# Sistemas Embebidos
# Trabajo práctico Nº3 - Año 2023
# FreeRTOS

**Objetivos**

- Utilizar FreeRTOS
- Repasar conceptos de sistemas operativos.

**Metodología**

Trabajo individual o grupal. 2 estudiantes por grupo máximo.
Tiempo de realización estimado: 2 clases.

**Aprobación**

- Mostrar en clases la aplicación funcionando correctamente.
- Enviar los programas de computación implementados a través de la plataforma Moodle.
- Escribir un breve informe y enviarlo a través de la plataforma Moodle.

**Materiales necesarios**

- Placas Arduino UNO (provistas por la cátedra).
- Entorno de desarrollo de Arduino UNO (puede descargarse de https://www.arduino.cc/en/software).
- Librerías para Arduino IDE:
    - FreeRTOS de Richard Barry. Verificar que diga “implementado para AVR (Uno, Nano, Leonado, Mega)”.

## Actividades

1. Instalar la librería para Arduino UNO de FreeRTOS y analizar los ejemplos.
2. Implementar una aplicación que realice las siguientes tareas usando FreeRTOS:
    a. Lea constantemente el valor de la intensidad luminosa (Nota: la lectura analógica demora un tiempo. No debe ser interrumpida por otra tarea).
    b. Cada 3 segundos, envíe a través del puerto serial el último valor leído (Nota: la escritura en el monitor serial demora cierto tiempo. No debe ser interrumpida por otra tarea). Muestre el valor leído en una página web y en la aplicación Python.
    c. Si detecta que el valor de intensidad luminosa supera 800, encienda una alarma que:
        i. Haga parpadear el led 12 con periodo de 0.5 segundo

ii. Indique en la aplicación web la situación.

d. La lectura pueda iniciarse y detenerse a través de los pulsadores de la placa Arduino y desde botones en una página web.
e. Parpadee el led 11 cada un segundo si la lectura está activada. No parpadee el led 11 si la lectura está desactivada.

Sugerencias de implementación:

- Cada ítem puede ser configurado como una tarea de FreeRTOS.
- El manejo de pulsadores debería realizarse mediante interrupciones. Sin embargo, la rutina de servicio de la interrupción debería ejecutar un código de mínima duración, como habilitar una tarea, siendo esta tarea la que active o desactive el sistema.
- Deberá asignar memoria para la pila de las tareas. Debe tener en cuenta que la memoria del Arduino Uno es limitada. Asignar poca memoria puede hacer que la tarea no funcione correctamente. Asignar demasiada memoria puede hacer que se agote la memoria del Arduino Uno, y el programa completo no funcione (ver Anexo 2).
- Las variables de tipo `string` hacen uso ineficiente de la memoria, por no ser variables de tamaño fijo. Su uso inadecuado puede llevar a sobreescribir otras variables e incluso código. Se sugiere utilizar strings de tamaño fijo y el menor tamaño posible. Una mejor solución puede ser el uso de array de caracteres (`char my_array[5]`), ya que los mismos hacen uso de una cantidad fija de memoria, aunque su manejo puede ser un poco más complejo.

## Anexo 1: Librerías necesarias

En https://www.freertos.org/a00106.html se listan todas las primitivas de FreeRTOS y las librerías que necesita para cada función.

Manejo de interrupciones: Revise la función `attachInterrupt` en https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

## Anexo 2: Tamaño de pila

```
UBaseType_t uxTaskGetStackHighWaterMark( TaskHandle_t xTask );
```

Retorna la porción de la pila no utilizada por la tarea indicada. Se retorna en cantidad de palabras (16 bits en Arduino). Permite saber si se puede disminuir el tamaño de la pila de una tarea.
La función indicada consume memoria en la pila, por lo que debe ser utilizada con precaución.