Universidad de las Ciencias Informáticas
Facultad 5

“Simulador para el protocolo de comunicación industrial AB Ethernet”

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

Autores: José Raúl Meriño Maceira

Enrique Yecier Pardo Vega

Tutores: Ing. Pedro A. Uriarte Rodríguez

Ing. Yunaime Noda Cid

“Año 57 de la Revolución”
La Habana, Cuba
Junio 2015


“El hombre nunca sabe de lo que es capaz hasta que lo intenta”
Charles Dickens

Declaración de autoría

Declaramos ser los únicos autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.

_____ _____

José Raúl Meriño Maceira
Autor

Enrique Yecier Pardo Vega
Autor

_____ _____

Ing. Pedro A. Uriarte Rodríguez
Tutor

Ing. Yunaime Noda Cid
Tutor

Datos de contacto

Tutor: Ing. Pedro A. Uriarte Rodríguez

Edad: 29

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Ingeniero en Ciencias Informáticas.

Síntesis del Tutor: Graduado de Ingeniería en Ciencias Informáticas en julio del 2010 en la Universidad de Ciencias Informáticas.

E-mail: pauriarte@uci.cu.

Tutor: Ing. Yunaime Noda Cid

Edad: 31

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Ingeniero en Ciencias Informáticas.

Síntesis del Tutor: Graduado de Ingeniería en Ciencias Informáticas en julio del 2010, en la Universidad de Ciencias Informáticas

E-mail: ynoda@uci.cu.

Dedicatoria

A mi madre, porque ella es el soporte de mi mundo.

A mi abuela Verónica, por todo su amor, comprensión y protección y que a pesar de que no me pudo ver cumplir mi sueño de graduarme sé que desde el cielo siempre ha estado cuidando de mí.

Enrique Yecier Pardo Vega

A mi madre, por darme todo sin pedir nada a cambio y ser lo más tierno de mi vida.

A mis tíos, porque son como padres para mí.

A mi bisabuela Nena por todo su amor y cariño, y aunque no me pudo ver alcanzar mi meta, sé que estaría orgullosa de mí.

José Raúl Meriño Maceira

Agradecimientos

A mi mamá por su amor puro y por todos los sacrificios que tuvo y todavía hace por complacerme y verme feliz, por siempre estar ahí en las buenas y en las malas. Por cuidar de mí y enseñarme a ser una buena persona y enseñarme que para lograr algo hay que luchar por ello.

A mi papá por su ejemplo, su constante preocupación y su amor que a su forma siempre me lo demuestra.

A mis abuelos y abuelas, a los que están y a los que por ley de la vida me faltan, por sus consejos y ese amor tan dulce que solo ellos saben dar.

A mi novia, por estos hermosos 3 años en los que lo hemos compartido todo juntos. En los que me ha hecho sacar la mejor parte de mí y por no dejar nunca que me dé por vencido por muy difícil que fuera la situación. Por estar siempre ahí, por comprenderme y por aguantarme en momentos de estrés. Gracias por todo.

A mi hermanita Camila por quererme incondicionalmente.

A mis tíos que me acogieron aquí en la habana e hicieron más llevadera la distancia de mis padres, en especial a mi tío David y mi tío Raymundo por su apoyo incondicional en estos 5 años que parecen muchos pero que se me fueron volando.

A mi familia en general por ser un pilar de apoyo en todo momento.

A los tutores por la guía y la ayuda que me dieron en este último año, que sin ellos este logro no hubiera sido posible.

A mi compañero de tesis, por todo el sacrificio y la tensión que pasamos juntos para llegar a cumplir esta bella meta.

A todos los profesores que influyeron de una forma y otra en mi formación personal.

A mis amistades, a mis compañeros de clase a todos ellos. A Eduardo, Ale, al Loco Frank, Adolfo a todos que a pesar de todas las maldades que hicimos juntos, siempre estuvieron ahí ante cualquier problema. No los menciono a todos porque serían 100 páginas y seguro que se me olvida alguien.

Enrique Yecier Pardo Vega

A mi mamá, por ser madre y padre a la vez, por estar conmigo en las buenas, las malas y las peores, gracias por todo lo que me has dado, gracias por traerme al mundo, cuidarme y hacerme una persona de bien, gracias por lo que me enseñaste y el amor que me inculcaste, gracias por hacerme comprender que el amor verdadero es ese que se entrega sin esperar nada a cambio, que cuando no tenía a quien acudir, sabía que podía contar contigo, y cuando todo se ponía difícil ahí estabas tú a mi lado diciéndome que todo saldría bien, sé que no me alcanzará la vida para devolverte todo lo que has hecho y haces por mí, sin ti no sería quien soy actualmente, todo te lo debo a ti.

A mi abuela Esther, por dar a luz a la persona que más yo quiero en la vida, que es mi madre, por todo su amor y comprensión.

A mi hermanito Carlitos, gracias por quererme tanto y estar en mi vida, tu presencia me da fuerzas para seguir adelante.

A mis tíos, Raúl, Javier y David, gracias por ser como mis padres, por todo lo que han hecho y hacen por mí, por el apoyo que me han dado y por ayudarme a tomar decisiones en la vida.

A mi tía Yaime, que es como una madre para mí, gracias por todo lo que has hecho y haces para ayudarme a cumplir mi sueño, por todo el apoyo y cariño que me has brindado.

A mi novia, que a pesar de encontrarse en estos momentos lejos de mí, sé que está orgullosa, gracias por aguantarme, quererme, apoyarme y darme ánimo cuando más lo necesitaba, por querer siempre lo mejor para mí y hacer de mí una mejor persona, gracias por existir, te quiero mucho mi princesa.

A mi compañero de tesis, por todo el sacrificio, la fuerza que me diste y los dolores de cabeza que pasamos juntos para alcanzar nuestra meta.

A mis amigos, Eduardo, Carlos, Frank el loco, Alfonso, Alejandro, Yaikel, Hayron, Eliades, Handy, Nino, Orlando, Hendrick, me es imposible mencionarlos a todos, porque no me

alcanzarían las hojas y no quiero que se me olvide alguno, gracias por estar a mi lado en los momentos malos y buenos que compartimos, y por todo lo que hicimos durante la carrera.

A mis tutores, gracias por su ayuda, sin ella no hubiera sido posible cumplir mi sueño.

A mi familia, gracias a su cariño, guía y apoyo he llegado a realizar uno de los anhelos más grandes de mi vida.

José Raúl Meriño Maceira

Resumen

En el presente trabajo se describen las principales características y funcionalidades de una herramienta que actúa como esclavo en el intercambio de mensajes con el manejador AB Ethernet. Este sistema es necesario debido a la ausencia de una aplicación de este tipo en la línea de Adquisición de Datos del Centro de Informática Industrial, perteneciente a la Universidad de las Ciencias Informáticas. Esta situación dificultó la etapa de pruebas del manejador, proceso vital para lograr que dicho producto presente la menor cantidad de errores posibles. El proceso de desarrollo comenzó con el estudio de la simulación de procesos industriales y el análisis de algunos simuladores de dispositivos, del cual se definieron los requisitos funcionales del sistema. Se describieron algunos conceptos relacionados con la investigación tales como: Controlador Lógico Programable, Modelo Maestro/Esclavo y protocolo de comunicación TCP/IP. Además se analizaron las especificaciones del protocolo AB Ethernet y se seleccionaron las herramientas y tecnologías utilizadas en la implementación. Se definió la arquitectura y el diseño de la aplicación, finalizando con la ejecución de pruebas que validaron la calidad y el correcto funcionamiento del simulador. Como resultado se obtuvo un simulador capaz de generar datos, visualizarlos y activar situaciones excepcionales que puedan ocurrir en las comunicaciones.

Palabras Claves: simulador, SCADA, manejador, PLC, protocolo AB Ethernet, TCP/IP.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

La informática ha ido aumentando su presencia en cada esfera de la actividad humana a medida que avanza el tiempo. Su objetivo principal es mejorar las condiciones de trabajo de los seres humanos garantizando mayor calidad y rapidez en el manejo de la información. Su utilización en el control y automatización de procesos industriales ha facilitado estas operaciones. Un ejemplo de esto, se pone de manifiesto en los sistemas de Supervisión, Control y Adquisición de Datos (SCADA, por sus siglas en inglés), que son sistemas basados en computadores que permiten supervisar y controlar variables de un proceso a distancia, proporcionando comunicación con dispositivos de campo y controlando el proceso de forma automática por medio de un software especializado [1].

Estos sistemas están concebidos para prevenir y alertar sobre situaciones inesperadas en cualquier industria. Además de la recuperación de datos, el sistema puede realizar otras funciones que incluyen la supervisión y control de información en tiempo real, la cual es posible gracias a sensores y actuadores que se instalan en la industria. Esta información es transmitida al sistema en determinados momentos pudiendo ser vista y modificada por los operadores del sistema [2].

La Universidad de las Ciencias Informáticas desempeña una función importante en la informatización de la sociedad cubana. La misma se compone por varias facultades, entre ellas la facultad 5, en la cual radica el Centro de Informática Industrial (CEDIN). Este centro tiene entre sus objetivos principales el desarrollo de aplicaciones o servicios para la automatización de procesos industriales, como los sistemas SCADA. El CEDIN cuenta con varias líneas tales como Adquisición e Interfaz Hombre Máquina (HMI, por sus siglas en inglés).

La línea de Adquisición del SCADA desarrolla los módulos de *drivers*, recolector y procesamiento. Los *drivers* son los encargados de implementar los protocolos específicos que se requieren para establecer comunicación con los dispositivos. El recolector es el encargado de configurar los *drivers* y de asignarles las tareas de lectura y escritura de datos que se necesiten realizar. En procesamiento se recibe la respuesta de estas tareas y se procesa dicha información para luego ser almacenada o visualizada.

En el CEDIN los procesos de desarrollo y pruebas del manejador para los dispositivos que se comunican a través del protocolo industrial AB Ethernet, se han visto afectados. Esto se debe

a la carencia de dispositivos de campo (Controlador Lógico Programable, PLC por sus siglas en inglés) o de un simulador para realizar dichas pruebas. La ausencia de estos elementos dificulta validar el correcto funcionamiento del *driver* AB Ethernet ante situaciones excepcionales (cortes y ruido en las transmisiones), lo que provoca atrasos en su desarrollo y liberación, impidiendo además crear ambientes simulados que permitan el entrenamiento del personal del SCADA.

A partir de la situación problemática se define el siguiente **problema de la investigación:**

¿Cómo comprobar que en su versión MirandaR2 del SCADA el manejador AB Ethernet logra establecer comunicación y responder de forma correcta ante situaciones excepcionales?

Teniendo en cuenta lo antes expuesto el **objeto de estudio** se enmarca en los simuladores para protocolos de comunicación industrial.

Para dar solución al problema se definió como **objetivo general:**

Desarrollar un simulador para el protocolo de comunicación industrial AB Ethernet que permita probar el driver AB Ethernet ante situaciones excepcionales que puedan presentarse en las comunicaciones.

Se delimita como **campo de acción** la simulación de la comunicación mediante el protocolo industrial AB Ethernet.

Para cumplir con el objetivo propuesto, se definieron las siguientes **tareas de investigación**:

- Elaboración del marco teórico de la investigación a partir del estudio del estado del arte acerca de los simuladores para dispositivos industriales.
- Estudio de las especificaciones del protocolo industrial AB Ethernet para la definición de los requisitos funcionales que va a cumplir el simulador.
- Estudio y selección de los patrones de diseño y arquitectura para su utilización en el diseño de la interfaz de usuario y la representación de los diagramas de clase que van a guiar el proceso de implementación de la solución.
- Implementación del simulador para el protocolo AB Ethernet según la arquitectura definida.
- Diseño y ejecución de casos de prueba del simulador AB Ethernet para la validación del correcto funcionamiento del mismo.

La **idea a defender** en este trabajo es que: con el simulador desarrollado será posible validar el correcto funcionamiento del manejador AB Ethernet ante situaciones excepcionales que puedan ocurrir, disminuir el tiempo de desarrollo de nuevas versiones del mismo y contar con ambientes simulados para el entrenamiento del personal del SCADA.

El **posible resultado** es un simulador para las comunicaciones con el manejador AB Ethernet, que permita activar situaciones excepcionales, tales como: cortes en las transmisiones y ruido en los datos recibidos o enviados.

Métodos científicos utilizados en la investigación:

Métodos teóricos:

- **Analítico-Sintético:** utilizado para realizar un análisis integral de los elementos y fundamentos teóricos principales que integran el protocolo AB Ethernet.
- **Modelación:** utilizado para elaborar los diagramas de clase que componen la estructura del simulador, así como sus relaciones, atributos y funcionalidades.

Métodos empíricos:

- **Entrevista**: utilizado para establecer una comunicación verbal con personal calificado para poder obtener más información acerca del protocolo de comunicación industrial AB Ethernet.
- **Experimental:** utilizado en la ejecución de pruebas, para verificar la calidad del simulador desarrollado en la presente investigación.

El presente trabajo está conformado de la siguiente manera:

Capítulo 1. Fundamentación teórica: este capítulo contiene el marco teórico de la investigación que define los conceptos y características principales de la simulación, y de la comunicación mediante el protocolo AB Ethernet. También contiene las tecnologías y herramientas utilizadas para realizar el diseño e implementación de la solución.

Capítulo2. Descripción y diseño del simulador: en este capítulo se abordan las características y el diseño del software. Se explican la estructura de la solución propuesta y las funcionalidades principales con las que debe contar.

Capítulo 3. Implementación y prueba: el capítulo muestra los aspectos vinculados con la implementación del simulador, así como los estándares de codificación que permiten crear un código fácil de entender, tareas de ingenierías realizadas para cumplir con las historias de usuario y se finaliza con la realización del proceso de pruebas.

Capítulo #1. Fundamentación teórica

1.1 Introducción

En este capítulo se definen los conceptos y características principales de la simulación, y las especificaciones técnicas del protocolo AB Ethernet. También contiene las tecnologías y herramientas utilizadas para realizar el diseño e implementación de la solución, así como la metodología usada.

1.2 Simulación

El uso de la simulación es muy común en la actualidad debido a las ventajas que provee. Según el diccionario de la Real Academia Española (RAE), simular es representar algo, fingiendo o imitando lo que no es [3]. Enfocándose en el área de la ciencia, la simulación es una imitación o representación de las operaciones de un sistema mediante otro proceso más simple. Facilita analizar y estudiar las partes y las características del proceso real que interesan (descartando los que no son considerados necesarios o puedan complicar el desarrollo) sin la necesidad de ejecutarlo físicamente.

La simulación es una herramienta de análisis que permite sacar conclusiones sin necesidad de trabajar directamente con el sistema real que se está simulando. Es especialmente útil cuando no se dispone de dicho sistema o resulta demasiado arriesgado realizar experimentos sobre él [4]. Es el arte y ciencia de crear una representación o sistema para los propósitos de experimentación y evaluación. Permite predecir el comportamiento de los sistemas bajo diversas situaciones reales o previsibles (o lo que es lo mismo, situaciones simuladas) [5].

Luego del análisis previo se puede definir como simulador a una herramienta que permite imitar un sistema o proceso. Reproduce el comportamiento y las respuestas en el tiempo, ante diversas situaciones que se le puedan presentar al proceso real. Los simuladores no obtienen resultados exactos, pero permiten detectar posibles errores que puedan afectar al evento original. Son un entorno interactivo, que permite al usuario modificar parámetros y ver cómo reacciona el sistema ante el cambio producido. Es un aparato que permite la simulación de un sistema, reproduciendo su conducta. No está obligado a reproducir virtualmente todo el sistema real, lo puede hacer por partes o por subprocesos. Además, varios simuladores con características y funcionalidades diferentes pueden pertenecer a un mismo sistema original lo que en distintas fases de su desarrollo.

1.2.1 ¿Cuándo utilizar la simulación?

Principalmente, la simulación es usada cuando desarrollar un sistema o proceso real es difícil. Se puede simular también cuando no se tienen los recursos necesarios para realizar el proceso real o cuando el objetivo es observar o analizar el funcionamiento y comportamiento del mismo en un período de tiempo; para realizar demostraciones o probar alguno de sus procesos.

1.2.2 Ventajas de la simulación

- Una vez construido, el modelo puede ser modificado de manera rápida con el fin de analizar diferentes políticas o escenarios.
- Generalmente es más barato mejorar el sistema vía simulación, que hacerlo directamente en el sistema real.
- Con los modelos de simulación es posible analizar sistemas de gran complejidad y con gran nivel de detalle.
- En algunos casos, la simulación es el único medio para lograr una solución [6].
- Reducir el tiempo de desarrollo de un sistema.
- Minimizar el riesgo de que el sistema se comportará de forma inesperada.
- Diagnosticar problemas [7].

1.2.3 Desventajas de la simulación

- Los modelos de simulación pueden ser costosos y requerir mucho tiempo para desarrollarse y validarse.
- Es difícil aceptar los modelos de simulación.
- Los modelos de simulación no dan soluciones óptimas.
- La solución de un modelo de simulación puede dar al analista un falso sentido de seguridad [6].
- Construir modelos precisa un entrenamiento especial.
- Por lo general son ignorados los factores humanos y tecnológicos [7].

1.3 Simuladores de dispositivos

Los simuladores de dispositivos son aquellos que su modelo representa las características y comportamientos en distintos escenarios, de un dispositivo real. Se pueden clasificar como

simuladores contra-producto puesto que son muy ventajosos para asegurar una correcta funcionalidad de un determinado producto en fase de diseño, y sirven de ayuda durante las pruebas unitarias del mismo. El simulador actúa, frente al producto a verificar, figurando el entorno real en el que se va a encontrar, creando incluso situaciones de “máxima carga” que garanticen su correcta respuesta en casos muy desfavorables [8].

1.3.1 Algunos simuladores de dispositivos

Con el fin de detectar algunos requerimientos necesarios para la implementación del simulador AB Ethernet se estudiaron varios simuladores de otros protocolos. Esto permitió detectar funcionalidades comunes e identificar las interfaces más sencillas. A continuación se mencionan los mismos:

Modsim: está pensado para operar como un esclavo dentro de una red de comunicación Modbus. Soporta la comunicación por puerto serie pero no soporta registros de 32 bit o cadenas de caracteres [9].

Modbus Omni: funciona como un esclavo capacitado para recolectar, interpretar y responder las solicitudes que un maestro le envíe a través del protocolo Modbus Omni. 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.

Dexter: es una herramienta de código abierto desarrollada para la automatización industrial, que proporciona al usuario una interfaz gráfica amigable para realizar simulaciones de esclavos Modbus y DNP3 sobre múltiples conexiones serial y TCP/IP [8].

1.4 Controladores Lógicos Programables (PLC)

Para una mejor comprensión de los dispositivos a simular se hizo un análisis de las principales características de los PLC.

Una de las principales bases de las industrias automatizadas es un dispositivo electrónico llamado Controlador Lógico Programable (PLC) a través del cual garantizan alta confiabilidad, gran eficiencia y flexibilidad en sus sistemas [10]. Un PLC es un dispositivo de estado sólido, diseñado para controlar procesos secuenciales que se ejecutan en un ambiente industrial. Van asociados a la maquinaria que desarrolla procesos de producción y controlan su trabajo [11]. La siguiente imagen representa un ejemplo de PLC.

Figura 1: PLC.

Los PLC brindan numerosas ventajas en procesos de automatización, pues cuentan con los instrumentos adecuados para realizar su desempeño, tales como: relés, temporizadores electrónicos y contadores. Además disponen de facilidades para acceder a subsistemas de comunicaciones. Por sus características de diseño, estos dispositivos tienen un campo de aplicación muy amplio, y la constante evolución del hardware y software aumentan incesantemente la dimensión de este campo.

Un controlador lógico programable se compone de cuatro unidades funcionales tal como se muestra en la siguiente imagen.

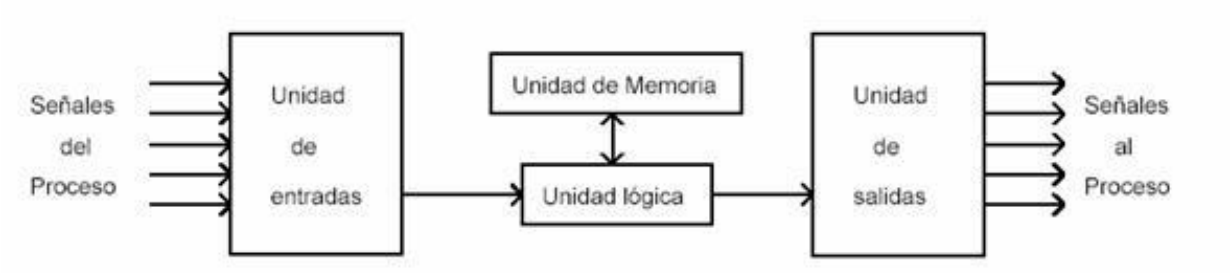


Figura 2: Unidades funcionales de un PLC

Estas unidades son [11]:

- **La unidad de entradas:** proporciona el aislamiento eléctrico necesario del entorno y adecua el voltaje de las señales eléctricas que recibe el PLC que provienen de los interruptores de los contactos. Las señales se ajustan a los niveles de voltaje que marca la Unidad Lógica.

- **La unidad de salidas:** acepta las señales lógicas provenientes de la Unidad Lógica y proporciona el aislamiento eléctrico a los interruptores de contactos que se conectan con el entorno.
- **La unidad lógica:** el corazón de un PLC es la unidad lógica, la cual se basa en un microprocesador. Esta unidad ejecuta las instrucciones programadas en la memoria, para desarrollar los esquemas de control lógico que se han diseñado previamente.
- **La unidad de memoria:** almacena el código de mensajes o instrucciones que tiene que ejecutar la unidad lógica del PLC.

1.4.1 PLC-5

El procesador PLC-5 es el núcleo de la arquitectura de control que combina los sistemas existentes y futuros mediante redes abiertas y conectividad a otros dispositivos. Sirve como núcleo de miles de soluciones de control Allen-Bradley en todo el mundo, cuenta con características como:

- Flexibilidad de programación, conexión de redes, opciones de Entradas/Salidas y selección de controladores.
- Confiabilidad con un valor nominal MTBF (tiempo medio entre fallos) de más de 400,000 horas.
- Compatibilidad con los productos que posee hoy y los nuevos productos introducidos continuamente por Rockwell Automation [12].

Los PLC-5 pueden usarse en un sistema diseñado para control centralizado o en un sistema diseñado para control distribuido.

- Control centralizado es un sistema jerárquico en donde el control sobre todo el proceso está concentrado en un procesador.
- Control distribuido es un sistema en el cual las funciones de control y administración están dispersas a través de la planta. Múltiples procesadores efectúan las funciones de administración y control y usan una red Data Highway +, una red Ethernet, o un sistema bus para comunicación [12].

La arquitectura modular permite que los sistemas automatizados se amplíen según sus necesidades sin sacrificar las inversiones de capital y capacitación.

1.4.2 Mapa de memoria del PLC-5

La memoria en el PLC se divide en la memoria de programas y la memoria de las variables. La memoria de programas contiene las instrucciones que deben ser ejecutadas y generalmente no puede ser cambiada mientras el PLC se encuentra en modo de ejecución. La memoria de las variables cambia constantemente mientras el PLC ejecuta el programa interno, reflejando por un lado los valores obtenidos de las entradas físicas del PLC y por otro los cálculos intermedios y variables de estado del PLC [13].

A diferencia de los PLC ControlLogix del mismo fabricante en que la memoria se organiza por nombres simbólicos (llamados "*tags*" o "alias") en el PLC5, que representa un modelo anterior, la memoria es organizada en bloques de hasta 1000 elementos consecutivos que se denominan "*files*" o ficheros. Existen 8 "*files*" de datos definidos por defecto (ver Figura 2) pero se pueden crear ficheros adicionales si son necesarios por el programa del PLC [13].

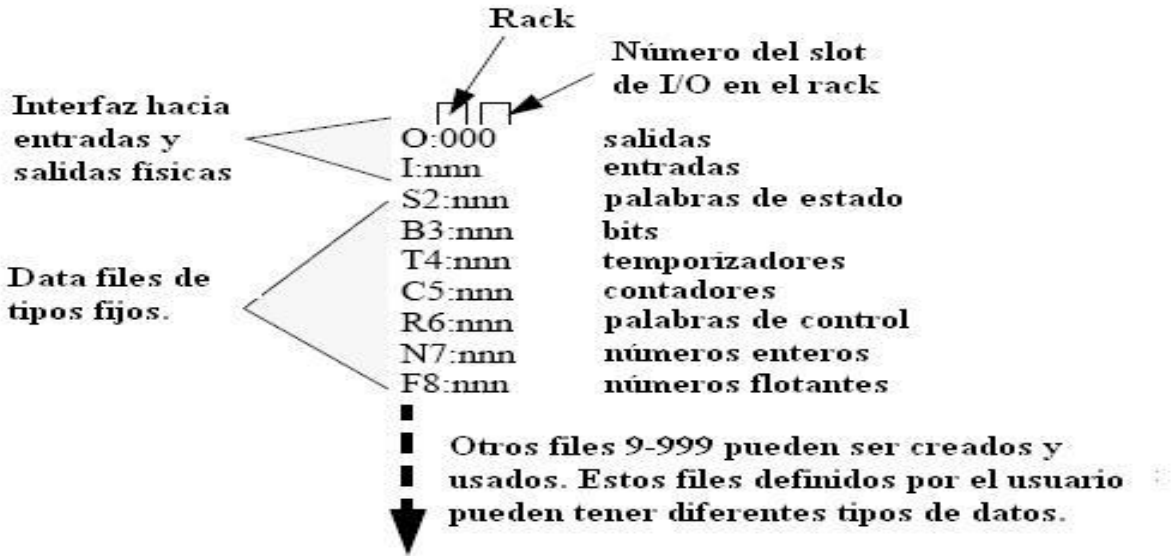


Figura 3: Organización de la Memoria de Variables en el PLC5

1.5 Especificaciones del protocolo AB Ethernet

Los protocolos de comunicación son el conjunto de pautas que posibilitan que distintos elementos que forman parte de un sistema establezcan comunicaciones entre sí, intercambiando información [14].

AB Ethernet es un protocolo de comunicaciones creado para la comunicación con los autómatas Allen Bradley de la firma Rockwell Automation. A través de este protocolo el SCADA se comunica con los autómatas PLC 5 y SLC. Este dialecto también permite acceder a los

valores de las variables a través de nombres simbólicos o "*tags*" que cumplen determinadas reglas sintácticas y semánticas [15].

1.5.1 Formato de direcciones de variables de los dispositivos AB Ethernet

Cada variable se referencia por, como mínimo, 3 elementos. El tipo del *file* que se expresa por la primera letra de la dirección, el número del *file* (que es el número que sigue a esa primera letra) y el desplazamiento de la variable dentro de *file* que se expresa por un número separado de la primera parte de la dirección por el carácter de dos puntos ":". De esta manera direcciones posibles son N27:5 o F41:12 [13].

Entre las reglas sintácticas y semánticas que cumple este protocolo esta por ejemplo la especificación de cómo acceder a los elementos de un arreglo por ejemplo: N10:125 y N10:126 son dos elementos del arreglo de enteros N10.

Por ejemplo en la siguiente tabla se muestran algunas direcciones de variables que son válidas para este protocolo:

Tabla 1. Ejemplo de direcciones válidas

Direcciones	Descripción	Tipo de Dato
F62:11	PRESIÓN AIRE ARRANQUE BOMBA 1	Flotante
F74:11	PRESIÓN AIRE ARRANQUE BOMBA 2	Flotante
N10:125	VELOCIDAD DEL MOTOR1	Entero
N10:126	VELOCIDAD DE REFERENCIA MOTOR1	Entero

1.5.2 Formato de direcciones de los dispositivos AB Ethernet

Se puedan validar las direcciones asociadas a dispositivos AB Ethernet.

El formato de una dirección válida en el AB Ethernet está definida por el parámetro de configuración *remoteIp* que debe ser una dirección Ipv4 por ejemplo: 167.175.164.38 [15].

1.5.3 Creación de bloques en el manejador AB Ethernet

El manejador AB Ethernet desarrollado por el CEDIN permite dividir las variables a administrar o los grupos de ellas en bloques diferentes. 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 que pertenecen a *files* distintos se ordenan en bloques diferentes tanto para lectura, como para escritura. Lo que permite que se limiten los errores pues si

resulta dañado un mapa de memoria o *file* solo se afecta la calidad de la transmisión de datos en el bloque al que este pertenece.

- Los bloques de variables no pueden contener más de 218 bytes [13].

1.5.4 Composición de la trama AB Ethernet

Una trama de datos, básicamente es un paquete de datos (la información se envía por trozos, no entera). Estos paquetes constan de cabecera (donde van los protocolos de enlace), datos (la información) y cola (donde suelen ir un chequeo de errores) [16].

Para delimitar estos paquetes se emplean cuatro métodos:

- Por conteo de caracteres: en la cabecera se pone el número de bytes que compone el paquete.
- Por caracteres de principio y fin: se emplea códigos ASCII (codificación correo) para delimitar el principio y fin. Al principio se pone STX (Inicio de Transmisión) y al final del paquete, ETX (Fin de Transmisión).
- Por secuencias de bits: Se emplea una secuencia de bits para indicar el principio y fin de una trama. Se suele emplear el guión, 01111110.
- Por violación del nivel físico: se trata de introducir una señal, o nivel de señal, que no se corresponda ni con un uno ni con un cero. Por ejemplo si la codificación física es bipolar se puede usar el nivel de 0 voltios, o en Codificación Manchester se puede tener la señal a nivel alto o bajo durante todo el tiempo de bit (evitando la transición de niveles, característica de este sistema) [16].

Realizando un estudio detallado del manejador que utiliza el protocolo en cuestión, a través de la ingeniería inversa se identificó la composición de las tramas. Este es un asunto de vital importancia para la implementación. Porque permite establecer las bases de un adecuado ensamblaje y una correcta interpretación de los mensajes que se intercambian durante la comunicación entre el manejador y el simulador que se desarrolla.

En el caso de las tramas del protocolo de comunicación AB Ethernet se utiliza el conteo de caracteres, donde la cabecera de la trama siempre tendrá un tamaño fijo de 28 bytes y el campo PCCC Length será el que indique el tamaño o el número de bytes que contiene el cuerpo del mensaje. La cabecera tiene la misma estructura, independientemente de si el

mensaje se envía o se recibe, es de lectura, escritura, respuesta o conexión. En la Tabla 2 se pueden observar los campos que componen la cabecera y una breve descripción de ellos.

Tabla 2: Estructura de la cabecera

Nombre del campo	Cantidad de bytes que emplea	Descripción
Mode	1	Descriptor del modo de comunicación que forma parte de la cabecera de la trama.
Submode	1	Submodo de comunicación que forma parte de la cabecera de la trama.
PCCC Length	2	Campo que contiene la longitud en bytes de la porción de datos del mensaje, es decir, la cantidad de bytes que sigue a la cabecera.
Connection	4	Almacena el identificador que utilizarán los mensajes una vez establecida la conexión.
Status	4	Código de estado.
Request ID	4	-
Name ID	4	-
Junk Data	8	Sección que corresponde a datos basura.

El mensaje de conexión solo comprende el envío de la cabecera. Durante el primer intercambio entre el manejador y el dispositivo de campo se establece el valor del campo Connection para el posterior tráfico de mensajes.

Los mensajes de lectura, escritura y el de respuesta contienen la cabecera más el cuerpo del mensaje, este último tiene muchos campos comunes para todos los mensajes pero sufre ligeras variaciones en el final de su estructura dependiendo del tipo de mensaje que se trate.

El mensaje de lectura tiene un cuerpo con tamaño de 18 bytes. En la siguiente tabla se puede observar la descripción de los campos que componen este tipo de mensaje y los bytes que representan cada campo.

Tabla 3: Estructura del cuerpo del mensaje de lectura

Nombre del	Cantidad de	Descripción

campo	bytes que emplea	
Destiny	1	Contiene el campo destino de la trama.
Control	1	Contiene el campo de control de la trama.
Source	1	Identifica la fuente de la trama.
Lsap	1	Contiene el campo lsap de la trama.
Command	1	Especifica el comando que se va a enviar o se recibe.
Status	1	Representa el campo estado del dato.
Tns	2	Indica el resultado de la transacción.
Function	1	Campo que especifica que se trata de un mensaje de lectura.
Offset	2	Indica el lugar a partir del cual se analiza la trama.
Trans	2	Identificador de la transacción.
Init Data	1	-
File	2	Especifica el número del *file* que se procede a leer.
Subelement	1	Establece el número del sub elemento a partir del cual se comienza a leer la información.
Bytecount	1	Contiene la cantidad de bytes que se leerán a partir del sub elemento especificado.

El mensaje de respuesta tiene un tamaño variable siempre a partir de 13 bytes, ya que puede aumentar la cantidad de bytes dependiendo de la cantidad de información que se envíe como respuesta. El campo Data de los mensajes de respuesta es el que contiene el valor de las variables que se leen. Estos mensajes son los que envía el dispositivo de campo. La estructura del cuerpo de estos mensajes se aprecia en la Tabla 4.

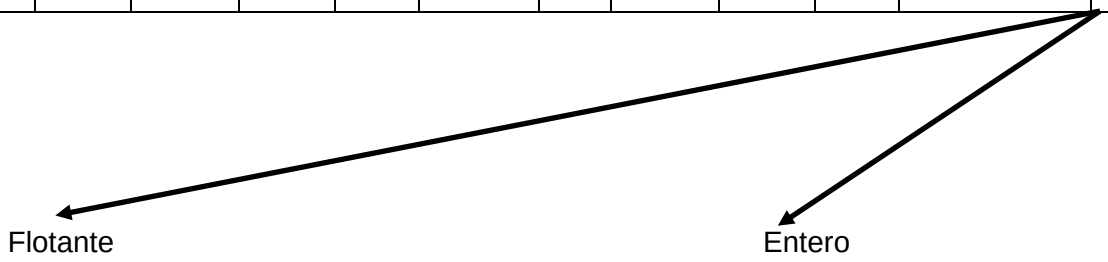
Tabla 4: Estructura del cuerpo del mensaje de respuesta

Destiny	Control	Source	Lsap	Command	Status	Tns	Function	Offset	Trans	Data
1	1	1	1	1	1	2	1	2	2	Varía

El mensaje de escritura tiene un tamaño de 23 bytes para los números flotantes y de 20 bytes para los números enteros. Los números flotantes se dividen en dos números enteros que se almacenan en campos de 2 bytes cada uno (ValuetoWriteFirst, ValuetoWriteSecond). Entonces para obtener el valor real que se ordena escribir se tiene que efectuar una unión de estos campos.

Tabla 5: Estructura del cuerpo del mensaje de escritura: flotantes y enteros

Des	Ctrl	Src	Lsap	Cmnd	Stus	Tns	Fction	Off set	Trans	Init Data	File	Subelement	…
1	1	1	1	1	1	2	1	2	2	1	2	2	…



Nombre	B	Descripción
Valueto WriteFirst	2	Primera parte de los bytes que representan el número flotante.
Valueto WriteSecond	2	Segunda parte de los bytes que representan el número flotante.
ByteValueto WriteFirst	1	Cantidad de bytes que representan el valor a escribir

Nombre	B	Descripción
ValuetoWrite	2	Conforman los bytes que representan el número entero.

1.6 Modelo Maestro/Esclavo

International Business Machines (IBM) define el modelo Maestro/Esclavo como: “La tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo u organización, en múltiples plataformas. El modelo soporta un medio 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" [9].

Figura 4: Modelo Maestro-Esclavo

El maestro es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el maestro. Y el esclavo es cualquier recurso de cómputo dedicado a responder a los requerimientos del maestro [9].

Los servidores esclavos pueden proveer a los servidores maestros de varios servicios tales como impresión, acceso a bases de datos y procesamiento de imágenes.

Durante la investigación este modelo será utilizado para establecer la comunicación donde los dispositivos de campo que se simularán se comportarán como esclavos.

1.7 Comunicación por el protocolo TCP/IP

TCP/IP son las siglas de Protocolo de Control de Transmisión/Protocolo de Internet (en inglés Transmission Control Protocol/Internet Protocol), un sistema de protocolos que hacen posibles servicios Telnet, FTP, E-mail. El Protocolo de Control de Transmisión (TCP) permite a dos anfitriones establecer una conexión e intercambiar datos. El TCP garantiza la entrega de datos,

es decir, que los datos no se pierdan durante la transmisión y también garantiza que los paquetes sean entregados en el mismo orden en el cual fueron enviados. El Protocolo de Internet (IP) utiliza direcciones que son series de cuatro números octetos (byte) con un formato de punto decimal, por ejemplo: 69.5.163.59 [17]. Los datos viajan utilizando como capa de transporte al protocolo TCP y a nivel de red al Protocolo de Internet (IP).

1.8 Herramientas seleccionadas para el desarrollo del simulador

Para darle solución al problema planteado, durante el desarrollo del simulador, se utilizó la biblioteca de transporte desarrollada en la línea de Adquisición del CEDIN TransportProvider2, la cual garantiza el envío y recepción de datos a través de redes Ethernet, sobre protocolos de nivel de transporte TCP/IP, UDP/IP o seriales. Contiene un conjunto de funcionalidades que permiten la abstracción en el intercambio de información entre diferentes nodos dentro de una red.

A continuación se describen las herramientas libres que fueron seleccionadas para lograr el desarrollo del simulador.

1.8.1 Qt

Qt es un *framework* para el desarrollo de aplicaciones multiplataforma, el cual utiliza el lenguaje C++ de forma natural y garantiza una excelente documentación. Es una biblioteca usada para desarrollar interfaces gráficas de usuarios, así como programas sin interfaz gráfica, aplicaciones de consola y servidores. [18].

1.8.2 Qt Creator

Qt Creator es un entorno de desarrollo integrado (IDE) que proporciona herramientas para diseñar y desarrollar aplicaciones con el *framework* de aplicaciones Qt. Está diseñado para desarrollar aplicaciones e interfaces de usuario y desplegarlas a través de varios sistemas operativos móviles y de escritorio [19].

1.8.3 Lenguaje de programación C++

C++ es un lenguaje de propósito general basado en el C, al que se han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones inline, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería [20]. Tiene como ventajas: que es un lenguaje de programación orientado a objetos y es muy potente en lo que se refiere a creación

de sistemas complejos, un lenguaje muy robusto. Tiene un conjunto completo de instrucciones de control, permite la agrupación de instrucciones y emplea la programación de bajo nivel [21].

1.8.4 Visual Paradigm 8.0

El modelado de la solución se llevó a cabo mediante la herramienta CASE (Ingeniería de Software Asistida por Computación) Visual Paradigm, la cual propicia el desarrollo de aplicaciones utilizando modelado UML; ya que garantiza la generación de código fuente de los programas y la generación de documentación, esta admite el ciclo de vida del desarrollo de un sistema a través de la representación de diferentes diagramas. Esta herramienta se encuentra disponible en múltiples plataformas (Windows, Linux). Se identifica por no tener complejidad en su instalación, poseer licencia gratuita y comercial y ser compatible entre ediciones.

1.8.5 Lenguaje de Modelado Unificado (UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que es usado para especificar, visualizar, construir y documentar artefactos de un sistema [22]. Es una técnica de modelado de objetos que persigue abstraer cualquier tipo de sistema, mediantes diagramas (de implementación, de comportamiento o interacción, de casos de uso, de clases, entre otros), los cuales son representaciones gráficas que contienen toda la información notable del sistema. Se utiliza para especificar o describir métodos o procesos, es el lenguaje que se usa para describir un modelo.

1.9 Metodología de desarrollo de software a utilizar

La UCI cuenta con 14 centros productivos en su gran mayoría pertenecientes a las facultades que conforman a la Universidad. Cada uno de estos centros se dedica al desarrollo de software y/o servicios asociados. Todos los desarrollos se encuentran organizados en proyectos, clasificados en: Gestión, Componentes y tecnología base, Portales y Sistemas operativos. Esta diversidad de centros y proyectos trae consigo la heterogeneidad en el proceso de desarrollo de software [23].

Por lo tanto la Universidad decide adaptar una metodología para establecerla como estándar en todos sus centros productivos. La metodología escogida fue el Proceso Unificado Ágil (AUP) que es una versión simplificada del Proceso Unificado de Desarrollo (RUP). Esta 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. Para adaptar esta metodología a las características de la Universidad se le realizaron variaciones y

se confeccionó la metodología AUP-UCI, que es la utilizada durante el transcurso de esta investigación.

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 [23]. A continuación se refiere acerca de las fases y las disciplinas de esta variación de la metodología AUP.

1.9.1 Fases AUP-UCI

Tabla 6. Fases AUP-UCI

Fases	Objetivos [23]
Inicio	• Llevar a cabo las actividades relacionadas con la planeación del proyecto. • Realizar un estudio inicial de la organización cliente. • Realizar estimaciones de tiempo, esfuerzo y costo.
Ejecución	• Ejecutar las actividades requeridas para desarrollar el software. • Ajustar los planes del proyecto considerando los requisitos y la arquitectura. • Modelado del negocio. • Obtener los requisitos. • Elaborar la arquitectura y el diseño. • Implementar y liberar el producto. • Transferir el producto al cliente. • Capacitar a los usuarios finales sobre la utilización del software.
Cierre	• Analizar los resultados del proyecto y su ejecución • Realizan las actividades formales de cierre del proyecto.

1.9.2 Disciplinas AUP-UCI

Tabla 7. Disciplinas AUP-UCI

Disciplinas	Objetivos [23]

Modelado de negocio (opcional)	Comprender los procesos de negocio
Requisitos	Administrar y Gestionar los requisitos funcionales y no funcionales.
Análisis y diseño	Modelar el sistema.
Implementación	Construir el sistema.
Pruebas interna	Verificar el resultado de la implementación.
Pruebas de liberación	Diseñar y Ejecutar pruebas por una entidad Externa Certificadora de la calidad.
Pruebas de Aceptación	Verificar que el software está listo.
Despliegue (Opcional)	Instalar, Configurar, Adecuar, y Poner en marcha.
Gestión y soporte	-

1.10 Conclusiones parciales

En este capítulo se realizó una investigación acerca de la simulación de dispositivos, cuando utilizarla, sus principales ventajas y desventajas. Al analizar los dispositivos PLC-5 se detectaron características que tendrán que estar incluidas en la solución, tales como el mapa de memoria y el protocolo AB Ethernet. Se establecieron las herramientas y la metodología de desarrollo a ser empleadas para la implementación de la solución.

Capítulo #2. Características y diseño del sistema

2.1 Introducción

En este capítulo se abordan los rasgos de la propuesta de solución del sistema, así como el análisis y diseño de la aplicación. Se describen las arquitecturas que se utilizarán y se definen los requisitos funcionales y no funcionales con los que debe contar el sistema. Además se muestran los diagramas de clases asociados a la solución.

2.2 Modelo del dominio

El modelo de dominio se realiza con el objetivo de comprender y describir las clases más importantes dentro del contexto del sistema y, por tanto, también contribuir a la comprensión de los requisitos de dicho sistema que se desprenden de este contexto [24]. El modelo de dominio representa clases conceptuales del mundo real y no de elementos del programa, este ayuda a comprender los conceptos con que trabajará la aplicación.

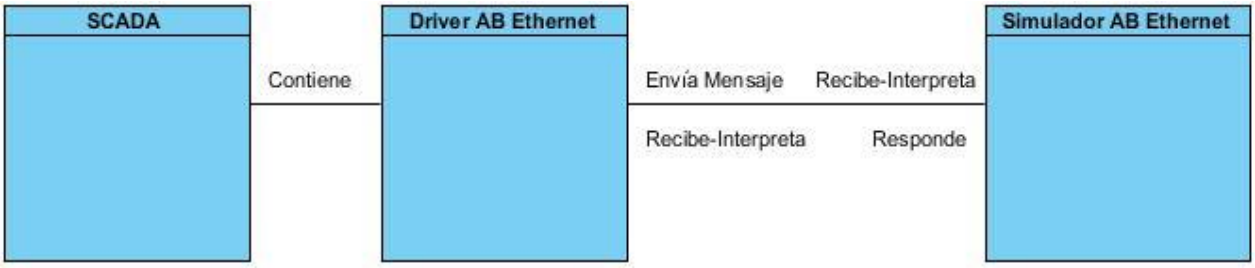


Figura 5: Modelo del Dominio

El SCADA contiene múltiples *drivers* dependiendo del dispositivo o simulador al que esté conectado. En el contexto en el que se trabaja se trata del manejador AB Ethernet. Este envía mensajes que pueden ser de conexión, lectura o escritura. El simulador es quien recibe estos mensajes y captura la información que le es solicitada. Luego envía la respuesta, mediante un nuevo mensaje, con los datos que le fueron solicitados. El *driver* recibe esta respuesta y es quien se encarga de interpretarla para el SCADA. Estos mensajes son encapsulados y enviados por el protocolo de comunicación industrial AB Ethernet.

2.3 Requisitos del sistema

A continuación se listan los requisitos que el sistema debe cumplir, teniendo en cuenta que los requisitos funcionales estarán expresados y presentados mediante las Historias de Usuario (HU). Las HU son descripciones cortas redactadas en el lenguaje del usuario, donde este determina su punto de vista respecto a las necesidades de la aplicación.

2.3.1 Requisitos funcionales del sistema. Historias de Usuarios.

Tabla 8. Historia de Usuario 1

Historia de Usuario	
Número: 1	**Usuario:** Línea de Adquisición
Nombre de historia: Configurar canal TCP.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** baja
Puntos estimados: 0.2	**Iteración asignada:** 2
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se debe brindar la opción visual al usuario de establecer el puerto que utilizará el canal de comunicación TCP para establecer la comunicación.	
Observaciones:	

Tabla 9. Historia de Usuario 2

Historia de Usuario	
Número: 2	**Usuario:** Línea de Adquisición
Nombre de historia: Gestionar los mapas de memoria.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se debe brindar la opción visual al usuario de Crear, Modificar y Eliminar uno o varios mapas de memoria de tipo entero o flotante.	
Observaciones:	

Tabla 10. Historia de Usuario 3

Historia de Usuario	
Número: 3	**Usuario:** Línea de Adquisición
Nombre de historia: Simular valores en los mapas de memoria.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se debe brindar la opción visual de generar valores aleatorios, crecientes o decrecientes en las	

direcciones del mapa de memoria. Debe permitir simular valores de forma general con la misma configuración en todo el mapa, o de forma individual para cada dirección.
Observaciones:

Tabla 11. Historia de Usuario 4

Historia de Usuario	
Número: 4	**Usuario:** Línea de Adquisición
Nombre de historia: Simular errores de tramas.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se debe brindar la posibilidad al usuario de activar y desactivar la opción de generar ruido, es decir, enviar tramas incorrectas o con datos erróneos. Una vez que se desactive esta opción el simulador tiene que retomar el envío de tramas sin errores.	
Observaciones:	

Tabla 12. Historia de Usuario 5

Historia de Usuario	
Número: 5	**Usuario:** Línea de Adquisición
Nombre de historia: Interpretar y responder mensajes de lectura enviados por el manejador.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: El sistema debe ser capaz de recibir los mensajes de lectura, interpretarlos correctamente para identificar los datos que debe enviar en la respuesta. Una vez definidos los datos de respuesta, tiene que ser capaz de ensamblar correctamente un mensaje de respuesta y enviarlo al manejador.	
Observaciones:	

Tabla 13. Historia de Usuario 6

Historia de Usuario	
Número: 6	**Usuario:** Línea de Adquisición
Nombre de historia: Interpretar y ejecutar mensajes de escritura enviados por el manejador.	

Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: La aplicación debe ser capaz de recibir los mensajes de escritura, interpretarlos correctamente para identificar los datos que debe modificar en el mapa de memoria, dígase dirección de memoria y su valor. Una vez definidos estos tiene que ser capaz de actualizar correctamente la dirección que se ordena modificar con los valores enviados en el mensaje.	
Observaciones:	

Tabla 14: Historia de Usuario 7

Historia de Usuario	
Número: 7	**Usuario:** Línea de Adquisición
Nombre de historia: Visualizar los mensajes enviados y recibidos por el dispositivo.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: El simulador debe representar mediante una interfaz gráfica el comportamiento del envío y la recepción de los mensajes, así como la estructura de los mensajes.	
Observaciones:	

2.3.2 Requisitos no funcionales (RNF)

Los RNF definen criterios que pueden emplearse para calificar la operación de un sistema en lugar de sus conductas específicas. Son todas las peculiaridades del sistema que no representan información a almacenar, ni funciones a realizar. Son propiedades o características que hacen al producto atrayente, usable, ligero y confiable.

Requerimientos de software:

- Funciona en el Sistema Operativo GNU/Linux (distribución Debian Wheezy).
- Uso del *framework* Qt.
- Instalación de la biblioteca de transporte TransportProvider2.

Requerimientos del hardware:

Debe ser ejecutado en computadoras que tengan como mínimo:

- Microprocesador Pentium 4 o superior.
- RAM: 512 Mb.

Requerimientos de usabilidad:

- Debe ser confiable y estable.
- Poseer una interfaz de gráfica amigable e intuitiva.
- Debe ser utilizado por cualquier usuario con conocimientos elementales de computación.

2.4 Descripción del sistema propuesto

El sistema propuesto será una aplicación informática de escritorio, la cual imitará el funcionamiento de un PLC. Estableciendo comunicación con el *driver* mediante el protocolo AB Ethernet y generando los datos que le sean solicitados. De la misma forma podrá activar errores en las comunicaciones representando una actividad similar a la de un PLC en funcionamiento.

2.5 Arquitectura del sistema

La arquitectura de software juega un papel fundamental en el diseño y la implementación de estructuras de software de alto nivel [25]. Esta resulta de unir elementos arquitectónicos para satisfacer las funcionalidades y los requerimientos de un sistema determinado. Los patrones de arquitectura son un ejemplo de los elementos antes mencionados, estos ofrecen una organización estructural y un conjunto de restricciones sobre cómo deben ser usados, permitiendo además que los miembros del equipo de desarrollo observen el software de una manera similar.

Para darle solución al problema planteado se decidió utilizar la arquitectura n capas, debido a la complejidad del sistema, para separar la lógica de la comunicación del manejo, la simulación y visualización de los datos; ya que cada capa puede ser modificada tanto como sea necesario sin provocar cambios en las demás. Una capa no tiene dependencias con la capa superior, cada capa depende solamente de la fachada que le permite comunicarse con la capa inmediata inferior. Esta dependencia entre capas es normalmente a través de fachadas, asegurando que el acoplamiento sea el más bajo posible y la abstracción del funcionamiento de la capa inferior, sea casi total. Esto permite el desarrollo de aplicaciones más robustas debido al encapsulamiento, lo que posibilita un mantenimiento y soporte más sencillo así como una mayor flexibilidad [26].

El sistema comprende tres capas: Aplicación, Protocolo y Transporte. En el siguiente diagrama se muestra su jerarquía.

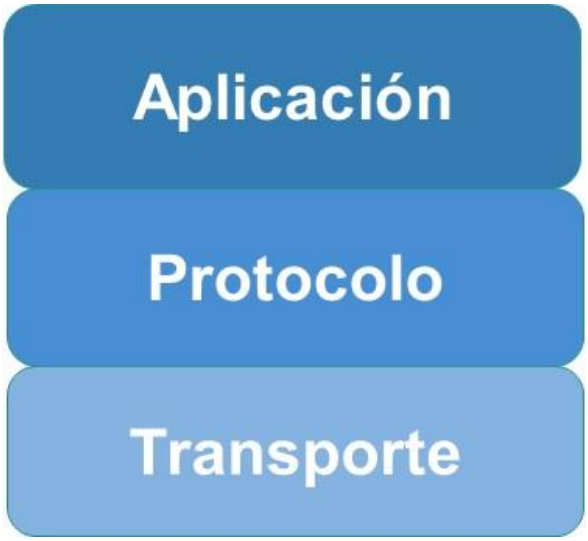


Figura 6: Arquitectura del simulador AB Ethernet

2.5.1 Capa Aplicación

Esta capa tiene como función el manejo, la simulación y la visualización de los datos. Para ello hace uso del patrón arquitectónico Modelo-Vista-Controlador (MVC). El mismo separa los datos de la aplicación (modelo), la interfaz de usuario (vista), y la lógica de control (controlador) en tres módulos disímiles; actuando el controlador como intermediario del flujo de datos entre el modelo y la vista.

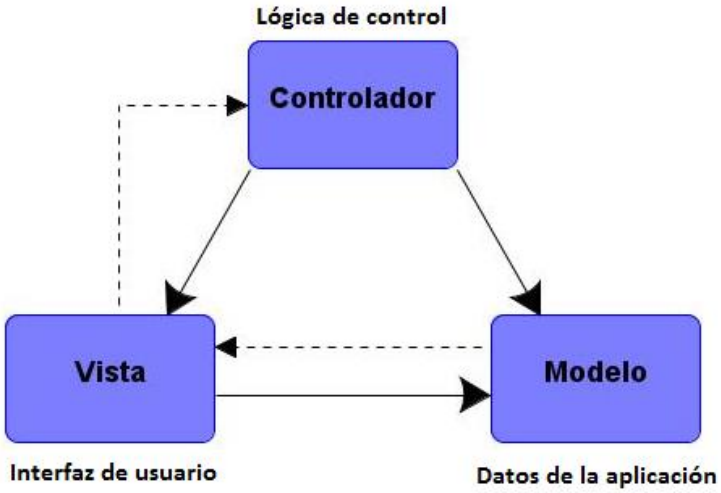


Figura 7: Patrón Modelo-Vista-Controlador

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar [27]. En la solución estará compuesto por clases, que permitirán manejar los datos de la aplicación.

Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo [27]. Para interactuar con el usuario, utilizará una interfaz gráfica, que tendrá las opciones demandadas por el cliente.

El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas [27]. La lógica de control del sistema se lleva a cabo en la creación, modificación y eliminación de los mapas de memoria.

2.5.2 Capa Protocolo

Tiene como objetivo garantizar la lógica de comunicación entre el maestro que solicita los servicios y el simulador que actúa como esclavo. Se encarga de iniciar la conexión, recibir, interpretar y enviar mensajes, así como de ensamblarlos, ya sean de escritura o conexión.

2.5.3 Capa Transporte

Comprende a la biblioteca TransportProvider2, la cual garantiza la lectura y escritura de los mensajes. La información es recibida en forma de *callback* a través de métodos específicos proporcionados por la misma; estos 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 y se denominan métodos 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.

2.6 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Son

soluciones a diferentes clases de problemas conocidos, que pueden ser aplicadas a diferentes códigos [8].

2.6.1 Patrones GRASP

GRASP es el acrónimo de *General Responsibility Assignment Software Patterns* (patrones generales de software para asignar responsabilidades). Aunque se considera más bien que son una serie de buenas prácticas para la aplicación en el software. Los patrones de GRASP, son una guía para encontrar los patrones de diseño (que son más concretos) [28].

A continuación se describen los patrones utilizados acorde con la solución que brindaron en la composición del sistema:

- **Experto:** la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada [28]. Su uso se evidencia en la clase Message, la cual se encuentra en la capa Protocolo, y tiene como función ensamblar los mensajes ya sean de conexión o de respuesta.

- **Bajo Acoplamiento:** debe haber pocas dependencias entre las clases, es decir, que cada una se comunique con el menor número de clases que sea posible. Se puede apreciar en la relación entre las capas Aplicación y Protocolo.

- **Alta Cohesión:** cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Ejemplos de buen diseño se producen cuando se crean las clases agrupadas por funcionalidades que son fácilmente reutilizables (bien por uso directo o por herencia) [28]. Se utiliza dentro de las capas Aplicación y Protocolo.
- **Controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades [28]. Se ve presente en la clase MemoryManager la cual se encarga de controlar el flujo de datos entre el modelo y la vista.
- **Creador:** Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

    - B contiene a A.
    - B es una agregación (o composición) de A.

- B almacena a A.
- B tiene los datos de inicialización de A (datos que requiere su constructor).
- B usa a A [28].

Por ejemplo la clase FileWidget que contiene un objeto de tipo MemoryModel, que es el modelo de cada variable.

2.6.2 Patrones GoF

Los patrones GoF (Gang of Four) son soluciones que corrigen problemas comunes que se presentan en el diseño, estos describen 23 patrones, que se dividen en tres categorías según su propósito:

- Creacionales: solucionan problemas de creación de instancias. Estos ayudan a encapsular y abstraer dicha creación.
- Estructurales: solucionan problemas de composición de clases y objetos.
- Comportamiento: ofrecen soluciones respecto a la interacción y responsabilidades entre clases y objetos.

En la solución se emplea el patrón creacional Singleton que también se conoce como Instancia única, su objetivo es restringir la creación de objetos pertenecientes a una clase, de modo que solo se tenga una única instancia de la clase para toda la aplicación, garantizando así un punto de acceso global al objeto creado [29].

Este patrón en el simulador, garantiza que los mapas de memoria sean únicos, es decir, el número del *file* es único ya sea entero o flotante. Para lograr su implementación, la clase MemoryManager contiene un constructor privado que solo será accedido desde la misma clase, se crea también una instancia privada de la clase, así como un método estático, el cual crea una instancia de la clase solo en el caso de que no exista. Luego devuelve la instancia creada o, sino la creó porque ya existía, devuelve la existente.

2.7 Diagramas de clases

Los diagramas de clases describen la estructura de un sistema mostrando sus clases, las relaciones, atributos y funcionalidades que posee. Los mismos permiten una mayor comprensión de la solución propuesta. Se da luego de cada diagrama una descripción de cada clase y de las relaciones entre ellas.

2.7.1 Relación entre las clases de la capa Aplicación.

A continuación se muestran los diagramas de clase de la capa Aplicación basados en el patrón arquitectónico Modelo-Vista-Controlador.

2.7.1.1 Diagrama de las clases del Modelo

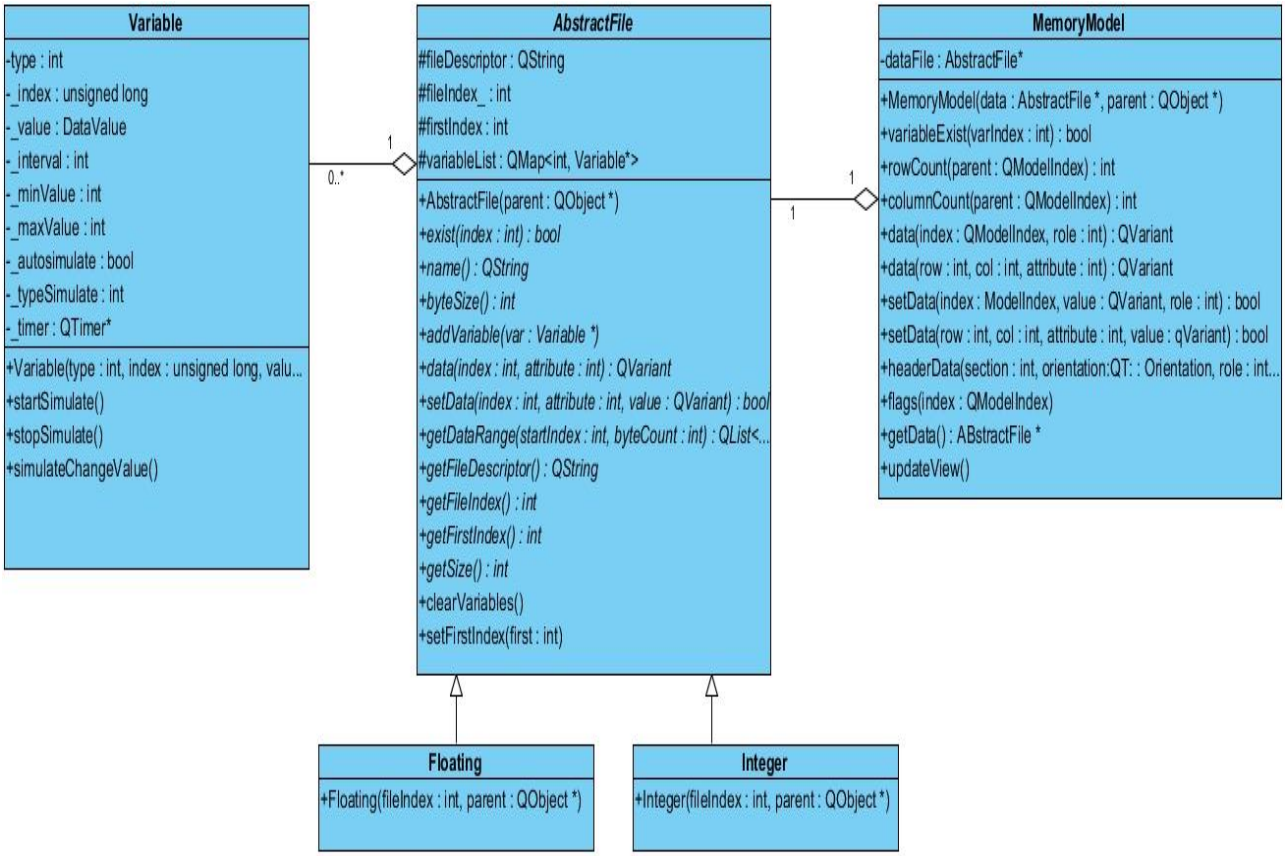


Figura 8: Diagrama de clases del Modelo

Descripción de las clases:

- **Variable:** representa una variable simple en el simulador. Su método principal es simulateChangeValue cuya función recae en implementar la autosimulación de valores en las variables, sea de forma creciente, decreciente o aleatoria verificando el tipo de dato correcto para cada variable. Esta clase colabora con AbstractFile.

- **AbstractFile:** clase abstracta en forma de mapa de memoria que agrupa un conjunto de variables del mismo tipo de fileDescriptor (entero N, flotante F), fileIndex (el número después del tipo de variable) y con un rango que va desde el firstIndex hasta el *size*. Ejemplo: cuando se crea un mapa de memoria que va desde N7:1 hasta N7:10. Esta clase colabora con MemoryModel.

- **Floating:** hereda de AbstractFile, esta clase agrupa un conjunto de variables flotantes.
- **Integer:** hereda de AbstractFile, esta clase agrupa un conjunto de variables enteras.
- **MemoryModel:** representa el modelo para cada mapa de memoria que se crea, este modelo se enlaza a una vista y a través de este accede la vista al mapa de memoria correspondiente. Sus métodos más importantes son: data que devuelve el valor de una variable, setData cambia el valor de una variable dado el índice y el valor, variableExist que dado un índice indica si existe el AbstractFile.

2.7.1.2 Diagrama de las clases de la Vista

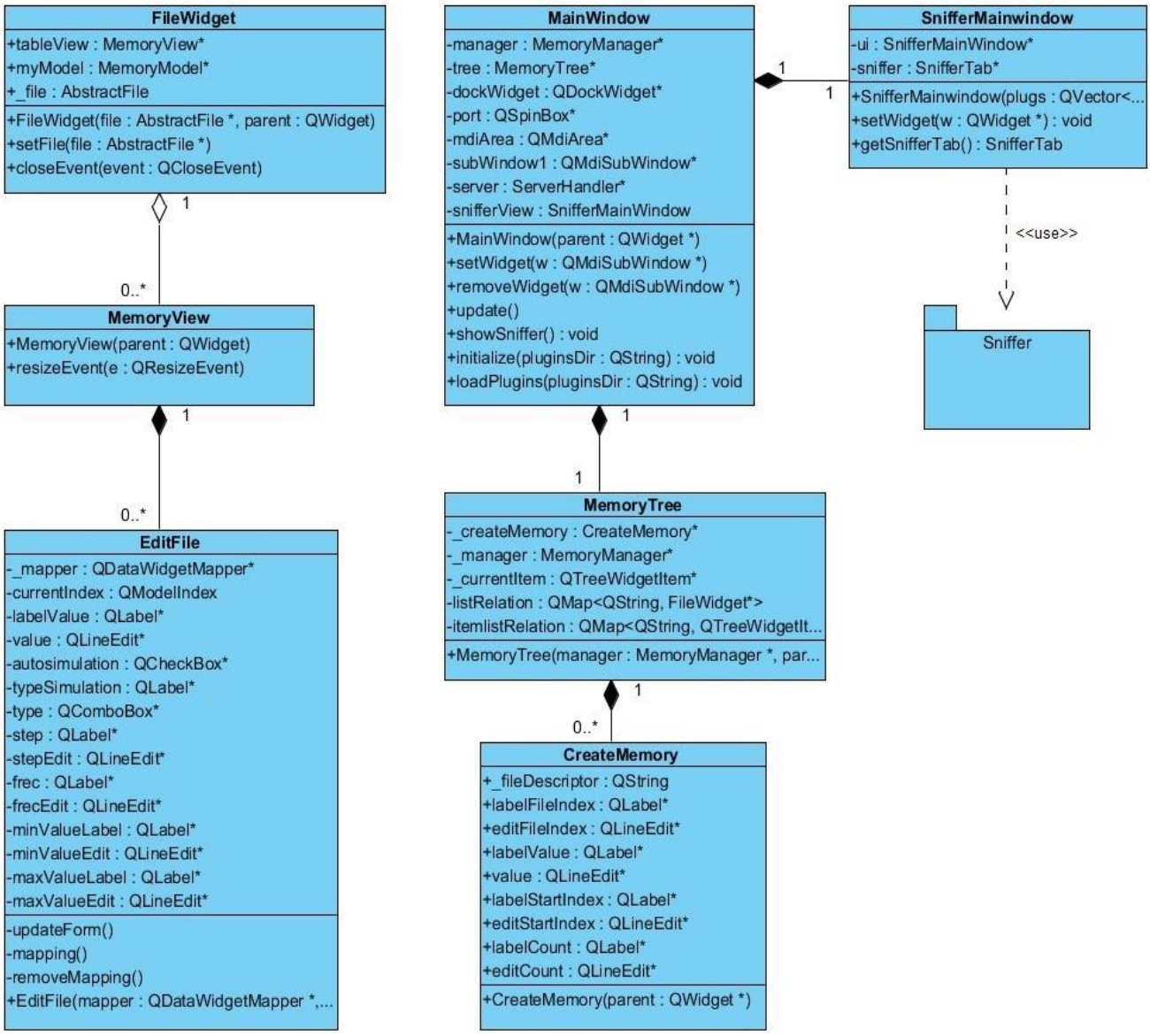


Figura 9: Diagrama de clases de la Vista

Descripción de las clases:

- **MainWindow:** es la ventana principal del simulador, contiene todas las interfaces del mismo.
- **SnifferMainWindow:** visualización de las tramas enviadas y recibidas por el dispositivo.
- **MemoryView:** representación de la vista en forma de tabla que se relaciona con el modelo MemoryModel.

- **MemoryTree:** es la representación en forma de árbol de los nodos de memoria que se creen en el simulador.
- **CreateMemory:** clase que crea la ventana donde se llenan los atributos necesarios para crear un hijo de AbstractFile.
- **EditFile:** clase que crea la ventana donde se modifican los atributos de una variable señalada. Estos atributos pueden ser el valor, si es autosimulada y los parámetros de esta simulación.
- **FileWidget:** completa el concepto del mapa de memoria del simulador, con todos sus componentes, vista, modelo y grupo de variables que conforman el mapa de memoria

2.7.1.3 Clase controladora

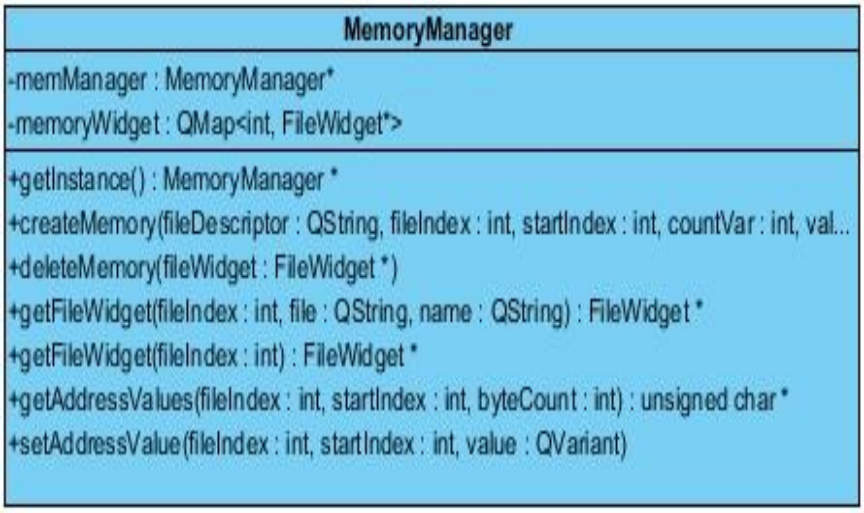


Figura 10: Clase controladora

Descripción de la clase:

MemoryManager: encargada de controlar todos los mapas de memoria que sean creados en el simulador. Tiene la capacidad de eliminar y crear los mapas de memoria. Permite al protocolo obtener los valores dadas las direcciones de protocolo así como modificar estos valores dependiendo de la operación que se desee. Entre sus métodos cuenta con el createMemory que consiste en crear o modificar un mapa de memoria dependiendo del fileDescriptor y el fileIndex, si ya existe otro mapa donde coincidan estos dos parámetros lo que hace es modificarlo con los nuevos valores de los demás parámetros de este método. El método deleteMemory elimina la memoria. El método getAddressValues tiene como función obtener el

valor correspondiente a la variable especificada del mapa de memoria y setAddressValue es el que se encarga de escribir en el mapa de memoria el valor que se mande a escribir desde el manejador.

2.7.2 Relación entre las clases de la capa Protocolo

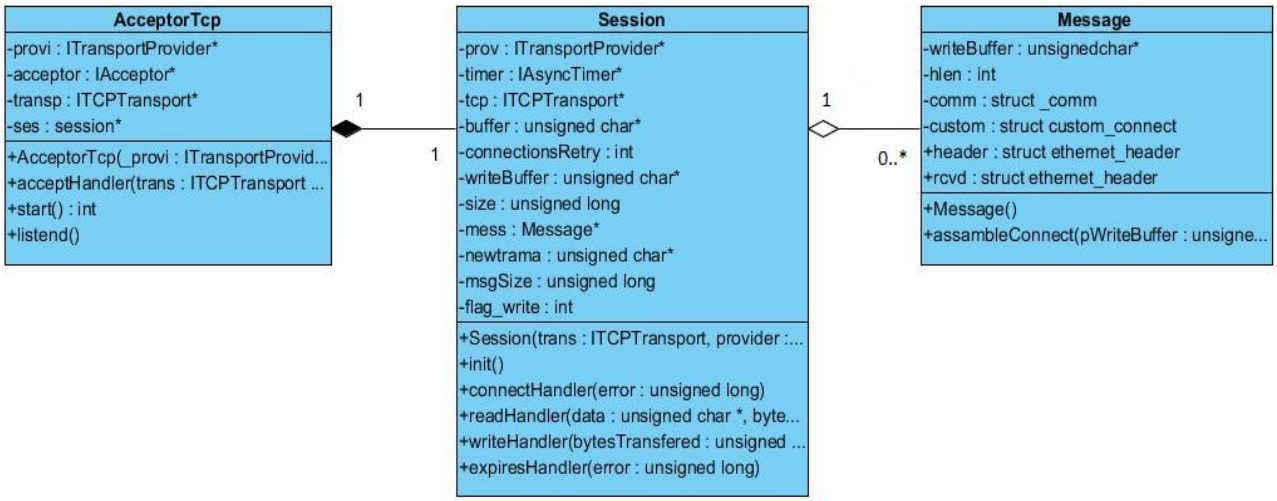


Figura 11: Diagrama de clases de la capa Protocolo

Descripción de las clases:

- **AcceptorTcp:** inicia la conexión por un puerto especificado y la acepta, mediante el método listend. Crea la sesión a través del método acceptHandler. Utiliza la biblioteca TransportProvider2.

- **Session:** se encarga de leer y responder los mensajes. Siendo su principal método readHandler el cual lleva a cabo las acciones anteriormente mencionadas. Utiliza la biblioteca trasnportProvider2.

- **Message:** encargada de ensamblar los mensajes. Utiliza las estructuras custom_connect (representa Buffers para la conexión), ethernet_header (representa Buffers para la escritura y lectura de tramas), _comm (almacena los datos referentes al estado de la conexión tcp con el dispositivo). Esta clase cuenta con el método assambleConnect que es el encargado de armar la trama de respuesta al mensaje de conexión.

2.8 Conclusiones parciales

En este capítulo se identificaron los requisitos funcionales y no funcionales del sistema. Se definió la arquitectura, así como los principales patrones de diseño. Se propuso como solución una aplicación que permite simular comportamientos de los dispositivos PLC-5 y establecer comunicación mediante el protocolo AB Ethernet. Además se realizó el diseño del sistema mediante los diagramas de clases.

Capítulo #3. Implementación y prueba del sistema

3.1 Introducción

Como resultado del análisis y diseño de la solución, se procede a la implementación del sistema. Debido a esto es de vital importancia que se satisfagan las expectativas del usuario final, de forma tal que la aplicación cumpla y funcione de acuerdo a los requerimientos. Por lo que es necesario entonces la realización de pruebas al sistema para garantizar la eficiencia y calidad del mismo. En el presente capítulo se presentan, además de los aspectos mencionados, aquellos vinculados con los estándares de codificación que permiten crear un código fácil de comprender y las tareas de ingeniería relacionadas con cada iteración establecida.

3.2 Implementación

Esta fase se puede ver como un proceso de fabricación, en la cual el modelo del sistema se convierte en código y luego este se transforma en un sistema ejecutable. Durante la misma se procede a implementar las HU descritas en el capítulo anterior a través de las tareas de ingeniería correspondientes a cada una de ellas, ya que estas son un conjunto de acciones asociadas a las HU que permiten organizar el proceso de implementación.

3.2.1 Estándares del código

Entre las buenas costumbres que facilitan la tarea de asegurar la calidad del software se encuentra la utilización de estilos y estándares de codificación. El uso de estos tiene varias ventajas entre ellas lograr un estilo de código homogéneo, facilitando su legibilidad y proporcionando una guía para el encargado del mantenimiento o actualización del sistema, con código claro y bien documentado. También ayuda a mejorar el proceso de codificación haciéndolo eficiente y en muchos casos reutilizable.

La codificación del sistema debe cumplir con ciertos requisitos, algunos de los cuales son detallados seguidamente:

- Los nombres de las variables y de las funciones deben utilizar la notación lowerCamelCase.
- Las variables deben ser siempre elocuentes y prácticas.

3.2.2 Iteración 1

Su objetivo es cumplir con la Historia de Usuario 2. Para esto se asignaron un total de dos tareas de ingeniería a los programadores del sistema.

3.2.2.1 Tareas de Ingeniería realizadas en la primera iteración

Tabla 15: Tarea de Ingeniería 1

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 2
Nombre de la tarea: Implementación de las clases del modelo y de la clase controladora.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16/2/2015	**Fecha fin:** 21/2/2015
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se implementan las clases que representan los modelos de los mapas de memoria y las variables o direcciones individuales, seguidamente se implementa la clase controladora MemoryManager con las funcionalidades que le permitan encargarse de controlar los mapas de memoria, es decir, crearlos, modificarlos y eliminarlos.	

Tabla 16: Tarea de Ingeniería 2

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 2
Nombre de la tarea: Diseño e implementación de la interfaz visual para la gestión de los mapas de memoria.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 23/2/2015	**Fecha fin:** 28/2/2015
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se diseña e implementa la interfaz visual que se encuentra representada por las clases de la vista, siendo la principal, la clase MainWindow donde se integra el resto de los componentes visuales que permiten la gestión de la memoria a un nivel gráfico, utilizando la clase MemoryManager previamente creada.	

3.2.3 Iteración 2

En esta iteración se realizan dos tareas de ingeniería para dar cumplimiento a las Historias de Usuario 1, 5 y 6.

3.2.3.1 Tareas de Ingeniería realizadas en la segunda iteración

Tabla 17: Tarea de Ingeniería 3

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 5 y 6
Nombre de la tarea: Implementación de las clases Session y Message.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 2/3/2015	**Fecha fin:** 12/3/2015
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se implementan las clases Session y Message, que son las encargadas de recibir los mensajes e interpretarlos, de ensamblar y enviar las respuestas.	

Tabla 18: Tarea de Ingeniería 4

Tarea de Ingeniería	
Número de tarea: 4	**Número de Historia de Usuario:** 1
Nombre de la tarea: Insertar componente visual para la configuración del canal TCP.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se inserta un SpinBox en la interfaz principal del simulador, para capturar el número del puerto del canal TCP.	

3.2.4 Iteración 3

Para cumplir con las Historias de Usuario 3 y 4, en esta iteración se realizan tres tareas de ingeniería.

3.2.4.1 Tareas de Ingeniería realizadas en la tercera iteración

Tabla 19: Tarea de Ingeniería 5

Tarea de Ingeniería	
Número de tarea:	**Número de Historia de Usuario:** 3
Nombre de la tarea: Implementación de la simulación de valores en el mapa de memoria.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio:	**Fecha fin:**
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se implementa en la clase Variable. Su activación en el visual se puede realizar seleccionando la opción cuando se crea o modifica un mapa de memoria, o cuando se escoge una dirección específica y realizando doble clic sobre ella, se selecciona la opción de autosimulación en la ventana que se genera.	

Tabla 20: Tarea de Ingeniería 6

Tarea de Ingeniería	
Número de tarea:	**Número de Historia de Usuario:** 4
Nombre de la tarea: Implementación de la simulación de errores en las tramas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio:	**Fecha fin:**
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se implementa en la clase Session y se incluye una opción en la barra de herramientas de la interfaz principal para que el usuario lo pueda activar fácilmente.	

Tabla 21: Tarea de Ingeniería 7

Tarea de Ingeniería	
Número de tarea:	**Número de Historia de Usuario:** 7
Nombre de la tarea: Implementar la clase SnifferMainWindow.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: José Raúl Meriño Maceira, Enrique Yecier Pardo Vega	
Descripción: Se implementa la clase SnifferMainWindow, que es el componente visual que utiliza el paquete Sniffer, para representar los mensajes enviados y recibidos, y la estructura de los mismos.	

3.2.5 Diagrama de despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución como un computador, un dispositivo o memoria [30].

A continuación se muestra el diagrama de despliegue modelado:

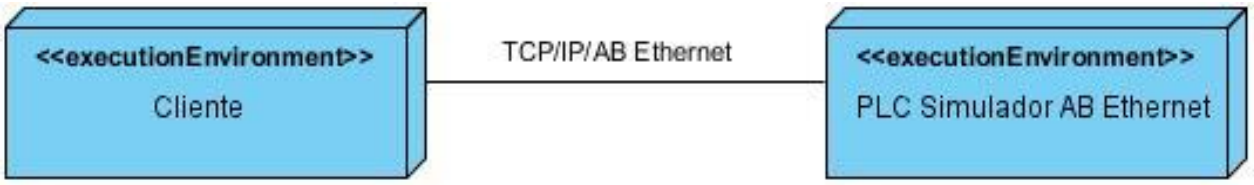


Figura 12: Diagrama de Despliegue

EL nodo Cliente, representa el ordenador donde se ejecuta el manejador AB Ethernet. El nodo PLC Simulador AB Ethernet, simboliza el ordenador donde se ejecuta el software desarrollado. La comunicación entre los nodos se realiza a través del protocolo TCP/IP y el envío y recepción de mensajes está basado en el protocolo de comunicación industrial AB Ethernet.

3.3 Pruebas

Las pruebas son un instrumento para determinar la calidad de un producto o software. Estas siguen un proceso donde se comprueban componentes del software o el sistema en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos y detectar posibles errores que tenga el resultado final.

La metodología AUP-UCI separa la fase de pruebas en tres disciplinas [23]:

- **Pruebas internas:** se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se generan como artefactos los diseños de casos de prueba.
- **Pruebas de liberación:** pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

- **Pruebas de aceptación:** es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Para validar la correcta funcionalidad del sistema y su óptima calidad se realizaron pruebas de aceptación. Estas incluyen la búsqueda de defectos, la validación de que el sistema funciona tal como está previsto, certificando que se cumplan los requisitos solicitados por el cliente.

3.3.1 Casos de prueba

Los casos de prueba abarcan todas las funciones que el programa es capaz de realizar (o debería ser capaz de realizar). Los casos de prueba deben tener en cuenta el uso de todo tipo de datos de entrada/salida, cada comportamiento esperado, todos los elementos de diseño, y cada clase de defecto. Todos los requisitos deberán ser cubiertos por los casos de prueba [31].

A continuación se muestran los casos de prueba de aceptación realizados:

Tabla 22: Caso de Prueba 1

CASO DE PRUEBA DE ACEPTACIÓN	
Caso de prueba: 1	**Número de historia:** 1
Nombre de prueba: Configurar canal TCP.	
Descripción: Prueba que verifica si el sistema permite configurar el puerto del canal TCP.	
Condiciones de ejecución: Debe estar instalada en el ordenador la biblioteca TransportProvider.	
Entrada/Pasos de ejecución: Se selecciona en la barra de herramientas el número del puerto por el cual se desea establecer la conexión y se presiona la opción de iniciar conexión.	
Resultado esperado: El simulador comienza a escuchar por el puerto establecido.	
Evaluación de la prueba: Satisfactoria.	

Tabla 23: Caso de Prueba 2

CASO DE PRUEBA DE ACEPTACIÓN	
Caso de prueba: 2	**Número de historia:** 2
Nombre de prueba: Crear mapa de memoria.	
Descripción: Prueba que verifica que el sistema permita crear uno o varios mapas de memoria.	
Condiciones de ejecución: Debe estar ejecutándose la aplicación.	
Entrada/Pasos de ejecución: Se selecciona en el árbol de memoria que se encuentra en el	

lado izquierdo de la interfaz el tipo de dato, entero o flotante, haciendo clic derecho sobre el tipo de dato, aparece la opción de crear y se selecciona la misma. Luego aparece una ventana para establecer las propiedades del mapa de memoria, con la opción de Aceptar o Cancelar. Si la opción es Aceptar, el sistema verifica que el campo del índice del *file* no este vacío y que no coincida con alguno que esté creado previamente, de ser así se crea el nuevo mapa de memoria. Si la opción es Cancelar la ventana de diálogo se cierra.
Resultado esperado: Si se escoge la opción Aceptar se agrega un componente visual representando el mapa de memoria creado con los valores y las propiedades definidas por el usuario. Si presiona el botón Cancelar se anula la operación de crear el mapa de memoria.
Evaluación de la prueba: Satisfactoria.

Tabla 24: Caso de Prueba 3

CASO DE PRUEBA DE ACEPTACIÓN	
Caso de prueba: 3	**Número de historia:** 2
Nombre de prueba: Modificar mapa de memoria.	
Descripción: Prueba que verifica que el sistema permita modificar los mapas de memoria creados.	
Condiciones de ejecución: Tiene que haber creado al menos un mapa de memoria.	
Entrada/Pasos de ejecución: Se puede realizar de dos formas: 1-Se selecciona en el árbol de memoria, el mapa previamente creado, realizando clic derecho sobre el mismo aparece la opción de modificar y se selecciona la misma. Luego se muestra una ventana para modificar las propiedades del mapa de memoria, con la opción de Aceptar o Cancelar. Si la opción escogida es Aceptar, el sistema modifica los datos del *file* seleccionado. Si la opción es Cancelar la ventana se cierra. 2-Dirigirse al componente visual que representa el mapa de memorias hacer doble clic sobre la dirección específica que se quiera modificar, luego en la ventana que se genera establecer el valor o las propiedades deseadas, se tiene la opción de Aceptar o Cancelar. Si la opción es Aceptar, el sistema modifica los datos de la dirección seleccionada. Si la opción es Cancelar la ventana se cierra.	
Resultado esperado: Siempre que luego de realizar los pasos para modificar las propiedades del mapa de memoria se presione el botón Aceptar el sistema actualiza los valores y propiedades del componente visual a los especificados por el usuario.	
Evaluación de la prueba: Satisfactoria.	

Tabla 25: Caso de Prueba 4

CASO DE PRUEBA DE ACEPTACIÓN	
Caso de prueba: 4	**Número de historia:** 2
Nombre de prueba: Eliminar mapa de memoria.	
Descripción: Prueba que verifica que el sistema permita eliminar los mapas de memoria existentes.	
Condiciones de ejecución: Tiene que haber creado al menos un mapa de memoria.	
Entrada/Pasos de ejecución: Se selecciona en el panel lateral izquierdo el mapa de memoria previamente creado, realizando clic derecho sobre el mismo aparece la opción de eliminar y se selecciona la misma.	
Resultado esperado: Se elimina el mapa de memoria y el componente visual que lo representa.	
Evaluación de la prueba: Satisfactoria.	

Tabla 26: Caso de Prueba 5

CASO DE PRUEBA DE ACEPTACIÓN	
Caso de prueba: 5	**Número de historia:** 3
Nombre de prueba: Simular valores en los mapas de memoria.	
Descripción: Prueba que verifica que el sistema permita generar valores aleatorios, crecientes o decrecientes en las direcciones del mapa de memoria.	
Condiciones de ejecución: Se deben crear los mapas de memoria.	
Entrada/Pasos de ejecución: : Se puede realizar de dos formas: 1-Cuando se crea un nuevo mapa de memoria o cuando se va a modificar uno existente en la ventana que se genera se selecciona la opción de autosimulación y se define el valor inicial por donde se quiere comenzar a simular, el intervalo del salto y el intervalo de tiempo para modificar los valores. Se escoge el tipo de simulación que puede ser creciente, decreciente o de valores aleatorios. Se tiene la opción de Aceptar o Cancelar. Si la opción es Aceptar, el sistema comienza a simular valores en todas las direcciones del mapa de memoria con la configuración escogida por el usuario. Si la opción es Cancelar la ventana se cierra. 2-Dirigirse al componente visual que representa el mapa de memoria realizar doble clic sobre la dirección deseada, luego en la ventana que se muestra se escoge la opción de autosimulación y se define el valor inicial por donde se quiere comenzar a simular, el intervalo del salto y el intervalo de tiempo para modificar los valores. Se escoge el tipo de simulación que puede ser creciente, decreciente o de valores aleatorios. Se tiene la opción de Aceptar o Cancelar. Si la	

opción es Aceptar, el sistema comienza a simular valores en esa dirección específica con la configuración escogida por el usuario. Si la opción es Cancelar la ventana se cierra.
Resultado esperado: El mapa de memoria o la dirección especificada actualiza sus valores automáticamente con las características de la simulación escogidas por el usuario.
Evaluación de la prueba: Satisfactoria.

Tabla 27: Caso de Prueba 6

CASO DE PRUEBA DE ACEPTACIÓN	
Caso de prueba: 6	**Número de historia:** 4
Nombre de prueba: Simular errores de tramas.	
Descripción: Prueba que verifica que el sistema permita generar ruido en las comunicaciones.	
Condiciones de ejecución: Debe estar establecida la conexión entre el manejador y el simulador.	
Entrada/Pasos de ejecución: Se selecciona en la barra de herramientas la opción de simular ruido.	
Resultado esperado: El simulador enviará tramas incorrectas con datos erróneos al manejador.	
Evaluación de la prueba: Satisfactoria.	

3.4 Conclusiones parciales

Luego de la implementación del sistema y la posterior ejecución de los casos de pruebas definidos, se determinó que la aplicación desarrollada cumple el objetivo general de la presente investigación, reuniendo los requisitos necesarios para establecer una correcta simulación de los dispositivos PLC-5 en la comunicación mediante el protocolo AB Ethernet.

Conclusiones generales

Con la finalización de la presente investigación se alcanzan las siguientes conclusiones:

- Se logró la implementación de un sistema que emula el comportamiento de los dispositivos PLC-5 cuando realizan transacciones de datos a través del protocolo AB Ethernet.
- Las herramientas y tecnologías seleccionadas para el desarrollo del simulador facilitaron el diseño de una interfaz gráfica sencilla e intuitiva.
- Las funcionalidades implementadas garantizan la gestión de mapas de memoria y permiten la simulación de errores en el envío de tramas, lo que hace posible comprobar la respuesta del manejador AB Ethernet ante situaciones excepcionales.
- La realización de los casos de prueba diseñados permitieron validar el correcto funcionamiento de la solución, confirmando el cumplimiento del objetivo general de la investigación.

Recomendaciones

Para darle continuidad a la investigación o realizar trabajos futuros se recomienda:

- Añadir al simulador la funcionalidad de simular retraso en las comunicaciones para que la solución pueda generar un mayor número de situaciones excepcionales.
- Permitir que el simulador AB Ethernet genere otros tipos de datos para aumentar la capacidad de detectar posibles errores en el *driver*.

Bibliografía

1. Montero, Joan. *Sistemas SCADA y PLC*. [En línea] [Citado el: 24 de noviembre de 2014.] http://joan-montero.blogspot.com/2011/10/sistemas-scada-y-plc-montero-joan.html.

2. Rodríguez, Pedro Uriarte. *Manejador para la comunicación con dispositivos de campo mediante el protocolo DF1*. [En línea] 2010. [Citado el: 10 de diciembre de 2014.] http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_03836_10.

3. Diccionario de la Real Academia de España. [En línea] [Citado el: 4 de diciembre de 2014.] http://buscon.rae.es/drae/srv/search?id=lPyNMvlaDDXX22cDL6nd.

4. Simulación. [En línea] [Citado el: 4 de diciembre de 2014.] http://www3.uji.es/~berlanga/Docencia/Simulacion/Apuntes/tema1.pdf.

5. Simulación de Procesos. [En línea] [Citado el: 4 de diciembre de 2014.] http://www.simergia.com/simulacion-de-procesos.html.

6. Ventajas y desventajas al utilizar la simulación. [En línea] [Citado el: 4 de diciembre de 2014.] http://simulacionunilibre.blogspot.com/2011/02/ventajas-y-desventajas-al-utilizar-la.html.

7. Law, A. M. *Simulation Modeling and Analysis*. [En línea] 2000. [Citado el: 4 de diciembre de 2014.] http://www.gbv.de/dms/ilmenau/toc/54369156X.PDF.

8. Rivero, Miguel A. Albuerne. *Simulador de dispositivos PLC-5 para realizar el proceso de pruebas al manejador DF1 del SCADA SAINUX.* La Habana : s.n., 2013.

9. Ravelo, Jose C. Abraham. *Desarrollo de un simulador para realizar las pruebas del manejador Modbus Omni.* La Habana : s.n., 2012.

10. Vallejo, Horacio D. *Los Controladores Lógicos Programables*. [En línea] 2005. [Citado el: 30 de noviembre de 2014.] http://www.todopic.net/utiles/plc.pdf.

11. Principios básicos del PLC. [En línea] [Citado el: 16 de enero de 2015.] http://recursostic.educacion.es/observatorio/web/gl/component/content/article/502-monografico-lenguajes-de-programacion?start=2.

12. Gabancho, Humberto Reátegui. *Sistema Redundante de supervisión y control de despacho de combustibles de CB No 5 –Refinería Talara*. [En línea] 2007. [Citado el: 28 de noviembre de 2014.] http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Ingenie/reategui_gh/cap4.pdf.

13. Hernández, Luis Enrique García. *Guía de Implementación del manejador ABEthernet-SCADA GALBA.* La Habana : s.n., 2009.

14. Definición de Protocolo de Comunicación. [En línea] [Citado el: 3 de abril de 2015.] http://definicion.de/protocolo-de-comunicacion/.

15. Batista, Yunier Velázquez. *Sintaxis de Protocolos Implementados por los Manejadores.* La Habana : s.n., 2009.

16. Trama de Datos. [En línea] [Citado el: 13 de marzo de 2015.] https://metalkris.wordpress.com/.

17. Definición de TCP/IP. [En línea] [Citado el: 1 de diciembre de 2014.] http://www.masadelante.com/faqs/tcp-ip.

18. Aprenda QT desde hoy mismo. [En línea] [Citado el: 18 de diciembre de 2014.] http://es.scribd.com/doc/120249264/Aprenda-Qt4-hoy-mismo#force_seo.

19. Qt Creator, IDE Overview . [En línea] [Citado el: 18 de marzo de 2015.] https://qt-project.org/doc/qtcreator-2.5/creator-overview.html.

20. C++. [En línea] [Citado el: 1 de diciembre de 2014.] http://www.ecured.cu/index.php/C%2B%2B.

21. Lenguaje de programación C++. [En línea] [Citado el: 1 de diciembre de 2014.] http://lenguajedeprogramacion21.blogspot.com/.

22. JACOBSON, RUMBAUGH BOOCH. *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley.

23. Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana : s.n., 2014.

24. Sommerville, I. *Ingeniería del software.* s.l. : Pearson Educación, 2005.

25. Reyes, Carlos Miguel Pérez. *FISIM: Simulador Físico – Matemático integrado a la plataforma de gestión del aprendizaje ZERA*. [En línea] 2011. [Citado el: 15 de enero de 2015.] http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_04632_11.

26. García, Carlos Alejandro Suárez. *Diseño e Implementación de un módulo de integración para la Solución Tecnológica Integral para la Modernización de la División de Antecedentes Penales.* La Habana : s.n., 2011.

27. Bascon, Ernesto. *El patrón de diseño Modelo-Vista-Controlador y su implementación en Java Swing*. [En línea] [Citado el: 18 de marzo de 2015.] http://ucbconocimiento.ucbcba.edu.bo/index.php/ran/article/download/84/81.

28. Mora, Roberto Canales. Patrones GRASP. [En línea] [Citado el: 18 de marzo de 2015.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp.

29. CoDejaVu.Ejemplo Patrón Singleton. [En línea] [Citado el: 20 de mayo de 2015.] http://codejavu.blogspot.com/2013/07/ejemplo-patron-singleton.html.

30. Análisis y Diseño de Sistemas II. Diagrama de despliegue. [En línea] [Citado el: 20 de mayo de 2015.] http://virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc.

31. Testeando el software.Definición de caso de prueba. [En línea] [Citado el: 25 de mayo de 2015.] http://testeandosoftware.com/casos-de-uso-vs-casos-de-prueba/.

Glosario de Términos

Allen-Bradley: marca de una línea de equipos de automatización de fábricas, pertenece a la compañía Rockwell Automation.

ASCII: son las siglas en inglés de *American Standard Code for Information Interchange*, el cual consiste en un modelo o patrón de caracteres a usarse durante el intercambio de información basado en el alfabeto latino tal como se usa en las lenguas occidentales.

Callback: devolución de llamada o retrollamada.

CASE: las herramientas CASE (*Computer Aided Software Engineering*) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en término de tiempo y de dinero.

Data Highway +: Es una red de área local diseñada para trabajar con programación remota y adquisición de datos para aplicaciones de la planta.

Dispositivo de campo: elementos físicos que miden, monitorean y en algunos casos, almacenan los datos de las variables del proceso. Estos dispositivos no se conectan directamente al SCADA.

Framework: en los sistemas orientados a objeto un *framework* es un conjunto de clases que encapsulan diseños abstractos de soluciones a un determinado número 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.

GNU/Linux: el núcleo Linux se complementa con una serie de aplicaciones desarrolladas por el grupo GNU para conformar el sistema operativo de *software* libre GNU/Linux.

IBM: la *International Business Machines* es una empresa multinacional estadounidense de tecnología y consultoría. Fabrica y comercializa hardware y software para computadoras, además ofrece servicios de infraestructura, alojamiento de Internet y consultoría en una amplia gama de áreas relacionadas con la informática, desde computadoras centrales hasta nanotecnología.

Interfaz gráfica, visual o de usuario: es la 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.

LAN: Siglas en inglés de *Local Area Network* o Red de Área Local.

Manejadores de Dispositivos (Drivers*)*: son módulos independientes en forma de bibliotecas dinámicas, que implementan un protocolo, permiten el intercambio de datos que provienen de disímiles equipos.

Multiplataforma: término que se usa para referirse a los programas, sistemas operativos, lenguajes de programación u otra clase de software, que puedan funcionar en diversas plataformas o sistemas operativos.

Protocolo de comunicación: conjunto de normas que regulan la comunicación entre los distintos componentes de una red informática.

Rockwell Automation: Es un proveedor mundial de soluciones de automatización, control e informatización industrial.

WAN: Siglas en inglés de *Wide Area Network* o Red de Área Amplia.

Anexos 1: Imágenes de la aplicación

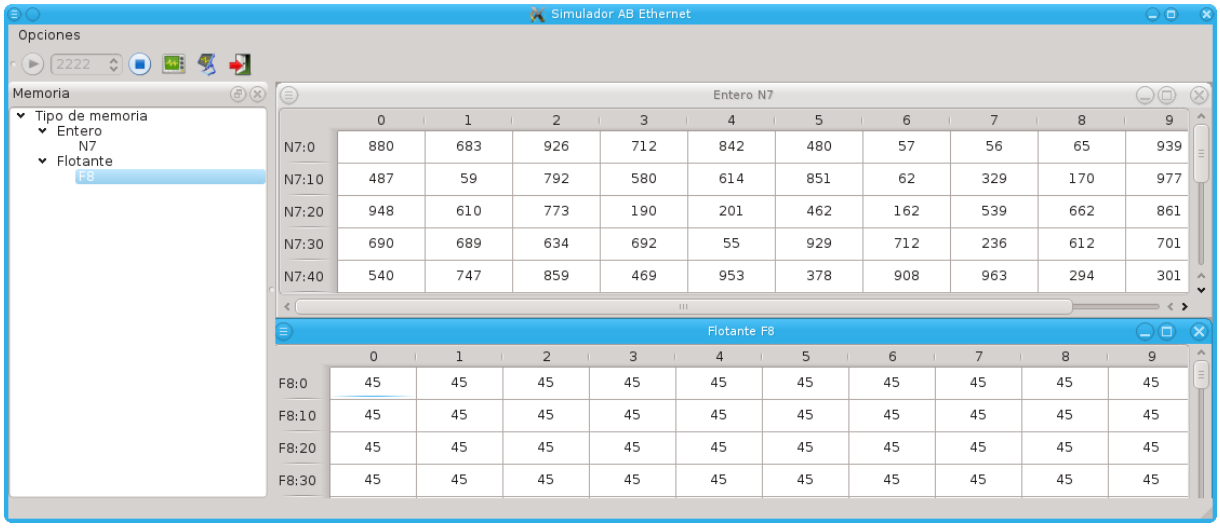


Figura 13: Interfaz principal con dos mapas de memoria creados simulando valores

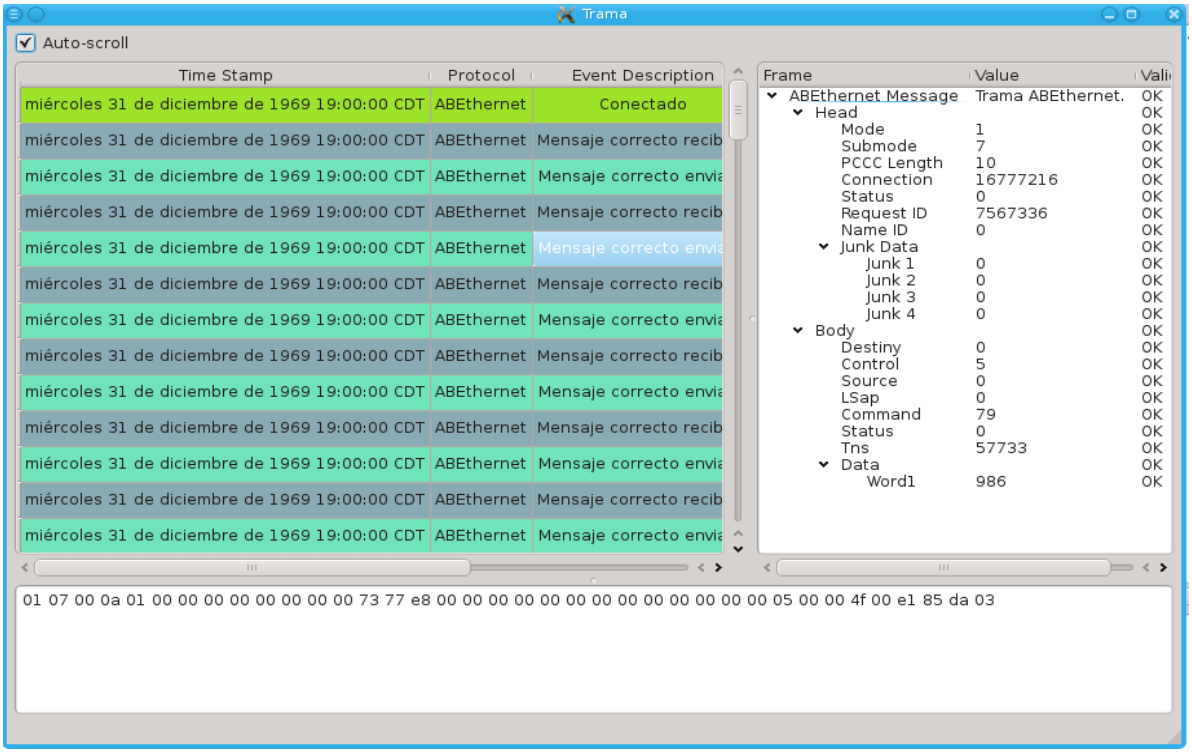


Figura 14: Ventana para visualizar el envío y la recepción de los mensajes.

Figura 15: Ventana para crear mapa de memoria

Figura 16: Ventana para modificar una dirección del mapa de memoria