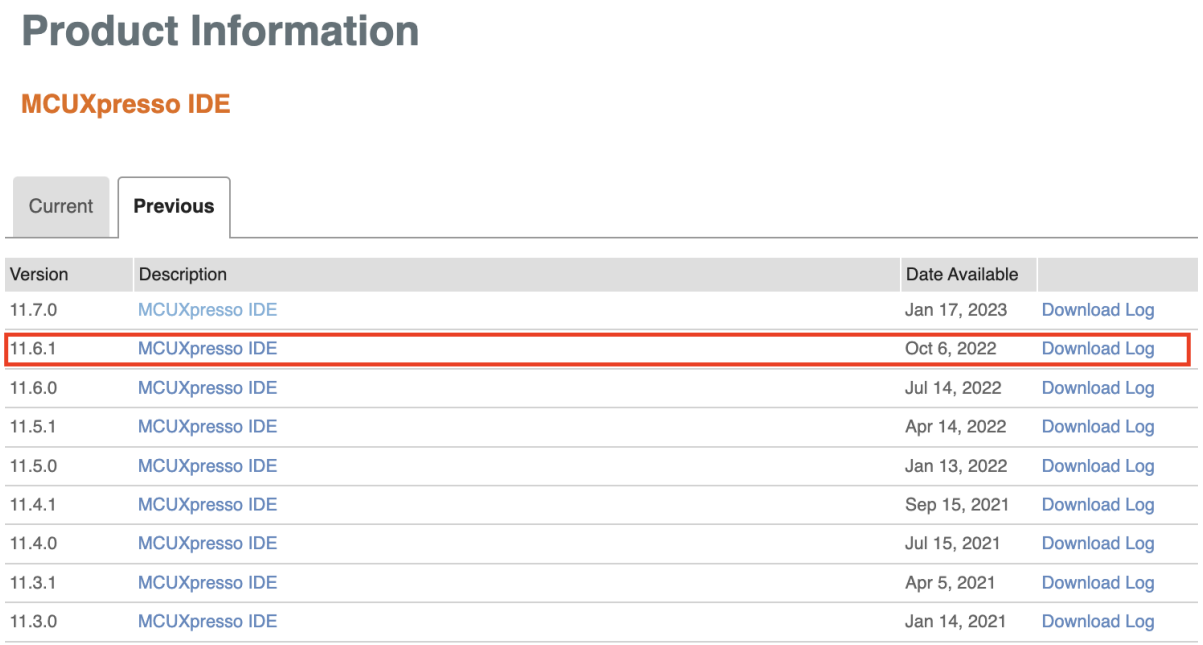
EDU-CIAA con MCUXpressoIDE y OpenOCD

El objetivo de este documento es ser una guía para instalar las herramientas necesarias para programar y debuggear la EDU-CIAA[1] con el IDE de MCUXpresso[2] y OpenOCD[3] como debugger. Además, incluye la inicialización de un proyecto desde cero y los pasos necesarios para compilar, flashear y debuggear.

1 MCUXpressoIDE

El primer paso es descargar el entorno de desarrollo (este tutorial usa la v11.6.1). Este es uno gratuito que provee NXP en el este link. Una cuenta de NXP es necesaria para poder descargar la última versión del IDE.



Product Information

MCUXpresso IDE

Current | Previous

Version	Description	Date Available	
11.7.0	MCUXpresso IDE	Jan 17, 2023	Download Log
11.6.1	MCUXpresso IDE	Oct 6, 2022	Download Log
11.6.0	MCUXpresso IDE	Jul 14, 2022	Download Log
11.5.1	MCUXpresso IDE	Apr 14, 2022	Download Log
11.5.0	MCUXpresso IDE	Jan 13, 2022	Download Log
11.4.1	MCUXpresso IDE	Sep 15, 2021	Download Log
11.4.0	MCUXpresso IDE	Jul 15, 2021	Download Log
11.3.1	MCUXpresso IDE	Apr 5, 2021	Download Log
11.3.0	MCUXpresso IDE	Jan 14, 2021	Download Log

Figura 1: Descarga de MCUXpresso IDE v11.6.1

Este software tiene soporte para Windows, Mac y Linux. Descargar la versión correspondiente y seguir la instalación con normalidad. En el caso de Linux, es necesario correr algunos comandos de consola previo a poder ejecutar el instalador. Cuando la instalación finalice, iniciar el IDE y elegir un directorio para el workspace donde se va a trabajar.

1.1 Instalación de IDE en Linux

Una vez descargado el *.deb.bin* es necesario darle permisos para ejecutar. Abrir la consola de comandos, dirigirse al directorio de descargas o donde se aloje el instalador y luego:

```
sudo chmod +x mcuxpressoide-11.6.1_9221.x86_64.deb.bin
sudo ./mcuxpressoide-11.6.1_9221.x86_64.deb.bin
```

Verificar el nombre del instalador para ejecutar correctamente el comando. Una vez ejecutado, se abrirá una interfaz en la consola para el instalador. Continuar la instalación con normalidad aceptando instalar los programas y drivers.

2 OpenOCD

OpenOCD es una herramienta open source que permite, entre otras cosas, flashear y debuggear una gran variedad de microcontroladores y arquitecturas. Lo único que requiere es una interfaz JTAG/SWD para comunicarse con el microcontrolador.

El cómo obtener este software dependerá del sistema operativo. La versión que usa esta guía es la v0.12.0.

2.1 Windows

Esta guía opta por la alternativa de descargar una versión precompilada desde este sitio. Descargar la versión más reciente (20230202 a la fecha) y descomprimir el archivo en disco local C. Una vez descomprimido, debería verse una carpeta llamada *OpenOCD-20230202-0.12.0*.

Luego, desde el menú de Windows, buscar “Variables de entorno del sistema”:

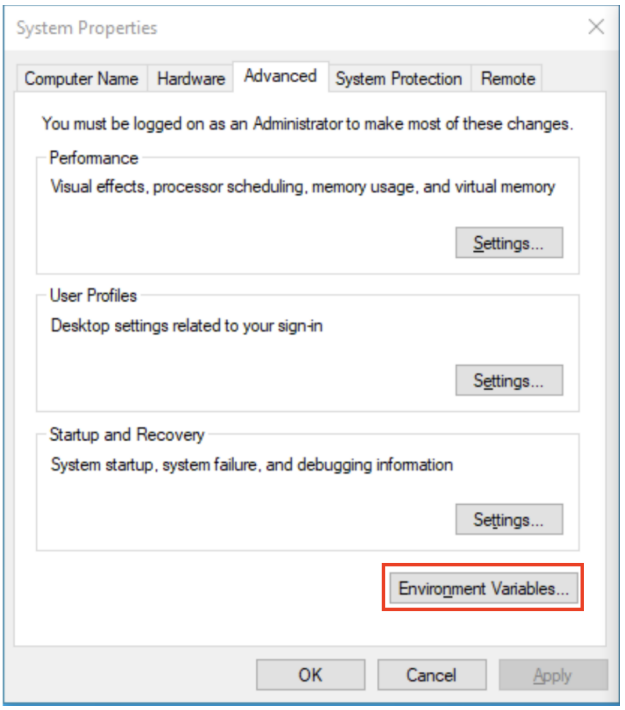


Figura 2: Menú de variables de entorno del sistema

Una vez en ese menú, vamos a la opción de “Variables de Entorno…” o “Environment Variables…” para acceder al siguiente menú.

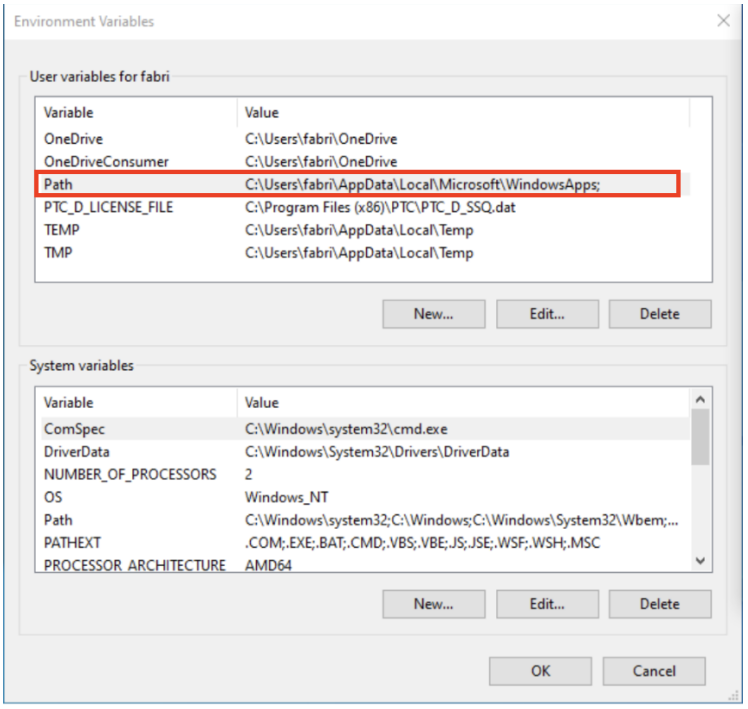


Figura 3: Variables de entorno

Desde esta segunda ventana, en la parte de variables de usuario, hacemos doble click en “Path” para editar las variables de entorno del usuario.

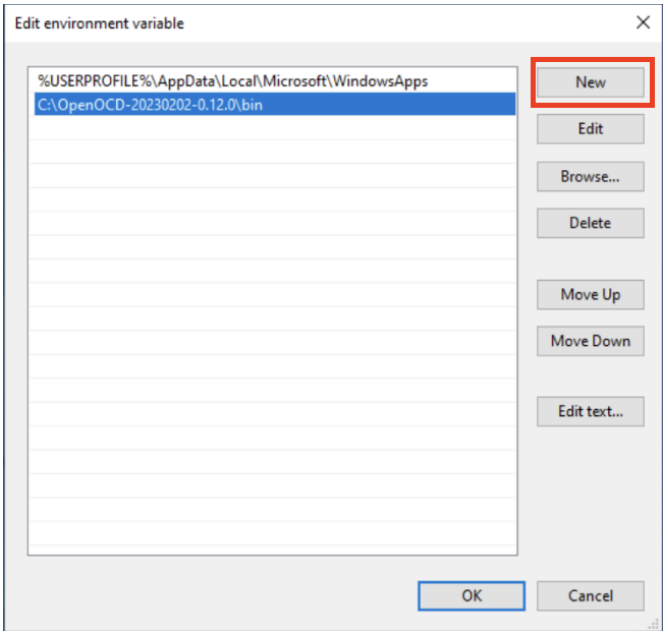


Figura 4: Ruta para añadir al PATH

Desde este último menú, agregamos la ruta al openocd.exe que en esta guía se encuentra en la ruta: *C:\OpenOCD-20230202-0.12.0\bin*. Luego de agregar la ruta, cerrar todas las ventanas dando OK o aplicar. Esto le da a Windows la dirección a donde buscar el OpenOCD cuando sea invocado desde el MCUXpressoIDE o consola.

2.2 GNU/Linux

En las distintas distros de GNU/Linux, alcanza con correr el comando:

```
sudo apt install openocd    # Para distros basadas en Debian
```

2.3 MacOS

Si se tiene instalado *homebrew*[4], alcanza con correr el comando en la terminal:

```
brew install open-ocd
```

2.4 Verificación de instalación

En cualquier sistema operativo que se haya instalado, para verificar que se haya instalado correctamente se debe correr en una terminal o consola de comandos:

```
openocd -v
```

El mensaje de salida debería ser *Open On-Chip Debugger 0.12.0* o la versión que se haya instalado. Si el comando no se reconoce, revisar la instalación en el paso correspondiente.

3 Plugin de OpenOCD para MCUXpressoIDE

El paso siguiente es descargar una extensión para el IDE que permita hacer uso del OpenOCD. Para eso, desde el MCUXpressoIDE vamos a *Help* > *Eclipse Marketplace*. En el buscador, vamos a escribir y descargar la herramienta *Eclipse Embedded C/C++*[5]. La versión a la fecha de este documento es la v6.3.1.

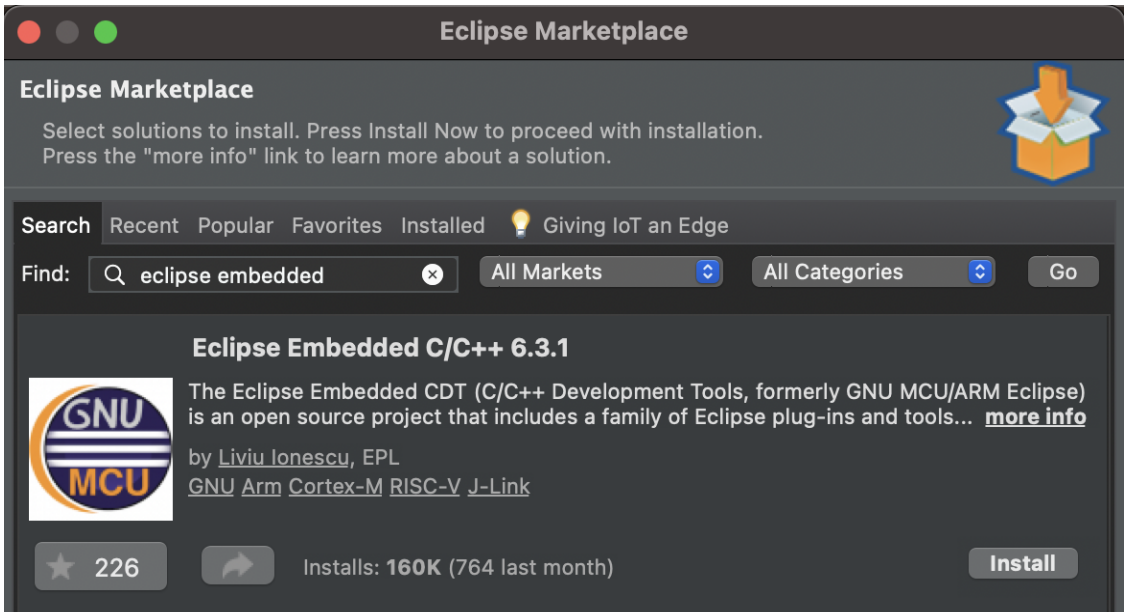


Figura 5: Extensión para MCUXpressoIDE

Durante la instalación, aceptar los términos y condiciones y luego reiniciar el IDE cuando se finalice.

4 Paquete de software LPCOpen para LPC4337

Cualquier proyecto que se cree va a requerir una serie de bibliotecas para trabajar sobre los registros del microcontrolador. NXP brinda esto en su página. De ella, descargar la versión 2.20 para LPCXpresso v8.2.0:

Older versions of LPCOpen 2.xx software package downloads

Supported board(s)/devices(s)	Software Download link	Toolchain[1]	Documentation download link[2]	Debugger(s)[3]	Related downloads	Version history and known issues
NXP LPCXpresso4337 and LPCXpresso43S37 boards (Compatible with LPCXpresso4367 and LPCXpresso43S67 boards)	v2.20 Release Date: 10/24/2016	LPCXpresso v8.2.0	Windows help file (chm) HTML help package	CMSIS-DAP (LPC-Link2)	Windows USB drivers LPCSPIFI Library Source	History
	v2.20 Release Date: 10/24/2016	IAR EWARM 7.30.3 Keil MDK-ARM v5.12		CMSIS-DAP (Link-II)		

Figura 6: Descarga de LPCOpen para LPC4337

Una vez descargado, descomprimir y desde el workspace elegido para el MCUXpresso vamos a importar uno de los proyectos yendo a *File* > *Import…* luego en la ventana que aparece vamos a *General* > *Existing Projects into Workspace* y luego vamos a elegir la opción *Select archive file* y clickeando en *Browse* vamos a buscar el .zip que fue descargado.

Esto va a darnos la opción de importar todos los ejemplos disponibles que tiene esa versión de LPCOpen para el LPC4337. Solo hace falta tildar dos de ellos: *lpc_chip_43xx* y *lpc_chip_43xx_m0*.

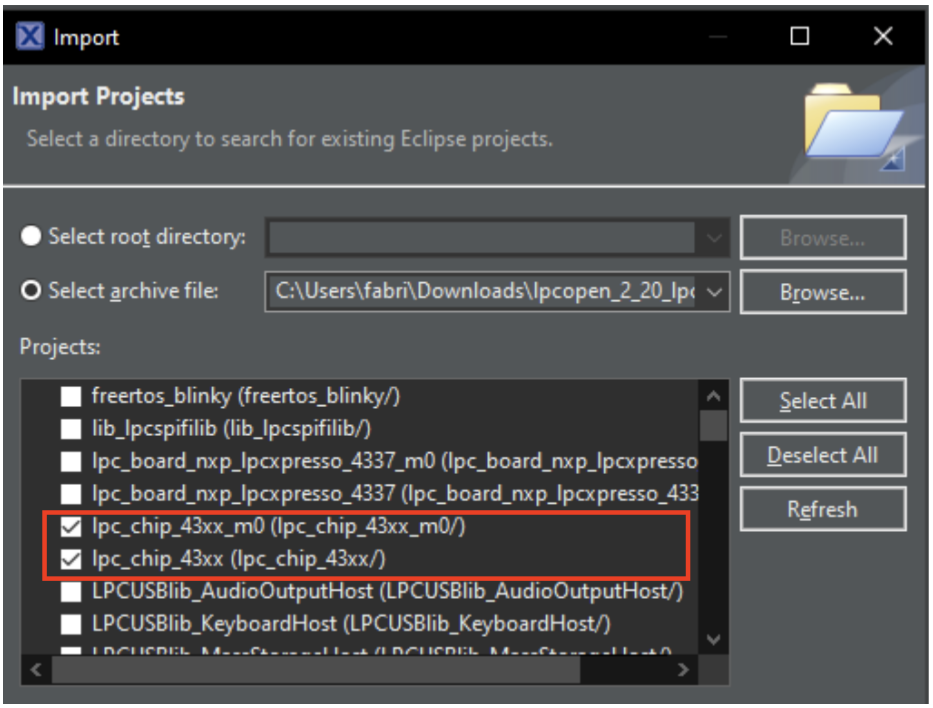


Figura 7: Import de biblioteca de LPCOpen para LPC4337 al workspace

Una vez que se seleccionaron ambos proyectos, damos click a finalizar y se importarán al workspace. Esto que importamos es la biblioteca para cada uno de los periféricos del LPC4337 para ambos procesadores, el M4 y el M0.

5 Nuevo proyecto

En esta sección se trabaja, la creación de un proyecto nuevo y luego configurar las variables necesarias para usar OpenOCD.

5.1 Crear proyecto

Para crear un nuevo proyecto para la EDU-CIAA, los pasos son:

1. Ir a *Archivo* > *Nuevo* > *Crear un nuevo proyecto en C/C++*.
2. De la lista de microcontroladores, elegir *LPC43xx* > *LPC4337*. Esto crea un proyecto para el procesador M4 de la CIAA.
3. A continuación, elegir de la lista de tipos de proyectos el que se llama *LPC43xx (Cortex-M4 basic)* > *LPCOpen - C Project*.
4. Asignar un nombre al proyecto, por ejemplo *ciaa_blinky*.
5. Los siguientes pasos de la configuración del proyecto se deben dejar por defecto en esta instancia.

El proyecto va a tener una estructura como esta:

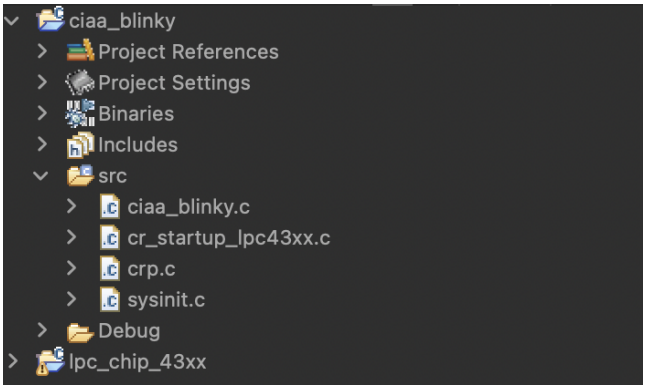


Figura 8: Estructura de proyecto

Luego, ir al *ciaa_blinky.c* y reemplazar el contenido por el siguiente:

```
#include "chip.h"
// Macros para los datos del LED
#define LED1_SCU_PORT          2
#define LED1_SCU_PIN   10
#define LED1_GPIO_PORT         0
#define LED1_GPIO_PIN  14
#define LED1_GPIO_FUNC         SCU_MODE_INACT | SCU_MODE_ZIF_DIS | SCU_MODE_INBUFF_EN
| SCU_MODE_FUNC0
/**
 * @brief Programa principal
 */
int main(void) {
```

```
	// Inicializo clocks
	SystemCoreClockUpdate();
	// Asigno la funcion GPIO al LED1
	Chip_SCU_PinMuxSet(LED1_SCU_PORT, LED1_SCU_PIN, LED1_GPIO_FUNC);
	// Configuro el LED1 como salida
	Chip_GPIO_SetPinDIROutput(LPC_GPIO_PORT, LED1_GPIO_PORT, LED1_GPIO_PIN);
	while(1) {
		// Conmuto el LED1
		Chip_GPIO_SetPinToggle(LPC_GPIO_PORT, LED1_GPIO_PORT, LED1_GPIO_PIN);
		// Delay
		for(uint32_t i = 0; i < 10000000; i++);
	}
}
```

Una vez hecho, guardar los cambios y compilar con el icono del martillo o con el comando *Ctrl + B*. Si se siguieron los pasos correctamente, la compilación no debería haber traído ningún error.

5.2 Archivo de configuración

Para poder usar el OpenOCD, va a hacer falta un archivo de configuración con ciertas reglas. Dentro del workspace, crear un directorio que se llame *openocd* y dentro de él, un archivo llamado *lpc4337.cfg*. El contenido de este archivo debe ser el siguiente:

```
adapter driver ftdi
ftdi_vid_pid 0x0403 0x6010
ftdi_channel 0
ftdi_layout_init 0x0708 0xFFFB
adapter speed 2000

set _CHIPNAME lpc4337
set _M4_JTAG_TAPID 0x4ba00477
set _M0_JTAG_TAPID 0x0ba01477

jtag newtap $_CHIPNAME m4 -irlen 4 -ircapture 0x1 -irmask 0xf -expected-id
$_M4_JTAG_TAPID
jtag newtap $_CHIPNAME m0 -irlen 4 -ircapture 0x1 -irmask 0xf -expected-id
$_M0_JTAG_TAPID

dap create $_CHIPNAME.m4.dap -chain-position $_CHIPNAME.m4
dap create $_CHIPNAME.m0.dap -chain-position $_CHIPNAME.m0
target create $_CHIPNAME.m4 cortex_m -dap $_CHIPNAME.m4.dap
target create $_CHIPNAME.m0 cortex_m -dap $_CHIPNAME.m0.dap

set _WORKAREASIZE 0x8000
$_CHIPNAME.m4 configure -work-area-phys 0x10000000 -work-area-size $_WORKAREASIZE

set _FLASHNAME $_CHIPNAME.flash
```

```
flash bank $_FLASHNAME lpc2000 0x1a000000 0x80000 0 0 $_CHIPNAME.m4 lpc4300 96000
calc_checksum

set _FLASHNAMEB $_CHIPNAME.flashb
flash bank $_FLASHNAMEB lpc2000 0x1b000000 0x80000 0 0 $_CHIPNAME.m4 lpc4300
96000 calc_checksum

reset_config none

cortex_m reset_config vectreset
targets $_CHIPNAME.m4
```

La versión completa con comentarios de este archivo puede encontrarse en este repositorio.

5.3 Drivers para Windows

Para sistemas operativos con Windows es necesario actualizar los drivers para el chip FTDI en la EDU-CIAA que se encarga de establecer la comunicación USB entre el LPC4337 y la computadora. Primero se debe enchufar el USB por el conector nombrado USB DEBUG en la placa.

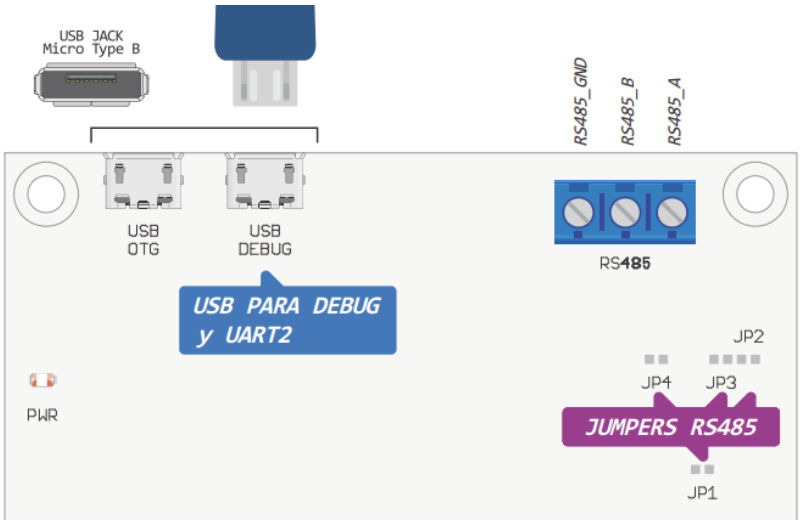


Figura 9: Conector USB para debug en la EDU-CIAA

Luego, como una solución fácil se puede descargar el Zadig (v2.8.782 al día de la fecha de este documento), que es un programa que permite listar dispositivos USB y actualizar drivers. Una vez descargado, se debe ejecutar y en el menú de *Options* ir a *List All Devices*.

Si la EDU-CIAA está conectada correctamente, deben figurar dos dispositivos: *Dual RS232-HS (Interface 0)* y *Dual RS232-HS (Interface 1)*. Seleccionar la interfaz 0 y verificar que figure como driver actual el *FTDIBUS (v2.12.36.4)* y reemplazarlo por el *WinUSB (v6.1.7600.16385)* haciendo click en *Replace Driver*. El proceso toma unos momentos.

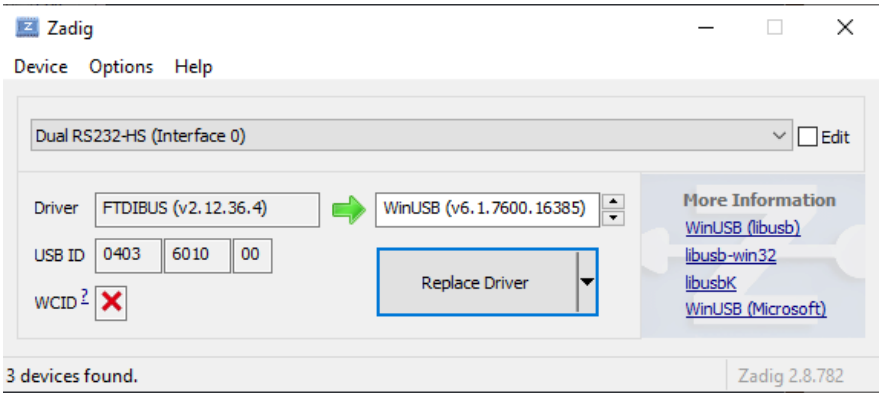


Figura 10: Reemplazo de driver con Zadig

Una vez actualizado el driver, ya es posible programar la EDU-CIAA con el OpenOCD a través del USB.

5.4 Configuración de OpenOCD

Para flashear y debuggear, es necesario configurar el OpenOCD para que el MCUXpresso sepa donde esta la ruta. Para eso, ir a *Run* > *Debug Configurations.* En la nueva ventana, hacer doble click en *GDB OpenOCD Debugging* (si esta opción no aparece, revisar este paso anterior). Esto crea una nueva configuración para debuggear con OpenOCD. En la pestaña de *Debugger*, se debe agregar una línea adicional en la ventana *Config options*:

```
-f ${workspace_loc:}/openocd/lpc4337.cfg
```

Esta línea le dice al OpenOCD a donde ir a buscar el archivo con las reglas para flashear y debuggear que dejamos previamente en el workspace. La configuración debería quedar como la siguiente imagen:

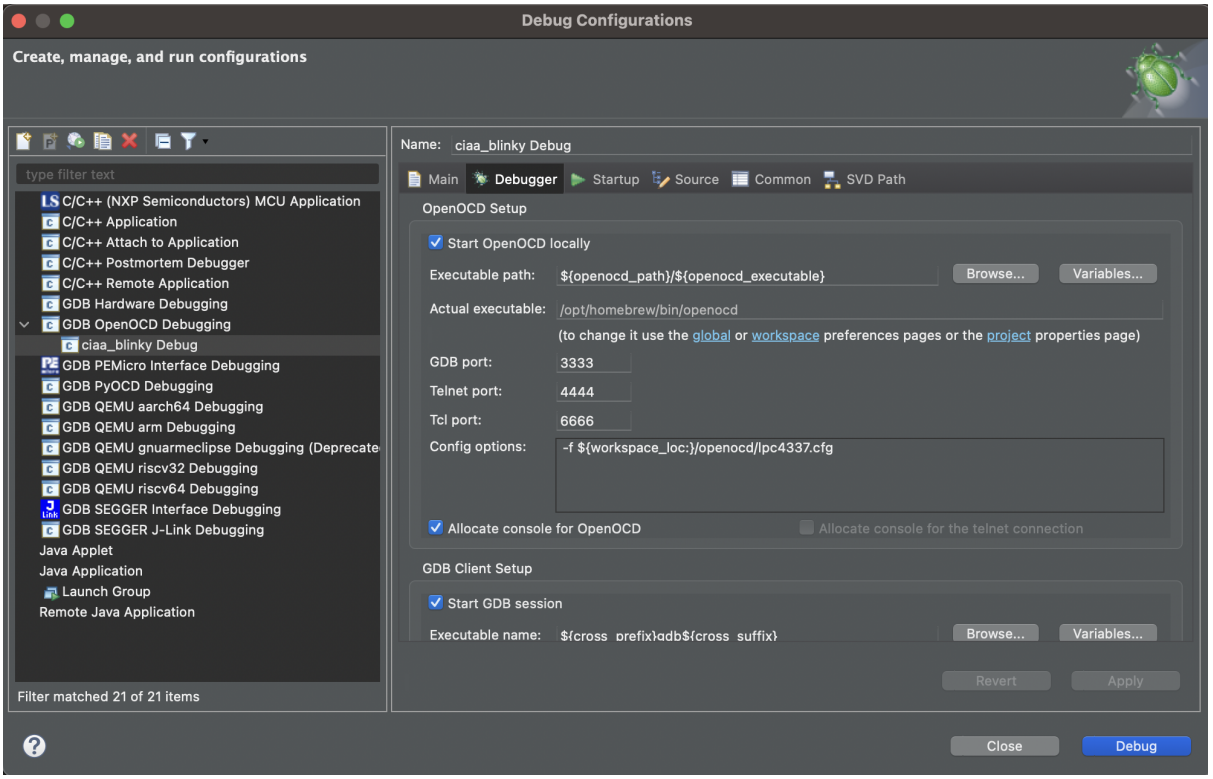


Figura 6: Configuración de OpenOCD para debuggear

Una vez que esté lista la configuración, damos click a *Debug* y, si todo está bien, debería abrirse una perspectiva de debugging.

6. Errores comunes

A continuación se recopilan algunos errores frecuentes en el proceso junto con algunas posibles soluciones al problema.

6.1 MCUXpresso no encuentra OpenOCD

En Windows especialmente, existe la posibilidad de que la ruta del OpenOCD no se haya configurado apropiadamente o que el MCUXpresso no levante correctamente la ruta a OpenOCD del sistema. Esto se puede verificar en la opción de *Actual executable*. Si este dice algo como */undefined_path/openocd.exe* entonces hay un error en la ruta.

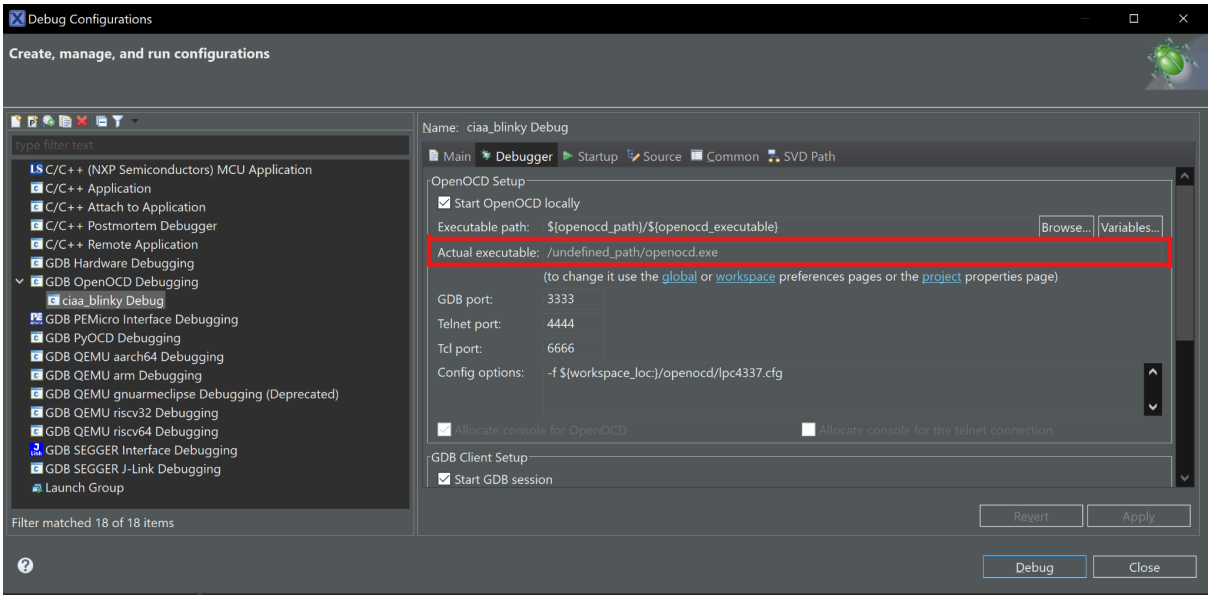


Figura 7: Solución para ruta de OpenOCD

Esto puede solucionarse haciendo click en *Browse* a la derecha y buscar el ejecutable de OpenOCD en la ruta correspondiente donde se haya instalado. Puede ajustarse de forma global haciendo click en *global* debajo de la ruta, luego en *Browse* y eligiendo la ruta donde se encuentra el directorio *bin* del OpenOCD.

6.2 OpenOCD arroja unexpected command line argument

Si al intentar debuggear OpenOCD arroja el error *"openocd unexpected command line argument"* es muy probable que la ruta donde se encuentra el archivo *lpc4337.cfg* contenga espacios o caracteres inválidos. Para solucionarlo, copiar la ruta al archivo de configuración literal desde el raíz del sistema y reemplazarla en *Config options* por la ruta anterior entre comillas:

-f "ruta_al_workspace/openocd/lpc4337.cfg"

7 Referencias

Referencias usadas para armar esta guía:

- [1] Página del Proyecto CIAA: http://www.proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:edu-ciaa:edu-ciaa-nxp
- [2] Página de MCUXpressoIDE: https://www.nxp.com/design/software/development-software/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE#:~:text=The%20MCUXpresso%20IDE%20offers%20advanced,debugging%2C%20and%20integrated%20configuration%20tools.
- [3] Página de OpenOCD: https://openocd.org/
- [4] Página de Homebrew: https://brew.sh/

- [5] Página del plugin de Eclipse Embedded C/C++: https://marketplace.eclipse.org/content/eclipse-embedded-cc