



Facultad de Ciencias  
Físico Matemáticas  
y Naturales



Universidad  
Nacional de  
San Luis

# ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

## ARQUITECTURA DE SISTEMAS EMBEBIDOS

Profesor Responsable: Dr. Mario Marcelo Berón

Colaboradores: Ing. Martín Murdocca, Ing. Alejandro Nuñez Manquez

Año: 2017

# Laboratorio 1:

## Introducción a LPCXpresso usando la librería CMCIS y la placa LPC1769

## Índice

<b>1. Introducción</b>	<b>2</b>
1.1. Material de Referencia . . . . .	2
1.2. Listado de Materiales . . . . .	3
<b>2. LPC1769 LPCXPRESSO BOARD</b>	<b>3</b>
<b>3. Kit Cortex Base Board</b>	<b>3</b>
<b>4. Ambiente de Desarrollo Integrado LPCXpresso</b>	<b>4</b>
4.1. Cortex Microcontroller Software Interface Standard (CMSIS) . . . . .	4
<b>5. Primeros pasos</b>	<b>5</b>
5.0.1. Importar librería CMSIS . . . . .	5
5.1. Cómo crear un proyecto . . . . .	5
5.2. Compilación del proyecto . . . . .	8
5.3. Implementación del proyecto . . . . .	8
5.4. Como crear una librería de usuario . . . . .	9
<b>6. Ejercicios</b>	<b>10</b>

## 1. Introducción

Al final del laboratorio tendrá los conocimientos básicos para el manejo de la herramienta de desarrollo que se utilizará para trabajar con los microcontroladores de 32 bits. Obtendrá los conocimientos necesarios para manejar las entradas y salidas de propósito general (GPIO) del microcontrolador LPC1769. Además, será capaz de modularizar códigos en C para crear sus propias bibliotecas (library).

### 1.1. Material de Referencia

- Manual de Usuario del LPC1769.
- Hoja de Datos del LPC1769.
- Esquemático del LPC1769.
- Esquemático del Kit Cortex Base Board.
- Guía de Usuario del LPCXpresso.
- Guía de Instalación del LPCXpresso.

## 1.2. Listado de Materiales

- PC con entorno LPCXpresso V 8.2.2
- Stick LPC1769
- Kit Cortex Base Board

## 2. LPC1769 LPCXPRESSO BOARD

El LPC1769 LPCXPRESSO BOARD (Figura 1) es una placa desarrollada por la empresa Embedded Artists[4] basado en el microcontrolador ARM Cortex-M3 LPC1769[8] para aplicaciones embebidas que ofrecen un alto nivel de integración, con un consumo de potencia bajo. La frecuencia máxima del CPU es de 120MHz. Incorpora un pipeline de tres etapas y su arquitectura es Harvard. Posee entradas/salidas de propósito general, un ADC de 12 bits, un DAC de 10bits, cuatro timers/contadores de propósito general, PWM, etc.

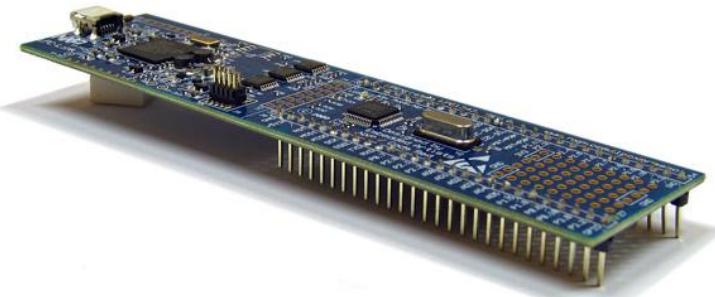


Figura 1: Stick LPC1769

El pinout del stick se puede ver en su esquemático[3] que ha sido adjuntado al material dado en el laboratorio.

## 3. Kit Cortex Base Board

El Kit Cortex Base Board (Figura 2) es una placa de desarrollo de la empresa R2M Ingenieria[6] que posibilita rápidamente implementar los proyectos que se realizarán en los laboratorios de la materia. Este kit es compatible con LPC176X y LPC134X y posee las siguientes características:

- USB host y client.
- Conector Ethernet RJ45 con magnetics incluidos.
- UART0 mediante bridge FTDI en conector USB “B”.
- UART1 con niveles EIA232 en conector DB9.
- UART3 con niveles TTL.
- LED RGB
- Ocho LED para propósitos generales.
- Conector de expansión con 10 líneas de GPIO.
- Dos pulsadores para propósito general.
- Pulsador Reset.
- Preset de entrada analógica.

- Zocalo para memoria SD.
- LCD alfanumérico de 2x16.



Figura 2: Kit Cortex Base Board

El pinout del *Kit Cortex Base Board* se puede ver en su esquemático[5] que ha sido adjuntado al material dado en el laboratorio.

## 4. Ambiente de Desarrollo Integrado LPCXpresso

LPCXpresso es una plataforma de desarrollo de bajo costo, habilitada directamente desde NXP, que provee un camino rápido para desarrollar aplicaciones avanzadas usando los microcontroladores de baja potencia y altamente eficiente de NXP. Esta herramienta está basada en Eclipse y posee una edición gratuita que soporta códigos de hasta 256kb. Además, permite programar tanto en C como en C++. Para instalar esta IDE hay que descargarla de la página de NXP[8]. Para la descarga hay que registrarse en la página de NXP. Una vez instalada la IDE se necesita una licencia free que se puede obtener desde la misma página ingresando con el usuario y la contraseña obtenida.

### 4.1. Cortex Microcontroller Software Interface Standard (CMSIS)

Los microcontroladores ARM Cortex-M son muy complejos en comparación con los microcontroladores clásicos de 8 bits. Esto forzó a los fabricantes a proporcionar una serie de bibliotecas que abstraen distintos aspectos del microcontrolador con el fin de simplificar el desarrollo de aplicaciones y, por tanto, competir mejor en el mercado. Esta realidad llevó a una fragmentación del ecosistema ARM Cortex-M, pues las bibliotecas de un fabricante no eran compatibles con las de otro. Esta fragmentación puede ser beneficiosa para algunos fabricantes predominantes, pero no es ventajosa para ARM, para los desarrolladores de aplicaciones ni para fabricantes pequeños. Para poder hacer frente a esta problemática, ARM, distintos fabricantes de micros (ST, Energy Micro, NXP, Toshiba, etc..), fabricantes de herramientas, etc. trabajaron en una propuesta común que permitiera la interoperabilidad de herramientas y facilitasen la migración entre microcontroladores de distintos proveedores. Como resultado, surge la librería *Cortex Microcontroller Software Interface Standard (CMSIS)*.

Entre otras cosas, CMSIS propone una arquitectura a distintos niveles para simplificar la manera de resolver distintos problemas. En la figura 3 se representa esta arquitectura.

El bloque CMSIS-core provee las funcionalidades para arranque del sistema (reloj, ...) hasta llegar al main(), acceso a las características específicas del núcleo y periféricos básicos, una visión consistente de los registros de periféricos y servicios de interrupción, etc.

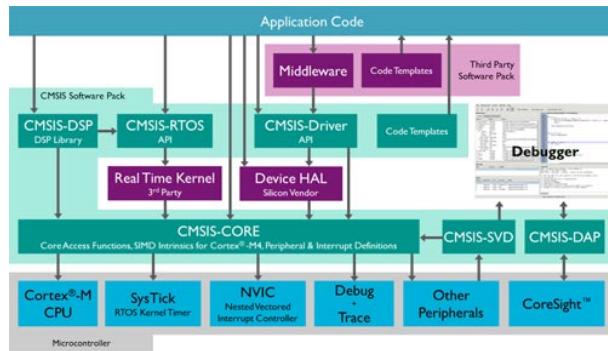


Figura 3: Arquitectura de la librería CMSIS

El bloque CMSIS-RTOS provee una abstracción de las primitivas de los Real-Time Operating Systems (RTOS) para microcontroladores. Intenta facilitar la migración entre distintos proveedores de microkernels.

El bloque CMSIS-DSP pretende proveer de funciones típicas de DSP de manera sencilla para que sea fácil explotar este potencial sin necesidad de ser un experto.

A este nivel se proporcionan primitivas para filtros digitales, PIDs, transformada de Fourier, operaciones con matrices, etc.

El bloque CMSIS-SVD pretende proveer una manera de especificar la descripción del sistema muy portable para permitir la rápida adaptación de las herramientas de depuración a nuevos miembros que aparezcan en el mercado.

CMSIS intenta ser simple, y propone (y no obliga) una manera de desarrollar el código siguiendo una reglas sencillas que se basan en la experiencia de los ingenieros y en buenas prácticas de desarrollo de código como por ejemplo cumplir MISRA 2004 [MISRA] de la Motor Industry Research Academy como base para aplicaciones en sistema críticos, documentar el código fuente empleando doxygen [doxygen] para que se automatice la generación de manuales, el empleo de la estandarización de datos enteros (stdint), el uso de camelCase (separación por mayúsculas) para nombres de variables, etc.

## 5. Primeros pasos

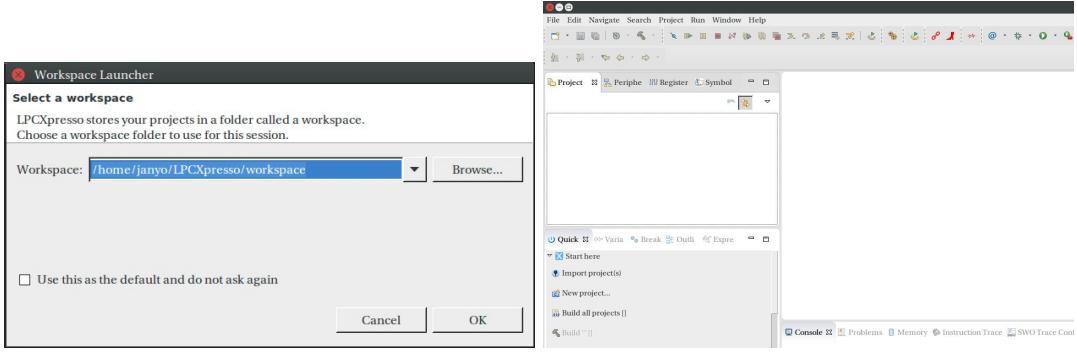
### 5.0.1. Importar librería CMSIS

Los ejercicios propuestos en este laboratorio están realizados con la librería CMSIS. Antes de realizar cualquiera de los ejercicios hay que importar esta librería en el espacio de trabajo de la IDE LPCXpresso. Los pasos a seguir son los que se detallan a continuación.

1. Se hace click en el ícono del lanzador de la IDE
2. Se abre la ventana *Workspace Launcher*, como se ve en la figura 4 a. En esta ventana se define el espacio de trabajo. Se recomienda que la ruta de acceso al espacio de trabajo no tenga nombres de carpetas con espacio.
3. Una vez definido el lugar de trabajo se abre la pantalla del IDE LPCXpresso, figura 4 b
4. Se debe importar la librería CMSIS al espacio de trabajo. Para esto se hace click en el ícono *Import project(s)* de la pestaña **Start here**. Se abre la ventana que se muestra en la figura 5
5. Se debe seleccionar el CMSIS\_CORE\_LPC17XX. Una vez seleccionada se hace click en *Finish*. El resultado se observa en la figura 6

### 5.1. Cómo crear un proyecto

Una vez linkeada la libreria CMSIS se pueden crear proyectos basados en esta librería. Los pasos para crear un proyecto se muestran a continuación.



(a) Pantalla del Workspace Launcher

(b) IDE LPCXpresso

Figura 4: Ejecutando la IDE LPCXpresso

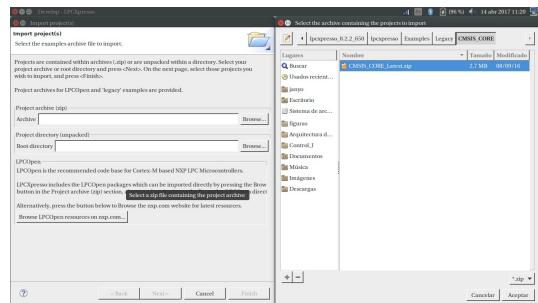
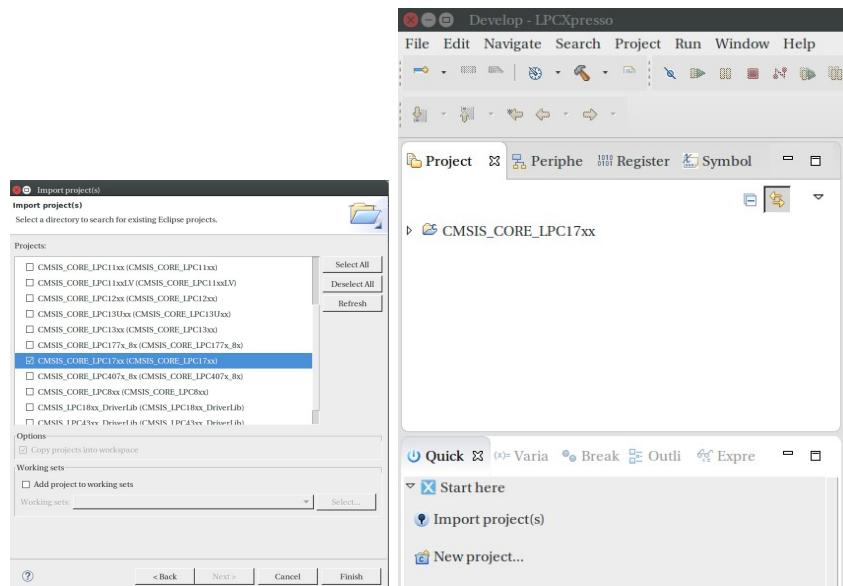


Figura 5: Ventana para importar proyectos

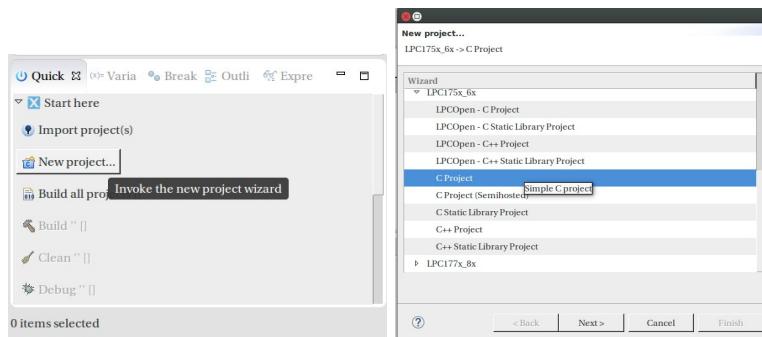


(a) Ventana de selección de CMSIS\_CORE\_LPC17XX.

(b) CMSIS\_CORE importada.

Figura 6: Librería CMSIS\_CORE\_LPC17XX importada al espacio de trabajo.

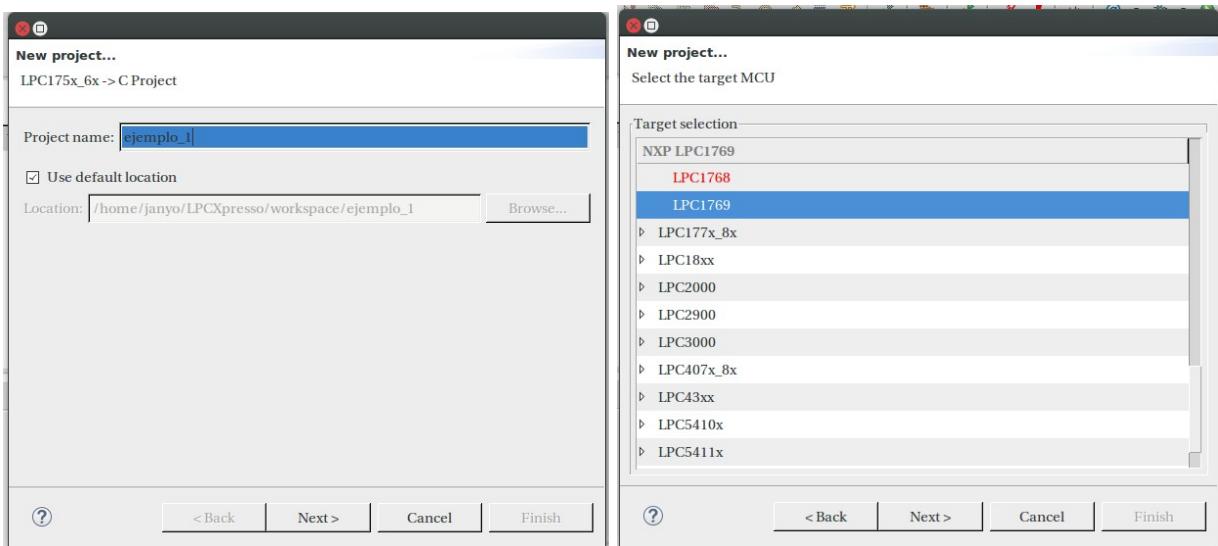
1. Se hace click en *New project*, figura 7 a
2. Se elije el LPC175x\_6x y se selecciona C Project, figura 7 b
3. Se debe nombrar el proyecto, figura 8 a. En este ejemplo nombrar al proyecto con el nombre de **Prueba**.
4. Se debe elegir el Microprocesador, figura 8 b
5. Se linkea el proyecto con la librería CMSIS, figura 9



(a) Nuevo proyecto

(b) Proyecto en C basado en el LPC1769

Figura 7: Crear nuevo proyecto



(a) Nombre del proyecto

(b) Selección de la MCU

Figura 8: Nombre del proyecto

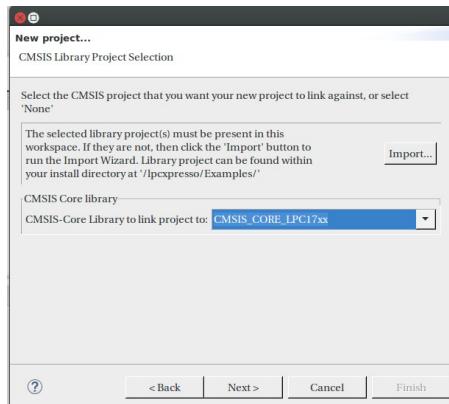


Figura 9: Crear nuevo proyecto

6. Las demás pantallas quedan por default.

Para probar que todo está funcionando correctamente abrir el archivo **Prueba.c**. Borrar todo el contenido de este archivo y pegar el siguiente código:

```
#ifdef __USE_CMSIS
    #include "LPC17xx.h"
#endif

#define LED2pIN 22 //LED2 => Pin 22

int main (void) {
    int i;
    LPC_GPIO0->FIODIR |= (1 << LED2pIN);

    while(1){
        LPC_GPIO0->FIOSET = (1 << LED2pIN);
        for(i=0;i<5000000;i++); //retardo
        LPC_GPIO0->FIOCLR = (1 << LED2pIN);
        for(i=0;i<5000000;i++); //retardo
    }
    return 0;
}
```

## 5.2. Compilación del proyecto

Para compilar se debe seleccionar el proyecto y luego oprimir Build, figura 10. La ventana Console mostrará el resultado de la compilación

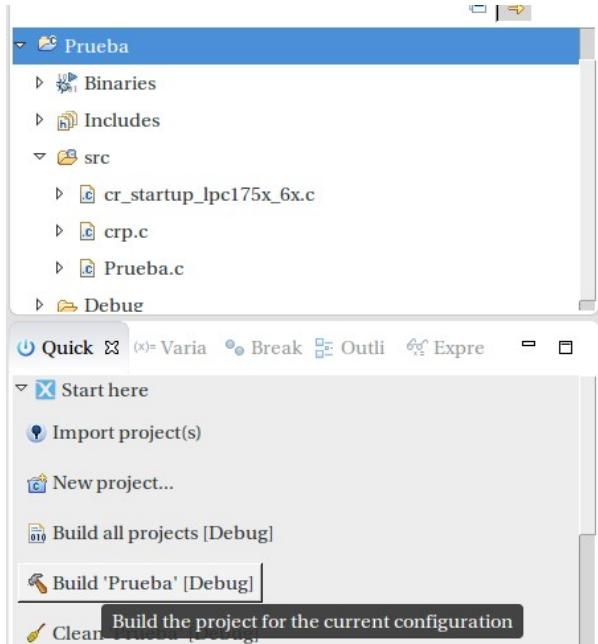


Figura 10: Crear nuevo proyecto

## 5.3. Implementación del proyecto

Antes de realizar la implementación del proyecto, conectar el USB del stick a la PC, figura 11



Figura 11: Crear nuevo proyecto

Para grabar el MCU y realizar el debugging del código se debe ir la ventana Quickstart Panel y oprimir Debug(); luego de presionarlo se debe abrir la ventana Debug como se muestra en la figura 12, hacer click en el ícono de *Resume(F8)*. Debe quedar titilando un led de la placa.

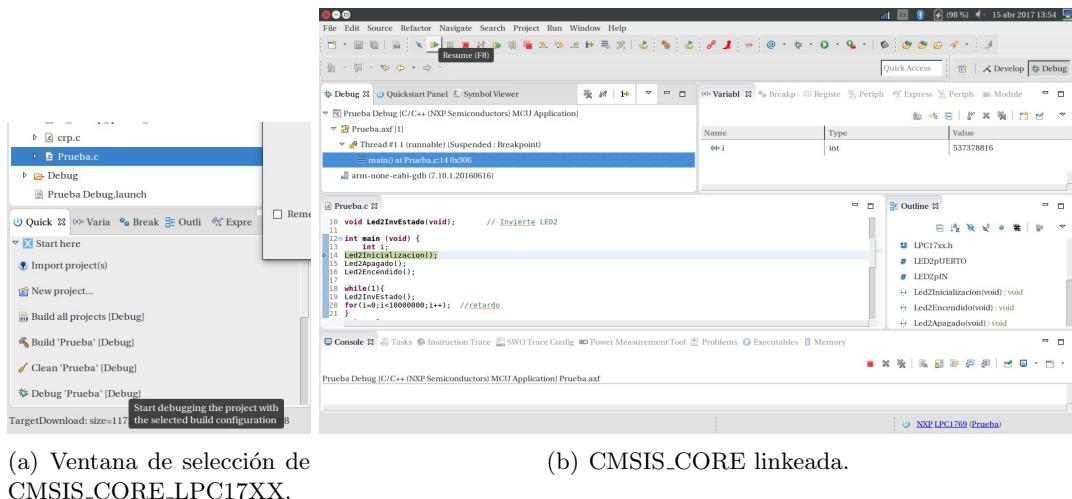


Figura 12: Librería CMSIS\_CORE\_LPC17XX linkada.

#### 5.4. Como crear una librería de usuario

Para crear una librería de usuario se deben seguir los mismos pasos que se han seguido para crear un proyecto. La diferencia radica en que se debe elegir **C Static Library Project**, como se ve en la figura 13.

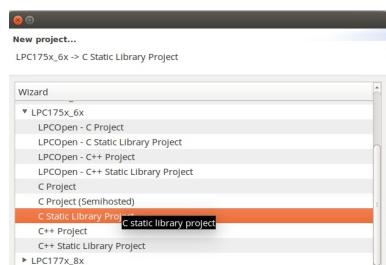


Figura 13: Crear librería de usuario

Una vez creada se deben anexar los archivos .h en la carpeta **inc** y los archivos .c en la carpeta **src** de la librería creada. Una vez hecho esto se debe compilar la librería, figura 14

Para adjuntar una librería de usuario a un proyecto se debe ingresar a las propiedades del proyecto y modificar las siguientes pantallas.

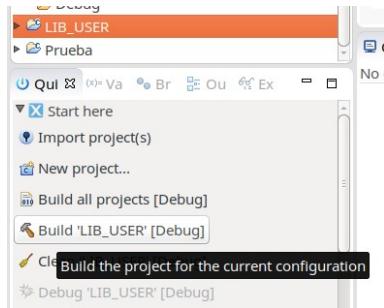


Figura 14: Crear librería de usuario

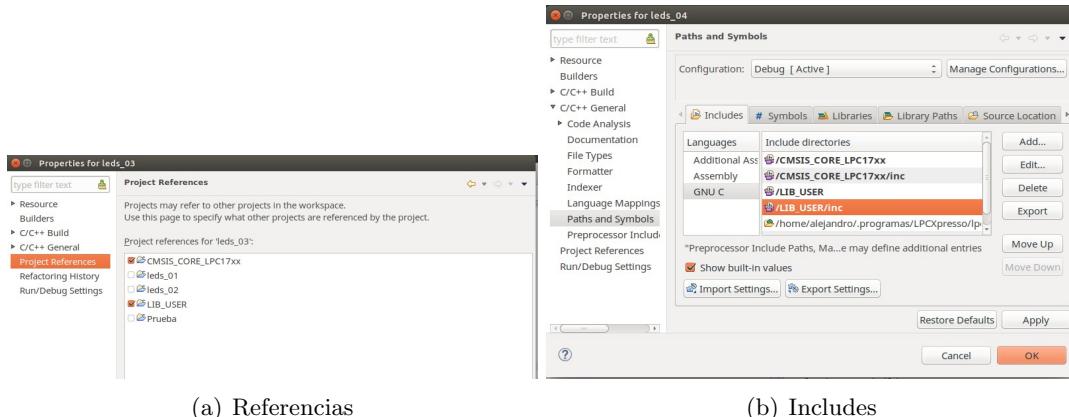


Figura 15: Ventana de References e Includes.

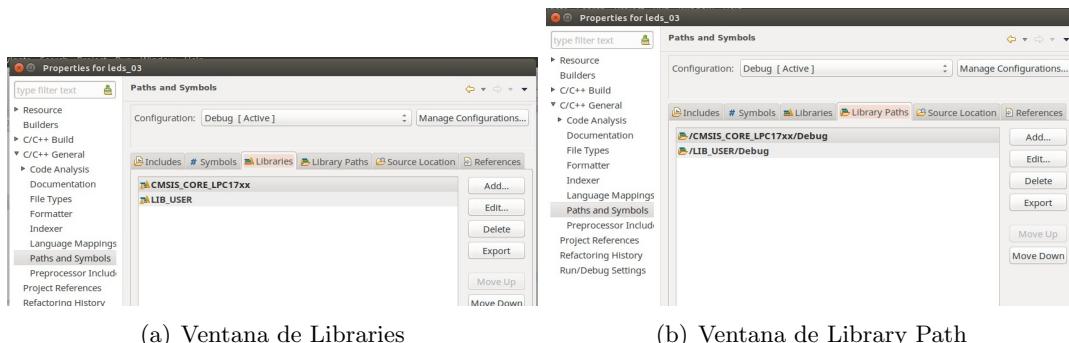


Figura 16: Ventanas de Libraries y Library Path

## 6. Ejercicios

### Ejercicio 1

Tomando como referencia los esquemáticos del *Stick* y de la placa *Kit Cortex Base Board*:

1. Diga cuantos puertos posee el Stick LPC1769
2. Diga cuantos puertos se conectan con la placa *Kit Cortex Base Board*
3. Enumere puertos y pines que se conectan con los leds de la placa *Kit Cortex Base Board*
4. Enumere puertos y pines que se conectan con los switch de la placa *Kit Cortex Base Board*
5. Indique, mediante línea de código, cómo se configuran y se acceden a los puertos para su lectura y escritura usando la librería CMSIS.

## Ejercicio 2

Teniendo en cuenta los datos del ejercicio 1:

1. Haga un programa que permita prender y apagar los ocho leds de la placa *Kit Cortex Base Board* en forma secuencial.
2. Usando el programa del ejercicio anterior, cree las funciones: **iniciarLeds()**, **prenderLeds()**, **apagarLeds()** que permitan modularizar el código anterior.
3. Cree una función **demora()** que reciba como parámetro un entero que indique una demora específica.
4. Usando las funciones definidas en el 2 y 3 resuelva el ejercicio 1. Para llevar a cabo esta tarea cree los archivos .c y .h que crea necesario para que la solución sea clara y modularizada.
5. Implemente el Tipo de Dato Abstracto Led el cual provee las siguientes funciones:
  - **inicializarLed**: esta función establece el pin del led y la demora con la que se trabajará en el programa.
  - **encenderLed**: permite encender el led.
  - **apagarLed**: apaga el led.

El tipo de dato abstracto debe proveer todas las operaciones necesarias para que el programador pueda modificar el estado interno del mismo.

6. Usando el tipo de dato abstracto definido previamente resuelva el ejercicio 1.

## Ejercicio 3

Teniendo en cuenta los datos del ejercicio 1:

1. Haga un programa que al apretar uno de los switch prenda o apague los ocho leds de la placa *Kit Cortex Base Board*. Se aconseja hacer una función mediante archivos .c y .h que permitan iniciar y manejar los switch.
2. Usando el mismo programa del punto anterior haga que el programa anexe el otro switch y que esto permita realizar una cuenta ascendente en binario cuya salida sean los ocho leds de la placa *Kit Cortex Base Board*.
3. Discuta la posibilidad de resolver los ejercicios anteriores utilizando tipos de datos abstractos. ¿Tiene ventajas utilizar la aproximación antes mencionada para resolver los ejercicios y problemas de programación en general?

## Ejercicio 4

Usando los archivos .c y .h generados cree una librería de usuario **LIB\_USER** dentro del Workspace. Pruebe esta librería con los ejercicio 2 y 3.

## Ejercicio 5

Usando el esquemático de la placa *Kit Cortex Base Board* y analizando la conexión de los leds RGB, haga un programa que permita encender el led rojo cuando se presione uno de los switch, que prenda el led azul cuando se presione el otro switch y que prenda el led verde cuando se presionen los dos switch al mismo tiempo.

## Referencias

- [1] LPC17XX User Manual UM10360. Rev. 01 - 4 January 2010. NXP
- [2] 32-bit ARM Cortex-M3 microcontroller; up to 512 kB flash and 64 kB SRAM with Ethernet, USB 2.0 Host/Device/OTG, CAN. Rev. 6- 25 August 2010. NXP
- [3] Esquemático LPCXpresso LPC1769 rev B. (C) Embedded Artists AB. 11-02-2011.
- [4] Embedded Systems Development - made Easy — Embedded Artists AB. [online] Available at: <http://www.embeddedartists.com/> [Accessed 18 Apr. 2017].
- [5] Esquemático del Kit Cortex Base Board. R2M Ingeniería. 26-8-2013
- [6] R2M Ingenieria. [online] Available at: <http://www.r2mingenieria.com.ar/home.php> [Accessed 18 Apr. 2017].
- [7] LPCXpresso IDE User Guide. Rev. 8.1. NXP. February 2016.
- [8] NXP Semiconductors — Automotive, Security, IoT. [online] Available at: <http://www.nxp.com/> [Accessed 18 Apr. 2017].