UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

CIAABOT: Robótica educativa abierta sobre EDU-CIAA

Autor:
Ing. Leandro Lanzieri Rodríguez

Director:
Ing. Eric Pernía

Jurados:
Dr. Ing. Pablo Gómez (FIUBA)
Esp. Ing. Ernesto Gigliotti (UTN FRA)
Esp. Ing. Patricio Bos (FIUBA)

Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires, entre febrero de 2017 y diciembre de 2017.

Resumen

En la presente memoria se describe el desarrollo de CIAABOT, una plataforma libre de robótica educativa que funciona sobre la EDU-CIAA-NXP. El objetivo es facilitar la enseñanza de materias básicas y programación en escuelas, utilizando la robótica como herramienta. Para esto se desarrolló una interfaz de programación gráfica, para reducir la dificultad de una sintaxis específica.

Para cumplir con los objetivos planteados se aplicaron técnicas de programación como división de responsabilidades por capas, modularización y servicios. También se utilizaron herramientas de desarrollo, entre ellas control de versiones, debug y testing. Se aplicaron a su vez conocimientos adquiridos en la especialización de diseño de PCB y gestión de proyectos.

Agradecimientos

En primer lugar agradezco a mis padres Silvano y Nelly, por apoyarme en todos mis objetivos y hacerme pensar.

A mis hermanas Camila y Sol, por estar siempre conmigo.

A Deborah, por su paciencia y apoyo en todas mis locuras, aunque implique menos tiempo juntos.

A Daniel Acerbi, director del Laboratorio Abierto de la UTN Facultad Regional Avellaneda, por su confianza y la beca que me permitió cursar esta especialización.

Finalmente, a mi director Eric Pernía, por su apoyo y entusiasmo constantes que hicieron posible este proyecto.

A todos ellos, muchas gracias.

Índice general

Índice de figuras

Índice de Tablas

A mi abuela Neli.

Capítulo 1

Introducción General

Aquí se exponen las problemáticas observadas que originaron el desarrollo del presente trabajo, junto con una breve descripción de la plataforma planteada para solucionarlas.

1.1. Motivación

Se observaron procesos de enseñanza, tanto en escuelas como en cursos de programación de sistemas embebidos. Se detectó que un gran obstáculo común entre alumnos es aprender en conjunto la lógica para resolución de problemas utilizando algoritmos, y la sintaxis de un lenguaje específico.

1.1.1. Robótica en la educación

En el último tiempo se ha extendido cada vez más el uso de la robótica educativa como una herramienta particularmente útil en ámbitos escolares. Es utilizada como un sistema de enseñanza multidisciplinaria, que potencia el desarrollo de habilidades en las áreas de ciencias, tecnología, ingeniería y matemáticas. Correctamente estructurada, la robótica educativa permite incentivar el trabajo en equipo, el liderazgo, el emprendimiento y el aprendizaje a partir los errores.

La robótica en las escuelas se presenta entonces como una opción tecnológica e innovadora de afrontar la problemática de capturar el interés de los alumnos en los temas propuestos en la currícula. Esto se debe a la necesidad de involucrarse y trabajar en conjunto con sus pares para la resolución de problemas donde deben aplicar los conocimientos de las diferentes asignaturas. Al realizarse de manera entretenida los contenidos se fijan de manera más simple y natural.

Sin embargo, a la hora de aplicar estas técnicas de enseñanza se presenta una problemática. En el ambiente de las escuelas primarias el público son estudiantes jóvenes en proceso de formación, que en general no poseen conocimientos técnicos en detalle como electrónica o manejo de lenguajes de programación. Esto se traduce en una limitación y dificultad a la hora de acercarse a este tipo de tecnologías, aunque el objetivo final no sea el aprendizaje de la robótica en sí misma sino utilizarla *como herramienta*.

Las opciones actuales se presentan como plataformas que son, en su mayoría, diseños privativos, cerrados, con pocas posibilidades de modificación y en general

desarrollos extranjeros importados a nuestro país. Se observa por lo tanto la necesidad de que la solución alternativa debe ser abierta y desarrollada localmente.

1.1.2. Introducción a la programación de sistemas embebidos

Los primeros pasos para la inserción en el ámbito de la programación de sistemas embebidos pueden ser difíciles para algunas personas. Esto se debe, en parte, al hecho de tener que trabajar sobre plataformas diferentes a una PC y utilizando lenguajes de programación de bajo nivel (generalmente C o assembler).

Una de las mayores dificultades a la hora de iniciarse en la programación, de sistemas embebidos o en general, es la sintaxis específica del lenguaje que debe ser utilizado. Esto se evidencia a la hora de expresar en código la idea que se tiene, es decir la lógica del programa que se está desarrollando.

Una opción que presenta una curva de aprendizaje más plana es abstraerse, por lo menos en una primera etapa, del lenguaje que se utilizará para programar más adelante. De esta manera el alumno puede concentrarse en definir la lógica de su solución propuesta al problema planteado, sin preocuparse en cómo se expresaría en código (dicha tarea queda para etapas posteriores del proceso de aprendizaje).

1.2. CIAABOT: Lineamientos principales

CIAABOT se presenta como una plataforma abierta orientada a la robótica educativa, es decir utilizada como una herramienta para la educación, con el fin de suplir las necesidades identificadas en la sección 1.1. A la hora de darle forma a este proyecto se plantearon unos lineamientos general que debía cumplir, más allá de los alcances a nivel técnico. A continuación se dará una breve descripción y se justificará cada uno de ellos.

1.2.1. Software y Hardware libres

La *cultura libre* es una corriente o movimiento que, en contraposición a las medidas restrictivas de los derechos de autor, promueve la libertad para distribuir y modificar trabajos.

Esta ideología aplicada al software se traduce en código que es libera bajo alguna de las licencias libres (por ejemplo BSD, GPL o MIT), lo que implica ciertas libertades a la hora de utilizarlo, modificarlo y distribuirlo. En general, aunque depende de la licencia elegida, se libera el código fuente de manera gratuita (aunque esto último no es condición obligatoria) y los usuarios pueden verlo o modificarlo.

Cuando se trata de hardware, la idea de un diseño libre se traduce en la liberación bajo alguna licencia de especificaciones, diagramas esquemáticos, diseños de circuitos impresos, memorias de cálculo y cualquier otro documento accesorio de interés para el funcionamiento del sistema. Que el hardware sea libre no implica necesariamente que se pueda adquirir físicamente de manera gratuita, ya que el la fabricación y armado suponen un costo.

La ventaja principal por la que se optó que el proyecto sea libre, tanto en hardware como en software, es que los interesados pueden descargar esquemáticos o código fuente respectivamente y armar sus circuitos o compilar sus programas, junto con cualquier modificación que crean conveniente. Particularmente el software de CIAABOT se liberó con la licencia GNU GPLv3 [3]. Esto implica, entre otras cosas, que cualquier modificación o utilización del mismo debe liberarse bajo la misma licencia.

1.2.2. Orientación a la enseñanza

Como se describió en la sección 1.1 la robótica como herramienta para promover el aprendizaje de materias básicas, lógica y programación está ganando terreno. Es por esto que CIAABOT se desarrolló con el objetivo de ser utilizado para introducirse a la programación de sistemas embebidos de manera fácil y didáctica. Permitiendo de esta manera a docentes, alumnos de escuelas o entusiastas que deseen iniciarse en la programación de embebidos, un posibilidad simplificada sin la complicación de encontrarse con una complicada sintaxis.

Teniendo en mente este lineamiento, se analizó el armado del DSL (*Domain-Specific Language*, Lenguaje de Dominio Específico) para que fuera natural su uso, como si es estuviera escribiendo el algoritmo con palabras.

1.2.3. Amplia documentación

La documentación a la hora de usar una plataforma, ya sea de software o hardware es esencial. Es una información fehaciente que tienen los usuarios para poder aprender a utilizarla o resolver los posibles problemas que se presenten. CIAABOT apuesta para ampliar la red de personas que la utilizan, a la formación de una comunidad de usuarios que aporte experiencias, conocimiento e incluso mejoras a la plataforma. Las plataformas abiertas más conocidas lo son porque son fáciles de utilizar y se pueden conseguir resultados visibles de manera inmediata. Todo esto sería difícil de conseguir si la documentación fuera escasa o inexistente, como ocurre con algunos proyectos.

Al seguir esta idea de poner al usuario en el centro, CIAABOT posee documentación de instalación, ejemplos y preguntas frecuentes que crece con cada versión [2]. Además de basarse en otros proyectos que presentan a su vez extensa documentación y ejemplos de uso (CIAA Firmware v2 y sAPI).

1.2.4. Escalabilidad

CIAABOT está basada en la plataforma de CIAA [**CIAA**](Computadora Industrial Abierta Argentina). Actualmente funciona sobre la placa EDU-CIAA-NXP, una versión educativa y reducida de la CIAA-NXP pensada para la educación y la formación en sistemas embebidos. El proyecto CIAA presenta una gran dinámica, con la liberación de herramientas nuevas, actualizaciones y hasta placas nuevas.

Un ejemplo de esto es la reciente placa CIAA Z3R0 [1]. Esta placa utiliza el microcontrolador EFM32HG de Silicon Labs, y tiene como objetivo un muy bajo consumo de energía, simplicidad y pequeño tamaño. Está pensada para aplicaciones en robótica e IoT (*Internet of Things*, Internet de la cosas).

Debido a esto, se pensó CIAABOT como una plataforma que pueda adaptarse a estos cambios. Internamente las placas soportadas del ecosistema CIAA tienen una definición de capacidades y pines disponibles, donde se pueden adicionar placas nuevas a medida que aparezcan.

Además, como se explicar en la sección **??**, CIAABOT pretende tener varios modelos de robots que funcionen con la plataforma, y que puedan basarse en diferentes placas del proyecto CIAA.

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 8.

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.

1.3. Alcance y Objetivos planteados

El objetivo principal fue desarrollar el entorno de desarrollo y programación para CIAABOT, junto con firmware específico que se requiera, basándose en bibliotecas existentes para la CIAA (por ejemplo sAPI). Además, la implementación y uso de lo anterior en un robot o maqueta existente adaptado para utilizar la EDU-CIAA-NXP. Esto se hace para demostrar las funcionalidades que se proveen. Para esto también se consideró el armado se todo el hardware adicional necesario.

Se busca que el proyecto sea utilizado para la enseñanza de programación de sistemas embebidos, apuntando principalmente a la robótica.

En el contexto del trabajo de especialización, se restringió el alcance por una cuestión de tiempo. Se resolvió que el desarrollo incluiría la aplicación de escritorio, que permitiera la programación gráfica de los algoritmos, utilizando bloques. Además debería traducir estos bloques a lenguaje C. Por último, se planteó que se utilizaría el protocolo Firmata para un monitoreo en tiempo real del estado de la placa cuando se desarrolla conectado a la PC.

No se incluyó en el desarrollo de un modelo específico de CIAABOT.

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*[1] como **`.png`**, **`.jpg`** o en formato vectoriales[2] 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.

[1] https://en.wikipedia.org/wiki/Raster_graphics

[2] https://en.wikipedia.org/wiki/Vector_graphics

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.

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 – este es un archivo auxiliar generado por BibTeX, si se borra BibTeX simplemente lo regenera cuando se compila el archivo principal **`memoria.tex`**. Mientras que el archivo **`.bib`** contiene todas las referencias que hay, este archivo **`.bbl`** contine sólo las referencias que han sido citadas y se utiliza para la construcción de la bibiografía.

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. Una vez instalado el programa y abierto el archivo memoria.tex se debería ver una pantalla similar a la figura 1.1.

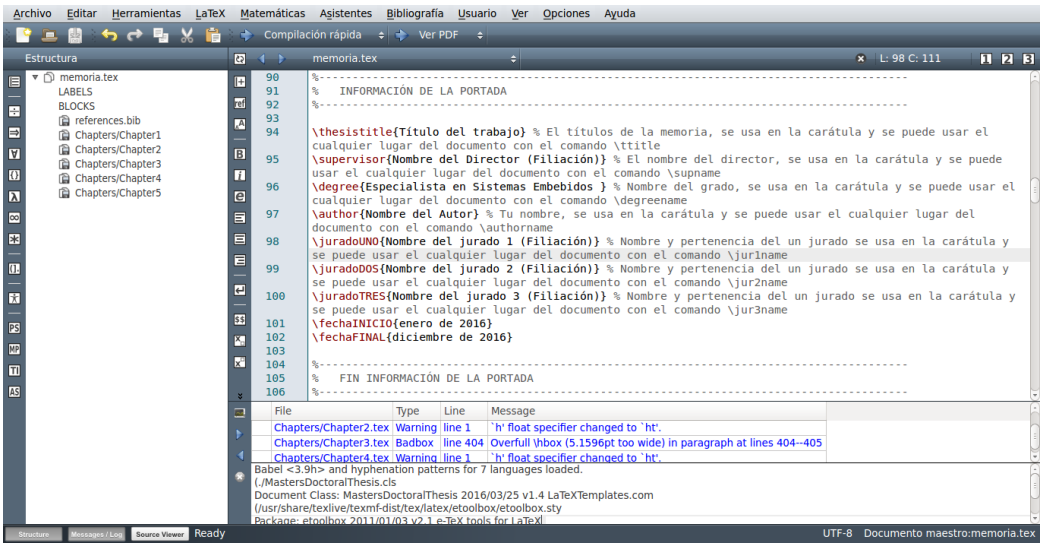

FIGURA 1.1: Entorno de trabajo del texMaker.

Notar que existe una vista llamada Estructura a la izquierda de la interface que nos permite abrir desde dentro del programa los archivos individuales de los capítulos. A la derecha se encuentra una vista con el archivo propiamente dicho para su edición. Hacia la parte inferior se encuentra una vista del log con información de los resultados de la compilación. En esta última vista pueden aparecen advertencias o *warning* que normalmente pueden ser ignorados y también los errores que se indican en color rojo.

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.

En el menú herramientas se encuentran las opciones de compilación. Para producir un archivo PDF a partir de un archivo .tex se debe ejecutar PDFLaTeX (el shortcut es F6). Para incorporar nueva bibliografía se debe utilizar la opción BibTeX del mismo menú herramientas (el shortcut es F11).

Notar que para actualizar las tablas de contenidos se debe ejecutar PDFLaTeX dos veces. Esto se debe a que es necesario actualizar algunos archivos auxiliares antes de obtener el resultado final. En forma similar, para actualizar las referecias se debe ejecutar primero PDFLaTeX, después BibTeX y finalmente PDFLaTeX dos veces por idénticos motivos.

1.5. Personalizando la plantilla 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.

1.6. El código del archivo `memoria.tex` explicado

El archivo **`memoria.tex`** contiene la estructura de la memoria y se encuentra densamente comentado para explicar qué páginas, secciones y elementos de formato el código LaTeX está creando en cada línea. Cada elemento de mayor jerarquía del documento está dividido en bloques con nombres en mayúsculas para que resulte evidente qué es lo que hace esa porción de código en particular. Inicialmente puede parecer que hay mucho código LaTeX, pero es principalmente código para dar formato a la memoria y al estar ya definido, no requiere intervención del usuario de la plantilla.

Se debe comenzar por chequear que la información en la portada es correcta.

Luego viene el resumen que contiene una versión abreviada de su trabajo. Se debería poder utilizar como un documento independiente para describir el contenido de su trabajo.

A continuación se encuentra la sección opcional de agradecimientos.

El índice de contenidos, las listas de figura de tablas se generan en forma automática y no requieren intervención ni edición manual por parte del usuario de la plantilla.

La siguiente página es opcional y puede contener una dedicatorio de una línea, en caso de que usted quiere dedicarle el trabajo a alguien.

Finalmente, se encuentra el bloque donde se incluyen los capítulos y los apéndices. Por defecto se incluyen los 5 capítulos propuestos que se encuentran en la carpeta /Chapters. Cada capítulo se debe escribir en un archivo .tex separado y se debe poner en la carpeta *Chapters* con el nombre **`Chapter1`**, **`Chapter2`**, etc... El código para incluir capítulos desde archivos externos se muestra a continuación.

```
\include{Chapters/Chapter1}
\include{Chapters/Chapter2}
\include{Chapters/Chapter3}
\include{Chapters/Chapter4}
\include{Chapters/Chapter5}
```

Los apéndices también deben ir en archivos .tex separados y se deben ubicar dentro de la carpeta *Appendices*. Los apéndices vienen comentados con el caracter `%` por defecto y para incluirlos se debe eliminar dicho caracter.

Luego del preambulo, los capítulos y los apéndices, finalmente viene la bibliografía. El estilo bibliográfico (llamado *`authoryear`*) es utilizado por LaTeX para generar las referencias y es un estilo con todas las características necesarias para su composición. No se debe subestimar lo agradecido que estarán sus lectores al encontrar que las referencias se encuentran a un clic de distancia. Por supuesto, esto depende de que usted haya puesto la url correspondiente en el archivo BibTex en primer lugar.

Capítulo 2

Introducción Específica

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.

2.1. Estilo y convenciones

2.1.1. Uso de mayúscula inicial para los título de secciones

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 2.1.2 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.

2.1.2. 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 [**IEEE:citation**]. Por ejemplo, "el presente trabajo se basa en la plataforma EDU-CIAA-NXP, la cual se describe en detalle en [**CIAA**]".

2.1.3. Figuras

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 tomada de la página oficial 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.

2.1.4. Tablas

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. La referencia se logra utilizando el comando `\ref{<label>}` donde label debe estar definida dentro del entorno de la tabla.

```
\begin{table}[h]
\centering
\caption[caption corto]{caption largo más descriptivo}
\begin{tabular}{l c c}
\toprule
\textbf{Especie}       & \textbf{Tamaño}  & \textbf{Valor aprox.}\\
\midrule
Amphiprion Ocellaris   & 10 cm  & \$ 6.000 \\
Hepatus Blue Tang      & 15 cm  & \$ 7.000 \\
Zebrasoma Xanthurus    & 12 cm  & \$ 6.800 \\
\bottomrule
\hline
\end{tabular}
\label{tab:peces}
\end{table}
```

TABLA 2.1: caption largo más descriptivo

Especie	**Tamaño**	**Valor aprox.**
Amphiprion Ocellaris	10 cm	$ 6.000
Hepatus Blue Tang	15 cm	$ 7.000
Zebrasoma Xanthurus	12 cm	$ 6.800

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.

2.1.5. 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.

Para generar la ecuación 2.1 se utilizó el siguiente código:

```
\begin{equation}
\label{eq:metric}
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)
\end{equation}
```

Y para la ecuación 2.2:

```
\begin{equation}
\label{eq:schrodinger}
\frac{\hbar^2}{2m}\nabla^2\Psi + V(\mathbf{r})\Psi =
-i\hbar \frac{\partial\Psi}{\partial t}
\end{equation}
```

Capítulo 3

Diseño e Implementación

3.1. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]

las líneas de código irían aquí...

\end{lstlisting}
```

A modo de ejemplo:

```
#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.

Capítulo 4

Ensayos y Resultados

4.1. Pruebas funcionales del hardware

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

Capítulo 5

Conclusiones

5.1. Conclusiones generales

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.

5.2. Próximos pasos

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

Bibliografía

[1] *CIAA Z3R0*. 2017. URL: TODO:AgregarlinkaCIAAZ3R0.
[2] *Documentación de CIAABOT*. 2017. URL: http://leandrolanzieri.github.io/ciaabot-ide/documentacion.
[3] Free Software Foundation. *GNU General Public Licence v3*. 2007. URL: https://www.gnu.org/licenses/gpl-3.0.en.html.