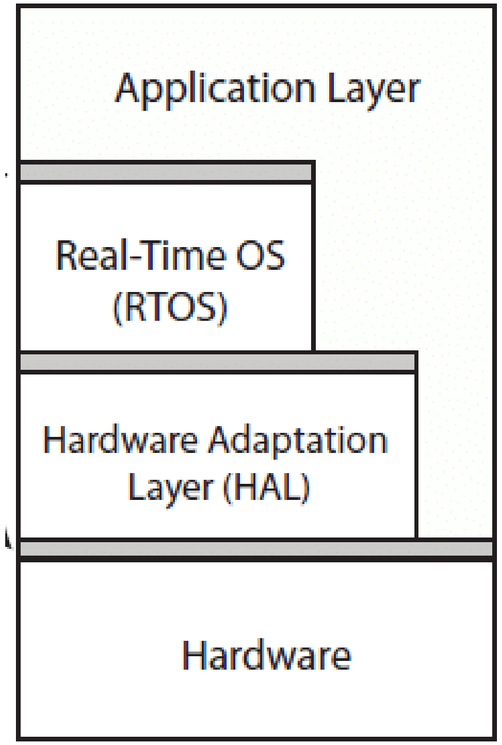
FreeRTOS funcionalidades

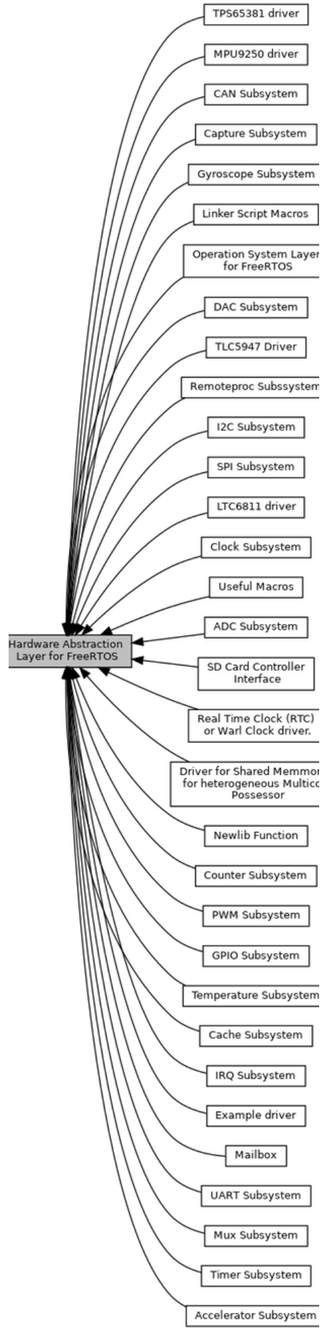
As principais funcionalidades deste sistema operativo são:

- Gestão de tarefas
- Comunicação Inter-Task
- Gestão de memoria
- Software Timer
- Interrupções

FreeRTOS Kernel

- Arquitetura núcleo mínimo
- Camada de abstração de hardware (HAL) configura e gere:
  - Interrupções
  - Acessos a periféricos
  - ...





Gestão de Tarefas

Gestão de Tarefas

O que são?

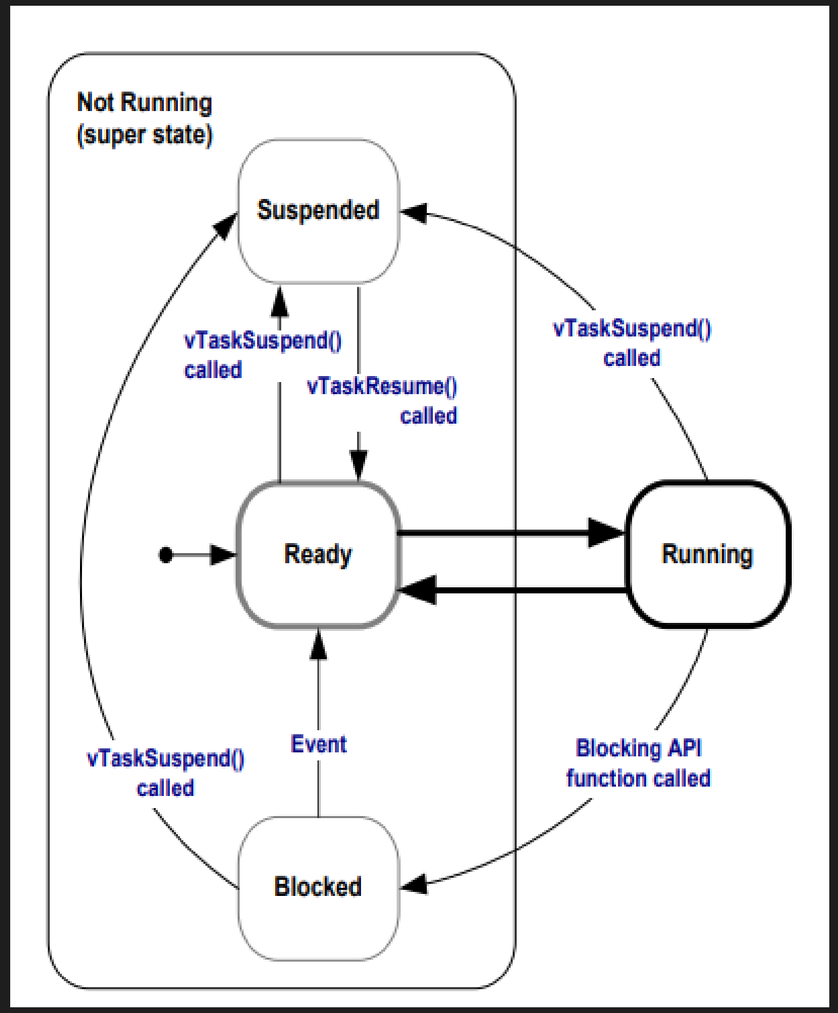
- Entidades representadas por funções em C
- Têm a sua própria stack
- Suportam preempção
- Possuem uma prioridade

Como são criadas?

```
BaseType_t xTaskCreate( TaskFunction_t pvTaskCode,
                        const char * const pcName,
                        uint16_t usStackDepth,
                        void *pvParameters,
                        UBaseType_t uxPriority,
                        TaskHandle_t *pxCreatedTask );
```

Gestão de Tarefas

Possíveis estados



Gestão de Tarefas

Prioridades (configMAX_PRIORITIES)

Método Genérico

Qualquer valor de prioridade
Maior consumo de RAM
Maior WCET

Método Otimizado

Prioridade inferior a 32
Menor consumo de RAM
Não afeta o WCET

Escalonamento

Single-core

Política preemptiva de prioridade fixa com base em round-robin time-slicing.

AMP

Mesmo algoritmo de escalonamento. Recorre a streams ou message buffers para troca de dados entre cores.

SMP

Mesma política, mas diferentes garantias. Uma tarefa de menor prioridade poderá correr ao mesmo tempo que outra de maior prioridade.

Comunicação Inter-Task

Mecanismos:

- RTOS Task Notifications
- Stream and Message Buffers
- Queues
- Semaphores (Binary or Counting)
- Mutexes, and Recursive Mutexes

Comunicação Inter-Task

Queues

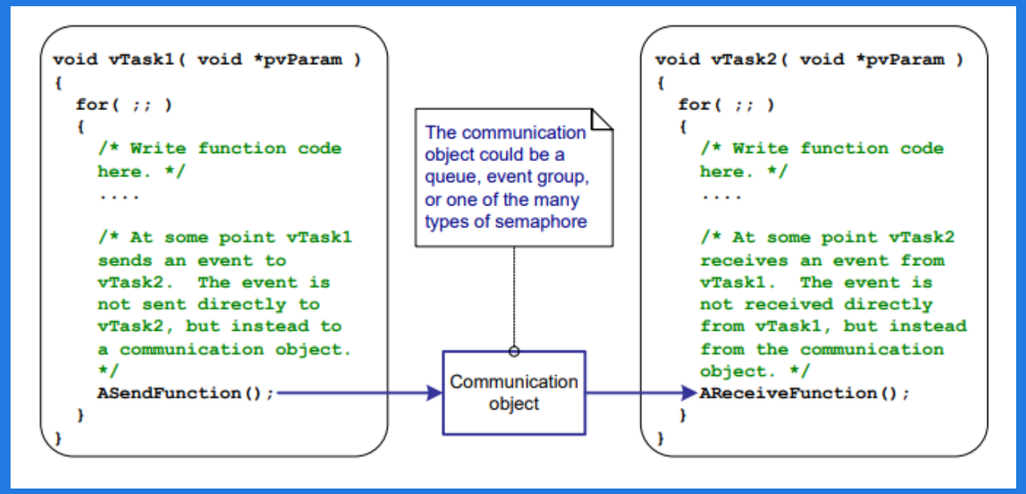
Simplicidade VS Flexibilidade

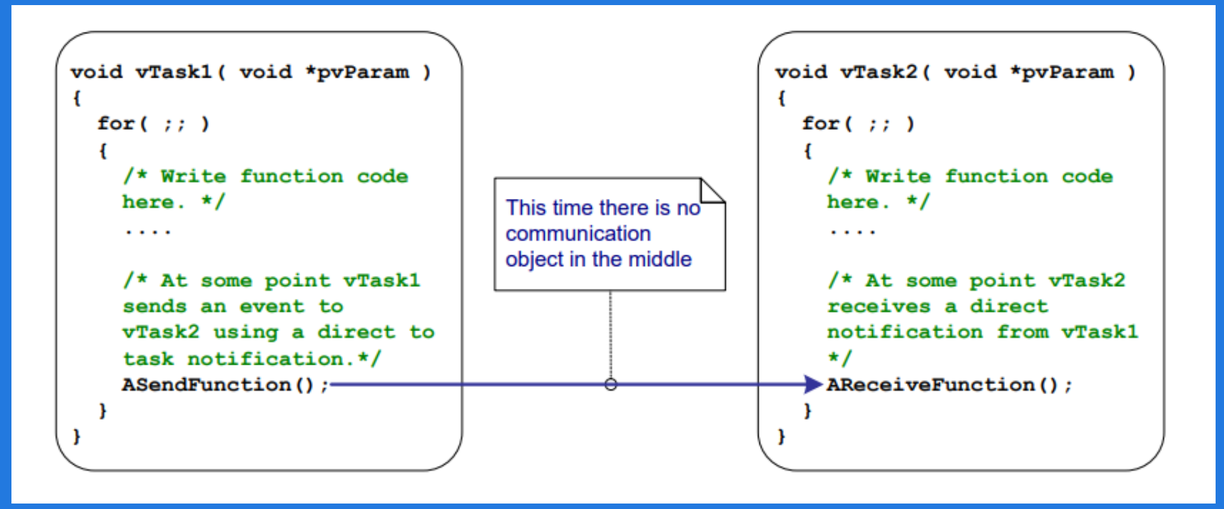
Mensagens enviadas por cópia	Cópia de apontadores
Total dissociação entre a tarefa que envia e a que recebe	Mensagens de tamanhos variáveis
RTOS kernel é reponsável pela alocação de memória da queue	Mensagens de diferentes tipos e localizações

Principais funções: xQueueCreate(), xQueueSendToBack(), xQueueSendToFront() e xQueueReceive().

Comunicação Inter-Task

Task Notifications





Comunicação Inter-Task

Task Notifications

Benefícios ✓

- Mais rápidas que o uso de queues, semáforos, ..., para as mesmas operações
- Menor uso de RAM

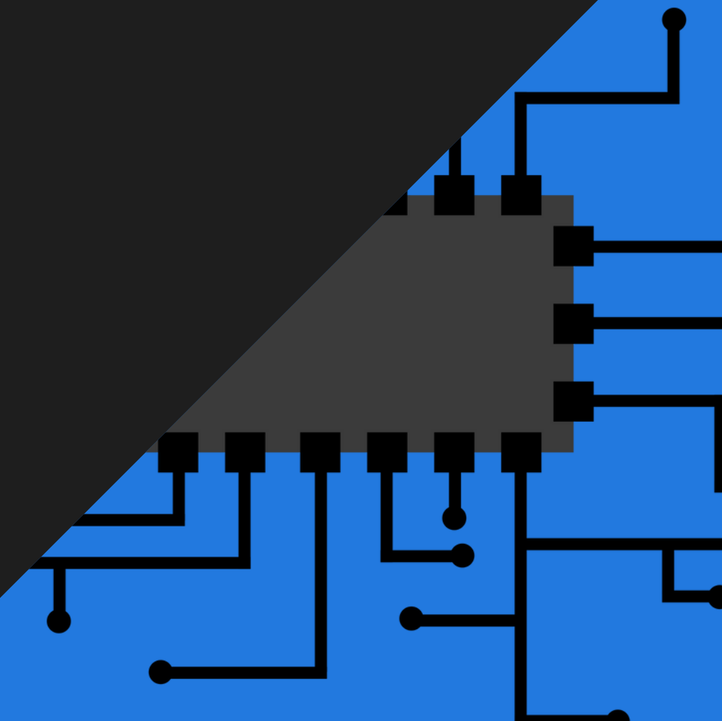
Limitações ✗

- Utilização não tão diversificada:
  - Não permite o envio de dados para um ISR
  - Não permite comunicação com múltiplos recetores
  - Não possui buffering
  - Quem envia não pode bloquear

Principais funções: xTaskNotifyIndexed(), xTaskNotifyWaitIndexed(), ou xTaskNotifyGiveIndexed() e ulTaskNotifyTakeIndexed().

Gestão de memória

- O kernel RTOS necessita de RAM para a criação de uma task, queue, mutex, software timer, semáforo ou eventos de grupo
- Este processo pode utilizar as funções malloc, free, and realloc.
  - Não estão sempre disponíveis em sistemas embebidos
  - Ocupam grande parte do código
  - Não são thread-safe
  - Não são deterministas
  - Podem causar fragmentação
- Em versões mais antigas, o freeRTOS fazia uso de um "memory pool allocation scheme"
- Atualmente é fornecida uma alternativa através de uma API de alocação de memória.



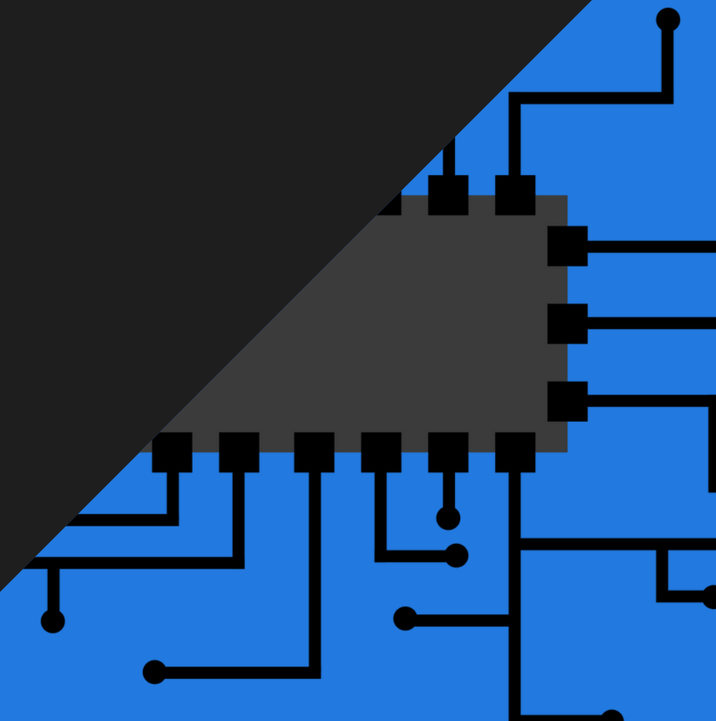
Gestão de memória

freeRTOS

- A alocação de memória faz parte de uma "portable layer"
  - um algoritmo de alocação de memória é adequado apenas para um subconjunto de aplicações

malloc → pvPortMalloc()
free → pvPortFree()

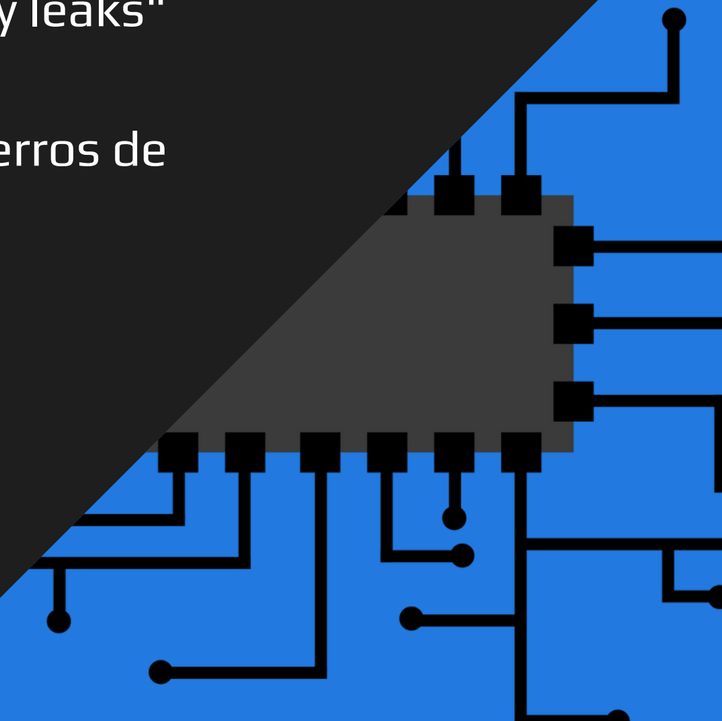
- FreeRTOS tem 5 implementações diferentes do algoritmo de alocação de memória
  - A aplicação freeRTOS pode fazer uso de qualquer implementação
  - Todas as implementações fornecidas fazem uso da heap
- Permite à própria aplicação fornecer uma implementação do algoritmo



Gestão de memória

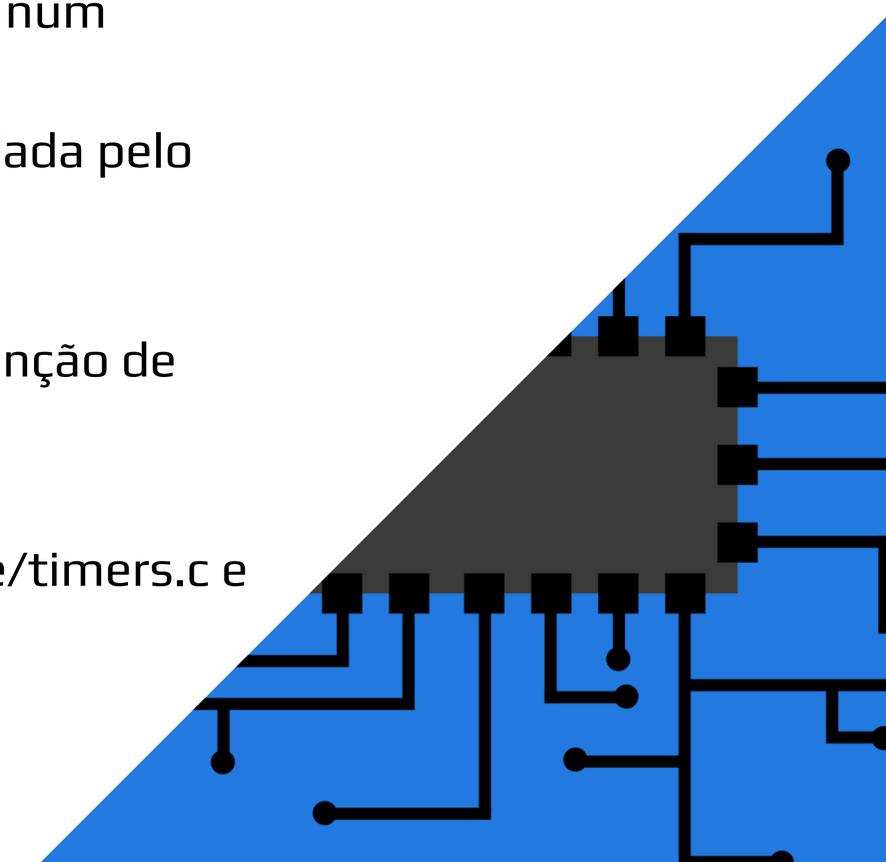
Benefícios RTOS

- Uso eficiente de memória
  - as funções desenhadas para serem efecientes e minimizarem a fragmentação de memória, reduzindo a probabilidade de "memory leaks"
- Comportamento previsível
  - as funções fornecem um comportamento previsível e lidam com erros de forma cuidadosa, reduzindo a probabilidade de crashes e comportamentos bizantinos
- Customizável
  - permite costumizar a estrátegia de alocação de acordo com as necessidades da aplicação.
- Reduz o risco de erros
- Portável



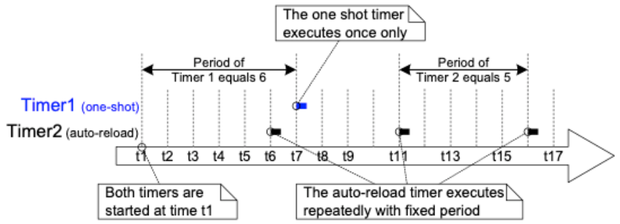
Software timer
FreeRTOS

- Timers são utilizados para agendar a execução de uma função num determinado instante ou periodicamente com uma frequência
- O mecanismo de timer do FreeRTOS é implementada e controlada pelo próprio kernel
    - Independente do hardware
- O timer não utiliza tempo de processamento a menos que a função de callback esteja a ser executada
- É opcional nas aplicações freeRTOS
    - Para o incluir é necessario criar o ficheiro FreeRTOS/Source/timers.c e ativar a variável configUSE_TIMERS
    - xTimerCreate() é a função da API freeRTOS



Software timer
One shot vs auto reload

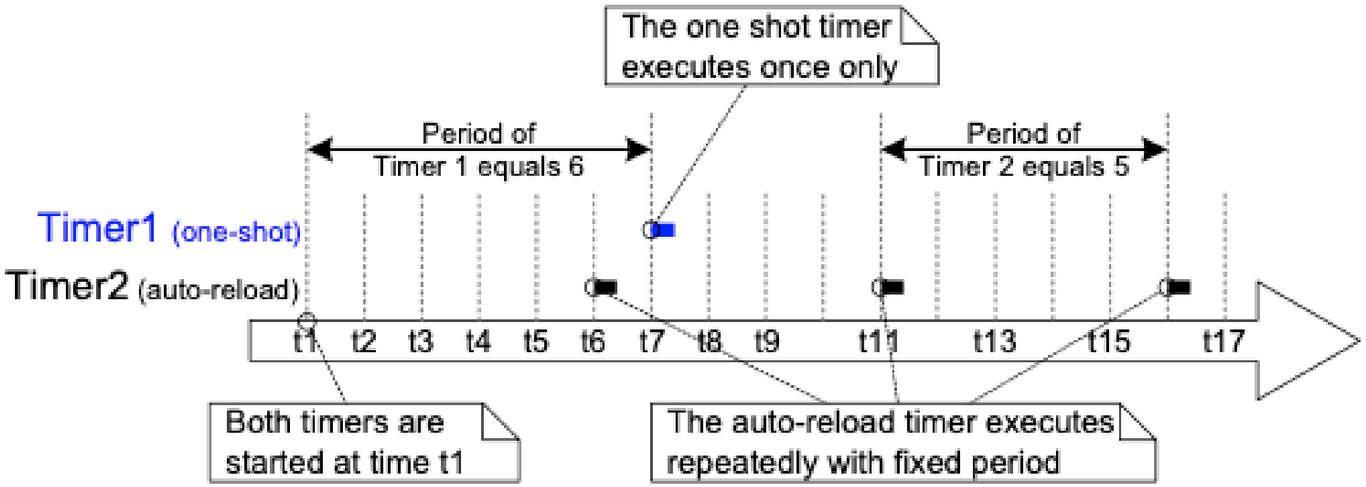
- Existem dois tipos de timers no freeRTOS
- **One shot timers**
  - Após inicializados, a função de callback é apenas executada uma vez
  - Podem ser reiniciados manualmente
- **Auto reload timers**
  - Reinicia-se automaticamente cada vez que expira
  - Resulta numa execução periódica da função de callback



Software timer

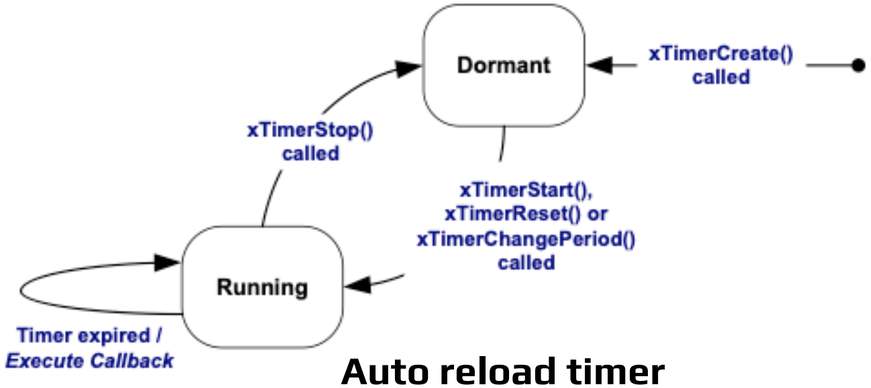
One shot vs auto reload

- Existe
- **One s**
  - Ap
  - P
- **Auto**
  - Re
  - Re

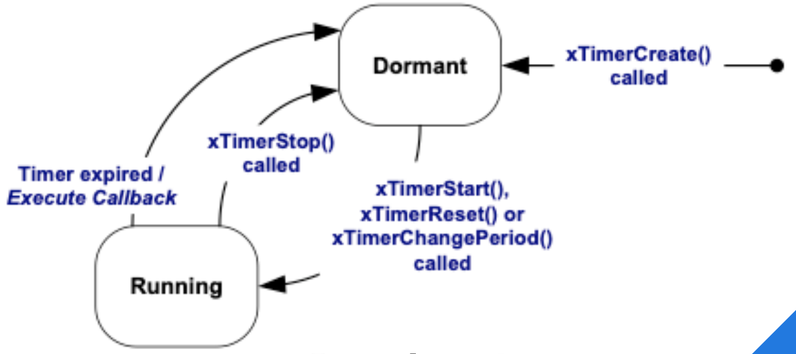


Software timer
Estados

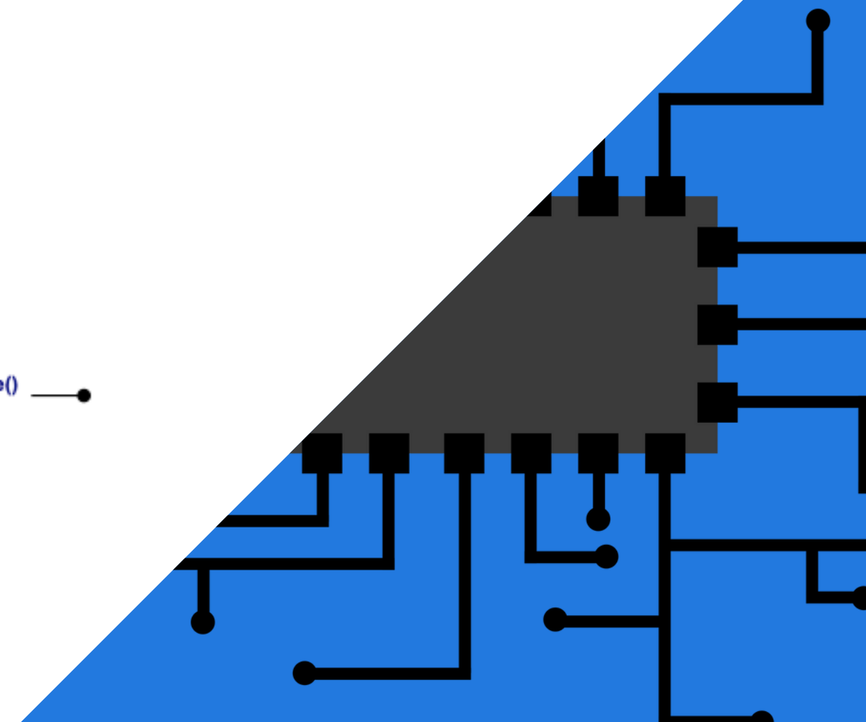
- Um timer pode estar em dois estados diferentes
- **Estado Dormant**
  - O timer existe e pode ser referenciado
  - A função de callback não é executada
- **Estado Running**
  - A função de callback é executada



Auto reload timer

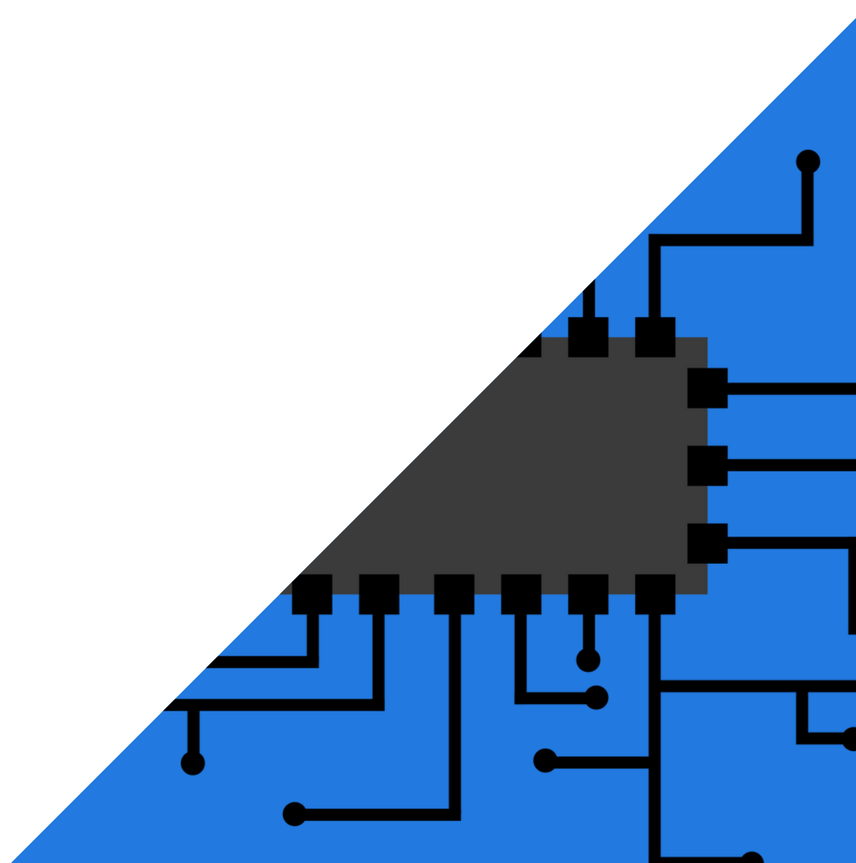


One shot timer



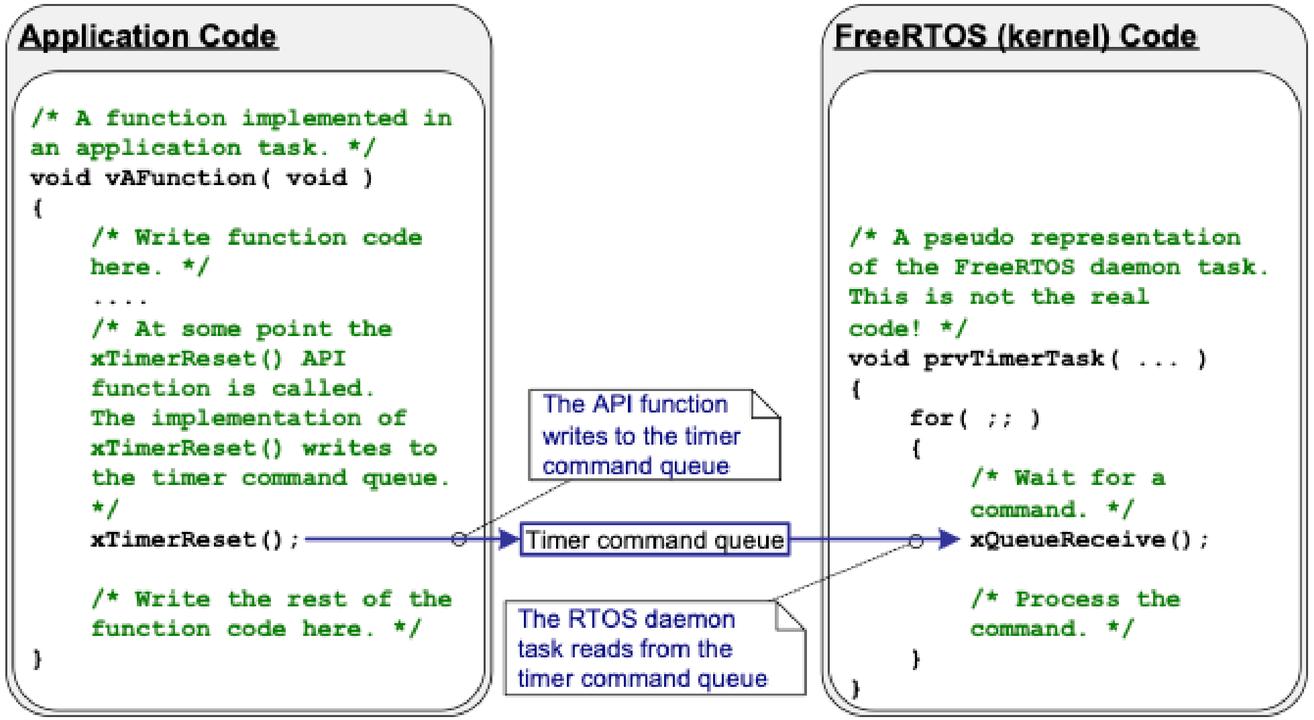
Software timer
Comunicação

- Todas as funções callback dos timer executam no contexto de uma tarefa RTOS daemon
  - tarefa criada automaticamente quando o scheduler é iniciado
  - prioridade e tamanho na stack da tarefa são constantes configuradas no ficheiro FreeRTOSConfig.h
- As funções da API do timer mandam os comandos à tarefa daemon através da 'timer command queue'



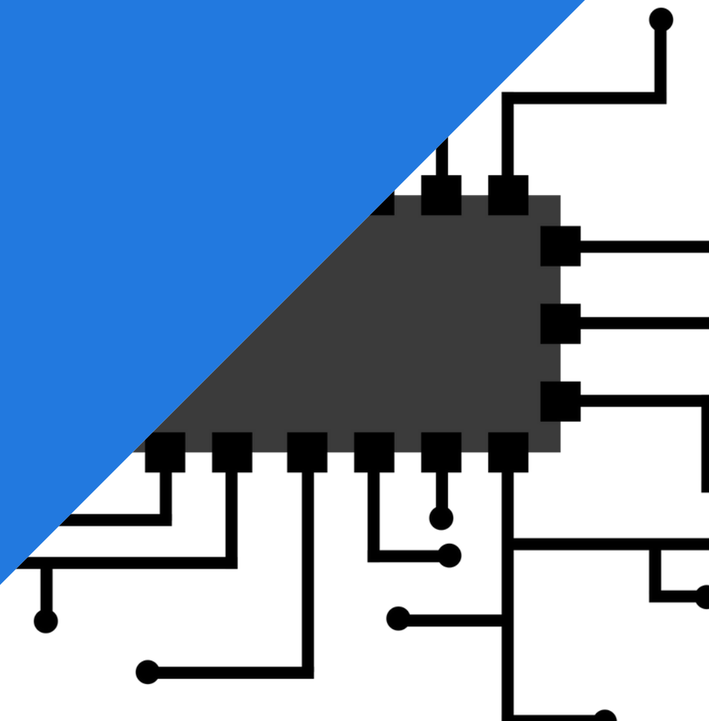
Software timer

Comunicação



Interrupções
ISRs

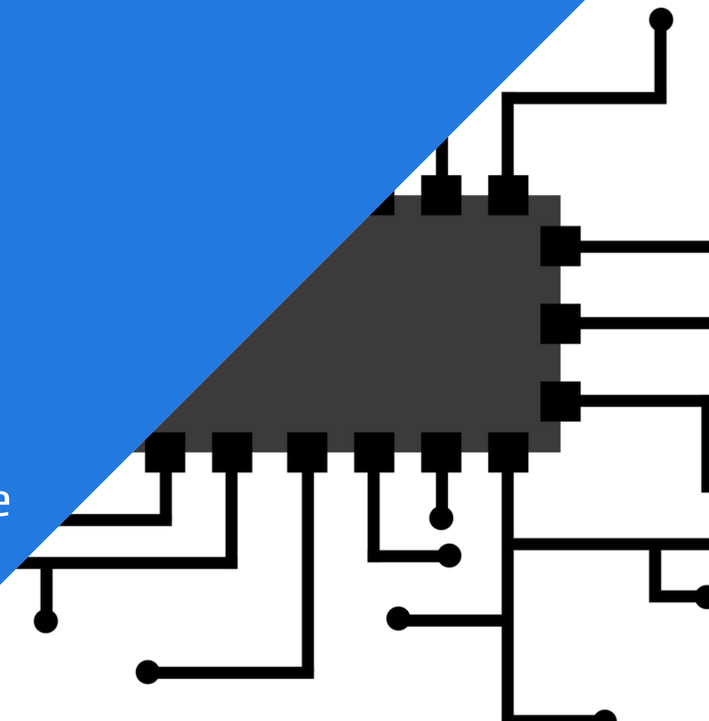
- ISRs (Interrupt Service Routines) são funções que são executadas em resposta a uma interrupção. Quando a interrupção ocorre, o cpu bloqueia a task atual e salta para a ISR para lidar com a interrupção
- Muitas das funções da API do freeRTOS realizam ações que não são válidas dentro de uma ISR
    - por ex. colocar a tarefa que chamou a API no estado blocked
- FreeRTOS fornece duas versões das mesmas funções, uma para tarefas e outra para ISRs
    - maior eficiência
    - impossibilita a integração de código externo que utilize ambas as versões das funções



Interrupções

Defer interrupting

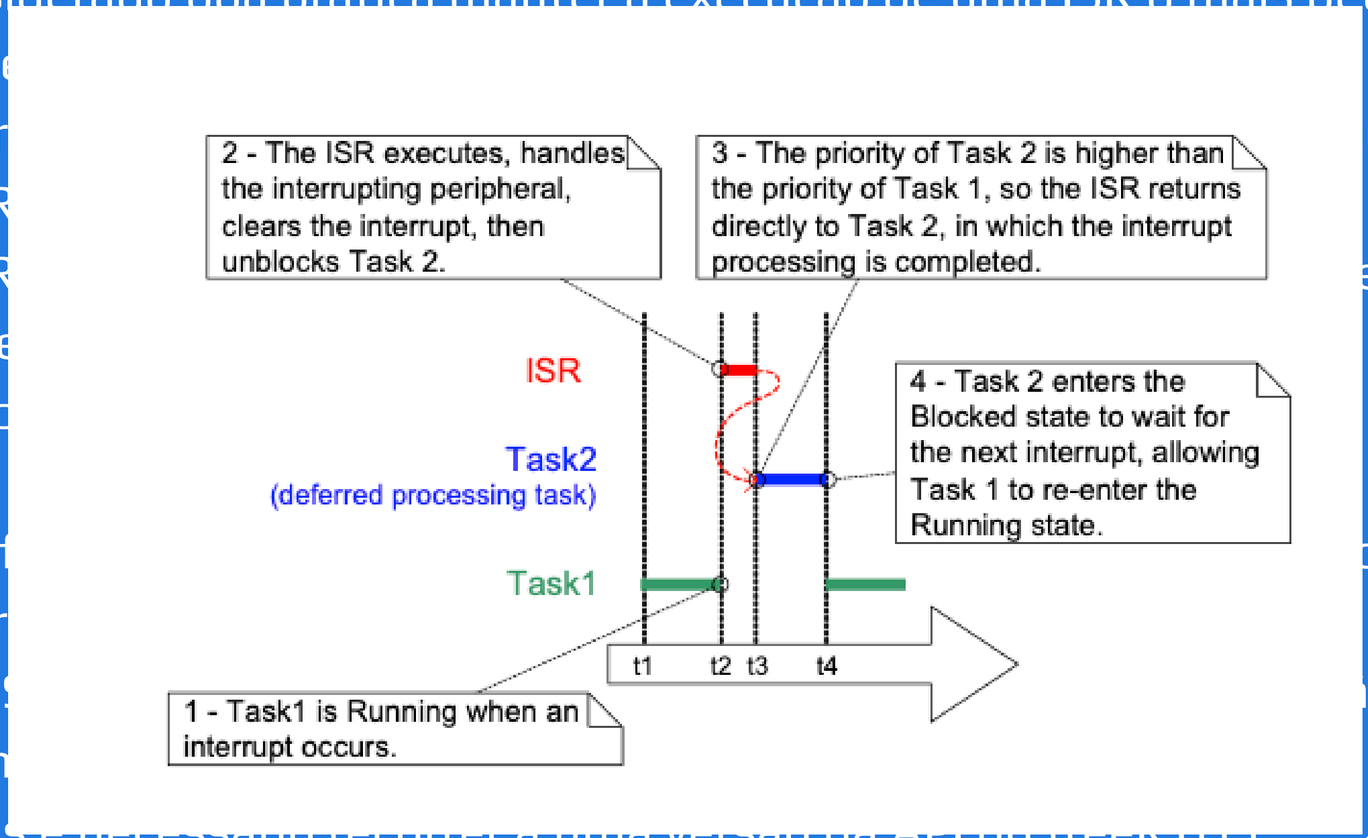
- É considerado boa prática manter a execução de uma ISR o mais pequena possível
  - tarefas de alta prioridade só são executadas se não existir nenhuma ISR a ser executada
  - ISRs podem causar interferência com o tempo inicial e tempo de execução de uma tarefa
  - Problemas de acesso a recursos mútuos
  - ...
- Em defer interrupting, a execução do processamento da interrupção é 'deferred' para uma tarefa
  - A ISR fica apenas responsável por registar a causa da interrupção e limpar a interrupção
- Apenas é necessário recorrer a uma versão da API do freeRTOS



Interrupções

Defer interrupting

- É considerado boa prática manter a execução de uma ISR o mais pequena possíve
  - tar uma ISR
  - ISR e exe
  - Pro
  - ...
- Em def o é 'deferr
  - A I o e lim
- Apenas é necessário recorrer à uma versão da API do FreeRTOS



Versões FreeRTOS

FreeRTOS 202012 LTS end of support

by Tanmoy Sen on 31 Mar 2023

It has been six months since we announced the second release of FreeRTOS Long Term Support (LTS) - FreeRTOS 202210 LTS. Today the support period ends for the previous LTS release, FreeRTOS 202012 LTS. You can migrate your project to FreeRTOS 202210.01 LTS if you are using the previous version. See the migration guide and […]

Read more…

Derivações FreeRTOS

FreeRTOS

FreeRTOS-Plus

Amazon FreeRtos

SafeRTOS

OpenRTOS

Derivações FreeRTOS

FreeRTOS-PLUS

- FreeRTOS-Plus-TCP
  - Internet stack TCP/IP escalável, codigo aberto e seguro.
  - Implementação baseada em uma interface de sockets Berkeley ou interface de callbacks.
  - Aplicável a MCU pequenos com baixa taxa de transferência e MCU maiores com alta taxa de transferência.
- FreeRTOS-Plus-CLI
  - Permite o processamento de linhas de comando.
    i. Criar função que implementa o comportamento do comando
    ii. Mapear comandos para a funções
    iii. Registar o comando no interpretador de comandos

Derivações FreeRTOS

Amazon FreeRTOS

- FreeRTOS com bibliotecas de suporte para Internet das coisas (IoT).
- Concebido para funcionar com os serviços AWS
  - AWS IoT Core
  - AWS Greengrass
  - AWS IoT Device Defender

Derivações FreeRTOS

SafeRTOS

- Implementação FreeRTOS para sistemas críticos.
- Desemvolvido exclusivamente em C.
- Quando implementado em RAM só pode ser utilizado na sua configuração original e certificada, sendo que alguns MCU já o incluem na memoria ROM.
- Certificação
  - Industrial - IEC 61508 SIL 3
  - Médica - IEC 62304
  - Automóvel - ISO 26262

IEC 61508
EN 50128
FDA 510(k)
IEC 62304

IEC 61513
IEC 62061
ISO 26262
DO 178C

Derivações FreeRTOS

OpenRTOS

- Licenciado comercialmente
- Remoção da licença M.I.T. de fonte aberta do FreeRTOS;
- Indemnização comercial e exclusão explícita do código-fonte aberto;
- Suporte profissional e garantia;
- Suporte na criação de drivers personalizados para hardware especifico.
- Portabilidade para novos processadores.

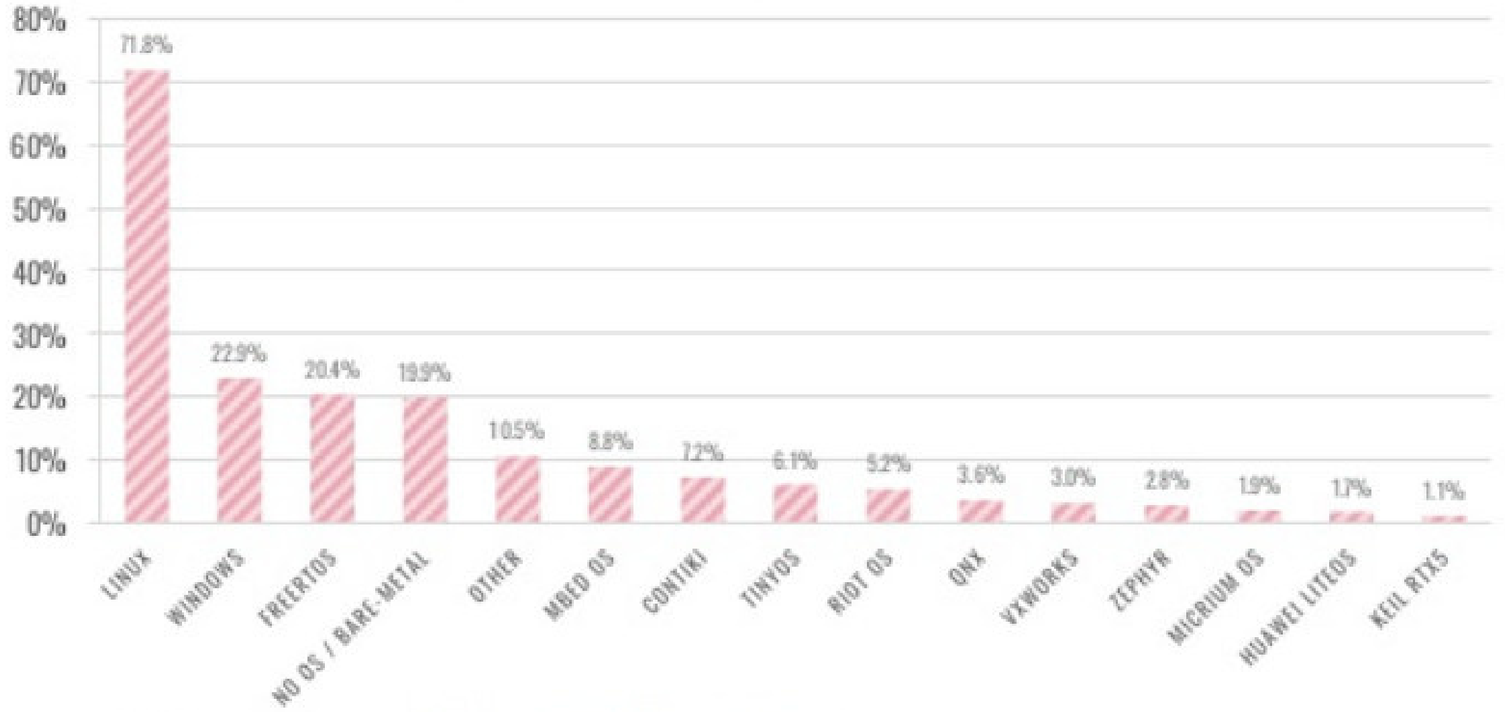
Casos de uso FreeRTOS

- Automação industrial - monitorização, medição e controlo.
- Automóvel - controlo do motor, transmissão e powertrain. Sistemas info-entretenimento gerir tarefas de controlo de áudio e comunicação.
- Dispositivos médicos - medidores de glucose no sangue, monitores de ECG e oxímetros de pulso.
- Aeroespacial - Sistemas de controlo de voo e navegação.
- Robotica - Controlo de tarefas em tempo real e planeamento de movimentos.
- Defesa
- Dispositivos IoT

Casos de uso FreeRTOS

IoT Operating Systems

Which operating system(s) do you use for your IoT devices?



Primeiros passos FreeRTOS

1. Descarregar o código fonte do RTOS
2. Localizar a página de documentação especifica do hardware
3. Construir o projecto
4. Executar a aplicação de demonstração
5. Criar o próprio projecto

Primeiros passos FreeRTOS

Richard Barry.
Mastering the FreeRTOS Real Time Kernel - a Hands On Tutorial Guide.

Amazon Web Services.
FreeRTOS V10.0.0 Reference Manual.

OBRIGADO