

# Pràctica 4: Sistemes Operatius en Temps Real (FreeRTOS amb ESP32)

## Objectiu

L'objectiu d'aquesta pràctica és comprendre el funcionament d'un sistema operatiu en temps real (RTOS), concretament FreeRTOS, en un entorn ESP32 utilitzant l'entorn Arduino. Es vol veure com es poden executar diverses tasques (multitasking) compartint el temps d'ús de la CPU.

---

## Introducció

FreeRTOS permet dividir el temps del microcontrolador en múltiples tasques que s'executen aparentment alhora. Això és molt útil quan es vol fer funcionar diferents processos independents, com ara encendre un LED mentre es realitza una altra acció, sense bloquejar l'execució del sistema.

---

## Exercici Pràctic 1

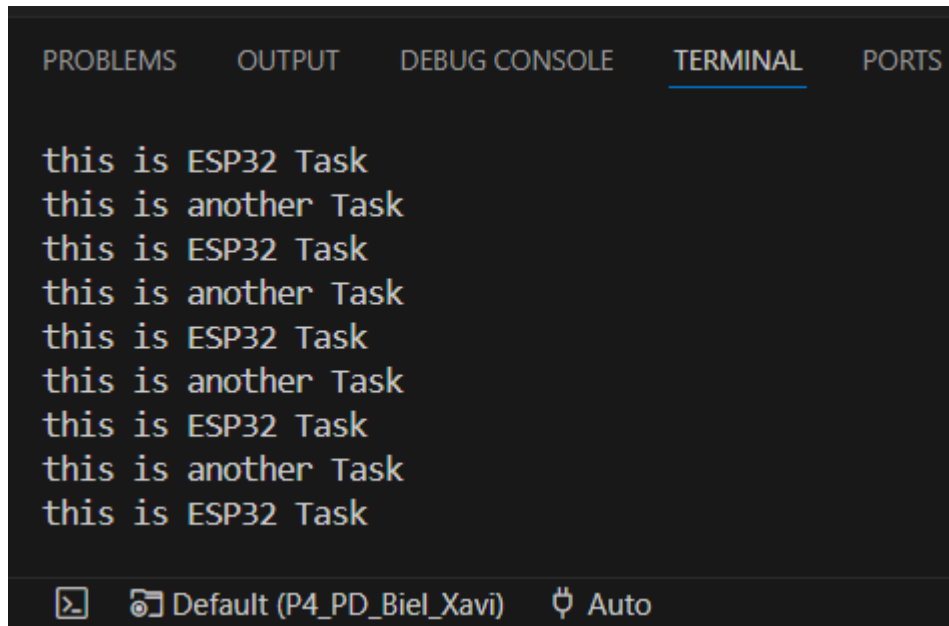
### Codi utilitzat

Has implementat dues tasques:

- La funció `loop()` (task principal)
- La funció `anotherTask()` (task addicional creada amb `xTaskCreate()`)

### Sortida pel port sèrie

La sortida és alternada, amb missatges com:



The screenshot shows an IDE terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The terminal output displays a sequence of messages: "this is ESP32 Task" followed by "this is another Task", repeating five times. At the bottom, the terminal settings are shown as "Default (P4\_PD\_Biel\_Xavi)" and "Auto".

```
this is ESP32 Task
this is another Task
this is ESP32 Task
this is another Task
this is ESP32 Task
this is another Task
this is ESP32 Task
this is another Task
this is ESP32 Task
```

Aquesta alternança mostra com dues tasques s'executen en paral·lel gràcies al planificador de FreeRTOS.

## Explicació del funcionament

- El `loop()` imprimeix "this is ESP32 Task" cada segon.
- La funció `anotherTask()` imprimeix "this is another Task" també cada segon.
- FreeRTOS gestiona l'execució repartint temps entre les dues tasques.

---

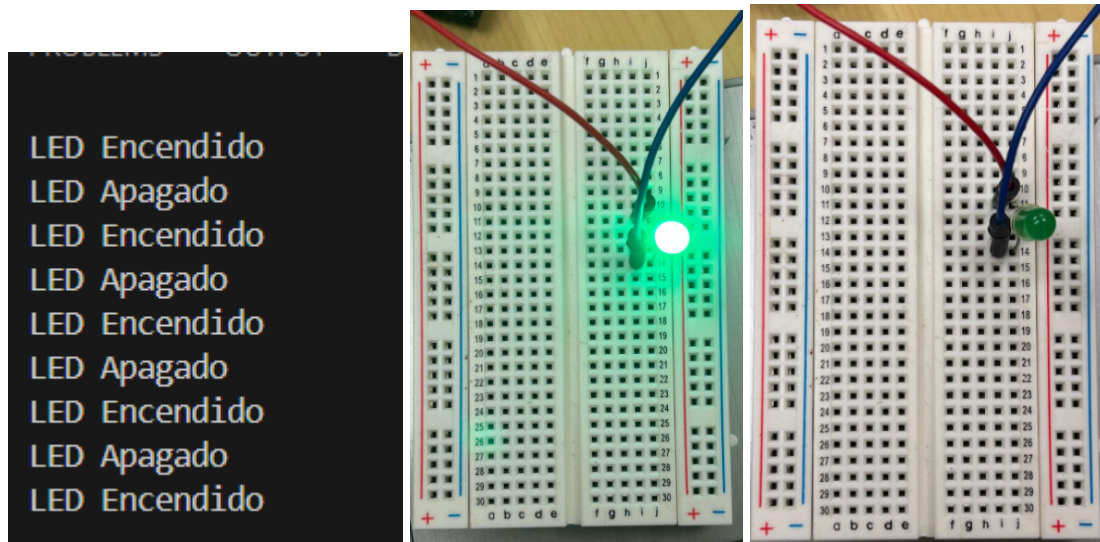
## Exercici Pràctic 2

### Versió amb 2 tasques

#### Codi utilitzat

- Una tasca encén el LED (`taskEncender`)
- L'altra el desactiva (`taskApagarLED`)
- Ambdós sincronitzats amb un **semàfor binari**.

#### Sortida pel port sèrie



## Explicació

El semàfor garanteix que només una tasca accedeix al LED a la vegada, assegurant l'alternança encès/apagat.

---

## Versió amb 3 tasques

### Codi utilitzat

- `task1`: Encén el LED
- `task2`: Apaga el LED
- `task3`: Escriu "Esperando..." (sense acció sobre el LED)
- Utilitza un **semàfor de comptatge** per compartir recursos entre tres tasques.

### Sortida pel port sèrie

```
Tarea 2: LED Apagado  
Tarea 3: Esperando...  
Tarea 1: LED Encendido  
Tarea 2: LED Apagado  
Tarea 3: Esperando...  
Tarea 1: LED Encendido  
Tarea 2: LED Apagado  
Tarea 3: Esperando...  
Tarea 1: LED Encendido
```

### Explicació

Amb el semàfor de comptatge (valor 1), cada tasca accedeix de manera exclusiva al recurs. Es veu com el sistema permet alternar més de dues tasques mantenint l'ordre i evitant conflictes.

---

## Conclusions

- Amb FreeRTOS podem dividir la CPU en diferents tasques de forma eficient.
- L'ús de semàfors és clau per sincronitzar recursos compartits (com un LED).
- La versió amb 3 tasques mostra com FreeRTOS pot gestionar sistemes més complexos sense bloquejos.