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


EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

**Instituto Tecnológico de Aguascalientes**

# REPORTE DE CODIGO 7 PARA ESP32

**Materia:**

Arquitectura de Computadoras

**Alumnos:**

Arantza Darina Gómez Hernández - 23151198

José Andrés Rodríguez Marmolejo - 23151344

Aguascalientes, Ags., 9 de diciembre de 2025

# Reporte de Código: Juego del Gato con entrada de matriz de botones 4x4 (ESP32)

## 1. Introducción

En este reporte se explica el funcionamiento de un programa desarrollado para la ESP32 que permite jugar el juego del “Gato” utilizando una matriz de botones 4×4 como entrada principal. El jugador humano utiliza las teclas del 1 al 9 del keypad para elegir sus movimientos, mientras que la CPU realiza jugadas generadas de manera aleatoria. El sistema muestra el tablero actual en el monitor serial después de cada movimiento y verifica si hay un ganador o si se produjo un empate. Además, el juego puede ser reiniciado presionando el botón ‘*’.

Este proyecto utiliza conceptos como matrices bidimensionales, lectura de teclas mediante la librería Keypad.h, uso del monitor serial, detección de estados del juego, funciones independientes y lógica de control para juegos simples.

## 2. Código

```
// Libreria para utilizar el keypad externo conectado a la ESP32
#include <Keypad.h>

// Se usaran para definir las dimensiones del teclado, en este caso son 4 filas y 4 columnas
#define filas     4
#define columnas  4

// Matriz que identifica el mapa de teclas, o sea, es una representacion fisica de cada tecla del keypad
char matriz[filas][columnas] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

// Pines para filas y columnas
byte pin_filas[filas]    = {4, 5, 6, 7};
```

```
byte pin_colum[columnas] = {9, 10, 11, 12};

// Establecimiento del objeto teclado de tipo keypad
// Utiliza la matriz previamente definida para ser utilizada como mapa
interno por la libreria e identificar asi cada tecla
Keypad teclado = Keypad(makeKeymap(matriz), pin_filas, pin_colum, filas,
columnas);

// Se establece la estructura del gato donde se juega
char tablero[3][3];
char jugador = 'X';   // Jugador
char cpu     = 'O';   // Computadora
bool juegoTerminado = false; // Booleano que ayuda a identificar si el
juego ha finalizado y no aceptar mas jugadas

void setup() {
  Serial.begin(115200); // se inicia la comunicacion serie a 115200
baudios
  randomSeed(analogRead(0)); // inicia el generador de numeros aleatorios
asignado a un pin no usado,

  limpiarTablero(); // primero se limpia el tablero del gato
  imprimirMatrizOriginal(); // se imprime la matriz original para mostrar
que botones se pueden presionar para jugar
  imprimirTablero(); // muestra el primer estado del tablero
  Serial.println("\nPresiona 1-9 para jugar. * para reiniciar."); //
instruccion de como jugar
}

void loop() {
  char entrada = teclado.getKey(); // revisa si un boton del keypad es
presionado y guarda el caracter correspondiente

  if (!entrada) return;

  // realiza una función de reseteo si el boton * es presionado
  if (entrada == '*') {
    limpiarTablero();
    juegoTerminado = false;
    imprimirMatrizOriginal();
    imprimirTablero();
    Serial.println("\nJuego reiniciado.");
    return;
  }
```

```
  if (juegoTerminado) return;

  // Transcurso del juego que define movimientos del jugador, imprime la
actualizacion
  // correspondiente del tablero despues del movimiento, revisa si hay
victoria o empate y llama
  // al metodo movimientoCPU para que la computadora haga su movimiento
  if (entrada >= '1' && entrada <= '9') {
    // resta 1 a la entrada asginada por el boton presionado para
adecuarlo a las posiciones disponibles
    // dentro de la matriz (0 - 8)
    int pos = entrada - '1';
    // convierte la posicion creada a partir de la entrada del keypad en
coordenadas [x,y]
    int fila = pos / 3;
    int col  = pos % 3;

    // Este if revisa si la casilla esta vacia
    if (tablero[fila][col] == ' ') {
      tablero[fila][col] = jugador;

      imprimirTablero();

      // Revisa si el jugador gana por medio del metodo hayGanador
      if (hayGanador(jugador)) {
        Serial.println("\n¡Ganaste!");
        juegoTerminado = true;
        return;
      }

      if (tableroLleno()) {
        Serial.println("\nEmpate.");
        juegoTerminado = true;
        return;
      }

      // Turno de la computadora
      movimientoCPU();

    } else {
      Serial.println("\nCasilla ocupada.");
    }
  }
}
```

```
// Metodo para limpiar el tablero cuando se pulsa el boton de reseteo (*)
void limpiarTablero() {
  for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
      tablero[i][j] = ' ';
}

// Metodo para mostrar la matriz original (teclas) como guia visual para
que el jugador sepa que botones puede utilizar
void imprimirMatrizOriginal() {
  Serial.println("\n--- MATRIZ DEL JUEGO (TECLAS) ---");
  Serial.println(" 1 | 2 | 3 ");
  Serial.println("---+---+---");
  Serial.println(" 4 | 5 | 6 ");
  Serial.println("---+---+---");
  Serial.println(" 7 | 8 | 9 ");
}

// Metodo para mostrar el estado actual del tablero cada que se realiza un
movimiento,
// independiente de si este movimiento es del jugador o la cpu
void imprimirTablero() {
  Serial.println("\n--- ESTADO ACTUAL ---");
  for (int i = 0; i < 3; i++) {
    Serial.print(" ");
    for (int j = 0; j < 3; j++) {
      Serial.print(tablero[i][j]);
      if (j < 2) Serial.print(" | "); // Separadores entre columnas
    }
    Serial.println();
    if (i < 2) Serial.println("---+---+---"); // Separadores entre filas
  }
}

// Metodo que constantemente verifica si hay ganador, recorriendo filas,
columnas y diagonales
// Se utiliza como parametro un caracter previamente definido (X para el
jugador y O para la CPU)
// para ver si estos hay 3 de estos caracteres seguidos en una linea
vertical, horizontal o diagonal
bool hayGanador(char p) {
  for (int i = 0; i < 3; i++)
    if (tablero[i][0] == p && tablero[i][1] == p && tablero[i][2] == p)
      return true;
```

```
  for (int j = 0; j < 3; j++)
    if (tablero[0][j] == p && tablero[1][j] == p && tablero[2][j] == p)
      return true;

  if (tablero[0][0] == p && tablero[1][1] == p && tablero[2][2] == p)
    return true;

  if (tablero[0][2] == p && tablero[1][1] == p && tablero[2][0] == p)
    return true;

  return false;
}

// Metodo que verifica si el tablero está lleno y definir un empate si es
así
bool tableroLleno() {
  for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
      if (tablero[i][j] == ' ')
        return false;
  return true;
}

// Metodo que define los movimientos de la computadora basandose en la
eleccion de
// casillas aleatorias dentro del trablero
void movimientoCPU() {
  int libres[9]; // arreglo que guarda las posiciones de las casillas
vacias
  int n = 0; // contador de cuantas casillas libres se han encontrado

  // Recorre el tablero para buscar alguna casilla vacia y guarda la
posicion en el arreglo libres[]
  for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
      if (tablero[i][j] == ' ')
        libres[n++] = i * 3 + j;

  if (n == 0) return; // si no hay casillas vacías, no hace nada

  // Elige una casilla al azar y convierte esa eleccion en coordenadas
[x,y]
  int eleccion = libres[random(n)];
  int fila = eleccion / 3;
  int col  = eleccion % 3;
```

```
  tablero[fila][col] = cpu; // usando las coordenadas, se coloca el
caracter asignado al cpu

  imprimirTablero(); // imprime el tablero actualizado despues del
movimiento

  // Verifica si la cpu gano
  if (hayGanador(cpu)) {
    Serial.println("\n¡La computadora gana!");
    juegoTerminado = true;
    return;
  }

  // Revisa el metodo de tableroLleno; si regresa true, se declara un
empate
  if (tableroLleno()) {
    Serial.println("\nEmpate.");
    juegoTerminado = true;
    return;
  }
}
```

## 3. Explicación general del código

**Antes de setup():**

El programa inicia configurando el teclado matricial 4×4 y declarando las variables necesarias para el juego del gato. El tablero se representa como una matriz 3×3 que se limpia al iniciar el programa y cada vez que se presiona '*'. Las teclas del 1 al 9 corresponden directamente a las casillas del tablero.

**En setup():**

Se declaran los comandos que inician la conexión con el serial monitor, el generador de valores aleatorios y las condiciones iniciales del juego como mostrar los botones que se pueden presionar en la matriz de botones para jugar, la

impresión del tablero inicial antes de iniciar y un mensaje de instrucción para el jugador.

**En loop():**

En el loop principal, el programa espera entradas del keypad y ejecuta movimientos válidos. Cuando el jugador realiza una jugada, el tablero se imprime en pantalla y se verifica si ganó o si el tablero se llenó. Si el juego continúa, la CPU selecciona de forma aleatoria una casilla vacía y realiza su movimiento. Todas las verificaciones se manejan mediante métodos como hayGanador(), tableroLleno() y movimientoCPU().

**Método limpiarTablero():**

Este método limpia el tablero donde se colocan los caracteres del jugador (X) y la CPU(O).

**Método imprimirMatrizOriginal():**

Imprime una guía visual en el serial monitor para indicar que botones se pueden presionar y en que celdas ponen el respectivo caracter.

**Método imprimirTablero():**

Imprime el tablero con los respectivos valores que se vayan añadiendo a lo largo del juego, ya sea un espacio vacío o algún carácter que hayan puesto el jugador o la CPU.

**Método hayGanador():**

Recorre la matriz tablero para verificar si hay tres caracteres iguales dentro del tablero que formen una línea, horizontal o vertical; o una diagonal.

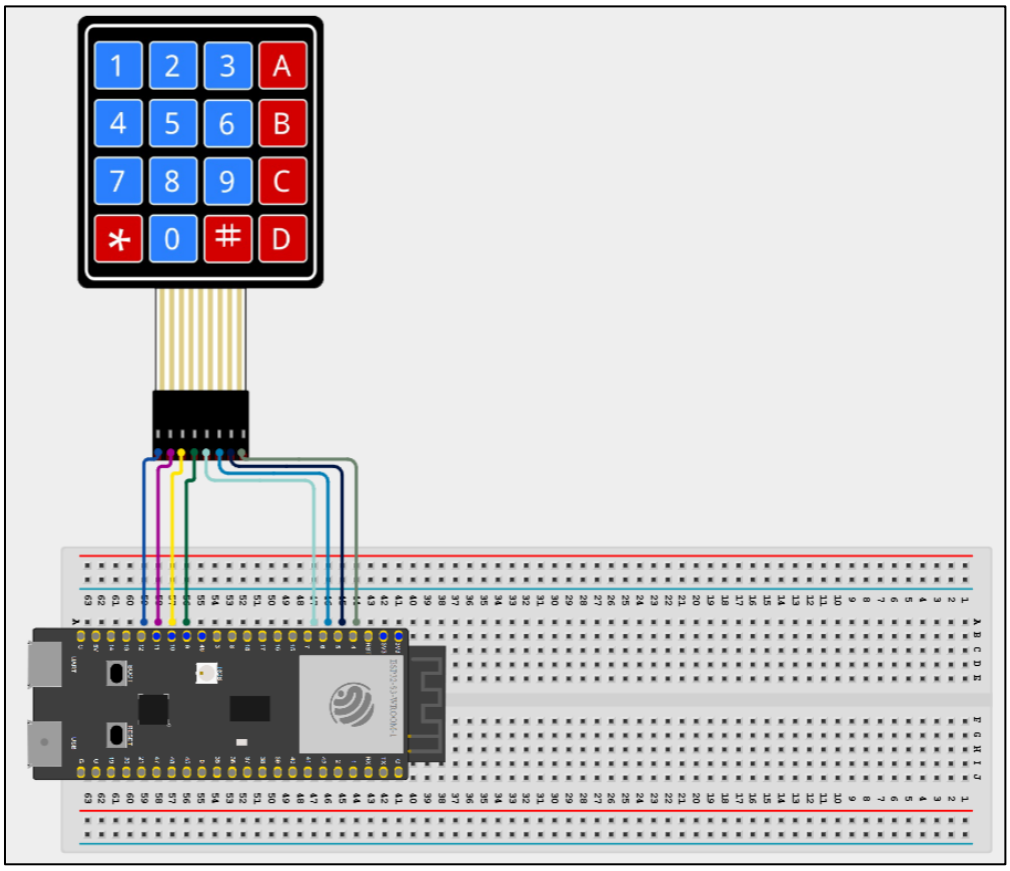
**Método tableroLleno():**

Verifica si la matriz de tablero está llena de caracteres, o sea, sin espacios vacíos. Este método se llama cada vez que alguna de las dos partes jugadoras haga un movimiento para colocar su respectivo caracter.

**Método movimientoCPU():**

Se define el movimiento de la computadora por medio de la aleatoriedad, elije un espacio vacío al azar donde colocar su respectivo caracter; después se imprime el estado actual del tablero y se realizan varias comprobaciones como definir si hay un ganador y revisar si el tablero está lleno para declarar un empate si es así.

## 4. Diagrama del circuito




## 5. Conclusión

Este proyecto permitió comprender el uso de un teclado matricial como entrada digital avanzada en la ESP32, así como la organización del uso de arreglos o matrices dentro de la memoria. El sistema de serial monitor permite visualizar el estado del juego además de presentar un sistema de instrucciones al usuario.

Al final, esta práctica ayudo a reforzar diversos temas como el manejo de arreglos y su impacto dentro de la memoria del microcontrolador; como llegan a afectarla además de afectar su rendimiento al momento de realizar ciertas acciones y guardar cambios provocados por el usuario cuando se hace uso de algún dispositivo de entrada.