Task 3:
Real Time Programming

i)
The main differences between the Hard Real Time and Soft Real Time is in the requirement of the time limit or deadlines, for fullfilling the tasks and especially in the consequences of missing the deadlines.

Hard real-time:
- the tasks must be completed in the timelimit
  --> otherwise the failure will lead the system to a crash, which can be catastrophic
    --> consequences: missing deadline is unacceptable and will have critical consequences
--> will be used in medcial devices or safety-critical applications like airplane

Soft real-time:
- its desirable to complete the task in the timelimit
  --> but missing the deadline will not crash the total system
    --> consequences: missing the deadline is not ideal but not critical and will cause
      only minor issues
--> will be used in online games, video streaming and so on

ii)
Steps for configuring real time settings in ROS2:

1) Choose Real - Time Operating Systems that operates in real time:
   - Linux Kernel with PREEMPT-RT patches
   - Xenomai, a POSIX-compliant co-kernel that provides a real-time kernel alongside the
     linux kernel
   - QNX Neutrino: A POSIX compliant RTOS for mission-critical sytems

2) DDS Qos Settings
DDS can be configured for real-time performance by adjusting QoS poilicies
   - Deadline: specifies the maximum expected time betwenn two messages
   - Reliability: adjust message delivery
   - Static DDS API: A specially optimized version of the standard DDS API that provides static type safety and is
     designed for embedded systems with stringent real-time requirements. It offers reduced overhead and deterministic
     behavior, making it ideal for hard real-time applications.

3) Memory Management:
   - prevent page fault to ensure real time programming
     --> page fault takes to much time because of loading the missing page into RAM
   - Lock Memory
     --> locking the virtual address space into RAM
   - Allocate Dynamic Memory Pool
     --> to avoiding dynamic memorry allocation during realtime operation

4) Threading and Scheduling.
- Use Real-Time Scheduling Policies: Configure the system to use real-time scheduling policies (e.g., SCHED_FIFO, SCHED_RR)
to ensure that critical tasks get prioritized CPU time.
- Thread Priorities: Assign priorities to threads to mark the importance and timing requirements of each task.
- Affinity Settings: Set CPU affinity for threads to ensure they run on specific cores, reducing context switching and
improving predictability.

5) Testing and Benchmark:
- using tools like "cyclictest" to measure scheduling jitter to ensure that the system fits to the real time requirements

(
7) ROS2 specific strategies:
- create configuration option for the stack to operate in "real-time friendly" mode
--> to swtich between real time and non-real-time mode
--> useful if hard-real-time is only partly necessary or for resource management

- real-time specific libraries like rclrt, rmwrt, etc will used to implement a new real-time stack for real time computing

- make a configuration option available for real-time safety up to a certain point in the software stack and
implement a wrapper to provide real-time communication between the different level of the stack and the different programming languages
)