1. On remarque que les tâches s’exécutent par ordre de priorité et non pas par leur ordre de création.
2. Le problème est que, malgré que la tâche 1 soit plus prioritaire que la tâche 2, à cause des tics d’attentes imposés après l’écriture de chaque caractère dans le buffer, les tâches écrivent dans le buffer l’une après l’autre et de manière désorganisée.

Comme le buffer d’écriture est partagé entre les 2 tâches, il faut donc ajouter un mutex pour protéger en écriture le buffer. Il faut que la tâche 1 finisse d’ajouter tous ses caractères dans le buffer puis celui-ci relâche le mutex, ensuite la tâche 2 commence et prend la main sur le mutex.

1. La taille minimum des files sont :

* Pour la file controller\_to\_tank\_A : il faut 24 cases mémoire, car le temps minimum possible entre chaque commande est de 5 ticks. Le temps maximum possible d’exécution d’une commande est de 33 ticks. Par conséquent, dans le pire des cas, au moment de la 1er commande et entre la 7eme-8eme commande envoyée par le contrôleur, un message va être lu par le tank A. Au moment de la deuxième lecture du tank A dans la file, il y a déjà 18 messages dans la file. Le tank A en consomme 3 pour exécuter la seconde commande. Ensuite, 9 autres message seront envoyés ce qui fera un totale de 24.
* Pour les files controller\_to\_tank\_mixer et controller\_to\_tank\_B, une taille de 1 est suffisante puisqu’elles servent de sémaphore.

1. On peut le faire en initialisant un sémaphore binaire à 1. En initialisant le sémaphore binaire à 1 on peut laisser passer une tâche pour la ressource et puis la bloquer une fois utilisée.

Ce laboratoire est une bonne mise en jambe pour comprendre le fonctionnement des différents mécanismes de synchronisation et aborder les premières fonctions de µC. Le temps attribué est largement suffisant et la difficulté est vraiment raisonnable.