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

/* Se utilizó la librería para el uso de la pantalla ILI9341 en modo 8 bits
  Basado en el código de martinayotte - https://www.stm32duino.com/viewtopic.php?t=637
  Adaptación, migración y creación de nuevas funciones: Pablo Mazariegos y José Morales
  Con ayuda de: José Guerra
  IE3027: Electrónica Digital 2 - 2019

  Proyecto por: Camila Lemus y Larry Paul
*/
//**************************************************************************************
*************************************************
#include <stdint.h>
#include <stdbool.h>
#include <cstdlib>
#include <TM4C123GH6PM.h>
//#include <graficos.c>
#include <SPI.h>
#include <SD.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"

#define LCD_RST PD_0
#define LCD_CS PD_1
#define LCD_RS PD_2
#define LCD_WR PD_3
#define LCD_RD PE_1

#define jump1 PUSH1
#define duck1  PUSH2
#define jump2 PE_3 //12
#define duck2 PE_2 //13

extern uint8_t pastel [];
extern uint8_t grama [];
extern uint8_t dino [];
extern uint8_t nube [];
extern uint8_t dino_agachado [];
extern uint8_t globo [];
extern uint8_t regalo [];

volatile int d1_s = 0;
volatile int d2_s = 0;
volatile int d1_d = 0;
volatile int d2_d = 0;
```

```
volatile int lastd1_s = !d1_s;
volatile int lastd2_s = !d2_s;
volatile int lastd1_d = !d1_d;
volatile int lastd2_d = !d2_d;

int s = 0;
int s2 = 0;
int obj_f_l = 1;
int obj_f_r = 1;
int obj_l  = 0;
int rx_r = 160;
int gx_r = 160;
int rx_l = 160;
int gx_l = 160;
int obj_r  = 0;
int contsalto = 0;
int contsalto2 = 0;
int agache_activo = 0;
int agache_activo2 = 0;
int jumping = 0;
int jumping2 = 0;

int coordy1 = 0;
int coordx1 = 0;
int obsty = 0;
int obstx = 0;
int coordy2 = 0;
int coordx2 = 0;
```

```
int DPINS[] = {PB_0, PB_1, PB_2, PB_3, PB_4, PB_5, PB_6, PB_7};

int i = 0;

int conta = 0;

int gameover1 = 0;


File myFile;

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

// Functions Prototypes

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

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_FONDO(unsigned int x, unsigned int y, unsigned int width, unsigned int height);

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);


void flag_d1s();

void flag_d2s();

void flag_d1d_r();
```

```
void flag_d2d_f();
void flag_d2d_r();

//*************************************************************************************************
// Inicialización
//*************************************************************************************************
void setup() {
  SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
  Serial.begin(115200);
  GPIOPadConfigSet(GPIO_PORTB_BASE, 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7, GPIO_STRENGTH_8MA,
GPIO_PIN_TYPE_STD_WPU);
  pinMode(jump1, INPUT_PULLUP);
  pinMode(jump2, INPUT_PULLUP);
  pinMode(duck1, INPUT);
  pinMode(duck2, INPUT);
  attachInterrupt(digitalPinToInterrupt(jump1), flag_d1s, FALLING);
  attachInterrupt(digitalPinToInterrupt(jump2), flag_d2s, FALLING);
  //  attachInterrupt(digitalPinToInterrupt(duck1), flag_d1d_r, CHANGE);
  //  attachInterrupt(digitalPinToInterrupt(duck2), flag_d2d_r, CHANGE);
  Serial.println("Inicio");
  LCD_Init();
  LCD_Clear(0x00);


  //Memoria SD
  SPI.setModule(0);  //iniciamos comunicacion SPI en el modulo 0
  Serial.print("Initializing SD card...");
  pinMode(12, OUTPUT);  //Colocamos el CS del modulo SPI-0 como Output
```

```
  //Se verifica que se haya iniciado correctamente la SD
  if (!SD.begin(12)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
  //****



  delay(10);
  LCD_FONDO(0, 0, 320, 240);
  delay(500);


  //FillRect(0, 0, 319, 239, 0x421b);
  FillRect(0, 0, 319, 223, 0xffff);
  String text1 = "Dino B-Day Party!";
  LCD_Print(text1, 20, 100, 2, 0x0000, 0xffff);
  //LCD_Sprite(int x, int y, int width, int height, unsigned char bitmap[],int columns, int index, char flip, char offset);


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


  for (int x = 0; x < 340; x++) {
    LCD_Bitmap(x, 224, 16, 16, grama);
    x += 15;
  }


}
```

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

// Loop Infinito

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

void loop() {

 i = 150;

 LCD_Sprite(i, 40, 50, 17, nube, 1, 0, 0, 0);

 //   V_line (i-1, 40, 50, 0xffff);

 conta++;

 int anim = (conta / 11) % 2;


 //*****************************************************Pastel y globo moviendose

 if (obj_f_l == 0) {

  switch (obj_l) { //Case de los objetos generados parte izquierda

    case 0:

      LCD_Bitmap(rx_l, 201, 18, 22, pastel);

      obstx = rx_l;

      obsty = 201;

      rx_l = rx_l - 1;

      V_line (rx_l + 34, 202, 18, 0xffff);

      V_line (rx_l + 33, 202, 18, 0xffff);

      V_line (rx_l + 32, 202, 18, 0xffff);

      V_line (rx_l + 31, 202, 18, 0xffff);

      V_line (rx_l + 30, 202, 18, 0xffff);

      V_line (rx_l + 29, 202, 18, 0xffff);

      V_line (rx_l + 28, 202, 18, 0xffff);

      V_line (rx_l + 27, 202, 18, 0xffff);

      V_line (rx_l + 26, 202, 18, 0xffff);
```

```
      V_line (rx_l + 25, 202, 18, 0xffff);


      if (rx_l == 0) {
        rx_l = 160; //Posicion inicial al cruzar limite izquierda
        obj_f_l = 1;
        for (int i = 0; i<18; i++){
      V_line (i, 150, 40, 0xffff); // Borro el sprite de la ultima posiciion ya que for no borrra el ultimo traslado en gx
      }
      }
      break;
     case 1 :
      LCD_Sprite(gx_l, 150, 12, 40, globo, 3, 0, 0, 0); // Globo
      obstx = gx_l;
      obsty = 150;
      gx_l = gx_l - 1;
      for (int i=12; i<25; i++){
        V_line (gx_l + i, 150, 40, 0xffff);
        }


//      V_line (gx + 10, 150, 40, 0xffff);
//      V_line (gx + 9, 150, 40, 0xffff);
//      V_line (gx + 8, 150, 40, 0xffff);
//      V_line (gx + 7, 150, 40, 0xffff);
//      V_line (gx + 6, 150, 40, 0xffff);
//      V_line (gx + 5, 150, 40, 0xffff);
//      V_line (gx + 4, 150, 40, 0xffff);
//      V_line (gx + 3, 150, 40, 0xffff);
```

```
      if (gx_l == 0) {
        gx_l = 160;
        obj_f_l = 1;
       for (int i = 0; i<12; i++){
        V_line (i, 150, 40, 0xffff); // Borro el sprite de la ultima posiciion ya que for no borrra el ultimo traslado en gx
      }
      }
      break;
   }
  }
  else if (obj_f_l == 1) {
   //int obj  = 2;
   obj_f_l = 0;
   obj_l = (rand() % 2); // Generación del numero aleatorio para el case izquierdo.
  }

  if (obj_f_r == 0) { // Codigo generación de obstaculos alaeatorios
   switch (obj_r) {
     case 0:
      LCD_Bitmap(rx_r, 201, 18, 22, pastel);  //Impresion de pastel en caso de tener el caso 0
      obstx = rx_r;
      obsty = 201;
      rx_r = rx_r + 1;
      V_line (rx_r - 18, 202, 18, 0xffff);
      V_line (rx_r - 17, 202, 18, 0xffff);
      V_line (rx_r - 16, 202, 18, 0xffff);
      V_line (rx_r - 15, 202, 18, 0xffff);
      V_line (rx_r - 14, 202, 18, 0xffff);
```

```
  V_line (rx_r - 13, 202, 18, 0xffff);
  V_line (rx_r - 12, 202, 18, 0xffff);
  V_line (rx_r - 11, 202, 18, 0xffff);
  V_line (rx_r - 10, 202, 18, 0xffff);
  V_line (rx_r - 9, 202, 18, 0xffff);

  if (rx_r == 320) { // Se chequea si el obstaculo ya salio de la pantalla cuando sale
    rx_r = 160; // Reiniciamos y levantamos bandera.
    obj_f_r = 1;
  }
  break;
 case 1 :
  LCD_Sprite(gx_r, 150, 12, 40, globo, 3, 0, 0, 0); // Globo
  obstx = gx_r;
  obsty = 150;
  gx_r = gx_r + 1;
  V_line (gx_r - 10, 150, 40, 0xffff);
  V_line (gx_r - 9, 150, 40, 0xffff);
  V_line (gx_r - 8, 150, 40, 0xffff);
  V_line (gx_r - 7, 150, 40, 0xffff);
  V_line (gx_r - 6, 150, 40, 0xffff);
  V_line (gx_r - 5, 150, 40, 0xffff);
  V_line (gx_r - 4, 150, 40, 0xffff);
  V_line (gx_r - 3, 150, 40, 0xffff);

  if (gx_r == 320) {
    gx_r = 160;
    obj_f_r = 1;
  }
```

```
      break;
    }
  }
  else if (obj_f_r == 1) {
    //int obj  = 2;
    obj_f_r = 0;
    obj_r = (rand() % 2);  // Numero aleatorio del lado derecho.
  }


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


  if (d1_s) { // Comparación que ejecuta el saltio del jugador 1
    contsalto++;
    jumping = 1;
    s = (contsalto) % 51; //variable que hace mod de 51 y se suma 1 vez cada vez que se repite el loop.
numeros del 0 al 50
    if (s < 25) { // De 0 a 25, se hara el salto para arriba, teniendo 25 como valor del salto maximo
      coordy1 = 180 - s; // Posicion original menos s
      LCD_Sprite(0, coordy1, 31, 42, dino, 2 , 0, 0, 0);
      for (int sub1 = 0; sub1 < 6; sub1++ ) {
        H_line(0, (180 + 42) - s - sub1, 31, 0xffff); // for para borrar el rastro del dinosaurio por el eje y
      }
    }
    else if (s == 25) { // Cuando s es igual a 25 el dinosaurio queda estatico en el maximo
      coordy1 = 180 - 25;
      LCD_Sprite(0, coordy1, 31, 42, dino, 2 , 0, 0, 0); //Salto del dinosaurio 1
    }
    else if (s > 25 and s < 50) { // Cuando sobrepasa s el valor de s empieza a caer a la posicion original.
```

```
    coordy1 = 180 - 25 + (s - 25);
    LCD_Sprite(0, coordy1 , 31, 42, dino, 2 , 0, 0, 0);
    for (int cae1 = 0; cae1 < 6; cae1++ ) {
      H_line(0, 180 - 25 + ((s) - 27) - cae1, 31, 0xffff);
    }


   }
   else if (s >= 50) { //al terminar el salto Bajo bandera de salto y reinicio variables
    d1_s = 0;
    jumping = 0;
   }


  }
  else if (digitalRead(duck1) == HIGH and d1_s == LOW and agache_activo == 0 and jumping == 0) { //
Esta es la posición normal, cuando nada esta presionado el codigo cae a esta zona.
   coordy1 = 180;
   LCD_Sprite(0, coordy1, 31, 42, dino, 2 , anim, 0, 0);      //Dinosaurio 1
// El dinosaurio estara estatico, cambiando de animacion
  }



  if (d2_s) { // Chequeo bandera de rutina salto
   contsalto2++;
   jumping2 = 1;
   s2 = (contsalto2) % 51;
   if (s2 < 25) {
    coordy2 = 180 - s2;
    LCD_Sprite(288, coordy2, 31, 42, dino, 2 , 0, 1, 0);
   }
```

```
  else if (s2 == 25) {
    coordy2 = 180 - 25;
    LCD_Sprite(288, coordy2, 31, 42, dino, 2 , 0, 1, 0);
  }
  else if (s2 > 25 and s2 < 50) {
    coordy2 = 180 - 25 + (s2 - 25);
    LCD_Sprite(288, coordy2, 31, 42, dino, 2 , 0, 1, 0);
    H_line(288, 180 - 25 + ((s2) - 26), 31, 0xffff);

  }
  else if (s2 == 50) { // apago bandera rutina y reinicio variable
    d2_s = 0;
    jumping2 = 0;
  }
}
else if (digitalRead(duck2) == HIGH and d2_s == LOW and agache_activo2 == 0 and jumping2 == 0) {
  coordy2 = 180;
  LCD_Sprite(288, 180, 31, 42, dino, 2 , anim, 1, 0);      //Dinosaurio 2
}

if (digitalRead(duck1) == LOW) {
  coordy1 = 180 + 11;
  LCD_Sprite(0, coordy1, 45, 31, dino_agachado, 2, anim, 0, 0);
  for (int dow = 0; dow <= 11; dow++) {
    H_line(0, 180 + 11 - dow, 45, 0xffff);
  }
  agache_activo = 1;
}
else if (digitalRead(duck1) == HIGH and d1_s == LOW and agache_activo == 1) {
```

```
  V_line(46 - 1, 180 + 11, 31, 0xffff);
  V_line(46 - 2, 180 + 11, 31, 0xffff);
  V_line(46 - 3, 180 + 11, 31, 0xffff);
  V_line(46 - 4, 180 + 11, 31, 0xffff);
  V_line(46 - 5, 180 + 11, 31, 0xffff);
  V_line(46 - 6, 180 + 11, 31, 0xffff);
  V_line(46 - 7, 180 + 11, 31, 0xffff);
  V_line(46 - 8, 180 + 11, 31, 0xffff);
  V_line(46 - 9, 180 + 11, 31, 0xffff);
  V_line(46 - 10, 180 + 11, 31, 0xffff);
  V_line(46 - 11, 180 + 11, 31, 0xffff);
  V_line(46 - 12, 180 + 11, 31, 0xffff);
  V_line(46 - 13, 180 + 11, 31, 0xffff);
  V_line(46 - 14, 180 + 11, 31, 0xffff);
  V_line(46 - 15, 180 + 11, 31, 0xffff);

  agache_activo = 0;
}

if (digitalRead(duck2) == LOW ) {
  coordy2 = 180 + 11;
  LCD_Sprite(275, coordy2, 45, 31, dino_agachado, 2, anim, 1, 0);
  for (int dow = 0; dow <= 11; dow++) {
    H_line(288, 180 + 11 - dow, 45, 0xffff);
  }
  agache_activo2 = 1;
}
else if (digitalRead(duck2) == HIGH and d2_s == LOW and agache_activo2 == 1) {
```

```
  V_line(275 + 1, 180 + 11, 31, 0xffff);
  V_line(275 + 2, 180 + 11, 31, 0xffff);
  V_line(275 + 3, 180 + 11, 31, 0xffff);
  V_line(275 + 4, 180 + 11, 31, 0xffff);
  V_line(275 + 5, 180 + 11, 31, 0xffff);
  V_line(275 + 6, 180 + 11, 31, 0xffff);
  V_line(275 + 7, 180 + 11, 31, 0xffff);
  V_line(275 + 8, 180 + 11, 31, 0xffff);
  V_line(275 + 9, 180 + 11, 31, 0xffff);
  V_line(275 + 10, 180 + 11, 31, 0xffff);
  V_line(275 + 11, 180 + 11, 31, 0xffff);
  V_line(275 + 12, 180 + 11, 31, 0xffff);
  V_line(275 + 13, 180 + 11, 31, 0xffff);
  V_line(275 + 14, 180 + 11, 31, 0xffff);
  V_line(275 + 15, 180 + 11, 31, 0xffff);

   agache_activo2 = 0;
 }



 int ycol1 = coordy1 + 41;
 int ycolpas = obsty + 22;
 if ((coordy1 < obsty < ycol1) or (obsty < ycolpas) and gameover1 == 0 and obstx < 31) {

   gameover1 = 1;
 }
}
```

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

// Función para inicializar LCD

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

void LCD_Init(void) {
  pinMode(LCD_RST, OUTPUT);
  pinMode(LCD_CS, OUTPUT);
  pinMode(LCD_RS, OUTPUT);
  pinMode(LCD_WR, OUTPUT);
  pinMode(LCD_RD, OUTPUT);
  for (uint8_t i = 0; i < 8; i++) {
    pinMode(DPINS[i], OUTPUT);
  }
  //*************************************
  // Secuencia de Inicialización
  //*************************************
  digitalWrite(LCD_CS, HIGH);
  digitalWrite(LCD_RS, HIGH);
  digitalWrite(LCD_WR, HIGH);
  digitalWrite(LCD_RD, HIGH);
  digitalWrite(LCD_RST, HIGH);
  delay(5);
  digitalWrite(LCD_RST, LOW);
  delay(20);
  digitalWrite(LCD_RST, HIGH);
  delay(150);
  digitalWrite(LCD_CS, LOW);
  //*************************************
```

```
LCD_CMD(0xE9);  // SETPANELRELATED
LCD_DATA(0x20);
//*************************************
LCD_CMD(0x11); // Exit Sleep SLEEP OUT (SLPOUT)
delay(100);
//*************************************
LCD_CMD(0xD1);    // (SETVCOM)
LCD_DATA(0x00);
LCD_DATA(0x71);
LCD_DATA(0x19);
//*************************************
LCD_CMD(0xD0);   // (SETPOWER)
LCD_DATA(0x07);
LCD_DATA(0x01);
LCD_DATA(0x08);
//*************************************
LCD_CMD(0x36);  // (MEMORYACCESS)
LCD_DATA(0x40 | 0x80 | 0x20 | 0x08); // LCD_DATA(0x19);
//*************************************
LCD_CMD(0x3A); // Set_pixel_format (PIXELFORMAT)
LCD_DATA(0x05); // color setings, 05h - 16bit pixel, 11h - 3bit pixel
//*************************************
LCD_CMD(0xC1);    // (POWERCONTROL2)
LCD_DATA(0x10);
LCD_DATA(0x10);
LCD_DATA(0x02);
LCD_DATA(0x02);
//*************************************
LCD_CMD(0xC0); // Set Default Gamma (POWERCONTROL1)
```

```
LCD_DATA(0x00);
LCD_DATA(0x35);
LCD_DATA(0x00);
LCD_DATA(0x00);
LCD_DATA(0x01);
LCD_DATA(0x02);
//*************************************
LCD_CMD(0xC5); // Set Frame Rate (VCOMCONTROL1)
LCD_DATA(0x04); // 72Hz
//*************************************
LCD_CMD(0xD2); // Power Settings  (SETPWRNORMAL)
LCD_DATA(0x01);
LCD_DATA(0x44);
//*************************************
LCD_CMD(0xC8); //Set Gamma  (GAMMASET)
LCD_DATA(0x04);
LCD_DATA(0x67);
LCD_DATA(0x35);
LCD_DATA(0x04);
LCD_DATA(0x08);
LCD_DATA(0x06);
LCD_DATA(0x24);
LCD_DATA(0x01);
LCD_DATA(0x37);
LCD_DATA(0x40);
LCD_DATA(0x03);
LCD_DATA(0x10);
LCD_DATA(0x08);
LCD_DATA(0x80);
```

```
 LCD_DATA(0x00);
 //*************************************
 LCD_CMD(0x2A); // Set_column_address 320px (CASET)
 LCD_DATA(0x00);
 LCD_DATA(0x00);
 LCD_DATA(0x01);
 LCD_DATA(0x3F);
 //*************************************
 LCD_CMD(0x2B); // Set_page_address 480px (PASET)
 LCD_DATA(0x00);
 LCD_DATA(0x00);
 LCD_DATA(0x01);
 LCD_DATA(0xE0);
 //  LCD_DATA(0x8F);
 LCD_CMD(0x29); //display on
 LCD_CMD(0x2C); //display on


 LCD_CMD(ILI9341_INVOFF); //Invert Off
 delay(120);
 LCD_CMD(ILI9341_SLPOUT);    //Exit Sleep
 delay(120);
 LCD_CMD(ILI9341_DISPON);    //Display on
 digitalWrite(LCD_CS, HIGH);
}
//**************************************************************************************
************************************************
// Función para enviar comandos a la LCD - parámetro (comando)
//**************************************************************************************
************************************************
```

```
void LCD_CMD(uint8_t cmd) {
  digitalWrite(LCD_RS, LOW);
  digitalWrite(LCD_WR, LOW);
  GPIO_PORTB_DATA_R = cmd;
  digitalWrite(LCD_WR, HIGH);
}
//**************************************************************************************************
// Función para enviar datos a la LCD - parámetro (dato)
//**************************************************************************************************
void LCD_DATA(uint8_t data) {
  digitalWrite(LCD_RS, HIGH);
  digitalWrite(LCD_WR, LOW);
  GPIO_PORTB_DATA_R = data;
  digitalWrite(LCD_WR, HIGH);
}
//**************************************************************************************************
// Función para definir rango de direcciones de memoria con las cuales se trabajara (se define una ventana)
//**************************************************************************************************
void SetWindows(unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2) {
  LCD_CMD(0x2a); // Set_column_address 4 parameters
  LCD_DATA(x1 >> 8);
  LCD_DATA(x1);
  LCD_DATA(x2 >> 8);
  LCD_DATA(x2);
  LCD_CMD(0x2b); // Set_page_address 4 parameters
```

```
  LCD_DATA(y1 >> 8);
  LCD_DATA(y1);
  LCD_DATA(y2 >> 8);
  LCD_DATA(y2);
  LCD_CMD(0x2c); // Write_memory_start
}
//*************************************************************************************
************************************************
// Función para borrar la pantalla - parámetros (color)
//*************************************************************************************
************************************************
void LCD_Clear(unsigned int c) {
  unsigned int x, y;
  LCD_CMD(0x02c); // write_memory_start
  digitalWrite(LCD_RS, HIGH);
  digitalWrite(LCD_CS, LOW);
  SetWindows(0, 0, 319, 239); // 479, 319);
  for (x = 0; x < 320; x++)
    for (y = 0; y < 240; y++) {
      LCD_DATA(c >> 8);
      LCD_DATA(c);
    }
  digitalWrite(LCD_CS, HIGH);
}
//*************************************************************************************
************************************************
// Función para dibujar una línea horizontal - parámetros ( coordenada x, cordenada y, longitud, color)
//*************************************************************************************
************************************************
void H_line(unsigned int x, unsigned int y, unsigned int l, unsigned int c) {
```

```
 unsigned int i, j;
 LCD_CMD(0x02c); //write_memory_start
 digitalWrite(LCD_RS, HIGH);
 digitalWrite(LCD_CS, LOW);
 l = l + x;
 SetWindows(x, y, l, y);
 j = l;// * 2;
 for (i = 0; i < l; i++) {
  LCD_DATA(c >> 8);
  LCD_DATA(c);
 }
 digitalWrite(LCD_CS, HIGH);
}
//*************************************************************************************
*************************************************
// Función para dibujar una línea vertical - parámetros ( coordenada x, cordenada y, longitud, color)
//*************************************************************************************
*************************************************
void V_line(unsigned int x, unsigned int y, unsigned int l, unsigned int c) {
 unsigned int i, j;
 LCD_CMD(0x02c); //write_memory_start
 digitalWrite(LCD_RS, HIGH);
 digitalWrite(LCD_CS, LOW);
 l = l + y;
 SetWindows(x, y, x, l);
 j = l; //* 2;
 for (i = 1; i <= j; i++) {
  LCD_DATA(c >> 8);
  LCD_DATA(c);
```

```
  }
  digitalWrite(LCD_CS, HIGH);
}
//*************************************************************************************************
// Función para dibujar un rectángulo - parámetros ( coordenada x, cordenada y, ancho, alto, color)
//*************************************************************************************************
void Rect(unsigned int x, unsigned int y, unsigned int w, unsigned int h, unsigned int c) {
  H_line(x  , y  , w, c);
  H_line(x  , y + h, w, c);
  V_line(x  , y  , h, c);
  V_line(x + w, y  , h, c);
}
//*************************************************************************************************
// Función para dibujar un rectángulo relleno - parámetros ( coordenada x, cordenada y, ancho, alto, color)
//*************************************************************************************************
void FillRect(unsigned int x, unsigned int y, unsigned int w, unsigned int h, unsigned int c) {
  unsigned int i;
  for (i = 0; i < h; i++) {
    H_line(x  , y  , w, c);
    H_line(x  , y + i, w, c);
  }
}
//*************************************************************************************************
// Función para dibujar texto - parámetros ( texto, coordenada x, cordenada y, color, background)
```

```
//*****************************************************************************************
************************************************
void LCD_Print(String text, int x, int y, int fontSize, int color, int background) {
 int fontXSize ;
 int fontYSize ;

 if (fontSize == 1) {
  fontXSize = fontXSizeSmal ;
  fontYSize = fontYSizeSmal ;
 }
 if (fontSize == 2) {
  fontXSize = fontXSizeBig ;
  fontYSize = fontYSizeBig ;
 }

 char charInput ;
 int cLength = text.length();
 Serial.println(cLength, DEC);
 int charDec ;
 int c ;
 int charHex ;
 char char_array[cLength + 1];
 text.toCharArray(char_array, cLength + 1) ;
 for (int i = 0; i < cLength ; i++) {
  charInput = char_array[i];
  Serial.println(char_array[i]);
  charDec = int(charInput);
  digitalWrite(LCD_CS, LOW);
  SetWindows(x + (i * fontXSize), y, x + (i * fontXSize) + fontXSize - 1, y + fontYSize );
```

```
   long charHex1 ;
   for ( int n = 0 ; n < fontYSize ; n++ ) {
    if (fontSize == 1) {
      charHex1 = pgm_read_word_near(smallFont + ((charDec - 32) * fontYSize) + n);
    }
    if (fontSize == 2) {
      charHex1 = pgm_read_word_near(bigFont + ((charDec - 32) * fontYSize) + n);
    }
    for (int t = 1; t < fontXSize + 1 ; t++) {
      if (( charHex1 & (1 << (fontXSize - t))) > 0 ) {
        c = color ;
      } else {
        c = background ;
      }
      LCD_DATA(c >> 8);
      LCD_DATA(c);
    }
   }
   digitalWrite(LCD_CS, HIGH);
  }
}
//*************************************************************************************
************************************************
// Función para dibujar una imagen a partir de un arreglo de colores (Bitmap) Formato (Color 16bit R
5bits G 6bits B 5bits)
//*************************************************************************************
************************************************
void LCD_Bitmap(unsigned int x, unsigned int y, unsigned int width, unsigned int height, unsigned char
bitmap[]) {
  LCD_CMD(0x02c); // write_memory_start
```

```
 digitalWrite(LCD_RS, HIGH);
 digitalWrite(LCD_CS, LOW);

 unsigned int x2, y2;
 x2 = x + width;
 y2 = y + height;
 SetWindows(x, y, x2 - 1, y2 - 1);
 unsigned int k = 0;
 unsigned int i, j;

 for (int i = 0; i < width; i++) {
  for (int j = 0; j < height; j++) {
   LCD_DATA(bitmap[k]);
   LCD_DATA(bitmap[k + 1]);
   //LCD_DATA(bitmap[k]);
   k = k + 2;
  }
 }
 digitalWrite(LCD_CS, HIGH);
}



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

// Leer archivos de la SD y escribirlos x, y, base, altura, archivo txt []

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

void LCD_FONDO(unsigned int x, unsigned int y, unsigned int width, unsigned int height) {
 LCD_CMD(0x02c); // write_memory_start
```

```
 digitalWrite(LCD_RS, HIGH);
 digitalWrite(LCD_CS, LOW);

 unsigned int x2, y2;
 x2 = x + width;
 y2 = y + height;
 SetWindows(x, y, x2 - 1, y2 - 1);
 unsigned int k = 0;
 unsigned int i, j;

 char data[4];
 int dataindex = 0;
 char caracter;

 myFile = SD.open("PAN.txt");
 if (myFile) {
  // read from the file until there's nothing else in it:
  while (myFile.available()) {
   caracter = myFile.read();
   //Serial.println(caracter);
   if ((caracter != ',') && (caracter != ' ')) { //cuando el caracter que esta detectando es distinto de coma
o espacio
    data[dataindex] = caracter; //Se copian los datos en la posicion indicada (se concatenan)
    dataindex++; //Se incrementa la posición
    //        data.concat(caracter);
   }
   else {
    if (caracter == ',') { //al detectar una coma
     //Serial.println(data);
```

```
        uint8_t mandar_data = (uint8_t)strtol(data, NULL, 16); //convertir str a int
        //Serial.println(mandar_data);
        LCD_DATA(mandar_data); // se manda la data
        dataindex = 0; //se borran los datos que tenia la variable para poder copiar y mandar datos nuevos
       }
     } //end else
   } //end while
  } //end if my File
  // close the file:
  digitalWrite(LCD_CS, HIGH);
  myFile.close(); //end if myFile

} //End funcion LCD_Fondo

//*****************************************************************************************************
// Función para dibujar una imagen sprite - los parámetros columns = número de imagenes en el sprite, index = cual desplegar, flip = darle vuelta
//*****************************************************************************************************
void LCD_Sprite(int x, int y, int width, int height, unsigned char bitmap[], int columns, int index, char flip, char offset) {
  LCD_CMD(0x02c); // write_memory_start
  digitalWrite(LCD_RS, HIGH);
  digitalWrite(LCD_CS, LOW);

  unsigned int x2, y2;
  x2 =   x + width;
  y2 =    y + height;
  SetWindows(x, y, x2 - 1, y2 - 1);
```

```
  int k = 0;
  int ancho = ((width * columns));
  if (flip) {
    for (int j = 0; j < height; j++) {
      k = (j * (ancho) + index * width - 1 - offset) * 2;
      k = k + width * 2;
      for (int i = 0; i < width; i++) {
        LCD_DATA(bitmap[k]);
        LCD_DATA(bitmap[k + 1]);
        k = k - 2;
      }
    }
  } else {
    for (int j = 0; j < height; j++) {
      k = (j * (ancho) + index * width + 1 + offset) * 2;
      for (int i = 0; i < width; i++) {
        LCD_DATA(bitmap[k]);
        LCD_DATA(bitmap[k + 1]);
        k = k + 2;
      }
    }



  }
  digitalWrite(LCD_CS, HIGH);
}
```

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

// Función Interrupciones

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


void flag_d1s() {

  if (lastd1_s != d1_s) { //chequea el estado pasado. Si el estado pasado es el mismo que el que se esta
leyendo, no deja pasar

    d1_s = !d1_s;

    lastd1_s = d1_s;

    delay(50);

  }

}


void flag_d2s() {

  if (lastd2_s != d2_s) {

    d2_s = !d2_s;

    lastd2_s = d2_s;

    delay(50);

  }

}



//void flag_d1d_r(){

//  if (lastd1_d != d1_d){

//  d1_d = !d1_d;

//  lastd1_d = d1_d;

//  delay(50);

//  }
```

```
//}
//
//void flag_d2d_r(){
//  if (lastd2_d != d2_d){
//  d2_d = !d2_d;
//  lastd2_d = d2_d;
//  delay(50);
//  }
//  }
```