Universidad de las Ciencias Informáticas

Facultad 4

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

Implementación del Protocolo de red Distribuida en su versión 3.

Autor: Luis Ernesto Clarke Samuels

Tutor: Ing. Alberto Arístides Puentes González

La Habana, 7 de diciembre del 2022
"Año 64 de la Revolución"

(…) "La clave del éxito …está en detectar hacia dónde va el mundo y llegar ahí primero". (…)

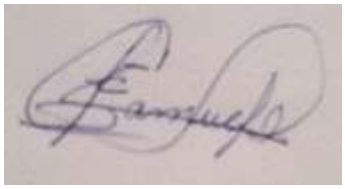
Bill Gates

Declaración de autoría

Declaro ser autor de la presente tesis que tiene por título: Implementación del Protocolo de red distribuida en su versión tres y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

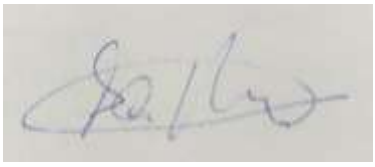
Para que así conste firmo la presente a los 7 días del mes de diciembre del año 2022.

Luis Ernesto Clarke Samuels



Firma del Autor.

Alberto Arístides Puentes González



Firma del Tutor.

Datos de Contacto

Autor:

Luis Ernesto Clarke Samuels

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: luisecs@estudiantes.uci.cu

Tutor:

Alberto Arístides Puentes

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: aapuentes@uci.cu

Dedicatoria

A mi familia que siempre me ha apoyado, en especial a mis padres, a mi hermana, a ese pequeñín de la casa que es mi razón de ser Thiaguito y a mi novia el amor de mi vida.

Agradecimientos

Primeramente, quiero agradecerle a mi madre, a ella que tanto me ha inspirado a no rendirme hasta cumplir mis metas y es el principal motivo de todos mis logros. A mi hermana que siempre me ha apoyado y ha sido como una madre para mí desde que tengo uso de razón. A mi padre que siempre me brindó su ayuda de forma incondicional, cumpliendo cada uno de mis caprichos acompañado siempre de sabias palabras de superación. Este título va dedicado a ustedes que nunca dudaron de mí, incluso cuando ni yo lo cría posible, gracias a eso ¡Hoy su niño es ingeniero!

También quisiera agradecerle a mi familia en general ya que han sido cruciales en esta etapa de mi vida, han sido un gran apoyo en esta larga carrera que por fin termina. Les agradezco por creer en mí, gracias por toda la ayuda que me han brindado siempre, por apoyándome y darme fuerzas para seguir adelante. Este título también es de ustedes. En especial dedicárselo a mi primo Daniel que en estos momentos está lejos, pero sé que sería muy feliz de estar aquí y celebrar esta gran victoria junto conmigo.

También quisiera dedicarle este logro a Daniela mi novia, la gratitud que siento hacia ti será lo más difícil de expresar en toda esta dedicatoria. Gracias por siempre confiar en mí, gracias por todo lo vivido, gracias por ser muchas veces mi único motivo a seguir, gracias por haber sido parte de esta etapa tan difícil de mi vida, gracias por todo te amo.

Me gustaría también agradecerle a mi facultad, a el claustro de profesores que me ha acompañado durante estos cinco años de superación. Empezando por mi entrenador y ejemplo a seguir Joe; gracias por todo ¡Ahora sí, por fin “El Eléctrico” se gradúa! También agradecerle de manera especial a la profe Yadira, de la cual que me atrevo a afirmar que ha sido como una madre para todo este quinto año, siempre dispuesta a escucharnos y a ayudarnos en todo lo que nos hiciera falta. A la profe Yasiris que indudablemente siempre me brindo su ayuda incondicional, aunque me metía un poco el pie para los festivales. Al profe Julio de quien estoy infinitamente agradecido, pues fue un gran apoyo en el desarrollo de este trabajo de diploma y siempre se preocupó por que saliera adelante. Por supuesto agradecerle a Andy mi oponente que más que un oponente ha sido un tutor y un ejemplo de superación, al cual le agradezco por toda la ayuda brindada y cada consejo. Y, por último, pero no menos importante agradecerle acompañado de un fuerte abrazo a mi tutor Albertico, gracias a él de cierta manera hoy se me cumple este sueño, un millón de gracias.

Me gustaría también agradecerles a mis amistades de toda la vida que de una forma u otra se han preocupado y han sido indispensables en estos años tan difíciles. Gracias

Felo, Roni y Adrián, de más está decir lo mucho que los amo, Gracias Meli, Roxana y Clau por tantos consejos y buenos momentos pasados juntos.

Por último, de manera general quiero agradecerle a mi familia UCI, a ustedes mi gente que los amo con la vida y nunca pero nunca voy a olvidar, gracias por tantos buenos y malos momentos pasados juntos, por tantas malas noches de estudio y fiestas, gracias por cada consejo y por siempre estar ahí, como la familia que considero que son. En especial quisiera agradecerle a Ferna y a Adrián, de nuevo, pues ambos han sido lo más cerca que he tenido de mi familia de la isla aquí en la UCI gracias “Corrillo”, agradecerle también a mi hermano prietorro Julito y a todos los de grupo de baile que juntos creamos, Latinos es hoy un sueño cumplido para ambos. Agradecerle a Yamisleidis mi hermanita, a Carlos (el mejor barbero de la UCI), a Damaris por tantos años de apoyo, a Felipe, a Jaydy, a Gretel, a Arian, , a Shabei, a Laura, a Kiyomys, a Ilennys, a Lili, a Yailet, a Pompa, a Ceilán, a Asael, a Dionis, a Jonathan, a Fernando,a Jesú, a Dairon, a Elier,a Roy, a Prebot, a Thalía, a Vivian, a la Cuca y a todos los que me faltan que son muchos. También quisiera de manera individual agradecerte a Samira, este título junto a casi todos los de este quinto año son tuyos también, gracias por ser tan especial.

Resumen

Los sistemas de redes manejan formatos bien definidos llamados protocolos de comunicación. A estos se les encomiendan la comunicación entre varios dispositivos, permitiendo comprender con mayor facilidad estas comunicaciones a través de las redes locales e Internet. El presente trabajo desarrolla una solución para la supervisión de procesos a distancia en las redes eléctricas a nivel nacional, mediante la implementación del protocolo de comunicación DNP3 para permitir el intercambio de información con los dispositivos Nulec. Para la elaboración del proyecto se emplea la metodología de Proceso Unificado Ágil (AUP-UCI) en su cuarto escenario, con el fin de continuar la línea de desarrollo trazada en el Sistema de medición y adquisición de Arex. Para el modelado de la solución se selecciona la herramienta Visual Paradigm. Y el Entorno de Desarrollo Integrado: QtCreator, como marco de trabajo en el lenguaje C++ para la implementación del protocolo DNP3. Finalmente se ha realizado un estudio de diferentes pruebas para validar el correcto funcionamiento del protocolo.

Palabras Clave:

Dispositivos, DNP3, Nulec, Protocolos, Redes.

Abstract

Network systems handle well-defined formats called communication protocols. These are entrusted with the communication between various devices, making it easier to understand these communications through local networks and the Internet. The present work develops a solution for the supervision of remote processes in electrical networks at a national level, through the implementation of the DNP3 communication protocol to allow the exchange of information with Nulec devices. For the elaboration of the project, the Agile Unified Process (AUP-UCI) methodology is used in its fourth scenario, in order to continue the line of development outlined in the Arex measurement and acquisition system. For the modeling of the solution, the Visual Paradigm tool is selected. And the Integrated Development Environment: QtCreator, as a framework in the C++ language for the implementation of the DNP3 protocol. Finally, a study of different tests has been carried out to validate the correct functioning of the protocol.

KEYWORDS

Devices, DNP3, Networks, Nulec, Protocols.

Índice

Índice de ilustraciones

Índice de Tablas

Introducción

Hoy en día con el aumento de las tecnologías y las innovaciones de la ciencia y la técnica, se ha propiciado un aumento de la complejidad de varios procesos industriales. Provocando la posibilidad casi nula de la manufactura y el soporte manual de los mismos. Por ende, se han creado varios estándares o protocolos de control a distancia como referentes para la comunicación local.

Con el objetivo de regular la comunicación entre los sistemas y la retroalimentación en tiempo real con los dispositivos de campo (sensores y actuadores). Proporcionando información que se genera en el proceso productivo (supervisión, control de calidad, control de producción, almacenamiento de datos, etc.) y permitiendo su gestión.

El protocolo DNP3, acrónimo del inglés *Distributed Network Protocol*, en su versión 3 o (protocolo de red distribuida) es un claro ejemplo de lo antes descrito. Este protocolo es creado a finales de los años 90 con la finalidad de obtener un protocolo libre y orientado a soluciones dentro de la industria eléctrica e hidráulica. Este es un protocolo integral que define las reglas mediante las que se comunican los ordenadores entre sí. El mismo define específicamente la interacción entre los sistemas informáticos para servicios públicos con la comunicación remota. Para este fin DNP3 se centra en ofrecer medios ligeros de transporte de valores de datos sencillos, con un alto grado de integridad.

En Cuba uno de los centros que lleva la delantera en este proceso de desarrollo de sistemas de monitoreo en la industria, es El Centro de Tecnología Interactiva (VERTEX) perteneciente a la Universidad de las Ciencias Informáticas (UCI). Este centro tiene entre sus objetivos principales el desarrollo de aplicaciones o servicios para la automatización de procesos industriales. A petición de la Unión Electica de Cuba (UNE) se pretende lograr la adaptación e implementación del protocolo DNP3. Para lograr gestionar la comunicación con dispositivos reconectores automáticos de redes eléctricas en el sistema de medición y adquisición de datos Arex.

Con este fin la UNE ha desplegado en diferentes zonas de la Ciudad de la Habana los dispositivos “Reconectadores Automáticos Nulec”, estos combinan varias funciones de: protección, monitoreo, medición, control, comunicación y calidad de energía, para proteger las líneas eléctricas, los transformadores y otros equipos de distribución de ser expuestos a altos niveles de voltaje. Por lo que se plantea que actualmente el sistema Arex 3.0 no cuenta con una solución para el intercambio de información con los dispositivos Nulec. Razón por la cual se quiere lograr la comunicación a través de la

implementación del protocolo DNP3, al ser más robusto, eficiente y complejo que otros protocolos más antiguos, tales como Modbus.

Los intereses investigativos que se expresan en la situación problemática anteriormente descritos, permitieron seleccionar como:

Problema científico

¿Cómo establecer un mecanismo de comunicación para permitir el intercambio de información con los dispositivos Nulec?

Teniendo en cuenta el problema científico antes expuesto se define como**:**

Objeto de estudio

Los Protocolos de comunicación en sistemas de medición y adquisición de datos, teniendo como **campo de acción**: El protocolo de comunicación de red distribuida en su versión 3.

La solución del problema podrá ser resuelto a partir del cumplimiento del

Objetivo general

Desarrollar el protocolo DNP3 como mecanismo de intercambio de información con los dispositivos Nulec.

Con el propósito de cumplir este objetivo fue necesario plantearse:

Objetivo Específico

1. Sistematizar el estado del arte de la temática objeto de estudio, elaborando el marco teórico referencial de la investigación.
2. Modelar mediante los artefactos ingenieriles la solución que sugiere la metodología de desarrollo de software seleccionada.
3. Implementar la solución (Protocolo de red distribuida en su versión 3).
4. Realizar las pruebas funcionales para el cumplimento de los requerimientos definidos.

Métodos Científicos

Con el objetivo de cumplir los objetivos específicos trazados se emplearon distintos **métodos de investigación**, los cuáles se separan en **métodos teóricos** y **métodos empíricos.**

Del Nivel Teóricos

- ✓ Analítico-Sintético: Se realizó un análisis de los documentos e información

referente a los protocolos de comunicación. Lo que permitió la identificación de las dificultades con que cuenta el proceso. Permitió comprender en esencia los fenómenos observados, separar lo esencial de lo secundario, determinar lo particularmente importante a partir de la bibliografía revisada y extraer los postulados teóricos necesarios para la solución del problema científico.

- ✓ Modelado: Permitió crear abstracciones con el objetivo de explicar la realidad. Es empleado como herramienta para la compresión del problema a resolver y para la creación de los modelos que permitan el diseño de la solución.

- ✓ Histórico-lógico: Proporcionó conocer todo lo referente a los protocolos de comunicación, específicamente el DNP3, tomando dichas observaciones como guía para el desarrollo de la presente investigación.

Del Nivel Empírico

- ✓ Observación: La observación aplicada consistió en presenciar y registrar los hechos tal y como se produjeron en la actividad práctica. Se apreciaron los procesos realizados, donde se obtuvo la información sobre el funcionamiento de la comunicación del protocolo DNP3.
- ✓ Entrevista: Se utiliza para la recopilación de información, ideas y opiniones de los especialistas del centro Vertex acerca de la investigación, contribuyendo así al desarrollo del futuro sistema informático y a la identificación de los requisitos del sistema.
- ✓ Análisis bibliográfico**:** Consistió en identificar, obtener y consultar las fuentes de información bibliográfica útiles para los propósitos del estudio.

Estructura del Trabajo de Diploma

El trabajo se desarrolla mediante una estructura que se compone de tres capítulos:

Capítulo1. Fundamentación Teórica. En este capítulo se concentra en la definición del marco teórico, así como la situación actual del objeto de estudio. Teniendo en cuenta las características principales del protocolo a desarrollar, además de las herramientas a usar posteriormente en la solución.

Capítulo2. Análisis y Diseño de la propuesta solución. En este capítulo se especifica la propuesta de resultado y se detallan las bases de la investigación que la sustenta. Se realizó un énfasis en el diseño de la propuesta de solución, compuesto por los requisitos funcionales y no funcionales.

Capítulo3. Implementación y pruebas del sistema. En este capítulo se describirán los estándares de codificación, los diagramas de componentes, los estilos de métodos. Buscando arribar a la implementación del protocolo, además de que se expondrán los conjuntos de pruebas realizadas al protocolo en cada versión.

A continuación, se exponen las conclusiones y las recomendaciones de la investigación teniendo en cuenta los resultados obtenidos. Además, se enlista la bibliografía pertinente, organizada y por último los anexos que apoyan los resultados expuestos a lo largo del trabajo.

Capítulo 1: Fundamentos Teóricos

En el siguiente capítulo se hará un análisis de la comprensión de la información con respecto a su marco teórico, el objeto de estudio y las herramientas a utilizar para su implementación. Para este capítulo se pretende concentrar la información disponible de los protocolos de red y su funcionamiento, haciendo énfasis en el protocolo DNP3.

1.1 Estudio del arte

Se considera que el estado del arte implica el desarrollo de una metodología resumida en tres grandes pasos: contextualización, clasificación y categorización, es decir, hacer el análisis (sinónimo de investigación) (Montoya, 2020) (universidad de la Salle, Bogotá, 2017). De esta manera se observa que la realización de estados del arte permite la circulación de la información. Genera una demanda de conocimiento y establece comparaciones con otros conocimientos paralelos a este, ofreciendo diferentes posibilidades de comprensión del problema tratado; pues brinda más de una alternativa de estudio.

1.1.1 Sistema de medición y adquisición de datos Arex

Es un sistema orientado a medir procesos domóticos. Está dirigido a servicios en áreas de Internet de las cosas para detectar el estado de las luces, temperatura y presencia de humedad en determinado lugar. Está diseñado para medir procesos de hasta 300 variables (Centro de Informática Industrial, 2019).

El sistema está compuesto por cinco módulos funcionales:

- ✓ Ejecución: Permite a los operadores, tener acceso a la información de los procesos que se miden.
- ✓ Recolección: Permite la comunicación directa con los dispositivos decampo, el procesamiento y adquisición de los datos.
- ✓ Históricos: Permite almacenar todos los datos recolectados y generados en el sistema, para que puedan ser consultados con posterioridad.
- ✓ Seguridad: Permite la autenticación de los usuarios.
- ✓ Edición: Permite al usuario crear y configurar dispositivos, variables y alarmas de un proceso, y la manera en que será visualizado el mismo mediante despliegues de componentes gráficos. Se pueden configurar además aspectos imprescindibles como alarmas y rutinas de control automático.

La comunicación entre el controlador lógico programable y la computadora se realiza mediante el protocolo de comunicación MODBUS utilizándola conexión Serie y/o Ethernet (Centro de informática, 2019). Por lo que se pretende lograr la adaptación del mismo al Protocolo DNP3. Ya que presenta características que permiten que el sistema presente una mayor flexibilidad y un mayor número de funcionalidades primordiales al momento de analizar fallas y sincronizar el accionamiento de todos los dispositivos.

1.1.2 RTU

Una unidad terminal remota (RTU) es un dispositivo de propósito general, utilizado para el monitoreo y control remoto de varios dispositivos y sistemas para la automatización. Por lo general, se implementa en un entorno industrial. Las RTU también se denominan unidades remotas de telemetría o unidades remotas de telecontrol.

Las RTU pasan los datos de los sensores desde los flujos de entrada en los bucles de control a un flujo de salida para ser enviados al control centralizado en un sistema de control industrial y distribuyen automáticamente las conexiones a los controles locales o remotos. Este dispositivo permite procesar y facilitar de manera continua información de estado y eventos. Presentan funciones predefinidas que permiten el manejo sencillo a los usuarios que no tienen conocimientos de programación. Estos dispositivos funcionan a través de las comunicaciones seriales.

Las RTU recolectan datos de campo, tanto analógico como digital. Una RTU obtiene los datos de los sensores que monitorean las variables objetivo del proceso industrial y luego envía esos datos a un monitoreo y control centralizados. El hardware de una RTU contiene el software de configuración necesario para conectar flujos de salida de datos, protocolos de comunicación y solución de problemas integrada.

1.1.3 Dispositivos Nulec

Los dispositivos Nulec o los reconectores automáticos de tres fases de la serie N son unidades RTU enfocadas a recolectar datos en diferentes circuitos eléctricos. Se han desarrollado para lograr un rendimiento y fiabilidad óptimos. Avalados por su trabajo seguro y fiable su diseño está optimizado para aplicaciones de automatización, control remoto, registro de datos y monitoreo. Un indicador claramente visible y conectado directamente proporciona una indicación visual de la posición del reconector.

Los reconectadores Nulec son dispositivos de protección empleados en redes de distribución primaria. Conjuntamente con su función de protección, se puede aprovechar otra propiedad suya: el registro de mediciones de los circuitos que protege, para la caracterización de sus cargas. Teniendo en cuenta esta segunda particularidad, se

puede mantener el control y evaluación de las redes eléctricas ante las variaciones de las magnitudes eléctricas (Arcias, 2015) (1Library, 2022).

1.1.4 Modelo OSI

El modelo OSI es el modelo de la interconexión de sistemas abiertos, y sus siglas provienen del inglés *Model Open System Interconection*. Este es un modelo de referencia para los protocolos de red. El modelo OSI estándar fue desarrollado en 1980 por una federación global de organizaciones que representan aproximadamente a 160 países (Fernandez, 2020). Esta normativa está formada por 7 capas que define las diferentes fases por las que deben pasar los datos para viajar entre dispositivos sobre una red de comunicaciones. En los cuales se usan tanto en la parte de sistemas como para la parte de redes.

En otras palabras, el modelo OSI, se utiliza principalmente para describir, discutir y comprender funciones de red individuales, al ser genérico, la mayoría de los protocolos y sistemas se adhieren a él de una forma u otra para su implementación. El modelo OSI está conformado por 7 capas (Ilustración 1).

Modelo OSI

Ilustración 1 Modelo OSI (FS. cummunity, 2021)

No todas las capas se utilizan en aplicaciones más simples. Si bien la capa física, de enlace de dato y de red son obligatorias para cualquier comunicación de datos, la aplicación puede usar alguna capa de interfaz única para la aplicación en lugar de las capas superiores habituales en el modelo (FS. cummunity, 2021).

El modelo OSI no es la definición de una tecnología, ni un modelo de red en sí mismo, lo que hace es definir la funcionalidad de ellos, para conseguir un estándar. Desde este modelo se han creado numerosos esquemas de protocolo de red.

1.1.5 Protocolos de red y comunicación

Se entiende por Protocolos de comunicación a un sistema de reglas y procedimientos que siguen mismos estándares y políticas formales. Conformados por restricciones, procedimientos y formatos que definen el intercambio de paquetes de información (Jithin, 2019). Para lograr la comunicación entre dos servidores o más dispositivos a través de una red. Permitiendo definir la semántica y estructura de los mensajes que se envían desde los dispositivos.

La interconexión de sistemas o redes de computadoras son la base de las comunicaciones hoy en día y están diseñadas bajo múltiples protocolos de comunicación (Margaret., 2019). Por ejemplo, existen muchos protocolos al establecer una conexión a internet y según el tipo que se necesite establecer, dichos protocolos van a variar. Además, la comunicación a internet no es el único tipo de comunicación cuando hablamos de transmisión de datos e intercambio de mensajes en redes. En todos los casos, los protocolos de red definen las características de la conexión.

1.1.6 TCP/IP

El modelo TCP/IP es un modelo conceptual utilizado para la descripción de las comunicaciones de red. A su vez, TCP/IP también es un protocolo importante que se utiliza en todas las operaciones de Internet. Generalmente, cuando hablamos de capa dos o capa tres en las que funciona un dispositivo de red, se hace referencia a la presencia de un modelo TCP/IP (Ilustración 2) (FS. cummunity, 2021).El modelo TCP/IP se usa tanto para modelar la arquitectura actual de Internet o para proporcionar un conjunto de reglas seguidas por todas las formas de transmisión a través de la red.

Modelo TCP/IP

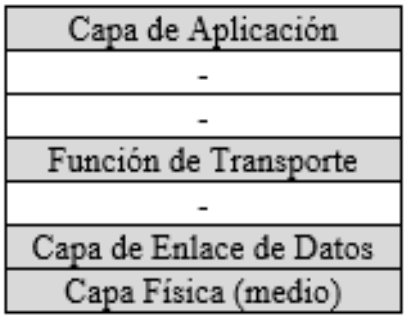


Ilustración 2 TCP/IP o EPA (FS. cummunity, 2021)

La capa de aplicación del modelo TCP/IP es similar a las capas cinco, seis, siete combinadas del modelo OSI, el modelo TCP/IP no tiene una capa de sesión. La capa

de transporte de TCP/IP abarca las responsabilidades de la capa de transporte OSI y algunas de las responsabilidades de la capa de sesión OSI (FS. cummunity, 2021). La capa de acceso a la red del modelo TCP/IP abarca el enlace de datos y las capas físicas del modelo OSI.

Según el significado y el funcionamiento del TCP/IP, las principales características de este modelo son:

- ✓ Está diseñado para resolver un conjunto específico de problemas.
- ✓ No está creado para funcionar como una descripción de generación para todas las comunicaciones de red.
- ✓ Se basa en protocolos estándar que internet ha desarrollado.
- ✓ Se utilizan cada una de las capas para cualquier comunicación de datos.

Es importante destacar que cada paquete de datos enviado puede tomar una ruta diferente entre la computadora de origen y el destino. Dependiendo de si la ruta original utilizada está congestionada o no disponible para que puedan negociar, separar y transferir conexiones. El protocolo TCP/IP se utiliza para realizar la comunicación entre la red informática. Sirve para hacer el login de red remoto, transferencia de archivos interactiva y la entrega de correo electrónico y páginas web.

1.1.7 Conclusión sobre el análisis realizado

Luego del análisis realizado en los epígrafes anteriores, se puede arribar a la conclusión de que el protocolo DNP3 cuenta con una arquitectura ajustada al modelo TCP/IP con algunas características del modelo OSI de siete capas. Incluyendo a su vez la comunicación entre RTU, de maestro a esclavo y aplicaciones de red. A demás de que constará de tres niveles de implementación, donde la capa de aplicación tendrá un pseudo nivel de transporte, el cual no será considerado un nivel como tal, debido a su forma de implementación.

Se concluye que el protocolo a implementar, al ser un estándar basado en protocolos de red ya definidos, es usable en aplicaciones de supervisión, control y adquisición de datos lo que cumple con las especificaciones necesarias para su desarrollo, cumpliendo con el objetivo general del presente trabajo de diploma.

1.2 Descripción del Protocolo de red Distribuida 3 (DNP3)

El protocolo industrial DNP3 hoy es un protocolo abierto y público (sin propietarios), DNP3 fue desarrollado por la empresa norteamericana Harris en noviembre de 1993, pero luego la responsabilidad de mantener y definir futuras especificaciones fue

entregada al Grupo de Usuarios de DNP3, un grupo compuesto por usuarios y fabricantes, actualmente con más de 300 miembros (network solutions, 2017).

El DNP3 se centra en ofrecer medios ligeros de transporte de valores de datos sencillos con un alto grado de integridad. Ha sido ampliamente utilizado en el sector eléctrico, en donde las estampas y sincronizaciones de tiempo, como el hecho de que un esclavo transmita información sin ser solicitada, son fundamentales al momento de analizar fallas y sincronizar el accionamiento de todos los dispositivos (Punzenberger, 2019). Algo esencial para el éxito del desarrollo de este proyecto, es considerar la compatibilidad que se tendrá el protocolo con otros equipos en el futuro, así como el diseño y la facilidad de implementación del mismo. También varios fabricantes de servidores de comunicación, han desarrollado drivers para los dispositivos que disponen de este protocolo con el objeto de integrarlos a las aplicaciones de control supervisor y adquisición de datos.

1.2.1Arquitectura del DNP3

El protocolo es tan extenso que desarrollarlo completamente en todos los dispositivos seria innecesario. Es por esto que el estándar define cuatro niveles de implementación. Un nivel de enlace (Data Link Layer), un nivel de Aplicación (Application Layer), y un tercer nivel de Transporte (Transport Layer). Este último realmente no cumple con todas las especificaciones del modelo OSI, y por lo cual se suele denominar pseudo-nivel de Transporte (Punzenberger, 2019) (Ilustración 3).

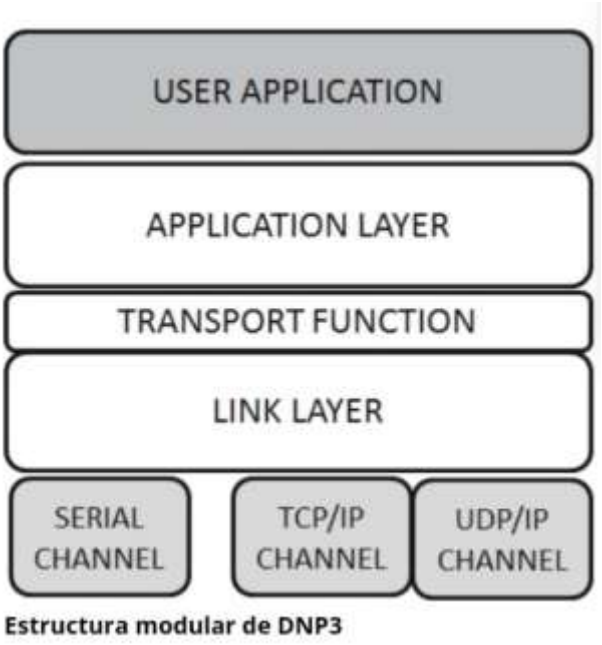


Ilustración 3 Estructura modular del DNP3 (Day, 2022)

1.2.2 Terminales del DNP3

El DNP3 define dos tipos de terminales que se comunican entre sí a través de medios de comunicación físicos un maestro y una unidad remota (Ilustración 4). Ambas presentan la misma estructura y se definen de la siguiente manera:

- ✓ **El maestro**

  El maestro es un ordenador o una red utilizados en un centro de control. Este ordenador es potente, almacenando todos los datos recibidos desde fuentes de unidades remotas y procesándolos para su visualización.

- ✓ **La unidad remota**

  También conocida como esclavo, la unidad remota es un ordenador que se utiliza sobre el terreno. Estas unidades remotas recopilan información de muchos dispositivos en distintas ubicaciones, como sensores de corriente y transductores de tensión, y transmiten la información a la estación maestra (freyrscada, 2017). De manera alternativa, una unidad remota DNP3 puede ser un dispositivo remoto que se comunica directamente con el maestro, como una *Remote Terminal Unit* (RTU) o un *Intelligent Electronic Device* (IED), un caudalímetro de agua o un medidor de flujo de energía, un inversor FV o cualquier tipo de estación controlada.

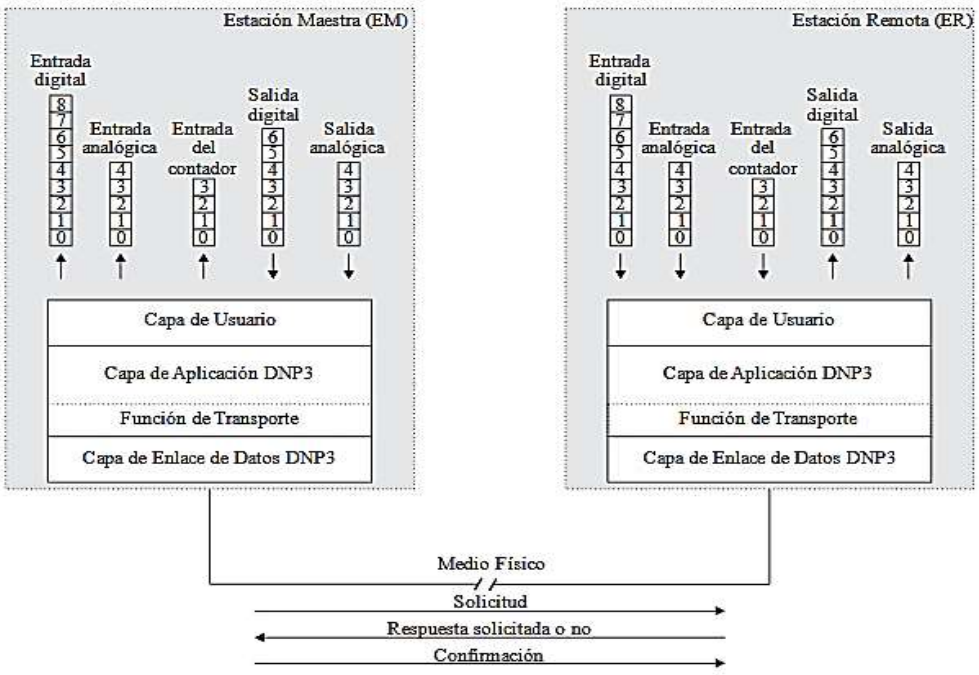


Ilustración 4 Esquema de la composición básica del protocolo DNP3 (Gómez, 2008)

El DNP3 define las variables de datos por tipo y comportamiento y las prioriza en función de; si representan o no un cambio del estado de base. Las variables con la que cuenta el protocolo DNP3 puede clasificarse en varios tipos: entradas binarias, salidas binarias, entradas analógicas, salidas analógicas, en contadores y entradas binarias dobles (muy común en interruptores y seccionadores eléctricos) (Clemente Pedrico, (2017)). Todos estos valores y reglas son configurados por el maestro al inicio mediante un sondeo de integridad, que solicita a la unidad remota que envíe el valor y el estado de todos los puntos configurados al maestro.

Después de este proceso de configuración, la unidad remota transmite selectivamente los eventos en función de si los datos han cambiado desde el último sondeo. Estas transmisiones suelen producirse según un programa cíclico, pero pueden enviarse espontáneamente si se cumplen ciertos parámetros.

Estas **reglas de comunicación** permiten que las unidades maestras y las unidades remotas se comuniquen utilizando un ancho de banda limitado para transportar valores de datos y comandos simples entre los dos extremos del protocolo (Clemente Pedrico, (2017)). Esto permite que las señales se envíen a través de enlaces en serie, multipunto, radioenlaces, conexiones de marcado y a través de redes dedicadas mediante TCP/IP o UDP.

Gracias a la **adaptabilidad** del protocolo, el DNP3 puede gestionar la mayoría de los escenarios de interrupción de la conexión, creando un protocolo de comunicación muy resiliente con pocos errores o fallos (freyrscada, 2017).A demás de presentar mayor flexibilidad y fiabilidad para el desarrollo del estándar DNP3 y su adopción para la comunicación remota en la industria.

El protocolo posee **interoperabilidad** pues es tan extenso que desarrollarlo completamente en todos los dispositivos seria innecesario. Es por esto que el estándar define cuatro niveles de implementación, cada uno con un nivel de complejidad mayor al anterior (Day, 2017). La regla es que un dispositivo esclavo de nivel x tiene que ser controlado por un dispositivo maestro de nivel x+1, por lo que el primer nivel está destinado únicamente para sistemas esclavos sencillos. Esta diferenciación se hace evidente sólo en la capa de aplicación ya que tanto las capas de transporte como la de enlace de datos deben ser prácticamente iguales en todos los dispositivos.

1.2.3 Mensajes del DNP3

El DNP3 presenta los datos y la estructuración en capas de forma jerárquica. La comunicación entre sus capas se realiza mediante mensajes, estos sufren cambios al pasar por cada capa, quedando de la siguiente forma (ilustración 5):

- ✓ Los mensajes a nivel de aplicación son denominados **Fragmentos**. El tamaño máximo de un fragmento está establecido en 1024 bytes (IEC, 2006).
- ✓ Los mensajes a nivel de transporte son denominados **Segmentos**. Este nivel es el encargado de adaptar los Fragmentos para poder encapsularlos, para lo cual, secciona el mensaje del nivel de aplicación si es necesario, y les agrega la cabecera de transporte (freyrscada, 2017).
- ✓ Los mensajes a nivel de enlace son denominados **Tramas**. El tamaño máximo de una trama DNP3 es de 292 bytes. En este nivel, los segmentos recibidos del nivel de transporte son empaquetados y se les añade a estos una cabecera de enlace. A demás, cada 16 bytes una comprobación de redundancia clínica (CRC) de dos bytes para la detección de errores (Punzenberger, 2019).

Cuando se **reciben** los datos a través de los mensajes entre las capas se suceden de mediante el siguiente proceso:

- ✓ El nivel de enlace se encarga de extraer de las tramas recibidas los Segmentos que son pasados al nivel de transporte.
- ✓ El nivel de transporte lee la cabecera de los segmentos recibidos del nivel de enlace, y con la información obtenida extrae y compone los fragmentos que serán pasados al nivel de aplicación.
- ✓ Respectivamente el nivel de aplicación analiza los fragmentos y los datos que son procesados según el modelo de objetos definido por las especificaciones del estándar.

Cuando se **transmiten** datos, estos sufren las siguientes transformaciones al pasar por las diferentes capas (Punzenberger, 2019):

- ✓ Los datos se encapsulan en fragmentos a nivel de aplicación.
- ✓ El nivel de transporte es el encargado de adaptar los Fragmentos para poder encapsularlos en segmentos, para lo cual, secciona el mensaje del nivel de aplicación si es necesario, y les agrega la cabecera de transporte, formando de este modo los segmentos.
- ✓ En el nivel de enlace, los segmentos recibidos del nivel de transporte son empaquetados en tramas, para lo cual se les añade a estos una cabecera de

enlace y, además, cada 16 bytes un Comprobación de redundancia clínica (CRC) de 2 bytes para la detección de errores.

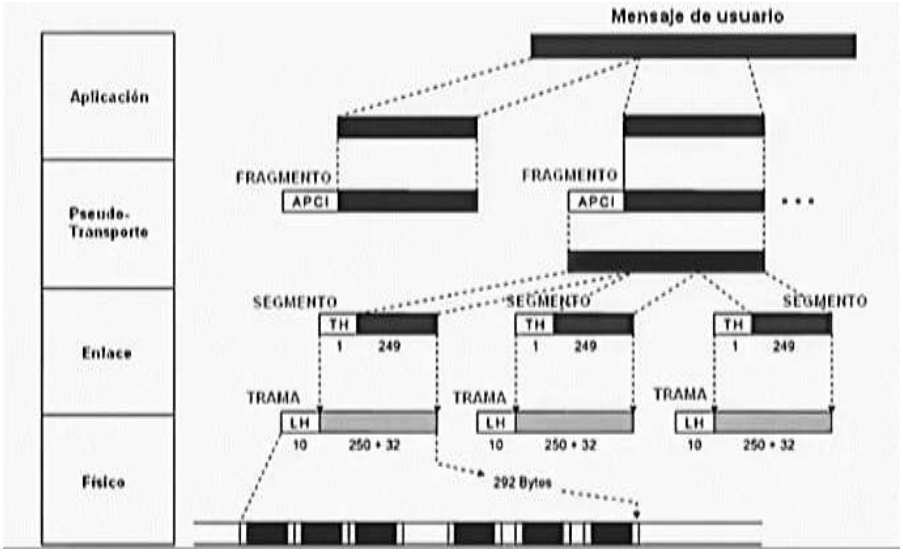


Ilustración 5 Composición de mensaje en DNP3 (Queens University Belfast, 2010)

A continuación, se describe la secuencia de eventos para enviar una orden del nodo maestro, el cual 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 adecuado:

1. En la estación maestra se recibe la señalización de un determinado punto a través de la capa física y se inicia un proceso de solicitud en la capa de aplicación.
2. La capa de aplicación fragmenta la información recibida y pasa los fragmentos obtenidos a la función de transporte.
3. Esta toma los fragmentos y los segmenta para pasarlos a la capa de enlace de datos obteniéndose los segmentos, para generar la trama DNP3 y enviarla a través del medio físico existente.
4. Luego la capa de enlace de datos del nodo remoto recibe la trama del medio físico. Elimina la cabecera de la trama obtenida y pasa la información a la función de transporte donde está, une los segmentos y pasa a la capa aplicación.
5. Después la capa de aplicación desfragmenta la información recibida e indica la solicitud a la capa física del nodo remoto correspondiente.

6. Para finalizar y darle respuesta a la solicitud del nodo maestro; el nodo remoto realiza el proceso inverso y espera por la confirmación del envío de los datos.

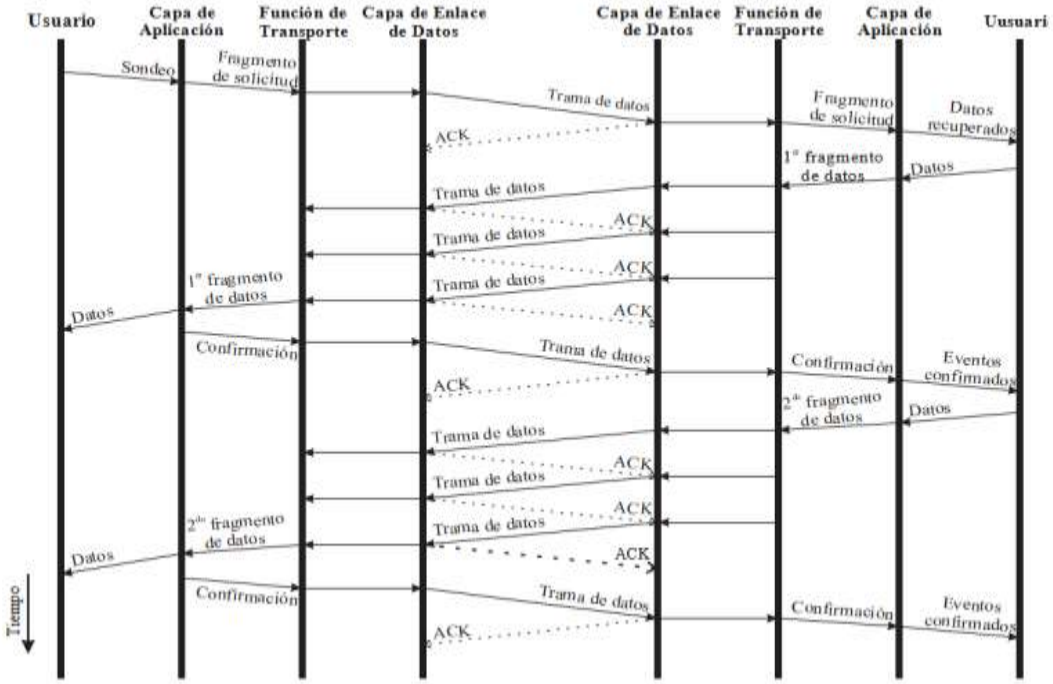


Ilustración 6 Flujo de mensajes en DNP3 (Ortiz, 2011)

En la (Ilustración 6) se muestra el flujo de mensajes desde un usuario el cual sería el nodo maestro a otro, el nodo esclavo (Ortiz, 2011). Con el fin de explicar el proceso de mensajes entre cada una de las capas de este protocolo.

1.2.4 Nivel de aplicación

El nivel o capa de aplicación se encarga de procesar los fragmentos que le pasa el nivel de transporte, y obtener la información de control y monitorización en ellos encapsulados atendiendo al modelo de datos.

Entre los servicios que proporciona este nivel, se encuentran la escritura y lectura de valores, la congelación de contadores y la selección y ejecución de mandos (Crespo Guerra, 2021). El código de función es el que permite indicar qué operación debe realizarse en este nivel.

Por otro lado, las estaciones controladas disponen de la posibilidad de informar a la estación controladora de diferentes aspectos relacionados con este nivel gracias a dos bytes denominados indicaciones (*Internal Indications, IIN*). La estación controlada puede servirse de estas indicaciones para informar acerca de la presencia de, de la necesidad de ser sincronizada o de la presencia de anomalías en la configuración o en la base de datos (IEC, 2006) (network solutions, 2017).

Entre las funciones básicas de DNP3 están (Day, 2022):

- ✓ 1: *READ*: lectura de valor estático o eventos.
- ✓ 2: *WRITE*: escritura de atributos del equipo.
- ✓ 3: *SELECT*: servicio de control de selección.
- ✓ 4: *OPERATE*: orden tras selección.
- ✓ 5: *DIRECT_OPERATE*: orden directa sin selección.
- ✓ 7: *IMMED_FREEZE*: orden de congelado inmediato.

En otras palabras, la capa de aplicación define Funciones (*service*) que se intercambian entre estaciones controladoras (*Master*) y estaciones o unidades remotas (*Out Station*).

Código de control de la Capa de Aplicación

Este proporciona la información necesaria para construir y reensamblar fragmentos de varios mensajes, e indicar sí el receptor debe devolver un mensaje desde la Capa de Aplicación. Además, ofrece información para evitar duplicar mensajes. El código de control se divide en cinco campos, los cuales se describen a continuación (Queens University Belfast, 2010) (Day, 2022), (ilustración 7):

- ✓ FIR: Campo de un sólo bit; cuando FIR=1 indica que es el primer fragmento de un mensaje, si FIR=0 indica que no es el primer fragmento del mensaje.
- ✓ FIN: Campo de un sólo bit; cuando FIN=1 indica que es el fragmento final del mensaje, de lo contrario el mensaje aún no termina.
- ✓ CON: Campo de un sólo bit que indica sí el receptor debe devolver un mensaje de confirmación desde la Capa de Aplicación.
- ✓ UNS: Campo de un sólo bit que cuando se habilita indica que el mensaje contiene una respuesta no solicitada o la respuesta a un mensaje no solicitado. Cuando el valor de UNS=0 indica que la secuencia de números está asociada a una solicitud o un mensaje de respuesta solicitada. (IEEE, 2011)La unidad remota habilita este bit en fragmentos que contienen una respuesta no solicitada y lo inhabilita cuando envía una respuesta solicitada. Mientras que la unidad maestra lo habilita en fragmentos de confirmación de la Capa de Aplicación para confirmar la recepción de una respuesta no solicitada. Luego lo inhabilita en fragmentos de confirmación de la Capa de Aplicación al confirmar la recepción de una respuesta solicitada.
- ✓ SEQ: Este campo consta de cuatro bits y se utiliza para verificar que los fragmentos sean recibidos en el orden correcto y para detectar fragmentos

duplicados. El campo SEQ tiene un rango de valores de cero a 15 que se incrementa mediante un contador que realiza operaciones de módulo 16. Lo anterior se realiza para evitar repetir fragmentos de solicitud de la unidad maestra, fragmentos subsecuentes después del primer fragmento en mensajes que contienen fragmentos múltiples y fragmentos de respuesta no solicitada (IEEE, 2011). Todos los dispositivos en la red deben mantener la secuencia de números independientes para cada dispositivo con el que se comunica, en donde una secuencia se utiliza para las peticiones solicitadas, las respuestas y confirmaciones, y otra secuencia se utiliza para las respuestas no solicitadas y confirmaciones; cabe señalar que no existe relación entre ambos números de secuencia.

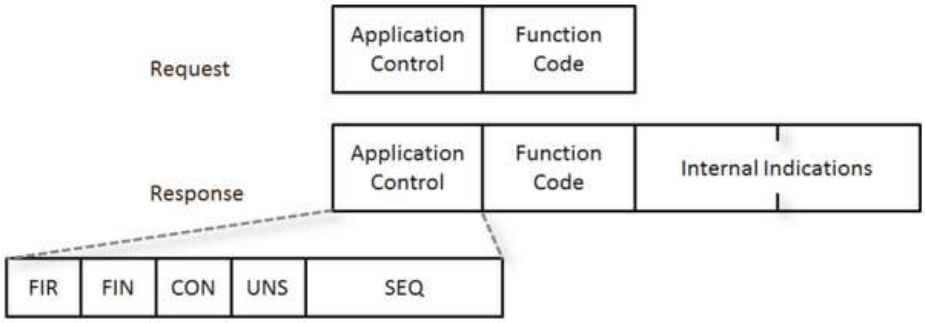


Ilustración 7 Cabecera de los fragmentos DNP3 en la capa aplicación (Day, 2022)

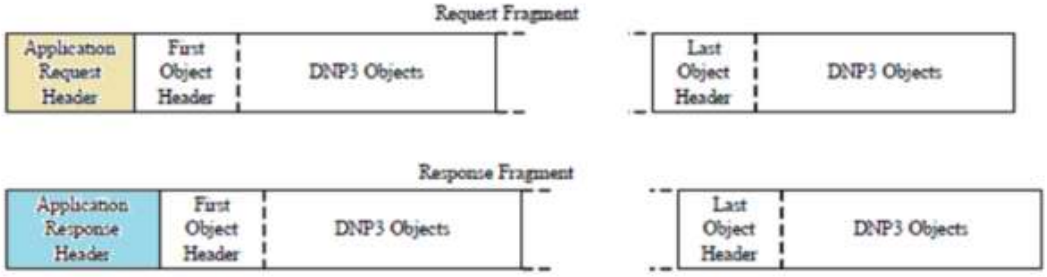


Ilustración 8 Formato de los fragmentos DNP3 (Day, 2022)

Un fragmento puede estar formado por secciones. Las secciones a su vez pueden incluir objetos DNP3 (ilustración 8). En la capa de aplicación los fragmentos se definen como: fragmento de petición (*Request*) y fragmento de respuesta (*Response*), siendo la principal diferencia entre ambos que el fragmento de respuesta lleva consigo un campo de 16 bits denominado *Internal Indications* con información de estado de la estación remota o del resultado de la Función solicitada.

Máquina de estados de la unidad remota

El propósito de la máquina de estados es especificar el comportamiento de la unidad remota en la recepción y transmisión de los fragmentos. Por cada fragmento recibido, la unidad remota examina los bits FIR, FIN y UNS. Además del valor del campo *SEQenelby* te de control y el código de función de la Capa de Aplicación (Queens University Belfast, 2010). También debe almacenar los valores SEQ de los bytes del fragmento de solicitud que se aceptan y de los fragmentos de respuesta que se transmiten. La unidad remota sólo acepta fragmentos de solicitud que contengan solicitudes válidas y que cumplan con los criterios establecidos en la tabla de estados, de lo contrario se descartan los fragmentos y se mantiene en el mismo estado.

Máquina de estados de la unidad maestra

Para fragmentos de solicitud El propósito de la máquina de estados de la unidad maestra es especificar su comportamiento cuando se reciben fragmentos con el bit UNS=0 en el byte de control de la Capa de Aplicación (Eaton, 2017). (freyrscada, 2017) El software de la unidad maestra necesita examinar los bits FIR y FIN, y el valor del campo SEQ en el byte de control de la Capa de Aplicación. También es necesario almacenar dichos bits junto con todos los bytes del último fragmento aceptado del envío de una solicitud de respuesta.

La unidad maestra acepta el fragmento sí cumple con los criterios establecidos, de lo contrario se descarta el fragmento. La unidad maestra no necesita almacenarlos bytes de control de la Capa de Aplicación o los valores de cualquier byte de los fragmentos descartados (Queens University Belfast, 2010). La unidad maestra también debe recordar el valor del campo SEQ a partir del último fragmento de solicitud que se envió a una unidad remota. Una unidad maestra debe mantener un conjunto separado de variables para cada unidad remota con la que se comunica.

Las variables incluyen: Los bits FIR y FIN, y el valor del campo SEQ a partir del fragmento del último fragmento aceptado de una respuesta solicitada. También los bytes del fragmento recientemente aceptados de una respuesta solicitada. Y el valor del campo SEQ del fragmento de solicitud recientemente transmitido.

1.2.5 Nivel de transporte

El nivel de transporte es el encargado de permitir mensajes únicos estructurados tanto en múltiples tramas como en múltiples fragmentos. Esta es una de las características diferenciadoras de DNP3 frente a otros protocolos de comunicación industriales del mismo ámbito, tales como IEC 60870. Permite el concepto de mensajes de tamaño ilimitado (freyrscada, 2017). La función de transporte es en realidad una subcapa de la

capa de aplicación que se sitúa en la parte inferior de la capa de aplicación, justo por encima de la capa de enlace de datos.

Todos los mensajes hacia y desde la Capa de Aplicación pasan por la Función de Transporte en su camino desde y hacia la otra estación. En otras palabras, el nivel de aplicación pasa los fragmentos al nivel de transporte, y este se encarga de trocearlos y agregarles al principio la cabecera de transporte. La cual ocupa un byte y contiene el número de secuencia que identifica el segmento dentro del fragmento (Day, 2022) (IEC, 2006). Cada segmento lleva un conjunto de bytes del fragmento y una cabecera de transporte. Dicha cabecera de un byte presenta el siguiente formato (ilustración 9):

- ✓ FIR: indica que este segmento es el primero de un fragmento.
- ✓ FIN: indica que este segmento es el último de un fragmento.
- ✓ SEQ: contador incremental para garantizar que no se pierden o duplican segmentos intermedios con los bits FIR y FIN desactivados.

Ilustración 9 Cabecera de la función de transporte (Eaton, 2017)

El tamaño de los fragmentos ha de ser tal, que una vez agregadas las cabeceras del nivel de enlace (diez bytes) y las correspondientes comprobaciones de redundancia cíclica (CRC). El tamaño total no exceda los 292 bytes máximos permitidos para una trama (ilustración 10).

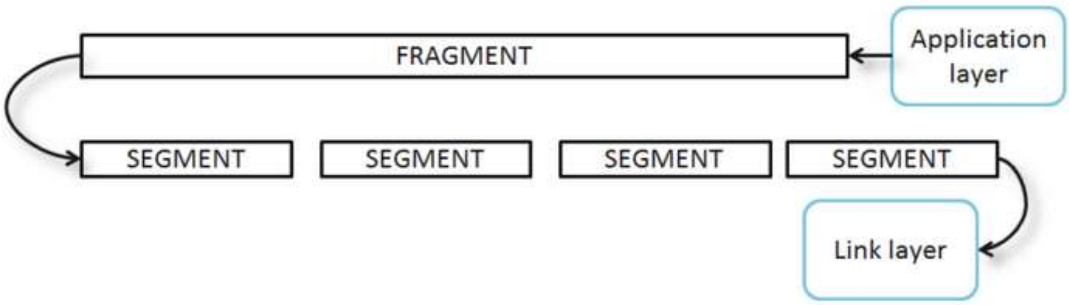


Ilustración 10 Segmentado de fragmentos (Day, 2022)

1.2.6 Nivel de enlace

Los mensajes DNP3 a nivel de enlace se encuentran en bloques de no más de 292 bytes denominados tramas (freyrscada, 2017). Esta capa se encarga de tareas de

direccionamiento y detención de errores. También permite el uso de confirmación de tramas para exigir al receptor la confirmación de cada mensaje. En enlaces TCP/IP la funcionalidad de la capa de enlace no se usa ya que TCP/IP permite por si solo garantizar el envío y recepción de tramas. La cabecera del DNP3: son los diez primeros bytes de la trama, y está constituida por los siguientes campos (Ilustración 11) (Clemente Pedrico, (2017)):

- ✓ Start: bytes de inicio (*start bytes*), cuyo valor es fijo. 0x05 (valor en hexadecimal) para el primero y 0x64 para el segundo.
- ✓ Len: Un byte con el tamaño de la trama.
- ✓ Ctrl: Un byte con el código de control, que permite fijar los servicios del nivel de enlace, el sentido del flujo, etc.
- ✓ Destination**:** Dos bytes con la dirección de destino, codificada en *big-endian.*
- ✓ Source: Dos bytes con la dirección de origen, codificada en *big-endian.*
- ✓ CRC: Dos bytes de comprobación de redundancia cíclica (Clemente Pedrico, (2017)).

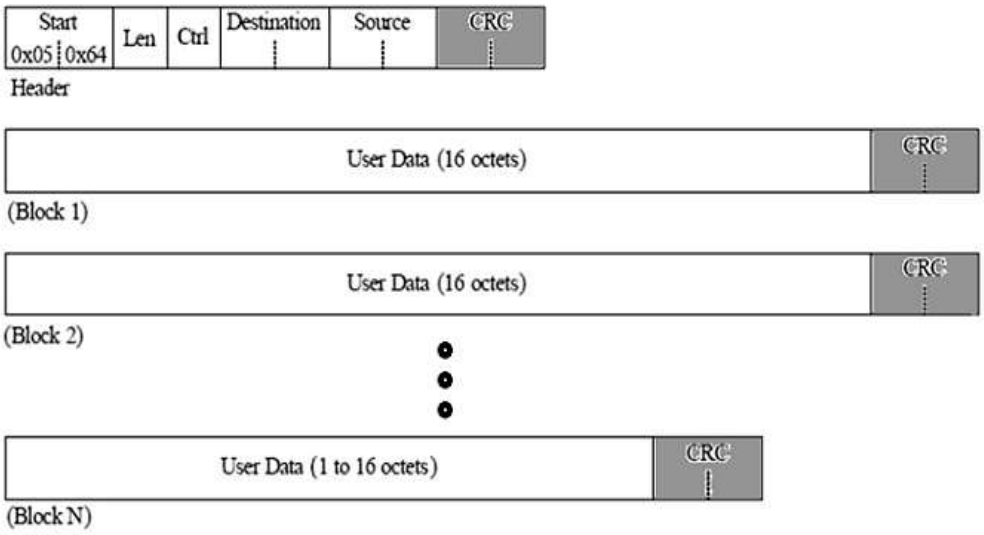


Ilustración 11 Trama de enlace (Day, 2022)

Las Comprobación de redundancia clínica (CRC)se utilizan para la detección de errores. Cada trama transmitida contiene una palabra CRC por cada 16 octetos de datos y una para la cabecera de la trama de enlace de datos. Estas palabras CRC se comprueban en cada trama recibida para que sólo se transmitan datos válidos a la función de transporte.

El nivel de enlace en DNP3 es balanceado, de modo que tanto la estación controladora como la controlada tienen responsabilidad tanto en los envíos de los datos como en la gestión (establecimiento y liberación) del nivel de enlace (fuera del alcance de las especificaciones del protocolo).

El nivel de enlace proporciona una serie de servicios para la gestión de la comunicación entre las estaciones, tales como:

- ✓ La petición o envío con o sin confirmación.
- ✓ Las confirmaciones de tramas recibidas (*ACK*).
- ✓ Las confirmaciones negativas (*NACK*).
- ✓ El reinicio de enlace (*Reset Link*).
- ✓ El chequeo del estado del enlace (*Link Status*).

El empleo de doble direccionamiento (dirección de origen y dirección de destino) se debe a la funcionalidad que proporciona DNP3 basado en funcionamiento por excepción. De tal modo las comunicaciones no son iniciadas únicamente por la estación controladora, enviando preguntas a las estaciones controladas. Sino que además estas últimas pueden iniciar una conversación dependiendo de la alteración de determinada información configurada en ella para ser reportada en estas condiciones (Crespo Guerra, 2021). A este tipo de mensajes, en los cuales la estación controlada transmite los eventos de determinados objetos configurados en ella, se les denomina "respuestas no solicitadas"(ACK) (IEC, 2006).

1.2.7 Modelo de objetos en DNP3

El modelo de objetos en DNP3 permite, en rasgos generales, definir los tipos de datos que se manejarán en las diferentes transacciones entre estación controlada y estación controlante. A pesar de ello, existen objetos orientados más bien a servicios a nivel de aplicación que a formato de datos en sí, como pueden ser el objeto 60 (Objeto de Clase) (FS. cummunity, 2021).

Mediante las denominadas variaciones, es posible establecer, además del tipo de dato definido por el objeto, el formato del mismo (tamaño y formato de los valores, por ejemplo).

A continuación, se describen algunos de los objetos más comúnmente utilizados en comunicaciones DNP3:

Tabla 1 Algunos objetos del DNP3 (Fuente: Elaboración propia)

Tipos	Descripciones
Entradas digitales	Este objeto hace referencia a las entradas digitales. Permite la lectura de las mismas, mediante el código de función uno, o la asignación de clase mediante el código de función 22 (códigos de función también soportados por otros objetos como contadores o entradas analógicas
Eventos de las entradas digitales	Es importante destacar que DNP3 maneja los valores estáticos y sus eventos como objetos diferentes. Así, los eventos de las entradas digitales (objeto uno) se agruparán en el objeto.
Mandos digitales	Este objeto hace referencia a los controles digitales. Mediante los códigos de función de selección, ejecución, selección y ejecución y ejecución sin confirmación, se podrán realizar estas operaciones sobre los elementos especificados bajo este objeto.
Contadores	Mediante este objeto, DNP3 permite la lectura o manipulación (congelación, reseteo, etc.) de contadores.
Eventos de contadores	Este es el objeto utilizado para agrupar la información relativa a eventos generados por contadores.
Entradas analógicas	Los valores analógicos se agrupan bajo este objeto.
Eventos de las entradas analógicas	Este es el objeto utilizado para los eventos de las entradas analógicas definidas mediante el objeto 30.
Mandos analógicos	Este es el objeto utilizado para ejecutar mandos analógicos o *Set Points*. Admite las mismas funciones que los mandos digitales.
Hora y fecha	La variación uno de este objeto permite a la estación controladora sincronizar a la estación controlada.

Objeto de clase	Como se comentó más arriba, este objeto no distingue exactamente entre un tipo de dato, sino más bien hace alusión a una serie de servicios del nivel de aplicación. Dependiendo del código de función utilizado, mediante este objeto la estación controlada puede realizar peticiones por clase, o asignar clases a los eventos de los objetos estáticos configurados en la estación controlada. (IEC, 2006)

1.2.8 Resumen de funcionalidades del DNP3

Las principales funcionalidades con que cuenta el protocolo DNP3 son:

- ✓ Direccionamiento (*Addressing*): Se utilizan dos Bytes para identificar la dirección origen y dos Bytes para identificar la dirección destino, con lo cual se tiene una capacidad de direccionamiento de hasta 65,000 dispositivos en un enlace simple.
- ✓ Mecanismo de confiabilidad mediante CRC múltiple (*Reliability Mechanism*): Implementa mediante una comprobación de redundancia clínica (CRC) de 16 bits de longitud por cada bloque de 16 bytes de los 255 bytes de datos que conforman la trama.
- ✓ Tipo de trama (*Frame Format*): Soporta transmisión asíncrona de FT3.
- ✓ Reconocimiento (*Acknowledgement*): Utiliza 10 Bytes para indicar el éxito de la comunicación y el establecimiento del enlace.
- ✓ Procedimiento (*Procedures*): Sólo permite establecer procedimientos balanceados.
- ✓ Flexibilidad: Soporta las topologías Maestro/Esclavo, red de igual a igual (*peer-to-peer*) y aplicaciones de red.
- ✓ Agrupamiento de los eventos en niveles diferentes (recurso útil para priorizar el esquema de peticiones del maestro). Las peticiones por clase permiten (el *polling*) por eventos. Un caso excepcional es una cuarta clase denominada clase 0, mediante la cual el maestro puede solicitar al esclavo el envío de todos los valores estáticos configurados en su base de datos (IEC, 2006).

A manera de conclusión se evidencia que con los análisis minuciosos realizado. sobre las descripciones de las políticas, conceptos, normas, operaciones, definiciones y restricciones presentes en el Protocolo de Red Distribuida (DNP3). Quedan plasmadas las actividades a computarizar en el futuro protocolo a desarrollar. Dándonos una visión de cómo quedaría estructurado la implementación del protocolo DNP3.

1.3 Metodologías de desarrollo y herramientas

Para la realización del proyecto se prevee la utilización de herramientas y metodologías. Se quiere que permitan determinar las insuficiencias y peculiaridades principales a tener en cuenta sobre el mismo. Para ello se realiza un análisis previo a las disimiles alternativas efectivas, donde posteriormente se elegirá la herramienta o metodología más idónea para darle solución a la problemática.

1.3.1 Metodología AUP-UCI

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software. (Sommerville, 2011) Su objetivo es guiar a un equipo de desarrollo durante la creación de un nuevo proyecto. Pero los requisitos a tener en cuenta para ello son tan variados que han dado lugar a diferentes enfoques o interpretaciones sobre el proceso de desarrollo de software.

El proceso unificado ágil AUP, por sus siglas en inglés (*Agile Unified Process*) de Scott Ambler, es una versión simplificada de la metodología de Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles, y al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva (Pressman, 2010).

Estas fases son:

1. Inicio: el objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. Elaboración: el objetivo es que el equipo de desarrollo, profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
3. Construcción: durante la fase de construcción el sistema es desarrollado en el ambiente de desarrollo.

4. Transición: el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

En cada iteración se repite el flujo de trabajo de las disciplinas, requisitos, análisis y diseño, implementación y pruebas internas. De esta forma se brinda un resultado más completo para un producto final de manera creciente. Para llegar a lograr este resultado, cada requisito debe tener un completo desarrollo en una única iteración. (Pressman, 2010) La metodología AUP en su variante UCI, propone cuatro escenarios para la disciplina de requisitos que ayudan a modelar el sistema en el proyecto, los cuales son:

- ✓ Escenario No 1: Proyectos que modelen el negocio con casos de uso del negocio solo pueden modelar el sistema con casos de uso del sistema.
- ✓ Escenario No 2: Proyectos que modelen el negocio con modelo conceptual solo pueden modelar el sistema con casos de uso del sistema.
- ✓ Escenario No 3: Proyectos que modelen el negocio con descripción de proceso de negocio solo pueden modelar el sistema con descripción de requisitos por proceso.
- ✓ Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con Historias de usuario. (Sánchez, 2015).

Este se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. (Tamara, 2015) El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos.

Para la realización del proyecto se escoge el escenario número cuatro de la metodología AUP-UCI debido a que se busca continuar la línea trazada en el desarrollo del Sistema Arex, buscando promover la estandarización de esta metodología en la universidad. A demás de que está metodología permite la comunicación constante y presencia del cliente en el proceso de desarrollo. Permitiendo garantizar la retroalimentación con respecto al producto final deseado.

1.3.2 Herramienta CASE (*Visual Paradigm*)

CASE es un acrónimo para (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora). Se considera una herramienta CASE a un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de software, completamente o en alguna de sus fases (tutorialspoint, 2022). La herramienta CASE se caracteriza por el uso de un lenguaje estándar común al equipo de trabajo, que facilita la comunicación entre sus integrantes, es una herramienta fácil

de instalar y actualizar, genera código para varios lenguajes de programación y exporta en formato HTML, es una tecnología libre y está disponible en varios idiomas (Peña, 2016).

Entre las principales características de la herramienta CASE se encuentran:

- ✓ Entorno de creación de diagramas para UML 2.0.
- ✓ Diseño centrado en Casos de Uso y enfocado al negocio que genera un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Ingeniería de Código.
- ✓ Disponibilidad en múltiples plataformas (tutorialspoint, 2022).

Esta herramienta permite aumentar la calidad del software, a través de la mejora en el desarrollo y mantenimiento del mismo, de igual forma potencia la reutilización del software y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería del Software.

La herramienta CASE que se utiliza en la presente investigación es el *Visual Paradigm* para UML 8.0. La cual soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue y contribuye a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo (Microsoft, 2019).

El *Visual Paradigm* propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Se caracteriza por (yoandroide, 2020):

- ✓ Disponibilidad en múltiples plataformas como (Windows y Linux).
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones, con diferentes especificaciones.

- ✓ Licencia: gratuita y comercial.
- ✓ Soporta aplicaciones Web.
- ✓ Las imágenes y reportes generados, no son de muy buena calidad.
- ✓ Varios idiomas.
- ✓ Generación de código para Java y exportación como HTML.
- ✓ Fácil de instalar y actualizar.

En otras palabras, el uso del *Visual Paradigm* permite el soporte del ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Por lo tanto, el uso de esta herramienta CASE será de vital importancia para el proceso de modelado del protocolo.

1.3.3 Lenguaje C++

Para llevar a cabo la implementación se decide escoger como lenguaje de programación: el lenguaje C++. El cual se encuentra orientado a objetos, ofrece una buena cantidad de recursos como son las formas de encapsulamiento, la sobrecarga de operadores y funciones, la herencia y el polimorfismo (Francisco_Cortijo_Bon, 2017).
Se utiliza el lenguaje C++, primeramente, porque el marco de trabajo de este proyecto trabaja con este lenguaje. Ya que este lenguaje aporta grandes beneficios que se aprecian a largo plazo como: alta calidad del software, mayor reutilización de código y mayor facilidad de adaptación (Content, 2018). También se pueden realizar aplicaciones distribuibles sin necesidad de pagar una licencia, lo cual posibilita la realización de la aplicación en su totalidad sin tener que pagar por la utilización de este lenguaje.

1.3.4 Marco de trabajo QtCreator

Para la realización de este proyecto se necesita de un entorno de desarrollo integrado (IDE, por sus siglas en inglés), el cual funciona como entorno de desarrollo de código, permitiendo su edición y compilación. En el caso de este sistema se utilizará el IDE *QtCreator* en su versión 5.12.0, el cual es un entorno multiplataforma, enfocado en aportar características que asisten a los usuarios noveles en su aprendizaje (qgis, 2022).

Entre las principales ventajas que posee este IDE, son: la disponibilidad de código fuente, la excelente documentación organizada que provee en el *Qt Assistant* y un editor para el diseño de formularios denominado *Qt Designer* (Ojeda, 2017).

Qt Creator además cuenta con:

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

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

QtCreator se utiliza fundamentalmente para desarrollar aplicaciones con interfaz gráfica, gracias al conjunto de controles independientes de la plataforma que ofrece. Aunque también es usado para crear herramientas de línea de comando o consolas de gestión para servicios. (qgis, 2022) (Ojeda, 2017).

QtCreator está disponible para sistemas tipo UNIX (Linux, BSD, UNIX, etc.) con servidor *gráfico X Windows System, Apple Mac OS X, Microsoft Windows* y sistemas Linux embebidos. Además, se puede hacer uso de la librería desde lenguajes diferentes a C++ gracias al empleo de *bindings*: *Python, Java, Ruby, Ada, Pascal, Perl, PHP, Haskell, Lua, D, .NET*, entre otros. (jmtorres, 2018).

Se escoge como IDE al QtCreator pues el marco de trabajo de este proyecto trabaja con él para continuar la línea de trabajo realizado. Además, posee un avanzado editor de código de C++ el cual fue el lenguaje utilizado para la implementación del Protocolo.

1.4 Conclusiones Parciales

Con la culminación de este capítulo se elaboró el estudio del arte de la temática abordada, obteniéndose el siguiente grupo de conclusiones:

- ✓ Se definen los conceptos esenciales para la investigación relacionados con los procesos de adquisición de datos mediante los dispositivos Nulec, permitiendo elaborar un marco teórico, el cual es empleado como base en la propuesta de solución.
- ✓ El análisis de los diferentes protocolos de comunicación en redes industriales, permitió conocer las principales características que exigen dichos protocolos, para tener en cuenta a la hora de modelar el DNP3.
- ✓ La propuesta de solución fue dirigida por la metodología de desarrollo AUP-UCI (escenario 4), Visual Paradigm como herramienta para el modelado, y para la implementación el lenguaje de programación C++, mediante el Entorno de Desarrollo Integrado, multiplataforma y de código abierto: QTCreator.

Capítulo 2: Análisis y Diseño de la propuesta solución

En el presente capítulo será realizada una descripción de las principales funcionalidades del protocolo mediante la declaración de los requisitos funcionales y no funcionales y las especificaciones de las historias de usuario. Facilitando la comprensión de las diferentes características a implementar en el software y definir la propuesta de solución para lograr su funcionamiento. Con el fin u objetivo de darle respuesta a la problemática propuesta.

2.1 Requisitos del sistema

La ingeniería de requisitos es el conjunto de actividades y tareas del proceso de desarrollo de sistemas software. Este tiene como objetivo definir, con la mejor calidad posible, las características de un sistema software que satisfaga las necesidades de negocio de clientes y usuarios Los requisitos del sistema se clasifican en requisitos funcionales y requisitos no funcionales, para el presente proyecto se realizará un análisis de los requisitos del sistema a desarrollar.

2.2.1 Requisitos funcionales

No es más que el resultado del sistema y sus componentes cuando el usuario realiza una tarea o función descrita como un conjunto de entradas, comportamientos y salidas sobre los mismos. (pmoinformatica, 2017)

Los requisitos funcionales fueron evaluados con el fin de establecer la prioridad y complejidad según el producto de trabajo de la evaluación de requisitos propuesto por la metodología AUP-UCI. Se logró obtener los requisitos a través de una entrevista realizada a un especialista del centro Vertex que cumple el rol de cliente para el sistema. Para determinar la Prioridad de los requisitos se asumieron los criterios de Importancia y Urgencia sobre los requisitos para el cliente. Mientras que para la clasificación tanto de la prioridad como de la complejidad de los requisitos se medirán los parámetros en cuanto a Alta, Media y Baja para cada requisito. Para la realización del protocolo se determinaron los siguientes requisitos funcionales:

Tabla 2 Requisitos Funcionales del sistema (Fuente: Elaboración propia)

Nombre	Descripción	Prioridad	Complejidad
RF1: Crear Variable DNP3	El protocolo debe permitir crear la variable DNP3 con sus	Alta	Media

	atributos en la clase XmlReader (clase, tipo de objeto, dirección).		
RF2: Leer Variable DNP3	El protocolo debe permitir detectar los objetos contenidos en la variable DNP3 a través de su lectura recorriendo cada uno de los de los objetos del método	Media	Media
RF3: Leer objetos analógicos	El protocolo debe permitir detectar los objetos analógicos a través de la lectura de la variable DNP3	Media	Media
RF4: Leer objetos binarios	El protocolo debe permitir detectar los objetos binarios a través de la lectura de la variable DNP3	Media	Media
RF5: Leer contadores	El protocolo debe permitir detectar los objetos contadores a través de la lectura de la variable DNP3	Media	Media
RF6: Escribir objetos analógicos	El protocolo debe permitir la escritura de objetos analógicos de entrada y salida para lograr una mayor comprensión entre la PC maestra y la unidad remota.	Alta	Alta
RF7: Escribir objetos binarios	El protocolo debe permitir la escritura de objetos binarios de entrada y salida para lograr una mayor comprensión entre	Alta	Alta

	la PC maestra y la unidad remota.		
RF8: Escribir contadores	El protocolo debe permitir la escritura de objetos contadores de entrada y salida para lograr una mayor comprensión entre la PC maestra y la unidad remota.	Alta	Alta

2.2.2 Requisitos no funcionales

Es un atributo de calidad, que especifica criterios que pueden usarse para juzgar la operación de un sistema, en cuanto a cualidades restricciones y características del software (pmoinformatica, 2017).

En si los requisitos no funcionales sirven como normas o condiciones que caracterizan como debe ser el correcto funcionamiento del sistema a implementar, garantizando su usabilidad y calidad óptima.

Requisitos no funcionales de software

- ✓ RnF 2.4.1.1 <Plataforma> <El protocolo debe ejecutarse en Linux y Windows. >.
- ✓ RnF 2.4.2.2 <Atributo de Calidad> <Debe garantizar el acceso a la información de manera confiable, cumpliendo los parámetros de uso y calidad>

Requisitos no funcionales de hardware

- ✓ RnF 2.4.2.1 <Microprocesador>. <El protocolo debe ejecutarse sobre un procesador Pentium cuatro o superior >
- ✓ RnF 2.4.2.2 <Memoria RAM>. <El protocolo debe ejecutarse con una memoria RAM mínima de 256 MB>

Restricciones de diseño e implementación

- ✓ RnF 2.4.3.1 <Lenguaje> <Se utilizará como lenguaje de programación C++>
- ✓ RnF 2.4.3.1. <IDE> <Se utilizará el *QTCreator* en su versión 5.12.0 como IDE>

2.2 Historias de usuario

Las historias de usuario son una explicación general e informal de una función de software escrita desde la perspectiva del usuario final. (Rehkopf, 2020) En las metodologías ágiles es muy factible el uso de las historias de usuario para el modelado y especificación de requisitos. Por ende, se tomará las historias de usuario para el modelado de los requisitos funcionales del sistema, según el escenario cuatro de la metodología AUP-UCI.

Propuesta de historia de usuarios:

Tabla 3 Historia de Usuario uno (Fuente: Elaboración propia)

Historia de usuario	
Número: 1	**Nombre:** Crear Variable DNP3
Programador: Luis Ernesto Clarke Samuels	**Iteración:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 semana y 3 días
Riesgo de Desarrollo: Medio	**Tiempo Real:** 1 semana y 4días
Descripción: Crear la instancia de la variable DNP3 con cada uno de sus atributos (índice de inicio y de fin, clase, tipo de objeto y dirección).	

Tabla 4 Historia de usuario dos (Fuente: Elaboración propia)

Historia de usuario	
Número: 2	**Nombre:** Lectura de la Variable DNP3
Programador: Luis Ernesto Clarke Samuels	**Iteración:** 1
Prioridad: Media	**Tiempo Estimado:** 1 semana y 3 días

Riesgo de Desarrollo: Medio	**Tiempo Real:**1 semana
Descripción: Se quiere, permitir la lectura de la instancia de la variable DNP3 para leer su valor, en un instante de tiempo y con una calidad asociada.	

Tabla 5 Historia de Usuario tres (Fuente: Elaboración propia)

Historia de usuario	
Número: 3	**Nombre:** Lectura de objetos analógicos
Programador: Luis Ernesto Clarke Samuels	**Iteración: 2**
Prioridad: Medio	**Tiempo Estimado:** 1 semana
Riesgo de Desarrollo: Medio	**Tiempo Real:** 1 semana
Descripción: Se quiere que, mediante el proceso de lectura de objetos analógicos en bloques de variable, se acceda al objeto DNP3 de tipo analógico, para obtener el valor de la variable inicialmente enviada.	

Tabla 6 Historia de Usuario cuatro (Fuente: Elaboración propia)

Historia de usuario	
Número: 4	**Nombre:** Lectura de objetos binarios
Programador: Luis Ernesto Clarke Samuels	**Iteración: 2**
Prioridad: Medio	**Tiempo Estimado:** 1 semana
Riesgo de Desarrollo: Medio	**Tiempo Real:** 1 semana

Descripción: Se quiere que, mediante el proceso de lectura de objetos binarios en bloques de variable, se acceda el objeto DNP3 de tipo analógico, para obtener el valor de la variable inicialmente enviada.

Tabla 7 Historia de Usuario cinco (Fuente: Elaboración propia)

Historia de usuario	
Número: 5	**Nombre:** Lectura de contadores
Programador: Luis Ernesto Clarke Samuels	**Iteración: 2**
Prioridad: Medio	**Tiempo Estimado:** 1 semana
Riesgo de Desarrollo: Medio	**Tiempo Real:** 1 semana
Descripción: Se quiere que, mediante el proceso de lectura de objetos analógicos en bloques de variable, se acceda al objeto DNP3 de tipo contador, para obtener el valor de la variable inicialmente enviada.	

Tabla 8 Historia de Usuario seis (Fuente: Elaboración propia)

Historia de usuario	
Número: 6	**Nombre:** Escritura de objetos analógicos
Programador: Luis Ernesto Clarke Samuels	**Iteración: 3**
Prioridad: Alta	**Tiempo Estimado:** 1 semana

Riesgo de Desarrollo: Alta	**Tiempo Real:** 1 semana
Descripción: La escritura de objetos analógicos es realizada para crear el tipo de variable con su valor y el canal por el que se realizara la transmisión además de un tiempo de espera.	

Tabla 9 Historia de Usuario siete (Fuente: Elaboración propia)

Historia de usuario	
Número: 7	**Nombre:** Escritura de objetos binarios
Programador: Luis Ernesto Clarke Samuels	**Iteración: 3**
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo de Desarrollo: Alta	**Tiempo Real:** 1 semana
Descripción: La escritura de objetos binarios es realizada para crear el tipo de variable con su valor y el canal por el que se realizara la transmisión además de un tiempo de espera.	

Tabla 10 Historia de Usuario ocho (Fuente: Elaboración propia)

Historia de usuario	
Número: 8	**Nombre:** Escritura de contadores
Programador: Luis Ernesto Clarke Samuels	**Iteración: 3**
Prioridad: Alta	**Tiempo Estimado:** 1 semana

Riesgo de Desarrollo: Alta	**Tiempo Real:** 1 semana
Descripción: La escritura de objetos contadores es realizada para crear el tipo de variable con su valor y el canal por el que se realizara la transmisión además de un tiempo de espera.	

2.3 Diseño de la propuesta de solución

Con el objetivo de cumplir con el proceso de desarrollo del protocolo se pretende realizar un análisis y diseño del protocolo en cuestión por medio de la propuesta solución. Queda claro que la implementación del protocolo de comunicación DNP3 se realiza por niveles de implementación. Con el fin de lograr regular el tráfico de información según las reglas del mismo. Separando los mensajes en varias sub-cláusulas para proporcionar un control óptimo de error y una secuencia rápidas de comunicación.

Para vencer el objetivo de brindar una solución factible a la situación problemática planteada. Primeramente, hay que dejar claro que en la capa aplicación se encuentra grandes volúmenes de información almacenada en los fragmentos. Por lo que se quiere que mediante la implementación de los métodos de creación de variables DNP3, de lectura y escritura de objetos analógicos, binarios y contadores se facilite la comunicación entre el protocolo y varios dispositivos o estaciones remotas.

Por lo cual el método ***readObject*** de la clase *ApplicationLayer se* encargará de detectar los objetos contenidos en la unidad remota. Recorriendo cada uno de los de los objetos del mismo método, para luego ser analizados a través de los cambios que se realizan al pasar por los diferentes niveles de implementación del protocolo. Donde las cabeceras de los objetos en la solicitud especifican qué datos desea el maestro y/o cuántos objetos utilizar en la respuesta.

En caso del ***writeObject*** solo se le permitirán escribir los objetos de salida para lograr una mayor comprensión entre la unidad maestra y la unidad esclava, por lo que la unidad esclava almacenará los datos especificados por los objetos de la solicitud.

Con el fin de visualizar la **creación de la variable DNP3** se implementó una interfaz capaz de generar instancias de objetos DNP3 a través de sus atributos. Para luego mostrar en esta interfaz el resultado de una sub-cláusula de diez bytes. Generada tras el proceso de envío y recepción de mensajes entre la capa de aplicación, a su paso por la pseudo-capa de transporte, finalizando en la capa de enlace del protocolo y generando dicha trama para permitir la comunicación final entre las terminales. Esta

sub-cláusula generada cumple con las especificaciones del protocolo en cuanto a su composición. Por lo que queda estructurada de la siguiente manera:

Header Frame Dnp3									
Start Byte	*Start Byte*	*Frame Size*	*Code Control*	*Destiny Address*		***Source Address***		*CRC*	
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
05	*64*	*05*	*C0*	*01*	*00*	*00*	*04*	*E9*	*21*
START		*LEN*	*CTRL*	*Destiny Address*		*Source Address*		*CRC*	

Ilustración 12 Marco de cabecera Dnp3 (Fuente: Elaboración propia).

Indicadores internos

Esta sub-cláusula describe el bloque o cabecera, de una trama de enlace de datos. Los campos de cabecera constan de dos octetos de inicio, un octeto de longitud, un octeto de control de enlace, una dirección de destino de dos octetos y una dirección de origen de dos octetos además de dos octetos de comprobación de redundancia cíclica.

- El campo *START*: tiene dos octetos de longitud. El primer octeto es un 0x05, y el segundo es un 0x64
- El campo *LEN*: tiene un octeto de longitud y especifica el número de octetos no-CRC que siguen en la cabecera y los bloques de datos. Este recuento incluye los campos *CRTL*, *DESTINY ADDRESS* y *SOURCE ADDRESS* en la cabecera y los campos datos del usuario en el cuerpo. Los campos CRC no se incluyen en el recuento. El valor mínimo de este campo es cinco, lo que indica que sólo está presente la cabecera, y el valor máximo es 255.
- El campo CRTL: tiene un octeto de longitud y contiene información sobre la dirección de la trama, el iniciador de la transacción, el control de errores y de flujo, y la función.
- El bit de dirección o *DESTINY ADDRESS*: Indica el origen físico de transmisión de la trama de enlace de datos. *DESTINY ADDRESS* = 1 indica una trama procedente de un maestro. *DESTINY ADDRESS* = 0 indica una trama procedente de una estación externa. Los dispositivos emisores deberán establecer este bit correctamente, pero los dispositivos receptores no están obligados a comprobar su estado.

- El bit de mensaje primario o *SOURCE ADDRESS:* Indica la dirección de la trama de enlace de datos con respecto a la estación iniciadora. Este bit puede tomar valor uno o cero indicando que una transacción de la capa de enlace de datos está siendo iniciada por un maestro o una estación externa, y el código de función se elige de la forma interna.
- El campo CRC: A cada bloque de una trama se le añade una comprobación de redundancia cíclica de dos octetos. Los campos *START, LENG, CTRL, DESTINY ADDRESS* y *SOURCE ADDRESS* se incluyen en el cálculo del CRC de la cabecera. La comprobación CRC de dos octetos se genera a partir del siguiente polinomio y luego se invierte antes de colocarse en el bloque para su transmisión:

$$X16 + X13 + X12 + X11 + X10 + X8 + X6 + X5 + X2 + 1$$

Con la realización del análisis y diseño del protocolo se ha logrado implementar algunas funcionalidades para permitir la correcta comunicación entre el protocolo y los futuros dispositivos que se vincularan a este. Permitiendo la reutilización de varias de las funcionalidades y características existentes en este estándar y adaptándolas para una mejor comprensión de este con futuros usuarios.

2.3.1 Patrón arquitectónico: Basado en capas

Cada estilo arquitectónico describe una categoría del sistema que contiene: un conjunto de componentes, que realiza una función requerida por el 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 (Gómez, 2008).

La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades (Ilustración 12). El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada (geeks, 2018). Las restricciones topológicas del estilo permiten que los elementos de una capa se entiendan sólo con otros elementos de la misma. Por lo que se supone que, si esta exigencia se disminuye, el estilo deja de ser puro y pierde algo de su capacidad heurística.

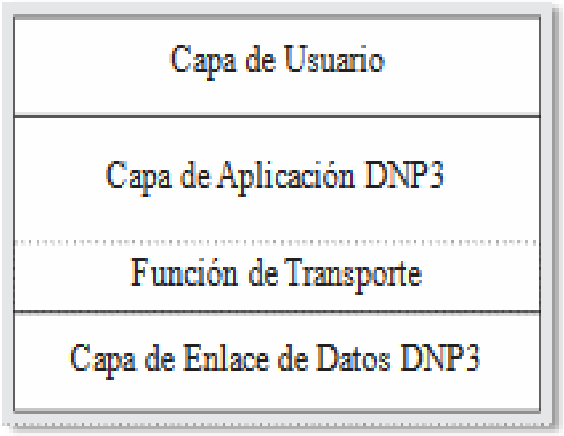


Ilustración 13 Patrón arquitectónico basado en capas (FS. cummunity, 2021)

El protocolo empleará la Arquitectura Basada en Capas. Buscando cumplir con el objetivo de mantener la compatibilidad con el sistema heredado (Ilustración 7). Estas capas solicitaran recursos a la terminal esclava (Dispositivos Nulec). Luego se proporcionarán los recursos solicitados de esta terminal hacia la terminal maestra (DNP3), logrando la comunicación entre cada una de sus capas. La última capa de enlace de datos proporciona a la capa de aplicaciones los datos necesarios para poder procesar y generar el servicio que solicito el cliente en un principio. Con el uso de esta arquitectura, se logra proporcionar al usuario final el acceso a las aplicaciones, datos y servicios, además de mejorar la seguridad del protocolo.

2.6 Diagrama de componente

Los diagramas de componentes se utilizan para visualizar la organización de los componentes del protocolo y las relaciones de dependencia entre ellos. Proporcionan una visión de alto nivel de los componentes de un sistema. (creately, 2022) Los componentes pueden ser un componente de software, como una base de datos o una interfaz de usuario; o un componente de hardware, como un circuito, un microchip o un dispositivo; o una unidad de negocio, como un proveedor, una nómina o un envío.

El diagrama de componente de la solución desarrollada revela la estructura del código y se muestra la relación entre sus componentes. Ocultando los detalles de las especificaciones, además de ayudar a comunicar y explicar las funciones del protocolo que se estarán implementando. En la ilustración 14 se evidencia como queda estructurado el cuerpo del sistema principal, para su posterior implementación.

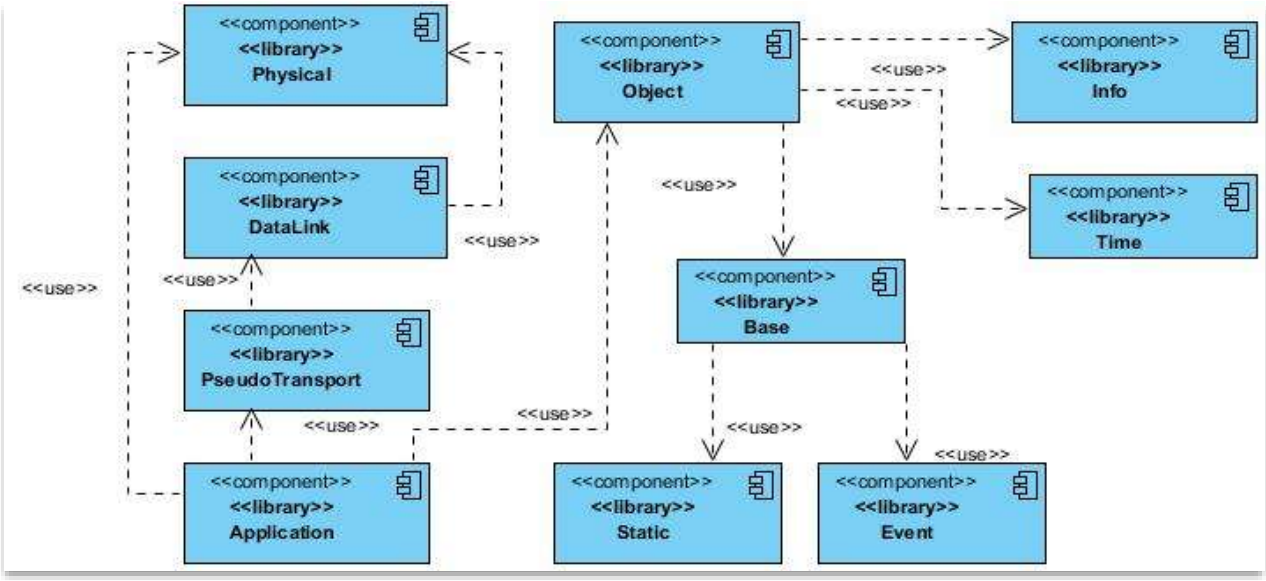


Ilustración 14 Diagrama de componentes del protocolo (Fuente: Elaboración propia)

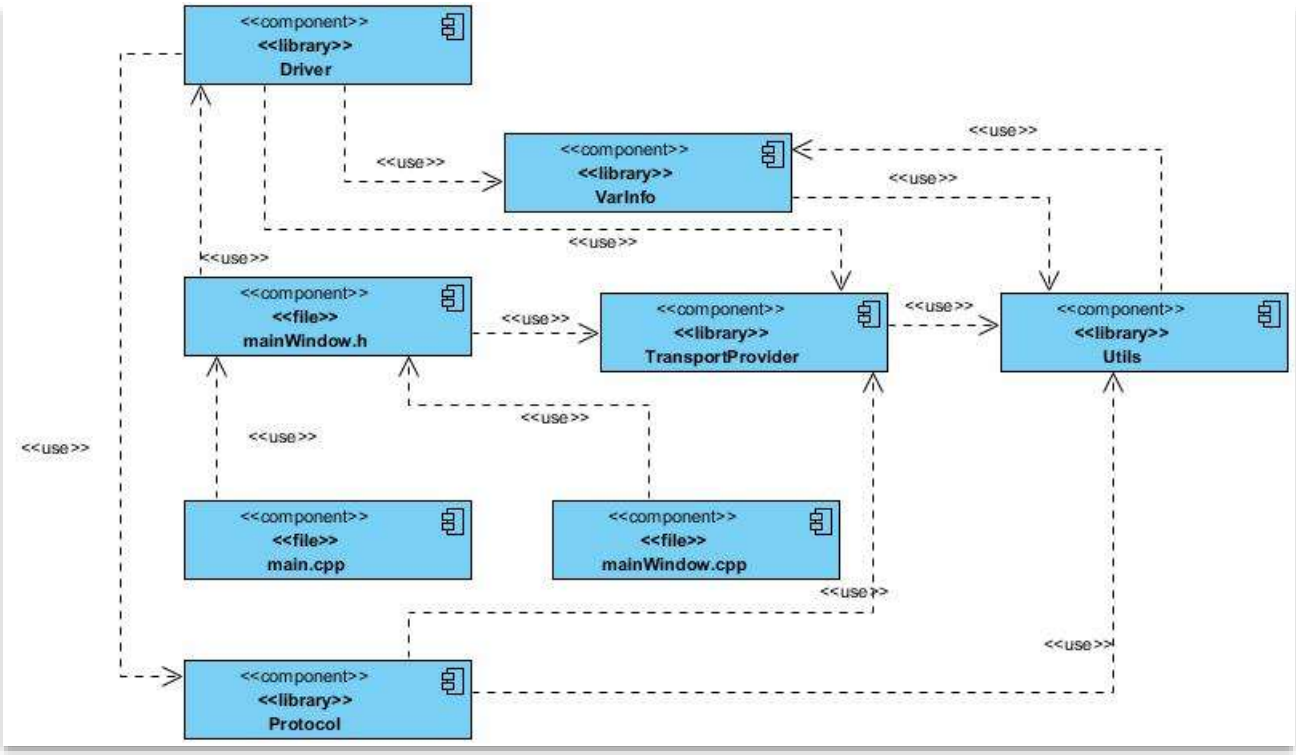


Ilustración 15 Diagrama de componente del componente Protocol (Fuente: Elaboración propia)

En la ilustración 15 se desfragmenta en los subcomponentes que está compuesto *<<library>> Protocol,* componente perteneciente al cuerpo del sistema principal. En la que se evidencia como queda fraccionado y se representa una parte modular del protocolo. Respetando la arquitectura y las reglas de protocolo DNP3. Los demás diagramas de componentes derivados del sistema principal, se encuentran en los Anexos del documento.

2.7 Diagrama de clases

Un diagrama de clases UML permite etiquetar clases y las asociaciones entre clases mediante el uso de estereotipos. El diagrama de clase UML de la propuesta solución representa las algunas de las clases del protocolo, con sus atributos y métodos, junto con las asociaciones entre dichas clases.

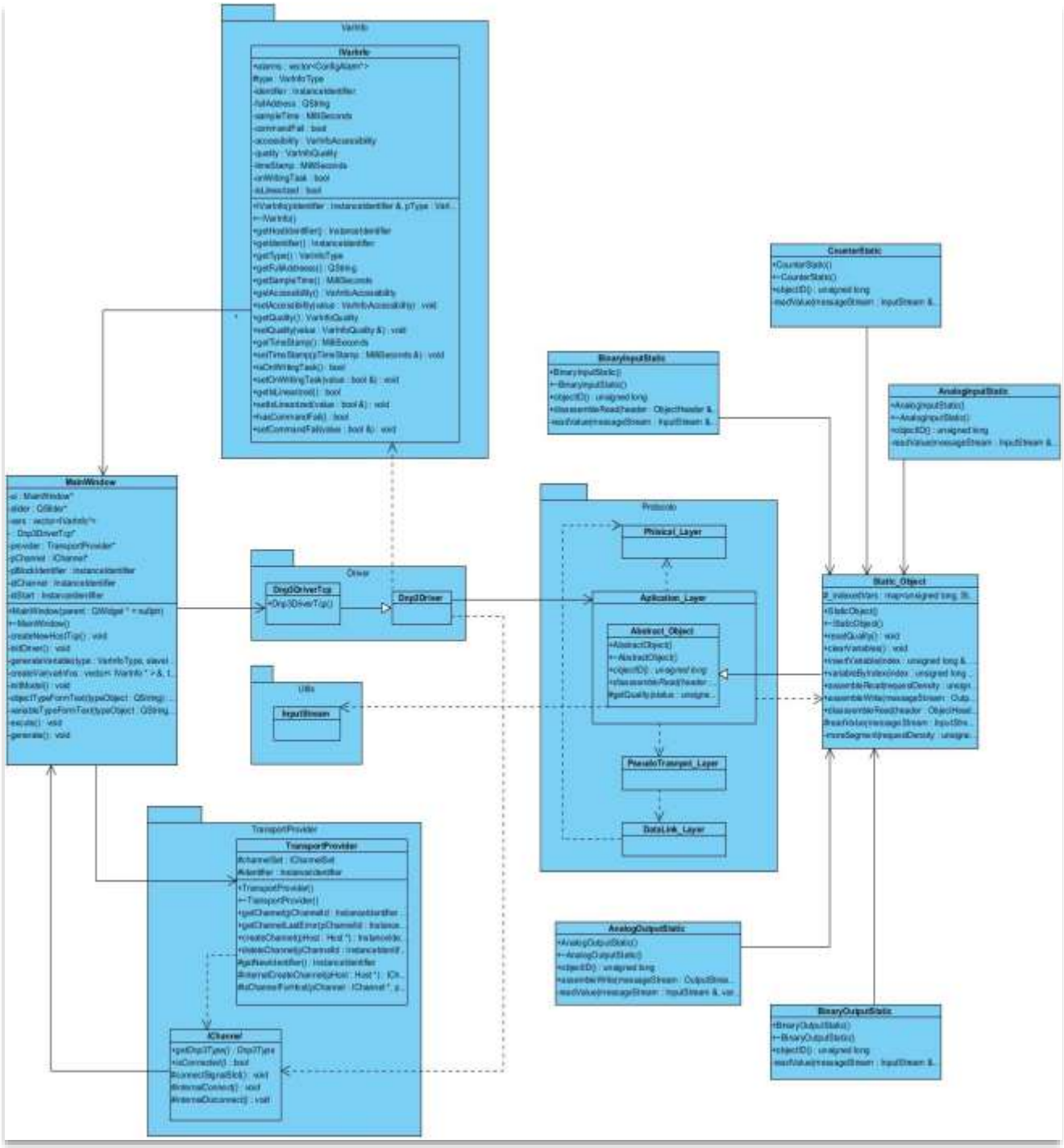


Ilustración 16 Diagrama de clases de la propuesta solución (Fuente: Elaboración propia)

En la ilustración 16 se modelan algunas de las clases existentes en el código además de los paquete o componentes a que pertenecen dichas clases, se muestra la descripción completa de estas clases reimplementadas o implementadas por completo en la propuesta solución, para el cumplimiento de los requisitos del sistema y el objetivo genera del presente trabajo de diploma.

2.8 Conclusiones Parciales

Con la culminación de este capítulo tras haber realizado el análisis y diseño del protocolo propuesto y generar los artefactos que dispone la metodología seleccionada. Se puede concluir lo siguiente:

- ✓ Con la especificación de los requisitos del sistema se identificaron los requisitos funcionales y no funcionales, con el fin de lograr definir la parte estructural del protocolo a desarrollar y las descripciones de las historias de usuario.
- ✓ Luego de establecer la arquitectura del software a partir del patrón arquitectónico, se logró la organización de los componentes del protocolo por niveles de implementación para facilitar su comprensión.
- ✓ Se realizaron los diagramas de componentes para permitir mostrar cada uno de los componentes del protocolo y las relaciones existentes entre ellos.
- ✓ Se obtuvo el modelado del protocolo, para su posterior implementación.
- ✓ La descripción de la propuesta solución a través del análisis de sus principales características, permitió identificar las funcionalidades del sistema, acorde a las necesidades del cliente.

Capítulo 3: Implementación y pruebas del sistema

El actual capítulo tiene como objetivo alcanzar la implementación del sistema a partir del diseño realizado en el capítulo anterior, la cual se debe realizar de forma iterativa e incremental, según la metodología escogida AUP-UCI. De esta forma se brinda un resultado más completo para un producto final de manera creciente en cada iteración, priorizando que cada requisito este en un completo desarrollo y pueda ser probado y mostrado al cliente. Permitiendo la retroalimentación continua entre los involucrados en el proyecto, con el fin de concretar los detalles de los requisitos y así poder implementarlos, probarlos y validarlos.

En este capítulo se describirán los estándares de codificación, los diagramas de componentes, los estilos de métodos y el diagrama de despliegue. Igualmente, se expondrán los conjuntos de pruebas realizadas al protocolo en cada versión.

3.1 Estándar de Codificación

Los estándares de codificación, también llamados estilos de programación o convenciones de código. Son parte de las llamadas buenas prácticas o mejores prácticas y se enfocan en definir la estructura y apariencia física (denominaciones, formatos, etc.) del código fuente (Merkury, 2017).Estas han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo. Con su buen uso pueden incrementar la calidad del código notablemente en ciertos lenguajes de programación. (Merkury, 2017)

La implementación del protocolo fue desarrollada utilizando el idioma inglés para los nombres de constantes, variables, estructuras, funciones, clases y métodos de clases y el idioma español para la realización de la documentación. Como estilo de codificación se utilizó *CamelCase*. En este caso para los nombres de estructuras, clases y enumeradores se utilizó la variante *UpperCamelCasecon* la primera letra de cada palabra en mayúscula. En el caso de los métodos o funciones y variables se utiliza *lowerCamelCase* siendo la primera letra minúscula y la primera letra de las palabras siguientes en mayúsculas

3.1.1 Elementos del lenguaje

Un código o programa en C++ es una secuencia de caracteres que se agrupan en componentes léxicos que comprenden el vocabulario básico del lenguaje. Estos

componentes de léxico son: (Ilustración 17) palabras reservadas, identificadores, constantes, constantes de cadena, operadores y signos de puntuación. (unizar, 2020)

```
void ApplicationLayer::clearStaticObjects()
{
    std::map<unsigned long, AbstractObject *>::iterator it;
    for (it = _objects.begin(); it != _objects.end(); ++it)
    {
        StaticObject *staticObject = dynamic_cast<StaticObject *>(it->second);
        if (staticObject)
        {
            staticObject->clearVariables();
        }
    }
    _nextState = stIdle; // volvemos al estado inicial (para hacer un integrity)
}
```

Ilustración 17 Ejemplo de código donde se muestran el uso de los componentes léxicos

3.1.2 Declaración de constantes

En C++, los identificadores de variables se pueden declarar constantes, significando que su valor se inicializa, pero no se puede modificar. Estas constantes se denominan simbólicas. Esta declaración se realiza con la palabra reservada const. (Lichtmaier, 2020)

```
const unsigned long errUnsupportedVariation = 138;
```

Ilustración 18 Declaración de una variable constante

El modificador de tipos const se utiliza en C++ también para proporcionar protección de sólo lectura para variables y parámetros de funciones. Las funciones miembros de una clase que no modifican los miembros dato a que acceden pueden ser declarados *const*. Este modificador evita también que parámetros parados por referencia sean modificados. (unizar, 2020)

```
unsigned long requestDensity() const;
```

Ilustración 19 Ejemplo de código de una función contante

3.1.3 Declaración de variables

Las declaraciones de variables se pueden situar en cualquier parte de un programa. Esta característica hace que el programador declare sus variables en la proximidad del lugar donde se utilizan las sentencias de su programa, además de que se pueden asignar valores a estas variables (unizar, 2020).

```
bool _moreAfterConfirm;
```

Ilustración 20 Declaración de una variable

```
bool _isUpdate = false;
```

Ilustración 21 Declaración de una variable con valor asignado

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.

```
for (unsigned long i(0); i < dataSize; ++i)
{
    operateData.data[i] = _messageBuffer.data[dataBegin + i];
}
```

Ilustración 22 Ejemplo de código conde se usa como índice de iteraciones la i

Su nombrado deberá dar una idea del uso que se le dará a la variable (unizar, 2020). Toda variable debería iniciarse con algún valor en su declaración.

3.1.4 Definición de clases

Las declaraciones de clases, cuentan con una llave de apertura presente una línea más abajo de la declaración y el nombre de la clase inicia con mayúscula. Las clases son comparables a los tipos primitivos tales como: int, char, double entre otras. Una clase es una descripción general del código y los datos y no contiene información. (Lichtmaier, 2020)

```
class  AnalogInputEvent: public EventsObject {...};
```

Ilustración 23 Ejemplo de código de la declaración de una clase

3.1.5 Estructuras de control

El código tendrá incluido dentro de cada estructura de control, contenido de funciones, clases, constructores, etc. Algunas sentencias condicionadas como (*if, for, while*) (Ilustración 24, 25, 26 y 27).

```
if (staticObject)
{
    staticObject->clearVariables();
}
```

Ilustración 24 Ejemplo de código de un if

```
if (!_error)
{
    _nextState = stDoWaitResponse;
}
else
{
    _nextState = stRequestCompleted;
}
```

Ilustración 25 Ejemplo de código de un if, else

Se usarán siempre las llaves en todas las estructuras de control, aunque sólo tengan una instrucción en su interior (Ilustración 24). Logrando esto evadir posibles errores.

```
while ((!messageStream.empty()) && (objectImplemented) && (!_error)) {...}
```

Ilustración 26 Ejemplo de código de un while

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, (Ilustración 25) sólo se insertará una instrucción por línea y deberá emplearse un espacio después de las comas. (Lichtmaier, 2020)

```
for (unsigned long i(0); i < dataSize; ++i)
{
    operateData.data[i] = _messageBuffer.data[dataBegin + i];
}
```

Ilustración 27 Ejemplo de código de un for

3.2 Patrones de diseño

Un patrón de diseño es un conjunto de requisitos, limitaciones y restricciones. Estas actúan como un sistema de fuerzas que influyen en la manera en la que puede interpretarse el problema y como podría aplicarse con eficacia la solución. Son una solución general, reutilizable y aplicable a diferentes problemas de diseño de software.

Se trata de plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas. Estas soluciones son de problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error. Y son consideradas buenas prácticas de programación (Canelo, 2020)

3.2.1 Patrones GRASP

GRASP es un acrónimo que significa: *General Responsibility Assignment Software Patterns* (Generales de Software para Asignar Responsabilidades). Estos son patrones para asignación de responsabilidades a objetos, expresados en formas de patrones (Pérez Mariñán). Aunque se considera que más que patrones, son una serie de “buenas prácticas" de aplicación recomendable en el diseño de software (Ciberaula.java, 2019).

A continuación, se describen los principios fundamentales del diseño de objetos. Expresados como patrones o principios de acuerdo a las soluciones que brindan dentro del contexto del sistema:

Creador: Permite asignar el responsable de la creación de una nueva instancia de alguna clase. Explica que clase es la encargada de crear objetos, en determinados escenarios de ejecución y guía la asignación de responsabilidades relacionadas con la creación de objetos (Kord, 2011).

El patrón creador se evidencia en la clase *TransportProvider*, esta clase se encarga de crear los distintos tipos de canales en tiempo de ejecución por los cuales se realizará la comunicación según las especificaciones del protocolo, comunicación que se realiza

mediante el método *createChannel* (ilustraciones 28 y 29). El cual recibe los parámetros de configuración a través de la clase *Host* que encapsula los diferentes tipos de canales.

```
class TransportProvider
{

public:
    //CLASSES
    class Host
    {
    public:
        //CONSTRUCTORS AND DESTROYERS
        static Host *Serial(const QString &pHostName,
                            const QString &pSerialPort,
                            const QSerialPort::BaudRate &pBaudRate,
                            const QSerialPort::DataBits &pDataBits,
                            const QSerialPort::StopBits &pStopBits,
                            const QSerialPort::FlowControl &pFlowControl,
                            const QSerialPort::Parity &pParity,
                            const MilliSeconds &pTimeOut);
        static Host *Tcp(const QString &pHostName,
                         const QString &pIp,
                         const std::uint16_t &pPort,
                         const MilliSeconds &pTimeOut);

        static Host *Udp(const QString &pHostName,
                         const QString &pIp,
                         const std::uint16_t &pPort,
                         const MilliSeconds &pTimeOut);

        static Host *RtuOverTcp(const QString &pHostName,
                                const QString &pIp,
```

Ilustración 28 Ejemplo de código donde se hace uso del patrón de diseño creador

```
InstanceIdentifier TransportProvider::createChannel(Host *host)
{
    if (host->getHostType() == InvalidDnp3 || host->getHostType() == Dnp3Virtual)
    {
        return 0;
    }
    qDebug () << QString("createchannel host->getHostType() : %1 , 2 is Dnp3Tcp").arg(host->getHostType());
    if (host->getHostType() == Dnp3Rtu || host->getHostType() == Dnp3RtuOverTcp)
    {
        for (IChannelSet::iterator it = channelSet.begin(); it != channelSet.end(); it++)
        {
            if (isChannelForHost(it->second, host))
            {

                return it->second->getIdentifier();
            }
        }
    }

    IChannel *newChannel = internalCreateChannel(host);
```

Ilustración 29 Ejemplo de código donde se hace uso del patrón de diseño creador

Experto: Asignar una responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir la responsabilidad. De forma general el diseño del sistema se basa en asignar a cada clase la responsabilidad que solo ella puede realizar (Kord, 2011).

El patrón experto se evidencia en la clase *ApplicationLayer* (ilustración 30) pues esta se encarga de codificar y descodificar los mensajes en la capa de aplicación según se especifica en el estándar del protocolo a través de la implementación de una *executeMachine*. Con el fin de enviar la información hacia la capa superior o inferior a ella, asignando responsabilidades a las clases implementadas dentro del *switch*.

```
void ApplicationLayer::executeMachine()
{

    ApplicationState currentState;

    do
    {
        _transitionState = false;
        currentState = _nextState;
        qDebug () << "currentState " << currentState;

#ifdef SHOW_DEBUG
        qDebug()<<"\tApplicationState:"<< getNameState(_nextState);
#endif
        switch (currentState)
        {
            case stIdle:
                stateIdle();
            break;

            case stDisableUnsolicited:
                stateDisableUnsolicited();
            break;
            case stWaitDisableUnsolicited:
                stateWaitDisableUnsolicited();
            break;
            case stOnDisableUnsolicited:
                stateOnDisableUnsolicited();
            break;

            case stWriteInternalIndication:
                stateWriteInternalIndication();
            break;
            case stWaitWriteInternalIndication:
                stateWaitWriteInternalIndication();
            break;
            case stOnWriteInternalIndication:
                stateOnWriteInternalIndication();
            break;
```

Ilustración 30 Ejemplo de código donde se hace uso del patrón de diseño experto

Bajo Acoplamiento: Es el estado ideal que siempre se intenta obtener para lograr una buena programación o un buen diseño. Cuanto menos dependiente sean las partes que constituyen un sistema informático, mejor será el resultado. (Kord, 2011) De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización disminuyendo la dependencia entre las clases.

Este patrón se evidencia en la clase *Datalinklayr* (ilustración 31) a través de la implementación de una *executeMachine* encargada de efectuar los intercambios de

mensajes codificados y descodificados entre las capas. Este intercambio es realizado según las especificaciones del DNP3. Evitando la dependencia entre las mismas, ya que esta capa realiza el proceso de descodificación de los mensajes proporcionando un servicio a las capas superiores, pero no necesita saber cuál es el contenido de los datos que transporta.

```
void DataLinkLayer::executeMachine()
{
    DataLinkState currentState;
    do
    {
        _transitionState = false;
        currentState = _nextState;

#ifdef SHOW_DEBUG
        qDebug()<<"\t\t\tDataLinkState:"<<getNameState(_nextState);
#endif
        switch (currentState)
        {
            case stIdle:
                stateIdle();
            break;
            case stDoResetLink:
                stateDoResetLink();
            break;
            case stOnResetLink:
                stateOnResetLink();
            break;
            case stDoWaitConfirm:
                stateDoWaitConfirm();
            break;
            case stOnWaitConfirm:
                stateOnWaitConfirm();
            break;
            case stLinkReady:
                stateLinkReady();
            break;
            case stDoSendData:
                stateDoSendData();
            break;
            case stOnSendData:
                stateOnSendData();
            break;
            case stDoReceiveData:
                stateDoReceiveData();
```

Ilustración 31 Ejemplo de código donde se hace uso del patrón de diseño de bajo acoplamiento

Alta Cohesión: Las clases están implementadas de forma tal que contengan las mínimas responsabilidades y colaboren con otras para llevar a cabo una tarea. Esto permitirá tener patrones más fáciles de mantener, entender y reutilizar (Kord, 2011).

La alta cohesión se evidencia en la clase *PhysicalLayerWrapper* (ilustración 32) pues en esta capa se garantiza que se relacione lo mínimo indispensable para lograr empaquetar y analizar los segmentos recibidos del nivel de transporte a través de la implementación de una *executeTransaction* que dentro de cada *case* realiza únicamente operaciones específicas según su diseño y funcionamiento, sin involucrar otras operaciones.

```
void PhysicalLayerWrapper::executeTransaction()
{

    if(lastError == errTimeOut)
    {
        operationCompleted(0,  errMsgTimeOut);
    }

    if(pendingRetries == 0){
        return;
    }

    if(!checkChannelIsConnected())
    {
        return;
    }

    if (_readOperation)
    {
        channel->read(_dataBuffer.data, _dataBuffer.size, timeOut, this);
    }
    else
    {
        channel->write(_dataBuffer.data , _dataBuffer.size ,timeOut, this , Dnp3TransactionType::WriteTransaction);
    }

}
```

Ilustración 32 Ejemplo de código donde se hace uso del patrón de diseño de alta cohesión

3.2.2 Patón GoF *Gang of Four* (Banda de los cuatro)

Los patrones de diseño son útiles para crear un diseño orientado a objetos reutilizables. Estos identifican las clases e instancias, sus roles, colaboraciones, y la distribución de responsabilidades. (Mestras, 2017) 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(Leiva, 2016).

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.

Patrón Iterator: Se utiliza cuando se necesita recorrer secuencialmente los objetos de un elemento agregado sin exponer su representación interna. Este patrón se implementa en la clase *ApplicationLayer* (ilustración 33) específicamente en el método *resetQuality* para recorrer la lista de variables que se deben enviar y recorrer el mapa de memoria que contiene los datos de las variables.

```
void ApplicationLayer::resetQuality()
{
    std::map<unsigned long, AbstractObject *>::iterator it;
    for (it = _objects.begin(); it != _objects.end(); ++it)
    {
        StaticObject *staticObject = dynamic_cast<StaticObject *>(it->second);
        if (staticObject)
        {
            staticObject->resetQuality();
        }
    }
}
```

Ilustración 33 Ejemplo de código donde se hace uso del patrón de diseño Iterator

Call-backs:

La función call-back es una función que se pasa a otra función como argumento, para luego invocarse dentro de la función externa para completar algún tipo de rutina o acción.

Este patrón se evidencia en la clase *Datalinklayer* (ilustración 34 y 35) donde las clases implementadas en *IPhysicalLayerHandler* se pasan por parámetro para luego ser llamadas y ejecutarse a medida que se cumplen algunas condiciones.

```
class IPhysicalLayerHandler    'IPhysicalLayerHandler' has no out-of-line virtual method definitior
{
public:
    /**
     * @brief Destructor por defecto.
     */
    virtual ~IPhysicalLayerHandler(){

    }


    /**
     * @see ITransportHandler::writeHandler
     */
    virtual void onSend(const unsigned long &bytesSent /**< Cantidad de bytes escritos*/,
                        const unsigned long &error/**< Posible error ocurrido.*/) = 0;

    /**
      * @see ITransportHandler::readHandler
      */
    virtual void onReceive(const unsigned long &bytesReceived /**< Cantidad de bytes recibidos.*/,
                           const unsigned long &error /**< Posible error ocurrido*/) = 0;


    /**
     * @see ITransportHandler::disconnectHandler
     */
    virtual void onDisconnect()
    {
    }
};
```

Ilustración 34 Ejemplo de código donde se hace uso del patrón de diseño Call-backs

```
void DataLinkLayer::onSend(const unsigned long & /*bytesSent*/, const unsigned long &error)
{
    _error = error;
    executeMachine();

}

void DataLinkLayer::onReceive(const unsigned long &bytesReceived, const unsigned long &error)
{
    _error = error;
    _transferBuffer.size = bytesReceived;
    executeMachine();
}

void DataLinkLayer::onDisconnect()
{
    _nextState = stIdle; // en la proxima ejecucion, se comenzara por aqui

    // quizas ocurrio algun error inesperado y es necesario cancelar la operacion pendiente
    if (_pendingOperation && _layerHandler)
    {
        if (_readOperation)
        {
            _layerHandler->onDataReceived(0, _error);
        }
        else
        {
            _layerHandler->onDataSent(_error);
        }
    }
    _pendingOperation = false;
}
```

Ilustración 35 Ejemplo de código donde se hace uso del patrón de diseño Call-backs

3.3 Pruebas

Para comprobar el estado de calidad de un producto de software se recomienda la realización del proceso de pruebas. El proceso de prueba es clave a la hora de detectar errores o fallas. Conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado, con el objetivo de calcular el grado en que el software cumple con los requerimientos.

3.3.1 Estrategia de Pruebas

Según Pressman:

“La Estrategia de Prueba de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba: la planificación, el diseño de casos de prueba, la ejecución y los resultados, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software” (Pressman, 2010).

En otras palabras, las estrategias de prueba describen el enfoque y los objetivos generales de las actividades de prueba. Define las consideraciones especiales relacionadas con los recursos, los criterios de éxitos y culminación de la prueba y las técnicas de pruebas (manual o automática) y herramientas a utilizar.

Siguiendo la metodología AUP-UCI en su cuarto escenario se realizarán pruebas unitarias y de aceptación. A continuación, se determinará la estrategia de prueba a seguir para la validación de la solución:

Tabla 11 Estrategia de prueba (Fuente: Elaboración propia)

Pruebas	Método a utilizar	Alcance de las pruebas
Unitarias	Pruebas de Caja blanca apoyado del módulo QTest en el lenguaje C++	Se le realizarán pruebas unitarias al código, separándolas por módulos para su mayor comprensión. Logrando validar el comportamiento del módulo y su lógica
Aceptación	Pruebas de caja negra a través de los casos de prueba de aceptación	Se analizarán cada uno de los requisitos funcionales para ver si cumplen con los requerimientos descritos

3.3.2 Métodos de Prueba

El principal objetivo del diseño de casos de prueba a través de los métodos de prueba es adquirir un conjunto de pruebas con la mayor probabilidad de descubrir los defectos del software. Buscando cumplir con esto, se emplean estos métodos de prueba. (Ruiz, 2010)

- **Prueba de Caja Blanca:** Se llevan a cabo sobre el código fuente del software, es decir, sobre la estructura de control del programa. Se obtienen casos de prueba que aseguren que durante la prueba se han ejecutado, por lo menos una vez y examinan el estado del programa en varios puntos. Para determinar si el estado real coincide con el esperado.
- **Prueba de Caja Negra:** Se llevan a cabo sobre la interfaz del software para validar los requisitos funcionales sin fijarse en el funcionamiento interno de un programa. Solo se fijan en las funciones que realiza el software buscando demostrar su operabilidad. Las técnicas de prueba de caja negra buscan que las entradas se acepten de forma adecuada, que se produzcan de forma correcta y que la integridad de la información externa se mantenga.

3.3.3 Pruebas Unitarias

Las pruebas unitarias pretenden probar que las unidades que forman el protocolo cumplen las especificaciones y tienen el comportamiento esperado. Cumpliendo con los objetivos de los componentes del software.

El primer nivel de las pruebas consiste en la verificación de unidades de software de forma aislada, es decir, probar el correcto funcionamiento de una unidad de código. Este tipo de prueba suelen ser realizadas por los desarrolladores, ya que es muy recomendable conocer el código fuente del programa y generalmente se realizan pruebas de caja blanca o se analiza el código para comprobar que cumple con las especificaciones del componente.

Pruebas unitarias automatizadas

Para la realización de las pruebas unitarias se empleó el Qt Test, framework para pruebas de este tipo a aplicaciones y librerías. Qt Test provee todas las funcionalidades comunes de los framework de pruebas unitarias, así como extensiones para probar interfaces gráficas de usuario (Qt Creator, 2019). Para la realización de estas pruebas se implementaron métodos de prueba, para validar las funcionalidades existentes. En la ilustración 37 se muestra un fragmento de código de estas pruebas automatizadas.

```
void on_sendFrm(SimpleBuffer buff){
    QString frame = "";

    for (int index = 0 ;static_cast<unsigned long>(index) < buff.size; index++)
    {
        frame += getNumberToHex(buff.data[index])+" ";
    }
    QString correct_frame = "5 64 5 C0 1 0 3 0 3A 48";
    QString incorrect_frame = "5 64 5 C0 1";
    QCOMPARE(correct_frame,frame.trimmed()) ;
}
void on_sendFrmInc(SimpleBuffer buff){
    QString frame = "";

    for (int index = 0 ;static_cast<unsigned long>(index) < buff.size; index++)
    {
        frame += getNumberToHex(buff.data[index])+" ";
    }
    QString correct_frame = "5 64 5 C0 1 0 3 0 3A 48";
    QString incorrect_frame = "5 64 5 C0 1";
    QCOMPARE(incorrect_frame,frame.trimmed()) ;
}
```

Ilustración 36 Código de pruebas automatizadas

En las ilustraciones 37 y 38 se muestran los resultados de las pruebas unitarias realizadas durante el proceso de implementación. En las que se evidencia la admisión de los valores de la trama a mostrar a través de pruebas unitarias automatizadas.

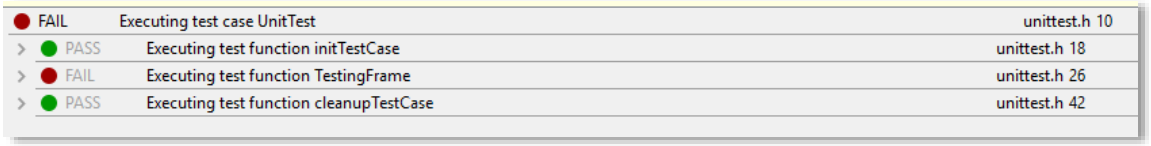


Ilustración 37 Prueba unitaria realizada

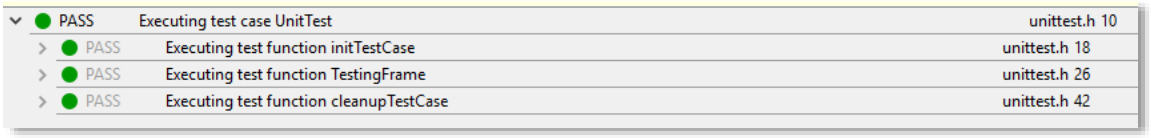


Ilustración 38 Prueba unitaria realizada

Como resultado de las pruebas unitarias realizadas a través de la herramienta QTests. Se logró analizar un total de 10 pruebas automatizadas al código (ilustración 39)

```
Totals: 10 passed, 0 failed, 0 skipped, 0 blacklisted, 97ms
********* Finished testing of UnitTest *********
```

Ilustración 39 Resultado de las pruebas unitarias

De los mismos se obtuvo un error relacionado con el método createChannel en la segunda iteración pues no se visualizaba el mensaje en la interfaz. Luego de detectar dicho error, este fue corregido.

3.3.4 Pruebas de Aceptación

La utilización de un producto de software debe de estar justificado por las ventajas que ofrece. Para esto, antes de su utilización se tiene que determinar el porciento de aceptación que justifican su uso. Una de las mejores formas de evaluar dicho software es a través de las pruebas de aceptación. (Nadia Cavalleri, 2021) Estas pruebas al protocolo se realizan en un entorno de desarrollo real y del usuario final. En otras palabras, es la realización de una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Con el fin de permitir 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 protocolo definitivamente. Para la preparación, ejecución y evaluación de estas pruebas no se requiere de conocimientos informáticos.

Casos de Prueba de Aceptación

Los casos de pruebas de aceptación se derivan de los criterios de aceptación de las historias de usuario. Por lo general, estos son los escenarios que se describen al detalle sobre lo que tiene que hacer el producto en diferentes condiciones. Se realizaron ocho

casos de pruebas en correspondencia con cada historia de usuario, para las pruebas de aceptación.

Tabla 12 Caso de prueba de aceptación uno (Fuente: Elaboración propia)

Casos de prueba de Aceptación	
Código: P1HU1	**Numero de historia de usuario:** 1
Historia de usuario: Crear Variable DNP3	
Condiciones de ejecución: Crear la instancia de la variable DNP3 con sus atributos en la clase XmlReader con cada uno de sus atributos (clase, tipo de objeto, dirección).	
Entrada/Pasos para la ejecución: Al introducir los campos el protocolo tiene que rellenar los campos (clase, tipo de objeto, dirección).	
Resultado esperado: Se deberá mostrar la variable con los campos registrados.	
Evaluación de la prueba: Resultado satisfactorio	

Tabla 13 Caso de prueba de aceptación dos (Fuente: Elaboración propia)

Casos de prueba de Aceptación	
Código: P2HU2	**Numero de historia de usuario: 2**
Historia de usuario: Lectura de la Variable DNP3	
Condiciones de ejecución: Se quiere que, mediante el proceso de lectura de la variable DNP3 en bloques de variable, se lea su valor, con un instante de tiempo y con una calidad asociada.	
Entrada/Pasos para la ejecución: 1. Se creó una instancia de la variable DNP3. 2. A través de la instancia anterior se debe obtener la variable inicialmente creada. 3. Se debe leer el valor con un instante de tiempo y calidad asociada	
Resultado esperado: Se lee el valor variable junto con el instante de tiempo y calidad asociada a él.	
Evaluación de la prueba: Resultado no satisfactorio	

Tabla 14 Caso de prueba de aceptación tres (Fuente: Elaboración propia)

Casos de prueba de Aceptación	
Código: P3HU3	**Numero de historia de usuario: 3**
Historia de usuario: Lectura de objetos analógicos	
Condiciones de ejecución: Se quiere que, mediante el proceso de lectura de objetos analógicos en bloques de variable, se lea su valor, en un instante de tiempo y con una calidad asociada.	
Entrada/Pasos para la ejecución: 1. Se creó una instancia del objeto DNP3 de tipo analógico 2. A través de la instancia anterior se debe obtener la variable inicialmente creada 3. Se debe leer el valor con un instante de tiempo y calidad asociada	
Resultado esperado: Se lee el valor del objeto analógico junto con el instante de tiempo y calidad asociada a él	
Evaluación de la prueba: Resultado no satisfactorio.	

Tabla 15 Caso de prueba de aceptación cuatro (Fuente: Elaboración propia)

Casos de prueba de Aceptación	
Código: P4HU4	**Numero de historia de usuario:** 4
Historia de usuario: Lectura de objetos binarios	
Condiciones de ejecución: Se quiere que, mediante el proceso de lectura de objetos binarios en bloques de variable, se lea su valor, en un instante de tiempo y con una calidad asociada.	
Entrada/Pasos para la ejecución: 1. Se creó una instancia del objeto DNP3 de tipo binarios 2. A través de la instancia anterior se debe obtener la variable inicialmente creada 3. Se debe leer el valor con un instante de tiempo y calidad asociada	
Resultado esperado: Se lee el valor del objeto binarios junto con el instante de tiempo y calidad asociada a él	
Evaluación de la prueba: Resultado no satisfactorio.	

Tabla 16 Caso de prueba de aceptación cinco (Fuente: Elaboración propia)

Casos de prueba de Aceptación	
Código: P5HU5	**Numero de historia de usuario:** 5
Historia de usuario: Lectura de contadores	
Condiciones de ejecución: Se quiere que, mediante el proceso de lectura de objetos contadores en bloques de variable, se lea su valor, en un instante de tiempo y con una calidad asociada.	
Entrada/Pasos para la ejecución: 1. Se creó una instancia del objeto DNP3 de tipo contadores 2. A través de la instancia anterior se debe obtener la variable inicialmente creada 3. Se debe leer el valor con un instante de tiempo y calidad asociada	
Resultado esperado: Se lee el valor del objeto contadores junto con el instante de tiempo y calidad asociada a él	
Evaluación de la prueba: Resultado no satisfactorio.	

Tabla 17 Caso de prueba de aceptación seis (Fuente: Elaboración propia)

Casos de prueba de Aceptación	
Código: P6HU6	**Numero de historia de usuario: 6**
Historia de usuario: Escritura de objetos analógicos	
Condiciones de ejecución: Se quiere que el protocolo sea capaz de lograr escribir objetos de tipo analógicos, obteniendo el tipo de variable con su valor, canal por el que se realizará la trasmisión y un tiempo de espera.	
Entrada/Pasos para la ejecución: 1. Se debe realizar la escritura de objetos analógicos a través de la clase WriteTask del módulo de recolección 2. Se debe obtener el tipo de variable con su valor y el canal por el que se realizara la transmisión 3. Debe de mostrarse un tiempo de espera de ejecución.	
Resultado esperado:	

Se tiene que lograr la escritura de los objetos analógicos y mostrarse su valor, su canal de transmisión y el tiempo de espera de ejecución
Evaluación de la prueba: Resultado satisfactorio

Tabla 18 Caso de prueba de aceptación siete (Fuente: Elaboración propia)

Casos de prueba de Aceptación	
Código: P7HU7	**Numero de historia de usuario:** 7
Historia de usuario: Escritura de objetos binarios	
Condiciones de ejecución: Se quiere que el protocolo sea capaz de lograr escribir objetos de tipo binario, obteniendo el tipo de variable con su valor, canal por el que se realizará la trasmisión y un tiempo de espera.	
Entrada/Pasos para la ejecución: 1. Se debe realizar la escritura de objetos binarios a través de la clase WriteTask del módulo de recolección 2. Se debe obtener el tipo de variable con su valor y el canal por el que se realizara la transmisión 3. Debe de mostrarse un tiempo de espera de ejecución.	
Resultado esperado: Se tiene que lograr la escritura de los objetos binario y mostrarse su valor, su canal de transmisión y el tiempo de espera de ejecución	
Evaluación de la prueba: Resultado satisfactorio	

Tabla 19 Caso de prueba de aceptación ocho (Fuente: Elaboración propia)

Casos de prueba de Aceptación	
Código: P8HU8	**Numero de historia de usuario:** 8
Historia de usuario: Escritura de contadores	
Condiciones de ejecución: Se quiere que el protocolo sea capaz de lograr escribir objetos de tipo contador, obteniendo el tipo de variable con su valor, canal por el que se realizará la trasmisión y un tiempo de espera.	
Entrada/Pasos para la ejecución:	

1. Se debe realizar la escritura de objetos contadores a través de la clase WriteTask del módulo de recolección 2. Se debe obtener el tipo de variable con su valor y el canal por el que se realizara la transmisión 3. Debe de mostrarse un tiempo de espera de ejecución.
Resultado esperado: Se tiene que lograr la escritura de los objetos contador y mostrarse su valor, su canal de transmisión y el tiempo de espera de ejecución
Evaluación de la prueba: Resultado satisfactorio

No conformidades detectadas en los casos de pruebas

Luego del previo análisis de los casos de prueba de aceptación, se detectaron cuatro No Conformidades (NC) en tres iteraciones realizadas, para un total de ocho casos de prueba de aceptación. A continuación, se muestran los resultados de las pruebas funcionales realizadas al protocolo:

Tabla 20 Resultados de las pruebas (Fuente: Elaboración propia)

Código	Iteración donde se encontró la NC	Requisito Funcional	Tipo de NC	Descripción	Estado
P2HU2	1	Lectura de la Variable DNP3	Funcional	No se muestra el valor de la variable con el instante de tiempo y su calidad asociada a él, en la interfaz	Resuelto
P3HU3	2	Lectura de objetos analógicos	Funcional	No se muestra el valor de la variable con el instante de tiempo y su calidad asociada a él, en la interfaz	Resuelto
P4HU4	2	Lectura de objetos binarios	Funcional	No se muestra el valor de la variable con el	Resuelto

				instante de tiempo y su calidad asociada a él, en la interfaz	
P5HU5	2	Lectura de contadores	Funcional	No se muestra el valor de la variable con el instante de tiempo y su calidad asociada a él, en la interfaz.	Resuelto

Ya realizado el análisis de las no conformidades detectadas en las pruebas de aceptación, y dándole solución a cada una de ellas. Se puede afirmar que el protocolo cumple con las especificaciones de los requisitos funcionales y que la calidad adecuada. A continuación, mostramos los resultados obtenidos en los casos de prueba de aceptación y las no conformidades detectadas, esto se evidencia en la gráfica (ilustración 40):

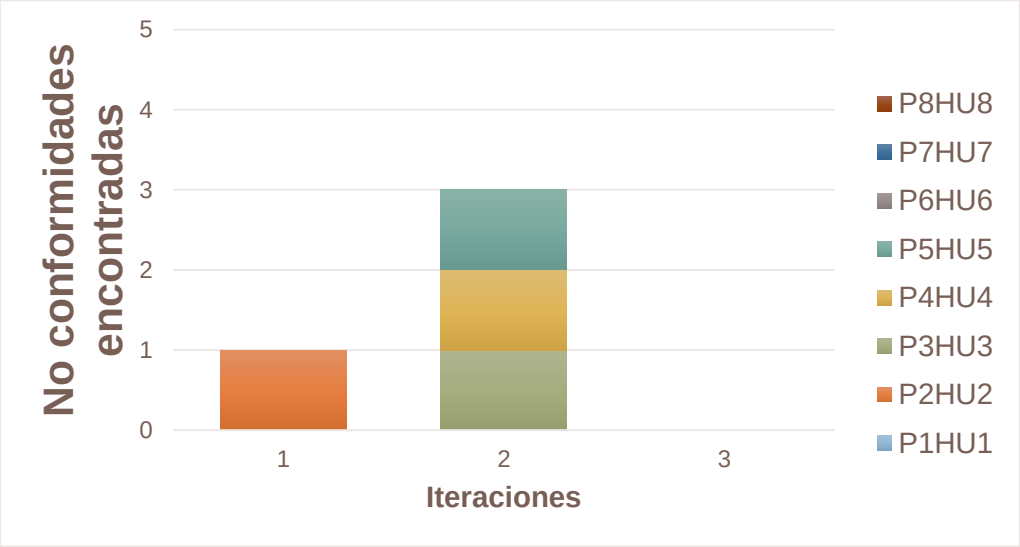


Ilustración 40 No conformidades encontradas en los casos de pruebas (Fuente: Elaboración propia)

3.3.5 Resultado de las pruebas

Como resultado de las pruebas unitarias se encontró una no conformidad, relacionada con un error funcional y en las pruebas de aceptación se obtuvieron un total de cuatro no conformidades, relacionadas con errores de validación y funcionalidad. Estas no conformidades se corrigieron en el proceso de pruebas realizado. Dándole un correcto cumplimiento a los requisitos funcionales del protocolo. Estos resultados se pueden observar en la ilustración 41.

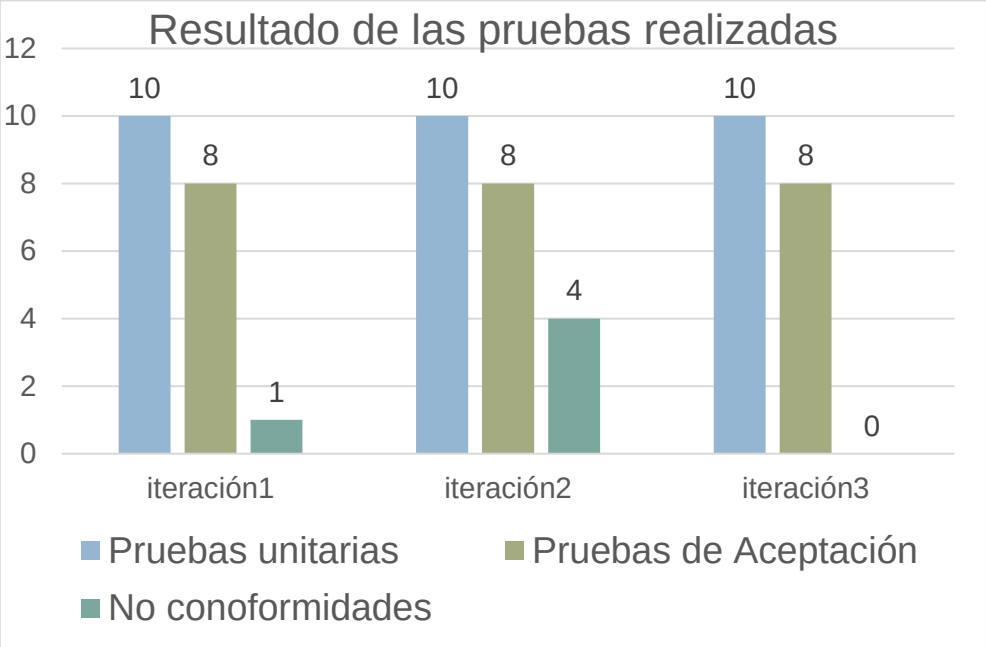


Ilustración 41 Resultados de las pruebas realizadas (Fuente: Elaboración propia)

3.4 Conclusiones Parciales

Durante el desarrollo de este capítulo se exhibieron los principales artefactos del modelo de implementación, llegando a las siguientes conclusiones:

- ✓ La implementación del protocolo DNP3, se realizó de forma eficiente, agilizando los procesos de ejecución en este.
- ✓ Se desarrolló el diseño de la solución mediante el uso de los patrones de diseño para agilizar el proceso de implementación.
- ✓ Se determinó a través de las pruebas realizadas que la solución desarrollada cumple con los objetivos generales de la presente investigación, reuniendo los requisitos necesarios para establecer una correcta implementación del protocolo DNP3.

Conclusiones Generales

Con el desarrollo del proyecto se procuró abarcar una panorámica sobre la fundamentación de herramientas que cumplan con los requisitos para crear un prototipo de software de fácil uso para los usuarios. También como resultado de dicho trabajo:

1. El empleo de las herramientas y tecnologías seleccionadas para la implementación de la solución, propició la correspondencia entre los resultados obtenidos y los esperados, lo cual aseguró el nivel de precisión en el análisis, diseño e implementación del protocolo DNP3.
2. La especificación de los requisitos, así como el análisis y diseño del protocolo DNP3, permitió definir los elementos necesarios de la implementación del mismo.
3. Se implementó un protocolo eficiente que permitió el control a distancia de los dispositivos Reconectadores Automáticos Nulec, con el fin de garantizar: la protección, monitoreo, medición, control, comunicación y calidad de energía eléctrica en determinados circuitos.
4. La realización de pruebas de software permitió comprobar el correcto funcionamiento del protocolo. Estas pruebas arrojaron resultados satisfactorios en relación al código y el conjunto de interfaces implementadas.

Recomendaciones

Se recomienda lograr integrar el protocolo DNP3 al sistema Arex a través de la implementación de un Driver.

Referencias Bibliográficas

1Library. 2022. *Reconectores Nulec.* [En línea] 2022. https://1library.co/article/reconectadores-nu-lec-cargas-circuitos-distribuci%C3%B3n-instrumentos.q5m15ejy.

Arcias, Marino A. Godoy. 2015. 1library. [En línea] 2015. https://1library.co/article/recerradores-nu-lec-caracterizaci%C3%B3n-los-dispositivos-empleados-automatizaci%C3%B3n.qmj0n28q. Automatización de la red de 13.8 kV en la ciudadSanta Clara.

Canelo, Mirian Martinez. 2020. profile. [En línea] 2020. https://profile.es/blog/patrones-de-diseno-de-software/.

Centro de Informática Industrial, CEDIN. 2019. *Manual de Usuario Sistema de Medición y Adquisición AREX 3.0.* La Habana, Cuba : s.n., 2019.

Centro de informática, industrial. 2019. *Manual de usuario .* La Habana : Universidad de las Ciencias Informáticas, 2019.

Ciberaula.java. 2019. Patrones de Diseño en aplicaciones Web. [En línea] 24 de abril de 2019. http://java.ciberaula.com/articulo/diseno_patrones_j2ee.

Clemente Pedrico, O. : . (2017). *Implementación del protocolo DNP3 en una red de sistemas SCADA.* Zaragoza. Escuela de Ingenieros y Arquitectos : s.n., (2017).

Content, Rock. 2018. rockcontent. [En línea] 27 de sep de 2018. https://rockcontent.com/es/blog/tipos-de-lenguaje-de-programacion/.

creately. 2022. [En línea] 11 de Mayo de 2022. https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-componentes/.

Crespo Guerra, L. L. (Agosto de 2017). 2021. CIGET. [En línea] Noviembre de 2021. http://www.ciget.pinar.cu/ojs/index.php/publicaciones/article/view/281/1090 .

Day, T. 2022. *DNP3, Distributed Network Protocol v3 an Introduction.* Londres : Eaton, 2022.

—. 2017. *DNP3, Distributed Network Protocol v3 an Introduction.* Londres : Eaton, 2017.

Eaton. 2017. DNP Spesification Version. [En línea] February de 2017.

2018. ENSOTEST. *ENSOTEST.* [En línea] 2018. https://www.ensotest.com/es/dnp3/introduccion-a-la-norma-ieee-1815-dnp3/.

Equipo técnico Copywriter. 2022. kryptonsolid. [En línea] 2022. https://kryptonsolid.com/uso-de-la-plataforma-qtest-para-emplear-herramientas-mejoradas-de-gestion-de-casos-de-prueba/.

Fernandez, Nerea Garcia. 2020. openwebinars. [En línea] 19 de octubre de 2020. https://openwebinars.net/blog/que-es-el-modelo-osi/.

Francisco_Cortijo_Bon. 2017. ugr. [En línea] 2017. https://elvex.ugr.es/decsai/builder/intro/5.html.

freyrscada. 2017. DNP3 Protocol. *DNP3 Protocol.* [En línea] 2017. https://www.freyrscada.com/dnp3-ieee-1815.php.

FS. cummunity. 2021. [En línea] 6 de 5 de 2021. https://community.fs.com/es/blog/tcpip-vs-osi-whats-the-difference-between-the-two-models.html.

geeks. 2018. [En línea] mayo de 2018. https://geeks.ms/jkpelaez/2018/05/30/arquitectura-basada-en-capas/.

Gómez, Enrique Chaviano. 2008. gestiopolis. [En línea] 2008. https://www.gestiopolis.com/arquitectura-software-modulo-inventario-erp-cubano/.

IEC, DNP3 (Distributed Network Protocol) e. 2006. copadata. *copadata.* [En línea] 2006. https://www.copadata.com/es/industrias/energia-infraestructura/energy-insights/dnp3-protocolo-de-red-distribuida/.

IEEE. 2011. *IEEE Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3). 1815* USA New York, NY, 4 de junio de 2011. Protocolo.

Jithin. 2019. Interserver. [En línea] 2019. Jithin. Interserver. (2019). Protocolos de red comunes https://www.interserver.net/tips/kb/common-network-protocols-ports/ consultado agosto, 2019.

jmtorres. 2018. Medium. [En línea] 2018. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjMhP-xufz3AhXmQjABHeTpD4wQFnoECAYQAQ&url=https%3A%2F%2Fmedium.com%2Fjmtorres%2Fproyecto-qt-framework-de-desarrollo-de-aplicaciones-2b2f895ac285&usg=AOvVaw2-Q4DuXl9uS0.

Kord, Maria. 2011. tuxnots. [En línea] 31 de diciembre de 2011. https://sites.google.com/site/tuxnots/materias-de-la-facu/metodologia-de-sistemas/patronesgrasppatronesgofdiferenciaentregraspygof.

Leiva, Antonio. 2016. devexperto. [En línea] 2016. https://devexperto.com/patrones-de-diseno-software/.

Lichtmaier, Nicolás. 2020. reloco. [En línea] 2020. http://www.reloco.com.ar/prog/normas.html.

Margaret., Rouse. 2019. TechTarget. (s/a). Protocolos de red. [En línea] 2019. https://searchnetworking.techtarget.com/definition/protocol consultado agosto, 2019..

Merkury. 2017 . ohmyroot. [En línea] 12 de enero de 2017 . https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/.

Mestras, Juan Pavón. 2018. fdi. [En línea] 2018. https://www.fdi.ucm.es/profesor/jpavon/poo/2018pdoo.pdf.

—. 2017. fdi. [En línea] 2017. https://www.fdi.ucm.es/profesor/jpavon/poo/2.14pdoo.pdf.

Microsoft. 2019. Microsoft. [En línea] 2019. https://www.microsoft.com/es-ww/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling#:~:text=El%20Lenguaje%20Unificado%20de%20Modelado,de%20un%20sistema%20o%20proceso..

Montoya, Nancy Piedad Molina. 2020. Ciencia y Tecnología para la Salud VIsual y Ocular. [En línea] 2020. https://ciencia.lasalle.edu.co/svo/vol3/iss5/10/.

Nadia Cavalleri. 2021. youtube. [En línea] 2021. https://www.youtube.com/watch?v=WhVDCTY6VbQ.

network solutions. 2017. DNP Spesification Version. [Online] February 2017.

Ojeda, Y Sanchez. 2017. repositorio. [En línea] 2017. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiRiP_cuPz3AhW1TjABHbgaAEIQFnoECAUQAQ&url=https%3A%2F%2Frepositorio.uci.cu%2Fjspui%2Fbitstream%2F123456789%2F9893%2F1%2FTD_08926_17.pdf&usg=AOvVaw1vpnkpXFcNAPOAgym1CUZ.

Ortiz, Diego Donizetti Pérez. 2011. *ESPECIFICACIÓN DEL PROTOCOLO DNP3 UTILIZANDO UN LENGUAJE DE DESCRIPCIÓN FORMAL".* México : OAXACA, 2011. Tesis.

Peña, D Mendoza. 2016. repositorio. [En línea] 2016. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjxk9iBt_z3AhXpdDABHW4nD8gQFnoECAIQAQ&url=https%3A%2F%2Frepositorio.uci.cu%2Fjspui%2Fbitstream%2F123456789%2F7944%2F1%2FTD_08726_16.pdf&usg=AOvVaw18tgrRqKncI58UK9qOfYW.

Pérez Mariñán, P. *"Patrones de Diseño (Design Patterns)".*

pmoinformatica. 2017. [En línea] 6 de febrero de 2017. https://pmoinformatica.com/2017/02/requerimentos-funcionales-ejemplos.html?m=1.

Pressman, Roger. 2010. *Ingeniería del software: un enfoque práctico.* 7ma Edición. 2010. págs. Páginas 25-52. Vol. I.

Punzenberger, Ing. 2019. copadata. [Online] 2019. https://www.copadata.com/es/industrias/energia-infraestructura/energy-insights/dnp3-protocolo-de-red-distribuida/.

—. copadata. [En línea] https://www.copadata.com/es/industrias/energia-infraestructura/energy-insights/dnp3-protocolo-de-red-distribuida/.

qgis. 2022. [En línea] 19 de abril de 2022. https://docs.qgis.org/3.22/es/docs/developers_guide/qtcreator.html.

Qt Creator. 2019. QTEST. [En línea] 2019. http://doc.qt.io/qt-5/qttest-index.html(vid. pág. 27)..

Queens University Belfast. 2010. *IEEE Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3).* New York : Queens University Belfast, 2010.

Rehkopf, Max. 2020. Atlassian. [En línea] 2020. https://www.atlassian.com/es/agile/project-management/user-stories.

Ruiz, R. 2010. *Las Pruebas de Software y su Importancia en las Organizaciones .* s.l. : TENORIO, 2010. (vid. págs. 24,26).

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI. L a Habana, : s.n., .* 2015.

Sommerville, Ian. 2011. *Ingenieria de Software.* 9na edicion. s.l. : Addison Wesley., 2011. págs. Pág. 27-56. Vol. Parte I.

Tamara, R. S. 2015. *Metodología de Desarrollo de Software Variación de AUP para la UCI.* [En línea] 6 de marzo de 2015. https://www.ecured.cu/Metodolog%C3%ADa_de_Desarrollo_de_Software_Variaci%C3%B3n_de_AUP_para_la_UCI.

tutorialspoint. 2022. tutorialspoint. [En línea] 2022. https://www.tutorialspoint.com/es/software_engineering/case_tools_overview.htm.

universidad de la Salle, Bogotá. 2017. dialnet. [En línea] 2017. https://dialnet.unirioja.es/servlet/articulo?codigo=5599263.

unizar. 2020. [En línea] 5 de octubre de 2020. http://webdiis.unizar.es/asignaturas/PROG1/doc/materiales/sintaxis_cpp.pdf.

yoandroide. 2020. yoandroide. [En línea] 2020. https://yoandroide.xyz/que-es-y-como-usar-visual-paradigm/.

Anexos

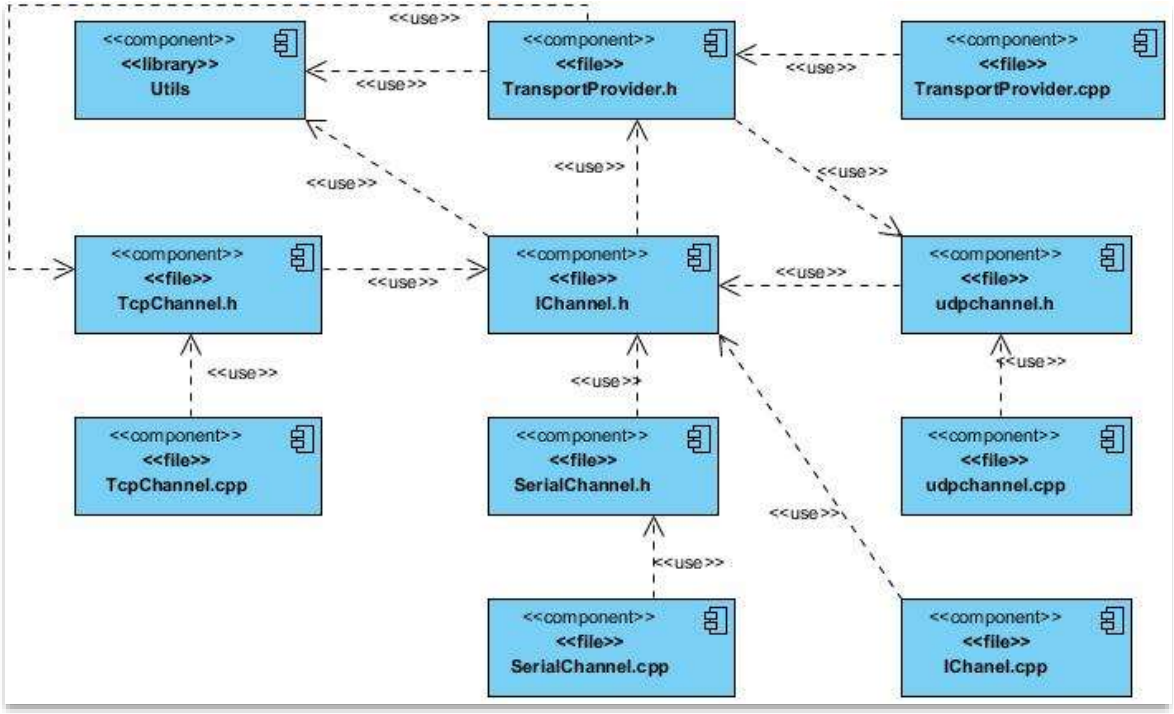


Ilustración 42 Diagrama de componente del componente TransporProvider del protocolo (Fuente: Elaboración propia)

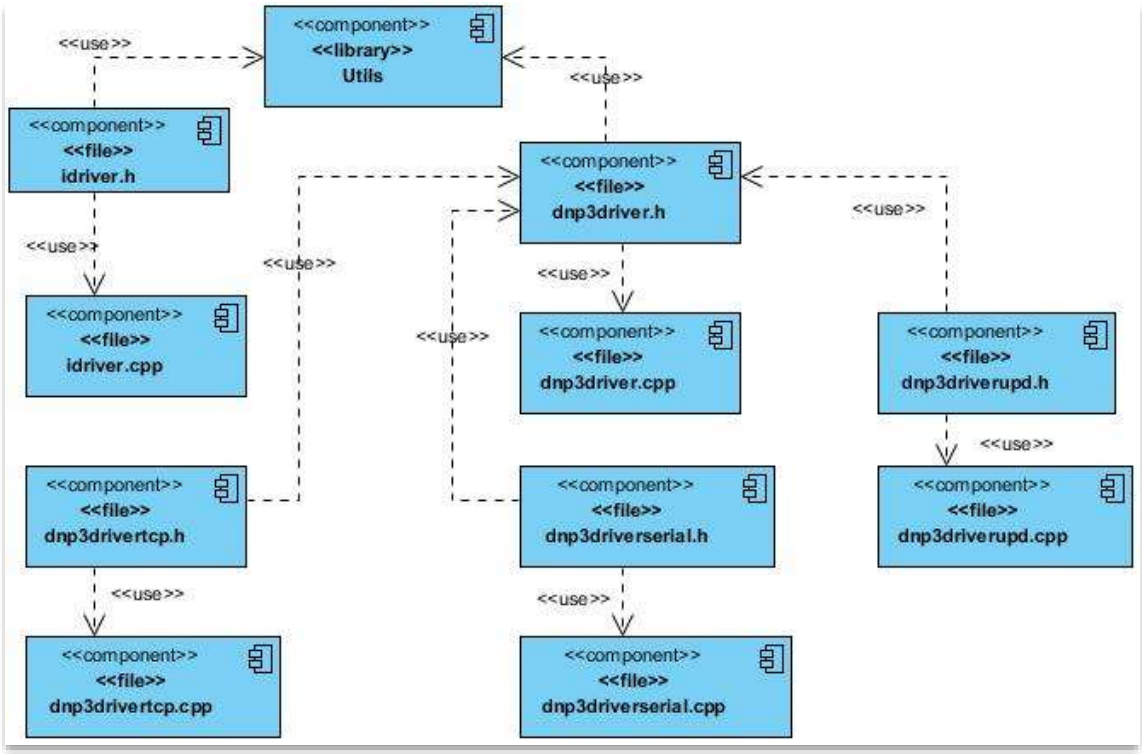


Ilustración 43 Diagrama de componente del componente Driver del sistema (Fuente: Elaboración propia)

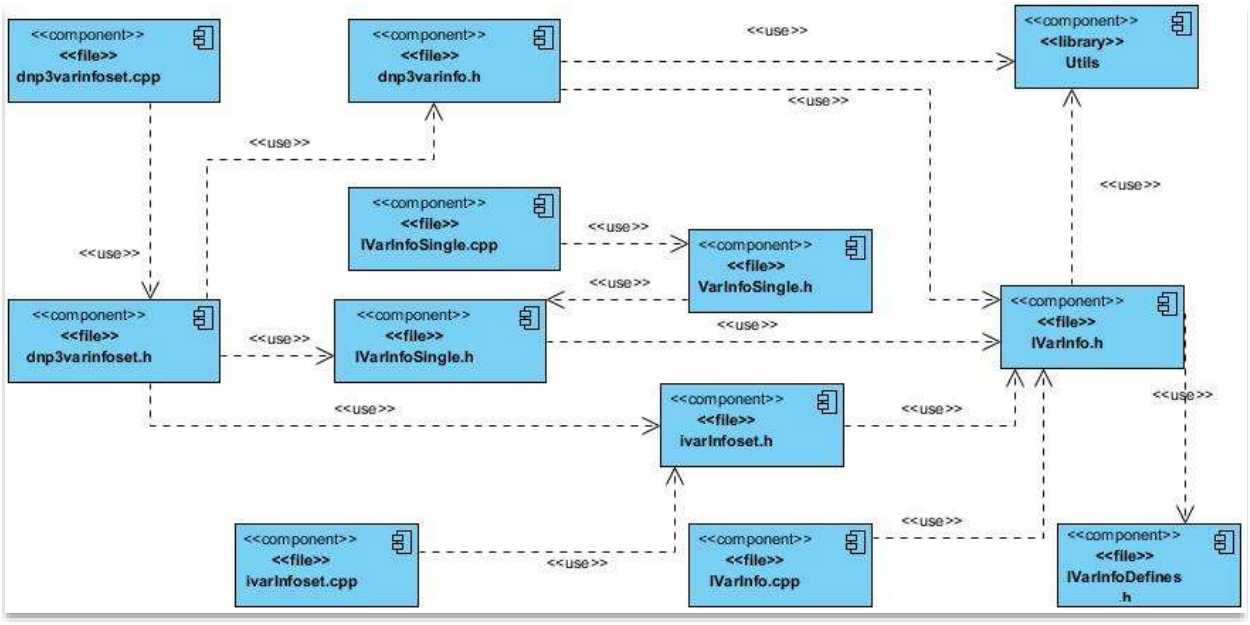


Ilustración 44 Diagrama de componente del componente VarInfo del protocolo (Fuente: Elaboración propia)

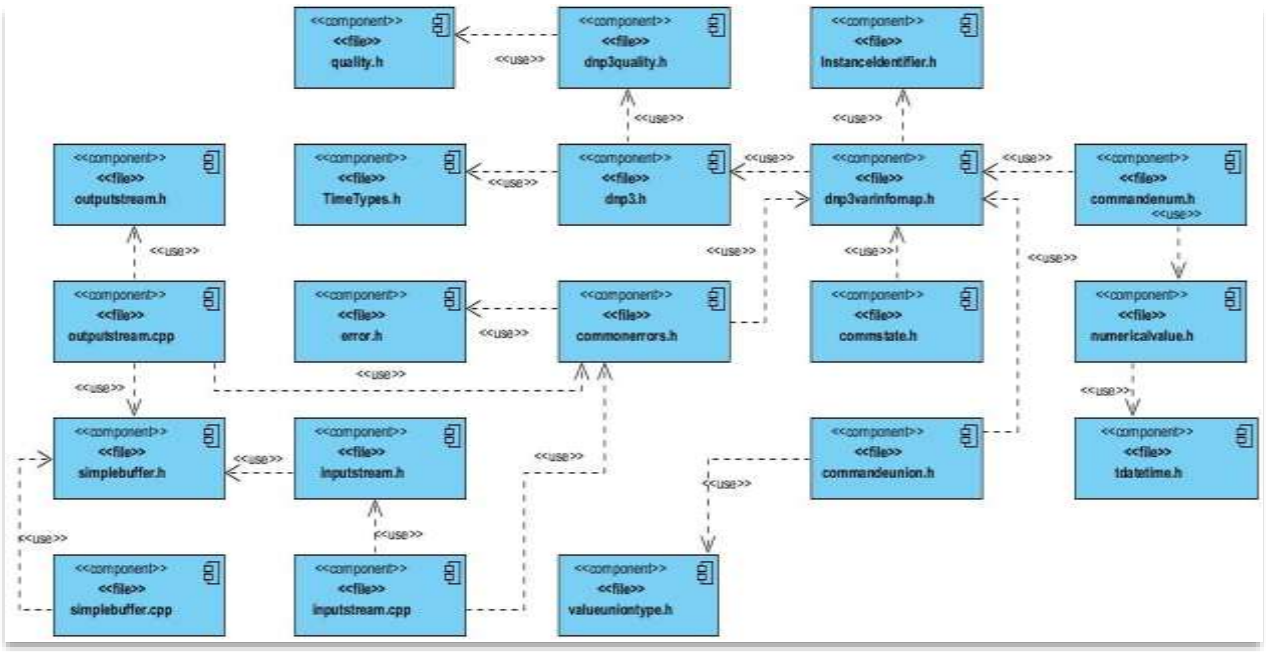


Ilustración 45 Diagrama de componente del componente Utils del protocolo (Fuente: Elaboración propia)