Rapport Lab 3 : Message Queues

by Walid GHETTAS EISE-5

Le but de cette séance de travaux pratiques est d'apprendre à utiliser Message Queues. Tous les codes sont joints et classés en fonction des exercices.

Exercice 1

Dans ce 1er exercice il faut écrire un programme composé de 3 tâches (*task1*, *task2* et *task3*) qui font pivoter un message à travers 3 files d'attente de messages différentes (*Q1*, *Q2* et *Q3*). Le message est une variable de type *unsigned int*. Chaque tâche, lorsqu'elle reçoit un message, affiche le message, incrémente la valeur du message, attend 1 seconde et envoie le message à une autre tâche via une autre file d'attente de messages. Pour suivre la progression du programme, chaque tâche doit afficher dans le moniteur série de l'IDE d'Arduino le nom de la tâche qui reçoit le message et le contenu du message.

```
load:0x40080400,len:5816
entry 0x400806ac
Task 1 message -> 1
Task 2 message -> 2
Task 3 message -> 3
Task 1 message -> 4
Task 2 message -> 5
Task 3 message -> 6
Task 1 message -> 7
Task 2 message -> 8
Task 3 message -> 9
Task 1 message -> 10
Task 2 message -> 11
Task 3 message -> 12
Task 1 message -> 13
```

Résultat obtenu à l'exercice 1

Exercice 2

Dans ce 2ème exercice, il faut écrire un programme avec 2 tâches qui affichent sur le moniteur série la valeur du capteur de température et d'humidité (DHT11) en utilisant des files d'attente de messages. En l'absence de capteur à ma disposition, j'ai généré des valeurs aléatoires (mais tout de même réelles, entre 10 et 30 pour la température et 40 et 60 pour l'humidité). Dans la tâche 1 j'obtiens la température et l'humidité et j'envoie les valeurs à la deuxième tâche à l'aide de files d'attente de messages. Mon code effectue ce travail toutes les 2 secondes. Dans la tâche 2 donc je reçois les valeurs de la 1ère tâche et je les affiche dans la fenêtre du moniteur série. Dans la seconde partie, je suis allé afficher les valeurs de température et d'humidité sur Internet via la connexion Wi-Fi en utilisant la plate-forme Ubidots. Ainsi, j'ai tout d'abord, appris et essayé de connecter le

ESP32 sur un réseau WiFi. Pour le réseau Wi-Fi, j'ai partagé la connexion de mon téléphone. J'ai par la suite créé un compte sur la plateforme Ubidots pour envoyer des données à la plateforme Ubidots en suivant l'exemple donné dans l'énoncé. Enfin, j'ai créé une 3ème tâche qui reçoit les valeurs du capteur via une nouvelle file d'attente de messages, je l'envoie à la plateforme ubidots via une requête http (POST) puis je l'affiche sur un tableau de bord.

2020-12-07 15:56:16 +01:00	59.00	{}	
2020-12-07 15:56:15 +01:00	22.00	{}	
2020-12-07 15:56:14 +01:00	55.00	{}	
2020-12-07 15:56:13 +01:00	11.00	{}	
2020-12-07 15:56:12 +01:00	56.00	{}	
2020-12-07 15:56:11 +01:00	16.00	{}	
2020-12-07 15:56:10 +01:00	52.00	{}	
2020-12-07 15:56:09 +01:00	17.00	{}	
2020-12-07 15:56:08 +01:00	52.00	{}	

Résultat obtenu sur Ubidots

Graphe représentant les valeurs obtenues sur Ubidots

Dans cette dernière image, on peut voir une disparité plutôt nette entre les valeurs hautes et les valeurs basses. Les valeurs hautes correspondent aux valeurs de l'humidité tandis que les valeurs basses correspondent aux valeurs de la température.

Exercice 3

Cet exercice est partagé en 4 parties. Je vais prendre le soin d'expliquer chaque partie.

Part 1

J'ai d'abord initialisé la période d'interruption de la minuterie à 10 Hz en m'aidant des explications précédant les questions. Dans l'ISR de la minuterie, j'ai également incrémenté une variable de comptage à chaque exécution du programme d'interruption avant d'afficher ce nombre sur la console série. Pour finir cette partie j'ai allumé une LED lorsque je suis dans l'ISR. Cela signifie que j'ai dû allumer la LED lorsque j'entre dans l'ISR et l'éteindre lorsque je quitte l'ISR. Cela nous donne une idée du temps passé dans les ISR.

```
entry 0x400806ac
init OK
Nombre d'interruption : 1
Nombre d'interruption : 2
Nombre d'interruption : 3
Nombre d'interruption : 4
Nombre d'interruption : 5
Nombre d'interruption : 6
Nombre d'interruption : 7
Nombre d'interruption : 8
Nombre d'interruption : 9
Nombre d'interruption : 10
Nombre d'interruption : 11
```

Résultat obtenu à la fin de Part 1

Part 2

Dans cette 2ème partie d'exercice, j'ai créé une tâche qui fait clignoter une deuxième LED avec une fréquence de 5 Hz à l'aide de la fonction vTaskDelay. Le résultat de cette est observable dans la vidéo jointe au dossier.

Part3

Ici, j'ai intégré la fonction fibonacci donnée dans l'énoncé. Cette fonction a la propriété d'avoir un temps de calcul très différent selon la valeur du paramètre. Dans le timer ISR créé dans la 1ère tâche, j'ai ajouté le code pour appeler la fonction fibonacci avec la valeur du compteur comme indiqué dans le sujet. Ensuite, toujours dans le timer ISR, j'ai affiché le résultat dans le moniteur série. Sur le temps passé dans l'interruption, j'observe qu'il est de plus en plus long (observation de la LED intégré sur la carte). Cela est en accord avec ce que j'observe à propos du clignotement de la deuxième LED qui devrait normalement clignoter à 5 Hz. Elle est moins régulière et va jusqu'à disparaître.

```
init OK
Nombre d'interruption : 1
Résultat : res = 1
Nombre d'interruption : 2
Résultat : res = 1
Nombre d'interruption : 3
Résultat : res = 2
Nombre d'interruption : 4
Résultat : res = 3
Nombre d'interruption : 5
Résultat : res = 5
Nombre d'interruption : 6
Résultat : res = 8
Nombre d'interruption : 7
Résultat : res = 13
Nombre d'interruption : 8
Résultat : res = 21
Nombre d'interruption : 9
```

Résultat observable dans le moniteur série

Au fur et à mesure de la progression (augmentation du nombre d'interruptions), celle-ci prend de plus en plus de temps, jusqu'à atteindre une erreur à la 31ème interruption. Cela est dû au dépassement de capacité du type int : en effet, le bit de signe est le poids le plus élevé donc si en dépassant 2^{31}, on passe en nombre négatif ! Pour contourner ce problème il suffit de passer à un type 64 bits. Le ESP32 choisit la méthode radicale à savoir, le redémarrage.

```
Résultat : res = 317811
Nombre d'interruption : 29
Résultat : res = 514229
Nombre d'interruption : 30
Résultat : res = 832040
Nombre d'interruption : 31
Guru Meditation Error: Core  1 panic'ed (Interrupt wdt timeout on CPU1)
Core 1 register dump:
PC      : 0x400e91dc  PS      : 0x00060034  A0      : 0x800e91e9  A1      : 0x3ffbe5b0
A2      : 0x00000003  A3      : 0x00000003  A4      : 0x00000000  A5      : 0x00000001
A6      : 0x00060621  A7      : 0x00000000  A8      : 0x800e91e9  A9      : 0x3ffbe590
A10     : 0x00000002  A11     : 0x00000003  A12     : 0x00000400  A13     : 0x3ffbc0e8
A14     : 0x00000000  A15     : 0x00000000  SAR     : 0x0000001e  EXCCAUSE: 0x00000006
EXCVADDR: 0x00000000  LBEG    : 0x400014fd  LEND    : 0x4000150d  LCOUNT  : 0xffffffff
Core 1 was running in ISR context:
EPC1    : 0x400e91f3  EPC2    : 0x00000000  EPC3    : 0x00000000  EPC4    : 0x400e91dc

Backtrace: 0x400e91dc:0x3ffbe5b0 0x400e91e6:0x3ffbe5d0 0x400e91e6:0x3ffbe5f0 0x400e91e6:0x3ffbe61

Core 0 register dump:
PC      : 0x400e92ce  PS      : 0x00060634  A0      : 0x800d40c2  A1      : 0x3ffbbff0
A2      : 0x00000000  A3      : 0x00000001  A4      : 0x00000000  A5      : 0x00000001
A6      : 0x00060120  A7      : 0x00000000  A8      : 0x800d3c8a  A9      : 0x3ffbbfc0
A10     : 0x00000000  A11     : 0x40084e80  A12     : 0x00060120  A13     : 0x00000020
A14     : 0x00000020  A15     : 0x00000000  SAR     : 0x00000000  EXCCAUSE: 0x00000006
EXCVADDR: 0x00000000  LBEG    : 0x00000000  LEND    : 0x00000000  LCOUNT  : 0x00000000

Backtrace: 0x400e92ce:0x3ffbbff0 0x400d40bf:0x3ffbc010 0x4008980e:0x3ffbc030 0x4008831d:0x3ffbc05

Rebooting...
```

Observation du dépassement de capacité suivi du redémarrage

Part 4

Dans cette dernière partie du dernier exercice, j'utilise une nouvelle file d'attente de messages. Je déplace alors le calcul et l'affichage obtenu par la fonction de fibonacci dans une nouvelle tâche. Ainsi, je commence par créer une nouvelle tâche avant de créer une file d'attente de messages. J'envoie ensuite la valeur du nombre d'interruption à la nouvelle tâche via la file d'attente de messages avant de déplacer le calcul et l'affichage du résultat de fibonacci dans la nouvelle tâche.

```
init OKIci interruption : Nombre d'interruption --> 1
Ici task2 : Nombre d'interruption --> 1
Résultat : res = 1
Ici interruption : Nombre d'interruption --> 2
Ici task2 : Nombre d'interruption --> 2
Résultat : res = 1
Ici interruption : Nombre d'interruption --> 3
Ici task2 : Nombre d'interruption --> 3
Résultat : res = 2
Ici interruption : Nombre d'interruption --> 4
Ici task2 : Nombre d'interruption --> 4
Résultat : res = 3
Ici interruption : Nombre d'interruption --> 5
Ici task2 : Nombre d'interruption --> 5
Résultat : res = 5
Ici interruption : Nombre d'interruption --> 6
Ici task2 : Nombre d'interruption --> 6
Résultat : res = 8
Ici interruption : Nombre d'interruption --> 7
Ici task2 : Nombre d'interruption --> 7
Résultat : res = 13
Ici interruption : Nombre d'interruption --> 8
Ici task2 : Nombre d'interruption --> 8
Résultat : res = 21
Ici interruption : Nombre d'interruption --> 9
Ici task2 : Nombre d'interruption --> 9
Résultat : res = 34
```

Résultat observable dans le moniteur série

On peut comparer les résultats de fibonacci et les nombres d'interruptions associés à la 1ère image de la partie précédente. On remarque qu'ils sont identiques.

FIN