**Introduction**

SquadRTOS is a real-time operating system designed to provide a reliable and efficient platform for embedded systems. It is specifically tailored for small to medium-sized embedded systems, and its highly customizable architecture can be adapted to meet the specific needs of each project.

One of the key features of SquadRTOS is its preemptive multithreading kernel, which allows multiple tasks to run concurrently while ensuring that critical tasks are given priority over non-critical tasks. This makes it an ideal choice for applications that require precise timing and responsiveness, such as robotics, control systems, and automotive electronics.

Another advantage of SquadRTOS is its comprehensive set of libraries, drivers, and tools that simplify the development process. This helps to reduce development time and costs, while improving the quality and reliability of the final product.

Overall, SquadRTOS is a powerful and flexible real-time operating system that provides an efficient and reliable platform for embedded systems. Its customizable architecture, real-time capabilities, and development tools make it an ideal choice for a wide range of applications in various industries, including aerospace, medical devices, and industrial automation. With its proven track record of success, SquadRTOS is a trusted and reliable choice for developers looking to build high-performance embedded systems.

**Defining RTOS**

RTOS stands for real-time operating system, which is a specialized operating system that is designed to provide a reliable and predictable platform for embedded systems that require precise timing and responsiveness.

Unlike general-purpose operating systems, which are designed to provide a wide range of features and functionality, RTOS is designed to prioritize real-time performance and ensure that critical tasks are given priority over non-critical tasks. This is achieved through features such as preemptive multitasking, task scheduling, and interrupt handling.

RTOS is commonly used in applications such as robotics, control systems, and automotive electronics, where precise timing and responsiveness are critical for the proper functioning of the system. It provides a highly customizable and efficient platform for developing embedded systems, with features such as low-level hardware access, optimized algorithms, and comprehensive libraries and tools.

Overall, RTOS is a powerful and specialized operating system that provides a reliable and predictable platform for embedded systems that require real-time performance.

**why are we made Real-Time Operating system (SquadRTOS vs Super loop and Interrupts )?**

In the context of embedded systems development with SquadRTOS, real-time operating systems (RTOS) offer several advantages over super loops and interrupts. RTOS provides a preemptive multitasking kernel, allowing multiple tasks to run concurrently while ensuring that critical tasks are given priority over non-critical tasks. This makes it ideal for applications that require precise timing and responsiveness.

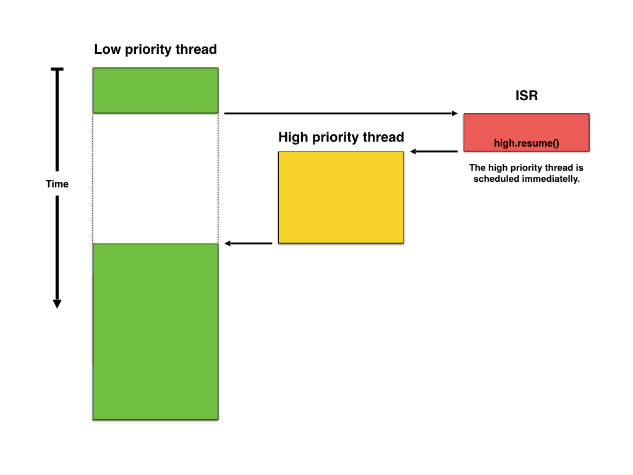
Super loops, on the other hand, can be simple to implement but may not be able to handle complex systems with multiple tasks. Interrupts can be used to handle events but may result in high system overhead and increased complexity.

SquadRTOS provides a highly efficient and reliable real-time operating system that offers minimal system overhead and simplified development. This includes features such as optimized algorithms, low-level hardware access, and full control over system resources.

Overall, while super loops and interrupts can be useful for simple systems, SquadRTOS provides a more reliable and efficient platform for larger and more complex systems that require precise timing and responsiveness.

Super loop Tasks with priorities if task have Priority higher than current thread it will work

Chart, waterfall chart

Description automatically generated 

**Hard real-time systems and Soft real-time systems**

Hard real-time systems and soft real-time systems are two categories of real-time systems that are differentiated by their degree of tolerance for missed deadlines.

Hard real-time systems are designed to be extremely time-sensitive and must meet strict timing constraints. Failure to meet these constraints can result in catastrophic consequences, such as system failure or loss of life. Examples of hard real-time systems include aerospace control systems, medical devices, and automotive safety systems. In hard real-time systems, missing a deadline can result in system failure, making it critical to ensure that all tasks are completed within the specified time limit.

Soft real-time systems, on the other hand, have less strict timing constraints, and missing a deadline may not have catastrophic consequences. Examples of soft real-time systems include multimedia applications, such as video and audio streaming, where occasional delays may be acceptable, but consistent delays can result in a degraded user experience.

Overall, the distinction between hard real-time systems and soft real-time systems is based on the degree of tolerance for missed deadlines, with hard real-time systems requiring strict adherence to timing constraints, while soft real-time systems can tolerate occasional delays.