Universidad Politécnica de Cartagena

industriales

etsii UPCT

IMPLEMENTACION DE PROTOCOLO DE COMUNICACIONES MODBUS/TCP PARA LINUX EN LENGUAJE C++. APLICACIÓN SOBRE ANALIZADORES DE REDES SIEMENS SENTRON PAC4200

Titulación: ITI esp Electrónica Industrial
Intensificación:
Alumno/a: Roberto Sánchez Llamas
Director/a/s: Manuel Jiménez Buendía

Cartagena, 27 de julio de 2012

Autor	Roberto Sánchez Llamas
E-mail del Autor	rsanchez1985@hotmail.com
Director(es)	Manuel Jiménez Buendía
E-mail del Director	manuel.jimenez@upct.es
Codirector(es)	José Ángel Lajarín Barquero
Título del PFC	**IMPLEMENTACION DE PROTOCOLO DE COMUNICACIONES MODBUS/TCP PARA LINUX EN LENGUAJE C++.** **APLICACIÓN SOBRE ANALIZADORES DE REDES SIEMENS SENTRON**
Descriptores	

Resumen

Se implementará el protocolo de comunicaciones MODBUS-TCP en lenguaje C++ para la recogida de datos desde un PC industrial de n equipos SENTRON PAC4200 conectados a la red industrial Ethernet, todo ello sobre plataforma LINUX, siguiendo así la arquitectura del SCADA implantado en la fábrica.

En dicha arquitectura, los PCs industriales (LINUX) establecen la comunicación con los dispositivos de campo (PLCs, analizadores de redes, dispositivos de temperaturas, servos, periferias distribuidas…) con los protocolos de comunicaciones particulares de cada dispositivo (MODBUS, PROFIBUS, CAN…), y envían la información recogida al nivel jerárquico superior a través de la red Ethernet, implementado un servidor SNMP. El nivel superior lo compone un servidor web TOMCAT sobre LINUX, que mediante servlets JAVA implementa el cliente SNMP quien interroga a los PCs industriales. Los datos recogidos son procesados y almacenados en servidor de base de datos ORACLE, también sobre LINUX. El interfaz con el usuario se hace a través de un navegador web común, mediante Applets JAVA que sirve el servidor TOMCAT, poniendo a disposición del usuario los datos en tiempo real de los procesos de campo, así como valores históricos de los mismos.

Titulación	Ingeniero Técnico Industrial esp. Electrónica Industrial
Intensificación	
Departamento	UPCT/Dpto. Tecnología Electrónica
Fecha de Presentación	27/7/2012

Agradecimientos

En primer lugar quisiera agradecer a José Ángel Lajarín por darme la oportunidad que me ha brindado de realizar este proyecto para ElPozo Alimentación.

A mis padre y mi madre, por su apoyo y confianza, y por supuesto a mi familia y amigos, mi hermano, abuelos y tíos, por estar siempre cuando los necesite

A mis compañeros de la oficina de Ingeniería de Sistemas, en especial a José Luis y Juanma, y a mi profesor Manuel Jiménez por su grandísima ayuda resolviendo las dudas que fueron surgiendo en este proyecto, sin ellos no podría haber sido posible terminarlo.

Índice de general

Índice de Tablas

Índice de Figuras

1

Contexto y objetivos

En este capítulo se hará una breve descripción de los objetivos de este proyecto, así como de sus antecedentes y la necesidad de implantación en el sistema SCADA actualmente operativo en la empresa.

1.1 Contexto del proyecto. Estructura del SCADA actual

ElPozo Alimentación es una empresa líder en la elaboración de alimentos con base cárnica comprometida con sus consumidores y clientes, que apuesta por la tecnología de última generación. Su principal fortaleza es contar con un Control Integral de Proceso (CIP) que le permite responder a las necesidades de los consumidores ofreciéndole alimentos que superen sus expectativas.

Para el control de las instalaciones y procesos, ElPozo Alimentación ya dispone en el SCADA de información recogida de analizadores de redes instalados, mediante protocolos de comunicaciones MODBUS y PROFIBUS sobre buses de campo serie; en concreto, los equipos SMART Piú del fabricante DUCATI Energía s.p.a. con protocolo MODBUS, y los SIMEAS P de SIEMENS con protocolo PROFIBUS. ElPozo pretende estandarizar los analizadores de redes existentes así como ampliar el número de los mismos a un único modelo, el SENTRON PAC4200 de SIEMENS, utilizando la red Ethernet industrial existente como bus de campo y el protocolo de comunicaciones Modbus/TCP.

En este proyecto se implementará el protocolo de comunicaciones Modbus/TCP en lenguaje C+ para la recogida de datos desde un PC industrial de una serie de equipos Sentron PAC-4200 conectados a la red industrial Ethernet, todo ello sobre plataforma Linux, siguiendo así la arquitectura del SCADA implantado en la fábrica, cuya estructura se puede observar en la Figura 1.1.

En dicha arquitectura, los PCs industriales (con sistema operativo Linux) establecen la comunicación con los dispositivos de campo (PLCs, analizadores de redes, dispositivos de temperaturas, servos,…) con los protocolos de comunicaciones particulares de cada dispositivo (Modbus, Profibus, Can,..) y envían la información recogida al nivel jerárquico superior a través de la red Ethernet, implementando un servidor SNMP. El nivel superior lo compone un servidor web Tomcat sobre Linux, que mediante servlets Java implementa el cliente SNMP que interroga a los PCs industriales. Los datos recogidos son procesados y almacenados en servidor de base de datos Oracle, también sobre Linux.

El interfaz con el usuario se hace a través de un navegador web común, mediante Applets Java que sirve el servidor Tomcat, poniendo a disposición del usuario los datos en tiempo real de los procesos de campo, así como valores históricos de los mismos.

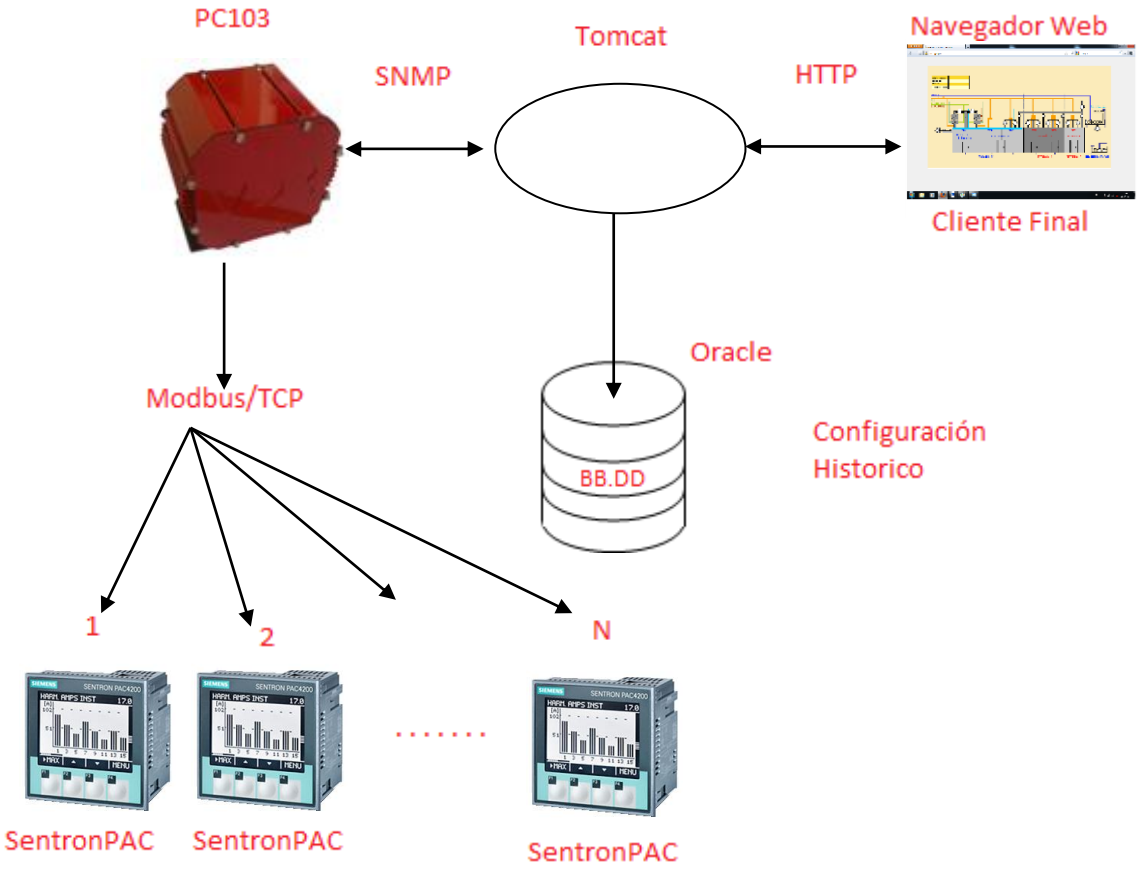

Figura 1.1. Estructura de la aplicación

1.2 Objetivos

Se trata de realizar un protocolo estándar Modbus en su versión TCP que pueda realizar las funciones de lectura y escritura de registros sea cual sea el aparato, siempre que funcione con dicha tecnología.

Para ello se implementarán unas librerías en C++ que sean capaces de interactuar con los dispositivos que usen dicho protocolo y con el sistema SCADA actual.

La programación de los PCs industriales constará de la creación de dos aplicaciones, una de ellas se encargará de recibir los mensajes del Servlet Oracle, la segunda aplicación realizará testeos de la información requerida por el Servlet y almacenarlo en unos archivos, de forma que cuando el Servlet haga una petición de los registros, estos estén siempre actualizados, para garantizar la mejor monitorización posible.

En este caso, mediante los analizadores de redes SentronPack4200, la empresa obtendrá la información de los diferentes consumos y potencias requeridas en la planta, pudiendo hacer una monitorización de los mismos, y por lo tanto, la posibilidad de realizar mejoras del rendimiento y consumo eléctrico de la planta.

El caso general de este proyecto es no solo centrarse en los analizadores de redes, sino en cualquier aparato que utilice dicho protocolo, con lo que también permitirá renovar las instalaciones antiguas, simplificando, unificando y modernizando el sistema de adquisición de datos.

Para la realización de este proyecto se deberán de realizar una investigación sobre el protocolo Modbus/TCP, para que este pueda ser implementado en lenguaje C++, así como del funcionamiento de los dispositivos, ya sea para acceso a las áreas de memoria y su posible modificación.

Esto permitirá adquirir un conocimiento amplio del lenguaje C++, un manejo intensivo del sistema operativo Linux y una puesta en práctica de los conocimientos sobre estos.

2

Arquitectura de la solución

En este capítulo se realizará una breve descripción del protocolo Modbus y su variante Modbus/TCP, así como de los diferentes componentes de los que consta este proyecto.

2.1 Protocolo Modbus/TCP

2.1.1 Descripción

Modbus es un protocolo de comunicaciones publicado por Modicon en 1979, diseñado para funcionar con equipos industriales tales como Controladores Lógicos Programables (PLCs), Computadoras, Motores, Sensores, y otros tipos de dispositivos físicos de entrada/salida.

Simple y robusto, se ha convertido en un protocolo estándar de comunicación, siendo uno de los más comunes en industria para la comunicación de dispositivos electrónicos.

Las principales razones por las que se ha extendido tanto el uso de este protocolo son las siguientes:

- Es público.
- Su implementación es sencilla y requiere tiempos de desarrollo reducidos.
- Maneja bloques de datos sin suponer restricciones.

Modbus/TCP fue introducido por Schneider Automation como una variante de la familia Modbus destinado a la supervisión y el control de equipos de automatización.

Específicamente, el protocolo cubre el uso de mensajes Modbus en un entorno Intranet o Internet usando los protocolos **TCP/IP**[1].

La especificación Modbus/TCP define un estándar interoperable en el campo de la automatización industrial, el cual es simple de implementar para cualquier dispositivo que soporta **sockets**[2] TCP/IP.

[1]Parte de esta sección esta tomado de www.modbusida.com

[2]Un socket es una abstracción proporcionada por el sistema operativo que permite a un programa de aplicación acceso a los protocolos TCP/IP.

2.1.1.1 Orientado a conexión

Modbus es un protocolo de comunicación .sin estado., es decir, cada solicitud del maestro es tratada independientemente por el esclavo y es considerada una nueva solicitud no relacionada a las anteriores, de esta forma haciendo a las transacciones de datos altamente resistentes a rupturas debido a ruido y además requiriendo mínima información de recuperación para ser mantenida la transacción en cualquiera de los dos terminales.

Las operaciones de programación, por otro lado, esperan una comunicación orientada a la conexión, es decir, las máquinas de origen y de destino establecen un canal de comunicaciones antes de transferir datos. Este tipo de operaciones son implementadas de diferentes maneras por las diversas variantes de Modbus (Modbus RTU, Modbus ASCII, Modbus PLUS).

Modbus/TCP maneja ambas situaciones. Una conexión es inicialmente establecida en esta capa de protocolo (nivel de aplicación), y esa conexión única puede llevar múltiples transacciones independientes.

En adición, TCP permite establecer un gran número de conexiones concurrentes, de este modo el cliente (maestro) puede ya sea re-usar una conexión previamente establecida o crear una nueva, en el momento de realizar una transacción de datos.

Es interesante analizar por qué se utiliza el protocolo TCP orientado a conexión en vez del protocolo **UDP**[3]orientado a datagramas. La principal razón es mantener control de una transacción individual encerrándola en una conexión la cual pueda ser identificada, supervisada, y cancelada sin requerir acción específica de parte de las aplicaciones cliente y servidor. Esto da al mecanismo una amplia tolerancia a cambios

[3]El UDP (User Datagram Protocol) proporciona un servicio de entrega sin conexión, utilizando el IP para transportar mensajes entre máquinas. Emplea el IP para llevar mensajes, pero agrega la capacidad para distinguir entre varios destinos dentro de una máquina host.

del desempeño de la red, y permite que herramientas de seguridad tal como **firewalls**[4]y proxies puedan ser fácilmente añadidos.

2.1.1.2 Codificación de datos

Modbus usa una representación **big-endian**[5] para direcciones y datos. Esto significa que cuando una cantidad numérica más grande que un byte es transmitido, el byte más significativo es enviado primero. Así, por ejemplo:

0x1234 será 0x12 0x34

2.1.1.3 Interpretación del modelo de datos

Modbus basa su modelo de datos sobre una serie de tablas las cuales tienen características distintivas. Las cuatro principales son:

- Entradas discretas. Bit simple, suministrado por un sistema I/O, de solo lectura.
- Salidas discretas. Bit simple, alterable por un programa de aplicación, de lectura-escritura.
- Registros de entrada. Cantidad de 16 bits, suministrado por un sistema I/O, de solo lectura.
- Registros de salida. Cantidad de 16 bits, alterable por un programa de aplicación, de lectura-escritura.

La distinción entre entradas y salidas, y entre datos direccionables al bit y direccionables a la palabra, no implica algún comportamiento de la aplicación. Es aceptable y común, considerar las cuatro tablas sobrelapando una con otra, si esta es la interpretación más natural sobre la máquina (esclavo Modbus) en cuestión.

[4]Un firewall (cortafuegos) es una configuración de encaminadores y redes colocados entre la organización interna de una red y su conexión con redes externas a fin de proporcionar seguridad.

[5]Big-endian es un formato en el cual el byte más significativo se encuentra primero.

2.1.1.4 Longitud implícita en el protocolo.

Todas las solicitudes y respuestas Modbus están diseñadas en tal forma que el receptor puede verificar que un mensaje está completo. Para códigos de función donde la solicitud y respuesta son una longitud fija, el código de función solo es suficiente. Para códigos de función llevando una cantidad variable de datos en la solicitud o respuesta, la porción de datos estará precedida por un campo que representa el número de bytes que siguen.

Cuando Modbus es llevado sobre TCP información de longitud se adiciona en el prefijo (o encabezado) para permitir al receptor reconocer los límites del mensaje, igual si el mensaje ha sido dividido en múltiples paquetes para la transmisión. La existencia de reglas de longitud implícitas o explícitas, y el uso de un código de chequeo de error **CRC-32**[6](sobre Ethernet) resulta en una probabilidad muy pequeña de corrupción no detectada sobre un mensaje de solicitud o respuesta.

2.1.2 Ventajas del protocolo Modbus/TCP

A continuación se detallan las principales ventajas que implica la utilización de la variante Modbus/TCP

- Es escalable en complejidad. Un dispositivo el cual tiene solo un propósito simple necesita solo implementar uno o dos tipos de mensaje.
- Es simple para administrar y expandir. No se requiere usar herramientas de configuración complejas cuando se añade una nueva estación a una red Modbus/TCP.
- No es necesario equipo o software propietario de algún vendedor. Cualquier sistema computador o microprocesador con una pila de protocolos TCP/IP puede usar Modbus/TCP.
- Puede ser usado para comunicar con una gran base instalada de dispositivos Modbus, usando productos de conversión los cuales no requieren configuración.

[6]CRC (CyclicRedundancyCode), verificación por redundancia cíclica.

- Es de muy alto desempeño, limitado típicamente por la capacidad del sistema operativo del computador para comunicarse. Altas ratas de transmisión son fáciles de lograr sobre una estación única, y cualquier red puede ser construida para lograr tiempos de respuesta garantizados en el rango de milisegundos.

2.1.3 Estructura del protocolo

A continuación se describe la forma general de encapsulación de una solicitud o respuesta Modbus cuando es llevada sobre una red Modbus/TCP. Es importante anotar que la estructura del cuerpo de la solicitud y respuesta, desde el código defunción hasta el fin de la porción de datos, tiene exactamente la misma disposición y significado como en las otras variantes Modbus, tal como:

- Modbus serial. codificación ASCII
- Modbus serial. codificación RTU
- Modbus PLUS

Las únicas diferencias en esos otros casos son la especificación de los delimitadores inicial y final del **mensaje**[7], el patrón de chequeo de error y la interpretación de la dirección.

Todas las solicitudes son enviadas vía TCP sobre el puerto registrado 502. Las solicitudes normalmente son enviadas en forma **half-duplex**[8]sobre una conexión dada. Es decir, no hay beneficio en enviar solicitudes adicionales sobre una única conexión mientras una respuesta está pendiente. Sin embargo, los dispositivos que desean obtener altas ratas de transferencia pueden establecer múltiples conexiones TCP al mismo destino.

El campo dirección esclavo de Modbus es reemplazado por un byte identificador de unidad, el cual puede ser usado para comunicar a través de dispositivos tales como puentes y gateways, los cuales usan una dirección IP única para soportar múltiples unidades terminales independientes.

[7]En MODBUS esto se denomina Framing.

[8]En half-duplex los datos pueden viajar en cualquier dirección, pero no en forma simultánea.

Los mensajes de solicitud y respuesta en Modbus/TCP poseen un prefijo o encabezado compuesto por seis bytes como se aprecia en la **Tabla 2.1**

Ref	Ref	00	00	00	Len

Tabla 2.1. Prefijo Modbus/TCP

El .Ref Ref anterior son los dos bytes del campo .referencia de transacción., un número que no tiene valor en el servidor pero son copiados literalmente desde la solicitud a la respuesta a conveniencia del cliente. Este campo se utiliza para que un cliente Modbus/TCP pueda establecer simultáneamente múltiples conexiones con diferentes servidores y pueda identificar cada una de las transacciones.

El tercer y cuarto campo del prefijo representan el identificador de protocolo., un número el cual debe ser establecido a cero.

El len especifica el número de bytes que siguen. La longitud es una cantidad dedos bytes, pero el byte alto se establece a cero ya que los mensajes son más pequeños que 256.

De esta forma, un mensaje Modbus/TCP completo posee una estructura como se muestra en la Tabla 2.2

Byte 0	Identificador de transacción. Copiado por el servidor, normalmente 0.
Byte 1	Byte 1 Identificador de transacción. Copiado por el servidor, normalmente 0.
Byte 2	Byte 2 Identificador de protocolo = 0.
Byte 3	Byte 3 Identificador de protocolo = 0.
Byte 4	Byte 4 Campo de longitud (byte alto) = 0. Ya que los mensajes son menores a 256.
Byte 5	Byte 5 Campo de longitud (byte bajo). Número de bytes siguientes.
Byte 6	Byte 6 Identificador de unidad (previamente dirección esclavo.).
Byte 7	Byte 7 Código de función Modbus.
Byte 8 y más	Byte 8 y más Los datos necesarios.

Tabla 2.2. Estructura de mensajes en Modbus/TCP

2.1.4 Esquema de encapsulación

Modbus/TCP básicamente encapsula un marco Modbus dentro de un marco TCP visto de manera simple en lo mostrado en la Tabla 2.2

2.1.5 Conformación de Clases

Modbus por su naturaleza es ya implementada en muchísimos lugares, por tanto se debe evitar una ruptura de las implementaciones existentes. De esta forma el conjunto de los tipos de transacción Modbus existente ha sido clasificado en clases, donde el nivel 0 representa funciones que son universalmente implementadas y totalmente consistentes, y el nivel 2 representa funciones útiles pero algo dependientes del esclavo. Esas funciones del conjunto, las cuales no son convenientes por interoperabilidad son también identificadas.

Debe anotarse que futuras extensiones al estándar pueden definir códigos de función adicionales para manejar situaciones donde el estándar existente es deficiente.

2.1.5.1 Comandos Clase 0

Este es el mínimo conjunto útil de funciones, tanto para el maestro como para el esclavo.

03	Leer múltiples registros holding[9].
16	Escribir múltiples registros holding.

Tabla 2.3. Comandos de la Clase 0

2.1.5.2 Comandos Clase 1

Este es el conjunto adicional de funciones, el cual es comúnmente implementado e interoperable. Como fue explicado antes, muchos esclavos deciden tratar entradas, salidas, registros, y valores discretos como equivalentes.

[9]En el protocolo MODBUS , .holding register. representa una cantidad de 16 bits, la cual representa una posición interna de la memoria.

01	Leer estado de salidas discretas.
02	Leer estado de entradas discretas.
04	Leer registros de entrada.
05	Forzar una salida discreta.
06	Prefijar un registro holding único.
07	Leer estados de excepción*.

** Esta función típicamente tiene un significado diferente para cada familia de esclavos.*

Tabla 2.4. Comandos de la Clase 1

2.1.5.3 Comandos Clase 2

Estas son las funciones de transferencia de datos necesarias para operaciones de rutina tal como supervisión y HMI[10].

15	Fijar múltiples salidas discretas.
20	Leer referencia general*.
21	Escribir referencia general*.
22	Enmascarar registro de escritura.
23	Leer/escribir registros**.
24	Leer cola FIFO***.

** Esta función será la más apropiada para manejar grandes espacios de registros y datos, los cuales carecen de números de referencia.*

*** Esta función permite la entrada y salida de un rango de registros como una transacción única. Es la forma más eficiente usando Modbus para desempeñar un intercambio regular de datos tal como con un módulo I/O.*

**** Una función algo especializada, destinada a permitir la transferencia de datos desde una tabla estructurada como una FIFO a un computador.*

Tabla 2.5. Comandos de la Clase 2

2.1.5.4 Comandos específicos de la máquina/red/vendedor

Todas de las siguientes funciones, aunque mencionadas en los manuales del protocolo Modbus, no son apropiadas para propósitos de interoperabilidad porque son dependientes de la máquina, un ejemplo de esto puede verse en la siguiente tabla.

[10]HMI : Human Machine Interface.

08	Pruebas de diagnóstico.
09	Programación.
10	Completar la programación.
11	Leer la palabra de estado del contador de eventos.
12	Leer el registro de eventos de comunicación.
13	Programación.
14	Completar la programación.
17	Reportar ID del esclavo.
18	Programación.
19	Reinicializar enlace de comunicaciones.
125	Sustitución de firmware.
126	Programación.
127	Reportar dirección local.

Tabla 2.6. Funciones dependientes de la máquina

2.2 El Protocolo SNMP

2.2.1 Introducción

La complejidad y variedad de las redes de comunicaciones actuales hace que la gestión adecuada de las mismas sea una tarea básica que hay que afrontar tanto en la planificación de las redes como en su mantenimiento. En una red de comunicaciones común, una red corporativa por ejemplo, no demasiado compleja, suele aparecer una enorme diversidad de dispositivos: estaciones, servidores, concentradores, conmutadores, routers, impresoras, etc. Es necesario gestionar el funcionamiento de los dispositivos de manera que no sólo funcionen, sino que lo hagan de manera óptima. Hay que dedicar recursos a esta tarea. El objetivo de la gestión de red no es el de solucionar los fallos que se produzcan en nuestro sistema sino el de evitar en la medida de lo posible que se produzcan estos fallos.

Simple Network Management Protocol (SNMP) apareció para responder a esta necesidad: la de poder gestionar una red fácilmente, automáticamente y de manera estándar. Su predecesor (SGMP) era un protocolo que permitía la gestión de routers exclusivamente. SNMP permite la gestión de dispositivos en general: sistemas Linux y Windows, impresoras, fuentes de alimentación, etc.

El núcleo de SNMP lo forma un conjunto de operaciones sencillas y la información que esas operaciones recopilan, que permiten al administrador tanto conocer el estado de un dispositivo como cambiar ese estado. Por ejemplo, se puede saber si la interfaz de un router está activa y, en su caso, activarla o desactivarla. El impacto de las operaciones de gestión en la carga de la red debe ser mínimo.

2.2.2 Simple Network Management Protocol

El núcleo de SNMP es un conjunto de especificaciones que describen qué información debe ofrecer un dispositivo, cómo se debe almacenar esa información y cómo se puede acceder y modificar esa información. En esta sección se describirá qué representan y cómo se llaman las diferentes entidades que forman SNMP y cuál es la arquitectura de un sistema que use SNMP.

SNMP consta de tres partes:

- Una base de datos de información de gestión (Management Información Base, MIB).
- Un conjunto de estructuras comunes y un esquema de identificación de variables. La estructura de la información de gestión (Structure of Management Information, SMI).
- El protocolo entre las entidades gestionadas y la estación de gestión, llamado SNMP.

2.2.2.1 RFC y versiones de SNMP

SNMP está especificado por el IETF como una serie de Internet Standards. SNMP ha seguido una evolución más o menos complicada y en la actualidad podemos encontrar los siguientes documentos:

- SNMP Versión 1 (SNMPv1). Es el primer estándar de SNMP. Definido en la RFC 1157. Tiene carencias en cuestiones de seguridad.
- SNMP Versión 2 (SNMPv2). Es una versión experimental que nunca ha llegado a ser estándar. Aun así varios vendedores la han implementado.
- SNMP Versión 3 (SNMPv3). La última versión del estándar. Añade soporte para seguridad: cifrado, autenticación. Cambia la arquitectura respecto a la primera versión. Especificado en las RFC 3410-3418.

2.2.2.2 Managers y agentes

Hasta SNMPv3, la especificación distinguía dos tipos de entidades que participaban en el modelo de gestión: managers y agentes.

Un manager (Network Management Stations, NMS), es el equipo que gestiona la red, es decir, un servidor que ejecuta un software capaz de realizar tareas de gestión de red. Hay que resaltar que SNMP se usa para realizar estas tareas de manera estándar, pero que por sí solo no realiza ninguna tarea de gestión, es decir, habrá una aplicación que use SNMP para realizar estas tareas.

Un manager es capaz de muestrear los dispositivos y de recibir traps de ellos. Muestrear un dispositivo consiste en pedirle cierta información. El manager también puede ordenar a un dispositivo que varíe el estado del mismo.

Un trap es un aviso que lanza un agente asíncronamente a un manager para informarle de algún suceso que ha ocurrido.

Un agente es un programa que se ejecuta en el dispositivo que se quiere gestionar. Puede ser un programa separado o formar parte del sistema operativo. La mayoría de los dispositivos de redes de comunicaciones incorporan un agente SNMP. El agente proporciona información de gestión al manager. El agente comprueba el funcionamiento de diversos aspectos del dispositivo. Por ejemplo, en un router del agente comprueba que interfaces están activadas y cuáles no. El manager puede pedir información al agente sobre el estado de estas interfaces y llevar a cabo las acciones apropiadas. Además, en caso de que se configure para ello, el agente puede enviar un trap al manager cuando se produzca un suceso determinado (véase la figura 2.1).

Figura 2.1. Funcionamiento SNMP.

En resumen, SNMP distingue dos entidades en un sistema: los agentes, que guardan información del estado de los dispositivos y los managers, que gestionan la red. Los managers piden información a los agentes y pueden ordenarles que modifiquen el estado de un dispositivo. Los agentes pueden enviar avisos a los managers cuando se producen determinados sucesos. A no ser que el agente se configure para enviar traps, nunca se conectará al manager, es decir, es el manager siempre el que inicia una conexión.

2.2.2.3 La estructura de la información de gestión y las MIB

La estructura de la información de gestión (Structure of Management Information, SMI) proporciona una manera de definir los objetos (variables) gestionados y su comportamiento. Un agente tiene una lista de los objetos que controla, Esta lista determina en conjunto la información que un NMS puede utilizar. Estos objetos pueden ser el estado operacional de una interfaz de un router o el número de tramas Ethernet descartadas por la interfaz. La SMI es un conjunto de estructuras comunes y un esquema para identificar variables en la MIB. Por ejemplo, SMI especifica que un counter es un entero no negativo cuyo valor varía entre 0 y 4.294.967.295 y que al llegar al valor final vuelve a 0.

La base de datos de información de gestión (Management Information Base, MIB) es una base de datos de los objetos que el agente gestiona. Cualquier tipo de información de estado o estadísticas que un NMS gestiona debe estar en esta base de datos, en la MIB.

SMI proporciona una forma de definir objetos que se pueden gestionar, mientras que la MIB es la definición de los propios objetos (usando la sintaxis que específica SMI). La MIB es como un diccionario: da un nombre al objeto gestionado y explica su significado. Hay diferentes MIB para diferentes sistemas. Por ejemplo, podemos tener una MIB para ATM, que definirá los objetos que se pueden gestionar en los dispositivos que usen esta tecnología.

Un agente siempre implementa (puede gestionar) la MIB-II (RFC 1213). Esta base de datos estándar define variables para cosas tales como estadísticas de interfaz (velocidad de interfaz, MTU, octetos enviados, octetos recibidos, etc.) y otras relacionadas con el propio sistema (tipo de sistema operativo, localización, etc.). El objetivo es proporcionar información general sobre TCP/IP. Pero la MIB-II no puede definir todo el tipo de información que cada dispositivo puede ofrecer. Por ejemplo, un conmutador ofrecerá estadísticas específicas sobre sus puertos, soporte para VLAN, etc. Por ello, hay multitud de MIB para diversos dispositivos: hay MIB estándar especificadas en sus correspondientes RFC para varias tecnologías (ATM, FrameRelay, DNS) y, lo más importante, un vendedor puede definir su propia MIB para sus

dispositivos (MIB propietaria), de manera que ofrezca la información que considere interesante para la gestión de su dispositivo.

2.2.2.4 El protocolo SNMP

SNMPv1 define cinco tipos de mensaje que se pueden intercambiar entre un manager y un agente. Las siguientes versiones añaden nuevos tipos de mensaje. Las PDU (unidades de datos del protocolo) definidas son las siguientes:

- Operador get. Pide el valor de una o más variables.
- Operador next. Pide el valor de la siguiente variable después de una o más variables especificadas. Más adelante explicaremos el significado de "siguiente".
- Operador set. Configura el valor de una o más variables.
- Operador get-response. Devuelve el valor de una o más variables. Es el mensaje devuelto por un agente en respuesta a un mensaje get, set o get-next.
- Operador trap. Notifica al manager la ocurrencia de un suceso.
- Operador get-bulk (SNMPv2 y SNMPv3). Permite recuperar el valor de una sección grande de una tabla en una única petición.
- Operador notification (SNMPv2 y SNMPv3). Las siguientes versiones, para unificar el formato de PDU de SNMP, definieron este tipo de mensaje que es totalmente equivalente a un trap.
- Operador inform (SNMPv2 y SNMPv3). Mensaje que permite la comunicación manager a manager.
- Operador report (SNMPv3). Es un mecanismo que permite la comunicación entre motores SNMP (SNMP engines). Este es un concepto de SNMPv3. Permite la notificación de errores en el procesado de los mensajes.

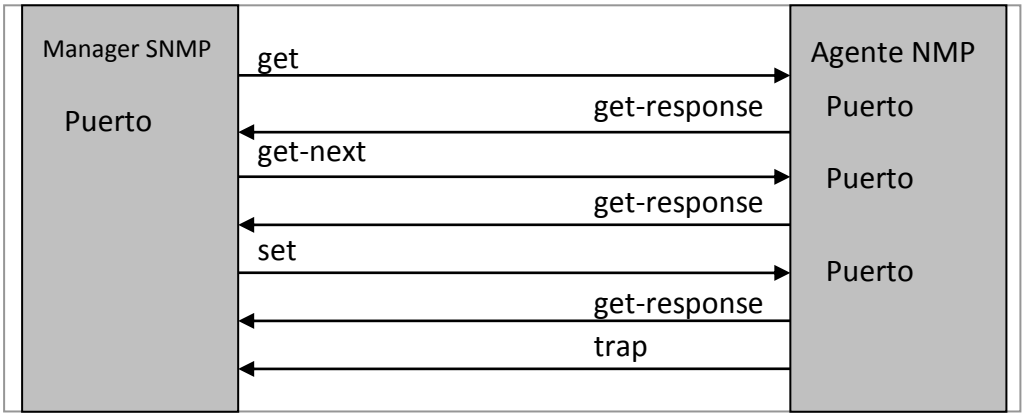

Figura 2.2. **Mensajes SNMP.**

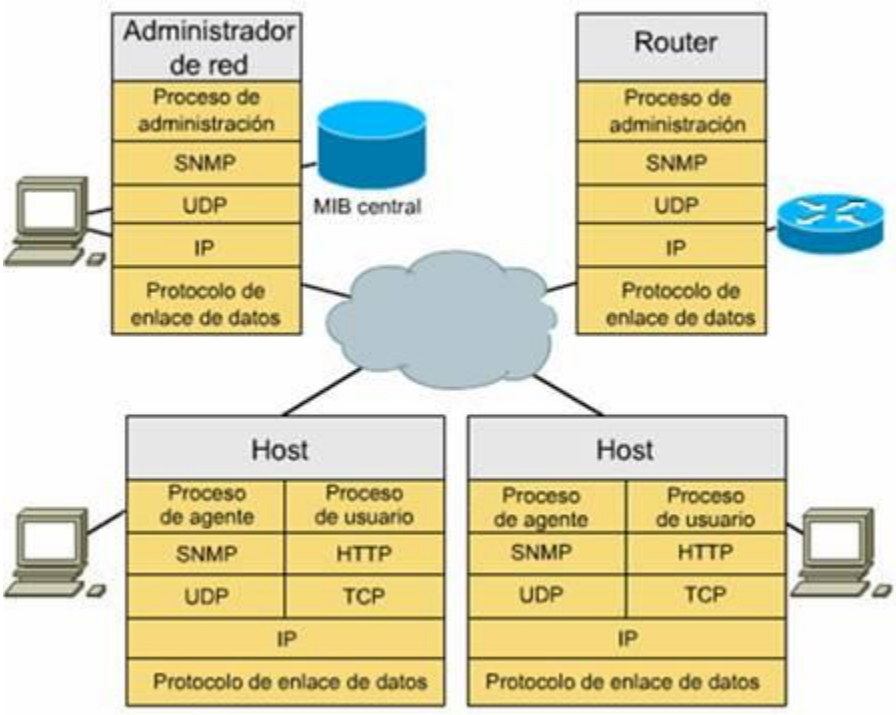

Figura 2.3. **Elementos del sistema SNMP**

La Figura 2.2 muestra el intercambio de algunos de los mensajes SNMP. Los primeros tres mensajes se envían del manager al agente. El último es enviado desde el agente al manager. SNMP es un protocolo de nivel de aplicación. SNMP utiliza UDP como capa de transporte. Los mensajes del manager al agente y sus respuestas usan el puerto 161 de UDP. Los traps usan el puerto 162 de UDP. Las peticiones SNMP por tanto no son fiables. Debe ser la capa de aplicación la que se encargue de recuperar las pérdidas. Normalmente el manager implementa una serie de temporizadores y si vencen antes de recibir respuesta, se produce una retransmisión.

Por tanto, el funcionamiento de SNMP se basa en el sondeo (polling) de los dispositivos por parte del manager (consulte la Figura 2.3). El manager envía paquetes get para examinar el valor de las variables. El agente responde con un get-response. Si el manager considera que debe modificar el valor de alguna variable, envía un paquete set con el nuevo valor de las variables. En este caso el agente devuelve un paquete get-response, en el que incluye el valor modificado de las variables. En caso de error el agente envía un código de error, indicando la variable en la que se produjo el error.

El mecanismo de sondeo se vuelve ineficiente si el manager tiene muchos equipos que sondear y estos equipos tienen muchos objetos, ya que se carga mucho la red. Para evitar este problema se usa el sondeo dirigido por trap. El manager pide información al agente cuando le llega un trap. El problema aquí, surge en la recepción del trap ya que al usar UDP el transporte no es fiable. Por tanto, el manager no se puede basar exclusivamente en la recepción de traps para obtener información de los dispositivos.

Para solucionar este problema se utiliza una mezcla de ambas técnicas: el manager sondea todos los agentes que conoce en la iniciación y a intervalos regulares, pidiendo información clave y características básicas de funcionamiento. El resto del tiempo, el manager espera que le lleguen traps para centrar su atención en el dispositivo. Así se consigue ahorrar capacidad en la red y tiempo de proceso en managers y agentes.

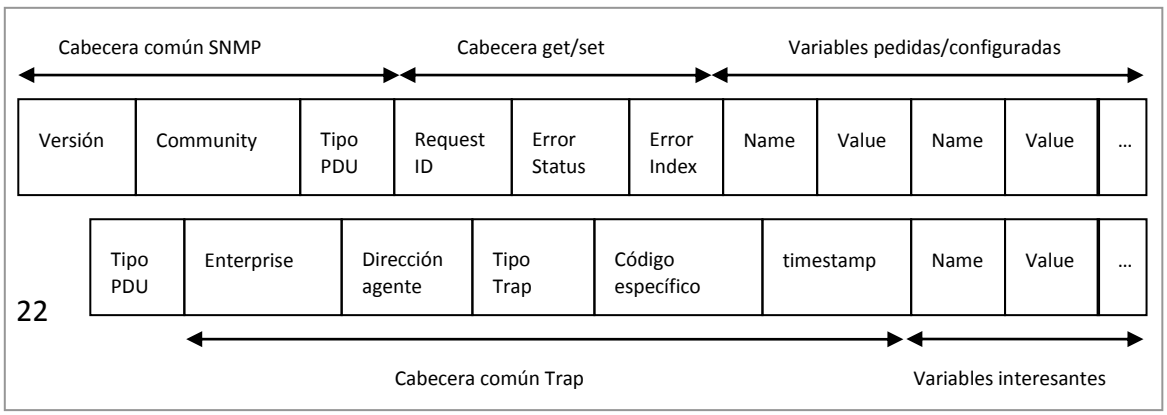

Figura 2.4. **Formato de la PDU de SNMP**

El formato de los mensajes SNMP se muestra en la Figura 2.4. La PDU de los traps y de los mensajes es diferente. En SNMPv2 se añade la nueva PDU notification para unificar el formato de los mensajes SNMP.

- Versión. Indica la versión de SNMP que se está usando.
- Community. Es una cadena de caracteres que se emplea como contraseña entre el manager y el agente. Se envía sin cifrar. En SNMP existen tres comunidades. Cada comunidad se asocia a una actividad. Las comunidades son: read-only, read-write y trap. Cada comunidad tiene su contraseña. Si queremos leer variables tenemos que proporcionar la contraseña de la comunidad read-only. Si además queremos modificar el valor de una variable debemos proporcionar la contraseña de read-write y si queremos recibir traps, hay que suministrar la contraseña de trap.
- Tipo de PDU. Indica el operador que se está usando (get, set, etc.).
- Request ID. Es un identificador que permite que el manager pueda saber a qué petición se dirige una respuesta. Es decir, el manager puede enviar peticiones a varios agentes y al recibir la respuesta, examinará este campo y sabrá que corresponden a la petición que llevaba el mismo ID.
- Error status. Indica que se ha producido un error. Posibles valores: noError(0), tooBig(1), noSuchName(2), badValue(3), readOnly(4), genErr(5).
- Error index. Es un entero que indica en qué variable pedida se ha producido un error. Por ejemplo, puede ser que el manager pida el valor de una variable que no existe, en ese caso indicará que variable de la lista de variables pedidas no existe.
- Lista de variables. El resto de los campos es una serie de parejas nombre_variable=valor. En el caso de get y get-next el campo valor se ignora. En el caso de set, el campo valor corresponde al valor que debe tomar la variable especificada.
- Enterprise. Objeto que genera el trap.
- Dirección de agente. Dirección IP de agente que ha generado el trap.

- Tipo de trap. Tipo de trap que ha producido el evento. Los hay especificados y se pueden definir traps genéricos.
- Timestamp. Tiempo transcurrido entre la última iniciación de la entidad y la ocurrencia del suceso.

2.2.2.5 La estructura de la información de gestión

El primer paso para entender qué tipo de información puede proporcionar un dispositivo es entender cómo se representa esta información en el contexto de SNMP. La representación de datos en SNMP tiene cierta complejidad en su comprensión. Como ya se ha dicho, la gestión de la información se maneja mediante una base de datos (la MIB) que contiene información sobre los objetos a gestionar. Cada recurso se representa por un objeto (el concepto de objeto es muy similar al significado que se le da en programación). La MIB es un conjunto estructurado de tales objetos. Las operaciones de monitorización consultan el valor de esos objetos, es decir, consultan el valor de una instancia determinada de un objeto.

SMIv1 y SMIv2 (RFC 1155, 2578) hacen exactamente eso: definen cómo se tienen que nombrar los objetos y qué tipos de datos tienen asociados. Siguiendo con el símil con la programación orientada a objetos, la SMI definiría la sintaxis del lenguaje y los tipos de datos primitivos. Además define un esquema de identificación de los objetos, que asegura que dos fabricantes no emplean el mismo nombre para objetos distintos.

La definición de los objetos se divide en tres atributos:

- Nombre. El nombre u ObjectIdentifier (OID) define unívocamente un objeto. Los nombres se pueden definir numéricamente o en formato textual. En cualquier caso son largos y engorrosos.
- Tipo y sintaxis. Los tipos de datos se definen empleando la notación ASN.1
- (AbstractSyntaxNotation), un estándar de ITU/ISO. Este es un lenguaje que se emplea para definir estructuras de datos y protocolos. La notación ASN es independiente de la máquina.
- Codificación. Cada instancia de un objeto se codifica en una cadena de octetos usando las reglas de BER (Basic Encoding Rules).

Entonces, la MIB define el objeto u objetos utilizados para representar un recurso en un nodo. Por ejemplo, un nodo puede almacenar el número de conexiones TCP abiertas.

Este número equivale al número de conexiones activas más el número de conexiones pasivas. En la MIB se indicaría:

- Que se debe almacenar el número de conexiones activas y pasivas. Esto implica que, de algún modo, el agente debe ser capaz de conocer ese número. Mediante llamadas al sistema operativo o de alguna otra forma, en función del dispositivo que se trate, el agente debe ir actualizando la información (cada vez que se cierra o abre una conexión).
- Se definen dos objetos para almacenar esta información: tcpActiveOpens y tcpPasssiveOpens, de tipo counter.
- Sus nombres son 1.3.6.1.2.1.6.5 y 1.3.6.1.2.1.6.6.

2.2.2.6 Tipos de datos

SNMP utiliza sólo unos cuantos tipos de datos distintos, un subconjunto de ASN.1. Los tipos de datos se dividen en 4 clases de tipos:

- Universal: tipos básicos, independientes de la aplicación.
- Application: relevantes a una aplicación en particular, por ejemplo, SNMP.
- Contex-specific: relevantes a una aplicación, pero aplicables en un contexto limitado.
- Private: definidos por los usuarios. No estandarizados.

Los tipos de datos universales usados en SNMP son:

- INTEGER: Entero de 32 bits. A menudo se usa para especificar tipos enumerados. Ejemplo: up(1), down(2).
- OCTEC STRING: Cadena de 0 o más octetos. Puede definirse la longitud de la cadena y su valor inicial.
- NULL: ningún valor. Es el tipo que se usa, por ejemplo, en un get para el valor de las parejas nombre-valor.
- OBJECT IDENTIFIER: Identificador de objetos. Es una secuencia de números que determina la posición de un objeto dentro de la estructura de árbol.
- SEQUENCE: lista ordenada de tipos. Es similar a una estructura en C. Por ejemplo, la MIB-II define la SEQUENCE llamada UdpEntry, que contiene dos entradas en la estructura:
  - udpLocalAddress, de tipo IpAddress, que contiene la dirección IP local.
  - udpLocalPort, de tipo INTEGER, que contiene el puerto Local.
- SEQUENCE OF: es la definición de un vector, cuyos elementos son todos del mismo tipo. Si el tipo de los elementos es simple, por ejemplo, INTEGER, tendremos un vector. Si el tipo del vector es SEQUENCE, tendremos una tabla bidimensional. Por ejemplo, la tabla udpTable se define mediante un SEQUENCEOF udpEntry.

Los tipos de datos de aplicación son:

- IpAddress: Una dirección IP (32 bits). Definido como un OCTEC STRING de longitud cuatro.
- Counter: un contador. Entero no negativo de 32 bits. Se puede incrementar pero no decrementar. Cuando llega al máximo vuelve a 0.
- Gauge: Entero no negativo de 32 bits. Se puede incrementar y decrementar. Cuando llega al máximo se queda bloqueado en ese valor.
- TimeTicks: entero no negativo de 32 bits. Cuenta el tiempo en centésimas de segundo.
- Opaque: datos arbitrarios (apenas se usa).

2.2.2.7 Esquema común de identificación

La SMI utiliza un esquema jerárquico de nombres, desarrollado por ISO y definido en la SMI. Se forma una estructura tipo árbol, donde cada nodo se define mediante una etiqueta compuesta de una breve descripción textual y un número. Cada objeto es un nodo del árbol, es decir, cada nodo representa un recurso, actividad o información relacionada (véase Figura 2.5).

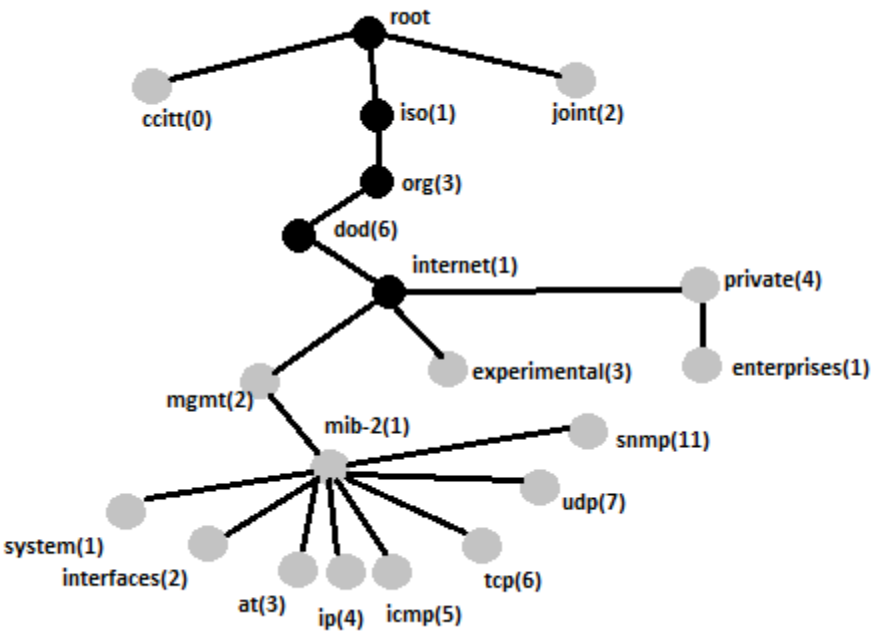

Figura 2.5. **Árbol de objetos de la SMI**

Un identificador de objeto (definido mediante el tipo OBJECT IDENTIFIER), es el nombre de un nodo. Es la cadena de enteros, separados por puntos, desde la raíz hasta el nodo en cuestión. También se puede emplear la notación textual, el nombre del nodo, aunque es sólo para clarificar, nunca se envía en peticiones SNMP.

El nodo internet (1.3.6.1) está administrado por el IAB. De él cuelgan todos los objetos de interés para SNMP.

Dentro del nodo internet se definen 4 objetos:

- directory: especifica el directorio X.500.
- mgmt: objetos definidos en documentos aprobados por el IAB, como la MIB-II.
- experimental: objetos de prueba e investigación.
- private: bajo esta rama pueden definir sus objetos las empresas y particulares.

La asignación de números la realiza el IANA. Las empresas pueden solicitar un número bajo la rama private y definir sus propios objetos, para formar sus MIB. Los

objetos estándar se prueban primero en la rama experimental. Cuando el IAB da su aprobación se pasan a la rama mgmt.

2.2.2.8 Definición de objetos y tablas

Cada objeto de una MIB de SNMP tiene una definición formal que especifica:

- El tipo de datos del objeto.
- El rango de valores que puede tomar.
- Su relación con otros objetos de la MIB, esto es, su posición dentro del árbol corresponde a su OID.

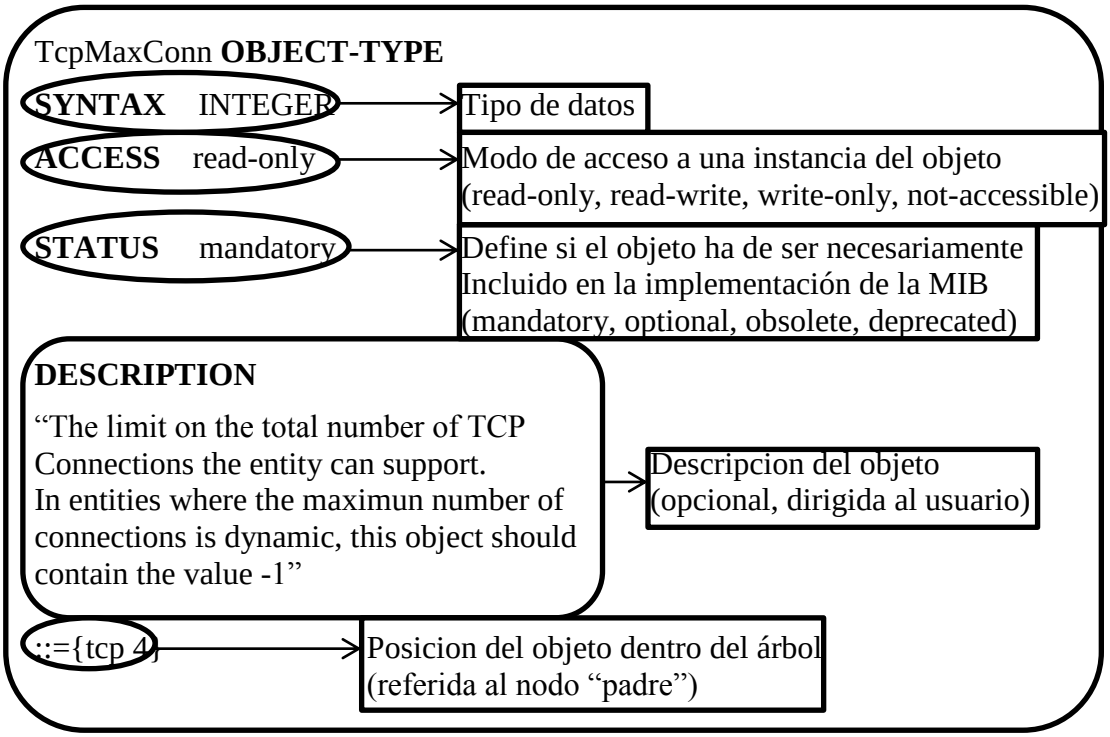

Figura 2.6. **Ejemplo de definicion de un objeto**

En la Figura 2.6 se puede ver un ejemplo de la definición de un objeto. Todos los objetos se definen mediante 4 campos: SINTAX, ACCESS, STATUS y DESCRIPTION.

En SNMP sólo hay dos tipos de variables, los objetos escalares o las tablas bidimensionales. SNMP sólo permite acceder a objetos hoja del árbol, es decir, no se puede recuperar una columna completa de una tabla, sino que hay que acceder a las variables individuales.

Para identificar una instancia de un objeto escalar, se le añade un 0 a su OID. Por ejemplo, si queremos leer el valor del objeto de la Figura 2.6 (tcpMaxConn), cuyo OID es 1.3.6.1.2.1.6.4, hay que referenciarlo como 1.3.6.1.2.1.6.4.0. Éste es el valor que realmente se enviará en un mensaje SNMP.

En el caso de las tablas, SNMP sólo permite estructurar los datos como tablas bidimensionales con valores escalares. La definición de las tablas es siempre igual (véase Figura 2.7), se define un objeto tabla, cuyo tipo de datos será un SEQUENCE OF, es decir un vector de un tipo de datos, que debe ser un SEQUENCE, es decir, una estructura.

```
TcpConnTable      OBJECT-TYPE                OJO: T  mayúscula
    SYNTAX        SEQUENCE  OF  TcpConnEntry
    ACCESS        not-accessible
    STATUS        mandatory
    DESCRIPTION
          "A table containing TCP connection-specific
          information"
    ::={tcp 13}
```

Figura 2.7. **Definición de un objeto tabla en la MIB**

En la tabla del ejemplo, se pretende dar información sobre las conexiones TCP de un dispositivo. Es decir, cada fila de la tabla almacenará información de una conexión. Una vez definida la tabla, se hace una definición del objeto fila, es decir, de las filas de cada tabla (véase la Figura 2.6). La definición es casi idéntica a la de una tabla, salvo que se ha añadido un nuevo campo, INDEX, que identifica una única fila en la tabla. El índice serán valores de uno o varios campos de la tabla. El diseñador de la tabla debe hacer que el índice de la fila sea único dentro del contexto de la tabla.

```
tcpConnEntry     OBJECT-TYPE          INDEX{ tcpConnLocalAddress
  SYNTAX         TcpConnEntry         tcpConnLocalPort,
  ACCESS         not-accessible       tcpConnRemAddress,
  STATUS         mandatory            ::={tcpConnTable 1}
  DESCRIPTION
  "Information about a particular     TcpConnEntry::=SEQUENCE{
  current TCP connection. An object       tcpConnState  INTEGER,
   of this type is transient, in that it  tcpConnLocalAddress, IpAddress,
  ceases to exist when (or son after)      tcpConnLocalPort INTEGER (0…65535)
  the connection makes the ransition       tcpConnRemAddress IpAddress,
  to the CLOSED state"                     tcpConnRemPort INTEGER (0…65535)
                                           }
```

Figura 2.8. **Definición de una fila**

Mediante la combinación del OID del objeto fila y los campos índice de las tablas se podrá acceder posteriormente a cada uno de los valores de los campos de la tabla. Por eso es necesario definir el objeto fila.

Por último, hay que definir los objetos de los campos de cada fila, es decir, los objetos que forman parte del objeto SEQUENCE (véase Figura 2.9).

```
tcpConnState OBJECT-TYPE
      SYNTAX INTEGER{
        Closed(1), listen(2), synSent(3),
        synRecived(4), estabilished(5),
        finWalt1(6), finWalt2(7), closeWalt(8)
        lastAck(9), closing(10), timeWall(11),
        deleteTCB(12)}
      ACCESS read-write
      STATUS mandatory
      DESCRIPTION
        "The state of this TCP connection…."
      ::={tcpConnEntry 1}
```

```
tcpConnLocalAddress        OBJECT-TYPE
            SYNTAX  IpAddress
            ACCESS  read.only
            STATUS  mandatory
            DESCRIPTION
            "The local IP address of this TCP
            connection,….."
            ::= {tcpConnEntry 2}
tcpConnLocalPort OBJECT-TYPE
            SYNTAX INTEGER (0…65535)
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
"The local port number for this TCP connection"
::={tcpConnEntry 3}
```

Figura 2.9. **Definición de los campos de una tabla (solo los 3 primeros campos)**

2.2.2.9 Tablas

Se ha visto que las operaciones de SNMP sólo se aplican a variables escalares, por tanto, sólo se puede realizar operaciones sobre entradas individuales de la tabla. Las variables escalares se instancian añadiendo un 0 a su OID. La forma de instanciar los campos de las tablas es un poco más compleja.

Cada entrada en una tabla se direcciona concatenando información de la instancia al final del OID del campo al que se intenta acceder. La primera parte de esa información identifica la fila y la segunda la columna.

En el ejemplo de la tabla anterior, tenemos que el OID de la tabla tcpConnTable es 1.3.6.1.2.1.6.13. Siguiendo el árbol, el OID para una fila será 1.3.6.1.2.1.6.13.1 (objeto tcpConnEntry). Entonces, los OID 1.3.6.1.2.1.6.13.1.1 ... 1.3.6.1.2.1.6.13.1.5 identifican

cada uno de las 5 campos de cada fila (recordar que esta tabla contenía registros de 5 campos).

Falta identificar una entrada de fila en particular. Para ello se usa el índice de la tabla. El índice lo elige el diseñador de la tabla. Está formado por valores de uno o varios de los campos de la tabla. En el ejemplo de tcpConnTable, está formado por el conjunto ordenado de valores {tcpConnLocalAddress, tcpConnLocalPort, tcpConnRemAddress, tcpConnRemPort}. Entonces, supongamos que la tabla tiene 3 entradas. Esto significa que el dispositivo mantiene 3 conexiones TCP activas. La información que aporta la tabla es la siguiente:

tcpConnState	tcpConnLocalAddres	tcpConnLocalPort	tcpConnRemAddress	tcpConnRemPort
2	0.0.0.0	9	0.0.0.0	0
2	0.0.0.0	19	0.0.0.0	0
6	158.42.160.18	1028	158.42.160.21	6000

Tabla 2.1. Datos de la tabla tcpConnTable

La instancia del objeto tcpConnState de la última fila se direcciona mediante el siguiente OID 1.3.6.1.2.1.6.13.1.1.158.42.160.18.1028.158.42.160.21.6000. Lo que quiere decir que si enviamos el siguiente mensaje:

get(1.3.6.1.2.1.6.13.1.1.158.42.160.18.1028.158.42.160.21.6000)

El agente devolverá el valor 6. En este caso lo que se ha hecho es utilizar el acceso aleatorio a las tablas, en el que se especifica la instancia completa y se usa el operador get. Pero esta no es la manera habitual de leer las tablas ya que para direccionar una variable necesitamos conocer el contenido de la propia tabla. Es decir, para direccionar la variable tcpConnState necesitamos saber que el campo tcpConnLocalAddress de la última fila tiene el valor 158.42.160.18, al igual que necesitamos saber el valor del resto de los campos que se utilizan de índice.

Como no es habitual que conozcamos a priori el valor de los datos de una tabla, normalmente se utiliza el acceso secuencial, mediante el operador get-next:

- Se utiliza especificando el OID de la columna que nos interesa.
- Se devuelve el valor y la instancia del identificador del siguiente objeto, en orden lexicográfico.

Hay un ordenamiento implícito en la MIB basado en el orden de los OID. El orden lexicográfico hace que se ordenen los objetos por el OID que los identifica, de menor a mayor. Esto quiere decir que para ordenar un objeto se comienza por el entero más hacia la izquierda de su OID y se hacen las comparaciones entre los enteros que ocupan la misma posición en el OID. Por ejemplo, el OID 1.3.6.1.1.14 aparecería antes que el OID 1.3.6.1.2.4 porque se comparan los enteros del OID de izquierda a derecha y el penúltimo entero del primer OID (1) es menor que el correspondiente al segundo OID (2).

El orden lexicográfico hace que:

- Puesto que todas las entradas de una variable dada (tcpConnState) aparecen antes que las instancias de la siguiente variable (tcpConnLocalAddress), las tablas se acceden en orden columna-fila.
- El orden de las filas de las tablas depende de los valores de los índices de la tabla.

La Figura 2.10 muestra el recorrido de la tabla en orden lexicográfico. Mediante el uso del operador get-next (Figura 2.11), se puede ir recorriendo columna a columna una tabla sin necesidad de conocer a priori los valores de los índices. El problema que tiene este operador es que hay que enviar y recibir un mensaje por cada entrada de una tabla. Es un acceso muy ineficiente.

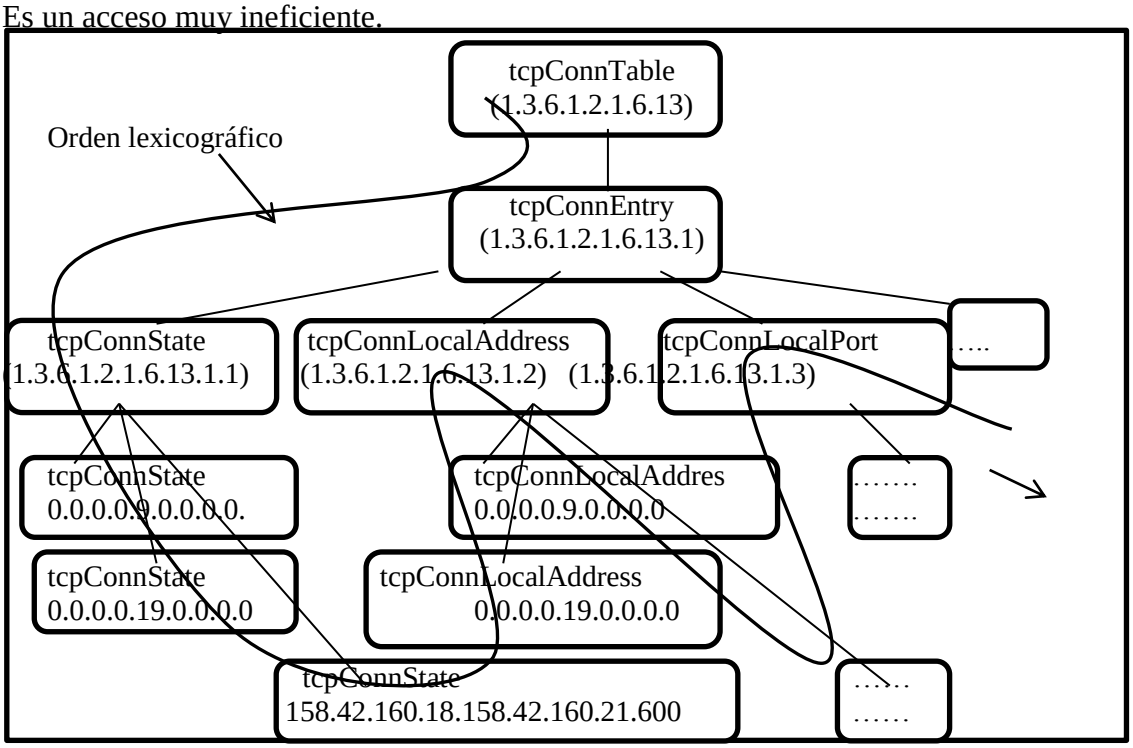

Figura 2.10. **Recorrido de la tabla en orden lexicográfico**

La NMS puede crear filas en las MIB. Para insertar filas en una tabla, la estación de gestión envía un mensaje set con el OID de una instancia de objeto no existente. El agente crea una fila con valores de los campos por defecto. Posteriormente la NMS varía con set cada campo particular.

El borrado de filas se realiza estableciendo un valor determinado (inválido o algo similar) al campo de la tabla que se decida que cumplirá esa función. En el ejemplo, el establecimiento de la variable tcpConnState de una fila al valor deleteTCB(12) provoca que se borre la fila y que se cierre esa conexión, queda claro, por tanto, que la modificación del estado de la MIB tiene su reflejo en el estado real del dispositivo, es decir, nos permite gestionar remotamente su funcionamiento.

OBJETO	OID	SIGUIENTE INSTANCIA
tcpConnTable	1.3.6.1.2.1.6.13	1.3.6.1.2.1.6.13.1. 0.0.0.0.9.0.0.0.0.0
tcpConnEntry	1.3.6.1.2.1.6.13.1	1.3.6.1.2.1.6.13.1. 0.0.0.0.9.0.0.0.0.0
tcpConnState	1.3.6.1.2.1.6.13.1.1	1.3.6.1.2.1.6.13.1. 0.0.0.0.9.0.0.0.0.0
tcpConnState. 0.0.0.0.9.0.0.0.0.0	1.3.6.1.2.1.6.13.1. 0.0.0.0.9.0.0.0.0.0	1.3.6.1.2.1.6.13.1. 0.0.0.0.19.0.0.0.0.0
tcpConnState. 0.0.0.0.19.0.0.0.0.0	1.3.6.1.2.1.6.13.1. 0.0.0.0.19.0.0.0.0.0	1.3.6.1.2.1.6.13.1.1.158.42.160.18.102 8.158.42.160.21.6000
tcpConnState.158.42.160.18. 1028.158.42.160.21.6000	1.3.6.1.2.1.6.13.1.1.158.42.160.18 .1028.158.42.160.21.6000	1.3.6.1.2.1.6.13.1.2. 0.0.0.0.9.0.0.0.0.0
tcpConnLocalAddress	1.3.6.1.2.1.6.13.1.2	1.3.6.1.2.1.6.13.1.2. 0.0.0.0.9.0.0.0.0.0
tcpConnLocalAddress 0.0.0.0.9.0.0.0.0.0	1.3.6.1.2.1.6.13.1.2. 0.0.0.0.9.0.0.0.0.0	1.3.6.1.2.1.6.13.1.2. 0.0.0.0.19.0.0.0.0.0

Tabla 2.2. Siguiente instancia en orden lexicográfico

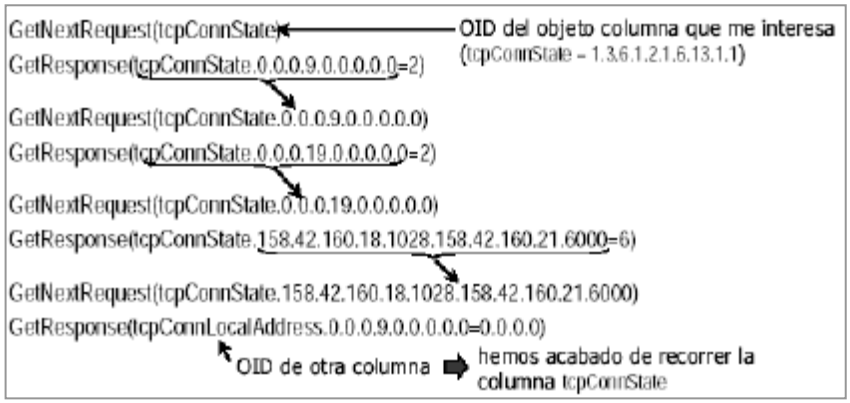

Figura 2.11. **Ejemplo del recorrido de una tabla mediante get-next.**

2.2.2.10 MIB-II

La MIB-II se define en la RFC 1213 y sustituye a la versión anterior (RFC 1156). Define información de gestión general sobre TCP/IP para los dispositivos. Se compone de varios grupos. Si un agente implementa un grupo, debe implementar todos los objetos del grupo. En caso contrario, no debe implementar el grupo.

GRUPO	NOMBRE	DESCRIPCIÓN
System	system	Descripción del sistema
Interfaces	interfaces	Descripción de los interfaces del sistema
Addresstraslation	at	Mapeo entre direcciones físicas e IP
Internet Protocol	ip	Estadísticas del protocolo IP
Internet Control MessageProtocol	icmp	Estadísticas del protocolo ICMP
Transmision Control Protocol	tcp	Estadísticas del protocolo TCP
UserDatagramProtocol	udp	Estadísticas del protocolo UDP
Exterior Gateway Protocol	egp	Estadísticas del protocolo EGP
Transmisión	transmission	MIB de los medios de transmisión
SNMP	snmp	Estadísticas del propio protocolo SNMP

Tabla 2.3. Grupos de la MIB II

2.2.2.11 Evolución de SNMP

En las secciones anteriores hemos repasado brevemente el funcionamiento de SNMPv1. Esta versión del protocolo presenta las siguientes ventajas:

- Es un protocolo maduro, ampliamente implementado en todo tipo de dispositivos.
- Es un estándar de facto de la industria.
- Es fácil de implementar y requiere pocos recursos del sistema.
- Sin embargo tiene importantes deficiencias:
- La falta de seguridad es uno de las inconvenientes fundamentales: cualquier estación puede modificar variables, no hay control de acceso y las claves se transmiten sin cifrar.
- El desperdicio de ancho de banda que supone no poder transferir la información en bloques se suma a la ineficiencia del sondeo para redes grandes.
- El mecanismo de traps está limitado por la ausencia de reconocimiento de los mismos, así como por la poca variedad de sucesos de los que pueden informar.

Las siguientes versiones de SNMP intentaron subsanar estas deficiencias. SNMPv2 intentó subsanar las deficiencias de seguridad pero finalmente no se llegó a un consenso sobre el método a utilizar y se pospuso el tema. Esta versión de SNMP extiende y mejora la SMI, con la introducción de nuevos tipos y nuevas cláusulas para la definición de objetos. Por otra parte, amplía el soporte a la gestión distribuida, con la introducción de la PDU Inform. Introduce la PDU GetBulkRequest, que permite la transferencia de bloques de información. Cambia el formato de las traps para facilitar su procesado en los equipos. Por último, especifica otros protocolos de transporte además de UDP.

La tercera versión, SNMPv3 se centra en las cuestiones de seguridad, aunque para añadir estos servicios especifica mucho más detalladamente que piezas forman la implementación de las entidades SNMP. Se abandona el concepto de manager y agente y en cambio se describen los bloques que forman las entidades. Se definen bloques que forman el motor SNMP, es decir, que proporcionan la funcionalidad SNMP básica y bloques de aplicación que son los que determinan el papel de un equipo en el modelo.

De acuerdo a los bloques que implemente una entidad, la función que realiza la entidad variará. La Figura 2.12 muestra el diagrama de bloques de la entidad SNMP versión 3.

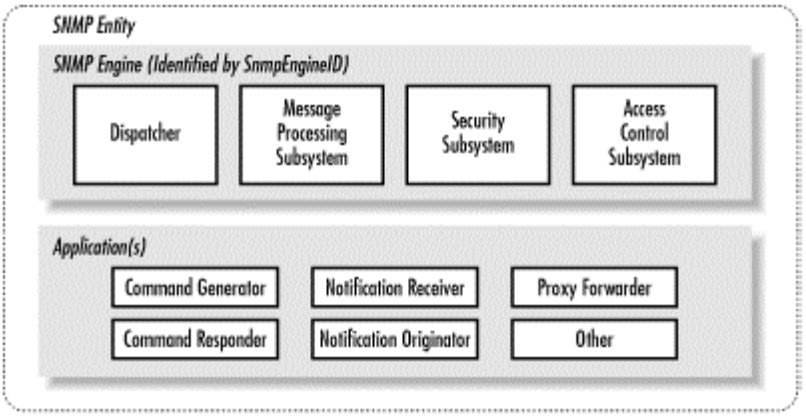

Figura 2.12. **Entrada de SNMPv3**

2.3 Arquitectura de visualización de datos (Google Web Toolkit)

2.3.1 Descripción

Google Web Toolkit es un entorno de desarrollo Java, basado en Software libre y que permite escribir aplicaciones AJAX fácilmente. GWT, permite escribir las aplicaciones en el lenguaje de programación Java, y luego se encarga de compilarlo, traduciendo la parte del cliente a lenguaje de programación JavaScript + HTML + CSS, generando código JavaScript más eficiente que el escrito a mano.

Permite integrar de forma nativa código JavaScript con los JSNI (JavaScript Native Interface).

La ventaja de programar en Java es que se pueden utilizar los IDE existentes para este lenguaje, como es el caso de Eclipse o NetBeans, así como sus herramientas de depuración.

Las aplicaciones generadas por GWT ejecutan código Java del lado del servidor, utilizando RPC para la comunicación entre el Cliente y el Servidor, llevando a cabo llamadas asíncronas.

2.3.2 Ventajas

- Similar a una aplicación desktop. Alto dinamismo de las pantallas
- No necesita conocimientos de Javascript
- Desarrollo ágil
- Permite avanzar desde un prototipo
- Multiplataforma, Multinavegador
- Reduce el ancho de banda una vez cargada la aplicación en la cache
- Reduce la carga en el servidor(los datos que se cargaban en la sesión del servidor ahora pasan al cliente)
- Permite la misma seguridad que otros frameworks (inclusive es más complicado inyectar datos debido al sistema de serialización que utiliza)

2.3.3 Desventajas

Solo desarrolladores JAVA

Curva de aprendizaje lenta al principio si el programador tiene mucha experiencia en otra tecnología

Dependiendo de las características de la aplicación puede ser un poco más costoso el despliegue

Consumo de memoria del navegador

No es 100% API Java Objects

2.4 SCADA

2.4.1 El Sistema SCADA

El Sistema de Control Supervisorio y de Adquisición de Datos (SCADA) es una tecnología que permite obtener y procesar información de procesos industriales dispersos o lugares remotos inaccesibles, transmitiéndola a un lugar para supervisión, control y procesamiento, normalmente una Sala o Centro de Control. Un SCADA permite entonces supervisar y controlar simultáneamente procesos e instalaciones distribuidos en grandes áreas, y generar un conjunto de información procesada como, por ejemplo, presentación de gráficos de tendencias e información histórica, de informes de operación y programación de eventos, programas de mantenimiento preventivo, etc. En la Figura 2.13 se muestra la configuración típica de un Sistema de Automatización Industrial a Nivel Operacional. En el Nivel 1, Figura 2.13 , se encuentran las Redes de Campo, cuya descripción veremos más adelante. El Nivel 2 es el dominio de la telecomunicación y el Nivel 3 es el dominio de la Supervisión y Control Global.

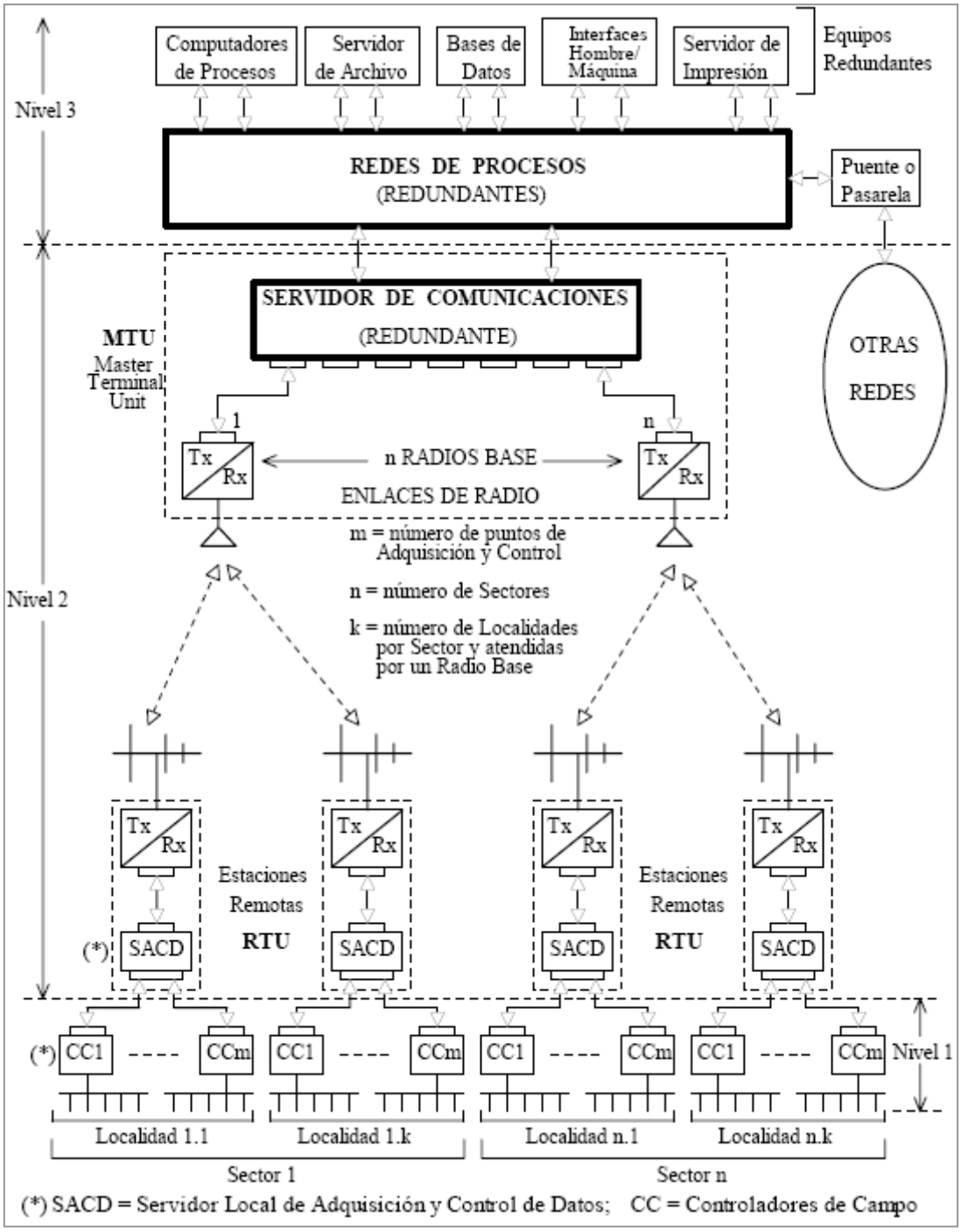

Figura 2.13. **Nivel Operacional de un Sistema Integrado de Automatización de Control.**

Un SCADA no debe confundirse con un Sistema de Control Distribuido ("Distributed Control System, DCS") aunque los principios y tecnologías que ambos utilizan son similares. La diferencia principal es que los DCS normalmente se utilizan

para controlar procesos industriales complejos dentro de un área pequeña, por ejemplo, una planta industrial y las restricciones en tiempo son muy diferentes. En cambio, el SCADA se emplea para el control y supervisión de áreas geográficas muy grandes, como, por ejemplo, un sistema de distribución de energía eléctrica o las instalaciones de las compañías petroleras, y la red de comunicaciones es su soporte físico.

La incorporación de un SCADA en un proceso permite al usuario conocer el estado de las instalaciones bajo su responsabilidad y coordinar eficazmente las labores de producción y mantenimiento en el campo, supervisando y controlando operaciones críticas y proporcionando los recursos para recibir la información en forma dinámica y en tiempo real, y proceder a su procesamiento ulterior.

2.4.1.1 Sistema de Automatización a Nivel Operacional

Como se puede observar en la Figura 2.13, se distinguen tres niveles o subsistemas: el Nivel 1 o Subsistema de Instrumentación y Control Local, el Nivel 2 o Subsistema de Comunicaciones y el Nivel 3 o Subsistema de Procesamiento y Control Global.

2.4.1.2 Subsistema de Instrumentación y Control Local

Este es el nivel que está en contacto directo con el proceso y por lo tanto se encuentra distribuido en las localidades remotas a las que se quiere controlar y supervisar. Aquí se encuentran las Redes de Campo que incluyen toda la instrumentación asociada con el proceso, los elementos finales de control, así como los medios de conversión de la información en un formato digital apropiado para su transmisión al Nivel 3 o Subsistema de Procesamiento de Datos y Control Global. Este Nivel 1 o Subsistema de Instrumentación y Control Local está constituido por equipos específicos (controladores y redes de campo) que se ubican lo más cerca posible del proceso: instrumentos de medición (temperatura, presión, flujo, velocidad, etc.), sistemas PLC (“ProgrammableLogicControllers”), sensores, actuadores, válvulas de control, bombas, compresores, etc. Bajo instrucciones desde el Centro de Control, en el Nivel 1 se realiza las operaciones de control y los ajustes en las tablas de configuración de parámetros tanto continuos como discretos de un lazo de control. Para su transmisión a los niveles superiores, algunas de estas funciones se integran en las denominadas

"unidades terminales remotas (remote teminal unit, RTU)", cuya descripción se verá más adelante.

2.4.1.3 Subsistema de Comunicaciones

En cada localización remota de interés se instala un servidor de adquisición y control que junto con el transceptor de comunicaciones constituye la Estación Remota o RTU ("Remote Terminal Unit"), la cual debe mantenerse en continua comunicación con el Centro de Control. Esta comunicación la realiza el subsistema de comunicaciones o Nivel 2, mostrado en la Figura 2.13 por un sistema de comunicaciones dado y utilizando protocolos especiales.

El Servidor de Comunicaciones junto con los Radios Base, constituyen la Estación Maestra ("Master Terminal Unit, MTU"). La función general de la MTU es la de realizar todas las labores de interrogación y comunicaciones entre el Nivel 1 y el Nivel 3. En la figura 1.3 se utilizan medios radioeléctricos de transmisión; sin embargo, se puede utilizar conductores metálicos, fibras ópticas, satélites, rayos infrarrojos, laser, etc.; la selección del medio de transmisión depende fundamentalmente, aparte de los aspectos económicos, de las condiciones climáticas o geográficas, y muchas veces el medio de transmisión es una combinación de estos medios.

Los protocolos utilizados en el subsistema de comunicaciones, denominados protocolos industriales o de campo, permiten la interacción entre los equipos de comunicación. Estos protocolos están constituidos por un conjunto de reglas y procedimientos para el intercambio de mensajes, detección y corrección de errores, y establecer las secuencias y lazos de control y supervisión.

2.4.1.4 Subsistema de Procesamiento y Control Global

Una vez que los datos han sido recolectados desde las localidades remotas y transmitidos el Centro de Control, Nivel 3 de la Figura 2.13, es necesario realizar sobre ellos un cierto procesamiento en tiempo real a fin de obtener información útil acerca de los procesos, presentarla al operador (o usuario) y emprender acciones de supervisión y control cuando sea necesario. Este trabajo lo realiza el subsistema de procesamiento de datos. Este subsistema es el brazo operativo del Centro de Control y es el encargado de

ordenar y procesar la información que es recibida del proceso mediante los enlaces de comunicación.

El Centro de Control debe poseer una alta capacidad de computación y normalmente está constituido por computadoras y redes de alta velocidad, interfaces hombre-máquina, bases de datos, servidores de aplicación (de impresión, de archivo, de datos históricos, de monitoreo, etc.). Todos estos recursos deberán ser redundantes para asegurar la confiabilidad e integridad en todas las operaciones. El subsistema de procesamiento y control global podrá comunicarse también con los niveles Tácticos y Estratégicos, figura 1.1, para la toma de las decisiones tácticas y estratégicas correspondientes.

La operación global del sistema está gobernada por programas informáticos que manejan el Sistema Operativo, las Bases de Datos, el Software SCADA y los Programas de Servicio y Aplicación.

3

Hardware

En este capítulo se hará una breve descripción del hardware utilizado

3.1 SIEMENS SENTRON PAC4200:

Figura 3.1. Imagen Siemens Sentron PAC4200

SENTRON PAC4200 es un multímetro tipo central de medida para la visualización, el almacenamiento y el monitoreo de todos los parámetros de red relevantes en la distribución de energía eléctrica en baja tensión. Puede realizar mediciones monofásicas, bifásicas y trifásicas, y puede utilizarse en redes (sistemas) en esquema TN, TT e IT de dos, tres o cuatro conductores.

Gracias a su diseño compacto en formato 96 x 96 mm, se adapta a cualquier recorte normalizado. SENTRON PAC4200 mide cerca de 200 magnitudes eléctricas con valores mínimos, valores máximos y medias.

Gracias a su amplio rango de tensión de medición, SENTRON PAC4200 con fuente de alimentación de amplio rango de entrada puede conectarse a cualquier red de baja tensión con una tensión nominal de hasta 690 V (máx. 600 V para UL).

Para la variante con fuente de alimentación de muy baja tensión está permitida la conexión directa a redes de hasta 500 V.

Pueden medirse tensiones superiores si se usan transformadores de tensión. Para la medición de corriente se pueden utilizar transformadores de corriente x/1 A o x/5 A. La gran pantalla gráfica de cristal líquido permite la lectura incluso a grandes distancias.

SENTRON PAC4200 dispone de una retroiluminación ajustable para garantizar una lectura óptima incluso en condiciones lumínicas desfavorables.

Ofrece un manejo intuitivo para el usuario gracias a cuatro teclas de función, e información multilingüe en texto claro. Adicionalmente, el usuario experimentado

dispone de una navegación directa que permite realizar una selección rápida del menú deseado.

SENTRON PAC4200 ofrece una alta precisión de medida. Permite la captación y el almacenamiento de curvas de carga siguiendo diferentes métodos. Dispone de una serie de útiles funciones de monitoreo, diagnóstico y servicio técnico, un contador de tarifa doble de energía aparente, activa y reactiva, dos contadores universales y un contador de horas de funcionamiento para monitorizar los consumidores conectados.

SENTRON PAC4200 guarda el consumo diario de energía aparente, activa y reactiva y la tarifa a lo largo de un año. Además, el multímetro posee un contador de energía aparente, activa y reactiva para medir el consumo de energía de un proceso de fabricación. Un contador propio de horas de funcionamiento determina la duración de este proceso. Para controlar los contadores de energía de proceso se utilizan las entradas digitales existentes.

Un completo sistema de avisos parametrizable permite el monitoreo específico del usuario de diversos eventos, por ejemplo, violaciones de límite o intervenciones de manejo.

La memoria de datos del dispositivo y el reloj interno están respaldados por pila.

Para la comunicación se puede utilizar la interfaz Ethernet 10/100 Mbits integrada o un módulo de ampliación opcional, por ej. el módulo SENTRON PAC RS485 o el módulo SENTRON PAC PROFIBUS DP.

SENTRON PAC4200 dispone de dos entradas digitales multifuncionales y dos salidas digitales multifuncionales con los módulos de ampliación opcionales SENTRON PAC 4DI/2DO es posible añadir a SENTRON PAC4200 hasta un máximo de 8 entradas digitales y 4 salidas digitales. Por lo tanto, se puede llegar a una configuración máxima de 10 entradas digitales y 6 salidas digitales. Las entradas y salidas digitales externas poseen las mismas funciones que las entradas y salidas digitales integradas. Gracias a la alimentación interna, las entradas y salidas digitales de los módulos de ampliación pueden utilizarse como interfaces S0 según IEC 62053-31. Esto permite, entre otras cosas, utilizar contactos simples aislados galvánicamente para conectar las entradas digitales.

La parametrización puede realizarse directamente en el dispositivo o con el software de configuración SENTRON powerconfig.

Para evitar accesos no autorizados se ha integrado un sistema de protección por clave.

3.2 TS-PC103

Figura 3.2. **Imagen PC103 de VDX**

El equipo TS-PC103 es un controlador de red basado en Linux 2.6.32.4 con diferentes interfaces de E/S, Ethernet y coprocesador digital.

Las características del equipo son las siguientes:

- Caja de aluminio siguiendo el estándar PC104. Gomas de sujeción.
- Fuente de alimentación 24VDC protegida con DC/DC.
- Procesador ARM9 S3C2440A con 200 bogomips.
- FPGA Xilinx Spartan3 para procesado digital.
- 64Mb RAM, SD 2Gb.
- Ethernet 10 Mbit.
- RTC.
- 7 Entradas y 7 salidas digitales opto-aisladas.
- Puertos RS-232 opto-aislados.

4

SOFTWARE

En este capítulo se hará una breve descripción del software utilizado

Figura 4.1. Icono del sistema operativo LINUX

4.1 Linux

4.1.1 Antecedentes

En 1983 Richard Stallman inició el Proyecto GNU, con el propósito de crear un sistema operativo similar y compatible con UNIX y los estándares POSIX. Dos años más tarde, 1985, creó la Fundación del Software Libre (FSF) y desarrolló la Licencia pública general de GNU (GNU GPL), para tener un marco legal que permitiera difundir libremente el software. De este modo el software de GNU fue desarrollado muy rápidamente, y por muchas personas. A corto plazo, se desarrolló una multiplicidad de programas, de modo que a principios de los años 1990 había casi bastante software disponible como para crear un sistema operativo completo. Sin embargo, todavía le faltaba un núcleo.

Esto debía ser desarrollado en el proyecto GNU Hurd, pero Hurd demostró desarrollarse muy inactivamente, porque encontrar y reparar errores (eliminación de fallos, debugging en inglés) era muy difícil, debido a las características técnicas del diseño del micro núcleo.

Otro proyecto de sistema operativo software libre, en los años 1980 fue BSD. Este fue desarrollado en la Universidad de Berkeley desde la 6ª edición de Unix de AT&T. Puesto que el código de AT&T Unix estaba contenido en BSD, AT&T presentó una

demanda a principios de los años 1990 contra la Universidad de Berkeley, la cual limitó el desarrollo de BSD y redujo el desarrollo. Así a principios de los años 1990 no produjo ningún sistema completo libre.

El futuro de BSD era incierto debido al pleito y detuvo el desarrollo. Además, el Proyecto GNU gradualmente se desarrollaba pero, este carecía de un bien formado núcleo UNIX. Esto dejó un nicho crítico abierto, que Linux llenaría muy pronto

4.1.2 GNU/Linux

La designación "Linux" al principio fue usada por Torvalds sólo para el núcleo. El núcleo fue, sin embargo, con frecuencia usado junto con otro software, especialmente con el del proyecto de GNU. Esta variante de GNU rápidamente se hizo la más popular, ya que no había ningún otro núcleo libre que funcionara en ese tiempo. Cuando la gente comenzó a referirse hacia esta recopilación como "Linux", Richard Stallman, el fundador del proyecto de GNU, solicitó que se usara el nombre GNU/Linux, para reconocer el rol del software de GNU.9 En junio de 1994, en el boletín de GNU, Linux fue mencionado como un "clon libre de UNIX", y el Proyecto Debian comenzó a llamar a su producto GNU/Linux. En mayo de 1996, Richard Stallman publicó al editor Emacs 19.31, en el cual el tipo de sistema fue renombrado de Linux a Lignux. Esta "escritura" fue pretendida para referirse expresamente a la combinación de GNU y Linux, pero esto pronto fue abandonado en favor de "GNU/Linux".

El producto terminado es más a menudo denominado simplemente como "Linux", como el más simple, el nombre original. Stallman anunció su demanda por un cambio de nombre sólo después de que el sistema ya se había hecho popular.

4.2 Oracle(base de datos)

Oracle surge en 1977 bajo el nombre de SDL (Software Development Laboratories), luego en 1979 SDL cambia su nombre por Relational Software, Inc. (RSI). La fundación de Software Development Laboratories (SDL) fue motivada principalmente a partir de un estudio sobre los SGBD (Sistemas Gestores de Base de Datos) de George Koch. Computer World definió este estudio como uno de los más completos jamás

escritos sobre bases de datos. Este artículo incluía una comparativa de productos que erigía a Relational Software como el más completo desde el punto de vista técnico. Esto se debía a que usaba la filosofía de las bases de datos relacionales, algo que por aquella época era todavía desconocido.

En la actualidad, Oracle (Nasdaq: ORCL) todavía encabeza la lista. La tecnología Oracle se encuentra prácticamente en todas las industrias alrededor del mundo y en las oficinas de 98 de las 100 empresas Fortune 100. Oracle es la primera compañía de software que desarrolla e implementa software para empresas 100 por ciento activado por Internet a través de toda su línea de productos: base de datos, aplicaciones comerciales y herramientas de desarrollo de aplicaciones y soporte de decisiones. Oracle es el proveedor mundial líder de software para administración de información, y la segunda empresa de software.

Oracle a partir de la versión 10g Release 2, cuenta con 6 ediciones:

- Oracle Database Enterprise Edition (EE).
- Oracle Database Standard Edition (SE).
- Oracle Database Standard Edition One (SE1).
- Oracle Database Express Edition (XE).
- Oracle Database Personal Edition (PE).
- Oracle Database Lite Edition (LE).

La única edición gratuita es la Express Edition, que es compatible con las demás ediciones de Oracle Database 10gR2 y Oracle Database 11g.

Recientemente, Oracle adquirió a Sun Microsystems y con ella la empresa encargada comercial de MySQL.

4.3 Apache (Servidor Tomcat)

4.3.1 Descripción

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés, a patchy server (un servidor "parcheado") suena igual que Apache Server.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft3).

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

4.3.2 Uso

Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

Apache es el componente de servidor web en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP/Perl/Python (y ahora también Ruby).

Este servidor web es redistribuido como parte de varios paquetes propietarios de software, incluyendo la base de datos Oracle y el IBM WebSphere application server. Mac OS X integra apache como parte de su propio servidor web y como soporte de su servidor de aplicaciones WebObjects. Es soportado de alguna manera por Borland en las herramientas de desarrollo Kylix y Delphi. Apache es incluido con Novell NetWare 6.5, donde es el servidor web por defecto, y en muchas distribuciones Linux.

Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde una computadora personal hacia Internet. Un usuario que tiene Apache instalado en su escritorio puede colocar arbitrariamente archivos en la raíz de documentos de Apache, desde donde pueden ser compartidos.

Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de previsualizar y probar código mientras éste es desarrollado.

Microsoft Internet Information Services (IIS) es el principal competidor de Apache, así como Sun Java System Web Server de Sun Microsystems y un anfitrión de otras aplicaciones como Zeus Web Server. Algunos de los más grandes sitios web del mundo

están ejecutándose sobre Apache. La capa frontal (front end) del motor de búsqueda Google está basado en una versión modificada de Apache, denominada Google Web Server (GWS). Muchos proyectos de Wikimedia también se ejecutan sobre servidores web Apache.

5

Desarrollo de la aplicación

En este capítulo se hará una breve descripción de los programas de los que se compone la aplicación y de las funciones que lo componen.

5.1 Arquitectura de la solución

Para la visualización de los datos obtenidos nos valdremos de un navegador web, a través del cual conectaremos con el servidor tomcat mediante http.

Este Servlet se encarga de dos tareas:

- Recibir los datos de las variables MIB que posee el pc industrial (pc103), a través del mencionado protocolo SNMP, todo mediante UDP
- Actualizar los datos de la base de datos Oracle, la cual guarda un archivo de configuración, y , un histórico de los datos recibidos por el Servlet

Los Pc industriales son los encargados de estableces la comunicación con los diferentes dispositivos a través del protocolo estándar Modbus/TCP.

Estos Pc(s) reciben un archivo de configuración (configuracion_snmp.conf), el cual establece las diferentes OIDs que serán interrogadas mediante SNMP

El programa que comunica con SNMP establece que es el Servlet el que guarda el archivo de configuración, por si se daña o hay perdida de datos este archivo es cargado cada vez que se ejecuta el programa.

5.2 Arquitectura del programa

La programación de los PCs industriales consta de dos programas.

- Uno encargado de la comunicación con el Servlet basado en el protocolo SNMPv1 a través de UDP
- El segundo encargado de la comunicación con los dispositivos SENTRON PAC 4200 a través de Modbus/TCP obviamente, sobre TCP

5.2.1 Programa Servidor SNMP

El programa snmp_scada_modbusTCP es el encargado del procesamiento de los datos, estos datos serán almacenados por el Servlet Oracle en la base de datos.

Este programa consta de una función principal (*Main*), el cual inicialmente se conecta al servidor para obtener el archivo de configuración, llamado servidor_snmp.conf, que poseen las direcciones de los archivos que contienen las variables MIB, estas variables MIB poseen la siguiente estructura:

- :OID :DB :DW :Bytes
  - La OID proporciona la dirección de snmp que posee el dato
  - La variable "DB" indica si los registros son contiguos en la memoria del dispositivo, esto permitirá hacer peticiones de más de un dato, ahorrando tiempo de conexión.
  - La variable "DW" indica la dirección del registro al que se quiere acceder
  - La variable "Bytes" indica la longitud del registro al que se va a acceder

A continuación son creadas las diferentes fifos, se utilizaran una fifo de lectura y una fifo de escritura por cada dispositivo. A estas fifos se les nombrara como **FIFO_LECTURA_XX** y **FIFO_ESCRITURA_XX**, siendo **XX** el número de terminal (ya sea 01,02,….n)

Los datos obtenidos en los dispositivos son almacenados en las fifo de lectura, y las peticiones para obtener dichos datos se crearan en las fifo de escritura, la configuración de las fifos se verá más adelante.

Para la gestión de los datos almacenados en las fifos y la comunicación con el protocolo snmp se crean dos objetos, uno referido a la clase Csnmpd(encargada de la gestión del snmp) y otro referido a la clase CGestionModbusTCP(encargado de la lectura de la cola de las fifos). Estos objetos se ejecutan en paralelo de forma continua, solo terminan si la fifo se encuentra vacía durante un tiempo determinado.

Un esquema sencillo de representar este programa se podría observar en el siguiente flujograma de la Figura 5.1.

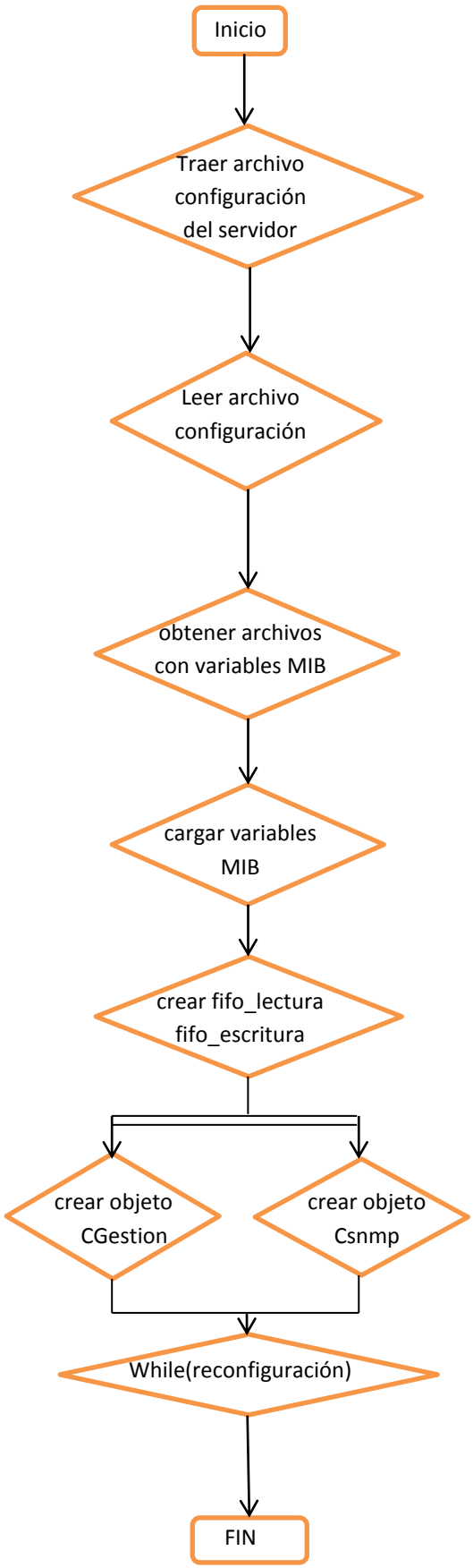

Figura 5.1. Flujograma Main Snmp_scada_modbustcp

5.2.1.1 Clase Csnmpd

A continuación se describirá la clase encargada de la gestión de la comunicación de SNMP

Esta clase consta de dos hilos principales:

- **Snmpd**: es el encargado de la comunicación de SNMP, creando un servidor SNMP al cual se le conectara un cliente que le realizara peticiones SNMP
- **FifoLectura**: este hilo se encarga de leer la fifo de lectura, de la cual se obtendrán los datos que se mandaran al cliente SNMP mediante el hilo Snmpd

5.2.1.1.1 Hilo SNMPD

Una forma sencilla y simplifica de describir los procesos que realiza este hilo seria el mostrado en el flujograma de la Figura 5.2**.**

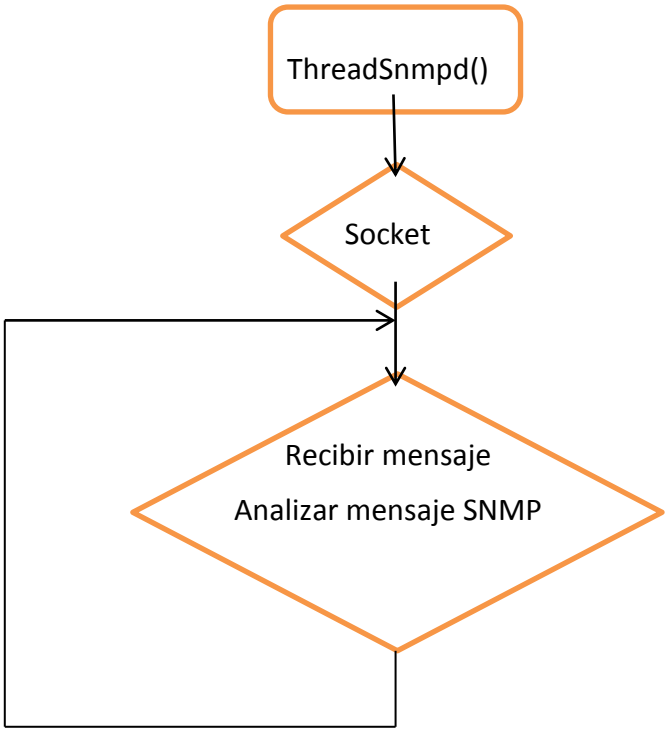

Figura 5.2. **Flujograma Snmpd**

Este hilo crea un socket a la espera de conexiones SNMP en el puerto 161, creando un bucle infinito (**While(1)**) el cual espera y analiza todas la peticiones SNMP entrantes

La función que se encarga de analizar dichos mensajes es la llamada **AnalizarTramaSNMP ().**

Dicha función realizara las siguientes operaciones con la trama SNMP entrante, dicha trama es almacenada en un buffer mediante la función **recvfrom()** (vease Figura 5.3.)

- Llama a la función EsUnSequence (): encargada de comprobar la longitud de la trama SNMP, si esta es correcta el programa continuara
- Llama a la función MaquinaEstadoProtocolo (): encargada de analizar la trama SNMP, comprobando si el formato de la trama se corresponde y si corresponde a la versión de SNMP que estamos manejando, en nuestro caso es la versión v1.
- Comprobación del OID recibido.
    - Si el OID es referido a todo el sistema se procederá a responder con todos los datos almacenados.
    - Si el OID es referido a una variable en concreto, se procederá a enviar solo los datos de esta variable.

La estructura simplificada de esta función la podemos observar en la Figura 5.3**.**, en ella podemos observar que existe un tratamiento diferente para las funciones de obtener datos (snmpget) o para la función de establecimiento de datos (snmpset), esta última se realiza de forma directa sobre el dispositivo, modificando dicho registro y respondiendo al cliente la misma variable que envió.

Los datos que enviados a través de la función snmpget son extraídos del archivo MIB

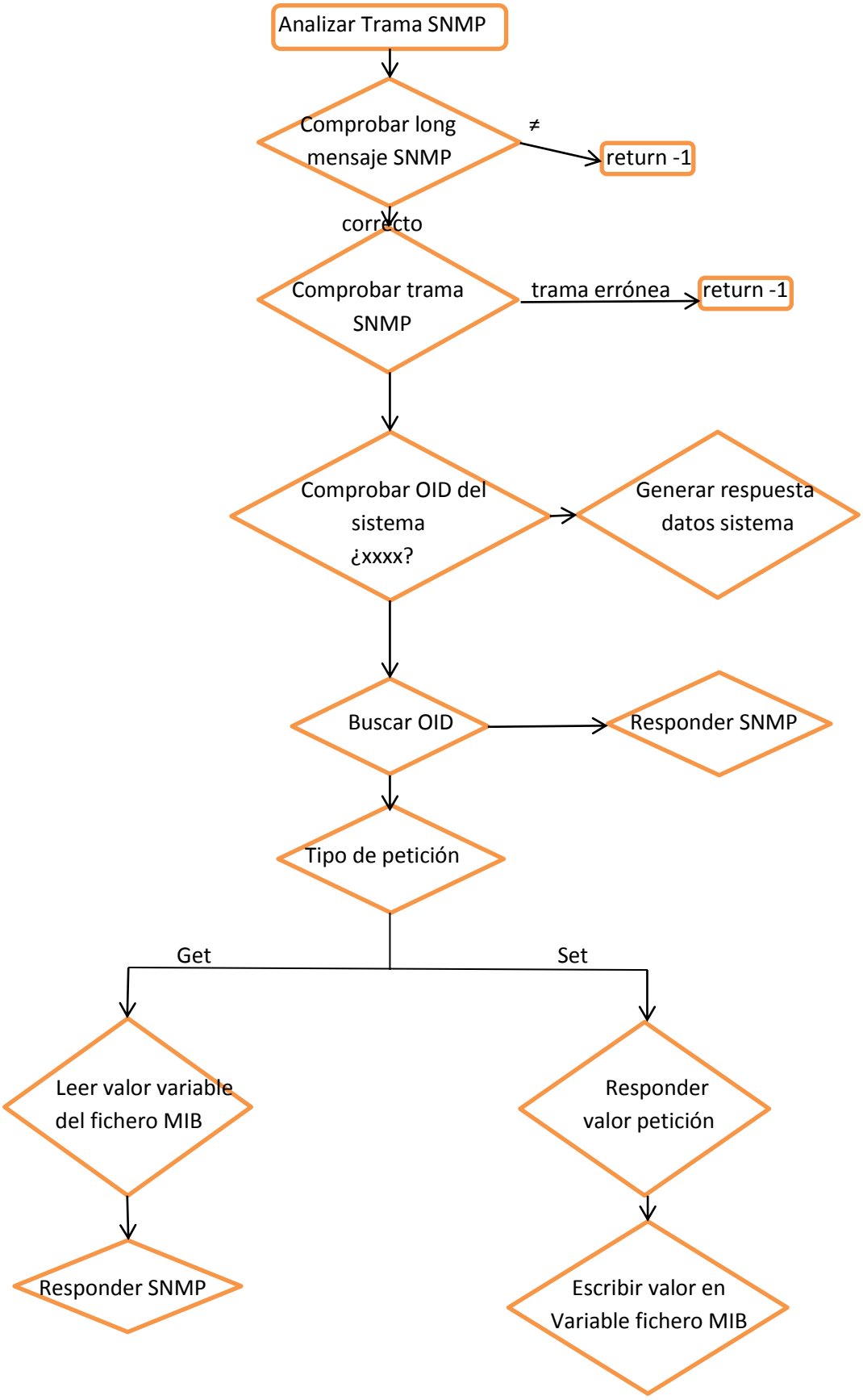

Figura 5.3. **Flujograma función AnalizarTramaSnmp()**

5.2.1.1.2 Hilo FifoLectura

El flujograma que describe este hilo seria el observado en la Figura 5.4

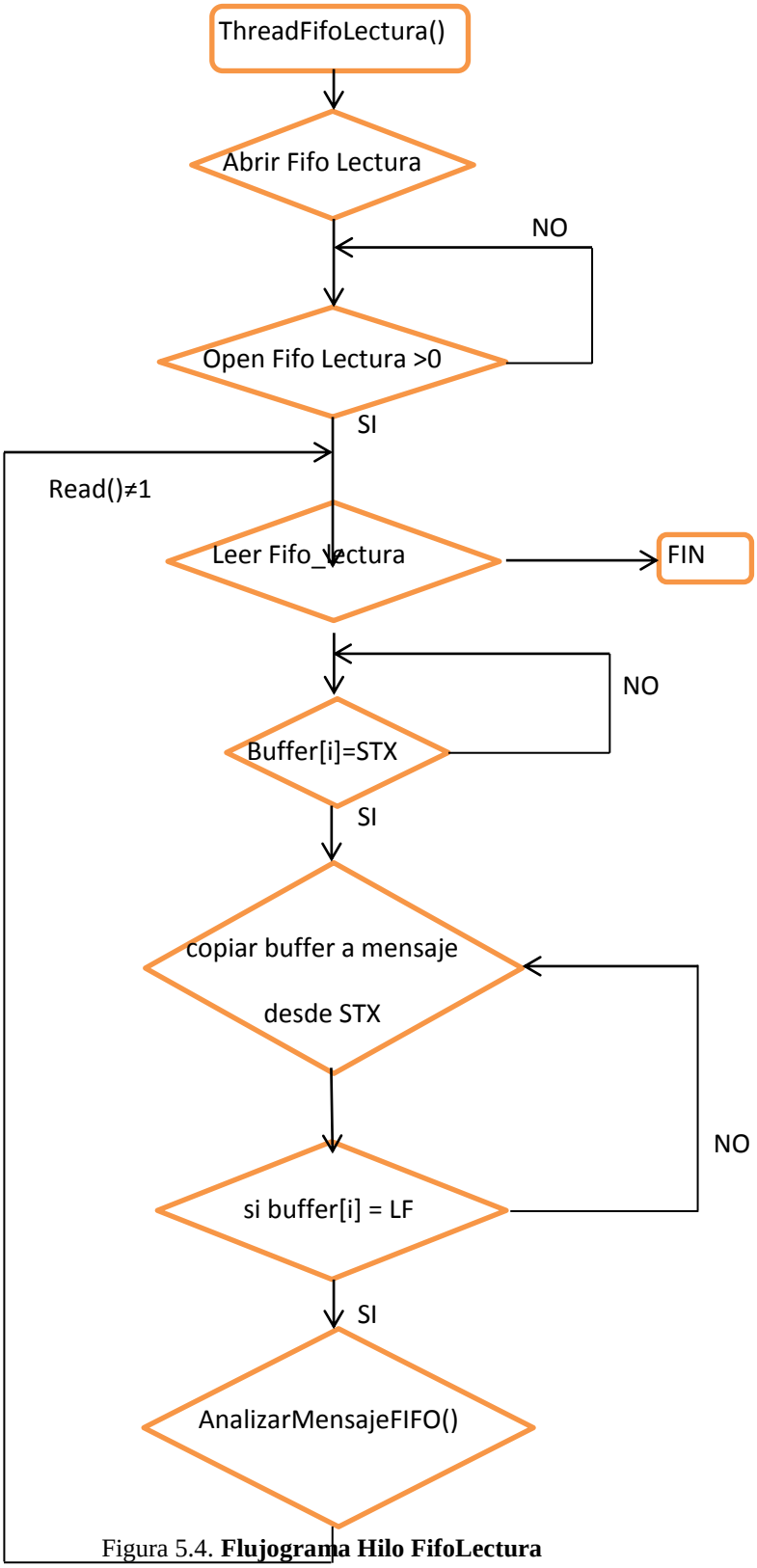

Figura 5.4. **Flujograma Hilo FifoLectura**

Este hilo realiza la lectura de la fifo_lectura de la cual se obtienen los datos que serán enviados a través de SNMP mediante la el hilo SNMPD (explicada en el punto 5.2.1.1.1.).

Los mensajes en la fifo se organizan en líneas, cada línea posee un carácter inicial ("STX") y dos caracteres finales ("CR" y "LF") que están definidos, una vez extraído el mensaje de la fifo se manda a analizar mediante la función AnalizarMensajeFifo().

Esta función se compone de la siguiente estructura:

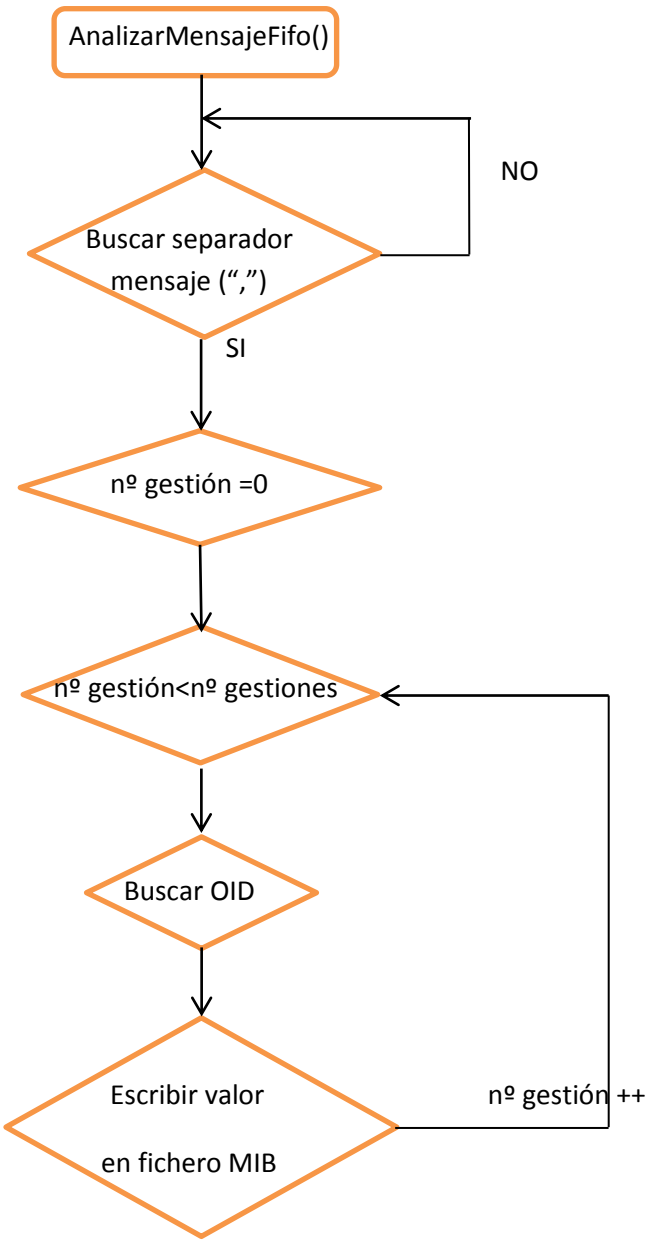

Figura 5.5. **Flujograma función AnalizarMensajeFifo()**

Cada dato pertenece a una OID, esta OID será buscada por la función, que localizara la línea donde se encuentra la variable en el archivo MIB, sustituyéndola, para que cuando el cliente SNMP pida esa variable este actualizada.

Las funciones encargadas de Buscar la OID de la variable y de escribir en el archivo MIB pertenecen a la clase CGestionModbusTCP, que se explica a continuación.

5.2.1.2 Clase CGestionModbusTCP

Clase CGestionModbusTCP gestiona los datos, generando un bucle infinito en el que lee la fifo de lectura y escribe nuevas peticiones de datos en la fifo de escritura. La fifo de escritura es leída por el programa ModbusTCP, el cual es el encargado de hacer las peticiones a los dispositivos.

A parte de la gestión de datos, también se encarga de obtener la fecha y la hora del dispositivo en el que está instalado, en este caso la fecha y la hora la obtendría del pc industrial PC103.

Todo esto se realiza mediante la creación de un hilo llamado Gestion, que se encarga de los siguiente puntos:

- Abre la fifo_lectura
- Carga las variables MIB
- Actualiza la fecha y la hora
- Crea nueva petición de lectura en la fifo_escritura
- Analiza el mensaje de obtenido de la fifo_lectura

La estructura de esta clase la podemos observar en el flujograma de la Figura 5.6

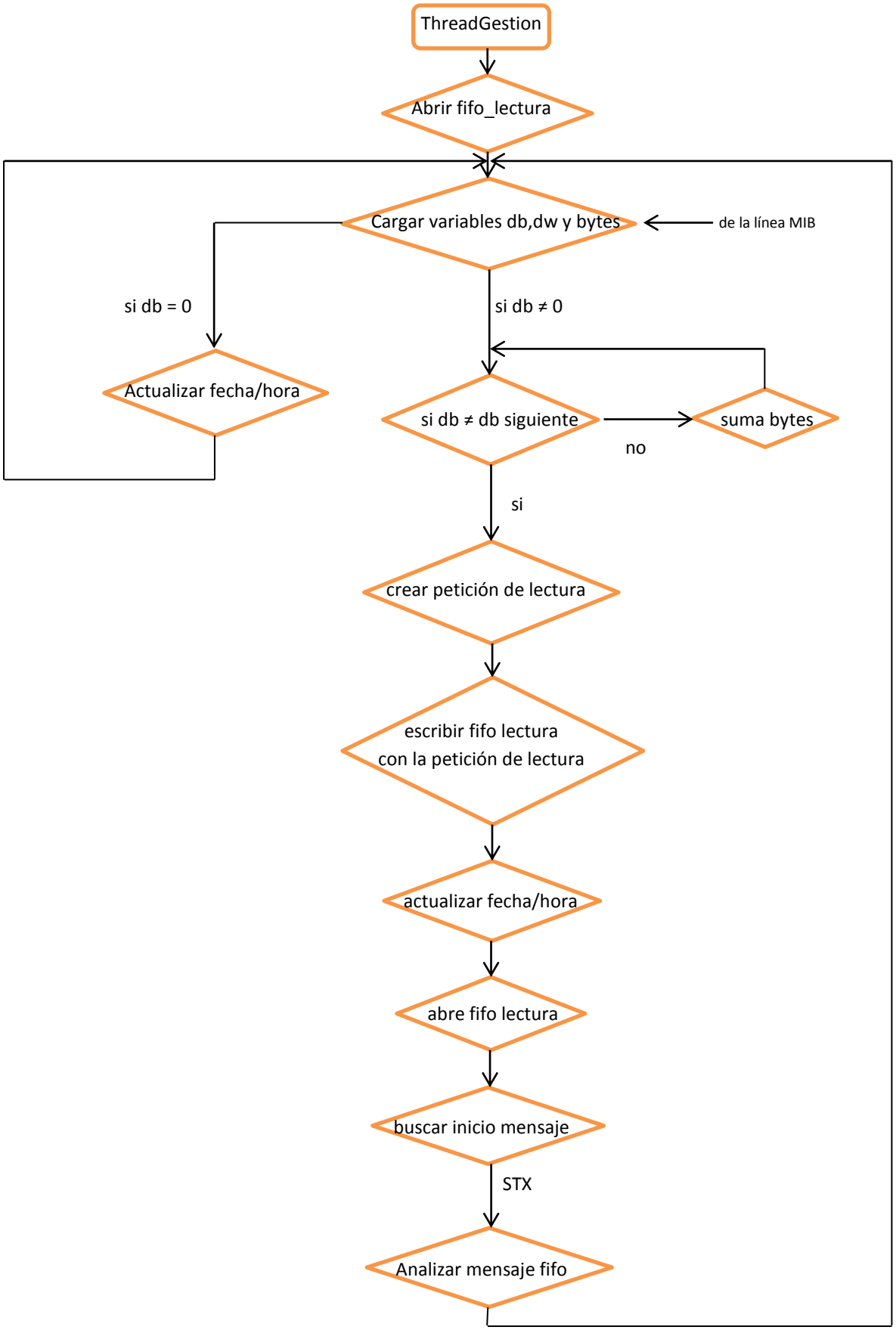
ThreadGestion
Abrir fifo_lectura
Cargar variables db,dw y bytes
de la línea MIB
si db = 0
si db ≠ 0
Actualizar fecha/hora
si db ≠ db siguiente
suma bytes
no
si
crear petición de lectura
escribir fifo lectura
con la petición de lectura
actualizar fecha/hora
abre fifo lectura
buscar inicio mensaje
STX
Analizar mensaje fifo

Figura 5.6. **Flujograma Clase CGestionModbusTCP**

Esta función se encarga de leer el buffer obtenido mediante la petición de lectura de datos del programa ModbusTCP que han sido guardados en la fifo_lectura.

Para realizar el menor número de peticiones posibles la variable DB, cuando es mayor que 1, marca los registros que poseen memoria contigua, o dicho de otro modo, los registros a los que se pueden acceder con una sola petición, ejemplo:

Si los registros 3, 5, 7 que son del tipo float y ocupan 4 bytes cada uno podríamos realizar tres peticiones, una al registro 3 pidiendo 4 bytes otra al registro 5 pidiendo otros 4 bytes,… o pedir al registro 3 12bytes.

La variable DB también indica datos que no nos de los dispositivos, como por ejemplo para un DB=0, DW =1 tendríamos la fecha del sistema, para DB=0, DW=2 obtendríamos la hora y para DB=1, DW=0 obtendríamos el estado del dispositivo.

Los datos obtenidos al leer el buffer pueden ir codificados de diferentes formas, ya sea un entero (int), un float (codificado en IEEE), etc…

Esta función identifica el tipo de dato que es y fuerza el tipo para que en la MIB siempre se guarden los datos en forma de carácter.

La función que decodifica el buffer posee el nombre de AnalizarMensajeFifo_gestion(), posee la siguiente estructura:

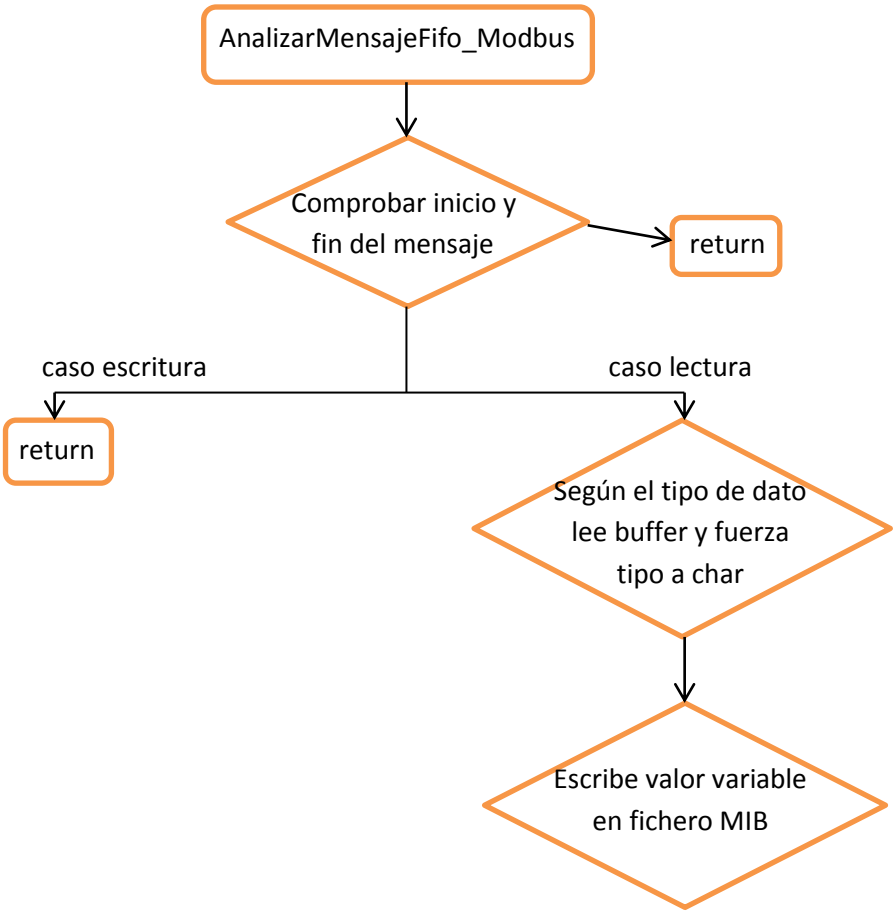

Figura 5.7. **Flujograma función AnalizarMensajeFifo_gestion**

5.2.2 Programa ModbusTCP

Este programa es el encargado de leer de la fifo las peticiones de lectura y de realizar dichas peticiones al dispositivo.

Se deberá arrancar un programa ModbusTCP por cada dispositivo, pasándole como parámetros:

- La ip del dispositivo
- El nombre de la fifo de lectura y la fifo escritura, como se menciona en el programa snmp_scada_modbustcp el nombre de la fifo ser:

  FIFO_LECTURA_XX y FIFO_ESCRITURA_XX

  Siendo XX en número de dispositivo

Este programa crea un hilo encargado de la lectura de la fifo y las peticiones a los dispositivos. La estructura que tomaría dicho hilo, llamado CModbusTCP sería la observada en la Figura 5.8

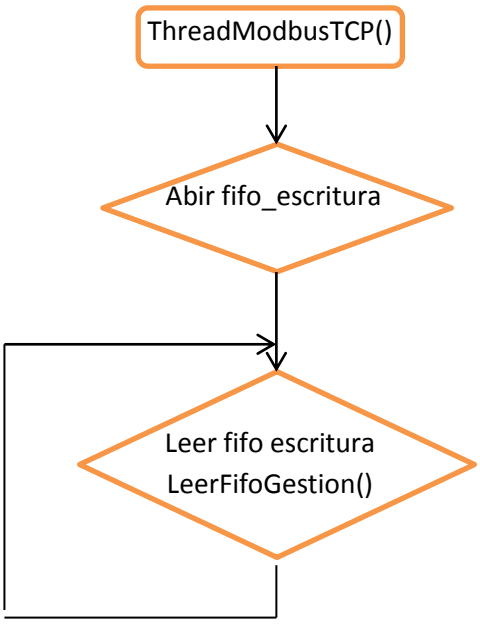

Figura 5.8. **Flujograma Hilo ModbusTCP**

Una vez abierta la fifo de escritura se mantendrá un bucle infinito para leerla, la función encargada de realizar esta operación sería LeerFifoGestion(), y posee la siguiente estructura:

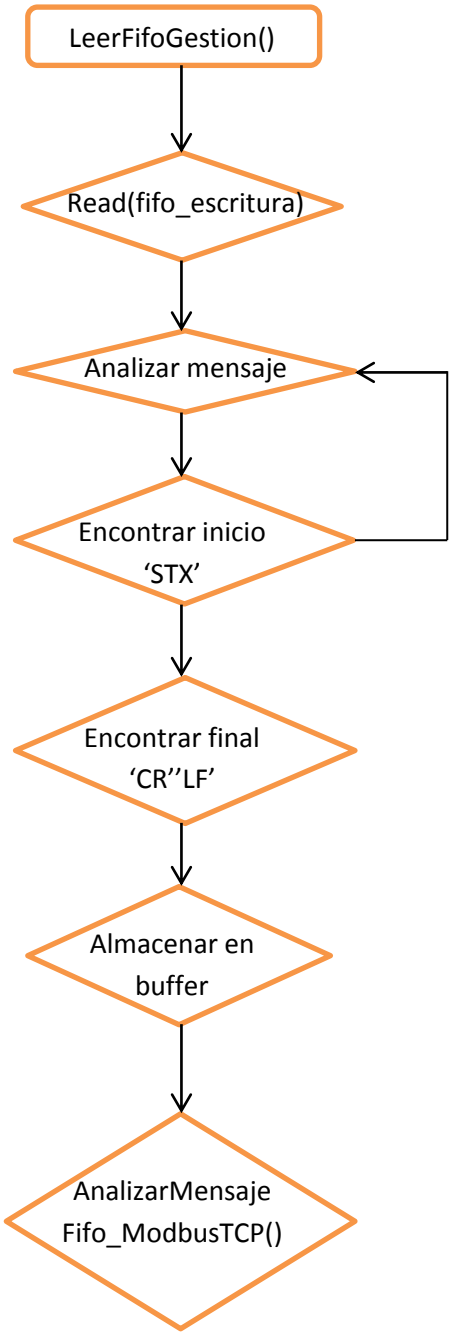

Figura 5.9. **Flujograma función LeerFifoGestion()**

Esta función lee la fifo de escritura, recogiendo los datos de esta en un buffer.

A continuación genera un bucle para encontrar el principio del mensaje, definido como “STX”, una vez encontrado copiara dicho mensaje hasta llegar a los comandos “CR” y “LF” que indican el fin del mensaje. Una vez hecho esto se procederá a analizar dicho mensaje mediante la función AnalizarMensajeFifo_ModbusTCP().

Esta función es la encargada de analizar el mensaje procedente de la fifo, conectar con el termina, realizar las peticiones de datos y guardar dichos datos en la fifo de lectura.

Esta sería la estructura que seguirá dicha función:

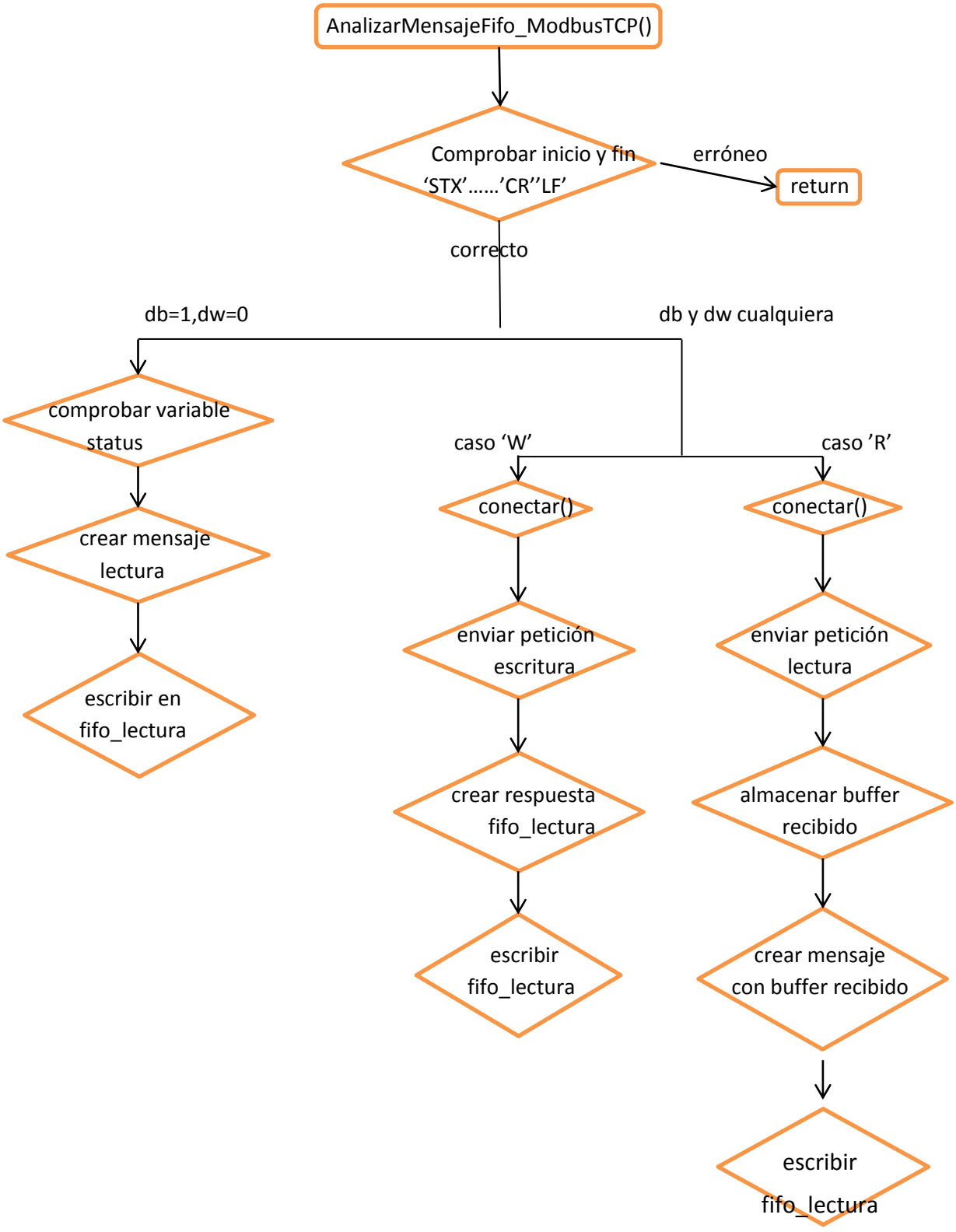

Figura 5.10. **Flujograma función AnalizarMensajeFifo_ModbusTCP()**

5.2.3 Arquitectura de los Archivos FIFO

Los archivos fifo, ya sea la fifo de lectura o la fifo de escritura, se organizan en líneas, en esas líneas se crean las peticiones de lectura y de escritura que serán utilizadas por los programas Csnmpd y Cmodbustcp para el funcionamiento del programa.

El protocolo que sigue la fifo se puede describir de la siguiente manera:

Petición de lectura:

'STX'	**'R'**	**dbH**	**dbL**	**dwH**	**dwL**	**BytesH**	**BytesL**	**'CR'**	**'LF'**
1 byte	1 byte	2 bytes		2 bytes		2 bytes		1 byte	1 byte

Tabla 5.1. Petición fifo lectura

Respuesta de lectura:

'STX'	**'R'**	**dbH**	**dbL**	**dwH**	**dwL**	**XXXXXX**	**'CR'**	**'LF'**
1 byte	1 byte	2 bytes		2 bytes		n bytes	1 byte	1 byte

Tabla 5.2. Respuesta fifo lectura

Petición de escritura:

'STX'	**'R'**	**dbH**	**dbL**	**dwH**	**dwL**	**BytesH**	**BytesL**	**XXXX**	**'CR'**	**'LF'**
1 byte	1 byte	2 bytes		2 bytes		2 bytes		n bytes	1 byte	1 byte

Tabla 5.3. Petición fifo escritura

Respuesta de escritura:

'STX'	**'R'**	**dbH**	**dbL**	**dwH**	**dwL**	**0x00**	**0/1**	**'CR'**	**'LF'**
1 byte	1 byte	2 bytes		2 bytes		2 bytes		1 byte	1 byte

Tabla 5.4. Respuesta fifo escritura

Los campos 'STX, 'CR' y 'LF' son iguales para todas las peticiones y están definidos en el programa

En la respuesta de lectura el campo XXXXX equivale a los n bytes que se recibieron en la petición de lectura, estos n bytes son rellenados con los datos recibidos por los terminales para dar forma a la respuesta de lectura

Los XXXX bytes que se muestra en la petición de escritura son los bytes que serán modificados en la memoria del dispositivo.

6

Conclusiones y trabajos futuros

6.1 Conclusiones

Una de las principales conclusiones que se puede sacar de este proyecto es su funcionalidad, ya que, aparte de encontrarse operativo en una de las empresas líderes en España en elaboración de alimentos cárnicos, permite, no solo ser usado para un tipo específico de terminal, sino que, viendo la evolución de la tecnología, podrá ser empleado en los nuevos terminales de medición, recogiendo información de estos para el SCADA. Todo esto permitirá a la empresa realizar el control y optimización de las instalaciones, suponiendo una mejora en el rendimiento.

Además, este proyecto me ha sido de gran utilidad para ahondar en el lenguaje de programación C++, ya que en lugar de utilizar una librería externa publicada por otro autor, la he creado desde un principio, para lo que he tenido que utilizar varias aplicaciones como **Wireshark** [11], que permite analizar las comunicaciones entre terminales, **KDevelop4**[12], para compilar los programas con librerías externas ARM9 para los PCs industriales 103. Estos PCs industriales están repartidos por todas las instalaciones de la fábrica, y gracias a las opciones que proporciona Linux en modo consola, como por ejemplo **SSH**[13], es posible el manejo de dichos PCs desde cualquier punto en el que se tenga acceso a la red en la que están instalados.

En resumen, creo que este proyecto aparte de suponer una aplicación real que se encuentra en funcionamiento en una empresa como ElPozo Alimentación, ha supuesto un enriquecimiento y puesta en práctica de mis conocimientos sobre lenguajes de programación (C++) adquiridos en la UPCT y la aplicación práctica de los conocimientos adquiridos en la asignatura Comunicaciones Industriales, que fueron de gran ayuda para el previo conocimiento de este protocolo y la realización de este trabajo.

---

[11] Wireshark, antes conocido como Ethereal, es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta didáctica para educación

[12] KDevelop es un entorno de desarrollo integrado para sistemas GNU/Linux y otros sistemas Unix, publicado bajo licencia GPL, orientado al uso bajo el entorno gráfico KDE, aunque también funciona con otros entornos, como Gnome.

[13] SSH (Secure SHell, en español: intérprete de órdenes segura) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos,

6.2 Trabajos futuros

Como ocurre siempre que se aborda un proyecto de este tipo, siempre es posible plantear líneas para la mejora y ampliación del trabajo realizado. Entre éstas, cabe destacar las siguientes por su interés:

- En el programa snmp_scada_modbustcp se podría implementar la versión v3 de SNMP, aunque esta resulte algo más compleja que la versión v1.
- En general se podría agilizar el proceso de obtención de los datos, manteniendo el socket abierto.
- Asimismo, también sería interesante optimizar el código, reduciendo la cantidad de operaciones, haciéndolo más fluido y simplificado.

Bibliografía

1. Julián Cócera Rueda, Pedro Morcillo Ruiz. "Comunicaciones Industriales". Internacional Thomson Editores Spain Paraninfo S.A. 2004.

1. Castro Gil, Manuel-Alonso ... [et al.]. "Comunicaciones industriales: principios básicos". Universidad Nacional de Educación a Distancia. 2007.

2. Castro Gil, Manuel-Alonso ... [et al.]. "Comunicaciones industriales: sistemas distribuidos y aplicaciones". Universidad Nacional de Educación a Distancia. 2007.

3. *Modbus*

   http://es.wikipedia.org/wiki/Modbus

   http://www.modbus.org/

4. Simple Network Management Protocol RFC1157

   http://www.faqs.org/rfcs/rfc1157.html

5. Núcleo *Linux*

   http://es.wikipedia.org/wiki/Núcleo_**Linux**

6. SIEMENS | siemens.com

   http://www.**siemens**.com/answers