UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

CIAABOT: Robótica educativa abierta basada en la 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

En este capítulo 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 [6]. 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 [5]. 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 (Computadora Industrial Abierta Argentina) [8]. 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 explica 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 hizo 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.

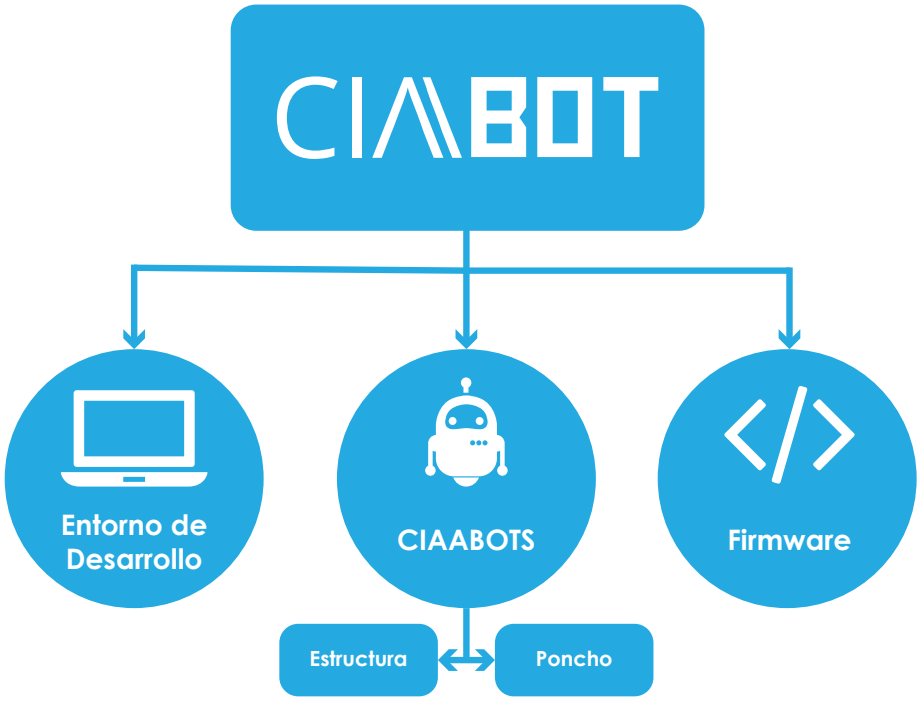

FIGURA 2.1: Diagrama de las partes que componen CIAABOT.

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 KiCad, como todos los diseños relacionados al proyecto CIAA.

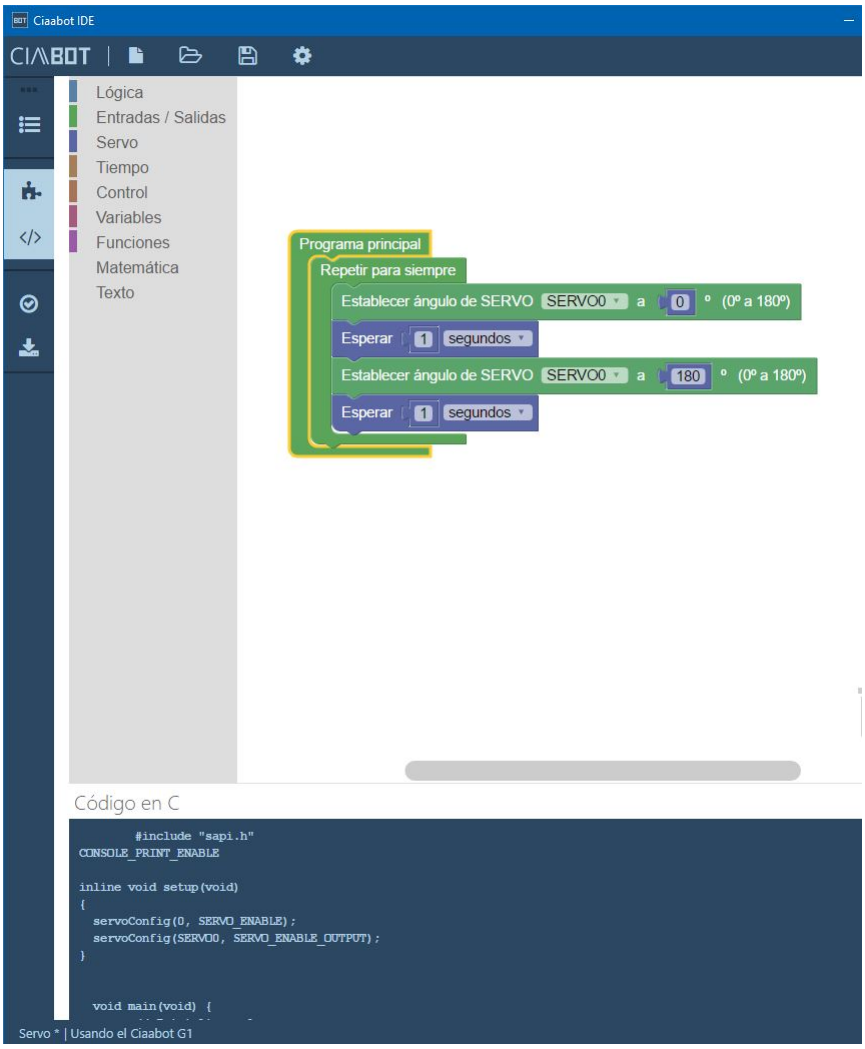

FIGURA 2.2: Editor gráfico de CIAABOT-IDE.

2.1.3. Firmware

CIAABOT utiliza como base para su firmware dos proyectos principales del ecosistema CIAA: Firmware v2 [1] y sAPI [9]. 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 [10]:

- 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.

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, al igual que la CIAA-NXP, el microcontrolador LPC4337 de NXP, un doble núcleo ARM Cortex-M4F y Cortex M0. Como se observa en la figura 2.3, la placa 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

Permite realizar programación y debugging, así como también acceder a una de sus UARTs, por medio de USB. Además posee otro puerto para que la placa funcione en modo OTG (*On The Go*).

2.3. Requerimientos

Se plantearon requerimientos que el proyecto debe cumplir a la hora de ser entregado. Se evaluaron sus posibilidades y se clasificaron en categorías.

2.3.1. Requerimientos del sistema

Se estableció que el IDE debe poder ejecutarse como mínimo en un entorno Linux o Windows. Por la orientación didáctica del proyecto su uso debe ser simple, amigable y didáctico. Además, la programación debe poder realizarse íntegramente de manera gráfica por medio de bloques, donde cada uno represente acciones simples.

El IDE permite un manejo de proyectos en forma de archivos, que posibilitan el guardado y distribución de los mismos. Dentro de las opciones del proyecto se puede seleccionar el modelo de CIAABOT utilizado. Actualmente la opción existe, pero hay un solo modelo disponible.

El algoritmo que se genera a partir de los bloques está disponible en lenguaje C, para poder compilar y programar desde la aplicación, por un puerto USB.

2.3.2. Componentes del Sistema

Como ya se mencionó en 2.2.1, se utilizará en principio la placa EDU-CIAA-NXP, y un poncho si es necesario para controlar la maqueta o robot implementado para la muestra. En el diseño de este poncho todos los periféricos deben funcionar independientes al resto.

2.3.3. Firmware

Para el desarrollo del software se implementó un sistema de control de versiones. Para la programación de los robots se utilizaron dos bibliotecas: En C la mencionada sAPI, y una de Firmata para un monitoreo del mismo por medio del puerto serie, con un cliente corriendo en la placa.

2.3.4. Procesos Finales

Se requiere un manual de uso e instalación, junto con ejemplos básicos y funcionales, que sean representativos de las características de la plataforma. Se implementó sobre una maqueta adaptada a la EDU-CIAA, y con ello se evaluaron los resultados del proyecto y su aplicación en ámbitos de enseñanza reales.

2.4. Planificación

2.4.1. Desglose en tareas

En primer lugar, para definir objetivos concretos, se planteó una serie de entregables para el proyecto:

- IDE en funcionamiento.
- Poncho CIAABOT G1 diseñado enteramente.
- Manual de uso, instalación y ejemplos funcionales.
- El presente informe final.

Siguiendo los lineamientos de las estrategias vistas en la materia Gestión de Proyectos, se procedió al diseño de una lista de desglose del trabajo en tareas. Se estimó un tiempo aproximado de 615 horas, distribuidas en grupos de tareas de la siguiente manera:

- Planificación del proyecto (40 hs.).
- Investigación preliminar (45 hs.).
- Selección de bibliotecas y plataformas (30 hs.).
- Desarrollo de la aplicación de escritorio (45 hs.).
- Desarrollo del editor gráfico (55 hs.).
- Programación por USB y monitoreo (60 hs.).
- Desarrollo del hardware (100 hs.).
- Desarrollo del firmware (70 hs.).
- Integración del sistema (50 hs.).
- Procesos finales (120 hs.).

2.4.2. Activity On-node

Si se toman las tareas antes descriptas y se evalúan sus dependencias, es decir las tareas que deben estar finalizadas antes de poder a realizarlas, se puede armar un diagrama denominado *Activity on-node*. En la figura 2.4.2 se pueden observar las tareas propuestas junto con el tiempo estimado en días para cada tarea. Todas las flechas entrantes a un nodo o tarea son las dependencias de la misma.

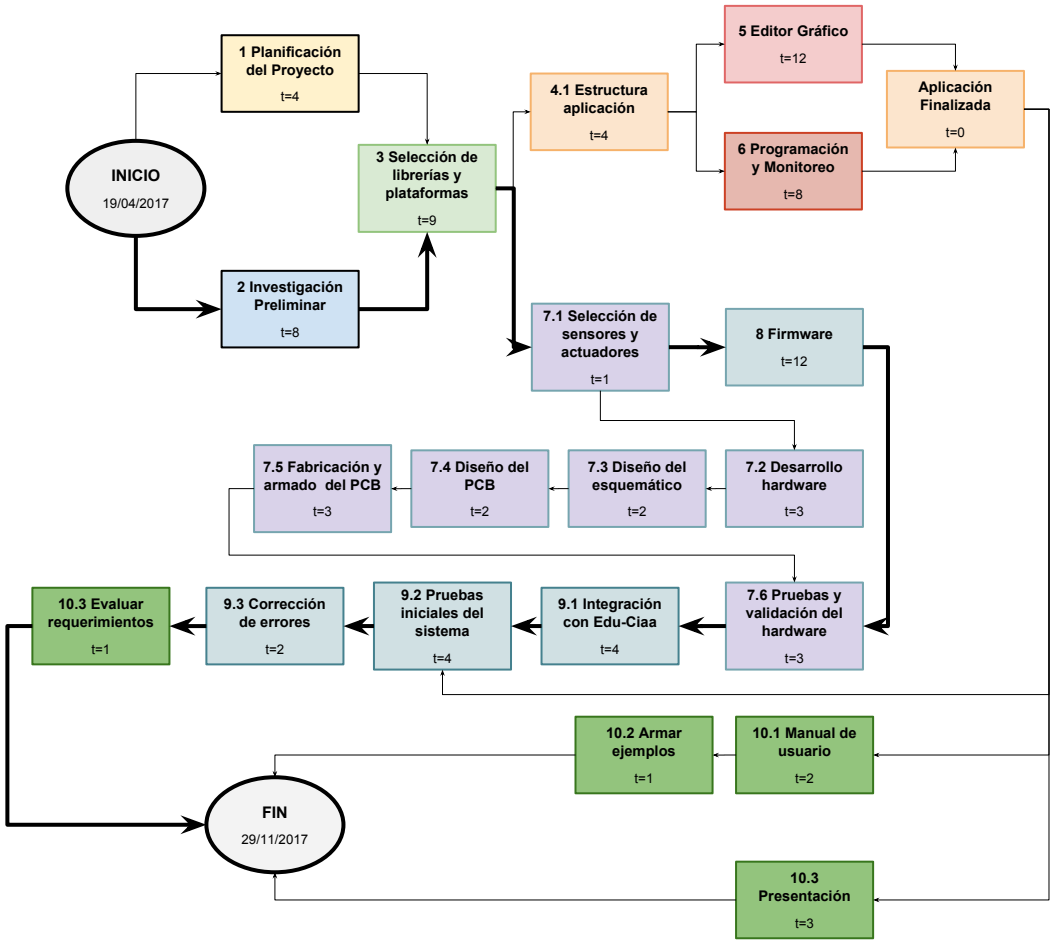

FIGURA 2.4: Diagrama *Activity on-node*

Los días están expresados en días laborales de aproximadamente 3 horas. Evaluando las duraciones y precedencias de las tareas, la sucesión que presente la mayor cantidad de horas es llamada *camino crítico*.

En la planificación de un proyecto es importante su identificación, ya que cualquier tarea que se encuentre sobre él tiene lo que se llama flotación cero. Esto significa que un incremento en su duración llevará indefectiblemente a una dilatación del plazo de finalización del proyecto. En el diagrama este camino está resaltado con flechas más gruesas.

2.4.3. Diagrama de Gantt

Si se establecen fechas concretas para los inicios de las tareas, teniendo en cuenta su duración y precedencias, se puede armar el llamado diagrama de Gantt. Esto da una referencia rápida de dónde se debería encontrar el desarrollo del proyecto según la planificación inicial en cada momento dado. En las figuras 2.5 y 2.6 se puede observar el diagrama para este proyecto.

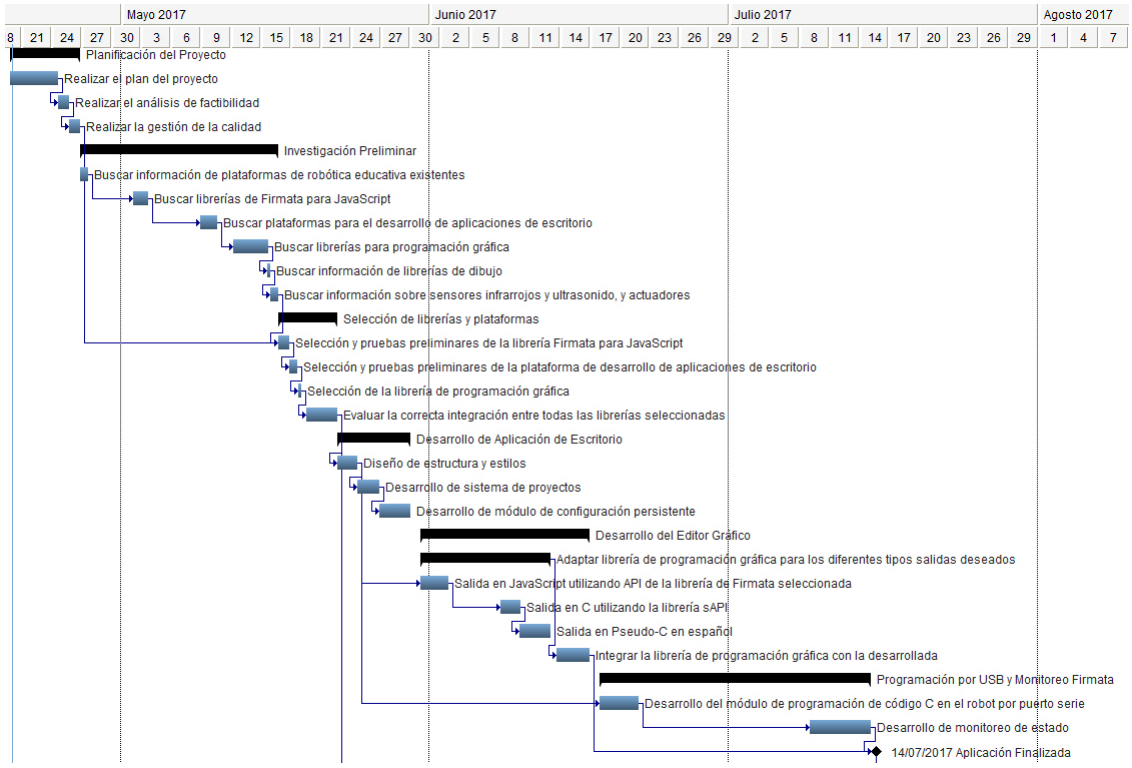

FIGURA 2.5: Diagrama de Gantt - Parte 1.

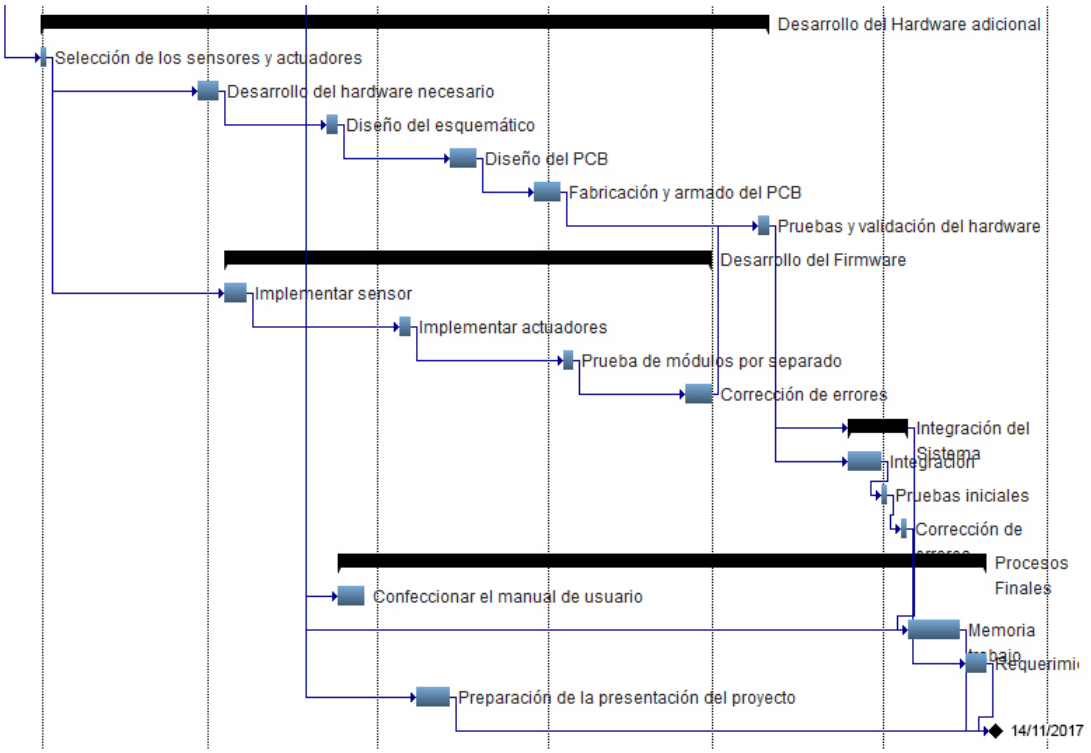

FIGURA 2.6: Diagrama de Gantt - Parte 2.

Capítulo 3

Diseño e Implementación

3.1. Estructura del Software

En esta sección se detallarán y justificarán las decisiones de diseño tomadas. También se desarrollará en profundidad el diseño del software del IDE.

3.1.1. Estructura de un proyecto base

Cuando se crea un proyecto nuevo en CIAABOT IDE, se crea una carpeta con varias subcarpetas y archivos. En la figura 3.1.1 se observa esta estructura simplificada. Está basada en la estructura de archivos que se utiliza en el Firmware V2 de la CIAA.

Firmware V2 es un proyecto, basado en makefile para el desarrollo de software en C, especialmente enfocado en sistemas embebidos, particularmente microcontroladores. Actualmente provee soporte para:

- LPC11U68.
- LPC1769.
- LPC4337 (núcleo M4 y M0).
- LPC54102 (núcleo M4 y M0+).

Se basa principalmente en el workspace de Pablo Ridolfi y la sAPI de Eric Pernía.

En la estructura del proyecto se encuentra el archivo con extensión `.cbp`. Éste contiene la información del proyecto, su nombre, el modelo de CIAABOT utilizado y el diagrama en bloques. Los archivos `config.mk` y `Makefile` se utilizan para la compilación del código C y la descarga del mismo a la placa.

En el directorio `libs` se encuentran las diferentes bibliotecas que incluye el proyecto. En `lpc_chip_43xx` están las bibliotecas de LPCOpen para el microcontrolador LPC4337, provistas por el fabricante. En `lpc_board_ciaa_edu_4337` está la biblioteca a nivel de placa puntualmente para manejar la placa EDU-CIAA-NXP.

En la carpeta `sapi` se encuentra la biblioteca sAPI. Esta biblioteca permite manejar varios periféricos del microcontrolador de manera simplificada. Estos son:

- 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.

Finalmente, la carpeta `app` contiene la aplicación de usuario. Puntualmente el archivo `main.c` es el que se actualiza al realizar modificaciones en el editor de bloques del IDE.

```
Estructura de un proyecto de CIAABOT IDE
|   config.mk
|   Makefile
|   mi-proyecto.cbp
|
|---app
|   |---inc
|   |       programa.h
|   |
|   |---src
|           main.c
|
|---libs
|   |---lpc_board_ciaa_edu_4337
|   |   |---inc
|   |   |---src
|   |
|   |---lpc_chip_43xx
|   |   |---inc
|   |   |---src
|   |---sapi
|       |---inc
|       |---src
|
|---scripts
```

FIGURA 3.1: Estructura simplificada de un proyecto base.

3.1.2. Editor de código en bloques

El editor de bloques es la sección central de CIAABOT IDE. Aquí el usuario toma e interconecta los diferentes bloques que permiten realizar los programas. Se utilizó la biblioteca de JavaScript Blockly de Google [4], realizada por Neil Fraser entre otros.

Blockly es una biblioteca que utiliza bloques gráficos encastrables para representar los conceptos de código como variables, espresiones lógicas, bucles y más. Permite que los usuarios apliquen conocimientos de programación sin preocuparse por una sintaxis específica.

Durante el proceso de selección de la biblioteca para armar el editor gráfico, se seleccionó Blockly por las siguientes ventajas que presenta:

- Permite exportar el código simplemente.
- Es un proyecto de software libre.
- Es ampliable, ya que se pueden agregar bloques personalizados.
- Ha sido traducido a más de 40 idiomas, y permite una traducción de los bloques propios.

Esta biblioteca permite, por un lado, definir bloques con diferentes tipos de conexiones y parámetros, y por otro armar generadores que reciban los bloques y generen código. Nativamente Blockly soporta varios lenguajes, pero no C. Para eso, se utilizó parte del desarrollo de Fred Lin, BlocklyDuino [7], que resolvió varios de las estructuras típicas de C.

Blockly permite definir los bloques independientemente de los generadores de código. La definición implica la forma que tendrá el bloque para el usuario (colores, secciones, textos, tipos de bloques admitidos en cada sección y configuraciones especiales).

Para definir los bloques se utilizan archivos JavaScript y una sintaxis especial. Por ejemplo, así se define el bloque de retardo bloqueante:

```
1
2 Blockly.Blocks['ciaa_sapi_blocking_delay'] = {
3     init: function () {
4         this.appendValueInput("delay_time")
5             .setCheck("Number")
6             .appendField("Esperar ticks");
7         this.setPreviousStatement(true, null);
8         this.setNextStatement(true, null);
9         this.setColour(230);
10        this.setTooltip('Retardo bloqueante');
11        this.setHelpUrl('');
12    }
13 };
```

CÓDIGO 3.1: Definición del bloque de retardo bloqueante.

Es posible definir varios generadores de código para los mismo bloques. Esto significa que un mismo programa podría ser traducido a varios lenguajes según la necesidad. A modo de ejemplo se muestra cómo se puede definir el generador de código para lenguaje C para el bloque de retardo bloqueante:

```
1
2 Blockly.CiaaSapi.ciaa_sapi_blocking_delay = function() {
3   var delay_time = Blockly.CiaaSapi.valueToCode(this, 'delay_time',
      Blockly.CiaaSapi.ORDER_ATOMIC) || '2000';
4   var code = 'delay(' + delay_time + ');\n';
5   return code;
6 };
```

CÓDIGO 3.2: Generador de código en lenguaje C para el retardo bloqueante.

Si se observa la definición del bloque, se está agregando una entrada del tipo *value* llamada *delay_time*, que luego en el generador es tomada y utilizada como argumento de la función *delay* de la sAPI.

Internamente en el programa, cuando el usuario modifica el esquema de bloques, se llama a la biblioteca. Recibe como parámetro el nuevo diagrama en formato XML (*eXtensible Markup Language*, Lenguaje de Marcado Extensible), y el generador de código que se desea aplicar. Como resultado se obtiene una cadena de caracteres que es el código generado. Este código puede suponerse sintácticamente correcto, y en la mayoría de los casos compilará sin problemas.

Como se aprecia en la figura 3.2, el editor tiene tres secciones principales: Una barra de acciones, una caja de herramientas y el área de edición.

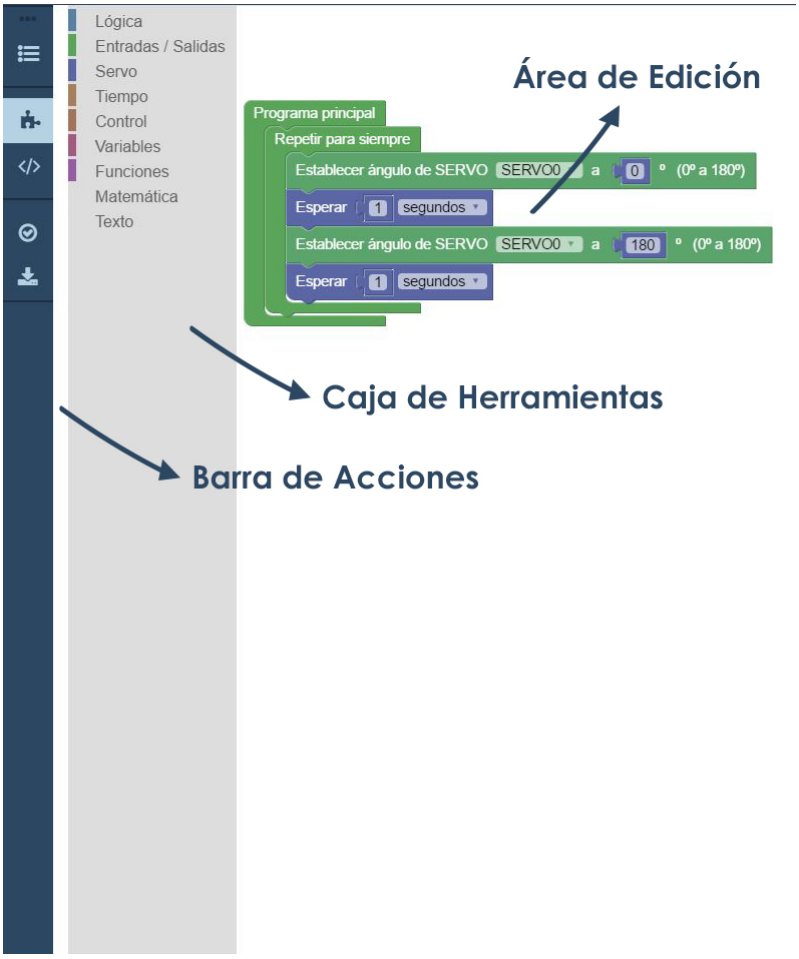

FIGURA 3.2: Secciones del editor de bloques.

En la figura 3.3 podemos ver una descripción detallada de las acciones disponibles en la Barra de Acciones. La caja de herramientas divide los bloques en categorías:

- Lógica
- Entradas / Salidas.
- Servo.
- Tiempo.
- Control.
- Variables.
- Funciones.
- Matemática.
- Texto

FIGURA 3.3: Detalle de la barra de acciones.

Los bloques se arrastran al área de edición y se interconectan con los demás. Hay diferentes tipos de bloques, según su tipo de conexión. Hay bloques que admiten solamente conexiones superior e inferior (figura 3.4), como el de *Esperar x segundos*. Estos bloques no retornan ningún valor, ni contienen otros bloques internamente. En el caso de este bloque, admite dos configuraciones: Unidad de tiempo en la que se esperará y cantidad de esas unidades. La unidad se selecciona con una lista de valores predeterminados, y la cantidad admite un bloque numérico (puede ser un número inmediato o el resultado de alguna operación).

FIGURA 3.4: Bloque Esperar x segundos.

Otros bloques, como la lectura del periférico conversor analógico a digital, retornan un valor que es tomado por algún bloque como argumento (por ejemplo una

comparación). Este tipo de bloque se muestra en la figura 3.5. No admite conexiones superior ni inferior.

FIGURA 3.5: Bloque Leer conversor analógico a digital.

Hay algunos bloques más complejos como el *Si...hacer* que permiten una configuración avanzada de la forma que presentan y los bloques que admiten. En la figura 3.6 se observa cómo se configura este bloque para que admita una segunda sección de bloques que se ejecutan de no cumplirse la condición estipulada en *Si...*.

FIGURA 3.6: Bloque Condicional.

A modo de ejemplo, en la figura 3.7 cómo se armaría en bloques el código para el barrido angular de un servomotor.

FIGURA 3.7: Ejemplo de código en bloques para el barrido de un servo.

El código de salida de la figura 3.7 se muestra a continuación:

```
1 #include "sapi.h"
2 CONSOLE_PRINT_ENABLE
3 inline void setup(void)
4 {
```

```
5   servoConfig(0, SERVO_ENABLE);
6   servoConfig(SERVO0, SERVO_ENABLE_OUTPUT);
7 }
8
9
10  void main(void) {
11  // Inicializar placa
12  boardConfig();
13
14  // Habilita cuenta de tick cada 1ms
15  tickConfig(1, 0);
16
17  // Inicializaciones del usuario prueba
18  setup();
19
20    while (TRUE) {
21      servoWrite(SERVO0, 0);
22      delay(1000 * 1);
23      servoWrite(SERVO0, 180);
24      delay(1000 * 1);
25    }
26
27  }
```

CÓDIGO 3.3: Salida del código en bloques de la figura 3.7.

Bibliografía

[1] *CIAA Firmware v2*. 2016. URL: https://github.com/ciaa/firmware_v2.

[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 Blockly*. URL: https://developers.google.com/blockly/.

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

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

[7] *Proyecto BlocklyDuino*. URL: https://github.com/BlocklyDuino/BlocklyDuino.

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

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

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