ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

IMPLEMENTACIÓN DE UN AGENTE EXTENSIBLE SNMP SOBRE PLATAFORMA RASPBERRY PI PARA CONTROL Y MONITORIZACIÓN DE SENSORES

(Implementation of an extensible snmp agent on raspberry pi platform for control and monitoring of sensors)

Para acceder al Título de

Graduado en

Ingeniería de Tecnologías de Telecomunicación

Autor: Javier García Núñez

Junio - 2017

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Javier García Núñez
Director del TFG: José Ángel Irastorza Teja

Título: "Implementación de un agente extensible SNMP sobre plataforma raspberry pi para control y monitorización de sensores "
Title: "Implementation of an extensible SNMP agent on raspberry pi platform for control and monitoring of sensors "

Presentado a examen el día:

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Roberto Sanz Gil
Secretario (Apellidos, Nombre): Jose Angel Irastorza
Vocal (Apellidos, Nombre): Juan Luis Cano Diego

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

RESUMEN

En este proyecto vamos a realizar el desarrollo e implementación de un agente de gestión SNMP sobre una Raspberry Pi 3. En el proyecto los agentes se implementan con el software libre Net-SNMP utilizando AgentX para extenderle. Mediante el desarrollo de las programas o demonios conseguimos el correcto funcionamiento de las funciones descritas las cuales necesarias para que el agente extendido funcione de forma correcta y autónoma. En el proyecto se implemente una pequeña estación meteorológica la cual dará soporte el soporte para la obtención de datos y comprobación del correcto funcionamiento de nuestra plataforma SNMP. Toda la parte de ejecución y desarrollo de las funciones y demonios se ha automatizado utilizando scripts en "bash" con los cuales nos facilitara la ejecución y comprobación de los resultados de la simulación. El desarrollo de este proyecto se utilizará también para fines docentes en la asignatura Gestión y Operación de Redes, dado que además se ha desarrollado una práctica con instrumentos para la comprobación del funcionamiento en el laboratorio de dicha asignatura.

SUMMARY

In this project we will perform the development and implementation of an SNMP management agent on a Raspberry Pi 3. In the project the agents are implemented with the free Net-SNMP software using AgentX to extend it. Through the development of the programs or demons we get the correct functioning of the described functions which are necessary for the extended agent to function properly and autonomously. The project will implement a small meteorological station which will support the support for obtaining data and checking the correct operation of our SNMP platform. All the execution and development of the functions and demons has been automated using scripts in "bash" with which we will facilitate the execution and verification of the results of the simulation. The development of this project will also be used for teaching purposes in the course" Gestión y Operación de Redes", since a practice has also been developed with instruments to verify the operation in the laboratory of this subject.

Índice Del Contenido

Índice Anexos

Índice Figuras

1. Introducción

En este capítulo inician se tratarán las cuestiones previas relativas al proyecto: motivaciones para la realización del proyecto, objetivos planteados y estructura de la memoria.

Desde el origen de TCP/IP (1969) se utiliza para gestión herramientas basadas en el protocolo ICMP (Internet-Control Message Protocol). La principal herramienta es PING (Packet INternet Groper), que permite comprobar la comunicación entre dos máquinas, calcular tiempos medios de respuesta y pérdidas de paquetes, dado que es parte de la familia de protocolos TCP/IP, todas las máquinas disponen de este protocolo. El problema surge con el crecimiento exponencial de Internet a partir de los años 80, lo que produce la necesidad de herramientas estándar de gestión más potentes porque el crecimiento de la red es de manera relativamente caótica por estar descentralizada y apoyada en recursos de red, hardware y software con fiabilidades y capacidades muy diversas.

En los años 80 surgen tres propuestas de estándar de protocolo de gestión para TCP/IP:

– HEMS (High-level entity-management system), que es una generalización del que fue quizás el primer protocolo de gestión usado en Internet (HMP).

– SNMP (Simple network management protocol), que es una versión mejorada de SGMP (Simple gateway-monitoring protocol).

– CMOT (CMIP over TCP/IP), que intenta incorporar, hasta donde sea posible, el protocolo (common management information protocol), servicios y estructura de base de dato que se están estandarizando por ISO.

A principios de 1988, el IAB recibe las propuestas y decide el desarrollo de SNMP como solución a corto plazo y CMOT a largo plazo (ya que supone que con el tiempo las redes TCP/IP evolucionarán a redes OSI). HEMS es más potente que SNMP, pero como se supone que se va a producir la transición a OSI, se elige SNMP por ser más simple y necesitar menos esfuerzo para desarrollarse.

SNMP se estandariza en los años 90/91, y aunque era una solución simple a corto plazo, el retraso en la aparición de redes OSI y la gran cantidad de redes TCP/IP, le augura una larga vida, mientras que los trabajos en CMOT se ralentizan. En la actualidad es un estándar utilizado universalmente y se está ampliando a todo tipo de redes (no sólo TCP/IP) incluido OSI. Durante su historia ha ido evolucionado desde el estándar simple original hasta una serie de mejoras. En las principales extensiones aumentan su funcionalidad y cubren problemas de seguridad detectados en el protocolo original. La mejora más destacable se trata RMON (remote-monitoring), que permite monitorizar subredes como un todo, además de equipos individuales.

1.1. Motivaciones

SNMP nos proporciona la opción de la gestión de red para el control distribuido en cualquier sistema de comunicaciones en red. Gracias a SNMP se ofrece la gestión remota de los dispositivos conectados en una red de comunicaciones.

Debido al auge de las computadoras de bajo coste se ha planteado en el proyecto el desarrollo de una simulación y gestión de una red doméstica. Se basará en el uso de la Raspberry Pi y de sensores de temperatura conectados a la misma.

Raspberry Pi se trata del dispositivo más conocido y más vendido dentro del mundo de las Single Boad Computer por lo que la comunidad está en continuo crecimiento gracias a su fácil manejo y el software libre en la que se basa los programas desarrollados para ella.

En el proyecto como anteriormente se mencionaba se busca el desarrollo de una plataforma que gestione y administre los sensores de temperatura y humedad añadidos a la Raspberry Pi como si de una aplicación de domótica se tratara.

La implementación de un agente SNMP extensible mediante AgentX, facilitará la opción de añadir nuevos dispositivos y de dar soporte a cualquier dispositivo o conjunto de dispositivos que se añadan a la red.

1.2. Objetivos

En el planteamiento del proyecto se proponen los siguientes objetivos:

- Creación de un agente extensible que funcione de forma autónoma basado en SNMP y desarrollado sobre una Raspberry Pi 3. Investigación y elección de los elementos más óptimos para las necesidades del entorno de desarrollo.

- Diseño del agente extensible, explicando el uso detallado del hardware y software utilizado

- Implementación del dispositivo de forma autónoma. Una vez inicializado funcionara de forma automática.

- Diseño de un entorno para la gestión y automatización del proyecto.

- Implementación del prototipo real basado en sensores de temperatura y humedad para la comprobación del correcto funcionamiento del protocolo SNMP.

- Implementación de una plataforma para la visualización de los sensores online

- Desarrollo de todo el proyecto en un entorno de software libre, para ofrecer la posibilidad a cualquier desarrollador de una posterior utilización o mejora de dicho proyecto.

- Comprobación del funcionamiento del dispositivo finalizado en el entorno del Laboratorio de Telemática de la ETSIIT de la Universidad de Cantabria con el fin de utilizarlo para uso educativo, concretamente en las prácticas de la asignatura *Gestión y Operación de Redes*, perteneciente al programa del Grado en Ingeniería de Tecnologías de Telecomunicación de la Universidad de Cantabria.

1.3. Estructura del Documento

El documento está estructurado en tres bloques que están organizados de la siguiente manera:

- Una parte teórica en el capítulo 2. En el capítulo se explicarán los todos los conceptos relacionados con la gestión de SNMP y los agentes SNMP extendidos.

- Una parte de aspectos prácticos en el capítulo 3, en el cual se explicarán todo lo referente al hardware y software (Raspberry Pi, Sensores, Net-SNMP) y sobre la creación de los agentes extendidos. Además, tendremos la explicación de los objetos gestionados de los agentes extendidos.

- Descripción y desarrollo de la aplicación práctica, en el capítulo 4. Esta aplicación práctica se trata de una simulación de una estación meteorológica con sensores reales conectados a la Raspberry Pi.

- Practica de laboratorio para la asignatura *Gestión y Operación de Redes*.

- Conjunto de anexos utilizados en el desarrollo del proyecto para la consulta más detallada de los aspectos desarrollados en el proyecto.
-

2. Aspectos Teóricos

En internet se conectan varias redes entre sí con el uso de routers y un protocolo de interconexión de redes, de modo que los routers usan el protocolo para encubrir las características de las redes y proporcionar un servicio uniforme entre ellas, es decir, aunque cada red use una tecnología distinta y unas reglas específicas de transmisión, los hosts de cada red ven a la red de igual manera.

La principal tecnología de interconexión de redes es el conjunto de protocolos de Internet llamados TCP/IP, que son los que se usan en la Red Internet, (red de redes global) pero también en la interconexión de redes menores internet, (redes locales).

SNMP se sitúa en el tope de la capa de transporte de la pila OSI, o por encima de la capa UDP de la pila de protocolos TCP/IP. Siempre en la capa de transporte como se puede observar en la Figura 2.1.

2.1. Necesidad de administrar las redes

Dispositivos diferentes: La interconexión de redes permite diferentes tipos de dispositivos y estos son de distintos vendedores, todos ellos soportando el protocolo TCP/IP. Por lo que, para administrar estas redes, habrá que usar una tecnología de administración de redes abierta.

Administraciones diferentes: Como se permite la interconexión entre redes de distinto propósito y distinto tamaño, también están administradas, gestionadas y financiadas de distinta forma.

2.1. Evolución Histórica del Protocolo Simple de Gestión de Red (SNMP)

El protocolo SNMPv1 fue diseñado a mediados de los 80 por Case, McCloghrie, Rose, and Waldbusser, como una solución a los problemas de comunicación entre diferentes tipos de redes.

<table>
<tr><th>OSI Model Layers</th><th>TCP/IP Protocol Architecture Layers</th><th>TCP/IP Protocol Suite</th></tr>
<tr><td>Application Layer
Presentation Layer
Session Layer</td><td>Application Layer</td><td>FTP
HTTP
SMTP
SNMP
TELNET</td></tr>
<tr><td>Transport Layer</td><td>Host-to-Host Transport Layer</td><td>TCP UDP</td></tr>
<tr><td>Network Layer</td><td>Internet Layer</td><td>IP
ARP RARP ICMP</td></tr>
<tr><td>Data link Layer
Physical Layer</td><td>Network Interface Layer</td><td>Ethernet
Fast Ethernet
Token Ring
FDDI</td></tr>
</table>

Figura 2.1. Esquema OSI SNMP

El manejo de este protocolo era simple, se basaba en el intercambio de información de red a través de mensajes(PDU's). Debido a ser un protocolo fácilmente extensible a toda la red su uso se estandarizo entre usuarios y empresas que no querían demasiadas complicaciones en la gestión de sus sistemas informáticos.

No obstante, este protocolo no era perfecto, además no estaba pensado para poder gestionar la inmensa cantidad de redes que cada día iban apareciendo. Para subsanar sus carencias surgió la versión 2. Las mayores innovaciones son mecanismos de seguridad (Protegen la privacidad de datos, confieren autentificación a los usuarios y controlan el acceso), mayor detalle en la definición de las variables y se añaden estructuras de la tabla de datos para facilitar el manejo de los datos. El hecho de poder usar tablas hace aumentar el número de objetos capaces de gestionar, con lo que el aumento de redes dejó de ser un problema.

2.2. Modelos

Se nombrará los distintos elementos que se observaran en un modelo de nuestro entorno SNMP. Aunque se mencionará todos solamente se explicará el modelo usado en el proyecto.

El primer elemento es uno en el que los nodos son administrables, conteniendo cada uno un agente. El segundo se trata de una estación de administración de red (NMS) con soporte para una o más aplicaciones de administración de la red. Otro elemento se trata de la posibilidad de una o más entidades de doble función, esto indica que pueden actuar tanto como agente o como administrador. El siguiente elemento es el que vamos a explicar con un poco más de detalle dado que se trata del modelo que usaremos en el desarrollo de nuestro proyecto, se trata de un modelo que se basa en un protocolo de administración de red, que es usado por el gestor y los agentes para el intercambio de información. El último elemento se trata de la información que debemos administrar.

Información de Administración: Una unidad de información de administración se denomina objeto administrado, y un conjunto de dichos objetos se denomina Módulo de Base de Información de Administración (Módulo MIB).

La característica más importante de los métodos usados para describir la información de administración es la extensibilidad.

Objetos Administrados: La instrumentación de administración actúa entre los protocolos del dispositivo y el protocolo de administración, tomando la información del dispositivo y presentándola como un conjunto de objetos administrados. Esta colección se denomina Recursos Objeto del Agente.

Modelo Administrativo: El modelo administrativo proporciona políticas de autorización y autentificación. Además, determina qué aplicación de administración puede acceder a qué objeto y con qué operaciones. Las operaciones de administración permitidas a una aplicación por un agente se denomina política de acceso; la colección de objetos administrados que son visibles para estas operaciones se denomina Vista del MIB, o simplemente Vista.

2.2.1. Modelo de Información

La Estructura de la Información de Administración (SMI) define las reglas de la información de administración independientemente de los detalles de implementación. La SMI se define usando ASN.1[1] (Abstract Syntax Notation).

Si se piensa que unas colecciones de objetos administrados están almacenadas, por ejemplo, en una base de datos, la SMI define el esquema de esa base de datos. En realidad, esa base de datos se denomina Base de Información de Administración (MIB).

Los Módulos MIB: El fundamento de la gestión de una red serán todas aquellas estructuras que se encarguen de almacenar los datos que se gestionarán en el sistema. Posteriormente una aplicación SNMP, ejecutándose sobre un administrador de dicho protocolo, será la encargada de visualizar parte o la totalidad de los datos que son soportados por dichas estructuras. Este módulo, será el encargado de almacenar estas y otras estructuras que harán falta para justificar la valía de los datos. En la Figura 2.2 se encuentra la estructura del Árbol OSI en la que están situadas las diferentes clases de módulos MIB que dividen en tres:

- *Estándar:* Diseñados por un grupo de trabajo del IETF (Internet Engineering task force) y estandarizado por el IESG (Internet Engineering Steering Group). Los prefijos de los identificadores de objetos se encuentran bajo el subárbol mgmt.

- *Experimental:* Mientras un grupo de trabajo desarrolla un MIB, los identificadores de objetos temporales se colocan bajo el subárbol experimental. Si el MIB adquiere la condición de estándar, se colocan los identificadores bajo el subárbol mgmt.

- *Private:* La mayor parte de las empresas desarrollan módulos MIB propios que soporten ciertas características particulares, las cuales no son generalmente contempladas en los módulos MIB estándar.

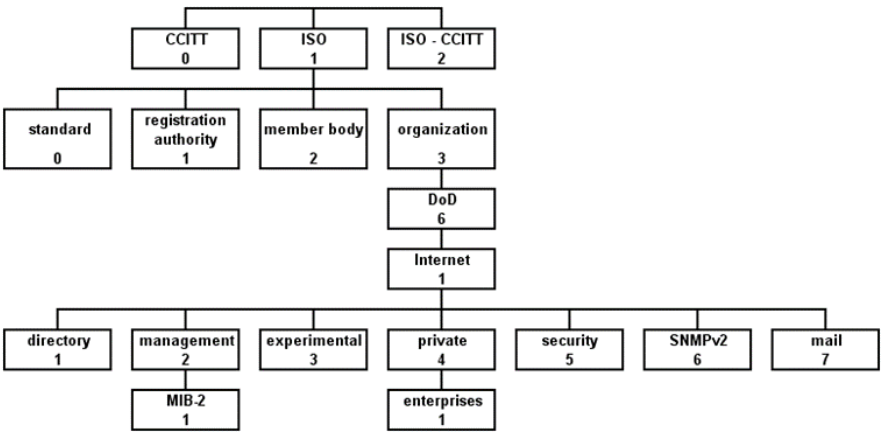


Figura 2.2 Árbol OSI

Importando Definiciones de Macros: Todos los módulos de información empiezan con una invocación de la macro MODULE-IDENTITY, que proporciona la lista de contactos y revisiones. Esta llamada debe aparecer siempre después de cualquier declaración IMPORT o EXPORT.

2.2.2. Modelo Administrativo

Consideraremos como se definen políticas administrativas para aplicaciones de gestión y agentes:

Conceptos: En el entorno de gestión, una entidad SNMP es una "entidad lógica" en nombre de la cual un agente o una aplicación de gestión están procesando un mensaje. El entorno de gestión es responsable de proporcionar los siguientes atributos.

- *Autentificación:* como las entidades SNMP identifican sus mensajes.
- *Privacidad:* como las entidades SNMP protegen sus mensajes.
- *Autorización:* como una entidad agente SNMP determina los objetos que son accesibles a una entidad de aplicación de gestión dada, y las operaciones que se pueden realizar en estos objetos.

Comunidades: SNMP v2 está diseñado para soportar múltiples entornos de administración. El entorno que veremos se denomina entorno de gestión basado en comunidades, debido a que usa el concepto de comunidad empleado en el SNMP original.

SNMP define una comunidad como una relación entre entidades SNMP. Una comunidad SNMP se escribe como una cadena de octetos sin interpretación. Esta cadena se llama nombre de comunidad. Cada octeto toma un valor entre 0 y 255.

Cuando se intercambian mensajes SNMP, contienen dos partes, una cadena de comunidad que es mandada en texto sencillo y datos, que contienen una operación SNMP y los operandos asociados.

La cadena de comunidad es un simple manejador para las relaciones de gestión. Propiedades de gestión de SNMP:

- Identificación origen: como las cadenas de comunidad son enviadas sin protección, cualquier tercera parte capaz de interceptar un mensaje SNMP puede usar el mismo nombre de comunidad y de esa forma demandar ser un miembro de la comunidad de mensajes.
- *Integridad del mensaje:* cualquier tercera parte puede modificar un mensaje SNMP que intercepte.
- *Protección limitada de reenvíos:* cualquier tercera parte puede retrasar un mensaje SNMP que haya interceptado.
- *Privacidad:* cualquier tercera parte puede leer el mensaje SNMP que haya interceptado.

- *Autorización:* los agentes son responsables de mantener información local, así como los MIB que contiene, o las relaciones de proxy válidas; será sencillo para una tercera parte obtener los accesos correctos de una entidad autorizada para monitorizar o controlar esos objetos.

Procedimientos: Cada mensaje SNMP tiene 3 campos que se definen en la siguiente estructura. En la Figura 2.3 se puede visualizar la estructura de un mensaje SNMP

- *Versión:* el número de versión del mensaje.

- *Comunidad:* la cadena de comunidad.

- *Datos:* una operación SNMP, definido como una estructura PDUs.

Figura 2.3. Estructura Mensaje SNMP

Originando un mensaje: Usando conocimiento local, la entidad origen comienza seleccionando la comunidad apropiada la cual tiene la autorización adecuada para usar las operaciones y el acceso a los objetos MIB deseados. Luego la estructura de mensaje es construida desde esta información, es convertida en una cadena de octetos y finalmente es enviada a la dirección de transporte de la entidad receptora.

Recibiendo un mensaje: Si la comunidad se refiere a recursos de objetos locales, entonces la operación se desarrolla de acuerdo con los MIBs apropiados asociados con la comunidad.

En cambio, si la comunidad se refiere a recursos de objetos remotos, entonces si la operación es una respuesta, es correlativa con la anterior petición y una respuesta es enviada a la entidad originaria de la petición. Si la operación es una Trap o un informe, entonces el agente proxy, usando conocimiento local, determina la entidad SNMP que debería enviar el mensaje. de otra forma, La petición se propaga por la relación proxy definida por la comunidad.

Esperando por mensajes: Normalmente las entidades SNMP esperan los mensajes en la dirección de transporte por defecto asociada con cada dominio de transporte válido para él.

2.2.3. Modelo Operacional

SNMP es un protocolo asíncrono de petición-respuesta basado en el modelo de interrupción-sondeo directo, esto significa que una entidad no necesita esperar una respuesta después de enviar un mensaje, por lo que puede enviar otros mensajes o realizar otras actividades.

SNMP tiene pocos requisitos de transporte ya que usa un servicio de transporte no orientado a conexión, y que normalmente es UDP. Aunque esto ratifica el Axioma Fundamental, hay una razón más importante: Las funciones de administración de red se realizan cuando hay problemas, de modo

que la aplicación de administración es la que decide qué restricciones realiza para el tráfico de administración. La elección de un servicio de transporte no orientado a conexión permite a la estación determinar el nivel de retransmisión necesario para complacer a las redes congestionadas.

- **Interacciones del Protocolo**

Una interacción SNMP consiste en una petición de algún tipo, seguida por una respuesta. Hay cuatro resultados posibles de una operación:

- Una respuesta sin excepción o error.
- Una respuesta con una o más excepciones.
- Una respuesta con error.
- Sobrepasar el tiempo de espera (timeout).

A continuación, se describen los campos del tipo de dato PDU y en la Figura 2.4 observaremos su estructura.

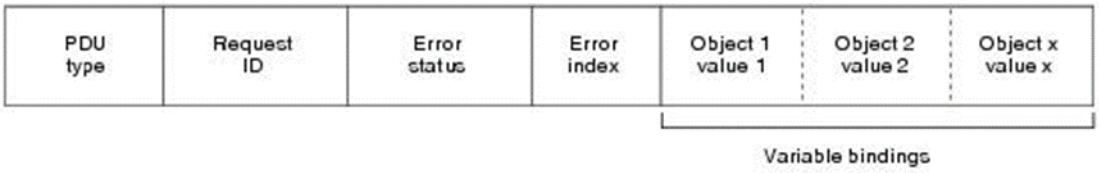


Figura 2.4. Campos PDU

- *request-id,* valor entero usado por la aplicación para distinguir entre peticiones pendientes, lo que permite a la aplicación mandar rápidamente varios mensajes SNMP, reconocer mensajes duplicados en la red y medir el tiempo del tráfico que genera. Los agentes no pueden modificar este campo.
- *error-status*, si no es cero, representa un error al procesar la petición y que debería ignorarse el campo variable-bindings.
- *error-index,* si no es cero indica qué variable de la petición es errónea.
- *variable-bindings*, Lista de variables, con su nombre y valor.

Las operaciones de SNMPv2 se pueden clasificar así:

- *Recuperation:* get, get-next y get-bulk.
- *Modificación:* set.

- *Invocación:* snmpv2-trap.

- *Administradores:* inform.

Peticiones de Recuperación

Para examinar eficientemente la información de administración, el entorno de administración usa un método inteligente para identificar las instancias: Es usado un OBJECT IDENTIFIER, formado por la concatenación del nombre del tipo objeto con un sufijo.

- **El Operador Get**

Si la aplicación de administración conoce las instancias que necesita, realiza una get-request. Sólo se pueden devolver los errores tooBig y genErr. Por otro lado, para cada variable de la petición, la instancia se recupera de la vista del MIB con esta operación:

Si el agente no implementa el tipo objeto asociado con la variable, se devuelve la - excepción noSuchObject. Si la instancia no existe o no la soporta el MIB, se devuelve la excepción noSuchInstance.

- **El Operador Get-Next**

Si la aplicación no conoce exactamente la instancia de la información que necesita, realiza una get-next-request. Sólo se pueden devolver los errores tooBig y genErr.

Por otro lado, para cada variable de la petición, se devuelve la primera instancia que siga a la variable que esté en la vista del MIB con esta operación:

- Si no hay una próxima instancia para la variable en el MIB, se devuelve una excepción endOfMibView.

- Si se reconoce la siguiente instancia de la variable en el MIB, se devuelve el valor asociado. El operador get-next puede usarse para comprobar si un objeto es soportado por un agente.

Debido al almacenamiento que se hace en el MIB, las tablas se examinan en un orden columna-fila; así, se examina cada instancia de la primera columna, cada instancia de la segunda y así seguidamente hasta el final de la tabla. Entonces, se devuelve la instancia del siguiente objeto, y sólo se devuelve una excepción si el operando de get-next es mayor, lexicográficamente hablando, que la mayor instancia del MIB.

Como el operador get-next soporta múltiples operandos, se puede examinar eficientemente la tabla entera. La aplicación de administración conoce que llega al final de la tabla cuando se devuelve un nombre cuyo prefijo no coincide con el del objeto deseado.

- **El Operador Get-Bulk**

Su función es minimizar las interacciones en la red, permitiendo al agente devolver paquetes grandes. Este esquema tiene que funcionar con un servicio de transporte no orientado a conexión de modo que la aplicación sea la responsable de controlar las interacciones. Las aplicaciones de administración también deben controlar los tiempos de las interacciones.

Cuando un agente recibe una get-bulk, calcula el mínimo del tamaño máximo del mensaje del emisor y del tamaño de la generación de mensajes del propio agente.

De aquí resta la suma de dos cantidades, el tamaño de las capas de privacidad/autentificación de la generación de la respuesta y del tamaño de una respuesta sin variables instanciadas.

Dicha diferencia indica la cantidad máxima de espacio posible para las instancias de las variables en la respuesta. Si la diferencia es menor de cero, la respuesta se pierde, si no, se genera la respuesta, que tendrá cero o más variables instanciadas.

El agente examina las variables de la petición usando el operador get-next y añadiendo cada nueva instancia con su valor a la respuesta y disminuyendo la cantidad de espacio libre. Si no hay suficiente espacio, se envía la respuesta antes de colapsar el espacio libre.

Finalmente, el espacio libre se ocupa, o se realiza el máximo número de repeticiones. Es importante tener en cuenta que el agente puede terminar una repetición en cualquier momento.

Peticiones de Modificación

El operador set es una aplicación de administración que conoce exactamente las instancias que necesita y genera un set-request.

La semántica de la operación set es tal que la operación tiene éxito si y sólo si todas las variables se actualizan. Antes de empezar, se asegura que la respuesta no sea tan grande como para no poder ser enviada, ya que, si no, se generaría un error tooBig.

Cada instancia se examina y se hace una comprobación para verificar que la variable es soportada por la vista del MIB para esa operación y si la variable existe, el agente puede modificar las instancias del tipo objeto correspondiente y el valor proporcionado es correcto sintácticamente.

Si la variable no existe, el agente devuelve el error apropiado y se liberan los recursos.

Interacciones de Invocación

Cuando un agente detecta un evento extraordinario, se genera una invocación snmpV2-trap, que se envía a una o más aplicaciones de administración. La invocación snmpV2-trap tiene un formato idéntico a las PDU usadas en otras peticiones.

Las dos primeras instancias son especiales:

- sysUpTime.0, momento en que se generó la invocación.

- snmpTrapOID.0, el identificador de objeto de la invocación.

Interacciones entre Administradores

Cuando una aplicación quiere informar a otra aplicación, genera una inform-request. El formato de la InformRequest-PDU es idéntico al de las otras PDU del resto de peticiones. Al igual que snmp V2-trap, las dos primeras instancias indican el momento del evento y su identidad.

Como es de esperar, sólo algunos dispositivos pueden tener el papel de administrador. Hay que tener cuidado para minimizar el número de informes que se generan. Actualmente, el control de la generación y retransmisión de informes es una tarea específica de implementación.

Las entidades de doble rol se tratan de dispositivos que contienen agentes y aplicaciones de administración. Estos dispositivos recogen y procesan información de los agentes y la proporcionan a los administradores. Así, según el entorno SNMPv2, una aplicación de administración es algo que inicia una interacción entre peticiones y respuestas.

- **Mapeo de transporte**

Las operaciones SNMP son independientes del protocolo de transporte, utilizando sólo un servicio de transporte no orientado a conexión. Para definir el mapeo de transporte, se especifican dos reglas, una para tomar la estructura de un mensaje y transformarlo en una cadena de octetos para formar un paquete y la otra para enviar el paquete por el servicio de transporte.

Hay varios mapeos de transportes definidos y usan el mismo conjunto de reglas para el primer paso. Como todos los objetos y estructuras SNMP se definen con el ASN.1 de OSI, el entorno de administración usa BER (Basic Encoding Rules) de OSI para codificar las estructuras en octetos. Cuando éstos se reciben, se transforman en una estructura de datos con una semántica idéntica.

- **Direcciones y Dominios de Transporte**

En la Figura 2.5 se puede observar la relación del protocolo de administración y el servicio de transporte, el cual se denomina Dominio de Transporte. La entidad que envía transforma y envía el mensaje como un único datagrama UDP a la dirección de transporte de la entidad receptora. Por convención, los agentes SNMP escuchan en el puerto UDP 161 y envían las notificaciones al puerto UDP 162, aunque una entidad debería configurarse para usar cualquier selector de transporte aceptable.

Todas las entidades receptoras deben admitir mensajes de al menos 1472 octetos de longitud y su estructura observaremos en la Figura 2.6[2].

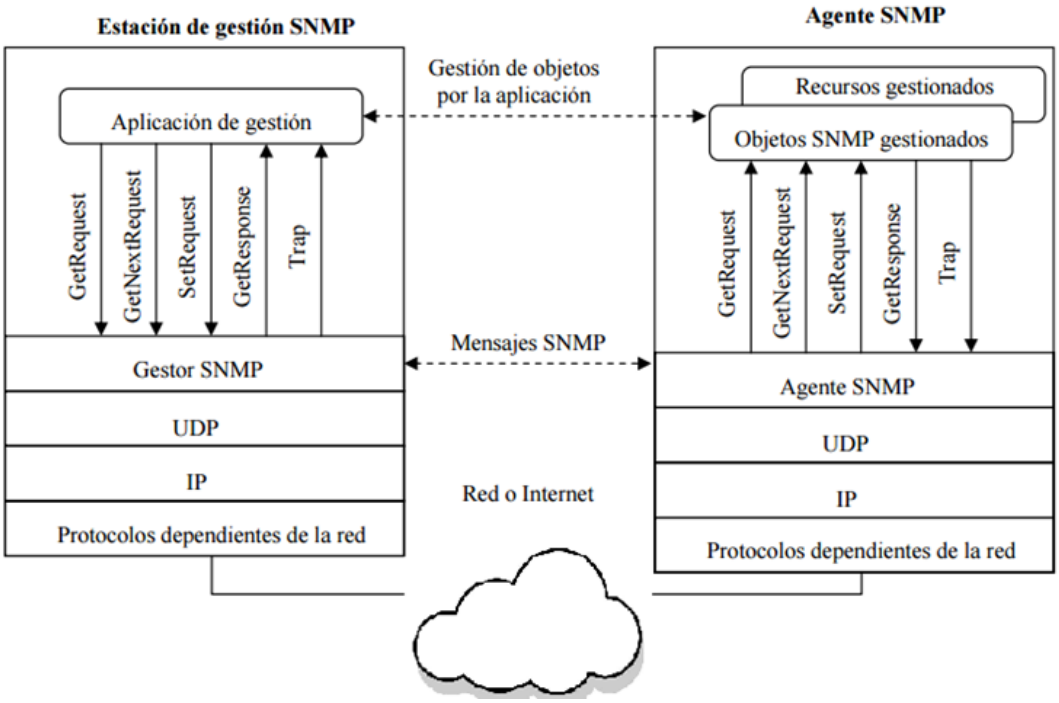


Figura 2.5. Gestor-Agente Transporte

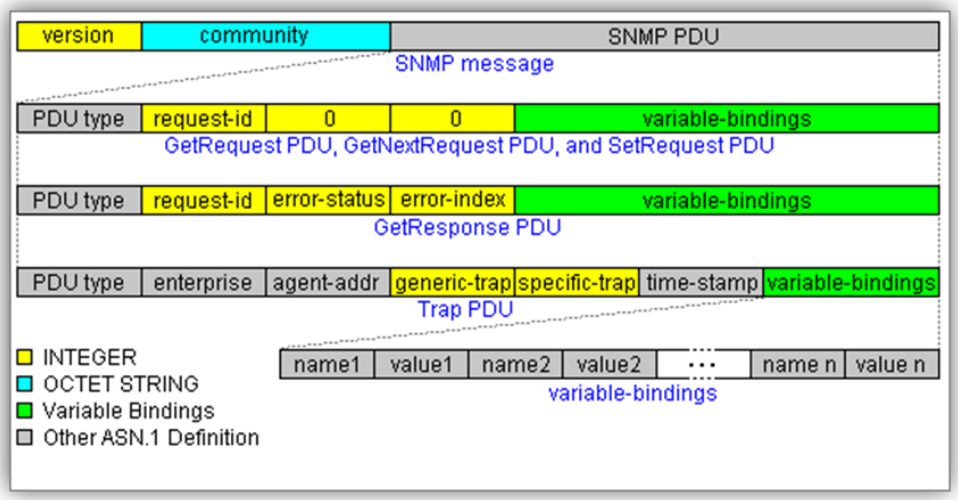


Figura 2.6. Esquema PDUs

2.3. MIB

La Base de Información para Gestión (Management Information Base o MIB) es un tipo de base de datos que contiene información jerárquica, estructurada en forma de árbol, de todos los dispositivos gestionados en una red de comunicaciones.

Es parte de la gestión de red definida en el modelo OSI. Define las variables usadas por el protocolo SNMP para supervisar y controlar los componentes de una red. Está compuesta por una serie de objetos que representan los dispositivos (como enrutadores y conmutadores) en la red. Cada objeto manejado en un MIB tiene un identificador de objeto único e incluye el tipo de objeto (tal como contador, secuencia o indicador), el nivel de acceso (tal como lectura y escritura), restricciones de tamaño, y la

información del rango del objeto. Los formatos del MIB de CMIP y del SNMP se diferencian en estructura y complejidad. Los objetos de una MIB se definen usando un subconjunto del ASN.1, la versión 2 de la estructura de la información de gestión (Structure of Management Information Version 2 o SMIv2) definido en el RFC 2578.

Los MIB tienen un formato común de modo que aun cuando los dispositivos sean de fabricantes distintos puedan ser administrados con un protocolo muy general. El protocolo de administración es el protocolo mediante el cual se consultan los objetos administrados enviando la información a la estación administradora. Las MIBs suelen ser modificadas cada cierto tiempo para añadir nuevas funcionalidades, eliminar ambigüedades y arreglar fallos.

La MIB-II es la base de datos común para la gestión de equipos en Internet. Esta MIB se ha actualizado bastantes veces. Originalmente estaba definida en el RFC 1213[3]. Con la aparición de SNMPv2 y SNMPv3 esta MIB se amplió y se dividió en varios RFCs: RFC 4293, RFC 4022, RFC 4113, RFC 2863 y RFC 3418.

Se apoya en la estructura de información de gestión SMIv1 definido en el RFC 1155[4], que establece las bases para definir la MIB, indica los tipos de objetos que se pueden usar y define el uso de ASN.1. Existen dos tipos de nodos: estructurales y de información.

- Los nodos estructurales sólo tienen descrita su posición en el árbol. Son "ramas".

  Por ejemplo: ip OBJECT IDENTIFIER ::= { 1 3 6 1 2 1 4 }

- Los nodos con información son nodos "hoja". De ellos no cuelga ningún otro nodo. Estos nodos están basados en la macro OBJECT TYPE, por ejemplo:

  ipInReceives OBJECT TYPE SYNTAX Counter

  ACCESS read-only

  STATUS mandatory

  DESCRIPTION "texto descriptivo indicando para qué vale “

  ::= { ip 3 }

Este fragmento ASN.1 nos indica que el objeto "ipInReceives" es un contador de sólo lectura que es obligatorio incorporar si se quiere ser compatible con la MIB-II (aunque luego no se utilice) y que cuelga del nodo ip con valor tres. Como antes se ha visto el nodo estructural "ip" con su valor absoluto, veremos ver que identificador de objeto de "ipInReceives" es "1.3.6.1.2.1.4.3".

- **Estructura MIB-II**

La MIB-II se compone de los siguientes nodos estructurales los cuales visualizaremos en la Figura 2.7:

- *System:* de este nodo cuelgan objetos que proporcionan información genérica del sistema gestionado. Por ejemplo, dónde se encuentra el sistema, quién lo administra...

- *Interfaces*: En este grupo está la información de los interfaces de red presentes en el sistema. Incorpora estadísticas de los eventos ocurridos en el mismo.

- *At (address translation o traducción de direcciones):* Este nodo es obsoleto, pero se mantiene para preservar la compatibilidad con la MIB-I. En él se almacenan las direcciones de nivel de enlace correspondientes a una dirección IP.

- *Ip:* En este grupo se almacena la información relativa a la capa IP, tanto de configuración como de estadísticas.
- *Icmp:* En este nodo se almacenan contadores de los paquetes ICMP entrantes y salientes.

- *Tcp*: En este grupo está la información relativa a la configuración, estadísticas y estado actual del protocolo TCP.

- *Udp:* En este nodo está la información relativa a la configuración, estadísticas del protocolo UDP.

- *Egp:* Aquí está agrupada la información relativa a la configuración y operación del protocolo EGP.

- *Transmission:* De este nodo cuelgan grupos referidos a las distintas tecnologías del nivel de enlace implementadas en las interfaces de red del sistema gestionado.
-

- **Esquema MIB Object**

Los objetos de la MIB mencionados anteriormente tienen un formato predeterminado que está definido por cinco campos. A continuación, se ve como es la estructura o esquema de un Objeto. RFC1155

En la RFC que se ha mencionado con anterioridad está definido la estructura de cómo tiene que ser una MIB, a continuación, en la Figura 2.8, se visualiza cómo sería una estructura de la MIB con los valores que podrían tomar cada campo mencionado. Los valores que se pueden explicar son los de *Access*, el cual define el tipo de acceso que tiene el gestor sobre los datos en la MIB (read-only, read-write, write-only, not-accesible) y el valor *Status*, el que define si el valor en la MIB es de tipo (mandatory, optional, obsolete).

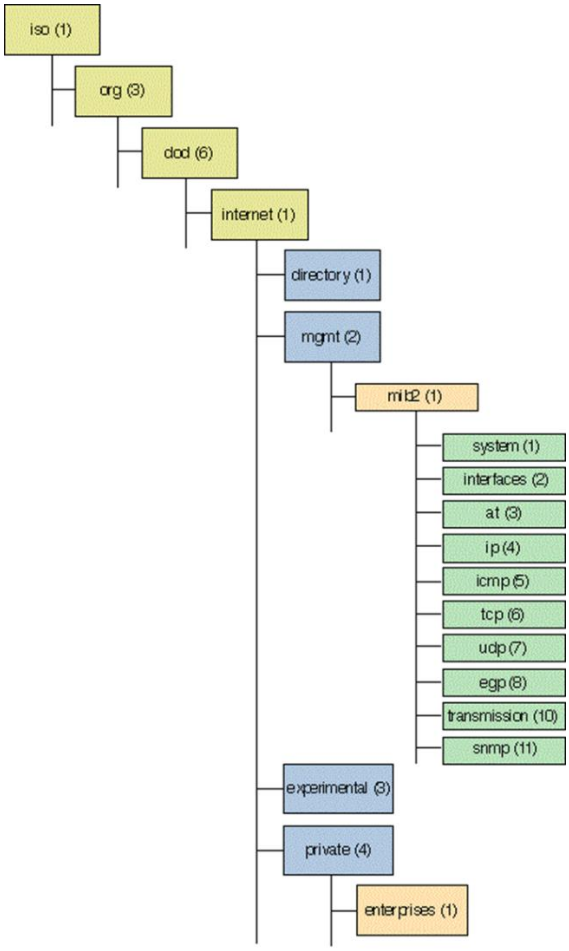


Figura 2.7. MIB-2 Tree

```
OBJECT-TYPE MACRO ::=    BEGIN


TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                          "ACCESS" Access
                          "STATUS" Status
VALUE NOTATION ::= value (VALUE ObjectName)


Access ::= "read-only"
                 | "read-write"
                 | "write-only"
                 | "not-accessible"
Status ::= "mandatory"
                 | "optional"
                 | "obsolete"
```

Figura 2.8. Estructura MIB Object. RFC 1155

En la siguiente MIB, Figura 2.9, se trata de un ejemplo de MIB obtenida de RFC1155 para visualizar cómo se completan los campos de una MIB como se ha descrito anteriormente.

En esta MIB está compuesta por un objeto de tipo INTEGER, OCTECT STRING y NetworkAddress y una Tabla (atTable) en la cual se almacenan los valores previamente descritos. Los campos de la tabla también hay que definirlos como se observa en atEntry, se trata de una secuencia de campos.

```
atIndex OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-write
        STATUS  mandatory
        ::= { atEntry 1 }

atPhysAddress OBJECT-TYPE
        SYNTAX  OCTET STRING
        ACCESS  read-write
        STATUS  mandatory
        ::= { atEntry 2 }

atNetAddress OBJECT-TYPE
        SYNTAX  NetworkAddress
        ACCESS  read-write
        STATUS  mandatory
        ::= { atEntry 3 }

atEntry OBJECT-TYPE
        SYNTAX  AtEntry
        ACCESS  read-write
        STATUS  mandatory
        ::= { atTable 1 }

atTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AtEntry
        ACCESS  read-write
        STATUS  mandatory
        ::= { at 1 }

AtEntry ::= SEQUENCE {
    atIndex
        INTEGER,
    atPhysAddress
        OCTET STRING,
    atNetAddress
        NetworkAddress
}
```

Figura 2.9. Ejemplo MIB RFC 1155

- **MIB Sensor-Temperatura-Mib**

En el apartado anterior se ha comentado las diferentes opciones que se puede observar en una MIB. En la Figura 2.10 podremos ver cómo está formada la MIB Sensor-Temperatura-Mib la cual se utilizará en el desarrollo del proyecto

```
SENSOR-TEMPERATURA-MIB DEFINITIONS ::= BEGIN

-- MIB para un TABLA para los valores de los sensores de temperatura y humedad para dht11
y dht22.

-- Escrita por Javier Garcia Nuñez para PFG para UC.

-- IMPORTS: Define el lugar en el que esta situada la MIB.

IMPORTS

       netSnmpExamples                          FROM NET-SNMP-EXAMPLES-MIB

       OBJECT-TYPE, Integer32,

       MODULE-IDENTITY                        FROM SNMPv2-SMI

       MODULE-COMPLIANCE, OBJECT-GROUP        FROM SNMPv2-CONF;

-- Informacion de la MIB:

sensorTemperaturaMIB MODULE-IDENTITY

    LAST-UPDATED "201606150000Z"            -- 15 de junio de 2016, 00.00h

    ORGANIZATION "UC"

    CONTACT-INFO "Informacion de contacto: Avenida de Los Castros"

    DESCRIPTION  "MIB para la gestion de los valores obtenidos por el sendor de temperatura
y humedad dht11 o dht22."

        ::= { netSnmpExamples 333 }    -- Esta li-nea define la ubicacion del primer nodo
de esta MIB. En concreto netSnmpExamples.333, OID: 1.3.6.1.4.1.8072.2.333.


-- Se define el nodo principal, bajo el cual estarán situados los objetos gestionados en
este prototipo.

sensorTemperaturaMIBObjects     OBJECT IDENTIFIER ::= { sensorTemperaturaMIB 1 }

-- Definicion de objetos:

--OID asociado al lugar donde esta situado los valores del sensor .

sensorPractica OBJECT-TYPE

    SYNTAX      Integer32

    MAX-ACCESS  read-write

    STATUS      current

    DESCRIPTION

    "Valor para modificar en la practica"

    ::= { sensorTemperaturaMIBObjects 1 }
```

```
--TABLA DE LOS VALORES DEL SENSoR.
sensorTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SensorEntry
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Esta tabla contiene los datos del sensor que maneja la temperatura y la humedad."
    ::= { sensorTemperaturaMIBObjects 2 }
sensorEntry OBJECT-TYPE
    SYNTAX      SensorEntry
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Fila de la tabla que contiene el nombre de lo que mide el sensor (Temperatura o
Humedad) y su valor de potencia consumida en watts."
    INDEX   { sensor }
    ::= {sensorTable 1 }
SensorEntry ::= SEQUENCE {
    sensor  OCTET STRING,
    valor   Integer32,
    }
sensor OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
    "Nombre de lo que esta midiendo el sensor (Temperatura , Humedad, etc)."
    ::= { sensorEntry 1 }
valor OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor que tiene cada una de las entradas de lo que mide el sensor."
    ::= { sensorEntry 2 }
```

```
temperaturaDHT11 OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor de la temperatura del sensor DH11"
    ::= { sensorTemperaturaMIBObjects 3  }

humedadDHT11 OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor de la humedad del sensor DH11"
    ::= { sensorTemperaturaMIBObjects 4  }

temperaturaDHT22 OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor de la temperatura del sensor DH22"
    ::= { sensorTemperaturaMIBObjects 5  }

humedadDHT22 OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor de la humedad del sensor DH22"
    ::= { sensorTemperaturaMIBObjects 6  }

-- TRAP TEMPERATURA: Trap asociado a la temperatura que queremos para que mande el aviso
de la temperatura.
trapTemperaturaDH11 NOTIFICATION-TYPE
OBJECTS {temperaturaDH11} -- cambiar al nombre del valor
```

```
STATUS  current

    DESCRIPTION

            "Trap que se envi-a cuando el sensor lee los valores de la temperatura y
supera el valor que pongamos como limite en el codigo"

    ::= { sensorTemperaturaMIBObjects 8}


-- TRAP HUMEDAD: Trap asociado a la humedad que queremos para que mande el aviso de la
humedad

trapHumedadDH11 NOTIFICATION-TYPE

       OBJECTS {humedadDH11} -- modificar

    STATUS  current

    DESCRIPTION

            "Trap que se envi-a cuando el sensor lee los valores de la humedad y supera
el valor que pongamos como limite en el codigo"

    ::= { sensorTemperaturaMIBObjects 9}



historicoTable OBJECT-TYPE

    SYNTAX      SEQUENCE OF historicoEntry

    MAX-ACCESS  read-write

    STATUS      current

    DESCRIPTION

    "Esta tabla contiene los datos del sensor que maneja la temperatura y la humedad."

    ::= { sensorTemperaturaMIBObjects 11 }


historicoEntry OBJECT-TYPE

    SYNTAX      historicoEntry

    MAX-ACCESS  read-write

    STATUS      current

    DESCRIPTION

    "Fila de la tabla que contiene el nombre de lo que mide el sensor (Temperatura o
Humedad) y su tiempo."

    INDEX   { time , sensor2 , valor2 }

    ::= {historicoTable 1 }

    }
```

```
historicoEntry ::= SEQUENCE {
       sensor2      OCTET STRING,
       valor2  OCTET STRING,
    time   OCTET STRING,
    }
sensor2 OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
    "Nombre de lo que esta midiendo el sensor (Temperatura , Humedad, etc)."
    ::= { historicoEntry 1 }

valor2 OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
    "Valor que tiene cada una de las entradas de lo que mide el sensor."
    ::= { historicoEntry 2 }

time OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
    "tiempo."
    ::= { historicoEntry 3 }

historicoControlEntry OBJECT-TYPE
    SYNTAX      historicoControlEntry
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Fila de la tabla que contiene el nombre de lo que mide el sensor (Temperatura o
Humedad) y su tiempo."
    INDEX   { sensor3 , sondeo , medida , max }
```

```
    ::= {historicoControlTable 1 }
historicoControlEntry ::= SEQUENCE {
       sensor3      OCTET STRING,
       valor3 OCTET STRING,
       sondeo       OCTET STRING,
       max          OCTET STRING,
       }

sensor3      OBJECT-TYPE
       SYNTAX       OCTET STRING
       MAX-ACCESS   read-create
       STATUS       current
       DESCRIPTION
       "Nombre de lo que esta midiendo el sensor (Temperatura , Humedad, etc)."
    ::= { historicoControlEntry 1 }

medida OBJECT-TYPE
       SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
    "Tipo de medida que tiene cada una de las entradas de lo que mide el sensor."
    ::= { historicoControlEntry 2 }

sondeo OBJECT-TYPE
       SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
    "Tiempo para cada sondeo"
    ::= { historicoControlEntry 3 }

max OBJECT-TYPE
       SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
```

```
  STATUS      current
    DESCRIPTION
    "Valor MAXIMO"
       ::= { historicoControlEntry 4 }
END
```

Figura 2.10 MIB SENSOR-TEMPERATURA-MIB

- **Objeto tipo Table**

Anteriormente se ha mencionado los distintos tipos por los que está compuesta una MIB y en este apartado se va explicar el objeto gestionado por SNMP de tipo tabla. La tabla se va a actualizar de forma dinámica a través de un archivo en el cual estarán incluidos los valores. Esto hace que cualquier elemento que modifiquemos en nuestro fichero se modifique en tiempo real en nuestra MIB.

El funcionamiento de las tablas en SNMP es muy sencilla y solo hay que entender cómo es la ordenación de los elementos en las tablas. Las tablas en SNMP siguen una organización que se llama orden lexicográfico. Las tablas están formadas por secuencias de elementos en los cuales cada elemento de esta fila corresponde a una columna y uno de los aspectos más importantes de las tablas es que al menos uno de estos elementos debemos identificarlo como el índice.

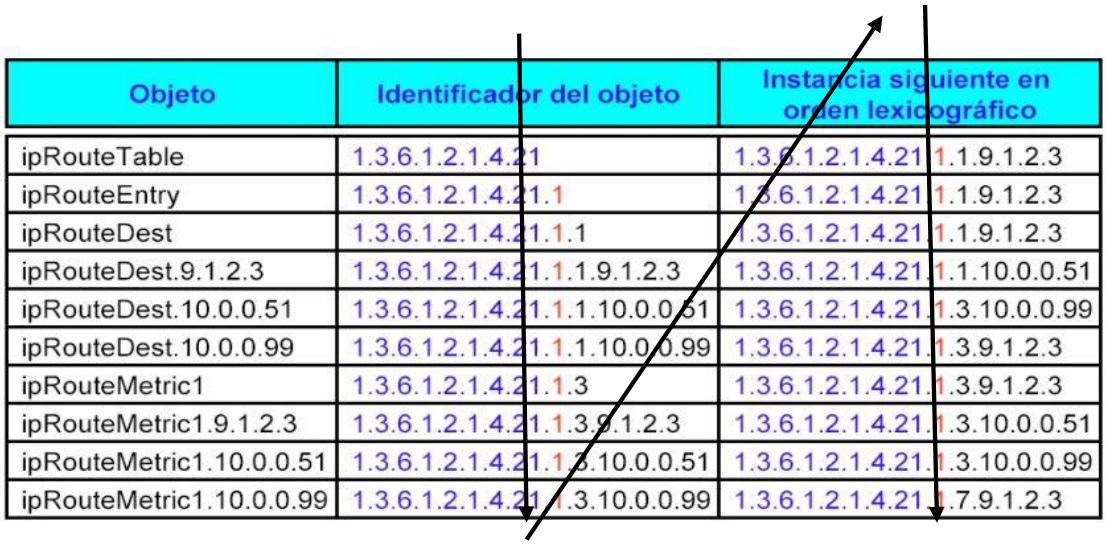
Gracias a este índice se puede ordenar la tabla y acceder a los elementos de la misma mediante las funciones SNMP como en la Figura 2.11

Se explicará cómo se accede a un elemento de la tabla. Esto se realiza utilizando el identificador del objeto (OID) y el índice que se ha definido previamente. Para acceder al elemento de la tabla vamos a hacerlo de la siguiente forma X.(NºColumna).(indice) donde X es el elemento OID al que queremos a acceder. Para acceder al elemento 4 cuyo índice tiene el valor 7 tenemos que poner el valor X.4.7. Si miramos en la tabla que acabamos de adjuntar podemos ver que el elemento X.4.7 tiene el valor de 1.3.6.1.2.1.5.21.1.3

Objeto	Identificador del objeto	Instancia siguiente en orden lexicográfico
ipRouteTable	1.3.6.1.2.1.4.21	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteEntry	1.3.6.1.2.1.4.21.1	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteDest	1.3.6.1.2.1.4.21.1.1	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteDest.9.1.2.3	1.3.6.1.2.1.4.21.1.1.9.1.2.3	1.3.6.1.2.1.4.21.1.1.10.0.0.51
ipRouteDest.10.0.0.51	1.3.6.1.2.1.4.21.1.1.10.0.0.51	1.3.6.1.2.1.4.21.1.3.10.0.0.99
ipRouteDest.10.0.0.99	1.3.6.1.2.1.4.21.1.1.10.0.0.99	1.3.6.1.2.1.4.21.1.3.9.1.2.3
ipRouteMetric1	1.3.6.1.2.1.4.21.1.3	1.3.6.1.2.1.4.21.1.3.9.1.2.3
ipRouteMetric1.9.1.2.3	1.3.6.1.2.1.4.21.1.3.9.1.2.3	1.3.6.1.2.1.4.21.1.3.10.0.0.51
ipRouteMetric1.10.0.0.51	1.3.6.1.2.1.4.21.1.3.10.0.0.51	1.3.6.1.2.1.4.21.1.3.10.0.0.99
ipRouteMetric1.10.0.0.99	1.3.6.1.2.1.4.21.1.3.10.0.0.99	1.3.6.1.2.1.4.21.1.7.9.1.2.3

Figura 2.11 Tabla Ejemplo Orden Lexicográfico

El índice que antes se ha mencionado indica cómo debemos ordenar la tabla, en la imagen que se adjunta podemos ver como la columna con el nombre objeto no se trata del índice de esta tabla por eso no se ordena a través de esa, sino que se ordena a través de la siguiente columna que es el primer índice, la columna que se llama identificador de objeto. Podemos ver como esta columna esta ordenada desde el valor más bajo al valor más alto.



Objeto	Identificador del objeto	Instancia siguiente en orden lexicográfico
ipRouteTable	1.3.6.1.2.1.4.21	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteEntry	1.3.6.1.2.1.4.21.1	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteDest	1.3.6.1.2.1.4.21.1.1	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteDest.9.1.2.3	1.3.6.1.2.1.4.21.1.1.9.1.2.3	1.3.6.1.2.1.4.21.1.1.10.0.0.51
ipRouteDest.10.0.0.51	1.3.6.1.2.1.4.21.1.1.10.0.0.51	1.3.6.1.2.1.4.21.1.3.10.0.0.99
ipRouteDest.10.0.0.99	1.3.6.1.2.1.4.21.1.1.10.0.0.99	1.3.6.1.2.1.4.21.1.3.9.1.2.3
ipRouteMetric1	1.3.6.1.2.1.4.21.1.3	1.3.6.1.2.1.4.21.1.3.9.1.2.3
ipRouteMetric1.9.1.2.3	1.3.6.1.2.1.4.21.1.3.9.1.2.3	1.3.6.1.2.1.4.21.1.3.10.0.0.51
ipRouteMetric1.10.0.0.51	1.3.6.1.2.1.4.21.1.3.10.0.0.51	1.3.6.1.2.1.4.21.1.3.10.0.0.99
ipRouteMetric1.10.0.0.99	1.3.6.1.2.1.4.21.1.3.10.0.0.99	1.3.6.1.2.1.4.21.1.7.9.1.2.3

Figura 2.12. Orden Lexicográfico

Se creará un subagente que inicializará y administrará la tabla SNMP de la MIB Sensor-Temperatura-MIB. La tabla SNMP con la que se trabajara se trata de una tabla, Figura 2.13, que está formada por dos columnas y cuatro filas. Los valores que se administrarán serán los valores de temperatura y humedad de los sensores (DHT11 y DHT22). En nuestro caso vamos a ver en la MIB la parte del código en la que está situada esta tabla. Vemos que está formado por un nodo de tipo tabla y las sucesivas entradas de esa tabla. En este caso la tabla está formada por dos elementos, sensor y valor, que son de tipo OCTET STRING y de estos dos valores el índice es el elemento sensor.

- **Objeto tipo Trap**

Las traps se tratan de los avisos de SNMP. Los avisos se envían desde los agentes a los gestores cuando ocurre algún evento generado por los objetos gestionados que se están monitorizando. Estos mensajes, en la versión 2c de SNMP, transportan la siguiente información:

- El tiempo que lleva el sistema funcionando (*sysUpTime*).
- El identificador del objeto *trap* (OID)
- En el campo *variable bindings*, una serie de valores de objetos de la MIB que se considera necesario enviar al gestor.

Se puede asociar el valor de cualquier objeto SNMP (como un umbral) a una *trap* para que el agente envíe una *trap* o notificación al gestor; de manera que se pueda monitorizar de forma automática.

```
sensorTable OBJECT-TYPE

    SYNTAX      SEQUENCE OF SensorEntry
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Esta tabla contiene los datos del sensor que maneja la temperatura y la
humedad."
    ::= { sensorTemperaturaMIBObjects 2 }

sensorEntry OBJECT-TYPE

    SYNTAX      SensorEntry
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Fila de la tabla que contiene el nombre de lo que mide el sensor (Temperatura
o Humedad) y su valor de potencia consumida en watts."
    INDEX   { sensor }
    ::= {sensorTable 1 }

SensorEntry ::= SEQUENCE {
    sensor  OCTET STRING,
    valor   OCTET STRING,
    }
sensor OBJECT-TYPE

    SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
    "Nombre de lo que esta midiendo el sensor (Temperatura , Humedad, etc)."
    ::= { sensorEntry 1 }

valor OBJECT-TYPE

    SYNTAX      OCTET STRING
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor que tiene cada una de las entradas de lo que mide el sensor."
    ::= { sensorEntry 2 }
```

Figura 2.13 MIB Tabla SENSOR-TEMPERATURA-MIB

2.4. Relación Agente-Gestor MIB

En la Figura 2.14 se observa un esquema global de cómo se relaciona el Agente y el Gestor. Cada nodo contiene un conjunto de software relacionado con la gestión y llamado entidad de gestión de red (NME - Network Management Entity)

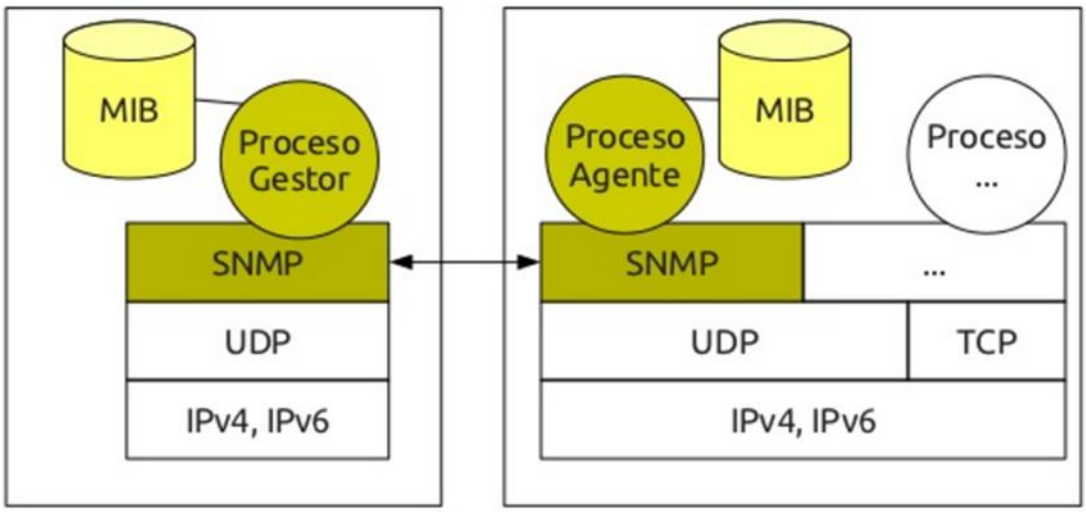


Figura 2.14. Esquema Global Agente-Gestor

Las tareas de cada NME son recoger estadísticas de actividades de comunicaciones y actividades de red, almacenar estadísticas localmente y responder a comandos del centro de control de red. Alguno de los comandos que puede responder es enviar estadísticas al centro de control de red, cambiar un parámetro, proporcionar información de estado de variables. Cada nodo con un NME se denomina agente y al menos un nodo de la red es el nodo de gestión de red o gestor (manager).

Además del NME tenemos la aplicación de gestión de red (NMA - network management application) que realiza tareas como incluir una interfaz de operador para permitir a usuarios autorizados gestionar la red, responder a comandos del usuario mostrando información y/o enviando comandos a los NME de la red y toda esta comunicación se realiza mediante un protocolo de gestión de red a nivel de "aplicación" de manera similar a cualquier otra aplicación distribuida.

Un aporte extra de estas redes es que por motivos de seguridad se suelen tener dos o más nodos de gestión de red los cuales no hacen nada o simplemente hacen funciones básicas como es recoger datos o estadísticas básicas y cuando el nodo de gestión principal falla es cuando se podría usar otro.

- **AgentX (Agente Extendido)**

El AgenteX[5] (Agente Extensibility Protocol) es un protocolo de red que permite la gestión de los objetos SNMP (Simple Network Management) definidos por los diferentes procesos a través de un solo agente maestro. Los agentes que exportan los objetos a través del AgentX al agente maestro se denominan subagentes. El estándar de AgentX no solo define el protocolo de AgenteX, sino que también el procedimiento del funcionamiento de los procesos de los subagentes.

Las operaciones internas del AgenteX son invisibles para el gestor SNMP, este no puede distinguir entre un agente no extendido y uno extendido mediante AgentX que tiene acceso a la misma instrumentación. La transparencia de los gestores es un requisito fundamental de AgentX lo cual los hace diferencia entre los subagentes AgentX de otros subagentes como agentes SNMP proxy. Esta transparencia esta descrita en el estándar RFC2741 y el esquema del agente extendido lo observamos en la Figura 2.15

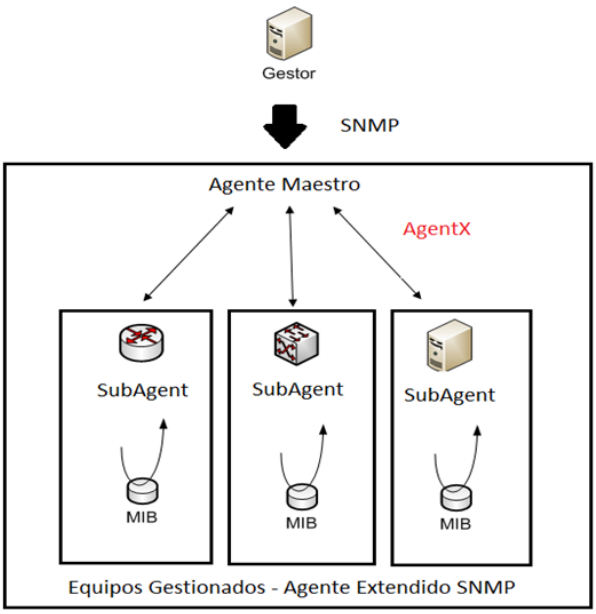


Figura 2.15. Esquema Agente Extendido SNMP con AgentX

- **Estructura AgentX**

Como se ha explicado en el apartado anterior, AgentX es un protocolo que funciona en la capa de aplicación el cual es orientado a la conexión, el agente maestro acepta las conexiones TCP en el puerto 705. Los subagentes se conectan al agente maestro usando este puerto.

La parte más importante de AgentX es la estructura de las PDUs la cual se puede observar en la Figura 2.16. Tiene campos en común en los cuales el más importante es el campo h.type dado que es el campo en el que se define qué tipo de PDU se trata.

h.version	h.type	h.flags	<reserved>
h.sesionID			
h.transactionID			
h.packetID			
h.payload length			

Figura 2.16. Cabecera PDUs AgentX

3. Aspectos Prácticos

En este apartado se explican los aspectos practicos del proyecto desarrollado. Se explicarán los componentes hardware, las aplicaciones de software utilizadas y la instalacion y desarrollo de Net-Snmp.

3.1. Hardware

Raspberry Pi

Destacamos la importancia de conocer la tecnología que integra directamente este proyecto, uno de ellos, el microcontrolador, el cual será necesario para la implementación de este proyecto y programado para realizar la medición de los sensores. En el año 2005, en el instituto de IVREA (Italia) nace el proyecto Arduino, una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

A partir de la fecha en la que nace esta plataforma surgen otras muchas, como por ejemplo Raspberry Pi, un ordenador de placa reducida o (placa única), que revoluciona el mundo de los SBC (Single Board Computer) por su pequeño tamaño y bajo costo, desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. En la Figura 3.1 se puede observar físicamente la Raspberry Pi 3[6]

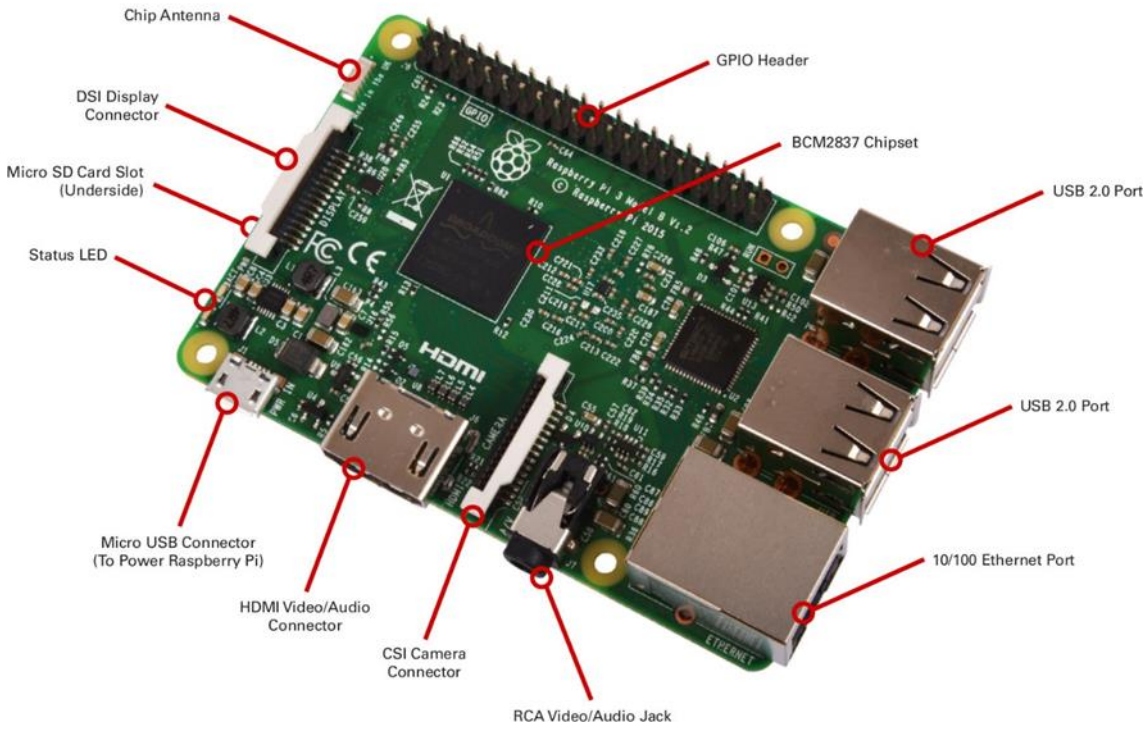


Figura 3.1. Raspberry Pi 3, PC de placa reducida.

El hardware está formado por un Chipset Broadcom BCM2837, un procesador de 64-bit quad-core ARM Cortex-A53 a 1.2GHz CPU, un procesador gráfico Videocore IV, y 1GB de memoria RAM. El sistema operativo es cargado desde una tarjeta de memoria SD que usa como dispositivo de

almacenamiento. Raspberry Pi funciona mayoritariamente con sistemas operativos basados en el núcleo Linux. En este caso se ha usado Ubuntu en el proyecto dado que la Raspberry Pi 3 tiene mejoras respecto a sus predecesoras y es capad de hacer funcionar este SO sin ningún problema. Se ha elegido la utilización de la Raspberry Pi por la gran popularidad y aceptación en el ámbito de la docencia, no solo por su bajo coste (40€), sino también por su fácil manejo y modificación gracias a ser todo open-source[7]. Este motivo es uno de los que más peso tienen para la utilización de la Raspberry Pi 3. Inicialmente se usará en el desarrollo del proyecto actual y para la después utilización del mismo en las prácticas de la asignatura de Gestión y Operación de Redes (GOR).

La Raspberry Pi 3 trae integrado un sistema GPIO (General Purpose Input/Output) es, como su propio nombre indica, un sistema de E/S (Entrada/Salida) de propósito general, es decir, una serie de conexiones que se pueden usar como entradas o salidas para usos múltiples. Estos pines están incluidos en todos los modelos de Raspberry Pi.

Los GPIO representan la interfaz entre la Raspberry Pi y el mundo exterior. Y con ellos se podrán desarrollar diversas funcionalidades, desde hacer parpadear un LED hasta la medición de valores mediante sensores. Debido a que cada Raspberry Pi tiene unos GPIO específicos se debe conocer sus características porque variarán en función de la revisión de placa o del modelo.

Figura 3.2. GPIO Raspberry Pi 3, General Purpose Input/Output

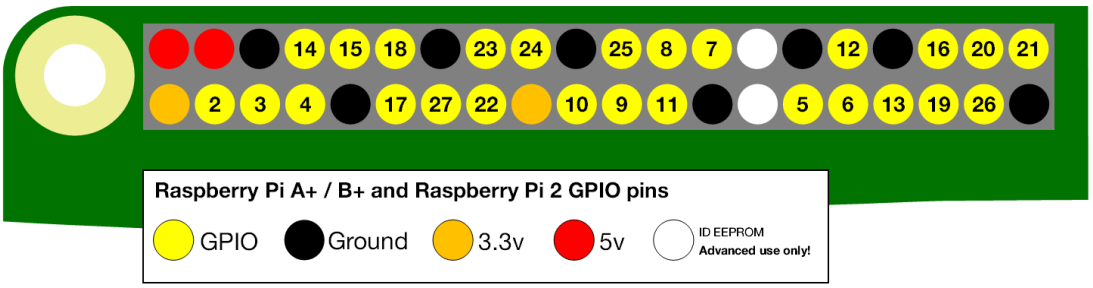


Figura 3.3. GPIO Raspberry Pi 3, GPIO NUMERACION.

SENSORES

Los sensores DHT11 y el DHT22 son dos modelos de una misma familia de sensores, que permiten realizar la medición simultánea de temperatura y humedad.

Estos sensores disponen de un procesador interno que realiza el proceso de medición, proporcionando la medición mediante una señal digital, por lo que resulta muy sencillo obtener la medición desde un microprocesador como Raspberry.

Estos sensores incluyen una medición de la humedad con un componente de tipo resistivo y un componente de medición de temperatura NTC, y se conectan a un microcontrolador de 8-bit de alto rendimiento, que ofrece una excelente calidad, una rápida respuesta, una capacidad de anti-interferencias y una buena relación calidad coste.

Cada elemento de los sensores está calibrado en el laboratorio, el cual es extremadamente precisa sobre todo en la calibración de humedad. Los coeficientes de calibración se almacenan como programas en la memoria OTP (*One Time Programmable* memory), que son utilizados por el proceso de detección de la señal interna del sensor. La interface en serie de un solo hilo hace que la integración de este sensor en el sistema digital sea de forma rápida y fácil además de que ofrece la ventaja de una transmisión por cable a distancia de hasta 20 metros.

En este proyecto se han incluido ambos sensores para hacer una comparación sobre la precisión que pueden llegar a tener cada uno de ellos. Para el uso de los sensores en proyectos que no necesitan una alta precisión como puede ser el entorno educativo se recomienda la utilización del sensor DHT11 debido a su relación calidad-precio que es mucho menos que la del sensor DHT22. En las Figuras 3.4 y 3.5 se observa como son físicamente los sensores[8][9].

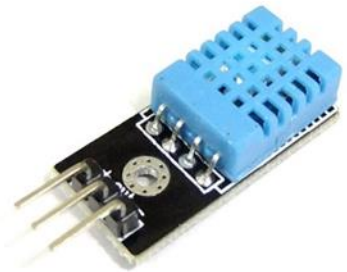


Figura 3.4. SENSOR DHT11

Figura 3.5. SENSOR DHT22

3.2. Software

En este apartado se presentará el software utilizado para el soporte de las funciones requeridas y durante el desarrollo del proyecto.

En el proyecto el sistema operativo que se ha decidido utilizar es Ubuntu, que es un sistema operativo basado en GNU/Linux y que se distribuye como software libre, el cual incluye su propio entorno de escritorio denominado Unity.

Está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario. Está compuesto de múltiple software normalmente distribuido bajo una licencia libre o de código abierto.

Al estar basado en GNU/Linux esto nos garantiza el correcto funcionamiento de las herramientas necesarias para el desarrollo e implementación del proyecto elegido basado en Net-SNMP[10], así como la mayor facilidad para la búsqueda de información y documentación sobre las herramientas que vamos a utilizar en el proyecto.

Una de las razones de mucho peso es que se necesitaba en hardware que el dispositivo fuera de bajo coste y en software que la distribución utilizada fuese de código abierto, aunque hay más distribuciones de código abierto y más ligeras que Ubuntu, como puede ser Raspbian, la Raspberry Pi 3 tiene mejoras importantes en sus componentes que hace funcionar esta distribución más pesada sin problema alguno y otro razón importante es que es la distribución de Linux con la que se trabaja en el Laboratorio de Gestión y Operación de Redes.

Para el uso de la distribución deseada se ha requerido de la utilización de una tarjeta SD vacía en la cual hay que meter la imagen obtenida desde la página oficial de Raspberry Pi. La instalación del SO Ubuntu esta desarrolla de forma automatizada y solo necesitamos inicialmente el interfaz gráfico para selección las opciones de configuración iniciales de la distribución.

Relación con del desarrollador – Raspberry Pi

Uno de los aspectos principales a nivel operativo, tanto para el desarrollo como para un uso posterior, es el acceso rápido y sencillo al dispositivo. La Raspberry Pi viene equipada con una salida de vídeo HDMI y conexiones USB, de forma que se le puede conectar directamente una pantalla, un teclado y un ratón. Pero en este caso se necesitará facilitar un acceso remoto al dispositivo, para tareas tanto de desarrollo como de configuración.

Este acceso remoto se puede realizar a través de la conexión LAN Ethernet o de forma inalámbrica utilizando una antena Wifi. En este proyecto se ha utilizado la conexión WiFi dado que en la Raspberry Pi 3 a diferencia de sus predecesoras viene equipada con WiFi incorporado y no es necesario el uso de un USB WiFi. Para el manejo remoto del escritorio de la Raspberry Pi desde un PC que tiene que estar conectada a la misma red.

Para la realización de la transferencia de archivos desde el PC a la Raspberry Pi se podía haber utilizado un USB, pero realmente es más cómodo el uso del protocolo SFTP.

Los dos métodos de comunicación que se han mencionado anteriormente para interactuar con el dispositivo:

- Escritorio remoto, para manejar el escritorio de Ubuntu desde cualquier PC. Se utiliza el protocolo de Windows para escritorios remotos, XRDP. La instalación de XRDP en el dispositivo garantiza que se pueda acceder al escritorio de Ubuntu desde cualquier equipo con Windows o GNU/Linux, dado que con la nueva Raspberry Pi tiene más potencia que sus predecesoras, esto hace que se pueda trabajar con un entorno grafico como si un PC se tratase y no es necesario el uso total de la consola para el desarrollo del proyecto.

- Cliente SFTP, para el intercambio y modificación de ficheros de forma segura y rápida. Se ha utilizado FileZilla (filezilla-project.org), también de código libre. Se trata de un cliente FTP por lo que para conectarnos a la Raspberry Pi 3 tenemos que añadir la IP que tiene en ese momento la Raspberry Pi y usar el puerto 22 que se trata del puerto que utiliza el servicio SFTP.

En la Figura 3.6 se observa un esquema simple del entorno utilizado en el desarrollo del proyecto.

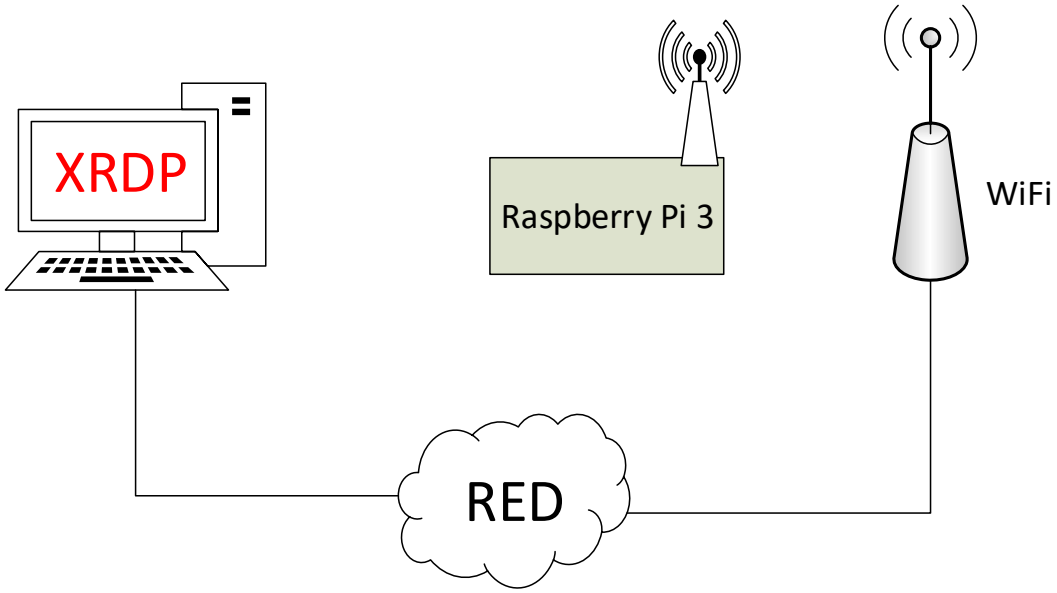


Figura 3.6 Entorno desarrollo proyecto

El ordenador esta conectado mediante escritorio remoto (XRDP) a la Raspberry Pi y esta está conecta a la red a traves de WiFi en el entorno desarrollado, aunque tambien podría estar conectada mediante Ethernet.

Aplicaciones usadas en el proyecto

El entorno de desarrollo queda a elección del desarrollador, y en este caso se ha elegido el siguiente software:

- Editores de texto, para crear y modificar el código de los programas. Aunque casi todo el código se ha desarrollado sobre Linux, también se han utilizado Notepad++ en Windows y GEDIT en Linux, que es el editor de textos por defecto del sistema operativo. Ambos son de código libre.

- Visualización de SNMP, para visualizar todo lo necesario en el proyecto sobre SNMP, objetos, arboles, notificaciones. Muchas de las pruebas se han hecho sobre Windows mediante el software MG-SOFT MIB-Browser[11], que es el software utilizado por la asignatura de Gestión y Operación de Redes (GOR). Para la visualización de Traps se ha ultilizado el programa SnmpB[12] y Wireshark. El resto de pruebas se han realizado sobre el terminal por defecto de Ubuntu (mate-terminal).

- Visualización de sensores, para la visualización de los sensores se ha usado simplemente la consola por defecto de Ubuntu (mate-terminal).

- Compiladores, se han utilizado exclusivamente los que se encuentran por defecto en cualquier sistema GNU/Linux: GCC.

- Cacti. Herramienta para la visualización online de los valores exportados por los sensores mediante el protocolo SNMP

SnmpB, tal y como está definido en su página oficial (http://snmpb.sourceforge.net/), es un navegador de MIBs SNMP que soporta SNMPv1, SNMPv2c y SNMPv3. SnmpB puede recorrer, editar, cargar y añadir archivos MIB y soporta todas las PDUs de SNMP. Soporta el descubrimiento de agentes conectados a la red, eventos generados por traps y la representación gráfica de valores asociados a los objetos gestionados alojados en cada agente. Se trata de un proyecto de software libre desarrollado y mantenido por la comunidad y actualmente se encuentra en fase beta avanzada

MG SOFT MIB Browser, por su parte, es una solución de pago más avanzada, utilizada en la asignatura de *Gestion y Operación de Redes*. Como SnmpB, se trata de un navegador de MIBs SNMP. Permite monitorizar y gestionar cualquier agente SNMP utilizando los estándares SNMPv1, SNMPv2c y SNMPv3. Aunque en el proyecto se usara siempre SNMPv2.

Soporta las PDUs Get, GetNext, GetBulk y Set. También permite capturar las traps enviadas desde cualquier dispositivo o aplicación SNMP de la red y representar gráficamente los valores de los objetos gestionados.

3.3. Net-SNMP

Simple Network Management Protocol (SNMP) es un protocolo ampliamente utilizado para el seguimiento de la salud y el bienestar de los equipos y dispositivos de red. Net-SNMP es un conjunto de aplicaciones que se utilizan para implementar SNMP v1, v2c y SNMP v3 usando IPv4 e IPv6. La suite incluye:

Aplicaciones de línea de comandos para recuperar información de un dispositivo SNMP que sea capaz, ya sea mediante solicitudes individuales (snmp_get, snmpgetnext), o múltiples peticiones (snmpwalk, snmptable), manipular la información de configuración en un dispositivo SNMP (snmpset), recuperar una colección fija de información de un dispositivo SNMP (snmpdf, snmpnetstat, snmpstatus) o convertir entre formas numéricas y textuales de MIB OID, y mostrar el contenido y la estructura MIB (snmptranslate).

Una aplicación demonio para recibir notificaciones SNMP (snmptrapd). Selección de notificaciones se pueden registrar, remitido a otro sistema de gestión SNMP, o se pasan a una aplicación externa.

Un agente extensible para responder a las consultas SNMP para la gestión de la información (snmpd). Esto incluye una función de soporte para una amplia gama de módulos de información MIB, y se puede ampliar a través de módulos de carga dinámica, scripts y comandos externos, y los protocolos tanto la multiplexación de SNMP (SMUX) y el agente de extensibilidad (AgentX).

Una biblioteca para el desarrollo de nuevas aplicaciones SNMP, tanto con las API de C y Perl.

Como hemos nombrado anteriormente usaremos demonios o demons para recibir notificaciones SNMP, pero realmente usaremos demonios para dar soporte y monitorizar a todos los objetos gestionados de la MIB. Los demonios son procesos los cuales corren en segundo plano y una vez los ejecutamos no tenemos que interactuar con ellos.

3.3.1. Instalación SNMP-MIB

Este es el proceso más importante del proyecto dado que es la instalación de la base sobre lo que funciona todo el proyecto. Se trata de la instalación del paquete Net-SNMP, el cual se realiza de forma muy sencilla dado que este paquete está en el repositorio de Ubuntu MATE.

Lo primero que vamos a hacer será actualizar e instalar los repositorios si fuera necesario y después instalaremos el agente snmp

```
sudo apt-get update (update & upgrade de los repositorios)
sudo apt-get install snmp
```

Cuando esté instalado el agente y actualizados los repositorios procederemos a instalar otro paquete que contiene información de las MIBs estándar lo que nos permitirá el acceso a la mayoría de MIBs del árbol (IETF MIB)

```
sudo apt-get install snmp-mibs-downloader
```

Si este comando fallaría podríamos forzar la descarga e instalación:

```
sudo download-mibs
```

Una vez instalado todo lo anterior procederemos a instalar el demonio, el cual será la herramienta que nos permitirá interactuar con los diferentes objetos gestionados de la MIB.

```
sudo apt-get install snmpd
```

3.3.2. Configuración SNMP-MANAGER

Anteriormente se ha descrito que como se instala algunas MIBs adicionales para acceder a la mayoría de MIBs del árbol, en este apartado podremos observar cómo se realizará la configuración para poder cargar y trabajar con las MIBs propias que se utilizaran en el proyecto dado que si no modificamos la

configuración solo se podrá trabajar y acceder a las MIBs que se incluyen en la instalación por defecto de NET-SNMP las cuales nos pueden servir para obtener información básica sobre nuestro dispositivo (Raspberry Pi 3) pero no para la realización del proyecto.

Realizaremos una modificación del fichero de configuración snmp.conf.

```
sudo nano /etc/snmp/snmp.conf
```

Al modificar el fichero se conseguirá que NET-SNMP busque en más directorios. Dentro del archivo de configuración veremos una línea que pone "mibs", en este momento tenemos dos opciones, una opción es que añadamos en la línea que se ha mencionado anteriormente el nombre de la MIB específicamente y entonces solo buscara esa MIB en concreto, la segunda opción se trata de poner "mibs +ALL". Con esto conseguiremos que se busquen todas las MIBs que haya en el directorio que tiene por defecto NET-SNMP.En la Figura 3.7 se observa la configuración del fichero snmp.conf mencionado anteriormente. Para comprobar los directorios en los cuales debemos copiar nuestra MIB tenemos que usar un comando por consola y el sistema nos devolverá la localización de donde debemos copiar los archivos MIB que vamos a utilizar en nuestro proyecto.

```
  GNU nano 2.5.3            File: /etc/snmp/snmp.conf

# As the snmp packages come without MIB files due to license reasons, loading
# of MIBs is disabled by default. If you added the MIBs you can reenable
# loading them by commenting out the following line.
#mibdirs /usr/share/snmp/mibs:/home/javier/.snmp/mibs
mibs +ALL
```

Figura 3.7 Fichero configuración snmp.conf

Ubicación MIBS

La MIB están descritas en varios directorios en la cual debemos añadirla. Pero buscando la información en los manuales realmente dicen que solo debemos añadir las MIBs en dos directorios. "*/usr/share/snmp/mibs*" y "`/home/javier/.snmp/mibs`". El primer directorio sí que estaba creado y con MIBs ya incluidas, pero el segundo directorio debemos crearlo y añadir nuestras MIBs en esos directorios.

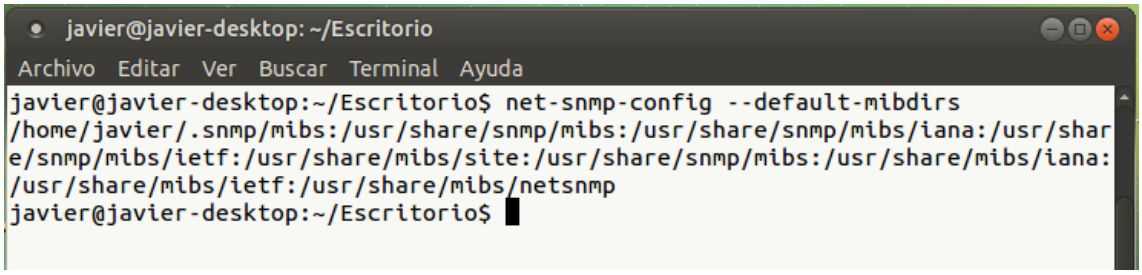


Figura 3.8. Ubicación MIBS

Cuando añadimos nuestra MIB a los directorios anteriormente mencionados es el momento en el cual podremos empezar a trabajar sobre la propia MIB. El archivo MIB incluido tiene que tener un formato ASN.1. Para que NET-SNMP detecte las nuevas MIBS debemos reiniciar el servicio snmpd . Para ello lo podemos realizar de dos formas, la primera haciendo `service snmpd restart` o la segunda y que se ve de una forma más intuitiva es con el comando `sudo /etc/init.d/snmpd restart` , al realizar este comando nos saldrá por consola que el servicio snmpd se ha reiniciado con éxito, cosa que con el otro comando no sucede, aun así debería funcionar con ambos comandos. Posteriormente se visualizará cómo será la estructura de la MIB del proyecto.

3.3.3. AgenteX SNMP

Mib2c y Configuración

Anteriormente se ha descrito los objetos que tenía la MIB del proyecto, pero para poder administrar dichos objetos se necesitan los demonios de cada objeto dado que, si no simplemente tendríamos la estructura de la MIB, pero no se podría interactuar con ellos. Para la ello se utilizará una herramienta que se llama “mib2c”[13] la cual viene incluida en el paquete NET-SNMP que se instaló inicialmente.

A la herramienta le tenemos que indicar el nombre del objeto de la MIB que queremos gestionar y luego una configuración que va a ser como este administrado este objeto. El esqueleto que genera de forma automática está formado por código en C y partes en pseudocodigo en las cuales está explicado cómo se debe completar ese código en C para poder compilarlo y obtener los demonios que manejen nuestros objetos de la MIB. Lo más complicado de esta parte es completar el código esqueleto en las partes que solo tenemos pseudocodigo dado que no hay información de cómo se debe completar y los códigos esqueletos que se generan son para casos generales y en cada MIB se trata de un caso particular.

Hay archivos de configuración predefinidos en la herramienta mib2c y en nuestro caso hemos usado “mib2c.int_watch.conf” para los escalares , “mib2c.table_data.conf” para las tablas y “mib2c.notify” para las traps.

Configuracion AgentX

Se debe conseguir que el agente se comunique con los subagentes mediante el protocolo AgentX y con los gestores mediante SNMP. Para ello hay que configurar le dispositivo (Raspberry Pi 3) para que actué como un agente extendido SNMP.

Al configurar el dispositivo como un agente extendido conseguimos que tenga la capacidad de aceptar y administrar de forma automática todos los subagentes de tipo AgentX que se requieran.

En NET-SNMP anteriormente se habló de un archivo que se llamaba “snmp.conf” en el cual se configuraba la localización y el nombre de las MIBs para que el dispositivo detectase las MIB personales. Ahora hay que modificar es el fichero de configuración “snmpd.conf”. En la Figura 3.9 se puede visualizar un fragmento del archivo snmpd.conf. Con este comando abriremos el fichero de configuración de nuestros demonios NET-SNMP.

```
sudo nano /etc/snmp/snmpd.conf
```

Con este comando abriremos el fichero de configuración de nuestros demonios NET-SNMP. A continuación, hablaremos de las modificaciones realizadas para que tenga funciones como AgentX y también alguna modificación para que funcionen cosas básicas de NET-SNMP.

Se modificará en el archivo de configuración cuatro cosas. La primera se trata del puerto que va a utilizar la herramienta SNMP, en este caso se trata del puerto UDP 161, adicionalmente se puede añadir puertos para IPv4 y para IPv6, pero en el proyecto no activaremos la opción de IPv6 porque no va será utilizada.

Otra función se trata de las comunidades las cuales son el único mecanismo de seguridad que tiene SNMP. Configuraremos las comunidades para que la comunidad public tenga acceso lectura desde cualquier dispositivo (remoto y local) y private tiene acceso de lectura y escritura.

```
###############
#  ACCESS CONTROL
                                  #  system + hrSystem groups only
view    all   included    .1.3.6.1.2.1.1
view    all   included    .1.3.6.1.2.1.25.1


rocommunity public 192.168.1.41

rocommunity public 192.168.4.27
                                   #  Full access from the local host
rocommunity public 192.168.110.30 #LAB

rocommunity public localhost

rwcommunity private localost

                              #  Default access to basic system info
rocommunity public default

rwcommunity private default

#  ACTIVE MONITORING
#

                                      #    send SNMPv1  traps
 trap2sink     localhost public
                                      #    send SNMPv2c traps
 trap2sink   default public

 trap2sink 192.168.1.34

 trap2sink 192.168.1.43

 trap2sink 192.168.1.33

 trap2sink 192.168.110.12
```

Figura 3.9. Fragmento Archivo snmpd.conf

Se debe habilitar el soporte de las traps para SNMP v2c. Se tiene que añadir las IPs de los gestores que se desea que reciban las TRAPS, en este caso están añadidas las IPs del entorno de desarrollo y posteriormente se añadirán las IPs del *Laboratorio de Gestion y Operación de Redes* en el cual se comprobara el proyecto.

Como última modificación se activará la opción para que sea un agente SNMP extensible atreves de subagentes AgentX. Para encender el demonio NET-SNMP simplemente se debe que reiniciar el servicio con sudo *snmpd restart* o activarle con el comando *sudo snmpd.*

4. Desarrollo de la Aplicación

4.1. Introducción

En este capítulo se visualiza el desarrollo del proyecto. Para ello se utiliza todo lo anteriormente comentado y después como se aplica en el proyecto.

En los capítulos previos se ha comentado que se implementara un entorno SNMP en la Raspberry incluyendo dos sensores de temperatura y humedad (DHT11 y DHT22) y un diodo LED. La Raspberry PI que actúa como agente y una serie de SCRIPTS manejan de forma automática todo el sistema.

El desarrollo del proyecto se realizará en varias partes claramente definidas. Las más importantes son acceso a los datos de los sensores, la administración y almacenamiento de los valores obtenidos con dichos sensores y el envío de los avisos (Traps) al gestor.

4.2. Esquema

En la Figura 4.1 se puede observar los distintos bloques de los que se compone el proyecto.

- *Scripts de control*. La comunicación entre los sensores de temperatura y humedad se realiza mediante unos drivers y scripts de lectura en PYTHON[14].
- *Valores Exportados*. Los valores exportados se obtienen mediante los scripts de lectura que al leer los valores de los sensores exportan dichos valores a los ficheros txt correspondientes, a cada objeto de la mib se le asignara un fichero distinto.
- *Demonios de control*. Como se mencionaba en el apartado 3.3.3, se utiliza la herramienta mib2c para obtener el código de los objetos de la mib que se gestionan. Los demonios de control son los ejecutables que se generan después de compilar el código obtenido mediante mib2c y las plantillas demonio y makefile que se obtienen de la página de SNMP. Se debe crear un demonio para cada objeto de la mib que se quiera gestionar.
- *Subagentes y Agente Maestro SNMP*. La conexión y comunicación entre el agente y el gestor es proporcionada por NET-SNMP al igual que para la conexión entre el agente maestro y los subagentes AgentX.

Una de las tareas del proyecto es el desarrollo de los scripts para la automatización de las funciones de los subagentes AgentX y del control de los sensores.

4.3. SubAgentes

Se implementará las funciones descritas y necesarias del subagante para el proyecto. La base de información de gestión del proyecto es la MIB (SENSOR-TEMPERATURA-MIB) que se ha mencionado anteriormente en la cual se incluyen los objetos gestionados. A continuación, se explican los distintos objetos gestionados en la MIB.

- *sensorPractica.* Se trata de un objeto escalar simple que se asocia a un diodo LED con el que se comprobara de forma visual el correcto funcionamiento del objeto.

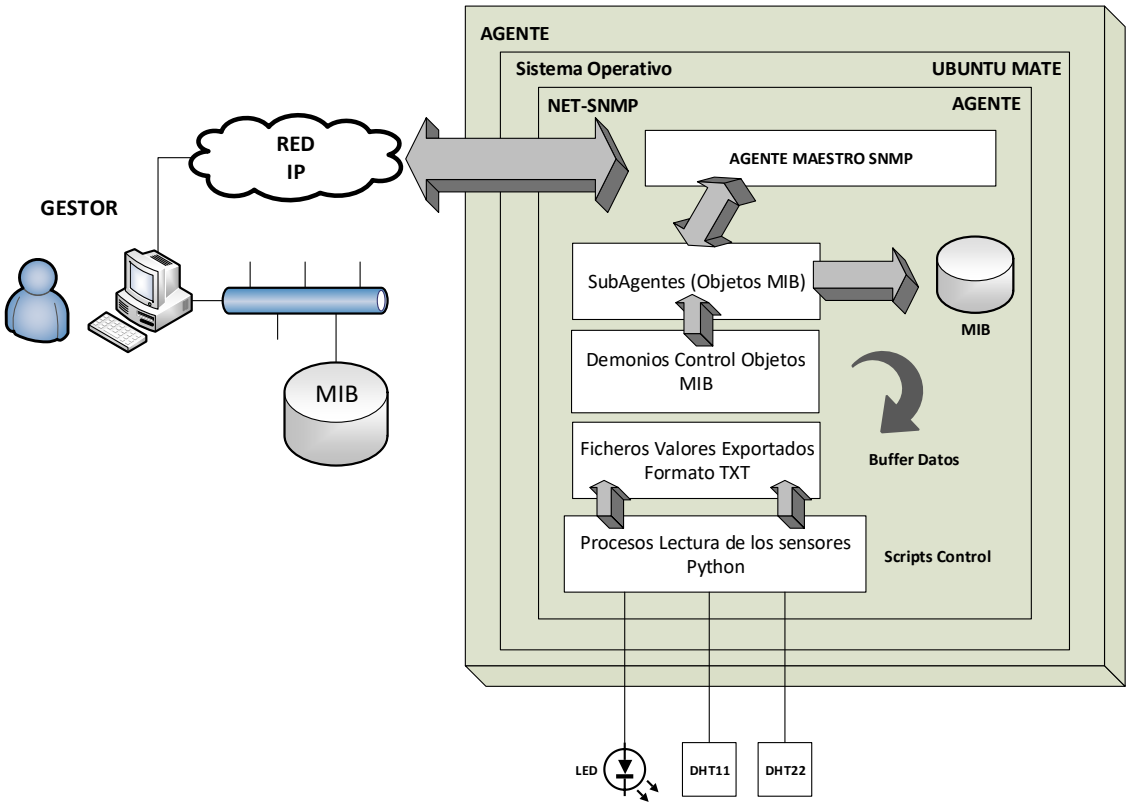


Figura 4.1. Esquema Proyecto

- *sensorTable.* Se trata de una tabla que tiene la información de los dos sensores conectados a la Raspberry Pi. Estos valores se actualizarán en tiempo real.

- *temperaturaDH11*. El escalar que representa la temperatura actualizada en tiempo real del sensor DHT11.

- *humedadDH11*. El escañar que representa la humedad actualizada en tiempo real del sensor DHT11.
- *temperaturaDH22*. El escalar que representa la temperatura actualizada en tiempo real del sensor DHT22.

- *humedadDHT22*. El escañar que representa la humedad actualizada en tiempo real del sensor DHT22.

- *trapTemperaturaDH11*. La Trap se emcarga de las notificaciones al gestor sobre la temperatura al llegar al nivel establecido.

- *trapHumedadDH11*. La Trap que se encarga de las notificaciones al gestor sobre la humedad al llegar al nivel establecido.

- *tabla_historico.* Tabla para el almacenamiento de valores históricos de los sensores DHT11 & DHT22

- *tabla_historicoControl.* Tabla para el almacenamiento de los valores de sondeo y máximos de los sensores DHT11&DHT22
-

El esquema de cómo debería ser la estructura del árbol de la MIB SENSOR-TEMPERATURA-MIB se muestra en la Figura 4.2 y el código de la MIB se encuentra en el anexo 15. Dicho árbol se ha introducido en la siguiente OID. 1.3.6.1.4.1.8072.2.333. Esta OID ha sido la elegida dado que netSmpExamples está localizado en OID: 1.3.6.1.4.1.8072.2 y añadiendo un nodo más comprobamos que da igual el número que se añada al OID, aunque sea muy grande, siempre que no esté ocupado previamente.

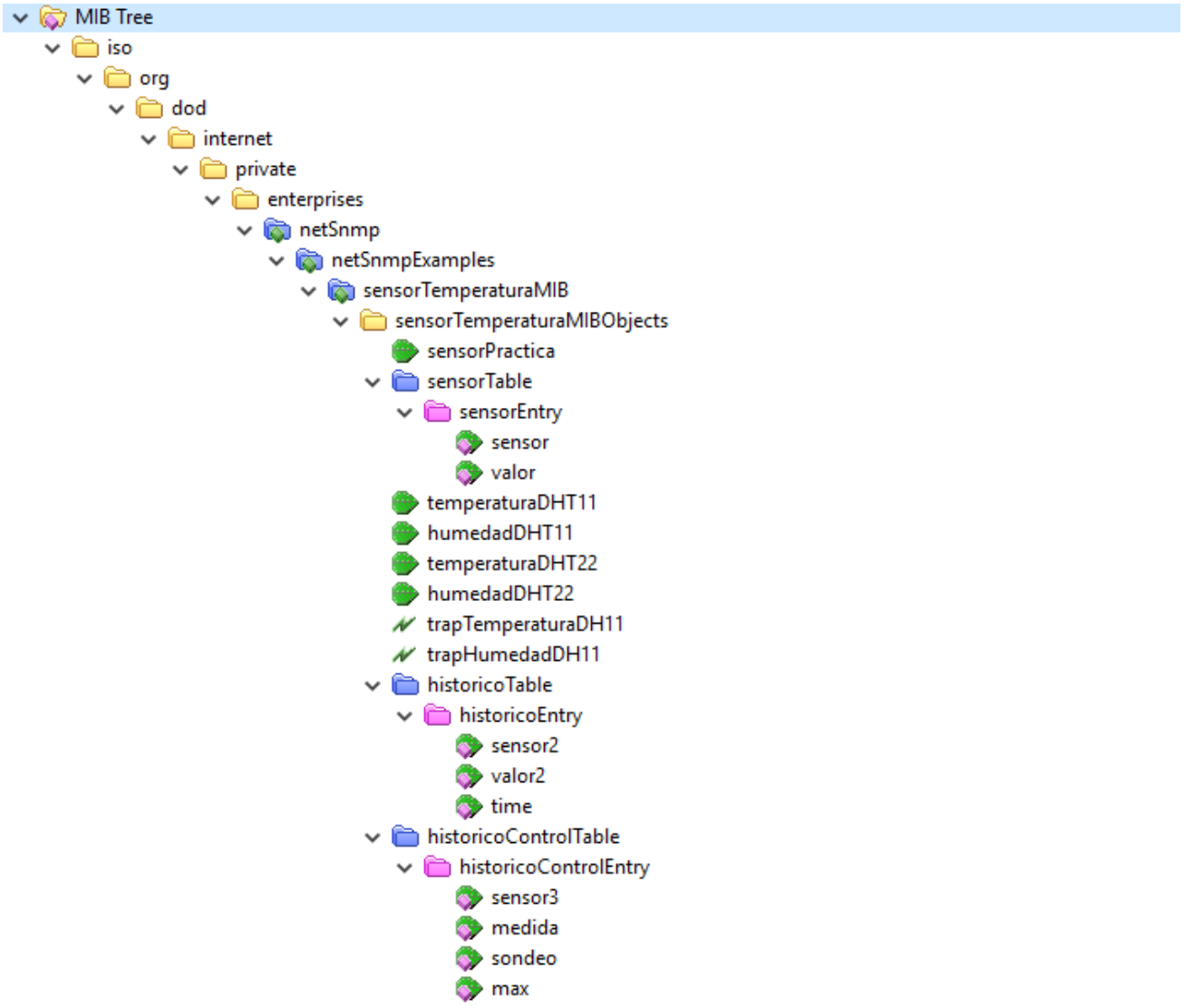


Figura 4.2. Árbol MIB Sensor-Temperatura-MiB

Tras tener la MIB codificada en formato ANS-1 se debe cargar en los dos directorios */usr/share/snmp/mibs* y */home/pi/.snmp/mibs* y una vez copiada esta MIB en los directorios hay que resetear el servicio *snmpd*. Para reiniciar el servicio snmpd hay dos formas de hacerlo, pero la que se utilizar que ya se ha mencionado alguna vez

```
sudo /etc/init.d/snmpd restart
```

Una vez realizado el reinicio del servidor ya se puede visualizar la estructura de nuestra MIB a través de los comandos disponibles por SNMP. En este caso para visualizar el árbol se usará el comando "snmptranslate". En la Figura 4.3 podemos ver la estructura.

```
snmptranslate -Tp -IR SENSOR-TEMPERATURA-MIB::sensorTemperaturaMIB
```

En el comando snmptranslate se utilizan las siguientes banderas para la correcta visualización.

- Tp se usa para que se dibuje el diagrama del árbol de la MIB
- IR para el acceso a la MIB a través del nombre de la misma. (SENSOR-TEMPERATURA-MIB::sensorTemperaturaMIB)

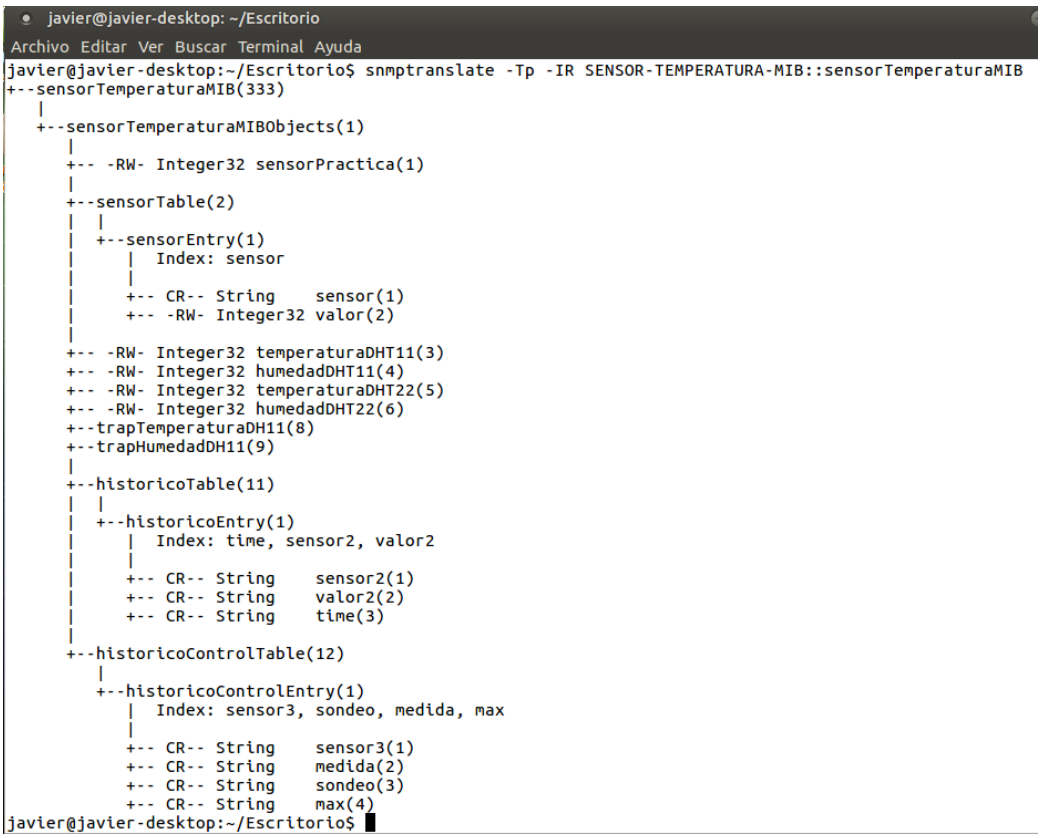


Figura 4.3. Estructura MIB Sensor-Temperatura-MIB usando Snmptranslate

4.4. Aplicaciones MIB

En este apartado vamos a explicar los distintos subagentes desarrollados para el proyecto. En la figura 4.4 se podrá visualizar un esquema de cómo se generarán las aplicaciones del proyecto.

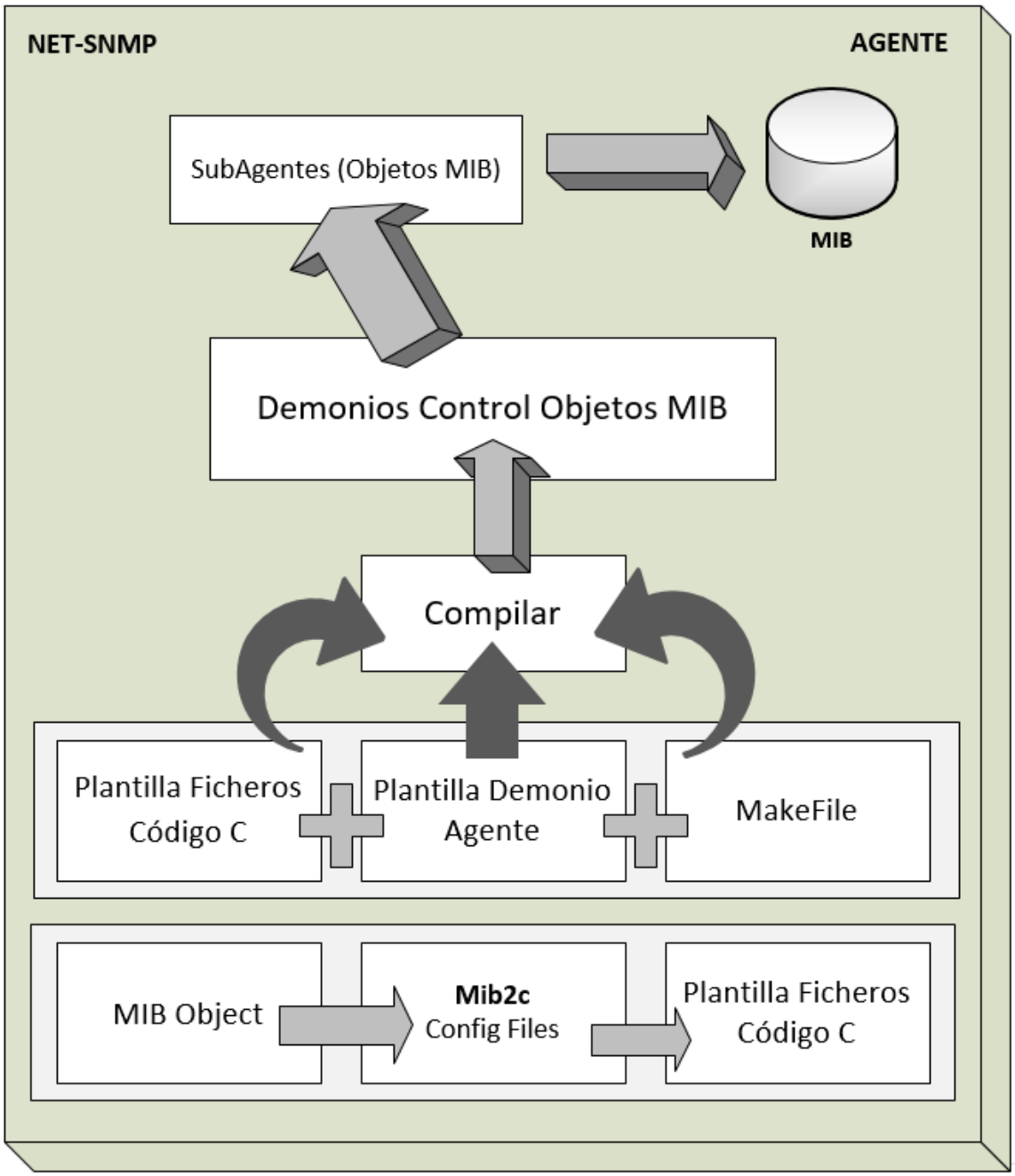


Figura 4.4 Esquema mib2c compilación

En la figura 4.4 se puede ver que primero tenemos un objeto de la MIB, puede ser un escalar, una tabla o una trap. Posteriormente mediante la herramienta mib2c de net-snmp se generan los ficheros de configuración de cada objeto de la mib, estos ficheros son dos plantillas en C que hay que modificar para cada mib o usuario dado que la herramienta genera unos ficheros genéricos y se deben completar para su correcto funcionamiento. Junto a las plantillas generadas por la herramienta mib2c se usarán dos plantillas más obtenidas de la web de net-snmp las cuales han de ser modificadas para cada objeto de la mib, una se tratará de la plantilla de demonio en la cual se indica la inicialización y la gestión de cada objeto y la otra se tratará de un makefile, a través de este fichero podremos compilar y generar los demonios de control de cada objeto de la mib. En la figura 4.5 se visualiza como es el makefile de

un objeto de la mib SENSOR-TEMPERATURA-MIB, para obtener el makefile de cada objeto de la mib simplemente se debe modificar la plantilla cambiando los nombres de los ficheros que se debe utilizar en cada objeto.

Por cada objeto que se tenga en la mib (escalar, tabla o trap) se deberá crear un subagente que controle el objeto gestionado.

De ahora en adelante aparecerá el concepto de aplicación que se trata simplemente del subagente que gestiona los objetos de la MIB.

```
CC=gcc

OBJS1=snmpdemoapp.o
OBJS2=tabla_historico_demon.o historicoTable.o
OBJS3=asyncapp.o
TARGETS=tabla_historico_demon snmpdemoapp asyncapp

CFLAGS=-I. `net-snmp-config --cflags`
BUILDLIBS=`net-snmp-config --libs`
BUILDAGENTLIBS=`net-snmp-config --agent-libs`

# shared library flags (assumes gcc)
DLFLAGS=-fPIC -shared

all: $(TARGETS)

snmpdemoapp: $(OBJS1)
    $(CC) -o snmpdemoapp $(OBJS1) $(BUILDLIBS)

asyncapp: $(OBJS3)
    $(CC) -o asyncapp $(OBJS3) $(BUILDLIBS)

tabla_historico_demon: $(OBJS2)
    $(CC) -o tabla_historico_demon $(OBJS2)  $(BUILDAGENTLIBS)

clean:
    rm $(OBJS2) $(OBJS2) $(TARGETS)

nstAgentPluginObject.so: nstAgentPluginObject.c Makefile
    $(CC) $(CFLAGS) $(DLFLAGS) -c -o nstAgentPluginObject.o nstAgentPluginObject.c
    $(CC) $(CFLAGS) $(DLFLAGS) -o nstAgentPluginObject.so nstAgentPluginObject.o
```

Figura 4.5 Fichero make file objeto MIB (tabla_historico)

4.4.1. Aplicación Integer (Sensores)

Se explica el desarrollo realizado para un objeto de la MIB, en este caso es un objeto escalar simple que se utiliza para el desarrollo de la práctica de la asignatura Gestión y Operación de Redes, para ellos se utiliza el objeto sensorPractica de la MIB del proyecto SENSOR-TEMEPERATURA-MIB. A continuación, se visualiza el objeto anteriormente nombrado sensorPractica en la figura 4.6.

```
sensorPractica OBJECT-TYPE
        SYNTAX          Integer32
        MAX-ACCESS      read-write
        STATUS          current
        DESCRIPTION
        "Valor para modificar en la práctica"
        ::= { sensorTemperaturaMIBObjects 1}
```

Figura 4.6 Fragmento SENSOR-TEMPERATURA-MIB. Objeto sensorPractica

Se analizan los diferentes apartados que tiene este objeto tan sencillo. Lo primero es ver qué tipo de objeto gestionado es, en este caso se trata de un escalar, es decir, un valor entero. Después se puede ver el tipo de acceso que tiene este objeto, puede ser leído y escrito y una pequeña descripción que es opcional la cual ayudara a la comprensión de cuál es la función que realiza ese objeto.

Y la parte posiblemente más importante seguramente del objeto es la última parte que podemos ver, que se trata de donde está localizado el objeto en el árbol de la MIB, en este caso se puede observar que el objeto precedente al anteriormente nombrado se trata de SensorTemperaturaMIBObjects. En el anexo A.1 se podrá ver todo el Árbol de la MIB.

La segunda parte del proceso para poder gestionar el OBJETO de la MIB necesita un demonio dado que si no se tiene lo que ocurrirá será que se tendrá un objeto con el que no se podrá interactuar. Para esto vamos a utilizar la herramienta anteriormente mencionada, mib2c. Esta herramienta lo que hará será generar un esqueleto en C (plantilla) la cual se deberá completar para poder trabajar con el objeto de la MIB. Para ello se utilizará la configuración que tiene por defecto mib2c que en este caso para el objeto escalar se trata del comando mib2c.int_watch.conf[15].

```
sudo mib2c -c mib2c.int_watch.conf SENSOR-TEMPERATURA-
MIB::sensorPractica
```

Este comando lo que hace es generar dos archivos, en concreto se trataría de los archivos sensorPractica.c y sensorPractica.h. Estos dos archivos son la base para el uso del objeto, pero es necesario dos cosas más, la primera se trata de un código C que tenga las funciones de demonio y otra un makefile que compile todo en común. El código del demonio, la plantilla es proporcionada por NET-SNMP así que simplemente se ha tenido que ir al proyecto de ejemplo y obtenerla. El proceso de completar y adaptar la plantilla del demonio al proyecto es una de las tareas básica que se debe realizar.

```
int agentx_subagent=1
If(agentx_subagent)

{
netsnmp_ds_set_boolean(NETSNMP_DS_APPLICATION_ID,

NETSNMP_DS_AGENT_ROLE, 1);

}
init_agent("sensorPractica_demon");

init sensorPractica();
```

Figura 4.7 Código inicialización subagente sensorPractica_demon

Se puede ver los cuatro grandes apartados en la Figura 4.7.

El primero *int agentx_subagent=1*, inicializándolo con el valor a 1 produce que actue como subagente, en el caso que se quisiera que se comportara como un agente maestro se debería cambiar en este apartado.

El segundo *netsnmp_ds_set_boolean(netsnmp_ds_application_id ,netsnmp_ds_agent_role, 1)* lo que realiza es el proceso de inicialización de los subagentes como clientes de AgentX.

Los dos últimos *`init agent("sensorPractica demon") y init sensorPractica();`* inicializan el agente SNMP y la variable del objeto que gestiona, en el caso del ejemplo se trata de sensorPractica.

El objeto que se gestiona es sensorPractica, y se observa dentro del código (*sensorPractica.c*) los OID que están incluidos. La variable que aparece en *sensorPractica.c* es *"oid_sensorPractica_oid[]={ 1,3,6,1,4,1,8072,2,333,1,1 };"* . Se puede comprobar en el anexo A.1 en los cuales se encuentra el árbol entero de la MIB que coincide con esa OID.

Una vez se obtiene los ficheros generados por mib2c y el demonio (sensorPractica_demon) ya modificado y listo para compilar faltaría el ultimo archivo que relaciona todo en común, se trata del makefile el cual se ha obtenido de los tutoriales de NET-SNMP. Una vez se tiene los cuatro ficheros necesarios para la compilación se precede a compilar todo en común mediante el makefile.

```
sudo make sensorPractica_demon
```

El comando siguiente generara el demonio que controla la aplicación el cual es un ejecutable que llamaremos demonio o demon. En la figura 4.8 visualiza el esquema de la aplicación creada.

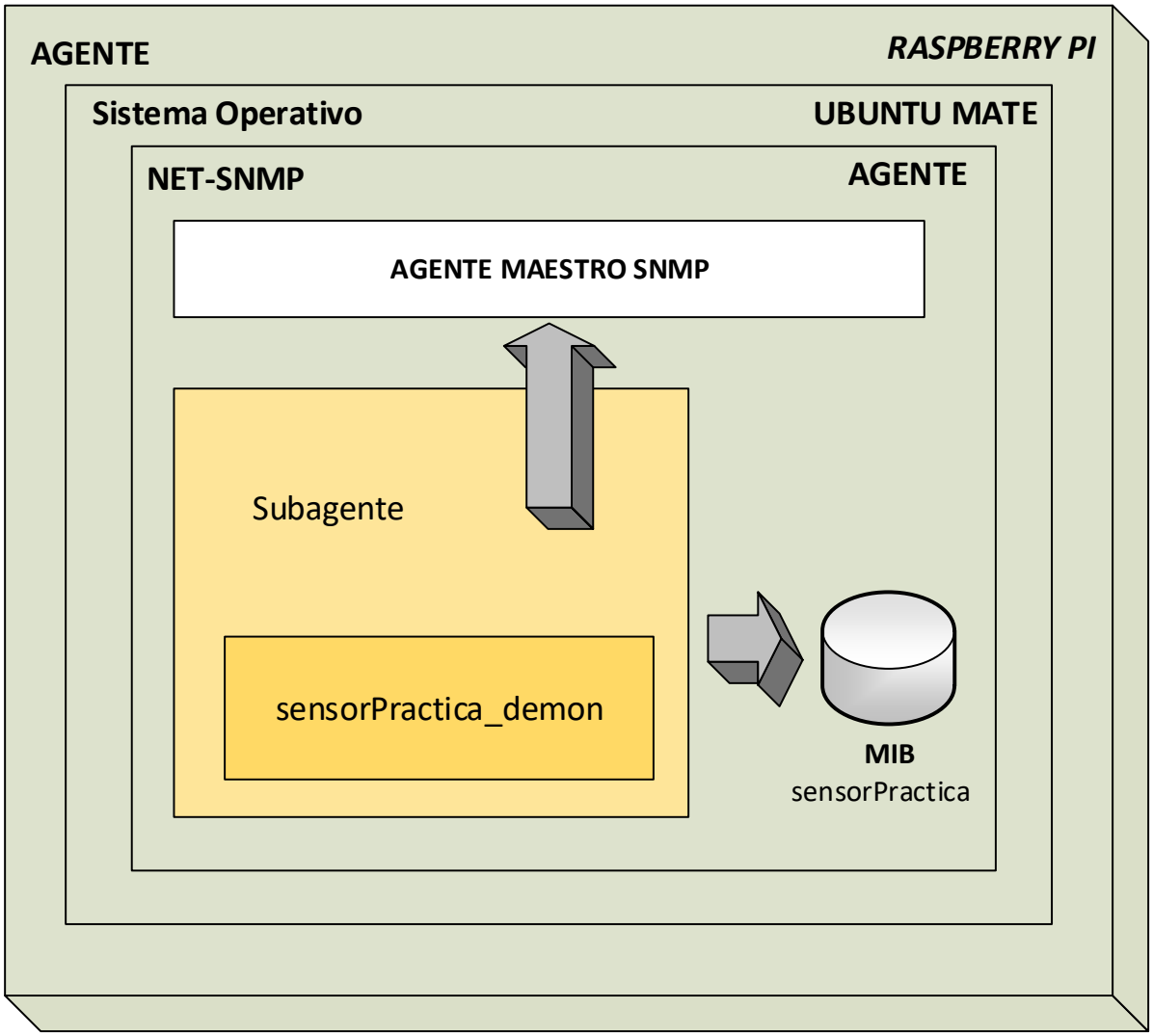


Figura 4.8. Esquema Aplicación sensorPractica

A este objeto se le asociara a un LED para la visualización y comprobación del valor introducido en la mib. En la Figura 4.9 se visualiza como se debe conectar el LED a la Raspberry PI. Se trata de un esquema de cómo se debe conectar el LED a la raspberry PI, en el proyecto se conecta a unos GPIO distintos dado que están ocupados por los sensores DHT11 y DHT22.

Al igual que el objeto previamente explicado se crean cuatro aplicaciones las cuales seguramente sean las más importantes del proyecto dado que se tratan de las aplicaciones que soportan los objetos principales sobre los que se basa el sistema desarrollado. Estas aplicaciones son:

- *temperaturaDH11_demon*
- *humedadDH11_demon*
- *temperaturaDHT22_demon*

- *humedadDHT22_demon*

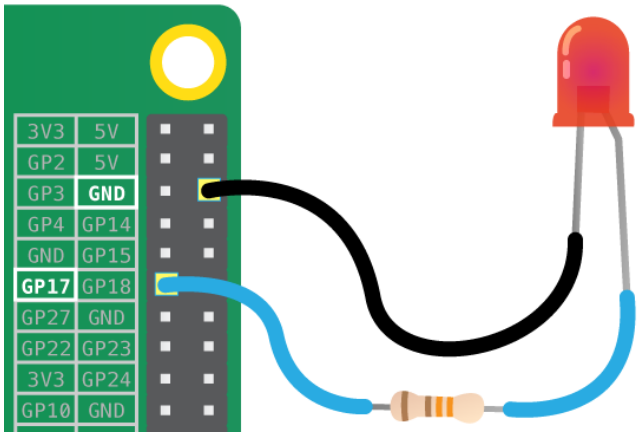


Figura 4.9 Esquema LED Raspberry Pi

Los objetos mencionados obtienen los valores de archivos de texto que actúan como unos buffers de datos, esto quiere decir que en el momento en el que se actualizan el valor de los datos en dichos archivos estos se actualizarán de forma automática e instantánea en el objeto gestionado.

Dichos archivos de texto en los que se encuentran los valores han sido obtenidos mediante los sensores conectados a los GPIO de la Raspberry PI En la Figura 4.10 se observa como es el sensor DHT22 y en la Figura 4.11 sus conexiones. Posteriormente se explicará cómo se realiza el acceso a la lectura y exportación de los valores obtenidos por los sensores. Una vez exportados estos valores a los ficheros de texto la aplicación lee y almacena en la MIB los valores actualizándolos de forma automática e instantánea.

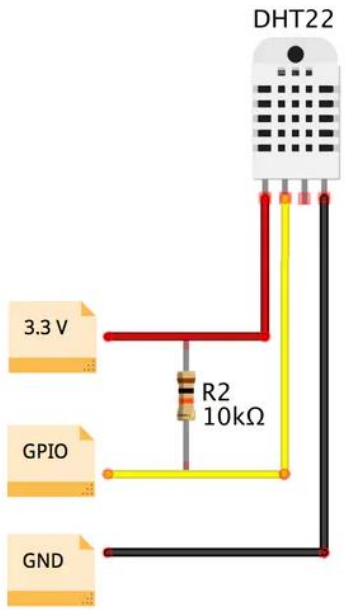


Figura 4.10 Esquema Sensor DHT

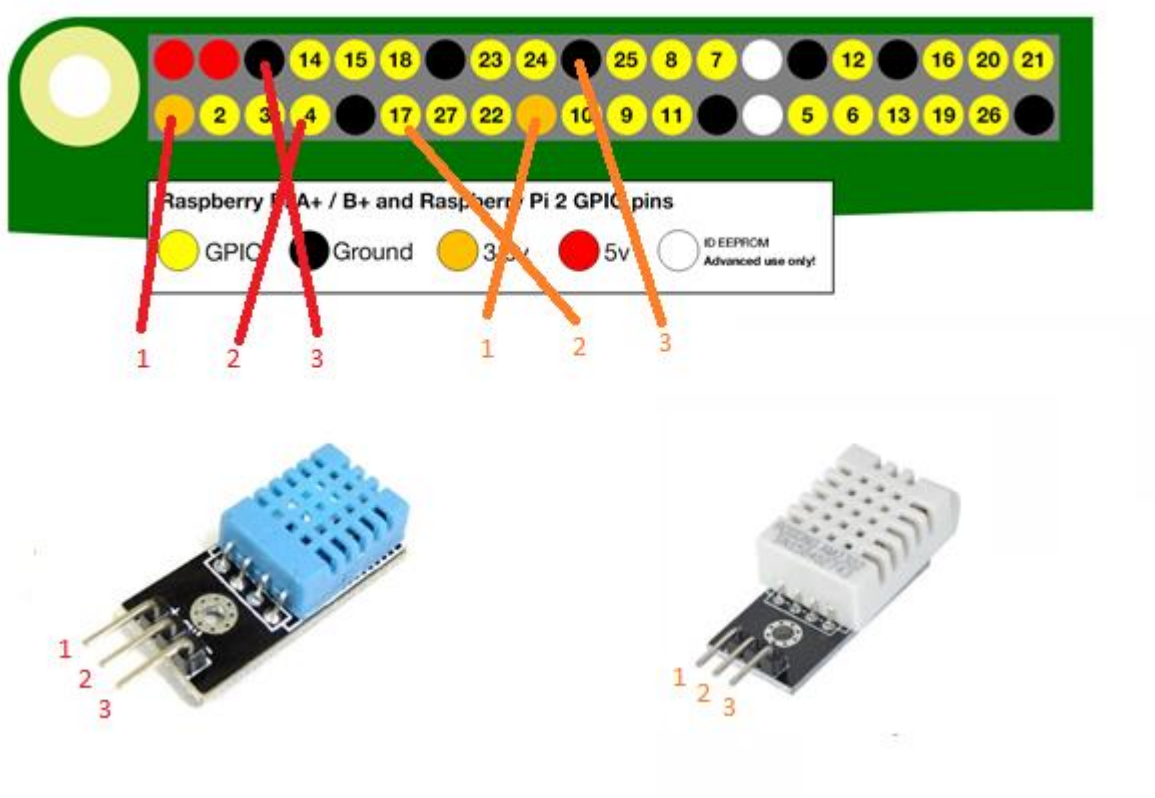


Figura 4.11 Conexión GPIO Raspberry Sensores

Hay que mencionar que este es el esquema del sensor DHT22 simple, los sensores comprados y usados en el proyecto tienen una placa integrada por lo que no hará falta tener ninguna resistencia porque está incluida en esa placa, Figura 3.4 y 3.5.

Se debe usar la herramienta proporcionada por SNMP, la herramienta mencionada es mib2c con la que generaremos el esqueleto del código. El tipo de plantilla que utiliza, en este caso es mib2c.int_watch.conf. El código necesario se obtiene a través de los comandos

```
mib2c -c mib2c.int_watch.conf SENSOR-TEMEPRATURA-MIB::temperaturaDH11
mib2c -c mib2c.int_watch.conf SENSOR-TEMEPRATURA-MIB::humedadDH11
mib2c -c mib2c.int_watch.conf SENSOR-TEMEPRATURA-MIB::temperaturaDHT22
mib2c -c mib2c.int_watch.conf SENSOR-TEMEPRATURA-MIB::humedadDHT22
```

Con estos comandos se obtienen los códigos en C *(temperaturaDHT11.c, temperaturaDHT11.h, humedadDHT11.c, humedadDHT11.h, temperaturaDHT22.c, temperaturaDHT22.h, humedadDHT22.c y humedadDHT22.h)* para compilar los programas previamente mencionados. Junto a los ficheros generados por mib2c se debe modificar las platillas de cada demonio *(temperaturaDHT11_demon, humedadDHT11_demon, temperaturaDHT22_demon,*

humedadDHT22_demon). Una vez compilados con el makefile de cada objeto se obtienen los siguientes programas:

- *temperaturaDHT11_demon*
- *humedadDHT11_demon*
- *temperaturaDHT22_demon*
- *humedadDHT22_demon*

Estos programas tienen archivos asociados con los valores de los sensores, estos archivos con los valores de los sensores son los que funcionan como buffer de datos. En la figura 4.12 se visualiza un fragmento del código del demonio donde se realiza la modificación para la lectura de los archivos de texto. El esquema de la aplicación se observa en la figura 4.13 y los archivos asociados a cada aplicación se relacionan a continuación

- *temperaturaDHT11_demon -> sensorValor1.txt*
- *humedadDHT11_demon -> sensorValor2.txt*
- *temperaturaDHT22_demon -> sensorValor4.txt*
- *humedadDHT22_demon -> sensorValor3.txt*

```
75  /* your main loop here... */
76  while(keep_running) {
77
78    /*Lee de un .txt y extrae el valor de la temp*/ /*NUEVO*/
79    FILE * fp;
80
81    fp = fopen ("sensorValor1.txt", "r");
82    /*Para el ciclo de lectura en el futuro*/
83
84    if (fscanf(fp, "%d", &temperaturaDHT11)==1){
85        printf("%d",temperaturaDHT11);
86        }else{
87        printf("Error \n");
88    }
89
90
91    fclose(fp);
92    /* if you use select(), see snmp_select_info() in snmp_api(3) */
93    /*     --- OR ---  */
94    agent_check_and_process(1); /* 0 == don't block */
95  }
96
```

Figura 4.12 Fragmento temperaturaDHT11_demon Lectura Archivo TXT

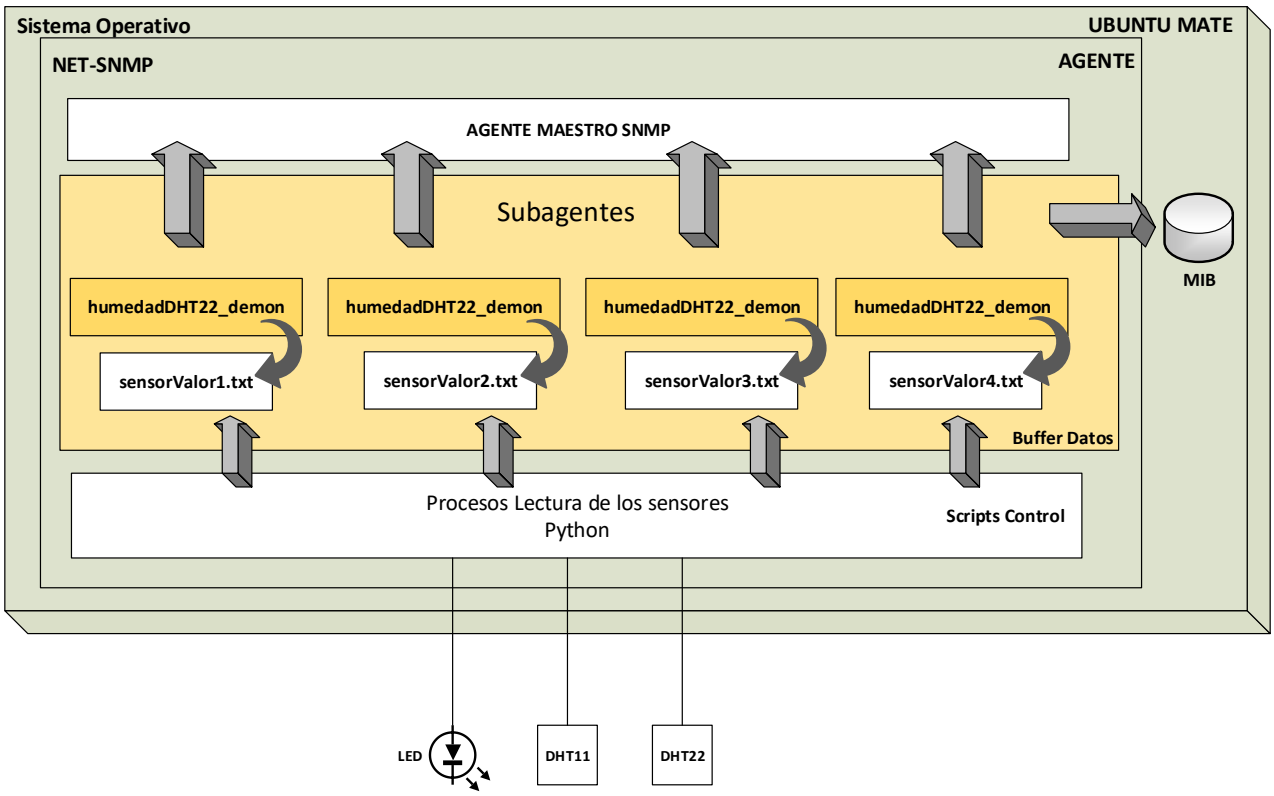


Figura 4.13 Esquema Aplicación Integers Sensores

Los valores de estos archivos están exportados del programa en PYTHON de los sensores, hay que mencionar que la temperaturaDHT11 y la humedadDHT11 de estos sensores obtenemos valores naturales que es lo que se puede almacenar en SNMP y el problema encontrado en el desarrollo del proyecto ha sido la intención de almacenar valores enteros. NET-SNMP desarrollando los objetos desde la herramienta mib2c después de muchos intentos sin éxitos se consiguió encontrar documentación que informaba que actualmente no era posible introducir valores enteros en nuestra MIB, por lo que se debe modificar los archivos de salida del código en PYTHON para convertir dichos valores enteros en naturales, esto lo se ha realizado mediante un redondeo para así poder administrar y almacenarlos en nuestra MIB. En la Figura 4.14 se visualiza como se realiza el acceso a la librería de PYTHON de los sensores DHT11 y DHT22 y como se exporta la salida a la variable RESULTS. Desde esa variable se realiza la exportación y redondeo de los valores.

```
RESULTS="`python
~/Escritorio/proyecto2/Adafruit_Python_DHT/examples/AdafruitDHT.py 11 4
| grep Temp `"

RESULTS2="`python
~/Escritorio/proyecto2/Adafruit_Python_DHT/examples/AdafruitDHT.py 22 17
| grep Temp `"
```

Figura 4.14 Acceso datos sensores PYTHON

A continuación, se visualiza en la Figura 4.15 y 4.16 el código por el cual se exporta los valores desde la librería de PYTHON con el código previamente utilizado a los valores TXT y cómo se realiza el redondeo de los valores del sensor DHT22 en la Figura 4.17.

```
#Almacena los valores para los Integers DHT11
echo "${TEMP}" > /home/javier/Escritorio/proyecto/Demonios/TemperaturaDHT11/sensorValor1.txt;
echo "${HUM}" > /home/javier/Escritorio/proyecto/Demonios/HumedadDHT11/sensorValor2.txt;
```

Figura 4.15. Exportación Valores Sensor DHT11 a archivos TXT

```
#Almacena los valores para los Integers DHT22
echo "${a}" > /home/javier/Escritorio/proyecto/Demonios/TemperaturaDHT22/sensorValor4.txt
echo "${b}" > /home/javier/Escritorio/proyecto/Demonios/HumedadDHT22/sensorValor3.txt
```

Figura 4.16. Exportación Valores Sensor DHT22 a archivos TXT

```
#Almacena los valores para la tabla del sensor DHT22
echo "${TEMP2}" > /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor44.txt;
echo "${HUM2}" > /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor33.txt;
read a < /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor44.txt
read b < /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor33.txt
echo "($a+0.5)/1" | bc > /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor444.txt;
echo "($b+0.5)/1" | bc > /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor333.txt;
read a < /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor444.txt
read b < /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor333.txt
echo "TemperaturaDHT22 ${a}" >> /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorLectura.txt;
echo "HumedadDHT22 ${b}" >> /home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorLectura.txt;
```

Figura 4.17. Redondeo de los valores del sensor DHT22

4.4.2. Aplicación TABLA

Usando el comando SNMP snmptranslate podemos ver la estructura de la tabla sensorTable en la mib SENSOR-TEMPERATURA-MIB. En la Figura 4.18 podemos observar la tabla cuando se introduce el siguiente comando.

```
snmptranslate -Tp -IR SENSOR-TEMPERATURA-MIB::sensorTable
```

```
javier@javier-desktop:~$ snmptranslate -Tp -IR SENSOR-TEMPERATURA-MIB::sensorTable
+--sensorTable(2)
   |
   +--sensorEntry(1)
      |  Index: sensor
      |
      +-- CR-- String    sensor(1)
      +-- -RW- String    valor(2)
javier@javier-desktop:~$
```

Figura 4.18. Esquema Tabla SENSOR-TEMPERATURA-MIB usando SNMPTRANSLATE

Se debe usar la herramienta proporcionada por SNMP, la herramienta mencionada es mib2c con la que generaremos el esqueleto del código. El archivo de configuración que se ha usado en este caso es

"mib2c.table_data.conf". Esta configuración hace que los datos sean obtenidos a través de una cache interna y que NET-SNMP haga de puente entre esta cache y nuestro archivo externo en el cual están almacenados los valores de la tabla. Para generar los archivos de la tabla se usará el siguiente comando.

```
mib2c -c mib2c.table_data.conf SENSOR-TEMPERATURA-MIB::sensorTable
```

Al introducir el comando el propio Sistema Operativo preguntará sobre la configuración de la tabla para así poder completar el esqueleto de como la tabla va a obtener los valores, si será de una ubicación conocida o desconocida. Como se ha mencionado anteriormente en este caso se tratará de una ubicación conocida. Este proceso generará dos archivos, los cuales se llaman sensorTable.c y sensorTable.h y los otros dos ficheros restantes, tabla_sensor_demon y makefile, serán completados mediante la plantilla proporcionada por SNMP por el desarrollador. En la Figura 4.19 se puede visualizar el fragmento de código en el que la tabla obtiene los valores de un archivo txt (sensorLectura.txt) y los introduce en la tabla de la MIB. El resto del código no se explicará dado que ha sido generado automáticamente por la herramienta mib2c.

En el punto 2.4.1 se veía el funcionamiento de las tablas en SNMP. El objeto gestionado (sensorTable) contiene dos columnas en las cuales la primera indica cómo se llama el sensor y en la segunda indica el valor medido por dicho sensor.

Este proceso es el encargado de almacenar y actualizar los valores de la tabla a través del archivo de texto *sensorLectura.txt*. En la Figura 4.20 se puede visualizar un esquema del funcionamiento de la tabla mencionada, se muestra como es la importación de los datos externos de los sensores a través del subagente hacia la mib.

```
    int sensorTable_load( netsnmp_cache *cache, void *vmagic ) {
    netsnmp_tdata     *table = (netsnmp_tdata *)vmagic;
    netsnmp_tdata_row *row;
    struct sensorTable_entry *this;
    FILE *fp;
    char buf[128];
    int i=0;
    char sensor_ind[32];
    int valor_temp;

    fp = fopen( "sensorLectura.txt", "r" );
    if ( !fp ) {
        return -1;
    }
    while ( fgets( buf, 128 , fp )) {
            sscanf(buf, "%s %d", sensor_ind, &valor_temp);
            row = sensorTable_createEntry(table,sensor_ind,
            strlen(sensor_ind));
            this=row->data;
            this->valor=valor_temp;
    }
    fclose(fp);
    return 0;
}
```

Figura 4.19. Fragmento sensorTable.c

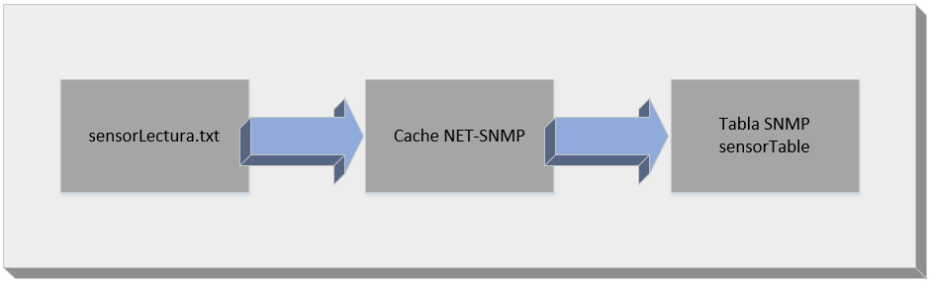


Figura 4.20. Esquema Buffer Tabla Datos

Como en el apartado anterior una vez generados los ficheros sensorTable.c y sensorTable.h con la herramienta mib2c y completando las plantillas de los demonios (tabla_sensor_demon) y el makefile se podrá compilar la aplicación.

Para compilar esta aplicación se usará el comando *"sudo make tabla_sensor_demon"*. Con esto se obtiene el ejecutable *"tabla_sensor_demon"* que será lo que debemos ejecutar para que la tabla pueda ser gestionada.

Compilaremos el demonio que dará el soporte para el AgentX como en los apartados anteriores con el comando. En la figura 4.21 se visualiza la aplicación tabla creada.

```
sudo make table_sensor_demon
```

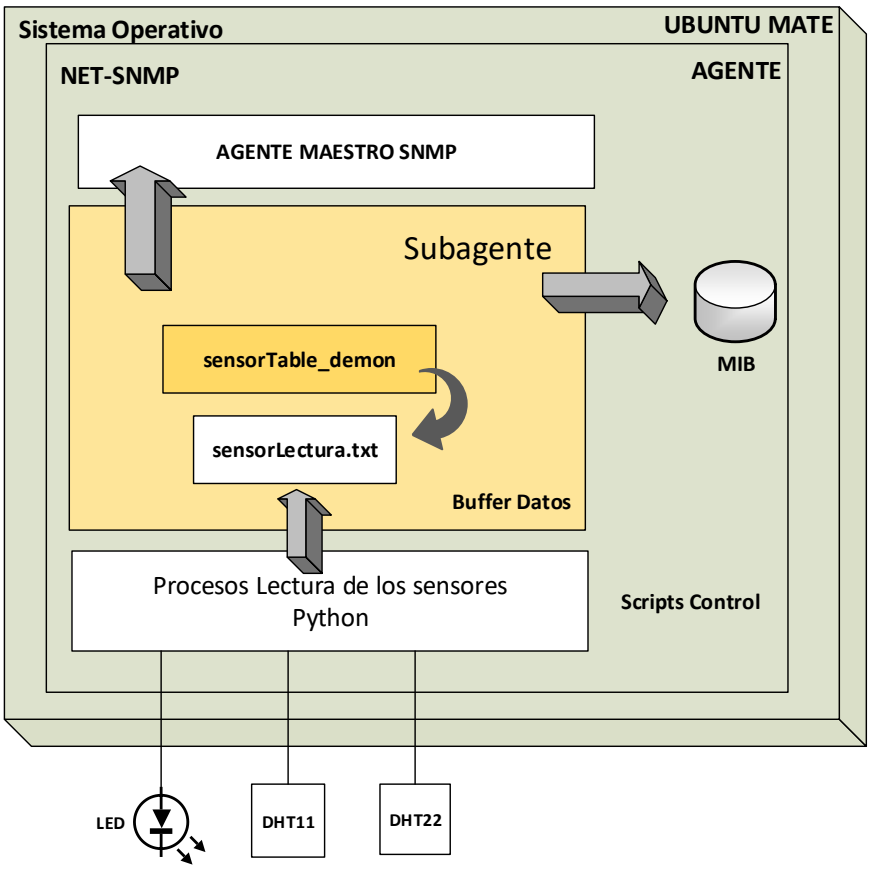


Figura 4.21 Esquema Aplicación Tabla

Para iniciar el demonio debemos ejecutar el comando

```
sudo ./table_sensor_demon
```

Al igual que la aplicación Tabla, la aplicación Historico está formado por dos tablas que se complementan entre sí.

Como en las anteriores aplicaciones se debe usar la herramienta mib2c para generar el esqueleto del código. El archivo de configuración que se ha usado es "mib2c.table_data.conf". Para generar los archivos de la tabla se usará el siguiente comando.

```
mib2c -c mib2c.table_data.conf SENSOR-TEMPERATURA-
MIB::HistoricoTable
```

```
mib2c -c mib2c.table_data.conf SENSOR-TEMPERATURA-
MIB::HistoricoControlTable
```

Este proceso generará dos archivos por cada objeto, los cuales se llaman sensorHistoricoTable.c , sesorHistoricoTable.h, sensorHistoricoControlTable.c y sensorHistoricoControlTable.h y los otros cuatro ficheros restantes, dos demons (tabla_historico_demon , tabla_historicoControl_demon) y dos makefiles que serán completados mediante la plantilla proporcionada por SNMP por el desarrollador.

En la Figura 4.22 se puede visualizar el fragmento de código generado por mib2c (historicoTable.c) en el que la tabla obtiene los valores de un archivo txt (histórico.txt) y los introduce en la tabla de la MIB. El resto del código no se explicará dado que ha sido generado automáticamente por la herramienta mib2c. Como nota adicional hay que mencionar que se han introducido todas las columnas como índice en las tablas para mayor facilidad en la compilación del código generado por la herramienta mib2c de SNMP.

La estructura del código para la tabla de control será el mismo que el de la figura 4.19 lo único que variaría serán las variables correspondientes a esa tabla.

```
historicoTable_load( netsnmp_cache *cache, void *vmagic ) {
netsnmp_tdata     *table = (netsnmp_tdata *)vmagic;
netsnmp_tdata_row *row;
struct historicoTable_entry *this;
FILE *fp;
char buf[128];
char time_ind[32];
char sensor2_ind[32];
char valor2_ind[32];

fp = fopen( "historico.txt", "r" );
```

```
if ( !fp ) {
    return -1;
}
while ( fgets( buf, 128, fp )) {
sscanf(buf, "%s %s %s",sensor2_ind,valor2_ind,time_ind);
row = historicoTable_createEntry(table , time_ind, strlen
(time_ind),sensor2_ind, strlen(sensor2_ind) , valor2_ind ,
strlen(valor2_ind));
 }
 fclose(fp);
 return 0;
```

Figura 4.22 Fragmento configuración tablaHistorico

En la figura 4.23 se observa el esquema de la aplicación tabla histórico e histórico control.

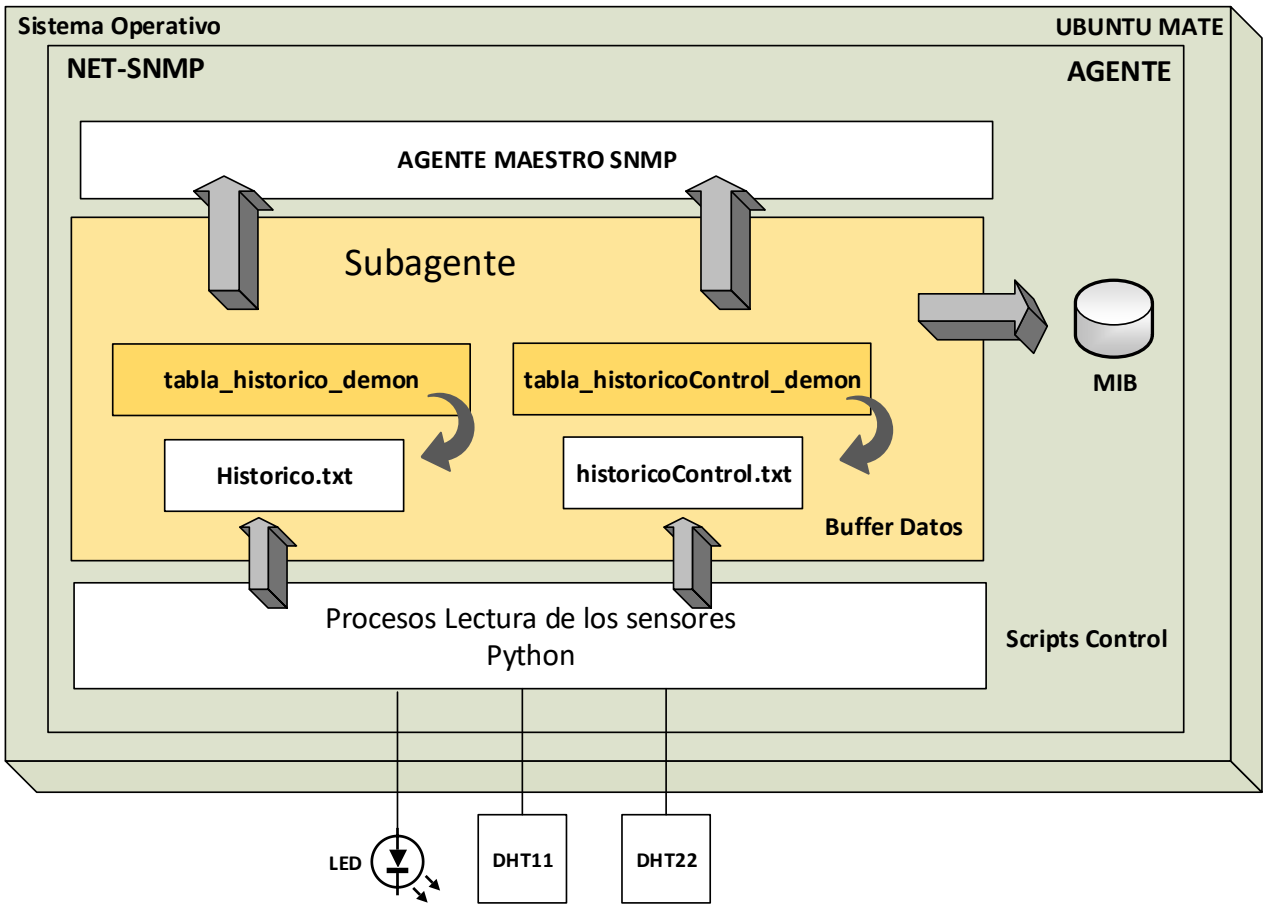


Figura 4.23 Esquema Aplicación Tabla Histórico e Histórico Control

4.4.3. Aplicación Traps

Estas aplicaciones son el complemento de los objetos temperaturaDHT11 y humedadDHT11. Se tratan de dos programas que soportan los objetos de tipo notificación, trap_temperaturaDHT11 y trap_humedadDHT11 y como se ha mencionado que son el complemento de los objetos vistos previamente, esto se debe a que el valor temperaturaDHT11 está asociado a una trap soportada por la aplicación trap_TemperaturaDHT11, de modo que cuando el sensor llegue a una temperatura que se determine mande un aviso al gestor. El valor humedadDHT11 esta soportado por trap_HumedadDHT11, el cual realizara la misma función que el anterior, cuando llegue a un valor marcado previamente por el desarrollador mandara el aviso al gestor. Se explicará el funcionamiento de las traps, se verán los fragmentos importantes de dichos objetos en la Figura 4.24 en la cual se puede observar el fragmento de la MIB con los OID correspondientes a las Traps. Los valores de OID de las traps serán en este caso dos, trapTemperaturaDH11 con una OID (.1.3.6.1.4.1.8072.2.333.1.3.8) y trapHumedadDH11 con OID (.1.3.6.1.4.1.8072.2.333.1.3.9).

El subagente será el encargado de gestionar y monitorizar el valor de la variable del Objeto SNMP sobre el que se quiere enviar avisos. En este caso se monitoriza las variables de temperatura y humedad del sensor DHT11.

```
-- TRAP TEMPERATURA: Trap asociado a la temperatura que queremos para
que mande el aviso de la temperatura.
trapTemperaturaDH11 NOTIFICATION-TYPE
    STATUS  current
    OBJECTS {temperaturaDH11} -- cambiar al nombre del valor
    DESCRIPTION
            "Trap que se envia cuando el sensor lee los valores de la
temperatura y supera el valor que pongamos como limite en el codigo"
    ::= { sensorTemperaturaMIBObjects 8}

-- TRAP HUMEDAD: Trap asociado a la humedad que queremos para que mande
el aviso de la humedad
trapHumedadDH11 NOTIFICATION-TYPE
    STATUS  current
    OBJECTS {humedadDH11} -- modificar
    DESCRIPTION
            "Trap que se envia cuando el sensor lee los valores de la
humedad y supera el valor que pongamos como limite en el codigo"
    ::= { sensorTemperaturaMIBObjects 9}
```

Figura 4.24. Fragmento Traps SENSOR-TEMPERATURA-MIB

sudo mib2c -c mib2c.notify.conf SENSOR-TEMPERATURA-MIB::trapTemperaturaDH11

sudo mib2c -c mib2c.notify.conf SENSOR-TEMPERATURA-MIB::trapHumedadDH11

Este comando genera los dos archivos esqueleto por cada comando, los de la temperatura serán trapTemperaturaDH11.c y trapTemperaturaDH11.h y los de la humedad serán trapHumedadDH11.c y trapHumedadDH11.h. El propio código esqueleto generado automáticamente da soporte para el envío de las traps desde el subagente AgentX al agente Maestro. Como en apartados anteriores se debe desarrollar el demonio () y el make para la compilación y funcionamiento del objeto que se quiere gestionar.

En la Figura 4.25 se visualizará un extracto del código generado por mib2c. Se trata del código esqueleto que se ha generado automáticamente. En el código se puede visualizar las dos OIDs importantes, la OID de la trap y la OID del objeto gestionado. En la Figura 4.26 se ve parte del código generado automáticamente (demon) donde se indica cual es el valor máximo a partir del cual el agente envía las traps al gestor. El problema encontrado es que este valor está incluido dentro del demonio que gestiona y administra el objeto, por lo cual está dentro del programa cuando se compila, así que para poder modificar este valor no solo se debería modificar el archivo, sino que se debería volver a compilar todo el objeto. Para compilar el objeto se hará como en apartados anteriores. En las Figuras 4.27 y 4.28 se observará lo mismo que en las figuras anteriormente nombradas, pero para la humedad.

```
int
send_trapTemperaturaDHT11_trap( void )
{
    netsnmp_variable_list  *var_list = NULL;
    const oid trapTemperaturaDHT11_oid[] = { 1,3,6,1,4,1,8072,2,333,1,8 };
    const oid temperaturaDHT11_oid[] = { 1,3,6,1,4,1,8072,2,333,1,3, 0 };
    int temp=0;

    /*Lee de un .txt y extrae el valor de la Temperatura*/
    FILE * fp;

    fp = fopen ("sensorValor1.txt", "r");

    if (fscanf(fp, "%d", &temp)==1){
        printf("%d",temp);
        }else{
        printf("Error al leer la sensorValor1.\n");
    }

snmp_varlist_add_variable(&var_list,
        temperaturaDH11_oid, OID_LENGTH(temperaturaDH11_oid),
        ASN_INTEGER,
        /* Set an appropriate value for temperaturaDH11 */
        &temp, sizeof(temp));
```

Figura 4.25. Lectura Valor Sensor Temperatura

```
 /*Envia un trap SNMP v2c cuando el valor de potencia total es superior a X
grados*/
        if(temp>24){
          send_trapTemperaturaDHT11_trap(); /*Envia un trap cada 5 segundos*/
          sleep(5);
        }
```

Figura 4.26. Configuración valor Máximo Temperatura y tiempo envío Traps (demon)

```
int
send_trapHumedadDHT11_trap( void )
{
    netsnmp_variable_list  *var_list = NULL;
    const oid trapHumedadDHT11_oid[] = { 1,3,6,1,4,1,8072,2,333,1,9 };
    const oid humedadDHT11_oid[] = { 1,3,6,1,4,1,8072,2,333,1,4, 0 };
int hum=0;
    /*Lee de un .txt y extrae el valor de la ptotal*/ /*MIO*/
    FILE * fp;

    fp = fopen ("sensorValor2.txt", "r");

    if (fscanf(fp, "%d", &hum)==1){
        printf("%d",hum);
        }else{
        printf("Error al leer la sensorValor2.\n");
    }

snmp_varlist_add_variable(&var_list,
        temperaturaDH11_oid, OID_LENGTH(temperaturaDH11_oid),
        ASN_INTEGER,
        /* Set an appropriate value for temperaturaDH11 */
        &hum, sizeof(hum));
```

Figura 4.27. Lectura Valor Sensor Humedad

```
        /*Valor de humedad a elegir*/
        if(hum>57){
          send_trapHumedadDHT11_trap(); /*Envia un trap cada 5 segundos*/
          sleep(5);
        }
```

Figura 4.28. Fragmento configuración valor máximo Humedad y tiempo envío Traps(demon)

Para que se realice el envío de traps, se añade este trozo de código que se muestran en las Figuras 4.26 y 4.28 al código de los demonios. En el código se puede ver dos partes importantes, la primera es que se indica el valor máximo a partir del cual se empieza a enviar los avisos, las traps. Y también se puede ver la función "send_trapTemperaturaDHT11_trap()" y "send_trapHumedadDHT11_trap()" y mediante el sleep se indica que la función es invocada cada 5 segundo cuando se cumpla la condición. En este caso está por defecto la temperatura de 24ºC y una humedad del 57%. Cuando se envien las traps el dispositivo esta configurado para activar la opcion de que dos diodos LED se iluminen en el momenento de que se sobrepasa el valor maximo predefinido. Los diodos estan conectados en el GPIO (26 y 19)

4.5. Comprobación Funcionamiento

En este apartado se procederá a la inicialización y comprobación de todas las aplicaciones previamente mencionadas con la finalidad de comprobar su correcto funcionamiento antes de la automatización.

En el desarrollo del proyecto se ha mencionado que para gestionar cada objeto de la MIB se debe generar una aplicación, dicha aplicación debe ser inicializada. Este proceso se realizará de forma manual porque los objetos no se ejecutan de forma automática. Una parte muy importante y de gran peso en este proyecto es el desarrollo de un entorno que permita la ejecución de automática de los elementos que se quieren gestionar. La forma en la que se ha desarrollado en el proyecto este proceso es mediante scripts, los cuales facilitaran el manejo y desarrollo del proyecto.

La primera comprobación que se debe saber es cómo acceder a los valores de los sensores, el comando por el cual se accede a los datos mediante el código PYTHON. Cuando el código de PYTHON accede a los sensores exporta los datos de una forma predetermina y dado que no se quiere modificar el código predeterminado en PYTHON lo que se ha realizado es una modificación al código de salida para ponerlo en el formato que más nos conviene. Eso se ha realizado mediante un SCRIPT en BASH. A continuación, se verá cómo es el código para acceder simplemente a los datos de los sensores.

```
/Adafruit_Python_DHT/examples/AdafruitDHT.py 11 4
/Adafruit_Python_DHT/examples/AdafruitDHT.py 22 17
```

Se observa que este comando accede al origen donde están instalados los recursos necesarios para acceder a los datos de los sensores. El primer número puede ser o el 11 o el 22, el cual indica que tipo de sensor es, si se trata del sensor DHT11 o del sensor DHT22 y el segundo número en el comando se debe indicar en el pin GPIO en el cual está conectado en la Raspberry Pi. El código utilizado para la automatización, desarrollo del acceso y modificación de los valores obtenidos por los sensores se puede ver en el anexo A.2. A continuación, en la Figura 4.29 se verá cómo son los valores obtenidos por los sensores.

```
vie may 12 01:36:58 CEST 2017

Ejecutando el script de lectura/escritura de los sensores DHT11 y DHT22


2017-05-12 01:37:03 SensorDHT11 Temperatura=24.0*C Humedad=57.0%
2017-05-12 01:37:03 SensorDHT22 Temperatura=24.2*C Humedad=50.3%

2017-05-12 01:37:06 SensorDHT11 Temperatura=24.0*C Humedad=58.0%
2017-05-12 01:37:06 SensorDHT22 Temperatura=24.3*C Humedad=50.5%

2017-05-12 01:37:18 SensorDHT11 Temperatura=24.0*C Humedad=57.0%
2017-05-12 01:37:18 SensorDHT22 Temperatura=24.3*C Humedad=50.4%

2017-05-12 01:37:23 SensorDHT11 Temperatura=25.0*C Humedad=56.0%
2017-05-12 01:37:23 SensorDHT22 Temperatura=24.3*C Humedad=50.4%

2017-05-12 01:37:33 SensorDHT11 Temperatura=24.0*C Humedad=57.0%
2017-05-12 01:37:33 SensorDHT22 Temperatura=24.3*C Humedad=50.4%
```

Figura 4.29. Salida consola script PYTHON sensores DHT11 & DHT22

4.5.1. Tabla (sensorTable)

A continuación, se muestran los resultados de la tabla (sensorTable) con los valores que tiene el archivo sensorLectura.txt de los sensores. Estos resultados son observados de dos formas, la primera es en forma de tabla en la Figura 4.30 y en la Figura 4.31 se visualizará el archivo sensorLectura.txt previamente mencionado donde están almacenados los valores de los sensores y la segunda se trata de la tabla organizada de forma lexicográfica.

```
snmptable -v 2c -c public -0s 192.168.1.41 sensorTable
```

```
javier@javier-desktop:~/Escritorio$ snmptable -v 2c -c public 192.168.1.41 sensorTable
SNMP table: SENSOR-TEMPERATURA-MIB::sensorTable

              sensor valor
     "HumedadDHT11"    64
     "HumedadDHT22"    47
 "TemperaturaDHT11"    23
 "TemperaturaDHT22"    23
```

Figura 4.30. Tabla sensorTable usando commando snmptable

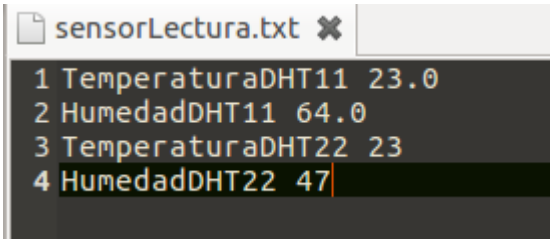


Figura 4.31. Valores sensorLecuta.txt

A través del comando *snmpwalk* se observa en la Figura 4.32 cómo se recorre la tabla ya almacenada. Se observa que recorre la tabla en un orden lexicográfico y no como esta almacenada en el archivo sensorLectura.txt.

```
javier@javier-desktop:~/Escritorio$ snmpwalk -v 2c -c public 192.168.1.41 sensorTable
SENSOR-TEMPERATURA-MIB::sensor."HumedadDHT11" = STRING: "HumedadDHT11"
SENSOR-TEMPERATURA-MIB::sensor."HumedadDHT22" = STRING: "HumedadDHT22"
SENSOR-TEMPERATURA-MIB::sensor."TemperaturaDHT11" = STRING: "TemperaturaDHT11"
SENSOR-TEMPERATURA-MIB::sensor."TemperaturaDHT22" = STRING: "TemperaturaDHT22"
SENSOR-TEMPERATURA-MIB::valor."HumedadDHT11" = INTEGER: 64
SENSOR-TEMPERATURA-MIB::valor."HumedadDHT22" = INTEGER: 47
SENSOR-TEMPERATURA-MIB::valor."TemperaturaDHT11" = INTEGER: 23
SENSOR-TEMPERATURA-MIB::valor."TemperaturaDHT22" = INTEGER: 23
javier@javier-desktop:~/Escritorio$
```

Figura 4.32 Tabla sensorTable usando snmpwalk. Orden Lexicográfico

4.5.2. Integers

La comprobación del correcto funcionamiento de los integers desarrollados en el proyecto se realiza mediante la comprobación de que los procesos estén activos y que los valores en la mib estén correctamente introducidos.

```
sudo ./temperaturaDHT11_demon
sudo ./humedadDHT11_demon
sudo ./temperaturaDHT22_demon
sudo ./humedadDHT22_demon
```

En la Figura 4.33 se observa los procesos de los cuatro demonios inicializados correctamente y en la Figura 4.34 que los valores en los objetos de la mib están correctamente almacenados.

```
root      1808  3.8  0.9  14568  8632 pts/3    S+   00:06   0:01 ./temperaturaDHT11_demon
root      1817  4.6  0.9  14556  8704 pts/3    S+   00:06   0:01 ./humedadDHT11_demon
root      1827  7.7  0.9  14568  8704 pts/3    S+   00:06   0:01 ./temperaturaDHT22_demon
root      1836 18.0  0.9  14636  8652 pts/3    S+   00:07   0:01 ./humedadDHT22_demon
```

Figura 4.33. Procesos Demonios Sensores DHT11 & DHT22

```
javier@javier-desktop:~/Escritorio$ snmpget -v 2c -c public 192.168.1.41 temperaturaDHT11.0
SENSOR-TEMPERATURA-MIB::temperaturaDHT11.0 = INTEGER: 25
javier@javier-desktop:~/Escritorio$ snmpget -v 2c -c public 192.168.1.41 humedadDHT11.0
SENSOR-TEMPERATURA-MIB::humedadDHT11.0 = INTEGER: 56
javier@javier-desktop:~/Escritorio$ snmpget -v 2c -c public 192.168.1.41 temperaturaDHT22.0
SENSOR-TEMPERATURA-MIB::temperaturaDHT22.0 = INTEGER: 25
javier@javier-desktop:~/Escritorio$ snmpget -v 2c -c public 192.168.1.41 humedadDHT22.0
SENSOR-TEMPERATURA-MIB::humedadDHT22.0 = INTEGER: 49
javier@javier-desktop:~/Escritorio$
```

Figura 4.34 Valores Integers Sensores DHT11 & DHT22

4.5.3. Traps

Cuando se ejecuta el demonio de las traps para que el objeto gestionado funcione correctamente y mande los avisos al gestor, tiene que estar funcionando a la vez el demonio del objeto que se va a monitorizar y gestionar.

En este ejemplo se visualizará funcionamiento del sensor de temperatura. Se ejecutará primero la aplicación temperatura con el siguiente comando:

```
sudo ./temperaturaDHT11_demon
```

Este comando ejecuta el objeto que se quiere monitorizar y una vez inicializado se debe ejecutar el comando para iniciar el demonio de las traps el cual mandara los avisos al gestor.

```
sudo ./trap_temperaturaDHT11_demon
```

Como nota adicional se debe tener en cuenta que hay que modificar el archivo snmpd.conf para poner la IP del gestor al que se enviara los avisos. Cuando se modifique o añada dicha IP se debe reiniciar el *servicio snmpd*. Esto se realiza con el comando *"sudo /etc/init.d/snmpd restart"*. A la hora de visualizar el funcionamiento de las traps se debe visualizar que los procesos están corriendo correctamente de forma inicial. Para visualizar los procesos se realizará introduciendo el comando por consola *"ps –aux"*. La figura 4.35, muestra un trozo del resultado de introducir dicho comando.

```
javier@javier-desktop:~/Escritorio$ ps aux | grep demon
root      2097  0.1  0.9  14768  8792 pts/3    S+   01:53   0:00 ./temperaturaDHT11_demon
root      2099  0.1  0.9  14768  8684 pts/3    S+   01:53   0:01 ./humedadDHT11_demon
root      2142  0.1  0.9  14768  8760 pts/3    S+   01:54   0:00 ./temperaturaDHT22_demon
root      2143  0.1  0.9  14768  8792 pts/3    S+   01:54   0:00 ./humedadDHT22_demon
root      2582  0.1  0.3   9584  3676 pts/3    S+   02:02   0:00 ./trap_temperaturaDHT11_demon
root      2584  0.1  0.3   9584  3696 pts/3    S+   02:02   0:00 ./trap_humedadDHT11_demon
```

Figura 4.35. Consola Procesos Traps Activas

Se observa que los dos procesos de la temperatura, tanto el objeto a gestionar, como la trap que manda los avisos se están ejecutando correctamente.

Ahora se visualizará en un programa externo al entorno del proyecto, a la Raspberry Pi. En este caso se trata del programa SnmpB, en el cual se visualizará la IP del gestor. En este caso se trata de un ordenador con Windows 10 en el cual las traps que llegan del agente cuando el valor es superado el umbral preestablecido inicialmente.

Podemos observar en la Figura 4.36 las traps que llegan a nuestro gestor, como son traps de dos tipos, la trap de temperatura y la de humedad.

Trap log

No	Date	Time	Timestamp	Notification Type	Message Type	Version	Agent Address	Agent port
0009	2017-02-08	13:34:20	0:46:46.56	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0010	2017-02-08	13:34:35	0:47:01.57	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0011	2017-02-08	13:34:39	0:47:04.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0012	2017-02-08	13:34:44	0:47:06.57	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0013	2017-02-08	13:34:48	0:47:09.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0014	2017-02-08	13:34:53	0:47:11.57	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0015	2017-02-08	13:34:57	0:47:14.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0016	2017-02-08	13:35:02	0:47:16.57	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0017	2017-02-08	13:35:06	0:47:19.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0018	2017-02-08	13:35:11	0:47:21.58	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0019	2017-02-08	13:35:15	0:47:24.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0020	2017-02-08	13:35:20	0:47:26.58	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0021	2017-02-08	13:35:24	0:47:29.10	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0022	2017-02-08	13:35:29	0:47:31.58	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0023	2017-02-08	13:35:33	0:47:34.10	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0024	2017-02-08	13:35:38	0:47:46.58	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0025	2017-02-08	13:35:42	0:47:51.59	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065

Figura 4.36. Traps Temperatura&Humedad gestor en SNMPB

4.6. Script Menú Automatización

El proyecto necesita de la inicialización de todos los programas que se han visto con anterioridad y ejecutándose de forma constante y paralela para que todo funcione correctamente. Para la más sencilla ejecución de todo el proyecto se ha desarrollado un menú con todas las opciones incluidas en el proyecto. Cabe destacar que inicialmente la generación de un entorno grafico no fue una de las principales propuestas del proyecto, pero durante el desarrollo se observó que era primordial la interfaz para la mayor facilidad y visualización del mismo.

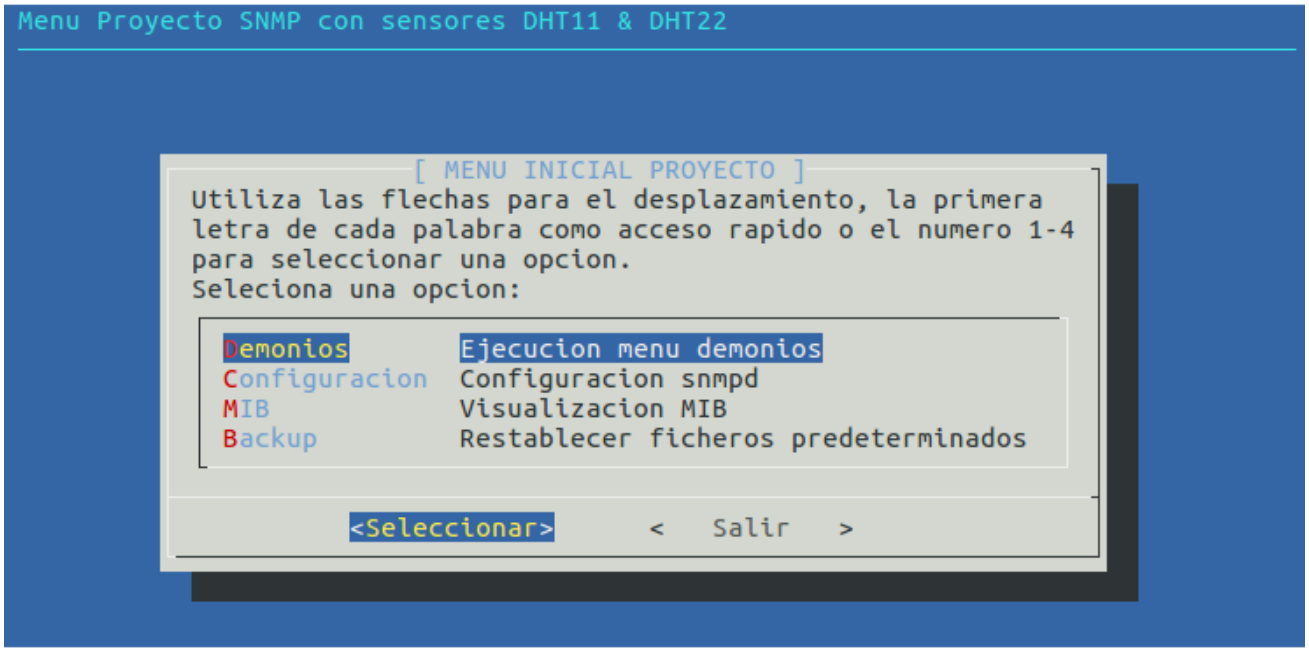


Figura 4.37. Menú Script Proyecto

Se puede observar en la Figura 4.37 que en la primera opción se encuentra la ejecución del menú de demonios la cual se explica más tarde. La segunda se trata de la configuración del fichero snmpd en el cual se encuentran como valores importantes las IPs de las Traps y los valores de las comunidades. La tercera opción se trata de una visualización tanto de la MIB su código en ASN.1 como el árbol con snmptranslate. Y la última opción se trata de la restauración de los ficheros del proyecto.

En el menú de los demonios que podemos observar en la Figura 4.38. La primera opción se trata de la visualización de los valores reales obtenidos por los sensores y de su actualización de forma periódica, además de visualizar dichos valores reales el script realiza la función de exportar dichos valores a los distintos archivos que serán utilizados para el almacenamiento y gestión en la MIB SENSOR-TEMPERATURA-MIB. El código completo de la ejecución del sensor y exportación de los valores de los mismos se podrá ver en el anexo A.2.

Esta ejecución se visualizó en la Figura 4.29, el script del sensor se ejecuta en un nuevo terminal para. Recordamos que la salida que visualizamos tiene los valores reales obtenidos por los sensores, pero posteriormente el sensor DHT22 se debe redondear dichos valores para poder almacenarlos correctamente en la MIB.

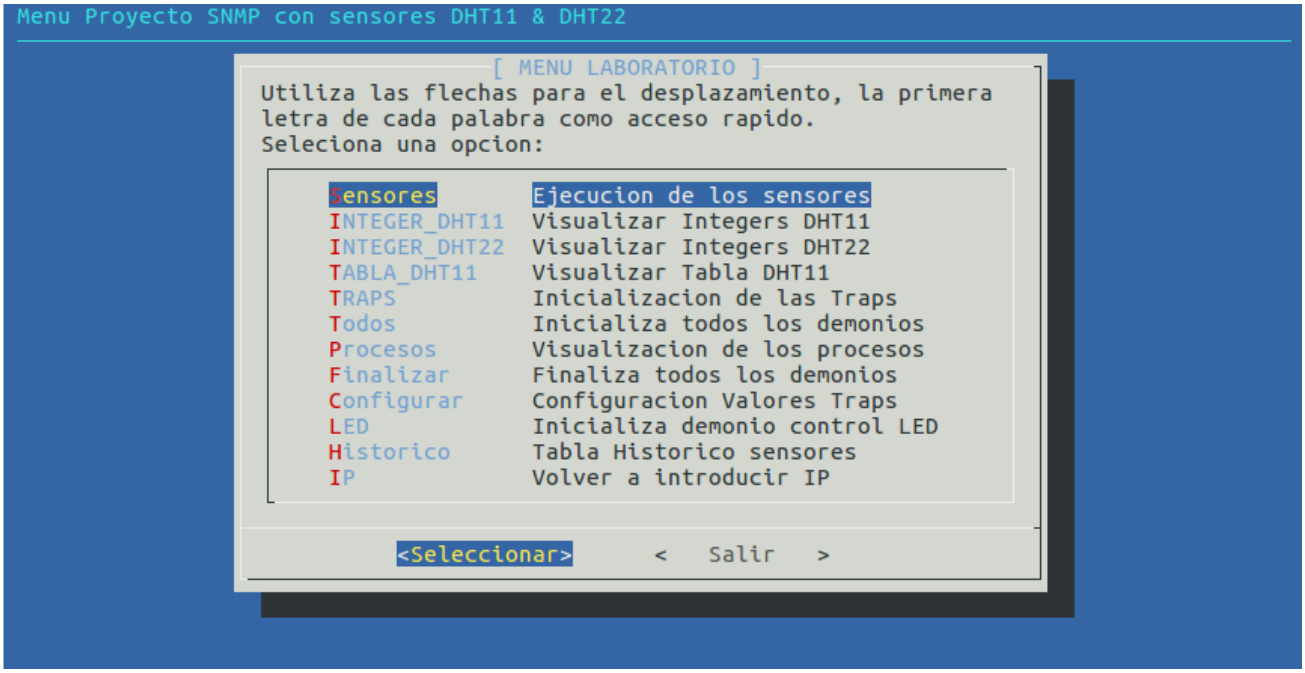


Figura 4.38 Menú Demonios

En la segunda, tercera y cuarta opción se tiene la inicialización de los integers y tabla, tanto los del sensor DHT11 como los del sensor DHT22, esto dará soporte para el correcto funcionamiento de los objetos "temperaturaDHT11" "humedadDHT11", "temperaturaDHT22", "humedadDHT22" y "sensorTable", al inicializar dichos objetos se pondrá en funcionamiento la lectura de los archivos generados por el script de los sensores y su posterior almacenamiento. La tabla lee los valores de los sensores exportados a un archivo en el cual están almacenados los valores de los cuatro integers desordenados para poder comprobar y ver cómo funciona el orden lexicográfico

La opción traps, las cuales como se ha mencionado anteriormente son el complemento de los objetos del apartado de los integers (temperaturaDHT11 y humedadDHT11), cuando se inicializan estas traps lo que realizan es la comprobación en tiempo real el valor que tiene dichos integers y si superan el valor preestablecido por el desarrollador mandara un aviso al gestor, las IPs de los gestores a las cuales se mandan los avisos ya se había visto que se configuraban en el archivo snmpd.conf.

La sexta opción simplemente es la ejecución de todas las opciones mencionadas anteriormente para inicializarlas de forma ordenada y automática y no ir de una en una además de tres procesos más se utilizará posteriormente, uno será el que gestiona el LED y los otros dos son unas tablas de valores históricos y control de los sensores.

Para una facilidad de comprobación y visualización de que los procesos previamente mencionados están activos se ha añadido una opción en la cual se visualizaran los procesos activos en la Raspberry Pi filtrándolos por los que solo puede utilizar el proyecto.

Después se tiene una opción en la cual se podrán cerrar todos los demonios ejecutados anteriormente, esto se realizar para asegurar de que todos los procesos se cierran correctamente dado que puede que algún proceso siga corriendo, aunque se haya cerrado el terminal el cual mantenía el proceso activo.

Una de las más importantes opciones creadas en el proyecto se trata de la configuración de las Traps, como se comentó anteriormente el valor máximo a partir del cual se empiezan a enviar los avisos de las Traps está configurado y compilado por el desarrollador. En esta opción se puede acceder a un script que simplemente te pide el valor que quieres introducir para la temperatura o humedad y el script simplemente busca y cambia lo necesario junto a la posterior compilación automática para el correcto funcionamiento de los nuevos valores de las traps.

La siguiente opción inicializa en este caso el proceso para que el objeto que se ha preparado para la realización de una práctica en el Laboratorio de *Gestión y Operación de Redes* que se basa en la utilización de los comandos snmpset y snmpget. Para la visualización de los comandos se ha conectado a la Raspberry Pi un diodo LED que nos ayudara a visualizar los valores introducidos como se mencionó en el apartado 4.4.1.

La opción de Historico nos abrirá un nuevo menú en el cual se podrá acceder y configurar los valores de lectura y almacenamiento de los sensores en las tablas de históricos.

Como última opción de este menú simplemente se ha añadido una opción para volver a introducir la IP por si acaso nos equivocamos cuando lo introducimos por primera vez dado que al entrar al menú de los demonios la IP es solicitada para la correcta visualización y modificación de los valores que hay en el proyecto y que hemos mencionado anteriormente.

4.6.1. Ejecución

En este apartado se comprueba el funcionamiento del proyecto de forma automatizada. Para esto primero se realiza la ejecución de los sensores los cuales exportan los valores a los distintos ficheros y posteriormente se procede a una inicialización de todos los demonios necesarios. Una vez inicializados todos los procesos como se puede ver en la Figura 4.39 se procede a hacer la comprobación de que los valores obtenidos por los sensores y exportados se corresponden con los valores gestionados y almacenados en la MIB.

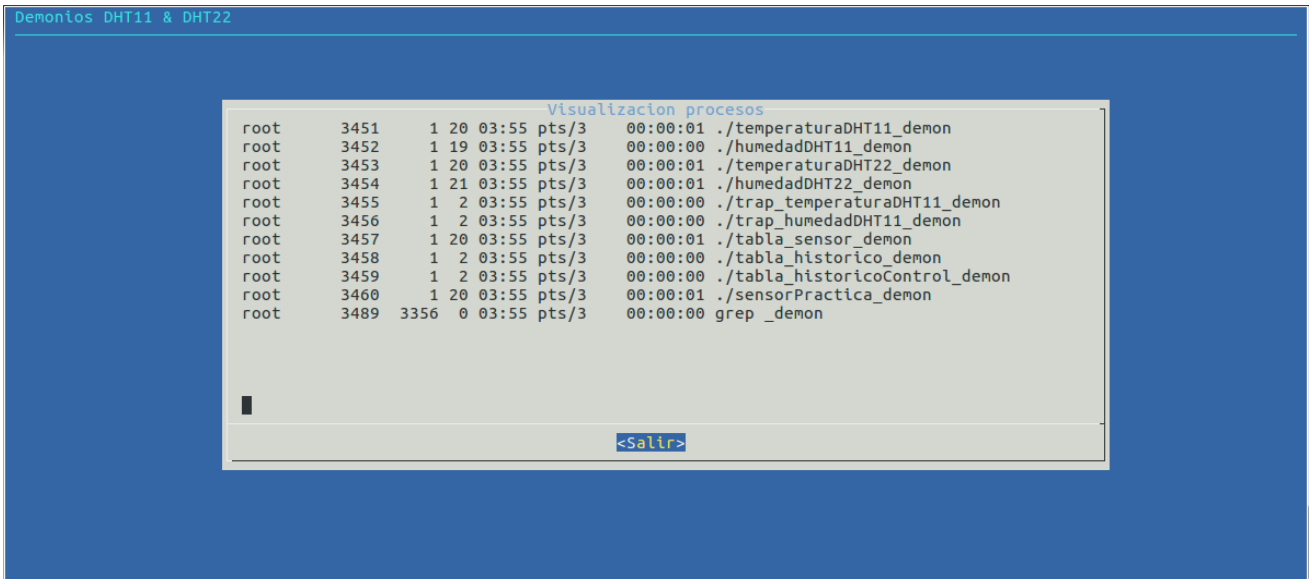


Figura 4.39. Menú todos los demonios Ejecutados

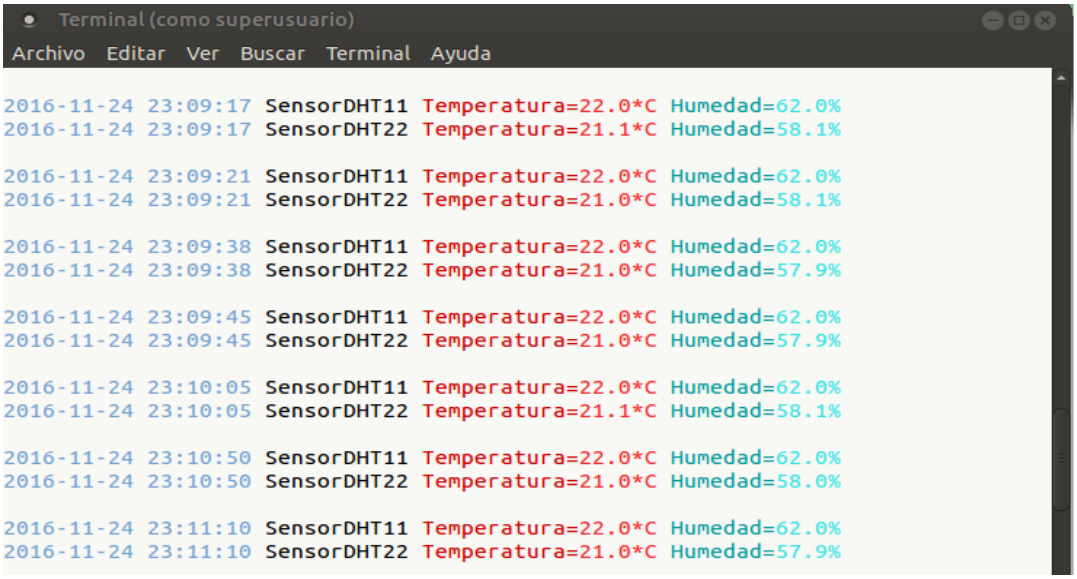


Figura 4.40. Valores obtenidos Script Sensores

En la Figura 4.41 y 4.42 pueden ver los valores que se han obtenido mediante los sensores y el script en PYTHON y se comprobará como los valores se corresponden con los obtenidos al usar los comandos *snmpget* de SNMP. También se observa como los valores de la tabla son correctos y que están ordenados de forma lexicográfica en la Figura 4.43, cosa que no sucedía en el archivo generado por el script de los sensores.

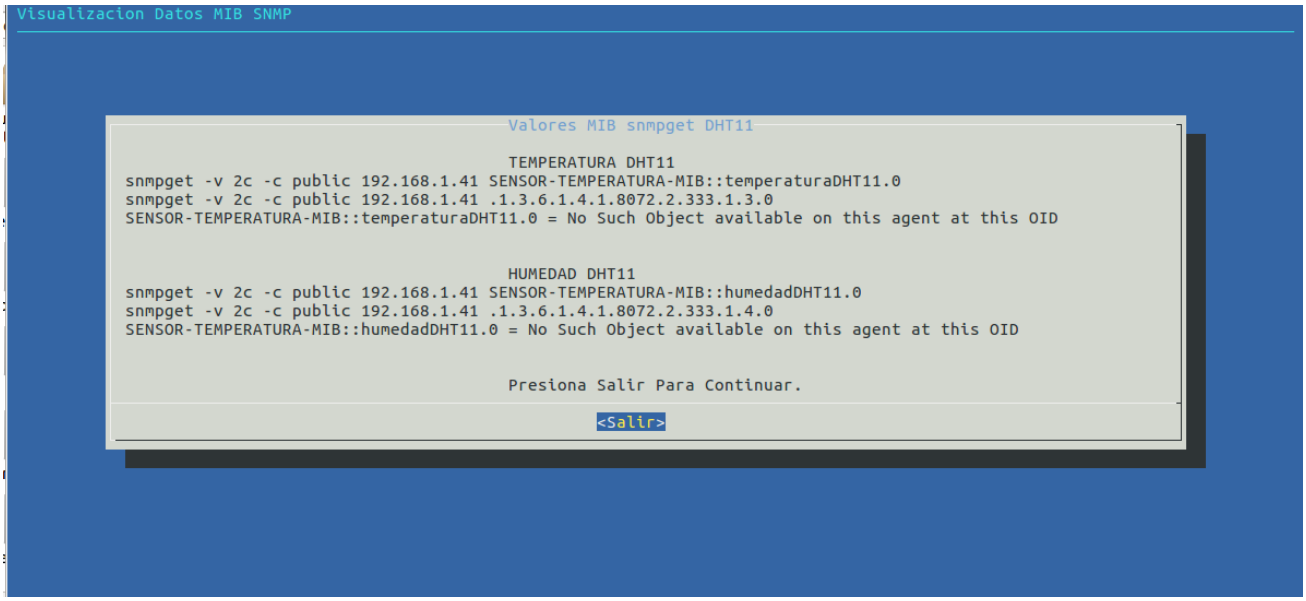


Figura 4.41. Snmpget y Snmpset DHT11. Comprobación Valores Sensor DHT11

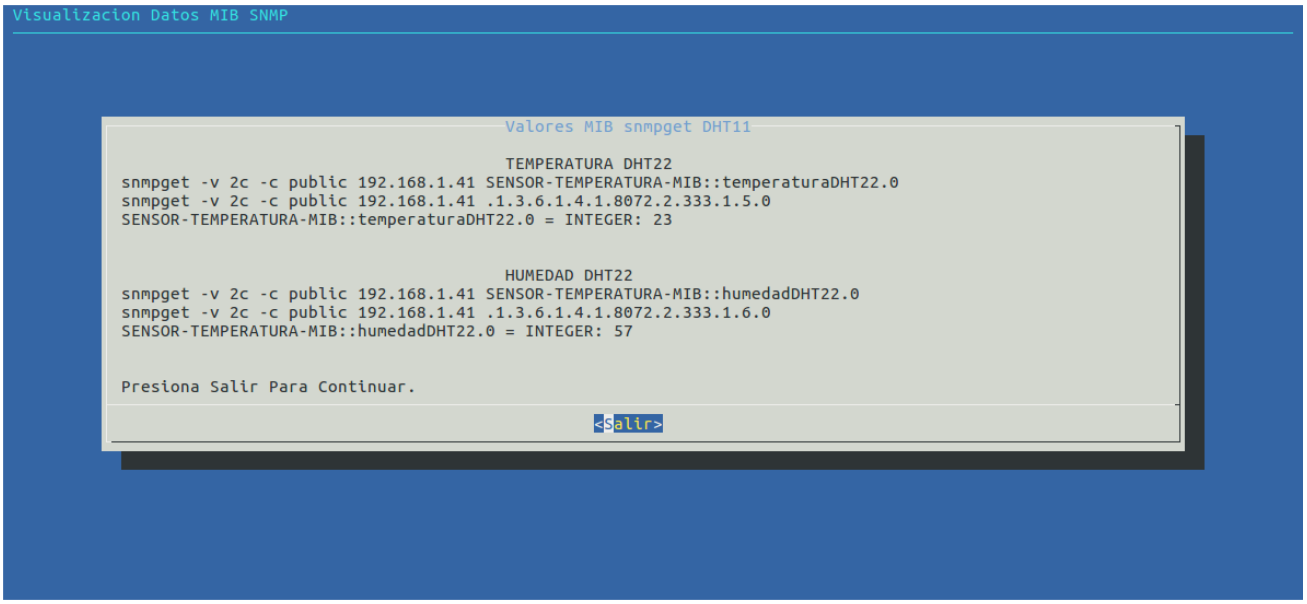


Figura 4.42 Snmpget y Snmpset DHT22. Comprobacion Valores Sensor DHT22

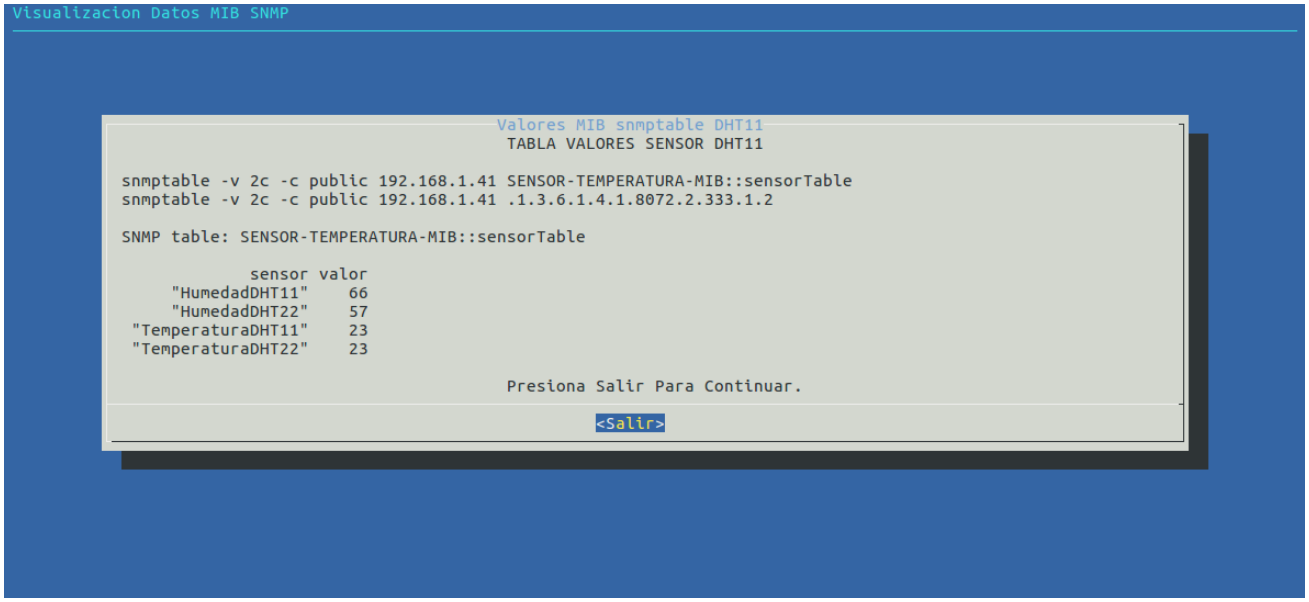


Figura 4.43 Tabla Valores Sensores

En la figura 4.44 podemos visualizar usando el comando snmpwalk como se recorre todos los valores inicializados, todos los valores gestionados en la MIB con los valores correspondientes a los sensores.

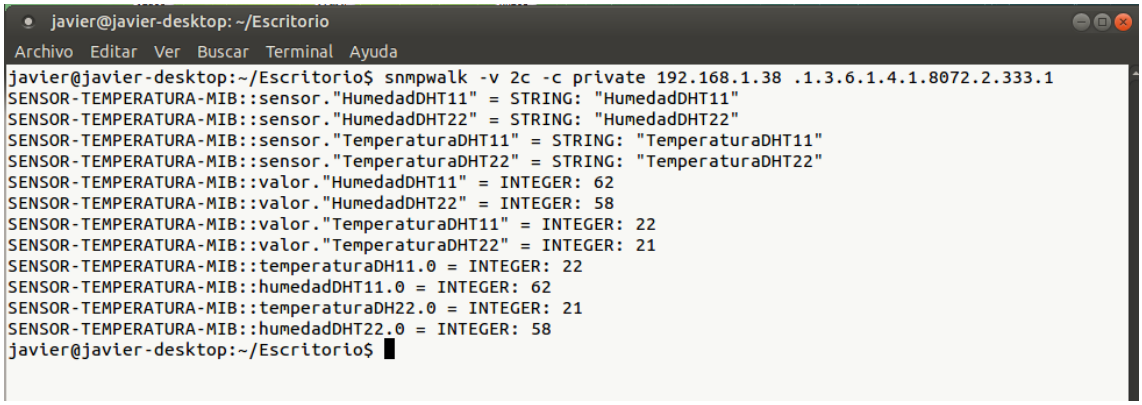
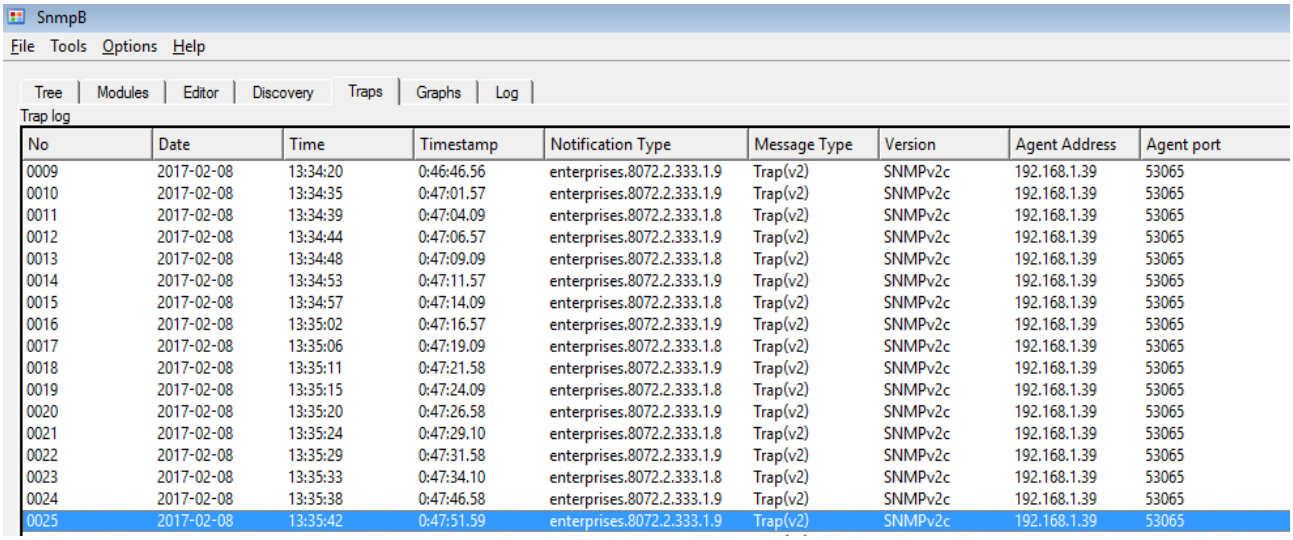


Figura 4.44. Valores MIB usando SNMPWALK

Otro punto de la comprobación de la ejecución será como al pasar el limite preestablecido por el desarrollador, el agente manda el aviso a través de las traps a los gestores. Para la visualización de la llegada constante de las traps se ha utilizado un programa de código libre llamado SnmpB. Dicho programa es corrido sobre un sistema operativo Windows 10 y se puede comprobar como cuando la temperatura o la humedad del sensor DHT11 supera el límite preestablecido se produce la llegada de las traps a los gestores. En la figura 4.45 se visualizar como se producen la llegada de las traps, de los avisos a los gestores en el programa SNMP. Hay que mencionar que en el menu se añadio la opcion de inicializar la visualizacion de las TRAPS mediante dos diodos LED, estoy quiere decir que cuando se llega al valor preestablecido de las TRAPS a parte de enviarlas a las IP de los gestores configurados tambien se iluminaran los LED en la Raspberry PI.



No	Date	Time	Timestamp	Notification Type	Message Type	Version	Agent Address	Agent port
0009	2017-02-08	13:34:20	0:46:46.56	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0010	2017-02-08	13:34:35	0:47:01.57	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0011	2017-02-08	13:34:39	0:47:04.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0012	2017-02-08	13:34:44	0:47:06.57	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0013	2017-02-08	13:34:48	0:47:09.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0014	2017-02-08	13:34:53	0:47:11.57	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0015	2017-02-08	13:34:57	0:47:14.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0016	2017-02-08	13:35:02	0:47:16.57	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0017	2017-02-08	13:35:06	0:47:19.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0018	2017-02-08	13:35:11	0:47:21.58	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0019	2017-02-08	13:35:15	0:47:24.09	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0020	2017-02-08	13:35:20	0:47:26.58	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0021	2017-02-08	13:35:24	0:47:29.10	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0022	2017-02-08	13:35:29	0:47:31.58	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0023	2017-02-08	13:35:33	0:47:34.10	enterprises.8072.2.333.1.8	Trap(v2)	SNMPv2c	192.168.1.39	53065
0024	2017-02-08	13:35:38	0:47:46.58	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065
0025	2017-02-08	13:35:42	0:47:51.59	enterprises.8072.2.333.1.9	Trap(v2)	SNMPv2c	192.168.1.39	53065

Figura 4.45. Captura traps SNMP

Otro funcionamiento será el apartado del proceso que se utiliza en el laboratorio “sensorPractica”. En este proceso lo que se comprueba será el funcionamiento del integer. Para ello se ha asociado al proceso

un diodo LED que hace más sencilla la comprobación visual del correcto funcionamiento del integer de la práctica. En la Figura 4.46 se puede ver el menú del diodo LED.

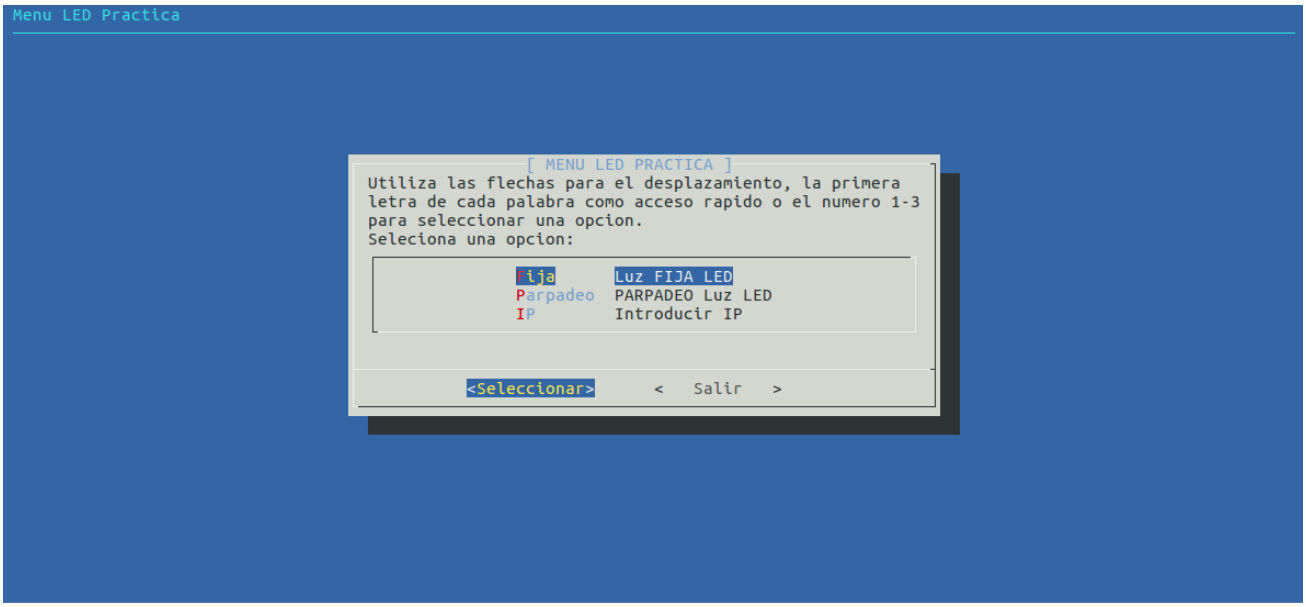


Figura 4.46. Menú sensorPractica – LED

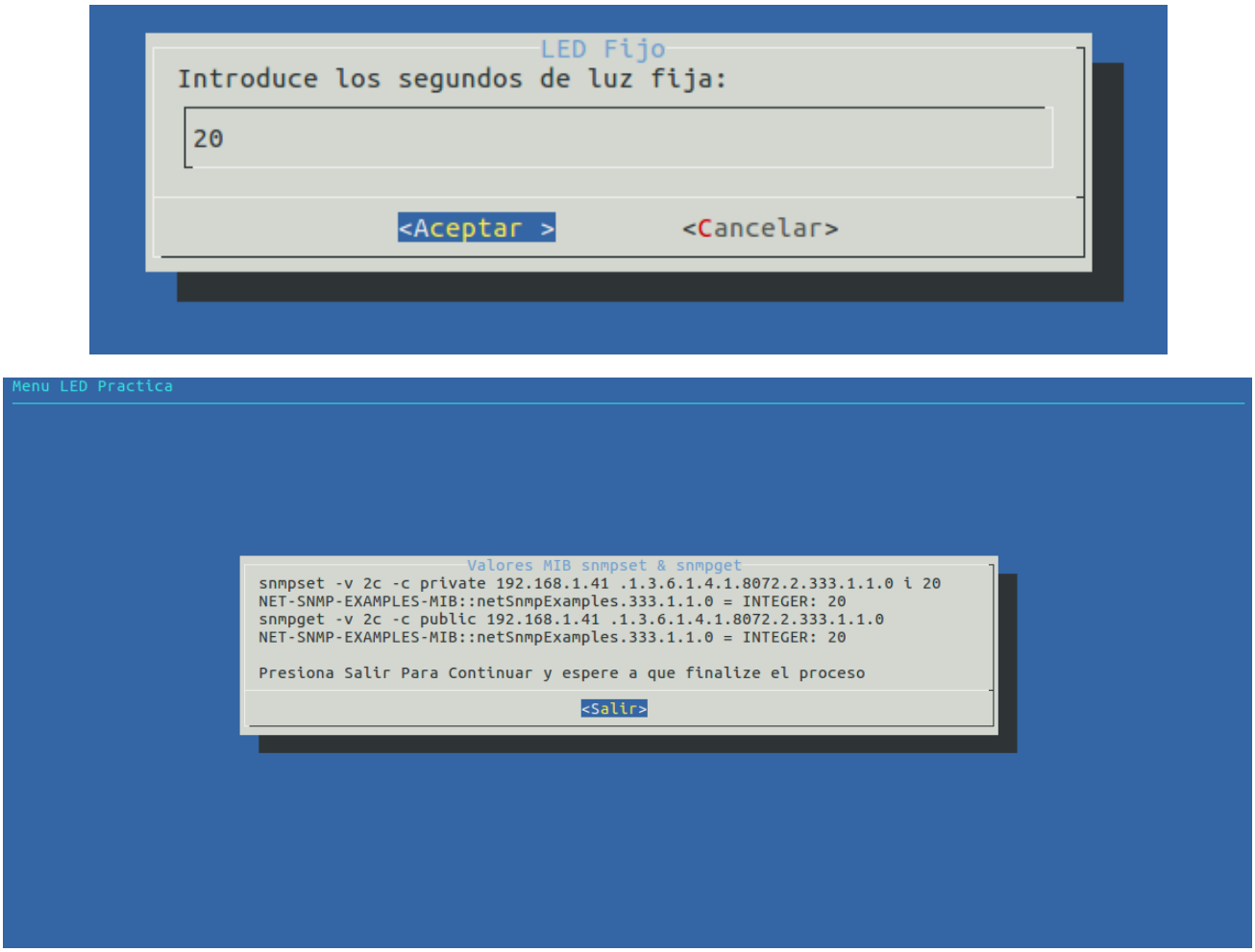


Figura 4.47. LED Fijo

En la Figura 4.47 se puede observar cuando se selecciona la opción de Luz Fija como se hace un snmpset al integer de la MIB y se introduce un valor y para la comprobación visual el diodo LED se encenderá durante 10 segundos en este caso concreto. En la Figura 4.48 cuando se selecciona la opción del parpadeo se puede observar que sucede lo mismo que en la opción anterior, se utiliza el comando snmpset para introducir el valor en la MIB y el comando snmpget para obtener el valor de dentro de la MIB, la diferencia es que en este caso el diodo LED parpadeara el número de veces introducido.

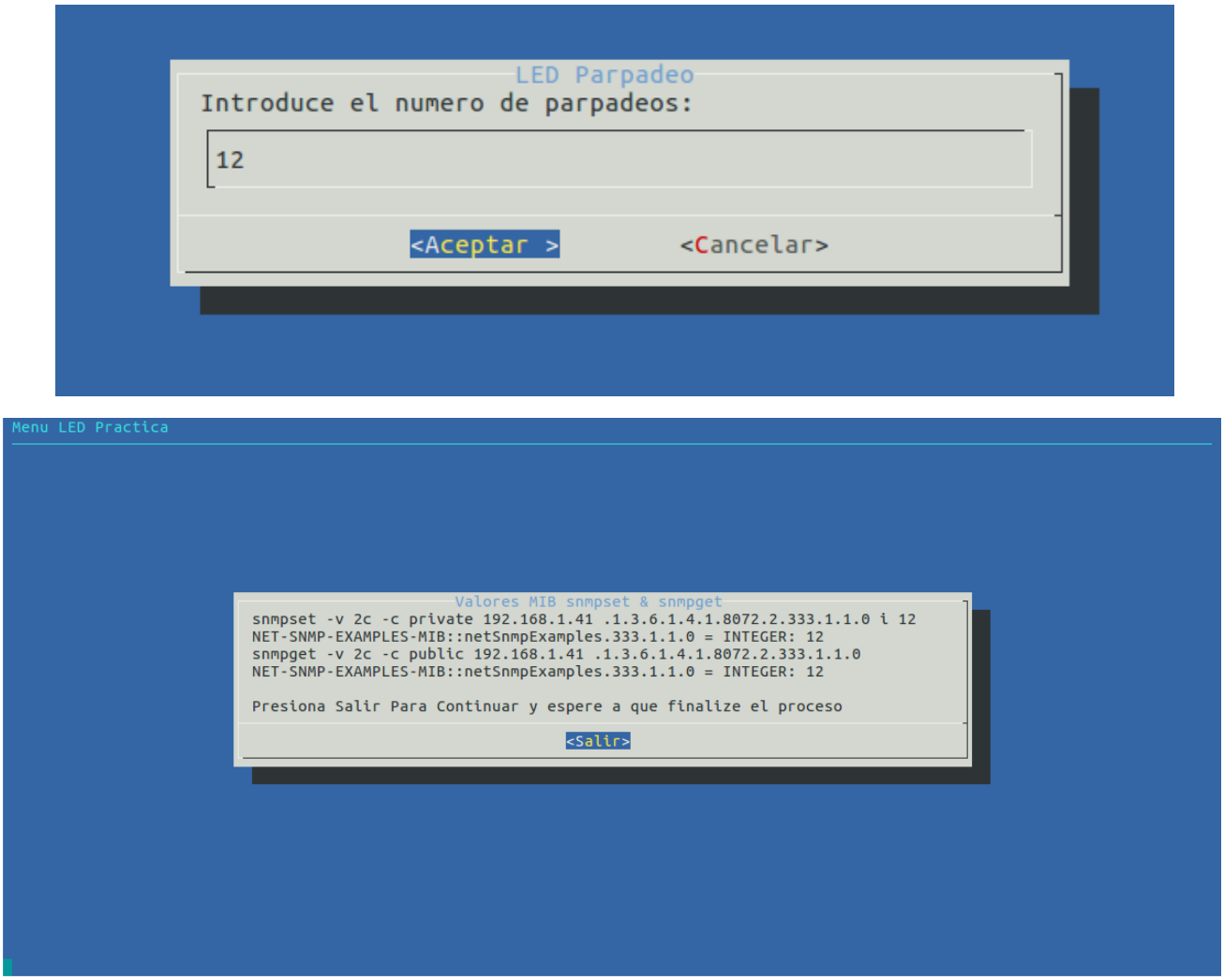


Figura 4.48. LED Parpadeo

La ultima comprobación es la de los procesos de la tabla de valores históricos. Como se mencionó anteriormente este proceso estaba formado por dos tablas, en una se almacenaban todos los valores leídos con el tiempo de lectura cuando se ha realizado y la segunda tabla se trata de una tabla de control en la cual se gestiona el tiempo de sondeo y se almacenan los valores máximos obtenidos por los sensores. Estas tablas solamente tienen la funcionalidad de lectura de los valores registrados por los sensores y el tiempo en el que se han sido registrado la lectura por ello la tabla se almacenara que se almacenara por el orden temporal en el que han sido leído los sensores. En la Figura 4.49 se observa el menú de la tabla de históricos.

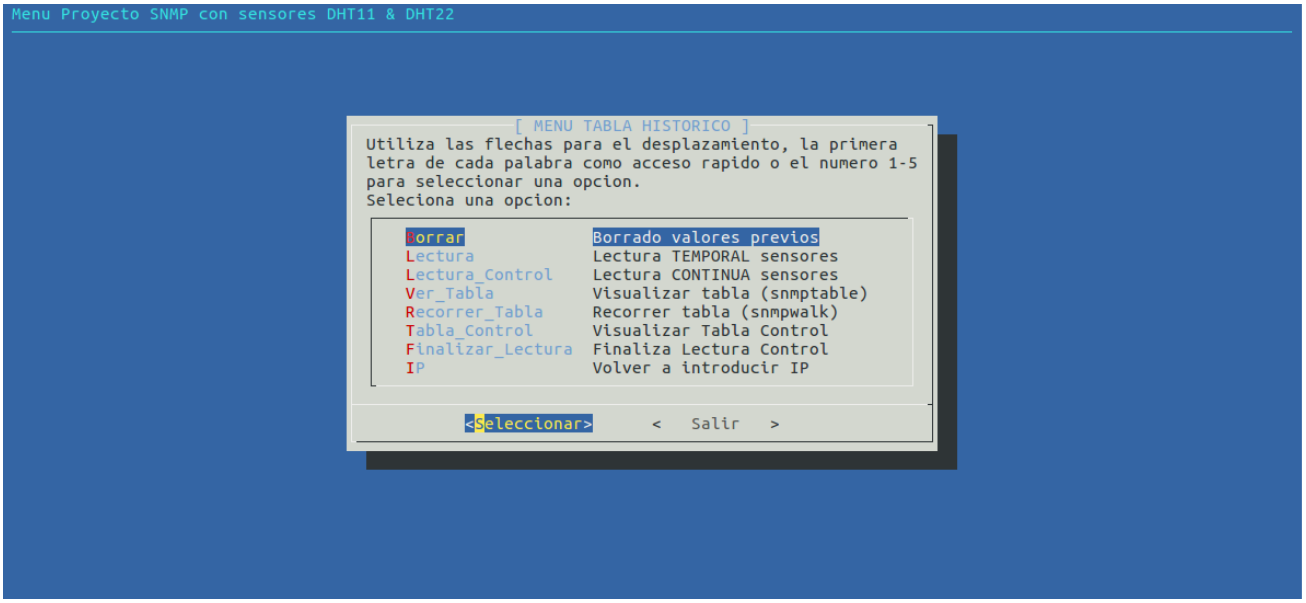


Figura 4.49 Menú Tabla Históricos

Se puede ver como la primera opción borra los valores previamente obtenidos. Las dos siguientes inicializan la lectura de los sensores, pero de maneras distintas. Una de ellas se realiza la lectura mediante la solicitud de un número determinado de lecturas y la otra se basa en una lectura y almacenamiento de los valores de los sensores en la tabla de forma continua mediante la solicitud del tiempo de sondeo tanto para el valor de la temperatura como para el valor de la humedad. Las dos siguientes opciones realizaran una visualización de los valores almacenados en la tabla de dos formas distintas. Una en forma de tabla con el comando *"snmptable"* y el otro realizara un recorrido por la tabla con el comando "snmpwalk". En la Figura 4.50 y 4.51 se visualizará un ejemplo de los comandos anteriormente mencionados y en la Figura 4.52 se podrá visualizar la tabla de control.

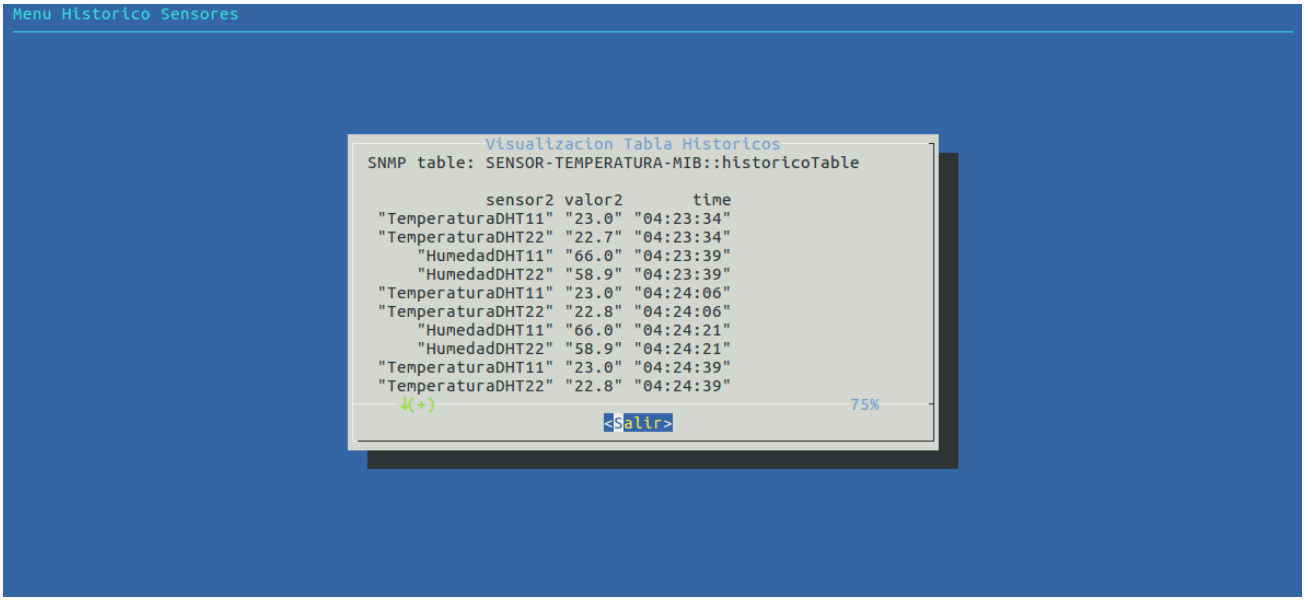


Figura 4.50 Tabla Históricos

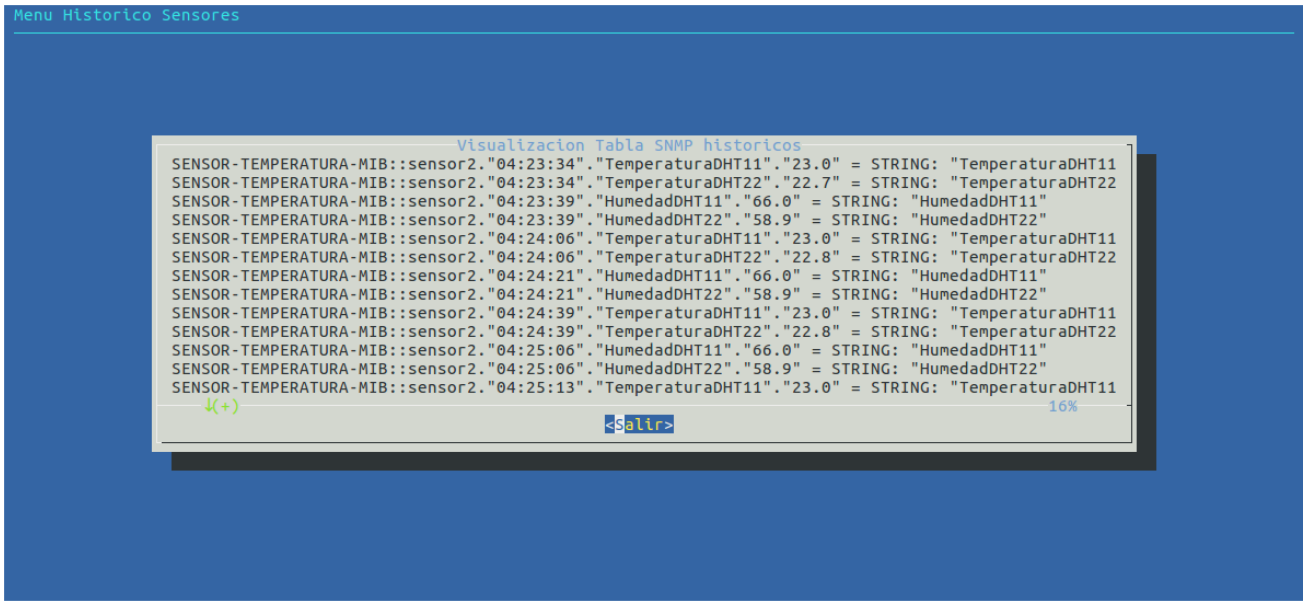


Figura 4.51 Comando snmpwalk historicoTable

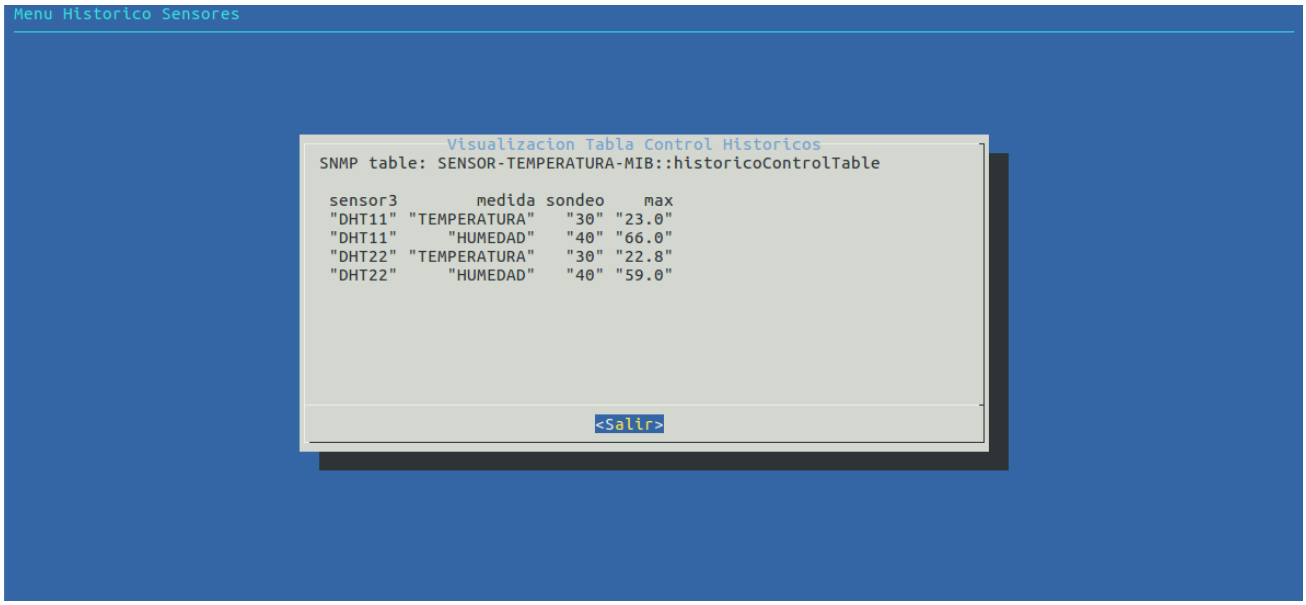


Figura 4.52 Tabla Control Históricos

En la figura 4.49 se visualiza la opción de finalizar la lectura de los sensores una vez inicializado el sondeo. Simplemente se trata de la opción para finalizar los scripts internos que sondean y exportan los valores para cuando se quiera finalizar el proceso.

La última opción que se incluye en este menú se trata simplemente de una opción para volver a introducir la IP como en otros apartados.

4.6.2. Visualización Cacti

Una parte extra añadida al proyecto ha sido un sistema en el que la Raspberry Pi exporta los valores obtenidos de forma automática y programada a un servidor online en el que se podrá visualizar los valores de los dos sensores. Para esto se ha utilizado un servicio llamado CACTI el cual utilizará servicios como MySQL, php5 y apache2. Para la instalación de CACTI se utilizará el comando[16]

```
sudo apt-get install cacti
```

Si se tienen problemas porque falten archivos para el correcto funcionamiento de CACTI se utilizará el siguiente comando

```
sudo apt-get install php5-cli php5-mysql php5-snmp snmp snmpd
rrdtool cacti
```

Una vez instalado CACTI probaremos su correcto funcionamiento accediendo a la página web de la aplicación, en este caso sería accediendo con el explorador a la página con la IP local.

```
http://192.168.1.34/cacti/
```

Dentro de la página web se han creado los templetes para el acceso a los datos de la Raspberry Pi. Estos templates serán cuatro ficheros.

```
cacti_data_template_raspberry_-_humedity
cacti_data_template_raspberry_-_temperature
cacti_data_template_raspberry_-_humedity22
cacti_data_template_raspberry_-_temperature22
```

Estos templates tendrán que ser incluidos dentro de la Raspberry Pi en la localización

```
/usr/share/cacti/resource/snmp_queries
```

Estos templates han sido generados por las variables de los datos que deben ser transmitidos y almacenados desde la Raspberry Pi hasta la página web de CACTIS. Para que los valores de los sensores sean exportados desde la Raspberry Pi hasta CACTIS se ha creado un script que lea los

valores y los almacene en la ubicación necesaria. Para realizar esto se debe hacer unos pasos previos. Primero se debe crear un script que exporte los valores leídos de los sensores a unos archivos de texto, esto se puede observar en la Figura 4.53 y posteriormente generar otro script que almacene los valores de los archivos de texto en unos OID que elegimos para que sean exportados estos valores de forma automática.

```
#!/bin/bash


############### Valores Exportar ############################

RESULTS="`python /home/javier/Escritorio/proyecto/Adafruit_Python_DHT/examples/AdafruitDHT.py 11 4 | grep Temp `"
RESULTS2="`python /home/javier/Escritorio/proyecto/Adafruit_Python_DHT/examples/AdafruitDHT.py 22 17 | grep Temp `"
TEMP="`echo ${RESULTS} | awk '{print $1}'`"
TEMP="${TEMP:5:-1}"
HUM="`echo ${RESULTS} | awk '{print $2}'`"
HUM="${HUM:9:-1}"
TEMP2="`echo ${RESULTS2} | awk '{print $1}'`"
TEMP2="${TEMP2:5:-1}"
HUM2="`echo ${RESULTS2} | awk '{print $2}'`"
HUM2="${HUM2:9:-1}"
HUM3="${HUM2:9:-1}"
echo "${TEMP}" > /home/pi/temp.txt;
echo "${HUM}" > /home/pi/humid.txt;
echo "${TEMP2}" > /home/pi/temp22.txt;
echo "${HUM2}" > /home/pi/humid22.txt;
read a < /home/pi/temp22.txt
read b < /home/pi/humid22.txt
echo "($a+0.5)/1" | bc > /home/pi/temp222.txt;
echo "($b+0.5)/1" | bc > /home/pi/humid222.txt;
read a < /home/pi/temp222.txt
read b < /home/pi/humid222.txt
echo "${a}" > /home/pi/temp2.txt
echo "${b}" > /home/pi/humid2.txt
```

Figura 4.53 Script Valores Exportados Cacti

El en la Figura 4.54 se visualizará el script de como es el esquema para exportar los valores de los sensores a los OID que leerá la herramienta CACTI y generará las gráficas. Solo se visualizará un script debido a que los otros tres son iguales para cambiando el archivo de texto de donde se lee el valor y el OID donde está almacenado. En el anexo A.18 se visualizará los restantes scripts que exportan estos valores a los OID.

temp.sh

```
#!/bin/bash
echo .1.3.6.1.2.1.25.1.8.1
echo gauge
cat /home/pi/temp.txt
```

Figura 4.54 Script Temperatura CACTI

Como último paso se debe configurar el fichero *snmpd.conf* que ya se ha mencionado con anterioridad. Se debe añadir las OID donde se almacenarán automáticamente los valores y la ruta de donde se ejecuta

el script para la lectura del valor de los sensores. En la Figura 4.55 se puede observar un fragmento del archivo snmpd.conf.

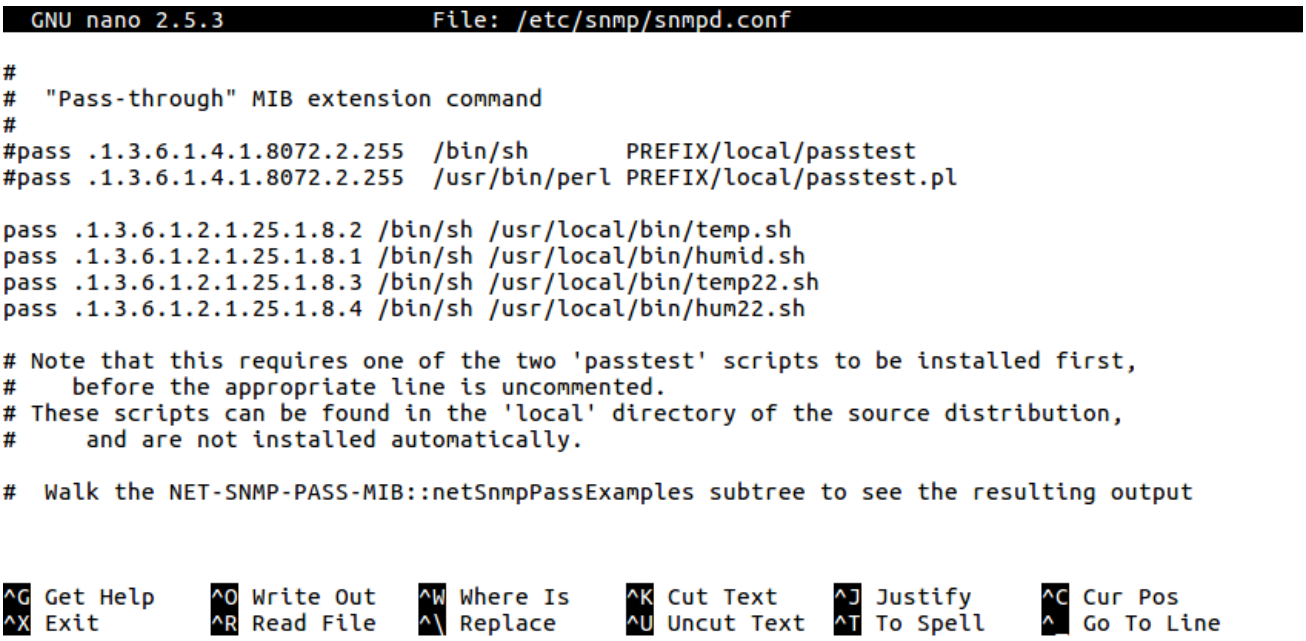


Figura 4.55 Fragmento archivo snmpd.conf OID CACTI

Una vez modificado el fichero *snmpd.conf* se debe reiniciar el servicio para que los cambios se vean afectados. Para reiniciar el servicio se utilizará el comando

```
service snmpd restart
```

El último paso que se debe realizar se trata de la ejecución periódica del script que automatiza todo, para esto se debe generar una tarea temporal en el sistema operativo de Ubuntu. Con el comando "*sudo crontab –e"* y se añadirá lo siguiente.

```
* * * * * /var/scripts/cacti_exportar.sh
```

En la Figura 4.56 se puede observar el fichero que se debe editar y añadir el comando anterior para la ejecución automática cada minuto.

Una vez configurado todo se debe acceder a la página de CACTIS para añadir la IP de la Raspberry PI y una vez añadida ya se puede visualizar las gráficas de Temperatura y Humedad de los sensores DHT11 y DHT22. En la Figura 4.57 se visualizará la gráfica del sensor DHT11 y en la Figura 4.58 se visualizará la gráfica del sensor DHT22 [17].

```
  GNU nano 2.5.3              File: /tmp/crontab.AUaMev/crontab

# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
* * * * * /var/scripts/cacti_exportar.sh
```

Figura 4.56 Archivo crontab

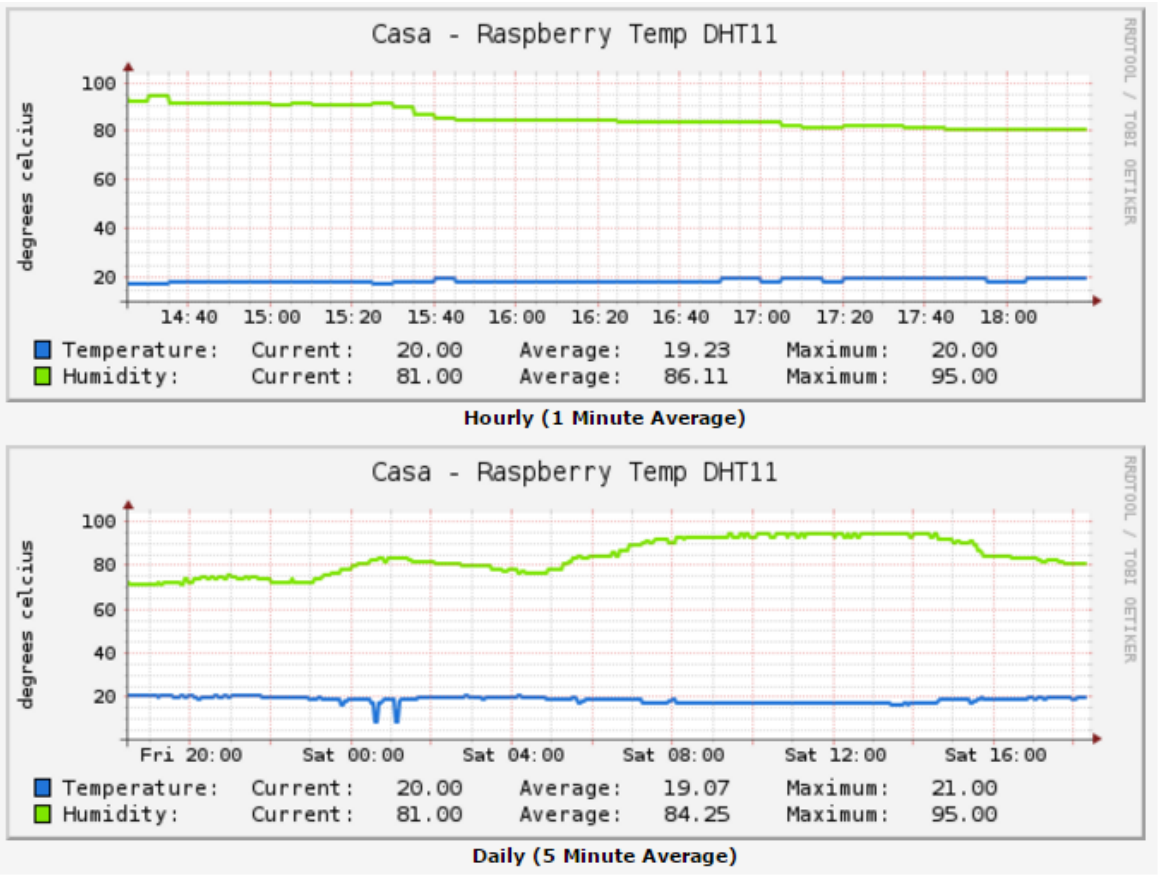


Figura 4.57 Grafica Cacti Sensor DHT11

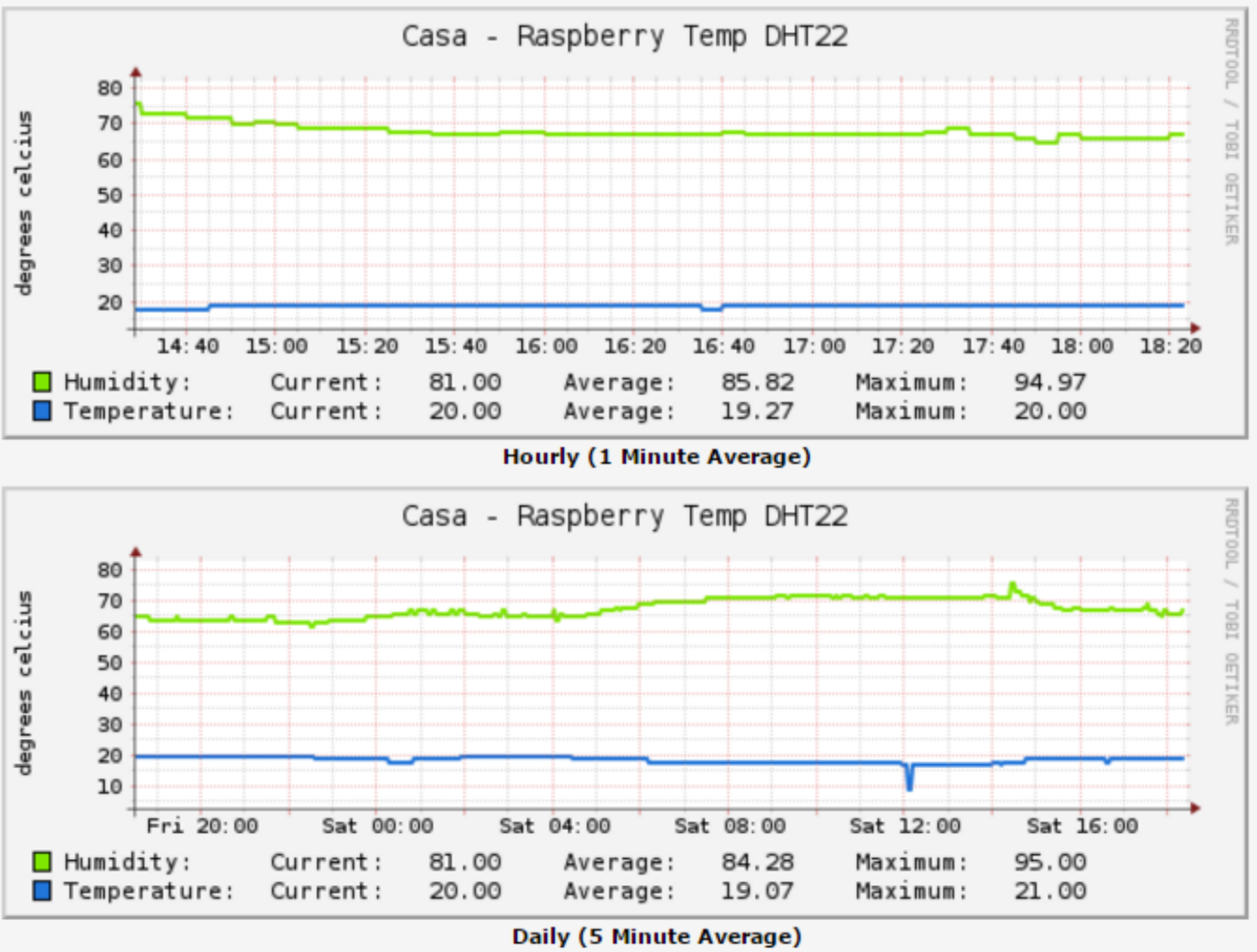


Figura 4.58 Grafica Cacti Sensor DHT22

4.6.3. Rendimiento

Al inicio del proyecto se ha mencionado que la Raspberry Pi 3 tenía la suficiente potencia en comparación con sus antecesores por lo que se usó Ubuntu, que es el sistema operativo utilizado en el Laboratorio de *Gestión y Operación de Redes*. Además de ser el SO utilizado en el laboratorio se ha decidido utilizar Ubuntu por su interfaz gráfico dado que es muy intuitivo y sencillo y permite un manejo más sencillo del proyecto. Usando el comando TOP en el terminal se visualizará como se reparte el uso de los recursos en el sistema.

Al estar todos los demonios en funcionamiento del proyecto se ve que los procesos que más recursos consumen no tiene nada que ver con el proyecto, son procesos del propio sistema operativo. En la Figura 4.56 como el SO indica que cuando están todos los procesos activos la CPU tiene una ocupación del 9,4% (El campo id “idle”, 90,6) por lo que la Raspberry Pi tiene la suficiente potencia para añadir más aplicaciones al proyecto y mover el SO a su vez de forma fluida.

En el apartado de la memoria usada se puede observar que la Raspberry Pi 3 tiene una memoria total de 948056 Kbs y están libres 359048 Kbs por lo que se tiene el 37,87% libre de memoria. La mayoría de la memoria utilizada es por los procesos internos del SO y cómo se observa en la Figura 4.57, la

memoria utilizada por los procesos del proyecto es muy pequeña y que ninguno de los procesos llega al 1% de la memoria utilizada ni de la CPU utilizada.

Hay que destacar que en la Figura 4.59, podemos ver el comando Xorg que se trata del entorno gráfico de Linux, el comando mate-terminal que se trata de los terminales utilizados para el desarrollo del proyecto, el escritorio remoto xrdp para el actual desarrollo del proyecto y el comando Python el cual es el utilizado para el correcto funcionamiento de la lectura de los sensores.

Todos estos comandos son los que tienen un valor destacable en el consumo de la CPU y memoria de la Raspberry Pi, lo que implica que en un futuro se podría desarrollar y añadir nuevas aplicaciones dado que los procesos actuales que se han generado y compilado cuando se ejecutan producen un gasto de recursos en el hardware muy bajo como se puede ver en la Figura 4.60.

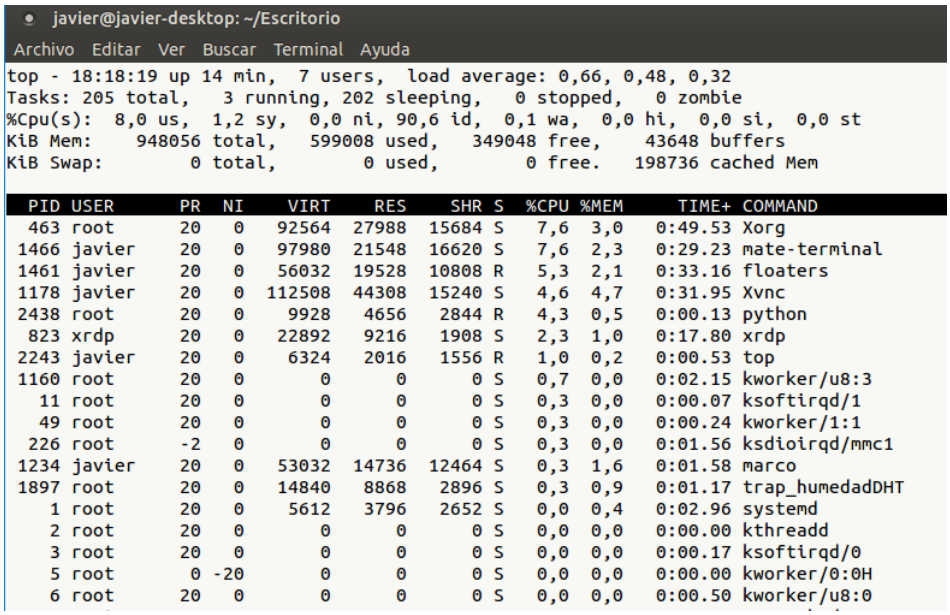


Figura 4.59. Procesos comando Top

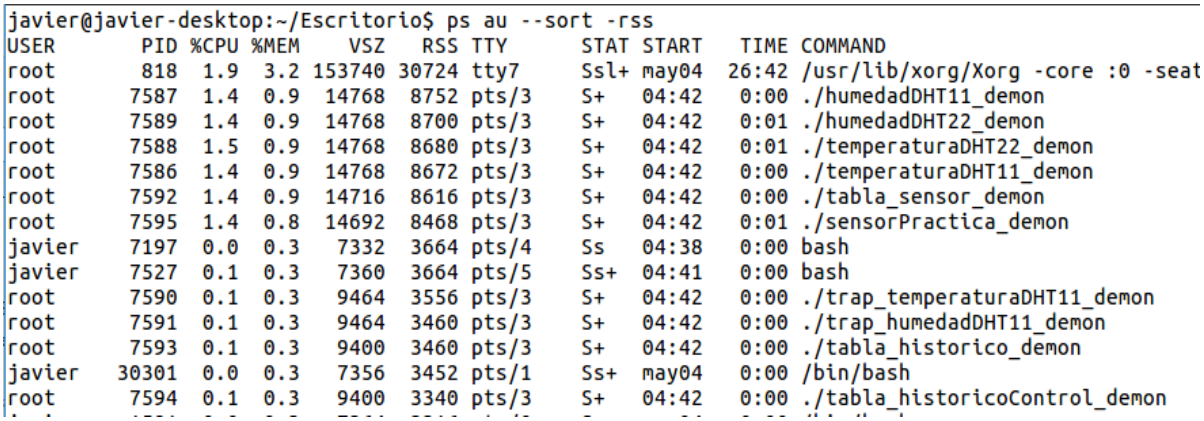


Figura 4.60. Procesos Proyecto comando ps au –sort -rss

4.7. Practica

Se ha desarrollado una práctica para la asignatura “Gestión y Operación de Redes” en la cual se usa el proyecto de forma didáctica para la visualización y comprensión del protocolo SNMP en un entorno real. Dicha práctica se puede visualizar en el Anexo 23.

5. Conclusiones

En el proyecto inicialmente tuvo el propósito de crear un agente SNMP extensible utilizando los recursos disponibles de software libre los cuales son accesibles a todos los usuarios. En el desarrollo del proyecto se ha comprado una Raspberry Pi 3 que se trata de un Mini Pc (Placa Reducida) dado que se trata de un hardware con una gran popularidad y de bajo coste.

En el proyecto después de desarrollar y comprobar todas las funciones del agente extensible y su correcto funcionamiento de forma autónoma se intentó llevar el proyecto un paso más lejos del simple hecho de simular las aplicaciones de forma teórica. Se ha introducido un entorno real en el que se gestionan dos sensores de temperatura y humedad. Con dichos sensores el agente extensible es capad de gestionar de forma autónoma los valores que recibe y mandar los avisos a los gestores configurados.

Como último aspecto a destacar del proyecto se hace hincapié en el desarrollo de los scripts en "bash" para la inicialización automática de todos los procesos necesarios para el correcto funcionamiento e implementación de los sensores y del agente. Dicha implementación se ha realizado a través de agentes extendidos AgentX. Al realizar el desarrollo mediante agentes extendidos se podrá añadir todos los dispositivos que se quiera administrar de forma sencilla dado que la propia herramienta SNMP nos permite hacerlo, esto permitirá añadir más dispositivos para gestionar.

Las dificultades que se han encontrado en el proyecto han estado basadas en la poca información que se tenía respecto a la herramienta con la que se desarrolló el entorno. Dicha herramienta producía un código genérico básico el cual tenía errores que hubo que corregir con la información obtenida de desarrolladores que habían sufrido dichos errores previamente. También cabe destacar el problema al desarrollar los scripts dado que están basados en un lenguaje del cual no se tenía ningún conocimiento.

Este proyecto ya deja entrever las líneas futuras de desarrollo dado que las nuevas Raspberry Pi tienen potencia suficiente para mover cualquier tipo de sensor o en su defecto muchos sensores a la vez. Con esto se podría mejorar el proyecto y por ejemplo realizar una instalación domestica de la Raspberry Pi conectada a los sensores y que esta a su vez estuviera conectada a la calefacción de modo que cuando los sensores detectasen cierta temperatura se encendiese o se apagase de manera totalmente autónoma.

Referencias

1. *ITU-T Study Group 17 ASN.1 Project, Paul Thorpe. Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation. ITU-T*

2. *William Stallings. SNMP and SNMPv2: The Infrastructure for Network Management. IEEE Communications Magazine, marzo de 1998.*

3. *K. McCloghrie, M. Rose. RFC 1213 - Management Information Base for Network Management of TCP/IP-based internets: MIB-II. Internet Engineering Task Force*

4. *M. Rose, K. McCloghrie. RFC 1155 - Structure and Identification of Management Information for TCP/IP-based Internets. Internet Engineering Task Force, mayo de 1990*

5. *Página AGENTE X (https://tools.ietf.org/html/rfc2741)*

6. *Página oficial de Raspberry Pi (www.raspberrypi.org)*

7. *Responsables del proyecto Raspberry Pi. Información oficial del proyecto Raspberry Pi (https://www.raspberrypi.org/). Raspberry Pi Foundation.*

8. *Página sensor DHT11 (http://www.micropik.com/PDF/dht11.pdf)*

9. *Página sensor DHT22 (https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf)*

10. *Responsables del proyecto Net-SNMP. Página oficial (www.net-snmp.org).*

11. *Página oficial de MG SOFT MIB Browser (www.mg-soft.si/mgMibBrowserPE.html)*

12. *Free Software Foundation. GNU General Public License version 2.0 (GPLv2). Free Software Foundation Incorporated, junio de 1991.*

13. *Comunidad de desarrolladores de Net-SNMP. Manual de mib2c (http://www.net-snmp.org/docs/man/mib2c.html). Proyecto NetSNMP (www.netsnmp.org), 2009.*

14. *Página DHT Sensor (https://github.com/adafruit/Adafruit_Python_DHT.git)*

15. *Comunidad de desarrolladores de Net-SNMP. Tutorial de Net-SNMP (http://www.net-snmp.org/tutorial). Proyecto Net-SNMP, 2005.*

16. *Página web Cacti (http://www.cacti.net/)*

17. *Página web Configuracion SNMP Cacti (http://www.kbza.org/2013/07/04/sensor-de-temperatura-snmp-con-raspberry-y-cacti/)*

Anexos

A.1 Árbol MIB SENSOR-TEMPERATURA-MIB

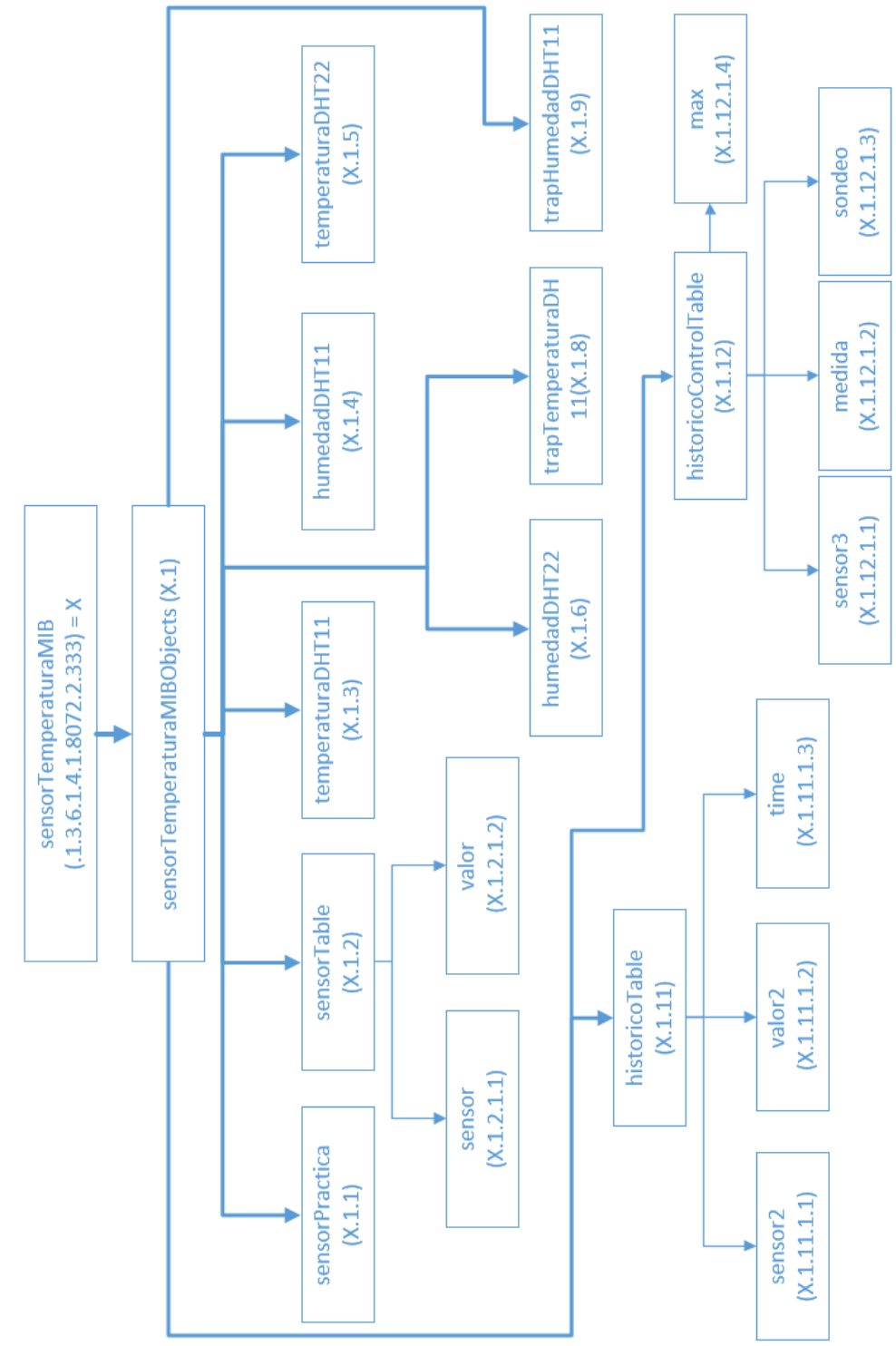


Figura A1.1. Árbol MIB SENSOR-TEMPERATURA-MIB (OIDs)

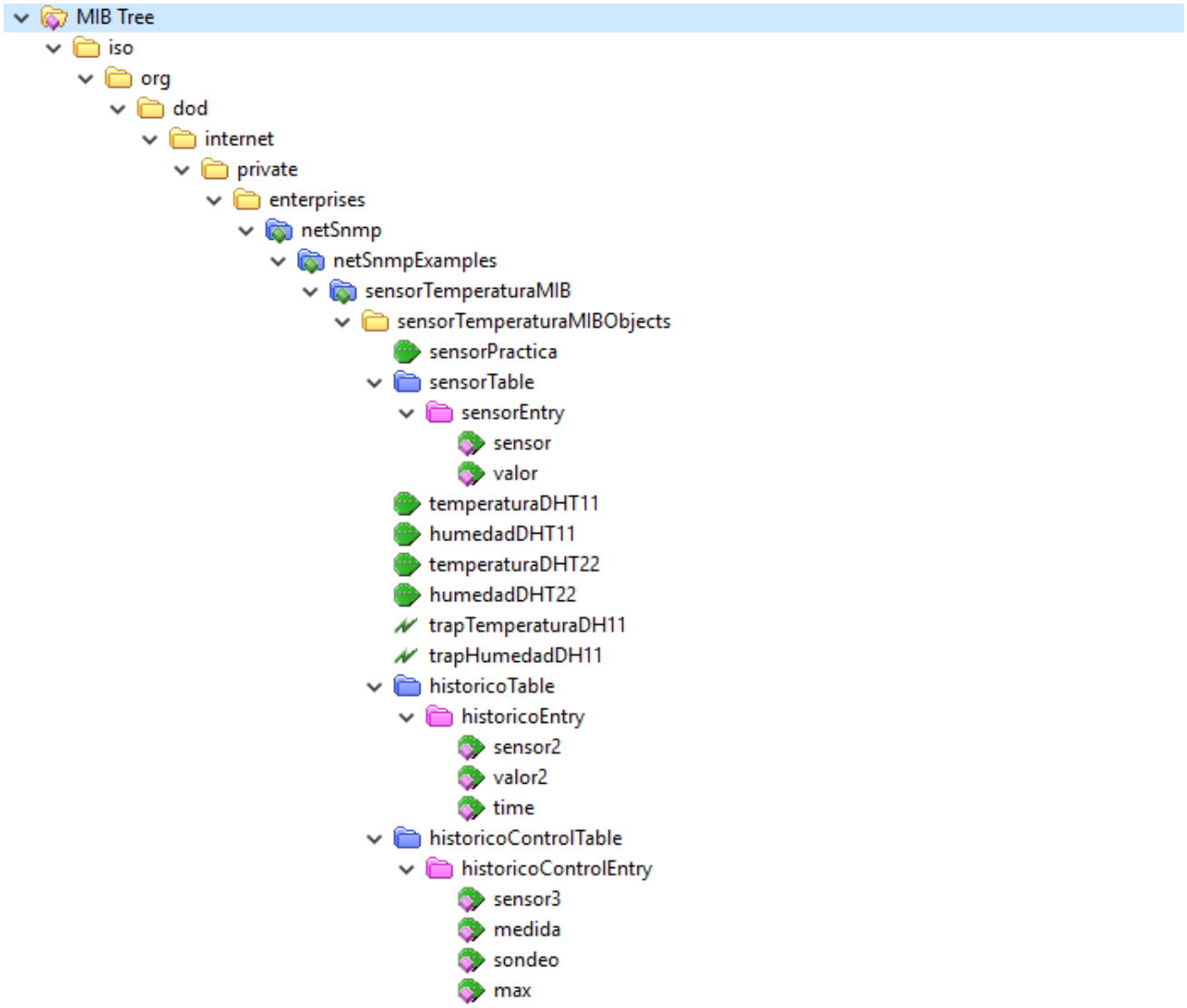


Figura A.1.2 Árbol MIB SENSOR-TEMPERATURA-MIB (mib-browser)

A.2 Script Sensores

```
#!/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi
n
export PATH

echo

#FECHA
date

echo

echo -e "\e[0;34mEjecutando el script de lectura/escritura de
los sensores DHT11 y DHT22"

echo

echo -e "\e[1;37m"

while true; do
RESULTS="`python
/root/Adafruit_Python_DHT/examples/AdafruitDHT.py 11 4 | grep
Temp `"
RESULTS2="`python
/root/Adafruit_Python_DHT/examples/AdafruitDHT.py 22 17 | grep
Temp `"
TEMP="`echo ${RESULTS} | awk '{print $1}'`"
TEMP="${TEMP:5:-1}"
HUM="`echo ${RESULTS} | awk '{print $2}'`"
HUM="${HUM:9:-1}"
TEMP2="`echo ${RESULTS2} | awk '{print $1}'`"
TEMP2="${TEMP2:5:-1}"
HUM2="`echo ${RESULTS2} | awk '{print $2}'`"
HUM2="${HUM2:9:-1}"
HUM3="${HUM2:9:-1}"
DATE="`date \"+%Y-%m-%d %H:%M:%S\"`"

#Escribe los valores obtenidos por el sensor DHT11
echo -e "\e[94m${DATE} \e[mSensorDHT11
\e[31mTemperatura=\e[1;31m${TEMP}*C\e[m
\e[36mHumedad=\e[1;36m${HUM}%\e[0m" >
/home/javier/Escritorio/proyecto/Demonios/SENSOR/temp+hum.log;
#Almacena los valores para la tabla del sensor DHT11
echo "TemperaturaDHT11 ${TEMP}" >
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorLect
ura.txt;
echo "HumedadDHT11 ${HUM}" >>
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorLect
ura.txt;
```

```
#Almacena los valores para los Integers DHT11
echo "${TEMP}" >
/home/javier/Escritorio/proyecto/Demonios/TemperaturaDHT11/sensor
Valor1.txt;
echo "${HUM}" >
/home/javier/Escritorio/proyecto/Demonios/HumedadDHT11/sensorValo
r2.txt;
#Escribe los valores obtenidos por el sensor DHT22
echo -e "\e[94m${DATE} \e[mSensorDHT22
\e[31mTemperatura=\e[1;31m${TEMP2}*C\e[m
\e[36mHumedad=\e[1;36m${HUM2}%\e[0m\n" >>
/home/javier/Escritorio/proyecto/Demonios/SENSOR/temp+hum.log;
#Almacena los valores para la tabla del sensor DHT22
#Almacena los valores para los Integers DHT22
echo "${TEMP2}" >
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor
44.txt;
echo "${HUM2}" >
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor
33.txt;
read a <
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor
44.txt
read b <
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor
33.txt
echo "($a+0.5)/1" | bc >
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor
444.txt;
echo "($b+0.5)/1" | bc >
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor
333.txt;
read a <
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor
444.txt
read b <
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorValor
333.txt
echo "TemperaturaDHT22 ${a}" >>
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorLectu
ra.txt;
echo "HumedadDHT22 ${b}" >>
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorLectu
ra.txt;
cat
/home/javier/Escritorio/proyecto/Demonios/SENSOR/temp+hum.log;
done
```

Figura A2. Código Script Sensores

A.3 Script Menú Proyecto

```
#!/bin/bash

# while-menu-dialog: a menu driven system information program

DIALOG_CANCEL=1
DIALOG_ESC=255
HEIGHT=14
WIDTH=60

printf '\e[8;32;132t'

function menu2(){
      ./lab.sh
}

function mib(){
./mib.sh
}


while true; do
  exec 3>&1
  selection=$(dialog \
    --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" \
    --title "[ MENU INICIAL PROYECTO ]" \
    --clear \
     --ok-label "Seleccionar" \
    --cancel-label "Salir" \
    --menu "Utiliza las flechas para el desplazamiento, la primera\n\
letra de cada palabra como acceso rapido o el numero 1-4  \n\
para seleccionar una opcion.\n\
Seleciona una opcion:" $HEIGHT $WIDTH 4 \
    "Demonios" "Ejecucion menu demonios" \
    "Configuracion" "Configuracion snmpd" \
     "MIB" "Visualizacion MIB" \
     "Backup" "Restablecer ficheros predeterminados" \
    2>&1 1>&3)
  exit_status=$?
  exec 3>&-
  case $exit_status in
    $DIALOG_CANCEL)
      clear
      #echo "Program terminated."
                                dialog --title "[ MENU INICIAL
PROYECTO ]" \
                                          --backtitle "Menu Proyecto
SNMP con sensores DHT11 & DHT22" \
--yesno "\nEstas seguro que quieres salir del proyecto?" 7 60

# Get exit status
# 0 means user hit [yes] button.
# 1 means user hit [no] button.
```

```
# 255 means user hit [Esc] key.
                                                response=$?
                                                case $response in
                                                      0) clear
                                                         exit;;
                                                      1) ;;
                                                      255) ;;
                                                esac
            ;;
    $DIALOG_ESC)
      clear
                                    dialog --title "[ MENU INICIAL
PROYECTO ]" \
                                                --backtitle "Menu Proyecto
SNMP con sensores DHT11 & DHT22" \
                                                --yesno "\nEstas seguro
que quieres salir del proyecto?" 7 60

                              # Get exit status
                              # 0 means user hit [yes] button.
                              # 1 means user hit [no] button.
                              # 255 means user hit [Esc] key.
                                                response=$?
                                                case $response in
                                                      0) clear
                                                         exit;;
                                                      1) ;;
                                                      255) ;;
                                                esac
            ;;
  esac
  case $selection in
    0 )
      clear
      echo "Program terminated."
      ;;
    Demonios )
      menu2;;
    Configuracion )
      ./configuracion.sh;;
     MIB) mib;;
     Backup) ./backup2.sh;;
  esac
done
```

Figura A.3 Script Menú Proyecto

A.4 Script Menú Demonios

```
#!/bin/bash

DIALOG_CANCEL=1
DIALOG_ESC=255
HEIGHT=14
HEIGHT2=21
WIDTH=60


function integer11(){
./lab_int11.sh
}


function integer22(){
./lab_int22.sh
}

function tabla(){
./lab_tab.sh

}

function sensores(){
      ./sensor.sh;
}

function procesos(){
 ps -ef | grep demon | pv -n 2>&1 >out | dialog --backtitle "Demonios
DHT11 & DHT22" --no-shadow --no-kill --title "Visualizacion procesos"
--begin 5 22 --tailbox out 20 90
 rm -f out
}

function todo(){
       dialog --backtitle "Demonios DHT11 & DHT22" --title
"Incializando" --infobox "\n\nInicializando todos los procesos. Espere
a que finalize" 7 70
       ./todos.sh
       sleep 4
       ps -ef | grep _demon | pv -n 2>&1 >out | dialog --backtitle
"Demonios DHT11 & DHT22" --no-shadow --no-kill --title "Visualizacion
procesos" --begin 5 22 --tailbox out 20 90
      dialog --title "[ MENU INICIAL PROYECTO ]" \
                                              --backtitle "Menu Proyecto
SNMP con sensores DHT11 & DHT22" \
                                              --yesno "\nQuieres iniciar
la notificacion mediante LED de los humbrales preestalecidos de las
Traps?" 7 60
```

```
                              # Get exit status
                              # 0 means user hit [yes] button.
                              # 1 means user hit [no] button.
                              # 255 means user hit [Esc] key.
                                          response=$?
                                          case $response in
                                          0) clear
                                       ./p.sh >/dev/null 2>&1 &
                                       ./p2.sh >/dev/null 2>&1 &
                                                ;;
                                           1) ;;
                                           255) ;;
                                      esac
}

# TRAPS
#
function traps(){
     dialog --backtitle "Demonio Traps DHT11" --title "Inicializando
TRAPS DHT11" --ok-label "Continuar" --msgbox "\nEspera a que finalize
el proceso de inicializacion" 7 70
     ./traps.sh
     sleep 1
     ps -ef | grep demon | pv -n 2>&1 >out | dialog --backtitle
"Demonios DHT11 & DHT22" --no-shadow --no-kill --title "Visualizacion
procesos" --begin 5 22 --tailbox out 20 90
     dialog --title "[ MENU INICIAL PROYECTO ]" \
                                          --backtitle "Menu Proyecto
SNMP con sensores DHT11 & DHT22" \
                                          --yesno "\nQuieres iniciar
la notificacion mediante LED de los humbrales preestalecidos?" 7 60

                                    # Get exit status
                                    # 0 means user hit [yes] button.
                                    # 1 means user hit [no] button.
                                    # 255 means user hit [Esc] key.
                                          response=$?
                                          case $response in
                                          0) clear
                                            ./p.sh >/dev/null 2>&1 &
                                           ./p2.sh >/dev/null 2>&1 &
                                                ;;
                                           1) ;;
                                           255) ;;
                                      esac
}

function ip(){
TEXTO="/tmp/ip.txt"
dialog --title "Nueva IP" --backtitle "Menu Proyecto SNMP con
sensores DHT11 & DHT22" \
--inputbox "Introduce nueva IP: " 8 60 2>"${TEXTO}"
sel=$?
IP=`cat /tmp/ip.txt`
case $sel in
```

```
0) dialog --backtitle "Menu Proyecto SNMP con sensores DHT11 &
DHT22" --title "Nueva IP" --msgbox "\nLa nueva IP introducida es
$IP" 7 70;;
     1) clear; exit 0;;
esac
#rm -f /tmp/texto.txt
}


# LED
function led(){
      #./sensorPractica.sh
     ./LED2.sh
}
##
# Historico
function historico(){
      sudo ./sensorhistorico3.sh
}

function finalizar(){
      dialog --backtitle "Demonios DHT11 & DHT22" --title
"Finalizar" --infobox "\nFinalizando los demonios iniciados. Espere
a que finalize" 7 70
      sleep 1
      sudo ./cerrar.sh >/dev/null 2>&1 &
      sudo ./fin_p.sh >/dev/null 2>&1 &
      sudo ./fin_p2.sh >/dev/null 2>&1 &
     sleep 6
     ps -ef | grep _demon | pv -n 2>&1 >out | dialog --backtitle
"Visualizacion Procesos Demonios" --no-shadow --no-kill --title
"Visualizacion procesos" --begin 5 22 --tailbox out 20 90
}

function configurar(){

./sustituir.sh

}

TEXTO="/tmp/ip.txt"
dialog --title "IP Raspberry Pi" --backtitle "Menu Proyecto SNMP con
sensores DHT11 & DHT22" \
--inputbox "Introduce la IP: " 8 60 2>"${TEXTO}"
sel=$?
IP=`cat /tmp/ip.txt`
case $sel in
     0) dialog --backtitle "Menu Proyecto SNMP con sensores DHT11 &
DHT22" --title "IP Raspberry Pi" --msgbox "\nLa IP introducida es
$IP" 7 70;;
     1) clear; exit 0;;
esac
#rm -f /tmp/texto.txt
```

```
while true; do
  exec 3>&1
  selection=$(dialog \
    --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" \
    --title "[ MENU LABORATORIO ]" \
    --clear \
     --ok-label "Seleccionar" \
    --cancel-label "Salir" \
    --menu "Utiliza las flechas para el desplazamiento, la primera\n\
letra de cada palabra como acceso rapido.\n\
Seleciona una opcion:" $HEIGHT2 $WIDTH 12 \
     "Sensores" "Ejecucion de los sensores" \
     "INTEGER_DHT11" "Visualizar Integers DHT11" \
    "INTEGER_DHT22" "Visualizar Integers DHT22" \
     "TABLA_DHT11" "Visualizar Tabla DHT11" \
     "TRAPS" "Inicializacion de las Traps" \
     "Todos" "Inicializa todos los demonios" \
     "Procesos" "Visualizacion de los procesos" \
     "Finalizar" "Finaliza todos los demonios" \
     "Configurar" "Configuracion Valores Traps" \
     "LED" "Inicializa demonio control LED" \
     "Historico" "Tabla Historico sensores" \
     "IP" "Volver a introducir IP" \
    2>&1 1>&3)
  exit_status=$?
  exec 3>&-
  case $exit_status in
    $DIALOG_CANCEL)
        rm -f /tmp/ip.txt
        dialog --title "[ MENU INICIAL PROYECTO ]" \
                                          --backtitle "Menu Proyecto
SNMP con sensores DHT11 & DHT22" \
                                          --yesno "\nQuieres
finalizar los demonios?" 7 60

                           # Get exit status
                           # 0 means user hit [yes] button.
                           # 1 means user hit [no] button.
                           # 255 means user hit [Esc] key.
                                          response=$?
                                          case $response in
                                               0) clear
                                                          dialog --
backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --title
"Finalizar" --infobox "\nFinalizando los demonios iniciados. Espere a
que finalize" 7 70
                                                sleep 1
                                    sudo ./cerrar.sh >/dev/null 2>&1 &
                                    sudo ./fin_p.sh >/dev/null 2>&1 &
                                    sudo ./fin_p2.sh >/dev/null 2>&1 &
                                    sleep 6
                                    clear
                                    exit;;
1) exit;;
                                               255) ;;
```

```
                                        esac;;
    $DIALOG_ESC)
       rm -f /tmp/ip.txt
        dialog --title "[ MENU INICIAL PROYECTO ]" \
                                              --backtitle "Menu Proyecto
SNMP con sensores DHT11 & DHT22" \
                                              --yesno "\nQuieres
finalizar los demonios?" 7 60

                                              # Get exit status
                                              # 0 means user hit [yes]
button.
                                              # 1 means user hit [no]
button.
                                              # 255 means user hit [Esc]
key.
                                              response=$?
                                              case $response in
                                                   0) clear
                                                                dialog --
backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --title
"Finalizar" --infobox "\nFinalizando los demonios iniciados. Espere a
que finalize" 7 70
                                                                sleep 1
                                                                sudo
./cerrar.sh >/dev/null 2>&1 &
                                                                sudo
./fin_p.sh >/dev/null 2>&1 &
                                                                sudo
./fin_p2.sh >/dev/null 2>&1 &
                                                               sleep 6
                                                               clear
                                                      exit;;
                                                   1) exit;;
                                                   255) ;;
                                              esac;;
  esac
  case $selection in
    0 ) clear
           rm -f /tmp/ip.txt;;
      Sensores) sensores;;
    INTEGER_DHT11 ) integer11 ;;
      INTEGER_DHT22 ) integer22 ;;
      TABLA_DHT11) tabla;;
      TRAPS) traps;;
      Todos) todo;;
      Configurar) configurar;;
      LED ) led;;
      Historico ) historico;;
      Procesos ) procesos;;
      Finalizar) finalizar;;
      IP) ip;;
  esac
done
```

Figura A4. Código Script Menú Demonios

A.5 Script Visualización MIB

```
#!/bin/bash

BACKTITLE="Menu Visualizacion MIB"
INPUT=/tmp/menu.sh.$$

ret=0

while [ $ret -eq 0 ]
do
    dialog --title "Visualizacion MIB" \
        --backtitle "$BACKTITLE"  \
           --cancel-label "Salir" \
        --menu "Utiliza las flechas para el desplazamiento o el
numero 1-2 para seleccionar una opcion.\n\
Seleciona una opcion:" 12 60 2 \
        1 "Visualizar MIB formato ASN.1" \
        2 "Visualizar ARBOL MIB" \
        2>"${INPUT}"

    ret=$?
    option=$(<"${INPUT}")

    if [ $ret -eq 0 ]
    then
        if [ $option -eq 1 ]
        then
            dialog --title "Visualizacion MIB ASN.1" \
                --backtitle "$BACKTITLE" \
                --textbox ~/Escritorio/proyecto2/MIBs/mib 18 100
        elif [ $option -eq 2 ]
        then
                echo " " > /tmp/mib.txt
                echo -e "\t\tARBOL MIB comando SNMPTRANSLATE" >>
/tmp/mib.txt
                echo " " >> /tmp/mib.txt
                echo "snmptranslate -Tp -IR SENSOR-TEMPERATURA-
MIB::sensorTemperaturaMIB" >> /tmp/mib.txt
                echo " " >> /tmp/mib.txt
                snmptranslate -Tp -IR SENSOR-TEMPERATURA-
MIB::sensorTemperaturaMIB >> /tmp/mib.txt
             dialog --title "Visualizacion Arbol MIB " \
                     --backtitle "$BACKTITLE" \
                    --textbox /tmp/mib.txt 18 70
         fi
           rm -f /tmp/mib.txt
    fi
done
```

Figura A.5 Script visualización MIB

A.6 Script Configuración

```
#!/bin/bash

#echo "Editando el archivo de configuracion..."

#nano /etc/snmp/snmpd.conf


#!/bin/bash
BACKTITLE="Menu configuracion/visualizacion fichero snmpd"
FILENAME="/etc/snmp/snmpd.conf"

touch $FILENAME

INPUT=/tmp/menu.sh.$$

ret=0

while [ $ret -eq 0 ]
do
    dialog --title "Configuracion snmpd" \
        --backtitle "$BACKTITLE"  \
           --cancel-label "Salir" \
        --menu "Utiliza las flechas para el desplazamiento, la
primera\n\
letra de cada palabra como acceso rapido o el numero 1-2  \n\
para seleccionar una opcion.\n\
Seleciona una opcion:" 12 60 2 \
        1 "Visualizar Fichero" \
        2 "Editar Fichero " \
        2>"${INPUT}"

    ret=$?
    option=$(<"${INPUT}")

    if [ $ret -eq 0 ]
    then
        if [ $option -eq 1 ]
        then
            dialog --title "Visualizacion Fichero snmpd" \
                --backtitle "$BACKTITLE" \
                --textbox $FILENAME 18 70
        elif [ $option -eq 2 ]
        then
                nano /etc/snmp/snmpd.conf
                sudo /etc/init.d/snmpd restart >/dev/null 2>&1
        fi
    fi
done
```

Figura A6. Código Script Configuración

A.7 Script Integers Sensor DHT11

```
#!/bin/bash

DIALOG_CANCEL=1
DIALOG_ESC=255
HEIGHT=14
WIDTH=60

IP=`cat /tmp/ip.txt`

while true; do
      exec 3>&1
       selection=$(dialog \
    --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" \
    --title "[ MENU LABORATORIO ]" \
    --clear \
     --ok-label "Seleccionar" \
    --cancel-label "Salir" \
    --menu "Utiliza las flechas para el desplazamiento, la
primera\n\
letra de cada palabra como acceso rapido o del numero 1-3 para
seleccionar una opcion.\n\
Seleciona una opcion:" $HEIGHT $WIDTH 3 \
    "INICIALIZAR" "Iniciar demonios DHT11" \
     "CERRAR" "Finalizar demonios DHT11" \
     "VISUALIZAR" "Visualizar valores DHT11" \
    2>&1 1>&3)
  exit_status=$?
  exec 3>&-
  case $exit_status in
    $DIALOG_CANCEL)
      clear
       exit 0;;
    $DIALOG_ESC)
      clear
       exit 1;;
  esac
  case $selection in
    0 )
      clear;;
    CERRAR ) dialog --backtitle "Demonios Integer DHT11" --title
"Finalizar" --infobox "\nFinalizando los demonios DHT11. Espere a
que finalize" 7 70
      sleep 1
                  kill $(ps aux | grep 'temperaturaDHT11_demon') 2>
/dev/null
                  kill $(ps aux | grep 'humedadDHT11_demon') 2>
/dev/null
      sleep 2
```

```
ps -ef | grep _demon | pv -n 2>&1 >out | dialog --backtitle "Demonios
Integer DHT11" --no-shadow --no-kill --title "Visualizacion procesos"
--begin 4 22 --tailbox out 16 90
                rm -f out;;

     INICIALIZAR ) dialog --backtitle "Demonios Integer DHT11" --
title "Inicializando INTEGERS" --ok-label "Continuar" --infobox
"\nEspera a que finalize el proceso de inicializacion" 7 70
                          ./integersDHT11.sh
                          sleep 1
                        ps -ef | grep demon | pv -n 2>&1 >out |
dialog --backtitle "Demonios DHT11 & DHT22" --no-shadow --no-kill --
title "Visualizacion procesos" --begin 4 22 --tailbox out 16 90
                          rm -f out;;
     VISUALIZAR) echo " " > temp.txt
                    echo -e "\t\t\t\t\tTEMPERATURA DHT11" >> temp.txt
                     echo -e "snmpget -v 2c -c public $IP SENSOR-
TEMPERATURA-MIB::temperaturaDHT11.0" >> temp.txt
                     echo -e "snmpget -v 2c -c public $IP
.1.3.6.1.4.1.8072.2.333.1.3.0" >> temp.txt
                     snmpget -v 2c -c public $IP SENSOR-TEMPERATURA-
MIB::temperaturaDHT11.0 >> temp.txt
                     echo " " >> temp.txt
                     echo " " >> temp.txt
                     echo -e "\t\t\t\t\tHUMEDAD DHT11" >> temp.txt
                     echo -e "snmpget -v 2c -c public $IP SENSOR-
TEMPERATURA-MIB::humedadDHT11.0" >> temp.txt
                    echo -e "snmpget -v 2c -c public $IP
.1.3.6.1.4.1.8072.2.333.1.4.0" >> temp.txt
                     snmpget -v 2c -c public $IP SENSOR-TEMPERATURA-
MIB::humedadDHT11.0 >> temp.txt
                     echo " " >> temp.txt
                     echo " " >> temp.txt
                     echo -e "\t\t\t\t\tPresiona Salir Para
Continuar." >> temp.txt
                     dialog --title "Valores MIB snmpget DHT11" \
                        --backtitle "Visualizacion Datos MIB SNMP" \
                      --textbox temp.txt 18 110
                               rm -f temp.txt  ;;
                          esac
done
}
```

Figura A7. Código Scripts Aplicación Integer DHT11

A.8 Script Integers Sensor DHT22

```
#!/bin/bash

DIALOG_CANCEL=1
DIALOG_ESC=255
HEIGHT=14
WIDTH=60

IP=`cat /tmp/ip.txt`

while true; do
     exec 3>&1
      selection=$(dialog \
    --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" \
    --title "[ MENU LABORATORIO ]" \
    --clear \
     --ok-label "Seleccionar" \
    --cancel-label "Salir" \
    --menu "Utiliza las flechas para el desplazamiento, la
primera\n\
letra de cada palabra como acceso rapido o del numero 1-3 para
seleccionar una opcion.\n\
Seleciona una opcion:" $HEIGHT $WIDTH 3 \
     "INICIALIZAR" "Iniciar demonios DHT22" \
    "CERRAR" "Finalizar demonios DHT22" \
    "VISUALIZAR" "Visualizar valores DHT22" \
    2>&1 1>&3)
  exit_status=$?
  exec 3>&-
  case $exit_status in
    $DIALOG_CANCEL)
      clear
       exit 0;;
    $DIALOG_ESC)
      clear
       exit 1;;
  esac
  case $selection in
    0 )
      clear;;
    CERRAR ) dialog --backtitle "Demonios Integer DHT22" --title
"Finalizar" --infobox "\nFinalizando los demonios DHT22. Espere a
que finalize" 7 70
      sleep 1
                kill $(ps aux | grep 'temperaturaDHT22_demon') 2>
/dev/null
                kill $(ps aux | grep 'humedadDHT22_demon') 2>
/dev/null
     sleep 2
```

```
ps -ef | grep _demon | pv -n 2>&1 >out | dialog --backtitle
"Visualizacion Procesos Demonios" --no-shadow --no-kill --title
"Visualizacion procesos" --begin 4 22 --tailbox out 16 90
     rm -f out;;

     INICIALIZAR ) dialog --backtitle "Demonios Integer DHT22" --
title "Inicializando INTEGERS" --ok-label "Continuar" --infobox
"\nEspera a que finalize el proceso de inicializacion" 7 70
                          ./integersDHT22.sh
                          sleep 1
                        ps -ef | grep demon | pv -n 2>&1 >out |
dialog --backtitle "Demonios DHT11 & DHT22" --no-shadow --no-kill --
title "Visualizacion procesos" --begin 4 22 --tailbox out 16 90
                                rm -f out;;
     VISUALIZAR) echo " " > temp.txt
                      echo -e "\t\t\t\t\tTEMPERATURA DHT22" >>
temp.txt
                      echo -e "snmpget -v 2c -c public $IP SENSOR-
TEMPERATURA-MIB::temperaturaDHT22.0" >> temp.txt
                      echo -e "snmpget -v 2c -c public $IP
.1.3.6.1.4.1.8072.2.333.1.5.0" >> temp.txt
                      snmpget -v 2c -c public $IP SENSOR-TEMPERATURA-
MIB::temperaturaDHT22.0 >> temp.txt
                      echo " " >> temp.txt
                      echo " " >> temp.txt
                      echo -e "\t\t\t\t\tHUMEDAD DHT22" >> temp.txt

                    echo -e "snmpget -v 2c -c public $IP SENSOR-
TEMPERATURA-MIB::humedadDHT22.0" >> temp.txt
                   echo -e "snmpget -v 2c -c public $IP
.1.3.6.1.4.1.8072.2.333.1.6.0" >> temp.txt
                      snmpget -v 2c -c public $IP SENSOR-TEMPERATURA-
MIB::humedadDHT22.0 >> temp.txt
                      echo " " >> temp.txt
                      echo " " >> temp.txt
                      echo -e "Presiona Salir Para Continuar." >>
temp.txt
                      dialog --title "Valores MIB snmpget DHT11" \
                         --backtitle "Visualizacion Datos MIB SNMP" \
                       --textbox temp.txt 18 110
                                rm -f temp.txt  ;;
               esac
done
```

Figura A8. Código Script Aplicación Integer DHT22

A.9 Script Tabla Sensor

```
#!/bin/bash

DIALOG_CANCEL=1
DIALOG_ESC=255
HEIGHT=14
WIDTH=60

IP=`cat /tmp/ip.txt`

while true; do
      exec 3>&1
       selection=$(dialog \
    --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" \
    --title "[ MENU LABORATORIO ]" \
    --clear \
     --ok-label "Seleccionar" \
    --cancel-label "Salir" \
    --menu "Utiliza las flechas para el desplazamiento, la primera\n\
letra de cada palabra como acceso rapido o del numero 1-3 para
seleccionar una opcion.\n\
Seleciona una opcion:" $HEIGHT $WIDTH 3 \
      "INICIALIZAR" "Iniciar demonio Tabla DHT11" \
    "CERRAR" "Finalizar demonio Tabla DHT11" \
    "VISUALIZAR" "Visualizar valores Tabla DHT11" \
    2>&1 1>&3)
  exit_status=$?
  exec 3>&-
  case $exit_status in
    $DIALOG_CANCEL)
      clear
       exit;;
    $DIALOG_ESC)
      clear
       exit 1;;
  esac
  case $selection in
    0 )
      clear;;
    CERRAR ) dialog --backtitle "Demonios Tabla DHT11" --title
"Finalizar" --infobox "\nFinalizando el demonio tabla DHT11. Espere a
que finalize" 7 70
      sleep 1
kill $(ps aux | grep 'tabla_sensor_demon') 2> /dev/null
       sleep 2
     ps -ef | grep _demon | pv -n 2>&1 >out | dialog --backtitle
"Visualizacion Procesos Demonios" --no-shadow --no-kill --title
"Visualizacion procesos" --begin 4 22 --tailbox out 16 90
                                rm -f out;;

INICIALIZAR ) dialog --backtitle "Demonios Tabla DHT11" --title
"Inicializando TABLA" --ok-label "Continuar" --infobox
```

```
"\nEspera a que finalize el proceso de inicializacion" 7 70
                            ./tabla.sh
                            sleep 1
                           ps -ef | grep demon | pv -n 2>&1 >out |
dialog --backtitle "Demonios Tabla" --no-shadow --no-kill --title
"Visualizacion procesos" --begin 4 22 --tailbox out 16 90
                            rm -f out;;
     VISUALIZAR) echo -e "\t\t\t\t\tTABLA VALORES SENSOR DHT11" >
temp.txt
                    echo " " >> temp.txt
                     echo -e "snmptable -v 2c -c public $IP SENSOR-
TEMPERATURA-MIB::sensorTable" >> temp.txt
                     echo -e "snmptable -v 2c -c public $IP
.1.3.6.1.4.1.8072.2.333.1.2" >> temp.txt
                     echo " " >> temp.txt
                     snmptable -v 2c -c public $IP SENSOR-
TEMPERATURA-MIB::sensorTable >> temp.txt
                     echo " " >> temp.txt
                     echo -e "\t\t\t\t\tPresiona Salir Para
Continuar." >> temp.txt
                     dialog --title "Valores MIB snmptable DHT11" \
                        --backtitle "Visualizacion Datos MIB SNMP" \
                      --textbox temp.txt 18 110
                                rm -f temp.txt  ;;

                esac
done
```

Figura A9. Código Script Tabla Sensor

A.10 Script Traps

```
#!/bin/bash

cd ~/Escritorio/proyecto2/Demonios/TemperaturaDHT11/
./trap_temperaturaDHT11_demon >/dev/null 2>&1 &
sleep 1
cd ~/Escritorio/proyecto2/Demonios/HumedadDHT11/
./trap_humedadDHT11_demon >/dev/null 2>&1 &
```

Figura A10. Script Traps

A.11 Script Inicializar Todo

```
#!/bin/bash

cd ~/Escritorio/proyecto2/Demonios/TemperaturaDHT11/
./temperaturaDHT11_demon >/dev/null 2>&1 &


cd ~/Escritorio/proyecto2/Demonios/HumedadDHT11/
./humedadDHT11_demon >/dev/null 2>&1 &


cd ~/Escritorio/proyecto2/Demonios/TemperaturaDHT22/
./temperaturaDHT22_demon >/dev/null 2>&1 &

cd ~/Escritorio/proyecto2/Demonios/HumedadDHT22/
./humedadDHT22_demon >/dev/null 2>&1 &


cd ~/Escritorio/proyecto2/Demonios/TemperaturaDHT11/
./trap_temperaturaDHT11_demon >/dev/null 2>&1 &


cd ~/Escritorio/proyecto2/Demonios/HumedadDHT11/
./trap_humedadDHT11_demon >/dev/null 2>&1 &

cd ~/Escritorio/proyecto2/Demonios/TablaSensor
./tabla_sensor_demon >/dev/null 2>&1 &

cd ~/Escritorio/proyecto2/Demonios/Tablav2/
./tabla_historico_demon >/dev/null 2>&1 &

cd ~/Escritorio/proyecto2/Demonios/TablaControl/
./tabla_historicoControl_demon >/dev/null 2>&1 &

cd ~/Escritorio/proyecto2/Demonios/IntegerPractica
./sensorPractica_demon >/dev/null 2>&1 &

sleep 1
```

Figura A.11 Código Script Ejecución Completa Proyecto

A.12 Script Finalizar Demonios

```
#!/bin/bash

#echo "Cerrando los procesos de los demonios"

           kill $(ps aux | grep 'sensorLectura_daemon') 2> /dev/null
           kill $(ps aux | grep 'temperaturaDHT11_demon') 2> /dev/null
           kill $(ps aux | grep 'humedadDHT11_demon') 2> /dev/null
           kill $(ps aux | grep 'tabla_sensor_demon' | 2> /dev/null
     kill $(ps aux | grep 'trap_temperaturaDHT11_demon') 2> /dev/null
           kill $(ps aux | grep 'trap_humedadDHT11_demon') 2> /dev/null
           kill $(ps aux | grep 'sensorPractica_demon') 2> /dev/null
          kill $(ps aux | grep 'temperaturaDHT22_demon') 2> /dev/null
           kill $(ps aux | grep 'humedadDHT22_demon') 2> /dev/null
            kill $(ps aux | grep 'tabla_historico_demon') 2> /dev/null
       kill $(ps aux | grep 'tabla_historicoControl_demon') 2> /dev/null
                          sleep 2
                          clear
```

Figura A.12 Código Script Finalizar Procesos

A.13 Script sensorPractica (LED)

```
#!/bin/bash


cd ~/Escritorio/proyecto2/Demonios/IntegerPractica/
sudo ./sensorPractica_demon >/dev/null 2>&1 &

sleep 2

DIALOG_CANCEL=1
DIALOG_ESC=255
HEIGHT=14
WIDTH=60

# Visualizacion de los valores de los sensores
#
##########################################################################
##########################################################################
#####
function fija(){

value2="/tmp/texto.txt"
dialog --title "LED Fijo" --backtitle "Menu LED Practica" \
--inputbox "Introduce los segundos de luz fija: " 8 60 2>"${value2}"
sel=$?
value=`cat /tmp/texto.txt`
case $sel in
     0) dialog --backtitle "Menu LED Practica" --title "Led Fijo" --
msgbox "\nLos segundos de LED fijo introducidos son: $value" 7 75;;
     1) clear; exit 0;;
esac
rm -f /tmp/texto.txt
echo -e "snmpset -v 2c -c private $IP .1.3.6.1.4.1.8072.2.333.1.1.0 i
$value " > temp.txt
snmpset -v 2c -c private $IP .1.3.6.1.4.1.8072.2.333.1.1.0 i $value
>> temp.txt
echo -e "snmpget -v 2c -c public $IP .1.3.6.1.4.1.8072.2.333.1.1.0"
>> temp.txt
snmpget -v 2c -c public $IP .1.3.6.1.4.1.8072.2.333.1.1.0 >> temp.txt
echo " " >> temp.txt
echo "Presiona Salir Para Continuar y espere a que finalize el
proceso" >> temp.txt
dialog --title "Valores MIB snmpset & snmpget" \
                --backtitle "Menu LED Practica" \
                --textbox temp.txt 10 80
rm -f temp.txt


value3="`snmpget -v 2c -c private $IP .1.3.6.1.4.1.8072.2.333.1.1.0
`"
value3="${value3:60}"
#Encendido Fijo LED
```

```
echo 26 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio26/direction
echo 1 > /sys/class/gpio/gpio26/value
sleep $value3
echo 0 > /sys/class/gpio/gpio26/value
echo 26 > /sys/class/gpio/unexport;
sleep 1;
clear;
echo;
}

##########################################################################
#########################################################

function parpadeo(){

#### Cambiar tamanho terminal ##########
value2="/tmp/texto.txt"
dialog --title "LED Parpadeo" --backtitle "Menu LED Practica" \
--inputbox "Introduce el numero de parpadeos: " 8 60 2>"${value2}"
sel=$?
value=`cat /tmp/texto.txt`
case $sel in
     0) dialog --backtitle "Menu LED Practica" --title "Parpadeo Led"
--msgbox "\nEl numero de parpadeos seran: $value" 7 75;;
     1) clear; exit 0;;
esac
rm -f /tmp/texto.txt
echo -e "snmpset -v 2c -c private $IP .1.3.6.1.4.1.8072.2.333.1.1.0 i
$value " > temp.txt
snmpset -v 2c -c private $IP .1.3.6.1.4.1.8072.2.333.1.1.0 i $value >>
temp.txt
echo -e "snmpget -v 2c -c public $IP .1.3.6.1.4.1.8072.2.333.1.1.0" >>
temp.txt
snmpget -v 2c -c public $IP .1.3.6.1.4.1.8072.2.333.1.1.0 >> temp.txt
echo " " >> temp.txt
echo "Presiona Salir Para Continuar y espere a que finalize el
proceso" >> temp.txt
dialog --title "Valores MIB snmpset & snmpget" \
                --backtitle "Menu LED Practica" \
                --textbox temp.txt 10 80
rm -f temp.txt


i="0"
value3="`snmpget -v 2c -c private $IP .1.3.6.1.4.1.8072.2.333.1.1.0 `"
value3="${value3:60}"
while [ $i -lt $value3 ] #SEGUNDOS PARPADEO
                                do
                                     #Parpadeo LED
                           echo 26 > /sys/class/gpio/export
                           echo out > /sys/class/gpio/gpio26/direction
                           echo 1 > /sys/class/gpio/gpio26/value
```

```
sleep 1
                                        echo 0 >
/sys/class/gpio/gpio26/value
                                        sleep 1
                                        echo 26 >
/sys/class/gpio/unexport
                                        i=$[$i+1]
                                    done;
clear;
}

#####
function ip(){
TEXTO="/tmp/texto.txt"
dialog --title "Nueva IP" --backtitle "Menu LED Practica" \
--inputbox "Introduce nueva IP: " 8 60 2>"${TEXTO}"
sel=$?
IP=`cat /tmp/texto.txt`
case $sel in
     0) dialog --backtitle "Menu LED Practica" --title "Nueva IP" -
-msgbox "\nLa nueva IP introducida es $IP" 7 70;;
     1) clear; exit 0;;
esac
rm -f /tmp/texto.txt
}

IP=`cat /tmp/ip.txt`


#####################################################################
###

#........................... MENU
.................................#

#####################################################################
##
while true; do
  exec 3>&1
  selection=$(dialog \
    --backtitle "Menu LED Practica" \
    --title "[ MENU LED PRACTICA ]" \
     --ok-label "Seleccionar" \
    --cancel-label "Salir" \
    --menu "Utiliza las flechas para el desplazamiento, la

primera\n\
letra de cada palabra como acceso rapido o el numero 1-3  \n\
para seleccionar una opcion.\n\
Seleciona una opcion:" $HEIGHT $WIDTH 3 \
    "Fija" "Luz FIJA LED" \
    "Parpadeo" "PARPADEO Luz LED" \
    "IP" "Introducir IP" \
      2>&1 1>&3)
```

```
exit_status=$?
  exec 3>&-
  case $exit_status in
    $DIALOG_CANCEL)
      clear
       kill $(ps aux | grep 'sensorPractica_demon') &> /dev/null
      #echo "Program terminated."
       clear
      exit
      ;;
    $DIALOG_ESC)
      clear
       kill $(ps aux | grep 'sensorPractica_demon') &> /dev/null
       clear
      #echo "Program aborted." >&2
      exit 1
      ;;
  esac
  case $selection in
    0 )
      clear
      #echo "Program terminated."
      ;;
     Fija ) fija ;;
     Parpadeo ) parpadeo ;;
     IP ) ip ;;
     esac
done
kill $(ps aux | grep 'sensorPractica_demon') &> /dev/null #Cerrar
proceso
```

Figura A.13 Código Script sensorPractica

A.14 Visualización Procesos Proyecto

```
function procesos(){
 ps -ef | grep demon | pv -n 2>&1 >out | dialog --backtitle "Demonios
DHT11 & DHT22" --no-shadow --no-kill --title "Visualizacion procesos" --
begin 5 22 --tailbox out 20 90
 rm -f out
}
```

Figura A.14 Fragmento código visualización procesos proyecto

A.15 Script Tabla Históricos

```
kill $(ps aux | grep 'tabla_historico_demon') &> /dev/null
kill $(ps aux | grep 'tabla_historicoControl_demon') &> /dev/null
#Cerrar proceso

cd ~/Escritorio/proyecto2/Demonios/Tablav2/
./tabla_historico_demon >/dev/null 2>&1 &

cd ~/Escritorio/proyecto2/Demonios/TablaControl/
./tabla_historicoControl_demon >/dev/null 2>&1 &


DIALOG_CANCEL=1
DIALOG_ESC=255
HEIGHT=18
WIDTH=60


# Visualizacion de los valores de los sensores
#
function borrar(){
      dialog --backtitle "Menu Historico Sensores" --title "Valores
Tabla" --infobox "\nBorrando Valores previamente almacenados" 7 70
      rm -f ~/Escritorio/proyecto2/Demonios/Tablav2/historico.txt
      rm -f
~/Escritorio/proyecto2/Demonios/TablaControl/historicoControlTable.t
xt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/max1.txt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/max2.txt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/max3.txt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/max4.txt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/Valor1.txt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/Valor2.txt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/Valor3.txt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/Valor4.txt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/sondeo1.txt
      rm -f ~/Escritorio/proyecto2/Demonios/TablaControl/sondeo2.txt
      kill $(ps aux | grep 'tabla_historico_demon') >/dev/null 2>&1 &
      kill $(ps aux | grep 'tabla_historicoControl_demon') >/dev/null
2>&1 &
      sleep 1
      cd ~/Escritorio/proyecto2/Demonios/Tablav2/
      ./tabla_historico_demon >/dev/null 2>&1 &
      cd ~/Escritorio/proyecto2/Demonios/TablaControl/
      ./tabla_historicoControl_demon >/dev/null 2>&1 &
      sleep 1
      dialog --backtitle "Menu Historico Sensores" --title "Valores
Tabla" --msgbox "\nValores previamente almacenados borrados con
exito !!!" 7 70
}
```

```
#
#
function lectura1(){

                                COUNT="/tmp/count.txt"
                                dialog --title "Contador" --backtitle
"Menu Historico Sensores" \
                                --inputbox "Introduce el numero de
Lecturas : " 8 60 2>"${COUNT}"
                                sel=$?
                                COUNT2=`cat /tmp/count.txt`
                                case $sel in
                                0) dialog --backtitle "Menu Historico
Sensores" --title "Contador" --msgbox "\nEl numero de lecturas
seleccionadas son $COUNT2" 7 70;;
                                1) clear; exit 0;;
                                esac
                                rm -f /tmp/count.txt

     ##################################################################

                dialog --backtitle "Menu Historico Sensores" --title
"Leyendo Valores" --ok-label "Continuar" --msgbox "\nEspera a que
finalize el proceso de lectura" 7 70


##################################################################
                                i=0
                                while [ $i -lt $COUNT2 ]
                                do
                                     RESULTS="`python
~/Escritorio/proyecto2/Adafruit_Python_DHT/examples/AdafruitDHT.py 11
4 | grep Temp `"
                                     RESULTS2="`python
~/Escritorio/proyecto2/Adafruit_Python_DHT/examples/AdafruitDHT.py 22
17 | grep Temp `"
                                     TEMP="`echo ${RESULTS} | awk
'{print $1}'`"
                                     TEMP="${TEMP:5:-1}"
                                     HUM="`echo ${RESULTS} | awk
'{print $2}'`"
                                     HUM="${HUM:9:-1}"
                                     TEMP2="`echo ${RESULTS2} | awk
'{print $1}'`"
                                     TEMP2="${TEMP2:5:-1}"
                                     HUM2="`echo ${RESULTS2} | awk
'{print $2}'`"
                                     HUM2="${HUM2:9:-1}"

                                     DATE="`date \"+%Y-%m-%d
%H:%M:%S\"`"
                                     DATE="${DATE:11}"
```

```
echo "TemperaturaDHT11 ${TEMP} ${DATE}" >>
~/Escritorio/proyecto2/Demonios/Tablav2/historico.txt;
                                    echo "HumedadDHT11 ${HUM} ${DATE}" >>
~/Escritorio/proyecto2/Demonios/Tablav2/historico.txt;
                                    echo "TemperaturaDHT22 ${TEMP2}
${DATE}" >> ~/Escritorio/proyecto2/Demonios/Tablav2/historico.txt;
                                    echo "HumedadDHT22 ${HUM2} ${DATE}"
>> ~/Escritorio/proyecto2/Demonios/Tablav2/historico.txt;
                                    sleep 5
                                    i=$[$i+1]
                                    done
}
#####
function ver(){
snmptable -v 2c -c public $IP SENSOR-TEMPERATURA-MIB::historicoTable
> /tmp/historico_ver.txt

FILENAME="/tmp/historico_ver.txt"

dialog --title "Visualizacion Tabla Historicos" \
                --backtitle "Menu Historico Sensores" \
                --textbox $FILENAME 17 60
rm -f /tmp/historico_ver.txt
}
#
#####
function recorrer(){

#### Cambiar tamanho terminal ##########
snmpwalk -v 2c -c public $IP SENSOR-TEMPERATURA-MIB::historicoTable >
/tmp/historico_recorrer.txt
#printf '\e[8;24;132t'
FILENAME="/tmp/historico_recorrer.txt"

dialog --title "Visualizacion Tabla SNMP historicos" \
                --backtitle "Menu Historico Sensores" \
                --textbox $FILENAME 17 100
rm -f /tmp/historico_recorrer.txt
}

#####
function ip(){
TEXTO="/tmp/texto.txt"
dialog --title "Nueva IP" --backtitle "Menu LED Practica" \
--inputbox "Introduce nueva IP: " 8 60 2>"${TEXTO}"
sel=$?
IP=`cat /tmp/texto.txt`
case $sel in
     0) dialog --backtitle "Menu Historico Sensores" --title "Nueva
IP" --msgbox "\nLa nueva IP introducida es $IP" 7 70;;
     1) clear; exit 0;;
esac
rm -f /tmp/texto.txt
}
```

```
function tabla_control(){

TEXTO="/tmp/sondeo1.txt"
dialog --title "Valor Sondeo" --backtitle "Menu Historico Sensores" \
--inputbox "Introduce Tiempo sondeo Temperatura " 8 60 2>"${TEXTO}"
sel=$?
SONDEO1=`cat /tmp/sondeo1.txt`
case $sel in
     0) dialog --backtitle "Menu Historico Sensores" --title
"SONDEO" --msgbox "\nEl valor de sondeo de temperatura introducido es
$SONDEO1" 7 70;;
     1) clear; exit 0;;
esac

#rm -f /tmp/sondeo1.txt

TEXTO="/tmp/sondeo2.txt"
dialog --title "Valor Sondeo" --backtitle "Menu Historico Sensores" \
--inputbox "Introduce Tiempo sondeo Humedad " 8 60 2>"${TEXTO}"
sel=$?
SONDEO2=`cat /tmp/sondeo2.txt`
case $sel in
     0) dialog --backtitle "Menu Historico Sensores" --title
"SONDEO" --msgbox "\nEl valor de sondeo de humedad introducido es
$SONDEO2" 7 70;;
     1) clear; exit 0;;
esac

#rm -f /tmp/sondeo2.txt

echo -e "${SONDEO1}" >
~/Escritorio/proyecto2/Demonios/TablaControl/sondeo1.txt;
echo -e "${SONDEO2}" >
~/Escritorio/proyecto2/Demonios/TablaControl/sondeo2.txt;

RESULTS="`python
~/Escritorio/proyecto2/Adafruit_Python_DHT/examples/AdafruitDHT.py 11
4 | grep Temp `"
RESULTS2="`python
~/Escritorio/proyecto2/Adafruit_Python_DHT/examples/AdafruitDHT.py 22
17 | grep Temp `"
TEMP="`echo ${RESULTS} | awk '{print $1}'`"
MAX1="${TEMP:5:-1}"
HUM="`echo ${RESULTS} | awk '{print $2}'`"
MAX2="${HUM:9:-1}"
TEMP2="`echo ${RESULTS2} | awk '{print $1}'`"
MAX3="${TEMP2:5:-1}"
HUM2="`echo ${RESULTS2} | awk '{print $2}'`"
MAX4="${HUM2:9:-1}"
```

```
echo -e "${MAX1}" >
~/Escritorio/proyecto2/Demonios/TablaControl/max1.txt;
echo -e "${MAX2}" >
~/Escritorio/proyecto2/Demonios/TablaControl/max2.txt;
echo -e "${MAX3}" >
~/Escritorio/proyecto2/Demonios/TablaControl/max3.txt;
echo -e "${MAX4}" >
~/Escritorio/proyecto2/Demonios/TablaControl/max4.txt;

echo -e "DHT11 TEMPERATURA ${SONDEO1} ${MAX1}" >
~/Escritorio/proyecto2/Demonios/TablaControl/historicoControlTable.txt
echo -e "DHT11 HUMEDAD ${SONDEO2} ${MAX2}" >>
~/Escritorio/proyecto2/Demonios/TablaControl/historicoControlTable.txt
echo -e "DHT11 TEMPERATURA ${SONDEO1} ${MAX3}" >>
~/Escritorio/proyecto2/Demonios/TablaControl/historicoControlTable.txt
echo -e "DHT11 HUMEDAD ${SONDEO2} ${MAX4}" >>
~/Escritorio/proyecto2/Demonios/TablaControl/historicoControlTable.txt

cd ~/Escritorio/proyecto2/scripts/
./t.sh &
cd ~/Escritorio/proyecto2/scripts/
./t2.sh &

dialog --backtitle "Menu Historico Sensores" --title "Leyendo Valores"
--msgbox "\nLeyendo Valores de forma continua...\n\n" 7 70
}

function ver_cont(){

snmptable -v 2c -c public $IP SENSOR-TEMPERATURA-
MIB::historicoControlTable > /tmp/historicoControl_ver.txt

FILENAME="/tmp/historicoControl_ver.txt"

dialog --title "Visualizacion Tabla Control Historicos" \
                --backtitle "Menu Historico Sensores" \
                --textbox $FILENAME 17 70
rm -f /tmp/historicoControl_ver.txt
}

function fin_lectura(){
kill $(ps aux | grep '/t.sh') >/dev/null 2>&1 &
kill $(ps aux | grep '/t2.sh') >/dev/null 2>&1 &
dialog --backtitle "Menu Historico Sensores" --title "Tabla Control" -
-msgbox "\nProceso lectura finalizado..." 7 70;
}

################ INICIO  -  Introducir IP ########################
IP=`cat /tmp/ip.txt`
# set infinite loop
#
while true; do
```

```
#printf '\e[8;24;80t'
  exec 3>&1
  selection=$(dialog \
     --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" \
     --title "[ MENU TABLA HISTORICO ]" \
     --clear \
     --ok-label "Seleccionar" \
    --cancel-label "Salir" \
     --menu "Utiliza las flechas para el desplazamiento, la
primera\n\
letra de cada palabra como acceso rapido o el numero 1-5  \n\
para seleccionar una opcion.\n\
Seleciona una opcion:" $HEIGHT $WIDTH 8 \
      "Borrar" "Borrado valores previos" \
      "Lectura" "Lectura TEMPORAL sensores" \
      "Lectura_Control" "Lectura CONTINUA sensores" \
      "Ver_Tabla" "Visualizar tabla (snmptable)" \
      "Recorrer_Tabla" "Recorrer tabla (snmpwalk)" \
      "Tabla_Control" "Visualizar Tabla Control" \
      "Finalizar_Lectura" "Finaliza Lectura Control" \
      "IP" "Volver a introducir IP" \
      2>&1 1>&3)
  exit_status=$?
  exec 3>&-
  case $exit_status in
    $DIALOG_CANCEL)
      clear
       fin_lectura
kill $(ps aux | grep 'tabla_historico_demon') &> /dev/null
kill $(ps aux | grep 'tabla_historicoControl_demon') &> /dev/null
       exit;;
    $DIALOG_ESC)
      clear
       fin_lectura
       kill $(ps aux | grep 'tabla_historico_demon') &> /dev/null
     kill $(ps aux | grep 'tabla_historicoControl_demon') &> /dev/null
        echo "Program aborted." >&2
      exit 1;;
  esac
  case $selection in
     0 ) fin_lectura
      clear;;
     Borrar ) borrar ;;
     Lectura ) lectura1 ;;
     Lectura_Control) tabla_control;;
     Ver_Tabla ) ver;;
     Recorrer_Tabla )recorrer;;
     Tabla_Control) ver_cont;;
     Finalizar_Lectura) fin_lectura;;
     IP ) ip;;
     esac
done
kill $(ps aux | grep 'tabla_historico_demon') &> /dev/null
kill $(ps aux | grep 'tabla_historicoControl_demon') &> /dev/null
```

Figura A.15 Menu Sensor Historico

A.16 Script Sustitución Valores Trap

```
#!/bin/bash

DIALOG_CANCEL=1
DIALOG_ESC=255
HEIGHT=12
WIDTH=60

##########################################################################
##########################################################################
##################################################
function temperatura(){

######### SI / NO #######  SUBMENU ######################
dialog --title "[ MENU CONFIGURACION TRAP ]" \
                                        --backtitle "Menu Proyecto
SNMP con sensores DHT11 & DHT22" \
                                        --yesno "\nEstas seguro
que quieres modificar el valor?" 7 60

                                        # Get exit status
                                        # 0 means user hit [yes]
button.
                                        # 1 means user hit [no]
button.
                                        # 255 means user hit [Esc]
key.
                                        response=$?
                                        case $response in
                                             0) clear
                                             #TRAP

     TEXTO="/tmp/trap.txt"
                                                  dialog --title
"Valor Trap Temperatura" --backtitle "Menu Proyecto SNMP con
sensores DHT11 & DHT22" \
                                                  --inputbox
"Introduce el nuevo valor: " 8 60 2>"${TEXTO}"
                                                  sel=$?
                                                  MAX=`cat
/tmp/trap.txt`
                                                       case $sel
in
                                                       0) dialog
--backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --title
"Valor Trap Temperatura" --msgbox "\nEl nuevo valor MAXIMO de la
trap Temperatura es $MAX" 7 70
sed -i "s|if(temp>.*|if(temp>$MAX){|"
~/Escritorio/proyecto2/Demonios/TemperaturaDHT11/trap_temperaturaDHT
11_demon.c
```

```
                                                                    let
"MAX+=1";

echo "${MAX}.0" >
~/Escritorio/proyecto2/Demonios/TemperaturaDHT11/maxT.txt;

sleep 1

dialog --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --
title "Valor Trap Temperatura" --infobox "\nNuevo valor introducido.
Compilando..." 7 70
                                                                    cd
~/Escritorio/proyecto2/Demonios/TemperaturaDHT11/

sudo make trap_temperaturaDHT11_demon >/dev/null 2>&1 &

sleep 3

dialog --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --
title "Valor Trap Temperatura" --msgbox "\nCompilando con exito" 7
70;;
                                                                   1)
clear; exit 0;;
                                                              esac
                                                              rm -f
/tmp/trap.txt;;
                                                         1) ;;
                                                         255) ;;
                                                  esac
}

##########################################################################
##########################################################################
################################################

function humedad(){

######### SI / NO #######  SUBMENU ########################
dialog --title "[ MENU CONFIGURACION TRAP ]" \
                                        --backtitle "Menu Proyecto
SNMP con sensores DHT11 & DHT22" \
                                        --yesno "\nEstas seguro
que quieres modificar el valor?" 7 60

                                  # Get exit status
                                  # 0 means user hit [yes] button.
                                  # 1 means user hit [no] button.
                                  # 255 means user hit [Esc] key.
                                        response=$?
                                        case $response in
                                             0) clear
#TRAP

     TEXTO="/tmp/trap.txt"
```

```
dialog --title "Valor Trap Humedad" --backtitle "Menu Proyecto SNMP
con sensores DHT11 & DHT22" \
                                                        --inputbox
"Introduce el nuevo valor: " 8 60 2>"${TEXTO}"
                                                        sel=$?
                                                        MAX=`cat
/tmp/trap.txt`
                                                        case $sel in
                                                             0) dialog -
-backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --title
"Valor Trap Humedad" --msgbox "\nEl nuevo valor MAXIMO de la trap
Humedad es $MAX" 7 70
                                                                sed -i
"s|if(hum>.*|if(hum>$MAX){|"
~/Escritorio/proyecto2/Demonios/HumedadDHT11/trap_humedadDHT11_demon.c
let "MAX+=1";
                                                                echo
"${MAX}.0" > ~/Escritorio/proyecto2/Demonios/HumedadDHT11/maxH.txt;
                                                                sleep 1
                                                                dialog --
backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --title
"Valor Trap Humedad" --infobox "\nNuevo valor introducido.
Compilando..." 7 70
                                                                cd
~/Escritorio/proyecto2/Demonios/HumedadDHT11/
                                                                sudo make
trap_humedadDHT11_demon >/dev/null 2>&1 &
                                                                sleep 3
                                                                dialog --
backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --title
"Valor Trap Humedad" --msgbox "\nCompilando con exito" 7 70;;
                                                                  1) clear;
exit 0;;
                                                        esac
                                                        rm -f
/tmp/trap.txt;;
                                                   1) ;;
                                                   255) ;;
                                              esac
}
#########################################################################
#########################################################################
##########################################

while true; do
  exec 3>&1
  selection=$(dialog \
    --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" \
    --title "[ MENU CONFIGURACION TRAP ]" \
    --clear \
     --ok-label "Seleccionar" \
    --cancel-label "Salir" \
    --menu "Utiliza las flechas para el desplazamiento, la primera\n\
letra de cada palabra como acceso rapido o el numero 1-2  \n\
para seleccionar una opcion.\n\
Seleciona una opcion:" $HEIGHT $WIDTH 2 \
```

```
"Temperatura" "Valor Maximo Temperatura" \
    "Humedad" "Valor Maximo Humedad" \
    2>&1 1>&3)
  exit_status=$?
  exec 3>&-
  case $exit_status in
    $DIALOG_CANCEL)
      clear
       exit;;
    $DIALOG_ESC)
      clear
       exit 1;;
  esac
  case $selection in
    0 )
      clear
      echo "Program terminated.";;
    Temperatura )
      temperatura ;;
    Humedad )
      humedad ;;
    esac
done
```

Figura A.16 Script Configuracion Traps

A.17 Script Backup

```
#!/bin/bash

# while-menu-dialog: a menu driven system information program

DIALOG_CANCEL=1
DIALOG_ESC=255
HEIGHT=12
WIDTH=60

function resturar(){
dialog --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --
title "Restaurar" --msgbox "\nRestaurando los ficheros originales" 7
70
dialog --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --
title "Restaurar" --infobox "\n\nEspera a que finalize el proceso" 7
70
tar zxf ~/Escritorio/proyecto2/scripts/BackupProyecto.tar.gz -C
~/Escritorio/
dialog --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" --
title "Restaurar" --msgbox "\nFicheros Restaurados !!!" 7 70
}


while true; do
  exec 3>&1
  selection=$(dialog \
    --backtitle "Menu Proyecto SNMP con sensores DHT11 & DHT22" \
    --title "[ MENU RESTAURAR PROYECTO ]" \
    --clear \
     --ok-label "Seleccionar" \
    --cancel-label "Salir" \
    --menu "Utiliza las flechas para el desplazamiento, la primera\n\
letra de cada palabra como acceso rapido o el numero 1-2  \n\
para seleccionar una opcion.\n\
Seleciona una opcion:" $HEIGHT $WIDTH 2 \
    "Restaurar" "Restablecer ficheros proyecto" \
    "Salir" "Mantener ficheros actuales" \
    2>&1 1>&3)
  exit_status=$?
  exec 3>&-
  case $exit_status in
    $DIALOG_CANCEL)
      clear
       exit;;
    $DIALOG_ESC)
      clear
      exit 1;;
  esac
  case $selection in
    0 )
      clear;;
```

```
    Restaurar ) dialog --title "[ MENU RESTAURAR PROYECTO ]" \
                              --backtitle "Menu Proyecto SNMP con
sensores DHT11 & DHT22" \
                            --yesno "\nEstas seguro que quieres
restaurar los ficheros?" 7 60
                                                  # Get exit status
                                                  # 0 means user hit [yes]
button.
                                                  # 1 means user hit [no]
button.
                                                  # 255 means user hit [Esc]
key.
                                                  response=$?
                                                  case $response in
                                                        0) clear
                                                           restaurar;;
                                                        1) ;;
                                                        255) ;;
                                                  esac ;;
      Salir ) exit;;
  esac
done
```

Figura A.17 Script Backup

A.18 Hardware Raspeberry Pi 3

- Procesador:

  Chipset Broadcom BCM2387.

  1,2 GHz de cuatro núcleos ARM Cortex-A53

- GPU

  Dual Core VideoCore IV ® Multimedia Co-procesador. Proporciona Open GL ES 2.0, OpenVG acelerado por hardware, y 1080p30 H.264 de alto perfil de decodificación.

  Capaz de 1 Gpixel / s, 1.5Gtexel / s o 24 GFLOPs con el filtrado de texturas y la infraestructura DMA

- RAM: 1GB LPDDR2.
- Conectividad

  Ethernet socket Ethernet 10/100 BaseT

  802.11 b / g / n LAN inalámbrica y Bluetooth 4.1

  Salida de vídeo

  - HDMI rev 1.3 y 1.4
  - RCA compuesto (PAL y NTSC)

  Salida de audio

  - jack de 3,5 mm de salida de audio, HDMI
  - USB 4 x Conector USB 2.0

  Conector GPIO

  - 40-clavijas de 2,54 mm (100 milésimas de pulgada) de expansión: 2x20 tira
  - Proporcionar 27 pines GPIO, así como 3,3 V, +5 V y GND líneas de suministro

  Conector de la cámara de 15 pines cámara MIPI interfaz en serie (CSI-2)

  Pantalla de visualización Conector de la interfaz de serie (DSI) Conector de 15 vías plana flex cable con dos carriles de datos y un carril de reloj

  Ranura de tarjeta de memoria Empuje / tire Micro SDIO

Figura A.18 Hardware-Especificaciones Raspberry Pi 3

A.19 MIB SENSOR-TEMPERATURA-MIB

```
SENSOR-TEMPERATURA-MIB DEFINITIONS ::= BEGIN


-- MIB para un TABLA para los valores de los sensores de temperatura y
humedad para dht11 y dht22.

-- Escrita por Javier Garcia Nuñez para PFG para UC.


-- IMPORTS: Define el lugar en el que esta situada la MIB.


IMPORTS

      netSnmpExamples                         FROM NET-SNMP-EXAMPLES-MIB

      OBJECT-TYPE, Integer32,

      MODULE-IDENTITY                          FROM SNMPv2-SMI

      MODULE-COMPLIANCE, OBJECT-GROUP         FROM SNMPv2-CONF;


-- Informacion de la MIB:

sensorTemperaturaMIB MODULE-IDENTITY

    LAST-UPDATED "201606150000Z"              -- 15 de junio de 2016, 00.00h

    ORGANIZATION "UC"

    CONTACT-INFO "Informacion de contacto: Avenida de Los Castros"

    DESCRIPTION  "MIB para la gestion de los valores obtenidos por el sendor
de temperatura y humedad dht11 o dht22."

        ::= { netSnmpExamples 333 }   -- Esta li-nea define la ubicacion del
primer nodo de esta MIB. En concreto netSnmpExamples.333, OID:
1.3.6.1.4.1.8072.2.333.


-- Se define el nodo principal, bajo el cual estaran situados los objetos
gestionados en este prototipo.


sensorTemperaturaMIBObjects     OBJECT IDENTIFIER ::= { sensorTemperaturaMIB
1 }


-- Definicion de objetos:

--OID asociado al lugar donde esta situado los valores del sensor .
```

```
sensorPractica OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor para modificar en la practica"
    ::= { sensorTemperaturaMIBObjects 1 }
--TABLA DE LOS VALORES DEL SENSoR.
sensorTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SensorEntry
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Esta tabla contiene los datos del sensor que maneja la temperatura y la
humedad."
    ::= { sensorTemperaturaMIBObjects 2 }
sensorEntry OBJECT-TYPE
    SYNTAX      SensorEntry
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Fila de la tabla que contiene el nombre de lo que mide el sensor
(Temperatura o Humedad) y su valor de potencia consumida en watts."
    INDEX   { sensor }
    ::= {sensorTable 1 }
SensorEntry ::= SEQUENCE {
    sensor  OCTET STRING,
    valor   Integer32,
    }
sensor OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
```

```
"Valor que tiene cada una de las entradas de lo que mide el sensor."
    ::= { sensorEntry 2 }
temperaturaDH11 OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor de la temperatura del sensor DH11"
    ::= { sensorTemperaturaMIBObjects 3  }
humedadDHT11 OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor de la humedad del sensor DH11"
    ::= { sensorTemperaturaMIBObjects 4  }
temperaturaDH22 OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor de la temperatura del sensor DH22"
    ::= { sensorTemperaturaMIBObjects 5  }
humedadDHT22 OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
    "Valor de la humedad del sensor DH22"
    ::= { sensorTemperaturaMIBObjects 6  }
```

```
-- TRAP TEMPERATURA: Trap asociado a la temperatura que queremos para que mande
el aviso de la temperatura.

trapTemperaturaDH11 NOTIFICATION-TYPE

    OBJECTS {temperaturaDH11} -- cambiar al nombre del valor

      STATUS  current

    DESCRIPTION

            "Trap que se envi-a cuando el sensor lee los valores de la
temperatura y supera el valor que pongamos como limite en el codigo"

    ::= { sensorTemperaturaMIBObjects 8}


-- TRAP HUMEDAD: Trap asociado a la humedad que queremos para que mande el aviso
de la humedad

trapHumedadDH11 NOTIFICATION-TYPE

      OBJECTS {humedadDH11} -- modificar

    STATUS  current

    DESCRIPTION

            "Trap que se envi-a cuando el sensor lee los valores de la humedad
y supera el valor que pongamos como limite en el codigo"

    ::= { sensorTemperaturaMIBObjects 9}


END
```

Figura A.19 MIB SENSOR-TEMPERATURA-MIB

A.20 Código Archivo SNMPD.CONF

```
##############################################################################
#######
#
# EXAMPLE.conf:
#   An example configuration file for configuring the Net-SNMP agent
('snmpd')
#   See the 'snmpd.conf(5)' man page for details
#
#  Some entries are deliberately commented out, and will need to be
explicitly activated
#
##############################################################################
#######
#
#  AGENT BEHAVIOUR
#

#  Listen for connections from the local system only
#agentAddress  udp:127.0.0.1:161
#  Listen for connections on all interfaces (both IPv4 *and* IPv6)
agentAddress udp:161,udp6:[::1]:161



##############################################################################
#######
#
#  SNMPv3 AUTHENTICATION
#
#  Note that these particular settings don't actually belong here.
#  They should be copied to the file /var/lib/snmp/snmpd.conf
#     and the passwords changed, before being uncommented in that file
*only*.
#  Then restart the agent

#  createUser authOnlyUser  MD5 "remember to change this password"
#  createUser authPrivUser  SHA "remember to change this one too"  DES
#  createUser internalUser  MD5 "this is only ever used internally, but
still change the password"

#  If you also change the usernames (which might be sensible),
#  then remember to update the other occurances in this example config
file to match.
```

```
##############################################################################
########
#
#  ACCESS CONTROL
#

                                                    #  system + hrSystem
groups only
view   all  included   .1.3.6.1.2.1.1
view   all  included   .1.3.6.1.2.1.25.1


rocommunity public 192.168.1.34
                                                    #  Full access from
the local host
rocommunity public 192.168.110.1

#rocommunity public  localhost
                                                    #  Default access to
basic system info
# rocommunity public  default    -V all
                                                    #  rocommunity6 is for
IPv6
# rocommunity6 public  default   -V all

 rwcommunity private                                #  Full access from an
example network
                                                    #     Adjust this
network address to match your local
                                                    #     settings, change
the community string,
                                                    #     and check the
'agentAddress' setting above
#rocommunity secret  10.0.0.0/16

                                                    #  Full read-only
access for SNMPv3
 rouser   authOnlyUser
                                                    #  Full write access
for encrypted requests
                                                    #     Remember to
activate the 'createUser' lines above
#rwuser   authPrivUser   priv

#  It's no longer typically necessary to use the full
'com2sec/group/access' configuration
#  r[ow]user and r[ow]community, together with suitable views, should
cover most requirements

com2sec readonly default public
com2sec readwrite default private
```

```
###############################################################################
#  SYSTEM INFORMATION
#

#  Note that setting these values here, results in the corresponding
MIB objects being 'read-only'
#  See snmpd.conf(5) for more details
sysLocation    Mi Casa
sysContact     Javier <javign10@hotmail.com>
                                                  # Application + End-
to-End layers
sysServices    72


#
#  Process Monitoring
#
                                  # At least one  'mountd' process
proc  mountd
                                  # No more than 4 'ntalkd' processes - 0
is OK
proc  ntalkd    4
                                  # At least one 'sendmail' process, but
no more than 10
proc  sendmail 10 1

#  Walk the UCD-SNMP-MIB::prTable to see the resulting output
#  Note that this table will be empty if there are no "proc" entries in
the snmpd.conf file


#
#  Disk Monitoring
#
                                  # 10MBs required on root disk, 5% free
on /var, 10% free on all other disks
disk       /     10000
disk       /var  5%
includeAllDisks  10%

#  Walk the UCD-SNMP-MIB::dskTable to see the resulting output
#  Note that this table will be empty if there are no "disk" entries in
the snmpd.conf file


#
#  System Load
#
                                  # Unacceptable 1-, 5-, and 15-minute
load averages
load   12 10 5

#  Walk the UCD-SNMP-MIB::laTable to see the resulting output
#  Note that this table *will* be populated, even without a "load"
entry in the snmpd.conf file
```

```
#  ACTIVE MONITORING
#
                                      #    send SNMPv1  traps
 trap2sink     localhost public
                                      #    send SNMPv2c traps
 trap2sink   default public

 trap2sink 192.168.1.34

 trap2sink 192.168.1.46

 trap2sink 192.168.110.1

 trap2sink 192.168.110.2

 trap2sink 192.168.110.3
                                     #    send SNMPv2c INFORMs
 informsink   localhost public

#  Note that you typically only want *one* of these three lines
#  Uncommenting two (or all three) will result in multiple copies of
each notification.


#
#  Event MIB - automatically generate alerts
#
                                     # Remember to activate the
'createUser' lines above
iquerySecName   internalUser
rouser          internalUser
                                     # generate traps on UCD error
conditions
defaultMonitors          yes
                                     # generate traps on linkUp/Down
linkUpDownNotifications  yes


#  EXTENDING THE AGENT
#  Arbitrary extension commands
#
 extend    test1   /bin/echo  Hello, world!
 extend-sh test2   echo Hello, world! ; echo Hi there ; exit 35
#extend-sh test3   /bin/sh /tmp/shtest
```

```
#  Note that this last entry requires the script '/tmp/shtest' to be
created first,
#    containing the same three shell commands, before the line is
uncommented

#  Walk the NET-SNMP-EXTEND-MIB tables (nsExtendConfigTable,
nsExtendOutput1Table
#     and nsExtendOutput2Table) to see the resulting output

#  Note that the "extend" directive supercedes the previous "exec" and
"sh" directives
#  However, walking the UCD-SNMP-MIB::extTable should still returns the
same output,
#     as well as the fuller results in the above tables.


#
#  "Pass-through" MIB extension command
#
#pass .1.3.6.1.4.1.8072.2.255  /bin/sh       PREFIX/local/passtest
#pass .1.3.6.1.4.1.8072.2.255  /usr/bin/perl PREFIX/local/passtest.pl

pass .1.3.6.1.2.1.25.1.8.2 /bin/sh /usr/local/bin/temp.sh
pass .1.3.6.1.2.1.25.1.8.1 /bin/sh /usr/local/bin/humid.sh
pass .1.3.6.1.2.1.25.1.8.3 /bin/sh /usr/local/bin/temp22.sh
pass .1.3.6.1.2.1.25.1.8.4 /bin/sh /usr/local/bin/hum22.sh

# Note that this requires one of the two 'passtest' scripts to be
installed first,
#    before the appropriate line is uncommented.
# These scripts can be found in the 'local' directory of the source
distribution,
#     and are not installed automatically.

#  Walk the NET-SNMP-PASS-MIB::netSnmpPassExamples subtree to see the
resulting output
#
#  AgentX Sub-agents
#
                                              #  Run as an AgentX master
agent
 master          agentx
                                               #  Listen for network
connections (from localhost)
                                               #    rather than the default
named socket /var/agentx/master
#agentXSocket    tcp:localhost:705
```

Figura A.20 Archivo configuración SNMPD

A.21 Script Restablecer Archivos

```
#!/bin/bash

echo " << Menu para Restablecer Archivos por defecto >>"
sleep 2
echo -e "\e[0;34mEstas seguro que quieres restablecer los valores por
defecto\e[m"
echo
choice=""

while [ "$choice" != "2" ]
do
        echo
           echo -e "\e[94m"
        echo "Selecciona una opcion!"
        echo "1) Restablecer Archivos"
        echo "2) Salir"
        read choice
        case $choice in

            '1')      echo -e "\e[94mRestableciendo los archivos del
proyecto...\e[m";
                           sleep 2;
                           tar -xzvf BackupProyecto.tar.gz -C
/home/javier/Escritorio/;
                           sleep 2;
                           clear;
                           echo -e "\e[94mArchivos restablecidos\e[m";;

            '2')      clear;
                           sleep 1;
                           echo -e "\e[31mSaliendo. \e[m";
                           sleep 1;
                           clear;
                           echo -e "\e[31mSaliendo.. \e[m";
                           sleep 1;
                           clear;
                           echo -e "\e[31mSaliendo... \e[m";
                           sleep 1;
                           clear;
                           sleep 1;;

             *)       clear;
                           echo -e "\e[31mIntroduce una opcion valida
!!! Intentalo de nuevo\e[m";;

        esac
done
```

Figura A.21 Script restablecer archivos por defecto

A.22 Script Valores Exportados Cactis

```
#!/bin/bash


############### Valores Exportar ############################

RESULTS="`python
/home/javier/Escritorio/proyecto/Adafruit_Python_DHT/examples/Adafrui
tDHT.py 11 4 | grep Temp `"
RESULTS2="`python
/home/javier/Escritorio/proyecto/Adafruit_Python_DHT/examples/Adafrui
tDHT.py 22 17 | grep Temp `"
TEMP="`echo ${RESULTS} | awk '{print $1}'`"
TEMP="${TEMP:5:-1}"
HUM="`echo ${RESULTS} | awk '{print $2}'`"
HUM="${HUM:9:-1}"
TEMP2="`echo ${RESULTS2} | awk '{print $1}'`"
TEMP2="${TEMP2:5:-1}"
HUM2="`echo ${RESULTS2} | awk '{print $2}'`"
HUM2="${HUM2:9:-1}"
HUM3="${HUM2:9:-1}"
echo "${TEMP}" > /home/pi/temp.txt;
echo "${HUM}" > /home/pi/humid.txt;
echo "${TEMP2}" > /home/pi/temp22.txt;
echo "${HUM2}" > /home/pi/humid22.txt;
read a < /home/pi/temp22.txt
read b < /home/pi/humid22.txt
echo "($a+0.5)/1" | bc > /home/pi/temp222.txt;
echo "($b+0.5)/1" | bc > /home/pi/humid222.txt;
echo "TemperaturaDHT22 ${a}" >>
/home/javier/Escritorio/proyecto/Demonios/TablaSensor/sensorLectura.t
xt;
read a < /home/pi/temp222.txt
read b < /home/pi/humid222.txt
echo "${a}" > /home/pi/temp2.txt
echo "${b}" > /home/pi/humid2.txt
```

Figura A.22 Script LED sensorPractica

A.23 Practica

Grado en Ingeniería de Tecnologías de Telecomunicación
3º Curso (2er Cuatrimestre)
Gestión y Operación de Redes
Curso 2017/2018

Práctica 4

Estudio del protocolo SNMP

OBJETIVOS DE LA PRÁCTICA

Los principales objetivos de esta práctica son:

- Familiarizarse con el manejo de un analizador de red
- Conocer la MIB 2 desde un punto de vista práctico
- Generar las diferentes primitivas del protocolo de gestión SNMP
- Estudiar los formatos de los mensajes SNMP (RFC 1157)

INTRODUCCIÓN

Formato de los Mensajes SNMP

SNMP permite el intercambio de información a través de la red entre la estación de gestión y el agente en forma de mensajes SNMP. Cada mensaje incluye un número de versión que indica la versión de SNMP, un nombre de comunidad utilizado en el intercambio, y uno de los cinco tipos de PDU's definidos: **GetRequest, GetNextRequest, SetRequest, GetResponse y Trap**.

Los mensajes tienen el siguiente formato:

Versión	Comunidad	SNMP PDU

Donde:

Versión indica la versión del protocolo. RFC 1157 es versión 1.

Comunidad es el nombre de la comunidad y sirve para autentificar el mensaje SNMP.

PDU SNMP depende del tipo de operación a realizar. Este puede ser:

- **Si se trata de GetRequest, GetNextRequest o SetRequest tendremos:**

PDU type	Request Id	0	0	Variable Bindings

PDU type: indica el tipo de PDU,

Request Id: se utiliza para diferenciar las distintas peticiones, añadiendo a cada una de ellas un único identificador.

Variable Bindings: es una lista de nombres de variables y sus correspondientes valores. En algunos casos (GetRequest), el valor de las mismas es NULL. En el caso de las Traps, proporcionan información adicional relativa a la Trap, dependiendo el significado de este campo de cada implementación en particular.

- **Si se trata de GetResponse:**

PDU type	Request Id	Error-status	Error-index	Variable Bindings

Error-status: se utiliza para indicar que ha ocurrido una excepción durante el procesamiento de una petición. Sus valores posibles son: **noError** (0), **tooBig** (1), **noSuchName** (2), **badValue** (3), **readOnly** (4), **genErr** (5).

Error-index: cuando el campo **Error-status** es distinto de 0, puede proporcionar información adicional indicando la variable que causó la excepción.

- **Si se trata de una Trap:**

PDU type	Enterprise	agent-addr	generic-trap	specific-trap	time stamp	Variable Bindings

Agent-addr: Dirección IP del agente que generó la Trap.

Generic-trap: Tipo de Trap genérico predefinido. Puede ser:

- **coldStart**(0): el agente se ha reinicializado, de forma que se puede alterar la configuración de los agentes o la implementación del protocolo. Típicamente reinicio por caída del sistema.
- **warmStart**(1):la entidad emisora SNMP se ha reinicializado sin haberse alterado la configuración de los Agentes ni la implementación del protocolo. Usualmente es una rutina de tipo restart.
- **linkDown**(2): señaliza un fallo en alguno de los enlaces de comunicación del Agente. El primer elemento en el campo Variable-Bindings indicará el interfaz en cuestión.
- **linkUp**(3): señaliza el restablecimiento de uno de los enlaces de comunicación del Agente. El primer elemento en el campo Variable-Bindings indicará el interfaz en cuestión.
- **authentificationFailure**(4): indica que la entidad emisora de la Trap ha recibido un mensaje en el que ha fallado la autentificación.
- **egpNeighborLoss**(5): indica que un EGP (External Gateway Protocol) vecino, para el cual la entidad emisora tenía asociado otro EGP, ha sido desmarcado y la relación entre ambos EGPs ha finalizado.
- **enterpriseSpecific**(6): significa que la entidad emisora reconoce que algún evento específico del fabricante ha ocurrido. El campo specific-trap indica el tipo de Trap.

Specific-trap: Código de Trap específico e indica de una forma más específica la naturaleza de la Trap.

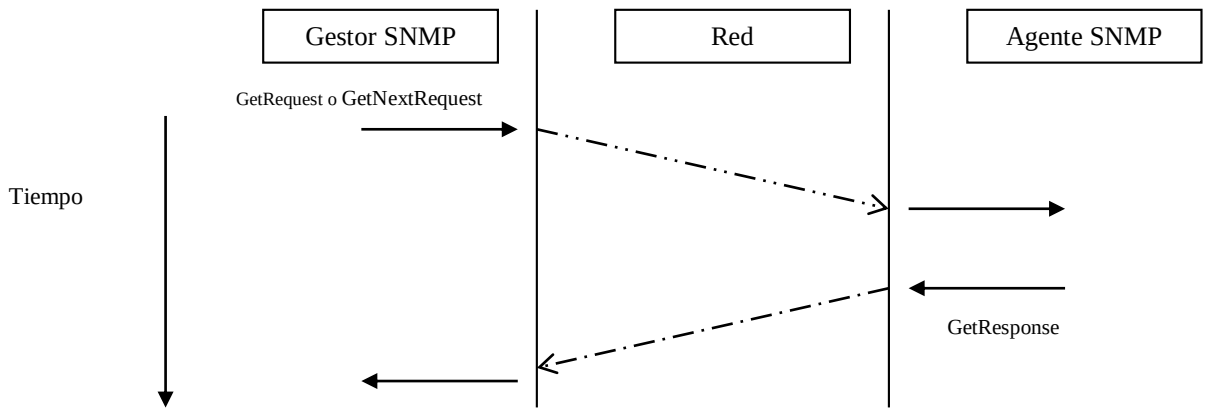
Timestamp: Tiempo transcurrido entre la última reinicialización de la entidad de red y la generación de la trap.

Transacciones SNMP

Como hemos comentado anteriormente las operaciones básicas en SNMP son:

- De obtención de datos: GetRequest, GetNextRequest;
- De modificación: SetRequest;
- De aviso: Trap.

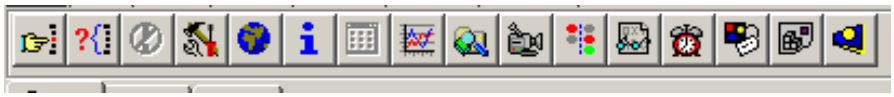
Ejemplo de la transacción de obtención de datos:

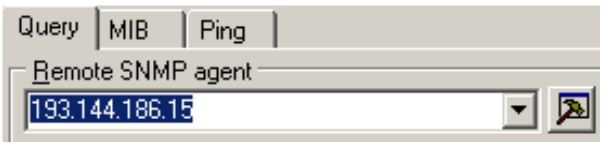


DESARROLLO DE LA PRÁCTICA

1. Familiarizarse con la aplicación de gestión Mib Browser

A) Breve explicación del funcionamiento de la aplicación: su menú principal, las solapas Query, MIB y otros aspectos generales del funcionamiento básico de la herramienta





B) Descubrimiento de agentes SNMP

Mediante la funcionalidad "Remote SNMP agent discovery", descubrir los agentes SNMP que existen en el laboratorio. Tener en cuenta que en el laboratorio hay tres subredes IP: 192.168.110.X, 192.168.200.X y 128.100.100.X. Rellena la tabla con los agentes descubiertos

Nombre	Dir IP

2. Generación y captura de transacciones SNMP

Con la ayuda de la aplicación gestora MIB Browser, generar los distintos tipos de transacciones SNMP posibles sobre un agente accesible en el laboratorio. (**GetRequest, GetNextRequest, SetRequest, Trap**).

A) Realizar un **GetRq** sobre el objeto "sysName" de la Mib2 (menú botón derecho sobre el objeto y seleccionar función get), capturarlo con el analizador Wireshark. Observando la captura anterior completar la siguiente tabla:

	Dir IP	Puerto UDP
Cliente SNMP		
Servidor SNMP		

Indicar los valores que tiene cada campo del mensaje GetRq capturado

Versión	Comunidad	PDU type	Rq ID	Err. Status	Err. Index	OID	Valor

B) En la captura anterior también aparece el **GetResponse** de respuesta a la petición realizada, indicar el valor de los campos de este mensaje

Versión	Comunidad	PDU type	Rq ID	Err. Status	Err. Index	OID	Valor

C) Realizar un **GetRq** solicitando el valor de los objetos "sysName" y "sysLocation" de la MIB 2, para ello usar la utilidad "Multiple Variable Bindings" (botón derecho sobre un objeto de la mib). Se abrirá una ventana que permite arrastrar los objetos a interrogar. Fijarse que debemos incluir la instancia de cada objeto (.0), si no, nos dará error. Indicar sobre la tabla los valores obtenidos en la PDU de respuesta.

PDU type	Rq ID	Err. Sta	Err. In	OID1	Valor1	OID2	Valor2

D) Realizar un **GetNextRq** (menú botón derecho sobre el objeto y seleccionar función getnext) al OID "1.3.6.1.2.1.1" y capturarlo junto con la respuesta del agente. Indicar los campos de la PDUs de petición y respuesta en la tabla siguiente:

	PDU type	Rq ID	Err. Status	Err. Index	OID	Valor
PETICIÓN						
RESPUESTA						

E) Realizaremos un **GetRq** sobre el objeto sensorPractica (OID = 1.3.6.1.4.1.8072.2.333.1.1) y capturaremos los valores obtenidos. ¿Debemos hacer el GetRq sobre el OID indicado? Añadir lo necesario para obtener los valores correspondientes.

	PDU type	Rq ID	Err. Status	Err. Index	OID	Valor
PETICIÓN						

RESPUESTA						

Seguimos viendo el problema aun añadiendo lo necesario para obtener los valores del objeto indicado. Debemos inicializar el demonio del sensorPractica en nuestro agente. Una vez inicializado completaremos la tabla otra vez.

	PDU type	Rq ID	Err. Status	Err. Index	OID	Valor
PETICIÓN						
RESPUESTA						

F) Utiliza la primitiva **SetRq** (botón derecho sobre el nodo a modificar y ejecutar función set) para cambiar el valor del objeto "sensorPractica" del equipo de la raspberry PI. Capturar los mensajes de petición y respuesta y completar la tabla siguiente.

Fíjate que necesitas el community de escritura para poder realizar la operación correctamente.

	Comm.	PDU type	Rq ID	Err. Status	Err. Index	OID	Valor
PETICIÓN							
RESPUESTA							

¿Qué ha ocurrido? Modificar el valor de lectura de la community

* Valor lectura "private"

G) Ahora inicializaremos los sensores de temperatura de la Raspberry PI junto con el demonio de la tabla para que así se puedan modificar los valores mediante los sensores y poder visualizar dicha Tabla. Lee la tabla "sensorTable" del agente de la Raspberry Pi. Dicha tabla se encuentra dentro de la MIB del proyecto. "OID SensorTable =1.3.6.1.4.1.8072.2.333.1.2"

A la vista de la captura completar la tabla:

Nº filas de la tabla (F)	Nº de columnas (C)	Nº de peticiones SNMP(P)	Nº respuestas SNMP(R)

Escribe la expresión que relaciona los cuatro valores

H) Utiliza la primitiva **GetBulkRq** (N=0,M=2) para leer la misma tabla, para ello utiliza el comando "Multiple Variable Bindings" con los objetos columna de la tabla. ¿Cuántas peticiones anidadas tendría que hacer para leer la tabla completa?

I) *Inicializamos los demás demonios del proyecto junto con las traps para este último apartado. Hay una opción de inicializar todo con la cual se inicializarán todos los procesos necesarios para el desarrollo de la práctica. Para obtener el envío de una TRAP en nuestro proyecto debemos visualizar los objetos de la MIB TemperaturaDHT11 y HumedadDHT11. Cuando estos valores superen unos valores predeterminados estos enviaran unas traps a las direcciones que este configurado. Una vez se estén enviando las traps a nuestro PC podemos capturar los valores de dichas traps mediante wireshark. A continuación, procederemos a rellenar la tabla con los valores visualizados.

PDU type	Enterprise	agent-addr	generic-trap	specific-trap	time stamp	OID1	Valor	OID2	Valor

J) Inicializar el proceso de tabla_historico_demon y realizar X lecturas. Utilizando la primitiva **GetBulkRq** (N=0, M=2) de nuevo. ¿Cuantas peticiones se deberá realizar para leer esta nueva tabla? OID=.1.3.6.1.4.1.8072.2.333.1.11