UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

CIAABOT: Robótica educativa abierta sobre 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 CIAA. 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 [5]. 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 [4]. 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 [6](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 [2]. 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 2.1, CIAABOT pretende tener varios modelos de robots que funcionen con la plataforma, y que puedan basarse en diferentes placas del proyecto CIAA.

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.

Capítulo 2

Introducción Específica

En este capítulo se presenta CIAABOT con más detalle, incluyendo especificaciones técnicas sobre la plataforma utilizada. Se establecen los requerimientos planteados y la planificación para el desarrollo del trabajo.

2.1. CIAABOT: Partes componentes

Se planteó que para lograr los objetivos propuestos, la plataforma debería estar formada por tres partes fundamentales, que funcionarían complementadas para armar un ecosistema CIAABOT, como está esquematizado en la figura 2.1. A continuación se describirá cada uno de ellos.

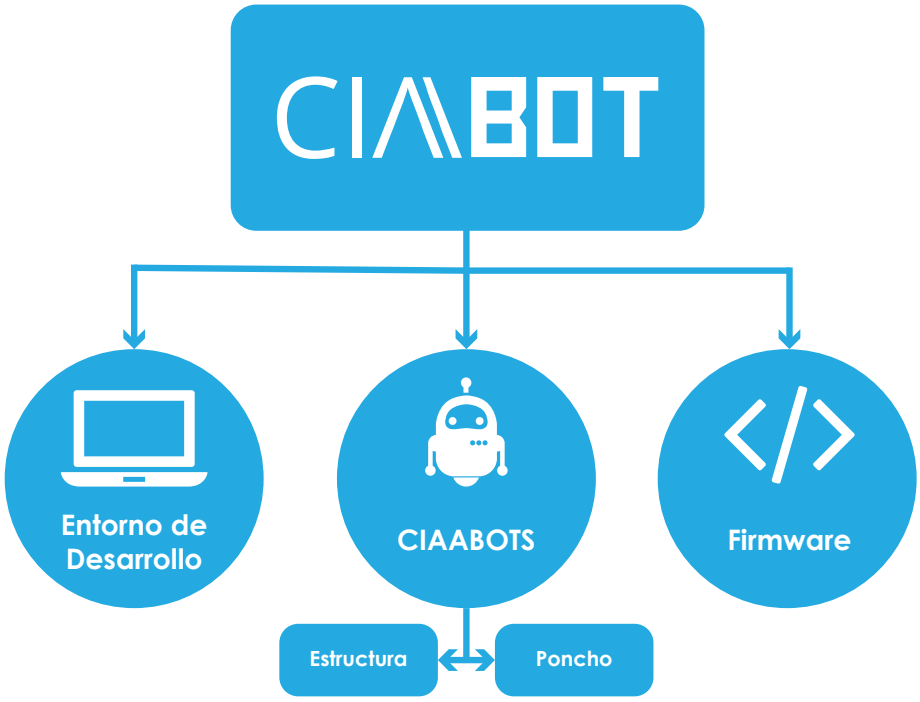

FIGURA 2.1: Diagrama de las partes que componen CIAABOT.

2.1.1. Entorno de desarrollo integrado

Un IDE (*Integrated Development Environment*, Entorno de Desarrollo Integrado) es un software que incluye varias herramientas para asistir a los desarrolladores con su trabajo. Generalmente incluye algún editor para el código, herramientas de construcción o compilación automáticas, programación de la plataforma y posible depuración.

El CIAABOT-IDE es el centro del trabajo de los usuarios. Aquí plasman sus ideas y soluciones a problemas planteados. Esto lo hacen por medio de un editor de código en el que utilizan bloques predefinidos y no texto, como se observa en la figura 2.2. Arrastrando e interconectando los mismos logran definir la lógica para su programa.

Una opción interesante para remarcar es la salida de código en C. Cada vez que se modifica el diagrama de bloques, el código en C producido se modifica en tiempo real. Esto es muy útil para gente que pretende aprender a programar en C, ya que ve de manera directa cómo afecta cada bloque a las líneas de código y puede apreciar las equivalencias.

La plataforma también permite compilar y luego descargar el código generado en la placa solamente conectándola por USB.

Una vez que se finalizó el trabajo se puede guardar el proyecto en un archivo *.cbp* que contiene la estructura de bloques en la que se estuvo trabajando, junto con las opciones adicionales del proyecto. Este archivo puede ser compartido y abierto en otra PC para ser reutilizado o modificado y guardado nuevamente.

Otra de las opciones que fue publicada en la última versión de CIAABOT-IDE (v0.0.8) es la de seleccionar los directorios de búsqueda para el toolchain. Las aplicaciones necesarias para todos los sistemas operativos son dos: El compilador arm-none-eabi-gcc y el debugger openOCD. Adicionalmente, en Windows se requieren otros ejecutables como make, para poder compilar y descargar el programa.

2.1.2. CIAABOTS

Los robots que funcionen bajo esta plataforma serán denominados *CIAABOTS*. Siguiendo la idea original de que CIAABOT sea utilizada en escuelas o por entusiastas, se planteó que estos robots estén diseñados estructuralmente para poder ser impresos con impresoras 3D.

De esta manera es posible enviar los archivos para imprimirlos en cualquier escuela que cuente con este sistema de impresión para replicarlo. Sólo sería necesario tener la placa CIAA correspondiente al modelo, armar el poncho (de diseño abierto) y adquirir en algún lugar de conveniencia los actuadores o sensores particulares del CIAABOT (motores DC, sensores IR, ultrasonido, etc.).

El diseño del poncho para el primer modelo de CIAABOT, el G1 se puede encontrar en github [3], realizado con el suite KiCad como todos los diseños relacionados al proyecto CIAA.

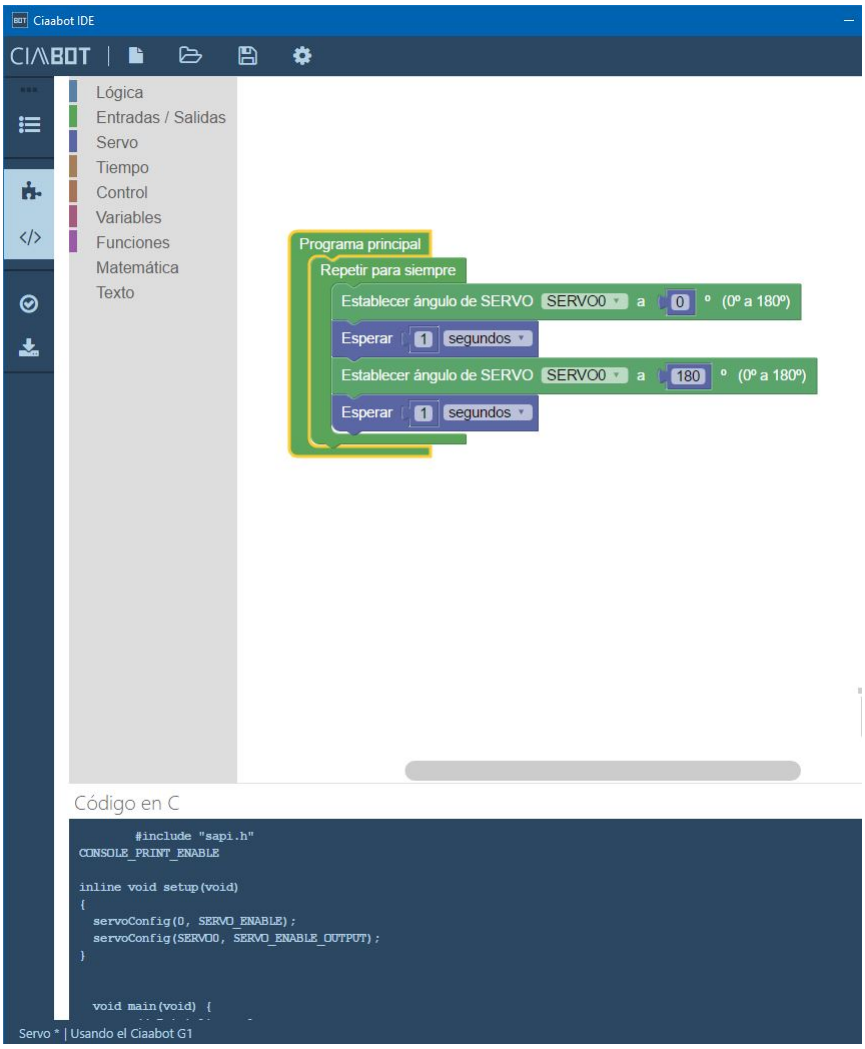

FIGURA 2.2: Editor gráfico de CIAABOT-IDE.

2.1.3. Firmware

CIAABOT utiliza como base para su firmware dos proyectos principales del ecosistema CIAA: Firmware v2 [1] y sAPI [7]. En conjunto proveen una forma simplificada de programar las placas CIAA.

La sAPI es una HAL (*Hardware Abstraction Layer*, Capa de Abstracción de Hardware) para los microcontroladores, que permite acceder de manera simple a varios de los diferentes periféricos.

Juntando varias funciones de la sAPI es posible armar funciones más complejas de mayor nivel para manejar características más específicas de cada modelo de CIAABOT.

Se prevé que con el avance del desarrollo de la plataforma cada modelo tendrá bloques específicos que estarán disponibles en el editor, cuando se lo seleccione a la hora de armar el proyecto en el IDE. Así, podrían armarse bloques que realicen por ejemplo una lectura de un sensor, que implicaría el manejo de pines y tiempos de espera.

2.2. Plataforma utilizada

Como base para el proyecto se optó por utilizar CIAA. Es un proyecto centrado en el trabajo colaborativo que busca la innovación para crear, diseñar y desarrollar soluciones electrónicas en la industria. Dentro de sus objetivos se encuentran:

- Impulsar el desarrollo tecnológico nacional, a partir de sumar valor agregado al trabajo y a los productos y servicios, mediante el uso de sistemas electrónicos, en el marco de la vinculación de las instituciones educativas y el sistema científico-tecnológico con la industria.
- Darle visibilidad positiva a la electrónica argentina.
- Generar cambios estructurales en la forma en la que se desarrollan y utilizan en nuestro país los conocimientos en el ámbito de la electrónica y de las instituciones y empresas que hacen uso de ella. [8]

Se utiliza esta plataforma por compartir la misma idea de ser un proyecto libre, poseer una creciente comunidad e incentivar la industria nacional. Se consideró que es un proyecto suficientemente maduro y con actualizaciones y desarrollos constantes como para ser utilizado de base.

2.2.1. EDU-CIAA-NXP

La EDU-CIAA-NXP es la versión educativa y de bajo costo de la CIAA-NXP. Utiliza el microcontrolador LPC4337 de NXP, un doble núcleo ARM Cortex-M4F y Cortex M0. Como se observa en la figura 2.3, la placa sólo posee 4 leds y 4 pulsadores de hardware adicional. Para ampliar sus capacidades se utilizan los conectores laterales, para adicionar alguna placa extensora llamada 'Poncho'.

FIGURA 2.3: Placa EDU-CIAA-NXP

2.3. Estilo y convenciones

2.3.1. 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 2.3 se presenta lo que sea", o "En la subsección 2.3.2 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.

2.3.2. 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 [6]".

2.3.3. 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 2.4”.

FIGURA 2.4: 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 2.5.

FIGURA 2.5: Imagen tomada de la página oficial del procesador[1].

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.

Nunca colocar una figura en el documento antes de hacer la primera referencia a ella, como se ilustra con la figura 2.6, porque sino el lector no comprenderá por qué de pronto aparece la figura en el documento, lo que distraerá su atención.

2.3.4. 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.

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

FIGURA 2.6: El lector no sabe por qué de pronto aparece esta figura.

```
\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.

2.3.5. 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. Estructura del Software

sAPI

- Interrupciones.
- SysTick.
- GPIO.
- UART.
- ADC.
- DAC.
- I^2C.
- SPI.
- RTC.
- SCT.
- Timers.

Además incluye funciones avanzadas de conversión de datos, impresión formateada por UART, manejo de displays 7 segmentos, teclados matriciales, servos y PWM entre otras.

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:

```
1 #define MAX_SENSOR_NUMBER 3
2 #define MAX_ALARM_NUMBER 6
3 #define MAX_ACTUATOR_NUMBER 6
4
```

```
5  uint32_t sensorValue[MAX_SENSOR_NUMBER];
6  FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
7  state_t alarmState[MAX_ALARM_NUMBER];           //ON or OFF
8  state_t actuatorState[MAX_ACTUATOR_NUMBER];     //ON or OFF
9
10 void vControl() {
11
12   initGlobalVariables();
13
14   period = 500 ms;
15
16   while(1) {
17
18     ticks = xTaskGetTickCount();
19
20     updateSensors();
21
22     updateAlarms();
23
24     controlActuators();
25
26     vTaskDelayUntil(&ticks, period);
27   }
28 }
```

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 Firmware v2*. TODO Agregar año de CIAA Firmware v2. URL: TODO:AgregarlinkaFirmwarev2.

[2] *CIAA Z3R0*. 2017. URL: TODO:AgregarlinkaCIAAZ3R0.

[3] *Diseño del poncho CIAABOT G1*. 2017. URL: https://github.com/leandrolanzieri/ciaabot-g1-poncho.

[4] *Documentación de CIAABOT*. 2017. URL: http://leandrolanzieri.github.io/ciaabot-ide/documentacion.

[5] Free Software Foundation. *GNU General Public Licence v3*. 2007. URL: https://www.gnu.org/licenses/gpl-3.0.en.html.

[6] *Proyecto CIAA*. URL: http://www.proyecto-ciaa.com.ar.

[7] *sAPI*. 2017. URL: https://github.com/epernia/sAPI.

[8] *Wiki del proyecto CIAA*. URL: http://www.proyecto-ciaa.com.ar.