ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO, DISEÑO Y DESARROLLO DE UN SOFTWARE DE MONITORIZACIÓN DE RED EN DISPOSITIVOS INALÁMBRICOS (PDA) PARA LA ADMINISTRACIÓN Y GESTIÓN DE RED

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

FLORES MEZA JOSÉ ALEJANDRO
(jose_floresm@hotmail.com)

JIMÉNEZ HEREDIA JUAN PABLO
(juanpa_jimenez@hotmail.com)

DIRECTOR: ING. XAVIER CALDERÓN, MSC.

Quito, Julio 2008

DECLARACIÓN

Nosotros, FLORES MEZA JOSÉ ALEJANDRO Y JIMÉNEZ HEREDIA JUAN PABLO, declaramos que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____	_____
Flores Meza José Alejandro	Jiménez Heredia Juan Pablo

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Flores Meza José Alejandro y Jiménez Heredia Juan Pablo, bajo mi supervisión.

_____

Ing. Xavier Calderón, Msc
DIRECTOR DEL PROYECTO

DEDICATORIA

El presente proyecto de titulación se lo dedico a Dios por guiarme y ayudarme en cada instante de mi vida. A mis padres, hermanas y amigos por su incondicional apoyo y orientación.

José Flores.

DEDICATORIA

El presente proyecto de titulación lo dedico a mis padres que con su apoyo esfuerzo y comprensión me supieron llevar siempre adelante. A mi hija que con su llegada a este mundo supo llenarme de amor y fortaleza.

Juan Pablo Jiménez

AGRADECIMIENTOS

Nuestro mas grande agradecimiento a Dios sobre todas las cosas, por darnos vida, salud y entendimiento para desarrollar y concretar satisfactoriamente éste proyecto de titulación.

A nuestros padres y hermanos por su apoyo incondicional.

Al Ingeniero Xavier Calderón por su confianza y amistad en la dirección y supervisión del presente proyecto.

A todas las personas que directa o indirectamente nos apoyaron con sus conocimientos, palabras de aliento y comprensión. ¡Muchas Gracias!

Los autores.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ESPACIOS DE CÓDIGO

RESUMEN

La presente tesis “Estudio, Diseño y Desarrollo de un software de Monitorización de Red en dispositivos inalámbricos (PDA) para la administración y gestión de red está constituida por cuatro capítulos los cuales contienen lo siguiente:

Capítulo Primero. Muestra un resumen de las principales características, funcionalidades y aplicaciones de las distintas clases de PDA´s dependiendo de su fabricante, versión y sistema operativo que ejecute.

Capítulo Segundo. Explica el proceso de desarrollo del software de monitorización de red, determinando los requerimientos funcionales y no funcionales de hardware y de software necesarios para su correcto funcionamiento. Se determina también el tipo de lenguaje de programación que facilite el desarrollo de aplicaciones sobre PDA´s. Además indica mediante diagramas de caso de uso, secuencia y colaboración el proceso e interacción de las diferentes clases y funciones de la aplicación.

Capítulo Tercero. Muestra las pruebas y resultados que se obtienen de la utilización del software de monitorización de red en dispositivos inalámbricos en la red de pruebas definida. Se analiza y tabula los resultados obtenidos luego del uso de todas las herramientas que presenta esta aplicación como SNMP (Get, Set y Trap), Ping y Browser. Con la ayuda de Ethereal, que es un software que me permite capturar los paquetes enviados entre la NMS y los dispositivos de red gestionados, demostramos que nuestra aplicación trabaja con SNMP de la versión uno y dos.

Capítulo Cuarto. Contiene las principales conclusiones y recomendaciones que se obtienen al realizar este proyecto de titulación.

PRESENTACIÓN

Con el avance de la tecnología y el acelerado crecimiento de las redes de datos surge la necesidad de implementar un sistema de gestión, ya que permite la coordinación, el despliegue y la interacción entre el hardware y software. Esto permite a los administradores de red desde un punto centralizado configurar, probar, monitorizar, sondear, analizar, evaluar y controlar todos los elementos de red en tiempo real.

Al implementar una aplicación que me permita monitorizar y administrar una red empresarial desde un dispositivo inalámbrico como es la PDA, se proporciona una rápida iteración y respuesta ante acontecimientos inesperados, facilitando las actividades del administrador de red, ya que no se le exige estar todo el tiempo tras un escritorio.

La base del software de monitorización de red es el protocolo SNMP (Simple Network Management Protocol) el cual permite el intercambio de información de administración entre dispositivos de red. Los dispositivos administrados son supervisados y controlados usando los comandos básicos SNMP de lectura, escritura y notificación, los mismos que interactúan con los agentes de los dispositivos para obtener la información de administración anidada en las MIB´s (Management Information Base) de cada equipo.

Además se implementará el uso de herramientas simples de monitoreo de redes como PING que no es soportada por el sistema operativo Windows CE para PDA´s. El uso de esta herramienta es indispensable ya que permite determinar la conectividad entre los equipos de red.

Por último se hace uso de herramientas adicionales como el browser la misma que permitirá la configuración remota de dispositivos que tengan esta opción y polling para determinar el estado de conexión de los dispositivos gestionados en la red.

CAPÍTULO 1

PDA (PERSONAL DIGITAL ASSISTANT)

1.1 CARACTERÍSTICAS

1.1.1 INTRODUCCIÓN

PDA (***Asistente Personal Digital***) es un computador de mano que fue diseñado para satisfacer las demandas de tamaño y movilidad de equipos computacionales, ya que este dispositivo puede transportarse fácilmente por su reducido tamaño.

Inicialmente las PDAs se usaron como agendas electrónicas prestando una gran variedad de servicios dependiendo de su casa fabricante. Entre las principales aplicaciones que encontramos en una PDA son: Calendario, lista de contactos, bloc de notas, recordatorios y sistema de reconocimiento de escritura.

En la actualidad se ha evolucionado el uso de este dispositivo ya que se lo puede usar como una computadora doméstica, únicamente teniendo limitaciones por el tamaño de la pantalla ya que es posible crear documentos, navegar en Internet, enviar y recibir correos electrónicos, escuchar música, ver películas, jugar y una infinidad de aplicaciones similares a la de cualquier desktop.

Con la evolución de las PDA se ha introducido un nuevo dispositivo al mercado, los Smartphone o teléfono inteligente, el cual integra la funcionalidad de un teléfono móvil y una PDA. Los Smartphone integran funciones como comunicarse a través de Wi-Fi y bluetooth, disponen de conexión a Internet, y permiten el envío de mensajería y correo electrónico.

Una PDA necesita al igual que una PC un sistema operativo que posibilite ejecutar el software instalado en el dispositivo. Entre los principales Sistemas Operativos que encontramos en el mercado sobresalen Palm OS y Windows Mobile que es el sucesor de Windows CE y conocido también como Pocket PC.

En menor cantidad encontramos dispositivos como smartphones o teléfonos inteligentes de marca Nokia, Sony, Motorola y Siemens entre los más difundidos que utilizan Symbian OS como su sistema operativo. También encontramos algunas versiones de Linux personalizadas para ejecutarse en este tipo de dispositivos.

1.1.2 HISTORIA

En 1989, el Atori Portfolio, el primer ordenador PC compatible PDA que fue construido utilizando un procesador 80C88 a 4,9152 MHz. Aunque técnicamente clasificado como palmtop. Le siguieron otros dispositivos como los Psion Organiser, el Sharp Wizard o la Amstrad Penpad que fueron sentando la base de las funcionalidades de las PDAs.

La PDA como tal tuvo su lanzamiento oficial el 7 de enero de 1992 por John Sculley al presentar el Apple Newton, en el Consumer Electronics Show de Las Vegas en los Estados Unidos. Sin embargo fue un sonoro fracaso financiero para la compañía Apple, dejando de venderse en 1998.

La tecnología no estaba completamente desarrollada y el reconocimiento de escritura en la versión original era bastante impreciso, entre otros problemas. Aún así, este aparato ya contaba con todas las características del PDA moderno: pantalla sensible al tacto, conexión a una computadora para sincronización, interfaz de usuario especialmente diseñada para el tipo de máquina, conectividad a redes vía módem y reconocimiento de escritura.

En 1995 con la aparición de la empresa Palm comenzó una nueva etapa de crecimiento y desarrollo tecnológico para el mercado de estos dispositivos. Tal fue el éxito de esta empresa que las PDAs son llamadas Palms.

La introducción de Microsoft Windows CE (2000) y Windows Mobile (2003) en el sector las dotó de mayores capacidades multimedia y conectividad.

La introducción de los Smartphones (híbridos entre PDA y teléfono móvil) trajeron nuevos competidores al mercado y permitieron a los usuarios tener varias aplicaciones en sus teléfonos móviles. Con la aparición de los Smartphones propuso la vuelta de Symbian OS que es un sistema operativo que había abandonado el mercado de las PDAs y ordenadores de mano en favor de los móviles.

Las PDAs de hoy en día nos permiten utilizar comunicaciones inalámbricas como Bluetooth, WiFi, IrDA, GPS, entre otros que las hacen tremendamente atractivas hasta para cosas tan inverosímiles como su uso en domótica, gestión de equipos o como navegadores GPS.

1.1.3 COMPONENTES DE UNA PDA

1.1.3.1 Procesador

Los ordenadores de bolsillo suelen incluir procesadores de arquitectura diferente a los que encontramos en nuestros ordenadores personales, ya que han de tener un consumo muy reducido de potencia y adecuarse a las características físicas de los PDA.

Algunas aplicaciones específicas están diseñadas para usar un tipo concreto de procesador. Por ello, es preferible asegurarnos la compatibilidad de ambos. En todo caso, las aplicaciones suelen estar limitadas por el sistema operativo que utilice el dispositivo.

1.1.3.2 Software

La funcionalidad de los ordenadores de bolsillo sólo está limitada por la de las aplicaciones que en ellos se instalen. Lo habitual es que los ordenadores de esta clase incorporen una serie de aplicaciones más comunes: agenda de contactos, calendario, notas, gestor de correo electrónico, etc.

Mas adelante, podremos añadir cualquier otro programa que adquiramos o descarguemos (dado que hay una buena selección de software gratuito) y que resuelva nuestras necesidades específicas.

1.1.3.3 Sistema operativo

El sistema operativo es un programa que permite la comunicación del usuario con la PDA y gestiona sus recursos de una forma eficaz. Los dos sistemas operativos para PDAs más extendidos del mercado son Palm OS, de Palm Inc., y Windows Mobile, de Microsoft, además existen otros sistemas operativos en el mercado, pero no han llegado a tener gran difusión. Todos éstos serán analizados mas adelante.

1.1.3.4 Conectividad

El ordenador de bolsillo intercambia datos con el exterior a través de los puertos de comunicación. Existen dos tipos de puerto: por cable e inalámbrico. Los primeros pueden usar el puerto serie llamado RS-232C; o el puerto USB, que es el estándar para los ordenadores de mano actuales.

Entre los inalámbricos, los infrarrojos son la vía más utilizada. Además hay otros sistemas sin las limitaciones de éste, como falta de rango y necesidad de mantener una línea visual entre los dispositivos, entre estas tecnologías tenemos: Bluetooth y WiFi.

Actualmente un mismo dispositivo incorpora dos o más sistemas de comunicación simultáneamente. Algunos de ellos incluso pueden tenerlos todos.

1.1.3.5 Memoria

La memoria es la encargada de almacenar los datos de estos ordenadores. Por lo general es relativamente reducida. Algunos sólo tienen 2 o 4 MB, pero la mayoría se sitúa sobre los 16, 32 o 64 MB. Los modelos más recientes están equipados

hasta con 256 MB, lo que aporta una gran comodidad a la hora de cargar varias aplicaciones, sobre todo las de multimedia.

Por otro lado, algunos dispositivos disponen de un espacio para que se pueda añadir tarjetas de memoria, aumentando así la capacidad del dispositivo. Existen varios modelos estándar:

1.1.3.5.1 Compact Flash: existen dos formatos que solo se diferencian por su espesor: el tipo I (3,3 mm) y el tipo II (5 mm). La mayor parte de las máquinas sólo aceptan las tarjetas del tipo I, cuya capacidad varía de 2 a 192 MB, aunque las tarjetas de más capacidad pueden almacenar varios GB.

1.1.3.5.2 Smart Media: esta tarjeta extra plana (0,76 mm) se utiliza muy poco en los ordenadores de bolsillo, a pesar de sus reducidas dimensiones (34 x 45 mm). Su capacidad máxima alcanza los 8 GB.

1.1.3.5.3 MMC/SD: la más pequeña (24x32 mm y espesor de 1,4 mm). Se presenta en dos versiones: la Multimedia CARD y la Secure Digital CARD. Esta última se diferencia por funciones internas destinadas a controlar la copia cuando contiene ficheros de música comprimidos. Su capacidad llega a 1 GB.

1.1.3.5.4 Memory Stick: usadas únicamente por los ordenadores Clié de Sony, pueden llegar a 2 GB de tamaño.

1.1.3.6 Ranuras de expansión

No todos los ordenadores de bolsillo disponen de espacio para alojar una tarjeta de memoria. La importancia de este criterio varía en función del uso más o menos intensivo que vayamos a hacer de nuestro ordenador de bolsillo, así como del espacio de almacenamiento necesario para los archivos.

1.1.3.7 Teclado o lápiz

La mayoría de PDAs no tienen un verdadero teclado. Para introducir texto existen dos métodos: el primero, un pequeño teclado que aparece en la pantalla, cuyas letras se seleccionan con el lápiz incorporado. El segundo, un sistema de reconocimiento de letras, cifras y caracteres especiales (signos de puntuación, operaciones aritméticas, etc.). Además, la mayoría de estos ordenadores soporta la conexión de un pequeño teclado externo, que nos facilitará la entrada de datos.

1.1.3.8 Pantalla

El tamaño de las pantallas LCD está en consonancia con el de los ordenadores de bolsillo: son bastante reducidas. La mayor parte de los modelos sin teclado presentan una definición comprendida entre 160x160 y 320x240 píxeles, mientras que los modelos con teclado son más ricos en resolución, que puede llegar hasta los 640x240 píxeles.

El tamaño puede ser un criterio importante para aquellos que deseen utilizar una hoja de cálculo. El color no es realmente imprescindible, sólo si deseas completar el ordenador con un módulo de cámara digital o si se desea consultar vídeos.

1.1.3.9 Batería

Un ordenador de bolsillo puede funcionar gracias a su batería interna o a las pilas, que se pueden cambiar por baterías recargables. En algunas PDA, una pila botón asegura la conservación de los datos en la memoria RAM cuando la alimentación principal se agota.

Uno de los problemas más comunes en este tipo de dispositivos es el fallo de la batería. Con el paso del tiempo y debido a la recarga, ésta acaba estropeándose o reduciendo notablemente la autonomía.

1.1.3.10 Accesorios integrados

1.1.3.10.1 Altavoces, micrófono y toma de auriculares: no todos los modelos los llevan, pero la tendencia es que los vayan incorporando. Hace algunos años el micrófono se utilizaba para grabar datos que luego eran recuperados gracias al altavoz. Con la llegada del MP3, apareció la toma de auriculares.

1.1.3.10.2 Base de conexión: la mayor parte de los ordenadores de bolsillo se entregan con el llamado Cradle, un dispositivo unido a la PDA, por lo que no es necesario enchufarlo cada vez que hay que hacer una sincronización o un intercambio de datos.

1.1.3.10.3 Lector biométrico: se trata de un sistema de seguridad para evitar el acceso a los datos contenidos en el ordenador por personas "no autorizadas". Básicamente es un pequeño lector de huellas dactilares que actúa de igual forma que la tradicional protección por contraseña.

1.1.4 SISTEMAS DE COMUNICACIÓN INALÁMBRICA UTILIZADOS POR LAS PDA.

Una PDA al ser un dispositivo móvil es necesario que utilice sistemas de comunicación inalámbricos para comunicarse con otros dispositivos, ya que un cable limitaría su utilización.

Entre los principales sistemas de comunicación inalámbricas empleadas por la PDA tenemos:

1.1.4.1 Infrared Data Association

La comunicación por infrarrojos es una de las comunicaciones inalámbricas más extendidas en todo el mundo debido a su facilidad de implementación y su reducido costo.

Infrared Data Association es una tecnología basada en rayos luminosos que se mueven en el espectro infrarrojo. Los estándares IrDA soportan una amplia gama de dispositivos eléctricos, informáticos y de comunicaciones, permite la comunicación bidireccional entre dos extremos a velocidades que oscilan entre los 9.600 bps y los 4 Mbps.

Esta tecnología se encuentra en muchos ordenadores portátiles, y en un creciente número de teléfonos celulares, sobre todo en los de fabricantes líderes como Nokia y Ericsson.

1.1.4.1.1 Características

Las PDA que poseen un puerto Infrarrojo (IrDA) incluido poseen las siguientes características:

- Adaptación compatible con futuros estándares.
- Cono de ángulo estrecho de 30º.
- Opera en una distancia de 0 a 1 metro.
- Conexión universal sin cables.
- Comunicación punto a punto.
- Soporta un amplio conjunto de plataformas de hardware y software.

1.1.4.1.2 Protocolos IrDA

Entre los Protocolos IrDA tenemos:

- PHY (Physical Signaling Layer) establece la distancia máxima, la velocidad de transmisión y el modo en el que la información se transmite.
- IrLAP (Link Access Protocol) facilita la conexión y la comunicación entre dispositivos.
- IrLMP (Link Management Protocol) permite la multiplexación de la capa IrLAP.
- IAS (Information Access Service) actúa como unas páginas amarillas para un dispositivo.

- Tiny TP mejora la conexión y la transmisión de datos respecto a IrLAP.
- IrOBEX diseñado para permitir a sistemas de todo tipo y tamaño intercambiar comandos de una manera estandarizada.
- IrCOMM para adaptar IrDA al método de funcionamiento de los puertos serie y paralelo.
- IrLan permite establecer conexiones entre ordenadores portátiles y LANs de oficina.

1.1.4.2 Bluetooth

Bluetooth define un estándar global de comunicación inalámbrica que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia segura, globalmente y sin licencia de corto rango (hasta 10 metros).

Los principales objetivos de esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

1.1.4.2.1 Características

La especificación de Bluetooth define un canal de comunicación de máximo 720 kbps (1 Mbps de capacidad bruta) con rango óptimo de 10 metros (opcionalmente 100 m con repetidores).

La frecuencia de radio con la que trabaja está en el rango de 2,4 a 2,48 GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir en Full Duplex con un máximo de 1600 saltos/s. Los saltos de frecuencia se dan entre un total de 79 frecuencias con intervalos de 1 MHz; esto permite dar seguridad y robustez. La potencia de salida para transmitir a una distancia máxima de 10 metros es de 0

dBm (1 mW), mientras que la versión de largo alcance transmite entre 20 y 30 dBm (entre 100 mW y 1 W).

El Hardware que compone el dispositivo Bluetooth está compuesto por dos partes:

- **Dispositivo de radio**, encargado de modular y transmitir la señal
- **Controlador digital**, compuesto por una CPU, por un procesador de señales digitales (DSP - Digital Signal Processor) llamado Link Controller y de los interfaces con la PDA.

El LC o Link Controller está encargado de hacer el procesamiento de la banda base y del manejo de los protocolos ARQ y FEC de capa física. Además, se encarga de las funciones de transferencia tanto asíncrona como síncrona, codificación de Audio y cifrado de datos.

El CPU del dispositivo se encarga de atender las instrucciones relacionadas con Bluetooth de la PDA, para así simplificar su operación. Para ello, sobre el CPU corre un software denominado Link Manager que tiene la función de comunicarse con otros dispositivos por medio del protocolo LMP[1].

Entre las tareas realizadas por el LC y el Link Manager, destacan las siguientes: envío y recepción de datos, empaginamiento y peticiones, determinación de conexiones, autenticación, negociación y determinación de tipos de enlace.

1.1.4.3 Wi-Fi

Wi-Fi (Wireless Fidelity) es un conjunto de estándares para redes inalámbricas basados en las especificaciones 802.11. Creado para ser utilizado en redes WLAN.

[1] Link Manager Protocol

El estándar original de Wi-FI es el IEEE 802.11, el cual tenía velocidades de 1 hasta 2 Mbps y trabajaba en la banda de frecuencia de 2,4 GHz.

1.1.4.3.1 Protocolos

1.1.4.3.1.1 802.11b

Tiene una velocidad máxima de transmisión de 11 Mbps y utiliza el mismo método de acceso CSMA/CA definido en el estándar original (802.11). El estándar 802.11b funciona en la banda de 2.4 GHz. Debido al espacio ocupado por la codificación del protocolo CSMA/CA, en la práctica, la velocidad máxima de transmisión con este estándar es de aproximadamente 5.9 Mbps sobre TCP y 7.1 Mbps sobre UDP.

1.4.3.1.2 802.11a

El estándar 802.11a utiliza el mismo juego de protocolos de base que el estándar original, opera en la banda de 5 GHz y utiliza 52 subportadoras OFDM (Orthogonal Frequency-Division Multiplexing) con una velocidad máxima de 54 Mbps, lo que lo hace un estándar práctico para redes inalámbricas con velocidades reales de aproximadamente 20 Mbps . La velocidad de datos se reduce a 48, 36, 24, 18, 12, 9 o 6 Mbps en caso necesario. 802.11a tiene 12 canales no solapados, 8 para red inalámbrica y 4 para conexiones punto a punto. No puede interoperar con equipos del estándar 802.11b, excepto si se dispone de equipos que implementen ambos estándares.

Dado que la banda de 2.4 GHz tiene gran uso, este estándar utiliza la banda de 5 GHz ya que se presentan menos interferencias. Sin embargo, la utilización de esta banda también tiene sus desventajas, dado que restringe el uso de los equipos 802.11a a únicamente puntos en línea de vista, con lo que se hace necesario la instalación de un mayor número de puntos de acceso. Esto significa también que los equipos que trabajan con este estándar no pueden penetrar tan lejos como los del estándar 802.11b dado que sus ondas son más fácilmente absorbidas.

1.1.4.3.1.3 802.11g

Este protocolo utiliza la banda de 2.4 GHz (al igual que el estándar 802.11b) pero opera a una velocidad teórica máxima de 54 Mbps, o cerca de 24.7 Mbps de velocidad real de transferencia, similar a la del estándar 802.11a. Es compatible con el estándar b y utiliza las mismas frecuencias.

1.1.4.4 Sistema de Posicionamiento Global (GPS)

El Global Positioning System (GPS) o Sistema de Posicionamiento Global es un sistema global de navegación por satélite (GNSS) el cual permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave, con una precisión hasta de centímetros usando GPS diferencial.

El GPS funciona mediante una red de 24 satélites de los cuales 21 son operativos y 3 son de respaldo, en órbita sobre el globo a 20.200 km de distancia con trayectorias sincronizadas para cubrir toda la superficie de la tierra.

Cuando se desea determinar la posición, el aparato que se utiliza para ello localiza automáticamente como mínimo cuatro satélites de la red, de los que recibe unas señales indicando la posición y el reloj de cada uno de ellos. En base a estas señales, el aparato sincroniza el reloj del GPS y calcula el retraso de las señales, es decir, la distancia al satélite. Por "triangulación" calcula la posición en que éste se encuentra. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los tres satélites.

El sistema global de comunicación por satélite lo componen:

- **Sistema de satélites.** Está formado por 24 unidades con trayectorias sincronizadas para cubrir toda la superficie del globo terráqueo. Más concretamente, repartidos en 6 planos orbitales de 4 satélites cada uno. La energía eléctrica que requieren para su funcionamiento la adquieren a partir de dos paneles compuestos de celdas solares adosadas a sus costados.

- **Estaciones terrestres**. Envían información de control a los satélites y realizar el mantenimiento de toda la constelación.
- **Terminales receptores**: Indica la posición en la que estamos, conocidas también como Unidades GPS. Estas terminales receptoras pueden ser la PDA, Teléfonos Celulares, etc.

1.2 FABRICANTES

La gran demanda de estos equipos por las facilidades y aplicaciones que prestan ha dado lugar a que grandes empresas desarrollen tanto el hardware como el software para su distribución en el mercado.

Entre las principales empresas fabricantes tenemos:

1.2.1 PALM Y PALMSOURCE:

El primer fabricante de hardware para PDAs basadas en el sistema operativo Palm Os, mientras la segunda se destinó al desarrollo del sistema operativo Palm OS usado por Palm, Inc. para incorporar en su hardware.

PalmSource trabajó en el desarrollo la versión 6 del sistema operativo PalmOS 6 también llamada Cobalt para dispositivos móviles, esta versión estaría en gran parte orientada hacia Smartphones aunque también sería incluida en los futuros PDAs (No obstante ninguna PDA ni smartphone lleva ni ha llevado con anterioridad PalmOS 6). También se trabaja sobre una versión de PalmOS que funcione sobre un núcleo Linux. PalmSource es también la actual propietaria del sistema operativo BeOS, desde su compra a Be Incorporated.

1.2.2 ACCESS CO. LTD:

Es una compañía japonesa desarrolladora de software para teléfonos móviles, PDAs y otros dispositivos portátiles. Fue fundada en 1984. Su producto más

conocido es el navegador web NetFront disponible para Symbian, Palm OS, PSP, Playstation 3, etc.

1.2.3 HIGH TECH COMPUTER (HTC):

Es un fabricante de Taiwán de dispositivos portátiles basados en Microsoft Windows CE.

La compañía ha crecido aceleradamente tras haber sido escogida por Microsoft como socio de desarrollo hardware para el sistema operativo Windows Mobile el cual está basado en Windows CE.

Los productos HTC se venden bajo diferentes marcas y nombres, algunas de las cuales incluyen a los mayores operadores de telecomunicaciones como Orange, T-Mobile, Singular, O_2, Movistar y Vodafone.

HTC es el Original Desing Manufacturer (ODM) que fabrica y diseña los dispositivos para las siguiente compañías tecnológicas que venden y distribuyen PDAs basadas en Windows Mobile: Dell, Fujitsu-Siemens,HP/Compaq, i-mate,Krome, Sharp Corporation, Dopod (compañía subsidiaria de HTC) y Qtek (companía subsidiaria de HTC).

1.2.4 HEWLETT-PACKARD (HP):

Es la mayor empresa de tecnologías de la información del mundo, fabrica y comercializa hardware y software además de brindar servicios de asistencia relacionados con la informática.

La primera computadora personal de HP fue la Touchscreen 150, una computadora personal con MS-DOS que obtuvo solamente una aceptación modesta. En 1985, presentó la VECTRA, que fue la primera de una línea completa de PC compatibles con IBM.

En 1989, HP adquirió Apollo Computer, que combinada con su propia línea hizo de HP el líder del mercado en estaciones de trabajo. Hewlett-Packard vende por encima de 10.000 productos diferentes en el campo de la electrónica y la computación y ha ganado una reputación en todo el mundo por su ingeniería robusta y fiable.

1.2.5 ACER:

Es uno de los 5 principales fabricantes de computadoras en el mundo. Es la mayor empresa de venta al por menor de computadoras en Taiwán.

La oferta de productos de Acer abarca los PC, PDA, servidores, memorias, soluciones de Internet para: empresas, educación, y usuarios domésticos.

Las Personal Digital Assistant de Acer que sobresalen en el mercado son las de la serie N como: Acer N30, Acer N50, Acer N35, etc.

1.2.6 TOSHIBA:

Es una compañía japonesa dedicada a la manufactura de aparatos eléctricos y electrónicos cuya sede está en Tokio. Ocupa el 7º puesto en la lista de grandes compañías mundiales de su campo.

El grupo se expandió con fuerza, tanto por el crecimiento interno como por adquisiciones, absorbiendo compañías de ingeniería e industria primaria, dando lugar a empresas subsidiarias como *Toshiba EMI*, *Toshiba Electrical Equipment* (Equipos eléctricos Toshiba), *Toshiba Chemical* (Química de Toshiba), *Toshiba Lighting and Technology* (Iluminación y Tecnología Toshiba) y la *Toshiba Carrier Corporation* (Corporación de Transportes Toshiba).

1.2.7 DELL COMPUTER CORPORATION:

Es la compañía más grande de distribución de computadoras personales en los Estados Unidos.

Dell se introdujo en el mercado no solo con productos referentes a las computadoras, sino también con equipos como el Dell Digital Jukebox (un reproductor de audio digital portátil), Keydrives del USB, televisores con pantalla LCD, PDA's Móvil compatibles con Windows, así como impresoras de la misma marca.

Dell utiliza varias marcas de fábrica como Axim para PDAs que utiliza el móvil de Windows de Microsoft.

Dell envía actualmente Microsoft Windows XP y Windows Vista como el sistema operativo de la opción para la mayoría de sus computadoras nuevas.

1.2.8 SONY ERICSSON MOBILE COMMUNICATIONS AB:

Es una empresa formada por la empresa sueca Ericsson y la empresa japonesa Sony, dedicada a la fabricación de teléfonos móviles y accesorios. La compañía se fundó en 2001 entre Ericsson, una de las empresas líderes en telecomunicaciones y Sony una de las empresas líderes en electrónica de consumo. Ambas empresas poseen partes iguales en Sony Ericsson.

Sony Ericsson anunció su primer producto en marzo de 2002 y a finales del 2006 ya cuenta con un portafolio lleno de productos capaces de cubrir todos los tipos de usuarios.

Los productos de la serie P Sony Ericsson están constituidos por los Smartphones con Symbian OS de rango Alto como por ejemplo: P990, P910, P900, P800, P1, etc.

1.2.9 RESEARCH IN MOTION LIMITED (RIM):

Es una compañía fabricante de dispositivos inalámbricos. Esta es mejor conocida como el desarrollador del dispositivo de comunicación Black Berry.

RIM desarrolla y vende software y hardware para dispositivos móviles, usando tecnología C++ y Java para el desarrollo de sus aplicaciones.

1.3 VERSIONES DE EQUIPOS PDA

1.3.1 PALMS

1.3.1.1 Palm Pilot

Pilot fue el nombre dado a la primera generación de PDAs fabricadas por Palm Computing en 1996.

Figura 1.1 Palm Pilot

Las Palm Pilot utilizaban inicialmente el procesador Motorola Dragonball, un derivado del Motorola 68000. Modelos más recientes usan una variante de la extensamente popular arquitectura ARM (familia de microprocesadores RISC[2]) la cual es conocida generalmente bajo la marca Intel XScale[3]. Ésta es una clase de

[2] Computadora con Conjunto de Instrucciones Reducido
[3] Núcleo de Microprocesador

microprocesadores RISC que se utiliza extensamente en dispositivos móviles y sistemas empotrados, y su diseño fue influenciado fuertemente por la tecnología MOS 6502[4].

Las Palm Pilot son cada día más avanzadas, incluyendo la capacidad de convertirse en discos duros removibles en ordenadores vía cable USB, y están comenzando a combinarse con los Smartphones. Treo 680 es la combinación de una Palm con un teléfono móvil, E-mail, SMS, y mensajería instantánea.

1.3.1.2 Palm III

La Palm III es el primer ordenador de mano de Palm Computing en soportar la transferencia de archivos por infrarojo y un sistema operativo actualizable alojado en Flash ROM. Luego de la aparición de PALM III fue seguida por la Palm IIIe y Palm IIIx cada uno con nuevo sistema operativo y mejores pantallas y por las Palm IIIc y Palm IIIxe. La Palm IIIc tiene una pantalla en color TFT y una CPU más rápida y la Palm IIIxe tiene más RAM. Las Palm IIIc y Palm IIIxe tienen 8 Mbytes de RAM y PALM OS 3.5.

Figura 1.2 Palm III

Entre las diferencias existentes entre la Palm III y la Palm Pilot destaca la desaparición del slot de memoria, un mejor ajuste del contraste y un puerto IrDA.

[4] Microprocesador de 8 bits diseñado por MOS Technology

La Palm III corre el nuevo PalmOS 3.0 que incorporaba un nuevo lanzador de aplicaciones, un tamaño de fuente adicional, corrección de errores y otras mejoras. Equipada con 2 Mbytes de memoria SDRAM[5] para almacenaje de los datos del usuario y software adicional, y 2 Megabytes de Flash ROM para almacenar el sistema operativo y las aplicaciones incluidas.
Las Palm III y IIIx fueron las únicas PDAs de la serie Palm III en tener la ROM del sistema operativo y la RAM montada en una tarjeta de memoria independiente de la placa madre.

1.3.1.3 Zire

La serie Zire es una gama de agendas electrónicas de la empresa Palm. Se concibieron como un nivel de entrada para quienes requerían de las prestaciones básicas de los primeros modelos o precisaban de un equipo económico, aunque al bajar el costo de las tecnologías acabaron incorporando la reproducción de MP3 y la cámara digital, pero siempre por debajo del nivel de los equipos más avanzados.

1.3.1.3.1 Palm Zire & Palm Zire 21

La Palm Zire y la Palm Zire 21 son económicas y diferentes del resto de la línea de Palm por tener pantallas monocromáticas sin iluminación de fondo, además de sólo dos botones de acceso rápido en vez de cuatro, y un botón tradicional de navegación de arriba/abajo en vez de un navegador de 5 vías.

Figura 1.3 Palm Zire 21

[5] Memoria RAM dinámica de acceso síncrono de tasa de datos simple (Single Data Rate Synchronous Dynamic Random Access Memory)

La Zire fue la primera de la serie. Tenía sólo 2 MB de RAM, una CPU Motorola Dragonball a 33 MHz, y sistema operativo Palm OS 4.1.x.

La Zire 21 venía con un paquete Personal Information Manager (PIM) mejorado conocido como la Palm PIM Plus, una CPU TI OMAP311 compatible ARM a 126 MHz, 8 MB de RAM, y sistema operativo Palm OS 5.2.1.

Tanto la Zire como la Zire 21 carecen de una ranura de tarjetas SD/SDIO/MMC, una característica evitada por los aficionados de las PDAs clásicas.

1.3.1.3.2 Palm Z22

La Z22 es una sucesora de la Zire 21, pero no se lanzó bajo el nombre de "Zire". Tiene una pantalla a color de 160x160 pixels, 32 MB de memoria RAM (20 MB disponibles para el usuario) contra 8 MB de la Zire 21, procesador Samsung de 200MHz, y un sistema operativo Palm OS Garnet 5.4.

1.3.1.3.3 Palm Zire 31

La Palm Zire 31 es una versión actualizada de la Palm Zire 21. Posee pantalla a color de 160x160 pixels, tiene 16 MB de memoria RAM, de los cuales 13.8 MB están disponibles para escritura, un procesador Intel PXA255 de 200MHz con tecnología intel XScale, expansión SD/MMC/SDIO, Sistema Operativo Palm OS 5.2.8, capacidad de audio/MP3 mediante Palm's Media Application y RealPlayer, jack estéreo de 3,5 mm, y un navegador de 5 vías.

Figura 1.4 Palm Zire 31

1.3.1.3.4 Palm Zire 71

La Palm Zire 71 fue el primer intento de PalmOne en construir un PDA con cámara digital. Está equipado con un procesador TI OMAP310 a 144 MHZ, 16 MB de RAM (13 MB disponibles), pantalla TFT[6] color de 16 bits de 320x320 pixels, Sistema Operativo Palm OS 5.2.1, y una cámara de calidad VGA (300 K pixels con soporte 640x480). Posee un pequeño Joystick para navegación rápida. La capacidad de reproducir Audio era posible con la inclusión de un slot Secure Digital, e incluía RealPlayer para Palm.

La Zire 71 tiene un sistema de comunicación inalámbrico basado en el puerto infrarrojo. Se le podía añadir Wi-fi mediante el slot SD, pero sólo una tarjeta SD Wi-Fi.

1.3.1.3.5 Palm Zire 72

La Palm Zire 72 es una versión actualizada de la Palm Zire 71. Incluye nuevas características como Bluetooth, grabación de voz y captura de video con sonido.

La cámara fue mejorada de 0.3 a 1.2 megapixeles. La Zire 72 tiene 32 MB de memoria RAM portátil (25 MB disponibles) en vez de 16 MB, un procesador Intel PXA270 a 312 MHz en lugar del Texas Instruments OMAP.

1.3.1.4 Treo 650

El teléfono Palm Treo 650 es un híbrido entre teléfono móvil y PDA. Entre las características a destacar está el teclado tipo QWERTY y su memoria no volátil, además se puede expandir su memoria interna con tarjetas de expansión.

[6] Thin Film Transistor: es un tipo especial de Transistor de efecto de campo que se construye depositando finas películas sobre contactos metálicos, donde cada capa activa cada pixel, obteniéndose mejores tiempos de respuesta, al no ser necesario el barrido.

Figura 1.5 Treo 650

Posee resolución de pantalla de 160*160 a 320*320 pixels. Como sistema de comunicación inalámbrica incluye Bluetooth.

Palm Inc. ha desarrollado nuevos modelos que son competencia directa del Treo 650. Los cuales no tienen sistema operativo Palm OS sino Windows Mobile como los Treo 700w, 700wx y Treo 750w.

1.3.2 POCKET PC

PocketPC se trata de un pequeño ordenador, diseñado para ocupar el mínimo espacio y ser fácilmente transportable que ejecuta el sistema operativo Windows CE de Microsoft entre otros, el cual le proporciona capacidades similares a los PCs de escritorio.

Microsoft sacó la línea al mercado en 1998, decidiendo denominarla Palm *PC*. Debido a una demanda de Palm, el nombre fue cambiado a PocketPC.

PocketPC es un estándar de Microsoft que impone varios requisitos al hardware y al software de dispositivos móviles para tener la etiqueta de PocketPC.

Cualquier dispositivo que sea clasificado como un PocketPC debe:

- Ejecutar el sistema operativo Microsoft Windows CE o Windows Mobile (versión PocketPC).
- Tener un conjunto de aplicaciones en ROM.
- Incluir una pantalla sensible al tacto.
- Incluir un dispositivo apuntador, llamado stylus o estilete.
- Incluir un conjunto de botones de hardware para activar aplicaciones.
- Estar basado en un procesador compatible con el StrongARM (los Pocket PCs más antiguos tienen un procesador MIPS o SH3).

Algunas de las aplicaciones que se incluyen con estos dispositivos son versiones reducidas de Microsoft Outlook, Internet Explorer, Word, Excel, Windows Media Player, etc.

1.3.2.1 iPAQ

Es la PDA para PocketPC de HP. Es el principal competidor del Palm Pilot, pero se diferencia de éste en una mayor capacidad multimedia y un entorno más parecido a Windows. Tiene numerosos accesorios como lectores de tarjetas, tarjetas de red inalámbricas, GPS, baterías extra, Bluetooth, identificación dactilar, etc.

Figura 1.6 iPAQ

1.3.2.1.1 Versiones

- **Compaq iPaq H3600 series**, es la serie original de iPAQ, tenían pantallas a color de 12 bits, 64 MB de RAM y 16 de ROM.

- **Compaq iPaq 3100 series**, son una serie más barata que la 3600, no tenían pantalla en color.
- **Compaq iPaq H3700 series**, son iguales que los modelos de la serie 3600 pero con más RAM y ejecutaban PocketPC 2002 nativamente.
- **Compaq iPaq H3800 series**, son los primeros en incluir pantallas de 16 bits, incorporaban lectores de Secure Digital (SD) y una mayor RAM.
- **Compaq iPaq 3900 series**, es la evolución de la serie 3800. Tenían un procesador Intel XScale y los últimos modelos añadieron Bluetooth e IrDA.
- **HP iPaq H1900 series**, la primera serie después de la compra de HP. Era más pequeña que sus antecesoras. La memoria principal es de 64 MB. Ejecutan PocketPC 2002 (serie 1910) o 2003. También permitían baterías extra, aunque muchos de los accesorios existentes no eran compatibles.
- **HP iPaq H5400 series**, añaden Wi-Fi, Bluetooth y escáner biométrico. Esta serie presentaba muchos errores que fueron corregidos con actualizaciones del firmware.
- **HP iPaq 5500 series**, es la versión corregida de la serie 5400 con el doble de RAM.
- **HP iPaq H5100 series**, una serie inferior a la 5500, sin soporte WiFi y RAM de 64MB.
- **HP iPaq H2200 series**, destinados al mercado de consumo. Memoria RAM de 64MB, PocketPC 2003 y muchos accesorios.
- **HP iPaq H4300 series**, son parecidos a la serie 2200. Integran WiFi, soporte para Bluetooth y un pequeño teclado.
- **HP iPaq H4100 series**, está orientada al mercado corporativo, incluye WiFi y SDIO.
- **HP iPaq hx4700 series**, tiene pantalla VGA, WiFi, Bluetooth 1.2, SD, CompactFlash, touchpad, un procesador de 624 MHz y ejecuta Windows Mobile 2003 Second Edition.
- **HP iPaq rz1700 series**, funciona sobre Windows Mobile 2003 Second Edition, sin wireless y con 32 MB de RAM.
- **HP iPaq h6300 series**, es un híbrido de PocketPC y teléfono móvil. Tiene GSM, 64 MB RAM, procesador TI de 195 MHz, un pequeño teclado.

1.3.3 SHARP ZAURUS

Es el nombre de una serie de PDA hechas por la corporación Sharp. La principal característica de este tipo de dispositivos es que su sistema operativo es basado en UNIX.

1.3.3.1 Modelos de Zaurus

- **Personal Information (PI) series**
    - PI-5000/FX/DA,
    - PI-4500
    - PI-6000/FX
    - PI-6500
    - PI-8000
    - PI-6600
- **K-PDA (ZR) series**
    - ZR-3000
    - ZR-3500
    - ZR-5000/FX
    - ZR-5700
    - ZR-5800
- **MI series**
    - MI-10DC/10
    - MI-506DC/506/504,
    - MI-P2-B
    - MI-P10-S
    - MI-L1
    - MI-E25DC

- **Otros modelos MI**

    - BI-L10
    - MT-200
    - MT-300
    - MT-300C
    - Browser Borrad

- **Series SL basadas en Linux**

    - SL-5000D
    - SL-5500
    - SL-A300
    - SL-C700
    - SL-C860
    - SL-6000
    - SL-C1000
    - SL-C3200

1.3.3 Blackberry

Una BlackBerry es un Smartphone inalámbrico que admite correo, telefonía móvil, SMS, navegación Web, acceso a datos corporativos y otros servicios de información inalámbricos. Transporta su información a través de las redes de datos inalámbricas de empresas de telefonía móvil.

BlackBerry es la solución móvil, integral y flexible que le permite permanecer conectado en movilidad a su correo electrónico, a su teléfono móvil, a su organizador y a Internet. Existen diferentes tipos de dispositivos BlackBerry como:

1.3.3.1 BlackBerry Curve 8300 smartphone

El smartphone incorpora funciones como: cámara, reproductor multimedia, memoria expandible, marcación por voz, BlackBerry Maps y navegación por trackball. Además incluye las características esenciales que se pueden esperar de un smartphone BlackBerry: correo electrónico, mensajes de texto, mensajería instantánea, explorador Web y funciones telefónicas avanzadas.

Figura 1.7 BlackBerry Curve 8300

1.3.3.2 BlackBerry 8800 smartphone

El smartphone BlackBerry 8800, ha sido diseñado para permitirle hacer su trabajo con la misma eficacia que siempre esté donde esté. Este dispositivo le ofrece funciones de teléfono, correo electrónico, organización, navegación Web y mensajería instantánea. Además va un paso más allá, ya que incorpora funciones de GPS para mejorar el acceso a aplicaciones y servicios basados en ubicaciones, incluida la aplicación BlackBerry Maps preinstalada. Además posee un reproductor multimedia para sus videoclips y pistas musicales, memoria expandible que le garantiza espacio de sobra para sus archivos multimedia y una batería de alta capacidad.

Figura 1.8 BlackBerry 8800

1.3.3.3 BlackBerry Pearl 8100 smartphone

El smartphone BlackBerry Pearl 8100 es uno de los teléfonos de su clase más pequeños del mundo y contiene toda la potencia de BlackBerry. Se completa con una cámara digital, funciones multimedia y memoria ampliable. Además ofrece a los usuarios todo lo que se puede esperar de un dispositivo BlackBerry, incluyendo aplicaciones de organizador, teléfono, correo electrónico, explorador Web, mensajería de texto, mensajera multimedia y mensajería instantánea.
Gracias a la tecnología cuatribanda en 850/900/1800/1900 MHz, GSM/GPRS y EDGE, el BlackBerry Pearl permite la itinerancia internacional entre Norteamérica, Europa y Asia-Pacífico.

El pequeño tamaño del BlackBerry Pearl le permitirá llevarlo a cualquier lugar. Constituye la combinación definitiva entre cerebro y belleza. Pequeño, inteligente y con estilo.

Figura 1.9 BlackBerry Peral 8100

1.3.3.4 BlackBerry 7290 handheld

Este dispositivo BlackBerry posee las siguientes características:

- Aplicaciones de correo electrónico, teléfono, SMS, navegador y organizador en un solo dispositivo de bolsillo integrado.
- Dispositivo de bolsillo de banda cuádruple que funciona en redes inalámbricas GSM/GPRS a 850/900/1800/1900 MHz y permite la itinerancia internacional entre Norteamérica, Europa y la región Asia/Pacífico.
- Soporte Bluetooth para las comunicaciones de voz con manos libres.
- Memoria amplia para el almacenamiento de datos y aplicaciones.
- Teclado QWERTY de 33 teclas estilo PC.
- Rueda de desplazamiento para navegar de fácil manejo e intuitiva interfaz de menús.
- Pantalla brillante retroiluminada de 240x160 con más de 65 000 colores.
- Plataforma de desarrollo Java basada en estándares abiertos.
- Visualización integrada de archivos adjuntos.
- Diseño pequeño para una sensación de agradable ligereza

Figura 1.10 BlackBerry 7290 handheld

1.5 SISTEMAS OPERATIVOS

Un sistema operativo es un programa destinado a permitir la comunicación del usuario con la PDA. Trabaja cuando se enciende la PDA, y gestiona el hardware de la máquina desde los niveles más básicos.

Es frecuente diferenciar a las PDAs en función de su sistema operativo y su relación con las empresas fabricantes. El porcentaje de penetración de las diferentes PDA en el mundo está distribuido así:

- **Palm** (antes Palm Pilot) utilizan el sistema operativo Palm OS (de PalmSource, Inc.) - 40.7% del mercado.
- **Pocket Pc**, utilizan el sistema operativo Windows Mobile (de Microsoft) - 40.2% del mercado.
- **BlackBerry** utilizan un sistema operativo propio para los BlackBerry - 14.8% del mercado.
- **Linux** - 1.9% del mercado.
- **Otros** - 2.4% del mercado.

1.5.1 Palm OS

Palm OS es un sistema operativo propietario para dispositivos móviles. Ha sido implementado en dispositivos móviles como PDA, Smartphones y dispositivos de GPS.

Entre las principales características de PALM OS son:

- Ambiente simple de aplicaciones con una interfaz gráfica común.
- Pantalla monocromática o a color con una resolución de 480x320 o 640x480 pixeles.
- Sistema Handwriting (reconocimiento de escritura).
- Tecnología para sincronización con PCs (HotSync).

- Capacidad de reproducción de sonidos.
- Modelo simple de seguridad: Dispositivo basado en password.
- Acceso a redes TCP/IP.
- Sistemas de conexión Serial/USB, infrarrojo, Bluetooth y WiFi.
- Soporte para expansión utilizando tarjetas de memoria.

Palm OS fue diseñado para ser abierto y modular para soportar aplicaciones desarrolladas para terceros. Esto no significa que los datos y código de los programas pueden ser accedidos y modificados por algunos usuarios u otras aplicaciones, dando dificultad entre las aplicaciones legítimas y maliciosas donde solamente podrán realizar lectura/escritura y llamadas al sistema.

1.5.1.1 Información general del sistema de archivos y estructura de la aplicación

- Palm OS no usa un sistema de archivos tradicional. Los datos son almacenados en sectores de memoria llamados “registros”, porque son agrupados dentro de una base de datos.
- Las aplicaciones Palm OS son generalmente ejecutadas en un solo hilo y programas manejados por eventos. Solo un programa corre a la vez. Cada aplicación tiene una función principal que es el equivalente del main en programación en el Lenguaje C. Futuras versiones de la Palm OS deberán permitir a las aplicaciones ser multihilo.

1.5.1.2 Versiones de Palm OS

1.5.1.2.1 Versión 1

La versión 1.0 es la versión original presente en la Pilot 1000 y 5000. Esta versión y todas las versiones previas a la versión 5.0 son basadas en el kernel AMX 6800. Este kernel técnicamente soporta multitareas.

Palm OS no diferencia entre almacenamiento RAM y almacenamiento de sistema de archivos. Las aplicaciones son directamente instaladas en la RAM y

ejecutadas. El sistema operativo realiza constantes ciclos de refresco de la RAM para mantener disponibilidad de memoria.

El sistema operativo soporta pantallas monocromáticas de 160x160 pixeles. Los dispositivos de entrada para esta versión soporta el sistema de reconocimiento handwriting o a través de software que realiza la función de teclado. El sistema soporta sincronización de datos con otra PC vía HotSync[7] sobre una interfaz serial.

Otra característica de esta versión es que posee aplicaciones como Memo Pad, directorio telefónico y calculadora.

1.5.1.2.2 Versión 2

La versión 2 de Palm OS fue introducida en marzo de 1997. Esta versión añade red TCP/IP, HotSync network, y soporte de pantalla monocromática.

Se añadió la aplicación de e-mail a esta nueva versión en relación a la anterior.

1.5.1.2.3 Versión 3

Esta versión fue introducida para las series Palm III. Esta versión añadió un sistema de comunicación infrarrojo (IrDa). Esta versión añadió apoyo para color, puertos de expansión múltiples, nuevos procesadores y otras prestaciones.

1.5.1.2.4 Versión 4

La versión 4.0 salió con la serie Palm m500. Esta versión añadió una interfaz estándar para el acceso del sistema de archivos externa como tarjetas SD y mejoró las bibliotecas de telefonía, seguridad y mejoras de la interfaz gráfica.

[7] Transferencia de la información entre una computadora de escritorio y un dispositivo Palm

En esta versión el código y datos de las aplicaciones necesitan ser cargadas en la memoria RAM del dispositivo. Palm OS 4.0 también añadió un sistema de comunicación Bluetooth.

1.5.1.2.5 Versión 5

La versión 5.0 fue la primera versión que soportó los dispositivos ARM. Anunciado como paso importante por apoyar a los procesadores ARM, las aplicaciones Palm se ejecutan en un entorno emulado denominado el Entorno de Compatibilidad de Aplicaciones Palm, disminuyendo velocidad pero permitiendo gran compatibilidad con programas antiguos.

Esta nueva versión puede aprovechar los procesadores de ARM con ARMlets, pequeñas unidades de código ARM. Las siguientes versiones de Palm OS 5 han tenido un API estándar para alta resolución y áreas de entrada dinámicas, junto con un cierto número de mejoras menores.

1.5.1.2.6 Versión 6

Esta versión permitió apoyar a las aplicaciones nativas ARM junto con apoyo multimedia mejorado. Actualmente NO existen equipos que usen el Palm OS 6.

No está muy claro el futuro de esta versión de Palm OS, derivado de la compra de PalmSource por la compañía japonesa ACCESS, LTDA.

1.5.2 WINDOWS CE

Windows CE es una variación del Sistema operativo Windows de Microsoft para PDA’s y sistemas embebidos. Windows CE combina la compatibilidad y los servicios de aplicaciones avanzadas de Windows con soporte para múltiples arquitecturas de CPU y opciones incluidas de comunicación y redes para crear una variedad de productos.

Windows CE está orientado a los dispositivos electrónicos del cliente, terminales Web, dispositivos de acceso a Internet, controladores industriales especializados, computadoras de bolsillo y dispositivos de comunicación incrustados. Esta plataforma modular permite a los desarrolladores crear software para que la nueva generación de dispositivos móviles de 32-bits se integre con Windows e Internet.

Windows CE no es un subconjunto de Windows XP, o de Windows NT, sino que fue desarrollado a base de nuevas arquitecturas y una nueva plataforma de desarrollo. Windows CE tiene sus propias APIs para desarrollo y necesita sus propios drivers para el hardware con el cual va a interactuar. Windows CE no es un sinónimo de Windows XP en forma pequeña, incrustada o modular.

Windows CE ha evolucionado en base a un componente embebido, es un sistema operativo en tiempo real. Muchas plataformas se han basado en el núcleo del sistema operativo de Windows CE, incluyendo Microsoft's Autop, Pocket PC 2000, Pocket PC 2002, Mobile 2003 SE, Mobile 5.0, Mobile 6.0, Smartphone 2002, Smartphone 2003 y muchos dispositivos industriales y sistemas embebidos.

La última versión del Windows CE actualmente es Windows Mobile 5.0, predecesor del Windows Pocket 2003 Second Edition y sirve tanto para pocket PC (PDA) como para SmartPhone. Este sistema ha desarrollado una evolución hasta la versión 6.0.

1.5.3 WINDOWS MOBILE

Windows Mobile es un sistema operativo compacto, con un conjunto básico de aplicaciones para dispositivos móviles basados en la API Win32 de Microsoft. Los dispositivos que llevan Windows Mobile son Pocket PC's, Smartphones y Portable Media Center. Ha sido diseñado para ser similar a las versiones de escritorio de Windows.

1.5.3.1 Características Comunes de Windows Mobile

Tanto Windows Mobile para Pocket PC, como Windows Mobile para Smartphones, poseen bastantes aspectos parecidos, pero en cuanto a aplicaciones desarrolladas, una aplicación de Windows Mobile PPC (Pocket PC) no servirá en Windows Mobile Sp (Smartphone).

- En la pantalla nos mostrará la fecha actual, la información del propietario, las citas próximas, los E-mail, y las tareas. En la parte inferior aparecerá, generalmente, una barra con dos botones. También incluye una barra que posee iconos para notificar el estado del Bluetooth, batería, cobertura, etc. Este tema predeterminado puede ser cambiado añadiendo o eliminando complementos, como por ejemplo, alarma, temperatura, estado de la batería.

- En la barra de tareas muestra: la hora actual, el volumen y el estado de la conectividad. La característica principal de la barra de tareas es el botón de *Inicio*, que está diseñado para que sea parecido al botón de Inicio de las versiones de escritorio de Windows. El menú de Inicio ofrece programas abiertos recientemente, nueve entradas del menú personalizadas, y accesos directos a programas, ajustes, búsquedas, y ayuda.

- Las versiones Pocket PC incluyen en Windows Mobile aplicaciones de Microsoft Office. Por ejemplo se incluyen Pocket Word y Pocket Excel. En Windows Mobile 5.0 se incluye Pocket PowerPoint. Estas versiones incluyen muchas de las características que se utilizan en versiones de escritorio, pero algunas otras características como la inserción de las tablas e imágenes no se han incluido en versiones anteriores a Windows Mobile 5.0.

- Outlook Mobile es también un programa que viene con Windows Mobile. Esto incluye tareas, calendario, contactos, y la bandeja de entrada.

Microsoft Outlook para las versiones de escritorio se incluye a veces en los CD-ROM's del fabricante del Pocket PC.

- Windows Media Player for Windows Mobile se añade con el software. Actualmente, todas las Pocket PC incluyen la versión 9 del reproductor. Windows Media Player reproduce: WMA, WMV, MP3, y AVI. Los archivos MPEG actualmente no están soportados, y se debe descargar un programa de terceros para reproducirlos, y los archivos de WAP se reproducen en un reproductor por separado. Algunas versiones son también capaces de reproducir M4A.

- Cliente para Point-to-Point Tunneling Protocol (PPTP) y Virtual Private Network (VPN)

1.5.3.2 Versiones

1.5.3.2.1 PocketPC 2002

PocketPC 2002, utiliza Windows CE 3.0. Diseñado para dispositivos Pocket PC con pantalla 240 × 320 (QVGA) (sin teclado), PocketPC 2002 fue el lanzamiento original de PocketPC 2000, una entidad independiente en la gama de dispositivos Microsoft Embedded. Con los lanzamientos futuros, las líneas de Pocket PC y Smartphone chocaban cada vez más, mientras que los términos de licencia se relajaron permitiendo que los Original Equipment Manufacturer (OEM) se aprovecharan de las ideas más innovadoras de diseño.

1.5.3.2.2 Windows Mobile 2003

La tercera versión nombrada Windows Mobile 2003. Fue lanzada el 23 de junio del 2003, y era el primer lanzamiento bajo el nombre Windows Mobile. Vino en tres ediciones diferentes. Dos de estas ediciones son muy similares: “Windows Mobile 2003 for Pocket PC Premium Edition” y “Windows Mobile 2003 for Pocket PC Phone Edition”, este último diseñado para los Pocket PC que tienen características de teléfonos móviles.

La tercera edición es Windows Mobile 2003 Smartphone Edition que a pesar de sus semejanzas con la de Pocket PC - es una plataforma substancialmente diferente ya que está limitada por las características especiales de este tipo de dispositivos. Algunas de estas limitaciones son: funcionamiento por teclas al no disponer de pantalla táctil, resolución de pantalla más baja, modelo de seguridad que impide instalar aplicaciones no firmadas y modelo de memoria diferente (diferente tipo de memoria y menor cantidad).

Windows Mobile 2003 es conocido también como Windows CE 4.20.

1.5.3.2.3 Windows Mobile 2003 Second Edition

Windows Mobile 2003 Second Edition, también conocida como Windows Mobile 2003SE, salió el 24 de marzo del 2004 y la Dell Axim x30 fue la primera en tenerlo. Esta versión del Sistema Operativo Incluye un número de mejoras sobre su precursor, como:

- La opción de cambiar la orientación de la pantalla. Esto no está disponible en la versión de Smartphone.
- Pocket Internet Explorer (también conocido como PIE) incluye la opción de forzar a una página en una disposición de una columna, haciendo la lectura más fácil puesto que solo se tiene que utilizar el scroll vertical.
- Soporte para una resolución de pantalla VGA (640×480). También se apoya un nuevo factor de forma del cuadrado (240×240 y 480×480 para las pantallas de VGA), que favorece a los fabricantes que desean incluir un teclado hardware. Aunque no era su idea original, Microsoft decidió agregarla debido a la presión de fabricantes del Pocket PC.
- Soporte para Wi-Fi.

Windows 2003SE Mobile utiliza Windows CE 4.21.111

1.5.3.2.4 Windows Mobile 5.0

Windows Mobile 5.0, anteriormente con el nombre en clave "Magneto", salió al mercado el 9 de mayo del 2005. Fue ofertado en la Dell Axim x51. Utiliza Windows CE 5.0 y utiliza .NET Compact Framework 1.0 SP2 (una plataforma de desarrollo .NET para los programas basados en .NET que utiliza).

Entre las principales características que presenta tenemos:

1. Una nueva versión de Office llamada "Office Mobile".
    - Se agregará una versión de Powerpoint denominada "Powerpoint Mobile".
    - Excel Mobile añade la capacidad de ver representaciones gráficas.
    - Word Mobile incluirá la capacidad de insertar tablas y gráficos.
2. Reproductor "Windows Media 10 Mobile".
3. Identificador de llamadas con fotos.
4. Un paquete multimedia que facilitará la administración de vídeos y fotos.
5. Ayuda mejorada de Bluetooth.
6. Interfaz de administración GPS para los programas de navegación instalados.
7. Mejoras de la funcionalidad de "Microsoft Exchange Server" las mejoras funcionan solamente con Exchange 2003 SP2 instalado.
8. Soporte para teclados QWERTY incluido por defecto.
9. Simplificación del sistema de informe de errores, como las versiones de Windows de los desktop y servidores.
10. ActiveSync 4.2, prometiendo 10-15% de aumento de la velocidad en la sincronización de datos.
11. Cliente para PPTP y L2TP/IPsec VPNs..
12. La memoria no volátil (ROM) está disponible en Pocket PC permitiendo un aumento de la batería. Ya que anteriormente más del 50% (suficiente para 72 horas de almacenaje) de energía de la batería se reservaba para mantener datos en la memoria RAM (volátil). Los dispositivos basados en

Windows usa la memoria RAM como su medio de almacenaje primario al uso de memoria flash.

Desventajas:

- Microsoft Money para Pocket PC no es soportado por Windows Mobile 5.0.
- Impresoras no son incluidas en el sistema operativo.
- La pantalla no se bloquea durante una llamada telefónica.
- GPRS posee un acceso automático y no puede desactivarse cuando una aplicación requiere el Internet.
- No es fácil revisar archivos de una LAN.
- ActiveSync no trabaja a través de WLAN (Solo con USB y Bluetooth).

1.5.3.2.5 Windows Mobile 6

Windows Mobile 6, antes conocido como Crossbow es la última versión de la plataforma Windows Mobile y fue lanzado el 12 de febrero del 2007 en el 3GSM World Congress 2007. Ofrece tres versiones: Windows Mobile 6 Standard para Smartphones (teléfonos sin pantalla táctil), Windows Mobile 6 Professional para PDAs con la funcionalidad del teléfono (Pocket PC Phone Edition), y Windows Mobile 6 Classic para PDAs sin radios celulares.

Windows Mobile 6.0 utiliza Windows CE 5.2 y está ligado fuertemente a los productos: Windows Vista, Windows Live, Microsoft Office y Exchange 2007.

El estándar de Windows Mobile 6 primero fue ofrecido en el Orange SPV E650 (HTC Vox).

Entre las características más importantes de esta versión tenemos:

- Windows Mobile 6.0 esta basado en Windows CE 5.0 (versión 5.2).
- Soporta resoluciones de 800x480 y 320x320 pixels.
- Windows Live para Windows Mobile.

- Acceso de escritorio remoto mejorado.
- Desarrollo y distribución de aplicaciones más rápido y más fácil.
- Soporte VoIP con los codec del audio AEC (Acoustic Echo Cancelling) y MSRT.
- Windows Live para Windows Mobile.
- Microsoft Bluetooth Stack mejoró notablemente.
- Cifrado de la tarjeta de almacenamiento - Windows Mobile 6 para Pocket PC y Smartphone soportan el cifrado de los datos almacenados en tarjetas externas de almacenamiento.
- Smartfilter para buscar más rápidamente emails, contactos, canciones, archivos, etc.
- Mejora de Internet Sharing para una fácil configuración de tu dispositivo como módem de computadora portátil.
- Outlook Mobile ahora soporta HTML .
- Capacidad de buscar para contactos en Exchange Server Address Book.
- Soporte AJAX, JavaScript y XMLDOM en Internet Explorer Mobile.
- Unlisenced Mobile Access (UMA).
- Server Search para buscar en toda la bandeja de entrada de Exchange desde el dispositivo. (Requiere Exchange 2007)
- .NET Compact Framework v2 SP1 en la ROM.
- SQL Server Compact Edition en la ROM.

1.5.3.2.6 Próximas versiones

Microsoft está trabajando en una actualización de la plataforma Windows Mobile con el nombre en clave de Photon y llevará Windows Embedded CE 6.0.

Photon (2008)

La actualización de Windows Mobile 6 con el nombre en clave de "Photon" y está programado para ser lanzado en la primera mitad del 2008.

Esta nueva versión de Windows Mobile poseerá las siguientes características:

- Basado en Windows CE 6.0.
- Shell nuevo y flexible.
- Contenedores modulares.
- Kernel CE “Yamazaki”.

1.5.4 SYMBIAN OS

Symbian OS es un sistema operativo que fue el producto de la alianza de varias empresas de telefonía celular, entre las que se encuentran Nokia, Sony Ericsson, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic y Sharp.

El objetivo de Symbian fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o el Smartphone de Microsoft.
Symbian contempla cuatro tipos de dispositivos para su sistema operativo, los denominados Serie60, Serie80, Serie90 y UIQ. La mayoría de Nokia son Serie60, todos los de Sony Ericsson trabajan bajo UIQ, así como también Motorola.

1.5.5 LINUX

Linux es la denominación de un sistema operativo tipo-Unix y el nombre de un núcleo. Es uno de los paradigmas más prominentes del software libre y del desarrollo del código abierto, cuyo código fuente está disponible públicamente, para que cualquier persona puede libremente usarlo, estudiarlo, redistribuirlo y modificarlo.

Los primeros sistemas Linux se originaron en 1992, al combinar utilidades de sistema y librerías del proyecto GNU con el núcleo Linux, completando un sistema también conocido como GNU/Linux. Desde fines de 1990 Linux ha obtenido el apoyo de diversas empresas multinacionales del mundo de la informática, tales como IBM, Sun Microsystems, Hewlett-Packard y Novell.

Linux es usado como sistema operativo en una amplia variedad de plataformas de hardware y computadores, incluyendo los computadores de escritorio (PCs x86 y x86-64, y Macintosh y PowerPC), servidores, supercomputadores, mainframes, PDAs y dispositivos empotrados así como teléfonos celulares.

Linux se refiere estrictamente al núcleo Linux, pero es comúnmente utilizado para describir al sistema operativo tipo Unix (que implementa el estándar POSIX), que utiliza primordialmente filosofía y metodologías libres y que está formado mediante la combinación del núcleo Linux con las bibliotecas y herramientas del proyecto GNU y de muchos otros proyectos/grupos de software (libre o no libre).

La expresión "Linux" es utilizada para referirse a las distribuciones GNU/Linux, colecciones de software que suelen contener grandes cantidades de paquetes además del núcleo. El software que suelen incluir consta de una enorme variedad de aplicaciones, como: entornos gráficos, suites ofimáticas, servidores web, servidores de correo, servidores FTP, etc.

Linux necesita una interfaz gráfica para que facilite las actividades de los usuarios, entre estas tenemos:

1.5.5.1 GPE Palmtop Environment

GPE[8] proporciona un entorno al usuario con una interfaz para PDAs que ejecuten GNU/Linux o cualquier otro sistema operativo basado en UNIX.

GPE no es un simple software, sino un completo entorno de componentes que hacen posible usar PDAs GNU/Linux para tareas comunes como administrador de información personal (Personal Information Management (PIM)), acceso a la web, multimedia y otras herramientas. Es útil en toda clase de dispositivos móviles con recursos limitados.

[8] Ambiente del Interfaz Grafico de Usuario para PDA

GPE proporciona una infraestructura de fácil y potente desarrollo de aplicaciones y la buena compatibilidad entre las soluciones de escritorios existentes. GPE utiliza sistema X Window y GTK+. Proporciona software del núcleo como bibliotecas compartidas, además el ambiente de GPE define los estándares para el diseño y la interacción de programas. Se basa en el lenguaje de programación C y otros estándares comunes tales como SQL, XML, D-BUS, etc.

GPE está comprometido con la idea de código abierto. Todos los componentes del núcleo de GPE están bajo la licencia de GNU, las aplicaciones usan GPL (General Public License) y las librerías compartidas usan LGPL(Lesser General Public License). Estó permiten la utilización libre del sistema GPE.

1.5.5.2 Opie

Opie (The Open Palmtop Integrated Environment) Es una interfaz de usuario gráfica, completamente en código abierto y una serie de aplicaciones, para PDAs y otros dispositivos móviles que se ejecuta bajo Linux. Está incluido en varias distribuciones de Linux empotradas como OpenZaurus, Familiar Linux y OpenSIMpad.

Opie es un contenedor del entorno de desarrollo Qtopia desarrollada por la empresa Trolltech. Soporta estándares ampliamente utilizados como XML, Obex e IrDA.

La versión 1.0 tiene soporte para las Sharp Zaurus, Compaq IPAQ y Siemens SIMpad en varios lenguajes, con soporte de Temas y Estilos, lo que permite a los usuarios personalizarlo.

1.6 APLICACIONES

Las PDA son computadores de carácter general, las cuales poseen aplicaciones básicas y específicas, las cuales están adaptadas a las necesidades de los usuarios.

1.6.1 APLICACIONES BÁSICAS

1.6.1.1 Libreta de Direcciones

El programa Libreta de Direcciones de una PDA almacena información personal, en cualquiera de las categorías definidas por el usuario. Se muestran las entradas y salidas se ordenan por los apellidos, y nombre. Hay cinco campos para el teléfono o correo electrónico, cada uno de los que se puede designar a Trabajo, a Casa, Fax u Otras, Correo Electrónico, Principal, Buscapersonas o Móvil (los nombres de los campos no se pueden cambiar).

1.6.1.2 Calculadora

La Calculadora convierte el ordenador de bolsillo en una calculadora estándar de 4 funciones con botones de tres tonos morados y azules que contrastan con los dos botones rojos para borrar. Incluye teclas de raíz cuadrada y de porcentaje y tiene memoria. También tiene una opción por mostrar un historial de los cálculos realizados, como muchas calculadoras que se usaban antes.

1.6.1.3 Calendario

El Calendario muestra un horario diario o semanal, o una vista mensual simple. El horario diario tiene una línea cada hora, entre dos horas del día que el usuario puede elegir. Al hacer clic en una línea vacía se crea una cita nueva. Las líneas vacías se llenan con las citas correspondientes, y la hora en que empiezan y su duración se muestran en el margen de la izquierda. El sistema operativo puede anunciar una cita con una alarma, en el momento indicado, minutos, horas o días antes. Estas alarmas suenan incluso cuando la unidad está apagada. Las citas se pueden repetir en un número especificado de días, semanas, meses o años y pueden contener notas.

1.6.1.4 Gastos

La aplicación Gastos permite a un usuario seguir los gastos comerciales habituales. El ordenador de bolsillo no realiza ningún cálculo para obtener el total. El usuario tiene que sincronizar con un ordenador de escritorio y ver los datos de los gastos en una hoja de cálculo en las que se incluyen las plantillas para Microsoft Excel (Palm OS).

1.6.1.5 Libreta de Notas

Las Notas de Texto permiten escribir notas de hasta 4,000 caracteres, clasificadas en categorías configurables por el usuario. Las notas se pueden ordenar alfabéticamente o manualmente (que permiten al usuario escoger el orden de las notas). Las Notas de Texto sólo pueden incluir texto, no dibujos. Por ésto, el texto en las Notas se ha de introducir utilizando el alfabeto Graffiti.

1.6.1.6 Notas

En las notas se pueden hacer dibujos y notas manuscritas. Se pueden ingresar hasta 10 palabras por página, si la escritura es pulcra. De lo contrario, es mejor poner texto en la Libreta de Notas. Hay tres medidas de lápices de dibujo, más una goma de borrar. Es posible dibujar un mapa muy simple.

1.6.1.7 Tareas

También denominada lista de tareas. Es el lugar adecuado por crear recordatorios personales y priorizar las cosas que tienes que hacer. Cada elemento de la lista de tareas también puede tener: una prioridad, categorías (por organizar y agrupar las tareas en grupos lógicos), adjuntar una Nota (por añadir una descripción o una aclaración de la tarea). Las tareas se pueden ordenar por: fecha, prioridad o categoría.

1.6.2 APLICACIONES ESPECÍFICAS

Las aplicaciones específicas desarrolladas para PDA se dividen en dos grupos de acuerdo al ambiente de desarrollo:

La primera que es desarrollada y compilada en lenguajes que requieren librerías para la ejecución como por ejemplo las de Visual Basic hecha en Windows o ambientes Java.

El otro tipo de aplicaciones es basada en browser. Por ejemplo se puede desarrollar una aplicación en Delphi[9]usando Intraweb para lo cual se debería usar HTML 3.2 para el servicio de páginas en los browser de la PDA. A veces, la gran desventaja en este tipo de aplicaciones es que éstas no pueden trabajar sino tiene una conexión a un servidor.

1.6.2.1 Herramientas para el desarrollo de Aplicaciones para PDA

Algunas de las herramientas para el desarrollo de aplicaciones para PDA se presentan a continuación:

1.6.2.1.1 Java 2 Micro Edition (J2ME)

J2ME es una versión de JAVA para dispositivos limitados en memoria y procesamiento, como PDA´s o dispositivos celulares. Cuenta con un conjunto de librerías, y máquinas virtuales para los dispositivos que lo soportan. Utilizando esta tecnología se pueden crear aplicaciones de captura de datos, sincronización, interacciones cliente / servidor etc. Es soportado por plataformas Palm OS aunque no completamente para plataformas Windows CE.

1.6.2.1.2 eMbedded Visual Tools

eMbedded Visual Tools es una herramienta de Microsoft para el desarrollo de aplicaciones para dispositivos PDA tipo Windows CE. Provee entornos similares a

[9] Delphi: es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual.

Visual Basic y Visual C++, pero para estas versiones se llaman eMbedded Visual Basic y eMbedded Visual C++.

1.6.2.1.3 WABA

WABA es una herramienta que define un lenguaje, una máquina virtual y manejo de clases idéntico a JAVA, la diferencia radica que WABA no es creado por Sun MicroSystems.

WABA se puede ejecutar en dispositivos Palm OS y Windows CE. tiene soporte para comunicaciones TCP, manejo de archivos y gráficos.

1.6.2.1.4 Personal Java

Es un API de JAVA creado para ejecutarse en dispositivos PDA pero con mayores capacidades, tales como PDAs tipo Windows CE Compaq iPaq, HP Jornada, etc. Personal JAVA soporta APIs de java como AWT, JDBC, comunicaciones y otras librerías comunes.

1.6.2.2 Usos de las aplicaciones para PDA

Los diferentes usos de las aplicaciones para PDA se basan siempre en las necesidades que los usuarios requieren, ya sea en los diferentes campos laborales, de diversión, o salud.

Las diferentes aplicaciones que tenemos son:

1.6.2.2.1 Orientado al campo laboral

- Sistemas de búsqueda de elementos farmacéuticos.
- Sistema de control automático del hogar (domótica) a través de una PDA.
- Configuración de equipos a través de una PDA.
- Sistema de registro de personal de planta en fábricas.

- Informativos de negocio mundial y bolsas de valores.

1.6.2.2.2 Orientado a la Salud

- Utilizado como una herramienta para diagnóstico básico para personas con válvulas cardiacas.
- Diccionario médico.
- Avisos de emergencia a centros de salud.
- Orientado a las terapias respiratorias.

1.6.2.2.3 Receptores GPS

Al tener una PDA un receptor GPS puede tener las siguientes aplicaciones:

- Navegación terrestre, marítima y aérea.
- Topografía y geodesia. Localización agrícola (*agricultura de precisión*).
- Salvamento.
- Deporte, acampada y ocio.
- Aplicaciones científicas en trabajos de campo.
- Se lo utiliza para el rastreo y recuperación de vehículos.
- Navegación deportiva.
- Deportes Aéreos: Parapente, Planeadores, etc.

CAPÍTULO 2

ANÁLISIS, DISEÑO Y DESARROLLO DEL SOFTWARE DE MONITORIZACIÓN DE RED SOBRE PDA

Antes de definir las especificaciones de requerimientos de software es necesario realizar un análisis preliminar del programa a desarrollarse.

El programa a desarrollarse permitirá monitorizar y administrar una red LAN desde un dispositivo inalámbrico PDA, haciendo uso del protocolo SNMP para consultar y modificar los valores de los objetos MIBs de cada dispositivo gestionado.

La aplicación debe ser capaz de recibir notificaciones desde el Agente del dispositivo gestionado, al detectar una condición predeterminada, como una alarma o desconexión.

Además, la aplicación tendrá funciones utilitarias que permitirán al administrador de red comprobar la conectividad de los equipos mediante el protocolo ICMP y configurar los dispositivos que son parte de la red a través del browser.

La aplicación se ejecutará desde una PDA, ya que este dispositivo al ser de menor tamaño tendrá mayor portabilidad, permitiéndole al administrador de red optimizar tiempo y recursos, informándole de cada uno de los eventos que sucedan en la red, sin necesitad de tener a mano un computador.

2.1 REQUERIMIENTOS DE SOFTWARE

Entre los requerimientos necesarios para el desarrollo del software se ha tomado en cuenta ciertos aspectos como:

- Requerimientos Funcionales.
- Requerimientos No Funcionales entre los que incluye requerimientos como de usabilidad, confiabilidad, hardware, entorno, seguridad y escalabilidad.

Cada uno de estos requerimientos se detalla a continuación:

2.1.1 REQUERIMIENTOS FUNCIONALES

Los Requerimientos Funcionales definen el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica.

Para el desarrollo del software de Monitorización de Red en dispositivos inalámbricos (PDA) es necesario tomar en cuenta los siguientes requerimientos funcionales:

2.1.1.1 Requerimientos de la administración de la red:

El sistema deberá permitir:

1. Agregar nuevos dispositivos a la red, identificando su dirección IP, nombre, tipo, grupo, comunidad de lectura y escritura.
2. Eliminar los equipos desconectados.
3. Modificar los atributos de los dispositivos.
4. Monitorizar las operaciones de la red utilizando las herramientas de gestión.
5. Configurar y probar los equipos.

2.1.1.2 Considerando los servicios SNMP

El sistema deberá permitir:

1. Obtener información de administración contenida en las MIBs de los dispositivos gestionados mediante la función GET.

2. Obtener los valores contenidos en la MIB de los dispositivos gestionados referente al objeto siguiente del valor especificado anteriormente mediante la función GetNext.
3. Modificar el valor contenido en el MIB referente a un determinado objeto de los dispositivos gestionados mediante la función SET.
4. Enviar mensajes espontáneos desde el Agente del dispositivo gestionado hacia el Administrador, al detectar una condición predeterminada, como una alarma o la conexión - desconexión de una estación.

2.1.1.3 Utilitarios del Programa:

El sistema deberá permitir:

1. Determinar si existe conexión física con los dispositivos gestionados.
2. Obtener los tiempos de respuesta hacia los dispositivos de red.
3. Configurar remotamente los dispositivos de red como switch, routers, server, etc. que soporten esta operación.
4. Verificar la disponibilidad de los dispositivos de red.
5. Establecer el estado de conexión de los equipos de red.

2.1.2 REQUERIMIENTOS NO FUNCIONALES

En los requerimientos No Funcionales analízaremos la usabilidad, confiabilidad, hardware, entorno, seguridad y escalabilidad del programa.

2.1.2.1 Requerimientos de Usabilidad

Este requerimiento toma en cuenta la facilidad de manejo de software, el número de pasos para realizar las acciones (Nuevo, Editar y Borrar), operaciones (Ping, Browser y SNMP) y notificaciones (Trap y polling) identificando la mecánica de cada paso.

Para el análisis de este requerimiento se utilizarán los diagramas de Caso de Uso para los siguientes escenarios.

2.1.2.1.1 Configuración de los Dispositivos

Se describen las acciones a realizar para poder ingresar, editar y borrar un dispositivo de la aplicación.

- **Agregar Nuevo Dispositivo**

En la Tabla 2.1 se detalla el caso de uso para agregar un nuevo dispositivo a la aplicación.

<table>
<tr><td colspan="2">CASO DE USO: Nuevo Dispositivo</td></tr>
<tr><td colspan="2">Actor: Administrador de Red</td></tr>
<tr><td colspan="2">Descripción: Permite agregar un nuevo dispositivo a ser gestionado por la NMS que se ejecuta en la PDA.</td></tr>
<tr><td colspan="2">Activación: Ingresando al menú Archivo - NUEVO.</td></tr>
<tr><td>Curso Normal</td><td>Alternativas</td></tr>
<tr><td>1. Se presenta una pantalla donde se ingresan los atributos del dispositivo.</td><td></td></tr>
<tr><td>2. Ingresados los datos presionamos el botón aceptar.</td><td>2. 1. Si el dispositivo a ser ingresado no se encuentra dentro de la red, o su IP es incorrecta, se presenta una pantalla de error.</td></tr>
<tr><td colspan="2">Precondiciones: Debe existir conexión física entre la PDA y los dispositivos de red.</td></tr>
<tr><td colspan="2">Postcondiciones: El Administrador tiene en pantalla el menú principal y puede realizar la gestión y monitorización de los equipos.</td></tr>
<tr><td colspan="2">Puntos de Extensión:</td></tr>
<tr><td colspan="2">Observaciones y Datos: La dirección IP es el atributo único para cada dispositivo, el cual será utilizado para agregar el dispositivo en la NMS.</td></tr>
</table>

Tabla 2.1 Caso de uso para agregar un nuevo dispositivo.

- **Editar Dispositivo**

En la Tabla 2.2 se detalla el caso de uso para editar un dispositivo de la aplicación.

<table>
<tr><td colspan="2">CASO DE USO: Editar Dispositivo</td></tr>
<tr><td colspan="2">Actor: Administrador de Red</td></tr>
<tr><td colspan="2">Descripción: Permite Editar los atributos principales de cada dispositivo ingresando en la aplicación.</td></tr>
<tr><td colspan="2">Activación: Ingresando al menú Edición - EDITAR.</td></tr>
<tr><td>Curso Normal</td><td>Alternativas</td></tr>
<tr><td>1. Se presenta una pantalla donde se presentan los atributos del dispositivo.</td><td></td></tr>
<tr><td>2. Se puede modificar los atributos del dispositivo.</td><td>2. 1. No se puede Alterar la dirección IP del dispositivo</td></tr>
<tr><td colspan="2">Precondiciones: Deben existir dispositivos agregados a la aplicación</td></tr>
<tr><td colspan="2">Postcondiciones: El Administrador tiene en pantalla el menú principal y puede realizar la gestión y monitorización de los equipos.</td></tr>
<tr><td colspan="2">Puntos de Extensión:</td></tr>
<tr><td colspan="2">Observaciones y Datos: Para poder editar un dispositivo este debe estar LinkUP o su estado Conectado.</td></tr>
</table>

Tabla 2.2 Caso de uso para Editar Dispositivo.

- **Borrar Dispositivo**

En la Tabla 2.3 se detalla el caso de uso para borrar un dispositivo de la aplicación.

CASO DE USO: Borrar Dispositivo
Actor: Administrador de Red
Descripción: Permite borrar un dispositivo ingresando en la aplicación.

Activación: Ingresando al menú Edición - BORRAR.	
Curso Normal	**Alternativas**
1. Se eliminara el dispositivo seleccionado del menú principal.	**1.1.** Si no existen dispositivos nos presenta una pantalla de error.
Precondiciones: Deben existir dispositivos agregados a la aplicación.	
Postcondiciones: El Administrador tiene en pantalla el menú principal.	
Puntos de Extensión:	
Observaciones y Datos:	

Tabla 2.3 Caso de uso para Borrar Dispositivo.

2.1.2.1.2 Monitorización de Dispositivos

Para monitorear un dispositivo, este primero debe estar agregado en la aplicación, una vez hecha esta operación podemos monitorear de acuerdo a las herramientas SNMP, Ping y Browser.

- **Herramienta SNMP**

En la tabla 2.4 se muestra el caso de uso para la herramienta SNMP

CASO DE USO: Herramienta SNMP	
Actor: Administrador de Red	
Descripción: Permite realizar operaciones SNMP (Get, GetNext y Set) a los dispositivos que soporten este protocolo.	
Activación: Ingresando al Menú Herramientas - SNMP.	
Curso Normal	**Alternativas**
1. Se presenta una pantalla donde podemos realizar las operaciones Get, GetNext y Set.	**1.1.** Si no existen dispositivos nos presenta una pantalla de error
Precondiciones: Deben estar agregados dispositivos que soporten el protocolo SNMP en la aplicación.	

Postcondiciones: El Administrador tiene en pantalla los resultados de esta operación.
Puntos de Extensión:
Observaciones y Datos:

Tabla 2.4. Caso de uso Herramienta SNMP.

CASO DE USO: Operación GET	
Actor: Administrador de Red	
Descripción: Permite realizar consultas SNMP de las MIB de los dispositivos gestionados.	
Activación: Presionando el botón GET	
Curso Normal	**Alternativas**
1. Ingresar el objeto identificador a consultar.	**1.1.** Si el Objeto Identificador es incorrecto se presentará un error.
2. Se presentará el valor de la MIB consultada.	
Precondiciones: El dispositivo debe soportar el protocolo SNMP.	
Postcondiciones: El Administrador tiene en pantalla los resultados de esta operación.	
Puntos de Extensión:	
Observaciones y Datos:	

CASO DE USO: Operación GET NEXT	
Actor: Administrador de Red	
Descripción: Permite consultar el siguiente valor de un objeto de la tabla de datos dentro de la MIB.	
Activación: Presionando el botón GET NEXT	
Curso Normal	**Alternativas**
1. Ingresar el objeto identificador a consultar.	**1.1.** Si el Objeto Identificador es incorrecto se presentará un error.
2. Se presentará el valor de la MIB	

consultada.	
Precondiciones: El dispositivo debe soportar el protocolo SNMP.	
Postcondiciones: El Administrador tiene en pantalla los resultados de esta operación.	
Puntos de Extensión:	
Observaciones y Datos:	

CASO DE USO: Operación SET	
Actor: Administrador de Red	
Descripción: Permite modificar el valor de un objeto dentro de la MIB.	
Activación: Presionando el botón SET	
Curso Normal	**Alternativas**
1. Ingresar el objeto identificador y el valor a modificar.	**1.1.** Si el Objeto Identificador es incorrecto se presentará un error.
2. Se presentará el nuevo valor del objeto de la MIB modificada.	
Precondiciones: El dispositivo debe soportar el protocolo SNMP.	
Postcondiciones: El Administrador tiene en pantalla los resultados de esta operación.	
Puntos de Extensión:	
Observaciones y Datos:	

- **Herramienta Ping**

En la tabla 2.5 se muestra el caso de uso para la herramienta Ping.

CASO DE USO: Herramienta Ping
Actor: Administrador de Red
Descripción: Permite realizar solicitudes eco con el fin de determinar si existe conexión física con los dispositivos de red.
Activación: Ingresando al Menú Herramientas - Ping.

Curso Normal	Alternativas
1. Se presenta una pantalla donde el usuario determina los parámetros para esta herramienta.	**1.1.** Si no existen dispositivos nos presenta una pantalla de error.
2. Se envía uno o más datagramas (Echo Request) al equipo remoto solicitando una respuesta (Echo Reply) y mide el tiempo que tarda en retornarla. Presentando estos resultados en pantalla.	
Precondiciones: Deben estar agregados dispositivos en la NMS.	
Postcondiciones: El Administrador tiene en pantalla los resultados de esta operación.	
Puntos de Extensión:	
Observaciones y Datos:	

Tabla 2.5 Caso de uso Herramienta Ping

- **Herramienta Browser**

En la tabla 2.6 se muestra el caso de uso para la herramienta Browser.

CASO DE USO: Herramienta Browser	
Actor: Administrador de Red	
Descripción: Permite conectarse vía browser al equipo remoto.	
Activación: Ingresando al Menú Herramientas - Browser.	
Curso Normal	**Alternativas**
1. Se ejecuta una pantalla de browser con la dirección IP del dispositivo gestionado.	**1.1.** Si no existen dispositivos nos presenta una pantalla de error.
Precondiciones: Deben estar agregados dispositivos en la NMS que soporten esta acción.	

Postcondiciones: El administrador está conectado al equipo remoto vía browser y puede realizar las configuraciones.
Puntos de Extensión:
Observaciones y Datos:

Tabla 2.6 Caso de uso Herramienta Browser

En la Figura 2.1 se muestra el Diagrama de caso de uso que involucra al administrador del sistema, representando cada una de las herramientas disponibles en la NMS y su respectiva secuencia.

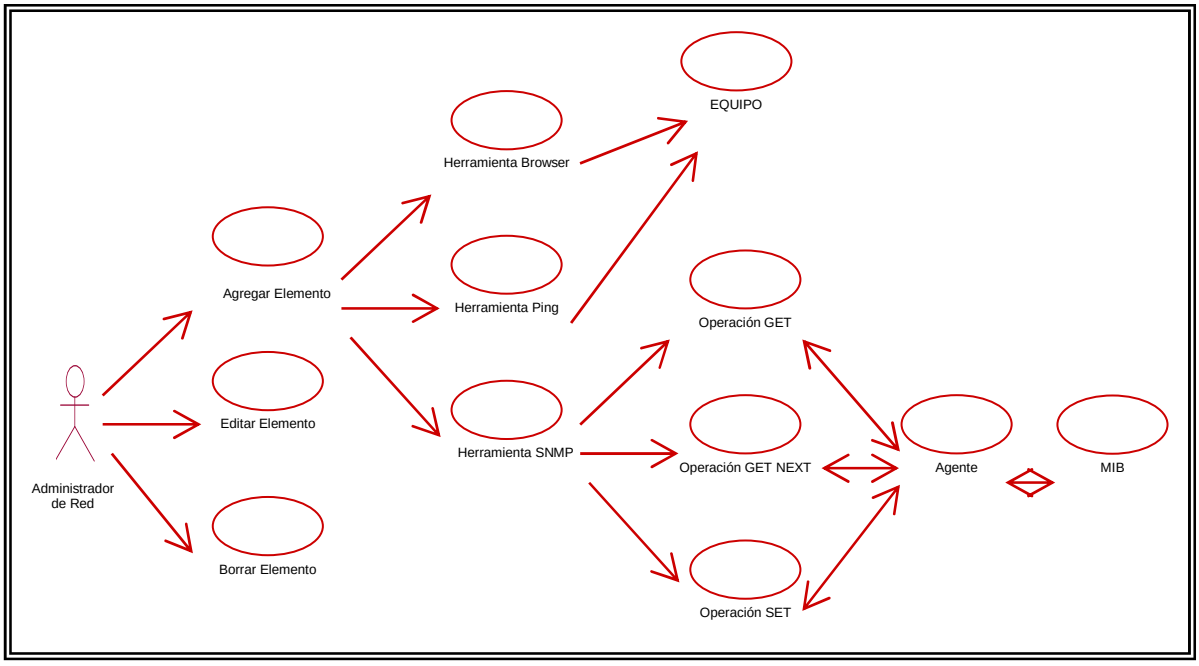


Figura 2.1 Diagrama de Caso de Uso Herramienta SNMP, Ping y Browser

2.1.2.1.3 Notificación de Eventos

La aplicación tiene la característica de aceptar todas las notificaciones de eventos que provienen de los dispositivos gestionados. Entre las herramientas a utilizarse tenemos: Trap y Polling.

- **Trap**

En la tabla 2.7 se muestra el caso de uso para las Traps.

CASO DE USO: Traps	
Actor: Agente	
Descripción: Envía notificaciones para reportar ciertas condiciones y cambios de estado a un proceso de administración.	
Activación: Al presentarse un evento de notificación.	
Curso Normal	**Alternativas**
1. Se envía una notificación hacia la NMS.	
Precondiciones: Deben estar agregados dispositivos a la NMS que soporten el protocolo SNMP.	
Postcondiciones: Llegan las notificaciones de las acciones a la NMS y se presentan en una ventana al Administrador.	
Puntos de Extensión:	
Observaciones y Datos:	

Tabla 2.7 Caso de uso Trap

En la figura 2.2 se muestra el diagrama de caso de uso de las traps.

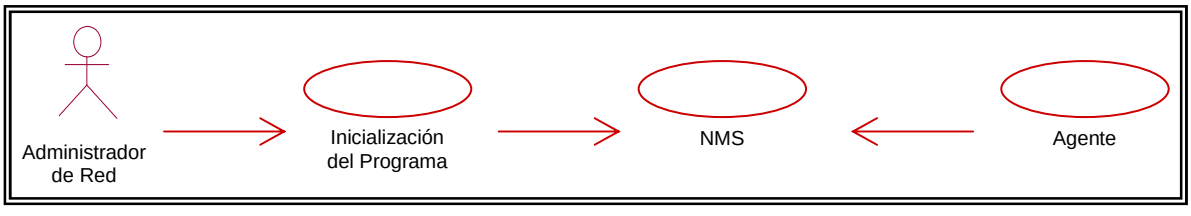


Figura 2.2 Diagrama de Caso de uso Trap

- **Polling**

En la tabla 2.8 se muestra el caso de uso para el sondeo o Polling.

CASO DE USO: Polling	
Actor: NMS	
Descripción: Envía pings cada cierto tiempo para determinar la conexión con el dispositivo remoto.	
Activación: Automática, cada 30 seg.	
Curso Normal	**Alternativas**
1. Se envía uno o más datagramas (Echo Request) al equipo remoto cada cierto tempo solicitando una respuesta (Echo Reply).	
Precondiciones: Deben estar agregados dispositivos en la NMS.	
Postcondiciones: El Administrador observa el estado de la conexión con el dispositivo remoto.	
Puntos de Extensión:	
Observaciones y Datos:	

Tabla 2.8 Caso de uso Polling

En la figura 2.3 se muestra el diagrama de caso de Uso para la herramienta polling

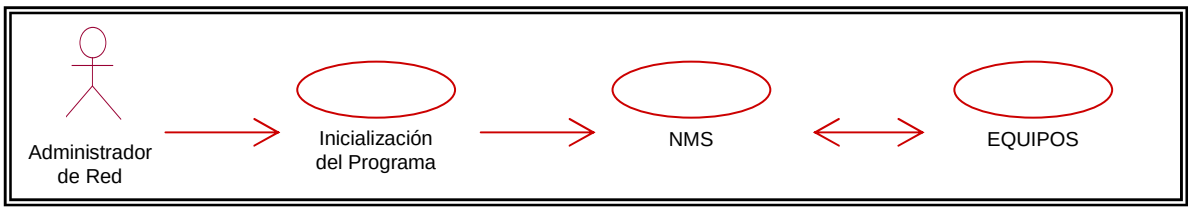


Figura 2.3 Diagrama de Caso de uso Polling

2.1.2.2 Requerimientos de Confiabilidad

La confiabilidad del programa (NMS para una PDA), depende de los Requerimientos Funcionales y No Funcionales tales como requerimientos de Hardware, de entorno y el requerimiento funcional del software.

2.1.2.2.1 Requerimientos en Hardware

A continuación se especificarán los requerimientos mínimos en hardware para que el programa NMS lleve a cabo su funcionalidad:

- **Puerto de conectividad inalámbrica:** Que permita el acceso de la información de los elementos de red gestionados a través de la red Wi-Fi.
- **Memoria RAM:** mínimo 64 MB, por la cantidad de información que se maneja.
- **Disco:** mínimo 10 MB, para almacenar la información de la NMS.
- **Procesador:** Mínimo 400 MHz.
- **Pantalla:** A color trasreflectiva TFT de alta resolución. Para facilitar la visualización de alarmas, y la información de los elementos de red.
- **Batería:** De alta duración y recargable, ya que el administrador de red debe mantener encendida la PDA las 24 horas del día para observar el correcto funcionamiento de la red.
- **Conector Universal:** Para conectar periféricos como teclados, para facilitar la inserción de caracteres en la etapa de creación de los elementos en la NMS.

2.1.2.2.2 Requerimientos del Entorno

Los requerimientos de entorno especifican la probabilidad de operación libre de fallas de un programa de computadora en un entorno determinado y durante un tiempo específico.

Para el desarrollo de nuestra aplicación es necesario que:

- La PDA donde se está ejecutando la NMS deberá estar conectada inalámbricamente vía WIFI en una red LAN, la cual va a administrar y monitorear los diferentes equipos que se encuentran operando en dicha red.

- Los diferentes equipos deberán estar configurados, es decir deben tener asignados una IP y levantados los servicios SNMP.

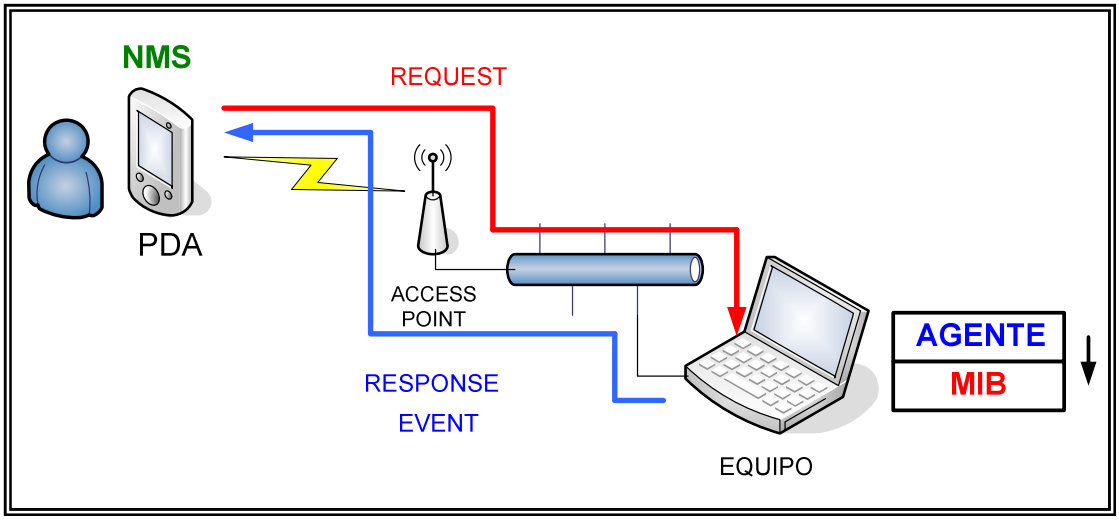


Figura 2.4 Requerimientos de entorno

2.1.2.3 Requerimientos de Seguridad, desempeño y escalabilidad

- ***Seguridad***

El software (NMS para PDA), trabaja con las versiones de SNMP uno y dos. Al utilizar el Protocolo SNMP versión uno, tendremos problemas de seguridad, ya que en esta versión se trabaja con comunidad “public”.

Se puede obtener seguridad autenticándose a través del nombre de la comunidad que utiliza el protocolo SNMP versión dos.

Se puede añadir seguridad estableciendo una conexión Wireless utilizando el protocolo WPA[10] para encripción de la información transmitida inalámbricamente.

En el Anexo A se hace una breve descripción del protocolo WPA.

[10] **Wi-Fi Protected Access:** Es una clase de sistemas de seguridad para redes inalámbricas soportado por el estándar IEEE 802.11i

- ***Desempeño y Escalabilidad***

El software (NMS para PDA) va a trabajar con el protocolo SNMP versión uno y dos. Esta aplicación trabajará con el concepto de “administrador” y “agente” por lo que no podría ser mejorado para trabajar con el protocolo SNMP V3, ya que este trabaja con el esquema de “Entidades” (Aplicación y Motor).

2.1.3 REQUERIMIENTOS AMBIENTALES

Los requerimientos ambientales describen el sistema de hardware, software y datos con los cuales va a trabajar nuestra NMS.

2.1.3.1 Requerimientos de hardware del software

Para el desarrollo de la NMS es necesario que la PDA cumpla los requerimientos de hardware que se detallan el la sección 2.1.2.2.1 de este capítulo.

Para esto haremos una comparación entre diferentes equipos que se muestran en el Anexo B.

Comparación General:

En la tabla 2.9 se muestra la comparación entre las diferentes PDA que se consideraron para la ejecución de nuestra aplicación.

Modelo	Fabricante	Procesador (MHz)	Memoria (MB)	Sistema Operativo	Soporte de Redes
Axim X5 PDA	Dell	400	64	Microsoft Pocket PC 2002	No
iPAQ hx2490b PDA	HP (Hewlett-Packard)	520	64	Windows Mobile 5	Wireless Ethernet - 11 Mbps

					IEEE802.11b
iPAQ hx2495 PDA	HP (Hewlett-Packard)	520	64	Windows Mobile 5	Wireless Ethernet - 11 Mbps IEEE802.11b
TX Handheld PDA	Palm	312	128	Palm OS	Wireless Ethernet - 11 Mbps IEEE802.11b
Tungsten E2 PDA - Multilingual	Palm	200	32	Palm OS	No
Pocket PC e310 PDA	Toshiba	206	32	Microsoft Pocket PC 2002	No
Clie PEG-T615C/S PDA	Sony	200	16	Palm OS	No

Tabla 2.9 Tabla de comparación de los equipos

Para el desarrollo de nuestra aplicación es indispensable que la PDA tenga soporte para comunicación Wi-Fi ya que nuestra aplicación será móvil y tendrá cobertura dentro de la empresa. Por ser una aplicación que trabajará con el protocolo SNMP y procesará gran cantidad de información se ve en la necesidad de que la memoria RAM sea mínimo de 64 MB y el procesador 400 MHz.

Al comparar costos y la disponibilidad de estos equipos en el mercado se escogió la PDA iPAQ hx2490b de HP para el desarrollo de nuestra aplicación.

2.1.3.2 Requerimientos del Lenguaje de programación para el desarrollo del software.

Los lenguajes de programación son utilizados para controlar el comportamiento de una máquina, en nuestro caso una PDA. Analizaremos las diferencias existentes entre C++, java y C# para determinar cual de estos favorece el desarrollo de aplicaciones en la PDA. La tabla 2.10 indica una comparación de las principales características de estos lenguajes de programación.

Lenguaje	C#	C++	JAVA
Facilidad de uso	Creación de funciones complejas desde cero y declaración de variables globales.	Creación de funciones complejas desde cero y declaración de variables globales.	Creación de funciones complejas desde cero y declaración de variables globales.
Programación orientada a objetos	Si, encapsulación, herencia simple y polimorfismo.	Si, encapsulación, herencia y polimorfismo.	Si, encapsulación, herencia y polimorfismo.
Archivos de Encabezado y directivas	En C# no existen archivos de encabezado, la directiva **using** se utiliza para hacer referencia a tipos en otros espacios de nombres sin necesidad de usar los nombres de tipo completos.	Existen archivos de encabezado y directivas #include en C++	Existen archivos de encabezado y directivas import en JAVA
Manejo de memoria	Manejo dinámico de memoria.	Manejo dinámico de memoria	Hace referencia de un objeto, mayor seguridad.
Compatibilidad con otros lenguajes	Permite incluir directamente en código escrito en C# fragmentos de código escrito en lenguajes C, C++ o Java.	Permite incluir directamente en código escrito en C++ fragmentos de código escrito en lenguajes C, C++.	Compatibilidad mediante JVM (Máquina virtual de JAVA).
Tipos de Datos	Existe un rango más amplio y definido de tipos de datos que los que se encuentran en C,	Tiene un rango definido de tipos de datos propios.	Tiene un rango definido de tipos de datos propios.

	C++ o Java.		
WINDOWS CE	Soportado con .NET Compact Framework.	Soportado con Win32.	No soportado. Soportado mediante una máquina virtual

Tabla 2.10 Tabla de comparación de los lenguajes de programación

C# compila a código intermedio (CIL)[11] independiente de la plataforma en que haya sido escrita la aplicación y del procesador donde vaya a ejecutarse. Es decir que una aplicación escrita en C# puede ser ejecutada en cualquier ambiente que soporte el .NET Framework[12]. El código CIL es verificado durante el tiempo de ejecución, proporcionando mayor seguridad y confiabilidad que los binarios compilados nativamente.

CLI está estandarizado, mientras que los bytecodes[13] de JAVA no lo están. El objetivo de CLI es hacer fácil la escritura de componentes y aplicaciones en lenguajes como C#, es decir permite definir tipos y componentes estandarizados y tener un alto rendimiento en la ejecución de una aplicación.

En C# no hay problemas de dependencias circulares, porque no existe el concepto de declaración e implementación de clases separadas. En C++ toda clase debe conocer la declaración de otra clase que quiera utilizar y el compilador debe ser capaz de encontrar la implementación.

C# tiene la capacidad de reflexión, permitiendo almacenar y obtener información en tiempo de ejecución sobre cualquier objeto. Se puede utilizar la reflexión para crear dinámicamente una instancia de un tipo, enlazar el tipo a un objeto existente u obtener el tipo a partir de un objeto existente, e invocar sus métodos o tener acceso a sus campos y propiedades.

[11] **CIL:** es un ensamblador orientado a objetos y basado en pilas.
[12] **.NET Framework:** es un componente de software que puede ser añadido al sistema operativo Windows y administra la ejecución de los programas escritos específicamente con la plataforma.
[13] **Bytecodes:** es un código intermedio más abstracto que el código máquina que contiene un programa ejecutable.

Para el desarrollo de nuestra aplicación escogimos C# ya que combina el control de lenguajes de bajo nivel como C y la velocidad de programación de lenguajes de alto nivel como Visual Basic. Además Windows CE soporta C# a través del paquete .NET Compact Framework.

2.1.3.3 Requerimientos de importación y exportación de datos.

Para la importación y exportación de datos utilizaremos el protocolo SNMP ya que facilita el intercambio de información de administración entre dispositivos de red.

Es indispensable que exista el Sistema de Administración de Red (NMS), el dispositivo gestionado y el Agente para poder realizar las importaciones y exportaciones de los datos. La figura 2.5 muestra el proceso de interacción de estos elementos.

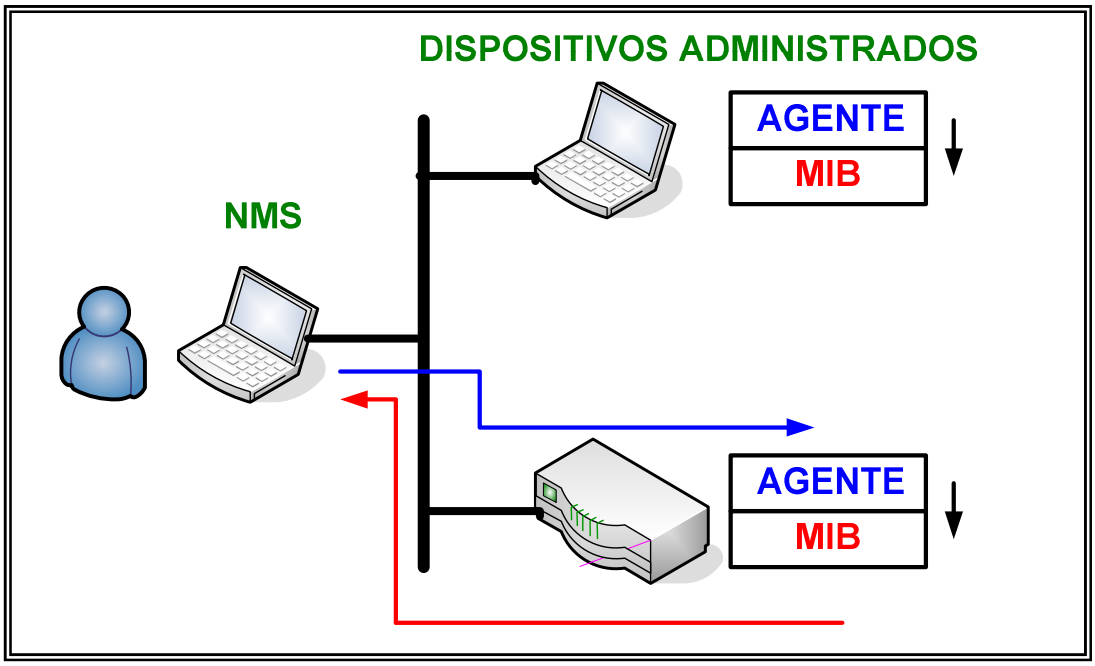


Figura 2.5 Componentes básicos de SNMP.

El Dispositivo administrado es un nodo de red que contiene un agente SNMP y reside en una red administrada. Estos recogen y almacenan información de administración, la cual es puesta a disposición de la NMS usando SNMP.

El Agente es un módulo de software de administración de red que reside en un dispositivo administrado. Un agente posee un conocimiento local de información de administración la cual es organizada en jerarquías descritas en la base de información de administración o MIB.

La NMS ejecuta aplicaciones que supervisan y controlan a los dispositivos administrados. La NMS proporciona el volumen de recursos de procesamiento y memoria requeridos para la administración de la red.

2.1.3.3.1 Base de información de administración SNMP (MIB)

Las MIBs son un tipo de base de datos que contiene información jerárquica, estructurada en forma de árbol, de todos los dispositivos gestionados en una red de comunicaciones. Define las variables usadas por el protocolo SNMP para supervisar y controlar los componentes de una red.

La MIB está compuesta por una serie de objetos que representan los dispositivos en la red. Cada objeto manejado en una MIB tiene un identificador de objeto único e incluye el tipo de objeto (tal como contador, secuencia o gauge), el nivel de acceso (tal como lectura y escritura), restricciones de tamaño, y la información del rango del objeto.

Un identificador de objeto (object ID) únicamente identifica un objeto administrado en la jerarquía MIB. La jerarquía MIB está representada en la figura 2.6 que se muestra como un árbol con una raíz y diferentes subniveles.

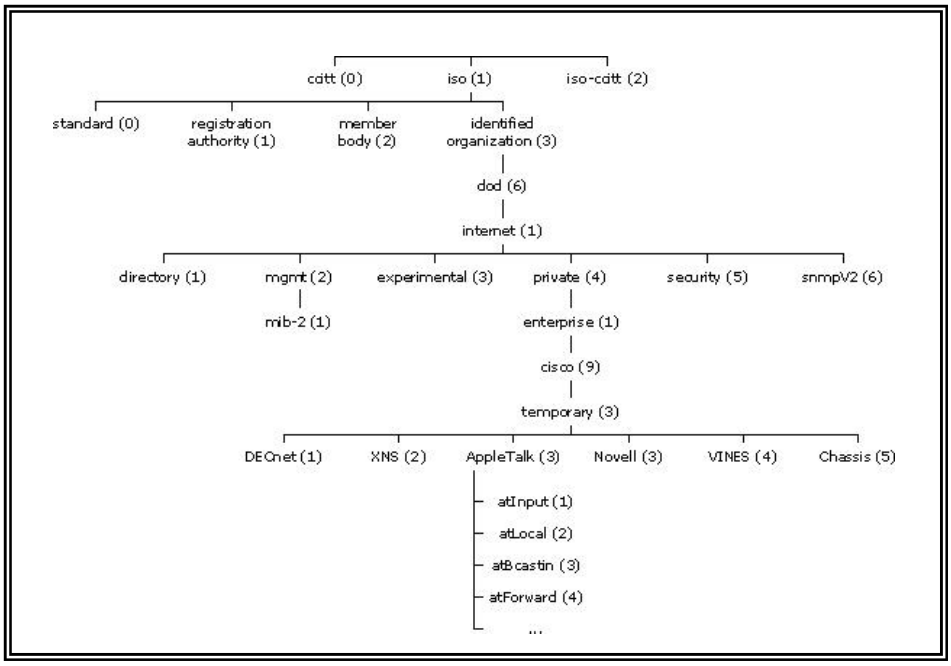


Figura 2.6 Árbol MIB

El árbol MIB ilustra las variadas jerarquías asignadas por las diferentes organizaciones.

Los identificadores de los objetos ubicados en la parte superior del árbol pertenecen a diferentes organizaciones estándares, mientras los identificadores de los objetos ubicados en la parte inferior del árbol son colocados por las organizaciones asociadas.

2.2 DIAGRAMA UML

Es necesario realizar un análisis preliminar que involucra un modelo conceptual, diagramas de secuencia y colaboración para obtener el diagrama UML.

2.2.1 MODELO CONCEPTUAL

El modelo conceptual explica los conceptos significativos de un dominio del problema.

Para obtener el modelo conceptual se debe seguir los siguientes pasos:

1. Listar e identificar los conceptos relacionados con los requerimientos en cuestión.
2. Dibujar los conceptos del modelo conceptual.
3. Incorporar las asociaciones necesarias para registrar las relaciones, para las cuales debe reservar un espacio en la memoria.
4. Agregar los atributos necesarios para cumplir con las necesidades de la información

2.2.1.1 Identificación de los Conceptos

Para la identificación de los conceptos, utilizaremos el método de obtención de conceptos a partir de una lista de conceptos, el cual consiste en preparar una lista de conceptos idóneos que contengan las categorías comunes sin importar el orden.

Categoría del Concepto	Ejemplo
Actores	NMS (Administrador) Agente (Dispositivo)
Funciones SNMP	GET GETNEXT SET TRAP
Funciones utilitarias	PING BROWSER
Esquema de codificación	BER
Tipos de datos	INT STRING GAUGE IPADDRESS OID COUNTER COUNTER32

	TIMESTICK NULL SEQUENCE
Comunicación	SESION SESION TRAP

Tabla 2.11 Tabla de categoría de conceptos

2.2.1.2 Diagrama de los Conceptos

A partir de la tabla 2.11 se obtuvieron los siguientes conceptos:

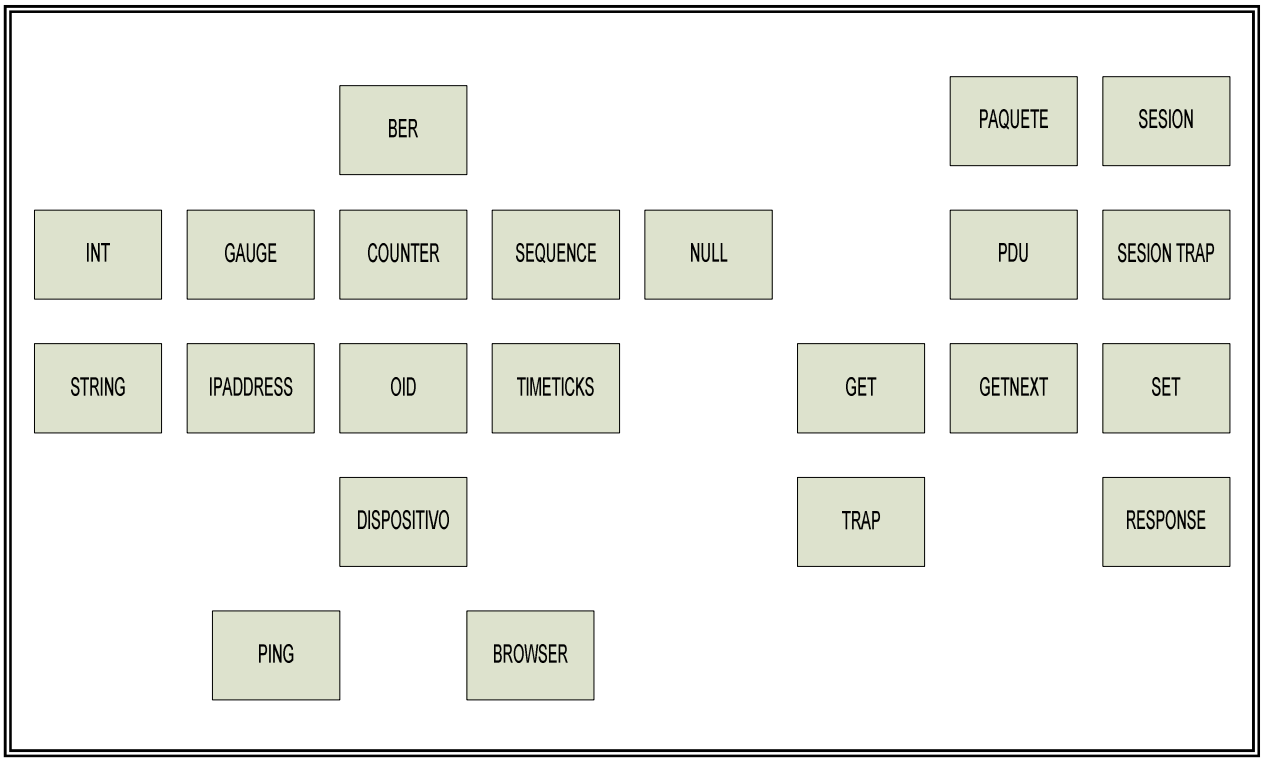


Figura 2.7 Diagrama de conceptos

2.2.1.3 Relación entre los Conceptos

A partir de los conceptos idóneos determinamos las relaciones entre estos:

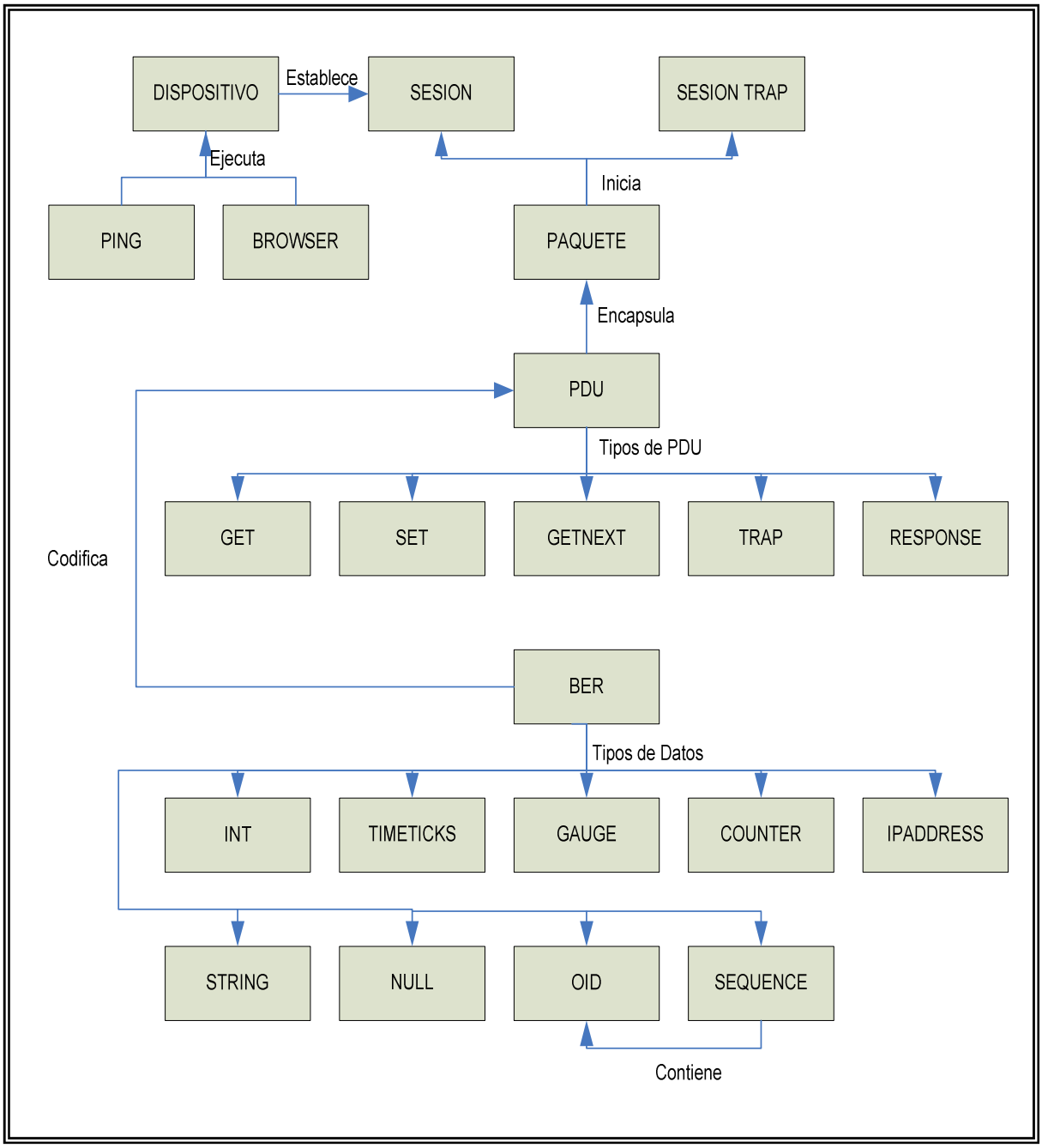


Figura 2.8 Relaciones entre los conceptos

2.2.1.4 Atributos de los Conceptos

Determinaremos los atributos principales de cada concepto:

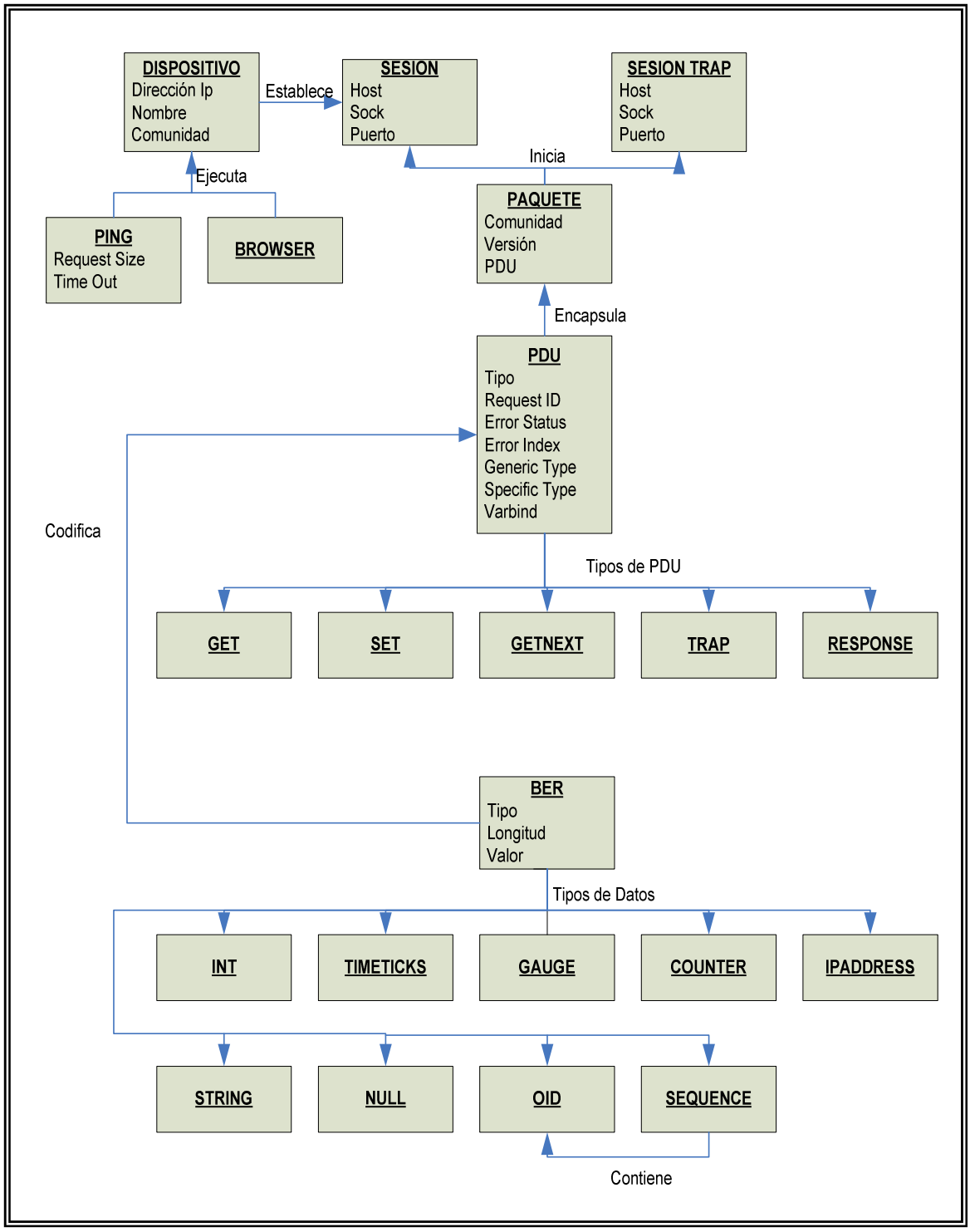


Figura 2.9 Modelo Conceptual

2.2.2 DIAGRAMAS DE SECUENCIA Y COLABORACIÓN

El diagrama de secuencia indica las interacciones entre los objetos y los mensajes que intercambian de una forma temporal, para llevar a cabo la funcionalidad descrita por el escenario.

Los diagramas de secuencia se utilizan para validar los diagramas de caso de uso, además nos ayudan a identificar los cuellos de botella potenciales, para así eliminarlos.

A diferencia de los diagramas de secuencia, los diagramas de colaboración muestran las interacciones que ocurren entre los objetos que participan en una situación determinada. Esta es más o menos la misma información que la mostrada por los diagramas de secuencia, pero destacando la forma en que las operaciones se producen en el tiempo, mientras que los diagramas de colaboración fijan el interés en las relaciones entre los objetos y su topología.

A continuación se presentará los diagramas de secuencia y colaboración basándose en los diagramas de caso presentados en las secciones anteriores.

2.2.2.1 Agregar Nuevo Dispositivo

Para monitorear un dispositivo necesitamos que el administrador ingrese la información requerida por el programa.

En la figura 2.10 y 2.11 se muestran los diagramas de secuencia y colaboración para está acción.

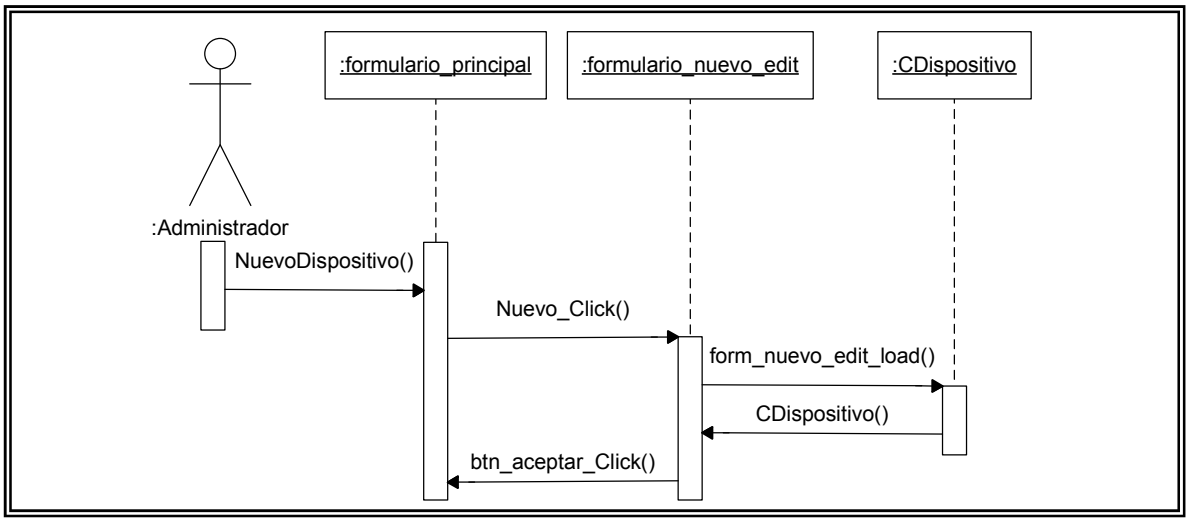


Figura 2.10 Diagrama de secuencia para agregar un nuevo dispositivo

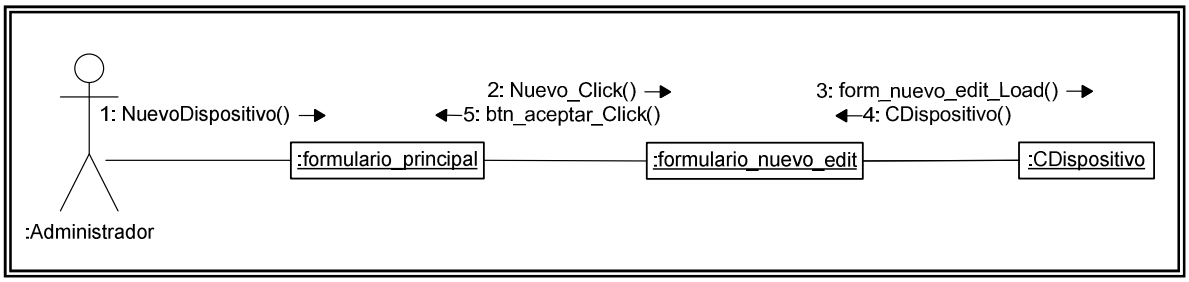


Figura 2.11 Diagrama de colaboración para agregar un nuevo dispositivo

2.2.2.2 Editar Información del Dispositivo

Ya ingresado un dispositivo a la aplicación puede ocurrir la necesidad de cambiar alguno de sus valores, para esto se usa la función editar.

En la figura 2.12 y 2.13 se muestran los diagramas de secuencia y colaboración para esta acción.

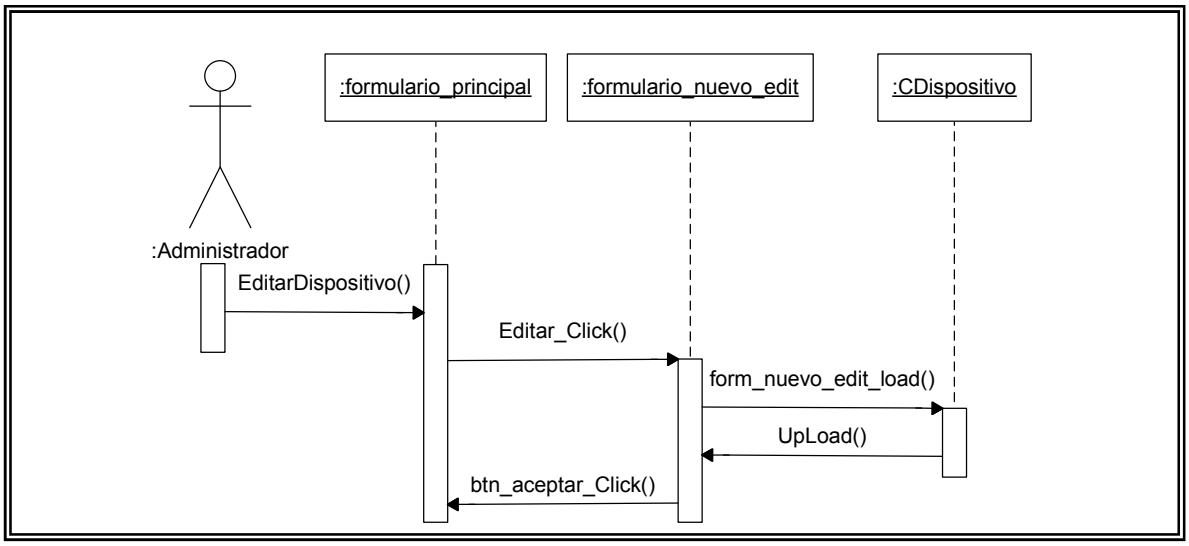


Figura 2.12 Diagrama de secuencia para editar la información de un dispositivo

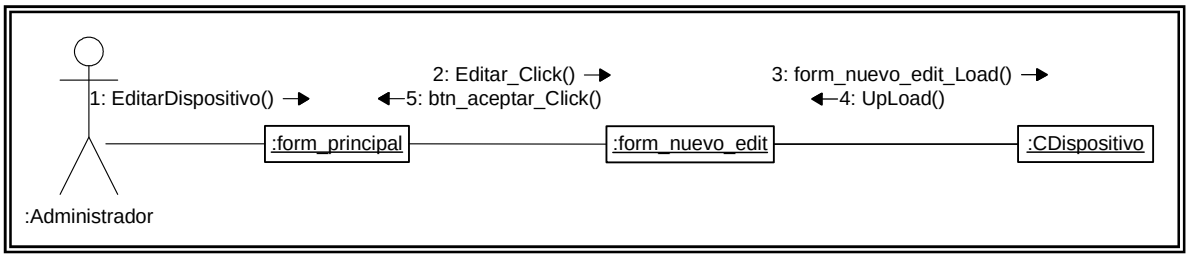


Figura 2.13 Diagrama de colaboración para editar la información de un dispositivo

2.2.2.3 Borrar Dispositivo

El administrador tiene la posibilidad de eliminar un dispositivo que se encuentre ingresado en el programa. Para este escenario el dispositivo es borrado del arreglo de dispositivos ingresados en el programa.

En la figura 2.14 y 2.15 se muestran los diagramas de secuencia y colaboración para está acción.

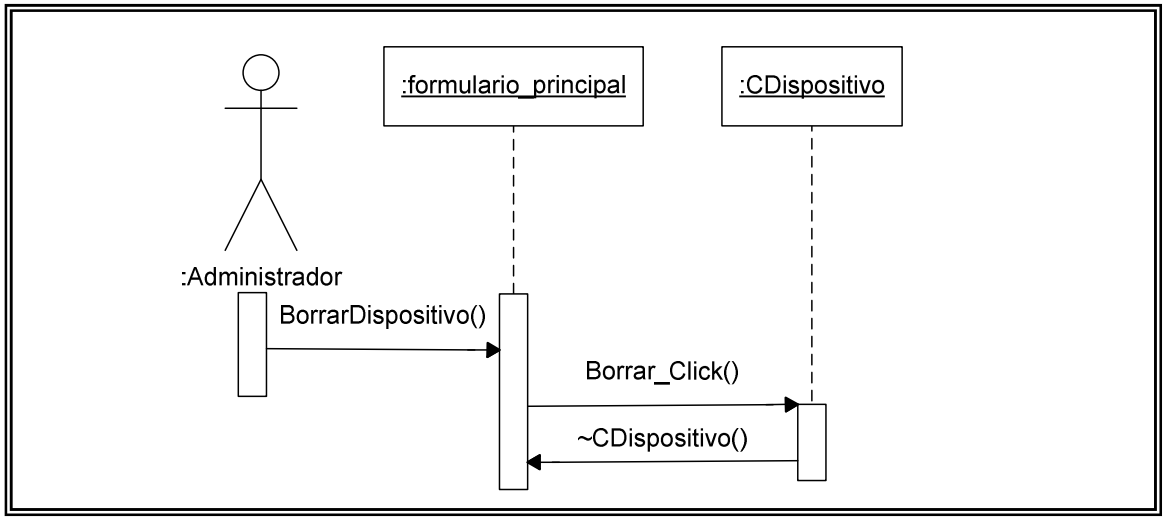


Figura 2.14 Diagrama de secuencia para borrar un dispositivo

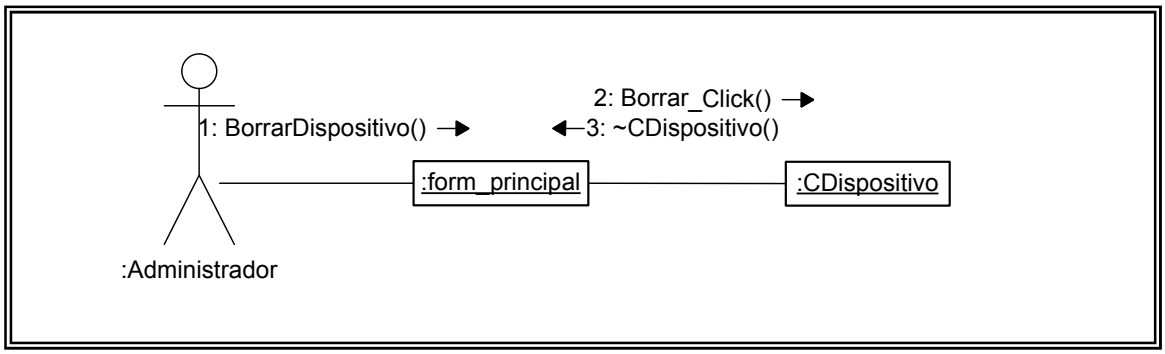


Figura 2.15 Diagrama de colaboración para borrar un dispositivo

2.2.2.4 Guardar Dispositivos

El administrador tiene la posibilidad de guardar los dispositivos en un fichero para ser utilizado en una próxima ocasión. Los dispositivos guardados en el fichero podrán ser cargados en el programa cuando éste inicialice nuevamente.

En la figura 2.16 y 2.17 se muestran los diagramas de secuencia y colaboración para está acción.

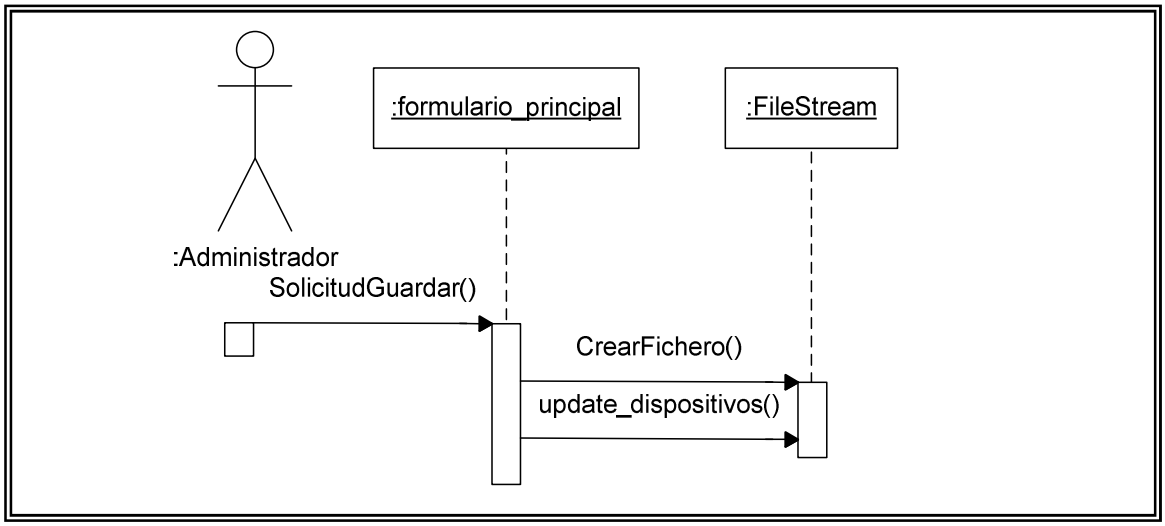


Figura 2.16 Diagrama de secuencia para guardar dispositivos

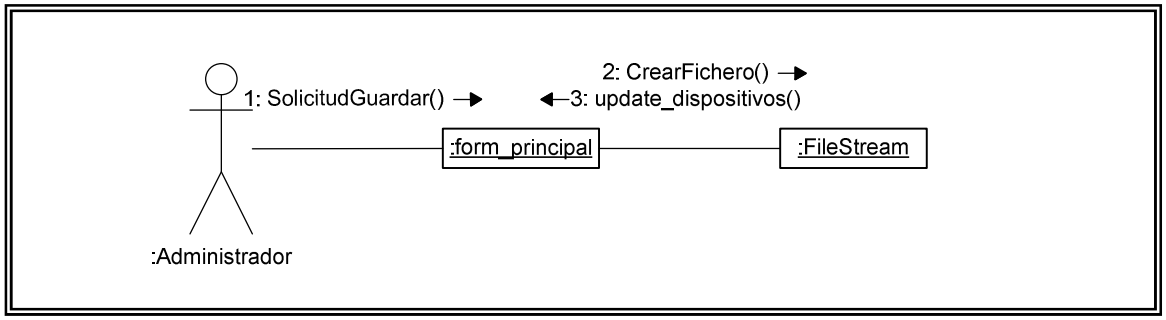


Figura 2.17 Diagrama de colaboración para guardar dispositivos

2.2.2.5 Operación SNMP Tipo Get

Para realizar una operación SNMP tipo GET, el administrador debe ingresar el OID (objeto identificador) del objeto MIB que se va a consultar. En el diagrama de secuencia se observa la conformación de la PDU tipo GET y su respectivo paquete SNMP, para luego ser enviado al agente del dispositivo implicado y recibir una respuesta.

En la figura 2.18 y 2.19 se muestran los diagramas de secuencia y colaboración para está operación.

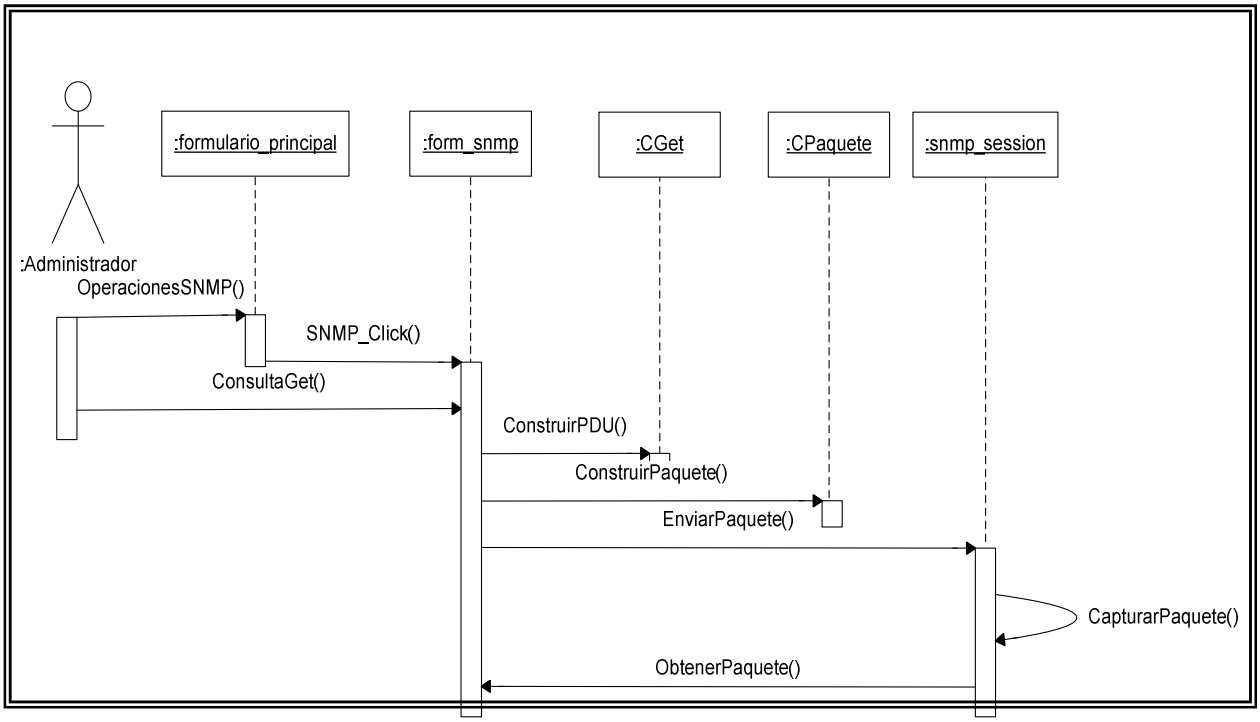


Figura 2.18 Diagrama de secuencia para la operación SNMP tipo GET

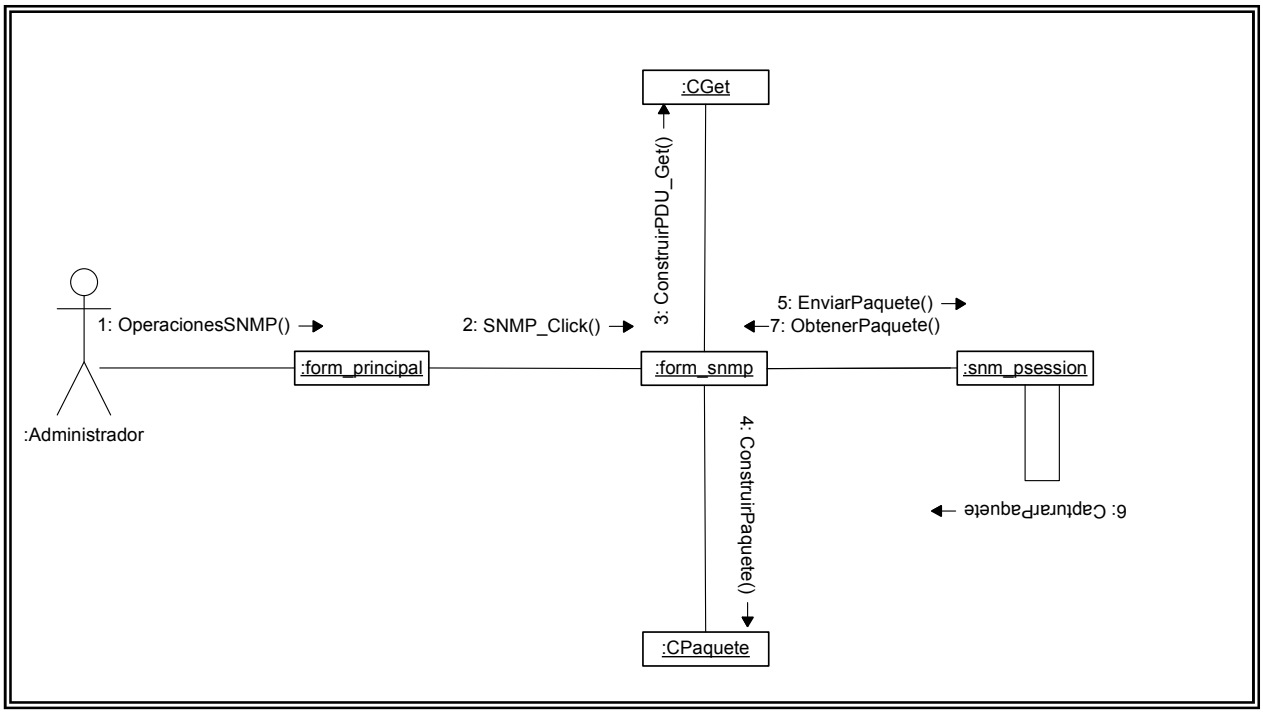


Figura 2.19 Diagrama de colaboración para la operación SNMP tipo GET

2.2.2.6 Operación SNMP tipo GET NEXT

Este escenario es similar a la operación SNMP tipo GET, excepto que el administrador puede consultar el siguiente valor de un objeto de la tabla de datos dentro de la MIB, permitiéndole consultar el valor de los objetos MIB en una forma ordenada y progresiva.

En la figura 2.20 y 2.21 se muestran los diagramas de secuencia y colaboración para está operación.

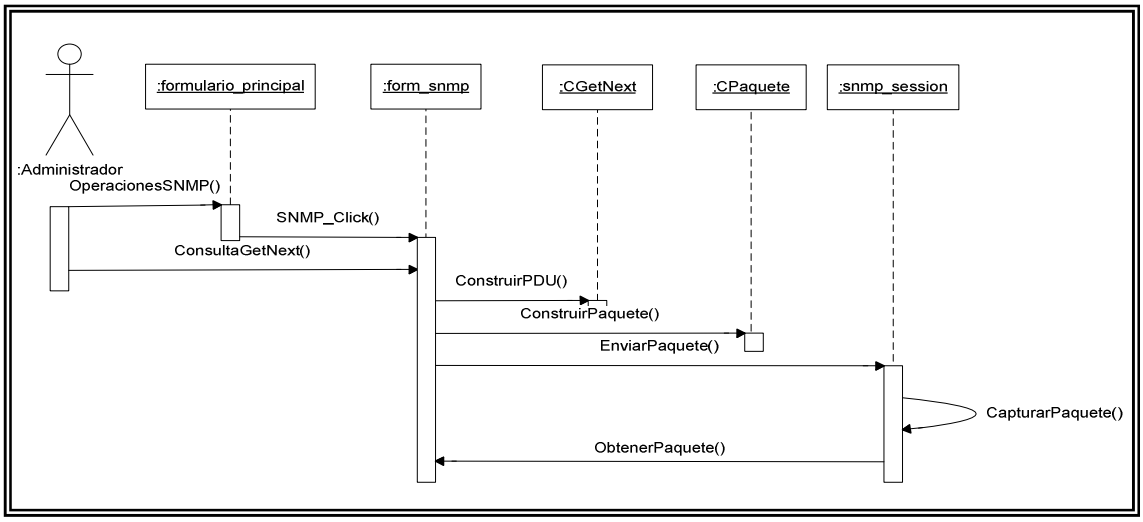


Figura 2.20 Diagrama de secuencia para la operación SNMP tipo GET NEXT

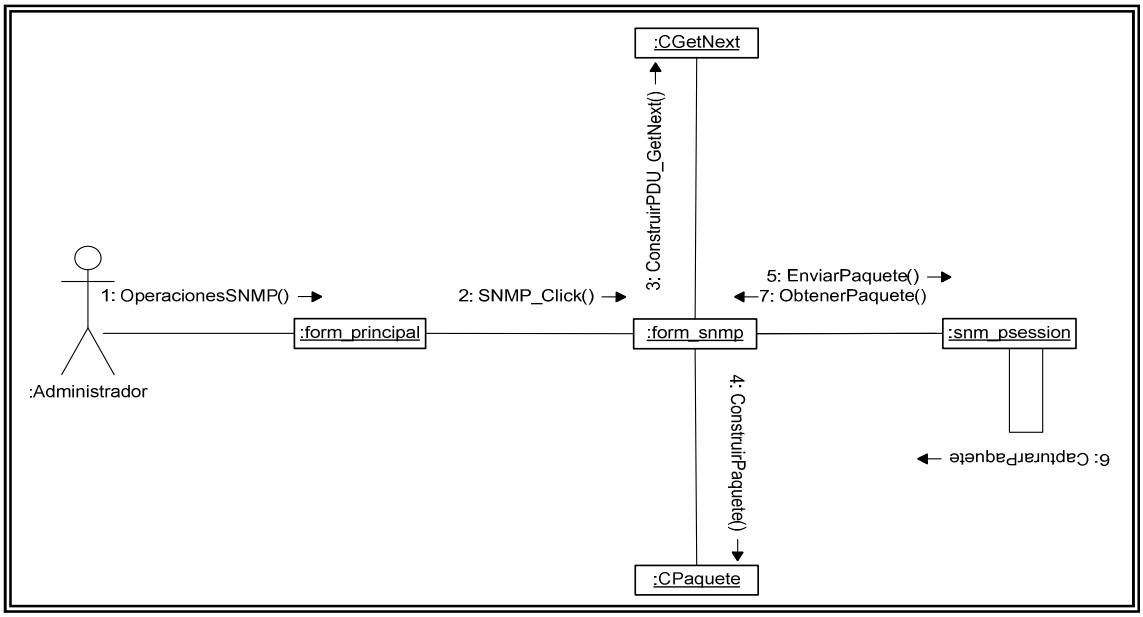


Figura 2.21 Diagrama de colaboración para la operación SNMP tipo GET NEXT

2.2.2.7 Operación SNMP tipo SET

En la operación SNMP tipo SET, el administrador debe ingresar el valor y su respectivo identificador del objeto MIB que se desea modificar. Esta operación obtendrá como resultado el valor del objeto MIB modificado.

En la figura 2.22 y 2.23 se muestran los diagramas de secuencia y colaboración para está operación.

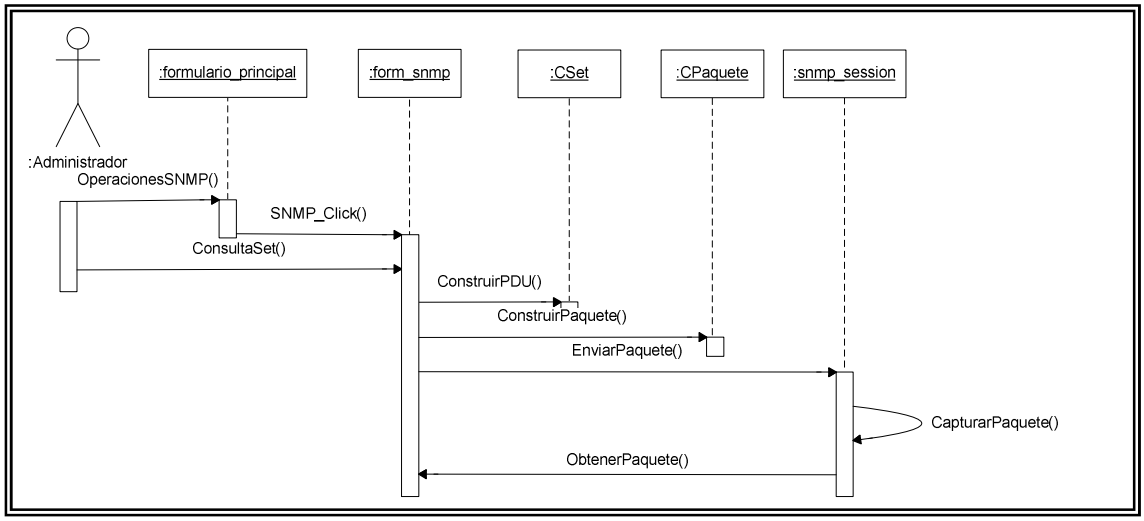


Figura 2.22 Diagrama de secuencia para la operación SNMP tipo SET

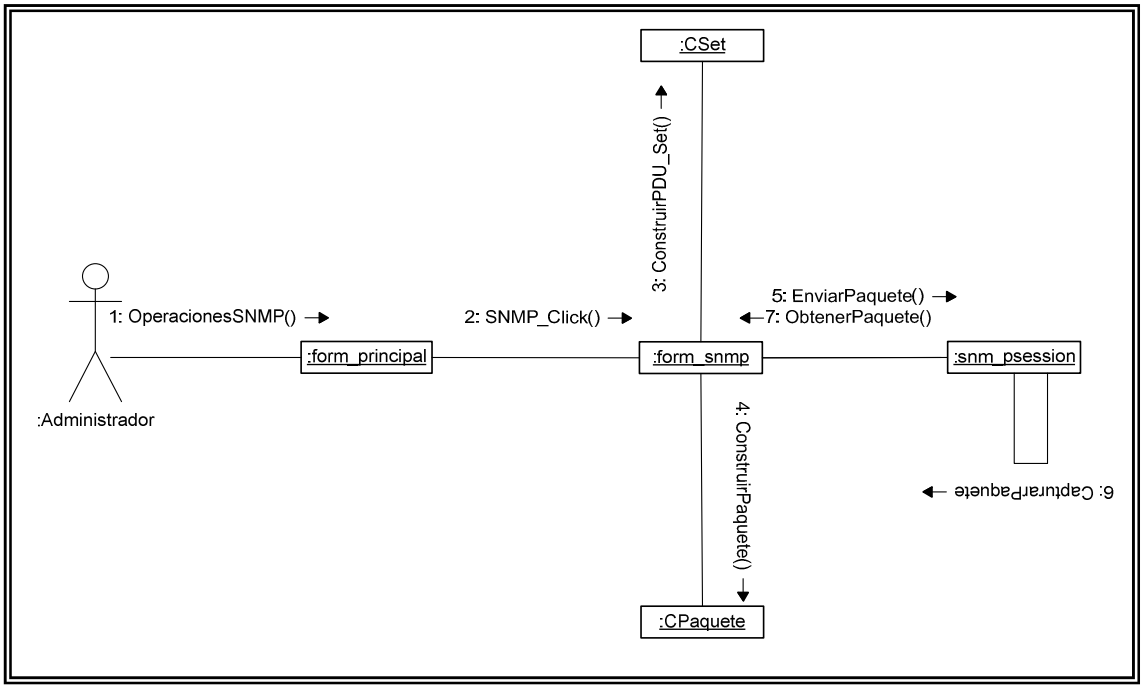


Figura 2.23 Diagrama de colaboración para la operación SNMP tipo SET

2.2.2.8 Operación SNMP tipo TRAP

Esta operación entra en ejecución una vez que se ha ingresado al menos un dispositivo, su operación se ejecuta paralelamente con el programa principal, recibiendo indefinidamente paquetes SNMP tipo TRAP, los cuales se presentan en un formulario cuando sucede algún evento.

En la figura 2.24 y 2.25 se muestran los diagramas de secuencia y colaboración para está operación.

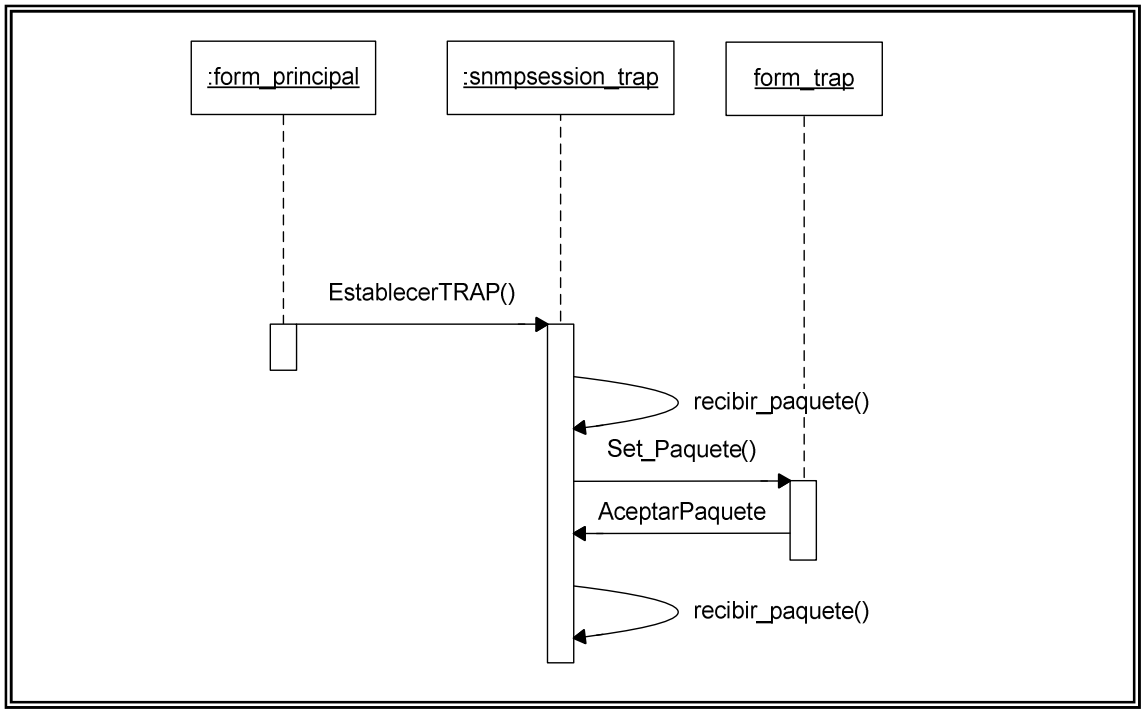


Figura 2.24 Diagrama de secuencia para la operación SNMP tipo TRAP

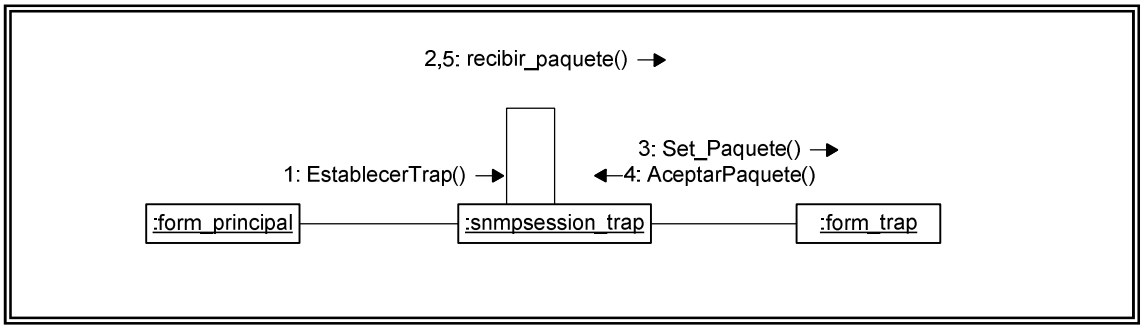


Figura 2.25 Diagrama de colaboración para la operación SNMP tipo TRAP

2.2.2.9 Herramienta Ping

El administrador puede solicitar esta herramienta permitiendo comprobar la conectividad del equipo monitorizado, cuyo resultado ICMP de la clase PING es presentado en el formulario respectivo.

En la figura 2.26 y 2.27 se muestran los diagramas de secuencia y colaboración de está herramienta.

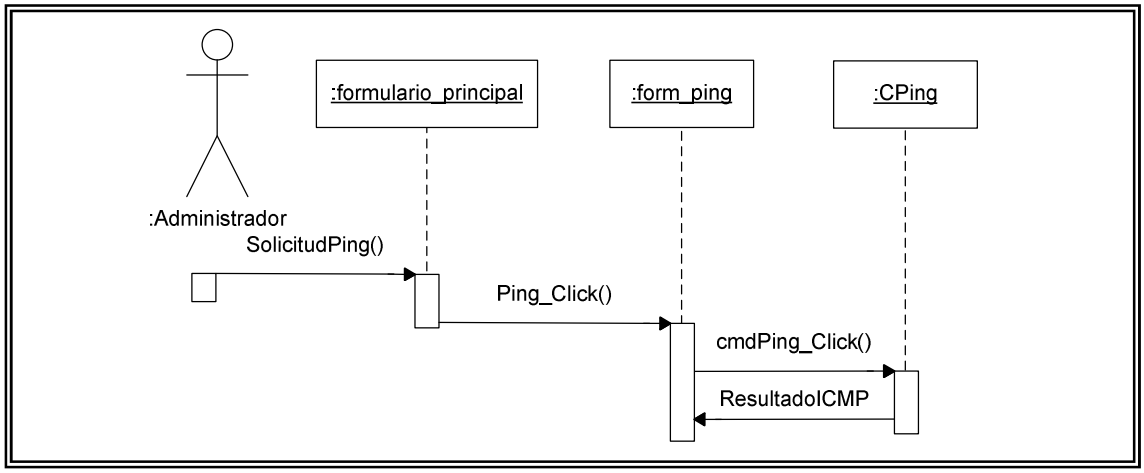


Figura 2.26 Diagrama de secuencia para la herramienta PING

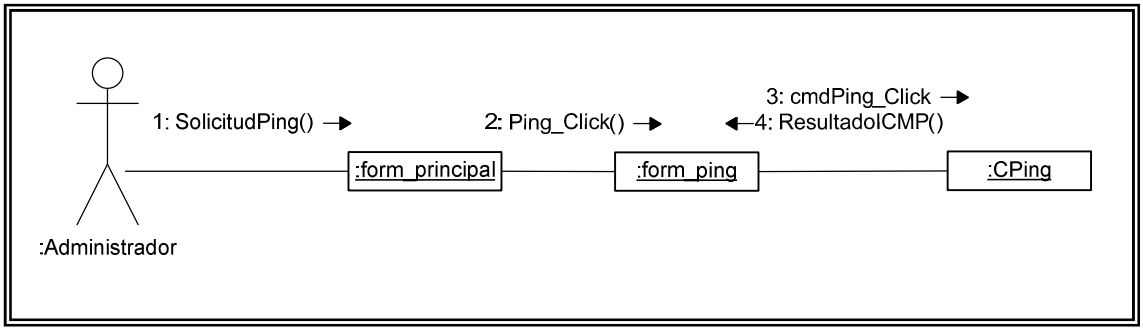


Figura 2.27 Diagrama de colaboración para la herramienta PING

2.2.2.10 Herramienta Browser

Existen dispositivos que permiten ser configurados remotamente mediante un browser. Para esta operación se crea un objeto tipo Process, el cual permite ejecutar la aplicación iexplore con la URL del equipo que se desea configurar.

En la figura 2.28 y 2.29 se muestran los diagramas de secuencia y colaboración de está herramienta.

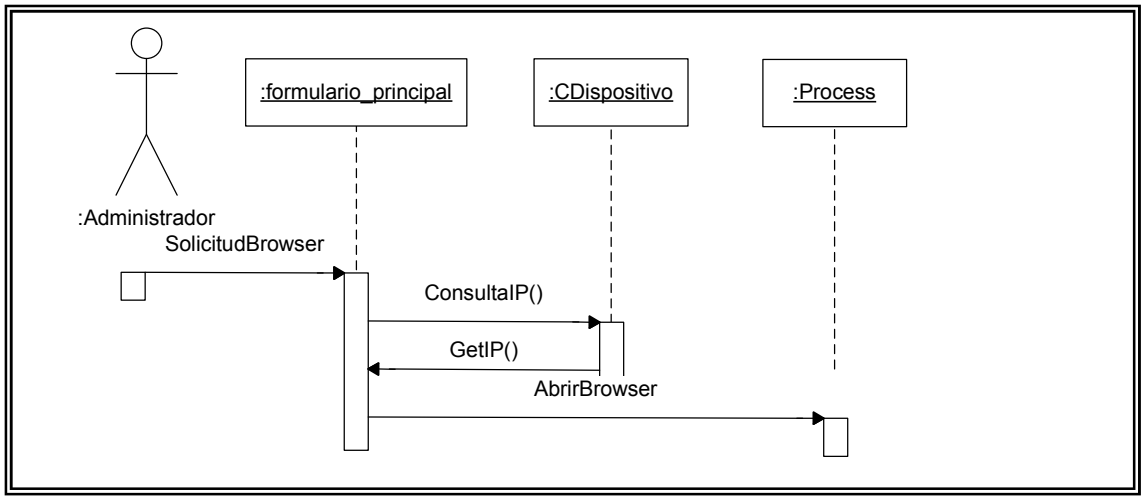


Figura 2.28 Diagrama de secuencia para la herramienta browser

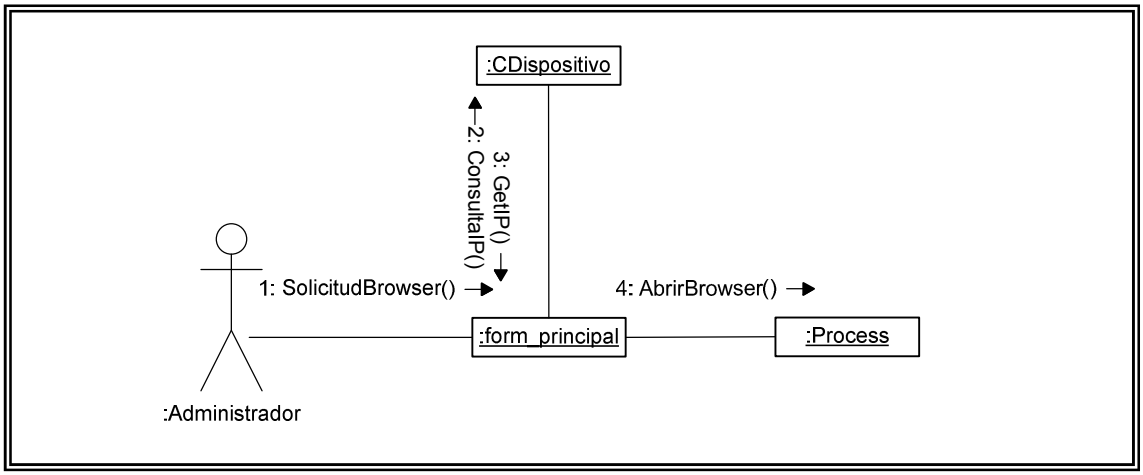


Figura 2.29 Diagrama de colaboración para la herramienta browser

2.2.3 CONTRATOS.

Los Contratos son documento que describe lo que una operación se propone lograr. Los contratos suelen expresarse a partir de los cambios de estado de las precondiciones y poscondiciones.

Para nuestra aplicación es necesario analizar los siguientes contratos:

CONTRATO SESIÓN SNMP	
Nombre:	- Csnmpsesion.
Responsabilidades:	- Establecer conexión con los dispositivos gestionados. - Enviar y Recibir paquetes SNMP.
Tipo:	- Clase.
Notas:	- Utiliza socket hacia el puerto 161.
Excepciones:	- No se establece la sesión con equipos que no soporten el protocolo snmp.
Salidas:	-
Precondiciones:	- Debe existir conexión física con los dispositivos gestionados. - Los equipos gestionados deben tener habilitados los servicios SNMP.
Poscondiciones:	- Envío y Recepción de paquetes SNMP.

CONTRATO SESIÓN TRAP	
Nombre:	- Csnmpsesiontrap.
Responsabilidades:	- Recibir las notificaciones enviadas por los agentes.
Tipo:	- Clase.
Notas:	- Utiliza socket hacia el puerto 162.
Excepciones:	- No se establece la sesión con equipos que no soporten el protocolo snmp.
Salidas:	-
Precondiciones:	- Debe existir conexión física con los dispositivos gestionados. - Los equipos gestionados deben tener habilitados los servicios SNMP
Poscondiciones:	- Recepción de notificaciones (traps).

CONTRATO DISPOSITIVO	
Nombre:	- CDispositivo.
Responsabilidades:	- Crear los dispositivos a ser gestionados.
Tipo:	- Clase.
Notas:	- Para definirlo es necesario la dirección IP, nombre, tipo y comunidad.
Excepciones:	-
Salidas:	-
Precondiciones:	- Debe existir conexión física con los dispositivos gestionados. - Nombre de comunidades definidas
Poscondiciones:	- Establece una sesión SNMP

CONTRATO PING	
Nombre:	- CPing.
Responsabilidades:	- Enviar y Recibir paquetes ICMP.
Tipo:	- Clase.
Notas:	- Necesita la librería Iphlpapi.dll
Excepciones:	- Si el equipo esta desconectado no existe respuesta.
Salidas:	-
Precondiciones:	- Debe existir conexión física con los dispositivos gestionados.
Poscondiciones:	- Recibe el paquete ICMP request.

CONTRATO PAQUETE SNMP	
Nombre:	- CPaquete.
Responsabilidades:	- Codificar y Decodificar el paquete SNMP.
Tipo:	- Clase.
Notas:	- Codificación y decodificación bit a bit.

Excepciones:	- No codifica paquetes SNMP v3.
Salidas:	-
Precondiciones:	- PDU SMNP construida.
Poscondiciones:	- Inicio de sesión SNMP.

CONTRATO PDU SNMP	
Nombre:	- CPDU.
Responsabilidades:	- Codificar y Decodificar PDU SNMP.
Tipo:	- Clase.
Notas:	- Codificación y decodificación bit a bit.
Excepciones:	- No codifica SNMP v3.
Salidas:	-
Precondiciones:	- Tipos de Datos codificados de acuerdo a BER (Reglas Básicas de Codificación)
Poscondiciones:	- Se encapsula en el paquete SNMP

CONTRATO BER	
Nombre:	- CBER.
Responsabilidades:	- Codificar y Decodificar tipos de objetos.
Tipo:	- Clase.
Notas:	- Codificación y decodificación bit a bit.
Excepciones:	- Codifica los tipos de datos INT, GAUGE, STRING, IPADRESS, OID, COUTER, COUNTER32, TIMETICKS, NULL Y SEQUENCE.
Salidas:	-
Precondiciones:	- Tipos de datos SNMP V1 y V2
Poscondiciones:	- Codificación de los tipos de datos.

CONTRATO GET	
Nombre:	- CGet.
Responsabilidades:	- Codificar y Decodificar PDU SNMP tipo Get.
Tipo:	- Clase.
Notas:	- Codificación y decodificación bit a bit.
Excepciones:	- No codifica SNMP v3.
Salidas:	-
Precondiciones:	- Tipos de Datos codificados de acuerdo a BER (Reglas Básicas de Codificación)
Poscondiciones:	- Se encapsula en el paquete SNMP

CONTRATO GETNEXT	
Nombre:	- CGetNext.
Responsabilidades:	- Codificar y Decodificar PDU SNMP tipo GetNext.
Tipo:	- Clase.
Notas:	- Codificación y decodificación bit a bit.
Excepciones:	- No codifica SNMP v3.
Salidas:	-
Precondiciones:	- Tipos de Datos codificados de acuerdo a BER (Reglas Básicas de Codificación)
Poscondiciones:	- Se encapsula en el paquete SNMP

CONTRATO SET	
Nombre:	- CSet.
Responsabilidades:	- Codificar y Decodificar PDU SNMP tipo Set.
Tipo:	- Clase.
Notas:	- Codificación y decodificación bit a bit.

Excepciones:	- No codifica SNMP v3.
Salidas:	-
Precondiciones:	- Tipos de Datos codificados de acuerdo a BER (Reglas Básicas de Codificación)
Poscondiciones:	- Se encapsula en el paquete SNMP

CONTRATO RESPONSE	
Nombre:	- CResponse.
Responsabilidades:	- Codificar y Decodificar PDU SNMP tipo Response.
Tipo:	- Clase.
Notas:	- Codificación y decodificación bit a bit.
Excepciones:	- No codifica SNMP v3.
Salidas:	-
Precondiciones:	- Tipos de Datos codificados de acuerdo a BER (Reglas Básicas de Codificación)
Poscondiciones:	- Se encapsula en el paquete SNMP

CONTRATO TRAP	
Nombre:	- CTrap.
Responsabilidades:	- Codificar y Decodificar PDU SNMP tipo Trap V1 y V2.
Tipo:	- Clase.
Notas:	- Decodificación bit a bit.
Excepciones:	- No codifica SNMP v3.
Salidas:	-
Precondiciones:	- Tipos de Datos codificados de acuerdo a BER (Reglas Básicas de Codificación)
Poscondiciones:	- Se decodifica en el paquete SNMP

CONTRATO FORMULARIO PRINCIPAL	
Nombre:	- Form_principal.
Responsabilidades:	- Interfaz grafica con acceso a todas las funciones de la NMS.
Tipo:	- Formulario.
Notas:	- Interactúa con las funciones Nuevo, Editar, Borrar, Salir, Guardar, servicios SNMP, Ping y Browser
Excepciones:	-
Salidas:	- Formularios de las funciones.
Precondiciones:	- El sistema debe ejecutarse correctamente.
Poscondiciones:	-

CONTRATO FORMULARIO NUEVO/EDITAR	
Nombre:	- Form_nuevo_edit
Responsabilidades:	- Interfaz grafica para agregar y editar los dispositivos de la red.
Tipo:	- Formulario.
Notas:	- Para editar un dispositivo este debe estar ingresado en el sistema.
Excepciones:	-
Salidas:	-
Precondiciones:	- Debe existir dispositivos conectados a la red.
Poscondiciones:	- Regresa al formulario principal.

CONTRATO FORMULARIO DE SERVICIOS SNMP	
Nombre:	- Form_snmp.
Responsabilidades:	- Interfaz grafica para acceder a los servicios snmp.
Tipo:	- Formulario.
Notas:	- Soporta SNMP V1 y V2.

Excepciones:	- Si se ingresa un OID incorrecta se presenta un mensaje de error.
Salidas:	-
Precondiciones:	- Debe existir dispositivos conectados a la red.
Poscondiciones:	- Resultado de consultas y modificaciones.

CONTRATO FORMULARIO DEL UTILITARIO PING	
Nombre:	- Form_ping.
Responsabilidades:	- Interfaz grafica para acceder a los servicios ping.
Tipo:	- Formulario.
Notas:	-
Excepciones:	-
Salidas:	-
Precondiciones:	-
Poscondiciones:	- Resultado de la consulta.

CONTRATO FORMULARIO TRAP	
Nombre:	- Form_trap.
Responsabilidades:	- Interfaz grafica donde muestra las notificaciones enviadas por los agentes de los dispositivos gestionados.
Tipo:	- Formulario.
Notas:	-
Excepciones:	- No soporta SNMP v3
Salidas:	-
Precondiciones:	- Debe existir dispositivos conectados a la red.
Poscondiciones:	- Resultado de la consulta.

2.2.4 DIAGRAMA UML DE CLASES.

En la Figura 2.30 se muestra el diagrama UML (Diagrama de clases) del software de Monitorización de Red en dispositivos inalámbricos (PDA) para la administración y gestión de red, basado en los diagramas de caso de uso, secuencia, colaboración y modelo conceptual.

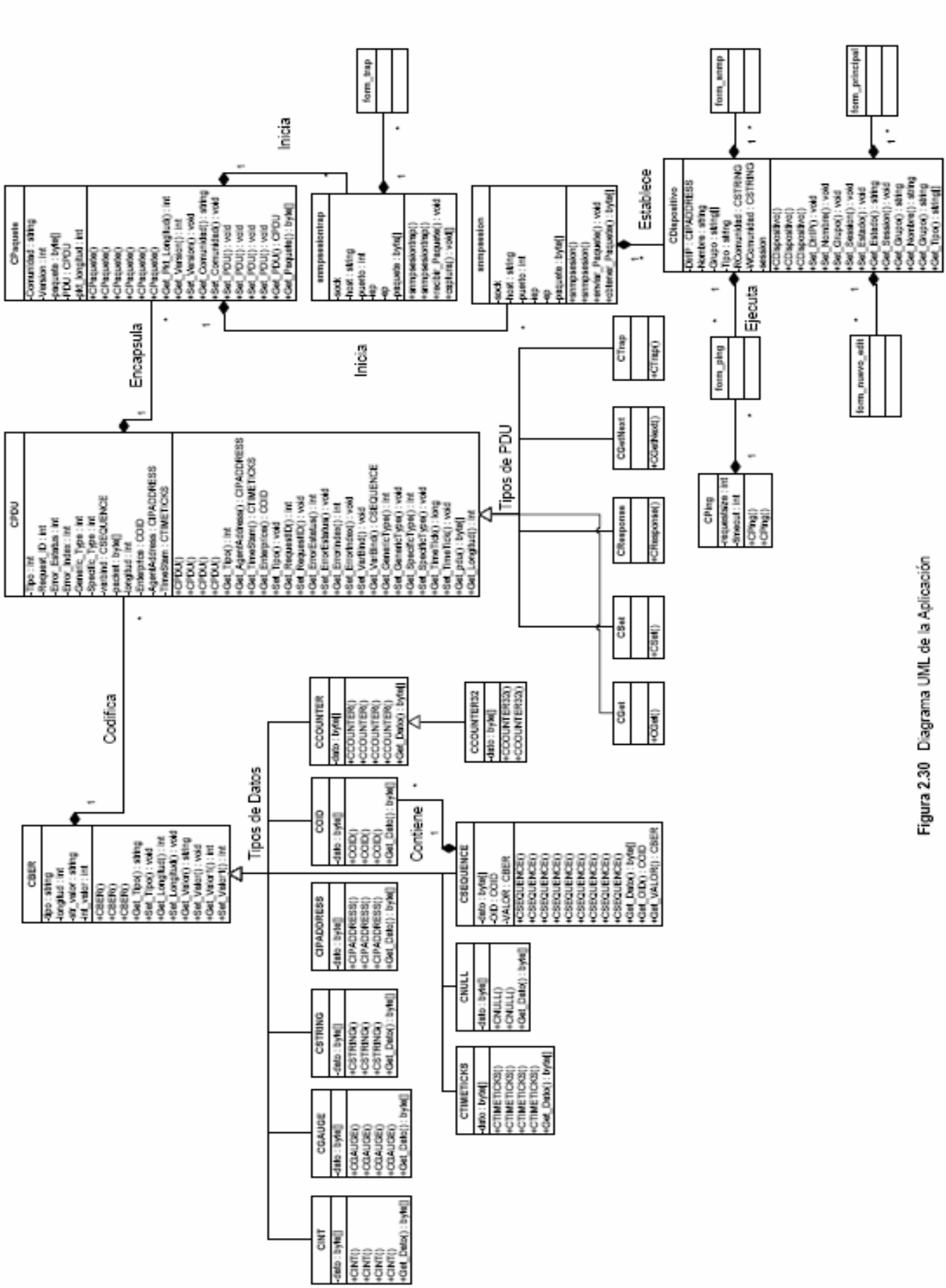


Figura 2.30 Diagrama UML de la Aplicación

2.3 DESARROLLO DE SOFTWARE

2.3.1. PLATAFORMA DE DESARROLLO Y LENGUAJE

Para el desarrollo de nuestra aplicación se escogió C# debido a las siguientes características:

- Es orientado a objetos permitiendo definir al programa en término de “clases de objetos”, logrando tener un software con mayor usabilidad, y escalabilidad.

- Su facilidad de combinar el control de bajo nivel como C, que en nuestro caso utilizaremos “sockets” para la comunicación entre el Agente y la NMS.

- La facilidad de programación de las interfaces visuales (ventanas del programa, botones, etc), para el manejo del usuario.

- La sintaxis de C# es bastante semejante a la de JAVA sobre todo en la utilización de estructuras de control, construcción de objetos, tipos, propiedades, métodos o eventos.

- Se utilizará como herramienta de programación el Paquete Visual Studio .NET, para el desarrollo de la aplicación para PDA.

2.3.2. COMUNICACIÓN

Para la comunicación del software (NMS para PDA) con el Agente del dispositivo gestionado, se ha basado en la utilización de “sockets”, permitiendo la comunicación entre el programa cliente y un programa del servidor en una red.

La utilización de los sockets nos permite establecer una interfaz con la capa de transporte (nivel 4) de la jerarquía OSI, con el fin de encapsular dentro de cada una de ellas, detalles específicos de cada capa.

El protocolo y operaciones SNMP se realizarán en la capa Aplicación del modelo OSI, donde se especificarán las diferentes operaciones SNMP (get, set, get next, trap, etc), las cuales van hacer encapsuladas en la capa Transporte, donde se especifica su puerto y tipo de conexión (UDP, 161), para finalmente indicar la dirección IP del Agente del dispositivo monitorizado, en la capa Red.

Para la utilización de los sockets en el desarrollo del programa se ha utilizado la librería ***System.Socket*** de Visual Studio 2005.

Los paquetes SNMP que se envían y reciben en el programa que se ejecuta en la PDA, se maneja mediante arreglos de bytes, donde cada una de las clases implicadas con el protocolo SNMP codificarán y decodificarán los arreglos de bytes.

2.3.3 IMPLEMENTACIÓN

Esta sección indica el desarrollo del programa, donde se explica el propósito de las clases, formularios y sus principales métodos.

2.3.3.1 Clase CBER

El propósito de la clase CBER es crear los tipos de valores que contiene el paquete SNMP.

La definición de esta clase CBER tiene atributos propios de acuerdo a la codificación BER, cuyos atributos principales son el tipo, longitud y valor.

```
private string tipo;
private int longitud;
private string str_valor;
private int int_valor;
private Int64 int_valor64;
```

Espacio de Código 2.1 Atributos de la clase CBER

2.3.3.2 Clase CCONTER, CCOUNTER32, CCOUNTER64, CGAUGE, CINT, CIPADDRESS, CNULL, COID, CSEQUENCE, CSTRING, CTIMETICKS

Todas estas clases tienen como clase base a CBER, que permite crear los diferentes tipos de valores de acuerdo a la codificación BER. Cada una de estas clases permiten trabajar con el protocolo SNMP versión uno y 2c. La implementación de cada una de estas clases es similar. Para ejemplo tomaremos a la clase CIPADDRESS.

La clase CIPADDRESS tiene como un atributo adicional un arreglo de bytes donde el valor de la dirección IP será codificada.

```
public CIPADDRESS(string val_valor)

  {

      base.Set_Tipo("IPADDRESS");
      base.Set_Valor(val_valor);

      dato = new byte[1024];

      byte[] IP = new byte[1024];
      string[] IPvals = val_valor.Split('.');
      int IPlen = IPvals.Length;
      int IPlen1 = IPlen;
      int cnt = 0, temp, i;


      for (i = 0; i < IPlen1; i++)
      {
          temp = Convert.ToInt16(IPvals[i]);

          IP[cnt] = Convert.ToByte(temp);
          cnt++;

      }

      base.Set_Longitud(IPlen);

      dato[0] = 0x40;
      dato[1] = 0x04;

      for (i = 0; i < IPlen; i++)
          dato[i + 2] = Convert.ToByte(IP[i]);
      }
```

Espacio de Código 2.2 Codificación de una dirección IP

Por ejemplo para codificar una dirección IP el campo valor es igual a 40H con una longitud de 4 bytes.

La clase CIPADDRESS tiene un constructor que permite decodificar una dirección IP codificada de acuerdo a BER.

```
public CIPADDRESS(byte [] paquete)
  {

      base.Set_Tipo("IPADDRESS");
      base.Set_Longitud(4);

      int i;

      string valor;
      valor = "";

      for (i = 0; i < 4; i++)
      {
          if (i < 3)
          {
            valor = valor + Convert.ToString(paquete[i + 2]) + ".";
          }

          if (i == 3)
          {
              valor = valor + Convert.ToString(paquete[i + 2]);
          }

      }

      base.Set_Valor(valor);
      dato = new byte[1024];
      dato = paquete;

  }
```

Espacio de Código 2.3 Decodificación de una dirección IP

2.3.3.3 Clase CPDU

La definición de esta clase es CPDU que es una clase pública. El propósito de CPDU es construir las diferentes PDU SNMP tales como GET, GETNEXT, SET y TRAP, para luego ser encapsulados mediante la clase CPaquete que analizaremos posteriormente.

La definición de esta clase CPDU tiene atributos propios de una PDU SNMP, que permiten identificar y obtener información de una PDU determinada. A continuación se observa los atributos de esta clase.

```
private int Tipo;
private int Request_ID;
private int Error_Estatus;
private int Error_Index;
private int Generic_Type;
private int Specific_Type;
private long Time_Tick;
private CSEQUENCE varbind;
private byte[] packet;
private int longitud;

private COID Enterprice;
private CIPADDRESS AgentAddress;
private CTIMETICKS TimeStam;
```

Espacio de Código 2.4 Atributos de la clase CPDU

2.3.3.3.1 Constructores de la Clase CPDU

Esta clase posee varios constructores sobrecargados entre los cuales se encuentra el constructor por defecto que inicializa cada uno de los atributos de esta clase. A continuación se detallará cada uno de los constructores y sus funcionalidades.

Uno de los constructores para codificar una PDU SNMP tipo GET, GETNEXT y SET, es el mostrado en el espacio de código 2.5, el cual requiere como argumentos el tipo de PDU, índices y estados de error y el atributo Varbind de tipo CSEQUENSE.

```
public CPDU(int tipo, int req_id, int err_est, int err_ind,
CSEQUENCE dvarbind)
  {

      packet = new byte[1024];

      this.Set_Tipo(tipo);
      this.Set_RequestID(req_id);
      this.Set_ErrorEstatus(err_est);
      this.Set_ErrorIndex(err_ind);
```

```
        this.Set_VarBind(dvarbind);

        packet[15] = Convert.ToByte(dvarbind.Get_Longitud()+2);
        packet[14] = 0x30;
        packet[1] = Convert.ToByte(dvarbind.Get_Longitud() + 16);
        longitud = Convert.ToByte(dvarbind.Get_Longitud() + 16);
}
```

Espacio de Código 2.5 Constructor clase CPDU

Esta clase además tiene un constructor que permite asignar valores a los atributos de la clase cuya información es recibida en bytes, para las PDU SNMP tipo GET, GETNEXT, SET Y TRAP.

Este constructor recibe PDU SNMP en forma de bytes. El constructor verifica que tipo de PDU va ha codificar de acuerdo al parámetro tipo; por ejemplo para las PDU RESPONSE, GET, GETNEXT y TRAP, el valor de tipo es 162, 160, 161, 164 167 respectivamente. La codificación de cada uno las PDU SNMP mencionadas son diferentes, tal como se observa en el espacio de código anterior.

2.3.3.3.2 Decodificación de las PDU GET, GETNEXT y SET

La decodificación de las PDU GET, GETNEXT y SET está basado en las reglas de codificación básica BER. La decodificación para este tipo de PDU es la misma ya que posee los mismos campos y tamaños, especialmente en los campos tipo, request id, error status y error id. La decodificación de estas PDU es compatible con el protocolo SNMP de la versión V1 y V2c.

```
if (Convert.ToInt16(this.packet[0]) != 162)
      {

          int tipo = Convert.ToInt16(this.packet[0]) - 160;
          this.Set_Tipo(tipo);
          this.longitud = Convert.ToInt16(this.packet[1]);

          this.Request_ID = Convert.ToInt16(this.packet[7]);
          this.Error_Estatus = Convert.ToInt16(this.packet[10]);
          this.Request_ID = Convert.ToInt16(this.packet[13]);


           byte[] BVARBIND = new byte[1024];
           int i = 0;
```

```
            for (i = 0; i < Convert.ToInt16(this.packet[17]) + 2;
            i++)
            {
                BVARBIND[i] = this.packet[16 + i];
            }

            this.varbind = new CSEQUENCE(BVARBIND);

        }
```

Espacio de Código 2.6 Decodificación de las PDU GET, GETNEXT y SET

Como se observa en el espacio de código 2.6 existe una condicionante de que la decodificación de la PDU Response con un valor de tipo 162, es diferente a las PDU tipo Get, GetNext y Set.

2.3.3.3.3 Decodificación de la PDU RESPONSE

La decodificación de la PDU RESPONSE está basada en las reglas de codificación básica BER. La decodificación de esta PDU es diferente a las PDU del punto anterior, especialmente en los campos Request ID, Error Status y Error ID, teniendo 3 bits menos que las PDU de tipo GET, GETNEXT y SET. La decodificación de estas PDU es compatible con el protocolo SNMP de la versión V1 y V2c.

```
if (Convert.ToInt16(this.packet[0]) == 162)
      {
          int tipo = Convert.ToInt16(this.packet[0]) - 160;
          this.Set_Tipo(tipo);

          int long_temp = Convert.ToInt16(this.packet[1]);
          int index =0;

          if (long_temp > 127)
          {
              long_temp =  Convert.ToInt16(this.packet[2]);
              index = 1;
              this.longitud = long_temp;
          }

          else
          {
              this.longitud = Convert.ToInt16(this.packet[1]);
          }

          this.Request_ID = Convert.ToInt16(this.packet[4+index]);
```

```
            this.Error_Estatus =
            Convert.ToInt16(this.packet[7+index]);
          this.Request_ID = Convert.ToInt16(this.packet[10+index]);

          byte [] BVARBIND = new byte[1024];
          int i = 0;

          int long_var = Convert.ToInt16(this.packet[12 + index]);
          int index_var = 0;

          if (long_var > 127)
          {
              long_var =  Convert.ToInt16(this.packet[13 + index]);
              index_var = 1;
          }
          else {

              long_var = Convert.ToInt16(this.packet[12 + index]);
          }


          for (i = 0; i <
          Convert.ToInt16(this.packet[14+index+index_var])+2; i++)
          {
              BVARBIND[i] = this.packet[13 + i+index+index_var];
          }

          this.varbind = new CSEQUENCE(BVARBIND);
      }
```

Espacio de Código 2.7 Decodificación de las PDU RESPONSE

2.3.3.3.4 Decodificación de las PDU TRAP del protocolo SNMP V1 y V2c

La decodificación de la PDU TRAP del protocolo SNMP de la versión 1, está basado en las reglas de codificación básica BER y es diferente a la PDU TRAP del protocolo SNMP de la versión 2. La diferencia radica en que estas PDU tienen diferentes campos.

```
if (Convert.ToInt16(this.packet[0]) == 164)
      {

          int tipo = Convert.ToInt16(this.packet[0]) - 160;
          this.Set_Tipo(tipo);

          int long_temp = Convert.ToInt16(this.packet[1]);
          int index = 0;

          if (long_temp > 127)
          {
              long_temp = Convert.ToInt16(this.packet[2]);
              index = 1;
              this.longitud = long_temp;
```

```
            }

            else
            {
                this.longitud = Convert.ToInt16(this.packet[1]);
            }


            int i = 0;
            byte[] BENTERPRICE = new byte[1024];
            int long_enterprice = this.packet[3 + index];

            for (i = 0; i < long_enterprice + 2; i++)
            {
                BENTERPRICE[i] = this.packet[i + 2 + index];
            }

            this.Enterprice = new COID(BENTERPRICE);


            byte[] BAGENTADDRESS = new byte[1024];
              int long_agentaddress = this.packet[5 + index +
              long_enterprice];

            for (i = 0; i < long_agentaddress + 2; i++)
            {
              BAGENTADDRESS[i] = this.packet[4 + i + index +
              long_enterprice];
            }

            this.AgentAddress = new CIPADDRESS(BAGENTADDRESS);

            this.Generic_Type = this.packet[8 +  index +
            long_enterprice + long_agentaddress];

            this.Specific_Type = this.packet[11 + index +
            long_enterprice + long_agentaddress];


            byte[] BTIMESTAM = new byte[1024];
              int long_timestam = this.packet[13 + index +
              long_enterprice + long_agentaddress];

            for (i = 0; i < long_timestam + 2; i++)
            {
              BTIMESTAM[i] = this.packet[12 + i + index +
              long_enterprice + long_agentaddress];
            }

            this.TimeStam = new CTIMETICKS(BTIMESTAM);

            byte[] BVARBIND = new byte[1024];
              int long_var = this.packet[17 + i + index +
              long_enterprice + long_agentaddress + long_timestam];

            for (i = 0; i < long_var + 2; i++)
            {
              BVARBIND[i] = this.packet[16 + i + index +
              long_enterprice + long_agentaddress + long_timestam];
            }
```

```
            this.varbind = new CSEQUENCE(BVARBIND);

        }
}
```

Espacio de Código 2.8 Decodificación de la PDU TRAP del protocolo SNMP versión 1

La PDU TRAP del protocolo SNMP versión 2c tiene campos similares a las PDU GET, GETNEXT, SET y RESPONSE, cuyo valor del campo tipo es 167 y la decodificación igualmente se basa en las reglas básicas de codificación BER de acuerdo a ASN 1.

```
if (Convert.ToInt16(this.packet[0]) == 167)
      {



            int tipo = Convert.ToInt16(this.packet[0]) - 160;
            this.Set_Tipo(tipo);

            int long_temp = Convert.ToInt16(this.packet[1]);
            int index = 0;

            if (long_temp > 127)

            {
                long_temp = Convert.ToInt16(this.packet[2]);
                index = 1;
                this.longitud = long_temp;
            }


            else
            {
                this.longitud = Convert.ToInt16(this.packet[1]);
            }

            this.Request_ID = Convert.ToInt16(this.packet[4 +
            index]);
            this.Error_Estatus = Convert.ToInt16(this.packet[7 +
            index]);
            this.Request_ID = Convert.ToInt16(this.packet[10 +
            index]);

            byte[] BVARBIND = new byte[1024];
            int i = 0;

            int long_var = Convert.ToInt16(this.packet[12 + index]);
            int index_var = 0;
```

```
            if (long_var > 127)
            {
                long_var = Convert.ToInt16(this.packet[13 + index]);
                index_var = 1;
            }
            else
            {

                long_var = Convert.ToInt16(this.packet[12 + index]);
            }


            for (i = 0; i < Convert.ToInt16(this.packet[14 + index +
            index_var]) + 2; i++)

            {
              BVARBIND[i] = this.packet[13 + i + index + index_var];
            }

            this.varbind = new CSEQUENCE(BVARBIND);

        }
```

Espacio de Código 2.9 Decodificación de la PDU TRAP del protocolo SNMP versión 2c

2.3.3.4 Clase CGET, CGETNEXT, CSET Y CTRAP

El propósito de las clases CGET, CGETNEXT, CSET Y CTRAP es construir las diferentes PDU SNMP tales como GET, GETNEXT, SET y TRAP respectivamente de una manera más simplificada utilizando como clase base a CPDU, anteriormente analizada.

Estas clases no tienen atributos propios, sino que los hereda de la clase base CPDU. A continuación la definición de cada una de estas clases.

```
public class CGet : CPDU
    {
        public CGet(int num_sec, CSEQUENCE dvarbind)
            : base(0, num_sec, 0, 0, dvarbind)
        {

        }

    }
```

```
    public class CSet : CPDU
    {
        public CSet(int num_sec, CSEQUENCE dvarbind)
            : base(3, num_sec, 0, 0, dvarbind)
        {

        }

    }

    public class CResponse : CPDU
    {
        public CResponse(byte [] pdu_response):base(pdu_response)
        {

        }

    }

    public class CGetNext : CPDU
    {
        public CGetNext(int num_sec, CSEQUENCE dvarbind)
            : base(1, num_sec, 0, 0, dvarbind)
        {

        }

    }

    public class CTrap: CPDU
    {
       public CTrap(byte[] pdu_response)
            : base(pdu_response)
        {

        }
}
```

Espacio de Código 2.10 Atributos de las clases CGET, CGETNEXT, CSET, y CTRAP

2.3.3.4.1. Constructores de las clases CGET, CGETNEXT, CSET, CRESPONSE y CTRAP

Cada uno de los constructores de estas clases utiliza el constructor de la clase base para establecer el valor de los atributos que heredan de CPDU. La implementación de estos constructores se observa en el espacio de código 2.10.

2.3.3.5 Clase CPaquete

El propósito de la clase CPaquete es construir un paquete SNMP encapsulando a las distintas PDU tipo GET, GETNEXT, SET y TRAP, añadiendo la versión y comunidad correspondientes al protocolo SNMP.

La definición de la clase CPaquete tiene atributos que permiten identificar y obtener información de la versión y comunidad correspondiente al protocolo SNMP. A continuación se observa los atributos de esta clase.

```
private string Comunidad;
private int Version;
private byte[] paquete;
private CPDU PDU;
private int pkt_longitud;
```

Espacio de Código 2.11 Atributos de la clase CPaquete

2.3.3.5.1. Constructores de la Clase CPaquete

Esta clase posee varios constructores sobrecargados entre los cuales se encuentra el constructor por defecto que inicializa cada uno de los atributos de esta clase. A continuación se detallará cada uno de los constructores y sus funcionalidades.

Esta clase tiene constructores que toman como un atributo objetos tipo CGET, CGETNEXT, CSET, y CRESPONSE para encapsular PDU SNMP tipo GET, GETNEXT, SET y RESPONSE respectivamente. En el espacio de código 2.12 se observa lo mencionado en este párrafo.

```
public CPaquete(int ver, string Comuni, CGet GET_PDU)
  {
      this.Comunidad = Comuni;
      this.paquete = new byte[1024];

      this.paquete[0] = 0x30;
      this.Set_Version(ver);
      this.Set_Comunidad(Comuni);
      this.Set_PDU(GET_PDU);
```

```
        int longitud = 0;
        longitud = Convert.ToInt16(paquete[6]) + 7 +
        GET_PDU.Get_Longitud();
        paquete[1] = Convert.ToByte(longitud);
        this.pkt_longitud = longitud + 2;
    }

    public CPaquete(int ver, string Comuni, CGetNext GETNEXT_PDU)
    {
        this.Comunidad = Comuni;
        this.paquete = new byte[1024];

        this.paquete[0] = 0x30;
        this.Set_Version(ver);
        this.Set_Comunidad(Comuni);
        this.Set_PDU(GETNEXT_PDU);

        int longitud = 0;
        longitud = Convert.ToInt16(paquete[6]) + 7 +
        GETNEXT_PDU.Get_Longitud();
        paquete[1] = Convert.ToByte(longitud);
        this.pkt_longitud = longitud + 2;
    }

    public CPaquete(int ver, string Comuni, CSet SET_PDU)
    {
        this.Comunidad = Comuni;
        this.paquete = new byte[1024];

        this.paquete[0] = 0x30;
        this.Set_Version(ver);
        this.Set_Comunidad(Comuni);
        this.Set_PDU(SET_PDU);

        int longitud = 0;
        longitud = Convert.ToInt16(paquete[6]) + 7 +
        SET_PDU.Get_Longitud();
        paquete[1] = Convert.ToByte(longitud);
        this.pkt_longitud = longitud + 2;
    }

    public CPaquete(int ver, string Comuni, CResponse RESPONSE_PDU)
    {
        this.Comunidad = Comuni;
        this.paquete = new byte[1024];

        this.paquete[0] = 0x30;
        this.Set_Version(ver);
        this.Set_Comunidad(Comuni);
        this.Set_PDU(RESPONSE_PDU);

        int longitud = 0;
        longitud = Convert.ToInt16(paquete[6]) + 5 +
        RESPONSE_PDU.Get_Longitud();
        paquete[1] = Convert.ToByte(longitud);
        this.pkt_longitud = longitud + 2;
    }
```

Espacio de Código 2.12 Constructores de la clase CPaquete

Esta clase además tiene un constructor que permite asignar valores a los atributos de la clase cuyos paquetes son recibidos en bytes por parte del agente SNMP.

```
public CPaquete(byte[] datos)

  {
      this.paquete = new byte[1024];
      this.paquete = datos;

      int long_paquete = Convert.ToInt16(this.paquete[1]);
      int index=0;

      if (long_paquete > 127)
      {
          long_paquete = Convert.ToInt16(this.paquete[2]);
          index = 1;
      }
      else
      {
          long_paquete = Convert.ToInt16(this.paquete[1]);
      }

      this.Version = Convert.ToInt16(datos[4+index]);

      byte[] BCOMUNIDAD = new byte[1024];
      int i = 0;

      for (i = 0; i < Convert.ToInt16(this.paquete[6+index]) + 2;
      i++)
      {
          BCOMUNIDAD[i] = this.paquete[i + 5+index];
      }

      CSTRING str_comunidad = new CSTRING(BCOMUNIDAD);
      this.Comunidad = str_comunidad.Get_Valor();

      byte[] BPDU = new byte[1024];
      for (i = 0; i < long_paquete - 5 + index; i++)
      {
      BPDU[i] = this.paquete[i +
      Convert.ToInt16(this.paquete[6+index]) + 7 + index];
      }
      this.PDU = new CPDU(BPDU);

      this.pkt_longitud = long_paquete + 2;
      }
```

Espacio de Código 2.13 Constructor que permite decodificar los bytes recibidos

2.3.3.6 Clase snmpsesion

El propósito de la clase snmpsesion es establecer una conexión y enviar paquetes SNMP al agente respectivo del equipo monitorizado. La conexión SNMP se realiza mediante el uso de Sockets compuesto por la dirección IP y el puerto respectivo 161. Para enviar un paquete SNMP, se utilizarán funciones propias de la clase Socket enviando estos paquetes como bytes de información.

La definición de la clase snmpsesion tiene atributos para la comunicación con el agente SNMP como objetos tipo Socket, IPEndPoint y EndPoint tal como se observa en el espacio de código 2.14.

```
private Socket sock;
private string host;
private int Puerto;
private IPEndPoint iep;
private EndPoint ep;
private byte[] paquete;
```

Espacio de Código 2.14 Atributos de la clase snmpsesion

2.3.3.6.1. Constructores de la clase snmpsesion

Esta clase tiene un constructor que permite establecer la conexión con el dispositivo monitorizado mediante el uso de sockets. Se crea un objeto tipo Socket con una familia de dirección Internetwork, tipo de socket y protocolo datagrama y UDP respectivamente, ya que SNMP basa su comunicación en el protocolo SNMP. A continuación, en el espacio de código 2.15 se observa lo anteriormente mencionado en este párrafo.

```
public snmpsesion(string host1, int Puerto1)
  {
      host = host1;
      Puerto = Puerto1;

      sock = new Socket(AddressFamily.InterNetwork,
      SocketType.Dgram,ProtocolType.Udp);

      IPHostEntry ihe = Dns.GetHostEntry(host);
```

```
        iep = new IPEndPoint(ihe.AddressList[0], Puerto);
        ep = (EndPoint)iep;
        paquete = new byte[1024];
}
```

Espacio de Código 2.15 Constructor clase snmpsesion

2.3.3.6.2. Métodos de la clase snmpsesion

El método más importante de esta clase es la de enviar y recibir paquetes SNMP, esta función ya establecida una conexión enviará paquetes SNMP en forma de bytes al agente respectivo, para esta operación se hace uso de la función sentTo de la clase Socket. La función sentTo requiere como argumentos el punto destinatario, el paquete SNMP en forma de bytes y su respectiva longitud, tal como se observa en el espacio de código 2.16.

```
public void enviar_paquete(CPaquete packet)
        {
            this.sock.SendTo(packet.Get_Paquete(),
            packet.Get_Pkt_Longitud(), SocketFlags.None, iep);
            paquete = new byte[1024];

            try
            {
                  int recv = this.sock.ReceiveFrom(paquete, ref
                  this.ep);
            }
            catch (SocketException)
            {
                  paquete[0] = 0xff;
            }
        }
```

Espacio de Código 2.16 Método para enviar y recibir paquetes snmpsesion

Como se observa en el espacio de código anterior, una ves enviado el paquete SNMP, se espera la respuesta del agente SNMP mediante el uso de ReceiveFrom de la clase Socket, el cual será decodificado mediante la clase CPaquete para obtener su información.

2.3.3.7 Clase snmpsesiontrap

El propósito de la clase snmpsesiontrap es establecer una conexión y recibir paquetes SNMP tipo TRAP del agente respectivo del equipo monitorizado. La conexión SNMP se realiza mediante el uso de Sockets compuesto por la dirección IP y el puerto respectivo 162.

La definición de la clase snmpsesiontrap tiene atributos para la comunicación con el agente SNMP como objetos tipo Socket, IPEndPoint y EndPoint tal como se observa en el espacio de código 2.17.

```
private Socket sock;
private string host;
private int Puerto;
private IPEndPoint iep;
private EndPoint ep;
private CPaquete PduTrap;
private CINT valor;
InfoTrap info;
private byte[] paquete;
```

Espacio de Código 2.17 Atributos de la clase snmpsesiontrap

2.3.3.7.1. Constructores de la clase snmpsesiontrap

Esta clase tiene un constructor que permite establecer la conexión con el dispositivo monitorizado mediante el uso de sockets. Se crea un objeto tipo Socket con una familia de dirección Internetwork, tipo de socket y protocolo datagrama y UDP respectivamente, ya que SNMP basa su comunicación en el protocolo SNMP. A continuación, en espacio de código 2.18 se observa lo anteriormente mencionado en este párrafo.

```
public snmpsesiontrap(string host1)
  {
      host = host1;
      Puerto = 162;

      sock = new Socket(AddressFamily.InterNetwork,
     SocketType.Dgram,ProtocolType.Udp);

      IPHostEntry ihe = Dns.GetHostEntry(host);
```

```
        iep = new IPEndPoint(ihe.AddressList[0], Puerto);
        ep = (EndPoint)iep;
        paquete = new byte[1024];
        j = 0;

        this.valor = valor;
}
```

Espacio de Código 2.18 Constructor clase snmpsesiontrap

2.3.3.7.2. Métodos de la clase snmpsesiontrap

El método más importante de esta clase es la de recibir paquetes SNMP tipo TRAP, esta función ya establecida una conexión empezará a recibir paquetes en forma de bytes por parte del agente respectivo, para esta operación se hace uso de la función captura() del espacio de código 2.19. La función captura hace uso de Threads, que permite que la función recibir_paquete() del espacio de código 2.20 se ejecute con el programa principal.

```
public void recibir_paquete()
  {
      sock.Bind(iep);
      j = 0;

      while (true)
      {
          paquete = new byte[1024];
          try
          {

              j = 0;
             int bytesReceived = sock.ReceiveFrom(this.paquete,
            ref this.ep);
              CPaquete PduTrap1 = new CPaquete(this.paquete);
              j = 1;
              valor.Set_Valor1(j);
              this.info(PduTrap1);

              //this.paquete = get_datos();
          }
          catch (SocketException)
          {
              paquete[0] = 0xff;
          }
      }
}
```

Espacio de Código 2.19 Método para recibir paquetes SNMP tipo TRAP

```
        public void captura()
                {
                    Thread thread1 = new Thread(new
                    ThreadStart(recibir_paquete));
                    thread1.Start();
                }
```

Espacio de Código 2.20 Método para capturar paquetes SNMP

2.3.3.8 Clase CDispositivo

El propósito de la clase CDispositivo es crear un objeto cuyos atributos sean propios de un dispositivo monitorizado.

La definición de la clase CDispositivo tiene atributos como nombre, dirección IP, tipo de dispositivo, comunidad de lectura y escritura tal como se observa en el espacio de código 2.21.

```
                    private CIPADDRESS DirIP;
                    private string Nombre;
                    private string Grupo;
                    private string Tipo;

                    private CSTRING RComunidad;
                    private CSTRING WComunidad;
                    private snmpsesion session;
                    private int polling;
                    private string estado;
```

Espacio de Código 2.21 Atributos de la clase CDispositivo

2.3.3.8.1. Constructores de la clase CDispositivo

Esta clase tiene un constructor que permite crear un objeto de tipo CDispositivo, estableciendo el valor respectivo de cada uno de los atributos de esta clase.

```
public CDispositivo(CIPADDRESS DirIp1, string Nombre1, string
Grupo1, string Tipo1, CSTRING RComunity1, CSTRING WComunity1, int
polling1)
  {

      Set_DirIP(DirIp1);
      Set_Nombre(Nombre1);
      Set_Grupo(Grupo1);
      Set_Tipo(Tipo1);
      Set_RCommunity(RComunity1);
      Set_WCommunity(WComunity1);
      Set_Polling(polling1);

  }
```

Espacio de Código 2.22 Constructor clase CDispositivo

2.3.3.8.2. Métodos de la clase CDispositivo

Los métodos más importantes de esta clase son la de codificar y decodificar información del dispositivo, que permite guardar y descargar información desde el archivo que almacena los dispositivos gestionados. La codificación y decodificación de la información de un dispositivo se observa en el espacio de código 2.23 y 2.24 respectivamente.

```
public string Codificacion()
  {
      string codificar = this.Get_DirIP().Get_Valor() + "," +
      this.Get_Nombre() + "," + this.Get_Grupo() + "," +
      this.Get_Tipo() + "," + this.Get_RCommunity().Get_Valor() +
      "," + this.Get_WCommunity().Get_Valor()+ ",";

      return codificar;
  }
```

Espacio de Código 2.23 Codificación de los dispositivos a guardar

```
public void Decodificacion(string str_equipo)
  {
      int tamano = str_equipo.Length;
      string parametro = "";

      int j = 0;

      CSTRING rcommunity = new CSTRING();
      CSTRING wcommunity = new CSTRING();

      for (int i = 0; i < tamano; i++)
```

```
        {

            if (Convert.ToString(str_equipo[i]) != ",")
            {
                    parametro = parametro +
                    Convert.ToString(str_equipo[i]);
            }
            if (Convert.ToString(str_equipo[i]) == ",")
            {
                if (j == 0)
                {
                    this.DirIP = new CIPADDRESS(parametro);
                    this.session = new snmpsesion(parametro, 161);
                    PingIPv4 ping = new PingIPv4();
                    ping.RequestSize = 32;
                    ping.RequestTimedOut += new
                    PingIPv4.PingTimeOutDelegate(icmp_RequestTimedOut
                    );
                    ping.ReplyReceived += new
                    PingIPv4.PingReplyDelegate(icmp_ReplyReceived);
                    ping.PingError += new
                    PingIPv4.PingErrorDelegate(icmp_PingError);
                    ping.Ping(parametro, 2);
                    parametro = "";

                }

                if (j == 1)
                {
                    this.Set_Nombre(parametro);
                    parametro = "";

                }

                if (j == 2)
                {
                    this.Set_Grupo(parametro);
                    parametro = "";

                }


                if (j == 3)
                {
                    this.Set_Tipo(parametro);
                    parametro = "";

                }

                if (j == 4)
                {

                    rcommunity = new CSTRING(parametro);
                    this.Set_RCommunity(rcommunity);
                    parametro = "";

                }

                if (j == 5)
                {
```

```
                wcommunity = new CSTRING(parametro);
                this.Set_WCommunity(wcommunity);
                parametro = "";

            }
            j = j + 1;

        }

    }

    this.Set_Polling(30);
   }
```

Espacio de Código 2.24 Decodificación de los dispositivos guardados

2.3.3.9 Clase CPing

Para la implementación de esta clase se hizo uso de la librería `iphlpapi.dll`, que es un módulo que contiene las funciones usadas por el Windows IP Helper API. La librería `iphlpapi.dll` permite enviar paquetes ICMP tipo echo hacia el equipo monitorizado, en el espacio de código 2.25 se observa la utilización de esta librería.

```
[DllImport("iphlpapi.dll", SetLastError = true)]
internal static extern IntPtr IcmpCreateFile();

[DllImport("iphlpapi.dll", SetLastError = true)]
internal static extern bool IcmpCloseHandle(IntPtr Handle);

[DllImport("iphlpapi.dll", SetLastError = true)]
internal static extern int IcmpSendEcho(
    IntPtr IcmpHandle,
    Int32 DestinationAddress,
    byte[] RequestData,
    Int16 RequestSize,
    IntPtr RequestOptions,
    byte[] ReplyBuffer,
    Int32 ReplySize,
    Int32 Timeout);
```

Espacio de Código 2.25 Uso de la librería iphlpapi.dll

2.3.3.9.1. Delegados de la clase CPing

Los delegados permiten pasar los métodos como parámetros, es decir cuando se asigna un método a un delegado, éste se comporta exactamente como el método.

La clase Cping tiene tres delegados que capturan eventos relacionados a una respuesta de un paquete ICMP tipo echo request, estos se los puede visualizar en el espacio de código 2.26.

```
public delegate void PingReplyDelegate(object sender,
System.Net.IPAddress ReplyIPAddress, int Bytes, int Time, int TTL);
  public event PingReplyDelegate ReplyReceived;

public delegate void PingTimeOutDelegate(object sender,
System.Net.IPAddress ReplyIPAddress, int Bytes);

public event PingTimeOutDelegate RequestTimedOut;

public delegate void PingErrorDelegate(object Sender, string
ErrorMessage);
public event PingErrorDelegate PingError;
```

Espacio de Código 2.26 Delegados de la clase CPing

Estos delegados permitirán obtener la respuesta de un paquete ICMP tipo echo request por parte de los métodos de otras clases o formularios. Este caso se observa en el espacio de código 2.36 de este capítulo.

2.3.3.10 Formulario form_principal

El propósito de este formulario es crear una interfaz gráfica donde el usuario tiene el listado de equipos y herramientas para monitorizarlos.

El formulario está compuesto por los siguientes elementos que son parte del formulario:

- Un objeto de tipo mainMenu donde están los menús principales Archivo, Edición y Herramientas.

- Un objeto de tipo datagrid donde se listan todos los equipos que son monitoreados.

En la figura 2.31 se observa cada uno de los componentes de este formulario.

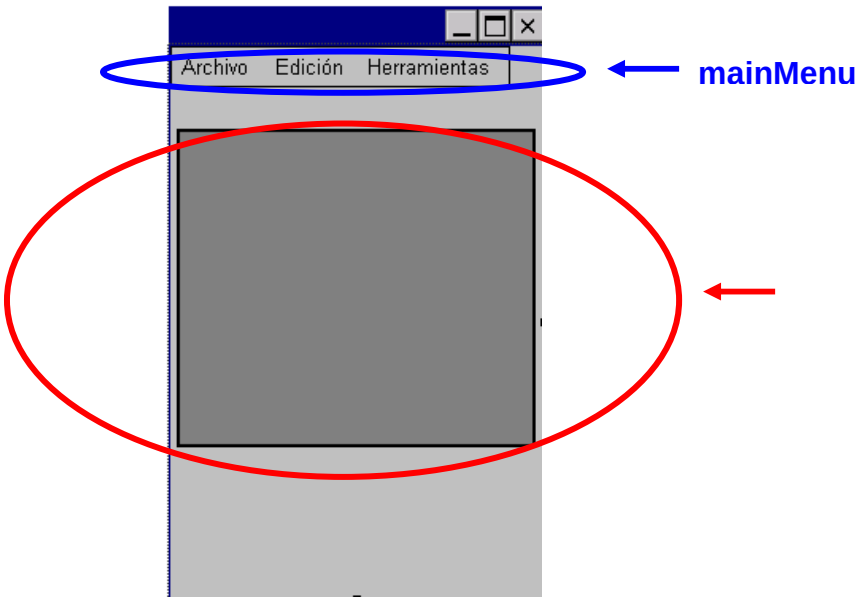


Figura 2.31 Formulario form_principal

2.3.3.10.1. Nuevo Dispositivo

Este método de la clase form_principal permite llamar al formulario form_nuevo_edit, donde se ingresa los atributos correspondientes al nuevo dispositivo.

```
private void nuevo_Click(object sender, EventArgs e)
  {

      form_nuevo_edit form_nuevo = new form_nuevo_edit();
      form_nuevo.set_equipos(this.Equipos);
      Equipos delegadoequipo = new Equipos(set_equipos);
      form_nuevo.delegado_equipos(delegadoequipo);
      form_nuevo.set_tipo("NUEVO");
      form_nuevo.Show();

  }
```

Espacio de Código 2.27 Método para llamar al formulario form_nuevo_edit

2.3.3.10.2. Guardar

Este método de la clase form_principal permite guardar los dispositivos monitoreados en un archivo de texto y en una nueva sesión cargar los dispositivos que se encuentran en el archivo en el programa.

```
private void menuItem3_Click(object sender, EventArgs e)
        {
            update_dispositivos();
         }
```

Espacio de Código 2.28 Método para llamar a la función para guardar dispositivos

2.3.3.10.3. Editar

El método permite editar ciertos parámetros de un dispositivo monitorizado que fue ingresado en el método Nuevo de esta clase, por ejemplo el nombre de las comunidades y tipo de dispositivo.

```
private void editar_Click(object sender, EventArgs e)
  {

      if (this.Equipos.Count == 0)
      {
          MessageBox.Show("NO EXISTE EQUIPOS");
      }

      if (this.Equipos.Count > 0)
      {

          form_nuevo_edit form_edit = new form_nuevo_edit();
          form_edit.set_equipos(this.Equipos);
          delegado_equipo = new Equipos(set_equipos);

          form_edit.delegado_equipos(delegado_equipo);
          form_edit.set_indice(fila);
          form_edit.set_tipo("EDIT");
          form_edit.Show();
      }

}
```

Espacio de Código 2.29 Método para editar un dispositivo monitorizado

En el espacio de código anterior se observa que el método crea un objeto de tipo form_nuevo_edit, se pasa como argumentos un objeto de tipo CDispositivo y el tipo de edición “EDIT”.

2.3.3.10.4. Método SNMP

El método SNMP permite abrir el formulario form_snmp para realizar las diferentes operaciones SNMP como GET, GETNEXT y SET. Este método verifica si el estado de dispositivo es “CONECTADO”, para pasar como argumentos a objetos de tipo CDispositvo y snmpsesion al formulario form_snmp.

```
private void menuItem8_Click(object sender, EventArgs e)
  {
      if (this.Equipos.Count > 0)
      {
            CDispositivo Dispositivo_temp =
            (CDispositivo)this.Equipos[fila];
          if (Dispositivo_temp.Get_Estado() == "CONECTADO")
          {
                  form_snmp formulario_snmp = new form_snmp();

                  formulario_snmp.set_session(Dispositivo_temp.Get_
                  Session());
                  formulario_snmp.set_session(Dispositivo_temp);
                  formulario_snmp.Show();
          }
          else
          {
              MessageBox.Show("EQUIPO ESTA DESCONECTADO");

          }
      }

      if (this.Equipos.Count == 0)
      {
          MessageBox.Show("NO EXISTE EQUIPOS");
      }

  }
```

Espacio de Código 2.30 Método para abrir el formulario de operaciones SNMP

2.3.3.10.5. Método PING

El método PING permite abrir el formulario form_ping para comprobar la conectividad del equipo monitorizado. Este método verifica si el estado de

dispositivo es “CONECTADO”, para pasar como argumento a un objetos de tipo CDispositvo al formulario form_ping.

```
private void menuItem9_Click(object sender, EventArgs e)
  {

      if (this.Equipos.Count > 0)
      {

            CDispositivo Dispositivo_temp =
            (CDispositivo)Equipos[fila];

            form_ping formulario_ping = new form_ping();
            formulario_ping.set_equipo(Dispositivo_temp);
            formulario_ping.Show();
      }

      if (this.Equipos.Count == 0)
      {
          MessageBox.Show("NO EXISTE EQUIPOS");
      }

}
```

Espacio de Código 2.31 Método para abrir el formulario form_ping

2.3.3.10.6. Método Browser

El método Browser permite configurar algunos dispositivos remotamente a través de un Browser. Este método crea un objeto de tipo Proccess con el fin de abrir el dispositivo gestionado utilizando como URL la dirección IP de equipo que va hacer configurado vía Browser.

```
private void menuItem11_Click(object sender, EventArgs e)
  {
      if (this.Equipos.Count > 0)
      {
            string URL_Equipo;
            CDispositivo Dispositivo_Temp =
            (CDispositivo)this.Equipos[fila];

          System.Diagnostics.Process proc = new
          System.Diagnostics.Process();
          proc.StartInfo.FileName = "iexplore";

          URL_Equipo = "http://" +
          Dispositivo_Temp.Get_DirIP().Get_Valor();
          proc.StartInfo.Arguments = URL_Equipo;
          proc.Start();
      }
```

```
        if (this.Equipos.Count == 0)
        {
            MessageBox.Show("NO EXISTE EQUIPOS");
        }
    }
```

Espacio de Código 2.32 Método para configurar dispositivos vía browser

2.3.3.11 Formulario form_snmp

El propósito de este formulario es crear una interfaz gráfica donde el usuario puede ejecutar las diferentes operaciones SNMP.

El formulario está compuesto por los siguientes elementos que son parte del formulario:

- Objetos de tipo textBox donde se ingresarán valores del objeto identificador y valores correspondientes al objeto MIB.
- Objetos de tipo button para ejecutar las operaciones GET, GETNEXT y SET.

En la figura 2.32 se observa cada uno de los componentes de este formulario.

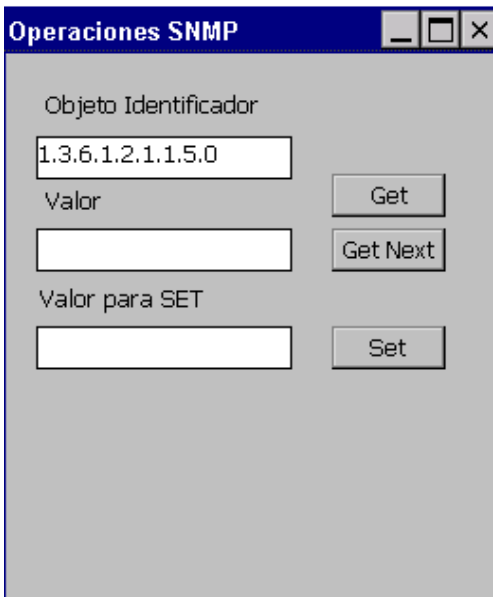


Figura 2.32 Formulario form_snmp

2.3.3.11.1. Método GET

El método GET permite realizar consultas SNMP de tipo GET, a un dispositivo anteriormente seleccionado en el formulario form_principal. En este método se construye la PDU SNMP tipo GET y lo encapsula en el paquete SNMP para ser enviado al agente respectivo.

```
private void btn_get_Click(object sender, EventArgs e)
  {
      int numero;
      numero = rnd.Next(0, 254);
      str_version = cmb_version.SelectItem.text;
      COID OID = new COID(txt_oid.Text.ToString());
      CNULL valor = new CNULL();
      CSEQUENCE secuencia = new CSEQUENCE(OID, valor);
      CGet pdu_get = new CGet(1, secuencia);

      CPaquete Paquete_SNMP = new CPaquete(str_version,
      this.Equipo.Get_RCommunity().Get_Valor(), pdu_get);
      session.enviar_paquete(Paquete_SNMP);

      CPaquete Paquete_SNMP1 = new
      CPaquete(session.obtener_paquete());

      CResponse pdu_resp = new
      CResponse(Paquete_SNMP1.Get_Paquete());

      CSEQUENCE VALORES;
      //VALORES = pdu_resp.Get_VarBind();
      VALORES = Paquete_SNMP1.Get_PDU().Get_VarBind();

      if (VALORES.Get_VALOR().Get_Tipo()=="IPADDRESS" ||
      VALORES.Get_VALOR().Get_Tipo()=="OID"||VALORES.Get_VALOR().Ge
      t_Tipo()=="OCTECT STRING")
      {
          txt_value2.Text = VALORES.Get_VALOR().Get_Valor();
      }

      if (VALORES.Get_VALOR().Get_Tipo()=="COUNTER" ||
      VALORES.Get_VALOR().Get_Tipo()=="GAUGE"||VALORES.Get_VALOR().
      Get_Tipo()=="INTEGER"||VALORES.Get_VALOR().Get_Tipo()=="NULL"
      ||VALORES.Get_VALOR().Get_Tipo()=="TIMETICKS")
      {
            txt_value2.Text =
            VALORES.Get_VALOR().Get_Valor1().ToString();
      }

      if (VALORES.Get_VALOR().Get_Tipo() == "COUNTER64")
      {
            txt_value2.Text =
            VALORES.Get_VALOR().Get_Valor64().ToString();
      }
  }
```

Espacio de Código 2.33 Método para recibir el valor de la consulta GET

Este método al recibir la respuesta del agente monitorizado discrimina cada uno de los valores de los objetos MIB de acuerdo al tipo de objeto. Entre los tipos de objetos de acuerdo a SMI para el protocolo SNMP versión uno y 2c.

2.3.3.11.2. Métodos GETNEXT y SET

Los métodos GETNEXT y SET permiten realizar operaciones SNMP de tipo GETNEXT y SET, a un dispositivo anteriormente seleccionado en el formulario form_principal. La implementación de estos métodos es similar al método GET de este formulario, con la única diferencia que se construyen PDU´s de tipo GETNEXT y SET.

2.3.3.12 Formulario form_trap

El propósito de este formulario es crear una interfaz gráfica que permite dar un reporte cuando el agente envía paquetes SNMP tipo TRAP. El formulario contiene un objeto de tipo datagrid donde se listan los reportes TRAP.

Dentro de las principales métodos de este formulario es llenar_datagrid(), cuyo objetivo es llenar al objeto de tipo datagrid con las Trap recibidas de los dispositivos monitorizados.

```
public void llenar_datagrid(int filas)
  {
      int m = filas;
      int k = 0;


      for (k = 0; k < m; k++)
      {
          CPaquete Trap_temp = (CPaquete)TRAPS[k];

          if (Trap_temp.Get_PDU().Get_Tipo() == 4)
          {
            data_grid[k, 0] =
            Trap_temp.Get_PDU().Get_AgentAddress().Get_Valor();
            data_grid[k, 1] =
            Convert.ToString(Trap_temp.Get_PDU().Get_GenericType())
            ;
             data_grid[k, 2] =
            Convert.ToString(Trap_temp.Get_PDU().Get_TimeStam().Get
            _Valor1());
```

```
            }

            if (Trap_temp.Get_PDU().Get_Tipo() == 7)
            {

              data_grid[k, 1] = "SysUpTime";
              data_grid[k, 2] =
              Convert.ToString(Trap_temp.Get_PDU().Get_VarBind().Get_
              VALOR().Get_Valor1());
            }

        }
    }
```

Espacio de Código 2.34 Método para mostrar el resultado de las Trap recibidas

En el espacio de código 2.34 se filtra las TRAP de acuerdo a la versión del protocolo SNMP, los valores del tipo de TRAP son 4 y 7 correspondientes a las versiones del protocolo SNMP versión uno y 2c.

Además como se observa en el espacio de código la PDU TRAP del protocolo SNMP versión uno, tiene en sus campos la dirección IP del agente, así como el valor del tipo de respuesta generada (SysUpTime, WarmStart, ..).

2.3.3.13 Formulario form_ping

El propósito de este formulario es crear una interfaz gráfica que permite comprobar la conectividad de un equipo monitorizado. El formulario contiene los siguientes elementos:

- Objetos de tipo textBox para especificar la dirección IP del equipo, el tamaño del paquete ICMP, tiempo de retardo máximo y el número de paquetes ICMP que se enviarán.
- Un objeto de tipo listBox donde se listarán los resultados a un ICMP echo request.

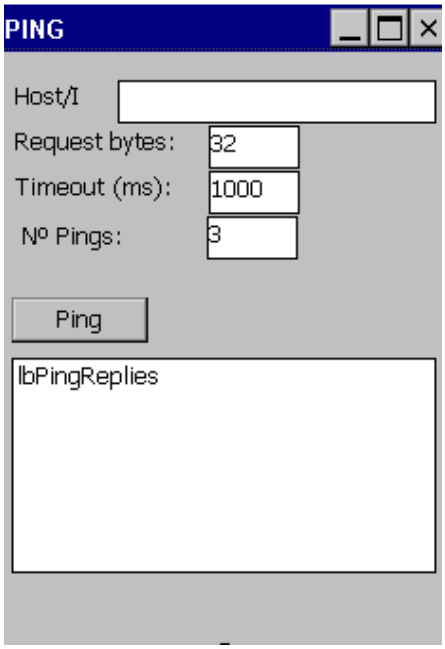


Figura 2.33 Formulario form_ping

2.3.3.13.1. Método Ping

El método Ping permite enviar paquetes ICMP echo request al equipo monitorizado. Este método construye un objeto de tipo CPing asignando atributos como la dirección IP y número de paquetes enviados como se observa en el espacio de código 2.35

```
private void cmdPing_Click_(object sender, EventArgs e)
  {

       icmp = new CPing();
       icmp.RequestSize = int.Parse(txtRequestSize.Text);
       icmp.RequestTimedOut += new
       CPing.PingTimeOutDelegate(icmp_RequestTimedOut);
       icmp.ReplyReceived += new
       CPing.PingReplyDelegate(icmp_ReplyReceived);
       icmp.PingError += new
       CPing.PingErrorDelegate(icmp_PingError);

       lbPingReplies.Items.Clear();

       icmp.Ping(this.Dispositivo.Get_DirIP().Get_Valor(),
       Convert.ToInt16(txt_npings.Text));

}
```

Espacio de Código 2.35 Método para enviar y recibir paquetes ICMP echo

Para verificar y observar el resultado del paquete ICMP echo request, se hace uso de los siguientes métodos:

```
private void icmp_PingError(object Sender, string ErrorMessage)
  {
      lbPingReplies.Items.Add("Error: " + ErrorMessage);
      Application.DoEvents();
  }


private void icmp_ReplyReceived(object sender, System.Net.IPAddress
ReplyIPAddress, int Bytes, int Time, int TTL)
  {
      lbPingReplies.Items.Add("Reply from " +
      ReplyIPAddress.ToString() + ":");
      lbPingReplies.Items.Add(" bytes=" + Bytes + " time=" + Time +
      "ms TTL=" + TTL);
      Application.DoEvents();
  }

private void icmp_RequestTimedOut(object sender,
System.Net.IPAddress ReplyIPAddress, int Bytes)
  {
      lbPingReplies.Items.Add("Request to " +
      ReplyIPAddress.ToString() + " timed out.");
      Application.DoEvents();
  }
```

Espacio de Código 2.36 Métodos para verificar y observar resultados ICMP echo

Cada uno de los métodos del espacio de código 2.36 están ligados con los delegados de la clase CPing, los cuales generan una respuesta del paquete ICMP echo request a los diferentes métodos del formulario form_ping.

El método icmp_PingError() informa sobre un error ocurrido cuando se envió un paquete ICMP echo request, errores de insuficiente buffer. El error es presentado en el objeto de tipo listBox.

El método icmp_RequestTimedOut() indica que el dispositivo no se encuentra conectado o la petición ICMP echo es timed out. El resultado es presentado en el objeto de tipo listBox.

Finalmente el método icmp_ReplyReceived() informa la respuesta a un paquete ICMP echo request, indicando los bytes enviados, tiempos de retardo del paquete, siempre y cuando el equipo monitorizado haya respondido la petición.

CAPÍTULO 3

PRUEBAS Y RESULTADOS OBTENIDOS

3.1 AMBIENTE DE PRUEBAS

En la figura 3.1 se muestra el diagrama de la red utilizado como ambiente de pruebas para el funcionamiento del software de monitorización de red. El mismo que consta de:

- PDA: La cual hace las funciones de la NMS. En esta se instalará nuestra aplicación la cual ejecutará aplicaciones que supervisan los dispositivos administrados.
- Access Point: Permite la comunicación entre la red cableada y la inalámbrica. Se transmite los datos entre los dispositivos conectados a la red cableada y la PDA.
- Dispositivo Administrado: Son nodos de red que contienen agentes SNMP. Estos elementos de red pueden ser: routers, switches, computadores, servidores, etc.

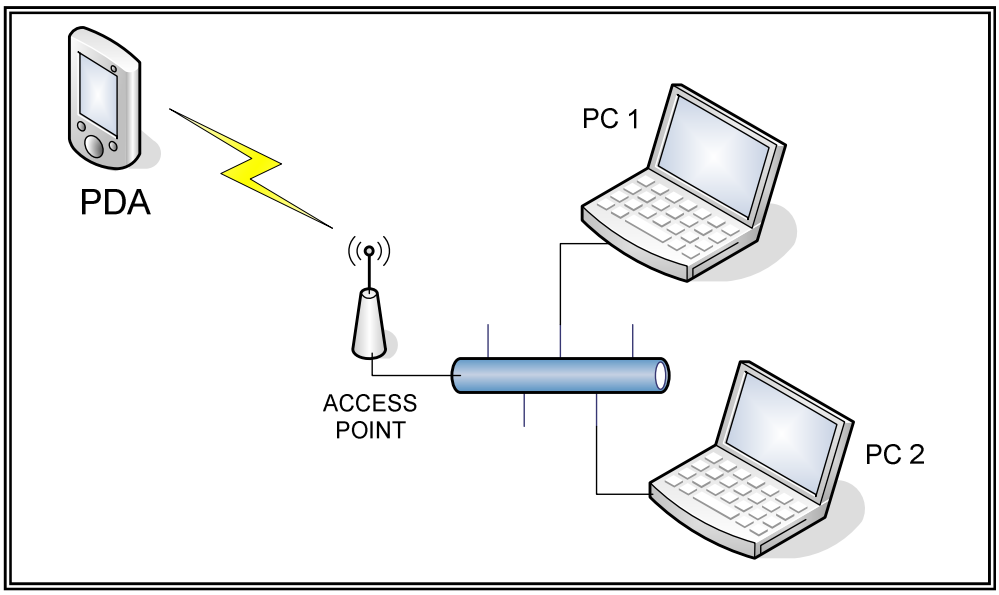


Figura: 3.1 Diagrama de Red

En el Anexo C se muestra el proceso de configuración de la red inalámbrica y la instalación del programa como parte del manual de usuario.

3.1.1 EL PROGRAMA

El software de monitorización de red para dispositivos inalámbricos consta de la ventana principal que se muestra en la figura 3.2.

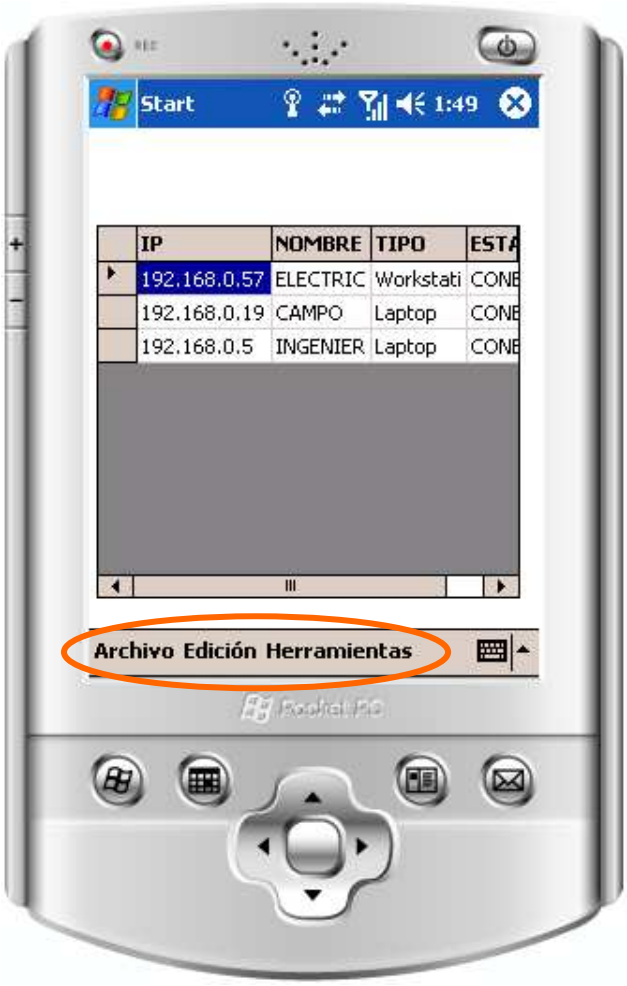


Figura 3.2 Ventana Principal

La barra de menú consta de: Archivo, Edición y Herramientas.

En el Menú Archivo encontramos las opciones: Nuevo y Salir.

Al seleccionar la opción Nuevo se desplegará la pantalla que se muestra en la figura 3.3. La misma que permite ingresar un nuevo dispositivo para ser gestionado. Para esto ingresamos datos como: la dirección IP, El nombre del dispositivo, el grupo de trabajo o la red donde se encuentra, el tipo de dispositivo (Workstation, Server, Laptop, Printer, Switch, Router, Access Poit, etc), La comunidad de Lectura y la comunidad de escritura.

Figura 3.3 Ventana para Agregar Dispositivos

Al seleccionar Salir, el programa dejará de ejecutarse y se cerrará.

En el Menú Edición tenemos las opciones de Edición y Borrar.

Al seleccionar Edición se desplegará la misma pantalla de Nuevo con la diferencia que me permite modificar los datos generales de los dispositivos gestionados a excepción de la dirección IP.

Figura 3.4 Ventana para Editar Dispositivos

La opción Borrar permite eliminar el dispositivo del software de monitorización de red.

En el Menú Herramientas encontramos las opciones: SNMP, Ping y Browser.

- **SNMP:** Despliega la pantalla que se muestra en la figura 3.5. En esta opción el usuario podrá explorar los objetos MIB de los dispositivos

gestionados, para esto deberá ingresar el Objeto Identificador y seleccionar la operación que desea realizar como GET, GET NEXT o SET.

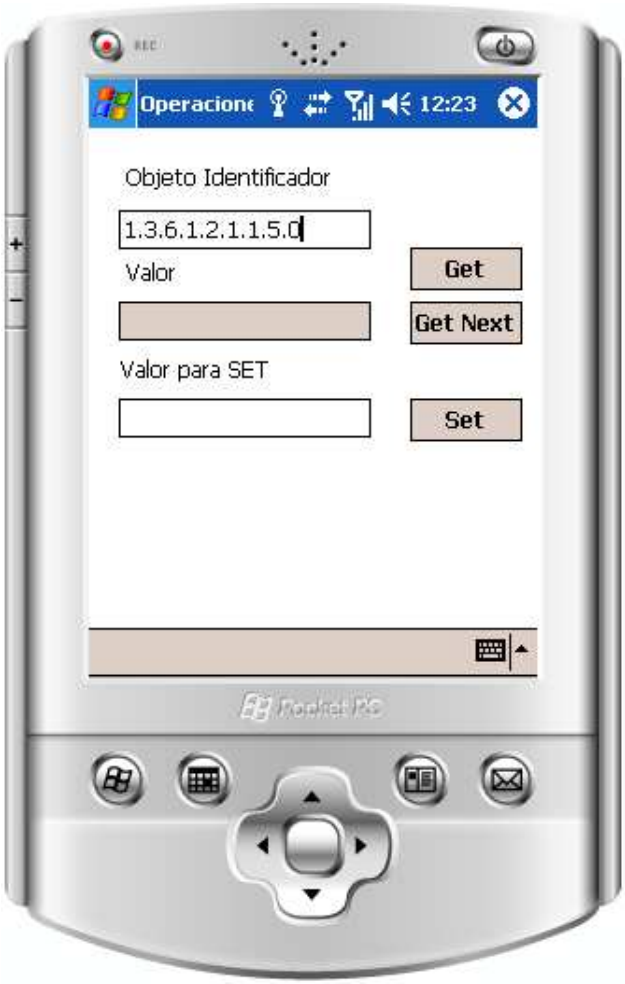


Figura 3.5 Ventana Operaciones SNMP

- **Ping:** Comprueba el estado de la conexión con el dispositivo seleccionado, por medio de paquetes de solicitud de eco y respuesta de eco. Al seleccionar esta opción se despliega la pantalla que se muestra en la figura 3.6.

Figura 3.6 Ventana PING

- Browser: Ejecutará un browser con la dirección IP del dispositivo seleccionado para configurar el equipo (Esto para equipos que permiten configurar sus parámetros por medio del browser).

3.2 PRUEBAS DE EXPLORACIÓN EN LA RED GESTIONADA

Se realizarán todas las pruebas de funcionamiento del software de monitorización de red (NMS) en dispositivos inalámbricos (PDA) sobre la red que se muestra en la figura 3.1.

En la opción NUEVO se ingresarán los datos de los dispositivos a ser gestionados. El Software utiliza la dirección IP del dispositivo como patrón principal para añadir el elemento, ya que antes de aceptarlo envía un ping. Si existe respuesta, aceptará el nuevo elemento para ser gestionado.

Ya ingresados todos los elementos de red en el software de monitorización de red, procedemos a realizar las pruebas de exploración en la red.

3.2.1. SNMP

Esta herramienta permite realizar solicitudes SNMP a los dispositivos de red que soporten este protocolo. El usuario debe ingresar en el campo *Objeto Identificador* el descriptor del objeto equivalente, por ejemplo 1.3.6.1.2.1.1.5.0 (iso.organizacion.dod.internet.mgmt.mib-2.system.SysName). Si desea obtener este valor deberá seleccionar GET. Si desea modificarlo debe también ingresar en el campo *Valor para SET* el nuevo valor a ser cambiado y seleccionar el botón SET. GET NEXT me permite obtener el siguiente objeto de la tabla. En la Figura 3.7 se muestra la ventana de esta herramienta.

Figura 3.7 Operaciones SNMP

3.2.2 TRAPS

Esta herramienta se ejecuta automáticamente al inicializar el programa y recibe todos los traps o reportes de ciertas condiciones o cambios de estado generados por los agentes de los dispositivos que se encuentren dentro de la red. En la Figura 3.8 muestra la respuesta ante un trap generado por un agente de un dispositivo dentro de la red.

Figura 3.8 Operación TRAP

3.2.3 PING

Esta herramienta es utilizada para verificar el estado de la conexión entre la NMS y los dispositivos que se encuentran dentro de la red. En la figura 3.9 muestra la ventana de ejecución de esta herramienta, donde se puede observar los tiempos

de respuesta. Esta herramienta es dinámica ya que el usuario puede ingresar en número de ping que desea de respuesta, el tamaño del paquete y los Timeouts.

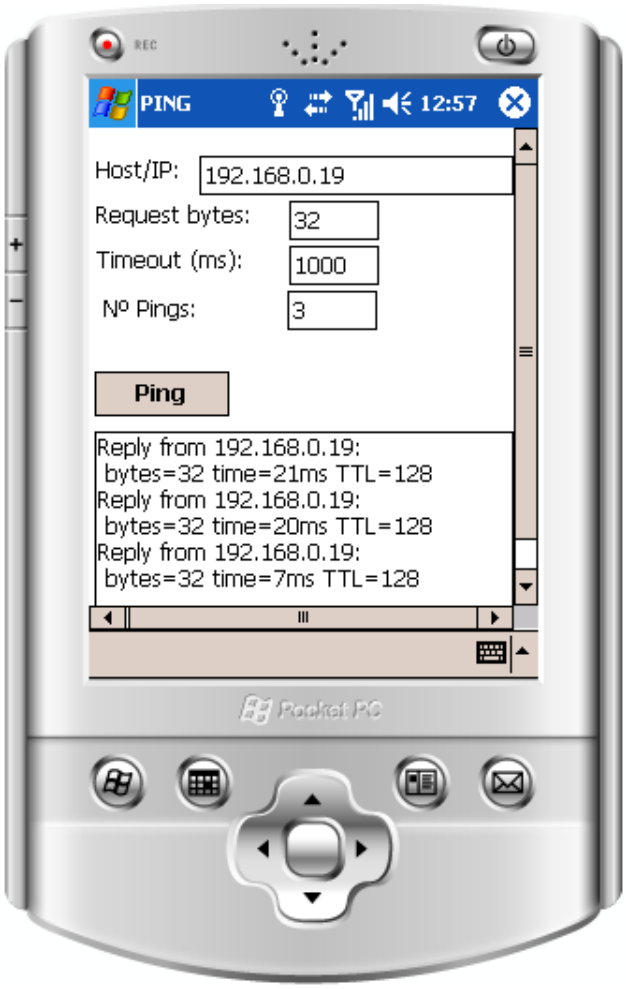


Figura 3.9 Operación PING

3.2.4 BROWSER

Al seleccionar esta opción se desplegará una pantalla de Explorer con la dirección IP del dispositivo gestionado. Esta herramienta será útil para configurar dispositivos que tengan esta opción como por ejemplo: routers, swiths, access point, etc. En la figura 3.10 se muestra la ventana de esta herramienta.

Figura 3.10 Operación Browser

3.3 TABULACIÓN DE RESULTADOS

Para tabular los resultados obtenidos utilizamos *ETHEREAL,* que es un software que permite capturar los paquetes enviados entre la NMS y los dispositivos de red gestionados.

ETHEREAL se ejecuta sobre una PC entre la NMS y los dispositivos gestionados, como se muestra en la figura 3.11

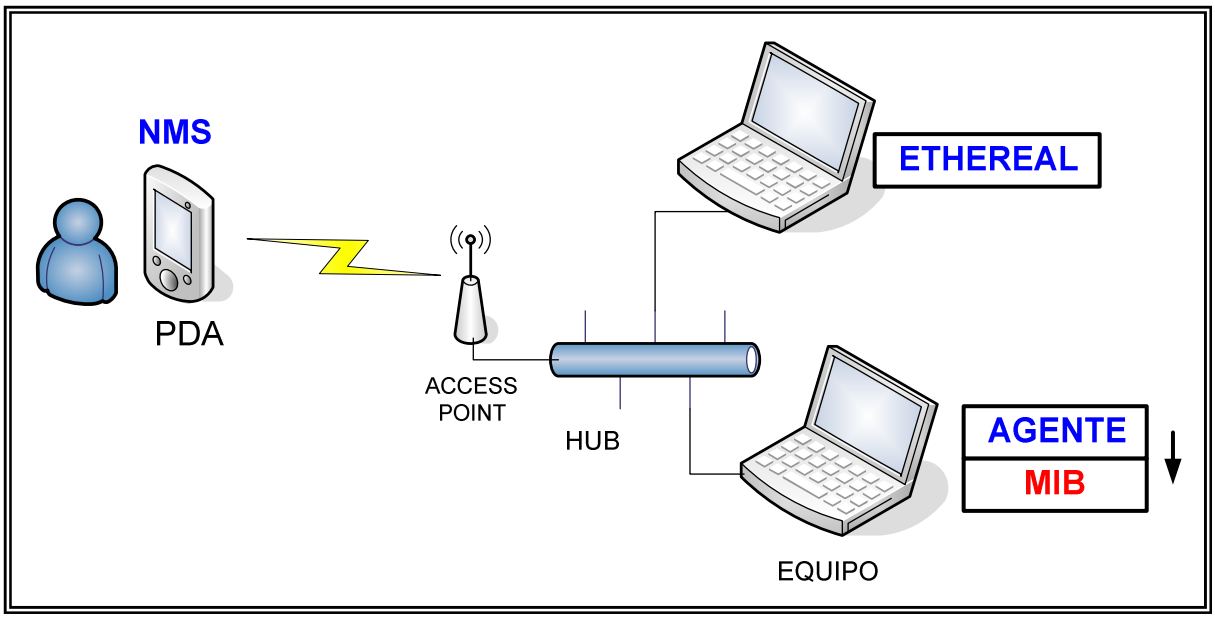


Figura 3.11 Capturas Ethereal

3.3.1 ESTABLECIMIENTO DE LA CONEXIÓN

Al ingresar un nuevo dispositivo a nuestra aplicación la NMS que corre sobre la PDA enviará un ping para verificar la conexión con el equipo remoto. Se envía un paquete de solicitud eco por parte de la NMS y se recibe un paquete de respuesta eco por parte del dispositivo a ser gestionado si existe conexión. Estos paquetes se muestran en la figura 3.12 de la captura del Ethereal. Luego de recibida la respuesta se abrirá un socket para establecer la conexión como se muestra en la figura 3.13.

```
⊞ Frame 122 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: 00:03:ff:ae:f8:f0, Dst: 00:16:36:0a:a9:9b
⊞ Internet Protocol, Src Addr: 192.168.0.7 (192.168.0.7), Dst Addr: 192.168.0.5 (192.168.0.5)
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xaafc (correct)
    Identifier: 0x0200
    Sequence number: 0x4b03
    Data (32 bytes)

⊞ Frame 123 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: 00:16:36:0a:a9:9b, Dst: 00:03:ff:ae:f8:f0
⊞ Internet Protocol, Src Addr: 192.168.0.5 (192.168.0.5), Dst Addr: 192.168.0.7 (192.168.0.7)
⊟ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xb2fc (correct)
    Identifier: 0x0200
    Sequence number: 0x4b03
    Data (32 bytes)
```

Figura 3.12 Captura PING

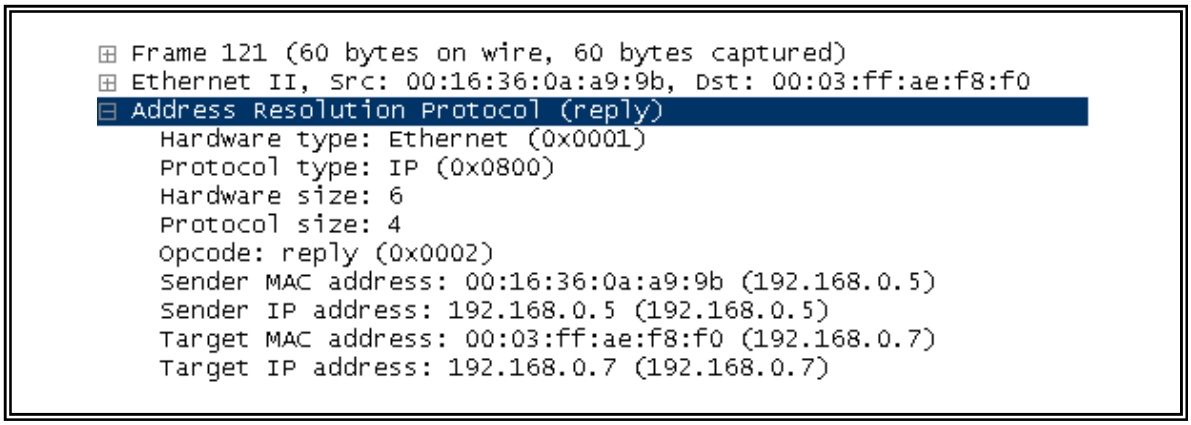


Figura 3.13 Establecimiento de la Conexión

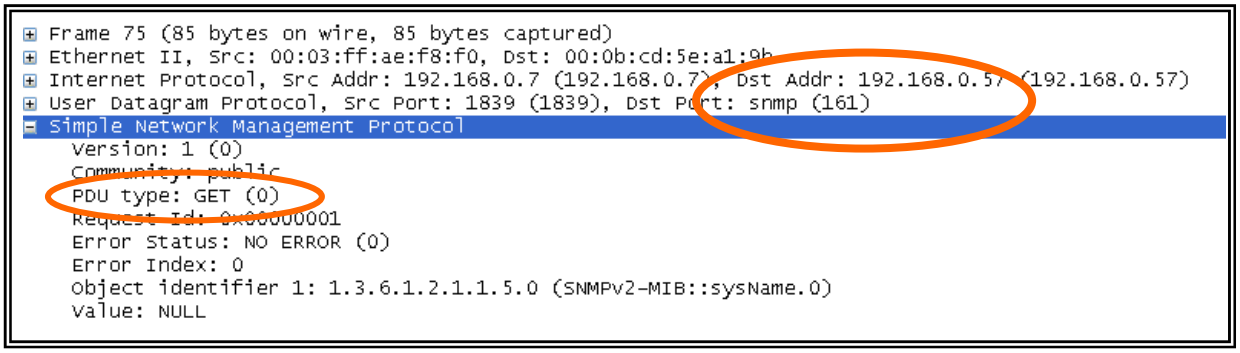
3.3.2 HERRAMIENTAS DE GESTIÓN DE RED

Al utilizar la Herramienta SNMP el usuario puede obtener o cambiar los valores de los objetos MIB del dispositivo gestionado.

3.3.2.1 GET

Al seleccionar GET se solicita al agente retornar el valor de un objeto de interés mediante su nombre. El agente contesta esta solicitud enviando una respuesta indicando el éxito o fracaso del requerimiento. Si el requerimiento fue adecuado, el mensaje resultante contendrá el valor del objeto solicitado. Estos paquetes de respuesta son enviados en un mensaje que se denomina GetResponse.

El proceso de los paquetes enviados y recibidos desde un equipo gestionado con Sistema operativo Windows XP se muestra en la figura 3.14.



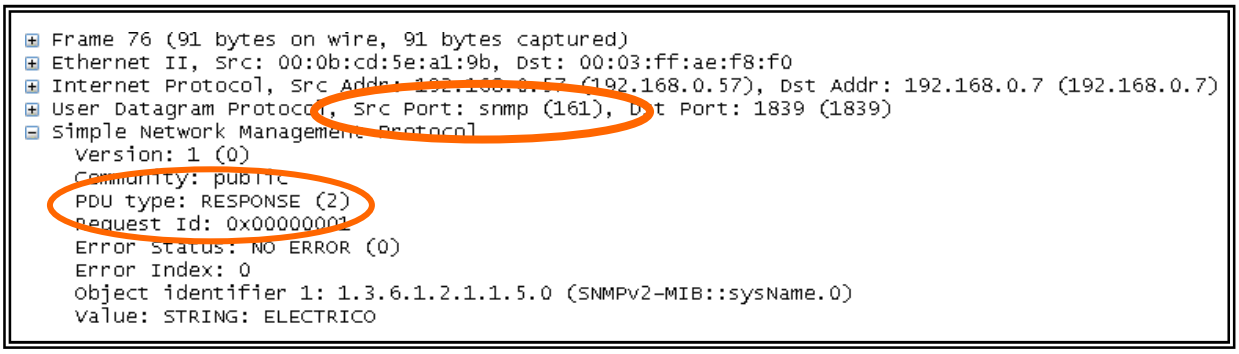


Figura 3.14 Paquete GET y RESPONSE para equipo con S.O. Windows

En la figura 3.15 se muestra la captura de una petición SNMP sysDescr hacia un dispositivo gestionado con sistema operativo Linux.

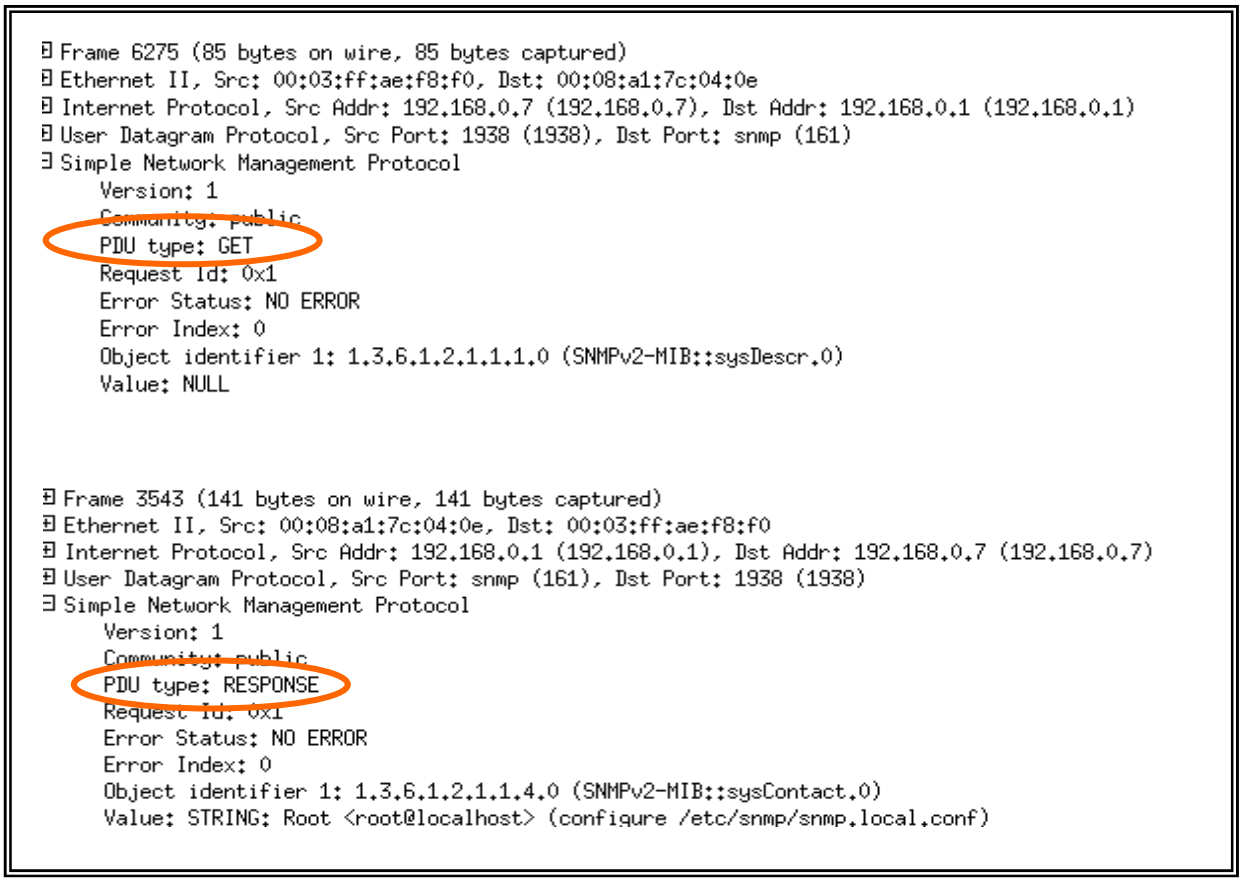


Figura 3.15 Paquete GET y RESPONSE para equipo con S.O. Linux

3.3.2.2 GET NEXT

Luego de haber utilizado GET para obtener el valor de un objeto, se utiliza GetNext para repetir la operación en el siguiente objeto de la tabla. El resultado

de la operación anterior siempre será utilizado para realizar la nueva consulta. Este proceso se muestra en la figura 3.16.

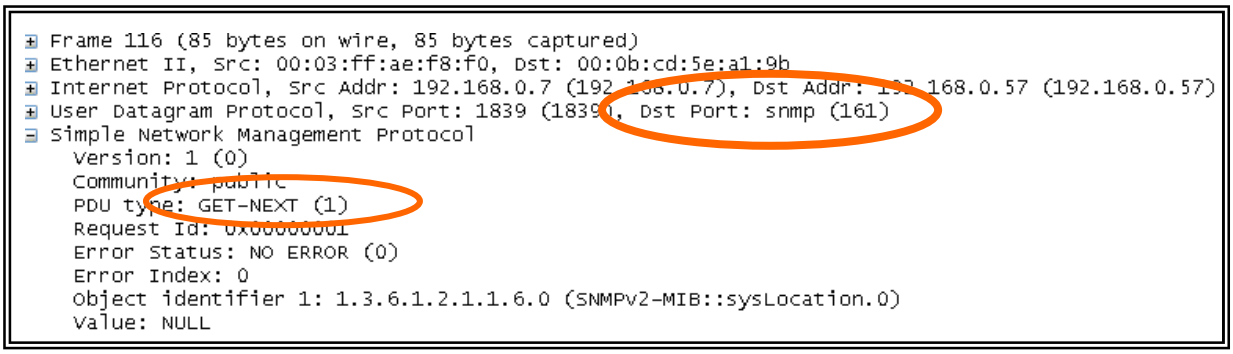


Figura 3.16 Paquete GET NEXT

3.3.2.3 SET

Al seleccionar SET la NMS solicitará a un agente modificar los valores de objetos. Para esto la NMS envía al agente una lista de nombres de objetos con sus correspondientes valores. Este proceso se observa en la figura 3.17.

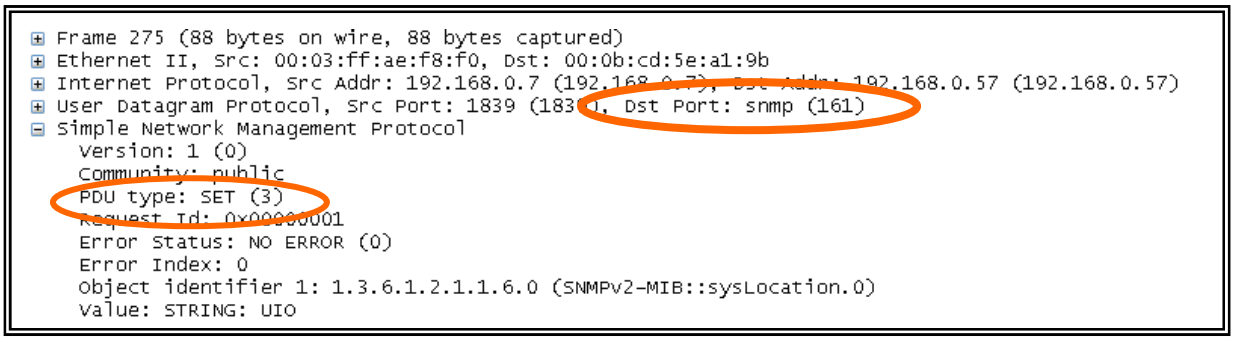


Figura 3.17: Paquete SET

3.3.2.4 TRAP

Las Traps son generadas por el agente, es decir son mensajes que se envían hacia la NMS en respuesta a ciertos acontecimientos.

En la figura 3.18 se muestra la captura de un paquete trap V1

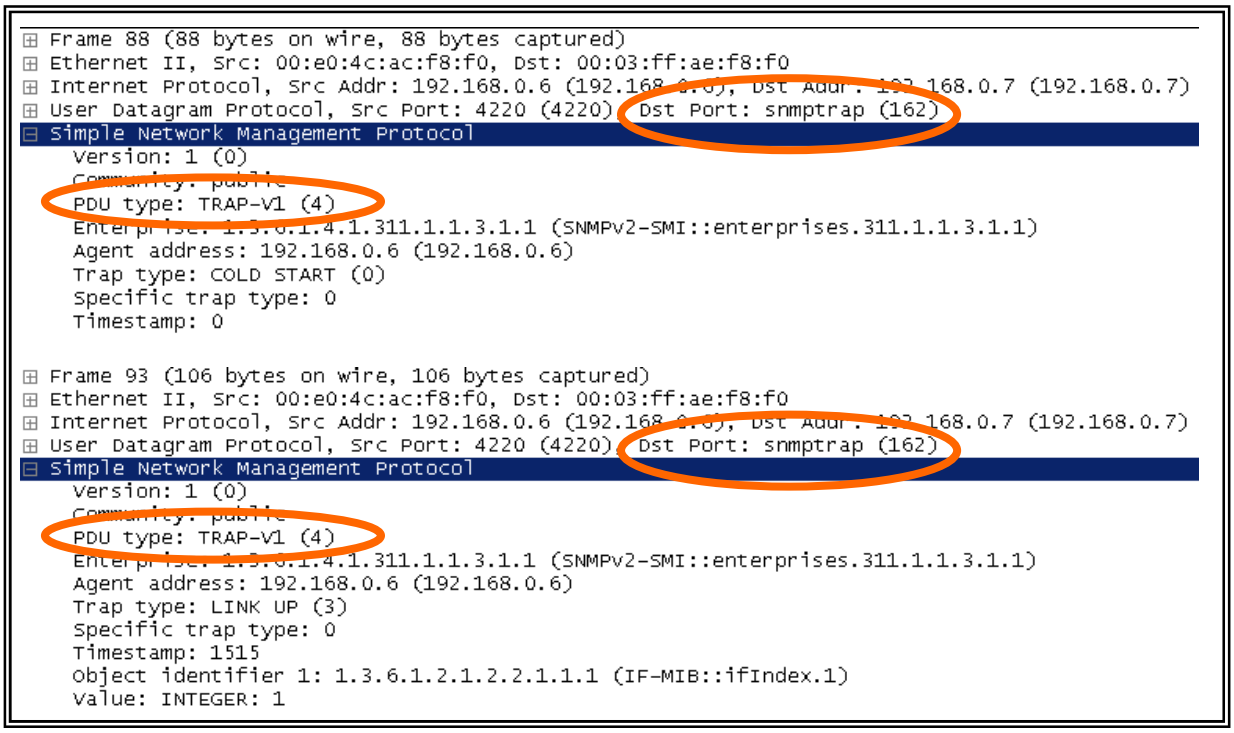


Figura 3.18 Paquetes enviados por el agente (Trap V1)

En la figura 3.19 se muestra la captura de un paquete trap V2

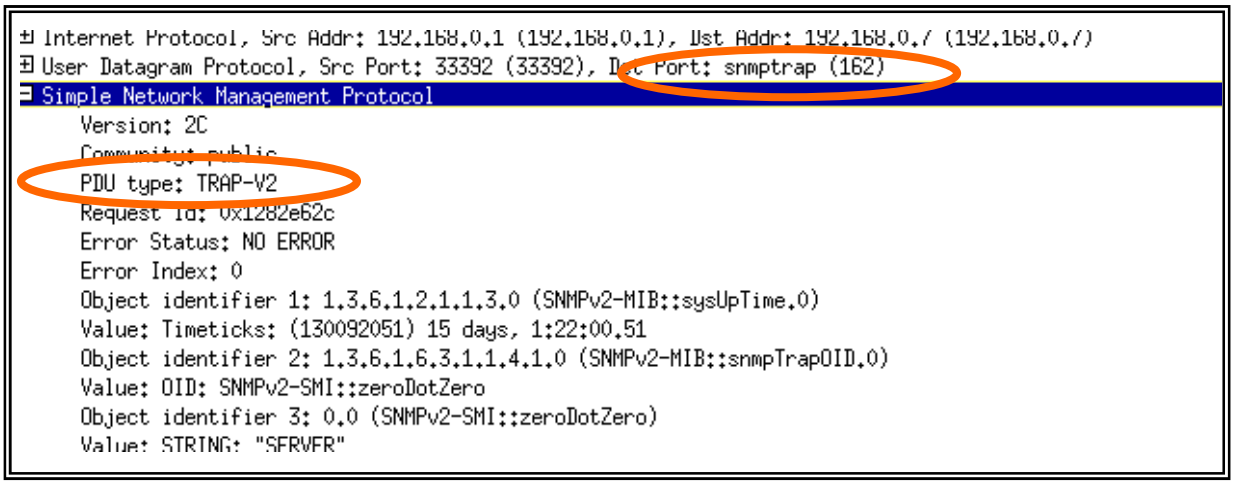


Figura 3.19 Paquete Trap V2

3.4 ANÁLISIS DE LOS RESULTADOS

A continuación se hará un análisis de los resultados obtenidos en el Ethereal, tales como los puertos utilizados, las direcciones origen y destino, así como el tipo de paquetes SNMP.

3.4.1 ESTABLECIMIENTO DE LA CONEXIÓN

En la figura 3.12 y 3.13 en la tabulación de resultados se observa como se establece la conexión hacia un equipo. El proceso comienza verificando si dicho equipo esta disponible enviando un paquete ICMP tipo ECHO REQUEST desde la PDA al equipo a ser gestionado y en espera de un paquete ICMP tipo ECHO REPLY por parte del EQUIPO.

Una vez recibido la respuesta ECHO (ICMP) por parte del equipo a ser monitorizado se procede abrir un socket con la dirección IP y el puerto 161 (SNMP) hacia el equipo monitorizado, este resultado se lo observa en la figura 3.13.

En la tabla 3.1 se muestra los valores en hexadecimal que se capturaron del *ETHEREAL* de la solicitud y respuestas ICMP Eco, cabe recalcar que los mensajes ICMP se transmiten como datagramas IP normales, con el campo de cabecera "protocolo" con un valor 1, y comienzan con un campo de 8 bits que define el tipo de mensaje de que se trata. La dirección IP del dispositivo remoto y el dispositivo local están contenidos en el paquete IP de capa 3.

ECO REQUEST		ECO REPLY		
Valor Hex	Valor Binario	Valor Hex	Valor Binario	Campo
08	0000 1000	00	0000 0000	**Type**
00	0000 0000	00	0000 0000	**Code (0)**
AA	1010 1010	B2	1011 0010	**Checksum (correct)**
FC	1111 1100	FC	1111 1100	
02	0000 0010	02	0000 0010	**Identifier 0X0200**
00	0000 0000	00	0000 0000	
4B	0100 1011	4B	0100 1011	**Sequence Number 0X4B03**
03	0000 0011	03	0000 0011	
00	0000 0000	00	0000 0000	**Data (32 Bytes)**
00	0000 0000	00	0000 0000	
00	0000 0000	00	0000 0000	
00	0000 0000	00	0000 0000	
.	.	.	.	
.	.	.	.	

.	.	.	.	
.	.	.	.	
00	0000 0000	00	0000 0000	
00	0000 0000	00	0000 0000	
00	0000 0000	00	0000 0000	
00	0000 0000	00	0000 0000	

Tabla 3.1 Solicitud y Respuesta ECO

3.4.2 GET

En la captura del ETHEREAL de la figura 3.20 se observa que un paquete SNMP tipo GET el campo valor es nulo, debido a que es una consulta, es decir no posee información a diferencia de los paquetes SNMP tipo RESPONSE, SET y TRAP. Generalmente los paquetes SNMP tipo GET poseen un estado de error NO ERROR, debido a que este paquete no es una respuesta a una consulta SNMP.

Además se observa claramente la versión, comunidad y el OID del paquete SNMP tipo GET, donde en este caso se está consultando sysName del equipo con dirección IP 192.168.0.57.

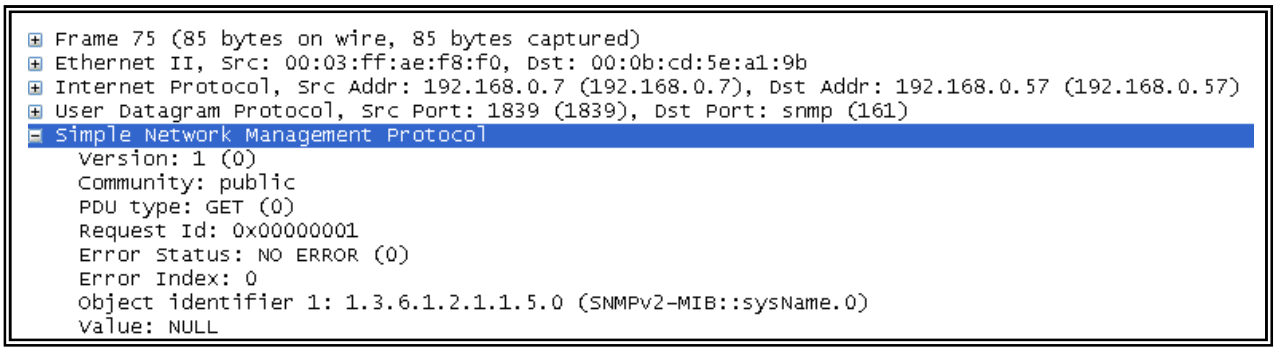


Figura 3.20 Paquetes GET

En el Anexo D se muestra las capturas en hexadecimal de *ETHEREAL* y el proceso de codificación de un paquete SNMP Get y trap.

En la tabla 3.2 se muestra los valores capturados del *ETHEREAL* en hexadecimal de un datagrama SNMP GET codificado con la solicitud SysName. El resto de columnas de la tabla indica los diferentes valores codificados de los campos del paquete SNMP GET.

Valor Hex	Valor Binario	Característica	Campo			
30	00 1 10000	Tipo = Sequence	Mensaje SNMP			MENSAJE SNMP
26	0010 0110	Longitud = 38 Bytes				
02	0000 0010	Tipo = Integer	Versión			
01	0000 0001	Longitud = 1				
00	0000 0000	Valor = 0				
04	0000 0100	Tipo = Octet String	Comunidad			
06	0000 0110	Longitud = 6				
70	0111 0000	P				
75	0111 0101	U				
62	0110 0010	B				
6C	0110 1100	L				
69	0110 1001	I				
63	0110 0011	C				
A0	10 1 00000	Tipo = GetRequest	SNMP PDU GetRequest		SNMP PDU	
19	0001 1001	Longitud = 25 Bytes				
02	0000 0010	Tipo = Integer	Request ID 0X00000001			
01	0000 0001	Longitud = 1				
01	0000 0001	Valor = 1				
02	0000 0010	Tipo = Integer	Error Status: No Error			
01	0000 0001	Longitud = 1				
00	0000 0000	Valor = 0				
02	0000 0010	Tipo = Integer	Error Index: 0			
01	0000 0001	Longitud = 1				
00	0000 0000	Valor = 0				
30	00 1 10000	Tipo = Sequence	Varbind List		Varbind List	
0E	0000 1110	Longitud = 14 Bytes				
30	00 1 10000	Tipo = Sequence	Varbind	Varbind		
0C	0000 1100	Longitud = 12 Bytes				
06	00 0 00110	Tipo = OBJECT ID	Object Identifier			
08	0000 1000	Longitud = 8 Bytes				
2B	0010 1011	Codificación 40*1+3 = 43 1 = iso; 3 = org				
06	0000 0110	Dod				
01	0000 0001	Internet				
02	0000 0010	Mgmt				
01	0000 0001	Mib-2				
01	0000 0001	System				
05	0000 0101	SysName				
00	0000 0000	0				
05	0000 0101	Tipo = NULL	Value			

00	0000 0000	Longitud = 0					

Tabla 3.2 Datagrama SNMP GET

3.4.3 RESPONSE

Cuando se ha realizado una operación GET, SET o GET NEXT se obtiene como resultado una respuesta por parte del agente del equipo monitorizado, en este caso se recupera un paquete SNMP tipo RESPONSE. Este paquete devuelve información del valor de la MIB consultada por parte de los paquetes SNMP tipo GET, GET NEXT y SET, tal como se observa en la figura 3.21.

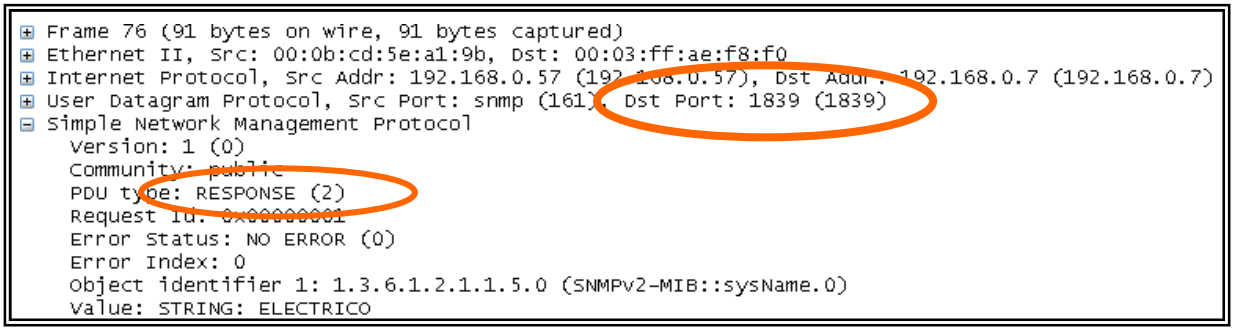


Figura 3.21 Paquete GET NEXT

Además se observa que un paquete SNMP tipo REQUEST tiene un valor de 2 en el campo tipo de PDU de este paquete.

En la tabla 3.3 se muestra los valores capturados del *ETHEREAL* en hexadecimal de un datagrama SNMP Responce codificado con la respuesta a la solicitud SysName.

Valor Hex	Valor Binario	Característica	Campo	
30	00 1 10000	Tipo = Sequence	**Mensaje SNMP**	**MENSAJE SNMP**
2F	0010 1111	Longitud = 47 Bytes		
02	0000 0010	Tipo = Integer	**Versión**	
01	0000 0001	Longitud = 1		
00	0000 0000	Valor = 0		
04	0000 0100	Tipo = Octet String	**Comunidad**	
06	0000 0110	Longitud = 6		
70	0111 0000	P		

75	0111 0101	U				
62	0110 0010	B				
6C	0110 1100	L				
69	0110 1001	I				
63	0110 0011	C				
A2	10 1 00010	Tipo = GetResponce	**SNMP PDU Responce**			SNMP PDU
22	0010 0010	Longitud = 34 Bytes				
02	0000 0010	Tipo = Integer	**Request ID 0X00000001**			
01	0000 0001	Longitud = 1				
01	0000 0001	Valor = 1				
02	0000 0010	Tipo = Integer	**Error Status: No Error**			
01	0000 0001	Longitud = 1				
00	0000 0000	Valor = 0				
02	0000 0010	Tipo = Integer	**Error Index: 0**			
01	0000 0001	Longitud = 1				
00	0000 0000	Valor = 0				
30	00 1 10000	Tipo = Sequence	**Varbind List**		Varbind List	
17	0001 0111	Longitud = 23 Bytes				
30	00 1 10000	Tipo = Sequence	**Varbind**	Varbind		
15	0000 1100	Longitud = 21 Bytes				
06	00 0 00110	Tipo = OBJECT ID	**Object Identifier**			
08	0000 1000	Longitud = 8 bytes				
2B	0010 1011	Codificación 40*1+3 = 43 1 = iso; 3 = org				
06	0000 0110	Dod				
01	0000 0001	Internet				
02	0000 0010	Mgmt				
01	0000 0001	Mib-2				
01	0000 0001	System				
05	0000 0101	SysName				
00	0000 0000	0				
04	0000 0100	Tipo = Octet String	**Value**			
09	0000 1001	Longitud = 9				
65	0110 0101	E				
6C	0110 1100	L				
65	0110 0101	E				
63	0110 0011	C				
74	0111 0100	T				
72	0111 0010	R				
69	0110 1001	I				
63	0110 0011	C				
6F	0110 1111	O				

Tabla 3.3 Datagrama SNMP GetResponce

3.4.4 GET NEXT

Este paquete es similar al paquete SNMP tipo GET a diferencia que en el campo tipo de PDU tiene un valor de 1, tal como se observa en la figura 3.22. Posee un valor de NULL en el campo valor, al igualmente que el paquete SNMP tipo GET.

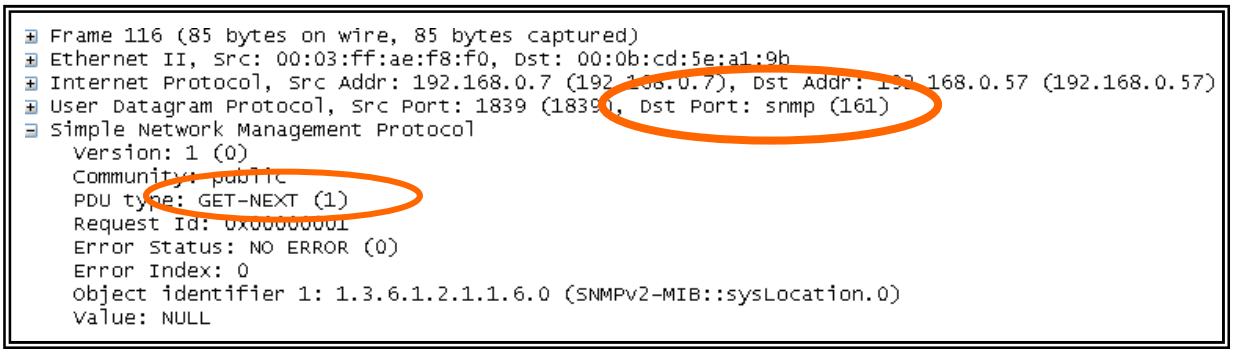


Figura 3.22 Paquete GET NEXT

En la tabla 3.4 se muestra los valores capturados del *ETHEREAL* en hexadecimal de un datagrama SNMP GetNext.

Valor Hex	Valor Binario	Característica	Campo		
30	00 1 10000	Tipo = Sequence	**Mensaje SNMP**		MENSAJE SNMP
26	0010 0110	Longitud = 38 Bytes			
02	0000 0010	Tipo = Integer	**Versión**		
01	0000 0001	Longitud = 1			
00	0000 0000	Valor = 0			
04	0000 0100	Tipo = Octet String	**Comunidad**		
06	0000 0110	Longitud = 6			
70	0111 0000	P			
75	0111 0101	U			
62	0110 0010	B			
6C	0110 1100	L			
69	0110 1001	I			
63	0110 0011	C			
A1	10 1 00001	Tipo = GetNextRequest	**SNMP PDU GetNextRequest**	SNMP PDU	
19	0001 1001	Longitud = 25 Bytes			
02	0000 0010	Tipo = Integer	**Request ID 0X00000001**		
01	0000 0001	Longitud = 1			
01	0000 0001	Valor = 1			

02	0000 0010	Tipo = Integer	**Error Status: No Error**				
01	0000 0001	Longitud = 1					
00	0000 0000	Valor = 0					
02	0000 0010	Tipo = Integer	**Error Index: 0**				
01	0000 0001	Longitud = 1					
00	0000 0000	Valor = 0					
30	00 1 10000	Tipo = Sequence	**Varbind List**		**Varbind List**		
0E	0000 1110	Longitud = 14 Bytes					
30	00 1 10000	Tipo = Sequence	**Varbind**	**Varbind**			
0C	0000 1100	Longitud = 12 Bytes					
06	00 0 00110	Tipo = OBJECT ID	**Object Identifier**				
08	0000 1000	Longitud = 8 Bytes					
2B	0010 1011	Codificación 40*1+3 = 43 1 = iso; 3 = org					
06	0000 0110	Dod					
01	0000 0001	Internet					
02	0000 0010	Mgmt					
01	0000 0001	Mib-2					
01	0000 0001	System					
06	0000 0110	SysLocation					
00	0000 0000	0					
05	0000 0101	Tipo = NULL	**Value**				
00	0000 0000	Longitud = 0					

Tabla 3.4 Datagrama SNMP GetNext

3.4.5 SET

En la Figura 3.23 se muestra la captura obtenida del Etherial de este paquete, donde se puede observar que el paquete SNMP tipo SET tiene un valor de 3 en el campo tipo de PDU, el valor UIO tipo string para la MIB sysLocation. Además se observa que la dirección y puerto destino son 192.168.0.57 y 161 respectivamente.

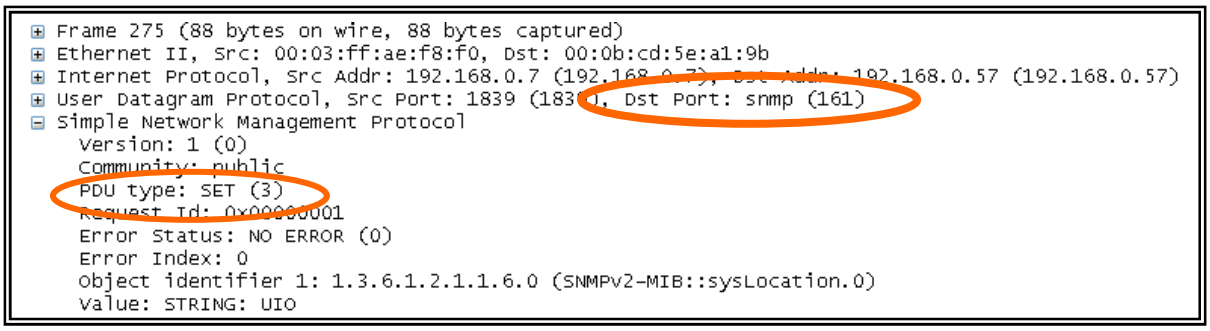


Figura 3.23 Paquete SET

En la tabla 3.5 se muestra los valores capturados del *ETHEREAL* en hexadecimal de un datagrama SNMP SET.

Valor Hex	Valor Binario	Característica	Campo		
30	00 1 10000	Tipo = Sequence	**Mensaje SNMP**		MENSAJE SNMP
29	0010 1001	Longitud = 41 Bytes			
02	0000 0010	Tipo = Integer	**Versión**		
01	0000 0001	Longitud = 1			
00	0000 0000	Valor = 0			
04	0000 0100	Tipo = Octet String	**Comunidad**		
06	0000 0110	Longitud = 6			
70	0111 0000	P			
75	0111 0101	U			
62	0110 0010	B			
6C	0110 1100	L			
69	0110 1001	I			
63	0110 0011	C			
A3	10 1 00010	Tipo = SetRequest	**SNMP PDU SetRequest**		SNMP PDU
1C	0001 1100	Longitud = 28 Bytes			
02	0000 0010	Tipo = Integer	**Request ID 0X00000001**		
01	0000 0001	Longitud = 1			
01	0000 0001	Valor = 1			
02	0000 0010	Tipo = Integer	**Error Status: No Error**		
01	0000 0001	Longitud = 1			
00	0000 0000	Valor = 0			
02	0000 0010	Tipo = Integer	**Error Index: 0**		
01	0000 0001	Longitud = 1			
00	0000 0000	Valor = 0			
30	00 1 10000	Tipo = Sequence	**Varbind List**		
11	0001 0001	Longitud = 17 Bytes			

30	00 1 10000	Tipo = Sequence	**Varbind**	**Varbind**
0F	0000 1100	Longitud = 15 Bytes		
06	00 0 00110	Tipo = OBJECT ID	**Object Identifier**	
08	0000 1000	Longitud = 8 bytes		
2B	0010 1011	Codificación 40*1+3 = 43 1 = iso; 3 = org		
06	0000 0110	Dod		
01	0000 0001	Internet		
02	0000 0010	Mgmt		
01	0000 0001	Mib-2		
01	0000 0001	System		
06	0000 0110	SysLocation		
00	0000 0000	0		
04	0000 0100	Tipo = Octet String	**Value**	
03	0000 0011	Longitud = 3		
75	0110 0101	U		
69	0110 1100	I		
6F	0110 0101	O		

Tabla 3.5 Datagrama SNMP SET

3.4.6 TRAP

En la figura 3.24 se observa el paquete SNMP tipo TRAP V1 el cual utiliza el puerto 162, pero el agente monitoreado envía a este puerto como destino, es decir la NMS que esta corriendo en la PDA permanece escuchando en este puerto en busca de paquetes SNMP tipo trap.

Se observa además que el valor tipo de PDU es 4, así como el tipo de TRAP y la dirección del agente son: LINK UP y 192.168.0.6 respectivamente.

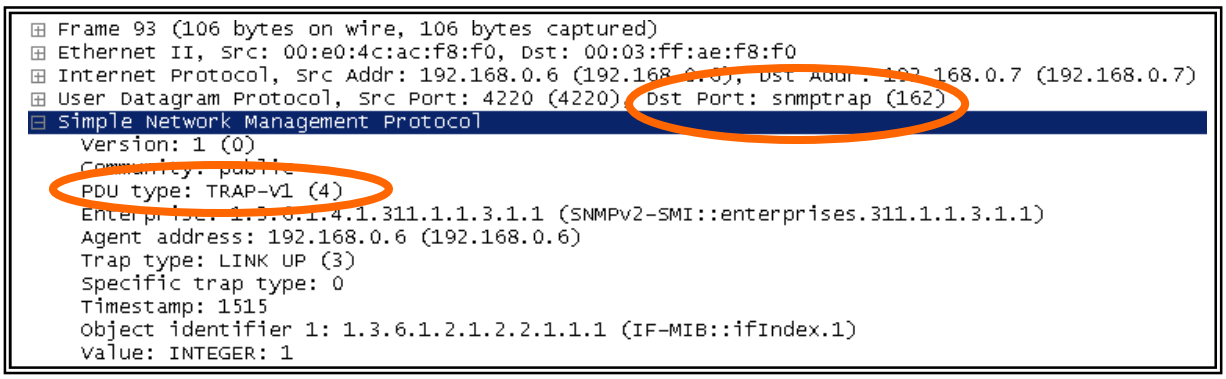


Figura 3.24 Paquetes enviados por el agente

En la figura 3.19 se muestra la captura de un paquete trap V2, el cual tiene una estructura similar a la de Get, GetNext, Responce y Set. La diferencia esta en que trap V2 tiene el valor de 7 en el campo Tipo de PDU

En la tabla 3.6 se muestra los valores capturados del *ETHEREAL* en hexadecimal de un datagrama TRAP V1.

Valor Hex	Valor Binario	Característica	Campo		
30	00 1 10000	Tipo = Sequence	**Mensaje SNMP**		**MENSAJE SNMP**
3E	0010 0000	Longitud = 62 Bytes			
02	0000 0010	Tipo = Integer	**Versión**		
01	0000 0001	Longitud = 1			
00	0000 0000	Valor = 0			
04	0000 0100	Tipo = Octet String	**Comunidad**		
06	0000 0110	Longitud = 6			
70	0111 0000	P			
75	0111 0101	U			
62	0110 0010	B			
6C	0110 1100	L			
69	0110 1001	I			
63	0110 0011	C			
A4	10 1 00010	Tipo = Trap-V1	**SNMP PDU TRAP**	**SNMP PDU**	
31	0011 0001	Longitud = 49 Bytes			
06	00 0 00110	Tipo = OBJECT ID	**Enterprise (1.3.6.1.4.1.311.1.1.3 .1.1)**		
0C	0000 1100	Longitud = 12 bytes			
2B	0010 1011	Codificación 40*1+3 = 43 1 = iso; 3 = org			

06	0000 0110	6				
01	0000 0001	1				
04	0000 0100	4				
01	0000 0001	1				
82	1000 0010	311				
37	0011 0111					
01	0000 0001	1				
01	0000 0001	1				
03	0000 0011	3				
01	0000 0001	1				
01	0000 0001	1				
40	1000 0000	Tipo = IP Address	**Agent-Address (192.168.0.6)**			
04	0000 0100	Longitud = 4				
C0	1100 0000	192				
A8	1010 1000	168				
00	0000 0000	0				
06	0000 0110	6				
02	0000 0010	Tipo = Integer	**Generic-Trap Type: LINK UP**			
01	0000 0001	Longitud = 1				
03	0000 0011	Valor = 3				
02	0000 0010	Tipo = Integer	**Specific-Trap (0)**			
01	0000 0001	Longitud = 1				
00	0000 0000	Valor = 0				
43	0100 0011	Tipo = Timestamp	**Time-stamp (1515)**			
02	0000 0010	Longitud = 2				
05	0000 0101	10111101011 = 1515				
EB	11101011					
30	00 1 10000	Tipo = Sequence	**Varbind List**			
11	0001 0001	Longitud = 17 Bytes				
30	00 1 10000	Tipo = Sequence	**Varbind**	**Varbind**	**Varbind List**	
0F	0000 1111	Longitud = 15 Bytes				
06	00 0 00110	Tipo = OBJECT ID	**Object Identifier**			
0A	0000 1010	Longitud = 10 Bytes				
2B	0010 1011	Codificación 40*1+3 = 43 1 = iso; 3 = org				
06	0000 0110	6				
01	0000 0001	1				
02	0000 0010	2				
01	0000 0001	1				
02	0000 0010	2				
02	0000 0010	2				
01	0000 0001	1				

01	0000 0001	1					
01	0000 0001	1					
02	0000 0010	Tipo = Integer					
01	0000 0001	Longitud = 1	**Value**				
01	0110 0001	Valor = 1					

Tabla 3.6 Datagrama TRAP V1

En la tabla 3.7 se muestra los valores capturados del *ETHEREAL* en hexadecimal de un datagrama TRAP V2.

Valor Hex	Valor Binario	Característica	Campo		
30	00 1 10000	Tipo = Sequence	**Mensaje SNMP**		MENSAJE SNMP
4B	0100 1011	Longitud = 75 Bytes			
02	0000 0010	Tipo = Integer	**Versión 2**		
01	0000 0001	Longitud = 1			
01	0000 0001	Valor = 1			
04	0000 0100	Tipo = Octet String	**Comunidad**		
06	0000 0110	Longitud = 6			
70	0111 0000	P			
75	0111 0101	U			
62	0110 0010	B			
6C	0110 1100	L			
69	0110 1001	I			
63	0110 0011	C			
A7	10 1 00111	Tipo = Trap-V2	**SNMP PDU TRAP**	SNMP PDU	
3E	0011 1110	Longitud = 62 Bytes			
02	0000 0010	Tipo = Integer	**Request ID 0X1282E62C**		
04	0000 0100	Longitud = 4 Bytes			
12	0001 0010	12			
82	1000 0010	82			
E6	1110 0110	E6			
2C	0010 1100	2C			
02	0000 0010	Tipo = Integer	**Error Status: No Error**		
01	0000 0001	Longitud = 1			
00	0000 0000	Valor = 0			
02	0000 0010	Tipo = Integer	**Error Index: 0**		
01	0000 0001	Longitud = 1			
00	0000 0000	Valor = 0			
30	00 1 10000	Tipo = Sequence	**Varbind List**		
30	0011 0000	Longitud = 48 Bytes			
30	00 1 10000	Tipo = Sequence	**Varbind**		

10	0001 0000	Longitud = 16 Bytes		
06	00 0 00110	Tipo = OBJECT ID	**Object Identifier (1.3.6.1.2.1.1.3.0)**	
08	0000 1010	Longitud = 8 Bytes		
2B	0010 1011	Codificación 40*1+3 = 43 1 = iso; 3 = org		
06	0000 0110	6		
01	0000 0001	1		
02	0000 0010	2		
01	0000 0001	1		
01	0000 0001	1		
03	0000 0011	3		
00	0000 0000	0		
43	0100 0011	Tipo = Timestamp	**Time-Ticks**	
04	0000 0100	Longitud = 4		
07	0000 0111	130092051		
C1	1100 0001			
0C	0000 1100			
13	0001 0011			
30	00 1 10000	Tipo = Sequence	**Varbind**	**Varbind**
0F	0001 0000	Longitud = 15 Bytes		
06	00 0 00110	Tipo = OBJECT ID	**Object Identifier (1.3.6.1.6.3.1.1.4.1.0)**	
0A	0000 1010	Longitud = 10 Bytes		
2B	0010 1011	Codificación 40*1+3 = 43 1 = iso; 3 = org		
06	0000 0110	6		
01	0000 0001	1		
06	0000 0110	6		
03	0000 0011	3		
01	0000 0001	1		
01	0000 0001	1		
04	0000 0100	4		
01	0000 0001	1		
00	0000 0000	0		
06	0000 0110	Tipo = OBJECT ID	**Object Identifier (0)**	
01	0000 0001	Longitud = 1		
00	0000 0000	Valor = 0		
30	00 1 10000	Tipo = Sequence	**Varbind**	**Varbind**
0B	0000 1011	Longitud = 11		
06	0000 0110	Tipo = OBJECT ID	**Object Identifier (0)**	
01	0000 0001	Longitud = 1		
00	0000 0000	Valor = 0		
04	0000 0100	Tipo = Octet String	**Value**	
06	0000 0110	Longitud = 6		
53	0101 0011	S		
45	0100 0101	E		

52	0101 0010	R						
56	0101 0110	V						
45	0100 0101	E						
52	0101 0010	R						

Tabla 3.7 Datagrama TRAP V2c

CAPÍTULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

1. Para el intercambio de información de administración entre dispositivos de red se utiliza el protocolo SNMP el cual utiliza un servicio no orientado a la conexión (UDP). El mismo que permite interactuar directamente con los dispositivos gestionados, obteniendo y modificando la información contenida en las MIB´s de cada elemento.

2. Es indispensable que todos los elementos de red a ser monitoreados soporten el protocolo SNMP, ya que es la base para la operación de nuestra NMS.

3. Al tener dispositivos en la red con opción a configuración en modo browser como: Switchs, Routers, Access Point, Servers, etc.; permitirán al administrador realizar esta acción remotamente desde la PDA, lo que optimiza los tiempos de respuesta ante cualquier acontecimiento.

4. La utilización de sockets en el desarrollo de aplicaciones que permitan la administración y gestión de redes es indispensable ya que permite el intercambio de datos en dos vías de una manera fiable y ordenada. Lo que permite que las operaciones de gestión se realicen paralelamente a la ejecución del programa.

5. El software de gestión de red para dispositivos inalámbricos PDA por estar basado en el protocolo SNMP interactúa sin ningún problema con dispositivos que utilizan el sistema operativo Windows o Linux.

6. Windows CE al ser una aplicación para dispositivos portátiles con limitaciones de procesamiento no posee las herramientas básicas de red, por lo que se tuvo que desarrollar una aplicación para determinar el estado de conexión con otros dispositivos (Ping). La misma que envía solicitudes Eco en espera de respuestas Eco.

7. Para la implementación de la clase CPing se utilizó la librería iphlpapi.dll, que es un módulo que contiene las funciones usadas por el Windows IP Heplper API, permitiendo enviar paquetes ICMP tipo echo request hacia el equipo monitorizado.

8. El uso de sockets facilitó la implementación de las clases SNMP, para enviar los paquetes SNMP al puerto correspondiente, debido a que no se encontró librerías de tipo SNMP que soporte la PDA.

9. Los Threads fueron utilizados en el desarrollo del programa, para que las operaciones de Polling y TRAP no interfieran con el funcionamiento del menú principal. Las operaciones de Polling y TRAP se ejecutan en modo background.

10. Para el desarrollo de la aplicación se escogió C# porque combina el control de lenguaje de bajo nivel como C y el manejo de ventanas como Visual Basic. Además Windows CE soporta C# a través del paquete .NET Compact Framework.

11. El programa se ejecuta bajo el sistema operativo Windows CE, debido a que el software fue desarrollado bajo el lenguaje de programación C#. El sistema operativo Windows CE tiene características similares a la plataforma Windows, facilitando el desarrollo de software para Windows CE.

12. Para establecer la comunicación con el agente SNMP del dispositivo monitoreado, se establecieron sockets hacia los puertos 161 para el envío y recepción de PDU tipo GET, GETNEXT, SET, Y RESPONSE y 162 para la recepción de las PDU tipo TRAP.

13. Para la codificación y decodificación de los paquetes SNMP se basó en el lenguaje de notación introducido por la ITU-T ASN 1 (Abstract Syntax Notacy) de acuerdo a las reglas de codificación básica BER, que definen un objeto de acuerdo a los campos tipo, longitud y valor.

14. En la configuración remota de equipos a través del browser, existe la desventaja que el contenido de configuración no encaja correctamente en la pantalla de la PDA, debido al tamaño limitado del display.

15. Para que la aplicación desarrollada pueda ejecutarse en la PDA con el sistema operativo Windows CE, fue necesario instalar el .NET Compact Framework que facilita las librerías necesarias para la ejecución de programas desarrollados en los lenguajes Visual Basic y C#.

16. Los equipos ingresados en el programa para monitorización, son guardados en un archivo de texto plano con los atributos principales como dirección IP, nombre de las comunidades, tipo de dispositivo, etc. para cuando el administrador ejecute el programa nuevamente los dispositivos no sean ingresados nuevamente.

17. En el ambiente de pruebas, donde se empleó la comunicación entre los equipos monitorizados y la PDA mediante comunicación inalámbrica Wireless, se empleó el protocolo WPA como mecanismo de seguridad.

18. Como un mecanismo de seguridad entre la PDA y los equipos monitorizados, en un ambiente de trabajo a través de Internet, se puede establecer una VPN encriptando la información entre los dispositivos involucrados en la monitorización.

19. El protocolo SNMP de la versión uno tiene problemas de seguridad ya que trabaja con la comunidad "public", mientras que en la versión SNMP dos se tendrá un cierto grado de seguridad autenticándose a través del nombre de la comunidad pública y privada.

4.2 RECOMENDACIONES

1. Es indispensable conocer los objetos identificadores de la MIB a explorar, ya que esta aplicación solamente permite realizar consultas y actualizaciones si se ingresa correctamente este valor.

2. El software desarrollado soporta los paquetes SNMP de la versión 1 y 2 como las PDU de tipo GET, GETNEXT, SET, RESPONSE y TRAP, permitiendo monitorizar equipos que soporten las versiones de este protocolo. Como un desarrollo adicional a este proyecto de titulación sería interesante desarrollarlo para que el software soporte SNMP V3.

3. En la actualidad se está comenzando a desarrollar aplicaciones más sofisticadas para los Smarth Phones, Palm y Pocket PC, debido al avance de procesamiento, memoria y sus sistemas operativos es estos. Lo mencionado en este párrafo trae como consecuencia una mayor movilidad de los usuarios, conservando aplicaciones similares a la de una PC de escritorio.

4. El programa podría ser utilizado para monitorear dispositivos remotos, siempre cuando la PDA tenga acceso a Internet. Actualmente las operadoras celulares como PORTA, ofrecen servicios de Internet mediante el servicio GPRS para los teléfonos celulares. La PDA puede utilizar el teléfono celular puede ser usado como MODEM para acceder a Internet.

BIBLIOGRAFÍA

LIBROS

- COMER Doglas, David L. Stevens, Interconectividad de redes con TCP/IP, Diseño e Implementación, Tercera Edición, Person Educación, Mexico 2000.

- COMER Doglas, Redes de Computadoras, Internet e Interrredes, A Simon & Schuster Company, Prentice Hall, primera edición, 1997

- RFC 1157; “A Simple Network Management Protocol (SNMP)”

- RFC 1441; “Introduction to SNMPv2”

- PFC 1443; “Textual Conventions for SNMPv2”

- RFC 1909; “An Administrative Infrastructure for SNMPv2”

- RFC 1910; “User-based Security Model for SNMPv2”

- RFC 1155; “Structure and Identification of Management Information for TCP/IP – based Internets”

- RFC 2578; “Structure of Management Information Version 2 (SMIv2)”

- RFC 1066; “Management Information Base for network management of TCP/IP-based internets”.

- RFC 1156; “Management Information Base for network management of TCP/IP-based internets”

- RFC 1212; “MIB Definitions”

- RFC 1213; “Management Information Base for Network Management of TCP/IP – based Internets”

- RFC 3641; “Generic String Encoding Rules (GSER) for ASN.1 Types”

- TANENBAUM Andrew, Redes de Computadoras”, Cuarta Edición. PrenticeHall, Mexico 2003.

DIRECCIONES ELECTRÓNICAS

- José Julio Ruiz. PDA Expertos.com. URL: http://www.pdaexpertos.com. 2008.

- PALM. PALM. URL: http://www.palm.com. 2007.

- Alejandro Mezcua Rodríguez. Instalación de Compact Framework en un dispositivo Windows Mobile. URL: http://www.byteabyte.net/utilidades/compactframe/default.aspx. 2007.

- Alejandro Mezcua Rodríguez. Ping para Compact Framework. URL: http://www.byteabyte.net/utilidades/pingcf/default.aspx. 2006.

- ARCESIO. Instalación Structure of Management Information (SMI). URL: http://www.arcesio.net/snmp/asn1.html. 2001.

- Vijay Mukhi’s Computer Institute. BER SNMP. URL: http://www.vijaymukhi.com/vmis/bersnmp.htm. 2000.

- Vijay Mukhi, Vinay Kalantri, Sonal Mukhi. C# Classes. URL: http://www.vijaymukhi.com/.

- Douglas Bruey Rane Corporation. SNMP: SIMPLE? NETWORK MANAGEMENT PROTOCOL. URL: http://www.rane.com/pdf/ranenotes/SNMP%20Simple%20Network%20Management%20Protocol.pdf. 2005.

- MSDN Library. Socket.Bind (Método). URL: http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/cpref/html/frlrfsystemnetsocketssocketclassbindtopic.asp. 2008.

Anexos

ANEXO A. SEGURIDAD EN REDES WLAN

WPA (Wifi Protect Access)

WPA es la abreviatura de Wifi Protect Access, y consiste en un mecanismo de control de acceso a una red inalámbrica. También se le conoce con el nombre de TSN (Transition Security Network).

WPA utiliza TKIP (Temporal Key Integrity Protocol) para la gestión de las claves dinámicas mejorando notablemente el cifrado de datos, incluyendo el vector de inicialización. En general WPA es TKIP con 802.1X. WPA funciona utilizando claves dinámicas, utiliza el algoritmo RC4 para generar un flujo de bits que se utilizan para cifrar con XOR y su vector de inicialización (IV) es de 48 bits.

Además WPA puede admitir diferentes sistemas de control de acceso incluyendo la validación de usuario-contraseña, certificado digital u otro sistema o simplemente utilizar una contraseña compartida para identificarse.

- **WPA-PSK**

WPA-PSK es un sistema que consiste en una clave común compartida, una gestión dinámica de claves. PSK se corresponde con las iniciales de PreShared Key, es decir, basa su seguridad en una contraseña compartida.

WPA-PSK usa una clave de acceso de una longitud entre 8 y 63 caracteres, que es la clave compartida. Esta clave hay que introducirla en cada una de las estaciones y puntos de acceso de la red inalámbrica. Cualquier estación que se identifique con esta contraseña, tiene acceso a la red.

Las características de WPA-PSK lo definen como el sistema, actualmente, más adecuado para redes de pequeñas oficinas o domésticas, la configuración es muy simple, la seguridad es aceptable y no necesita ningún componente adicional.

ANEXO B. CARACTERÍSTICAS DE EQUIPOS PDA

En este anexo se detallan las características generales de algunos equipos PDA tomados en cuenta para la implementación de la NMS sobre la PDA.

- **Dell Axim X5 PDA**

Características: 	
Descripción:	Axim X5 PDA
Fabricante:	Dell
Descripción Corta:	
Memoria Instalada:	64
Peso:	6.9
Sistema Operativo:	Microsoft Pocket PC 2002
Tipo de Memoria:	MMC Slot, Secure Digital (SD) Slot
Sistema:	
Bahía:	MMC Slot, Secure Digital (SD) Slot
Sistema Operativo:	Microsoft Pocket PC 2002
Processor Speed:	400 MHz
Memoria:	
Memoria ROM:	48 MB
Memoria Instalada:	64 MB

Display:	
Profundidad de Color:	65,536 Colors (16-bit)
Tipo de Pantalla:	Active Matrix LCD (TFT)
Tamaño de Pantalla:	3.5 in
Power:	
Fuente de Poder:	Lithium-Ion
Número de Baterías:	1
Tipo de Batería:	Lithium Ion
General:	
Funciones Incluídas:	Address Book, Calculator, Calendar, Phone Book, Scheduler
Aparatos Incluídos:	AC Adapter, Microphone, Speaker
Método de Entrada:	Touchscreen
Memoria Instalada Normalizada:	64
Tamaño Normalizado de Pantalla:	3.5
Pantalla Back-lit:	Yes
Dimensiones:	
Peso:	6.9 oz
Profundidad:	.71 in
Ancho:	3.21 in
Altura:	5.04 in

- **HP (Hewlett-Packard) iPAQ hx2490b PDA Organizador**

Características: 	
Descripción:	iPAQ hx2490b PDA
Fabricante:	HP (Hewlett-Packard)
Descripción Corta:	
Memoria Instalada:	64
Peso:	5.8
Sistema Operativo:	Windows Mobile 5
Tipo de Memoria:	MMC Slot, SDIO Card, Secure Digital (SD) Slot
Sistema:	
Bahía:	MMC Slot, SDIO Card, Secure Digital (SD) Slot
Conectores:	Audio - Headphone Out (2.5mm Mini), Serial - Infrared (IrDA-SIR)
Sistema Operativo:	Windows Mobile 5
Processor Speed:	520 MHz
Memoria:	

Memoria ROM:	128 MB
Memoria Instalada:	64 MB
Display:	
Profundidad de Color:	65,536 Colors (16-bit)
Tipo de Pantalla:	Transflective LCD (TFT)
Tamaño de Pantalla:	3.5 in
Power:	
Fuente de Poder:	Lithium-Ion
Tipo de Batería:	Lithium Ion
Networking:	
Soporte de Redes:	Wíreles Ethernet – 11 Mbps IEEE802.11b
General:	
Funciones Incluidas:	Audio Playback, E-mail, Spreadsheet, Video Playback, Voice Recording, Word Processor
Aparatos Incluidos:	AC Adapter, Carrying Case, Docking Cradle, Microphone, Speaker, USB Cable
Método de Entrada:	Control Pad, Stylus, Touchscreen

Tamaño Normalizado de Pantalla:	3.5
Memoria Instalada Normalizada:	64
Dimensiones:	
Peso:	5.8 oz
Profundidad:	0.65 in
Ancho:	4.71 in
Altura:	3.01 in

- **HP (Hewlett-Packard) iPAQ hx2495 Organizador PDA**

Características: 	
Descripción:	iPAQ hx2495 PDA
Fabricante:	HP (Hewlett-Packard)
Descripción Corta:	
Memoria Instalada:	64
Peso:	5.8
Sistema Operativo:	Windows Mobile 5
Tipo de Memoria:	Compact Flash Type II, MMC Slot, SDIO Card, Secure Digital (SD) Slot

Sistema:	
Bahía:	Compact Flash Type II, MMC Slot, SDIO Card, Secure Digital (SD) Slot
Conectores:	Audio - Headphone Out (1/8" Mini), Serial - Infrared (IrDA-SIR)
Sistema Operativo:	Windows Mobile 5
Processor Speed:	520 MHz
Memoria:	
Memoria ROM:	128 MB
Memoria Instalada:	64 MB
Display:	
Profundidad de Color:	65,536 Colors (16-bit)
Tipo de Pantalla:	Transflective LCD (TFT)
Tamaño de Pantalla:	3.5 in
Power:	
Fuente de Poder:	Lithium-Ion
Número de Baterías:	1
Tipo de Batería:	Lithium Ion
Networking:	
Soporte de Redes:	□íreles Ethernet – 11 Mbps IEEE802.11b
General:	
Funciones Incluídas:	Audio Playback, Integrated MP3 Player, Presentations, Spreadsheet,

	Video Playback, Web Browser, Word Processor
Aparatos Incluídos:	AC Adapter, Carrying Case, Docking Cradle, Microphone, Speaker
Método de Entrada:	Control Pad, Stylus, Touchscreen
Tamaño Normalizado de Pantalla:	3.5
Memoria Instalada Normalizada:	64
Dimensiones:	
Peso:	5.8 oz
Profundidad:	0.65 in
Ancho:	3.01 in
Altura:	4.71 in

- **Palm TX Organizador PDA**

Características: 	
Descripción:	TX Handheld PDA

Fabricante:	Palm
Descripción Corta:	
Memoria Instalada:	128
Peso:	5.25
Sistema Operativo:	Palm OS
Tipo de Memoria:	MMC Slot, SDIO Card, Secure Digital (SD) Slot
Sistema:	
Bahía:	MMC Slot, SDIO Card, Secure Digital (SD) Slot
Conectores:	Audio - Headphone Out (1/8" Mini)
Sistema Operativo:	Palm OS
Processor Speed:	312 MHz
Memoria.	
Memoria Instalada:	128 MB
Display:	
Profundidad de Color:	65,536 Colors (16-bit)
Tipo de Pantalla:	Transflective LCD (TFT)
Tamaño de Pantalla:	7 in
Power:	
Fuente de Poder:	Lithium-Ion
Tipo de Batería:	Lithium Ion
Networking:	
Soporte de Redes:	Wíreles Ethernet – 11 Mbps IEEE802.11b

General:	
Funciones Incluídas:	Audio Playback, Handwriting Recognition, Integrated MP3 Player, Presentations, Spreadsheet, Web Browser
Aparatos Incluídos:	AC Adapter, Speaker, USB Cable
Método de Entrada:	Stylus, Touchscreen
Tamaño Normalizado de Pantalla:	7
Memoria Instalada Normalizada:	128
Dimensiones:	
Peso:	5.25 oz
Profundidad:	0.61 in
Ancho:	3.08 in
Altura:	4.76 in

- **Palm Tungsten E2 PDA - Multilingual**

Características:

Descripción:	Tungsten E2 PDA - Multilingual
Fabricante:	Palm
Descripción Corta:	
Memoria Instalada:	32
Peso:	4.6949
Sistema Operativo:	Palm OS
Tipo de Memoria:	MMC Slot, SDIO Card
Sistema:	
Bahía:	MMC Slot, SDIO Card
Conectores:	Audio - Headphone Out (1/8" Mini), Serial - Infrared (IrDA-SIR), USB - Universal Serial Bus "B"
Sistema Operativo:	Palm OS
Processor Speed:	200 MHz
Memoria:	
Memoria Instalada:	32 MB
Display:	
Profundidad de Color:	65,536 Colors (16-bit)
Tipo de Pantalla:	Transflective LCD (TFT)
Power:	
Número de Baterías:	1
Tipo de Batería:	Lithium Ion, Rechargeable
Vida de Batería:	8 day(s)

General:	
Método de Entrada:	Control Pad, Stylus
Funciones Incluídas:	Address Book, Audio Playback, Bluetooth Compatibility, Calendar, E-mail, Expense Tracker, Handwriting Recognition, Note Pad, Presentations, Scheduler, Spreadsheet, Video Playback, Word Processor
Aparatos Incluídos:	AC Adapter, Speaker, Stylus
Power Options:	Rechargeable Battery
Memoria Instalada Normalizada:	32
Pantalla Back-lit:	Yes
Dimensiones:	
Peso:	133 g
Profundidad:	15 mm
Ancho:	78 mm
Altura:	114 mm

- **Toshiba e310 Color 32MB Pocket PC**

Características: 	
Descripción:	Pocket PC e310 PDA
Fabricante:	Toshiba
Descripción Corta:	
Memoria Instalada:	32
Peso:	4.9
Sistema Operativo:	Microsoft Pocket PC 2002
Sistema:	
Conectores:	Audio - Headphone Out (1/8" Mini), Serial - Infrared (IrDA-SIR)
Sistema Operativo:	Microsoft Pocket PC 2002
Processor Speed:	206 MHz
Memoria:	
Memoria de Video Instalada:	256 kB
Memoria ROM:	32 MB
Memoria Instalada:	32 MB
Display:	
Profundidad de Color:	65,536 Colors (16-bit)
Tipo de Pantalla:	Active Matrix LCD (TFT)

Tamaño de Pantalla:	3.5 in
Power:	
Fuente de Poder:	Lithium-Ion
Tipo de Batería:	Lithium Ion
Vida de Batería:	10 hour(s)
General:	
Método de Entrada:	Stylus, Touchscreen
Funciones Incluidas:	E-mail, Handwriting Recognition, Spreadsheet, Word Processor
Aparatos Incluidos:	AC Adapter
Tamaño Normalizado de Pantalla:	3.5
Power Options:	Rechargeable Battery
Memoria de Video Instalada Normalizada:	0.256
Memoria Instalada Normalizada:	32
Pantalla Back-lit:	Yes
Dimensiones:	
Peso:	4.9 oz
Profundidad:	0.4 in
Ancho:	3.1 in
Altura:	4.9 in

- **Sony Clie PEG-T615C/S Organizador PDA**

Características: 	
Descripción:	Clie PEG-T615C/S PDA
Fabricante:	Sony
Descripción Corta:	
Memoria Instalada:	16
Peso:	4.9
Sistema Operativo:	Palm OS
Tipo de Memoria:	Sony Memory Stick
Sistema:	
Bahía:	Sony Memory Stick
Conectores:	Proprietary, USB - Universal Serial Bus "A"
Sistema Operativo:	Palm OS
Memoria:	
Memoria ROM Instalada:	4 MB
Memoria Instalada:	16 MB
Display:	
Profundidad de Color:	65,536 Colors (16-bit)
Tipo de Pantalla:	Active Matriz LCD (TFT)
Power:	

Número de Baterías:	1
Tipo de Batería:	Lithium Ion
General:	
Docking:	Hot (Full Power)
Método de Entrada:	Keypad, Stylus
Funciones Incluidas:	Address Book, Audio Playback, Calculator, Calendar, E-mail, Note Pad, Phone Book, Scheduler
Aparatos Incluídos:	AC Adapter, Connectivity Cable, Docking Cradle, Installation Software, Stylus
Power Options:	AC Input, Rechargeable Battery
Memoria Instalada Normalizada:	16
Pantalla Back-lit:	Yes
Dimensiones:	
Peso:	4.9 oz
Profundidad:	0.5 in
Ancho:	2.83 in
Altura:	4.65 in

ANEXO C. MANUAL DE USUARIO

C.1 Requerimientos

Para la ejecución y funcionamiento del programa necesitamos los siguientes elementos:

- Una PDA con el sistema operativo WINDOWS CE.
- La PDA debe tener una tarjeta WIFI para la conexión inalámbrica con el Access Point.
- Un Access Point
- Tener instalado el .NET Compact Framework para WINDOWS CE.
- Equipos configurados el protocolo SNMP para la monitorización.

C.2 Configuración

Para la ejecución del programa es necesaria la configuración WLAN de la PDA y la instalación del .NET Compact Framework.

C.2.1 Configuración WLAN de la PDA

En esta sección se configurará una conexión inalámbrica desde la PDA hacia el Access Point, asignando una dirección IP estática a la PDA.

En la mayoría de PDA con sistemas operativos Windows CE poseen herramientas para la configuración de redes inalámbricas, a continuación se describirá cada uno de los pasos a seguir para dicho propósito:

- Antes de establecer la conexión hacia el Access Point, la tarjeta WIFI de la PDA debe estar activada.

- Una vez activada la tarjeta WIFI de la PDA, ingresamos al panel Settings y escogemos Connections.

Figura C.1: Panel de Conexiones

- Dentro de este panel seleccionamos Netwok Cards, para configurar la interfaz inalámbrica. Seleccionado Network Cards aparece el panel Configure Network Adapters, donde seleccionamos la conexión y su respectiva interfaz de red.

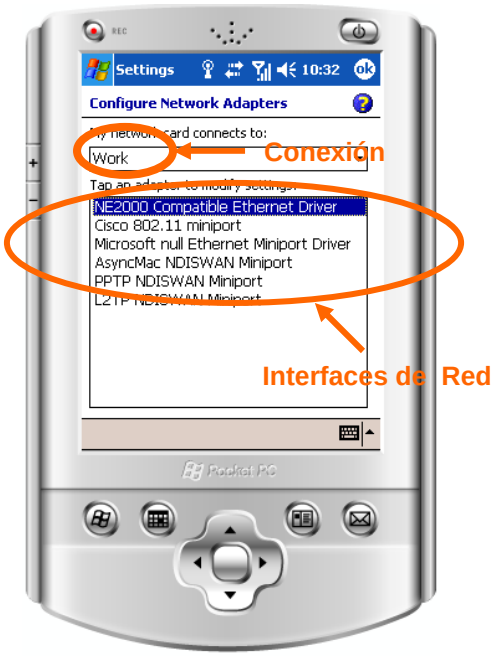


Figura C.2: Panel Configure Network Adapters

- Seleccionada la tarjeta de red podemos configurar la dirección IP y puertas de enlace para está conexión. En este panel podemos asignar una dirección IP estática o dinámica, en este caso es una dirección IP estática.

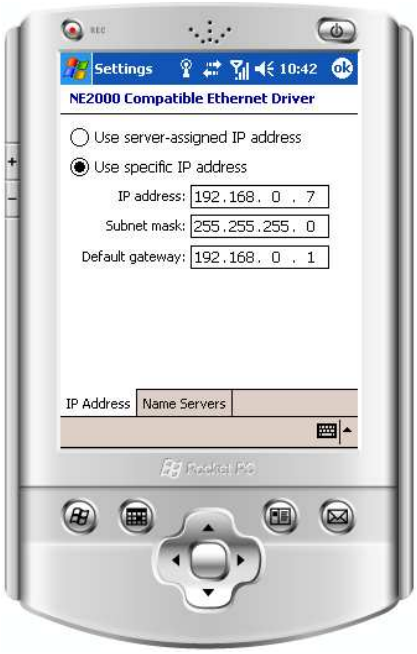


Figura C.3: Panel Configure Network Adapters

- Una vez configurada la dirección IP y guardada, aparecerá un mensaje donde nos indica que los cambios hechos al adaptador de red fueron realizados correctamente.

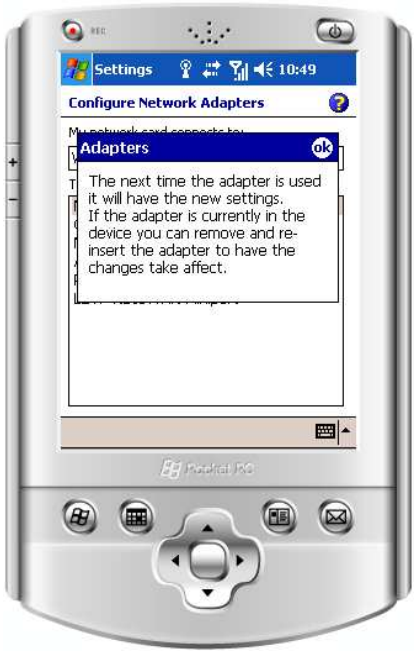


Figura C.4: Mensaje de configuración finalizada

C.2.2 Instalación .NET Compact Framework

.NET Compact Framework contiene una serie de librerías y archivos .dll que permiten ejecutar el programas desarrollados en Visual Studio .NET. En caso de no tener instalado podemos descargarlo de la siguiente dirección electrónica www.microsoft.com/downloads. A continuación se detalla brevemente la instalación de .NET Compact Fremework en la PDA.

- Sincronizamos la PDA con el computador de escritorio, generalmente la conexión PDA – computadora se realiza mediante cable USB.

- Para la sincronizar la PDA con el computador se hace uso del software Microsoft ActiveSync, este es un programa utilitario propio de las PDA.

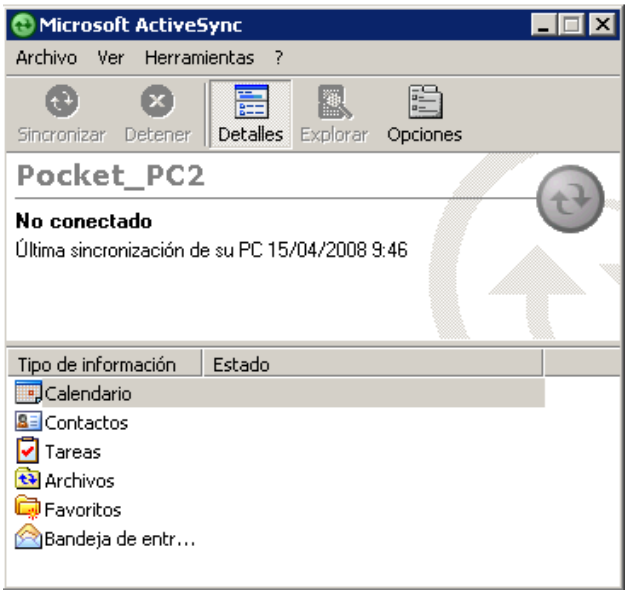


Figura C.5: Programa para Sincronizar PDA

- Dentro del menú Archivo del programa Microsoft ActiveSync pulsamos Sincronizar y siguiendo ciertas instrucciones podemos sincronizar la PDA.

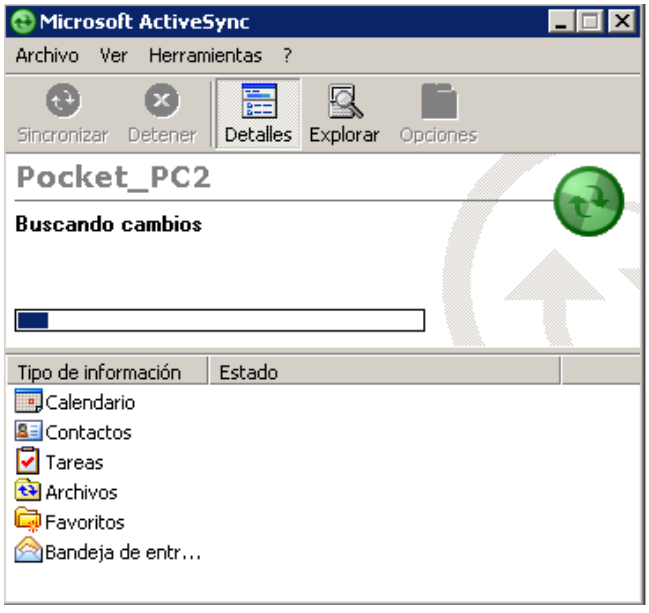


Figura C.6: Sincronizando PDA

- Sincronizada la PDA usamos el Explorador de Microsoft AcivSync para guardar el archivo .NET Compact Fremework en la PDA, este archivo debe tener una extensión .wm.armv4i para ser ejecutado e instalado en la PDA.

- Una ves guardado el archivo .NET Compact Fremework, podemos instalarlo en la PDA y proceder a ejecutar el programa desarrollado.

C.3 INSTALACIÓN DEL PROGRAMA

Igualmente como en la sección anterior sincronizamos la PDA para guardar el ejecutable del programa desarrollado y el archivo de configuración. El archivo de configuración contiene los equipos que van hacer monitoreados, este archivo también puede estar vacío en caso que se use por primera vez.

El archivo de configuración esta en formato de texto y debe ser guardado en el mismo directorio donde se ejecutará el programa para monitorización.

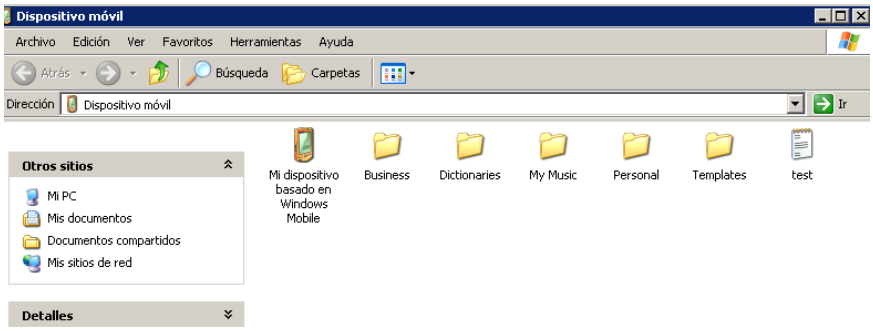


Figura C.7: Directorios principales de la PDA

C.4 USO DEL PROGRAMA

El software de monitorización de red para dispositivos inalámbricos consta de la ventana principal que se muestra en la figura C.8.

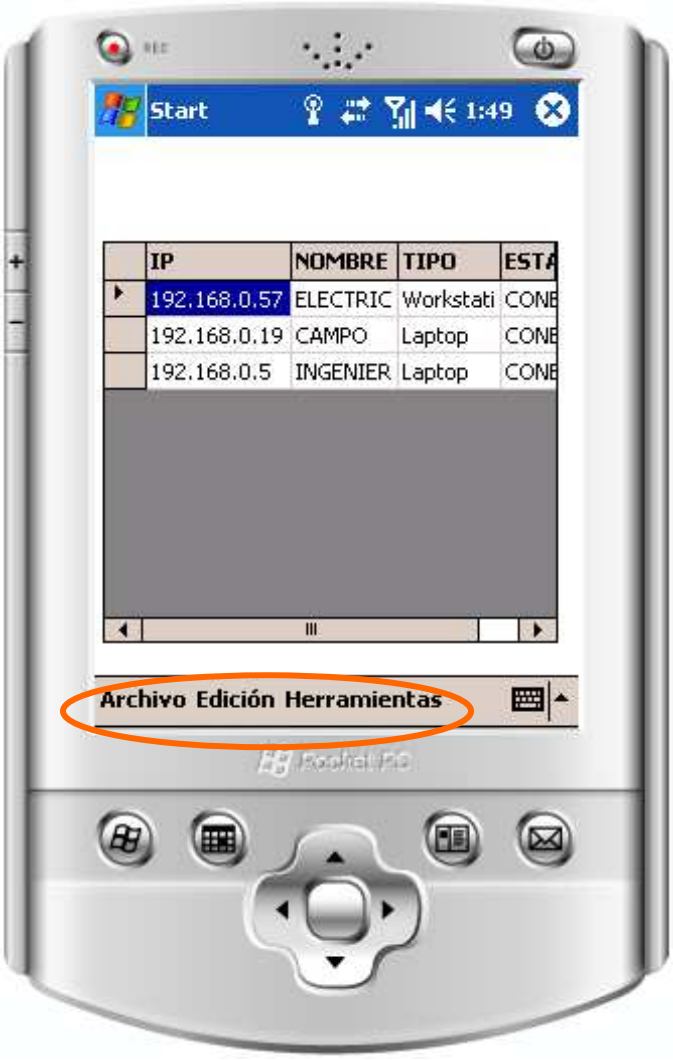


Figura C.8 Ventana Principal

La barra de menú consta de: Archivo, Edición y Herramientas.

En el Menú Archivo encontramos las opciones: Nuevo y Salir.

Al seleccionar la opción Nuevo se desplegará la pantalla que se muestra en la figura C.9. La misma que permite ingresar un nuevo dispositivo para ser

gestionado. Para esto ingresamos datos como: la dirección IP, El nombre del dispositivo, el grupo de trabajo o la red donde se encuentra, el tipo de dispositivo (Workstation, Server, Laptop, Printer, Switch, Router, Access Poit, etc), La comunidad de Lectura y la comunidad de escritura.

Para que el dispositivo sea guardado en el programa, es necesario que la dirección IP o su respectivo nombre sean válidos, caso contrario dicho dispositivo no podrá ser ingresado.

Figura C.9 Ventana para Agregar Dispositivos

Al seleccionar Salir, el programa dejará de ejecutarse y se cerrará.

En el Menú Edición tenemos las opciones de Edición y Borrar.

Al seleccionar Edición se desplegará la misma pantalla de Nuevo con la diferencia que me permite modificar los datos generales de los dispositivos gestionados a excepción de la dirección IP.

Figura C.10 Ventana para Editar Dispositivos

La opción Borrar permite eliminar el dispositivo del software de monitorización de red.

En el Menú Herramientas encontramos las opciones: SNMP, Ping y Browser.

- **SNMP:** Despliega la pantalla que se muestra en la figura C 11. En esta opción el usuario podrá explorar los objetos MIB de los dispositivos gestionados, para esto deberá ingresar el Objeto Identificador y seleccionar la operación que desea realizar como GET, GET NEXT o SET.

Figura C.11 Ventana Operaciones SNMP

- **Ping:** Comprueba el estado de la conexión con el dispositivo seleccionado, por medio de paquetes de solicitud de eco y respuesta de eco. Al seleccionar esta opción se despliega la pantalla que se muestra en la figura C.12.

Figura C.12 Ventana PING

- Browser: Ejecutará un browser con la dirección IP del dispositivo seleccionado para configurar el equipo (Esto para equipos que permiten configurar sus parámetros por medio del browser).

Figura C.13 Browser

ANEXO D. CODIFICACIÓN DE LOS PAQUETES SNMP

D.1 Paquetes SNMP

En ente anexo se muestra el proceso de codificación de un paquete SNMP, el cual posee el siguiente formato:

Versión	Comunidad	SNMP PDU

Figura D.1: Paquete SNMP

- **Versión:** Número de versión de protocolo que se está utilizando (por ejemplo 0 para SNMPv1);

- **Comunidad:** Nombre o palabra clave que se usa para la autenticación. Generalmente existe una comunidad de lectura llamada "public" y una comunidad de escritura llamada "private".

- **SNMP PDU:** Contenido de la unidad de datos del protocolo, el que depende de la operación que se ejecute.

La estructura de SNMP PDU depende del tipo de mensaje que se ejecute, entre estos tenemos:

Los mensajes GetRequest, GetNextRequest, SetRequest, GetResponse de la versión uno y 2c y TRAP de la versión 2c tienen la siguiente estructura en el campo SNMP PDU:

Tipo de PDU	Identificador	Estado de error	Índice de error	Variables Vinculadas

Figura D.2: Formato de las PDU GetRequest, GetNextRequest, SetRequest, GetResponse y TRAP 2c

- **Tipo de PDU:** Especificado por un número dependiendo si es GET(0), GETNEXT(1), RESPONCE(2), SET(3) y TRAP(7).

- **Identificador:** Es un número utilizado por el NMS y el agente para enviar solicitudes y respuesta diferentes en forma simultánea.
- **Estado de error**: Sólo se usan en los mensajes GetResponse cuando existe un error (en las consultas o mensajes GetRequest siempre se utiliza cero).

- **Índice de error:** sólo se usa cuando "estado de error" es distinto de 0 y posee el objetivo de proporcionar información adicional sobre la causa del problema. El campo "estado de error" puede tener los siguientes valores:

    - 0: No hay error;
    - 1: Demasiado grande;
    - 2: No existe esa variable;
    - 3: Valor incorrecto;
    - 4: El valor es de solo lectura;
    - 5: Error genérico.

- **Variables vinculadas o asociadas:** Es una serie de nombres de variables con sus valores correspondientes (codificados en ASN.1).

Para Trap del protocolo SNMP versión uno, el formato de la PDU es diferente:

Tipo de PDU	Enterprise	Dirección del agente	Tipo genérico de trap	Tipo específico de trap	Timestamp	Variables Vinculadas

Figura D.3: Formato de las PDU TRAP del protocolo SNMP versión uno

- **Tipo de PDU:** Especificado por un número, para TRAP 4.

- **Enterprise:** Identificación del subsistema de gestión que ha emitido el trap.

- **Dirección del agente:** Dirección IP del agente que ha emitido el trap.

- **Tipo genérico de trap:**

    - Cold start (0): Indica que el agente ha sido inicializado o reinicializado;
    - Warm start (1): Indica que la configuración del agente ha cambiado;
    - Link down (2): Indica que una interfaz de comunicación se encuentra fuera de servicio (inactiva);
    - Link up (3): Indica que una interfaz de comunicación se encuentra en servicio (activa);
    - Authentication failure (4): Indica que el agente ha recibido un requerimiento de un NMS no autorizado (normalmente controlado por una comunidad);
    - EGP neighbor loss (5): Indica que en sistemas en que los routers están utilizando el protocolo EGP, un equipo colindante se encuentra fuera de servicio;
    - Enterprise (6): En esta categoría se encuentran todos los nuevos traps incluidos por los vendedores.

- **Tipo específico de trap**: Es usado para traps privados (de fabricantes), así como para precisar la información de un determinado trap genérico.

- **Timestamp:** Indica el tiempo que ha transcurrido entre la reinicialización del agente y la generación del trap.

- **Variables Vinculadas o Asociadas:** Se utiliza para proporcionar información adicional sobre la causa del mensaje.

D.2 BER (Reglas de Codificación Básica)

Las reglas de codificación básica describen el método para codificar valores de cada tipo ASN.1 como una cadena de octetos. Los campos para la codificación son:

Figura D.4: Campos de un objeto

- **Tipo:** Define las características propias de la información, Además indica si la codificación es simple o estructurada.

- **Longitud:** Define el numero de octetos que tiene el campo valor.

- **Valor**: Representa el valor de tipo ASN.1 como una cadena de octetos.

D.3 Codificación BER para los paquetes SNMP GetRequest, GetNextRequest, Response, SetRequest y TRAP v2c.

La codificación de los paquetes SNMP Get, GetNext, Response, Set y TRAP v2c es similar, como ejemplo tomaremos un paquete SNMP GetRequest capturado mediante Ethereal.

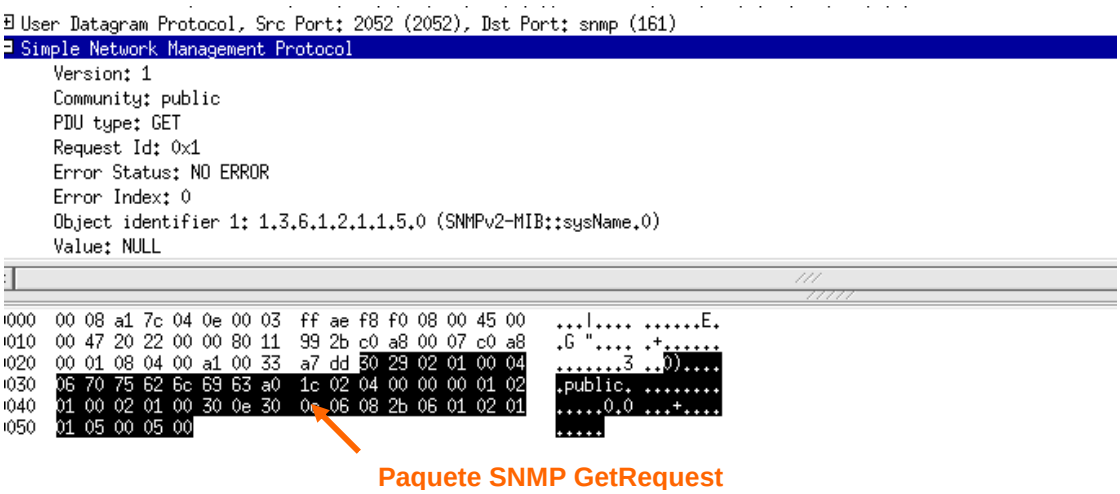


Figura D.5: Captura del paquete SNMP GetRequest mediante Ethereal

Como se observa en la figura D.5 el paquete SNMP GetRequest comienza y termina con valores $0x30_H$ y $0x00_H$.

D.3.1 Codificación del Mensaje SNMP

En la figura D.5 del paquete SNMP GetRequest comienza con los campos tipo y longitud.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	0	1	1	0	0	0	0
Universal		Codificación Estructurada	Sequence = 16				

Tabla D.1: Valor Binario

El valor del campo tipo en hexadecimal de acuerdo a la captura en Ethereal es $0x30_H$.

- **Longitud:** Número de octetos del campo valor:

0	0	1	0	1	0	0	1
41 Bytes							

Tabla D.2: Valor Binario

El valor del campo longitud en hexadecimal de acuerdo a la captura en Ethereal es $0x29_H$, esto nos indica que el paquete SNMP estará compuesto por 41 bytes + 2 bytes correspondientes a los campos tipo y longitud del paquete SNMP.

D.3.2 Codificación del Campo Versión

La captura del campo versión del paquete SNMP tiene un valor hexadecimal correspondiente a 02_H 01_H 00_H, como se observa en la figura D.6.

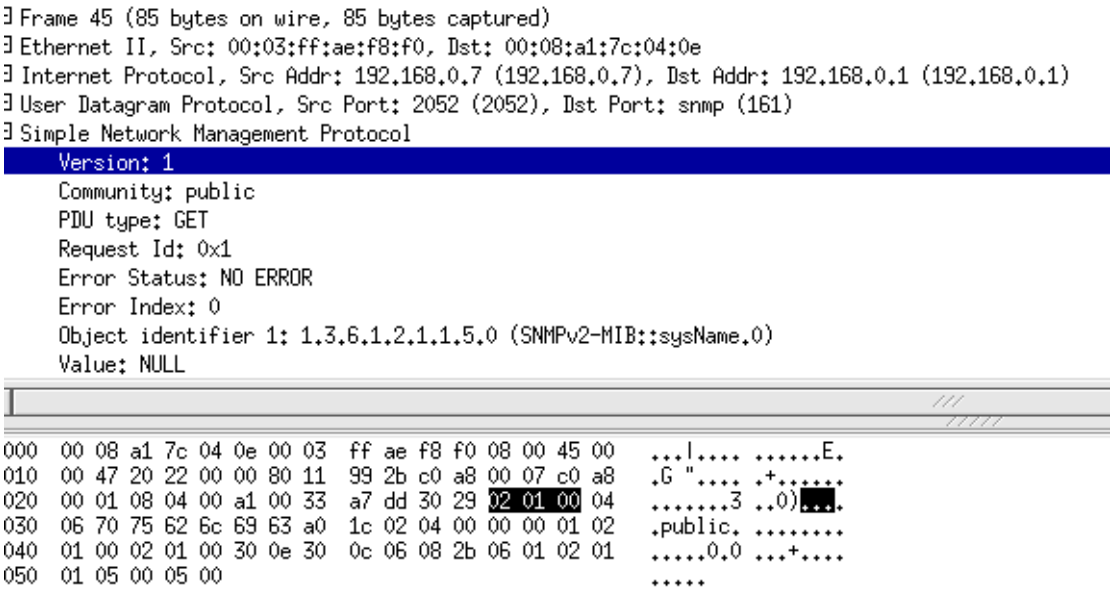


Figura D.6: Valor correspondiente al campo Version del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**
Universal		**Codificación Simple**	**Integer = 2**				

Tabla D.3: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	0	0	1
1 Byte							

Tabla D.4: Valor Binario

- **Valor:** Octetos del campo valor, correspondiente al protocolo SNMP versión uno:

0	0	0	0	0	0	0	0
0							

Tabla D.5: Valor Binario

En el caso del protocolo SNMP versión 2c el valor correspondiente a este campo es:

0	0	0	0	0	0	0	1
1							

Tabla D.6: Valor Binario

D.3.3 Codificación del Campo Comunidad

La captura del campo comunidad del paquete SNMP tiene un valor hexadecimal correspondiente a 04_H 06_H 70_H 75_H 62_H $6c_H$ 69_H 63_H, como se observa en la figura D.7.

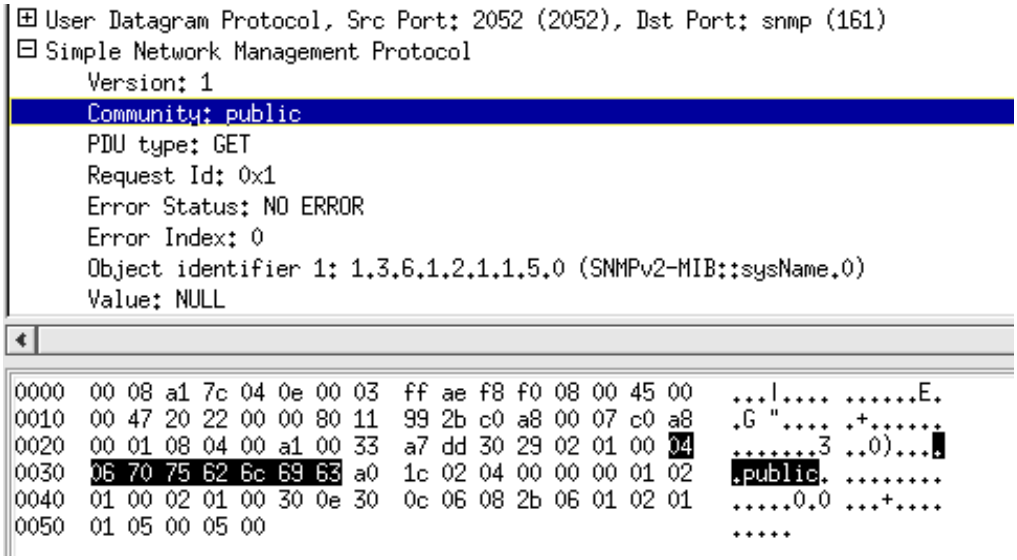


Figura D.7: Valor correspondiente al campo Comunity del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	0	0	0	0	1	0	0
Universal		Codificación Simple	Octet String = 4				

Tabla D.7: Valor Binario

Este campo es de tipo Octet String correspondiente a un valor de 4_H.

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	1	1	0
6 Bytes							

Tabla D.8: Valor Binario

- **Valor:** Octetos del campo valor:

0	1	1	1	0	0	0	0
P							

0	1	1	1	0	1	0	1
U							

0	1	1	0	0	0	1	0
B							

0	1	1	0	1	1	0	0
L							

0	1	1	0	1	0	0	1
I							

0	1	1	0	0	0	1	1
C							

Tabla D.9: Valor Binario

En este caso se observa que el valor de la Comunidad del paquete SNMP GetRequest corresponde a “public”.

D.3.4 Codificación del Campo SNMP PDU

La captura del campo tipo de PDU del paquete SNMP tiene un valor hexadecimal correspondiente a $a0_H$ $1c_H$, como se observa en la figura D.8. Donde $a0_H$ y $1c_H$ nos indica el tipo y longitud de la PDU respectivamente, correspondiente a la PDU GetRequest.

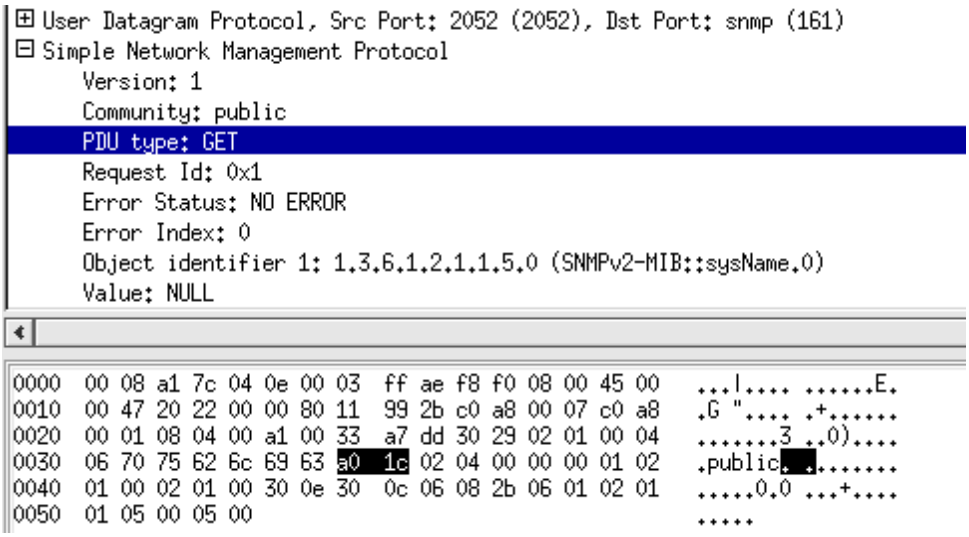


Figura D.8: Valor correspondiente al campo Tipo de PDU del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
1	**0**	**1**	**0**	**0**	**0**	**0**	**0**
Contexto Específico		**Codificación Estructurada**	**GetRequest = 0**				

Tabla D.10: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	1	1	1	0	0
25 Bytes							

Tabla D.11: Valor Binario

D.3.5 Codificación del Campo Identificador

La captura del campo identificador del paquete SNMP, tiene un valor hexadecimal correspondiente a 02_H 04_H 00_H 00_H 00_H 01_H, como se observa en la figura D.9.

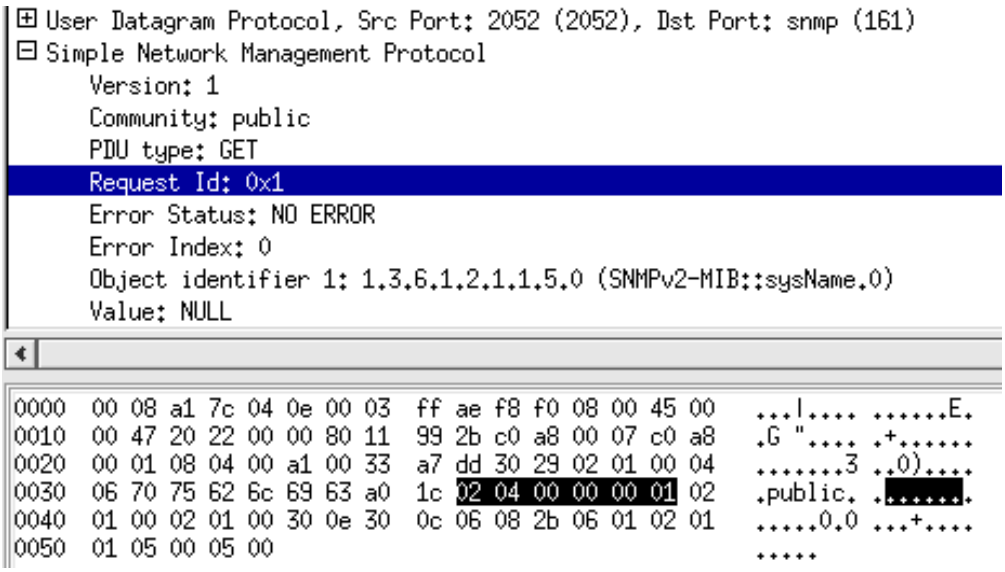


Figura D.9: Valor correspondiente al campo Identificador de la PDU del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**
Universal		**Codificación Simple**	**Integer = 2**				

Tabla D.12: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	1	0	0
1 Byte							

Tabla D.13: Valor Binario

- **Valor:** Octetos del campo valor:

0	0	0	0	0	0	0	0
0							

0	0	0	0	0	0	0	0
0							

0	0	0	0	0	0	0	0
0							

0	0	0	0	0	0	0	1
1							

Tabla D.14: Valor Binario

Como se observa en la codificación el identificador para este paquete SNMP es igual a uno, el valor de este campo puede tener una longitud de 4 bytes.

D.3.6 Codificación del Campo Estado de Error

La captura del campo estado de error del paquete SNMP, tiene un valor hexadecimal correspondiente a 02_H 01_H 00_H, como se observa en la figura D.10.

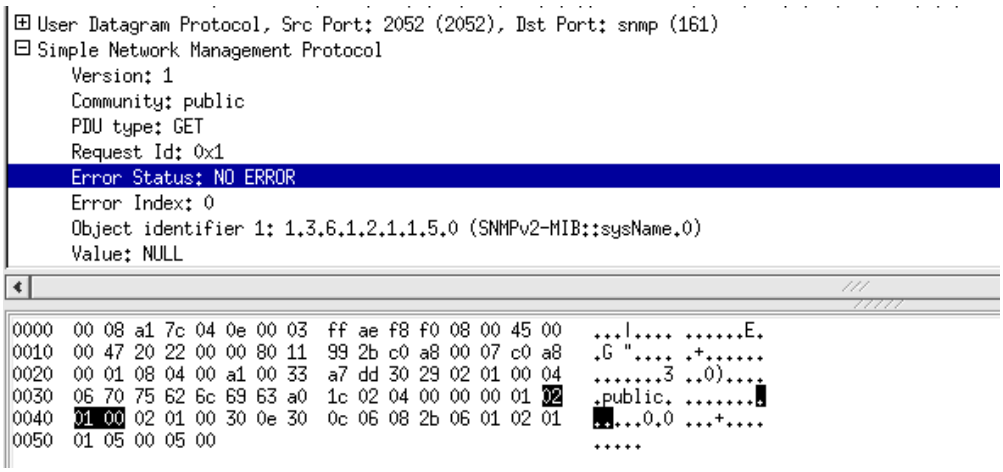


Figura D.10: Valor correspondiente al campo estado de error del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	0	0	0	0	0	1	0
Universal		Codificación Simple	Integer = 2				

Tabla D.15: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	0	0	1
1 Byte							

Tabla D.16: Valor Binario

- **Valor:** Octetos del campo valor:

0	0	0	0	0	0	0	0
0							

Tabla D.17: Valor Binario

El valor igual a cero de este campo, nos indica que no existe ningún error en este paquete.

D.3.7 Codificación del Campo Índice de Error

La captura del campo índice de error estado de error del paquete SNMP, tiene un valor hexadecimal correspondiente a 02_H 01_H 00_H, como se observa en la figura D.11.

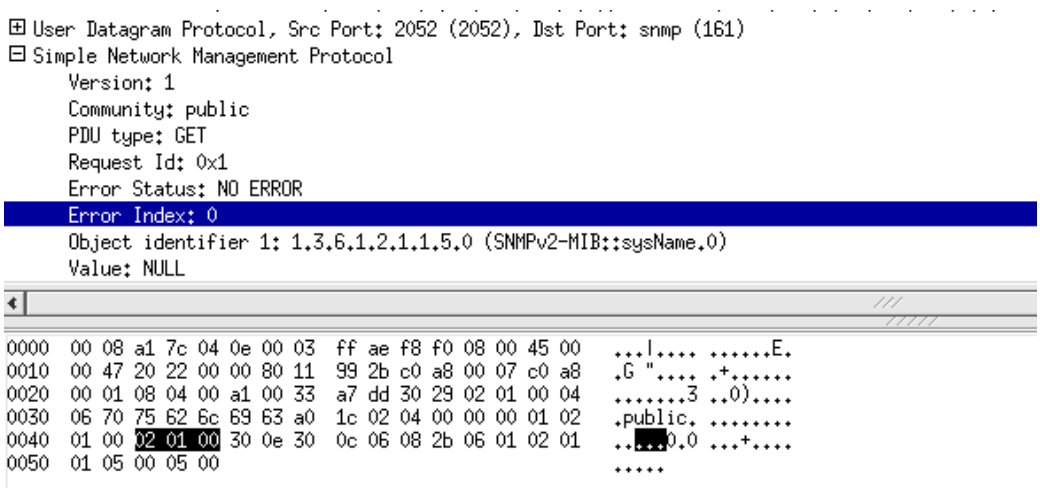


Figura D.11: Valor correspondiente al indice de error del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**
Universal		**Codificación Simple**	**Integer = 2**				

Tabla D.18: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	0	0	1
1 Byte							

Tabla D.19: Valor Binario

- **Valor:** Octetos del campo valor:

0	0	0	0	0	0	0	0
0							

Tabla D.20: Valor Binario

En este caso como no existe error, el valor del índice de error es igual a cero, caso contrario nos da la ubicación del campo donde existe un error.

D.3.8 Codificación del Campo Variables Vinculadas o Asociadas

La captura del campo variables vinculadas del paquete SNMP, tiene un valor hexadecimal correspondiente a 30_H $0e_H$, como se observa en la figura D.12.

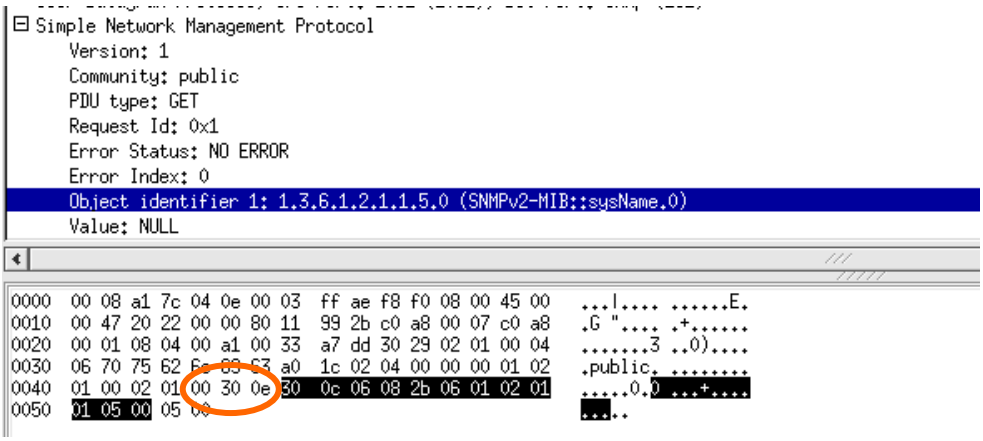


Figura D.12: Valor correspondiente al campo variables vinculadas del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	0	1	1	0	0	0	0
Universal		Codificación Estructurada	Sequence = 16				

Tabla D.21: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	1	1	1	0
14 Bytes							

Tabla D.22: Valor Binario

- **Valor:** Campo Varbind.

Para este campo su valor corresponde al campo Varbind, el cual contiene los objetos identificadores.

D.3.9 Codificación del Campo Varbind

La captura del campo varbind del paquete SNMP, tiene el valor del objeto identificador que se va ha consultar mediante el paquete SNMP GetRequest.

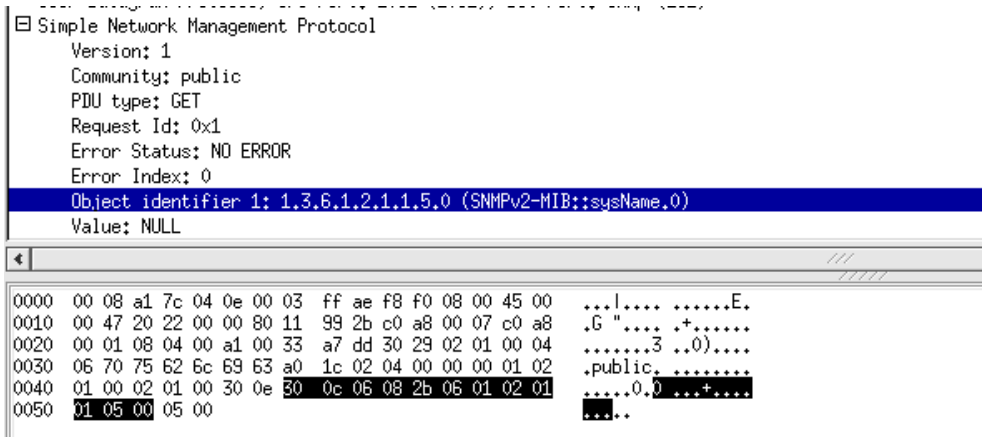


Figura D.13: Valor correspondiente al campo varbind del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	**0**	**1**	**1**	**0**	**0**	**0**	**0**
Universal		**Codificación Estructurada**	**Sequence = 16**				

Tabla D.23: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	1	1	0	0
12 Bytes							

Tabla D.24: Valor Binario

- **Valor:** Campo Object Identifier.

El valor de este campo corresponde al objeto identificador 1.3.6.1.2.1.1.5.0 SysName:0.

D.3.10 Codificación del Campo Object Identifier

La captura del campo objeto identificador del paquete SNMP, tiene el valor del objeto identificador que se va ha consultar mediante el paquete SNMP GetRequest, cuyo valor es 06_H 08_H $2b_H$ 06_H 01_H 02_H 01_H 01_H 05_H 00_H.

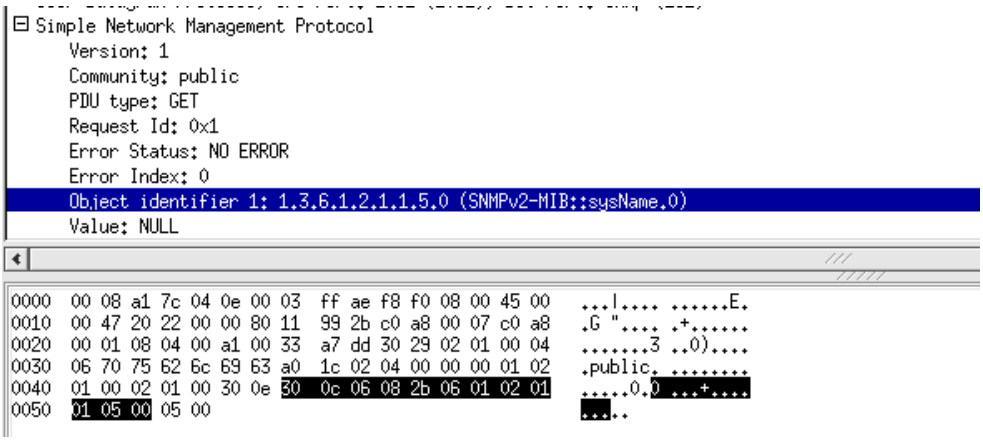


Figura D.14: Valor correspondiente al campo objeto identificador del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
0	**0**	**0**	**0**	**0**	**1**	**1**	**0**
Universal		**Codificación Simple**	**Object Identifier = 6**				

Tabla D.25: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	1	0	0	0
8 Bytes							

Tabla D.26 Valor Binario

- **Valor:** Octetos del campo valor:

0	0	1	0	1	0	1	1
40*1+3 = 43 1 = iso 3 = org							

0	0	0	0	0	1	1	0
6 = dod							

0	0	0	0	0	0	0	1
1 = internet							

0	0	0	0	0	0	1	0
2 = mgmt							

0	0	0	0	0	0	0	1
1 = mib-2							

0	0	0	0	0	0	0	1
1 = system							

0	0	0	0	0	0	0	1
5 = sysname							

0	0	0	0	0	0	0	0
0							

Tabla D.27 Valor Binario

D.3.11 Codificación del Campo Value

La captura del campo valor del paquete SNMP es igual a 05_H 00_H, este valor es por defecto para los paquetes SNMP GET y GETNEXT, en el caso de los

paquetes SNMP SET, RESPONSE y TRAP(2c) corresponde al valor del objeto identificador de la MIB.

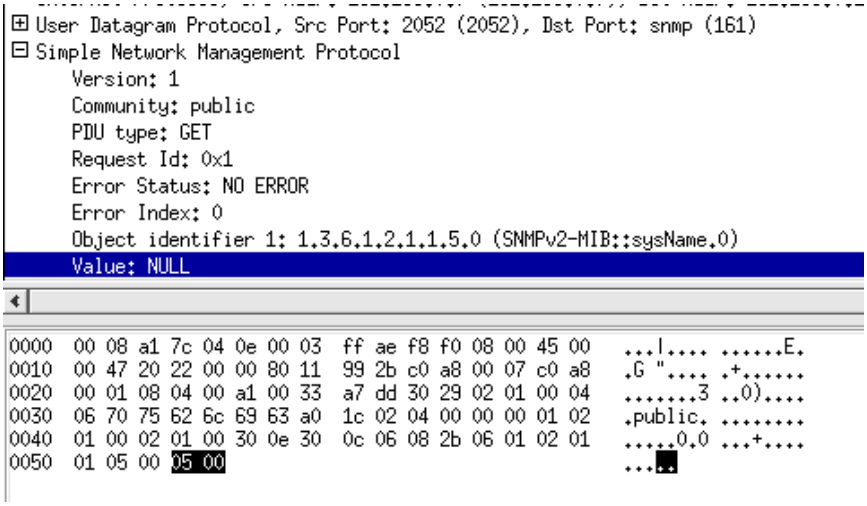


Figura D.15: Valor del paquete SNMP GetRequest.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
0	**0**	**0**	**0**	**0**	**1**	**0**	**1**
Universal		**Codificación Simple**	**Null = 5**				

Tabla D.28 Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	0	0	0
0 Bytes							

Tabla D.29 Valor Binario

En el caso de los paquetes SNMP SET, RESPONSE y TRAP(2c) corresponde al valor del objeto identificador de la MIB.

D.4 Codificación BER para los paquetes SNMP TRAP de la versión uno

La codificación del paquetes SNMP TRAP de la versión uno, es diferente a la codificación de los paquetes SNMP Get, GetNext, Set, Response y TRAP (2c) del punto analizado anteriormente.

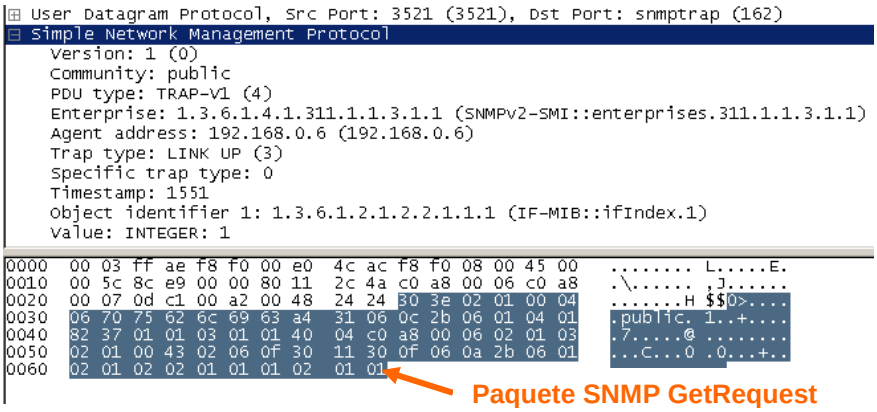


Figura D.16: Captura del paquete SNMP TRAP mediante Ethereal

Como se observa en la figura D.16 el paquete SNMP TRAP comienza con el valor $0x30_H$.

D.4.1 Codificación del Mensaje SNMP

En la figura D.16 del paquete SNMP TRAP comienza con los campos tipo y longitud.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	0	1	1	0	0	0	0
Universal		Codificación Estructurada	Sequence = 16				

Tabla D.30: Valor Binario

El valor del campo tipo en hexadecimal de acuerdo a la captura en Ethereal es $0x30_H$.

- **Longitud:** Número de octetos del campo valor:

0	0	1	1	1	1	1	0
62 Bytes							

Tabla C.31: Valor Binario

El valor del campo longitud en hexadecimal de acuerdo a la captura en Ethereal es $0x3e_H$, esto nos indica que el paquete SNMP estará compuesto por 62 bytes + 2 bytes correspondientes a los campos tipo y longitud del paquete SNMP.

D.4.2 Codificación del Campo Versión

La captura del campo versión del paquete SNMP tiene un valor hexadecimal correspondiente a 02_H 01_H 00_H, como se observa en la figura D.17.

```
⊞ User Datagram Protocol, Src Port: 3521 (3521), Dst Port: snmptrap (162)
⊟ Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: TRAP-V1 (4)
    Enterprise: 1.3.6.1.4.1.311.1.1.3.1.1 (SNMPv2-SMI::enterprises.311.1.1.3.1.1)
    Agent address: 192.168.0.6 (192.168.0.6)
    Trap type: LINK UP (3)
    Specific trap type: 0
    Timestamp: 1551
    Object identifier 1: 1.3.6.1.2.1.2.2.1.1.1 (IF-MIB::ifIndex.1)
    Value: INTEGER: 1

0000  00 03 ff ae f8 f0 00 e0  4c ac f8 f0 08 00 45 00   ........ L.....E.
0010  00 5c 8c e9 00 00 80 11  2c 4a c0 a8 00 06 c0 a8   .\...... ,J......
0020  00 07 0d c1 00 a2 00 48  24 24 30 3e 02 01 00 04   .......H $$0>....
0030  06 70 75 62 6c 69 63 a4  31 06 0c 2b 06 01 04 01   .public. 1..+....
0040  82 37 01 01 03 01 01 40  04 c0 a8 00 06 02 01 03   .7.....@ ........
0050  02 01 00 43 02 06 0f 30  11 30 0f 06 0a 2b 06 01   ...C...0 .0...+..
0060  02 01 02 02 01 01 01 02  01 01                     ........ ..
```

Figura D.17: Valor correspondiente al campo Versión del paquete SNMP TRAP.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**
Universal		**Codificación Simple**	**Integer = 2**				

Tabla D.32: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	0	0	1
1 Byte							

Tabla D.33: Valor Binario

- **Valor:** Octetos del campo valor, correspondiente al protocolo SNMP versión uno:

0	0	0	0	0	0	0	0
0							

Tabla D.34: Valor Binario

En el caso del protocolo SNMP versión 2c el valor correspondiente a este campo es:

0	0	0	0	0	0	0	1
1							

Tabla D.35: Valor Binario

D.4.3 Codificación del Campo Comunidad

La captura del campo comunidad del paquete SNMP tiene un valor hexadecimal correspondiente a 04_H 06_H 70_H 75_H 62_H $6c_H$ 69_H 63_H, como se observa en la figura D.18.

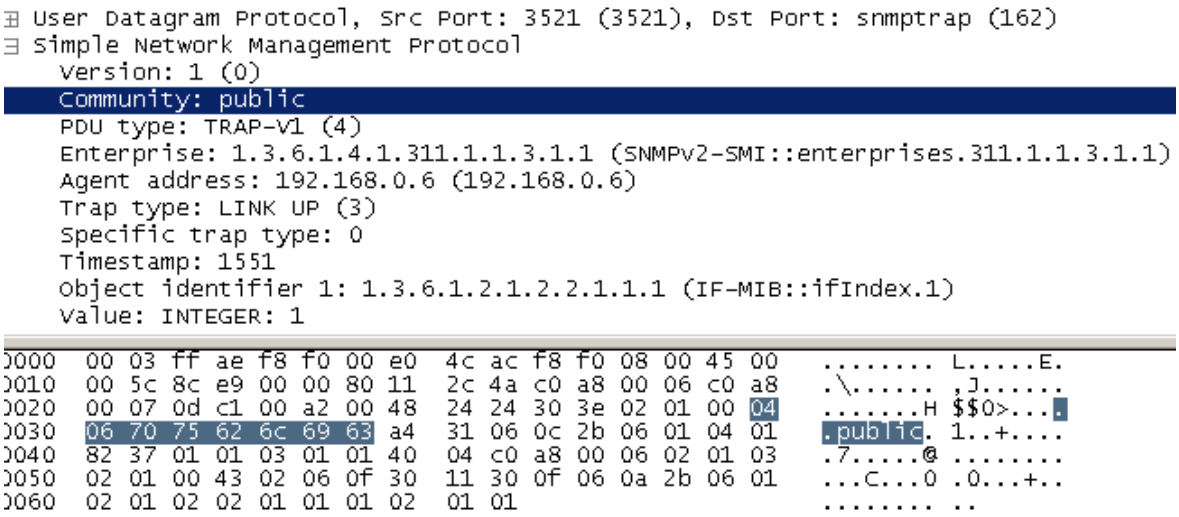


Figura D.18: Valor correspondiente al campo Comunity del paquete SNMP TRAP.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	0	0	0	0	1	0	0
Universal		Codificación Simple	Octet String = 4				

Tabla D.36: Valor Binario

Este campo es de tipo Octet String correspondiente a un valor de 4_H.

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	1	1	0
6 Bytes							

Tabla D.37: Valor Binario

- **Valor:** Octetos del campo valor:

0	1	1	1	0	0	0	0
P							

0	1	1	1	0	1	0	1
U							

0	1	1	0	0	0	1	0
B							

0	1	1	0	1	1	0	0
L							

0	1	1	0	1	0	0	1
I							

0	1	1	0	0	0	1	1
C							

Tabla D.38: Valor Binario

En este caso se observa que el valor de la Comunidad del paquete SNMP TRAP corresponde a “public”.

D.4.4 Codificación del Campo SNMP PDU

La captura del campo tipo de PDU del paquete SNMP tiene un valor hexadecimal correspondiente a $a4_H$ 31_H, como se observa en la figura D.19. Donde $a0_H$ y 31_H nos indica el tipo y longitud de la PDU respectivamente, correspondiente a la PDU TRAP.

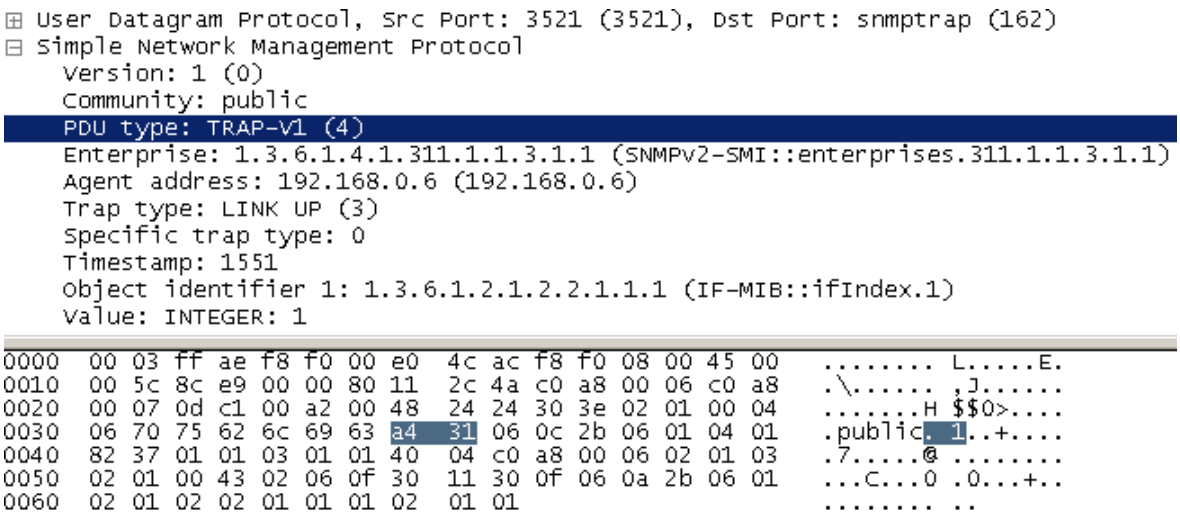


Figura D.19: Valor correspondiente al campo Tipo de PDU del paquete SNMP TRAP.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
1	0	1	0	0	1	0	0
Contexto Específico		Codificación Estructurada	TRAP = 4				

Tabla D.39: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	1	1	0	0	0	1
49 Bytes							

Tabla D.40: Valor Binario

D.4.5 Codificación del Campo Enterprise

La captura del campo Enterprise del paquete SNMP TRAP, tiene el objeto identificador del subsistema de gestión que ha emitido el TRAP, en este caso se observa que se trata de un equipo Windows correspondiente al objeto identificador 1.3.6.1.4.1.311.

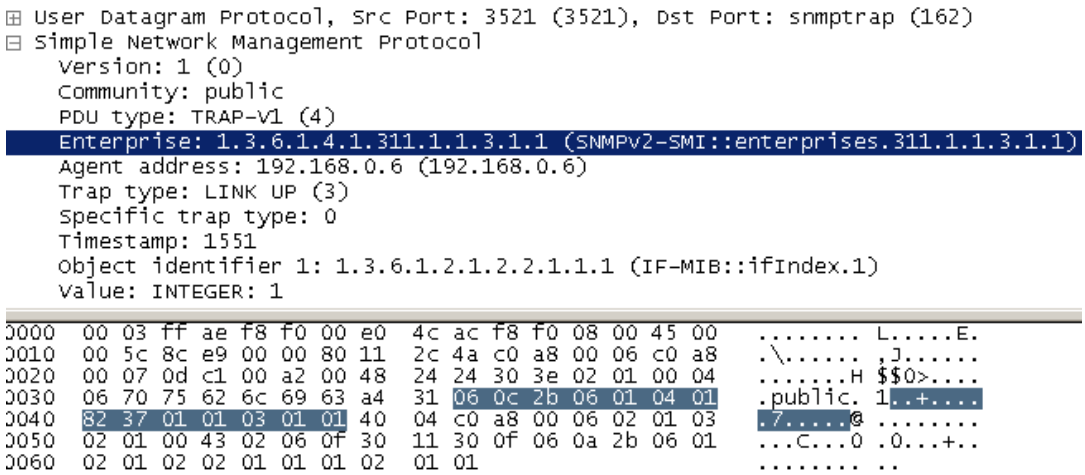


Figura D.20: Valor correspondiente al campo Enterprise del paquete SNMP TRAP

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	**0**	**0**	**0**	**0**	**1**	**1**	**0**
Universal		**Codificación Simple**	**Object Identifier = 6**				

Tabla D.41: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	1	1	0	0
12 Bytes							

Tabla D.42 Valor Binario

- **Valor:** Octetos del campo valor:

0	0	1	0	1	0	1	1
40*1+3 = 43 1 = iso 3 = org							

0	0	0	0	0	1	1	0
6 = dod							

0	0	0	0	0	0	0	1
1 = internet							

0	0	0	0	0	1	0	0
4							

0	0	0	0	0	0	0	1
1							

1	0	0	0	0	0	1	0
82							

0	0	1	1	0	1	1	1
37							

0	0	0	0	0	0	0	1
1							

0	0	0	0	0	0	0	1
1							

0	0	0	0	0	0	1	1
3							

0	0	0	0	0	0	0	1
1							

0	0	0	0	0	0	0	1
1							

Tabla D.43 Valor Binario

D.4.6 Codificación del Campo Agent Address

La captura del campo Agent Address del paquete SNMP tiene un valor hexadecimal correspondiente a 40_H 04_H $c0_H$ $a8_H$ 00_H 06_H, como se observa en la figura D.21. Este campo nos da la dirección del agente SNMP que emitió el TRAP.

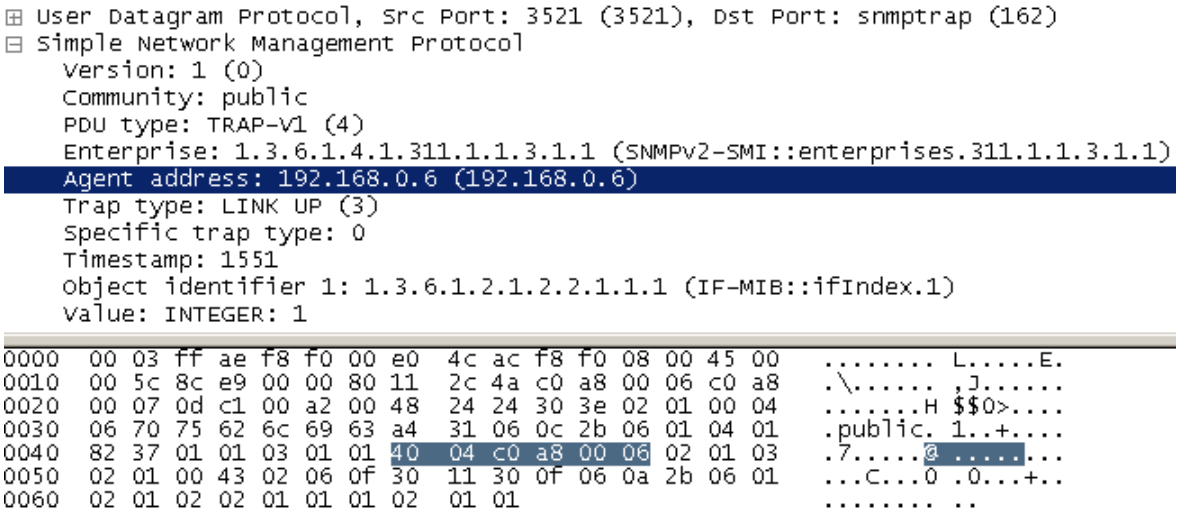


Figura D.21: Valor correspondiente al campo Agent Address del paquete SNMP TRAP.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**
Aplicación		**Codificación Simple**	**IPAddress = 0**				

Tabla D.44: Valor Binario

Este campo es de tipo IPAddress correspondiente a un valor de 40_H.

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	1	0	0
4 Bytes							

Tabla D.45: Valor Binario

- **Valor:** Octetos del campo valor:

1	1	0	0	0	0	0	0
192							

1	0	1	0	1	0	0	0
168							

0	0	0	0	0	0	0	0
0							

0	0	0	0	0	1	1	0
6							

Tabla C.46: Valor Binario

Para este ejemplo la dirección IP del agente SNMP que emitió el TRAP es 192.168.0.6.

D.4.7 Tipo de TRAP

La captura del campo tipo de TRAP del paquete SNMP, tiene un valor hexadecimal correspondiente a 02_H 01_H 03_H, como se observa en la figura D.22. En este caso se trata de una TRAP tipo LINK UP.

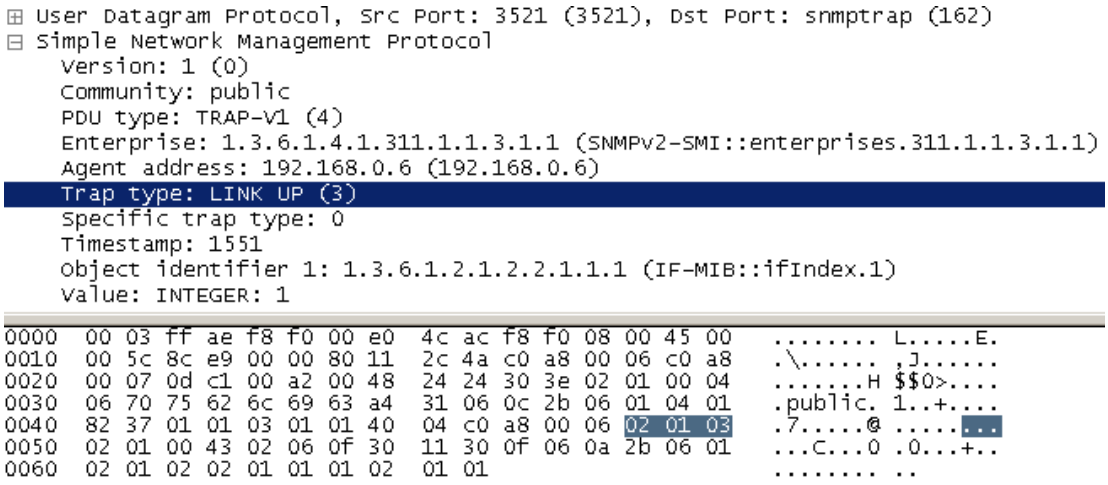


Figura D.22: Valor correspondiente al campo tipo de TRAP

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**
Universal		**Codificación Simple**	**Integer = 2**				

Tabla D.47: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	0	0	1
1 Byte							

Tabla D.48: Valor Binario

- **Valor:** Octetos del campo valor:

0	0	0	0	0	0	1	1
3							

Tabla D.49: Valor Binario

Como se observa en la codificación se trata de una TRAP tipo LINK UP.

D.4.8 Codificación del Campo Tipo de TRAP Específica

Utilizado para traps privados (de fabricantes), así como para precisar la información de un determinado trap genérico. La captura del campo tipo de trap específica del paquete SNMP, tiene un valor hexadecimal correspondiente a 02_H 01_H 00_H, como se observa en la figura D.23.

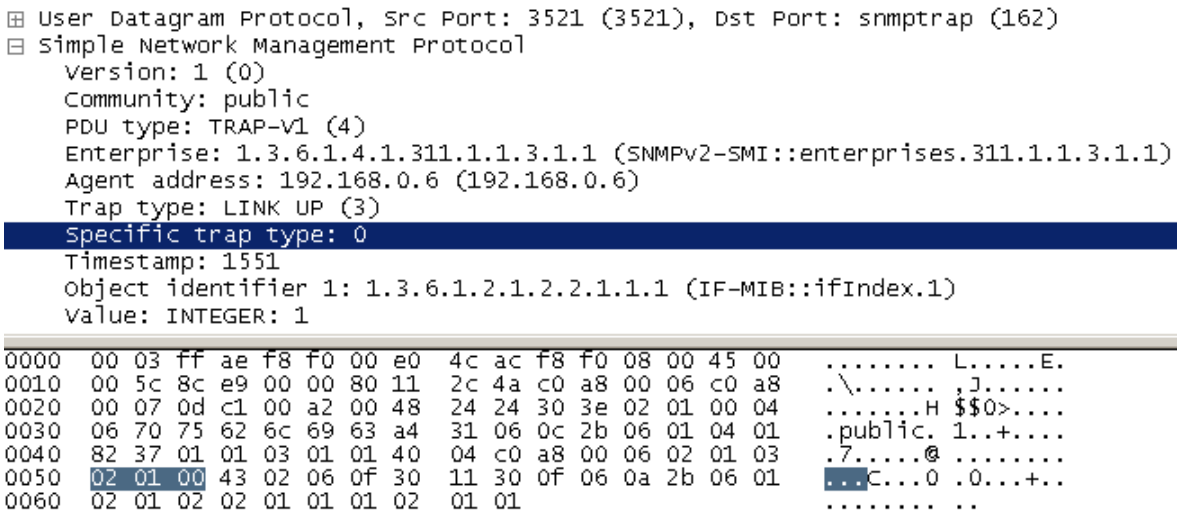


Figura D.23: Valor correspondiente al campo tipo de TRAP específica

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**
Universal		**Codificación Simple**	**Integer = 2**				

Tabla D.50: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	0	0	1
1 Byte							

Tabla D.51: Valor Binario

- **Valor:** Octetos del campo valor:

0	0	0	0	0	0	0	0
0							

Tabla D.52: Valor Binario

El valor igual a cero de este campo, nos indica que no existe ningún TRAP específico.

D.4.9 Codificación del Campo TimeStamp

La captura del campo índice TimeStamp del paquete SNMP, tiene un valor hexadecimal correspondiente a 43_H 02_H 06_H $0f_H$, como se observa en la figura D.24.

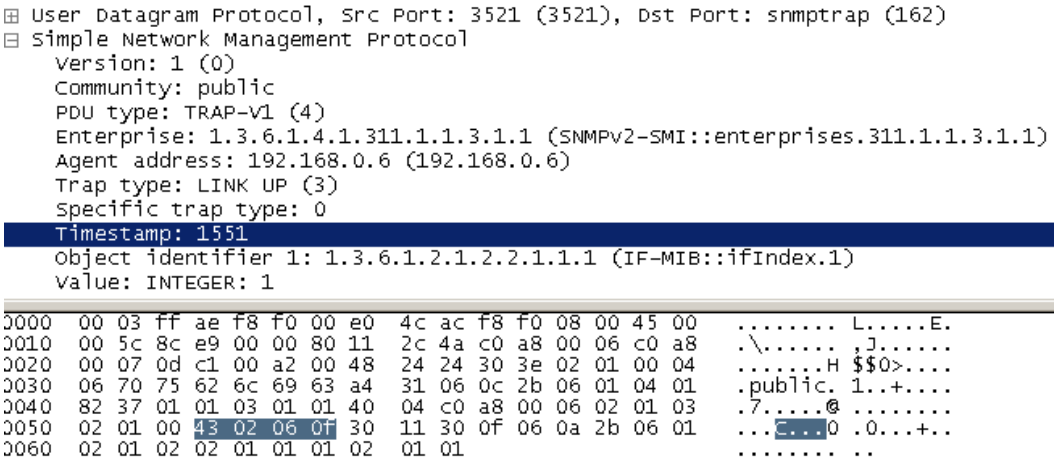


Figura D.24: Valor TimeStamp del paquete SNMP TRAP.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
0	**1**	**0**	**0**	**0**	**0**	**1**	**1**
Aplicación		**Codificación Simple**	**TimeTicks**				

Tabla D.53: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	0	1	0
2 Byte							

Tabla D.54: Valor Binario

- **Valor:** Octetos del campo valor:

0	0	0	0	0	1	1	0
6							

0	0	0	0	1	1	1	1
15							

Tabla D.55: Valor Binario

En este caso el tiempo que ha transcurrido entre la reinicialización del agente y la generación del trap es de 1551 milisegundos.

D.4.10 Codificación del Campo Variables Vinculadas o Asociadas

La captura del campo variables vinculadas del paquete SNMP, tiene un valor hexadecimal correspondiente a 30_H 11_H, como se observa en la figura D.25.

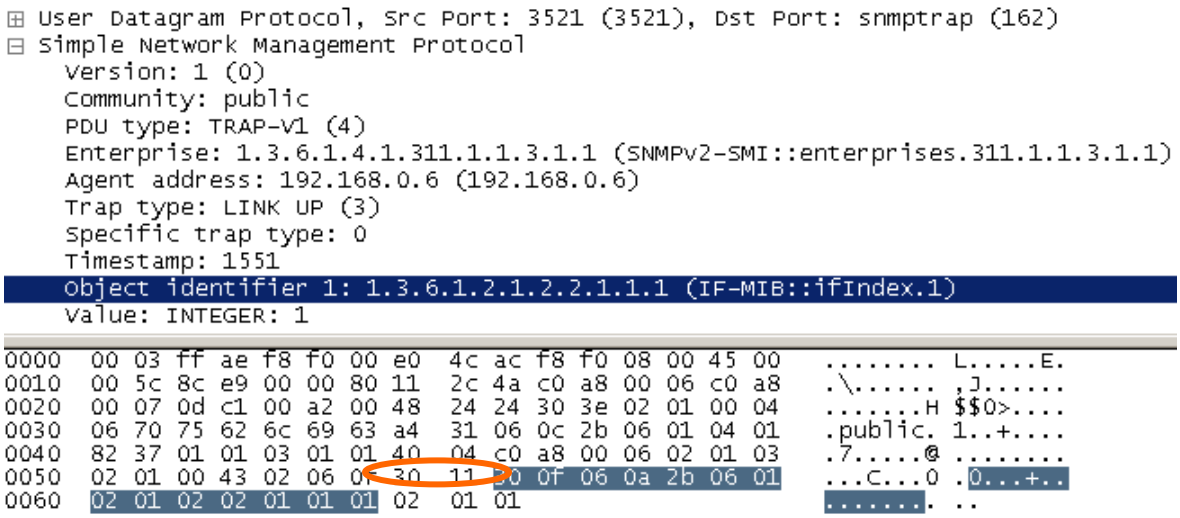


Figura D.25: Valor correspondiente al campo variables vinculadas del paquete SNMP TRAP.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
0	**0**	**1**	**1**	**0**	**0**	**0**	**0**
Universal		**Codificación Estructurada**	**Sequence = 16**				

Tabla D.56: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	1	0	0	0	1
17 Bytes							

Tabla D.57: Valor Binario

- **Valor:** Campo Varbind.

Para este campo su valor corresponde al campo Varbind, el cual contiene los objetos identificadores.

D.4.11 Codificación del Campo Varbind

La captura del campo varbind del paquete SNMP, tiene el valor del objeto identificador que envió el agente SNMP.

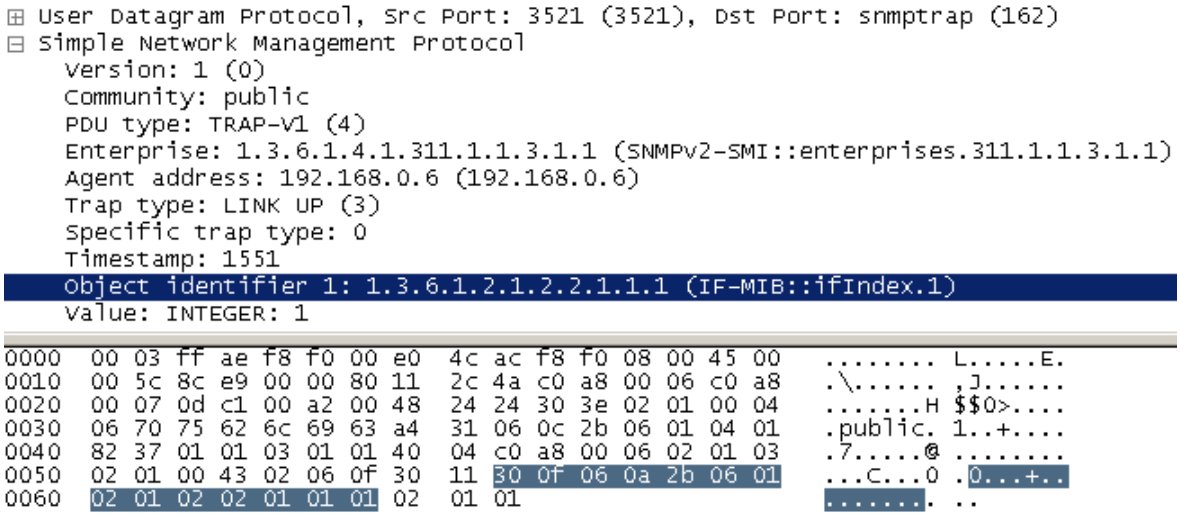


Figura D.26: Valor correspondiente al campo varbind del paquete SNMP TRAP.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		**Formato**	**Número de Marca**				
0	**0**	**1**	**1**	**0**	**0**	**0**	**0**
Universal		**Codificación Estructurada**	**Sequence = 16**				

Tabla D.58: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	1	1	1	1
15 Bytes							

Tabla D.59: Valor Binario

- **Valor:** Campo Object Identifier.

El valor de este campo corresponde al objeto identificador 1.3.6.1.2.1.2.2.1.1.1.

D.4.12 Codificación del Campo Object Identifier

La captura del campo objeto identificador del paquete SNMP, cuyo valor es 06_H $0a_H$ $2b_H$ 06_H 01_H 02_H 01_H 02_H 02_H 01_H 01_H 01_H.

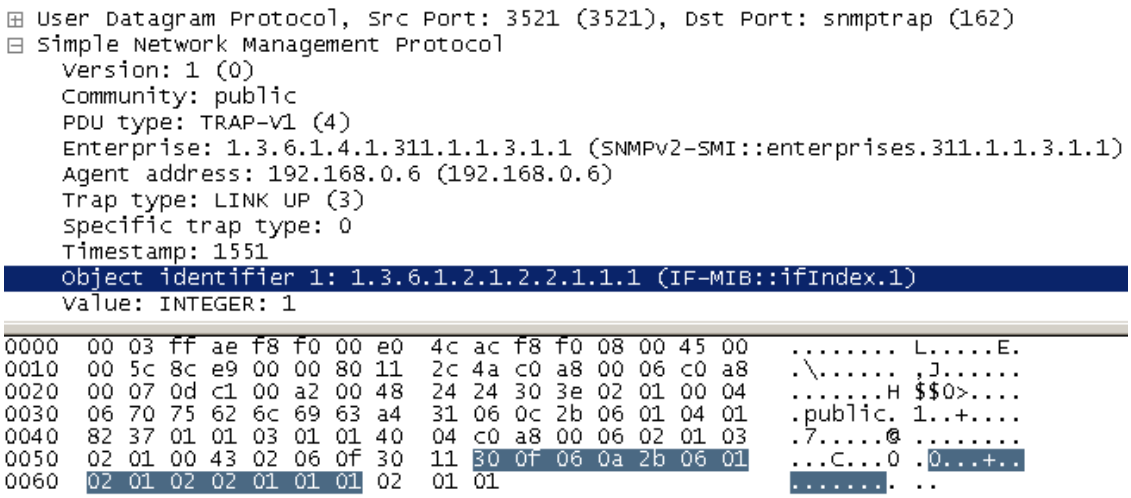


Figura D.27: Valor correspondiente al campo objeto identificador del paquete SNMP TRAP.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	0	0	0	0	1	1	0
Universal		Codificación Simple	Object Identifier = 6				

Tabla D.60: Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	1	0	0	0
8 Bytes							

Tabla D.61 Valor Binario

- **Valor:** Octetos del campo valor:

0	0	1	0	1	0	1	1
40*1+3 = 43 1 = iso 3 = org							

0	0	0	0	0	1	1	0
6 = dod							

0	0	0	0	0	0	0	1
1 = internet							

0	0	0	0	0	0	1	0
2 = mgmt							

0	0	0	0	0	0	0	1
1 = mib-2							

0	0	0	0	0	0	1	0
2							

0	0	0	0	0	0	1	0
2							

0	0	0	0	0	0	0	1
1							

0	0	0	0	0	0	0	1
1							

0	0	0	0	0	0	0	1
1							

Tabla D.62 Valor Binario

D.4.13 Codificación del Campo Value

La captura del campo valor del paquete SNMP es igual a 02_H 01_H 01_H.

```
⊞ User Datagram Protocol, Src Port: 3521 (3521), Dst Port: snmptrap (162)
⊟ Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: TRAP-V1 (4)
    Enterprise: 1.3.6.1.4.1.311.1.1.3.1.1 (SNMPv2-SMI::enterprises.311.1.1.3.1.1)
    Agent address: 192.168.0.6 (192.168.0.6)
    Trap type: LINK UP (3)
    Specific trap type: 0
    Timestamp: 1551
    Object identifier 1: 1.3.6.1.2.1.2.2.1.1.1 (IF-MIB::ifIndex.1)
    Value: INTEGER: 1

0000  00 03 ff ae f8 f0 00 e0  4c ac f8 f0 08 00 45 00   ........ L.....E.
0010  00 5c 8c e9 00 00 80 11  2c 4a c0 a8 00 06 c0 a8   .\...... ,J......
0020  00 07 0d c1 00 a2 00 48  24 24 30 3e 02 01 00 04   .......H $$0>....
0030  06 70 75 62 6c 69 63 a4  31 06 0c 2b 06 01 04 01   .public. 1..+....
0040  82 37 01 01 03 01 01 40  04 c0 a8 00 06 02 01 03   .7.....@ ........
0050  02 01 00 43 02 06 0f 30  11 30 0f 06 0a 2b 06 01   ...C...0 .0...+..
0060  02 01 02 02 01 01 01 02  01 01                     ........ ..
```

Figura D.28: Valor del paquete SNMP TRAP.

A continuación se codifica este campo de acuerdo a las reglas de codificación básica BER.

- **Tipo:** El campo tipo está compuesto por 8 bits de los cuales:

Clase de Marca		Formato	Número de Marca				
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**
Universal		**Codificación Simple**	**Integer= 2**				

Tabla D.63 Valor Binario

- **Longitud:** Número de octetos del campo valor:

0	0	0	0	0	0	0	1
1 Bytes							

Tabla D.64 Valor Binario

- **Valor:** Campo valor:

0	0	0	0	0	0	0	1
1 Bytes							

Tabla D.65 Valor Binario

El valor de este campo nos indica que la interfaz que se reinició es la Ethernet 1 del equipo monitoreado.