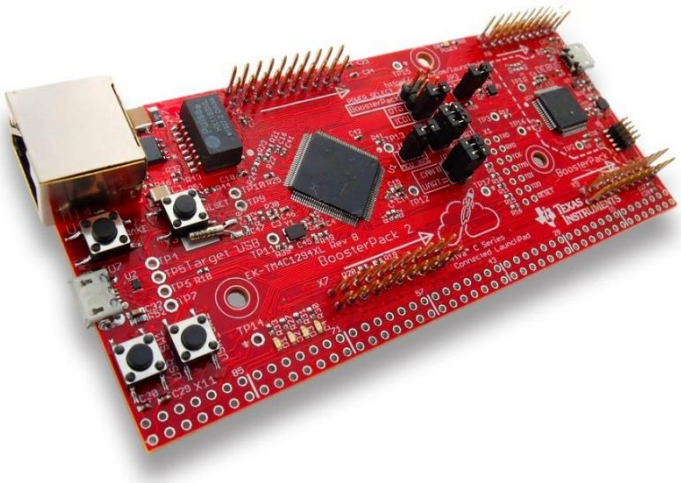

SISTEMAS ELECTRÓNICOS PARA AUTOMATIZACIÓN

Práctica 5

Álvaro Calvo Matos
Damián Jesús Pérez Morales

Índice

1. 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.

2. 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:

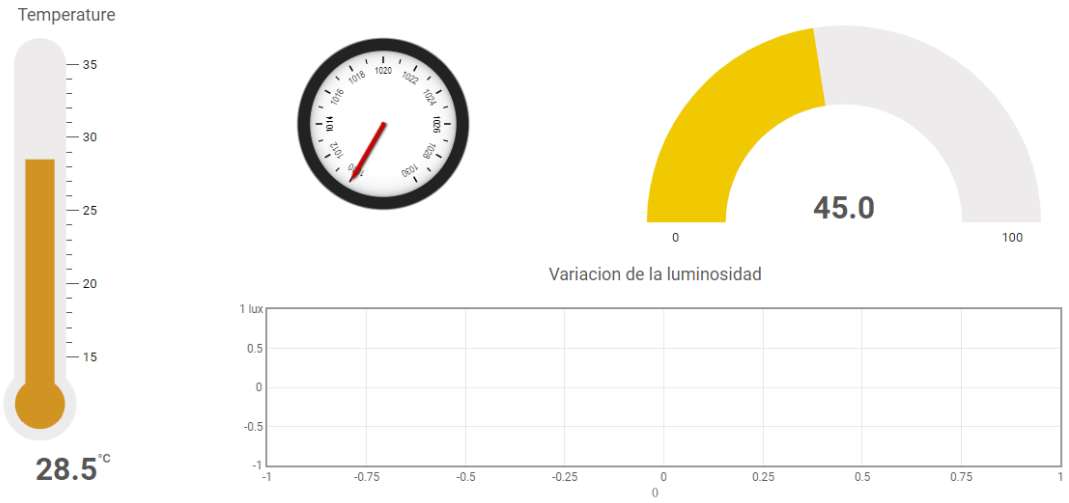

Ilustración 1. Resultado de la interfaz del apartado 1 (no se muestra la luminosidad porque se ha rescatado la interfaz sin tener el microcontrolador presente)

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”.

3. 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:

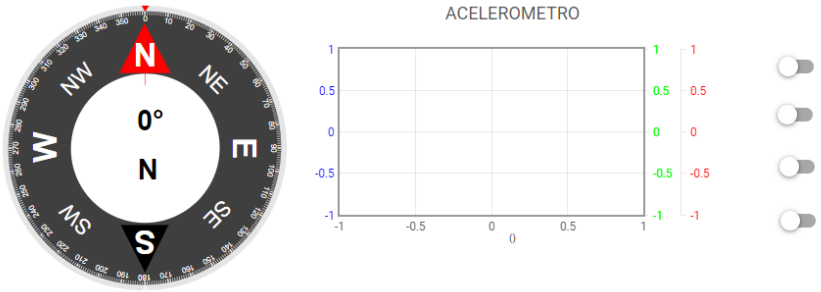

Ilustración 2. Interfaz (sin simular) del apartado 2

La interfaz resultante en funcionamiento es la siguiente:

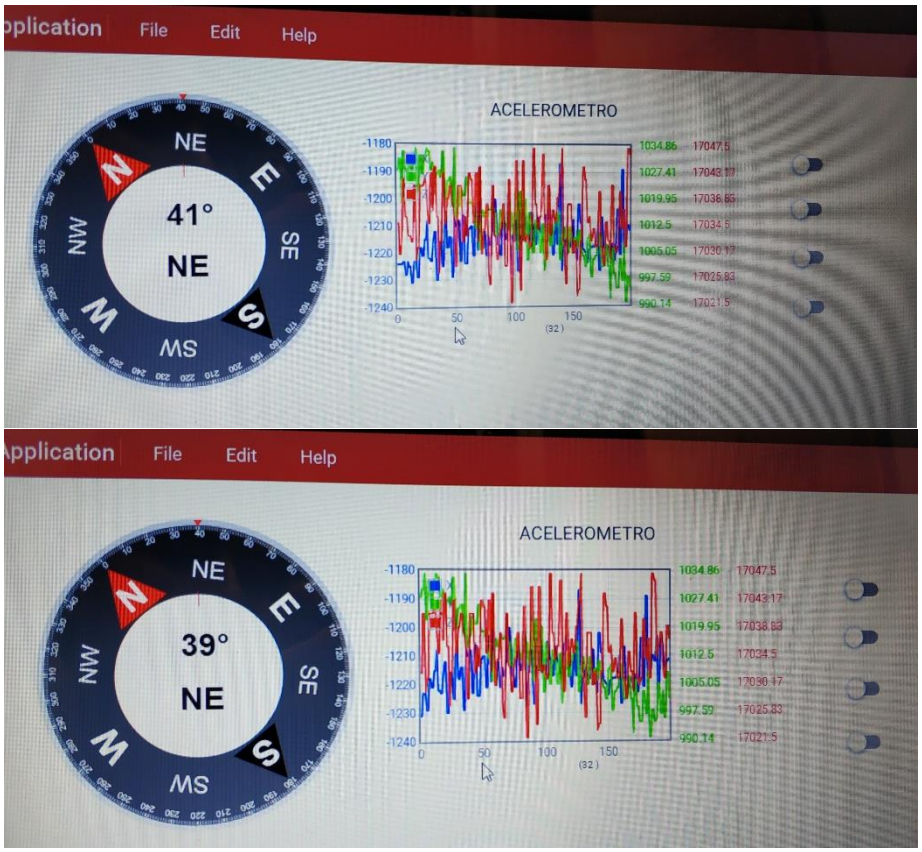

Ilustración 3. Resultados del apartado 2 en funcionamiento

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;
}
```

4. 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.