UNIVERSIDAD DEL VALLE DE GUATEMALA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA, MECATRÓNICA

Nombre: María Alejandra Rodríguez Roldán
Carné: 21620
Curso: Electrónica Digital 2
Sección: 10
Catedrático: Pablo Mazariegos y José Morales
Fecha: 31/10/2023

Proyecto 2 – SPI, TFT y SD

Con este proyecto se busca que el estudiante experimente con los microcontroladores TIVA C y ESP32 para realizar una aplicación utilizando la comunicación SPI y utilice el almacenamiento SD como la pantalla TFT.

El proyecto consiste en implementar un data logger de un sensor utilizando el almacenamiento SD además de tener una interfaz gráfica para desplegar los resultados de las mediciones en la pantalla. El sensor estará conectado al microcontrolador ESP32 el cual se tendrá que comunicarse con este y tendrá que enviar la información tanto al microcontrolador TIVA C como a la computadora mediante comunicación UART. Adicionalmente se tendrá dos botones en el microcontrolador TIVA C para seleccionar la tarea que se desea realizar. También se tendrá un indicador auditivo utilizando un buzzer pasivo.

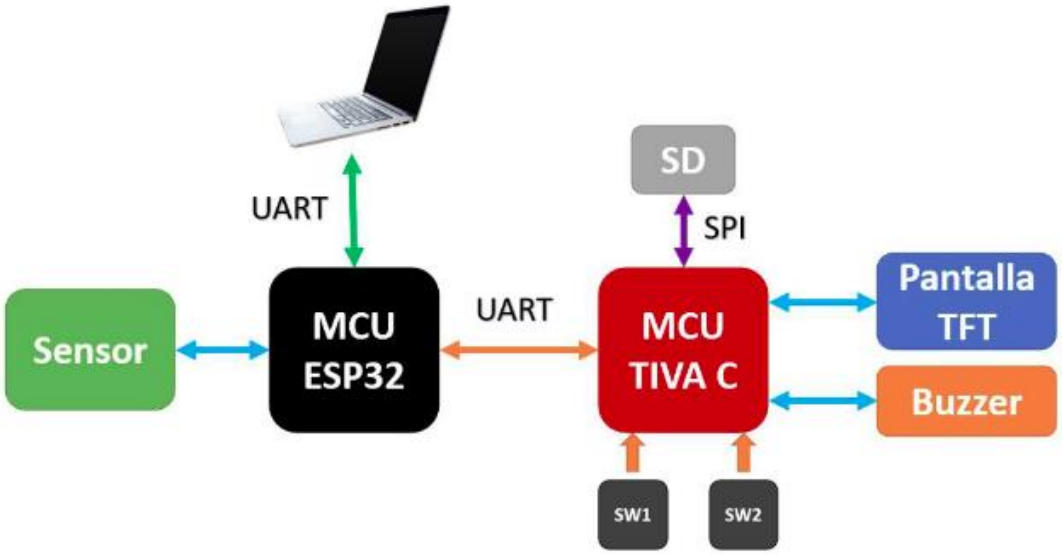

Figura 1. Diagrama general del proyecto

Link de GitHub: https://github.com/aler21620/Proyecto-2-Digital-2-21620
Link de vídeo de funcionamiento: https://www.youtube.com/watch?v=hwd57wLqspI

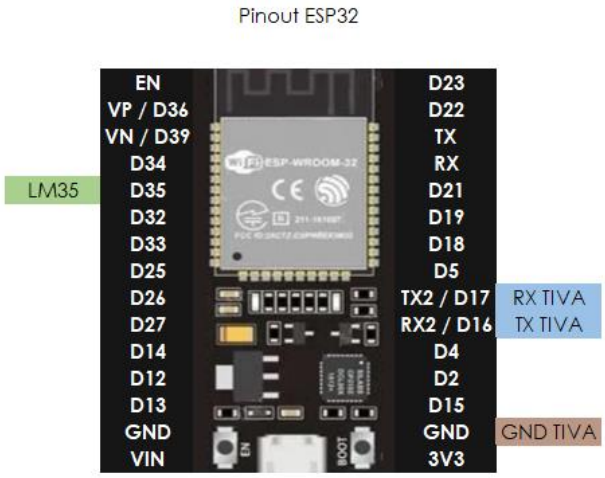

Figura 2. Pinout ESP32

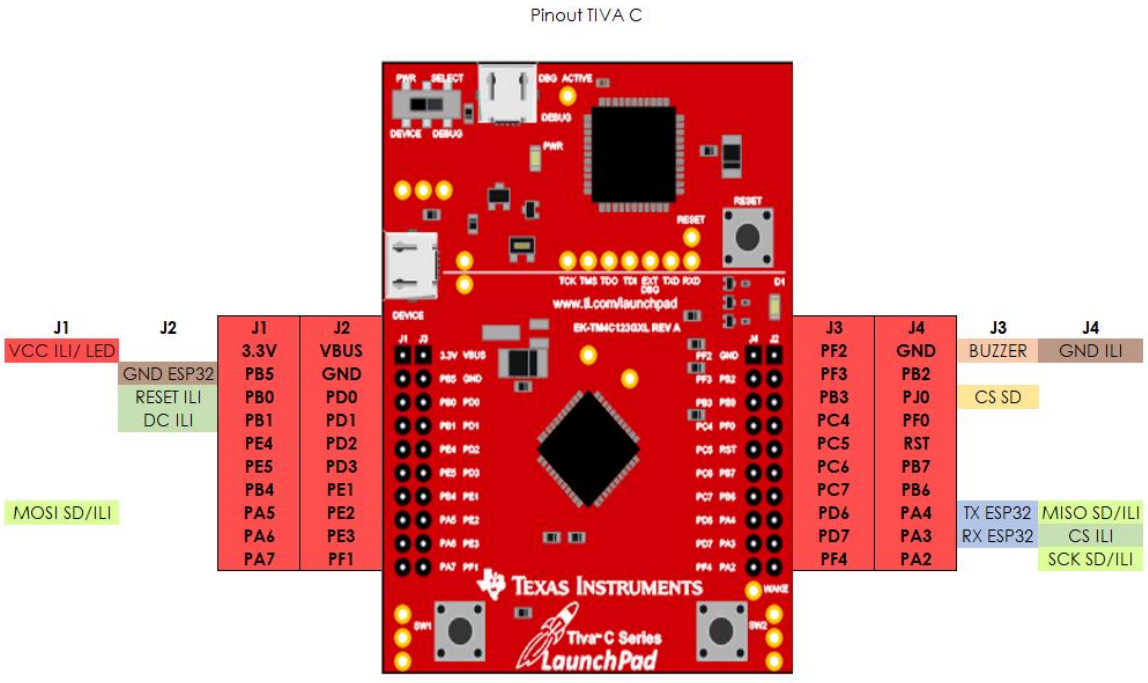

Figura 3. Pinout TIVA C

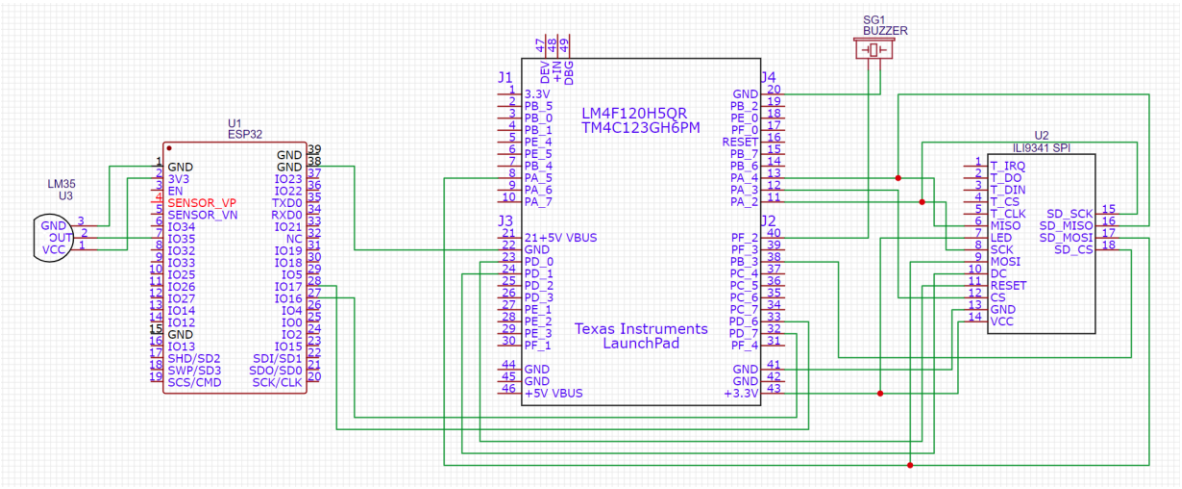

Figura 4. Esquemático del circuito utilizado para el proyecto

Pseudocódigos/Diagrama de flujo:

- **ESP32**:

1. Incluir librerías necesarias
2. Definir el pin en el que se conectará el LM35, sensor de temperatura y los pines de comunicación serial.
3. Definir los prototipos de funciones para poder tomar una lectura de temperatura con el ADC del microcontrolador.
4. Colocar las variables globales para almacenar los datos que obtener la temperatura en grados centígrados.
5. En el setup colocar:
    a. Iniciar la comunicación serial con el monitor serial de la computadora
    b. Iniciar la comunicación serial entre microcontroladores (ESP32 – TIVA C)
6. En el loop colocar:
    a. Recibir los datos que envía TIVA C por medio de la comunicación serial, por lo que se debe verificar si hay datos para leer.
    b. Si hay datos verificar que el dato enviado desde TIVA C sea la señal correcta
        i. Si es la señal correcta, llamar a la función que lee la temperatura
        ii. Guardar la lectura en una nueva variable
        iii. Imprimir el valor de lectura en el serial 2
        iv. Imprimir en el monitor serial del ESP32 el dato que se está enviando
        v. Colocar la variable que verifica la señal enviada desde TIVA C en 0 nuevamente
7. Colocar las funciones declaradas como prototipos de función que se utilizan dentro del proyecto
    a. Función para leer el ADC del ESP32
    b. Función para calcular la temperatura, utilizando analog Read y la función del ADC para obtener el dato en grados centígrados.

- **TIVA C:**

1. Incluir las librerías necesarias
2. Definir los pines de los 2 botones, incluidos en la TIVA C, así como los pines de comunicación serial.
3. Definir los pines de la pantalla TFT ILI9341, los pines de la memoria SD y el pin del buzzer para poder colocar un indicador auditivo.
4. Se colocan los prototipos de función que se utilizan para guardar los datos en la memoria SD y la codificación de la pantalla ILI SPI.
5. Se declaran las variables globales, que se utilizan para guardar en la SD y obtener los datos de temperatura desde el ESP32, también se declaran los límites de temperatura para definir si es baja, ambiente o alta.
6. En el setup se coloca:
    a. Se indica que se trabaja con el módulo 0 de SPI
    b. Se inicializa el monitor serial de la TIVA
    c. Se inicializa el monitor de la comunicación serial entre ambos microcontroladores
    d. Se configuran los botones como entradas
    e. El buzzer se configura como salida
    f. Se inicializa la comunicación con la SD
    g. Se inicializa la pantalla SPI
    h. Se configura el diseño de la pantalla SPI
7. En el loop se coloca:
    a. El condicional del primer botón, para definir que sí realiza al presionarse
        i. Primero manda una bandera al ESP32
        ii. Luego lee los datos enviados por comunicación serial del ESP32 y los imprime en el monitor serial
        iii. Se convierte la variable float enviada por el ESP a un String
        iv. Este String se imprime en la pantalla SPI
        v. Se coloca el condicional de los límites de temperatura
        vi. Se coloca la frecuencia y tono que emitirá el buzzer
    b. El condicional del segundo botón, para definir que sí realiza al presionarse
        i. Llama a la función para guardar los datos en la SD
        ii. Se coloca la frecuencia y tono que emitirá el buzzer
8. Colocar las funciones declaradas como prototipos de función que se utilizan dentro del proyecto
    a. Función para guardar datos en el archivo de la SD
    b. Funciones varias para el funcionamiento de la pantalla SPI

Deberá implementar librerías en sus códigos.

Parte 1 ESP32:

Diseñe e implemente una rutina en donde se comunique con un sensor de su preferencia utilizando el microcontrolador ESP32.

```
//**************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 2 - María Alejandra Rodríguez
// Sensor de temperatura - Comunicación con TIVA C y Pantalla SPI
//**************************************************************************
//**************************************************************************
// Librerías
//**************************************************************************
#include <Arduino.h>
#include "esp_adc_cal.h"

//**************************************************************************
//  Definición de pines
//**************************************************************************
#define SensorTemp 35 //Sensor del proyecto

//**************************************************************************
// Prototipos de función
//**************************************************************************
uint32_t readADC_Cal(int ADC_Raw);
void temperatura(void);

//**************************************************************************
// Variables Globales
//**************************************************************************
int Sensor_Raw = 0;
float voltaje =0.0;
float Sensor1 = 0.0;

//**************************************************************************
// Funciones
//**************************************************************************
uint32_t readADC_Cal(int ADC_Raw) {
  esp_adc_cal_characteristics_t adc_chars;
  esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12,
1100, &adc_chars);
  return (esp_adc_cal_raw_to_voltage(ADC_Raw, &adc_chars));
}
```

```
void temperatura(void) {
// Leer el pin LM35_Sensor1 ADC
  Sensor_Raw = analogRead(SensorTemp);
  // Calibrar ADC y tomar el voltaje en mV
  voltaje = readADC_Cal(Sensor_Raw);
  Sensor1 = ((voltaje/4095)*3.25)/0.01; // De ser necesario se multiplica
por un factor para que lea correctamente la temperatura
}
```

En el código mostrado anteriormente se observa el uso de 2 librerías, para codificar con Arduino y la utilización del ADC del ESP32. Luego se muestra la definición de los pines, donde se conecta el sensor de temperatura siendo en el pin 35 del microcontrolador. Además, se definen 2 prototipos de función, los cuales permiten hacer la lectura del sensor, haciendo uso del ADC del ESP32 con la función uint32_t readADC_Cal(int ADC_Raw) y la función de void temperatura(void), que permite obtener el dato crudo de la temperatura, convertirlo a voltaje y luego aplicar una ecuación que devuelve la temperatura en grados centígrados.

Parte 2 Comunicación UART:

Diseñe e implemente una rutina para un microcontrolador TIVA C en donde se pueda utilizar uno de los botones para preguntar el valor del sensor al microcontrolador del ESP32. Además, cuando se le pida el resultado del sensor lo deberá mandar tanto como a la TIVA C como a la computadora utilizando la comunicación UART.

Código en ESP32:

```
//**************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 2 - María Alejandra Rodríguez
// Sensor de temperatura - Comunicación con TIVA C y Pantalla SPI
//**************************************************************************
//**************************************************************************
// Librerías
//**************************************************************************
#include <Arduino.h>
#include "esp_adc_cal.h"

//**************************************************************************
//  Definición de pines
//**************************************************************************
#define RX_2 16 //Para comunicación serial con TIVA
#define TX_2 17 //Para comunicación serial con TIVA

//**************************************************************************
// Variables Globales
```

```
//*****************************************************************************
float temp;
int senal;

//*****************************************************************************
// Configuración
//*****************************************************************************
void setup() {
  //Comunicación UART0 con la computadora Serial (0)
  Serial.begin(115200);
  Serial.println("Se configuró Serial 0");
  Serial2.begin(115200, SERIAL_8N1, RX_2, TX_2); //Establecer comunicación
serial con TIVA
}

//*****************************************************************************
// Loop
//*****************************************************************************
void loop() {
  // Recibir datos de la TIVA C para colocar en la LCD
  if (Serial2.available()) {
    senal = Serial2.read();
  }

  if(senal == '1') {
    temperatura();
    temp = Sensor1;
    Serial2.println(temp);
    Serial.print("Dato enviado a TIVA C: ");
    Serial.print(temp);
    Serial.print("°C 🌡 \n");
    senal = 0;
  }
}
```

Esta parte de código es la encargada de realizar la comunicación serial con el microcontrolador TIVA C. Esto se configura en el setup, agregando un serial que será el que comunique la información entre el ESP y la TIVA. Además, en el loop se agrega la variable bandera, la cual es la que permite revisar si desde la TIVA se envía una bandera cuando se presiona el botón. Luego esta bandera se coloca en 0 para poder presionar nuevamente el botón en TIVA C y pedir un nuevo dato.

Código en TIVA C:

```
//**************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 2 - María Alejandra Rodríguez
//Sensor de temperatura - Comunicación con TIVA C y Pantalla SPI
//**************************************************************************
//**************************************************************************
// Librerías
//**************************************************************************
#include <stdint.h>

//**************************************************************************
//  Definición de pines
//**************************************************************************
//Pines botones
#define boton1 PUSH1 //Definición del botón para la varibale de contador
//Pines comunicación serial
#define RX_2 PD6 //Para comunicación serial con ESP32
#define TX_2 PD7 //Para comunicación serial con ESP32

//**************************************************************************
// Variables Globales
//**************************************************************************
float temp; //Para almacenar el valor de temperatura del sensor del ESP32

//**************************************************************************
// Configuración
//**************************************************************************
void setup() {
  Serial.begin(115200); //Velocidad del monitor serial
  Serial.println("Se configuró Serial 0");

  //Comunicación UART2 con el ESP32, Serial (2)
  Serial2.begin(115200); //Velocidad de la comunicación

  pinMode(boton1, INPUT_PULLUP); //Configuración del botón como entrada
}

//**************************************************************************
// Loop
//**************************************************************************
void loop() {
  //Boton para leer el dato del sensor
```

```
  int data = digitalRead(boton1);
  //Condiciones para sumar o restar con los botones en la variable contador
  if (data == LOW) {
    //Envío de un entero a ESP32 para que el microcontrolador sepa que debe
enviar la última lectura
    Serial2.println('1');
    if(Serial2.available() > 0) {
      temp = Serial2.parseFloat();
      Serial.print("🌡Tu temperatura actual es: ");
      Serial.print(temp);
      Serial.print("  °C 🌡\n");
    }
}
```

En la parte de código observada, que corresponde a TIVA C se muestra solamente la parte de configuración de la comunicación serial con el ESP32. Está comunicación se realiza con el serial 2, donde al presionar el botón se imprime en el monitor de la comunicación para funcionar como una variable bandera en el código del ESP32 y activar la función que lee la temperatura en el sensor LM35 con el ADC, en lugar de utilizar la analogread().

Parte 3 Despliegue de datos:
Diseñe e implemente una rutina para que en la pantalla TFT se pueda desplegar el resultado de la medición del sensor Sea creativo con la interfaz que deberá implementar.

Esta rutina se implementó en el código de TIVA C, utilizando una pantalla TFT ILI9341.

```
//*****************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 2 - María Alejandra Rodríguez
//Sensor de temperatura - Comunicación con TIVA C y Pantalla SPI
//*****************************************************************************
//*****************************************************************************
// Librerías
//*****************************************************************************
#include <SPI.h>
#include <SD.h>
#include "pitches.h"

#include <stdbool.h>
#include <TM4C123GH6PM.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"

//**************************************************************************
//  Definición de pines
//**************************************************************************
//Pines pantalla
#define LCD_RST PD_0 //Definición de pin RESET pantalla SPI
#define LCD_DC PD_1 //Definición de pin DC pantalla SPI
#define LCD_CS PA_3 //Definición de pin CS pantalla SPI
// El SPI es el 0
//MOSI va a PA_5
//MISO va a PA_4
//SCK va a PA_2

//**************************************************************************
// Prototipos de función
//**************************************************************************
//Prototipos de función que puedo utilizar con la pantalla SPI
void LCD_Init(void);
void LCD_Clear(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[]);

//**************************************************************************
// Configuración
//**************************************************************************
void setup() {
  SPI.setModule(0);
  Serial.begin(115200); //Velocidad del monitor serial

  //Inicialización pantalla SPI
```

```
  SysCtlClockSet(SYSCTL_SYSDIV_2_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTA
L_16MHZ);

  Serial.println("Inicio");
  LCD_Init();
  LCD_Clear(0x00);

  FillRect(0,0, 320, 240,  0x37FC);
  FillRect(0, 60, 320, 220, 0xF7BD);
  FillRect(0,180, 320, 220,  0x37FC);
  String text1 = "TEMPERATURA";
  LCD_Print(text1, 70, 8, 2, 0x1105, 0x37FC);
  String text2 = "ACTUAL";
  LCD_Print(text2, 115, 30, 2, 0x1105, 0x37FC);
  String text3 = "ALEJANDRA";
  LCD_Print(text3, 90, 190, 2, 0x1105, 0x37FC);
  String text4 = "RODRIGUEZ";
  LCD_Print(text4, 90, 212, 2, 0x1105, 0x37FC);
  LCD_Bitmap(35, 70, 35, 80, termometro);
  LCD_Bitmap(250, 70, 35, 80, termometro);
}

//*****************************************************************************
// Loop
//*****************************************************************************
void loop() {
  //Boton para leer el dato del sensor
  int data = digitalRead(boton1);
  //Condiciones para sumar o restar con los botones en la variable contador
  if (data == LOW) {
    //Envío de un entero a ESP32 para que el microcontrolador sepa que debe
enviar la última lectura
    Serial2.println('1');
    if(Serial2.available() > 0) {
      temp = Serial2.parseFloat();
      Serial.print("🌡Tu temperatura actual es: ");
      Serial.print(temp);
      Serial.print("  °C 🌡\n");
    }

    int temperatura = temp * 100;
    //Se obtiene cada número por separado
    int unidad = (temperatura/1) %10;
    int decena = (temperatura/10) %10;
```

```
    int decimal = (temperatura/100) %10;
    int centena = (temperatura/1000) %10;

    String uni = String(unidad);
    String dec = String(decena);
    String deci = String(decimal);
    String cent = String(centena);

    String tempe = cent + deci + "." + dec + uni;
    LCD_Print(tempe, 120, 100, 2, 0x1105, 0xF7BD);

    if(temp < TEMP_LOW) {
      String baja = "  BAJA  ";
      LCD_Print(baja, 100, 130, 2, 0x1105, 0xF7BD);
    } else if (temp >= TEMP_LOW && temp < TEMP_MEDIUM) {
      String ambiente = "AMBIENTE";
      LCD_Print(ambiente, 100, 130, 2, 0x1105, 0xF7BD);
    } else if (temp >= TEMP_MEDIUM && temp <= TEMP_HIGH) {
      String alta = "  ALTA  ";
      LCD_Print(alta, 100, 130, 2, 0x1105, 0xF7BD);
    }
}

//*****************************************************************************
// Función para inicializar LCD
//*****************************************************************************
void LCD_Init(void) {
  pinMode(LCD_RST, OUTPUT);
  pinMode(LCD_CS, OUTPUT);
  pinMode(LCD_DC, OUTPUT);
  //****************************************
  // Secuencia de Inicialización
  //****************************************
  digitalWrite(LCD_CS, HIGH);
  digitalWrite(LCD_DC, 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_DC, LOW);
  SPI.transfer(cmd);
```

```
}
//*************************************************************************
// Función para enviar datos a la LCD - parámetro (dato)
//*************************************************************************
void LCD_DATA(uint8_t data) {
  digitalWrite(LCD_DC, HIGH);
  SPI.transfer(data);
}
//*************************************************************************
// 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_DC, 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 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 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_DC, 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_DC, 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 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_DC, 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);
}
//*************************************************************************
// 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_DC, 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);
}
```

Parte 4 Almacenamiento de la SD:
Diseñe e implemente la comunicación con la tarjeta SD para que se pueda almacenar el dato obtenido utilizando el segundo botón.

La implementación de la tarjeta SD también se coloca en el código de TIVA C.

```
//*************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 2 - María Alejandra Rodríguez
//Sensor de temperatura - Comunicación con TIVA C y Pantalla SPI
//*************************************************************************
//*************************************************************************
// Librerías
//*************************************************************************
#include <SPI.h>
#include <SD.h>

//*************************************************************************
//  Definición de pines
//*************************************************************************
//Pines SD
#define SCK A2
#define MOSI A5
#define MISO A4
#define CS 38

//*************************************************************************
// Prototipos de función
//*************************************************************************
//Prototipos de función SD
void guardar(String);
```

```
//*****************************************************************************
// Variables Globales
//*****************************************************************************
String nombre; //Nombre del archivo que abre o crea
float temp; //Para almacenar el valor de temperatura del sensor del ESP32

//*****************************************************************************
// Configuración
//*****************************************************************************
void setup() {
  SPI.setModule(0);
  Serial.begin(115200); //Velocidad del monitor serial

  // Inicializa la comunicación con la tarjeta SD
  if (!SD.begin(CS)) {
    //Indica que algo pasó y no se inicializó correctamente
    Serial.println("No se pudo inicializar la tarjeta SD.");
    return;
  }
  //Indica que se inicializó correctamente
  Serial.println("Tarjeta SD inicializada correctamente.");
}

//*****************************************************************************
// Loop
//*****************************************************************************
void loop() {
  //Boton para guardar el dato del sensor en la SD
  if (digitalRead(boton2) == LOW) {
    guardar("Sensor.txt");
    delay(250);
  }
  delay(100);
}

//*****************************************************************************
// Funciones
//*****************************************************************************
//Función para guardar el dato en la memoria SD
void guardar(String nombre) {
  File archivo = SD.open("Sensor.txt", FILE_WRITE);

  if (archivo) {
    archivo.print("🌡Tu temperatura actual es: ");
```

```
    archivo.print(temp);
    archivo.print("  °C 🌡");
    archivo.println();
    archivo.close();
    Serial.println("Datos de temperatura registrados correctamente en la
SD");
  } else {
    Serial.println("No se pudo abrir el archivo para guardar datos.");
  }
}
```

En esta sección de código se puede observar la implementación de la tarjeta SD. Esta tarjeta la se coloca en la pantalla TFT y por medio del CS y comunicación SPI se leen los datos de la tarjeta y permite guardar nuevos datos, en este caso del sensor de temperatura. Además, se observa que se creó una función, la cual permite guardar el valor del sensor enviado por medio de comunicación serial en un archivo llamado Sensor.txt. Asimismo, de no poder abrir el archivo, en el monitor serial se mostrará que no pudo realizarlo, por lo que se debe revisar que el archivo este creado correctamente en la SD y que la codificación sea correcta al igual que el cableado.

Parte 5 Indicador Auditivo:
Implemente un indicador auditivo utilizando un buzzer pasivo para indicar que se realizó una medición, otro sonido para saber que se almacenó el dato en la memoria SD.

```
//*************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 2 - María Alejandra Rodríguez
//Sensor de temperatura - Comunicación con TIVA C y Pantalla SPI
//*************************************************************************
//*************************************************************************
// Librerías
//*************************************************************************
#include "pitches.h"

//*************************************************************************
//  Definición de pines
//*************************************************************************
//Pines botones
#define boton1 PUSH1 //Definición del botón para la varibale de contador
#define boton2 PUSH2 //Definición del botón 2 para la variable de contador
//Pin buzzer
#define buzz 40 //Definición del buzzer
```

```
//*****************************************************************************
// Variables Globales
//*****************************************************************************
float temp; //Para almacenar el valor de temperatura del sensor del ESP32
const float TEMP_LOW = 24.0;
const float TEMP_MEDIUM = 25.0;
const float TEMP_HIGH = 26.0;

//*****************************************************************************
// Configuración
//*****************************************************************************
void setup() {
  Serial.begin(115200); //Velocidad del monitor serial
  Serial.println("Se configuró Serial 0");

  pinMode(boton1, INPUT_PULLUP); //Configuración del botón como entrada
  pinMode(boton2, INPUT_PULLUP); //Configuración del botón como entrada
  pinMode(buzz, OUTPUT);
}

//*****************************************************************************
// Loop
//*****************************************************************************
void loop() {
  //Boton para leer el dato del sensor
  int data = digitalRead(boton1);
  //Condiciones para sumar o restar con los botones en la variable contador
  if (data == LOW) {
    tone(buzz, 349);
    delay(500);
    noTone(buzz);
    tone(buzz, 523);
    delay(500);
    noTone(buzz);
    tone(buzz, 349);
    delay(500);
    noTone(buzz);
    tone(buzz, 330);
    delay(500);
    noTone(buzz);
  }

  if (digitalRead(boton2) == LOW) {
    tone(buzz, 494);
```

```
    delay(500);
    noTone(buzz);
    tone(buzz, 494);
    delay(500);
    noTone(buzz);
    tone(buzz, 440);
    delay(500);
    noTone(buzz);
    tone(buzz, 392);
    delay(500);
    noTone(buzz);
    tone(buzz, 440);
    delay(500);
    noTone(buzz);
    tone(buzz, 494);
    delay(500);
    noTone(buzz);
    tone(buzz, 392);
    delay(500);
    noTone(buzz);
  }
  delay(100);
}
```

En este código se observa cómo se configuró el buzzer para emitir diferentes sonidos dependiendo del botón que se presione. Además, esto ayuda a dar un indicador auditivo de que se está realizando la función correcta. Para ello se utilizaron diferentes frecuencias para crear pequeñas melodías.