

## PRÀCTICA 2: INTERRUPCIONS

### 1. Introducció

En aquesta pràctica s'ha treballat amb el concepte d'interrupcions en sistemes embeguts, en particular amb l'ESP32. L'objectiu és comprendre com funcionen les interrupcions per tal d'optimitzar el codi i millorar la resposta del sistema davant d'esdeveniments externs. Es diferencien dos tipus d'interrupcions en aquesta pràctica: per GPIO (botó) i per temporitzador.

### 2. Introducció Teòrica

Una interrupció és un mecanisme que permet interrompre el cicle normal d'execució d'un microcontrolador per atendre un esdeveniment extern o intern. Aquest mecanisme és fonamental per millorar l'eficiència del sistema, evitant l'ús del *polling* (consulta constant de l'estat d'un dispositiu). Les interrupcions es poden classificar en:

- **Interrupcions per esdeveniment extern (Hardware):** Com un polsador o un sensor.
- **Interrupcions per temporitzador (Timer):** Basades en comptadors interns del microcontrolador.
- **Interrupcions per programari (Software):** Tot i que no estan suportades a l'ESP32, es troben en altres sistemes.

L'ESP32 permet configurar qualsevol GPIO com a entrada d'interruptió mitjançant la funció `[attachInterrupt(GPIOPin, ISR, Mode)]`, que defineix el pin a monitoritzar, la funció a executar i la condició de dispar.

## 3. Part A: Interrupció per GPIO

### 3.1 Objectiu

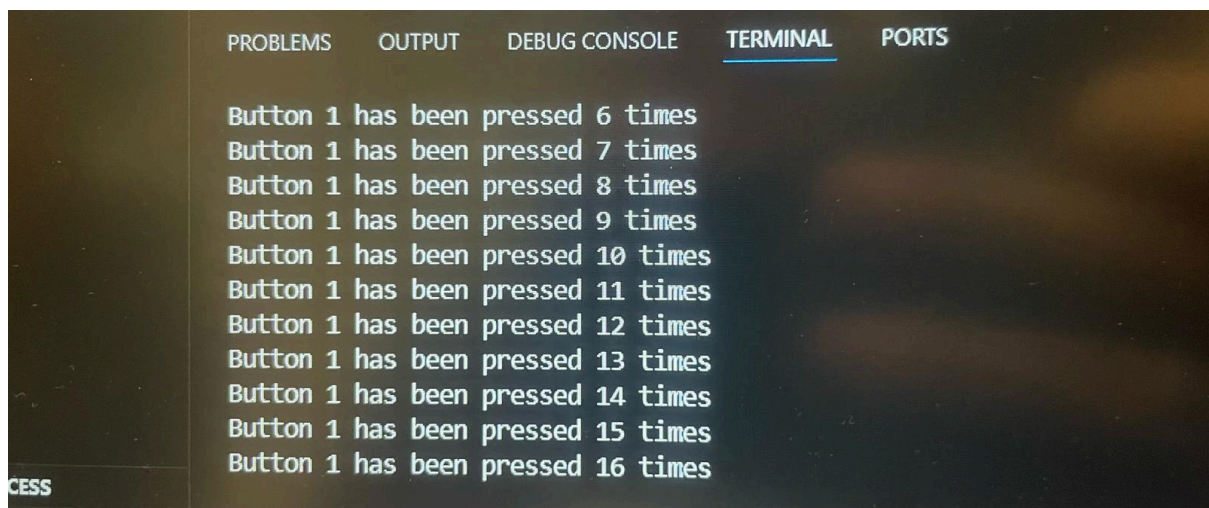
L'objectiu d'aquesta part és implementar una interrupció que detecti la pressió d'un botó i incrementi un comptador cada vegada que es prem.

### 3.2 Funcionament

S'ha utilitzat el GPIO 18 com a entrada d'interrupció. Quan es detecta una transició *FALLING* (baixada de senyal), s'executa la rutina de servei d'interrupció (*ISR*), que incrementa el comptador i actualitza una variable booleana per indicar que hi ha hagut un canvi.

### 3.3 Resultats Obtinguts

En la següent imatge es mostra la sortida obtinguda a la consola sèrie, on es pot veure com es registra cada pressió del botó:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Button 1 has been pressed 6 times
Button 1 has been pressed 7 times
Button 1 has been pressed 8 times
Button 1 has been pressed 9 times
Button 1 has been pressed 10 times
Button 1 has been pressed 11 times
Button 1 has been pressed 12 times
Button 1 has been pressed 13 times
Button 1 has been pressed 14 times
Button 1 has been pressed 15 times
Button 1 has been pressed 16 times

CESS
```

Cada vegada que es prem el botó, el comptador augmenta i s'imprimeix el nombre total de vegades que s'ha activat la interrupció.

## 4. Part B: Interrupció per Temporitzador

### 4.1 Objectiu

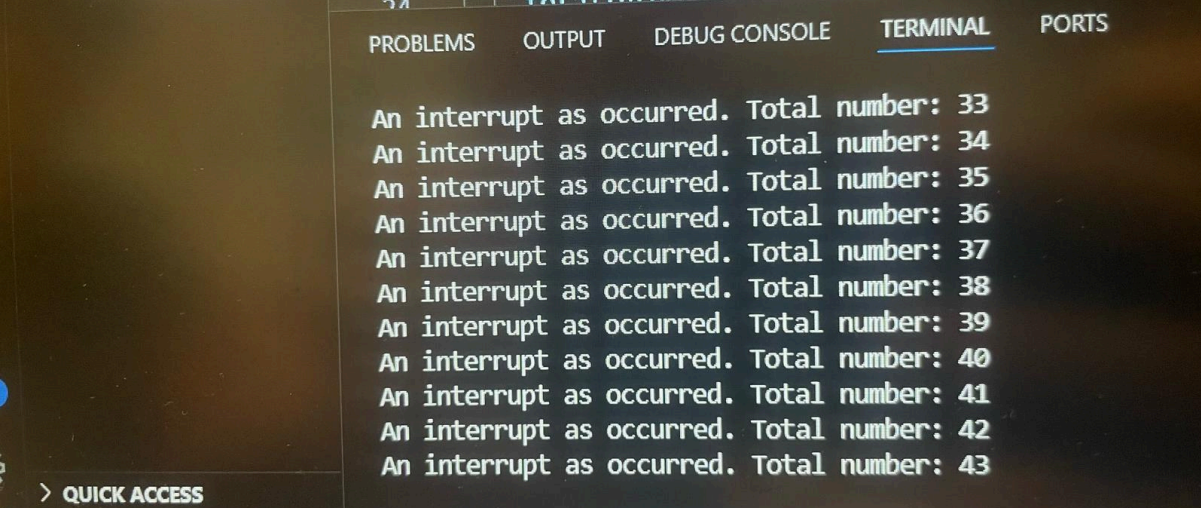
L'objectiu d'aquesta part és generar una interrupció periòdica mitjançant un temporitzador de l'ESP32, per demostrar com gestionar interrupcions sense la intervenció de l'usuari.

### 4.2 Funcionament

S'ha configurat un temporitzador que genera una interrupció cada segon. Cada vegada que es genera la interrupció, s'executa la funció *ISR* que incrementa un comptador global i mostra el total d'interrupcions ocorregudes.

### 4.3 Resultats Obtinguts

En la següent imatge es poden veure els resultats de l'execució del codi, on s'observa que el comptador d'interrupcions s'incrementa periòdicament:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
An interrupt as occurred. Total number: 33
An interrupt as occurred. Total number: 34
An interrupt as occurred. Total number: 35
An interrupt as occurred. Total number: 36
An interrupt as occurred. Total number: 37
An interrupt as occurred. Total number: 38
An interrupt as occurred. Total number: 39
An interrupt as occurred. Total number: 40
An interrupt as occurred. Total number: 41
An interrupt as occurred. Total number: 42
An interrupt as occurred. Total number: 43
> QUICK ACCESS
```

Cada segon, el sistema genera un missatge indicant que ha ocorregut una nova interrupció i s'actualitza el comptador total.

## 5. Conclusions

Aquesta pràctica ha permès comprendre la importància i l'eficiència de les interrupcions en microcontroladors, evitant la consulta contínua de l'estat d'un dispositiu. S'ha pogut observar que:

- Les interrupcions per GPIO són útils per detectar esdeveniments asíncrons, com la pressió d'un botó.
- Les interrupcions per temporitzador permeten executar tasques periòdiques sense bloquejar el funcionament del sistema.
- L'ús de *ISR* ha de ser eficient per evitar problemes de rendiment.

Finalment, es pot concloure que l'ús d'interrupcions és fonamental en sistemes en temps real per millorar la resposta i l'eficiència del codi.