THIBAULT Clément
GAUDIN Matéo

26 avril 2024

# Système d'information en temps réel : Tâche finale

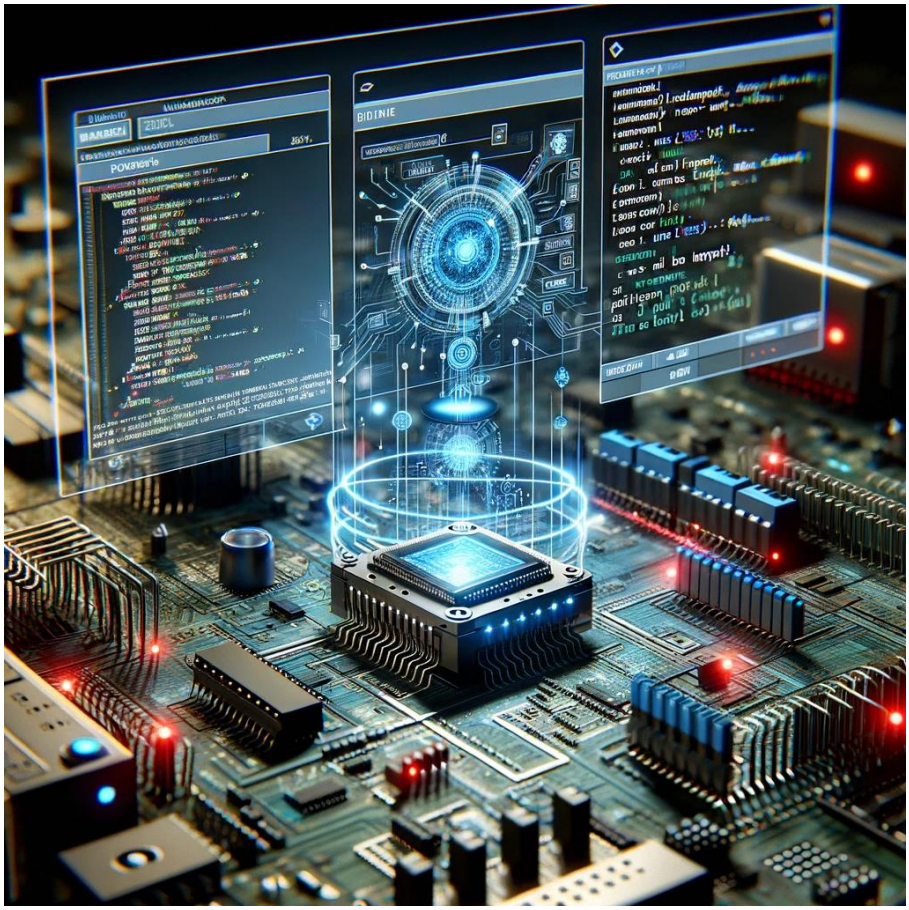


*Figure 1 : Représentation de système d'information temps réel généré par l'IA DALLE-3*

Professeur : M. Singh JASDEEP

Année 2024

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

# Table des matières

# Introduction

Dans le cadre de notre cours et de nos TDs de système d'information en temps réel à l'IPSA, nous avons pour projet d'en concevoir un, avec 4 tâches. Ceci pour mieux en comprendre le fonctionnement en le mettant en pratique.

Nous analyserons premièrement leurs temps d'exécution maximaux sur de nombreux tests pour ensuite déterminer leurs temps d'exécution maximale théorique (une valeur de temps que la tâche ne devrait jamais dépasser). Après cela nous pourrons imaginer un ordonnancement en priorité fixe des tâches. Premièrement nous regardons si l'utilisation du processeur est bien inférieure à 100% puis nous vérifions à la main que l'ordonnancement est bien faisable, c'est-à-dire que les tâches s'exécutent avant leurs deadlines. Nous choisirons un ordonnancement non préemptif, l'objectif étant d'optimiser le temps d'exécution.

Après quoi nous implémenterons notre système d'information en temps réel sur FreeRTOS pour vérifier qu'il est bien fonctionnel. Les fichiers que nous avons modifiés pour exécuter ce code sont joints avec ce rapport.

# Méthode

Détaillons dans cette partie comment nous avons déterminé l'ordonnancement à partir des 4 tâches suivantes :

- Tâche 1 : Afficher « Working » afin de vérifier que l'exécution se déroule normalement.
- Tâche 2 : Convertir une température de Fahrenheit à degré Celsius
- Tâche 3 : Définir deux grands entiers et les multiplier entre eux pour finalement afficher le résultat
- Tâche 4 : Faire une recherche « binaire » sur une liste de 50 éléments

La première chose est de déterminer le WCET (Worst Case Execution Time) noté C de chaque tâche. Pour nous faire une idée du temps maximal que pourrait prendre l'exécution d'une tâche, nous les avons exécutés 10000 fois en calculant à chaque fois leurs temps d'exécution et en sélectionnant finalement le plus grand. Après quoi nous avons multiplié ces valeurs par 1,2 afin d'être sûrs que cette durée ne soit jamais dépassée. Voici le résumé de ces résultats :

| Tâche | Après les tests (ms) | WCET (ms) |
|---|---|---|
| $\tau_1$ | 15 | 18 |
| $\tau_2$ | 17 | 20 |
| $\tau_3$ | 19.6 | 24 |
| $\tau_4$ | 1.6 | 2 |

À présent, choisissons l'ordre de priorité des tâches. Nous avons choisi que la tâche 4 ait la plus grande priorité, elle a un court temps d'exécution et ne gênera pas trop les autres tâches. Ensuite les tâches 2 et 3 sont relativement similaires en termes de durées, donc on leur associe la deuxième et troisième plus grande priorité. Finalement, en ce qui concerne l'affichage de la fonction « printworking », nous considérons que cette tâche est la moins urgente et moins importante. L'ordre des priorités étant de notre ressort, nous avons décidé de cette manière, mais elle n'était pas décisive, le plus important étant les périodes choisies pour chaque fonction.

En ce qui concerne les périodes d'exécution notées T, nous les avons choisies en fonction de trois critères. Le premier est que nous nous permettons d'exécuter beaucoup de fois la tâche 1, cela représente un certain défi, car elle reviendra toutes les 25ms. Le second est que l'utilisation soit inférieure ou égale à 1 (sinon quoi l'ordonnancement ne pourrait pas fonctionner pour sûr). Nous voulons aussi qu'elle soit tout de même relativement proche de 1 afin d'exploiter le plus possible les ressources du processeur. Nous trouvons pour notre cas,

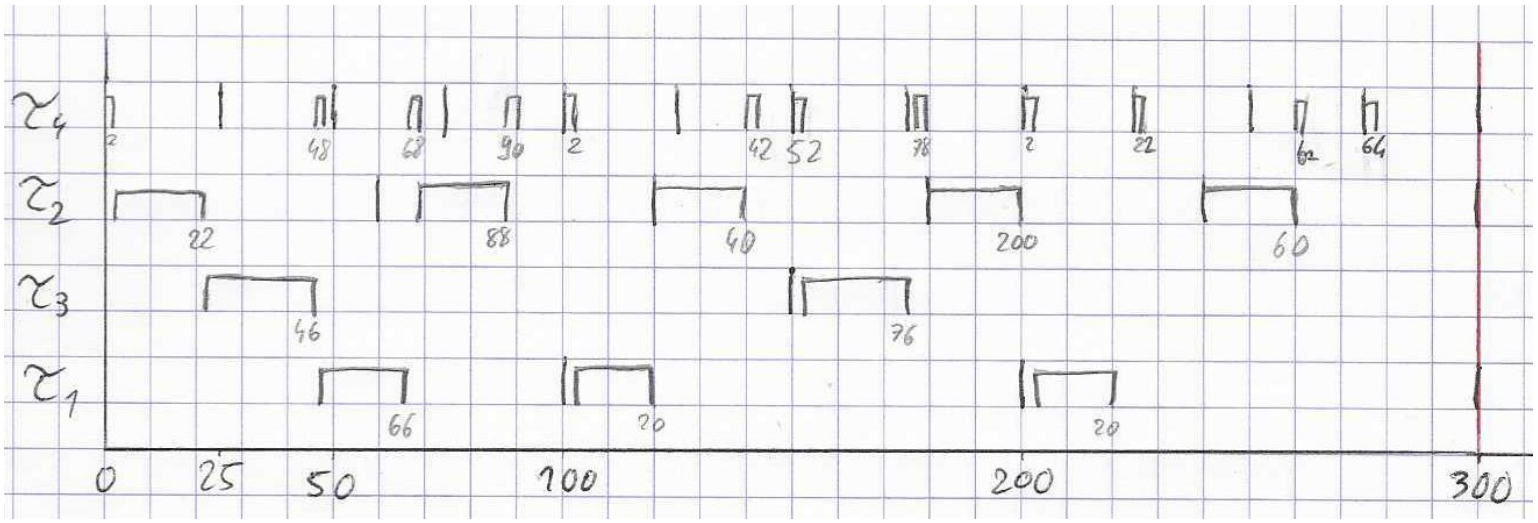
$$U = \sum_{i=1}^{4} \frac{c_i}{T_i} = \frac{2}{25} + \frac{20}{60} + \frac{24}{150} + \frac{18}{100} \approx 0{,}75$$

Le dernier critère est que nous ayons une hyperfréquence (la fréquence à laquelle le même motif théorique se répètera) relativement petite afin de bien pouvoir dessiner notre ordonnancement. C'est le plus petit multiplicateur commun, dans notre cas, c'est 300ms.

Ce critère étant défini, nous pouvons à présent nous occuper de l'ordonnancement de nos différentes tâches, en commençant par la création d'un tableau récapitulatif.

| Tâche | $C_i$ (ms) | $T_i$ (ms) | $P_i$ |
|---|---|---|---|
| $\tau_4$ | 2 | 25 | 0 |
| $\tau_2$ | 20 | 60 | 1 |
| $\tau_3$ | 24 | 150 | 2 |
| $\tau_1$ | 18 | 100 | 3 |

Avec ce nouveau tableau, nous pouvons montrer que l'ordonnancement est bien possible. Pour cela, nous avons tracé un schéma qui résume cela :




Nous avons représenté ici l'exécution des tâches au cours d'une seule hyperfréquence. La tâche 4, étant la plus prioritaire s'exécute en premier, suivie de la deuxième, de la troisième, puis la tâche 4 se réexécute avant la tâche 1, car elle est disponible et prioritaire. Aussi, elle ne s'exécute pas forcément directement une fois qu'elle est disponible, car l'ordonnancement n'est pas préemptif. Nous avons indiqué le temps total à la fin de l'exécution de chaque tâche et nous avons omis l'écriture des centaines volontairement, par souci de lisibilité.

On remarquera qu'il y a cinq instants auxquels aucune tâche n'est exécutée, sur vingt-deux exécutions au total. Cela nous semble cohérent, le processeur est relativement bien utilisé. Cela pourrait être optimisable d'une certaine manière. Nous avons, à période similaire à celles-ci, testés d'autres ordres de priorités, mais étaient moins efficace que celui-ci.

Nous retrouvons bien l'hyperfréquence à 300ms, pas de dépassement du WCET et une utilisation du processeur qui semble être optimisée aux alentours de ¾ (le mieux étant 1). À noter que trouver un ordonnancement non préemptif de 4 tâches ayant une grande utilisation n'est pas forcément facile, et certain tâches pourraient être coupées à de nombreuses reprises ce qui n'est pas forcément souhaitable, même si dans notre cas, nous ne sommes pas dans un système temps réel « hard ».

# Résultats

Nous avons implémenté toutes les fonctions en C dans notre fichier *ipsa_shed.c.* Dans ce même fichier, nous avons déclaré toutes nos priorités et exécuté nos tâches. Ce fichier est combiné à un fichier du même nom en *.h*.

Quand nous exécutons le programme avec *main.c,* qui était donné, nous observons que le système semble bel et bien fonctionner comme nous l'avions montré sur notre graphique.

Chaque période de chaque tache semble respectée, nous avons compté le nombre de tâches sur les 300ms d'hyperpériode, et on retombe bien sur 22 exécutions, ce qui semble cohérent avec le fait que le système est coordonné. Les WCET étant calculés pour que les tâches ne débordent pas, il y a donc peu de risque que l'hyperpériode soit dépassée.

L'environnement que nous avons créé pour ce projet semble efficace, et stable espérons-le. Ce projet aura été très intéressant sur plusieurs plans, notamment sur la compréhension de l'ordonnancement.