Práctica 1 : Tetris usando ESP32

Marlon Sneider Mora Cortes - 20152005034, Sara Valentina Barrero Medina -20191005170
Miguel Ángel Fuentes Ramírez-20182005007
Ingeniería Electrónica, *Universidad Distrital "Francisco José de Caldas"*
Bogotá, Colombia

***Resumen*—El diseño de un juego de tetris empleando ESP32 requiere conceptos como la demultiplexación, codificación, así como requiere manejo de las interrupciones en un microcontrolador, en este caso el ESP32, el concepto de rebote en los pulsadores es algo que puede presentar serios problemas, la visualización en una pantalla LCD hace que el juego sea mas intuitivo,así como la ayuda de un integrado como el 74LS154 para el ahorro de pines.**

***Palabras Clave*–**Esp32, Matriz led, tetris, Almacenamiento en Memoria, Interrupción

I. Introducción

En este informe se presenta la propuesta de solución al proyecto 1 de la asignatura Sistemas Embebidos 1, el cual que consiste en la elaboración de un tetris empleando la tarjeta de desarrollo ESP32, se propone el desarrollo de este juego empleando

II. Formulación del Problema

El problema consiste en diseñar e implementar un juego de tetris con el ESP32, para la visualización del juego se deben utilizar dos matrices de LEDs de 8x8 cada una, se tendrán además cinco pulsadores N.A, también incluirá una LCD alfanumérica para mostrar el puntaje acumulado y el nivel actual del juego.

El juego funcionará de la siguiente manera:

- Se dará inicio al juego con uno de los cinco pulsadores N.A. incluidos en el diseño, inicializando el contador de puntos y el indicador del nivel (se podrán tener hasta cinco niveles de dificultad). Este mismo pulsador luego de iniciado el juego servirá para pausar el juego o para salir de la pausa.
- Con los cuatro pulsadores N.A. restantes se moverán o rotarán las figuras que deberán aparecer de manera aleatoria en la parte superior del tablero de juego. En la figura 1 se muestran las 7 posibles figuras que se podrán generar en el juego.
- Los pulsadores se organizarán de la siguiente manera: uno arriba, otro a la izquierda, otro a la derecha y el último abajo para formar una cruz.
- Los movimientos generados al oprimir los pulsadores serán los siguientes: pulsador de la izquierda movimiento de la figura que cae a la izquierda, pulsador de la derecha movimiento de la figura que cae a la derecha, pulsador superior rotación de la figura, pulsador inferior hace caer rápidamente la figura a la orilla inferior del tablero.
- Se sumará un punto cuando se forme una línea horizontal completa, tres puntos con dos líneas simultáneas, seis puntos con tres líneas y 10 puntos con 4 líneas (valor acumulado que se ira mostrando en la LCD). Se aumentará de nivel a los 20, 50, 100, 1500 y 200 puntos, el aumento del nivel se verá reflejado aumentando la velocidad de caída de las figuras y con un cambio numérico en el número del nivel en la LCD.
- El juego finaliza cuando se acumulen tantas figuras una encima de la otra que no quepan en la pantalla del juego. El valor del record mayor se mostrará siempre en la LCD al lado del puntaje actual.

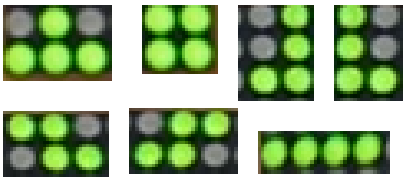

Figure 1: Posibles Figuras

III. Diseño y Modelo de solución

Lo primero que se contemplo para la elaboración del proyecto son los elementos requeridos, en primer lugar se emplearon 2 matrices LED 8×8 1088AS para conformar una matriz de 16x8, dichas matrices cuentan con 16 pines cada una y es por esto que se presenta el primer problema ya que únicamente se pueden compartir columnas entre ambas matrices, por lo que se tendrían aun 24 pines de conexión, 16 correspondientes a las filas y 8 para las columnas, se propone implementar el demultiplexor 4 A 16 (74154), el cual al ser implementado en las filas lograra que solo se usen 4 pines de la ESP para controlarlas, por lo cual solo se requerirán 12 pines para el control de las matrices.

Para el control de los botones encargados del movimiento de las figuras, inicio y pausa del juego, ese emplean pines don PULL-UP interno, se requieren un total de 5 pines para esta tares, finalmente para la pantalla LCD se requieren de 6 pines, por lo que en total se requerirán de 23 pines para el

desarrollo de la practica, estos pines se encuentran resumidos en la tabla I.

Tarea	#Pin	Funcionalidad
Control de Columnas	GPIO12	Controla la columna 1
	GPIO1	Controla la columna 2
	GPIO22	Controla la columna 3
	GPIO3	Controla la columna 4
	GPIO23	Controla la columna 5
	GPIO13	Controla la columna 6
	GPIO14	Controla la columna 7
	GPIO27	Controla la columna 8
Control de Filas usando 74LS154	GPIO2	Controla Selector A
	GPIO16	Controla Selector B
	GPIO5	Controla Selector C
	GPIO17	Controla Selector D
Control de Botones	GPIO21	Controla Pulsador 1
	GPIO34	Controla Pulsador 2
	GPIO19	Controla Pulsador 3
	GPIO18	Controla Pulsador 4
	GPIO35	Controla Pulsador 5
Control LCD	GPIO15	Controla Pin RS
	GPIO4	Controla Pin EN
	GPIO26	Controla Pin D4
	GPIO25	Controla Pin D5
	GPIO33	Controla Pin D6
	GPIO32	Controla Pin D7

Table I: Distribucón Pines Empleados

Para el control de velocidad de cada nivel y que se encarga de controlar la caída de las figuras se emplearon interrupciones por temporizador, empleando estos Timers a medida que se sube el nivel, se tiene un menor valor de conteo, este Timer presenta una complicación don la función de impresión de la matriz, para corregir esto se debe implementar un tiempo mínimo, el cual es de 90 µS.

El juego tiene la capacidad de almacenar el puntaje máximo en la memoria EEPROM, para realizar esto se empleo la librería de , la cual implementa funciones que facilitan la escritura y lectura de esta memoria.

Posteriormente se procedió a desarrollar e implementar con la ayuda del entorno de desarrollo de arduino IDE el código adecuado para el funcionamiento del tetris y el cual podemos en el anexo 1.

IV. Resultados

V. Conclusiones

- Se observa como la tarjeta de desarrollo Esp32 a pesar de contar con grandes características que facilitan el desarrollo de diferentes programas, posee la limitante con respecto al numero de pines de entrada y salida que tiene dicha tarjeta.
- Gracias a que esta tarjeta tiene la posibilidad de ser programada en el entorno de desarrollo correspondiente a Arduino IDE el cual utiliza C como lenguaje de programación y nos facilita diversas librerías que son de gran utilidad al momento de usar diferentes características o componentes externos como por ejemplo la pantalla LCD, acceso a la memoria EEPROM, etc.

VI. Anexos

```
1
2  //#################### 1.Definici n de Librerias ###################
3  using namespace std;
4  #include <LiquidCrystal.h>
5  #include <Preferences.h>
6
7  Preferences preferences;
8  //#################### 2. Definici n de Variables ###################
9
10 //Pines de las Columnas
11
12
13 #define pinC1 12
14 #define pinC2 1
15 #define pinC3 22
16 #define pinC4 3
17 #define pinC5 23
18 #define pinC6 13
19 #define pinC7 14
20 #define pinC8 27
21
22
23
24
25 //Pines de las filas
26 #define pinSA 2
27 #define pinSB 16
28 #define pinSC 5
29 #define pinSD 17
30
31 /*
32   Pines con resistencia de pull-up interna (INPUT_PULLUP):
33   GPIO14, GPIO16, GPIO17, GPIO18, GPIO19, GPIO21, GPIO22 y GPIO23.
34 */
35
36
37 //Pines de los Pulsadores
38
39
40
41 //#define pinP1 16 //Pulsador de Inicio
42 #define pinP1 21 //Pulsador de Inicio
43
44 #define pinP2 34 //Pulsador de Arriba     Sentido = 1
45 #define pinP3 19 //Pulsador de Abajo       Sentido = 2
46 #define pinP4 18//Pulsador de Derecha        Sentido = 3
47 #define pinP5 35 //Pulsador de Izquierda       Sentido = 4
48
49
50 // ----------- PRUEBAS  -------------------
51 //#define LED 2 //Define GPIO2
52
53
54
55 //Pines de la Pantalla
56
57 //Define los pines utilizados del ESP32 de acuerdo a la sintaxis de la libreria
58 const uint8_t RS = 15, EN = 4, D4 = 26, D5 = 25, D6 = 33, D7 = 32;
59
60 //#################### 3.Inicializaci n de Variables ###################
61
62 LiquidCrystal lcd(RS, EN, D4, D5, D6, D7); //Define el objeto "lcd"
63
64 // Memoria
65 unsigned int puntaje_maximo;
66
67
68 //--------------- Pulsadores -----------------
69 volatile uint8_t bandera_P1, bandera_P2, bandera_P3, bandera_P4, bandera_P5 ;
70 volatile uint16_t tiempo_P1, tiempo_P2, tiempo_P3, tiempo_P4, tiempo_P5;
```

```
volatile uint32_t rebote_P1, rebote_P2, rebote_P3, rebote_P4, rebote_P5;
int contador_P1 = 0;
int contador_P2 = 0;
int contador_P3 = 0;
int contador_P4 = 0;
int contador_P5 = 0;

//--------------------------

//Variables Auxiliares
int Nivel = 1;
int Sentido = 0;

//Banderas

bool Pantalla_inicio = true;
bool Seleccion_Nivel = false;
bool Inicio_Juego = false;
bool Pausa_Juego = false;
bool Siguiente_Figura = false;

int Velocidad = 2000000;
int Score = 0;
int num_fig_ale;

int posFil = -1;
int posCol = 0;
int contG = 0;


int Tiempo_Rebote = 250;
int Tiempo_Rebote_sup = 300;
int Tiempo_Rebote_inf = 100;

int x = 0;
int y = 0;
int n = 3;

int Columnas_Encendidas = 0;
int Filas_Llenas = 0;

bool Eliminar_Fila = false;
int Fila_Borrada = 0;
int Filas_Eliminadas = 0;


int limite = 8;
//************* Matrices **********************
volatile int rotation = 0;
//********************* TIMMERS ****************************
volatile uint8_t bandera; //
volatile uint8_t bandera1; //
volatile uint16_t tiempo; //

hw_timer_t *timer1 = NULL;


// Definici n de Figuras
String Figura_Random;

// Figura 1
int L[3][4] = {
  { 1, 0, 0, 0},  // 1
  { 1, 0, 0, 0},  // 2
  { 1, 1, 0, 0},  // 3
};

// Figura 2
int Linv[3][4] = {
  { 0, 1, 0, 0 },  // 1
  { 0, 1, 0, 0 },  // 2
```

```
  { 1, 1, 0, 0 },  // 3
};

// Figura 3
int Z[3][4] = {
  { 1, 1, 0, 0 },  // 1
  { 0, 1, 1, 0 },  // 2
  { 0, 0, 0, 0 },  // 2
};

// Figura 4
int Zinv[3][4] = {
  { 0, 1, 1, 0 },  // 1
  { 1, 1, 0, 0 },  // 2
  { 0, 0, 0, 0 },  // 2
};

// Figura 5
int line[4][4] = {
  { 1, 0, 0, 0 },  // 1
  { 1, 0, 0, 0 },  // 2
  { 1, 0, 0, 0 },  // 3
  { 1, 0, 0, 0 },  // 4
};
int line1[4][4] = {
  { 0, 0, 0, 1 },  // 1
  { 0, 0, 0, 1 },  // 2
  { 0, 0, 0, 1 },  // 3
  { 0, 0, 0, 1 },  // 4
};

// Figura 6
int T[3][4] = {
  { 0, 1, 0, 0 },  // 1
  { 1, 1, 1, 0 },  // 2
  { 0, 0, 0, 0 },  // 3
};

// Figura 7
int sqr[2][4] = {
  { 1, 1, 0, 0 },  // 1
  { 1, 1, 0, 0 },  // 2
};

int mem[17][8] = {
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 1
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 2
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 3
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 4
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 5
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 6
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 7
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 8
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 9
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 10
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 11
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 12
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 13
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 14
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 15
  { 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0 }   // 16
};

int canvas[17][8] = {
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 1
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 2
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 3
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 4
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 5
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 6
```

```
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 7
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 8
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 9
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 10
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 11
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 12
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 13
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 14
  { 0, 0, 0, 0, 0, 0, 0, 0 },  // 15
  { 0, 0, 0, 0, 0, 0, 0, 0 },
  { 1, 1, 1, 1, 1, 1, 1, 1 }    // 16
};




//auxiliar
int figRotate[4][4] = {
  { 0, 0, 0, 0 },  // 1
  { 0, 0, 0, 0 },  // 2
  { 0, 0, 0, 0 },  // 3
  { 0, 0, 0, 0 },  // 4
};


// DEFINICION INTERRUPCIONES :



//********************** PARA PULSADORES **********************
void IRAM_ATTR Int_P1() {
 contador_P1++;
  Rebote_P1();

  // 1 Funcionalidad : Iniciar el Juego
  if (Pantalla_inicio) {
    //Nivel = contador_P1;
    contador_P1 = 0;
    contador_P2 = 0;
    contador_P3 = 0;
    contador_P4 = 0;
    contador_P5 = 0;
    Cambio_Nivel();
    Velocidad_Nivel();
    Seleccion_Nivel = false;
    Pantalla_inicio = false;
    Inicio_Juego = true;
    delay(1000);
    Pantalla_Juego(Nivel);
    lcd.setCursor(11, 1);
    lcd.print(puntaje_maximo);
  }

  // 2 Funcionalidad : Boton de Pausa

  if (Inicio_Juego) {
    //Se activa la Funcionalidad de Pausa y Despausa
    if (contador_P1 > 0) {

      if (Pausa_Juego) {
        Pausa_Juego = false;
        Pantalla_Juego(Nivel);
      } else {
        Pausa_Juego = true;
        Desactivar_Timer();
        Pantalla_Pausa();
      }
    }

```

```
  }

}
void IRAM_ATTR Int_P2() {
  contador_P2++;
  Rebote_P2();
  // 1 Funcionalidad : Boton de arriba
  if (Inicio_Juego) {

    if (Pausa_Juego) {

    } else {
      doRotable();
      uint8_t t = 0;
      switch (num_fig_ale) {
        case 1:
          rotacion("L");
          t = 3;
          createRotation(L, t, "L", rotation);
          break;
        case 2:
          rotacion("Linv");
          t = 3;
          createRotation(Linv, t, "Linv", rotation);
          break;
        case 3:
          rotacion("Z");
          t = 3;
          createRotation(Z, t, "Z", rotation);
          break;
        case 4:
          rotacion("Zinv");
          t = 3;
          createRotation(Zinv, t, "Zinv", rotation);
          break;
        case 5:
          rotacion("Line");
          t = 4;
          createRotation(line, t, "Line", rotation);
          break;
        case 6:
          rotacion("T");
          t = 3;
          createRotation(T, t, "T", rotation);
          break;
        case 7:
          rotacion("Cube");
          t = 2;
          createRotation(sqr, t, "sqr", rotation);
          break;
      }
      Sentido = 1; //Arriba
      clearmem();
      printFig(posFil, posCol, t);
    }
  }
}
void IRAM_ATTR Int_P3() {
  contador_P3++;
  Rebote_P3();
  // 1 Funcionalidad : Boton Abajo
  if (Inicio_Juego) {
    if (Pausa_Juego) {
      Sentido = 0;
    } else {
      Sentido = 2; //Abajo
      if (posFil < 16) {
        posFil = posFil + 1;
      }
    }
  }
```

```
}

//void IRAM_ATTR Int_P4() {
//  contador_P4++;
//  Rebote_P4();
//  // 1 Funcionalidad : Boton Derecha
//  if (Inicio_Juego) {
//    if (Pausa_Juego) {
//      Sentido = 0;
//    } else {
//      Sentido = 3; //Derecha
//        if (posCol < 7) {
//          posCol = posCol + 1;
//        }
//
//    }
//  }
//}

void IRAM_ATTR Int_P4() {
  contador_P4++;
  Rebote_P4();
  // 1 Funcionalidad : Boton Derecha
  if (Inicio_Juego) {
    if (Pausa_Juego) {
      Sentido = 0;
    } else {
      Sentido = 3; //Derecha
      if (num_fig_ale == 1) { // L
        if (rotation == 0 || rotation == 2) {
            if (posCol < 6 ) {
              posCol++;
            }
        }else{
          if (posCol < 5 ) {
              posCol++;
            }
        }
      }else if (num_fig_ale == 2) { // L Linv
        if (rotation == 0 || rotation == 2) {
            if (posCol < 7 ) {
              posCol++;
            }
        }else{
          if (posCol < 6 ) {
              posCol++;
            }
        }
      }else if (num_fig_ale == 6) { //T
        if (rotation == 1 || rotation == 3) {
            if (posCol < 6 ) {
              posCol++;
            }
        }else{
          if (posCol < 5 ) {
              posCol++;
            }
        }
      }else if (num_fig_ale == 3) { // Z
        if (rotation == 1) {
            if (posCol < 6 ) {
              posCol++;
            }
        }else{
            if (posCol < 5 ) {
              posCol++;
            }
        }
      }else if (num_fig_ale == 4) { // Zinv
        if (rotation == 0) {
            if (posCol < 5 ) {
```

```
              posCol++;
            }
        }else{
            if (posCol < 6 ) {
              posCol++;
            }
        }
      }else if (num_fig_ale == 5) {// Line
        if (rotation == 0) {
            if (posCol < 7) {
              posCol++;
            }
        }else{
          if (posCol < 4 ) {
            posCol++;
          }
        }
      }else if (num_fig_ale == 7) {// Square
        if (posCol < 6 ) {
          posCol++;
        }
      }
    }
  }
}


void IRAM_ATTR Int_P5() {
  contador_P5++;
  Rebote_P5();
  // 1 Funcionalidad : Boton Izquierda
  if (Inicio_Juego) {
    if (Pausa_Juego) {
      Sentido = 0;
    } else {
      Sentido = 4; //Izquierda
      if (posCol > 0) {
        posCol = posCol - 1;
      }
    }
  }
}
//****************** TIMMERS ******************************

void IRAM_ATTR Int_Timer1() {
  Generacion_Figura();
  /*createRotation(line, 4, "Line", rotation);
  clearmem();
  printFig(posFil, posCol, 4);*/
  Siguiente_Figura = true;
}


void setup() {
  lcd.begin(16, 2); //Configurar LCD de tama o 16x2

  //Configuracion Pulsadores
  pinMode(pinP1, INPUT_PULLUP); //Configurar GPIO16 como entrada para leer estado de pulsador N.A.
  pinMode(pinP2, INPUT_PULLUP); //Configurar GPIO17 como entrada
  pinMode(pinP3, INPUT_PULLUP); //Configurar GPIO18 como entrada
  pinMode(pinP4, INPUT_PULLUP); //Configurar GPIO19 como entrada
  pinMode(pinP5, INPUT_PULLUP); //Configurar GPIO21 como entrada

  //Configuracion de las Columnas
  pinMode(pinC1, OUTPUT);
  pinMode(pinC2, OUTPUT);
  pinMode(pinC3, OUTPUT);
  pinMode(pinC4, OUTPUT);
  pinMode(pinC5, OUTPUT);
  pinMode(pinC6, OUTPUT);
  pinMode(pinC7, OUTPUT);
```

```
  pinMode(pinC8, OUTPUT);

  //Configuracion de las Filas
  pinMode(pinSA, OUTPUT);
  pinMode(pinSB, OUTPUT);
  pinMode(pinSC, OUTPUT);
  pinMode(pinSD, OUTPUT);

  //Configuraci n Interrupciones Pulsadores
  attachInterrupt(digitalPinToInterrupt(pinP1), Int_P1, FALLING); //Habilita int. generada por el GPIO16
  attachInterrupt(digitalPinToInterrupt(pinP2), Int_P2, FALLING); //Habilita int. generada por el GPIO16
  attachInterrupt(digitalPinToInterrupt(pinP3), Int_P3, FALLING); //Habilita int. generada por el GPIO16
  attachInterrupt(digitalPinToInterrupt(pinP4), Int_P4, FALLING); //Habilita int. generada por el GPIO16
  attachInterrupt(digitalPinToInterrupt(pinP5), Int_P5, FALLING); //Habilita int. generada por el GPIO16
  tiempo_P1 = Tiempo_Rebote; tiempo_P2 = Tiempo_Rebote; tiempo_P3 = Tiempo_Rebote; tiempo_P4 = Tiempo_Rebote; tiempo_P5 = Tiempo_Rebote;
  bandera_P1 = 0; bandera_P2 = 0; bandera_P3 = 0; bandera_P4 = 0; bandera_P5 = 0;


  bandera = 0; tiempo = 0; bandera1 = 0;
  timer1 = timerBegin(1, 80, true); //Configura Timer 1 con preesc.x80 (=1MHz) y cambio por flanco de bajada
  //Serial.begin(115200);
  Lectura_memoria();

}

void loop() {
  /*lcd.setCursor(0, 0); //Ubicar cursor en la columna 3 fila 0
    lcd.print("Maximo : "); //Imprimir mensaje en la LCD
    lcd.print(puntaje_maximo);*/
  //

  if (Pantalla_inicio) {
    Pantalla_Inicio();
  }
  if (Inicio_Juego) {
    contador_P1 = 0;
    contador_P2 = 0;
    contador_P3 = 0;
    contador_P4 = 0;
    contador_P5 = 0;
    if (Pausa_Juego) {
    } else {
      if (posFil == -1) {
        num_fig_ale = random(1, 8);
        posCol = 4;
        rotation = 0;
        perdiste();
      }
      Activar_Timer();
      printDisplay();
    }
  }

  Pulsadores();

}

void perdiste() {
  for (int c = 0; c < 8; c++) {
    if ((canvas[1][c]) == 1) {
      Desactivar_Timer();
      lcd.clear();
      lcd.setCursor(3, 0); //Ubicar cursor en la columna 3 fila 0
      lcd.print(" PERDISTE "); //Imprimir mensaje en la LCD
```

```
561         clearcanvas();
562         delay(1000);
563         Pantalla_Juego(Nivel);
564         if (Score > puntaje_maximo) {
565           puntaje_maximo = Score;
566           Escritura_memoria();
567         }
568         Score = 0;
569         Nivel = 1;
570       }
571     }
572 }
573
574 void Cambio_Nivel() {
575   if (Score >= 20 && Score < 50) {
576     //Entrar a nivel 2
577     Nivel = 2;
578   } else if (Score >= 50 && Score < 100) {
579     //Entrar a nivel 3
580     Nivel = 3;
581   } else if (Score >= 100 && Score < 150) {
582     //Entrar a nivel 4
583     Nivel = 4;
584   } else if (Score >= 150) {
585     //Entrar a nivel 5
586     Nivel = 5;
587   } else {
588     Nivel = 1;
589   }
590 }
591
592 void doRotable() {
593   if (num_fig_ale == 1 || num_fig_ale == 2) { // L Linv
594     if (posCol == 6 ) {
595       if (rotation == 0 || rotation == 2) {
596           posCol--;
597       }
598     }
599   }else if (num_fig_ale == 6) { //T
600     if (posCol == 6 ) {
601       if (rotation == 1 || rotation == 3) {
602           posCol--;
603       }
604     }
605   }else if (num_fig_ale == 3 || num_fig_ale == 4) { // Z Zinv
606     if (posCol == 6 ) {
607       if (rotation == 1) {
608           posCol--;
609       }
610     }
611   }else if (num_fig_ale == 5) {// Line
612     if (posCol >= 5) {
613       if (rotation == 0) {
614           //posCol = posCol - (posCol-4);
615       }
616     }
617   }
618 }
619
620 void Lectura_memoria() {
621   preferences.begin("puntaje_maximo", false);
622   puntaje_maximo = preferences.getUInt("puntaje_maximo", 0);
623   preferences.end();
624
625 }
626 void Escritura_memoria() {
627   preferences.begin("puntaje_maximo", false);
628   preferences.putUInt("puntaje_maximo", puntaje_maximo);
629   preferences.end();
630 }
631
```

```
632 void Activar_Timer() {
633
634   timerAttachInterrupt(timer1, Int_Timer1, true); //Hab. int del Timer 1
635   timerAlarmWrite(timer1, Velocidad, true); //Timer 1 genera 1.000.000 de conteos (=1s) con
        autorecarga
636   timerAlarmEnable(timer1); //Habilitar el Timer 1
637 }
638 void Desactivar_Timer() {
639   timerRestart(timer1);
640   timerAlarmDisable(timer1); //Habilitar el Timer 1
641 }
642
643 int checkRowAll() {
644   for (int f = 0; f < 16; f++) {
645     if (canvas[f][0] & canvas[f][1] & canvas[f][2] & canvas[f][3] & canvas[f][4] & canvas[f][5] &
      canvas[f][6] & canvas[f][7]) {
646       return f;
647     }
648   }
649   return -1;
650 }
651
652 void downCanvas(int row) {
653   for (int f = row; f > 0; f--) {
654     canvas[f][0] = canvas[f - 1][0];
655     canvas[f][1] = canvas[f - 1][1];
656     canvas[f][2] = canvas[f - 1][2];
657     canvas[f][3] = canvas[f - 1][3];
658     canvas[f][4] = canvas[f - 1][4];
659     canvas[f][5] = canvas[f - 1][5];
660     canvas[f][6] = canvas[f - 1][6];
661     canvas[f][7] = canvas[f - 1][7];
662   }
663   for (int c = 0; c < 8; c++) {
664     canvas[0][c] = 0;
665   }
666 }
667
668 void getScore() {
669   uint8_t lines = 0;
670   int row = 0;
671   do {
672     row = checkRowAll();
673     if (row != -1) {
674       downCanvas(row);
675       lines++;
676     }
677   } while (row != -1);
678   if (lines > 0) {
679     switch (lines) {
680       case 1:
681         Score += 1;
682         break;
683       case 2:
684         Score += 3;
685         break;
686       case 3:
687         Score += 6;
688         break;
689       case 4:
690         Score += 10;
691         break;
692       default:
693         Score += 10;
694         break;
695     }
696     lcd.setCursor(7, 1);
697     lcd.print(Score);
698     Cambio_Nivel();
699     lcd.setCursor(7, 0);
700     lcd.print(Nivel);
```

```
  }

}


void Velocidad_Nivel() {

  switch (Nivel) {
    case 1:
      Velocidad = 500000;
      break;
    case 2:
      Velocidad = 400000;
      break;
    case 3:
      Velocidad = 350000;
      break;
    case 4:
      Velocidad = 300000;
      break;
    case 5:
      Velocidad = 250000;
      break;
  }
}

void Generacion_Figura() {
  switch (num_fig_ale) {
    case 1:
      Figura_Random = "L";
      stepDown(L, 3, "L", rotation);
      break;
    case 2:
      Figura_Random = "Linv";
      stepDown(Linv, 3, "Linv", rotation);
      break;
    case 3:
      Figura_Random = "Z";
      stepDown(Z, 3, "Z", rotation);
      break;
    case 4:
      Figura_Random = "Zinv";
      stepDown(Zinv, 3, "Zinv", rotation);
      break;
    case 5:
      Figura_Random = "Line";
      stepDown(line, 4, "Line", rotation);
      break;
    case 6:
      Figura_Random = "T";
      stepDown(T, 3, "T", rotation);
      break;
    case 7:
      Figura_Random = "Cube";
      stepDown(sqr, 2, "Cube", rotation);
      break;
  }
}


void Pulsadores() {
  if (bandera_P1) {
    Pulsador1_();
  } else if (bandera_P2) {
    Pulsador2_();
  } else if (bandera_P3) {
    Pulsador3_();
  } else if (bandera_P4) {
    Pulsador4_();
  } else if (bandera_P5) {
    Pulsador5_();
```

```
  }
}

//-----------------------------------------------FUNCIONES MATRICES -------------------------------

bool rowIsUsed(uint8_t row) {
  for (int col = 0; col < 8; col++) {
    if (canvas[row][col] == 1) return true;
  }
}

void setOFF() {
  digitalWrite(pinC1, LOW);
  digitalWrite(pinC2, LOW);
  digitalWrite(pinC3, LOW);
  digitalWrite(pinC4, LOW);
  digitalWrite(pinC5, LOW);
  digitalWrite(pinC6, LOW);
  digitalWrite(pinC7, LOW);
  digitalWrite(pinC8, LOW);
}

void setCol(int col) {
  digitalWrite(pinC1, col & 0x01);
  digitalWrite(pinC2, col & 0x02);
  digitalWrite(pinC3, col & 0x04);
  digitalWrite(pinC4, col & 0x08);
  digitalWrite(pinC5, col & 0x10);
  digitalWrite(pinC6, col & 0x20);
  digitalWrite(pinC7, col & 0x40);
  digitalWrite(pinC8, col & 0x80);
}

void setFil(int fil) {
  digitalWrite(pinSA, (15 - fil) & 0x01);
  digitalWrite(pinSB, (15 - fil) & 0x02);
  digitalWrite(pinSC, (15 - fil) & 0x04);
  digitalWrite(pinSD, (15 - fil) & 0x08);
}

void printDisplay() {
  int f = 0;
  int c = 1;
  for (int fil = 0; fil < 16; fil++) {
    for (int col = 0; col < 8; col++) {
      if (canvas[fil][col] == 1 || mem[fil][col] == 1) {
        setCol(c);
        setFil(fil);
        delayMicroseconds(100);
      }else{
        setOFF();
      }
      c <<= 1;
    }
    c = 1;
  }
}
void clearmem() {
  for (int i = 0; i < 16; i++) {
    for (int j = 0; j < 8; j++) {
      mem[i][j] = 0;
    }
  }

}
void clearcanvas() {
  for (int i = 0; i < 16; i++) {
    for (int j = 0; j < 8; j++) {
      canvas[i][j] = 0;
    }
  }
```

```
843
844 }
845 void stepDown(int fig[][4], int t, string figName, int rotation) {
846   createRotation(fig, t, figName, rotation);
847   if (check_printFig(posFil + 1, posCol, t)) {
848     posFil++;
849     clearmem();
850     printFig(posFil, posCol, t);
851   } else {
852     mem_to_canvas();
853     posFil = -1;
854   }
855   getScore();
856 }
857
858 void createRotation(int fig[][4], int t, string figName, int rotation) {
859   clear_figRotate();
860   for (int i = 0; i < t; i++) {
861     for (int j = 0; j < t; j++) {
862       if (figName == "L" or figName == "Linv" or figName == "T") {
863         switch (rotation) {
864           case 0:
865             figRotate[i][j] = fig[i][j];
866             break;
867           case 1:
868             if(figName == "L" and i !=0){
869               figRotate[i-1][j] = fig[j][t - 1 - i];
870             }else if(figName == "Linv" and i !=0){
871               figRotate[i-1][j] = fig[j][t - 1 - i];
872             }else{
873               figRotate[i][j] = fig[j][t - 1 - i];
874             }
875             break;
876           case 2:
877             if(figName == "L" and j !=0){
878               figRotate[i][j-1] = fig[t - 1 - i][t - 1 - j];
879             }else if(figName == "Linv" and j !=0){
880               figRotate[i][j-1] = fig[t - 1 - i][t - 1 - j];
881             }else{
882               figRotate[i][j] = fig[t - 1 - i][t - 1 - j];
883             }
884             break;
885           case 3:
886             figRotate[i][j] = fig[t - 1 - j][i];
887             break;
888         }
889       } else if ((figName == "Z") || (figName == "Zinv")) {
890         switch (rotation) {
891           case 0:
892               figRotate[i][j] = fig[i][j];
893             break;
894           case 1:
895               figRotate[i][j] = fig[j][t - 1 - i];
896             break;
897         }
898       }else if (figName == "Line") {
899         switch (rotation) {
900           case 0:
901               figRotate[i][j] = fig[i][j];
902             break;
903           case 1:
904               if(i>2){
905                 figRotate[i-3][j] = fig[j][t - 1 - i];
906               }else{
907                 figRotate[i][j] = fig[j][t - 1 - i];
908               }
909             break;
910           default:
911             figRotate[i][j] = fig[i][j];
912             break;
913         }
```

```
      } else {
        figRotate[i][j] = fig[i][j];
      }
    }
  }
}

void clear_figRotate() {
  for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
      figRotate[i][j] = 0;
    }
  }
}

bool check_printFig(int x, int y, int t) {
  for (int f = 0; f < t; f++) {
    for (int c = 0; c < t; c++) {
      if (figRotate[f][c] == 1) {
        if (canvas[x + f][y + c] == 1) {
          return false;
        }
      }
    }
  }
  return true;
}

void mem_to_canvas() {
  for (int f = 0; f < 16; f++) {
    for (int c = 0; c < 8; c++) {
      if (mem[f][c] == 1) {
        canvas[f][c] = 1;
      }
    }
  }
}

bool printFig(int x, int y, int t) {
  if (num_fig_ale == 5 && posCol >= 5 && rotation == 1) {// Line
      y = y - (y-4);
  }
  for (int f = 0; f < t; f++) {
    for (int c = 0; c < t; c++) {
      if (figRotate[f][c] == 1) {
        mem[x + f][y + c] = 1;
      }
    }
  }
}

void rotacion(string figName) {
  if (figName == "L" or figName == "Linv" or figName == "T") {
    if (rotation < 3) {
      rotation++;
    } else {
      rotation = 0;
    }

  } else if (figName == "Z" or figName == "Zinv" or figName == "Line") {
    if (rotation == 0) {
      rotation = 1;
    } else {
      rotation = 0;
    }
  } else {
    rotation = 0;
  }
}
```

```
985
986 void Pantalla_Inicio() {
987
988   lcd.setCursor(2, 0); //Ubicar cursor en la columna 3 fila 0
989   lcd.print("Bienvenido"); //Imprimir mensaje en la LCD
990   lcd.setCursor(2, 1); //Ubicar cursor en la columna 3 fila 0
991   lcd.print("Pulse P1"); //Imprimir mensaje en la LCD
992 }
993
994 void Pantalla_Nivel() {
995   delay(1000);
996
997   lcd.setCursor(2, 0); //Ubicar cursor en la columna 3 fila 0
998   lcd.print("Nivel: "); //Imprimir mensaje en la LCD
999   lcd.setCursor(4, 1);
1000  lcd.print("Jugar(P2)"); //Imprimir mensaje en la LCD
1001 }
1002 void Pantalla_Juego(int nivel) {
1003   //delay(1000);
1004   lcd.clear();
1005   lcd.setCursor(0, 0); //Ubicar cursor en la columna 3 fila 0
1006   lcd.print("Nivel: "); //Imprimir mensaje en la LCD
1007   lcd.print(nivel); //Imprimir mensaje en la LCD
1008   lcd.setCursor(10, 0);
1009   lcd.print("Maximo:"); //Imprimir mensaje en la LCD
1010   lcd.setCursor(0, 1);
1011   lcd.print("Score: "); //Imprimir mensaje en la LCD
1012   lcd.print(Score);
1013   lcd.setCursor(11, 1);
1014   lcd.print(puntaje_maximo);
1015 }
1016
1017 void Pantalla_Pausa() {
1018   lcd.clear();
1019   lcd.setCursor(2, 0); //Ubicar cursor en la columna 3 fila 0
1020   lcd.print("Juego Pausado"); //Imprimir mensaje en la LCD
1021   lcd.setCursor(2, 1);
1022   lcd.print("Reanudar (P1)"); //Imprimir mensaje en la LCD
1023 }
1024
1025
1026 //*************** PULSADORES ************************
1027
1028 void Pulsador1_() {
1029
1030   //delay(tiempo_P1);
1031   if (millis() - rebote_P1 > Tiempo_Rebote_sup && bandera_P1) {
1032     bandera_P1 = 0;
1033     attachInterrupt(digitalPinToInterrupt(pinP1), Int_P1, FALLING); //Habilita nuevamente int.
       generada por el GPIO16
1034   }
1035 }
1036 void Pulsador2_() {
1037
1038   //delay(tiempo_P2);
1039   if (millis() - rebote_P2 > Tiempo_Rebote_sup && bandera_P2) {
1040     bandera_P2 = 0;
1041     attachInterrupt(digitalPinToInterrupt(pinP2), Int_P2, FALLING); //Habilita nuevamente int.
       generada por el GPIO16
1042   }
1043 }
1044 void Pulsador3_() {
1045
1046   //delay(tiempo_P3);
1047   if (millis() - rebote_P3 > Tiempo_Rebote_sup && bandera_P3) {
1048     bandera_P3 = 0;
1049     attachInterrupt(digitalPinToInterrupt(pinP3), Int_P3, FALLING); //Habilita nuevamente int.
       generada por el GPIO16
1050   }
1051 }
1052 void Pulsador4_() {
```

```
1053
1054    if (millis() - rebote_P4 > Tiempo_Rebote_sup && bandera_P4) {
1055      bandera_P4 = 0;
1056      attachInterrupt(digitalPinToInterrupt(pinP4), Int_P4, FALLING); //Habilita nuevamente int.
          generada por el GPIO16
1057    }
1058  }
1059  void Pulsador5_() {
1060
1061    //delay(tiempo_P5);
1062    if (millis() - rebote_P5 > Tiempo_Rebote_sup && bandera_P5) {
1063      bandera_P5 = 0;
1064      attachInterrupt(digitalPinToInterrupt(pinP5), Int_P5, FALLING); //Habilita nuevamente int.
          generada por el GPIO16
1065    }
1066  }
1067  void Rebote_P1() {
1068    if (tiempo_P1 == Tiempo_Rebote) {
1069      tiempo_P1 = Tiempo_Rebote_inf;
1070    } else {
1071      tiempo_P1 = Tiempo_Rebote;
1072    }
1073    bandera_P1 = 1; //Activa bandera para indicar ingreso a rutina de interrupci n
1074    rebote_P1 = millis(); //Lee el valor actual de la funci n millis()
1075    detachInterrupt(digitalPinToInterrupt(pinP1)); //Deshabilita int. del GPIO16
1076  }
1077
1078  void Rebote_P2() {
1079    if (tiempo_P2 == Tiempo_Rebote) {
1080      tiempo_P2 = Tiempo_Rebote_inf;
1081    } else {
1082      tiempo_P2 = Tiempo_Rebote;
1083    }
1084    bandera_P2 = 1; //Activa bandera para indicar ingreso a rutina de interrupci n
1085    rebote_P2 = millis(); //Lee el valor actual de la funci n millis()
1086    detachInterrupt(digitalPinToInterrupt(pinP2)); //Deshabilita int. del GPIO16
1087  }
1088  void Rebote_P3() {
1089    if (tiempo_P3 == Tiempo_Rebote) {
1090      tiempo_P3 = Tiempo_Rebote_inf;
1091    } else {
1092      tiempo_P3 = Tiempo_Rebote;
1093    }
1094    bandera_P3 = 1; //Activa bandera para indicar ingreso a rutina de interrupci n
1095    rebote_P3 = millis(); //Lee el valor actual de la funci n millis()
1096    detachInterrupt(digitalPinToInterrupt(pinP3)); //Deshabilita int. del GPIO16
1097  }
1098  void Rebote_P4() {
1099    if (tiempo_P4 == Tiempo_Rebote) {
1100      tiempo_P4 = Tiempo_Rebote_inf;
1101    } else {
1102      tiempo_P4 = Tiempo_Rebote;
1103    }
1104    bandera_P4 = 1; //Activa bandera para indicar ingreso a rutina de interrupci n
1105    rebote_P4 = millis(); //Lee el valor actual de la funci n millis()
1106    detachInterrupt(digitalPinToInterrupt(pinP4)); //Deshabilita int. del GPIO16
1107  }
1108  void Rebote_P5() {
1109    if (tiempo_P5 == Tiempo_Rebote) {
1110      tiempo_P5 = Tiempo_Rebote_inf;
1111    } else {
1112      tiempo_P5 = Tiempo_Rebote;
1113    }
1114    bandera_P5 = 1; //Activa bandera para indicar ingreso a rutina de interrupci n
1115    rebote_P5 = millis(); //Lee el valor actual de la funci n millis()
1116    detachInterrupt(digitalPinToInterrupt(pinP5)); //Deshabilita int. del GPIO16
1117  }
```

Listing 1: Ejemplo Proyecto 1