# Universidad de las Ciencias Informáticas

# Facultad 5

# Título: Módulos de adquisición, análisis y compresión de datos para el SCADA “Guardián del ALBA”.

**Memoria Individual para optar por el título de**

**Ingeniero en Ciencias Informáticas**

**Autor:** Yosep Yasmany Pérez Pérez

**Tutor:** Ing. Antonio Cedeño Pozo

**Co-Tutor:** Ing. Luis Enrique García

**Consultante:** Dr. Rafael Arturo Trujillo Codorniu

**Cuidad de la Habana, noviembre de 2010**

**“Año del 52 Aniversario del Triunfo de la Revolución”**

**Declaración de autoría**

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

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

| _____ | _____ | _____ |
|---|---|---|
| Firma del Autor | Firma del Tutor | Firma del Co-tutor |

**Datos de contacto**

Ing. Antonio Cedeño Pozo (acedeno@uci.cu)

Graduado de Ingeniero Informático de la Universidad de las Ciencias Informáticas en el 2009. Cinco años de experiencia en el desarrollo de software.

Ing. Luis Enrique García (legarcia@uci.cu)

Graduado de Ingeniero Informático de la Universidad de las Ciencias Informáticas en el 2009. Cinco años de experiencia en el desarrollo de software.

Dr. Rafael Arturo Trujillo Codorniu (rtrujillo@ismm.edu.cu).

Máster en Ciencias Físico-matemáticas en 1979, Universidad de Odessa, URSS. Doctor en Ciencias Físico-matemáticas en 1986, Universidad de Rostov del Don, URSS. Ostenta la categoría docente de Profesor Titular.

**Agradecimientos**

A mis padres María Victoria y Jose, y a mi tía Norma que es mi segunda mamá, por su cariño y apoyo incondicional. Por educarme en como soy y seré.

A mi hermana Lissandra y a mi prima Yadira, que es como una hermana también, junto a su esposo Roger por toda su ayuda y apoyo durante estos años.

A mi amada novia Lisandra por su paciencia, ayuda, comprensión e infinito amor. Gracias por amarme, soy feliz de tenerte a mi lado. Te amo.

A mis suegros por todo su apoyo en esta etapa de mi vida.

A mis abuelos, y a los que ya no están, por todo su amor y sacrificio.

A mis tutores por su ayuda y amistad.

A mis familiares y amistades.

**Resumen**

En el presente material se exponen las principales características de los manejadores de dispositivos desarrollados para el Sistema de Supervisión y Control de Procesos “Guardián del ALBA”. El trabajo explicará además el funcionamiento del Recolector Gráfico, desarrollado con el objetivo de brindar mantenimiento a la comunicación del sistema SCADA con los dispositivos de campo. Se explica el funcionamiento de un algoritmo de compresión de datos utilizado para reducir el consumo de espacio en el almacenamiento de la información generada en el proceso de adquisición. A partir de una serie de actividades desarrolladas, el autor realiza una explicación sobre temas relacionados con el diseño, implementación y pruebas de los manejadores referentes a los protocolos Modbus y Ethernet/IP, así como los elementos fundamentales que hicieron posible el desarrollo del Recolector Gráfico. Se ofrece al lector una visión sobre los principales resultados obtenidos en materia de productos palpables y puestos en producción, así como los aportes realizados al país desde el punto de vista económico. En el presente trabajo se abordará un conjunto amplio de tecnologías que se encuentran a nivel del estado del arte, y que sirvieron de base para el desarrollo de los productos descritos.

**Palabras clave**

- Sistemas SCADA
- Manejador de dispositivos
- Analizador de tramas
- Compresión de datos
- Protocolos Modbus
- Protocolo Ethernet/IP
- Wavelet
- SPIHT

## Índice

## Índice de Ilustraciones

## Índice de Tablas

## Introducción

El acontecimiento más relevante y dramático ocurrido durante toda la historia de la industria petrolera venezolana ha sido sin dudas el sabotaje cometido contra PDVSA entre diciembre del año 2002 y enero de 2003. Los hechos fueron organizados por dirigentes de los sindicatos petroleros de sectores de la oposición que promovieron la paralización de la industria. Los objetivos estaban muy bien delimitados, lograr la desestabilización de la principal fuente de ingresos de Venezuela que terminaría por asfixiar al gobierno bolivariano y obligaría a la renuncia del presidente Hugo Chávez. Una de las maniobras más significativas del sabotaje estuvo marcada por el bloqueo de los sistemas de automatización existentes para la supervisión y el control de los procesos dentro de la industria petrolera. Los sectores opositores una vez más tuvieron a su favor la mano del imperio, pues gran parte de los sistemas de software con los que contaba PDVSA estaban controlados por INTESA, una empresa norteamericana dedicada a este tipo de actividades, que ante el golpe petrolero, adoptó una actitud servil a los intereses de la oposición. A partir de este lamentable incidente, el Gobierno Bolivariano decidió comenzar el largo camino para lograr una verdadera soberanía tecnológica, la concepción de la idea de desarrollar un sistema de supervisión y control para la industria petrolera, constituye sin dudas uno de los primeros pasos para lograr el objetivo deseado. A mediados del año 2006 comenzó a desarrollarse el sistema que daría solución a la supervisión y el control de los procesos en la industria petrolera de Venezuela. El equipo de trabajo se dividió en dos, la parte cubana y la venezolana. El capital humano de la parte cubana se nutrió de estudiantes y profesores de la Universidad de las Ciencias Informáticas y de profesionales en diferentes especialidades de la Universidad Central de las Villas Marta Abreu, del Instituto Superior Minero Metalúrgico de Moa y del CEDAI, organismo al que pertenecen gran parte de los especialistas en automática que asesoran el desarrollo. El “Equipo Cuba” inició sus actividades a partir de la primera iteración de la fase de elaboración, se realizó la definición de una arquitectura candidata y se crearon diferentes líneas de trabajo para dar cumplimiento a los requisitos recibidos de la parte venezolana (Pozo, 2009)

En PDVSA existe un gran número de elementos de hardware destinados a la automatización de los procesos fundamentales para la producción, el sistema debía ser capaz de ofrecer un mecanismo para el acceso a la información tanto de proceso como de estado de diferentes dispositivos, sensores y actuadores, así como lograr almacenar dicha información de forma persistente por largos períodos de tiempo. El intercambio de datos con dichos instrumentos se realiza mediante protocolos de comu-

nicación, que no son más que el "lenguaje" mediante el cual se realizan las peticiones de escritura y lectura de las variables configuradas y de relevancia para el sistema SCADA.

La información que se obtiene está formada por registros que contienen 4 campos: el identificador de la variable a que se refiere el registro, el instante de tiempo en que se efectuó la medición, el valor tomado por esa variable en el instante de tiempo mencionado y la calidad del dato. El almacenado de esta información es responsabilidad del componente Base de Datos Histórica (BDH) que constituye un insumo para muchos otros procesos de análisis, tales como los subsistemas de reportes, monitoreo estadístico, ajuste de lazos de control, detección de fallas, diagnóstico de averías, minería de datos, etc. Su diseño está sometido a requerimientos de naturaleza contradictoria, que se relacionan todos con el gran volumen de información que es necesario almacenar. Por un lado debe ser capaz de aceptar la adición de nuevos registros en tiempo real y de ejecutar consultas que involucren un número significativo de datos en un tiempo aceptable para los usuarios. La implementación natural de la base de datos con los cuatro campos mencionados necesitaría de al menos 18 bytes por registro, lo que en un sistema que mida sólo 3000 variables analógicas con un período de muestreo de 1 segundo, implicaría la generación de 4,5 GB diarios de información. Esta cantidad es excesiva para muchos ordenadores y afectaría las capacidades de respuesta a las solicitudes de los clientes. Consecuentemente en muchos sistemas de supervisión se utilizan diferentes esquemas de compresión para los datos históricos, esto ofrece varias ventajas, entre las que se encuentran la disminución del tiempo de E/S y el ahorro de espacio en los medios de almacenamiento masivo. El uso eficiente de éstos es importante aún con las grandes capacidades que ofrecen los discos actuales. La compresión de datos tiene, sin embargo, costos ocultos si los datos resultan inutilizables para los propósitos deseados. Algunos sistemas, para evadir la compresión, optan por almacenar los datos puntuales sólo por un período relativamente corto (días) y a más largo plazo guardar solamente los valores promediados en cierto intervalo (hora, turno, día o semana). Sin embargo, puede apreciarse que esto, en esencia, es un mecanismo de compresión con una gran pérdida de información (Trujillo Codorniu, et al., 2008).

Ante esta situación, se analizó el siguiente **problema científico**: ¿Cómo realizar la comunicación con dispositivos de campo y comprimir los datos provenientes de ellos de forma eficiente, con mecanismos para el mantenimiento de la comunicación ante posibles fallos?

Para resolver este problema se planteó como **objeto de estudio** la manipulación de la información en sistemas SCADA.

El **objetivo general** consistió en el desarrollo de un conjunto de manejadores encargados del acceso a los dispositivos, de una aplicación de mantenimiento para monitorear el flujo de información entre el sistema de supervisión y los elementos de campo, así como el desarrollo del módulo para la compresión de la información obtenida de los dispositivos antes mencionados.

Como **campo de acción** los mecanismos para el acceso y compresión de la información de proceso o de estado de los dispositivos de campo en los sistemas SCADA.

Para dar cumplimiento al objetivo planteado, en la línea se propuso el desarrollo de las siguientes **tareas de investigación**:

- Estudio de las tecnologías y mecanismos de transporte de tramas a través de los medios físicos más usados para el uso en la interfaz genérica de los manejadores.
- Diseño e implementación de una interfaz genérica para los manejadores.
- Desarrollo de los manejadores Modbus ASCII, Modbus RTU y Modbus TCP.
- Desarrollo del manejador Ethernet/IP.
- Estudio y selección del lenguaje de programación y framework para interfaces gráficas para ser utilizadas en el desarrollo del Recolector Gráfico.
- Estudio de los mecanismos de análisis de tramas para ser incorporados en el componente de análisis de tramas del Recolector Gráfico.
- Diseño e implementación del Recolector Gráfico.
- Diseño e implementación de las extensiones de interpretación de tramas Modbus ASCII, Modbus RTU, Modbus TCP y Ethernet/IP para el analizador de tramas.
- Estudio de los esquemas de compresión usados comúnmente en los sistemas SCADA para ser incorporados como parte del estado del arte.
- Estudio e implementación de los posibles modelos de transformada Wavelet aplicables a los datos obtenidos de los dispositivos.

- Evaluación de resultados de cada modelo de transformada Wavelet sobre diferentes conjuntos de datos.
- Implementación de un mecanismo para la selección adaptativa del modelo de transformada Wavelet a utilizar según el conjunto de datos entrante.
- Estudio de mecanismos de compresión de datos para seleccionar el de mejor desempeño en el entorno de los sistemas.
- Implementación del algoritmo SPITH para la compresión de los datos transformados.

De manera general, las tareas relacionadas ilustran los retos fundamentales a los que se enfrentaron los integrantes del "Equipo Cuba". El autor de este material participó de manera activa y directa en la realización y cumplimiento de cada una de ellas. Para el desarrollo no se requirió de ningún tipo de inversión, los recursos humanos se prepararon sobre la marcha, la bibliografía consultada se encuentra de forma gratuita en Internet y las tecnologías utilizadas desde el punto de vista del software fueron siempre software libre. Contar con un equipo propio para el desarrollo de manejadores de dispositivos y otros componentes constituyó un reto enorme, el impacto del resultado está basado en diferentes criterios: el primero de ellos es el relacionado con la independencia tecnológica, las transnacionales proveedoras de tecnologías para la automatización a nivel mundial, obtienen enormes ganancias por concepto de software, y claro está, por las actualizaciones periódicas de los mismos, los manejadores para acceder a los dispositivos constituyen una parte importante de esas ganancias; al contar con un equipo de desarrollo de manejadores no sólo se reutilizarían los manejadores creados para los instrumentos instalados en PDVSA, sino que se contaría con suficiente conocimiento acumulado para enfrentar el desarrollo de nuevos manejadores para futuras adquisiciones de hardware. Por otra parte Cuba se encuentra en el proceso de transición hacia el software libre, y en esa tarea el desarrollo de varios componentes del SCADA resulta de vital importancia para todas las empresas nacionales que pretenden sustituir sus sistemas propietarios por tecnologías libres, ya que es posible reutilizar todos estos componentes para crear nuestro propio sistema (Pozo, 2009).

Se utilizarán varios **métodos científicos** de investigación como:

- Analítico – Sintético: permitió conocer los principales fundamentos y teorías relacionadas con la adquisición de datos y los modelos de compresión de datos.
- Histórico - Lógico: con el objetivo de conocer la evolución de las teorías y tendencias de los mecanismos de adquisición.

# 1 Fundamentación Teórica

## 1.1 Medios de comunicación industriales

En las comunicaciones industriales el principal objetivo es trasmitir la mayor cantidad de información en el menor tiempo posible. Esto es particularmente cierto si tenemos en cuenta las condiciones en las que se desarrollan, como son las posibles interferencias de máquinas eléctricas y otros. Por esta razón el medio de comunicación más utilizado en este campo es el serial, debido a su bajo coste de instalación y su gran fiabilidad, que compensa su baja velocidad de transmisión. Otro de los medios que se utiliza para estos fines es Ethernet, que ofrece mayores velocidades de transmisión y mayor fiabilidad, aunque a un costo mayor.

### 1.1.1 Comunicación Serial

La comunicación serial es un protocolo muy común para transmisión de datos entre dispositivos. Consiste en que el puerto serial envía y recibe bytes de información un bit a la vez. Aun cuando es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias. Por ejemplo, la especificación IEEE 488 para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre dos dispositivos cualesquiera; por otro lado, utilizando comunicación serial el largo del cable puede llegar a los 1200 metros. Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión: Tierra (o referencia), Transmitir, Recibir. Debido a que la transmisión es asincrónica, es posible enviar datos por una línea mientras se reciben por otra. Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

- **Velocidad de transmisión (baud rate)**: Indica el número de bits por segundo que se transfieren, y se mide en baudios (bauds).

- **Bits de datos**: Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits.

- **Bits de parada**: Usado para indicar el fin de la comunicación de un solo paquete. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes.

- **Paridad**: Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad a un valor para asegurarse que la transmisión tenga un número par o impar de bits. La paridad marcada y espaciada en realidad no verifica el estado de los bits de datos, simplemente fija el bit de paridad en 1 para la marcada, y en 0 para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos.

Un ejemplo de protocolo que utiliza fundamentalmente la comunicación serial es el Modbus, que es un protocolo de comunicaciones situado en el nivel 7 del modelo OSI y está basado en la arquitectura maestro/esclavo (MODICON, 1996). Se ha convertido en un protocolo de comunicaciones estándar en la industria, siendo el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales (Gómez Johnson, 2008).

### 1.1.2 Comunicación Ethernet

Ethernet (también conocido como estándar IEEE 802.3) es un estándar de transmisión de datos para redes de área local que se basa en el principio de que todos los equipos de una red Ethernet están conectados a la misma línea de transmisión y la comunicación se lleva a cabo por medio de la utilización un protocolo denominado CSMA/CD (Carrier Sense Multiple Access with Collision Detect, que significa que es un protocolo de acceso múltiple que monitorea la portadora: detección de portadora y detección de colisiones). Con este protocolo cualquier equipo está autorizado a transmitir a través de la línea en cualquier momento y sin ninguna prioridad entre ellos.

Ethernet, que ha experimentado un gran éxito y recibió la aceptación universal en el mundo empresarial, ha empezado a encontrar su camino en los más exigentes ambientes industriales. Las ventajas de Ethernet incluyen la capacidad de supervisar las transmisiones de datos, la existencia de servicios públicos (como Telnet) que se utilizan para configurar y reconfigurar equipos, y la capacidad

de carga de los programas de control de dispositivos desde una ubicación central. Estos tipos de aplicaciones se refieren a menudo como "a nivel de gestión", aunque hay un movimiento en marcha para adoptar Ethernet "en el nivel de dispositivo". El uso de conexiones TCP/IP permite transmitir señales de control, tales como la robótica, la operación de equipos de alta velocidad, y aplicaciones SCADA (Supervisory Control And Data Acquisition) (Introducción a Ethernet Industrial, 2005).

Aunque los buses de campo continúan dominando las redes industriales, las soluciones basadas en Ethernet se están utilizando cada vez más en el sector de las tecnologías de automatización, donde las secuencias de procesos y producción son controladas por un modelo cliente/servidor con controladores, PLC y sistemas SCADA, teniendo acceso a cada sensor que se conecta a la red.

## 1.2 Elementos de comunicaciones industriales

Cuando se habla de dispositivo en la terminología de las ciencias informáticas, se hace referencia a los componentes de la computadora, es decir, elementos de hardware que interactúan para brindar un conjunto de funcionalidades útiles para los usuarios. ¿Cuál es el mecanismo que permite la comunicación del sistema operativo con los diferentes periféricos? Los manejadores de dispositivos son los encargados de realizar esta tarea. Un manejador de dispositivo (en inglés, driver), no es más que programa informático o un elemento de hardware que permite al sistema operativo interactuar con los periféricos, haciendo una abstracción del hardware y permitiendo, mediante una interfaz bien definida, el acceso a los mismos. En el presente capítulo se abordarán temas relacionados con los dispositivos y manejadores de dispositivos, pero trasladando los conceptos hacia el mundo de la automatización de procesos industriales.

### 1.2.1 Dispositivos

En la actualidad existe un gran número de dispositivos, entre los que se destacan los llamados dispositivos de entrada-salida (E/S), divididos en tres grandes grupos: dispositivos de interfaz de usuario, son los que permiten la comunicación entre los usuarios y la computadora; dispositivos de almacenamiento, que se utilizan para proporcionar almacenamiento de datos y memoria; y dispositivos de comunicaciones, que permiten la comunicación entre nodos a través de una red.

Al mencionar los términos de dispositivos industriales y dispositivos de campo, estamos haciendo referencia a elementos de hardware, tales como Controladores Lógicos Programables (PLC), Computadoras Industriales, Sistemas de Control Distribuidos, sensores o actuadores inteligentes con capa-

cidad de comunicación. Cada uno de estos elementos puede almacenar información de proceso o estado de sí mismo y forma parte de un sistema automatizado.

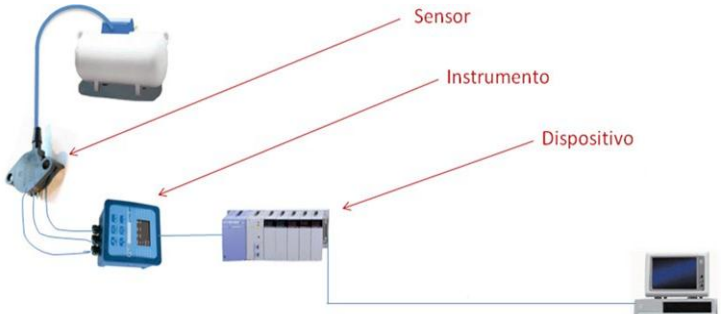


**Figura 1 - Elementos de campo**

Un sensor es un elemento que detecta manifestaciones de cualidades o fenómenos físicos, como la energía, velocidad, aceleración, tamaño, cantidad, etc. El instrumento por su parte es una especie de traductor de la magnitud física obtenida por el sensor en otra que facilita su medida. Y los dispositivos controlan la lógica de funcionamiento de las máquinas, plantas y procesos industriales, realizando además operaciones aritméticas para manejar señales y realizar estrategias de control (Herrera, 2008).

Dentro de los dispositivos de campo más difundidos encontramos a los Controladores Lógicos Programables (PLC). Un autómata programable industrial, como también se los conoce, es un equipo electrónico de control con un cableado interno independiente del proceso a controlar, que se adapta a dicho proceso mediante un programa de software específico que contiene la secuencia de operaciones a realizar.

### 1.2.2 Protocolos

En el mundo de las ciencias informáticas se conoce como protocolo de comunicación a determinadas reglas que deben cumplir los dispositivos que desean comunicarse, de forma análoga a un idioma, los nodos deben aprender la gramática, la sintaxis y todas las reglas del idioma para lograr comunicación entre ellos. En los protocolos se pueden reconocer algunas propiedades, por ejemplo:

- Rigen los pasos para comenzar la comunicación entre dos nodos.
- Determinan el inicio y el fin de los mensajes, así como el resto del formato de los mismos.

- Realizan la corrección de los errores.
- Marcan la terminación de la sesión de conexión.
- Plantean estrategias de seguridad.

Cuando se trata de protocolos de comunicación no puede dejar de mencionarse al modelo de referencia de Interconexión de Sistemas Abiertos (OSI), considerado un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones y formado por las capas o niveles que se muestran a la izquierda en la **Figura 2**.

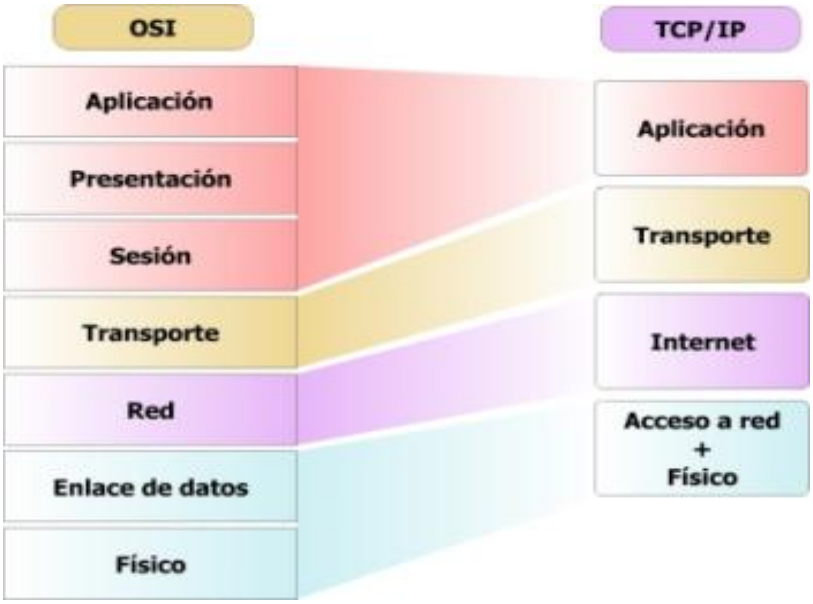


**Figura 1 - Modelos OSI y TCP/IP**

El modelo OSI fue desarrollado en 1983 y adoptado en 1984 por la Organización Internacional para la Estandarización (ISO, por sus siglas en inglés), dicho modelo de referencia muestra cómo debe transmitirse un mensaje entre nodos en una red de datos, posee siete niveles o capas, donde la función de una capa “n” es la de proveer un servicio a una capa “n+1” (Organización Internacional de Normalización, 2004).Por su parte en el modelo TCP/IP, como se muestra en la **Figura 2**, cada nivel corresponde a uno o más niveles del modelo de referencia OSI, y constituye, por lo tanto, una simplificación de éste último.

### 1.2.3 Manejadores de dispositivos

Los manejadores de dispositivos constituyen piezas esenciales de los sistemas computarizados, pues sin ellos no se podría usar el hardware. Se puede afirmar que existen tantos tipos de manejadores como tipos de periféricos, y con mucha frecuencia se puede encontrar más de un manejador para un mismo dispositivo, donde cada uno ofrece un nivel distinto de funcionalidades. Normalmente son los fabricantes del hardware quienes escriben sus controladores, ya que conocen mejor el funcionamiento interno de cada aparato, pero también se encuentran controladores libres, por ejemplo en los sistemas operativos libres. En este caso, los creadores no son siempre miembros de la empresa fabricante, aunque a veces hay una cooperación con ellos, lo que facilita el desarrollo.

## 1.3 Interfaces de comunicación para dispositivos de campo

En el campo de la informática industrial existen diversas organizaciones que definen sus propios estándares de comunicación para sus dispositivos. Algunos de estos se han usado en mayor o menor medida en dependencia de las necesidades de las empresas. Entre estos estándares se ha destacado fundamentalmente el propuesto por la OPC Foundation, como una forma de integración entre los demás existentes, que pudiera satisfacer las necesidades de las empresas que decidieron formar parte de la fundación.

### 1.3.1 Estándar OPC para el acceso a datos

OPC (OLE for Process Control) es un protocolo de comunicaciones abierto que permite la comunicación entre aplicaciones informáticas y que permite la interoperabilidad entre diferentes fabricantes de software y hardware.

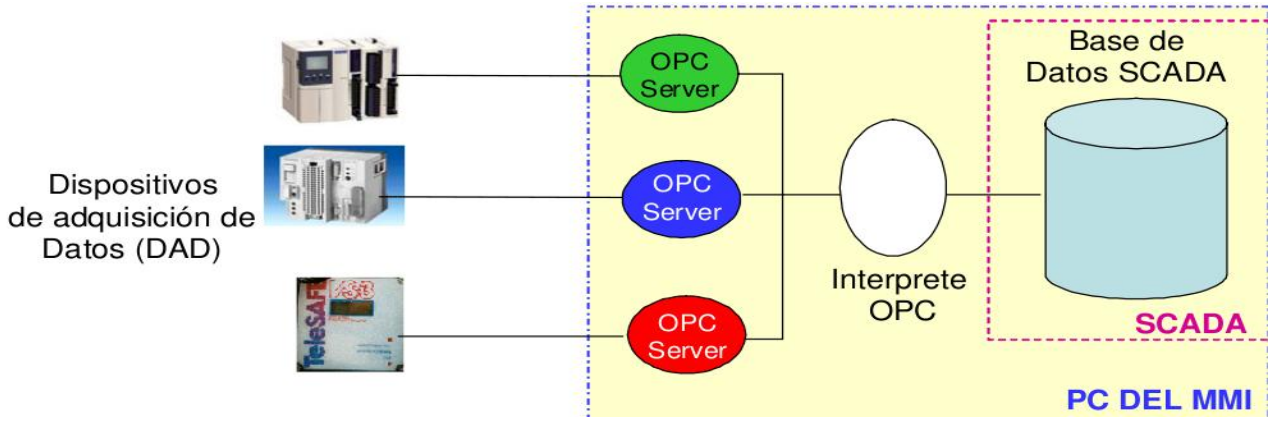


**Figura 2 - Arquitectura OPC**

OPC establece que los fabricantes de cada dispositivo desarrollan un servidor OPC, el cual por un lado se comunica con su dispositivo y por el otro es capaz de responder o recibir información en un formato estándar. La ventaja de este método es que permite, con un solo intérprete, comunicarse con cualquier dispositivo que tenga asociado un servidor OPC, por lo que la actualización o incorporación de nuevos dispositivos será fácil y se ampliará el ciclo de vida de los paquetes.

El estándar OPC consta de varias especificaciones:

- OPC-DA (Data Access): Permite el intercambio de datos a tiempo real entre servidores y clientes usando tecnología DCOM (Distributed Component Object Model).
- OPC-AE (Alarms & Events): Proporciona alarmas y notificaciones de eventos.
- OPC B (Batch)- Útil en procesos discontinuos.
- OPC DX (Data eXchange)- Proporciona interoperabilidad entre varios servidores.
- OPC HDA (Historical Data Access)- Acceso histórico a datos OPC.
- OPC S (Security)- Especifica cómo controlar el acceso de los clientes a los servidores.
- OPC XML-DA (XML Data Access)- Sirve para el intercambio de datos entre servidores y clientes como OPC-DA, pero en vez de utilizar la tecnología COM/DCOM, utiliza mensajes SOAP (sobre HTTP) con documentos en XML.
- OPC CD (Complex Data)- Permite a los servidores exponer y describir tipos de datos complejos en forma de estructuras binarias y documentos XML (García, 2009).

## 1.4 Compresión de datos históricos

La compresión de datos se define como el proceso de reducir la cantidad de datos necesarios para reproducir eficazmente una información, es decir, la eliminación de datos redundantes. La operación de restaurar la señal original de los datos comprimidos es llamada reconstrucción. Hay dos tipos de compresión: sin pérdida y con pérdida. En el primer caso el interés es reconstruir los datos de manera exacta sin pérdida de información, como puede ser la compresión de texto. En el segundo caso se puede aceptar cierto error siempre y cuando la calidad (medida por algún criterio) de la reconstrucción sea aceptable. En este último caso se encuentra la compresión de sonido (el oído hu-

mano solo percibe los sonidos en una banda reducida), de imágenes y video (la pérdida de algunos píxeles no es perceptible por el ojo humano), entre otras.

Las necesidades de la telemetría espacial motivaron a los investigadores a analizar la compresión de datos de campo desde los años 60 y 70 del siglo pasado. A principios de los 80, Hale y Sellars describieron los algoritmos de compresión “Boxcar” y “Backward Slope” que han sido usados en el tratamiento de históricos (Bader , et al., 1987) (Kennedy , 1988). Más recientemente se popularizaron los algoritmos tipo "Swinging Door" (Bristol, 1990).

Estos algoritmos se basan esencialmente en la predicción de los valores que “deben” tomar las variables, a partir de los valores anteriormente tomados por la misma. Si la medición arroja un valor que coincide (con una precisión prefijada) con la predicción realizada (para esa variable y en ese instante de tiempo) se asume que esa medición no arroja información nueva y por tanto su valor se desprecia (opcionalmente se desprecia también la marca de tiempo). Como muchas variables industriales tienen variaciones lentas y, por tanto, coeficientes de auto correlación altos, este esquema logra una buena compresión de los archivos históricos y resulta suficientemente rápido como para que la compresión se pueda efectuar en tiempo real. En la práctica para la predicción se usa el valor anterior o los últimos dos valores solamente. Estos métodos usan esencialmente la extrapolación.

A modo de ejemplo se muestran los pseudo códigos del algoritmo “Boxcar”, usado actualmente por el SCADA cubano “EROS” (Grupo de Desarrollo EROS, 2002), y del algoritmo “Backward Slope” (Hale, et al., 1981).

```
IF (el valor del punto actual difiere del último valor salvado en más de max_deviation)
THEN
{
        IF (punto anterior ≠ último punto salvado)
        THEN {salvar el punto anterior}
        ELSE {salvar el punto actual}
}
```

**Figura 3 - Algoritmo “Boxcar”**

Estimar el valor actual de la variable por extrapolación lineal a partir de los dos últimos valores.

*IF (el valor del punto actual difiere del valor estimado en más de max_deviation)*
*THEN*
*{*
    *IF (punto anterior ≠ último punto salvado)*
    *THEN {salvar el punto anterior}*
    *ELSE {salvar el punto actual}*
*}*

**Figura 4 - Algoritmo "Backward Slope"**

En (Thornhill, et al., 2004) se hace un excelente análisis de los efectos de estos algoritmos de compresión. Algunos de los elementos que se mencionan los enumeramos a continuación:

- Si se desea alta fidelidad en los datos los niveles de compresión son muy bajos.
- Las propiedades estadísticas de las señales pueden ser alteradas. En particular los valores medios de la señal original y de la reconstruida pueden ser diferentes. Lo mismo ocurre con la integración. La varianza de la señal reconstruida es siempre menor que la varianza de la señal original. Esto puede tener implicaciones serias en aplicaciones tales como balances de masas.
- La reconstrucción puede omitir eventos de interés en la señal, o generar eventos falsos. Por ejemplo pueden no mostrarse adecuadamente extremos locales de la señal y generarse falsos extremos en la reconstrucción.
- Cuando se decide no almacenar un punto, por no ser significativo, se pierde toda la información del mismo (incluyendo su marca de tiempo). Sin embargo, en ocasiones, la información de en qué momento se efectuó la medición es necesaria, independientemente de si el valor resultó significativo o no.

- La obtención de un juego de datos con menor grado de detalle (y mayor tasa de compresión) requiere la descompresión de los datos y la repetición de la compresión con los nuevos parámetros.

## 1.5 Herramientas para el desarrollo.

En este epígrafe se resaltan las tecnologías y herramientas utilizadas en el desarrollo de los componentes que se mencionan en este documento. A continuación se exponen algunas de las características de estos productos que justifican su utilización.

### 1.5.1 Lenguaje de programación.

Un lenguaje de programación es un idioma diseñado para expresar operaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para expresar algoritmos con precisión o crear programas que controlen el comportamiento físico y lógico de una máquina. C++ es un lenguaje imperativo orientado a objetos derivado del C. Al igual que su ancestro sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel así como una gran velocidad para la ejecución, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.

### 1.5.2 Entorno Integrado de Desarrollo.

Un entorno de desarrollo integrado o en inglés *Integrated Development Environment* (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Eclipse es un IDE multiplataforma que emplea módulos (*plugins*) para la extensión de sus funcionalidades, gracias a los mencionados *plugins* es posible utilizar múltiples lenguajes, entre ellos C++, permite la integración con varios *frameworks* de desarrollo como es el caso de Qt, así como el empleo de mecanismos de control de versiones como CVS y subversion (SVN) (Eclipse Foundation, 2010).

### 1.5.3 Framework para interfaces gráficas.

Qt es un framework multiplataforma para desarrollar interfaces gráficas de usuario y otros tipos de aplicaciones. Es mantenido por la división de software Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el desarrollador original del framework. Utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en otros lenguajes de programación. Las API del framework ofrecen capacidades como: acceso a

bases de datos, manipulación de archivos, soporte para XML, gestión de hilos, soporte de red, desarrollo de interfaces gráficas, portabilidad hacia sistemas empotrados como teléfonos móviles, entre otras (Nokia Corporation, 2010).

### 1.5.4 Herramienta de modelado.

Las Herramientas CASE (*Computer Aided Software Engineering* o Ingeniería de Software Asistida por Ordenador) son aplicaciones informáticas utilizadas para aumentar la productividad en el desarrollo de software.

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite modelar todas las fases del desarrollo, así como generar código desde diagramas, generar documentación, realizar ingeniería inversa, entre otras. También proporciona abundante documentación sobre UML. Es distribuido bajo licencia gratuita y comercial.

## 2 Manejadores de dispositivos industriales

La función principal de los drivers del sistema SCADA Guardián del ALBA es enlazar el SCADA con los diferentes dispositivos del campo. La interacción entre los manejadores y los dispositivos se efectúa mediante el envío y recepción de mensajes a través de un medio físico determinado. Los manejadores se encargan de la adquisición de los datos traduciendo los protocolos de campo a un protocolo genérico (Interfaz Genérica). Se permite la programación de estos en módulos independientes al sistema para evitar la re-compilación del sistema ante incorporaciones de nuevos protocolos.

Los protocolos de comunicación con los dispositivos definen la semántica de los mensajes, y su estructura dejando la trasmisión de los mismos a capas inferiores de transporte (por ejemplo Ethernet para TCP/IP o RS-232 para comunicación serie).

### 2.1 Interfaz Genérica de Manejadores

Para proporcionar una interfaz estándar de acceso a los manejadores de dispositivos y ocultar al SCADA las diferencias entre los protocolos y características de hardware de los dispositivos enlazados a los mismos, fue necesaria la creación de una arquitectura y una interfaz de comunicación que se denominó Interfaz Genérica. Esta arquitectura debía ser lo suficientemente general para asimilar la implementación de un gran conjunto de manejadores y al propio tiempo dicha generalidad no debía impactar negativamente en el rendimiento de los manejadores, aspecto muy importante en la supervisión y control de procesos.

Unos de los diseños más simples para la comunicación entre el SCADA y los manejadores de dispositivos consiste en informarle inicialmente al manejador las direcciones que son de interés para el SCADA y la frecuencia con que se desean obtener los valores en estas direcciones y establecer una interfaz a través de la cual el SCADA reciba periódicamente los valores referidos a esas direcciones. Este diseño presupone que cada manejador cree uno o varios contextos de para llevar a término la tarea encomendada. Este diseño, sin embargo, al dejar demasiadas responsabilidades al manejador, no permite una planificación centralizada de las tareas del SCADA. Con este modelo cada manejador despacha sus tareas de acuerdo a sus propios criterios, haciendo difícil que el planificador del SCADA detenga las tareas de recolección en unos dispositivos para priorizar otras.

En la interfaz genérica (IG de ahora en adelante) se utilizó un diseño más complejo pero que facilita que las tareas de recolección queden bajo el mando del planificador del SCADA y que se minimicen los contextos que tengan que ser creados internamente en los manejadores. De esta manera, en

lo posible, los manejadores serían reactivos, respondiendo a las solicitudes del SCADA en los propios contextos del mismo (Trujillo Codorniu, et al., 2008).

### 2.1.1 Soporte de múltiples modelos de cooperación.

Por **modelo de cooperación** entendemos la forma en que dos o más entidades de la red industrial deciden realizar los intercambios de información. Los modelos que se decidieron soportar en la IG son el modelo Cliente / Servidor y el modelo Productor / Consumidor. La IG tiene funciones dedicadas a ambos mecanismos, permitiendo que incluso en el modelo productor / consumidor, no sea necesaria la creación de contextos de ejecución en el manejador.

### 2.1.2 Solicitudes reducibles

Es común que los protocolos de acceso a dispositivos permitan la lectura de un conjunto de variables a la vez bajo determinadas condiciones. Por ejemplo, el protocolo Modbus (Modbus IDA, 2004) (Modbus IDA, 2006) permite leer un conjunto de registros consecutivo siempre y cuando la longitud del mensaje que se genere no sobrepase los 256 bytes. A su vez, en el protocolo Ethernet/IP (Rockwell Automation, 2000) (Open DeviceNet Vendor Assoc., 2001) pueden leerse variables de tipo arreglo siempre y cuando la cuota de bytes a transferir no exceda los 488 bytes. Al generalizar la interfaz a cualquier protocolo es imposible que el SCADA conozca si una solicitud múltiple es realizable por el protocolo o si para ejecutarla es necesario segmentar la solicitud en varias equivalentes. Para manejar eficientemente esta situación se introduce el concepto de solicitud reducible.

Aplicado al modelo de cooperación Cliente-Servidor entendemos por solicitudes reducibles a aquellas solicitudes del cliente que pueden ser divididas en dos o más solicitudes sin que el costo de la transacción aumente. Este costo de la transacción generalmente se refiere al tiempo de procesamiento, al número de mensajes a trasmitir por el canal, o al tiempo de ocupación del dispositivo o canal.

Las solicitudes reducibles son ineficientes para el SCADA ya que ocupan demasiado tiempo el dispositivo impidiendo el despacho rápido de solicitudes de mayor prioridad. Se presupone que el SCADA asignará diferentes prioridades a los procesos de lectura y escritura. Suponemos además que una operación de alta prioridad debe tratar de tener una latencia mínima en el SCADA. Pueden encontrarse solicitudes del SCADA al manejador que requieran la transmisión de varios mensajes hacia el dispositivo para su realización. En este caso el dispositivo podría no atender solicitudes de mayor prioridad por un tiempo relativamente largo. Para evitar este fenómeno la IG exige que todas

las solicitudes que realice el SCADA al manejador sean irreducibles. Para ello ofrece una función que dada una solicitud del SCADA devuelve la lista de solicitudes irreducibles funcionalmente equivalente (Trujillo Codorniu, et al., 2008).

### 2.1.3 Parametrización de los manejadores.

Dentro de los requerimientos que se establecieron para los manejadores está que sean configurables y que la configuración de los mismos se almacene en la Base de Datos de Configuración del SCADA. Para atender este requerimiento se dotó a la interfaz genérica de funciones para obtener el listado de los parámetros de configuración del manejador, de las redes y de los dispositivos asociados al mismo, especificando el nombre de cada parámetro, una descripción ampliada del mismo, el tipo de dato, los valores posibles (en los casos enumerados o el rango para parámetros flotantes) y el valor por defecto a los efectos de facilitar la edición visual de estos parámetros en los despliegues del SCADA. Esta configuración se intercambia en formato texto.

### 2.1.4 Acceso a información de diagnóstico.

Independientemente de que en cada operación de lectura el manejador debe reportarle al SCADA la calidad de los datos leídos, se diseñaron funciones para acceder a una información de diagnóstico más detallada que facilita la detección de fallas de hardware y de comunicación.

La cantidad y el tipo de la información de diagnóstico será dependiente del protocolo pero todos deben reportar, al menos, los siguientes parámetros:

1. Estado de comunicación del Dispositivo (**COMMSTATE)**.
2. Estado del dispositivo (**STATE**).
3. Porcentaje de Éxito de Comunicación del Dispositivo (**COMMSUCESS**)

### 2.1.5 Arquitectura multicapa

Generalmente los protocolos de comunicación con los dispositivos definen la semántica de los mensajes, y su estructura dejando la transmisión de los mismos a capas inferiores de transporte (por ejemplo TCP/IP para Ethernet o comunicación serie para RS-232). El SCADA debe soportar una interfaz suficientemente general para que a ella puedan conectarse los diferentes manejadores, independientemente de su naturaleza, y suficientemente eficiente como para que se aprovechen al máximo las posibilidades de cada protocolo, simplificando el diseño de los manejadores. Basados en esta concepción general los manejadores deben desarrollarse como un sistema multicapa (García, 2009).

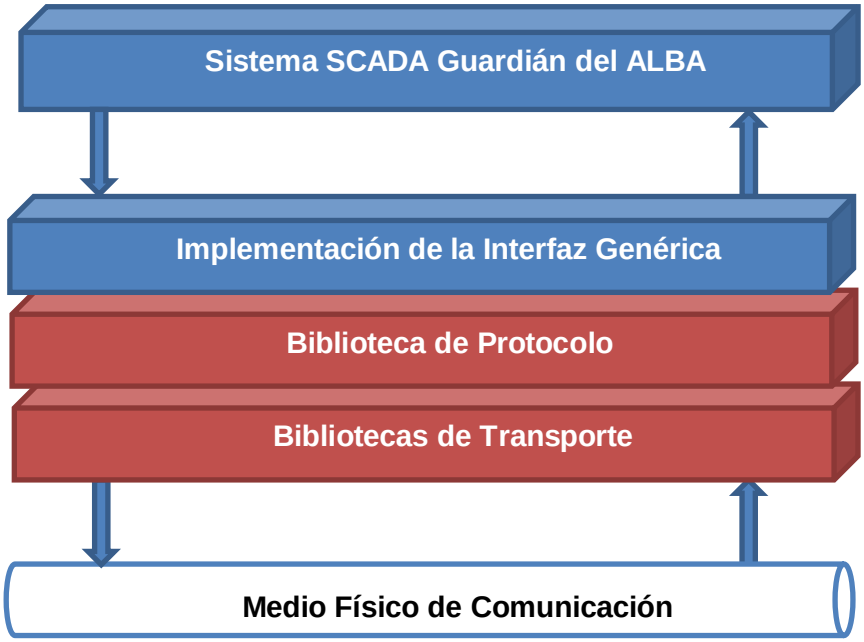


**Figura 5 – Arquitectura Multicapas de la Interfaz Genérica**

Esta arquitectura presenta las siguientes ventajas:

- Las dos primeras capas de los manejadores son reutilizables y tienen valor por sí mismos. En particular disponer de las bibliotecas de protocolo permite modificar de manera más simple la interfaz genérica en caso necesario.
- Cada capa tiene una funcionalidad lógica propia. La capa de transporte envía y recibe mensajes, la capa del protocolo construye e interpreta los mensajes y la capa de implementación traduce la IG en términos del protocolo.

Los manejadores deben residir en bibliotecas dinámicas que se cargan por el recolector de acuerdo a la configuración. Aunque la especificación propuesta de la interfaz genérica es compatible con el ANSI C, subyace en la misma una jerarquía de clases.

Como es usual, cuando se mapean objetos a lenguajes que no soportan esta funcionalidad, en vez de usar métodos se usan procedimientos en los cuales uno de los parámetros es el identificador de la instancia de la clase a que pertenece el método. En la siguiente tabla se ofrecen los procedimientos de la biblioteca, agrupados de acuerdo a la clase a que se asocian.

### 2.1.6

### 2.1.7 Biblioteca DriversCore

La Implementación de la interfaz genérica debe traducir en términos de llamadas a la biblioteca del protocolo la interfaz genérica del SCADA.

| Clases | Métodos |
|---|---|
| Biblioteca | listDrivers, createDriver, closeHandle |
| Manejador | createDevice, getParametersValues, setParametersValues, validateDeviceAddress, listDiagnosticParameters, getDiagnosticParameterValue, readAsynchronous, getVendorInformation, getCapabilities, setSniffer, enableSniffer, getErrorString, driverOpenEnumeration |
| Dispositivo | deviceAttachVariableList, deviceGetParametersValues, deviceSetParametersValues, deviceGetAddress, deviceSetAddress, deviceValidVariableAddress, deviceOpenEnumeration, deviceListDiagnosticParameters, deviceGetDiagnosticParameterValue, deviceRead, deviceWrite, deviceBlockPeriod |
| Enumerador | enumerate |

**Tabla 1 – Funciones de la Interfaz Genérica**

### 2.1.8 Biblioteca TransportProvider

La biblioteca de transporte es la encargada de manejar la conexión (si existiera) y la transmisión de un flujo de datos a través del medio físico. La biblioteca del protocolo proporciona una serie de funciones (API) que encapsulan la construcción e interpretación de los mensajes necesarios para la comunicación. La biblioteca del protocolo utiliza la capa de transporte de modo que no debe entrar en las especificidades de cómo se envían o reciben los mensajes.

## 2.2 Manejadores Modbus

El manejador para el protocolo Modbus fue desarrollado tomando como base la biblioteca DriversCore. Para el envío de las tramas a través del medio físico se hace uso de la biblioteca de transporte TransportProvider.

Modbus es un protocolo de comunicaciones situado en el nivel 7 del Modelo OSI, basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLC). Convertido en un protocolo de comunicaciones estándar de facto en la industria, es el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales. Algunas de las razones por las cuales el uso de Modbus es superior a otros protocolos de comunicaciones son:

- Es público
- Su implementación es fácil y requiere poco desarrollo
- Maneja bloques de datos sin suponer restricciones

Modbus permite el control de una red de dispositivos y comunicar los resultados a un ordenador. También se usa para la conexión de un ordenador de supervisión con una unidad remota (RTU) en sistemas de supervisión adquisición de datos (SCADA). Existen versiones del protocolo para puerto serie y para Ethernet (Modbus/TCP).

Cada dispositivo de la red Modbus posee una dirección única y cualquier dispositivo puede enviar órdenes, aunque lo habitual es permitirlo sólo a un dispositivo maestro. Cada comando contiene la dirección del dispositivo destinatario de la orden. Todos los dispositivos reciben la trama pero sólo el destinatario la ejecuta (salvo un modo especial denominado "Broadcast"). Cada uno de los mensajes incluye información redundante que asegura su integridad en la recepción. Los comandos básicos permiten controlar un dispositivo RTU para modificar el valor de alguno de sus registros o bien solicitar el contenido de dichos registros.

El protocolo define la estructura de los mensajes que los controladores reconocen y usan, independientemente del tipo de red sobre la cual se comunican. Describe además el proceso que debe implementar cada controlador para realizar una solicitud a otro dispositivo, como debe responder ante las solicitudes de otros dispositivos y como los errores son detectados y reportados. Además establece el formato de los mensajes.

### 2.2.1 Modelo de Datos de Modbus

Modbus basa su modelo en una serie de tablas que tienen características distintivas. Las 4 tablas primarias son:

- **Entradas discretas (Input Bits)**: Son entradas digitales para el dispositivo. Su tamaño es de un BIT por lo que tiene sólo dos posibles estados: Activada (ON), desactivada (OFF). El acceso a ellas es de sólo lectura por lo que el estado de estos elementos dentro del dispositivo no se puede modificar a través de las funciones del protocolo.

- **Salidas discretas (Coils):** Son salidas digitales para el dispositivo. Al igual que las entradas discretas tienen sólo dos posibles estados: Activada (ON), desactivada (OFF). El acceso a estos elementos es de lectura y escritura por lo que a través de las funciones del protocolo pueden leerse y modificarse sus valores.

- **Registros de entrada (Input Registers):** Son entradas de 16 bits al dispositivo. Generalmente se refieren a entradas analógicas del dispositivo. El acceso a estos elementos es de solo lectura.

- **Registros de salida (Holding Registers):** Son elementos de salida del dispositivo de 16 bits asociados generalmente a salidas analógicas del dispositivo. El acceso a estos elementos es de lectura y escritura.

Para cada una de las tablas primarias el protocolo permite hasta 65536 elementos. Las operaciones de lectura y escritura con estos elementos están diseñadas de manera que se puedan acceder a múltiples elementos consecutivos hasta un tamaño límite que depende del tipo de transacción y del dispositivo.

Cada elemento de las tablas de Modbus puede ser referenciado, para ellos se utilizan los llamados números de referencias y los números de registro. Por razones históricas los números de referencia se expresan como números enteros que comienzan a partir del 1, sin embargo Modbus utiliza una numeración más natural para sus registros comenzando por el registro 0.

Además de las referencias del elemento dentro de cada tabla, Modbus utiliza referencias globales. Las referencias 1xxxx (es decir de la 0x10001 a 0x1FFFF) se relacionan con las entradas discretas, las referencias 0xxxx (es decir de la 0x00001 a 0x0FFFF) se relacionan con las salidas discretas o bobinas; las referencias 3xxxx (de la 0x30001 a 0x3FFFF) a registros de entrada y finalmente las referencias 4xxxx (de la 0x40001 a 0x4FFFF) a los registros de entrada/salida (holding registers).

### 2.2.2 Modelo de Transacciones de Modbus

Los controladores Modbus, que se conectan a través de redes seriales, se comunican usando el paradigma "Amo / esclavo" ("Master / Slave"). Bajo este paradigma sólo un dispositivo (el Amo o "Master") puede iniciar una transacción. Los demás dispositivos (esclavos) responden a las solicitudes, o bien enviando los datos solicitados, o bien ejecutando las acciones requeridas por el amo (Ver Figura 4).

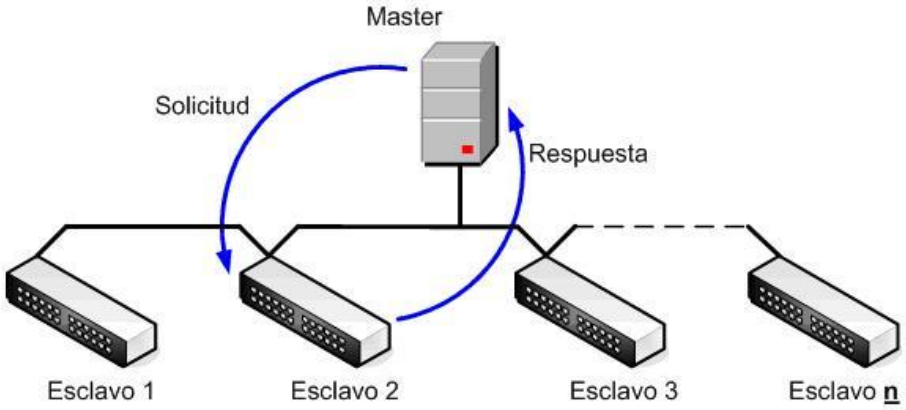


**Figura 6 - Paradigma Maestro/Esclavo**

### 2.2.3 Estructura de los Mensajes Modbus

El protocolo define una unidad de datos (PDU) independiente del tipo de transporte subyacente. Cada variante del protocolo introduce algunas diferencias en el encabezado o en el chequeo de errores. El encabezado, el PDU y el campo de chequeo de errores forman de conjunto la unidad de datos de la Aplicación (Application Data Unit), (Ver Figura 5).

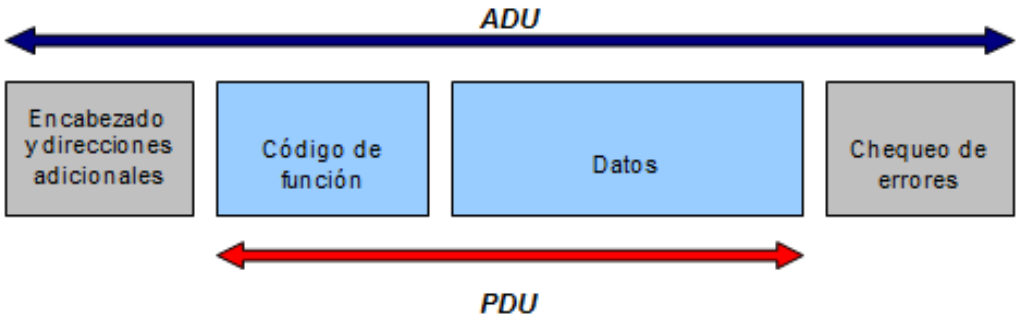


**Figura 7 - Unidad de Datos de Aplicación**

### 2.2.4 Características de Modbus TCP

Los dispositivos Modbus TCP no necesitan un identificador como sus homólogos serie ya que se identifican en la red con su dirección IP y el puerto.

Teniendo en cuenta que un cliente puede iniciar un número de transacciones consecutivas con cualquier servidor Modbus TCP, se establece entre los datos técnicos de los dispositivos el parámetro "**NumberMaxOfClientTransaction**", que indica precisamente la cantidad máxima de transacciones consecutivas admisibles por ese dispositivo. Su valor normalmente va entre 1 y 16 y depende de la capacidad del dispositivo y el tamaño de lo que denominan ventana TCP.

Para este protocolo el campo de Encabezado tiene una longitud de 7 bytes con la estructura mostrada en la siguiente tabla:

| Campo | Longitud | Descripción |
|---|---|---|
| Identificador de transacciones | 2 bytes | Identificador de la Transacción |
| Identificador del protocolo | 2 bytes | 0 = Protocolo MODBUS |
| Longitud del mensaje | 2 bytes | Número de bytes que siguen en el mensaje. |
| Identificador de la Unidad | 1 byte | Identificador del esclavo. |

**Tabla 2 - Encabezado de la trama Modbus**

El campo de chequeo de errores en se omite ya que se delega el chequeo de la trama en el protocolo TCP.

### 2.2.5 Características de Modbus RTU

Modbus RTU es una representación binaria compacta de los datos que se utiliza sobre medios de comunicación seriales. Para el protocolo Modbus RTU el campo de Encabezado y direcciones adicionales consta de solo 1 byte que contiene la dirección del dispositivo esclavo. Esta dirección debe estar en el rango de 1 a 247. A su vez el campo de chequeo de errores consta de un entero de 16 bits que contiene el CRC (Chequeo de Redundancia Cíclico) calculado a partir del mensaje.

### 2.2.6 Características de Modbus ASCII

Modbus ASCII es una representación legible del protocolo pero menos eficiente, y también se utiliza sobre medios de comunicación seriales. En este protocolo todos los elementos de la trama Modbus RTU están presentes con la diferencia que los datos son codificados de manera diferente y se le agregan separadores ASCII al inicio y opcionalmente al final de la trama. El campo de chequeo de errores en este caso es un byte generado por el algoritmo LRC. El sistema de codificación es ASCII hexadecimal. Por tanto por cada byte RTU se trasmiten dos caracteres ASCII hexadecimales ("0" – "9" y "A" – "F").

En el modo ASCII la trama comienza con el carácter "**:**" (ASCII 0x3A) y termina con el par de caracteres retorno del carro – cambio de línea (CRLF) que son los caracteres ASCII 0x0D 0x0A. El byte de control LRC es calculado antes de hacer la conversión a ASCII y no incluye el encabezado ("**:**").

Los caracteres a transmitir permitidos para el resto de los campos son los caracteres que corresponden a los dígitos del sistema hexadecimal.

Los dispositivos esclavos conectados al bus, monitorean continuamente la red en espera del carácter "**:**". Cuando éste es recibido entonces cada dispositivo decodifica el siguiente campo (para esto deben leer dos caracteres) para averiguar si la dirección del esclavo es la suya.

Intervalos de hasta un segundo pueden transcurrir entre dos caracteres consecutivos dentro del mensaje. Si un intervalo de tiempo mayor ocurre, el dispositivo receptor asume que un error ha tenido lugar. La trama ASCII típica es:

| Encabezado | Identificador* | Función* | Datos* | Byte de LRC* | Fin de Trama |
|---|---|---|---|---|---|
| 1 byte "**:**" | 2 bytes | 2 bytes | 2n bytes | 2 bytes | 2 bytes CRLF |
| (* = en hexadecimal) | | | | | |

**Tabla 3 - Trama Modbus ASCII típica**

Para algunos dispositivos Modbus ASCII se admite que el mensaje termine después del byte de control LRC sin que los caracteres CRLF sean enviados. Un intervalo de tiempo de al menos un segundo debe transcurrir y el controlador considerar que el mensaje ha terminado exitosamente.

### 2.2.7 Servicios soportados por el protocolo Modbus

Modbus ofrece 24 funciones diferentes cada una de las cuales está identificada por un código de 1 byte de tamaño. No necesariamente los dispositivos soportan todas las funciones. La columna "Nivel de compatibilidad" de la siguiente tabla indica en qué medida un servicio específico puede ser soportado por un dispositivo Modbus. Si el Nivel de Compatibilidad de un servicio es igual a cero, esto indica que todos los dispositivos obligatoriamente lo soportan. Mayores valores del nivel de compatibilidad indican menores probabilidades de que el servicio se soporte por un dispositivo o servicios muy dependientes del fabricante y por tanto no aconsejables para la interoperabilidad en general.

| Código | Descripción de la función asociada | Nivel de compatibilidad |
|---|---|---|
| **0x01** | **Leer múltiples bobinas (0xxxx).** | **1** |
| **0x02** | **Leer múltiples entradas discretas (1xxxx).** | **1** |
| **0x03** | **Leer múltiples registros de entrada/salida (4xxxx).** | **0** |
| **0x04** | **Leer múltiples registros de entrada (3xxxx).** | **1** |
| **0x05** | **Escribir una bobina (0xxxx).** | **1** |
| 0x06 | Escribir un registro de entrada/salida (4xxxx). | 1 |
| 0x07 | Lectura del estado de error | 1 |
| 0x08 | Acceso a los contadores de diagnóstico | 3 |
| 0x09 | Modos de operación y carga / descarga remota | 3 |
| 0x0A | Solicitud de reporte de operación | 3 |
| 0x0B | Lectura del contador de eventos | 3 |
| 0x0C | Lectura de los eventos de conexión | 3 |
| 0x0D | Modos de operación y carga / descarga remota | 3 |

| Código | Descripción de la función asociada | Nivel de compatibilidad |
|---|---|---|
| 0x0E | Solicitud de reporte de operación | 3 |
| **0x0F** | **Escritura de múltiples bobinas** | **2** |
| **0x10** | **Escritura de múltiples registros de entrada/salida.** | **0** |
| 0x11 | Leer identificación del esclavo | 3 |
| 0x12 | Modos de operación y carga / descarga remota | 3 |
| 0x13 | Reseteo de esclavo después de error no resuelto | 3 |
| 0x14 | Lectura de referencia general | 2 |
| 0x15 | Escritura de referencia general | 2 |
| 0x16 | Escritura con máscara de Registros (4xxxx). | 2 |
| 0x17 | Lectura / escritura de Registros (4xxxx). | 2 |
| 0x18 | Lectura de la cola FIFO | 2 |

**Tabla 4 – Servicios brindados por el protocolo Modbus**

### 2.2.8 Interpretación de los datos almacenados en los registros.

Cuando se accede a los registros, se hace necesario tener en cuenta el formato en que vienen los datos empaquetados en la trama Modbus y como estos son interpretados por el ordenador en dependencia de sus características.

El término inglés Endianness designa el formato en el que se almacenan los datos de más de un byte en un ordenador. Hay dos criterios que se han denominado little-endian y big-endian, cuyo nombre proviene de la novela “Los viajes de Gulliver” de Jonathan Swift en la que los habitantes de los imperios de Lilliput y Blefuscu libran una encarnizada guerra por una disputa sobre el lado por el que debían empezar a comerse los huevos. El problema es similar a los lenguajes que se escriben de derecha a izquierda, como el árabe, o el hebreo, frente a los que se escriben de izquierda a derecha.

El sistema big-endian adoptado por Motorola entre otros, consiste en representar los bytes en el orden "natural": así el valor hexadecimal 0x4A3B2C1D se almacenaría en memoria en la secuencia

{4A, 3B, 2C, 1D}. En el sistema little-endian adoptado por Intel, entre otros, el mismo valor se almacenaría como {1D, 2C, 3B, 4A}. Algunas arquitecturas de microprocesador pueden trabajar con ambos formatos (ARM, PowerPC, DEC Alpha, PA-RISC, MIPS), y a veces son referidas como sistemas middle-endian.

Modbus es un sistema con un formato big-endian. Como puede apreciarse este tema sólo pudiera crear dificultad en las operaciones sobre registros ya que estos están compuestos por dos bytes.

## 2.3 Manejador Ethernet/IP

Ethernet/IP (Ethernet/Industrial Protocol) es un protocolo de comunicación diseñado para entornos industriales. Permite que los dispositivos de control intercambien información crítica con requerimientos estrictos de tiempo. Entre los dispositivos que pueden interconectarse se encuentran sensores, actuadores y, en general, dispositivos simples de entrada/salida, dispositivos de alta complejidad tales como robots, controladores lógicos programables y otros. Esta particularidad lo hace muy atractivo para enlazar, bajo un mismo protocolo, todos los eslabones del proceso, desde el SCADA hasta los dispositivos más simples.

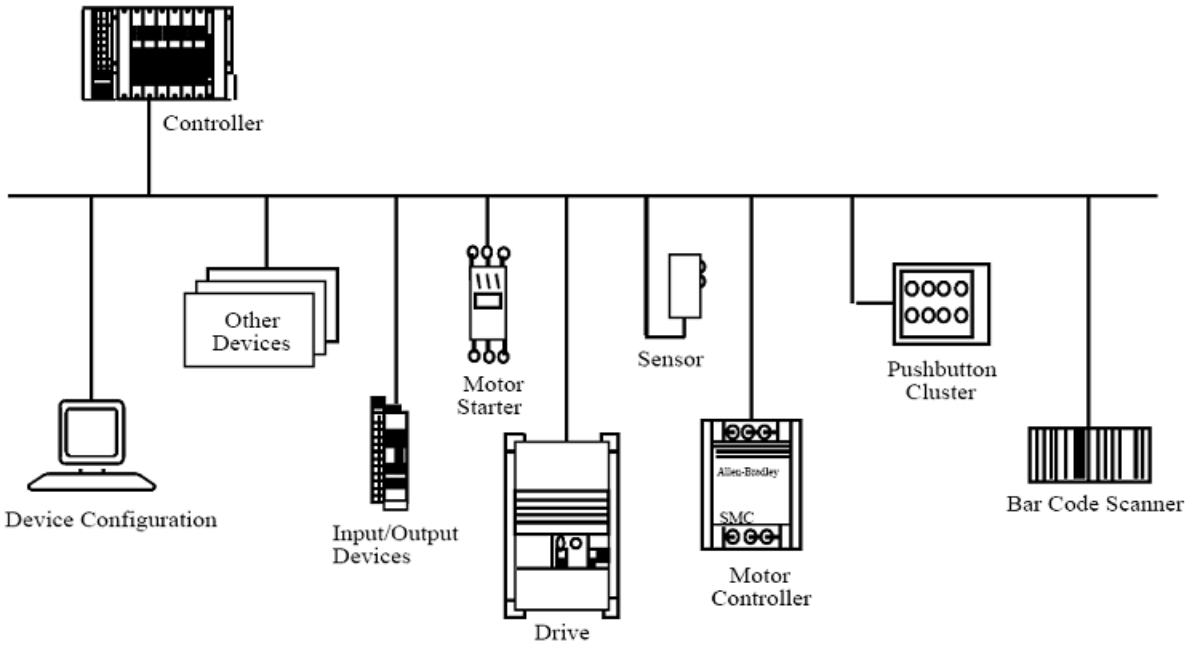


**Figura 8 – Elementos interconectables vía Ethernet/IP**

Ethernet/IP usa el protocolo CIP (Control and Information Protocol). El CIP es un protocolo que abarca las capas de red, transporte y aplicación del modelo OSI y es compartido también por ControlNet y por DeviceNet:

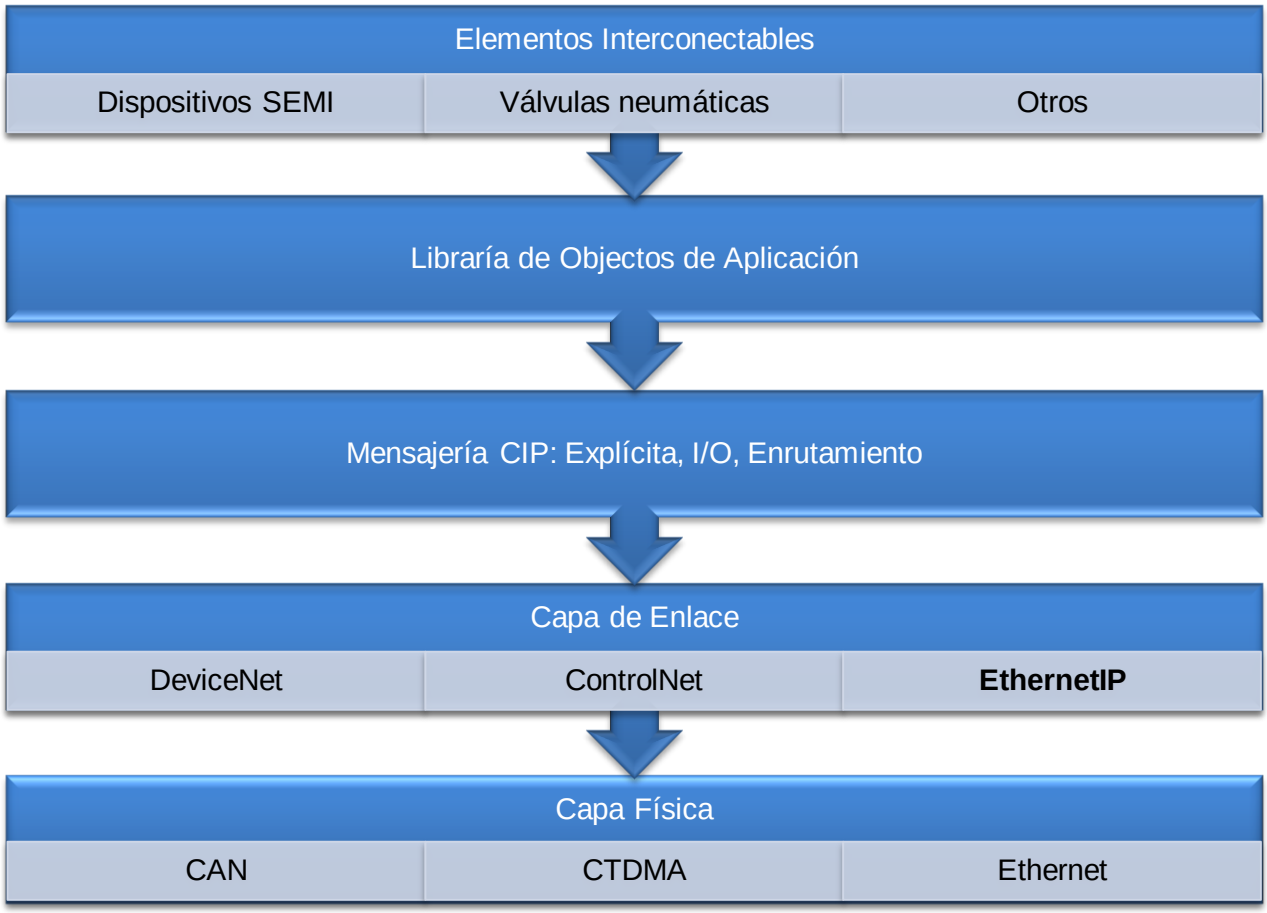


**Figura 9 - Capas del protocolo Ethernet/IP**

El protocolo de Control e Información (CIP) es un protocolo “peer to peer” orientado a objetos que proporciona conexiones entre dispositivos industriales y dispositivos de alto nivel (controladores). CIP es independiente del medio físico y de la capa de enlace de datos.

CIP define un esquema orientado a conexión para facilitar todas las comunicaciones. Una conexión CIP proporciona un camino entre múltiples end-points. Los end-points de una conexión son aplicaciones que requieren compartir datos. A la trasmisión asociada a una conexión en particular se le asigna un identificador cuando se establece la conexión (CID). Este esquema define dos tipos de conexiones que pueden ser establecidas:

- Conexiones I/O. Proporcionan un camino dedicado de comunicación entre la aplicación productora y una o varias aplicaciones consumidoras.

- Conexiones de mensajería explícita. Proporciona un camino genérico, de propósito múltiple, entre dos dispositivos. Los mensajes explícitos proporcionan la interacción típica de solicitud / respuesta en la red.

### 2.3.1 Encapsulación.

Ethernet/IP proporciona un protocolo que encapsula el protocolo CIP, o incluso otros protocolos en tramas Ethernet. El protocolo de encapsulación define un puerto TCP reservado (0xAF12) que debe ser soportado por todos los dispositivos Ethernet/IP y a través del cual se deben aceptar, como mínimo, dos conexiones TCP. De igual forma, el protocolo de encapsulación define un puerto UDP reservado (0xAF12) que debe ser soportado por todos los dispositivos Ethernet/IP y a través de él deben aceptar datagramas UDP. Dado que UDP no tiene la posibilidad de reordenar los paquetes, solo se usa para enviar mensajes encapsulados, el mensaje completo debe contenerse en un único datagrama.

Todos los mensajes encapsulados, enviados vía TCP, o vía UDP al puerto 0xAF12, deben estar compuestos por una cabecera de longitud fija de 24 bytes seguida de una porción opcional de datos. La longitud total del mensaje encapsulado (incluyendo la cabecera) está limitada a 65535 bytes. Su estructura es la siguiente:

| Estructura | Tipo de dato | Valor del campo |
|---|---|---|
| Cabecera del mensaje encapsulado (obligatorio) | UINT* | Comando encapsulado |
| | UINT* | Longitud, en bytes, de la porción de datos |
| | UDINT** | Identificador de sesión |
| | UDINT** | Código de estado. |
| | Arreglo de 8 bytes | Información pertinente al emisor del mensaje. |
| | UDINT** | Bandera de opciones |
| Datos específicos del comando | Arreglo de hasta 65511 bytes | La parte de datos del mensaje encapsulado. |
| *** = Entero sin signo de 16 bits, ** = Entero sin signo de 32 bits** | | |

**Tabla 5 - Cabecera de los mensajes Ethernet/IP**

Los campos enteros de más de un byte deben ser trasmitidos en el formato "little-endian". Aunque la cabecera no contiene información explícita que permita distinguir entre una solicitud y una respuesta, el carácter del mensaje puede ser determinado:

- Implícitamente, por el comando y el contexto en el cual el comando es generado.
- Explícitamente por el contenido de la parte de datos del mensaje.

Los principales comandos posibles de encapsulación son los siguientes:

| Código | Nombre | Comentario |
|---|---|---|
| 0x0000 | NOP | Este comando es usado como un latido para señalizar periódicamente al receptor la necesidad de mantener la conexión viva. Puede ser enviado sólo a través de TCP. |
| 0x0004 | ListServices | Permite obtener los servicios que ofrece el dispositivo. Puede ser enviado tanto a través de TCP como de UDP. |
| 0x0063 | ListIdentity | El que origina una conexión puede usar el comando ListIdentity para localizar e identificar posibles destinos. El comando puede ser enviado como un broadcast usando UDP y no requiere que la sesión esté establecida. La porción de datos de la respuesta a este mensaje proporciona información acerca del dispositivo. |
| 0x0064 | ListInterfaces | El comando opcional ListInterfaces puede ser usado por el que origina una conexión para identificar interfaces de comunicación que respondan a CIP asociadas con el destino. No requiere que la sesión esté establecida. |
| 0x0065 | RegisterSession | Este comando le solicita al destino iniciar una sesión. El destino debe enviar en la respuesta el identificador de sesión que estará presente en los mensajes subsiguientes entre ambos puntos. Puede ser enviado sólo a través de TCP. |
| 0x0066 | UnRegisterSession | El receptor de este comando debe comenzar el cierre de la conexión TCP correspondiente. Puede ser enviado sólo a través |

| | | |
|---|---|---|
| | | de TCP. |
| 0x006F | SendRRData | Transfiere un paquete de solicitud o respuesta hacia el destino encapsulado en la porción de datos del comando. Cuando el protocolo que se encapsula es CIP, el comando se usa para transferir mensajes explícitos. Puede ser enviado sólo a través de TCP. |
| 0x0070 | SendUnitData | Transfiere un paquete de solicitud o respuesta hacia el destino encapsulado en la porción de datos del comando. Cuando el protocolo que se encapsula es CIP, el comando se usa para transferir datos conectados en mensajes de I/O. Puede ser enviado sólo a través de TCP. |

**Tabla 6 – Principales comandos de encapsulación**

### 2.3.2 Protocolo Industrial Abierto.

El protocolo industrial abierto es una extensión de CIP desarrollada por Allan Bradley para acceder a la información de los dispositivos de una manera simple, a través de nombres simbólicos. Esta extensión se encapsula por el mecanismo anteriormente explicado, en paquetes IP.

Los "tags" o nombres simbólicos por los cuales se accede a la información son identificadores con no más de 40 caracteres de longitud. Los tags tienen un alcance. De acuerdo a su alcance se clasifican en tags del controlador (o sea globales) a los cuales se pueden acceder directamente y tags del programa (locales) a los cuales no se puede acceder desde un dispositivo externo. Cada tag tiene un tipo de dato que define la organización interna del dato y las posibles operaciones sobre el mismo. Se soportan tipos atómicos tales como bit, byte, palabras de 16 bits, de 32 bits y tipos estructurados.

En los PLC clásicos (por ejemplo con el protocolo Modbus) donde los datos se arreglan convenientemente en una lista consecutiva de registros o de bits, la obtención de un gran volumen de datos es simplemente cuestión de especificar el punto de comienzo en la memoria y la longitud del segmento a leer. Con el protocolo extendido Ethernet IP de ControlLogix (Rockwell Automation, 2000), esto deja de ser así, precisamente debido a las amplias posibilidades en la organización interna de la memoria que los mismos presentan. Para lograr tasas adecuadas de recuperación de la información se necesitan diferentes esquemas de optimización:

- **Conexiones persistentes:** los resultados de una serie de pruebas realizadas arrojan una diferencia estable y significativa entre el esquema de usar conexiones persistentes y no usarlas del orden de 4 a 1, lo cual indica que la apertura y cierre de la conexión TCP y la creación y destrucción del objeto CIP conexión, en el PLC, es una operación costosa y que si bien el esquema de conexiones volátiles es admisible para pequeñas aplicaciones (admite más de 40 peticiones de lectura por segundo) , para la operación de un SCADA es inadmisible.
- **Lectura de arreglos. Densidad de los paquetes:** una de las maneras clásicas de optimizar las lecturas es tratar de recuperar la mayor cantidad de valores en una transacción simple. En virtud de que mediante el protocolo podemos obtener los elementos de un arreglo (siempre y cuando el tamaño total del mensaje no exceda 488 bytes) si necesitáramos leer en un instante de tiempo los tags Temp[1] y Temp[9] podríamos optar por hacerlo en una transacción leyendo los 9 elementos del arreglo a partir de Temp[1]. Por supuesto que en este caso estamos forzando al PLC a entregarnos información no útil (en este caso los valores Temp[2], Temp[3], ..., Temp[8]), lo cual también tiene un costo. En el documento "Guía de Implementación del Manejador Modbus" se introduce el concepto de densidad de un paquete o bloque de registros que en esencia refleja la proporción de datos útiles, con respecto al total, en el paquete solicitado. Una densidad muy baja en los paquetes podría tener un impacto negativo en el rendimiento al incrementar innecesariamente la cantidad de datos a transferir y ocupar el CPU del PLC en transferencias innecesarias. Por otra parte obligar el uso de densidades muy altas puede provocar fragmentación en las solicitudes lo que también puede disminuir el rendimiento. Los resultados de las pruebas realizadas muestran que no existen diferencias estadísticamente significativas entre la lectura de tags simples y de arreglos. Por tanto se tomó la decisión, para el manejador de no exigir una densidad mínima en los paquetes.

### 2.3.3 Implementación del manejador Ethernet/IP.

La biblioteca del Protocolo Ethernet/IP se desarrolló en dos capas fundamentales; una capa se encarga de implementar el protocolo de encapsulación Ethernet. La otra implementa la capa CIP que es encapsulada en paquetes Ethernet. Los principales comandos que ofrece la capa de encapsulación Ethernet implementada son:

- RegisterSession: Se encarga de registrar la sesión TCP en el dispositivo.
- SendRRData: Permite el envío de los datos encapsulados que usan los mensajes explícitos hacia el dispositivo. Estos mensajes son procesados por el Ruteador presente en todos los dispositivos Ethernet/IP y posteriormente enviados al objeto correspondiente.
- SendUnitData: Permite el envío de los datos encapsulados que usan los mensajes conectados hacia el dispositivo. Estos mensajes van dirigidos directamente a las aplicaciones u objetos con las cuales se concretó la conexión CIP.
- UnRegisterSession: Cierra la sesión TCP abierta anteriormente con el comando RegisterSession.

Encima de esta capa de encapsulación se encuentra la capa de control CIP que se encarga de todo el control de los datos del dispositivo. Los dispositivos son modelados por CIP como una colección de objetos. Cada clase a la que pertenecen los objetos ofrece servicios y posee atributos y comportamiento. Algunos de los servicios comunes implementados son:

- GetAttributesAll. Retorna el contenido de todos los atributos de una clase CIP u objeto.
- SetAttributesAll. Modifica los valores de todos los atributos de una clase CIP u objeto.
- GetAttributeSingle. Retorna el contenido de un atributo específico de un objeto CIP.
- SetAttributeSingle. Modifica el contenido de un atributo especifico de un objeto CIP.
- MultipleServicePacket. Solicita varios servicios a la vez empaquetados, estos servicios son ejecutados independientemente en el dispositivo y los resultados de dichos servicios son enviados de vuelta empaquetados previamente.

La capa CIP permite la comunicación entre el SCADA y las aplicaciones productoras en el dispositivo usando mensajes "conectados", así como la comunicación usando mensajes explícitos o "desconectados". Para que la comunicación del SCADA con un dispositivo se lleve a cabo el protocolo debe ser capaz de generar las direcciones de las variables de forma que el dispositivo las entienda correctamente. Para esto se implementaron todos los segmentos de una dirección, permitiendo cubrir el esquema de direccionamiento presente en las extensiones del protocolo CIP basado en tags. Los segmentos que pueden componer una dirección y están implementados en el protocolo son:

- Port Segment. Indica el puerto por el que se accederá al dispositivo.
- Logical Segment. Utilizado para definir una dirección particular a un objeto.

- Network Segment. Usado para especificar parámetros de red que pueden ser requeridos para transmitir un mensaje al dispositivo a través de la red.
- Symbolic Segment. Contiene un mensaje simbólico en forma de cadena textual.
- Data Segment. Este es el segmento que permite el envío de datos entre las aplicaciones. Conjuntamente con los servicios comunes esta implementación del protocolo cuenta además con los servicios de la extensión Logix para dispositivos Ethernet/IP.

El manejador implementado para el SCADA PDVSA ofrece un mecanismo configurable y adaptable para la comunicación del SCADA con dispositivos Ethernet/IP que soporten el Protocolo Industrial Abierto, cumpliendo además con las características de rendimiento requeridas por el sistema SCADA. Esto satisface plenamente los requerimientos de un sistema de supervisión industrial.

## 3 Recolector Gráfico

La aplicación denominada Recolector Gráfico posibilita la recolección de variables de acuerdo a sus periodos de encuesta. Facilita la puesta a punto y detección de fallas de los manejadores existentes y futuros, así como la corrección de los errores de configuración que imposibilitan la recolección correcta de las variables configuradas. Esta aplicación tiene integrado un componente llamado Analizador de Tramas (Sniffer) que posibilita la exploración de las tramas de comunicación del SCADA, el cual además incluye extensiones (plugins) que permiten el análisis detallado de las tramas de cada protocolo.

Entre sus característica más significativas está que permite configurar de manera gráfica y dinámica los canales, sub-canales, manejadores, dispositivos y variables, además de aquellas propiedades de las mismas que tienen que ver directamente con la recolección. Esta configuración dinámica permite recrear o duplicar configuraciones existentes en el SCADA donde se han detectado fallos. El sistema garantiza la recolección de los puntos configurados de acuerdo a sus periodos de encuesta con la mayor concurrencia posible y muestra, en despliegues tabulares, los valores obtenidos de las variables así como sus marcas de tiempo y calidad.

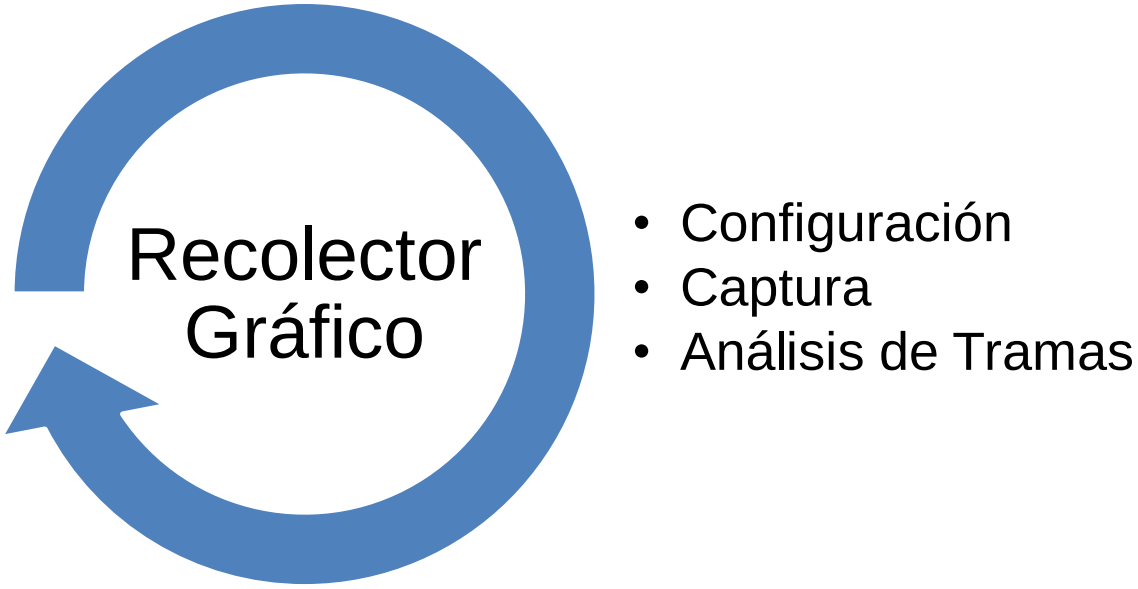


**Figura 10 – Componentes del Recolector Gráfico.**

Cada uno de los componentes de la aplicación establece una comunicación con la biblioteca DriverCore y algunos hacen uso además de las bibliotecas con los manejadores de dispositivos disponibles en el sistema.

Empotrado dentro de cada uno de los componentes existe además un mecanismo que permite a un recolector grafico conectarse a un recolector remoto y sincronizar su contenido, es decir, la configuración, los datos de campo y la información de diagnóstico. Para funcionar, este mecanismo requiere la presencia de un servidor que emite los datos de un recolector hacia los interesados que se suscriben al mismo. Dicho servidor está implementado en una biblioteca que actúa como fuente de información para un recolector gráfico externo. El objetivo de este mecanismo es permitir a los desarrolladores detectar errores durante la ejecución del SCADA, ya que en el recolector del mismo se puede sincronizar con el servidor antes mencionado emitiendo esa información hacia un recolector gráfico remoto.

La aplicación se desarrolló usando el framework Qt que permite el desarrollo de interfaces gráficas, entre otras muchas funcionalidades que fueron de vital importancia para el cumplimiento de los requerimientos de los clientes (Nokia Corporation, 2010).

### 3.1 Componente de Configuración

El componente de configuración es un elemento dinámico capaz de obtener la meta-información brindada por las bibliotecas de manejadores y con ella generar un componente visual de configuración. Este componente permite la modificación de los parámetros de los dispositivos y manejadores previstos en la Interfaz Genérica, además de las propiedades de los puntos de medición que se relacionan directamente con el proceso de recolección. Dicho componente además es capaz de, haciendo uso de la información obtenida, validar los parámetros de configuración en función de los tipos de datos.

Para lograr la validación de los datos, es necesario conocer la información de cada uno de los parámetros mostrados al usuario, por lo que la meta-información que ofrecen los manejadores incluye, entre otros, los siguientes elementos:

- Tipo de dato
- Valor predeterminado
- Rango de valores (en caso de ser un tipo de dato numérico)
- Expresión regular (en caso de ser una cadena de caracteres)
- Lista de valores admisibles (en caso de ser un tipo enumerativo)

Esta información puede ser usada además para mostrar los componentes visuales de una manera más precisa, por ejemplo, si tenemos un parámetro de tipo booleano es preferible que, para configurarlo, se muestre la información en un componente de chequeo, a que se muestren en un componente de texto, en el que se tendría que escribir "Verdadero" o "Falso". Para ello, se hizo uso de una biblioteca que contiene un componente gráfico capaz de generar un conjunto de componentes a partir de meta-información, dicho componente fue adaptado para que utilizara la información ofrecida por los manejadores y así se consiguió un componente completamente dinámico para realizar la configuración del proceso de recolección de la aplicación.

### 3.2 Componente de Captura

Este componente se encarga de brindar una interfaz visual a los usuarios en la que se muestran los valores de las variables, la calidad de las lecturas, las marcas de tiempo obtenidas, entre otras informaciones. Además permite la modificación de las variables mostrando el resultado de dicha modificación, lo que facilita la detección de fallas en el mecanismo de ejecución de comandos.

Para el desarrollo de este componente se implementó un nuevo mecanismo de recolección ajeno al existente en el SCADA, de forma tal que permitiera contrastar los resultados de la ejecución de ambos mecanismos y de esa forma facilitar el proceso de detección de fallas en el SCADA.

### 3.3 Componente de Análisis de Tramas

El analizador de tramas es capaz de interpretar las tramas entrantes y salientes del recolector de acuerdo a cada protocolo transformando la información a un formato entendible por los usuarios. En los casos en que el protocolo que se esté usando no esté soportado por el analizador, la información se presenta, byte a byte, en los sistemas numéricos decimal o hexadecimal, a elección del usuario. La información de cada trama se muestra de manera jerárquica, con diferentes niveles de detalle que se muestran a solicitud del usuario. El analizador de tramas es extensible a nuevos protocolos que se desarrollen a través de una interfaz que permite asociar los protocolos a dicho analizador en forma de plugins. De esta forma la adición de nuevos protocolos se logra sin necesidad de recompilar la aplicación. A continuación se muestra la interfaz que se utiliza entre el recolector gráfico y las extensiones para la interpretación de los protocolos:

```
virtual IPluginNode* parseMessage (const char* data, unsigned length,
                                   unsigned short event, unsigned *error) = 0;
virtual unsigned deletePluginNode (IPluginNode* nodeParam) = 0;
virtual VendorInfo * getVendorInformation() = 0;
virtual const char* getErrorString (unsigned errorCode) = 0;
```

**Figura 11 – Interfaz de los plugins.**

# 4 Compresión de Datos

La compresión de datos se define como el proceso de reducir la cantidad de datos necesarios para reproducir eficazmente una información, es decir, la eliminación de datos redundantes. La operación de restaurar la señal original de los datos comprimidos es llamada reconstrucción. Hay dos tipos de compresión: la compresión sin pérdida y la compresión con pérdida. En el primer caso el interés es reconstruir los datos de manera exacta sin pérdida de información. En el segundo caso se puede aceptar cierto error siempre y cuando la calidad (medida por algún criterio) de la reconstrucción sea aceptable.

En (Thornhill, et al., 2004) se hace un excelente análisis de los efectos de estos algoritmos de compresión. Algunos de los elementos que se mencionan los enumeramos a continuación:

1. Si se desea alta fidelidad en los datos los niveles de compresión son muy bajos.
2. Las propiedades estadísticas de las señales pueden ser alteradas. En particular los valores medios de la señal original y de la reconstruida pueden ser diferentes. Lo mismo ocurre con la integración. La varianza de la señal reconstruida es siempre menor que la varianza de la señal original. Esto puede tener implicaciones serias en aplicaciones tales como balances de masas.
3. La reconstrucción puede omitir eventos de interés en la señal, o generar eventos falsos. Por ejemplo pueden no mostrarse adecuadamente extremos locales de la señal y generarse falsos extremos en la reconstrucción.
4. Cuando se decide no almacenar un punto, por no ser significativo, se pierde toda la información del mismo (incluyendo su marca de tiempo). Sin embargo, en ocasiones, la información de en qué momento se efectuó la medición es necesaria, independientemente de si el valor resultó significativo o no.
5. La obtención de un juego de datos con menor grado de detalle (y mayor tasa de compresión) requiere la descompresión de los datos y la repetición de la compresión con los nuevos parámetros.

### 4.1 Algoritmo de Transformación

Las técnicas de compresión pues ser divididas en dos grandes grupos funcionales: los métodos directos y los métodos sobre transformadas. Los métodos directos son aquellos que pueden ser aplicados en tiempo real sobre los datos. La compresión en estos casos se efectúa “al vuelo” a medida que se obtienen los datos del campo.

Los métodos sobre transformadas realizan una transformación integral previa del conjunto original de datos. La compresión no se realiza sobre los datos originales, sino sobre los datos transformados. No se aplican a medida que se obtienen los valores del campo ya que requieren de un conjunto de datos determinado. Un ejemplo de este tipo de método es la compresión con wavelets.

La Transformada Wavelet, más conocida por su nombre en inglés, Discrete Wavelet Transform (DWT), es una herramienta matemática de reciente desarrollo que permite descomponer una señal no sólo en componentes frecuenciales sino también en componentes espaciales. La DWT, aplicada sobre un arreglo de datos típico, lo convierte en un arreglo en el cual la energía se concentra en solo unos pocos coeficientes. Como los demás coeficientes son pequeños pueden codificarse con muy pocos bits. Por ello se obtienen tasas de compresión muy altas conservando, al mismo tiempo, una buena calidad en la reconstrucción de la señal.

Existen múltiples enfoques para la definición de los wavelets. El que requiere menos prerrequisitos se basa en el esquema “lifting” que consta de tres fases: división, predicción y actualización:

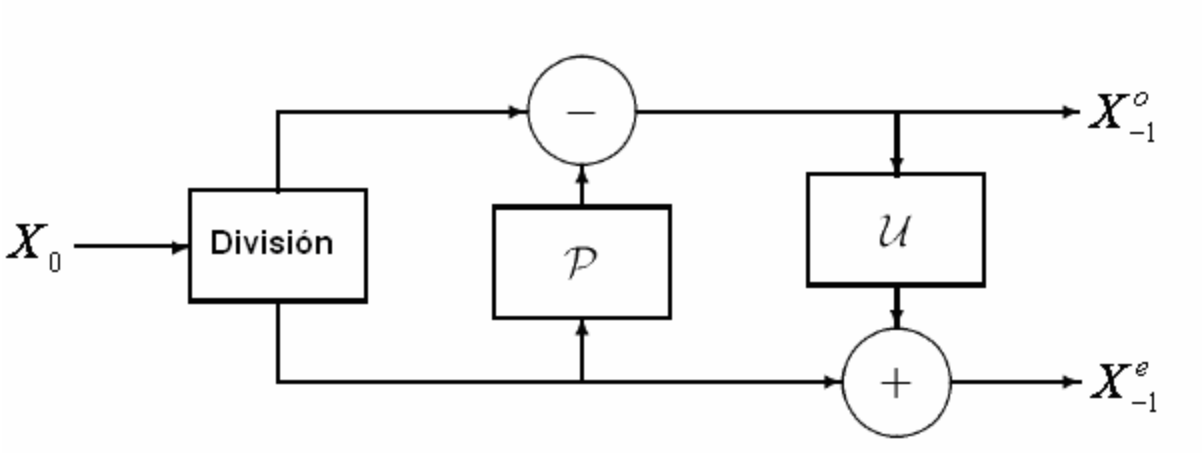


**Figura 12 – Fases del esquema “lifting”: División, Predicción y Actualización.**

### 4.2 Algoritmo de Compresión

Después de aplicar la transformada wavelet al conjunto de datos y una vez cuantificados los valores resultantes se utilizan mecanismos de compresión que permitan disminuir al máximo la cantidad de bits necesaria para la codificación de los datos. Se ha usado (por ejemplo en (Cárdenas-Barrera, et al., 2001)) el algoritmo RLE (Run Length Encoding) para la supresión de las largas cadenas de ceros resultantes de la cuantificación de los coeficientes wavelets pequeños. Uno de los algoritmos que mejores resultados ofrece para la codificación es el algoritmo SPIHT (“Set Partitioning in Hierarchical Trees”) (Said, et al., 1996).

Este algoritmo ha recibido una aceptación universal en la compresión de imágenes al igual que en la compresión de electrocardiogramas (Lu, et al., 2000) que son señales analógicas que requieren una alta calidad en la reconstrucción para que conserven su valor de diagnóstico. El algoritmo basa su funcionamiento en las propiedades estadísticas de los coeficientes wavelet obtenidos luego de aplicar sucesivamente la transformada Wavelet. En particular, se fundamenta en el ordenamiento por la magnitud de éstos y por sus posiciones dentro de la estructura en forma de árbol de la transformada wavelet. Permite la transmisión progresiva de los coeficientes, enviando primero aquellos que tienen mayor relevancia, lo que posibilita detener el algoritmo en un grado de resolución deseado.

### 4.3 Descripción del Algoritmo Propuesto

El algoritmo propuesto para la compresión de datos históricos analógicos consta de las siguientes etapas:

- *División del conjunto de datos en bloques*. Cada bloque representa el conjunto de datos medidos en un intervalo de tiempo determinado. Contiene el identificador de la variable, la primera marca de tiempo de los datos del bloque, la duración, en milisegundos, del intervalo de tiempo que cubre el bloque, la cantidad de valores agrupados y un campo variable que contiene los valores comprimidos. Los mejores resultados del algoritmo se obtienen si la cantidad promedio de datos en cada bloque es mayor que 256 e inferior a 8192 y si, en lo posible, el número de datos es una potencia de dos. La presencia en el bloque de la marca de tiempo inicial y del intervalo de tiempo que cubre permite determinar fácilmente si el bloque debe ser tratado para el manejo de alguna consulta del usuario.

- *Separación de los datos de cada bloque en tres vectores*: el vector de valores; el vector de marcas de tiempo y el vector de calidades. Cada uno será procesado por separado. Estos vectores tienen características diferentes que aconsejan su tratamiento diferenciado.

- *Selección del modelo de transformada a utilizar para el vector de valores.* Para esto, con un subconjunto pequeño de valores, se evalúan diferentes funciones de predicción y actualización: tales como la transformada con interpolación (2, 2), la transformada con interpolación (4, 2) y la transformada simétrica biortogonal (9, 7) (Sweldens, 1997). En base a los resultados obtenidos en este pequeño subconjunto se decide cual esquema se aplicará a todos los datos. Esta evaluación, que al efectuarse con pocos puntos, no incrementa sustancialmente el tiempo de procesamiento, permite mejorar los resultados del esquema.

- *Aplicación de la transformada Wavelet al vector de valores*. A cada elemento se le resta el valor medio de los datos para garantizar que el vector resultante tenga media cero. Esta transformación propicia mejores compresiones en los pasos siguientes. Luego se aplica la transformada wavelet elegida de acuerdo al punto anterior.

- *Cuantificación del vector de coeficientes wavelet con un parámetro configurable*. El vector transformado se lleva a un vector de números enteros mediante un proceso de cuantificación.

El parámetro de cuantificación usado es dependiente de la variable y debe ser objeto de configuración en el SCADA.

- *Comparación con umbrales*. Los coeficientes wavelet cuantificados cuyos valores absolutos sean menores que un umbral configurable son igualados a cero. Este paso permite la eliminación selectiva de ruido en la señal.

- *Codificación del vector resultante con SPIHT*. El vector resultante de los pasos anteriores se codifica utilizando el algoritmo SPIHT. Para ello se utiliza opcionalmente una cuota de bits máxima que es dependiente de la variable y debe ser objeto de configuración en el SCADA.

- *Transformación del vector de marcas de tiempo.* Las variables de un SCADA se miden generalmente en intervalos periódicos, por tanto es presumible que los valores representen una progresión aritmética, o sea, puedan determinarse con alta precisión de acuerdo a la fórmula:

$$t_k = t_0 + k \cdot \frac{t_{n-1} - t_0}{(n-1)}; \quad k = 0,1,\ldots n-1;$$

Por ello, el vector de marcas de tiempo se somete a la siguiente transformación:

$$\tilde{t}_k = t_k - (t_0 + k \cdot \frac{t_{n-1} - t_0}{n-1});$$

Esta transformación debe reducir sustancialmente la norma del vector de marcas de tiempo y provocar que para períodos de muestreos muy exactos el vector resultante sea nulo. Si el vector resultante es nulo (con un margen de tolerancia configurable) no se codifica. Si no es nulo se codifica este vector en el próximo paso.

- *Aplicación de la transformada Wavelet (2, 2) entera al vector de marcas de tiempo*.

- *Comparación con umbrales*. Los coeficientes wavelets del vector de marcas de tiempo transformado, cuyos valores absolutos sean menores que un umbral configurable, son igualados a cero.

- *Codificación con SPITH del vector entero resultante*.

- *Codificación con RLE (Run Length Encoding) del vector de calidades*. La calidad es una función que, en el tiempo, es constante a trozos. Usualmente la calidad de los valores se mantiene estable en largos períodos de tiempo lo que facilita su compresión mediante RLE.

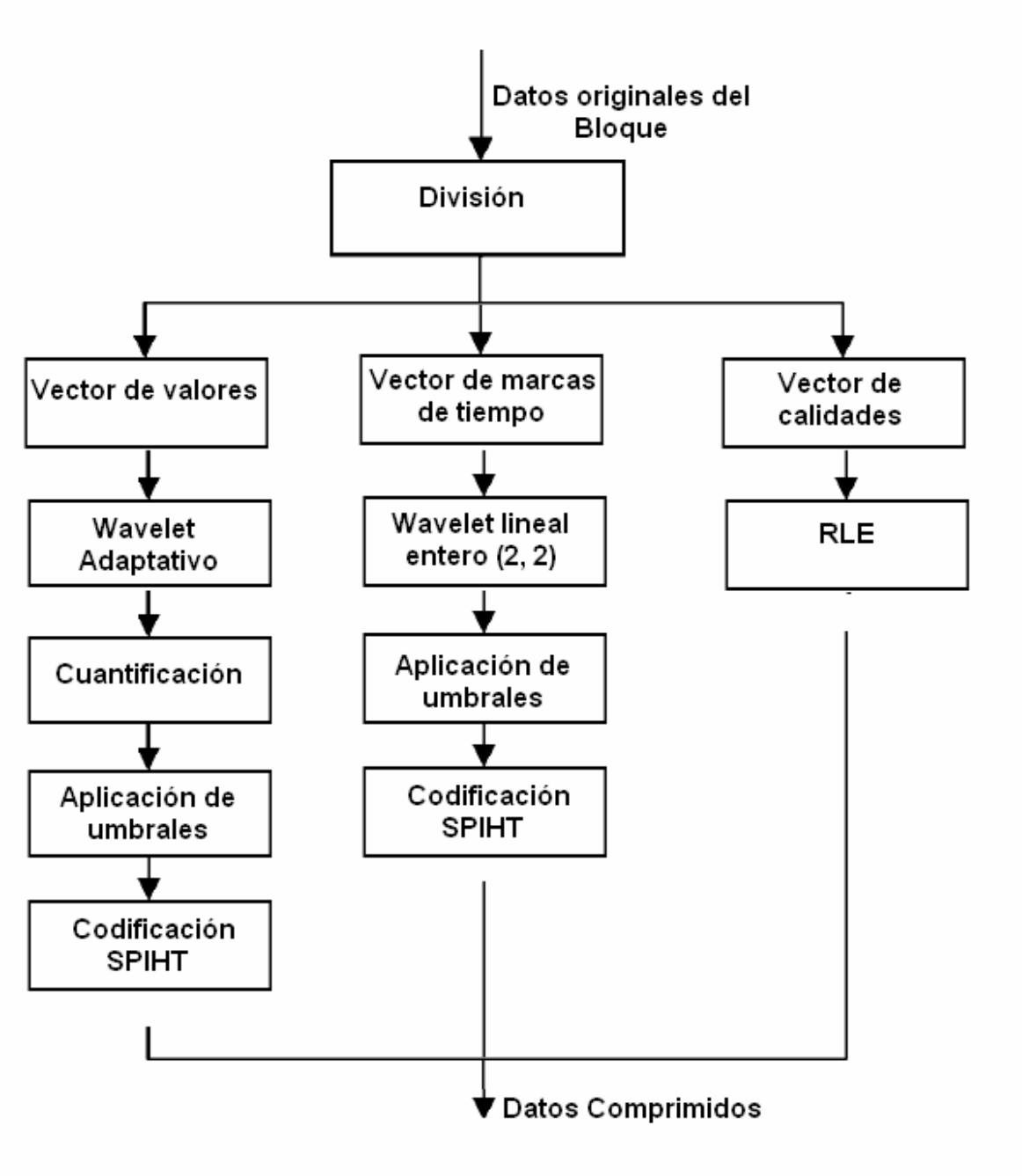


**Figura 13 – Compresión de datos usando Wavelet y SPITH.**

El algoritmo propuesto presenta las siguientes ventajas:

1. La compresión de los datos analógicos lograda mediante la transformada wavelet y SPIHT es, en general, mayor que la lograda por los algoritmos tradicionales. Se logran tasas aceptables de compresión incluso cuando se requieren reconstrucciones sumamente exactas de la señal.

2. La conservación de momentos que se realiza en la fase de actualización de la DWT posibilita una mejor conservación de las propiedades estadísticas de la señal no sólo con bajas tasas de compresión sino también con altas tasas de compresión.

3. Control exacto de la compresión. Se puede establecer "a priori" una tasa de bits por muestra. Esto posibilita prefijar el tamaño ocupado por los registros y por los archivos. La tasa de bits puede reducirse en la medida que los datos van envejeciendo, de manera que se pase de un alto nivel de detalle y baja compresión en los datos más recientes a un bajo nivel de detalle y una alta compresión para los datos más antiguos.

4. La transformación necesaria, del flujo de bits que contiene los datos comprimidos, para pasar a una tasa de bits menor, en la compresión, es simple, debido a las propiedades del algoritmo SPIHT. Esta tarea no requiere una recodificación ya que se reduce a despreciar los últimos bits en los datos comprimidos de los vectores de valores y de marcas de tiempo.

5. Se conserva la información de la marca de tiempo de cada uno de los puntos capturados independientemente de si son significativos o no. La conservación de todas las marcas de tiempo ocupa solo una pequeña parte de la cuota de bits de los datos comprimidos debido a la naturaleza lineal de las marcas de tiempo de una variable que se muestrea de manera periódica.

6. El algoritmo permite la eliminación selectiva del ruido en la señal original.

7. Es posible mantener el formato de base de dato relacional en la BDH si se utiliza un campo binario variable para el almacenamiento de los vectores comprimidos.

8. La representación de la señal a partir de su transformada wavelet facilita los procesos de minería de datos (Tao, et al., 2003).

### 4.4 Resultados del Algoritmo Propuesto

Para evaluar los resultados de la compresión de datos analógicos se han usado datos reales de señales medidas por el Sistema de Supervisión y Control "EROS" (Grupo de Desarrollo EROS, 2002) en la fábrica de Níquel "René Ramos Latour". La compresión obtenida respecto a la señal original se evalúa normalmente por dos parámetros; mediante la tasa de compresión (CR) y mediante la distorsión producida al comprimir la señal (PRD).

La primera señal, que llamaremos Señal A y se muestra a modo de ejemplo, está constituida por 1024 puntos tomados a intervalos de 1 segundo. La transformada Wavelet usada para la compresión fue la Biortogonal 9-7. Los resultados para diferentes niveles de compresión se muestran en la siguiente tabla:

| Bits por muestra | CR | PRD |
|---|---|---|
| 1 bps | 32 | 3.771 |
| 2 bps | 16 | 1.135 |
| 3 bps | 10.7 | 0.720 |
| 4 bps | 8 | 0.340 |

**Tabla 7 – Resultados obtenidos en la compresión de la señal "A"**

Se puede apreciar que incluso con tasas sumamente altas de compresión se alcanzan distorsiones residuales bajas. En esto influye la baja relación ruido/señal que presenta este juego de datos. En las figuras 3, 4 y 5 se muestran los gráficos de la señal original y de las reconstrucciones, apreciándose que, a la percepción, la reconstrucción con 2 bps es excelente.

En todos los casos se calculó la entropía del flujo de bits de salida del SPIHT para evaluar si era factible aplicar un paso adicional con un codificador probabilístico (como el código de Huffman o un codificador aritmético). Los resultados de estas pruebas consistentemente arrojaron que las ganancias en compresión serían mínimas y no compensarían el costo computacional de ese paso adicional.

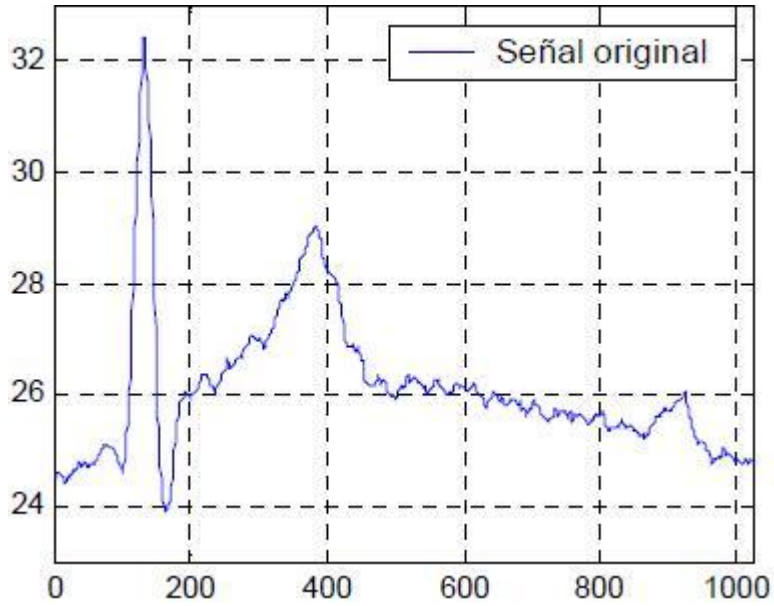


**Figura 14 – Gráfica de la señal “A” original.**

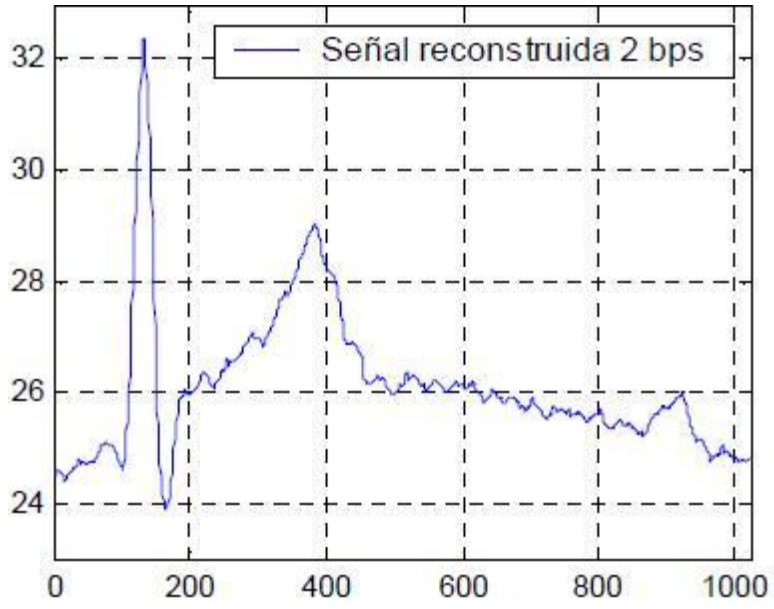


**Figura 15 - Gráfica de la señal “A” reconstruida (2 bps).**

La segunda señal, que llamaremos “B” está constituida por 256 puntos tomados a intervalos de 1 segundo. Esta señal presenta mucha mayor variabilidad y una mayor relación ruido/señal que la señal “A” por lo que la distorsión residual, para los diferentes niveles de compresión, es mucho mayor.

| Bits por muestra | CR | PRD |
|---|---|---|
| 1 bps | 32 | 34.37 |
| 2 bps | 16 | 20.83 |
| 3 bps | 10.7 | 14.95 |
| 4 bps | 8 | 8.29 |

**Tabla 8 – Resultados obtenidos en la compresión de la señal “B”**

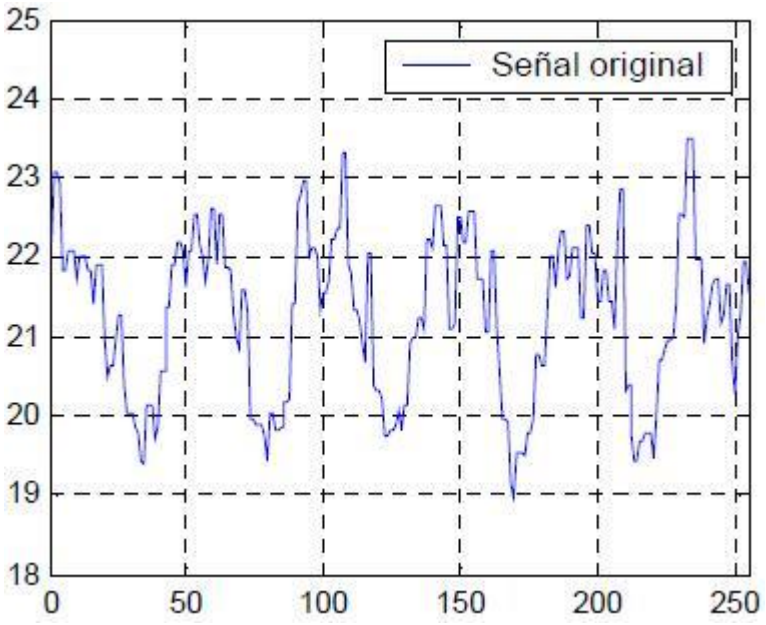


**Figura 16 - Gráfica de la señal “B” original.**

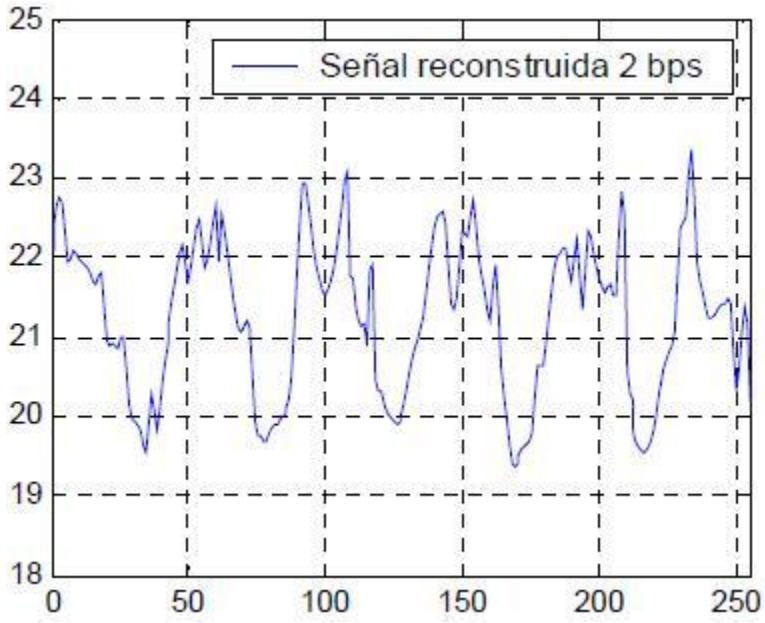


**Figura 17 - Gráfica de la señal “B” reconstruida (2 bps).**

Se puede apreciar (ver Figuras 13 y 14) que la señal reconstruida, incluso para tasas altas de compresión (2 bps), mantiene las características esenciales de la misma, en especial los máximos y mínimos locales de amplitud significativa. El efecto más notable de la compresión es la suavización de la señal con pérdida de armónicos de alta frecuencia.

Los resultados de las pruebas realizadas muestran que el esquema propuesto para la compresión de históricos proporciona altas tasas de compresión y al propio tiempo buena calidad en la reconstrucción. Permite conservar los parámetros estadísticos básicos de la señal, la calidad de cada valor y, dentro de un rango de tolerancia configurable, conserva todas las marcas de tiempo de los valores medidos. Posibilita además incrementar las tasas de compresión sin necesidad de reprocesar la información lo que permite, disminuir el nivel de detalle en los datos.

**Resultados**

Como parte importante del ciclo de desarrollo de los módulos y herramientas tratadas en este material se encuentran las actividades de investigación, pruebas de concepto, análisis de tecnologías, interacción con los clientes, entre otras; como resultado de las mismas se realizaron varios trabajos que fueron presentados en diferentes eventos tanto a nivel nacional como internacional, los más relevantes, así como los premios alcanzados se muestran en la tabla siguiente:

| Título del trabajo | Evento | Premio |
|---|---|---|
| Framework para el desarrollo de Manejadores 4.2 | Jornada Científica a nivel de Universidad 2007 y Concurso Nacional de Computación 2008 | Relevante |
| Analizador de Tramas para Sistemas SCADA | Jornada Científica a nivel de Universidad 2008 y Fórum Provincial 2008 | Relevante |
| Interfaz Genérica para los Manejadores de Dispositivos en el Proyecto SCADA PDVSA | Evento Internacional FIE 2008 | Publicación del artículo |
| Manejador Ethernet/IP para los PLC ControlLogix en el Proyecto SCADA PDVSA | Evento Internacional FIE 2008 | Publicación del artículo |
| Compresión de datos históricos mediante la transformada Wavelet. | Evento Internacional FIE 2008 | Publicación del artículo |
| Compresión de datos históricos mediante la transformada Wavelet. | International Workshop of Mathematics and Computing 2008 | Presentación del artículo |
| Prototipo de Recolector Gráfico para un sistema SCADA. | Jornada Científica a nivel de Universidad 2009 | Relevante |
| Recolector Gráfico para un sistema SCADA. | Jornada Científica a nivel de Universidad 2010 | Relevante |

**Tabla 9 - Participación en eventos**

A continuación se hace referencia a diferentes pilotos instalados en varias plantas pertenecientes a la empresa PDVSA en la República Bolivariana de Venezuela. Los buenos resultados obtenidos en las pruebas realizadas avalan el desempeño de los productos desarrollados por el equipo Cuba.

| Planta | Ubicación | Fecha de Instalación |
|---|---|---|
| Patio de tanques Bajo Grande | San Francisco - Edo. Zulia | Diciembre 2008 |
| Planta de Compresión Altagracia | Altagracia de Otituco - Edo. Guárico | 2008 |
| AIT – Barinas | Localidad Campo de Mesa - Edo. Barinas | Enero 2009 |
| Complejo Jusepin | Localidad Jusepin - Edo. Monagas | Septiembre 2008 |
| Tren de pruebas y flujo total, Makolla K203 | División Faja Distrito Morichal | Octubre 2008 |
| Estacione de flujo de Muri | Localidad de Pinta Mata, Distrito Norte - Edo. Monagas | Septiembre 2008 |
| Estación de flujo de Orocual | Localidad Orocual, Distrito Norte - Edo. Monagas | Septiembre 2008 |
| José | Puerto La Cruz | Octubre 2008 |
| Patio de tanques Silvestre | Localidad San Silvestre - Edo. Barinas | Enero 2009 |
| Patio de tanques Punta de Palmas | Punta de Palmas | Diciembre 2008 |
| Patio de Tanques Boscan | Maracaibo – Estado Zulia | Diciembre 2008 |
| Estación de flujo de Santa Bárbara | Municipio Ezequiel Zamora – Estado Monagas | Marzo 2009 |
| Estación de flujo Aragua I | Maracay – Estado Aragua | Marzo 2009 |

**Tabla 10 - Principales pilotos del SCADA GALBA**

Por último es importante destacar, que el conjunto de productos analizados en este trabajo forman parte de anexos que han aportado al país un monto de 1 865 222 dólares, pactados en diferentes contratos pertenecientes a los dos Convenios Marco PDVSA – ALBET. A continuación se muestra la tabla resumen de los anexos en cuestión:

| Número | Nombre | Monto Total |
|---|---|---|
| **Primer Convenio Marco PDVSA-ALBET, S.A** | | |
| Anexo 13 | Desarrollo del Subsistema de Drivers del SCADA Nacional | USD 417,422.00 |
| Anexo 31 | Proyecto de Desarrollo del Subsistema de Drivers del SCADA Nacional. (Fase II) | USD 226,720.00 |
| **Segundo Convenio Marco PDVSA-ALBET, S.A** | | |
| Anexo 12 | Proyecto de Extensiones al subsistema de Históricos de la versión 2.0 del Guardián del ALBA. | USD 326,296.00 |
| Anexo 14 | Proyecto de Extensiones al subsistema de Drivers de la versión 2.0 del Guardián del ALBA. | USD 315,408.00 |
| Anexo 17 | Proyecto de Desarrollo de la Interfaz Genérica 5.0 de los Drivers de la versión 2.0 del Guardián del ALBA. | USD 326,704.00 |
| Anexo 20 | Proyecto de Extensiones al subsistema de Drivers de la versión 2.0 del Guardián del ALBA | USD 252,672.00 |

**Tabla 11 - Anexos del SCADA GALBA relacionados con el trabajo de diploma.**

## Conclusiones

Las herramientas y módulos de software analizados en este material constituyen una plataforma de componentes reutilizables para el desarrollo y el mantenimiento de nuevos manejadores de dispositivos, dicha plataforma está constituida por la biblioteca DriverCore y la biblioteca TransportProvider. La empresa petrolera de Venezuela cuenta hoy con los manejadores de la familia Modbus (RTU, ASCII y TCP), así como el manejador Ethernet/IP para la comunicación con los dispositivos de campo existentes en sus instalaciones. Además cuenta con una herramienta para brindar mantenimiento y realizar pruebas a los manejadores instalados.

Por último señalar que los resultados que ofrecen las pruebas realizadas al algoritmo de compresión muestran que proporciona altas tasas de compresión y al propio tiempo buena calidad en la reconstrucción. Posibilita además incrementar las tasas de compresión sin necesidad de reprocesar la información, lo que permite disminuir el nivel de detalle en los datos sin aumentar significativamente el procesamiento necesario.

## Referencias

**Bader , F. P. y Tucker, T. W. 1987.** *Real-Time Data Compression Improves Plant Performance Assessment.* s.l. : InTech, 1987. págs. 53-56.

**Bristol, E. H. 1990.** *Swinging Door Trending: Adaptive Trend Recording.* s.l. : ISA National Conf. Proc.,, 1990. págs. 749-754.

**Cárdenas-Barrera, J., Lorenzo-Ginori, J. y Rodríguez Valdivia, E. 2001.** *Algoritmo basado en wavelets packet para la compresión de electroencefalogramas.* Habana : Memorias II Congreso Latinoamericano de Ingeniería Biomédica, 2001.

**Eclipse Foundation. 2010.** 2010.

**García, Luis Enrique. 2009.** *Framework para el desarrollo de manejadores de dispositivos para el proyecto SCADA Guardián del ALBA.* Ciudad de la Habana : s.n., 2009.

**Gómez Johnson, Rubén. 2008.** *Capa de acceso a datos síncrona y asíncrona para dispositivos Modbus TCP.* 2008.

**Grupo de Desarrollo EROS. 2002.** *Manual del Usuario del EROS versión 5.0.* 2002.

**Hale, J. C. y Sellars, H. L. 1981.** *Historical Data Recording for Process Computers.* s.l. : CEP, 1981. págs. 38-43.

**Herrera, Moisés. 2008.** *Recolección y Manejadores en el Guardián del ALBA.* 2008.

*Introducción a Ethernet Industrial.* **2005.** 2005.

**Kennedy , J. P. 1988.** *Data Storage for CIM.* s.l. : Process Engineering, 1988. págs. 31-35.

**Lu, Zhitao, Kim Dong, Youn y Pearlman, W. 2000.** *Wavelet Compression of ECG signals by the Set Partitioning in Hierarchical Trees Algorithm.* s.l. : IEEE Transactions on Biomedical Engineering, 2000. Vol. 47.

**Modbus IDA. 2004.** Modbus IDA. *Modbus Messaging On Tcp/Ip Implementation Guide V1.0a.* [En línea] Junio de 2004. http://www.Modbus-IDA.org.

**—. 2006.** Modbus IDA. *Modbus Application Protocol Specification V1.1b.* [En línea] Diciembre de 2006. http://www.Modbus-IDA.org.

**MODICON. 1996.** 1996.

**Nokia Corporation. 2010.** 2010.

**Open DeviceNet Vendor Assoc. 2001.** EtherNet/IP Specification, Preliminary Release 8. [En línea] Marzo de 2001. http://www.odva.org.

**Organización Internacional de Normalización. 2004.** *ISO 7498-1.* 2004.

**Pozo, Antonio Cedeño. 2009.** *Módulos de adquisición y análisis para la interacción con dispositivos de campo de un SCADA.* Ciudad de la Habana : s.n., 2009.

**Rockwell Automation. 2000.** Logix5000 Data Access. [En línea] Marzo de 2000. http://www.ab.com. 1756-RM005A-EN-E.

**Said, A. y Pearlman, W. 1996.** *A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees.* s.l. : IEEE Transactions on Circuits and Systems for Video Technology, 1996. Vol. 6.

**Sweldens, W. 1997.** *The lifting scheme: A construction of second generation wavelets.* s.l. : Siam J. Math. Anal, 1997. págs. 511-546. Vol. 29.

**Tao, Li, y otros. 2003.** *A Survey on Wavelet Applications in Data Mining.* s.l. : SIGKDD Explorations, 2003. Vol. 4.

**Thornhill, N., Shoukat, C. y Shah, S. 2004.** The impact of compression on data-driven process analyses. *Journal of Process Control.* 2004, 14, págs. 389-398.

**Trujillo Codorniu, Rafael Arturo y Pérez Pérez, Yosep Yasmany. 2008.** *Compresión de datos históricos mediante la transformada Wavelet.* Ciudad de la Habana : s.n., 2008.

**Trujillo Codorniu, Rafael Arturo, Cedeño Pozo, Antonio y García Hernández, Luis Enrique. 2008.** *Interfaz Genérica para los Manejadores de Dispositivos en el proyecto SCADA GALBA.* Ciudad de La Habana : s.n., 2008.