PRACTICA 2 : INTERRUPCIONES

En esta práctica vamos a realizar dos tipos de interrupciones mediante nuestro microprocesador ESP32. En primer lugar haremos un programa que al pulsar un botón analógico se produzca la interrupción y en segundo lugar lo haremos mediante un timer del microprocesador.

B: Interrupción por timer

Código

```
#include <Arduino.h>

volatile int interruptCounter;
int totalCounter;

hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

void IRAM_ATTR onTimer() {
portENTER_CRITICAL_ISR(&timerMux);
interruptCounter++;
portEXIT_CRITICAL_ISR(&timerMux);
}

void setup() {

Serial.begin(9600);
timer = timerBegin(0, 80, true);
timerAttachInterrupt(timer, &onTimer, true);
timerAlarmWrite(timer, 1000000 , true);
timerAlarmEnable(timer);
}

void loop() {

if (interruptCounter > 0) {
portENTER_CRITICAL(&timerMux);
interruptCounter--;
portEXIT_CRITICAL(&timerMux);
totalCounter++;
Serial.print("An interrupt as occurred. Total number: ");
Serial.println(totalCounter);
}

}
```

Funcionamiento del programa

En primer lugar, vamos a declarar las variables "interruptCounter" y "totalCounter". La primera nos servirá para manejar el contador y la segunda funcionará como un contador del número total de interrupciones desde el inicio del programa.

```
volatile int interruptCounter;
int totalCounter;
```

Ahora debemos declarar un puntero "hw_timer_t * timer = NULL;" y por último una variable tipo "portMUX_TYPE" que nos ayude a sincronizar el main loop y la ISR.

```
hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
```

A continuación declaramos la ISR, que funciona de manera que va a contabilizar el número de interrupciones sucedidas.

```
void IRAM_ATTR onTimer() {
portENTER_CRITICAL_ISR(&timerMux);
interruptCounter++;
portEXIT_CRITICAL_ISR(&timerMux);
}
```

Una vez hemos declarado la ISR, ahora vamos a declarar una función llamada Setup que funciona de la siguiente forma: declaramos un timer a una frecuencia de 80MHz ya que la placa ESP32 tiene un reloj que funciona a esa frecuencia. Entonces inicamos el timer y lo configuramos a 1000000 tics/segundo. Seguidamente usamos timerAttachInterrupt para detectar y ejecutar el ISR en cada salto de nuestro timer. Y finalmente habilitamos el contador.

```
void setup() {

Serial.begin(9600);
timer = timerBegin(0, 80, true);
timerAttachInterrupt(timer, &onTimer, true);
timerAlarmWrite(timer, 1000000 , true);
timerAlarmEnable(timer);
}
```

Finalmente encontramos la función Loop en la que vamos a incrementar nuestro contador con el número total de interrupciones "totalCounter" y posteriormente lo imprimiremos por el puerto serie.

```
void loop() {
```

```
if (interruptCounter > 0) {
portENTER_CRITICAL(&timerMux);
interruptCounter--;
portEXIT_CRITICAL(&timerMux);
totalCounter++;
Serial.print("An interrupt as occurred. Total number: ");
Serial.println(totalCounter);
}
}
```

Impresión por puerto serie

Este programa va a imprimir por pantalla el número de veces que encuentre una interrupción. Por lo tanto veríamos algo tal que así:

```
An intrrupt as occurred. Total number: 1
An intrrupt as occurred. Total number: 2
An intrrupt as occurred. Total number: 3
An intrrupt as occurred. Total number: 4
An intrrupt as occurred. Total number: 5
An intrrupt as occurred. Total number: 6
An intrrupt as occurred. Total number: 7
...
```

Y no pararía hasta que no se pare de forma manual.