

Índice

1. Introducción	1
2. Desarrollo	2
2.1. Origen del proyecto CIAA	2
2.2. Descripción de la placa	2
3. Instalacion de Software	5
3.1. Conceptos previos	5
3.2. Firmware de la EDU CIAA	6
3.3. Estructura de Directorios de Firmware de EDU CIAA	7
3.4. Iniciación a través de ejemplos	8
3.5. Instalación de IDE	8
4. Desinstalación	14
5. Configuración del entorno CIAA-IDE	15
5.1. Compilación de primer proyecto	15
5.1.1. Configuración del Makefile	19
5.2. Depuración con Windows	22
5.3. Depuración en la placa	25
5.3.1. Posible Problema: No reconocimiento	27
6. Primeros Pasos con la EDU-CIAA	27
6.1. Programación en Baremetal	27

1. Introducción

En este trabajo se pretende desarrollar un tutorial que permita al usuario neófito poder iniciar sus primeros proyectos usando RTOS y comprender la arquitectura de los procesadores Cortex. Este estudio aborda en primer lugar, las características principales de la placa y la correcta instalación del software sobre Windows para permitir el desarrollo de códigos sobre ella; además se incluyen posibles problemas que puedan suceder en el proceso junto con sus soluciones.

Sobre el final de dicha sección se desarrolla una guía para la correcta configuración de el entorno gráfico, para poder ejecutar la compilación del primer ejemplo que nos indica que hemos finalizado satisfactoriamente la instalación del software de la EDU CIAA.

La siguiente sección comienza introduciendo al usuario en el uso de la placa a través de la biblioteca *sAPI* (realizada por Eric Pernia) la cual nos permite el desarrollo de programas utilizando lenguaje C, nuestro propósito es brindar una explicación simple de la arquitectura de los procesadores Cortex y a su vez brindar una base para la siguiente sección del informe.

Sobre el inicio del siguiente apartado, se introduce al usuario sobre la programación a través de el sistema operativo *OSEK OS*, el cual es el método de programación en el que se proyectó cuando se realizó el diseño de la placa. Nuestra meta es fijar las bases conceptuales principales de los sistemas operativos en tiempo real, e introducir al usuario a la programación de códigos simples y alentar al usuario a profundizar sobre este estudio.

2. Desarrollo

2.1. Origen del proyecto CIAA

Sobre julio de 2013, la Secretaría de Planeamiento Estratégico Industrial del Ministerio de Industria de la Nación (SPEI) y la Secretaría de Políticas Universitarias del Ministerio de Educación de la Nación (SPU) convocaron a la Asociación Civil para la Investigación, Promoción y Desarrollo de los Sistemas Electrónicos Embebidos (ACSE) y a la Cámara de Industrias Electrónicas, Electromecánicas y Luminotécnicas (CADIEEL) a participar en el "Plan Estratégico Industrial 2020". A partir de dicha convocatoria se inició el desarrollo de la Computadora Industrial Abierta Argentina (CIAA).

El pedido inicial fue que desde el sector académico (ACSE) y desde el sector industrial (CADIEEL) se presenten propuestas para agregar valor en distintas ramas de la economía (maquinaria agrícola, bienes de capital, forestal, textil, alimentos, etc.) a través de la incorporación de sistemas electrónicos en procesos productivos y en productos de fabricación nacional. Debe destacarse que muchas empresas argentinas de diversos sectores productivos no incorporaban electrónica en sus procesos productivos o en sus productos, otras utilizaban sistemas electrónicos obsoletos, muchas utilizaban sistemas importados y sólo unas pocas utilizaban diseños propios basados en tecnologías vigentes y competitivas.

A partir de esta situación, la ACSE y CADIEEL propusieron desarrollar un sistema electrónico abierto de uso general, donde toda su documentación y el material para su fabricación estuviera libremente disponible en internet, con el objetivo de que dicho sistema pueda ser fabricado por la mayoría de las empresas PyMEs nacionales, y realizar modificaciones en base a las necesidades específicas que puedan tener.

Hoy en día la CIAA está disponible en la versión CIAA-NXP y otras seis versiones están en elaboración: CIAA-ATMEL, CIAA-FSL, CIAA-PIC, CIAA-RX, CIAA-ST, CIAA-TI. Además, se está trabajando en el firmware y en el software, para que la CIAA se pueda programar en lenguaje C utilizando una API especialmente diseñada para ser compatible con los estándares POSIX y que sea portable a diversos sistemas operativos de tiempo real.

Desde la concepción del proyecto, el diseño de la placa se encuentra pensada para soportar las condiciones hostiles de los ambientes industriales los que abundan ruidos, vibraciones, temperaturas extremas, picos de tensión e interferencias electromagnéticas, y además se diseñó de modo tal que pueda ser fabricada en Argentina.

2.2. Descripción de la placa

La CIAA es una placa electrónica provista de un microcontrolador y puertos de entrada y salida, cuyo diseño se encuentra disponible en Internet, dicha placa fue concebida para ser utilizada para sistemas de control de procesos productivos, agroindustria, automatización, entre otras; es notable destacar que gracias a la posibilidad del acceso a la información de dicha plataforma, cualquier empresa que desee utilizarla para la elaboración de sus productos puede rediseñarla; de modo que esto fomenta el diseño y la fabricación nacional de sistemas electrónicos.



Figura 1: Placa EDU CIAA

La placa EDU CIAA es la versión educativa de esta, la cual se encuentra diseñada con el propósito de conseguir una plataforma base para el desarrollo de proyectos educativos, en este caso, se busca proporcionar las bases del desarrollo de códigos utilizando RTOS.

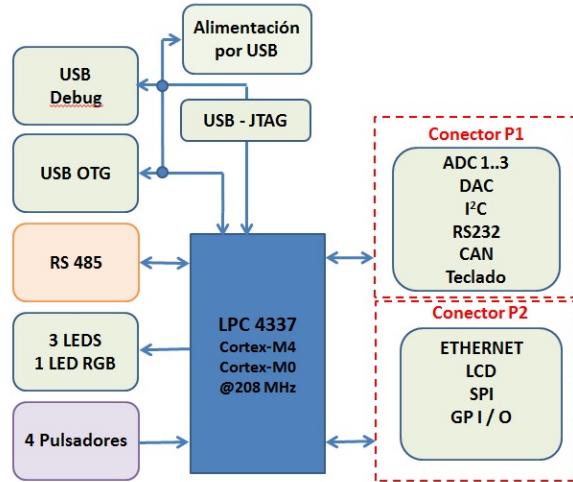


Figura 2: Diagrama en bloques de EDU CIAA basado en LPC4337.

El procesador sobre que utiliza es el LPC4337, basado en el procesador ARM Cortex M4, utilizado para aplicaciones sobre sistemas embebidos, dicho sistemas incluyen el procesador Cortex M0. Utiliza una memoria flash de 1Mb, memoria de 264 kB de SRAM, memoria ROM de 64 kB, E2PROM de 16kB, y una memoria OTP de 64 bit.

La frecuencia de trabajo del procesador alcanza los 204 Mhz, una característica vital del procesador, es que provee soporte para la depuración en JTAG, con posibilidad de incluir hasta 8 breakpoints, y 4 watchpoints.

Dicho procesador nos brinda una interface que nos possibilita extender hasta 164 pines de entrada-salida de propósito general (GPIO), provee una interface USB Host/Device 2.0 de alta velocidad con soporte para acceso directo de memoria, una interface UART 550 con soporte DMA , tres USART 550 con soporte para DMA.

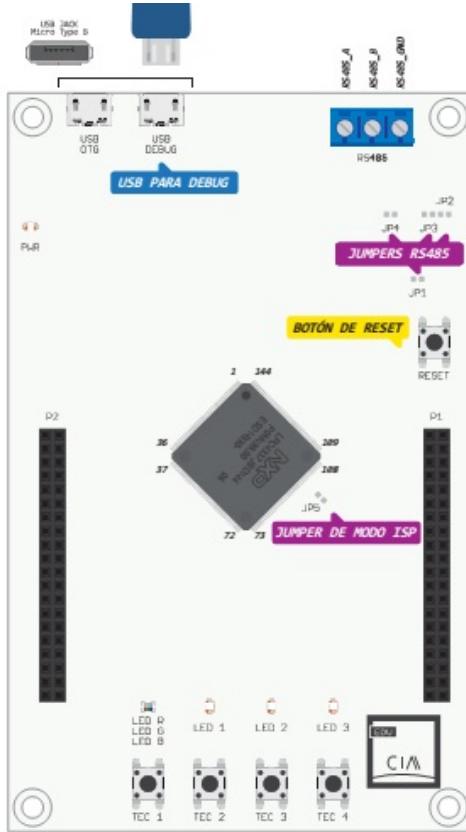


Figura 3: Imagen frontal de placa

Como periféricos analógicos, debe destacarse la inclusión de un DAC de 10 bits, con soporte DMA y frecuencia de conversión de 400000 muestras por segundo. Dos ADC's con soporte DMA, y frecuencia de conversión de 400000 muestras por segundo, con un número máximo de 8 canales sobre cada ADC. Y un ADC de 12 bits de 6 canales con soporte DMA, y frecuencia de conversión que puede alcanzar hasta los $80,10^6$ muestras por segundo.

El cristal oscilador posee un rango de operación desde 1 Mhz hasta 25 Mhz, este procesador incluye un reloj de tiempo real de baja potencia, el cual utiliza un oscilador de cristal.

La placa provee alimentación de 3,3 V (cuyo rango oscila entre 2,2 V hasta 3,6 V), y es capaz de operar en cuatro modos, los cuales se denominan *sleep*, *deep-sleep*, *power-down*, y *deep power-down*. Es posible restablecer la operación de la placa desde los modos *deep-sleep*, *power-down*, y *deep power-down*, a través de interrupciones externas.

Sobre la Figura 1 2 se proporciona el diagrama en bloques de la placa, puede observarse que la placa cuenta con 2 puertos micro-USB (uno para aplicaciones y debugging, otro para alimentación); 4 salidas digitales implementadas con leds RGB, 4 entradas digitales con pulsadores; 1 puerto de comunicaciones RS485 con bornera. La Figura 2 3 nos muestra una imagen frontal de la placa; nótese la presencia de dos puertos sobre los cuales se ubican los pines correspondientes a la placa,

la Figura 3 4 ilustra el distribución de dichos pines sobre cada puerto.

Sobre el puerto P1, se ubican los siguientes módulos:

1. 3 entradas analógicas ($ADC0_1, 2y3$)
2. 1 salida analógica (DAC0).
3. 1 puerto I2C.
4. 1 puerto asincrónico full duplex (para RS-232).
5. 1 puerto CAN.
6. 1 conexión para un teclado 3x4.

Sobre el puerto P1, se ubican los siguientes módulos:

1. 1 puerto Ethernet
2. 1 puerto SPI
3. 1 puerto para Display LCD con 4 bits de datos, Enable y RS.
4. pines genéricos de I/O.

3. Instalacion de Software

3.1. Conceptos previos

El desarrollo de códigos para sistemas embebidos tiene ciertas semejanzas con el desarrollo de aplicaciones en las PC, en nuestro caso particular se utiliza un compilador llamado *GCC* con soporte para la compilación de proyectos sobre los procesadores basados en la arquitectura ARM, en este caso particular, el compilador utilizado para el procesador de la EDU CIAA (el cual es el LPC4337) se lo denomina *arm-none-eabi-gcc*.

Para la ejecución de la depuración de algun programa previamente compilado, el hardware de la CIAA viene provisto con el chip *FT2232H*, que se encarga de hacer un puente entre la interfase JTAG del microcontrolador, y el USB que conecta a la PC en el puerto USB dedicado al debug. Mediante la herramienta de código abierto *OpenOCD (On Chip Debugger)* se controla el chip *FT2232H* por el USB y ademas todo lo referido al JTAG. Luego la herramienta de depuración *GDB* utilizado en el IDE-Eclipse que se instala, se comunica sobre el puerto 3333 (TCP) que el *OpenOCD* tiene en escucha esperando la conexión.

Debe tenerse en cuenta que el chip *FT2232H* posee 2 canales de comunicación independientes (A y B), sin embargo, ambos salen por el mismo USB, de modo que la PC detecta 2 dispositivos distintos (en realidad es uno compuesto). Uno de ellos, se conecta al JTAG manejado por *OpenOCD* como fue mencionado, mientras que el otro se ve como un puerto virtual COM. Este último sirve principalmente para la depuración.

Dado que al funcionará como dos dispositivos distintos, para cada uno de ellos debe realizarse la instalación de un driver adecuado, en principio debe optarse por realizar la instalación de los drivers por defecto del fabricante FTDI para puerto virtual.

3.2. Firmware de la EDU CIAA

Considerando que el usuario previamente ha trabajado sobre placas de desarrollo tales como la MCE Debug, etc, y sobre microcontroladores PIC. Es necesario destacar un concepto teórico que nos brinda la posibilidad de fundamentar el trabajo sobre la placa EDU CIAA. Al trabajar sobre los otros dispositivos, es común la utilización de programas tales como *MPLABX*, o *PICC* a través del compilador *CCS Compiler*; puntualmente; cuando se inicia un nuevo proyecto a través de la herramienta de creación de la misma, es usual configurar este proyecto de manera que el software IDE genera un archivo *makefile* para la compilación del proyecto.

En este caso particular, el software IDE de la EDU CIAA trabaja de forma ligeramente distinta, el usuario debe crear un archivo *makefile* (basándose en un archivo proporcionado previamente, denominado *Makefile.config*) para poder efectuar la compilación del archivo y lograr la correcta configuración del programa, sobre la placa EDU CIAA. Dentro de dicho archivo se establece la configuración para la arquitectura del procesador utilizado. Cuando se desea realizar el primer proyecto sobre la placa, el usuario debe crear su propio archivo *Makefile.mine*, de manera que ésta se encuentra basada en el archivo *Makefile.config* brindado previamente al momento de establecer un nuevo proyecto añadiendo un Firmware que previamente ha sido diseñado por los creadores de la placa.

Debe tenerse en cuenta que la dinámica de trabajo sobre la placa se encuentra pensada para trabajar sobre la plataforma de versionado *Git*; en este caso en particular, el archivo *Makefile.mine* se encuentra diseñado de forma tal que dicho archivo sea ignorado al sincronizar su repositorio local, con su repositorio remoto (ubicado sobre *Github*).

En el *Makefile.mine* se pueden editar y configurar los siguientes parámetros:

1. **ARCH** indica la arquitectura del hardware para la cual se desea compilar. Ej: x86, cortexM4.
2. **CPUTYPE** indica el tipo de CPU. Ej: none, ia32, ia64, lpc43xx.
3. **CPU** indica la CPU para la que se desea compilar. Ej: none, lpc4337.
4. **COMPILER** es el compilador a utilizar. Ej: gcc.
5. **BOARD** es la placa sobre la cual se trabajará (CIAA-NXP, EDU-CIAA-NXP, etc.)
6. **PROJECT** es el Path al proyecto a compilar. Ej: examples\$(*DS*)*blinkingbase*.

Se utiliza la variable *\$(DS)* para indicar el separador de directorios (de manera automática se usa '/' para linux y '\ para windows).

En el mismo *Makefile* aparecen al comienzo comentarios donde se indican los valores que pueden tomar estos parámetros.

Otro concepto importante sobre el cual se tiene en cuenta cuando se desarrollan proyectos propios, es que cada proyecto tiene también su propio archivo *Makefile*. El mismo se encuentra bajo el directorio **mak** en el directorio principal del proyecto o ejemplo. En el ejemplo **examples/blink** el *makefile* del ejemplo se encuentra en **examples/blink/mak** y se llama **Makefile**.

Sobre este *Makefile* contiene las siguientes definiciones:

1. **project** el nombre del proyecto y por ende nombre del ejecutable.

2. **`$(project)_PATH`** es el directorio del proyecto.
3. **INCLUDE** los paths a indicar al compilador para buscar includes files.
4. **SRC_ FILES** archivos a compilar ya sean archivos c como c++.
5. **OIL_ FILES** configuración del sistema operativo (si es utilizado).

Cada proyecto incluye en su Makefile los módulos (aca digo que son los módulos) a compilar en una variable llamada MODS, por ejemplo:

_____decidir si poner como imagen o como texto

```
MODS += modules$(DS)posix modules$(DS)ciaak modules$(DS)config modules$(DS)bsp
modules$(DS)platforms
```

Es recomendable utilizar `$(DS)` en vez de / o \ para mantener la compatibilidad entre sistemas operativos (Linux, Windows, MAC OS).

3.3. Estructura de Directorios de Firmware de EDU CIAA

En el directorio principal luego de hacer un git clone o al bajar una release oficial se pueden encontrar los siguientes Directorios y Archivos:

```
Firmware/
├── doc
├── examples          /* ejemplos de utilización del Firmware */
│   ├── adc_dac        /* ejemplo de conversor analogico digital */
│   └── blinking        /* ejemplo básico que hace parpadear un led */
│   /*
│   └── blinking_echo   /* ejemplo básico que hace echo en el bus
serial */
│   ├── blinking_lwip   /* ejemplo utilizando lwip (ethernet) */
│   ├── blinking_modbus /* ejemplo utilizando modbus RS485 (ascii) */
│   └── blinking_modbus_master /* ejemplo utilizando modbus master */
│   └── rtos_example    /* ejemplo de utilización de FreeOSEK */
└── externals
    ├── base           /* archivos básicos necesarios para
compilar/linkear */
    │   ├── ceedling      /* ceedling utilidad para correr los unit
test */
    │   ├── drivers        /* drivers para el módulo posix */
    │   └── lcov          /* utilidad para analizar el coverage de los
unit test */
    └── lwip            /* stack tcpip */
    └── Makefile         /* makfile principal del proyecto */
    └── Makefile.config  /* plantilla para crear un Makefile.mine
propio */
    └── Makefile.mine    /* Makfile con la configuración del usuario
*/
    └── modules
        ├── base          /* módulos del Firmware */
        └── linkear        /* archivos básicos necesarios para compilar
y linkear */
            ├── ciaak         /* kernel de la ciaa */
            ├── drivers        /* drivers para posix */
            ├── libs           /* algunas librerías genéricas */
            ├── modbus          /* módulo para manejo de modbus */
            ├── plc             /* módulo de PLC (en desarrollo) */
            ├── posix           /* posix like library */
            ├── rtos            /* RTOS de CIAA-Firmware (FreeOSEK) */
            └── systests        /* utilidad para correr los tests en HW (en
```

- <http://proyecto-ciaa.com.ar/devwiki/>

Figura 4: Estructura de directorios del Firmware

Directorio "externals" (Software y Tools Externos)

Este directorio contiene el Software y Tools externos al CIAA-Firmware, que son necesarios para compilar, testear, etc. el Firmware. Tenga en cuenta que el Software y Tools en esta carpeta no son parte de CIAA-Firmware y pueden contener otras licencias. Sobre la 1 se ilustra los contenidos del directorio y su descripción.

ceedling	Tool utilizada para los Unit Tests o Pruebas Unitarias
base	Fuentes, headers y linker scripts necesarios para poder compilar y linkear el código en la plataforma
drivers	Drivers provistos por el proveedor del chip, los cuales son luego adaptados al formato de la CIAA

Cuadro 1: Tabla1

modules (out (Archivos de salida)

La 3 contiene todos los archivos generados por el CIAA-Firmware:

bin	Contiene el binario del proyecto, es el archivo que se va a correr en la PC o a cargar en el CIAA-Firmware
gen	Archivos generados de OSEK RTOS
lib	Por cada Módulo el make genera un archivo .a, osea una librería
obj	Todos los archivos fuentes son compilados a object files y almacenados en este directorio

Cuadro 2: Tabla 2

3.4. Iniciación a través de ejemplos

Sobre el Firmware de la placa se distribuyen varios ejemplos los cuales se encuentran en la carpeta **examples** y pueden ser utilizados como base para iniciar cualquier proyecto. Cualquiera de los ejemplos puede ser copiado y utilizado de base para nuevos proyectos. Por ejemplo con el siguiente comando: *cp -r examples/blinkingprojects/my_proyect*

Y adaptando el Makefile.mine indicado: *PROJECTPATH = projects/my_project.*

3.5. Instalación de IDE

El entorno de desarrollo integrado (IDE) posibilita el trabajo en un ambiente ameno, tambien provee las herramientas necesarias para el desarrollo de aplicaciones en el Firmware de forma automatica. La CIAA utiliza una version modificada de la plataforma de software (IDE) *Eclipse*,denominada *CIAA-Software-IDE* ,la cual contiene herramientas de programación tales como editor de texto, compilador,plataforma para depuración, etc. Sobre la página web del proyecto, se provee un instalador llamado *CIAA-IDE-SUITE*, desde donde se puede configurar automáticamente todas las herramientas necesarias para trabajar con la placa. Este instalador solamente es para los usuarios que poseen Windows XP o superior.

El paquete de instalación incluye:

1. **Eclipse**
2. **PHP (Hypertext Pre-processor)** es un lenguaje de programación de uso general de código desde el lado del servidor, originalmente diseñado para el desarrollo de contenido dinámico. En este caso, se utiliza solamente en forma de scripts para poder generar algunos archivos del **Sistema Operativo OSEK**
3. **Cygwin** es una consola que se ejecuta en Windows, de modo de emular la consola de comandos de Linux. Cuenta con todos los comandos, y el compilador GCC, propio del sistema operativo libre.

Una vez realizada la descarga del instalador, se ejecuta dicha aplicación, la figura 6 muestra el arranque del instalador, sobre ella, debe seleccionarse *Siguiente*



Figura 5: Arranque del instalador del software IDE

A continuación se presenta la siguiente ventana, sobre ella deben aceptarse los términos de uso:

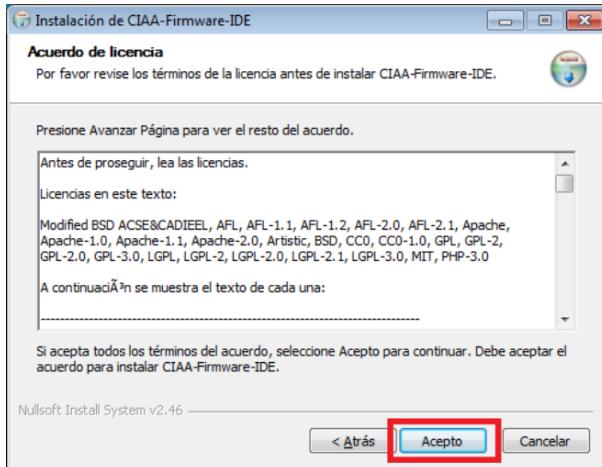


Figura 6: Arranque del instalador del software IDE

Sobre la ventana siguiente deben elegirse cuáles componentes se desea instalar, en el caso que el usuario no posea la placa disponible, no es necesario instalar los drivers, si se adquiere dicha placa en un momento posterior,dado que los drivers se instalan junto con el IDE, los mismos quedarán en la carpeta de destino para su instalación en forma manual; otra forma de instalar los los controladores es ejecutar el instalador del CIAA-IDE Suite y tildar únicamente la opción **drivers** al momento de seleccionar los componentes a instalar.La figura 7 ilustra lo explicado anteriormente.

A continuación debe establecerse la dirección en donde se desea instalar el entorno. La ventana que corresponde a este proceso se ilustra en la figura 8. En caso de que se desee cambiar dicha

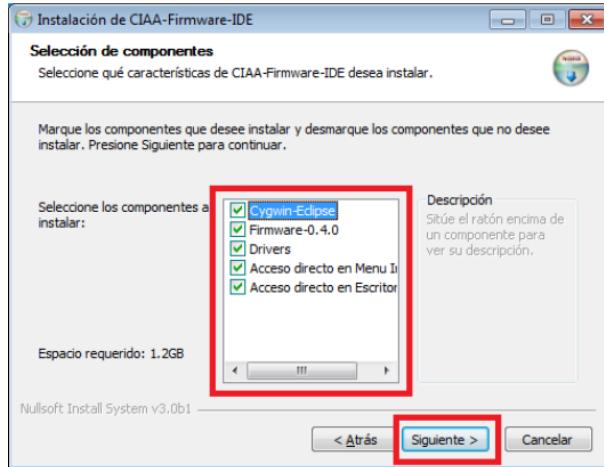


Figura 7: Selección de componentes del instalador

dirección, debe tenerse la precaución de no elegir una dirección donde los directorios posean espacios en sus nombres. Es recomendable no cambiar la unidad de instalación, pues en los siguientes pasos del documento se utilizarán direcciones que harán referencia a esta carpeta de instalación, y si se cambia, se deberán cambiar consecuentemente dichas direcciones.

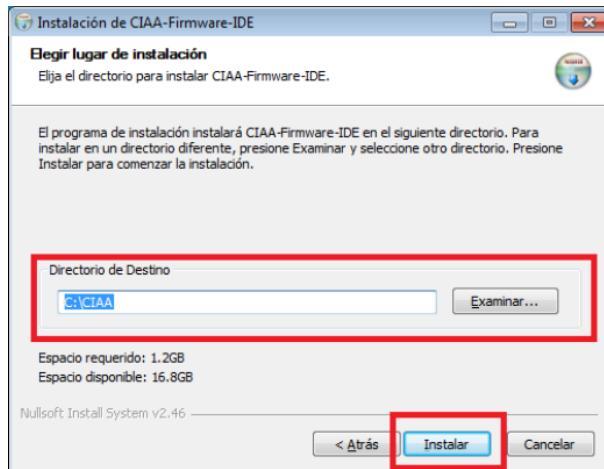


Figura 8: Elección de la ruta de instalación

Luego de dicha configuración, se inicia automáticamente el proceso de instalación. En un momento aparecerá una ventana emergente, similar a la que se muestra en la Figura 8 en donde el programa pregunta si disponemos del hardware, pues para la instalación del driver es necesario conectar la placa. De no ser así, aún puede continuar la instalación haciendo click en 'No'. Por el contrario, si disponemos de la EDU-CIAA, hacemos click en 'Yes', y emergirá otra ventana, como se muestra en la Figura 9



Figura 9: Instalación de los drivers: primera instancia

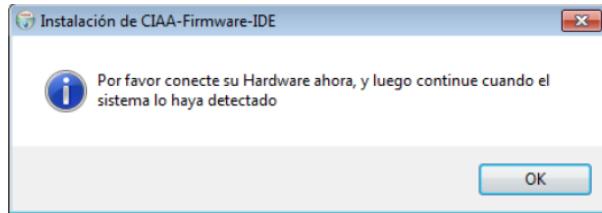


Figura 10: Instalación de drivers si se dispone del hardware

Una vez finalizada esta etapa, se procede a la instalación de los drivers por defecto del fabricante FTDI para puerto virtual. Este proceso se ilustra en las figuras 11 ,12,13

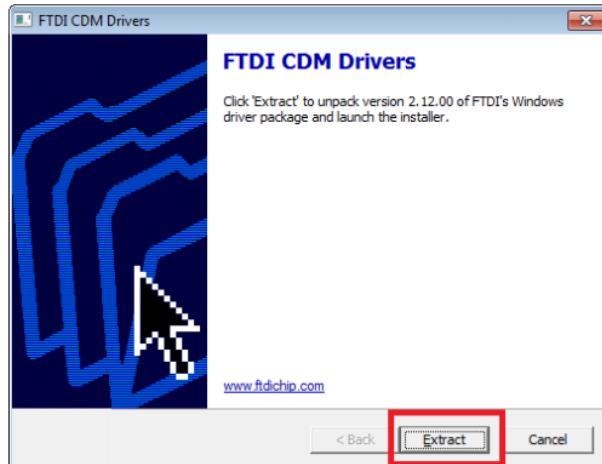


Figura 11: Instalador de drivers FTDI parte 1

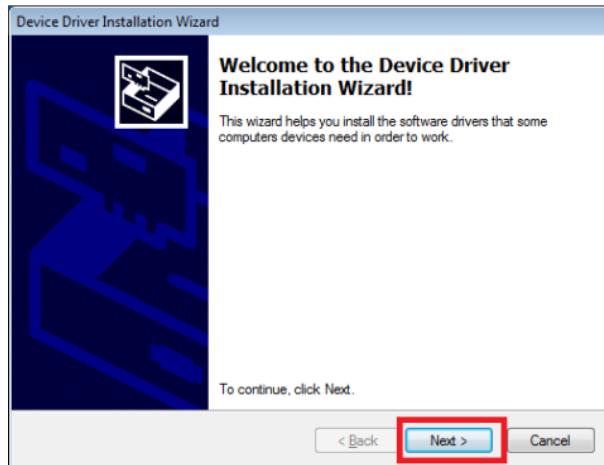


Figura 12: Instalador de drivers FTDI parte 2

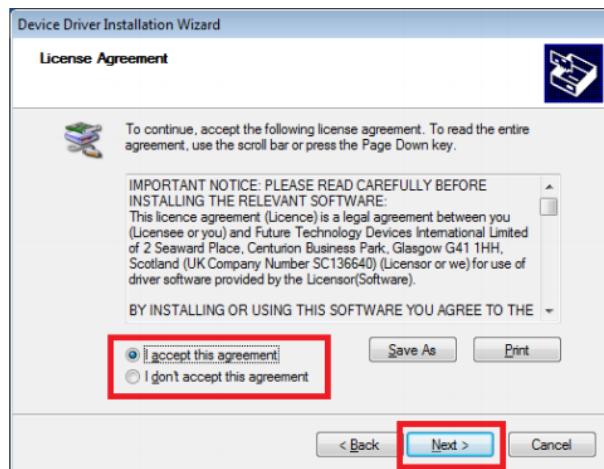


Figura 13: Instalador de drivers FTDI parte 3

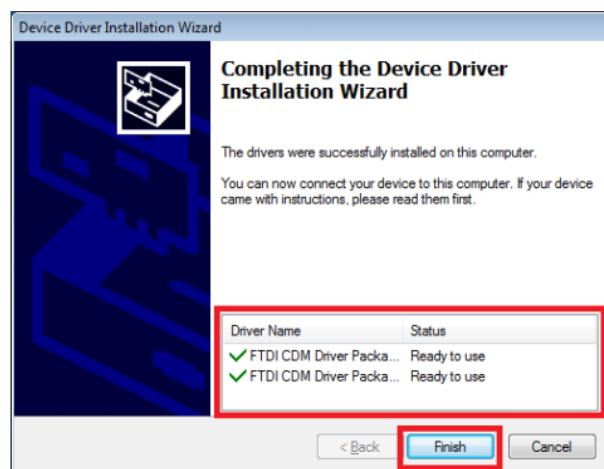


Figura 14: Instalador de drivers FTDI parte 4

Una de las posibles fallas que pueden surgir a través de este proceso, se presenta al producirse una falla en la comunicación a través del puerto virtual FTDI, que impide la correcta comunicación

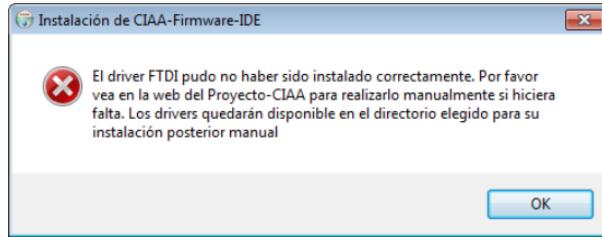


Figura 15: Instalador de drivers FTDI parte 4

entre la placa y el entorno IDE. Su corrección debe efectuarse manualmente, fuera del instalador, y es posible la aparición de una ventana de error emergente como la que se muestra en la figura 15.

El instalador incluye en la carpeta donde se instaló el software (Por defecto, $C : /CIAA$) un programa que configura el driver del controlador serie, emulado por la placa, para que uno de ellos pueda ser utilizado como interfaz JTAG. Dicho programa se llama *Zadig_Win_7_2_1_1.exe*, la figura 16 nos muestra una ilustración de la ubicación del programa.

Nombre	Fecha de modifica...	Tipo	Tamaño
cygwin	29/08/2016 13:05	Carpeta de archivos	
Firmware	29/08/2016 21:21	Carpeta de archivos	
IDE4PLC	28/08/2016 0:17	Carpeta de archivos	
Linux	28/08/2016 0:17	Carpeta de archivos	
local-repo	28/08/2016 0:16	Carpeta de archivos	
usbdriver	28/08/2016 0:17	Carpeta de archivos	
driver_winusb_zadig_ft2232h	10/04/2015 16:43	Imagen PNG	25 KB
Setup_Win_7_FTDI_2.12.00	08/06/2015 10:15	Aplicación	2.188 KB
SetUsers	31/03/2015 10:49	Archivo por lotes ...	2 KB
uninstall	29/08/2016 13:26	Aplicación	61 KB
zadig_Win_7_2.1.1	08/06/2015 10:16	Aplicación	5.069 KB

Figura 16: Instalador de drivers FTDI parte 4

Al abrir la aplicación, se presenta la Figura 17. Antes de proceder, el usuario debe conectar la placa, posteriormente, debe abrir el menú contextual *Options* y presionar sobre *List all devices*.

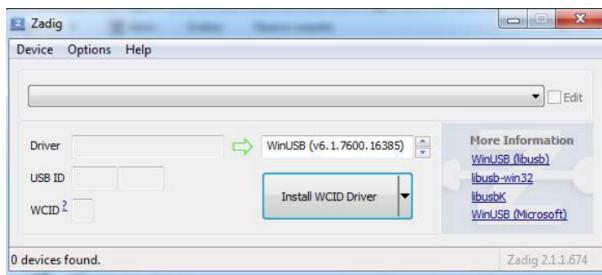


Figura 17: Entorno del software corrector Zadig para Windows

Aparecerá una lista de dispositivos de comunicación relacionados al USB. Tenemos que buscar aquellos cuyos nombres tengan relación con el puerto serie (puede aparecer Dual RS232-HS, USB Serial Converter, o algo similar).

Por lo general, aparecerán 2 con el mismo nombre, excepto que uno es Interface 0 y el otro Interface 1, como se muestra en la Figura 18 (la lista de drivers que se muestra puede diferir, dependiendo de la computadora que se utilice).

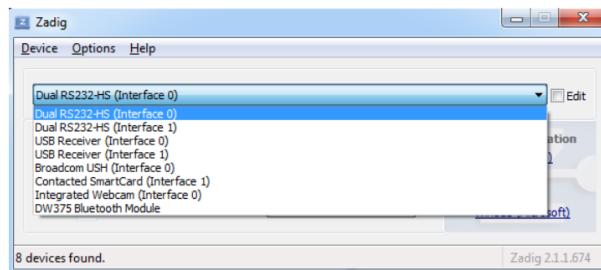


Figura 18: Lista de dispositivos vinculados a USB

Para configurar el driver:

- 1 seleccionar la **Interfase 0**
- 2 elegir el “**WinUSB v6.1**”
- 2 hacer click en el botón “**Replace Driver**”.

La Figura 19 muestra la ventana ya configurada:

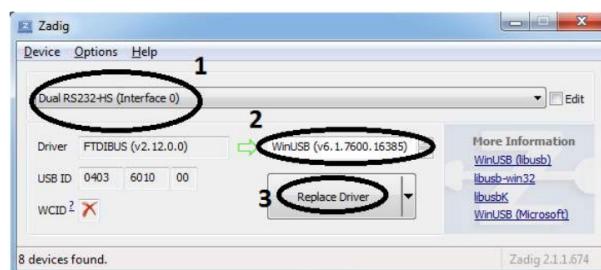


Figura 19: Configuración del Zadig para el reemplazo del driver

4. Desinstalación

Si se instaló el Software de CIAA-IDE y luego se desea desinstalarlo, se debe tener especial cuidado en quitar cualquier contenido que se quiera conservar de la carpeta *C : \ CIAA*, o el directorio de instalación elegido. Esto se debe a que el desinstalador del Software CIAA-IDE elimina el directorio y todo su contenido.

5. Configuración del entorno CIAA-IDE

Cuando se realiza la inicialización del entorno *CIAA-IDE*, dicho programa solicita al usuario que seleccione una carpeta de trabajo, la cual contendrá al Workspace, particularmente, es posible elegir que dicha carpeta se encuentre dentro del directorio de instalación del software.

La Figura 20 ilustra dicho concepto:

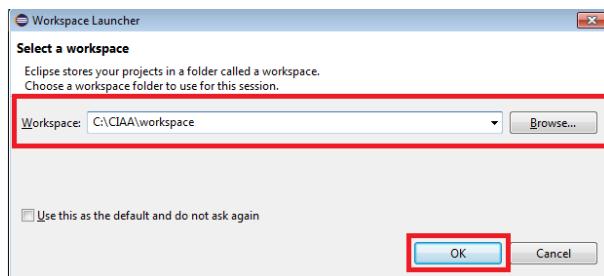


Figura 20: Selección de carpeta de trabajo

Debe notarse que todo lo relativo al trabajo que se esta realizando (como las configuraciones, variables globales, del entorno y demás) se guarda en esa carpeta. Si siempre se va a utilizar la misma ubicación, se puede tildar la opción “*Use this as the default and do not ask again*”; en otro caso, el programa siempre pide por esta ubicación, particularmente, este directorio de trabajo puede elegirse dentro del directorio de instalación del IDE.

5.1. Compilación de primer proyecto

Cuando un usuario que desea cerciorarse si ha realizado una correcta configuración del software que realiza la carga de un código sobre un microcontrolador, microprocesador, etc; es común trabajar sobre un ejemplo (usualmente provisto por el programa, o por los diseñadores de la placa, entre otros) el cual se lo suele llamar *blinking*. El objetivo de este programa es establecer una luz intermitente a través de un LED, cuyos intervalos de tiempo de encendido y apagado se encuentren temporizados a un cierto tiempo.

Sin más preambulo, para iniciar este proyecto debe seleccionarse sobre el menú contextual *File → New → Makefile Project with Existing Code*, como muestra la Figura 21.

En el caso que el usuario no haya instalado el Firmware, junto con el entorno CIAA-IDE, existen 2 opciones:

1. Puede descargar el repositorio del Firmware desde la web mediante la ejecución de la sentencia *git clone*.
2. Volver a ejecutar el instalador del software IDE de la CIAA, y sobre la ventana de componentes, seleccionar únicamente el item Firmware.

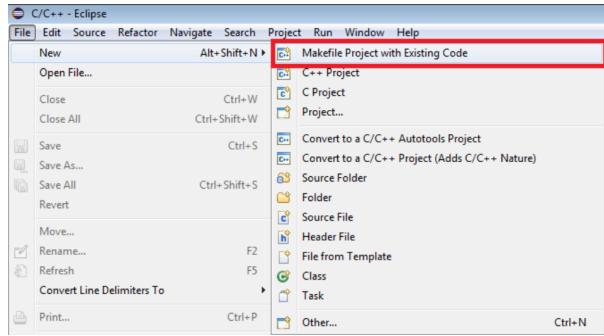


Figura 21: Carga de proyecto con código existente

A continuación, se presenta la ventana ilustrada en la figura 22, la cual solicita donde se encuentra el proyecto que se va a cargar, en este caso se elige la carpeta *Firmware*; la cual se encuentra por defecto sobre la ubicación

C : /CIAA/Firmware

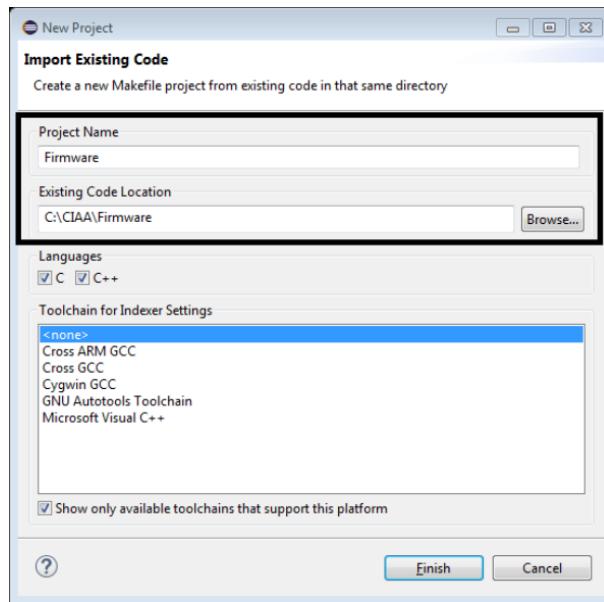


Figura 22: Carga de proyecto con código existente

Sobre la ventana *Toolchain for Indexer Settings* seleccionaremos la opción *none*, lo cual deja las opciones por defecto, configuradas en el Makefile.

Luego de oprimir *Finish*, se finaliza la creación del proyecto, al cerrar la ventana de bienvenida que nos muestra *Eclipse* nos encontramos con un entorno creado similar al de la figura 23

Una vez creado el proyecto, la siguiente tarea consiste en realizar la indexación de las cabeceras del estándar **POSIX**. **POSIX** es un conjunto de interfaces estándar para sistemas operativos

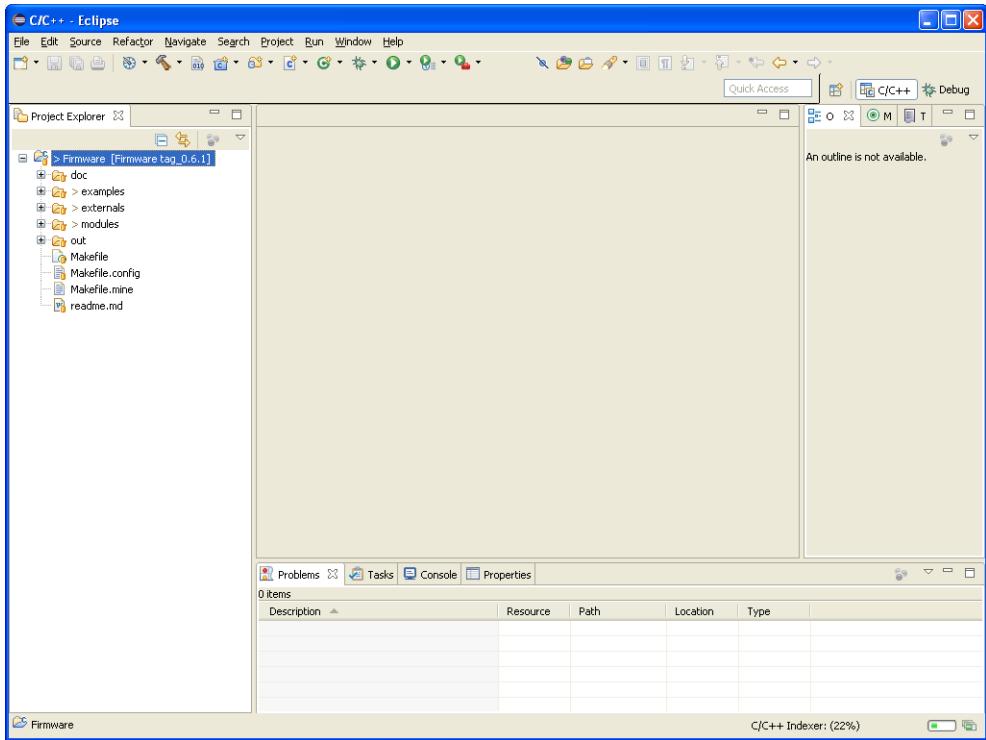


Figura 23: Carga de proyecto con código existente

basados en *UNIX*, la cual fue necesaria para que los distintos fabricantes de computadoras o desarrolladores de programas pudieran desarrollar sus aplicaciones independientemente de la plataforma sobre la cual iba a correr.

Para efectuar dichas indexaciones deben agregarse los archivos *Includes* del *GCC*. Esto se realiza configurando el proyecto creado y oprimiendo sobre la pestaña *C/C++ General*, luego se selecciona la opción *Paths and Symbols*, para indicar las rutas que proporcionarán las inclusiones de las cabeceras. La Figura 24 nos ilustra la ventana y las configuraciones descriptas anteriormente.

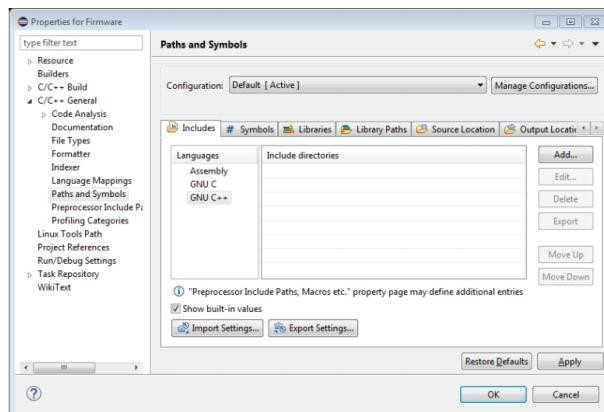


Figura 24: Indexación de las definiciones POSIX

En esta ventana se debe seleccionar '**GNU C**' o '**GNU C++**' según las POSIX que se desea agregar; luego se presiona el botón *Add*. Una vez oprimido dicho botón, emerge una ventana como

la mostrada en la Figura 25, en la cual se debe hacer click sobre la opción *File System ...*, luego buscar la carpeta correspondiente a agregar.

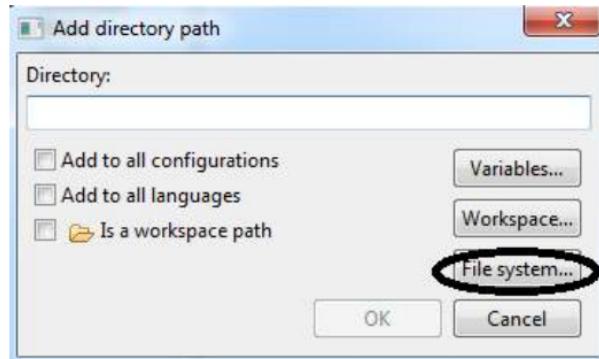


Figura 25: Indexación de POSIX

Los **Includes** que deben configurarse, según el lenguaje (Language) elegido, son los siguientes:

Language = 'GNU C'

- < *dIDE* >/cygwin/lib/gcc/usr/include Si se aplican los directorios por defecto, éste quedaría **C:/CIAA/cygwin/usr/include**
- < *dIDE* >/cygwin/lib/gcc/i686-pc-cygwin/4.9.2/include (por defecto: **C:/CIAA/cygwin/lib/gcc/i686-pc-cygwin/4.9.2/include**)
(dIDE = directorio de instalación del software-IDE: por defecto es **C:/CIAA**)

Language = 'GNU C++'

- < *dIDE* > /cygwin/usr/include
(por defecto: **C:/CIAA/cygwin/usr/include**)
- < *dIDE* > /cygwin/lib/gcc/i686-pc-cygwin/4.9.2/include/c++
(por defecto: **C:/CIAA/cygwin/lib/gcc/i686-pc-cygwin/4.9.2/include/c++**)
(dIDE = directorio de instalación del software-IDE: por defecto es **C:/CIAA**)

La ventana superior sobre la Figura 25 ilustra como quedan los includes en caso de seleccionar Lenguaje C, mientras que la ventana inferior ilustra como quedan los includes en caso de seleccionar Lenguaje C++.

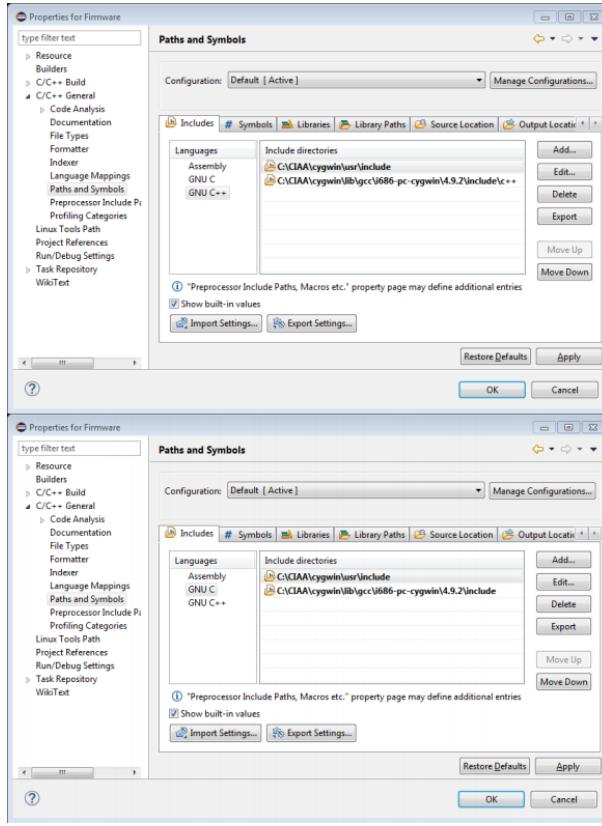


Figura 26: Configuración de includes

5.1.1. Configuración del Makefile

El *makefile* es un fichero de texto plano que define una serie de reglas con el objetivo de automatizar la compilación del código de forma simple y organizada, a través de la aplicación *Make*; la cual es una herramienta de gestión de dependencias, para dirigir su recompilación o "generación". Téngase en cuenta que si bien su función básica consiste en determinar automáticamente qué partes de un programa necesitan ser recompilados, y ejecutar los comandos necesarios para hacerlo, también se encarga de actualizar automáticamente un conjunto de archivos a partir de otro, cada vez que éste cambie.

Los archivos **Makefile** son archivos de texto escritos con una sintaxis predeterminada. Junto con la utilidad '**Make**', permiten construir el software desde sus archivos-fuente, en el sentido de organizar el código, su compilación y enlace (link) correcto.

El Proyecto CIAA tiene su propio Makefile, por lo que se debe indicarle al IDE cómo manejarse con él: de lo contrario generaría un Makefile automáticamente (lo cuál no es conveniente debido a que se tendría que formular sistemáticamente las reglas para la compilación del código).

La primera vez que se compila el proyecto, es necesario que el usuario aplique *Clean Project*. Esto ejecuta el comando *Clean_generate* del make, creando todos los archivos necesarios para la compilación con el RTOS.

El motivo por el cuál se ejecuta dicho comando, reside en el hecho de que al borrar los archivos objeto generados previamente, debe actualizarse el código PHP correspondiente al sistema operativos *RTOS OSEK*. El comando *Clean_generate* realiza ambas operaciones de forma consecutiva y automática. Si se trabaja sin RTOS-OSEK, sólo hace falta usar el comando **clean**.

Para efectuar esta configuración, el usuario debe ubicarse sobre la ventana de propiedades del proyecto **Firmware** y seleccionar la rama **C/C++ Build**; la Figura 27 ilustra la ventana que se presenta.

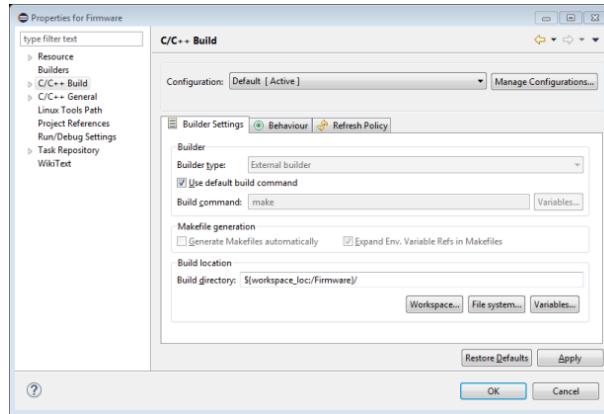


Figura 27: Configuración de makefile

Dentro de la rama 'C/C++ Build', configurar la pestaña 'Behaviour' como muestra la Figura 28.

Las configuraciones importantes son las siguientes:

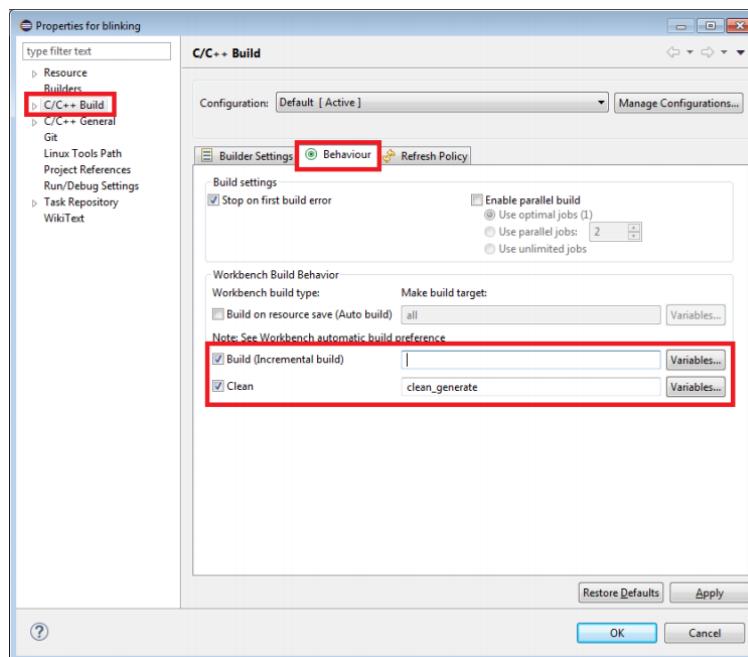


Figura 28: Configuración de comportamiento del IDE con el MakeFile

- Tildar la opción '*Stop on first build error*' y destildar '*Enable parallel build*'.

- Destildar el casillero ‘Build on resource save’ y tildar ‘Build (Incremental Build)’ y ‘Clean’. En el campo Clean, escribir: *clean_generate*
- Borrar el contenido del campo Build y dejarlo en blanco.

La Figura 28 ilustra la ventana resultante de la configuración

Al finalizar, debe presionarse en Ok, y luego dirigirse sobre el proyecto Firmware, debe oprimirse clic derecho, y buscar la opción *Clean Project*. Por último, se debe ejecutar *Build Project*. Dichos opciones se muestran en la Figura 29, y si todo sale correctamente, en la consola del IDE debería mostrar una ventana como la que se muestra en la Figura 30. Particularmente, vamos a ver una línea que nos dice que se ha creado un archivo *Blinking.axf*.

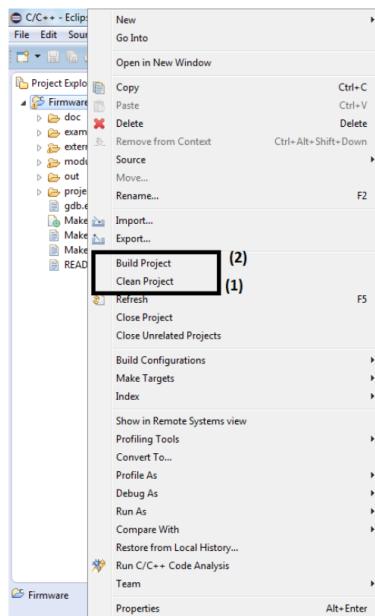


Figura 29: Menú de opciones del proyecto para hacer clean y build Project

```
=====
Creating library lib
ar -crs ./out/lib/lib.a ./out/obj/ciaalibs_Maths.o ./out/obj/ciaalibs_Matrix.o ./out/obj/ciaalibs_PoolBuf.o ./out/obj/ciaalibs_CircBuf.o

=====
Compiling c file: examples/blinking/src/blinking.c
gcc -c -Wall -ggdb3 -c -Wall -ggdb3 -I`cygpath -w examples/blinking/inc` -I./modules/posix/inc -I./modules/ciaak/inc -I./modules/drivers/inc -I./modules/drivers/x86/inc -I./modules/drivers/rtos/inc -I./modules/drivers/ext_drivers/inc
Generating dependencies...
gcc -MM -c -Wall -ggdb3 -c -Wall -ggdb3 -I`cygpath -w examples/blinking/inc` -I./modules/posix/inc -I./modules/ciaak/inc -I./modules/drivers/inc -I./modules/drivers/x86/inc -I./modules/drivers/rtos/inc -I./modules/drivers/ext_drivers/inc
=====
Linking file: ./out/bin/blinking.exe
gcc ./out/obj/blinking.o -Lalinker --start-group ./out/lib/posix.a ./out/lib/ciaak.a ./out/lib/drivers.a ./out/lib/ext_drivers.a ./out/lib/rtos.a ./out/lib/libs.a -Lalinker --end-group
=====
Post Building blinking
|_
07:39:16 Build Finished (took 5m:26s.91ms)
```

Figura 30: Consola del software IDE luego de una compilación exitosa

5.2. Depuración con Windows

Es posible que el usuario desee probar este ejemplo para ver de qué manera interactúa el software creado con el hardware de la EDU-CIAA, o verificar que no hay errores en nuestra lógica de programación.

Para ello, debe compilarse el proyecto, y luego efectuar una depuración del código. Sin embargo, si no se cuenta con el hardware, no es posible realizar ésta acción, no obstante, se ha desarrollado para esos casos una aplicación para el ejemplo *Blinking* que hace funcionar la aplicación del Firmware sobre la PC del usuario, brindando respuestas similares a la que proporciona la placa. En este caso, se pretende que aparezca un mensaje sobre la consola del software IDE, de manera que aparezca un cartel que indique que se *"enciende"* el Led.

Dado que el código se encuentra configurado para brindar un archivo *.exe*, el usuario debe realizar ciertas configuraciones previas; en primer lugar debe especificarse al IDE la ruta donde se encuentra el ejecutable que ha generado el *Makefile*; para ello, debe oprimir clic derecho sobre el proyecto, seleccionar '**Debug as**', y a continuación '**Debug Configurations**', la Figura 31 ilustra lo expresado anteriormente.

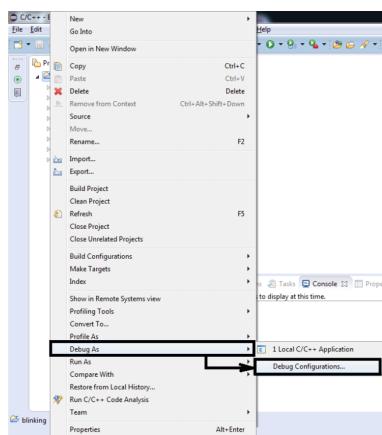


Figura 31: Acceso a Debug Configurations

Una vez hecho esto, se oprime doble click en la opción '**C/C++ Application**', lo cual creará un módulo donde es posible configurar un programa para que emule el comportamiento de la placa. El resultado se muestra en la Figura 32.

El ejecutable que genera el *Makefile* se almacena en la carpeta < Carpeta de instalación del software-IDE > **/Firmware/out/bin/** bajo el nombre de '*blinking.exe*'. La configuración final se muestra en la Figura 30, y si se usan las rutas por defecto, los valores son los siguientes:

- C/C++ Application: **C:/CIAA/Firmware/out/bin/blinking.exe**
- Project: **blinking**
- Tildar la opción '**Use Workspace settings**'

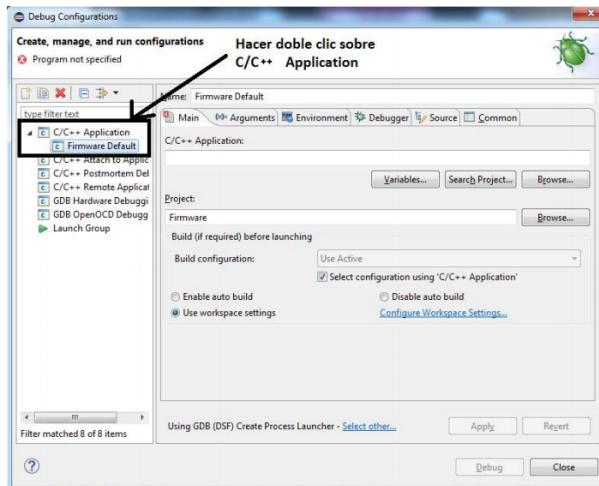


Figura 32: Creación de una opción de depuración por medio de un emulador

- Tildar la opción ‘Select configuration using ‘C/C++ Application’

Es posible que el IDE informe que no puede encontrar el Código Fuente cuando se intenta hacer Debug sobre este proyecto. Aparecerá una leyenda en el medio de la consola que se parece a la de la Figura 34. Para solucionarlo se debe crear un Path Mapping que convierta el estilo de ruta de CygWin al formato Windows. El usuario debe abrir la ventana de configuración del software IDE haciendo click en '*Windows* → *Preferences*'. Seleccione la rama ‘C/C++’, luego la rama ‘Debug’ y ubique la opción ‘Source Lookup Path’.

Finalmente, en el lado derecho debe oprimirse el botón ‘Add’: Luego aparece una ventana como la que se muestra en la Figura 35.

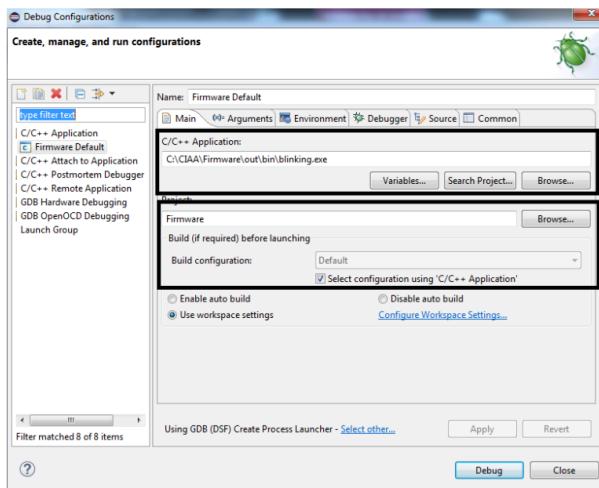


Figura 33: Configuración del emulador para el proyecto Blinking

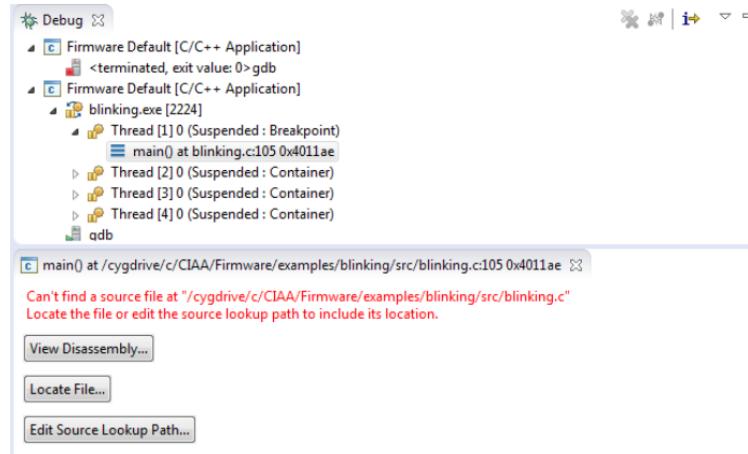


Figura 34: Error de IDE. No se encuentra el código fuente

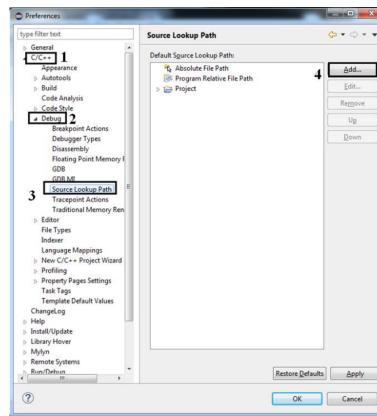


Figura 35: Configuración de mapeo de rutas para el entorno Windows

Si el usuario desea agregar una fuente (Source), se muestra el recuadro de la Figura 36, donde puede elegirse el tipo '**Path Mapping**'.

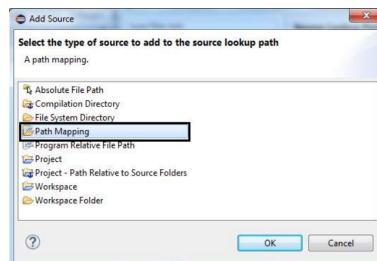


Figura 36: Agregando un Path Mapping

A continuación aparece un nuevo recuadro, como el de la Figura 37, donde se debe elegir la opción '**Add**', y llenar con los siguientes datos:

- Name: **Firmware**

- Compilation Path: `/cygdrive/c/CIAA/Firmware`
- Local file system path: `C : /CIAA/Firmware` (si usamos los directorios por defecto)

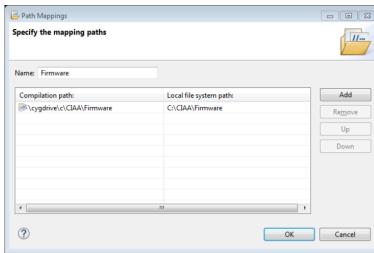


Figura 37: Configuración para crear un Path Mapping en Windows

Con todas estas configuraciones es posible depurar el ejemplo Blinking, emulando las respuestas a través de la consola.

IMPORTANTE: Si bien la creación de un Path Mapping nos permite emular el comportamiento del titileo de LEDs, se producen conflictos cuando se depura con la propia placa, pues bloquea el uso de breakpoints.

Si se cuenta con la placa y se desea hacer Debug con la misma, debe borrar el *Path Mapping* creado.

5.3. Depuración en la placa

Durante la instalación del CIAA-IDE, además de los drivers para la conexión de la placa se instalan las herramientas para Debug.

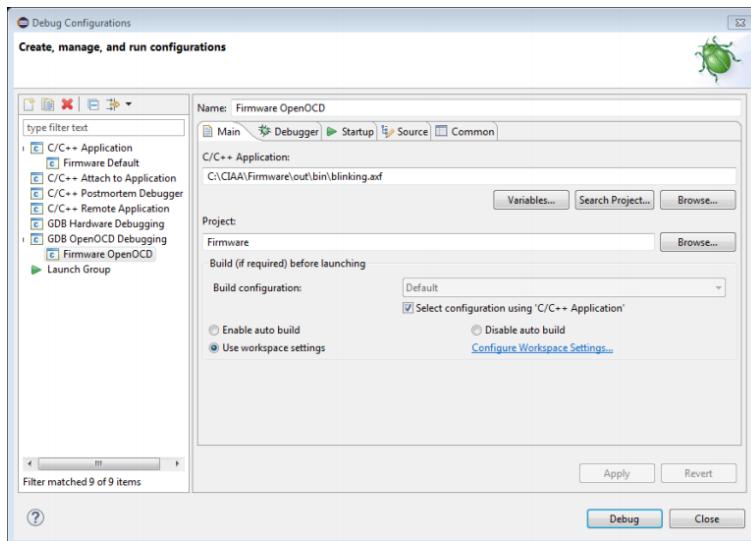


Figura 38: Configuración para poder hacer Debug sobre la placa: pestaña Main

Se utiliza OpenOCD (una herramienta OpenSource creada para estos propósitos) para hacer el nexo JTAG-GDB mediante la conexión USB. Para configurar esta herramienta, desde el menú

Run → DebugConfigurations... se debe crear un módulo nuevo de '**Debug configuration**' del tipo '**GDB OpenOCD Debugging**'. A continuación, coloque los valores que se muestran en las Figuras 38 y 39. Las mismas se listan a continuación:

- Pestaña Main:

- 1 Name:**Blinking OpenOCD**
- 2 Project:**blinking**
- 3 C/C++ Application:**C:/CIAA/Firmware/out/bin/blinking.axf**
- 4 Tildar **Disable auto build**
- 5 Build configuration: Default

- Pestaña Debugger:

- 1 OpenOCD Setup: Destildar la opción *Start OpenOCD locally*
- 2 a Executable: **C:/CIAA/cygwin/usr/arm-none-eabi/bin/arm-none-eabigdb.exe**
 - b *Other options* y *Commands*: dejarlo como está
 - c Destildar '*Force thread list update on suspend*'

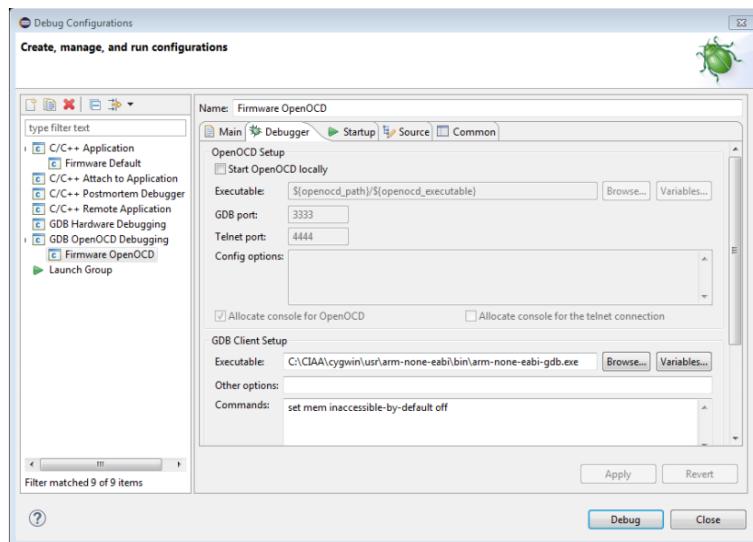


Figura 39: Configuración para poder hacer Debug sobre la placa: pestaña Debugger

Para hacer Debug, es necesario que se abra un puerto a través del OpenOCD. Hay muchas formas de hacerlo. En este tutorial, lo haremos usando la consola CygWin, que viene incluida con el instalador del CIAA-IDE.

Al abrir la consola. Aparece una ventana como la que se muestra en la Figura 40

Entonces, se escriben los siguientes comandos:

Luego de iniciado el servicio del OpenOCD, se mantiene la ventana abierta, al volver al entorno IDE, se oprime el botón Debug de la ventana que dejamos abierta previamente.

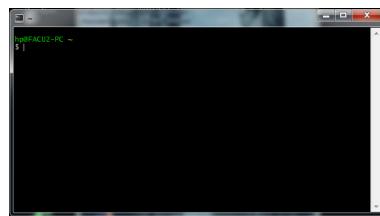


Figura 40: Consola CygWin

Comando	Descripción
cd C:/CIAA/Firmware	Nos posiciona en la carpeta Firmware (si usamos los Path por defecto)
make openocd	Comienza a correr el servicio del OpenOCD

Cuadro 3: Tabla 2

5.3.1. Possible Problema: No reconocimiento

Si se desconecta la placa de la PC y se la vuelve a conectar en un puerto USB distinto al anterior, Windows detectará el nuevo Hardware encontrado, instalará los drivers y la placa se encenderá; pero al intentar hacer Debug, en la consola CygWin es posible que emerja un mensaje de error como la de la Figura 41.

```

/cygdrive/c/CIAA/Firmware
Joaquin@Joaquin /cygdrive/c/CIAA/Firmware
$ make openocd
=====
Starting OpenOCD...
openocd -f ./modules/tools/openocd/cfg/cortexM4/lpc43xx/lpc4337/ciaa-nxp.cfg
Open On-Chip Debugger 0.8.0 (2014-04-28-08:42)
Licensed under GNU GPL v2
For bug reports, read
      http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
adapter speed: 2000 kHz
none separate
cortex_m reset_config vectreset
Error: no device found
Error: unable to open ftdi device with vid 0403, pid 6010, description '' and s
erial ''
in procedure 'init'
Makefile:510: Fallo en las instrucciones para el objetivo 'openocd'
make: *** [openocd] Error 1
Joaquin@Joaquin /cygdrive/c/CIAA/Firmware
$ |

```

Figura 41: Consola CygWin

Si se hiciera un par de pruebas, este error también corresponde al que se daría si la placa no estuviera conectada. La causa es que el driver que se acaba de instalar no es el corregido por el programa Zadig, sino es el instalado por el instalador del Software-IDE. Para solucionar este inconveniente simplemente hay que repetir los pasos que se destacan sobre el tutorial, y con ello, la placa vuelve a funcionar correctamente.

6. Primeros Pasos con la EDU-CIAA

6.1. Programación en Baremetal

Si bien el objetivo de este tutorial es introducir al usuario en la programación mediante *RTOS OSEK*, es conveniente comenzar a explicar previamente como implementar proyectos en Bareme-

tal, éste método de programación se destaca por no utilizar SO alguno; de modo que permite una transición más natural hacia los temas posteriores.

El primer ejemplo que se presenta ejecuta el encendido intermitente de los cuatro leds instalados en la placa. Sobre las placas del tipo *LPC43XX/LPC43SXX*, los pines digitales se encuentran agrupados en 16 conjuntos de pines, los cuales se encuentran definidos desde P0 a P9, y desde PA hacia PF; cada uno de estos pines soporta hasta 8 diferentes funciones digitales, entre ellas se incluye la función *General Purpose I/O*, el cual es seleccionable a través de la configuración de registros de pines SCU. La Unidad de Control del Sistema (SCU) determina la función y el comportamiento eléctrico de la mayoría de los pines del LPC4337, por defecto, se selecciona la función digital 0 (que corresponde a la habilitación de las resistencias de pull-up) para todos los pines.

Algunos de estos pines, soportan la multiplexación entre funciones digitales y analógicas, sin embargo, todas las entradas y salidas de los pines que corresponden a el ADC y/o DAC pueden configurarse sin establecer esta multiplexación, además, el SCU contiene registros que establecen la configuración de la función del pin multiplexado.

Otra funcionalidad que utiliza este ejemplo es el *RIT (Repetitive Interrupt Timer)*, el cual provee una forma versátil de generar interrupciones repetitivas en intervalos de tiempo específicos, y puede utilizarse como alternativa al *Systick Timer*. Dicho timer consta de un contador de 32 bits que puede incrementarse de forma libre, o puede reiniciarse mediante una interrupción generada. El funcionamiento de este Timer puede describirse de la siguiente forma:

A partir del reinicio, el contador empieza a incrementar desde 0x00000000. Cuando el valor del contador iguala el valor programado en el registro *COMPVAL*, se establece en alto el flag que corresponde a la interrupción del Timer. Cualquier combinación de bits del registro *COMPVAL* pueden ser enmascarada a través de un registro *MASK*. Si el bit *enable_clr* se encuentra en bajo (estado por defecto), una comparación válida solamente causa que el flag de interrupción se establezca; mientras que si el bit *enable_clr* se encuentra en alto, se establece el reinicio del conteo.

Este ejemplo utiliza la librería LPCOpen correspondiente a la placa LPC4337, con el objetivo de permitir la configuración correspondiente de los pines digitales y de la configuración del timer, independizándose de el hardware utilizado, esto implica que el usuario no necesita conocer en profundidad la arquitectura del procesador.

Referencias