

**Mini\_RTOS**

**Nombre:**

Jorge Mizael Rodríguez Gutiérrez

Luis Roberto Lomelí Plascencia

**Expediente:**

IE698323

IE700093

**Correo:**

[ie698323@iteso.](mailto:ie698323@iteso.)mx

[ie700093@iteso.](mailto:ie700093@iteso.)mx

**Carrera:**

Ing. Electrónica

**Fecha:**

17/Enero/2017

**Materia:**

Sistemas embebidos 2

**Maestro:**

Rodrigo Aldana López



**Pagina**

**Portada**.……….……………………………………………………………………………………….. 01

**Índice**………….…………………………………………………………………………………………02

**Introducción**……………………………………………………………………………………………03

**Planteamiento del problema**………………………………………………………………………..04

**Modificaciones al código**……………………………………………………………………………05

**Análisis y conclusiones**………………………………………………………………………….…03

**Referencias**……………………………………………………………………………………………03

**Introducción**

Como primer laboratorio de la clase de sistemas embebidos 2, se realizó un sistema operativo para la tarjeta freedom K64f, la implementación de dicho sistema operativo se llevó a cabo gracias a la ayuda del profesor de la clase, al implementar este sistema operativo podemos decir que es la implementación de un sistema operativo muy básico, que incluso no tiene interfaz grafica.

**Planteamiento del problema**

Para esta actividad deberás clonar el proyecto que se encuentra en esta liga: [https://github.com/raldanal/rtos.](https://github.com/raldanal/rtos) En ella encontrarás el código fuente para que coloques en un proyecto nuevo de MCUXpresso para la tarjeta freedom board K64.

El archivo rtos\_main.c contiene el punto de entrada del programa, en el cual se definen 3 tareas las cuales el objetivo es que terminen corriendo en paralelo manteniendo su funcionalidad y contexto.

El módulo rtos contiene la API e implementación incompleta de un calendarizador por prioridades que deberás completar. Por lo pronto no hay nada implementado a excepción del SysTick timer configurado para generar una señal de "vida” que servirá para saber si el sistema operativo está vivo.

Tu labor será terminar de implementar el resto del archivo rtos.c de manera que las tareas en el archivo main corran como se desea. A continuación, se muestran pseudocódigos de lo que las funciones deberán realizar.

**Modificaciones realizadas al código**

**Archivo “rtos.h”**

Para estearcivo cosas que corregimos fueron etiquetas **#define** y **enum**, con la finalidad de quitar los números mágicos en el programa. Ejemplo de líneas que agregamos son las siguientes:

Los defines son para nueros que representaron algo en el código, que tal vez para el usuario no era claro. Por lo tanto, agregamos estas etiquetas para respetar los lineamientos de codificación y hacer más entendible el código

**#define** ALL\_TASKS\_CREATED\_CORRECTLY 66

**#define** RESERVED\_MEMORY 10

**#define** STACK\_OFFSET\_ISR\_AND\_EXEC 9

**#define** TASK\_IDLE 1

**#define** END\_OF\_STACK 1

**#define** VALUE\_START\_PERIOD 1

Pasa algo similar con los enums, ya que podríamos decir que estos tienen la misma función, la diferencia radica en que los números que representan estas etiquetas se representan con más frecuencia y por lo general son dos o más etiquetas en las cuales su número es una secuencia del otro.

/\*! @brief type of variable start \*/

**typedef** **enum**

{

*START\_ZERO* = 0, *START\_ONE*

} start\_values\_type\_e;

/\*! @brief priority of tasks \*/

**typedef** **enum**

{

*LOWEST\_PRIORITY* = -1, *PRIORITY0* = 0, *PRIORITY1*, *PRIORITY2*

} task\_priority\_type\_e;

/\*! @brief led state \*/

**typedef** **enum**

{

*ON* = 0, *OFF*

} led\_state\_type\_e;

**Funciones del archivo “rtos.c”:**

como primer paso, se revisaron todas las funciones y se les quitaron los números mágicos.

Una de las funciones a las cuales se les hizo una modificación importante fue “, ya que a diferencia del sistema operativo en que el profe corría en Ubuntu, el offset es diferente para los sistemas operativos Windows, ya que los datos guardados en el stack se reflejaban recorridos un número más, el cual era nueve y no ocho.

Este pequeño detalle nos ocasiono problemas ya que al momento de guardar el contexto y tratarlo de recuperar el stack pointer que agarrábamos no coincidía con el de la tarea por el desfase del offset.

En general gracias a las clases y al seudocódigo que nos dio el profe fue fácil implementar las otras funciones y no hubo grandes modificaciones, que no aparecieran en el seudocódigo.