S.E.P. **S.E.S.** **TecNM**

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Aguascalientes

REPORTE DE CODIGO 4 PARA ESP32

Materia:

Arquitectura de Computadoras

Alumnos:

Arantza Darina Gómez Hernández - 23151198

José Andrés Rodríguez Marmolejo - 23151344

Aguascalientes, Ags., 17 de noviembre de 2025

Reporte de Código: Señal SOS, Blink Multicolor y Efecto Pulse seleccionados por Botones además de interrupciones externas (ESP32)

1. Introducción

En este documento se presenta un programa hecho para el ESP32 donde se controlan tres efectos de luz (SOS, Blink y Pulse) mediante botones físicos. Además, se usa una interrupción sencilla para que el microcontrolador pueda reaccionar de inmediato cuando se presiona un botón especial. Este tipo de ejercicios nos ayuda a entender cómo funcionan las interrupciones y cómo controlar un LED RGB de manera práctica.

2. Código

```
// Constantes a las que se les asgina el numero de pin físico que estan
conectados a los botones
int BOTON_SOS = 4;   // Pin 4 conectado al boton que ejecuta 'SOS()'
int BOTON_Blink = 5; // Pin 5 conectado al boton que ejecuta 'Blink()'
int BOTON_Pulse = 6; // Pin 6 conectado al boton que ejecuta 'Pulse()'
int BOTON_Int = 7;   // Pin 7, ejecuta el metodo de interrupcion

// Método de interrupción que se almacena en la RAM(para más velocidad)
// Se ejecutan las instrucciones especificadas dentro del método
void IRAM_ATTR FunInterrup(){
  rgbLedWrite(LED_BUILTIN,0,0,0);
}

void setup() {
  // Comandos para detectar el LED y los botones conectados
  pinMode(LED_BUILTIN, OUTPUT);

  pinMode(BOTON_SOS, INPUT_PULLUP);
  pinMode(BOTON_Blink, INPUT_PULLUP);
  pinMode(BOTON_Pulse, INPUT_PULLUP);
  pinMode(BOTON_Int, INPUT_PULLUP);

  // Función principal que configura la interrupción. Le dice al
microcontrolador que pause su tarea actual
```

```
  // y ejecute una función especial cuando ocurra un evento específico en el pin previamente determinado por la constante BOTON_Int
  // El evento esta conformado por la pulsación del boton de interrupción conectado al pin 7(BOTON_Int) para llamar al método FunInterrup
  attachInterrupt(digitalPinToInterrupt(BOTON_Int),FunInterrup,FALLING);
}

void loop() {
  // Condicionales que detectan los estados de los botones para ejecutar los metodos especificados
  // Si el boton conectado al pin 4(BOTON_SOS) se presiona: Ejecuta el metodo del efecto de destellos SOS
  if (digitalRead(BOTON_SOS) == LOW) {
    SOS();
  }

  // Si el boton conectado al pin 5(BOTON_Blink) se presiona: Ejecuta el metodo del efecto Blink
  else if (digitalRead(BOTON_Blink) == LOW) {
    Blink();
  }

  // Si el boton conectado al pin 6(BOTON_Pulse) se presiona: Ejecuta el metodo del efecto Pulse
  else if (digitalRead(BOTON_Pulse) == LOW) {
    Pulse();
  }
}

void SOS() {
  // Simboliza S en codigo morse
  rgbLedWrite(LED_BUILTIN, 255,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,255,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,255);
  delay(200);

  rgbLedWrite(LED_BUILTIN, 0,0,0); //Apagado del LED para simbolizar el nuevo bloque de parpadeos que representan una letra nueva
  delay(600);
```

```
  // Simboliza O en codigo morse
  rgbLedWrite(LED_BUILTIN, 255,0,0);
  delay(800);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,255,0);
  delay(800);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,255);
  delay(800);

  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(600);

  // Simboliza S en codigo morse
  rgbLedWrite(LED_BUILTIN, 255,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,255,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,255);
  delay(200);

  rgbLedWrite(LED_BUILTIN, 0,0,0);
}

void Blink() {
  rgbLedWrite(LED_BUILTIN, 255,0,0); // Rojo
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 255,165,0); // Naranja
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 255,255,0); //Amarillo
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,255,0); // Verde
```

```
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,255); // Azul
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 25,25,112); // Azul oscuro
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,0);
  delay(200);
  rgbLedWrite(LED_BUILTIN, 128,0,128); // Morado
  delay(200);

  rgbLedWrite(LED_BUILTIN, 0,0,0);
}

void Pulse() {
  for (int r=0; r<=255; r++){
    rgbLedWrite(LED_BUILTIN, r,r,r);
    delay(10);
    }

    for (int r=255; r>=0; r--){
    rgbLedWrite(LED_BUILTIN, r,r,r);
    delay(10);
    }
}
```

3. Explicación general del código

Constantes:

Las líneas iniciales definen tres constantes enteras (BOTON_SOS, BOTON_Blink y BOTON_Pulse) que indican los números de pines digitales del ESP32 a los que están conectados los botones. Cada botón tiene la función de ejecutar una rutina distinta dependiendo de cuál sea presionado. También se define un botón adicional que está conectado a una interrupción, lo cual permite apagar el LED inmediatamente sin importar qué efecto se esté ejecutando.

Función FunInterrupt():

Es una función que define las acciones que se realizara cuando se pulse el botón de interrupción, en este caso se especifica que el LED se apague. El valor de IRAM_ATTR especifica que el método se guarde dentro de la RAM del procesador del ESP32.

Función setup():

Esta función se ejecuta una sola vez al iniciar o reiniciar la placa. Dentro de setup() se configuran los modos de los pines: pinMode(LED_BUILTIN, OUTPUT), el cual indica que el LED será una salida; además de pinMode(BOTON_SOS, INPUT_PULLUP), pinMode(BOTON_Blink, INPUT_PULLUP) y pinMode(BOTON_Pulse, INPUT_PULLUP). Además de que define la función de la interrupción usando attachInterrupt() la cual ejecuta la función FunInterrup(), previamente definida; y asigna esta propiedad al botón especificado (BOTON_Int).

Función loop():

loop() se ejecuta de forma continua, verificando constantemente si alguno de los botones ha sido presionado. Se usa la función digitalRead(pin) para leer el estado lógico del pin: si el valor leído es LOW, significa que el botón fue presionado. Según el botón activado, el programa ejecuta una de las tres funciones: SOS(), Blink() o Pulse().

Función rgbLedWrite(pin, rojo, verde, azul):

Esta función controla los tres componentes de color del LED RGB. Cada parámetro (rojo, verde y azul) acepta valores entre 0 y 255, lo que permite generar una amplia gama de colores combinando distintas intensidades.

Bloque SOS():

Representa la señal internacional de emergencia “SOS” (· · · — — — · · ·) en colores rojo, verde y azul. Los destellos cortos simbolizan los puntos y los largos las rayas. Se usa delay() para establecer la duración de cada destello y los intervalos entre letras.

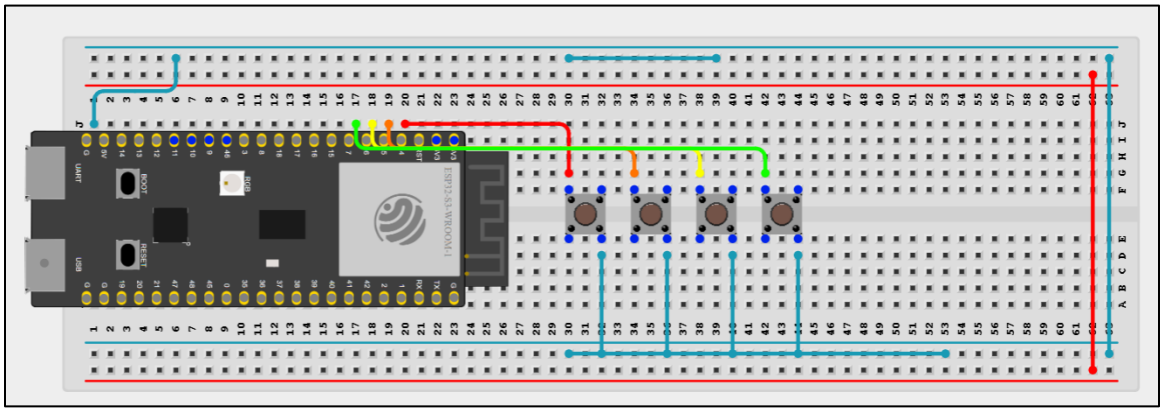
Bloque Blink():

Crea una secuencia de siete colores (rojo, naranja, amarillo, verde, azul, azul oscuro y morado), encendiendo el LED durante 200 milisegundos y apagándolo brevemente entre cada cambio. Este efecto muestra el uso de la función rgbLedWrite() para generar distintos tonos RGB.

Bloque Pulse():

Genera un efecto de 'respiración' o 'pulso luminoso' mediante dos bucles for: el primero incrementa el brillo del LED de 0 a 255, y el segundo lo reduce de 255 a 0. Se usan los mismos valores para los tres canales RGB para obtener luz blanca, cuya intensidad varía suavemente.

4. Diagrama del circuito



5. Conclusión

Con este código aprendimos cómo usar una interrupción básica en el ESP32 para reaccionar rápidamente a un evento externo. También reforzamos la lectura de botones y el control de un LED RGB con diferentes efectos. Este tipo de prácticas nos ayudan a entender cómo funciona el hardware y cómo hacer programas más útiles y dinámicos usando Arduino y ESP32.