Johny Silva

PRACTICA 2 : INTERRUPCIONES

Práctica B: Interrupción por temporizador, usando un contador del microprocesador.

B: Interrupción por temporizador

- Header

Se declara las variables *interruptCounter* & *totalInterruptCounter*

interruptCounter: maneja el contador internamente por si ocurre algo que no debería

totalInterruptCounter: cuenta el num total de interrupciones des del inicio del programa.

hw_timer_t declara un temporizador de hardware utilizado para las interrupciones.

Se declara la variable timer de tipo *portMUX_TYPE* para sincronizar el loop principal y el ISR en el manejo de variables compartidas.

```
volatile int interruptCounter;

int totalInterruptCounter;

hw_timer_t * timer = NULL;

portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
```

- ISR

IRAM_ATTR: Contabiliza el número de interrupciones sucedidas Entra y sale del modo CRITICAl para evitar comportamientos erroneos

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

- SETUP

La ESP32 tiene un reloj que trabaja a 80Mhz.

SETUP: ajusta el reloj de 80Mhz de la placa ESP32 al num de tics deseados

Funciona de la siguente mantera: frequency(80Mhz)/escalado(2ndo parametro de la funcion timerBegin)

Temporizador: se inicia y se configura a 1000000 de tics por segundo.

timerAttachInterrupt: para detectar y ejecutar el ISR a cada salto de alarma.

timerAlarmWrite: especifica en que punto del contador la interrupcion debe ser generada. a 1000000 de tics, saltará la alarma y sucederá la interrupción. true llama a reiniciar el contador timer una vez esto haya ocurrido. Finalmente, se habilita el contador.

```
void setup() {

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

- LOOP

LOOP: manipula la interrupción. Si el num de interrupciones > 0 entonces el cont global se reduce. interruptCounter es compartida con el ISR y el loop, por tanto, se ejecuta esta accion como CRITICAL. por último se incrementa el num de interrupciones desde el inicio del programa y se muestra por pantalla.

```
void loop() {

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

}
```