Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica

Diseño de un Entorno de Desarrollo Integrado para una Unidad Controladora de Procesos

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Oscar Mauricio Caravaca Mora

Cartago, febrero de 2010

INSTITUTO TECNOLOGICO DE COSTA RICA

ESCUELA DE INGENIERIA ELECTRONICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Dr. Ing. Pablo Alvarado Moya
Profesor lector

MSc. Ing. Ana Gabriela Ortiz León
Profesor lector

MSc. Ing. Eduardo Interiano Salguero
Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 1 de febrero de 2010

Declaratoria de autenticidad

Declaro que el presente informe de proyecto de graduación ha sido realizado enteramente por mi persona. En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el contenido del correspondiente informe de proyecto de graduación.

Cartago, febrero 2010

Oscar Mauricio Caravaca Mora

Carné: 200124637

Resumen

El proyecto TeleLAB, es una iniciativa de parte de la Escuela de Ingeniería Electrónica, cuyo propósito es la creación de un laboratorio en línea enfocado a temas relacionados al área del Control Automático. Este laboratorio en línea, brindará a los estudiantes la posibilidad de realizar distintos experimentos de manera remota, ya sea, por medio de un sistema de red de área local (LAN) o Internet.

Para lograr el propósito del proyecto TeleLAB, se requiere la adquisición del equipo adecuado, el cual debe cumplir con las siguientes características:

- Conectividad a través de una interfaz *Ethernet*.
- Dispositivos de entrada/salida.
- Conjunto de funciones programables para el desarrollo de sistemas de control.

Actualmente la Escuela ha desarrollado un dispositivo electrónico que cumple con los requisitos mencionados para formar parte del laboratorio en línea. Éste fue diseñado como parte de un proyecto de graduación anterior llamado: "Unidad Controladora de Procesos para el diseño, análisis, simulación e implementación de sistemas de Control Automático a través de redes TCP/IP" [2].

Este dispositivo electrónico no cuenta con un sistema de interacción adecuado con el usuario, de tal manera que éste pueda realizar los experimentos que desee en poco tiempo y con un mínimo de errores. El presente proyecto, trata del diseño y desarrollo de un nuevo sistema de interacción con el usuario denominado Entorno de Desarrollo Integrado para una Unidad Controladora de Procesos. El proceso de diseño consistió de cinco etapas: la creación de un lenguaje intermedio, el desarrollo de un traductor de código intermedio, el diseño de una interfaz gráfica, la creación de un traductor de lenguaje gráfico a código intermedio y finalmente la implementación de la herramienta de simulación.

El nuevo sistema de interacción humano-máquina se diseñó con el fin de disminuir el tiempo de desarrollo de las aplicaciones, minimizar errores y simplificar la implementación y diseño de nuevos sistemas de Control Automático.

Palabras claves: laboratorio en línea, Unidad Controladora de Procesos, Entorno de Desarrollo Integrado, simulación, estructuras de Control Automático, tiempo de desarrollo, GNU Octave, red de área local (LAN), TCP/IP, Internet.

Abstract

The project TeleLAB is an idea of the Electronic Engineering School. Its purpose is to create an online laboratory which would focus on topics related to Automatic Control. This online laboratory would provide students the possibility of doing different experiments remotely through a local area network system or Internet.

In order to achieve the purpose of the project TeleLAB it is necessary to adquire the right equipment. This equipment must have the following characteristics:

- Connectivity through an Ethernet interface.
- in/out devices.
- Programmable functions set for development of Automatic Control Systems.

Currently the school has developed an electronic device that included all the requirements mentioned before to be part of the online laboratory. This device was designed as part of a graduation project called "Process Control Unit for the design, analysis, simulation and implementation of automatic control systems via TCP/IP" [2].

This electronic device did not have an appropriate interaction system with the user, so that the users could do the experiments that they want in a short time and minimum errors. This project is about creation of a new interaction system called Integrated Development Environment for a Process Control Unit. The design process involved five stages: creation of intermediate language, development of the intermediate code compiler, design of the graphic user interface, design of a compiler from graphic language to intermediate code, and finally, development of the simulation tool.

The new human-machine interaction system was designed with the purpose to decrease application development time, minimize errors and facilitate performing and designing of new Automatic Control Systems.

Key words: Online laboratory, Unit Process Control, Integrated Development Environment, simulate, simulation, Automatic Control structures, development time, GNU Octave, local area network (LAN), TCP/IP, Internet.

Dedicatoria

A Dios en quien tengo puesta mi fe, y a mis padres Gerardo Caravaca Hernández y Yamileth Mora Villalobos que con su esfuerzo, cariño y apoyo incondicional me han ayudado a salir adelante en las diferentes etapas de mi vida.

Agradecimiento

En primer lugar a Dios por la oportunidad de vivir, por darme a mis padres Gerardo Caravaca Hernández y Yamileth Mora Villalobos, así como a mis hermanos Iliana, Ronald y Carlos, a quienes agradezco por su apoyo y comprensión durante toda mi carrera.

Al profesor asesor MSc. Ing. Eduardo Interiano Salguero por su correcta orientación durante el proceso de desarrollo del proyecto.

A Daniel Castro Molina por su contribución en el mejoramiento y corrección de algunos detalles técnicos relacionados con la Unidad Controladora de Procesos.

A las estudiantes de Diseño industrial Fiona Zoe Villalobos Custance, Ana Gabriela Cárdenas Carmona, Arlette Calvo Montero y al diseñador gráfico Niger Jirón Mora por sus colaboraciones en el diseño de los elementos gráficos utilizados en este proyecto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1: Introducción

La Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica, tiene como meta la creación de un laboratorio de Control Automático de acceso remoto vía redes TCP/IP para el diseño y experimentación de sistemas de Control Automático. Dicho laboratorio, contará con diversas herramientas, las cuáles, facilitarán a los estudiantes su aprendizaje en diferentes temas relacionados al Control Automático.

Como parte de las ventajas principales que conlleva la creación de este laboratorio, es la posibilidad de acceder a cualquiera de los equipos de diseño, simulación e implementación disponibles en TeleLAB desde cualquier lugar por medio de Internet. Esto permitirá a los estudiantes realizar sus diseños y experimentos desde su propio hogar o algún otro sitio con acceso a Internet y en todo momento u hora que lo deseen, la Figura 1.1 representa el diagrama general de TeleLAB.

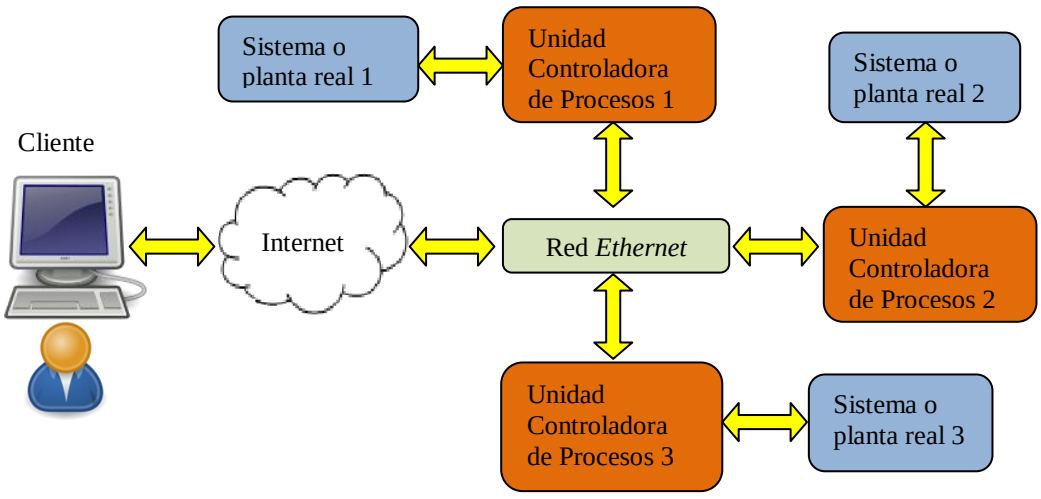

Figura 1.1 Diagrama general de TeleLAB.

En cuanto a los requisitos técnicos necesarios para que un sistema electrónico funcione como experimento en TeleLAB se mencionan los siguientes: “capacidad de integrarse a una red TCP/IP, facilidad de modificación de los parámetros de control, posibilidad de ampliación de funciones, documentación” [2].

Como parte de las herramientas con las cuales contará TeleLAB, en la Escuela de Ingeniería Electrónica se ha llevado a cabo el desarrollo inicial de una "Unidad Controladora de Procesos para el diseño, análisis, simulación e implementación de sistemas de Control Automático a través de redes TCP/IP" [2]. Esta unidad electrónica, fue implementada, en su primera fase, como un proyecto de graduación anterior, que a la vez fue planteado como una iniciativa para lo que eventualmente se pretende obtener con TeleLAB.

Dicha Unidad Controladora permite estimular, medir variables y crear estructuras de control que facilitan el análisis de sistemas lineales, mediante la aplicación de un sistema operativo en tiempo real sobre un *hardware* especializado. Cuenta además, con conectividad vía *Ethernet*, capacidad de adquisición de datos, y diferentes interfaces de entrada/salida. La Figura 1.2 representa el diagrama básico de la Unidad Controladora de Procesos.

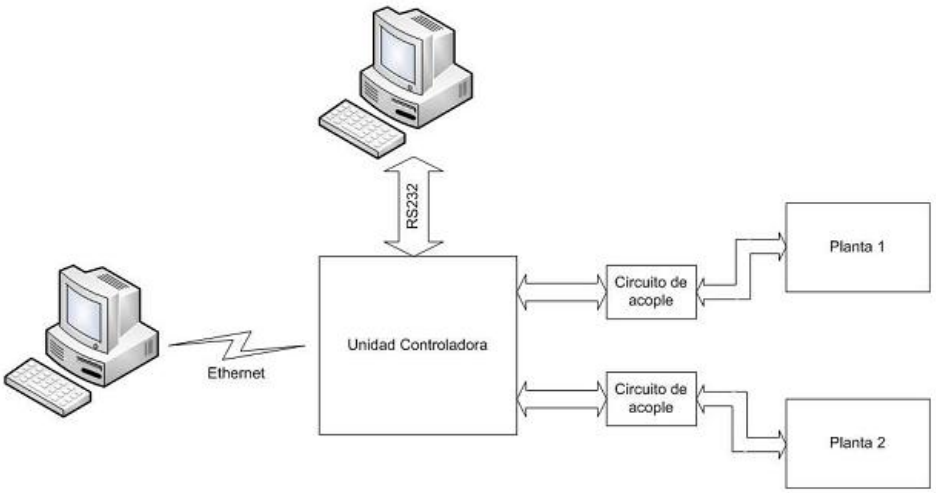

Figura 1.2 Diagrama básico de la Unidad Controladora de Procesos desarrollada en la Escuela de Ingeniería Electrónica [2].

Como se mencionó anteriormente, solo se ha llevado a cabo la primera fase del desarrollo de este sistema controlador, la cual consistió del diseño e implementación del *hardware* especializado, y un protocolo de comunicación para la programación del dispositivo en lenguaje de máquina virtual para aplicaciones de Control Automático, denotado aquí como lenguaje de máquina. Debido a esto, el sistema controlador presenta dos limitantes: un conjunto de operaciones no muy extenso y un sistema de programación de aplicaciones

basado en un lenguaje de máquina. Tales limitantes dificultan la implementación de las aplicaciones en la Unidad Controladora de Procesos, lo cual se pretende resolver en las siguientes fases de desarrollo en dicho dispositivo, ya que éste es un sistema abierto a mejoras y actualizaciones.

1.1 Problemática de la Unidad Controladora de Procesos

El sistema electrónico que actualmente se ha desarrollado, tiene su propio protocolo y lenguaje de programación. Cada vez que un usuario desea implementar una aplicación, debe realizar una conexión *Telnet* a dicho sistema por medio de una computadora vía *Ethernet*, una vez establecida dicha conexión, el usuario programa la unidad electrónica por medio de instrucciones en lenguaje de máquina, introduciendo las representaciones hexadecimales de cada instrucción involucrada en la aplicación que desea implementar, lo cual significa, que existe una brecha entre el usuario y el lenguaje de máquina, esto causa que el sistema de programación resulte propenso a errores y con un tiempo de desarrollo lento, lo cual dificulta la implementación de las aplicaciones, limitando además el aprovechamiento de la unidad electrónica.

1.2 Entorno de Desarrollo Integrado

Se propone diseñar un Entorno de Desarrollo Integrado que permita realizar los experimentos de control por medio de aplicaciones gráficas, en las cuales se verían representadas las instrucciones del sistema electrónico.

Esta forma de programación, le brindará al usuario, la posibilidad de realizar las tareas en un tiempo menor al que le toma actualmente, contará además con un sistema implícito de detección de errores y la posibilidad de simular aplicaciones de Control Automático, esto con el fin de cerrar la brecha entre el usuario y el lenguaje de máquina, simplificar la programación de aplicaciones y obtener así un mejor aprovechamiento de la Unidad Controladora de Procesos.

La Figura 1.3 representa el diagrama de bloques de la solución desarrollada. Dicha figura muestra un Entorno de Desarrollo Integrado que cuenta con símbolos gráficos y capas de traducción de símbolos gráficos a lenguaje de máquina, en las cuales pueden existir varios niveles de traducción intermedios.

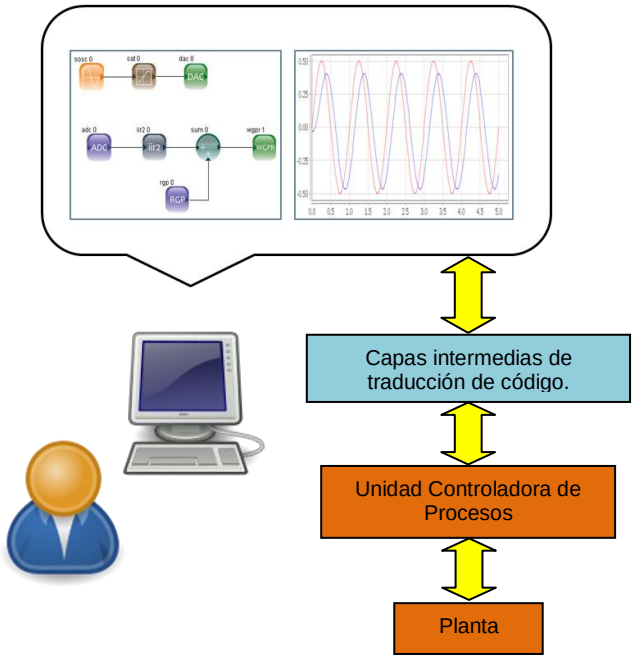

Figura 1.3 Diagrama de bloques de la solución desarrollada.

El proceso de diseño seguido para la creación del Entorno de Desarrollo Integrado se presenta en la Figura 1.4, donde se muestra que el proceso consistió de 5 etapas, las cuales reflejan que el diseño y la implementación se hicieron siguiendo una metodología de construcción de abajo hacia arriba (*down-top*).

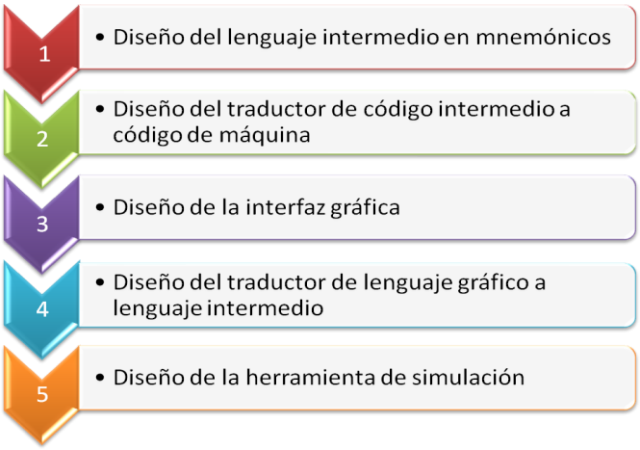

Figura 1.4 Etapas del proceso de diseño seguido.

La Figura 1.5 describe el diagrama de flujo de la interacción entre el usuario y el Entorno de Desarrollo Integrado.

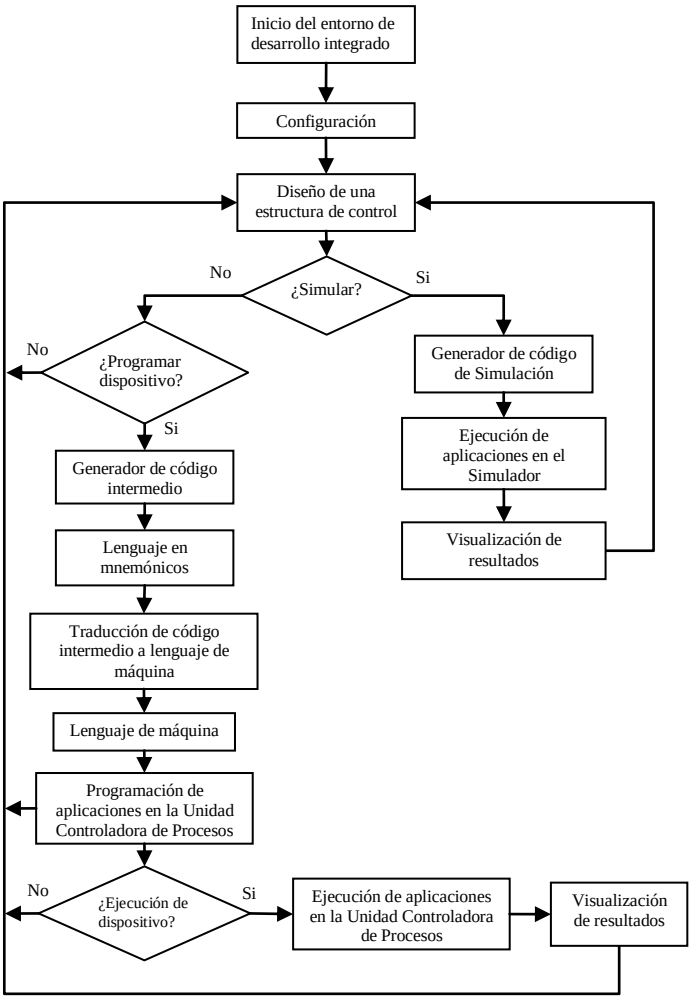

Figura 1.5 Diagrama de flujo de la interacción entre el usuario y el IDE.

Capítulo 2: Meta y Objetivos

2.1 Meta

Crear un laboratorio de Control Automático de acceso remoto vía redes TCP/IP, con el fin de brindar mayores facilidades de diseño, implementación y experimentación que las disponibles actualmente, a los estudiantes de la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

2.2 Objetivo general

Diseñar un Entorno de Desarrollo Integrado que permita al usuario la programación de aplicaciones gráficas de Control Automático, para cerrar la brecha entre el usuario y el lenguaje de máquina de la Unidad Controladora de Procesos, y disminuir el tiempo de desarrollo de nuevas aplicaciones.

2.3 Objetivos específicos

1) Crear un lenguaje intermedio (mnemónicos) para la programación de aplicaciones en la Unidad Controladora de Procesos.

2) Desarrollar un generador de código de máquina que traduzca del lenguaje intermedio (mnemónicos) al lenguaje máquina entendido por la Unidad Controladora de Procesos.

3) Desarrollar una interfaz gráfica orientada a diagramas de bloques y funciones tales como visualización de tablas, gráficas, etc.

4) Diseñar un traductor de las aplicaciones programadas en el *software* gráfico al lenguaje intermedio.

5) Brindarle al usuario la posibilidad de simular o emular las aplicaciones.

6) Realizar un manual de usuario del Entorno de Desarrollo Integrado diseñado.

Capítulo 3: Marco teórico

3.1 Laboratorios remotos para la enseñanza de sistemas de Control Automático [11]

Los laboratorios remotos son sistemas que permiten a un usuario controlar un experimento de manera remota por medio de un computador. Actualmente estos sistemas permiten la enseñanza de Control Automático a distancia, lo cual tiene ventaja sobre los laboratorios presenciales.

Desventajas de los laboratorios presenciales:

- Disponibilidad de horario limitada: los estudiantes deben realizar sus experimentos en horarios establecidos dentro de la jornada habitual de lecciones.
- Limitación de cupo en los laboratorios: el costo de los equipos limita el cupo de los laboratorios y hace que en muchos casos los estudiantes deban compartir los sistemas disponibles.
- Gastos económicos y de tiempo: el estudiante debe desplazarse desde su hogar hasta la localidad del laboratorio.

Ventajas de los laboratorios remotos:

- Aumentan la disponibilidad de los equipos de laboratorio: los equipos se pueden utilizar en horarios ampliados que pueden llegar incluso a las 24 horas todos los días.
- Mejora la seguridad de los equipos de laboratorio: elimina el acceso físico a los equipos, lo que minimiza el riesgo de robo, daño o alteraciones en los equipos.
- Ahorro económico y de tiempo: el estudiante realiza sus experimentos desde su hogar sin la necesidad de desplazarse gastando dinero y tiempo.

Algunos ejemplos de laboratorios remotos son:

- RECOLAB: *Remote Control Laboratory*.
- ACT: *Automatic Control Telelab*.
- TeleLab: *Remote Automations Lab*.

3.1.1 RECOLAB: *Remote Control Laboratory* [10]

Recolab es un laboratorio remoto del Departamento de Ingeniería de la Universidad Miguel Hernández en España. Este laboratorio permite al usuario aplicar una estructura de control a un servomotor, así también como a un cilindro deslizante. El sistema permite visualizar por medio de una cámara en línea la ejecución en tiempo real de las aplicaciones de control disponibles, además provee una gráfica y una tabla de resultados descargable.
Recolab proporciona una arquitectura para la ejecución a distancia en tiempo real de estructuras de control sobre sistemas físicos por medio de Matlab/Simulink. El sistema permite ejecutar a través de Internet aplicaciones de Simulink en tiempo real sobre sistemas físicos en el laboratorio.

Componentes fundamentales de RECOLAB:

- Computadora con tarjeta E/S National Instruments 6024E.
- Servomotor de corriente continua, y cilindro deslizante para levitación por aire.
- Cámara *Web*.
- Servidor *Web* Apache.
- Matlab/Simulink, *Real -Time Windows Target Toolbox* y *Real -Time Workshop Toolbox*.

Este sistema tiene la desventaja de que las estructuras de control aplicables son limitadas, y utiliza a MATLAB/Simulink el cual es *software* propietario. La interfaz de usuario se basa en formularios de relleno y menús de selección, y no permite la manipulación de bloques gráficos.

3.1.2 ACT: *Automatic Control Telelab* [23]

Automatic Control Telelab es un laboratorio remoto para sistemas de Control Automático de la Universidad de Siena en Italia. Este laboratorio permite controlar servos de posición y de velocidad, así como un levitador magnético y otros experimentos con fluidos. El ACT provee configuración en tiempo real, observación de los experimentos, así como la adquisición de datos de manera remota. Este laboratorio se basa en MATLAB/Simulink para el diseño de las estructuras de control, y su interfaz no permite la manipulación de bloques gráficos, además para diseñar una nueva estructura de control el usuario primero debe crearla utilizando Simulink para luego agregarla al sistema de interacción del laboratorio remoto.

3.1.3 TeleLab: *Remote Automations Lab* [20]

Telelab es una plataforma diseñada para proveer a los estudiantes acceso remoto a un controlador de lógica programable para realizar experimentos de automatización industrial. Los estudiantes pueden programar este dispositivo a través de Internet. Este controlador está físicamente conectado a un sistema real y cuenta con sensores y actuadores. Un sistema de adquisición de imagen permite a los estudiantes recibir audio y video en vivo con el fin de verificar el comportamiento del sistema según la aplicación programada. La Figura 3.1 presenta una ilustración de Telelab.

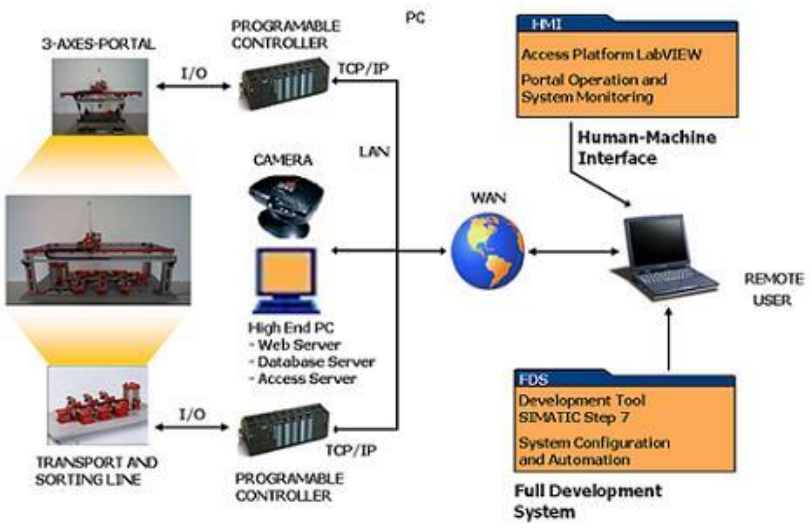

Figura 3.1 Esquema general de Telelab: *Remote Automations Labs.*

3.2 Lenguajes de Programación [18] [27]

Los lenguajes de programación, se pueden definir como lenguajes artificiales diseñados por el ser humano para la comunicación con sistemas computacionales. Se utilizan para desarrollar programas que controlan el comportamiento lógico o físico de una máquina, implementar algoritmos e incluso para la comunicación entre personas.

Los lenguajes de programación están conformados por un conjunto de reglas sintácticas y semánticas que dan orden a su estructura y el significado de sus expresiones. La definición de un lenguaje de programación consiste básicamente de tres elementos:

- Sintaxis.
- Semántica.
- Pragmática.

3.2.1 Sintaxis

Se refiere a la manera en la que los símbolos y expresiones pueden ser combinados para crear sentencias válidas dentro del lenguaje. Describe la estructura dentro de la cual las expresiones se consideran válidas en una cadena de instrucciones, frases o símbolos en el lenguaje. La sintaxis se enfoca en la forma y estructura de los símbolos y expresiones pero no en su significado.

3.2.2 Semántica

La semántica se refiere al significado de una expresión o sentencia sintácticamente válida en un lenguaje de programación.

3.2.3 Pragmática

Pragmática de un lenguaje de programación tiene que ver con la manera en que el lenguaje está destinado a ser utilizado en la práctica.

3.2.4 Gramática de un lenguaje de programación y BNF

La gramática de un lenguaje de programación consiste de 4 elementos:
<∑, N, P, S>

1. Un conjunto ∑ de elementos terminales, básicamente el alfabeto del lenguaje, las cuales se utilizan para construir las sentencias del lenguaje.
2. Un conjunto N de elementos no-terminales, o categorías sintácticas, cada una de ellas representa un conjunto de instrucciones o sentencias propias del lenguaje.
3. Un conjunto finito P de reglas o producciones, éstas indican de que manera cada sentencia es definida en términos de elementos terminales y no terminales.
4. Un característico símbolo no terminal S, el símbolo de inicio que especifica la sentencia o categoría que está siendo definida, por ejemplo, sentencia o programa.

La forma tradicional de representar lenguajes de programación es de la siguiente manera:

<declaración> ::= variable <lista variable> : <tipo>;

Donde “variable”, “:” y “;” son símbolos terminales, o sea que no dependen de otra regla, el símbolo “::=” indica que “<declaración> es compuesta de…, o defina por…”. Esta forma de representación se conoce como ***Backus-Naur Form*** **o BNF**, esta manera de representación de lenguajes de programación fue creada por John Backus, con el fin de representar el lenguaje de ALGOL, y ayuda a simplificar la sintaxis y semántica de los lenguajes de programación [25]. En este proyecto se utilizó una gramática libre de contexto tipo 2 según [18].

3.3 Compiladores [15]

Como se mencionó en secciones anteriores, los lenguajes de programación son creados para la comunicación entre el ser humano y sistemas computacionales o máquinas, con el fin de lograr un comportamiento deseado de tales sistemas, implementando por medio de ellos los algoritmos necesarios que llevan a cabo tal comportamiento.

Tomando en cuenta, que los lenguajes de programación son diseñados para ser entendidos por el ser humano, normalmente utilizan nombres de instrucciones que tengan un valor significativo para el usuario, así también su gramática; si embargo, estos lenguajes de programación por sí solos no son entendidos ni interpretados por los sistemas computacionales o máquinas con los que se pretende establecer comunicación, ya que una máquina solo entiende combinaciones de números hexadecimales o binarias en alguna secuencia establecida, a lo cual se le conoce como lenguaje de máquina, resulta entonces necesario la creación de sistemas que traduzcan los programas escritos en lenguaje de programación entendido por el usuario al lenguaje de máquina.

A los sistemas que se encargan de traducir código entre lenguajes de programación entendidos por el usuario y lenguaje de máquina se le conocen como traductores, compiladores o intérpretes. Aunque existe diferencia entre los conceptos de compilador e intérprete, ambos son sistemas de traducción de código cuyo fin es lograr el entendimiento entre los lenguajes de programación entendidos por el usuario y el lenguaje de máquina.

3.3.1 Traductor

Un compilador o traductor se define como un sistema o programa que traduce código escrito en un lenguaje fuente a un código equivalente escrito en un lenguaje objeto o destino, indicando los posibles errores eventualmente existentes durante la traducción.

La Figura 3.2 representa gráficamente el concepto de traductor.

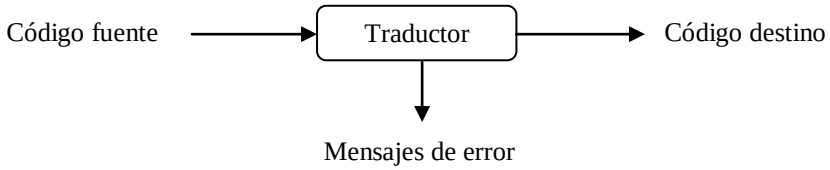

Figura 3.2 Representación de un traductor.

Tipos de traductores

A. Compiladores

Sistema que traduce un código fuente o de origen a un código destino, produce un fichero objeto o ejecutable final.

B. Intérpretes

El intérprete es similar a un compilador, con la diferencia que su salida es la ejecución del código traducido, es decir, no se obtiene un fichero con el código de máquina como salida.

C. Ensambladores y macroensambladores

Es un compilador sencillo que permite la traducción de código fuente directamente a instrucciones en lenguaje de máquina, en esta clase de compiladores existe una correspondencia uno a uno entre las instrucciones de código fuente y las instrucciones de lenguaje de máquina. Existen además los macro ensambladores, los cuales traducen macro instrucciones a varias instrucciones en lenguaje de máquina y son la antesala a los lenguajes de alto nivel.

D. Compilador cruzado

Son compiladores que producen código para ser empleado en otra máquina. De esta manera se puede crear el sistema operativo de otro computador utilizando lenguajes de alto nivel.

3.3.2 Estructura de un traductor

Un traductor se compone básicamente de dos partes principales, análisis y síntesis. La parte de análisis es la que se encarga de realizar las correcciones correspondientes al programa fuente, así también de generar las estructuras requeridas según su gramática para realizar la síntesis. La etapa de síntesis se encarga de generar el código destino a partir de las estructuras resultantes en la etapa de análisis.

A la etapa de análisis se le conoce como etapa inicial o *"Front end"* y a la etapa de síntesis se le conoce como etapa final ó*"Back end"*. La Figura 3.3 representa las partes de un traductor.

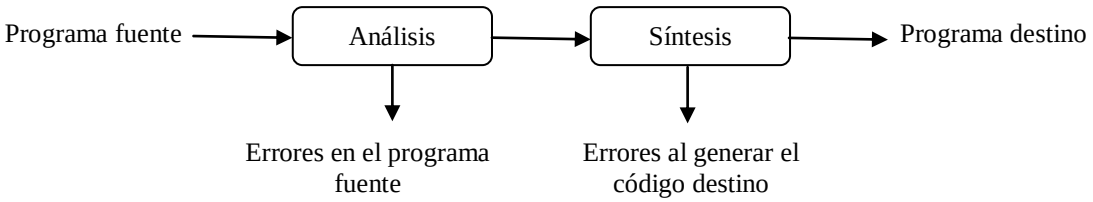

Figura 3.3 Representación de las partes principales de un traductor.

Fases de la etapa de análisis o *"Front end"*

La etapa de análisis comprende las fases de análisis léxico, sintáctico y semántico, así también como la generación de código intermedio.

Análisis Lexicográfico

La fase de análisis léxico o lexicográfico divide el programa y analiza cada uno de los elementos que lo componen en caracteres, luego agrupa tales caracteres en secuencias significativas llamadas *"lexemas"*, en el caso de que cada lexema pertenezca a alguna categoría gramatical dentro de la estructura del lenguaje así como variables definidas por el usuario, tipos, números, signos de puntuación, palabras reservadas, el analizador léxico genera una salida llamada "*token"* que luego sería utilizado en el analizador sintáctico.

Análisis Sintáctico

Verifica que la estructura del lenguaje corresponda a las reglas gramaticales definidas para el lenguaje, verifica que los *"tokens"* generados por el analizador léxico formen una estructura aceptada o permitida según la gramática definida por el lenguaje.

Los elementos de cada lenguaje se organizan dentro de una estructura jerárquica, por lo que el analizador sintáctico verifica que tal organización jerárquica sea respetada.

Análisis Semántico

Verifica todo lo relacionado con el significado, en cuanto a tipos, rango de valores, existencia de variables, etc.

Cada una de las etapas de análisis léxico, sintáctico y semántico pueden generar errores, por ejemplo al escribir mal una variable se genera un error lexicográfico, una expresión aritmética o paréntesis no equilibrados podría generar un error sintáctico y la asignación de un valor a otro de distinto tipo genera errores de significado o de semántica.

Fases de la etapa de síntesis o *"Back end"*

Generación de código intermedio

Una vez realizada la fase de análisis se genera una representación intermedia de código. Esta representación intermedia debe tener la característica de ser fácil de generar y sencilla de traducir al código de máquina.

Optimización de código

Esta fase trata de reducir el código de tal manera que sea rápido de ejecutar, facilitando la tarea de la siguiente fase, la cual es la generación de código de máquina.

Esta fase identifica las sentencias o instrucciones redundantes, y busca representar el código equivalente al programa originalmente escrito en el lenguaje fuente, de manera que su ejecución sea rápida, sencilla y no redundante.

Generación de código de máquina

Esta fase se encarga de generar el código de máquina a partir del código intermedio optimizado, toma cada una de las instrucciones en lenguaje intermedio y las traduce a su equivalente en código de máquina.

Tabla de símbolos

En general la tabla de símbolos es una estructura de datos, la cual contiene la información de los identificadores que el usuario define, tales como constantes, variables, funciones, etc. La tabla de símbolos puede ser una tabla *hash* o un árbol binario de búsqueda, y hace la función de una especie de diccionario. La Figura 3.4 ilustra la estructura general de un traductor.

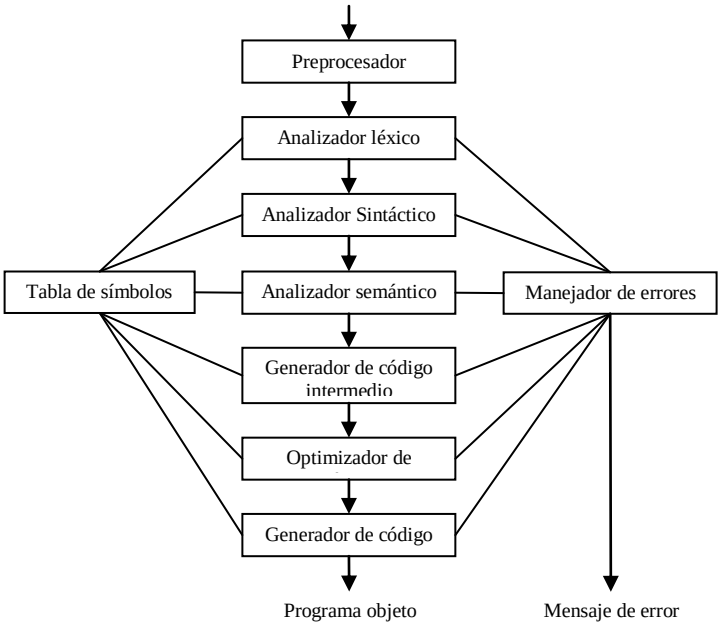

Figura 3.4 Esquema general de un compilador y sus fases.

3.4 Interacción hombre-máquina

Se conoce como interacción hombre-máquina, al área de estudio, diseño y desarrollo relacionado con el intercambio de información entre el ser humano y las máquinas. El objetivo de esta disciplina es lograr que tal intercambio o transferencia de información sea eficiente, minimizando errores, incrementando la satisfacción del usuario y disminuyendo su frustración; además, de mejorar la productividad de las labores que involucran a las personas y las máquinas [26].

3.4.1 Objetivos de diseño de Sistemas hombre-máquina [12]

En cuanto al diseño de los sistemas de interacción hombre-máquina, existen tres objetivos generales que se deben satisfacer, estos son: usabilidad, seguridad de utilización y libre de deficiencias, estos objetivos de diseño se ilustran en la Figura 3.5.

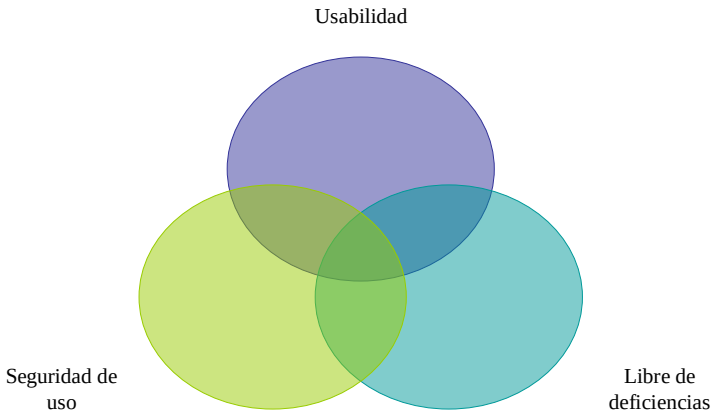

Figura 3.5 Objetivos de diseño de los sistemas de interacción hombre-máquina.

Usabilidad

La usabilidad es un concepto que se refiere básicamente a la facilidad con la que los usuarios logran aprender y utilizar una herramienta para un fin dado. La usabilidad abarca tres aspectos: la eficacia, eficiencia y satisfacción del usuario.

El siguiente gráfico resume los aspectos que involucra el concepto de usabilidad.

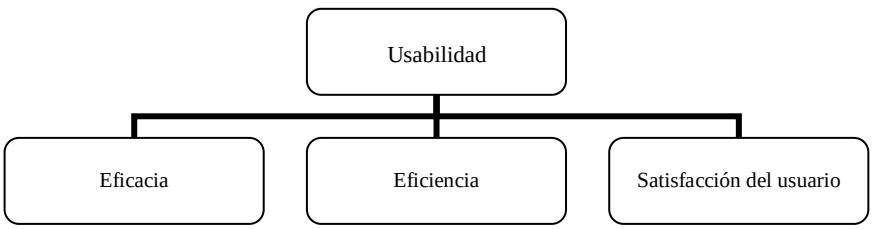

Figura 3.6 Aspectos de la usabilidad.

Seguridad de Utilización

La seguridad de utilización de un sistema o herramienta hombre-máquina se refiere a la capacidad que tiene un sistema en ser comprensible, predecible, controlable *y* robusto para el usuario. Dentro de estos aspectos, comprensible quiere decir que el usuario tiene conocimiento y comprende el funcionamiento de las diferentes tareas que puede realizar el sistema, predecible indica que el usuario es consciente de los límites de operación del sistema, controlable es la capacidad que el sistema le concede al usuario para detener y anular las funciones del sistema dentro de un tiempo de respuesta dado, y en cuanto a robusto, se refiere a cómo reacciona el sistema ante situaciones impredecibles, las cuales pueden suceder en el proceso de aprendizaje y familiarización de las funciones del sistema, en el caso de un malfuncionamiento o al probar los límites de operación.

La Figura 3.7 resume los aspectos concernientes a la seguridad de utilización.

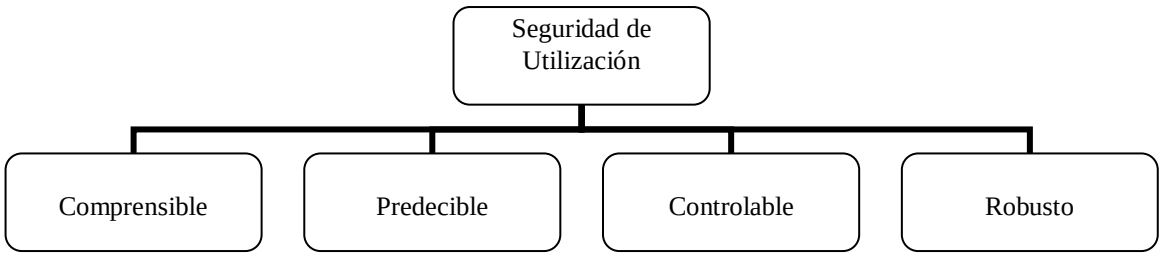

Figura 3.7 Aspectos sobre seguridad de utilización.

Libre de deficiencias

Este aspecto indica que un sistema hombre-máquina se debe diseñar procurando que sea libre de errores, así como errores de construcción debido a la no consideración de todas las posibles condiciones de operación, errores de instrucción debido a la falta de documentación para el usuario, entre otros. El objetivo es minimizar toda clase de posibles deficiencias en el diseño y utilización de un sistema humano-máquina.

3.4.2 Interfaces gráficas de usuario [7]

El diseño de interfaces de usuario es un área del campo de estudio de las interacciones hombre-máquina. En este caso la interfaz de usuario es la parte de una computadora, en cuanto a *software* se refiere, que el usuario puede ver, escuchar, tocar, hablar a, o de otra manera entender o dirigir.

La interfaz de usuario tiene dos componentes: entrada y salida. La entrada es la manera en la que el usuario expresa, indica o comunica al computador sus ideas, intenciones o deseos sobre lo que pretende que la computadora realice, entre los elementos que funcionan como entradas se pueden mencionar: el teclado, el ratón, pantallas táctiles, la voz en dispositivos cuyas instrucciones se pueden especificar de manera oral, entre otros. La salida es la manera en la que el computador retorna los resultados de sus cálculos o tareas solicitadas por el usuario, comúnmente estas salidas son una pantalla en la que la computadora proyecta la información solicitada por el usuario, así como ciertos mecanismos de audio.

Interfaz Gráfica de usuario

La interfaz gráfica de usuario es aquella cuyo principal mecanismo de interacción es un dispositivo señalador, el cual sería el equivalente electrónico a una mano humana. El usuario interactúa con una colección de objetos, los cuales el usuario puede ver, escuchar o percibir de alguna manera. El usuario puede realizar distintas operaciones, acciones o modificaciones con los objetos, señalando, seleccionando y manipulándolos.

Las interfaces gráficas de usuario abarcan los siguientes aspectos:

A. Estilos de interacción.
B. Características de interfaces gráficas.
C. Ventajas y desventajas de la interfaz gráfica de usuario.

A. Estilos de interacción

Los estilos de interacción se refieren a la manera, forma, método o mecanismo que el usuario utiliza para comunicarse con la computadora y viceversa. Hoy día, existen diferentes estilos que el diseñador puede escoger para su interfaz gráfica, estos se mencionan a continuación:

- Línea de instrucciones.
- Menús de selección.
- Formularios de relleno.
- Manipulación directa.
- Antropomórficos.

Línea de Instrucciones

La línea de instrucciones es el método de interacción más antiguo. En este método, se introducen las instrucciones en un área indicada en la pantalla, estas instrucciones pueden ser caracteres, palabras o múltiples combinaciones de palabras.

Este método tienen la desventaja de que sus instrucciones deben ser memorizadas, puesto que no se indican en la pantalla las instrucciones existentes, la sintaxis de las instrucciones resulta complicada, además es un método propenso e intolerante a los errores que puede producir el usuario.

Menús de selección

Trata de un conjunto de opciones que el usuario tiene a disposición para escoger, el usuario selecciona una opción en la pantalla ya sea con un mecanismo de indicación y selección o por medio de una combinación de teclas (*hotkeys*).

Este método brinda la ventaja de que el usuario no necesita memorizar el conjunto de instrucciones, además, por medio de menús se puede lograr realizar una tarea compleja en

pocos pasos y ayuda al proceso de toma de decisiones, lo cual es de gran ayuda para usuarios pocos familiarizados con el sistema.

Formularios de relleno

Los formularios de relleno se utilizan frecuentemente para la recolección de información. Éstos se componen de una serie de campos en los cuales los usuarios introducen la información solicitada. Este sistema se deriva de los formularios en papel y tiene la ventaja, que debido a eso, este sistema resulta familiar.

Estilo de manipulación directa

Este sistema permite al usuario interactuar directamente con elementos visualizados en la pantalla (llamados objetos), por medio de un mecanismo señalador, reemplazando la necesidad de utilizar el teclado para ingresar las instrucciones que representan cada objeto y las acciones relacionadas.

Normalmente en este estilo de interacción el usuario aplica las acciones por medio de barras de menús, menús desplegables y existe además, un área de trabajo sobre la cual se manipulan los objetos.

Estilo antropomórfico

Una interfaz antropomórfica tiene el fin de lograr una interacción entre personas y computador de la misma manera en que una persona interactúa con otra. Este tipo de interfaz involucra lenguaje hablado, movimiento de las manos, expresiones faciales y movimiento de ojos. El desarrollo de este tipo de interfaz requiere del estudio del comportamiento humano, en cuanto a gestos, lenguaje hablado, etc.

La Tabla 3.1 resume las ventajas y desventajas de los distintos estilos de interacción.

Tabla 3.1 Resumen de las ventajas y desventajas de los diferentes estilos de interacción para una interfaz gráfica de usuario [7].

Estilo	Ventajas	Desventajas
Línea de instrucciones	Conserva el espacio en pantalla Flexible para el usuario experto	La instrucciones debe ser memorizadas Requiere un proceso de aprendizaje previo Intolerante a errores Difícil de utilizar para el usuario no experto
Menús de selección	Memoria de reconocimiento Complejidad de interacción reducida Ayuda en el proceso de toma de decisiones Minimiza la escritura de código Ayuda al usuario no familiarizado	Consume mucho espacio en pantalla
Formulario de relleno	Formato familiar Simplifica la introducción de información Requiere un tiempo de aprendizaje mínimo	Consume espacio en pantalla No prevé errores de escritura
Estilo de manipulación directa	Aprendizaje rápido Facilidad de recordar las instrucciones Aprovecha características visuales y espaciales Facilidad de recuperación de errores	Complejidad de diseño Posibilidad de desorden en la pantalla
Estilo antropomórfico	Natural	Difícil de implementar

B. Características de la interfaz gráfica de usuario

Las interfaces gráficas involucran algunos conceptos que es necesario entender y definir.

Presentación visual. La presentación visual de la interfaz se refiere a lo que el usuario ve en la pantalla. Los elementos que normalmente incluye este tipo de sistema son menús, ventanas, iconos, controles basados en la visualización en pantalla, el ratón u otro dispositivo como señalador.

Interacción *Pick-and-Click.* Hace referencia a la necesidad de identificar los elementos para aplicar las acciones que se requieren. A la acción del usuario de reconocer un elemento se le conoce como *Pick*, y a la tarea de aplicar una acción se le conoce como *Click*. El *mouse* o ratón es el dispositivo más común para el mecanismo de *Pick-and-Click*.

Conjunto limitado de opciones de interfaz. El conjunto de alternativas gráficas que existen en el sistema son todas aquellas presentadas en la pantalla.

Visualización. La visualización es un proceso cognitivo que permite a la gente entender la información que es difícil de percibir, ya que a veces es muy voluminosa o abstracta. No se trata de representar la imagen realista de algo sino lo que refleje la información más relevante.

Orientación a objetos. Los sistemas gráficos se componen de objetos y acciones. Los objetos son los que el usuario ve y puede manipular como una unidad. Un buen diseño mantiene al usuario enfocado en los objetos y no en cómo realizar las acciones.

Los objetos de una interfaz se clasifican en tres clases significativas: datos, contenedores y dispositivos.

Los datos, son los objetos que presentan información sobre el control y las acciones que se pueden realizar o modificar.

Los objetos contenedores son aquellos que contienen a otros objetos. Se utilizan para almacenar dos o más objetos relacionados entre sí y para facilidad de acceso. Existen tres tipos de contenedores: lugar de trabajo o *workplace*, carpetas o *folders* y área de trabajo o *workarea*. El lugar de trabajo o escritorio, es el lugar de almacenamiento para todos los objetos; las carpetas son lugares de almacenamiento de duración prolongada y el área de trabajo es el lugar de almacenamiento temporal de objetos que se están utilizando para realizar alguna tarea.

Los dispositivos son aquellos objetos que representan objetos físicos en el mundo real, como impresoras, puerto serie, monitores, memorias, otra computadora, etc.

Uso de la memoria de reconocimiento del usuario. La continua visualización de los objetos y acciones estimula el uso de la memoria de reconocimiento y no se necesita memorizar instrucciones puesto que siempre están a la vista.

3.5 Unidad Controladora de Procesos [2]

3.5.1 Descripción de la Unidad Controladora de Procesos

La Unidad Controladora de Procesos es un sistema embebido que utiliza como unidad central de procesamiento un microcontrolador de la serie PIC33F de microchip, el cual es un controlador diseñado para el procesamiento digital de señales. Este sistema tiene la capacidad de controlar una planta, responder a solicitudes de los periféricos, detectar errores y responder a órdenes del usuario. Además es un sistema multitarea, diseñado para aplicaciones de tiempo real y utiliza MicroC/OS-II como sistema operativo.

El sistema fue diseñado siguiendo el modelo que se presenta en la Figura 3.8.

Figura 3.8 Estructura del sistema embebido [2].

En la Figura 3.8 se muestra el diseño del sistema basado en cuatro capas, iniciando en una capa de *hardware*, la siguiente de *drivers*, RTOS y la capa de *Aplicación*. La capa de *hardware* comprende todos los componentes físicos del sistema, tales como: convertidores analógico/digital y digital/analógico, control de motores por QEI y PWM, buses I^2C, SPI, puerto de comunicación UART y *Ethernet* mediante Lantronix XPortAR, entre otros dispositivos, la Figura 3.9 ilustra la capa de *hardware* del sistema.

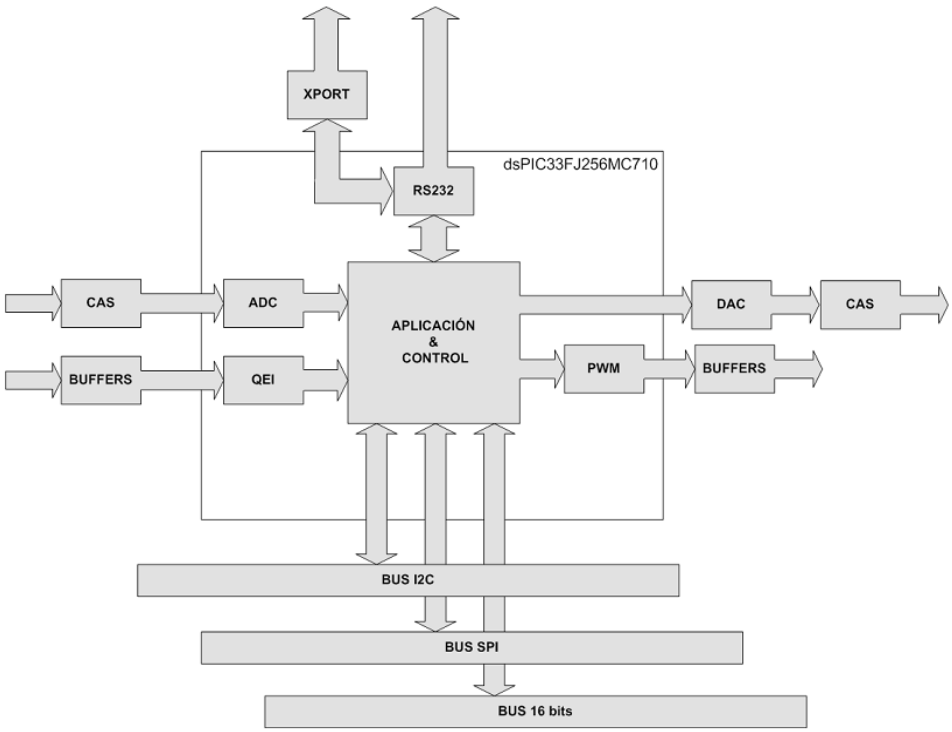

Figura 3.9 Estructura de la capa de *hardware* [2].

En cuanto a la capa de *controladores*, ésta brinda la capacidad de cambiar la configuración del *hardware* sin necesidad de recompilar el *firmware* y consta de cuatro etapas:

Archivo de configuración, el cual contiene la configuración mínima y por defecto requerida para el correcto funcionamiento del sistema y se utiliza como configuración de arranque del sistema. **Instrucciones de configuración**, las cuales se utilizan para modificar la configuración mínima de arranque sin necesidad de recompilar el *firmware*, se guardan en memoria y se ejecutan luego de iniciada la configuración mínima. **Funciones de configuración,** la cuales son funciones que configuran el *hardware* según lo que el usuario desea y tenga a disposición. **Funciones de operación,** son aquellas que realizan operaciones utilizando el *hardware* según sus funciones.

En cuanto a la capa RTOS, el sistema utiliza la versión 2.85 del *kernel* MicroC/OS–II de Micrium, y esta capa está ubicada entre la capa de drivers y la capa de aplicación.

En cuanto a la capa de aplicación, ésta se divide en varias subcapas de funciones, las cuales se ilustran a continuación en la Figura 3.10.

Figura 3.10 Subcapas de la capa de aplicación [2].

Aplicación de inicio de sistema y diagnóstico. Es una secuencia que se ejecuta siempre al inicio con el fin de arrancar el sistema, y cada vez que el sistema es reiniciado.

Aplicación de comunicaciones. Se encarga de permitir la comunicación con el usuario, trata de que dicha comunicación sea confiable y adecuada.

Aplicación de máquina virtual. El sistema brinda la posibilidad de crear bloques funcionales, configurarlos y ejecutarlos, estos bloques funcionales se denominan módulos virtuales, por medio de los cuales el usuario puede crear una estructura de Control Automático, la cual desea que el sistema ejecute. La máquina virtual se encarga de ejecutar los bloques que el usuario haya creado y configurado.

Aplicación de captura de datos. Es una función que se encarga de capturar muestras de una señal analógica o digital, y almacenarlos en memoria con el fin de que el usuario pueda manipular y analizar dicha información.

Aplicación de calibración. Esta aplicación se utiliza para ajustar los rangos de las entradas y salidas analógicas.

Compilador e intérprete del archivo *netlist*. Ésta es una función que se encarga de verificar, interpretar y decodificar el archivo Netlist, además, crea o destruye en la memoria dinámica todos los módulos virtuales. El archivo netlist es un archivo con todos los enlaces de conexión de los módulos. Cuando el usuario envía el archivo netlist, esta función verifica que el éste sea consistente y que no contenga errores de sintaxis. Si el archivo es correcto, se procede a compilar el archivo en un arreglo de punteros a funciones y módulos que posteriormente la tarea de máquina virtual utilizará para ejecutar la estructura de control creada por el usuario [2].

3.5.2 Protocolo de comunicación de la Unidad Controladora de Procesos

El sistema posee su propio protocolo de comunicación, el cual utiliza el usuario para lograr establecer comunicación con el dispositivo. Este protocolo o sistema de comunicación se explica a continuación:

1. El usuario envía o transmite el código 0x53 el cual es el caracter "S" que indica que va a iniciar la transmisión de algún código de instrucción, si la Unidad Controladora está funcionando correctamente, ésta regresa el mismo código 0x53 o caracter "S", si existe algún problema de comunicación el sistema regresa el código 0x45 o caracter "E".
2. Seguidamente el usuario envía dos bytes, el primero indica el código de la instrucción a ejecutar y el segundo el tamaño de la trama de datos correspondiente a esta instrucción, si el código de la instrucción es reconocida por la Unidad Controladora, ésta responde enviando el mismo código de instrucción, si existe algún problema y la instrucción no es reconocida la Unidad Controladora regresa el caracter "E" o código 0x45.
3. Por último el usuario envía la trama de datos, si la Unidad Controladora reconoce la trama sin problema alguno, entonces responde con un caracter "S" o código 0x53, si no logra recibir la trama correctamente, regresa un caracter "E" indicando error.

La Figura 3.11 presenta el diagrama de flujo del protocolo de comunicación de la Unidad Controladora de Procesos.

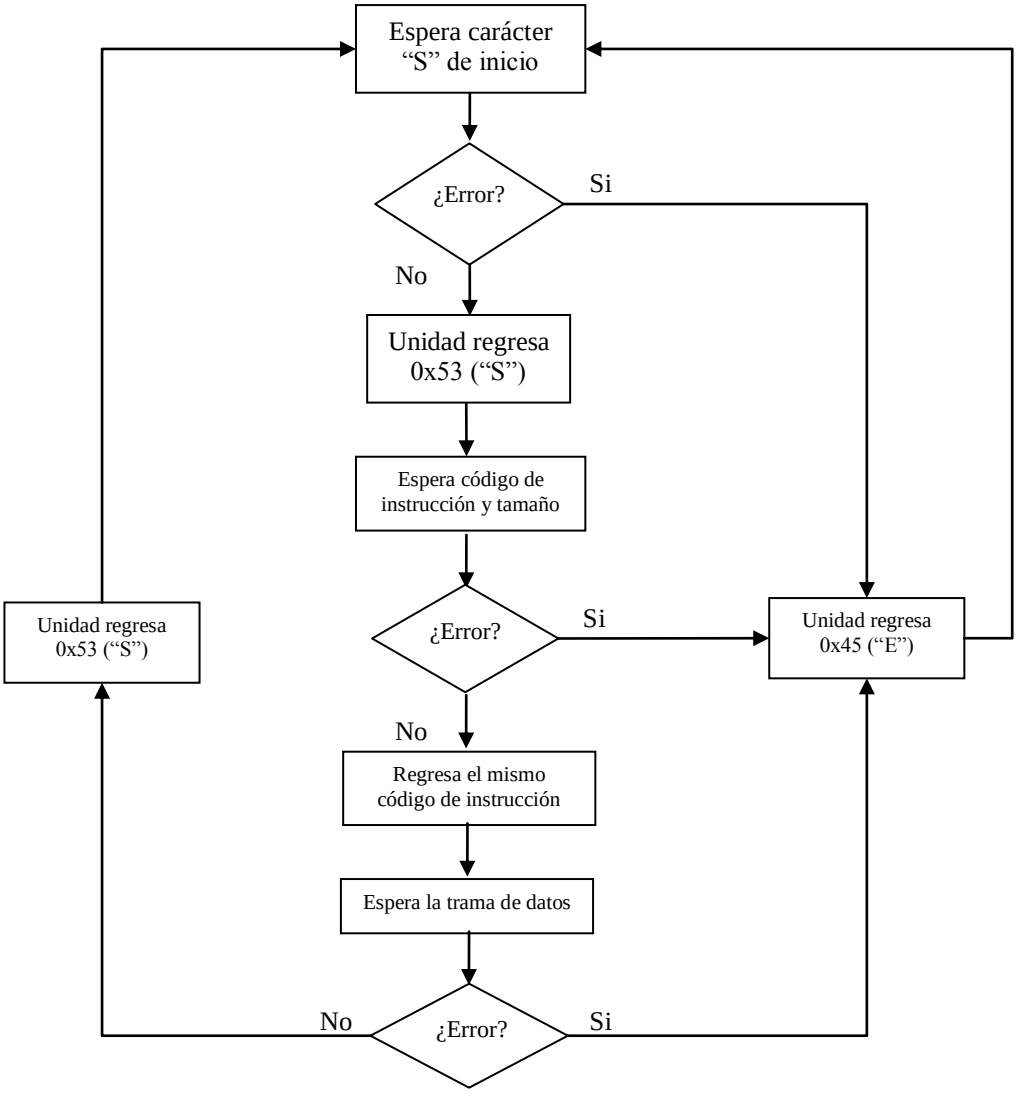

Figura 3.11 Diagrama de flujo del protocolo de comunicación.

Para conocer más detalles de las características y funcionalidades de la Unidad Controladora de Procesos ver [1] y [2].

Capítulo 4: Etapas de diseño del Entorno de Desarrollo Integrado

4.1 Diseño del lenguaje intermedio en mnemónicos

Para la etapa de creación del lenguaje intermedio, se establece que los elementos que lo componen deben representar las instrucciones permitidas por la Unidad Controladora de Procesos, así como la estructura general con la que este sistema es configurado.

No existe una regla general para la selección de los mnemónicos en la creación de un lenguaje; sino, que es el diseñador quien elige si utiliza palabras de un idioma como el español o el inglés con algún significado o regla, o si simplemente utiliza combinaciones de letras al azar. Otro detalle a tomar en cuenta, es que parte de las instrucciones especificadas en el manual del dispositivo ya vienen con un mnemónico que las identifica.

Aspectos de diseño del lenguaje intermedio

Para la etapa de creación del lenguaje intermedio, se decidió seguir los siguientes lineamientos:

- Utilización de mnemónicos mencionados en la documentación de la Unidad Controladora de Procesos con los que se hace referencia a algunas instrucciones.
- Mnemónicos basados en el idioma inglés, ya que la mayoría del vocabulario de sistemas de Control Automático se deriva de este idioma.
- Utilización de combinaciones de palabras o letras para especificar la función de instrucciones que realizan tareas relacionadas a un mismo tema.
  Ejemplo: RANGEADC para configurar el rango de medición del ADC, y READADC para leer el valor puro e instantáneo del convertidor analógico-digital.

La descripción completa de cada uno de los mnemónicos establecidos para el lenguaje intermedio se especifica en el apéndice B.

4.2 Traductor de código intermedio a lenguaje de máquina

Esta etapa consistió en la creación de un sistema, el cual fuera capaz de interpretar un lenguaje de programación codificado en un lenguaje intermedio, analizar dicho lenguaje en busca de los posibles errores léxicos, sintácticos y semánticos para finalmente traducir del lenguaje intermedio al lenguaje de máquina. Para esto se utilizó un metacompilador.

Tabla 4.1 Comparación de los metacompiladores consultados.

Metacompilador	Licencia	Especificaciones léxico-sintácticas	Código en la especificación de la gramática	Lenguaje de salida
Lex&Yacc [14]	*Software* propietario	Separadas	mezclado	C
Flex&Bison [8]	Código abierto	Separadas	mezclado	C
JFlex/Cup [15]	Código abierto	Separadas	mezclado	Java
JavaCC [15]	Código abierto	Un mismo archivo	mezclado	Java

En cuanto a las herramientas de metacompilación disponibles para realizar la labor de traducción, se estudiaron cuatro alternativas que se resumen en la Tabla 4.1. Para la selección del metacompilador se tomaron en consideración los siguientes criterios: costo económico mínimo y facilidad de implementación. El metacompilador seleccionado fue JavaCC, esto debido a que implementa el analizador léxico y sintáctico en un mismo archivo de especificación, mientras que Flex&Bison y JFlex&Cup los implementan por separado, por lo que JavaCC ofrece facilidad de implementación, y sin costo económico por ser una herramienta de código abierto.

4.2.1 Aspectos de diseño del traductor de código intermedio

Para la construcción del traductor fue necesario definir la estructura léxico-sintáctica del programa. La sección anterior definió una parte en cuanto al tema lexicográfico, ya que establece las palabras reservadas del lenguaje, tanto en instrucciones como en parámetros; sin embargo, se especificaron, además de los mnemónicos, algunos símbolos que complementan este componente léxico y que ayudan a especificar la estructura o sintaxis de los programas escritos en dicho lenguaje intermedio.

La máquina virtual es la que establece las normas de programación de las instrucciones, desde la creación de los módulos virtuales, su configuración y la secuencia de ejecución de tales bloques. Estas normas fijan un orden para la programación del dispositivo, primero que todo, el usuario debe crear los módulos virtuales que va a necesitar en la estructura de control que desea implementar, segundo debe especificar las instrucciones de configuración de tales módulos y por último, debe establecer cómo se relacionan estos módulos entre sí por medio de una lista de enlace, denominada *netlist*. Debido a este orden se define una sintaxis general de un programa escrito en el lenguaje intermedio de la siguiente manera:

1. **Crear módulos:** Primera sección del código en la que se crean los módulos que el usuario desea implementar, especificando el tipo de módulo y su cantidad.
2. **Configurar módulos:** Segunda sección del código en la que se especifica la configuración de cada uno de los módulos creados.
3. ***Netlist*:** Tercera y última sección del programa en la que se establece la relación de los módulos entre sí.

Debido a que el lenguaje intermedio es el siguiente escalón dentro de la jerarquía de lenguajes de programación, después del lenguaje de máquina, y tomando en cuenta que el lenguaje de máquina es el de más bajo nivel, los mnemónicos descritos en el apartado anterior tienen una correspondencia uno a uno con el código de lenguaje de máquina, esto quiere decir que el mnemónico READADC corresponde al código hexadecimal 0x21 y su parámetro ya sea adc0, adc1, adc2 o adc3 corresponde a 0x00, 0x01, 0x02, 0x03

respectivamente, en otras palabras, la instrucción **READADC adc0;** en lenguaje intermedio corresponde al código **0x21 0x00** en lenguaje de máquina. Debido a estas características, se define que el lenguaje intermedio es de tipo ensamblador, por lo que parte de su sintaxis se estableció de manera similar. Una de sus similitudes es que se escriben las instrucciones y sus parámetros separados por comas ",", un detalle adherido a esto es la utilización de un punto y coma ";" para indicar la finalización de la instrucción, por ejemplo:

SOSC 0, 1.0000, 0.5000;

donde esta instrucción indica la configuración del oscilador sinusoidal número 0, con frecuencia de oscilación de 1 Hz y de amplitud 0.5. El equivalente hexadecimal para esta instrucción de ejemplo sería:

26 00 312E30303030 302E35303030

Creación de módulos

Ésta es la sección del programa que la máquina virtual del dispositivo establece como el primer paso, ya que la máquina virtual no puede configurar un módulo si éste no ha sido definido o creado por el usuario.

En cuanto a la sintaxis que se definió para esta sección, ésta se describe de la siguiente manera:

```
MOD
begin
        Módulo, cantidad;
end
```

donde se tiene la instrucción **MOD** para indicar que se trata de la sección de crear módulos, sumado a esto se agregaron los términos **begin** y **end** para indicar el inicio y final de sección, cabe destacar que **begin** y **end** no tienen un código equivalente en lenguaje de máquina, solamente se agregaron para establecer orden y claridad a la hora de escribir el

programa. Dentro de esta sección identificada por la instrucción **MOD** y delimitada por los términos **begin** y **end**, se especifican los módulos a crear utilizando los mnemónicos correspondientes y especificados en las tablas B.1, B.2, B.3, B.4, B.5, B.6 y B.7 de la sección de Apéndices, luego separado por coma, la cantidad de módulos a crear y por último un punto y coma.

Un ejemplo en el que se crean un oscilador sinusoidal, un módulo de saturación, un filtro de tipo iir2 y dos módulos de suma se presenta a continuación:

```
MOD
begin
        sosc, 1;
        sat, 1;
        iir2, 1;
        sum, 2;
end
```

Código en lenguaje de máquina equivalente: **31 09 04 4b 01 4a 01 45 01 49 01**

Los módulos de *hardware* especificados en la tabla B.1 y B.2 no se deben crear puesto que son dispositivos que ya existen.

Configuración de módulos

La sección de configuración de módulos sigue las mismas normas descritas para la creación de módulos. Se escribe un término que identifica a la sección de configuración de módulos y delimitada nuevamente por **begin** y **end**, el término que se escribe para identificar esta sección es **CONFIG**, y al igual que **begin** y **end**, este término no posee correspondiente código hexadecimal en el lenguaje de máquina, solamente se agregó para establecer congruencia con las demás secciones del programa, mientras que los mnemónicos **MOD** y **NETLIST**, si tienen un correspondiente código hexadecimal en el lenguaje de máquina.

Ejemplo:

```
CONFIG
begin
        Instrucciones;
end
```

Dentro de los delimitadores **begin** y **end** de esta sección, se escriben todas las instrucciones que tiene que ver con la configuración de los módulos creados, ejemplo: **RANGEADC**, **SAT**, **FILTER**, **SOSC**, mientras que las instrucciones que no tiene que ver con la configuración de un módulo virtual creado o de *hardware* no se escriben en esta sección, pues se podría generar un comportamiento erróneo e inesperado del que se requiere. Cabe resaltar que muchas de las instrucciones tienen el mismo mnemónico que algunos parámetros, por ejemplo la instrucción **GAIN** y el parámetro **gain**, para lo cual se estableció la norma de que si el mnemónico se utiliza como una instrucción, se debe escribir en mayúscula y si es un parámetro se debe escribir en minúscula. El traductor hace diferencia entre mayúsculas y minúsculas, por lo tanto si un término **begin** se escribe en mayúscula el traductor reportará un error.

Un ejemplo en el que se configura un módulo de saturación **sat**, el rango de medición del módulo **adc** y su tiempo de muestreo, un filtro tipo **iir2**, un valor en el registro **rgp**, rango de salida del módulo **dac**, la frecuencia y amplitud del módulo oscilador sinusoidal **osc** se presenta a continuación:

```
CONFIG
begin
        SAT 0, 0.8000, -0.800;
         RANGEADC adc0, 2, 1;
         FILTER iir_2, 0, 2, 1, bessellp;
         RGP 0, -0.03;
         RANGEDAC dac0, 2;
         SOSC 0, 1, 0.5;
end
```

Código en lenguaje de máquina equivalente:

```
33  0d 00 302e38303030 2d302e383030
1a  04 00 02 0100
```

21 0f 00 00 303030303032 303030303031 02
55 07 00 2d302e303330
1b 02 00 02
26 0d 00 303030303031 303030302e35

Netlist

Esta sección es la que determina la relación que existe entre los módulos. Es donde se especifica cuál salida y de cuál módulo está conectada a cuál entrada y de cuál otro módulo. Esta sección también lleva de encabezado un mnemónico que la identifica, el cual se denomina **NETLIST**, así como sus respectivos delimitadores **begin** y **end**.

Ejemplo:

```
NETLIST
begin
        Parámetros;
end
```

Dentro de los delimitadores **begin** y **end**, se escriben de izquierda a derecha y separados por coma el módulo cuya salida va conectada a la entrada de otro módulo de salida, su número de bloque, su número de salida, el módulo al cual va conectado, su número de bloque, su número entrada y finalmente un punto y coma, y así las siguientes conexiones que sean necesarias, ejemplo:

```
NETLIST
begin
        sat, 0, 1, dac, 0, 1;
        adc, 0, 1, iir2, 0, 1;
        iir2, 0, 1, sum, 0, 1;
        rgp, 0, 1, sum, 0, 2;
end
```

Código en lenguaje de máquina equivalente:

16 18 4a 00 01 c0 00 01 01 00 01 45 00 01 45 00 01 49 00 01 02 00 01 49 00 02

En general un programa escrito en el lenguaje intermedio diseñado para este proyecto, posee tres secciones. Aparte de esto, el programa tiene normas y símbolos propios que ayudan a dar orden y claridad a los programas. Entre estos símbolos, falta mencionar los que indican comentarios, estos se presentan a continuación:

//: Símbolo de comentario para una sola línea.
/*…*/: Símbolo de comentario para varias líneas.

Ejemplo de un programa completo:

```
//Cometarios 1
/*
Comentarios 2
*/
MOD  //sección de crear módulos
begin
        sosc, 1;
        sat, 1;
        sum, 2;
end
CONFIG //sección de configurar módulos
begin
        SAT 0, 0.8000, -0.800;
        RANGEADC adc0, 2, 1;
        RGP 0, -0.03;
        RANGEDAC dac0, 2;
        SOSC 0, 1, 0.5;
end
NETLIST  //sección donde se especifica las conexiones entre los módulos
begin
        sat, 0, 1, dac, 0, 1;
        adc, 0, 1, sum, 0, 1;
        rgp, 0, 1, sum, 0, 2;
        sum, 0, 1, wgpr, 1, 1;
        sosc, 0, 1, sat, 0, 1;
end
```

4.2.2 Implementación del lenguaje intermedio en JavaCC

Para crear el traductor del lenguaje intermedio con la estructura y sintaxis descrita anteriormente se utilizó la herramienta de compilación JavaCC. Gracias a la característica que posee JavaCC de incluir el analizador léxico y sintáctico en una misma herramienta, se logró describir el programa en un mismo archivo, lo que facilitó su implementación. Este archivo es de extensión *.jj y se llama **compilador.jj**, en este archivo se definieron las palabras reservadas, las expresiones regulares del lenguaje, su estructura general y los símbolos que representan los comentarios, así también se definieron las acciones a tomar en cuanto a traducción de código cada vez que se reconozca un lexema o expresión.

Al compilar el archivo **compilador.jj**, JavaCC genera los siguientes archivos:

- **Compilador.java**
- **CompiladorTokenManager.java**
- **CompiladorConstants.java**
- **ParseException.java**
- **SimpleCharStream.java**
- **Token.java**
- **TokenMagrError.java**

Este conjunto de archivos generados en el código fuente del programa componen el traductor de código intermedio, siendo **Compilador.java** la clase principal.

Debido a que JavaCC posee gestión de errores, se tiene un traductor que indica la línea y la columna donde ocurrió el error, debido a escribir mal una instrucción o parámetro, y también por la falta de algún elemento como una coma, punto y coma, la falta de términos obligatorios como **MOD**, **CONFIG**, **NETLIST**, **begin**, **end,** e incluso la falta de toda una sección del programa.

El lenguaje intermedio resultante no permite la creación de términos variables, ni establece tipos para los elementos del lenguaje como normalmente se tiene en lenguajes comunes

como C, C++ y Java, ya que éstos si permiten definir tipos de datos, variables o parámetros como **integer**, **String**, **Boolean**, **Float**, etc.

La definición del lenguaje se puede encontrar en el apéndice E y está escrito en notación ***Backus-Naur form*** o **BNF**.

4.3 Diseño de la Interfaz gráfica de usuario

Para el diseño de la interfaz gráfica, se partió de considerar que la Unidad Controladora de Procesos permite la utilización de módulos, configuración y conectividad entre ellos. El tipo de interacción seleccionado fue mediante la manipulación directa y gráfica de bloques que representan a los módulos de dicha unidad electrónica. De igual manera se diseñó el sistema para que estos bloques fueran configurables con las opciones determinadas por el dispositivo, y además, que se pudieran conectar entre sí.

4.3.1 Aspectos de diseño de la interfaz gráfica

La interfaz gráfica de usuario requiere las siguientes características:

- Brindar funcionalidades típicas de un programa de escritorio, tales como abrir y guardar archivo.
- Utilizar el método de manipulación directa de componentes gráficos sobre un área de trabajo, realizar conexiones y configurar sus propiedades.
- Permitir la posibilidad de simular, graficar datos y programar la Unidad Controladora.
- Ser accesible remotamente por medio de protocolo TCP/IP.

Para lograr esta características, fue necesaria la selección de un lenguaje de programación que permitiera la manipulación de imágenes y gráficos en 2D, así como el desarrollo de interfaces gráficas de usuario; la programación de sistemas *Web* y de red, facilidad de implementación, buena documentación, multiplataforma, variedad de tecnologías y bajo licencia de *software* libre. Debido a esto, se analizaron algunas de las tecnologías más utilizadas para el desarrollo de interfaces gráficas de usuario, tales tecnologías fueron las basadas en los lenguajes de programación Python, .NET Framework de Microsoft y Java de Sun Microsystems. La Tabla 4.2 presenta un resumen comparativo entre las distintas tecnologías.

Tabla 4.2 Comparación de los lenguajes de programación consultados.

Lenguaje de Programación	Gráficos 2D y 3D	Interfaces gráficas de usuario	Aplicaciones Web	Soporte de sistemas de red	Multi-plataforma	Orientado a objetos
Python [17]	Si	Si	Si	Si	Si	Si
Tecnologías .NET Framework [13]	Si	Si	Si	Si	No (solo Windows)	Si
Java [3]	Si	Si	Si	Si	Si	Si

Considerando los criterios antes mencionados se seleccionó Java como la tecnología de desarrollo de la interfaz gráfica enfocada en la utilización de Java Applets, ya que permite integrar una aplicación en un navegador de Internet, lo que le brinda la posibilidad al usuario de acceder a la aplicación de manera remota, y sin preocuparse por el sistema operativo que esté utilizando.

4.3.2 Implementación de la interfaz gráfica

La implementación de la interfaz gráfica se realizó de manera similar a un sistema simulador de circuitos, con una barra de herramientas donde se encuentran todas las funciones comunes del manejo de archivos como nuevo, abrir, guardar; funciones de edición como copiar, pegar, cortar, deshacer; funciones de ejecución de la Unidad Controladora de procesos como programar dispositivo, iniciar, pausa, detener, reiniciar; funciones de graficar como la de capturar datos, simulación, osciloscopio flotante; otros como líneas de conexión, conectar y desconectar el dispositivo; herramientas de configuración de IP, compilar a texto, y una interfaz que invoca al compilador o traductor del lenguaje intermedio al lenguaje de máquina. Las figuras 4.1 y 4.2 muestran la interfaz gráfica diseñada y su herramienta de traducción de código intermedio respectivamente.

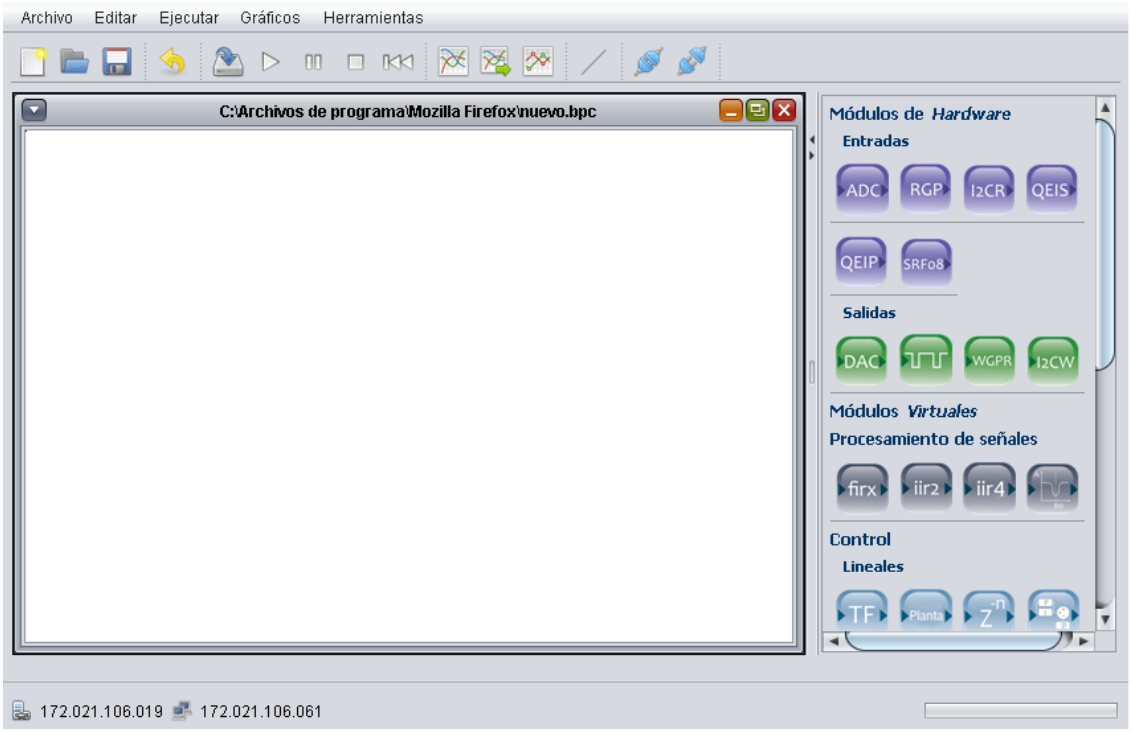

Figura 4.1 Interfaz gráfica del Entorno de desarrollo Integrado.

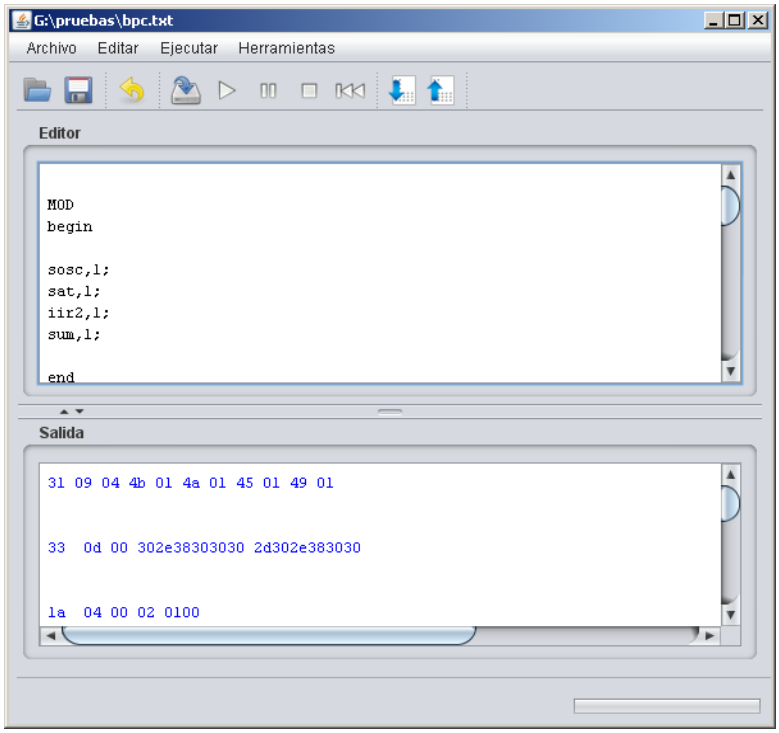

Figura 4.2 Interfaz gráfica del traductor de código intermedio.

La interfaz se creó para brindarle al usuario la capacidad de realizar todas las funciones de tres maneras: por medio de menús en la barra de menús en la parte superior, utilizando el teclado con el uso de *hotkeys* y por medio de botones en la barra de herramientas. Ver figura 4.3 y 4.4.

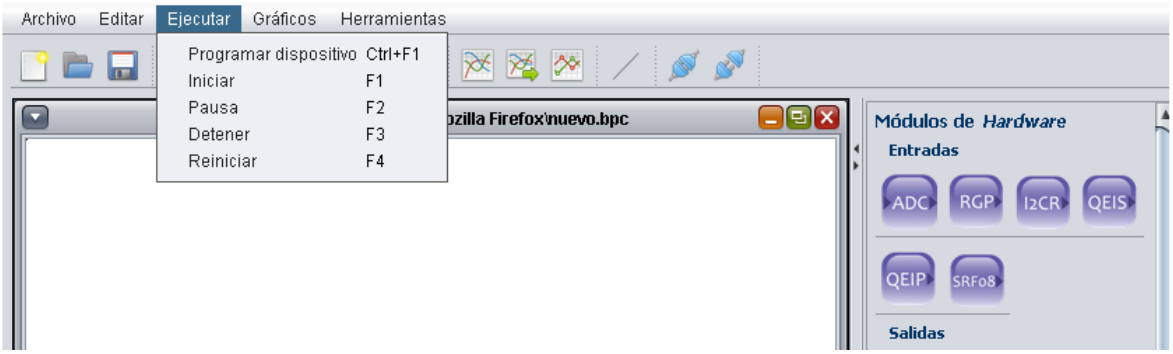

Figura 4.3 Uso de menús para realizar las funciones.

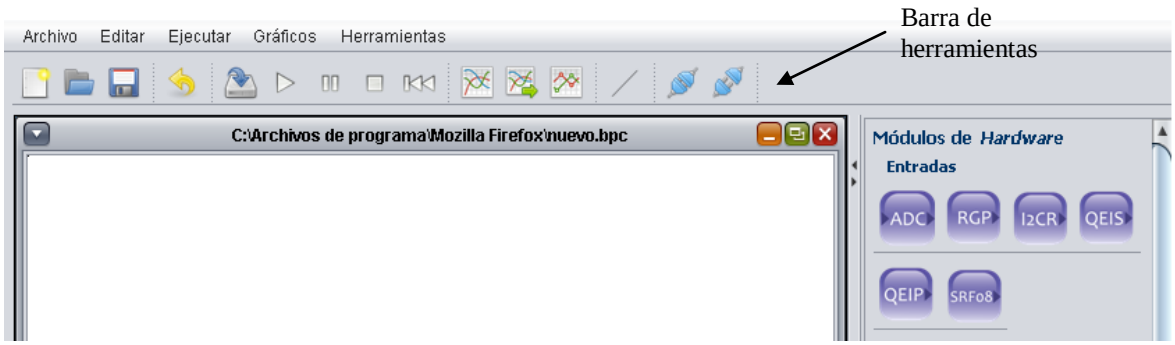

Figura 4.4 Uso de botones en la barra de herramientas para realizar funciones.

Por otra parte, se diseñó un área de la interfaz que funciona como una caja de herramientas o *Toolbox*, y es donde se encuentran los módulos de *hardware*, de entrada y salida, así como los módulos virtuales disponibles para programar y configurar en la Unidad Controladora. Ver figura 4.5.

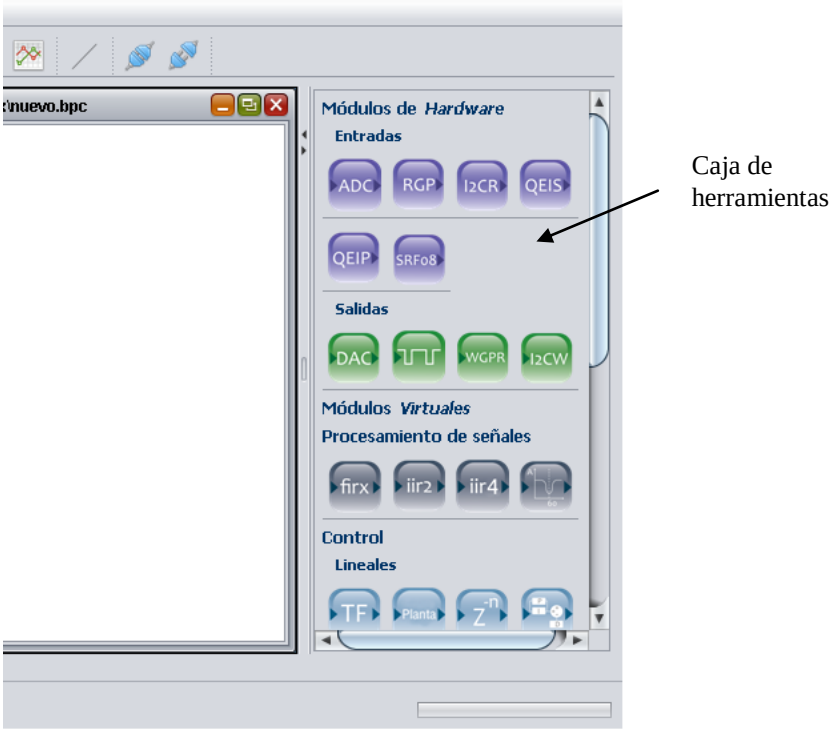

Figura 4.5 Caja de herramientas o *Toolbox*.

Una tercera parte de la interfaz de usuario es el área de trabajo o *workarea*, en la cual se colocan los elementos de la caja de herramientas, para así poder trabajar con ellos en el diseño de alguna estructura de control. En esta área de trabajo, los módulos se pueden conectar por medio de sus entradas y salidas, utilizando una línea de conexión que aparece de manera automática al acercar el puntero del *mouse* a un eje de conexión. La interfaz es capaz de manejar errores comunes, por ejemplo al tratar de conectar dos entradas o dos salidas, en cuyo caso el sistema lanza un mensaje de error como se muestra en la figura 4.6. Los errores que este sistema puede manejar se especifican en el apéndice D.

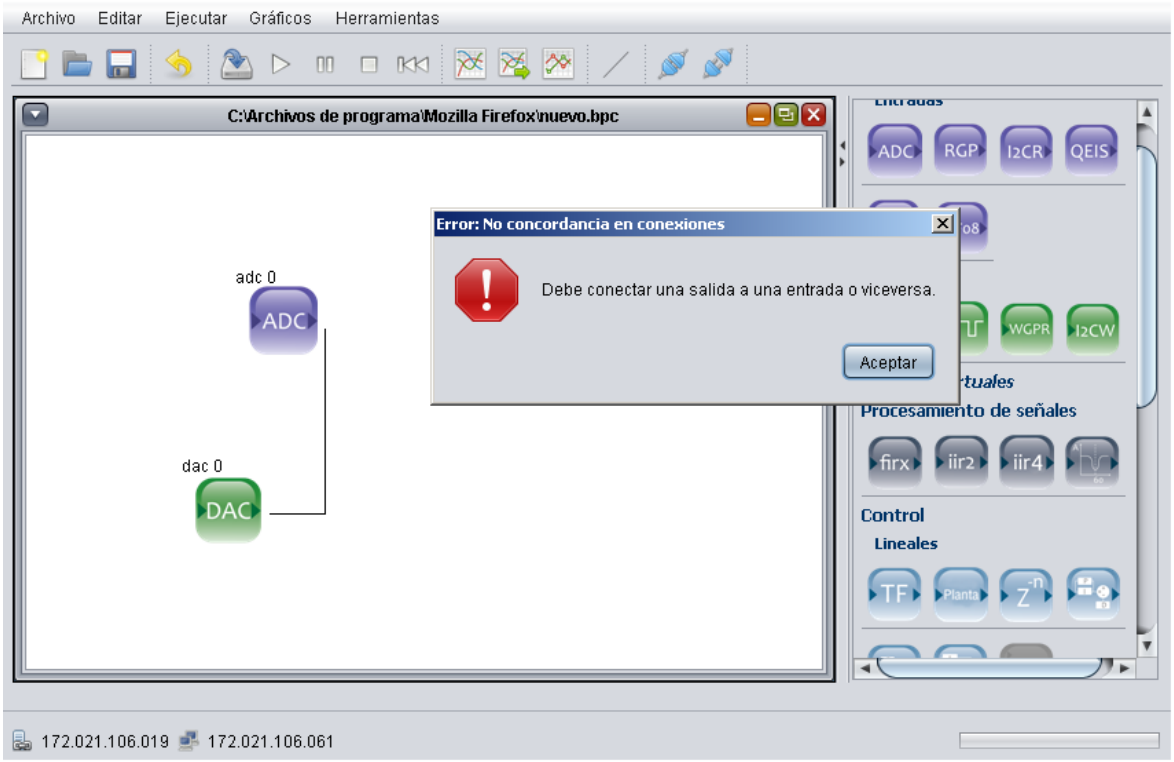

Figura 4.6 Manejo de errores durante la conexión de bloques.

El despliegue gráfico de bloques y líneas de conexión, así como del área de trabajo se diseñó utilizando la API Java 2D [4] la cual es un conjunto de funciones para el diseño de gráficos en dos dimensiones.

El diseño de botones, menús, barra de herramientas, paneles, y demás se crearon utilizando las funciones comunes que ofrece Java para el diseño de interfaces gráficas. Por otra parte los iconos utilizados en la barra de herramientas se tomaron de [5] y [19].

Internamente el sistema maneja dos estructuras principales de tipo Vector, el cual es un objeto de Java que tiene características dinámicas. Una de estas estructuras fue utilizada para el manejo de los componentes gráficos en el área de trabajo y la otra para las líneas de conexión entre módulos.

En cuanto a los sistemas de despliegue de datos como los de captura, simulación y el osciloscopio flotante se utilizaron las versiones jcommon-1.0.16.jar y jfreechart-1.0.13.jar

de la biblioteca **JFreeChart** disponible de manera gratuita para la creación de gráficos. Estas funciones fueron implementadas en los siguientes archivos del código fuente:

- **Graficos.java:** para la función de captura de datos.
- **Osciloscopio.java:** para visualización de señales en tiempo real.
- **Simulador.java:** para la simulación utilizando GNU Octave.

Ver figura 4.7 y 4.8.

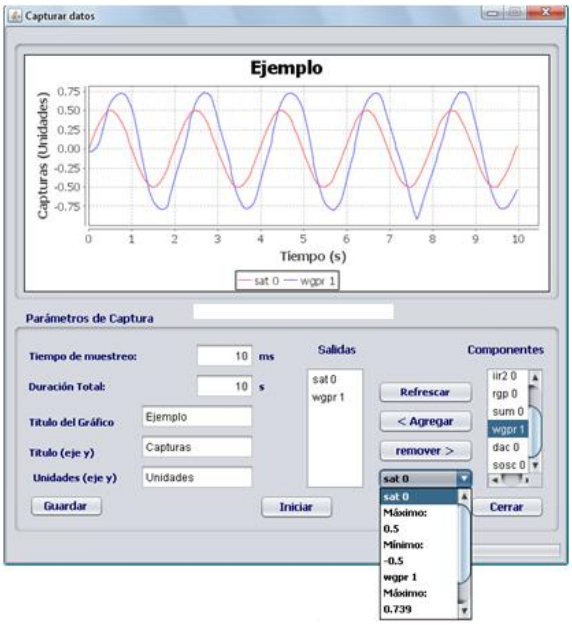

Figura 4.7 Resultados de captura visualizados.

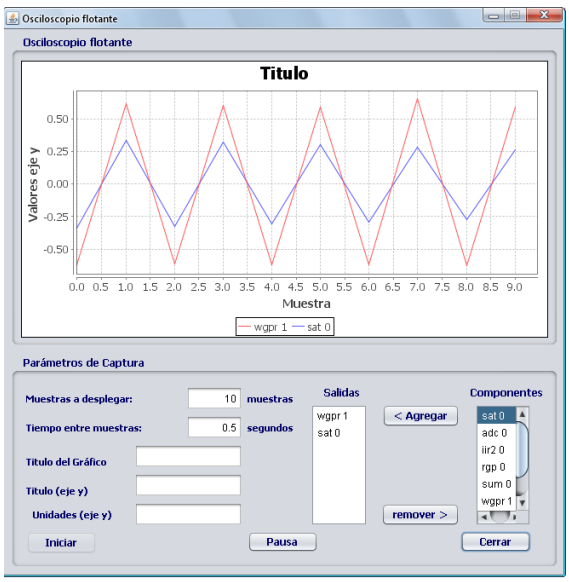

Figura 4.8 Osciloscopio flotante para visualizar señales en tiempo real.

La interfaz gráfica está contenida básicamente en dos archivos principales en el código fuente: **Principal.java** y **PanelComponentes.java**, el primero implementa todo lo referente a la interfaz de usuario como la barra de menús, menús, barra de herramientas, botones, área de trabajo y caja de herramientas, mientras que el segundo archivo mencionado, implementa todo lo referente al despliegue gráfico de las figuras u objetos manipulables por el usuario en el área de trabajo, así como las acciones de arrastrar, conectar, seleccionar, clic izquierdo, doble clic y clic derecho sobre los objetos desplegados a manipular.

El sistema general, se compone, aparte de la interfaz gráfica, de dos programas internos, uno de ellos se encarga de la trasmisión y recepción de datos vía TCP/IP, este programa consta de dos archivos: **Tcpip.java** y **Tcpip_serie.java**, ambos archivos conforman el programa de transmisión y recepción de datos con la Unidad Controladora de procesos. El otro programa a mencionar es un sistema basado en la tecnología Java RMI o *Remote Method Invocation*, la cual es una tecnología similar a RPC o *Remote Procedure Calls*.

Java RMI permite la comunicación remota de programas Java que se encuentran funcionando en distintas computadoras, como si estos programas formaran parte de un mismo sistema en la misma computadora. Este sistema funciona por medio de una configuración cliente-servidor, y fue la manera en la que se implementó la herramienta de simulación que más adelante se explicará.

4.4 Diseño del Generador de código intermedio

El generador de código intermedio debe tomar la información contenida en la estructura de datos gráfica diseñada; y generar a partir de ella las tres secciones de código que componen la estructura del programa en lenguaje intermedio y escribir el código resultante a un archivo de texto.

4.4.1 Aspectos de diseño de generador de código intermedio.

El generador de código intermedio requiere cumplir los siguientes requisitos:

- El proceso de generación de código intermedio debe ser transparente para el usuario.
- Debe representar la estructura gráfica en código intermedio según su gramática.
- Debe generar un archivo de texto con el código intermedio requerido.

4.4.2 Implementación del generador de código intermedio.

El generador de código intermedio se implementó de manera implícita en la interfaz, es una rutina que se invoca cada vez que el usuario desee programar el dispositivo electrónico, o cuando desea compilar a texto. El usuario con la interfaz del traductor de código intermedio tiene la posibilidad de traducir dicho código a lenguaje de máquina que se escribe en un archivo *.hex, y con la función de "Programar dispositivo" puede descargar la aplicación en la Unidad Controladora de Procesos.

Existe otra alternativa para programar el dispositivo, la cual es utilizando la función "Programar dispositivo" que brinda la interfaz principal de usuario, este método realiza las operaciones de verificar errores, traducir a código intermedio, ejecutar el traductor de código intermedio y programar el dispositivo de manera automática. Luego mediante las funciones de iniciar, pausar, detener y reiniciar, el usuario puede verificar si el comportamiento de la planta física responde a la estructura programada.

El archivo *.txt en el cual se genera el código intermedio, es nombrado por la interfaz como bpc.txt, y el archivo *.hex como bpc.hex. Realmente estos archivos son transparentes para el usuario, son temporales, por lo que el usuario no puede asignarles un nombre, pero en el caso de que requiera analizarlos, la interfaz indica por medio de un mensaje de información el directorio donde se crean tales archivos.

El generador de código intermedio, es un conjunto de subrutinas programadas en el archivo de código fuente llamado **PanelComponentes.java.**

4.5 Implementación de la herramienta de Simulación

Esta etapa se refiere al diseño de una herramienta de simulación como parte de las funcionalidades del Entorno de Desarrollo Integrado. Esta debe ser simular de manera precisa y confiable la mayor cantidad de módulos disponibles en la Unidad Controladora de Procesos y ser de fácil utilización.

4.5.1 Aspectos de diseño de la herramienta de simulación

La herramienta de simulación debe cumplir los siguientes requisitos:

- Facilidad de selección de las señales a simular.
- Confiabilidad en los resultados.
- Maximizar la cantidad de funciones a simular.
- Bajo licencia de código abierto.
- Posibilidad de guardar el código de simulación y los resultados.
- Ejecución remota.

La Tabla 4.3 contiene un resumen comparativo entre los programas de cálculo numérico consultados como posibles alternativas a utilizar en el diseño de la herramienta de simulación.

Tabla 4.3 Comparación de lenguajes de programación de alto nivel orientados al cálculo numérico.

Herramienta de Simulación	**Licencia**	**Interfaz gráfica**	**Funciones requeridas (Teoría de control y procesamiento de señales)**	**Lenguaje**
MATLAB [22]	Propiedad de *The Mathworks*	Si	Si	MATLAB
LabVIEW [16]	Propiedad de *National Instruments*	Si	Si	Lenguaje G
Scilab [9]	*Software* libre	Si	Si	Similar a MATLAB
FreeMat [6]	*Software* libre	Si	insuficientes	Similar a MATLAB
GNU Octave [24]	*Software* libre	No	Si	Similar a MATLAB

Se seleccionó GNU Octave como motor de simulación, debido a que éste no posee interfaz gráfica, sino que el usuario interactúa en una línea de instrucciones o intérprete de instrucciones y por ello, se puede manipular mejor desde Java que los programas que poseen interfaz gráfica. Además es distribuido bajo licencia de *software* libre, su documentación es amplia y posee todas las funciones necesarias para simular la mayoría operaciones de la unidad electrónica.

4.5.2 Implementación de la herramienta de simulación

Para la implementación de la herramienta de simulación, se utilizó el programa GNU Octave como motor de simulación remota, esto minimiza los requisitos del computador del usuario. Se realizó implementando el sistema de invocación remota mediante Java RMI [4] lo cual permite que el sistema de interfaz residente en el cliente interactúe con el programa de ejecución de las instrucciones para GNU Octave de manera remota, con el único requisito de que ambos sistemas deben estar en una red de área local o por medio de una red privada virtual si se quiere acceder al sistema desde Internet.

Una vez establecido el enlace entre la interfaz gráfica, la cual es un Java Applet descargado en el cliente, y la aplicación java que ejecuta instrucciones en GNU Octave por medio de Java RMI, el usuario puede realizar una simulación basada en la estructura que previamente ha diseñado. Para realizar esto la interfaz posee un generador de código GNU Octave, el cual toma la estructura de datos que contiene a los módulos gráficos y líneas, y por medio del algoritmo de ordenamiento topológico implementado en la Unidad Controladora de Procesos [2] el sistema determina el orden correcto de ejecución de los elementos y genera el código GNU Octave equivalente. Este código generado es trasmitido por medio de la interacción cliente-servidor Java RMI, la aplicación **rmi_servidor.jar** ejecuta las instrucciones en código GNU Octave en ese mismo programa, redirige su salida y trasmite los resultados de la simulación al cliente para que la interfaz grafique los resultados y el usuario pueda visualizarlos.

La Figura 4.9 ilustra la interacción cliente-servidor Java RMI para realizar la simulación por medio de la ejecución remota de GNU Octave.

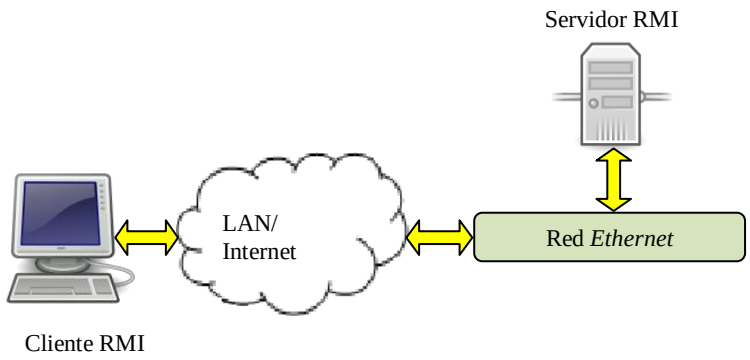

Figura 4.9 Sistema cliente-servidor para realizar los cálculos de simulación de manera remota.

La herramienta de simulación le brinda al usuario la posibilidad de guardar los resultados, y además de obtener el código GNU Octave por si desea simular las aplicaciones de forma independiente.

La Figura 4.10 muestra los resultados de una simulación realizada utilizando la herramienta de simulación de la interfaz gráfica.

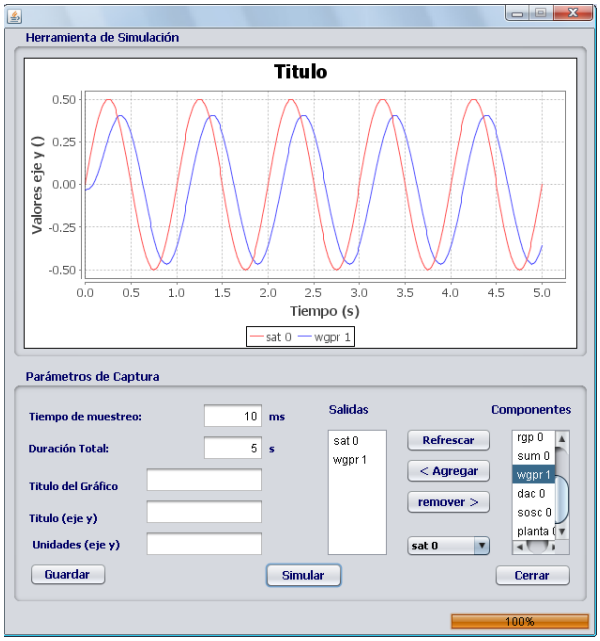

Figura 4.10 Resultados obtenidos por la herramienta de simulación**.**

4.6 Diagrama del sistema general y sus componentes

El sistema general se compone de la interacción del Entorno de Desarrollo Integrado descrito en este proyecto y la Unidad Controladora de Procesos. Este sistema trabaja en una configuración cliente-servidor, donde la Unidad Controladora se comporta como un sistema servidor el cual es configurado y controlado por parte del cliente de manera remota, existiendo de por medio algunos otros elementos que colaboran para establecer una correcta interacción entre el cliente y la Unidad Controladora. Tales elementos que colaboran para dicha interacción cliente servidor son: un servidor HTTP y un servidor de interfaz Java RMI (*Remote Method Invocation*). Cabe destacar además, que el sistema cliente es la entrada para el usuario para interactuar con el sistema servidor, cuya salida pueden ser: a) los resultados de una simulación mostrados en pantalla, b) la aplicación de una estructura de control sobre un sistema real, y c) sus datos capturados y visualizados en pantalla.

Físicamente la Unidad Controladora, entiéndase el servidor, se encuentra conectada a una red de área local, mientras que el cliente podría estar conectado en la misma red de área local teniendo acceso directo al servidor, en una red vecina cuyo acceso al servidor se fija por medio de permisos establecidos por el administrador de red o por medio de internet. La Figura 4.11 ilustra la topología del sistema general cliente-servidor.

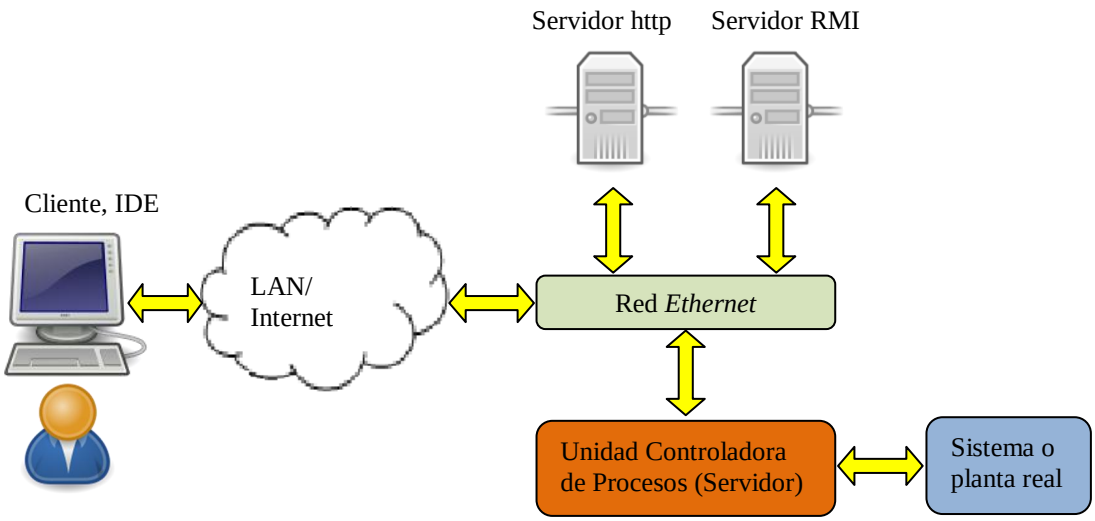

Figura 4.11 Diagrama general del Entorno de Desarrollo Integrado (IDE).

En general el Entorno de Desarrollo Integrado se compone de un conjunto de subprogramas. La estructura de este sistema se ilustra en La Figura 4.12.

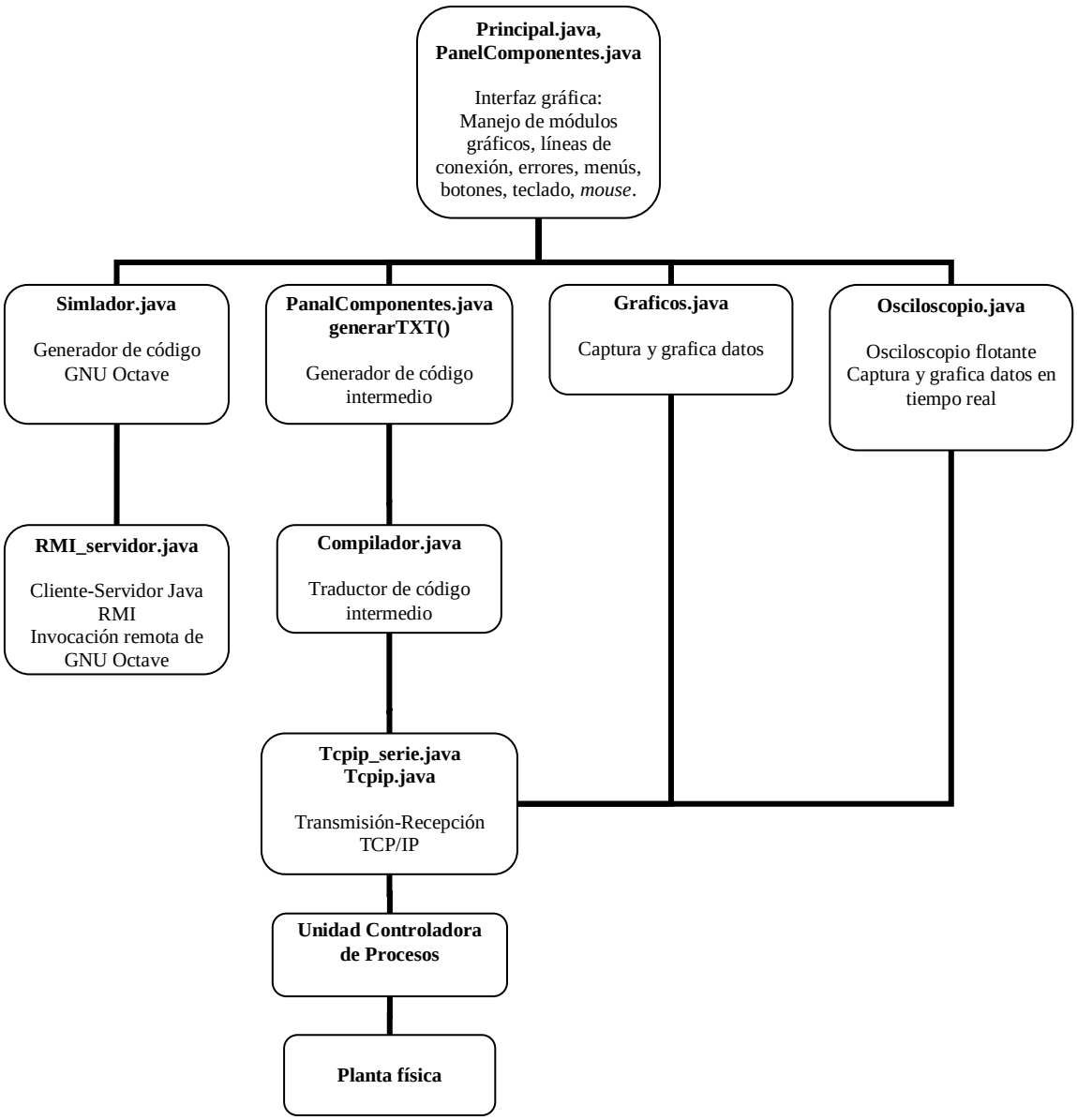

Figura 4.12 Estructura general del Entorno de Desarrollo Integrado.

4.7 Reevaluación y rediseño

Existen otras alternativas que podrían ser tomadas en cuenta en el futuro para realizar mejoras o rediseñar la interfaz gráfica, las cuales podrían ser la utilización de tecnologías como Flash de Adobe Systems Inc., Silverlight de Microsoft o Java FX de Sun Microsystems. Tales tecnologías proveen capacidades para el desarrollo de gráficos y orientadas a entornos Web.

El entorno de desarrollo integrado fue diseñado específicamente para la Unidad Controladora de Procesos, por lo que se podría considerar rediseñar el sistema para que sea independiente del *hardware* o dispositivo electrónico de control.

Otro aspecto sería hacer de la herramienta de simulación un sistema multiusuario, ya que este sistema solo permite su manipulación por un usuario a la vez.

Capítulo 5: Análisis y resultados del Entorno de Desarrollo Integrado

El proceso de diseño del Entorno de Desarrollo Integrado se basó en cinco etapas. La primera etapa consistió en la definición de los mnemónicos de lenguaje intermedio que representan cada una de las instrucciones del lenguaje de máquina. Como resultado, se cubrió el 100% de las instrucciones del lenguaje de máquina por medio de nemónicos de lenguaje intermedio. La Tabla B.1 del apéndice B contiene el total de instrucciones que soporta la Unidad Controladora de Procesos y sus mnemónicos correspondientes. La creación de este lenguaje intermedio facilitó la comprensión de las funciones de la Unidad Controladora de Procesos y el desarrollo de la segunda etapa de diseño, la cual se analizará en los siguientes párrafos.

La segunda etapa de diseño fue la creación de un traductor de código intermedio a lenguaje de máquina. El sistema resultante es similar a un lenguaje ensamblador y es capaz de detectar los siguientes tipos de error:

1. **Errores léxicos:**
   - Palabra reservada mal escrita o que no pertenece al conjunto de mnemónicos del lenguaje.

2. **Errores de sintaxis:**
   - Ausencia de un elemento en una expresión regular.
   - Ausencia de elementos obligatorios en el código como las palabras **MOD**, **CONFIG** y **NETLIST**, **begin** y **end**.
   - Ausencia de secciones completas, como las referentes a **MOD**, **CONFIG** y **NETLIST.**

3. **Errores de semántica:**
   - Palabra o símbolo perteneciente al lenguaje dentro de una expresión regular en la cual no debe ir, como al escribir un módulo en mayúscula en la sección MOD o NETLIST, ya que un módulo escrito en mayúscula se utiliza para indicar una instrucción de

configuración que debe ir en la sección CONFIG, mientras que un módulo en minúscula indica un parámetro que debe estar en las secciones MOD y NETLIST.

Ejemplo de error de semántica:

NETLIST
begin
sat, 0, 1, dac, 0, 1;
ADC, 0, 1, iir2, 0, 1; **//Error de semántica**
end

La gramática del lenguaje intermedio especificada en el traductor de código, no permite la creación de términos variables, ni establece tipos para los elementos del lenguaje que normalmente se tienen en lenguajes como C, C++, Java, etc. Los cuales definen tipos de datos, variables o parámetros como **integer**, **String**, **Boolean**, **Float**, etc.

Por otra parte, la implementación del traductor de código intermedio facilitó la depuración durante el proceso de diseño de la interfaz gráfica. Este sistema permite verificar si los módulos son creados, configurados y conectados correctamente según establece la Unidad Controladora de Procesos. Además, el traductor de código intermedio es capaz de indicar la ubicación exacta donde se genera el error.

En cuanto a la tercera etapa de diseño, sobre el desarrollo de la interfaz gráfica, se obtuvo como resultado, un total de 30 representaciones gráficas para los 30 módulos virtuales y de *hardware* requeridos para ser visualizados y manipulados por el usuario.

Para comparar el tiempo de desarrollo de aplicaciones utilizando la interfaz gráfica de usuario con el tiempo de desarrollo sin la interfaz, se implementaron dos aplicaciones por dos usuarios expertos. Se aclara que este experimento se realizó solamente con dos usuarios, debido a la falta de usuarios expertos en ambos métodos de interacción. Las aplicaciones implementadas se ilustran en las figuras 5.1 y 5.2.

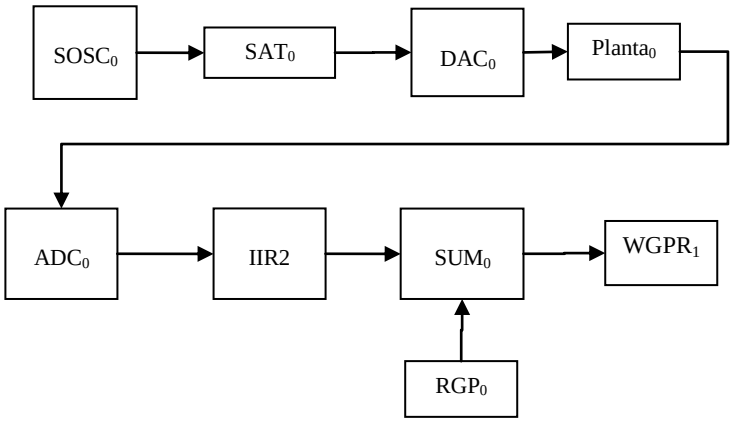

Figura 5.1 Diagrama de bloques de una excitación sinusoidal sobre una planta.

Especificación de las propiedades de los bloques para la estructura de la Figura 5.1.

- SOSC: frecuencia 1Hz, amplitud 1.
- SAT: límite superior 0.8000, límite inferior -0.8000.
- DAC: canal 0, rango de medición +/-2V.
- ADC: canal 0, rango de medición +/-2V, tiempo de muestreo 1 ms.
- IIR2: frecuencia de corte 1Hz, tipo Bessel paso bajo.
- RGP : registro 0, valor -0.3.
- WGPR: registro 1.

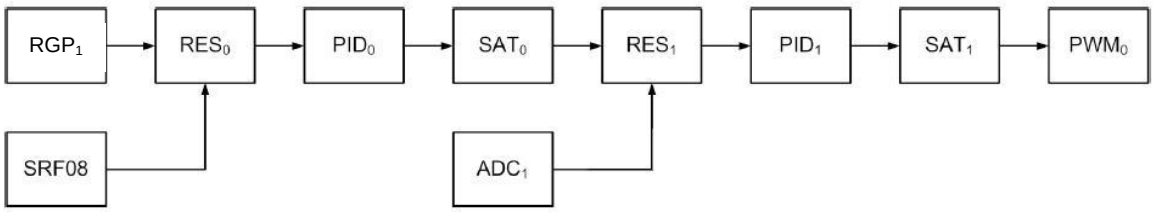

Figura 5.2 Diagrama de bloques para el control de un sistema *ball&beam.*

Especificación de las propiedades de los bloques para la estructura de la Figura 5.2.

- RGP: registro 1, valor 0
- ADC: canal 1, rango de medición +/-5V, tiempo de muestreo 1ms.
- PWM: dispositivo 0, modo bidireccional, offset 10%, frecuencia 5kHz.
  PID: bloque 0, tipo pid, tiempo de muestreo 10ms, valor P = 0.5, valor I = 0.0, valor D = 0.75, límite 1.
- PID: bloque 1, tipo pid, tiempo de muestreo 100ms, valor P = 0.1, valor I = 0.0, valor D = 0.75, límite 1.
  SAT: bloque 0, valor máximo 0.5, valor mínimo -0.5.
- SAT: bloque 1, valor máximo 0.75, valor mínimo -0.75

Tabla 5.1 Tiempo de desarrollo de las diferentes implementaciones.

	Implementación gráfica	**Implementación no gráfica**
Aplicación Figura 5.1	1minuto, 21 segundos	14 minuto, 35 segundos
Aplicación Figura 5.2	8 minutos, 30 segundos	39 minutos, 30 segundos

En cuanto a los resultados obtenidos, se tiene en la Tabla 5.1 el tiempo de desarrollo para la aplicación de la Figura 5.1 se redujo de 14 minutos, 35 segundos a 1 minuto, 21 segundos, lo cual representa una reducción del 90,74% del tiempo de desarrollo sin interfaz gráfica, mientras que en la aplicación de la Figura 5.2 el tiempo se logró reducir de 39 minutos, 30 segundos a 8 minutos, 30 segundos, lo cual representa una reducción del 78.5% del tiempo correspondiente a la implementación sin interfaz gráfica. Durante el proceso de implementación de las aplicaciones en ambos sistemas de interacción, los usuarios hicieron las siguientes observaciones:

- Mediante la implementación no gráfica, es posible cometer errores en el código de máquina que no son reportados, mientras que el sistema de interfaz gráfica permite la detección de errores durante el diseño de las aplicaciones.
- El diseño de las aplicaciones digitando código en formato hexadecimal requiere de mayor concentración que el utilizar el sistema de manipulación directa de bloques gráficos.
- El usuario requiere de un manual que especifica los códigos hexadecimales de cada instrucción o de la memorización de las instrucciones, mientras que el sistema de interfaz gráfica elimina la necesidad de manuales y no requiere memorización de funciones.

Estos resultados demuestran que la utilización del sistema de interacción gráfica facilita el desarrollo de las aplicaciones, reporta errores, elimina la necesidad de memorizar instrucciones o de depender de un manual, además reduce el tiempo de desarrollo de aplicaciones, lo cual permite el rápido modelado de sistemas de control.

Por otra parte, debido a que el sistema fue diseñado específicamente para la Unidad Controladora de procesos, tiene la limitante de ser compatible únicamente con este dispositivo, por lo que es recomendable hacer del entorno de desarrollo integrado un sistema flexible en cuanto a compatibilidad con otros dispositivos que puedan diferir ya sea en sus propiedades, cantidad de instrucciones e incluso en su gramática de lenguaje de programación. Sumado a esto, debido a que solamente hay disponible una Unidad Controladora de Procesos, el sistema solo permite la interacción de un usuario a la vez en cuanto a tareas de programación y ejecución del dispositivo.

El generador de código intermedio se diseñó como un sistema implícito dentro de la interfaz gráfica de usuario. Éste es capaz de generar el código intermedio que representa la estructura de control diseñada por el usuario a nivel gráfico. En cuanto a las ventajas que posee este sistema es que genera dicho código automáticamente, y sin errores léxico-sintácticos ni de semántica, ya que el sistema verifica si lo que el usuario realiza es permitido o no.

El traductor de código intermedio resultante es capaz de detectar 5 tipos de errores:

1. Léxicos.
2. Sintácticos.
3. Semánticos.
4. Uso de recursos físicos.
5. De comunicación y de sistema.

La lista de errores que el sistema puede soportar se encuentra en el apéndice D.

En cuanto a la quinta etapa de diseño, la herramienta de simulación ejecuta el diagrama de bloques traducido al lenguaje de GNU Octave en el servidor RMI, es decir el usuario no debe tener instalado GNU Octave para poder realizar la simulación, esto minimiza los requisitos de la computadora del usuario, los cuales se reducen a tener: a) el ambiente de ejecución de Java, b) un navegador de Internet, y c) acceso a la red en la cual se encuentra la Unidad Controladora de Procesos y los servidores HTTP y RMI. Cabe mencionar que esta característica de utilizar las aplicaciones de manera remota, está muy acorde con las nuevas tendencias de "computación en nube".

El sistema de simulación resultante, implementa 19 de los 30 módulos disponibles en la caja de herramientas para ser simulados, los demás, tales como los módulos de *hardware* se dejaron para una eventual fase de expansión.

Para obtener el porcentaje de error de los resultados de simulación respecto a otra simulación patrón se realizó el siguiente experimento:

Se implementó la siguiente estructura para simulación:

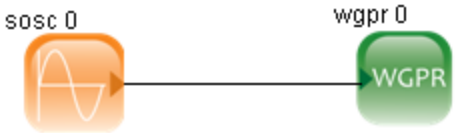

Figura 5.3 Estructura implementada en la herramienta de simulación que genera una señal sinusoidal cuya salida se escribe en un registro de propósito general.

La Figura 5.3 representa un generador de onda sinusoidal de amplitud 1 y frecuencia de 1Hz, y su salida se escribe en el registro de propósito general 0. Esta estructura fue implementada en la herramienta de simulación diseñada y en MATLAB. El código GNU Octave generado por la herramienta de simulación que representa la estructura de la Figura 5.3 es el siguiente:

```
n = [0:1000];
Ts = 0.01;
sosc0 = 1*sin(2*pi*1*n*Ts);
wgpr0 = sosc0;
wgpr0
```

donde, wgpr0 es el vector de salida.

Para realizar el experimento se obtuvo un vector de salida de 1000 muestras mediante la herramienta de simulación diseñada, y otro vector de salida con la misma cantidad de muestras empleando el mismo código para simulación mediante MATLAB. El margen de error se calculó mediante la siguiente ecuación de la desviación de raíz media cuadrática (RMSD), y la desviación de raíz media cuadrática normalizada (NRMSD):

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n} wgpr0_{MATLAB,i} - wgpr0_{Octave,i}^{\ 2}}{n}} \tag{5.1}$$

$$\%error = NRMSD = \frac{RMSD}{wgpr0_{MATLAB,Max} - wgpr0_{MATLAB,Min}} \cdot 100 \tag{5.2}$$

Al realizar los cálculos utilizando las ecuaciones 5.1 y 5.2 se obtuvo el siguiente porcentaje de error:

% error = 0.000013059

Analizando el resultado, se tiene que el error fue mucho menor al 5% planteado inicialmente como objetivo. Este margen de error le brinda confianza al usuario y evita que éste tenga que trascribir sus experimentos a otro simulador para verificar los resultados.

El sistema tiene la limitante de que solo un usuario a la vez puede realizar operaciones de simulación en el servidor RMI.

Por último, el Entorno de Desarrollo Integrado resultante disminuye la brecha entre el usuario y el lenguaje de maquina. Además permite el acceso de manera remota a la Unidad Controladora de Procesos, este funcionamiento fue probado en la red de área local de la Escuela de Ingeniería Electrónica. De igual manera funcionó correctamente desde Internet por medio de una red privada virtual.

En general, el Entorno de Desarrollo Integrado creado posee las siguientes ventajas desde el punto de vista del uso de una interfaz gráfica basada en diagramas de bloques:

- Mayor rapidez de uso y solución de problemas que el sistema de interacción sin interfaz gráfica.
- Facilidad de recordar los conceptos de operación para el usuario no experto.
- Explota las características visuales y espaciales.
- Elimina la necesidad de descomponer mentalmente una tarea en distintas instrucciones para entender su funcionamiento.
- Proporciona el contexto directamente, ya que las figuras representan exactamente los módulos y operaciones en el dispositivo.
- Realimentación inmediata, los resultados de las acciones del usuario son visualizados inmediatamente.
- Las acciones se pueden deshacer y volver hacer fácilmente.
- El mecanismo de uso intensivo del ratón utilizado, minimiza la necesidad de escribir texto.

La Figura 5.4 muestra los resultados y objetivos generales alcanzados por medio de la creación del nuevo sistema de interacción humano-máquina (IDE) creado para la Unidad Controladora de Procesos.

Figura 5.4 Objetivos alcanzados por medio de la creación de un Entorno de desarrollo Integrado para la Unidad Controladora de procesos.

Capítulo 6: Conclusiones y recomendaciones

6.1 Conclusiones

- La selección de los mnemónicos para cada código de instrucción facilitó la comprensión de las funciones de la Unidad Controladora de Procesos y el desarrollo de la etapa de creación del traductor de código intermedio.

- El lenguaje intermedio resultante es de tipo ensamblador, y posee una correspondencia uno a uno con las instrucciones y parámetros en lenguaje de máquina. No soporta definiciones de tipos de datos o variables.

- La creación de un lenguaje intermedio facilitó la depuración del Entorno de Desarrollo Integrado durante el proceso de diseño y desarrollo.

- El entorno de desarrollo integrado diseñado provee facilidad para recordar las funcionalidades soportadas por el sistema, y elimina la necesidad de escribir código de programación, lo cual simplifica el desarrollo de las aplicaciones.

- El uso de una interfaz gráfica, redujo el tiempo de programación de aplicaciones lo que permite el modelado rápido de sistema de control.

- El entorno de desarrollo integrado fue diseñado específicamente para la Unidad Controladora de Procesos, por lo que no es un sistema independiente del dispositivo electrónico.

- El traductor de código intermedio resultante es un sistema capaz de detectar errores de tipo léxico, sintáctico y semántico, lo cual hace del IDE un sistema poco propenso a errores.

- El utilizar GNU Octave como motor de simulación con un porcentaje de error mucho menor al 1%, induce al usuario a confiar en los resultados obtenidos.

- El uso de un servidor RMI no reentrante limita su uso a un usuario a la vez.

- El hecho de que GNU Octave sea ejecutado por el sistema de interfaz gráfica de manera remota, está muy acorde con las nuevas tendencias de "computación en nube".

- El Entorno de Desarrollo Integrado creado disminuye la brecha entre el usuario y el lenguaje de máquina.

6.2 Recomendaciones

Como una de las recomendaciones, se sugiere independizar el Entorno de Desarrollo Integrado del dispositivo electrónico con el fin de que sea reutilizable en distintos prototipos de Unidades Controladoras de Procesos, sin necesidad de cambiar y recompilar el código fuente.

Se recomienda además, la creación de un sistema que administre el acceso al dispositivo o distintos dispositivos, según avance el desarrollo del proyecto TeleLAB. Este sistema, podría establecer sesiones de usuario y visualización en tiempo real de la planta a controlar, así el usuario, tendría mayor conocimiento del comportamiento del sistema físico en estudio.

Un aspecto a considerar, es que en este proyecto, no todos los bloques gráficos que aparecen en la caja de herramientas fueron implementados en la aplicación de simulación, por lo que se recomienda implementar las funciones faltantes en dicha herramienta de simulación para aumentar la cantidad de sistemas de control que se pueden simular.

Debido a que la herramienta de simulación solo puede ser utilizada por un usuario a la vez, se recomienda hacer de esta herramienta un sistema multiusuario para maximizar su utilidad.

Bibliografía

[1] Castro Molina, Daniel. *Manual de usuario de la unidad controladora*. 2008.

[2] Castro Molina, Daniel. *Unidad Controladora de Procesos para el diseño, análisis, simulación e implementación de sistemas de control automático a través de redes TCP/IP*, 2008.

[3] Deitel, H. M., Deitel, P. J. *Java™ How to program*. Sexta Edición. Prentice Hall, 2005.

[4] Deitel, H. M., Deitel, P. J., Santry S. E. *Advanced Java 2 platform How to program*. Primera Edición. Prentice Hall, 2001.

[5] FatCow. 1000 Free "Farm-Fresh Web Icons" [en línea]. Enero, 2010 [17 de enero, 2010]. URL http://www.fatcow.com/free-icons/

[6] FreeMat. FreeMat [en línea]. Diciembre, 2009, [24 de diciembre, 2009]. URL http://freemat.sourceforge.net/index.html#

[7] Galitz, Wilbert O. *The Essential Guide to User Interface Design. An Introduction to GUI Design Principles and Techniques*. Tercera Edición. Wiley Publishing, 2007.

[8] GNU, *Bison – GNU parser generator* [en línea]. Febrero, 2010, [14 de febrero, 2010]. URL http://www.gnu.org/software/bison/

[9] INRIA-Unité de recherche de Rocquencourt. Introduction to Scilab [en línea]. Diciembre, 2009, [24 de diciembre, 2009]. URL http://www.scilab.org/doc/intro/intro.pdf

[10] ISA-UMH. RECOLAB Remote: Control Laboratory [en línea]. Febrero, 2010, [4 de febrero, 20120]. URL http://isa.umh.es/temas/recolab/index_en.html

[11] Jiménez, L.M., Reinoso, O., Puerto, R., Azorín, J.M. Laboratorios remotos para las prácticas ingeniería de sistemas y automática en la universidad de Miguel Hernández [en línea]. Febrero, 2009, [4 de febrero, 2010]. URL http://isa.umh.es/arvc/documentos/articulos/UMHLaboratoriosRemotos03.pdf

[12] Karl-Friedrich Kraiss. *Advanced Man-Machine Interaction*. Primera Edición. Springer, 2006.

[13] Microsoft. Centro de desarrollo de .NET Framework [en línea]. Enero, 2010, [10 de enero, 2010]. URL http://msdn.microsoft.com/es-cr/netframework/default.aspx

[14] MKS. *MKS Lex & Yacc* [en línea]. Febrero, 2010, [14 de febrero, 2010]. URL http://www.mkssoftware.com/products/ly/

[15] Mora Mata, Miguel Ángel; Gálvez Rojas, Sergio. *Java a Tope: Compiladores, Traductores y Compiladores con Lex/Yacc,JFlex/Cup y JavaCC*. Primera Edición. Universidad de Málaga, 2005.

[16] National Instruments. ¿Qué es LabVIEW? [en línea]. Diciembre, 2009, [25 de diciembre, 2009]. URL http://www.ni.com/labview/whatis/esa/

[17] Python Software Foundation, Python Programming Language -- Official Website [en línea]. Enero, 2010, [10 de enero, 2010]. URL http://www.python.org/

[18] Slonnerger, Kenneth; Kurtz, Barry L. *Formal Sintax and Semantics of Programming Languages*. Primera Edición. Addison Wesley, 1995.

[19] Tango Desktop Project, Tango Icon Library [en línea]. Enero, 2010, [17 de enero, 2010]. URL http://tango.freedesktop.org/Tango_Icon_Library

[20] Tecnológico de Moterrey. Telelab: Automations Lab [en línea]. Febrero, 2010, [4 de febrero, 2010]. URL http://telelab.mty.itesm.mx/

[21] The Flex Project, *The Fast Lexical Analyzer* [en línea]. Febrero, 2010, [14 de febrero, 2010]. URL http://flex.sourceforge.net/

[22] The MathWorks, Inc. The MathWorks-Company [en línea]. Diciembre, 2009, [24 de diciembre, 2009]. URL http://www.mathworks.com/company/

[23] Università degli Studi di Siena. Automatic Control Telelab [en línea]. Febrero, 2010, [4 de febrero, 2010]. URL http://act.dii.unisi.it/home.php

[24] University of Wisconsin, Department of Chemical Engineering. *Octave* [en línea]. Diciembre, 2009, [25 de diciembre, 2009]. URL http://www.gnu.org/software/octave/index.html

[25] Wikimedia Foundation, Inc. Backus-Naur Form [en línea]. Diciembre, 2009, [26 de diciembre, 2009]. URL http://es.wikipedia.org/wiki/Backus-Naur_form

[26] Wikimedia Foundation, Inc. *Interacción persona-computador* [en línea]. Diciembre, 2009, [30 de diciembre, 2009]. URL http://es.wikipedia.org/wiki/Interacci%C3%B3n_persona-computador

[27] Wikimedia Foundation, Inc. *Lenguaje de Programación* [en línea]. Diciembre, 2009, [26 de diciembre, 2009]. URL http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n

Apéndices

A. Glosario

ADC: Convertidor analógico-digital.

Área de trabajo: Área de la interfaz gráfica diseñada sobre la cual se diseñan las estructuras gráficas.

Back End: Etapa final de un sistema traductor de código.

Barra de herramientas: Objeto gráfico sobre el cual se encuentran distintas funciones aplicables por medio de botones en una interfaz gráfica.

BNF: Backus-Naur Form, manera formal de describir gramática de lenguajes de programación.

Caja de herramientas: Objeto gráfico que contiene los elementos manipulables en la interfaz gráfica usuario basada en manipulación directa.

Compilador: Programa que traduce código escrito en un lenguaje fuente en otro código en lenguaje destino.

Errores de semántica: Errores referentes al significado o interpretación de los elementos que componen un lenguaje.

Errores de sintaxis: Errores referentes a la estructura de un lenguaje.

Errores léxicos: Errores referentes a la escritura de los elementos individuales de un lenguaje.

Ethernet: Familia de productos de red de área local cubiertos por el estándar IEEE 802.3 que comúnmente se conoce como CSMA / CD.

FreeMat: Herramienta de cálculo numérico bajo licencia de código abierto desarrollada por Samit Basu.

Front End: Etapa inicial de la estructura de un traductor de código.

GNU Octave: Herramienta de cálculo numérico bajo licencia de código abierto desarrollada por John W. Eaton de la Universidad de Wisconsin.

GUI: Siglas en inglés para *Graphical User Interface*.

Herramienta de Captura: Herramienta para la captura de datos implementada como parte del entorno de desarrollo integrado diseñado para la Unidad Controladora de procesos.

Herramienta de Simulación: Herramienta para la simulación de estructura de control implementada como parte del entorno de desarrollo integrado diseñado para la Unidad

Controladora de procesos.

HTTP: *HyperText Transfer Protocol*, o Protocolo de Transferencia de Hipertexto.

IHC: Interacción humano-computadora.

Intérprete: Tipo de traductor de código cuya salida es una ejecución y no un código objeto.

Java 2D: Conjunto de aplicaciones de Java para el diseño de gráficos en dos dimensiones.

Java Applet: Aplicación java embebida en un explorador de Internet.

JavaCC: herramienta de metacompilación para la construcción de compiladores en Java.

LabVIEW: Entorno de programación gráfico para el desarrollo de aplicaciones de medición, pruebas y sistemas de control.

LAN: rede área local

Lenguaje de máquina: Lenguaje directamente interpretado por un sistema microcontrolado.

Lenguaje Intermedio: Código intermedio entre el lenguaje fuente y el lenguaje de máquina.

MATLAB: Entorno de desarrollo de aplicaciones para el cálculo numérico desarrollado por *The Mathworks* y es *software* propietario.

Netlist: Sección de código que especifica la relación entre distintos objetos.

Osciloscopio flotante: Herramienta para la visualización en tiempo real implementada en el Entorno de Desarrollo Integrado para la Unidad Controladora de Procesos.

PWM: modulación de ancho de pulso.

RMI: *Remote Method Invocation*, o método de invocación remota.

RPC: *Remote Procedure Call*, o llamada a procedimiento remoto

RS232: Interfaz para el intercambio serie de datos binarios.

Scilab: Herramienta para cálculos numéricos bajo licencia de código abierto desarrollado por Digiteo.

TCP/IP: Familia de protocolos de Internet.

TF: *Transfer Function.*

Unidad Controladora de Procesos: sistema embebido que posee funcionalidades para el diseño de estructura de control.

VPN: *Virtual Private Network*, o red privada virtual.

B. Descripción de los mnemónicos seleccionados para el lenguaje intermedio

La Tabla B.1 presenta los mnemónicos del lenguaje intermedio definidos para cada código hexadecimal del lenguaje de máquina.

Tabla B.1 Cobertura de las instrucciones en mnemónicos.

START	**0x11**
STOP	**0x11**
PAUSE	**0x11**
READREG	**0x13**
ERASE	**0x14**
SAVE	**0x15**
NETLIST	**0x16**
IOCONFIG	**0x17**
RESET	**0x18**
SETVALUE	**0x19**
RANGEADC	**0x1A**
RANGEDAC	**0x1B**
READADC	**0x1C**
PWM	**0x1D**
FILTER	**0x21**
TF	**0x23**
PID	**0x25**
SOSC	**0x26**
ROSC	**0x27**
TOSC	**0x28**
VOSC	**0x29**
K	**0x2B**
MOD	**0x31**
GAIN	**0x32**
SAT	**0x33**
NOISE	**0x34**
CAPTURE	**0x51**
CAPTURECE	**0x52**
CAPTURER	**0x53**
READMOD	**0x54**
RGP	**0x55**
USECPU	**0x56**

Funciones de *hardware* y sistema

Estas instrucciones se refieren a todas aquellas que tienen que ver con la configuración o manipulación directa del *hardware* del sistema

STOP;

Código de operación hexadecimal interno: 0x11 0x00
Detiene la máquina virtual de la Unidad Controladora.
Parámetros: no tiene parámetros, aunque internamente tiene 0x00 como parámetro para indicar que se desea detener la máquina.

START;

Código de operación hexadecimal interno: 0x11 0x01
Inicia la máquina virtual de la Unidad Controladora.
Parámetros: no tiene parámetros, aunque internamente tiene 0x01 como parámetro para indicar que se desea iniciar la máquina.

PAUSE;

Código de operación hexadecimal interno: 0x11 0x02
Inicia la máquina virtual de la Unidad Controladora.
Parámetros: no tiene parámetros, aunque internamente tiene 0x02 como parámetro para indicar que se desea iniciar la máquina, desde el punto de vista del lenguaje intermedio ésta instrucción no tiene parámetros.

READREG REGX [0-15];

Código de operación hexadecimal interno: 0x13 [0x00-0xFF]
Lee el valor de un registro de estado.
Parámetros: tiene un parámetro y es un valor entre 0 y 15.

ERASE;

Código de operación hexadecimal interno: 0x14 X
X: significa que internamente el sistema pone cualquier valor para cumplir con el formato interno de que todas las instrucciones deben tener un parámetro.
Borra la memoria EEPROM.
Parámetros: no tiene

SAVE;

Código de operación hexadecimal interno: 0x15 X
X: significa que internamente el sistema pone cualquier valor para cumplir con el formato interno de que todas las instrucciones deben tener un parámetro.
Guarda la configuración de sistema en EEPROM
Parámetros: no tiene.

NETLIST
begin
Parámetros
end;

Código de operación hexadecimal interno: 0x16 N [cadena de bytes variable]
Recibe el archivo de aplicación para la máquina virtual.
Parámetros: archivo *netlist*.

IOCONFIG N, F;

Código de operación hexadecimal interno: 0x17 N F
Recibe un archivo de configuración para un periférico IO
Parámetros:
N: número de módulo IO.
F: archivo de configuración

RESET;

Código de operación hexadecimal interno: 0x18 X

X: significa que internamente el sistema pone cualquier valor para cumplir con el formato interno de que todas las instrucciones deben tener un parámetro.

Reinicia el sistema

Parámetros: no tiene.

SETVALUE N, A;

Código de operación hexadecimal interno: 0x19 N [0x00-0x05] A [6 bytes en formato ASCII]

Asigna un valor a un elemento DAC o PWM

Parámetros:

N: dispositivo DAC o PWM.

Opciones en mnemónicos de N

- dac0
- dac0
- pwm0
- pwm1
- pwm2
- pwm03

A: Valor numérico entre 0 y 1 en formato de 6 dígitos incluyendo el signo [0.0000, 1.0000] ó +/-[0.000,1.000].

RANGEADC N, R;

Código de operación hexadecimal interno: 0x1A N [0x00-0x03] R [0x00-0x0F]

Define un rango de medición a un dispositivo ADC

Parámetros:

N: dispositivo ADC

Opciones en mnemónicos de N

- adc0
- adc1
- adc2
- adc3

R: valor entre 0 y 15

RANGEDAC N, R;

Código de operación hexadecimal interno: 0x1B N [0x00-0x03] R [0x00-0x07]

Define un rango de medición a un dispositivo DAC

Parámetros:

N: dispositivo DAC

Opciones en mnemónicos de N

- dac0
- dac1

R: valor entre 0 y 7

READADC N;

Código de operación hexadecimal interno: 0x1C N [0x00-0x03]

Lee un dispositivo ADC, retorna el valor normalizado.

Parámetros:

N: dispositivo ADC

Opciones en mnemónicos de N

- adc0
- adc1
- adc2
- adc3

Funciones de módulos (I)

Estas instrucciones se refieren a aquellas cuyas funciones tienen que ver con la creación configuración y manipulación de los módulos virtuales.

FILTER T, N, Fc, BW, R, TF;

Código de operación hexadecimal interno: 0x21 T [0x00-0x03] N [0x00-0xFF] Fc [6 bytes en ASCII] BW [6 bytes en ASCII] R[6 bytes en ASCII] TF [0x00-0x05]
Cambia la configuración de los filtros IIR, FIR o NOTCH.
Parámetros:

T: tipo de filtro
Opciones en mnemónicos de T

- iir_2
- iir_4
- fir
- notch

N: número de bloque no mayor a 255.
Fc: Frecuencia de corte en Hz en formato de 6 dígitos positivo (solos para IIR2, IIR4, FIR).
BW: Ancho de banda en Hz en formato de 6 dígitos positivo (solo para IIR2, IIR4).
R: radio de polos conjugados en formato de 6 dígitos positivo (solo para NOTCH).
TF: tipo de filtro IIR2, IIR4

- butterwlp
- butterwhp
- bessellp
- besselhp
- butterwbp
- butterwrp

TF: tipo de filtro FIR

- De 1 a 16 etapas

- 32 etapas.
- 64 etapas.

TF N, T (TL, TH), TL = pole, G, P1r, Pii, P2r, P2i, Z1r, Z1i, Z2r, Z2i;

TF N, T (TL, TH), TL = k_f, B0, B1, A1, B2, A2;

Código de operación hexadecimal interno: 0x23 N [0x00-0xFF] TL [0x00-0xFF] TH [0x00-0xFF] TL[0x01] G[6 bytes en ASCII] P1[12 bytes en ASCII] P2[12 bytes en ASCII] Z1[12 bytes en ASCII] Z2[12 bytes en ASCII]

Configura un módulo de función de transferencia.

Parámetros:

N: Número de bloque, valor numérico no mayor a 255.

T: Tiempo de muestreo en ms valor numérico positivo no mayor a 65535

TL: mnemónicos

pole (formato ganancia, P1 = > r,i, P2 => r,i y ceros Z1 => r,i y Z2 => r,i), donde r es la parte real en 6 dígitos e i es la parte imaginaria también en 6 dígitos.

k_f (formato B0, B1, A1, B2, A2, donde cada valor es una constante de 6 dígitos).

Caso TL = pole

G: valor de 6 dígitos

P1r: polo 1 parte real, valor de 6 dígitos

P1i: polo 1 parte imaginaria, valor de 6 dígitos

P2r: polo 2 parte real, valor de 6 dígitos

P2i: polo 2 parte imaginaria, valor de 6 dígitos

Z1r: cero 1 parte real, valor de 6 dígitos

Z1i: cero 1 parte imaginaria, valor de 6 dígitos

Z2r: cero 2 parte real, valor de 6 dígitos

Z2i: cero 2 parte imaginaria, valor de 6 dígitos

Caso TL = k_f

B0: constante de 6 dígitos

B1: constante de 6 dígitos

A1: constante de 6 dígitos

B2: constante de 6 dígitos

A2: constante de 6 dígitos

PID TF, N, T (TL, TH), P, I, D;

Código de operación hexadecimal interno: 0x25 N [0x00-0x03]

Configura un dispositivo PID por medio de parámetros P, I, D.

Parámetros:

TF: tipo de PID

- pid0 (PID)
- pid1 (PI_D)
- pid2 (I_PD)

N: Número de bloque, valor numérico no mayor a 255.

T: Tiempo de muestreo en ms no mayor a 65535

P: Constante de 6 dígitos

I: Constante de 6 dígitos

D: Constante de 6 dígitos

SOSC N, X, G;

Código de operación hexadecimal interno: 0x26 N [0x00-0xFF] X [6 bytes ASCII] G [6 bytes ASCII]

Configura un oscilador sinusoidal

Parámetros:

N: Número de bloque, valor numérico no mayor a 255.

X: frecuencia en Hz de 6 dígitos.

G: Amplitud de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000].

ROSC N, T1 (TL1, TH1), T2 (TL2, TH2), T3 (TL3, TH3), GP, GN;

Código de operación hexadecimal interno: 0x27 N [0x00-0xFF] TL1 [0x00-0xFF] TH1 [0x00-0xFF] TL2 [0x00-0xFF] TH2 [0x00-0xFF] TL3 [0x00-0xFF] TH3 [0x00-0xFF] GP [6 bytes ASCII] GN [6 bytes ASCII].

Configura un oscilador rectangular

Parámetros:

N: Número de bloque, valor numérico no mayor a 255.

T1: Tiempo 1 en ms no mayor a 65535

T2: Tiempo 2 en ms no mayor a 65535

T3: Tiempo 3 en ms no mayor a 65535

GP: Amplitud positiva de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000].

GN: Amplitud negativa de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000].

TOSC N, T1 (TL1, TH1), T2 (TL2, TH2), GP, GN;

Código de operación hexadecimal interno: 0x28 N [0x00-0xFF] TL1 [0x00-0xFF] TH1 [0x00-0xFF] TL2 [0x00-0xFF] TH2 [0x00-0xFF] GP [6 bytes ASCII] GN [6 bytes ASCII].

Configura un oscilador triangular

Parámetros:

N: Número de bloque, valor numérico no mayor a 255.

T1: Tiempo 1 en ms no mayor a 65535

T2: Tiempo 2 en ms no mayor a 65535

GP: Amplitud positiva de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000].

GN: Amplitud negativa de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000].

VOSC N, T1 (TL1, TH1), T2 (TL2, TH2), GP, GN;

Código de operación hexadecimal interno: 0x29 N [0x00-0xFF] TL1 [0x00-0xFF] TH1 [0x00-0xFF] TL2 [0x00-0xFF] TH2 [0x00-0xFF] GP [6 bytes ASCII] GN [6 bytes ASCII].

Configura un oscilador rectangular de ancho de pulso variable

Parámetros:

N: Número de bloque, valor numérico no mayor a 255.

T1: Tiempo 1 en ms no mayor a 65535

T2: Tiempo 2 en ms no mayor a 65535

GP: Amplitud positiva de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000].

GN: Amplitud negativa de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000].

K N, TL, TH, X, K;

Código de operación: 0x2B

Configuración de constante K

Instrucción no implementada

Funciones de módulos (II)

Segunda parte de instrucciones relacionadas a la creación y configuración de módulos

MOD
begin
    Módulo, cantidad;
End

Código de operación: 0x31

Crea y destruye módulos, más adelante se explicará en detalle la función de esta instrucción.

GAIN N, X;

Código de operación hexadecimal interno: 0x32 N [0x00-0xFF] X [6 bytes ASCII]

Configuración de ganancia de módulo GAIN

Parámetros:

N: Número de bloque, valor numérico no mayor a 255.

X: ganancia, valor de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000] ó +/- [0.000,1.000].

SAT N, X, Y;

Código de operación hexadecimal interno: 0x33 N [0x00-0xFF] X [6 bytes ASCII]
Y [6 bytes ASCII]
Configuración de límites del módulo Saturación
Parámetros:
N: Número de bloque, valor numérico no mayor a 255.
X: Límite superior, valor de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000] ó +/- [0.000,1.000].
Y: Límite inferior, valor de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000] ó +/- [0.000,1.000].

NOISE N, X, Y;

Código de operación hexadecimal interno: 0x34 N [0x00-0xFF] X [6 bytes ASCII]
Y [6 bytes ASCII]
Configuración de límites del módulo Saturación
Parámetros:
N: Número de bloque, valor numérico no mayor a 255.
X: Límite superior, valor de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000] ó +/- [0.000,1.000].
Y: Límite inferior, valor de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000] ó +/- [0.000,1.000].

Funciones de captura de datos

Las funciones de captura de datos se refieren a todas aquellas instrucciones dedicadas a la captura de datos de salida de los módulos virtuales y de *hardware*.

CAPTURE S (SL, SH), SS, T, N, O;

Código de operación hexadecimal interno: 0x51SL [0x00-0xFF] SH [0x00-0xFF] SS [0x00-0xFF] T [0xFF] N [0xFF] O [0xFF]

Configuración para la tarea de captura

Parámetros:

S: tiempo de muestreo en ms, no mayor a 65535

SS: Tiempo total en segundos no mayor a 255

T: Módulo (mnemónico)

N: Número de bloque

O: Número de salida

CAPTURECE OP;

Código de operación hexadecimal interno:

0x52 OP [0x00|0x01|0x02|0x03|0x05|0x05]

Ejecución de la tarea de captura

Parámetros:

OP: Estado de la tarea de captura

- stop (desactivar o detener)
- start (iniciar la captura)
- ready (datos listos)
- send (enviar los datos)
- erase (borrar los datos)

El mnemónico CAPTURECE proviene de la palabra CAPTURE mas CE de *Command Execution*, ya que esta instrucción es la que ejecuta la tarea de captura en sus distintas opciones

CAPTURER R;

Código de operación hexadecimal interno: 0x53 R [0x00|0x01|0x02]

Retorna el valor del registro de la tarea de captura

Parámetros:

OP: Estado de la tarea de captura

- 0 (Máquina que tiene la tarea de captura)
- 1 (Estado de la tarea de captura en máquina virtual 0)
- 2 (Estado de la tarea de captura en máquina virtual 1)

READMOD M, N, O;

Código de operación hexadecimal interno: 0x54 M [0x00-0xFF], N [0x00-0xFF],

O [0x00-0xFF]

Regresa el valor de salida de los módulos virtuales.

Parámetros:

M: módulo virtual

N: número de bloque del módulo virtual

O: número de bloque del módulo virtual

Si se desea obtener la salida de más de un módulo, solo se debe agregar otros tres parámetros separados por coma y la máquina devolverá los datos en el mismo orden en el que se solicitaron.

RGP N, V;

Código de operación hexadecimal interno: 0x55 N [0x00-0x0F], V [6 byte ASCII],

Guarda un valor en un registro de propósito general.

Parámetros:

N: número de registro [0-15].

V: Valor numérico de 6 dígitos entre 0 y 1 incluido el signo [0.0000, 1.0000] ó +/- [0.000,1.000].

USECPU;

Código de operación hexadecimal interno: 0x56 X

X: significa que internamente el sistema pone cualquier valor para cumplir con el formato interno de que todas las instrucciones deben tener un parámetro.

Retorna el porcentaje de utilización del CPU, el cual es un número entre 0 y 1, donde 1 representa 100%.

Parámetros: no tiene.

Mnemónicos para los módulos virtuales

Para los módulos virtuales, se tomó el mismo nombre o mnemónico con el que se hace referencia a cada uno de ellos en la documentación de la Unidad Controladora de Procesos.

Módulos de *hardware*

Tabla B.2 Módulos de entrada.

Mnemónico	Código hexadecimal
adc	0x01
rgp	0x02
i2cr	0x03
qeis	0x04
qeip	0x05
srf08	0x08

Tabla B.3 Módulos de salida.

Mnemónico	Código hexadecimal
dac	0xC0
pwm	0xC1
wgpr	0xC2
i2cw	0xC3

Módulos virtuales

Tabla B.4 Procesamiento de señales.

Mnemónico	Código hexadecimal
iir2	0x45
iir4	0x46
fir	0x47
notch60	0x41

Tabla B.5 Control Lineal.

Mnemónico	Código hexadecimal
k	0x40
pid	0x42
pi_d	0x43
I_pd	0x44
tf	0x48
delay	0x53

Tabla B.6 Control no lineal.

Mnemónico	Código hexadecimal
sat	0x4A

Tabla B.7 Operaciones matemáticas

Mnemónico	Código hexadecimal
sum	0x49
res	0x4F
gain	0x50
mul	0x51

Tabla B.8 Generadores de señal.

Mnemónico	Código hexadecimal
sosc	0x4B
rosc	0x4C
tosc	0x4D
vosc	0x4E
noise	0x52

C. Manual de usuario

C.1 Estructura de archivos del Entorno de Desarrollo Integrado

El sistema de interfaz gráfica es un programa embebido en un explorador de Internet, el cual se compone de un archivo de extensión html y una carpeta llamada "classes", la cual contiene todos los archivos de extensión *.class y *.jar de java necesarios para ejecutar la aplicación.

C.2 Inicio del Entorno de Desarrollo Integrado

Existen dos maneras para iniciar el entorno de desarrollo integrado:

1. Haciendo clic en el link del Java Applet disponible en un servidor http.
2. Haciendo doble clic al archivo *.html comúnmente llamado index.html

C.3 Estructura y funciones de la interfaz gráfica de usuario

La interfaz gráfica en general se compone de tres áreas u objetos principales:

- Barra de Menús y barra de herramientas
- Caja de herramientas
- Área de trabajo

Ver siguiente la Figura C.1.

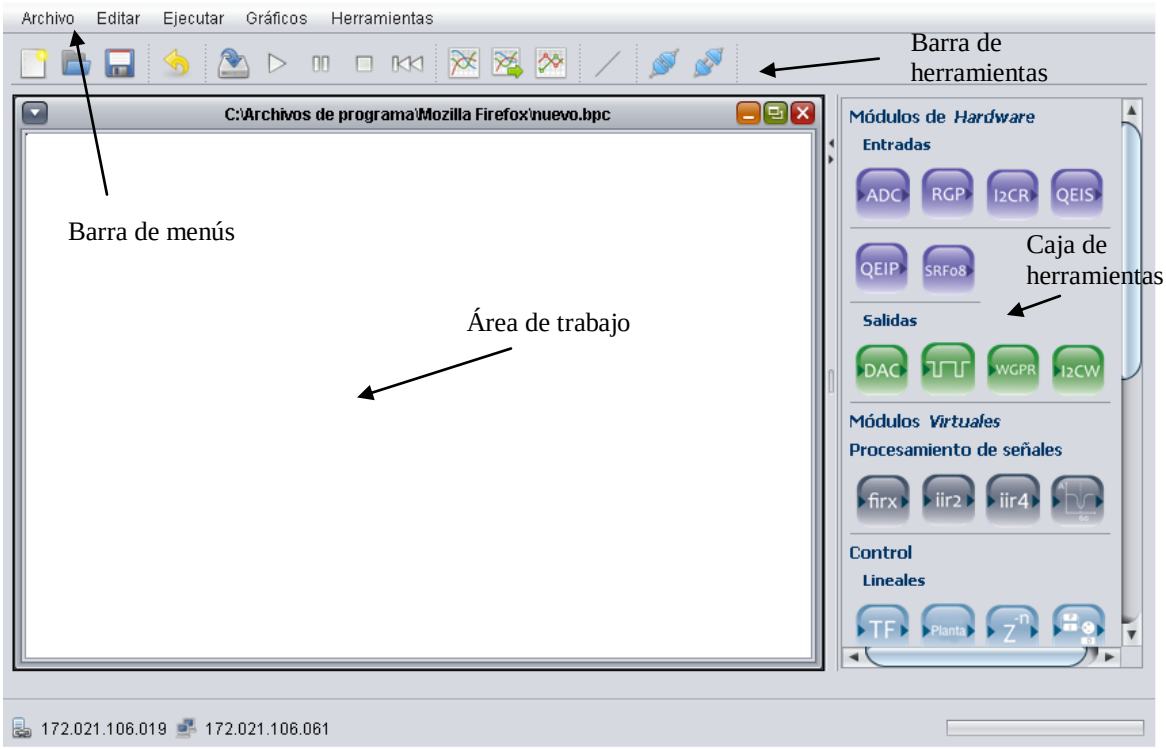

Figura C.1 Entorno de Desarrollo Integrado y sus componentes.

Barra de Menús y barra de herramientas

La barra de menús, contiene un conjunto de menús desplegables, en los cuales se encuentran todas las funciones que se pueden realizar con el entorno de desarrollo integrado, y contiene además información sobre las combinaciones de teclas que se pueden utilizar para ejecutar dichas funciones utilizando el teclado.

Menú archivo: Contiene las funciones típicas de manejo de archivo: Nuevo, Abrir, Guardar Como…, y Guardar.

- Nuevo: se refiere a crear un documento nuevo en blanco, con el fin de realizar alguna tarea.
- Abrir: se refiere a abrir un documento ya existente.
- Guardar como: se refiere a la tarea de guardar un archivo con la posibilidad de asignarle un nombre, y/ó un lugar de almacenamiento específico.

- Guardar: se refiere a guardar un archivo en el mismo lugar donde se ubica y con el mismo nombre.

Menú Editar. Contiene las funciones:

- Deshacer: deshace lo hecho en el área de trabajo solo una vez.
- Cortar: elimina el componente seleccionado.
- Copiar: copia el componente seleccionado.
- Pegar: pega el componente copiado en un punto especificado.

Menú Ejecutar. Contiene las siguientes funciones:

- Programar dispositivo: se utiliza para programar la Unidad Controladora de Procesos con el código que representa la estructura diseñada en el área de trabajo.
- Iniciar, Pausa, Detener y Reiniciar: funciones que se utilizan para iniciar, pausar, detener o reiniciar el dispositivo electrónico.

Menú Gráficos. Contiene las siguientes funciones:

- Simular: Abre la herramienta de simulación, la cual se utiliza para simular las aplicaciones diseñadas en el área de trabajo.
- Capturar Datos: Despliega la herramienta para capturar de datos.
- Osciloscopio Flotante: Despliega el osciloscopio flotante para la visualización de datos en tiempo real.
- Línea de conexión: habilita una línea para realizar conexiones entre dispositivos.
- Conectar al dispositivo: Establece conexión vía TCP/IP con el dispositivo electrónico
- Desconectar dispositivo: Interrumpe la conexión con el dispositivo electrónico.

Menú Herramientas. En este menú se encuentran las siguientes funciones:

- Configuración IP: herramienta para la configuración de las direcciones IP del dispositivo electrónico, y el servidor RMI que ejecuta GNU Octave para realizar los cálculos de simulación.
- Compilar a texto: Herramienta que genera el archivo bpc.txt con el código intermedio que representa las estructura gráficas en el lenguaje intermedio, por defecto se guarda en el directorio en el cual se ejecuta la interfaz gráfica.
- Compilador: herramienta de edición y compilación del lenguaje intermedio a lenguaje de máquina.
- Tema: muestra otro menú en el cual se puede escoger el tema o "*skin"* de la interfaz.

En cuanto a la barra de herramientas, ésta implementa algunas de las funciones de los menús antes descritas, por medio de botones, las funciones que implementa son las siguientes, en el orden de los botones de izquierda a derecha:

Nuevo, Abrir, Guardar, Deshacer, Programar dispositivo, Iniciar, Pausa, Detener, Reiniciar, Simular, Capturar Datos, Osciloscopio Flotante, Línea de conexión, Conectar al dispositivo, Interrumpir la conexión con el dispositivo.

Caja de herramientas

Esta área de la interfaz es la que contiene todo el conjunto de bloques funcionales y soportados por la Unidad Controladora de Procesos, para utilizarlo cada bloque, solo debe hacer clic en un bloque para seleccionarlo y luego hacer clic en el área de trabajo para crear la figura.

Área de trabajo

El área de trabajo es el objeto sobre el cual se colocan los bloques de la caja de herramientas, en esta área los bloques pueden ser manipulados, se pueden arrastrar con el

mouse, se pueden configurar, conectar entre sí, seleccionar, cortar, copiar y pegar de manera individual, y de manera múltiple solo se tiene la opción de seleccionar y cortar.
Por otra parte, para los bloques que son configurables, al hacer doble clic izquierdo sobre un bloque gráfico aparece una ventana de propiedades, si el bloque no es configurable dicha ventana no aparece, se debe presionar el botón de "Aplicar" para aplicar las propiedades, si no se hace y se cierra la ventana, las propiedades no se aplicaran, ver Figura C.2. Además al hacer clic derecho sobre un bloque gráfico aparece un menú de opciones que incluye las opciones de cortar, copiar y pegar. Las líneas de conexión también pueden eliminarse de manera individual y colectiva, éstas no poseen propiedades, y solo tienen la opción de cortar en el menú de opciones de clic derecho.

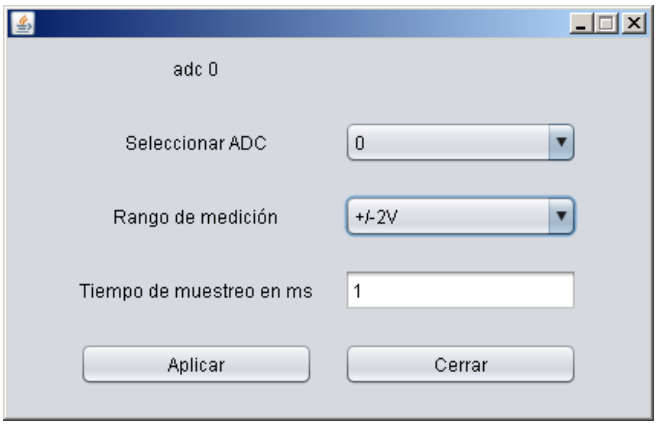

Figura C.2 Ventana de propiedades de un componente gráfico.

Conexión de bloques

Para conectar un bloque con otro, debe acercar el puntero del *mouse* a una entrada o salida de un bloque hasta que aparezca una marca circular de color negro, la cual indica que la conexión se puede realizar, se debe dar clic derecho para conectar el primer extremo de la línea, luego se busca la entrada o salida de otro dispositivo para conectar el otro extremo de dicha línea, una vez seleccionado el último punto de conexión, se debe dar doble clic para soltar la línea del puntero, mientras no se aplique el doble clic en una entrada o salida válida, la línea seguirá sujeta al puntero. Mientras la línea esté sujeta al puntero y no se

haya aplicado el doble clic, se puede dibujar una trayectoria de la conexión más cómoda y comprensible con solo hacer un clic en distintos puntos del área de trabajo.

Ver Figura C.3.

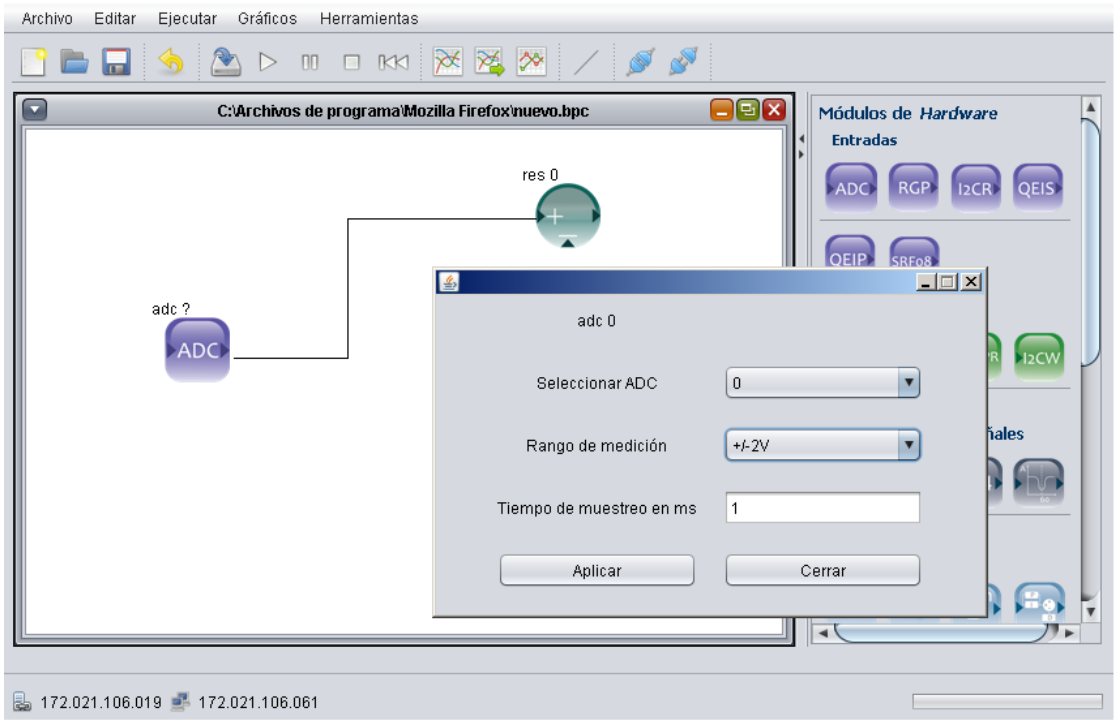

Figura C.3 Conexión de bloques y la ventana de propiedades.

Herramienta de Simulación

Para lograr utilizar correctamente la herramienta de simulación, debe asegurase de que el programa Servidor_rmi.jar se encuentra correctamente instalado y que se encuentra en la misma red del cliente, o en otra red a la cual el cliente tiene acceso.

En el caso de requerir instalar el programa Servidor_rmi.jar, siga los siguientes pasos solamente para Windows:

1. Instale **GNU Octave.**
2. Copie el programa Servidor_rmi.jar en una carpeta con el mismo nombre.
3. Descomprima el archivo Servidor_rmi.jar, puede utilizar el programa WinRAR para ello.
4. Abra el **panel de control**.
5. Doble clic en **Sistema**.
6. En la pestaña **Opciones avanzadas** clic en la opción **Variables de entorno.**
7. En Variables del Sistema, si no existe la variable **classpath,** debe crear una nueva con este nombre haciendo clic en **Nueva** y agregarle en la opción **Valor de variable** la ruta completa de la carpeta creada en el punto 2.
8. Reiniciar el sistema.
9. Ejecutar el programa Servidor_rmi.jar haciendo doble clic en él, luego copie la ruta del ejecutable de GNU Octave en donde el programa indica, ejemplo: **C:\Archivos de programa\Octave\bin\octave-3.0.3.exe.**
10. Clic en el botón iniciar. En el caso de que al presionar el botón iniciar y el programa lance un mensaje de error diciendo que no puede iniciar, reinicie de nuevo el sistema operativo o repita los pasos del 4 al 8.

La herramienta de simulación se obtiene en el menú **Gráficos**, en la opción **Simular**, o por medio de la combinación de teclas **Alt+F1**, también está disponible en la barra de herramientas en el botón Simular representado por el primer icono de gráficos de izquierda a derecha. Una vez visualizada la herramienta de simulación se obtiene una ventana como la que se muestra en la Figura C.4.

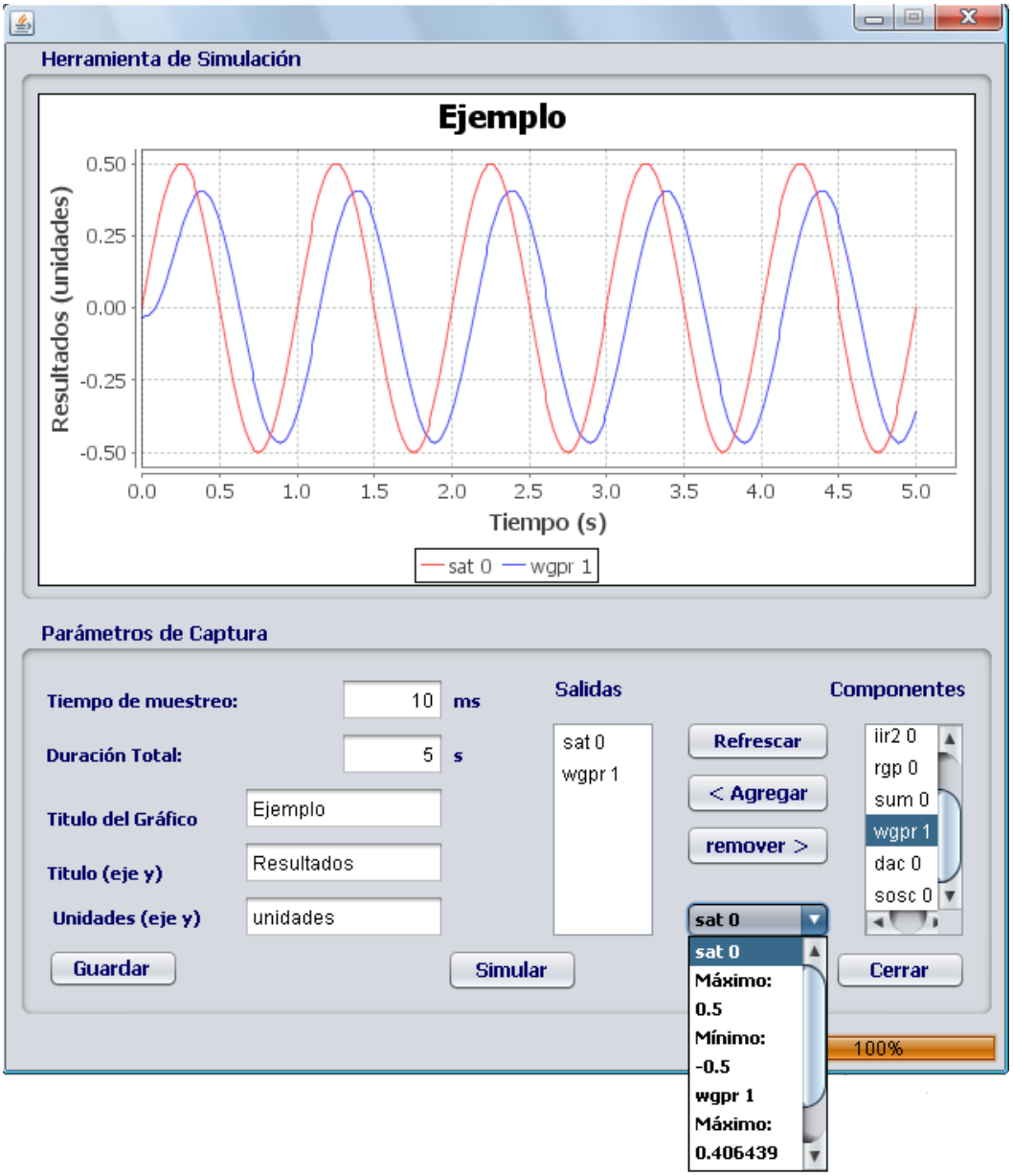

Figura C.4 Interfaz de la herramienta de simulación y resultados de una simulación.

En la ventana de la herramienta de simulación mostrada, se encuentran las siguientes opciones:

Tiempo de muestreo: Tiempo de muestreo para la simulación de datos en ms, debe ser un número entero entre 0 y 60000.

Duración Total: Tiempo de duración total para la simulación de datos en segundos, debe ser un número entero entre 0 y 255.

Titulo del Gráfico: Especifica el título del gráfico.

Titulo (eje y): Especifica el título del eje y.

Unidades (eje y): Especifica las unidades para los datos simulados.

Componentes: Contiene un lista de los componentes presentes en el área de trabajo.

Salidas: Contiene una lista de las señales que el usuario desea simular.

Refrescar: Refresca la lista de "Componentes" y "Salidas" en el caso de que el diseño en el área de trabajo haya sido modificado.

Agregar: Toma un componente seleccionado de la lista de "Componentes" y lo agrega a la lista de "Salidas".

Remover: Remueve un componente seleccionado de la lista de "Salidas".

Datos: Muestra los valores máximos y mínimos de las señales simuladas.

Guardar: brinda dos opciones de guardado, una para guardar los resultados de la simulación y la otra para guardar el código que representa la simulación en un archivo de extensión *.m.

Simular: Inicia la tarea de simulación.

Cerrar: Cierra la herramienta de simulación.

Capturar datos

La herramienta de captura de datos se obtiene en el menú **Gráficos**, en la opción **Capturar Datos**, o por medio de la combinación de teclas **Alt+F2**, también está disponible en la barra de herramientas en el botón Capturar datos representado por el segundo ícono de gráficos de izquierda a derecha. Una vez visualizada la herramienta de captura de datos se obtiene una ventana como la que se muestra en Figura C.5.

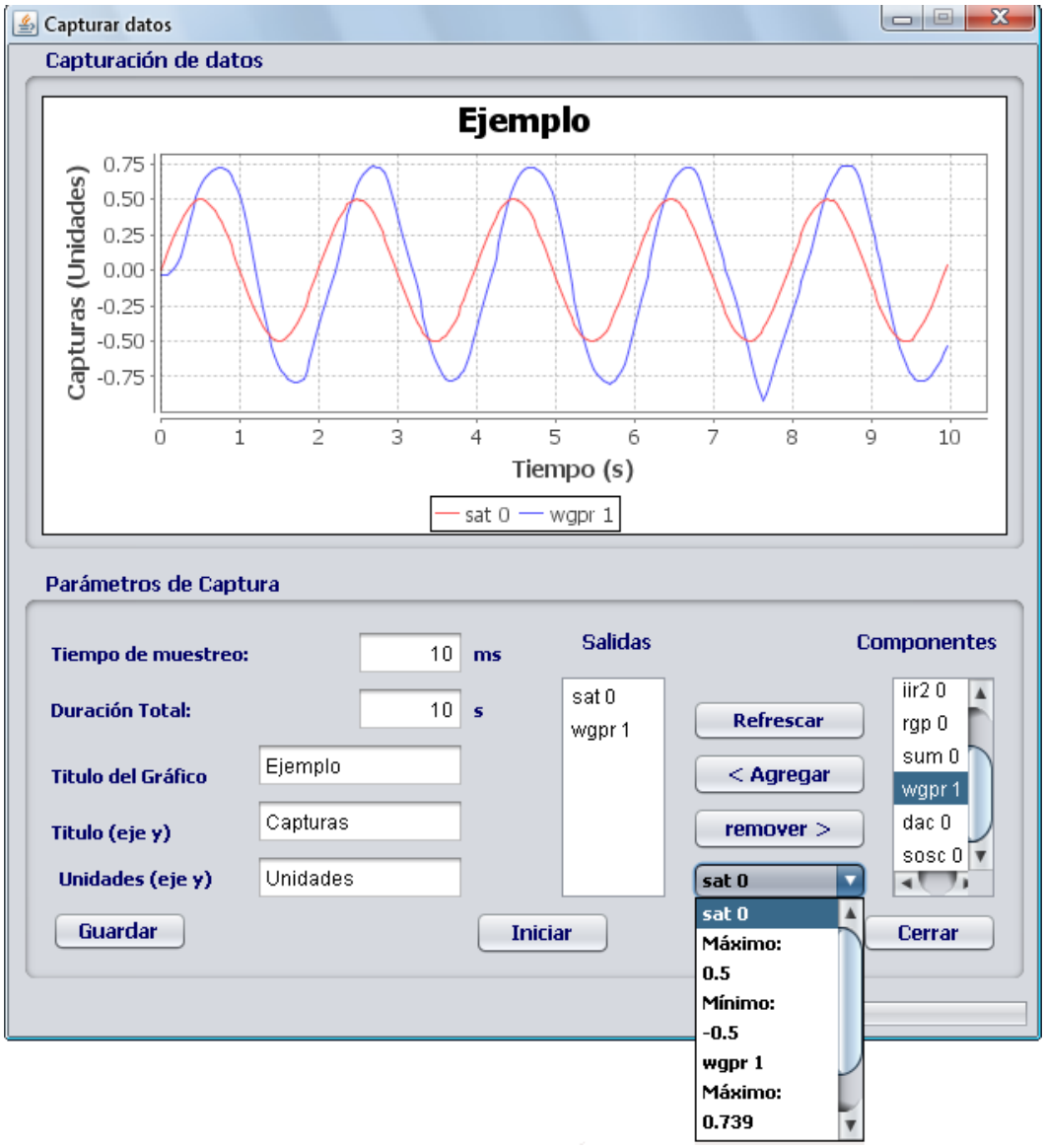

Figura C.5 Interfaz de la herramienta de captura de datos y resultados de una captura.

En la ventana de la herramienta de captura de datos mostrada, se encuentran las siguientes opciones:

Tiempo de muestreo: Tiempo de muestro para la captura de datos en ms, debe ser un número entero entre 0 y 60000.
Duración Total: Tiempo de duración total para la captura de datos en segundos, debe ser un número entero entre 0 y 255.
Titulo del Gráfico: Especifica el título del gráfico.
Titulo (eje y): Especifica el título del eje y.
Unidades (eje y): Especifica las unidades para los datos obtenidos.
Componentes: Contiene un lista de los "Componentes" presentes en el área de trabajo.
Salidas: Contiene una lista de las señales que el usuario desea capturar.
Refrescar: Refresca la lista de "Componentes" y "Salidas" en el caso de que el diseño en el área de trabajo haya sido modificado.
Agregar: Toma un componente seleccionado de la lista de "Componentes" y lo agrega a la lista de "Salidas".
Remover: Remueve un componente seleccionado de la lista de "Salidas".
Datos: Muestra los valores máximos y mínimos de las señales simuladas.
Guardar: guarda los datos obtenidos en un archivo de texto.
Iniciar: Inicia la tarea de captura.
Cerrar: Cierra la herramienta de captura de datos.

Osciloscopio flotante

La herramienta de Osciloscopio flotante se obtiene en el menú **Gráficos**, en la opción **Osciloscopio Flotante**, o por medio de la combinación de teclas **Alt+F3**, también está disponible en la barra de herramientas en el botón Osciloscopio flotante representado por el tercer ícono de gráficos de izquierda a derecha. Una vez visualizada la herramienta de Osciloscopio Flotante se obtiene una ventana como la que se muestra en la Figura C.6.

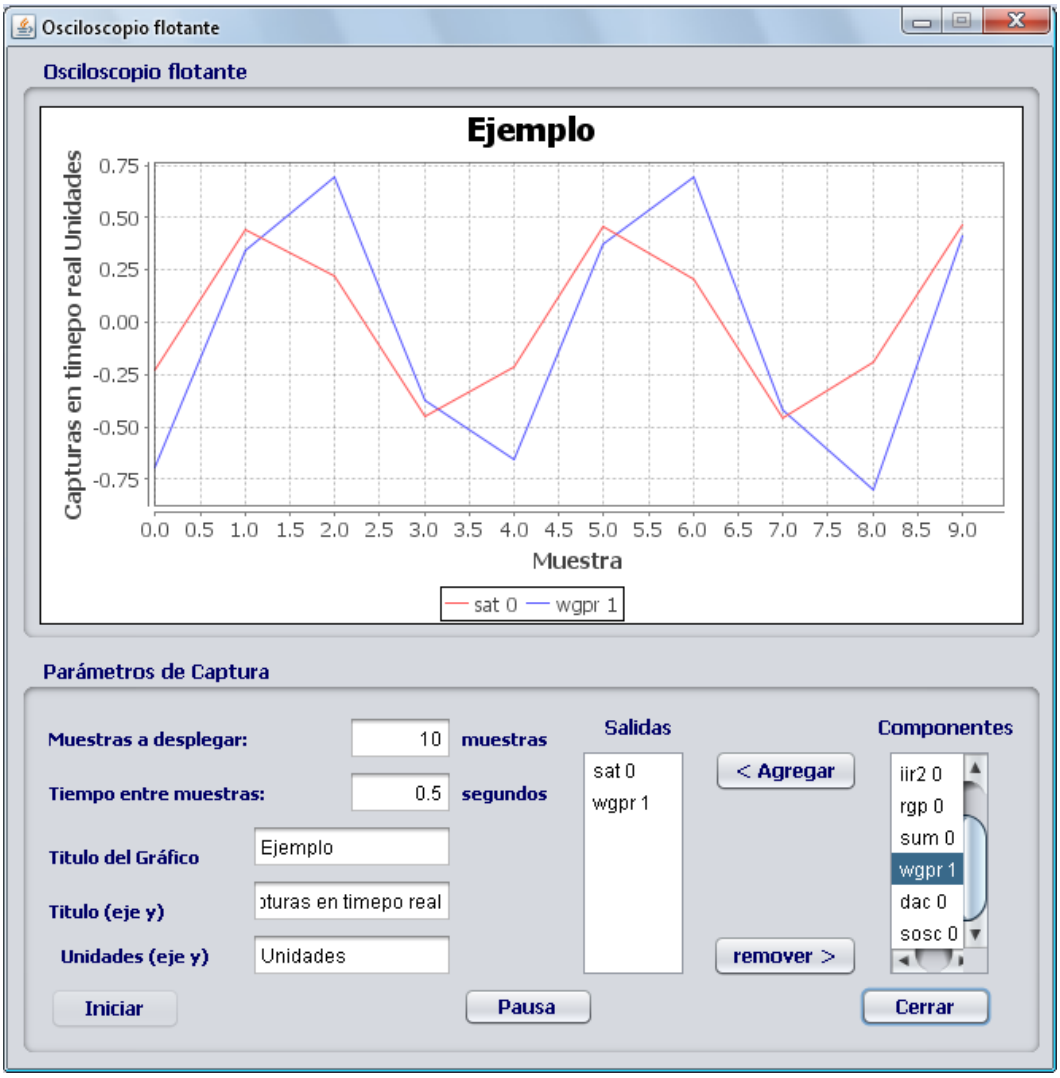

Figura C.6 Interfaz de la herramienta de Osciloscopio flotante.

En la ventana de la herramienta de captura de datos mostrada, se encuentran las siguientes opciones:

Muestras a desplegar: Cantidad de muestras a desplegar en el gráfico, este debe ser un número entero.

Tiempo entre muestras: Tiempo entre cada muestra.

Titulo del Gráfico: Especifica el título del gráfico.

Titulo (eje y): Especifica el título del eje y.

Unidades (eje y): Especifica las unidades para los datos obtenidos.

Componentes: Contiene un lista de los componentes presentes en el área de trabajo.

Salidas: Contiene una lista de las señales que el usuario desea visualizar.

Refrescar: Refresca la lista de "Componentes" y "Salidas" en el caso de que el diseño en el área de trabajo haya sido modificado.

Agregar: Toma un componente seleccionado de la lista de "Componentes" y lo agrega a la lista de "Salidas".

Remover: Remueve un componente seleccionado de la lista de "Salidas".

Iniciar: Inicia la tarea de visualización en tiempo real.

Pausa: Pausa la tarea de visualización en tiempo real.

Cerrar: Cierra la herramienta de visualización en tiempo real.

Compilador

La herramienta **Compilador** es una función que se utiliza con fines de depuración para el entorno de desarrollo integrado. Esta función se obtiene en el menú **Herramientas**, la opción **Compilador** o por medio de la combinación de teclas **Alt+C**, esta herramienta se visualiza en la Figura C.7.

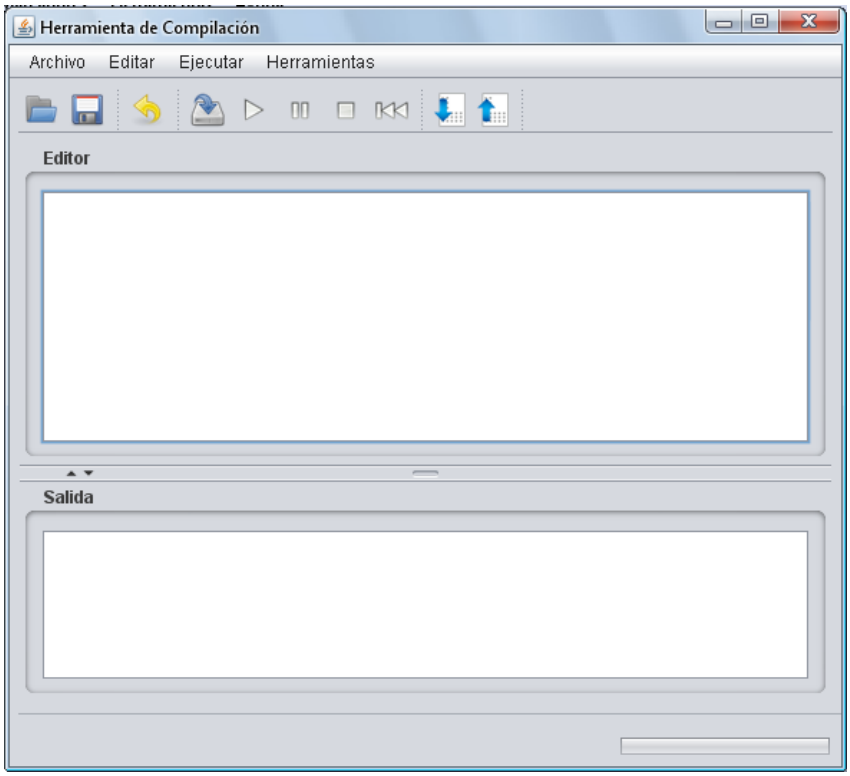

Figura C.7 Herramienta Compilador.

La interfaz de la herramienta de compilación contiene un área de texto llamada editor en la cual se escribe el código intermedio o de máquina, y un área de salida en la que los errores son reportados, normalmente los errores de traducción de código intermedio son reportados en esta área de salida, mientras que los errores de "Desensamblar" son reportados por medio de mensajes de errores desplegados en una ventana.

Contiene las siguientes funciones:

Menú Archivo:

Nuevo, Abrir, Guardar Como, Salir.

Menú Editar:

Deshacer.

Menú Ejecutar:

Programar dispositivo, Iniciar, Pausa, Detener, Reiniciar.

Menú Herramientas:

Compilar, Desensamblar.

En cuanto a las funciones referentes a **Menú Archivo**, **Menú Editar** y **Menú Ejecutar**, éstas realizan la misma función que las explicadas anteriormente.
El menú herramientas incorpora dos funciones propias de la herramienta de compilación, las cuales son:

Compilar: Esta función ejecuta directamente el traductor de código para analizar la gramática de un programa escrito en código intermedio, si existen errores en el código, estos son reportados indicando la línea y la columna donde se encuentra el error.

Desensamblar: Esta función toma código de máquina y lo traduce a código intermedio, reportando errores en el caso de que existan.

Para poder realizar la traducción de un código a otro debe asegurarse que el código esté guardado y contenido en un archivo llamado bpc.txt para el lenguaje intermedio y bpc.hex para código de máquina, esto debido a que por ahora el sistema solo reconoce los archivos con este nombre y extensión en los casos en que se desea **Compilar** o **Desensamblar**.

C.4 Funciones que soporta la simulación

En las siguientes tablas se especifican los módulos soportados por la Unidad Controladora, los que están en negrita han sido implementados en la herramienta de simulación.

Módulos de *hardware*

Tabla C.1 Módulos de entrada.

Mnemónico	Código hexadecimal
adc	0x01
rgp	0x02
i2cr	0x03
qeis	0x04
qeip	0x05
srf08	0x08

Tabla C.2 Módulos de salida.

Mnemónico	Código hexadecimal
dac	0xC0
pwm	0xC1
wgpr	0xC2
i2cw	0xC3

Módulos virtuales

Tabla C.3 Procesamiento de señales.

Mnemónico	Código hexadecimal
iir2	0x45
iir4	0x46
fir	0x47
notch60	0x41

Tabla C.4 Control Lineal.

Mnemónico	Código hexadecimal
k	0x40
pid	0x42
pi_d	0x43
I_pd	0x44
tf	0x48
delay	0x53

Tabla C.5 Control no lineal.

Mnemónico	Código hexadecimal
sat	0x4A

Tabla C.6 Operaciones matemáticas

Mnemónico	Código hexadecimal
sum	0x49
res	0x4F
gain	0x50
mul	0x51

Tabla C.7 Generadores de señal.

Mnemónico	Código hexadecimal
sosc	0x4B
rosc	0x4C
tosc	0x4D
vosc	0x4E
noise	0x52

D. Lista de Errores soportados

D.1 Errores Léxicos

Error: Caracteres no válidos

Ingreso de caracteres no validos (Caracteres que no corresponden a un número por ejemplo).

Error: Código inválido

Ingreso de caracteres no validos en el archivo *.hex al aplicar la función "Desensamblar" (Caracteres que no corresponden a un número por ejemplo).

D.2 Errores de Sintaxis

Error: ADC no seleccionado

Un bloque ADC sin canal físico asignado.

Error: DAC no seleccionado

Un bloque DAC sin canal físico asignado.

Error: PWM no seleccionado

Un bloque PWM sin canal físico asignado.

Error: Dispositivos no seleccionados

Error general cuando un bloque ADC, DAC o PWM no ha sido seleccionado.

Error: Elementos sin conectar

Error que se genera cuando aún existen elementos sin conectar en el área de trabajo y se intenta programar el dispositivo o compilar a texto.

Error: Creación de módulos no reconocido

Ocurre cuando en el archivo de configuración "bpc.hex" no se reconoce la instrucción 0x31 que corresponde a la sección de Creación de Módulos.

Error: "netlist" no reconocido

Ocurre cuando en el archivo de configuración "bpc.hex" no se reconoce el instrucción 0x16 que corresponde a la sección de "netlist", la cual representa las conexiones de los módulos creados.

Error: No hay componentes

Ocurre cuando se intenta programar el dispositivo o se desea compilar a texto y no hay componentes en el área de trabajo.

Error: Cantidad de dígitos excedida

Parámetros cuyos valores contienen más de 6 dígitos válidos.

Error: No hay señales a observar

Ocurre cuando se ejecuta el Osciloscopio flotante y no existe una estructura diseñada en el área de trabajo, por lo que no hay módulos ni señales que se puedan observar.

Error: No hay señales a simular

Ocurre cuando se ejecuta la herramienta de simulación y no existe una estructura diseñada en el área de trabajo, por lo que no hay módulos ni señales que se puedan simular.

Error: No hay señales a capturar

Ocurre cuando se ejecuta la herramienta de capturar datos y no existe una estructura diseñada en el área de trabajo, por lo que no hay módulos ni señales en los que se puedan capturar datos.

D.3 Errores de semántica

Error: Fuera de rango

Ingresar valores fuera del rango correspondiente válido según su parámetro.

La siguiente es una lista de los parámetros clasificados por su rango de valor correspondiente:

1. **(1 bytes) [0,255]:**
   "Ciclo de trabajo (0-100)"
2. **(2 bytes) [0,60000]:**
   "Tiempo de muestreo en ms"
   "Tiempo 1 en ms"
   "Tiempo 2 en ms"
   "Tiempo 3 en ms"
   "Tiempo máximo en ms"
   "Tiempo mínimo en ms"
   “Frecuencia PWM en Hz”
3. **(6 bytes) [0,+999999]:**
   "Frecuencia de corte en Hz"
   "Ancho de banda en Hz"
   "Frecuencia en Hz"
4. **(6 bytes) [0,+1]:**
   "Amplitud"
   "Amplitud positiva"
   "Radio de polos conjugados" [0,+1]
5. **(6 bytes) [-1,+1]:**
   "Ganancia"
   "Valor máximo"
   "Valor mínimo"
   "Amplitud negativa"
   "Valor"
   "Asignar Valor"

Error: No concordancia en conexiones

Error que se genera en los siguientes casos:

1. Conexión de una entrada de un bloque a la entrada de otro bloque.
2. Al conectar las salidas entre bloques diferentes.
3. Conexión entre patillas del mismo bloque.

D.4 Errores de uso de recursos físicos

Error: ADC en uso

Bloque ADC cuyo canal físico asignado ya está siendo utilizado por otro bloque ADC.

Error: DAC en uso

Bloque ADC cuyo canal físico asignado ya está siendo utilizado por otro bloque ADC.

Error: PWM en uso

Bloque PWM cuyo canal físico asignado ya está siendo utilizado por otro bloque PWM.

D.5 Errores de Comunicación y de Sistema

D.5.1 Errores de Comunicación

Error: Fallo de conexión

Fallo en la conexión TCP/IP hacia un dispositivo.

Error: Falló envío de comando

Error que se genera al no lograr enviar una instrucción de operación al dispositivo.

Posibles causas:

1. Fallo al establecer conexión con el dispositivo
2. El dispositivo no responde según el protocolo de comunicación esperado.

Error: Tiempo de espera agotado

Ocurre cuando el tiempo de espera para recibir datos del dispositivo se agota debido a un fallo en la comunicación.

Error: Inicio de comunicación

El dispositivo no envía el byte 0x53 que indica inicio de comunicación según el protocolo establecido.

Error: Instrucción no reconocida

El dispositivo no reconoce la instrucción solicitada por el usuario.

Error: Cierre de comunicación

El dispositivo no envió el byte 0x53 que indica cierre de comunicación.

Error: Parámetro LISTO no recibido

El dispositivo no envía el parámetro listo en un proceso de captura de datos.

Error: Programación fallida del dispositivo

Ocurre cuando no se logra programar correctamente el dispositivo electrónico debido a un fallo en el protocolo de comunicación.

D.5.2 Errores de Sistema

Error: Error de escritura

Error que ocurre al fallar la escritura de un archivo en memoria.

Error: Fallo en lectura de directorio

Error al intentar determinar la ruta del directorio actual.

Error: Archivo no válido

Error que se genera al no encontrar el archivo requerido, ya sea por abrir un archivo cuyo formato no es el que se requiere o simplemente el archivo no existe.

D.6 Mensajes de información

Información: Archivo bpc.txt generado

Indica el directorio donde se almacena el archivo bpc.txt, luego de programar el dispositivo o simplemente al utilizar la función "compilar a texto".

Información: Programar dispositivo

Advierte que la operación que se va a proceder a programar el dispositivo electrónico.

Información: Archivo existente

Advierte que el archivo que desea guardar ya existe.

Información: Dispositivo Reiniciado

Indica que la operación de reiniciar ha sido aplicada.

Información: Dispositivo no iniciado

Informa que la operación de iniciar dispositivo no se logró aplicar, normalmente ocurre debido un error de comunicación.

Información: D/A´s no inicializados

Indica que los dispositivos DAC no han sido inicializados.

Información: Datos recibidos

Indica que los datos de una captura han sido correctamente recibidos

Información: Programación de dispositivo

Indica que la Unidad Electrónica ha sido programada correctamente.

E. Especificación de la gramática del lenguaje intermedio

TOKENS

```
//*****************Definición de símbolos que indican comentarios*******
<DEFAULT> SKIP : {
" "
| "\t"
| "\n"
| "\r"
}


<DEFAULT> SKIP: {
"//" : EnComentarios1
}


<EnComentarios1> SKIP : {
"\n” : DEFAULT
}


<EnComentarios1> MORE : {
<~[]>
}


<DEFAULT> SKIP : {
"/*" : EnComentarios2
}


<EnComentarios2> SKIP : {
"*/" : DEFAULT
}
```

```
<EnComentarios2> MORE : {
<~[]>
}


//*****************************************************************
<DEFAULT> TOKEN : {
<START: "START">
| <STOP: "STOP">
| <PAUSE: "PAUSE">
| <ERASMEM: "ERASE">
| <SAVESIS: "SAVE">
| <RISET: "RESET">
| <USEUPC: "USECPU">
| <READ_REG: "READREG">
| <NET_LIST: "NETLIST">
| <SET_VALUE: "SETVALUE">
| <RANGE_ADC: "RANGEADC">
| <RANGE_DAC: "RANGEDAC">
| <READ_ADC: "READADC">
| <PWM_1D: "PWM">
| <FILTERR: "FILTER">
| <T_F: "TF">
| <P_I_D: "PID">
| <S_OSC: "SOSC">
| <R_OSC: "ROSC">
| <T_OSC: "TOSC">
| <V_OSC: "VOSC">
| <KF: "K">
| <MODULES: "MOD">
| <GAIN: "GAIN">
| <SA_T: "SAT">
| <NOISE_: "NOISE">
| <CAPTURE_: "CAPTURE">
| <CAPTURE_CE: "CAPTURECE">
| <CAPTURE_R: "CAPTURER">
| <RET_GPR: "READMOD">
| <SAVE_GPR: "RGP">
```

```
| <CONFIG: "CONFIG">
| <PUNTO: ".">
| <SIGNO: "+" | "-">
| <COMA: ",">
| <PUNTOYCOMA: ";">
| <INICIO: "begin">
| <FINAL: "end">
| <DEC_0_a_5: ["0"-"5"]>
| <DEC_0_a_7: <DEC_0_a_5> | ["6"-"7"]>
| <DEC_6_a_9: <DEC_0_a_7> | ["8"-"9"]>
| <DEC_0_a_9: <DEC_0_a_5> | <DEC_6_a_9>>
| <REGX: <DEC_0_a_9> | "1" | "1" <DEC_0_a_5>>
| <N: <DEC_0_a_5>>
| <IIR_2: "iir_2">
| <IIR_4: "iir_4">
| <FIR: "fir">
| <NOTCH: "notch">
| <ADC_01: "adc">
| <RGP_02: "rgp">
| <I2CR_03: "i2cr">
| <QEIS_04: "qeis">
| <QEIP_05: "qeip">
| <SRF08_08: "srf08">
| <K_40: "k">
| <FIRX_41: "firx">
| <PID_42: "pid">
| <PI_D_43: "pi_d">
| <I_PD_44: "i_pd">
| <IIR2_45: "iir2">
| <IIR4_46: "iir4">
| <NOTCH60_47: "notch60">
| <TF_48: "tf">
| <SUM_49: "sum">
| <SAT_4a: "sat">
| <SOSC_4b: "sosc">
| <ROSC_4c: "rosc">
| <TOSC_4d: "tosc">
| <VOSC_4e: "vosc">
| <RES_4f: "res">
```

```
| <GAIN_50: "gain">
| <MUL_51: "mul">
| <NOISE_52: "noise">
| <DELAY_53: "delay">
| <MSWITCH_54: "mswitch">
| <TSWITCH_55: "tswicth">
| <DAC_c0: "dac">
| <PWM_c1: "pwm">
| <WGP_c2: "wgpr">
| <I2CW_c3: "i2cw">
}
```

NO-TERMINALES

```
/*******Definicion y Gramatica del programa completo******************/
Programa ::= ( ( Modulos ) ( ConfiguraModulos ) ( Nettlist ) ) | <EOF>
/**********************************************************************/
/*****************Crea o destruye modulos******************************/
/**********************************************************************/
Modulos ::= <MODULES> <INICIO> ( ( Modulo Virtual ) <COMA> ( Valor )
<PUNTOYCOMA> )+ <FINAL>
/**********************************************************************/
/**********Configura los modulos creados*****************************/
/**********************************************************************/
ConfiguraModulos ::= <CONFIG> <INICIO> ( ( Comandos ) )+ <FINAL>
/**********************************************************************/
/***************Netlist (lista enlazada de conexion de modulos)********/
/**********************************************************************/
Nettlist ::= <NET_LIST> <INICIO> ( ( Modulo Virtual ) <COMA> ( Valor )
<COMA> ( Valor ) <COMA> ( Modulo Virtual ) <COMA> ( Valor ) <COMA> (
Valor ) <PUNTOYCOMA> )+ <FINAL>
/**********************************************************************/
Comandos ::=
(
<STOP> <PUNTOYCOMA>
| <START> <PUNTOYCOMA>
| <PAUSE> <PUNTOYCOMA>
| <ERASMEM> <PUNTOYCOMA>
| <SAVESIS> <PUNTOYCOMA>
```

| <RISET> <PUNTOYCOMA>
| <USEUPC> <PUNTOYCOMA>
| <READ_REG> (NumeroRango16) <PUNTOYCOMA>
| <SET_VALUE> (DEVICE DAC | DEVICE PWM) <COMA> (Constante) <PUNTOYCOMA>
| <RANGE_ADC> (DEVICE ADC) <COMA> (NumeroRango16)
(<COMA> (Tiempo muetreoADC))? <PUNTOYCOMA>
| <RANGE_DAC> (DEVICE DAC) <COMA> (NumeroRango8) <PUNTOYCOMA>
| <READ_ADC> (DEVICE ADC) <PUNTOYCOMA> | <PWM_1D> (DEVICE PWM) <COMA> (unibidi) <COMA> (offset) <COMA> (frecpwm) <PUNTOYCOMA>
| <FILTERR> (((<IIR_2> | <IIR_4>) <COMA> (Valor) <COMA> (Constante) <COMA> (Constante) COMA> (TFIIR2_4) <PUNTOYCOMA>)
| (<FIR> <COMA> (Valor) <COMA> (Constante) <COMA> (TFFIR) <PUNTOYCOMA>)
| (<NOTCH> <COMA> (Valor) <COMA> (Constante) <PUNTOYCOMA>))
|<T_F> (Valor) <COMA> (Tiempo ms) <COMA> (("pole" <COMA> (Constante) <COMA> (Polo Cero) <COMA> (Polo Cero) <COMA> (Polo Cero) <COMA> (Polo Cero))
| ("k_f" <COMA> (Constante) <COMA> (Constante) <COMA> (Constante) <COMA> (Constante) <COMA> (Constante))) <PUNTOYCOMA>
|<P_I_D> (DEVICE PID) <COMA> (Valor) <COMA> (Tiempo ms) <COMA> (Constante) <COMA> (Constante) <COMA> (Constante) <PUNTOYCOMA>
|<S_OSC> (Valor) <COMA> (Constante) <COMA> (Constante) <PUNTOYCOMA>
|<R_OSC> (Valor) <COMA> (Tiempo ms) <COMA> (Tiempo ms) <COMA> (Tiempo ms) <COMA> (Constante) <COMA> (Constante) <PUNTOYCOMA>
|<T_OSC> (Valor) <COMA> (Tiempo ms) <COMA> (Tiempo ms) <COMA> (Constante) <COMA> (Constante) <PUNTOYCOMA>
|<V_OSC> (Valor) <COMA> (Tiempo ms) <COMA> (Tiempo ms) <COMA> (Constante) <COMA> (Constante) <PUNTOYCOMA>
|<KF> (Valor) <COMA> (Tiempo ms) <COMA> (Constante) (<PUNTOYCOMA>
|(<COMA> (Constante) (<PUNTOYCOMA> | (<COMA> (Constante) (<PUNTOYCOMA> | (<COMA> (Constante) <PUNTOYCOMA>)))))
|<GAIN> (Valor) <COMA> (Constante) <PUNTOYCOMA>
|<SA_T> (Valor) <COMA> (Constante) <COMA> (Constante) <PUNTOYCOMA>
|<NOISE_> (Valor) <COMA> (Constante) <COMA> (Constante) <PUNTOYCOMA>
|<CAPTURE_> (Tiempo ms) <COMA> (Tiempo_s) (<COMA> (Modulo Virtual) <COMA> Num Bloque <COMA> (Valor))+ <PUNTOYCOMA>

```
|<CAPTURE_CE> ( "stop" | "start" | "ready" | "send" | "erase" )
<PUNTOYCOMA> | <CAPTURE_R> ( "m" | "m0" | "m1" ) <PUNTOYCOMA> | <RET_GPR>
( ( Modulo_Virtual) <COMA> ( Valor ) <COMA> ( Valor ) )+ <PUNTOYCOMA>
| <SAVE_GPR> ( Valor ) <COMA> ( Constante ) <PUNTOYCOMA> )
/*****************************************************************/
Modulo_Virtual ::=
(
<ADC_01>
|<RGP_02>
|<I2CR_03>
|<QEIS_04>
|<QEIP_05>
|<SRF08_08>
|<K_40>
|<FIRX_41>
|<PID_42>
|<PI_D_43>
|<I_PD_44>
|<IIR2_45>
|<IIR4_46>
|<NOTCH60_47>
|<TF_48>
|<SUM_49>
|<SAT_4a>
|<SOSC_4b>
|<ROSC_4c>
|<TOSC_4d>
|<VOSC_4e>
|<RES_4f>
|<GAIN_50>
|<MUL_51>
|<NOISE_52>
|<DELAY_53>
|<MSWITCH_54>
|<TSWITCH_55>
|<DAC_c0>
|<PWM_c1>
|<WGP_c2>
|<I2CW_c3>
```

```
)
/*************************************************************************/
Valor1 ::= ( ( <DEC_0_a_5> | <DEC_0_a_7> | <DEC_6_a_9> | <REGX> | "16" |
"100" ) )+
/*************************************************************************/
Valor_dec_Ascii ::= ( <DEC_0_a_5> | <DEC_0_a_7> | <DEC_6_a_9> | <REGX> |
"16" | "100" )
/*************************************************************************/
Valor ::= ( Valor1 )
/*************************************************************************/
Tiempo_ms ::= ( Valor1 )
/*************************************************************************/
Tiempo_s ::= ( Valor1 )
/*************************************************************************/
Num_Bloque ::= <DEC_0_a_5>
/*************************************************************************/
Polo_Cero ::= ( Constante ) <COMA> ( Constante )
/*************************************************************************/
Constante ::= ( <SIGNO> )? ( ( Valor dec Ascii ) )+ ( <PUNTO> ( (
Valor dec Ascii ) )+ )?
/*************************************************************************/
NumeroRango16 ::= ( <DEC_0_a_5> | <DEC_0_a_7> | <DEC_6_a_9> | <REGX> )
/*************************************************************************/
NumeroRango8 ::= ( <DEC_0_a_5> | <DEC_0_a_7> )
/*************************************************************************/
De_0_a_5 ::= <DEC_0_a_5>
/*************************************************************************/
DEVICE_DAC ::= "dac0" | "dac1"
/*************************************************************************/
DEVICE_PWM ::= "pwm0" | "pwm1" | "pwm2" | "pwm3"
/*************************************************************************/
DEVICE_ADC ::= "adc0" | "adc1" | "adc2" | "adc3"
/*************************************************************************/
DEVICE_PID ::= "pid0" | "pid1" | "pid2"
/*************************************************************************/
TFIIR2_4 ::= "butterwlp" | "butterwhp" | "besselIp" | "besselhp" |
"butterwbp" | "butterwrp"
/*************************************************************************/
TFFIR::= "16" | "32" | "64"
```

```
/*************************************************************************/
offset ::= ( ( ( <DEC_0_a_5> | <DEC_0_a_7> | <DEC_6_a_9> ) )? (
<DEC_0_a_5> | <DEC_0_a_7> | <DEC_6_a_9> ) | ( <REGX> ) | ( "16" ) | (
"100" ) )
/*************************************************************************/
unibidi ::= "ud" | "bd"
/*************************************************************************/
frecpwm ::= ( Valor1 )
/*************************************************************************/
Tiempo_muetreoADC ::= ( Valor1 )
```