EMBEDDED OPERATING SYSTEM

Introduction of Embedded Systems

- **System** is a set of interrelated parts/components which are designed/developed to perform common tasks or to do some specific work for which it has been created.
- **Embedded** means including something with anything for a reason. Or simply we can say something which is integrated or attached to another thing.

What is Embedded System?

- **Embedded system** is a computational system that is developed based on an integration of both hardware and software in order to perform a given task.
- It can be said as a dedicated computer system has been developed for some particular reason.
- But it is not our traditional computer system or general-purpose computers, these are the Embedded systems that may work independently or attached to a larger system to work on a few specific functions.
- These embedded systems can work without human intervention or with little human intervention.

Components of Embedded Systems:

1. Hardware

2. Software

3. Firmware

Examples of Embedded Systems:

- Digital watches
- Washing Machine
- Toys
- Televisions
- Digital phones
- Laser Printer
- Cameras
- Industrial machines
- Electronic Calculators
- Automobiles
- Medical Equipment

Application of Embedded System:

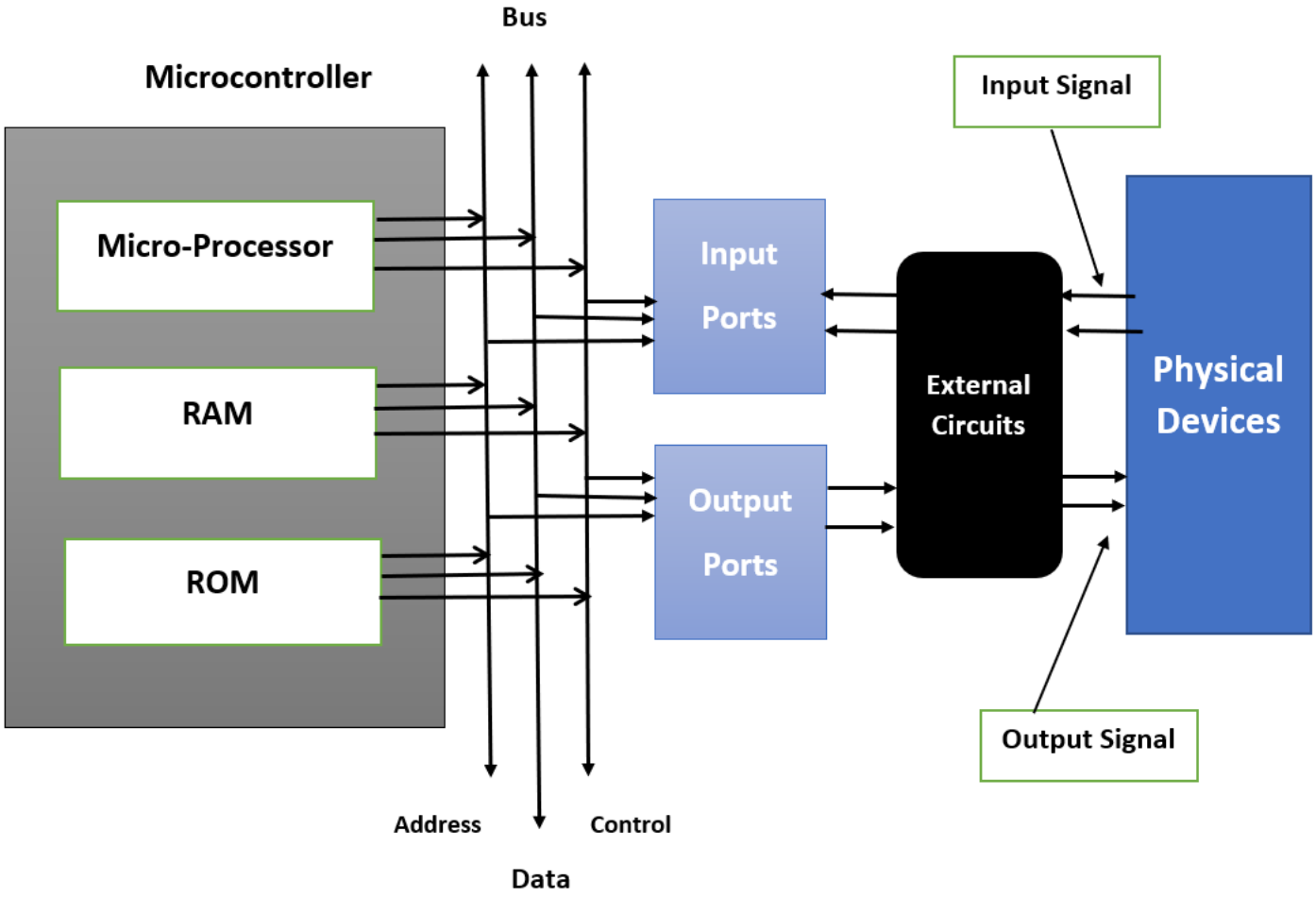
- Home appliances
- Transportation
- Health care
- Business sector & offices
- Defense sector
- Aerospace
- Agricultural Sector

Characteristics of an Embedded System:

- **Performs specific task:** Embedded systems perform some specific function or tasks.

- **Low Cost:** The price of an embedded system is not so expensive.
- **Time Specific:** It performs the tasks within a certain time frame.
- **Low Power:** Embedded Systems don't require much power to operate.
- **High Efficiency:** The efficiency level of embedded systems is so high.
- **Minimal User interface:** These systems require less user interface and are easy to use.
- **Less Human intervention:** Embedded systems require no human intervention or very less human intervention.
- **Highly Stable:** Embedded systems do not change frequently mostly fixed maintaining stability.
- **High Reliability:** Embedded systems are reliable they perform tasks consistently well.
- **Use microprocessors or microcontrollers:** Embedded systems use microprocessors or microcontrollers to design and use limited memory.
- **Manufacturable:** The majority of embedded systems are compact and affordable to manufacture. They are based on the size and low complexity of the hardware.

Block Structure of Embedded System



Embedded System

Advantages of Embedded System:

- Small size.
- Enhanced real-time performance.
- Easily customizable for a specific application.

Disadvantages of Embedded System:

- High development cost.
- Time-consuming design process.
- As it is application-specific less market available.

Top Embedded Programming Languages:

- Some of the programming languages used in the development of embedded systems include, Embedded C, Embedded C++, Embedded Java, Embedded Python etc.
- But it completely rests on the developer which programming language he selects for the development of the embedded systems.

How does an Embedded System Work?

- Embedded systems operate from the combination of hardware and software that focuses on certain operations.

- An embedded system at its heart has microcontroller or microprocessor hardware on which user writes the code in form of software for control of the system.
- Here is how it generally works:

- **Hardware Layer:** Some of the hardware elements that are incorporated in an embedded system include the sensor, actuator, memory, current I/O interfaces as well as power supply. These components are interfaced with the micro controller or micro processor depending up on the input signals accepted.
- **Input/Output (I/O) Interfaces:** They to give the system input in form of data from sensors or inputs made by the users and the microcontroller processes the data received. The processed data is then utilized to coordinate the output devices such as displays, motors or communication modules.
- **Firmware:** Firmware which is integrated within a system's hardware comprises of certain instructions to accomplish a task. Such software is often used for real time processing and is tuned to work in the most optimal manner on the system hardware.
- **Processing:** Depending on the given software and the input data received from the system's inputs the microcontroller calculates the appropriate output or response and manages the system's components.
- **Real-time Operation:** Some of the most common systems are real time, this implies that they have the ability to process events or inputs at given time. This real time capability makes sure that the system accomplishes its intended function within stated time demands.

For instance therein an embedded system in a washing machine, the microcontroller would interface with the buttons (selections made by a user), sensors, for instance water levels, temperature and timers; it would control outputs such as motors, heaters and displays among others based on the program intended for washing cycles.

What is an embedded operating system?

- An embedded operating system is a specialized operating system (OS) designed to perform a specific task for a device that is not a computer.
- The main job of an embedded OS is to run the code that allows the device to do its job.
- The embedded OS also makes the device's hardware accessible to software that is running on top of the OS.
- An embedded OS often works within an embedded system. An embedded system is a computer that supports a machine.

- It performs one task in the bigger machine. Examples include computer systems in cars, traffic lights, digital televisions, ATMs, airplane controls, point of sale (PoS) terminals, digital cameras, GPS navigation systems, elevators and Smart meters.
- Networks of devices containing embedded systems make up the internet of things (IoT).
- The embedded systems perform basic operations inside IoT devices, such as transferring data over a network without human interaction.

How does an embedded OS work?

- An embedded OS enables an embedded device to do its job within a larger system.
- It communicates with the hardware of the embedded system to perform a specific function. For example, an elevator might contain an embedded system, such as a microprocessor or microcontroller, that lets it understand which buttons the passenger is pressing.
- The embedded software that runs on that system is the embedded OS.
- In contrast to an OS for a general-purpose computer, an embedded OS has limited functionality. Depending on the device in question, the system may only run a single embedded application. However, that application is likely crucial to the device's operation. Given that, an embedded OS must be reliable and able to run with constraints on memory and processing power.
- In the case of a Raspberry PI system on a chip, an SD card acts as the device's hard drive and contains the code that runs on the device. The SD card is removable, so its contents can be modified on demand. Various operating systems can run on Raspberry PI devices.
- The embedded OS makes the device's hardware -- such as USB and HDMI ports -- accessible to the application running on top of the OS.

Examples of embedded OS devices:

Some examples of devices with embedded OSes include the following:

- ATMs
- cellphones
- electric vehicles
- industrial control systems (ICS)
- Arduino-based devices

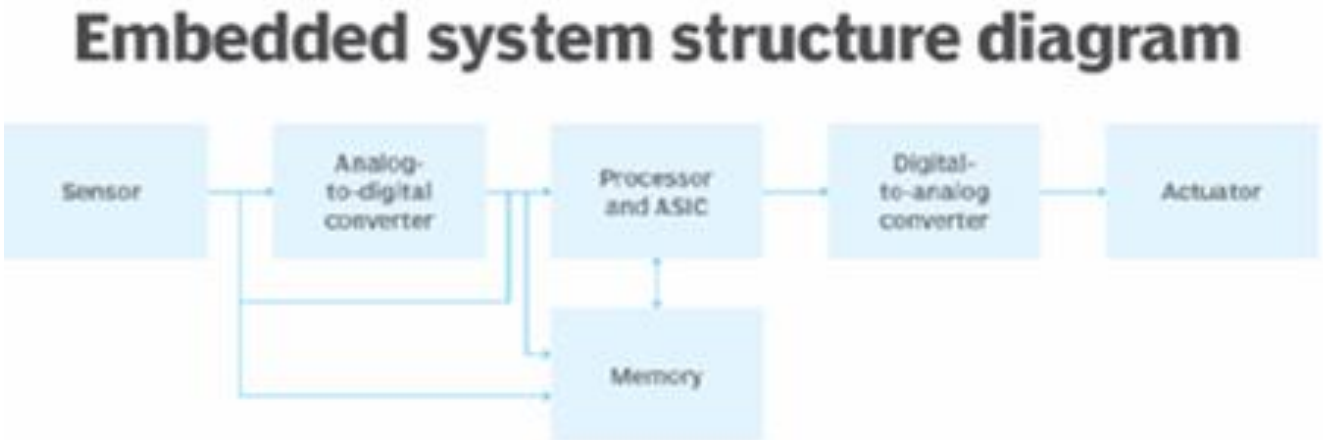
Arduino is an Open Source platform with a microcontroller that processes simple inputs, such as temperature or pressure, and turns them into outputs. These devices have a basic embedded OS that acts like a boot loader and a command interpreter.

An example of an Arduino-based device is a remote control car. The Arduino reads inputs from the car's controller and sends output information and commands to other components, such as the brakes.

Common uses of embedded OSes

Embedded OSes are put to a variety of uses, including the following:

- **ATMs.** ATMs have basic OSes that enable the machine to read a user's debit card and personal identification number input and perform bank account functions like withdrawal or checking balances. The OS does little else but react to user inputs and communicate with the ATM hardware.
- **Cellphones.** Cellphones require an OS like Android or iOS to boot the phone and enable applications to communicate with other phone hardware.
- **Electric vehicles.** Microcontrollers host embedded OSes that handle functions like braking or pressure sensing. For example, a certain amount of pressure on the front bumper may cause the airbag to go off. This type of function is known as reactive operation because it reacts to an input.
- **Industrial control systems.** Sensors are used in industrial control systems to measure factory conditions and send alerts if they become dangerous. Sensors contain an embedded OS that enable them to perform these tasks.
- **Traffic lights.** Embedded OSes enable a traffic light to cycle through different signals at programmed intervals.
- **Basic input/output system.** In some cases, BIOS could be considered an embedded OS because it is the firmware that enables a desktop computer's more complex OS to interact with the computer hardware.



Embedded systems contain hardware components that the embedded operating system organizes to perform a task for the larger device.

Types of embedded OSes:

Embedded OS are designed for the task they will perform. The various types of operating systems include the following:

- **Multitasking operating system.** A Multitasking OS can perform several tasks at once. It uses job scheduling to perform basic tasks. For example, a cellphone OS divides up CPU resources among multiple tasks.
- **Real-time operating system.** A real-time OS is designed to be reactive. It processes inputs when they are received and responds within a specific timeframe. If the response time falls outside of the specified time period, the system could fail. Real-time OSes sometimes use rate monotonic scheduling, which assigns priorities to tasks.
- **Single loop control system.** This type of embedded OS exercises control over a single variable. An example would be temperature control in a smart home. A smart thermostat measures the temperature in the house and if it exceeds the limit set by the user, turns off the heat.

What is a Real-Time Operating System (RTOS)?

- A Real-Time Operating System (RTOS) is a specialized operating system designed to handle time-critical tasks with strict deadlines.
- The primary goal of an RTOS is to ensure deterministic behavior and predictable response times for real-time applications.
- RTOSes are commonly used in embedded systems, industrial control systems, automotive systems, and other domains where timely execution of tasks is of utmost importance.

Key Characteristics of RTOS

1. **Determinism**: RTOS guarantees that tasks are executed within predefined time constraints, ensuring predictable behavior.
2. **Real-Time Scheduling**: RTOS employs real-time scheduling algorithms, such as Rate Monotonic Scheduling (RMS) or Earliest Deadline First (EDF), to prioritize and schedule tasks based on their timing requirements.
3. **Preemptive Multitasking**: RTOS allows higher-priority tasks to preempt lower-priority tasks, ensuring that critical tasks are executed in a timely manner.
4. **Low Latency**: RTOS minimizes the latency between the occurrence of an event and the system's response to that event, enabling quick reaction times.

5. **Resource Management**: RTOS provides efficient resource management mechanisms, such as semaphores, mutexes, and message queues, to handle inter-task communication and synchronization.
6. **Small Footprint**: RTOS is designed to have a small memory footprint and minimal overhead, making it suitable for resource-constrained embedded systems.

Popular RTOS Examples

- VxWorks
- QNX
- FreeRTOS
- RTEMS
- ThreadX

What is a General-Purpose Operating System (GPOS)?

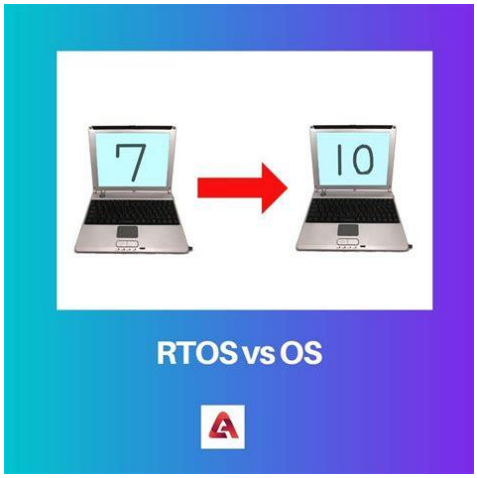
- A General-Purpose Operating System (GPOS) is a versatile operating system designed to support a wide range of applications and user needs.
- GPOSes are commonly used in personal computers, servers, and mobile devices.
- They provide a rich set of features and services to facilitate the development and execution of various software applications.

Key Characteristics of GPOS:

1. **Versatility**: GPOS supports a wide range of hardware and software platforms, making it suitable for diverse computing needs.
2. **User-Friendly Interface**: GPOS provides a user-friendly graphical user interface (GUI) or command-line interface (CLI) to interact with the system and applications.
3. **Resource Sharing**: GPOS allows multiple users and applications to share system resources, such as memory, CPU, and I/O devices, in a controlled and secure manner.
4. **Process Management**: GPOS manages processes, allocates resources, and provides mechanisms for inter-process communication and synchronization.
5. **File System**: GPOS includes a file system that organizes and manages files and directories, enabling persistent storage and retrieval of data.
6. **Networking**: GPOS supports networking capabilities, allowing communication and resource sharing among multiple systems.
7. **Security**: GPOS includes security features, such as user authentication, access control, and data protection, to ensure the integrity and confidentiality of the system and its data.

Popular GPOS Examples

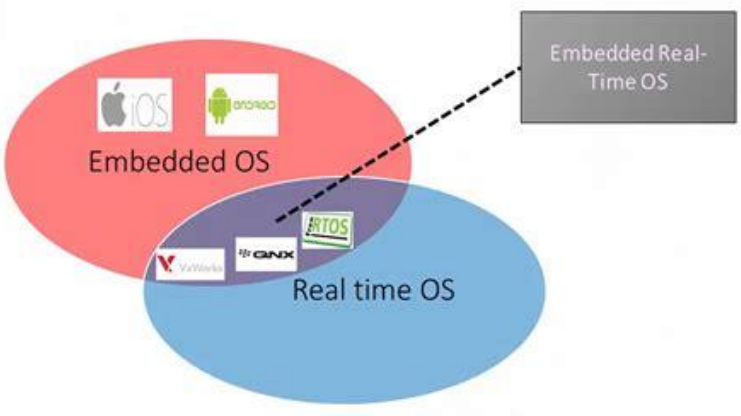
- Microsoft Windows
- macOS
- Linux
- Unix
- Android



Use Cases for RTOS

RTOS is commonly used in the following domains:

1. **Automotive Systems**: RTOS is used in automotive electronic control units (ECUs) for tasks such as engine control, braking systems, and advanced driver assistance systems (ADAS).
2. **Industrial Control Systems**: RTOS is employed in industrial automation, process control, and supervisory control and data acquisition (SCADA) systems to ensure real-time monitoring and control of industrial processes.
3. **Aerospace and Defense**: RTOS is used in avionics systems, flight control systems, and mission-critical defense applications that require deterministic behavior and strict timing constraints.
4. **Medical Devices**: RTOS is utilized in medical devices, such as patient monitoring systems, surgical equipment, and life-support systems, where real-time response and reliability are crucial.
5. **Robotics**: RTOS is used in robotic systems to control actuators, sensors, and real-time decision-making algorithms, enabling precise and responsive robot behavior.



Use Cases for GPOS:

GPOS is widely used in the following scenarios:

1. **Personal Computing**: GPOS is the primary operating system for personal computers, providing a user-friendly interface and supporting a wide range of productivity applications and multimedia.
2. **Server Systems**: GPOS is used in server systems to host websites, databases, and networked applications, offering scalability, reliability, and security features.
3. **Mobile Devices**: GPOS, such as Android and iOS, powers smartphones and tablets, providing a rich ecosystem of applications and services for communication, entertainment, and productivity.
4. **Software Development**: GPOS provides a comprehensive development environment with tools, libraries, and frameworks for creating and testing software applications across various domains.
5. **Scientific Computing**: GPOS is used in scientific and research environments to run complex simulations, data analysis, and visualization tasks, leveraging its versatility and extensive software support.

Real Time Operating System (RTOS):

Real-time operating systems (RTOS) are used in environments where a large number of events, mostly external to the computer system, must be accepted and processed in a short time or within certain deadlines. such applications are industrial control, telephone switching equipment, flight control, and real-time simulations.

- Designed to respond to tasks **quickly and predictably**.
- Ensures tasks are completed within a **fixed time limit** (deadline).
- Used in systems where **timing is critical** (e.g., robots, medical devices, cars).
- Supports **mult**itaski**ng** with precise scheduling.

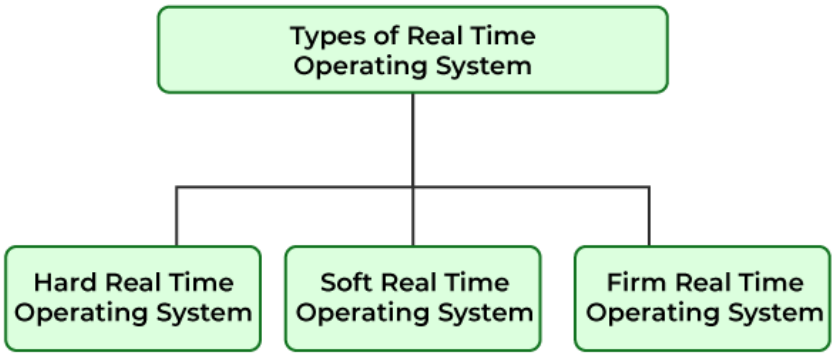
Examples: Airline traffic control systems, Command Control Systems, airline reservation systems, Heart pacemakers, Network Multimedia Systems, robots, etc.

- A real-time operating system (RTOS) is a special kind of operating system designed to handle tasks that need to be completed quickly and on time.
- Unlike general-purpose operating systems (GPOS), which are good at multitasking and user interaction, RTOS focuses on doing things in real time.
- The idea of real-time computing has been around for many years. The first RTOS was created by Cambridge University in the 1960s.
- This early system allowed multiple processes to run at the same time, each within strict time limits.

- Over the years, RTOS has improved with new technology and the need for reliable real-time performance.
- These systems are now more powerful, efficient, and full of features, and they are used in many industries, including aerospace, defense, medical science, multimedia, and more.

Types of Real-Time Operating System:

The real-time operating systems can be of 3 types -



- **Hard Real-Time Operating System :** These operating systems guarantee that critical tasks are completed within a range of time. For example, a robot is hired to weld a car body. If the robot welds too early or too late, the car cannot be sold, so it is a hard real-time system that requires complete car welding by the robot hardly on time., scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.
- **Soft Real-Time Operating System :** This operating system provides some relaxation in the time limit. For example - Multimedia systems, digital audio systems, etc. Explicit, programmer-defined, and controlled processes are encountered in real-time systems. A separate process is changed by handling a single external event. The process is activated upon the occurrence of the related event signaled by an interrupt. Multitasking operation is accomplished by scheduling processes for execution independently of each other. Each process is assigned a certain level of priority that corresponds to the relative importance of the event that it services. The processor is allocated to the highest-priority processes. This type of schedule, called, priority-based preemptive scheduling is used by real-time systems.
- **Firm Real-time Operating System :** RTOS of this type have to follow deadlines as well. In spite of its small impact, missing a deadline can have unintended consequences, including a reduction in the quality of the product. Example: Multimedia applications.

Purpose of RTOS:

Unlike general-purpose operating systems (GPOS) like Windows or Linux, which are good at multitasking and handling various applications, a real-time operating system (RTOS) is designed to manage time-sensitive tasks precisely.

The main goal of an RTOS is to perform critical tasks on time. It ensures that certain processes are finished within strict deadlines, making it perfect for situations where timing is very important. It is also good at handling multiple tasks at once.

An RTOS provides real-time control over hardware resources, like random access memory (RAM), by ensuring predictable and reliable behavior. It uses system resources efficiently while maintaining high reliability and responsiveness. By managing multiple tasks effectively, an RTOS ensures smooth operation even when the system is under heavy use or changing conditions.

Uses of RTOS:

- Defense systems like RADAR .
- Air traffic control system.
- Networked multimedia systems.
- Medical devices like pacemakers.
- Stock trading applications.

Different Between Regular and Real-Time operating systems

Regular OS	Real-Time OS (RTOS)
Complex	Simple
Best effort	Guaranteed response
Fairness	Strict Timing constraints
Average Bandwidth	Minimum and maximum limits
Unknown components	Components are known
Unpredictable behavior	Predictable behavior
Plug and play	RTOS is upgradeabl

Advantages

- **Maximum Consumption:** Maximum utilization of devices and systems. Thus more output from all the resources.
- **Task Shifting:** Time assigned for shifting tasks in these systems is very less. For example, in older systems, it takes about 10 microseconds. Shifting one task to another and in the latest systems, it takes 3 microseconds.
- **Focus On Application:** Focus on running applications and less importance to applications that are in the queue.
- **Real-Time Operating System In Embedded System:** Since the size of programs is small, RTOS can also be embedded systems like in transport and others.
- **Error Free:** These types of systems are error-free.
- **Memory Allocation:** Memory allocation is best managed in these types of systems.

Disadvantages

- **Limited Tasks:** Very few tasks run simultaneously, and their concentration is very less on few applications to avoid errors.
- **Use Heavy System Resources:** Sometimes the system resources are not so good and they are expensive as well.
- **Complex Algorithms** : The algorithms are very complex and difficult for the designer to write on.
- **Device Driver And Interrupt Signals:** It needs specific device drivers and interrupts signals to respond earliest to interrupts.
- **Thread Priority:** It is not good to set thread priority as these systems are very less prone to switching tasks.
- **Minimum Switching:** RTOS performs minimal task switching.