Carrera de Especialización en Sistemas Embebidos

Sistemas Operativos en Tiempo Real

Clase 2 (Práctica): Temporización en FreeRTOS

ACSE
Asociación Civil para la Investigación, Promoción y Desarrollo de los Sistemas Electrónicos Embebidos
FACULTAD DE INGENIERIA
Universidad de Buenos Aires
CIM
CADIEEL

RTOS: Aspectos fundamentales

- Sabemos, de la clase anterior, que los RTOS se encargan de gestionar los recursos de nuestro SE, destacando:
  - Tiempo de procesador
    Generando la ilusión de **multitareas** y asegurando la ejecución de cada tarea, de acuerdo con su disponibilidad y prioridad, en un **plazo de tiempo acotado**
  - Memoria
    Reservando los espacios de memoria de cada tarea y asegurando el almacenamiento del **Contexto de Ejecución**

RTOS: Aspectos fundamentales

- Además los RTOS nos ofrecen herramientas de programación tales como:
  - Temporización
    Permitiendo manejar esperas y pausas sin gestionar manualmente temporizadores de hardware ni contadores de software
  - Sincronización
    De tareas con eventos externos y con otras tareas
  - Comunicación
    De tareas con manejadores de periféricos u otras tareas en forma segura y ordenada
  - Gestión de otros recursos de Hardware

Comenzando con FreeRTOS

- Para comenzar a trabajar con FreeRTOS, debemos incorporar la biblioteca a nuestro proyecto (en caso de CIAA-IDE a través de config.mk) y crear el archivo de configuración FreeRTOSConfig.h

- Una vez que hemos **diseñado** nuestro sistema para la aplicación a implementar, escribiremos como funciones, las tareas que serán ejecutadas por el SO utilizando el siguiente prototipo:

```
void tarea(void *pvParameters);
```

- Finalmente, será necesario inicializar el Hardware, crear las tareas en el contexto del sistema operativo e inicializar el RTOS. Todo esto, en el main de nuestro programa en C.

Ejercicio #1 - Demoras Fijas

- Para comenzar implementaremos un **Heartbeat** en el led azul de la EDU-CIAA, utilizando FreeRTOS, donde el período sea de **1s** y el led cambie de estado cada **500ms**, como se muestra en el siguiente gráfico:

Ejercicio #1 - Demoras Fijas

- Para implementar este Heartbeat necesitaremos:
  - Tarea de control de encendido/apagado del led
    Que deberá activarse cada 500ms encendiendo el led al primer llamado y apagándolo al segundo, repitiéndose cíclicamente.
  - Un temporizador
    Que asegure la llamada de la tarea cada 500ms y que permita al sistema liberar el procesador entre cada llamada de la tarea, una vez que ésta haya finalizado su labor.

    Para este último utilizaremos una API de FreeRTOS que puede ser llamada desde la tarea, a la cual bloquea durante el tiempo seteado:
    `void vTaskDelay( const TickType_t xTicksToDlay );`

Ejercicio #1 - Demoras Fijas

- Con esta herramienta, la tarea podría ser escrita de la siguiente forma:

```
void Heartbeat( void* taskParmPtr )
{
    uint8_t LedState = 0;
    while(TRUE) {
      // Escribe el estado en el Led
      gpioWrite( LEDB, LedState );
      // Intercambia el estado del Led Azul para la próxima ejecución
      if ( 0 == LedState ) LedState = 1; else LedState = 0;
      // Bloquea la tarea durante 500ms, liberando el uso del CPU
      vTaskDelay( 500 / portTICK_RATE_MS );
   }
}
```

Ejercicio #1 - Demoras Fijas

- Nótese que la definición de la tarea contiene una inicialización y luego una rutina encerrada en un loop.

  Gracias a que FreeRTOS es expropiativo, esto nos permite escribir cada tarea como si fuera la única en ser ejecutada por el procesador.

  Una vez definida cada una de las tareas solamente nos queda preparar el main, con la inicialización del Hardware, la creación de la tarea y la inicialización del SO.

Ejercicio #1 - Demoras Fijas

- Escribimos el main de la siguiente manera:

```
int main(void)
{
   // Inicializamos la EDU-CIAA
   boardConfig();
   // Creamos la tarea en freeRTOS
   xTaskCreate(Heartbeat, (const char *)"Heartbeat", configMINIMAL_STACK_SIZE*2,
    NULL, tskIDLE_PRIORITY+1, NULL);
   // Iniciar scheduler
   vTaskStartScheduler();
   //Aseguramos que no salga del programa principal, en caso de un error de inicialización
   while( TRUE ) {
   }
   return 0;
}
```

Ejercicio #1 - Demoras Fijas

- Resta compilar y cargar en la EDU-CIAA

¡MANOS A LA OBRA!

Ejercicio #2 - Período Fijo

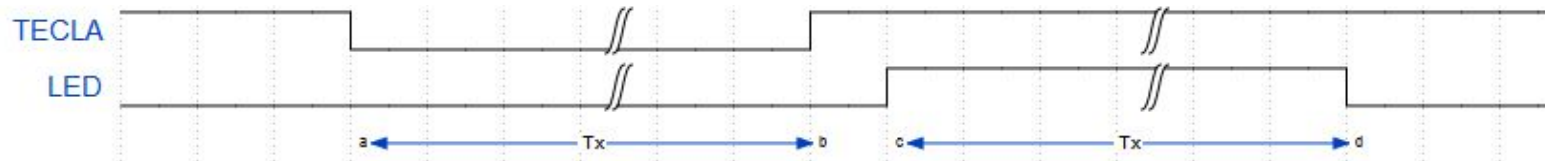
- Ahora implementaremos una **Onda Cuadrada** en el led azul de la EDU-CIAA, utilizando FreeRTOS, donde el período sea de **1s** y el tiempo de led encendido crezca de **100 a 900ms en pasos de 100ms**, de forma cíclica, como se muestra en el siguiente gráfico:

- Recomendación: Apoyarse en la segunda API de temporización que ofrece FreeRTOS, para asegurar el período.

```
void vTaskDelayUntil( TickType_t *pxPreviousWakeTime,
                      const TickType_t xTimeIncrement )
```

Ejercicio #3 - Medir tiempo

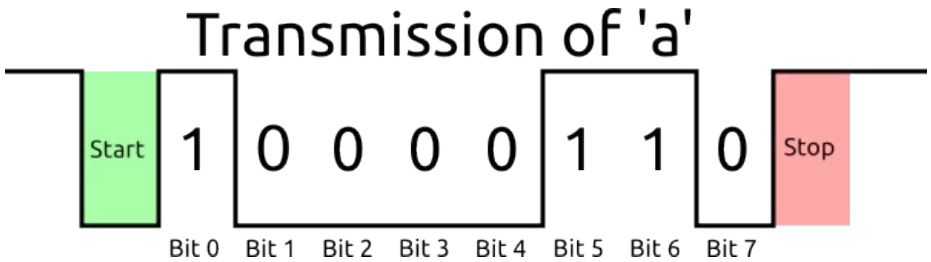
- Para comprobar el estado de funcionamiento de un pulsador, queremos reproducir el tiempo de pulsación en un led de la EDU-CIAA. Implementando un mecanismo anti-rebote por software, desarrolla un programa que mida el tiempo de pulsación y que luego lo reproduzca en el led azul como se muestra en el siguiente gráfico:



- Recomendación: Se puede consultar el contador de ticks del SO para medir el tiempo. Sin embargo, se debe tomar en cuenta que la variable que contiene el contador de ticks puede desbordarse.

Ejercicio Extra - UART por Software

- En una aplicación con la EDU-CIAA, todos los puertos UART se encuentran ocupados, por lo que se quiere implementar una salida de comunicación serial asincrónica a baja tasa de transmisión (< 500 bps) para enviar señales a un sistema de monitoreo, como se muestra a continuación.



- Implementa una biblioteca que permita ser llamada desde una tarea en FreeRTOS, para utilizar esta salida y que contenga las siguientes funciones:

void sw_uart_sent(uint8_t byte_a_transmitir)

void sw_uart_config(uint16_t baurdate)

Bibliografia

- RTOS 1 - Clase 1 - Introducción a los RTOS, Franco Bucafusco, 2018
- Sistemas Operativos de Tiempo Real - Guía de Ejercicios, Franco Bucafusco, 2016
- Sistemas Operativos de Tiempo Real - Guía de Ejercicios Adicionales, Franco Bucafusco, 2017