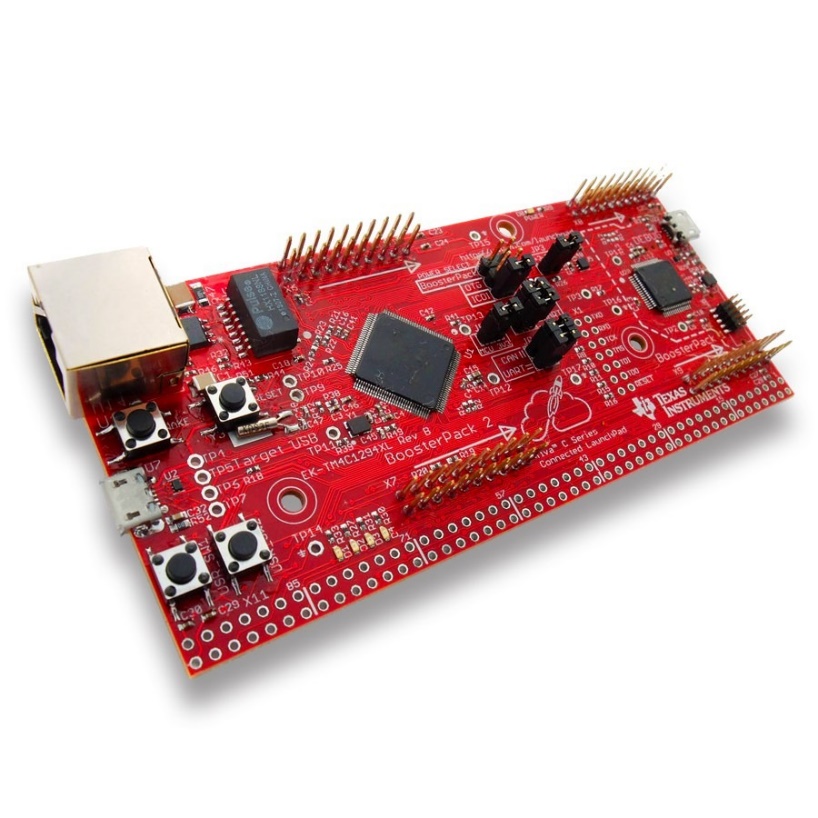
Sistemas electrónicos para Automatización

Práctica 4



Álvaro Calvo Matos

Damián Jesús Pérez Morales

Índice

[1. Introducción 2](#_Toc23261779)

[2. Ejercicio 1 3](#_Toc23261780)

[3. Ejercicio 2 8](#_Toc23261781)

[4. Ejercicio 3 13](#_Toc23261782)

[5. Comentarios acerca de la práctica 20](#_Toc23261783)

# 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:

(INSERTAR IMÁGENES DE LA PANTALLA Y HACIÉNDOLO FUNCIONAR)

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

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <stdio.h>

**#include** <math.h>

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

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

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

**#include** "inc/hw\_i2c.h"

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

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

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

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

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

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

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

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

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

**#include** "driverlib/i2c.h"

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

**#include** "utils/uartstdio.h"

**#include** "utils/uartstdio.c"

**#include** "HAL\_I2C.h"

**#include** "sensorlib2.h"

**void** **Timer0IntHandler**(**void**);

**int** RELOJ;

**char** Cambia = 0;

**char** string[80];

**int** DevID = 0;

// BMI160/BMM150

**struct** bmi160\_mag\_xyz\_s32\_t s\_magcompXYZ;

uint8\_t Sensor\_OK = 0;

**#define** BP 2

uint8\_t Opt\_OK, Tmp\_OK, Bme\_OK, Bmi\_OK;

**float** pi = 3.141592654;

**float** angulo;

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

{

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

// Configuracion del timer para escribir datos por pantalla cada medio segundo

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0);

**TimerClockSourceSet**(TIMER0\_BASE, TIMER\_CLOCK\_SYSTEM);

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC);

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, RELOJ/2 -1);

**TimerIntRegister**(TIMER0\_BASE, TIMER\_A,Timer0IntHandler);

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

**TimerEnable**(TIMER0\_BASE, TIMER\_A);

// Configuracion de al UART para comunicarnos con el ordenador por puerto serie

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

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

**GPIOPinConfigure**(GPIO\_PA0\_U0RX);

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

**UARTStdioConfig**(0, 115200, RELOJ);

// Deteccion de al posición en la que está pinchado el sensors BoosterPack

**if**(Detecta\_BP(1))

{

**UARTprintf**("\n--------------------------------------");

**UARTprintf**("\n BOOSTERPACK detectado en posicion 1");

**UARTprintf**("\n Configurando puerto I2C0");

**UARTprintf**("\n--------------------------------------");

Conf\_Boosterpack(1, RELOJ);

}

**else** **if**(Detecta\_BP(2))

{

**UARTprintf**("\n--------------------------------------");

**UARTprintf**("\n BOOSTERPACK detectado en posicion 2");

**UARTprintf**("\n Configurando puerto I2C2");

**UARTprintf**("\n--------------------------------------");

Conf\_Boosterpack(2, RELOJ);

}

**else**

{

**UARTprintf**("\n--------------------------------------");

**UARTprintf**("\n Ningun BOOSTERPACK detectado :-/ ");

**UARTprintf**("\n Saliendo");

**UARTprintf**("\n--------------------------------------");

**return** 0;

}

// Comunicamos el estado de los sensores por si ha habido un error

**UARTprintf**("\033[2J \033[1;1H Inicializando OPT3001... ");

Sensor\_OK = Test\_I2C\_Dir(OPT3001\_SLAVE\_ADDRESS);

**if**(!Sensor\_OK)

{

**UARTprintf**("Error en OPT3001\n");

Opt\_OK=0;

}

**else**

{

OPT3001\_init();

**UARTprintf**("Hecho!\n");

**UARTprintf**("Leyendo DevID... ");

DevID=OPT3001\_readDeviceId();

**UARTprintf**("DevID= 0X%x \n", DevID);

Opt\_OK = 1;

}

**UARTprintf**("Inicializando ahora el TMP007...");

Sensor\_OK = Test\_I2C\_Dir(TMP007\_I2C\_ADDRESS);

**if**(!Sensor\_OK)

{

**UARTprintf**("Error en TMP007\n");

Tmp\_OK = 0;

}

**else**

{

sensorTmp007Init();

**UARTprintf**("Hecho! \nLeyendo DevID... ");

DevID=sensorTmp007DevID();

**UARTprintf**("DevID= 0X%x \n", DevID);

sensorTmp007Enable(true);

Tmp\_OK = 1;

}

**UARTprintf**("Inicializando BME280... ");

Sensor\_OK=Test\_I2C\_Dir(BME280\_I2C\_ADDRESS2);

**if**(!Sensor\_OK)

{

**UARTprintf**("Error en BME280\n");

Bme\_OK = 0;

}

**else**

{

bme280\_data\_readout\_template();

bme280\_set\_power\_mode(BME280\_NORMAL\_MODE);

**UARTprintf**("Hecho! \nLeyendo DevID... ");

readI2C(BME280\_I2C\_ADDRESS2,BME280\_CHIP\_ID\_REG, &DevID, 1);

**UARTprintf**("DevID= 0X%x \n", DevID);

Bme\_OK = 1;

}

Sensor\_OK = Test\_I2C\_Dir(BMI160\_I2C\_ADDR2);

**if**(!Sensor\_OK)

{

**UARTprintf**("Error en BMI160\n");

Bmi\_OK = 0;

}

**else**

{

**UARTprintf**("Inicializando BMI160, modo NAVIGATION... ");

bmi160\_initialize\_sensor();

bmi160\_config\_running\_mode(APPLICATION\_NAVIGATION);

**UARTprintf**("Hecho! \nLeyendo DevID... ");

readI2C(BMI160\_I2C\_ADDR2,BMI160\_USER\_CHIP\_ID\_ADDR, &DevID, 1);

**UARTprintf**("DevID= 0X%x \n", DevID);

Bmi\_OK = 1;

}

// Bucle principal

**while**(1)

{

// Si el sensor está listo, leemos el magnetómetro

**if**(Bmi\_OK)

{

bmi160\_bmm150\_mag\_compensate\_xyz(&s\_magcompXYZ);

}

// Calculamos el angulo al que esta apuntando el magnetómetro

**if**(s\_magcompXYZ.x == 0)

{

// Si la componente x es cero, estamos en +-90 grados

**if**(s\_magcompXYZ.y > 0) angulo = pi/2.0;

**else** angulo = 3.0\*pi/2.0;

}

**else**

{

// Si la componnente x no es cero, aplicamos la arcotangente de y/x

angulo = **atanf**((**float**)s\_magcompXYZ.y/(**float**)s\_magcompXYZ.x);

// Ahora identificamos el cuadrante a partir de los signos de x e y

**if**(s\_magcompXYZ.x > 0 && s\_magcompXYZ.y > 0) angulo = angulo;

**else** **if**(s\_magcompXYZ.x < 0 && s\_magcompXYZ.y > 0) angulo += pi;

**else** **if**(s\_magcompXYZ.x < 0 && s\_magcompXYZ.y < 0) angulo += pi;

**else** **if**(s\_magcompXYZ.x > 0 && s\_magcompXYZ.y < 0) angulo += 2\*pi;

}

// Imprimimos los datos por pantalla cada medio segundo

**if**(Cambia == 1)

{

Cambia = 0;

**sprintf**(string, "Angulo = %f (rad)\n",angulo);

**UARTprintf**(string);

}

}

**return** 0;

}

// En la rutina de interrupcion solo cambiaos la variable que permite imprimir por pantalla y borramos el flag

**void** **Timer0IntHandler**(**void**)

{

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

Cambia = 1;

}

# 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:

(INSERTAR IMÁGENES)

El código del funcionamiento es el siguiente:

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <stdio.h>

**#include** <math.h>

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

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

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

**#include** "inc/hw\_i2c.h"

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

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

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

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

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

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

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

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

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

**#include** "driverlib/i2c.h"

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

**#include** "utils/uartstdio.h"

**#include** "utils/uartstdio.c"

**#include** "HAL\_I2C.h"

**#include** "sensorlib2.h"

**void** **Timer0IntHandler**(**void**);

**int** RELOJ;

**char** Cambia = 0;

**char** string[80];

**int** DevID = 0;

// BMI160/BMM150

**struct** bmi160\_mag\_xyz\_s32\_t s\_magcompXYZ;

uint8\_t Sensor\_OK = 0;

**#define** BP 2

uint8\_t Opt\_OK, Tmp\_OK, Bme\_OK, Bmi\_OK;

**float** pi = 3.141592654;

**float** angulo, grados;

**volatile** **int** direccion;

// Vector con las direcciones almacenadas para facilitar el proceso

**const** **char**\* Rumbo[16]= {"N ",

"NNO",

"NO ",

"ONO",

"O ",

"OSO",

"SO ",

"SSO",

"S ",

"SSE",

"SE ",

"ESE",

"E ",

"ENE",

"NE ",

"NNE"};

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

{

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

// Configuracion del timer para escribir datos por pantalla cada medio segundo

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0);

**TimerClockSourceSet**(TIMER0\_BASE, TIMER\_CLOCK\_SYSTEM);

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC);

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, RELOJ/2 -1);

**TimerIntRegister**(TIMER0\_BASE, TIMER\_A,Timer0IntHandler);

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

**TimerEnable**(TIMER0\_BASE, TIMER\_A);

// Configuracion de al UART para comunicarnos con el ordenador por puerto serie

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

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

**GPIOPinConfigure**(GPIO\_PA0\_U0RX);

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

**UARTStdioConfig**(0, 115200, RELOJ);

// Deteccion de al posición en la que está pinchado el sensors BoosterPack

**if**(Detecta\_BP(1))

{

**UARTprintf**("\n--------------------------------------");

**UARTprintf**("\n BOOSTERPACK detectado en posicion 1");

**UARTprintf**("\n Configurando puerto I2C0");

**UARTprintf**("\n--------------------------------------");

Conf\_Boosterpack(1, RELOJ);

}

**else** **if**(Detecta\_BP(2))

{

**UARTprintf**("\n--------------------------------------");

**UARTprintf**("\n BOOSTERPACK detectado en posicion 2");

**UARTprintf**("\n Configurando puerto I2C2");

**UARTprintf**("\n--------------------------------------");

Conf\_Boosterpack(2, RELOJ);

}

**else**

{

**UARTprintf**("\n--------------------------------------");

**UARTprintf**("\n Ningun BOOSTERPACK detectado :-/ ");

**UARTprintf**("\n Saliendo");

**UARTprintf**("\n--------------------------------------");

**return** 0;

}

// Comunicamos el estado de los sensores por si ha habido un error

**UARTprintf**("\033[2J \033[1;1H Inicializando OPT3001... ");

Sensor\_OK=Test\_I2C\_Dir(OPT3001\_SLAVE\_ADDRESS);

**if**(!Sensor\_OK)

{

**UARTprintf**("Error en OPT3001\n");

Opt\_OK=0;

}

**else**

{

OPT3001\_init();

**UARTprintf**("Hecho!\n");

**UARTprintf**("Leyendo DevID... ");

DevID=OPT3001\_readDeviceId();

**UARTprintf**("DevID= 0X%x \n", DevID);

Opt\_OK=1;

}

**UARTprintf**("Inicializando ahora el TMP007...");

Sensor\_OK=Test\_I2C\_Dir(TMP007\_I2C\_ADDRESS);

**if**(!Sensor\_OK)

{

**UARTprintf**("Error en TMP007\n");

Tmp\_OK=0;

}

**else**

{

sensorTmp007Init();

**UARTprintf**("Hecho! \nLeyendo DevID... ");

DevID=sensorTmp007DevID();

**UARTprintf**("DevID= 0X%x \n", DevID);

sensorTmp007Enable(true);

Tmp\_OK=1;

}

**UARTprintf**("Inicializando BME280... ");

Sensor\_OK=Test\_I2C\_Dir(BME280\_I2C\_ADDRESS2);

**if**(!Sensor\_OK)

{

**UARTprintf**("Error en BME280\n");

Bme\_OK=0;

}

**else**

{

bme280\_data\_readout\_template();

bme280\_set\_power\_mode(BME280\_NORMAL\_MODE);

**UARTprintf**("Hecho! \nLeyendo DevID... ");

readI2C(BME280\_I2C\_ADDRESS2,BME280\_CHIP\_ID\_REG, &DevID, 1);

**UARTprintf**("DevID= 0X%x \n", DevID);

Bme\_OK=1;

}

Sensor\_OK=Test\_I2C\_Dir(BMI160\_I2C\_ADDR2);

**if**(!Sensor\_OK)

{

**UARTprintf**("Error en BMI160\n");

Bmi\_OK=0;

}

**else**

{

**UARTprintf**("Inicializando BMI160, modo NAVIGATION... ");

bmi160\_initialize\_sensor();

bmi160\_config\_running\_mode(APPLICATION\_NAVIGATION);

**UARTprintf**("Hecho! \nLeyendo DevID... ");

readI2C(BMI160\_I2C\_ADDR2,BMI160\_USER\_CHIP\_ID\_ADDR, &DevID, 1);

**UARTprintf**("DevID= 0X%x \n", DevID);

Bmi\_OK=1;

}

**while**(1)

{

// Si el sensor esta disponible, leemos el magnetómetro

**if**(Bmi\_OK)

{

bmi160\_bmm150\_mag\_compensate\_xyz(&s\_magcompXYZ);

}

// Calculamos el angulo en funcion de las lecturas del magnetometro

**if**(s\_magcompXYZ.x == 0)

{

// Si la componente x es cero, estamos en +-90 grados

**if**(s\_magcompXYZ.y > 0) angulo = pi/2.0;

**else** angulo = 3.0\*pi/2.0;

}

**else**

{

// Si la componnente x no es cero, aplicamos la arcotangente de y/x

angulo = **atanf**((**float**)s\_magcompXYZ.y/(**float**)s\_magcompXYZ.x);

// Ahora identificamos el cuadrante a partir de los signos de x e y

**if**(s\_magcompXYZ.x > 0 && s\_magcompXYZ.y > 0) angulo = angulo;

**else** **if**(s\_magcompXYZ.x < 0 && s\_magcompXYZ.y > 0) angulo += pi;

**else** **if**(s\_magcompXYZ.x < 0 && s\_magcompXYZ.y < 0) angulo += pi;

**else** **if**(s\_magcompXYZ.x > 0 && s\_magcompXYZ.y < 0) angulo += 2\*pi;

}

// Calculamos la direccion de la brújula

grados = angulo\*180/pi; // Cambio de unidades de radianes a grados

direccion = **floor**((grados + 11.25)/22.5); // Division que permite obtener el sector de 0 a 16

**if**(direccion == 16) direccion = 0; // El sector 16 en realidad es parte del cero

// Imprimimos los datos por pantalla cada medio segundo

**if**(Cambia == 1)

{

Cambia = 0;

**sprintf**(string, "\033[10;1HÁngulo = %f (rad)\tRumbo: %s\n",angulo,Rumbo[direccion]);

**UARTprintf**(string);

}

}

**return** 0;

}

// En la rutina de interrupcion solo cambiaos la variable que permite imprimir por pantalla y borramos el flag

**void** **Timer0IntHandler**(**void**)

{

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

Cambia=1;

}

# 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.

SAODHOAS

ASJBD

ASJBDOAS

ASKDB