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 **15 de octubre de 2021** y finaliza 72hs después, el lunes **18 de octubre de 2021**.
- En caso de no aprobar y tener que corregir algo, el plazo de entrega se extiende otras 72 hs(a modo de **recuperatorio**) hasta el **21 de octubre de 2021.** En este caso existe una penalidad en la nota.

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 . En el caso de utilizar otra plataforma deberán enviar un video demostrativo junto a la resolución.

Consigna:

Implementar un juego whack a mole con la EDUCIAA, donde los LEDS son los topos y las teclas representan el martillo del usuario.

Definiciones:

- “Topo”: led
- “Martillazo” : presionar la tecla frente al led correspondiente
- “Salir de su madriguera”: encender un led
- “Esconderse en su madriguera”: apagar un led

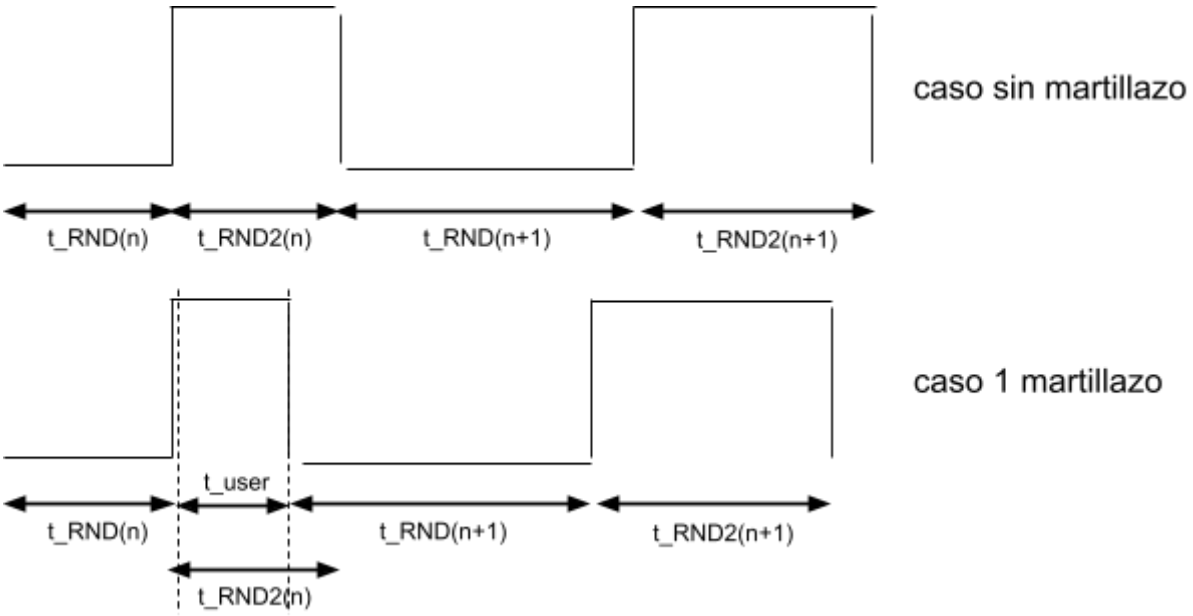
Reglas del juego (Vea este video para entender el funcionamiento LINK):

- Cada topo "sale de su madriguera" durante un tiempo, y en tiempos aleatorios.
- Los tiempos de salida de cada topo son independientes, pueden coincidir o no.
- El jugador debe reaccionar a la aparición del topo lo más rápido posible dándole un martillazo.
- Cuanto más rápida es la reacción al pegarle, mayor será el puntaje.
- Si el topo no es golpeado durante su aparición, restará puntos.
- Si el topo se intenta golpear cuando sigue en su madriguera, también resta puntos.
- El juego termina luego de un cierto tiempo fijo de antemano.

Implementación con FreeRTOS:

- El juego deberá esperar a que se pulse cualquier tecla durante más de 500ms, para comenzar.
- Se consideran 4 topos, asociados a cada LED de la EDUCIAA.
- Cada topo estará asociado a la tecla que tiene al frente.
- Cada topo estará comandado por una tarea (instanciable).(Tarea Topo[i])
- Se considera martillazo al evento de button_pressed.
- Tarea Principal:
    - Recibirá los martillazos desde el driver de teclas.
    - Calculará los puntajes parciales en función del puntaje informado por cada topo.
    - Enviará el martillazo al topo correspondiente
    - Contará el tiempo de juego, y finalizará el mismo.
    - Deberá informar por UART cada actualización del puntaje.
    - Cuando finalice el juego, las otras tareas, deberán cesar su actividad.
    - Al finalizar cada juego, se deberá volver al punto de inicio, permitiendo un juego nuevo.
    - Si transcurren más de 15 segundos sin que el jugador pulse alguna tecla, deberá cancelar el juego.
- Tarea Topo[i]:
    - En cada ciclo deberá esperar un tiempo random para asomarse por el agujero, salir del agujero, esperar otro tiempo random, y esconderse (utilizar macros con valores máximos y mínimos)
    - Si, durante el plazo que está fuera del agujero, el topo recibe un martillazo, éste deberá esconderse.
    - Si el topo estaba "afuera del hoyo" cuando ocurrió el martillazo, medirá el tiempo entre que "salió del hoyo" hasta que recibe el martillazo, y se calculará el puntaje. (función `whackamole_points_success`)
    - Si el martillazo llega cuando el topo está oculto, resta -20 puntos (cada vez que la tecla sea pulsada en este estado) (función `whackamole_points_no_mole`)
    - Si no se martilla durante la aparición del topo, resta -10 puntos. (función `whackamole_points_miss`)
    - El puntaje, lo deberá informar a Tarea Principal en cada evento.
    - Deberá informar por UART cada vez que reciba un martillazo, enviando el tiempo de reacción del usuario.

Ayudas gráficas:



Aclaraciones:

- Si bien no se espera una perfecta modularización, intentar encapsular todo lo referido al juego en whackamole.c /.h
- El alumno puede agregar más tareas al kernel si así lo desea.
- La generación de números random puede utilizarse random.c y random.h.

Recomendaciones:

- Piense la lógica en un papel primero, incluyendo los elementos de sincronización que va a utilizar entre tareas.
- El alumno puede usar el template provisto (ver repositorio) o arrancar un proyecto de cero, basado en algún ejercicio desarrollado en la cursada.