GUÍA DE EJERCICIOS N° 11
RTOS

1. Estudiar en profundidad el código de `ucosiii_labo_de_micros_project`.
    a. ¿Cuántos threads hay corriendo, luego de la inicialización?
    b. ¿Qué tarea hacen esos threads?
    c. ¿Qué hacen los threads la mayor parte del tiempo?
    d. ¿Qué thread usa la CPU la mayor parte del tiempo (~99%)?

2. Estudiar en profundidad la estructura del proyecto `ucosiii_labo_de_micros_project`.
    a. Observar el código en `source\rtos\uCOSIII\src\uCOS-III\Ports\ARM-Cortex-M4\Generic\GNU\os_cpu_c.c`. ¿Qué timer de hardware se usa para el tick() del RTOS? ¿Puede usarse ese timer para los drivers?
    b. Observar el header `source\ucosiii_config\os_cfg_app.h`. ¿Qué frecuencia tiene el tick() del RTOS?
    c. Intentar entender las configuraciones en los headers que están en `source\ucosiii_config`, usando el manual del RTOS: (https://doc.micrium.com/display/kernel304/uC-OS-III+Documentation+Home)

3. Abrir la documentación de uC/OS III y leer el capítulo "*Getting Started with uC/OS III*".

4. Crear un nuevo Thread (Thread3) que utilice el LED azul, titilando cada 100ms, usando `OSTimeDly` en lugar de `OSTimeDlyHMSM`. ¿Qué diferencia hay entre las dos?

5. En lugar de usar un delay para hacer titilar el LED azul, usar el semáforo (`semExample`) para hacer que el Thread2 le mande una señal (OSSemPost) al Thread3, cada 500ms. El Thread3 quedará suspendido (OSSemPend) hasta recibir la señal, luego hará un toggle del LED, y volverá a dormir.

6. En lugar de usar un semáforo para hacer titilar el LED azul, implementar una MessageQueue para hacer que el Thread2 le mande un mensaje (OSQPost) al Thread3, cada 500ms. El Thread 2 le indicará el color del LED a togglear, mediante el mensaje. El Thread3 quedará suspendido (OSQPend) hasta recibir el mensaje, luego hará un toggle del LED que corresponda, y volverá a dormir.

7. Conectar la placa del TP1. Identificar las modificaciones necesarias para integrar el código del TP1 en el proyecto:
    a. Si el TP1 usaba SysTick, ¿a qué frecuencia?
    b. ¿Se puede "hookear" el tick() que usa el RTOS?
    c. Abrir SDK\startup\startup_mk64f12.c.
        i. Identificar la tabla de vectores de interrupción. ¿Qué código se ejecuta inmediatamente luego del reset?
        ii. Seguir la secuencia (sin saltear pasos) desde el reset hasta el main.
        iii. ¿El main arranca con las interrupciones habilitadas o deshabilitadas?
    d. Copiar el código del main() del TP1 al ThreadStart del ejemplo. Quitar la inicialización del Thread2 (OSTaskCreate(&Task2TCB, …)). ¿El TP1 funciona?
    e. Si no funciona, hacer que funcione antes de seguir.

8. Delinear un plan para:
    a. Modificar el driver de la tarjeta magnética para que señalice, mediante un semáforo (OSSemPost) cuando hay un ID de tarjeta leído.
    b. Modificar el driver del encoder para que señalice, mediante un semáforo (OSSemPost) cuando hay un evento de encoder.
    c. Modificar el main para evitar pollear al driver de la tarjeta y del encoder, y en su lugar usar OSPendMulti().
    d. Re-activar el Thread2 e implementar una MessageQueue.
    e. Postear en la MessageQueue un mensaje, cada vez que hay un ID leído.
    f. En el Thread2, utilizar OSQPend() para esperar un mensaje, y togglear un LED cada vez que se reciba uno.