“Simulador del dispositivo PLC Series 90-70 para realizar las pruebas al manejador GE Fanuc del SCADA GALBA”

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

Autor: Zahiro Lázaro Lamadrid Almeida

Tutores: Ing. Yordanis Bridón Danger

Ing. José Carlos Abraham Ravelo

La Habana, Junio 2015

“Año 57 del triunfo de la revolución”

Declaración de autoría

Declaro ser el autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo el presente a los____ días del mes de _____ del año _____.

_____

Firma del Autor

Zahiro Lázaro Lamadrid Almeida

_____

Firma del Tutor

Ing. Yordanis Bridón Danger

_____

Firma del Tutor

Ing. José Carlos Abraham Ravelo

Datos de contacto

Tutor: Ing. Yordanis Bridón Danger

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas

Instituto donde se graduó: Universidad de las Ciencias Informáticas (UCI)

Correo electrónico: ybridon@uci.cu

Años de graduado: 7 años de graduado

Años de experiencia en el tema: 6 años de experiencia

Tutor: Ing. José Carlos Abraham Ravelo

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas

Instituto donde se graduó: Universidad de las Ciencias Informáticas (UCI)

Correo electrónico: jravelo@uci.cu

Años de graduado: 2 años de graduado

Años de experiencia en el tema: 3 años de experiencia

Agradecimientos

Quisiera comenzar agradeciendo a los mayores responsables de que haya culminado satisfactoriamente este trabajo de diploma, ellos son mis tutores Yordanis y José Carlos. Gracias por su entrega y sacrificio. Siempre podrán contar conmigo.

A mi novia Lidia, por estar todo el tiempo a mi lado dándome amor, apoyándome y halándome las orejas cuando quería hacer otra cosa que no fuera la tesis.

A mis padres porque gracias a ellos estoy vivo y en especial a mi madre, la persona que siempre está junto a mí, en las buenas y en las malas.

A mis hermanas Rosario y Ana Cecilia, quienes son como dos luces que me iluminan.

A mi familia que siempre me apoya Luis Miguel que sigue mis pasos, Roxana mi primita linda y mi tío Eugenio quien ha sido un ejemplo a seguir para mí.

A la familia de Lidia quienes son mi familia ahora, a mi cuñado Yuri y su mamá.

A mis amigos durante estos últimos años Héctor mi compañero de cuarto, quien más me rectificó el documento y sé que lo extrañaré cuando salga para Camagüey, a Nael quien es como mi hermano, al genio Leiser quien siempre tiene la respuesta a mis dudas, Gio, Norberto, Hansel y Perú.

A mi brigada Nelson, Yugui, Lozano, Yurelis, Yairilee, el niche, el White, a los matanceros, a los peque y a Pizza.

A Adolfo, José Ernesto, Batista, Sergio, Rubén, Pedro y Yunaime compañeros de SCADA, quienes me aportaron conocimientos necesarios para realizar el trabajo.

Al tribunal por ser tan exigente.

A la universidad, Fidel y la Revolución por brindarme la oportunidad de convertirme en algo útil para la sociedad y desempeñarme en lo que me gusta.

A mis profesores quienes me formaron a lo largo de esta carrera Yidian, Luis Eduardo, Zoraida, Adriana, Villanueva, Pedro, Millet y Prevot.

A todos mis compañeros en general, gracias por su ayuda.

Dedicatoria

Siempre le dedico todos mis triunfos a una persona que está conmigo en todo momento, a quien le debo la educación y la formación que tengo, a un ejemplo de amor incondicional, quien me lo da todo sin pedirme nada a cambio, a la persona que más le alegra mis triunfos y sufre mis derrotas, la que me admira y me eleva sobre todos los demás, a la persona que más amo en esta vida y también le debo mi vida, a mi madre.

Resumen

El proyecto Desarrollo del SCADA GALBA es un trabajo de colaboración entre la Universidad de las Ciencias Informáticas y Petróleos de Venezuela S. A., empresa medular de la economía venezolana. En él se desarrolla el manejador GE Fanuc que forma parte del Sistema de Supervisión, Control y Adquisición de Datos del proyecto.

El objetivo de la presente investigación consiste en desarrollar un simulador que funcionará como servidor en la comunicación con el manejador GE Fanuc. El simulador es necesario debido a la carencia del dispositivo en la Línea de Adquisición del Centro de Informática Industrial perteneciente a la Universidad de las Ciencias Informáticas y al no contar con este, se dificulta la etapa de pruebas del manejador, proceso indispensable para lograr que dicho producto presente la menor cantidad de errores posibles. La investigación comienza con un estudio de los principales conceptos enmarcados dentro del proceso de adquisición y envío de datos, utilizando la arquitectura TCP/IP así como el protocolo GE Fanuc. Se lleva a cabo un estudio de las tecnologías y herramientas de desarrollo afines con este tipo de investigación, para determinar cuáles utilizar en el diseño e implementación del simulador. Se exponen las características funcionales y no funcionales del simulador, así como la arquitectura general que se seleccionó para desarrollar la aplicación con la descripción de cada uno de sus componentes. Se concluye el proceso de desarrollo con la ejecución de las pruebas que garantizan la calidad y el correcto funcionamiento del simulador.

Palabras clave: GE Fanuc, protocolo, QT, SCADA, simulador, TCP/IP, TransportProvider

Índice

Índice de tablas

Índice de figuras

Introducción

En la actualidad, el desarrollo de la informática ha favorecido la automatización de diversos procesos en instituciones u organizaciones y ha provocado una mejoría en el manejo de la información pasando del contexto físico al contexto digital. (1) En el área industrial muchos de los procesos que para su realización era indispensable la presencia del humano actualmente han sido automatizados, como ejemplo se tiene a los Sistemas de Supervisión, Control y Adquisición de Datos (SCADA, por sus siglas en inglés). (2) Un sistema de este tipo permite supervisar y controlar procesos industriales a distancia. Una empresa que utiliza un SCADA es Petróleos de Venezuela, S. A. (PDVSA) cuyas actividades son la explotación, producción, refinación, mercadeo y transporte del petróleo venezolano. Esta empresa sufrió entre diciembre del 2002 y enero del 2003 un sabotaje que implicó la paralización de varias de sus actividades, provocando consecuencias negativas para la economía de ese país. Es por ello que el estado decidió buscar soluciones orientadas a la soberanía tecnológica a través del uso de software libre en las aplicaciones utilizadas. (3)

Fue el Centro de Informática Industrial (CEDIN) de la Universidad de la Ciencias Informáticas (UCI) el responsable de dar solución al desarrollo de un SCADA en conjunto con la empresa PDVSA. Para esto se creó el proyecto Desarrollo del SCADA para el Guardián del Alba (SCADA GALBA) y así lograr la soberanía tecnológica. (4) Este centro tiene como objetivo desarrollar productos y servicios informáticos de automatización industrial con un alto valor agregado y que cumplan con las necesidades y expectativas de los clientes, además de potenciar la formación especializada y la investigación. (5)

SCADA GALBA posee varios módulos o subsistemas y cada uno con sus funcionalidades específicas. Estos subsistemas son: HMI[1], configuración, seguridad, middleware[2], base de datos histórica y adquisición de los datos. (6)

La función del subsistema de adquisición de los datos consiste en la transferencia de información entre el sistema SCADA y el conjunto de dispositivos que se encuentran en el campo. Está formado por diferentes componentes entre los que podemos encontrar los manejadores de dispositivos y el módulo de recolección

[1] **HMI:** siglas en inglés de Human Machine Interface, en español significa interfaz hombre-máquina.

[2] **Middleware:** es la capa que permite la comunicación entre aplicaciones.

de datos. (7)

Los manejadores (drivers en inglés) son programas que permiten al sistema interactuar con un dispositivo, proporcionando al usuario una interfaz para utilizarlo. Estos son el primer eslabón en la captura de información de un SCADA por lo que deben estar libre de errores para asegurar la integridad de los datos y el correcto funcionamiento del sistema. Dado el importante papel que juegan, la línea de Adquisición del proyecto Desarrollo del SCADA GALBA utiliza el dispositivo de campo, para realizarle las pruebas a los manejadores implementados. (8)

La línea de Adquisición del centro implementó la segunda versión del manejador **GE Fanuc**[3] y esta se encuentra en su etapa de pruebas, sin embargo, aún en esta etapa la línea de desarrollo carece del dispositivo de campo necesario para probar su correcto funcionamiento a través de la detección de los errores.

A pesar de la constante insistencia de la dirección del centro, la universidad y la compañía PDVSA no han podido realizar la compra del dispositivo, debido a su elevado precio en el mercado internacional.

Estas deficiencias imposibilitan en gran medida el cumplimiento del ciclo de desarrollo del producto, impide su entrega en tiempo y con la calidad requerida. Además, provoca la insatisfacción del cliente. Es pertinente aclarar que la primera versión del manejador se desarrolló en Venezuela, donde se contó con el dispositivo de campo y se le realizaron las pruebas pertinentes.

La **situación problemática** anteriormente planteada permite formular el siguiente **problema científico**:

¿Cómo probar el funcionamiento del manejador GE Fanuc, implementado en el proyecto Desarrollo del SCADA GALBA?

A partir del problema que se presenta se define como **objeto de estudio**: la simulación de dispositivos industriales, especificándose en el **campo de acción**: la transferencia de información a través del protocolo de comunicación GE Fanuc.

Para dar solución al problema científico se toma como **objetivo general:** Desarrollar una aplicación, basada en tecnologías libres, que simule el dispositivo PLC Series 90-70 para realizar las pruebas al

[3] **GE Fanuc:** protocolo de comunicación que utiliza el manejador implementado para poder intercambiar información con el dispositivo de campo.

manejador GE Fanuc del SCADA GALBA.

La **idea a defender** que guiará este trabajo es: si se desarrolla una aplicación que simule el dispositivo PLC Series 90-70 y logre una correcta comunicación con el manejador, permitirá probar el funcionamiento del manejador GE Fanuc implementado en el proyecto Desarrollo del SCADA GALBA.

Para darle cumplimiento al objetivo general definido, se elaboran las siguientes **tareas de investigación**:

1. Elaborar el marco teórico de la investigación a partir del estudio del estado del arte sobre el desarrollo de simuladores de dispositivos industriales.
2. Caracterizar la arquitectura TCP/IP a partir del estudio de sus protocolos de las capas de transporte y red.
3. Definir las funcionalidades del sistema a partir del estudio de las especificaciones del protocolo GE Fanuc.
4. Definir la arquitectura y las clases para la elaboración del diseño del simulador.
5. Implementar el simulador del dispositivo para la realización de pruebas al manejador GE Fanuc.
6. Diseñar y aplicar las pruebas al simulador implementado para verificar su correcto funcionamiento.

Para darle cumplimiento a las tareas de investigación, fueron utilizados varios métodos científicos, entre los que se destacan como **métodos teóricos:**

- **Analítico-Sintético:** se utilizó para identificar los principales fundamentos teóricos, así como las características fundamentales de la simulación de dispositivos y la comunicación mediante el protocolo GE Fanuc, a través del estudio de documentación recopilada.
- **Modelación:** se utilizó para la elaboración de diagramas que modelan la estructura, relaciones internas y características del objeto de investigación. Estos permitieron crear abstracciones del problema real y del software.

Y como **métodos empíricos**:

- **Experimental:** se empleó en la realización de pruebas, donde se crean las condiciones propicias para verificar el correcto funcionamiento del simulador desarrollado en la presente investigación.
- **Entrevista:** aplicada a especialistas con experiencia en el desarrollo de simuladores y manejadores de dispositivos con el fin de definir las funcionalidades que tendría la aplicación.

Al finalizar la investigación se obtendrá como resultado: un simulador que puede ser utilizado para la realización de pruebas al manejador del protocolo Ge Fanuc. Este brindará la posibilidad de realizar demostraciones de las capacidades del manejador antes mencionado. Además, ayudará en el desarrollo de otras versiones que se realicen del manejador.

El presente documento está estructurado en tres capítulos:

En el **Capítulo 1** se muestran características, conceptos y definiciones de los elementos relacionados con la simulación, los manejadores, dispositivos a simular, así como el análisis de simuladores de dispositivos ya existentes. También se realiza un estudio de los protocolos de la capa de transporte de la arquitectura TCP/IP y las características del protocolo GE Fanuc. Al final del capítulo se expone el estudio realizado acerca de algunas de las metodologías afines con este tipo de investigación.

En el **Capítulo 2** comienza con la selección de las herramientas, tecnologías y metodología empleada en el desarrollo del proyecto. A continuación se describe detalladamente cada una de las funcionalidades que posee el sistema a partir de las historias de usuario, para una mayor claridad de lo que se desea desarrollar. Se mencionan los requisitos no funcionales de gran utilidad para el usuario final, ya que se deben tener en cuenta para la correcta utilización del sistema desarrollado. Además, se describe la arquitectura del sistema y las principales clases que la componen.

En el **Capítulo 3** se le da paso a la implementación del sistema y para describirlo se utilizan las tareas de ingeniería. Una vez que este proceso culmina se realizan las pruebas sobre el producto terminado pruebas unitarias, de aceptación y de estrés.

Capítulo 1 Fundamentación teórica

Introducción

En la actualidad el uso de la simulación es muy común en diferentes áreas, debido a las ventajas que esta proporciona. En el ámbito industrial, particularmente, es de vital importancia ya que permite prevenir eventos indeseables, mediante su apoyo a la toma de decisiones y corrección de errores. En este capítulo se muestran características, conceptos y definiciones de los elementos relacionados con la simulación, el manejador GE Fanuc, el dispositivo a simular, protocolos que intervienen en la comunicación, así como el análisis de simuladores de dispositivos ya existentes.

1.1. Simulación

Existen muchas definiciones de simulación, una de ellas es la planteada por David Himmelblau y Kenneth Bischoff en el libro “Análisis y simulación de procesos”, en la que conciben la simulación como la “representación de un fenómeno a través de modelos, lo que permite analizar sus características con mayor facilidad sin tener que desarrollar el fenómeno, con lo que se ahorra tiempo y recursos, uno de los objetivos primordiales de una simulación es analizar los resultados para así conocer con anterioridad su comportamiento y en caso posible mejorarlos en el momento que se lleve a cabo el fenómeno en la vida real”. (9)

Desde otra perspectiva, la simulación se puede enfocar como un proceso experimental, como proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender su comportamiento o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos para el funcionamiento del sistema. (10)

Ambas definiciones permiten ver la simulación como el proceso donde se lleva un fenómeno real a uno o varios modelos que representen sus características y comportamientos en diversas situaciones. Permite su análisis con el objetivo de mejorar el sistema real que se estudia. Los simuladores constituyen herramientas de gran utilidad en la mejora de la calidad del trabajo del hombre.

Entre las ventajas que ofrece la simulación se destacan las siguientes:

- Es un método de mayor facilidad de aplicación que el analítico.

- Permite estudiar sistemas de los cuales se cuenta con información incompleta y ayuda a generar información complementaria.
- Mejora el conocimiento del proceso actual al permitir que el analista vea cómo se comporta el modelo generado bajo diferentes escenarios.
- Puede utilizarse como medio de capacitación para la toma de decisiones.
- Es más económico realizar un estudio de simulación que hacer muchos cambios en los procesos reales.
- Permite probar varios escenarios en busca de las mejores condiciones de trabajo de los procesos que simula.
- En problemas de gran complejidad, la simulación permite generar una buena solución.
- En la actualidad los paquetes de software para simulación tienden a ser más sencillos, lo que facilita su aplicación. (11)

Entre las desventajas que se pueden llegar a presentar en la simulación están:

- La simulación puede ser costosa cuando se quiere emplear en problemas relativamente sencillos de resolver, en lugar de utilizar soluciones analíticas que se hayan desarrollado de manera específica para ese tipo de casos.
- Se requiere bastante tiempo – generalmente meses – para realizar un buen estudio de simulación; por desgracia no todos los analistas tienen la disposición (o la oportunidad) de esperar ese tiempo para obtener una respuesta.
- Es preciso que el analista domine el uso del paquete de simulación y que tenga sólidos conocimientos de estadística para interpretar los resultados. (11)

En el desarrollo de la solución propuesta, la simulación aporta una herramienta informática que se utiliza para la realización de las pruebas al manejador GE Fanuc.

1.2. Otros simuladores de dispositivos

Se analizaron varias herramientas con el objetivo de determinar si las mismas reúnen las condiciones necesarias para ayudar en el proceso de prueba del manejador GE Fanuc. A continuación se describen algunos de estos productos y el por qué no se pudieron utilizar.

Simulador para realizar las pruebas del manejador Modbus Omni: desarrollado por José Carlos Abraham Ravelo, está capacitado para recolectar, interpretar y responder las solicitudes que un maestro le

envíe a través del protocolo Modbus Omni. Este simulador brinda la opción de simular redes seriales, o sea, en el sistema se podrá crear más de un esclavo, cada uno con su propio canal de comunicación. Además el sistema muestra la información referente a cada canal de comunicación y cada esclavo, estos datos serán actualizados cada segundo. También emula cambios complejos en la comunicación, por tanto el simulador brindará la opción de generar cuatro tipos de eventos capaces de alterar dicha comunicación. Este simulador posee eventos que pueden ocurrir en la comunicación con el dispositivo real, pero no se puede utilizar para dar solución a nuestro problema ya que se comunica por el protocolo Modbus Omni, no por el GE Fanuc. (12)

Simulador de dispositivos PLC-5 para realizar el proceso de pruebas al manejador DF1 del SCADA SAINUX: desarrollado por Miguel Ángel Albuerne Rivero, este simula el comportamiento en la transferencia de datos del dispositivo PLC-5, pero al igual que el anterior no utiliza el protocolo GE Fanuc, sino el protocolo full dúplex DF1. (13)

Simulador para el protocolo de comunicación DNP3 en su variante serial: desarrollado por Sheyla García Quiñonez, es capaz de recibir, interpretar y enviar diferentes mensajes al manejador DNP3. No se puede utilizar para dar solución al problema ya que el protocolo no es GE Fanuc, sino DNP3 y su comunicación es por serial solamente no se comunica mediante TCP/IP. (14)

MODSIM: Esta herramienta (Fig. 1) está concebida para comportarse como un esclavo dentro de una red de comunicación Modbus, no GE Fanuc. Además, no permite recrear eventos. Por tanto, se desechó la opción de utilizarlo. (12)

Fig. 1 Barra de herramientas del MODSIM.

1.3. Modelo Maestro/Esclavo

En el modelo Maestro/Esclavo (Fig. 2), conocido también como jefe/trabajador, una entidad maestra recibe una o más peticiones de trabajo y esta crea entidades esclavas para que la ejecuten. Normalmente el maestro controla a los esclavos y lo que hacen. Un esclavo ejecuta su tarea de forma independiente a los demás esclavos. El modelo soporta un ambiente distribuido, en el cual los requerimientos de servicio hechos

por estaciones de trabajo inteligentes o maestros resultan en un trabajo realizado por otros computadores llamados esclavos. (15) Los esclavos pueden proveer de múltiples servicios a los maestros tales como impresión, acceso a bases de datos, fax y procesamiento de imágenes. (12)

Fig. 2 Modelo Maestro/Esclavo

1.4. Comunicación mediante el protocolo TCP/IP

El protocolo de internet (IP por sus siglas en inglés) es un protocolo que trabaja a nivel de red donde la información se envía en paquetes de datos o datagramas IP (Tabla 1). Este protocolo ofrece un servicio “sin garantías” también llamado “mejor esfuerzo”, es decir, que nada garantiza que los paquetes lleguen al destino, sin embargo se hará lo posible por hacerlos llegar. (16)

El formato de un paquete IP está diseñado para llevar información que permita diseccionarlo a su destino y obviamente que permita volver a ensamblar los paquetes en el destino para recuperar la información útil. Estos pueden tomar diferentes caminos para llegar al destino, o sea, que hay redundancia de caminos y es menos probable que todos se pierdan. El protocolo IP procesa datagramas de manera independiente al definir su representación, ruta y envío. (16)

Tabla 1 Datagrama del protocolo IP

Primer octeto								Segundo octeto								Tercer octeto								Cuarto octeto							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Versión				Longitud				Tipo de servicio								Longitud total															

Identificador		Flags[4]	Posición de los datos
Tiempo de vida	Protocolo	Suma de control de cabecera	
Dirección de origen			
Dirección de destino			
Opciones			Relleno
Datos			

Por otra parte el Protocolo de Control de Transmisión (TCP por sus siglas en inglés) es un protocolo de transporte orientado a conexión. Esto hace que los datos se entreguen sin errores, sin omisión y en secuencia. Este tiene las siguientes características:

- **Orientado a conexión:** es decir, las aplicaciones solicitan la conexión al destino y luego usan esta conexión para entregar los datos, garantizando que estos serán entregados sin problemas.
- **Punto a punto:** una conexión TCP tiene dos extremos, emisor y receptor.
- **Confiabilidad:** garantiza que los datos transferidos serán entregados sin ninguna pérdida, duplicación o errores de transmisión.
- **Full dúplex:** los extremos que participan en una conexión TCP pueden intercambiar datos en ambas direcciones simultáneamente.
- **Conexión de inicio confiable:** el uso de three-way handshake[5] garantiza una conexión confiable y sincronizada de inicio entre los dos extremos de la conexión.
- **Conexión de finalización aceptable:** garantiza la entrega de todos los datos antes de la finalización de la conexión. (17)

TCP es un protocolo que trabaja a nivel de transporte por eso necesita valerse de IP para el envío de sus segmentos o mensajes. De esta manera IP trata el mensaje TCP como la información que debe entregar y en ningún momento intenta interpretar su contenido. Debido a eso, cualquier dispositivo de nivel de red solo puede observar los encabezados IP para el reenvío de los datagramas. El encargado de interpretar los

[4] **Flags:** banderas que se compone de tres bits, el primero reservado a 0, el segundo significa no fragmentar y el tercero que hay más fragmentos.

[5] **Three-way handshake:** tres pasos que utiliza TCP para iniciar una conexión, establecimiento de la conexión, transferencia de datos y por último fin de la conexión.

mensajes TCP, después de recibirlos de la capa de red, es el TCP de la máquina de destino. (17)

1.5. Manejadores de dispositivos

Se entiende como manejador (drivers en inglés) a un software, o en algunos casos componentes hardware, encargado del control de un dispositivo o una familia de ellos. Permite la comunicación entre un sistema y el hardware de un periférico. El manejador oculta al sistema las particularidades del hardware, permitiendo que este pueda acceder al dispositivo a través de una interfaz estandarizada. Después de que cada operación realizada por el mismo se haya completado, este debe verificar si hubo errores. (7)

Los manejadores de dispositivos constituyen piezas esenciales de los sistemas computarizados, pues sin ellos no se podría usar el hardware. Se puede afirmar que existen tantos tipos de controladores como tipos de periféricos, y con mucha frecuencia se puede encontrar más de un manejador para un mismo dispositivo, donde cada uno ofrece un nivel distinto de funcionalidades. En el caso de los sistemas SCADA, utilizan los manejadores para comunicarse con los dispositivos de campo e intercambiar información con ellos, de esta manera el sistema puede manejar la información que proporcionan los equipos. (2)

Las especificaciones de protocolos no solo tratan sobre la semántica de las tramas que se intercambian entre nodos de una red, sino también de los procedimientos para el uso de los mensajes. El tratamiento de los errores, los estados por los que puede transitar la comunicación, así como el modo de funcionamiento son aspectos que se deben tener presentes a la hora de implementar un protocolo determinado. (18)

La capa de protocolo dentro de la arquitectura de los manejadores constituye el elemento lógico que representa la comunicación con los dispositivos de campo. Una de las funcionalidades principales de un manejador es la de regir la mecánica de funcionamiento de los protocolos, haciendo uso de los elementos necesarios para trabajar con la semántica de los mensajes, básicamente en cuanto al ensamblado y desensamblado de los mismos y utilizando las funcionalidades que brinda la capa de transporte para el envío y la recepción de tramas a través del medio físico correspondiente. (18)

1.5.1 Manejador GE Fanuc

El manejador para el protocolo GE Fanuc permite las comunicaciones entre el sistema SCADA GALBA y los PLCs GE Fanuc Series 90-70 (Modelo tarjeta Ethernet- IC697CMM742). Fue desarrollado tomando como base la biblioteca DriverCore. Para el envío de las tramas a través del medio físico se hace uso de la biblioteca de transporte TransportProvider. (19)

Como parte de los orígenes del protocolo GE Fanuc, se puede destacar que en 1993 el protocolo SNP-X (Protocolo para la Serie Noventa) fue agregado a los módulos COMM CMM311/CMM711 para acelerar las transferencias de memoria del PLC para aplicaciones multipunto. Este protocolo es una extensión del SNP es simple de usar y provee significantes mejoras con respecto a su antecesor. No soporta la programación o la configuración de operaciones en el PLC. (20)

1.5.2 Características fundamentales del manejador GE Fanuc

El manejador GE Fanuc soporta la lectura y escritura de variables analógicas y digitales. Un ejemplo de dirección válida para una variable digital es: M1066; siempre que comiencen con la letra M el manejador las ubica en el segmento 76. En el caso de las variables analógicas podemos mencionar que un ejemplo válido de dirección es: R1025; siempre que comience con la letra R el manejador las ubica en el segmento 8. (21)

Un bloque es la abstracción del conjunto de variables que tienen igual frecuencia de muestreo y que pueden ser recuperadas en una sola petición al PLC. Las variables digitales se ubican en bloques de como máximo 8 bits. Y las variables analógicas se ubican en bloques hasta un máximo de 256 bytes. (21)

El manejador GE Fanuc también posee las funcionalidades necesarias para establecer la comunicación con dispositivos que utilicen el protocolo GE Fanuc. Permite realizar lecturas y escrituras de variables de forma asíncrona, además de poder obtener y configurar algunas propiedades de los dispositivos. Las llamadas a las funciones de lectura o escritura asíncronas retornan inmediatamente. Luego, a través del handler[6], se informará en forma de callback[7] por el hilo del provider el resultado de la operación. (22)

1.6. Controladores Lógicos Programables

Para un mayor entendimiento de los dispositivos que se van a simular se realizó un análisis de las características fundamentales de los Controladores Lógicos Programables (PLC por sus siglas en inglés).

---

[6] **Handler:** un método ejecutado sobre un hilo independiente a partir del retorno de otro método al cual está asociado.

[7] **Callback:** forma en que se manda a ejecutar el handler.

Las industrias automatizadas deben lograr en sus sistemas, alta confiabilidad, eficiencia y flexibilidad. Para lograr estas metas deben poseer como elemento fundamental un conjunto de dispositivos electrónicos denominados PLC. Estos dispositivos son conocidos como máquinas electrónicas diseñadas para controlar en tiempo real y en el medio industrial, procesos secuenciales de control. (13)

Los PLC ofrecen diversas ventajas en procesos de automatización, puesto que cuentan con las herramientas adecuadas para realizar su desempeño, tales como: relés, temporizadores electrónicos, contadores y controles mecánicos como el tipo tambor. Además, dispone facilidades de comunicación para acceder a subsistemas de comunicaciones. (13)

Por sus especiales características de diseño, estos dispositivos tienen un campo de aplicación muy extenso, y la constante evolución del hardware y software amplían continuamente este campo.

1.6.1 Componentes del PLC

El PLC se compone esencialmente de algunas partes comunes a todos los modelos, y otras que dependen de la envergadura del mismo y la aplicación donde se usará. Los componentes comunes son: (13)

- **Fuente de alimentación**: brinda energía a todo el conjunto de módulos y también puede alimentar a los dispositivos de entrada. La tensión de alimentación puede variar entre 5 y 24 voltios.
- **CPU (Unidad central)**: procesa toda la información recibida del exterior y activa la señal de las salidas una vez procesada la programación. Para esto cuenta con los siguientes elementos: procesador, memoria y comunicación.
- **Módulo de entradas** (discretas y analógicas): permite la conexión de los sensores y detectores que controlan el proceso.
- **Módulo de salidas** (discretas y analógicas): permite la conexión de los actuadores del sistema y recibe la información de la CPU mediante bus interno.
- **Módulos especiales**: Estos se pueden acoplar al bus. (módulo de Entradas/Salidas (E/S) analógica).

1.6.2 Características de la comunicación con el PLC.

El flujo normal de eventos del proceso de comunicación entre los dispositivos de campo (PLC) y un sistema que acceda sus servicios se rige por el modelo maestro/esclavo. El objetivo de este tipo de comunicación es que un operador central pueda controlar diversas actividades, cada una de ellas será

controlada por un dispositivo de campo, el cual, para los efectos prácticos el dispositivo será tratado como un esclavo y el sistema instalado en el puesto de trabajo del operador cumplirá la función del maestro. Al iniciar el sistema en el maestro se configurarán todas las direcciones de los esclavos a quien este controlará. Una vez configurado el maestro, el operador podrá enviar mensajes solicitando la lectura o escritura de los registros de un esclavo. Ambas partes podrán estar comunicándose vía Ethernet o serial. Los mensajes enviados, tanto por el dispositivo maestro como por los dispositivos esclavos, serán concebidos según las especificaciones del protocolo de comunicación que ambos utilizan. O sea, los mensajes enviados por el manejador GE Fanuc y las respuestas de los dispositivos de campo, serán mensajes GE Fanuc TCP/IP descritas en la sección 1.4. (13)

1.6.3 PLC GE Fanuc Series 90-30 y 90-70

El PLC GE Fanuc Series 90-30 (Fig. 3) es una familia de controladores y sistemas de E/S diseñados para satisfacer la demanda mundial de controladores industriales, que sean efectivos a nivel de entrada. Con su arquitectura innovadora, las Series 90-30 abren el camino a las soluciones eficaces en el costo de muchas aplicaciones PLC. La CPU está situada en la placa base, así reduce el costo y aumenta la capacidad de disponibilidad para tarjetas de E/S de alta densidad. (23)

Fig. 3 PLC GE Fanuc Series 90-30.

El PLC Series 90-30 ha sido instalado en diversas aplicaciones tales como: sistemas empaquetadores de alta velocidad, planta de tratamiento de agua, monitoreo de emisiones continuas, planta de procesamiento de alimentos entre otras. La capacidad de comunicación de los PLC 90-30 abarca los siguientes protocolos: Bus Genius, Ethernet TCP/IP, World FIP, Profibus-DP, DeviceNet, SDS, LonWorks e interbus-S. Poseen una capacidad de memoria de programa de hasta 240 Kbytes y pueden manejar un máximo de 4096 puntos I/O discretas. Respecto a las señales analógicas el máximo es de 2048 entradas y 512 salidas. (24)

Fácil integración:

- El PLC de la Serie 90-30 es de bajo costo pero de altas prestaciones. Como resultado de ello, puede integrarse fácilmente en aplicaciones que van desde la simple sustitución de un relé hasta la automación sofisticada de rango intermedio.
- La CPU proporciona potentes prestaciones tales como PID interno, programación estructurada, control de interrupción, direccionamiento indirecto y una gran variedad de bloques funcionales que ejecutan operaciones complicadas.
- Otras opciones de la Serie 90-30 incluyen potentes módulos de posicionamiento de ejes, módulos de contadores de alta velocidad, módulos del coprocesador programable en lenguajes BASIC TM y “C” y módulos de comunicaciones serie Genius y Ethernet.
- La amplia línea de sistemas de diagnóstico de E/S y CPU de GE Fanuc, simplifica la puesta en marcha y la localización de averías además de su fácil integración con otro PLC y ordenadores.

Compatibilidad:

- Con la idea de proporcionar soluciones en corto espacio de tiempo a las necesidades de los clientes, GE Fanuc trabaja en estrecha colaboración con otros fabricantes de periféricos y paquetes de software. Los resultados de esta colaboración expanden aún más las capacidades de la Serie 90-30. Otros productos compatibles incluyen termopares, RTDS, módulos de motor paso a paso y relés de alto amperaje.
- Diversos proveedores de software de adquisición de datos y control han desarrollado productos que permiten la utilización de la Serie 90-30 con ordenadores personales. Estos paquetes proporcionan una total integración entre su software, el PLC de la Serie 90-30 y la E/S. Hay muchas interfaces de operador que utilizan el protocolo SNP para comunicarse con la Serie 90-30 a través de su puerto incorporado.
- Comunicación Ethernet TCP/IP, Genius LAN y Serie son integrables, así como la utilización de otros sistemas de comunicación basados en arquitecturas abiertas con módulos VME. (23)

Los PLC GE Fanuc Series 90-70 (Fig. 4) son una versión más avanzada que los 90-30 y han sido diseñados para aplicaciones complejas, tales como: sistemas modulares triples redundantes, procesamiento de alta velocidad, y aplicaciones que requieren un gran número de I/O y memoria. La Serie 90-70 puede manejar aplicaciones que requieren un máximo de 12000 I/O y hasta 6MB de memoria. La Serie 90-70 posee un módulo Ethernet TCP/IP que soporta la transferencia global de datos o *Ethernet Global Data* (EGD) desde un dispositivo a otro en la red. (24)

Fig. 4 PLC GE Fanuc Series 90-70.

El PLC GE Fanuc Series 90-70 (Modelo tarjeta Ethernet IC697CMM742) es el dispositivo a simular.

1.7. Metodologías de desarrollo de software

Las metodologías de desarrollo consisten en una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. Constituye una guía que nos indica cómo actuar y qué hacer para lograr el resultado esperado en una investigación. La metodología que se utilizará debe estar en correspondencia con las características que tenga el proyecto que se pretende desarrollar. La finalidad de una metodología de desarrollo es garantizar la eficacia (cumplir los requisitos iniciales) y la eficiencia (minimizar las pérdidas de tiempo) en el proceso de generación de software. En los últimos años se han desarrollado dos corrientes fundamentales en lo referente a los procesos de desarrollo, las metodologías ágiles y las tradicionales. La diferencia fundamental entre ambos tipos de metodologías consiste en que las tradicionales pretenden lograr sus objetivos mediante el orden y la documentación, mientras que las ágiles intentan mejorar la calidad del software mediante la comunicación directa e inmediata entre las personas que intervienen en el proceso. (25)

1.7.1 Proceso Unificado de Desarrollo (RUP)

Es una metodología de desarrollo de software que está basada en componentes e interfaces bien definidas, y junto con el UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es un proceso que puede especializarse para una gran variedad de sistemas de software, en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

En su modelación define como sus principales elementos:

- **Trabajadores (“quién”):** define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- **Actividades (“cómo”):** es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- **Artefactos (“qué”):** productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- **Flujo de actividades (“cuándo”):** secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

Por sus características se clasifica como un método pesado, pues requiere de un grupo grande de programadores para su uso, además de que un cambio en las etapas de vida del sistema incrementaría notablemente el costo. RUP define un total de 4 fases. En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto) y dentro de cada una de ellas seguirá un modelo de cascada para los flujos de trabajo que requieren las nuevas actividades anteriormente citadas. Las fases son:

1. Inicio (puesta en marcha)
2. Elaboración (definición, análisis, diseño)
3. Construcción (implementación)
4. Transición (fin del proyecto y puesta en producción)

1.7.2 Metodología eXtreme Programming (XP)

Desarrollada por Kent Beck, la metodología XP se clasifica como ágil. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto. Está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo y propiciando un buen clima de trabajo. XP se basa en realimentación continua y comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. Intenta reducir la complejidad del software por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

El ciclo de vida ideal de XP consiste en las siguientes fases:

1. Exploración.
2. Planificación de la Entrega (Release).
3. Iteraciones.
4. Producción.
5. Mantenimiento.
6. Muerte del Proyecto.

1.7.3 Metodología SCRUM-XP (SXP)

SXP es una metodología ágil de desarrollo de software. Está compuesta por las metodologías SCRUM y XP que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo. Consta de 4 fases principales:

1. **Planificación-Definición:** donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
2. **Desarrollo:** es donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
3. **Entrega:** puesta en marcha.
4. **Mantenimiento:** donde se realiza el soporte para el cliente.

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario, diseño, implementación, pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, lo que nos permite mejorar el diseño cada vez que se le añada una nueva funcionalidad. SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad.

Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes pueden ver día a día cómo progresa el trabajo.

1.7.4 Proceso Unificado Ágil (AUP) con el modelo CMMI-DEV v 1.3.

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (test driven development - TDD en inglés).
- Modelado ágil.
- Gestión de Cambios ágil.
- Refactorización de Base de Datos para mejorar la productividad.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva:

1. **Inicio:** El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. **Elaboración:** El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
3. **Construcción:** Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
4. **Transición:** El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

Conclusiones parciales

En este capítulo se logró una mejor claridad de la investigación, abordando diversos conceptos asociados al problema como son la simulación, el manejador que se desea probar y el dispositivo que se simula. A partir del estudio de los simuladores existentes resultó que estos no pueden dar solución al problema ya que varían los protocolos de comunicación.

Capítulo 2 Diseño de la solución propuesta

Introducción

El objetivo de este capítulo es diseñar el sistema propuesto como solución. Comienza con la selección de las herramientas, tecnologías, así como la selección de la metodología que guía el proceso de desarrollo de la solución. Se explica mediante una descripción detallada cada una de las funcionalidades que debe poseer el sistema a partir de las historias de usuario para una mayor claridad de lo que se desea desarrollar. Se describen también los requisitos no funcionales con gran utilidad para el usuario final ya que debe tenerlos en cuenta para la utilización correcta del sistema desarrollado. Además, se brinda la propuesta de arquitectura y se describen las principales clases que le componen todo esto para ayudar al desarrollo de la aplicación y una correcta estructuración del código.

2.1. Herramientas seleccionadas para el desarrollo del simulador

Para el desarrollo del simulador que se propone como solución al problema planteado se utilizaron tecnologías probadas por la línea de Adquisición del CEDIN. Entre las que se encuentran las que se describen a continuación.

2.1.1 Biblioteca de comunicación TransportProvider 2.0

La interacción entre los manejadores y los dispositivos se efectúa mediante el envío y recepción de mensajes a través de un medio físico determinado, por tanto, uno de los primeros retos a la hora de tratar de desarrollar algún tipo de controlador es la implementación de un transporte, que no es más que un conjunto de funcionalidades que permiten la abstracción en el intercambio de información entre diferentes nodos dentro de una red. La Capa de Transporte es la encargada de brindar los servicios para el envío y la recepción de tramas mediante el medio físico Ethernet bajo la semántica de los protocolos TCP/IP. La biblioteca TransportProvider2 (desarrollada en la línea), fue utilizada para garantizar el transporte de datos, por su extensibilidad, flexibilidad y poseer una interfaz intuitiva. (18)

2.1.2 Marco de trabajo QT 5.3.2

Qt es una plataforma de desarrollo que incluye clases, bibliotecas y herramientas para la producción de aplicaciones de interfaz gráfica en C++ entre otras funcionalidades. Está distribuida bajo los términos de

GNU Lesser General Public License, es software libre y de código abierto. Qt es multiplataforma e incluye soporte de tecnologías como OpenGL, XML, bases de datos y programación para redes. QT dispone de una amplia gama de herramientas que facilitan entre otras cosas la creación de formularios, botones y ventanas de diálogo con el uso del ratón. Además provee soporte para la internacionalización de su contenido. (26)

Este framework fue de gran utilidad para la visualización y manejo de la memoria del dispositivo (Fig. 5) y las tramas intercambiadas en la comunicación.

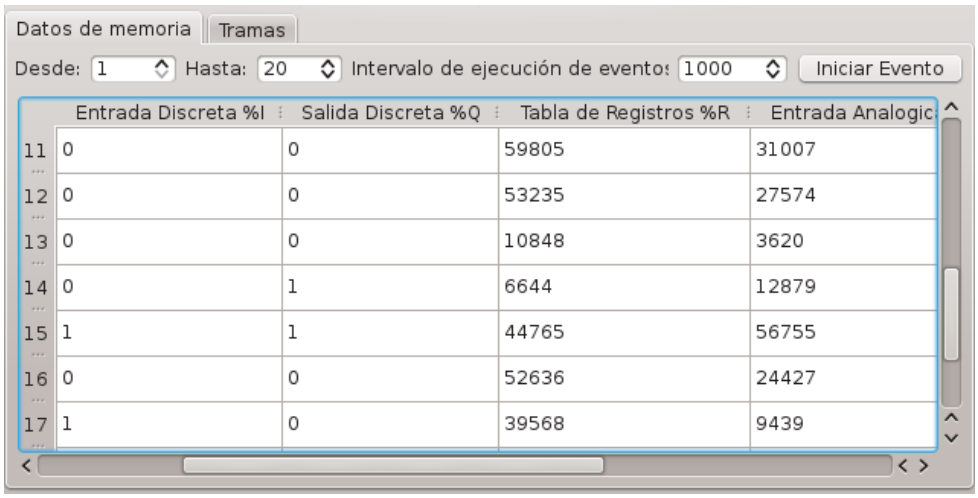


Fig. 5 Visualización de la memoria del dispositivo.

2.1.3 IDE de desarrollo QT Creator 3.2.1

Qt Creator, un entorno de desarrollo (IDE) multiplataforma muy completo. Sus principales características son:

- Posee un avanzado editor de código C++.
- Además soporta los lenguajes: C#/.NET Languages (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
- Posee también una GUI integrada y diseñador de formularios.
- Herramienta para proyectos y administración.
- Ayuda sensible al contexto integrado.
- Depurador visual.
- Resaltado y auto-completado de código.

- Soporte para refactorización de código. (27)

Fue de gran ayuda para agilizar el desarrollo gracias a su completamiento, generación de código, búsquedas avanzadas y remplazamiento de frases.

2.1.4 Lenguaje de programación C++ 11

Es un lenguaje imperativo orientado a objetos derivado del C. En realidad, un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. Este lenguaje tiene una alta potencia para la programación a bajo nivel. Se la han añadido elementos que le permite un estilo de programación con alto nivel de abstracción. Entre las grandes ventajas de C++ se pueden mencionar la variedad tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones inline, sobrecarga de operadores, referencias y operadores para manejo de memoria persistente. (28)

Es orientado a objetos, lo que permite estructurar mejor las ideas y encapsular los conceptos de la realidad en clases.

2.1.5 Lenguaje de modelado Unified Modeling Language 2.0

El Lenguaje de Modelado Unificado (UML por sus siglas en inglés), es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funcionalidades, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Permite modelar sistemas utilizando conceptos orientados a objetos y crear un lenguaje de modelado que puede ser utilizado tanto por humanos como por máquinas. Es un lenguaje de modelado para especificar o para describir métodos o procesos. (29)

2.1.6 Herramienta de modelado Visual Paradigm 8.0

La herramienta CASE utilizada fue el Visual Paradigm para el modelado de la solución. La decisión está basada en las facilidades que brinda esta aplicación de representar los diagramas de clases, generar código desde diagramas y diagramas desde el código, facilita el modelado de UML, ya que proporciona herramientas específicas para ello. También permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta. Controla que el modelado con UML sea correcto. Facilita la reutilización, ya que es una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos. Permite generar código de forma automática, reduce los tiempos de desarrollo y evita

errores en la codificación del software. Además, es un software multiplataforma (30). Gracias a esta herramienta se logró modelar los diagramas del sistema de manera ágil.

2.2. Selección de la metodología de desarrollo de software

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo y recursos), se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. (31)

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello nos apoyaremos en el Modelo CMMI-DEV v1.3, el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. (31)

Entre las técnicas ágiles que utiliza AUP se encuentra el Modelado ágil. Se hará uso de esta técnica para los proyectos que necesiten por sus características encapsular sus requisitos funcionales en Historias de usuarios o en Descripción de requisitos por procesos. La otra forma de encapsular los requisitos se mantiene por Casos de Uso (CU). De forma general se siguen utilizando los productos de trabajos definidos en el Expediente de proyecto, algunos son obligatorios independientemente del tipo de proyecto y otros son opcionales basándose en las particularidades del mismo.

Considerando las características de las funcionalidades a implementar y teniendo en cuenta que se necesita obtener un software en poco tiempo y que a la vez sea confiable se analizaron metodologías de desarrollo del software ágiles y robustas. AUP es una metodología flexible que no requiere de una gran cantidad de desarrolladores. Por las características que posee y para favorecer a la estandarización de una metodología en la universidad es seleccionada la metodología de desarrollo de software AUP-UCI.

2.3. Funcionalidades del sistema

A través de los requisitos funcionales, se puede expresar una especificación más detallada de las responsabilidades del sistema. Con ellos, se pretende determinar de manera clara y concisa lo que debe hacer el sistema siguiendo un enfoque funcional. Como nos permite la metodología AUP-UCI, las funcionalidades del sistema serán descritas a través de historias de usuario generadas.

A continuación se describen las principales funcionalidades del sistema:

- **Configurar el canal de comunicación:** El estado inicial del dispositivo es apagado (Fig. 6), para encenderlo este debe ser primeramente configurado. Para poder configurar el canal de comunicación del dispositivo es necesario introducir el puerto IP por donde se aceptará la conexión remota, la cual tendrá un tamaño máximo de 5 caracteres y solo permite números (Fig. 7). Luego de seleccionar el puerto y presionar en el botón de encender, el sistema comprueba si el puerto no está siendo utilizado por otro proceso. En caso de estar libre el puerto seleccionado, el dispositivo inicia con esa configuración cambiando a su estado de encendido (Fig. 8). En caso de ya estar ocupado el puerto por otro proceso, el sistema lo notificará emitiendo un mensaje de error (Fig. 9).

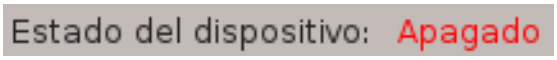


Fig. 6 Dispositivo apagado.

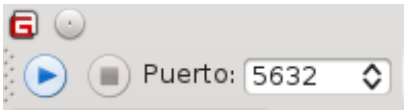


Fig. 7 Configuración del canal de comunicación.

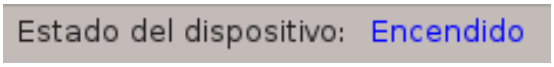


Fig. 8 Dispositivo encendido.

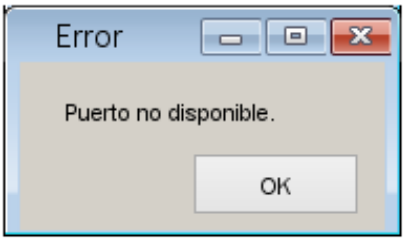


Fig. 9 Mensaje de error.

- **Mostrar contenido de la memoria:** La memoria del dispositivo consiste en seis listas de números, las cuales son visualizadas por columnas. Las tres primeras son de valores binarios y muestran el valor de las entradas y salidas digitales, denotadas como %I y %Q respectivamente, así como valores digitales

internos %M. Las otras tres son de valores enteros sin signo, oscilan entre 0 y 65535. Estas muestran el valor de las entradas y salidas analógicas denotadas por %AI y %AQ respectivamente, así como los valores de la tabla de registros del dispositivo %R (Fig. 10). Dado que la memoria posee muchos sectores, el usuario debe especificar la sección que desea observar de esta seleccionando un rango (Fig. 11).

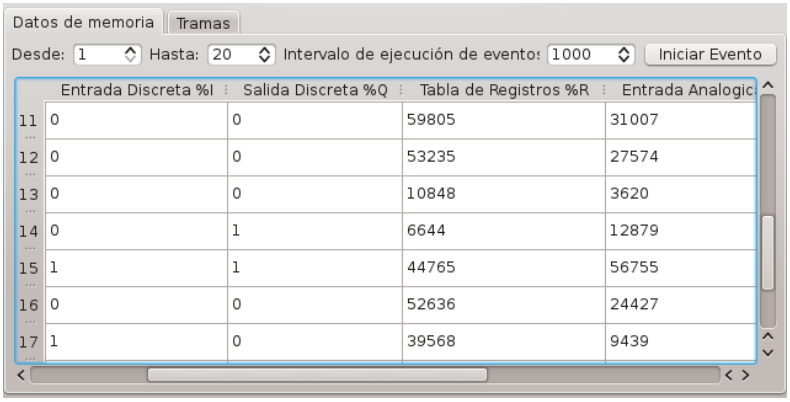


Fig. 10 Memoria del dispositivo.

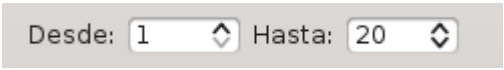


Fig. 11 Sección de memoria a visualizar.

- **Simular valores de memoria:** para modificar el contenido de la memoria del dispositivo, luego de ser adicionada la sesión de registro, se debe de seleccionar el/los valor(es) que se desee(n) modificar. Para ello se presiona clic derecho y posteriormente un clic en la opción “Eventos”. Todos los datos que se muestran en la ventada de diálogo “Eventos” pueden ser modificados (Fig. 12). Además, se pueden iniciar y detener estos eventos, así como definir cada qué tiempo los valores serán actualizados (Fig. 13).

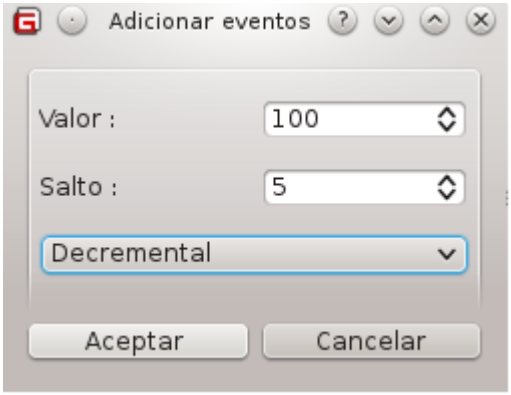


Fig. 12 Simular decremento de valores de la memoria.

Fig. 13 Tiempo de actualización de valores.

- **Mostrar los eventos ocurridos en la comunicación:** el establecimiento de la comunicación comienza con una petición que le realiza el manejador al simulador. Al recibir esta petición el simulador responde aceptando la conexión y espera próximas peticiones. Los eventos de conexión, mensaje recibido, mensaje enviado y desconexión son mostrados por la aplicación (Fig. 14). Además, se muestra el protocolo, nombre del dispositivo y el momento ocurrido. Las peticiones pueden ser de lectura, escritura y desconexión.

Fig. 14 Eventos ocurridos en la comunicación.

- **Mostrar las tramas enviadas y recibidas:** para mostrar las tramas enviadas y recibidas luego de haberse configurado el puerto IP remoto, los eventos en las sesiones y los tipos de registros, además de haberse conectado el protocolo, se deben visualizar las tramas GE Fanuc que se transmiten entre el manejador y el dispositivo en valores hexadecimales (Fig. 15).

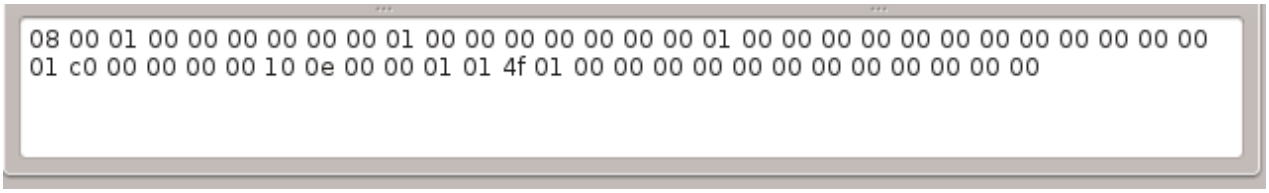


Fig. 15 Trama transmitida en valores hexadecimales.

- **Simular errores en la comunicación:** para simular ruido en el canal el usuario puede marcar y desmarcar esta opción que se encontrará en la barra de herramientas (Fig. 16). Esta funcionalidad realizará modificaciones en las tramas enviadas al manejador con el objetivo de analizar la capacidad del manejador al existir errores en la comunicación. Es necesario que haya transmisión para que sea efectiva esta funcionalidad.

Fig. 16 Simular errores en la comunicación.

2.4. Requisitos no funcionales del sistema

En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto debido a que forman una parte significativa de la especificación.

Requisitos no funcionales de software:

- Instalada la biblioteca TransportProvider.
- Framework QT.
- PC con Sistema Operativo GNU/Linux.

Requisitos no funcionales de hardware:

- PC con microprocesador Pentium 4 o superior.

- PC con memoria RAM mínima de 256 MB.

2.5. Arquitectura del sistema

Definir una buena arquitectura ofrece los beneficios de tomar decisiones tempranas. La arquitectura es una vista estructural de alto nivel, que define estilo o combinación de estilos para una solución. La arquitectura es esencial para éxito o fracaso de un proyecto. Además, es necesaria para comprender el sistema, organizar el desarrollo del mismo, fomentar la reutilización y controlar la evolución del proyecto.

2.5.1 Arquitectura de n-capas

Dada las características del sistema deseado se seleccionó la arquitectura n-capas dado que esta arquitectura establece un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño. Ya que el sistema posee gran interacción con el usuario, para que este inicie, configure, detenga y a su vez visualice las tramas que intervienen en la comunicación con el manejador, se definen tres capas: transporte, protocolo y aplicación (Fig. 17). En la capa de transporte se utilizó la biblioteca TransportProvider con el objetivo de permitir a la aplicación comunicarse a través de una red. En la capa de protocolo se implementó la lógica del sistema y la semántica de las tramas, según las especificaciones de GE Fanuc. Por último, la capa de aplicación tiene la responsabilidad de servir como intermediario entre el sistema y el usuario.

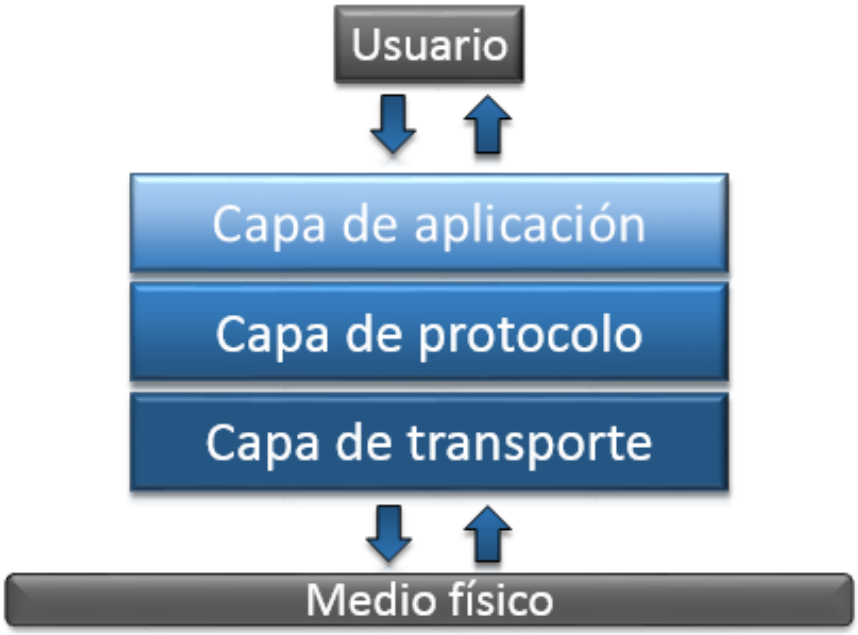


Fig. 17 Arquitectura del sistema

Ventajas:

- Reutilización de capas.
- Facilita la estandarización.
- Dependencias se limitan a intra-capa.
- Contención de cambios a una o pocas capas.

Desventajas:

- A veces no se logra la contención del cambio y se requiere una cascada de cambios en varias capas.
- Pérdida de eficiencia.
- Trabajo innecesario por parte de capas más internas o redundante entre varias capas.
- Dificultad de diseñar correctamente la granularidad de las capas.

2.6. Modelo de dominio

El modelo del dominio explica cuáles son y cómo se relacionan los conceptos relevantes en la descripción del problema, también conocido como modelo conceptual. El modelo está compuesto por tres elementos que se describen a continuación. (Fig. 18)

Manejador GE Fanuc: desarrollado en la línea de adquisición del proyecto Desarrollo del SCADA GALBA y se encarga de brindarle al sistema comunicación con el conjunto de dispositivos que se encuentran en el campo.

Simulador GE Fanuc: la aplicación desarrollada para realizar las pruebas del manejador GE Fanuc dada la no existencia del dispositivo PLC GE Fanuc Series 90-70.

Protocolo GE Fanuc: protocolo utilizado en la comunicación entre el manejador y el simulador. Además, el simulador lo implementa para comunicarse con el manejador.

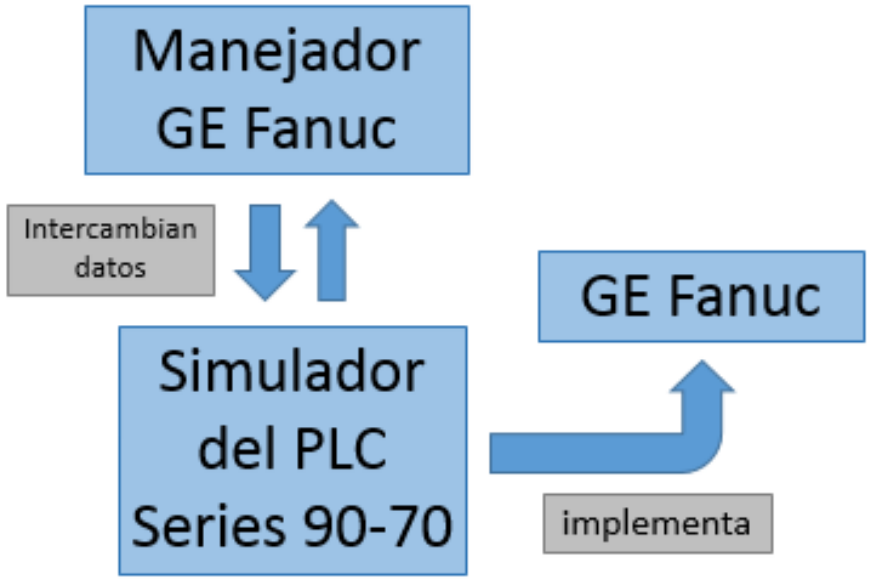


Fig. 18 Modelo de dominio del sistema

2.7. Diagrama de clases del sistema

Los diagramas de clases muestran cómo se compone un sistema. Se dice son diagramas «estáticos» porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: qué clases «conocen» a qué otras o cuales «son parte» de otras, pero no muestran los métodos mediante los que se invocan entre ellas.

2.7.1 Clases de la capa de transporte

La capa de transporte se encarga del intercambio de información entre el medio físico y la capa de

protocolo. Para ello se utilizaron funcionalidades de la biblioteca TransportProvider. Esta herramienta brinda una serie de interfaces las cuales el programador puede utilizar para acceder a sus funcionalidades. La información es recibida en forma de callback a través de métodos específicos proporcionados por la biblioteca; estos métodos pueden o no ser reimplementados por el programador, dependiendo si se desea o no realizar una operación una vez que la biblioteca termine de ejecutar una función determinada. Estos métodos son llamados handler. Para la implementación del simulador se utilizó la interfaz ITransport que brinda la opción de escribir y leer datos de un puerto y los objetos que recibirían esta información se hicieron hijos de ITransporHandler, interfaz que posee los métodos handler relacionados con estas operaciones. De igual forma se utilizó la interfaz IAsiyncTimer para el manejo del tiempo e ITimerHandler que brinda los handler pertinentes. Y el IAceptor e IAceptorHandler para la gestión de las conexiones que lleguen a un puerto Ethernet.

2.7.2 Clases principales de la capa de protocolo

En la implementación de este componente se definieron dos clases fundamentales. La clase *Device* encargada del manejo de la lógica en la comunicación por un canal exclusivo. Y la clase *Session* encargada de manejar la lógica de la comunicación TCP/IP. Además fueron necesarias otro grupo de clases sobre las que se apoyarían las ya mencionadas para cumplir con su función. En el siguiente diagrama se muestran las clases principales de esta capa y las relaciones entre ellas.

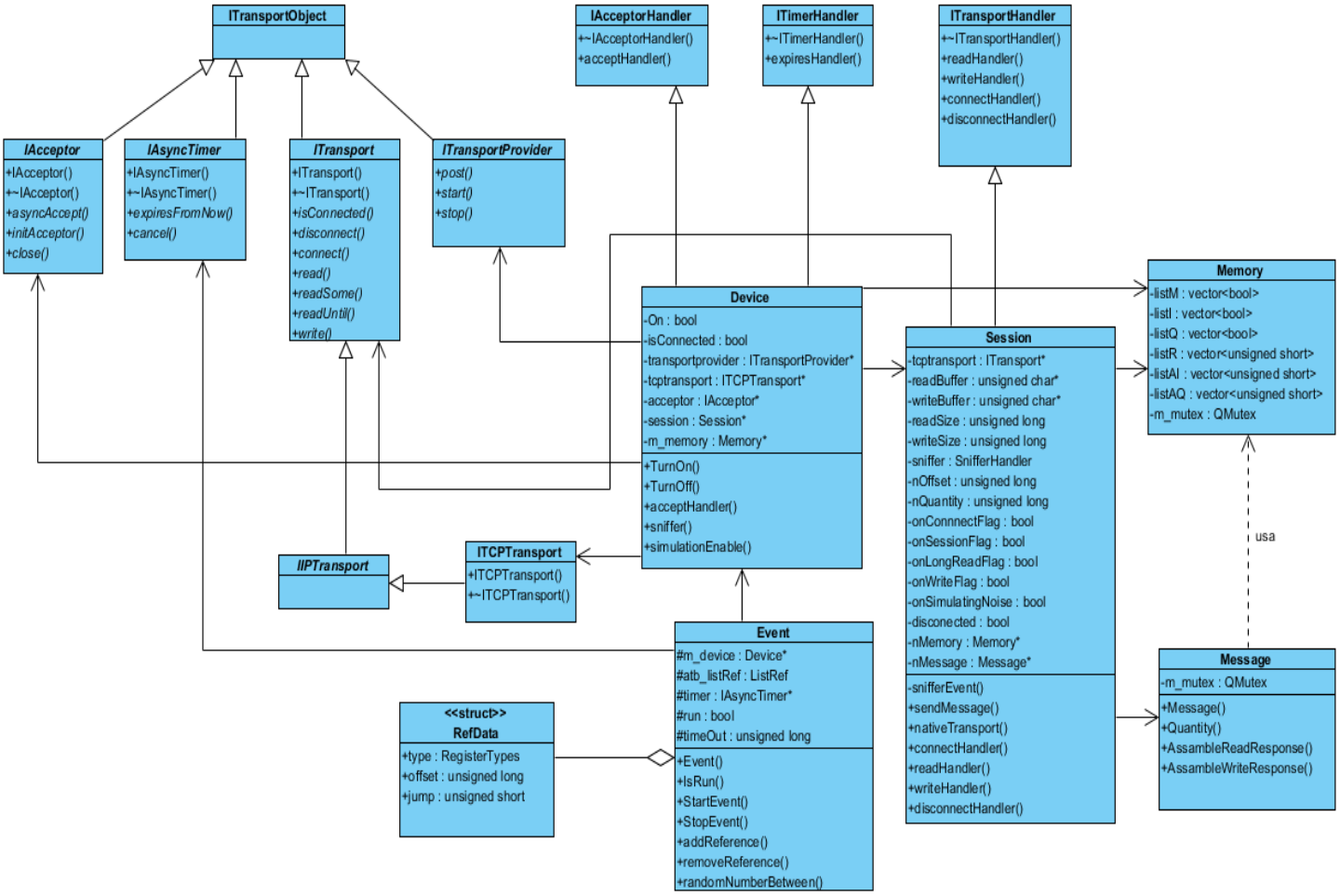


Fig. 19 Clases principales de las capas de protocolo y de transporte.

2.7.3 Clases principales de la capa de aplicación

Esta capa se concibió para servir de enlace entre la capa de protocolo y el usuario. Su objetivo específico es enviar a la capa de protocolo las órdenes del usuario y mostrar al usuario la información generada en la capa de protocolo. De esta forma, la manera en que el usuario interactúa con el simulador puede ser cambiada con facilidad sin tener que cambiar el funcionamiento interno del sistema. Para ello se utilizaron las clases y funcionalidades que brinda el marco de trabajo de Qt para el desarrollo de GUI.

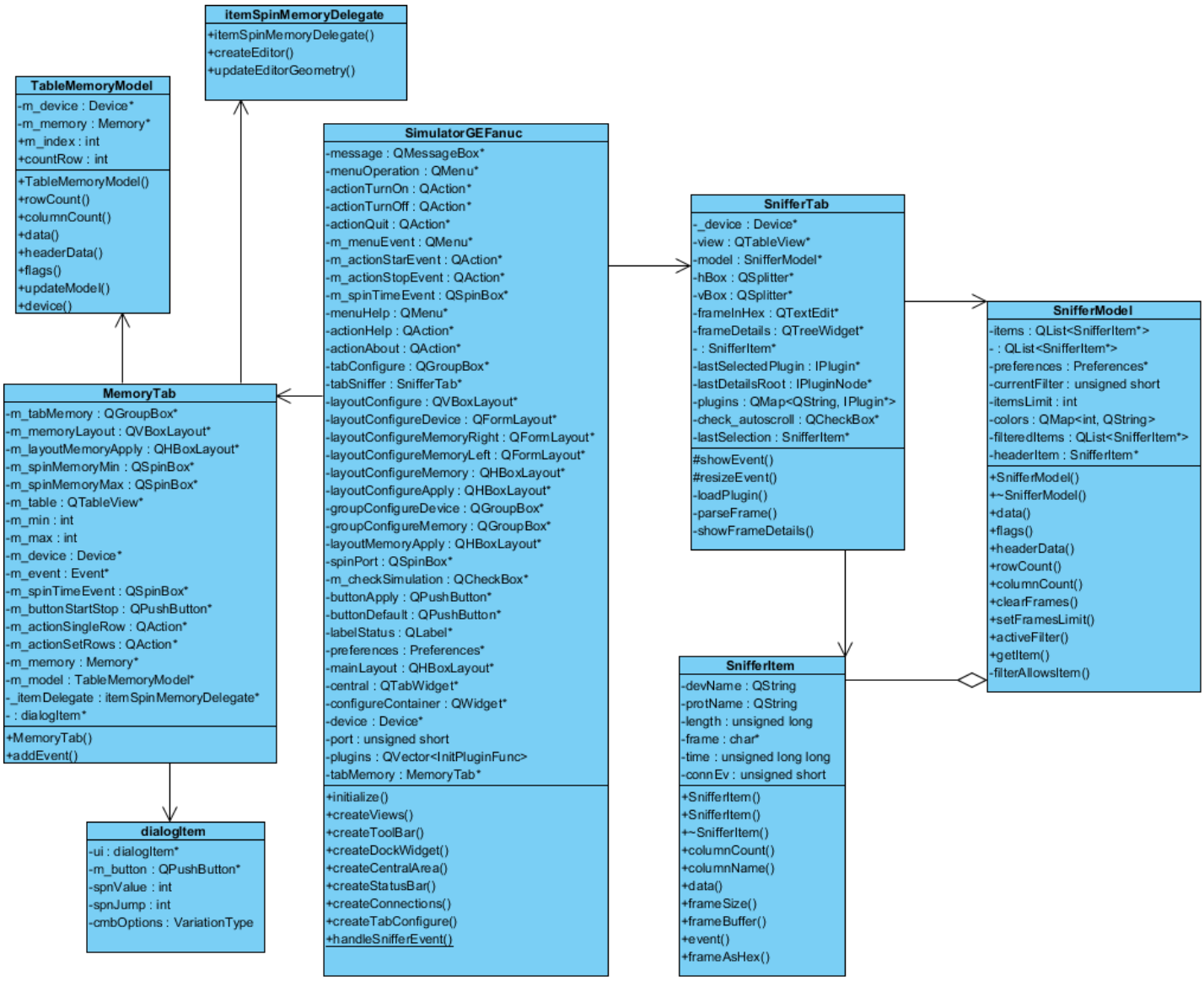


Fig. 20 Clases de la capa de aplicación

2.8. Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño cuya efectividad ha sido comprobada por haber sido empleada para resolver problemas similares en ocasiones anteriores. Otra característica fundamental es que deben ser reusables, lo que significa que sean aplicables a diferentes problemas de diseño en distintas circunstancias. Los patrones de diseño pretenden (32):

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño. Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

2.8.1 Patrones GRASP

Los patrones GRASP, más que patrones propiamente dichos, son una serie de “buenas prácticas” de aplicación recomendable en el diseño de software. Entre ellos, los patrones Experto, Creador, Bajo acoplamiento, Alta cohesión y Controlador. Son patrones generales de software para asignación de responsabilidades, es el acrónimo de "GRASP (*object-oriented design General Responsibility Assignment Software Patterns*)". (33)

Experto en información

El patrón Experto plantea el problema “¿Cuál es el principio fundamental mediante el cual se asignan responsabilidades a los objetos?” y aporta la solución “Asignar la responsabilidad a la clase que tiene la información necesaria para cumplirla”. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). El patrón Experto posibilita una adecuada asignación de responsabilidades facilitando la comprensión del sistema, su mantenimiento y adaptación a los cambios con reutilización de componentes. Los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener. (33)

En el simulador se le da uso a este patrón en varias de sus clases, tales como: Device, Session y Memory.

Creador

El patrón Creador aporta un principio general para la creación de objetos, una de las actividades más frecuentes en programación. Nos ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que cumpla una de las siguientes condiciones:

- Tiene la información necesaria para realizar la creación del objeto.

- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.
- Contiene o agrega la clase.

Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización. (33)

En el simulador este patrón es utilizado en muchas de sus clases tales como: SimulatorGEFanuc, Device y Message.

Controlador

El patrón controlador sirve como intermediario entre la interfaz de usuario y las clases que encapsulan los datos. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. (33)

En el simulador desarrollado algunas de las clases cumplen con ser controladoras, las principales controladoras son: SimulatorGEFanuc, MemoryTab y SnifferTab.

Bajo acoplamiento

El patrón Bajo Acoplamiento es una medida de la fuerza con que una clase se relaciona con otras, porque las conoce y recurre a ellas; una clase con bajo acoplamiento no depende de muchas otras, mientras que otra con alto acoplamiento presenta varios inconvenientes: es difícil entender cuando está aislada, es ardua de reutilizar porque requiere la presencia de otras clases con las que esté conectada y es cambiante a nivel local cuando se modifican las clases afines. Los conceptos de cohesión y acoplamiento no están íntimamente relacionados, sin embargo se recomienda tener un mayor grado de cohesión con un menor grado de acoplamiento. De esta forma se tiene menor dependencia y se especifican los propósitos de cada objeto en el sistema. Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potencia la reutilización, y disminuye la dependencia entre las clases

- ✓ **Acoplamiento de Contenido:** Cuando un módulo referencia directamente el contenido de otro módulo. (En lenguajes de alto nivel es muy raro).
- ✓ **Acoplamiento Común:** Cuando dos módulos acceden (y afectan) a un mismo valor global.
- ✓ **Acoplamiento de Control:** Cuando un módulo le envía a otro un elemento de control que determina la lógica de ejecución del mismo. (33)

Este patrón se evidencia claramente en las clases: Memory y Event.

Alta cohesión

El patrón Alta Cohesión es una medida que determina cuán relacionadas y adecuadas están las responsabilidades de una clase, de manera que no realice un trabajo colosal; una clase con baja cohesión realiza un trabajo excesivo, haciéndola difícil de comprender, reutilizar y conservar. Nos dice que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. (33)

En el simulador se le da uso a este patrón en varias de sus clases, tales como: Memory, Event y Device.

2.8.2 Patrones GoF

El término GoF proviene del inglés Gang of Four, en español pandilla de cuatro. Dicha pandilla estaba compuesta por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides. Este grupo escribió el libro "Design Patterns". En este famosísimo libro, a menudo referenciado como GoF, los autores recogían 23 patrones comunes en el diseño de software, explicando al detalle en qué consistían y describiendo las soluciones adoptadas típicamente para cada problema. Toda una enciclopedia sobre diseño de software para el programador. (34)

Singleton

El patrón de diseño Singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. No se encarga de la creación de objetos en sí, sino que se enfoca en la restricción en la creación de un objeto. (32)

Este patrón fue utilizado en la creación de una única instancia de la clase Memory. Device es la encargada de esta creación y de brindar el acceso a la memoria a las demás clases que lo necesiten.

Fachada

El patrón de diseño Fachada sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. (32)

Este patrón se utilizó en la creación de la interfaz visual principal, la cual unifica las funcionalidades que posee el sistema.

Observador

Este patrón define una dependencia del tipo *uno-a-muchos* entre objetos, de manera que cuando uno de los objetos cambia su estado, el *Observador* se encarga de notificar este cambio al resto de los objetos dependientes. El objetivo de este patrón es desacoplar la clase de los objetos clientes del objeto, aumentando la modularidad del lenguaje, así como evitar bucles de actualización. (32)

Este patrón se usó en la visualización de la memoria, ya que esta al ser modificada necesita notificar el cambio a la interfaz del usuario.

Conclusiones parciales

En este capítulo se seleccionó como metodología a seguir AUP-UCI, la cual se ajusta a las características de este desarrollo ya que es muy flexible y se cuenta con un solo desarrollador. Se logró una mejor comprensión de la estructura, a través de los diagramas del sistema y la descripción de la arquitectura. A partir de la descripción de las herramientas utilizadas en el desarrollo de la tesis se logró un mayor acercamiento a las funcionalidades y potencialidades que brindan. También, se esclarecen las funcionalidades que serán implementadas a partir de las historias de usuario descritas.

Capítulo 3 Implementación y pruebas del sistema

Introducción

En este capítulo comienza la implementación del sistema. Para describir este proceso se utilizan las tareas de ingeniería. Estas permiten organizar el proceso de implementación, además, posibilitan que sea conocido el grado de complejidad de cada historia de usuario teniendo en cuenta la cantidad de tareas asociadas a ella. Una vez que la implementación culmine se realizan las pruebas al sistema.

3.1. Tareas de ingeniería

Las tareas de ingeniería son pequeñas actividades que los desarrolladores conocen que el sistema debe hacer, y son deducibles a partir de las historias de usuario. Estas deben ser estimables, su tiempo de implementación debe ser corto (aproximadamente entre uno y tres días) y responden directamente a la historia de usuario de la cual se deriva. Una historia de usuario puede tener una o varias tareas, en dependencia de su complejidad. También pueden existir tareas de ingeniería técnica, que son aquellas que no responden a ninguna historia de usuario, pero son necesarias para que el sistema funcione bajo los requisitos especificados.

Para el desarrollo de este acápite se propone el siguiente modelo para tareas de ingeniería:

Tabla 2 Modelo para las tareas de ingeniería

Tareas de ingeniería	
Número de la tarea: N	**Número de HU:** N
Nombre de la tarea: Nombre	
Tipo de tarea: Tipo	**Días estimados:** Cantidad

Programador responsable: Responsable
Descripción: Descripción

Argumentada en los siguientes campos:

- **Número de la tarea:** Número consecutivo asignado a la tarea, ayuda a organización de dicho artefacto.
- **Número de la historia de usuario:** Una tarea de ingeniería debe responder a una, o varias historias de usuario, en este campo se especifica la historia o historias a las que responde la tarea. En caso de ser una tarea técnica, este campo se deja en blanco.
- **Nombre de la tarea:** Nombre asignado a la tarea de ingeniería, debe expresar explícitamente su objetivo.
- **Tipo de tarea:** Una tarea puede variar su tipo en dependencia de su objetivo, puede ser, como se abordó anteriormente, una tarea técnica, pero también, respondiendo a una historia de usuario, una tarea de desarrollo, de diseño, o de una revisión y/o prueba específica.
- **Programador responsable:** Dispone quién es el responsable de la ejecución de la tarea.
- **Días estimados:** Una tarea de ingeniería se puede expresar en horas o días, en dependencia del sistema de trabajo y el calendario del equipo de desarrollo. La sumatoria del tiempo de las tareas de ingeniería en una historia de usuario no puede sobrepasar el valor convertido a puntos estimados de la propia historia.
- **Descripción:** Expresa una breve descripción sobre los objetivos de la tarea de ingeniería.

Tabla 3 Resumen de las tareas de ingeniería.

No. HU	Tareas	No. Tarea
	Instalación y configuración del entorno de trabajo.	1
1	Diseño e implementación de los elementos visuales para la configuración del dispositivo.	2

1	Implementación de las clases Dispositivo (Device) y Sesión (Session).	3
1	Obtención de las tramas GE Fanuc.	4
1	Implementación de métodos para cambiar el estado del dispositivo.	5
2,3,4	Implementación de la clase Mensaje (Message).	6
2,3,4	Desensamblar mensajes de peticiones provenientes del manejador.	7
2,3,4	Ensamblar mensajes de respuesta a las peticiones del manejador.	8
5	Implementación de la clase Memoria (Memory).	9
5	Obtención de los registros del dispositivo.	10
5,6	Modificación del valor de un registro del dispositivo.	11
6	Simular valores ascendentes en el mapa de memoria.	12
6	Simular valores descendentes en el mapa de memoria.	13
6	Simular valores aleatorios en el mapa de memoria.	14
7	Diseño e implementación de la visualización de tramas.	15
8	Implementación de fallas en las tramas de envío.	16
	Ejecución de las pruebas necesarias para validar el correcto funcionamiento software.	17

3.2. Diagrama de despliegue del sistema

Los diagramas de despliegue describen la arquitectura física del sistema durante la ejecución y su topología: la estructura de los elementos de hardware y software que ejecuta cada uno de ellos. El diagrama de despliegue está constituido por nodos físicos en los cuales se ejecutan los componentes.

Fig. 21 Diagrama de despliegue del sistema

Manejador GE Fanuc: en el nodo de la izquierda se corre el manejador desarrollado en la línea de adquisición del proyecto Desarrollo del SCADA GALBA.

Simulador GE Fanuc: en el nodo de la derecha estará ejecutándose la aplicación desarrollada para realizar las pruebas del manejador GE Fanuc dada la no existencia del dispositivo de campo aquí en el país.

TCP/IP GE Fanuc: protocolos utilizados en la comunicación entre el manejador GE Fanuc y el simulador GE Fanuc. Además, el simulador lo utiliza para poder comunicarse con el manejador.

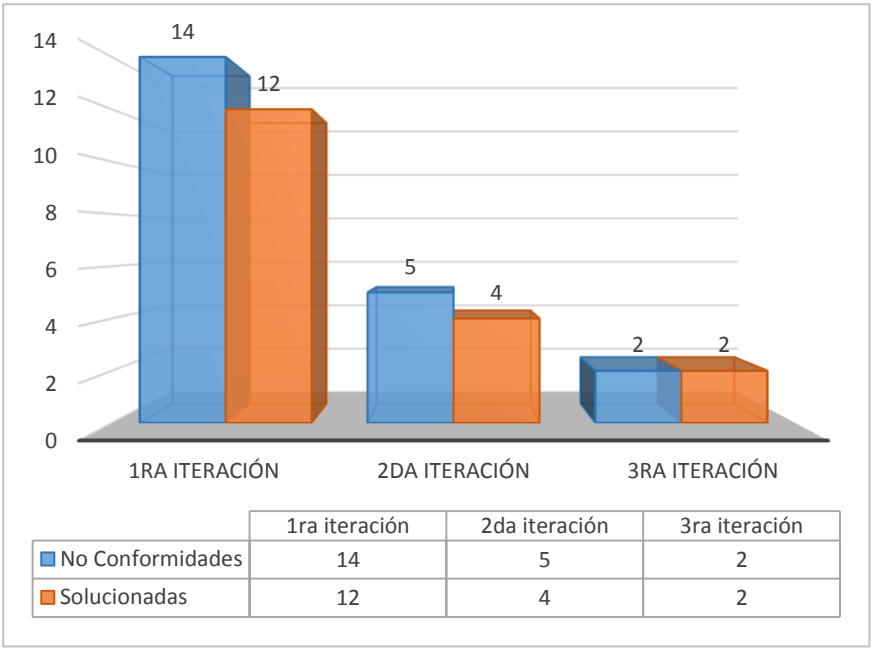
3.3. Pruebas del sistema

Las pruebas son procesos que se llevan a cabo con el objetivo de verificar la calidad de un producto, detectando los posibles errores que puede presentar. Existen disímiles tipos de pruebas, y cada una de ellas fue diseñada con el objetivo de evaluar diferentes aspectos de un producto.

Al sistema se le realizaron tres tipos de pruebas:

- Las **pruebas unitarias** se realizaron con el objetivo de validar que el código sea funcional. Para ello se utilizó la técnica o **método de caja negra** sobre la interfaz del programa. Estas pruebas comenzaron a mucho antes de terminar la implementación, ya que era necesario probar las funcionalidades a medida que se iban desarrollando. Esto garantizó la culminación de una aplicación que posee el comportamiento esperado por el usuario.
- Las **pruebas de aceptación** se realizaron con el objetivo de validar que el sistema cumple con el funcionamiento esperado y permitió al usuario determinar su aceptación, desde el punto de vista de

su funcionalidad y rendimiento. Las pruebas de aceptación fueron definidas por el usuario del sistema y preparadas por el desarrollador, aunque la ejecución y aprobación final correspondió al usuario. Cada HU sólo se consideró terminada cuando pasó correctamente todas las pruebas de aceptación.



	1ra iteración	2da iteración	3ra iteración
No Conformidades	14	5	2
Solucionadas	12	4	2

Fig. 22 Pruebas de aceptación.

La gráfica (Fig. 22) muestra el resultado de las pruebas de aceptación realizadas al sistema, la misma fue dividida en tres iteraciones: en la primera iteración fueron detectadas catorce no conformidades de las cuales se resolvieron doce, en la segunda iteración cinco no conformidades de las cuales a cuatro de estas se les dio solución y en la última iteración fueron detectadas dos no conformidades las cuales fueron solucionadas. Al sistema carecer de no conformidades se puede concluir que este funciona correctamente, además de que cuenta con la calidad requerida por el usuario.

- Finalmente se realizaron al simulador las **pruebas de estrés** con el objetivo de conocer la robustez de la aplicación cuando la carga es extrema. En esta prueba se midió el uso de memoria RAM de la aplicación cuando se aumenta el número de registros de memoria a visualizar, además se puso a generar eventos aleatorios a todos estos registros. Como resultado se obtuvo que: la aplicación utiliza 4mb de memoria al ejecutarse, aumenta a 16mb al comenzar a responder a las peticiones de

lectura y escritura del manejador y aumenta a 23mb al poner a generar números aleatorios a los 48000 valores visualizados. Se pudo arribar a la conclusión de que la aplicación no sobrepasa el nivel permisivo para una computadora convencional en cuanto a memoria RAM se refiere.

Conclusiones parciales

En este capítulo se logró implementar el sistema a través de la descripción de las Tareas de Ingeniería. Una vez que este proceso culminó se realizaron las pruebas unitarias, de aceptación y de estrés sobre el producto terminado, con las cuales se logró una aplicación que posee el comportamiento esperado por el usuario, con la calidad requerida por este y que no sobrepasa el nivel permisivo para una computadora convencional en cuanto a memoria RAM se refiere.

Conclusiones

En el presente trabajo se propuso desarrollar un simulador del dispositivo PLC Series 90-70 para realizar las pruebas al manejador GE Fanuc del SCADA GALBA. Al finalizar la investigación sobre el desarrollo del simulador se puede concluir que se le dio solución al problema planteado, con lo cual se logró:

- Identificar las principales tendencias en el proceso de simulación de dispositivos industriales, a partir del estudio del estado del arte.
- Descartar la existencia de alguna herramienta que brinde solución al problema identificado en esta investigación, luego de la búsqueda y el análisis de varios simuladores existentes.
- Diseñar el modelo a desarrollar a través del uso de la metodología, tecnologías y herramientas seleccionadas.
- Desarrollar un programa que funciona correctamente, pero además cumple con las exigencias y necesidades requeridas por el cliente, garantizado por la ejecución de las pruebas, que permitieron detectar y corregir los errores cometidos durante la implementación.
- Desarrollar una aplicación con estructura bien sólida y reutilizable, todo gracias al uso el estilo arquitectónico n-capas.
- Desarrollar un sistema con la capacidad de recibir, procesar y responder peticiones de datos, dando cumplimiento a los requisitos de mayor importancia para el cliente, a partir de la implementación del protocolo GE Fanuc y la gestión de la memoria.
- Implementar funcionalidades al sistema que permiten generar y visualizar datos en diferentes formatos, así como, recrear situaciones excepcionales que pueden presentarse en las comunicaciones, haciendo posible la detección de errores en el comportamiento del manejador que se prueba.

Recomendaciones

- Adicionar a la capa de protocolo, perteneciente a la arquitectura del sistema, la funcionalidad de simular retraso en la comunicación con el objetivo de ampliar el número de situaciones que la aplicación podría simular.
- Desarrollar un simulador genérico de dispositivos, que implemente varios protocolos de comunicación.

Bibliografía

1. **Almenara, Julio Cabero.** Nuevas tecnologías, comunicación y educación. *Revista electrónica de tecnología educativa.* [En línea] 1996. http://www.edutec.es/revista/index.php/edutec-e/article/download/576/305.

2. **Boyer, Stuart A.** SCADA: supervisory control and data acquisition. *International Society of Automation.* [En línea] 2009. http://dl.acm.org/citation.cfm?id=1717879.

3. **Chavez Frias, Hugo Rafael.** El golpe fascista contra Venezuela. *discursos e intervenciones.* [En línea] Enero de 2003.

4. **Nieto Doce, Yanet.** Sistema SCADA. *Serie Científica.* [En línea] 3 de Julio de 2010. https://publicaciones.uci.cu/index.php/SC/article/view/435.

5. **Perez Javier, Maikel.** Manual Organizacional: CEDIN. [En línea] Febrero de 2015. [Citado el: 12 de Marzo de 2015.] http://gespro.cedin.prod.uci.cu/.

6. **Breña Pérez, Yerenia.** Aplicación de estándares de calidad de servicios para el Subsistema de Comunicación con Terceros del sistema SCADA Guardián del ALBA. *La Habana.* [En línea] Junio de 2011. http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04742_11/1/TD_04742_11.pdf.

7. **León García, Héctor Ernesto .** Manejador para la comunicación fiable en un sistema de supervisión y control basado en microcontroladores. [En línea] 2012. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_05608_12.

8. **Hechavarría Rodríguez, Dayra Iris y Herrera Pérez, Lannie Octavio .** Procedimientos para la Gestión de Requisitos en el SCADA SAINUX. *Revista Cubana De Ciencias informáticas.* [En línea] 2012.

9. **Himmelblau, David M y Bischoff, Kenneth B.** *Análisis y simulación de procesos.* Barcelona : Reverté, 1992.

10. **Shannon, Robert E.** *Systems simulation: the art and science.* s.l. : Englewood Cliffs, NJ : Prentice-Hall, 1975. pág. 387. Vol. 1.

11. **García Dunna, Eduardo, García Reyes, Heriberto y Cárdenas Barrón, Leopoldo Eduardo.** *Simulación y análisis de sistemas con ProModel.* s.l. : Pearson Educación, 2006.

12. **Abraham Ravelo, José Carlos.** *Desarrollo de un simulador para realizar las pruebas del manejador*

Modbus Omni. Primera. La Habana : s.n., 2012. pág. 90. Vol. 1.

13. **Albuerne Rivero, Miguel Angel.** Simulador de dispositivos PLC-5 para realizar el proceso de pruebas al manejador DF1 del SCADA SAINUX. [En línea] 2013. http://bibliodoc.uci.cu/RDigitales/2013/octubre/24/TD_06604_13.pdf.

14. **García Quiñonez, Sheyla.** Simulador para el protocolo de comunicación DNP3 en su variante serial. *La Habana.* [En línea] 2013. http://bibliodoc.uci.cu/RDigitales/2013/octubre/18/TD_06711_13.pdf.

15. **International Business Machines (IBM).** http://www.ibm.com/us/en/. [En línea] IBM AIX V7.1 documentation, AIX Technology Level descriptions and comparisons, 2015. [Citado el: 25 de Mayo de 2015.] https://www-01.ibm.com/support/knowledgecenter/ssw_aix_71/com.ibm.aix.genprogc/master_slave_model.htm.

16. **Tanenbaum, Andrew S.** *Redes de Computadoras.* Cuarta. s.l. : Pearson, 2003. ISBN 0-13-066-102-3.

17. **Conde García, Luis.** gestiopolis. [En línea] 2015. http://www.gestiopolis.com/protocolo-de-control-de-transmision-tcp/.

18. **Marín Moreno, William.** Modelo OSI. [En línea] 2003. http://www.ie.itcr.ac.cr/marin/telematica/trd/01_modelo_OSI_v2.pdf.

19. **GE Fanuc.** SNP Communications Reference. [En línea] 2005.

20. **Automation, GE Fanuc.** TCP/IP Ethernet Communications for the Series 90TM PLC. *GE Fanuc Automation.* [En línea] 2002.

21. **Moreno Borges, Adrián Carlos y García Hernández, Luis Enrique.** *Guía de implementación del manejador GEFanuc.* La Habana : s.n., 2009.

22. —. *Diseño del Manejador para el protocolo GEFanuc.* La Habana : s.n., 2009.

23. **Autómatas Programables.** Comunicación de Autómatas Programables. [En línea] Mayo de 2006. http://redeselectricas.com/archivos/gefanuc/gefanuccatalogoespanol.pdf.

24. **Empresas NorteSur.** Prpductos GE Fanuc. *Empresas Norte Sur, C.A. .* [En línea] 2001. http://www.empresasnortesur.com/lineas2.htm.

25. **Pressman, Roger S.** Ingeniería del Software: Un enfoque práctico. *Mikel Angoar.* [En línea] 1997.

https://www.google.com/books?hl=en&lr=&id=8UV5jxkuBZIC&oi=fnd&pg=PP3&dq=Las+metodolog%C3%ADas+de+desarrollo+de+software&ots=wJQr_TKmCO&sig=-yLiS3M6nG4L3s1fhV_oRgyDJ0Q.

26. **QT, Company.** Framework QT. [En línea] 2015. https://www.qt.io/qt-framework/.

27. **Qt Documentation.** Qt Creator Manual. [En línea] 2015. http://doc.qt.io/qtcreator/.

28. **Stroustrup, Bjarne.** El lenguaje de programación C++. *Addison Wesley.* [En línea] 2007. http://dialnet.unirioja.es/servlet/libro?codigo=370770.

29. **Fontela, Carlos.** *UML: modelado de software para profesionales.* Primera. Buenos Aires : Alfaomega Grupo Editor Argentino, 2011. pág. 184.

30. **UML, Guión Visual Paradigm for.** 2013.

31. **Roselló Diaz, Katia.** Metodología de desarrollo para la Actividad productiva de la UCI. [En línea] 7 de Marzo de 2015.

32. **JUNTA DE ANDALUCÍA.** Patrones de Diseño. *Marco de Desarrollo de la Junta de Andalucía.* [En línea] 2015. http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/13.

33. **Botero Tabares, Ricardo.** GRASP: Más patrones para asignar responsabilidades. *Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación.* [En línea] 2010. [Citado el: 10 de Junio de 2015.] http://biblioteca.ucp.edu.co/ojs/index.php/entrecei/article/download/760/721.

34. **GENBETA.** Los patrones de diseño de software. [En línea] 6 de Abril de 2011. http://www.genbetadev.com/metodologias-de-programacion/los-patrones-de-diseno-de-software.

35. **Hernández Meléndrez, Edelsis.** *Metodología de la invaetigación. Cómo escribir una tesis.* La Habana : Escuela Nacional de Salud Pública, 2006. pág. 51. Vol. 1.

36. **Zambrano Álvarez, Miguel Angel.** *¿CÓMO HACER Y DEFENDER UNA TESIS?* San Cristobal : EDITORIAL TEXTO, C.A., 2011.

37. **García Hernández, Luis Enrique y Pérez Pérez, Yosep Yasmany.** *Recolector Gráfico: Diseño del Producto.* La Habana : s.n., 2011.

38. **Anónimo.** APRENDA Qt4 DESDE HOY MISMO. [En línea] 10 de 2010. http://sunshine.prod.uci.cu/gridfs/sunshine/books/Aprenda_Qt4_Hoy_Mismo.pdf.

39. **GE Fanuc, Automation.** TCP/IP Ethernet Communications for the Series 90™ PLC. *User's Manual.*

[En línea] Mayo de 2002. http://support.ge-ip.com/support/resources/sites/GE_FANUC_SUPPORT/content/staging/DOCUMENT/0/DO213/en_US/1.0/gfk1541b.pdf. GFK-1541B.

40. **Gamma, Erich, y otros.** Design Patterns. *Elements of Reusable Object-Oriented Software.* [En línea] 1994.

41. **Eckel, Bruce .** Pensar en C++. [En línea] Enero de 2012.

42. **Larman, Craig.** UML y Patrones. *Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* [En línea] http://sunshine.prod.uci.cu/gridfs/sunshine/books/Craig_Larman_-_UML_y_Patrones.pdf.

43. **MARTÍNEZ JUAN, FRANCISCO JAVIER.** GUÍA DE CONSTRUCCIÓN DE SOFTWARE EN JAVA CON PATRONES DE DISEÑO. *PROYECTO FIN DE CARRERA.* [En línea] http://sunshine.prod.uci.cu/gridfs/sunshine/books/GUIA_DE_CONSTRUCCION_DE_SOFTWARE_EN_JAVA_CON_PATRONES_DE_DISEO.pdf.

Glosario de términos

Controlador Lógico Programable (PLC): dispositivos electrónicos usados en automatización industrial para realizar estrategias de control básicas. Por su robustez y características sencillas de control, están cercanas al proceso, permiten ejecutar las tareas básicas del control, aunque no tenga conexión a las capas superiores del control.

Framework: en los sistemas orientados a objeto, un *framework* es un conjunto de clases que encapsulan diseños abstractos de soluciones a un determinado números de problemas en relación. Los objetivos principales que persigue un *framework* son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Protocolo Ethernet-IP: (Ethernet/Industrial *Protocol*) es un sistema de comunicación diseñado para entornos industriales. *Ethernet-IP* permite que los dispositivos de control intercambien información crítica con requerimientos estrictos de tiempo.

Protocolos de comunicación: son como reglas de comunicación que permiten el flujo de información entre computadoras o dispositivos diferentes que manejan lenguajes distintos.

Sistemas de Supervisión, Control y Adquisición de Datos (SCADA): aplicación de software especialmente diseñada para funcionar sobre computadores en el control de producción. Proporciona comunicación con los dispositivos de campo y controla el proceso de forma automática desde la pantalla del operador y provee toda la información asociada al proceso.

Software libre: Las cuatro reglas esenciales que deben cumplir las aplicaciones para ser consideradas como software libre son:

- Libertad 0: la libertad de ejecutar el programa como se desea, con cualquier propósito.
- Libertad 1: la libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que se quiera. El acceso al código fuente es una condición necesaria para ello.
- Libertad 2: la libertad de redistribuir copias para ayudar a su prójimo.
- Libertad 3: la libertad de distribuir copias de sus versiones modificadas a terceros. Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello.

TCP/IP: familia de protocolos sobre los cuales funciona Internet, que se ha convertido en el estándar

actual de comunicación entre computadoras. Conocida por estas siglas debido a que los dos protocolos más importantes son: el protocolo IP, que se ocupa de transferir los paquetes de datos hasta su destino adecuado y el protocolo TCP, que se ocupa de garantizar que la transferencia se lleve a cabo de forma correcta y confiable.

Trama: Es una unidad de envío de datos. Viene a ser sinónimo de paquete de datos.

Dispositivo: Se denomina dispositivo a un elemento de hardware que alberga información de proceso o estado de sí mismo, que forma parte de un sistema automatizado. Comúnmente los dispositivos son Controladores Lógicos Programables, Computadoras Industriales, Sistemas de Control Distribuidos, sensores o actuadores inteligentes con capacidad de comunicación.

GNU/Linux: El núcleo Linux se complementa con una serie de aplicaciones desarrolladas por el grupo GNU para conformar el sistema operativo *software* libre GNU/Linux. Linux/GNU es además multiusuario, multitarea, multiprocesador, multiplataforma y multilingüe, nacido en la red de redes Internet.

Interfaz: Zona de contacto o conexión entre dos componentes de *hardware*, entre dos aplicaciones o entre un usuario y una aplicación. Apariencia externa de una aplicación informática.

RAM: Siglas de *Random Access Memory*, que en español significa Memoria de Acceso Aleatorio. Es donde el computador guarda los datos que está utilizando en el momento presente. El almacenamiento es considerado temporal por que los datos y programas permanecen en ella mientras que la computadora esté encendida o no sea reiniciada.