Tecnológico de Monterrey Campus Cludad de México Escuela de Ingeniería y Ciencias Departamento de Computación	TE2004B - Diseño de sistemas embebidos avanzados
	Profesor: Rolando Bautista Montesano
	Práctica
	Nombres y matrículas
	Fecha

PRÁCTICA 9

Equipos de 3

Actividades previas

1. Asegúrate de tener una NUCLEO H745ZI-Q.
2. Consigue el material necesario.

Equipo y material

Equipo	Material
Computadora con STM32CubeIDE 1 NUCLEO H745ZI-Q 1 Cable micro-USB	

Desarrollo

1. Primero desarrollaremos un proyecto de FreeRTOS basado en la biblioteca CMSIS usando *polling*.
    a. Genera un proyecto nuevo para el núcleo M7. Asegúrate de configurar todos los LEDs y USART.
    b. Localiza *Middleware >> FREERTOS_M7*.
    c. Selecciona CMSIS_V2.
    d. No es necesario modificar nada dentro de *Coding parameters*.
    e. Localiza *Advanced settings*. Habilita el parámetro USE_NEWLIB_REENTRANT.
    f. Localiza *Tasks and Queues*.
        i. Agrega un nuevo *Task*.
            1. Asigna el nombre *uartTask*.
            2. Asigna la prioridad como normal.
            3. Asigna el nombre de la *Entry Function* a *StartUartTask*.
        ii. Agrega un nuevo *Task*.
            1. Asigna el nombre *blink01Task*.
            2. Asigna la prioridad como *above normal*.
            3. Asigna el nombre de la *Entry Function* a *StartBlink01Task*.
        iii. Agrega un nuevo *Task*.
            1. Asigna el nombre *blink02Task*.

2. Asigna la prioridad como *below normal*.
3. Asigna el nombre de la *Entry Function* a *StartBlink02Task*.

Cada uno de los tasks genera un *thread* para una tarea en específico. Las prioridades indican cuál de las tareas generadas debe ser revisada primero.

g. Genera código.
   i. *CubeIDE* debe de arrojar un mensaje de *warning*. Esto se debe a que FreeRTOS necesita un timer de 1ms para poder correr. De la misma forma, HAL requiere tener una fuente de reloj dedicada para todas las funciones que implementa.
   ii. Localiza y selecciona un *timer* sin salida a pin. Se recomienda elegir alguno con identificador grande. Únicamente selecciona el núcleo al cual va a estar asignado. NO habilites otras propiedades.
   iii. Localiza *System Core >> SYS*. Selecciona el timer que habilitaste.
h. Genera código. Nota que se incluyen diversos archivos .c y .h dentro de Inc y Src. En estos archivos se encuentran todas las funciones de FreeRTOS.
i. Revisa el código para M7.
   i. Nota que se generaron handlers del tipo *osThreadId_t* para cada tarea que definiste anteriormente.
   ii. Los respectivos atributos se encuentran dentro de las estructuras del tipo *osThreadAttr_t*.
   iii. Los *Tasks* que definiste anteriormente se encuentran definidos como funciones con los nombres que asignaste.
   iv. Los handles son usados para generar *Tasks* nuevos.
j. Busca la implementación de los *tasks uartTask*, *blink01Task* y *blink02Task*. Puedes tener varios hilos corriendo en paralelo dentro del núcleo en el que estés trabajando.
   i. Comenzaremos con *blink01Task*.
      1. Nota que la función tiene un ciclo infinito y un delay diferente al que has implementado con *HAL_Delay*.
      2. Estando dentro del ciclo infinito cambia el estado de un LED y asígna un delay de 500ms. Asegúrate de usar *osDelay*.
   ii. Ahora modificaremos *blink02Task*.
      1. Estando dentro del ciclo infinito cambia el estado de un distinto LED y asígna un delay de 1000ms. Asegúrate de usar *osDelay*.
   iii. Finalmente manda un *Hello world* cada 250ms en *uartTask*.

2. Ahora desarrollaremos un proyecto de FreeRTOS basado en la biblioteca CMSIS usando *timers*.
   a. Existen dos dipos de timers en FreeRTOS: los periódicos y los que se disparan una sola vez. Los periódicos corren continuamente después de ser activados. Se disparan cuando el periodo de tiempo que les fue especificado se cumple. Los de un disparo se activan únicamente durante un periodo de tiempo determinado al ser

llamados. Después se mantienen desactivados indefinidamente hasta que se les vuelve a llamar.

b. Genera un proyecto nuevo. Configura 3 LEDs y el botón de usuario de la NUCLEO para CM7.
c. Básate en el punto anterior para crear 2 *tasks*. Llámalos *uartTask* y *ledTask*. Asígnales prioridad normal y baja respectivamente.
d. En el archivo *.ioc* dentro de la configuración de *Timers and Semaphores* realiza la siguiente configuración:
    i. Agrega un nuevo timer y llámalo *periodicTimer01*.
        1. Llama el callback como *PTCallback*.
        2. En Type asígnale *osTimerPeriodic.*
    ii. Agrega un nuevo timer y llámalo *oneShotTimer*.
        1. Llama el callback como *OSTCallback*.
        2. En Type asígnale *osTimerOnce.*
e. Genera código.
    i. Nota que ahora hay definiciones, handlers y funciones para los timers también. Puedes hacer que tanto *tasks* como *timers* corran en paralelo usando hilos distintos.
    ii. Envía un *Hello world* cada 2 segundos usando delays dentro de *uartTask*.
    iii. Dentro del main, antes de entrar en el ciclo principal activa el timer periódico para que entre cada 1s a su definición con la siguiente instrucción.

        ```
        osTimerStart(periodicTimer01Handle, 1000);
        ```

        Esto hará que el timer periódico corra infinitamente.
    iv. En el callback de *ledTask* lee por polling el botón de usuario cada 1ms. Cuando se presione cambia de estado el LED2.
        Activa el *osTimerOnce* para que esté activo durante 4000ms usando la siguiente instrucción:

        ```
        osTimerStart(oneShotTimerHandle, 4000);
        ```

        Esto hará que se active el timer. Al terminar de contar entrará en su callback y ejecutará las instrucciones que se encuentren ahí. Si vuelves a presionar el botón antes de que se cumpla el plazo de 4s, el timer volverá a contar desde el inicio.
    v. Dentro de *PTCallback* envía el string “*Periodic timer”* por serial y cambia el estado del LED1.
    vi. Dentro de *OSTCallback* únicamente cambia el estado del LED2.
f. Programa la NUCLEO y observa el comportamiento de tu programa en los LEDs y en una terminal por serial. Reporta cuál es el funcionamiento esperado y cuál es observado.

3. Criterios de evaluación. Tu video debe contener las siguientes evidencias en código y demostración:
    a. (50 puntos) FreeRTOS con polling.
    b. (50 puntos) FreeRTOS con timers.

Consideraciones

1. Crea un proyecto diferente para cada inciso. Te ayudará a tener trazabilidad y puntos de revisión para verificar y respaldar tu avance.
2. Revisa qué rutinas son necesarias. Un diagrama de flujo puede resultar útil.

Entregables

1. Elabora un documento utilizando formato IEEE en el que reportes tus sistemas. Usa tablas para resumir tus constantes, mediciones y resultados, diagramas para ilustrar los circuitos implementados y diagramas de flujo cuando así sea pertinente. NO INCLUYAS CÓDIGO sin importar su longitud.
    a. Respeta el tipo y tamaño de letra, tamaño de columnas, tablas, figuras y referencias. Cada modificación al formato original será penalizada con 10 puntos.
    b. Cada falta de ortografía será penalizada con 2 puntos.
    c. De incluirse figuras y tablas, estas deben estar referenciadas dentro del texto. De lo contrario pueden ser omitidas.
    d. Incluye información relevante para tu reporte: indica cuál fue el problema a resolver, cuáles fueron las problemáticas enfrentadas, cómo se resolvieron. Presenta los resultados obtenidos. En la sección de conclusiones realiza una reflexión individual (y objetiva) de tu desempeño en la práctica.
    e. Emplea fotografías únicamente cuando sea indispensable: evidencia de armado de circuito, ilustrar componentes usados o similares. Incluye esquemáticos de los circuitos en vez de fotografías.
2. Adjunta tus archivos fuente procura que estén bien documentados. No adjuntes todo el proyecto. No es necesario comentar cada línea. Se sugiere comentar qué hace cada segmento, función o rutina.
3. No agregues todos tus archivos como un archivo comprimido a Canvas. Adjunta cada archivo por separado.
4. Graba un video en el que expliques tu código y en el que muestres el funcionamiento completo de cada punto desarrollado. Pueden ser videos separados. Para los videos de funcionamiento asegúrate de tener una buena fuente de iluminación. Es indispensable que relates qué está sucediendo en el video, de no haber narración se considerará el video como no entregado.
    a. La duración no debe ser mayor a 15 minutos.
    b. No incluyas ligas a Google Drive. Anexa tu video a Canvas o súbelo a YouTube.
    c. Explica tu código grabando tu pantalla usando Zoom. Evita grabar tu pantalla usando tu teléfono.
    d. La igual participación de todos los integrantes en el video es indispensable. Activa tu cámara al realizar tu video.
5. Entrega tus archivos en Canvas antes del inicio de la sesión del martes 30 de noviembre de 2021.