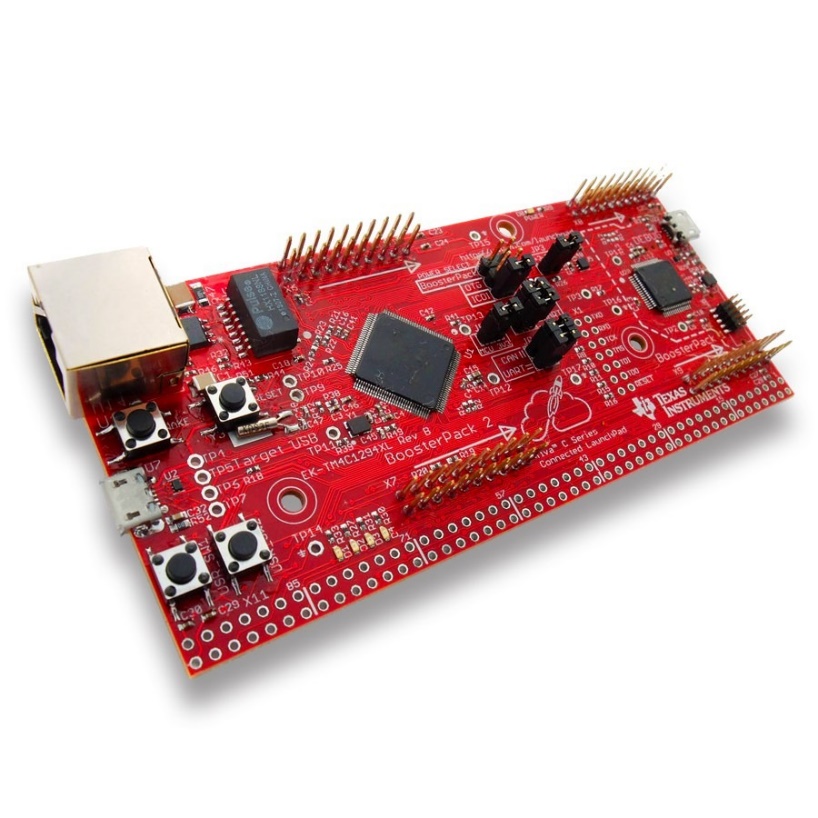
Sistemas electrónicos para Automatización

Práctica 4



Álvaro Calvo Matos

Damián Jesús Pérez Morales

Índice

[1. Introducción 2](#_Toc24390275)

[2. Ejercicio 1 3](#_Toc24390276)

[3. Ejercicio 2 9](#_Toc24390277)

[4. Comentarios acerca de la práctica 15](#_Toc24390278)

# Introducción

En la cuarta práctica de la asignatura, se pretende aprender a manejar la pantalla VM800 con el microcontrolador TM4C1294CLP y se hará, para ello, un par de ejercicios:

* Primer ejercicio: Se hará una interfaz en la pantalla en la que se pueda interactuar a través de la pantalla con los LEDs del microcontrolador; y poder interaccionar con la pantalla gracias a los botones del microcontrolador, indicando a través de la pantalla si se está pulsando el botón 1 o el botón 2. La interfaz indicada es personalizada y se mostrará el aspecto que tiene más adelante.
* Segundo ejercicio: Se diseñará gracias a la pantalla un cronómetro que contenga en ella un reloj de aguja y un reloj digital, pudiendo presionar un botón de RESET dentro de ella para reiniciar el contador y empleando los dos botones del microcontrolador para hacer el inicio de la cuenta y la parada de la misma. La interfaz del cronómetro es personalizada y se mostrará más adelante.



Ilustración 1. Pantalla VM800

# Ejercicio 1

En el ejercicio 1, como se ha comentado en la introducción de la memoria de la práctica, se programa la interfaz de un entorno en el que la pantalla interactúa directamente con el microcontrolador: si se pulsa un botón del MC se muestra un mensaje en la pantalla del microcontrolador indicando qué botón se ha pulsado y, si se pulsa uno de los botones de la pantalla, se hará notar haciendo encender uno de los LEDs del microcontrolador. La interfaz es la siguiente:



Ilustración 3. Pantalla principal del apartado 1 en reposo

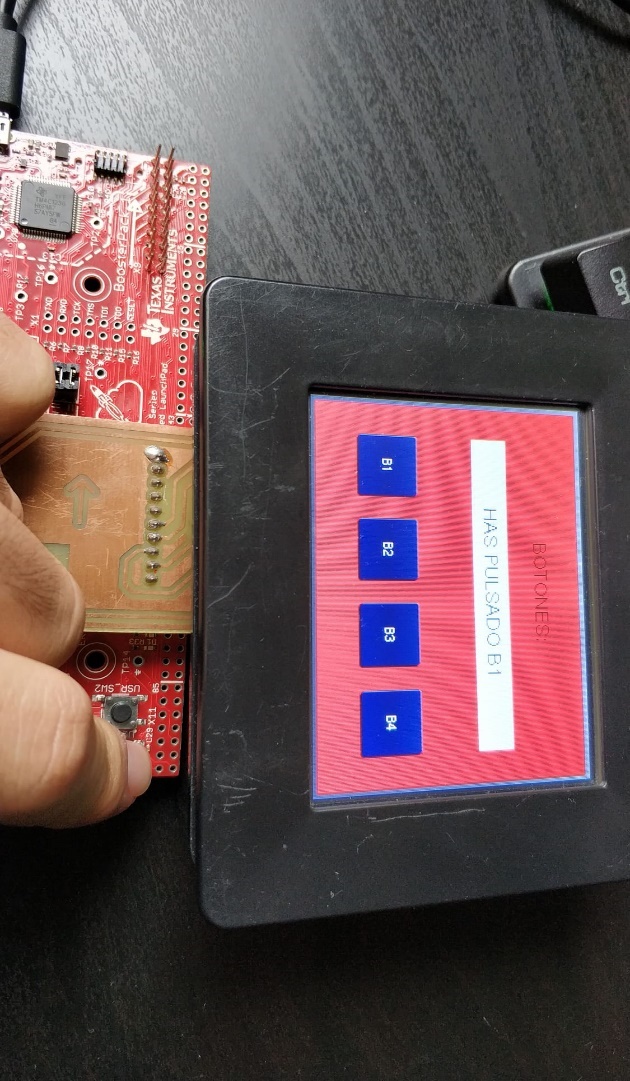


Ilustración 4. Comportamiento de la pantalla cuando se pulsa un botón del MC

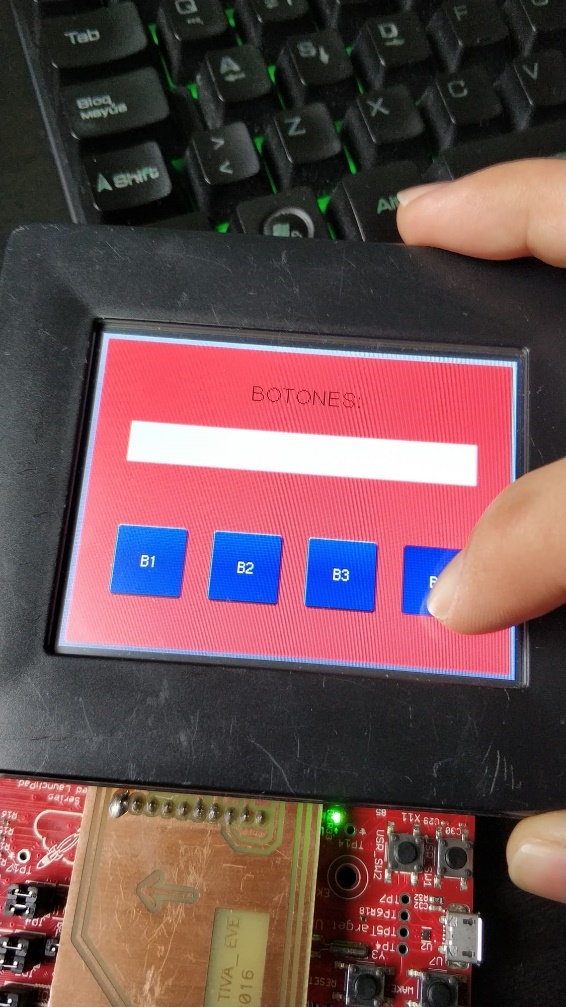


Ilustración 5. Encendido de los leds cuando se pulsa un botón de la pantalla

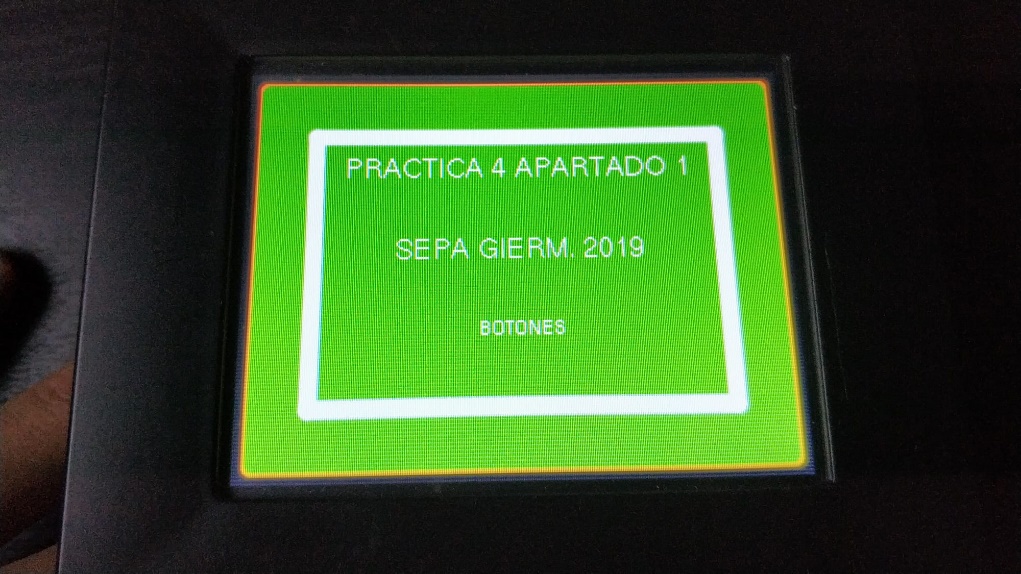


Ilustración 2. Pantalla de inicio del apartado 1.

A continuación, se mostrará el código implementado para su funcionamiento, que está comentado paso por paso:

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/hw\_types.h"

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_gpio.h"

**#include** "driverlib/ssi.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/rom\_map.h"

**#include** "driverlib/gpio.h"

**#include** "FT800\_TIVA.h"

**#include** "driverlib2.h"

//Definiciones para facilitar la lectura de los pulsadores

**#define** B1\_OFF GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0)

**#define** B1\_ON !(GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0))

**#define** B2\_OFF GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_1)

**#define** B2\_ON !(GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_1))

**#define** dword **long**

**#define** byte **char**

**#define** MSEC 40000

// Variables para el uso de la pantalla tactil

**char** chipid = 0; // Holds value of Chip ID read from the FT800

**unsigned** **long** cmdBufferRd = 0x00000000; // Store the value read from the REG\_CMD\_READ register

**unsigned** **long** cmdBufferWr = 0x00000000; // Store the value read from the REG\_CMD\_WRITE register

**unsigned** **int** t=0;

**int** Fin\_Rx = 0;

**char** Buffer\_Rx;

**unsigned** **long** POSX, POSY, BufferXY;

**unsigned** **long** POSYANT = 0;

**unsigned** **int** CMD\_Offset = 0;

**unsigned** **long** REG\_TT[6];

**const** **unsigned** **long** REG\_CAL[6]={21959,177,4294145463,14,4294950369,16094853};

**#define** NUM\_SSI\_DATA 3

**int** RELOJ;

// Variables auxiliares para los botones

**volatile** **int** B1press = 0, B2press = 0;

// Rutina de interrupcion de los botones

**void** **rutina\_interrupcion**(**void**)

{

**if**(B1\_ON){ //Si se pulsa B1 -> Max\_pos

**SysCtlDelay**(10\*MSEC);

B1press = 1;

**GPIOIntClear**(GPIO\_PORTJ\_BASE, GPIO\_PIN\_0);

}

**else** **if**(B2\_ON){ //Si se pulsa B2 -> Min\_pos

**SysCtlDelay**(10\*MSEC);

B2press = 1;

**GPIOIntClear**(GPIO\_PORTJ\_BASE, GPIO\_PIN\_1);

}

}

**int** **main**(**void**)

{

**int** i;

RELOJ = **SysCtlClockFreqSet**((SYSCTL\_XTAL\_25MHZ | SYSCTL\_OSC\_MAIN | SYSCTL\_USE\_PLL | SYSCTL\_CFG\_VCO\_480), 120000000);

HAL\_Init\_SPI(2, RELOJ); //Boosterpack a usar, Velocidad del MC

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOJ);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPION);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF);

// Configuracion de los leds

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0 |GPIO\_PIN\_4); //F0 y F4: salidas

**GPIOPinTypeGPIOOutput**(GPIO\_PORTN\_BASE, GPIO\_PIN\_0 |GPIO\_PIN\_1); //N0 y N1: salidas

// Ponemos resistencias de pull-up en los pulsadores

**GPIOPinTypeGPIOInput**(GPIO\_PORTJ\_BASE, GPIO\_PIN\_0|GPIO\_PIN\_1);

**GPIOPadConfigSet**(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0|GPIO\_PIN\_1,GPIO\_STRENGTH\_2MA,GPIO\_PIN\_TYPE\_STD\_WPU);

// Configuracion de las interrupciones en los pulsadores

**GPIOIntEnable**(GPIO\_PORTJ\_BASE, GPIO\_PIN\_0|GPIO\_PIN\_1); //Habilitar pines de interrupción J0, J1

**GPIOIntRegister**(GPIO\_PORTJ\_BASE, rutina\_interrupcion); //Registrar (definir) la rutina de interrupción

**IntEnable**(INT\_GPIOJ); //Habilitar interrupción del pto J

**IntMasterEnable**(); //Habilitar globalmente las ints

Inicia\_pantalla();

// Note: Keep SPI below 11MHz here

// =======================================================================

// Delay before we begin to display anything

// =======================================================================

**SysCtlDelay**(RELOJ/3);

// ================================================================================================================

// PANTALLA INICIAL

// ================================================================================================================

// Borramos la pantalla y la rellenamos del color indicado (gris)

Nueva\_pantalla(0x10,0x10,0x10);

// Marco rectangular anaranjado al filo de la pantalla

ComColor(255,160,6);

ComLineWidth(5);

Comando(CMD\_BEGIN\_RECTS);

ComVertex2ff(10,10);

ComVertex2ff(310,230);

// Hacemos un rectangulo verde que hará las veces de fondo

ComColor(65,202,42);

ComVertex2ff(12,12);

ComVertex2ff(308,228);

Comando(CMD\_END);

// Imprimimos texto informativo inicial

ComColor(0xff,0xff,0xff);

ComTXT(160,50, 22, OPT\_CENTERX,"PRACTICA 4 APARTADO 1");

ComTXT(160,100, 22, OPT\_CENTERX," SEPA GIERM. 2019 ");

ComTXT(160,150, 20, OPT\_CENTERX,"BOTONES");

// Pintamos cuatro rectangulos blancos para hacer un marco alrededor del texto

Comando(CMD\_BEGIN\_LINES);

ComVertex2ff(40,40);

ComVertex2ff(280,40);

ComVertex2ff(280,40);

ComVertex2ff(280,200);

ComVertex2ff(280,200);

ComVertex2ff(40,200);

ComVertex2ff(40,200);

ComVertex2ff(40,40);

Comando(CMD\_END);

// Comando para proceder a pintar por pantalla

Dibuja();

// Esperamos a que alguien toque la pantalla para continuar

Espera\_pant();

**for**(i=0;i<6;i++) Esc\_Reg(REG\_TOUCH\_TRANSFORM\_A+4\*i, REG\_CAL[i]);

**while**(1)

{

// Lee la posicion del dedo y la escribe en dos var globales x e y

Lee\_pantalla();

// Borramos el contenido de la pantalla y lo rellenamos de gris

Nueva\_pantalla(0x10,0x10,0x10);

// Pintamos un gradiente de claro a oscuro en el fondo

ComGradient(0,0,GRIS\_CLARO,0,240,GRIS\_OSCURO);

// Pintamos rectangulo rojo del fondo

ComColor(0xff,0x00,0x00);

Comando(CMD\_BEGIN\_RECTS);

ComVertex2ff(5,5);

ComVertex2ff(315,235);

Comando(CMD\_END);

// Pintamos el rectangulo blanco para el texto

ComColor(0xFF,0xFF,0xFF);

Comando(CMD\_BEGIN\_RECTS);

ComVertex2ff(30,70);

ComVertex2ff(290,100);

Comando(CMD\_END);

// Texto "BOTONES" sobre el cuadro blanco del texto

ComColor(0x00,0x00,0x00);

ComTXT(160,30, 22, OPT\_CENTERX,"BOTONES:");

// Botones tactiles, apretados o no en funcion de la posicion actual del dedo

ComColor(0xff,0xff,0xff);

**if**(POSX>30 && POSX<80 && POSY>150 && POSY<200) // Boton 1

{

// Pitamos el boton apretado

ComButton(30,150,50,50,20,256,"B1");

// Encendemos el led 1

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_PIN\_0);

}

**else**

{

// Pintamos el boton sin pulsar

ComButton(30,150,50,50,20,0,"B1");

// Apagamos el led 1

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, 0);

}

**if**(POSX>100 && POSX<150 && POSY>150 && POSY<200) //Boton Left

{

// Pitamos el boton apretado

ComButton(100,150,50,50,20,256,"B2");

// Encendemos el led 2

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4, GPIO\_PIN\_4);

}

**else**

{

// Pintamos el boton sin pulsar

ComButton(100,150,50,50,20,0,"B2");

// Apagamos el led 2

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4, 0);

}

**if**(POSX>170 && POSX<220 && POSY>150 && POSY<200) //Boton Left

{

// Pitamos el boton apretado

ComButton(170,150,50,50,20,256,"B3");

// Encendemos el led 3

**GPIOPinWrite**(GPIO\_PORTN\_BASE, GPIO\_PIN\_0, GPIO\_PIN\_0);

}

**else**

{

// Pintamos el boton sin pulsar

ComButton(170,150,50,50,20,0,"B3");

// Apagamos el led 3

**GPIOPinWrite**(GPIO\_PORTN\_BASE, GPIO\_PIN\_0, 0);

}

**if**(POSX>240 && POSX<290 && POSY>150 && POSY<200) //Boton Left

{

// Pitamos el boton apretado

ComButton(240,150,50,50,20,256,"B4");

// Encendemos el led 4

**GPIOPinWrite**(GPIO\_PORTN\_BASE, GPIO\_PIN\_1, GPIO\_PIN\_1);

}

**else**

{

// Pintamos el boton sin pulsar

ComButton(240,150,50,50,20,0,"B4");

// Apagamos el led 4

**GPIOPinWrite**(GPIO\_PORTN\_BASE, GPIO\_PIN\_1, 0);

}

// Si estamos pulsando un boton fisico, mostramos el mensaje

**if**(B1\_OFF) B1press = 0;

**if**(B2\_OFF) B2press = 0;

**if**(B1press)

{

ComColor(0x00,0x00,0x00);

ComTXT(160,75, 22, OPT\_CENTERX,"HAS PULSADO B1");

}

**if**(B2press)

{

ComColor(0x00,0x00,0x00);

ComTXT(160,75, 22, OPT\_CENTERX,"HAS PULSADO B2");

}

// Damos la orden para que se pinte **todo** lo anterior por pantalla

Dibuja();

}

}

# Ejercicio 2

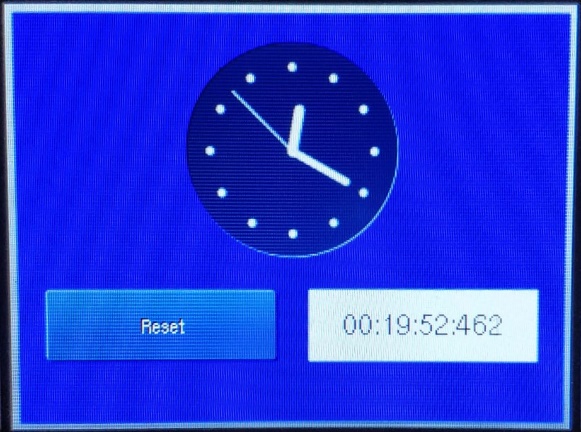
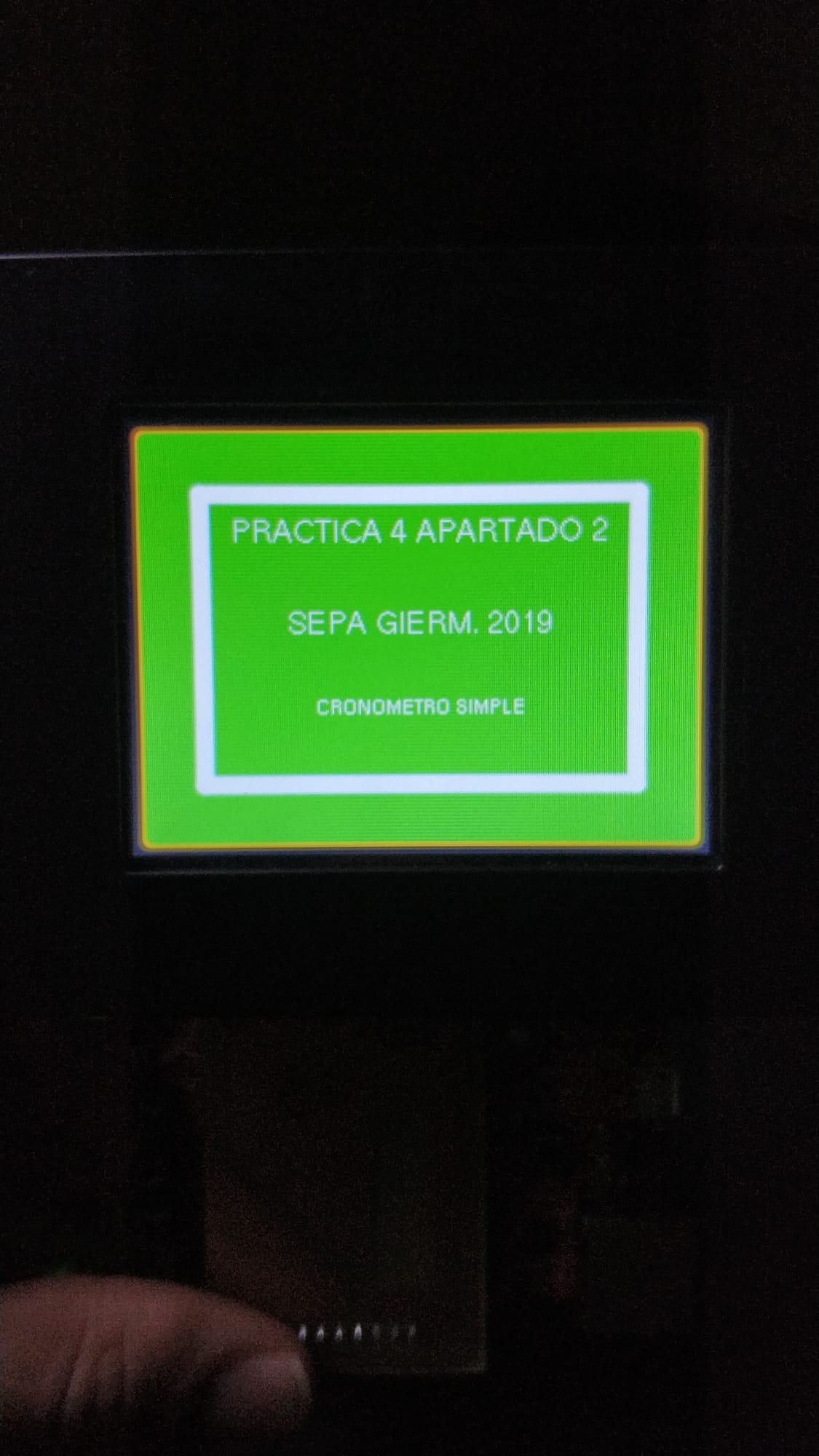
En este ejercicio se pretende hacer un cronómetro tanto con reloj de agujas como digital que se mostrará a través de la pantalla VM800 y, para su funcionamiento, se ha empleado un timer a 120MHz. Para poder interactuar con el cronómetro, se hace uso de los botones del microcontrolador para iniciar y para parar de cronometrar; y se inserta un botón adicional en la pantalla para poder reiniciar la cuenta. La interfaz resultante es la siguiente:

Ilustración 7. Interfaz principal del apartado 2.

Ilustración 6. Pantalla de inicio del apartado 2.

El código del funcionamiento es el siguiente:

**#include** <stdio.h>

**#include** <string.h>

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/hw\_types.h"

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_gpio.h"

**#include** "driverlib/ssi.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/rom\_map.h"

**#include** "driverlib/gpio.h"

**#include** "FT800\_TIVA.h"

**#include** "driverlib2.h"

//Definiciones para facilitar la lectura de los pulsadores

**#define** B1\_OFF GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0)

**#define** B1\_ON !(GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0))

**#define** B2\_OFF GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_1)

**#define** B2\_ON !(GPIOPinRead(GPIO\_PORTJ\_BASE,GPIO\_PIN\_1))

**#define** dword **long**

**#define** byte **char**

**#define** MSEC 40000

// Variables para el uso de la pantalla tactil

**char** chipid = 0; // Holds value of Chip ID read from the FT800

**unsigned** **long** cmdBufferRd = 0x00000000; // Store the value read from the REG\_CMD\_READ register

**unsigned** **long** cmdBufferWr = 0x00000000; // Store the value read from the REG\_CMD\_WRITE register

**unsigned** **int** t=0;

**int** Fin\_Rx = 0;

**char** Buffer\_Rx;

**unsigned** **long** POSX, POSY, BufferXY;

**unsigned** **long** POSYANT=0;

**unsigned** **int** CMD\_Offset = 0;

**unsigned** **long** REG\_TT[6];

**const** **unsigned** **long** REG\_CAL[6]={21959,177,4294145463,14,4294950369,16094853};

**#define** NUM\_SSI\_DATA 3

**int** RELOJ;

// La variable estado indica el estado del cronometro

// estado = 1 -> cronometro encendido

// estado = 0 -> cronometro apagado

**volatile** **int** estado = 0;

// Variables para el tiempo del cronometro

**volatile** **int** horas = 0, minutos = 0, segundos = 0, milisegundos = 0;

**char** cadena[24];

// Rutina de interrupcion de los botones

**void** **rutina\_interrupcion**(**void**)

{

**if**(B1\_ON){ //Si se pulsa B1 -> Max\_pos

**SysCtlDelay**(10\*MSEC);

estado = 1;

**GPIOIntClear**(GPIO\_PORTJ\_BASE, GPIO\_PIN\_0);

}

**else** **if**(B2\_ON){ //Si se pulsa B2 -> Min\_pos

**SysCtlDelay**(10\*MSEC);

estado = 0;

**GPIOIntClear**(GPIO\_PORTJ\_BASE, GPIO\_PIN\_1);

}

}

// Rutina de interrupcion del timer0

**void** **IntTimer0**(**void**)

{

// Variables que contabilizan el tiempo del sistema

// Actualizamos segundos, minutos y horas segun corresponda

**if**(estado == 1)

{

milisegundos ++;

**if** (milisegundos >= 1000)

{

milisegundos = 0;

segundos ++;

**if** (segundos == 60)

{

segundos = 0;

minutos ++;

**if** (minutos == 60)

{

minutos = 0;

horas ++;

}

}

}

}

// Limpiamos la interrupcion del timer

**TimerIntClear**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);

}

// Definicion de la funcion que pinta el reloj por pantalla

**void** **cmd\_clock**(int16\_t x, int16\_t y, int16\_t r, uint16\_t options, uint16\_t h, uint16\_t m, uint16\_t s, uint16\_t ms)

{

EscribeRam32(CMD\_CLOCK);

EscribeRam16(x);

EscribeRam16(y);

EscribeRam16(r);

EscribeRam16(options);

EscribeRam16(h);

EscribeRam16(m);

EscribeRam16(s);

EscribeRam16(ms);

PadFIFO();

}

**int** **main**(**void**)

{

**int** i;

RELOJ = **SysCtlClockFreqSet**((SYSCTL\_XTAL\_25MHZ | SYSCTL\_OSC\_MAIN | SYSCTL\_USE\_PLL | SYSCTL\_CFG\_VCO\_480), 120000000);

HAL\_Init\_SPI(2, RELOJ); //Boosterpack a usar, Velocidad del MC

// Configuracion de los perifericos

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOJ);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPION);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF);

// Configuracion de los leds

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0 |GPIO\_PIN\_4); //F0 y F4: salidas

**GPIOPinTypeGPIOOutput**(GPIO\_PORTN\_BASE, GPIO\_PIN\_0 |GPIO\_PIN\_1); //N0 y N1: salidas

// Ponemos resistencias de pull-up en los pulsadores

**GPIOPinTypeGPIOInput**(GPIO\_PORTJ\_BASE, GPIO\_PIN\_0|GPIO\_PIN\_1);

**GPIOPadConfigSet**(GPIO\_PORTJ\_BASE,GPIO\_PIN\_0|GPIO\_PIN\_1,GPIO\_STRENGTH\_2MA,GPIO\_PIN\_TYPE\_STD\_WPU);

// Configuracion de las interrupciones de los botones

**GPIOIntEnable**(GPIO\_PORTJ\_BASE, GPIO\_PIN\_0|GPIO\_PIN\_1); //Habilitar pines de interrupción J0, J1

**GPIOIntRegister**(GPIO\_PORTJ\_BASE, rutina\_interrupcion); //Registrar (definir) la rutina de interrupción

**IntEnable**(INT\_GPIOJ); //Habilitar interrupción del pto J

// Configuracion del TIMER

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0); //Habilita T0

**TimerClockSourceSet**(TIMER0\_BASE, TIMER\_CLOCK\_SYSTEM); //T0 a 120MHz

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); //T0 periodico y conjunto (32b)

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, 120000 - 1); //Para cada ms

**TimerIntRegister**(TIMER0\_BASE,TIMER\_A,IntTimer0);

**IntEnable**(INT\_TIMER0A); //Habilitar las interrupciones globales de los timers

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); // Habilitar las interrupciones de timeout

**IntMasterEnable**(); //Habilitacion global de interrupciones

**TimerEnable**(TIMER0\_BASE, TIMER\_A); //Habilitar Timer0, 1A

**IntMasterEnable**(); //Habilitar globalmente las ints

Inicia\_pantalla();

// Note: Keep SPI below 11MHz here

// =======================================================================

// Delay before we begin to display anything

// =======================================================================

**SysCtlDelay**(RELOJ/3);

// ================================================================================================================

// PANTALLA INICIAL

// ================================================================================================================

// Borramos la pantalla y la rellenamos del color indicado (gris)

Nueva\_pantalla(0x10,0x10,0x10);

// Marco rectangular anaranjado al filo de la pantalla

ComColor(255,160,6);

ComLineWidth(5);

Comando(CMD\_BEGIN\_RECTS);

ComVertex2ff(10,10);

ComVertex2ff(310,230);

// Hacemos un rectangulo verde que hará las veces de fondo

ComColor(65,202,42);

ComVertex2ff(12,12);

ComVertex2ff(308,228);

Comando(CMD\_END);

// Imprimimos texto informativo inicial

ComColor(0xff,0xff,0xff);

ComTXT(160,50, 22, OPT\_CENTERX,"PRACTICA 4 APARTADO 2");

ComTXT(160,100, 22, OPT\_CENTERX," SEPA GIERM. 2019 ");

ComTXT(160,150, 20, OPT\_CENTERX,"CRONOMETRO SIMPLE");

// Pintamos cuatro rectangulos blancos para hacer un marco alrededor del texto

Comando(CMD\_BEGIN\_LINES);

ComVertex2ff(40,40);

ComVertex2ff(280,40);

ComVertex2ff(280,40);

ComVertex2ff(280,200);

ComVertex2ff(280,200);

ComVertex2ff(40,200);

ComVertex2ff(40,200);

ComVertex2ff(40,40);

Comando(CMD\_END);

// Mandamos la orden a la pantalla para que pinte lo anterior

Dibuja();

// Esperamos a que alguien toque la pantalla para continuar

Espera\_pant();

**for**(i=0;i<6;i++) Esc\_Reg(REG\_TOUCH\_TRANSFORM\_A+4\*i, REG\_CAL[i]);

**while**(1)

{

// Lee la posicion del dedo y la escribe en dos var globales x e y

Lee\_pantalla();

// Borramos el contenido de la pantalla y lo rellenamos de gris

Nueva\_pantalla(0x10,0x10,0x10);

// Pintamos un gradiente de claro a oscuro en el fondo

ComGradient(0,0,GRIS\_CLARO,0,240,GRIS\_OSCURO);

// Pintamos un rectangulo azul en el fondo

ComColor(0x00,0x00,0xaa);

Comando(CMD\_BEGIN\_RECTS); //Pintar rectángulo del fondo

ComVertex2ff(5,5);

ComVertex2ff(315,235);

Comando(CMD\_END);

//Pintamos el rectángulo del cronometro digital

ComColor(0xFF,0xFF,0xFF);

Comando(CMD\_BEGIN\_RECTS);

ComVertex2ff(170,160);

ComVertex2ff(300,200);

Comando(CMD\_END);

// Pintamos el boton tactil de reset

ComColor(0xff,0xff,0xff);

**if**(POSX>20 && POSX<150 && POSY>160 && POSY<200) //Boton Left

{

// Pitamos el boton apretado

ComButton(20,160,130,40,20,256,"Reset");

// Reestablecemos el tiempo a cero

horas = 0;

minutos = 0;

segundos = 0;

milisegundos = 0;

}

**else**

{

// Pitamos el boton sin pulsar

ComButton(20,160,130,40,20,0,"Reset");

}

// WIDGET RELOJ. Pintamos el cronómetro analógico

cmd\_clock(160, 80, 60, 0, horas, minutos, segundos, milisegundos);

// Mostramos el texto del cronometro digital por pantalla

**sprintf**(cadena,"%.2d:%.2d:%.2d:%.3d",horas, minutos, segundos, milisegundos);

ComColor(0x00,0x00,0x00);

ComTXT(235, 180, 22, OPT\_CENTER,cadena);

// Damos la orden para pintar **todo** lo anterior en la pantalla

Dibuja();

}

}

# Comentarios acerca de la práctica

En cuanto a la realización de la práctica, se ha tenido el inconveniente de que no se ha dispuesto de la pantalla para ir comprobando los resultados obtenidos e ir corrigiéndolos con prueba y error, sino que se ha programado para una posterior comprobación.

La elección de los colores también merece una mención especial, ya que no es sencillo, con una codificación RGB, elegir exactamente el color que uno quiere. Para agilizar pues este proceso, teniendo en cuenta que la mayor parte del tiempo no se ha dispuesto de la pantalla para mostrar el color que se estaba eligiendo, se ha optado por colores primarios y secundarios, dejando de lado las mezclas de colores intermedias.