A. Gestión de tareas

A.1- Implementar tres tareas que no posean llamados a funciones de la API de FreeRTOS con puntos de re-planificación con la misma prioridad. Una de las tareas deberá monitorear el valor mínimo utilizado del stack de cada tarea.

Valide el funcionamiento del mecanismo de detección.

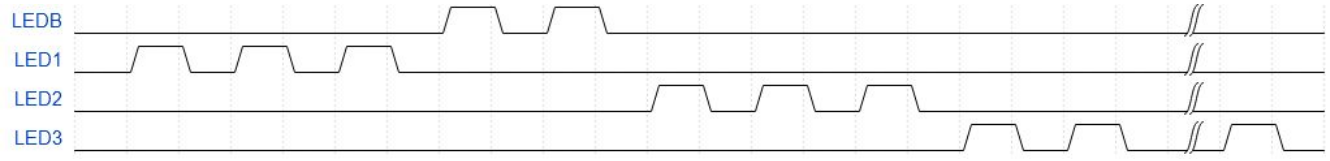
A.2- Para el caso de A.1, utilice el IDLE hook para la validación. ¿ Qué ocurre ?

A.3- Implemente un sistema de 4 tareas:

Tarea A - Prioridad IDLE+4 LED asociado LEDB
Tarea B - Prioridad IDLE+3 LED asociado LED1
Tarea C - Prioridad IDLE+2 LED asociado LED2
Tarea D - Prioridad IDLE+1 LED asociado LED3

Arrancando solamente la tarea A antes de comenzar el scheduler, genere la siguiente secuencia de encendido y apagado (500ms/500ms):

LED1 LED1 LED1 LEDB LEDB LED2 LED2 LED2 y posterior a esta secuencia, permaneces titilando a LED3

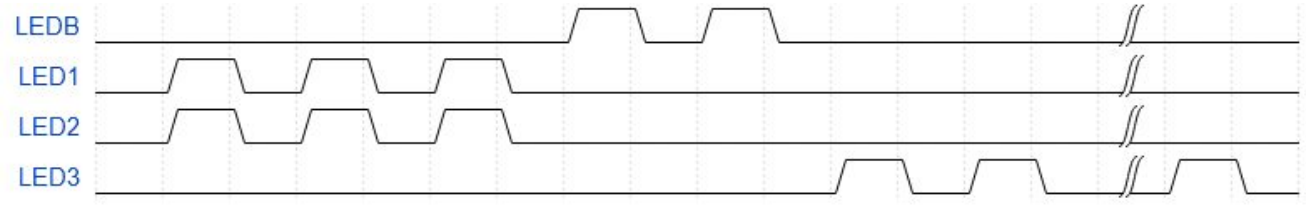


Solo la tarea D podrá destruir las otras, cuando comience a operar.

A.4- Partiendo del ejercicio A.3 implemente un sistema de 4 tareas:

Tarea A - Prioridad IDLE+4 LED asociado LEDB
Tarea B - Prioridad IDLE+2 LED asociado LED1
Tarea C - Prioridad IDLE+2 LED asociado LED2
Tarea D - Prioridad IDLE+1 LED asociado LED3

Valide que la secuencia es la siguiente.



Analice el funcionamiento.

Ahora, configure en freertosconfig.h: #define configUSE_TIME_SLICING 0

¿Qué sucedió?

Proponga una manera de contrarrestar el efecto sin tocar la configuración mencionada (no utilice la Suspend/Resume para solucionarlo)