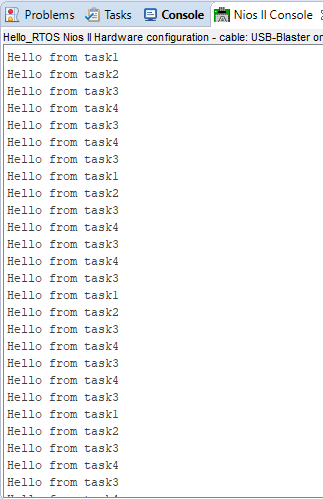
Practica 1 RTOS

Jose Luis Rocabado Rocha

1. Ejercicio 1

Sale esto. Task 3 cada 1 segundos, task4 cada 1.5 segundos.



1. Ejercicio 2

**¿Qué ocurre cuando se utiliza el pulsador KEY[1]?**

Se genera una interrupción cuyo contador se muestra en la pantalla MTL.



**¿Qué papel desempeñan las funciones de manejo de LEDs?**

Nos permiten comprobar de forma visual la temporización de las tareas.

**¿Y las de MTL?**

Visualizar el curso de la ejecución sin consumir demasiados recursos de la cpu.

**¿Qué problemas pueden existir con el debug por printf?**

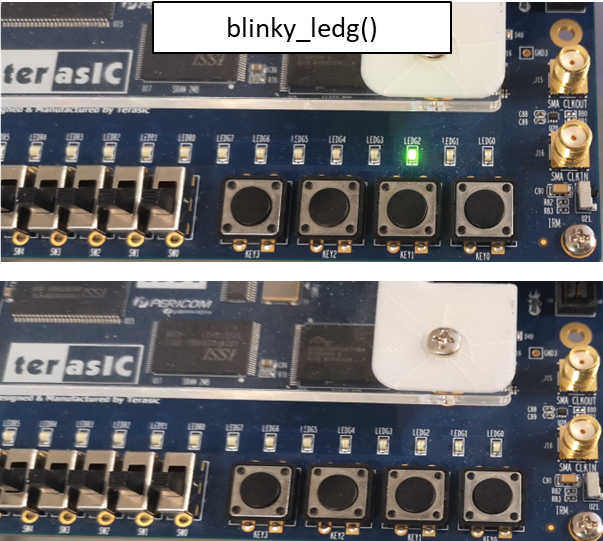
Pues que el printf consume muchos recursos para realizar la comunicación con el JTAG.

**¿Existe una buena comunicación entre tareas? ¿Qué método de comunicación se usa?**

No es la mejor comunicación puesto que el sistema es expulsivo y podría producir conflictos al acceder a la variable global “líneas”.

1. Ejercicio 3

La función *“blinky\_ledg()”* nos permite realizar un *toggle* del LEDG2 cada vez que se ejecuta la tarea 3. El parpadeo producido en las otras tareas nos permite observar el tiempo durante el que se ejecutan sus tareas dado que al inicio de la tarea se encienden y al final se apagan sus respectivos LEDs.



1. Ejercicio 4

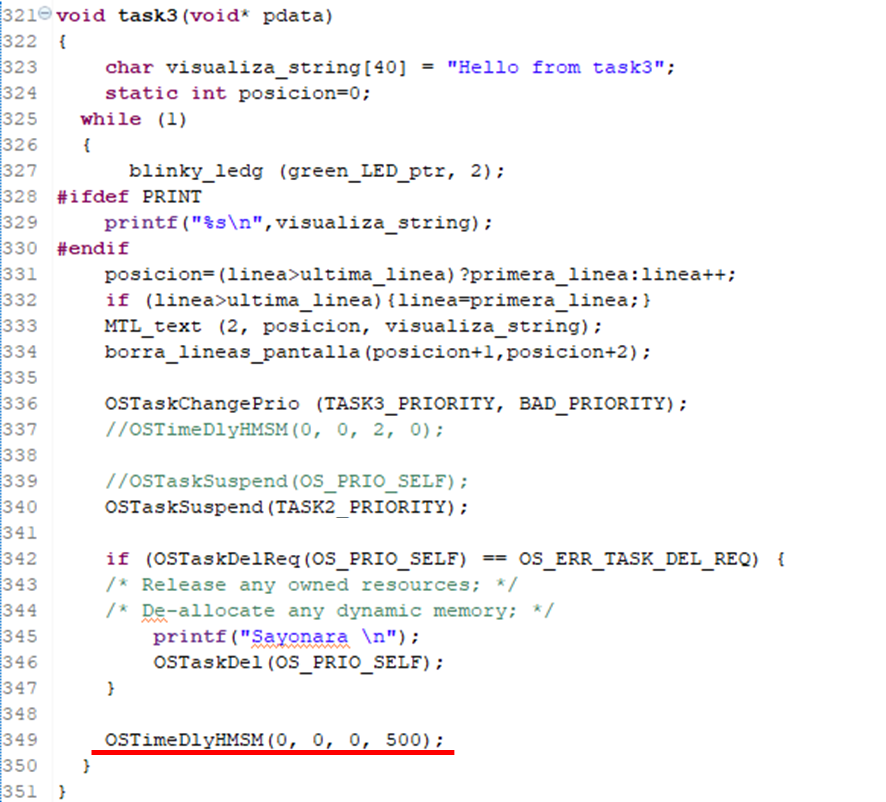
*Snippet* de código, se observa que se puede cambiar tanto la prioridad de las otras tareas como la propia de la tarea.



1. Ejercicio 5

Se comprueba que al tener una prioridad muy mala pero mismo tiempo de ejecución

se ejecuta después de la tarea 1 y 2.



1. Ejercicio 6

Tras su primera ejecución, la tarea 3 se suspende dado que se ha usado la macro “OS\_PRIO\_SELF”.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Sucede lo mismo, pero con la tarea 2 una vez se ejecuta la tarea 3.

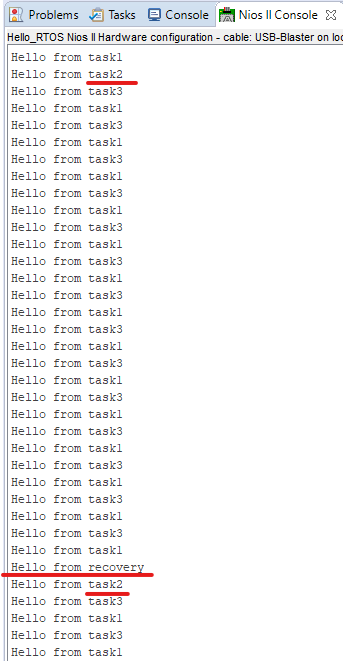
Se demuestra así que se puede suspender tareas de mayor prioridad dentro de tareas de menor prioridad.

Interfaz de usuario gráfica, Texto, Aplicación, Tabla

Descripción generada automáticamente

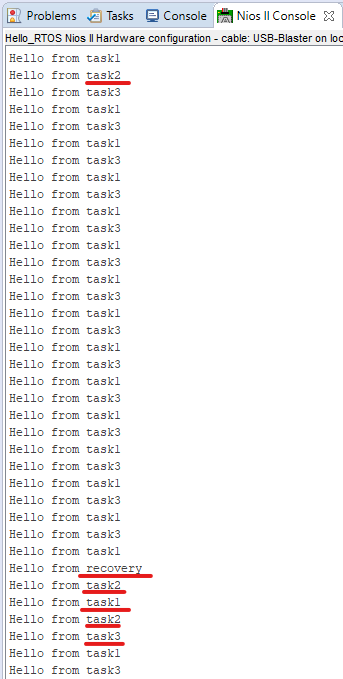
1. Ejercicio 7

Se suspende la tarea 2 tras ejecutar la tarea 3 y no se vuelve a activar hasta que se ejecute la tarea 4 que la reactiva. Sin embargo, tras ejecutarse la tarea 3 se vuelve a suspender.



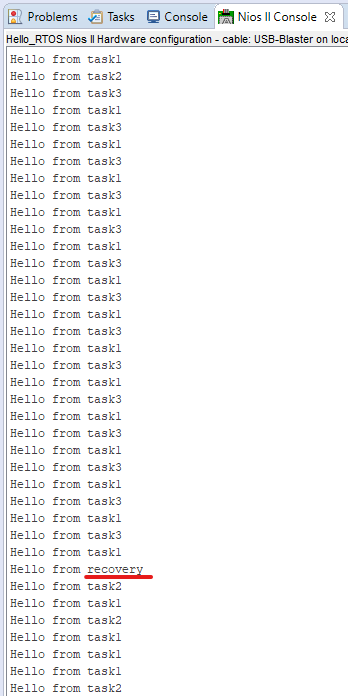
1. Ejercicio 8

Lo mismo que en el ejercicio anterior, sin embargo, tras ejecutarse la tarea 4 se ponen en modo *RUNNING* las demás tareas.



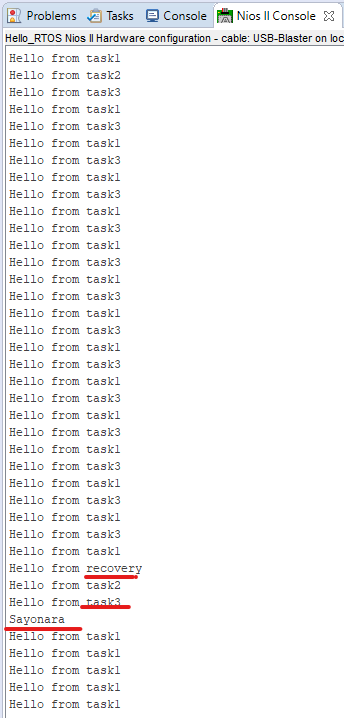
1. Ejercicio 9

Tras ejecutar la tarea 4 se “duerme” definitivamente la tarea 3 de forma directa.



1. Ejercicio 10

Sin embargo, si se utiliza la solicitud de eliminar tarea, se ejecuta el código que imprime “sayonara” antes de “dormir definitivamente la tarea 3”



1. Ejercicio 11

Se han añadido nuevos elementos a nuestro proyecto tal y como se ve en el *Workspace*

