Evaluación final de Sistemas Operativos en Tiempo Real I

Docentes:

- Mg. Ing. Franco Bucafusco > franco_bucafusco@yahoo.com.ar
- Mg. Ing. Martín Menéndez > mmenendez@fi.uba.ar

Consideraciones:

- La resolución del examen es individual.
- Se deberá adjuntar la carpeta src inc y el config.mk en un archivo comprimido .rar o .zip y enviarlo por correo con copia a ambos docentes.
- El examen comienza a las 19.00hs del día 23 de Abril de 2021 y **finaliza 3hs después**. Se aceptan entregas hasta la medianoche, pero con penalidad.
- El examen se puede recuperar una semana después, mediante la misma modalidad.

Se evalúa:

- Administración y diseño de las **tareas**:
- **Modularización** del sistema:
    - Separar correctamente los archivos.
    - Utilizar headers para cada archivo.
    - Utilizar **variables globales** solamente si es necesario.
- **Prolijidad** del código:
    - **Comentar** lo más posible el código.
    - NO dejar código comentado.
    - NO dejar **números mágicos**.
- No dejar cosas **inicializadas** sin verificar.
- Uso de **interrupciones** es altamente recomendable para una buena calificación.
- Uso de colas/semáforos cuando corresponda.
- Protección de zonas críticas.
- Plataforma recomendada: EDU-CIAA (Fig. 1). En el caso de utilizar otra plataforma deberán enviar un video demostrativo con plazo máximo de 5 hs después de iniciado el examen.

Recomendaciones:

- Piense la lógica en un papel primero, incluyendo los elementos de sincronización que va a utilizar entre tareas.
- Use el template para avanzar más rápido. Si no, se recomienda usar el ejercicio F3 resuelto. del repositorio.

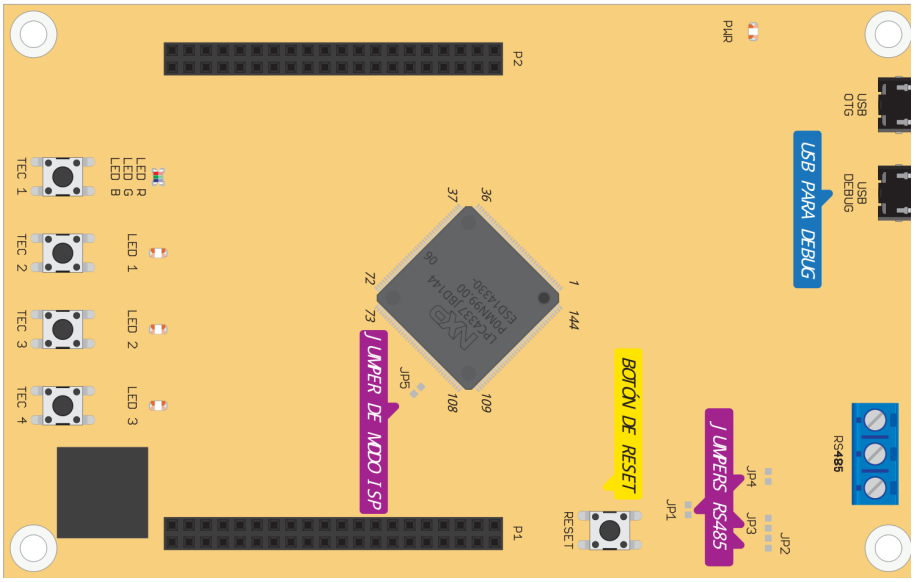

Figura 1. Plataforma recomendada

Antes de empezar:

- Actualizar el repositorio local con los cambios en master de RTOS_13CO
- Vea este video para entender el funcionamiento LINK

Reglas del juego:

- El juego genera una secuencia de leds de hasta N de elementos.
- El juego iniciará con una secuencia de 1 led al pulsar cualquier tecla.
- En cada ronda, el jugador deberá replicar la secuencia de leds pulsando las teclas correspondientes (Fig. 2a).
- En caso de acertar la secuencia, se adicionará un nuevo elemento a la secuencia anterior.
- La secuencia será mostrada al usuario mediante los leds, para ser replicada por el usuario.
- Si el usuario no acierta en alguno de los elementos de la secuencia, perderá el juego (Fig. 2b).
- Si el usuario no presiona ninguna tecla durante un tiempo , perderá el juego (Fig. 2c).

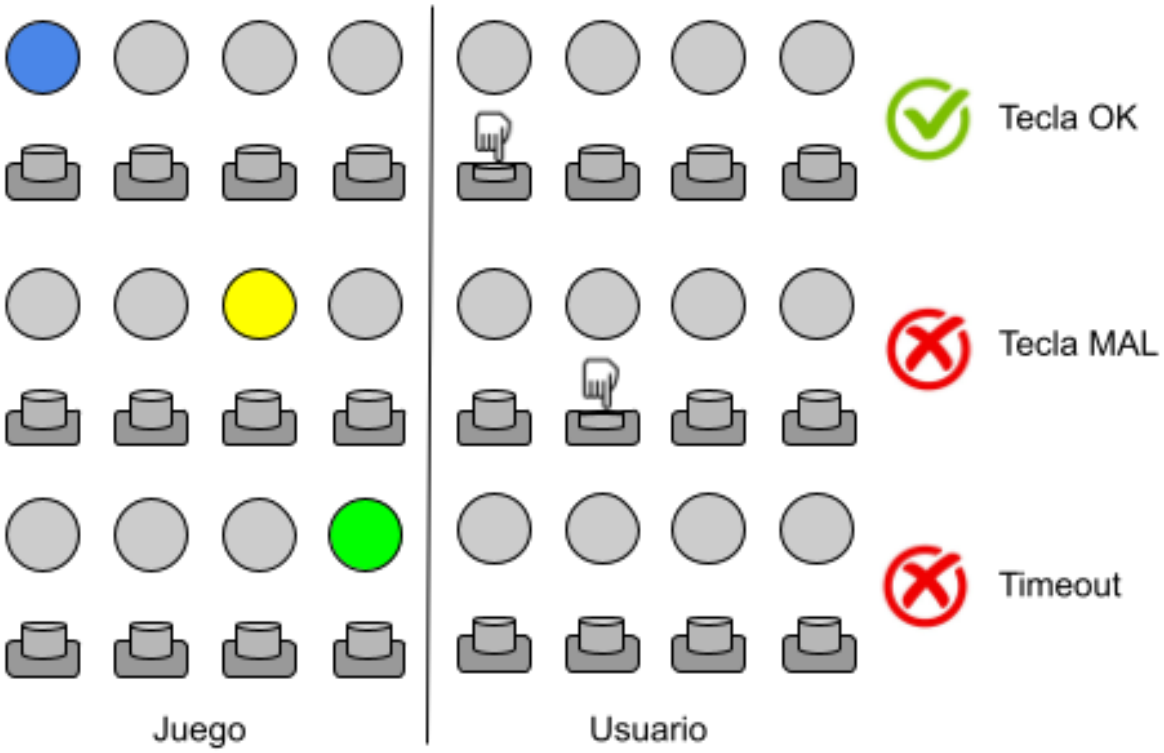

Figura 2. Algunos casos de uso

Por ejemplo:

1. Ronda 1: Juego: ejecuta la secuencia [LED1]
2. Ronda 1: Usuario: presiona TEC1
3. Ronda 2: Juego: ejecuta la secuencia [LED1 LED3]
4. Ronda 2: Usuario: presiona TEC1 y TEC3
5. Ronda 3: Juego: ejecuta la secuencia [LED1 LED3 LED2]
6. Ronda 3: Usuario: presiona TEC1, TEC3 y TEC2

Implementación con FreeRTOS:

- El kernel del juego deberá estar implementado con dos tareas.
- Tarea 1:
    - Se encargará de generar secuencias y validarlas.
    - El juego comienza con cualquier tecla pulsada, y termina con "game over" o timeout.
    - Generará las secuencias sucesivas.
    - Presentará las secuencias utilizando LEDS
    - Informará a la tarea 1 que la secuencia está lista para ser validada.
    - Esperará las entradas de usuario y las validará a medida que el usuario va ingresando.
    - Esperar la señal de timeout de tarea 2.
    - Generará mensajes a la UART
- Tarea 2:
    - Esperará las acciones del usuario.
    - Estará activa cuando haya una secuencia lista.
    - Si la secuencia no está activa, no debe hacer nada.
    - Esperará ingresos del usuario, y los informará a Tarea 1.
    - En cada elemento de la secuencia ingresado por el usuario, la Tarea 1 deberá validarla.
    - Si durante un tiempo el usuario no ingresa nada, se cancela el juego.
    - Generará mensajes a la UART.

Aclaraciones:

- Si bien no se espera una perfecta modularización, intentar encapsular todo lo referido al juego en simon.c /.h
- El alumno puede agregar tareas al kernel si así lo desea.
- La generación de numero random puede utilizarse random.c y random.h de RTOS_13CO (copiarse los archivos en el proyecto al examen)
- La generación de números random puede dejarse para lo último. Puede probarse inicialmente con secuencias preestablecidas.