FACULTAD 4

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

SIMULADOR PARA EL PROTOCOLO DE COMUNICACIÓN INDUSTRIAL BSAP SERIAL

AUTOR:

José Luis Medero Bermúdez

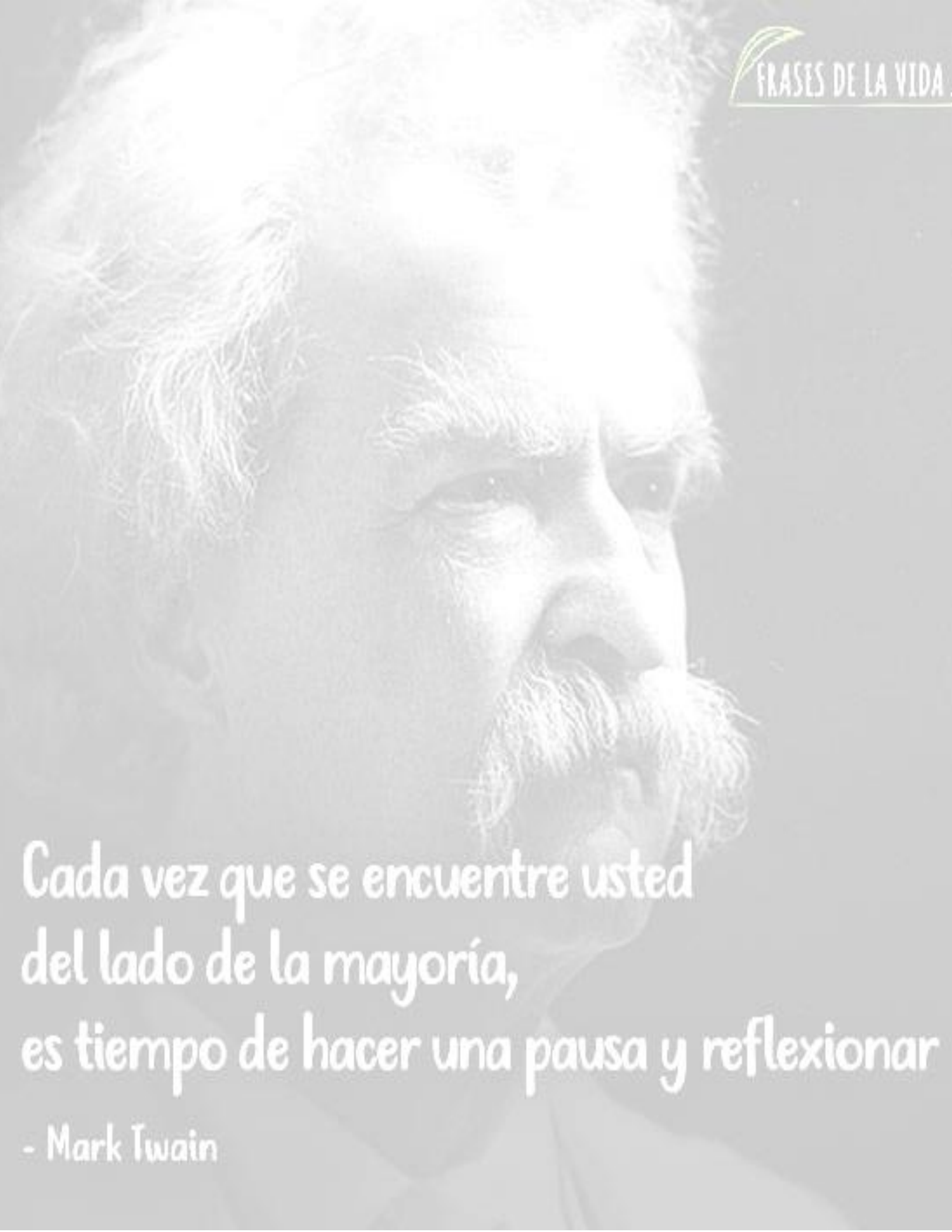
TUTORES:

MSc. Andy Hernández Paez

Ing. Alberto Arístides Puentes

La Habana, 2018

Año 60 de la Revolución



Cada vez que se encuentre usted
del lado de la mayoría,
es tiempo de hacer una pausa y reflexionar

- Mark Twain

Declaración de autoría

Declaro ser el único autor de este trabajo y autorizo a la Universidad de Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste, firmo el presente a los ___ días del mes de _____ del año _____.

_____

José Luis Medero Bermúdez

Firma de Autor

_____

MSc. Andy Hernández Paez

Firma de Tutor

_____

Ing. Alberto Arístides Puentes

Firma de Tutor

Dedicatoria

A mis padres, mis abuelos y mi hija que son las personas más importantes en mi vida, y para ellos van dedicados todos mis triunfos.

Agradecimientos

Dedicar este trabajo a 2 personas que desgraciadamente hoy no se encuentran entre nosotros por caprichos de la vida, a mi amigo más viejo Miguel, papa de unos de mis mejores amigos y a mi hermano Raymil.

Agradecer a mi mama porque gracias a ella hoy estoy aquí, por estar siempre a mi lado y apoyarme incondicionalmente sin poner nunca una queja. Gracias

A mi papa que no dejo de llamarme ni un solo día para saber cómo estaba y se necesitaba algo y para preguntarme si ya había comido. Gracias

A mi abuelo que más que un abuelo es un padre porque no se ha despegado de mí nunca, siempre dando los mejores consejos aunque sabe que conmigo eso es muy complicado. Gracias

A mi abuela que no se ha cansado se mimarme día tras día y que ha sufrido tanto o más que yo toda mi carrera. Gracias

A mi amigos de toda la vida Fabio, Papo y Osmani, literalmente toda la vida, porque estamos juntos desde los 5 años, me siento privilegiado de contar con su amistad, más que amigos, son mis hermanos. Gracias

Dicen que la amistad entre un hombre y una mujer no existe, aquí me di cuenta que no es cierto, porque me llevo 2, Jessica, no sé qué sería de mi aquí sin ella, mi mejor confidente, la persona con la que puede contar para lo que fuera en cualquier momento y siempre me recibió con una sonrisa y un 'que paso ahora Medu' y Duriet que se convirtió en poco tiempo en esa amiga en la que puedes confiar y sin decirlo sabes que puedes contar con ella para lo que sea. Gracias

A mis amigos, esos que hicieron mi paso por la universidad la mejor etapa de mi vida siempre estarán conmigo, los principios de Jogito, la sinceridad de Yosvany, la alegría de Gustavo, las ocurrencias de Miguel, las trastadas de Oscar, las burlas del otro Gustavo, los consejos de Maikel, los sentimientos de Jonathan, las inteligencia del flaco, las cosas de Barbarito, la risa de Janderd, la protestas de Keiger, los disparates de Almirola, las fiestas Mario, las clases de Kevin, los repasos de Frank un día antes, las discusiones de Gabriel el futbol de Yalbert y Oscar, la inocencia de Lamis y el vicio de juego de Fernando, Hanssel y El chino.

Agradecer a una persona que estos últimos años ha sido un apoyo esencial para mí, una de las mejores personas que he conocido en mi vida, una persona que fue capaz de encontrar agua en un desierto a base de perseverancia y le agradezco de corazón por eso, Gracias ANA.

Agradecerle a Amy la mama de mi hija, primero porque me dio lo más importante en mi vida y segundo porque me ayudó muchísimo en una de las etapas más difíciles para mí en la universidad.

Agradecerles a todos los profesores que han contribuido a mi formación personal y profesional, Osmany, Noel, Leosvany, Olga, la decana Mayra, Sailyn, Andrea, Yadira, el decano Rigo, mis entrenadores Gregorio, Omar y William.

A mis tutores Albertico y Andy que de no ser por ellos este no trabajo no hubiera salido adelante y por sacarme siempre el máximo y tener muchísima paciencia.

Por ultimo agradecerle a una persona que aunque ella no los sepa con solo pensarla me saca una sonrisa y mejora mi día, la persona que más quiero en este mundo, mi niña, mi chumi, Emily.

Datos de contacto

Autor:

José Luis Medero Bermúdez

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: jlmedero@estudiantes.uci.cu

Tutor:

Alberto Arístides Puentes

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: aapuentes@uci.cu

Tutor:

Andy Hernández Paez

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: andyhp@uci.cu

Resumen

En el presente trabajo se describe el proceso de desarrollo de un simulador para el protocolo de comunicación industrial Bsap Serial. En el módulo de adquisición del Centro de Informática Industrial (CEDIN) se desarrollan manejadores (*drivers,* en inglés) que implementan protocolos de comunicación industrial, actualmente el centro no cuenta con Controladores Lógicos Programables (PLC, por sus siglas en inglés) o simuladores que implementen dichos protocolos para establecer comunicación con estos manejadores. Debido a esta problemática fue necesario desarrollar un simulador que permita validar el correcto funcionamiento del manejador para el protocolo Bsap Serial del sistema para la Supervisión, Control y Adquisición de Datos (SCADA, por sus siglas en inglés) SAINUX 2.0. Con este objetivo se realizó un estudio del estado del arte de herramientas para la simulación de dispositivos industriales y los principales conceptos utilizados en el proceso de adquisición y envío de datos a través del protocolo de comunicación industrial Bsap Serial. Se representó la arquitectura definida para la aplicación, los requisitos funcionales y no funcionales que rigieron el proceso de desarrollo de la misma, concluyendo con un proceso de pruebas que validó su funcionamiento. Como resultado se obtuvo un simulador para la comunicación mediante el protocolo Bsap Serial, que permite generar y visualizar datos en diferentes formatos y activar situaciones excepcionales que pueden ocurrir en las comunicaciones.

Palabras clave: simulador**,** manejador**,** protocolo Bsap Serial**,** Controladores Lógicos Programables, SCADA

Índice

Índice de Ilustraciones

Índice de Tablas

Introducción

La industrialización es el proceso en el que una sociedad o país se transforma de una comunidad principalmente agrícola a un entorno de fabricación de bienes y servicios. El trabajo manual a menudo se sustituye por una producción masiva mecanizada y los artesanos se sustituyen por las líneas de montaje. Características de la industrialización incluyen el uso de la innovación tecnológica para resolver problemas como la dependencia de las condiciones fuera del control humano, gracias al avance de las tecnologías los procesos en las industrias tienen un seguimiento más exhaustivo.

Generalmente las plantas industriales necesitan un sistema de supervisión y control para asegurar una operación segura y económica de los procesos que se llevan a cabo. El sistema debe ser capaz de traducir los comandos del operador en las acciones necesarias, así como mostrar el estado de los procesos que se realizan en la planta. A estos sistemas se les conoce como SCADA. “Un sistema SCADA es un software diseñado para el control de la producción, permite obtener y procesar información de procesos industriales dispersos o en lugares remotos inaccesibles, transmitiéndola a un lugar para supervisión, control y procesamiento, normalmente una sala o centro de control” [10].

La adquisición y control de información en la industria son funcionalidades básicas de cualquier sistema SCADA, estas funcionalidades permiten recolectar, procesar y almacenar los datos provenientes del campo. Esta tarea es llevada a cabo mediante los manejadores de dispositivos.

Un manejador de dispositivo (en inglés, *driver*), no es más que un programa informático que permite al sistema operativo interactuar con los periféricos, haciendo una abstracción del hardware y permitiendo, mediante una interfaz bien definida, el acceso a los mismos. Los manejadores de dispositivos constituyen piezas esenciales de los sistemas computarizados, pues sin ellos no se podría utilizar el hardware. Su función principal en los sistemas de supervisión de industrias es establecer una comunicación entre el SCADA y los diferentes dispositivos del campo. Durante esta comunicación se obtiene información estadística del estado de los dispositivos y canales asociados a la red de campo. La comunicación se rige por un lenguaje denominado protocolo de comunicación [11].

Un protocolo de comunicación es un conjunto de reglas y procedimientos que proporcionan una técnica uniforme para gestionar un intercambio de información. En la adquisición de datos para los sistemas de supervisión el protocolo define la semántica y estructura de los mensajes que envía el manejador a los dispositivos. Estos mensajes comúnmente contienen solicitudes para las variables configuradas en el SCADA, que representan el comportamiento de los procesos industriales.

Cada variable configurada en el sistema está asociada a un dispositivo y tiene un periodo de muestreo determinado, que no es más que el tiempo establecido para que se actualice su valor de forma periódica. Este tiempo es configurado en dependencia del nivel de criticidad del proceso en cuestión y es de vital importancia para el SCADA que las variables se actualicen en el tiempo establecido. De este tiempo depende la posibilidad para los operadores de monitorear los procesos que se llevan a cabo y poder tomar las medidas necesarias ante cualquier situación extraordinaria. Es de vital importancia conocer detalladamente el funcionamiento de todo manejador y no hay mejor manera que mediante la realización de pruebas al mismo, así se pueden observar las respuestas en diferentes situaciones, para ello se puede hacer uso de dispositivos de campo o de un simulador que pueda lograr reproducir un comportamiento similar al real.

Un simulador es una máquina o un programa informático que reproduce el comportamiento de un sistema en ciertas condiciones, con el objetivo de recrear experiencias lo más reales posibles para que así se puedan detectar fallas o errores en el funcionamiento del sistema.

La Universidad de las Ciencias Informáticas desempeña una función importante en la informatización de la sociedad cubana. La misma se compone por varias facultades, entre ellas la facultad 4, en la cual radica el Centro de Informática Industrial (CEDIN). Este centro tiene entre sus objetivos principales el desarrollo de aplicaciones o servicios para la automatización de procesos industriales, como los sistemas SCADA. El proyecto SCADA SAINUX 2.0 del CEDIN cuenta con varios módulos como Adquisición e Interfaz Hombre Máquina (HMI, por sus siglas en inglés).

El módulo de Adquisición del SCADA SAINUX 2.0 es el encargado del procesamiento y recolección de la información proveniente de campo a través de los manejadores. Los manejadores son los encargados de implementar los protocolos específicos que se requieren para establecer comunicación con los dispositivos. El recolector es el encargado de configurar los manejadores y de asignarles las tareas de lectura y escritura de datos que se necesiten realizar. En el procesamiento se recibe la respuesta de estas tareas y se procesa dicha información para luego ser almacenada o visualizada.

En el CEDIN los procesos de desarrollo y pruebas del manejador para los dispositivos que se comunican a través del protocolo industrial Bsap Serial se han visto afectados, por lo que en el centro se llevan a cabo varias estrategias. La primera variante es adquirir el dispositivo de campo, para así realizar las pruebas de comunicación. En la mayoría de los casos no es posible principalmente por el precio del equipo en el mercado internacional. La segunda variante es adquirir un software que simule el dispositivo. Esta adquisición puede ser por compra o porque se encuentre libre en internet. La primera posee el mismo

inconveniente que la adquisición de dispositivo, aunque en algunos casos el costo es menor. La otra, que es la más utilizada en el centro, en muchas ocasiones no existe un software en internet que realice las funciones que se necesitan para probar el manejador desarrollado. En el módulo de Adquisición se finalizó la implementación de este manejador. Al iniciar la fase de pruebas para detectar errores el equipo especializado en esta área no cuenta con los dispositivos de campo ni software que permita probar el correcto funcionamiento del manejador.

Dada la **situación problemática** antes planteada, surge el siguiente **problema de investigación:**

¿Cómo contribuir al proceso de detección de errores del manejador Bsap Serial, desarrollado en el proyecto SAINUX2.0 del Centro de Informática Industrial?

Para darle solución a dicho problema se abordó como **objeto de estudio:** Simuladores de protocolo de comunicación industrial.

Como **objetivo general:** Desarrollar una aplicación que implemente el protocolo de comunicación industrial Bsap Serial para la detección de errores en la fase de pruebas del SCADA SAINUX 2.0.

Como **campo de acción:** Simuladores para el protocolo de comunicación industrial Bsap Serial.

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

1. Elaboración del marco teórico de la investigación a partir del estado del arte existente actualmente sobre el tema.
2. Estudio de las especificaciones del protocolo Bsap Serial.
3. Selección de la metodología y tecnología de desarrollo acorde con las políticas de la línea de Adquisición.
4. Diseño de la arquitectura del simulador Bsap Serial.
5. Diseño e implementación del modelo de clases de la aplicación.
6. Realización de las pruebas del simulador Bsap Serial.

Como **métodos de investigación científica** se emplearon los siguientes:

Métodos teóricos:

Análisis-histórico lógico: Se utiliza para realizar un análisis del estado del arte relacionado con el desarrollo de herramientas para la simulación de protocolos de comunicación industrial, sus principales, ventajas, desventajas, elementos que las componen y estructura de la información de las mismas.

Analítico-sintético: Se emplea para analizar la información encontrada sobre las herramientas para la simulación de protocolos de comunicación industrial y las características específicas del protocolo de comunicación Bsap Serial.

Modelación: Se utiliza para crear la representación de la herramienta a desarrollar y ayudar a la comprensión de los aspectos fundamentales que debe contener la misma para que cumpla con los objetivos de esta investigación.

Métodos empíricos:

Entrevista: Se utiliza principalmente con el objetivo de obtener información sobre los simuladores de dispositivos industriales desarrollados en el área sobre protocolos de comunicación industrial.

Observación: Se utiliza con el objetivo de describir y explicar el comportamiento de la simulación de dispositivos mediante protocolos de comunicación industrial. También se utiliza para identificar las herramientas a investigar y comprobar si cumplen con los objetivos necesarios para obtener el resultado esperado una vez realizada la evaluación.

Estructura de la investigación:

La investigación está estructurada en tres capítulos que abordan lo siguiente:

Capítulo 1. Fundamentación teórica: En este capítulo se realiza un estudio sobre el protocolo de comunicación industrial Bsap Serial y las herramientas que emplean el mismo. También se realiza un estudio de homólogos sobre herramientas que implementan protocolos de comunicación industrial para identificar características funcionales comunes. Además, se describen las herramientas y tecnologías a disponer para el desarrollo de la propuesta de solución.

Capítulo 2. Características y diseño del sistema: Se describen los requisitos funcionales de la propuesta de solución encapsulados en historias de usuario de acuerdo al escenario No. 4 propuesto por la metodología AUP-UCI. Se especifican los requisitos no funcionales, la arquitectura de software soportada, así como las clases del diseño de la propuesta.

Capítulo 3: Implementación y pruebas del sistema: En este capítulo se describe el estándar de codificación a seguir para la propuesta, los componentes que la conforman, la distribución física mediante el diagrama de despliegue, así como la ejecución de las pruebas de funcionalidad, aceptación y regresión.

Capítulo 1: Fundamentación teórica

Introducción

El presente capítulo tiene como objetivo brindar una breve descripción de los conceptos asociados al dominio del problema que serán de utilidad en la comprensión del mismo, se hará alusión a teorías relacionadas con el protocolo Bsap Serial tales como sus especificaciones técnicas. Además, se realizará un estudio del estado del arte, en el ámbito internacional y nacional, las tendencias, tecnologías y metodologías usadas para el desarrollo de simuladores como parte de la propuesta de solución.

1.1 Estado del arte

Para esta investigación se llevará a cabo un estudio exhaustivo de los principales simuladores informáticos utilizados, herramientas que implementan el protocolo Bsap Serial haciendo énfasis en simuladores que trabajen con protocolos de características similares al Bsap Serial y en qué sistemas son utilizados.

La simulación según David Himmelblau y Kenneth Bischoff en el libro “Análisis y simulación de procesos” es 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].

1.2 Simulación

El uso de la simulación es muy común en la actualidad debido a las ventajas que provee. Según el diccionario de la Real Academia Española (RAE), simular es representar algo, fingiendo o imitando lo que no es [4]. Enfocándose en el área de la ciencia, la simulación es una imitación o representación de las operaciones de un sistema mediante otro proceso más simple. Facilita analizar y estudiar las partes y las características del proceso real que interesan (descartando los que no son considerados necesarios o puedan complicar el desarrollo) sin la necesidad de ejecutarlo físicamente.

La simulación es una herramienta de análisis que permite sacar conclusiones sin necesidad de trabajar directamente con el sistema real que se está simulando. Es especialmente útil cuando no se dispone de dicho sistema o resulta demasiado arriesgado realizar experimentos sobre él [16]. Es el arte y ciencia de crear una representación o sistema para los propósitos de experimentación y evaluación. Permite predecir el comportamiento de los sistemas bajo diversas situaciones reales o previsibles (o lo que es lo mismo, situaciones simuladas) [16].

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

1.3 Simuladores de dispositivos

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

1.3.1 Simulador para el manejador DF1:

Simulador desarrollado en el Centro de Informática Industrial (CEDIN).

Tiene como objetivo evaluar el comportamiento que tienen los dispositivos PLC-5 cuando realizan transacciones de datos utilizando el protocolo de comunicación DF1.

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 respuestas implícitas) de la especificación ANSI x3.28. Existen dos categorías del protocolo DF1 [17]:

Protocolo Full-dúplex (Comunicación punto a punto).

Protocolo Half-dúplex (Comunicación Maestro/Esclavo).

Este protocolo tiene como principales características:

- 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 BCC2 o CRC3 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.

Este simulador no utiliza el protocolo Bsap Serial por lo que se hace imposible su uso en la solución del problema.

1.3.2 Simulador para el protocolo de comunicación industrial Ethernet/IP:

Simulador desarrollado en la Línea Adquisición del Centro de Informática Industrial (CEDIN).

Ethernet/IP es un protocolo de red en niveles para aplicaciones de automatización industrial. Es una red abierta que utiliza tecnologías como:

- El estándar de vínculo físico y de datos IEEE3 802.3.
- El conjunto de protocolos TCP/IP, estándar del sector para Ethernet.
- El Protocolo de Control e Información (CIP, por sus siglas en inglés), para la transmisión de mensajes de entrada/salida en tiempo real e información/transmisión de mensajes entre dispositivos similares.

Basado en los protocolos estándar TCP/IP, utiliza los ya bastante conocidos hardware y software Ethernet para establecer un nivel de protocolo para configurar, acceder y controlar dispositivos de automatización industrial. Ethernet/IP clasifica los nodos de acuerdo a los tipos de dispositivos preestablecidos, con sus actuaciones específicas. Brinda soporte al protocolo CIP utilizado en DeviceNet4 y ControlNet5. Apoyado en esos protocolos, Ethernet/IP ofrece un sistema integrado completo, enterizo, desde la planta industrial hasta la red central de la empresa [34].

La utilización del protocolo CIP le permite a Ethernet/IP organizar los mecanismos en red como una colección de objetos (o elementos) y define los accesos, atribuciones y extensiones con los cuales se puede acceder a una gama muy vasta de mecanismos mediante la utilización de un protocolo en común. Ethernet/IP está basado en un estándar ampliamente conocido y probado [34].

Este simulador no utiliza el protocolo Bsap Serial por lo que se hace imposible su uso en la solución del problema.

1.3.3 Simulador para el protocolo de comunicación industrial AB Ethernet:

Simulador desarrollado en la Línea Adquisición del Centro de Informática Industrial (CEDIN).

AB Ethernet es un protocolo de comunicaciones creado para la comunicación con los autómatas Allen Bradley de la firma Rockwell Automation. A través de este protocolo el SCADA se comunica con los autómatas PLC 5 y SLC. Este dialecto también permite acceder a los valores de las variables a través de nombres simbólicos o "*tags*" que cumplen determinadas reglas sintácticas y semánticas.

Entre las reglas sintácticas y semánticas que cumple este protocolo esta por ejemplo la especificación de cómo acceder a los elementos de un arreglo, por ejemplo: N10:125 y N10:126 son dos elementos del arreglo de enteros N10 [35].

Se puedan validar las direcciones asociadas a dispositivos AB Ethernet.

El formato de una dirección válida en el AB Ethernet está definida por el parámetro de configuración *remoteIp* que debe ser una dirección Ipv4 por ejemplo: 167.175.164.38.
El manejador AB dividir las variables a administrar o los grupos de ellas en bloques diferentes. Un bloque es la abstracción del conjunto de variables que tienen igual frecuencia de muestreo y que pueden ser recuperadas en una sola petición al PLC [35].

- Las variables que pertenecen a files distintos se ordenan en bloques diferentes tanto para lectura, como para escritura. Lo que permite que se limiten los errores pues si resulta dañado un mapa de memoria o file solo se afecta la calidad de la transmisión de datos en el bloque al que este pertenece.
- Los bloques de variables no pueden contener más de 218 bytes.

Este simulador no utiliza el protocolo Bsap Serial por lo que se hace imposible su uso en la solución del problema.

1.3.4 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 Bsap Serial, 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.

1.3.5 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 Bsap Serial, 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.

1.3.6 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 [18]. 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.

1.3.7 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 [19]. Este simulador no implementa la lógica de comunicación de protocolo Bsap Serial, por lo que no cuenta con los requisitos básicos para realizarle las pruebas al manejador Bsap Serial.

Estas herramientas son de gran apoyo a la comunicación con dispositivos industriales mediante un protocolo específico, no obstante, ninguna de ellas puede ser utilizada ya que no implementan el protocolo Bsap Serial.

1.4 Dispositivos Bristol

Los dispositivos Bristol que en su mayoría son PLC (Controlador Lógico Programable) cumplen con los requerimientos para cualquier aplicación de control industrial. Pueden ser instalados fácilmente en lugares remotos para aplicaciones de control. Normalmente presentan procesadores ARM (*Advanced RISC Machine*) de alta velocidad y su consumo de energía es bajo. Pueden trabajar en un amplio rango de temperatura (-40 a +70ºC) y utilizan como protocolo de comunicación BSAP *(Bristol Babcock Synchronous/Asynchronous Communication Protocol)* y Ethernet IP. Para la comunicación soporta las interfaces RS232, RS485 y Ethernet.

Existe gran variedad de dispositivos Bristol, entre ellos Bristol *ControlWave Express*, DPC 3330, DPC 3335, GFC 3308-x, RTU 3305, RTU 3310[5]

Todos los dispositivos de redes 3000 utilizan ACCOL, (*Advanced Communication and Control Oriented Language de Bristol Babcock*). ACCOL es un lenguaje multitarea que permite a un dispositivo Bristol realizar control continuo y discreto por lotes, así como medición, cálculos de flujo, gestión de alarmas y almacenamiento de datos.

ACCOL incluye un conjunto de más de 90 módulos pre-programados que se pueden combinar para formar una medición completa y estrategia de control para aplicaciones de SCADA y automatización [5].

1.5 Protocolo Bsap

El protocolo BSAP brinda soluciones para la comunicación con los dispositivos Bristol Babcock de redes 3000. Está orientado a la comunicación por encuestas y ofrece una seguridad alta para sus mensajes, ha sido diseñando de acuerdo al modelo OSI, donde cada capa tiene una función lógica propia y es operable bajo la modalidad síncrona y asíncrona.

BSAP es un protocolo propietario de red, el cual es aplicado para topologías de árbol donde cada nodo tiene una dirección única basada en su posición en la red y puede ser maestra de los niveles inferiores o esclava de los niveles superiores. Este protocolo puede soportar diversas configuraciones y más de un nodo por cada uno de los seis niveles que son permisibles en el árbol. Los mensajes pueden ser enviados entre

nodos del mismo nivel hasta nodos de otros niveles, cada mensaje contiene una identificación única y agrega un sistema de corrección de redundancia cíclica CRC [12].

1.5.1 Estructura jerárquica de red

El protocolo BSAP soporta una estructura de árbol donde el nodo raíz juega el papel del nodo maestro y está ubicado en el primer nivel del protocolo. Los nodos que se posicionan en el segundo nivel del árbol forman parte de una red de nodos esclavos del nodo maestro que está en el primer nivel. De esta manera se va formando la red, donde el nodo del nivel enésimo es el maestro de los nodos hijos del nivel enésimo +1 como se muestra en la Figura 1.

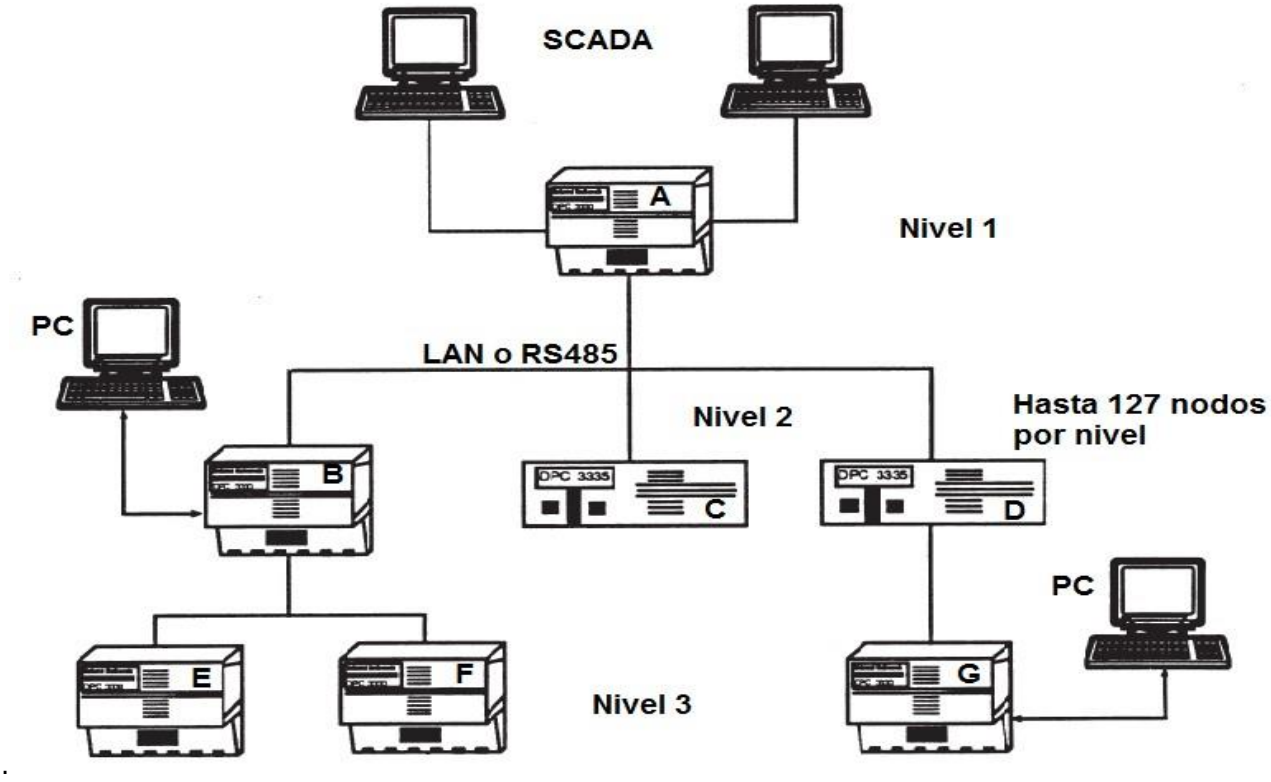


Ilustración 1: Estructura jerárquica del protocolo Bsap

El número de nodos esclavos que se pueden colocar en un nivel está condicionado por dos factores: El primer factor se debe a que la cantidad de nodos existentes deben permitir un tiempo de respuesta óptimo para los mensajes críticos.

El número de nodos permisibles se condiciona al direccionamiento físico de la dirección nodal (7 bits= 127 nodos). Por ser el protocolo BSAP un protocolo orientado a interrogaciones, el tiempo que tarde en llegar un mensaje de un nodo a otro, está condicionado con el tiempo de interrogación, por lo que se les da mayor prioridad a mensajes relacionados con las alarmas. Además de la limitación con el proceso de interrogación está la configuración del direccionamiento de los nodos. Cada nodo tiene una dirección única la cual es

basada en una secuencia dependiendo de su posición en la red como su posición en un nivel dado. Para cada nodo hay una secuencia en las posiciones que está dentro del rango de 1 a 127 y estas posiciones se definen como direcciones locales. Las direcciones locales no tienen razón para ser consecutivas, se puede configurar la red con ciertos huecos y luego colocar en ese espacio otro dispositivo sin necesidad de volver a cambiar el direccionamiento. Hay que tener en cuenta que el número más alto utilizado determina la cantidad de espacio requerido para definir la próxima dirección del nodo interno. Los niveles de las direcciones locales se concatenan para dejar una única dirección para cada nodo en la red. Esta dirección única es conocida como Dirección Global. El valor de esta dirección no puede exceder 32767 [5].

1.5.2 Relación de los nodos en la red

Cualquier nodo dentro de la red (excepto los extremos: nivel 0 y nivel último) tiene un doble papel: puede ser maestro de sus nodos inferiores o puede ser esclavo del nodo inmediatamente superior. Esta doble relación se define como una “relación local”, pues los nodos en cuestión son verticalmente adyacentes entre sí. Se denomina entonces “mensajes locales” al intercambio de información entre un nodo maestro y un esclavo o nodo sin pasar por ningún otro nodo; en este caso se aplica las direcciones locales. Los mensajes que pasan por uno o más nodos hasta alcanzar su destino, se denominan “mensajes globales” en donde se aplica las direcciones globales [6]. Por ejemplo, en la Figura. 1, un mensaje de B a E, o de B a F, son mensajes locales; mientras que mensajes desde A hasta G son mensajes globales.

1.5.3 Formatos generales de comunicación

El protocolo BSAP hace referencia a las variables como señales y estas pueden ser accedidas mediante su nombre o dirección MSD (*Master Signal Database*). El nombre para cada señal tiene el formato Base.Extensión.Atributo, en este caso hay que tener en cuenta que los tres campos pueden ser de longitud variable terminado en el carácter nulo. La base tiene como longitud máxima 9 caracteres ASCII, 7 caracteres ASCII máximo en caso de la extensión y 5 en caso del atributo. Sin embargo, las direcciones MSD se representan con un número entero asociado a la dirección física de la señal en el dispositivo, ocupando solamente 2 bytes en el buffer de transferencia. Cada dispositivo Bristol posee una base de datos donde se guarda la información para todas sus señales. Cada señal guarda su información en distintos campos, entre ellos su valor, su dirección MSD y el tipo de la señal que puede ser analógica o digital. Además, el protocolo brinda la posibilidad de especificar los campos deseados en cada petición. De esta manera se controla la cantidad de datos transmitidos por la red, ya que una vez conocida la dirección MSD para una señal no es necesario volver a solicitarla. Esto permite minimizar la cantidad de datos a transmitir y por tanto un mejor aprovechamiento de la comunicación.

1.5.4 Capas del protocolo

BSAP se ha diseñado e implementado de acuerdo con las normas del modelo OSI. Este modelo define claramente las interfaces entre cada capa permitiendo que diferentes sistemas operativos y protocolos de red puedan trabajar juntos.

El protocolo BSAP utiliza las cuatro capas inferiores del modelo OSI, estas se describen a continuación:

- Capa de transporte: es responsable de la correcta transmisión del mensaje a nivel funcional. Cuando la capa de transporte determina que está listo para transmitir, el control se pasa a la siguiente capa.
- Capa de red: Tiene la responsabilidad de determinar la ruta del mensaje a través de la red y las direcciones que debe utilizar.
- Capa de enlace de datos: Es responsable de mejorar el mensaje para incluir la comprobación de errores y mecanismos de corrección. También controla el acceso al canal físico sobre el cual se envía el mensaje.
- Capa física: Esta capa se compone principalmente del hardware y el software necesario para su control. Esta capa es totalmente independiente del formato final del mensaje que se transmite.

La ventaja de esta arquitectura es que, al aislar las funciones de comunicación de la red en capas, se minimiza el impacto de cambios tecnológicos en el protocolo, es decir, podemos añadir nuevas aplicaciones sin cambios en la red física y también podemos añadir nuevo hardware a la red sin tener que reescribir el software de aplicación.

1.5.5 Formato general de las tramas

Fundamentalmente, este protocolo tiene dos clases de tramas: las tramas de información y las tramas de supervisión y control.

Tramas de información: Las tramas o mensajes de información se dividen en Mensajes de Datos Globales y Mensajes de Datos Locales. El formato del mensaje global tiene la siguiente estructura.

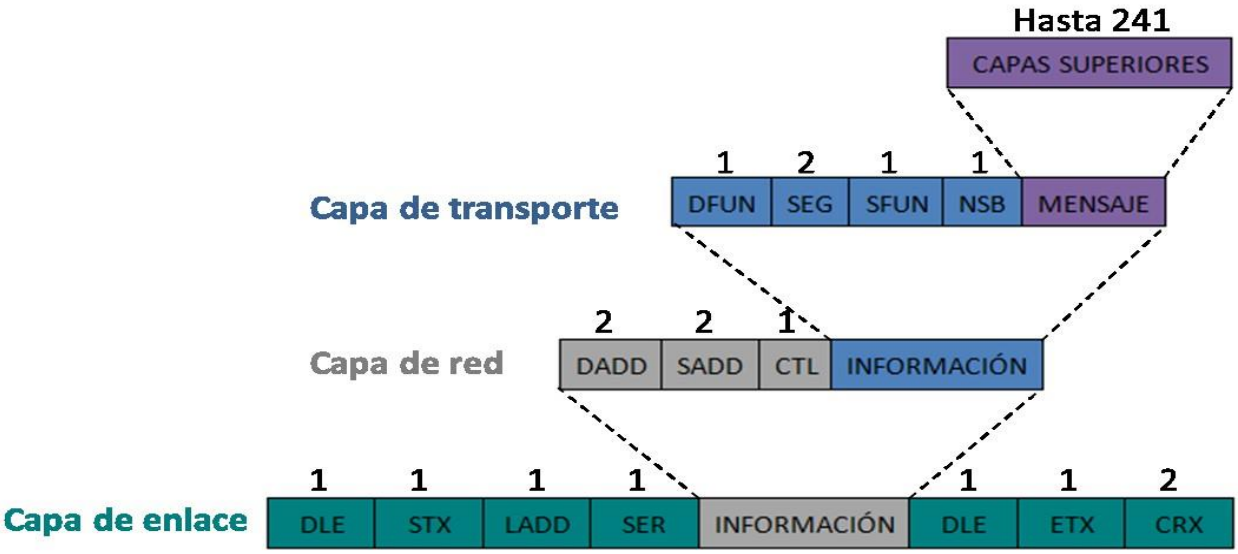


Ilustración 2 Formato de los mensajes globales

El mensaje global presenta varias cabeceras que son distribuidas por las capas del modelo OSI. Cuando se ensambla un paquete global la capa superior contiene el mensaje a transmitir y la capa del protocolo contiene la información para que el mensaje sea enrutado por la red dispositivos con éxito.

La capa de transporte contiene los siguientes campos:

Tabla 1 Descripción de códigos de la capa transporte

Código	Descripción
DFUN	Código de función del destino
SEQ	Número de secuencia del mensaje. Se requiere para evitar la duplicación accidental de mensajes en condiciones de ruido extremo.
SFUN	Código de función de la fuente. El número de funciones puede llegar hasta 20.
NSB	Octeto de Estado del Nodo. Si hay una falla en la comunicación, el dígito de mayor peso del octeto se pone a UNO y los otros 7 dígitos indican el tipo de falla.
MENSAJE	Depende de la aplicación y puede contener hasta 241 octetos.

La capa de Red aporta la cabecera siguiente:

Tabla 2 Descripción de códigos de la capa de red

Código	Descripción

DADD	Dirección de destino global.
SADD	Dirección de fuente global. Nótese que DADD y SADD son las direcciones de las maestras destino y origen, respectivamente.
CTL	Octeto de control. Contiene el tipo de mensaje (petición o respuesta) y el estado de la respuesta o número de nivel si hubo fallas.

A nivel de Capa Enlace:

Tabla 3 Descripción de códigos de la capa enlace

Código	Descripción
DLE	Caracter ASCII10H
STX	Caracter ASCII 02H
ETX	Caracter ASCII 03H
LADD	Dirección local. El dígito de mayor peso es una bandera que cuando está en UNO, indica que el mensaje contiene una estructura a nivel de red. LADD es la dirección del esclavo hacia dónde va dirigido el mensaje.
SER	Número de serie del mensaje. Este número es asignado por el maestro; el esclavo debe retornarlo en la respuesta. Se utiliza para evitar la confusión que se presenta cuando un mensaje o reconocimiento se pierde debido al ruido, etc. El número de serie cero no se utiliza porque está reservado para uso interno en el nodo.
CRC	Campo para la verificación de error. Se calcula desde STX a ETX y no toma en cuenta los DLE intermedios. Utiliza el algoritmo CRC UIT-T V.41.

El formato para los mensajes locales es el siguiente:

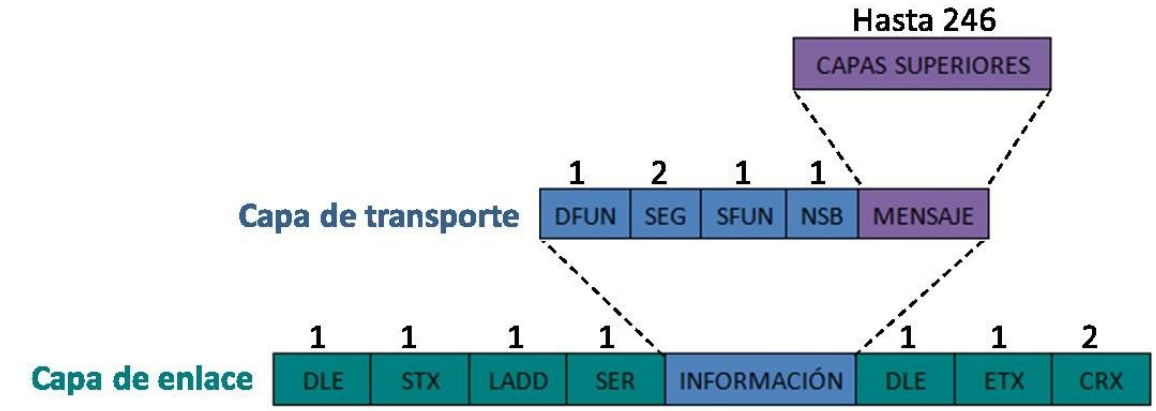


Ilustración 3 Formato de mensajes locales

La trama de datos para mensajes locales a diferencia de la trama global no contiene capa de red, para este caso el digito de mayor peso en la dirección local LADD es puesto en cero; los otros siete dígitos indican la dirección local de 1 a 127. La dirección del maestro por lo general es cero.

Mensajes de Supervisión y Control: Las tramas de supervisión y control más utilizadas son las siguientes:

1. Mensaje de Interrogación (Poll), Código 85H.
2. Mensaje de Reconocimiento (ACK o DOWN-ACK), Código 86H.
3. Mensaje de Reconocimiento/ “Nada para Transmitir” (ACK-NO DATA), Código 87H.
4. Mensaje de Reconocimiento Negativo (NAK), Código 95H.
5. Mensaje de Reconocimiento desde arriba (UP-ACK), Código 8BH.
6. Último Mensaje Descartado (DIS), Código 83H.

Mensaje de Interrogación (Poll), Código 85H

Este mensaje es utilizado por el nodo maestro para interrogar a sus nodos esclavos y determinar si están activos. De estar activos, se les solicitará información con el mensaje respectivo. El formato del mensaje es el siguiente:

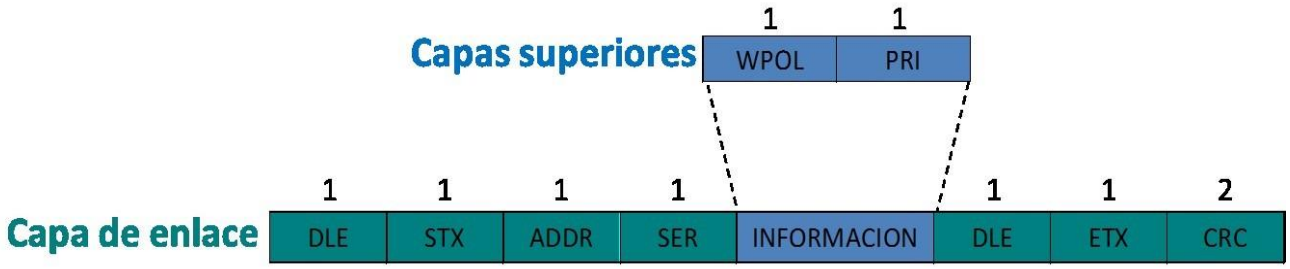


Ilustración 4 Formato de mensajes de interrogación

Tabla 4 Descripción de códigos de interrogación

Código	Descripción
ADDR	Dirección de la esclava o nodo interrogado.
SER	Número de serie del mensaje.
WPOLL	Código de Función 85H.
PRI	Prioridad de los datos requeridos. PRI = 00H indica que se puede aceptar alarmas o mensajes de datos. PRI = 10H indica que no se puede aceptar alarmas.

Reconocimiento (ACK o DOWN-ACK), Código 86H.

Este mensaje es una respuesta; la utiliza el nodo esclavo para reconocer a su maestro la recepción de un mensaje, excepto cuando se trata de una Interrogación (Poll). El formato correspondiente tiene la siguiente forma:

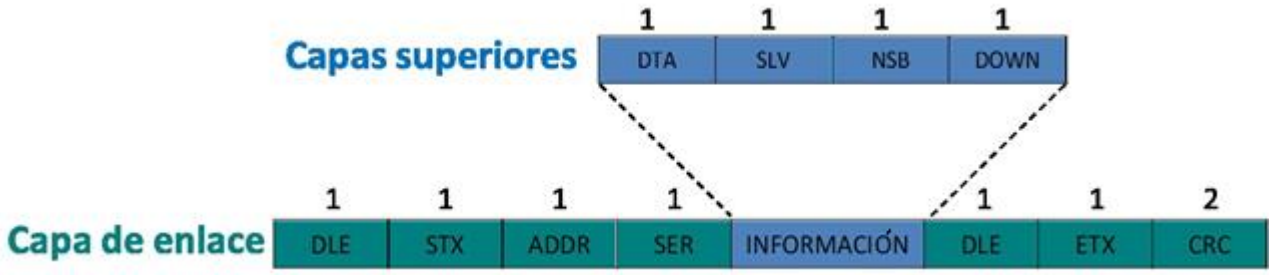


Ilustración 5 Formato de mensajes de reconocimiento

Tabla 5 Descripción de códigos de reconocimiento

Código	Descripción
ADDR	Dirección de la maestra (siempre a CERO).
SER	Número de serie del mensaje reconocido.
DTA	Código de Función 86H.
SLV	Dirección local del nodo esclavo o nodo que responde.
NSB	Octeto de Status del nodo esclavo. Con este octeto se le notifica al maestro ciertas condiciones existentes dentro del nodo esclavo.
DOWN	Número de buffers en uso.

Reconocimiento/Nada para Transmitir (ACK - NO DATA), Código 87H.

Este mensaje es utilizado por un nodo esclavo para reconocer la recepción de una Interrogación (Poll) indicando que no tiene mensajes de datos para transmitir.

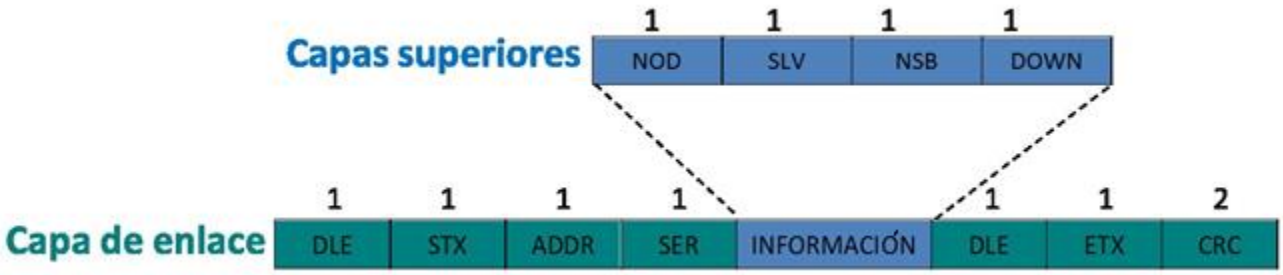


Ilustración 6 Formato de mensaje de reconocimiento/Nada para Transmitir

Tabla 6 Descripción de códigos reconocimiento/Nada para transmitir

Código	Descripción
NOD	Código de Función 87H.

Reconocimiento/No se dispone de espacio en el buffer (NACK), Código 87H.

Con este mensaje de respuesta, el nodo esclavo le indica a su maestro que además de una Interrogación (Poll) ha recibido también otro mensaje pero no dispone de espacio en los *buffers*.

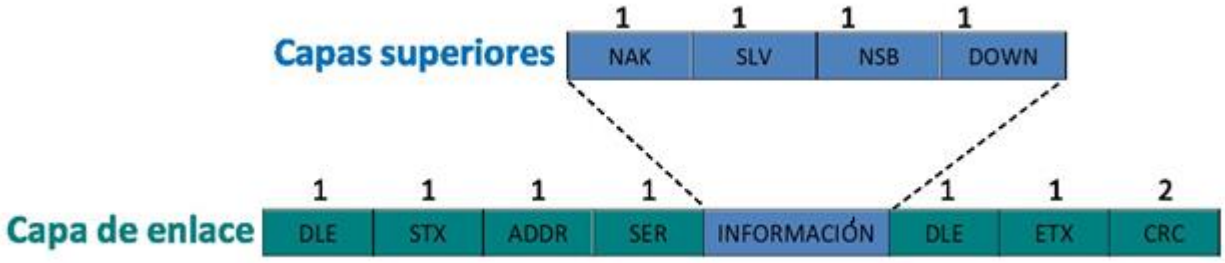


Ilustración 7 Formato de mensaje Reconocimiento/No se dispone de espacio en el buffer.

Tabla 7 Descripción de códigos de reconocimiento/No dispone de espacio en el buffer

Código	Descripción
SER	Número de serie del mensaje reconocido con NAK.
NAK	Código de Función 95H.

Reconocimiento desde arriba (UP- ACK), Código 8BH.

Este mensaje es utilizado por el nodo maestro para informar al esclavo que ha recibido correctamente y almacenado el mensaje.

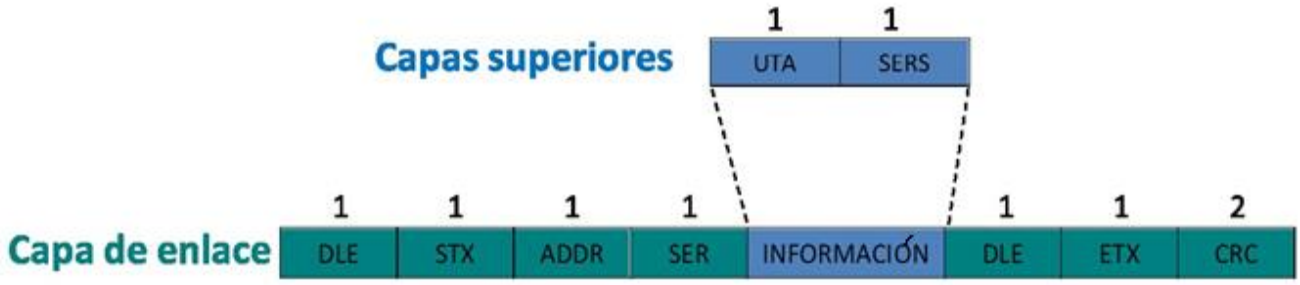


Ilustración 8 Formato de mensajes de reconocimiento desde arriba

Tabla 8 Descripción de códigos de reconocimiento desde arriba

Código	Descripción
ADDR	Dirección local de la esclava.
ERM	Número de serie del mensaje de la maestra.

UTA	Código de Función 8BH.
SERS	Número de serie del mensaje reconocido con UP-ACK.

1.5.6 Flujo de mensajes

El nodo maestro inicia el flujo de mensajes haciendo una solicitud al nodo esclavo. Si el direccionamiento es global, el mensaje debe pasar por varios nodos en el camino a su destino, la solicitud se dirige a su nodo intermedio adecuado.

El nodo intermedio responde con un mensaje 'DOWN TRANSMIT ACK' a su maestro con el mismo número de serie de la petición y como dirección local „0", que indica que es el maestro local. El mensaje con la petición entonces es enrutado hacia los próximos niveles con la nueva dirección local generada por el nodo intermedio [4].

Si la respuesta del esclavo no es inmediata, el maestro local debe interrogar a su esclavo para una respuesta. Si el esclavo responde con 'NO DATA TO TRANSMIT', esto significa que no tiene información disponible terminando la secuencia de interrogación. Cuando el esclavo responde a una interrogación el maestro envía a su esclavo un 'UP TRANSMIT ACK' indicando que recibió la trama satisfactoriamente y terminar el flujo de mensajes. O puede responder con 'UP TRANSMIT ACK' con una interrogación para solicitar más información [5].

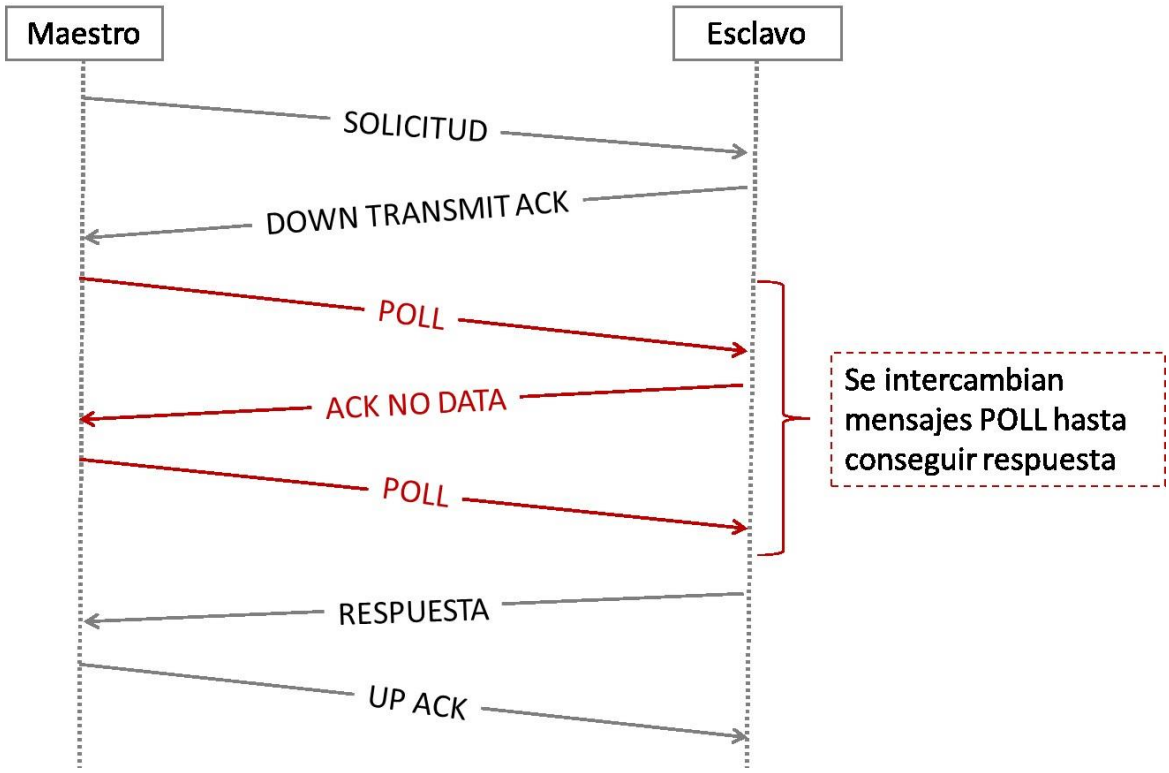


Ilustración 9 Secuencia de mensajes para una solicitud de datos exitosa

Después de completar el procesamiento en el nodo de destino global debe enviarse una respuesta al autor del mensaje. Para ello, las direcciones de los mensajes origen y destino se intercambian, haciendo que el mensaje de respuesta retorne a su remitente.

1.5.7 Tendencias y desarrollo del protocolo BSAP

Muchos SCADAs utilizan herramientas que proveen una interfaz de comunicación para los dispositivos Bristol, de estas herramientas las más conocidas son: *OpenBSI*, desarrollada por Bristol Babcock y *Universal Server* de *MTL Open System Technologies LP*. Sin embargo, otros SCADAs implementan el protocolo BSAP para comunicarse con este tipo de dispositivos, como es el caso de Oasys de la compañía *Valmet Automation.*

Open Bristol system interface (OpenBSI)

OpenBSI es un software de comunicación que proporciona acceso a la red de dispositivos Bristol Babcock. En la base cuenta con un servidor de comunicaciones para Windows 98/NT y 2000 a través del cual otras aplicaciones cliente se comunican con las redes Bristol. Provee un conjunto de herramientas que se comunican a través del servidor para recopilar y gestionar los datos recogidos en

la red, además de generar archivos basados en datos históricos, capturar las alarmas, supervisar y controlar las comunicaciones.

Entre las herramientas principales se encuentran:

- **LocalView:** Este programa permite establecer comunicación localmente con el controlador. A través de LocalView, otros programas de aplicación intercambian datos con él [7].
- **NetView:** Permite crear una arquitectura de control la cual da la posibilidad de monitorear la conexión en tiempo real de las variables, ver la programación en cada variable, modificar los parámetros de manera remota sin la necesidad de ir al sitio; es decir hay el control remoto y acceso a los datos y configuración sobre todas las variables [7].
- **DataView:** Permite recolectar diferentes tipos de información de los dispositivos Bristol. Señales, Listas, arreglos, archivos y registros de alarmas, así como búsqueda con criterios específicos pueden ser realizadas. DataView también permite cambiar los valores de las señales en línea y modificar las banderas de estado. Es un programa para trabajar en línea con los datos [7].
- **StandAlone OPC Server:** Proporciona una conectividad de OPC (OLE for Process Control) a la familia de controladores Bristol Babcock, de esta manera OpenBSI se puede comunicar con los distintos SCADAs. Se entiende por OPC como la conectividad abierta dentro la automatización industrial que garantiza la interoperabilidad mediante la creación y mantenimiento de especificaciones de estándares abiertos [7].

Universal server

Universal Server es un software desarrollado para sistemas basados en Windows, que brinda soporte para varios dispositivos utilizando protocolos como DF1, Modbus y BSAP. Además, cuenta con un servidor OPC para transmitir los datos recolectados a cualquier sistema de supervisión y control.

Universal Server provee un excelente soporte para dispositivos Bristol Babcock usando el protocolo BSAP. El módulo BSAP permite al usuario crear configuraciones para dispositivos que usan este protocolo y soporta los dispositivos 3305, 3310, 3330, 3335, 3340 y 3530 [15].

OASyS

OASyS es un SCADA, que incorpora aplicaciones interoperativas conectadas a través de interfaces estándares. TCP/IP sobre Ethernet y "Valmet´s SQL Backbone Middleware" proveen conexión directa a terminales de servidores, Sistemas Comerciales de Manejo relacional de Base de Datos (RDBMSs), y compuertas *(gateways*). [8]

OASyS es compatible con diversas plataformas de "software": Microsoft Windows NT y Windows 95/98, Digital UNIX, AIX, HP-UX, y Solaris. Estructuralmente OASyS está basado en tres elementos funcionales: Interfaz al Usuario, Manejador de Base de Datos, y Herramientas de Trabajo. Todos los elementos de OASyS están interconectados vía SQL Backbone; es decir en un ambiente cliente servidor para el transporte de datos y aplicaciones en tiempo real dentro de una arquitectura distribuida de OASyS y externamente incorpora a los otros subsistemas intranet [8].

Implementa gran variedad de protocolos entre ellos BSAP, este es utilizado para la comunicación con dispositivos de redes 3000. Para el transporte de los datos utilizan una biblioteca llamada libcomm que se integra con la arquitectura distribuida del SCADA garantizando la comunicación de manera eficiente.

1.6 Tecnologías y Herramientas

1.6.1 Framework Qt

Qt es un framework de desarrollo para aplicaciones multiplataforma que simplifica el desarrollo de aplicaciones en C++ de forma nativa [13]. 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 plugins 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.

1.6.2 IDE Qt Creator

El IDE utilizado para el desarrollo de la aplicación es Qt Creator en su versión 2.5.0. Qt Creator es un IDE multiplataforma que se ajusta a las necesidades de los desarrolladores. Este se centra en proporcionar características que ayudan a los nuevos usuarios del IDE a aprender y comenzar a desarrollar rápidamente, también aumenta la productividad de los desarrolladores con experiencia en Qt [13].

Qt Creator cuenta con:

- Un editor de código con soporte para C++.
- Herramientas para la rápida navegación por el código.
- Resaltado de sintaxis y auto-completado de código.

- Control estático de código y estilo a medida.
- Soporte para refactorización de código.
- Paréntesis coincidentes y modos de selección.

1.6.3 Lenguaje de Programación C++

C++ es un lenguaje imperativo orientado a objetos derivado del C. En realidad un súper conjunto 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 online, sobrecarga de operadores, referencias y operadores para manejo de memoria persistente [14].

1.6.4 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. Presenta licencia gratuita y comercial. Es fácil de instalar, actualizar y compatible entre ediciones.

1.6.5 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.

1.7 Metodologías de desarrollo de software

Se decide que AUP-UCI sea la que dirija el proceso de desarrollo puesto que:

- Esta metodología consiste centrar la atención en actividades que son esenciales para el desarrollo, no en todas las que forman parte del proyecto. Por lo que el equipo de trabajo no

va a leer detalladamente el proceso de documentación, ya que se sabe en lo que se está trabajando.

- Con el uso de esta metodología se puede utilizar cualquier conjunto de herramientas, pero lo aconsejable son las simples, por ser fáciles de manejar y entender o las de código abierto.
- Es una metodología de fácil adaptación, siempre satisfaciendo las necesidades propias de sus usuarios, por lo que no es necesario comprar una herramienta especial o tomar un curso para poder adaptar un proyecto utilizando AUP-UCI.

1.8 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 Bristol 3000, se definieron características específicas, como el formato de las tramas y la comunicación, que deberán estar presentes en el diseño de la solución propuesta, como son el protocolo de comunicación Bsap Serial. Se analizaron simuladores de dispositivos acordes a la investigación, determinando que no contaban con las características necesarias para realizarle pruebas al manejador Bsap Serial. Por otro lado, se definieron las herramientas y la metodología de desarrollo a ser utilizadas en la implementación del simulador.

Capítulo 2: Características y diseño del sistema

Introducción

En el presente capítulo se realiza una propuesta de solución del sistema a partir del análisis y la descripción del proceso real vinculado al campo de acción, así como el comportamiento que pueden presentar ambos actores en determinadas circunstancias. 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 mediante las herramientas seleccionadas para el desarrollo, además de la descripción de los requisitos funcionales por medio de las Historias de Usuario y los requisitos no funcionales, se definirán también los patrones de diseño a utilizar para resolver problemas en diversas circunstancias.

2.1 Descripción del proceso real vinculado al campo de acción

Con el objetivo de simular el proceso de comunicación del manejador Bsap Serial, se realizó un análisis para lograr un modelo con el cual se puedan representar las características y comportamientos fundamentales que deben tener estos dispositivos. En el proceso real, el manejador Bsap Serial deberá permitir la interacción del SCADA SAINUX2.0 con los dispositivos de campo, 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 Bsap Serial 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 Bsap en uno de sus modos de comunicación. 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 dispositivos, 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 Bsap Serial como por los dispositivos de campo deben tener la estructura requerida por el protocolo a utilizar.

A continuación, se define un mapa conceptual para dar solución al problema de la investigación.

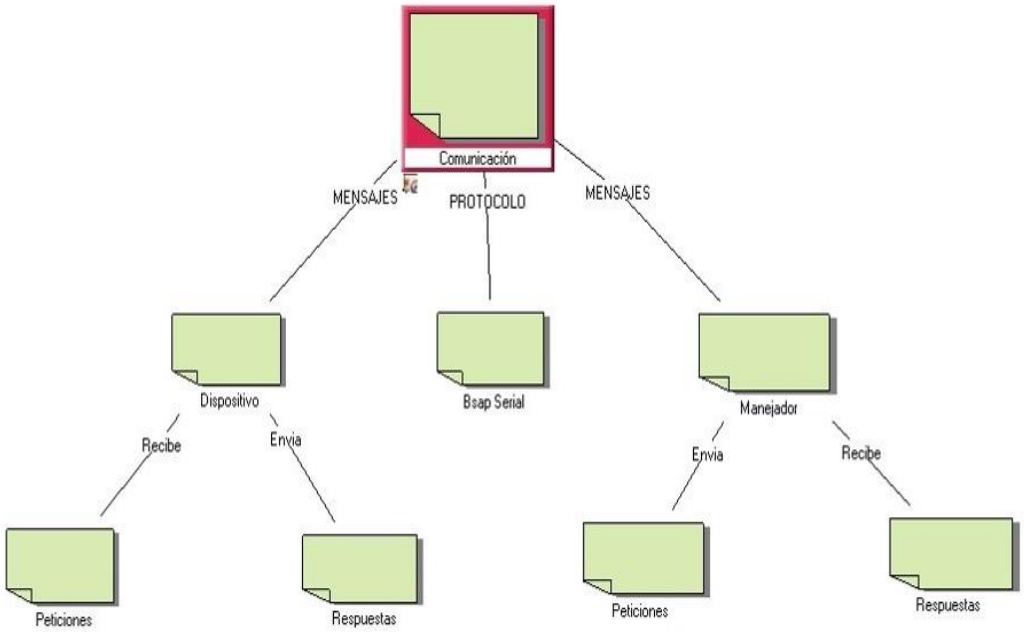


Ilustración 10 Descripción de la solución

El proceso comienza con un mensaje enviado desde el manejador hacia el dispositivo por medio del protocolo de comunicación Bsap Serial, el dispositivo recibe la petición enviada por el manejador y dependiendo de cómo esté estructurado el mensaje el dispositivo ensambla una determinada respuesta y la envía al manejador en forma de mensaje.

2.2 Requisitos del sistema

2.2.1 Requisitos Funcionales

Los requisitos funcionales fueron evaluados para determinar su Complejidad según el producto de trabajo Evaluación de requisitos propuesto por la metodología AUP-UCI. Los criterios que se evaluaron para determinar la complejidad técnica de cada requisito se relacionaban con: Interfaces, Influencia humana, Equipo, Programación, Comunicación, Diferentes comportamientos, Formas de inicialización, Fuentes de datos, Grado de reutilización, Restricciones de validación, entre otros. Mientras que para determinar la Prioridad de los requisitos se tuvieron en cuenta la influencia de los criterios de Importancia y Urgencia sobre los requisitos para el cliente, los cuales se proponen en el artefacto Especificación de requisitos de software propuesto también por la metodología adoptada. Tanto el resultado de la Complejidad técnica como el de la Prioridad de cada requisito se obtienen en: Alta, Media o Bajo. A continuación, se muestran en la Tabla 9 las descripciones de los requisitos funcionales junto a su Complejidad técnica y Prioridad para el cliente:

Tabla 9 Requisitos funcionales

ID	Nombre	Descripción	Complejidad	Prioridad
RF1	Simular dispositivo	El sistema debe permitir crear un dispositivo y comenzar la simulación.	Media	Alta
RF2	Detener simulación del dispositivo	El sistema debe permitir al usuario detener la simulación del dispositivo y eliminar el mismo.	Baja	Alta
RF3	Crear Variable	El sistema debe permitir al usuario crear variables para simular.	Alta	Alta
RF4	Modificar Variable	El sistema debe permitir al usuario modificar los parámetros de las variables.	Alta	Alta
RF5	Eliminar Variable	El sistema debe permitir al usuario eliminar la variable.	Baja	Alta
RF6	Cambiar clave del dispositivo	El sistema permite al usuario cambiar la clave del dispositivo.	Baja	Media
RF7	Simular fallas de comunicación	El sistema debe permitir al usuario simular fallas de comunicación.	Media	Media
RF8	Simular variables	El sistema debe permitir al usuario simular direcciones o variables	Media	Media

2.2.2 Requisitos no funcionales

RNF1 Apariencia o interfaz externa:

- Tendrá una interfaz intuitiva y amigable.

RNF2 Restricciones del diseño y la implementación:

- Para la implementación se debe utilizar el framework de desarrollo Qt.
- Para el diseño de los diagramas se debe utilizar el software Visual Paradigm en su versión 8.0 y para el modelado se debe utilizar el lenguaje de modelado UML.
- El lenguaje de programación debe ser C++.

RNF3 Software:

- El sistema operativo que se debe utilizar es GNU/Linux**.**

RNF4 Hardware:

- Será necesario como mínimo un ordenador con al menos 512 MB de RAM.

RNF5 Usabilidad:

- Los grupos de botones y vínculos deben estar organizados por funcionalidad, con el objetivo de facilitar al usuario la interacción con el sistema.

2.3 Historias de Usuario

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 10 HU del RF 1

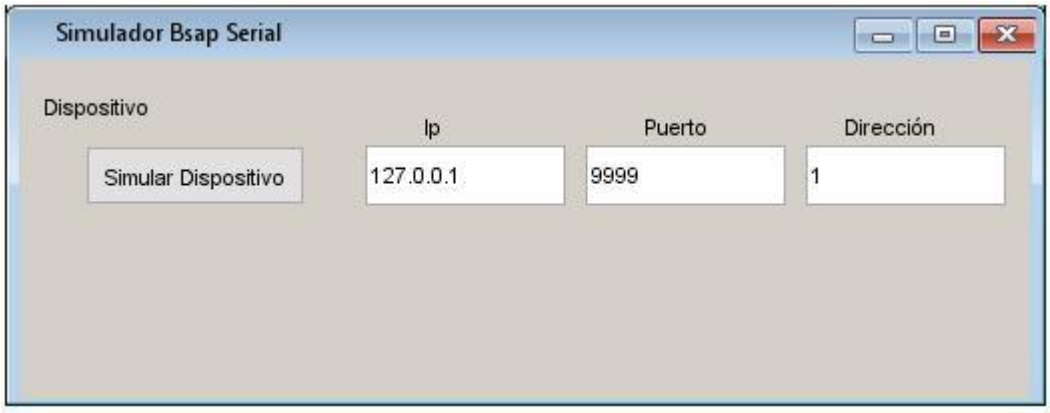
Número: HU 1	**Nombre del requisito:** Simular Dispositivo
Programador: José Luis Medero	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: • Planificación incorrecta.	**Tiempo Real:** 16 horas
Descripción: El sistema deberá permitir al usuario al iniciar la simulación del dispositivo.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 11 HU del RF 2

Número: HU 2	**Nombre del requisito:** Detener simulación
Programador: José Luis Medero	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: • Planificación incorrecta.	**Tiempo Real:** 16 horas
Descripción: El sistema deberá permitir al usuario detener la simulación del dispositivo y eliminar el mismo.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 12 HU del RF 3

Número: HU 3	**Nombre del requisito:** Crear Variable.
Programador: José Luis Medero	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: • Planificación incorrecta.	**Tiempo Real:** 16 horas
Descripción: El sistema deberá permitir al usuario crear variables brindándole la posibilidad de asignarle nombre, dirección y tipo.	

Observaciones:
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 13 HU del RF 4

Número: HU 4	**Nombre del requisito:** Modificar variables.
Programador: José Luis Medero	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: • Planificación incorrecta.	**Tiempo Real:** 24 horas
Descripción: El sistema deberá permitir al usuario modificar las variables, aplicando cambios en el valor y el tipo de dato en determinada variable.	
Observaciones:	

Prototipo elemental de interfaz gráfica de usuario:

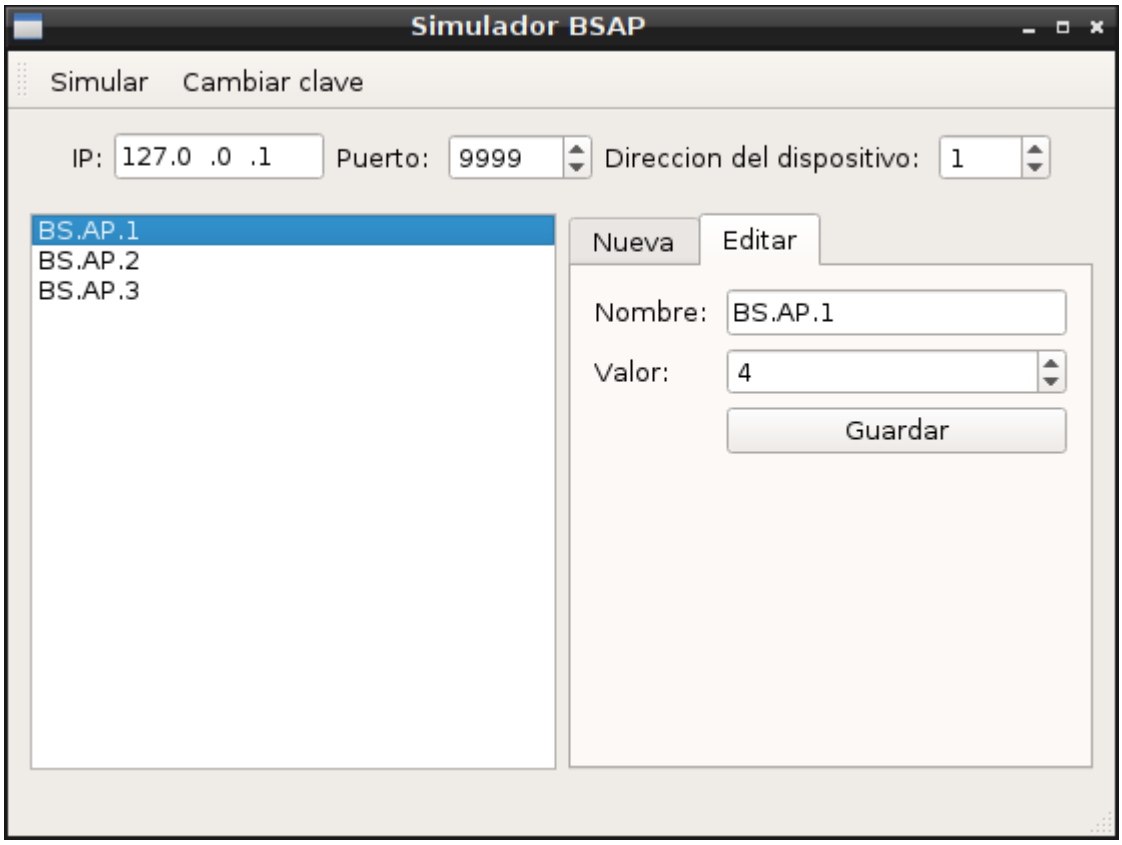


Tabla 14 HU del RF 5

Número: HU 5	**Nombre del requisito:** Eliminar variable.
Programador: José Luis Medero	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: • Planificación incorrecta.	**Tiempo Real:** 16 horas
Descripción: El sistema deberá permitir al usuario eliminar una determinada variable mostrándole siempre el mensaje ¨Seguro que desea eliminar la variable seleccionada¨.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

2.4 Distribución de esfuerzo estimado de Historias de Usuario (HU) por Requisitos Funcionales (RF)

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 15 Estimación de esfuerzo por HU

ID	Nombre	Iteración	Tiempo Estimado
HU1	Simular Dispositivo	1	8 horas
HU2	Detener Simulación	1	16 horas
HU3	Crear Variable	1	8 horas
HU4	Modificar Variable	2	16 horas
HU5	Eliminar Variable	2	8 horas
HU 6	Cambiar clave del dispositivo	3	8 horas
HU 7	Simular fallos de conexión	3	16 horas
HU 8	Simular variables	3	24 horas

2.5 Arquitectura del Sistema

2.5.2 Estilo Arquitectónico

Un estilo arquitectónico define un conjunto de principios que le dan forma o rigen el diseño del sistema a desarrollar. Estos principios van encaminados a cómo interactúan y están estructurados los componentes o módulos del sistema, así como las responsabilidades de cada uno. La arquitectura de un sistema de software casi nunca está atada a un solo estilo arquitectónico, sino que es más bien una combinación de estos estilos los que dan como resultado la arquitectura final del sistema.

Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema con el objetivo de establecer una estructura para los componentes del mismo. Con el fin de establecer una estructura para todos los componentes del sistema se escoge el estilo arquitectónico Orientado a objeto. El mismo plantea que los componentes de un sistema encapsulan los datos y las operaciones que deben aplicarse para manipular estos datos. La comunicación y coordinación entre los componentes se consigue mediante el paso de mensajes.

2.5.2 Patrón arquitectónico

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.

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar [22]. 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 el canal de comunicación, el cual implementa la lógica de comunicación, así como la ejecución de los comandos recibidos. Además, permiten la interacción del usuario con las propiedades de los dispositivos.

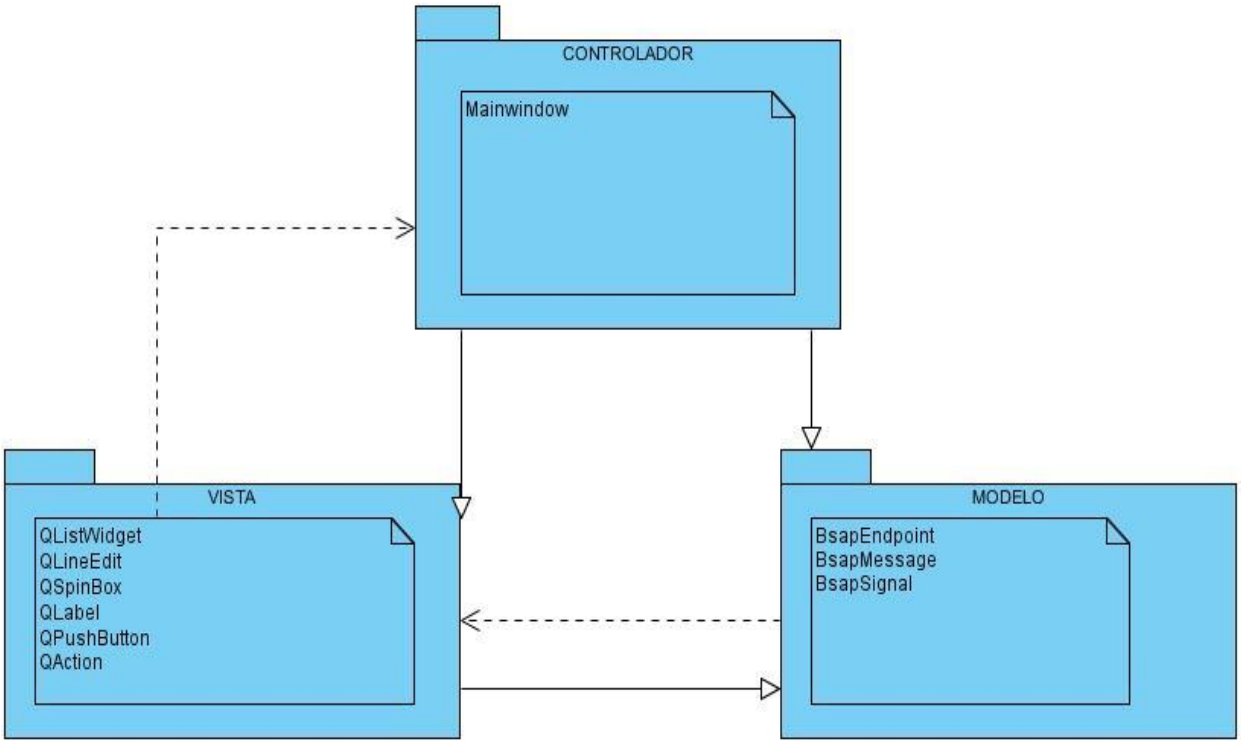


Ilustración 11 Patrón Arquitectónico

2.6 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces [23]. Son soluciones a diferentes clases de problemas conocidos que pueden ser aplicadas a diferentes códigos [24].

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.

2.6.1 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 [25]:

Tabla 16 Patrones GRASP

Nombre del patrón	Características	Clases

Patrón Experto	Asignan una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.	BsapMessage
Patrón Bajo Acoplamiento	Asignan las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible.	BsapEndpoint
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	BsapEndpoint
Patrón Controlador	Asignan la responsabilidad del manejo de mensajes de los eventos del sistema a una clase.	MainWindow

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 colabora con las otras para llevar a cabo las tareas.	BsapEndpoint

2.6.2 Patrones Grupo de los cuatro (GoF, Gang of Four)

Un patrón de diseño nomina, abstrae e identifica los aspectos claves de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizables. Este identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. Cada uno se centra en un problema concreto, describiendo cuando aplicarlo y si tiene sentido hacerlo teniendo en cuenta otras restricciones de diseño, así como, las consecuencias y las ventajas e inconvenientes de su uso. Por otro lado, como normalmente se tienen que implementar los diseños, un patrón proporciona código de ejemplo en C++ y a veces en Smalltalk10 para ilustrar una implementación [26].

El uso de los patrones de diseño en la implementación del sistema brinda la posibilidad de ahorrar gran cantidad de tiempo y recursos, además permite comprender, mantener y extender el software construido, así como, centrarse en desarrollar sólo las funcionalidades específicas requeridas por la aplicación.

2.6.3 Patrones Grupo de los cuatro (GoF, Gang of Four*)*

Patrón Iterator: Se utiliza cuando se necesita recorrer secuencialmente los objetos de un elemento agregado sin exponer su representación interna [27].

Se implementa este patrón en la clase BsapMessage para recorrer la lista de variables que se deben enviar y recorrer el mapa de memoria que contiene los datos de las variables.

Patrón State: Este patrón resulta útil cuando se necesita que un objeto se comporte de forma diferente dependiendo del estado interno en el que se encuentre en cada momento.

Se utiliza este patrón en la clase BsapEndpoint donde en dependencia del tipo de mensaje que se recibe se actualiza la propiedad ***state*** y en dependencia de su valor se determina el comportamiento en el momento de procesar el mensaje [28].

2.7 Diagrama de clases

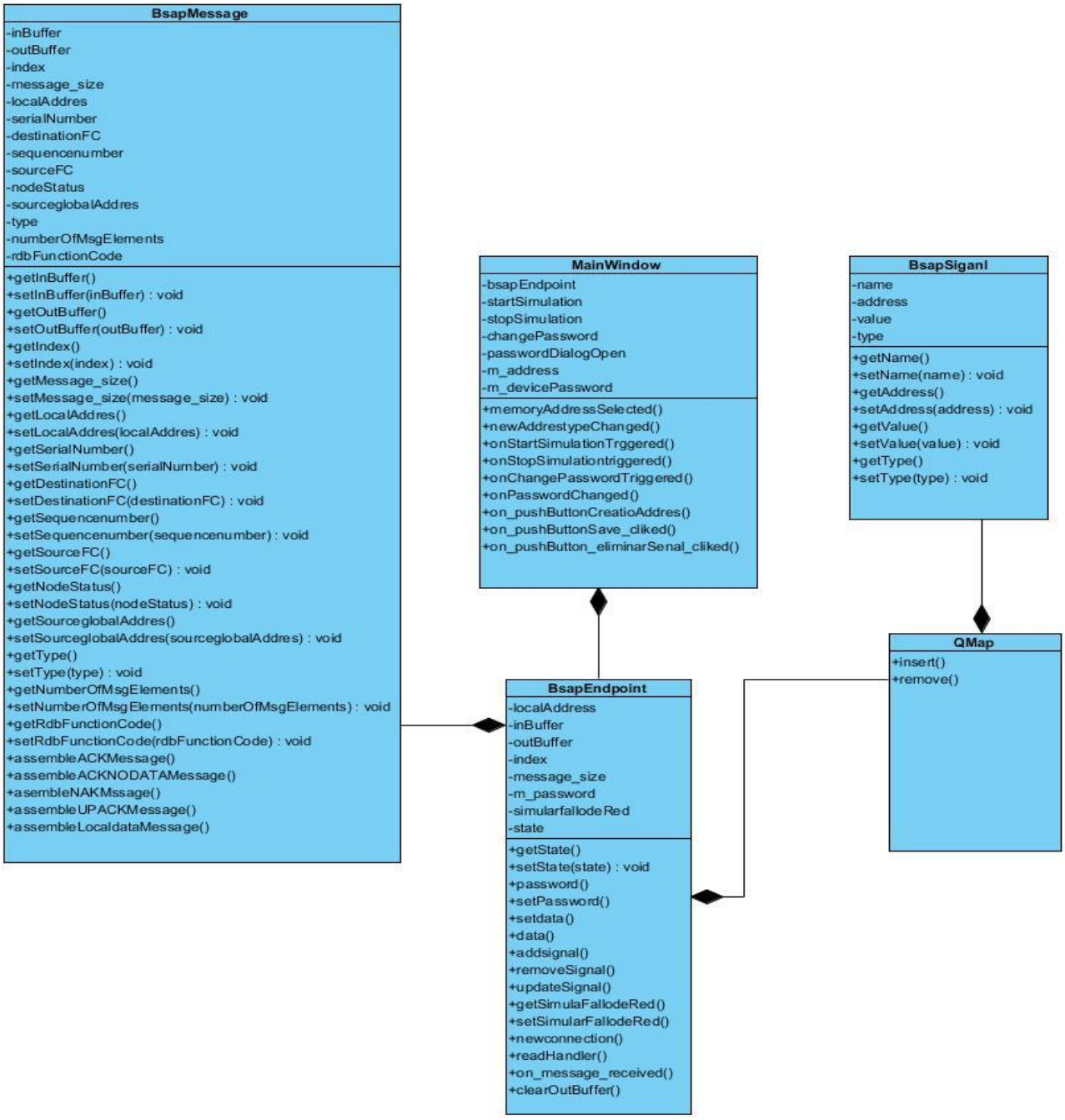


Ilustración 12 Diagrama de clases

2.8 Conclusiones parciales

Como resultado del desarrollo del presente capítulo se obtuvo el diseño de la aplicación a desarrollar. Donde se logró identificar los requerimientos funcionales a implementar, los cuales, según la metodología seleccionada se describen como HU. Así mismo se identificaron los requisitos no funcionales que permiten

que el sistema funcione de manera correcta. La arquitectura del software permitió estructurar y organizar los componentes del programa, además generar el diagrama de dominio del sistema, así como, los diagramas de clases con sus respectivas relaciones. Todo este proceso llevado a cabo durante el concluyente capítulo permitió obtener el modelado del negocio, listo para ser implementado.

Capítulo 3: Implementación y pruebas del sistema

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 AUP-UCI. 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 definiciones de clases, diagramas de componentes, los estilos de métodos y el diagrama de despliegue. Además, se muestran las pruebas realizadas al sistema, que permitieron la aceptación, por parte del cliente, de cada versión del sistema.

3.1 Estándar de programación

Los estándares de programación se enfocan en definir la estructura y apariencia física del código a programar, lo que facilita su entendimiento y lectura. En la implementación de la aplicación se utilizaron diferentes estándares que se describen a continuación:

3.1.1 Definición de clases

Las declaraciones de clases tienen su llave de apertura una línea más abajo de la declaración y el nombre de la clase comienza con mayúscula. A continuación, un ejemplo de la definición de una clase.

```
class BsapEndpoint : QObject
{
    Q_OBJECT
    Q_PROPERTY(State state READ state WRITE setState)
public:
    enum State {
        WaitingForTsNrt = 0,
        WaitingForFirstPoll,
        WaitingForAuth,
        AuthResponsePending,
        WaitingForPoll,
        WaitingForUpAckPoll,
        RdbSignalResponsePending
```

Ilustración 13 Declaración de clases

3.1.2 Definiciones de métodos

El nombre de los métodos comienza con una letra minúscula, en caso de ser un nombre compuesto por dos o más palabras, estas comenzarán con letra mayúscula. A continuación, un ejemplo:

Ilustración 14 Definición de métodos

3.1.2 Declaración de variables

Como norma general, nunca debemos usar variables globales. Las variables usadas como índices en iteraciones serán la i, j, k… típicamente darán comienzo en la i. Su nombrado deberá dar una idea del uso que se le dará a la variable.

Es recomendable usar un formato camelCase para el caso de variables de nombre compuesto. De esta forma, si tenemos una variable que almacenará el resultado de una suma, podría llamarse con formato camelCase como sumaDigitos por ejemplo.

Los valores constantes deberán declararse al comienzo del programa usando #define y su nombre deberá escribirse en mayúsculas. Toda variable debería iniciarse con algún valor en su declaración. Para el caso de variables tipo puntero, siempre deberán inicializarse a NULL.

3.1.3 Estructuras de control

El código deberá presentarse indentado dentro de cada estructura de control, como pueden ser los bloques *if, for, while*, y también para el contenido de funciones, registros, constructores, etc. De este modo logramos resaltar de forma visual una estructura lógica del código, lo cual simplifica enormemente su lectura.

Ejemplo:

```
if( localAddress == 0x01) //ACK recibido o mensaje local.
{
    destinationFC = readByte();

    if( destinationFC == ACK_DOWN || destinationFC == ACK_NO_DATA
        || destinationFC == ACK || destinationFC == NACK)
    {
        readByte(); //Direccion del esclavo.
        nodeStatus = readByte(); //
        readByte(); //cantidad de Buffers utilizados
```

Ilustración 15 Estructura de control 1

Siempre utilice llaves en todas las estructuras de control, aún incluso si sólo tienen 1 instrucción en su interior. Con esto se evitarán posibles errores si, posteriormente, deseamos añadir más instrucciones bajo la estructura de control afectada. La única excepción aceptable a esta regla es cuando ponemos la instrucción a ejecutar en la misma línea que la instrucción de control. Deberemos usar de forma consistente

un estilo de apertura y cierre de llaves. No se escribirá código justo después de una llave de apertura, sólo se insertará una instrucción por línea y deberá emplearse un espacio después de las comas.

Ejemplo:

```
for(chksm = ~0; index < size; ++ index)
{
    if( buffer[ index ] == DLE )
    {
        ++ index;
    }
```

Ilustración 16 Estructura de control 2

Usaremos paréntesis para especificar el orden de prioridades en operaciones aritméticas complejas, evitando que el programador deba recordar todas las reglas de precedencia de operadores. La única excepción aceptable es para los índices de los arreglos [30].

```
            chksm ^= buffer[ index ];
            for(unsigned char bit = 0x00; bit < 0x08; ++ bit)
            {
                chksm = (chksm & 0x0001) ? ((chksm >> 1) ^ 0x8408) : (chksm >> 1);
            }
        }
    }

    return chksm;
```

Ilustración 17 Uso de paréntesis

3.2 Diagrama de componentes

Un componente es una parte modular, desplegable y sustituible de un sistema, que encapsula implementación y expone un conjunto de interfaces [21]. Para representar la interacción entre los componentes de un producto de software se emplea el diagrama de componentes, el cual muestra cómo el sistema está dividido en componentes y las dependencias entre ellos. Además, permite tomar decisiones respecto a las tareas de programación y ayuda a los desarrolladores a visualizar el camino de la implementación [21]. A continuación se muestra el diagrama de componentes de la solución desarrollada.

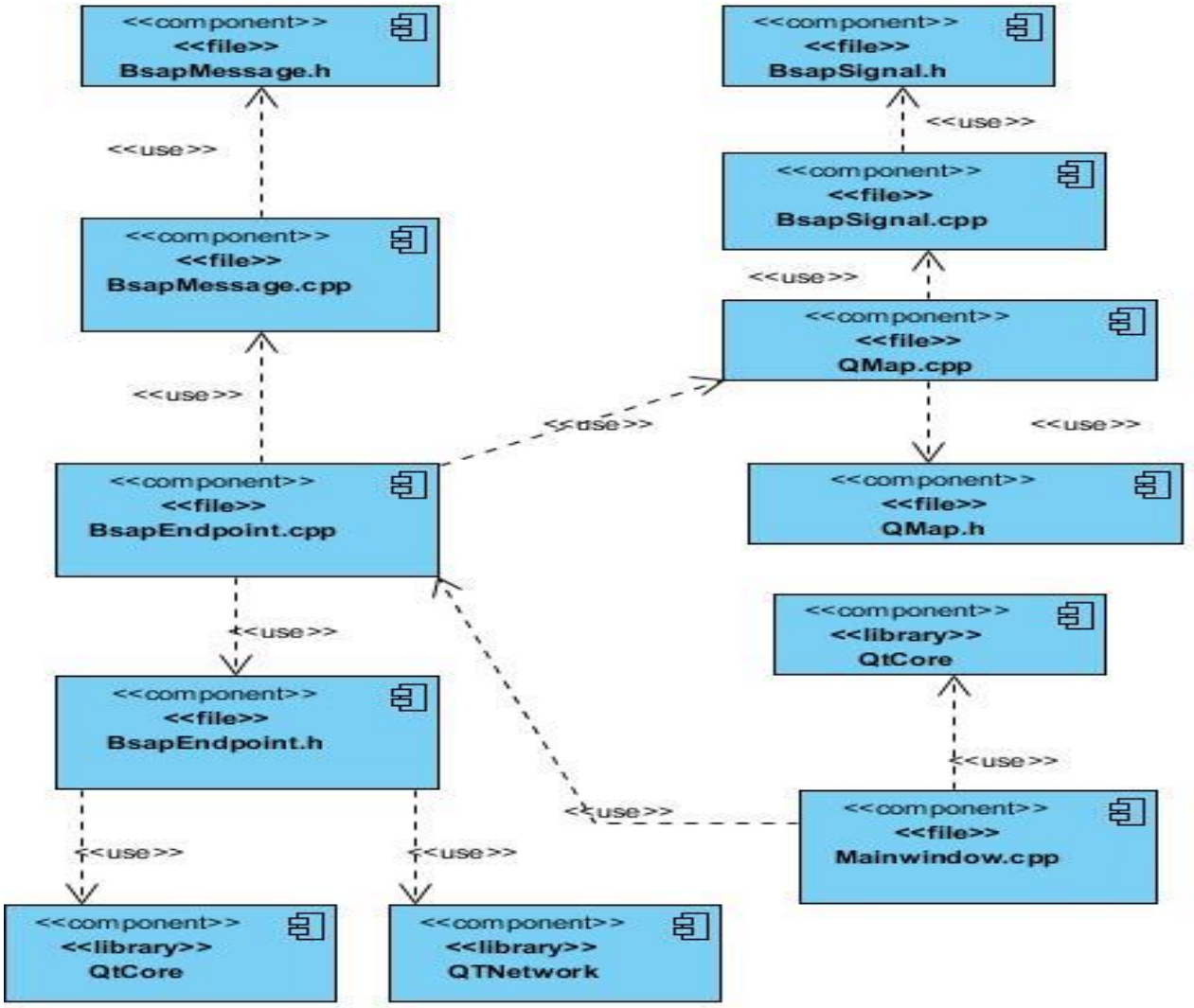


Ilustración 18 Diagrama de componentes

3.3 Diagrama de despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. Los estereotipos permiten precisar la naturaleza del equipo [31].

- Dispositivos.
- Procesadores.
- Memoria.

En la siguiente figura se muestra el diagrama de desligue del sistema:

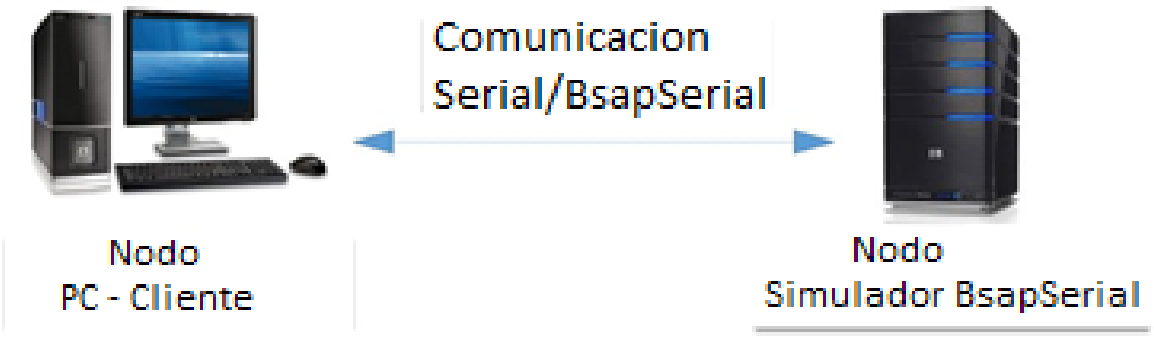


Ilustración 19 Diagrama de despliegue

El nodo PC-Cliente, representa el ordenador donde se ejecuta el manejador Bsap Serial. El nodo Simulador Bsap Serial representa el ordenador donde se ejecuta la aplicación desarrollada. La comunicación entre ambos nodos se realiza a través del protocolo de comunicación industrial Bsap Serial.

3.4 Pruebas

El único instrumento adecuado para determinar el estatus de la calidad de un producto de software es el proceso de pruebas. Estas se ejecutan dirigidas a componentes del producto o al sistema en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. Para ejecutarlas se utilizan los casos de prueba, especificados de forma estructurada mediante técnicas de prueba. Este proceso, sus objetivos y los métodos y técnicas usados se describen en el plan de prueba [32].

La metodología de desarrollo de software AUP-UCI desagrega la fase de pruebas en tres disciplinas: Pruebas Internas, de Liberación y Aceptación. Para verificar el resultado de la implementación y comprobar la calidad del producto se realizaron Pruebas Internas, utilizando como método de prueba, las Pruebas de Caja Negra y como tipo, las Pruebas Funcionales. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido y verificando que se cumplan los requisitos.

3.4.1 Diseño de Casos de Prueba (DCP)

Los Casos de Prueba (CP) son diseñados en dependencia de los distintos tipos de pruebas a utilizar. Cada CP va acompañado del resultado que ha de producir el software al ejecutarlo para detectar un posible fallo en el mismo. Se definen un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada método de prueba proporciona distintos criterios para generar estos casos o datos de prueba.

Para el desarrollo de estas pruebas existen dos métodos fundamentales: caja blanca y caja negra, las últimas se llevan a cabo sobre la interfaz del software. Su objetivo fundamental es demostrar que las funcionalidades son operativas, que las entradas se aceptan de forma adecuada y se produce el resultado correcto. La técnica aplicada fue la de partición de equivalencia para examinar los valores válidos e inválidos de las entradas existentes en la aplicación.

Para las pruebas realizadas a la propuesta se definen como posibles clasificaciones a las entradas a los campos como: válido (V), inválido (I) y no aplica (NA), teniendo en cuenta las variables definidas para el presente caso de prueba (CP) [33]:

Tabla 17 Diseño de Pruebas Crear Variable

Escenario	Descripción	Tipo	Nombre	Valor	Respuesta del Sistema	Flujo Central
EC 1.1 Entrada de datos válidos.	Se deberán introducir los campos correctos para crear una variable exitosa.	V	V	V	Guarda la variable con los parámetros.	Se selecciona en la pantalla principal la pestaña "Nueva" donde se llenan los campos tipo, nombre y valor y se presiona el botón crear.
		Entero	BS.AP.1	1		
EC 1.2 Ausencia de campos llenos de datos válidos.	Se deberá mantener el campo número o nombre de forma vacía.	V	I	V	Muestra en pantalla el siguiente mensaje "Existen Campos Vacíos".	Se selecciona en la pantalla principal la pestaña "Nueva"´ donde se llena el campo nombre, se presiona el botón crear.
		Entero	(Nulo)	1		
		V	V	I		
		Float	BS.AP.1	(Nulo)		
EC 1.3 Entrada de datos inválidos.	Se deberá introducir el campo número o nombre con caracteres no válidos.	V	V	I	Muestra el mensaje "Campo no válido".	Se selecciona en la pantalla principal en la pestaña Nueva los campos nombre y valor y se introducen los parámetros correctos.
		Entero	BS.AP.1	4,5		
		V	I	V		
		Float	BS*23.A	2,5		

Tabla 18 Diseño de Pruebas Modificar Variable

Escenario	Descripción	Tipo	Nombre	Valor	Respuesta del Sistema	Flujo Central
EC 1.1 Entrada de datos válidos.	Se deberán introducir los campos correctos para modificar una variable.	V	V	V	Modifica la variable con los nuevos parámetros.	Se selecciona en la pantalla principal la pestaña Editar y se introducen los datos en los campos indicados y se presiona el botón Editar.
		Bool	BS.AP.1	Verdadero		

EC 1.2 Ausencia de campos llenos de datos válidos.	Se deberá mantener el campo número o nombre de forma vacía.	V	V	I	Muestra en pantalla el siguiente mensaje “Existen Campos Vacíos”.	Se selecciona en la pantalla principal la pestaña Editar y se introducen los datos en los campos indicados y se presiona el botón Editar.
		Bool	BS.AP.1	(Nulo)		
		V	I	V		
		Float	(Nulo)	3,2		
EC 1.3 Entrada de datos inválidos.	Se deberá introducir el campo número o nombre con caracteres no válidos.	V	I	V	Muestra el mensaje “Campo no válido”.	Se selecciona en la pantalla principal la pestaña Editar y se modifican correctamente los datos en los campos indicados y se presiona el botón Editar.
		Entero	B34.**/AP	2		
		V	V	I		
		Float	BS.AP.1	-5		

Tabla 19 Diseño de Pruebas Simular Dispositivo

Escenario	Descripción	IP	Dirección del dispositivo	Puerto	Respuesta del Sistema	Flujo Central
EC 1.1 Entrada de datos válidos.	Se deberán introducir los campos correctos para simular el dispositivo.	V	V	V	Comienza la simulación del dispositivo.	Se selecciona en la pantalla principal los campos ip, dirección y puerto, se introducen los datos correctos se presiona el botón Simular.
		10.55.20.10	1	9999		
EC 1.2 Ausencia de campos llenos de datos válidos.	Se deberá mantener el campo número o nombre de forma vacía.	I	V	V	No permite dejar ningún campo vacío.	Se selecciona en la pantalla principal los campos ip, dirección y puerto, se introducen los datos correctos se presiona el botón Simular.
		(Nulo)	1	9999		
		V	I	V		
		10.55.20.10	(Nulo)	9999		
EC 1.3 Entrada de datos inválidos.	Se deberá introducir el campo número o	V	V	I	Comienza la simulación pero no	Se introducen los datos correctos en los campos ip, dirección y puerto y se
		10.55.20.10	1	R3r4		

	nombre con caracteres no válidos.	V 10.55.20.10	I 23 * ABC	V 9998	recibe peticiones del manejador.	presiona el botón Simular.

Tabla 20 Diseño de Pruebas Cambiar Clave

Escenario	Descripción	Clave	Respuesta del Sistema	Flujo Central
EC 1.1 Entrada de datos válidos.	Se introduce el campo de forma correcta.	V 66666	Guarda la clave introducida.	Se selecciona en la pantalla principal el campo clave y se introduce la nueva clave, luego se presiona el botón Aceptar.
EC 1.2 Ausencia de campos llenos de datos válidos.	Se deja el campo vacío.	I (Nulo)	Se muestra en pantalla el siguiente mensaje “Existen Campos Vacíos”.	Se selecciona en la pantalla principal el campo clave y se introduce la nueva clave, luego se presiona el botón Aceptar.
EC 1.3 Entrada de datos inválidos.	Se deberá introducir el campo número o nombre con caracteres no válidos.	I Bsapserial1	No permite introducir una clave con más de 6 caracteres.	Se selecciona en la pantalla principal el campo clave y se introduce la nueva clave, luego se presiona el botón Aceptar.

3.4.2 Pruebas de aceptación

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece el mismo. Sin embargo, antes de comenzar a usarlo es muy difícil determinar si sus ventajas realmente justifican su utilización. La mejor estrategia para esta determinación es la conocida prueba de aceptación.

Las pruebas de aceptación permiten determinar el grado de satisfacción de los usuarios finales, quienes deben informar de todas las deficiencias o errores que encuentren antes de dar por aprobado el sistema definitivamente. Para la preparación, ejecución y evaluación de estas pruebas no se requiere de conocimientos informáticos. Una prueba de aceptación puede ir desde el análisis de un informal caso de prueba hasta la ejecución sistemática de una serie bien planificada. Estas pruebas tienen vital importancia,

pues los resultados proyectados deciden la aceptación o el rechazo del producto y definen el paso a nuevas fases como el despliegue y mantenimiento.

Una vez concluida la implementación se planificó un encuentro con usuarios finales para determinar si realmente el producto cumple con sus expectativas. De un total de 6 usuarios que probaron la propuesta de solución implementada en un contexto diferente al entorno de desarrollo, 5 mostraron un nivel de satisfacción Alto y 1 Medio, de acuerdo a las respuestas de las preguntas realizadas como parte del análisis de satisfacción que se muestran a continuación:

- ¿Le resulta cómodo interactuar con la aplicación propuesta?
- ¿Considera útil la aplicación para las pruebas al manejador Bsap Serial?
- ¿Contribuye el resultado de la aplicación a facilitar el trabajo al realizar las pruebas al manejador Bsap Serial?
- ¿Considera que la información generada a partir de la simulación realizada es suficiente?

Estableciendo un análisis de los resultados, se considera, que es factible la aplicación para las pruebas al manejador Bsap Serial. A continuación, se muestran los resultados obtenidos de las pruebas de aceptación en la Ilustración 20:

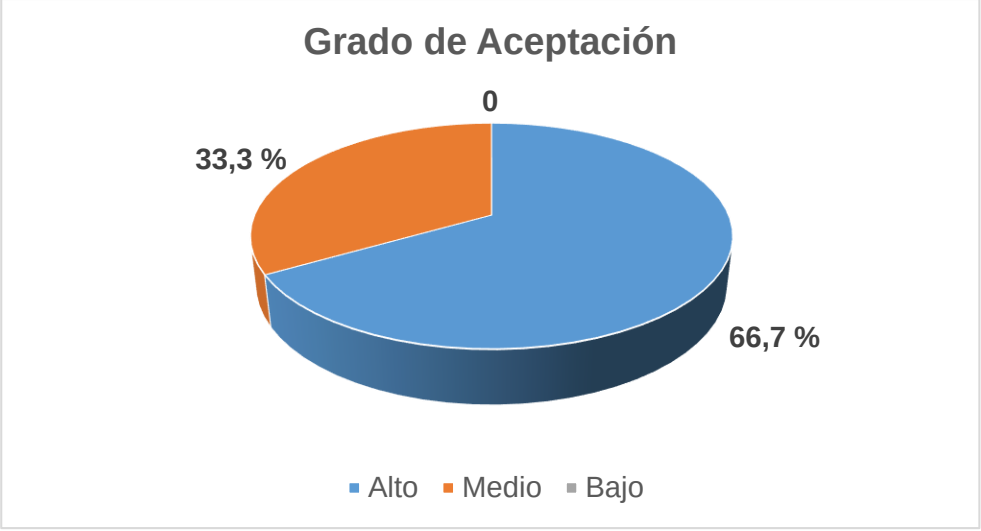


Ilustración 20 Resultados de las pruebas realizadas

3.5 Resultados de las pruebas realizadas

Durante la realización del tipo de prueba de Funcionalidad mediante el método de prueba de caja negra aplicando la técnica de partición de equivalencia, se detectaron no conformidades relacionadas con errores de validación y funcionalidad. Los resultados se pueden observar en la tabla 21, donde se muestran la cantidad de casos de prueba ejecutados y las no conformidades detectadas, las cuales fueron corregidas. Se realizaron tres iteraciones, durante la primera iteración se ejecutaron nueve casos de pruebas, de los cuales cuatro resultaron tener un total de cinco no conformidades. En la segunda iteración a través de las pruebas de regresión se verificó que las no conformidades anteriores estuviesen solucionadas sobre los casos de prueba que incidían; de estas pruebas se obtuvo una única no conformidad, quedando resuelta en la tercera iteración y cumpliéndose correctamente los requisitos funcionales.

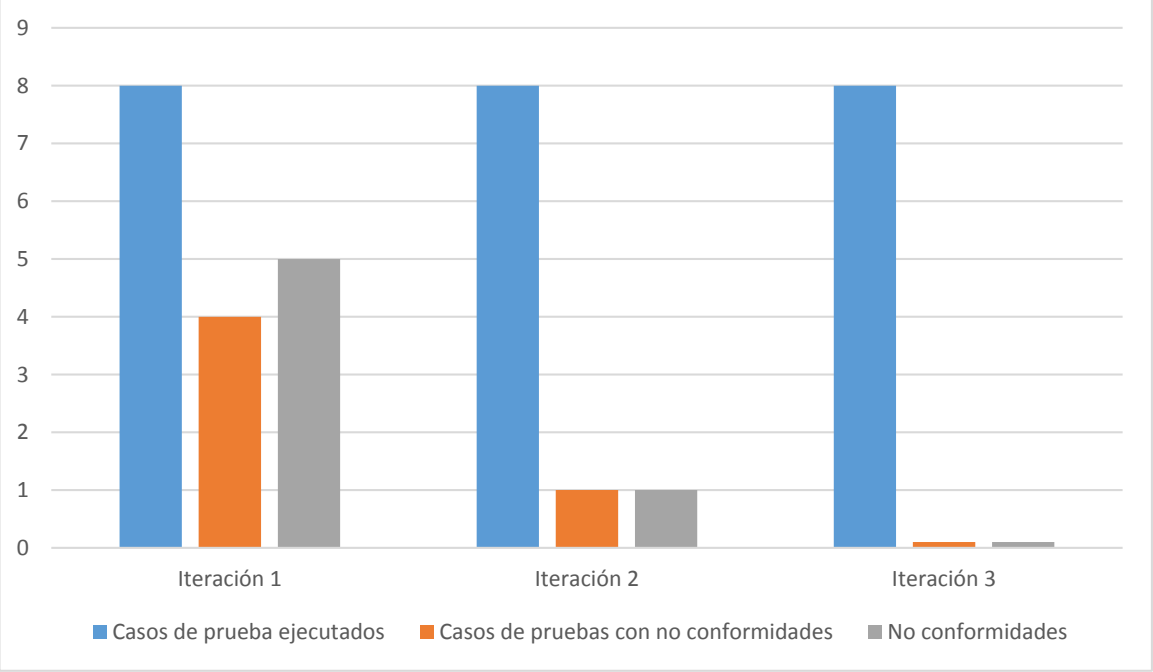


Ilustración 21 resultados de las pruebas realizadas

Conclusiones parciales

Luego de la implementación del sistema y la posterior ejecución de los casos de pruebas definidos, se determinó que la aplicación desarrollada cumple el objetivo general de la presente investigación, reuniendo los requisitos necesarios para establecer una correcta simulación del manejador Bsap Serial.

Conclusiones generales

Con el desarrollo de la presente investigación se obtuvo una herramienta que simula la comunicación de un dispositivo de campo con el manejador Bsap Serial, para el protocolo Bsap Serial, para lo cual:

- La identificación de las principales tendencias en el proceso de simulación de dispositivos industriales, a partir del estudio del estado del arte posibilitó descartar la existencia de alguna herramienta que brindara solución al problema identificado en la investigación.
- Las herramientas y tecnologías seleccionadas como framework Qt, entorno de desarrollo Qt-Creator y Visual Paradigm permitieron la implementación de la solución que simula el protocolo de comunicación industrial Bsap Serial.
- La selección de la metodología AUP-UCI condujo el proceso de desarrollo de la propuesta permitiendo describir sus funcionalidades mediante ocho historias de usuario en tres iteraciones, diseñar la estructura funcional mediante la arquitectura Modelo-Vista-Controlador, así como especificar las características operacionales encapsuladas mediante cinco clases del diseño y doce componentes.
- La implementación de la propuesta de solución permite generar y visualizar datos en diferentes formatos, así como, recrear situaciones excepcionales que pueden presentarse en las comunicaciones, haciendo posible la detección de errores en el comportamiento del manejador que se prueba.
- La ejecución de las pruebas de funcionalidad, regresión y aceptación para la evaluación de la propuesta permitió solucionar los defectos identificados en tres iteraciones de comprobación, a partir del análisis de ocho diseños de casos de pruebas relacionados con las historias de usuario descritas.

Recomendaciones

Para dar continuidad a la investigación se recomienda:

- Permitir el control desde los clientes a través de escrituras de valores en el simulador haciendo uso del protocolo Bsap Serial.
- Implementar los niveles al protocolo de forma tal que se puede simular una red de dispositivos Bsap.

Referencias Bibliográficas

[1] David Himmelblau y Kenneth Bischoff en el libro “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] Sánchez, M. Martínez, A. (2014). Informática biomédica. México. Editorial Elsevier, UNAM, Facultad de Medicina. 2da edición. Capítulo 13

[3] Gómez, Modelamiento y simulación de un horno túnel industrial. Escuela de procesos y energía. Universidad Nacional de Colombia. Medellín

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

[5] Babcock, B., Network 3000 Communications Application Programmer’s Reference. 2005, Watertown.

[6] Márquez, J.E.B., Transmisión de datos, Tercera edición. 2005

[7] Naranjo, V.H., DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL Y MONITOREO DE VALVULAS DE BLOQUEO PARA EL PASO DE COMBUSTIBLE. 2010: Quito

[8] Teletrol. Metso Automation, Sistema avanzado de supervisión, control y gerencia de información OASyS. 2011.

[9] Pozo, A.C., Módulos de adquisición y análisis para la interacción con dispositivos de campo en un SCADA. 2009, Universidad de las Ciencias Informáticas: La Habana.

[11] Instituto Mexicano de Soporte, 26 junio, 2013.

[12] Hernández, L.E.G., R.G. Johnson, and A.C.M. Borges, Manejadores GE Fanuc y BSAP Serial para el proyecto SCADA PDVSA. 2009.

[13] QT Creator. [En línea] [Citado el: 16 de marzo de 2015.] https://wiki.qt.io/Category:Tools::QtCreator_Spanish.

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

[15] Symantec, Informe Técnico: Criptografía. Disponible en http://enterprise.symantec.com

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

[17] Albuerne, Miguel Ángel. Trabajo de Diploma, “Simulador de dispositivos PLC-5 para realizar el proceso de pruebas al manejador DF1 del SCADA SAINUX”, Ciudad de La Habana, 2013.

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

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

[20] 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

[21] 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-enfoquepractico-roger-pressman-sexta-edicion

[22] 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.

[23] 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.

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

[25] 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

[26] Pérez Mariñán, P. “Patrones de Diseño (Design Patterns)”.

[27] Patrón iterador. [Citado el: 20 de mayo de 2015.] https://msdn.microsoft.com/eses/library/bb972240.aspx.

[28] Patrón estado. Marco de Desarrollo de la Junta de Andalucía. [En línea] febrero de 2014. [Citado el: 19 de junio de 2015.] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/202.

[29] Patrones de diseño. [En línea] [Citado el: 20 de mayo de 2015.] http://www.utnianos.com.ar/foro/attachment.php?aid=3577

[30] Acuña, Cesar Javier. Arquitecturas de Software, Universidad Rey Juan Carlos, España, 2013.

[31] Diagrama de Despliegue. ANALISIS Y DISEÑO DE SISTEMAS II. [Citado el: 15 de mayo de 2015.] virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc

[32] Fase de pruebas del sistema. Pruebas de Software. [Citado el: 15 de mayo de 2015.] http://pruebasdesoftware.com/laspruebasdesoftware.htm

[33] Campillo Santos, Eylin. Generación de productos de trabajo mediante Sistema de Apoyo en la disciplina de Requisitos. La Habana, Junio, 2016

[34] Ramos Macías, Reitel. Simulador para el protocolo de comunicación industrial Ethernet/IP. La Habana, Junio, 2015.

[35] Pardo Guerra, Enrique Yecier. Simulador para el protocolo de comunicación industrial AB Ethernet. La Habana, Junio, 2015.

Anexos

Tabla 20 HU de RF 6

Número: HU 6	**Nombre del requisito:** Cambiar clave del dispositivo
Programador: José Luis Medero	**Iteración Asignada:** 3
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: • Planificación incorrecta.	**Tiempo Real:** 8 horas
Descripción: El sistema deberá que el usuario puede cambiar la clave del dispositivo siempre y cuando cumpla con los parámetros.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 21 HU del RF 7

Número: HU 7	**Nombre del requisito:** Simular fallas de conexión
Programador: José Luis Medero	**Iteración Asignada:** 3

Prioridad: Media	**Tiempo Estimado:** 16 horas
Riesgo en Desarrollo: • Planificación incorrecta.	**Tiempo Real:** 32 horas
Descripción: El sistema deberá permitir al usuario simular fallas en la comunicación con el manejador, dándole la posibilidad al usuario de interrumpir la comunicación, de ser así es sistema mostrará un mensaje de Error en la Comunicación.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 22 HU de RF 8

Número: HU 8	**Nombre del requisito:** Simular variables
Programador: José Luis Medero	**Iteración Asignada:** 3
Prioridad: Media	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: • Planificación incorrecta.	**Tiempo Real:** **3**2 horas
Descripción: El sistema deberá permitir al usuario simular variables, brindándole al usuario la posibilidad de observar en pantalla el comportamiento de las variables insertadas.	
Observaciones:	

Tabla 23 Diseño de Pruebas Simular Variables

Escenario	Descripción	Tipo	Nombre	Dirección del dispositivo	Clave	Puerto	Valor	Respuesta del Sistema	Flujo Central
EC 1.1 Entrada de datos válidos.	Se deberán introducir los campos correctos para modificar una variable	V	V	V	V	V	V	Comienza la simulación de la variable.	Se selecciona los campos tipo, nombre, dirección del dispositivo, clave, puerto y valor y se insertan los datos correctamente luego se presiona el botón Simular.
		El tipo de dato introducido debe ser (int, bool o float).	El nombre de la variable debe ser con el formato base.extension.dato.	La dirección debe ser la configurada en el recolector gráfico.	La clave insertada en el simulador debe coincidir con la clave del recolector	El puerto insertado en el simulador debe coincidir con el puerto del recolector.	El valor seleccionado debe ir estar relacionado con el tipo de dato seleccionado.		
EC 1.2 Ausencia de campos llenos de datos válidos.	Se deberá mantener el campo de forma vacía.	V	V	V	I	I	V	Se muestra en pantalla el siguiente mensaje ¨Existen Campos Vacíos¨.	Se selecciona los campos tipo, nombre, dirección del dispositivo, clave, puerto y valor y se insertan los datos correctamente luego se presiona el botón Simular.
		El tipo de dato introducido debe ser (int, bool o float).	El nombre de la variable debe ser con el formato base.extension.dato.	La dirección debe ser la configurada en el recolector gráfico.	Nulo	Nulo	El valor seleccionado debe ir estar relacionado con el tipo de dato seleccionado		
EC 1.3 Entrada de campos no válidos	Se deberá introducir campos no válidos	V	V	V	I	I	V	No permite introducir datos incorrectos.	Se selecciona los campos tipo, nombre, dirección del dispositivo,
		El tipo de dato introducido debe ser (int, bool o float).	El nombre de la variable debe ser con el formato	La dirección debe ser la configurada en el	3456394	5464	El valor seleccionado debe ir estar relacionado con el tipo de		

			base.extension.dato.	recolectorgráfico.			dato seleccionado		clave, puerto y valor y se insertan los datos correctamente luego se presiona el botón Simular

Tabla 24 Diseño de Prueba Eliminar Variable

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Selecciona el botón Eliminar.	Al comenzar la simulación decide eliminar una variable.	Elimina la variable seleccionada.	Selecciona del listado de variables simuladas y luego el botón Eliminar.

Tabla 25 Diseño de Prueba Detener Simulación

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Selecciona el botón Detener.	Al comenzar la simulación se decide detener la misma.	Detiene la simulación de variables.	Simula el comportamiento de determinadas variables y luego selecciona el botón Detener.

Tabla 26 Diseño de Prueba Simular fallo de conexión

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Selecciona el *checkbox* Simular fallo de conexión.	Al comenzar la simulación se decide simular un fallo de conexión.	Continúa la simulación de variables pero no recibe peticiones del manejador	Simula el comportamiento de determinadas variables y luego selecciona el *checkbox* Simular fallo de conexión.