Manon THOMAS 5TS

20/01/2025

Real-Time Embedded Systems

FINAL ASSIGNMENT

Part 1 : What is the optimal schedule ?

I worked in Python. The code takes a list of tasks, and with the *permutations()* function from the itertools library, I generate all possible schedules. Then, I pass them one by one into a function that checks if they are schedulable. This function also returns the total waiting time between tasks. In the end, I only keep the schedulable schedules with the smallest total waiting time (there are multiple schedules with the same waiting time).

I verified that the code worked with 4 tasks (hyperperiod 20). However, when testing with 5 or more tasks (with a hyperperiod larger than 20), the code runed for too long and eventually threw a memory error. I think that my computer cannot handle it because my code is probably not optimized enough.

Part 2 : Implementation of an RTOs with 5 tasks

Task 1 : Print “Working.”

Task 2 : Convert a fixed temperature in Fahrenheit into Celsius.
I used the formula: Celsius = (Fahrenheit - 32) / 1.8.

Task 3: Multiply two long integer numbers.
I chose 123456789 × 987654321, and the result is -67153019, but I expected 121932631112635269. This strange result comes from an overflow. The excess bits are lost, so the result (in signed integers) is incorrect.

Task 4: Find a number in a list of 50 elements.
I created an array (simpler in C) of 50 elements from 1 to 50. I defined an element to find. I calculated the middle of the list. The middle is an integer. Its value comes from a division by two, so it may be a float, but it will be truncated (e.g., 25.5 becomes 25). This middle value is recalculated if the number sought is larger or smaller, until it is found.
In C, indexing starts at 0 (0, 1, ...), but my list starts at 1. So, for example, the number 6 is at position 5.

Task 5: Reset when you press key 1 on your computer.
When I created my code in C, I used the functions *kbhit()* to check if a key was pressed and *getch()* to read the pressed key. The code works fine when compiled in Visual Studio, for example, but not at all with FreeRTOS. This is because the *kbhit()* function is not compatible with FreeRTOS. This function runs in an infinite loop, whereas FreeRTOS is based on tasks that are scheduled and executed by its scheduler. I couldn't find an alternative to make this task work. I left the code in the file, but I commented it out so it wouldn't block the rest.