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

Instituto Tecnológico de Aguascalientes

REPORTE DE CODIGO 1 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 en Morse con LED (ESP32)

1. Introducción

Para esta práctica se busca empezar a generar conocimientos sobre el funcionamiento de un ESP32 y algunas bases para empezar a programar código para este.

Se realizará un programa cuya función sea la introducción al entorno de desarrollo y la creación de un código básico o sencillo que permita el control sobre un LED conectado al ESP32 y hacerlo que parpadee de una manera específica. El parpadeo por realizar se trata de representar la señal SOS en código Morse, un mensaje internacional de auxilio.

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
  int blinkCount = 0; // Variables para iniciar los contadores
  int blinkCount2 = 0;
  int blinkCount3 = 0;

  // Destello del LED 3 veces usando un do-while y un contador
  // Este ciclo simboliza S en codigo morse
  do {
    digitalWrite(LED_BUILTIN, HIGH); // Enciende el LED
    delay(200);                      // Espera por 200 milisegundos
    digitalWrite(LED_BUILTIN, LOW);  // Apaga el LED
    delay(200);                      // Espera por 200 milisegundos
    blinkCount++;                    // Incremento del contador +1
```

```
  } while (blinkCount < 3);     // El ciclo se repite hasta que el
contador tenga un valor de 3

  digitalWrite(LED_BUILTIN, LOW); //Apagado del LED para simbolizar el
nuevo ciclo de parpadeo que representa una letra nueva
  delay(600);

  // Este ciclo simboliza O en codigo morse
  do {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(800);
    digitalWrite(LED_BUILTIN, LOW);
    delay(200);
    blinkCount2++;
  } while (blinkCount2 < 3);

  digitalWrite(LED_BUILTIN, LOW);
  delay(600);

  // Este ciclo simboliza S en codigo morse
  do {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(200);
    digitalWrite(LED_BUILTIN, LOW);
    delay(200);
    blinkCount3++;
  } while (blinkCount3 < 3);

  digitalWrite(LED_BUILTIN, LOW); // Apagado de LED durante un tiempo más
largo en comparación a los que separa cada letra de SOS para simbolizar un
reinicio
  delay(2000);

#endif
}
```

3. Explicación general del código

Externo a loop(): Función setup()

La función setup() se ejecuta una sola vez, justo cuando se enciende o se reinicia el ESP32. Aquí normalmente se configura qué pines serán entradas o salidas; para este caso en concreto se indica que el LED_BUILTIN es la salida.

Inicio: Función loop()

La función loop() se ejecuta una y otra vez sin parar. La línea #ifdef LED_BUILTIN revisa si el LED interno está disponible en la placa.

Primeras líneas: Declaración de variables

Se definen unas variables contadoras, las cuales se usaran para controlar cuántas veces parpadea el LED en cada parte del mensaje (la primera 'S', la 'O' y la segunda 'S'). El valor de inicio del contador es el 0, y puede modificarse para empezar desde otro valor y reducir la cantidad de destellos; sin embargo se recomienda hacerlo dentro de los ciclos donde se aumenta el contador.

Bloque 1: Letra “S” (· · · en Morse)

Dentro de este ciclo do-while se repite el mismo patrón: enciende el LED durante 200 milisegundos, lo apaga otros 200 milisegundos; estos valores pueden ser cambiados para que las acciones de prendido y apagado duren más o menos tiempo según se especifique, aunque hay que tomar en cuenta la letra a representar.

La parte de “do” hace las acciones definidas previamente mientras en “while” se especifica cuantas veces se repite el ciclo. Aquí se aumenta el valor de la variable del contador hasta que llega a 3 y se detiene; se puede cambiar el limite para hacer que el ciclo se repita el número de veces que se desea.

Bloque 2: Pausa entre letras

El LED se apaga durante un poco más de tiempo (600 ms) para separar las letras en el mensaje Morse. Puede cambiarse el tiempo para que haya más o menos diferencia de tiempo entre cada letra, es meramente una elección practica que define un contraste para evitar confusiones y formar el mensaje coherentemente.

Bloque 3: Letra “O” (— — — en Morse)

Nuevamente se utiliza el ciclo do-while que igualmente se repite 3 veces y con otra variable de contador, son embargo esta vez los parpadeos son más largos (800 ms encendido) para representar las líneas que definen la letra O.

Bloque 4: Otra pausa entre letras

Nuevamente una pequeña pausa (600 ms) antes de la siguiente letra para crear coherencia en el mensaje.

Bloque 5: Segunda letra “S”

Se repite el mismo patrón con las mismas especificaciones de parpadeo corto para representar la última 'S'.

Líneas finales: Pausa larga (reinicio del mensaje)

Después de terminar el “SOS”, el LED se apaga durante 2 segundos antes de repetir el mensaje. Esto es igualmente una elección práctica para representar el final del código y su posterior reinicio; puede ser modificado sin problemas de afectar la coherencia del mensaje.

3. Conclusión

Al final se entiende que este programa hace que el LED del ESP32 parpadee para enviar una señal SOS en código Morse. El uso de los ciclos do-while y las variables contadoras permite que cada parte del mensaje (S–O–S) se repita el número correcto de veces.

Gracias a esta práctica se logro entender mejor como utilizar el entorno de Arduino IDE para modificar y cargar el programa para ESP32 además de entender la estructura que puede llevar el código.