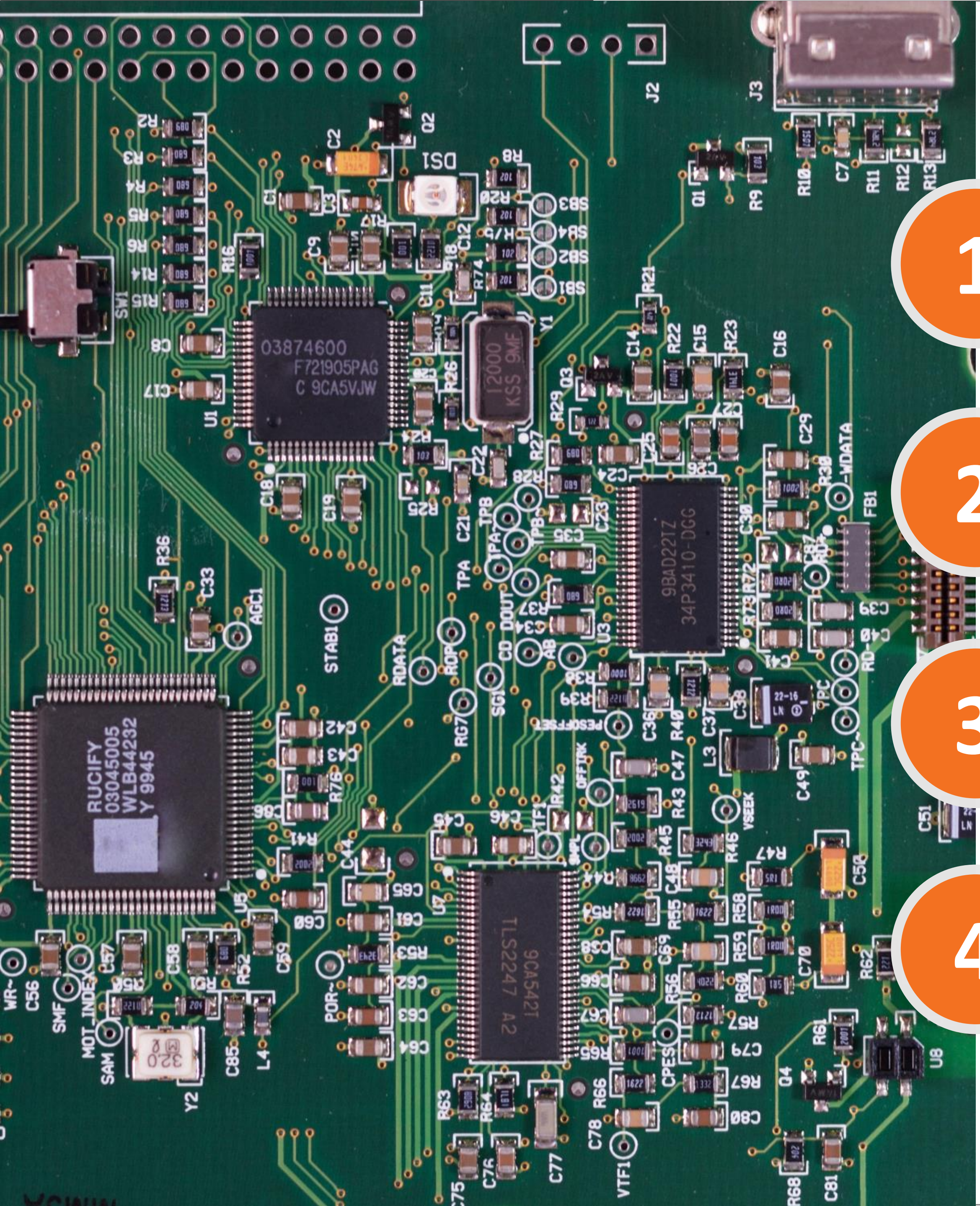
MCT-338: Embedded Systems-II

Lecture 5

Intro. to Real-time Operating Systems

Shujat Ali



1. REAL-TIME APPLICATIONS
2. REAL-TIME OPERATING SYSTEMS
3. REAL-TIME VS GENERAL-PURPOSE OS
4. Q&A

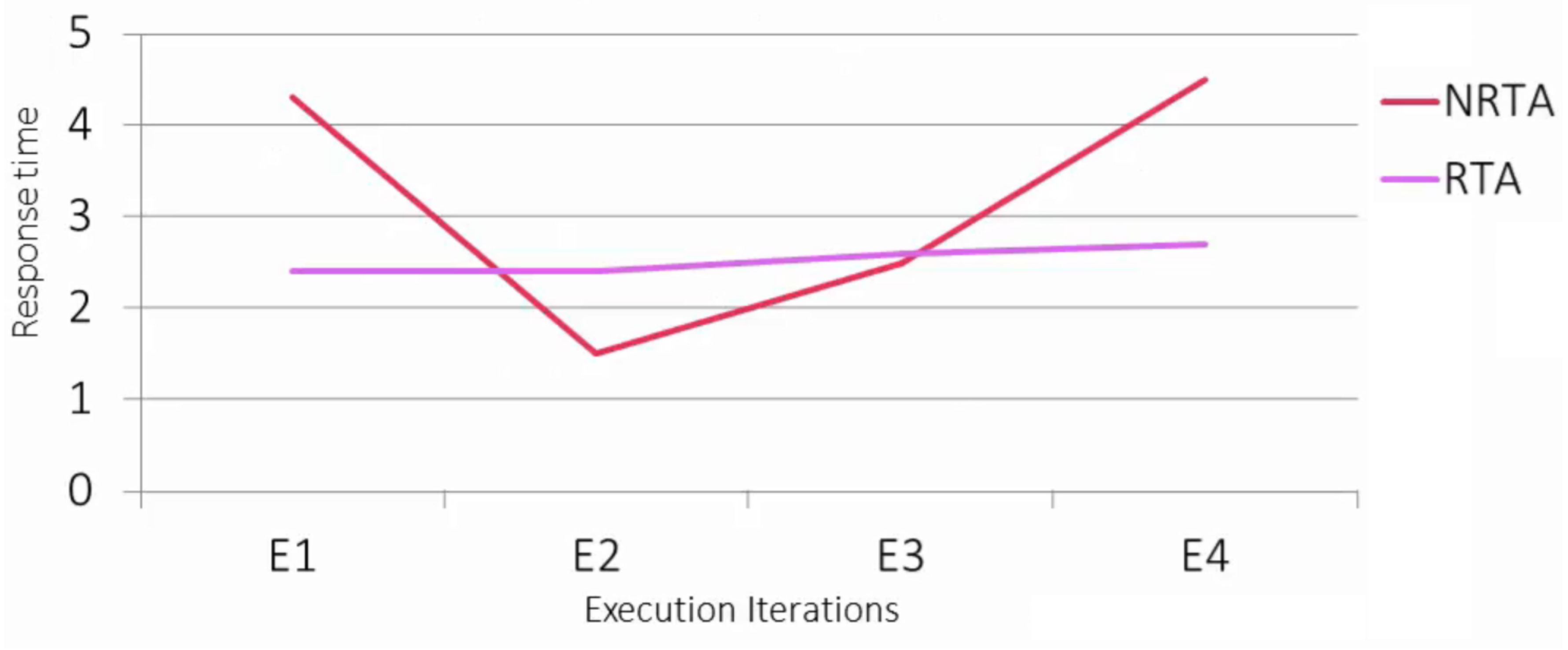
REAL-TIME APPLICATIONS (RTAs)

RTA & it’s examples, Hard vs Soft RTA

Introduction

REAL-TIME APPLICATIONS (RTAs)

- **Real-time applications** (RTAs) are time deterministic applications, that means, their response time to events is almost constant



- RTAs are not necessarily fast executing applications

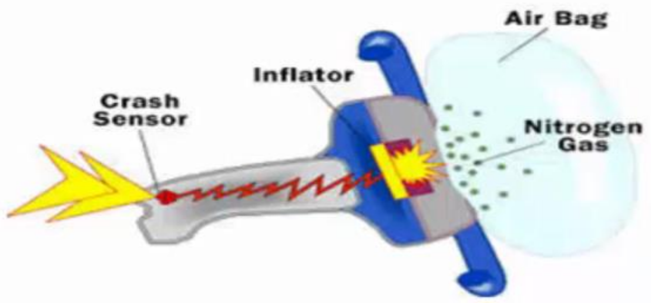
Examples

REAL-TIME APPLICATIONS (RTAs)

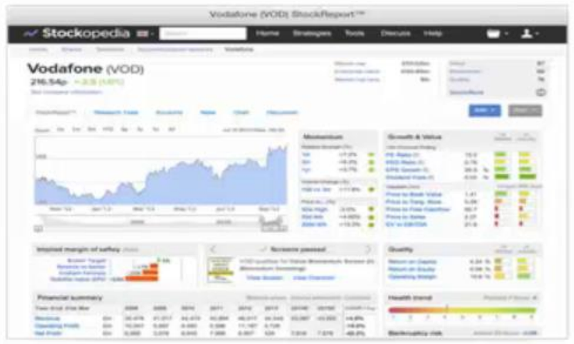
Examples of RTAs are

Missile guidance and control systems

Ani-Lock Breaking System



Airbag Deployment



Vodafone (VOD) StockReport

Stockopedia

Vodafone (VOD)

Stock Market Website

Examples

REAL-TIME APPLICATIONS (RTAs)

Examples of RTAs are

1. **Communications**: Instant messaging platforms, video conferencing, Voice over Internet Protocol (VoIP)
2. **Financial Services**: Stock trading platforms, electronic payment systems, and algorithmic trading
3. **Healthcare and Medicine**: Medical devices, patient monitoring systems, and telemedicine applications
4. **Industrial Automation and Control**: Manufacturing processes, robotics, automatic material handling
5. **Gaming and Entertainment**: Online gaming platforms
6. **Energy and Utilities**: Smart grid systems, energy monitoring, and control systems
7. **Weather Forecasting**: Meteorological systems, weather forecasting, environmental monitoring systems
8. **E-commerce**: Real-time recommendation systems, inventory management, dynamic pricing algorithms
9. **Transportation and Navigation**: GPS navigation systems, traffic monitoring applications, ride-sharing services, and logistics management systems
10. **Emergency Response and Security**: Emergency response systems, surveillance systems, security monitoring applications

Hard vs Soft RTAs

REAL-TIME APPLICATIONS (RTAs)

- RTAs can be classified into

Hard RTAs

1.Timing Constraints: Hard real-time applications have strict and inflexible timing requirements. Tasks must be completed within a specific timeframe, failing which can lead to severe consequences like system failure, safety hazards, or damage.

Soft RTAs

1.Flexible Timing Requirements: Soft real-time applications also have timing constraints, but they are more flexible compared to hard real-time systems. They prioritize timely completion of tasks but can tolerate occasional delays without causing critical failure.

Hard vs Soft RTAs

REAL-TIME APPLICATIONS (RTAs)

- RTAs can be classified into

Hard RTAs

2. **Critical Systems**: These applications are typically found in critical systems where timing guarantees are vital for safety or proper functionality. Examples include aerospace (flight control systems), medical devices (life support systems), automotive (braking systems), and industrial control systems.

Soft RTAs

2. **Non-Critical Systems**: These applications are often found in non-critical systems where occasional delays or missed deadlines may be acceptable. Examples include multimedia streaming, online gaming, certain types of data processing, and user interface responsiveness in applications.

Hard vs Soft RTAs

REAL-TIME APPLICATIONS (RTAs)

- RTAs can be classified into

Hard RTAs

3. **No Tolerance for Failure**: Missing deadlines in hard real-time systems can lead to catastrophic outcomes. These systems are engineered to ensure that deadlines are met with an extremely high degree of certainty.

Soft RTAs

3. **Tolerance for Missed Deadlines**: While timely responses are preferred, soft real-time systems can function acceptably even if a deadline is occasionally missed, without causing significant harm or failure.

REAL-TIME OPERATING SYSTEM (RTOS)

Introduction of RTOS, it’s Key Characteristics, Types, and Components

Introduction

REAL-TIME OPERATING SYSTEM (RTOS)

- To run real-time applications (RTAs), we need real-time operating system (RTOS)
- A **Real-Time Operating System** (RTOS) is a specialized operating system designed to support real-time applications by managing tasks with stringent timing requirements.
- These systems are engineered to ensure that tasks are executed within specified time constraints, providing timely and deterministic responses to external events.
- Real-Time Operating Systems find applications in various domains such as aerospace, automotive, industrial automation, medical devices, telecommunications, and more, where precise timing, reliability, and responsiveness are paramount.

Key Characteristics

REAL-TIME OPERATING SYSTEM (RTOS)

1. **Predictability and Determinism**: RTOSes prioritize predictability in task execution. They provide deterministic behavior by guaranteeing specific time bounds for task completion, allowing for precise control over timing requirements.
2. **Task Scheduling**: RTOSes employ various scheduling algorithms (such as priority-based scheduling or time-driven scheduling) to manage tasks efficiently and meet deadlines. Tasks with higher priority are executed first, ensuring critical tasks receive immediate attention.
3. **Low Latency**: These systems minimize delays between task activation and response, maintaining low-latency behavior to meet real-time constraints.
4. **Resource Management**: RTOSes efficiently manage system resources such as CPU, memory, and I/O devices to ensure optimal performance and timely task execution.
5. **Interrupt Handling**: They have robust mechanisms for handling interrupts and responding promptly to external events, ensuring timely processing of critical tasks.

Types

REAL-TIME OPERATING SYSTEM (RTOS)

Hard Real Time

In Hard RTOS, the deadline is handled very strictly which means that given task must start executing on specified scheduled time and must be completed within the assigned time duration. Example: Medical critical care system, Aircraft systems, etc.

Firm Real time

These type of RTOS also need to follow the deadlines. However, missing a deadline may not have big impact but could cause undesired affects, like a huge reduction in quality of a product. Example: Various types of Multimedia applications.

Soft Real Time

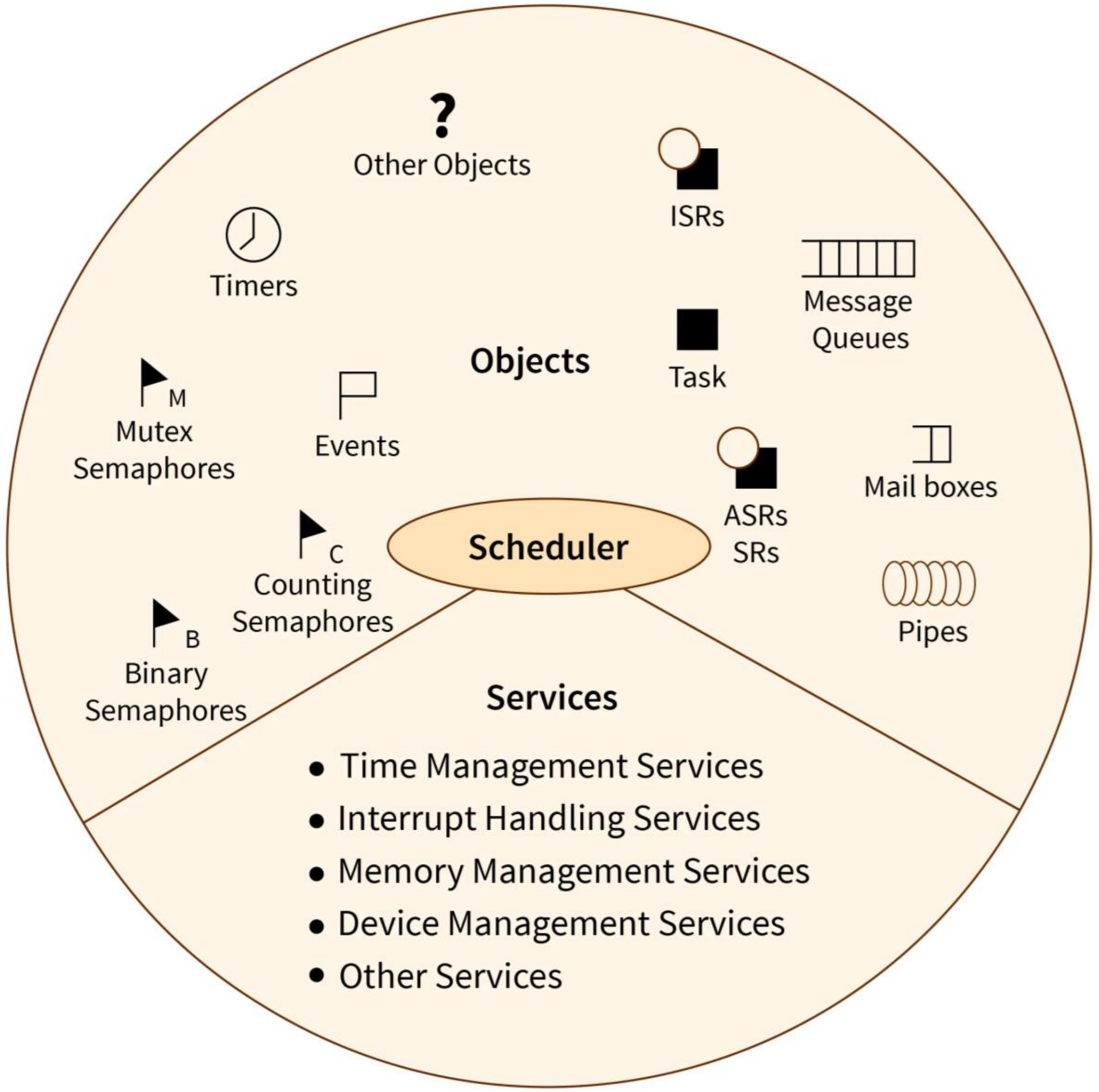
Soft Real time RTOS, accepts some delays by the Operating system. In this type of RTOS, there is a deadline assigned for a specific job, but a delay for a small amount of time is acceptable. So, deadlines are handled softly by this type of RTOS. Example: Livestock price quotation System.

Components

REAL-TIME OPERATING SYSTEM (RTOS)

An RTOS has typically the following components:

1. **Kernel Objects** – special constructs like tasks, message, queue, mailBox, etc.
2. **Kernel Services** – operations that the kernel performs on objects
3. **Task Scheduler** – Determines which task is running at each time instant



Components – 1. Kernal Objects

REAL-TIME OPERATING SYSTEM (RTOS)

Task − to achieve concurrency in RTA program, the application is decomposed into small, schedulable, and sequential program units known as “Task”. It has only 4 possible states: Running, Ready, Blocked, Suspended.

Semaphore − one or more tasks can acquire or release it for the purpose of synchronization or mutual exclusion, i.e., it is just like a key to access a shared resource. It has 3 types: Binary semaphores, Counting semaphores, and MuTex.

MailBox − a data buffer managed by RTOS and is used for sending a message to a task. A task can have a mailbox into which others can post a message

Queue − an array of mailboxes. A task or an ISR deposit the messages in the message queue and other tasks take the messages.

Pipe – A pipe is an RTOS object that provide simple communication channel used for unstructured data exchange among tasks. Pipes can be opened, closed, written to and read from.

Components – 2. Kernal Services

REAL-TIME OPERATING SYSTEM (RTOS)

Kernel is the vital and central component of an operating system. It provides an interface for software applications to use the services (API):

Task Management – allows programmers to design their software as several separate “chunks” of codes with each handling a distinct goal and deadline

Task Synchronization – a mechanism for tasks coordinating with one another to achieve a common objective, e.g., Task 1 = Acquires ADC data and Writes in Memory, Task 2 = Fetch data from Memory and Writes in LCD

Resource Synchronization – required to access a shared resource, e.g., Task 1 to Display - "Temperature=100C“, Task 2 to Display - "Humidity=40%" on a shared resource (i.e., LCD)

InterTask Communication – mechanism for tasks to communicate with each other to manage program execution. It is needed as only one task can run at a time & other tasks must be informed.

Timer Management – to schedule system and user tasks, a periodic timer interrupt is required to keep track of the time delays, timeout, etc.

Components – 2. Kernal Services

REAL-TIME OPERATING SYSTEM (RTOS)

Memory Management − use of non-fragmenting memory allocation techniques, e.g., memory pools, in RTOS to avoid both memory fragmentation and garbage collection & its consequences

Interrupts & Events Handling − Interrupt & event handling mechanism of an RTOS help to ensure

1. Data integrity by restricting interrupts from occurring when modifying a data structure
2. Fastest possible interrupt responses that marked the preemptive performance of an RTOS
3. Shortest possible interrupt completion time with minimum overheads
4. Minimum interrupt latencies due to disabling of interrupts when RTOS is performing critical operations

Device I/O Management − provide a uniform framework (application programmer's interface-"API") for accessing hardware resources of a processor. it is a supervision facility for an embedded system to organize and access large numbers of diverse hardware device drivers.

Components – 3. Task Scheduler

REAL-TIME OPERATING SYSTEM (RTOS)

Task scheduling in a RTOS is the process of determining which task should be executed next, based on the system's resources and the tasks' priorities and deadlines.

RTOS task scheduling is critical for ensuring that real-time tasks meet their deadlines, as even a small delay in executing a task can have serious consequences.

There are two main types of RTOS task scheduling algorithms: preemptive and non-preemptive.

1. **Preemptive scheduling algorithms** allow the RTOS to interrupt a running task and switch to a higher-priority task. This is important for ensuring that critical tasks meet their deadlines, even if lower-priority tasks are already running.
2. **Non-preemptive scheduling algorithms** do not allow the RTOS to interrupt a running task. This can be useful for tasks that need to run to completion without being interrupted, but it can also lead to missed deadlines for higher-priority tasks if a lower-priority task takes a long time to complete.

Components – 3. Task Scheduler

REAL-TIME OPERATING SYSTEM (RTOS)

In addition to preemption, RTOS task scheduling algorithms also consider the following factors:

- **Task priority**: Each task in an RTOS is assigned a priority. Higher-priority tasks are always scheduled to run before lower-priority tasks.
- **Task deadline**: Some RTOS task scheduling algorithms also consider the deadlines of tasks when making scheduling decisions. For example, a deadline-based scheduling algorithm might give priority to tasks that are closer to their deadlines.
- **Task execution time**: Some RTOS task scheduling algorithms also consider the estimated execution time of tasks when making scheduling decisions. This can help to ensure that all tasks are scheduled to complete within their deadlines.
- **Resource availability**: Some tasks may require resources, such as memory or peripherals. The RTOS scheduler will only schedule tasks if the required resources are available.

Components – 3. Task Scheduler

REAL-TIME OPERATING SYSTEM (RTOS)

Some common RTOS task scheduling algorithms include:

- **Priority-based scheduling**: This is the most common type of RTOS task scheduling algorithm. It simply assigns a priority to each task and schedules the task with the highest priority to run first.
- **Round-robin scheduling**: This algorithm gives each task an equal amount of time to run, regardless of its priority. This can be useful for tasks that need to be executed periodically, but it can also lead to missed deadlines for higher-priority tasks.
- **Deadline-based scheduling**: This algorithm schedules tasks based on their deadlines. Tasks with earlier deadlines are scheduled to run first. This algorithm can help to ensure that all tasks meet their deadlines, but it can be complex to implement.

The choice of RTOS task scheduling algorithm depends on the specific needs of the RT system. e.g.,

  - A system with a mix of critical and non-critical tasks might use a **priority-based scheduling** algorithm.
  - A system with tasks that need to be executed periodically might use a **round-robin scheduling** algorithm.
  - A system with tasks that have hard deadlines might use a **deadline-based scheduling** algorithm.

Components – 3. Task Scheduler

REAL-TIME OPERATING SYSTEM (RTOS)

Examples of Task Scheduler:

A car engine control unit (ECU) uses an RTOS to schedule tasks such as fuel injection, ignition, and emissions control. These tasks must be completed within very strict timing constraints in order to ensure that the engine runs smoothly and efficiently.

A mobile phone uses an RTOS to schedule tasks such as the user interface, the cellular network stack, and the media player. These tasks must be scheduled efficiently in order to provide a good user experience and to conserve battery life.

A medical device such as a pacemaker uses an RTOS to schedule tasks such as heart pacing and monitoring. These tasks must be completed within very strict timing constraints in order to ensure the safety of the patient.

REAL-TIME VS GENERAL-PURPOSE OS

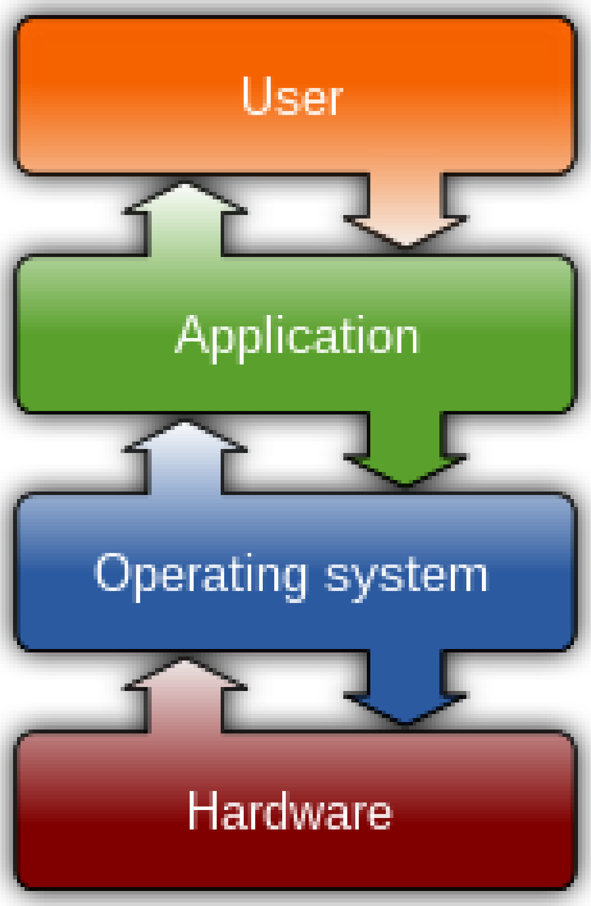
What is an Operating System, Head-to-head Comparison of RTOS and GPOS

What is an Operating System (OS)?

REAL-TIME VS GENERAL-PURPOSE OS

An **operating system** (OS) is a system software that manages computer hardware and software resources and provides common services for computer programs.

- An OS brings powerful benefits to computer software and software development.
- Without an operating system, every application would need to include its own UI, as well as the comprehensive code needed to handle all low-level functionality of the underlying computer, such as disk storage, network interfaces and so on.
- Considering the vast array of underlying hardware available, this would vastly bloat the size of every application and make software development impractical.



Head-to-head Comparison

REAL-TIME VS GENERAL-PURPOSE OS

Real-time Operating System (RTOS)

1. The RTOS always uses priority-based scheduling.
2. The time response of the RTOS is deterministic.
3. A low-priority job in an RTOS would be pre-empted by a high-priority one if required, even executing a kernel call.

General-purpose Operating System (GPOS)

1. Task scheduling in a GPOS isn't necessarily based on which application or process is the most important. Threads and processes are often dispatched using a "fairness" policy.
2. The time response of the general-purpose operating system is not deterministic.
3. A high-priority thread in a GPOS cannot preempt a kernel call.

Head-to-head Comparison

REAL-TIME VS GENERAL-PURPOSE OS

Real-time Operating System (RTOS)	General-purpose Operating System (GPOS)
4. The real-time operating system optimizes memory resources.	4. The GPOS does not optimize the memory resources.
5. The RTOS is mainly used in the embedded system.	5. GPOS is mainly used in PC, servers, tablets, and cell phones.
6. The real-time operating system has a task deadline.	6. The general-purpose operating system has no task deadline.
7. It doesn't have large memory.	7. It has a large memory.
8. RTOS kernel code is intended to be scalable, allowing developers to selectively select kernel objects.	8. GPOS code is not often modular in nature when it comes to development.

Head-to-head Comparison

REAL-TIME VS GENERAL-PURPOSE OS

Real-time Operating System (RTOS)

9. RTOS is designed and developed for a single-user environment.
10. Examples:

VxWorks
QNX
freeRTOS
INTEGRITY

General-purpose Operating System (GPOS)

9. GPOS is designed for a multi-user environment.
10. Examples:

Linux
Windows 10
iOS
android

THANK YOU

Any Questions???