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.*

# *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 contínua 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

# Índice de figuras

# Índice de Tablas

***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 Çomputadora 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 estandariza- ción 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 perifŕicos 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[1].

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.
- 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.

[1] Debe cumplir la función de capa de abstracción de hardware, o *Hardware Abstraction Layer* (HAL), en inglés

# Capítulo 2

# Introducción Específica

## 2.1. Plataformas del Proyecto CIAA

Saraza...

### 2.1.1. CIAA-NXP

Saraza...

### 2.1.2. EDU-CIAA-NXP

Saraza...

### 2.1.3. Pico-CIAA

Saraza...

### 2.1.4. CIAA-Z3R0

Saraza...

### 2.1.5. Arquitectura de Hardware

Saraza...

### 2.1.6. Bibliotecas disponibles

Saraza...

## 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 | Nombre | Duración | Inicio | Fin |
|---|---|---|---|---|
| 1 | **⊟Investigación preliminar.** | 48horas | 03/05/2018 | 17/05/2018 |
| 1.1 | Investigar las bibliotecas para microcontroladores disponibles. | 24horas | 03/05/2018 | 10/05/2018 |
| 1.2 | Investigar las bibliotecas de C y documentación de las plataformas de hardware. | 24horas | 10/05/2018 | 17/05/2018 |
| 2 | **⊟Diseño de la biblioteca independiente del hardware y lenguaje de programación.** | 35.5días | 17/05/2018 | 16/08/2018 |
| 2.1 | **⊟Modelar las propiedades y métodos de los periféricos:** | 22días | 17/05/2018 | 12/07/2018 |
| 2.1.1 | GPIO. | 16horas | 17/05/2018 | 22/05/2018 |
| 2.1.2 | ADC. | 24horas | 22/05/2018 | 29/05/2018 |
| 2.1.3 | DAC. | 22horas | 31/05/2018 | 05/06/2018 |
| 2.1.4 | TIMER. | 40horas | 07/06/2018 | 19/06/2018 |
| 2.1.5 | RTC. | 6horas | 19/06/2018 | 21/06/2018 |
| 2.1.6 | UART. | 16horas | 21/06/2018 | 26/06/2018 |
| 2.1.7 | SPI. | 24horas | 26/06/2018 | 03/07/2018 |
| 2.1.8 | I2C. | 28horas | 03/07/2018 | 12/07/2018 |
| 2.2 | Modelar las propiedades y métodos del núcleo de procesamiento (CORE). | 28horas | 12/07/2018 | 21/07/2018 |
| 2.3 | Modelar SoC. | 16horas | 21/07/2018 | 26/07/2018 |
| 2.4 | Modelar Board. | 16horas | 26/07/2018 | 31/07/2018 |
| 2.5 | Realización de diagramas UML. | 12horas | 31/07/2018 | 04/08/2018 |
| 2.6 | Verificación del modelo. | 8horas | 04/08/2018 | 07/08/2018 |
| 2.7 | Diseñar un archivo de descripción de la biblioteca. | 24horas | 07/08/2018 | 14/08/2018 |
| 2.8 | Verificación del archivo de descripción. | 4horas | 14/08/2018 | 16/08/2018 |
| 3 | **⊟Implementación de la biblioteca.** | 20.25días | 16/08/2018 | 06/10/2018 |
| 3.1 | **⊟Implementar el archivo de descripción para las plataformas de hardware:** | 2días | 16/08/2018 | 21/08/2018 |
| 3.1.1 | CIAA-NXP | 4horas | 16/08/2018 | 16/08/2018 |
| 3.1.2 | EDU-CIAA-NXP | 4horas | 16/08/2018 | 18/08/2018 |
| 3.1.3 | CIAA-Z3R0 | 4horas | 18/08/2018 | 21/08/2018 |
| 3.1.4 | PicoCIAA | 4horas | 21/08/2018 | 21/08/2018 |
| 3.2 | Realizar un generador de archivos ".c" y ".h" en base a los archivos de descripción. | 24horas | 21/08/2018 | 28/08/2018 |
| 3.3 | **⊟Programar en lenguaje C la biblioteca para cada plataforma de hardware:** | 14.25días | 28/08/2018 | 04/10/2018 |
| 3.3.1 | CIAA-NXP y EDU-CIAA-NXP. | 40horas | 28/08/2018 | 11/09/2018 |
| 3.3.2 | CIAA-Z3R0. | 36horas | 11/09/2018 | 20/09/2018 |
| 3.3.3 | PicoCIAA. | 38horas | 20/09/2018 | 04/10/2018 |
| 3.4 | Verificación del funcionamiento de cada periférico en cada una de las plataformas. | 8horas | 04/10/2018 | 06/10/2018 |
| 4 | **⊟Validación.** | 3.75días | 06/10/2018 | 16/10/2018 |
| 4.1 | Realización de ejemplos funcionales para cada periférco. | 30horas | 06/10/2018 | 16/10/2018 |
| 5 | **⊟Procesos finales.** | 10días | 16/10/2018 | 10/11/2018 |
| 5.1 | Redacción de memoria de trabajo final. | 40horas | 16/10/2018 | 30/10/2018 |
| 5.2 | Confeccionar un manual de referencia de la API de la biblioteca. | 12horas | 30/10/2018 | 01/11/2018 |
| 5.3 | Desarrollar un tutorial de instalación de las herramientas para utilizar la biblioteca. | 16horas | 01/11/2018 | 06/11/2018 |
| 5.4 | Release on line del código fuente con documentación. | 4horas | 06/11/2018 | 08/11/2018 |
| 5.5 | Informar a la comunidad del Proyecto CIAA y programadores de S.E. | 1hora | 08/11/2018 | 08/11/2018 |
| 5.6 | Evaluar el cumplimiento de requerimientos. | 3horas | 08/11/2018 | 08/11/2018 |
| 5.7 | Preparación de la presentación del proyecto. | 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 dependiende del hardware

## 4.5. Ejemplos de utilización

En esta sección se exponen los ejmplos 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 Embebidos32

# 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 contínua

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

# 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.*

## 6.2. Próximos pasos

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