Instituto Tecnológico de Costa Rica

Maestría en Computación

Tesis para optar al grado de

Magister Scientae en Computación

SDK para desarrollo de aplicaciones en el manejo de sistemas de seguridad en casas inteligentes basado en SNMP

Prof. Asesor: Ulises Agüero, PhD.

Christian Sanabria Jiménez

Junio, 2001

Aprobación de Tesis

"SDK para desarrollo de aplicaciones en el manejo de sistemas de seguridad en casas inteligentes basado en SNMP"

Resumen

Una casa inteligente involucra la aplicación de tecnologías modernas dentro del ambiente del hogar, de modo que se automaticen muchas tareas, se faciliten otras y en general, se viva con mayor comodidad y seguridad.

Actualmente, existe mucha tecnología alrededor de las casas inteligentes y muchas empresas desarrollan dispositivos y software para administrarlos, pero no existen estándares generalizados que permitan la interoperabilidad entre tecnologías y el uso de dispositivos de distintos fabricantes.

Para resolver este problema se propone SNMP como protocolo estándar para administrar dispositivos en una casa inteligente, ya que es ampliamente utilizado dentro del ambiente de administración de redes de computadoras y que por sus características permite administrar cualquier dispositivo desde una interfaz común. Y dado que el conjunto de dispositivos de una casa inteligente corresponde a una red de datos, SNMP puede ser utilizado para administrarlos.

En este trabajo el diseño se ha centrado específicamente en el control y administración de dispositivos para los sistemas de seguridad de una casa inteligente implementado un conjunto de herramientas de software y documentación que permitan posteriormente desarrollar nuevas aplicaciones de seguridad basadas en SNMP.

Estas herramientas se implementaron en forma de componentes COM reutilizables desde cualquier lenguaje de programación que soporte este estándar. Los componentes, junto con las herramientas generadas y la documentación conformarán un Software Development Kit (SDK) que proveerá servicios y servirá de base para el posterior desarrollo de nuevas aplicaciones.

Como parte de este proyecto y para comprobar la funcionalidad del SDK, se desarrollaron agentes que controlan algunos dispositivos de seguridad comunes (sensores y sirenas, principalmente) y una aplicación administrativa que brinda una interfaz sencilla y permite controlar los dispositivos y conocer su estado.

El software desarrollado y el escenario de pruebas utilizando dispositivos reales en que se ha aplicado han permitido observar el funcionamiento del hardware, los agentes y las herramientas administrativas implementadas, ha permitido demostrar que SNMP es una solución factible y con muchas ventajas en la administración de dispositivos no solo de seguridad, sino de cualquier otro tipo, en una casa inteligente.

Dedicatoria

A mis papás y abuelitas por estar ahí siempre y ser mi guía y mi modelo a seguir.

A mis sobrinos María y Mauricio, a quienes veo crecer y aprender cosas nuevas cada día y representan para mí todo lo bueno que hay en el mundo y la esperanza para el futuro.

Agradecimientos

A Dios porque sin Él nada es posible.

Mi mayor agradecimiento al Ing. Mario Daniel Ramírez, por su apoyo incondicional como jefe y como amigo.

Al Dr. Ulises Agüero por su guía y sus consejos desde el inicio de este trabajo.

A todos mis compañeros del TEC, por su ayuda y el apoyo que cada cual a su manera me ha brindado, especialmente mis amigos del Centro de Cómputo.

Al TEC por darme la beca que me permitió estudiar la maestría.

Al Ing. José Jiménez por ser parte de esta idea y por su ayuda en la búsqueda de información, en el análisis de los requerimientos y en las pruebas del software.

Al personal de la Carrera de Criminología y la Biblioteca del Colegio Universitario de Cartago, por la información que nos brindaron.

Tabla de Contenidos

Índice de Figuras

1. Introducción

En este trabajo se han combinado dos tecnologías:

- la tecnología de casas y dispositivos inteligentes,
- la tecnología de administración de redes SNMP

Aunque en el mercado actual no existen dispositivos de seguridad ni de automatización del hogar basados en SNMP, la propuesta que se presenta trata de explotar esta tecnología y comprobar su funcionamiento en el ambiente de sistemas de seguridad para casas inteligentes.

El resultado que se presenta es un Kit de Desarrollo de Software que explota el protocolo SNMP y lo aplica a la administración de una serie de dispositivos de seguridad implementados por hardware y software, permitiendo la integración de casos y la creación de escenarios que demuestran la aplicabilidad del protocolo a los sistemas de seguridad.

Este trabajo presenta los objetivos, las bases teóricas, el diseño y la implementación del SDK para Casas Inteligentes basado en SNMP, al que se llamó SDKCI_SNMP. Y contiene además una descripción de cómo se puede aplicar el producto a un caso real con dispositivos de seguridad conectados a una computadora que los controla y les brinda su “inteligencia” a través de un agente SNMP.

1.1. Objetivos

1.1.1. Objetivo General

Desarrollar un modelo de software (SDK) que utilice como base el protocolo SNMP para controlar el sistema de seguridad en una casa inteligente.

1.1.2. Objetivos Específicos

- Analizar la factibilidad, ventajas y desventajas del uso del protocolo SNMP para la administración de dispositivos en casas inteligentes, utilizando la tecnología que existe actualmente en redes LAN y WAN.

- Investigar acerca de la tecnología de hardware y software utilizada actualmente para implementación de casas inteligentes.

- Analizar los requerimientos de administración de dispositivos en casas inteligentes, tomando el caso de los dispositivos que conforman el sistema de seguridad.

- Determinar los requerimientos de software para la implementación de SNMP como protocolo de administración de casas inteligentes.

- Diseñar un modelo de objetos que implemente la funcionalidad requerida por las aplicaciones de control de dispositivos de seguridad tradicionales, utilizando las herramientas de UML.

- Determinar requerimientos y elaborar MIBs para administración de algunos dispositivos comunes en casas inteligentes.

- Realizar un análisis de requerimientos y diseñar un software administrativo básico que permita administrar dispositivos de seguridad en casas inteligentes.

- Implementar emuladores para obtener un ambiente de simulación de una casa inteligente.

- Probar el software desarrollado y la funcionalidad de la propuesta por medio de la interconexión física de dispositivos y agentes en un escenario de pruebas real.

1.2. Marco Teórico

Los temas a los que se ha hecho referencia en este proyecto son:

- El Protocolo SNMP
- Hewlett Packard SNMP++ y Dart Power TCP
- Casas Inteligentes
- Sistemas de seguridad
- Desarrollo de SDKs

A continuación se incluye un resumen de la información recopilada acerca de cada uno de ellos.

1.2.1. SNMP

Un sistema de administración de red contiene dos elementos primarios: un administrador y un conjunto de agentes. El administrador es la consola por medio de la cual el administrador de red ejecuta sus funciones. Los agentes son las entidades que unen al dispositivo que se administra. Puentes, concentradores, routers y servidores de red son ejemplos de dispositivos administrados que contienen agentes y objetos administrables.

Estos objetos administrables pueden ser hardware, parámetros de configuración, estadísticas de rendimiento y otros que directamente se relacionan con el funcionamiento del dispositivo en cuestión. Estos objetos se colocan en lo que es conocido como un banco de datos de información virtual llamado una base de información de administración, o MIB. El SNMP permite a administradores y agentes comunicarse con el propósito de acceder a estos objetos. [RAD95]

1.2.1.1. El estándar SNMP [DDRI, 99]

Se puede considerar que el protocolo SNMP está conformado por:

1. **Un formato estándar de mensaje.** El SNMP es un protocolo de comunicación estándar que define un mensaje de formato UDP. Esta parte de la norma está muy envuelta, y es de pequeña consecuencia a los usuarios (pero de gran interés a programadores de SNMP)

2. **Un conjunto estándar de objetos administrados.** SNMP es un juego estándar de valores (llamado “objetos” SNMP) que pueden preguntarse desde un dispositivo. Específicamente, el estándar incluye valores por supervisar TCP, IP, UDP, e interfaces del dispositivo. Cada objeto administrable es identificado con un nombre oficial y también con un identificador numérico expresado en notación punto.

3. **Un estándar para agregar objetos.** Una razón por la que el SNMP se ha vuelto popular hoy y es el estándar de la industria es que permite modificar e incrementar la cantidad de posibles dispositivos y variables de configuración de cada uno de ellos.

1.2.1.2. Agentes SNMP [MATTHEWS, 96]

Cada equipo administrable vía SNMP tiene un pequeño agente construido dentro de él. Cada uno de estos agentes está conectado a la red y su objetivo es poder responder a solicitudes futuras desde un programa que se ejecuta en un administrador SNMP en otra parte en la red.

Los agentes son pequeños programas incluidos dentro de los dispositivos, de bajo costo y sencillos de implementar. Las aplicaciones administrativas pueden ser tan complejas y poderosas como se quiera, pueden tener enormes requerimientos de hardware y su costo es usualmente alto.

Hay solamente cinco comandos en SNMP, tres son enviados por el administrador al agente (GET, GETNEXT, y SET) y dos son enviados por el agente al administrador (GET RESPONSE y TRAP)

Los programas de administración envían periódicamente comandos Get (o Get Next) a los agentes que ellos conocen sobre ciertos valores de los objetos, y los agentes envían mensajes Get Response con el valor actual de cada uno de estos objetos. Los programas administradores trabajan en la recolección, organización e interpretación de estas respuestas.

Esto se hace a través de "MIBs" la base de información de administración, que es el diccionario formal o conjunto de nombres de objetos manejables y define cual es el significado de cada objeto y los valores que puede tomar cada variable.

1.2.1.3. Ventajas de SNMP [INFORAMP, 98]

La principal ventaja del protocolo SNMP es que su diseño es sencillo y fácil de implementar en cualquier red, lo cual lo hace fácil para que un usuario programe variables que les gustaría supervisar, desde una perspectiva de más alto nivel. Cada variable consiste de la siguiente información:

- El título de la variable
- El tipo de dato de la variable (ej. Integer, String)
- Si la variable es solamente de lectura o lectura-escritura.
- El valor de la variable.

El resultado total de esta simplicidad es un software de administración de red fácil de implementar y que no sobrecarga la red existente.

Otra ventaja es que su uso es muy amplio hoy. Casi todos los vendedores de hardware de interconexión de redes como hubs, switches y routers, servidores, impresoras diseñan sus productos con soporte para SNMP, haciendo muy fácil su administración conjunta a través de aplicaciones basadas en este protocolo.

La expansión es otro beneficio del SNMP. Debido a su diseño simple, es fácil para el protocolo ser actualizado para cubrir las necesidades futuras de los usuarios y el crecimiento de la red.

1.2.1.4. Usos de SNMP en la administración de redes [TECHNET, 2000]

La administración de red es crítica para el aprovechamiento de los recursos y la auditoría. El SNMP puede usarse de varias maneras:

- **Configuración remota de dispositivos**: Es posible administrar la configuración de cada dispositivo desde la terminal de administración vía SNMP

- **Monitoreo del rendimiento de la red**: Se puede rastrear la velocidad y rendimiento de la red y se puede coleccionar información sobre el éxito de transmisiones de los datos.

- **Detectar fallas en la red o accesos inapropiados**: Puede configurar la activación de alarmas en dispositivos de red que alertan sobre la ocurrencia de eventos específicos. Cuando una alarma se activa, el dispositivo remite un mensaje de evento al sistema de administración. Los tipos comunes de eventos para los que una alarma puede configurarse incluyen:

    - El cierre o reinicio de un dispositivo.
    - La detección de fallas en enlaces o router.
    - El acceso no autorizado

- **Auditoría de la red**: Puede supervisar el uso de la red global para identificar usuarios, grupos de accesos, tipos de uso para dispositivos de la red o servicios. Esta información puede usarse directamente para generar facturación individual, las cuentas de grupo o justificar costos actuales de la red o gastos planeados.

1.2.2. Desarrollo de Aplicaciones con SNMP

Con la finalidad de poner en práctica y realizar pruebas de aplicaciones reales basadas en SNMP, se ha localizado información relacionada con APIs, SDKs, componentes y controles ActiveX que permiten desarrollar aplicaciones utilizando herramientas de programación comunes como Visual C++ y Visual Basic.

Al utilizar estas herramientas es posible desarrollar los dos tipos de aplicaciones involucradas dentro del protocolo SNMP: Agentes y Aplicaciones Administrativas. A continuación se da una breve descripción de las principales herramientas evaluadas y su funcionalidad [SANABRIA, 2000].

1.2.2.1. SNMP++ [HP97]

Este API consiste en una serie de clases que facilitan, desde C++, desarrollar aplicaciones que utilicen SNMP v1. Estas clases permiten implementar herramientas administrativas que utilicen todas las operaciones que provee el protocolo, como la posibilidad de poder escuchar en el puerto 162 y poder enviar traps, este conjunto de clases puede también ser utilizado para implementar agentes SNMP que sean portables entre plataformas y que puedan colocarse en distintos dispositivos.

Por las facilidades que ofrece esta API y por la completa documentación y ejemplos que la acompañan, se decidió utilizarla como la principal herramienta para el desarrollo del SDK propuesto.

1.2.2.2. Dart PowerTCP SNMP Tool [Dart 98]

Consiste en un conjunto de componentes COM y controles ActiveX que permiten desarrollar aplicaciones utilizando cualquier herramienta de desarrollo compatible con el estándar COM (Visual Basic, Visual C++)

Este conjunto de controles provee toda la funcionalidad de SNMP con la facilidad de Visual Basic, ocultando todos los detalles dentro de los componentes COM.

En Visual Basic se puede utilizar el SNMP Agent Control para desarrollar agentes SNMP que atiendan solicitudes y envíen traps, así como el SNMP Manager Control para desarrollar aplicaciones administrativas para SNMP. Además se incluye el SNMP Service Control para convertir una aplicación desarrollada en un servicio para Windows NT.

1.2.3. Casas Inteligentes (Home Automation)

Al implementar un sistema automatizado es posible eliminar la intervención humana y proporcionar un nivel más consistente de calidad. Casi cualquier tarea que requiere la intervención humana puede automatizarse. Para automatizar una tarea debemos primero definirla y entonces refinar los pasos específicos para completarla [VISS, 2000].

La automatización se refiere a que los eventos de un sistema o las actividades de un método pueden ser controlados sin el pensamiento consciente [ASTRO, 97].

1.2.3.1. Beneficios de la automatización

Entre los beneficios que otorga la automatización [NETCOM, 2000] están:

- Conveniencia personal
- Adiciona seguridad
- Medición de la seguridad
- Eficiencia en la energía

De una manera general, una casa inteligente dispondrá de una red de comunicación que permite la interconexión de una serie de equipos con el fin de obtener información sobre el entorno doméstico y basándose en ésta, realizar determinadas acciones sobre dicho entorno.

En este sentido, una casa inteligente se puede definir como: "aquella vivienda en la que existen agrupaciones automatizadas de equipos (dispositivos), normalmente asociados por funciones, y que disponen de la capacidad de comunicarse interactivamente entre sí a través de una red".

Entre las aplicaciones que puede ofrecer una casa inteligente podemos señalar:

- Seguridad
- Administración de la energía
- Automatización de tareas domésticas
- Formación cultura y entretenimiento
- Teletrabajo
- Monitoreo de la salud
- Operación y mantenimiento de las instalaciones, etc.

A continuación se presentan diferentes definiciones que ha ido tomando el término [DOMO2000]:

- La nueva tecnología de los automatismos de maniobra, gestión y control de los diversos aparatos de una vivienda que permiten aumentar el confort del usuario, su seguridad, y el ahorro en el consumo eléctrico.

- Un conjunto de servicios en las viviendas, asegurados por sistemas que realizan varias funciones, pudiendo estar conectados entre ellos y a redes internas y externas de comunicación.

- La informática aplicada a la vivienda. Agrupa el conjunto de sistemas de seguridad y de la regulación de las tareas domésticas destinadas a facilitar la vida cotidiana automatizando sus operaciones y funciones

1.2.4. Sistemas de Seguridad

Cualquier sistema, que proporcione a su usuario un grado razonable de protección contra peligros reales o imaginarios, amenazas o molestias (tales como agresión, robo, intrusión indeseada de seres humanos o de animales, averías de máquinas o riesgo de incendios, descarga eléctricas etc.), puede ser descrito como un sistema de seguridad [MARSTON, 99].

La unidad de seguridad electrónica más conocida y más ampliamente utilizada es la llamada "alarma antirrobo", diseñada generalmente para detectar a los intrusos que intentan penetrar en locales protegidos, permitiendo al mismo tiempo que personas autorizadas puedan circular libremente por los mencionados locales y entrar o abandonar sin problemas su perímetro [MARS99].

Tomar medidas de seguridad en los hogares está estrechamente relacionado con una serie de factores externos como la ubicación geográfica, la presencia de lotes, la altura de las tapias y las comunidades adyacentes, al igual que la tecnología disponible[SEGU98].

Los sistemas de alarmas por computación, se instalan en lugares estratégicos previamente deducidos en el análisis de vulnerabilidad respectivo. Estas alarmas de asalto o robo generalmente están conectadas en puertas, ventanas y otros lugares; al introducirse cualquier persona en una planta por alguno de estos lugares sin tener el acceso correspondiente, activará la alarma y el reporte se recibirá casi de inmediato [FALLAS, 92].

1.2.5. Diseño de un sistema de seguridad [MORA, 96]

Para diseñar un sistema de alarma seguro no es necesario saturar a un lugar de dispositivos sino más bien emplear el número mínimo necesario de éstos ubicándolos en forma estratégica, en aquellos lugares por donde ingresaría un presunto intruso.

1.2.5.1. Factores básicos

Un sistema de alarma está compuesto por:

- El control central
- La sirena o señal audible
- Los detectores de intrusos
- Cámaras y otros sistemas de vigilancia
- La señal de aviso remoto

1.2.5.2. El control central

Este indica el estado del sistema, alarma ocurrida y otros detalles. Permite configurar cada zona y controlar detalles que ayudan a obtener un sistema estable.

1.2.5.3. La sirena o señal audible

Dado que, generalmente un sistema de alarma pretende intimidar al intruso y alertar a los ocupantes del área violada, la señal audible viene a desempeñar un papel muy importante. Se debe ubicar la sirena en un lugar oculto y a resguardo de ser desconectada, puesto que en muchas ocasiones es el único medio de alerta usado, especialmente si sólo se utiliza la alerta local sin aviso remoto alguno.

1.2.5.4. Detectores de intruso

Se conoce como detectores a los dispositivos que activan el sistema de alarma al identificar una situación irregular como apertura de una puerta, movimiento interno o la quebradura de un vidrio.

1.2.5.5. Señal de aviso remoto

La señal de auxilio en caso de alarma viene a ser un importante complemento del sistema de seguridad. Para esto es importante que el sistema comunique la emergencia a las autoridades o interesados, ésta generalmente se realiza a través de la línea telefónica.

1.2.6. Desarrollo de SDKs

Un SDK es una herramienta orientada a desarrolladores de aplicaciones que desean utilizar una tecnología como base para crear nuevos programas.

Cuando se desarrolla una tecnología de cualquier tipo y se desea que otros la utilicen y creen sus propias soluciones basadas en ella, se expone ésta a través de un SDK, ocultando todos los detalles de bajo nivel y exponiendo la funcionalidad a través de una interfaz de mayor nivel.

Hay varios tipos de SDKs:

- Puede ser un conjunto de archivos con código fuente que incluya declaraciones, funciones y procedimientos, a partir de los cuales un desarrollador puede crear una nueva aplicación.
- Puede consistir en un API almacenado en bibliotecas dinámicas (DLLs) y utilizable principalmente desde C++.
- Puede ser un conjunto de componentes utilizando distintas tecnologías que exponen interfaces más sencillas de usar desde una mayor variedad de lenguajes.

Al exponer una tecnología a través de componentes COM podrá ser utilizado tanto por programadores en C++ como por quienes desarrollan en lenguajes de más alto nivel como Visual Basic, Delphi, Powerbuilder o cualquier otro lenguaje que soporte el estándar COM.

El SDK oculta los detalles de bajo nivel de la tecnología que implementa, pero provee una interfaz que permite explotar la tecnología y desarrollar nuevas aplicaciones. Por esto, el SDK debe desarrollarse pensando en la facilidad de uso, pero no por esto limitando la funcionalidad.

Un SDK se puede utilizar también para proteger algoritmos o estructuras que sean propiedad privada pero cuya funcionalidad puede ser útil para otros desarrolladores.

2. Desarrollo del Modelo

En los puntos siguientes se muestra el análisis de los requerimientos de un sistema de seguridad típico y el diseño e implementación realizados en este trabajo.

2.1. Análisis del sistema de seguridad

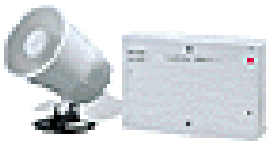
Se ha consultado distintas fuentes que incluyen revistas especializadas y catálogos de compañías que fabrican sistemas de alarma para recopilar todos los requerimientos y ajustarse a las necesidades reales de cualquier sistema de seguridad.

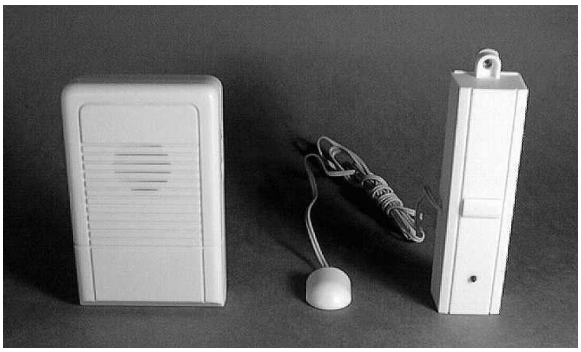
2.1.1. Descripción del Sistema de Seguridad Típico

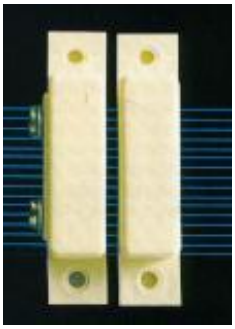
Un sistema típico consta de distintos tipos de dispositivos que monitorean y controlan eventos relacionados con la seguridad, y controladores que se encargan de recibir información de los dispositivos y generar distintos tipos de alarmas.

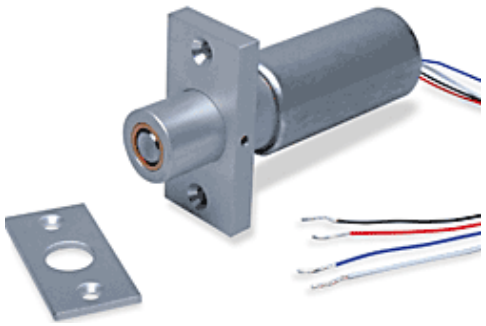
Entre los dispositivos que se pueden encontrar en un sistema de seguridad están:

Dispositivos de Control y Administración	
Unidades de control	Dispositivos que administran el resto del sistema de seguridad
Controles remotos	Permiten al usuario realizar distintas funciones desde un control remoto pero solamente dentro o en un perímetro cercano al lugar

Dispositivos de reporte de alarmas	
Sirenas	Se encienden en caso de alarma  **Figura 1.** Sirena para aviso de alarmas
Luces de alarma	Se encienden en caso de alarma
Modems y líneas telefónicas	Permiten al sistema llamar a distintos números en caso de una situación de alerta

Sensores	
Sensores de movimiento	Detectan el movimiento en la zona en que se coloquen.  **Figura 2.** Sensor de movimiento
Sensores de humo	Detectan la presencia de humo en el lugar donde se coloquen
Sensores de agua	Detectan el ingreso de agua en el lugar en que se coloquen.  **Figura 3.** Sensor de agua

Sensores magnéticos para puertas y ventanas	Detectan la apertura de puertas y ventanas.  **Figura 4.** Sensor magnético para puertas y ventanas
Sensores de vidrios rotos	Detectan las frecuencias sonoras provocadas por un vidrio al romperse
Sensores infrarrojos	Detectan un intruso si se interrumpe un rayo infrarrojo entre un transmisor y un receptor (o reflector)

Otros Dispositivos	
Cerrojos electrónicos para puertas	Permiten abrir y/o cerrar puertas.  **Figura 5.** Cerradura electrónica
Timers	Encienden o apagan distintos elementos en ciertos momentos del día
Baterías de respaldo	En caso de cortes de corriente, mantienen el sistema de seguridad funcionando

Dispositivos de Circuito Cerrado de TV	
Cámaras	Permiten observar distintas zonas  **Figura 6.** Sistema de video para seguridad
Equipos de monitoreo y control	Permiten observar el video de las cámaras de seguridad y en algunos casos controlarlas: moverlas, acercar o alejar, enfocar, etc.
Videograbadoras	Graban constantemente el video obtenido de las cámaras de seguridad instaladas

2.1.2. Usuarios del sistema de seguridad

Los usuarios del sistema de seguridad son las personas que de una u otra forma interactúan con él. Entre ellos:

- El / los usuarios que administran el sistema
- Los usuarios que solamente encienden o apagan la alarma (realizan un subconjunto de las funciones de los anteriores)
- Los usuarios que reciben los reportes de alarmas (pueden ser oficiales de seguridad, oficinas de vigilancia remota, o los mismos usuarios anteriores)
- Los intrusos que pueden provocar alertas en el sistema de seguridad

2.1.3. Funciones del sistema de seguridad

Entre las funciones que el sistema de seguridad debe proveer a sus usuarios se encuentran:

- **Programar el sistema**: cambiar parámetros del sistema o de los dispositivos
- **Modificar el estado del sistema**: habilitar o deshabilitar dispositivos, o armar/desarmar la alarma
- **Controlar dispositivos**: si es posible controlarlos, tal como mover una cámara de vigilancia
- **Monitorear el sistema**: conocer en cualquier momento y por distintos medios el estado del sistema de seguridad y de cada uno de sus dispositivos
- **Generar reportes**: el usuario puede ver una lista histórica de eventos ocurridos en el sistema
- **Generar alarmas**: el sistema debe proveer distintos medios de informar al usuario de la ocurrencia de eventos de seguridad
- **Descubrir dispositivos**: debe encontrar nuevos dispositivos instalados, y debe poder administrarlos transparentemente

2.1.4. Análisis utilizando Casos de Uso

A partir del sistema de seguridad descrito, se ha elaborado un diagrama de casos de uso, de acuerdo con la metodología UML. En el diagrama de la Figura 7 se muestran los actores involucrados y los casos de uso requeridos por cada actor para interactuar con el sistema.

El diagrama de casos de uso se muestra a continuación:

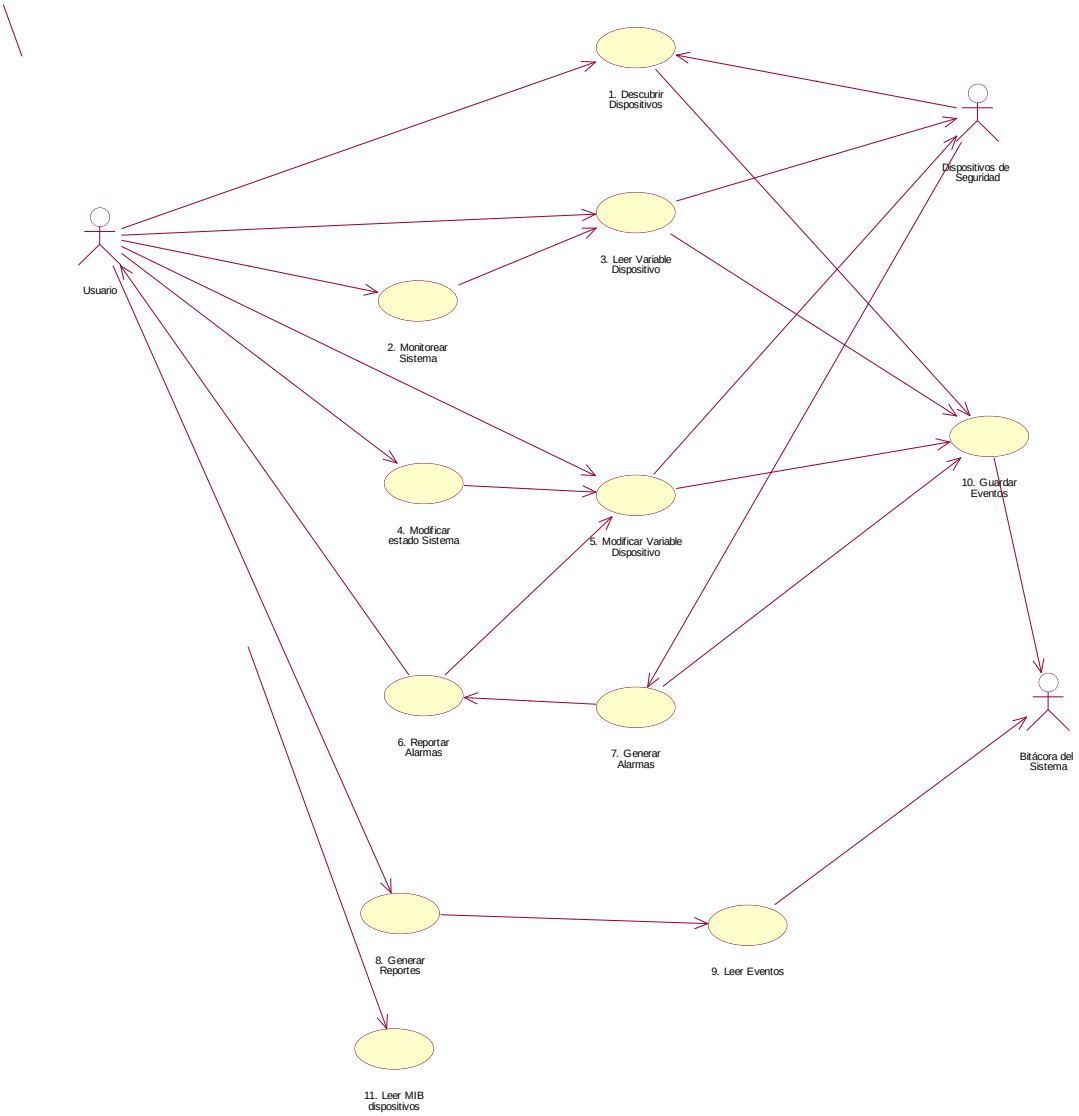


Figura 7. Diagrama de Casos de uso para un sistema de seguridad típico.

2.1.4.1. Actores

Los actores identificados son:

- **Usuario**: utiliza o administra el sistema, o recibe alarmas de seguridad
- **Dispositivos**: cada dispositivo (hardware) es un actor que puede recibir o generar acciones sobre el sistema de seguridad.
- **Bitácora del sistema**: este actor tiene la funcionalidad de almacenar y brindar información sobre los eventos ocurridos al sistema de seguridad

2.1.4.2. Especificación de los Casos de Uso

Caso de uso # 1	
Nombre	Descubrir Dispositivos
Descripción	Cuando el usuario utiliza una interfaz de administración ésta realiza una verificación en la red para conocer todos los dispositivos administrables que existen.
Precondiciones	Equipo administrativo conectado a la red
Condición final exitosa	Se tiene una colección con 1 o más dispositivos administrables
Condición final no exitosa	Se tiene una colección de dispositivos vacía
Actores	El usuario del sistema Los dispositivos administrables en la red
Iniciado por	El usuario del sistema

Caso de uso # 2	
Nombre	Monitorear sistema
Descripción	El usuario a través de una aplicación administrativa puede conocer el estado y las características de cada uno de los dispositivos en la red.
Precondiciones	Se ha realizado el descubrimiento de dispositivos
Condición final exitosa	Se puede ver el estado de las variables de todos los dispositivos
Condición final no exitosa	No se puede ver el estado de las variables de algún dispositivo.
Actores	El usuario del sistema
Iniciado por	El usuario del sistema

Caso de uso # 3	
Nombre	Leer variable dispositivo
Descripción	La aplicación administrativa que utiliza el usuario solicita una variable del dispositivo
Precondiciones	Se ha realizado el descubrimiento de dispositivos
Condición final exitosa	Se obtiene el valor de la variable
Condición final no exitosa	No se obtiene el valor de la variable, se obtiene un mensaje de error porque la variable no existe o se cumple el tiempo de espera.
Actores	Usuario del sistema Dispositivos administrados
Iniciado por	El usuario del sistema Caso de uso 2

Caso de uso # 4	
Nombre	Modificar estado sistema
Descripción	El usuario a través de la interfaz administrativa cambia el estado del sistema o de algún dispositivo específico
Precondiciones	Se ha realizado el descubrimiento de dispositivos
Condición final exitosa	Se modifica el estado de los dispositivos administrados
Condición final no exitosa	No se modifica el estado de los dispositivos
Actores	El usuario del sistema
Iniciado por	El usuario del sistema

Caso de uso # 5	
Nombre	Modificar variable dispositivo
Descripción	La aplicación administrativa solicita modificar una variable de un dispositivo
Precondiciones	Se ha realizado el descubrimiento de dispositivos
Condición final exitosa	Se modifica la variable en el dispositivo
Condición final no exitosa	No se modifica la variable en el dispositivo. Se obtiene un mensaje de error, la variable es de solo lectura o se cumple el tiempo de espera.
Actores	El usuario del sistema Los dispositivos administrados
Iniciado por	El usuario del sistema Caso de uso 5 Caso de uso 6

Caso de uso # 6	
Nombre	Reportar Alarmas
Descripción	La interfaz administrativa que el sistema utiliza puede indicar mediante diferentes medios la aparición de alarmas generadas por los dispositivos administrados
Precondiciones	Se ha realizado el descubrimiento de dispositivos
Condición final exitosa	Se indica la alarma mediante distintos medios
Condición final no exitosa	No se indica la alarma
Actores	El usuario del sistema Dispositivos administrados
Iniciado por	Caso de uso 7: generar alarmas

Caso de uso # 7	
Nombre	Generar alarmas
Descripción	Un dispositivo activa su señal de alarma
Precondiciones	Dispositivo armado
Condición final exitosa	El sistema reconoce e indica la alarma al usuario de las distintas maneras programadas
Condición final no exitosa	No se indica que se produjo la alarma y se produce un mensaje de error
Actores	Dispositivos Administrados
Iniciado por	Dispositivos Administrados

Caso de uso # 8	
Nombre	Generar Reportes
Descripción	El usuario desde una interfaz adminsitrativa solicita reportes de eventos del sistema o información sobre un evento específico
Precondiciones	Ninguna
Condición final exitosa	Se tiene un listado de eventos
Condición final no exitosa	No se obtiene el listado de eventos
Actores	El usuario del sistema
Iniciado por	El usuario del sistema

Caso de uso # 9	
Nombre	Leer eventos
Descripción	Se lee el archivo de eventos generado por el subsistema de bitácora del sistema
Precondiciones	Ninguna
Condición final exitosa	Se tiene el archivo de eventos
Condición final no exitosa	No se tiene el archivo de eventos
Actores	Bitácora del sistema
Iniciado por	Caso de uso 8: Generar Reportes

Caso de uso # 10	
Nombre	Guardar eventos
Descripción	Se almacena un evento en el archivo de eventos del subsistema de bitácora del sistema
Precondiciones	Se produce un evento en el sistema y se informa al subsistema de bitácora
Condición final exitosa	Se almacena el evento en la bitácora
Condición final no exitosa	No se almacena el evento en la bitácora, se produce un mensaje de error.
Actores	Bitácora del sistema
Iniciado por	Caso de uso 1: descubrir dispositivos Caso de uso 3: leer variable dispositivo Caso de uso 5: modificar variable dispositivo Caso de uso 7: Generar alarmas

Caso de uso # 11	
Nombre	Leer MIB
Descripción	Lee un archivo de información de dispositivo (SNMP MIB). Para conocer las variables y los traps del dispositivo
Precondiciones	Ninguna
Condición final exitosa	Se almacena internamente la información de este MIB
Condición final no exitosa	No se almacena la información del MIB, se produce un mensaje de error
Actores	Bitácora del sistema
Iniciado por	El usuario del sistema

2.2. Diseño del SDK

Con base en el análisis de los casos de uso y tomando en cuenta el diseño del esquema administrativo de una red SNMP, en donde deben haber agentes y aplicaciones administrativas. Se ha separado la funcionalidad general del sistema en tres elementos: los agentes, el SDK internamente y las aplicaciones administrativas.

2.2.1. Funciones de los agentes

Los agentes deben ser aplicaciones muy livianas y sencillas, pues deben ser parte de cada dispositivo. Además, para mantener consistencia con el SNMP, deben atender y realizar únicamente las operaciones especificadas por el protocolo.

Al comparar el agente con los casos de uso, se obtiene que es posible relacionar las operaciones SNMP con los requerimientos de algunos casos de uso, como se muestra en la siguiente tabla y se resalta en la Figura 8.

Caso de uso	Operación SNMP
3. Leer Variables Dispositivo	SNMP Get SNMP GetNext
5. Modificar variables Dispositivo	SNMP Set
7. Generar alarmas	SNMP Trap

De estas operaciones el agente debe atender Get, GetNext y Set. Y debe poder generar Traps en caso de detección de alarmas u otros eventos, dependiendo de la funcionalidad del dispositivo.

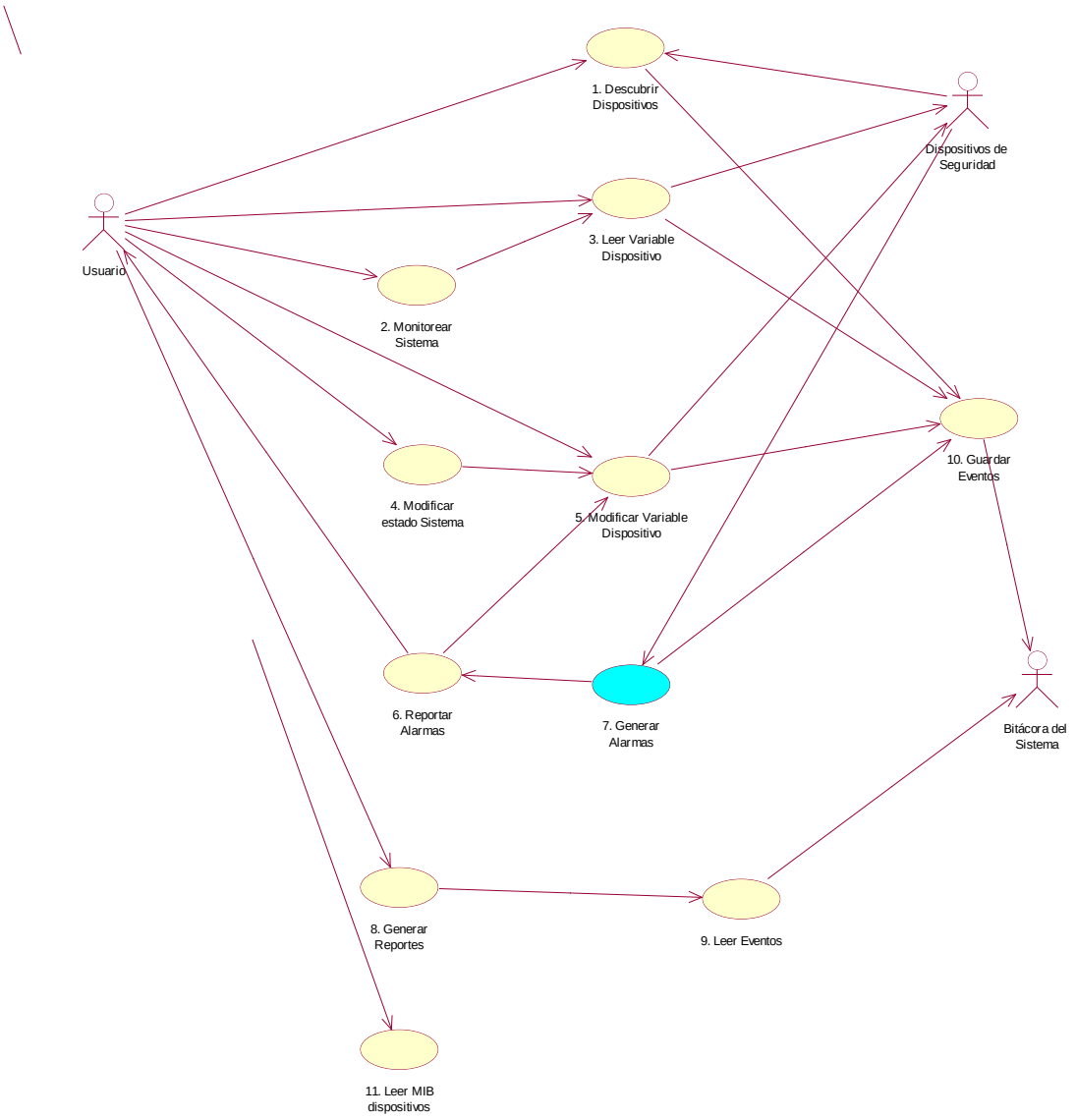


Figura 8. Se resalta en celeste el caso de uso que deben implementar los agentes SNMP

2.2.2. Funciones internas del SDK

El software desarrollado debe proveer la mayor parte de su funcionalidad de manera transparente para las aplicaciones administrativas. Esta funcionalidad incluye el manejo de bitácoras, las listas de dispositivos, el descubrimiento y cualquier otra labor de interacción con los dispositivos.

El SDK debe encargarse de traducir las solicitudes del usuario en mensajes SNMP que se envíen a los agentes, y deben encargarse también del proceso inverso, incluyendo el procesamiento automático de traps, el reporte de estas a las aplicaciones administrativas, y su almacenamiento en la bitácora de eventos.

En relación con los casos de uso, el SDK internamente resuelve la mayoría de ellos:

Caso de uso	Operaciones internas del SDK
1. Descubrir dispositivos	Generar los mensajes SNMP necesarios para crear una colección con los dispositivos existentes en la red
2. Monitorear sistema	Desde el momento en que se inicializa el SDK debe mantener disponible e informar a la aplicación usuario cualquier cambio en el estado de los dispositivos del sistema
3. Leer variable dispositivo	Generar el mensaje SNMP Get correspondiente y procesar la respuesta del agente
4. Modificar estado del sistema	Generar los mensajes SNMP Set necesarios para cambiar el estado a todos los dispositivos del sistema
5. Modificar variable dispositivo	Generar el mensaje SNMP Set correspondiente

6. Reportar alarmas	Generar los eventos necesarios para que el usuario o su interfaz administrativa reciban el reporte de alarma. Activar los dispositivos de alarma del sistema.
7. Generar alarmas	Procesar los mensajes SNMP Trap y generar los eventos necesarios para informar acerca de la alarma. También debe registrar el evento en la bitácora
9. Leer eventos	Leer los eventos del subsistema de bitácora y colocarlos en forma disponible para que una aplicación administrativa pueda generar reportes
10. Guardar eventos	Almacenar en forma permanente los eventos que ocurren en el sistema
11. Leer MIBS	Leer y almacenar la especificación de los dispositivos administrables. Para brindar a la aplicación usuario acceso a sus variables e información sobre las posibles alarmas que podría generar.

En la Figura 9 se muestran todos los casos de uso que resuelve internamente el SDK.

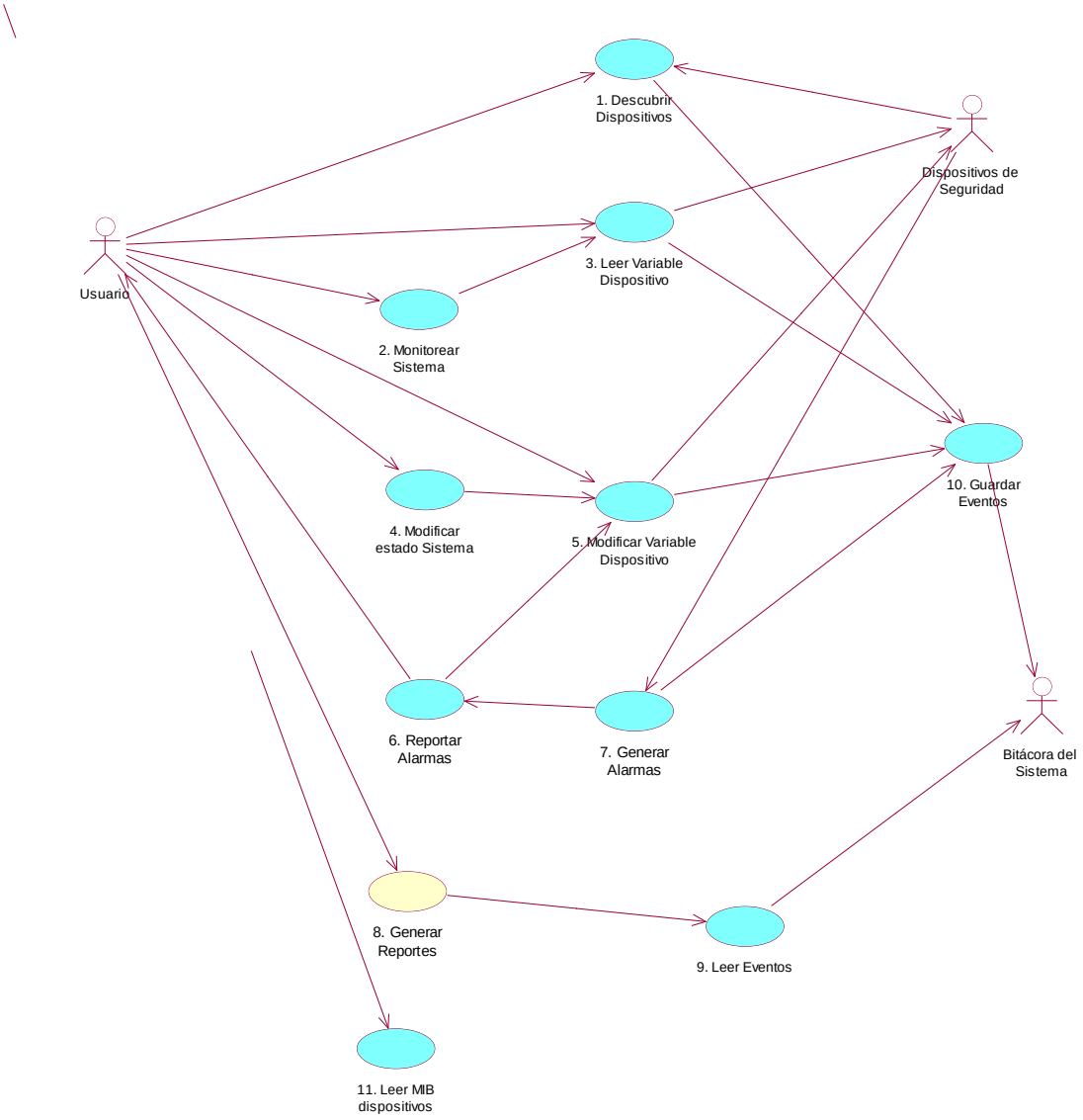


Figura 9. Se muestra en celeste los casos de uso que debe implementar internamente el SDK

2.2.3. Funciones de las aplicaciones administrativas

Como se ha visto hasta este punto, la mayoría de los casos de uso son resueltos internamente por el SDK. Esto deja a las aplicaciones administrativas la labor de ser la interfaz por medio de la cual un usuario interactúa con el sistema.

El único caso de uso que no se ha incluido como parte del SDK, debido a que también se relaciona más con la interfaz de usuario es el de la Generación de reportes (caso de uso 8), como se muestra en la Figura 10.

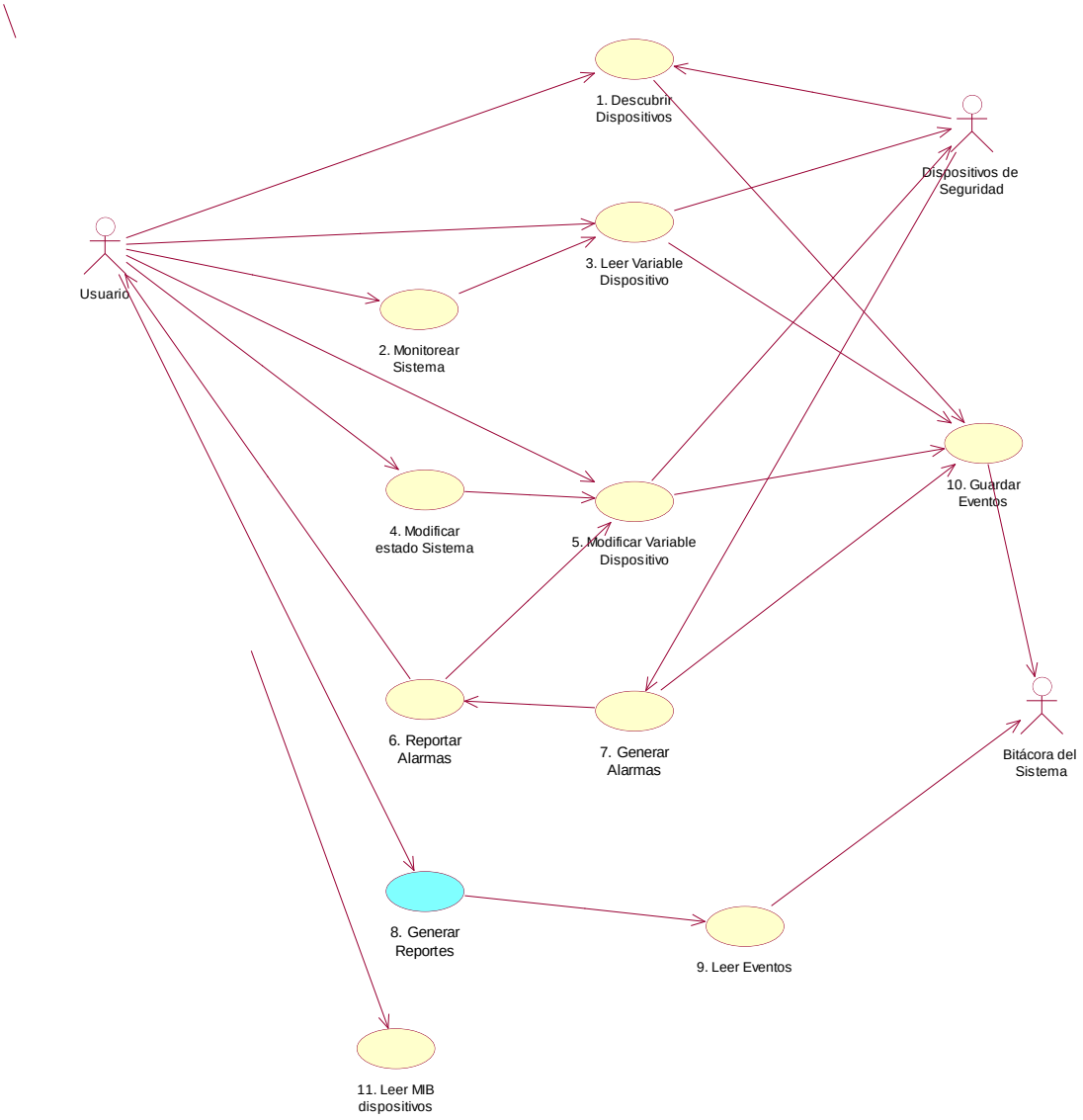


Figura 10. Se resalta en celeste el caso de uso que pueden implementar las aplicaciones administrativas

2.2.4. Diagrama de clases

De acuerdo con el análisis realizado y para cumplir con los requerimientos de un sistema de seguridad típico se han identificado las clases que se muestran en el siguiente diagrama:

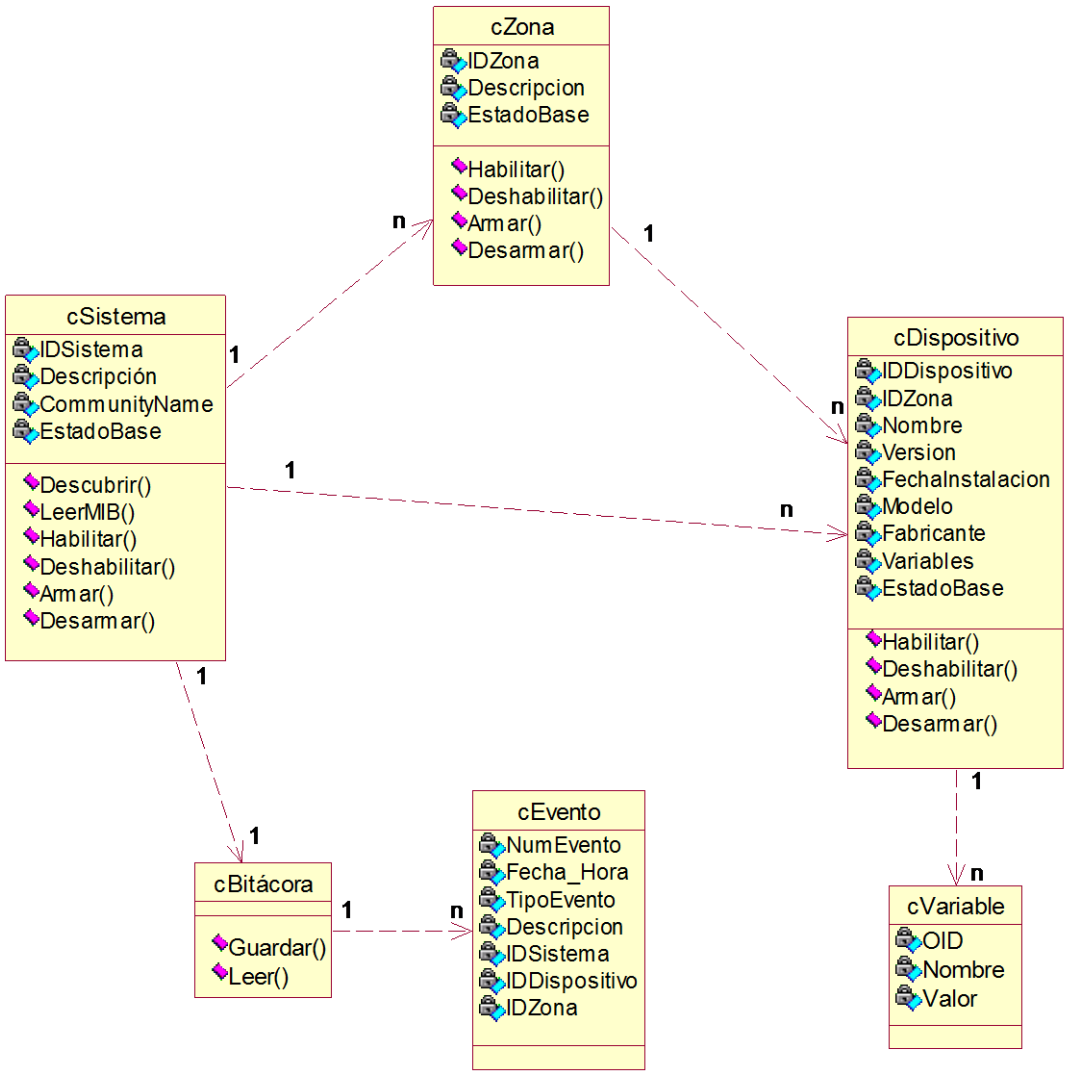


Figura 11. Diagrama de clases lógico para el diseño del SDK

A continuación se da una descripción de la funcionalidad de cada una de estas clases y los casos de uso que resuelven.

2.2.4.1. cSistema

Define el sistema como tal, con sus características generales y el conjunto de operaciones que se podrían realizar sobre todo el sistema.

Por medio de sus propiedades, eventos y métodos, realizará la implementación de los siguientes casos del uso:

- 1. Descubrir Dispositivos
- 2. Monitorear sistema
- 4. Modificar estado sistema
- 6. Reportar alarmas
- 11. Leer MIB

2.2.4.2. cZona

Se identificó con el fin de cumplir el requerimiento de agrupar los dispositivos por zonas, como lo hacen los sistemas de alarma tradicionales. Tiene la misma funcionalidad de monitoreo y control de la clase cSistema pero realiza su funcionalidad únicamente sobre los dispositivos de una zona.

Esta clase no resuelve ningún caso de uso. Se utiliza solamente para agrupar dispositivos.

2.2.4.3. cDispositivo

Implementa cada uno de los dispositivos descubiertos en el sistema. Contiene todos los atributos y operaciones que permiten conocer las características, el estado y controlar los dispositivos.

Resuelve los siguientes casos de uso:

- 3. Leer variable dispositivo
- 5. Modificar variable dispositivo
- 7. Generar alarmas

2.2.4.4. cBitacora

Almacena y retorna información acerca de todos los eventos que ocurren en el sistema. Éstos pueden ser generados por cualquier dispositivo u aplicación administrativa que los controle.

Implementa los siguientes casos de uso:

- 9. Leer eventos
- 10. Guardar eventos

2.2.4.5. cEvento

Es utilizada por cBitacora para el almacenamiento y acceso a eventos, y en conjunto implementan los casos de uso 9 y 10.

2.2.5. Estados Base

Una decisión de diseño es la definición y la manera en que se controlarán los estados del sistema. No es posible limitar los posibles estados, dado que debe controlarse cualquier dispositivo, y la cantidad de estados y su definición puede variar.

Es por esto que se ha definido un conjunto de estados base que deben cumplir todos los dispositivos para poder ser administrados. Cada dispositivo específicamente podrá, por medio de sus variables, implementar otros estados.

Cada uno de los dispositivos y zonas, así como el sistema en general, tienen un atributo llamado **EstadoBase**, que se utiliza para controlar estos estados, los cuales se describen en la siguiente tabla:

Estados	Definición
Armado	Estado en que el dispositivo esta activado y realizando sondeos para determinar si ha sido violada la seguridad.
Habilitado	Si un dispositivo se encuentra habilitado no genera alarmas, pero se encuentra listo para ser armado en cualquier momento
Deshabilitado	Un dispositivo deshabilitado no genera señales de alarma aunque las detecte. Cuando se arma la alarma no se arman los dispositivos deshabilitados
Alarma	El dispositivo se coloca en estado de alarma mientras se encuentre detectando e informando de cualquier situación anómala.
Falla	Estado en que el dispositivo envía una notificación señalando alguna deficiencia de su mecanismo.

Tabla 1. Estados base para el sistema y los dispositivos

En la siguiente figura se muestra un diagrama de estados y las condiciones que provocan los cambios de estado:

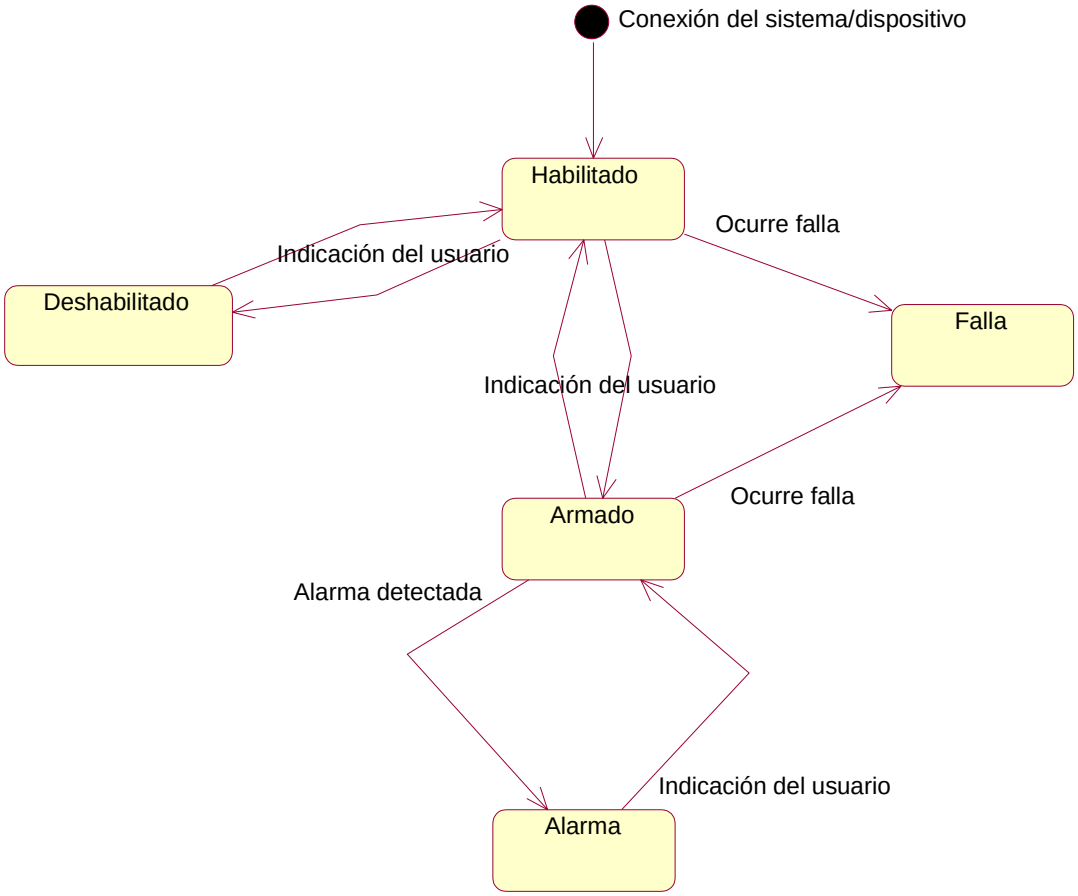


Figura 12. Diagrama de los estados base para el sistema y los dispositivos

2.3. Implementación del SDK

Con base en el diseño realizado se desarrolló el SDK para administración de dispositivos inteligentes. A continuación se detallan la implementación, la tecnología utilizada y el modelo de objetos que un desarrollador podrá utilizar.

2.3.1. Tecnología utilizada

La implementación se dividió en dos partes:

a. El control interno de los dispositivos, las zonas y la bitácora.
b. La traducción de todas las operaciones a mensajes SNMP.

Para el control interno se utilizó código en C++ junto con la biblioteca SNMP++ y Dart PowerTCP 4.1 for Visual C++ para generar clases e interfaces con ATL (ActiveX Template Library) y obtener componentes COM. Y para la implementación de las interfaces de usuario y los agentes se utilizó Visual Basic.

Dado que se ha planteado la necesidad de utilizar el SDK desde distintos lenguajes de programación se ha utilizado la tecnología COM para proveer al SDK de una interfaz única que puede ser utilizada desde cualquier aplicación que soporte este estándar.

Una vez finalizada la implementación, se obtuvo como resultado un solo componente COM accesible mediante una biblioteca de vínculos dinámicos (DLL – Dynamic Link Library) llamado SDKCI_SNMP.DLL (**SDK** para **C**asas **I**nteligentes basado en **SNMP**).

Este componente implementa las clases desarrolladas y permite acceder a toda la funcionalidad del SDK con el fin de crear aplicaciones que controlen un sistema de seguridad.

Para acceder a las clases y a la jerarquía de objetos dentro del componente COM se ha implementado un modelo de objetos basado en el diagrama de clases de la Figura 11.

2.3.2. Modelo de objetos

El modelo de objetos de la Figura 13 ilustra el modelo de objetos implementado en el componente COM. Indicando por medio de óvalos los objetos y por medio de rectángulos las colecciones que implementan algunas de las relaciones entre los objetos.

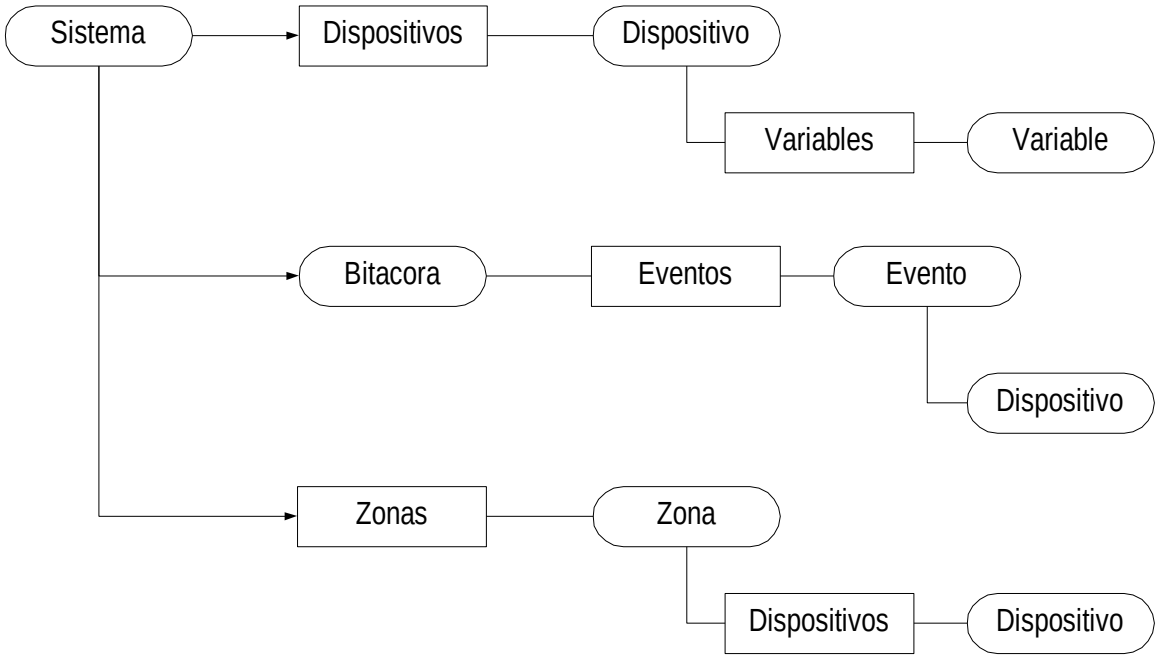


Figura 13. Modelo de objetos que se implementa en el SDK para brindar acceso de alto nivel al sistema de seguridad

El objeto padre en la jerarquía es el sistema como tal, el cual corresponde a la implementación de la clase cSistema identificada en la etapa de diseño.

A partir del sistema se puede acceder a todas las demás clases del modelo de objetos, la forma de utilizar este modelo se detallará más adelante en la sección “Especificación del SDK”.

2.3.2.1. Colecciones en el modelo de objetos

Un aspecto importante de la implementación es el uso de colecciones con el fin de obtener las relaciones 1:n entre las clases que se ilustran en la Figura 11. Así por ejemplo, para reflejar el hecho de que la clase cSistema tiene una relación 1:n con la clase cDispositivo, se implementó una colección de dispositivos y se colocó como parte de la clase Sistema.

2.4. Diseño e implementación de agentes

Un agente es una aplicación sencilla que recibe mensajes SNMP, puede generar Traps SNMP, y provee la interfaz de comunicación con el hardware que administra.

Normalmente es parte integral del hardware, incluido dentro de la misma circuitería del equipo que se conecta a la red, por lo que el control sobre él es directo. Además, cada agente de dispositivo tiene su propia dirección IP que lo hace accesible desde las aplicaciones administrativas.

Basa su funcionamiento en la información administrable del dispositivo, que debe encontrarse en el MIB (Management Information Base) para el equipo, que debe incluirse al adquirir el equipo y es el elemento que permite a cualquier aplicación administrativa SNMP conocer las variables y los traps que el dispositivo puede recibir y generar.

El desarrollo de cada agente depende del diseño de los MIBs para el dispositivo. Y para su implementación en este trabajo se utilizó la biblioteca de Dart PowerTCP SNMP Tool, que provee un control ActiveX para un agente estándar y al que se pueden agregar las variables y traps de los MIBs diseñados.

2.5. Diseño de MIBs

La base de datos de información de administración (MIB por sus siglas en inglés) es la que permite la comunicación efectiva entre una aplicación administrativa y un agente.

Cada agente en una red administrada por SNMP conoce las variables de su MIB y puede responder a ellas. Además, conoce y puede generar un conjunto de traps. Sin embargo, es imposible que una aplicación administrativa pueda conocer las variables de todos los posibles dispositivos; pues cabe recordar que el objetivo es administrar cualquier dispositivo, sin importar sus características.

Es por esto que el MIB se convierte en parte esencial del control con SNMP: la aplicación administrativa puede leer y compilar el MIB de un dispositivo, y con esta información conoce todas sus variables administrables y todos los posibles traps que puede generar.

2.5.1. Enterprise Number

Cada empresa que desee desarrollar MIBs debe seguir los estándares establecidos por los RFCs 1155, 1156 y 1157. Que definen el protocolo, la estructura y numeración de las variables en el MIB.

Parte de esta estructura es la asignación de un número de empresa (Enterprise Number) que permita crear ramificaciones de la estructura del MIB sin preocuparse por conflictos con otros fabricantes.

La organización encargada de la asignación de estos números es la IANA (Internet Assigned Numbers Authority) (www.iana.org). A través de un formulario en su sitio Web se realizó la solicitud de un número de empresa a nombre del Instituto Tecnológico de Costa Rica.

En el Anexo 1 se incluye el e-mail con que se confirma la asignación de este número de empresa por la IANA.

2.5.2. Enterprise Number para el ITCR y estructura interna

El número asignado a la institución fue el 9283, por lo que siguiendo la jerarquía del MIB SNMP cualquier objeto definido para este trabajo deberá quedar dentro de la siguiente rama del MIB:

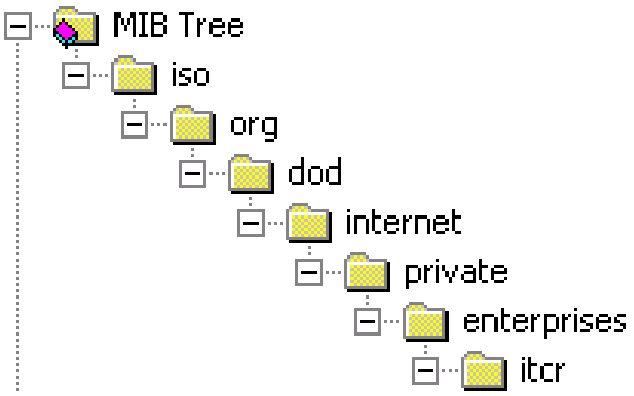


Figura 14. Estructura jerárquica del MIB hasta llegar al número asignado al ITCR

O por numeración:

1.3.6.1.4.1.9283

Dentro de esta rama se ha diseñado una estructura interna para prever futuros desarrollos dentro de la institución.

El árbol del MIB para el ITCR que se muestra a continuación contiene esta estructura:

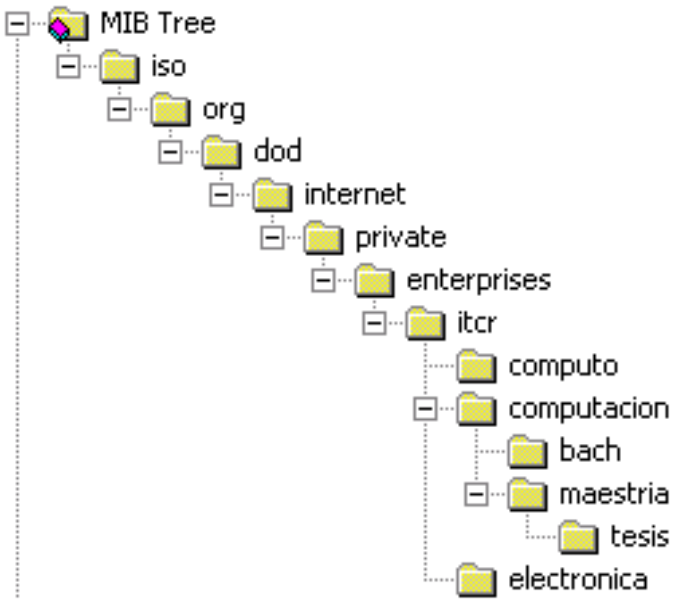


Figura 15. Diseño interno del MIB para el ITCR

Para definir esta estructura se creó el archivo ITCR.MIB que se puede compilar dentro de cualquier herramienta administrativa SNMP para poder administrar agentes desarrollados en la institución.

Este archivo es de texto plano en lenguaje ASN.1[1] y contiene lo siguiente:

```
ITCR-MIB DEFINITIONS ::= BEGIN

IMPORTS
    enterprises
        FROM RFC1155-SMI;

itcr         OBJECT IDENTIFIER ::= { enterprises 9283 }
computo      OBJECT IDENTIFIER ::= { itcr 1 }
computacion  OBJECT IDENTIFIER ::= { itcr 2 }
bach         OBJECT IDENTIFIER ::= { computacion 1 }
maestria     OBJECT IDENTIFIER ::= { computacion 2 }
tesis        OBJECT IDENTIFIER ::= { maestria 1 }
electronica  OBJECT IDENTIFIER ::= { itcr 3 }

END
```

[1] ASN.1 (Abstract Syntax Notation) es el lenguaje utilizado como estándar para la escritura de MIBs

En la siguiente tabla se listan las ramas (Object Identifiers u OIDs) SNMP para uso interno de la institución:

Rama	Object Identifier (OID) base
Itcr	1.3.6.1.4.1.9283
computo	1.3.6.1.4.1.9283.1
computacion	1.3.6.1.4.1.9283.2
bachillerato	1.3.6.1.4.1.9283.2.1
maestria	1.3.6.1.4.1.9283.2.2
tesis	1.3.6.1.4.1.9283.2.2.1
electronica	1.3.6.1.4.1.9283.3

2.5.3. MIB estándar para el SDKCI_SNMP

A la herramienta desarrollada en este trabajo (SDKCI_SNMP) se le asignó el número 1 dentro de la rama de tesis de maestría. Por lo que todos los dispositivos iniciarán en la rama 1.3.6.1.4.1.9283.2.2.1.1 y como estándar deberán contar con un conjunto base de variables que permitan administrarlos a través del SDK.

Este conjunto de variables se muestran como nodos hoja en la siguiente figura:

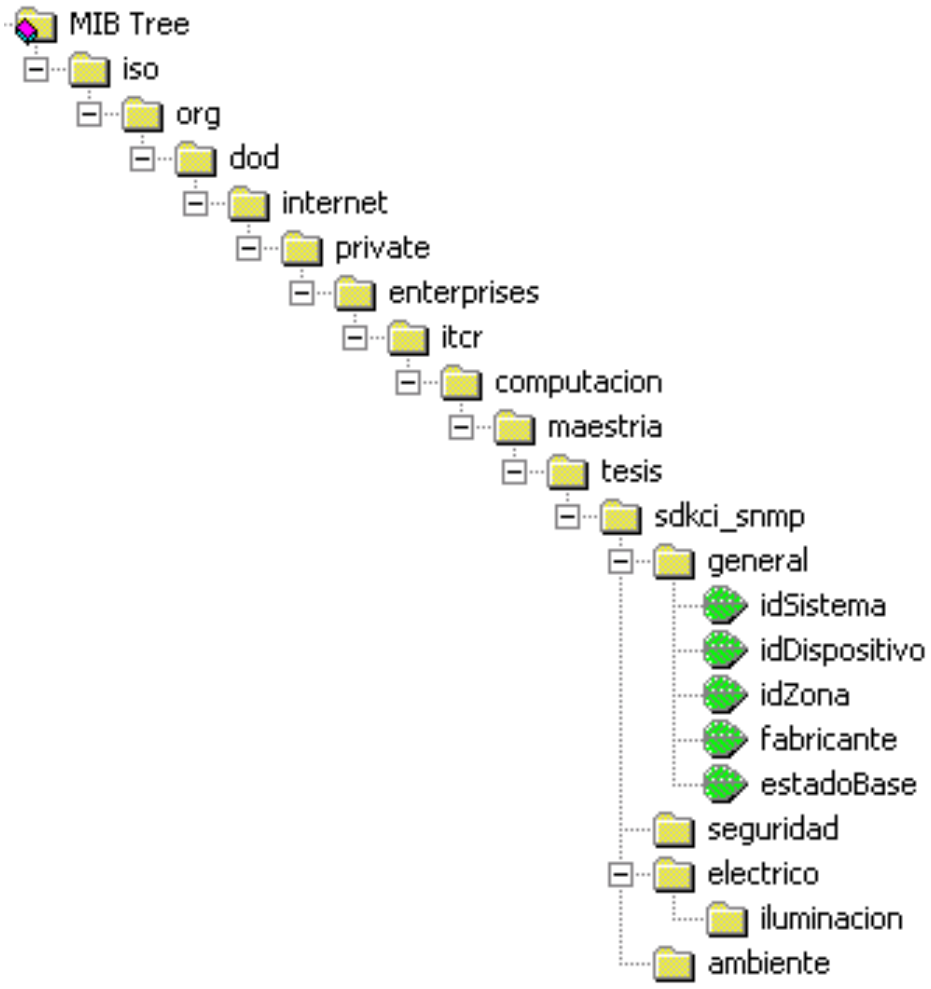


Figura 16. Conjunto de OIDs estándar para los dispositivos dentro del SDKCI_SNMP

Esto se encuentra en el archivo llamado SDKCI_SNMP.MIB que contiene el siguiente texto:

```
SDKCI_SNMP-MIB DEFINITIONS ::= BEGIN


IMPORTS
    tesis
        FROM ITCR-MIB;

-- Definicion del tipo base para el SDK
sdkci_snmp       OBJECT IDENTIFIER ::= { tesis 1 }
```

```
-- La rama GENERAL contendra todos los objetos comunes a
cualquier
-- dispositivo de cualquier sistema
general         OBJECT IDENTIFIER ::= { sdkci_snmp 1 }

-- Objetos especificos para sistemas de seguridad
seguridad        OBJECT IDENTIFIER ::= { sdkci_snmp 2 }
electrico        OBJECT IDENTIFIER ::= { sdkci_snmp 3 }
iluminacion      OBJECT IDENTIFIER ::= { electrico 1 }
ambiente         OBJECT IDENTIFIER ::= { sdkci_snmp 4 }


          idSistema OBJECT-TYPE
              SYNTAX  INTEGER
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
                      "Identificador del sistema al que
                       pertenece el dispositivo"
              ::= { general 1 }


          idDispositivo OBJECT-TYPE
              SYNTAX  INTEGER
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
                      "Un identificador unico asociado a
                       cada dispositivo"
              ::= { general 2 }


          idZona OBJECT-TYPE
              SYNTAX  INTEGER
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
                     "El numero de la zona a la que pertenece
                     el dispositivo"
              ::= { general 3 }
```

```
            fabricante OBJECT-TYPE
                SYNTAX  OCTET STRING
                ACCESS  read-only
                STATUS  mandatory

                DESCRIPTION
                        "Nombre del fabricante del dispositivo"
                ::= { general 4 }

            estadoBase OBJECT-TYPE
                SYNTAX  INTEGER
                ACCESS  read-write
                STATUS  mandatory
                DESCRIPTION
                        "Indicador del estado del dispositivo"
                ::= { general 5 }
END
```

Los MIBs diseñados para los dispositivos utilizados en las pruebas del SDK se describen en el capítulo “Aplicación Práctica del Modelo Propuesto”, y su código se muestra en el Anexo 3.

2.6. Especificación del SDK

En esta sección se incluye una especificación de cada uno de los componentes del SDK y su funcionamiento. Se indica el conjunto de clases que lo componen y las propiedades, eventos y métodos de cada uno de ellos, y además los ejemplos representativos en cada caso.

Se ha escogido Visual Basic como lenguaje cliente para el código de ejemplo, sin embargo, al ser COM un estándar, el SDK es utilizable desde cualquier lenguaje que lo soporte.

2.6.1. ¿En qué consiste SDKCI_SNMP?

SDKCI_SNMP es un modelo de objetos que permite interactuar con un conjunto de dispositivos inteligentes basados en SNMP desde un lenguaje de programación de alto nivel, sin preocuparse por los detalles de comunicación ni de implementación de los protocolos.

Se compone de un conjunto de componentes COM que proveen una interfaz muy sencilla pero que no limita al desarrollador a un tipo de aplicación o a un lenguaje específico. Se ha pretendido más bien que sea un modelo abierto y sencillo.

Se basa totalmente en el protocolo SNMP para el control de los dispositivos, sin embargo el desarrollador usuario no requiere conocer sus detalles para poder utilizar el SDKCI_SNMP, sino que mediante el modelo de objetos puede realizar todas las operaciones que requiere.

Este modelo de objetos se muestra en la siguiente figura:

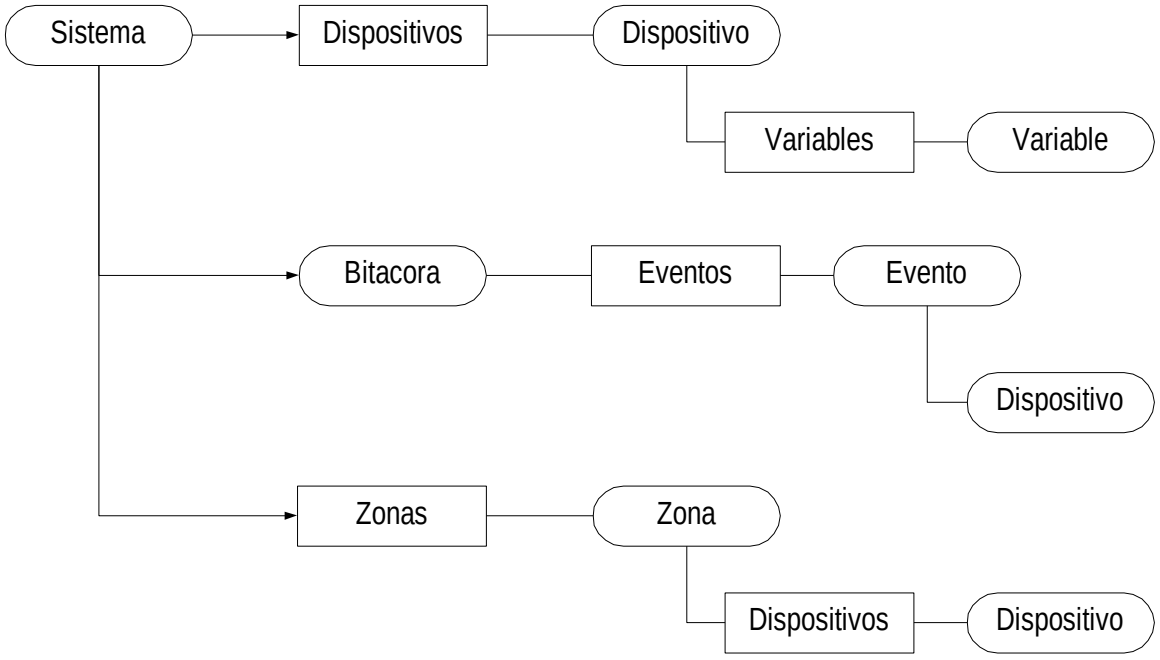


Figura 17. Modelo de objetos implementado en el componente COM SDKCI_SNMP

Este modelo implementa el conjunto de clases que se muestran en detalle en el siguiente diagrama:

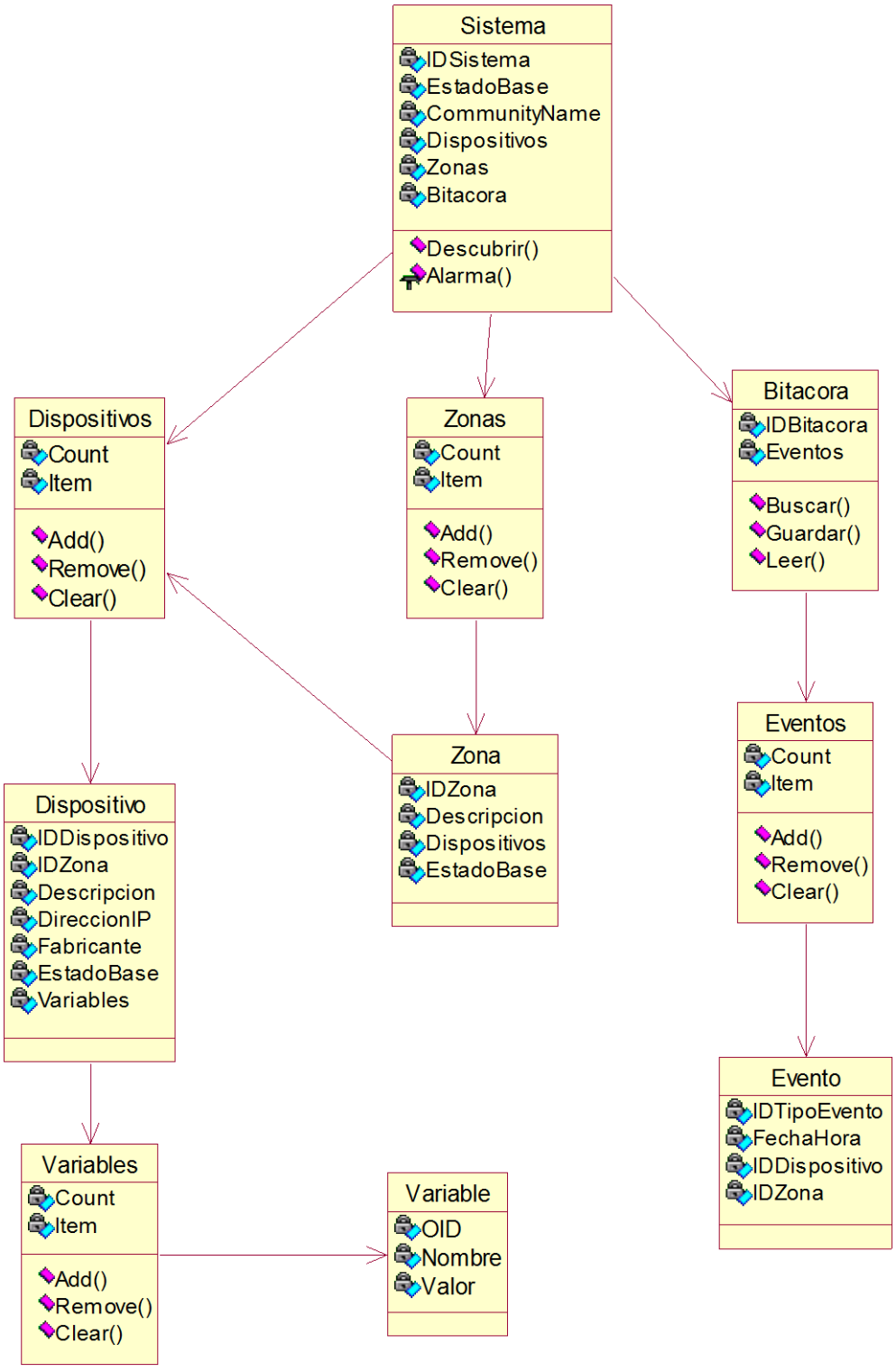
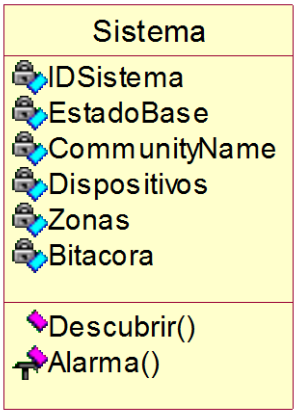


Figura 18. Diagrama de clases físico implementado en el SDKCI_SNMP

2.6.2. Referencia del modelo de objetos

A continuación se incluye la documentación completa para el desarrollador que desee utilizar el SDKCI_SNMP.

2.6.2.1. Clase Sistema



Descripción

Para poder utilizar toda la funcionalidad de SDKCI_SNMP es necesario crear un objeto Sistema (la clase de más alto nivel en el modelo de objetos) y a partir de sus propiedades acceder a todos las demás clases del modelo.

Sin embargo, si se desea desarrollar una aplicación para administrar un dispositivo específico, es posible entonces utilizar objetos de la clase *Dispositivo* en forma independiente del resto del modelo.

Declaración y uso

En cualquier lenguaje de programación antes de poder utilizar un componente COM es necesario declararlo y crear una instancia del objeto en memoria. Para esto se utilizan instrucciones específicas de cada lenguaje, sin embargo el esquema general siempre es el mismo:

1. Declarar la variable de tipo objeto
2. Crear una instancia de la clase en memoria
3. Utilizar la variable objeto
4. Destruir la instancia al terminar su uso

Ejemplo

```
'declaración
Public Sist as Sistema

'creación de una instancia
Set Sist = New Sistema

… uso del objeto …

'destrucción del objeto
Set Sist = Nothing
```

Propiedades

IDSistema

Descripción:

Identifica al sistema. En un ambiente de múltiples sistemas, permite asignar una identificación a cada uno de los sistemas administrados.

Sintaxis:

Visual Basic	`{varobjeto}.IDSistema = {numero As Long}`

donde:

varobjeto es una variable de tipo Sistema

Comentarios:

Esta propiedad puede utilizarse en caso de que se desee administrar distintos sistemas independientemente. Si no se especifica el valor por defecto es 1.

Ejemplo

```
Public Sist as Sistema

Set Sist = New Sistema

'se trabajara con el sistema ID 3
Sist.IDSistema = 3
```

EstadoBase

Descripción

Se ha definido un conjunto de estados básicos que cualquier dispositivo del sistema debe entender y soportar en su MIB.

Estos estados son los siguientes:

- Deshabilitado
- Desarmado
- Armado
- Alarma
- Falla

Sintaxis

Visual Basic	`{varobjeto}.EstadoBase = {número As eEstadoBase}`

donde:

varobjeto es una variable de tipo Sistema

eEstadoBase es una enumeración que contiene los posibles estados, de acuerdo a la siguiente declaración:

```
Public Enum eEstadoBase
    sdkciDeshabilitado = 0
    sdkciHabilitado = 1
    sdkciArmado = 2
    sdkciAlarma = 3
    sdkciFalla = 4
End Enum
```

Comentarios

Las constantes utilizadas para esta propiedad pueden utilizarse también para cambiar el estado de una zona o de un dispositivo.

Ejemplo

```
Private Sub bArmar
     Dim s as Sistema

     Set s = new Sistema
     s.Descubrir()

     s.EstadoBase = sdkciArmado
End Sub
```

CommunityName

Descripción

Esta propiedad contiene el nombre de comunidad para los dispositivos, de acuerdo a lo especificado por el protocolo SNMP. Los dispositivos responderán únicamente si se indica el nombre de comunidad correcto. Se puede utilizar como un mecanismo básico de seguridad para la administración de los dispositivos. El valor por defecto es *public*,

Sintaxis

Visual Basic	`{varobjeto}.CommunityName = {nombre As String}`

donde:

varobjeto es una variable de tipo Sistema

Comentarios

Si se intenta ejercer control sobre cualquier dispositivo utilizando un nombre de comunidad incorrecto, él generará un trap indicando a cualquier aplicación administrativa en la red el acceso inválido y la aplicación administrativa podrá decidir el curso de acción.

Ejemplo

```
Dim s as Sistema

Set s = new Sistema
s.CommunityName = "segundopiso"
```

Dispositivos

Descripción

Es una colección que contiene todos los objetos de tipo dispositivo descubiertos en el sistema. A través de ella es posible acceder a los dispositivos del sistema y trabajar con ellos. Es una colección con los métodos estándar de utilizados en modelos de objetos COM por lo que soporta las operaciones de colección. El índice base es 1.

Además, es posible acceder directamente a un dispositivo utilizando como llave su dirección IP.

Sintaxis

Visual Basic	`Acceso a un dispositivo por índice`
	`{varobjeto}.Dispositivos(indice as Long)`
	
	`Acceso a un dispositivo por dirección IP`
	`{varobjeto}.Dispositivos(DireccionIP as String)`

donde:

varobjeto es una variable de tipo Sistema

Comentarios

Esta colección contiene todos los dispositivos descubiertos o agregados manualmente al sistema, a diferencia de la colección Dispositivos de la clase Zona, la cuál contiene los dispositivos de la zona correspondiente.

Ejemplo

Este ejemplo coloca la dirección IP, el OID y la descripción de todos los dispositivos en un ListView llamado lvDispositivos.

```
Sub ActualizarListaAgentes()
    Dim Disp As Dispositivo
    Dim liFila As ListItem

    For Each Disp In Sist.Dispositivos
        Set l = lvDispositivos.ListItems.Add
        liFila.Text = Disp.DireccionIP
        liFila.SubItems(1) = Disp.oid
        liFila.SubItems(2) = Disp.Descripcion

        'si el dispositivo esta habilitado entonces
        'poner la fila en negrita
        If Disp.EstadoBase <> sdkciHabilitado Then
            liFila.Bold = True
        End If
    Next
End Sub
```

Zonas

Es una colección que contiene todas las zonas a través de las que se ha organizado el sistema. A través de ella es posible crear nuevas zonas, modificar las existentes, habilitar o deshabilitar los dispositivos de una zona. Es una colección con los métodos estándar de utilizados en modelos de objetos COM por lo que soporta las operaciones de colección. El índice base es 1.
Además, es posible acceder directamente a una zona utilizando como llave su número.

Sintaxis

Visual Basic	`Acceso a una zona por índice` `{varobjeto}.Zonas(indice as Long)`

donde:

varobjeto es una variable de tipo Sistema

Comentarios

Inicialmente se crea la zona 1 y todos los dispositivos se encuentran en esta zona. Luego una aplicación administrativa podrá permitir al usuario organizar los dispositivos por zonas a su gusto.

Si se descubre en el sistema un nuevo dispositivo, se colocará inicialmente en la zona 1.

Ejemplo

Este ejemplo crea una nueva zona 2 y mueve el dispositivo indicado como parámetro de la zona 1 a la 2.

```
Sub EjemploZonas(direccionIP as String)
     Dim zNueva as Zona
     Dim z as Zona

     Set zNueva = Sist.Zonas.Add

     'coloca el dispositivo en la nueva zona
     Sist.Dispositivos(direccionIP).IDZona = zNueva.IDZona

    For Each z in Sist.Zonas
          Debug.Print z.EstadoBase
    Next
End Sub
```

Bitácora

Descripción

Permite tener un registro interno de todos los eventos ocurridos en el sistema durante la ejecución. Permite leer archivos de bitácora almacenados por el administrador y guardarlos. Los archivos de bitácora se almacenan utilizando XML.

Sintaxis

Visual Basic	`{varobjeto}.Bitacora`

donde:
varobjeto es una variable de tipo Sistema

Comentarios

Cada objeto sistema tiene una sola bitácora que contiene una colección con los eventos ocurridos.

Ejemplo

```
Function CantEventos()
     CantEventos = Sist.Bitacora.Eventos.Count
End Function

Sub LimpiarBitacora()
     Sist.Bitacora.Eventos.Clear
End Sub
```

Eventos

Alarma

Descripción

Se produce para informar a la aplicación que está utilizando el SDK acerca de la ocurrencia de una alarma, indicando en forma de parámetro el dispositivo que produjo la alarma.

Sintaxis

Visual Basic	`Private Sub Sist_Alarma(DispGenera As Dispositivo)` `End Sub`

Comentarios

Si ocurren en forma continua alarmas en distintos dispositivos, se producen múltiples eventos alarma que una aplicación administrativa puede recibir y procesar.

Ejemplo

```
Private Sub Sist_Alarma(DispGenera As SDKCI_SNMP.Dispositivo)

    MsgBox "Se produjo una alarma en el dispositivo " & _
                DispGenera.Descripcion

End Sub
```

Métodos

Descubrir

Descripción

Realiza el descubrimiento de los agentes existentes en la red, y actualiza la colección de dispositivos.

Sintaxis

Visual Basic	`{varobjeto}.Descubrir`

donde:

varobjeto es una variable de tipo Sistema

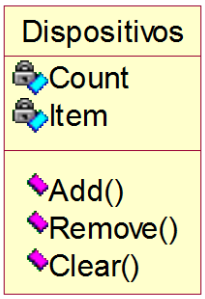
Comentarios

Dependiendo de las características de rendimiento de la red, el descubrimiento de dispositivos puede no ser inmediato, sino que puede llegar a tardar hasta varios segundos. Por lo tanto es recomendable dar cierto tiempo (al menos 1 segundo) antes de consultar la colección de dispositivos.

Ejemplo

```
Dim Sist as Sistema

Private Sub bDescubrir_Click()

    Set Sist = New Sistema

    Sist.Descubrir

    'actualizar un Timer para que muestre los
    'dispositivos un segundo despues
    timerActualizar.Interval = 1000
End Sub


Sub TimerActualizar_Timer()
    Dim Disp As Dispositivo
    Dim liFila As ListItem

    'limpiar la lista en pantalla
    lvDispositivos.ListItems.Clear

    For Each Disp In Sist.Dispositivos
        Set liFila = lvDispositivos.ListItems.Add
        liFila.Text = Disp.DireccionIP
        liFila.SubItems(1) = Disp.OID
        liFila.SubItems(2) = Disp.Descripcion
    Next
    timerActualiz.Interval = 0
End Sub
```

2.6.2.2. Colección Dispositivos



Propiedades

Count

Descripción

Indica la cantidad de dispositivos en la colección.

Sintaxis

Visual Basic	`{varobjeto}.Count`

donde:

varobjeto es una variable de tipo colección Dispositivos

Comentarios

Propiedad solo-lectura

Ejemplo

```
Sub MostrarCantidadDispositivos()
     Dim Cantidad as Long

     Cantidad = Sist.Dispositivos.Count
     If Cantidad > 0 then
          Msgbox "Cantidad de dispositivos: " & Cantidad
     Else
          Msgbox "El sistema no tiene dispositivos"
     End if
End Sub
```

Item

Descripción

Permite acceso individual a cada uno de los dispositivos en la colección.

Sintaxis

Visual Basic	Acceso a un dispositivo por índice {varobjeto}.Dispositivos.Item(indice as Long) Acceso a un dispositivo por dirección IP {varobjeto}.Dispositivos.Item(Dirección as String)

Comentarios

Propiedad por defecto para la colección Dispositivos. El posible acceder al arreglo por índice numérico (Long) o utilizando como llave la dirección IP del dispositivo.

Ejemplo

```
Sub DeshabilitarDispositivo(DireccionIP as String)
     Dim Disp as dispositivo

     Set Disp = Sist.Dispositivos.Item(direccionIP)

     Disp.EstadoBase = sdkciDeshabilitado

End Sub
```

Métodos

Add

Descripción

Permite agregar un dispositivo a la colección Dispositivos.

Sintaxis

Visual Basic	`{varobjeto}.Add(Disp as Dispositivo)`

donde:

varobjeto es una variable de tipo colección Dispositivos

Comentarios

Si no se indica lo contrario, el objeto agregado a la colección se coloca por defecto en la zona 1.

Ejemplo

```
Dim Disp as dispositivo

Set Disp = New Dispositivo
Disp.DireccionIP = "10.10.10.22"
Disp.IDDispositivo = 14
Disp.IDZona = 2

Sist.Dispositivos.Add Disp
```

Remove

Descripción

Permite eliminar un dispositivo de la coleccion Dispositivos

Sintaxis

Visual Basic	`{varobjeto}.Remove([Indice as Long], [DireccionIP as String])`

Donde:

Varobjeto: Es una variable de tipo colección Dispositivos

Indice: Opcional. Es el índice en la colección del dispositivo que se desea eliminar

DireccionIP: Opcional. Es la dirección IP del dispositivo que se desea eliminar.

Comentarios

Es posible eliminar el dispositivo indicando su índice dentro de la colección o su dirección IP.

Al eliminar un dispositivo de la colección ya no se enviarán comandos de control al agente ni se recibirán alarmas de él.

Ejemplo

```
Sub EliminarDisp(direcccionIP as string)
     Sist.Dispositivos.Remove direccionIP
End Sub
```

Clear

Descripción

Limpia la colección de dispositivos, eliminando todos los objetos que contiene.

Sintaxis

Visual Basic	`{varobjeto}.Clear`

Donde:

Varobjeto: Es una variable de tipo colección Dispositivos

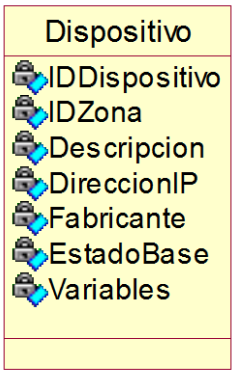
Comentarios

La colección se limpia automáticamente al utilizar el método Descubrir.

Ejemplo

```
Sub Limpiar()

     Sist.Dispositivos.Clear

End Sub
```

2.6.2.3. Clase Dispositivo



Propiedades

IDDispositivo

Descripción

Identifica al dispositivo. Es asignado automáticamente al agregar el dispositivo o al realizar el descubrimiento.

Sintaxis

Visual Basic	`{varobjeto}.IDDispositivo = {numero As Long}`

donde:

varobjeto es una variable de tipo Dispositivo

Comentarios

Propiedad de solo lectura, contiene un consecutivo asignado por el sistema a cada dispositivo. Es posible que en distintas ocasiones que se ejecute una aplicación, el mismo dispositivo pueda tener distintos Ids. Por tanto se recomienda acceder a los dispositivos mediante su OID o su dirección IP.

Ejemplo

```
Sub MostrarIDDispositivo(DireccionIP as string)
     Dim Disp as Dispositivo

     Set Disp = Sist.Dispositivos(DireccionIP)

     Msgbox Disp.IDDispositivo
End Sub
```

IDZona

Descripción

Identifica la zona a la que pertenece el dispositivo.

Sintaxis

Visual Basic	`{varobjeto}.IDZona = {numero As Long}`

donde:

varobjeto es una variable de tipo Dispositivo

Comentarios

Inicialmente cualquier dispositivo es asignado a la zona 1, y mediante esta propiedad se puede asignar a otra zona.

Ejemplo

```
Sub MostrarZonaDispositivo(DireccionIP as string)
     Dim Disp as Dispositivo

     Set Disp = Sist.Dispositivos(DireccionIP)

     Msgbox Disp.IDZona
End Sub
```

Descripcion

Descripción

Contiene la descripción indicada por el fabricante para el dispositivo.

Sintaxis

Visual Basic	`{varobjeto}.Descripcion = {numero As String}`

donde:

varobjeto es una variable de tipo Dispositivo

Comentarios

Contiene inicialmente el string asociado por el fabricante al dispositivo. Es la propiedad equivalente al objeto sysDescr de SNMP.

Ejemplo

```
Sub MostrarDescrDispositivo(DireccionIP as string)
     Dim Disp as Dispositivo

     Set Disp = Sist.Dispositivos(DireccionIP)

     Msgbox Disp.Descripcion
End Sub
```

DireccionIP

Descripción

Indica y permite modificar la dirección IP de un dispositivo.

Sintaxis

Visual Basic	`{varobjeto}.DireccionIP = {numero As Long}`

donde:

varobjeto es una variable de tipo Dispositivo

Comentarios

Si se modifica la dirección IP de un dispositivo, el acceso al dispositivo mediante las colecciones debe realizarse a través de la nueva dirección IP. Para el caso de algunos dispositivos el cambio de dirección puede no ser inmediato, sino requerir tiempo o el reinicio del dispositivo.

Ejemplo

```
Sub ModifDireccionIP(IPOriginal, NuevaIP)
     Dim Disp as Dispositivo

     Set Disp = Sist.Dispositivos(IPOriginal)


     Disp.DireccionIP = NuevaIP
End Sub
```

EstadoBase

Descripción

Se ha definido un conjunto de estados básicos que cualquier dispositivo del sistema debe entender y soportar en su MIB.

Estos estados son los siguientes:

- Deshabilitado
- Habilitado
- Armado
- Alarma
- Falla

Sintaxis

Visual Basic	`{varobjeto}.EstadoBase = {numero As eEstadoBase}`

donde:

varobjeto es una variable de tipo Dispositivo

eEstadoBase es una enumeración que contiene los posibles estados, de acuerdo a la siguiente declaración:

```
Public Enum eEstadoBase
    sdkciDeshabilitado = 0
    sdkciHabilitado = 1
    sdkciArmado = 2
    sdkciAlarma = 3
    sdkciFalla = 4
End Enum
```

Comentarios

Las constantes utilizadas para esta propiedad pueden utilizarse también para cambiar el estado de una zona o del sistema en su totalidad.

Ejemplo

```
Private Sub DeshabilitarDisp(DireccionIP as String)
     Dim Disp as Dispositivo

     Set Disp = Sist.Dispositivos.Item(DireccionIP)

     Disp.EstadoBase = sdkciDeshabilitado

End Sub
```

Fabricante

Descripción

Contiene la información acerca del fabricante del dispositivo.

Sintaxis

Visual Basic	`{varobjeto}.Fabricante = {Valor As String}`

donde:

varobjeto es una variable de tipo Dispositivo

Comentarios

Propiedad de solo lectura. Contiene el string asignado por el fabricante al dispositivo.

Ejemplo

```
Sub MostrarFabricDispositivo(DireccionIP as string)
     Dim Disp as Dispositivo

     Set Disp = Sist.Dispositivos(DireccionIP)

     Msgbox Disp.Fabricante
End Sub
```

Variables

Descripción

Esta colección contiene todas las variables programables específicas para cada dispositivo.

Sintaxis

Visual Basic	'acceso a una variable por índice {varobjeto}.Variables(indice As Long) 'acceso a una variable por nombre {varobjeto}.Variables(nombre As String)

donde:

varobjeto es una variable de tipo Dispositivo

Comentarios

Esta colección es la que brinda la flexibilidad de administrar cualquier tipo de dispositivo, pues su contenido se basa en el MIB del dispositivo.

Para que esta propiedad funcione adecuadamente el SDK debe conocer el MIB para el dispositivo, esto se logra utilizando el método LeerMIB de la clase Sistema. Todas las variables del MIB se convierten en elementos de esta colección, y a través de ella es posible leer y modificar el valor de estas variables.

Ejemplo

```
Sub VerVariable(DirIP as string, var as String)
     Dim Disp as Dispositivo

     Set Disp = Sist.Dispositivos(DirIP)

     Msgbox Disp.Variables(var).Nombre & ": " & _
            Disp.Variables(var).Valor
End Sub
```

2.6.2.4. Clase Variable

Propiedades

OID

Descripción

Contiene el identificador único asignado por el protocolo SNMP (Object ID) de la variable.

Sintaxis

Visual Basic	`{varobjeto}.OID = {valor As String}`

Donde:

Varobjeto: es un objeto de clase Variable

Comentarios

Propiedad de solo lectura, contiene el número de identificación asignado a la variable en el MIB SNMP.

Ejemplo

```
Sub VerOID(DirIP as string, var as String)
     Dim Disp as Dispositivo

     Set Disp = Sist.Dispositivos(DirIP)

     Msgbox Disp.Variables(var).OID
End Sub
```

Nombre

Descripción

Contiene el nombre asignado por el MIB del protocolo SNMP a la variable.

Sintaxis

Visual Basic	`{varobjeto}.Nombre = {valor As String}`

Donde:

Varobjeto: es un objeto de clase Variable

Comentarios

Propiedad de solo lectura, contiene el nombre asignado a la variable en el MIB SNMP.

Ejemplo

```
Sub VerNombres(DirIP as string)
     Dim Disp as Dispositivo
     Dim v as Variable

     Set Disp = Sist.Dispositivos(DirIP)

     For each v in Disp.Variables
          Msgbox v.Nombre
     Next
End Sub
```

Valor

Descripción

Contiene el nombre valor de una variable.

Sintaxis

Visual Basic	`{varobjeto}.Valor = {valor As Variant}`

Donde:

Varobjeto: es un objeto de clase Variable

Comentarios

Por medio de esta propiedad es posible leer y modificar el valor de cualquier variable del MIB SNMP.

Ejemplo

```
Sub CambiarSonido(DirIP as string)
     Dim Disp as Dispositivo

     Set Disp = Sist.Dispositivos(DirIP)

     Disp.Variables("tipoSonido") = 3

End Sub
```

2.6.2.5. Clase Bitácora

Propiedades

IDBitácora

Descripción

Identifica la bitácora. Es asignado automáticamente al crear la variable tipo Bitácora al momento de inicializar el Sistema.

Sintaxis

Visual Basic	`{varobjeto}.IDBitacora = {numero As Long}`

donde:

varobjeto es una variable de tipo Bitacora

Comentarios

Propiedad de solo lectura, contiene un consecutivo asignado por el sistema a cada bitácora. Si se almacena o se lee una bitácora se utiliza este consecutivo para identificarla.

Ejemplo

```
Sub MostrarIDBitacora()
     Msgbox Sist.IDBitacora
End Sub
```

Eventos

Descripción

Contiene la colección de eventos almacenados en la bitácora.

Visual Basic	`'acceso a la colección por índice` `{varobjeto}.Eventos(indice as Long)`

donde:

varobjeto es una variable de tipo Bitacora

Comentarios

La colección de eventos contiene todos los eventos almacenados en la bitácora. Si se desea almacenar una bitácora o leer una bitácora almacenada, se pueden utilizar los métodos Leer y Guardar de la clase Bitácora, respectivamente.

Métodos

Buscar

Descripción

Sintaxis

Visual Basic	`{varobjeto}.Buscar([TipoEvento as Long], [FHInicio as date], [FHFin as Date], [DirIP as string)) As Eventos`

Donde:

VarObjeto: Es una variable de la clase Bitacora

TipoEvento: Opcional. Es el tipo de eventos que se desea buscar

FHInicio: Opcional. La fecha y hora de inicio de los eventos que se desea buscar

FHFin: Opcional. La fecha y hora final de los eventos que se desea buscar

DirIIP: Opcional. Direccion IP del dispositivo que generó los eventos que se desea buscar.

Comentarios

Este método permite filtrar los eventos que se encuentran en la bitácora. Es posible combinar los parámetros del método como criterios de búsqueda.
Retorna una colección con los eventos que cumplen los criterios de búsqueda indicados.

Ejemplo

```
Sub EjemploBuscar()
     Dim ev as Eventos
     Dim fInicio as Date
     Dim fFin as Date

     'Buscar por tipo de evento
     Set ev = Sist.Bitacora.Eventos.Buscar(1)

     'Buscar por fecha y hora de inicio y fin
     fInicio = "15/5/2001 4:50pm"
     fFin = "15/5/2001 5:50pm"
     set ev = Sist.Bitacora.Eventos.Buscar(,fInicio, fFin)

     'Buscar por direccion IP del dispositivo
     set ev = Sist.Bitacora.Eventos.Buscar(,,,"10.10.1.3")

End Sub
```

Guardar

Descripción

Almacena los eventos de la bitácora actual en el archivo indicado.

Sintaxis

Visual Basic	`{varobjeto}.Guardar(ArchivoXML as String, [ev as Eventos])`

donde:

varobjeto es una variable de tipo Bitacora

ArchivoXML: Es el nombre del archivo con que se desea almacenar la lista de eventos.

Ev: Opcional. Corresponde a la colección de eventos que se desea almacenar el archivo.

Comentarios

Si no se especifica una colección de eventos a almacenar, se almacenarán todos los eventos de la bitácora.

Ejemplo

```
Sub GuardarBitacora()
     Dim ev as Eventos

     Sist.Bitacora.Guardar "lunes.xml"

     set ev = Sist.Bitacora.Eventos.Buscar(,,,"10.10.1.3")
     Sist.Bitacora.Guardar "d:\eventos.xml", ev

End Sub
```

Leer

Descripción

Lee eventos de un archivo de bitácora almacenado y los carga en una colección Eventos.

Sintaxis

Visual Basic	`{varobjeto}.Leer(ArchivoXML as String, [ev as Eventos])`

donde:

varobjeto es una variable de tipo Bitacora

ArchivoXML: Es el nombre del archivo con que se desea almacenar la lista de eventos.

Ev: Opcional. Corresponde a la colección de eventos en que se desea cargar los eventos leídos del archivo.

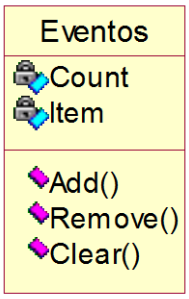
Comentarios

Si no se especifica una colección de eventos para almacenar lo leído del archivo, se almacenarán todos los eventos en la colección de eventos de la bitácora del sistema, sin eliminar los eventos que ya existían en ella.

Ejemplo

```
Sub LeerBitacora()
     Dim ev as Eventos

     Sist.Bitacora.Leer "lunes.xml"

     Sist.Bitacora.Leer "d:\eventos.xml", ev

End Sub
```

2.6.2.6. Colección Eventos



Propiedades

Count

Descripción

Indica la cantidad de eventos en la colección.

Sintaxis

Visual Basic	`{varobjeto}.Count`

donde:

varobjeto es una variable de tipo colección Eventos

Comentarios

Propiedad solo-lectura

Ejemplo

```
Sub MostrarCantidadEventos()
     Dim Cantidad as Long

     Cantidad = Sist.Bitacora.Eventos.Dispositivos.Count
     If Cantidad > 0 then
          Msgbox "Cantidad de eventos: " & Cantidad
     Else
          Msgbox "No han ocurrido eventos"
     End if
End Sub
```

Item

Descripción

Permite acceso individual a cada uno de los eventos en la colección.

Sintaxis

Visual Basic	`{varobjeto}.Item(Indice as Long)`

Donde

Varobjeto es una variable tipo colección Eventos

Comentarios

Propiedad por defecto para la colección Eventos. El posible acceder al arreglo por índice numérico del evento (Long).

Ejemplo

```
Sub MostrarEvento(Num as Long)
     Dim eEvento as Evento

     Set eEvento = Sist.Bitacora.Eventos.Item(Num)

     Msgbox eEvento.Descripcion

End Sub
```

Métodos

Add

Descripción

Permite agregar un evento a la colección Eventos.

Sintaxis

Visual Basic	`{varobjeto}.Add(Ev as Evento)`

donde:

varobjeto es una variable de tipo colección Eventos

Ev: el objeto tipo Evento que se desea agregar a la colección

Comentarios

Los eventos del sistema se agregan automáticamente a la colección, sin embargo, una aplicación puede utilizar este método para manipular sus propios eventos de administración.

Ejemplo

```
Dim eEvento as Evento

Set eEvento = New Dispositivo
eEvento.IDEvento = 20001
eEvento.Descripcion = "Contraseña Inválida"

Sist.Bitacora.Eventos.Add eEvento
```

Remove

Descripción

Permite eliminar un evento de la colección Eventos

Sintaxis

Visual Basic	`{varobjeto}.Remove(Indice as Long)`

Donde:

Varobjeto: Es una variable de tipo colección Eventos

Indice: Es el índice en la colección del evento que se desea eliminar

Comentarios

Si se desea eliminar todos los eventos de la colección se recomienda utilizar el método Clear.

Ejemplo

```
Sub EliminarDisp(Indice as Long)
     Sist.Bitacora.Eventos.Remove Indice
End Sub
```

Clear

Descripción

Limpia la colección Eventos, eliminando todos los objetos que contiene.

Sintaxis

Visual Basic	`{varobjeto}.Clear`

Donde:

Varobjeto: Es una variable de tipo colección Eventos

Comentarios

Antes de limpiar la colección es posible almacenar los eventos utilizando el método *Guardar* de la clase *Bitacora*.

Ejemplos

```
Sub LimpiarBitacora()

     Sist.Bitacora.Guardar "bitacora.xml"
     Sist.Bitacora.Eventos.Clear

End Sub
```

2.6.2.7. Clase Evento

IDEvento

Descripción

Identificador consecutivo asociado a cada evento al agregarlo a la bitácora

Sintáxis

Visual Basic	`{varobjeto}.IDEvento = {numero As Long}`

donde:

varobjeto es una variable de tipo Bitacora

Comentarios

Propiedad de solo lectura, contiene un consecutivo asignado por el sistema a cada evento de la bitácora. Si se almacena o se lee evento se utiliza este consecutivo para identificarlo.

Ejemplo

```
Sub MostrarIDEvento(indice as Long)

     Msgbox Sist.Bitacora.Eventos(indice).IDEvento

End Sub
```

IDTipoEvento

Descripción

Indica el tipo de evento que ocurrió en el sistema. Los posibles tipos de evento son:

- Descubrimiento: se ejecutó el método Descubrir
- Alarma: se produjo una alarma en un dispositivo
- Falla: se produjo una falla en un dispositivo
- Otro: otro tipo de evento

Estos tipos se encuentran en forma de enumeración de acuerdo con la siguiente declaración:

```
Public Enum eTipoEvento
     sdkcievDescubrimiento = 1
     sdkcievAlarma = 2
     sdkcievFalla = 3
     sdkciOtro = 4
End Enum
```

Sintaxis

Visual Basic	`{varobjeto}.TipoEvento = {numero As eTipoEvento}`

Comentarios

Propiedad solo lectura, el tipo de evento es asignado automáticamente por el sistema al agregar el evento a la bitácora.

Ejemplos

```
Sub MostrarTipoEvento(IDEvento as Long)
    Msgbox Sist.Bitacora.Eventos(IDEvento).TipoEvento
End Sub
```

FechaHora

Descripción

Guarda la fecha y hora en que ocurrió el evento.

Sintaxis

Visual Basic	`{varobjeto}.FechaHora = {d As Date}`

Comentarios

Propiedad solo lectura, se asigna automáticamente cuando se agrega el evento a la colección.

Ejemplo

```
Sub MostrarFHEvento(IDEvento as Long)
     Msgbox Sist.Bitacora.Eventos(IDEvento).FechaHora
End Sub
```

IDSistema

Descripción

Identifica el sistema al que pertenece el dispositivo que generó el evento.

Sintaxis

Visual Basic	{varobjeto}.IDSistema = {numero As Long}

Comentarios

Propiedad solo lectura

Ejemplo

```
Sub MostrarDispEvento(IDEvento as Long)
     Msgbox Sist.Bitacora.Eventos(IDEvento).IDDispositivo
End Sub
```

IDDispositivo

Descripción

Identifica el dispositivo que generó el evento.

Sintaxis

Visual Basic	`{varobjeto}.IDDispositivo = {numero As Long}`

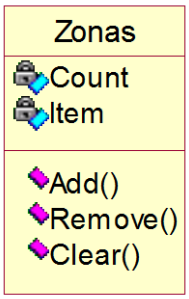
Comentarios

Propiedad solo lectura.

Ejemplo

```
Sub MostrarZonaEvento(IDEvento as Long)
     Msgbox Sist.Bitacora.Eventos(IDEvento).IDZona
End Sub
```

2.6.2.8. Colección Zonas



Propiedades

Count

Descripción

Indica la cantidad de zonas en la colección.

Sintaxis

Visual Basic	`{varobjeto}.Count`

donde:

varobjeto es una variable de tipo colección Zonas

Comentarios

Propiedad solo-lectura

Ejemplo

```
Sub MostrarCantidadZonas()
     Dim Cantidad as Long

     Cantidad = Sist.Zonas.Count
     If Cantidad > 1 then
          Msgbox "Cantidad de zonas: " & Cantidad
     Else
          Msgbox "El sistema no tiene zonas adic."
     End if
End Sub
```

Item

Descripción

Permite acceso individual a cada una de las zonas en la colección.

Sintaxis

Visual Basic	`{varobjeto}.Item(Indice as Long)`

Donde:

Varobjeto: es una variable tipo colección Zonas

Indice: es el índice de la zona a la que se desea hacer referencia

Comentarios

Propiedad por defecto para la colección Zona. El posible acceder al arreglo por ID de Zona (Long).

Ejemplo

```
Sub DeshabDispositivosZona(IDZona as Long)
     Dim Zn as Zona

     Set Zn = Sist.Zonas.Item(IDZona)

     Zn.Dispositivos.EstadoBase = sdkciDehabilitado

End Sub
```

Métodos

Add

Descripción

Permite agregar una zona a la colección Zonas.

Sintaxis

Visual Basic	`{varobjeto}.Add([Zn as Zona])`

donde:

varobjeto es una variable de tipo colección Zonas

Zn: Opcional. El objeto tipo Zona que se desea agregar a la colección

Comentarios

Si no se especifica como parámetro el objeto Zona, se agregará una nueva Zona numerada en forma consecutiva del resto de las zonas.

Ejemplo

```
Dim Z as Zona
Set Z = Sist.Zonas.Add()
Z.Descripcion = "Perimetro Externo"
Sist.Dispositivos("10.10.10.22").IDZona = Z.IDZona
```

Remove

Descripción

Permite eliminar un evento de la colección Zonas

Sintaxis

Visual Basic	`{varobjeto}.Remove(IDZona as Long, [EliminarDisp as Boolean = False])`

Donde:

Varobjeto: Es una variable de tipo colección Zonas

Indice: Es el índice en la colección o el IDZona de la zona que se desea eliminar

EliminarDisp: Indica si se desea eliminar los dispositivos que se encuentran asociados a la Zona.

Comentarios

No es posible eliminar la Zona 1 del sistema.
Si se omite o se indica *True* en el parámetro EliminarDisp, se eliminarán todos los dispositivos relacionados con la zona. Si se indica *False* los dispositivos asociados a la zona se colocarán en la Zona 1.

Ejemplo

```
Sub EliminarZona(Indice as Long)
     Sist.Zonas.Remove Indice
End Sub
```

Clear

Descripción
Limpia la colección Zonas, eliminando todos los objetos que contiene, excepto la Zona 1.

Sintaxis

Visual Basic	`{varobjeto}.Clear([EliminarDisp as Boolean])`

Donde:

Varobjeto: Es una variable de tipo colección Eventos

EliminarDisp: Opcional. Indica si se desea eliminar los dispositivos que se encuentran asociados a las zonas eliminadas.

Comentarios

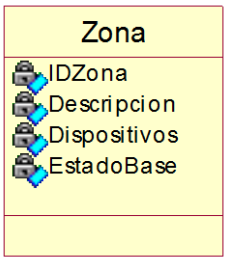
Este método elimina de la colección todas las zonas, dejando únicamente la zona 1.

Si se omite o se indica *True* en el parámetro EliminarDisp, se eliminarán todos los dispositivos relacionados con las zonas eliminadas. Si se indica *False* los dispositivos asociados a esas zonas se colocarán en la Zona 1.

Ejemplos

```
Sub LimpiarZonas()

     Sist.Zonas.Clear False

End Sub
```

2.6.2.9. Clase Zona



Propiedades

IDZona

Descripción

Identifica a la zona. Es asignado automáticamente al agregar la zona.

Sintaxis

Visual Basic	`{varobjeto}.IDZona = {numero As Long}`

donde:

varobjeto es una variable de tipo Zona

Comentarios

Propiedad de solo lectura, contiene un consecutivo asignado por el sistema a cada zona.

Ejemplo

```
Sub MostrarIDZonas()

     For each z in Sist.Zonas
          Msgbox z.IDZona
     Next

End Sub
```

Descripcion

Descripción

Contiene una descripción de la zona asignada que puede ser asignada por la aplicación administrativa que utilice el SDK.

Sintaxis

Visual Basic	`{varobjeto}.Descripcion = {desc As String}`

Comentarios

Inicialmente la zona no tiene descripción. Se recomienda asignar una descripción para facilidad del usuario.

Ejemplo

```
Sist.Zonas(1).Descripcion = "Sala-Comedor"
```

Dispositivos

Descripción

Es una colección que contiene todos los objetos de tipo dispositivo asignados a la zona. A través de ella es posible acceder a los dispositivos de la zona y trabajar con ellos. Es una colección con los métodos estándar de utilizados en modelos de objetos COM por lo que soporta las operaciones de colección. El índice base es 1. Además, es posible acceder directamente a un dispositivo utilizando como llave su dirección IP.

Sintaxis

Visual Basic	`Acceso a un dispositivo por índice`
	`{varobjeto}.Dispositivos(indice as Long)`
	
	`Acceso a un dispositivo por dirección IP`
	`{varobjeto}.Dispositivos(Dirección as String)`

donde:

varobjeto es una variable de tipo Zona

Comentarios

Esta colección contiene todos los dispositivos descubiertos o agregados manualmente a una zona, a diferencia de la colección Dispositivos de la clase Sistema, la cuál todos los dispositivos.

Ejemplo

sintaxis

Este ejemplo coloca la dirección IP, el OID y la descripción de todos los dispositivos de la zona 1 en un ListView llamado lvDispositivos.

```
Sub ActualizarListaAgentes()
    Dim Disp As Dispositivo
    Dim liFila As ListItem

    For Each Disp In Sist.Zonas(1).Dispositivos
        Set l = lvDispositivos.ListItems.Add
        liFila.Text = Disp.DireccionIP
        liFila.SubItems(1) = Disp.oid
        liFila.SubItems(2) = Disp.Descripcion

        'si el dispositivo esta habilitado entonces
        'poner la fila en negrita
        If Disp.EstadoBase <> sdkciHabilitado Then
            liFila.Bold = True
        End If
    Next
End Sub
```

EstadoBase

Descripción

Se ha definido un conjunto de estados básicos que cualquier dispositivo del sistema debe entender y soportar en su MIB.

Estos estados son los siguientes:

- Deshabilitado
- Habilitado
- Armado
- Alarma
- Falla

Sintaxis

Visual Basic	`{varobjeto}.EstadoBase = {numero As eEstadoBase}`

donde:

varobjeto es una variable de tipo Zona

eEstadoBase es una enumeración que contiene los posibles estados, de acuerdo a la siguiente declaración:

```
Public Enum eEstadoBase
    sdkciDeshabilitado = 0
    sdkciHabilitado = 1
    sdkciArmado = 2
    sdkciAlarma = 3
    sdkciFalla = 4
End Enum
```

Comentarios

Al cambiar el estado de una zona automáticamente se modificará el estado de todos los dispositivos que pertenecen a ella.
Las constantes utilizadas para esta propiedad pueden utilizarse también para cambiar el estado de un dispositivo individualmente o del sistema en su totalidad.

Ejemplo

```
Private Sub DeshabilitarZona(Num as Long)
     Dim Zn as Zona

     Set Zn = Sist.Zonas.Item(Num)

     Zn.EstadoBase = sdkciDeshabilitado

End Sub
```

3. Aplicación práctica del SDK

En este capítulo se describen detalladamente las pruebas realizadas al modelo de software desarrollado y algunos casos que se proponen para comprobar cómo el SDK permite crear una solución completa y a la vez sencilla de implementar.

Se detallan las pruebas realizadas, los casos aplicados y los resultados de estas pruebas. En los anexos 2 y 3 se da información más detallada de las aplicaciones y los agentes que se desarrollaron para implementar las pruebas del SDK.

3.1. Pruebas del SDK

Se comprobó el funcionamiento de los componentes COM desarrollados mediante el desarrollo de interfaces en Visual Basic que interactuaran con el SDK y con dispositivos de seguridad, algunos de ellos emulados mediante software, y otros conectados físicamente a estaciones en la red.

3.1.1. Ambiente de pruebas

Todas las pruebas fueron realizadas utilizando la Red Institucional del Instituto Tecnológico de Costa Rica, en el laboratorio y oficinas del Centro de Cómputo.

El desarrollo de las aplicaciones de prueba y los agentes se realizó en Visual Basic utilizando los componentes COM del SDK.

Dado que la cantidad de dispositivos de seguridad con que se contó fue limitada, se implementaron emuladores que funcionan de forma similar que los dispositivos, pudieran generar alarmas y dar notificaciones de eventos.

Estos emuladores permitieron comprobar el funcionamiento del SDK con mayor variedad y cantidad de dispositivos de los que se tenía en realidad.

A continuación se especifica el hardware y software utilizado para la realización de las pruebas del SDK.

3.1.1.1. Hardware

Equipo de desarrollo y ejecución de herramientas administrativas SNMP:

- Computador Pentium III 850 MHz
- 128 Mb de memoria RAM
- 40 Gb Disco Duro
- Tarjeta de Red 10/100 Mbps

Equipos para ejecución de agentes:

- Computadores Pentium II 200 MHz
- 32 Mb de memoria RAM
- 6 Gb Disco Duro
- Tarjeta de Red 10/100 Mbps
- Puerto serial estándar para comunicación con dispositivos físicos

Equipos de red y cableado:

- Cableado Cat 5
- Red a 100 Mbs
- Utilizando switches para conectividad entre estaciones

3.1.1.2. Software

El SDK y las aplicaciones pueden ejecutarse sobre cualquier equipo que cumpla el siguiente requisito único de software:

- Windows 9x, NT 4.0 ó 2000

Como parte de este trabajo no se desarrolló ni se realizaron pruebas utilizando otras plataformas, sin embargo, esta posibilidad se recomienda y se deja abierta para futuros proyectos.

3.2. Dispositivos – Agentes

Un elemento importante en las pruebas son los dispositivos utilizados y los agentes que permiten la comunicación entre una computadora y el dispositivo.

Entre los dispositivos que se utilizaron para las pruebas se encuentran:

- sensores magnéticos,
- sensores de movimiento,
- sirenas,
- cerraduras eléctricas
- luces.

Algunos de estos dispositivos se implementaron mediante emuladores, otros fueron conectados físicamente a una computadora mediante el puerto serial.

La computadora contiene el agente que provee la “inteligencia” del dispositivo, y que implementa las operaciones de un agente SNMP y puede transmitir señales de control al dispositivo a través del puerto serial.

3.3. Dispositivos físicos

A continuación se da una descripción de los tres dispositivos físicos que se utilizaron en las pruebas, estos son:

- Sensores magnéticos para puertas y ventanas
- Sensores de movimiento
- Sirenas para señal audible en caso de alarma

Se da una especificación del dispositivo, y se explica la forma de conexión realizada por medio del puerto serial de una computadora y fuentes de poder externas.

El desarrollo de las aplicaciones de control y agentes se realizó en Visual Basic utilizando el Comm ActiveX Control para la interacción con el puerto serial de la computadora.

La conexión de los dispositivos, los conectores y los circuitos necesarios se realizó con apoyo del Ing. Roy Alpizar, Ingeniero en Electrónica del Instituto Tecnológico de Costa Rica.

3.3.1. Sensor magnético

El sensor magnético utilizado es un 49-497 de Radio Shack, el cual tiene el esquema de “normalmente-cerrado”, por lo que genera un voltaje cuando se separan los dos contactos del sensor.

Funciona con una tasa de voltaje de 130 VDC, y una corriente de 50mA máxima. Debido a que las características del sensor son compatibles con la especificación del puerto serial RS-232 de una PC, para la conexión de este dispositivo es posible conectar sus cables directamente a los pines del puerto serial.

Para este caso se escogió utilizar el pin 8 del conector DB9, que corresponde a la señal de Clear To Send, que puede ser leída desde cualquier lenguaje de programación.

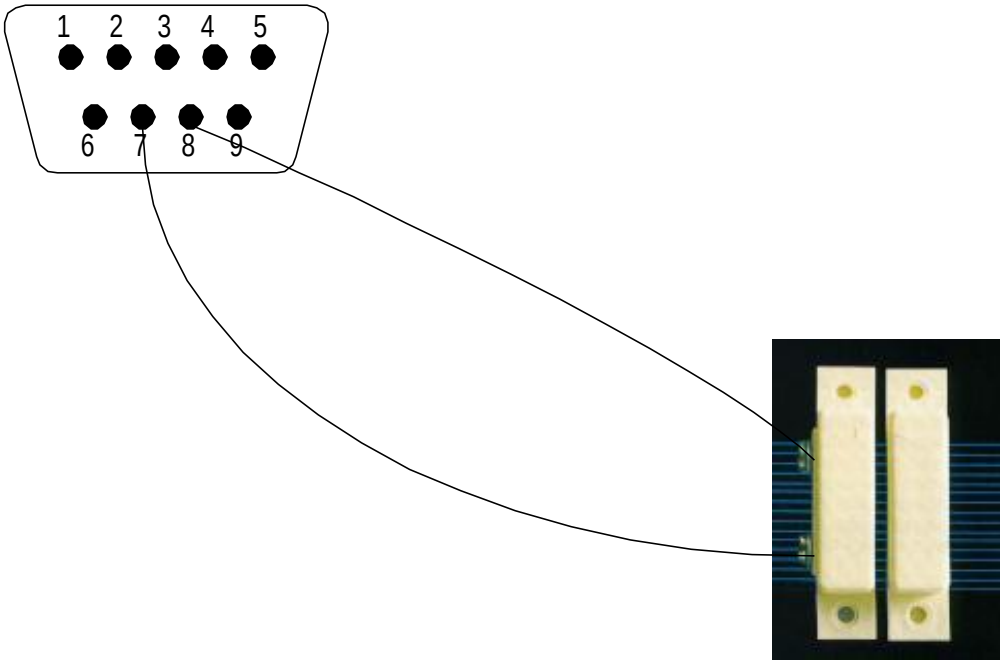


Figura 19. Conexión del sensor magnético a un conector DB9

3.3.2. Sensor de Movimiento

El sensor de movimiento que se ha utilizado es un Sensor de Movimiento Infrarrojo Pasivo Radio Shack 49-208A. Utilizado para montaje en paredes o techo.

El sensor incluye 4 diferentes tipos de lentes intercambiables, dependiendo de las condiciones en las que se coloque el sensor. Estas lentes son:

- Ángulo ancho (cubre 180º)
- Larga Distancia
- Mascotas
- Cortinas

Requiere al menos cuatro conexiones:

- Dos para el envío de señales de alarma
- Dos para alimentación externa de 12 VDC.

Para la alimentación externa se ha utilizado un adaptador AC-DC Nippon America Modelo 5467, con entrada de 110 VAC y salida ajustable de 3, 4.5, 6, 7.5, 9 y 12 VDC, y con una corriente de 500mA máximo.

Debido a que las características del sensor son compatibles con la especificación del puerto serial RS-232 de una PC, para la conexión de este dispositivo es posible conectar sus cables de señal de alarma directamente a los pines del puerto serial.

Se utilizó el pin 8 del conector DB9, que corresponde a la señal de Clear To Send, como se muestra en la siguiente figura:

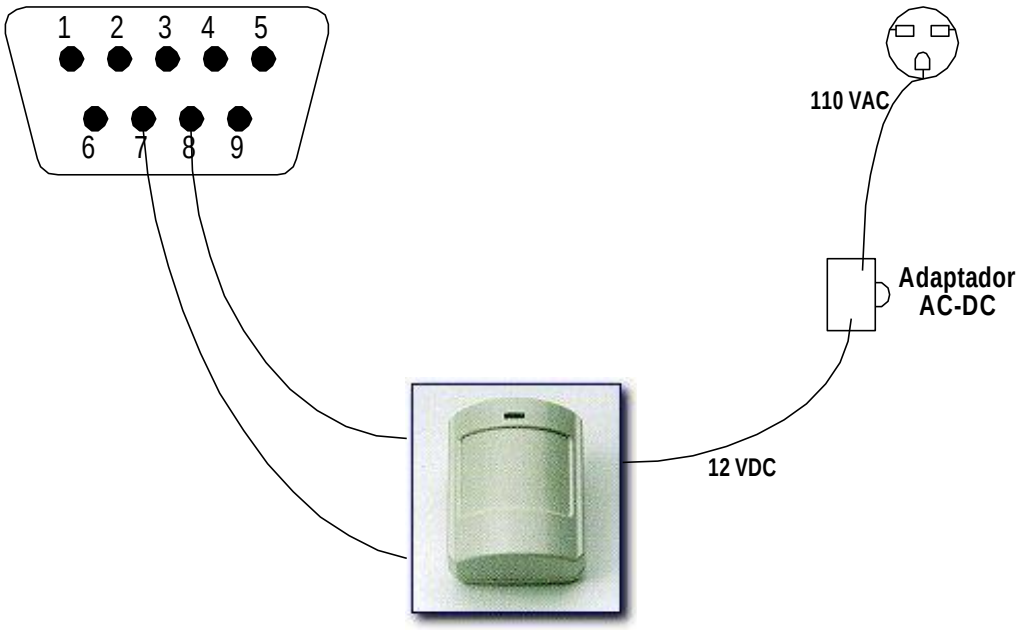


Figura 20. Conexión del sensor de movimiento a un conector DB9 y a una fuente de alimentación externa

3.3.3. Sirena

Se utilizó una Sirena Electrónica Dual Radio Shack 49-480 que puede emitir dos tipos distintos de sonido dependiendo de a cuál de sus cables positivos se le coloque un voltaje.

Puede emitir un sonido de alarma de robo de 105±5 dB, o un sonido de alarma contra fuego de 107±5dB. Para su funcionamiento requiere un voltaje de 12 VDC y una corriente de 600 mA.

Debido a que el puerto serial no puede darle a la sirena el voltaje ni la corriente que requiere para funcionar, se utilizó una fuente de corriente externa por medio de un adaptador AC-DC Nippon America Modelo 5467, con entrada de 110 VAC y

salida ajustable de 3, 4.5, 6, 7.5, 9 y 12 VDC, y con una corriente de 500mA máximo.

La conexión de la sirena al puerto serial se realizó utilizando los pines 4 (Data Terminal Ready), 5 (Signal Ground), y 7 (Request To Send) que pueden ser controlados individualmente desde un lenguaje de programación para producir los distintos tipos de sonido que provee la sirena.

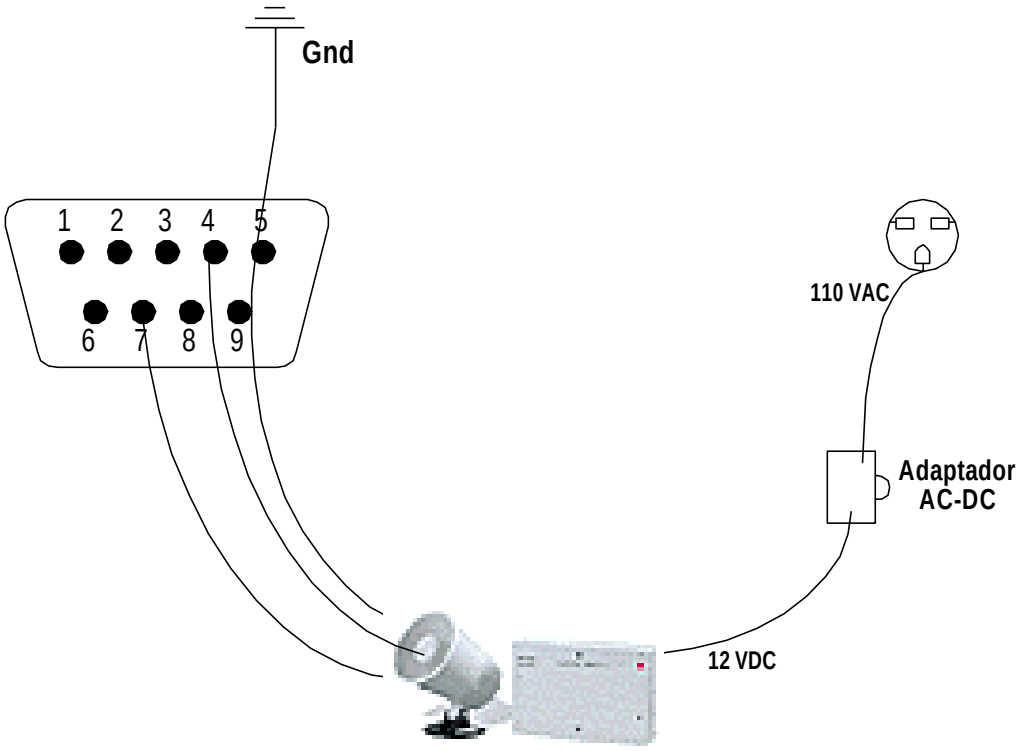


Figura 21. Conexión la sirena a un conector DB9 y a una fuente de alimentación externa

Para la conexión de este dispositivo además fue necesario el diseño de un circuito que permitiera pasar la corriente de 12 VDC de la fuente de poder externa dependiendo de cuál de los pines del conector DB9 se encendiera.

Con el apoyo del Ing. Alpizar se diseño el circuito para resolver el problema, utilizando dos resistencias de 1 K Ohm, un opto acoplador NTE3086 y dos transistores. Su diagrama se muestra a continuación:

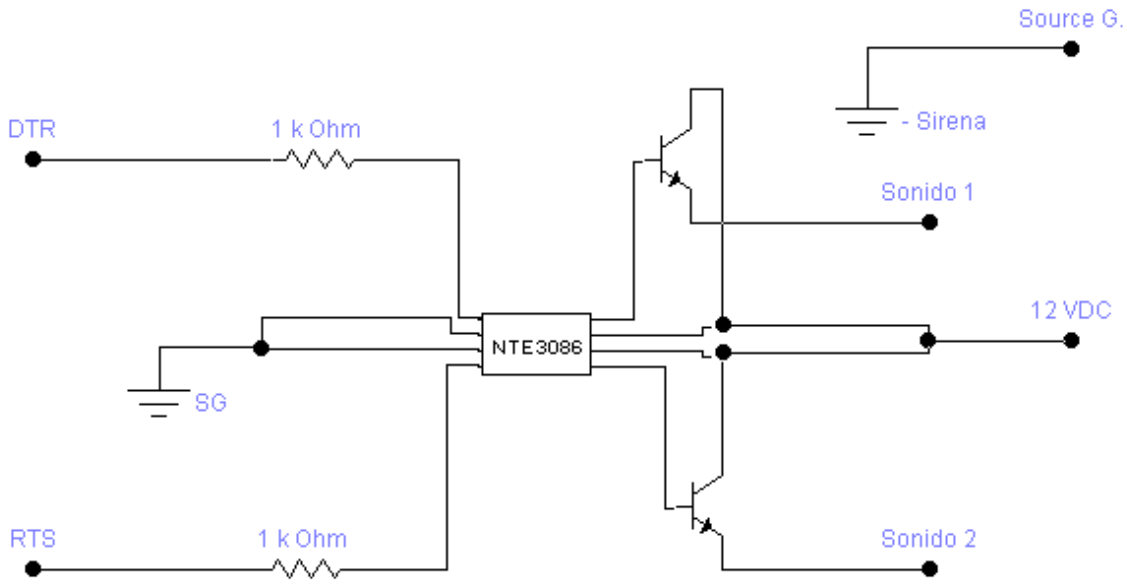


Figura 22. Circuito utilizado para conectar la sirena al conector DB9 y a la alimentación externa

Mediante este circuito es posible, utilizando los pines RTS y DTR del conector DB9, producir los dos tipos de sonido en la sirena, utilizando una fuente externa de alimentación.

3.4. Emuladores

Dado que la cantidad de dispositivos físicos con que se contó fue limitada, y para poder simular los casos que se plantearon, fue necesario desarrollar programas que simularan el comportamiento de los dispositivos, y que pudieran servir de interfaz para generación de alarmas.

Los emuladores se desarrollaron en Visual Basic y proveen una sencilla interfaz que permite simular el comportamiento de los dispositivos. En el Anexo 2 se incluye una descripción del funcionamiento de cada uno de estos agentes.

3.5. Agentes

Los agentes se desarrollaron utilizando los controles ActiveX de Dart PowerTCP SNMP Tool, que proveen los métodos de bajo nivel para la recepción y envío de mensajes SNMP.

Se utilizó Visual Basic pues permite implementar los agentes de una forma sencilla y rápida, como se espera que sean estos programas.

Cada agente se encarga de traducir las señales del puerto serial a mensajes SNMP y viceversa, dependiendo del tipo de dispositivo que se encuentre controlando.

Un aspecto muy importante del agente es que debe conocer el MIB del dispositivo que administra y poder interpretar cada uno de los posibles valores de las variables y traducirlos a señales de control para el dispositivo.

Como parte de este proyecto se desarrollaron MIBs y agentes para los siguientes tipos de dispositivos:

- Sensor magnético (SMAG01.MIB)
- Sensor de movimiento (SMOV01.MIB)
- Sirena Dual (SIRD01.MIB)
- Luz (LUZ01.MIB)
- Cerrojo Eléctrico (CELEC01.MIB)

En el Anexo 2 se incluye el texto completo de los MIBS para el sensor magnético, de movimiento y la sirena dual utilizada.

3.6. Aplicaciones administrativas

Para realizar las pruebas del SDK se implementó una interfaz administrativa para el sistema de seguridad en Windows que permite controlar los dispositivos directamente en un equipo administrativo.

La funcionalidad de las interfaz administrativa es el descubrimiento de dispositivos en el sistema, vista y edición de las variables del MIB de cada dispositivo, y la recepción de eventos de alarma y falla de los dispositivos.

3.7. Caso de prueba

Una vez implementados los componentes del modelo de hardware y software se diseñó un caso para comprobar su funcionamiento. Se ha pretendido que este caso refleje el tipo de aplicaciones para las que el SDK fue diseñado y sea una situación real para un sistema de seguridad.

Este caso se describe a continuación, se indica su objetivo, los requerimientos por cumplir, la configuración física, los dispositivos involucrados y la solución de software que se implementó utilizando el SDKCI_SNMP.

3.7.1. Caso. Administración de un sistema de alarma de una zona

Objetivo

Utilizar el SDKCI_SNMP para administrar un sistema de alarma de una zona utilizando sensores de movimiento y magnéticos, y una sirena para dar aviso de alarmas.

Requerimientos

El caso se presenta en una oficina que consta de dos habitaciones, una habitación tiene una puerta de ingreso y una ventana, y la otra tiene dos ventanas. Las dos se encuentran comunicadas mediante una puerta adicional. De acuerdo con el siguiente diagrama:

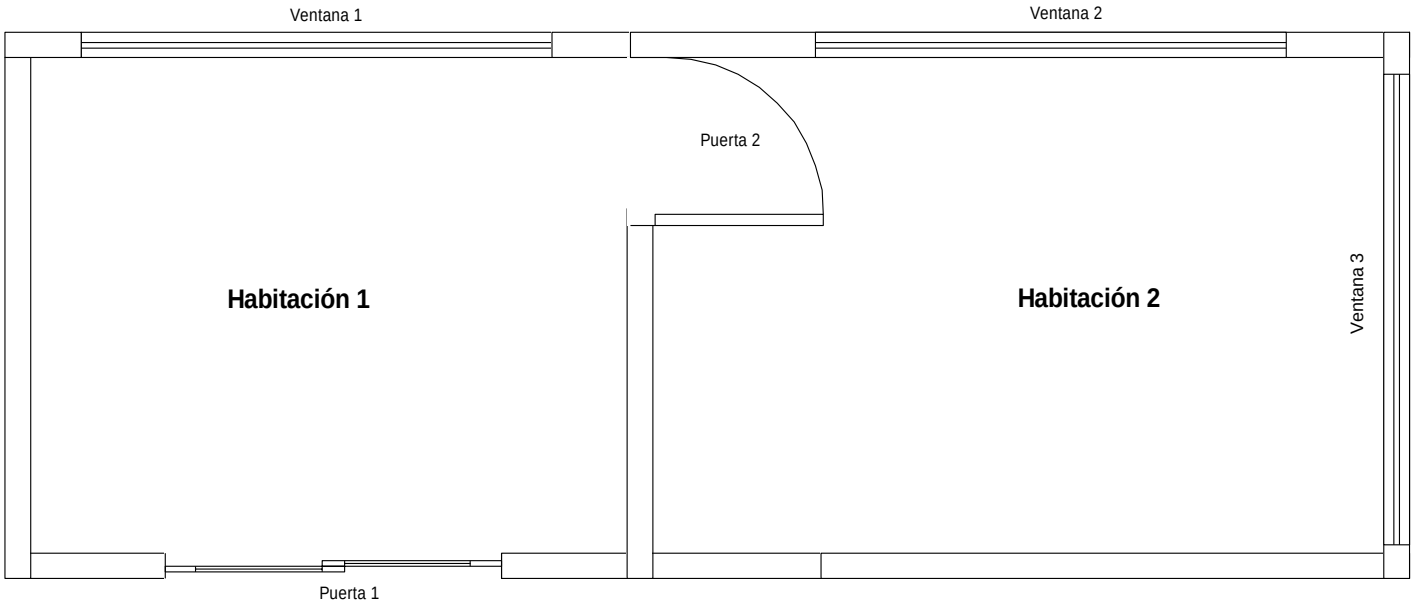


Figura 23. Plano de las habitaciones utilizadas en el caso de ejemplo

Los requerimientos del sistema de alarma son:

- Colocar un sensor de movimiento en cada habitación
- Colocar sensores magnéticos en las puertas y ventanas para detectar si se abren
- Colocar una sirena que dé aviso en caso de que se produzca una alarma.

Dispositivos Involucrados

Para cumplir con los requerimientos del caso se requieren los siguientes dispositivos:

- 2 sensores de movimiento
- 4 sensores magnéticos para la puerta y ventanas externas
- 1 sirena

Configuración física

Los dispositivos serán colocados de acuerdo al siguiente diagrama:

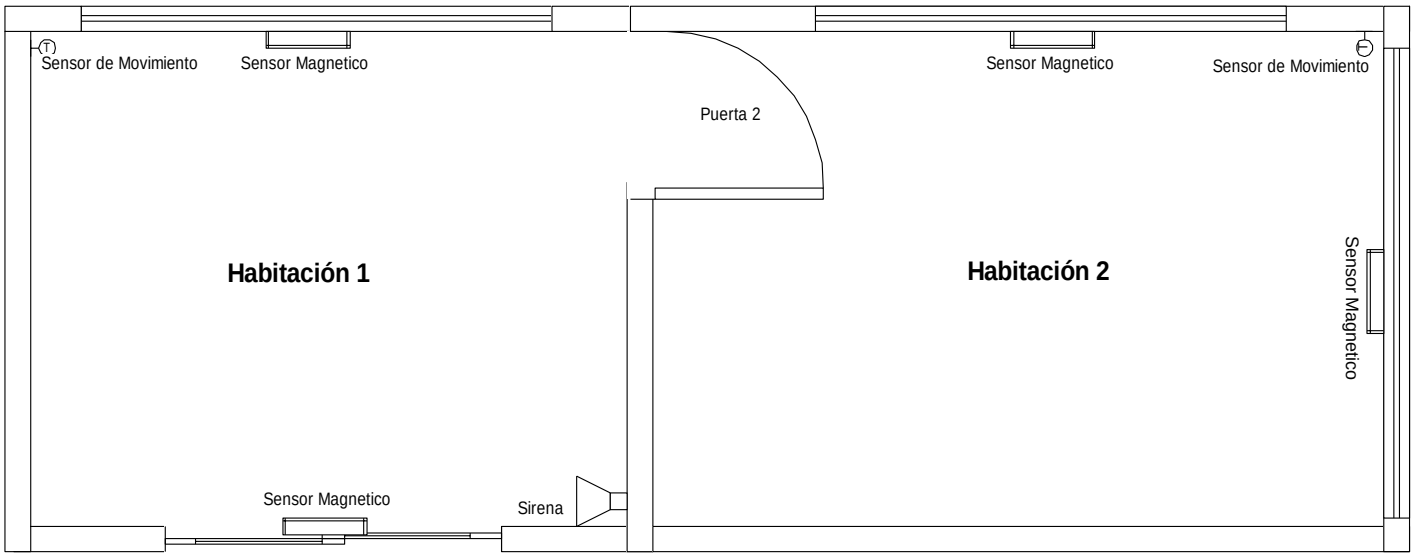


Figura 24. Colocación de los dispositivos de seguridad para el caso de prueba

Administración de agentes

Utilizando la aplicación administrativa desarrollada, es posible controlar este escenario y detectar cualquier alarma que reporten los sensores.

La siguiente figura muestra la interfaz de la aplicación administrativa que permite controlar el escenario mostrado, una vez que se ha ejecutado el proceso de descubrimiento de dispositivos:

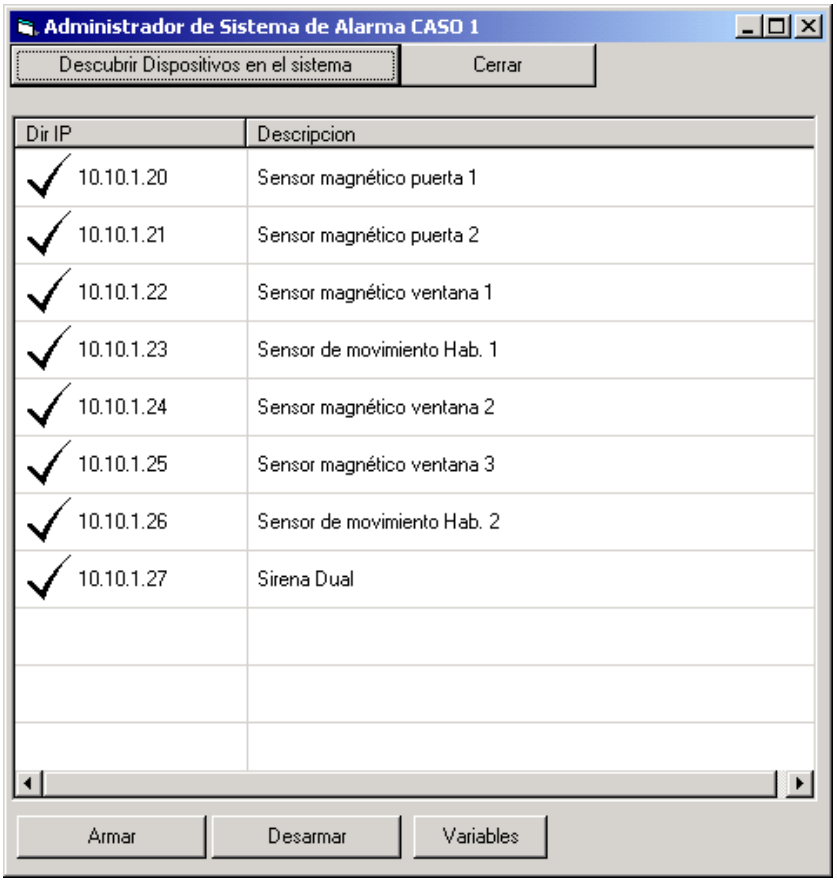


Figura 25. Interfaz administrativa que muestra los dispositivos existentes en el sistema

Tratamiento de Alarmas

La aplicación administrativa a través del evento Alarma puede detectar una alarma ocurrida en cualquier dispositivo e indicar a la sirena que produzca su sonido, además de informar al usuario acerca de la ocurrencia del evento.

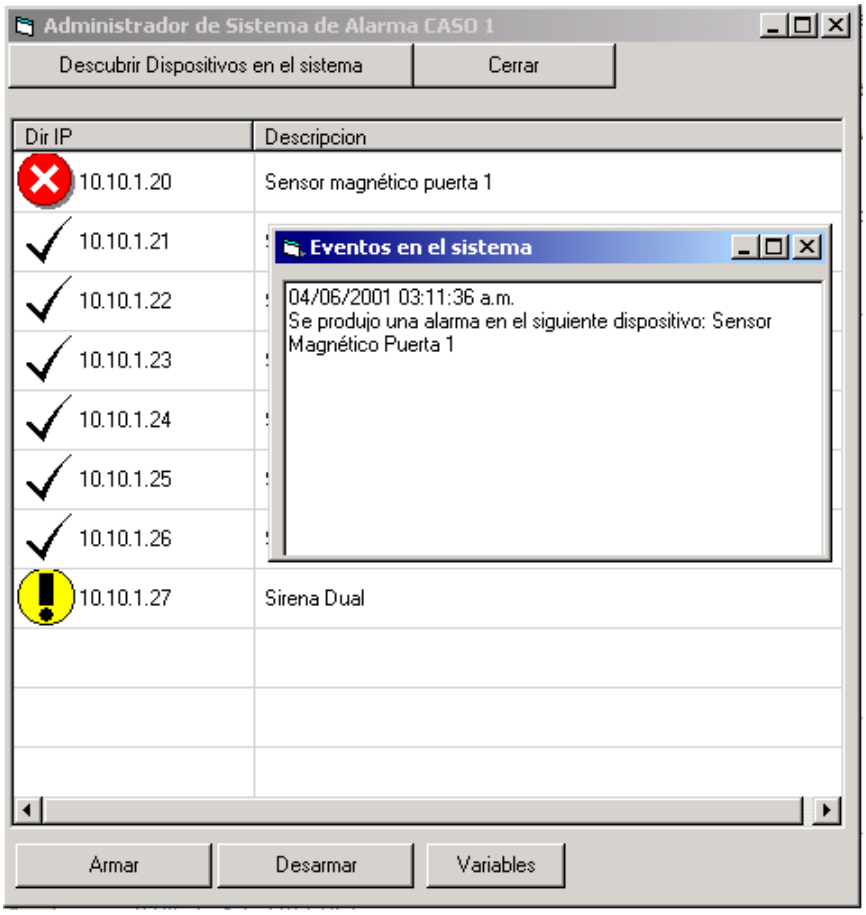


Figura 26. Interfaz que muestra las alarmas y el estado de los dispositivos

Se observa el funcionamiento correcto del SDK en el paso de eventos y la configuración de los estados de los dispositivos.

4. Conclusiones y recomendaciones

4.1. Conclusiones

Las principales conclusiones obtenidas en de este trabajo:

- El protocolo SNMP es lo suficientemente flexible para poder aplicarlo no solamente a la administración de dispositivos de red, sino a cualquier otro tipo de dispositivo, para lo cuál solamente es necesario desarrollar el MIB correspondiente y dotar al dispositivo del hardware necesario para conectarse a una red LAN y de la lógica necesaria para comunicarse vía TCP/IP e interpretar mensajes SNMP.

- En la actualidad existe en el mercado un sinnúmero de dispositivos de seguridad, sin embargo, no existen en el mercado dispositivos de seguridad que soporten un protocolo estándar y abierto como lo es SNMP, y si se llegarán a fabricar serían costosos hasta que su consumo sea masivo, haya mayor competencia y más proveedores lo soporten.

- La creación de un SDK no solamente facilita el desarrollo aplicaciones para utilizar la tecnología implementada, sino que deja la puerta abierta a nuevas investigaciones, más desarrollos y mejores aplicaciones que aprovechen la tecnología, o la apliquen a otras áreas. En el caso del SDKCI_SNMP, su uso no solamente se aplica a dispositivos de seguridad, sino que podría utilizarse para administrar por medio de SNMP cualquier tipo de dispositivo inteligente, incluyendo equipos para el hogar, edificios y aplicaciones industriales.

- Los dos elementos principales en el diseño de un agente SNMP para un dispositivo son el diseño de su MIB y el agente que lo soportará y que controlará el hardware. Esto es suficiente para que cualquier aplicación administrativa pueda conocer todas las posibilidades de administración del dispositivo y el usuario pueda manejarlo.

- El uso de protocolos estándar de red permite la posibilidad de utilizar el cableado y los equipos de red existentes en una casa o edificio inteligente no solamente para transporte de datos o acceso a Internet, sino para poder controlar el sistema de seguridad y todos los demás dispositivos administrables mediante SNMP.

- Si una empresa ha realizado una alta inversión en software de administración de redes, podría maximizar el aprovechamiento de ésta utilizando la herramienta no solamente para administrar sus equipos activos de red, sino también para administrar su sistema de seguridad, control de ambiente, y cualquier otro sistema administrable mediante SNMP.

- La conexión de dispositivos externos a equipos de cómputo es sencilla utilizando conexiones seriales, paralelas o en algunos casos, interfaces propietarias. Si los controladores para estos equipos fueran abiertos para los desarrolladores, se podrían crear agentes que se ejecuten en las computadoras a las que los dispositivos se conectan y permitan administrarlos a través de SNMP.

- SDKCI_SNMP implementa el protocolo SNMP v1, que no provee seguridad y algunas de sus características son ineficientes y han complicado el desarrollo del SDK. Debido a esto es necesario implementar en la red otras medidas que garanticen la confiabilidad y seguridad de la información de administración y de las alarmas generadas por los dispositivos.

4.2. Recomendaciones

- Una mejora recomendable para el SDKCI_SNMP es utilizar las versiones mejoradas de SNMP para incrementar la seguridad, funcionalidad y eficiencia en la transmisión de los datos de administración de dispositivos.

- Al desarrollar aplicaciones administrativas se recomienda brindar al usuario tantas facilidades como sea posible para el control de los dispositivos, y además brindar todas las interfaces de reporte de alarmas que los sistemas de alarma convencionales tienen, los cuales incluyen envío de mensajes a localizadores, llamadas telefónicas a ciertos números programables o alarmas silenciosas que den aviso a centros de monitoreo o a la policía.

- Se recomienda al usuario de un sistema de seguridad como los propuestos en este trabajo conocer la organización de zonas, la localización de los sensores y sus identificaciones, con el fin de detectar fácilmente el lugar por donde un intruso está violando la seguridad, de acuerdo a los avisos de los sistemas de administración de dispositivos.

- Se recomienda desarrollar dispositivos con variables bien definidas que permitan crear MIBs autodocumentados para facilitar al usuario su programación.

- En futuros proyectos que involucren el desarrollo de software se recomienda desarrollar software abierto y extensible utilizando componentes, que faciliten el desarrollo de nuevas ideas utilizando como base el software existente.

- Para la implementación de una casa inteligente se recomienda el desarrollo de más interfaces (web ó telefónica, por ejemplo) que brinden al usuario más facilidades de acceso al control y administración en general de los dispositivos.

5. Referencias bibliográficas

[ASTRO, 97] Astroautomation, Inc. www.astroautomation.com/astro.html

[COMER, 97] Comer, D.E. Redes de computadoras, Internet e Interedes. Primera edición. Prentice Hall. 1997.

[DDRI, 99] Diversified Data Resources Inc. AN OVERVIEW OF SNMP.1999. http://www.ddri.com

[DOMO2000] Tecnología Electrónica con X-10. www.domotica.net

[FALLAS, 92] Fallas, N.H.Seguridad Empresarial. 29 de septiembre de 1992. Colegio Universitario de Cartago.

[HP, 97] Hewlett Packard Company. SNMP++, C++ Based Application Programmers Interface for the Simple Network Management Protocol.

[INFORAMP, 98] www.inforamp.net/kjvallil/t/work.html#sumary1998.

[MARSTON, 99] Marston, R.M.Manual de circuitos electrónicos para seguridad. Grupo editorial Ceac, S.A. 1999. España.

[MATTHEWS, 96] MATTHEWS., D. 1996. A bluffer's guide to SNMP, the Simple Network Management Protocol. Drm@kagi.com.

[MORA, 96] Mora, E. Guía para diseñar un buen sistema de alarma contra robo. Colegio Universitario de Cartago, 1996.

[MUELLER, 1998] Mueller, S. Upgrading and Repairing PCs. Tenth Aniversary Edition. QUE. United States, 1998.

[NETCOM, 2000] EarthLink. www.netcom.com

[RADC, 95] SNMP - Simple Network Managment Protocol.1995

[SANABRIA, 2000]. Sanabria, C., Jiménez, J. Seminario de Investigación II, Desarrollo de Aplicaciones con SNMP. Instituto Tecnológico de Costa Rica. 2000

[TECHNET, 2000] Windows 2000 implementation of SNMP. 2000 http://windows.microsoft.com/windows2000/reskit/webresources

[TOWNSEND, 1995] Tonwsend, R. SNMP Application Developer's Guide. Van Nostrand Reinhold Communications Library. USA. 1995.

[VISS, 2000] Vision Systems Automation. www.vissys.com

6. Anexos

ANEXO 1. Asignación de un número de empresa (Enterprise Number) para el ITCR

Christian Sanabria Jiménez

De: IANA Private Enterprise Number [iana-pen@iana.org]
Enviado: Lunes, 23 de Abril de 2001 04:40 p.m.
Para: Christian Sanabria Jiménez
Asunto: RE: Application for Enterprise-number (9283)

Dear Christian,

The IANA has assigned the following Private Enterprise
Number to Costa Rica Institute of Technology, with you as the point of
contact:

9283 Costa Rica Institute of Technology Christian Sanabria

csanabria@itcr.ac.cr

Please notify the IANA if there is a change in your contact
or company information.

Thank you,

Bill Huang
IANA - Private Enterprise Numbers

****
Internet Assigned Numbers Authority
4676 Admiralty Way, Suite 330
Marina del Rey, CA 90292
USA

iana-pen@iana.org

+1-310-823-9358
+1-310-823-8649 (fax)

****

ANEXO 2. Descripción de los emuladores de dispositivo implementados para las pruebas del SDK

Emulador del sensor de movimiento

Este programa funciona de manera similar al sensor de movimiento por hardware utilizado para las pruebas, se utiliza incluso el mismo código de agente para dar inteligencia a ambos.

Este emulador genera una alarma al mover el mouse sobre la zona central del sensor.

Figura 27. Interfaz del emulador de un sensor de movimiento

Emulador del sensor magnético

Emula el funcionamiento de un sensor magnético. Cuando las dos terminales del sensor son separadas cierta distancia, se genera la alarma.

Figura 28. Interfaz del emulador de un sensor magnético

Emulador de la sirena

Este emulador produce un sonido similar al de un dispositivo de señal audible de una alarma antirrobo.

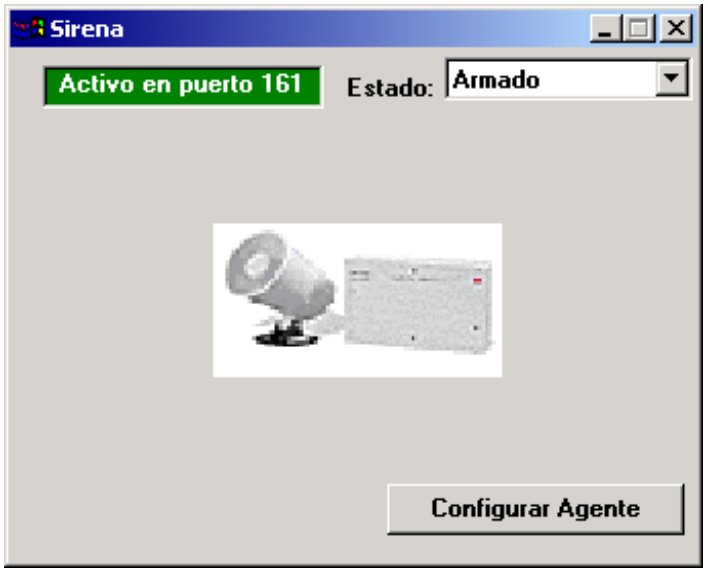


Figura 29. Interfaz del emulador de una sirena

Emulador de una luz

Simula el encendido-apagado de una luz, no genera alarmas sino solamente recibe comandos Get y Set SNMP que leen o cambian el estado del dispositivo (encienden o apagan la luz).

Figura 30. Interfaz del emulador de una luz

Emulador de un cerrojo eléctrico

Este emulador muestra el funcionamiento de un cerrojo eléctrico que abre una puerta remotamente, como los utilizados en portones eléctricos estándar.

Figura 31. Interfaz del emulador de un cerrojo eléctrico

ANEXO 3. MIBs para algunos agentes desarrollados

A continuación se muestra el código ASN.1 para los principales dispositivos implementados durante las pruebas del SDK.

Sensor de movimiento (SMOV01.MIB)

```
SMOV01-MIB DEFINITIONS ::= BEGIN


IMPORTS
     seguridad
     FROM SDKCI_SNMP-MIB;



Smov01         OBJECT IDENTIFIER ::= { seguridad 1 }


-- MIB para el sensor de movimiento SMOV01

  totalNumAlarmas OBJECT-TYPE
      SYNTAX  INTEGER
      ACCESS  read-only
      STATUS  mandatory
      DESCRIPTION
            "Cantidad de veces que el sensor ha
            activado la alarma"
      ::= { smov01 1 }


  sensibilidad OBJECT-TYPE
      SYNTAX  INTEGER
      ACCESS  read-write
      STATUS  mandatory
      DESCRIPTION
            "Nivel de sensibilidad para generacion de
            alarmas"
      ::= { smov01 2 }
END
```

Sensor magnético (SMAG01.MIB)

```
          SMAG01-MIB DEFINITIONS ::= BEGIN


          IMPORTS
               seguridad
               FROM SDKCI_SNMP-MIB;



         smag01    OBJECT IDENTIFIER ::= { seguridad 2 }


         -- MIB para el sensor magnetico 1 (normally closed)


        estado OBJECT-TYPE
           SYNTAX  INTEGER
           ACCESS  read-only
           STATUS  mandatory
           DESCRIPTION
                "Estado actual del sensor: 0 cerrado, 1
                abierto"
           ::= { smag01 1 }

           totalNumAlarmas OBJECT-TYPE
               SYNTAX  INTEGER
               ACCESS  read-only
               STATUS  mandatory
               DESCRIPTION
                "Cantidad de veces que el sensor ha activado
                la alarma"
               ::= { smag01 2 }

END
```

Sirena Dual (SIRD01.MIB)

```
   SIRD01-MIB DEFINITIONS ::= BEGIN

   IMPORTS
        seguridad
        FROM SDKCI_SNMP-MIB;

 Sird01         OBJECT IDENTIFIER ::= { seguridad 2 }

 -- MIB para la sirena dual SIRD01

   controlSirena OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
             "Controla el estado de la sirena, si se
             coloca en 1 activa el sonido,
             se puede combinar con testSirena para
             realizar pruebas, si se encuentra
             en 0 no se producira sonido aunque se
             coloque en 1 la variable sirenaTest"
       ::= { sird01 1 }


tiempoSirenaActiva OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-write
       STATUS  mandatory
       DESCRIPTION
             "Cantidad de segundos que sonara la
             sirena cuando se active"
       ::= { sird01 2 }

   sirenaTest OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
             "Cuando se coloca en 1 esta variable,
             la sirena se activa durante
             el tiempo configurado en
             tiempoSirenaActiva"
       ::= { sird01 3 }
```

```
        activarSirenaIncond OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                  "Si se coloca en 1 (on) el sonido de la
                  sirena se activa hasta que se
                  vuelva a colocar en 0, sin importar los
                  valores de las variables
                  controlSirena y tiempoSirenaActiva"
            ::= { sird01 4 }


        tipoSonido OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                  "Indica el tipo de sonido que la sirena
                  emitira al ser activada"
            ::= { sird01 5 }

END
```