Mutex/semáforos

Objectivo

- Estudo dos mecanismos de exclusão mútua e sincronização no FreeRTOS:
  - *Mutexes*;
  - Semáforos binários;
  - Semáforos de contagem;

Exercícios propostos

Mutexes

1. Substitua no projecto 3SO_FreeRTOS a main.c pela disponível no Moodle/Teams main_mutex.c. Resolva o problema das mensagens misturadas garantindo que o acesso à USART2 de cada tarefa é protegido por um *mutex*.
   Comente o mecanismo utilizado e o resultado obtido.

Semáforos binários

Nota: Do exercício anterior manter apenas a correr a tarefa que mostra actividade no *display*. As outras duas podem ser comentadas.

2. Criar duas novas tarefas com prioridades iguas mas superiores à da tarefa que apresenta atividade no *display*. Uma das tarefas é responsável por sinalizar, recorrendo a um semáforo binário (*xSemaphoreGive*), sempre que o botão de pressão GPIOA1 for pressionado (usar a técnica de *pooling* do GPIO). A segunda tarefa deverá mudar o estado do LED ligado ao GPIOB1 quando o semáforo é libertado (*xSemaphoreTake*).

   Fazer *pooling* de um botão de pressão (esperar que seja pressionado) da seguinte forma:

   ```
   while( GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1) == SET );
   ```

   O funcionamento pretendido é o correto? O que se verifica com a tarefa que vai apresentanto atividade no *display* e cuja prioridade é a mais baixa?

Semáforos binários com interrupções

3. Como vimos anteriormente, a técnica de *pooling* não deve ser usada num RTOS. Alterar a resolução do problema da pergunta anterior usando agora uma interrupção externa para o GPIOA1 (*EXTI1_IRQHandler*) em vez do *pooling* do GPIO para libertar o semáforo.

   Exemplo genérico para libertar um semáforo a partir de uma interrupção:

```
void GENERIC_IRQHandler ( void )
{
static BaseType_t pxHigherPriorityTaskWoken;

xSemaphoreGiveFromISR( xSemaphore_signal, &pxHigherPriorityTaskWoken );
if( pxHigherPriorityTaskWoken == pdTRUE )
        taskYIELD(); /* forces the context change */
GENERIC_ClearITPendingBit ( GENERIC_bit );
}
```

4. Mantendo a funcionalidade da pergunta anterior, adicione uma nova funcionalidade que permita que o LED também mude de estado sempre que for recebido um caráter na USART2.

Semáforos de contagem

5. Criar uma aplicação com duas tarefas que simule um sistema de contagem de eventos recorrendo a um botão da placa de desenvolvimento. Sempre que o botão for pressionado deve ser incrementado um semáforo de contagem dentro de uma tarefa. A outra tarefa, que atende os eventos de forma automática, consegue processá-los a cada 10 segundos. Este sistema só pode atingir um valor máximo de 10 (uxMaxCount=10). Sempre que houver disponibilidade de adicionar novos eventos, ligar o LED verde (RGB), caso contrário deve permanecer aceso o LED vermelho.

   Para resolver este problema é necessário ter em conta os seguintes pontos:

- Criar um semaforo de contágem com um valor máximo de 10.
- Criar de duas tarefas:
  - Uma tarefa é responsável por processar os eventos, ou seja, é responsável por chamar a função *xSemaphoreTake* do semáforo de contagem, esperar os 10 segundos (simular esta funcionalidade usando a função vTaskDelay) e repetir este processo;
  - A outra tarefa é responsável por receber um sinal de um semáforo binário libertado a partir da interrupção externa correspondente ao botão de pressão, e após receber esse sinal, chamar a função *xSemaphoreGive* do semáforo de contagem;
- Atuar nos LED em ambas as tarefas mediante a especificação do problema. Como o acesso ao dos LED é realizado por tarefas diferentes, não esqueçer de fazer a devida proteção com um *mutex*.

API FreeRTOS

6. Adicione à pergunta anterior uma tarefa que atualiza o display a cada 100 ms com informação sobre o estado do semáforo de contagem (uxSemaphoreGetCount()).