

UNIVERSIDAD DE BUENOS AIRES  
FACULTAD DE INGENIERÍA  
MAESTRÍA EN SISTEMAS EMBEBIDOS



MEMORIA DEL TRABAJO FINAL

**sAPI (simpleAPI): diseño e  
implementación de una biblioteca para  
sistematizar la programación de sistemas  
embebidos**

**Autor:**  
**Esp. Ing. Eric Nicolás Pernia**

Director:  
MSc. Ing. Félix Gustavo E. Safar.

Jurados:  
Dr. Ing. Pablo M. Gómez (UBA)  
Mg. Ing. Pablo O. Ridolfi (UTN FRBA)  
Ing. Juan Manuel Cruz (FIUBA,UTN-FRBA)

*Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires, entre marzo de 2018 y diciembre de 2018.*



*sAPI (simpleAPI): diseño e implementación de una biblioteca para sistematizar la programación de sistemas embebidos por Esp. Ing. Eric Nicolás Pernia se distribuye bajo una **Licencia Creative Commons Atribución-CompartirIgual 4.0 Internacional**. Para ver una copia de esta licencia, visita:*

<http://creativecommons.org/licenses/by-sa/4.0/>.

## *Resumen*

En esta memoria se presenta el diseño de una biblioteca para la programación de sistemas embebidos portable entre plataformas de hardware y lenguajes de programación. Se realizó una implementación de referencia en lenguaje C para las plataformas del Proyecto CIAA. Se realizó un banco de pruebas de hardware, junto a la utilización de testeo unitario e integración continua para la validación.

Además de la biblioteca, se creó una herramienta de código abierto para desarrolladores que automatiza la implemetación de bibliotecas a partir de un modelo que las describe. De esta manera se facilita la futura ampliación de la biblioteca y su implementación en otras plataformas de hardware.



## *Agradecimientos*

Agradecimientos personales. [OPCIONAL]

No olvidarse de agradecer al tutor.

No vale poner anti-agradecimientos (este trabajo fue posible a pesar de...)



# Índice general

<b>Resumen</b>	<b>III</b>
<b>1. Introducción General</b>	<b>1</b>
1.1. Contexto y justificación . . . . .	1
1.2. Motivación . . . . .	2
1.2.1. Lenguajes de programación y Hardware en Sistemas Embedidos . . . . .	2
1.2.2. Bibliotecas para microcontroladores ofrecidas por los fabricantes . . . . .	2
1.2.3. Proyecto CIAA . . . . .	2
1.3. Objetivos y alcance . . . . .	4
1.3.1. Objetivos . . . . .	4
1.3.2. Alcance . . . . .	4
<b>2. Introducción Específica</b>	<b>7</b>
2.1. Plataformas del Proyecto CIAA . . . . .	7
2.1.1. CIAA-NXP . . . . .	8
2.1.2. EDU-CIAA-NXP . . . . .	9
2.1.3. Pico-CIAA . . . . .	10
2.1.4. CIAA-Z3R0 . . . . .	11
2.1.5. Arquitectura de Hardware . . . . .	12
2.1.6. Bibliotecas disponibles . . . . .	12
2.2. Requerimientos . . . . .	12
2.3. Planificación . . . . .	13
<b>3. Diseño</b>	<b>15</b>
3.1. Descripción general . . . . .	15
3.2. Modelado de plataforma de hardware genérica . . . . .	15
3.2.1. Clasificación de módulos de hardware . . . . .	15
3.2.2. Plataforma y componentes . . . . .	15
3.2.3. SoC . . . . .	15
3.2.4. Núcleo de procesamiento (Core) . . . . .	15
3.2.5. GPIO . . . . .	15
3.2.6. ADC . . . . .	15
3.2.7. DAC . . . . .	15
3.2.8. TIMER . . . . .	15
3.2.9. RTC . . . . .	15
3.2.10. UART . . . . .	15
3.2.11. SPI . . . . .	15
3.2.12. I2C . . . . .	15
3.3. Verificación del modelo . . . . .	15
3.4. Modelo abstracto de biblioteca . . . . .	15
3.4.1. Módulo de biblioteca . . . . .	15

3.4.2. Biblioteca . . . . .	15
3.5. Diseño de archivo de texto para la descripción de una plataforma de hardware . . . . .	15
<b>4. Implementación</b>	<b>17</b>
4.1. Descripción general . . . . .	17
4.2. Archivos de descripción de las paltaformas del proyecto CIAA . . .	17
4.2.1. CIAA-NXP . . . . .	17
4.2.2. EDU-CIAA-NXP . . . . .	17
4.2.3. Pico-CIAA . . . . .	17
4.2.4. CIAA-Z3R0 . . . . .	17
4.3. Generación automática de código fuente . . . . .	17
4.4. Implementación del código C dependiente del hardware . . . . .	17
4.5. Ejemplos de utilización . . . . .	17
4.6. Documentación y difusión . . . . .	17
4.6.1. Generación automática de manual de referencia de la API en base al modelo . . . . .	17
4.6.2. Tutoriales de instalación y uso . . . . .	17
4.6.3. Difusión a la comunidad del Proyecto CIAA y Embebidos	32
5. Ensayos y Resultados	19
5.1. Testeo Unitario . . . . .	19
5.2. Banco de pruebas de hardware . . . . .	19
5.3. Integración continua . . . . .	19
5.4. Utilización de la biblioteca para la enseñanza de programación de Sistemas Embebidos . . . . .	19
<b>6. Conclusiones</b>	<b>21</b>
6.1. Trabajo realizado . . . . .	21
6.2. Próximos pasos . . . . .	22

# Índice de figuras

2.1. Plataforma CIAA-NXP . . . . .	9
2.2. Plataforma EDU-CIAA-NXP . . . . .	11
2.3. Plataforma PicoCIAA. . . . .	11
2.4. Plataforma CIAA-Z3R0. . . . .	12



# Índice de Tablas

2.1. Planificación del Trabajo Final. . . . .	14
---	----



*Dedicado a... [OPCIONAL]*



# Capítulo 1

## Introducción General

### 1.1. Contexto y justificación

*La idea de esta sección es presentar el tema de modo que cualquier persona que no conoce el tema pueda entender de qué se trata y por qué es importante realizar este trabajo y cuál es su impacto.*

En la actualidad la programación de plataformas basadas en microcontrolador se realiza su mayoría en lenguaje C utilizando bibliotecas para el manejo del núcleo de procesamiento (o los núcleos) y periféricos. En consecuencia, una biblioteca de C es parte integral de cualquier diseño de sistema embebido basado en microcontrolador.

En este trabajo se presenta un diseño de biblioteca de programación de plataformas de Sistemas Embebidos basadas en microcontrolador de forma sencilla. La misma permite utilizar los modos más comunes de los periféricos típicos de un microcontrolador (tanto del núcleo de procesamiento como sus periféricos). Se expone además las características de implementación de referencia sobre la plataforma EDU-CIAA-NXP.

En la actualidad existe una enorme variedad plataformas de Sistemas Embebidos en el mercado, y si bien todas cuentan con dispositivos programables con características similares y periféricos compatibles, se observa que en la práctica son muy distintos. En consecuencia se debe invertir mucho tiempo en aprender a programar cada una de ellas, con sus particularidades antes de su utilización. Estas diferencias se deben a varios factores:

- Los fabricantes de los dispositivos programables y empresas asociadas carecen de diseños estándar de arquitectura de hardware (tanto en núcleos de procesamiento, como periféricos). Si bien esto trae el beneficio de permitir elegir el dispositivo programable que más se adecúe a un proyecto, también es la principal causa de la necesidad de conocer en detalle la arquitectura en particular.
- Dichas empresas en su mayoría se limitan a ofrecer información de bajo nivel para programar el hardware directamente, o bien, sus propias bibliotecas escritas en lenguaje C, que están diseñadas con una gran dependencia de la arquitectura de hardware subyacente, es decir, carecen de abstracción del hardware.

Se están portando cada vez más lenguajes de programación a las plataformas de Sistemas Embebidos, que antes se reservaban para las computadoras de propósito general (como la PC). Esto se debe a que las nuevas plataformas poseen más

poder de procesamiento y debido a la complejidad creciente de las aplicaciones se exige más características y abstracción a los lenguajes de programación.

Existen algunos desarrollos de bibliotecas que logran una abstracción de hardware aceptable en varias plataformas, pero que están escritas en un único lenguaje de programación y para el ecosistema de plataformas que soportan la empresa o comunidad involucrada. Ninguna de ellas se ha adoptado como estándar de facto.

Debido a la amplia variedad de plataformas de hardware y lenguajes de programación, se propone en este proyecto la realización del diseño de una biblioteca modelada independientemente del lenguaje de programación y arquitectura del hardware, con la intención de convertirlo en una propuesta de estándar para la programación de Sistemas Embebidos. Se tendrá en cuenta en el modelado lograr una buena relación de compromiso entre el nivel de abstracción para independizarse del hardware y los conceptos que espera encontrar un programador de Sistemas embebidos. Como el lenguaje más utilizado en la actualidad para programación de sistemas embebidos basados en microcontroladores continúa siendo el lenguaje C, se realizará una implementación de referencia para dicho lenguaje.

Dado que el presente autor cumple el rol de Coordinador General del proyecto de hardware y software abierto Computadora Industrial Abierta Argentina (CIAA)"se realizará la implementación de referencia tomando las plataformas de hardware diseñadas en el marco de este proyecto como casos de validación del diseño de biblioteca a realizar. Esto permitirá a la comunidad de usuarios de las plataformas del Proyecto CIAA programar las diferentes plataformas utilizando la misma biblioteca, reduciendo tiempos de aprendizaje y desarrollo.

Este trabajo es parte de un grupo de iniciativas promovidas por la Universidad Nacional de Quilmes para colaborar en el marco del Proyecto CIAA, para facilitar el desarrollo y la enseñanza de Sistemas Embebidos en Argentina. Otras iniciativas incluyen: entorno de programación PLC Ladder (IDE4PLC), programación Java para CIAA (CIAA-HVM), Firmata4CIAA (para facilitar la programación de bloques gráficos en BYOB Snap! lenguaje para uso en escuelas secundarias) y CIAABOT IDE.

## 1.2. Motivación

### 1.2.1. Lenguajes de programación y Hardware en Sistemas Embebidos

Saraza...

### 1.2.2. Bibliotecas para microcontroladores ofrecidas por los fabricantes

Saraza...

### 1.2.3. Proyecto CIAA

De esta forma, es posible programarlos sin necesidad de conocer detalles sobre la arquitectura subyacente. Este diseño promueve y permite la programación independiente del hardware, reduciendo la complejidad general del desarrollo de

sistemas embebidos. Esta biblioteca se utilizará para programar las diferentes plataformas que componen el Proyecto Computadora Industrial Abierta Argentina (CIAA).

Saraza...

En el mercado se encuentra una gran variedad de plataformas basadas en microcontrolador y aunque todas ellas poseen microcontroladores con características similares y periféricos compatibles, sin embargo, sus bibliotecas son muy diferentes. Esto se debe a que cada fabricante y/o empresas asociadas ofrece sus propias bibliotecas en lenguaje C las cuales están diseñadas fuertemente dependientes de cada arquitectura de cada microcontrolador que estas plataformas contienen. Existen también muchas bibliotecas que logran una buena abstracción del hardware en varias plataformas, pero ninguna se ha adoptado como estándar general. Esto se debe a múltiples causas, entre ellas:

En el sector de la industria automotriz existe un estándar llamado AUTOSAR[] para la estandarización de la arquitectura de sistemas electrónicos que aún no ha logrado extenderse a otras industrias y cuya definición de bibliotecas propuesta es muy extensa y compleja de implementar y también para aprender a utilizar. Bibliotecas de drivers basadas en POSIX[] (que podemos hallar en sistemas con en Linux Embebido como Raspberry Pi[]). Su abstracción ha sido muy útil en la estandarización de drivers para PCs con sistema operativo Unix-compatible. Sin embargo, es tan alejada del hardware físico que en la práctica provoca una muy baja utilización por parte de los profesionales electrónicos y afines que se dedican a la programación de sistemas embebidos.

Programadores hobbistas de sistemas embebidos han impulsado la estandarización de la programación mediante una biblioteca conocida como Wiring[] disponible para múltiples plataformas (como la popular Arduino[]). Esta biblioteca si bien logra una gran facilidad de uso y rápido aprendizaje contiene algunas imprecisiones técnicas que provoca vicios indeseados en el aprendizaje de programación de microcontroladores. También carece de definición de una API para la utilización del periférico temporizador el cual es muy utilizado en un microcontrolador.

En la actualidad existen muchos fabricantes de microcontroladores que adquieren licencias para la fabricación de microcontroladores con núcleos de procesamiento de arquitecturas Cortex M[] diseñados por la empresa ARM[]. Para los mismos existe una biblioteca estándar llamada CMSIS para la programación del núcleo de procesamiento y controlador de interrupciones pero que no se extiende a los periféricos donde cada fabricante busca diferenciarse de sus competidores.

Otras empresas muy difundidas en el campo de la enseñanza en la programación como Microchip[] proveen bibliotecas muy dependientes del hardware y generadores automatizados de código que si bien en principio aceleran los tiempos de desarrollo, cuando una aplicación requiere modos más avanzados terminan dificultando la programación pues unas configuraciones pisan a las otras provocando que no funcione.

De los ejemplos anteriores se observa que para la realización de una biblioteca estándar que satisfaga a los diferentes usuarios se debe lograr un balance entre:

- Extensión de la definición de la API.

- Dependencia del hardware.
- Nivel de abstracción.
- Complejidad de aprendizaje y uso.
- Periféricos y modos soportados.
- Escalabilidad.

Por estos motivos se decide realizar la definición de una API de una biblioteca estándar para microcontroladores en lenguaje C que supere todas estas dificultades.

### 1.3. Objetivos y alcance

En esta sección se definen los objetivos (sección 1.3.1) y el alcance (sección 1.3.2) del presente Trabajo Final.

#### 1.3.1. Objetivos

El objetivo de este proyecto es diseñar e implementar una biblioteca de software para la programación de sistemas embebidos basados en microcontroladores con las siguientes características:

- Estar modelada independientemente de los lenguajes de programación.
- Definir una interfaz de programación de aplicaciones (API) sencilla que abstraiga los modos de uso más comunes de los periféricos típicos que hallados en cualquier microcontrolador del mercado.
- Ser totalmente portable entre diferentes arquitecturas de hardware sobre donde se ejecuta, manteniendo una API uniforme a lo largo de las mismas<sup>1</sup>.

Dicha biblioteca se deberá implementar en lenguaje C para las plataformas del Proyecto CIAA.

#### 1.3.2. Alcance

Este Trabajo Final incluye realización de:

- Diseño de la biblioteca. Archivos de descripción de la biblioteca mediante diferentes diagramas y código independiente del lenguaje de programación.
- Implementación en lenguaje C de la biblioteca para las plataformas de hardware:
  - CIAA-NXP.
  - EDU-CIAA-NXP.
  - CIAA-Z3R0.
  - PicoCIAA.

---

<sup>1</sup>Debe cumplir la función de capa de abstracción de hardware, o *Hardware Abstraction Layer* (HAL), en inglés

- Manual de instalación de las herramientas para utilizar la biblioteca con las plataformas de hardware citadas.
- Manual de referencia de la biblioteca.
- Ejemplos de utilización.



## Capítulo 2

# Introducción Específica

### 2.1. Plataformas del Proyecto CIAA

El autor forma parte de una comunidad de desarrolladores de herramientas de software y hardware abierto llamado Proyecto CIAA (siglas en español de Computadora Industrial Abierta Argentina) [3], que desde sus inicios a fines de 2013 intenta impulsar el desarrollo tecnológico nacional proveyendo herramientas abiertas de software y hardware, para mejorar la situación industrial, darle visibilidad a la electrónica y generar cambios estructurales en la forma en que se generan, comparten y utilizan los conocimientos en Argentina. Este proyecto está formado por profesionales de la electrónica y carreras afines.

En sus comienzos el proyecto CIAA desarrolla una computadora industrial basada en un microcontrolador NXP LPC4337 (JDB 144) Dual-core Cortex-M4+Cortex-M0 a 204MHz, con 1 MB de memoria Flash y 136 KB de memoria SRAM nombrada CIAA-NXP [4]. En base a este diseño los integrantes del proyecto CIAA diseñan una versión educativa sin las protecciones industriales, nombrada EDU-CIAA-NXP [5]. En la figura 1 se presenta esta plataforma.

Esta plataforma incluye todos los periféricos típicos que podemos encontrar en los microcontroladores disponibles en el mercado permitiendo la enseñanza con herramientas modernas.

Mediante la colaboración de la Red Universitaria de Sistemas Embebidos (RUSE) [6] se han distribuido en 2015 entre 10 y 40 placas en Universidades de Argentina con carreras afines a la electrónica.

Utilizando la EDU-CIAA-NXP como plataforma base para la enseñanza se ha desarrollado, o colaborado en el desarrollo de diferentes herramientas de software y hardware abiertas, las cuales se describen en las siguientes secciones.

El proyecto de la Computadora Industrial Abierta Argentina (CIAA) nació en 2013 como una iniciativa conjunta entre el sector académico y el industrial, representados por la ACSE<sup>1</sup> y CADIEEL<sup>2</sup>, respectivamente.

Los objetivos del proyecto CIAA son:

<sup>1</sup>Asociación Civil para la investigación, promoción y desarrollo de los Sistemas electrónicos Embebidos. Sitio web: <http://www.sase.com.ar/asociacion-civil-sistemas-embebidos>

<sup>2</sup>Cámara Argentina de Industrias Electrónicas, Electromecánicas y Luminotécnicas. Sitio web: <http://www.cadieel.org.ar/>

- Impulsar el desarrollo tecnológico nacional, a partir de sumar valor agregado al trabajo y a los productos y servicios, mediante el uso de sistemas electrónicos, en el marco de la vinculación de las instituciones educativas y el sistema científico-tecnológico con la industria.
- Darle visibilidad positiva a la electrónica argentina.
- Generar cambios estructurales en la forma en la que se desarrollan y utilizan en nuestro país los conocimientos en el ámbito de la electrónica y de las instituciones y empresas que hacen uso de ella.

Todo esto en el marco de un trabajo libre, colaborativo y articulado entre industria y academia.

Con esta iniciativa, se han desarrollado en la actualidad varias plataformas de hardware y entornos de programación para utilizarlas.

Al momento de la presentación de este trabajo, existen dos versiones de la plataforma CIAA cuyo desarrollo ha sido completado:

- CIAA-NXP, basada en el microcontrolador NXP LPC4337, que ya se comercializa.
- CIAA-FSL, que utiliza, en cambio, el microcontrolador Freescale MK60FX512VLQ15, pero únicamente hay prototipos de esta plataforma.

Además, existe una versión educativa de bajo costo de la CIAA-NXP, nombrada EDU-CIAA-NXP, que ya se distribuyeron alrededor de 1000 unidades y ya hay otras 1000 reservadas en producción.

Debido a estas razones, el trabajo se enfoca en el desarrollo de herramientas para programar las dos plataformas basadas en el microcontrolador NXP LPC4337. Se introducen a continuación las características de las mismas.

Siendo el autor participante de este proyecto desde mediados de 2014, ocupando el rol de Responsable de Software-PLC mediante el aporte al proyecto CIAA de un IDE<sup>3</sup> que permite programar esta plataforma con lenguajes de PLC industriales (IEC-661131-3), se desea agregar en esta oportunidad la posibilidad de programar a esta plataforma con un lenguaje de programación orientado a objetos mediante el desarrollo de un IDE para tal fin.

### 2.1.1. CIAA-NXP

La CIAA-NXP es la primera y única computadora del mundo que reúne dos cualidades:

- Ser **Industrial**, ya que su diseño está preparado para las exigencias de confiabilidad, temperatura, vibraciones, ruido electromagnético, tensiones, cortocircuitos, etc., que demandan los productos y procesos industriales.
- Ser **Abierta**, ya que toda la información sobre su diseño de hardware, firmware, software, etc. está libremente disponible en Internet bajo la Licencia BSD, para que cualquiera la utilice como quiera.

---

<sup>3</sup>IDE4PLC. Sitio web: <http://proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:software-plc>

Esta plataforma se compone de:

- CPU: Microcontrolador NXP LPC 4337 JDB 144 (Dual-core Cortex-M4 + Cortex-M0 @ 204MHz).
- Debugger: USB-to-JTAG FT2232H. Soportado por OpenOCD.
- Memorias:
  - IS42S16400F - SDRAM. 64Mbit @ 143MHz.
  - S25FL032P0XMFI011 - Flash SPI. 32 Mbit, Quad I/O Fast read: 80 MHz.
  - 24AA1025 - EEPROM I2C. 1 Mbit, 400 kHz. Almacenamiento de propósito general, datos de calibración del usuario, etc.
  - 24AA025E48 - EEPROM I2C. 2 kbit, 400 kHz. Para implementación de MAC-Address o almacenamiento de propósito general.
- Entradas y salidas:
  - 8 entradas digitales opto-aisladas 24VDC.
  - 4 Entradas analógicas 0-10V/4-20mA.
  - 4 salidas Open-Drain 24VDC.
  - 4 Salidas con Relay DPDT.
  - 1 Salida analógica 0-10V/4-20mA.
- LV-GPIO:
  - 14 GPIOs.
  - I2C.
  - SPI.
  - 4 canales analógicos.
  - Aux. USB.
- Interfaces de comunicación:
  - Ethernet.
  - USB On-The-Go.
  - RS232.
  - RS485.
  - CAN.
- Múltiples fuentes de alimentación.

En la figura [2.1] se muestra una fotografía de la plataforma.

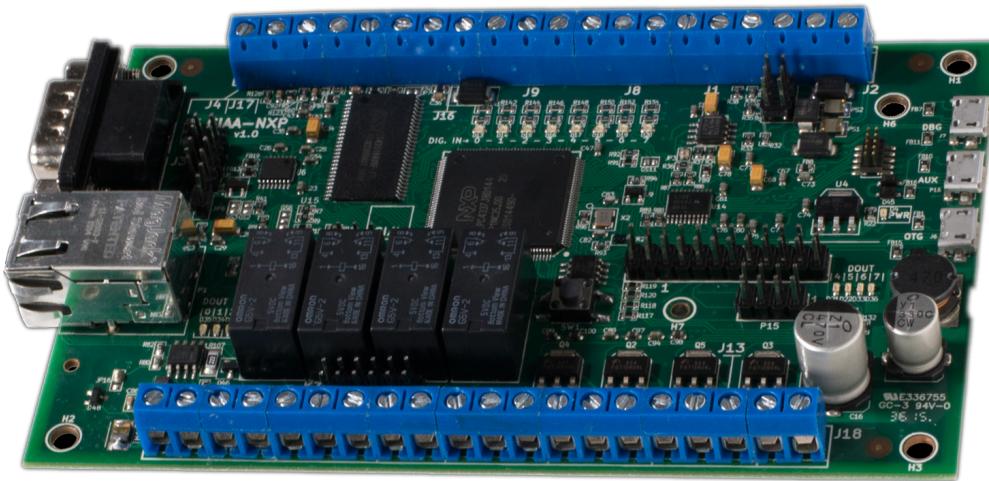


FIGURA 2.1: Plataforma CIAA-NXP.

### 2.1.2. EDU-CIAA-NXP

La plataforma EDU-CIAA-NXP es un desarrollo colaborativo, realizado por miembros de la Red Universitaria de Sistemas Embebidos (RUSE), en el marco del Proyecto CIAA. RUSE se compone de docentes pertenecientes a más de 60 Universidades a lo largo y a lo ancho del país.

Los propósitos de la plataforma son:

- Proveer una plataforma de desarrollo moderna, económica y de fabricación nacional basada en la CIAA-NXP, que sirva a docentes y a estudiantes en los cursos de sistemas embebidos.
- Lograr una amplia inserción en el sistema educativo argentino.
- Realizar un aporte eficaz al desarrollo de vocaciones tempranas en electrónica, computación e informática.
- Demostrar que las universidades argentinas son capaces de realizar un desarrollo colaborativo exitoso en el área de los sistemas embebidos, cumpliendo con requerimientos de tiempo y forma.

Características de la EDU-CIAA-NXP:

- CPU: Microcontrolador NXP LPC 4337 JDB 144 (Dual-core Cortex-M4 + Cortex-M0 @ 204MHz).
- Debugger: USB-to-JTAG FT2232H. Soportado por OpenOCD.
- 2 puertos micro-USB (uno para aplicaciones y debug, otro OTG).
- 6 salidas digitales implementadas con leds (3 normales y uno RGB).
- 4 entradas digitales con pulsadores.
- 1 puerto de comunicaciones RS-485 con bornera.
- 2 conectores de expansión:
  - P0:

- 3 entradas analógicas (ADC0 a ADC2).
- 1 salida analógica (DAC0).
- 1 conexión para un teclado de 3 x 4.
- 12 pines genéricos de I/O.
- P1:
  - 1 puerto Ethernet.
  - 1 puerto CAN.
  - 1 puerto SPI.
  - 1 puerto I2C.
  - 12 pines genéricos de I/O.

En la figura [2.2] se muestra una fotografía de esta plataforma.

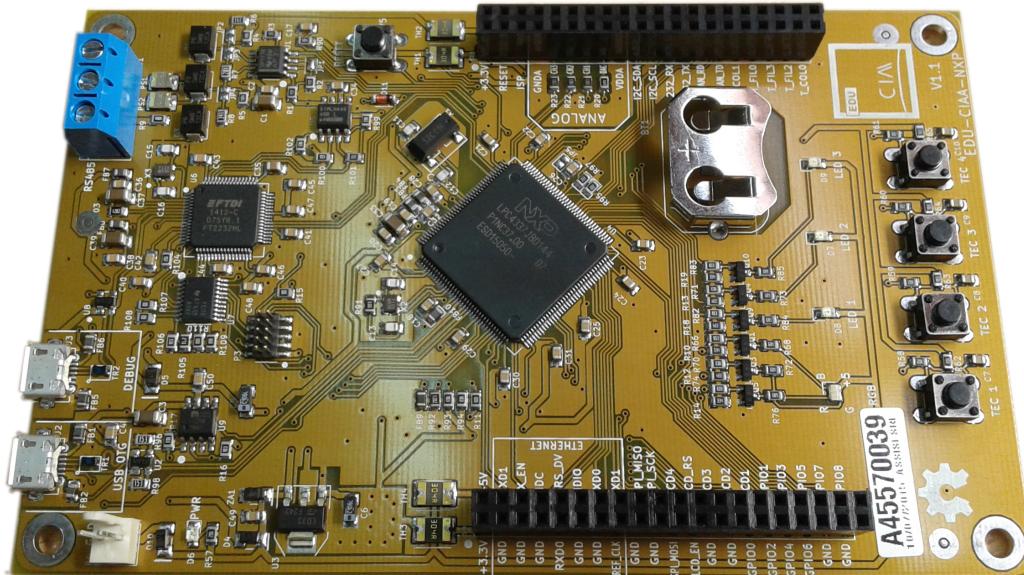


FIGURA 2.2: Plataforma EDU-CIAA-NXP.

### 2.1.3. Pico-CIAA

<http://www.proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:hardware:picociaa>

Procesador Dual-Core de 32 bits

Tiene un procesador Dual Core de 32-bit ARM Cortex-M4F / Cortex-M0+ @100 MHz con 512kB flash y 104 kB SRAM, incluye debug vía un LPC11U35 programable.

Diagrama en bloques

Posee diversos puertos de comunicación, entre ellos USB, PCIExpress, UART, SPI, I2C y soporte para PWM y entradas y salidas digitales de propósito general. special

Es la CIAA más pequeña

Con sólo 51 x 30 mm la picoCIAA es la placa más pequeña de la familia CIAA, lo que la hace ideal para aplicaciones de internet de las cosas (IoT).

Ideal en Single Board Computers

Su interfaz PCI Express permite integrarla fácilmente en SBC de propósito general, para dar soporte de bajo nivel al manejo de dispositivos en tiempo real.



FIGURA 2.3: Plataforma PicoCIAA.

#### 2.1.4. CIAA-Z3R0

Esta plataforma es ideal para aplicaciones de bajo consumo y proyectos de robótica educativa. Está diseñada para ser integrada como componente en un diseño mayor debido a su reducido tamaño conectándola mediante tiras de pines, o soldada a través de su borde de agujeros para montaje castellated.

Se puede comprar en Argentina por aproximadamente AR \$400 a inicios de 2018.

Posee la mayoría de los periféricos que se encuentran en la EDU-CIAA-NXP pero utiliza un microcontrolador siete veces más económico que esta última, de la empresa Silicon Labs, modelo EFM32HG322F64 (QFP48) con núcleo ARM Cortex-M0+ a 25MHz, 64KB de memoria Flash y 8KB de memoria SRAM, que es suficiente para que un alumno entre en el mundo de los microcontroladores modernos de 32 bits.

El dispositivo de depuración se debe comprar por separado, sin embargo, mediante un único dispositivo de depuración se pueden programar muchas plataformas CIAAZ3R0. No es necesario dejar este circuito de depuración en el diseño final

#### 2.1.5. Arquitectura de Hardware

Saraza...

#### 2.1.6. Bibliotecas disponibles

Saraza...

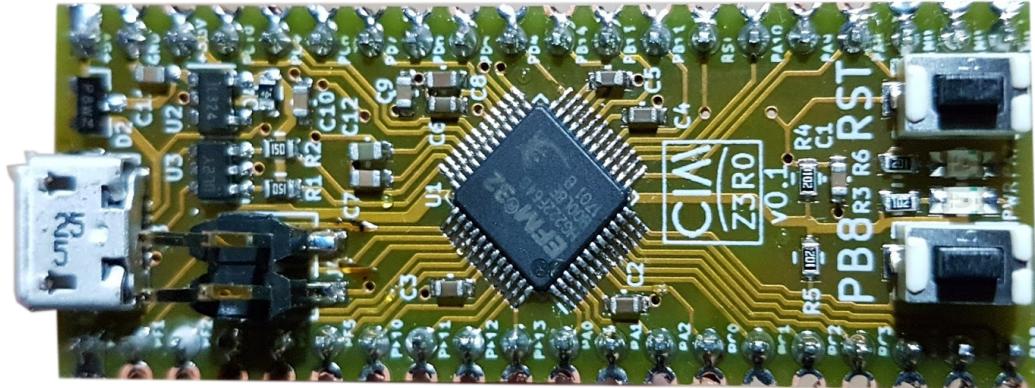


FIGURA 2.4: Plataforma CIAA-Z3R0.

## 2.2. Requerimientos

Los requerimientos se establecieron en base a los objetivos expuestos en la sección 1.3, reuniones con desarrolladores del Proyecto CIAA y el director del presente trabajo. Además se tuvieron en cuenta las opiniones de alumnos de grado de UNQ, posgrado de FIUBA y cursos CAPSE. Estos son:

1. Fecha de finalización: 19/11/2018.
2. Diseño de la biblioteca.
  - a) Realizar un diseño independiente del hardware y lenguaje de programación, teniendo en cuenta los conceptos familiares al programador de Sistemas Embebidos.
  - b) Debe estar modelada con objetos y contar con una descripción mediante diagramas UML.
  - c) Debe modelar al menos:
    - 1) CORE: un núcleo de procesamiento.
    - 2) GPIO: periférico que consiste en un único pin de entrada/salida de propósito general (*pin*), así como un grupo de pines (*port*).
    - 3) ADC: periférico conversor analógico-digital.
    - 4) DAC: periférico conversor digital-analógico.
    - 5) TIMER: periférico temporizador.
    - 6) RTC: periférico reloj de tiempo real.
    - 7) UART: periférico de comunicación serial asincrónico.
    - 8) SPI: periférico interfaz serie sincrónica.
    - 9) I2C: periférico de comunicación serie entre circuitos integrados.
3. Implementación de la biblioteca.
  - a) Utilizar un sistema de control de versiones con repositorios on line.
  - b) Programar en lenguaje C la biblioteca para cada plataforma de hardware particular utilizando como plantilla los archivos generados.

- c) Desarrollar ejemplos de utilización para las diferentes plataformas.
- 4. Documentación y difusión.
  - a) Confeccionar un manual de referencia de la API de la biblioteca.
  - b) Desarrollar un tutorial de instalación de las herramientas para utilizar la biblioteca con las plataformas de hardware citadas.
  - c) Publicación on line del código fuente.
  - d) Informar a la comunidad del Proyecto CIAA y a la comunidad de programadores de Sistemas Embebidos.

### **2.3. Planificación**

En la tabla 2.1 se resume la planificación del trabajo. En la misma se pueden observar cada una de las tareas planificadas, junto a su duración, fecha de inicio y fecha de finalización estimadas.

EDT	Duración	Inicio	Fin
1	48horas	03/05/2018	17/05/2018
1.1	24horas	03/05/2018	10/05/2018
1.2	24horas	10/05/2018	17/05/2018
2	35.5días	17/05/2018	16/08/2018
2.1	22días	17/05/2018	12/07/2018
2.1.1	16horas	17/05/2018	22/05/2018
2.1.2	24horas	22/05/2018	29/05/2018
2.1.3	22horas	31/05/2018	05/06/2018
2.1.4	40horas	07/06/2018	19/06/2018
2.1.5	6horas	19/06/2018	21/06/2018
2.1.6	16horas	21/06/2018	26/06/2018
2.1.7	24horas	26/06/2018	03/07/2018
2.1.8	28horas	03/07/2018	12/07/2018
2.2	28horas	12/07/2018	21/07/2018
2.3	16horas	21/07/2018	26/07/2018
2.4	16horas	26/07/2018	31/07/2018
2.5	12horas	31/07/2018	04/08/2018
2.6	8horas	04/08/2018	07/08/2018
2.7	24horas	07/08/2018	14/08/2018
2.8	4horas	14/08/2018	16/08/2018
3	20.25días	16/08/2018	06/10/2018
3.1	2días	16/08/2018	21/08/2018
3.1.1	4horas	16/08/2018	16/08/2018
3.1.2	4horas	16/08/2018	18/08/2018
3.1.3	4horas	18/08/2018	21/08/2018
3.1.4	4horas	21/08/2018	21/08/2018
3.2	24horas	21/08/2018	28/08/2018
3.3	14.25días	28/08/2018	04/10/2018
3.3.1	40horas	28/08/2018	11/09/2018
3.3.2	36horas	11/09/2018	20/09/2018
3.3.3	38horas	20/09/2018	04/10/2018
3.4	8horas	04/10/2018	06/10/2018
4	3.75días	06/10/2018	16/10/2018
4.1	30horas	06/10/2018	16/10/2018
5	10días	16/10/2018	10/11/2018
5.1	40horas	16/10/2018	30/10/2018
5.2	12horas	30/10/2018	01/11/2018
5.3	16horas	01/11/2018	06/11/2018
5.4	4horas	06/11/2018	08/11/2018
5.5	1hora	08/11/2018	08/11/2018
5.6	3horas	08/11/2018	08/11/2018
5.7	4horas	08/11/2018	10/11/2018

TABLA 2.1: Planificación del Trabajo Final.



## Capítulo 3

# Diseño

### 3.1. Descripción general

### 3.2. Modelado de plataforma de hardware genérica

#### 3.2.1. Clasificación de módulos de hardware

#### 3.2.2. Plataforma y componentes

#### 3.2.3. SoC

#### 3.2.4. Núcleo de procesamiento (Core)

#### 3.2.5. GPIO

Port y Pin

#### 3.2.6. ADC

#### 3.2.7. DAC

#### 3.2.8. TIMER

#### 3.2.9. RTC

#### 3.2.10. UART

#### 3.2.11. SPI

#### 3.2.12. I2C

### 3.3. Verificación del modelo

### 3.4. Modelo abstracto de biblioteca

#### 3.4.1. Módulo de biblioteca

#### 3.4.2. Biblioteca

### 3.5. Diseño de archivo de texto para la descripción de una plataforma de hardware



## Capítulo 4

# Implementación

### 4.1. Descripción general

### 4.2. Archivos de descripción de las paltaformas del proyecto CIAA

#### 4.2.1. CIAA-NXP

#### 4.2.2. EDU-CIAA-NXP

#### 4.2.3. Pico-CIAA

#### 4.2.4. CIAA-Z3R0

### 4.3. Generación automática de código fuente

Generación de código en lenguaje C independiente del hardware.

### 4.4. Implementación del código C dependiente del hardware

### 4.5. Ejemplos de utilización

En esta sección se exponen los ejemplos de uso de la biblioteca realizados y los criterios para realizarlos independientes de las plataforma de hardware.

### 4.6. Documentación y difusión

#### 4.6.1. Generación automática de manual de referencia de la API en base al modelo

#### 4.6.2. Tutoriales de instalación y uso

#### 4.6.3. Difusión a la comunidad del Proyecto CIAA y Embebidos<sup>32</sup>



## Capítulo 5

# Ensayos y Resultados

*Pruebas funcionales del hardware: La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.*

### 5.1. Testeo Unitario

### 5.2. Banco de pruebas de hardware

### 5.3. Integración continua

### 5.4. Utilización de la biblioteca para la enseñanza de programación de Sistemas Embebidos

#### CASOS DE USO (Paper sAPI)

Desde la primer versión de la biblioteca sAPI realizada en el marco del proyecto de Java[] en 2015 se ha utilizado como ejemplo de capa de abstracción de hardware en las asignaturas dictadas por el autor. Estas asignaturas son: el curso de posgrado “Programación de microprocesadores”[] de la Carrera Especialización en Sistemas Embebidos (CESE[]) de la FI-UBA. Donde el autor se ha desempeñado como docente a cargo del curso durante tres ediciones, y la asignatura “Sistemas Digitales” de la carrera Ingeniería en Automatización y Control Industrial (IACI[]) de la Universidad Nacional de Quilmes (UNQ) donde el autor se desempeña en la actualidad como instructor.

A partir de 2016 se decide utilizar la biblioteca como en el marco de los Cursos Abiertos de Programación de Sistemas Embebidos (CAPSE[]) organizados por la ACSE[]. De esta forma se ha extendido la biblioteca de forma considerable para explicar la utilización de todos los periféricos típicos de microcontroladores con excelentes resultados en cuanto a aprendizaje por parte de los alumnos tanto de niveles avanzados como quienes dan sus primeros pasos en el aprendizaje de programación de microcontroladores. Además, la biblioteca sAPI se puso a disposición de cualquier persona ya que se encuentra publicada de forma libre y gratuita por internet bajo una licencia BSD modificada[] en el sitio de github del autor[] y ha sido adoptada por una gran cantidad de usuarios.

Finalmente, en diciembre de 2016 se decide utilizar la biblioteca como biblioteca estándar para las plataformas del Proyecto CIAA. Esto llevó a una profunda revisión y mejora de la misma y todavía se está trabajando en la actualidad.

---

### CASOS DE USO (paper tools)

Estas herramientas se han utilizado en múltiples, entre ellos, el curso "Sistemas digitales" de la carrera Ingeniería en Automatización y Control Industrial en la UNQ, donde el autor actualmente se desempeña como Profesor Instructor desde 2014. El plan de estudio de este curso incluye programación avanzada en lenguaje C para microcontroladores, con temas tales como modularización de código, máquinas de estado finito, sistemas operativos en tiempo real que usan programación cooperativa y apropiativa en microcontroladores; el curso de posgrado "Programación de microprocesadores" dentro de la CESE de FI-UBA donde el autor ha sido profesor a cargo de tres ediciones. Este curso incluye: programación básica de microcontroladores en lenguaje C, modularización, máquinas de estados finitos. También en cursos organizados por ACSE [1]. En estos cursos las herramientas junto con la secuencia didáctica propuesta fueron intensamente utilizadas, mejorando esta secuencia entre el autor y Pablo Gómez. En los cursos de ACSE se distinguen dos grupos, Cursos Abiertos de Programación de Sistemas Embebidos"(CAPSE), orientados a cualquier persona interesada en aprender la programación de Sistemas Embebidos; y cursos impartidos al Instituto Nacional de Escuelas Técnicas"(INET) [2], enfocados en la capacitación de docentes de Escuelas Técnicas Secundarias, para posteriormente poder retransmitir a sus alumnos en todo el país. En ambos casos, se han observado muy buenos resultados de aprendizaje. En el caso de los cursos CAPSE, se impartieron cuatro cohortes entre 2016 y 2017; en la primera cohorte, el 46 % de los estudiantes completaron todos los niveles con un promedio de 67 % de índice de aprobación por nivel; mientras que en la cuarta cohorte, el 73 % completó todos los niveles, con una tasa de aprobación promedio de 81 % por nivel. En el caso de los cursos INET, con dos cohortes en 2017, el 68 % de los estudiantes completaron todos los niveles, con una tasa promedio de aprobación del 74 % por nivel. Además, después de completar sus estudios, algunos de estos estudiantes se inscribieron posteriormente en la CESE FI-UBA para continuar profundizando su aprendizaje. Además, se han realizado varios talleres en escuelas secundarias y universidades de todo el país. Vale la pena mencionar los múltiples tutoriales y workshop en el "Simposio Argentino de Sistemas Embebidos"(SASE) [3]. Todas estas herramientas son de código abierto, publicadas de forma gratuita en la cuenta web de github del Proyecto CIAA y han sido adoptadas por un número importante de usuarios y docentes.

# Capítulo 6

## Conclusiones

### 6.1. Trabajo realizado

*La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.*

En el presente Trabajo Final se ha logrado obtener un entorno de desarrollo para aplicaciones Java SCJ sobre las plataformas CIAA-NXP y EDU-CIAA-NXP, que además de ser software libre, cubre las necesidades planteadas, tanto al ofrecer programación orientada a objetos, así como funcionalidades de tiempo real para entornos industriales, sobre sistemas embebidos.

Como subproducto, se obtiene además una biblioteca con una API sencilla para el manejo de periféricos de las plataformas CIAA-NXP y EDU-CIAA-NXP, que puede utilizarse en aplicaciones Java, o directamente en lenguaje C, debido a su diseño como módulo de Firmware de la CIAA. Se ha tenido especial cuidado en el diseño de esta biblioteca para que la misma sea lo más genérica posible logrando que además se comporte como una HAL.

El desarrollo de este Trabajo Final demandó la articulación de conocimientos adquiridos a lo largo de la Carrera de Especialización en Sistemas Embebidos, en especial las asignaturas:

- **Arquitectura de microprocesadores.** De esta asignatura se emplean los conocimientos adquiridos sobre la arquitectura ARM Cortex M necesarios para implementar en lenguaje *assembler* las funciones que realizan el cambio de contexto, necesarias para que funcione el concepto de Proceso SCJ.
- **Programación de microprocesadores.** De esta asignatura se aprovecha la experiencia sobre lenguaje C para microcontroladores de 32 bits y el manejo de sus periféricos. Fue de especial importancia, debido a que; a excepción de unas pocas, todas las funciones para portar HVM a la CIAA debían realizarse en lenguaje C. Este lenguaje también se utilizó en la creación de la API para el manejo de periféricos.
- **Ingeniería de software en sistemas embebidos.** Se aplican las metodologías de trabajo, provenientes de la ingeniería de software, que aportan calidad y eficiencia al desarrollo. En particular, el diseño iterativo, manejo de repositorios de software y diseño modular en capas.

- **Gestión de Proyectos en Ingeniería.** Durante ésta se desarrolló el Plan de Proyecto del Trabajo Final, permitiendo desde un principio tener una clara planificación del trabajo a realizar.
- **Sistemas Operativos de Propósito General.** Se usan los conocimientos adquiridos sobre Linux para probar las herramientas desarrolladas sobre este sistema operativo.
- **Sistemas Operativos de Tiempo Real (I y II).** De estas asignaturas se aplica el conocimiento obtenido sobre planificadores de tareas expropiativos y la manera en que trabajan. Esto ha sido muy importante para la realización de este Trabajo Final. También la creación de módulos de Firmware para la CIAA.
- **Desarrollo de Sistemas Embebidos en Android.** La plataforma Android es un claro caso de éxito de programación en Java de sistemas embebidos (aunque no sea para aplicaciones industriales). Si bien se contaba con experiencia en programación orientada a objetos en otros lenguajes, esta asignatura fue para el autor el primer acercamiento a dicho lenguaje. En consecuencia, mucho de lo aprendido colaboró en la decisión de llevar a cabo este trabajo.
- **Diseño de Sistemas Críticos.** Los conceptos de esta materia contribuyeron a comprender, inmediatamente, las importantes implicancias de poder programar aplicaciones SCJ en sistemas embebidos para aplicaciones industriales.

También, se han adquirido aprendizajes en las temáticas:

- Programación de aplicaciones en lenguaje Java.
- Especificaciones de Java, entre ellas, RTSJ y SCJ.
- Programación de aplicaciones SCJ.
- Experiencia en implementación de cambio de contexto, procesos y planificadores.
- Máquinas Virtuales de Java para sistemas embebidos y su funcionamiento interno.
- Desarrollo de una biblioteca Java para manejo de periféricos en sistemas embebidos. Conexión con bibliotecas nativas en lenguaje C.

Por lo tanto, se llega a la conclusión que los objetivos planteados al comenzar el trabajo han sido alcanzados satisfactoriamente, y además, se obtienen conocimientos muy importantes para la formación profesional del autor.

---

#### CONCLUSIONES Y TRABAJOS FUTUROS (paper sAPI)

En base a los resultados obtenidos al utilizar la biblioteca en el dictado de diferentes cursos, asignaturas de diferentes niveles en la enseñanza, y la aceptación de múltiples programadores para sus proyectos personales, se cree que se ha logrado diseñar una biblioteca que permite la programación en lenguaje C de forma sencilla en plataformas embebidas basadas en microcontrolador sin la necesidad de conocer en detalle la arquitectura particular de cada uno de ellos. En particular, para

programadores experimentados la biblioteca sAPI logra agilizar el desarrollo de aplicaciones mediante su directa utilización, extender o reconfigurar la misma de acuerdo a sus necesidades; o bien, tomarla como base para entender la programación de una plataforma particular revisando su código fuente. En el campo de la enseñanza de la programación de sistemas embebidos, permite concentrar los esfuerzos de los estudiantes novatos en entender la programación aplicaciones sobre sistemas embebidos en general, en lugar de una aprender la programación de una única plataforma particular, logrando que se entiendan los conceptos más importantes independientemente del hardware subyacente. Luego, con el avance de la formación un alumno puede entender como está realizada la biblioteca para una plataforma en particular y tomarla como un ejemplo de capa de abstracción de hardware. De esta manera se hace hincapié en todo momento en el hecho fundamental de diseñar aplicaciones independientes del hardware debido a la velocidad de cambio de estos dispositivos en el mercado.

Aún se continúa portando la biblioteca a otros micro-controladores y mejorando su definición. Se prevé concluir para finales del presente año 2017 la implementación completa de biblioteca para las plataformas:

CIAA-NXP. Pico-CIAA CIAA-PIC.

Se desea además trabajar en el futuro cercano en el desarrollo de un simulador capaz de ejecutar un programa en C realizado con la biblioteca sAPI sobre una placa virtual en la PC para facilitar la enseñanza de la programación sin disponer de la plataforma física.

---

#### CONCLUSIONES Y TRABAJO A FUTURO (paper tools)

En base a los resultados obtenidos al utilizar la secuencia didáctica con las herramientas propuestas para enseñar programación de Sistemas Embebidos a lo largo de los cursos se cree que el enfoque utilizado es correcto y que el desarrollo y selección de las herramientas ha colaborado enormemente para lograr el objetivo que estudiantes sin experiencia previa, aprendan a programar Sistemas Embebidos desde lo más sencillo hasta utilizar las herramientas profesionales.

El diseño de los cursos ha permitido que los estudiantes concentren sus esfuerzos en entender cómo se realizan aplicaciones con Sistemas Embebidos, en lugar de aprender la programación de una única plataforma particular, lo que hace posible comprender los conceptos importantes independientemente del hardware subyacente. Una vez adquiridas las nociones propuestas, el estudiante puede seguir profundizando sus conocimientos. De esta forma, se enfatiza en todo momento el hecho fundamental de diseñar aplicaciones independientes del hardware, un beneficio real dada la velocidad de cambio de los dispositivos en el mercado actual.

Aún existe mucho trabajo por delante para continuar mejorando las herramientas, como el desarrollo de un simulador que permita virtualizar las plataformas de hardware, para tener la capacidad de ejecutar un programa escrito en C utilizando la biblioteca sAPI, dentro de la PC, para facilitar la enseñanza de la programación sin necesidad de una plataforma física del microcontrolador.

---

## 6.2. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.

Trabajo a futuro

Como labor a futuro, pueden realizarse las siguientes tareas:

---

### AGRADECIMIENTOS (paper tools)

A Mg. Ing. Félix Safar, director del Programa de Investigación en el que participo en UNQ, y el departamento de ciencia y tecnología, dirigido por Dr. Alejandra Zinni por el apoyo al Programa de investigación.

A Dr. Ing. Ariel Lutemberg, Dr. Ing. Pablo Gómez, Ing. José Juarez y otros docentes que confiaron en el criterio del autor, colaborando activamente en el uso de las herramientas y metodología propuestas.

A Martín Ribleotta que ha sido un invaluable colaborador tanto en diseño como en la implementación de las herramientas.

A Leandro Lanzieri Rodriguez por su colaboración en el Proyecto CIAA con el desarrollo de CIAABOT IDE.

A la comunidad del proyecto CIAA que ha adoptado estas herramientas con gran entusiasmo, motivando al autor a continuar con la mejora continua de las mismas.

---

AGRADECIMIENTOS (paper sAPI) A Martín Ribeletta cuya experiencia ha sido y continúa siendo un aporte fundamental durante la revisión actual de la biblioteca.

Al Dr. Ing. Ariel Lutemberg y al Dr. Ing. Pablo Gómez quienes confiaron en utilizar la misma para los CAPSE[].

A los coordinadores del Proyecto CIAA[] para quienes han decidido utilizarla como biblioteca estándar para sus plataformas, en especial, al coordinador general Esp. Ing. Pablo Ridolfi.

A los alumnos de la CESE (FI-UBA)[], IACI (UNQ)[] y los CAPSE para los cuales se fue desarrollando la biblioteca y recibieron muy entusiasmados impulsando el desarrollo.

Finalmente, al Ing. Leonardo Gassman que ideó el nombre de la biblioteca durante el proyecto de Java[].