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

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Aguascalientes

REPORTE DE CODIGO 6 PARA ESP32

Materia:

Arquitectura de Computadoras

Alumnos:

Arantza Darina Gómez Hernández - 23151198

José Andrés Rodríguez Marmolejo - 23151344

Aguascalientes, Ags., 2 de diciembre de 2025

Reporte de Código: Juego de Buscaminas con entradas de botón y salidas LED (ESP32)

1. Introducción

En este reporte se explica el funcionamiento de un programa desarrollado para el ESP32 que simula un pequeño juego de 'Buscaminas' usando botones como entradas y LEDs como salidas visuales.

El objetivo del juego es encontrar la posición donde se encuentra una mina colocada aleatoriamente, mientras el sistema lleva un conteo de intentos y permite reiniciar la partida mediante un botón especial.

Este proyecto permite comprender el uso de arreglos, entradas digitales, control de LEDs, generación de números aleatorios y organización del código en funciones para crear un programa interactivo.

2. Código

```
// Constantes que especifican los pines para botones y LEDs
int boton1 = 4;
int boton2 = 6;
int boton3 = 15;
int boton4 = 17;
int boton5 = 8;
int boton6 = 46;
int LED1 = 5;
int LED2 = 7;
int LED3 = 16;
int LED4 = 18;
int LED5 = 3;
int LED6 = 9;

int boton_reset = 10;

// Array para almacenar los valores de las 'zonas' donde pueden
encontrarse la mina
// Cada indice del array se le asigna un LED y un boton (Zona donde poder
encontrar la mina)
```

```
// Al boton1 y el LED1 se le asigna el indice 0 y así sucesivamente
int PosMina[6] = {0,0,0,0,0,0};

// Constante para los intentos de encontrar la mina, también llamado
contador
int intentos = 0;

void setup() {
  // Inicializadores de LEDS y botones
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(LED5, OUTPUT);
  pinMode(LED6, OUTPUT);

  pinMode(boton1, INPUT_PULLUP);
  pinMode(boton2, INPUT_PULLUP);
  pinMode(boton3, INPUT_PULLUP);
  pinMode(boton4, INPUT_PULLUP);
  pinMode(boton5, INPUT_PULLUP);
  pinMode(boton6, INPUT_PULLUP);
  pinMode(boton_reset, INPUT_PULLUP);

  // Es una función que reinicia el generador de números aleatorios
pseudoaleatorios con un valor de inicio especificado (la semilla)
  // Se inicializa en el pin 0
  randomSeed(analogRead(0));

  // Metodo inicial que "situa la mina" en alguna posición dentro del
array
  plantarMina();
}

void loop() {
  // Si el boton_reset es presionado, se realiza el metodo de reseteo
  if (digitalRead(boton_reset) == LOW){
    reseteo();
  }

  // Metodos que encienden o mantienen apagado su respectivo LED
dependiendo de si el valor de la posición asignada al metodo es 1 o 0
  // Además de aumentar el número de intentos +1 al presionar el boton
  Z1();
  Z2();
```

```
  Z3();
  Z4();
  Z5();
  Z6();

  // Metodo que constantemente compara la cantidad de intentos respecto al
numero de intentos maximos
  compIntentos(3);
}

// En este metodo se comprueba si el numero de intentos es igual al numero
de intentos maximos establecido como el parametro del metodo
// Si el numero de intentos es igual al parametro de intentosMax, todos
los LED se prenden simbolizando que el jugador perdió
void compIntentos(int intentosMax){
  if (intentos == intentosMax){
    delay(200);
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, HIGH);
    digitalWrite(LED5, HIGH);
    digitalWrite(LED6, HIGH);
  }
}

// Metodo que:
// Vuelve a rellenar el array con 0
// Apaga todos los LED
// Vuelve a establecer la cantidad de intentos realizados
// Llama al metodo de plantarMina para repetir su proceso de randomizado
void reseteo(){
  for (int i = 0; i < 6; i++) {
    PosMina[i] = 0;
  }
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
  digitalWrite(LED5, LOW);
  digitalWrite(LED6, LOW);
  intentos = 0;
  plantarMina();
}
```

```
// Metodo que randomiza una variable de indice entre 0 y 6 (las posiciones disponibles dentro del array)
// Una vez seleccionado el indice, el valor guardado ahí es cambiado por 1; simbolizando así que en esa posición se encuentra la mina
void plantarMina(){
  int indice = random(0, 6);
  PosMina[indice] = 1;
}

// Zona 1
void Z1(){
  // Lee si el boton es presionado
  if (digitalRead(boton1) == LOW){
    // Si el indice 0 dentro del array es 1, el jugador encontro la mina y el LED1 se prende
    // En caso contrario el LED1 se mantiene apagado
    if (PosMina[0] == 1){
      digitalWrite(LED1, HIGH);
    } else {
      digitalWrite(LED1, LOW);
    }
    // Delay necesario para sumar a la variable intentos solo una vez cuando el boton ya no esta siendo presionado
    // Esto para evitar que el loop este constantemente añadiendo valores al contador en la fracción de tiempo que se esta presionando el boton
    delay(500);
    // Suma +1 al contador de intentos
    intentos++;
  }
}

// Zona 2
void Z2(){
  // Lee si el boton es presionado
  if (digitalRead(boton2) == LOW){
    // Si el indice 1 dentro del array es 1, el jugador encontro la mina y el LED2 se prende
    // En caso contrario el LED2 se mantiene apagado
    if (PosMina[1] == 1){
      digitalWrite(LED2, HIGH);
    } else {
      digitalWrite(LED2, LOW);
    }
    // Delay necesario para sumar a la variable intentos solo una vez cuando el boton ya no esta siendo presionado
```

```
    // Esto para evitar que el loop este constantemente añadiendo valores
al contador en la fracción de tiempo que se esta presionando el boton
    delay(500);
    // Suma +1 al contador de intentos
    intentos++;
  }
}

// Zona 3
void Z3(){
  // Lee si el boton es presionado
  if (digitalRead(boton3) == LOW){
    // Si el indice 2 dentro del array es 1, el jugador encontro la mina y
el LED3 se prende
    // En caso contrario el LED3 se mantiene apagado
    if (PosMina[2] == 1){
      digitalWrite(LED3, HIGH);
    } else {
      digitalWrite(LED3, LOW);
    }
    // Delay necesario para sumar a la variable intentos solo una vez
cuando el boton ya no esta siendo presionado
    // Esto para evitar que el loop este constantemente añadiendo valores
al contador en la fracción de tiempo que se esta presionando el boton
    delay(500);
    // Suma +1 al contador de intentos
    intentos++;
  }
}

// Zona 4
void Z4(){
  // Lee si el boton es presionado
  if (digitalRead(boton4) == LOW){
    // Si el indice 3 dentro del array es 1, el jugador encontro la mina y
el LED1 se prende
    // En caso contrario el LED1 se mantiene apagado
    if (PosMina[3] == 1){
      digitalWrite(LED4, HIGH);
    } else {
      digitalWrite(LED4, LOW);
    }
    // Delay necesario para sumar a la variable intentos solo una vez
cuando el boton ya no esta siendo presionado
```

```
    // Esto para evitar que el loop este constantemente añadiendo valores
al contador en la fracción de tiempo que se esta presionando el boton
    delay(500);
    // Suma +1 al contador de intentos
    intentos++;
  }
}

// Zona 5
void Z5(){
  // Lee si el boton es presionado
  if (digitalRead(boton5) == LOW){
    // Si el indice 4 dentro del array es 1, el jugador encontro la mina y
el LED1 se prende
    // En caso contrario el LED1 se mantiene apagado
    if (PosMina[4] == 1){
      digitalWrite(LED5, HIGH);
    } else {
      digitalWrite(LED5, LOW);
    }
    // Delay necesario para sumar a la variable intentos solo una vez
cuando el boton ya no esta siendo presionado
    // Esto para evitar que el loop este constantemente añadiendo valores
al contador en la fracción de tiempo que se esta presionando el boton
    delay(500);
    // Suma +1 al contador de intentos
    intentos++;
  }
}

// Zona 6
void Z6(){
  // Lee si el boton es presionado
  if (digitalRead(boton6) == LOW){
    // Si el indice 5 dentro del array es 1, el jugador encontro la mina y
el LED1 se prende
    // En caso contrario el LED1 se mantiene apagado
    if (PosMina[5] == 1){
      digitalWrite(LED6, HIGH);
    } else {
      digitalWrite(LED6, LOW);
    }
    // Delay necesario para sumar a la variable intentos solo una vez
cuando el boton ya no esta siendo presionado
```

```
    // Esto para evitar que el loop este constantemente añadiendo valores
al contador en la fracción de tiempo que se esta presionando el boton
    delay(500);
    // Suma +1 al contador de intentos
    intentos++;
  }
}
```

3. Explicación general del código

Primeras líneas:

El programa inicia declarando los pines correspondientes para los seis botones de juego, los seis LEDs y el botón de reinicio. También se declara un arreglo llamado PosMina, donde sólo una de las seis posiciones tendrá el valor 1, indicando la ubicación de la mina.

En setup():

Se configuran los LEDs como salidas y los botones como entradas con pull-up. Luego se usa randomSeed(analogRead(0)) para generar diferentes posiciones aleatorias en cada partida.

En loop():

El código revisa si el botón de reinicio fue presionado para llamar a la función reseteo(), la cual limpia el arreglo, apaga los LEDs, reinicia el contador e inicia un nuevo juego. Después se ejecutan las funciones Z1() a Z6(), cada una encargada de detectar si su botón fue presionado, encender su LED si la mina está en esa zona, y sumar un intento con un pequeño delay para evitar múltiples registros.

Función complntentos():

Revisa si el número máximo de intentos fue alcanzado y, de ser así, enciende todos los LEDs para indicar que el jugador perdió.

Función reseteo():

Permite repetir el proceso del juego; después de volver a llenar el arreglo de ceros, apagar todos los LED, reiniciar la variable de intentos a cero y llamar a la función para plantar la mina.

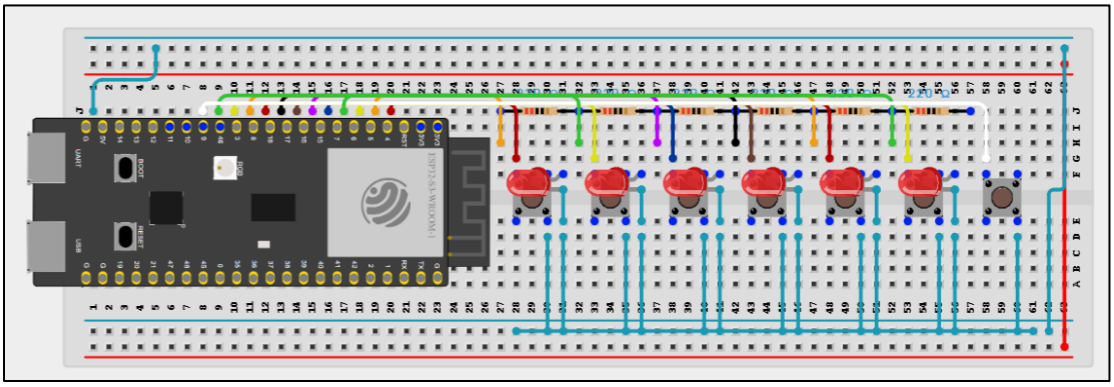
Función plantarMina():

Asegura que cada ronda inicie con una sola mina distribuida aleatoriamente, asignándola por medio de una selección aleatoria de un indice dentro del arreglo y dentro de esta posición se coloca un uno que representa la mina.

Funciones Z1() a Z6():

Comprueban si su respectivo botón es presionado para después verificar si es el índice correspondiente contiene un 1, o sea, una mina. Si se encuentra la mina, el respectivo LED se enciende y, en caso contrario, se queda apagado; ambos casos añaden un +1 al contador de intentos.

4. Diagrama del circuito



5. Conclusión

Gracias a esta practica se pudo observar el comportamiento de la memoria dentro de un microcontrolador ESP32. La memoria guardaba la información del arreglo para la colocación de la mina mientras que paralelamente verificaba si el número de intentos de encontrarla se igualaba al límite previamente configurado. Fue interesante de analizar y realizar al momento de programarlo además de que ayudo a comprender como un procesador puede llevar a cabo varias tareas.