UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5

Centro de Informática Industrial

"Simulador de dispositivos PLC-5 para realizar el proceso de pruebas al manejador DF1 del SCADA SAINUX"

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: **Miguel Angel Albuerne Rivero**

Tutor: **Ing. Pedro Alberto Uriarte Rodríguez**

Co-tutores: **Ing. José Carlos Abraham Ravelo**
Ing. Enelis Cuba Rondón

Ciudad de La Habana

2013

"Año 55 de la Revolución"

Agradecimientos

Agradezco por su apoyo y amor incondicional a toda mi familia, especialmente a mis padres Mamita y Papito, los cuales han sido los verdaderos creadores de mi persona, en cuerpo y espíritu. A mi hermanita querida, que aunque lejos siempre ha sabido apoyarme y estar pendiente de mis cosas. A mi sobrina linda que es la alegría de la familia y un motivo para seguir adelante. No puedo dejar de mencionar a mis abuelos Monga, Angel y Chicha; a los cuales quiero tanto y han sido excelentes maestros en mi vida, gracias por existir y enseñarme que con una sonrisa y un buen café, basta para ser feliz.

Quiero agradecer también a mis tíos y tías, por su apoyo y todas las cosas buenas que me han permitido compartir con su presencia, especialmente a mi tía Juana, que siempre fue un motor impulsor en mi carrera y en mi vida personal, gracias por tanto cariño, gracias por estar siempre tan presente.

A todos mis primos y primas, desde los más viejos hasta los que están por nacer, y forman parte de lo más grande que puede tener una persona, su familia.

A lo profesores que me supieron inculcar sentimientos y lecciones para la vida laboral y en el ámbito personal, muchas gracias.

A los compañeros, amigos y amigas que me han acompañado estos cinco años de carrera, y a los que he tenido lejos pero muy cerca en el corazón, gracias por dejarme pasar tan buenos ratos de alegría.

A los tutores de mi trabajo de diploma, gracias por dar lo mejor de sí para guiarme en esta laboriosa tarea, y ayudarme a alcanzar este resultado.

Agradezco a la Revolución y a nuestro sistema por permitirme estudiar y cumplir mis sueños, además de hacer de mí un ser humano objetivo y comprometido.

A todos los que hicieron que esto fuera posible, a todos los que pusieron su granito de arena, muchas gracias.

Miguel Angel

Dedicatoria

A mis padres, Eulalia y Miguel, por darme todo.

A mi hermanita linda Yaimí y toda su familia.

A mis abuelitos queridos Gertrudis, Ramona y Angel.

A la familia en general.

Miguel Angel

Declaración de autoría

Declaro ser autor 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 firmo la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Autor

Miguel Angel Albuerne Rivero

_____

Firma del Tutor

Ing. Pedro Alberto Uriarte Rodríguez

Datos de contacto

Ing. Pedro Alberto Uriarte Rodríguez

Graduado con el título de Ingeniero en Ciencias Informáticas en el 2010 en la Universidad de las Ciencias Informáticas.

E-mail: pauriarte@uci.cu

Resumen

Los simuladores constituyen herramientas muy utilizadas para la realización de pruebas y la toma de decisiones en el proceso de desarrollo de sistemas informáticos, debido a la disminución de gastos y la protección ante posibles efectos negativos, entre otros beneficios.

En el Centro de Informática Industrial (CEDIN) actualmente se desarrolla el Sistema de Automatización Industrial UX (SAINUX), el cual tiene entre sus componentes fundamentales el manejador DF1, encargado de realizar la recolección de datos provenientes de dispositivos de campo, y que permitirá el control y la supervisión de disímiles procesos industriales.

En el presente trabajo se muestran las características y funcionalidades que posee el simulador de dispositivos PLC-5 (Controlador Lógico Programable modelo 5), el cual fue desarrollado a partir de la ausencia en el CEDIN de dispositivos PLC-5 y de una aplicación que posea sus características fundamentales, dificultando de esta manera la detección de errores llevada a cabo en la etapa de pruebas al manejador DF1, lo que impide verificar el correcto funcionamiento de este componente antes de ser utilizado por el SCADA SAINUX.

La investigación abarca el estado del arte de la simulación de dispositivos industriales, así como los principales conceptos utilizados en el proceso de adquisición y envío de datos a través de Ethernet y puerto serial, especificaciones del protocolo de comunicación DF1 y las tecnologías y herramientas utilizadas en el desarrollo. Además se representa la arquitectura definida para la aplicación, así como los requisitos funcionales y no funcionales que regirán el proceso de desarrollo de la misma. Se concluye el proceso de desarrollo con la realización de las pruebas que validan el funcionamiento del simulador. Como resultado de esta investigación se obtiene una aplicación que permite realizar pruebas al manejador DF1, para comprobar su correcta implementación.

Palabras claves: simuladores, SCADA, manejador, PLC-5.

Índice

Índice de tablas

Índice de figuras

Introducción

El ser humano, como eslabón principal de la sociedad humana, ha enfrentado diversos procesos, en muchos de los cuales su presencia directa ha sido indispensable, dado el contexto histórico y el avance científico tecnológico de la época. En la actualidad, el desarrollo de la informática ha permitido la automatización de muchos procesos, principalmente en el área industrial; en los que anteriormente era imposible su desarrollo sin la presencia del hombre. Un ejemplo, son los sistemas de Supervisión, Control y Adquisición de Datos (SCADA, por sus siglas en Ingles), diseñados con la finalidad de controlar y supervisar procesos a distancia, basándose en la adquisición de datos de los procesos remotos [1].

La Universidad de las Ciencias Informáticas desempeña un papel importante en la producción de software en Cuba. La misma está compuesta por varias facultades, entre las que se encuentra la facultad 5, en la cual radica el Centro de Informática Industrial (CEDIN) que tiene entre sus objetivos principales el desarrollo de aplicaciones para la automatización de procesos industriales.

El CEDIN actualmente tiene como producto en desarrollo el SCADA Sistema de Automatización Industrial UX (SAINUX), con el objetivo de ofrecer servicios de control y automatización de procesos industriales a clientes nacionales y extranjeros.

El SCADA SAINUX para ejercer la supervisión y control de los procesos industriales, tiene entre sus funciones elementales la recolección de datos provenientes de dispositivos que se encuentran en la industria, como es el caso del Controlador Lógico Programable 5 (*PLC-5*, por sus siglas en inglés), el cual tiene la capacidad de comunicarse con otros dispositivos y estaciones de trabajo a través el protocolo DF1. El proceso de recolección es realizado por manejadores de dispositivos (*drivers*, en inglés), programas que sirven de intermediarios, entre los dispositivos mencionados anteriormente y el sistema SCADA. Cada driver implementa un protocolo de comunicación específico mediante el cual se realiza la conexión.

El proceso de desarrollo de los manejadores de dispositivos en el CEDIN, se ha visto afectado debido a la ausencia, por cuestiones económicas, de dispositivos de campo que permitan realizar las pruebas a estos componentes de manera eficiente. Una variante para enfrentar esta problemática son los “simuladores de dispositivos”, aplicaciones que simulan el comportamiento de dispositivos de campo cuando se comunican mediante un determinado protocolo de comunicación.

La línea de Adquisición del departamento de Construcción de Componentes perteneciente al CEDIN, finalizó la etapa de implementación del manejador DF1, el cual implementa las

especificaciones del protocolo de comunicación del mismo nombre, con el objetivo de ser utilizado para la recolección de datos por el SCADA SAINUX. Esto se ha visto imposibilitado, debido a que al llegar a la etapa de pruebas, no se cuenta con un dispositivo PLC-5, ni con un simulador que cumpla con los requisitos necesarios para realizarle pruebas a este manejador. La carencia de estos elementos dificulta en gran medida la detección de posibles errores en el funcionamiento del manejador DF1 e impide su validación para ser utilizado por SCADA SAINUX.

La **situación problémica** anteriormente planteada permite formular el siguiente **problema científico**: ¿Cómo realizar el proceso de pruebas al manejador DF1 del SCADA SAINUX?

El problema planteado define el siguiente **objeto de estudio**: Proceso de simulación de dispositivos industriales y transferencia de datos a través de un protocolo de comunicación.

Para dar solución al problema se definió como **objetivo general**: Desarrollar una herramienta que simule el comportamiento en la transferencia de datos del dispositivo PLC-5 e implemente las especificaciones del protocolo de comunicación DF1.

Por todo lo anterior queda definido como **campo de acción**: Envío y recepción de información utilizando el protocolo DF1.

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

- Elaboración del marco teórico de la investigación a partir del estudio del estado del arte sobre el desarrollo de simuladores para dispositivos industriales.
- Estudio de las especificaciones del protocolo DF1 para la definición de los requerimientos a implementar.
- Definición de la arquitectura y las clases para la elaboración del diseño del simulador.
- Implementación del simulador para la realización de pruebas al manejador DF1.
- Diseño y aplicación de las pruebas al simulador para verificar su funcionamiento.

Como **idea a defender** de la tesis el autor considera que: Con la aplicación de la herramienta desarrollada, que simule el comportamiento en la transferencia de datos de los dispositivos PLC-5, se podrá realizar el proceso de pruebas al manejador DF1 del SCADA SAINUX.

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

- **Analítico-Sintético**: Utilizado para identificar los principales fundamentos teóricos, así como las características fundamentales de la simulación de dispositivos y la

comunicación mediante el protocolo DF1, a través del estudio de documentación recopilada.

- **Modelación**: Utilizado para la elaboración de diagramas que modelan la estructura, relaciones internas y características del objeto de investigación. Creando abstracciones que permiten explicar la realidad.

Como **método empírico** se utilizó:

- **Experimental**: Empleado en la realización de pruebas, donde se crean las condiciones propicias para verificar el correcto funcionamiento del software desarrollado en la presente investigación.

El presente documento está estructurado de la siguiente manera:

Capítulo 1: Fundamentación teórica. En este capítulo se elabora el marco teórico de la investigación mediante la definición de conceptos y características fundamentales de la simulación de dispositivos y la comunicación mediante el protocolo DF1. Además se definen las tecnologías y herramientas que se utilizarán para realizar el diseño, y posterior implementación de la solución propuesta.

Capítulo 2: Descripción y diseño del simulador. Durante este capítulo se elaboran las Historias de Usuarios para identificar los requisitos que debe cumplir el simulador, y se lleva a cabo el modelado de las funcionalidades y el diseño de las clases que se utilizarán en la implementación.

Capítulo 3: Implementación y prueba. Se realiza la implementación del simulador de acuerdo a las funciones y clases diseñadas previamente. Además se muestran las pruebas realizadas al mismo y los resultados obtenidos.

Capítulo 1: Fundamentación teórica

Introducción

En la actualidad el uso de la simulación es muy común en diferentes áreas, debido a las ventajas que esta proporciona. En el ámbito industrial, particularmente, es de vital importancia ya que permite prevenir eventos indeseables, mediante su apoyo a la toma de decisiones y corrección de errores. En este capítulo se muestran características, conceptos y definiciones de los elementos relacionados con la simulación y la transferencia de datos utilizando el protocolo DF1, así como el análisis de simuladores de dispositivos que existen y son afines con la investigación. Además se mencionan las herramientas y las tecnologías utilizadas para el desarrollo de la solución propuesta.

1.1. Simulación

Existen muchas definiciones de simulación, una de ellas es la planteada por David Himmelblau y Kenneth Bischoff en el libro "Análisis y simulación de procesos", en la que conciben la simulación como la *"representación de un fenómeno a través de modelos, lo que permite analizar sus características con mayor facilidad sin tener que desarrollar el fenómeno, con lo que se ahorra tiempo y recursos, uno de los objetivos primordiales de una simulación es analizar los resultados para así conocer con anterioridad su comportamiento y en caso posible mejorarlos en el momento que se lleve a cabo el fenómeno en la vida real*"[1].

Por otro lado el colectivo de autores del libro "Simulación de sistemas" plantea que "*La simulación se define como la imitación o réplica del comportamiento de un sistema o de una situación, usando un modelo que lo representa de acuerdo al objetivo por el cual se estudia el sistema*"[2].

Ambas definiciones permiten ver la simulación como el proceso donde se lleva un fenómeno real a uno o varios modelos que representen sus características y comportamientos en diversas situaciones, permitiendo su análisis con el objetivo de mejorar el sistema real que se estudia.

Desde otra perspectiva, la simulación se puede enfocar como un proceso experimental, como la define Robert E. Shannon en el libro "Systems Simulation: The Art and Science", planteando que "*simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos para el funcionamiento del sistema*" [3].

Son muchas las definiciones y usos de la simulación. En el desarrollo de la solución propuesta se utiliza como herramienta informática, donde se representa el comportamiento que tienen los dispositivos PLC-5 cuando realizan transacciones de datos utilizando el protocolo de comunicación DF1.

1.1.1. Tipos de simulación

De acuerdo al tipo de sistema o problema que se estudia, o más precisamente, de acuerdo al tipo de modelo que lo representa, la simulación puede ser: [2]

- **De tiempo continuo:** este tipo de simulación trata modelos descritos por ecuaciones diferenciales ordinarias o ecuaciones diferenciales parciales.
- **De tiempo Discreto:** Trata modelos descritos por ecuaciones de diferencias.
- **De eventos Discretos:** Trata con sistemas que pueden ser modelados como una secuencia de eventos contables, asumiendo que nada importante ocurre entre esos eventos. Sistemas tales como los de colas, manufacturas, redes de comunicación, inventarios, son representables así.

Para el desarrollo de la solución propuesta se utiliza el tipo de simulación de eventos discretos dado que es la más adecuada para la presente investigación.

1.1.2. Ventajas de la simulación

Entre las ventajas que ofrece la simulación se destacan las siguientes: [2]

- Es un método de mayor facilidad de aplicación que el analítico.
- Con un mismo modelo se puede conocer una gran variedad de comportamientos del sistema, mediante el uso repetitivo del modelo cambiando algunos elementos.
- Permite estudiar sistema de los cuales se cuenta con información incompleta, ayudando a generar información complementaria.
- Permite expandir el tiempo.
- Permite comprimir el tiempo.

1.1.3. Simuladores de dispositivos

Los simuladores son herramientas desarrolladas con el objetivo de simular un determinado proceso real mediante la construcción de modelos virtuales, con la posibilidad de interactuar con elementos externos. La mayoría son sistemas informáticos capaces de manejar un gran número de variables y tienen gran importancia en las predicciones, el entrenamiento y la educación. Su

importancia está dada por la capacidad de representar procesos reales, reducir costos y en ocasiones prevenir pérdidas materiales y de vidas humanas.

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 [16].

1.2. Controladores Lógicos Programables (PLC)

Para un mayor entendimiento de los dispositivos que se van a simular se realizó un análisis de las características fundamentales de los PLC.

Las industrias automatizadas deben lograr en sus sistemas, alta confiabilidad, eficiencia y flexibilidad. Para lograr estas metas deben poseer como elemento fundamental un conjunto de dispositivos electrónicos denominados Controladores Lógicos Programables (PLC). Estos dispositivos son conocidos como máquinas electrónicas diseñadas para controlar en tiempo real y en el medio industrial, procesos secuenciales de control [6]. La siguiente figura representa un ejemplo de PLC.

Figura 1: Controlador lógico programable.

Los PLC ofrecen diversas ventajas en procesos de automatización, puesto que cuentan con las herramientas adecuadas para realizar su desempeño, tales como: relés, temporizadores

electrónicos, contadores y controles mecánicos como el tipo tambor. Además de disponer de facilidades de comunicación para acceder a subsistemas de comunicaciones.

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

El PLC se compone esencialmente de algunas partes comunes a todos los modelos, y otras que dependen de la envergadura del mismo y la aplicación donde se usará: [7]

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

1.2.1. Dispositivo PLC-5

El procesador PLC-5 sirve como núcleo de miles de soluciones de control Allen-Bradley[1] en todo el mundo, debido en gran parte a las siguientes características: [20]

- Flexibilidad de programación, conexión de redes, opciones de E/S y selección de controladores.
- Confiabilidad con un valor nominal MTBF (Tiempo medio entre fallos) de más de 400 000 horas.

[1] Allen-Bradley: Es la marca de una línea de equipos de automatización de fábricas, creada por la compañía Rockwell Automation.

- Compatibilidad con los productos que posee hoy y los nuevos productos introducidos continuamente por *Rockwell Automation*.

Los PLC-5 pueden usarse en un sistema diseñado para control centralizado o en un sistema diseñado para control distribuido. [20]

- **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.

Estos PLC pueden comunicarse mediante una red con otros controladores y con una estación de trabajo. Para establecer comunicación con otros dispositivos cuenta con diversos periféricos entre los cuales se encuentra, el puerto serie, que es configurable para RS-232, RS-423 ó la comunicación en serie compatible con RS-422A. Se usa para conectar dispositivos que [25]:

- Se comunican usando el protocolo DF1 tales como módems, módulos de comunicación, estaciones de trabajo de programación u otros dispositivos.
- Envían y reciben caracteres ASCII, tales como los terminales ASCII, lectores de código de barras e impresoras.

Entre los modos de comunicación que se le pueden configurar para utilizar el protocolo DF1, se encuentran: [25]

- Punto a punto comunicación entre un controlador PLC-5 y otros dispositivos compatibles con DF1. En el modo punto a punto el controlador PLC-5 usa el protocolo Full-duplex DF1.
- DF1 maestro control de encuestas (*polling,* en inglés) y transmisión de mensajes entre el maestro y cada nodo remoto. En el modo maestro el controlador PLC-5 usa el protocolo de encuestas Half-duplex DF1.
- DF1 esclavo usa el controlador como estación esclavo en una red en serie maestro/esclavo. En el modo esclavo el controlador PLC-5 usa el protocolo Half-duplex DF1.

El puerto serie también es compatible con las aplicaciones SCADA, los cuales permiten monitorear y controlar las funciones y procesos remotos mediante redes de comunicación en serie entre maestro y esclavos.

1.2.2. Mapa de memoria del PLC-5

Una de las partes del PLC-5 que conforman la estructura mencionada en el epígrafe 1.2, es el área de memoria que se encuentra dentro del CPU, la cual se utiliza para diversas funciones: memoria de programa de usuarios, de tabla de datos (mapa de memoria), del sistema y de almacenamiento.

Se le llama memoria al lugar físico que almacena todo cuanto necesita para ejecutar la tarea de control un dispositivo PLC-5. Para realizarla correctamente cuenta con diferentes datos: [7]

- Datos del proceso:
  - Señales de planta, entradas y salidas.
  - Variables internas, de bit y de palabra.
  - Datos alfanuméricos y constantes.
- Datos de control:
  - Instrucciones de usuario (programa).
  - Configuración del autómata (modo de funcionamiento, número de E/S conectadas, entre otros).

En el PLC-5, la memoria es organizada en bloques de hasta 1000 elementos consecutivos que se denominan "*files*" o ficheros [11]. Existen ocho ficheros de datos definidos por defecto pero se pueden crear ficheros adicionales si son necesarios por el programa del PLC, como se ilustra en la Figura 2.

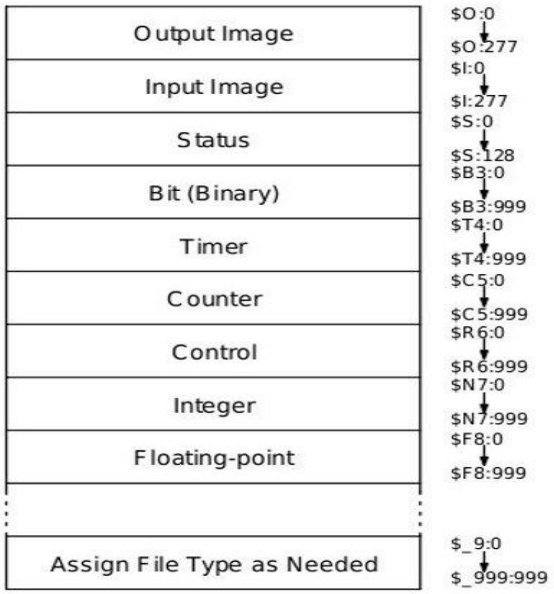

Figura 2: Mapa de memoria.

El PLC-5 permite el acceso a los datos (variables) que se encuentran en el mapa de memoria, mediante al menos tres elementos fundamentales que conforman direcciones de variables: el tipo de fichero (primera letra de una dirección), el número del fichero (número que sigue a la primera letra), y el desplazamiento de la variable en el fichero (número separado por el símbolo de dos puntos ":" de los elementos anteriores). Se pueden apreciar ejemplos de direcciones de variables en la Figura 2, antecedidas por el símbolo de dinero "$".

En la siguiente tabla se describen algunos de los tipos de datos de los ficheros almacenados en el mapa de memoria, y son utilizados por la solución propuesta, omitiendo aquellos que no utiliza el manejador DF1. Además se representa la cantidad de palabras (2 bytes) que ocupa cada uno de ellos.

Tabla 1: Tipos de datos del mapa de memoria.

Tipo de datos	Prefijo	Descripción	Tamaño en bytes
I/O Image	(I/O)	Entradas y salidas físicas del PLC	1 word(2 bytes)
Status File	S	Palabras de estado del PLC de 16 bits	1 word(2 bytes)
Binary	B	Palabras de 16 bits que pueden almacenar bits independientes	1 word(2 bytes)
Floating Point	F	Números flotantes de 32 bits	2 word(4 bytes)
Integer	N	Números enteros con signo de 16 bits	1 word(2 bytes)
Timer	T	Temporizadores	3 word(6 bytes)
Counter	C	Contadores	3 word(6 bytes)
Control	R	Palabras de Control	3 word(6 bytes)
SFC Status	SC	Palabras de estado SFC del PLC	3 word(6 bytes)
PID Control	PD	Palabras de control del PID	82 word(164 bytes)

1.3. Especificaciones del Protocolo DF1

Un protocolo de comunicación se define como el conjunto de reglas que son utilizadas para la interpretación de las señales transmitidas sobre un medio físico. Como es el caso del protocolo DF1, ampliamente utilizado en las industrias.

Este último protocolo tiene como principales funcionalidades: [9]

- Llevar un mensaje, libre de errores, desde un extremo del enlace al otro. Sin tener ninguna preocupación por el contenido del mensaje, la función del mensaje, o el último propósito del mensaje. Esto se logra adjuntando el chequeo de caracteres BCC[2] o CRC[3] a cada comando y respuesta. BCC y CRC son códigos de detección de errores usados frecuentemente en redes digitales y en dispositivos de almacenamiento para detectar cambios accidentales en los datos. El receptor recibe el mensaje y verifica su BCC o CRC, retornando un ACK (del inglés *acknowledgement*, en español acuse de recibo o asentimiento) si es correcto, o un NAK (del inglés *negative acknowledgement*, en español acuse de recibo negativo o asentimiento negativo) en caso contrario.
- Indicar el fallo con un código de error. Internamente debe delimitar el mensaje, detectar y señalar los errores, así como realizar el reintento tras error.

DF1 es un protocolo de la capa de enlace de datos que combina las características de las subcategorías D1 (transparencia de datos) y F1 (transmisión bidireccional simultánea con

[2] BCC: Del inglés *block check character*, en español Caracter de Chequeo de Bloque.
[3] CRC: Siglas en inglés de Chequeo de Redundancia Cíclica.

respuestas implícitas) de la especificación ANSI x3.28. Existen dos categorías del protocolo DF1 [9]:

- Protocolo **Full-duplex** (Comunicación punto a punto).
- Protocolo **Half-duplex** (Comunicación Maestro/Esclavo).

1.3.1. Capas del protocolo DF1

El protocolo DF1 utiliza dos capas a nivel de software para establecer correctamente la comunicación: [11]

- **Capa de enlace de datos**: Controla el flujo de comunicación sobre el medio físico. Determina la codificación de los datos, controla quien transmite datos y quien "escucha" usando un protocolo arbitrario. Además transmite paquetes de datos intactos desde una fuente emisora hasta un nodo destino, sobre el enlace físico.
- **Capa de aplicación**: Esta capa controla y ejecuta el comando especificado en la comunicación entre nodos. Es la interfaz entre los procesos de usuarios y las bases de datos.

1.3.2. Formato general de las tramas

El protocolo DF1 utiliza varias tramas de mensajes, en dependencia del modo de comunicación que se utilice, ya sea Full-duplex o Half-duplex. Ambos son orientados a caracteres, utilizando los de control ASCII, extendidos con ocho bits, adicionando cero en el bit 7 [11], como se muestra en las siguientes tablas.

Tabla 2: Caracteres utilizados por Full-duplex.

Abreviatura	Valor hexadecimal
STX	02
ETX	03
EOT	04
ENQ	05
ACK	06
DLE	10
NAK	0F

Tabla 3: Caracteres utilizados por Half-duplex.

Abreviatura	Valor hexadecimal
STX	02
SOH	01
ETX	03
EOT	04
ENQ	05
ACK	06
DLE	10
NAK	0F

1.3.3. Tramas empleadas en el modo Full-duplex

El modo Full-duplex implementa diferentes tramas de mensajes en dependencia de la capa de red. La siguiente figura muestra el formato de la trama de un mensaje utilizado por el protocolo Full-duplex y la capa en la que es implementado cada elemento [9].

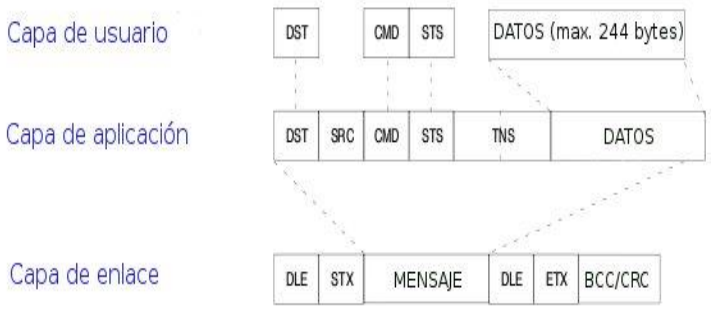

Figura 3: Trama de mensaje Full-duplex.

Este protocolo, al igual que Half-duplex, utiliza símbolos que tienen un significado específico en el funcionamiento del protocolo, y están compuestos por una secuencia de uno o más caracteres, los cuales deben ser enviados uno detrás de otro, sin otros caracteres entre ellos. Los símbolos utilizados pueden ser de datos (símbolos fijados requeridos por el protocolo DF1 para leer un mensaje) o de control (símbolos variables, de contenido de datos de la aplicación para un determinado mensaje).

A continuación se hace una descripción de los símbolos que son utilizados por el protocolo Full-duplex:

Tabla 4: Símbolos de transmisión Full-duplex.

Símbolo	Tipo de símbolo	Descripción
DLE STX	Control	Indica el comienzo de un mensaje.
DLE ETX BCC/CRC	Control	Indica la terminación de un mensaje.
DLE ACK	Control	Respuesta que indica que el mensaje se recibió satisfactoriamente.
DLE NAK	Control	Respuesta que indica que el mensaje no se recibió satisfactoriamente.
DLE ENQ	Control	Respuesta que se le envía al receptor en espera de símbolo de respuesta.
MENSAJE	Datos	Caracteres simples que tienen valores entre 00-0F y 11-FF. Incluye datos de la capa de aplicación, programas de usuario y rutinas de aplicaciones comunes. Un dato 10^{16} es enviado como un 10 10 (DLE DLE).
DLE DLE	Datos	Representa el valor del dato de 10^{16}.

1.3.4. Tramas empleadas en el modo Half-duplex

El protocolo Half-duplex utiliza tres tipos de transmisiones: [9]

- Trama *polling* (invitación a emitir).
- Trama de mensaje maestro.
- Trama de mensaje esclavo.

El nodo maestro transmite tramas *polling* y tramas de mensaje maestro, y los nodos esclavos transmiten las tramas de mensaje esclavo. La Figura 4 ilustra la estructura de las tramas mencionadas anteriormente.

La trama de mensaje esclavo tiene el mismo formato que la trama de mensaje utilizada por el protocolo Full-duplex. Por otro lado la trama del mensaje maestro es la misma que la trama de mensaje esclavo, solamente diferenciada por el símbolo DLE SOH y una dirección que especifica una estación esclava (STN) [9].

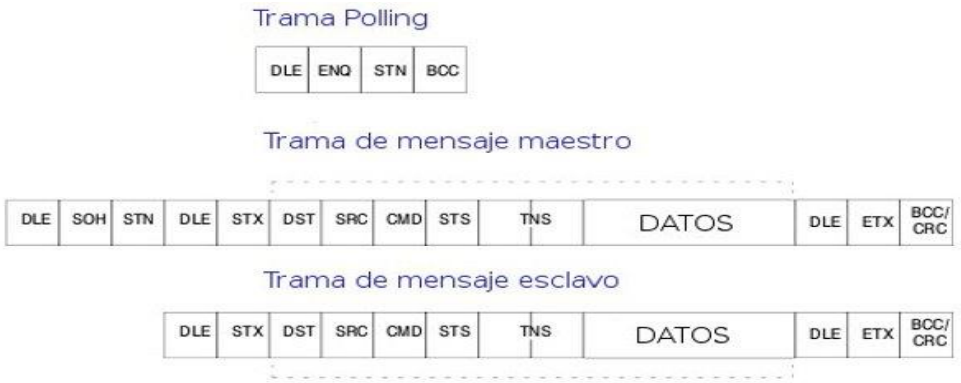

Figura 4: Tramas de mensajes Half-duplex.

Los símbolos utilizados por el modo Half-duplex para realizar las transacciones de datos son los siguientes:

Tabla 5: Símbolos de transmisión Half-duplex.

Símbolo	Tipo de símbolo	Descripción
DLE SOH	Control	Indica el comienzo de un mensaje.
DLE STX	Control	Separa los datos de la cabecera de comunicación.
DLE ETX BCC/CRC	Control	Indica la terminación de un mensaje.
DLE ACK	Control	Respuesta que indica que el mensaje se recibió satisfactoriamente.
DLE NAK	Control	Respuesta que indica que el mensaje no se recibió satisfactoriamente.
DLE ENQ	Control	Respuesta que se le envía al receptor en espera de símbolo de respuesta.
DLE EOT BCC	Control	Respuesta que dan los esclavos a los mensajes encuesta enviados por el maestro cuando no tienen mensajes que transmitir.
STN	Datos	Indica el número del nodo esclavo en un vínculo Half-dúplex.
DATOS	Datos	Caracteres simples que tienen valores entre 00-0F y 11-FF. Incluye datos de la capa de aplicación, programas de usuario y rutinas de aplicaciones comunes. Un dato 10^{16} es enviado como un 10 10 (DLE DLE).
DLE DLE	Datos	Representa el valor del dato de 10^{16}.

1.3.5. Características de comunicación del protocolo Full-duplex

Sobre este protocolo de enlace son posibles dos vías simultáneas de transmisión. Puede ser visto como un puente de dos sendas, donde el tráfico puede viajar en ambas direcciones al mismo tiempo. Existen dos frecuencias una para transmitir y otra para recibir. Ejemplos de este tipo abundan en el terreno de las telecomunicaciones, el caso más típico es la telefonía, donde el transmisor y el receptor se comunican simultáneamente utilizando el mismo canal, pero usando dos frecuencias [11].

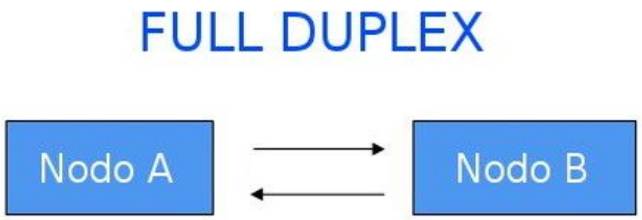

Figura 5: Comunicación Full-duplex.

Para la transmisión de mensajes el protocolo Full-duplex define las siguientes restricciones: [9]

- El tamaño mínimo de la capa de enlace de datos es 6 bytes.
- El tamaño máximo de la capa de enlace de datos depende de la capa de comando de la aplicación.
- Algunas implementaciones requieren que el primer byte del mensaje concuerde con la dirección del nodo, o de lo contrario el receptor ignora dicho mensaje.
- Como parte del algoritmo de detección de mensajes duplicados, el receptor compara el segundo, tercer, quinto y sexto byte de la capa de enlace con los mismos bytes del mensaje anterior. Si no hay diferencia entre este grupo de bytes, el mensaje es clasificado como una retransmisión del mensaje anterior.

1.3.6. Características de comunicación del protocolo Half-duplex

Half-duplex es un protocolo multipunto donde existe un nodo maestro y varios nodos esclavos. Se le pueden conectar simultáneamente desde 2 hasta 255 nodos, sobre un único enlace. Permite el envío de información en ambas direcciones pero con la limitación de que solo puede realizarse en una dirección a la vez [11]. Los nodos maestros y los nodos esclavos que se encuentren conectados deben compartir la misma frecuencia.

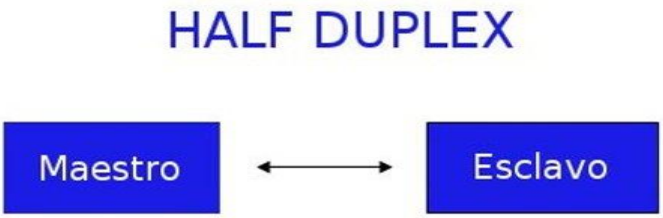

Figura 6: Comunicación Half-duplex.

Este protocolo define como restricciones en el envío de mensajes las siguientes condiciones:

- El tamaño mínimo de la capa de enlace de datos es 6 bytes.
- El tamaño máximo de la capa de enlace de datos depende de la capa de comando de la aplicación.
- Algunas implementaciones requieren que el primer byte del mensaje concuerde con la

dirección del nodo, o de lo contrario el receptor ignora dicho mensaje.

- Para la detección de mensajes duplicados, el receptor compara la fuente (SRC), el comando (CMD) y el campo de transacción (TNS) de cada mensaje. Luego se comparan con los mismos campos en el mensaje anterior, si son iguales el mensaje es clasificado como una retransmisión del mensaje anterior.

1.4. Modelo Maestro/Esclavo

Según International Business Machines (IBM) el modelo Maestro/Esclavo se define 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 y/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"* [4].

En este modelo el "maestro" es el que inicia una petición de servicio. La petición inicial puede convertirse en múltiples peticiones de trabajo a través de redes LAN o WAN. El maestro se abstrae de la ubicación de los datos o las aplicaciones.

El "esclavo" es cualquier recurso de cómputo dedicado a responder a los requerimientos del maestro. Los esclavos pueden proveer de múltiples servicios a los maestros (tales como impresión, acceso a bases de datos, fax y procesamiento de imágenes).

En la presente investigación se utilizará este modelo para realizar la comunicación, donde los dispositivos que serán simulados se comportarán como esclavos.

1.5. Comunicación mediante el protocolo TCP/IP

Entre los protocolos más populares utilizados para la comunicación sobre Ethernet, se encuentra el Protocolo de Control de Transmisión (TCP por sus siglas en inglés), el cual se caracteriza por garantizar que la transmisión enviada fue exitosamente recibida por el receptor; en otras palabras que todos los datos fueron recibidos en el mismo orden que fueron enviados. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto. TCP proporciona soporte a muchas de las aplicaciones más populares de Internet, incluidos el Protocolo de Transferencia de Hipertexto (HTTP), el Protocolo para la Transferencia Simple de Correo Electrónico (SMTP) y el Intérprete de Órdenes Seguras (SSH) [24].

Todos los datos en este tipo de comunicación viajan utilizando como capa de transporte al protocolo TCP y a nivel de red al Protocolo de Internet (IP).

1.6. Comunicación por puerto serial

El puerto serial permite la comunicación entre dos dispositivos a través de un canal exclusivo. Un bit se transmite durante un período de tiempo, por tanto, la velocidad en baudios (*baudrate*, en inglés) de un puerto serie de la computadora es igual al número de bits por segundo que se transmiten o reciben. Para enviar información codificada de esta manera, el transmisor y receptor deben estar a la misma frecuencia y sincronizados. Cada carácter se envía en un contenedor, que consiste en un bit “0” llamado bit de inicio, seguido por el mismo carácter y luego, opcionalmente, un bit de paridad y después un bit “1” llamado bit de paro [21]. La lógica del bit bajo de inicio le dice al receptor que está empezando un contenedor, y la lógica del bit alto de parada denota el final del mismo.

1.7. Algunos simuladores de dispositivos característicos

Con el fin de encontrar una herramienta que reuniera los requisitos necesarios para realizar las pruebas al manejador DF1, fueron estudiados varios simuladores. A continuación se mencionan y se explica el motivo por el cual no se utilizaron como solución.

Modsim: Esta herramienta está concebida para comportarse como un esclavo dentro de una red de comunicación Modbus, lo que significa que su protocolo de comunicación no es el utilizado por el manejador DF1, y aunque tiene soporte para puerto serie, no soporta registros de 32 bit o cadenas de caracteres. En estas condiciones, se hace imposible su uso como solución del problema.

MatrikonOPC Server for Omni Flow Computers: El servidor OPC para Computadores de flujo OMNI proporciona conectividad interoperable y segura de flujo OMNI 3000 y 6000 y sus equipos de medición de datos críticos. Como en el caso anterior, su protocolo de comunicación no es DF1, lo que significa que el simulador no se podría comunicar con este.

Además tiene la deficiencia de ser un software propietario que solamente tiene soporte para el sistema operativo Windows, esto significa que no se puede revisar ni modificar su código fuente.

Simulador Modbus RTU: Este simulador fue escrito originalmente para permitir las pruebas a un manejador Modbus Unidad Terminal Remota (RTU, por sus siglas en inglés) serial, sin embargo, creció y está creciendo para soportar otros protocolos entre los que se encuentran Modbus TCP/IP, Allen Bradley DF1 esclavo y maestro también [35]. Entre sus funcionalidades se destacan permitir la edición y la visualización de todos los registros, cargar y guardar valores que escriba en cada registro, así como simular cambios en los valores.

Este simulador está concebido para ejecutarse en el sistema operativo Windows, lo que imposibilita su modificación, esto trae como consecuencia que se deba descartar la posibilidad

de adecuarlo a las necesidades de la investigación.

Por otro lado aunque este simulador implementa la lógica de comunicación del protocolo DF1, no tiene definido un mecanismo de gestión de datos similar al mapa de memorias del dispositivo PLC-5, lo que impide que los datos que sean enviados al manejador DF1 en una transacción sean los correctos.

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 [36].

Este simulador no implementa la lógica de comunicación de protocolo DF1, además no cuenta con todos los registros que incorpora el mapa de memoria del PLC-5, por lo que no cuenta con los requisitos básicos para realizarle las pruebas al manejador DF1.

1.8. Herramientas seleccionadas para el desarrollo del simulador

Para el desarrollo del simulador que se propone como solución al problema planteado, se utilizaron tecnologías utilizadas y probadas por la línea de Adquisición del CEDIN. Entre las que se encuentra la biblioteca TransportProvider2 (desarrollada en la línea), utilizada para el transporte de datos, por su extensibilidad, flexibilidad y poseer una interfaz intuitiva.

Además se seleccionaron herramientas libres que brindan facilidades para el desarrollo. A continuación se hace una descripción de las características de las mismas.

1.8.1. Qt

Qt es un *framework* de desarrollo para aplicaciones multiplataforma que simplifica el desarrollo de aplicaciones en C++ de forma nativa [15]. Es una biblioteca que permite desarrollar interfaces gráficas de usuarios y programas sin interfaz gráfica como programas de consolas y servidores. Cuenta con métodos para acceder a bases de datos mediante SQL, así como el uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos, además de estructuras de datos tradicionales.

Qt creó un *plug-ins* con el cual es posible integrarlo en Eclipse. Incluye opciones integradas de depuración, editor de archivos de proyectos e incluso un editor de recursos. La integración de Qt con este Entorno Integrado de Desarrollo (IDE) funciona muy bien, sin embargo el gran consumo de recursos necesarios para su correcto funcionamiento es su mayor punto negativo.

1.8.2. Entorno Integrado de Desarrollo Eclipse

Eclipse es un IDE de código abierto, desarrollado inicialmente por IBM, aunque en la actualidad lo mantiene la Fundación Eclipse, organización independiente que fomenta otros productos de código abierto. Su valor recae en gran medida en los *plug-ins* que se integran a su plataforma facilitando el uso de herramientas y aprovechando las comodidades que este provee. Gracias a los *plug-ins* antes mencionados en el Eclipse es posible programar en C/C++, Python, Java y PHP. Además permite la integración con bibliotecas de desarrollo como es el caso de Qt. [17]
Eclipse también posee un excelente corrector de errores y un sistema para completar códigos. Ambos elementos facilitan el trabajo del programador. Por otro lado este IDE da soporte para sistema de gestión de bases de datos, control de versiones (CVS), subversión (SVN) y además para la realización de pruebas de unidad.

1.8.3. Lenguaje de programación C++

C++ es un lenguaje imperativo orientado a objetos derivado del C. En realidad un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. Este lenguaje de programación tiene una alta potencia para la programación a bajo nivel, y se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción. Entre las grandes ventajas de C++ se pueden mencionar la variedad tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones *inline*, sobrecarga de operadores, referencias y operadores para manejo de memoria persistente [18].

1.8.4. Visual Paradigm

Para el modelado de la solución se escogió como herramienta *CASE* (*Computer Aided Software Engineering* o Ingeniería de Software Asistida por Ordenador) el software Visual Paradigm, herramienta UML profesional que soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Es un software multiplataforma que permite representar diagramas de clases, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.
Presenta licencia gratuita y comercial. Es fácil de instalar, actualizar y compatible entre ediciones [11].

1.8.5. Lenguaje de Modelado Unificado (UML)

El lenguaje de modelado utilizado fue el UML (Lenguaje de Modelado Unificado, por sus siglas en inglés), el cual es un lenguaje gráfico que permite visualizar, especificar, construir y documentar un sistema [11]. Es utilizado para especificar y describir métodos o procesos. Además, ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

1.9. Metodologías de desarrollo de software

Las metodologías de desarrollo consisten en una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software [11]. Constituye una guía que nos indica cómo actuar y qué hacer cuando para lograr el resultado esperado en una investigación.

Normalmente el ciclo de vida del software es complejo, por eso se hace necesario el uso de metodologías que apoyen el desarrollo. La metodología que se utilizará debe que estar en correspondencia con las características que tenga el proyecto que se pretende desarrollar.

La finalidad de una metodología de desarrollo es garantizar la eficacia (cumplir los requisitos iniciales) y la eficiencia (minimizar las pérdidas de tiempo) en el proceso de generación de software.

En los últimos años se han desarrollado dos corrientes fundamentales en lo referente a los procesos de desarrollo, los métodos pesados y los métodos ligeros (también llamados métodos ágiles). La diferencia fundamental entre ambos métodos consiste en que los métodos pesados pretenden lograr sus objetivos mediante el orden y la documentación, mientras que los métodos ligeros intentan mejorar la calidad del software mediante la comunicación directa e inmediata entre las personas que intervienen en el proceso.

1.9.1. Proceso Unificado de Desarrollo (RUP)

Es una metodología de desarrollo de software que está basada en componentes e interfaces bien definidas, y junto con el UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Es un proceso que puede especializarse para una gran variedad de sistemas de software, en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización [28].

En su modelación define como sus principales elementos:

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

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

- **Inicio** (puesta en marcha)
- **Elaboración** (definición, análisis, diseño)
- **Construcción** (implementación)
- **Transición** (fin del proyecto y puesta en producción)

1.9.2. Metodología eXtreme Programming (XP)

Desarrollada por Kent Beck, la metodología XP se clasifica como ágil. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto [21]. Está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo y propiciando un buen clima de trabajo.

XP se basa en realimentación continua y comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico [4]. Intenta reducir la complejidad del software por medio de un

trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

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

- Exploración.
- Planificación de la Entrega (*Release*).
- Iteraciones.
- Producción.
- Mantenimiento.
- Muerte del Proyecto.

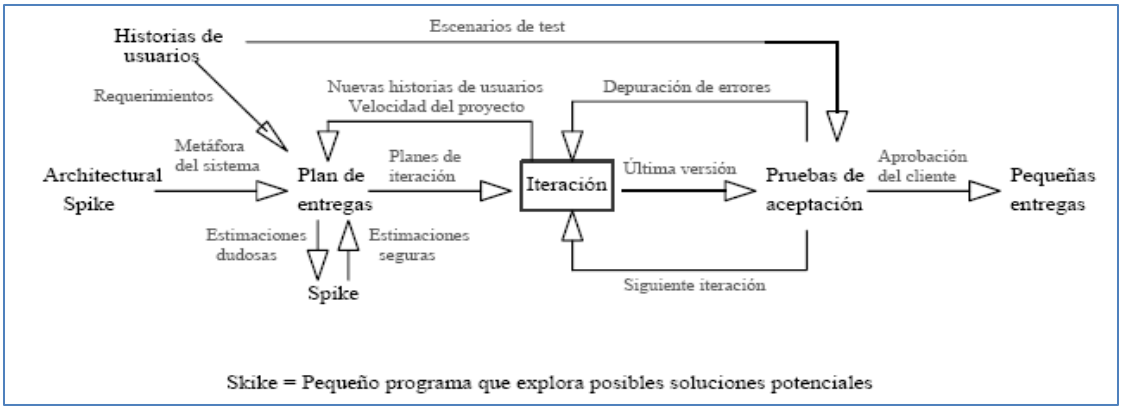

Figura 7: Ciclo de desarrollo de XP.

Esta metodología define, como bases del desarrollo de software, “Historias de Usuarios”, que escribe el cliente y describen escenarios sobre el funcionamiento del software, pueden describir el modelo, dominio, entre otros aspectos. A partir de las Historias de Usuarios (HU) y de la arquitectura perseguida, se crea un plan de entregables entre el equipo de desarrollo y el cliente. [4]

XP no produce demasiados gastos sobre las actividades de desarrollo, y no impide el avance de los proyectos, reduce el costo del cambio en las etapas de vida del sistema y es muy recomendable para equipos de trabajo pequeños, de 2 a 15 personas, donde los programadores pueden ser ordinarios.

1.9.3. Metodología SCRUM-XP (SXP)

SXP es una metodología ágil de desarrollo de software. Está compuesta por las metodologías SCRUM y XP que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de

preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo [19].

Consta de 4 fases principales:

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

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario, diseño, implementación, pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, lo que nos permite mejorar el diseño cada vez que se le añada una nueva funcionalidad.

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes pueden ver día a día cómo progresa el trabajo.

1.10. Selección de la metodología de desarrollo de software

Para el desarrollo del simulador de dispositivos PLC-5, se escogió la metodología ágil XP, puesto que se ajusta a las condiciones actuales del equipo de desarrollo, y está enfocada a satisfacer por completo las necesidades del cliente. Teniendo en cuenta que tiene como pilar de funcionamiento la comunicación directa y constante entre ambas partes, permite definir los requisitos de una manera clara y eficiente. Además, la metodología XP está diseñada para adaptarse a un ambiente en el que los cambios ocurren de forma periódica, donde el cliente llega a formar parte del equipo de desarrollo, lo que posibilita una rápida retroalimentación.

Por los elementos anteriormente planteados, además de la sencillez en su aprendizaje y aplicación, la metodología XP constituye una guía de desarrollo segura y eficaz, con la cual se pretende alcanzar los objetivos trazados al comienzo de esta investigación.

1.11. Conclusiones parciales

Como resultado de la revisión de varios documentos y estudios teóricos relacionados con la simulación y los simuladores de dispositivos, se mostró la relación, importancia y ventajas que tienen ambos elementos. Además con el análisis de los dispositivos PLC-5, se definieron características específicas que deberán estar presentes en el diseño de la solución propuesta, como son el protocolo de comunicación DF1 y el mapa de memoria. Se analizaron simuladores de dispositivos acordes a la investigación, determinando que no contaban con las características necesarias para realizarle pruebas al manejador DF1.

Por otro lado, se definieron las herramientas y la metodología de desarrollo a ser utilizadas en la implementación del simulador.

Capítulo 2: Descripción y diseño del simulador

Introducción

En el presente capítulo se realiza una propuesta de solución del sistema a partir del análisis y la descripción del proceso de comunicación que existe entre el manejador DF1 y los dispositivos PLC-5, así como el comportamiento que pueden presentar ambos actores en determinadas circunstancias. Se utilizaron las tres primeras fases de la metodología XP: Exploración, Planificación de la Entrega, e Iteraciones; a partir de las cuales se definen las Historias de Usuarios, el Plan de Iteraciones, el Plan de Entregables y las Tareas de Ingeniería.

Además, se define la arquitectura que se utilizará y se modelan los atributos y funcionalidades de las clases que deberán conformar la aplicación que se propone como solución a la problemática.

2.1. Descripción del proceso real vinculado al campo de acción

Con el objetivo de simular el proceso de comunicación del manejador DF1 con los dispositivos PLC-5, se realizó un análisis para lograr un modelo con el cual se puedan representar la características y comportamientos fundamentales que deben tener estos dispositivos.

En el proceso real, el manejador DF1 deberá permitir la interacción del SCADA SAINUX con los dispositivos de campo, en este caso los PLC-5, permitiéndole recolectar la información necesaria para controlar los procesos industriales que ocurran en una determinada fábrica o entidad. Para lograr la interacción antes mencionada, el manejador DF1 tiene que, primeramente, establecer la comunicación con los dispositivos de campo con que se cuenten funcionalmente, a través del puerto serial, y luego realizar el envío de peticiones utilizando el protocolo DF1 en uno de sus modos de comunicación, ya sea Full-duplex o Half-duplex. Se debe tener en cuenta que el manejador siempre tendrá la característica de maestro, pues será en todos los casos quien inicie una transacción de datos, y por su parte, el dispositivo de campo toma un carácter de esclavo al estar siempre en espera de peticiones.

En este contexto, el manejador puede realizar operaciones de lectura y escritura sobre los PLC-5, los cuales tienen la capacidad de procesar el flujo de peticiones recibidas y enviar las respuestas al maestro, además de controlar diversas actividades; para esto cuentan con varios elementos como se explicó en el capítulo 1, entre ellos el mapa de memoria.

Los mensajes enviados tanto por el manejador DF1 como por los dispositivos de campo deben tener la estructura requerida por el protocolo que se utilice.

Es necesario destacar que en la comunicación antes descrita pueden ocurrir fallas y errores, y los actores que intervienen en ella deben estar preparados para estas afectaciones.

2.1.1. Transferencia de mensajes mediante Full-duplex

A continuación se representan algunos comportamientos comunes de un dispositivo PLC-5 que se comunica mediante el protocolo Full-duplex. El transmisor representa al manejador DF1 y el receptor al dispositivo de campo. Los bytes de datos de la capa de enlace están representados por “xxxx” y los datos corruptos con “???”.

- Transferencia de un mensaje normal:
  - El transmisor envía los datos al receptor.
  - La pila envía un mensaje comunicando que no está llena.
  - El receptor envía los datos a la pila y envía un DEL ACK al transmisor.
  - El transmisor le comunica a la fuente que el mensaje fue entregado.

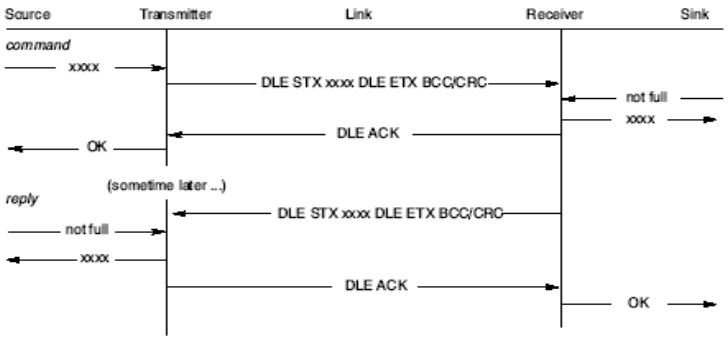

Figura 8: Transferencia normal.

- Transferencia de un mensaje con NAK:
  - El transmisor envía datos corruptos al receptor, y este responde con un DLE NAK.
  - El transmisor retransmite y la transmisión es correcta.
  - El receptor envía un DLE ACK al transmisor.
  - La respuesta es correctamente retornada.

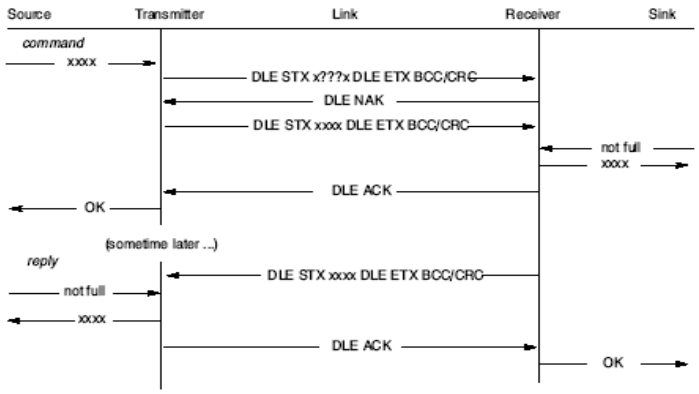

Figura 9: Transferencia con NAK.

- Transferencia de un mensaje con *time out* (tiempo excedido) y ENQ:
  - El receptor recibe correctamente el mensaje pero el DLE ACK que envía es corrupto.
  - El transmisor da un *time out* esperando el DLE ACK y envía un DLE ENQ.
  - El receptor envía un DLE ACK.

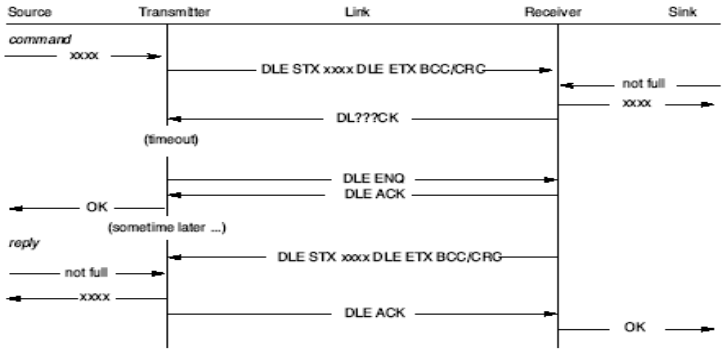

Figura 10: Transferencia con *time out* y ENQ.

- Transferencia de un mensaje con pila de mensajes llena:
  - El transmisor envía un mensaje pero la pila de mensajes del receptor se encuentra llena, y el receptor envía de respuesta un DLE NAK.
  - El transmisor retransmite y la pila de mensajes ya no está llena, y el receptor responde con un DLE ACK.
  - La respuesta es correctamente retornada.

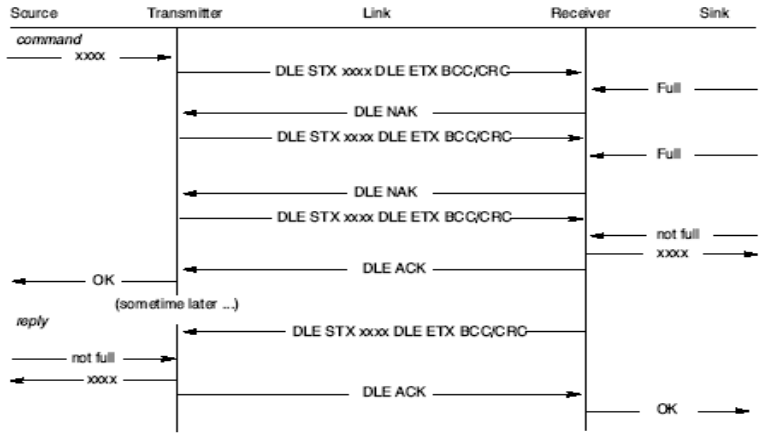

Figura 11: Transferencia con pila de mensajes llena.

- Transferencia de un mensaje con NAK como respuesta:
  - El mensaje es correctamente transmitido por la red.
  - La respuesta es corrupta y el transmisor responde con un DLE NAK.
  - La respuesta se envía nuevamente y es correcta.

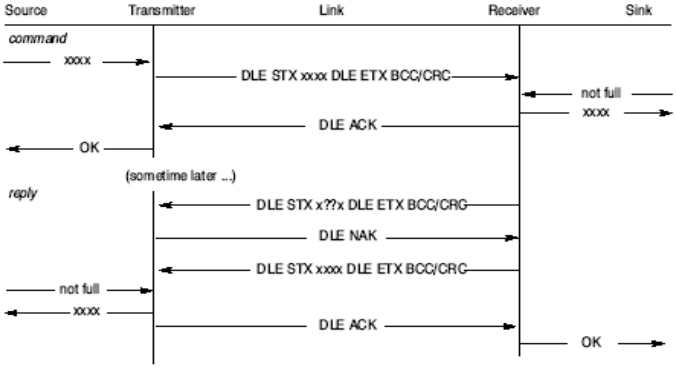

Figura 12: Transferencia con NAK como respuesta.

2.2. Propuesta de solución

Después de realizar el análisis del proceso real, se determinó que el simulador de dispositivos PLC-5, deberá contar con un grupo de características específicas que permitan realizar las pruebas requeridas al manejador DF1, de las cuales la comunicación mediante el protocolo DF1 en el modo Full-duplex, y la gestión de datos a través del mapa de memoria tienen la mayor relevancia.

Se propone como solución un sistema que se comporte como esclavo en la comunicación y tenga la capacidad de conectarse por puerto serial o por Ethernet, recibir, procesar y responder a cada petición enviada por el manejador DF1. Para esto el simulador debe tener la opción de crear canales de comunicación, sobre los cuales uno o más dispositivos puedan tener conexión.

Además los canales de comunicación deben permitir al usuario seleccionar el modo de comunicación que utilizarán, ya sea Full-duplex o Half-duplex.
Los dispositivos simulados contarán con un mapa de memoria individualmente, sobre el cual se ejecutarán los comandos de escritura o lectura recibidos. El mapa de memoria permitirá agregar o eliminar memorias, con la condición de que nunca se repitan memorias del mismo tipo. Por otro lado, la aplicación deberá simular fallas en la transmisión, como es la ocurrencia de ruido y el retraso de respuestas, así como la falla en la comunicación con los dispositivos o en el canal de comunicación.

2.3. Fase de exploración

El ciclo de desarrollo del software, según la metodología XP, comienza con esta fase, en la cual los clientes plantean a grandes rasgos las Historias de Usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizaran en el proyecto.
En esta fase además, se definen los requisitos no funcionales del sistema.

2.3.1. Historias de Usuario (HU)

Las Historias de usuarios son descripciones cortas y escritas en el lenguaje del usuario, donde el nivel de detalle debe ser el mínimo posible, para que de manera sencilla se determine cuánto costará la implementación del sistema. El cliente redacta, según su consideración, los requisitos que debe tener la aplicación, mediante las HU, expresando de esta manera su punto de vista en cuanto a las necesidades del sistema.
A continuación se muestran las Historias de Usuarios definidas.

Tabla 6: Historia de Usuario 1.

HISTORIA DE USUARIO	
Número: 1	**Usuario:** Linea de Adquisición
Nombre historia: Gestionar y configurar canales serial y TCP.	
Prioridad en negocio: Alta	**Registro en desarrollo:** Alta
Descripción:	
Se debe brindar la opción visual al usuario de Crear, Eliminar y Configurar canales de comunicación serial y TCP.	
Observaciones:	

Tabla 7: Historia de Usuario 2.

HISTORIA DE USUARIO	
Número: 2	**Usuario:** Linea de Adquisición
Nombre historia: Utilizar el tipo de comunicación Full-duplex o Half-duplex	
Prioridad en negocio: Alta	**Registro en desarrollo:** Alta
Descripción:	
Se debe brindar la opción visual al usuario de seleccionar el protocolo que será utilizado para la comunicación de un canal específico.	
Observaciones:	

Tabla 8: Historia de Usuario 3.

HISTORIA DE USUARIO	
Número: 3	**Usuario:** Linea de Adquisición
Nombre historia: Gestionar y configurar las propiedades de los dispositivos.	
Prioridad en negocio: Media	**Registro en desarrollo:** Media
Descripción:	
Se debe brindar la opción visual al usuario de Crear, Eliminar y Configurar los dispositivos sobre un canal de comunicación.	
Observaciones:	

Tabla 9: Historia de Usuario 4.

HISTORIA DE USUARIO	
Número: 4	**Usuario:** Linea de Adquisición
Nombre historia: Gestionar y configurar los mapas de memoria.	
Prioridad en negocio: Alta	**Registro en desarrollo:** Media
Descripción:	
Se debe brindar la opción visual al usuario de Crear y Eliminar un mapa de memorias por dispositivos, permitiendo configurar las memorias que contendrá.	
Observaciones:	

Tabla 10: Historia de Usuario 5.

HISTORIA DE USUARIO	
Número: 5	**Usuario:** Linea de Adquisición
Nombre historia: Simular fallas de comunicación en canal.	
Prioridad en negocio: Media	**Registro en desarrollo:** Media
Descripción:	
El usuario puede seleccionar un canal de comunicación e indicarle que simule la caida de todos los dispositivos que se comunican a través de él.	
Observaciones:	

Tabla 11: Historia de Usuario 6.

HISTORIA DE USUARIO	
Número: 6	**Usuario:** Linea de Adquisición
Nombre historia: Simular errores de tramas.	
Prioridad en negocio: Media	**Registro en desarrollo:** Media
Descripción:	
Se le debe brindar la posibilidad al usuario de seleccionar un canal de comunicación o un dispositivo determinado e indicarle que simule errores en las tramas.	
Observaciones:	
Cuando se seleccione un canal, todos los dispositivos que se encuentren conectados sobre él, simularan errores en las tramas de envío.	

Tabla 12: Historia de Usuario 7.

HISTORIA DE USUARIO	
Número: 7	**Usuario:** Linea de Adquisición
Nombre historia: Simular fallas de comunicación con el dispositivo.	
Prioridad en negocio: Media	**Registro en desarrollo:** Media
Descripción:	
Se le debe brindar la posibilidad al usuario de seleccionar un dispositivo determinado e indicarle que simule su caida.	
Observaciones:	

Tabla 13: Historia de Usuario 8.

HISTORIA DE USUARIO	
Número: 8	**Usuario:** Linea de Adquisición
Nombre historia: Simular retraso en las comunicaciones.	
Prioridad en negocio: Media	**Registro en desarrollo:** Media
Descripción:	
Se le debe brindar la posibilidad al usuario de seleccionar un canal de comunicación e indicarle que simule retraso en las comunicaciones de los dispositivos.	
Observaciones:	

2.3.2. Requisitos no funcionales del sistema

Los requisitos no funcionales especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos. Son propiedades o cualidades que reflejan las características que hacen al producto atractivo, usable, rápido y confiable.

Como requerimientos del software se necesita el Sistema Operativo GNU/Linux (distribución

Debian Squeeze), así como la instalación de la biblioteca de transporte TransportProvider2 y el *framework* Qt. Como requerimientos del hardware es necesario como mínimo un ordenador con al menos 512 MB de RAM. Además, se establece como requerimiento de apariencia, que la aplicación tendrá una interfaz gráfica amigable e intuitiva.

2.4. Fase de planificación de la entrega

En esta fase se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Primeramente el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizaron una estimación del esfuerzo necesario para desarrollar cada una de ellas.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida puntos de estimación. Un punto, equivale a una semana ideal de programación [22]. Para esto se definió que las historias valdrían de 1 a 3 puntos.

2.4.1. Estimación de esfuerzo

En este epígrafe se muestra la estimación de esfuerzo asociado a la implementación de las HU, realizada por el equipo de desarrollo. Las HU están ordenadas de acuerdo a la prioridad que escogió el cliente en función de sus necesidades.

Es importante que se tenga en cuenta que para estas estimaciones se utilizan las historias de usuarios que especifiquen características funcionales del sistema.

Tabla 14: Estimación de esfuerzo por HU.

HU	Estimación de esfuerzo (puntos)
1	3
2	1
3	2
4	2
5	1
6	1
7	1
8	1

2.5. Fase de iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de Iteraciones son: Historias de Usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior [22]. Teniendo en cuenta lo planteado, se definió que el desarrollo del sistema tendrá tres iteraciones, como se explica a

continuación.

- **Plan de Iteraciones**
  - **Iteración 1**: Tiene como objetivo cumplir las Historias de Usuarios 1 y 2, dado la importancia que tienen para la aplicación y ser las HU de mayor prioridad para el cliente. Están enfocadas a la lógica de comunicación del sistema, a ser implementada a través de canales exclusivos y compartidos que proporcionen el envió y la recepción de datos. En esta iteración se diseñan e implementan los componentes visuales de la interfaz de usuario que se relacionan con las HU antes mencionadas.
  - **Iteración 2**: En esta iteración se implementan las funcionalidades básicas para la gestión de información en las transacciones de datos de los dispositivos simulados, mediante los mapas de memoria que presentan cada uno de ellos. Además, se desarrollan los componentes visuales que permiten al usuario interactuar con estos elementos, cumpliendo de esta manera las Historias de Usuarios 3 y 4.
  - **Iteración 3**: En esta iteración se pretende implementar funcionalidades del sistema, que permitan al usuario activar simulaciones de fallos en la transmisión. Abarca las Historias de Usuarios 5, 6, 7 y 8, debido a que todas están centradas en la simulación de fallas.

A continuación se presenta en la Tabla 15 el Plan de duración de iteraciones, el cual tiene como finalidad mostrar la duración de cada iteración, así como el orden en que serán implementadas las Historias de Usuarios.

Tabla 15: Plan de duración de iteraciones.

Iteraciones	HU	Duración
1	1	3 semanas
	2	
2	3	3 semanas
	4	
3	5	3 semanas
	6	
	7	
	8	

2.6. Plan de entregables

Una vez definidas las Historias de Usuarios y las iteraciones que tendrá el desarrollo, se propone un Plan de entregables, el cual tiene como objetivo especificar las entregas de funcionalidades que se harán en correspondencia con una HU y la iteración en que se realizará.

Tabla 16: Plan de entregables.

HU	Iteración 1	Iteración 2	Iteración 3
1	Entregable 1		
2	Entregable 1		
3		Entregable 2	
4		Entregable 2	
5			Entregable 3
6			Entregable 3
7			Entregable 3
8			Entregable 3

Una vez establecido el Plan de entregables se definen las versiones que se realizarán en cada iteración según las fechas estimadas, hasta alcanzar el producto final.

Tabla 17: Versiones.

Entregable	Iteración 1 2da semana de marzo	Iteración 2 2da semana de abril	Iteración 3 2da semana de mayo
1	Versión 1	Versión 2	Finalizada
2		Versión 1	Finalizada
3			Finalizada

2.7. Tareas de Ingeniería

A partir de cada Historia de Usuario se generan Tareas de Ingeniería que responden a los diferentes pasos que se deben ejecutar para satisfacer o cumplir las funcionalidades planteadas en las HU mencionadas inicialmente. Las tareas generadas son realizadas por programadores dentro del equipo de trabajo, aplicando normalmente la práctica de programación en pareja. Como el equipo de desarrollo está compuesto por un programador solamente, éste será el encargado de realizar cada Tarea de Ingeniería.

La siguiente tabla muestra las Tareas de Ingeniería que se derivaron de cada Historia de Usuario. La descripción de cada una de tareas se presenta en el siguiente capítulo, vinculadas a la iteración en que son realizadas.

Tabla 18: Tareas de Ingeniería.

HU	Tareas de Ingeniería
1	Diseño del Explorador de Canales (Channel Explorer).
	Implementación de la lógica de funcionamiento del Explorador de Canales.
	Diseño e implementación de los elementos visuales para la configuración de los canales.
2	Implementación de los protocolos Full-duplex y Half-duplex.
	Implementar la opción visual de selección de protocolo a usar.
3	Implementación de la clase Dispositivo (Device).
	Diseño e implementación de los elementos visuales para la configuración de los dispositivos.
4	Implementación de la clase Mapa de memoria (MemoryMap).
	Diseño e implementación visual de la gestión de memorias en los mapas de memoria.
5	Implementación de fallas en canal.
6	Implementación de fallas de trama.
7	Implementación de fallas de comunicación de los dispositivos.
8	Implementación de retraso en las comunicaciones.

2.8. Arquitectura del sistema

La arquitectura de software juega un papel fundamental en el diseño y la implementación de estructuras de software de alto nivel [26]. Es resultado de unir varios elementos arquitectónicos en función de satisfacer las funcionalidades y requerimientos de un sistema. Los patrones arquitectónicos constituyen un ejemplo de los elementos antes mencionados, proporcionando plantillas de trabajo que rigen el diseño de la estructura general de un sistema [28], permitiendo además que los miembros del equipo de desarrollo lo observen de una manera similar.

Para el simulador de dispositivos PLC-5 se propone una arquitectura basada en el patrón arquitectónico el Modelo-Vista-Controlador (MVC), el cual permite separar los datos de la aplicación (modelo), la interfaz de usuario (vista), y la lógica de control (controlador) en tres componentes distintos; donde el controlador funciona como intermediario entre el flujo de datos entre el modelo y la vista, como se ilustra a continuación.

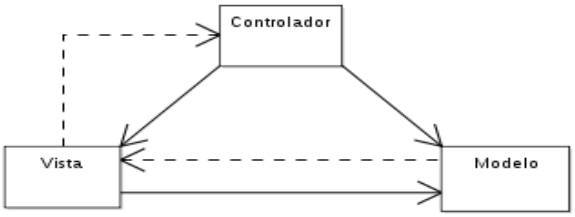

Figura 13: Patrón Modelo-Vista-Controlador.

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar [32]. En el simulador estará compuesto por un grupo de clases, que permitirán manejar los datos de la aplicación, mediante el uso de ficheros donde se guardarán

los datos de configuración, así como los registros de datos que debe gestionar la aplicación.
Para la interacción con el usuario, la vista utiliza una interfaz gráfica, la cual tendrá de forma amigable e intuitiva las opciones requeridas por el cliente. Además mediante herramientas visuales permite conocer el estado de las comunicaciones y los eventos que ocurren en el sistema.
La lógica de control del sistema radica principalmente en los canales de comunicación, los cuales implementan la lógica de comunicación, así como la ejecución de los comandos recibidos. Además permiten la interacción del usuario con las propiedades de los dispositivos.

2.9. 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 [33]. Se decir que son soluciones a diferentes clases de problemas conocidos que pueden ser aplicadas a diferentes códigos [30].
Entre los requisitos que debe cumplir una solución para ser considerada un patrón, debe contar con ciertas características como son la efectividad, por haber resuelto problemas similares con anterioridad, y ser reusable, o sea, ser aplicable a diferentes problemas de diseño en diversas circunstancias.

2.9.1. Método Factoría

El patrón Método Factoría define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos. El mismo es utilizado en esta investigación específicamente en la clase Channel, la cual deja en sus clases hijas, la responsabilidad de creación de objetos de las clases FullMessage y HalfMessage, a través del método createMessage(), como se puede apreciar en la Figura 16.

2.9.2. Patrones GRASP

GRASP es un acrónimo que significa *General Responsibility Asignment Software Patterns* (patrones generales de software para asignar responsabilidades), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones.
A continuación se describen aquellos patrones utilizados de acuerdo a las soluciones que brindan dentro del contexto del sistema: [34]

Tabla 19: Patrones GRASP.

Nombre del patrón	Características
Patrón Experto	Asignan una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad
Patrón Bajo Acoplamiento	Asignan las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible.
Patrón Alta Cohesión	Asignan a las clases responsabilidades para que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad. • Muy baja cohesión: Una clase es la única responsable de muchas cosas en áreas funcionales muy heterogéneas. • Baja cohesión: Una clase tiene la responsabilidad exclusiva de una tarea compleja dentro de un área funcional. • Alta cohesión: Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.
Patrón Controlador	Asignan la responsabilidad del manejo de mensajes de los eventos del sistema a una clase.
Patrón Creador	Asignan a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos: • B agrega los objetos A. • B contiene los objetos A. • B registra las instancias de los objetos A. • B utiliza especialmente los objetos A. • B tiene los datos de inicialización que serán transmitidos a A cuando este objeto

	sea creado (así que B es Experto respecto a la creación de A). B es un creador de los objetos A. Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A.

2.10. Tarjetas CRC

Las tarjetas Clase-Responsabilidades-Colaboradores (CRC) representan una técnica de la metodología XP, muy útil para realizar el diseño de un sistema ya que permiten representar objetos, sus responsabilidades y los colaboradores que utiliza para cumplir su tarea.

A continuación se presentan las tarjetas CRC que tienen mayor relevancia en el funcionamiento del sistema. Las tarjetas restantes se pueden observar en el Anexo 1.

- Clases del Modelo:

Tabla 20: Tarjeta CRC: Clase Device.

Clase: Device
Responsabilidades: Se encarga de representar los dispositivos PLC-5 y sus propiedades. Además esta clase permite la gestión de los ficheros de datos (memorias) de su mapa de memoria y el casteo de los datos almacenados por este.
Colaboradores: MemoryMap.

Tabla 21: Tarjeta CRC: Clase MemoryMap.

Clase: MemoryMap
Responsabilidades: Representar los mapas de memoria y sus propiedades. Clase encargada de la gestión directa de las memorias que pueden almacenar los dispositivos PLC-5.
Colaboradores: Memory, MemoryInteger, MemoryFloating, MemoryTimer, MemoryPID.

Tabla 22: Tarjeta CRC: Clase Memory.

Clase: Memory
Responsabilidades: Interface encargada de representar de forma abstracta las propiedades de los ficheros de datos que utiliza el mapa de memoria

delos dispositivos simulados.
Colaboradores:

Tabla 23: Tarjeta CRC: Clase SingleMemory.

Clase: SingleMemory
Responsabilidades: Interface encargada de representar de forma genérica las propiedades de los ficheros de datos simples (con tamaño de 1 o 2 words).
Colaboradores: Memory.

Tabla 24: Tarjeta CRC: Clase StructMemory.

Clase: StructMemory
Responsabilidades: Interface encargada de representar de forma genérica las propiedades de los ficheros de datos estructurados (con tamaño de 3 o más words).
Colaboradores: Memory.

- Clases de la Vista:

Tabla 25: Tarjeta CRC: Clase MainWindow.

Clase: MainWindow
Responsabilidades: Clase encargada de contener y organizar todos los elementos visuales del sistema.
Colaboradores: QMainWindow, ExplorerTree, ExplorerItem, ChannelTab, MemoryTab, SimInspector, Controller.

Tabla 26: Tarjeta CRC: Clase ExplorerTree.

Clase: ExplorerTree
Responsabilidades: Clase encargada de organizar, a través de un árbol visual, los elementos visuales que representa canales de comunicación, los dispositivos que lo utilizan y sus mapas de memoria.
Colaboradores: QTreeWidget, ExplorerItem, MainWindow.

Tabla 27: Tarjeta CRC: Clase ExplorerItem.

Clase: ExplorerItem
Responsabilidades: Clase encargada de representar visualmente los canales de comunicación, los dispositivos que lo utilizan y sus mapas de memoria.
Colaboradores: QTreeWidgetItem, Channel, Device, Memory.

Tabla 28: Tarjeta CRC: Clase ChannelTab.

Clase: ChannelTab
Responsabilidades: Componente visual mediante el cual se pueden configuras las propiedades de los canales serial y TCP.
Colaboradores: QWidget, Channel.

Tabla 29: Tarjeta CRC: Clase MemoryTab.

Clase: MemoryTab
Responsabilidades: Componente visual mediante el cual se puede visualizar y modificar los valores de las memorias utilizadas.
Colaboradores: QWidget, QAbstractTableModel, Memory.

- Clases del Controlador:

Tabla 30: Tarjeta CRC: Controler.

Clase: Controler
Responsabilidades: Se encarga de contener los elementos relacionados con el control del sistema. Permite la creación y eliminación de canales de comunicación serial y TCP.
Colaboradores: DF1Message, ITransportHandler, ITransport, ITransportProvider, Device.

Tabla 31: Tarjeta CRC: SerialChannel.

Clase: SerialChannel
Responsabilidades: Permite la configuración de las propiedades del canal de comunicación serial.
Colaboradores: ITransporProvider, ITransport, Channel.

Tabla 32: Tarjeta CRC: TCPChannel.

Clase: TCPChannel
Responsabilidades: Permite la configuración de las propiedades del canal de comunicación TCP.
Colaboradores: ITransporProvider, ITransport, Channel.

Tabla 33: Tarjeta CRC: Clase Channel.

Clase: Channel
Responsabilidades: Interface encargada de representar de forma abstracta las propiedades y funcionalidades de los canales que implementan las lógicas de comunicación Full-duplex y Half-Duplex.
Colaboradores: DF1Message, ITransportHandler, ITransport, ITransportProvider, Device.

Tabla 34: Tarjeta CRC: Clase FullChannel.

Clase: FullChannel
Responsabilidades: Clase encargada de implementar la lógica de comunicación del protocolo Full-duplex.
Colaboradores: Channel, IStopCondition.

Tabla 35: Tarjeta CRC: Clase HalfChannel.

Clase: HalfChannel
Responsabilidades: Clase encargada de implementar la lógica de comunicación del protocolo Half-duplex.
Colaboradores: IStopCondition.

2.11. Diagrama de clases

Los diagramas de clases permiten representar las relaciones, atributos y funcionalidades que poseen las clases del sistema. La metodología XP descarta la realización de diagramas de clase para realizar el diseño del sistema, sin embargo el equipo de desarrollo decidió utilizarlas para lograr una mayor comprensión de la solución propuesta.

2.11.1. Relación entre clases del Modelo

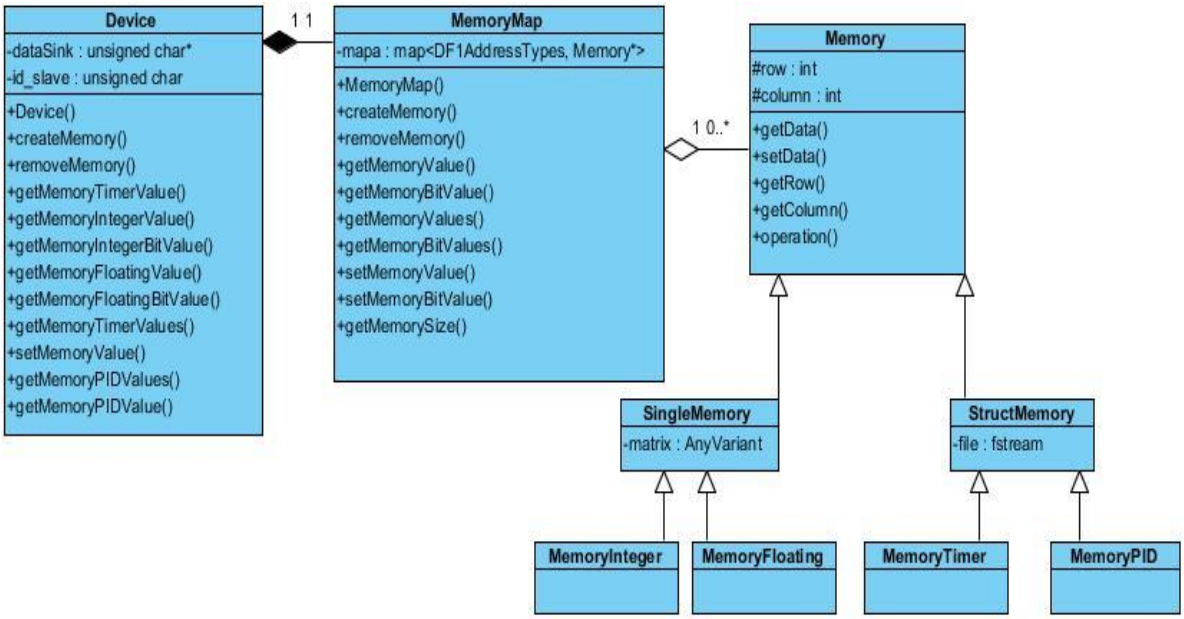

Figura 14: Clases del Modelo.

2.11.2. Relación entre clases de la Vista

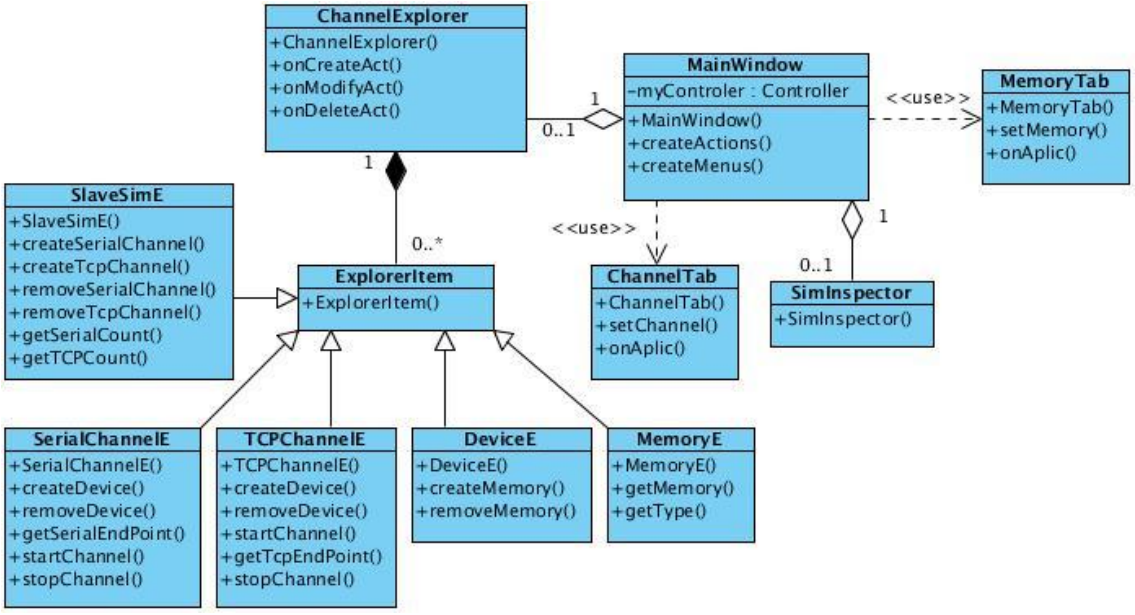

Figura 15: Clases de la Vista.

2.11.3. Relación entre clases del Controlador

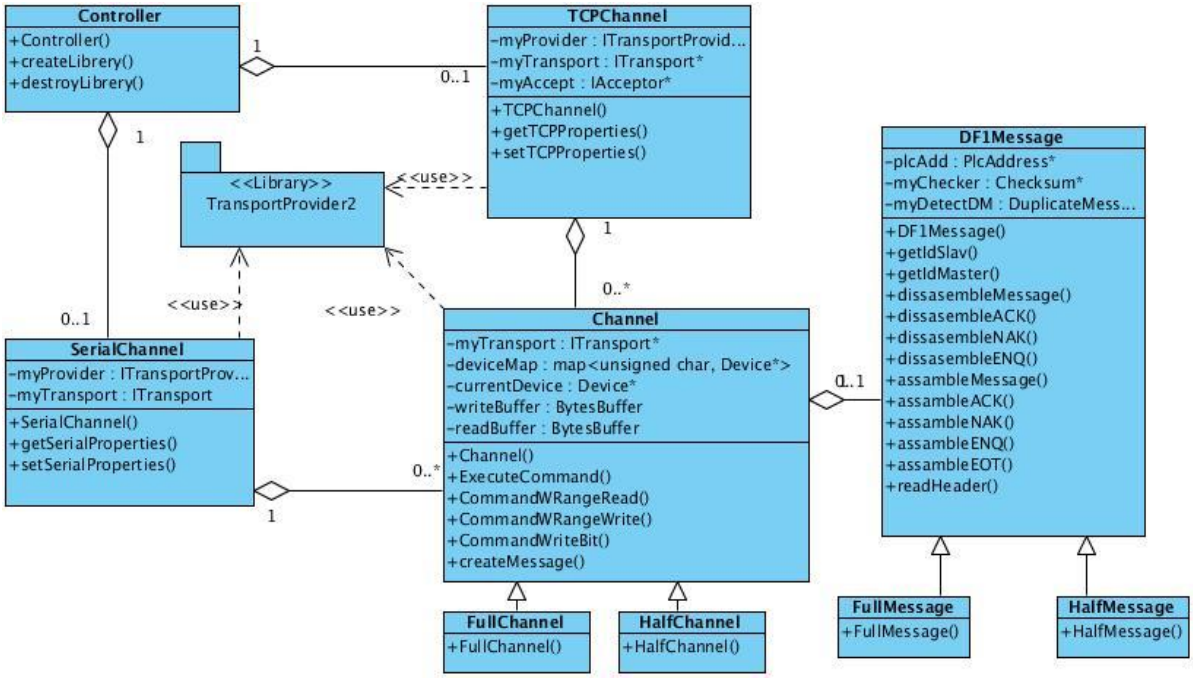

Figura 16: Clases del Controlador.

2.12. Conclusiones parciales

Una vez descrito el proceso real, se propuso como solución un simulador que permite simular comportamientos de los dispositivos PLC-5. Se realizó un levantamiento de un total de ocho Historias de Usuario, lo que permitió la elaboración de trece Tareas de Ingeniería que serán implementadas para dar cumplimiento a los requisitos funcionales del sistema. Se definieron además tres iteraciones para el desarrollo del sistema y se establecieron las fechas de entrega de versiones una vez elaborado el Plan de Entregables. Además se realizó el diseño del sistema, basado en una arquitectura Modelo-Vista-Controlador, utilizando las tarjetas CRC y los diagramas de clases.

Capítulo 3: Implementación y prueba

Introducción

A partir del diseño realizado en el capítulo anterior se inicia la implementación del sistema, la cual se debe realizar de forma iterativa e incremental, según la metodología XP. De manera que se obtenga un producto al final de cada iteración, que debe ser probado y mostrado al cliente, permitiendo la retroalimentación continua entre los involucrados en el desarrollo.

En el presente capítulo se describen las Tareas de Ingeniería relacionadas con cada iteración definida. Además, se muestran las pruebas realizadas al sistema, que permitieron la aceptación, por parte del cliente, de cada versión del sistema.

3.1. Fases de la implementación

En el capítulo anterior se definieron un conjunto de Historias de Usuarios las cuales deben ser cumplidas mediante las Tareas de Ingeniería, las cuales son implementadas por los programadores responsables en las diferentes iteraciones de desarrollo.

Seguidamente se detallan las tres iteraciones de desarrollo que deberán realizarse para alcanzar finalmente el producto deseado por el cliente.

3.1.1. Iteración 1

La primera iteración en el desarrollo del sistema está enfocada a darle cumplimiento a las Historias de Usuario 1 y 2. Para esto fueron asignadas un total de cinco tareas de Ingeniería al programador del sistema, con las cuales se implementan las funcionalidades de mayor relevancia para el cliente.

3.1.1.1. Tareas de Ingeniería realizadas en la primera iteración

Tabla 36: Tarea de Ingeniería 1.

TAREA DE INGENIERÍA	
Número: 1	**Número de historia:** 1
Nombre de tarea: Diseño del Explorador de Canales (Channel Explorer).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 16/2/2013	**Fecha fin:** 20/2/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
La interface de usuario cuenta con un panel a su parte izquierda, donde se mostrarán los canales de comunicación y los dispositivos que estén siendo usados.	

Tabla 37: Tarea de Ingeniería 2.

TAREA DE INGENIERÍA	
Número: 2	**Número de historia:** 1
Nombre de tarea: Implementación de la lógica de funcionamiento del Explorador de Canales.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 22/3/2013	**Fecha fin:** 27/3/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Cuando se presiona el botón derecho del mouse sobre el elemento Simulador PLC-5 del Explorador de Canales, brinda la opción de agregar un canal de comunicación el cual puede ser serie o TCP. El menú contextual de un canal existente, permite la adición de dispositivos. Una vez creados alguno de estos elementos, si el usuario hace doble clic sobre uno de ellos, se muestra una ventana que permite modificar su configuración.	

Tabla 38: Tarea de Ingeniería 3.

TAREA DE INGENIERÍA	
Número: 3	**Número de historia:** 1
Nombre de tarea: Diseño e implementación de los elementos visuales para la configuración de los canales.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 29/2/2013	**Fecha fin:** 3/3/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Si se hace doble clic izquierdo sobre un canal creado, se muestra en el elemento visual Propiedades de Elementos opciones que permiten la configuración del canal seleccionado, ya sea serie como TCP.	

Tabla 39: Tarea de Ingeniería 4.

TAREA DE INGENIERÍA	
Número: 4	**Número de historia:** 2
Nombre de tarea: Implementación de los protocolos Full-duplex y Half-duplex.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 4/3/2013	**Fecha fin:** 12/3/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Se implementa la lógica de comunicación de los protocolos Full-duplex y Half-duplex mediante las clases Channel, FullChannel y HalfChannel. De esta manera se proporciona la transacción de datos al sistema.	

Tabla 40: Tarea de Ingeniería 5.

TAREA DE INGENIERÍA	
Número: 5	**Número de historia:** 2
Nombre de tarea: Implementar la opción visual de selección de protocolo a usar.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 14/3/2013	**Fecha fin:** 16/3/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Cuando se pretende crear un canal el sistema genera una ventana de diálogo la cual permite escojer el tipo de canal (serie o TCP), y el protocolo de comunicación a utilizar por dicho canal.	

3.1.2. Iteración 2

En esta iteración se utilizan cuatro Tareas de Ingeniería, que dan cumplimiento a las Historias de Usuario 3 y 4, enfocadas en la gestión de datos del sistema.

3.1.2.1. Tareas de Ingeniería realizadas en la segunda iteración

Tabla 41: Tarea de Ingeniería 6.

TAREA DE INGENIERÍA	
Número: 6	**Número de historia:** 3
Nombre de tarea: Implementación de la clase Dispositivo (Device).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 20/3/2013	**Fecha fin:** 25/3/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Se implementa la clase Device la cual tiene como objetivo principal el acceso a datos del sistema. Se implementa la pila de mensajes a enviar.	

Tabla 42: Tarea de Ingeniería 7.

TAREA DE INGENIERÍA	
Número: 7	**Número de historia:** 3
Nombre de tarea: Diseño e implementación de los elementos visuales para la configuración de los dispositivos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 26/3/2013	**Fecha fin:** 30/3/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Se diseña un elemento visual que represente un dispositivo de campo, se implementa la funcionalidad de que al dar doble clic sobre él aparezca un cuadro de diálogo que permita visualizar y modificar las propiedades del dispositivo seleccionado.	

Tabla 43: Tarea de Ingeniería 8.

TAREA DE INGENIERÍA	
Número: 8	**Número de historia:** 4
Nombre de tarea: Implementación de la clase Mapa de memoria (MemoryMap).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha inicio: 2/4/2013	**Fecha fin:** 8/4/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Se deben implementar ademas las clases que representan los ficheros de datos a utilizar, seguidamente la clase MemoryMap, y las funcionalidades que debe poseer para crear, eliminar y modificar ficheros de datos. Esta clase debe funcionar como interface entre los ficheros de datos y el sistema, puesto que cuentan con registros de diversos tamaños.	

Tabla 44: Tarea de Ingeniería 9.

TAREA DE INGENIERÍA	
Número: 9	**Número de historia:** 4
Nombre de tarea: Diseño e implementación visual de la gestión de memorias en los mapas de memoria.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10/4/2013	**Fecha fin:** 14/4/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Se diseña e implementa el componente visual MemoryTab el cual muestra a través de componentes visuales las características de una memoria seleccionada, creada previamente en el Explorador de Canales y asociada a un dispositivo.	

3.1.3. Iteración 3

En la presente iteración se pretende proporcionar al sistema de un conjunto de funcionalidades que permitan la simulación de fallas. Para esto se asignaron un total de 4 Tareas de Ingeniería que cumplen las HU 5, 6, 7 y 8.

3.1.3.1. Tareas de Ingeniería realizadas en la tercera iteración

Tabla 45: Tarea de Ingeniería 10.

TAREA DE INGENIERÍA	
Número: 10	**Número de historia:** 5
Nombre de tarea: Implementación de fallas en canal.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17/4/2013	**Fecha fin:** 22/4/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Cuando se presiona el clic derecho sobre un canal, se brinda la opción de generar simulación de fallas sobre el canal, en este caso todos los dispositivos que se encuentren conectados por medio del canal, se desconectarán.	

Tabla 46: Tarea de Ingeniería 11.

TAREA DE INGENIERÍA	
Número: 11	**Número de historia:** 6
Nombre de tarea: Implementación de fallas de trama.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 23/4/2013	**Fecha fin:** 27/4/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Cuando se presiona el clic derecho sobre un canal o un dispositivo, se brinda la opción de generar simulación de fallas en tramas de mensajes. En el caso del canal, activa la falla de tramas en todos los dispositivos que se encuentren sobre el canal escogido. Las fallas de mensajes pueden activarse para los mensajes enviados o los que se reciben.	

Tabla 47: Tarea de Ingeniería 12.

TAREA DE INGENIERÍA	
Número: 12	**Número de historia:** 7
Nombre de tarea: Implementación de fallas de comunicación de los dispositivos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 1/5/2013	**Fecha fin:** 5/5/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Cuando se presiona el clic derecho sobre un dispositivo, se brinda la opción de generar simulación de fallas en la comunicación de dicho dispositivo. El dispositivo tiene un mecanismo que permite simular su apagado, desconectado, etc.	

Tabla 48: Tarea de Ingeniería 13.

TAREA DE INGENIERÍA	
Número: 13	**Número de historia:** 8
Nombre de tarea: Implementación de retraso en las comunicaciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 7/5/2013	**Fecha fin:** 12/5/2013
Programador responsable: Miguel Angel Albuerne Rivero	
Descripción:	
Cuando se presiona el clic derecho sobre un canal o un dispositivo, se brinda la opción de generar simulación de retraso en las comunicaciones. En el caso del canal, activa el retraso en todos los dispositivos que se encuentren sobre el canal escogido.	

3.2. Diagrama de despliegue

Los diagramas de despliegue muestran la disposición física de los nodos que componen el sistema y el reparto de los componentes sobre dichos nodos, modelando la topología de hardware sobre la cual se ejecutará el sistema. Estos nodos representan un recurso computacional que generalmente cuenta con memoria y capacidad de procesamiento. [12]

En la Figura 17 se puede apreciar el nodo PC Cliente, que representa el ordenador donde se estará ejecutando el manejador DF1. El nodo Simulador PLC-5, representa la computadora donde se ejecuta el simulador desarrollado, la misma debe tener como sistema operativo alguna distribución GNU/Linux y contar con al menos 512 MB de RAM.

La comunicación entre ambos nodos se puede realizar a través del puerto serie RS 232, o

Ethernet utilizando el protocolo TCP/IP.

Figura 17: Diagrama de despliegue.

3.3. Pruebas realizadas

Para comprobar la correcta implementación de las funcionalidades, fueron realizadas un conjunto de pruebas a la aplicación con el objetivo de identificar y corregir fallos u omisiones cometidas. La metodología XP propone como mecanismo de detección de errores las Pruebas de Aceptación, las cuales son realizadas por el cliente en compañía de al menos un integrante del equipo de desarrollo y se orientan a las funcionalidades del sistema. Son diseñadas a partir de las HU y tiene como objetivo 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.

A continuación se muestra las Pruebas de Aceptación más relevantes realizadas al sistema.

Tabla 49: Caso de prueba No. 1.

PRUEBA DE ACEPTACIÓN	
Caso de prueba: 1	**Número de historia:** 1
Nombre de prueba: Crear canales de comunicación.	
Descripción: Prueba que verifica la creación de canales de comunicación serial o TCP/IP.	
Condiciones de ejecución: Se debe haber cargado la biblioteca de transporte correctamente.	
Entrada/Pasos de ejecución: En el menú contextual del elemento Simulador PLC-5 dentro del árbol Explorador de Canales, se selecciona Crear canal, luego se selecciona el tipo de comunicación y protocolo a utilizará.	
Resultado esperado: El canal creado debe aparecer en el Explorador de Canales.	
Evaluación de la prueba: Satisfactoria.	

Tabla 50: Caso de prueba No. 2.

PRUEBA DE ACEPTACIÓN	
Caso de prueba: 2	**Número de historia:** 1
Nombre de prueba: Visualizar y modificar propiedades de los canales.	
Descripción: Prueba que verifica la funcionalidad de gestionar la configuración de los canales serial y TCP/IP.	
Condiciones de ejecución: Se debe haber creado al menos un canal correctamente.	
Entrada/Pasos de ejecución: Se hace doble clic sobre un canal creado en el Explorador de Canales, y se modifican las propiedades a conveniencia mediante el componente visual Propiedades de Elementos. Seguidamente se hace clic sobre el botón Aplicar configuración.	
Resultado esperado: El sistema debe enviar el mesaje: "Configuración aplicada correctamente".	
Evaluación de la prueba: Satisfactoria.	

Tabla 51: Caso de prueba No. 3.

PRUEBA DE ACEPTACIÓN	
Caso de prueba: 3	**Número de historia:** 1
Nombre de prueba: Eliminar canales de comunicación.	
Descripción: Esta prueba valida la funcionalidad de eliminar canales serial o TCP/IP creados.	
Condiciones de ejecución: Se debe haber creado al menos un canal correctamente.	
Entrada/Pasos de ejecución: En el menú contextual de un canal se selecciona la opción Eliminar canal, el sistema genera un cuadro de diálogo donde pregunta si realmente quiere eliminar el canal. Seguidamente se selecciona el botón Yes.	
Resultado esperado: El elemento visual que representa el canal debe desaparecer del Explorador de Canales, asi como los elementos asociados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 52: Caso de prueba No. 4.

PRUEBA DE ACEPTACIÓN	
Caso de prueba: 4	**Número de historia:** 3
Nombre de prueba: Crear dispositivo.	
Descripción: Comprueba que la aplicación permita crear dispositivos dentro de los canales de comunicación.	
Condiciones de ejecución: Debe estar creado al menos un canal.	
Entrada/Pasos de ejecución: Se selecciona un canal existente y seguidamente se selecciona en su menú contextual la opción Crear dispositivo. Luego se introduce el ID que tendrá el nuevo dispositivo, y se presiona el botón OK.	
Resultado esperado: Se agrega un componente visual al Explorador de Canales que representa el dispositivo creado.	
Evaluación de la prueba: Satisfactoria.	

Tabla 53: Caso de prueba No. 5.

PRUEBA DE ACEPTACIÓN	
Caso de prueba: 5	**Número de historia:** 4
Nombre de prueba: Crear memoria.	
Descripción: Comprueba que la aplicación permita crear memorias dentro de los dispositivos.	
Condiciones de ejecución: Debe estar creado al menos un canal con un dispositivo asocionado.	
Entrada/Pasos de ejecución: Se selecciona un dispositivo existente y seguidamente se selecciona en su menú contextual la opción Crear memoria. Se introducen los valores de las características de la nueva memoria y se presiona el botón OK.	
Resultado esperado: Se agrega un componente visual al Explorador de Canales que representa la memoria creada creado.	
Evaluación de la prueba: Satisfactoria.	

Tabla 54: Caso de prueba No. 6.

PRUEBA DE ACEPTACIÓN	
Caso de prueba: 6	**Número de historia:** 2
Nombre de prueba: Ejecutar la espera de peticiones de los canales.	
Descripción: Esta prueba valida la capacidad de comunicación de los canales dado que ejecuta la lógica de comunicación implementada.	
Condiciones de ejecución: Debe estar creado al menos un canal, con un dispositivo y memorias asociadas, además de estar desconectado.	
Entrada/Pasos de ejecución: Se selecciona un canal existente y seguidamente se hace clic izquierdo en la opción Comenzar conexión, dentro de la barra de tareas Conexión.	
Resultado esperado: Al lado del canal seleccionado se inserta el cartel "Conectado".	
Evaluación de la prueba: Satisfactoria.	

3.4. Conclusiones parciales

Luego de la implementación del sistema a través de las Tareas de Ingeniería y la posterior realización de los diferentes casos de pruebas definidos, se determinó que el simulador desarrollado cumple el objetivo general de la presente investigación, reuniendo los requisitos necesarios para lograr una correcta simulación de los dispositivos PLC-5 en la transferencia de datos mediante el protocolo DF1.

Conclusiones

Al finalizar la presente investigación sobre el desarrollo del simulador de dispositivos PLC-5, se hace posible arribar a las siguientes conclusiones:

- Con el simulador desarrollado se obtuvo una aplicación que simula el comportamiento de los dispositivos PLC-5 cuando realizan transacciones de datos.
- La implementación del protocolo DF1 y la gestión de memorias a través de los dispositivos, proporcionó al sistema la capacidad de recibir, procesar y responder peticiones de datos, dando cumplimiento a los requisitos de mayor importancia para el cliente.
- Fueron implementadas funcionalidades al sistema que permiten la simulación de fallas en la comunicación, haciendo posible la detección de errores en el comportamiento del manejador que se prueba.
- Las pruebas realizadas a la aplicación permitieron validar su correcto funcionamiento, confirmando el cumplimiento del objetivo general de la investigación.

A través del sistema desarrollado es posible la realización de pruebas al manejador DF1, que permitan la detección de posibles errores en su funcionamiento y validen su implementación para ser utilizado por el SCADA SAINUX.

Recomendaciones

Para darle continuidad a la investigación desarrollada el autor recomienda:

- Agregar nuevas funcionalidades que permitan a la aplicación contar con un mayor número de fallas a simular.
- Incorporar al simulador la capacidad de mostrar detalladamente los mensajes recibidos y enviados, con el objetivo de comprobar el correcto estado de los datos utilizados en cada transacción.
- Utilizar la investigación realizada como referencia en el desarrollo de simuladores de dispositivos en la Universidad de las Ciencias Informáticas.
- Desarrollar un simulador de dispositivos genérico, que permita cargar dispositivos con diferentes características, como el protocolo de comunicación y la memoria utilizada.

Bibliografía

1. **Himmelblau, David M. y Bischoff, Kenneth B.** *Análisis y simulación de procesos. : .* Barcelona : Reverté, S.A, 1976. 84-291-7235-1. Disponible en: http://books.google.com.cu/books?isbn=8429172351

2. **Berger Vidal, Esther, Gambini López, Inés y Velázquez Pino, Carmela.** *Simulación de sistemas.* 2000. Disponible en: http://sisbib.unmsm.edu.pe/bibvirtualdata/libros/Matematicas/Notas_instituto/Simulacion_sistemas.pdf

3. **Shannon, Robert E y Johannes, James D.** *Systems Simulation: The Art and Science.* SERBIULA - Universidad de Los Andes, Venezuela : OAI Repositorio (Sistema LIBRUM), 1975. ISSN 0018-9472. Disponible en: http://books.google.com.cu/books/about/Systems_simulation.html?id=cWpRAAAAMAAJ&redir_esc=y

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

5. **GUEVARA AVALOS, GLORIA F. y ROJAS ZAVALA, ROBINSON A.** *IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE REFRIGERACIÓN MEDIANTE LAZO CERRADO CON UN CONTROLADOR Y SENSOR EN EL MÓDULO DE AUTOMATIZACIÓN CON PANTALLAS TÁCTILES.* RIOBAMBA – ECUADOR : s.n., 2011. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/1668

6. **Vallejo, Horacio D.** *Los Controladores Lógicos Programables.* 2005. Disponible en: http://www.todopic.net/utiles/plc.pdf

7. **Owono Marti, Daniel.** *Simulación y control de procesos automatizados con PLCs OMRON y LabVIEW.* Catalunya : s.n., 2011. Disponible en: http://upcommons.upc.edu/pfc/handle/2099.1/11919

8. **Hernández León, Rolando Alfredo.** *El Proceso de Investigación Científica.* Ciudad de La Habana : Editorial Universitaria, 2011. 978-959-16-1307-3. Disponible en:

9. **Allen-Bradley.** *Addressing Reference Manual: DF1 Protocol and Command Set.* 1995. Disponible en: http://revistas.mes.edu.cu/greenstone/collect/repo/import/repo/20110113/9789591613073.pdf

10. **Allen-Bradley.** *Reference Manual: 1785 PLC-5 Programmable Controllers.* 1996. Disponible en: http://www.efesotomasyon.com/Allen_Bradley_rockwell/5000-rm002_-en-p.pdf

11. **Uriarte Rodriguez, Pedro.***Manejador para la comunicacion con dispositivos de campo mediante el protocolo DF1.* Cuidad de La Habana : s.n., 2010. Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_03836_10

12. **Guerra, Javier Lorié.** *Capa de acceso a datos para dispositivos Bristol.* Ciudad de La Habana : s.n., 2011.

13. **De La Cruz, Iván Fernando.** *Estudio de las comunicaciones de un controlador lógico programable (PLC) a través de una interface serial.* Latacunga : s.n., 2003. Disponible en: http://repositorio.espe.edu.ec/handle/21000/4370

14. **Chacón Morales, Ricardo Salvador.** *Simulación SCADA (Control, supervisión y adquisición de datos) de una planta generadora de energía eléctrica a base de energía geotérmica.* San Salvador : s.n., 2012. Disponible en: http://ri.ues.edu.sv/1778/

15. **Blanchette, Jasmin y Summerfield, Mark.** *C++ GUI Programming with Qt 4, Second Edition.* 2008. 0132354160. Disponible en: http://katunblock.com/c-gui-programming-with-qt-4-2nd-edition-sucax-t7311758.html

16. ***De Antonio, Angélica y Villalobos*, M. y Luna, E**. *Cuándo y cómo usar la Realidad Virtual en la enseñanza*. Universidad de Guadalajara : s.n., 2000. Disponible en: http://www.geocities.ws/blancajrm/ihai/Virtual/DOC8.htm

17. **Eclipse Foundation.** *The Eclipse Foundation open source community website.* 2013. Disponible en: http://www.eclipse.org

18. **García de Jalón, Javier**. *Aprenda C++ como si estuviera en primero*. San Sebastian : s.n., 1998. Disponible en: http://mat21.etsii.upm.es/ayudainf/aprendainf/Cpp/manualcpp.pdf

19. **Kniberg, Henrik.** *Scrum y Xp desde las trincheras.* s.l. : C4Media, 2007. 978-1-4303-2264-1. Disponible en: http://infoq.com/minibooks/scrum-xp-fromthetrenches

20. **Reátegui Gabancho, Humberto.** *Sistema Redundante de supervisión y control de despacho de combustibles de CB No 5 –Refinería Talara. Capítulo IV.* 2007. Disponible en: http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Ingenie/reategui_gh/cap4.pdf

21. **ZIMMERANN, H.** *OSI Reference Model-The ISO Model of Architecture for Open System Interconnections.* 1980. Disponible en: http://dl.acm.org/citation.cfm?id=59310

22. **Letelier, Patricio y Penadés, Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : s.n. Disponible en: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm

23. **Segreda Johanning, Juan Andrés.** *Unidad control programable vía Ethernet.* Cartago : s.n., 2011. Disponible en: http://bibliodigital.itcr.ac.cr:8080/xmlui/handle/2238/2980

24. **Comer, Douglas.** *Interconectividad de redes con TCP/IP. Volumen II.* 2005. Disponible en: http://www.docstoc.com/docs/34796937/Redes-Globales-de-Informaci%C3%B3n-con-Internet-y-TCPIP

25. **Automation, Rockwell.** *Guía de selección del sistema de controladores programables PLC-5.* s.l. : Rockwell Automation, 2004. 1785-SG001A-ES-P. Disponible en: http://literature.rockwellautomation.com/idc/groups/literature/documents/sg/1785-sg001_-es-p.pdf

26. **Pérez Reyes, Carlos Miguel.** *FISIM: SIMULADOR FÍSICO – MATEMÁTICO INTEGRADO*

A LA PLATAFORMA DE GESTIÓN DEL APRENDIZAJE ZERA. La Habana : s.n., 2011. Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_04632_11

27. Rockwell Automation. [En línea] 2013. Disponible en: http://www.rockwellautomation.com.

28. Pressman, Roger S. *Ingeniería del software. Un enfoque práctico.*s.l. : Mc Graw Hill, 2005. 9701054733. Disponible en: http://www.intercambiosvirtuales.org/libros-manuales/ingenieria-del-software-un-enfoque-practico-roger-pressman-sexta-edicion

29. Nokia, Qt. *Qt SDK - Qt - A cross-platform application and UI framework*. [En línea] 2013. [Citado el: 20 de abril de 2013.]. Disponible en: http://qt.nokia.com/products/qt-sdk.

30. Delgado, Karel; Jiménez, Alejandro. *Módulo HMI para una tarjeta basada en microcontroladores*. La Habana : s.n., 2012.

31. Cuba, Enelis. *Fusión de la información de los servicios de la comunidad universitaria de la UCI con el metaverso OpenSim*. La Habana : s.n., 2012.

32. *El patrón de diseño Modelo-Vista-Controlador y su implementación en Java Swing.* **Bascon, Ernesto.** s.l. : Revistas Bolivianas, 2004, Vol. II. 1683-0789. Disponible en: http://ucbconocimiento.ucbcba.edu.bo/index.php/ran/article/download/84/81

33. Vidal, Santiago. *IMPLEMENTACIÓN DEL SIMULADOR DEL TEMA DE GESTIÓN DE DISPOSITIVOS DE ENTRADA Y SALIDA PARA EL LABORATORIO VIRTUAL DE LA ASIGNATURA SISTEMAS OPERATIVOS.* La Habana : s.n., 2011.

34. CIBERAULA.JAVA. *Patrones de Diseño en aplicaciones Web*. 2010, [Citado el: 24 de abril 2013]. Disponible en: http://java.ciberaula.com/articulo/diseno_patrones_j2ee

35. CODEGURU. *MODBUS Serial RTU + TCP/IP Simulator*. 2003, [Citado el: 10 de mayo 2013]. Disponible en:

http://www.codeguru.com/cpp/i-n/network/serialcommunications/article.php/c5401/MODBUS-Serial-RTU--TCPIP-Simulator.htm

36. SOURCEFORCE. *Dexter v.1.0.0.1*. 2012, [Citado el: 10 de mayo 2013]. Disponible en: http://www.winsite.com/Home-Education/Science/Dexter/

37. Peñalver Romero. *Metodología Ágil para proyectos de software libre*. Ciudad de La Habana : s.n., 2008.

Glosario

A

Allen-Bradley: Es la marca de una línea de equipos de automatización de fábricas, creada por la compañía Rockwell Automation.

ANSI: Viene de las siglas en inglés de *American National Standards Institute*, que significa Instituto Nacional Estadounidense de Estándares y llamado comúnmente ANSI, el cual es una organización encargada de supervisar el desarrollo de normas para los servicios, productos, procesos y sistemas en los Estados Unidos. El ANSI forma parte de la Organización Internacional para la Estandarización (ISO) y de la Comisión Electrotécnica Internacional (IEC).

ASCII: Se refiere al código ASCII (siglas en inglés de *American Standard Code for Information Interchange*, es decir Código Americano (estadounidense) Estándar para el intercambio de Información), 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 inglés moderno y en otras lenguas occidentales.

C

CASE: Las herramientas CASE (*Computer Aided Software Engineering*, o en español Ingeniería de Software Asistida por Computadora) son diversas 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.

CPU: Acrónimo en inglés de *Central Processing Unit*, que en español significa Unidad Central de Procesamiento o Microprocesador, es el componente en un ordenador, que interpreta las instrucciones y procesa los datos contenidos en los programas de la computadora.

D

Data Highway +: Se refiere a la red *Data Highway Plus* (DH+). Es una red de área local diseñada para trabajar con programación remota y adquisición de datos para aplicaciones de la planta. También se pueden usar módulos de comunicación remota para implementar una red pequeña de dispositivos similares.

F

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.

I

IBM: Acrónimo en inglés de *International Business Machines* es una empresa multinacional estadounidense de tecnología y consultoría. IBM 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.

L

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

M

Modbus: Protocolo de comunicaciones situado en el nivel 7 (capa de aplicación) del Modelo OSI, basado en el modelo maestro/esclavo. [4] Es un protocolo público, exige poco desarrollo y permite el control de una red de dispositivos.

O

OMNI: Se refiere al protocolo de comunicación, Modbus Omni, desarrollado por *Enron Corporation*. El mismo comparte grandes similitudes con el protocolo Modbus. Las principales diferencias entre los dos protocolos son la numeración de las direcciones de registro, el soporte de registros de 32 bits y cadenas de caracteres, así como la capacidad de transmitir registros de eventos y datos históricos.

R

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

ROM: Se refiere a la memoria de sólo lectura, normalmente conocida por su acrónimo *Read Only Memory (ROM)*. Es un medio de almacenamiento utilizado en ordenadores y otros dispositivos electrónicos. La ROM no se puede modificar al menos no de manera rápida o fácil y se utiliza principalmente para contener el firmware software que está estrechamente ligado a un hardware específico o PC.

Rockwell Automation: Es un proveedor mundial de soluciones de automatización, control e informatización industrial. *Rockwell Automation* es una de las mayores compañías de automatización industrial en el mundo, empleando a cerca de 21.000 personas en más de 80 países, e incluyen marcas como *Allen-Bradley* y *Rockwell Software*.

RS-232: Se refiere al modelo estándar del puerto serie o puerto serial. Es una interfaz de comunicaciones de datos digitales, frecuentemente utilizada por computadoras y periféricos,

donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits simultáneamente. La comparación entre la transmisión en serie y en paralelo se puede explicar usando una analogía con las carreteras.

S

SCADA SAINUX: Se refiere al Sistema de Automatización Industrial UX (las siglas UX provienen del desarrollo de aplicaciones sobre el sistema operativo Linux), producto en desarrollo del CEDIN.

T

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

W

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

Anexo 1

Tabla 55: Tarjeta CRC: Clase MemoryInteger.

Clase: MemoryInteger
Responsabilidades: Representa todos los fichero de datos con un tamaño de 1 word (2 bytes).
Colaboradores: SingleMemory.

Tabla 56: Tarjeta CRC: Clase MemoryFloating.

Clase: MemoryFloating
Responsabilidades: Representa todos los fichero de datos con un tamaño de 2 words (4 bytes).
Colaboradores: SingleMemory.

Tabla 57: Tarjeta CRC: Clase MemoryTimer.

Clase: MemoryTimer
Responsabilidades: Representa todos los fichero de datos con un tamaño de 3 words(6 bytes).
Colaboradores: StructMemory.

Tabla 58: Tarjeta CRC: Clase MemoryPID.

Clase: MemoryPID
Responsabilidades: Representa todos los fichero de datos con un tamaño de 82 words (164 bytes).
Colaboradores: StructMemory.

Tabla 59: Tarjeta CRC: Clase SimInspector.

Clase: SimInspector
Responsabilidades: Componente visual mediante el cual se pueden visualizar y controlar las simulaciones creadas por el usuario.
Colaboradores:

Tabla 60: Tarjeta CRC: DF1Message.

Clase: DF1Message
Responsabilidades: Clase encargada de desensamblar y ensamblarlos campos comunes de los mensajes Full-duplex y Half-duplex.
Colaboradores: PlcAddress, DuplicateMessage, Checksum.

Tabla 61: Tarjeta CRC: FullMessage.

Clase: FullMessage
Responsabilidades: Clase encargada de desensamblar y ensamblar los campos específicos de los mensajes Full-duplex.
Colaboradores:

Tabla 62: Tarjeta CRC: HalfMessage.

Clase: HalfMessage
Responsabilidades: Clase encargada de desensamblar y ensamblar los campos específicos de los mensajes Half-duplex.
Colaboradores:

Anexo 2

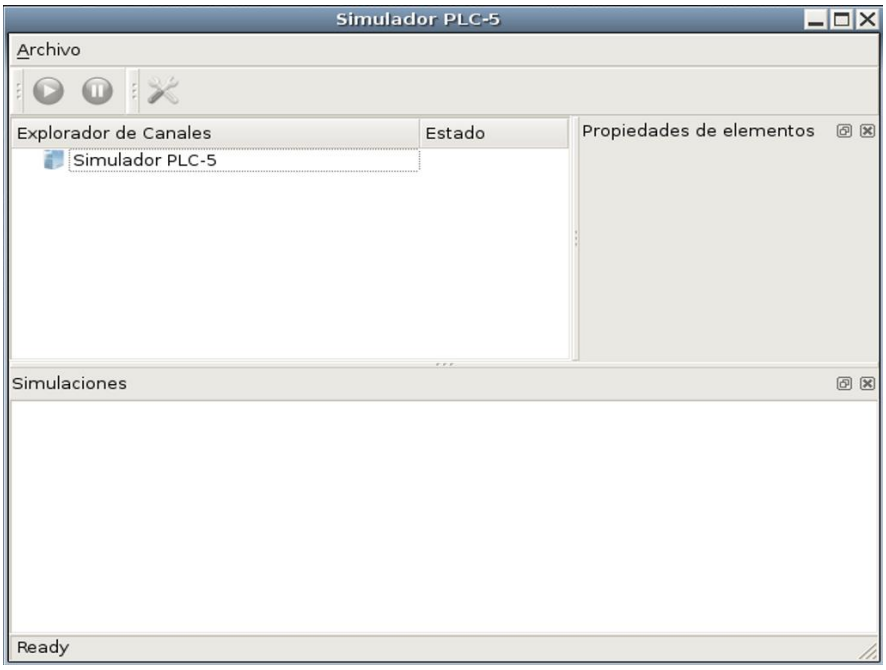

Figura 18: Simulador PLC-5.

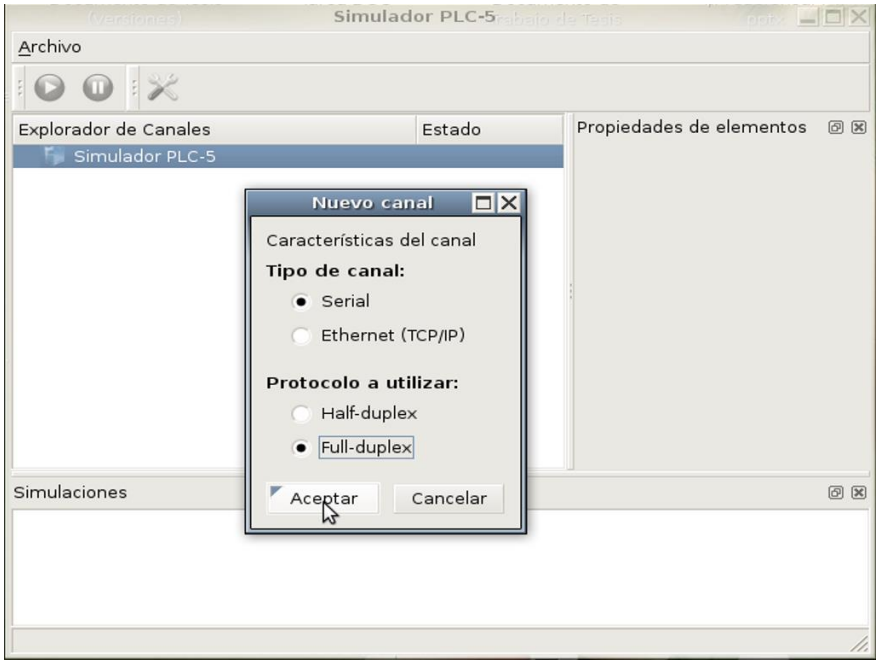

Figura 19: Creando un canal.

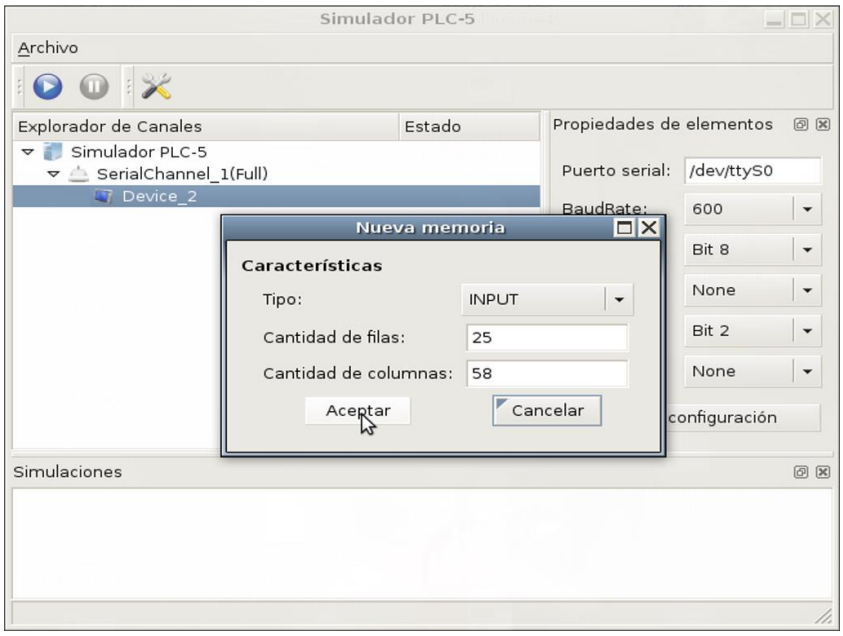

Figura 20: Creando memoria.

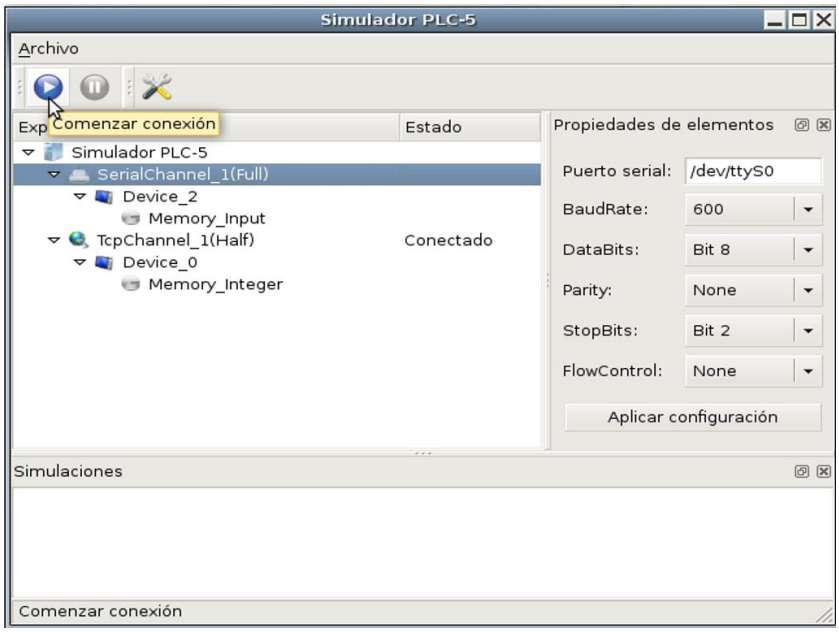

Figura 21: Conectando canales.