UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

Título del trabajo

Autor:
Nombre del Autor

Director:
Nombre del Director (Filiación)

Jurados:
Nombre del jurado 1 (Filiación)
Nombre del jurado 2 (Filiación)
Nombre del jurado 3 (Filiación)

Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires, entre enero de 2016 y diciembre de 2016.

Resumen

Acá va el resumen del trabajo. Debe ser lo más breve posible. No más de dos o tres párrafos, de unas cuatro o cinco oraciones cada uno. Leyendo esto debe quedar muy claro en qué consiste el trabajo realizado, por qué el trabajo es importante, por qué el trabajo muestra que el estudiante aplicó correctamente lo aprendido en la Carrera y qué información va a encontrar el lector en esta Memoria.

No usar en este resumen ninguna referencia bibliográfica del tipo [1], ni tampoco notas a pie de página ni siglas que no estén aclaradas como parte de este texto, ni tipografía en negritas, subrayada o cursiva. Dicho de otra forma, el texto en este resumen debe ser escrito de forma tal que si se recorta el mismo y se lo pega en un archivo .txt entonces este conserve su formato y sea perfectamente entendible sin ningún agregado adicional, es decir, quede autocontenido.

Agradecimientos

Agradecimientos personales. **[OPCIONAL]**

No 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. Aprendiendo LaTeX

LaTeX no es WYSIWYG (What You See is What You Get), a diferencia de los procesadores de texto como Microsoft Word o Pages de Apple o incluso LibreOffice en el mundo open-source. En lugar de ello, un documento escrito para LaTeX es en realidad un archivo de texto simple, llano que *no contiene formato* . Nosotros le decimos a LaTeX cómo deseamos que se aplique el formato en el documento final escribiendo comandos simples entre el texto, por ejemplo, si quiero usar *texto en cursiva para dar énfasis*, escribo `\emph{texto}` y pongo el texto en cursiva que quiero entre medio de las llaves. Esto significa que LaTeX es un lenguaje del tipo «mark-up», muy parecido a HTML.

1.1.1. Una introducción (no tan corta) a LaTeX

Si usted es nuevo a LaTeX, hay un muy buen libro electrónico - disponible gratuitamente en Internet como un archivo PDF - llamado, «A (not so short) Introduction to LaTeX». El título del libro es generalmente acortado a simplemente *lshort*. Puede descargar la versión más reciente (ya que se actualiza de vez en cuando) desde aquí: http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

Está disponible en varios otros idiomas. Se puede encontrar la versión en español en la lista en esta página: http://www.ctan.org/tex-archive/info/lshort/

1.1.2. Guía matemática rápida para LaTeX

Si usted está escribiendo un documento con mucho contenido matemático, entonces es posible que desee leer el documento de la AMS (American Mathematical Society) llamado, «A Short Math Guide for LaTeX». Se puede encontrar en línea aquí: http://www.ams.org/tex/amslatex.html en la sección «Additional Documentation» hacia la parte inferior de la página.

1.2. Utilizando esta plantilla

Si usted está familiarizado con LaTeX, entonces puede explorar la estructura de directorios de esta plantilla y proceder a personalizarla agregando su información en el bloque *INFORMACIÓN DE LA PORTADA* en el archivo **`memoria.tex`**.

Se puede continuar luego modificando el resto de los archivos siguiendo los lineamientos que se describen en la sección 1.5 en la página 5.

Asegúrese de leer el capítulo 2 acerca de las convenciones utilizadas para las Memoria de los Trabajos Finales de la Carrera de Especialización en Sistemas Embebidos de FIUBA.

Si es nuevo en LaTeX se recomienda que continue leyendo el documento ya que contiene información básica para aprovechar el potencial de esta herramienta.

1.2.1. Acerca de esta plantilla

Esta plantilla LaTeX está basada originalmente en torno a un archivo de estilo LaTeX creado por Steve R. Gunn de la University of Southampton (UK), department of Electronics and Computer Science. Se puede encontrar su trabajo original en el siguiente sitio de internet: http://www.ecs.soton.ac.uk/~srg/softwaretools/document/templates/

El archivo de Gunn, **`ecsthesis.cls`** fue posteriormente modificado por Sunil Patel quien creó una plantilla esqueleto con la estructura de carpetas. El template resultante se puede encontrar en el sitio web de Sunil Patel: http://www.sunilpatel.co.uk/thesis-template

El template de Patel se publicó a través de http://www.LaTeXTemplates.com desde donde fue modificado muchas veces en base a solicitudes de usuarios. La versión 2.0 y subsiguientes representan cambios significativos respecto a la versión de la plantilla modificada por Patel, que es de hecho, dificilmente reconocible. El trabajo en la version 2.0 fue realizado por Vel Gayevskiy y Johannes Böttcher.

Uno de los primeros graduados de la Carerra de Especialización en Sistemas Embebios de la UBA, el Ing. Patricio Bos modificó los contenidos de la versión 2.3 para crear una plantilla altamente adaptada a la Carrera de Especialización de la UBA.

1.3. Qué incluye esta plantilla

1.3.1. Carpetas

Esta plantilla se distribuye como una único archivo .zip que se puede descomprimir en varios archivos y carpetas. Los nombres de las carpetas son (o pretender ser) auto-explicativos.

Appendices – Esta es la carpeta donde se deben poner los apéndices. Cada apéndice debe ir en su propio archivo **`.tex`**. Se incluye un ejemplo y una plantilla en la carpeta.

Chapters – Esta es la carpeta donde se deben poner los capítulos de la memoria. Cada capítulo debe ir un su propio archivo **`.tex`** por separado. Se ofrece por defecto, la siguiente estructura de capítulos y se recomienda su utilización dentro de lo posible:

- Capítulo 1: Introducción general
- Capítulo 2: Introducción específica
- Capítulo 3: Diseño e implementación
- Capítulo 4: Ensayos y resultados
- Capítulo 5: Conclusiones

Esta estructura de capítulos es la que se recomienda para las memorias de la especialización.

Figures – Esta carpeta contiene todas las figuras de la memoria. Estas son las versiones finales de las imágenes que van a ser incluidas en la memoria. Pueden ser imágenes en formato *raster* como `.png`, `.jpg` o en formato vectoriales como `.pdf`, `.ps`. Se debe notar que utilizar imágenes vectoriales disminuye notablemente el peso del documento final y acelera el tiempo de compilación por lo que es recomendable su utilización siempre que sea posible.

1.3.2. Archivos

También están incluidos varios archivos, la mayoría de ellos son de texto plano y se puede ver su contenido en un editor de texto. Después de la compilación inicial, se verá que más archivos auxiliares son creados por LaTeX o BibTeX, pero son de uso interno y que no es necesario eliminarlos o hacer nada con ellos. Toda la información necesaria para complilar el documento se encuentra en los archivos `.tex` y en las imágenes de la carpeta Figures.

referencias.bib - este es un archivo importante que contiene toda la información bibliográfica y de referencias que se utilizará para las citas en la memoria en conjunto con BibTeX. Usted puede escribir las entradas bibliográficas en forma manual, aunque existen también programas de gestión de referencias que facilitan la creación y gestión de las referencias y permiten exportarlas en formato BibTeX. También hay disponibles sitios web como books.google.com que permiten obtener toda la información necesaria para una cita en formato BibTeX.

MastersDoctoralThesis.cls – este es un archivo importante. Es el archivos con la clase que le informa a LaTeX cómo debe dar formato a la memoria. El usuario de la plantilla no debería necesitar modificar nada de este archivo.

memoria.pdf – esta es su memoria con una tipografía bellamente compuesta (en formato de archivo PDF) creado por LaTeX. Se distribuye con la plantilla y después de compilar por primera vez sin hacer ningún cambio se debería obtener una versión idéntica a este documento.

memoria.tex – este es un archivo importante. Este es el archivo que tiene que compilar LaTeX para producir la memoria como un archivo PDF. Contiene un marco de trabajo y estructuras que le indican a LaTeX cómo diagramar la memoria. Está altamente comentado para que se pueda entender qué es lo que realiza cada línea de código y por qué está incluida en ese lugar. En este archivo se debe completar la información personalizada de las primeras sección según se indica en la sección 1.5.

Finalmente, en este archivo es donde se incluyen en la complilación los archivos **.tex** con los capítulos indiviluales que se encuentran en la carpeta Chapters. Esto se logra con el siguiente código LaTeX:

```
\include{Chapters/Chapter1}
\include{Chapters/Chaptr2}
\include{Chapters/Chapter3}
\include{Chapters/Chapter4}
\include{Chapters/Chapter5}
```

Si se agregara un 6to capítulo o se sacara alguno se debe agregar una línea equivalente a las precedentes o comentarla, respectivamente.

En forma análoga a los capítulos los apéndices se incluyen con las siguientes líneas de códido:

```
%\include{Appendices/AppendixA}
%\include{Appendices/AppendixB}
%\include{Appendices/AppendixC}
```

Notar que la inclusión de los apéndices está comentada con el símbolo %, lo que significa que no se incluyen por defecto.

Archivos que *no* forman parte de la distribución de la plantilla pero que son generados por LaTeX como archivos auxiliares necesarios para la producción de la memoria.pdf son:

memoria.aux – este es un archivo auxiliar generado por LaTeX, si se borra LaTeX simplemente lo regenera cuando se compila el archivo principal **memoria.tex**.

memoria.bbl – this is an auxiliary file generated by BibTeX, if it is deleted, BibTeX simply regenerates it when you run the **memoria.aux** file. Whereas the **.bib** file contains all the references you have, this **.bbl** file contains the references you have actually cited in the thesis and is used to build the bibliography section of the thesis.

memoria.blg – este es un archivo auxiliar generado por BibTex, si se borra BibTex simplemente lo regenera cuando se compila el archivo principal **memoria.tex**.

memoria.lof – este es un archivo auxiliar generado por LaTeX, si se borra LaTeX simplemente lo regenera cuando se compila el archivo principal **memoria.tex**. Le indica a LaTeX cómo construir la sección *Lista de Figuras*.

memoria.log – este es un archivo auxiliar generado por LaTeX, si se borra LaTeX simplemente lo regenera cuando se compila el archivo principal **memoria.tex**. Contiene mensajes de LaTeX. Si se reciben errores o advertencias durante la compilación, se guardan en este archivo **.log**.

memoria.lot – este es un archivo auxiliar generado por LaTeX, si se borra LaTeX simplemente lo regenera cuando se compila el archivo principal **memoria.tex**. Le indica a LaTeX cómo construir la sección *Lista de Tablas*.

memoria.out – este es un archivo auxiliar generado por LaTeX, si se borra LaTeX simplemente lo regenera cuando se compila el archivo principal **memoria.tex**.

De esta larga lista de archivos, sólo aquellos con la extensión **.bib**, **.cls** y **.tex** son importantes. Los otros archivos auxiliares pueden ser ignorados o borrados ya que LaTeX y BibTeX los regenerarán durante la compilación.

1.4. Entorno de trabajo

Ante de comenzar a editar la plantilla debemos tener un editor LaTeX instalado en nuestra computadora. En forma análoga a lo que sucede en lenguaje C, que se puede crear y editar código con casi cualquier editor, exiten ciertos entornos de trabajo que nos pueden simplificar mucho la tarea. En este sentido, se recomienda, sobre todo para los principiantes en LaTeX la utilización de TexMaker, un programa gratuito y multi-plantaforma que está disponible tanto para windows como para sistemas GNU/linux.

La versión más reciente de TexMaker es la 4.5 y se puede descargar del siguiente link: http://www.xm1math.net/texmaker/download.html. Se puede consultar el manual de usuario en el siguiente link: http://www.xm1math.net/texmaker/doc.html.

1.4.1. Configurando TexMaker

La instalación de TexMaker se encarga de instalar todos los paquetes necesarios de LaTeX

1.5. Incorporando su información personalizada en el archivo `memoria.tex`

Para personalizar la plantilla se debe incorporar la información propia en los distintos archivos **`.tex`**.

Primero abrir **`memoria.tex`** con TexMaker (o el editor de su preferencia). Se debe ubicar dentro del archivo el bloque de código titulado *INFORMACIÓN DE LA PORTADA* donde se deben incorporar los primeros datos personales con los que se constuirá automáticamente la portada.

Notar que existe una vista llamada Estructura como se puede ver en la figura 1.1 que nos permite abrir desde dentro del programa los archivos individuales de los capítulos.

Recordar que el archivo que se debe compilar con PDFLaTex es **`memoria.tex`**, si trataramos de compilar alguno de los capítulos directamente nos saldría un error. Para salvar la molestia de tener que cambiar de archivo para compilar, se puede definer el archivo **`memoria.tex`** como "documento maestro" yendo al menú opciones ->"definir documento actual como documento maestro", lo que nos permite compilar cualquier archivo, sea memoria.tex, el capítulo donde estemos trabajando o incluso un apéndice si lo hubiera y texmaker se encargará automáticamente de compilar memoria.tex.

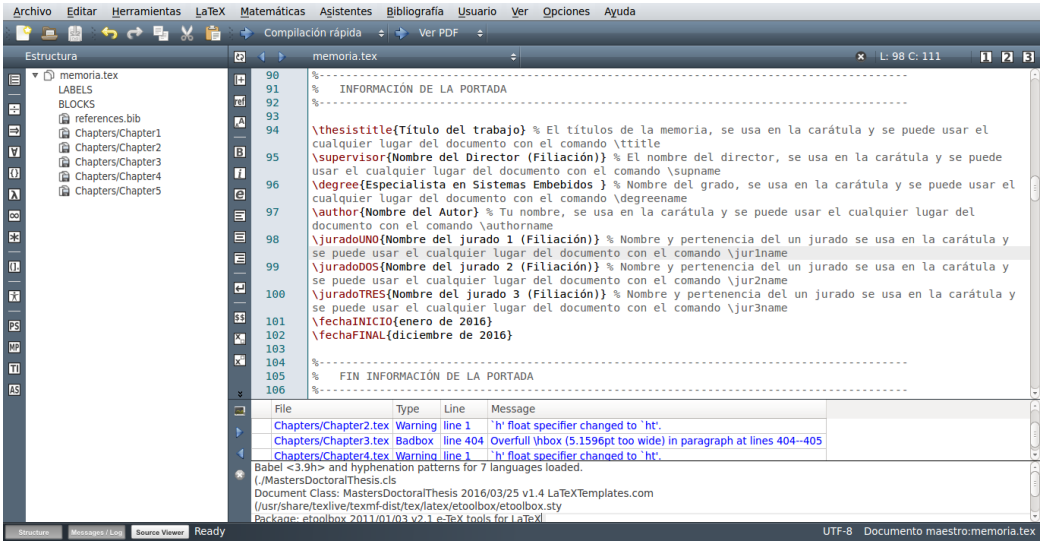

FIGURA 1.1: Entorno de trabajo del texMaker.

1.6. The `memoria.tex` File Explained

The **`memoria.tex`** file contains the structure of the thesis. There are plenty of written comments that explain what pages, sections and formatting the LaTeX code is creating. Each major document element is divided into commented blocks with titles in all capitals to make it obvious what the following bit of code is doing. Initially there seems to be a lot of LaTeX code, but this is all formatting, and it has all been taken care of so you don't have to do it.

Begin by checking that your information on the title page is correct. For the thesis declaration, your institution may insist on something different than the text given. If this is the case, just replace what you see with what is required in the *DECLARATION PAGE* block.

Then comes a page which contains a funny quote. You can put your own, or quote your favourite scientist, author, person, and so on. Make sure to put the name of the person who you took the quote from.

Following this is the abstract page which summarises your work in a condensed way and can almost be used as a standalone document to describe what you have done. The text you write will cause the heading to move up so don't worry about running out of space.

Next come the acknowledgements. On this page, write about all the people who you wish to thank (not forgetting parents, partners and your advisor/supervisor).

The contents pages, list of figures and tables are all taken care of for you and do not need to be manually created or edited. The next set of pages are more likely to be optional and can be deleted since they are for a more technical thesis: insert a list of abbreviations you have used in the thesis, then a list of the physical constants and numbers you refer to and finally, a list of mathematical symbols used in any formulae. Making the effort to fill these tables means the reader has a one-stop place to refer to instead of searching the internet and references to try and find out what you meant by certain abbreviations or symbols.

The list of symbols is split into the Roman and Greek alphabets. Whereas the abbreviations and symbols ought to be listed in alphabetical order (and this is *not* done automatically for you) the list of physical constants should be grouped into similar themes.

The next page contains a one line dedication. Who will you dedicate your thesis to?

Finally, there is the block where the chapters are included. Uncomment the lines (delete the `%` character) as you write the chapters. Each chapter should be written in its own file and put into the *Chapters* folder and named **`Chapter1`**, **`Chapter2`**, etc...Similarly for the appendices, uncomment the lines as you need them. Each appendix should go into its own file and placed in the *Appendices* folder.

After the preamble, chapters and appendices finally comes the bibliography. The bibliography style (called *`authoryear`*) is used for the bibliography and is a fully featured style that will even include links to where the referenced paper can be found online. Do not underestimate how grateful your reader will be to find that a reference to a paper is just a click away. Of course, this relies on you putting the URL information into the BibTeX file in the first place.

1.7. Thesis Features and Conventions

To get the best out of this template, there are a few conventions that you may want to follow.

One of the most important (and most difficult) things to keep track of in such a long document as a thesis is consistency. Using certain conventions and ways of doing things (such as using a Todo list) makes the job easier. Of course, all of these are optional and you can adopt your own method.

1.7.1. Printing Format

This thesis template is designed for double sided printing (i.e. content on the front and back of pages) as most theses are printed and bound this way. Switching to one sided printing is as simple as uncommenting the *`oneside`* option of the `documentclass` command at the top of the **`memoria.tex`** file. You may then wish to adjust the margins to suit specifications from your institution.

The headers for the pages contain the page number on the outer side (so it is easy to flick through to the page you want) and the chapter name on the inner side.

The text is set to 11 point by default with single line spacing, again, you can tune the text size and spacing should you want or need to using the options at the very start of **`memoria.tex`**. The spacing can be changed similarly by replacing the *`singlespacing`* with *`onehalfspacing`* or *`doublespacing`*.

1.7.2. References

The `biblatex` package is used to format the bibliography and inserts references such as this one [**Reference1**]. The options used in the **`memoria.tex`** file

mean that the in-text citations of references are formatted with the author(s) listed with the date of the publication. Multiple references are separated by semicolons (e.g. [**Reference2**, **Reference1**]) and references with more than three authors only show the first author with *et al.* indicating there are more authors (e.g. [**Reference3**]). This is done automatically for you. To see how you use references, have a look at the **`Chapter1.tex`** source file. Many reference managers allow you to simply drag the reference into the document as you type.

Scientific references should come *before* the punctuation mark if there is one (such as a comma or period). The same goes for footnotes[1]. You can change this but the most important thing is to keep the convention consistent throughout the thesis. Footnotes themselves should be full, descriptive sentences (beginning with a capital letter and ending with a full stop). The APA6 states: «Footnote numbers should be superscripted, [...], following any punctuation mark except a dash.» The Chicago manual of style states: «A note number should be placed at the end of a sentence or clause. The number follows any punctuation mark except the dash, which it precedes. It follows a closing parenthesis.»

The bibliography is typeset with references listed in alphabetical order by the first author's last name. This is similar to the APA referencing style. To see how LaTeX typesets the bibliography, have a look at the very end of this document (or just click on the reference number links in in-text citations).

1.7.3. Tables

Tables are an important way of displaying your results, below is an example table which was generated with this code:

```
\begin{table}
\caption{The effects of treatments X and Y on the four groups studied.}
\label{tab:treatments}
\centering
\begin{tabular}{l l l}
\toprule
\tabhead{Groups} & \tabhead{Treatment X} & \tabhead{Treatment Y} \\
\midrule
1 & 0.2 & 0.8\\
2 & 0.17 & 0.7\\
3 & 0.24 & 0.75\\
4 & 0.68 & 0.3\\
\bottomrule\\
\end{tabular}
\end{table}
```

You can reference tables with `\ref{<label>}` where the label is defined within the table environment. See **`Chapter1.tex`** for an example of the label and citation (e.g. Table 1.1).

[1] Such as this footnote, here down at the bottom of the page.

TABLA 1.1: The effects of treatments X and Y on the four groups studied.

Groups	Treatment X	Treatment Y
1	0.2	0.8
2	0.17	0.7
3	0.24	0.75
4	0.68	0.3

1.7.4. Figures

There will hopefully be many figures in your thesis (that should be placed in the *Figures* folder). The way to insert figures into your thesis is to use a code template like this:

```
\begin{figure}
\centering
\includegraphics{Figures/Electron}
\decoRule
\caption[An Electron]{An electron (artist's impression).}
\label{fig:Electron}
\end{figure}
```

Also look in the source file. Putting this code into the source file produces the picture of the electron that you can see in the figure below.

Sometimes figures don't always appear where you write them in the source. The placement depends on how much space there is on the page for the figure. Sometimes there is not enough room to fit a figure directly where it should go (in relation to the text) and so LaTeX puts it at the top of the next page. Positioning figures is the job of LaTeX and so you should only worry about making them look good!

Figures usually should have captions just in case you need to refer to them (such as in Figure **??**). The `\caption` command contains two parts, the first part, inside the square brackets is the title that will appear in the *List of Figures*, and so should be short. The second part in the curly brackets should contain the longer and more descriptive caption text.

The `\decoRule` command is optional and simply puts an aesthetic horizontal line below the image. If you do this for one image, do it for all of them.

LaTeX is capable of using images in pdf, jpg and png format.

1.7.5. Typesetting mathematics

If your thesis is going to contain heavy mathematical content, be sure that LaTeX will make it look beautiful, even though it won't be able to solve the equations for you.

The «Not So Short Introduction to LaTeX» (available on CTAN) should tell you everything you need to know for most cases of typesetting mathematics. If you

need more information, a much more thorough mathematical guide is available from the AMS called, «A Short Math Guide to LaTeX» and can be downloaded from: ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf

There are many different LaTeX symbols to remember, luckily you can find the most common symbols in The Comprehensive LaTeX~Symbol List.

You can write an equation, which is automatically given an equation number by LaTeX like this:

```
\begin{equation}
E = mc^{2}
\label{eqn:Einstein}
\end{equation}
```

This will produce Einstein's famous energy-matter equivalence equation:

$$E = mc^2 \tag{1.1}$$

All equations you write (which are not in the middle of paragraph text) are automatically given equation numbers by LaTeX. If you don't want a particular equation numbered, use the unnumbered form:

```
\[ a^{2}=4 \]
```

1.8. Sectioning and Subsectioning

You should break your thesis up into nice, bite-sized sections and subsections. LaTeX automatically builds a table of Contents by looking at all the `\chapter{}`, `\section{}` and `\subsection{}` commands you write in the source.

The Table of Contents should only list the sections to three (3) levels. A `chapter{}` is level zero (0). A `\section{}` is level one (1) and so a `\subsection{}` is level two (2). In your thesis it is likely that you will even use a `subsubsection{}`, which is level three (3). The depth to which the Table of Contents is formatted is set within **`MastersDoctoralThesis.cls`**. If you need this changed, you can do it in **`memoria.tex`**.

1.9. In Closing

You have reached the end of this mini-guide. You can now rename or overwrite this pdf file and begin writing your own **`Chapter1.tex`** and the rest of your thesis. The easy work of setting up the structure and framework has been taken care of for you. It's now your job to fill it out!

Good luck and have lots of fun!

Guide written by —
Sunil Patel: www.sunilpatel.co.uk
Vel: LaTeXTemplates.com

Capítulo 2

Introducción Específica

2.1. Título de la sección con este uso de las mayúsculas

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.

Si en el texto se hace alusión a diferentes partes del trabajo referirse a ellas como Capítulo, Sección o subsección según corresponda. Por ejemplo: "En el Capítulo 1 se explica tal cosa", o "En la Sección 2.1 se presenta lo que sea", o "En la subsección 2.1.1 se discute otra cosa".

Entre párrafos sucesivos dejar un espacio, como el que se observa entre este párrafo y el anterior. Pero las oraciones de un mismo párrafo van en forma consecutiva, como se observa acá. Luego, cuando se quiere poner una lista tabulada se hace así:

- Este es el primer elemento de la lista.
- Este es el segundo elemento de la lista.

Notar el uso de las mayúsculas y el punto al final de cada elemento.

Si se desea poner una lista numerada el formato es este:

1. Este es el primer elemento de la lista.
2. Este es el segundo elemento de la lista.

Notar el uso de las mayúsculas y el punto al final de cada elemento.

2.1.1. Este es el título de una subsección

Se recomienda no utilizar **texto en negritas** en ningún párrafo, ni tampoco texto subrayado. En cambio sí se sugiere utilizar *texto en cursiva* donde se considere apropiado.

Se sugiere que la escritura sea impersonal. Por ejemplo, no utilizar "el diseño del firmware lo hice de acuerdo con tal principio", sino "el firmware fue diseñado utilizando tal principio". En lo posible hablar en tiempo pasado, ya que la memoria describe un trabajo que ya fue realizado.

Se recomienda no utilizar una sección de glosario sino colocar la descripción de las abreviaturas como parte del mismo cuerpo del texto. Por ejemplo, RTOS (*Real*

Time Operating System, Sistema Operativo de Tiempo Real) o en caso de considerarlo apropiado mediante notas a pie de página.

Si se desea indicar alguna página web utilizar el siguiente formato de referencias bibliográficas, dónde las referencias se detallan en la sección de bibliografía de la memoria,utilizado el formato establecido por IEEE en [2]. Por ejemplo, "el presente trabajo se basa en la plataforma EDU-CIAA-NXP, la cual se describe en detalle en [6]".

2.2. Figuras y tablas

Al insertar figuras en la memoria se deben considerar determinadas pautas. Para empezar, usar siempre tipografía claramente legible. Luego, tener claro que es incorrecto escribir por ejemplo esto: "El diseño elegido es un cuadrado, como se ve en la siguiente figura:"

La forma correcta de utilizar una figura es la siguiente: "Se eligió utilizar un cuadrado azul para el logo, el cual se ilustra en la figura 2.1".

FIGURA 2.1: Ilustración del cuadrado azul que se eligió para el diseño del logo.

El texto de las figuras debe estar siempre en español, excepto que se decida reproducir una figura original tomada de alguna referencia. En ese caso la referencia de la cual se tomó la figura debe ser indicada en el epígrafe de la figura e incluida como una nota al pie, como se ilustra en la figura 2.2.

FIGURA 2.2: Imagen ilustrativa de tal cosa, tomada de la página oficial del fabricante del procesador[1].

[1] https://goo.gl/images/i7C70w

La figura y el epígrafe deben conformar una unidad cuyo significado principal pueda ser comprendido por el lector sin necesidad de leer el cuerpo central de la memoria. Para eso es necesario que el epígrafe sea todo lo detallado que corresponda y si en la figura se utilizan abreviaturas entonces aclarar su significado en el epígrafe o en la misma figura.

FIGURA 2.3: El lector no sabe por qué de pronto aparece esta figura.

Nunca colocar una figura en el documento antes de hacer la primera referencia a ella, como se ilustra con la figura 2.3, porque sino el lector no comprenderá por qué de pronto aparece la figura en el documento, lo que distraerá su atención.

Se pueden colocar dos figuras una al lado de la otra como se puede ver en la figura 2.4 y referirse tanto a la subfigura 2.4a como a la subfigura 2.4b.

(A) Paracanthurus Hepatus[2]

(B) Amphiprion Ocellaris[3]

FIGURA 2.4: Especies exóticas de peces tropicales.

Para las tablas utilizar el mismo formato que para las figuras, sólo que el epígrafe se debe colocar arriba de la tabla, como se ilustra en la tabla 2.1. Observar que sólo algunas filas van con líneas visibles y notar el uso de las negritas para los encabezados.

En cada capítulo se debe reiniciar el número de conteo de las figuras y las tablas, por ejemplo, Fig. 2.1 o Tabla 2.1, pero no se debe reiniciar el conteo en cada sección. **Por suerte la plantilla se encarga de esto por nosotros.**

[1]GFDL 1.2, https://commons.wikimedia.org/w/index.php?curid=654994

[3]De Tewy - Trabajo propio, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=1382534

[4]Fuente: http://mercadolibre.com.ar

TABLA 2.1: Costos estimados de distintas especies de peces[4].

Especie	**Tamaño**	**Valor aprox.**
Amphiprion Ocellaris	10 cm	$ 6.000
Hepatus Blue Tang	15 cm	$ 7.000
Zebrasoma Xanthurus	12 cm	$ 6.800
Cirujano Naso Rubio	14 cm	$ 5.700
Zebrasoma Cirujano	10 cm	$ 4.500
Zebrasoma Desjardinii	10 cm	$ 3.200

2.3. Ecuaciones

Al insertar ecuaciones en la memoria estas se deben numerar de la siguiente forma:

$$ds^2 = c^2 dt^2 \left(\frac{d\sigma^2}{1 - k\sigma^2} + \sigma^2 \left[d\theta^2 + \sin^2 \theta d\phi^2 \right] \right) \tag{2.1}$$

Es importante tener presente que en el caso de las ecuaciones estas pueden ser referidas por su número, como por ejemplo "tal como describe la ecuación 2.1", pero también es correcto utilizar los dos puntos, como por ejemplo "la expresión matemática que describe este comportamiento es la siguiente:"

$$\frac{\hbar^2}{2m} \nabla^2 \Psi + V(\mathbf{r}) \Psi = -i\hbar \frac{\partial \Psi}{\partial t} \tag{2.2}$$

Para las ecuaciones se debe utilizar un tamaño de letra equivalente al utilizado para el texto del trabajo, en tipografía cursiva y preferentemente del tipo Times New Roman o similar. El espaciado antes y después de cada ecuación es de aproximadamente el doble que entre párrafos consecutivos del cuerpo principal del texto. **Por suerte la plantilla se encarga de esto por nosotros.**

Capítulo 3

Diseño e Implementación

3.1. Diseño

En esta sección se tratan las consideraciones de diseño que se utilizaron para desarrollar el sistema siguiendo los lineamientos de trabajo planteados en la asignatura "Ingeniería de Software en Sistemas Embebidos". Se tomó como guía el *port* de LPCOPEN v2.16 [4], provisto por NXP para la familia de microcontroladores LPC43XX, que contiene un ejemplo de aplicación sencillo que integra freeRTOS y lwIP. Para el servidor web se utilizó un ejemplo de los desarrolladores del stack TCP/IP lwIP [9]. A su vez, también fueron consultados otros ejemplos de utilización de lwIP con la CIAA [7] y [5].

3.1.1. Principio de funcionamiento

En términos generales, un servidor web es un programa que permite publicar información en la *World Wide Web* (WWW) o una red de área local (LAN). Esta información puede ser consultada desde cualquier navegador web estándar como Mozilla Firefox, Internet Explorer o Google Chrome, entre otros. El servidor opera utilizando el protocolo HTML [1] en la capa de aplicación del modelo OSI [11] y establece un canal de comunicación entre un servidor y un cliente. El servidor ejecuta las peticiones del cliente.

En el presente trabajo, el servidor web se encuentra embebido en la plataforma CIAA y proporciona al usuario una interfaz para monitorear y controlar el sistema. El servidor web es capaz de soportar los siguientes servicios:

- HTTP 2.0: HyperText Transfer Protocol (HTTP) es un protocolo orientado a transacciones y sigue el esquema petición/respuesta entre un cliente y un servidor [1]. En particular, la version 2.0 posibilita la utilización de conexiones persistentes o *keep-alive sessions* que permiten transmitir múltiples pares de petición/respuesta multiplexados sobre una única conexión TCP. De esta manera se puede reducir el tráfico y mejorar el desempeño en cuanto al tiempo de respuesta del servidor web. Resulta necesario para implementar métodos AJAX.
- SSI: *Server Side Include* (SSI) es un lenguaje interpretado sencillo que consiste en un conjunto de directivas que se incluyen dentro de la codificación de una página HTML y que se evalúan en el servidor web cuando la página HTML es solicitada. Permite incluir contenido generado en forma dinámica en tiempo de ejecución.

- AJAX: *Asynchonous JavaScript And XML* (AJAX) es un conjunto de técnicas utilizadas desde el lado del cliente para enviar y recibir información de un servidor en forma asincrónica. Es una forma de desacoplar el intercambio de información y la visualización de la página web. Permite que se cambie el contenido de una página web en forma dinámica sin necesidad de recargar la totalidad de la página. Para implementar AJAX, el servidor debe soportar directivas SSI. El cliente debe soportar JavaScript para peticionar información en segundo plano y es responsable de realizar dichas peticiones a intervalos regulares.
- Formularios web (peticiones GET): Los formularios se utilizan para enviar información al servidor. GET es un método del protocolo HTTP que se utiliza para solicitar información de un servidor [3]. Durante una petición GET se produce un *request* HTTP por parte del cliente y un *response* del servidor. La solicitud de información se transmite a través de la *Uniform Resource Identifier* (URI) agregando parámetros a la *Uniform Resource Locator* (URL). En la presente implementación el método GET se utiliza para enviar información al servidor en lugar de utilizar lo que sería más correcto, el método POST, debido a la mayor sencillez del primero por sobre el segundo.
- CGI (peticiones GET): *Common Gateway Interface* (CGI) es un estándar que provee una interfaz sencilla para ejecutar programas externos en un servidor HTTP [8]. En esta implementación se utiliza CGI para ejecutar rutinas específicas dentro del microcontrolador y generar una nueva página web como respuesta. Las funciones asociadas a un CGI se ejecutan cuando se envía un formulario web con el método GET al servidor para su procesamiento.

3.1.2. Arquitectura

El sistema está diseñado con una arquitectura de capas, según se puede observar en la figura 3.1. Agrupados por color se pueden apreciar los distintos niveles de abstracción y, dentro de cada capa, los bloques funcionales que la integran .

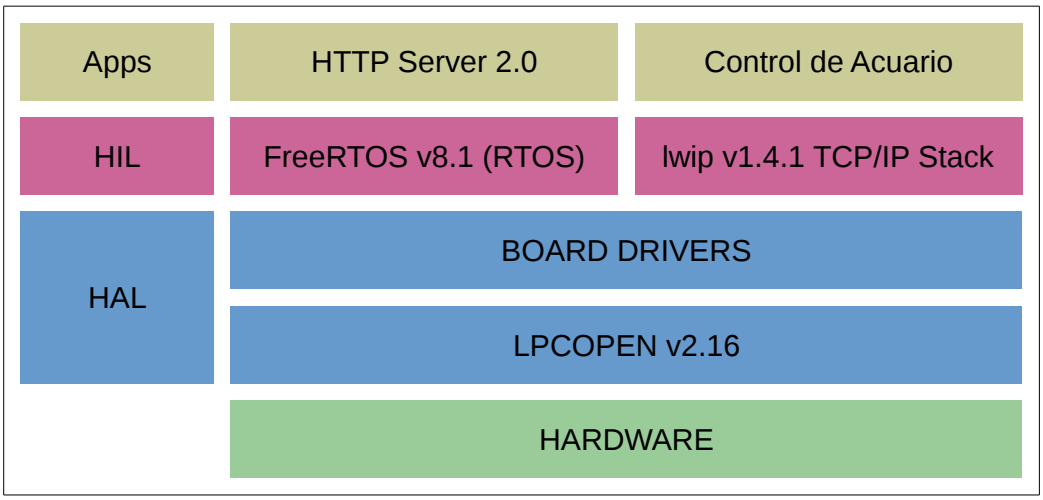

FIGURA 3.1: Diagrama en bloques del *firmware*.

En cuanto a los niveles de abstracción, la primer capa la constituye el *hardware* de la plataforma CIAA. La segunda capa, *Hardware Abstraction Layer* (HAL), permite desacoplar las capas superiores del *hardware*. Aquí se encuentran los *drivers* del fabricante del microcontrolador, en el bloque funcional LPCOPEN, y los desarrollados para la plataforma CIAA en el bloque BOARD. La capa *Hardware Independent Layer* (HIL) incluye los módulos del RTOS y el *stack* TCP/IP y, como su nombre lo indica, no está asociada a ningún *hardware* en particular. Finalmente, la última capa contiene dos aplicaciones, el servidor web y el control del acuario.

Se puede apreciar la estructura jerárquica de archivos del sistema en la figura 3.2, donde se han incluido los archivos fuente relevantes de cada bloque.

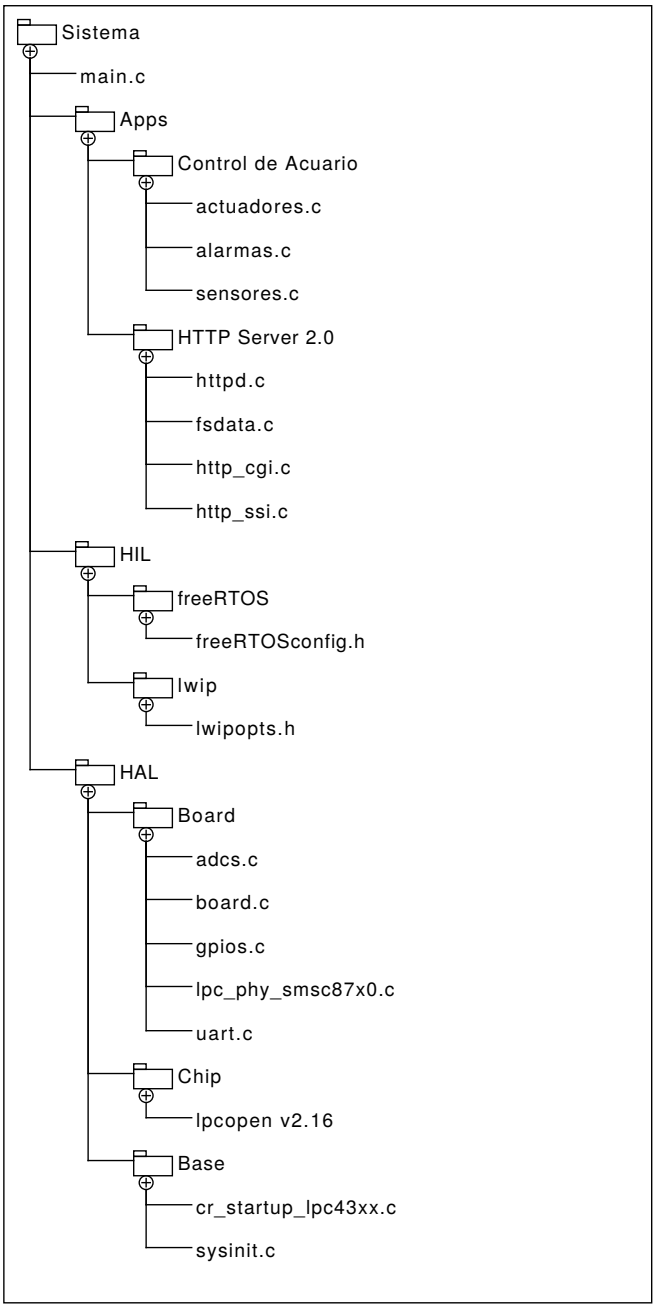

FIGURA 3.2: Estructura jerárquica de archivos.

3.1.3. Lógica de control

Se presenta una descripción en pseudocódigo de las funciones más relevantes de la lógica de control. El lazo principal de control consiste en una tarea de freeRTOS, llamada vControl, que se ejecuta periódicamente a intervalos regulares de 500 ms. Esta tarea llama secuencialmente a tres funciones auxiliares para actualizar los valores de los sensores, actualizar los valores de las alarmas y controlar los actuadores del sistema, respectivamente.

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];           //ON or OFF
state_t actuatorState[MAX_ACTUATOR_NUMBER];     //ON or OFF

void vControl() {

  initGlobalVariables();

  period = 500 ms;

  while(1) {

    ticks = xTaskGetTickCount();

    updateSensors();

    updateAlarms();

    controlActuators();

    vTaskDelayUntil(&ticks, period);
  }
}
```

ALGORITMO 3.1: Pseudocódigo del lazo principal de control.

```
void updateSensors() {

  // for every sensor
  for (i=0; i < MAX_SENSOR_NUMBER; i++) {

    // read ADCs
    data_i = readDataFromADC(sensorChannel_i);

    // apply scale convertion
    data_i = scaleConvertion(data_i, i );

    // impact sensor value to global variable
    sensorValue[i] = data_i
  }
}
```

ALGORITMO 3.2: Pseudocódigo del control de los sensores.

```
void updateAlarms() {

  // for every sensor; i goes from 0 to 2
  for ( i=0; i < MAX_SENSOR_NUMBER; i++ ) {

    //check sensor upper threshold
    if ( sensorValue[i] > sensorMaxLimit[i] ) {
      setAlarmState(2*i); // alarmState's index goes from 0 to 5
    }
    else {
      clearAlarmState(2*i);
    }

    //check sensor lower threshold
    if ( sensorValue[i] < sensorMinLimit[i] ) {
      setAlarmState( (2*i)+1 );
    }
    else {
      clearAlarmState( (2*i)+1 );
    }
  }
}
```

ALGORITMO 3.3: Pseudocódigo del control de las alarmas.

```
void controlActuators() {

  state_t actuatorFakeState[MAX_ACTUATOR_NUMBER];

  // for every actuator,
  for( i=0; i < MAX_ACTUATOR_NUMBER; i++ ) {
    // Get current state
    actuatorFakeState[i] = getActuatorState(i);

    // If control is enable checks alarm and operates actuators x,y,...
    if (ENABLE == alarmControl[i]){

      if( ON == alarmState[i] ) {
        actuatorFakeState[x] = (Necesary action); // ON or OFF
        actuatorFakeState[y] = (Necesary action);
        alarmFlag[i] = ON;    // flag indicates control actions underway
      }
      // alarm = OFF & flag = ON then control actions should be stoped
      else if ( ON == alarmFlag[i] ) {
        actuatorFakeState[x] = !(Necesary action);
        actuatorFakeState[y] = !(Necesary action);
        alarmFlag[i] = OFF;
      }
    }
  }

  // Impact changes to actuators
  for( i=0; i < MAX_ACTUATOR_NUMBER; i++ ) {

    actuatorState[i] = actuatorFakeState[i];
  }
}
```

ALGORITMO 3.4: Pseudocódigo del control de los actuadores.

3.2. Implementación

Para acceder a los *sockets*, el servidor web utiliza la RAW API[1] de lwIP que emplea una serie de *callbacks* para manejar los eventos de la comunicación. La función httpd_init() inicia el demonio del servidor y debe ser llamada luego de que el stack se encuentre inicializado con la función lwip_init(). Para alojar el contenido HTML que se debe servir se utiliza un sistema de archivos dedicado implementado en ROM [10]. A través del *script* makefsdata, provisto por lwIP, se puede personalizar los archivos HTML e integrarlos al sistema mediante el archivo fuente fsdata.c. Se debe tener especial cuidado al utilizar las funciones de la RAW API en un entorno con múltiples hilos de ejecución como freeRTOS, ya que las funciones no cuentan con protección frente a accesos concurrentes. Las funciones de esta API deben ser invocadas desde un único hilo de ejecución.

El servidor inicia con la configuración detallada en la tabla 3.1.

TABLA 3.1: Configuración MAC del servidor web.

Parámetro	*Valor*
Dirección MAC	00:60:37:12:34:56
DHCP	Deshabilitado
Dirección IP	192.168.200.99
Máscara de red	255.255.255.0
Puerta de enlace	192.168.200.1
Puerto	80

3.2.1. Interfaz web embebida

Se presenta la interfaz web implementada para monitorear y controlar el acuario. Esta consta de un menú de navegación en la parte izquierda, que permite acceder a tres pantallas, detalladas a continuación:

- Inicio, con un resumen del estado del sistema sin posibilidad de modificaciones. Se puede observar en la figura 3.3.
- Control, donde se encuentran los controles para operar las alarmas y actuadores. Se puede observar en la figura 3.4.
- Configuración, donde se puede cambiar la configuración de red del servidor. Se muestra en la figura 3.5.

En la página inicio se pueden observar tres bloques de información referida a los sensores, las alarmas y los actuadores, respectivamente. Esta información se actualiza una vez por segundo mediante el método AJAX, según está documentado en la sección 3.2.4. El estado de las alarmas se indica como "NORMAL" en color verde o "ALARMA" en color rojo cuando el control automático está habilitado, o el mismo texto en gris cuando el control automático está deshabilitado. El estado de los actuadores se indica como "ENCENDIDO" en color verde o "APAGADO" en color rojo sin importar el estado del control de alarmas.

[1] http://lwip.wikia.com/wiki/Raw/TCP

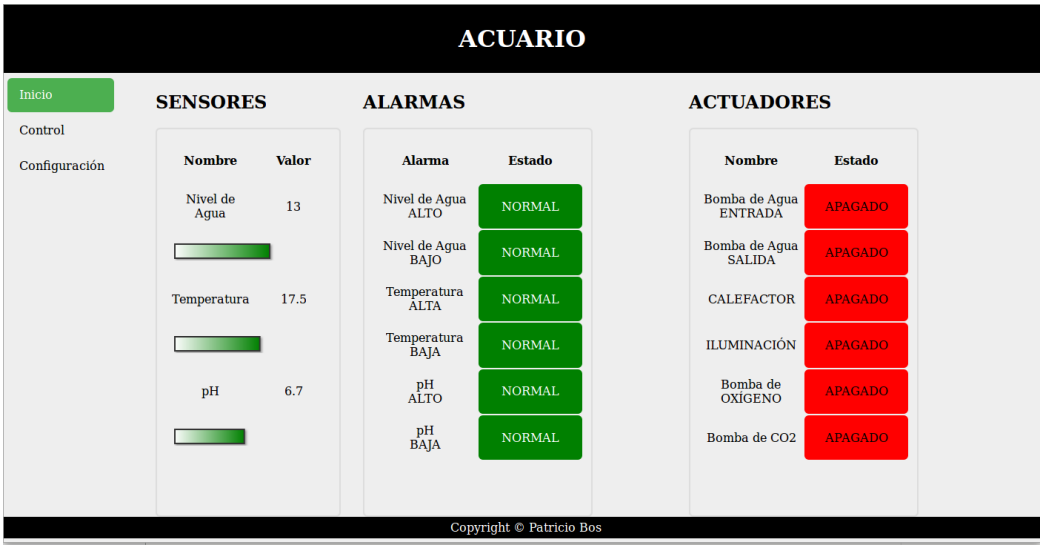

FIGURA 3.3: Interfaz web: página principal Inicio.

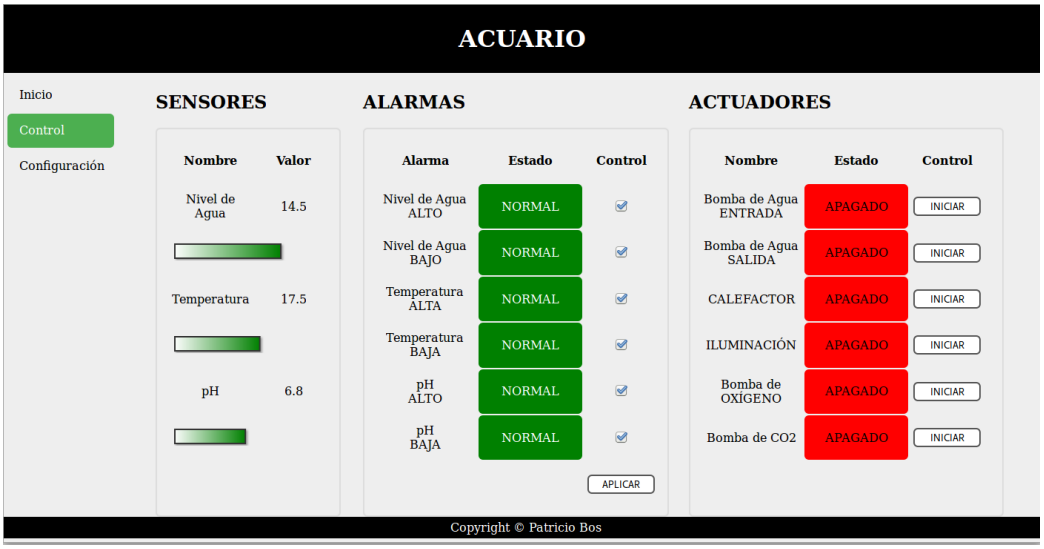

FIGURA 3.4: Interfaz web: página para controlar el sistema.

FIGURA 3.5: Interfaz web: configuración de red.

Tanto en la página Inicio como en la página Control, en el apartado SENSORES se muestra el valor numérico de cada sensor junto con una representación gráfica de dicho valor. Se utiliza un gráfico de barra horizontal proporcional al valor numérico que se construye en tiempo de ejecución mediante rutinas de JavaScript. Cada sensor tiene una escala que se ajusta según su valor máximo definido dentro del microcontrolador. Mientras exista una condición de alarma para un sensor, tanto por exceso como por defecto, la gráfica asociada se muestra en color rojo, caso contrario se muestra en color verde.

La página Control, según se puede observar en la figura 3.6, posee además de la información mostrada en la página Inicio, columnas llamadas "control" para operar sobre las alarmas y los actuadores, respectivamente.

En el apartado ALARMAS se encuentra un *checkbox* para cada alarma que permite habilitar el control automático sobre la condición de alarma o deshabilitarlo. Notar que existe un botón "APLICAR" al final de la columna de control que impacta el contenido de los *checkboxes* en la lógica de control dentro del microcontrolador. Para lograr esto se utiliza el método CGI según se documenta en la sección 3.2.3. El valor seleccionado/no seleccionado de los *checkboxes* se guarda en los datos de sesión del navegador web del cliente.

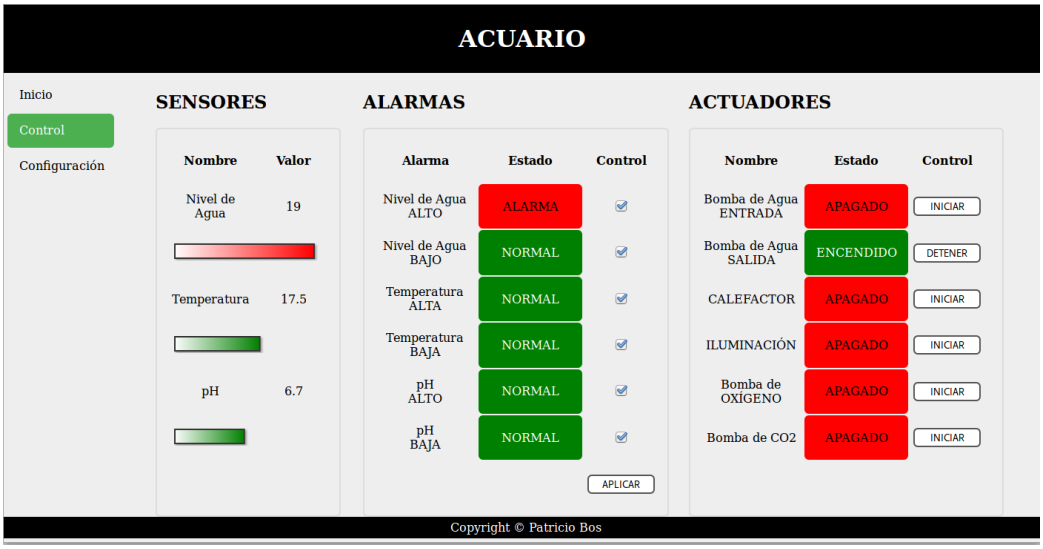

FIGURA 3.6: Interfaz web: condición de alarma presente.

En el apartado actuadores, y dentro de la columna control, se pueden observar botones para iniciar o detener los actuadores en forma manual. En el caso particular que se muestra en la figura 3.6, frente a la condición de alarma por nivel de agua alto, la lógica de control mantiene la bomba de entrada apagada y la bomba de salida encendida. Los botones de los actuadores mencionados se encuentran deshabilitados mientras la alarma esté presente y si fueran presionados no producirían efectos sobre el sistema. Al posar el cursor sobre un botón deshabilitado aparece un cuadro de diálogo como se muestra en la figura 3.7, indicando que se debe deshabilitar el control de alarma para operar el actuador manualmente.

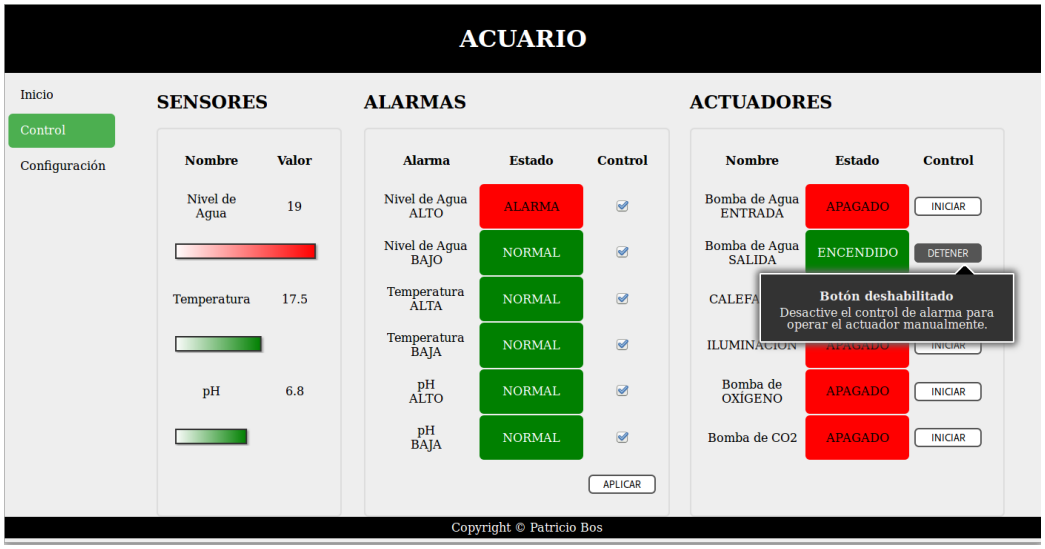

FIGURA 3.7: Interfaz web: botón deshabilitado por el control.

Cuando el control automático de una alarma se encuentra deshabilitado, el texto "NORMAL" o "ALARMA" se muestra sobre un fondo gris, como se puede observar en la figura 3.8. Como ya fuera mencionado, los actuadores asociados a esta condición son las dos bombas de agua que para el caso del control automático deshabilitado no se encuentran inhibidos y pueden ser operados manualmente. Cabe mencionar que en la figura 3.8 se puede ver cómo la bomba de agua SALIDA se encuentra apagada pese a que la condición de alarma se encuentra presente. Sin pérdida de generalidad, este comportamiento puede extenderse para la totalidad de las condiciones de alarma y sus actuadores asociados.

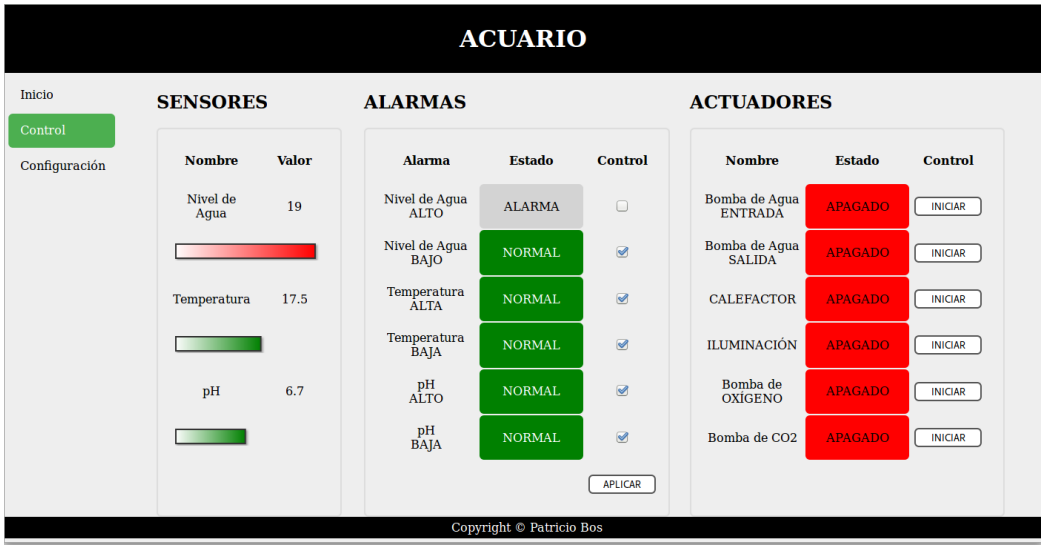

FIGURA 3.8: Interfaz web: control deshabilitado.

3.2.2. Server Side Includes

La interfaz web contiene información del estado del sistema que se genera en forma dinámica en tiempo de ejecución. Mediante directivas SSI embebidas en las páginas HTML, el servidor web reemplaza cadenas de caracteres debidamente señalizadas por el contenido dinámico. En el algoritmo 3.5 se muestra la función que inicializa el soporte para SSI, registrando la función que se debe llamar para manejar las directiva SSI, la lista de etiquetas válidas y el número de etiquetas.

```
/** Set the SSI handler function.
  *
  * @param ssi_handler the SSI handler function
  * @param tags an array of SSI tag strings to search for in SSI-enabled
     files
  * @param num_tags number of tags in the 'tags' array
  */
void http_set_ssi_handler(tSSIHandler ssi_handler, const char **tags,
    int num_tags) {

  g_pfnSSIHandler = ssi_handler;
  g_ppcTags = tags;
  g_iNumTags = num_tags;
}
```

ALGORITMO 3.5: Inicialización del SSI handler

Se instala en manejador de directivas SSI con la siguiente línea de código:

```
http_set_ssi_handler(SSIHandler, pccSSITags, sizeof( pccSSITags ) /
    sizeof( char * ) );
```

Las etiquetas tienen la forma de comentario HTML, <! - - # tag - - >donde la palabra “tag” se debe reemplazar por los distintos valores particulares. Se utilizaron las etiquetas definidas en el algoritmo 3.6 para el estado encendido/apagado de los actuadores, los valores de los sensores, el estado normal/alarma de las alarmas y el control automático activado/desactivado, respectivamente.

```
// The SSI strings that are embedded in the served html files.
static const char *pccSSITags[] = {
  "act0",
  "act1",
  "act2",
  "act3",
  "act4",
  "act5",
  "sensor0",
  "sensor1",
  "sensor2",
  "alarma0",
  "alarma1",
  "alarma2",
  "alarma3",
  "alarma4",
  "alarma5",
  "ctrlAlrm0",
  "ctrlAlrm1",
  "ctrlAlrm2",
  "ctrlAlrm3",
  "ctrlAlrm4",
  "ctrlAlrm5" };
```

ALGORITMO 3.6: Definición de etiquetas SSI

A continuación se presenta, en el algoritmo 3.7, un ejemplo parcial de la función para reemplazar las etiquetas embebidas en el código HTML por el contenido dinámico. El ejemplo se restringe por simplicidad a las primeras seis etiquetas, referidas al estado de los actuadores y puede ser extendido al resto de las etiquetas de manera análoga.

Se muestra la función SSIHandler que es invocada cada vez que el servidor HTTPD detecta una etiqueta de la Forma <! - - # tag - - >en un archivo con extensión .shtml, .ssi o .shtm, donde "tag" aparece como una de las etiquetas suministradas a http_set_ssi_handler en el *array* ppcTags. El parámetro iIndex proporciona el índice de base cero de la etiqueta como se encuentra en el *array* ppcTags e identifica la etiqueta que se va a procesar.

```
1  uint16_t SSIHandler( int iIndex, char *pcBuffer, int iBufferLength )
2  {
3
4    // Unused parameter.
5    ( void ) iBufferLength;
6
7    // The SSI handler function that generates text depending on the index
       of the SSI tag encountered.
8
9    char *ptrState;
10
11   switch( iIndex )
12   {
13
14   case ssiACT0_INDEX:
15     ptrState = getActuatorCharState(portNum_0);
16     strcpy( pcBuffer, ptrState );
17     break;
18
19   case ssiACT1_INDEX:
20     ptrState = getActuatorCharState(portNum_1);
21     strcpy( pcBuffer, ptrState );
22     break;
23
24   case ssiACT2_INDEX:
25     ptrState = getActuatorCharState(portNum_2);
26     strcpy( pcBuffer, ptrState );
27     break;
28
29   case ssiACT3_INDEX:
30     ptrState = getActuatorCharState(portNum_3);
31     strcpy( pcBuffer, ptrState );
32     break;
33
34   case ssiACT4_INDEX:
35     ptrState = getActuatorCharState(portNum_4);
36     strcpy( pcBuffer, ptrState );
37     break;
38
39   case ssiACT5_INDEX:
40     ptrState = getActuatorCharState(portNum_5);
41     strcpy( pcBuffer, ptrState );
42     break;
43
44   return strlen( pcBuffer );
45 }
```

ALGORITMO 3.7: Handler de etiquetas SSI

La cadena de caracteres que se copia en pcBuffer se agrega luego de la etiqueta "<! - - # tag - - >" en el archivo que se envía al cliente.

3.2.3. Common Gateway Interface

Se utilizan funciones CGIs asociadas a la petición de un nombre específico de archivo. La interfaz web, a través de un formulario GET, solicita un archivo con extensión .cgi al servidor y envía en la URL los argumentos para la función CGI. En el algoritmo 3.8 se muestra la función para iniciar el soporte CGI que recibe como argumento un *array* de pares "nombre de archivo"/función. Las funciones registradas se ejecutan cuando el cliente solicita el archivo con extensión .cgi asociado.

```
1  /** Set an array of CGI filenames/handler functions
2    * @param cgis an array of CGI filenames/handler functions
3    * @param num_handlers number of elements in the 'cgis' array
4    */
5
6  void http_set_cgi_handlers(const tCGI *cgis, int num_handlers) {
7
8    LWIP_ASSERT("no cgis given", cgis != NULL);
9    LWIP_ASSERT("invalid number of handlers", num_handlers > 0);
10
11   g_pCGIs = cgis;
12   g_iNumCGIs = num_handlers;
13 }
```

ALGORITMO 3.8: Handler de funciones CGI

Se instala en manejador de funciones CGI con la siguiente línea de código:

```
1 http_set_cgi_handlers(cgi_handlers, (sizeof (cgi_handlers) / sizeof (
    tCGI) ) );
```

En el algoritmo 3.9 se pueden ver las definiciones de pares "nombre de archivo"/función. Estas funciones manejan el estado encendido/apagado de los actuadores y el estado habilitado/deshabilitado del control automático de las alarmas, respectivamente.

```
1 tCGI cgi_handlers[]={
2
3     {"/actuadores.cgi",actuatorsHandler},
4     {"/alarmas.cgi",alarmsHandler}
5 };
6 tCGI * ptrCGIHandlers;
```

ALGORITMO 3.9: Registro de nombres y funciones CGI

3.2.4. AJAX

La implementación del servidor web soporta *Asynchronous JavaScript and XML* (AJAX). AJAX es un método que permite actualizar el contenido de una página web en segundo plano, o sin recargar la totalidad de la página. El navegador web es responsable de solicitar regularmente un pequeño archivo con la información que se debe actualizar y de ubicar dicha información en lugares específicos de la página web. Todas las transacciones se inician a través del navegador web del lado del cliente. La utilización de este método da un aspecto dinámico al contenido web.

Para la utilización de AJAX se requiere que :

- El cliente solicite el archivo con extensión .ssi o .shtml a intervalos regulares (1 segundo) y que el servidor pueda suministrar el archivo.
- El código fuente HTML embebido en el servidor contenga sentencias AJAX.
- El archivo solicitado esté escrito usando directivas SSI. De esta manera, dentro del servidor se reemplazan las etiquetas SSI y la operatoria resulta transparente para el cliente.
- El servidor web soporte la característica HTTP *persistent connection* que está incluida en el *stack* TCP/IP lwIP.

Se presenta en la figura 3.9 un diagrama de secuencia con las transacciones entre el navegador del cliente y el servidor. Como ya fuera mencionado, el navegador es responsable de iniciar la transacción y luego procesar y posicionar la respuesta del servidor. En el código fuente HTML embebido en el servidor se encuentran una serie de funciones en JavaScript que periódicamente realizan una petición de la página ajax.shtml, que contiene las directivas SSI, y luego procesan la respuesta.

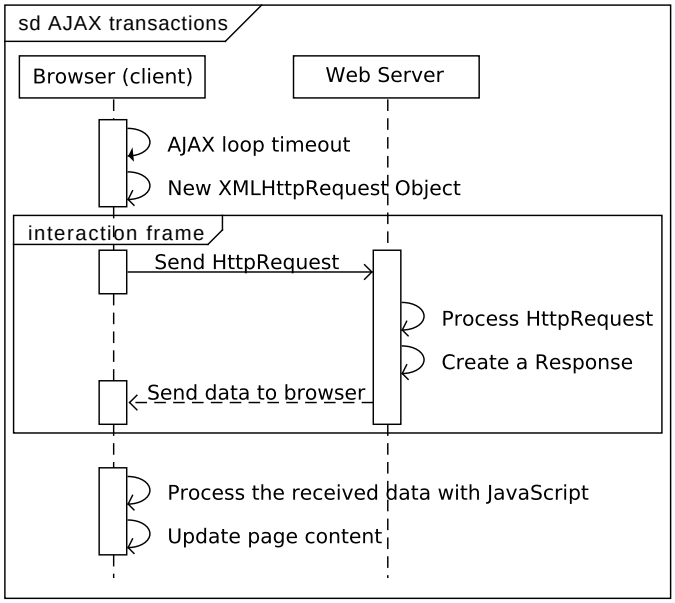

FIGURA 3.9: Diagrama de secuencia AJAX.

Capítulo 4

Ensayos y Resultados

4.1. Ensayos

Para verificar y validar la correcta implementación del sistema se procede a evaluar el funcionamiento aplicando técnicas de ensayos de caja negra.

4.1.1. Ensayos de caja negra

Se utiliza la técnica de inyección de fallas para evaluar la respuesta del sistema frente a las distintas condiciones de alarma posibles. El ensayo consiste en forzar las entradas de los sensores a valores que disparen las condiciones de alarma y contrastar las acciones de control del sistema con las esperadas.

Se utiliza la UART para realizar un seguimiento en el tiempo del valor de los sensores, del estado de las alarmas y del estado de los actuadores. Se tomaron muestras a una tasa de una muestra por segundo mientras se hacían variar las entradas del sistema para recorrer todas las posibles combinaciones de alarmas. La información fue procesada mediante rutinas de MATLAB elaboradas *ad-hoc* para obtener gráficas del estado del sistema que permitieran corroborar su correcto funcionamiento. Los resultados se pueden observar en la sección 4.2.

Si bien el sistema tiene seis condiciones de alarma, cada uno de los tres sensores tiene dos condiciones, una por exceso y otra por defecto, que son mutuamente excluyentes. Por este motivo, el máximo número de alarmas simultáneas que puede haber en un momento dado es tres. Para analizar el comportamiento del sistema se agrupan las respuestas según la cantidad de alarmas en conjuntos de una única alarma, dos alarmas o tres alarmas simultáneas.

Para garantizar que se recorre todo el espacio de posibilidades de alarma se elaboran tablas de decisión con las acciones que se deben tomar para las distintas combinaciones posibles. Se representa mediante una “Y” la condición de alarma presente. En caso contrario se utiliza una “N”. Las acciones de contingencia necesarias se indican con una “X” en el casillero correspondiente. Un casillero en blanco significa que esa acción en particular puede tener cualquiera de sus dos estados posibles sin que eso afecte la condición de alarma.

Se muestra en el cuadro 4.1 las acciones de contingencia para cuando hay una única condición de alarma en el sistema. En el cuadro 4.2 se indican las acciones para las distintas combinaciones de dos alarmas. En el cuadro 4.3 se indican las acciones de contingencia para las combinaciones de tres alarmas.

TABLA 4.1: Tabla de decisión para el control de una sola alarma.

CONDICIONES						
Nivel de agua alto	Y	N	N	N	N	N
Nivel de agua bajo	N	Y	N	N	N	N
Temperatura alta	N	N	Y	N	N	N
Temperatura baja	N	N	N	Y	N	N
pH alto	N	N	N	N	Y	N
pH bajo	N	N	N	N	N	Y
ACCIONES						
Encender bomba de entrada de agua		X	X			X
Apagar bomba de entrada de agua	X					
Encender bomba de salida de agua	X		X			X
Apagar bomba de salida de agua		X				
Encender calefactor				X		
Apagar calefactor			X			
Encender bomba de CO2					X	
Apagar bomba de CO2						X

TABLA 4.2: Tabla de decisión para el control de dos alarmas.

CONDICIONES												
Nivel de agua alto	Y	Y	Y	Y	N	N	N	N	N	N	N	N
Nivel de agua bajo	N	N	N	N	Y	Y	Y	Y	N	N	N	N
Temperatura alta	Y	N	N	N	Y	N	N	N	Y	Y	N	N
Temperatura baja	N	Y	N	N	N	Y	N	N	N	N	Y	Y
pH alto	N	N	Y	N	N	N	Y	N	Y	N	Y	N
pH bajo	N	N	N	Y	N	N	N	Y	N	Y	N	Y
ACCIONES												
Encender bomba de entrada de agua					X	X	X	X	X	X		X
Apagar bomba de entrada de agua	X	X	X	X								
Encender bomba de salida de agua	X	X	X	X					X	X		
Apagar bomba de salida de agua					X	X	X	X				X
Encender calefactor		X				X					X	
Apagar calefactor	X				X				X	X		X
Encender bomba de CO2			X				X		X		X	
Apagar bomba de CO2				X				X		X		

TABLA 4.3: Tabla de decisión para el control de tres alarmas.

CONDICIONES								
Nivel de agua alto	Y	Y	Y	Y	N	N	N	N
Nivel de agua bajo	N	N	N	N	Y	Y	Y	Y
Temperatura alta	Y	Y	N	N	Y	Y	N	N
Temperatura baja	N	N	Y	Y	N	N	Y	Y
pH alto	Y	N	Y	N	Y	N	Y	N
pH bajo	N	Y	N	Y	N	Y	N	Y
ACCIONES								
Encender bomba de entrada de agua					X	X	X	X
Apagar bomba de entrada de agua	X	X	X	X				
Encender bomba de salida de agua	X	X	X	X				
Apagar bomba de salida de agua					X	X	X	X
Encender calefactor			X	X			X	X
Apagar calefactor	X	X			X	X		
Encender bomba de CO2	X		X		X		X	
Apagar bomba de CO2		X		X		X		X

4.2. Resultados

En esta sección se muestran los resultados correspondientes a las distintas combinaciones de alarmas presentes en el sistema. Se puede observar el valor de los sensores junto con sus valores límite graficados en línea punteada y el estado lógico de los actuadores. Por simplicidad de la gráfica, el estado de los actuadores de iluminación y bomba de O_2 no se muestran para ningún caso de análisis, ya que no tienen efecto sobre las condiciones de alarma.

Se agrupan los resultados con el criterio ya mencionado de cantidad de alarmas simultáneas. En la sección 4.2.1 se muestran los resultados para una única alarma presente. En la sección 4.2.2 se pueden ver los resultados para dos alarmas presentes y finalmente, en la sección 4.2.3 se encuentran los resultados para tres alarmas simultáneas.

4.2.1. Respuesta a una única alarma

Existen seis combinaciones de una única alarma, según fueron listadas en el cuadro 4.1. En la figura 4.1 se muestra la respuesta del sistema frente a las alarmas de nivel de agua alto y nivel de agua bajo . En la figura 4.2 se puede ver la respuesta frente a las alarmas de temperatura alta y temperatura baja. Finalmente, en la figura 4.3 se grafica la respuesta frente a las alarmas de pH alto y pH bajo.

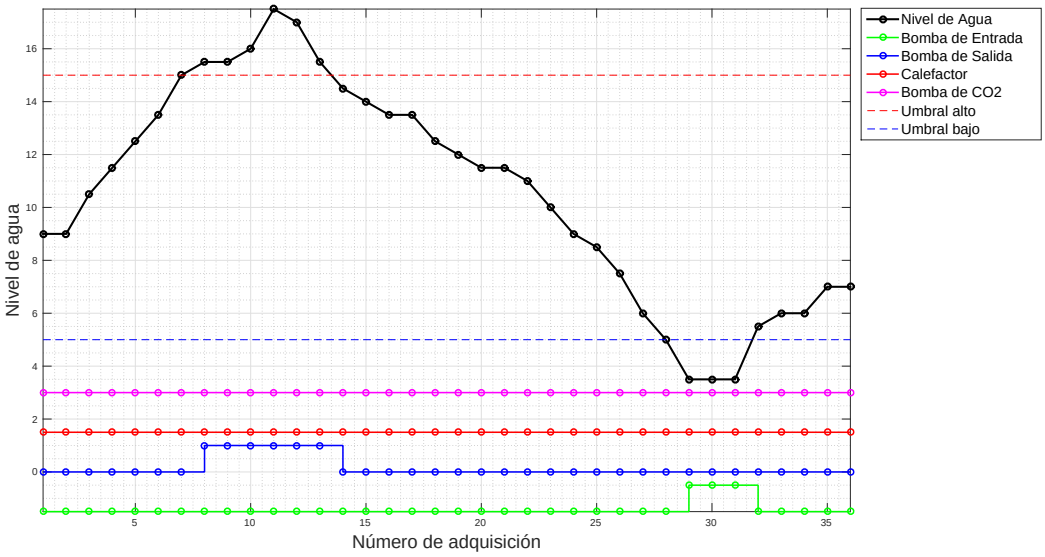

FIGURA 4.1: Respuesta del sistema frente a variaciones en el nivel de agua.

Observando la figura 4.1, se puede apreciar que en el momento en que el nivel de agua excede el valor límite superior se enciende la bomba de salida, que luego se apaga cuando valor vuelve a estar en el rango de valores permitidos. En forma análoga, cuando el nivel de agua supera el valor límite inferior se enciende la bomba de entrada para luego apagarse cuando el valor retorna a niveles normales.

Las variaciones en la temperatura en la figura 4.2 producen que se enciendan las bombas de entrada y salida para recircular el agua cuando el valor excede el

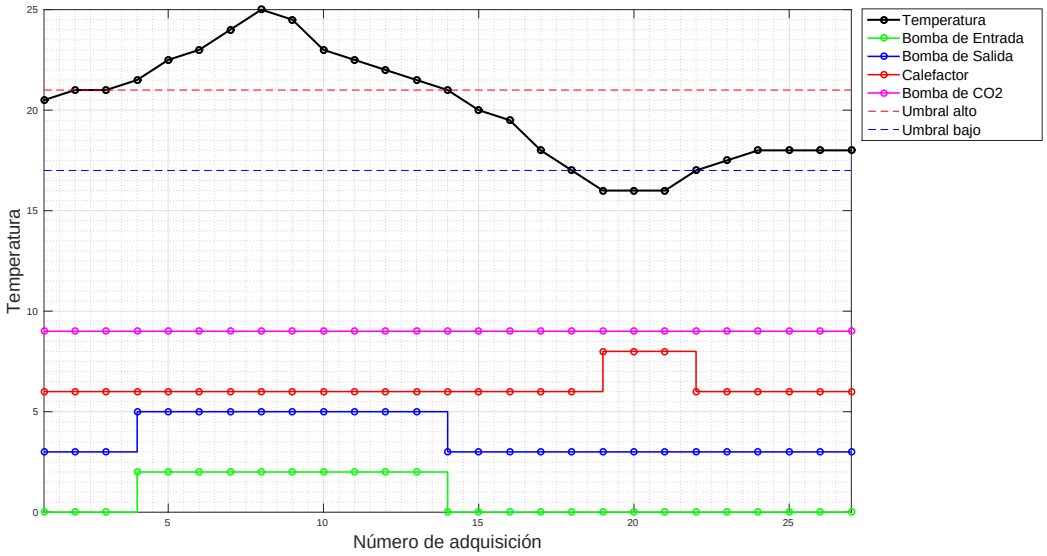

FIGURA 4.2: Respuesta del sistema frente a variaciones en la temperatura.

límite superior y que se encienda el calefactor cuando el valor de temperatura se encuentra por debajo del límite inferior. Todas los actuadores que se encienden o apagan por medidas de control vuelven al estado anterior al restablecerse los valores normales de temperatura.

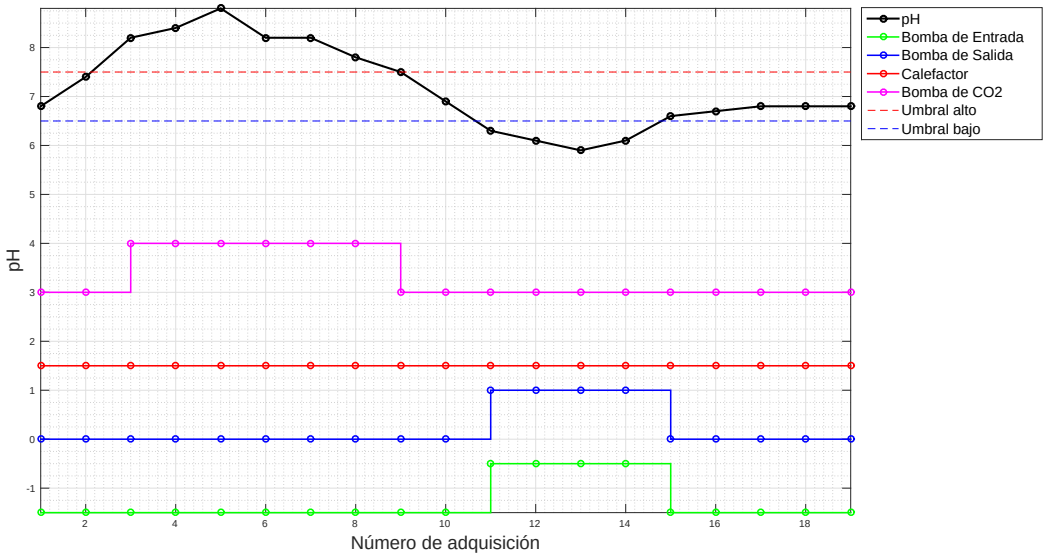

FIGURA 4.3: Respuesta del sistema frente a variaciones en el pH.

Como se puede observar en la figura 4.3, cuando varía el pH se enciende la bomba de CO_2 si el valor excede el límite superior, y este actuador se apaga cuando el valor se encuentra por debajo de dicho límite. En forma similar al comportamiento frente a la alarma de temperatura alta, cuando el valor de pH se encuentra por debajo del límite inferior, se encienden las bombas de entrada y salida para recircular el agua.

4.2.2. Respuesta a dos alarmas

Existen doce combinaciones de dos alarmas al mismo tiempo, según fueron listadas en el cuadro 4.2. En la figura 4.4 se muestra la respuesta del sistema para las distintas combinaciones de dos alarmas que incluyen la condición de nivel de agua alto. En la figura 4.5 se puede ver las respuestas frente a las combinaciones de dos alarmas que incluyen la condición de nivel de agua bajo. Finalmente, en la figura 4.6 se grafica la respuesta frente a las combinaciones de alarma de temperatura alta y baja junto con pH alto y bajo, respectivamente.

A partir de esta sección, los valores límite en línea punteada se grafican del mismo color que el sensor al cual refieren por simplicidad en la visualización.

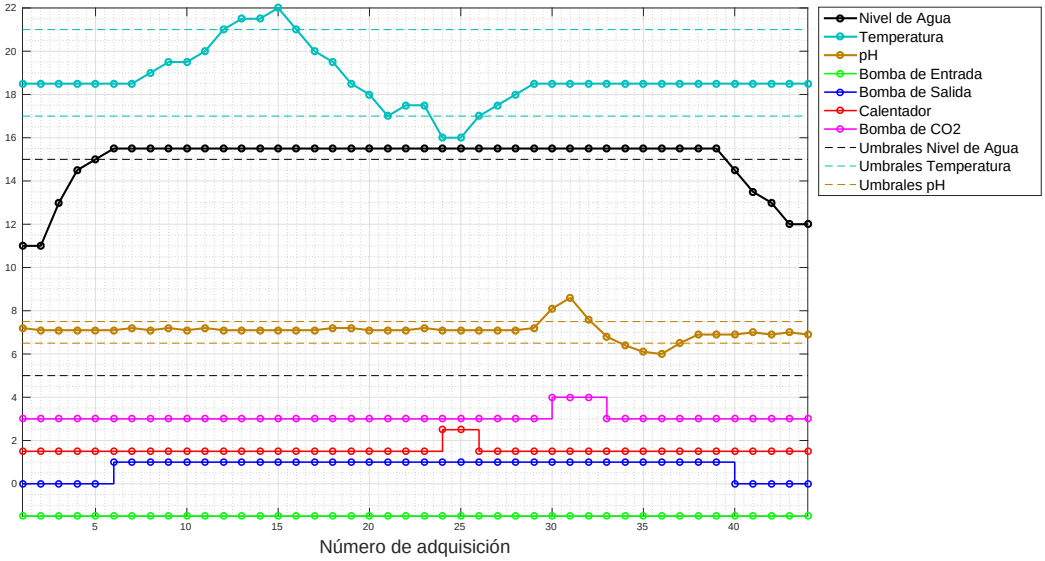

FIGURA 4.4: Respuesta a dos alarmas simultáneas.
Nivel de agua alto y otras.

Se puede observar en la figura 4.4 la condición de nivel de agua alto junto con las de temperatura alta y baja y junto a las de pH alto y bajo posteriormente. Cabe destacar que cuando la temperatura se encuentra por encima de su valor límite, la bomba de entrada se encuentra inhibida y no se enciende pese a que la acción de contingencia para temperatura alta es ciclar el agua. Esto se debe a que la condición de nivel de agua alto tiene mayor prioridad. Lo mismo ocurre con la condición de pH bajo que tiene la misma acción de contingencia que la de temperatura alta. Durante todo el tiempo que el nivel de agua supera su valor límite, la bomba de salida se encuentra encendida.

De manera análoga al análisis precedente, en la figura 4.5 se muestran condiciones de dos alarmas simultáneas donde una de ellas es la de nivel de agua bajo. En este caso, frente a las condiciones de temperatura alta y pH bajo, el actuador que se encuentra inhibido es la bomba de salida porque la condición de nivel de agua bajo tiene mayor prioridad que las anteriores. Durante todo el tiempo que el nivel de agua se encuentra por debajo de su límite inferior, la bomba de entrada permanece encendida y se apaga cuando el nivel de agua vuelve a estar entre los valores permitidos.

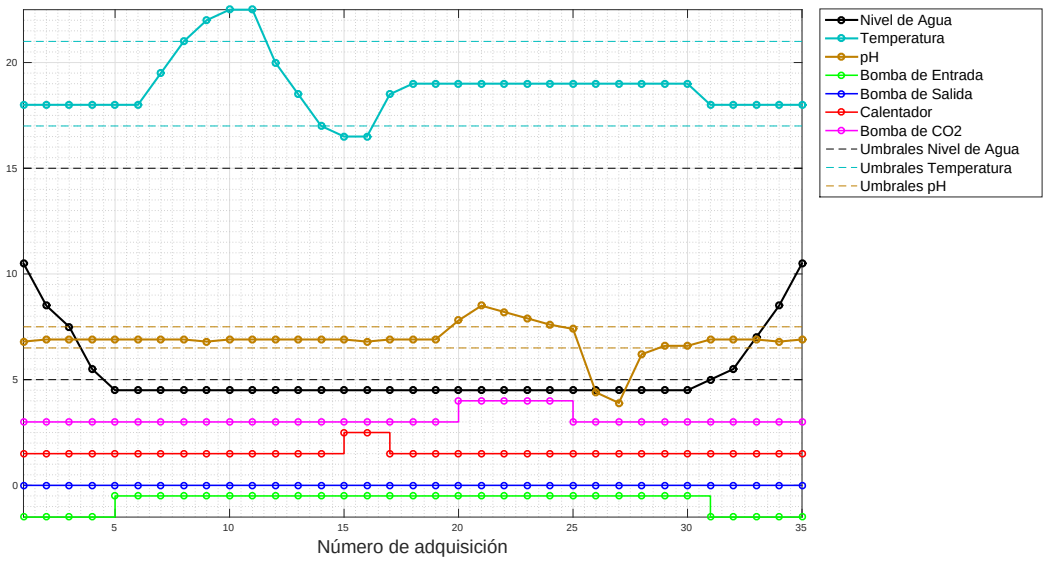

FIGURA 4.5: Respuesta a dos alarmas simultáneas.
Nivel de agua bajo y otras.

Las últimas cuatro combinaciones de dos alarmas se producen entre las asociadas a los sensores de temperatura y pH. En la figura 4.6 se puede observar cómo se encienden las bombas de entrada y salida si al menos una de las condiciones de temperatura alta o pH bajo están presentes. Por otra parte, si la temperatura se encuentra por debajo de su valor mínimo permitido se enciende el calefactor y si el pH supera su límite se enciende la bomba de CO_2.

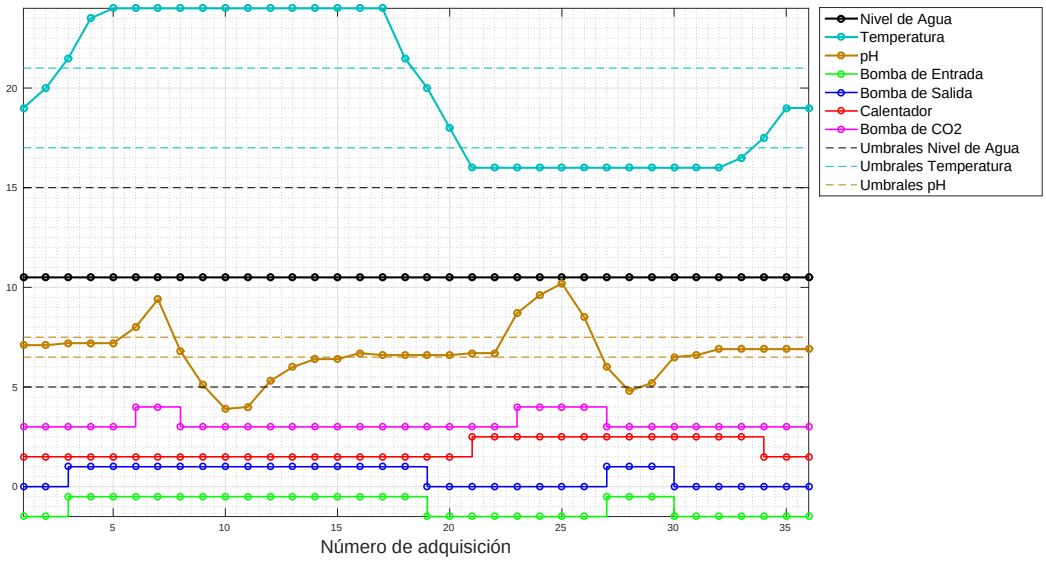

FIGURA 4.6: Respuesta a dos alarmas simultáneas.
Temperatura y pH.

4.2.3. Respuesta a tres alarmas

Existen 8 combinaciones de 3 alarmas simultáneas según fueron listadas en el cuadro 4.3. Se presentan agrupadas en dos gráficas, en las figuras 4.7 y 4.8, según se mantenga fija la condición de alarma por nivel de agua alto o bajo, respectivamente.

Se puede observar en la figura 4.7, que las acciones de control de alarma son las esperadas, teniendo en cuenta que, al igual que en el caso de dos alarmas, cuando se encuentra presente la condición de alarma de nivel de agua alto, se inhibe el encendido de la bomba de entrada.

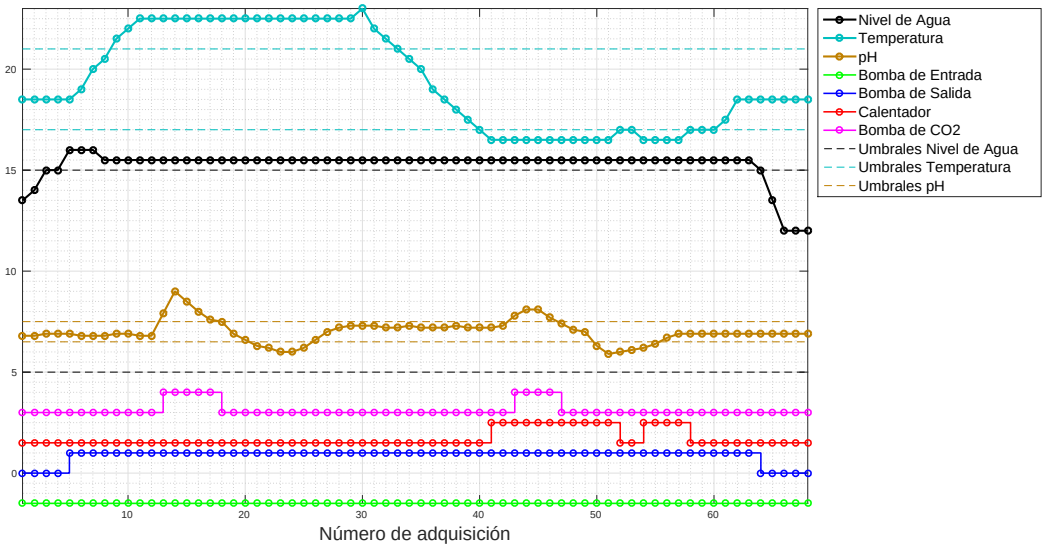

FIGURA 4.7: Tres alarmas: nivel de agua alto y otras.

En forma análoga al análisis precedente, en la figura 4.8 se puede observar que las acciones de control son las esperadas, teniendo en cuenta que en este caso el actuador inhibido es la bomba de salida por la condición de alarma de nivel de agua bajo.

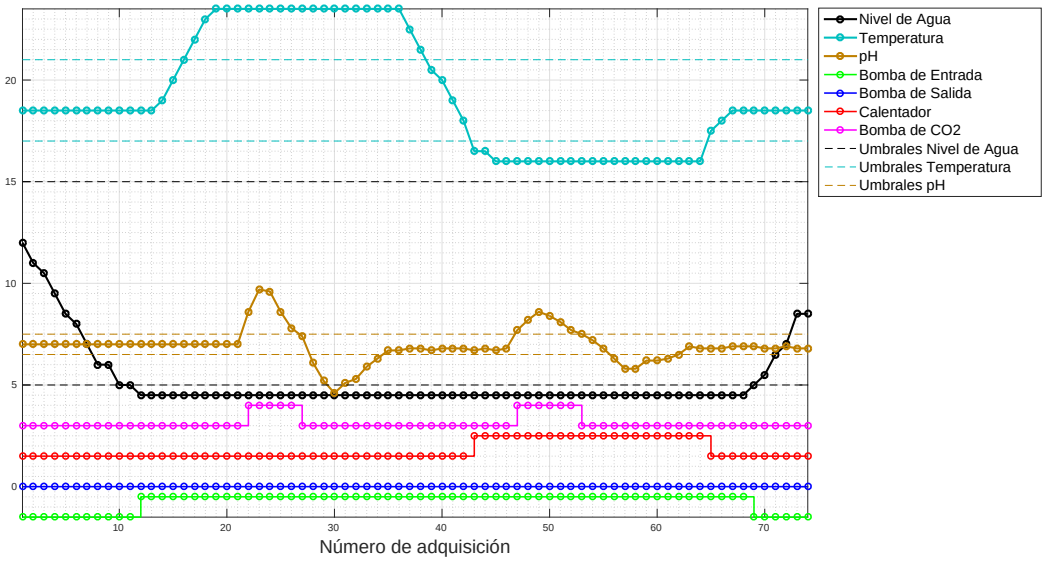

FIGURA 4.8: Tres alarmas: nivel de agua bajo y otras.

Capítulo 5

Conclusiones

5.1. Conclusiones del trabajo realizado

En la presente memoria de tesis se ha documentado el Trabajo Final para la obtención del grado de Especialista en Sistemas Embebidos .

Se obtuvo un *firmware* que permite controlar un acuario en forma remota mediante una interfaz web embebida en la plataforma CIAA-NXP. Se aplicaron simplificaciones al modelo de acuario, principalmente debido a las restricciones de tiempo establecidas para la finalización del trabajo y a la ejecución de acciones de contingencia de riesgos identificados al comienzo de la planificación del trabajo, a saber:

- Riesgo: No contar con un subsidio para la adquisición de sensores y actuadores de acuario.
- Riesgo: No disponer de una CIAA-NXP a tiempo para el comienzo del proyecto.
- Contingencia: Simular sensores y actuadores del modelo de acuario.

Con las mencionadas simplificaciones se obtuvo una "planta piloto" que permitió avanzar en el desarrollo del *firmware* de control y cumplir con los requerimientos de tiempo de finalización.

Se pudo integrar el *stack* TCP/IP lwIP con el RTOS freeRTOS al *firmware* de control sobre la base de un *port* desarrollado por NXP, fabricante del microcontrolador LPC4337.

Durante el desarrollo de este Trabajo Final se aplicaron conocimientos adquiridos a lo largo de la Carrera de Especialización en Sistemas Embebidos. Si bien el conjunto total de asignaturas cursadas aportaron conocimientos necesarios para la práctica profesional en el área de los Sistemas Embebidos, se quiere dejar constancia en particular, de las asignaturas con mayor relevancia para el trabajo presentado.

- **Arquitectura de microprocesadores**. Resultó necesario tener conocimientos sobre la Arquitectura ARM Cortex M para la programación de la plataforma CIAA-NXP y el uso de los periféricos.

- **Programación de microprocesadores**. Se utilizaron buenas prácticas de programación de Lenguaje C para microcontroladores y periféricos, aprendidas en esta asignatura. Se empleó un formato de código consistente: comentarios en las declaraciones de funciones y partes importantes del código, constantes en mayúsculas, *camelCase* para poner nombres significativos a funciones y variables. Se utilizaron APIs[1] para abstraer distintas capas del código. Se obtuvo un código más legible y reutilizable.
- **Ingeniería de Software en Sistemas Embebidos**. Se utilizaron técnicas provenientes de la Ingeniería de Software. Siempre que fue posible se utilizó un método sistemático e iterativo de desarrollo pensando en el ciclo de vida del *software*. Se hizo uso extensivo de Git como sistema de control de versiones del software. Asimismo, se aplicaron metodologías de ensayo de *software* aprendidas en esta asignatura.
- **Gestión de Proyectos en Ingeniería**. Resultó de gran utilidad elaborar un Plan de Proyecto para organizar el Trabajo Final. Parte del material elaborado en esta asignatura como la planificación y el desglose de tareas se pueden encontrar en la presente memoria, en el capítulo 2.
- **Sistemas Operativos de Tiempo Real (I y II)**. Se aplicaron los conocimientos adquiridos sobre freeRTOS respecto a tareas, cambios de contexto y semáforos entre otros. Asimismo, se utilizaron herramientas de desarrollo propias de freeRTOS para medir el uso de las pilas de las tareas creadas y optimizar el uso de la memoria.
- **Protocolos de Comunicación**. Se aplicaron ampliamente los conocimientos obtenidos en el área de Ethernet. En particular, resultó útil el material sobre el *stack* TCP/IP lwIP.
- **Diseño de Sistemas Críticos**. Siempre que fue posible se utilizaron técnicas de programación defensiva para que el comportamiento del *firmware* resulte lo más predecible posible. Se buscó obtener un sistema de control que no falle en primer lugar, y que si lo hace, falle en forma segura evitando daños tanto a los seres vivos dentro y fuera del acuario como a la propiedad.

Asimismo, durante el desarrollo del trabajo final se adquirieron conocimientos en las áreas de:

- Programación en lenguaje HTML5.
- Programación en lenguaje JavaScript.
- Utilización de *linker scripts* para el entorno de desarrollo LPCXpresso.

Por lo tanto, se concluye que los objetivos planteados al comienzo del trabajo han sido alcanzados satisfactoriamente, habiéndose cumplido con los criterios de aceptación del sistema final y además, se han obtenido conocimientos valiosos para la formación profesional del autor.

[1] *Application Program Interface*

5.2. Trabajo futuro

Se considera oportuno identificar las líneas de trabajo futuro para dar continuidad al esfuerzo invertido. Se listan a continuación las principales.

- Trabajar con una planta piloto real, con sensores y actuadores de acuario. Caracterizar la dinámica del entorno real y ajustar la lógica de control si fuera necesario.
- Escalar el *firmware* desarrollado para obtener un *framework* que posibilite cambiar el dominio de aplicación a otras áreas de control donde el modelo de "medir, alertar y actuar" aplique.
- Permitir la introducción de nuevos sensores y actuadores debidamente parametrizados mediante la interfaz de usuario.
- Mejorar la portabilidad de la interfaz web utilizando técnicas de RWD (*Responsive Web Design*) para que permita su correcta visualización en distintos dispositivos, navegadores y resoluciones de pantalla.

Bibliografía

[1] Mike Belshe, Roberto Peon y Martin Thomson. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC 7540. Nov. de 2015. DOI: 10.17487/rfc7540. URL: https://rfc-editor.org/rfc/rfc7540.txt.

[2] IEEE. *IEEE Citation Reference*. 1.ª ed. IEEE Publications, 2016. URL: http://www.ieee.org/documents/ieeecitationref.pdf (visitado 26-09-2016).

[3] Jeffrey Mogul y col. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. Mar. de 2013. DOI: 10.17487/rfc2616. URL: https://rfc-editor.org/rfc/rfc2616.txt.

[4] NXP Semiconductors. *LPCOPEN v2.16 Drivers, Middleware and Examples*. Disponible: 2016-06-25. URL: http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/lpc-cortex-m-mcus/software-tools/lpcopen-libraries-and-examples:LPC-OPEN-LIBRARIES.

[5] Proyecto CIAA. *CIAA Firmware Project.* Disponible: 2016-06-25. URL: https://github.com/ciaa/Firmware.

[6] Proyecto CIAA. *Computadora Industrial Abierta Argentina*. Disponible: 2016-06-25. 2014. URL: http://proyecto-ciaa.com.ar/devwiki/doku.php?id=start.

[7] Pablo Ridolfi. *Embedded software development workspace for microcontrollers.* Disponible: 2016-06-25. URL: https://github.com/pridolfi/workspace.

[8] David Robinson. *The Common Gateway Interface (CGI) Version 1.1*. RFC 3875. Mar. de 2013. DOI: 10.17487/rfc3875. URL: https://rfc-editor.org/rfc/rfc3875.txt.

[9] Savannah. *lwIP - A Lightweight TCP/IP stack*. Disponible: 2016-06-25. URL: http://git.savannah.gnu.org/cgit/lwip/lwip-contrib.git.

[10] Savannah. *lwIP - Wiki - Sample Web Server*. Disponible: 2016-06-25. URL: http://lwip.wikia.com/wiki/Sample_Web_Server.

[11] H. Zimmermann. «OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection». En: *IEEE Transactions on Communications* 28.4 (abr. de 1980), págs. 425-432. ISSN: 0090-6778. DOI: 10.1109/TCOM.1980.1094702.