PRÀCTICA 4: SISTEMES OPERATIUS EN TEMPS REAL

1. Introducció

L'objectiu d'aquesta pràctica és comprendre el funcionament d'un sistema operatiu en temps real (RTOS) i la gestió de múltiples tasques en un microcontrolador ESP32 utilitzant FreeRTOS. La implementació es realitza mitjançant l'entorn de desenvolupament Arduino, permetent la creació i sincronització de diferents tasques que es comparteixen el temps de CPU.

2. Desenvolupament de la Pràctica

La pràctica es divideix en tres parts amb diferents nivells de complexitat en la gestió de tasques:

2.1 Primera part: Creació d'una tasca addicional

Es crea una tasca independent que imprimeix un missatge al port sèrie en paral·lel amb el loop principal. Aquesta tasca s'executa indefinidament i és gestionada pel planificador de FreeRTOS. L'ús de ***`delay()`*** a cada tasca permet a FreeRTOS alternar entre l'execució de la tasca principal i l'addicional.

Sortida pel port sèrie:

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

Els missatges es mostren intercalats, ja que cada tasca s'executa durant un segon abans de cedir la CPU.

2.2 Segona part

2.2.1 Creació de dues tasques per controlar un LED

Es defineixen dues tasques, una per encendre i una altra per apagar un LED, amb un retard d'un segon entre cada canvi d'estat. FreeRTOS executa aquestes tasques segons la seva disponibilitat de CPU i la prioritat assignada.

Aquest enfocament permet el control independent del LED sense bloquejar el codi principal (***`loop()`***).

Observacions:

- L'ús de ***`vTaskDelay()`*** en comptes de ***`delay()`*** assegura que la CPU pugui executar altres tasques mentre una espera.
- La creació de tasques amb ***`xTaskCreate()`*** garanteix l'execució simultània.

Sortida pel port sèrie:

```
LED ENCENDIDO
LED APAGADO
LED ENCENDIDO
LED APAGADO
LED ENCENDIDO
LED APAGADO
LED ENCENDIDO
LED APAGADO
LED ENCENDIDO
LED APAGADO
LED ENCENDIDO
```

2.2.2 Sincronització de tres tasques mitjançant semàfors

En aquest cas, es creen tres tasques: una per encendre el LED, una d'espera i una per apagar el LED. Aquestes tasques es sincronitzen mitjançant semàfors binaris per garantir que el LED segueixi una seqüència específica.

Funcionament del sistema de semàfors:

1. La tasca ***turnOnLED*** s'activa quan el semàfor corresponent ho permet, encenent el LED i activant el següent semàfor (***xSemaphoreGive(xSemaphoreWait)***).
2. La tasca ***waitTask*** espera que el semàfor anterior es desbloquegi, imprimeix un missatge de "esperant..." i allibera el semàfor següent (***xSemaphoreGive(xSemaphoreOff)***).
3. La tasca ***turnOffLED*** pren el semàfor, apaga el LED i allibera el primer semàfor per repetir el cicle.

Aquest mecanisme assegura que les tasques es realitzin en l'ordre correcte, evitant condicions de carrera.

Sortida pel port sèrie:

```
LED ENCENDIDO
ESPERANDO...
LED APAGADO
LED ENCENDIDO
ESPERANDO...
LED APAGADO
LED ENCENDIDO
ESPERANDO...
LED APAGADO
LED ENCENDIDO
ESPERANDO...
LED APAGADO
```

3. Conclusions

La pràctica permet entendre la gestió de múltiples tasques en un microcontrolador mitjançant FreeRTOS i Arduino. A partir d'aquesta experiència, podem concloure:

- FreeRTOS permet una millor distribució de recursos i facilita la implementació de sistemes concurrents.
- L'ús de ***`vTaskDelay()`*** i semàfors és fonamental per sincronitzar correctament les tasques.
- La programació concurrent en ESP32 és útil per aplicacions IoT on múltiples processos han de funcionar simultàniament sense bloquejar la CPU.

Aquest coneixement és essencial per al desenvolupament de sistemes en temps real, on la gestió eficient del processador és clau per al correcte funcionament del dispositiu.