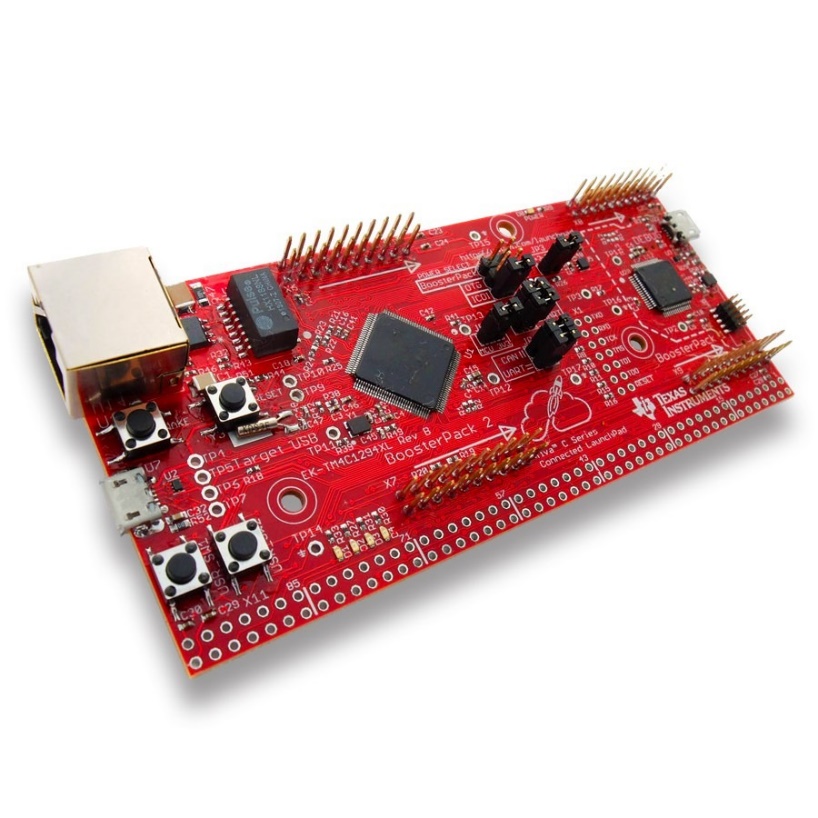
Sistemas electrónicos para Automatización

Práctica 3



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 tercera práctica de la asignatura aprenderemos a manejar los sensores del *Sensors Boosterpack*. Además, se pretende afianzar el uso del motor paso a paso y de la UART, con la que se mostrará en todo momento por pantalla la lectura de los sensores necesarios.

La práctica se desarrolla a lo largo de tres ejercicios que añaden especificaciones al código de los apartados anteriores.

* Primer ejercicio: se programa la lectura del magnetómetro, la conversión de estos datos en ángulo respecto al norte, así como la UART para visualizar el ángulo por pantalla.
* Segundo ejercicio: al ejercicio anterior se le añade el cálculo y disposición por pantalla de la dirección de la brújula, discretizada en los cuatro puntos cardinales, los cuatro rumbos laterales y los ocho rumbos colaterales tal como se muestra en la figura 1.
* Tercer ejercicio: ahora añadimos el motor paso a paso para que haga las veces de brújula, de forma que apunte siempre “al norte”.

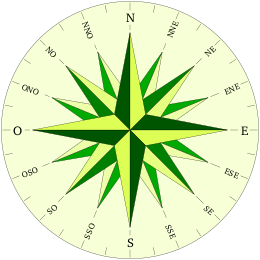


Figura 1. Rosa de los vientos.

# Ejercicio 1

En el ejercicio 1, como se ha comentado en la introducción de la memoria de la práctica, se programa la lectura de los sensores necesarios del *Sensors BoosterPack*. Previamente a este se lleva a cabo la comprobación del boosterpack, determinando su posición en la placa y configurándolo de acorde a ella. El código está basado en el ejemplo 8 de la asignatura, con la diferencia de que solo se usa el magnetómetro y de que la información se actualiza en la pantalla cada medio segundo.

La conversión de la lectura en ángulo se realiza con la arcotangente, con especial cuidado ante casos concretos como el de x = 0 que, al hacer la división y/x resultaría infinito. Una vez calculada la arcotangente, se calcula el cuadrante y se le suma el ángulo correspondiente para llevar el resultado hasta él.

La programación de la UART se lleva a cabo del mismo modo que en la práctica anterior. Esta vez no se usan los modos de bajo consumo, así que la información se muestra por pantalla cada vez que la interrupción del timer 0 pone una variable auxiliar a 1. El código está comentado línea por línea:

**#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 apartado, respecto al código del ejercicio 1, solo hay que añadir unas pocas líneas para calcular y mostrar el rumbo según indica la rosa de los vientos.

Para facilitar la tarea, se define un puntero a vector de tipo char en el que están almacenados de forma ordenada los nombres de los rumbos.

El cálculo del sector se lleva a cabo mediante una simple división con un offset, con una atención especial al sector que apunta al norte. El código está comentado línea por línea:

**#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;

}

# Ejercicio 3

En el ejercicio 3, se emplea un motor paso a paso para la implementación de la brújula. Es necesario un timer extra, que se encargará de interrumpir para dar los pasos del motor. Estas interrupciones se producirán cada 5 ms, que es el tiempo mínimo que permite el motor entre pasos, y moverán al motor si aún le quedan pasos por dar para alcanzar la referencia marcada por el magnetómetro.

Cabe destacar que el motor realmente no apunta al norte, ya que no tiene encoder absoluto. El norte será la posición inicial del motor cada vez que la placa se encienda o resetee. El motor irá montado pues encima de la placa, moviéndose solidario a ella, y girando para apuntar siempre a ese norte.

Con cada lectura del magnetómetro se calcula una nueva referencia, que es comparada con la variable que almacena la posición del motor. A partir de estas dos variables se calcula el sentido de giro más corto para alcanzar la referencia y se comienza el giro. En el código está comentado el procedimiento:

**#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**);

**void** **TimerStepper**(**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;

// Variables para el stepper

**volatile** **int** referencia, posicion = 0, placa;

**volatile** **int** mov = 0, secuencia = 0, sentido = 0;

// 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"};

/\*

\* Puertos, pines y secuencias de activacion de las fases del motor paso a paso

\* que permiten simplificar cada paso a un bucle for que recorra estos vectores

\*/

uint32\_t Puerto[]={

GPIO\_PORTF\_BASE,

GPIO\_PORTF\_BASE,

GPIO\_PORTF\_BASE,

GPIO\_PORTG\_BASE,

};

uint32\_t Pin[]={

GPIO\_PIN\_1,

GPIO\_PIN\_2,

GPIO\_PIN\_3,

GPIO\_PIN\_0,

};

**int** Step[4][4]={1,0,0,0,

0,0,0,1,

0,0,1,0,

0,1,0,0

};

**#define** FREC 200 //Frecuencia en hercios del tren de pulsos: 5ms

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

{

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

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

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

// Configuracion del reloj para el stepper. Interrupciones cada 5ms (menos no permite el motor)

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1); //Habilita T1

**TimerClockSourceSet**(TIMER1\_BASE, TIMER\_CLOCK\_SYSTEM); //T0 a 120MHz

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); //T1 periodico y conjunto (32b)

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, (RELOJ/FREC)-1);

**TimerIntRegister**(TIMER1\_BASE,TIMER\_A,TimerStepper);

**IntEnable**(INT\_TIMER1A); //Habilitar las interrupciones globales de los timers

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); // Habilitar las interrupciones de timeout

**TimerEnable**(TIMER1\_BASE, TIMER\_A); //Habilitar Timer0, 1, 2A y 2B

// Configuracion de los pines a los que va conectado el motor paso a paso

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);

**GPIOPinTypeGPIOOutput**(GPIO\_PORTG\_BASE, GPIO\_PIN\_0);

// Timer para los sensores. Cambia el valor que ve el stepper y que se muestra 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 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);

}

// Calculamos la referencia a seguir por el stepper

placa = **round**(grados\*514.0/360.0); // Angulo de la placa en pasos

referencia = **round**((360-grados)\*514.0/360.0); // Referencia en pasos para el motor

}

**return** 0;

}

/\*

\* En la rutina de interrupcion del timer0 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;

}

// Rutina de interrupcion del timer1, el del stepper (cada 5 ms)

**void** **TimerStepper**(**void**)

{

**int** i;

// Si la referencia ha cambiado, y el motor se tiene que desplazar

**if**(referencia - posicion)

{

mov = 1; // Activamos el movimiento

/\*

\* Calculamos el sentido de giro en funcion del error (referencia - posicion)

\* Sentido = 1: giro antihorario

\* Sentido = 0: giro horario

\*/

**if**((referencia - posicion) > 0) sentido = 1;

**else** sentido = 0;

/\*

\* Si la distancia es más de media vuelta, el sentido de giro debe cambiar para

\* moverse hacia la referencia por el camino mas corto

\*/

**if**(abs(referencia - posicion) > 257) sentido ^= 1;

}

**if**(mov == 1)

{

// Si hay que girar positivo, secuencia --

// Si hay que girar negativo, secuencia ++

**if**(sentido)

{

secuencia --;

//Actualizamos el valor de la posicion a cada paso que damos

posicion ++;

**if**(posicion == 514) posicion = 0;

}

**else**

{

secuencia ++;

// Actualizamos el valor de las posicion a cada paso que damos

posicion --;

**if**(posicion == -1) posicion = 513;

}

// Saturaciones de la variable secuencia

**if**(secuencia == 4) secuencia = 0;

**if**(secuencia == -1) secuencia = 3;

// Damos el paso siguiente

**for**(i = 0; i < 4; i ++)

**GPIOPinWrite**(Puerto[i],Pin[i],Pin[i]\*Step[secuencia][i]);

// Dejamos de movernos si hemos dado los pasos necesarios

**if**(referencia == posicion) mov = 0;

}

//Borramos el flag de interrupcion antes de salir

**TimerIntClear**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

}

# Comentarios acerca de la práctica

En cuanto a la realización de la práctica, solo se ha tenido un contratiempo, y es que el *sensors boosterpack* da error de vez en cuando al comprobar si está disponible para su lectura. Este problema no tiene relación alguna con la programación empleada, y parecía solucionarse reseteando varias veces la placa y conectándola de nuevo al usb.

También cabe mencionar un comportamiento curioso del conjunto en el ejercicio 3. Durante el desarrollo de la sesión práctica en el laboratorio se nos especificó que no hiciéramos una brújula, sino que hiciéramos una especie de seguidor. El motor desde la mesa debía replicar los movimientos de la placa.

Nosotros hemos hecho una brújula, pero si desacoplas el motor de la placa, lo dejas en la mesa, y giras la placa boca abajo, el comportamiento pasa a ser el del seguidor pedido.