Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Systèmes Embarqués

Examen — janvier 2023

Durée : 1h30 — Documents autorisés

1 – 5pts Soit le programme suivant :

```
 1 static SemaphoreHandle_t lock;
 2 TickType_t cs_wait = 250;
 3 TickType_t med_wait = 5000;
 4
 5 void doTaskL(void *parameters) {
 6   TickType_t timestamp;
 7   while (1) {
 8     timestamp = xTaskGetTickCount() * portTICK_PERIOD_MS;
 9     xSemaphoreTake(lock, portMAX_DELAY);
10     timestamp = xTaskGetTickCount() * portTICK_PERIOD_MS;
11     while ( (xTaskGetTickCount() * portTICK_PERIOD_MS) - timestamp < cs_wait);
12     xSemaphoreGive(lock);
13     vTaskDelay(500 / portTICK_PERIOD_MS);
14   }
15 }
16 void doTaskM(void *parameters) {
17   TickType_t timestamp;
18   while (1) {
19     timestamp = xTaskGetTickCount() * portTICK_PERIOD_MS;
20     while ( (xTaskGetTickCount() * portTICK_PERIOD_MS) - timestamp < med_wait);
21     vTaskDelay(500 / portTICK_PERIOD_MS);
22   }
23 }
24 void doTaskH(void *parameters) {
25   TickType_t timestamp;
26   while (1) {
27     timestamp = xTaskGetTickCount() * portTICK_PERIOD_MS;
28     xSemaphoreTake(lock, portMAX_DELAY);
29     timestamp = xTaskGetTickCount() * portTICK_PERIOD_MS;
30     while ( (xTaskGetTickCount() * portTICK_PERIOD_MS) - timestamp < cs_wait);
31     xSemaphoreGive(lock);
32     vTaskDelay(500 / portTICK_PERIOD_MS);
33   }
34 }
35 void setup() {
36   lock = xSemaphoreCreateBinary();
37   xSemaphoreGive(lock);
38   xTaskCreate(doTaskL,"Task L",1024,NULL,1,NULL);
39   vTaskDelay(1 / portTICK_PERIOD_MS);
40   xTaskCreate(doTaskH,"Task H",1024,NULL,3,NULL);
41   xTaskCreate(doTaskM,"Task M",1024,NULL,2,NULL);
42   // Delete "setup and loop" task
43   vTaskDelete(NULL);
44 }
45 void loop() {
46 }
```

a. À quoi sert la ligne 37 ? *(1pt)*

b. Quelle différence entre un `vTaskDelay` et l'utilisation d'un timestamp dans une tâche ? *(1pt)*

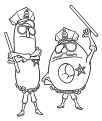
c. Décrivez comment va se dérouler l'exécution des différentes tâches ? *(2pt)*

d. Est-ce normal ? *(1pt)*

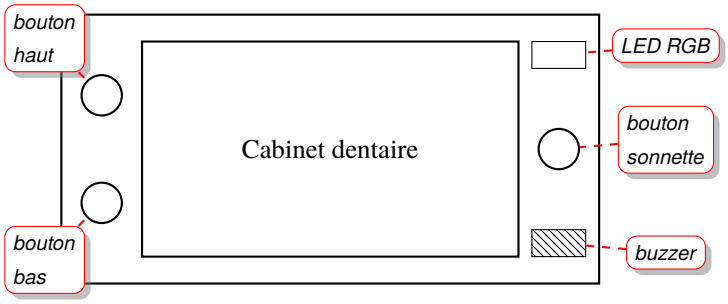
2 – 2pts

a. En programmation assembleur, est-ce que les registres sont utilisés de manière différente en « *baremetal* » par rapport à Linux et pourquoi ? *(1pt)*

b. Pourquoi dans le linker, on fait une différence entre `lma` et `vma` ? *(1pt)*



3 – 13pts Soit l'interphone d'immeuble suivant :



Il dispose de :

- ☐ un afficheur 1 ligne ;
- ☐ 3 boutons connectés aux PINs 3,4,5 ;
- ☐ une LED RGB connectée à la PIN 7 ;
- ☐ un buzzer ;
- ☐ un relais d'ouverture porte sur la PIN 8.

Le fonctionnement est le suivant :

- ▷ lorsque l'interphone n'est pas utilisé, l'afficheur est éteint ;
- ▷ dès que l'on appui un des boutons « bas » et « haut », l'afficheur s'allume et affiche les noms des occupants de l'immeuble :
  - ◇ on affiche le nom du premier occupant de la liste des occupants ;
  - ◇ on appuie sur les boutons « haut » et « bas » pour faire défiler les noms dans la liste ;
  - ◇ si on va trop haut, c-à-d qu'on affiche déjà le premier nom de la liste : la LED RGB s'éclaire en rouge et le buzzer se déclenche une fois pendant 1s ;
  - ◇ si on va trop bas, c-à-d qu'on affiche déjà le dernier nom de la liste : la LED RGB s'éclaire en rouge et le buzzer se déclenche une fois pendant 1s ;
- ▷ lorsque l'on appui sur la sonnette :
  - ◇ si l'afficheur est éteint, rien ne se passe ;
  - ◇ sinon :
    - ⋆ on appelle l'occupant dont le nom est affiché ;
    - ⋆ on déclenche le buzzer en cycle 1s actif, 500ms non actif répété 3 fois ;
    - ⋆ on allume la LED en vert ;
    - ⋆ suivant la valeur de retour de la fonction d'appel :
      - ▷ si c'est `true` la LED reste verte et on ouvre la porte (niveau 1 sur la PIN associée au relais) ;
      - ▷ si c'est `faux` on mets la LED en rouge pendant 2s et on éteint l'interphone.

L'afficheur est utilisable avec les fonctions suivantes :

`void afficheur_switch(bool etat);`	si `etat=true` alors l'afficheur est allumé, et éteint sinon
`void afficheur_texte(char *texte);`	affiche le `texte` d'au plus 32 caractères

Le buzzer :

`void buzzer_bip(bool etat);`	si `etat=true` le buzzer *bip*, sinon il ne fait rien

La LED RGB :

`void LED_RGB(uint8_t rouge,uint8_t vert,uint8_t bleu);`	affiche la valeur demandée, la valeur (0,0,0) éteint la LED

L'appel vers l'occupant sélectionné :

`bool appel_occupant(uint8_t numero);`	déclenche une sonnerie dans l'appartement identifié par le rang dans la liste des noms et retourne un booléen si l'occupant ouvre ou non la porte

La liste des occupants est dans un tableau de `char` :

Monsieur Durand	Madame Dupont	Cabinet dentiste	…

Où chaque entrée correspond à 32 caractères complétés avec des espaces.

a. Décrivez comment les boutons « haut » et « bas » vont être gérés ? *(2pts)*
Comment va-t-on faire le lien entre le bouton, la gestion de la liste des occupants et le buzzer ?
Quels éléments de FreeRTOS allez vous utiliser ?
Est-ce que le fait d'appuyer sur les deux boutons simultanément est problèmatique ?

b. Comment peut-on faire pour gérer les différents temps d'attente de manière efficace (LED, buzzer) ? *(1pt)*

c. Comment peut-on gérer le temps d'attente variable et humain de la fonction d'appel ? *(1pt)*

d. Donnez le code réalisant le travail de l'interphone en utilisant au mieux les fonctionnalités offertes par FreeRTOS. *(7pts)*

e. On veut mettre un système anti-gêneur qui impose d'appuyer au moins $2s$ sur le bouton « sonnette » avant de déclencher l'appel vers l'appartement concerné. *(2pts)*
Comment peut-on mettre au point ce système dans votre code ?