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: 13/11/2023

Proyecto 3 – I2C y Neopixel

Con este proyecto se busca que el estudiante experimente con los microcontroladores TIVA C y ESP32 para realizar una aplicación utilizando la comunicación I2C y el led RGB NeoPixel.

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 utilizando el protocolo de comunicación I2C 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 visual utilizando un led RGB NeoPixel. Deberá implementar librerías en sus códigos.

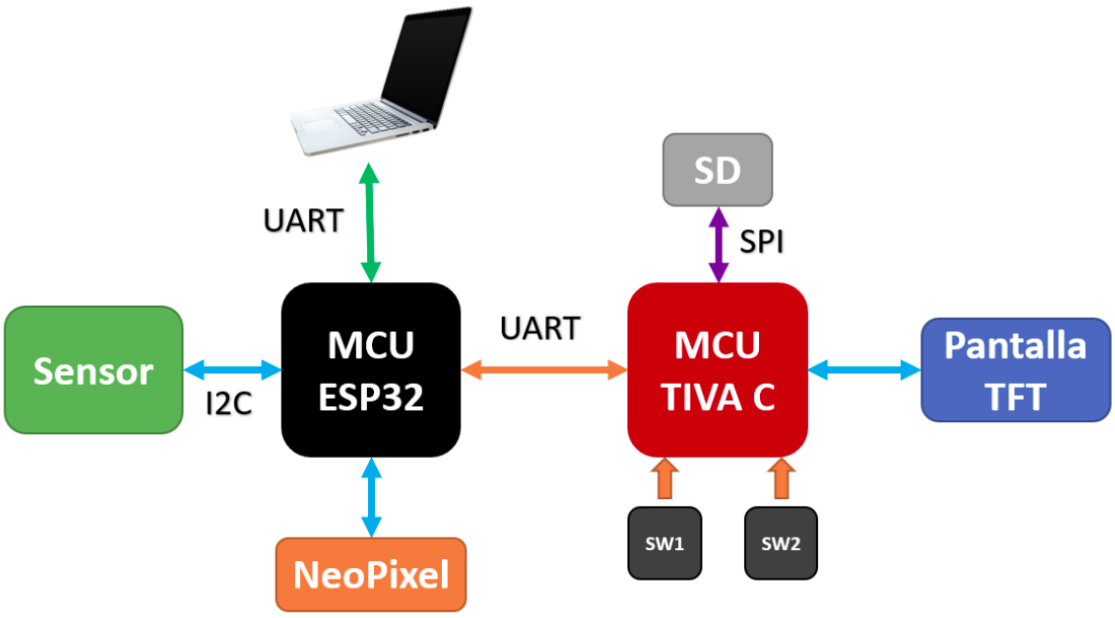

Figura 1. Diagrama general del proyecto

Link de GitHub: https://github.com/aler21620/Proyecto-3-Digital-2-21620
Link de vídeo de funcionamiento: https://www.youtube.com/watch?v=HLw4pjdfavI

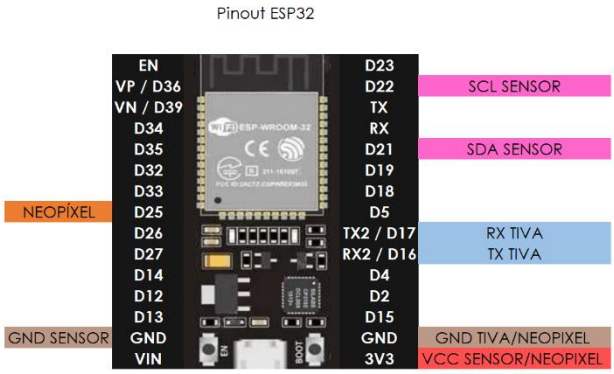

Figura 2. Pinout ESP32

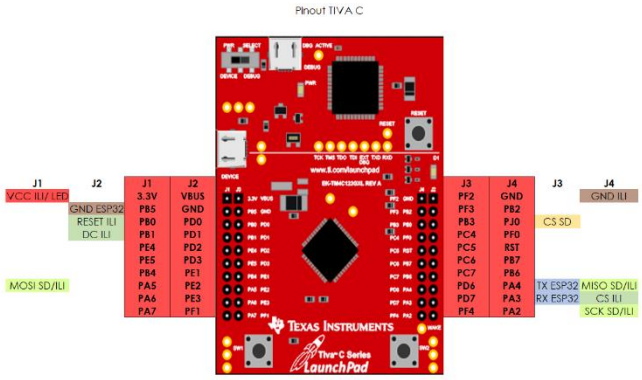

Figura 3. Pinout TIVA C

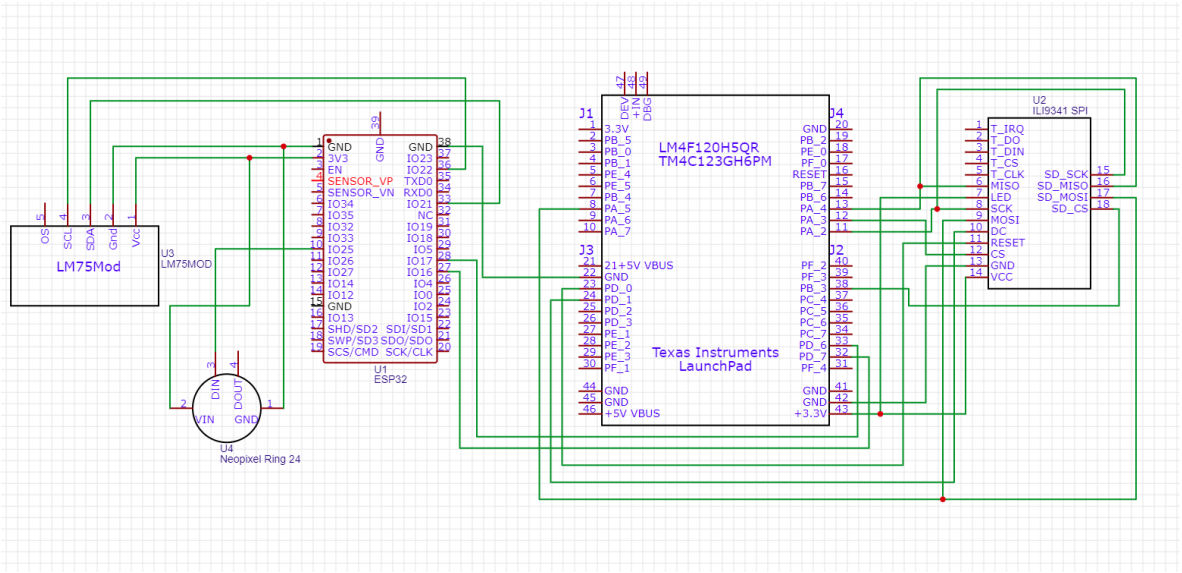

Figura 4. Esquemático del circuito utilizado para el proyecto

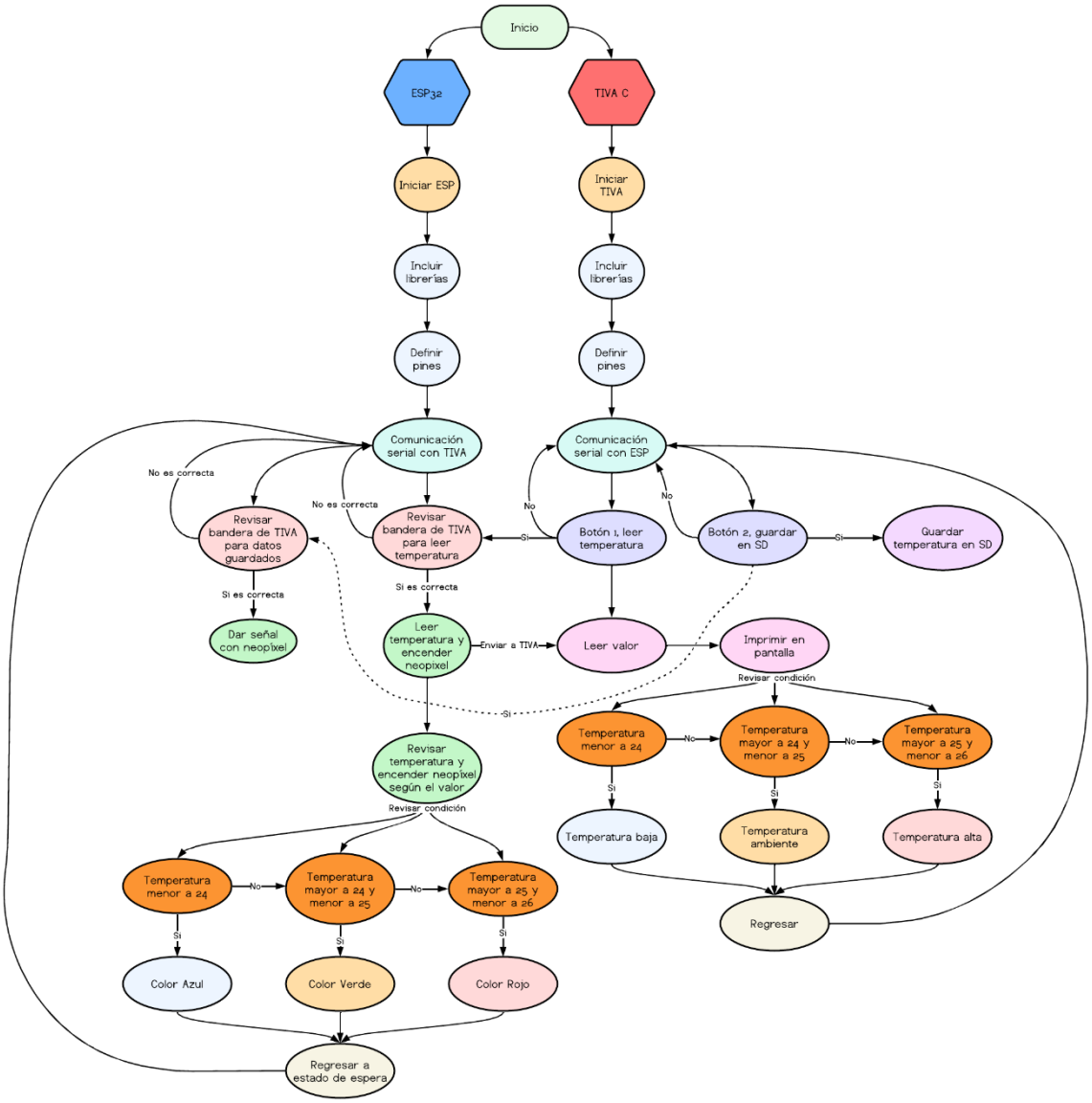

Figura 5. Diagrama de flujo general del proyecto

Pseudocódigos/Diagrama de flujo:

- **ESP32**:

1. Incluir librerías necesarias
2. Definir el pin en el que se conectará el LM75, sensor de temperatura con comunicación I2C, los pines de comunicación serial y del neopíxel.
3. Definir los prototipos de funciones para poder tomar una lectura de temperatura utilizando las librerías del sensor y la comunicación I2C.
4. Colocar las variables globales para almacenar los datos para obtención de temperatura y la señal de la comunicación serial.
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)
    c. Iniciar el neopixel
6. En el loop colocar:
    a. Colocar estado de espera a instrucciones en el neopíxel
    b. Recibir los datos que envía TIVA C por medio de la comunicación serial, por lo que se debe verificar si hay datos para leer.
    c. 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. Dar señal en el neopíxel de que está enviando la temperatura
        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. Verificar el valor de temperatura con los límites establecidos y dependiendo de su valor, colocar un color en el neopíxel
        vi. Colocar la variable que verifica la señal enviada desde TIVA C en 0 nuevamente
    d. Si el dato verificado en el monitor serial es la segunda bandera
        i. Si la señal recibida es correcta, imprimir en el monitor serial del ESP32 que recibió que se guardaron los datos
        ii. Colocar color en el neopíxel para indicar que se está guardando
        iii. 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 temperatura
    b. Función para encender los neopíxeles en estado de espera
    c. Función para apagar el neopíxel
    d. Función para colocar color cuando envía el dato a la TIVA
    e. Función para verificar el color a mostrar en el neopíxel dependiendo el valor de la temperatura
    f. Función para indicar que está guardando los datos en la SD

- **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 y los pines de la memoria SD
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. Además, colocar como obtener el fondo para utilizar la memoria Flash.
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. Se inicializa la comunicación con la SD
    f. Se inicializa la pantalla SPI
    g. Se configura el diseño de la pantalla SPI utilizando el almacenamiento en la memoria Flash
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. La variable de temperatura se vuelve a poner en cero, para no repetir datos
    b. El condicional del segundo botón, para definir que sí realiza al presionarse
        i. Primero envía una bandera al ESP32
        ii. Llama a la función para guardar los datos en la SD
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 utilizando el protocolo de comunicación I2C.

```
//*****************************************************************************
***
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 3 - María Alejandra Rodríguez
// Sensor de temperatura I2C - Comunicación con TIVA C y Pantalla SPI
//*****************************************************************************
***
//*****************************************************************************
***
// Librerías
//*****************************************************************************
***
#include <Arduino.h>
#include <Wire.h> //Para la comunicación I2C

//*****************************************************************************
//  Definición de pines
//*****************************************************************************
#define LM75_ADDRESS 0x48 // Dirección I2C del sensor LM75

//*****************************************************************************
// Prototipos de función
//*****************************************************************************
float readTemperature(); //Para leer temperatura con sensor I2C

//*****************************************************************************
// Variables Globales
//*****************************************************************************
float temp; //Para guardar la lectura de temperatura

//*****************************************************************************
// Funciones
//*****************************************************************************
//Función para leer la temperatura con el sensor I2C
float readTemperature() {
  Wire.beginTransmission(LM75_ADDRESS); //Leer la temperatura a través de
I2C y Wire
  Wire.write(0x00); // Registro de lectura de temperatura (0x00 para
lectura)
```

```
  Wire.endTransmission();

  Wire.requestFrom(LM75_ADDRESS, 2); // Se solicitan 2 bytes de datos de
temperatura
  int16_t tempData = (Wire.read() << 8) | Wire.read(); // Combinar los bytes
recibidos

  float temperature = tempData / 256.0; // Convertir datos a grados Celsius
  return temperature;
}
```

En el código mostrado anteriormente se observa el uso de la librería Wire.h para realizar la comunicación I2C y obtener la lectura de temperatura del sensor. Esta se realizó indicando la dirección del sensor siendo 0x48, luego se crea una función que devuelva la variable de temperatura. Esta función va a ser utilizada cuando el ESP32 reciba por comunicación serial el comando/señal para tomar una medición. Además, se hace uso de funciones de la librería Wire, así como inicia la transmisión de datos, escribir y leer los datos. Al mismo tiempo se aplica una función para obtener el cálculo de 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 ESP 32:

```
//*****************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 3 - María Alejandra Rodríguez
// Sensor de temperatura I2C - Comunicación con TIVA C y Pantalla SPI
//*****************************************************************************
//*****************************************************************************
// Librerías
//*****************************************************************************
#include <Arduino.h>

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

```
//**************************************************************************
// Variables Globales
//**************************************************************************
int senal; //Para leer y almacenar la señal que envía TIVA C

//**************************************************************************
// 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() {
  if (Serial2.available()) {
    senal = Serial2.read();
  }

  //Verificar si la señal es para leer temperatura
  if (senal == '1') {
    float temperature = readTemperature(); //Leer temperatura del sensor
    enviando(); //Estado de envío de datos en el neopíxel
    delay(3000);
    Serial2.println(temperature); //Enviar dato a TIVA
    Serial.print("Dato enviado a TIVA C: ");
    Serial.print(temperature);
    Serial.print("°C 🌡 \n");
    apagarTodos(); //Apagar el Neopíxel
    delay(500);
    color_TEMP(); //Encender Neopíxel según estado de la temperatura
    delay(3000);
    senal = 0; //Regresar la señal a 0, para esperar intrucciones de TIVA
  }

  //Verificar si la señal es para guardar datos
  if (senal == '2') {
    Serial.print("Señal recibida de TIVA C: ");
```

```
    Serial.print("Datos guardados en SD \n"); //Indicar que está guardando
los datos
    guardando(); //Encender el Neopíxel en el color que indica que los
guardó
    delay(3000);
    senal = 0; //Regresar la señal a 0, para esperar intrucciones de TIVA
  }
}
```

En esta parte del código siendo la colocada para el ESP 32, se declaran los pines de comunicación serial del UART 2, para en el setuo configurar la comunicación. Luego en el loop se coloca que mientras el serial 2 esté disponible, va a leer los datos que esté enviando. Por lo que luego se colocan 2 condicionales, que verifican si los datos que envía son iguales y sí si lo son, va a realizar una serie de instrucciones con el sensor o el neopíxel. ‘

Código TIVA C:

```
//*****************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 3 - María Alejandra Rodríguez
// Sensor de temperatura I2C - 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
#define boton2 PUSH2 //Definición del botón 2 para la variable 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() {
  //Iniciar comunicación serial con la TIVA C
  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

  //Configuración de los botones incluidos en la TIVA
  pinMode(boton1, INPUT_PULLUP); //Configuración del botón como entrada
  pinMode(boton2, 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) {
      //Delay para esperar a que termine de leer el sensor en el ESP y el
neopixel indique que lo envió
      delay(4000);
      //Leer el buffer del monitor serial y obtener los datos en tipo float
      temp = Serial2.parseFloat();
      //Imprimir la temperatura que recibe del ESP para verificar que sea la
misma
      Serial.print("🌡Tu temperatura actual es: ");
      Serial.print(temp);
      Serial.print("  °C 🌡\n");
      delay(500);
    }
  }

  //Instrucciones para el segundo botón, para guardar los datos
  if (digitalRead(boton2) == LOW) {
```

```
    //Envío de un entero a ESP32 para que sepa de que color poner el
neopíxel
    Serial2.println('2');
  }
  delay(100);
}
```

En este código se observa la configuración de la comunicación serial para TIVA C con ESP32. Aquí, primero se declaran los pines de los botones y de la comunicación serial del puerto 2. Luego en el setuo se habilita el monitor serial de la TIVA y el monitor Serial 2 para poder estar enviando y recibiendo datos. Esto permite que luego en el loop se envíen variables bandera al ESP 32, donde puede revisar cual es para poder realizar la acción correspondiente ya sea que este pidiendo el dato de temperatura o diciendo que está guardando los datos.

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.

```
//*****************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 3 - María Alejandra Rodríguez
// Sensor de temperatura I2C - Comunicación con TIVA C y Pantalla SPI
//*****************************************************************************
//*****************************************************************************
// Librerías
//*****************************************************************************
#include <SPI.h>
#include <SD.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 SD
void guardar(String);
//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[]);

//Función/Variable para llamarla desde el archivo de gráficos, de esta
manera se utiliza la memoria flash
extern uint8_t fondo[];

//*****************************************************************************
// Variables Globales
//*****************************************************************************
float temp; //Para almacenar el valor de temperatura del sensor del ESP32
//Los límites de temperatura pueden variar dependiendo de la aplicación o el
lugar donde se encuentre
const float TEMP_LOW = 25.0;
const float TEMP_MEDIUM = 27.0;
```

```
const float TEMP_HIGH = 30.0;

//*****************************************************************************
// Configuración
//*****************************************************************************
void setup() {
  //Iniciar Módulo SPI
  SPI.setModule(0);

  //Iniciar comunicación serial con la TIVA C
  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");
  //Inicia la pantalla SPI
  LCD_Init();
  LCD_Clear(0x00);

  //Colocar fondo del proyecto
  LCD_Bitmap(0, 0, 320, 240, fondo);
  String text1 = "ALEJANDRA RODRIGUEZ";
  LCD_Print(text1, 8, 210, 2, 0x1105, 0xD7FD);
}

//*****************************************************************************
// 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) {
      //Delay para esperar a que termine de leer el sensor en el ESP y el
neopixel indique que lo envió
      delay(4000);
      //Leer el buffer del monitor serial y obtener los datos en tipo float
      temp = Serial2.parseFloat();
```

```
      //Imprimir la temperatura que recibe del ESP para verificar que sea la
misma
      Serial.print("🌡Tu temperatura actual es: ");
      Serial.print(temp);
      Serial.print("  °C 🌡\n");
      delay(500);
    }

    //Separar el float de temperatura, para poder convertirlo en string
    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);

    //Hacer string de temperatura e imprimir en pantalla SPI
    String tempe = cent + deci + "." + dec + uni;
    LCD_Print(tempe, 55, 120, 2, 0x1105, 0xD7FD);

    //Evaluar los límites de temperatura
    if(temp < TEMP_LOW) {
      String limite = "  BAJA  ";
      LCD_Print(limite, 30, 160, 2, 0x1105, 0xD7FD);
    } else if (temp >= TEMP_LOW && temp < TEMP_MEDIUM) {
      String limite = "AMBIENTE";
      LCD_Print(limite, 30, 160, 2, 0x1105, 0xD7FD);
    } else if (temp >= TEMP_MEDIUM && temp <= TEMP_HIGH) {
      String limite = "  ALTA  ";
      LCD_Print(limite, 30, 160, 2, 0x1105, 0xD7FD);
    }
  }
}

//*******************************************************************************
// Funciones
//*******************************************************************************
// 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);
}
```

Este es el código utilizado en TIVA C para configurar la pantalla SPI ILI 934, esto se realiza con la librería vista en clase. Además, para el despliegue de datos se crea una interfaz donde se aclara que es un sensor de temperatura y aparece el dato que lee el sensor conectado al ESP32. Así mismo, aparece si la temperatura está catalogado como baja, ambiente o alta.

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.

```
//*************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 3 - María Alejandra Rodríguez
// Sensor de temperatura I2C - 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
//Los límites de temperatura pueden variar dependiendo de la aplicación o el
lugar donde se encuentre
const float TEMP_LOW = 25.0;
const float TEMP_MEDIUM = 27.0;
const float TEMP_HIGH = 30.0;

//*************************************************************************
// Configuración
//*************************************************************************
void setup() {
  //Iniciar Módulo SPI
  SPI.setModule(0);

  //Iniciar comunicación serial con la TIVA C
  Serial.begin(115200); //Velocidad del monitor serial
  Serial.println("Se configuró Serial 0");
```

```
  // 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() {
  //Instrucciones para el segundo botón, para guardar los datos
  if (digitalRead(boton2) == LOW) {
    //Envío de un entero a ESP32 para que sepa de que color poner el
neopíxel
    Serial2.println('2');
    //Guardar los datos en la SD
    guardar("I2C.txt");
    delay(250);
  }
  delay(100);
}

//**************************************************************************
// Funciones
//**************************************************************************
//Función para guardar el dato en la memoria SD
void guardar(String nombre) {
  File archivo = SD.open("I2C.txt", FILE_WRITE);

  if (archivo) {
    archivo.print("🌡Tu temperatura actual es: ");
    archivo.print(temp);
    archivo.print("  °C 🌡");
    archivo.println();
    //Límites de temperatura, para evaluar el valor y guardar en que estado
está la temperatura
    if(temp < TEMP_LOW) {
      String limite = "  Baja  ";
      archivo.print("Esta es temperatura: ");
```

```
      archivo.print(limite);
      archivo.println();
    } else if (temp >= TEMP_LOW && temp < TEMP_MEDIUM) {
      String limite = "Ambiente";
      archivo.print("Esta es temperatura: ");
      archivo.print(limite);
      archivo.println();
    } else if (temp >= TEMP_MEDIUM && temp <= TEMP_HIGH) {
      String limite = "  Alta  ";
      archivo.print("Esta es temperatura: ");
      archivo.print(limite);
      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.");
  }
}
```

Para el almacenamiento en la SD, primero se coloca el pin donde está conectado el CS (Chip Select) para que puede inicializar la SD en el Setup. Luego de haberla inicializado y mostrar en el monitor serial de TIVA C que no hubo problema durante la inicialización, en el loop se coloca que al presionar el botón 2, se guarden los datos. Este se hace por medio de una función declarada, donde verifica que esté un archivo .txt en la SD para luego abrirlo e imprimir los datos de temperatura y el estado de la temperatura.

Parte 5 Indicador Visual:

Implemente un indicador visual utilizando un led RGB NeoPixel para indicar con colores el estado del dispositivo (encendido, midiendo, guardando datos, etc).

```
//*****************************************************************************
// Universidad del Valle de Guatemala
// BE3015 - Electrónica Digital 2
// Proyecto 3 - María Alejandra Rodríguez
// Sensor de temperatura I2C - Comunicación con TIVA C y Pantalla SPI
//*****************************************************************************
//*****************************************************************************
// Librerías
//*****************************************************************************
#include <Arduino.h>
#include <Adafruit_NeoPixel.h> //Librería que permitirá codificar el
neopíxel
```

```
#ifdef __AVR__
  #include <avr/power.h>
#endif

//*********************************************************************
//  Definición de pines
//*********************************************************************
#define CIRCLE_PIN 25 // Para la conexión del Neopíxel
#define NUM_CIRCLE_LEDS 24 //Número de pines del Neopíxel
#define BRIGHT 50   // Brillo del Neopíxel

//Creación del objeto del Neopíxel para poder hacer diferentes diseños con
colores
Adafruit_NeoPixel circle = Adafruit_NeoPixel(NUM_CIRCLE_LEDS, CIRCLE_PIN,
NEO_GRB + NEO_KHZ800);

//*********************************************************************
// Prototipos de función
//*********************************************************************
void encenderTodos(void); //Para encender todos los leds en estado de espera
void apagarTodos(void); //Función para apagar el neopíxel
void enviando (void); //Para indicar que está enviando el dato leído de
temperatura
void color_TEMP(void); //Para indicar en que estado está la temperatura
void guardando(void); //Para indicar que está guardando los datos en la SD

//*********************************************************************
// Variables Globales
//*********************************************************************
const float TEMP_LOW = 25.0; //Valor mínimo de temperatura para considerarlo
en estado bajo
const float TEMP_MEDIUM = 27.0; //Valor medio de temperatura para
considerarlo en estado medio
const float TEMP_HIGH = 30.0; //Valor medio de temperatura para considerarlo
en estado alto

//*********************************************************************
// Configuración
//*********************************************************************
void setup() {
  // Comunicación UART0 con la computadora Serial (0)
  Serial.begin(115200);
  Wire.begin(21, 22); //Para el funcionamiento de I2C
```

```
  //Inicialización del Neopíxel
  circle.begin();
  circle.clear();
  circle.setBrightness(BRIGHT);
}

//**************************************************************************
// Loop
//**************************************************************************
void loop() {
  //Colocar el Neopíxel en estado de espera
  encenderTodos(); //Titila en color amarillo
  delay(500);
  apagarTodos();
  delay(500);
  // Recibir datos de la TIVA C
  if (Serial2.available()) {
    senal = Serial2.read();
  }

  //Verificar si la señal es para leer temperatura
  if (senal == '1') {
    float temperature = readTemperature(); //Leer temperatura del sensor
    enviando(); //Estado de envío de datos en el neopíxel
    delay(3000);
    Serial2.println(temperature); //Enviar dato a TIVA
    Serial.print("Dato enviado a TIVA C: ");
    Serial.print(temperature);
    Serial.print("°C 🌡 \n");
    apagarTodos(); //Apagar el Neopíxel
    delay(500);
    color_TEMP(); //Encender Neopíxel según estado de la temperatura
    delay(3000);
    senal = 0; //Regresar la señal a 0, para esperar intrucciones de TIVA
  }

  //Verificar si la señal es para guardar datos
  if (senal == '2') {
    Serial.print("Señal recibida de TIVA C: ");
    Serial.print("Datos guardados en SD \n"); //Indicar que está guardando
los datos
    guardando(); //Encender el Neopíxel en el color que indica que los
guardó
    delay(3000);
```

```
    senal = 0; //Regresar la señal a 0, para esperar intrucciones de TIVA
  }
}

//*********************************************************************
// Funciones
//*********************************************************************
//Función para encender todos los leds del Neopíxel para el estado de espera
void encenderTodos() {
  for (int i = 0; i < NUM_CIRCLE_LEDS; i++) {
    circle.setPixelColor(i, circle.Color(130, 130, 0)); // Establecer color
amarillo en el LED actual
  }
  circle.show(); // Mostrar los cambios en los LEDs
}

//Función para apagar todos los leds
void apagarTodos() {
  for (int i = 0; i < NUM_CIRCLE_LEDS; i++) {
    circle.setPixelColor(i, circle.Color(0, 0, 0)); // Apagar el LED actual
  }
  circle.show(); // Mostrar los cambios en los LEDs
}

//Función para encender todos los leds e indicar que está enviando el dato
void enviando () {
  for (int i = 0; i < NUM_CIRCLE_LEDS; i++) {
    circle.setPixelColor(i, circle.Color(random(255), random(255),
random(255))); // Establecer color aleatorio a cada led
  }
  circle.show(); // Mostrar los cambios en los LEDs
}

//Función para verificar los límites de temperatura y encender los leds de
acuerdo al valor
void color_TEMP () {
  float temperature = readTemperature();
  if(temperature < TEMP_LOW) {
    for (int i = 0; i < NUM_CIRCLE_LEDS; i++) {
      circle.setPixelColor(i, circle.Color(0, 0, 255)); // Color AZUL,
temperatura baja
    }
    circle.show(); // Mostrar los cambios en los LEDs
  } else if (temperature >= TEMP_LOW && temperature < TEMP_MEDIUM) {
```

```
    for (int i = 0; i < NUM_CIRCLE_LEDS; i++) {
      circle.setPixelColor(i, circle.Color(0, 255, 0)); // Color VERDE,
temperatura ambiente
    }
    circle.show(); // Mostrar los cambios en los LEDs
  } else if (temperature >= TEMP_MEDIUM && temperature <= TEMP_HIGH) {
    for (int i = 0; i < NUM_CIRCLE_LEDS; i++) {
      circle.setPixelColor(i, circle.Color(255, 0, 0)); // Color ROJO,
temperatura alta
    }
    circle.show(); // Mostrar los cambios en los LEDs
  }
}

//Función para encender los leds e indicar que está en estado de guardar
datos
void guardando () {
  for (int i = 0; i < NUM_CIRCLE_LEDS; i++) {
    circle.setPixelColor(i, circle.Color(200, 120, 120)); // Color para
indicar que está guardando
  }
  circle.show(); // Mostrar los cambios en los LEDs
}
```

Este es el código que se utiliza para configurar el neopixel, este se coloca en el ESP32. Primero se declaran las librerías necesarias para realizar la codificación, en este caso siendo una de Adafruit. Luego se declara el pin en donde está conectado el neopixel y también se definen variables como el número de pixeles y el brillo con el que encenderá. Luego se declaran los prototipos de función necesarios para realizar los diferentes estados que se mostrarán con diversos colores. Hay una función para el estado de espera donde se encenderán de color amarillo, así como una función que apaga todos los leds. También hay una para indicar que está enviando el dato a TIVA C luego de medirlo y hay una función que compara el dato de temperatura con valores límite preestablecidos para colocar un color dependiendo si es catalogada como baja (azul), ambiente (verde) o alta (rojo). Por último, hay una función que indica que recibió el mensaje desde TIVA C de que se guardaron los archivos en la SD.