



# Universidad Nacional Autónoma de México

## Facultad de Ingeniería

## Sistemas Operativos

*Profesor(a): Ing. Gunnar Eyal Wolf Iszaevich*

*Semestre 2025-1*

## Exposición

Nombre Trabajo

_Sistemas Operativos de _

__Tiempo Real__

Grupo: 6

**Alumno:**

Nava Benítez David Emilio

Tavera Castillo David Emmanuel

Cd. Universitaria a 20 de Marzo de 2025

# Introducción

Un sistema operativo en tiempo real (RTOS) es un sistema operativo especializado que procesa datos y realiza operaciones dentro de límites de tiempo específicamente definidos.
Un sistema operativo tradicional, como Windows, Mac OS o Linux, proporciona una interfaz de software entre las aplicaciones y el hardware. Acepta órdenes de dispositivos de entrada y las ejecuta secuencialmente.
Un sistema operativo de uso general no está diseñado para satisfacer las exigencias de los sistemas integrados o de seguridad crítica, que requieren tiempos de respuesta constantes a entradas de múltiples fuentes, como cámaras, radares y lidares.
Los sistemas operativos en tiempo real se basan en eventos y son preventivos. En otras palabras, el sistema operativo puede supervisar la prioridad relativa de las tareas en competencia y modificar sus prioridades. La forma más rápida de definir un RTOS es que estos son Sistemas Operativos que se utilizan para aplicaciones específicas con algunas restricciones de respuestas en el tiempo.
Para generalizar, se dice que una aplicación es en tiempo real cuando el tiempo de espera es mayor a 0 (t>0).

# ¿De qué me sirve un RTOS?

Un sistema operativo de propósito general se encarga de programar tareas en segundo plano y aplicaciones de usuario, administra una serie de recursos virtuales como archivos, bibliotecas y carpetas, lo que nos permite que las aplicaciones y procesos accedan a ellos cuando sea necesario. También pueden administrar controladores de dispositivos para poder interactuar con el hardware. Estos sistemas operativos están diseñados con la interacción humana como la característica más importante por lo que un programador puede priorizar ciertas tareas lo que implica que se puedan perder o retrasar plazos de tiempo y como consecuencia tener cierto retraso en la capacidad de respuesta y esto es aceptable ya que una persona no lo nota. Pero, ¿esto de que me sirve si tal cual es lo que se ha visto en la clase teórica?
Bueno, no nos interesa explicar un SO de propósito general, sino remarcar las notables diferencias entre estos y los de Tiempo Real para de ahí partir hacia lo más interesante.
La mayoría de los Sistemas Operativos de Tiempo real ofrecen similitudes en funciones de uno de propósito general, pero están diseñados de manera tal que se pueda garantizar el cumplimiento de los plazos de tiempo en las tareas. Por lo que para un RTOS es importante no perder tiempo, ya que cada segundo es vital. Pongámonos en retrospectiva para entender esto: Los RTOS son utilizados en la industria médica donde cada milisegundo importa; por lo tanto es indispensable no perder tiempo, ¿qué pasaría si por cuestiones del Sistema Operativo que maneja un marcapasos introduce una latencia inesperada? Conlleva a que el impulso eléctrico llegase tarde, poniendo en riesgo la vida del paciente. Para ello

están los RTOS, para no perder ni un segundo ya que asegura que las tareas de alta prioridad, como la generación del pulso eléctrico, se ejecuten de inmediato y sin interrupciones inesperadas.

# Arquitectura de un RTOS

Un sistema RTOS normalmente utiliza una arquitectura monolítica o de microkernel.

- **En una arquitectura monolítica:**

El kernel es el corazón del sistema operativo, proporciona servicios básicos para sus otros componentes y actúa como capa principal entre el sistema operativo y el hardware.
El kernel RTOS monolítico y el proceso operativo comparten un espacio, lo que permite un mayor rendimiento en comparación con las configuraciones de microkernel.
Los RTOS monolíticos son rápidos, pero son difíciles de actualizar y los errores de programación en el sistema de archivos, la pila de protocolos o los controladores pueden provocar que el sistema se bloquee.

- **En una arquitectura de microkernel :**

El núcleo y los procesos operativos se encuentran en ubicaciones separadas. Esta arquitectura es más lenta que un RTOS monolítico. Esto se debe a que todas las operaciones deben regresar al núcleo antes de pasarse al componente al que hacen referencia.
El microkernel no tiene un sistema de archivos.
Las configuraciones de microkernel son más fáciles de programar y actualizar, y son más resistentes a los errores de programación.

# Componentes de un RTOS

- **Gestión de tareas**

Los RTOS pueden cambiar de tarea rápidamente, a menudo en tres microsegundos o menos. Esta transferencia de tareas garantiza que los procesos críticos se completen a tiempo, lo que mejora la eficiencia general del sistema. Centrarse en la ejecución de las tareas en cola más importantes y las aplicaciones críticas puede contribuir aún más a su eficacia.

- **Temporizadores**

Los temporizadores en un RTOS son fundamentales para la gestión de tareas, la sincronización y la planificación. Se utilizan para realizar operaciones críticas con precisión temporal.

- **Bloques de control de eventos**

Los bloques de control de eventos (ECB) son estructuras de datos utilizadas para manejar eventos y sincronización entre tareas. Estos bloques almacenan información sobre eventos como señales, temporizadores, semáforos y colas de mensajes.

- **Semáforos y Mutex**

Un semáforo se utiliza cuando se necesita sincronización entre tareas o eventos.
Un mutex se utiliza cuando se requiere acceso exclusivo a un recurso compartido.
El uso adecuado de semáforos y mutex ayuda a evitar problemas de concurrencia y mejora la estabilidad en un RTOS.

- **Banderas de eventos**

Las banderas de eventos (Event Flags) en un RTOS son mecanismos de sincronización que permiten que una tarea espere a que ocurra uno o más eventos antes de continuar su ejecución. Se utilizan cuando una tarea necesita ser notificada sobre eventos específicos generados por otra tarea o una interrupción

- **Buzones de mensajes y colas de mensajes**

En un RTOS, los buzones de mensajes y colas de mensajes son mecanismos esenciales para la comunicación segura entre tareas. Los buzones de mensajes son ideales cuando solo se necesita el último dato disponible, ya que almacenan un único mensaje y lo sobrescriben si llega uno nuevo antes de ser leído. En cambio, las colas de mensajes permiten almacenar múltiples mensajes en orden FIFO, asegurando que la información se procese en el mismo orden en que fue enviada, lo que las hace más adecuadas para el registro de eventos o la transmisión de datos en serie. La elección entre uno u otro depende de las necesidades del sistema, priorizando la eficiencia y la integridad de los datos en la comunicación entre tareas.

- **Gestión de memoria**

En RTOS, la memoria se divide en diferentes secciones, como la pila, el montón y los segmentos de datos. El segmento de pila se utiliza para almacenar variables locales e información de llamadas a funciones. Crece hacia abajo en la memoria y requiere una gestión cuidadosa para evitar errores de desbordamiento o subdesbordamiento. Por otro lado, el segmento del montón asigna memoria dinámicamente para variables u objetos durante la ejecución del programa.
Para gestionar estos segmentos eficientemente, RTOS emplea diversas técnicas, como algoritmos de asignación dinámica (como el primer ajuste o el mejor ajuste) para asignar memoria del segmento del montón.

Es importante gestionar de manera adecuada el uso de la memoria en un RTOS, debido a que estos sistemas operativos es fundamental que sean deterministas (lo que supone tener certeza de lo que sucederá en el tiempo con dicho sistema). En este aspecto, un RTOS maneja la memoria de dos maneras principalmente, de manera estática y dinámica (tal cual un Sistema Operativo sin restricciones de tiempo.

Para entender mejor cómo funciona la gestión de memoria en un RTOS, tomaremos como ejemplo a FreeRTOS y como este asigna memoria, primeramente vamos a explicar de manera concisa como es este proceso y después en un ejemplo más abajo se podrá visualizar de manera más eficiente como se hace.
FreeRTOS asigna memoria en tiempo de ejecución cuando crea una nueva tarea, a esa tarea se le asigna una parte de la memoria del *HEAP.* Dicha parte se divide en un bloque de control de tareas y una pila única específica para dicha tarea. Este bloque de control es una estructura que mantiene información crucial sobre la tarea que se creó, como la ubicación de la pila asignada a la tarea y la prioridad de esta misma.
Con la pila se le informa al sistema operativo cuanta porción de memoria reservar para que actúe como la pila de nuestra tarea; si en dado caso, no se reserva la suficiente memoria, llegaría a sobreescribir información de manera involuntaria lo que causa un comportamiento indefinido o no determinista (algo que no es lo que queremos). Cada tarea que creemos, reservara un bloque de tareas con su correspondiente pila asociada dentro del *HEAP.*
Cuando hablamos de objetos dentro del kernel como colas y semáforos, esto serán almacenados de igual forma dentro del *HEAP.*

Hablando específicamente sobre FreeRTOS, hay una manera de asignar memoria estática para este tipo de objetos y tareas, lo que lo hace versátil y muy útil ya que por ejemplo, en sistemas médicos es importante evitar fugas de memoria que podrían llegar a ser catastróficas. Este enfoque de asignación de memoria estática garantiza que, valga la redundancia, la memoria está preasignada, mejorando la estabilidad del sistema.

- **Atención de interrupciones**

La atención de interrupciones en un sistema operativo en tiempo real (RTOS) se realiza mediante una solicitud de interrupción (IRQ) y una rutina de servicio de interrupción (ISR).

- Solicitud de interrupción (IRQ)

Es una señal que indica al RTOS que debe detener la tarea actual y determinar la siguiente acción.
Ayuda a priorizar tareas y a gestionar múltiples interrupciones.

- Rutina de servicio de interrupción (ISR)

Implementa las funciones de sincronización de un RTOS.
Maneja las interrupciones generadas por el reloj de hardware.

Hablando específicamente de las interrupciones de hardware, debemos saber que los microcontroladores (que usan mayoritariamente RTOS) tienen periféricos integrados que pueden generar *interrupciones de hardware,* estos pueden ser simples temporizadores que acaban su tiempo o un botón que es presionado por un usuario. Importante entender que para un RTOS una interrupción de hardware tendrá las más alta prioridad que cualquier otra tarea o proceso en ejecución aun cuando dicha tarea o proceso tenga su propia prioridad y

por lo tanto dicha interrupción de hardware obligará al procesador a ejecutar la rutina de servicio de interrupción que está asociada a dicho componente.

Una vez atendida la interrupción de hardware el sistema retomará las tareas que ignoro por atender dicha interrupción pero (y un grande pero) puede que regrese a la tarea que estaba ejecutando antes de la irrupción o a alguna otra dependiendo su prioridad. Por ello, en un RTOS se tiene la posibilidad de que para ciertas tareas se deshabilite una o todas las interrupciones, pero esto genera un problema ya que debemos estar conscientes de que tipo de tarea va a ignorar las interrupcion y esa deshabilitación debe hacerse con moderación y aún más si se necesita hacer para proteger algún recurso compartido crítico.

**Conclusiones**

Los RTOS son componentes esenciales para aplicaciones donde el tiempo y la precisión son factores determinantes. Su diseño eficiente, junto con las nuevas tendencias tecnológicas, aseguran que sigan siendo una herramienta clave en el desarrollo de sistemas embebidos y otras aplicaciones críticas en el futuro. El auge tecnológico para satisfacer las necesidades que requieren de eficiencia y sin retrasos hacen que los RTOS sean un tipo de sistema operativo que va a perdurar por el largo curso de la computación y sin duda alguna seguirá evolucionando en el transcurso de los años.

El hecho que los RTOS puedan parecer algo completamente diferente a un SO convencional no los hace para anda diferentes, ya que los sistemas en tiempo real toman la base de lo que son desde los sistemas de propósito general, por lo que no es difícil analizar los RTOS y eso hace que pueda parecer lo mismo, por lo que es requisito indiscutible el pensar a un Sistema Operativo en Tiempo Real como una herramienta que controla el aquí y el ahora, ¿a qué me refiero?, ya que es una herramienta que no debe perder tiempo y en el que no es posible valorar este funcionamiento porque la razón de vivir de este tipo de sistema operativo es esa, no perder tiempo. Por lo que pensándolo bien, a veces quiero ser un RTOS para no perder tiempo en actividades críticas.

# Glosario

RTOS - Real Time Operating System (Sistema Operativo en Tiempo Real)
ECB - Bloque de Control de Eventos
FIFO - First In First Out, concepto de estructura de datos basado en ‘el primero que entra,es el primero en salir’
HEAP - Montículo, espacio de memoria dinámica que se crea para almacenar datos creados durante la ejecución de un programa.
IRQ - Solicitud de interrupción, mecanismo para interrumpir al procesador y poder que se pueda realizar su tarea.

**Referencias**


Aptiv. (2024, 6 diciembre). *¿Qué es un sistema operativo en tiempo real?* Aptiv.https://www.aptiv.com/es/tendencias/art%C3%ADculo/que-es-un-sistema-operativo-en-tiempo-real

Zeifman, I. (2024, 9 junio). *What Is RTOS, How It Works, and 9 RTOS Platforms to Know*. Sternum IoT. https://sternumiot.com/iot-blog/crash-course-introduction-to-real-time-operating-system-rtos/

*Embien Technologies::Using Interrupts for context switching in RTOS*. (s. f.). Embien.

https://www.embien.com/blog/using-interrupts-for-context-switching-in-rtos

*Real-Time Operating System (RTOS): Working and Examples | Spiceworks*. (2025, 10 marzo). Spiceworks Inc. https://www.spiceworks.com/tech/hardware/articles/what-is-rtos/

Barney, N., & Gillis, A. S. (2024, 1 octubre). *What is a real-time operating system (RTOS)?* Search Data Center. https://www.techtarget.com/searchdatacenter/definition/real-time-operating-system

*FreeRTOS documentation - FreeRTOS™*. (s. f.). https://www.freertos.org/Documentation/00-Overview

DigiKey. (2021, 25 enero). *Introduction to RTOS Part 4 - Memory Management | Digi-Key Electronics* [Vídeo]. YouTube. https://www.youtube.com/watch?v=Qske3yZRW5I

DigiKey. (2021a, enero 4). *Introduction to RTOS Part 1 - What is a Real-Time Operating System (RTOS)? | Digi-Key Electronics* [Vídeo]. YouTube. https://www.youtube.com/watch?v=F321087yYy4

Ing. Muro, Gustavo. (s. f.). Introducción a FreeRTOS. Universidad Nacional de Rosario. FACULTAD DE CIENCIAS EXACTAS, INGENIERÍA Y AGRIMENSURA DEPARTAMENTO DE SISTEMAS E INFORMÁTICA

*Index of /docencia/TiempoReal/Recursos/temas*. (s. f.). http://www.isa.uniovi.es/docencia/TiempoReal/Recursos/temas/sotr.pdf