UNIVERSIDAD DEL VALLE DE GUATEMALA

Electrónica digital 2

Sección 30

Ing. Pablo Mazariegos

Proyecto # 3

Videojuego Portable

Carmen Natalia de León Bercián - 18193

Katharine Senn Salazar - 18012

Link video

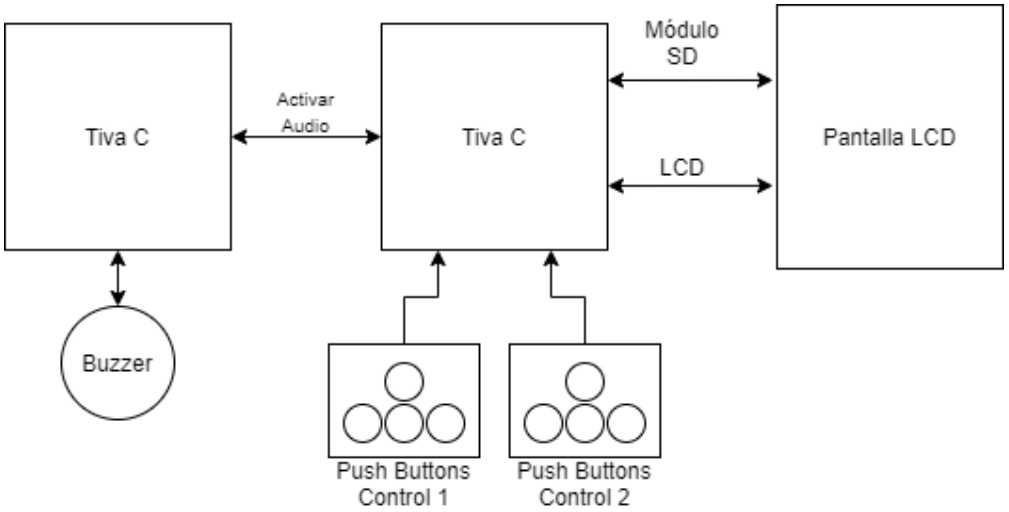
https://youtu.be/xwKO3aYqiLA

Github

https://github.com/nataliadlb/LABS_REPOSITORIO.git

https://github.com/sen18012/Labs_Digital_2/tree/main/Proyecto_3

Circuitos



Datos

Definición de pines para conexiones

LCD_RST PD_0 **//LCD**
LCD_CS PD_1
LCD_RS PD_2
LCD_WR PD_3
LCD_RD PE_1

PUSH_LEFT_J1 PA_7 **//CONTROLES J1**
PUSH_RIGTH_J1 PE_3
PUSH_UP_J1 PA_6
PUSH_DOWN_J1 PE_2

PUSH_LEFT_J2 PD_7 **//CONTROLES J2**
PUSH_RIGTH_J2 PC_7
PUSH_UP_J2 PD_6
PUSH_DOWN_J2 PC_6

BUZZER PC_5 **//ACTIVAR SONIDO**

Banderas Inicio Juego

```
flag_jugar = LOW; //BANDERA QUE INDICA QUE YA TERMINARON  DE ESCOGER
flag_boton_jugar = LOW; //BANDERA QUE INDICA QUE YA  TERMINARON DE ESCOGER
ganar_N1 = LOW; //INDICA QUE ALGUNO DE LOS JUGADORES  GANÓ EL NIVEL 1
ganar_N2 = LOW; //INDICA QUE ALGUNO DE LOS JUGADORES  GANÓ EL NIVEL 2
ganar_N3 = LOW; //INDICA QUE ALGUNO DE LOS JUGADORES  GANÓ EL NIVEL 3
Listo_per_J1 = LOW; //INDICA QUE EL JUGADOR 1 YA ELIGIÓ  SU PERSONAJE
Listo_per_J2 = LOW; //INDICA QUE EL JUGADOR 2 YA ELIGIÓ  SU PERSONAJE
JUEGO_EN_PROGRESO = LOW; //INDICA QUE EL JUGADOR 1  YA ELIGIÓ SU PERSONAJE
```

Banderas para Movimiento de Jugadores

```
b_LEFT_J1 = LOW;  //INDICAN SI ALGÚN BOTÓN DE LOS CONTROLES FUE PRESIONADO
b_LEFT_J2 = LOW;
b_RIGHT_J1 = LOW;
b_RIGHT_J2 = LOW;
b_UP_J1 = LOW;
b_UP_J2 = LOW;
b_DOWN_J1 = LOW;
b_DOWN_J2 = LOW;
```

Banderas para Estrellas

```
STAR_1_J1 = LOW; //INDICAN SI EL JUGADOR YA PASÓ POR  LAS COORDENADAS DE CADA ESTRELLA
STAR_1_J2 = LOW;
STAR_2_J1 = LOW;
STAR_2_J2 = LOW;
STAR_3_J1 = LOW;
STAR_3_J2 = LOW;
STAR_4_J1 = LOW;
STAR_4_J2 = LOW;
STAR_5_J1 = LOW;
STAR_5_J2 = LOW;
```

Contadores

```
cont_PUSH1 = 0;  //AL PRESIONAR PUSH1 PASA DE CAMBIO  DE PERSONAJE A INICAR EL JUEGO
cont_personajes_J1 = 0; //INDICA QUE PERSONAJE ELIGIÓ EL JUGADOR 1
cont_personajes_J2 = 0; //INDICA QUE PERSONAJE ELIGIÓ EL JUGADOR 2
```

Información

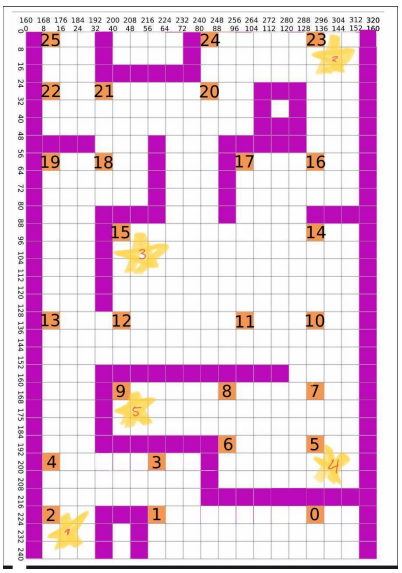
nivel = 0; //INDICA EL NIVEL ACTUAL
num_personaje_J1 = 0; //GUARDA EL NÚMERO CORRESPONDIENTE AL PERSONAJE ELEGIDO POR J1
num_personaje_J2 = 0; //GUARDA EL NÚMERO CORRESPONDIENTE AL PERSONAJE ELEGIDO POR J2
posicion_J1 = 0; //INDICA LA POSICIÓN ACTUAL DE J1
posicion_J2 = 0; //INDICA LA POSICIÓN ACTUAL DE J2
STARS_J1 = 0; //INDICA CUÁNTAS ESTRELLAS LLEVA RECOLECTADAS J1 EN TOTAL
STARS_J2 = 0; //INDICA CUÁNTAS ESTRELLAS LLEVA RECOLECTADAS J2 EN TOTAL
STARS_N1_J1 = 0; //INDICA LA CANTIDAD DE ESTRELLAS RECOLECTADAS POR J1 EN EL N1
STARS_N1_J2 = 0; //INDICA LA CANTIDAD DE ESTRELLAS RECOLECTADAS POR J2 EN EL N1
STARS_N2_J1 = 0; //INDICA LA CANTIDAD DE ESTRELLAS RECOLECTADAS POR J1 EN EL N2
STARS_N2_J2 = 0; //INDICA LA CANTIDAD DE ESTRELLAS RECOLECTADAS POR J2 EN EL N2
STARS_N3_J1 = 0; //INDICA LA CANTIDAD DE ESTRELLAS RECOLECTADAS POR J1 EN EL N3
STARS_N3_J2 = 0; //INDICA LA CANTIDAD DE ESTRELLAS RECOLECTADAS POR J2 EN EL N3
Juegos_ganados_J1 = 0; //INDICA LA CANTIDAD DE NIVELES GANADOS POR J1
Juegos_ganados_J2 = 0; //INDICA LA CANTIDAD DE NIVELES GANADOS POR J2

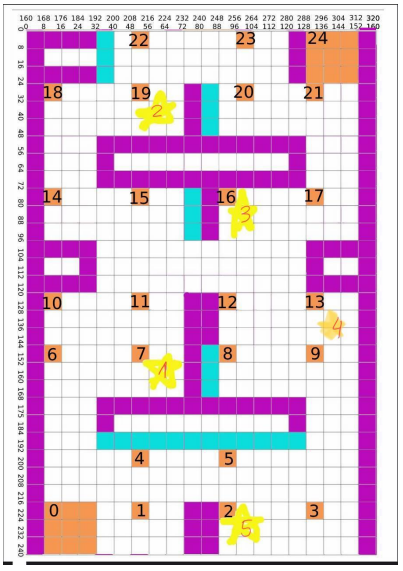
Canción Utilizada

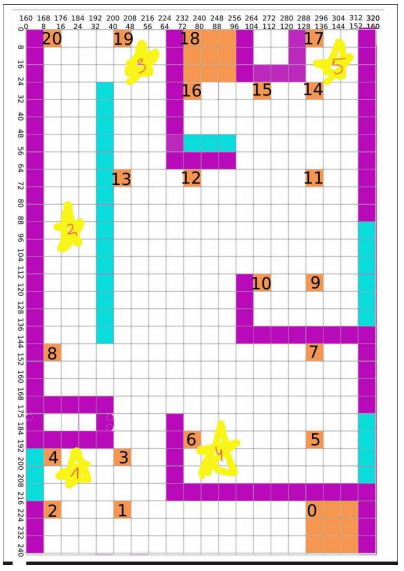
El arreglo " melody[]" contiene las notas y los tiempos de cada nota de la canción Green Hills Zone's Theme. Original del juego Sonic the Hedgehog.

Gráficos

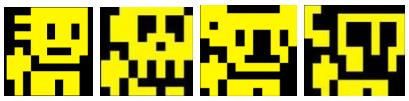
Diseños propios de cada nivel





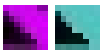


Personajes Disponibles



Estrellas

Ladrillos



Logo (Pantalla Inicio)

Explicaciones

La idea del juego: El videojuego realizado se basó en el juego para celular 'Tomb of the mask', pero se le hizo modificaciones para que este se pudiera jugar con dos jugadores simultáneamente. Lo que se hizo fue separar la pantalla en dos, para que en cada lado estuviera el mismo mapa y ambos jugadores compitan al mismo tiempo.

Instrucciones: Son 3 niveles y en cada uno hay 5 estrellas, por lo que, el primer jugador en obtener las 5 estrellas y llegar a la meta gana el nivel. Luego de los 3 niveles el jugador con más estrellas gana.

Almacenamiento SD: La implementación de la SD fue para los personajes. Debido a que podían escoger entre 4 personajes, esos los guardamos en la SD, por lo que únicamente el personaje que escogía cada jugador era el que se llamaba. De esta manera evitamos cargar la RAM con personajes que no se iba a utilizar

Controles: Los controles consistían en push buttons. Cada jugador tenía 4 push para las 4 direcciones. También se utilizaron los push que ya tiene la Tiva C incorporados en la pantalla de inicio. Al presionar 1 vez el push1 se escoge el personaje del jugador 1, al presionar el push 2 se escoge el personaje del jugador 2 y al presionar nuevamente el push1 se inicia el juego.

Sonido: Para implementar el sonido se utilizó otra Tiva C, en la que estaba la canción y funcionaba con un buzzer. Para activar la música se activaba un pin desde la Tiva en donde se ejecutaba el juego al iniciar.

Código

Todos los prototipos de funciones

```
//**************************************

// Functions Prototypes

//**************************************

//--- FUNCIONES LCD ---//

void LCD_Init(void);

void LCD_CMD(uint8_t cmd);

void LCD_DATA(uint8_t data);

void SetWindows(unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2);

void LCD_Clear(unsigned int c);

void H_line(unsigned int x, unsigned int y, unsigned int l, unsigned int c);

void V_line(unsigned int x, unsigned int y, unsigned int l, unsigned int c);

void Rect(unsigned int x, unsigned int y, unsigned int w, unsigned int h, unsigned int c);

void FillRect(unsigned int x, unsigned int y, unsigned int w, unsigned int h, unsigned int c);

void LCD_Print(String text, int x, int y, int fontSize, int color, int background);



void LCD_Bitmap(unsigned int x, unsigned int y, unsigned int width, unsigned int height, unsigned char bitmap[]);

void LCD_Sprite(int x, int y, int width, int height, unsigned char bitmap[],int columns, int index, char flip, char offset);

//---- FUNCIONES PANTALLA INICIO ----//

void Static_Pantalla_Inicio(void); //lo que aparece en el menu y es fijo

void Mov_Pantalla_inicio(void); // lo que se mueve en la pagina de inicio

void Listo_personajes(void); //Mostrar texto de listo cuando se elige personaje


//---- FUNCIONES PANTALLAS ----//

void Nivel_pantalla(int Num_Nivel); //para mostrar la pantalla del nivel que toca

void Marcador_pantalla(void); //pantalla que muestra el marcador y cuantas estrella tiene cada jugador

void Pantalla_instrucciones(void); //pantala que muestra las instrucciones del juego


//FUNCIONES APARECER EN LOS MAPAS //

void Posicion_inicial_munecos(int nivel_pos_i); //funcion para poner o munecos

void Posicion_meta(int nivel_pos_i); //funcion para poner la meta en cada mapa

void Posicion_estrellas(int nivel_pos_i); //funcion par aponer las estrellas en cada nivel

void Mostrar_stars_en_mapa(int nivel_pos_i); //funcion para colocar las estrellas en cada mapa


//---- FUNCION DE CADA MAPA (BLOQUES, ESTRELLAS, ENEMIGOS) ----//

void Mapa_nivel(int nivel_mapa); //funcion para crear los mapas e imprimirlos


//---- FUNCIONES PARA ESCOGER PERSONAJE EN TODO EL JEUGO ----//
```

```
void Personajes_usar(int num_per_J1, int num_per_J2); //funcion para asignar de la SD el personaje que se escogio

//--- FUNCIONES PARA SD ---//

void open_SD_bitmap(unsigned char Bitmap_SD[], unsigned long Size_bitmap, char* filename); //funcion obtiene archivo de la SD

int ACII_to_HEX(char *puntero);// funcion que convierte los caracteres del arreglo a HEX


//--- FUNCIONES AL APACHAR CADA BOTON EN CADA JUGADOR ---//

void switch_posicion_LEFT_J1(int num_nivel);

void switch_posicion_RIGHT_J1(int num_nivel);

void switch_posicion_UP_J1(int num_nivel);

void switch_posicion_DOWN_J1(int num_nivel);


void switch_posicion_LEFT_J2(int num_nivel);

void switch_posicion_RIGHT_J2(int num_nivel);

void switch_posicion_UP_J2(int num_nivel);

void switch_posicion_DOWN_J2(int num_nivel);


//--- FUNCIONES BORRAR BANDERA ESTRELLAS ---//

void bajar_banderas_stars(void); //funcion para bajar banderas al iniciar cada nivel
```

Loop principal

En este loop se llama principalmente a las funciones, no se realizan las instrucciones como tal. Primero se muestra la pantalla de inicio y se permite escoger personaje, una vez hecho eso se va a un switch case dependiendo del nivel. En cada nivel se muestra una pantalla con el nivel y una pantalla con el marcador y cantidad de estrellas de cada jugador. Luego se muestra el mapa y por último en cada nivel se entra a un while en el que se revisa si se presiona algún botón y si es así revisa en que posición está y si puede moverse lo hace y si no, se queda estático.

```
void loop() {

      if (flag_boton_jugar == LOW && JUEGO_EN_PROGRESO == LOW){
   Static_Pantalla_Inicio();
   Mov_Pantalla_inicio();
   Listo_personajes();
   }
    else if (flag_boton_jugar == HIGH && JUEGO_EN_PROGRESO == LOW) {
   String text1 = "JUGAR";
   LCD_Print(text1, 111, 200, 2, 0x000, 0x07FF);
         Personajes_usar(num_personaje_J1, num_personaje_J2);
   JUEGO_EN_PROGRESO = HIGH;
   delay(500);
   }
 //else {}

 while (JUEGO_EN_PROGRESO == HIGH){
   //digitalWrite(BUZZER, HIGH);
   switch (nivel){
```

```
    case 1:
      digitalWrite(BUZZER, LOW);
        Nivel_pantalla(1); //Titulo del nivel al que pasan
      delay(500);
      digitalWrite(BUZZER, HIGH);
      Mapa_nivel(1); //bloques del nivel
      bajar_banderas_stars();
      while (ganar_N1 != HIGH){
        Mostrar_stars_en_mapa(1); //muestra valor de las etrellas agarradas en el nivel

      //-- MOVIMIENTOS JUGADOR 1 --//
      if (b_LEFT_J1 == HIGH){
       switch_posicion_LEFT_J1(1);
       }
      if (b_RIGHT_J1 == HIGH){
       //Rect(8,8,24,24,0xFFFF);
       switch_posicion_RIGHT_J1(1);
       }
      if (b_UP_J1 == HIGH){
       switch_posicion_UP_J1(1);
       }

     if (b_DOWN_J1 == HIGH){
       switch_posicion_DOWN_J1(1);
       }

      //-- MOVIMIENTOS JUGADOR 2 --//
     if (b_LEFT_J2 == HIGH){
       switch_posicion_LEFT_J2(1);
       }

     if (b_RIGHT_J2 == HIGH){
       switch_posicion_RIGHT_J2(1);
       }

     if (b_UP_J2 == HIGH){
       switch_posicion_UP_J2(1);
       }
     if (b_DOWN_J2 == HIGH){
       switch_posicion_DOWN_J2(1);
       }
     }
    break;
    case 2:
        Marcador_pantalla(); //muestra juegos ganados y cantidad de estrellas
      delay(2500);
      digitalWrite(BUZZER, LOW);
        Nivel_pantalla(2); //Titulo del nivel al que pasan
      delay(500);
      digitalWrite(BUZZER, HIGH);
      Mapa_nivel(2); //bloques del nivel
      bajar_banderas_stars();
      while(ganar_N2 != HIGH){
```

```
    Mostrar_stars_en_mapa(2);


    if (b_LEFT_J1 == HIGH){
          switch_posicion_LEFT_J1(2); //muestra valor de las etrellas agarradas en el nivel
    }

   if (b_RIGHT_J1 == HIGH){
    //Rect(8,8,24,24,0xFFFF);
    switch_posicion_RIGHT_J1(2);
    }

   if (b_UP_J1 == HIGH){
    switch_posicion_UP_J1(2);
    }

  if (b_DOWN_J1 == HIGH){
    switch_posicion_DOWN_J1(2);
    }
  //-- MOVIMIENTOS JUGADOR 2 --//
  if (b_LEFT_J2 == HIGH){
    switch_posicion_LEFT_J2(2);
    }

   if (b_RIGHT_J2 == HIGH){
    switch_posicion_RIGHT_J2(2);
    }
   if (b_UP_J2 == HIGH){
    switch_posicion_UP_J2(2);
    }
   if (b_DOWN_J2 == HIGH){
    switch_posicion_DOWN_J2(2);
    }
   }

  break;
  case 3:
        Marcador_pantalla(); //muestra juegos ganados y cantidad de estrellas
    delay(2500);
    digitalWrite(BUZZER, LOW);
      Nivel_pantalla(3); //Titulo del nivel al que pasan
    delay(500);
    digitalWrite(BUZZER, HIGH);
    Mapa_nivel(3); //bloques del nivel
    bajar_banderas_stars();

    while(ganar_N3 != HIGH){
      Mostrar_stars_en_mapa(3); //muestra valor de las etrellas agarradas en el nivel

     //-- MOVIMIENTOS JUGADOR 1 --//
    if (b_LEFT_J1 == HIGH){
      switch_posicion_LEFT_J1(3);
      }
    if (b_RIGHT_J1 == HIGH){
```

```
  //Rect(8,8,24,24,0xFFFF);
  switch_posicion_RIGHT_J1(3);
  }
 if (b_UP_J1 == HIGH){
  switch_posicion_UP_J1(3);
  }
if (b_DOWN_J1 == HIGH){
  switch_posicion_DOWN_J1(3);
  }
 //-- MOVIMIENTOS JUGADOR 2 --//
if (b_LEFT_J2 == HIGH){
  switch_posicion_LEFT_J2(3);
  }
 if (b_RIGHT_J2 == HIGH){
  switch_posicion_RIGHT_J2(3);
  }
 if (b_UP_J2 == HIGH){
  switch_posicion_UP_J2(3);
  }
 if (b_DOWN_J2 == HIGH){
   switch_posicion_DOWN_J2(3);
   }
   }
 break;

 case 4:
 while (JUEGO_EN_PROGRESO == HIGH){
   Mapa_nivel(4);
  }

  //delay(1000);

 break;
 }
 }
}
```

Función de la pantalla de inicio

En esta pantalla se muestran los personajes y una variable va cambiando en base al personaje que se va mostrando, una vez se escoge el personaje el valor de la variable se queda estático.

```
void Mov_Pantalla_inicio(void){ //Todo lo que va cambiando de la pantalla de inicio

                                LCD_Sprite(29, 111,24,24,next_amarillo_24,1,0,1,0);   //FLECHAS PARA CAMBIAR DE PERSONAJE
                                LCD_Bitmap(259,  111,  24,  24, next_amarillo_24);

  switch(cont_personajes_J1){
```

```
      case 0:                                 //EN BASE A LA CANTIDAD DE VECES QUE PRESIONA EL BOTON SE MUESTRA EL PERSONAJE

            //open_SD_bitmap(personajes_inicio_50, 10001, "Mun_50.txt");//POR DEFAULT PERONAJE QUE SE OBTIENE DE LA SD

          LCD_Bitmap(66, 98, 50, 50, Muneco_50); //SE MUESTRA EL PERSONAJE

       Rect(28,110,25,25,0x0000);

                          num_personaje_J1 = 0; //VARIABLE QUE INDICA CUAL PERSONAJE SE ESCOGIO

       break;

     case 1:

            //open_SD_bitmap(personajes_inicio_50, 10001, "Cal_50.txt");

       LCD_Bitmap(66, 98, 50, 50, Calavera_50);

       Rect(28,110,25,25,0x0000);

       num_personaje_J1 = 1;

       break;

     case 2:

            //open_SD_bitmap(personajes_inicio_50, 10001, "Koala_50.txt");

       LCD_Bitmap(66, 98, 50, 50, Koala_50);

       Rect(28,110,25,25,0x0000);

       num_personaje_J1 = 2;

       break;

     case 3:

            //open_SD_bitmap(personajes_inicio_50, 10001, "Mono_50.txt");

       LCD_Bitmap(66, 98, 50, 50, Mono_50);

       Rect(28,110,25,25,0x0000);

       num_personaje_J1 = 3;

       break;

     }


   // PERSONAJE 2

   switch(cont_personajes_J2){

     case 0:

            //open_SD_bitmap(personajes_inicio_50, 10001, "Mun_50.txt");

       LCD_Bitmap(196, 98, 50, 50, Muneco_50);

       Rect(258,110,25,25,0x0000);

       num_personaje_J2 = 0;

       break;

     case 1:

            //open_SD_bitmap(personajes_inicio_50, 10001, "Cal_50.txt");

       LCD_Bitmap(196, 98, 50, 50, Calavera_50);

       Rect(258,110,25,25,0x0000);

       num_personaje_J2 = 1;

       break;

     case 2:

            //open_SD_bitmap(personajes_inicio_50, 10001, "Koala_50.txt");

       LCD_Bitmap(196, 98, 50, 50, Koala_50);

       Rect(258,110,25,25,0x0000);

       num_personaje_J2 = 2;

       break;

     case 3:

            //open_SD_bitmap(personajes_inicio_50, 10001, "Mono_50.txt");
```

```
    LCD_Bitmap(196, 98, 50, 50, Mono_50);
    Rect(258,110,25,25,0x0000);
    num_personaje_J2 = 3;
      break;
    }
}
```

Función en donde se manda a llamar los personajes desde la SD

En esta primera función es en donde se manda a llamar las funciones de la SD se guarda en una variable global, por lo que no importa que personaje escoja siempre se guarda en la misma variable para ahorrar memoria

```
void Personajes_usar(int num_per_J1, int num_per_J2){ //EN BASE A LO QUE ESCOGIERON EN EL INICIO SE OBTIENE Y GUARDA DE SD EL PERSONAJE A USAR EN EL MAPA

  //J1//

  switch (num_per_J1){

    case 0:

        open_SD_bitmap(J1_Abajo_Derecha, 2305, "Mun_24.txt");

      break;

    case 1:

        open_SD_bitmap(J1_Abajo_Derecha, 2305, "Cal_24.txt");

      break;

    case 2:

        open_SD_bitmap(J1_Abajo_Derecha, 2305, "Koala_24.txt");

      break;

   case 3:

        open_SD_bitmap(J1_Abajo_Derecha, 2305, "Mono_24.txt");

      break;

    }

 //J2//

  switch (num_per_J2){

    case 0:

        open_SD_bitmap(J2_Abajo_Derecha, 2305, "Mun_24.txt");

      break;

    case 1:

        open_SD_bitmap(J2_Abajo_Derecha, 2305, "Cal_24.txt");

      break;

    case 2:

        open_SD_bitmap(J2_Abajo_Derecha, 2305, "Koala_24.txt");

      break;

   case 3:

        open_SD_bitmap(J2_Abajo_Derecha, 2305, "Mono_24.txt");

      break;

    }

  }
```

Función para obtener personaje de la SD

En estas funciones se llama al arrglo guardado en un .txt en la SD y luego se convierte de ASCII a Hex.

```
void open_SD_bitmap(unsigned char Bitmap_SD[], unsigned long Size_bitmap, char* filename) {

  File myFile = SD.open(filename);     // ABRIR EL ARCHIVO

  unsigned long i = 0;

   char Bitmap_SD_HEX[] = {0, 0};          //SE HACE ARREGLO DE DOS NUM, PARA CADA UNA DE LAS POSICIONES

  int Pos_1, Pos_2;                  //VARIABLES DE LAS POSICIONES

  if (myFile) {

    do {

      Bitmap_SD_HEX[0] = myFile.read(); //LEE

          Pos_1 = ACII_to_HEX(Bitmap_SD_HEX); //TRANSOFRMA

      Bitmap_SD_HEX[0] = myFile.read(); //LEE

          Pos_2 = ACII_to_HEX(Bitmap_SD_HEX); //TRANSFORMA

       Bitmap_SD[i] = (Pos_1 << 4) | (Pos_2 & 0xF); //SE CONCATENA CONCATENA

      i++;

    } while (i < (Size_bitmap + 1));

  }

  myFile.close();
}

//**************************************

int ACII_to_HEX(char *puntero) {

  int i = 0;

  for (;;) {

    char num = *puntero;

    if (num >= '0' && num <= '9') {

      i *= 16;

      i += num - '0';

    }

    else if (num >= 'a' && num <= 'f') {

      i *= 16;

      i += (num - 'a') + 10;

    }

    else break;

    puntero++;

  }

  return i;

}
```

Función para revisar en que posición se encuentra el personaje y su movimineto

Esta es únicamente el case de cuando se presiona el botón de la izquierda del jugador 1 en el nivel 1, pero todos los demás son iguales tanto para J1 como para J2.

```
void switch_posicion_LEFT_J1(int num_nivel){

  switch(num_nivel){

//#################### NIVEL 1 IZQUIERDA J1 ################################//

        case 1: //NIVEL 1 MOVERSE A LA IZQUIERDA

     switch(posicion_J1){

            case 0://POSICION INICIO NIVEL 1 JUGADOR 1

          for(int x = 128; x >56; x = x-1){


LCD_Sprite(x,216,24,24,J1_Abajo_Derecha,1,0,1,0);

            V_line( x + 24, 216, 24, 0x0000);

          }

          posicion_J1 = 1;

          b_LEFT_J1 = LOW;

       break; //break pos 0 /LEFT/N1/J1



       case 1:

         b_LEFT_J1 = LOW;

       break; //break pos 1 /LEFT/N1/J1



       case 2:

         b_LEFT_J1 = LOW;

       break; //break pos 2 /LEFT/N1/J1



       case 3:

         for(int x = 56; x >8; x = x-1){


LCD_Sprite(x,192,24,24,J1_Abajo_Derecha,1,0,1,0);

            V_line( x + 24, 192, 24, 0x0000);

            FillRect(56,208,24,16,0x0000);

          }

         posicion_J1 = 4;

         b_LEFT_J1 = LOW;

       break; //break pos 3 /LEFT/N1/J1


       case 4:

         b_LEFT_J1 = LOW;

       break; //break pos 4 /LEFT/N1/J1


       case 5:

          for(int x = 128; x >88; x = x-1){


LCD_Sprite(x,184,24,24,J1_Abajo_Derecha,1,0,1,0);

            V_line( x + 24, 184, 24, 0x0000);

            FillRect(128,176,24,16,0x0000);

          }

         posicion_J1 = 6;

         b_LEFT_J1 = LOW;

       break; //break pos 5 /LEFT/N1/J1


       case 6:

         b_LEFT_J1 = LOW;

       break; //break pos 6 /LEFT/N1/J1
```

```
      case 7:
       for(int x = 128; x >40; x = x-1){
LCD_Sprite(x,160,24,24,J1_Abajo_Derecha,1,0,1,
0);
        V_line( x + 24, 160, 24, 0x0000);
       }
       if (STAR_5_J1 == LOW){
        STARS_N1_J1++;
        STARS_J1++;
        STAR_5_J1 = HIGH;
       }
      posicion_J1 = 9;
      b_LEFT_J1 = LOW;
     break; //break pos 7 /LEFT/N1/J1

     case 8:
      for(int x = 88; x >40; x = x-1){
LCD_Sprite(x,160,24,24,J1_Abajo_Derecha,1,0,1,
0);
        V_line( x + 24, 160, 24, 0x0000);
        FillRect(88,176,24,16,0x0000);
       }
       if (STAR_5_J1 == LOW){
        STARS_N1_J1++;
        STARS_J1++;
        STAR_5_J1 = HIGH;
       }
      posicion_J1 = 9;
      b_LEFT_J1 = LOW;
     break; //break pos 8 /LEFT/N1/J1

     case 9:
      b_LEFT_J1 = LOW;
     break; //break pos 9 /LEFT/N1/J1

     case 10:
      for(int x = 128; x >8; x = x-1){
LCD_Sprite(x,128,24,24,J1_Abajo_Derecha,1,0,1,
0);
        V_line( x + 24, 128, 24, 0x0000);
       }
      posicion_J1 = 13;
      b_LEFT_J1 = LOW;
     break; //break pos 10 /LEFT/N1/J1

     case 11:
      for(int x = 96; x >8; x = x-1){
LCD_Sprite(x,128,24,24,J1_Abajo_Derecha,1,0,1,
0);
        V_line( x + 24, 128, 24, 0x0000);
        FillRect(96,120,24,16,0x0000);
       }
      posicion_J1 = 13;
      b_LEFT_J1 = LOW;
     break; //break pos 11 /LEFT/N1/J1
```

```
      case 12:
       for(int x = 40; x >8; x = x-1){
LCD_Sprite(x,128,24,24,J1_Abajo_Derecha,1,0,1,
0);
        V_line( x + 24, 128, 24, 0x0000);
        FillRect(40,120,24,16,0x0000);
       }
       posicion_J1 = 13;
       b_LEFT_J1 = LOW;
      break; //break pos 12 /LEFT/N1/J1

      case 13:
       b_LEFT_J1 = LOW;
      break; //break pos 13 /LEFT/N1/J1

      case 14:
       for(int x = 128; x >40; x = x-1){
LCD_Sprite(x,88,24,24,J1_Abajo_Derecha,1,0,1,0)
;
        V_line( x + 24, 88, 24, 0x0000);
        FillRect(128,104,24,16,0x0000);
       }
       if (STAR_3_J1 == LOW){
        STARS_N1_J1++;
        STARS_J1++;
        STAR_3_J1 = HIGH;
       }
       posicion_J1 = 15;
       b_LEFT_J1 = LOW;
      break; //break pos 14 /LEFT/N1/J1

      case 15:
       b_LEFT_J1 = LOW;
      break; //break pos 15 /LEFT/N1/J1

      case 16:
       for(int x = 128; x >96; x = x-1){
LCD_Sprite(x,56,24,24,J1_Abajo_Derecha,1,0,1,0)
;
        V_line( x + 24, 56, 24, 0x0000);
        FillRect(128,48,24,16,0x0000);
       }
       posicion_J1 = 17;
       b_LEFT_J1 = LOW;
      break; //break pos 16 /LEFT/N1/J1

      case 17:
       b_LEFT_J1 = LOW;
      break; //break pos 17 /LEFT/N1/J1

      case 18:
       for(int x = 32; x >8; x = x-1){
LCD_Sprite(x,56,24,24,J1_Abajo_Derecha,1,0,1,0)
;
        V_line( x + 24, 56, 24, 0x0000);
        FillRect(32,48,24,16,0x0000);
```

```
        }
       posicion_J1 = 19;
       b_LEFT_J1 = LOW;
      break; //break pos 18 /LEFT/N1/J1

      case 19:
       b_LEFT_J1 = LOW;
      break; //break pos 19 /LEFT/N1/J1

      case 20:
       for(int x = 80; x >8; x = x-1){
LCD_Sprite(x,24,24,24,J1_Abajo_Derecha,1,0,1,0)
;
         V_line( x + 24, 24, 24, 0x0000);
         FillRect(80,16,24,16,0x0000);
        }
       posicion_J1 = 22;
       b_LEFT_J1 = LOW;
      break; //break pos 20 /LEFT/N1/J1

      case 21:
       for(int x = 32; x >8; x = x-1){
LCD_Sprite(x,24,24,24,J1_Abajo_Derecha,1,0,1,0)
;
         V_line( x + 24, 24, 24, 0x0000);
         FillRect(32,40,24,16,0x0000);
        }
       posicion_J1 = 22;
       b_LEFT_J1 = LOW;
      break; //break pos 21 /LEFT/N1/J1

      case 22:
       b_LEFT_J1 = LOW;
      break; //break pos 22 /LEFT/N1/J1

      case 23:
       for(int x = 128; x >80; x = x-1){
LCD_Sprite(x,0,24,24,J1_Abajo_Derecha,1,0,1,0);
         V_line( x + 24, 0, 24, 0x0000);
         FillRect(128,16,24,16,0x0000);
        }
       posicion_J1 = 24;
       b_LEFT_J1 = LOW;
      break; //break pos 23 /LEFT/N1/J1

      case 24:
       b_LEFT_J1 = LOW;
      break; //break pos 24 /LEFT/N1/J1

      case 25:
       b_LEFT_J1 = LOW;
      break; //break pos 25 /LEFT/N1/J1
      }
    break; //break del case 1 (nivel 1 - LEFT - J1)
```

Función para crear el mapa de un nivel

Esta es únicamente el mapa del nivel 1, pero los demás mapas se generaron de la misma manera.

```
void Mapa_nivel(int nivel_mapa){

  FillRect(0, 0, 320, 240, 0x0000);

  switch (nivel_mapa){


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

//          ----------------------          MAPA          1
-----------------------------//

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

    case 1:

          Posicion_inicial_munecos(1); //poner a los
munecos en su posicion inicial

      Posicion_meta(1);

      Posicion_estrellas(1);



       for(int y = 0; y <240; y++){ // 1 LINEAS DE
LAS ORILLAS

                          LCD_Bitmap(312,   y,   8,   8,
Bloque_8_morado);

                          LCD_Bitmap(160,   y,   8,   8,
Bloque_8_morado);

                          LCD_Bitmap(152,   y,   8,   8,
Bloque_8_morado);

          LCD_Bitmap(0, y, 8, 8, Bloque_8_morado);

          y += 7;

          }


      for(int x = 240; x <312; x++){ //2

                          LCD_Bitmap(x,   208,   8,   8,
Bloque_8_morado);

                      LCD_Bitmap(x-160,   208,   8,   8,
Bloque_8_morado);

          x += 7;

          }


      for(int x = 200; x <240; x++){ // 3
```

```
                LCD_Bitmap(x, 184, 8, 8, Bloque_8_morado);

                LCD_Bitmap(x-160, 184, 8, 8, Bloque_8_morado);

        x += 7;

        }


    for(int y = 184; y <208; y++){ // 4

        LCD_Bitmap(80, y, 8, 8, Bloque_8_morado);

                LCD_Bitmap(240, y, 8, 8, Bloque_8_morado);

        y += 7;

        }


    for(int y = 216; y <240; y++){ // 5

                LCD_Bitmap(208, y, 8, 8, Bloque_8_morado);

                LCD_Bitmap(192, y, 8, 8, Bloque_8_morado);

        LCD_Bitmap(48, y, 8, 8, Bloque_8_morado);

        LCD_Bitmap(32, y, 8, 8, Bloque_8_morado);

        y += 7;

        }


                LCD_Bitmap(40, 216, 8, 8, Bloque_8_morado); //6

                LCD_Bitmap(200, 216, 8, 8, Bloque_8_morado);


    for(int y = 152; y <192; y++){ //7

        LCD_Bitmap(32, y, 8, 8, Bloque_8_morado);

                LCD_Bitmap(192, y, 8, 8, Bloque_8_morado);

        y += 7;

        }


    for(int x = 200; x <280; x++){ //8

                LCD_Bitmap(x, 152, 8, 8, Bloque_8_morado);

                LCD_Bitmap(x-160, 152, 8, 8, Bloque_8_morado);

        x += 7;

        }


    for(int y = 80; y <128; y++){ //9

        LCD_Bitmap(32, y, 8, 8, Bloque_8_morado);

                LCD_Bitmap(192, y, 8, 8, Bloque_8_morado);

        y += 7;

        }


    for(int y = 48; y <88; y++){ //10

        LCD_Bitmap(56, y, 8, 8, Bloque_8_morado);

        LCD_Bitmap(88, y, 8, 8, Bloque_8_morado);

                LCD_Bitmap(216, y, 8, 8, Bloque_8_morado);

                LCD_Bitmap(248, y, 8, 8, Bloque_8_morado);

        y += 7;

        }


    for(int y = 0; y <24; y++){ //11
```

```
        LCD_Bitmap(34, y, 8, 8, Bloque_8_morado);
        LCD_Bitmap(72, y, 8, 8, Bloque_8_morado);
                        LCD_Bitmap(194, y, 8, 8, Bloque_8_morado);
                        LCD_Bitmap(232, y, 8, 8, Bloque_8_morado);
        y += 7;
        }

    for(int y = 24; y <56; y++){ //12
                        LCD_Bitmap(104, y, 8, 8, Bloque_8_morado);
                        LCD_Bitmap(120, y, 8, 8, Bloque_8_morado);
                        LCD_Bitmap(264, y, 8, 8, Bloque_8_morado);
                        LCD_Bitmap(280, y, 8, 8, Bloque_8_morado);
        y += 7;
        }

    for(int x = 8; x <32; x++){ //13
                         LCD_Bitmap(x, 48, 8, 8, Bloque_8_morado);
                    LCD_Bitmap(x+160, 48, 8, 8, Bloque_8_morado);
        x += 7;
        }

    for(int x = 40; x <56; x++){ //14
                         LCD_Bitmap(x, 80, 8, 8, Bloque_8_morado);
                    LCD_Bitmap(x+160, 80, 8, 8, Bloque_8_morado);
        x += 7;
        }

    for(int x = 40; x <72; x++){ //15
                        LCD_Bitmap(x, 16, 8, 8, Bloque_8_morado);
                    LCD_Bitmap(x+160, 16, 8, 8, Bloque_8_morado);
        x += 7;
        }

    for(int x = 128; x <152; x++){ //16
                        LCD_Bitmap(x, 80, 8, 8, Bloque_8_morado);
                    LCD_Bitmap(x+160, 80, 8, 8, Bloque_8_morado);
        x += 7;
        }

    for(int y = 40; y <56; y++){ //17
                        LCD_Bitmap(112, y, 8, 8, Bloque_8_morado);
                        LCD_Bitmap(272, y, 8, 8, Bloque_8_morado);
        y += 7;
        }

    LCD_Bitmap(96, 48, 8, 8, Bloque_8_morado);
                    LCD_Bitmap(256, 48, 8, 8, Bloque_8_morado);
```

```
                LCD_Bitmap(112, 24, 8, 8, Bloque_8_morado);

                LCD_Bitmap(272, 24, 8, 8, Bloque_8_morado);

      //delay(1000);

    break;
```

Función para crear la pantalla del marcador

Esta es la funcion en donde se crea la pantalla del marcador y cantidad de estrellas que se van a cumulando en los tres niveles. Solo se muestra antes del nivel 2 y 3.

```
void Marcador_pantalla(void){ //MOSTRAR EL NIVEL


  FillRect(0, 0, 320, 240, 0x0000);

  for(int y = 0; y <240; y++){ //lineas decoracion

   LCD_Bitmap(0, y, 8, 8, Bloque_8_morado);

   LCD_Bitmap(312, y, 8, 8, Bloque_8_morado);

   y += 7;

  }


  for(int x = 0; x <320; x++){ //lineas decoracion

   LCD_Bitmap(x, 120, 8, 8, Bloque_8_morado);

   LCD_Bitmap(x, 0, 8, 8, Bloque_8_morado);

   LCD_Bitmap(x, 232, 8, 8, Bloque_8_morado);

   x += 7;

  }
```

```
  String text_strellas = "ESTRELLAS";

    LCD_Print(text_strellas, 85, 20, 2, 0x0000, 0xFF40);

      String text_juegosganados = "JUEGOS GANADOS";

    LCD_Print(text_juegosganados, 45, 150, 2, 0x0000, 0xFF40);


  String text_J1_estrellas = "J1";

  String text_J2_estrellas = "J2";

   LCD_Print(text_J1_estrellas, 65, 55, 2, 0x07FF, 0x0000);

  LCD_Print(text_J2_estrellas, 215, 55, 2, 0x07FF, 0x0000);

  H_line(60,75,40,0xFF40);

  H_line(210,75,40,0xFF40);
```

```
   if (STARS_J1 <=9 && STARS_J2 <= 9){

     String Str_estrellas_J1 = String(STARS_J1);

     String Str_estrellas_J2 = String(STARS_J2);

      LCD_Print(Str_estrellas_J1, 60, 85, 2, 0x07FF,
0x0000);

           LCD_Print(Str_estrellas_J2,  210,  85,  2,
0x07FF, 0x0000);



     LCD_Bitmap(80, 85, 16, 16, estrella_16);

     LCD_Bitmap(230, 85, 16, 16, estrella_16);

     }


   else if (STARS_J1 <=9 && STARS_J2 >= 10){

     String Str_estrellas_J1 = String(STARS_J1);

     String Str_estrellas_J2 = String(STARS_J2);

      LCD_Print(Str_estrellas_J1, 60, 85, 2, 0x07FF,
0x0000);

           LCD_Print(Str_estrellas_J2,  205,  85,  2,
0x07FF, 0x0000);



     LCD_Bitmap(80, 85, 16, 16, estrella_16);

     LCD_Bitmap(240, 85, 16, 16, estrella_16);

     }


   else if (STARS_J1 >=10 && STARS_J2 <= 9){

     String Str_estrellas_J1 = String(STARS_J1);

     String Str_estrellas_J2 = String(STARS_J2);

      LCD_Print(Str_estrellas_J1, 55, 85, 2, 0x07FF,
0x0000);

           LCD_Print(Str_estrellas_J2,  210,  85,  2,
0x07FF, 0x0000);



     LCD_Bitmap(90, 85, 16, 16, estrella_16);

     LCD_Bitmap(230, 85, 16, 16, estrella_16);

     }


   else if (STARS_J1 >=10 && STARS_J2 >= 10){

     String Str_estrellas_J1 = String(STARS_J1);

     String Str_estrellas_J2 = String(STARS_J2);

      LCD_Print(Str_estrellas_J1, 55, 85, 2, 0x07FF,
0x0000);

           LCD_Print(Str_estrellas_J2,  205,  85,  2,
0x07FF, 0x0000);



     LCD_Bitmap(90, 85, 16, 16, estrella_16);

     LCD_Bitmap(240, 85, 16, 16, estrella_16);

     }



           String     Str_juegos_gnados_J1     =
String(Juegos_ganados_J1);

           String     Str_juegos_gnados_J2     =
String(Juegos_ganados_J2);

   String text_guion = "-";


      LCD_Print(Str_juegos_gnados_J1,  65,  190,  2,
0xFF40, 0x0000);//juegos ganados J1

      LCD_Print(text_guion,  145,  190,  2,  0xFF40,
0x0000);//guion

     LCD_Print(Str_juegos_gnados_J2, 230, 195, 2,
0xFF40, 0x0000);//JUEGOS ganadas J2
```

```
    delay(300);
}
```

Función para Reproducir Audio

Este es el main loop del programa en el que se reproduce la música, para ellos tenemos un arreglo de datos "melody[]" el cual está compuesto por cada nota seguida de su duración. Por lo que leemos ese arreglo de 2 en 2 datos para obtener cada nota con su respectiva duración, y luego se realizan los comandos necesarios para reproducir la canción de forma correcta.

```
void loop() {
    for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
    // Calculamos la duración de cada nota
   if (digitalRead(buttonPin) == HIGH){
    divider = melody[thisNote + 1];
    if (divider > 0) {
        // Si la nota es regular, seguimos sin hacer cambios
      noteDuration = (wholenote) / divider;
    } else if (divider < 0) {
        // Si el valor es negativo se representa una nota con punto (nota y media)
      noteDuration = (wholenote) / abs(divider);
      noteDuration *= 1.5; //Por lo que se multiplica por 1.5
    }
    // Suena únicamente el 90% de la duración de la nota para tener un 10% de pausa entre notas
      tone(buzzer, melody[thisNote], noteDuration * 0.9);
        // Esperamos la duración de cada nota específica antes de seguir a la siguiente nota
     delay(noteDuration);
      // Tocamos una nota "vacía" antes de tocar la siguiente nota
     noTone(buzzer);
   }
   else {
    thisNote = 0;
   }
  }
}
```