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 de la clase 8. Se aceptan entregas hasta la medianoche de 3 días después.
- 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.

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 G3 resuelto. del repositorio.

Antes de empezar:

- Actualizar el repositorio local con los cambios en master de RTOS_15CO
- Vea este video para entender el funcionamiento LINK

Definiciones:

- Slot: mínima división de la pantalla a la cual atribuimos un “led”. La pelota solo puede ocupar un slot simultáneamente.
- Tiempo de saque (t_shoot): tiempo que demora la bola en lanzarse. Es decir, cuanto debo esperar para ver el primer led encenderse.. Esto solo ocurre con el primer led.
- Tiempo de avance (t_step): tiempo que pasará la bola en cada slot.
- Posición de golpe (chance): último slot disponible.

Reglas del juego:

El juego pretende simular un lanzador de pelotas para practicar algún deporte con raqueta o bate.
El efecto del avance de la pelota se realizará encendiendo un led consecutivo a otro por cada paso de avance, arrancando por el de la izquierda y llegando hasta la derecha. El usuario deberá sincronizar la pulsación del botón justo cuando se encienda el último led. simulando el raquetazo.

- t_shoot es random entre 300ms y 800ms.
- t_step es random entre 100 ms y 300ms.
- Una vez elegidos, t_shoot y t_step deben ser constantes en cada pelota, solo deben actualizarse al lanzar una nueva pelota.
- Al llegar a la "posición de golpe", el usuario tendrá la chance de "golpear" a la pelota durante una ventana de tiempo (FIJA) de 300 ms.
- En caso de acertar el golpe, el contador de puntos sumará una cantidad de puntos provista por la api *ledtennis_points_hit()*. Esta función asignará mayor puntaje si el jugador golpeó la pelota más cerca del comienzo de la ventana y menos si lo hace cerca del timeout.
- En caso de errarle, se sumará una cantidad de puntos provista por la api *ledtennis_points_miss()*.
- En caso de recibir un raquetazo en un momento incorrecto (por ejemplo, antes de la posición de golpe), se sumará una cantidad de puntos provista por la api *ledtennis_points_invalid()*. Fallar antes de tiempo NO altera el recorrido de la pelota. Es decir, podes fallar cuantas veces quieras hasta que la pelota llegue a destino, restando cada vez más puntaje.

Implementación con FreeRTOS:

- El kernel del juego deberá estar implementado con dos tareas.
- Tarea 1:
    - Como las plataformas poseen pocos leds, utilizar la librería stdout_leds.c/h y con ella, simular 20 leds (slots) para el trayecto (pueden ser más).
    - Se encargará de generar secuencias y encender/apagar los leds.

    - El juego consta de una ronda de 30 pelotas.
    - El juego comienza con cualquier tecla pulsada, y termina con "game over", "you win".
    - Esperará las entradas de usuario provenientes de la tarea 2.
    - Generará mensajes a la UART con formato (dependiendo del contexto):
        - “Ball {number}: {HIT/MISS} | {score} points”
        - “Total: {#HIT / 30} | {score} points”
        - “Press any button to start”
    - Cada led debe estar encendido también un t random equivalente a t_step / 3 y apagado el resto del tiempo hasta completar el t_step.
- Tarea 2:
    - Esperará las acciones del usuario, y los informará a Tarea 1.
    - Si el juego no comenzó, no debe hacer nada.
    - Si durante un tiempo equivalente a 3 pelotas que recorren todos los leds con t_step máximos (ignorando t_shoot) el usuario no ingresa nada, se cancela el juego. Se enviará una señal de game over a la tarea 1, y el juego se cancelará. Es la ÚNICA condición de derrota.
    - Generará mensajes a la UART con formato:
        - “Wake up! Game over!”

Aclaraciones:

- Si bien no se espera una perfecta modularización, intentar encapsular todo lo referido al juego en ledTennis.c /.h
- El alumno puede agregar tareas al kernel si así lo desea.
- La generación de número random puede utilizar random.c y random.h.
- La generación de números random puede dejarse para lo último. Puede probarse inicialmente con secuencias preestablecidas.