

Sesión de Laboratorio 6-II: Multitarea con freeRTOS

Tiempo estimado: 1,5 h (1 sesión)

1. Trabajando con hardware real

1.1 Probar IMU (sensor inercial)

Lee y lanza el programa `IMU.ino` para entender cómo trabajar con el sensor en M5Core2.

1.2. Probar motor vibración

Lee y lanza el programa `vibration.ino` para entender cómo trabajar con el actuador en M5Core2.

1.3. Multitarea con sensor y actuador

Haz un script para M5Core2 usando FreeRTOS que haga lo siguiente:

1. Se van a utilizar 4 tareas. En principio, todas se ejecutarán en el core 0.
2. La tarea 1 sirve para monitorizar la pulsación de los botones **A**, **B** y **C**. Tendrá prioridad 2 y se ejecutará con una frecuencia de 100Hz.
3. La tarea 2 escribirá los datos de la IMU en la pantalla del M5, de forma similar al programa `IMU.ino`. En este caso, los datos se escribirán cuando se haya pulsado el botón **B**. Si se vuelve a pulsar, la pantalla se queda en el estado actual. Es decir, los datos no se actualizan. Si se pulsa de nuevo, se volverán a mostrar. Y así, sucesivamente. Esta tarea tendrá prioridad 1 y se ejecutará con una frecuencia de 25Hz.
4. La tarea 3 establece un patrón de vibración a modo de alarma si se ha alcanzado una aceleración igual o superior a $2G$ `sqrt(accX*accX + accY*accY + accZ*accZ > 2)`. El patrón hará que el motor vibre 5 veces con un intervalo de 200ms. Tendrá prioridad 3 y se ejecutará con una frecuencia de 5Hz.
5. La última tarea escribirá por el puerto serie cada cambio ocurrido en la aplicación. Sólo escribirá si ha ocurrido algún evento (pulsación botón **A**, **B** o **C**, o vibración). Tendrá prioridad 0 y se ejecutará con una frecuencia de 10Hz. Los datos a publicar son los siguientes:
 - Cuántas veces se ha pulsado **A**.
 - Cuántas veces se ha pulsado **C**.
 - Si se ha pulsado **B** y se están publicando los datos de la IMU, escribirá: `"Usando datos IMU"`. En caso contrario: `"No usando datos IMU"`

PREGUNTA:

- Como el dispositivo tiene 2 núcleos, se podrían emplear ambos para balancear mejor la carga de las tareas. De todas las tareas anteriores, ¿Qué tarea/s pondrías en cada núcleo? ¿Por qué? Prueba a ejecutar el programa cambiando dichas tareas y comprueba si hay alguna diferencia en el comportamiento.

1. Trabajando en simulación

Para hacer esta práctica en simulación se puede trabajar con el ESP32 de la siguiente manera:

- Tres botones externos que hagan de botones **A**, **B**, y **C**. Habrá que configurar las interrupciones externas.
- Para trabajar con la IMU hay que añadir una externa. En wokwi se puede utilizar la IMU *mpu6050*. [Aquí](#) se puede ver la documentación. La comunicación IMU-ESP32 se hará por I2C. [Aquí](#) podéis encontrar un ejemplo con Arduino.
- En wokwi no hay un motor de vibración para la alerta cuando la aceleración sea mayor de 2G. Este motor se puede sustituir por un LED parpadee o que varíe su intensidad usando diferentes valores del *duty cycle* de una PWM.
- Para escribir los datos por pantalla hay dos opciones:
 1. Lo más sencillo es enviarlos por el puerto serie.
 2. Si se quiere, se puede utilizar una pantalla TFT similar a la que tiene el M5Core2 (esto lo haremos en el siguiente tema). Wokwi dispone de una pantalla TFT que se puede utilizar. [Aquí](#) se puede ver la documentación. Podéis encontrar ejemplos de uso de esa pantalla [aquí](#) y [aquí](#).