Índice

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 modificaciónes 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

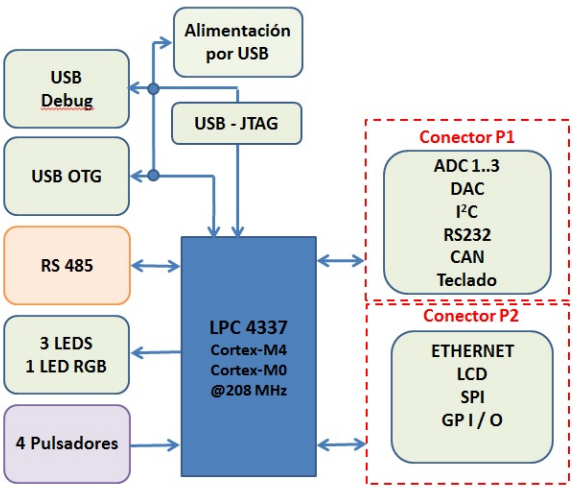

Figura 1: Diagrama en bloques de EDU CIAA basado en LPC4337.

La CIAA es una plaqueta 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.

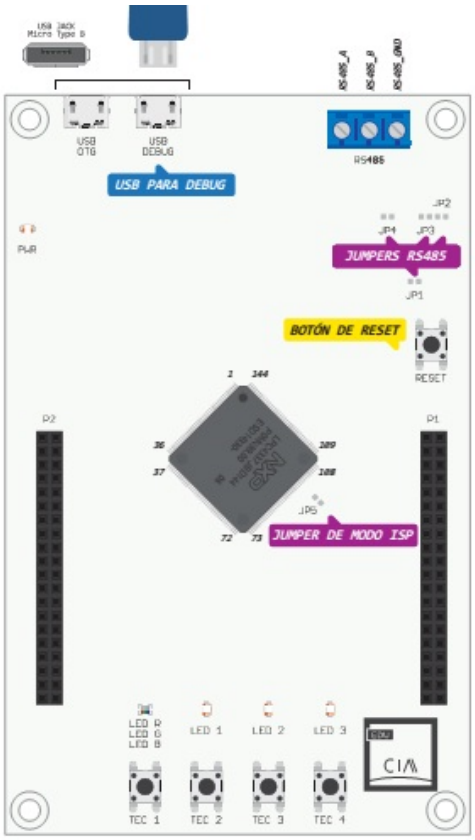

Figura 2: Imagen frontal de placa

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.

Sobre la Figura 1 1 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 2 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 3 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/0.

3. Instalacion de Software

3.1. Conceptos previos

El desarrollo de codigos 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 *Open OCD* 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 traves 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 configuracion 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 trabajaca (CIAA-NXP, EDU-CIAA-NXP, etc.)
6. **PROJECT** es el Path al proyecto a compilar. Ej: examples$(*DS*)$blinking_base$.

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/blinking** el makefile del ejemplo se encuentra en **examples/blinking/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                     /* módulos del Firmware */
│   ├── base                    /* archivos básicos necesarios para compilar
y linkear */
│   ├── ciaak                   /* kernel de la ciaa */
│   ├── drivers                 /* drivers para posix */
│   ├── libs                    /* algunas librerias 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 3: 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 platafor
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 2 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-Firm
gen	Archivos generados de OSEK RTOS
lib	Por cada Módulo el make genera un archivo .a, osea una libreria
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: $PROJECT_PATH = projects/my\_project$.

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 5 muestra el arranque del instalador, sobre ella, debe seleccionarse *Siguiente*

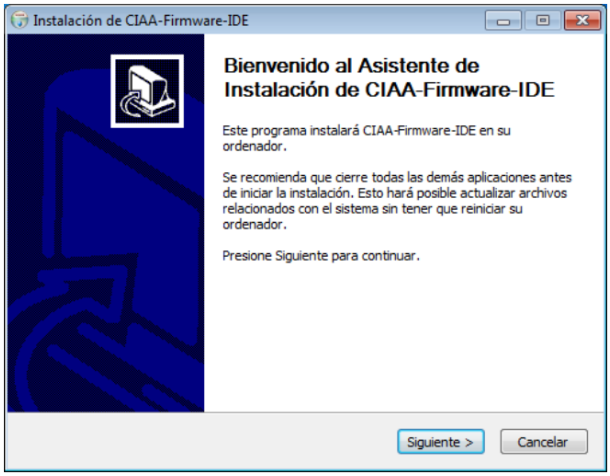

Figura 4: Arranque del instalador del software IDE

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

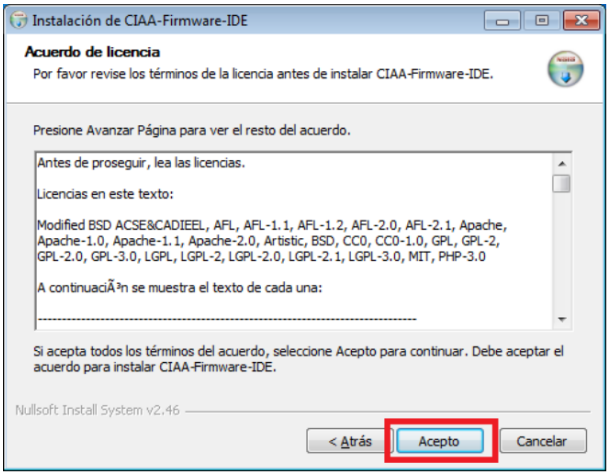

Figura 5: Arranque del instalador del software IDE

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 6 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 ilusta en la figura 7. En caso de que se desee cambiar dicha

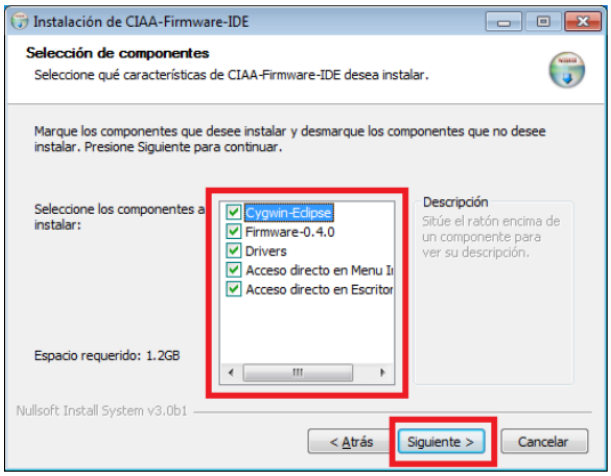

Figura 6: Selección de componentes del instalador

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.

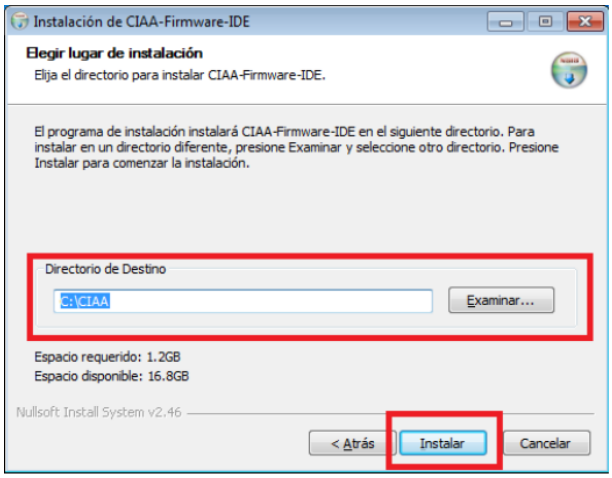

Figura 7: Elección de la ruta de instalación

4. Conclusiones

Referencias