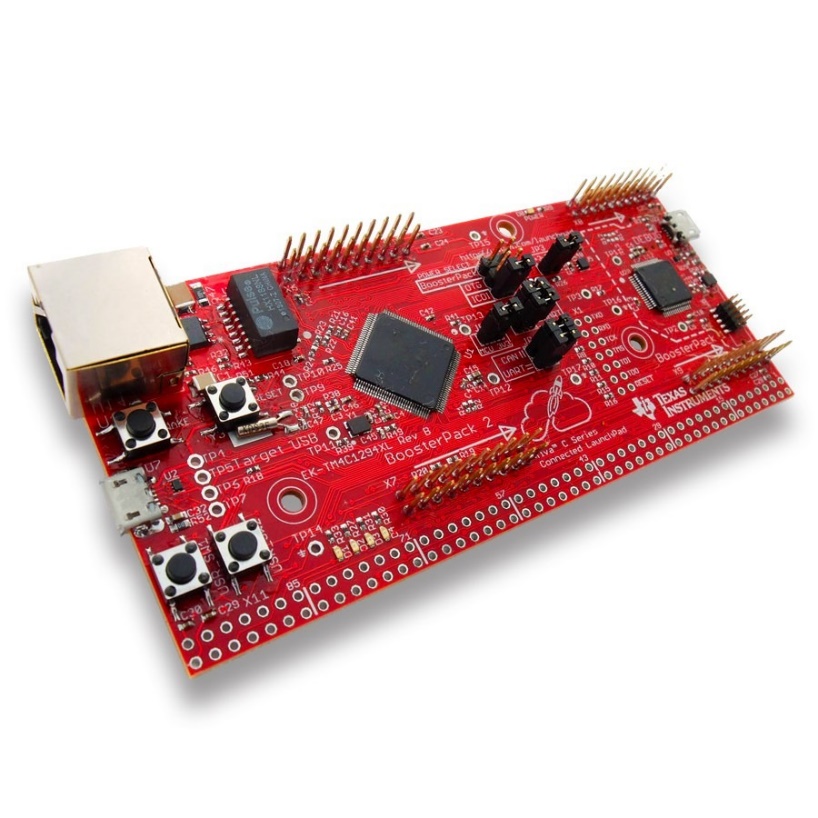
Sistemas electrónicos para Automatización

Práctica 5



Álvaro Calvo Matos

Damián Jesús Pérez Morales

Índice

[1. Introducción 2](#_Toc24390275)

[2. Ejercicio 1 3](#_Toc24390276)

[3. Ejercicio 2 4](#_Toc24390277)

[4. Comentarios acerca de la práctica 12](#_Toc24390278)

# Introducción

En la quinta práctica de la asignatura, se trata de manejar el entorno de GUI Composer a través de la página web de *Texas Instruments* para mostrar información desde el GUI sobre la situación de la placa a partir de un programa “xxx.out”.

En primer lugar, se hará usodel *Ejemplo 8* de la asignatura a modo de corroborar que todas las conexiones funcionan correctamente.

En segundo lugar, se pretende que desde la GUI se haga encender los LEDs de la placa y, además, se lleve la información del magnetómetro para representarlo a través del GUI en forma de brújula (código rescatado de la práctica 3) y del acelerómetro para representarlo desde el GUI con un plot de los tres ejes.

# 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 que se visualiza desde la computadora y en el que el microcontrolador pueda mandarle información al interfaz para poder representar información en este caso sobre la temperatura, humedad y presión que está contemplando el microcontrolador. Se muestra a continuación un resultado de cómo ha quedado la interfaz:

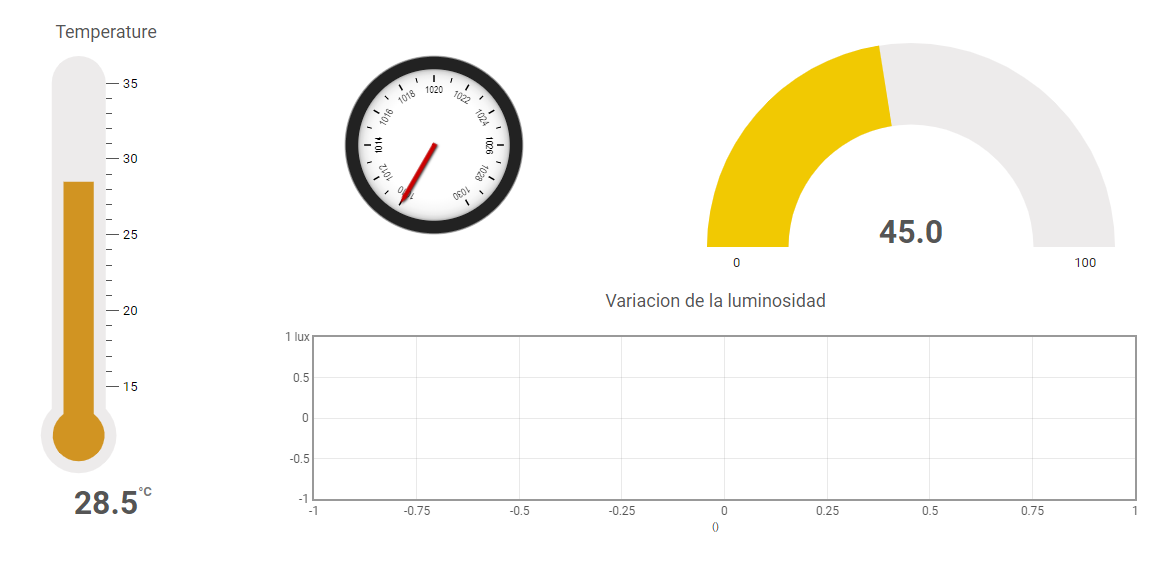


Ilustración 1. Resultado de la interfaz del apartado 1 (no se muestra la luminosidad porque se ha rescatado la interfaz sin tener el microcontrolador presente)

Ya que el código empleado es el del *Ejemplo 8*, no se adjuntará el código.

Los resultados están presentes en el comprimido “gui\_apartado1.zip”.

# Ejercicio 2

En este ejercicio se pretende hacer interactuar desde el GUI con el microcontrolador, haciendo que desde el GUI se puedan encender los LEDs de la placa y esta manda datos acerca del acelerómetro y del campo magnético incidente sobre la misma, con la misión de hacer ver desde la interfaz una gráfica donde se represente una brújula funcional y una gráfica con la evolución del acelerómetro. La interfaz:

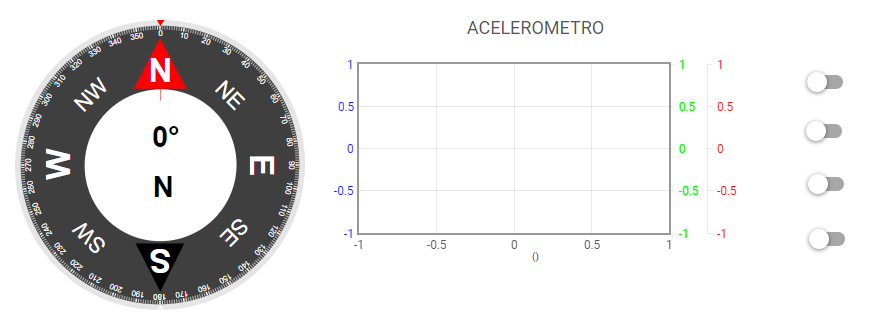
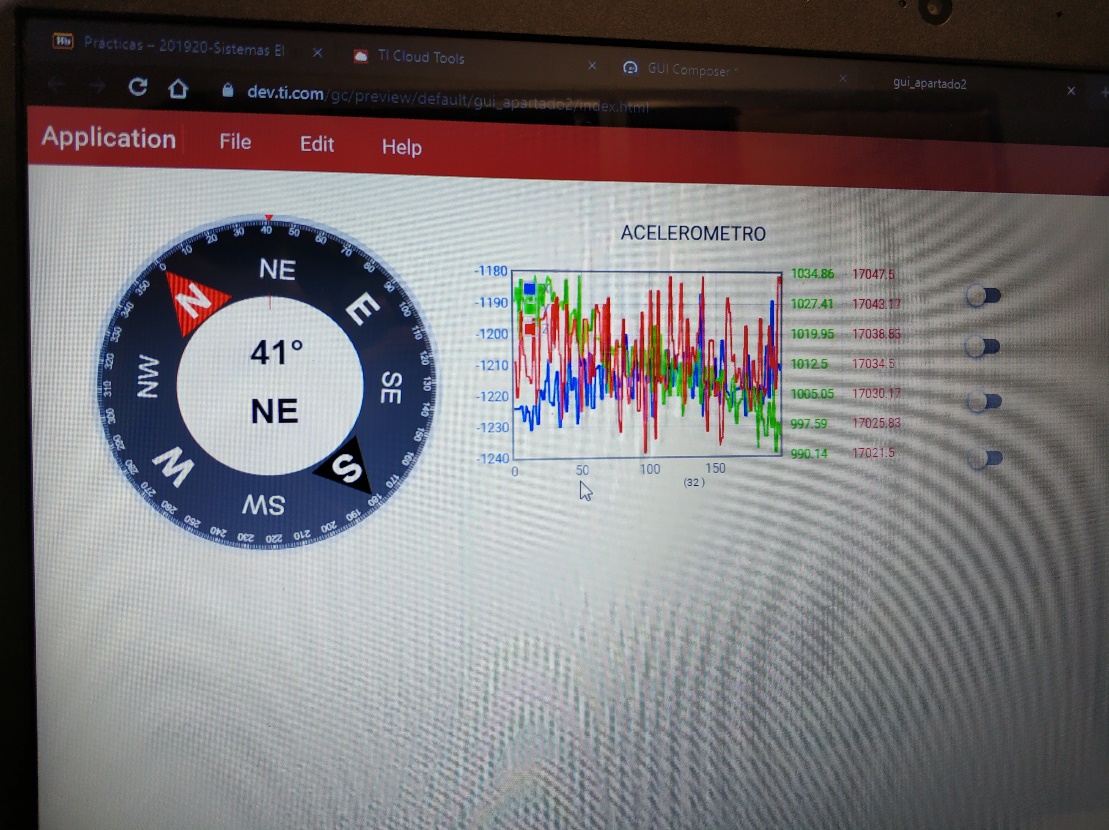
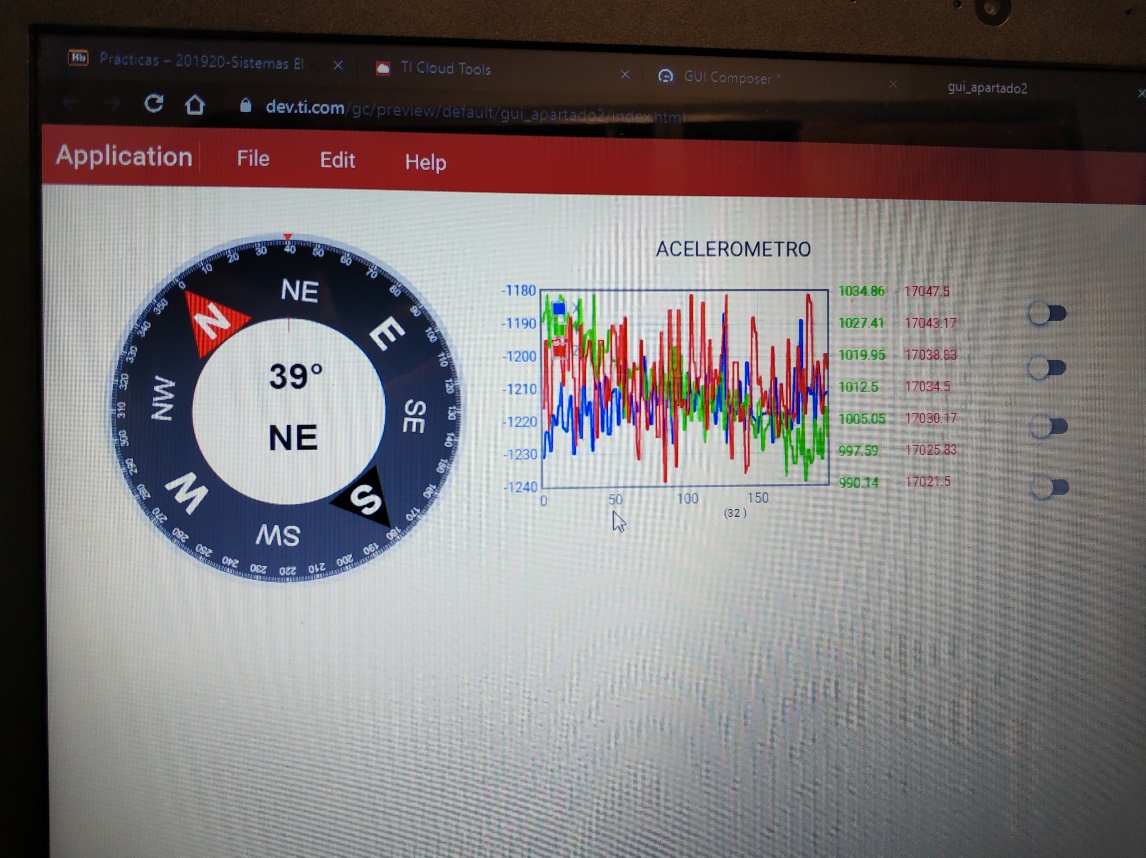


Ilustración 2. Interfaz (sin simular) del apartado 2

La interfaz resultante en funcionamiento es la siguiente:

Ilustración 3. Resultados del apartado 2 en funcionamiento



Los resultados están presentes en el archivo “gui\_apartado2.zip”.

El código del funcionamiento es el siguiente:

/\*

\* Adaptacion del ejemplo 8 junto con el ultimo apartado

\* de la tercera practica de la asignatura.

\*

\* Esta modificacion no implica a penas lineas de codigo

\* adicionales, solo unir trozos para posteriormente realizar

\* una interfaz grafica en gui composer con la cual visualizarlos

\*/

**#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"

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

// Function Declarations

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

**int** RELOJ;

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

**char** Cambia=0;

**float** lux;

**char** string[80];

**int** DevID=0;

int16\_t T\_amb, T\_obj;

**float** Tf\_obj, Tf\_amb;

**int** lux\_i, T\_amb\_i, T\_obj\_i;

// BME280

**int** returnRslt;

**int** g\_s32ActualTemp = 0;

**unsigned** **int** g\_u32ActualPress = 0;

**unsigned** **int** g\_u32ActualHumity = 0;

// struct bme280\_t bme280;

// BMI160/BMM150

int8\_t returnValue;

**struct** bmi160\_gyro\_t s\_gyroXYZ;

**struct** bmi160\_accel\_t s\_accelXYZ;

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

//Calibration off-sets

int8\_t accel\_off\_x;

int8\_t accel\_off\_y;

int8\_t accel\_off\_z;

int16\_t gyro\_off\_x;

int16\_t gyro\_off\_y;

int16\_t gyro\_off\_z;

**float** T\_act,P\_act,H\_act;

bool BME\_on = true;

**int** T\_uncomp,T\_comp;

**char** mode;

**long** **int** inicio, tiempo;

**volatile** **long** **int** ticks=0;

uint8\_t Sensor\_OK=0;

**#define** BP 2

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

/\* VARIABLES BOOLEANAS PARA CONTROLAR LOS LEDS DESDE GUI COMPOSER \*/

bool enciende1 = false;

bool enciende2 = false;

bool enciende3 = false;

bool enciende4 = false;

**int** AX, AY, AZ;

/\* VARIABLES DE LA PRACTICA 3 PARA LA BRUJULA \*/

**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"};

**void** **IntTick**(**void**){

ticks++;

}

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

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

// Configuracion de los perifericos a usar

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

**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

// Configuracion del timer0

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

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

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

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, RELOJ/4 -1);

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

**IntEnable**(INT\_TIMER0A);

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

**IntMasterEnable**();

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

**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);

// Identificacion del estado del sensors booster pack

**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;

}

**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;

}

**SysTickIntRegister**(IntTick);

**SysTickPeriodSet**(12000);

**SysTickIntEnable**();

**SysTickEnable**();

**while**(1)

{

// Control de los leds desde la interfac grafica

**if**(enciende1) **GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_PIN\_0);

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

**if**(enciende2) **GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4, GPIO\_PIN\_4);

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

**if**(enciende3) **GPIOPinWrite**(GPIO\_PORTN\_BASE, GPIO\_PIN\_0, GPIO\_PIN\_0);

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

**if**(enciende4) **GPIOPinWrite**(GPIO\_PORTN\_BASE, GPIO\_PIN\_1, GPIO\_PIN\_1);

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

// Lectura de los sensores cada vez que el timer lo indique

**if**(Cambia==1){

Cambia=0;

inicio=ticks;

**if**(Opt\_OK)

{

lux=OPT3001\_getLux();

lux\_i=(**int**)**round**(lux);

}

**if**(Tmp\_OK)

{

sensorTmp007Read(&T\_amb, &T\_obj);

sensorTmp007Convert(T\_amb, T\_obj, &Tf\_obj, &Tf\_amb);

T\_amb\_i=(**short**)**round**(Tf\_amb);

T\_obj\_i=(**short**)**round**(Tf\_obj);

}

**if**(Bme\_OK)

{

returnRslt = bme280\_read\_pressure\_temperature\_humidity(

&g\_u32ActualPress, &g\_s32ActualTemp, &g\_u32ActualHumity);

T\_act=(**float**)g\_s32ActualTemp/100.0;

P\_act=(**float**)g\_u32ActualPress/100.0;

H\_act=(**float**)g\_u32ActualHumity/1000.0;

}

**if**(Bmi\_OK)

{

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

bmi160\_read\_accel\_xyz(&s\_accelXYZ);

bmi160\_read\_gyro\_xyz(&s\_gyroXYZ);

AX = s\_accelXYZ.x;

AY = s\_accelXYZ.y;

AZ = s\_accelXYZ.z;

}

// CODIGO DE LA BRUJULA DE LA PRACTICA 3

// 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

tiempo=ticks;

// Informacion de los sensores mostrada por pantalla (codigo del ejemplo 8)

**UARTprintf**("\033[10;1H---------------------------------------\n");

**sprintf**(string," OPT3001: %.3f Lux\n",lux);

**UARTprintf**(string);

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

**sprintf**(string," TMP007: T\_a:%.3f, T\_o:%.3f \n", Tf\_amb, Tf\_obj);

**UARTprintf**(string);

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

**sprintf**(string, " BME: T:%.2f C P:%.2fmbar H:%.3f \n",T\_act,P\_act,H\_act);

**UARTprintf**(string);

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

**sprintf**(string, " BMM: X:%6d\033[17;22HY:%6d\033[17;35HZ:%6d \n",s\_magcompXYZ.x,s\_magcompXYZ.y,s\_magcompXYZ.z);

**UARTprintf**(string);

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

**sprintf**(string, " ACCL: X:%6d\033[19;22HY:%6d\033[19;35HZ:%6d \n",s\_accelXYZ.x,s\_accelXYZ.y,s\_accelXYZ.z);

**UARTprintf**(string);

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

**sprintf**(string, " GYRO: X:%6d\033[21;22HY:%6d\033[21;35HZ:%6d \n",s\_gyroXYZ.x,s\_gyroXYZ.y,s\_gyroXYZ.z);

**UARTprintf**(string);

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

tiempo=(tiempo-inicio);

**sprintf**(string, "TConv: %d (0.1ms)",tiempo);

**UARTprintf**(string);

}

}

**return** 0;

}

// En la rutina de interrupcion solo se activa una variable para proceder a leer los sensores

**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, no se ha tenido ningún inconveniente, ya que se ha podido realizar a tiempo en la misma práctica presencial y no nos ha supuesto ningún problema.