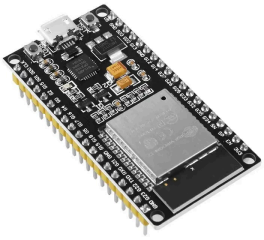
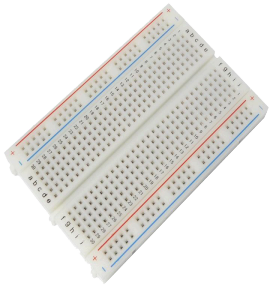
PRACTICA 2 : INTERRUPCIONES

MATERIAL

Para esta practica necesitaremos el microcontrolador ESP32, cables, protoboard (opcional), y un pulsador.

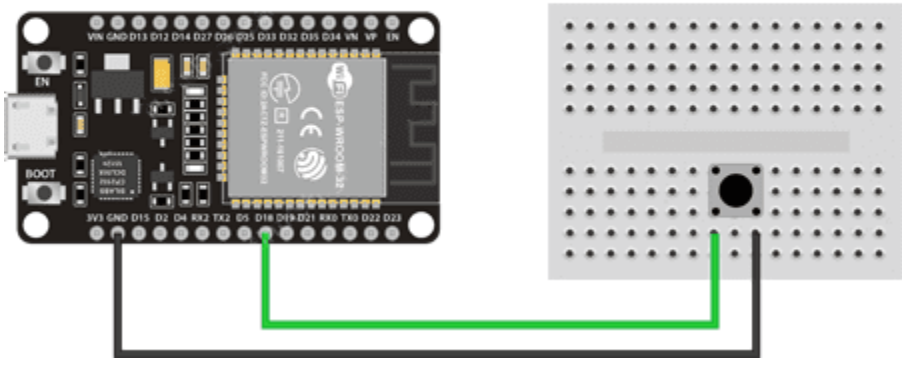
OBJETIVO Y FUNCIONALIDAD DE LA PRACTICA

El objetivo de la practica es comprender el funcionamiento de las interrupciones.

Para lo cual realizaremos una practica donde controlaremos 2 leds de una forma periódica y una entrada ; de forma que el uso de la entrada provoque un cambio de frecuencia de las oscilaciones pero solo en un led.

Practica A interrupción por GPIO

El montaje para esta primera parte es el siguiente:



1. Estructura Button: Define un botón con su pin de conexión, un contador de presiones y

un indicador de si está presionado.

2. Objeto button1: Se crea un botón en el pin 18, con un contador de presiones inicial en 0 y pressed en false.
3. Interrupción (isr()): Cada vez que se presiona el botón, se incrementa el contador de presiones y se marca el botón comopresionado.
4. Configuración (setup()): Se inicializa la comunicación serial, se configura el pin 18 como entrada y se activa la interrupción para ejecutar isr() cuando el botón se presiona.
5. Bucle principal (loop()): Si el botón ha sido presionado, seimprime cuántas veces se ha presionado y luego se restablece elindicador de “presionado”.

Código comentado

```ino
#include <Arduino.h>  // Librería base de Arduino

// Estructura contendrá información sobre un botón
struct Button {
    const uint8_t PIN;            // Define el pin de entrada para el botón.
    uint32_t numberKeyPresses;    // Variable nº veces que presiona el botón. (Variable
    bool pressed;                 // Variable booleana si esta presionado.
};

// Inicialización de una variable de tipo Button llamada button1
Button button1 = {18, 0, false};
// Está conectado al pin 18,
// Inicia con 0 presiones
// Está en estado no presionado.

// ISR (Interrupción de servicio) que se llama cuando ocurre la interrupción
void IRAM_ATTR isr() {
    // Se incrementa cada vez que se detecta un evento de interrupción.
    button1.numberKeyPresses += 1;

    // Se establece que el botón ha sido presionado.
    button1.pressed = true;
}

void setup() {
    // Inicia la comunicación serie por el monitor serial
    Serial.begin(115200);

    delay(1000);  // Un pequeño retraso para asegurar que la comunicación serial se haya

    // Configura el pin del botón como una entrada con resistencia pull-up interna (esto
    pinMode(button1.PIN, INPUT_PULLUP);

    // Asocia una interrupción al pin del botón (pin 18 en este caso), la interrupción s
    attachInterrupt(button1.PIN, isr, FALLING);

    // Imprime un mensaje indicando que el ESP32 ha arrancado correctamente
    Serial.println("ESP32-S3 iniciado corectamnte!");
}
```

```
void loop() {
    // Verifica si el botón ha sido presionado (lo que se indica con 'pressed' como true
    if (button1.pressed) {
        // Imprime el número de veces que el botón ha sido presionado en el monitor seri
        Serial.printf("Button 1 has been pressed %u times\n", button1.numberKeyPresses);

        // Restablece el estado de la variable 'pressed' a false para evitar imprimir re
        button1.pressed = false;
    }

    // Desactivar la interrupción después de 1 minuto (60000 ms)
    static uint32_t lastMillis = 0;  // Variable estática para almacenar el último tiemp

    // Comprueba si ha pasado 1 minuto desde la última vez que se desactivó la interrupc
    if (millis() - lastMillis > 60000) {
        lastMillis = millis();  // Actualiza el tiempo actual
        detachInterrupt(button1.PIN);  // Desactiva la interrupción en el pin del botón
        Serial.println("Interrupt Detached!");  // Imprime un mensaje indicando que la i
    }
}
```

Salida en el monitor serial

- Cuando arranca el sistema aparece un mensaje de inicialización.

> ESP32-S3 iniciado corectamnte!

- Cada vez que presionas el botón aparece un mensaje por pantalla, y la variable totalInterruptCounter va incrementando y la muestra por pantalla.

> Button 1 has been pressed 1 times
> Button 1 has been pressed 2 times
> Button 1 has been pressed 3 times

- Una vez haya transcurrido un minuto, se desactiva la interrupción y imprime un mensaje por pantalla.

> Interrupt Detached!

Practica B interrupción por timer

1. Variables globales: Se declaran dos contadores: interruptCounterpara contar las interrupciones y totalInterruptCounter para llevar untotal de todas las interrupciones. Se configura un temporizador dehardware (hw_timer_t).
2. Interrupción (onTimer()): Se define una función que incrementa el contador de interrupciones cada vez que el temporizador genera una interrupción.
3. Configuración (setup()): Se inicializa la comunicación serial, se configura el temporizador para que se active cada 1 segundo (1,000000 microsegundos) y se habilita la interrupción.
4. Bucle principal (loop()): Cada vez que ocurre una interrupción,se incrementa el contador total y se imprime en el monitor serial elnúmero total de interrupciones que han ocurrido.

Código comentado

```ino
#include <Arduino.h> // Librería base de Arduino

volatile int interruptCounter; // Variable que llevará el nº interrupciones
int totalInterruptCounter = 0; // Guarda el total de interrupciones
hw_timer_t * timer = NULL; // Puntero al temporizador de hardware

// Mutex para la protección de acceso a recursos compartidos
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

// Función que se ejecuta cuando ocurre la interrupción del temporizador
void IRAM_ATTR onTimer() {
  // Entramos en la sección crítica para evitar acceso simultáneo al contador.
  portENTER_CRITICAL_ISR(&timerMux);
  interruptCounter++;   // Incrementa el contador de interrupciones
  portEXIT_CRITICAL_ISR(&timerMux); // Salimos de la sección crítica
}

void setup() {
  // Inicializamos la comunicación serie
  Serial.begin(115200);

  // Inicializamos el temporizador
  timer = timerBegin(0, 80, true);
  // canal 0, prescaler 80, dirección de cuenta hacia abajo

  // Adjuntamos la función onTimer a la interrupción del temporizador
  timerAttachInterrupt(timer, &onTimer, true);
  //La interrupción se llama cada vez que el temporizador se desborda

  // Configuramos el temporizador para que se dispare cada 1 segundo
  timerAlarmWrite(timer, 1000000, true);

  // Habilitamos la alarma para que se active el temporizador
  timerAlarmEnable(timer);
}

void loop() {
  // Si el contador de interrupciones ha sido incrementado
  if (interruptCounter > 0) {
```

```
    // Entramos en la sección crítica para evitar el acceso simultáneo
    portENTER_CRITICAL(&timerMux);

    // Decrementa el contador de interrupciones
    interruptCounter--;
    //(porque se ha procesado una interrupción)

    // Salimos de la sección crítica
    portEXIT_CRITICAL(&timerMux);

    // Incrementa el contador total de interrupciones
    totalInterruptCounter++;

    // Imprime la cantidad total de interrupciones que han ocurrido
    Serial.print("An interrupt as occurred. Total number: ");
    Serial.println(totalInterruptCounter);
  }
}
```

Salida en el monitor serial

1. El temporizador se desborda cada 1 segundo.
2. El contador interruptCounter se incrementa cada vez que ocurre el desbordamiento.
3. El valor de totalInterruptCounter se incrementa cada vez que se procesa una interrupción (cuando el código en el loop() detecta que interruptCounter > 0).
4. Cada vez que se procesa una interrupción, se muestra el mensaje "An interrupt has occurred. Total number: X", donde X es el número total de interrupciones procesadas.

- Después de haber pasado un segundo, el contador de interrupciones se incrementará, y el programa mostrará el mensaje en el monitor serie.

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

```
An interrupt as occurred. Total number: 8
An interrupt as occurred. Total number: 9
An interrupt as occurred. Total number: 10
An interrupt as occurred. Total number: 11
An interrupt as occurred. Total number: 12
An interrupt as occurred. Total number: 13
An interrupt as occurred. Total number: 14
An interrupt as occurred. Total number: 15
An interrupt as occurred. Total number: 16
An interrupt as occurred. Total number: 17
An interrupt as occurred. Total number: 18
An interrupt as occurred. Total number: 19
An interrupt as occurred. Total number: 20
```

Este comportamiento seguiría repitiéndose, siempre aumentando el contador totalInterruptCounter en 1 cada segundo que pasa.

EJERCICIO VOLUNTARIO

Código generado por ChatGPT comentado

```
  #include <Arduino.h>

const int ledPin = 2;        // Pin donde está conectado el LED
unsigned long previousMillis = 0;
int blinkFrequency = 1000;    // Frecuencia inicial (1000ms)
bool ledState = false;

// Variables para simular los pulsadores
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 200;   // Filtro de rebote (200ms)
bool lastButtonStateUp = LOW;
bool lastButtonStateDown = LOW;

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
  Serial.println("Programa iniciado...");
}

void loop() {
  unsigned long currentMillis = millis();

  // Verifica si el monitor serial está funcionando
  Serial.println("Esperando pulsaciones...");

  // Simulamos la presion de un pulsador para aumentar la frecuencia
  if (digitalRead(23) == HIGH && currentMillis - lastDebounceTime > debounceDelay) {
    blinkFrequency = max(100, blinkFrequency - 100);  // Limita la frecuencia mínima a 100
    Serial.print("Frecuencia aumentada: ");
    Serial.print(blinkFrequency);
    Serial.println("ms");
    lastDebounceTime = currentMillis;
  }

  // Simula la presion de otro pulsador para disminuir la frecuencia
  if (digitalRead(22) == HIGH && currentMillis - lastDebounceTime > debounceDelay) {
    blinkFrequency = min(5000, blinkFrequency + 100);  // Limita la frecuencia máxima a 5
    Serial.print("Frecuencia disminuida: ");
    Serial.print(blinkFrequency);
```

```
    Serial.println("ms");
    lastDebounceTime = currentMillis;
  }

  // Control del LED usando el temporizador
  if (currentMillis - previousMillis >= blinkFrequency) {
    previousMillis = currentMillis;

    // Cambia el estado del LED
    ledState = !ledState;
    digitalWrite(ledPin, ledState);
  }
}
```

Como no tenia pulsadores fisicos he adaptado el código para utilizar el teclado del ordenador como si fuera el pulsador y asi poder augmentar o disminuir la frecuencia del parpadeo del LED.

Código adaptado a teclado

```
#include <Arduino.h>

const int ledPin = 2;        // Pin donde está conectado el LED
unsigned long previousMillis = 0;
int blinkFrequency = 1000;    // Frecuencia inicial (1000ms)
bool ledState = false;

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
  Serial.println("Programa iniciado...");
  Serial.println("Usa 'u' para aumentar la frecuencia y 'd' para disminuirla.");
}

void loop() {
  unsigned long currentMillis = millis();

  // Verifica si hay datos disponibles en el Monitor Serial
  if (Serial.available() > 0) {
    char receivedChar = Serial.read();

    // Simula el comportamiento del pulsador UP (aumentar frecuencia)
    if (receivedChar == 'u') {
      blinkFrequency = max(100, blinkFrequency - 100);  // Limita la frecuencia mínima a
      Serial.print("Frecuencia aumentada: ");
      Serial.print(blinkFrequency);
      Serial.println("ms");
    }

    // Simula el comportamiento del pulsador DOWN (disminuir frecuencia)
    if (receivedChar == 'd') {
      blinkFrequency = min(5000, blinkFrequency + 100);  // Limita la frecuencia máxima
      Serial.print("Frecuencia disminuida: ");
      Serial.print(blinkFrequency);
      Serial.println("ms");
    }
  }

  // Control del LED usando el temporizador
```

```
    if (currentMillis - previousMillis >= blinkFrequency) {
      previousMillis = currentMillis;

      // Cambia el estado del LED
      ledState = !ledState;
      digitalWrite(ledPin, ledState);
    }
  }
```

Una vez cargas el programa a la ESP32 y abres el monitor no aparece nada, y su frecuencia se fija en 1000ms.

- Si clicas a la tecla 'u' disminuye el valor 100ms menos y el parpadeo del LED es más rapido.
- Si clicas a la 'd' augmentas el valor 100ms y por lo tanto ralentizas el parpadeo.
  Teneis que hacer que chatgpt os genere un codigo para esp32 arduino

He realizado una prueba en el monitor serie clicando 2 veces a la 'u' y 3 veces a la 'd':

> Frecuencia aumentada: 900ms
> Frecuencia aumentada: 800ms
> Frecuencia disminuida: 900ms
> Frecuencia disminuida: 1000ms
> Frecuencia disminuida: 1100ms

La frecuencia se cambia correctamente.
(He subido los 2 códigos a Github para poder comprobar que funcionan correctamente)