Título: Módulo HMI para una tarjeta basada en microcontroladores.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores:
Karel Delgado Alón
Alejandro Jiménez López

Tutor: Ing. Antonio Cedeño Pozo
Co-tutor: Ing. Daisy Diana Vargas Vento

La Habana
2012

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Informática Industrial de la Universidad de las Ciencias Informáticas a que haga uso del mismo.

Para que así conste, firmamos la presente a los _____ días del mes de _____ del año _____.

Alejandro Jiménez López

_____
Firma del autor

Karel Delgado Alón

_____
Firma del autor

Ing. Antonio Cedeño Pozo

_____
Firma del tutor

Ing. Daisy Diana Vargas Vento

_____
Firma del co-tutor

Datos de contacto

Ing. Antonio Cedeño Pozo (acedeno@uci.cu)

Graduado con el título de Ingeniero en Ciencias Informáticas en el 2009, en la Universidad de las Ciencias Informáticas.

Ing. Daisy Diana Vargas Vento (ddvargas@uci.cu)

Graduada con el título de Ingeniero en Ciencias Informáticas en el 2011, en la Universidad de las Ciencias Informáticas.

Dedicatoria

A mis seres incondicionales: mi madre, mi padre y mi hermanita.

Alejandro Jiménez López

A la mejor hermana del mundo, a mi madre y a mi padre.

Karel Delgado Alón

Agradecimientos

A mi madre, por apoyarme siempre en mis decisiones. Por ser una excelente madre.

A mi padre, que me inició en el mundo de la computación. Por ser un ejemplo a seguir para mí.

A mi amigo Karel, por aconsejarme y soportarme siempre.

A mi amigo Jesús, por venir arrastrando con él desde el técnico y compartir siempre cuarto conmigo.

A mis amigos de siempre, aún cuando estén lejos: Aimara, Diana, Darién, Mauricio, Rebeca, Ingris, Anabel, Miguel, Alexander, Israel y el otro Jesús.

A mi novia, por ser la super especial persona que comparte sus días conmigo.

A mi otra familia, por hacerme sentir muy bien a su lado: Olguita, Olguitín, Héctor y Rafa.

A mis compañeros de la universidad: la gente del 94105, del 88104 y del 96103. Son un montón, jeje.

A Tony, Daisy, Roberto y Alexander, por ayudarnos con todo el proceso de este trabajo.

A mis profesores, que hicieron todo su esfuerzo en hacer de mí un nuevo profesional.

A esta universidad, por abrirme sus puertas.

Alejandro Jiménez López

A mi hermana, por estar siempre conmigo, ser mi mayor apoyo y ejemplo. Gracias por tanto amor.

A mis padres, por estar cada segundo pendientes de mí y darme tanto amor.

A Alejandro mi compañero de tesis, amigo y hermano.

A mis amigos y hermanos: José Alberto, Maité, Milian, Senia, Campo, Jesús y Pancho. Gracias por estar siempre ahí, a pesar de todo.

A Nely, por todo su apoyo.

A la gente del grupo, saben que son muy importantes para mí.

A la gente del apto, tanto tiempo viviendo juntos, que ya son como mi familia.

A mis amigos en Camagüey, que a pesar de estar tan lejos he tenido su apoyo y cariño.

A Tony nuestro tutor y a la co-tutora Diana, gracias por todo el apoyo y la confianza depositada.

A Espí, Marelis, Alexander, Julio y demás integrantes de la Línea.

A las personas que han formado parte de los proyectos donde he trabajo, por ayudar en mi formación.

A todos los profesores que han participado en mi preparación.

A la universidad y a la Revolución.

Karel Delgado Alón

Resumen

Un sector en el cual la informática ha tenido una amplia repercusión es el industrial. En la actualidad existen aplicaciones de gran envergadura que se encargan de monitorear y controlar procesos altamente complejos. Estas aplicaciones son conocidas como sistemas SCADA. En nuestro país existen algunas soluciones de este tipo, las cuales no abarcan todo el espectro de escenarios donde pueden ser aplicadas, debido a que los despliegues de estos sistemas resultan costosos por los recursos de hardware necesarios para su implantación. Existe un gran número de escenarios en los que se realiza una automatización básica, en la que intervienen un pequeño número de variables y las actividades de control son poco complejas. En consecuencia, la implantación de las soluciones de factura nacional, implicaría un gasto de recursos innecesario, debido a la subutilización de los mismos. Con el objetivo de brindar una solución eficaz y económica, capaz de cubrir las necesidades de automatización en otros sectores y entidades, en la línea Sistemas Empotrados, perteneciente al Centro de Informática Industrial, se comenzó a desarrollar un sistema de supervisión y control de procesos haciendo uso de la tarjeta CID-300/9 basada en un microcontrolador ARM.

Los módulos HMI o de Interfaz de usuario, se encuentran entre los más importantes que componen un sistema SCADA. Aunque en el Centro existen otros módulos de este tipo, ninguno puede ser empotrado en la tarjeta antes mencionada. Por lo que surge la necesidad de desarrollar un nuevo módulo HMI con funcionalidades específicas que posibilite cumplir con esta meta. La solución propuesta abarca la creación de dos submódulos, los cuales en su conjunto conforman el módulo HMI Light. Estos submódulos se denominan HMI-Editor y HMI-Runtime. El primero permite exportar la configuración de un proyecto a un archivo XML, a partir del cual, el segundo submódulo visualiza los despliegues configurados, permitiendo ejercer acciones de control sobre el proceso supervisado y gestionar alarmas. Como resultado de este trabajo se obtuvo un módulo HMI que puede ser empotrado en la tarjeta CID-300/9.

Palabras claves: SCADA, HMI, microcontrolador.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

A pesar de que la informática es una ciencia joven, en los últimos años ha alcanzado considerables avances que han contribuido en gran medida al desarrollo económico de muchos países. Un sector en el cual la informática ha tenido una amplia repercusión es el industrial, llegando a existir en la actualidad aplicaciones de gran envergadura que se encargan de monitorear y controlar procesos altamente complejos. Estas aplicaciones son conocidas como sistemas SCADA (Supervisory Control And Data Acquisition o Control con Supervisión y Adquisición de Datos) [1].

El Estado cubano está destinando una gran cantidad de recursos en temas de automatización de procesos. En ese sentido se realizan licitaciones de soluciones integrales en el extranjero para suplir las necesidades tecnológicas, incurriendo en considerables gastos. El SCADA desarrollado en la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Informática Industrial (CEDIN) y otros sistemas de este tipo de producción nacional, como por ejemplo el EROS, desarrollado por el grupo SERCONI de la provincia Holguín, están llamados a sustituir las importaciones de este tipo de tecnología. El despliegue de estos sistemas resulta costoso por los recursos de hardware necesarios para su implantación.

Existe un gran número de escenarios en los que se realiza una automatización básica, en la que intervienen un pequeño número de variables y las actividades de control son poco complejas. En consecuencia, la implantación de los sistemas mencionados anteriormente, implicaría un gasto innecesario de recursos, debido a la subutilización de los mismos. Como ejemplo de estos escenarios se pueden mencionar algunos sectores del turismo, de la industria azucarera, así como un sinnúmero de pequeñas y medianas entidades que requieran algún tipo de automatización en procesos poco complejos.

Con el objetivo de brindar una solución eficaz y económica, capaz de cubrir las necesidades de automatización en los sectores y entidades mencionadas, en la línea Sistemas Empotrados se comenzó a desarrollar un sistema de supervisión y control de procesos haciendo uso de la tarjeta denominada CID-300/9, la cual fue desarrollada por el Instituto Central de Investigación Digital (ICID). La misma puede ser utilizada como núcleo de procesamiento del sistema mencionado

anteriormente y además, permite realizar visualizaciones sobre la pantalla que trae integrada. Esta tarjeta constituye una solución viable, debido a que es económica, de producción nacional y por ende, cuenta con amplio soporte.

Entre los módulos más importantes que componen un sistema SCADA se encuentra el HMI (Human-Machine Interface o Interfaz de usuario), el cual se encarga de permitir la visualización e interacción de los operadores con los procesos automatizados [2]. Es necesario mencionar que existen en el Centro otros módulos HMI desarrollados para ejecutarse sobre hardware con amplias prestaciones, siendo imposible que puedan ser utilizados en la tarjeta antes mencionada. Además, en la bibliografía consultada no se encontró ningún módulo HMI que cumpliera con los principios del Software Libre, ni con las especificaciones de la CID-300/9.

Ante la problemática enunciada se arriba al siguiente **problema científico**: ¿Cómo visualizar y permitir la interacción de los operadores en la automatización de procesos utilizando la tarjeta CID-300/9 basada en un micontrolador ARM?

Se define como **objeto de estudio**: HMI para sistemas SCADA.

Mientras que el **campo de acción** se enmarca en: HMI sobre dispositivos basados en microcontroladores.

Para dar respuesta al problema de la investigación se define como **objetivo**: Desarrollar un módulo HMI para la tarjeta CID-300/9 basada en un micontrolador ARM.

Se puede enunciar la **idea a defender** como: Si se desarrolla un módulo HMI que solo incluya las funcionalidades básicas de este tipo de sistemas, entonces podrá ser empotrado en la tarjeta CID-300/9 basada en un micontrolador ARM.

Para dar cumplimiento al objetivo se derivan las siguientes **tareas de investigación**:

1. Estudio del arte sobre los sistemas SCADA para determinar las principales características de algunos de los módulos HMI más reconocidos.
2. Selección de las tecnologías y herramientas adecuadas para el desarrollo.
    a) Sistema operativo.
    b) Lenguaje de programación.
    c) Framework de desarrollo.
    d) Entorno integrado de desarrollo.
    e) Lenguaje de modelado.
    f) Herramienta de modelado.

g) Metodología de desarrollo de software.

3. Definición de la propuesta de solución.
4. Implementación de un prototipo no funcional del HMI-Editor y del HMI-Runtime.
5. Diseño e implementación de los submódulos HMI-Editor y HMI-Runtime.
6. Diseño de pruebas de software.
   a) Pruebas de aceptación.
   b) Pruebas de rendimiento.
7. Evaluación de los resultados de las pruebas de software.

Los métodos científicos de investigación presentes en este trabajo son los siguientes:

- **Método de la observación**. Con el objetivo de observar cómo se comportan las salidas y el rendimiento del sistema.
- **Método de análisis histórico y lógico**. Con el objetivo de hacer un recorrido por la evolución de los sistemas SCADA, haciendo especial hincapié en los módulos HMI presentes en los mismos.
- **Método de análisis y síntesis**. Para realizar el análisis de la literatura especializada consultada y lograr obtener de forma sintetizada la información necesaria para la realización del trabajo.
- **Método de la modelación**. Con el objetivo de representar conceptos de la vida real a través de la abstracción, para modelar los elementos del dominio que forman parte de la solución.

Como **posible resultado** de la tesis se obtendrá un sistema compuesto por dos submódulos: un submódulo HMI en tiempo de edición, que podrá ser ejecutado sobre un ordenador, y un submódulo HMI para tiempo de ejecución, que será empotrado en la tarjeta CID-300/9, mediante el cual los operadores podrán visualizar y ejecutar acciones de control sobre los procesos supervisados.

La **estructura del trabajo** es la siguiente:

- **Capítulo 1:** Fundamentación teórica. En este capítulo se realiza un recorrido a través de algunos de los sistemas SCADA con mayor reconocimiento a nivel mundial y nacional, así como la determinación de las tecnologías y herramientas a utilizar.
- **Capítulo 2:** Diseño e implementación. En este capítulo se brinda una panorámica de la propuesta de solución, así como la modelación e implementación de la misma.

- **Capítulo 3:** Diseño y realización de pruebas al sistema. En este capítulo quedan registrados los casos de prueba realizados al sistema y los resultados de los mismos como validación de la solución.

2 Fundamentación Teórica

Entre los temas abordados en este capítulo, se encuentra un breve acercamiento a los sistemas SCADA, describiéndose sus principales componentes, haciendo especial énfasis en los módulos HMI. Otro de los temas, es la mención a diferentes sistemas de este tipo, donde se exponen algunos de ellos que constituyen referentes a nivel mundial. Por último, se recogen las tecnologías y las herramientas a utilizar.

2.1. Sistemas SCADA

El objetivo principal de la automatización es disminuir la intervención humana en la ejecución de los procesos de producción. Con vista a cumplir esa exigente meta se han desarrollado en los últimos años algunos sistemas de supervisión, control y adquisición de datos, denominados sistemas SCADA. Se asocia el nombre SCADA a cualquier software que ofrezca funcionalidades para el acceso a datos remotos de un proceso y permita, utilizando las herramientas de comunicación necesarias en cada caso, el control del mismo [1].

La presencia de estos sistemas aumenta los niveles de eficiencia, minimiza los costos y optimiza los procesos de producción. El término SCADA usualmente se refiere a un sistema central que monitorea y controla un sitio completo o un sistema que se extiende sobre una gran distancia (kilómetros/millas). La instalación de un sistema SCADA necesita de sensores, actuadores, controladores, redes, comunicaciones, base de datos, HMI, entre otros [2].

En la siguiente figura se muestra un esquema general de un sistema SCADA.

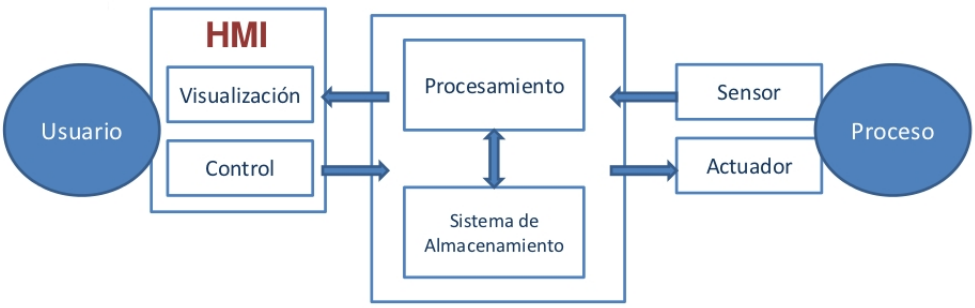

Figura 2.1: Estructura general de un sistema SCADA

2.1.1. Características de los sistemas SCADA

Entre las prestaciones de una herramienta de este tipo destacan:

- La monitorización. Representación de datos en tiempo real a los operadores.
- La supervisión. Permiten la supervisión, mando y adquisición de datos de un proceso y poseen herramientas de gestión para la toma de decisiones (mantenimiento predictivo, por ejemplo). Cuentan además con la capacidad de ejecutar programas que puedan supervisar y modificar el control establecido y, bajo ciertas condiciones, anular o modificar tareas asociadas a los dispositivos. Evita una continua supervisión humana.
- La adquisición de datos de los procesos en observación. Permite la adquisición de datos a través de los dispositivos, así como el procesamiento y almacenamiento de los mismos.
- El control. Posibilita que los operadores puedan cambiar parámetros u otros datos claves del proceso directamente desde el ordenador, escribiendo datos sobre los elementos de control [1].

Algunos módulos o bloques de software que componen un sistema SCADA:

- Manejadores. Aseguran mediante una interfaz genérica la comunicación del sistema de supervisión y control con los distintos dispositivos que existen en el campo, ya sean autómatas, PLC[1](Programmable Logic Controller o Controlador Lógico Programable), sensores inteligentes, entre otros [3].
- Configuración. Permite al usuario definir el entorno de trabajo de su SCADA, adaptándolo a la aplicación particular que se desea desarrollar.

[1]Son dispositivos electrónicos muy usados en automatización industrial, diseñados para programar y controlar procesos secuenciales en tiempo real.

- Interfaz gráfica. Proporciona al operador las funciones de control y supervisión de los procesos supervisados, los cuales se representan mediante sinópticos gráficos[2].
- Módulo de proceso. Ejecuta las acciones de mando preprogramadas a partir de los valores actuales de variables leídas. La programación se realiza por medio de bloques de programa en lenguaje de alto nivel .
- Gestión y archivo de datos. Se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.
- Comunicaciones. Se encarga de la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y entre esta y el resto de los elementos informáticos de gestión [4].

Los sistemas SCADA intervienen en una gran variedad de procesos, no solo del tipo industrial, logrando así convertirse en excelentes medios para lograr un mejor aprovechamiento de los recursos, así como la humanización del trabajo.

2.1.1.1. HMI

Existen dispositivos tales como los controladores lógicos programables que realizan automáticamente control sobre un proceso. Tradicionalmente, estos no poseen una forma de mostrar la información al operador, haciendo engorroso el trabajo de los mismos, por lo que los sistemas SCADA incluyen un componente que proporciona una vía de solución a este problema. Una Interfaz de usuario o HMI (Human Machine-Interface) en este ámbito, es un software que permite la representación gráfica de un proceso determinado, así como la interacción con el mismo (Figura 2.2), entiéndase visualización de las variables que describen su evolución, así como la ejecución de acciones sobre él. Además, proporcionan una amplia gestión de las alarmas del sistema [5].

[2]Disposición gráfica que muestra o representa cosas relacionadas entre sí, facilitando su visión conjunta.

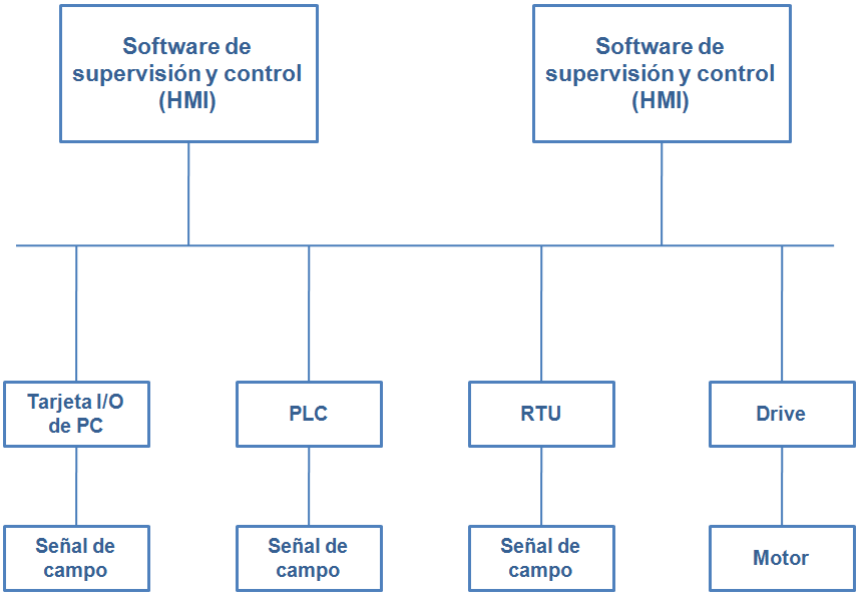

Figura 2.2: Posición del módulo HMI dentro de los sistemas SCADA

Algunas de las funcionalidades con las que un HMI debe contar son:

- Permitir una comunicación con dispositivos de campo.
- Actualizar una base de datos con las variables del proceso.
- Visualizar las variables mediante pantallas con objetos animados.
- Permitir que el operador pueda enviar señales al proceso, mediante botones, controles on/off o ajustes continuos con el mouse o el teclado.
- Supervisar niveles de alarma y alertar/actuar en caso de que las variables excedan los límites normales.
- Almacenar los valores de las variables para un análisis estadístico y/o control.
- Controlar en forma limitada ciertas variables de proceso [5].

Como se puede apreciar los sistemas SCADA se componen de diferentes módulos de software, y aunque existe diversidad en cuáles incluir, el módulo HMI es absolutamente necesario para lograr un verdadero aprovechamiento de una aplicación de esta naturaleza. Cuando se hace uso de una interfaz gráfica de usuario, las prestaciones del sistema aumentan considerablemente. Esto es fácilmente apreciable pues los usuarios poseen un número mayor de posibilidades, como por ejemplo: la visualización del comportamiento de las variables que representan los procesos, el uso de gráficas de tendencia, una mayor facilidad en

la realización del control, la generación de reportes también pasa un plano más importante, pues con el uso de los mismos, los datos son presentados al operador de una forma más organizada. Lo antes descrito, refleja que los HMI permiten una interacción más dinámica de los operadores con los procesos supervisados.

2.1.2. Microcontroladores

Los microcontroladores son circuitos integrados programables, capaces de ejecutar las órdenes grabadas en su memoria. Están compuestos de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres unidades funcionales principales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida [6]. Dentro de las ventajas del uso de microcontroladores, se encuentra el hecho de que están diseñados tanto para sectores industriales como para consumidores no industriales, reducen la necesidad de conexiones externas debido a la incorporación de la memoria y los periféricos, poseen una alta eficiencia en cuanto al consumo de energía y por último, se puede decir que son soluciones efectivas y de bajo costo, aplicables a una gran gama de sectores [7].

Para el desarrollo de este trabajo, se hace uso de la tarjeta basada en microcontroladores CID-300/9, desarrollada por el ICID en Cuba. La misma cuenta con las siguientes prestaciones de hardware: un microcontrolador ARM-920T con frecuencia de 400 MHz, una memoria RAM de 128 MB, cuatro puertos USB, uno Ethernet, dos puertos Serie, dos puertos de entrada/salida de audio de 3.5 mm y un puerto para una tarjeta SD. Además, cuenta con una pantalla LCD táctil [8]. Esta tarjeta constituye una solución viable, debido a que es económica, de producción nacional y por ende, cuenta con amplio soporte.

2.1.3. Ejemplos de sistemas SCADA

A continuación se hará referencia a diferentes aplicaciones que son representativas de sistemas SCADA a nivel nacional e internacional. Se realizará una panorámica de cada una, haciendo énfasis en el componente HMI.

2.1.3.1. Fast/Tools

Fast/Tools es un poderoso, flexible y reciente sistema distribuido de supervisión y control creado por la Corporación Eléctrica Yokogawa. Es líder en las operaciones de control e integración de la información disponible en un amplio espectro de

plataformas industriales. Ha sido desarrollado durante veinte años combinando la experticia en sistemas SCADA con los requisitos de usuarios e industrias líderes.

El sistema está conformado por módulos, lo que significa que es configurable para que posea las funcionalidades que se necesiten y puedan ser agregados fácilmente otros módulos si se desea expandir la aplicación.

Uno de esos módulos es el de Configuración y Presentación -conocido también como HMI- el cual posee varias características, entre las que se encuentran:

- Funcionalidad multipáginas de alarmas con opciones para su ordenación basadas en la prioridad o el tiempo y redireccionamiento de las mismas.
- Gráficos orientados a objetos. No necesitan actualizar símbolos de manera independiente, un cambio en un símbolo actualiza todas sus copias.
- Gráficos de tendencia predefinidos.
- Edición (agregar/eliminar) en línea de componentes, reportes y drivers de entrada/salida, sin afectar las operaciones llevadas a cabo por el sistema.
- Habilitación de contenido web. Lo que significa que un navegador web se puede utilizar como interfaz del sistema.
- Posee un potente editor de configuración en el cual pueden ser definidos todos los elementos que componen el sistema, como son: los drivers de entrada/salida, los reportes, los elementos gráficos, entre otros [9].

2.1.3.2. RSView32

RSView32 es un HMI integrado perteneciente a la compañía Rockwell Automation, que tiene como objetivo monitorear y controlar procesos y máquinas de automatización. Está diseñado para ejecutarse sobre entornos de Microsoft Windows.

El mismo cuenta con un poderoso editor gráfico que permite crear la configuración deseada para el sistema a supervisar. Entre sus principales funcionalidades cuenta con:

- La inclusión de contenedores OLE[3] (Object Linking and Embedding o Incrustación y Enlazado de Objetos) para ActiveX, lo que posibilita la inclusión de componentes de terceros.
- El diseño de gráficos de alto nivel, incluso para la aplicación más compleja haciendo uso del entorno de dibujo.

[3]Es el nombre de un sistema de objetos distribuidos que permite a un editor encargar a otro la elaboración de cierto contenido para importarlo posteriormente.

- Permite importar gráficos de otras aplicaciones de diseño como son: AutoCAD, CorelDRAW y Photoshop.
- Permite la importación y exportación de despliegues completos en formato XML[4] (eXtensible Markup Language o Lenguaje Extensible de Etiquetas).
- Ofrece una completa configuración de las propiedades de los componentes gráficos, incluyendo color de fondo, color en primer plano, escalado, posición, tamaño, entre otras.

Por su parte la versión en ejecución del HMI, brinda algunas ventajas como son:

- Ahorrar tiempo permitiendo cambios en línea que no requieren que se detengan los procesos, actualizando los cambios la próxima vez que se abra ese despliegue.
- Ofrecer un completo y flexible sistema de alarmas. Mostrándolas en la pantalla, en los registros de sucesos, incluso permite exportarlas en formato .DBF, para que puedan ser cargadas desde cualquier gestor de base de datos que soporte este formato [10].

2.1.3.3. WinCC

WinCC es desarrollado por Siemens y diseñado para ejecutarse sobre sistemas operativos Microsoft Windows 2003, XP o Vista [11]. El mismo está diseñado sobre bases no específicas de tecnología o industria, es modular y se puede extender de una forma flexible. Puede ser utilizado como una aplicación para un solo usuario en la ingeniería mecánica, así como en soluciones multiusuario complejas, o incluso en sistemas distribuidos que incluyen varios servidores y clientes. Este sistema se ha establecido como un estándar en la industria, en el campo de la monitorización de procesos [12].

El componente HMI del WinCC está caracterizado por una amplia variedad de controles y funciones configurables. Además de permitir la generación de reportes y mantener un registro de sucesos del sistema, brinda la posibilidad de habilitación de contenido web, lo cual permite que pueda ser operado desde un navegador. Ofrece un editor de los más reconocidos cuando se habla de eficiencia o de diseño amigable. El uso de bibliotecas y asistentes hacen que la generación de un proyecto sea fácil, rápida y menos propenso a errores. Como un sistema pensado para ejecutar las tareas más complicadas desde el HMI, es capaz de manejar exhaustivos proyectos y un gran volumen de datos [13]. Permite configuración en línea sin tener que detener los procesos [11].

[4]Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

2.1.3.4. GALBA

El SCADA "Guardián del ALBA" (GALBA), es un sistema distribuido para la supervisión de los procesos de producción de petróleo en la República Bolivariana de Venezuela. Este sistema está compuesto por varios módulos que realizan su comunicación a través de una infraestructura de comunicación llamada middleware; mientras la comunicación con los dispositivos de campo se realiza a través del servidor de base de datos en tiempo real [14].

El editor HMI del GALBA se ajusta a los estándares de usabilidad y diseño de interfaces gráficas de usuario presentes en otras áreas de la industria del software. El ambiente de configuración del editor HMI permite a un mantenedor definir el ambiente de trabajo del SCADA, adaptándolo mejor a la aplicación particular que se desee desarrollar. En el ambiente de configuración actual se pueden crear varios nodos y cada nodo puede contar con varios módulos, como módulos HMI donde se crean los despliegues sobre los cuales se configuran los widgets[5] necesarios para representar la información al operador. Se lleva a cabo también la asociación de estos widgets con las variables que representan y se permite modificar las propiedades de los elementos gráficos y los despliegues. Se configuran los módulos de históricos. Maneja conceptos de base de datos en tiempo real, que incluyen canales, subcanales, dispositivos, y los distintos tipos de puntos y alarmas con las que cuenta el sistema, que representan toda la información de configuración necesaria para este módulo [15].

Como se puede apreciar en lo descrito anteriormente, los módulos HMI poseen funcionalidades comunes tales como: manejo de alarmas, habilitación de contenido web, edición en línea, potentes editores, entre otras. Las mismas deben tenerse en cuenta a la hora de desarrollar una aplicación de este tipo. Por otra parte, existen otros elementos importantes que estos sistemas no contemplan, como por ejemplo su aplicación en hardware con pocas prestaciones, o el hecho de que en la gran mayoría de los casos, están implementados para ejecutarse solamente sobre un sistema operativo privativo, y a su vez ellos mismos también lo son. Resumiendo, se puede decir que es importante además de analizar y estudiar sus características, valorar la eliminación de dichas limitaciones.

[5] Elemento de una interfaz gráfica de usuario utilizado para mostrar información. Combinados en una aplicación permiten contener los datos procesados por la misma y las interacciones disponibles con estos datos.

2.2. Tecnologías y herramientas

A continuación se describen las tecnologías y herramientas a utilizar. Esta elección responde a múltiples razones, entre las que se encuentra el hecho de que las mismas son exponentes del movimiento de Software Libre, lo que contribuye al desarrollo del país y a la generalización de los principios y metas que propone este movimiento. Estas tecnologías y herramientas, además de poseer características que se ajustan a las necesidades de la presente investigación, son ampliamente utilizadas en la línea Sistemas Empotrados, perteneciente al CEDIN.

2.2.1. Sistema operativo

Ubuntu es un sistema operativo predominantemente enfocado en la facilidad de uso e instalación, la libertad de los usuarios, y los lanzamientos regulares (cada 6 meses). El nombre proviene del concepto africano ubuntu, que significa "humanidad hacia otros" o "yo soy porque nosotros somos". También es el nombre de un movimiento humanista sudafricano. Ubuntu aspira a impregnar de esa mentalidad al mundo de las computadoras. El eslogan de Ubuntu “Linux para seres humanos” resume una de sus metas principales: hacer de Linux un sistema operativo más accesible y fácil de usar. El proyecto Ubuntu está totalmente basado en los principios del Software libre y anima a que la gente use, mejore y distribuya software libre. Es de código abierto. Cada versión de Ubuntu recibe soporte al menos durante 18 meses con actualizaciones genéricas y de seguridad. Cada 2 años se publica una versión especial (LTS, Long Time Support) con soporte extendido, 3 años para sistemas de escritorio y 5 años para servidores [16].

Por otra parte, se cuenta con un sistema operativo empotrado en la tarjeta CID-300/9. El mismo fue construido en la línea Sistemas Empotrados, en colaboración con el ICID, usando la versión 2.6.29 del núcleo de Linux. En su concepción, se tuvo en cuenta que el sistema quedara de la forma más optimizada posible, siendo ligero en RAM a la hora del arranque para que se pudieran incorporar un mayor número de aplicaciones [8].

2.2.2. Lenguaje de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina [17].

C++ es un lenguaje de propósito general, de la familia de los lenguajes imperativos y orientado a objetos, basado en el lenguaje de programación C, al que se le

han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones inline y sobrecarga de operadores [18].

Es un lenguaje potente para el desarrollo de aplicaciones críticas o complejas, por la facilidad que brinda la manipulación de la memoria del ordenador a través de los punteros. Como lenguaje orientado a objetos brinda ventajas propias de esta filosofía de programación, como son: la abstracción, el encapsulamiento, la herencia y el polimorfismo. Lo que nos permite entre otras funcionalidades, poder representar un conjunto de características y comportamientos similares de objetos de la vida real y agruparlos en una unidad básica, brindando una interfaz para permitir la interacción con la misma.

Las primeras aplicaciones tendieron a tener un fuerte componente de programación de sistemas. Por ejemplo, varios de los sistemas operativos más importantes se han escrito en C++ y muchos más poseen módulos claves escritos en este lenguaje.

A pesar de que es un lenguaje orientado a objetos, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para la programación a bajo nivel. Esto nos permite utilizarlo para escribir controladores de dispositivos y otros software que se basen en la manipulación directa de hardware, bajo restricciones de tiempo real. C++ fue diseñado para que cada característica del lenguaje se pudiera utilizar en virtud de limitaciones severas de tiempo y espacio [19].

2.2.3. Framework de desarrollo

Qt es un framework multiplataforma ampliamente usado para desarrollar aplicaciones que necesiten una interfaz gráfica de usuario amigable, así como programas de otra naturaleza. Se encuentra licenciado bajo los términos de la GNU Lesser General Public License (LGPL) versión 2.1. La API de Qt está implementada en C++, y ofrece funciones adicionales para facilitar el desarrollo multiplataforma. Existen adaptaciones o binding (en inglés) para otros lenguajes de programación como Java, C#, Python, Ada, Pascal, Perl, PHP, Ruby, entre otros.

Qt va más allá de C++ en áreas como la comunicación entre objetos y la flexibilidad para el desarrollo avanzado de interfaces gráficas de usuario, añadiéndole las siguientes características a C++:

- Poderoso mecanismo para la comunicación entre objetos conocido como: signals y slots.
- Facilidades para consultar y establecer las propiedades de un objeto.
- Poderoso filtro de eventos.

- Cadena de traducción contextual para la internacionalización.
- Punteros guardados que se igualan automáticamente a cero cuando el objeto se destruye, a diferencia de los punteros normal en C++ que se convierten en punteros colgantes [20].

Entre los principales módulos que conforman Qt podemos mencionar los siguientes:

- QtCore. Constituye el fundamento para todas las aplicaciones basadas en Qt, elementos no visuales que utilizan otros módulos.
- QtGui. Componentes visuales para las interfaces gráficas de usuario.
- QtNetwork. Clases para la programación de aplicaciones de red.
- QtOpenGL. Clases para el soporte de OpenGl.
- QtSQL. Clases de integración con bases de datos usando SQL.
- QtSvg. Clases para desplegar archivos SVG.
- QtXml. Clases para la manipulación de documentos XML.
- QtMultimedia. Clases para funcionalidades de bajo nivel para multimedias.
- QtDBus. Clases para la comunicación entre procesos empleando D-Bus.
- QtPropertyBrowser. Clases que proporcionan un inspector de propiedades para objetos.

2.2.4. Entorno integrado de desarrollo

Un Entorno Integrado de Desarrollo (IDE, Integrated Development Environment) es una aplicación que consiste generalmente en la combinación de un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Puede ser exclusivo para un solo lenguaje de programación o ser utilizado para varios.

Qt Creator es un entorno multiplataforma hecho a la medida para los desarrolladores de Qt, desarrollado en sus inicios por Trolltech el cual fue posteriormente comprado por Nokia. Se puede ejecutar sobre los sistemas operativos Windows, Linux/X11 y Mac OS X, permitiendo crear aplicaciones para diferentes plataformas de escritorio y de dispositivos móviles. Entre las facilidades que proporciona, podemos encontrar: un editor de código para C++ y QML (Javascript), un editor integrado de interfaz de usuario, herramientas para la administración de proyectos y para la construcción de los mismos, permitiendo cambiar fácilmente entre destinos de construcción, con posibilidad de depuración y soporte para control de versiones integrándose con los sistemas más populares incluyendo: Git, Subversion, Bazaar, CVS y Mercurial [21].

Una de las mayores ventajas del Qt Creator es que permite a los equipos de desarrolladores compartir un proyecto a través de diferentes plataformas de desarrollo (Microsoft Windows, Mac OS X y Linux) mediante una herramienta común para el desarrollo y la depuración. Posee integración con la herramienta (Valgrind) para el análisis del código en busca de problemas de gestión de memoria [22]. Además, permite la integración con dispositivos empotrados que posean un sistema operativo Linux, lo que permite el despliegue de las aplicaciones de una manera rápida y sencilla, el monitoreo de procesos y la depuración de forma remota.

El objetivo principal de Qt Creator es satisfacer las necesidades de los desarrolladores de Qt que buscan simplicidad, facilidad de uso, productividad y extensibilidad [22].

2.2.5. Lenguaje de modelado

Las grandes empresas de software deben ser más que simplemente un conjunto de módulos de código. Estas deben estar estructuradas de manera tal que permita escalabilidad, seguridad y una ejecución sólida bajo condiciones de estrés; y sus arquitecturas deben ser definidas con suficiente claridad para que los programadores puedan encontrar y reparar rápidamente errores que se puedan presentar mucho después de que los autores originales se hayan trasladado hacia otros proyectos. Otro de los beneficios de poseer algún tipo de estructura es que posibilita la reutilización de código. El tiempo de diseño es la forma más fácil de estructurar una aplicación como un conjunto de módulos o componentes independientes.

El proceso de modelado es el diseño de aplicaciones de software antes de llevar a cabo la codificación. Un modelo en el desarrollo de software juega el rol análogo al de los planos u otros esquemas en la construcción de un rascacielos. Haciendo uso de los modelos, aquellos responsables por el éxito del proyecto pueden asegurarse que las funcionalidades del negocio están completas y son correctas, que las necesidades del usuario final están cubiertas y que el diseño de la aplicación soporta requisitos de escalabilidad, robustez, seguridad, extensibilidad y otras características, antes de que la implementación del código haga que los cambios sean difíciles y caros de realizar.

El Lenguaje Unificado de Modelado (UML, Unified Modeling Language) ayuda a especificar, visualizar y documentar modelos de sistemas de software, incluyendo su estructura y diseño, de forma tal que cumpla con todos los requisitos descritos anteriormente. Permite modelar casi cualquier tipo de aplicación, que se ejecute en cualquier tipo de combinación de hardware, sistema operativo, lenguaje de programación o infraestructura de red. Está diseñado para ser usado sobre los

fundamentos de la orientación a objetos, convirtiéndolo en un marco ideal para los lenguajes orientados a objetos como C++, Java y el reciente C#, teniendo uso limitado para otros paradigmas de programación. El proceso de recopilación y análisis de los requisitos de una aplicación para incorporarlos a un diseño, es una tarea compleja y la industria actualmente es compatible con muchas metodologías que definen procedimientos formales que especifican como llevar a cabo esto. Una de las características de UML (de hecho la que permite que sea ampliamente utilizado en la industria de software) es que es independiente de la metodología que se seleccione. Independientemente de la metodología que se utilice para llevar a cabo el análisis y el diseño, se puede utilizar UML para expresar los resultados.

UML 2.0 define 13 tipos de diagramas, divididos en tres categorías: seis de ellos representan estructuras estáticas de la aplicación, otros tres representan tipos generales de comportamiento y los cuatro restantes representan diferentes aspectos de interacción [23].

2.2.6. Herramienta de modelado

Hoy en día, muchas empresas se han extendido a la adquisición de herramientas CASE (Computer Aided Software Engineering, Ingeniería Asistida por Computadora), con el fin de automatizar los aspectos claves de todo el proceso de desarrollo de un sistema y por ende, destinadas a aumentar la productividad de este proceso, reduciendo costes y tiempo [24].

Visual Paradigm es una herramienta para la creación de diagramas UML que es de gran ayuda en el proceso de desarrollo de software. Especialmente en las fases de análisis y diseño de este proceso, esta aplicación ayuda a conseguir un producto de alta calidad.

Este sistema puede manejar gran parte de los diagramas definidos en el lenguaje UML, ya sea creándolos manualmente o importándolos a partir de código en C++, Java, Python, Pascal, Delphi, entre otros, lo cual ofrece posibilidades de ingeniería inversa. Permite crear un modelo y generar el código automáticamente en varios lenguajes para ayudar a comenzar la implementación del proyecto. El código generado incluye definiciones de clases con sus métodos y atributos proporcionando rapidez en el proceso inicial de desarrollo y haciéndolo menos propenso a errores.

2.2.7. Metodología de desarrollo de software

Las metodologías de desarrollo de software son un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de

información. Hoy en día existen numerosas propuestas metodológicas, un ejemplo de ellas son las propuestas tradicionales centradas específicamente en el control del proceso, estas han demostrado ser efectivas en proyectos de gran tamaño. Sin embargo, la experiencia ha demostrado que las metodologías tradicionales no ofrecen una buena solución a proyectos donde el entorno es volátil y donde los requisitos no se conocen con exactitud. Como respuesta a las dificultades presentadas por las metodologías tradicionales, surgen las metodologías ágiles, las cuales tratan de adaptarse a la realidad del software.

La metodología XP (eXtreme Programming, Programación Extrema) es una de las más recientes en el desarrollo de software. Se clasifica dentro de la categoría de las Metodologías Ágiles. La filosofía de XP es satisfacer por completo las necesidades del cliente, por eso se integra como uno más al equipo de desarrollo. Está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, dónde la comunicación sea más factible. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes.

Entre las características de XP podemos destacar:

- **Simplicidad:** Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se quieren ampliar resulta imposible hacerlo, se tienen que desechar y partir de cero.
- **Comunicación:** Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa, no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.
- **Retroalimentación:** Mediante la retroalimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a sus necesidades ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto.
- **Coraje:** Se debe tener coraje o valentía para cumplir los tres puntos anteriores. Hay que tener valor para comunicarse con el cliente y enfatizar algunos puntos, a pesar de que esto pueda dar sensación de ignorancia por parte del programador, hay que tener coraje para mantener un diseño simple y no optar por el camino más fácil y por último hay que tener valor y confiar en que la retroalimentación sea efectiva [25].

La metodología Programación Extrema se ajusta a las condiciones de desarrollo existentes, ya que está diseñada para adaptarse a un ambiente donde los cambios

ocurren de forma periódica, donde el cliente forma parte del equipo de trabajo, lo que posibilita una rápida retroalimentación, donde se hace necesario reducir el volumen de artefactos generados y además, propone el uso de la programación en parejas lo que proporciona una visión más amplia y mejor del problema a resolver, posibilitando generar soluciones con mayor calidad.

Conclusiones

De forma general los sistemas SCADA se componen de diferentes módulos o subsistemas, entre los que se destaca el módulo de interfaz de usuario o HMI. El mismo, permite la representación visual de un proceso determinado, así como la interacción con el mismo. Entre las principales funcionalidades que posee un módulo HMI, se seleccionan las siguientes: poseer un editor gráfico para la creación y configuración de despliegues, visualizar el valor de las variables mediante elementos gráficos, permitir la ejecución de acciones de control sobre el proceso supervisado y la gestión de alarmas. Otro aspecto de este capítulo es la descripción de las herramientas y tecnologías a utilizar. En el mismo, se destaca la selección de XP como metodología de desarrollo, la determinación de C++ como lenguaje de programación, de Qt como framework de desarrollo y por último, la selección del Lenguaje Unificado de Modelado junto con la herramienta Visual Paradigm para la modelación de la solución.

3 Diseño e implementación

En el ámbito de la ingeniería de software, un buen sistema es aquel que cumple con las necesidades del cliente y posee facilidades de soporte luego de ser liberado. Con el objetivo de lograr un software con estas características, en el presente capítulo se exponen varios elementos que forman parte del proceso de desarrollo, como son: la propuesta del sistema, el modelo arquitectónico propuesto, una visión general del diseño y la implementación de la solución.

3.1. Propuesta del sistema

La solución que se propone abarca la creación de dos submódulos, los cuales en su conjunto conforman el módulo HMI Light. El submódulo HMI en tiempo de edición (HMI-Editor), que se ejecuta sobre un ordenador, le permite al usuario crear y configurar varios despliegues y variables. Estas variables están vinculadas a los elementos gráficos que aparecen en los despliegues. Dichos elementos pueden ubicarse en cualquier posición determinada por el operador, además son reconfigurables en cuanto a tamaño, escalado, rotación y otros parámetros. El HMI-Editor exporta toda la configuración antes descrita a un archivo XML, a partir del cual, el HMI en tiempo de ejecución (HMI-Runtime) visualiza los despliegues configurados anteriormente. Además, el mismo posibilita la actualización de los componentes mediante la comunicación con un recolector, el cual brinda la nueva información de las variables disponibles en los dispositivos de campo. Otra de las funcionalidades de este submódulo, es la de permitir al usuario realizar acciones de control sobre el proceso supervisado. Es importante mencionar que el HMI-Runtime se ejecuta sobre la tarjeta CID-300/9 (Figura 3.1).

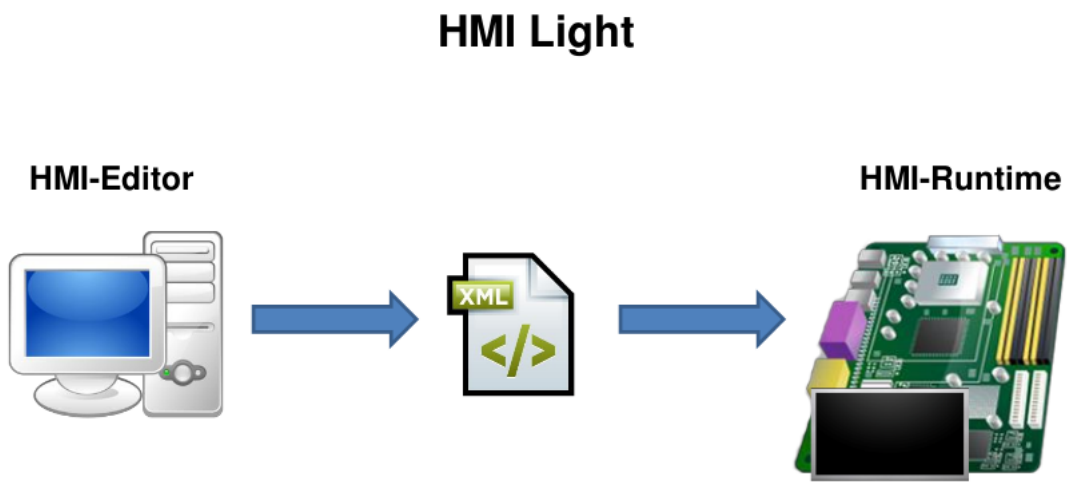

Figura 3.1: Propuesta del sistema

3.2. Modelo arquitectónico propuesto

En la construcción de software existen diferentes estilos arquitectónicos. Los mismos proporcionan una plantilla de trabajo, que unifica la manera en que todos los miembros del equipo ven el sistema y además, impone una transformación al diseño del mismo.

Un patrón arquitectónico, al igual que un estilo, impone una transformación en el diseño de una arquitectura. Sin embargo, un patrón difiere de un estilo en varios elementos fundamentales: 1) el alcance de una patrón es menor, ya que se concentra en un aspecto en lugar de hacerlo en toda la arquitectura; 2) un patrón impone una regla sobre la arquitectura, pues describe la manera en que el software manejará algún aspecto de su funcionalidad al nivel de la infraestructura (por ejemplo, concurrencia); 3) los patrones arquitectónicos tienden a abarcar aspectos específicos del comportamiento dentro del contexto de la arquitectura (por ejemplo, cómo maneja una aplicación en tiempo real la sincronización o las interrupciones). Los patrones se usan junto con un estilo arquitectónico para determinar la forma de la estructura general de un sistema [26].

El diseño arquitectónico representa la estructura de los componentes necesarios para construir un sistema computacional. Asume el estilo arquitectónico que tomará el sistema, la estructura y las propiedades de los componentes que constituyen el sistema y las interrelaciones entre ellos.

Con el objetivo de establecer una estructura para todos los componentes del

sistema se escoge el estilo arquitectónico Orientado a objeto. El mismo plantea que los componentes de un sistema encapsulan los datos y las operaciones que deben aplicarse para manipular estos datos. La comunicación y coordinación entre los componentes se consigue mediante el paso de mensajes [26].

Por otra parte, el principio de separación Modelo-Vista se selecciona como patrón arquitectónico, el cual es una de las variantes existentes del patrón Capas . En este contexto, el modelo es un sinónimo de la capa del dominio de los objetos, mientras que la vista es un sinónimo para los objetos de la presentación, como son las ventanas. Una parte adicional de este principio es que las clases del dominio encapsulan la información y el comportamiento relacionado con la lógica de la aplicación. Las clases de la vista son relativamente delgadas; son responsables de la entrada y salida, y de capturar los eventos de la interfaz gráfica de usuario, pero no mantienen datos ni proporcionan directamente ninguna funcionalidad de la aplicación. Es conveniente que no exista un acoplamiento directo de otros componentes con los objetos ventana, puesto que las ventanas están relacionadas con una aplicación específica, mientras que (idealmente) los componentes que no son ventanas podrían reutilizarse en nuevas aplicaciones o conectarse a una nueva interfaz [27].

La figura que se muestra a continuación muestra la estructura general del estilo y el patrón arquitectónico descritos anteriormente.

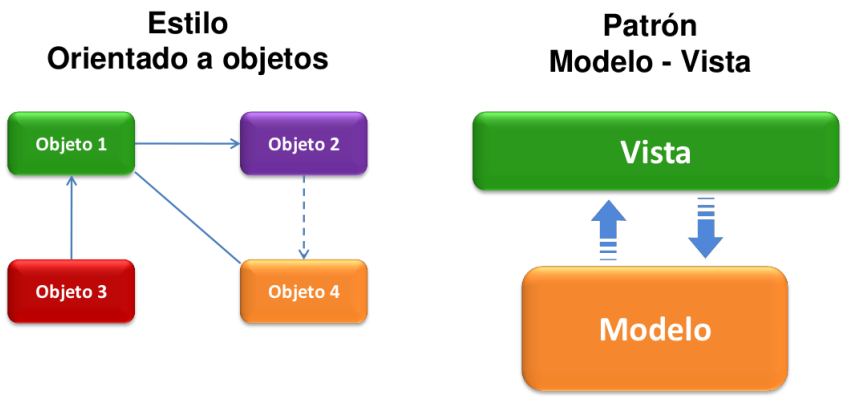

Figura 3.2: Estilo y patrón arquitectónico

3.3. Historias de usuario y tareas de ingeniería

Entre los artefactos que define la metodología seleccionada se encuentran las historias de usuario, las cuales representan una breve descripción del

comportamiento del sistema, emplean terminología del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación.

Se redactaron un conjunto de historias de usuario en las cuales se recogen las principales funcionalidades con las que debe contar el sistema (Anexo 1), a partir de las cuales se derivan un grupo de tareas de ingeniería que responden a los diferentes pasos que se deben ejecutar para satisfacer lo descrito en cada una de las historias de usuario anteriormente mencionadas (Anexo 2).

A continuación se muestra un resumen de las historias de usuario y sus tareas de ingeniería.

Historia de usuario	
Número: 1	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Gestión de los elementos del árbol semántico de proyecto.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2.5	**Iteración asignada:** 1
Programador responsable: Karel Delgado Alón	
Descripción: Permitir la adición, eliminación y edición de elementos como: proyecto, vista de despliegues, vista de variables, despliegues, variables y otros, todos ordenados de forma jerárquica.	
Observaciones:	

Cuadro 3.1: Historia de usuario No. 1

Tarea	
Número: 1	**Número historia:** 1
Nombre tarea: Diseño del inspector de proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 02/11/2011	**Fecha fin:** 04/11/2011
Programador responsable: Karel Delgado Alón	
Descripción: La interfaz de usuario cuenta con un panel en su parte izquierda superior donde se van a desplegar los proyectos que se encuentren abiertos y por cada uno de ellos los elementos que lo compongan.	

Cuadro 3.2: Tarea de ingeniería No. 1 / Historia de usuario No. 1

	Tarea
Número: 2	**Número historia:** 1
Nombre tarea: Implementación de la lógica de funcionamiento del inspector de proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 07/11/2011	**Fecha fin:** 18/11/2011
Programador responsable: Karel Delgado Alón	
Descripción: Si se presiona el botón derecho del mouse sobre el inspector se da lo opción de agregar un nuevo proyecto, si se hace sobre un proyecto se da la opción de agregar una vista de despliegues, una vista de variables o de eliminar el proyecto. Si el mouse es presionado sobre una vista de despliegues se da la opción de agregar despliegues o de eliminar la vista. Cuando ocurre sobre una vista de variables se muestra una ventana donde se introduce el nombre y la dirección de la variable. Estos elementos que componen el árbol de proyecto, pueden ser clickeados con el botón derecho del mouse lo que provoca que se muestren o no, según sea su naturaleza, en la zona de trabajo que se define entre el inspector del proyecto, el inspector de propiedades y la paleta de componentes.	

Cuadro 3.3: Tarea de ingeniería No. 2 / Historia de usuario No. 1

	Historia de usuario
Número: 2	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Inspección de propiedades de los elementos del árbol semántico.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 2.5	**Iteración asignada:** 1
Programador responsable: Karel Delgado Alón	
Descripción: Todos los componentes del árbol semántico, dígase: proyecto, vista de despliegues, vista de variables, despliegues, elementos gráficos, entre otros, deben mostrar sus propiedades en el inspector al ser seleccionados.	
Observaciones:	

Cuadro 3.4: Historia de usuario No. 2

	Tarea
Número: 1	**Número historia:** 2
Nombre tarea: Estudio y prueba de de la biblioteca QPropertyBrowser.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 21/11/2011	**Fecha fin:** 23/11/2011
Programador responsable: Karel Delgado Alón	
Descripción: Esta biblioteca proporciona una API para trabajar con las propiedades de elementos que instrumenten el sistema de propiedades propuesto por Qt.	

Cuadro 3.5: Tarea de ingeniería No. 1 / Historia de usuario No. 2

	Tarea
Número: 2	**Número historia:** 2
Nombre tarea: Diseño del inspector de propiedades.	
Tipo de tarea: Investigación	**Puntos estimados:** 0.5
Fecha inicio: 23/11/2011	**Fecha fin:** 25/11/2011
Programador responsable: Karel Delgado Alón	
Descripción: La interfaz de usuario cuenta con un panel en su parte izquierda inferior donde se van a desplegar las propiedades del elemento seleccionado en un momento dado.	

Cuadro 3.6: Tarea de ingeniería No. 2 / Historia de usuario No. 2

	Tarea
Número: 3	**Número historia:** 2
Nombre tarea: Implementación del inspector de propiedades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 28/11/2011	**Fecha fin:** 07/12/2011
Programador responsable: Karel Delgado Alón	
Descripción: El inspector de propiedades muestra las propiedades definidas para cada elemento que compone el árbol semántico, así como de los elementos gráficos que pueden aparecer en un despliegue, también brinda la posibilidad de modificar dichas propiedades.	

Cuadro 3.7: Tarea de ingeniería No. 3 / Historia de usuario No. 2

Historia de usuario	
Número: 5	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Vinculación de los elementos gráficos con variables.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Karel Delgado Alón	
Descripción: Los elementos gráficos en los despliegues deben representar el comportamiento de alguna variable recogida en algún proceso, por lo que debe existir un vínculo entre los mismos.	
Observaciones:	

Cuadro 3.8: Historia de usuario No. 5

Tarea	
Número: 1	**Número historia:** 5
Nombre tarea: Diseño de la vinculación de los elementos gráficos con las variables.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 20/02/2012	**Fecha fin:** 22/02/2012
Programador responsable: Karel Delgado Alón	
Descripción: Los elementos gráficos que se encuentran ubicados en alguna posición dentro de un despliegue, al ser seleccionados con el botón derecho del mouse muestran un menú contextual donde se brinda la opción de vincular el mismo con una variable de las que pertenezcan al proyecto. Para este propósito se muestra una ventana emergente con las variables disponibles.	

Cuadro 3.9: Tarea de ingeniería No. 1 / Historia de usuario No. 5

	Tarea
Número: 2	**Número historia:** 5
Nombre tarea: Vinculación de los elementos gráficos con las variables.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 22/02/2012	**Fecha fin:** 02/03/2012
Programador responsable: Karel Delgado Alón	
Descripción: Cuando se muestra la ventana emergente con las variables disponibles, el usuario tiene la posibilidad de escoger una de ellas, en caso de escogerla se procede a vincular esta variable mediante su identificador con el elemento seleccionado.	

Cuadro 3.10: Tarea de ingeniería No. 2 / Historia de usuario No. 5

	Historia de usuario
Número: 6	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Edición de la apariencia y ubicación de los elementos gráficos.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Karel Delgado Alón	
Descripción: Debe permitirse escalar, rotar, reposicionar y redimensionar los elementos gráficos dentro de un despliegue.	
Observaciones:	

Cuadro 3.11: Historia de usuario No. 6

	Tarea
Número: 1	**Número historia:** 6
Nombre tarea: Mover y redimensionar elementos gráficos en un despliegue.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 05/03/2012	**Fecha fin:** 14/03/2012
Programador responsable: Karel Delgado Alón	
Descripción: Mediante la operación arrastrar/soltar se permite reposicionar los elementos gráficos seleccionados. Además, mediante el inspector de propiedades se permite redimensionar los elementos gráficos.	

Cuadro 3.12: Tarea de ingeniería No. 1 / Historia de usuario No. 6

	Tarea
Número: 2	**Número historia:** 6
Nombre tarea: Rotar y escalar elementos gráficos en un despliegue.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 14/03/2012	**Fecha fin:** 23/03/2012
Programador responsable: Karel Delgado Alón	
Descripción: Los elementos seleccionados dentro de un despliegue muestran un manejador de forma. La función de este manejador es permitir el escalado de los elementos. Mediante el uso de un menú contextual se debe permitir la rotación de los elementos gráficos.	

Cuadro 3.13: Tarea de ingeniería No. 2 / Historia de usuario No. 6

	Historia de usuario
Número: 9	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Actualización de los elementos gráficos de los despliegues.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2.5	**Iteración asignada:** 2
Programador responsable: Alejandro Jiménez López	
Descripción: Debe permitirse la comunicación con algún recolector, mediante la cual se obtengan los nuevos valores de las variables, las cuales están vinculadas a su vez con los elementos gráficos que permiten su visualización.	
Observaciones: Aquí se tiene en cuenta solamente lo concerniente a la lectura de las variables.	

Cuadro 3.14: Historia de usuario No. 9

	Tarea
Número: 1	**Número historia:** 9
Nombre tarea: Envío de las tareas de lectura al recolector.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 29/02/2012	**Fecha fin:** 02/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Al conectarse con el recolector se le envía una tarea de lectura por cada variable.	

Cuadro 3.15: Tarea de ingeniería No. 1 / Historia de usuario No. 9

Tarea	
Número: 2	**Número historia:** 9
Nombre tarea: Actualización del valor de las variables.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 05/03/2012	**Fecha fin:** 09/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Al producirse un cambio en algunas de las variables, el HMI en tiempo de ejecución es notificado con una tarea de lectura, la cual contiene la nueva información de la variable.	

Cuadro 3.16: Tarea de ingeniería No. 2 / Historia de usuario No. 9

Tarea	
Número: 3	**Número historia:** 9
Nombre tarea: Actualización de los elementos gráficos de los despliegues.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12/03/2012	**Fecha fin:** 16/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: El nuevo valor de la variable es propagado para que se actualice el valor de todos los componentes que están asociados a la misma.	

Cuadro 3.17: Tarea de ingeniería No. 3 / Historia de usuario No. 9

Historia de usuario	
Número: 10	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Permitir el envío de tareas de escritura.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1.5	**Iteración asignada:** 2
Programador responsable: Alejandro Jiménez López	
Descripción: Existen elementos gráficos que están dispuestos para brindar la posibilidad de realizar cambios en el estado de los procesos. Para esto es necesaria la comunicación con algún recolector para la actualización de los datos brindados por el operador.	
Observaciones:	

Cuadro 3.18: Historia de usuario No. 10

	Tarea
Número: 1	**Número historia:** 10
Nombre tarea: Enlace de los elementos gráficos a los comandos de escritura.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 19/03/2012	**Fecha fin:** 21/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Los elementos gráficos se pueden asociar a comandos de escritura para permitir el envío de actualizaciones a los valores de las variables.	

Cuadro 3.19: Tarea de ingeniería No. 1 / Historia de usuario No. 10

	Tarea
Número: 2	**Número historia:** 10
Nombre tarea: Actualización de las variables (operación de escritura).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 21/03/2012	**Fecha fin:** 28/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Al realizar una operación que desencadene un cambio en el proceso supervisado se debe enviar al recolector una tarea de escritura que contenga la variable a modificar, así como su nuevo valor.	

Cuadro 3.20: Tarea de ingeniería No. 2 / Historia de usuario No. 10

	Historia de usuario
Número: 17	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Requisitos de desempeño.	
Descripción: Estos requisitos describen los valores que indican el adecuado comportamiento del sistema para las exigencias del cliente, las cuales consisten en el manejo como máximo de 100 variables y 10 despliegues con 30 elementos gráficos cada uno. Además, se le pueden asociar un máximo de 5 elementos gráficos por despliegue a una misma variable. Para el HMI-Runtime se define 30 segundos para la carga de la configuración y 1 segundo para el tiempo de actualización de sus elementos gráficos. Mientras que para el HMI-Editor se define 20 y 30 segundos para salvar y cargar respectivamente, la configuración de un proyecto.	
Observaciones: Los tiempos anteriormente presentados, fueron determinados por un conjunto de expertos pertenecientes a la línea Sistemas Empotrados, los cuales tuvieron en cuenta los requisitos de desempeño del módulo HMI desarrollado para el GALBA.	

Cuadro 3.21: Historia de usuario No. 17

3.4. Tarjetas CRC

Usar las tarjetas Clase-Responsabilidades-Colaboradores (CRC) ayuda a diseñar un sistema. Las mismas permiten que equipos completos de proyecto contribuyan al diseño de la aplicación. Mientras más personas ayuden en esta tarea, mayor número de buenas ideas pueden ser incorporadas. Las tarjetas CRC son usadas para representar objetos, sus responsabilidades y los colaboradores para llevar a cabo su tarea.

Las siguientes tarjetas CRC describen las clases que conforman el sistema.

1. **Submódulo HMI-Editor**

Clase: ISerializable
Responsabilidades: Esta clase ofrece una interfaz común para todos los elementos pertenecientes a un proyecto que son persistidos.
Colaboradores: XmlVisitor, HMIPropertyBrowser

Cuadro 3.22: Tarjeta CRC: Clase ISerializable

Clase: GraphicsItem
Responsabilidades: Esta clase ofrece una interfaz común para todos los componentes gráficos. Implementa el comportamiento de estos frente a los eventos más usuales como pueden ser los provocados por el mouse. Dentro de esta implementación se encuentran las funcionalidades que permiten rotar, escalar y redimensionar los componentes.
Colaboradores: QGraphicsProxyWidget, ISerializable, Movepoint, Variable

Cuadro 3.23: Tarjeta CRC: Clase GraphicsItem

Clase: Variable
Responsabilidades: Esta clase es la representación de las variables que se desean monitorear. Estas variables son vinculadas a los elementos gráficos.
Colaboradores: QWidget, Treelogicalelements, Alarm

Cuadro 3.24: Tarjeta CRC: Clase Variable

Clase: HMIPropertyBrowser
Responsabilidades: Todos los componentes del árbol jerárquico, dígase proyectos, varsviews, screenviews, variables y los elementos gráficos dispuestos en un determinado despliegue, poseen un grupo de propiedades las cuales necesitan ser mostradas al usuario, algunas para su edición y otras solamente para brindar información. Para lograr este objetivo se concibe esta clase, la cual ofrece funcionalidades que permiten visualizar y editar las propiedades de cualquier objeto que instrumente el sistema de propiedades establecido por Qt. Se encarga además de actualizar los valores de las propiedades en caso de que estos cambien.
Colaboradores: QWidget, QtVariantProperty, QtTreePropertyBrowser, QMetaProperty

Cuadro 3.25: Tarjeta CRC: Clase HMIPropertyBrowser

Clase: ProjectExplorer
Responsabilidades: Esta clase se encarga de visualizar y editar la estructura de uno o varios proyectos. Ofrece funcionalidades que permiten crear y eliminar todo tipo de componente dentro de un proyecto, excepto por los elementos gráficos que son manejados fuera de esta clase. Establece quién se muestra en la zona central de la interfaz de usuario y en el inspector de propiedades.
Colaboradores: QTreeWidget, TreeWidgetItem, HMIEditor, CreateVarWindow, AddAlarmsWindow, GraphicsItemsPalette

Cuadro 3.26: Tarjeta CRC: Clase ProjectExplorer

Clase: TreeLogicalElements
Responsabilidades: Esta clase es una interfaz para todos los elementos que se muestran en el árbol jerárquico. Proporciona al HMIPropertyBrowser las propiedades comunes entre todos los elementos.
Colaboradores: ISerializable, TreeWidgetItem

Cuadro 3.27: Tarjeta CRC: Clase TreeLogicalElements

Clase: VarsViews
Responsabilidades: Es una clase organizativa que se encarga de agrupar, crear y eliminar variables.
Colaboradores: QWidget, TreeLogicalElements, Variable, TreeWidgetItem

Cuadro 3.28: Tarjeta CRC: Clase VarsViews

Clase: ScreenViews
Responsabilidades: Esta clase tiene el objetivo de gestionar los despliegues. También se encarga de proporcionarles un contexto adecuado para su visualización.
Colaboradores: QTabWidget, TreeLogicalElements, Screen

Cuadro 3.29: Tarjeta CRC: Clase ScreenViews

Clase: Project
Responsabilidades: Esta clase de encarga de gestionar los VarsViews y los ScreenViews.
Colaboradores: QWidget, TreeLogicalElements, ScreenView, VarsViews, ChooseVarWindow, TreeWidgetItem

Cuadro 3.30: Tarjeta CRC: Clase Project

Clase: Screen
Responsabilidades: Dentro del contexto de los sistemas SCADA podemos encontrar el concepto de despliegue, el cual a grandes rasgos es el de un escenario donde se muestran un conjunto de dispositivos de campo y el comportamiento de las variables que estos exponen. Esta clase es la representación de este concepto.
Colaboradores: QGraphicsScene, QGraphicsView, TreeLogicalElements, DialGraphicsitem, ButtonGraphicsitem, ProgressbarGraphicsitem, LcdnumberGraphicsitem, LineeditGraphicsitem, LabelGraphicsitem, LedGraphicsitem, PlotterGraphicsitem

Cuadro 3.31: Tarjeta CRC: Clase Screen

Clase: GraphicsPalette
Responsabilidades: Esta clase proporciona una paleta de componentes gráficos, donde estos pueden ser seleccionados y agregados a un despliegue. Permite la agrupación de componentes por categorías.
Colaboradores: QTreeWidget, PaletteCategory

Cuadro 3.32: Tarjeta CRC: Clase GraphicsPalette

Clase: MainWindow
Responsabilidades: Esta es la clase que ofrece la estructura donde se ubican todos los demás componentes visuales, como el explorador de proyectos, el inspector de propiedades, la paleta de componentes, entre otros. Se encarga de visualizar en su zona central los despliegues.
Colaboradores: QMainWindow, HMIPropertyBrowser, ProjectExplorer, GraphicsItemsPalette, XmlHmiSerialize, LoadManagement

Cuadro 3.33: Tarjeta CRC: Clase MainWindow

2. **Submódulo HMI-Runtime**

Clase: Task
Responsabilidades: Sirve como contenedor de un objeto Variable, manteniendo información sobre la naturaleza de la tarea: si es de lectura o de escritura.
Colaboradores: QObject

Cuadro 3.34: Tarjeta CRC: Clase Task

Clase: RuntimeLoadHandler
Responsabilidades: Se encarga de leer el archivo XML que exporta el HMI-Editor con las configuraciones de los despliegues.
Colaboradores: QXmlDefaultHandler

Cuadro 3.35: Tarjeta CRC: Clase RuntimeLoadHandler

Clase: LinkerHandler
Responsabilidades: Se encarga de mantener una relación entre las variables y los componentes visuales, con el objetivo de permitir la actualización de los mismos.
Colaboradores: QObject, GraphicsItem

Cuadro 3.36: Tarjeta CRC: Clase LinkerHandler

Clase: TaskClientHandler
Responsabilidades: Se encarga de recibir todas las tareas iniciales de lectura y enviárselas al recolector, además permite enviar al recolector una tarea de escritura para sobreescribir el valor de una determinada variable. Por otra parte, permite conocer el último valor recibido de las variables manejadas.
Colaboradores: QObject, Task, Alarm, Variable, Collector

Cuadro 3.37: Tarjeta CRC: Clase TaskClientHandler

Clase: RuntimeHMI
Responsabilidades: Sirve como contenedor e interfaz de los objetos LinkerHandler y TaskClientHandler.
Colaboradores: QObject, Alarm, GraphicsItem, TasksClientHandler, LinkerHandler

Cuadro 3.38: Tarjeta CRC: Clase RuntimeHMI

Clase: VisualHMI
Responsabilidades: Se encarga de representar los despliegues cargados permitiendo la interacción con el operador.
Colaboradores: QTabWidget, GraphicsItem, Alarm, Screen

Cuadro 3.39: Tarjeta CRC: Clase VisualHMI

3.5. Patrones de diseño

Existen una serie de situaciones comunes en las que los programadores caen una y otra vez, incluso llegando a codificar de forma diferente cada vez. Los patrones de diseño vienen a mitigar un poco este dilema. Se puede decir que son soluciones a diferentes clases de problemas conocidos que pueden ser aplicadas a diferentes códigos, por lo que no hay necesidad de reinventar la rueda. De manera más simple, un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos [27].

Se pueden encontrar una gran variedad de patrones de diseño en el campo de la programación, pero en particular, un conjunto de ellos ha sido aceptado

como estándar. Entre estos patrones se encuentran los conocidos como patrones GoF (Gang of Four) [28] y los denominados como patrones GRASP (General Responsibility Assignment Software Patterns, Patrones de Principios Generales para Asignar Responsabilidades), que ayudan a elegir las clases adecuadas y a decidir cómo estas deben interactuar.

3.5.1. Patrones GoF

1. **Singleton**

   Con el uso de este patrón se asegura que se cree una sola instancia de una clase y se provee un solo punto de acceso global para accederlo. Se crea una clase que gestiona una sola instancia de ella misma y cada vez que se necesite, simplemente se consulta la clase y esta devolverá siempre, la misma instancia.

   En el sistema, este patrón se ve reflejado en la clase que representa el inspector de propiedades. Como se aprecia en la siguiente figura, la clase HMIPropertyBrowser, posee un constructor privado con el objetivo de utilizar el método getInstance para lograr obtener una única instancia a esta clase.

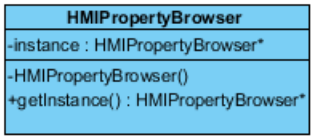

Figura 3.3: Patrón Singleton

2. **Visitor**

   Este patrón debe representar una operación a ser ejecutada sobre los elementos que conforman una estructura. El mismo, permite definir una nueva operación sin tener que cambiar las clases sobre las cuales opera [28].

   Este patrón se utiliza en el mecanismo que permite lograr la persistencia de los elementos de un proyecto. Como se refleja en la figura, la clase ISerializable ofrece una interfaz genérica para los elementos que van ser persistidos. Esta clase incluye un método que se redefine en las clases que heredan de ella, con el objetivo de acceder a diferentes métodos de la clase XmlHmiSerialize.

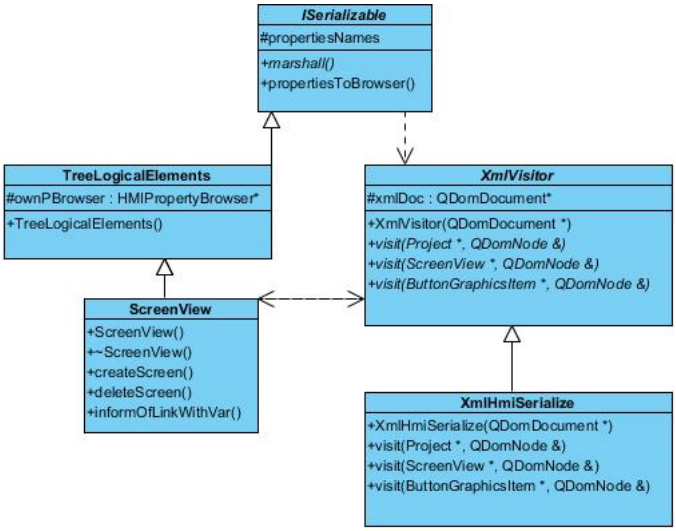

Figura 3.4: Patrón Visitor

3. **Simple Factory**

Este no es exactamente un patrón de diseño, más bien es un idioma de programación, pero es comúnmente usado. Algunos programadores lo confunden con el patrón Factory [29]. El uso del mismo consiste en la existencia de una clase que construye objetos de diferentes clases (las cuales tienen una clase padre común) en dependencia del valor que le sea suministrado.

En la figura, se puede apreciar el uso del mismo, en el mecanismo de creación de los diferentes componentes visuales que son agregados a un despliegue. La clase HMIScene recibe el tipo de elemento gráfico que se desea adicionar, y en dependencia del mismo, esta crea una instancia del elemento especificado.

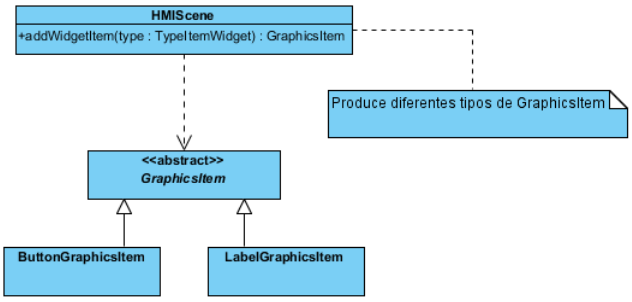

Figura 3.5: Idioma Simple Factory

3.5.2. Patrones GRASP

1. **Experto:** la responsabilidad de realizar una labor es de la clase que tiene o puede tener la información necesaria (atributos), respondiendo al problema de cuál es un principio general para asignar responsabilidades a las clases [27].
2. **Creador:** se asigna la responsabilidad de que una clase B cree un objeto de la clase A, si se cumple uno o más de los casos siguientes (respondiendo al problema de quién debería ser el responsable de la creación de una nueva instancia de alguna clase):

- B agrega objetos de A.
- B contiene objetos de A.
- B registra instancias de objetos de A.
- B utiliza más estrechamente objetos de A.
- B tiene los datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un Experto con respecto a la creación de A) [27].

  La aplicación de este patrón se ve reflejado en la mayoría de las clases que componen el sistema. Como se aprecia en la siguiente figura, la clase Project posee los datos necesarios para la inicialización de objetos de las clases ScreenView y VarsView.

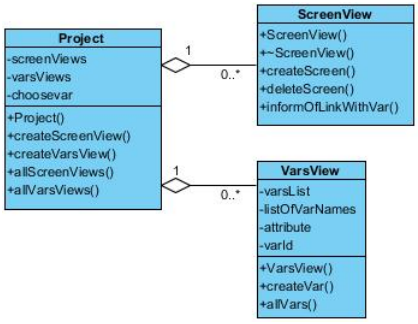

Figura 3.6: Patrón Creador

3.6. Diagramas de clases

Un diagrama de clases de diseño representa las especificaciones de las clases en una aplicación. Entre la información general se encuentran: clases, asociaciones, atributos, métodos, navegabilidad y dependencias [27].

A continuación se muestran separadas por capas, las principales clases que componen ambos submódulos, así como las asociaciones entre ellas.

1. **Submódulo HMI-Editor**

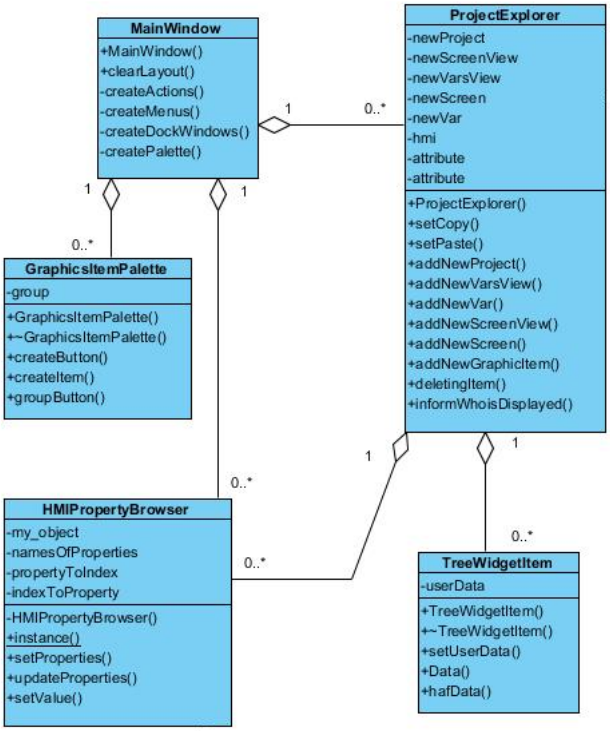

Figura 3.7: Capa Presentación / HMI-Editor

Como se aprecia en este diagrama, la clase MainWindow es la que incluye y organiza a los demás elementos de la interfaz. Estos son: el inspector de propiedades (HMIPropertyBrowser), la paleta de componentes (GraphicsItemPalette) y el explorador de proyectos (ProjectExplorer). Este último, es el encargado de manejar una serie de eventos del mouse que se disparan sobre él, para gestionar los componentes de un proyecto.

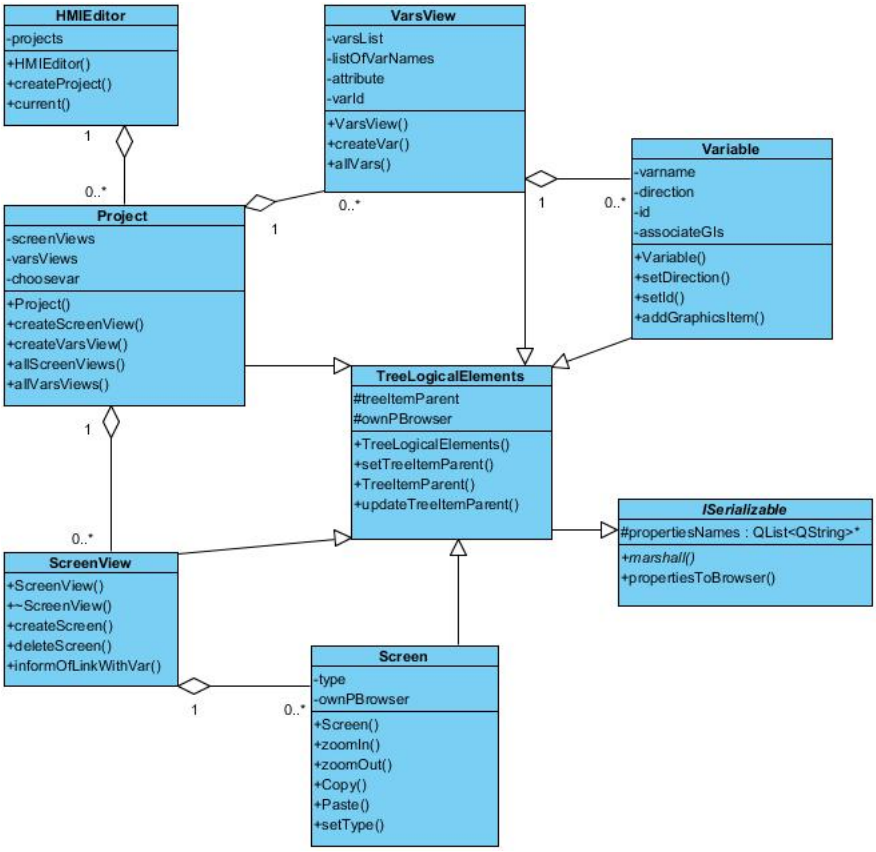

Figura 3.8: Capa Modelo / HMI-Editor

En la capa Modelo existe una estructura que comienza por la clase HMIEditor que gestiona todos los proyectos que pueden estar abiertos en la aplicación. A su vez cada proyecto (Project), manipula a las vistas de despliegues (ScreenView) que incluyen a los despliegues (Screen) y a las vistas de variables (VarsView) que gestionan a las variables (Variable). Todos estos elementos que componen un proyecto poseen una interfaz común (ISerializable) con el objetivo de facilitar el mecanismo de persistencia.

2. **Submódulo HMI-Runtime**

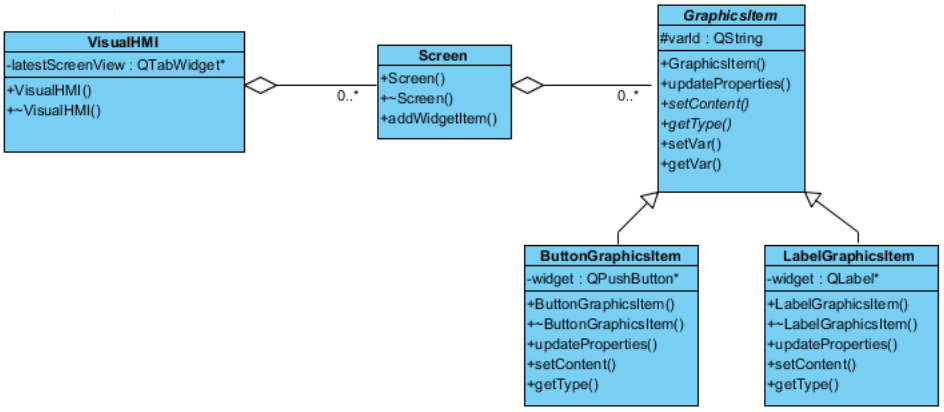

Figura 3.9: Capa Presentación / HMI-Runtime

La clase VisualHMI es la encargada de manipular los diferentes despliegues (Screen) configurados. Esta a su vez, es la que incluye a los elementos gráficos (GraphicItem) que van representarse en cada despliegue.

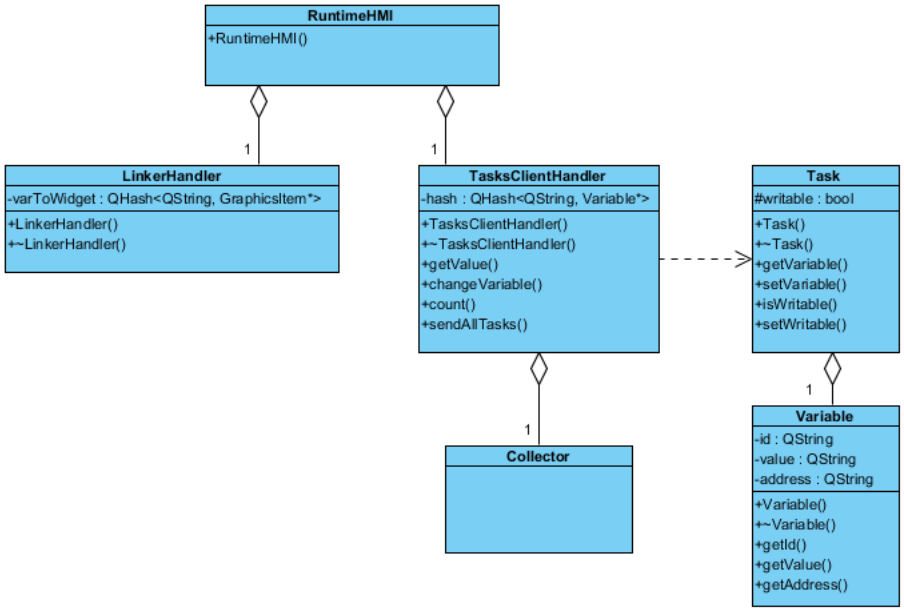

Figura 3.10: Capa Modelo / HMI-Runtime

La clase RuntimeHMI es la responsable de manipular al manejador de vinculación entre variables y elementos gráficos (LinkerHandler) y al manejador de tareas (TasksClientHandler). Este último se relaciona con un recolector (Collector) al cual se envían tareas de escritura/lectura (Task), que se encuentran asociadas a las diferentes variables (Variable) configuradas.

3.7. Diagramas de paquetes

Los paquetes se pueden utilizar para mostrar agrupaciones lógicas de objetos. Cada paquete agrupa un conjunto cohesivo de responsabilidades. Esta es la práctica básica de aplicar la modularidad para dar soporte a la separación de intereses [27].

Seguidamente se muestran estas agrupaciones lógicas de objetos y las dependencias entre ellos, reflejados en los submódulos que componen el sistema, evidenciando en el mismo el patrón arquitectónico seleccionado.

1. **Submódulo HMI-Editor**

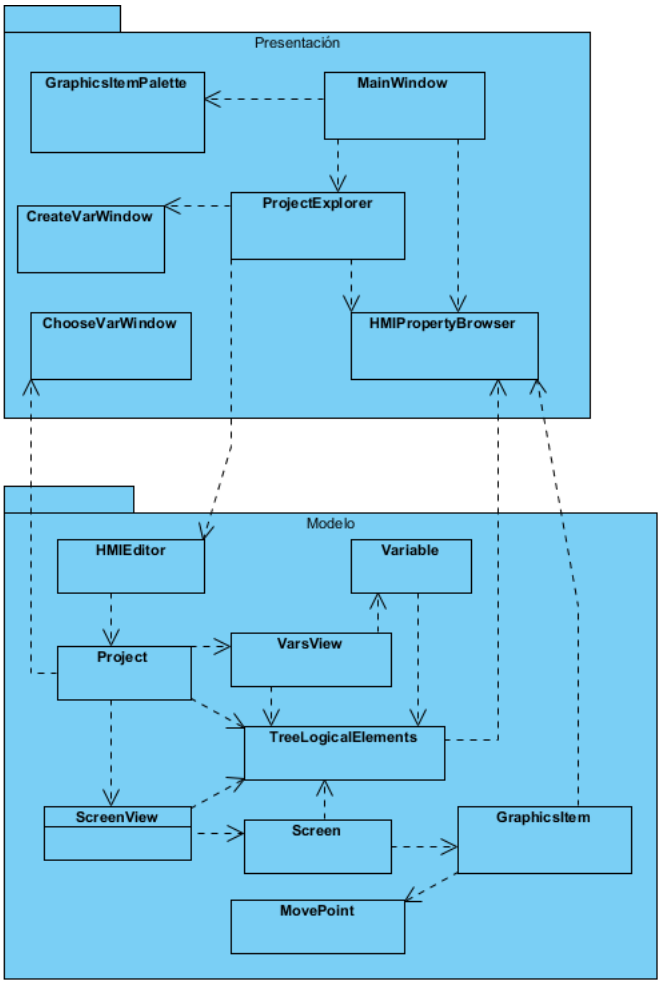

Figura 3.11: Dependencias entre las capas Presentación y Modelo del submódulo HMI-Editor

2. **Submódulo HMI-Runtime**

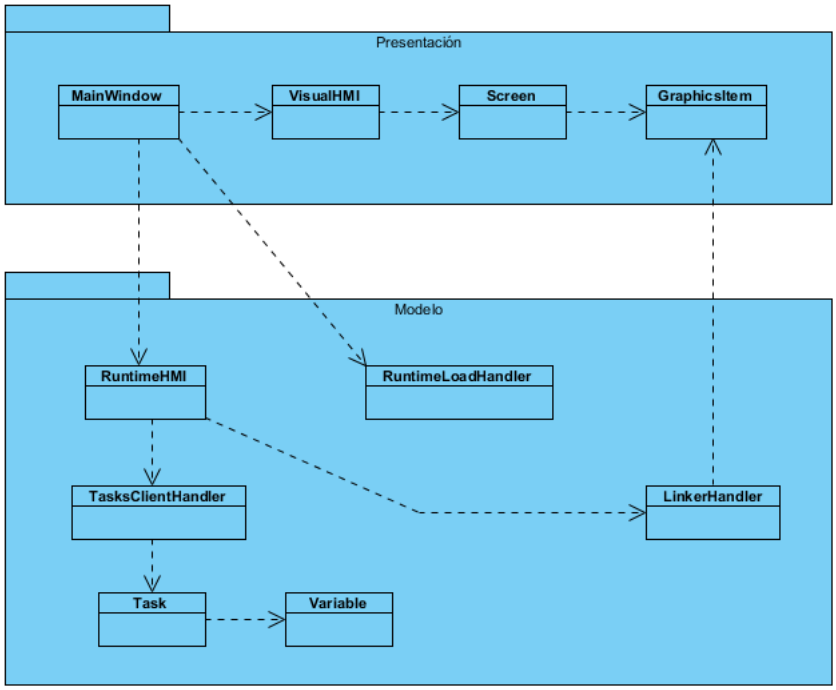

Figura 3.12: Dependencias entre las capas Presentación y Modelo del submódulo HMI-Runtime

3.8. Diagrama de despliegue

Un diagrama de despliegue muestra cómo se configuran las instancias de los componentes de software para su ejecución en los nodos (hardware con memoria y servicios de procesamiento) y la comunicación entre ellos.

En la configuración propuesta se cuenta con un ordenador en el cual se ejecuta el HMI-Editor. Además, está presente una tarjeta basada en microcontroladores -en este caso la CID-300/9- sobre la cual se ejecuta el HMI-Runtime, que está comunicada con otra que sirve de tarjeta de adquisición. El HMI-Editor genera un archivo XML con la configuración realizada, el cual es utilizado por el HMI-Runtime. Esto se realiza sin la presencia de una comunicación directa entre los submódulos. La siguiente imagen muestra la configuración de los nodos en el sistema propuesto.

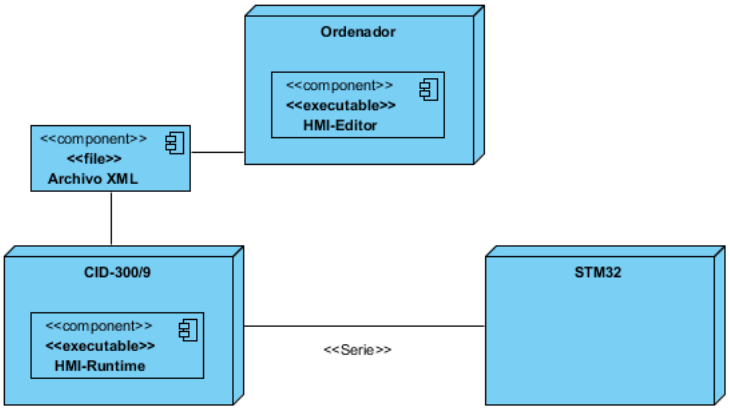

Figura 3.13: Diagrama de despliegue

Conclusiones

En este capítulo, quedaron definidas las características para el desarrollo de la solución, como por ejemplo: la adopción del principio de separación Modelo-Vista como patrón arquitectónico y además, la definición de los principales requisitos funcionales y no funcionales con que debe contar el sistema. Además, fueron concebidas las tareas de ingeniería en relación a las historias de usuario para lograr una planificación en el desarrollo de las mismas. Las tarjetas CRC fueron redactadas para las clases más importantes de la aplicación, con el objetivo de plasmar sus responsabilidades y colaboradores. Además, se definieron los patrones de diseño utilizados para lograr una correcta estructuración del sistema. Por último, fueron realizados diferentes diagramas UML para los dos submódulos, donde se refleja la dependencia entre las capas y las clases que las componen y el diagrama de despliegue que refleja la configuración física y la comunicación entre los nodos.

4 Diseño y realización de pruebas al sistema

Entre los elementos más importantes en el proceso de desarrollo de software, se encuentran las pruebas. A continuación, se muestra un compendio de las pruebas realizadas al sistema obtenido como resultado de este trabajo.

4.1. Pruebas realizadas

En la metodología de desarrollo XP, las historias de usuario son la principal fuente de información a la hora de construir las pruebas de aceptación. A una historia de usuario se le puede definir más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento, y no se considera completa hasta que no superan estas pruebas. El objetivo de las mismas es verificar el cumplimiento de los requisitos, y es responsabilidad del cliente comprobar su ejecución para tomar decisiones al respecto.

Por otra parte, las pruebas de sistema tienen como objetivo ejercitar profundamente el producto, comprobándolo de forma global. Esto posibilita alcanzar una visión similar a su comportamiento en el entorno de producción. Uno de los tipos de pruebas de sistema son las pruebas de rendimiento, las cuales consisten en determinar que los tiempos de respuesta estén dentro de los intervalos establecidos en las especificaciones del sistema (Historia de usuario No. 17).

4.1.1. Ambientes de pruebas

Las pruebas fueron realizadas en dos ambientes diferentes. Uno de ellos dedicado para la realización de las pruebas del submódulo HMI-Editor y otro para el HMI-Runtime. El primero, cuenta con un microprocesador Intel Pentium IV a una frecuencia de 2.2 GHz, una memoria RAM de 512 MB y 10 GB de disco duro, sobre la distribución 10.04 de Ubuntu como sistema operativo. El segundo, cuenta con un microcontrolador ARM-920T a una frecuencia de 400 MHz, una memoria

dinámica de 128 MB y una memoria estática de 512 MB, sobre un sistema operativo Linux empotrado.

4.1.2. Pruebas de aceptación

1. **Submódulo HMI-Editor**

Caso de prueba: 1
Número historia: 1
Nombre: Crear elementos en el árbol semántico.
Descripción: Esta prueba se realiza con el objetivo de verificar que la funcionalidad de crear nuevos elementos, funcione de forma esperada.
Condiciones de ejecución:
Entradas/Pasos de ejecución: Se selecciona la opción New Project del menú File. En el menú contextual del proyecto creado, se selecciona New ScreenView y New VarsView. De forma similar se procede a la creación de los restantes elementos que conforman el árbol.
Resultado esperado: Los elementos creados deben aparecer en el explorador de proyectos.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.1: Caso de prueba No. 1 / Historia de usuario No. 1

Caso de prueba: 2
Número historia: 1
Nombre: Eliminar elementos del árbol semántico.
Descripción: Esta prueba se realiza con el objetivo de verificar que la funcionalidad de eliminar elementos del árbol, funcione de forma esperada.
Condiciones de ejecución: Deben existir elementos en el árbol.
Entradas/Pasos de ejecución: Se selecciona la opción Delete del menú contextual para cada elemento que deba ser removido.
Resultado esperado: Los elementos eliminados deben ser removidos del explorador de proyectos.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.2: Caso de prueba No. 2 / Historia de usuario No. 1

Caso de prueba: 1
Número historia: 3
Nombre: Adición simple de elementos gráficos.
Descripción: Esta prueba se realiza con el objetivo de verificar que la funcionalidad de adicionar un elemento a un despliegue, funcione de forma esperada.
Condiciones de ejecución: Debe existir un proyecto y un screen (despliegue).
Entradas/Pasos de ejecución: Se selecciona un elemento gráfico de la paleta de componentes y se pulsa el botón izquierdo del mouse sobre el despliegue activo.
Resultado esperado: El elemento debe visualizarse en el despliegue activo.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.3: Caso de prueba No. 1 / Historia de usuario No. 3

Caso de prueba: 2
Número historia: 3
Nombre: Adición múltiple de elementos gráficos.
Descripción: Esta prueba se realiza con el objetivo de verificar que la funcionalidad de adicionar de forma múltiple un elemento a un despliegue, funcione de forma esperada.
Condiciones de ejecución: Debe existir un proyecto y un screen (despliegue).
Entradas/Pasos de ejecución: Se selecciona un elemento gráfico de la paleta de componentes y manteniendo además presionada la tecla Shift, se pulsa el botón izquierdo del mouse sobre el despliegue activo, tantas veces como cantidad de componentes se quiera agregar.
Resultado esperado: Los elementos deben visualizarse en el despliegue activo.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.4: Caso de prueba No. 2 / Historia de usuario No. 3

Caso de prueba: 3
Número historia: 3
Nombre: Adición de elementos gráficos mediante la operación copiar/pegar.
Descripción: Esta prueba se realiza con el objetivo de verificar que la funcionalidad de copiar/pegar elementos gráficos, funcione de forma esperada.
Condiciones de ejecución: Debe existir un proyecto, un screen (despliegue) y al menos un elemento gráfico.
Entradas/Pasos de ejecución: Se selecciona uno o varios elementos gráficos del despliegue activo y se procede a ejecutar el mecanismo de copiar/pegar, por cualquiera de las vías conocidas.
Resultado esperado: Los elementos seleccionados se deben duplicar, visualizándose en el despliegue activo.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.5: Caso de prueba No. 3 / Historia de usuario No. 3

Caso de prueba: 1
Número historia: 5
Nombre: Vincular un elemento gráfico con una variable.
Descripción: Esta prueba se realiza con el objetivo de verificar que la funcionalidad de vincular variables a elementos gráficos, funcione de forma esperada.
Condiciones de ejecución: Debe existir un proyecto, un screen (despliegue), un elemento gráfico y una variable.
Entradas/Pasos de ejecución: En el menú contextual del elemento gráfico, se selecciona la opción Link to var. En la ventana emergente que se muestra a continuación, se selecciona la variable que se quiere vincular.
Resultado esperado: El elemento gráfico debe quedar vinculado con la variable.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.6: Caso de prueba No. 1 / Historia de usuario No. 5

Caso de prueba: 1
Número historia: 6
Nombre: Rotar un elemento gráfico.
Descripción: Esta prueba se realiza con el objetivo de verificar que la funcionalidad de rotar un elemento gráfico, funcione de forma esperada.
Condiciones de ejecución: Debe existir un proyecto, un screen (despliegue) y un elemento gráfico.
Entradas/Pasos de ejecución: En el menú contextual del elemento gráfico se selecciona la opción Rotate to Left / Rotate to Right.
Resultado esperado: El elemento gráfico debe quedar rotado.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.7: Caso de prueba No. 1 / Historia de usuario No. 6

Caso de prueba: 2
Número historia: 6
Nombre: Redimensionar un elemento gráfico.
Descripción: Esta prueba se realiza con el objetivo de verificar que la funcionalidad de redimensionar un elemento gráfico, funcione de forma esperada.
Condiciones de ejecución: Debe existir un proyecto, un screen (despliegue) y un elemento gráfico.
Entradas/Pasos de ejecución: Se selecciona el elemento gráfico, se presiona el botón izquierdo del mouse sobre el modificador inferior-derecho y se arrastra.
Resultado esperado: El elemento gráfico debe quedar redimensionado.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.8: Caso de prueba No. 2 / Historia de usuario No. 6

2. **Submódulo HMI-Runtime**

Caso de prueba: 1
Número historia: 9
Nombre: Actualizar el valor de los elementos gráficos.
Descripción: Esta prueba se realiza con el objetivo de verificar que los elementos gráficos muestren el valor de la variable vinculada.
Condiciones de ejecución: Deben existir al menos una variable y un elemento gráfico asociado a esa variable.
Entradas/Pasos de ejecución: Se debe cargar el proyecto.
Resultado esperado: Los elementos gráficos deben visualizar el valor de la variable vinculada de forma continua.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.9: Caso de prueba No. 1 / Historia de usuario No. 9

Caso de prueba: 1
Número historia: 10
Nombre: Permitir el envío de tareas de escritura.
Descripción: Esta prueba se realiza con el objetivo de verificar que las tareas de escritura son enviadas satisfactoriamente.
Condiciones de ejecución: Deben existir al menos una variable y un elemento gráfico (Botón) asociado a esa variable.
Entradas/Pasos de ejecución: Se presiona un elemento gráfico (Botón).
Resultado esperado: El valor de la variable asociada a la tarea de escritura debe ser actualizado.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.10: Caso de prueba No. 1 / Historia de usuario No. 10

Caso de prueba: 1
Número historia: 12
Nombre: Silenciar alarmas.
Descripción: Esta prueba se realiza con el objetivo de verificar que las alarmas seleccionadas cambien su estado a silenciadas.
Condiciones de ejecución: Deben existir alarmas activas y seleccionadas.
Entradas/Pasos de ejecución: Se seleccionan las alarmas y se presiona el botón Silenciar.
Resultado esperado: Las alarmas seleccionadas deben quedar silenciadas.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.11: Caso de prueba No. 1 / Historia de usuario No. 12

Caso de prueba: 2
Número historia: 12
Nombre: Reconocer alarmas.
Descripción: Esta prueba se realiza con el objetivo de verificar que las alarmas seleccionadas cambien su estado a reconocidas.
Condiciones de ejecución: Deben existir alarmas activas y seleccionadas.
Entradas/Pasos de ejecución: Se seleccionan las alarmas y se presiona el botón Reconocer.
Resultado esperado: Las alarmas seleccionadas deben cambiar su estado a reconocidas.
Evaluación de la prueba: Prueba satisfactoria.

Cuadro 4.12: Caso de prueba No. 2 / Historia de usuario No. 12

4.1.3. Pruebas de rendimiento

1. **Submódulo HMI-Editor**

Caso de prueba: 1				
Número historia: 17				
Nombre: Salvar la configuración de un proyecto en el HMI-Editor.				
Descripción: Esta prueba se realiza con el objetivo de verificar que el tiempo que se demora en guardar la configuración del proyecto no sea superior al definido.				
Iteraciones				
No	**Entrada**	**Resultado esperado**	**Resultado obtenido**	**Evaluación**
1	5 despliegues, 50 variables, 50 elementos gráficos y 300 alarmas.	20 segundos	0.0095 segundos	Satisfactoria
2	10 despliegues, 100 variables, 100 elementos gráficos y 600 alarmas.	20 segundos	0.020 segundos	Satisfactoria
3	10 despliegues, 100 variables, 1000 elementos gráficos y 600 alarmas.	20 segundos	0.074 segundos	Satisfactoria

Cuadro 4.13: Caso de prueba No. 1 / Historia de usuario No. 17

Caso de prueba: 2				
Número historia: 17				
Nombre: Cargar la configuración de un proyecto en el HMI-Editor.				
Descripción: Esta prueba se realiza con el objetivo de verificar que el tiempo que se demora en cargar la configuración del proyecto no sea superior al definido.				
Iteraciones				
No	**Entrada**	**Resultado esperado**	**Resultado obtenido**	**Evaluación**
1	5 despliegues, 50 variables, 50 elementos gráficos y 300 alarmas.	30 segundos	0.096 segundos	Satisfactoria
2	10 despliegues, 100 variables, 100 elementos gráficos y 600 alarmas.	30 segundos	0.19 segundos	Satisfactoria
3	10 despliegues, 100 variables, 1000 elementos gráficos y 600 alarmas.	30 segundos	3.6 segundos	Satisfactoria

Cuadro 4.14: Caso de prueba No. 2 / Historia de usuario No. 17

2. **Submódulo HMI-Runtime**

Caso de prueba: 3				
Número historia: 17				
Nombre: Cargar la configuración de un proyecto en el HMI-Runtime.				
Descripción: Esta prueba se realiza con el objetivo de verificar que el tiempo que se demora en cargar la configuración del proyecto no sea superior al definido.				
Iteraciones				
No	**Entrada**	**Resultado esperado**	**Resultado obtenido**	**Evaluación**
1	5 despliegues, 50 variables, 50 elementos gráficos y 300 alarmas.	30 segundos	1.6 segundos	Satisfactoria
2	10 despliegues, 100 variables, 100 elementos gráficos y 600 alarmas.	30 segundos	3.3 segundos	Satisfactoria
3	10 despliegues, 100 variables, 1000 elementos gráficos y 600 alarmas.	30 segundos	27.1 segundos	Satisfactoria

Cuadro 4.15: Caso de prueba No. 3 / Historia de usuario No. 17

Caso de prueba: 4
Número historia: 17
Nombre: Actualización de los valores de los elementos gráficos.
Descripción: Esta prueba se realiza con el objetivo de verificar que el tiempo que se demora en actualizar los elementos gráficos asociados a variables, no sea superior al definido.
Entrada: 10 despliegues, 26 variables y 336 elementos gráficos.
Resultado esperado: Tiempo de actualización de los elementos gráficos inferior a un segundo.
Resultado obtenido: El mismo puede ser apreciado en la figura 4.1 la cual muestra una gráfica de tiempo de actualización (en microsegundos) contra número de muestras (2048) en el tiempo, la misma describe el desempeño del submódulo HMI-Runtime.

Cuadro 4.16: Caso de prueba No. 4 / Historia de usuario No. 17

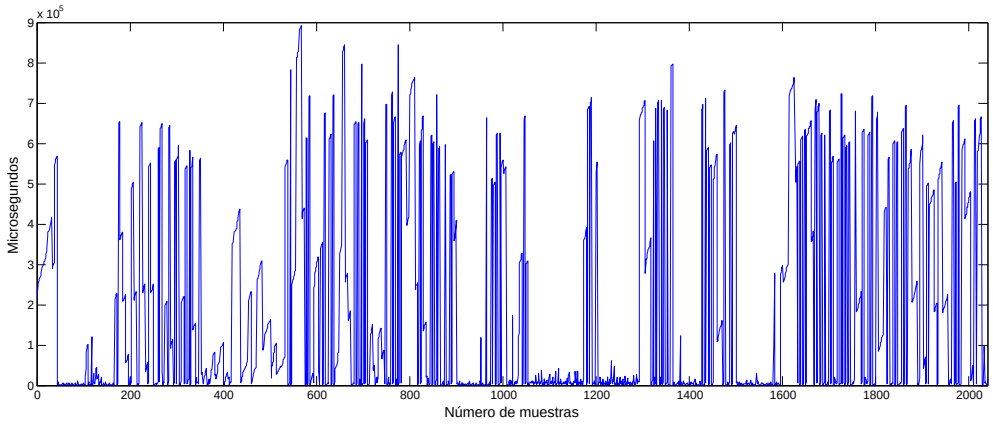

Figura 4.1: Tiempo de actualización de elementos gráficos

Conclusiones

A la solución se le aplicaron un total de 16 casos de prueba de forma general. De este total, 12 de ellas fueron pruebas de aceptación y 4 pruebas de rendimiento. Las pruebas de aceptación brindaron una medida de la correcta implementación de los requisitos funcionales y de su completitud. Mientras que las pruebas de rendimiento estuvieron encaminadas a verificar los requisitos de desempeño recogidos en la historia de usuario No. 17, lo que se logró de manera satisfactoria.

Por lo que se puede concluir que el módulo desarrollado cumple con el objetivo general de la investigación y con las necesidades del cliente.

5 Conclusiones

Como resultado de todo este proceso de desarrollo se obtuvo un módulo HMI que cuenta con funcionalidades básicas que le permiten ser empotrado en la tarjeta CID-300/9 basada en un microcontrolador ARM. Entre las mismas se encuentran: configurar varios despliegues, visualizar la evolución del proceso supervisado, así como permitir la realización de acciones de control sobre el mismo. También permite la gestión de alarmas y ofrece un sumario de variables. Este módulo HMI puede ser integrado a un sistema de supervisión y control para procesos que no sean críticos. A la solución se le efectuaron varias pruebas de aceptación y de rendimiento, que validaron su correcta concepción y ejecución.

6 Recomendaciones

- Desarrollar otros componentes gráficos que puedan enriquecer la configuración de los despliegues.

- Completar el desarrollo del mecanismo de comandos, que permita ejercer un mayor control sobre los procesos.

- Continuar la adición de funcionalidades a la solución, siempre teniendo en cuenta que pueda ser empotrado en la tarjeta CID-300/9.

7 Referencias bibliográficas

1. **Penin, Aquilino Rodríguez.** Sistemas de visualización industrial. s.l. : MARCOMBO, S.A., 2007.
2. **Ortega Estévez, Ramón y Cruz Membrado, Julio César.** Desarrollo del módulo de configuración del HMI web para el sistema de supervisión y control de equipamientos a distancia. Universidad de las Ciencias Informáticas. La Habana : s.n., 2010.
3. **Sehara Driggs, Yosell Luis.** Implementación de un modelo para la configuración de un sistema SCADA. Universidad de las Ciencias Informáticas. La Habana : s.n., 2008.
4. **de Castro Lozano, Carlos y Romero Morales, Cristóbal.** Introducción a SCADA. Universidad de Córdoba. Córdoba : s.n.
5. Introdución a HMI. Universidad Nacional de Quilmes. págs. 1, 2.
6. **Gridling, Gunther y Weiss, Bettina.** Introduction to Microcontrollers. Universidad Tecnológica de Viena. Viena : s.n., 2007.
7. **Samsung.**
   Microcontroller Overview. [En línea] [Citado el: 15 de Octubre de 2011.] http://www.samsung.com/global/business/semiconductor/product/microcontroller/overview.
8. **Cedeño Pozo, Antonio y Espí Muñoz, Roberto Alejandro.** Construcción de sistemas operativos basados en Linux con Buildroot. Universidad de las Ciencias Informáticas. La Habana : s.n., 2012.
9. **ISS, Yokogawa.** Fast/Tools - Product Overview. [En línea] 2003. [Citado el: 15 de Octubre de 2011.] http://www.yokogawa.com.
10. **Bradley, Allen.** RSView32 - Product Profile. [En línea] 2010. [Citado el: 13 de Octubre de 2011.] http://www.rockwellsoftware.com.
11. **AG, Siemens.** Tender Specification. WinCC v7. [En línea] 2009. [Citado el: 14 de Octubre de 2011.] http://www.automation.siemens.com/mcms/tender-specifications/en/process-and-factory-automation/automation-systems/scada-systems/Documents/WinCC_V7-e.doc.

12. —. WinCC - Process visualization for all industries and technologies -SIMATIC HMI, the leading Human Machine Interface solution- Siemens. [En línea] 2010. [Citado el: 14 de Octubre de 2011.] http://www.automation.siemens.com/mcms/human-machine- interface/en/visualization-software/scada/simatic-wincc/Pages/Default.aspx.

13. —. SIMATIC WinCC - Basic Software - SIMATIC HMI, the leading Human Machine Interface solution - Siemens. [En línea] 2010. [Citado el: 14 de Octubre de 2011.] http://www.automation.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/simatic-wincc/Pages/Default.aspx.

14. **Vargas Vento, Daisy Diana.** Sistema para la Seguridad de las Comunicaciones del SCADA Guardián del ALBA. Universidad de las Ciencias Informáticas. La Habana : s.n., 2011. pág. 21.

15. **Sehara Driggs, Yosell Luis.** Manual de usuario del Editor HMI del SCADA GALBA. Universidad de las Ciencias Informáticas. La Habana : s.n., 2008.

16. **Ubuntu.org.** Sobre Ubuntu. [En línea] [Citado el: 15 de Octubre de 2011.] http://doc.ubuntu-es.org/Sobre_Ubuntu.

17. **Florentino, Deidry.** Historia del computador y evolución, arquitectura, lenguajes de programación y algoritmos. [En línea] 2007. [Citado el: 15 de Octubre de 2011.] http://nectacellcomputer.jimdo.com/historia-del-computador-arquitectura-lenguajes-de- programación-y-algoritmos/.

18. **Ecured.** EcuRed. [En línea] [Citado el: 14 de Octubre de 2011.] http://www.ecured.cu/index.php/C++.

19. **Stroustrup, Bjarne.** The Design and Evolution of C++. s.l. : Addison-Wesley, 1994.

20. **Nokia, Qt.** Qt SDK - Qt - A cross-platform application and UI framework. [En línea] 2011. [Citado el: 15 de Octubre de 2011.] http://qt.nokia.com/products/qt-sdk.

21. —. Qt Creator IDE and tools - Qt - A cross-platform application and UI framework. [En línea] 2011. [Citado el: 15 de Octubre de 2011.] http://qt.nokia.com/products/developer-tools/.

22. —. QtCreator Whitepaper. [En línea] 2011. [Citado el: 15 de Octubre de 2011.] http://qt- project.org/wiki/QtCreatorWhitepaper.

23. **Object Management Group.**

Introduction to UML. [En línea] 2005. [Citado el: 15 de Octubre de 2011.] http://www.omg.org/gettingstarted/what_is_uml.htm.

24. **EcuRed.** Herramienta CASE - EcuRed. [En línea] [Citado el: 16 de Octubre de 2011.] http://www.ecured.cu/index.php/Herramienta_CASE.
25. **Castillo, Oswaldo, Figueroa, Daniel y Sevilla, Héctor.** Programación Extrema. [En línea] [Citado el: 16 de Octubre de 2011.] http://programacionextrema.tripod.com/index.htm.
26. **Pressman, Roger S.** Ingeniería del software. Un enfoque práctico. s.l. : Mc Graw Hill, 2005.
27. **Larman, Craig.** UML y Patrones. s.l. : Prentice Hall, 2002.
28. **Holzner, Steve.** Design Patterns for Dummies. s.l. : Wiley Publishing, Inc., 2006. pág. 30.
29. **Freeman, Eric, y otros.** Design Patterns. s.l. : O'Reilly, 2006. pág. 150.

8 Anexos

8.1. Anexo 1

	Historia de usuario
Número: 1	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Gestión de los elementos del árbol semántico de proyecto.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2.5	**Iteración asignada:** 1
Programador responsable: Karel Delgado Alón	
Descripción: Permitir la adición, eliminación y edición de elementos como: proyecto, vista de despliegues, vista de variables, despliegues, variables y otros, todos ordenados de forma jerárquica.	
Observaciones:	

Cuadro 8.1: Historia de usuario No. 1

	Historia de usuario
Número: 2	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Inspección de propiedades de los elementos del árbol semántico.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 2.5	**Iteración asignada:** 1
Programador responsable: Karel Delgado Alón	
Descripción: Todos los componentes del árbol semántico, dígase: proyecto, vista de despliegues, vista de variables, despliegues, elementos gráficos, entre otros, deben mostrar sus propiedades en el inspector al ser seleccionados.	
Observaciones:	

Cuadro 8.2: Historia de usuario No. 2

	Historia de usuario
Número: 3	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Adición de elementos gráficos a los despliegues.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2.5	**Iteración asignada:** 1
Programador responsable: Karel Delgado Alón	
Descripción: Debe existir una paleta de componentes que contenga los elementos gráficos que se pueden agregar en un despliegue determinado. La misma debe estar organizada por las diferentes categorías en las que se pueden agrupar los gráficos.	
Observaciones:	

Cuadro 8.3: Historia de usuario No. 3

	Historia de usuario
Número: 4	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Serialización de la configuración de un proyecto.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 5	**Iteración asignada:** 1
Programador responsable: Karel Delgado Alón	
Descripción: Los proyectos sufren una serie de cambios y transformaciones, las cuales constituyen la configuración del mismo. Esta es utilizada por el HMI en tiempo de ejecución, y por el HMI en tiempo de edición, por lo que es necesario salvar y cargar dicha configuración.	
Observaciones:	

Cuadro 8.4: Historia de usuario No. 4

	Historia de usuario
Número: 5	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Vinculación de los elementos gráficos con variables.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Karel Delgado Alón	
Descripción: Los elementos gráficos en los despliegues deben representar el comportamiento de alguna variable recogida en algún proceso, por lo que debe existir un vínculo entre los mismos.	
Observaciones:	

Cuadro 8.5: Historia de usuario No. 5

	Historia de usuario
Número: 6	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Edición de la apariencia y ubicación de los elementos gráficos.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Karel Delgado Alón	
Descripción: Debe permitirse escalar, rotar, reposicionar y redimensionar los elementos gráficos dentro de un despliegue.	
Observaciones:	

Cuadro 8.6: Historia de usuario No. 6

	Historia de usuario
Número: 7	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Gestión de alarmas.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1.5	**Iteración asignada:** 2
Programador responsable: Karel Delgado Alón	
Descripción: Permitir la adición, eliminación y edición de las alarmas. Las mismas siempre van a estar asociadas a una variable.	
Observaciones:	

Cuadro 8.7: Historia de usuario No. 7

	Historia de usuario
Número: 8	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Inicialización de los despliegues.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3.5	**Iteración asignada:** 2
Programador responsable: Alejandro Jiménez López	
Descripción: Al iniciar el HMI en tiempo de ejecución se deben cargar los despliegues configurados pertenecientes a un proyecto. La información de la configuración de un proyecto se obtiene a través de un fichero en formato XML que exporta el editor.	
Observaciones: En estos momentos se está utilizando una tarjeta basada en microcontroladores creada por el ICID.	

Cuadro 8.8: Historia de usuario No. 8

	Historia de usuario
Número: 9	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Actualización de los elementos gráficos de los despliegues.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2.5	**Iteración asignada:** 2
Programador responsable: Alejandro Jiménez López	
Descripción: Debe permitirse la comunicación con algún recolector, mediante la cual se obtengan los nuevos valores de las variables, las cuales están vinculadas a su vez con los elementos gráficos que permiten su visualización.	
Observaciones: Aquí se tiene en cuenta solamente lo concerniente a la lectura de las variables.	

Cuadro 8.9: Historia de usuario No. 9

	Historia de usuario
Número: 10	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Permitir el envío de tareas de escritura.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1.5	**Iteración asignada:** 2
Programador responsable: Alejandro Jiménez López	
Descripción: Existen elementos gráficos que están dispuestos para brindar la posibilidad de realizar cambios en el estado de los procesos. Para esto es necesaria la comunicación con algún recolector para la actualización de los datos brindados por el operador.	
Observaciones:	

Cuadro 8.10: Historia de usuario No. 10

	Historia de usuario
Número: 11	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Brindar un sumario de variables.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1.5	**Iteración asignada:** 3
Programador responsable: Alejandro Jiménez López	
Descripción: Se debe mostrar un resumen con todas las variables configuradas en los despliegues, para que de una forma centralizada se puedan consultar los valores de las mismas.	
Observaciones:	

Cuadro 8.11: Historia de usuario No. 11

	Historia de usuario
Número: 12	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Gestión de alarmas en tiempo de ejecución.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 4	**Iteración asignada:** 3
Programador responsable: Alejandro Jiménez López	
Descripción: Se le debe presentar al operador, las alarmas que se disparen en el sistema, con los datos necesarios para su reconocimiento. Además, se debe permitir la manipulación de las mismas, entiéndase modificar su estado a silenciadas o a reconocidas.	
Observaciones:	

Cuadro 8.12: Historia de usuario No. 12

	Historia de usuario
Número: 13	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Requisitos de software.	
Descripción: Se debe poseer un sistema operativo Linux que consuma pocos recursos de hardware y además cuente con los siguientes módulos: Qt 4.7.* y Xorg server 1.7.5.	
Observaciones:	

Cuadro 8.13: Historia de usuario No. 13

Historia de usuario	
Número: 14	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Requisitos de hardware.	
Descripción: Para la ejecución del HMI-Runtime se debe contar con una tarjeta basada en microcontroladores que posea como mínimo las siguientes prestaciones: - uC: ARM-920T. - Frecuencia del CPU: 400 MHz. - Memoria estática: 512 MB. - Memoria dinámica: 128 MB. - Puertos: 1 Serie, 2 USB, 1 Ethernet y 1 SD. - Pantalla LCD. Por otra parte para la ejecución del HMI-Editor se debe disponer de un ordenador que tenga como mínimo las prestaciones siguientes: - uP: Intel Pentium IV. - Frecuencia del CPU: 2.2 GHz. - Memoria estática: 10 GB. - Memoria dinámica: 512 MB.	
Observaciones:	

Cuadro 8.14: Historia de usuario No. 14

Historia de usuario	
Número: 15	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Restricciones en el diseño e implementación.	
Descripción: Para el desarrollo de la solución se definen una serie de restricciones: - Lenguaje de programación: C++. - Framework: Qt 4.7.4. - Entorno integrado de desarrollo: Qt Creator.	
Observaciones:	

Cuadro 8.15: Historia de usuario No. 15

Historia de usuario	
Número: 16	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Requisitos de apariencia o interfaz externa.	
Descripción: El HMI-Runtime debe presentar una interfaz sencilla que permita una adecuada visualización e interacción, teniendo en cuenta los requisitos de hardware descritos anteriormente. Se debe hacer un buen uso del poco espacio del que se dispone, destinando una mayor área para la visualización de los despliegues.	
Observaciones: Ver historia de usuario #14.	

Cuadro 8.16: Historia de usuario No. 16

Historia de usuario	
Número: 17	**Usuario:** Línea de Sistemas Empotrados
Nombre historia: Requisitos de desempeño.	
Descripción: Estos requisitos describen los valores que indican el adecuado comportamiento del sistema para las exigencias del cliente, las cuales consisten en el manejo como máximo de 100 variables y 10 despliegues con 30 elementos gráficos cada uno. Además, se le pueden asociar un máximo de 5 elementos gráficos por despliegue a una misma variable. Para el HMI-Runtime se define 30 segundos para la carga de la configuración y 1 segundo para el tiempo de actualización de sus elementos gráficos. Mientras que para el HMI-Editor se define 20 y 30 segundos para salvar y cargar respectivamente, la configuración de un proyecto.	
Observaciones: Los tiempos anteriormente presentados, fueron determinados por un conjunto de expertos pertenecientes a la línea Sistemas Empotrados, los cuales tuvieron en cuenta los requisitos de desempeño del módulo HMI desarrollado para el GALBA.	

Cuadro 8.17: Historia de usuario No. 17

8.2. Anexo 2

	Tarea
Número: 1	**Número historia:** 1
Nombre tarea: Diseño del inspector de proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 02/11/2011	**Fecha fin:** 04/11/2011
Programador responsable: Karel Delgado Alón	
Descripción: La interfaz de usuario cuenta con un panel en su parte izquierda superior donde se van a desplegar los proyectos que se encuentren abiertos y por cada uno de ellos los elementos que lo compongan.	

Cuadro 8.18: Tarea de ingeniería No. 1 / Historia de usuario No. 1

	Tarea
Número: 2	**Número historia:** 1
Nombre tarea: Implementación de la lógica de funcionamiento del inspector de proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 07/11/2011	**Fecha fin:** 18/11/2011
Programador responsable: Karel Delgado Alón	
Descripción: Si se presiona el botón derecho del mouse sobre el inspector se da lo opción de agregar un nuevo proyecto, si se hace sobre un proyecto se da la opción de agregar una vista de despliegues, una vista de variables o de eliminar el proyecto. Si el mouse es presionado sobre una vista de despliegues se da la opción de agregar despliegues o de eliminar la vista. Cuando ocurre sobre una vista de variables se muestra una ventana donde se introduce el nombre y la dirección de la variable. Estos elementos que componen el árbol de proyecto, pueden ser clickeados con el botón derecho del mouse lo que provoca que se muestren o no, según sea su naturaleza, en la zona de trabajo que se define entre el inspector del proyecto, el inspector de propiedades y la paleta de componentes.	

Cuadro 8.19: Tarea de ingeniería No. 2 / Historia de usuario No. 1

	Tarea
Número: 1	**Número historia:** 2
Nombre tarea: Estudio y prueba de de la biblioteca QPropertyBrowser.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 21/11/2011	**Fecha fin:** 23/11/2011
Programador responsable: Karel Delgado Alón	
Descripción: Esta biblioteca proporciona una API para trabajar con las propiedades de elementos que instrumenten el sistema de propiedades propuesto por Qt.	

Cuadro 8.20: Tarea de ingeniería No. 1 / Historia de usuario No. 2

	Tarea
Número: 2	**Número historia:** 2
Nombre tarea: Diseño del inspector de propiedades.	
Tipo de tarea: Investigación	**Puntos estimados:** 0.5
Fecha inicio: 23/11/2011	**Fecha fin:** 25/11/2011
Programador responsable: Karel Delgado Alón	
Descripción: La interfaz de usuario cuenta con un panel en su parte izquierda inferior donde se van a desplegar las propiedades del elemento seleccionado en un momento dado.	

Cuadro 8.21: Tarea de ingeniería No. 2 / Historia de usuario No. 2

	Tarea
Número: 3	**Número historia:** 2
Nombre tarea: Implementación del inspector de propiedades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 28/11/2011	**Fecha fin:** 07/12/2011
Programador responsable: Karel Delgado Alón	
Descripción: El inspector de propiedades muestra las propiedades definidas para cada elemento que compone el árbol semántico, así como de los elementos gráficos que pueden aparecer en un despliegue, también brinda la posibilidad de modificar dichas propiedades.	

Cuadro 8.22: Tarea de ingeniería No. 3 / Historia de usuario No. 2

	Tarea
Número: 1	**Número historia:** 3
Nombre tarea: Diseño de la paleta de componentes.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 07/12/2011	**Fecha fin:** 09/12/2011
Programador responsable: Karel Delgado Alón	
Descripción: La interfaz de usuario cuenta con un panel derecho que proporciona varios botones, cada uno de los cuales representa un elemento gráfico adicionable a los despliegues. Estos deben estar agrupados en categorías.	

Cuadro 8.23: Tarea de ingeniería No. 1 / Historia de usuario No. 3

	Tarea
Número: 2	**Número historia:** 3
Nombre tarea: Implementación de la paleta de componentes.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12/12/2011	**Fecha fin:** 16/12/2011
Programador responsable: Karel Delgado Alón	
Descripción: La paleta de componentes permite mostrar los botones que representan los componentes gráficos, así como seleccionar uno de ellos a la vez, lo que establece el tipo de elemento que se desea agregar si se da click sobre algún despliegue.	

Cuadro 8.24: Tarea de ingeniería No. 2 / Historia de usuario No. 3

	Tarea
Número: 3	**Número historia:** 3
Nombre tarea: Implementación de la adición de los elementos gráficos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 09/01/2012	**Fecha fin:** 13/01/2012
Programador responsable: Karel Delgado Alón	
Descripción: Al dar click izquierdo sobre algún despliegue en la zona de trabajo, se agregará un componente gráfico si existe alguno seleccionado previamente en la paleta de componentes.	

Cuadro 8.25: Tarea de ingeniería No. 3 / Historia de usuario No. 3

	Tarea
Número: 1	**Número historia:** 4
Nombre tarea: Diseño de la interfaz para salvar la configuración de un proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 16/01/2012	**Fecha fin:** 18/01/2012
Programador responsable: Karel Delgado Alón	
Descripción: En el menú archivo y en la barra de herramienta que se encuentra en la parte superior de la interfaz, existe una opción que abre una ventana que permite ubicar el archivo que se desea salvar.	

Cuadro 8.26: Tarea de ingeniería No. 1 / Historia de usuario No. 4

	Tarea
Número: 2	**Número historia:** 4
Nombre tarea: Diseño de la interfaz para cargar un proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 18/01/2012	**Fecha fin:** 20/01/2012
Programador responsable: Karel Delgado Alón	
Descripción: En el menú archivo y en la barra de herramienta que se encuentra en la parte superior de la interfaz, existe una opción que abre una ventana que permite localizar el archivo que se desea cargar.	

Cuadro 8.27: Tarea de ingeniería No. 2 / Historia de usuario No. 4

	Tarea
Número: 3	**Número historia:** 4
Nombre tarea: Salvar la configuración de un proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 23/01/2012	**Fecha fin:** 03/02/2012
Programador responsable: Karel Delgado Alón	
Descripción: Se crea un archivo XML y se ubica en algún directorio, acto seguido se procede a guardar toda la información relevante correspondiente a los diferentes elementos que componen el árbol semántico.	

Cuadro 8.28: Tarea de ingeniería No. 3 / Historia de usuario No. 4

	Tarea
Número: 4	**Número historia:** 4
Nombre tarea: Cargar la configuración de un proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 06/02/2012	**Fecha fin:** 17/02/2012
Programador responsable: Karel Delgado Alón	
Descripción: Después de localizar el fichero, se da paso al proceso de lectura, mediante el cual se capturan los datos pertenecientes a los elementos guardados, permitiendo crearlos nuevamente.	

Cuadro 8.29: Tarea de ingeniería No. 4 / Historia de usuario No. 4

	Tarea
Número: 1	**Número historia:** 5
Nombre tarea: Diseño de la vinculación de los elementos gráficos con las variables.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 20/02/2012	**Fecha fin:** 22/02/2012
Programador responsable: Karel Delgado Alón	
Descripción: Los elementos gráficos que se encuentran ubicados en alguna posición dentro de un despliegue, al ser seleccionados con el botón derecho del mouse muestran un menú contextual donde se brinda la opción de vincular el mismo con una variable de las que pertenezcan al proyecto. Para este propósito se muestra una ventana emergente con las variables disponibles.	

Cuadro 8.30: Tarea de ingeniería No. 1 / Historia de usuario No. 5

	Tarea
Número: 2	**Número historia:** 5
Nombre tarea: Vinculación de los elementos gráficos con las variables.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 22/02/2012	**Fecha fin:** 02/03/2012
Programador responsable: Karel Delgado Alón	
Descripción: Cuando se muestra la ventana emergente con las variables disponibles, el usuario tiene la posibilidad de escoger una de ellas, en caso de escogerla se procede a vincular esta variable mediante su identificador con el elemento seleccionado.	

Cuadro 8.31: Tarea de ingeniería No. 2 / Historia de usuario No. 5

	Tarea
Número: 1	**Número historia:** 6
Nombre tarea: Mover y redimensionar elementos gráficos en un despliegue.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 05/03/2012	**Fecha fin:** 14/03/2012
Programador responsable: Karel Delgado Alón	
Descripción: Mediante la operación arrastrar/soltar se permite reposicionar los elementos gráficos seleccionados. Además, mediante el inspector de propiedades se permite redimensionar los elementos gráficos.	

Cuadro 8.32: Tarea de ingeniería No. 1 / Historia de usuario No. 6

	Tarea
Número: 2	**Número historia:** 6
Nombre tarea: Rotar y escalar elementos gráficos en un despliegue.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 14/03/2012	**Fecha fin:** 23/03/2012
Programador responsable: Karel Delgado Alón	
Descripción: Los elementos seleccionados dentro de un despliegue muestran un manejador de forma. La función de este manejador es permitir el escalado de los elementos. Mediante el uso de un menú contextual se debe permitir la rotación de los elementos gráficos.	

Cuadro 8.33: Tarea de ingeniería No. 2 / Historia de usuario No. 6

	Tarea
Número: 1	**Número historia:** 7
Nombre tarea: Diseño de la interfaces para gestionar las alarmas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 26/03/2012	**Fecha fin:** 28/03/2012
Programador responsable: Karel Delgado Alón	
Descripción: La interfaz cuenta con dos diálogos, el primero ofrece la posibilidad de agregar y eliminar alarmas de diferentes tipos, a una variable determinada. El segundo permite editar los atributos de una alarma en particular.	

Cuadro 8.34: Tarea de ingeniería No. 1 / Historia de usuario No. 7

	Tarea
Número: 2	**Número historia:** 7
Nombre tarea: Implementación de la gestión de las alarmas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 28/03/2012	**Fecha fin:** 04/04/2012
Programador responsable: Karel Delgado Alón	
Descripción: Los elementos de tipo variable, al ser seleccionados con el botón derecho del mouse se muestra un menú contextual que da la opción de agregarle alarmas. Con este propósito se muestra un cuadro de diálogo, donde al presionar el botón Agregar se captura el tipo y la prioridad escogidos y se crea una nueva alarma, la cual es adicionada a la variable seleccionada, y pasa a mostrarse en un listado en dicha ventana. Seleccionando una de las alarmas mostradas se puede eliminar o pasar a otro cuadro de diálogo que permite editar las propiedades de la misma.	

Cuadro 8.35: Tarea de ingeniería No. 2 / Historia de usuario No. 7

	Tarea
Número: 1	**Número historia:** 8
Nombre tarea: Diseño de la interfaz general del HMI-Runtime.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 06/02/2012	**Fecha fin:** 08/02/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Se debe contar con una interfaz minimalista y de fácil interacción, debido al poco espacio disponible en la pantalla de la CID-300/9.	

Cuadro 8.36: Tarea de ingeniería No. 1 / Historia de usuario No. 8

	Tarea
Número: 2	**Número historia:** 8
Nombre tarea: Cargar la configuración de un proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 08/02/2012	**Fecha fin:** 17/02/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Se debe cargar la información de un proyecto que se encuentra contenida en un fichero XML exportado por el HMI-Editor.	

Cuadro 8.37: Tarea de ingeniería No. 2 / Historia de usuario No. 8

	Tarea
Número: 3	**Número historia:** 8
Nombre tarea: Inicialización de los despliegues.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 20/02/2012	**Fecha fin:** 29/02/2012
Programador responsable: Alejandro Jiménez López	
Descripción: La información cargada se utiliza para crear los diferentes despliegues con sus respectivos elementos gráficos y la relación de estos con las variables a los que están asociados, así como la relación de las variables con las alarmas.	

Cuadro 8.38: Tarea de ingeniería No. 3 / Historia de usuario No. 8

	Tarea
Número: 1	**Número historia:** 9
Nombre tarea: Envío de las tareas de lectura al recolector.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 29/02/2012	**Fecha fin:** 02/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Al conectarse con el recolector se le envía una tarea de lectura por cada variable.	

Cuadro 8.39: Tarea de ingeniería No. 1 / Historia de usuario No. 9

	Tarea
Número: 2	**Número historia:** 9
Nombre tarea: Actualización del valor de las variables.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 05/03/2012	**Fecha fin:** 09/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Al producirse un cambio en algunas de las variables, el HMI en tiempo de ejecución es notificado con una tarea de lectura, la cual contiene la nueva información de la variable.	

Cuadro 8.40: Tarea de ingeniería No. 2 / Historia de usuario No. 9

	Tarea
Número: 3	**Número historia:** 9
Nombre tarea: Actualización de los elementos gráficos de los despliegues.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12/03/2012	**Fecha fin:** 16/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: El nuevo valor de la variable es propagado para que se actualice el valor de todos los componentes que están asociados a la misma.	

Cuadro 8.41: Tarea de ingeniería No. 3 / Historia de usuario No. 9

	Tarea
Número: 1	**Número historia:** 10
Nombre tarea: Enlace de los elementos gráficos a los comandos de escritura.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 19/03/2012	**Fecha fin:** 21/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Los elementos gráficos se pueden asociar a comandos de escritura para permitir el envío de actualizaciones a los valores de las variables.	

Cuadro 8.42: Tarea de ingeniería No. 1 / Historia de usuario No. 10

	Tarea
Número: 2	**Número historia:** 10
Nombre tarea: Actualización de las variables (operación de escritura).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 21/03/2012	**Fecha fin:** 28/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Al realizar una operación que desencadene un cambio en el proceso supervisado se debe enviar al recolector una tarea de escritura que contenga la variable a modificar, así como su nuevo valor.	

Cuadro 8.43: Tarea de ingeniería No. 2 / Historia de usuario No. 10

	Tarea
Número: 1	**Número historia:** 11
Nombre tarea: Diseño de la interfaz del sumario de variables.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 28/03/2012	**Fecha fin:** 30/03/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Se debe contar con una tabla que recoja la totalidad de las variables configuradas, mostrando en cada instante el valor que poseen las mismas.	

Cuadro 8.44: Tarea de ingeniería No. 1 / Historia de usuario No. 11

	Tarea
Número: 2	**Número historia:** 11
Nombre tarea: Actualización del sumario de variables.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 02/04/2012	**Fecha fin:** 06/04/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Esta tabla debe ser actualizada de igual forma a como se actualizan los elementos gráficos presentes en los despliegues.	

Cuadro 8.45: Tarea de ingeniería No. 2 / Historia de usuario No. 11

	Tarea
Número: 1	**Número historia:** 12
Nombre tarea: Diseño de la interfaz del sumario de alarmas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 09/04/2012	**Fecha fin:** 13/04/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Se debe mostrar de forma simplificada la información básica de las alarmas que se activan en el sistema y permitir cambiar el estado de las mismas.	

Cuadro 8.46: Tarea de ingeniería No. 1 / Historia de usuario No. 12

	Tarea
Número: 2	**Número historia:** 12
Nombre tarea: Chequeo de alarmas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 16/04/2012	**Fecha fin:** 25/04/2012
Programador responsable: Alejandro Jiménez López	
Descripción: Al recibir una actualización del valor de una variable, esta debe pasar por un proceso de chequeo, para comprobar si ese valor está dentro de los límites definidos por algunas de las alarmas de la variable. En caso afirmativo, la alarma es disparada y reflejada en una vista de alarmas, asignándole un color en dependencia de su prioridad.	

Cuadro 8.47: Tarea de ingeniería No. 2 / Historia de usuario No. 12

	Tarea
Número: 3	**Número historia:** 12
Nombre tarea: Reconocimiento de alarmas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 25/04/2012	**Fecha fin:** 02/05/2012
Programador responsable: Alejandro Jiménez López	
Descripción: El operador tiene la posibilidad de cambiar el estado de las alarmas. Esta operación se refiere a que el mismo puede seleccionar alarmas activas y silenciarlas (manteniéndolas en la vista de alarmas, con un color diferente a las activas) o reconocerlas (eliminándolas de la vista de alarmas). Las mismas son lanzadas nuevamente si no se encuentran activas.	

Cuadro 8.48: Tarea de ingeniería No. 3 / Historia de usuario No. 12