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

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Aguascalientes

REPORTE DE CODIGO 2 PARA ESP32

Materia:

Arquitectura de Computadoras

Alumnos:

Arantza Darina Gómez Hernández - 23151198

José Andrés Rodríguez Marmolejo - 23151344

Aguascalientes, Ags., 10 de octubre de 2025

Reporte de Código: Señal SOS Multicolor y Efecto Pulse (ESP32)

1. Introducción

Se propone la idea de crear un programa diseñado con el fin de aprender y analizar las posibilidades de trabajar con un LED RGB en el ESP32; ya sea realizar cambios de diferentes colores para después utilizar estos para representar la señal SOS en código Morse o realizar una secuencia de destellos de varios colores además de recrear un efecto de 'pulso' donde la intensidad del LED sube y baja gradualmente.

2. Código

```
// La función corre una vez cuando se presiona reset o power en el ESP32
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
}

// Función loop para hacer una repetición del codigo
void loop() {
#ifdef LED_BUILTIN

  //rgbLedWrite(valor de rojo, valor de verde, valor de azul) se utiliza
para ajustar los valores de color rojo, verde y azul
  // con el fin de modificar la intensidad y/o el color del LED
  // 255 es el valor que representa la mayor intensidad mientras que 0
representa nada de intensidad o apagado.
  //  - - - P R I M E R  S O S - - -
  // Simboliza un S en codigo morse
  rgbLedWrite(LED_BUILTIN, 255,0,0); // Enciende el LED en color rojo
  delay(200);                 // Espera por 200 milisegundos
  rgbLedWrite(LED_BUILTIN, 0,0,0);  // Apaga el LED
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,255,0); // Enciende el LED en color verde
  delay(200);                 // Espera por 200 milisegundos
  rgbLedWrite(LED_BUILTIN, 0,0,0);  // Apaga el LED
```

```
  delay(200);
  rgbLedWrite(LED_BUILTIN, 0,0,255); // Enciende el LED en color azul
  delay(200);                 // Espera por 200 milisegundos

  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); // Separador de tipos de destello o
acciones diferentes sobre el LED
  delay(1000);

  // - - - 7  B L I N K S - - -
  // 7 destellos en diferentes colores dependiendo de las variaciones
puestas en las entradas de valores rojo, verde y azul
  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);

  digitalWrite(LED_BUILTIN, 0); // Separador de tipos de destello o
acciones diferentes sobre el LED
  delay(1000);

  // - - - S E G U N D O  S O S - - -
  // Simboliza un 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); // Separador de tipos de destello o
acciones diferentes sobre el LED
  delay(1000);

  // - - - P U L S E - - -
  // 2 ciclos for para realizar la subida y la bajada de la intensidad del
brillo del LED
  // r representa un valor unitario que sube o baja, dependiendo de la
situación, por ejemplo; 0,0,0; 1,1,1; 2,2,2; etc.
  // el valor de r representa la intensidad del brillo que se añade a esa
sección; la resta o suma se hacen de manera uniforme
  // para sumar los tres valores de rojo, verde y azul con el fin de
realizar una adición que resulta en el color blanco
  // NOTA: este es el principio de la teoría del color RGB

  int pulseCount = 0; // Se crea una variable que representa un contador
con la finalidad de repetir el proceso 7 veces
```

```
  do {
    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);
    }
    pulseCount++;
  } while(pulseCount < 7); // El ciclo de pulse se repite hasta que el
contador llegue a 7


  digitalWrite(LED_BUILTIN, 0); // Apagado de LED durante un tiempo largo
para simbolizar un reinicio de todo el proceso
  delay(2000);

#endif
}
```

3. Explicación general del código

Externo a loop(): Función setup()

La función setup() se ejecuta una sola vez al iniciar o reiniciar el ESP32. Aquí se deberían inicializar los pines que se usarán como salida, aunque en el código solo se deja un comentario indicando que el pin LED_BUILTIN será una salida.

Inicio: Función loop()

La función loop() se ejecuta de forma continua. Dentro de ella se encuentra todo el código que modifica el comportamiento del LED RGB. La línea #ifdef LED_BUILTIN verifica que el LED esté disponible antes de ejecutar el resto del código.

NOTA: Explicación de rgbLedWrite()

La función rgbLedWrite(pin, rojo, verde, azul) permite controlar los tres componentes de color del LED RGB. Cada valor va de 0 a 255: 0 significa apagado y 255 máxima intensidad. Al combinar estos valores se generan diferentes colores e intensidades de luz.

Bloque 1: Primer SOS en colores rojo, verde y azul

Esta sección representa la señal SOS en código Morse (· · · — — — · · ·). Cada punto (S) usa destellos de duraciones cortas representados con tres colores (rojo, verde y azul), y cada raya (O) usa destellos de más larga duración con los mismos colores. El LED se apaga entre letras para separar las señales que representan las letras. Cambiar cualquiera de los valores del LED (rojo, verde o azul) resulta en intensidades diferentes o la posibilidad de cambiar el color al hacer combinaciones de valores.

Bloque 2: Secuencia de 7 destellos de colores

Después del primer SOS, el LED hace una secuencia de 7 destellos con distintos colores: rojo, naranja, amarillo, verde, azul, azul oscuro y morado. Cada destello dura 200 milisegundos y se apaga brevemente entre cada cambio de color. Esto demuestra el control preciso del color mediante combinaciones de valores RGB; además de como la acción de cambiar cualesquiera de los valores resulta en colores diferentes.

Bloque 3: Segundo SOS

Se repite el mismo primer bloque de código para representar otro SOS con las mismas características de colores y tiempos.

Bloque 4: Efecto Pulse

Esta parte crea un efecto de 'respiración' o 'pulso' del LED. Se usan dos ciclos for: el primero incrementa gradualmente el brillo del LED (de 0 a 255) y el segundo lo disminuye (de 255 a 0). La combinación de mismos valores de los tres colores (rojo, verde y azul) genera el color blanco. El efecto se repite 7 veces, creando un patrón suave de aumento y disminución de luz. Se puede hacer cambios para utilizar solo un valor de color y disminuir o aumentar su intensidad gradualmente; sin embargo, no existe la posibilidad de hacer un color distinto a los bases, esto es dado que se necesita una combinación exacta de valores.

Líneas finales: Reinicio del ciclo

Al final, el LED se apaga durante 2 segundos para indicar el final del ciclo completo. Después de eso, el programa comienza de nuevo desde el principio, repitiendo todo el proceso.

4. Conclusión

Este programa combina el uso del código Morse y los efectos visuales de un LED RGB para mostrar un mensaje de auxilio (SOS) y varios patrones de luz; se utilizó una nueva función denominada rgbLedWrite() que permite controlar el color e intensidad de manera precisa; mientras los ciclos for y do-while permiten generar efectos dinámicos, como el de pulso.

El resultado es una secuencia visual atractiva que demuestra cómo se puede llegar a modificar el control de un LED RGB mediante programación en el ESP32 además de reconocer el propósito del código como una forma de demostrar la posibilidad del control sobre el color y la intensidad del LED utilizando funciones como rgbLedWrite().