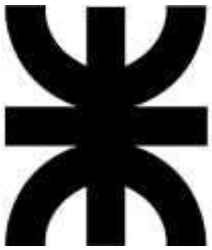

Universidad Tecnológica Nacional

Facultad Regional Avellaneda

Departamento de Ingeniería Electrónica

Cátedra: Técnicas Digitales III

Título del Trabajo Práctico:	Trabajo Práctico Nº1: FreeRTOS - Tareas
Autores:	

Resumen:
Contenidos analizados: • Manejo y API correspondiente a Tareas • Conmutación de Tareas • Scheduling/Política de Scheduling

Sistemas Operativos en Tiempo Real: FREERTOS

OBSERVACION: La entrega del presente trabajo se realizará mediante GitHub, en una carpeta llamada **tp1_tareas** dentro de la sección **2.trabajos_practicos**. El correspondiente branch será nombrado **tp1_tareas** y el commit **tp1_tareas_entrega**. La parte teórica debe ser entregada en formato PDF.

Parte Teórica

1) Sistemas Operativos - Generalidades

1-a) ¿Quién es el encargado de administrar el tiempo de la CPU y cuál es su función principal dentro del sistema?

1-b) Explique a que se denomina "Contexto de Ejecución". ¿Cuáles son las variables intervinientes?

1-c) ¿A qué se denomina Sistema Operativo de Tiempo Real? ¿Por qué usar un Sistema Operativo de Tiempo Real?

2) Kernel de FREERTOS

2-a) Explique y desarrolle las características de una "Tarea" en FreeRTOS. Proponga un ejemplo de una tarea manteniendo la sintaxis correcta.

2-b) Explique y desarrolle los "Estados de una Tarea" y sus transiciones. Indique las funciones que permiten las transiciones.

2-c) ¿A qué nos referimos con el concepto de "tareas de procesamiento continuo"? ¿y a qué nos referimos con el concepto de "tareas manejadas por eventos"?

2-d) ¿Cuáles son los tipos de eventos por los que una tarea puede estar esperando al entrar en el estado bloqueado? Explicar.

2-e) ¿Cuál es la función de la IDLE TASK? ¿Puede el procesador no estar ejecutando ninguna tarea en algún momento? Explicar.

Parte Práctica

1) Se desea generar el parpadeo de un LED mediante dos tareas, las cuales cumplirán las siguientes funciones:

Tarea 1: Controlará únicamente el encendido del LED. El mismo debe permanecer encendido por 600ms.

Tarea 2: Controlará únicamente el apagado del LED. El mismo debe permanecer apagado por 400ms.

* No se permite el uso de variables globales a modo de "flags". El ejercicio deberá resolverse íntegramente

2) CONDICION DE PROMOCION: Habiendo cumplido con lo solicitado anteriormente, se solicita: añadir un pulsador, el cual al ser presionado permitirá mantener el estado que el LED tenía al momento de la activación. El LED no cambiará de estado mientras mantenga presionado el pulsador.

Parte Teórica

1-a) ¿Quién es el encargado de administrar el tiempo de la CPU y cuál es su función principal dentro del sistema?

El scheduler es el núcleo del sistema operativo.
Su función principal es administrar la ejecución de las tareas, dándole a cada una un tiempo de la CPU dependiendo de la política de Scheduling.
Dicha política nos indica como llevar a cabo las tareas de una forma organizada y planificada en base a los recursos, prioridades, recursos bloqueantes y al estado de las tareas. El scheduler ya está programado y forma parte del Kernel.
Hay distintos tipos de scheduler con distintas políticas para decidir la ejecución de las tareas.

1-b) Explique a que se denomina “Contexto de Ejecución”. ¿Cuáles son las variables intervinientes?

El contexto de ejecución es aquello que necesita una tarea guardar cuando deja de ser ejecutada para la próxima vez poder retomar.
Lo que debe realizarse al cambiar una tare:

- Salvar el estado (registros, información de punteros de memoria) que se están ejecutando.
- Cambiar el estado de la tarea que estaba ejecutando al que corresponda.
- Cargar el estado asignado a la CPU.
- Cambiar el estado, de la siguiente tarea a ejecutarse, a “ejecutando”

1-c) ¿A qué se denomina Sistema Operativo de Tiempo Real? ¿Por qué usar un SistemaOperativo de Tiempo Real?

Un Sistema Operativo de Tiempo Real es un sistema operativo ligero capaz de poder ser integrado en microprocesadores, por lo cual son de tamaño ligero para poder entrar en dicho micro. Se llaman de tiempo real porque tienen una respuesta determinística en el tiempo más que velocidad,

lo que prioriza es que el tiempo de ejecución sea siempre preciso. Por lo cual, satisface los requerimientos de tiempo real duro debido a su determinismo.
Además, debe tener recursos que van a ser líneas de códigos especificas que en su conjunto permitan administrar las funciones del sistema operativo. Para lo cual, contará con APIS, una asociada a las tareas, otra asociada a las colas, semáforos y a las distintas funciones del sistema.
Este tipo de sistemas operativos se utilizan en aplicaciones donde es importante que el sistema responda a un tiempo específico. Por ejemplo, el mecanismo de disparo de un Airbag debe desplegarse en un cierto tiempo después del impacto, si la respuesta se demora, el conductor podría sufrir heridas serias.

2-a) Explique y desarrolle las características de una "Tarea" en FreeRTOS. Proponga unejemplo de una tarea manteniendo la sintaxis correcta.

Una tarea en FreeRTOS es la unidad básica de ejecución de código que puede ser programada y administrada por el sistema operativo. Para nosotros una tarea va a ser una función en C con algunas características particulares. ¡Observación: Una tarea es una función en C pero una función en C no es una tarea!
Las características particulares son:

- Prioridad: Cada tarea tiene su propia prioridad, lo que le permite al Scheduler asignar recursos de CPU de manera eficiente en función de la importancia de cada tarea.
- Contexto de ejecución: Cada tarea tendrá su propio contexto de ejecución para poder ejecutarse sin afectar a las otras tareas.
- Estados: Una tarea puede estar en estado Ejecución y No Ejecución. Dentro de este último puede estar en Bloqueada, Suspendida o Ready (lista para ejecutarse).
- Tiempo de ejecución asignado: El Scheduler asigna un tiempo de ejecución a cada tarea, lo que evita que una tarea monopolice la CPU y garantiza que todas las tareas reciban una cantidad justa.

Ejemplo de una tarea que hace titilar un led:

```
xTaskCreate(tarea_blinky,
"blinky",
configMINIMAL_STACK_SIZE,
NULL,
1,
NULL);

void tarea_blinky(void *p)
{
while(1)
{
        HAL_GPIO_TogglePin(GPIOC, GPIO_PIN13);
        vTaskDelay(1000);
}
}
```

2-b) Explique y desarrolle los "Estados de una Tarea" y sus transiciones. Indique lasfunciones que permiten las transiciones.

Estados de una Tarea:

- Ejecución: La tarea se está ejecutando en la CPU.
- No Ejecución: El cual se puede expandir en los siguientes estados:
- Bloqueado: La tarea está esperando un evento para estar disponible, los eventos que espera pueden ser:
    - Eventos temporales: Se debe cumplir un retardo de un cierto tiempo y una vez pase dicho tiempo la tarea deja de estar bloqueada. Por ejemplo, con el uso de al función vTaskDelay()
    - Eventos de Sincronización: Donde el evento es originado desde otra tarea o

interrupción. Por ejemplo, una tarea puede bloquearse con un semáforo o cola.

- Suspendido: Las tareas en este estado no están disponibles para que el Scheduler las pase a Ejecución. La única manera de entrar en este estado es con la función vTaskSuspend(), y la forma de salir de este estado es con las funciones vTaskResume() p vTaskResumeFromISR().
- Disponible: Las tareas que no se están ejecutando, pero no se encuentran en ninguno de los otros 2 estados. Significa que el scheduler les puede asignar tiempo de CPU. Con vTaskCreate() se crea una tarea y se le asigna este estado.

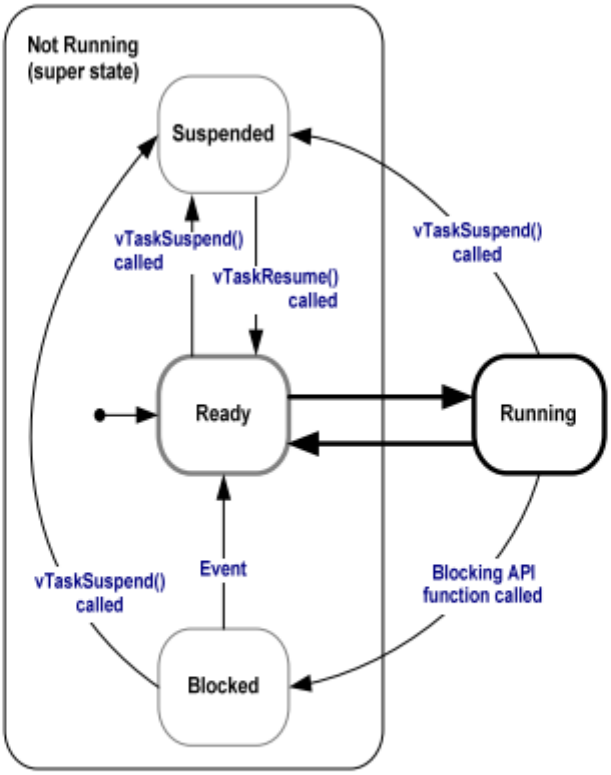

Figura 7 – Diagrama completo de estados y transiciones.

Nota: Suspend = Suspendido, Ready = Listo o Disponible, Blocked = Bloqueado, Running = En Ejecución.

2-c) ¿A qué nos referimos con el concepto de “tareas de procesamiento continuo”? ¿y a quénos referimos con el concepto de “tareas manejadas por eventos”?

Las “tareas de procesamiento continuo” son aquellas que están siempre disponibles para su ejecución, ya que no deben esperar ningún tiempo o evento para ser ejecutadas. Tienen una utilidad limitada y siempre deberán tener el menor nivel de prioridad disponible, porque sino impedirán la ejecución de tareas de menor prioridad.

Las “tareas manejadas por eventos” sólo entran en ejecución luego que haya ocurrido cierto evento que la dispara, y no pueden ejecutarse antes de que dicho evento ocurra. Estas tareas pueden tener distintos niveles de prioridad, porque si una tarea de mayor prioridad se encuentra bloqueada debido a que está esperando un evento, una tarea de menor prioridad que se encuentra disponible puede ser ejecutada por el Scheduler.

2-d) ¿Cuáles son los tipos de eventos por los que una tarea puede estar esperando al entrar enel estado bloqueado? Explicar.

Los eventos que puede esperar una tarea al entrar en estado Bloquedo son

- Eventos temporales: Se debe cumplir un retardo de un cierto tiempo y una vez pase dicho tiempo la tarea deja de estar bloqueada. Por ejemplo, con el uso de al función vTaskDelay()

- Eventos de Sincronización: Donde el evento es originado desde otra tarea o interrupción. Por ejemplo, una tarea puede bloquearse con un semáforo o cola.

2-e) ¿Cuál es la función de la IDLE TASK? ¿Puede el procesador no estar ejecutandoninguna tarea en algún momento? Explicar.

El procesador no puede estar ejecutando ninguna tarea, por lo que existe una tarea ociosa (IDLE TASK) que se crea automáticamente cuando se llama a la función vTaskStartScheduler(). Dicha tarea no hace más que entrar en un bucle y siempre es capaz de ejecutarse.
Esta tarea cuenta siempre con la misma prioridad (cero 0) para asegurarse permitir que las tareas de mayor prioridad puedan ejecutarse.
La tarea IDLE cuenta con ciertas si así se desea:

- Ejecución de baja prioridad, de fondo o de procesamiento continuo, estando siempre disponible para ejecución.
- Medir la capacidad de procesamiento extra. Debido a que esta tarea, entra en ejecución cuando ninguna tarea está consumiendo tiempo del procesador, es muy practica para medir el tiempo que no se está utilizando.
- Colocar el procesador en un modo de bajo consumo. Poner en ahorro de energía al procesador siempre que no haya procesamiento a realizar.