Uso de registros - bloque Syscon y GPIO:

1) Realizar la función SetPINMODE_IN y MySetDIR. Las mismas deberán recibir el puerto y el pin a configurar, y un tercer parámetro que será, para el caso de la funcion SetPINMODE_IN, el modo a configurar para ese pin (R_pullup, R_pulldown, repeater mode o nada, con los números correspondietes según el manual de usuario del microcontrolador), y en el caso de SetDIR, la dirección de un pin ya configurado como GPIO. Incluir el archivo LPC845.h para realizar las funciones (no hace falta volver a declarar los #defines que ya existen)

   Nota: Recordar habilitar el clock de los bloques gpio y syscon al momento de realizar las configuraciones mediante el registro SYSAHBCLKCTRL. No es necesario deshabilitar el mismo al finalizar la configuración.

2) Realizar las funciones uint8_t GetPIN(uint8_t puerto, uint8_t pin) y SetPIN (uint8_t puerto, uint8_t pin). Las mismas deberán devolver el estado lógico del pin (en el caso de GetPIN) o establecer el mismo (en el caso de SetPIN).

   Probar el correcto funcionamiento de los dos primeros puntos a partir del siguiente código:

```
#include “LPC845.h”

void main ( void )
{
	InitHw();

	//Configuración de los puertos del led RGB y del pulsador del
	stick como GPIO:

	SetPINMODE_IN(0,4,NONE); //Pin del PULSADOR sin resistencia

	SetDIR(1,0,SALIDA); //Pin ROJO como SALIDA
	SetDIR(1,1,SALIDA); //Pin VERDE como SALIDA
	SetDIR(1,2,SALIDA); //Pin AZUL como SALIDA

	SetDIR(0,4,ENTRADA); //Pin del PULSADOR como ENTRADA

	while ( 1 )
	{
		//Mientras esté presionado el pulsador
		if ( GetPIN(0,4,1) == 0 )
		{
			//Prendo los 3 leds
			SetPIN(1,0,0);
			SetPIN(1,1,0);
```

```
                SetPIN(1,2,0);
            }
            //Si no está presionado:
            else
            {
                //Los apago
                SetPIN(1,0,1);
                SetPIN(1,1,1);
                SetPIN(1,2,1);
            }
        }
}
```

Diagramación en capas:

3) Dividir el programa anterior haciendo uso de la diagramación en capas, y desarrollar para tal fin las primitivas LedsRGBStick(uint8_t color, uint8_t estado) para prender o apagar cada uno de los leds, y la función uint8_t GetKeyStick(), que devuelve un 1 si el pulsador está presionado o un 0 si está suelto. Generar una carpeta para cada una de las capas (aplicación, primitivas y drivers) y dividir los .c en las mismas. El programa anterior deberá quedar como sigue:

```
#include “Drivers.h”
#include “Primitivas.h”

void main ( void )
{
      InitHw();

      //Configuración de los puertos del led RGB y del pulsador del
      stick como GPIO:

      InicializarRGBStick();
      InicializarPulsadorStick();

      while ( 1 )
      {
            //Mientras esté presionado el pulsador
            if ( GetKeyStick() )
            {
                  //Prendo el led rojo
                  LedsRGBStick(ROJO,ON);
            }
            //Si no está presionado:
            else
            {
```

```
                    //Apago el led rojo
                    LedsRGBStick(ROJO,OFF);
                }
        }
}
```

Interrupciones:

4) A partir del ejemplo anterior, modificar la configuración del Pulsador para que genere una interrupción cada vez que se presione el mismo (configurado como interrupción externa). Realizar una rutina de atención a dicha interrupción que levante un flag en forma global cada vez que se detecta la opresión, y una máquina de estados en el bucle principal, que cada vez que este flag se levanta, modifique el color del led RGB. Mantener la división del programa en aplicación, primitvas y drivers.

Timers - Systick:

5) Generar la función de inicialización del Systick InitSystick(), que deberá configurar el periférico para que genere una interrupción periódica a intervalos de 1mseg. Desarrollar la función de atención a esta interrupción (buscar su nombre en el vector de interrupciones para su desarrollo). La función deberá únicamente incrementar una variable global, levantar un flag cuando la misma cuente hasta 1000, y volver a empezar desde cero. La aplicación (en el bucle principal) deberá usar ahora este flag para cambiar el color del led RGB, tal como se hacía en el ejemplo anterior. Si todo funciona bien, el led debería cambiar de color una vez por segundo.

6) A partir de las funciones realizadas en los puntos 4 y 5, realizar un programa que cada vez que reciba una interrupción externa, comience a cambiar el color del led RGB a intervalos de un segundo, y cuando vuelva a recibirse la interrupción, detenga el cambio de color. Realizar esta aplicación diseñándola como una máquina de estados, y utilizando las primitivas desarrolladas en los puntos anteriores para su control (en caso de que no se hayan hecho primitivas en el punto 3 y 4, realizarlas)