Real Time Operating Systems
2021/2022

Tutorial 3: FreeRTOS introduction

1: The call used for Assignment [A2] and a short justification why it performs better than the original one.

The call used was *vTaskUntilDelay()*. While *vTaskDelay()* specifies a time at which the task wishes to unblock *relative* to the time at which *vTaskDelay()* is called, *vTaskDelayUntil()* specifies an *absolute* time at which the task wishes to unblock.

Since the task may take a different path through the code between each call, or get interrupted a different number of times when it executes, it's hard to use *vTaskDelay()* to get a consistent execution frequency.

2: A brief description of the IPC mechanism considered in Assignment [A3] and a justification for the choice made with respect to the other possibilities.

RTOS Task Notifications was the mechanism used to achieve synchronization between the tasks. Simplifying, each RTOS task contains *task notifications*, which can have either the 'pending' or 'not pending' state. Events are sent directly to each task, rather than indirectly via intermediary objects like semaphores, making it inherently faster. Just as tasks can block on intermediary objects like semaphores, tasks can also block on a *task notification* to wait for that notification's state to become pending.

Semaphores and task notifications were both simple to implement, but as the documentation refers, task notifications provide a lightweight alternative to binary semaphores in many situations, unblocking an RTOS task with a direct notification is 45% faster and uses less RAM than unblocking a task using a binary semaphore.

David Rocha nº 84807
Tiago Adónis nº 88896