Sistemas Embebidos - Trabajo Práctico N° 2
Entradas y Salidas Digitales de Propósito General (GPIO)

Objetivos:

- Uso del IDE (edición, compilación y depuración de programas)
- Uso de GPIO (manejo de Salidas y de Entradas Digitales)
- Documentar lo que se solicita en c/ítems

Referencias (descargar del Campus Vitual del curso a fin de usarlas durante la realización del TP):

- LPCXpresso-Intro: http://campus.fi.uba.ar/pluginfile.php/155949/mod_resource/content/4/Sistemas_Embebidos-2016_2doC-LPCXpresso-Intro-Cruz.pdf
- LPCXpresso-Salidas: http://campus.fi.uba.ar/pluginfile.php/156011/mod_resource/content/4/Sistemas_Embebidos-2016_2doC-LPCXpresso-Salidas-Cruz.pdf
- LPCXpresso-Entradas: http://campus.fi.uba.ar/pluginfile.php/156013/mod_resource/content/4/Sistemas_Embebidos-2016_2doC-LPCXpresso-Entradas-Cruz.pdf
- LPCXpresso-Systick: http://campus.fi.uba.ar/pluginfile.php/156031/mod_resource/content/5/Sistemas_Embebidos-2016_2doC-LPCXpresso-Systick-Cruz.pdf
- LPC435X_3X_2X_1X Product Data Sheet: http://campus.fi.uba.ar/mod/resource/view.php?id=28519
- LPC43XX User Manual (Chapter 1, 18 & 19):http://campus.fi.uba.ar/mod/resource/view.php?id=77765
- EDU-CIAA-NXP (web site): http://proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:edu-ciaa:edu-ciaa-nxp
- EDU-CIAA-NXP (esquemático): http://www.proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=desarrollo:edu-ciaa:edu-ciaa-nxp:edu-ciaa-nxp_color.pdf
- EDU-CIAA-NXP (pinout): http://proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=desarrollo:edu-ciaa:edu-ciaa-nxp_pinout_a4_v4r2_es.pdf

1. Manejo de salidas y entradas digitales

El objetivo a continuación es crear las funciones **gpioInit**, **gpioWrite** y **gpioRead**, las cuales implementan completamente el uso de los GPIOs de la placa en una manera simple y general:

- **gpioInit**: Inicializa el pin utilizado como GPIO. En la familia de chips LPC43xx esta instrucción consiste en dos partes:
  - Configurar eléctricamente el pin a utilizar a través del periférico System Control Unit (SCU).
  - Configurar el modo de uso del GPIO a través de sus propios registros en el memory mapping.
- **gpioWrite**: Escribe un valor binario en el pin del GPIO cuando está configurado como salida.
- **gpioRead**: Lee el valor del pin utilizado como GPIO

a. Dentro de la carpeta “projects” del repositorio grupal, **crear un directorio TP2 que contenga las carpetas inc y src, y el archivo config.mk** que permita compilar correctamente un programa para cargar a la placa.

b. Para implementar correctamente estas funciones, primero hay que hacer varias cosas, enumeradas a continuación. Cada una de las siguientes instrucciones deben estar correctamente situadas dentro de los archivos contenidos en la carpetas inc y src, teniendo en cuenta lo estudiado anteriormente sobre modularización y OOP.

   i. **Identificar las estructuras** que representan los periféricos SCU y GPIO y las funciones implementadas para manejarlas, provistos por el fabricante. En el caso de la EDU-CIAA, las librerías del chip LPC4337 se encuentran en la carpeta libs/lpcopen/lpc_chip_43xx del repositorio grupal.

ii. Enumerar los diferentes tipos de configuraciones que puede adoptar el pin GPIO (output, pullup, etc.) a partir de la información provista por la hoja de datos, y **crear un tipo enumerativo gpioInit_t** que contenga los nombres de cada uno de ellos.

iii. **Crear una estructura** que contenga los siguientes campos:
   1. el puerto y el pin a configurar por el SCU,
   2. la función para que el pin anterior se configure como GPIO, y
   3. el puerto y el pin del GPIO utilizado

iv. Luego, **crear un vector global de estas estructuras** que contenga todas las posibles configuraciones que pueden adoptar los GPIOs que quiero configurar. Por ejemplo, si tengo que configurar un conjunto de leds (LED1, LED2 y LED3), voy a crear un vector gpioPinsInit de (en este caso 3) estructuras que contengan la información de configuración de cada led. Es decir, que sí el LED1 se conecta al puerto 2, pin 10 del SCU, la función para configurarlo como GPIO tiene un valor igual a 0, y el GPIO configurado es el GPIO0[14], entonces la configuración correspondiente al LED1 en este vector tiene la forma: const conf_t gpioPinsInit[]{ // ... { 2,10, 0, 0,14 }, // Configuración del LED1: { P2_10, 0, GPIO0[14] } // ... } donde conf_t es una estructura (la del punto 3) de la siguiente forma: typedef struct { int8_t scu_port; int8_t scu_pin; int8_t func; int8_t gpio_port; int8_t gpio_pin; } conf_t;

v. Dado que resultará muy conveniente posteriormente, se pide **crear un tipo enumerativo gpioMap_t** con todos los posibles GPIOs (en el ejemplo anterior serían LED1, LED2, LED3).

c. Con todo esto, ya es posible hacer las funciones gpioInit, gpioWrite y gpioRead:
   i. void gpioInit(gpioMap_t pin, gpioInit_t conf);
   ii. void gpioWrite(gpioMap_t pin, bool_t value);
   iii. void gpioRead(gpioMap_t pin);

   Dentro de cada una de ellas se pretende que se utilicen las funciones de la librería Chip de LPCOpen analizadas en el punto 1.b.i.

2. Testeo de las funciones

Para corroborar que las funciones del inciso anterior hayan sido correctamente implementadas, se pide cargar el proyecto **application** en la carpeta **projects** del repositorio grupal e incluir en ella los archivos editados en el punto anterior, según corresponda. Luego, se pide compilar el proyecto y descargarlo a la placa completando en el archivo **main.c** del mismo la inicialización de las teclas y los LEDs como GPIOs de entradas y salidas, respectivamente.

3. Implementación para el chip utilizado en el TPF.

a. Luego de haber completado los pasos del ítem anterior, se pretende que estas funciones sirvan para la implementación del trabajo final. En el caso de no utilizar un chip de la familia LPC43xx o equivalentes, va a ser necesario buscar las funciones provistas por el fabricante correspondiente y realizar un trabajo similar al del apartado anterior.
b. Implementar un prototipo del diagrama de estados del Trabajo Práctico Final que pueda ser descargado y simulado en la placa EDU-CIAA-NXP, reemplazando las funciones utilizadas en ese TP para manejar los GPIO (ej, Board_LED_Toggle, Board_LED_set, etc.) por las implementadas en este TP.
c. Cargar la librería sAPI del repositorio firmware_v3 en la carpeta libs del repositorio grupal, que será de utilidad para los TPs siguientes.