UNIVERSIDAD DEL VALLE DE GUATEMALA
ELECTRÓNICA DIGITAL 2
ELECTRÓNICA Y MECATRONICA
TRABAJO ESCRITO
RAÚL AGUILAR 17581
PABLO ARELLANO

PROYECTO 3 - VIDEOJUEGO PORTABLE

Introducción:

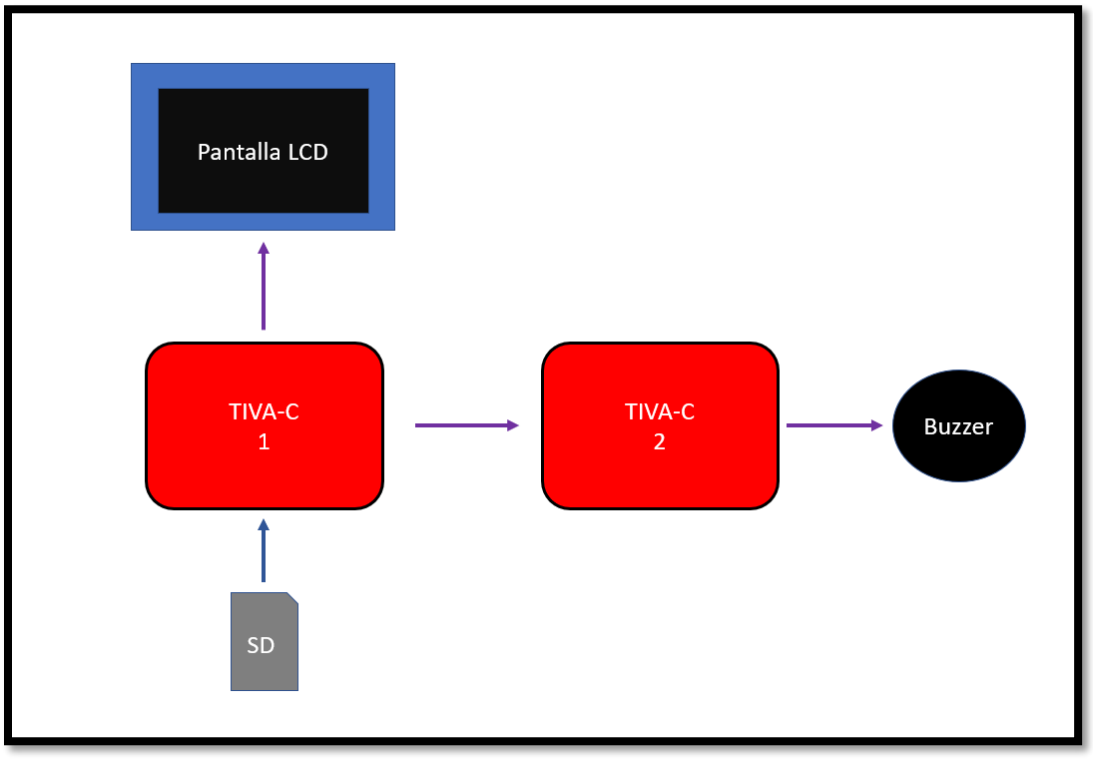
En el presente trabajo escrito se detalla y se copila toda la información necesaria para llevar a cabo el proyecto 3 de la clase de electrónica digital 2. El cual se basa en la elaboración de un videojuego implementado en la pantalla LCD ILI9341 en conjunto con el microcontrolador TIVA-C elaborado en el programa de energia. Como controles para el videojuego se usaron los dos botones que tiene le Tiva -C. Se uso una segunda Tiva para la implementación de dos melodías en 8 bits las cuales fueron Gourmet Race – Kirby y Take On me – a-ha para la melodía del jugador ganador.

Sobre el videojuego:

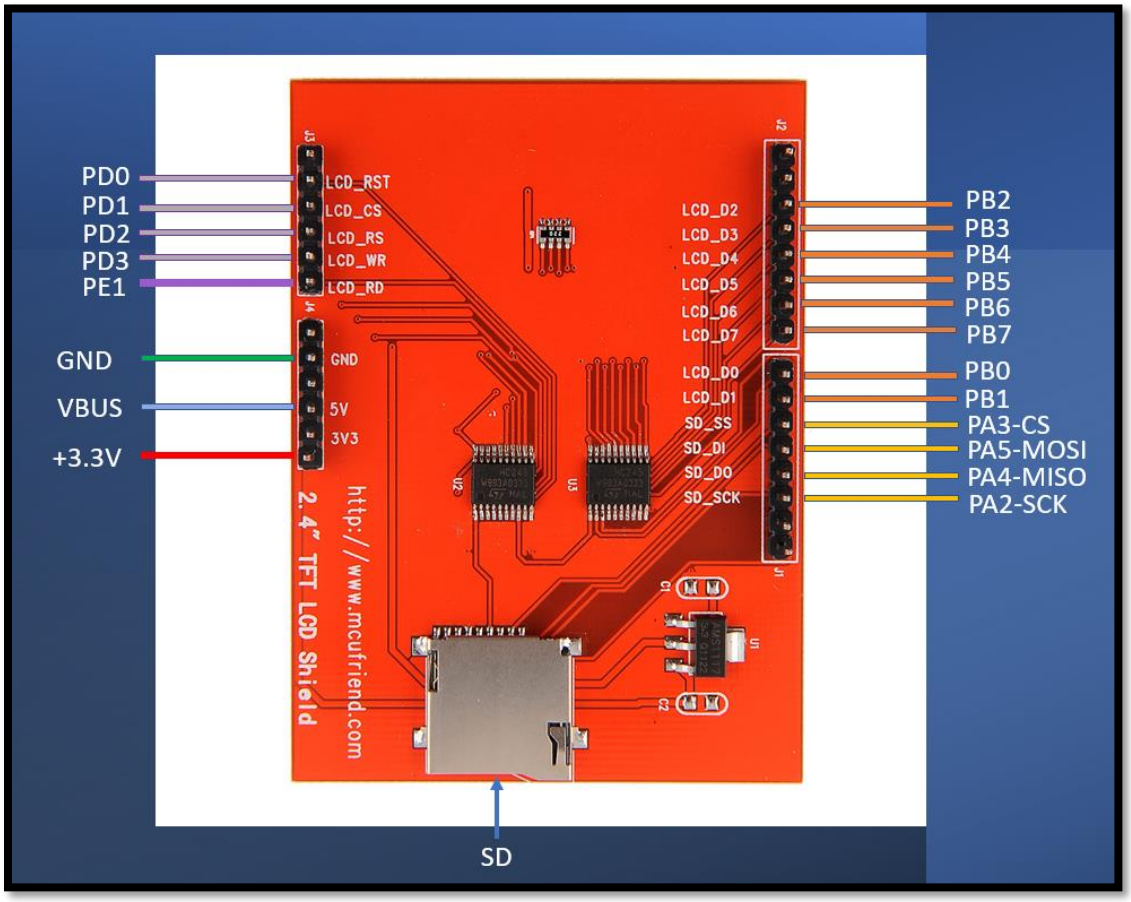
El videojuego realizado fue Flappy Bird con dos jugadores. El cual tiene como objetivo alcanzar la puntuación máxima de 9 puntos sin chocar con los pilares que se van generando. Cada vez que un jugador choca con un pilar se le irá descontando un punto del Score. El primero en lograr el Score máximo será el ganador del videojuego.

Circuitos empleados:

La siguiente imagen muestra el circuito principal del proyecto. Se puede observar el uso de la Pantalla LCD ILI9341 conectada a la Tiva uno. La cual se interconectada a la Tiva dos y el buzzer.



En esta imagen se observa la conexión de todos los pines que se usaron de la pantalla LCD ILI9341 a la Tiva-C uno. Cada uno con su respetivo pin de entrada en el microcontrolador.



Gráficos utilizados:

- Player1

- Player2

- Gráfico Portada1

- Gráfico Portada2

Análisis realizado:

Se eligió flappyBird ya que se llegó a la conclusión que era completamente implementable con las condiciones dadas. De Igual forma se observó que se podía cubrir con todos los requerimientos estipulados en la guía del proyecto, además de ser un juego divertido y hasta cierto punto retador para los usuarios.

Para poder llevar a cabo el Flappybrid se utilizó la plataforma energía en la Tiva-C, se utilizaron las respectivas librerías para la correcta inicialización de la pantalla LCD ILI9341 en energía.

```
#include <stdint.h>
#include <stdbool.h>
#include <TM4C123GH6PM.h>
#include "pitches.h"
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/rom_map.h"
#include "driverlib/rom.h"
#include "driverlib/sysctl.h"
#include "driverlib/timer.h"
#include "bitmaps.h"
#include "font.h"
#include "lcd_registers.h"
#include <SPI.h>
#include <SD.h>
```

Estructura de la programación TIVA-C Uno

1. **Inicio**- Bienvenida al jugador:
   i) Configuración de reloj.
   ii) Inicialización de la Tiva-C.
   iii) Títulos de los diseñaros del juego.
   iv) Títulos generales (Universidad, curso, departamento)
   v) Se llama a las imágenes de portada en la SD.

2. **Loop principal**:
   i) Se pone en 1 el pin OUT_MUSICA.
   ii) Se define la condición de el valor máximo de Score.
   iii) Si se cumple se manda a llamar a gameover.
   iv) Se define las coordenadas de los pilares.
   v) Se dibujan los pilares.
   vi) Se define la caída y la aceración de caída del ave.
   vii) Se define la condición de generación de los pilares.
   viii) Se define la altura random de los pilares superiores e inferiores.
   ix) Se lee los botones.

3. **Drawbird:**
   i) Se define la función de aleteo.
   ii) Se define en las coordenadas que se mostrara el Sprite del bird.
   iii) Se definen condiciones de vuelo del ave.
   iv) Se define cuando el bird pasa entre los pilares.

4. **DrawPilares:**
   i) Se define el Score de los jugadores.
   ii) Se muestra un cuadro celeste en donde se visualizarán.
   iii) Se definen pilares arriba y debajo de la pantalla.
   iv) Se definen pilares del mismo color que el fondo.
   v) Se define la generación de pilares de colores verde y atrás color del fondo.

5. **Gameover:**
   i) Se vuelve a definir la condición de el puntaje máximo en los dos jugadores.
   ii) Si uno de los dos jugadores gana, el pin OUT_MUSICA se pone en 0.
   iii) Se muestra texto definiendo el jugador que ganó.

```
//-------------------------------------------------------------------
// Variables
//-------------------------------------------------------------------

int anim = 0;                           // sprite
int aumentar = 0;
int anim2 = 0;
int aumentar2 = 0;

int buzzerPin = 40;                     // musica
int ledblue= 19;

const int buttonPush1 = PUSH1;          // controles
const int buttonPush2 = PUSH2;
int buttonState1 = 0;
int buttonState2 = 0;

int x;                                  // pilares uno
int y;
int xP = 239;
int yP = 66;

int yB = 25;                            // ave uno
int movingRate = 3;
int fallRateInt = 0;
float fallRate = 0;

int x2;                                 // pilares dos
int y2;
int xP2 = 239;
int yP2 = 226;

int yB2 = 185;                          // ave dos
int movingRate2 = 3;
int fallRateInt2 = 0;
float fallRate2 = 0;
```

1. **Setup.** Se inicializa la pantalla, se muestra la portada con datos generales del proyecto. Se le incluyen imágenes de bienvenida llamadas desde la SD. Se configuran los botones en pull-up.

```
void setup()
  {
  SysCtlClockSet(SYSCTL_SYSDIV_2_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
  Serial.begin(9600);
  GPIOPadConfigSet(GPIO_PORTB_BASE, 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7, GPIO_STRENGTH_8MA, GPIO_PIN_TYPE_STD_WPU);
  Serial.println("Inicio");
  LCD_Init();
  LCD_Clear(0x00);
  FillRect(0, 0, 239, 319, 0x1513);

  String PA7 = "Pablo Arellano 151379";
  String RL = "Raul Aguilar 17581";
  String U = "--Universidad del Valle--";
  String depa = "DEPTO. MECATRONICA";
  String curso = "Electronica Digital II";
  String pablo = "Ing. Pablo Mazariegos";
  String SC = "Score";
  String sep = "----------------------------------------";
  int h = 10;
  LCD_Print(U, 21, h, 1, 0xffff, 0x421b);
  LCD_Print(depa, 43, h + 15, 1, 0xffff, 0x1513);
  LCD_Print(curso, 28, h + 30, 1, 0xffff, 0x1513);
  LCD_Print(pablo, 30, h + 45, 1, 0xffff, 0x1513);
  LCD_Print(sep, 4, h + 60, 1, 0xffff, 0x1513);
  LCD_Print(PA7, 30, 100, 1, 0xffff, 0x1513);
  LCD_Print(RL, 37, 115, 1, 0xffff, 0x1513);



  pinMode(buttonPush1, INPUT_PULLUP);
  pinMode(buttonPush2, INPUT_PULLUP);
  pinMode(OUT_MUSICA, OUTPUT);
```

Se detalla a las coordenadas que se quieren las imágenes recortadas del Bird de bienvenida.

```
 Serial.print("Initializing SD card...");                // 3. Inicializar SD
 pinMode(PA_3, OUTPUT);                                  //    Salida por default para que libreria SD funcione

 if (!SD.begin(PA_3))                                    //    No se ha iniciado la comunicacion al mandar el SS?
 {
    Serial.println("initialization failed!");            //    Fallo la inicializacion
    return;
 }
 Serial.println("initialization done.");                 //    Ya se inicializo

LCD_SD_Bitmap(77, 200, 85, 45, "intr.txt");

delay (5000);

 LCD_SD_Bitmap(5, 160, 40, 40, "tw0.txt");
 LCD_SD_Bitmap(45, 160, 40, 40, "tw1.txt");
 LCD_SD_Bitmap(85, 160, 40, 40, "tw2.txt");
 LCD_SD_Bitmap(125, 160, 40, 40, "tw3.txt");
 LCD_SD_Bitmap(165, 160, 40, 40, "tw4.txt");
 LCD_SD_Bitmap(205, 160, 29, 40, "tw5.txt");

 LCD_SD_Bitmap(5, 200, 40, 40, "tw6.txt");
 LCD_SD_Bitmap(45, 200, 40, 40, "tw7.txt");
 LCD_SD_Bitmap(85, 200, 40, 40, "tw14.txt");
 LCD_SD_Bitmap(125, 200, 40, 40, "tw9.txt");
 LCD_SD_Bitmap(165, 200, 40, 40, "tw10.txt");
 LCD_SD_Bitmap(205, 200, 29, 40, "tw11.txt");

 LCD_SD_Bitmap(5, 240, 40, 40, "tw12.txt");
 LCD_SD_Bitmap(45, 240, 40, 40, "tw13.txt");
 LCD_SD_Bitmap(85, 240, 40, 40, "tw14.txt");
 LCD_SD_Bitmap(125, 240, 40, 40, "tw15.txt");
 LCD_SD_Bitmap(165, 240, 40, 40, "tw16.txt");
 LCD_SD_Bitmap(205, 240, 29, 40, "tw17.txt");

 LCD_SD_Bitmap(5, 280, 40, 35, "tw18.txt");
 LCD_SD_Bitmap(45, 280, 40, 35, "tw19.txt");
 LCD_SD_Bitmap(85, 280, 40, 35, "tw20.txt");
 LCD_SD_Bitmap(125, 280, 40, 35, "tw21.txt");
 LCD_SD_Bitmap(165, 280, 40, 35, "tw22.txt");
 LCD_SD_Bitmap(205, 280, 29, 35, "tw23.txt");



   delay(5000);

 LCD_Clear(0x0000);
 FillRect(0, 0, 240, 160, 0x9EDDb);
 FillRect(0, 161, 240, 320, 0x051Db);
```

2. **Loop.** Se pone en 1 el pin OUT_MUSICA. El cual da inicio a la melodía de Gourmet Race. Se define la condición del valor máximo de Score. Si se cumple se manda a llamar a gameover. Se define las coordenadas de los pilares. Se dibujan los pilares. Se define la caída y la aceración de caída del ave. Posteriormente se define las condiciones de generación de los pilares. Se define la altura random de los pilares superiores e inferiores y se lee los botones de la Tiva-c. De ultimo se detalla el choque de bird con los pilares y su decremento al score.

```
void loop()
 {
 digitalWrite(OUT_MUSICA, HIGH);
 H_line(0, 160, 240, 0x0000b);
 if (conta == 9 || conta2 == 9){
     gameover();
 }

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

 xP = xP - movingRate;         // coordenada de los pilares
 drawPilars(xP, yP);           // dibujar pilares
 drawBird(yB, xP, yP);
 yB+=fallRateInt;              // caida
 fallRate = fallRate+0.5;      // aceleracion
 fallRateInt = int(fallRate);

 if (xP<=-51)
 {
    xP=239;                    // Resets xP to 319
    yP = random(15,89);        // Random number for the pillars height
    control_conta = 1;
    control_choque = 1;
    movingRate = conta + 3;
 }

 buttonState2 = digitalRead(buttonPush2);
 if (buttonState2 == LOW)
 {
   fallRate = -5;
 }

 if( ((xP>50) && (xP<68)) && ((yB < yP) || ((yB+13)>(yP+60) )))
 {
   if (conta >= 1)
   {
     conta = conta - 1;

  else if (conta < 1)
  {
    conta = 0;
  }
  yB = yP + 30;
  control_choque = 0;
}
```

3. **drawBird.** En general esta función define los parámetros en el cual se dibujará el ave. Se estipula que el ave estará en una coordenada (50, yB). yB ira variando. De igual forma se restringe su plano de movimiento y se detalla su moviente entre los pilares y el aumento al Score.

```
//--------------------------------------------------------------------------
// bird Uno
//--------------------------------------------------------------------------

void drawBird(int yB, int xP, int yP)
{
  aumentar = aumentar + 1;
  if (aumentar > 2)
  {
    anim = anim + 1;
    aumentar = 0;
    if (anim > 2)
    {
      anim = 0;
    }
  }

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

  if ((xP < 0) || (xP > 68))
  {
    if((yB > 12)&&(yB < 147))
    {
      LCD_Sprite(50, yB, 18, 13, ave1,3, anim, 0, 0);
      FillRect(50, 12, 18, yB-12, 0x9EDDb);
      FillRect(50, yB+13, 18, 160-(yB+13), 0x9EDDb);
    }
    else if (yB <= 12)
    {
      yB = 12;
      fallRate = 10;
    }
    else
    {
      yB = 147;
      fallRate = -10;
  // --------------------------------------------------------------------------

  if ((xP >= 0) && (xP <= 68))
  {
    if( (yB > yP) && ( (yB+13) < (yP+60)) )
    {
      LCD_Sprite(50, yB, 18, 13, ave1,3, anim, 0, 0);
      FillRect(50, yP, 18, abs(yB-yP), 0x9EDDb);
      FillRect(50, yB+13, 18, abs((yP+60)-(yB+13)), 0x9EDDb);
      if ((xP < 10) && (control_conta == 1) && (control_choque == 1))
      {
         conta++;
         control_conta = 0;
         control_choque = 0;
      }
    }
    else if (yB < yP)
    {
      yB = yB;
      fallRate = 5;
    }
    else if ((yB+13)>(yP+60))
    {
      yB = yB;
      fallRate = -5;
    }
  }
}
```

4. **drawPilar.** Aquí se programa el Score acendente de los jugadores y su vizulisación en la pantalla. Posteriormente se detalla el dibujado de pilares superiores e inferirores en diferntes coordenadas y se automatico dibujado del mismo color del fono para dar el efecto de movimiento.

```
//-------------------------------------------------------------------------------
// Pilares
//-------------------------------------------------------------------------------

void drawPilars(int x, int y)
  {
  String text1 = " Score:";
  String stringOne =  String(conta);
  FillRect(0, 0, 240, 12, 0x1513);
  LCD_Print(text1, 170, 0, 1, 0xffff, 0x1513);
  LCD_Print(stringOne, 225, 0, 1, 0xffff, 0x1513);

  if (x>=190)
  {
     FillRect(x, 12, 240-x, y-12, 0x0642b);  // VERDE ARRIBA
     FillRect(x, y+60, 240-x, 160-(y+60), 0x0642b); // VERDE ABAJO
     FillRect(0, 12, 10, 160-12, 0x9EDDb);
  }
  else if((x<=189) && (x > 0))
  {
     FillRect(x, 12, 50, y-12, 0x0642b);
     FillRect(x, y+60, 50, 160-(y+60), 0x0642b);
     FillRect(x+50, 12, 15, y-12, 0x9EDDb);
     FillRect(x+50, y+60, 15, 160-(y+60), 0x9EDDb);
  }
  else if ((x>-51) && (x < 0))
  {
     FillRect(0, 12, 50+x, y-12, 0x0642b);
     FillRect(0, y+60, 50+x, 160-(y+60), 0x0642b);
     FillRect(50+x, 12, 15, y-12, 0x9EDDb);
     FillRect(50+x, y+60, 15, 160-(y+60), 0x9EDDb);
  }

}
```

5. **gameover.** Función que permite la correcta finalización del videojuego.

```
//-------------------------------------------------------------------------
// GAMEOVER
//-------------------------------------------------------------------------
void gameover()
{
  while(conta == 9 || conta2 == 9)
  {
    if(conta==9)
    {
      digitalWrite(OUT_MUSICA, LOW);
      String Wp= "Congratulation";
      String W= "Player1 Win";
      FillRect(0, 0, 240, 320, 0x0000b);
      LCD_Print(Wp, 0, 135, 2, 0xffff, 0x0000b);
      LCD_Print(W, 1, 170, 2, 0xffff, 0x0000b);

    }
    if(conta2==9)
    {
      digitalWrite(OUT_MUSICA, LOW);
      String Wp= "Congratulation";
      String W= "Player2 Win";
      //FillRect(0, 0, 240, 320, 0x9EDDb);
      FillRect(0, 0, 240, 320, 0x0000b);
      LCD_Print(Wp, 0, 135, 2, 0xffff, 0x0000b);
      LCD_Print(W, 1, 170, 2, 0xffff, 0x0000b);

    }
  }
}
```

6. **Vertical Screen:** Cabe resaltar el siguiente registro que se uso para poder visualizar la pantalla en vertical con el comando 0x36 y luego se envia el dato 0x48 ^ 0xC0.

```
//*****************************************
LCD_CMD(0x36);  // (MEMORYACCESS)
LCD_DATA(0x48 ^ 0xC0); // LCD_DATA(0x19);
```

7. **Carácter a Numero:** Otra función importante es la siguiente. En la que se detalla la conversión de un carácter en número para poder visualizar las imágenes en la SD.

```
//*************************************************************************************************************
//  convertir caracter en numero
//*************************************************************************************************************
unsigned char Char_to_uChar(char letra){     // letra = "0"   letra = 48
  unsigned char num;
  if(letra>=48 && letra <=57){
    num = letra - 48;                        // num = 0
  }
  else if (letra >= 97 && letra <=102){      // convierte a letra
    num = letra -87;                         // a valor numerico
  }
  return num;
}
```

8. **Dibujar de SD:** El siguiente código permitió dibujar las imágenes guardadas en la SD por medio de un arreglo de colores.

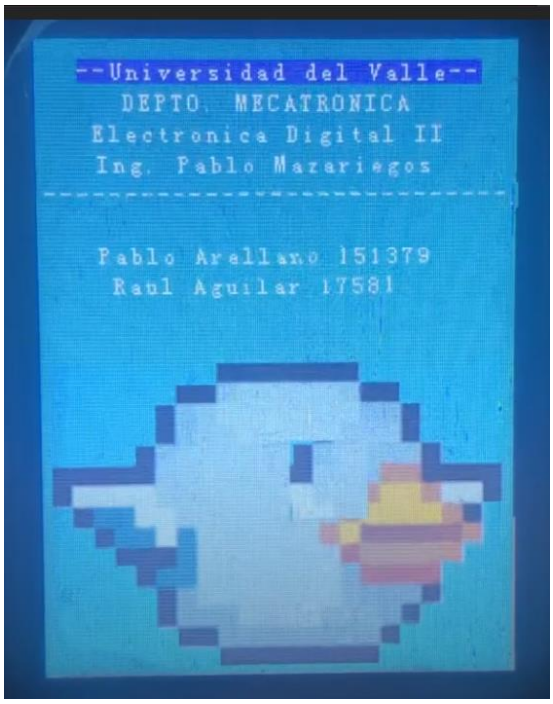
```
/*****************************************************************************************************************
/ Función para dibujar una imagen a partir de un arreglo de colores
/*****************************************************************************************************************

oid LCD_SD_Bitmap(unsigned int x, unsigned int y, unsigned int width, unsigned int height, char * direccion){
 File myFile = SD.open(direccion);
 uint16_t n = 0;                                          // contador
 uint16_t dimension = width*height*2;                     // tamanio del array
 unsigned char bitmap_SD[dimension] = {};                 // array vacio de dimension ancho por alto por 2
 if (myFile) {

   while (myFile.available()) {

     unsigned char numero = 0;                            // inicializo variable
     for(uint8_t m = 0; m < 2; m++){                      // cada elemento va de dos en dos
       char caracter = myFile.read();
       unsigned char digito = Char_to_uChar(caracter);    // lo escribo en forma unsigned char
       if (m == 0){
         numero = digito*16;                              // variable lo multiplico por 16 porque son dos bytes,
       }
       else if (m == 1){
         numero = numero + digito;                        // segundo digito
       }
     }
     bitmap_SD[n] = numero;
     n ++;                                                // contando el tamanio del arreglo
   }
   myFile.close();
 } else {
   Serial.println("error opening ");
 }
  LCD_Bitmap(x,y,width,height,bitmap_SD);
```

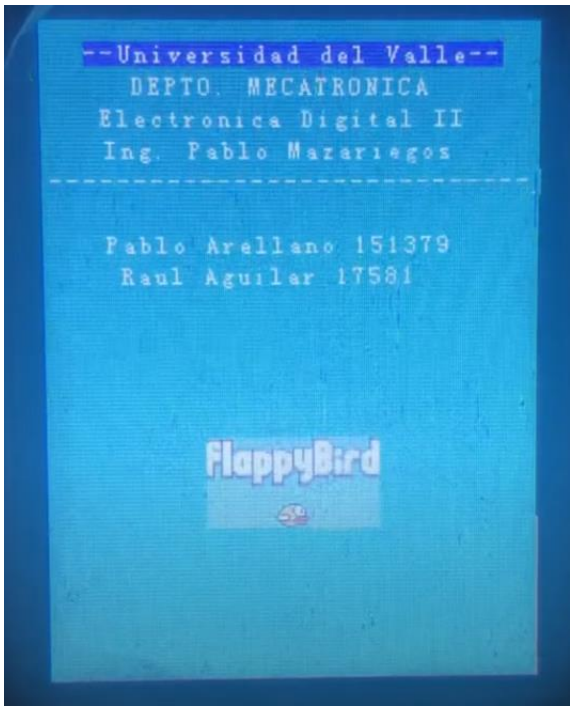
Estructura de la programación TIVA-C 2 (Música)

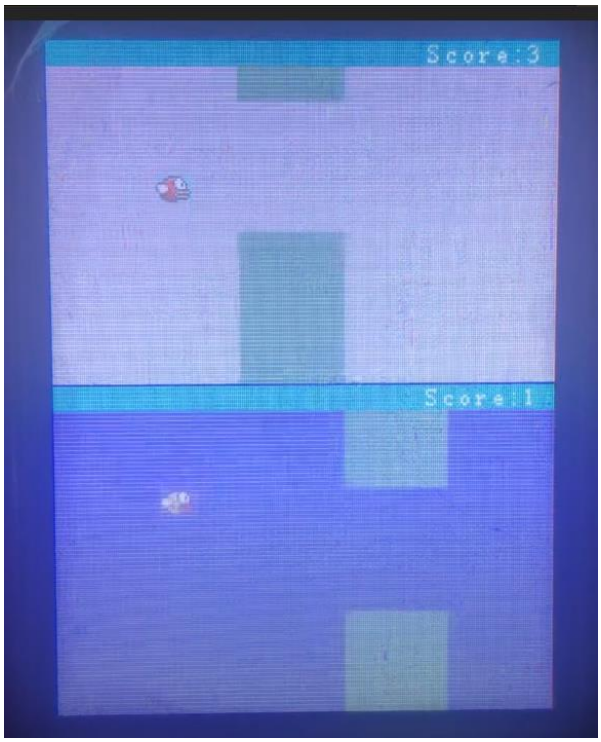
1. Se agrega la librería <pitches.h>
2. Se definen las notas de la melodía.
3. Se define la duración e la notas.
4. Se define el largo de la canción.
5. Loop.
    i. Se define la duración de las notas.
    ii. Se define las pausas entre las notas.

```
void loop()
{
  buttonState = digitalRead(ledblue);
  if (buttonState == HIGH)
    {
  for (int thisNote = 0; thisNote < 130; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000/noteDurations[thisNote];
    tone(buzzerPin, melody[thisNote],noteDuration);
    int pauseBetweenNotes = noteDuration *1.30-50; //delay between pulse
    delay(pauseBetweenNotes);
    delay(50);

    noTone(buzzerPin);                // stop the tone playing
  }
  }

  else if (buttonState == LOW)
    {
    for (int thisNote2 = 0; thisNote2 < songLength2; thisNote2++){
    // determine the duration of the notes that the computer understands
    // divide 1000 by the value, so the first note lasts for 1000/8 milliseconds
    int duration2 = 1000/ durations[thisNote2];

    tone(buzzerPin, melody2[thisNote2], duration2);
    // pause between notes
    int pause = duration2 * 1.3;
    delay(pause);
    // stop the tone
    noTone(8);
  }
    }
```

Proyecto Terminado:



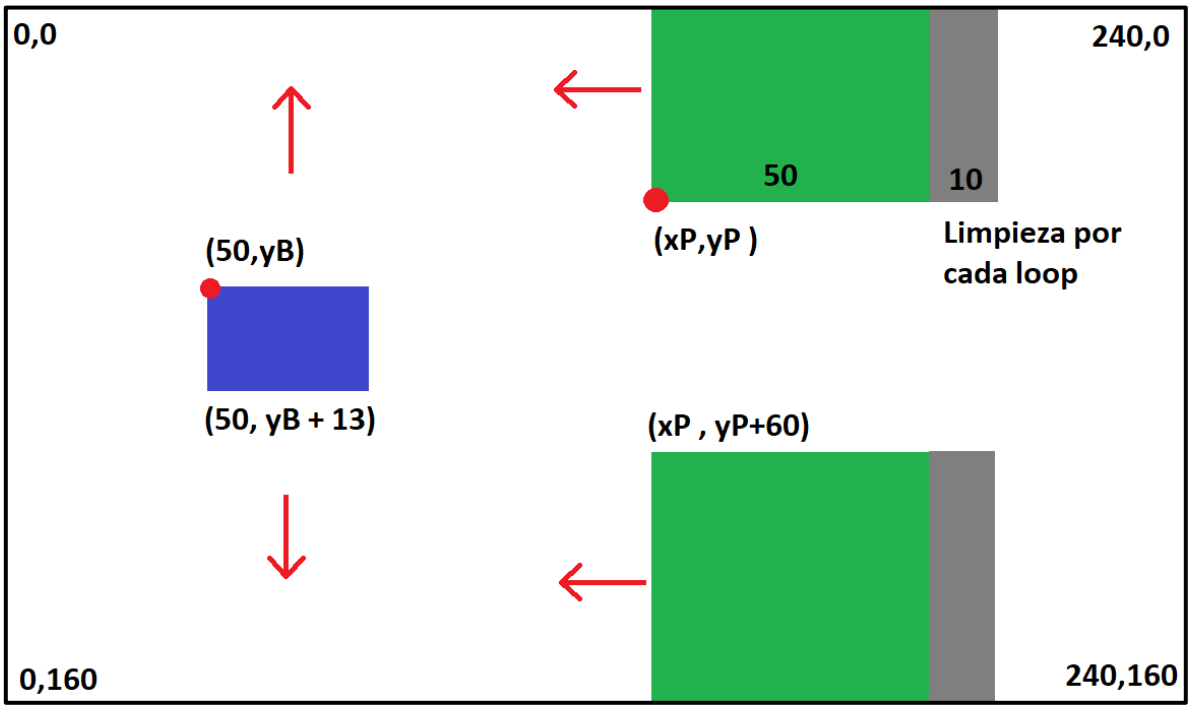



Score:3
Score:1


Congratulation
Player1 Win

Anexos:

Diagrama para Posicionamiento de Elementos



Bitmaps en Tiva

```
 unsigned char ave1[]={
0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e,
0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x51, 0xc8, 0x51, 0xc8, 0x51,
0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x51, 0xc8, 0x51, 0xc8, 0xfb, 0x84, 0xfb, 0x84, 0xfb, 0x84, 0x51,
0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x51, 0xc8, 0xfb, 0x84, 0xfb, 0x84, 0xf9, 0xc3, 0xf9, 0xc3, 0x51, 0xc8, 0xff,
0x9e, 0xdd, 0x9e, 0xdd, 0x51, 0xc8, 0x51, 0xc8, 0x51, 0xc8, 0x51, 0xc8, 0xf9, 0xc3, 0xf9, 0xc3, 0xf9, 0xc3, 0x51, 0xc8, 0xd7,
0x9e, 0xdd, 0x51, 0xc8, 0xff, 0xdf, 0xff, 0xdf, 0xff, 0xdf, 0xff, 0xdf, 0x51, 0xc8, 0xf9, 0xc3, 0xf9, 0xc3, 0x51, 0xc8, 0xd7,
0x9e, 0xdd, 0x51, 0xc8, 0xff, 0xdf, 0xff, 0xdf, 0xff, 0xdf, 0xff, 0xdf, 0xff, 0xdf, 0x51, 0xc8, 0xf9, 0xc3, 0xf9, 0xc3, 0x51,
0x9e, 0xdd, 0x51, 0xc8, 0xd7, 0x39, 0xff, 0xdf, 0xff, 0xdf, 0xff, 0xdf, 0xd7, 0x39, 0x51, 0xc8, 0xf9, 0xc3, 0xf9, 0xc3, 0xf9,
0x9e, 0xdd, 0x9e, 0xdd, 0x51, 0xc8, 0xd7, 0x39, 0xd7, 0x39, 0xd7, 0x39, 0x51, 0xc8, 0xf9, 0xc3, 0xf9, 0xc3, 0xf9, 0xc3, 0x51,
0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x51, 0xc8, 0x51, 0xc8, 0x51, 0xc8, 0xd1, 0x82, 0xd1, 0x82, 0xd1, 0x82, 0x51, 0xc8, 0xf5,
0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x51, 0xc8, 0xd1, 0x82, 0xd1, 0x82, 0xd1, 0x82, 0xd1, 0x82, 0xd1, 0x82, 0xd1, 0x82, 0x51,
0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x51, 0xc8, 0x51, 0xc8, 0xd1, 0x82, 0xd1, 0x82, 0xd1, 0x82, 0xd1, 0x82, 0xd1,
0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x51, 0xc8, 0x51, 0xc8, 0x51, 0xc8, 0x51, 0xc8, 0x51,
0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e, 0xdd, 0x9e,
};
```

Archivos en la SD