

S.E.P.

S.E.S.

TecNM



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO

Instituto Tecnológico de Aguascalientes

REPORTE DE CODIGO 3 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 (ESP32)

1. Introducción

El siguiente proyecto tiene como objetivo implementar tres efectos visuales diferentes en el LED RGB integrado del microcontrolador ESP32, los cuales son seleccionados mediante botones físicos conectados a distintos pines digitales.

Los efectos incluyen una señal SOS en código Morse con colores rojo, verde y azul; un efecto Blink multicolor, que muestra una secuencia de destellos de varios tonos; y un efecto Pulse, que simula una respiración luminosa mediante el aumento y disminución gradual de la intensidad del LED.

Este código busca demostrar el manejo de entradas digitales (botones) y salidas PWM (control de color RGB), así como el uso de estructuras de control como ciclos y condicionales en el entorno de programación de Arduino para ESP32. De esta manera, se refuerzan los conceptos de interacción entre hardware y software dentro de la materia de Arquitectura de Computadoras.

2. Código

```
// Constantes a las que se les asigna 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()'

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);
}
```

```

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();
    }
}

// Efecto de SOS
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);

    // Apagado del LED
    rgbLedWrite(LED_BUILTIN, 0,0,0);
}

// Efecto de blink
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);

    // Apagado del LED
    rgbLedWrite(LED_BUILTIN, 0,0,0);
}

// Efecto de pulse
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.

Externo a loop(): Función setup()

La función setup() se ejecuta una sola vez al iniciar o reiniciar el ESP32.

Constantes:

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) indica que el LED será una salida mientras que pinMode(BOTON_SOS, INPUT_PULLUP), pinMode(BOTON_Blink, INPUT_PULLUP) y pinMode(BOTON_Pulse, INPUT_PULLUP) configuran los pines de los botones como entradas con resistencia pull-up interna, lo que significa que estarán normalmente en estado HIGH (1) y pasarán a LOW (0) al presionarse.

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, verifica que botones son pulsados y ejecuta los métodos correspondientes. 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.

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: 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.

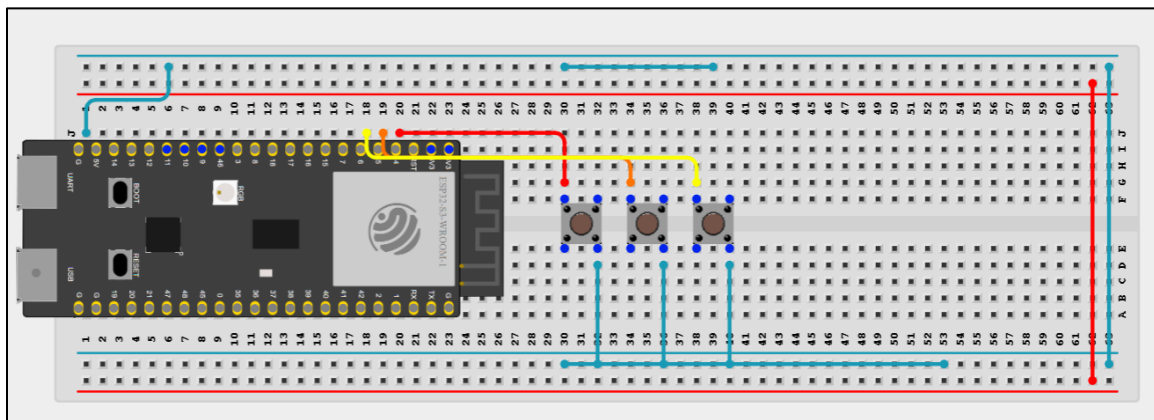
Bloque 2: Secuencia de 7 destellos de colores

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.

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.

4. Diagrama del circuito



5. Conclusión

El proyecto permitió comprender el funcionamiento del ESP32 en la lectura de entradas digitales y el control de salidas RGB mediante programación. A través del uso de botones, se logró seleccionar distintos efectos visuales implementados por software, demostrando el manejo de estructuras condicionales, bucles y retardos. Además, se reforzaron conceptos de interacción hardware–software, como la configuración de pines, el uso de resistencias pull-up internas y la escritura analógica para control de color.

En conjunto, este código muestra una aplicación práctica del microcontrolador ESP32 para generar efectos luminosos interactivos con componentes sencillos.