RIELAC, Vol.XXXII 2/2011 p.65 - 77 Abril-Julio ISSN: 1815-5928

Herramienta para programar un controlador lógico programable basado en hardware reconfigurable”

Maikel Ramírez - Despaine
Eduardo Millán - Núñez
Valery Moreno – Vega

Recibido: Mayo 2011
Aprobado: Junio 2011

RESUMEN

El presente trabajo se basa en el desarrollo de una herramienta que permita programar un controlador lógico programable (PLC) basado en hardware reconfigurable. El desarrollo de la aplicación está sustentado por una de las metodologías ágiles descrita por XP, utilizando las técnicas de modelación establecidas por el Lenguaje Unificado de Modelado (UML) y haciendo uso de una potente herramienta de programación y diseño de interfaces gráficas como es Qt. La programación del PLC se realiza mediante el lenguaje de escalera, atendiendo la norma IEC 61131-3.

Palabras claves: controlador lógico programable, lenguaje escalera, Microblaze, traducir.

ABSTRACT

This work is based on developing a tool that allows programming a programmable logic controller (PLC) based on reconfigurable hardware. Application development is supported by an agile methodologies described by XP, using modeling techniques established by the Unified Modeling Language (UML) and using a powerful programming tool and graphical interface design as Qt. PLC programming is done using the ladder language, following the standard IEC 61131-3.

Key words: programmable logic controller, ladder, Microblaze, translate.

Tool to programming a programmable logic controller based on reconfigurable hardware.

INTRODUCCION

Los PLCs son importantes dispositivos de control industrial basados en tecnología microprocesador. Características del PLC tales como alta generalidad y flexibilidad, fácil de programar y usar, lo hacen ampliamente utilizado en la industria[1].

Existen en el mundo importantes fabricantes de PLC: Siemens, Allen Bradley, Omron, entre otros. Todos ellos brindan la aplicación que permite programarlo utilizando uno o varios lenguajes de programación de PLC establecidos en la norma IEC 61131-3.

Las técnicas de manufacturas han evolucionado con el tiempo, desde la producción masiva de ayer a la automatización flexible de hoy y a la automatización reconfigurable de mañana[2].

El crecimiento de esquemas complejos de control obliga a ingenieros e investigadores a explorar nuevas arquitecturas. El hardware reconfigurable, específicamente las FPGAs, han surgido como una alternativa a la demanda de aplicaciones[3].

Como tema de tesis en la 6ta edición de la maestría de Sistemas Digitales se desarrolló un Controlador Lógico Programable basado en hardware reconfigurable, que utiliza como núcleo de procesamiento al procesador Microblaze, de 32 bit, tecnología RISC y optimizado para las FPGA de Xilinx; se hace necesario desarrollar una herramienta que permita programarlo mediante un lenguaje de programación de PLC.

El objetivo de este trabajo es: Desarrollar la aplicación de escritorio que permita programar el controlador lógico programable basado en hardware reconfigurable mediante lenguaje escalera.

La estructura de este trabajo es la siguiente: en la sección 1 se hace una introducción a los lenguajes de programación de PLC, la sección 2 trata sobre las fases del proceso de compilación, en la sección 3 se presentan las características y descripción de la herramienta, en la sección 4 se explican los resultados alcanzados y finalmente en la sección 5 se abordan las conclusiones y recomendaciones.

1. Introducción a los Lenguajes de programación de PLC

Al igual que los PLC se han desarrollado y expandido, los lenguajes de programación se han desarrollado junto a ellos. Los lenguajes de programación permiten al usuario entrar un programa de control dentro del PLC usando una sintaxis establecida. Los lenguajes actuales tienen instrucciones más versátiles, las cuales proporcionan más poder computacional por cada operación ejecutada por ellas[1]. Estas instrucciones expanden las posibilidades de programación en áreas como: manipulación de datos, comunicación en redes, transferencia de datos, y control del flujo de programa, solo por mencionar algunas[4].

Los lenguajes de programación para PLC son de dos tipos, visuales y escritos. Los visuales admiten estructurar el programa por medio de símbolos gráficos, similares a los que se utilizan para describir los sistemas de automatización, planos esquemáticos y diagramas de bloques. Los escritos son listados de sentencias que describen las funciones a ejecutar[1].

Los programadores de PLC poseen formación en múltiples disciplinas y esto determina que exista diversidad de lenguajes. Los programadores de aplicaciones familiarizados con el área industrial prefieren lenguajes visuales, por su parte quienes tienen formación en electrónica e informática optan, inicialmente por los lenguajes escritos.

- ✓ Lenguajes visuales :

  Utilizan los símbolos de planos esquemáticos y diagramas de bloques. El acceso a los recursos se ve restringido a los símbolos que proporciona el lenguaje. En este grupo se destacan los lenguajes LD, SFC y FBD.

- ✓ Lenguajes escritos:

  Utilizan sentencias similares a las de programación de computadoras. Brindan acceso total a los recursos de programación. En este grupo se destacan los lenguajes LI y ST.

Todos ellos tienen la finalidad de generar código objeto para que sea ejecutado en la CPU del PLC.

2. Proceso de compilación

Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es lenguaje de máquina, pero también puede ser simplemente texto. Este proceso de traducción se conoce como compilación[5].

Un compilador es un programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior (típicamente lenguaje de máquina). De esta manera un programador puede diseñar un programa en un lenguaje mucho más cercano a cómo piensa un ser humano, para luego compilarlo a un programa más manejable por una computadora.

El proceso de compilación se divide en seis fases fundamentales.

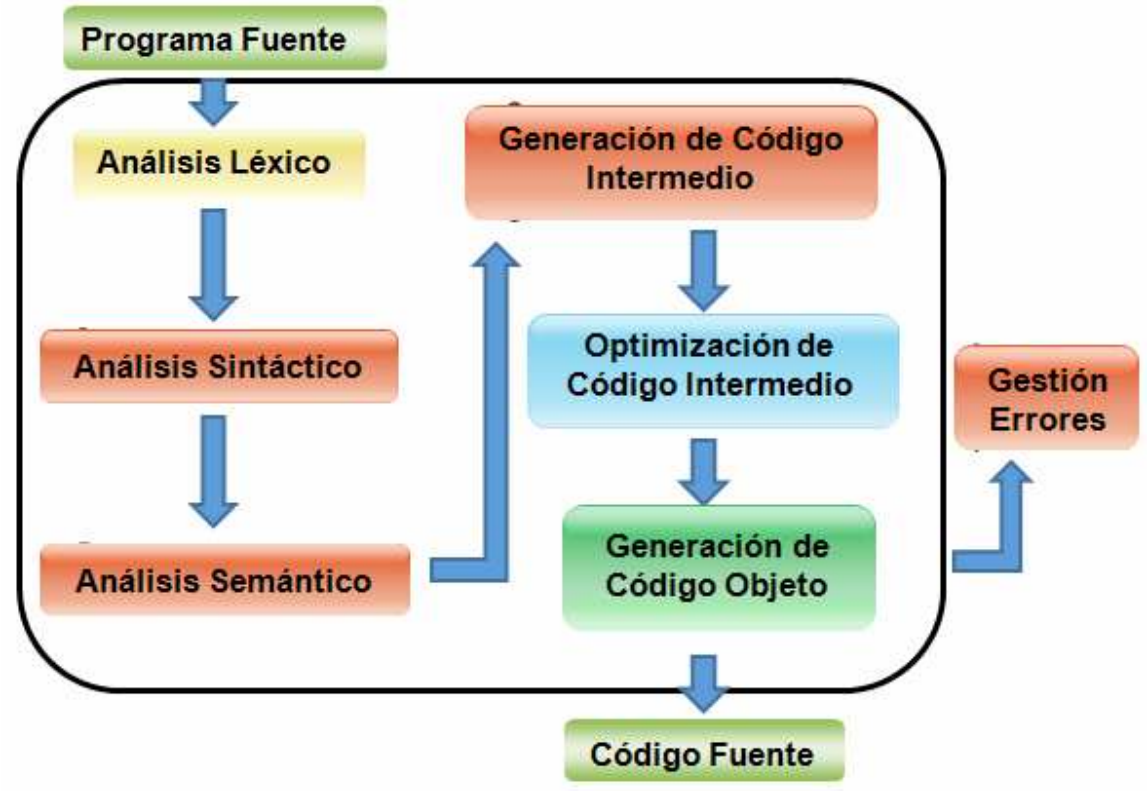


Figura 1 Fases del proceso de compilación.

La herramienta de programación del PLC solo tiene 3 fases:

- Análisis sintáctico.
- Análisis semántico.
- Generación de código intermedio.

En cada fase se realiza la gestión de errores.

3. Herramienta de programación del PLC reconfigurable

La herramienta brindará facilidades para implementar diagramas escalera, permitiendo la ejecución de los mismos y realizando un chequeo de la sintaxis, con el fin de generar una lista de errores en caso de que existan, ayudando al usuario en su identificación y corrección. El código generado a partir de la generación del código intermedio será ejecutado por el procesador Microblaze del PLC reconfigurable.

3.1. Requerimientos de la herramienta

Un requerimiento: es una condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema, para satisfacer un contrato, estándar, u otro documento impuesto formalmente[6].

Requerimientos funcionales

Un requerimiento funcional define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los historias de usuario serán llevados a la práctica. Los requerimientos funcionales se mantienen invariables sin importar con qué cualidades o propiedades se relacionen.

A continuación se muestran los requerimientos funcionales:

- ✓ R.1 El sistema debe brindar un mecanismo de almacenamiento para los valores de los componentes del programa en desarrollo.
- ✓ R.2 El sistema debe ser capaz de guardar ficheros con los datos del programa en desarrollo.
- ✓ R.3 El sistema debe ser capaz de cargar ficheros con datos de programas que hayan sido guardados con anterioridad.
- ✓ R.4 El sistema debe ser capaz de informar los diferentes errores encontrados durante el chequeo del programa en desarrollo.
- ✓ R.5 El sistema debe ser capaz de reconocer e interpretar las diferentes instrucciones creadas por el usuario.
- ✓ R.6 El sistema debe ser capaz de exportar y traducir los datos del programa desarrollado a funciones en lenguaje C.
- ✓ R.7 El sistema debe brindar la posibilidad de crear nuevos programas.
- ✓ R.8 El sistema debe brindar la posibilidad de introducir, cambiar y consultar los datos del programa en desarrollo en cualquier instante de tiempo.

Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. En muchos casos, estos requerimientos son fundamentales en el éxito del producto y generalmente están vinculados a requerimientos funcionales. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Requerimientos de Usabilidad

- ✓ El sistema deberá tener una interfaz amigable con una buena utilización de los elementos de diseño, con adecuada combinación de colores.
- ✓ El sistema deberá ser usado por usuarios capacitados para el uso de la herramienta y que hayan leído previamente el manual de ayuda.

Requerimientos de Soporte

- ✓ El sistema debe ser capaz de dar las mismas salidas para diferentes ambientes.
- ✓ El sistema debe ser flexible ante la necesidad de cambios de sus salidas.
- ✓ El sistema debe tener alguna documentación o ayuda.
- ✓ Los errores del sistema no pueden afectar a los sistemas clientes.

Requerimientos de portabilidad y operatividad

- ✓ El sistema debe ser compatible con las plataformas Windows y Linux.

Requerimientos de Hardware del Sistema

- ✓ Las computadoras que utilizarán el software a desarrollar deberán tener 128 MB de Memoria tipo RAM como mínimo.

Requerimientos de Software del Sistema

- ✓ Las computadoras que utilizarán el software deben tener instalado: Windows XP Professional, Windows 7 ó alguna distribución GNU/Linux.
- ✓ Qt 4.7.0 o superior.

3.2. Actores del sistema.

Tabla 1. Actores del sistema.

Actor	Descripción
Usuario	Representa a una persona capaz de desarrollar un programa en lenguaje escalera, accediendo a todas las funcionalidades que brinde la aplicación.

3.3. Definición de historias de usuario.

Las historias de usuario es la técnica utilizada en XP para especificar los requisitos del software[7]. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.
El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.
Las HU conducen al proceso de creación de los test de aceptación, los cuales servirán para verificar que estas historias se han implementado correctamente.

3.4. Descripción de las historias de usuario.

Tabla 2. HU Guardar programa.

Historia de Usuario	
Número: 1	**Nombre de la Historia de Usuario:** Guardar programa
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 2
Descripción: Permite crear un fichero XML que contiene la información del programa que se encuentra en desarrollo (imágenes y datos modificados por el usuario), el fichero es generado mediante la biblioteca QXMLStreamWriter().	
Observaciones: La creación del fichero de tipo XML le permitirá al usuario contar con un respaldo en caso de cualquier situación extrema, además de acceder a los datos registrados de manera organizada en la estructura definida dentro del código de la aplicación.	

Tabla 3. HU Abrir programa.

Historia de Usuario	
Número: 2	**Nombre de la Historia de Usuario:** Abrir programa
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 2
Descripción: Permite abrir un fichero XML que contiene información de un programa guardado previamente (imágenes y datos), el fichero es cargado mediante la biblioteca que se encarga de parsear el XML, QXMLStreamReader();	

Observaciones: Cualquier aplicación que trabaje con XML necesita un módulo de clases, su función es leer documentos y proporcionar acceso a su contenido y estructura. Para poder llevar a cabo esta función, la aplicación debe identificar la información XML y cómo se encuentra almacenada esta a través de un DTD (Document Type Declaration).

Tabla 4. HU Analizar Sintaxis.

Historia de Usuario	
Número: 3	**Nombre de la Historia de Usuario:** Analizar Sintaxis
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 2	**Iteración Asignada:** 1
Descripción: Permite parsear el diagrama desarrollado por el usuario, ayudándolo a identificar los posibles errores cometidos durante la creación del mismo y habilita la opción de generar el código intermedio correspondiente al programa escrito, una vez corregidos todas las fallas.	
Observaciones: El proceso de análisis de la sintaxis, es el resultado del parseo del diagrama en cuestión. Permite la generación de una lista de errores en caso de que existan (HU. No.4). Una vez finalizada esta acción habilita la opción de generar el código intermedio, solo si la primera funcionalidad no arrojó fallo alguno. Este código intermedio será exportado y finalmente descargado en un dispositivo externo (HU. No.5).	

Tabla 5. HU Generar lista de errores.

Historia de Usuario	
Número: 4	**Nombre de la Historia de Usuario:** Generar lista de errores
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 1
Descripción: Permite mostrar al usuario los errores cometidos durante la creación del programa.	

Observaciones: Posibilita el chequeo del diagrama mediante un algoritmo desarrollado, que tiene como fin arrojar un resultado (lista de errores) entendible para el usuario. Esta actividad forma parte de una de las principales funcionalidades de la aplicación (HU. No.3).

Tabla 6. HU Generar código intermedio.

Historia de Usuario	
Número: 5	**Nombre de la Historia de Usuario:** Generar código intermedio.
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 3
Descripción: Permite traducir el diagrama desarrollado en un lenguaje de alto nivel (C), con la finalidad de hacerlo entendible a dispositivos externos que manejen este tipo de información.	
Observaciones: Posibilita la generación de un código en otro lenguaje de mayor nivel (C), que facilite la comunicación entre el usuario y la maquina. Esta actividad forma parte de una de las principales funcionalidades de la aplicación (HU. No.3), y es dependiente de otra funcionalidad (HU. No.4).	

Tabla 7. HU Crear nuevo programa.

Historia de Usuario	
Número: 6	**Nombre de la Historia de Usuario:** Crear nuevo programa.
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 2
Descripción: Permite crear nuevos programas dentro de la aplicación.	
Observaciones: Habilita el editor para introducir nuevas sentencias con el fin de crear un nuevo diagrama, limpia todas las variables involucradas en el proceso y da la posibilidad de guardar los datos en caso de que se estuviese desarrollando un programa anterior.	

Tabla 8. Introducir, cambiar y consultar datos.

Historia de Usuario	
Número: 7	**Nombre de la Historia de Usuario:** Introducir, cambiar y consultar datos.
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 1
Descripción: Permite introducir, cambiar y consultar datos en la aplicación.	
Observaciones: Es una de las principales funciones que realiza el usuario en la aplicación, dándole características particulares al programa desarrollado, todas las acciones que encierra esta funcionalidad se realizan mediante un inspector de propiedades.	

3.5. Descripción de la herramienta

La aplicación actualmente se encuentra en su versión 1.0, consta de una ventana principal compuesta por tres paneles principales, anexo 6; el panel 1 ubicado en la parte izquierda se encarga de ubicar de forma organizada los diferentes componentes a los que el usuario tiene acceso para construir el programa en lenguaje escalera; el panel 2 corresponde al editor principal, parte en la cual el usuario construye los diferentes programas que desee, este editor brinda de forma automática un algoritmo de organización que ubica a cada componente en un área determinada con el fin de evitar desorganización y mantener en una misma línea los componentes que el usuario va insertando en caso de desearlo; el panel 3 se ubica en la parte inferior de la ventana, y es el área destinada a mostrar los errores contenidos en el programa en desarrollo.

Los componentes que brinda la aplicación como medio para desarrollar los programas en lenguaje escalera constan de una serie de propiedades definidas que permiten introducir atributos a las funciones a las cuales será traducido el programa, anexo 2, a estas propiedades se accede de manera dinámica a través de una ventana de propiedades que poseen todos los componentes, estas propiedades poseen elementos particulares en correspondencia con el tipo de componente y elementos comunes para todos los componentes como son, el identificador (único en cada caso), la posición que ocupa en el editor y el cuadrante en el que se encuentra, parámetros que ubican al usuario a la hora de conocer detalles de un elemento determinado.

Existen un conjunto de reglas a tener en cuenta a la hora de construir un programa, entre ellas se encuentran:

- Cada instrucción debe de comenzar en el primer cuadrante de cada renglón horizontal, de igual forma la primera línea del programa a desarrollar debe de esta ubicada en el primer renglón del editor, esta medida permite utilizar el espacio del editor al máximo, evitando dejar espacios innecesarios que podrían ser utilizados para la inserción de algún componente necesario en el programa desarrollado.
- Los componentes que forman parte de condiciones entre los que encontramos los contactos abiertos y cerrados estarán ubicados en la parte izquierda del editor, por su parte los componentes que son instrucciones o acciones a ejecutar van a estar ubicados en la parte derecha del editor, con el fin de mantener el orden del flujo de datos, ya que es una de las principales características del lenguaje escalera que las lecturas se realicen siempre de izquierda a derecha y de arriba hacia abajo.
- Las componentes tipo líneas no poseen características particulares, aunque si son necesarias ya que permiten continuar el flujo de información hacia que componente sigue detrás de otro. Se encuentran varios tipos de líneas, es importante señalar las líneas tipo T, permiten la creación de condiciones OR, estas líneas siempre deben de estar seguidas por líneas tipo L, que se encargaran de enlazar los componentes a analizar en cada condición.

- Los componentes como los Contadores y Temporizadores, siempre deben de ir ubicados al final de cada línea, para mantener el orden en el editor y ajustarse a las reglas de la gramática implementada, en caso contrario se arrojará un error indicándole al usuario de la incorrecta ubicación del componente.
- Un componente muy importante es el componente tipo FIN, el mismo debe de estar presente al final de cada programa construido, y esto en gran medida se debe a que en la implementación del algoritmo de traducción del lenguaje escalera al lenguaje C, siempre se chequea siguiendo el flujo de datos el próximo elemento en la cadena, este componente, le indica a este algoritmo que la secuencia de datos llega hasta el punto especificado y que debe de terminar la espera para pasar a realizar otras operaciones como la impresión por pantalla de las funciones nativas de cada componente en el lenguaje final (C).

El editor permite ubicar 11 elementos en secuencia continua, espacio que en acciones no es suficiente para una instrucción determinada, esto no quiere decir que las instrucciones deben tener máximo 11 componentes, pues se le da la posibilidad al usuario de continuar la secuencia de la instrucción en la línea inmediata inferior y no se altera el resultado de la misma.

Los datos que forman parte de atributos pueden ser cambiados en cualquier instante de tiempo, permitiéndole al usuario tener una interacción dinámica con la aplicación e ir probando diferentes juegos de datos para un mismo programa, estos datos se actualizan de forma automática en el lenguaje C generado finalmente.

El mecanismo de salvar un programa se realizó usando bibliotecas que permitieran la interacción con XML, este formato permite generar ficheros con una estructura determinada que le permitan al usuario acceder de forma rápida y eficiente a los datos guardados e incluso realizar cambios en estas plantillas y serán actualizados en el código C una vez cargado el fichero en la aplicación, la organización de los datos esta creada en forma de árbol jerárquico para una mayor comprensión de los mismos.

4. Resultados y discusión.

La contribución de este trabajo se basa en la obtención de una herramienta capaz de editar programa en lenguaje PLC, específicamente escalera, analizar su sintaxis y generar un código intermedio a partir del programa desarrollado por el usuario para que pueda ser ejecutado por el procesador Microblaze del PLC.

Con la culminación de de este trabajo podemos arribar a algunas conclusiones teórico-prácticas tales como:

- ✓ Ahorro en tiempo y recursos para las personas que posteriormente interactúen con el PLC en cuestión.
- ✓ Fácil mantenimiento, de modo que se le puedan adicionar nuevos módulos a la aplicación.
- ✓ Fácil uso debido al desarrollo de un conjunto de plantillas o contenidos prediseñados.
- ✓ Reutilización de código y componentes en proyectos con características similares.
- ✓ Utilización de la herramienta en diferentes sistemas operativos.

5. Conclusiones y Recomendaciones.

El presente trabajo se enfocó en el estudio y desarrollo de una aplicación de usuario capaz de compilar y traducir programas desarrollados en lenguaje gráfico (LD) a lenguajes escritos de tipo C. El sistema cuenta con la documentación necesaria para su entendimiento, mantenimiento y posterior escalabilidad debido al uso de la metodología de desarrollo ágil XP, que permitió construir los artefactos requeridos.

Los objetivos planteados para el desarrollo de este trabajo fueron alcanzados en el mismo, aunque se considera necesario continuar su perfeccionamiento con vistas a incrementar las prestaciones de la herramienta. A continuación se exponen un conjunto de recomendaciones que consideramos importantes a tener en cuenta para futuras versiones:

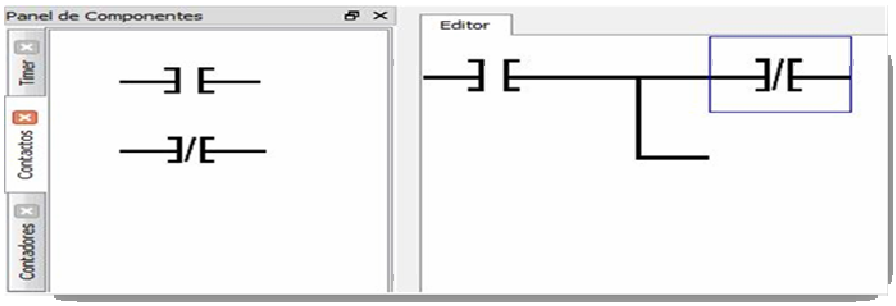
- ✓ Continuar el perfeccionamiento del diseño del editor, con el fin de hacer más fácil su uso.
- ✓ Desarrollar nuevos módulos con el objetivo de generar códigos intermedios en diferentes lenguajes.
- ✓ Dotar la aplicación de un modulo de simulación que permita el chequeo del programa.

REFERENCIAS

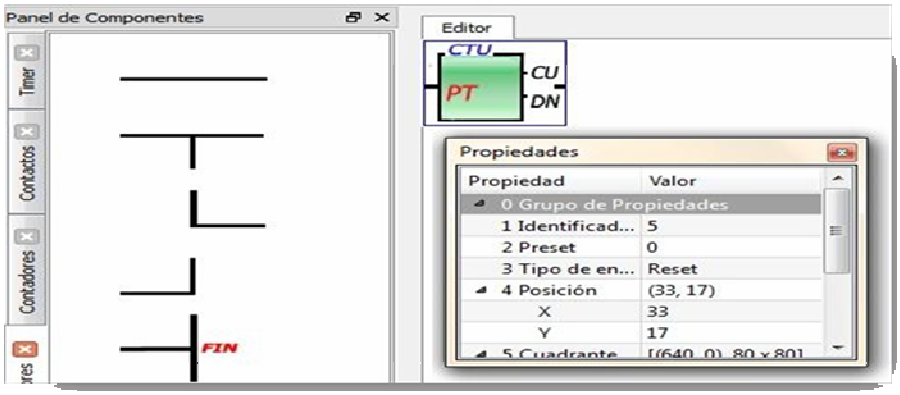
1. Bolton, W., *Programmable Logic Controllers, Fourth Edition*. Fourth ed. 2006: Elsevier Newnes.
2. Kulkarni, R., *The rise of embedded technologies is allowing control engineers greater flexibility in designing control applications.* IEE Computing & Control Engineering, 2006.
3. Carlos Paiz, M.P., *The Utilization of Reconfigurable Hardware to Implement Digital Controllers: a Review.* IEEE Industrial Electronics, 2007.
4. Bryan, L.A.B.a.E.A., *Programmable Controllers Theory and Implementation*. Second ed. 1997: An Industrial Text Company Publication.
5. http://es.wikipedia.org/wiki/Compilador. [cited.
6. Craig, L., *UML y Patrones Introducción al Análisis y Diseño Orientado a Objeto*. México : Person. 1999.
7. Ferrer, J., *Metodologías Ágiles.* [Online] http://libresoft.dat.escet.urjc.es/html/downloads/ferrer-20030312.pdf.

Anexos

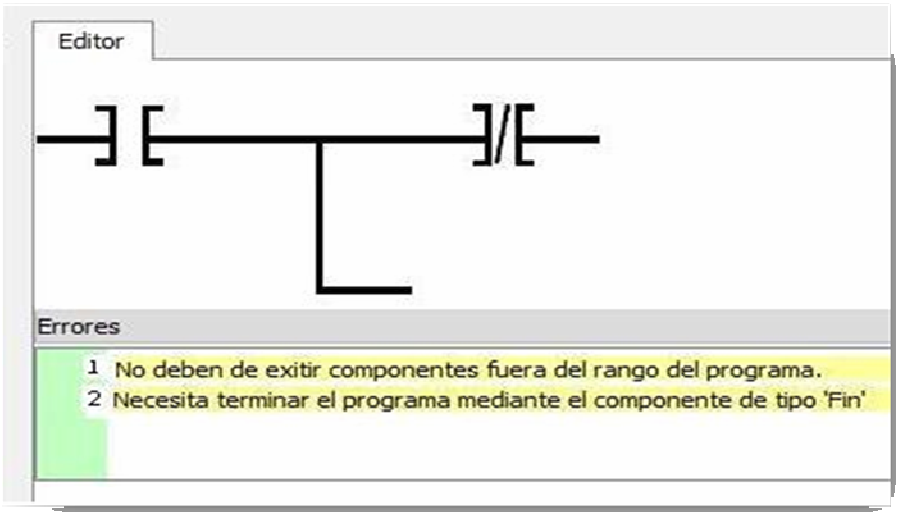
Anexo#1. Gestionar diagramas escalera.



Anexo#2 Gestionar las propiedades de cada componente.



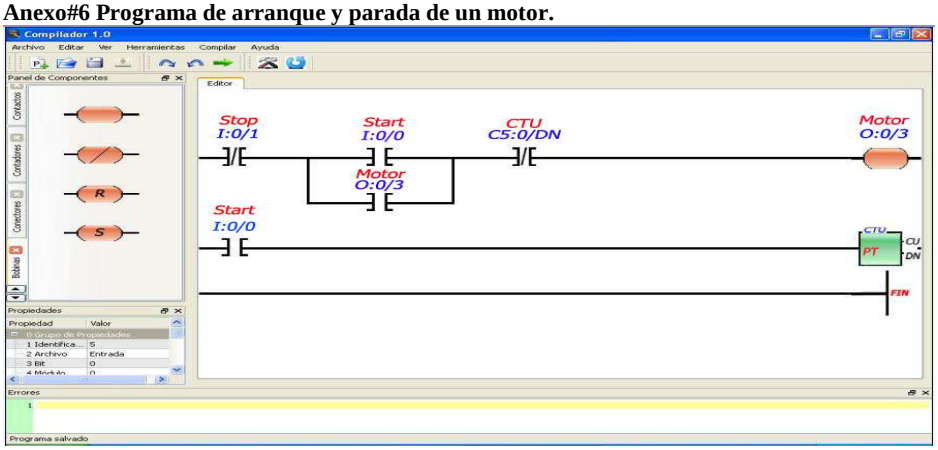
Anexo#3 Gestionar los errores contenidos en los diagramas desarrollados.



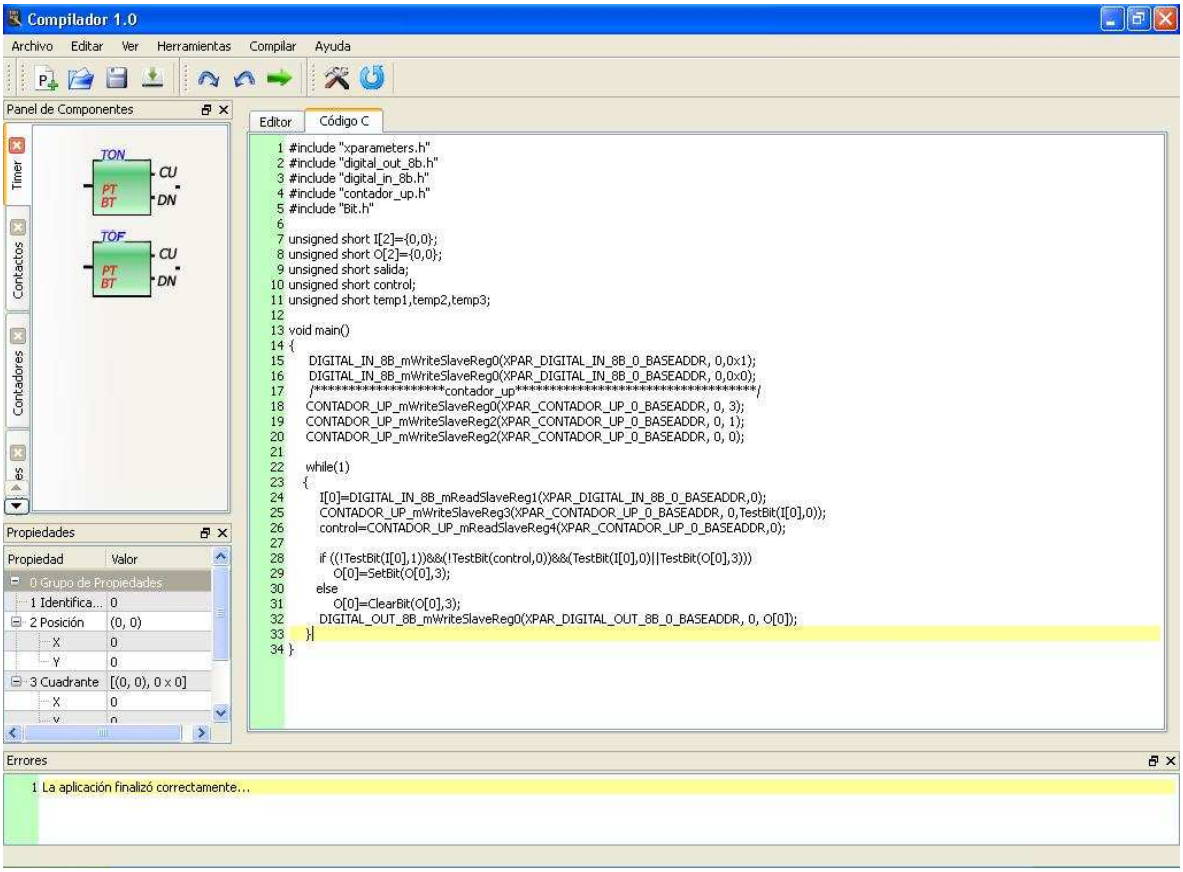
Anexo#4 Gestionar mecanismos de salvar y abrir ficheros de programas desarrollados.

Anexo#5 Gestionar los mecanismos de generar y exportar código intermedio.

Anexo#6 Programa de arranque y parada de un motor.



Anexo#7 Código en lenguaje C obtenido del anexo#6 “Programa de arranque y parada de un motor”.



Datos de los autores

Maikel Ramírez Despaine
Ingeniero en Control Automático, Profesor Asistente, Departamento de Sistemas Digitales de la facultad 2 de la Universidad de Ciencias Informáticas, La Habana, Cuba
e-mail: mike@uci.cu

Valery Moreno Vega
Ingeniero en Máquinas Computadoras, Máster en Informática Aplicada, Doctor en Ciencias Técnicas,Profesor Titular, Departamento de Automática de la Facultad de Eléctrica del Instituto Superior Politécnico “José Antonio Echeverría”, La Habana, Cuba.
e-mail: valery@electrica.cujae.edu.cu