ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO PARA LA MONITORIZACIÓN DE PARÁMETROS AMBIENTALES UTILIZANDO UN RASPBERRY PI Y UN AGENTE MULTIPROTOCOLO

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN "ELECTRÓNICA Y REDES DE INFORMACIÓN"

ANDRÉS SEBASTIÁN VILLACÍS LARA
andres.villacis@epn.edu.ec

DIRECTOR: ING. XAVIER ALEXANDER CALDERÓN HINOJOSA MSC
xavier.calderon@epn.edu.ec

Quito, julio 2019

AVAL

Certifico que el presente trabajo fue desarrollado por Andrés Sebastián Villacís Lara, bajo mi supervisión.

_____

M.Sc. XAVIER CALDERÓN
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Andrés Sebastián Villacís Lara, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

ANDRES SEBASTIÁN VILLACÍS LARA

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE SEGMENTOS DE CÓDIGO

RESUMEN

En el presente proyecto se desarrolla un prototipo de monitorización de parámetros ambientales utilizando un Raspberry Pi y un agente multiprotocolo. Los parámetros ambientales son captados por sensores conectados a un Arduino Uno, el Arduino envía los datos capturados a través de comunicación serial hacia el Raspberry Pi, en el Raspberry Pi los datos son guardados en archivos de texto.

Los archivos de texto son utilizados como fuentes de información del agente multiprotocolo situado también en el Raspberry Pi; para la generación del agente es necesario la utilización de una MIB que define los objetos que pueden ser consultados a través de SNMP y RMI al agente.

Se creó una aplicación de escritorio utilizando el lenguaje de programación Java que se conecta a través de RMI al agente, obtiene los objetos expuestos a través de la arquitectura JMX y los presenta. La aplicación permite realizar consultas de valores históricos para una determinada fecha, para esto se creó una base de datos contenida en otro Raspberry Pi que constantemente se alimenta de los valores cargados en los archivos de texto que son fuentes de información del agente.

PALABRAS CLAVE: Raspberry Pi, JMX, MIB, SNMP

ABSTRACT

In the present project, a monitoring prototype was developed. The environmental parameters are gotten by sensors which are connected to an Arduino Uno, this Arduino sends the data through serial communication to the Raspberry Pi, then, in the Raspberry Pi the data is saved inside of text files.

These text files are used as information source of the multiprotocol agent also inside of the Raspberry Pi; for the generation of the agent it's necessary to use a MIB which defines objects that can be consulted through SNMP and RMI to the agent.

A desktop application was created using Java as language programming, this application is connected through RMI to the agent and obtains the objects through the JMX architecture and it is used to draw every new value received. The application allows queries of historical values for a given date, for this purpose a database has been created and it is contained in another Raspberry Pi which is constantly fed by the values located in the text files which are sources of agent information.

KEYWORDS: Raspberry Pi, JMX, MIB, SNMP

1. INTRODUCCIÓN

En algunas industrias, a causa de las acciones propias del proceso de producción, se generan gases y partículas contaminantes, por ejemplo, en las empresas dedicadas a la manufactura de ventanas de PVC (Cloruro de polivinilo), se emite monóxido de carbono por causa de la utilización de motores de combustión, el monóxido de carbono es común en el ambiente, sin embargo, cuando las concentraciones son muy elevadas se convierten en un peligro para la salud de las personas; aquellas con enfermedades cardiovasculares, respiratorias y mujeres embarazadas son aún más sensibles a este gas [1]. Otro elemento que aparece es el material particulado fino, el cual incluye todos los elementos presentes en el aire cuyo diámetro no supera los 2.5 micrones, este elemento es peligroso para aquellas personas que tienen enfermedades en pulmones como asma, o del corazón como congestiones cardíacas [2].

Algunas empresas no poseen equipos ni soluciones que les permitan monitorear activamente estos elementos principalmente por desconocimiento y por el precio requerido para adquirir estos sistemas, por lo cual se plantea el desarrollo de un prototipo de monitoreo de monóxido de carbono y de material particulado fino utilizando un Raspberry Pi y un agente multiprotocolo que permita entregar los valores recogidos de lo sensores a través del protocolo SNMP y la tecnología RMI, para la visualización se implementa una aplicación de escritorio que mediante RMI se conecta al agente y obtiene los datos a través de la arquitectura JMX para su posterior graficación, esta aplicación permite también recoger datos históricos albergados en una base de datos presente en otro Raspberry Pi.

1.1. OBJETIVOS

El objetivo general de este estudio técnico es desarrollar un prototipo para la monitorización de parámetros ambientales utilizando un Raspberry Pi y un agente multiprotocolo.

Los objetivos específicos de este estudio técnico son:

- Analizar los fundamentos teóricos necesarios para el desarrollo e implementación del proyecto.
- Diseñar los módulos de la solución propuesta.
- Implementar los módulos previamente diseñados.
- Analizar los resultados obtenidos.

1.2. ALCANCE

En el prototipo se utilizará información proveniente de la empresa Yongping situada en las periferias de Quito, se hará uso de la plataforma de desarrollo Raspberry Pi 3 para gestionar los valores de material particulado fino ($PM_{2.5}$) y monóxido de carbono (CO), utilizando un Arduino Uno como dispositivo intermedio, y sensores de bajo costo. Para el muestreo, almacenamiento de los datos y calibración, se considerará lo definido en el documento "Guide to Meteorological Instruments and Methods of Observation".

El Raspberry Pi 3 albergará al agente multiprotocolo, el cual gestionará una MIB propia y utilizará los datos obtenidos de los sensores como fuente de información, estos datos serán accesibles a través de SNMP y RMI. Se desarrollará una aplicación de escritorio que pueda acceder a los datos del agente mediante RMI y utilizar la arquitectura JMX para obtener los valores, esta aplicación contará con 2 ventanas. En la ventana principal se realizarán gráficos con los valores de los sensores obtenidos utilizando la arquitectura JMX, existirá un gráfico por cada sensor utilizado. La ventana secundaria permitirá mostrar tablas de reportes de datos históricos según la selección de la fecha deseada, para acceder a estos datos, la aplicación se conectará con una base de datos ubicada en otra PC dentro de la red interna, la cual albergará datos obtenidos desde los archivos de los valores de los sensores utilizados como fuente de información por el agente multiprotocolo. La aplicación de escritorio utilizará únicamente RMI para la conexión y no SNMP.

A continuación, se presenta de manera desglosada el proyecto, el cual se conformará de 3 secciones o módulos principales:

Módulo de Adquisición

En este módulo se da la obtención de los valores de los parámetros ambientales, esto se realizará mediante el uso de sensores de $PM_{2.5}$ y CO conectados a un Arduino Uno, se codificará y cargará un script en el Arduino que, permita entregar los valores capturados a través de comunicación serial hacia el Raspberry Pi 3, se codificará un script para obtener los datos por un puerto serial en el Raspberry Pi 3 y guardar los valores en archivos de texto, considerando como referencia el documento mencionado anteriormente.

Módulo Agente

En este módulo se realiza la compilación y uso de un agente multiprotocolo, este agente se encontrará en el Raspberry Pi 3, se necesitará codificar una MIB propia y la programación de código extra al código esqueleto del agente para qué de esta manera, este permita utilizar los archivos de texto mencionados anteriormente como fuente de información. Se podrá acceder a la información del agente a través de SNMP y RMI.

Módulo de Visualización

Para el último módulo será implementada una aplicación de escritorio, utilizando el lenguaje de programación Java, que pueda acceder a la información del agente mediante la arquitectura JMX utilizando RMI para la comunicación. Esta información recogida será utilizada para su presentación en gráficos. Se instalará y configurará una base de datos en otro dispositivo, la cual será alimentada con información proveniente de los archivos de texto antes mencionados; se utilizará un script para la inserción de datos en dicha base de datos. Desde la aplicación de escritorio también se podrá acceder a la información de esta base de datos, para la presentación de reportes de datos históricos.

La Figura 1.1 muestra el escenario que se pretende crear en el presente prototipo. Este trabajo de titulación si tiene producto final demostrable.

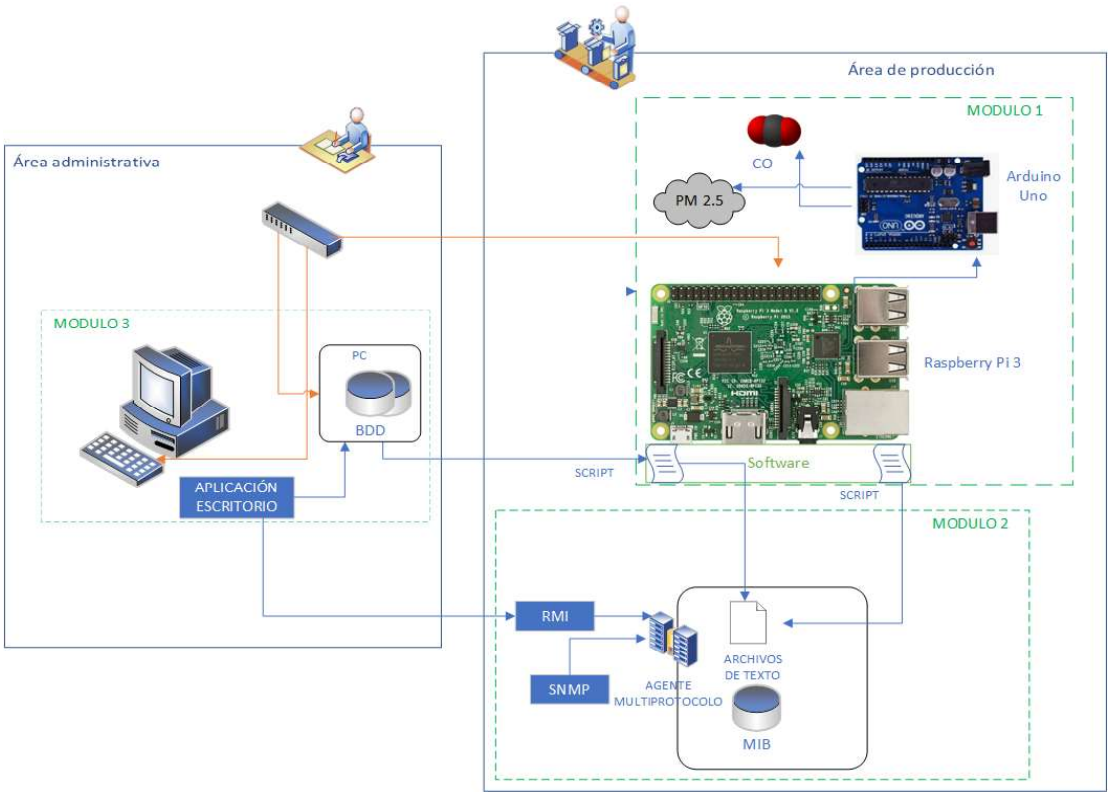


Figura 1.1. Esquema de conexión del prototipo

1.3. MARCO TEÓRICO

En este capítulo se realiza una breve descripción de las características principales de las tecnologías utilizadas para el desarrollo del presente proyecto. Empezando por las características e información relevante de la plataforma de hardware, así como un breve resumen sobre la gestión de redes clásica, para entonces hablar sobre la tecnología JMX[1], seguido de los sistemas de gestión de bases de datos. De todo esto se hará énfasis en las tecnologías a utilizar.

1.3.1. RASPBERRY PI

El Raspberry Pi o también conocido como RPi, es una minicomputadora utilizada principalmente como plataforma de desarrollo debido a su bajo costo y por sus altas prestaciones. Una de sus grandes virtudes es la gran comunidad existente que trabaja en

[1] **JMX:** Java Management Extensions, es una tecnología de Java enfocada en la gestión de software.

el desarrollo de proyectos de toda índole, permitiendo de esta manera que más personas se interesen en la plataforma y que las áreas de desarrollo y aplicación puedan expandirse. El objetivo de su existencia fue dar a las personas el poder de resolver sus problemas utilizando su creatividad de la mano de la tecnología. Según cifras oficiales hasta finales del 2017, se habían vendido 17 millones de RPi alrededor del mundo, esto en tan solo 5 años de existencia [3].

El modelo principal durante el paso del tiempo ha tenido mejoras, llegando actualmente hasta el modelo 3B+ [4]. A continuación, se presenta una tabla comparativa con las principales placas RPi, se dejó a un lado los modelos iniciales de las versiones 1 y 2 dado que la versión 3 es la más actual y contiene mejores características.

Tabla 1.1. Comparación de modelo Raspberry Pi [3]

	Raspberry Pi Zero	**Raspberry Pi Zero w**	**Raspberry Pi 3B**	**Raspberry Pi 3B+**
				
CPU	Single Core 1GHz	Single Core 1GHz	Quad Core 1.2GHz Broadcom 64bit.	Quad Core 1.4GHz Broadcom 64bit
RAM	512 MB	512MB	1GB	1GB
Video / in	CSI v1.3	CSI v1.3	CSI	CSI
Video/OUT	Mini HDMI	Mini HDMI	HDMI	HDMI
USB 2.0	USB OTG	USB OTG	4	4
Almacenamiento	MicroSD	MicroSD	MicroSD	MicroSD
Tarjeta de Red			10/100 base T	10/100Base T 1000Base T (USB)
Wi-Fi			802.11 b/g/n	802.11 b/g/n/ac
Bluetooth			4.1	4.2
BLE		Si		Si
POE				Si (hat)
GPIO	40	40	40	40

El modelo por utilizar en el presente proyecto es el RPi 3B debido a los múltiples núcleos de su procesador y su mayor capacidad de RAM frente a los RPi Zero.

1.3.1.1. Sistemas Operativos

La mayoría de los sistemas operativos disponibles para el RPi poseen kernel Linux. Según estimaciones existen más de 45 distribuciones GNU/Linux compatibles con el RPi [5]. Las distribuciones más importantes son las siguientes:

1.3.1.1.1. Ubuntu mate

Es una distribución no oficial basada en Ubuntu armhf[2], es compatible con las versiones RPi 2 y RPi 3, se indica en la página oficial que es necesario instalarlo en una microSD de clase 6 o 10 de 6GB mínimo pues el rendimiento de la microSD para entrada y salida es deficiente [6].

1.3.1.1.2. Raspbian

A pesar de que Raspbian es una versión de Debian extraoficial, este fue nombrado por parte de RPi como su sistema operativo oficial de todos los modelos y por tal motivo su soporte permite tener compatibilidad en la mayoría de las dependencias y herramientas, característica importante al momento de utilizar un sistema operativo para desarrollar un proyecto [7].
Un año después del lanzamiento oficial de Debian Stretch, el 18 de abril del 2018 se lanza la versión Raspbian Stretch la cual será usada en el presente proyecto.

1.3.1.2. Interfaces de comunicación.

El RPi posee pines y puertos para la interacción con dispositivos externos que se suelen ver en otros dispositivos como el USB, el HDMI, entre otros; y otros no tan comunes como son los pines GPIO[3]. Este conjunto de pines le da diferentes posibilidades de comunicación al RPi, puesto que permite la utilización de varios protocolos de comunicación para la interacción con más dispositivos [8]. La Figura 1.2 presenta los pines GPIO del RPi 3.

[2] **armhf**: son las siglas de ARM hard float, hace referencia a sistemas operativos para dispositivos cuyos procesadores son ARM.
[3] **GPIO** es acrónimo de *General Purpose Input Output,* cuya traducción al español indica “entradas y salidas de propósito general”.

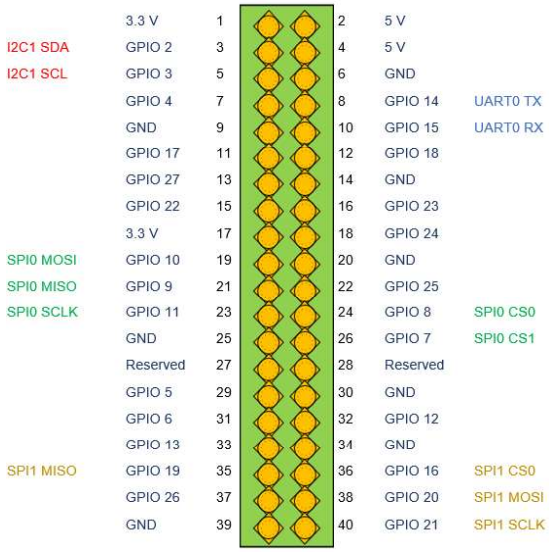


Figura 1.2. Distribución de pines del RPi 3

El grupo de pines GPIO alcanza los 26, numerados desde el GPIO 2 hasta el GPIO 27, los GPIO 0 y 1 también están presentes, sin embargo, fueron reservados para futuras actividades, por tal motivo se los catalogaron como “reservados”.

Además de las funciones básicas antes mencionadas, algunos pines GPIO permiten la utilización de protocolos de comunicación, como por ejemplo el Protocolo I2C que trata sobre el uso de un bus en serie principalmente para la comunicación entre microcontroladores y también para la comunicación con sensores, el esquema es maestro esclavo en el que cada esclavo necesita una dirección única de 7 bits y su tipo de comunicación es Half Duplex [9]. Otro tipo de comunicación es SPI que también es utilizado entre microcontroladores, el esquema es maestro esclavo y es un bus de datos que utiliza 4 líneas para la comunicación, el tipo de comunicación es Full Duplex [10]. Finalmente, el protocolo UART que permite la comunicación entre 2 dispositivos, permite los tipos de comunicación Simplex, Half Duplex y Full Duplex, y necesita 2 líneas para la comunicación [11].

1.3.1.3. Lenguajes de Programación.

Los lenguajes de programación soportados por el sistema operativo Raspbian son muchos, básicamente los mismos lenguajes compatibles con Debian, sin embargo, no todos tendrán soporte para manipular los puertos GPIO.

Los lenguajes de programación más comunes y ampliamente utilizados en desarrollo de proyectos sobre Raspbian son aquellos que poseen soporte continuo.

Varios de estos lenguajes vienen ya integrados con Raspbian, ellos son: Scratch, Java y Python. Encontrar estos lenguajes ya instalados nos da información importante, nos indica que son lenguajes que utiliza la comunidad de RPi activamente en sus proyectos, y que posee mucho soporte. A continuación, se hablará brevemente sobre los lenguajes de programación más utilizados en un ambiente RPi, así como otras consideraciones al utilizar dispositivos embebidos.

1.3.1.3.1. Scratch

Scratch es un lenguaje de programación visual basada en bloques, su propósito principal fue el dar la oportunidad a niños que aprendan secuencias de programación sin que necesiten aprender reglas de sintaxis de lenguajes de programación tradicionales sino solo arrastrando bloques de acciones. Una de las falencias de Scratch es que no permite ejecutar sus aplicaciones y posee una fuerte dependencia con su entorno de desarrollo web, existen en la web información sobre formas de exportar sus aplicaciones sin embargo no es un método oficial por lo tanto no hay garantía de total funcionamiento [12].

1.3.1.3.2. Java

Java es un lenguaje de programación de código abierto y orientado a objetos, es el lenguaje más utilizado durante el primer semestre del 2018 según Tiobe[4]. Para la ejecución de las aplicaciones es necesario la presencia de una máquina virtual java o conocida por su acrónimo JVM. Java es multiplataforma, permite una vez creada la aplicación, esta pueda ser utilizada en cualquier sistema operativo [13].

1.3.1.3.3. Python

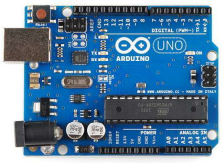
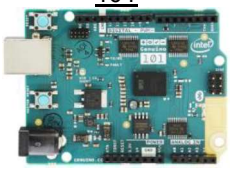
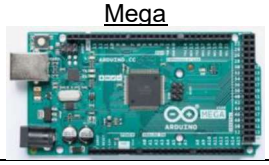
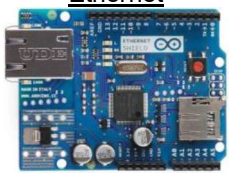
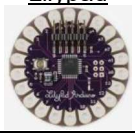
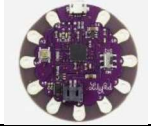
Python también es un lenguaje de programación de código abierto, multiparadigma e interpretado. Se presentan dos versiones ya instaladas en el RPi, la versión 2.7 y la 3.5. Es el lenguaje preferido para desarrollo de proyectos sobre el RPi. Al ser de código abierto existen varias librerías para diferentes propósitos, y sobre todo mucha documentación. Estas últimas características se consideraron esenciales para elegir este lenguaje para la elaboración del script para guardado de datos [14].

[4] **Tiobe:** es una empresa que mide la calidad de los programas desarrollados por cualquier entidad que lo requiera, posee un índice de popularidad de lenguajes de programación

1.3.2. ARDUINO

Arduino es una plataforma de desarrollo de hardware que utiliza microcontroladores Atmel y posee varios interfaces de comunicaciones para lectura y escritura, Arduino posee varios modelos en el mercado, se mencionan algunos modelos clasificados por su categoría:

Tabla 1.2. Modelos relevantes Arduino *[15]*

NIVEL DE ENTRADA		
Uno	Leonardo	101
		
CARACTERISTICAS MEJORADAS		
Mega	Zero	Due
		
INTERNET OF THINGS		
Yun	Ethernet	MKR1000
		
WEARABLE		
Gemma	Lilypad	Lilypad ArduinoUSB
		

A diferencia de Raspberry Pi, en la mayoría de placas Arduino no se utilizan sistemas operativos, al ser un microcontrolador, se debe cargar un programa con las instrucciones a realizar, esto se lo realiza mediante su propio entorno de programación (IDE) llamado Arduino, es multiplataforma, y su lenguaje de programación es C, sin embargo, también puede utilizarse el lenguaje de programación AVR C que es en el cual se basa la plataforma [15].

La clara ventaja sobre la placa Raspberry Pi, es la posibilidad de lectura de señales analógicas directas a través de sus pines analógicos.

1.3.2.1. Arduino Uno

El Arduino Uno es el modelo más vendido y documentado de la marca, es considerada para nivel de ingreso en el desarrollo de prototipos de electrónica. Utiliza el microcontrolador embebido ATmega328p, posee 14 pines digitales de entrada/salida incluyendo 6 salidas PWM, 6 entradas analógicas y conector USB. Una característica atractiva es la posibilidad de reemplazar el microcontrolador si este fallara, pues este, está colocado en un adaptador removible [16].

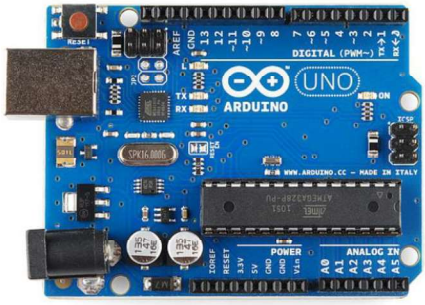


Figura 1.3. Arduino Uno *[16]*

El Arduino UNO maneja varios buses para la comunicación, varios de ellos pueden utilizarse para la comunicación con el Raspberry Pi, ellos son: I2C, SPI, UART y Serial USB.

1.3.3. GESTIÓN DE REDES

La gestión de redes trata sobre la capacidad de una red de poder entregar información importante al administrador, esta red debe permitir el manejo de los recursos o entidades exponiéndolos a ser gestionados. Una red debe poseer ciertos elementos para poder declararse como gestionable [17]. Estos elementos son:

- NMS (Network Management System): Es una entidad de gestión que permite consultar a las entidades situadas en los elementos de red sobre información o recursos que dicha entidad maneja.
- NME (Network Management Entity): Es una entidad para situarse en un elemento de red, que permite que dicho elemento sea gestionable. Trabaja en conjunto con las MIB, y su objetivo es recoger información y entregarlo a la entidad que lo requiera.

- MIB (Management Information Base): Es una representación de base de datos que entrega información sobre determinados recursos presentes en el elemento de red en el que se sitúa, la MIB trabaja en conjunto con la NME para recoger y entregar la información a la entidad que lo requiera.

Estas entidades necesitan de un protocolo de gestión para poder transmitir información desde donde se recoge la información (NME) y hacia donde se la muestra (NMS).

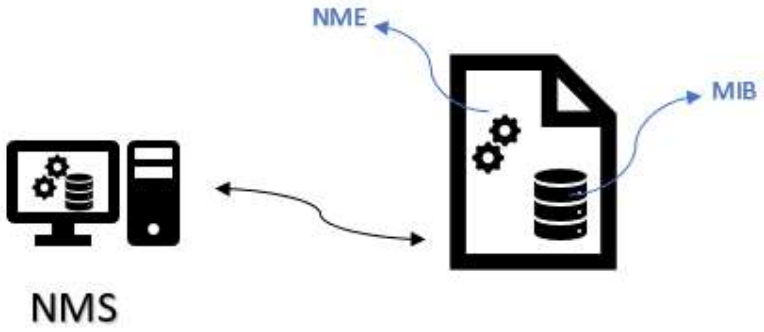


Figura 1.4. Esquema elementos de red gestionable

1.3.3.1. SNMP

SNMP es el acrónimo de Simple Network Management Protocol, es un protocolo de gestión de redes el cual trabaja con la arquitectura TCP/IP para la comunicación [17]. Se definió que el protocolo sobre el cual funcione sea UDP sin embargo dependiendo de la implementación puede definirse sobre TCP. Está definido en las RFC 1157, su propósito fue el proveer un protocolo simple para la gestión de los elementos de una red. Funciona con el protocolo UDP en los puertos 161 y 162. Posee 3 versiones las cuales se denominaron: v1, v2c y v3 [18]. La Figura 1.5 muestra los posibles escenarios de comunicación.

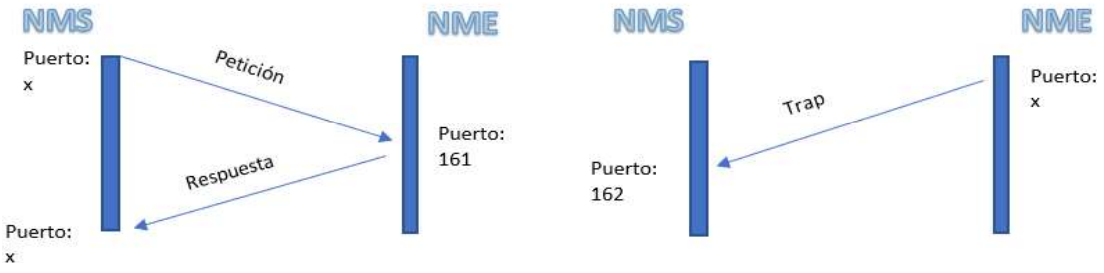


Figura 1.5. Escenarios de comunicaciones SNMP

Las versiones 1 y 2c poseen las mismas características en cuanto a seguridad las cuales son:

- Autenticación: La NMS solo recibe información de las entidades seleccionadas.
- Políticas de acceso: Las entidades gestionadas solo deben permitir el acceso a entidades de gestión y tanta información como se estableció previamente.
- Servicio Proxy: puede utilizarse una entidad gestionada como estación intermedia o cascada hacia otras entidades gestionadas.
- Perfiles de comunidad: permite asociar los permisos de acceso junto a los nombres de los objetos.

SNMP maneja diferentes tipos de PDU (Protocol Data Unit), en la versión 2 se implementaron más tipos de PDUs.

Cada PDU se debe construir siguiendo una estructura predefinida, esta se llama ASN.1. (Abstract Syntax Notation) esta es una norma que define la estructura de los datos de tal manera que sea legible independientemente de la máquina [19]. Una vez estructurado todo el mensaje SNMP, se lo debe codificar utilizando BER.

1.3.3.1.1. SNMP v1

El formato de la trama de SNMP es común para todos los diferentes tipos de PDU, excepto para la trama de las traps[5]. Se definen 5 PDUs [18]:

- GetRequest: utilizada para solicitar información de un recurso específico.
- GetNextRequest: utilizada para solicitar información del recurso siguiente al especificado.
- GetResponse: utilizada para responder a una petición de información.
- SetRequest: utilizada para establecer un nuevo valor del recurso especificado.
- Trap v1: utilizada para notificar al gestor sobre una novedad de un recurso gestionado.

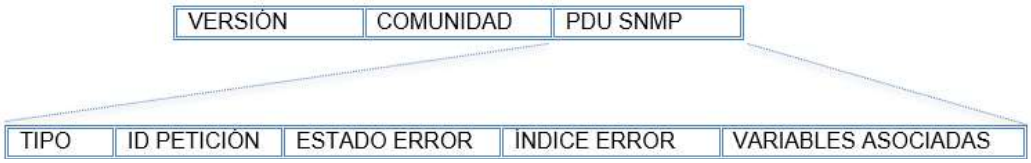


Figura 1.6. PDU SNMP

[5] **Trap**: Una Trap es una trama de notificación enviada desde una entidad gestionada hacia una entidad de gestión previamente definida, esta puede ser configurada para que sea enviada cuando se cumpla una condición determinada.

- Versión: indica la versión de la PDU.
- Comunidad: indica el nombre de la comunidad.
- Tipo: Indica el tipo de PDU utilizado.
- ID Petición: número aleatorio usado para emparejar la petición con la respuesta.
- Estado Error: indica si ha ocurrido un evento en la petición. Sus valores son: noError (0), tooBig (1), noSuchName (2), entre otras.
- Índice Error: cuando el campo Estado Error es diferente de 0, este campo puede contener información adicional sobre el evento.
- Variables Asociadas: Tupla entre OID consultado y valor.

El término Variables Asociadas suele ser nombrado “varbind” y se refiere a una instancia de un objeto administrado. Es el emparejamiento del nombre de una variable con el valor de la variable. En una petición el valor es NULO mientras que en una respuesta el valor toma el contenido de dicha respuesta.

La estructura de una PDU del tipo Trap cambia y se presenta de la siguiente manera:

TIPO	EMPRESA	DIRECCIÓN AGENTE	TRAP GENÉRICA	TRAP ESPECÍFICA	MARCA DE TIEMPO	VARIABLES ASOCIADAS

Figura 1.7. PDU Trap

- Tipo: Indica el tipo de trama.
- Empresa: tipo de objeto que genera traps, como sysObjectID.
- Dirección Agente: dirección de generación de objetos NetworkAddress.
- Trap-genérica: trap del tipo genérica. Coldstart (0), Warmstart (1), LinkDown (2), etc.
- Trap Específica: presente si no es trap genérica. Trap propia de empresa.
- Marca de tiempo: tiempo desde el último reinicio de la red-entidad.
- Variables asociadas: asociación de variables y valores.

El comando de consulta posee una estructura básica la cual es:

$$snmpPDU - v\ 1 - c\ private|public\ ipAdress\ OID$$

Figura 1.8. Formato de comando SNMP v1

Dependiendo del tipo de PDU, esta estructura puede ampliarse con la adición de más opciones.

1.3.3.1.2. SNMP v2c

En esta versión se introduce la capacidad de comunicación entre 2 NMS, así como 4 nuevas PDU. Las cuales son: GetBulkRequest, Inform Request, Trap v2, y Report.
El formato de la PDU Trap v2 cambia y utiliza el mismo formato de PDU de solicitud, para la PDU GetBulkRequest se introduce una pequeña variación, los campos de Estado de error e Índice de error cambian y son nombrados “Non-Repeates” y “Repeticiones Max” respectivamente [18].

1.3.3.1.3. SNMP v3

Para la versión 3 se dio prioridad a la implementación de más aspectos de control de acceso y seguridad. Se incorporó un modelo de seguridad de usuario USM y un modelo de control de acceso VACM. Con estas mejoras la sintaxis de los comandos SNMP varió quedando de esta manera [18]:

$$snmpPDU - u\ \ NAME - a\ MD5|SHA - A\ \ PASSPHRASE1\ \ - x\ AES|DES\ \ - X\ \ PASSPHRASE2 \\ - l\ \ noAuthNoPriv|authNoPriv|authPriv\ \ ipAddress\ \ oid$$

Figura 1.9. Formato comando SNMP v3

1.3.3.2. MIB

Designado para ser utilizado junto con protocolos de gestión de redes en arquitectura TCP/IP. Es un almacén de información de gestión de objetos que representan un recurso, atributo o un valor compuesto de un elemento de una red [20]. Una MIB utiliza estándares para la especificación de su estructura interna, utiliza SMI (Structure Management Information) el cual está en su versión 2, define cláusulas para identificar atributos de los objetos o nodos, y es una adaptación de ASN.1 (Abstract Syntax Notation one) el cual es un estándar para la representación de tipos de datos y sus valores [21].
Cada objeto tiene un nombre, una sintaxis y una codificación. El nombre es un identificador de objeto (OID) asignado administrativamente el cual especifica el tipo de objeto. La sintaxis del objeto define la estructura de datos del tipo de objeto, se utiliza ASN.1 para este propósito. La codificación sirve para la manera que el tipo es representado cuando es transmitido [21].

Existen tipos de datos catalogados como “super--tipo” utilizados para extender la estructura normal ASN.1 permitiendo agrupar objetos relacionados dentro este mismo tipo. Se utiliza la etiqueta “MACRO” para identificar a este tipo de dato.

Cada objeto definido en una MIB contiene un nombre o identificador de objeto (OID), su sintaxis (estructura de datos abstracta) y su codificación (compatibles con ASN.1).

Una MIB debe orientar la estructura de sus objetos hacia el árbol OID definido en la recomendación UIT-T X.660. Un OID tiene 2 maneras de ser representado, la primera es numérico utilizando números positivos por cada nodo, seguido de un punto, y la otra manera es mediante un nombre único que representa los números de los nodos antes mencionados y también separado por puntos.

El árbol OID posee una rama para colocar MIB empresariales o experimentales, la dirección OID para dicho nodo es:

.1	.3	.6	.1	.4	.1	.X
.iso	.org	.dod	.internet	.private	.enterprise	.X

Figura 1.10. OID empresarial

Para registrar un OID empresarial, se debe contactar con la IANA[6] la cual es la organización oficial para entregar y registrar nombres privados utilizados para identificación por LDAP[7] como SNMP. A continuación, se presenta un pequeño ejemplo de estructuración de una MIB.

[6] **IANA:** Son las siglas de Internet Assigned Numbers Authority y es la entidad que asigna las direcciones IP y otros recursos relacionados a protocolos de internet de manera global.

[7] **LDAP:** Son las siglas de Lightweight Directory Access Protocol, este protocolo permite acceder a un servicio de directorio para buscar información de un entorno de red.

```
Prueba-MIB  DEFINITIONS ::= BEGIN
        IMPORTS
                OBJECT-TYPE
                        FROM SNMPv2-SMI

        empresaX    MODULE-IDENTITY
                LAST-UPDATED            "201806091513Z"
                CONTACT-INFO            direccion empresaX
                ::= { enterprises 1 }

        enterprises OBJECT IDENTIFIER
                ::= { private 1 }

        nodo1 OBJECT IDENTIFIER
                ::= { empresaX 1 }

        objeto1 OBJECT-TYPE
                SYNTAX                  OCTET STRING
                MAX-ACCESS              read-only
                STATUS                  current
                DESCRIPTION             "algo"
                ::= { nodo1 1 }

END
```

Código 1.1. Ejemplo MIB

Una vez listo la estructuración de la MIB, se debe codificar las etiquetas y los valores de los objetos, para dicha actividad se requiere utilizar una codificación específica. Para el caso de las MIB se utiliza las reglas de codificación básicas (BER).

1.3.3.3. BER

Es una técnica de codificación perteneciente a ASN.1. Es un acrónimo de "Basic Encoding Rules". Cada objeto es codificado como un tipo, una longitud y un valor. Estos tipos de codificación son llamados TLV (Type, Length, Value) [21]. Es decir, por cada objeto codificable se deben obtener los valores de las 3 variables mencionadas anteriormente. Para el caso singular de un tipo de dato MACRO, la aplicación de la técnica TLV es recursiva, es decir, se debe aplicar TLV a todos los objetos que se encuentren dentro de una MACRO de manera secuencial.

Tipo	Longitud	Tipo	Longitud	Valor

Figura 1.11. TLV para una macro

- Tipo

El campo tipo es de 8 bits e indica el tipo de dato que viene a continuación. Se conforma de 3 subcampos de longitud definida los cuales son:

Clase		Tipo	Número de clase				
8	7	6	5	4	3	2	1

Figura 1.12. Campo Tipo

La Clase es un número de 2 bits que indica el tipo de clase del dato, se debe regir según la tabla de tipos de datos especificado por ASN.1, los cuales son:

Tabla 1.3. Subcampo Clase

Nombre de Clase	**Número**
Universal	00
Aplicación	01
Contexto Específico	10
Privado	11

El Tipo es un número de 1 bit que indica si el tipo de dato es primitivo (0) o estructurado (1).

El Número de Clase indica el tipo de dato de la clase, los más utilizados son:

Tabla 1.4. Subcampo números de clase

	# de Clase
Boolean	1
Integer	2
Octet String	4
Object Identifier	6
Sequence	16
Sequence Of	16

Existe una versión extendida para representar tipos de datos cuya longitud exceda los 32 valores posibles en la versión normal. Para ello los 5 bits menos significativos se marcan en 1, y existirá un byte más, siempre y cuando el bit #8 de cada byte extra se marque en 1, cuando se marque en 0 significa que ese es el último byte.

- Longitud

La longitud se expresa usualmente con 1 byte, en el que el bit más significativo indica si este byte es el último (0) byte en cuyo caso los 7 bits restantes indican la longitud del valor, o si después vienen más bytes (1), en cuyo caso se indica los bytes necesarios para representar el número que indica la longitud total del campo valor.

- Valor

Es la representación total del valor que representa el tipo, si el tipo en realidad se trataba de un super-tipo, entonces el campo valor representa la longitud total de los tipos contenidos en el super-tipo.

1.3.4. JMX

Java Management Extensions (JMX), es una tecnología propia de Oracle (Java) perteneciente a la edición SE (Standard Edition) que permite de una manera simple la administración de recursos como aplicaciones, dispositivos, servicios y objetos, utilizando un tipo de encapsulación denominado MBean. Cada recurso encapsulado ahora toma la forma de un objeto MBean, cada MBean debe ser registrado en un MBean Server para que puedan ser accedidos mediante aplicaciones de gestión que utilicen también JMX. Este servidor es denominado también “Agente JMX” [22]. Uno de los objetivos de la existencia de JMX fue el poder gestionar software y a la vez elementos de telecomunicaciones al integrar protocolos de gestión de redes.

Una de las virtudes de JMX es que define diferentes métodos de comunicación estándar para poder acceder al agente JMX, estos métodos se dividen en conectores y adaptadores. JMX posee una arquitectura propia, a continuación, se define la composición de dicha arquitectura.

1.3.4.1. Arquitectura

La arquitectura de JMX se divide en 3 niveles:

- Nivel de instrumentación.
- Nivel de agente.
- Nivel de gestión o de servicios.

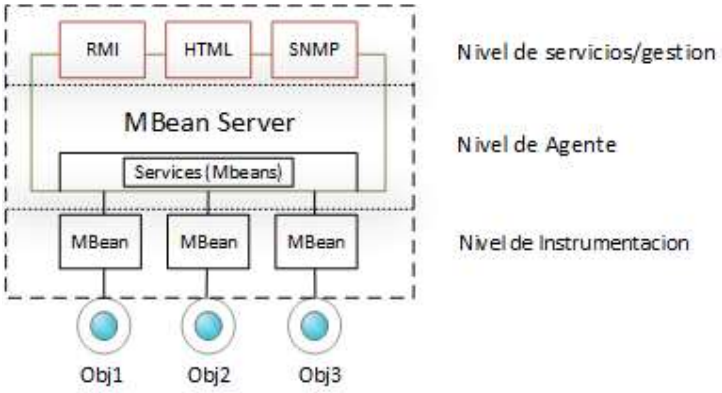


Figura 1.13. Arquitectura JMX

1.3.4.1.1. Nivel de Instrumentación

En esta capa se especifica el recurso a gestionar, algunos ejemplos de recursos se detallan a continuación:

- Aplicaciones.
- Dispositivos.
- Objetos.

Estos recursos son encapsulados como MBeans, un MBean es una clase Java que implementa una interfaz, se puede realizar operaciones utilizando los métodos especificados dentro de los MBeans. De tal manera que en el nivel de instrumentación se crean los MBeans, y estos serán gestionados en el nivel de agente [22].

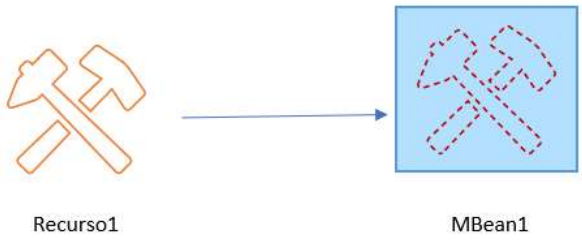


Figura 1.14. Abstracción de recursos

Existen algunos tipos de MBeans, los cuales son:

- MBean Estándar: la forma más simple, la interfaz de gestión se describe según los nombres de los métodos.

- MBean Dinámico: se implementa una interfaz a través de la cual se exponen los métodos, otorga más flexibilidad en la gestión en tiempos de ejecución. Existen 2 tipos de MBeans dinámicos: MBean abierto y MBean modelo.

1.3.4.1.2. Nivel de Agente

En este nivel se contiene al agente JMX el cual se encarga de gestionar los MBeans registrados, y exponerlos para aplicaciones de gestión remotas. Un agente JMX consiste en:

- Servidor MBean: contiene el registro de todos los MBeans registrados en él. Un MBean puede ser registrado por otro MBean, esto da la característica de ofrecer servicios.
- Servicios para manejo de MBeans. Entre los posibles servicios se encuentran Monitores y Timers.

También define un modelo de notificaciones, que permite a través del monitoreo de objetos propagar notificaciones hacia el nivel de gestión.
El MBean debe ser registrado bajo un nombre único denominado "ObjectName" el cual consiste en 2 partes: un nombre de dominio y un conjunto de una o más propiedades [22].

$$[domainName]: property = value$$

Figura 1.15. Sintaxis de declaración de ObjectName

1.3.4.1.3. Nivel de Gestión

Los recursos convertidos en MBeans se deben exponer para el acceso a ellos por parte de aplicaciones de gestión, la forma de exposición es a través de la implementación de interfaces que representan los recursos gestionados.
En este nivel se realizan las implementaciones de los conectores y adaptadores que permitirán el acceso a los MBeans a través de aplicaciones de gestión compatibles con JMX. Estos conectores se habilitarán como servidores en el agente y los clientes serán las aplicaciones de gestión, mientras que los adaptadores de protocolos mapearán las peticiones recibidas con operaciones especificadas en el agente. Tanto conectores como adaptadores también son implementados como MBeans y registrados en el servidor MBean [22].

1.3.4.2. Comunicación

Los métodos de comunicación manejados por JMX permiten que los MBeans expuestos por agentes JMX puedan ser gestionados por aplicaciones fuera de la JVM del agente.

1.3.4.2.1. Conectores

Son nombrados de esta manera porque permiten conectarse con el agente desde una aplicación de gestión remota compatible con JMX, el conector consiste en un conector-cliente y un conector-servidor, el conector servidor se adhiere al servidor MBean y escucha conexiones de clientes. La documentación oficial define a RMI como conector principal [22].

- **RMI**

  Siglas de “Remote Method Invocation”, como se indica en su nombre es un mecanismo para invocar métodos remotos para ambientes distribuidos utilizando Java. Otras tecnologías comparables son CORBA[8] y SOAP[9].

  Utiliza paso de objetos por referencia, recolección de basura distribuida, así como paso de tipos arbitrarios. Una aplicación puede exponer un objeto para que este pueda ser pedido en un determinado puerto TCP, posteriormente acceder a los métodos de dicho objeto [24].

  Para definir un objeto como remoto se debe definir una interfaz y el objeto remoto debe implementar dicha interfaz. La gran desventaja es que es una tecnología única de Java.

  RMI se compone de 3 capas:

  - Capa1 (Proxy): interactúa con la aplicación, las llamadas a objetos remotos y retorno de estos ocurren en esta capa.
  - Capa2 (Referencia Remota): estructura semántica de invocaciones remotas y estrategias para recuperación de conexiones caídas.
  - Capa3 (Transporte): Aquí se sitúa el protocolo de transporte, JRMP (Java Remote Method Protocol) o Internet Inter-ORB Protocol (IIOP) [23].

[8] **CORBA:** Son las siglas de Common Object Request Broker Arquitecture y es un estándar de objetos distribuidos, a diferencia de RMI el propósito del presente es lograr la interoperabilidad entre plataformas.

[9] **SOAP:** Son las siglas de Simple Object Access Protocol y es un protocolo que define la manera que dos objetos se comunican a través de una estructura de datos del tipo XML.

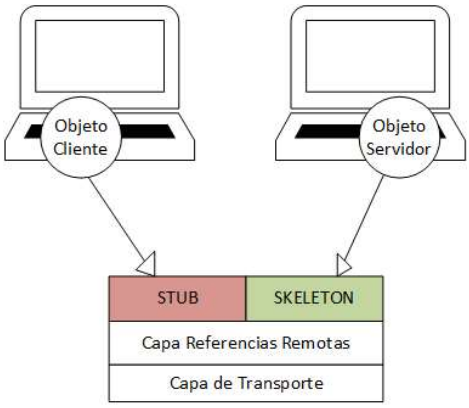


Figura 1.16. Arquitectura de RMI

- Stub y Skeleton

  Un stub es una representación de un objeto local para la conexión con un objeto remoto, puede verse como un proxy para dicho objeto remoto. El propósito del stub es simular los métodos remotos haciéndolos parecer que se realiza una comunicación local. Skeleton es el nombre del stub del servidor.

1.3.4.2.2. Adaptadores

Son nombrados de esta manera porque "adaptan" las operaciones del MBean en el servidor MBean de tal manera que se muestra como una representación del protocolo. Permiten acceder al agente JMX a través de operaciones que son mapeadas en el servidor MBean, un protocolo compatible es SNMP [23].

1.3.4.3. Plataformas de desarrollo

A continuación, se nombran las plataformas de desarrollo de agentes JMX.

1.3.4.3.1. iReasoning

Es una plataforma de desarrollo de agentes SNMP o multiprotocolo, que utilizan el lenguaje Java para la instrumentación en ambos tipos de agentes. Es una plataforma de pago sin embargo existe un tiempo de prueba de 30 días.

Este agente multiprotocolo da soporte para SNMP y TL1. SNMP es el protocolo predilecto para gestión de redes mientras que TL1 también es conocido como protocolo de gestión en telecomunicaciones [24].

1.3.4.3.2. WebNMS

Esta plataforma perteneciente a la compañía de telecomunicaciones Zoho Corp. Permite el desarrollo de agentes SNMP y agentes multiprotocolo basados en una instrumentación en lenguaje JAVA. Para los agentes multiprotocolo utiliza la tecnología JMX.
Los agentes multiprotocolo soportan tecnologías como: SNMP, TL1, HTTP, RMI, CORBA, entre otras. Proveen de un conjunto de herramientas de desarrollo como editores, compiladores y buscadores facilitando la tarea del desarrollador para la prueba o testeo de sus proyectos. Es una multiplataforma totalmente compatible con Linux. Es de pago sin embargo permite utilizar una licencia gratuita por 45 días con todas las características [25].

1.3.5. SISTEMAS DE BASE DE DATOS

Un Sistema de Base de Datos está conformado por un motor de base de datos y una Database Management System (DBMS) o Sistema de Gestión de Base de Datos, el cual es un programa que sirve para manipular una colección de datos lógicamente relacionados entre sí (base de datos) [26].
Los motores de base de datos más utilizados pueden albergar información relacional o no-relacional, y acorde a esto también existirán DBMS compatibles con cada uno. Los DBMS compatibles con el modelo relacional son nombrados RDBMS mientras que los restantes son conocidos como NoSQL-System [27].

1.3.5.1. Motores

Los motores son aquellos que ofrecen las características nuevas por lo tanto son los que más a menudo son evaluadas por los usuarios. Existen comunidades o grupos dedicados a dar un ranking de los motores de bases de datos, uno de ellos es DB-Engines [28], según su ranking de popularidad de usuarios de bases de datos, para el mes de noviembre de 2018 se presentan los 5 primeros lugares de esta manera:

Tabla 1.5. Top 5 de motores de bases de datos

Posición	Nombre
1	Oracle
2	MySQL
3	Microsoft SQL Server
4	PostgreSQL
5	MongoDB

El actual ranking ha ido fluctuando mes a mes durante el presente año, sin embargo, existe un ranking al final de cada año. El ranking del año 2017 el mejor motor fue PostgreSQL. Para la calificación y el posicionamiento del ranking se considera lo siguiente [29]:

- Número de menciones del sistema en la web.
- Frecuencia de búsqueda basándose en Google Trends.
- Frecuencia de discusión técnica.
- Número de ofertas de trabajo.

Para la selección del motor a utilizar durante el presente proyecto, no se consideran aquellos motores que sean licenciados, así como aquellos que no sean relacionales. Después de esta filtración la lista se reduce a: MySQL y PostgreSQL.

1.3.5.1.1. MySQL

MySQL presente desde 1995 ha tenido una alta acogida, tuvo un cambio de dueño en el 2000 al ser adquirido por Oracle, desde ese momento Oracle fue el encargado de las mejoras y mantenimientos, pero bajo políticas contrarias a las iniciales. El mismo creador de MySQL en el momento que esta fue vendida, el creó MariaDB, una RDBMS muy similar en todos los aspectos al MySQL original. Actualmente posee 2 enfoques de versiones, una community que es gratuita bajo licencia GPL[10] y el resto son pagadas más enfocadas al uso empresarial cuyo valor va desde los $2000. Existen cuestionamientos en cuanto al desarrollo de MySQL debido a que Oracle tiene su producto propio en la línea de competencia frente a MySQL [30].

1.3.5.1.2. PostgreSQL

Es una RDBMS de distribución libre, su fecha de creación fue en 1986 como proyecto de la universidad Berkeley, actualmente se ha liberado la versión 11 en Beta. Su reputación ha mejorado debido a su confiabilidad, integridad y el constante desarrollo por parte de la comunidad de desarrolladores detrás de las innovaciones integradas. Una característica superior frente a MySQL es el cumplimiento completo del ACID[11] [30].

[10] **GPL:** Acrónimo de GNU General Public License es una licencia de autoría usada principalmente en el mundo del software libre que garantiza al usuario final la libertad de usar, compartir y modificar el software.

[11] **ACID:** (Atomicity, Consistency, Isolation, Durability). Este conjunto de propiedades debe garantizar la confiabilidad de las transacciones en una base de datos

1.3.5.2. Herramientas de administración

Considerando los motores de bases de datos citados previamente, a continuación, se presentan los mejores DBMS para cada uno de estos motores.

1.3.5.2.1. Workbench

Recomendado el uso desde la propia página de MySQL es una herramienta multiplataforma cuyas principales características son: proveer un modelado de datos, un desarrollo SQL y herramientas de gestión intuitivas para la configuración, gestión de usuarios, respaldos, entre otros. También posee una función para la migración desde otros RDBMS hacia MySQL [31].

1.3.5.2.2. PgAdmin

Esta herramienta es dedicada para la utilización junto a PostgreSQL ya que soporta todas sus características, es multiplataforma y open source. Su interfaz gráfica es intuitiva, realiza un subrayado de colores de sintaxis para una mejor orientación del administrador y está en constante desarrollo considerando recomendaciones de usuarios de la comunidad [32].

1.3.6. DEMONIOS Y TAREAS

Cuando se necesita que una actividad se ejecute automáticamente sin la intervención de un usuario dentro de un pc se tienen algunas opciones que se pueden elegir considerando los requerimientos para dicha acción. A continuación, se describen las opciones enfatizando su utilización.

1.3.6.1. Demonios

Un demonio es un pequeño programa que se ejecuta en segundo plano (no controlado por el usuario) que se encuentra pendiente esperando por un evento que impulse a que cambien su estado y ofrezcan servicios. Los demonios por lo general se utilizan para realizar actividades complejas [33].

Por ejemplo, para la configuración de net-snmp[12] en un ambiente Linux, se debe habilitar el demonio *snmpd* el cual está siempre esperando por peticiones snmp hacia el localhost y el cual, según las restricciones definidas en el archivo de configuración, entregará o no la información requerida.

[12] **Net-SNMP**: conjunto de herramientas para la implementación de agente SNMP en ambientes unix.

1.3.6.2. Tareas

Las tareas son más simples que los demonios, las tareas pueden ser definidas en un archivo de configuración llamado crontab así como en otro archivo llamado rc.local ambos ubicados en el path `/etc`, la diferencia radica en que aquellas tareas definidas en el archivo `rc.local` son ejecutadas al iniciar el sistema mientras que aquellas tareas definidas en el archivo crontab se especifica cuando ejecutarse. Cron es un demonio cuya finalidad es gestionar las tareas definidas en él. Las tareas pueden ser comandos o scripts del Shell que necesitan ser corridas periódicamente o cada determinado tiempo, el cual es especificado al momento de añadir cada nueva tarea. Usualmente las tareas son utilizadas cuando se desea programar el mantenimiento de un sistema.
Existe una estructura determinada para especificar tareas en el cron el cual es el siguiente:

Minutos	Horas	Día del mes	Mes	Día de la semana	Script
0 – 59	0 – 23	1 – 31	1 – 12	0 – 6 (Domingo->0)	PATH del script

Figura 1.17. Estructuras de tareas para cron

Cada uno de los campos debe ser escrito dejando un espacio entre ellos. Se puede utilizar el símbolo asterisco (*) para especificar todos los valores de un campo. Por ejemplo:
“En el minuto 10 de las 3 de la mañana, de cada día del mes, de todos los meses y cuando sea jueves”.

```
10 3 * * 4 /etc/miScript.sh
```

Figura 1.18. Ejemplo de declaración de script en crontab

En cada campo se permite también utilizar rangos, por ejemplo, si se desea que únicamente se ejecute de lunes a viernes, entonces en el campo del día de la semana se escribiría 1-5.

1.3.7. DIAGRAMA UML DE DESPLIEGUE

Un diagrama de despliegue pertenece al grupo de diagramas UML y permite modelar la ubicación de los elementos de software dentro de los elementos de hardware, y las relaciones existentes entre ambos elementos. Los elementos por utilizar son: nodos, artefactos, componentes y rutas [34].

- Los nodos son presentados por cajas y representan hardware o dispositivos, contienen una <<etiqueta>> la cual indica que tipo de hardware es. Los nodos contienen otros elementos como los artefactos y componentes.

Figura 1.19. Elemento Nodo

- Los artefactos son presentados por cuadrados y son aquellos que se despliegan en el sistema, como programas, librerías o archivos de configuración.

Figura 1.20. Elemento Artefacto

- Los componentes tienen la misma apariencia que los artefactos, pero con su propia etiqueta además de un logo esquinero. Los componentes son manifestaciones realizadas por los artefactos, es decir, su existencia o actualización depende de los artefactos.

Figura 1.21. Elemento Componente

- Las rutas o conexiones son presentadas por líneas y estas interconectan todos los demás elementos, permiten ser etiquetadas para indicar el tipo de comunicación que está siendo utilizado entre los elementos.

Figura 1.22. Elemento Ruta o Conexión

1.3.8. KANBAN

Kanban significa “tabla visual” es un concepto de producción de justo a tiempo (JIT: Just in Time). Forma parte de las metodologías Lean, las cuales buscan reducir desperdicios, variabilidad e inflexibilidad en una cadena de valor de una empresa. Kanban está enfocado en el desarrollo de software se centra en controlar el WIP (Work in Progress) considerando una lista de actividades priorizadas que sirve como fuente de alimentación para el WIP. Esta lista suele denominarse Product Backlog. Cuando se dice que se

controla el WIP lo que se quiere decir es que gestiona la lista de actividades actuales, cuando la lista se reduce, entonces se toma la siguiente actividad del Backlog y se desplaza hacia el WIP. Kanban permite la visualización sencilla para cualquier persona participante del proceso de un proyecto a través de un tablero en el cual se especifican las actividades a realizar y su estado actual. En el proceso de trabajo normal utilizando Kanban se pueden distinguir 3 pasos claros [35]:

- Visualización del Workflow.
    - División del proyecto en ítems (actividades) que serán representados por etiquetas y colocados en el tablero o Kanban Board.
    - Uso de columnas con títulos definidos en los que se situarán las etiquetas e indicarán en qué etapa del flujo de trabajo o workflow se encuentran.
- Limitación del WIP.
    - Asignación de límites respecto a cuantos ítems pueden ser procesados a la vez en cada columna del workflow.
- Medición del Lead Time.
    - Un objetivo de Kanban reducir el desperdicio del tiempo por lo que permite medir en cada etapa del workflow el ítem, encontrando el tiempo empleado para dicho ítem.

No existe una forma definida para la utilización del tablero, todo depende del enfoque de quien lo vaya a utilizar y su necesidad, pero considerando la condición de hacerlo entendible para cualquier persona.
Existen recomendaciones para identificar un mal dimensionamiento del límite del WIP; si en una revisión del tablero no se distingue ningún cuello de botella entonces el límite es muy bajo, y, si en una revisión se observa que ítems permanecen en una etapa del workflow demasiado tiempo entonces el límite es muy alto. La Figura 1.23 muestra el tablero a utilizar durante el desarrollo del presente prototipo.

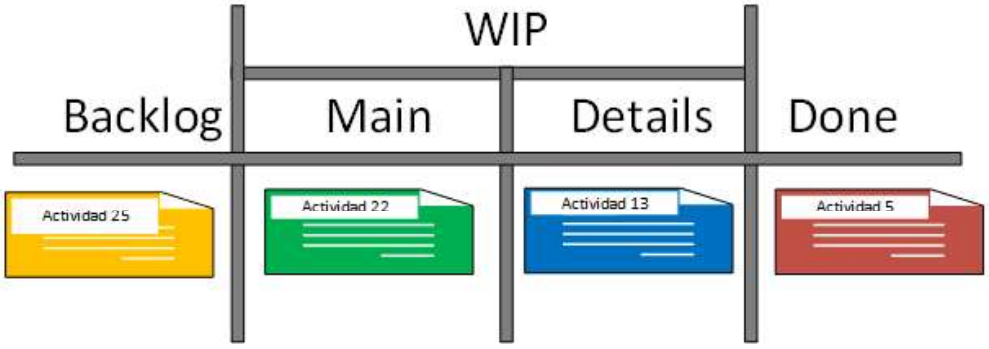


Figura 1.23. Ejemplo de Tablero Kanban

2. METODOLOGÍA

Durante el presente capítulo se presentan las etapas correspondientes al diseño y la implementación del prototipo, empezando por la elaboración del tablero Kanban con las actividades por realizar, la recolección de los requerimientos y el diseño de cada módulo del prototipo, para después abordar el proceso de implementación, el cual también se desarrollará considerando los módulos planteados.

Para alcanzar un mejor desarrollo del proyecto, se dividió en 3 módulos o secciones:

- Adquisición de valores.
- Configuración del agente.
- Codificación y configuración de la aplicación y de la base de datos.

2.1. DISEÑO DEL PROTOTIPO

Los 3 módulos del prototipo son los tres pilares fundamentales para el funcionamiento del prototipo. El primero es el módulo de adquisición, este abarca lo necesario para obtener los valores obtenidos a través de los sensores y procesarlos a través del Arduino y RPi, el módulo agente abarca la utilización del agente JMX, la utilización de los valores de los archivos de texto y de la MIB propia, para finalmente en el módulo de visualización presentar en la aplicación de escritorio los valores obtenidos a través del agente JMX y los valores históricos almacenados en la base de datos.

2.1.1. TABLERO KANBAN

En la Figura 2.1 se puede observar el tablero Kanban para el diseño del prototipo, la lista de actividades se presenta en la primera columna al lado izquierdo, estas actividades forman parte del backlog actual, estas actividades fueron previamente ordenadas según el proceso de desarrollo del proyecto, la primera actividad a realizar se ubica en la sección del WIP en la subsección Main. Al terminar la sección se actualizará cada tablero.

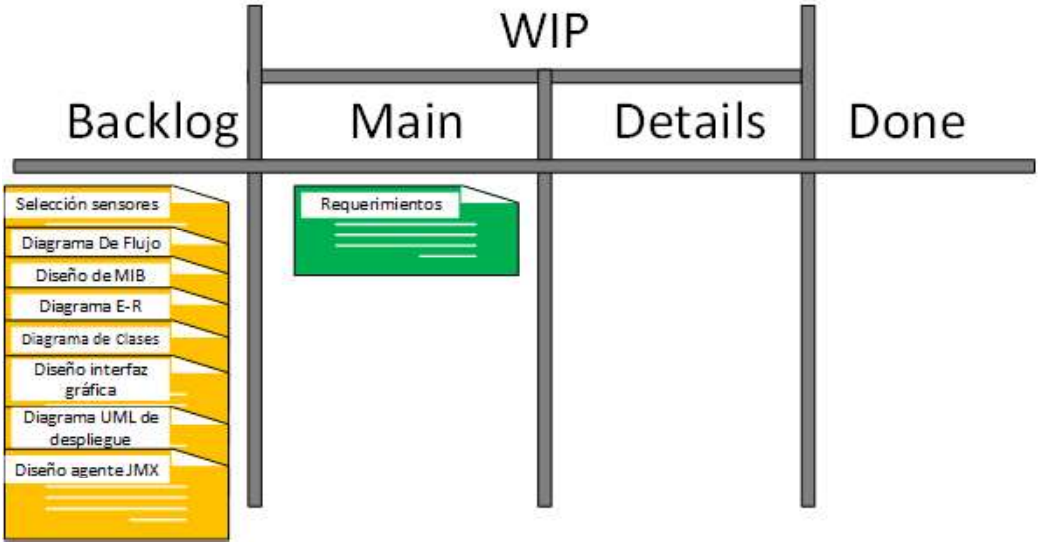


Figura 2.1. Tablero Kanban Fase de Diseño

2.1.2. ANÁLISIS DE REQUERIMIENTOS

El primer paso del desarrollo de este proyecto es presentar el análisis de requerimientos, el cual es empleado para la determinación de las necesidades que debe cubrir el prototipo.

2.1.2.1. Encuesta para determinación de requerimientos

Para la obtención de los requerimientos del cliente se realizaron encuestas al personal del área administrativo de la empresa Yongping las mismas que pueden ser encontradas en el ANEXO I. A continuación, se presentan los resultados de las preguntas efectuadas.

- Pregunta 1. ¿Considera posible que, durante el periodo laboral los empleados de producción pueden estar expuestos a elementos en el ambiente que si se presentasen en altas concentraciones podrían ser perjudiciales para su salud? Respuestas: considere 5 como muy posible y 1 como nada posible.

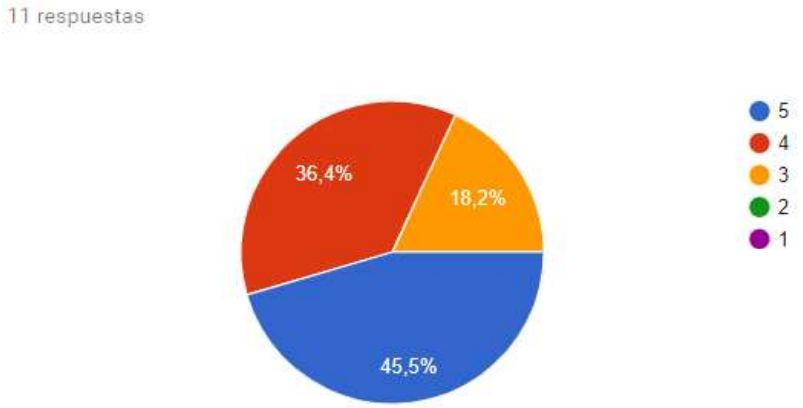


Figura 2.2. Gráfico correspondiente a respuestas de pregunta 1.

- Pregunta 2. Considerando la pregunta anterior. ¿Considera útil monitorear estos elementos en aquel lugar donde podrían presentarse? Respuestas: si, no.

Figura 2.3. Gráfico correspondiente a respuestas de pregunta 2.

- Pregunta 3. ¿Estaría dispuesto a monitorear estos elementos desde la computadora de su estación de trabajo? Respuestas: si, no.

Figura 2.4. Gráfico correspondiente a respuestas de pregunta 3.

- Pregunta 4. Considerando el costo monetario en la adquisición de una solución comercial de monitoreo. ¿Preferiría usted invertir en el desarrollo de una solución acorde a sus necesidades? Respuestas: si, no.

Figura 2.5. Gráfico correspondiente a respuestas de pregunta 4.

2.1.2.2. Obtención de requerimientos

Considerando las respuestas mostradas en la Figura 2.2, existe un entendimiento acerca de posibles peligros para la salud de las personas por altas concentraciones de elementos en el ambiente. La Figura 2.3 indica que el 100% de los encuestados considera útil monitorear estos elementos, así como monitorearlos desde su estación de trabajo según lo indicado en la Figura 2.4. La Figura 2.5 indica que los encuestados preferirían invertir en una solución acorde a sus necesidades, mientras que, con respecto a la compatibilidad y crecimiento de una empresa.

La empresa Yongping se dedica a la manufactura de ventanas, puertas y pisos en PVC, además de prefabricados de hormigón; en el proceso de producción, se desprenden pequeños residuos sólidos y gases que contaminan el aire de las áreas de producción. Estos parámetros al llegar a una concentración elevada se convierten en contaminantes pues llegan a ser peligrosos para la salud de los trabajadores y aquí nace la necesidad de la empresa de monitorear activamente las concentraciones de estos parámetros ambientales.

El planteamiento de este proyecto trata sobre el desarrollo de un prototipo de monitorización de monóxido de carbono (CO) y de material particulado fino ($PM_{2.5}$).

2.1.3. DISEÑO DEL MÓDULO DE ADQUISICIÓN

Este módulo trata sobre el método de adquisición de los valores captados por los sensores conectados al Arduino, la comunicación y la adquisición de estos datos desde el Raspberry Pi, la forma de lograr la automatización de dicho proceso y el tipo de archivo utilizado para guardar dichos valores.

2.1.3.1. Selección de sensores

Los parámetros ambientales por medir son 2: monóxido de carbono (CO) y material particulado fino $PM_{2.5}$. En el mercado existen diferentes tipos de sensores cuyos precios varían dependiendo de la calidad de materiales y la precisión de cada uno. Para el presente proyecto se definió en el plan de trabajo la utilización de sensores de bajo costo pues se prioriza la funcionalidad y no la precisión de los sensores, por tal motivo la elección se basa en el costo. En la categoría de bajo costo para la detección de gases existe la familia de sensores MQ, cada modelo puede detectar uno o más gases o sustancias químicas. El sensor cuya detección se centra en el monóxido de carbono es el modelo MQ-7.

El material particulado fino es aquel cuyo diámetro es menor a 2.5 um, existe un tipo de sensor que logra detectar tales partículas gracias a la utilización de fotodiodos y de fototransistores. El nombre común es Sensor $PM_{2.5}$. A continuación, se presenta un resumen de cada uno de los sensores a utilizar.

2.1.3.1.1. MQ-7

El sensor MQ-7 percibe hidrógeno (H2) y monóxido de carbono (CO), sin embargo, tiene mayor sensibilidad para monóxido de carbono. En el mercado se puede adquirir el módulo de MQ-7 el cuál a diferencia de solo el sensor, este módulo presenta al sensor acompañado de un circuito de acondicionamiento [36]. Este módulo posee 4 pines los cuales se muestran en la Figura 2.6.

Figura 2.6. Módulo sensor MQ-7 *[37]*

Este sensor requiere que se cumplan ciertas condiciones que garantizan un funcionamiento apropiado, por ejemplo, previo a cualquier medición se recomienda realizar un periodo de calentamiento y calibración, esto trata sobre alimentar al sensor durante 48 horas ininterrumpidas a un período variable de alimentación, a fin de que pueda garantizar un funcionamiento apropiado en la fase de lectura. Datos importantes sobre el sensor se presentan en la Tabla 2.1:

Tabla 2.1. Condiciones de funcionamiento sensor MQ7

Símbolo	Nombre de parámetro	Condición técnica
$V_H(H)$	Voltaje de calentamiento (alto)	5 V
$V_L(L)$	Voltaje de calentamiento (bajo)	1.4 V
$T_H(H)$	Tiempo de calentamiento (alto)	60 seg
$T_L(L)$	Tiempo de calentamiento (bajo)	90 seg

2.1.3.1.2. Sharp GP2Y1010AU

El sensor de $PM_{2.5}$ de Sharp identifica las partículas cuyo diámetro es menor a 2.5 micrones también conocidas como partículas finas, este tipo de sensor es frecuentemente utilizado en sistemas de purificación de aire [38].

Figura 2.7. Sensor de PM2.5 *[39]*

La Figura 2.7 muestra el sensor, se puede apreciar que este posee 6 pines los cuales se detallan en la Tabla 2.2:

Tabla 2.2. Pines sensor PM2.5

Nombre de pin	Descripción
V led	5 V
LED-GND	Tierra de led
LED	Señal de control de led
S-GND	Tierra de fuente
Vo	Señal de lectura
Vcc	Alimentación

Las características requeridas para su correcto funcionamiento se enlistan en la siguiente la Tabla 2.3:

Tabla 2.3. Condiciones de funcionamiento del Sensor PM2.5.

Característica	Descripción
Voltaje:	3.3 – 7 V
Temperatura de operación:	-10 a 65 grados centígrados
Corriente de consumo:	Max:.20mA

2.1.3.2. Diagramas de flujo

A continuación, se presentan diagramas de flujo correspondientes a los procesos seguidos para la adquisición, comunicación y guardado de los valores captados por los sensores.
La Figura 2.8 muestra el diagrama de flujo de los datos manejados en el Arduino, se evalúa si fue seleccionado el proceso de calibración o no, cuando se selecciona calibración se procede a efectuar dicho proceso, mientras que si no es seleccionado entonces se procede al proceso de lectura de los sensores, se obtendrán los valores y se guardarán en variables, estas variables formaran parte de un arreglo de datos, a partir de esto se obtendrán valores de un FCS[13] que en la recepción será utilizado para corroborar la veracidad de la información recibida. Para la conformación de una trama se requiere de banderas de inicio y de fin de trama que contendrán en campos internos los datos y al FCS, una vez listo se enviará por el puerto serial hacia el RPi. Este programa funcionará permanentemente en el Arduino.

[13] **FCS:** (Frame Check Sequence) campo utilizado para detección de errores en una trama, su funcionamiento depende de cada implementación.

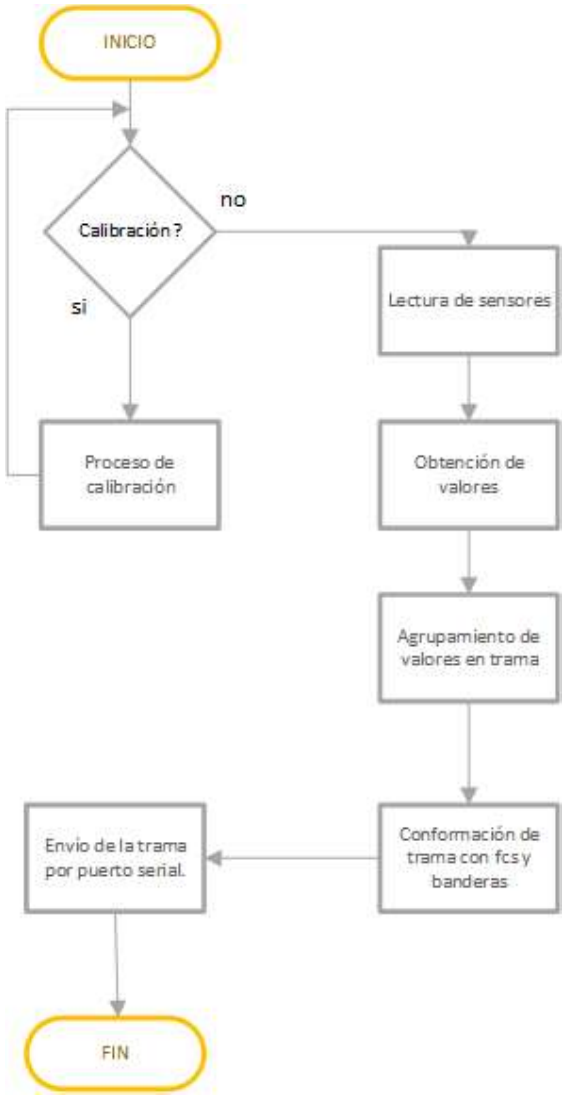


Figura 2.8. Diagrama de flujo Arduino

El siguiente diagrama de flujo correspondiente a la Figura 2.9, muestra el proceso de obtención de datos desde el RPi a través de la lectura del puerto serial. Por defecto el puerto de lectura en el RPi se ubica en el path `/dev/ttyACM0`. El script lee los caracteres ingresados por el puerto esperando recibir la bandera que es el valor entero 126, el cual corresponde al caracter ASCII "~", después todo aquel carácter que no sea una bandera es adjuntado en un arreglo, y cuando recibe una bandera entonces cierra la trama y direcciona a la ejecución del resto de funciones del script.

La siguiente función se encarga de identificar los campos de la trama, y comparar si la información recibida es la correcta, comparando con los datos del campo FCS, si es que la información es correcta entonces se continua con el resto de las funciones sino se cancela el proceso, se desecha la lectura y se espera por otra trama.

La siguiente función permite obtener los valores enviados, estos valores después son transformados al valor de presentación, a través de la aplicación de la fórmula de calibración, y entonces son finalmente guardados, cada uno en su archivo de texto correspondiente.

Figura 2.9. Diagrama de flujo RPi

Los archivos de texto serán utilizados como fuente de información del Agente JMX.

2.1.3.3. Configuración de tarea

Para la codificación de la tarea a añadir en el archivo de tareas se procede a ubicar el archivo crontab el cual se localiza en el directorio contenedor de los archivos de configuración, el PATH es: `/etc/crontab`.

En este archivo cada tarea que se desee ejecutar periódicamente debe ser definida según la estructura requerida. Según los requerimientos el script de lectura debe ser ejecutado cada minuto por lo cual la estructura quedaría de la siguiente manera:

Figura 2.10. Formato de tarea en crontab

También se da uso del archivo `rc.local` para la ejecución de un script a ejecutarse una vez iniciado el RPi. En este archivo no se utiliza el formato del comando por usar en el archivo crontab. Para el programa que será cargado en el Arduino, se utilizará el propio IDE Arduino para la codificación y su lenguaje predefinido. Y, para los scripts a ejecutarse en el RPi, se utilizará el lenguaje de programación Python y el IDE Spyder [40] así como scripts de bash de Linux.

2.1.4. DISEÑO MÓDULO AGENTE

Este módulo trata sobre la generación del agente JMX y todos los requisitos necesarios para su compilación, entre ellos están la creación de la MIB y la utilización de los archivos de texto que albergan los valores captados por los sensores como fuente de ingreso de información para las variables definidas en la MIB.

2.1.4.1. Diseño de la MIB

La MIB propia por implementarse se utilizará para definir los objetos que se utilizarán para monitorear los parámetros establecidos previamente. Para el diseño de la presente MIB es de vital importancia definir los tipos de los objetos a ser utilizados para albergar la

información, así como la estructura de esta para el acceso ordenado a dichos recursos priorizando el aprovechamiento de nodos y objetos en la MIB.

Dentro del árbol de las MIB existe una rama específica para albergar MIBs propias de empresas, mismas que albergan objetos definidos en su totalidad por dichas empresas. El OID de esta rama es `.1.3.6.1.4.1`, cuya equivalencia textual es: `.iso.org.dod.internet.private.enterprises [20].`

2.1.4.1.1. Definición de objetos

Según los requerimientos son 2 los parámetros a captar mediante los sensores, por lo cual serán 2 los objetos que presentarán dichos valores, sin embargo, se necesita también mostrar el nombre específico de cada uno de estos sensores por lo cual se utilizarán 2 objetos más para entregar dicha información.

Los nombres de los objetos nacen de acrónimos de los parámetros que se obtienen para tener una clara referencia al momento de implementar la MIB. De esta manera los objetos que representan los valores de los parámetros ambientales quedan de esta manera: valorCO y valorPM25. Mientras que los objetos que representan los nombres de los parámetros ambientales quedan de esta manera: sensCO y sensPM25.

Estos 4 objetos definidos previamente son los obligatorios para poder representar los datos recogidos por los sensores, sin embargo, se encontró la necesidad de definir más objetos para poder entregar información considerada también importante, como por ejemplo el estado de cada uno de los sensores presentes. Por lo que 2 objetos también se definirán, cada uno representando el estado de cada sensor. Dichos objetos serán: estado CO y estado PM25.

También se encontró la necesidad de utilizar un objeto para obtener el tiempo actual del agente, es decir, el tiempo del RPi. Para la representación del tiempo se utilizó 2 objetos, uno que representará el tiempo en formato Long, el cual únicamente será entendible para el código de la aplicación de escritorio, y otro que sea legible para el ser humano. Los objetos serán: tiempoLocalGauge y tiempoLocalString.

2.1.4.1.2. Elección de tipos

El nodo que alberga los nombres de los parámetros ambientales va a manejar únicamente la descripción textual de los sensores involucrados, y estos servirán únicamente para mostrar información útil para la persona que lo lea, por tal motivo estos objetos serán del tipo:

- OCTET STRING.[14]

[14] **OCTET-STRING**: una cadena de 0 o más bytes usado para representar cadenas de texto.

Para la definición del tipo de los objetos que representan los valores de los parámetros ambientales se consideraron aspectos relacionados al funcionamiento, en primera instancia sería necesario un tipo de número, el cual puede ser INTEGER[15] o GAUGE32[16] sin embargo, considerando que el valor calibrado podría mostrar valores en decimales el tipo de datos de los objetos de la presente MIB será:

- OCTET STRING.

Aquellos nodos que manejarán el estado de los sensores entregarán únicamente información binaria. Por lo tanto, el tipo de datos de estos objetos será:

- INTEGER.

Mientras que, para los objetos que entregarán la fecha y hora del sistema, como se mencionó con anterioridad, uno de ellos entregará información que será utilizado para la manipulación desde la aplicación de escritorio, este objeto deberá ser de un tipo que sea compatible para la entrega de información por lo cual es del tipo:

- GAUGE32.

Finalmente, para el objeto que entregará la fecha y hora en formato legible para el ser humano, deberá ser presentado en un tipo:

- OCTET-STRING.

2.1.4.1.3. Elección de Acceso máximo

En una MIB un objeto puede definirse de 4 tipos de acceso diferentes, los cuales son; `read-only, read-write, write-only y not-accessible`. Esto es principalmente útil para el uso en agentes SNMP pues permite limitar el acceso para los usuarios no registrados y proteger los valores de los objetos. En el caso del prototipo también es aplicable pues a través de la aplicación de escritorio el código permitirá obtener los valores para la graficación, sin embargo, como también puede accederse a los valores a través de consultas SNMP al agente entonces habría la posibilidad de ejecutar consultas SET a dichos objetos y alterar sus valores. Por tal motivo debe elegirse el tipo de acceso de cada objeto de la MIB definida.

Para los dos objetos que entregan información del nombre de los sensores, el tipo de acceso será `read-write`, pues esta información podría ser cambiada si los sensores son reemplazados.

Para los dos objetos que entregan los valores de los sensores, el tipo de acceso será `read-only`, pues los valores no pueden ser alterados por nadie.

[15] **INTEGER**: número de 32 bits

[16] **GAUGE**: número de hasta 32 bits, pero además permite definir umbrales y ser monitoreado.

Para los dos objetos que entregan información sobre el estado actual de los sensores, el tipo de acceso será `read-write`, pues esta información está sujeta a la condición de funcionamiento actual de los sensores, es decir, si están conectados o si están siendo reparados y reemplazados.
Finalmente, para los dos objetos que entregan la hora actual del agente, el tipo de acceso será `read-only`, pues esta información no puede ser alterada por nadie.

2.1.4.1.4. Estructura de la MIB

Como se mencionó en párrafos anteriores se subdividió los objetos dentro de 4 grupos o nodos diferentes, cada uno de estos grupos son de diferentes tipos. Por lo tanto, cada grupo posee su OID propio y ambos se encuentran dentro de una misma rama padre, o nodo padre, el cual es definido utilizando el nombre de la ubicación física a colocar el RPi en la empresa, mientras que el nodo padre primario ubicado justo por debajo del nodo Enterprises, es el nombre que hace referencia a la empresa. Por lo que la estructura de la MIB se representa en la Figura 2.11:

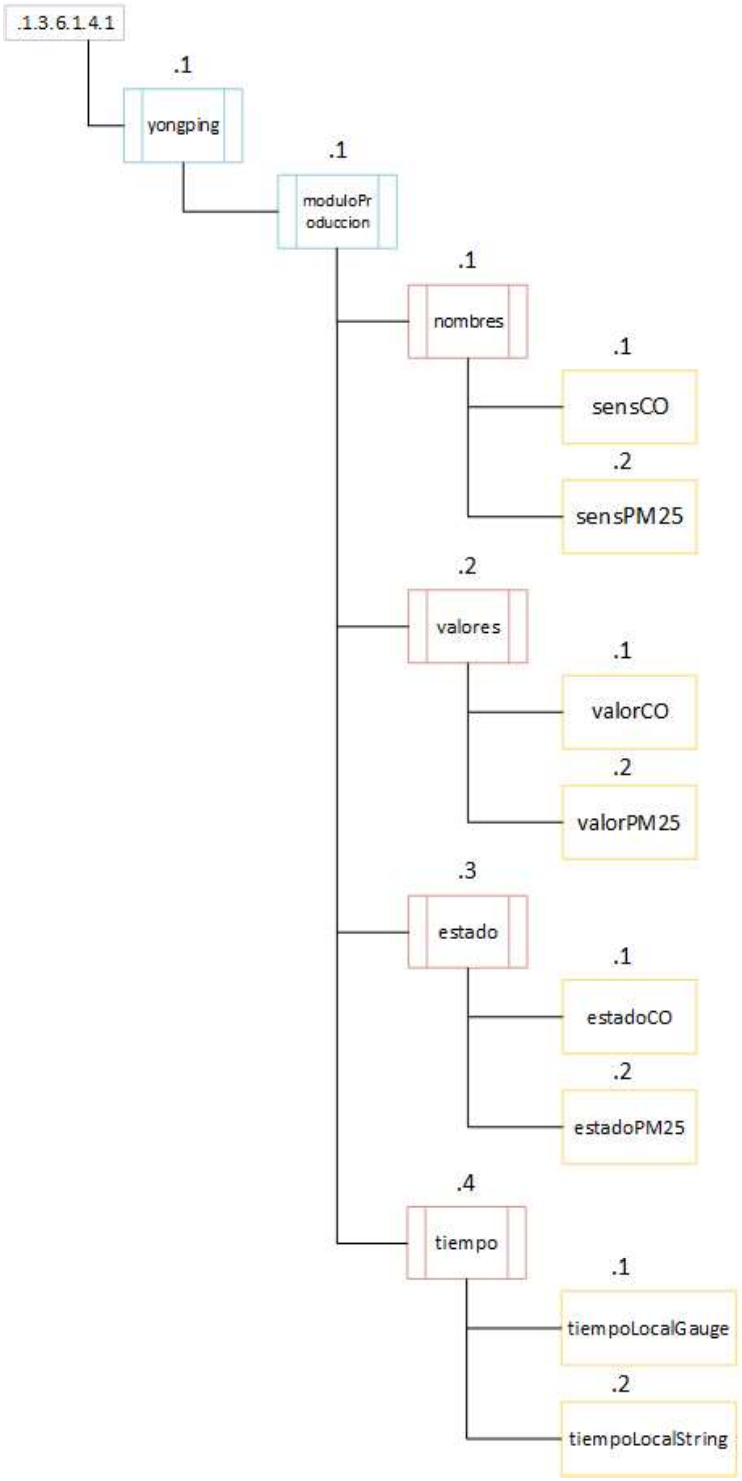


Figura 2.11. Árbol de la MIB

2.1.4.2. Diseño agente JMX

Para el desarrollo del agente JMX en este proyecto se eligió la utilización de la plataforma WebNMS por los siguientes motivos:

- Completa compatibilidad con JMX brindando un soporte de varias tecnologías.
- Soporte durante el tiempo de evaluación.
- Amplia documentación sobre uso del agente.
- Uso de MIB para generación de agente.
- Acceso a código esqueleto del agente para añadir características propias.

El programa dedicado para la generación y compilación de agentes JMX de la empresa WebNMS es conocido como JmxCompiler, este programa permite la utilización de archivos MIB para la generación del agente y la elección de múltiples protocolos para su uso como adaptadores y conectores [25].

Al escoger la MIB a utilizar, el programa genera archivos y código esqueleto por defecto, este código generado es lo suficiente para compilar el agente y que este sea funcional, sin embargo, también es posible agregar código propio para agregar funcionalidades al agente. En ese momento se debe añadir código extra para la obtención de los datos definidos en los objetos de la MIB. A continuación, en la Figura 2.12 se presenta una abstracción de la idea.

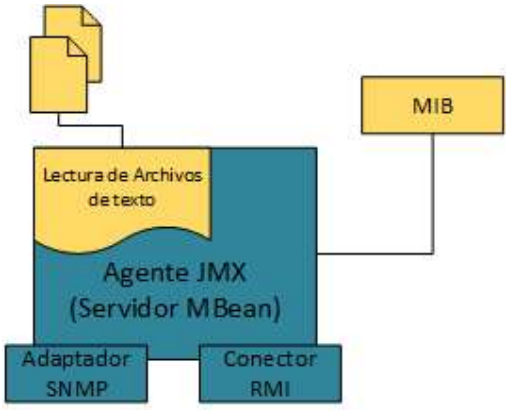


Figura 2.12. Abstracción gráfica de modificación del agente JMX

En la Figura 2.12 se identifican 2 colores: azul y amarillo, el azul corresponde a las características generadas automáticamente por el programa JmxCompiler, mientras que aquello de color amarillo corresponde a lo que se debe configurar y añadir para una correcta compilación del agente.

En los métodos generados para manipular los datos de los objetos se deberá definir la utilización de archivos de texto para recoger la información. También se debe añadir

código para obtener la información de la hora propia del RPi para la presentación a través del objeto de hora.

Finalmente se deberá definir los tipos de comunicación a soportar. Este agente debe permitir entregar la información utilizando un adaptador SNMP, así como utilizar un conector RMI para la conexión desde la aplicación de escritorio.

2.1.5. DISEÑO DEL MÓDULO DE VISUALIZACIÓN

El módulo comprende los componentes finales del sistema que permiten la visualización de datos desde una aplicación de escritorio, la base de datos alberga la información obtenida desde los archivos de texto de cada sensor, esta información es utilizada para la presentación de reportes en la aplicación. Se accede a esta información desde la aplicación de escritorio para poder representarla mediante un gráfico. Esta aplicación también permite conectarse por RMI y obtener los valores de los objetos definidos en la MIB que fueron encapsulados en MBeans en el agente JMX y puestos a disposición para acceso por RMI.

2.1.5.1. Base de datos

La base de datos se localiza en otro Raspberry Pi el cual actuará como PC. Este RPi almacenará información inherente a los sensores y al RPi contenedor del agente.

2.1.5.1.1. Diagrama entidad-relación

El diagrama entidad relación nos permite presentar de manera gráfica las relaciones entre las entidades de la base de datos, para el diseño de esta se utilizaron los siguientes requisitos como base para el diseño:

- En el RPi que contiene el agente podrán estar conectados uno o más sensores a través de su conexión con el Arduino Uno, este RPi se ubicará en un determinado sitio en la empresa.
- Los sensores deben entregar un valor de acuerdo con los parámetros medidos. La principal diferencia de los sensores es el tipo de parámetro.
- En cada lectura es importante conocer además del valor medido, el sensor del que proviene dicho valor.
- Se utilizará una entidad para consulta de nivel o rango de contaminación según el valor, esta entidad no tiene relación con las entidades definidas previamente.

La Figura 2.13 detalla el modelo Entidad – Relación.

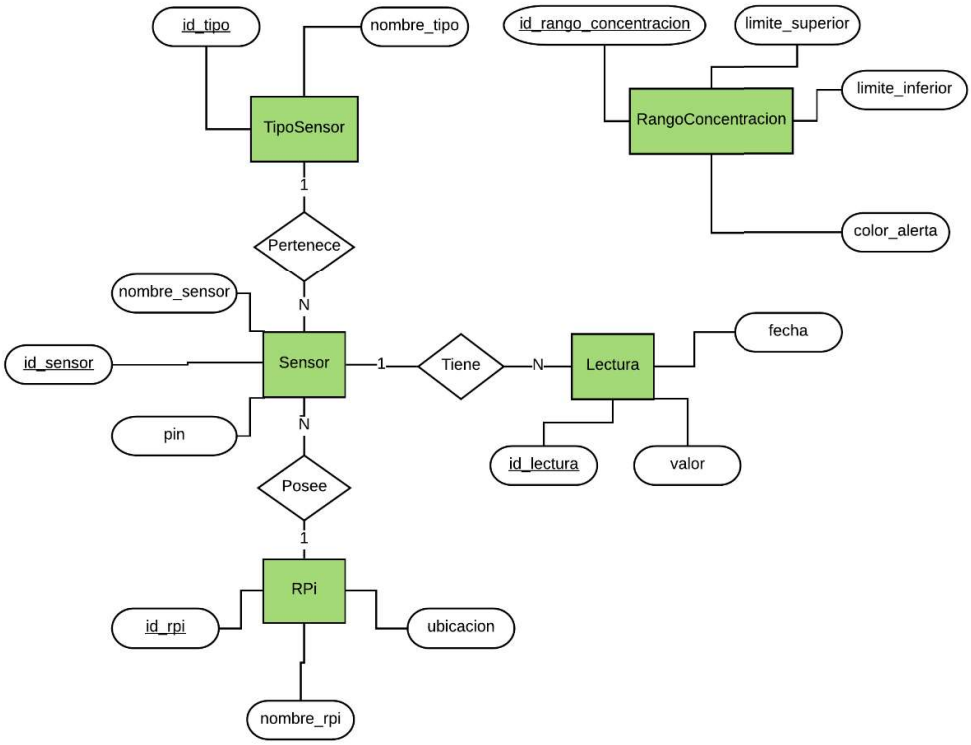


Figura 2.13. Modelo Entidad – Relación

Se consideró la existencia de 4 entidades relacionadas y 1 entidad aislada:

- RPi: representa al RPi utilizado, este posee los atributos:
  - id_rpi: es su identificación única.
  - nombre_rpi: es su nombre asignado al RPi utilizado.
  - ubicación: es el nombre de la ubicación física en la que se encuentra.
- SENSOR: hace referencia a los sensores que entregan información al RPi, este posee los atributos:
  - id_sensor: es su identificación única.
  - nombre_sensor: el nombre del sensor.
  - pin: representa el número del pin de lectura analógica asociado al sensor en el Arduino.
- TIPO_SENSOR: representa el tipo de parámetro a medir, este posee los atributos:
  - id_tipo: es su identificación única.
  - nombre_tipo: el nombre del tipo de sensor.
- LECTURA: alberga la información recopilada por los sensores.

- id_lectura: es la identificación única de cada inserción de datos.
- valor: es el valor recogido por un sensor.
- fecha: representa la fecha en la que se realiza el ingreso del nuevo dato.

- RANGO_CONCENTRACION: utilizado para referencia de rangos de afectación de los parámetros ambientales.
  - id_rango_concentracion: es la identificación única de cada rango.
  - limite_inferior: define el límite inferior de cada rango definido.
  - limite_superior: define el límite superior de cada rango definido.
  - color_alerta: define el color de la alerta según el rango ingresado.

En cada una de las entidades se consideró la utilización de cada identificador único como su clave primaria. Con el diagrama entidad relación listo, se presenta el diagrama relacional, el cual servirá para la generación de los scripts de la base de datos.

2.1.5.1.2. Diagrama relacional

En la Figura 2.14 se presenta el Diagrama Relacional de la base de datos, se utilizó la herramienta LucidChart[17].

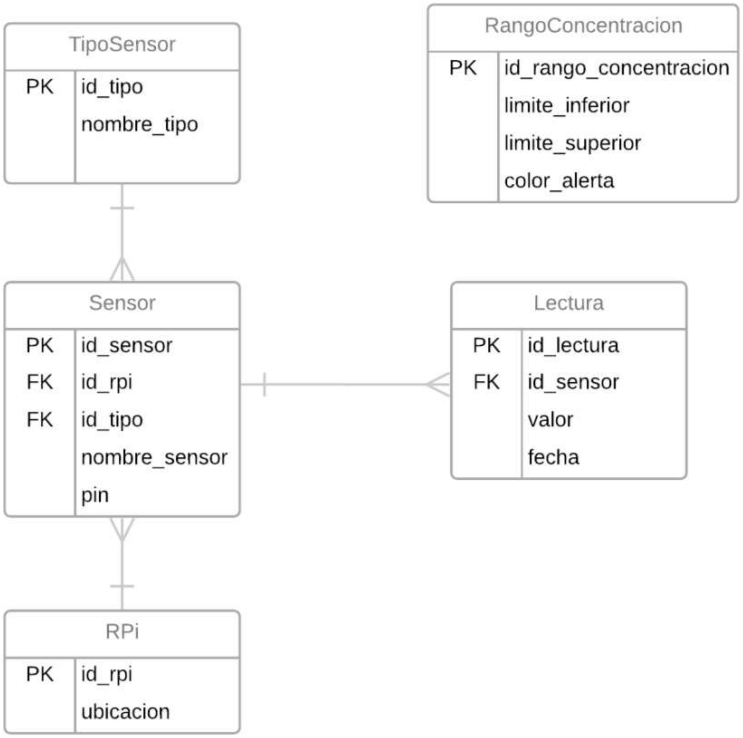


Figura 2.14. Diagrama Relacional de Base de Datos

[17] **LucidChart**: herramienta en línea para la realización de diagramas UML.

En el diagrama relacional, cada entidad previamente definida es representada ahora como tabla dentro de la base de datos con sus respectivos atributos, claves primarias y claves foráneas. Las tablas relacionadas son vinculadas con enlaces, dependiendo de la cardinalidad los enlaces presentan una o más ramas. Por ejemplo, la relación existente entre la tabla Sensor y la tabla Lectura indica una relación de uno a muchos respectivamente, es decir, por cada sensor pueden existir varias lecturas por lo cual aquella tabla donde indica el plural implementará como clave foránea la clave primaria de la tabla donde indica singular. La tabla "RangoConcentración" no presenta relaciones debido a que su existencia fue diseñada para la utilización en la ubicación de los valores obtenidos a través de la conexión RMI desde la aplicación.

2.1.5.1.3. Selección del motor y herramienta de administración

La Tabla 2.4 muestra las características de los motores de base de datos.

Tabla 2.4. Comparación MySQL vs PostgreSQL

MySQL		**PostgreSQL**	
Ventajas	**Desventajas**	**Ventajas**	**Desventajas**
Gran popularidad	Facilidad de uso	Gran popularidad	Velocidad
Velocidad	Limitación funcional de SQL	Compatibilidad con más lenguajes de programación	Configuración más complicada
Producto open source	Características limitadas por licencia	Proyecto open source	
	Poco desarrollo desde su nueva casa en 2009	Mayor número de características y plugin.	

Después de presentar las características de los motores de bases de datos MySQL y PostgreSQL [31], se eligió la utilización de PostgreSQL debido a la comunidad de desarrolladores que posee, la apertura hacia la implementación de nuevas características y también debido a que pueden existir conflicto de intereses en Oracle por MySQL que puede determinar en un abandono del proyecto.

Por lo tanto, la herramienta de administración a utilizar es PgAdmin la cual está optimizada para la utilización junto a PostgreSQL.

2.1.5.2. Aplicación de Escritorio

La aplicación de escritorio será desarrollada con lenguaje de programación JAVA y permitirá visualizar 2 ventanas, la principal presentará mediante 2 gráficos los valores de los parámetros obtenidos por los sensores conectados al Arduino, los gráficos se dibujarán periódicamente cuya recolección será accionada por un timer, esta sección de la aplicación trabajará con el agente JMX y accederá a dichos valores mediante la manipulación de MBean. La ventana secundaria será utilizada para obtener valores históricos recogidos de los sensores y para la presentación de reportes.
El entorno de desarrollo a utilizar será NetBeans IDE 8.2 [41] pues este entorno ya posee librerías para la implementación de interfaces de usuario gráficas. Para presentación de las gráficas se utilizará la librería JFreeChart [42].

2.1.5.2.1. Diagrama de clases

La Figura 2.15 presenta el diagrama de clases de la aplicación de escritorio, las tablas de la base de datos fueron mapeadas utilizando la librería JPA por lo cual se visualizan como clases dentro del diagrama y son las que más relaciones presentan principalmente por las claves foráneas de las relaciones entre tablas.
El diagrama UML fue generado utilizando un plugin dentro del IDE Netbeans. Las tres clases principales de la aplicación son “MainProgram”, “Monitor” y “Reporte”; las relaciones existentes entre estas clases corresponden a la entrega de los valores de los objetos MBean. Los atributos de estas clases tienen declaración ‘private’ lo que indica qué si fuere necesario acceder a ellos, únicamente seria a través de los métodos de obtención y escritura o también conocidos como getters y setters.

La clase Reporte no tiene relaciones debido a que únicamente es utilizada para consultar valores históricos. La clase “Lectura” tiene relación con la clase “MainProgram” y también con la clase “Monitor”, pues como objeto puede albergar valores de los sensores y su identificación, esto fue utilizado para paso de datos como argumento entre constructores de clases.

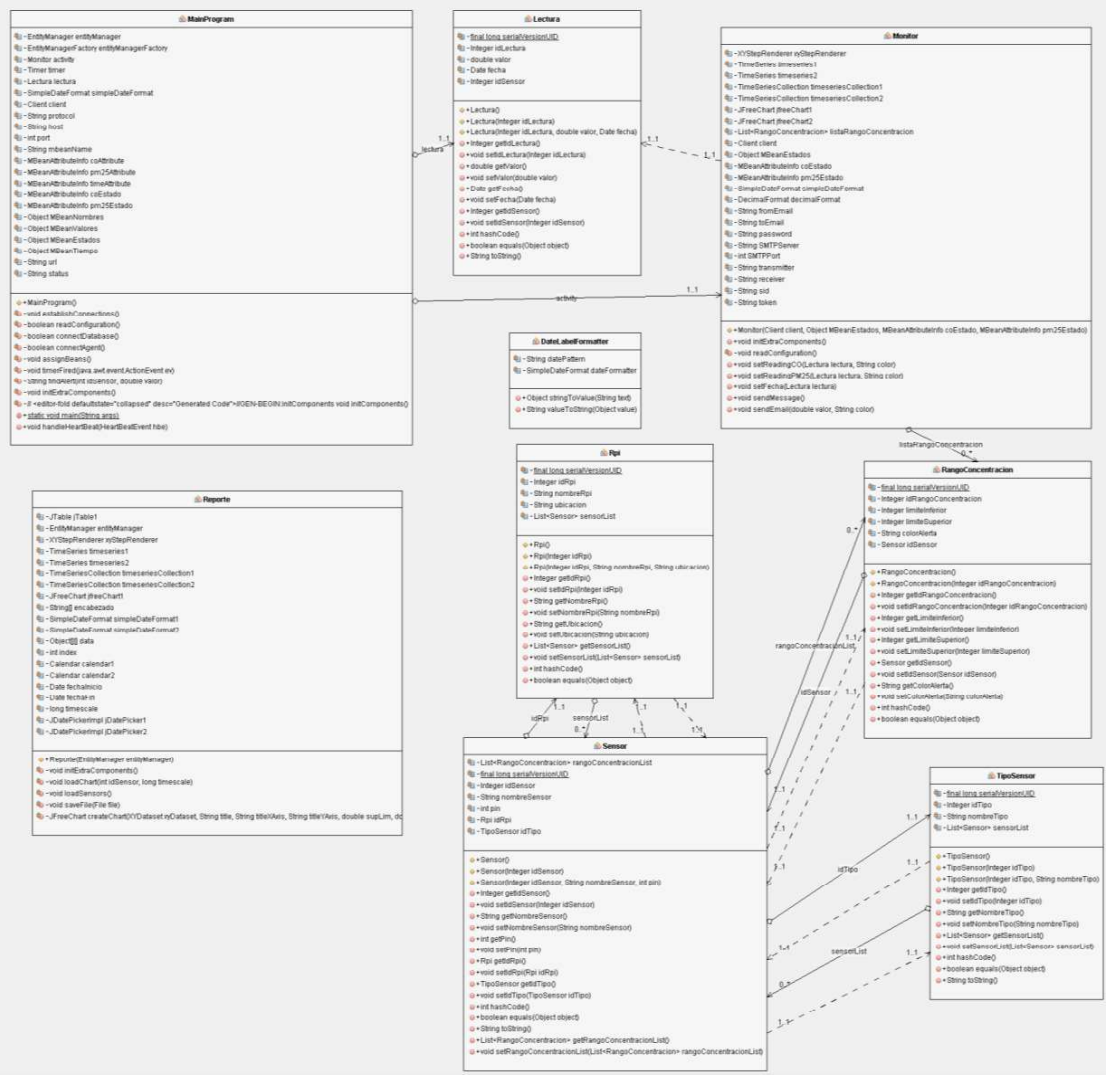


Figura 2.15. Diagrama de Clases de aplicación de escritorio

2.1.5.2.2. Interfaz gráfica de la aplicación de escritorio

Mediante una interfaz gráfica se mostrarán dos ventanas, en la principal que se puede observar su bosquejo en la Figura 2.16 se mostrarán dos gráficos en los que se dibujen periódicamente los valores de los datos recogidos por los sensores, estos valores también son guardados en la base de datos mencionada en el literal anterior.

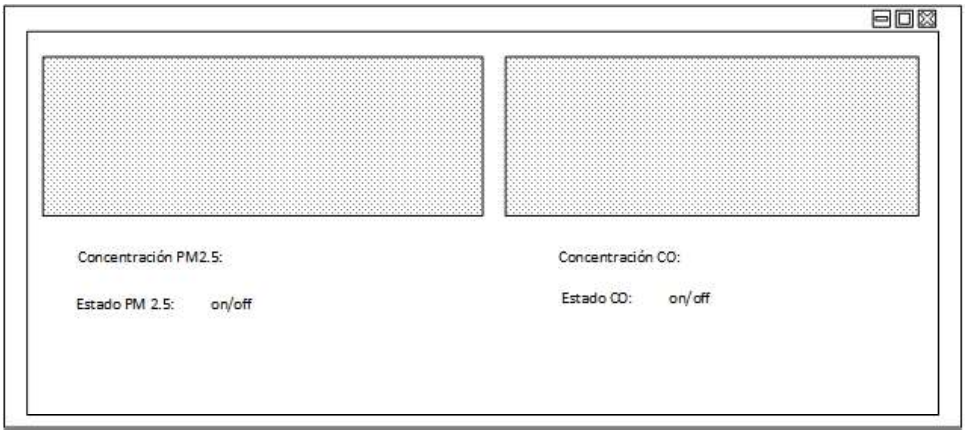


Figura 2.16. Ventana de gráficos de aplicación

La ventana secundaria presenta reportes de datos históricos consultados mediante la elección de una determinada fecha, esta consulta se realizará al RPi que contiene la base de datos. La dificultad de manejo de esta interfaz será poca o nula, tratando de ser intuitiva pues quien manipulará la interfaz no forma parte de un área técnica.

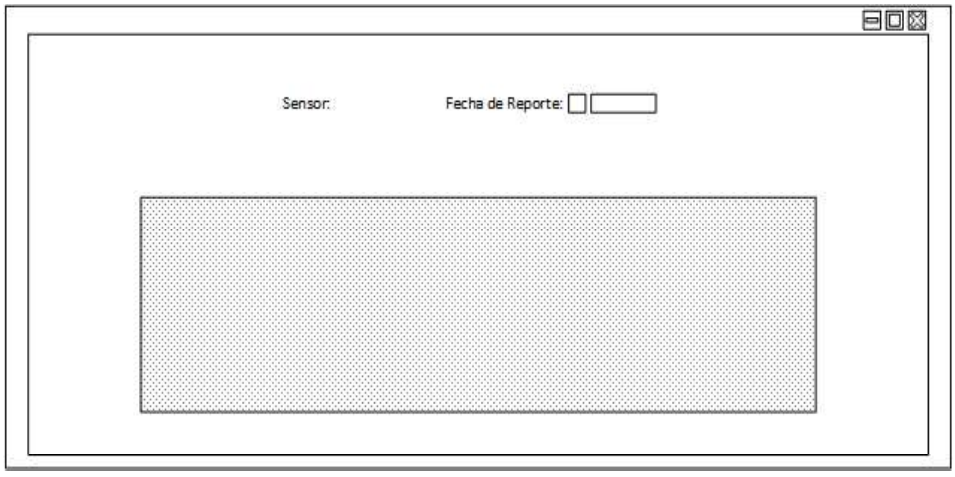


Figura 2.17. Ventana de reportes de aplicación

2.1.6. MÓDULOS INTEGRADOS

Una vez concluido el diseño de los 3 módulos se debe acoplar cada uno donde corresponde y verificar si cumple con el diseño original, para un mejor entendimiento se procede a presentar mediante un diagrama UML el despliegue del sistema completo.

2.1.6.1. Diagrama de despliegue del sistema

En el diagrama de despliegue mostrado en la Figura 2.19 se consideran 4 nodos principales, el Arduino que tiene los sensores conectados en él, mismos que también son considerados como nodos.

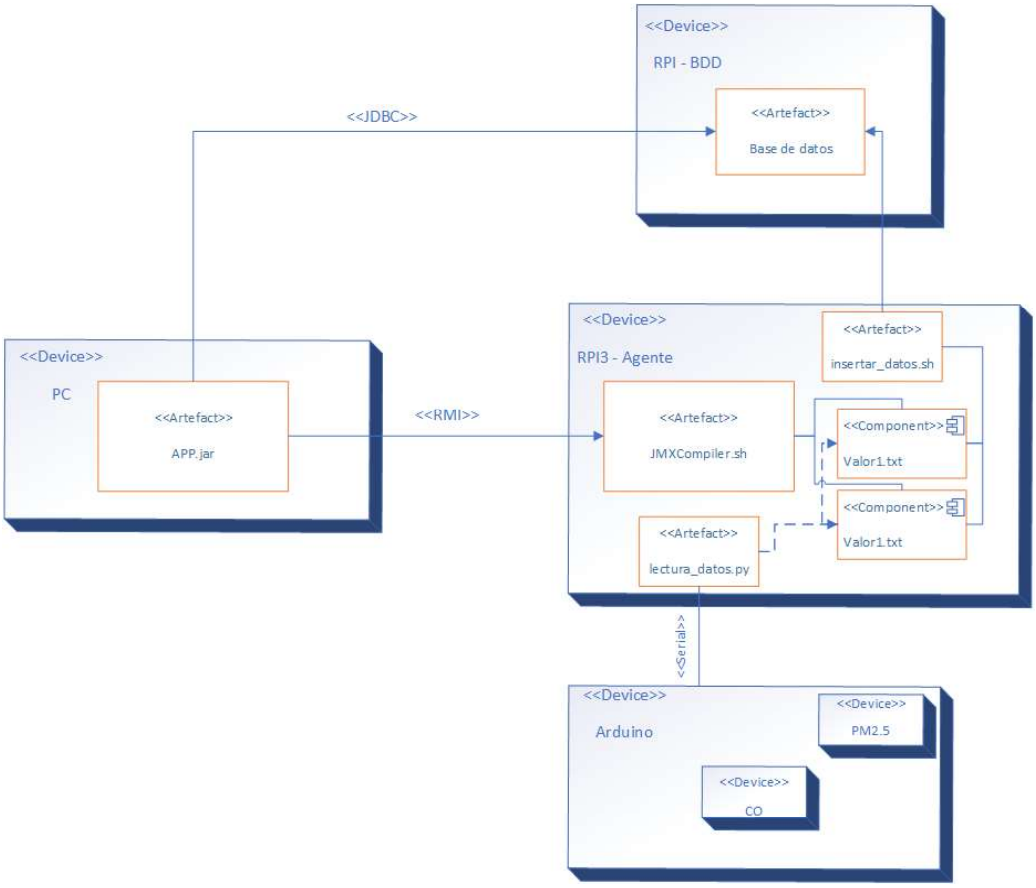


Figura 2.18. Diagrama UML de despliegue del sistema

El RPi – Agente es el nodo que más elementos contiene en su interior, pues este realiza la mayor cantidad de procesos como la adquisición de los valores orquestado por el artefacto readNsave2File.py, de este nacen los componentes valorCO.txt y valorPM25.txt que a su vez son utilizados por el artefacto JMXCompiler.sh que supone el inicio del agente JMX y los utiliza como fuente de información, los componentes también son utilizados por el artefacto insert_in_tables.sh para también obtener los valores e insertar los datos en la base de datos localizada en el nodo RPi-BDD.

El nodo RPi-BDD contiene como su artefacto principal la base de datos, que es llenada a través del trabajo realizado por el antes mencionado artefacto insert_in_tables.sh.

Finalmente, el nodo restante corresponde a la PC que alberga al artefacto app.jar que corresponde a la aplicación de escritorio, esta se comunica con el artefacto JMXCompiler.sh a través de la ruta RMI para obtener los objetos deseados y presentarlos en la ventana monitor de la aplicación y también tiene la ruta de conexión a la base de datos para obtener la información de las consultas a presentar en la ventana de reportes de la aplicación.

2.2. IMPLEMENTACIÓN

A continuación, en la siguiente sección se iniciará con la actualización del tablero Kanban con las tareas a realizar en este subcapítulo para después seguir con la implementación de todos los componentes de cada módulo.

2.2.1. TABLERO KANBAN

Una vez concluidas todas las fases del diseño del prototipo se continúa con la actualización del tablero Kanban, la mayoría de los ítems o actividades de la previa fase están localizadas en la sección "*Done*" mientras que pocas se encuentran aún en WIP, pero en la sección de "*Details*" lo que indica que aguardan por detalles para su completa terminación por lo tanto no afecta en el desarrollo de las actividades a desarrollar en esta fase. En esta fase se toman las actividades del backlog correspondientes. La Figura 2.19 presenta el tablero Kanban para la presente fase.

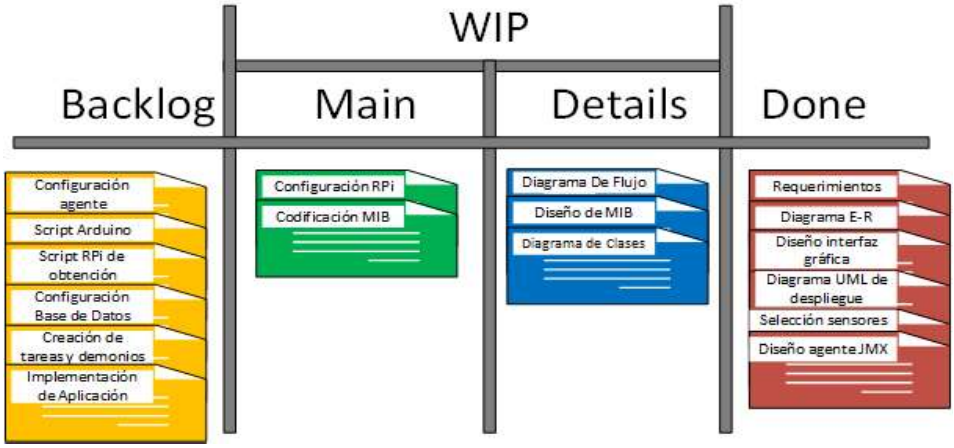


Figura 2.19. Tablero Kanban para la fase de implementación

2.2.2. IMPLEMENTACIÓN DEL MÓDULO DE ADQUISICIÓN

2.2.2.1. Configuración del Raspberry Pi

Para empezar en el desarrollo del módulo de adquisición se debe realizar la configuración del RPi.

2.2.2.1.1. Instalación del Sistema Operativo

El sistema operativo puede ser instalado de 2 maneras, la primera es utilizando el kit para instalación rápida llamada NOOBS el cual guía a través de todas las etapas necesarias y otras opcionales para la configuración del RPi, mientras que la otra opción es un proceso normal de instalación de un sistema operativo para una PC, el cual es descargando la imagen y quemándola en un disco o tarjeta. Se utiliza la segunda opción.

La imagen del sistema operativo más actual se encuentra en la página oficial del RPi, ésta debe ser descargada. Utilizando la herramienta Etcher [43] se elige el archivo de la imagen y se selecciona la tarjeta de almacenamiento como destino de grabación. Una vez listo se procede a realizar la instalación del sistema operativo. La versión por utilizar se denomina Raspbian Stretch. En el gráfico 2.20 se presenta una imagen del proceso de grabación del sistema operativo en la tarjeta de almacenamiento.

Figura 2.20. Grabado de SO

Una vez iniciado el sistema operativo, se realizan las configuraciones restantes, se debe realizar una actualización del SO, esto lo realizamos ejecutando el comando:

```
pi@raspberrypi: ~ $ sudo apt-get update
```

2.2.2.1.2. Configuraciones necesarias

Se han requerido configuraciones adicionales del entorno de trabajo en el RPi, como es el caso de la habilitación del protocolo SSH para el acceso y control remoto por medio del terminal de comandos y la instalación de una herramienta NetSarang [44] para el manejo del escritorio remoto. Para el primer caso, SSH puede ser habilitado a través de la herramienta de configuración ingresando el comando*:*

```
pi@raspberrypi: ~ $ sudo raspi-config
```

La Figura 2.21 presenta la interfaz desplegada al ingresar dicho comando.

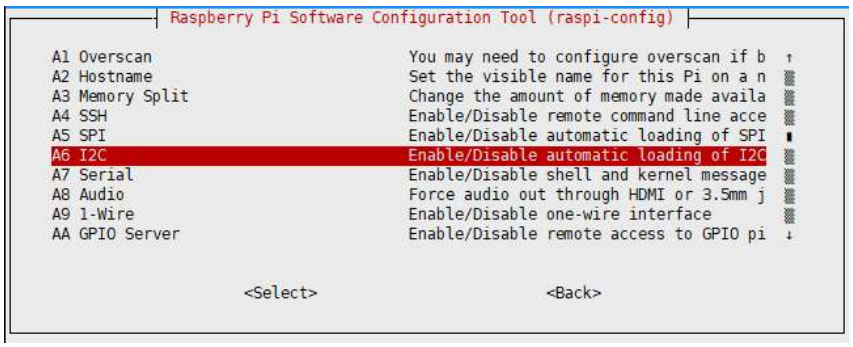


Figura 2.21. interfaz herramienta raspi-config

En esta interfaz pueden habilitarse más de una característica propia del RPi que por default vienen deshabilitadas.

2.2.2.2. Configuración de Arduino

El arduino en el presente proyecto es utilizado para la conversión de señales analógicas a señales digitales, a él se conectan los sensores de CO y PM2.5, se debe programar y cargar un script que permita constantemente leer los valores obtenidos por dichos sensores, guardarlos en una trama definida y enviarlos a través del puerto serial.

2.2.2.2.1. Conexión del sensor MQ-7.

Como se mencionó en la sección del diseño, el módulo MQ-7 necesita ser calibrado y configurado de una manera diferente a la mayoría de los sensores de esta familia. Se requiere que sea alimentado en un periodo de 150 segundos y a voltajes diferentes durante 48 horas como parte del periodo de calentamiento. Para lograr el voltaje de 1.4 [V] de la alimentación se utiliza un regulador de voltaje LM317T, el cual puede entregar voltajes variables según la elección de resistencias conectadas a su pin de ajuste.

Para el cálculo de las resistencias necesarias para alcanzar el voltaje deseado de 1.4 [V] se utilizó la fórmula descrita en la hoja de datos del regulador de voltaje.

$$V_o = 1.25\ x\left(1 + \frac{R_H}{R_L}\right)$$

Donde R_H corresponde al valor de la resistencia entre el pin de voltaje de salida y el pin de ajuste, y R_L corresponde al valor de la resistencia entre el pin de ajuste y tierra.
Considerando que se desea obtener un valor de voltaje de salida de 1.4 [V], se asigna un valor fijo para la resistencia $R_H = 1k\Omega$, y se obtiene un valor de $R_L = 120\Omega$**.**
Mientras que para lograr el alternado de la alimentación se utiliza un módulo relé el cual mediante una señal digital enviada desde un pin digital del Arduino permite accionar el cambio en él. A continuación, en la Figura 2.22 se muestra el diagrama de conexión empleado.

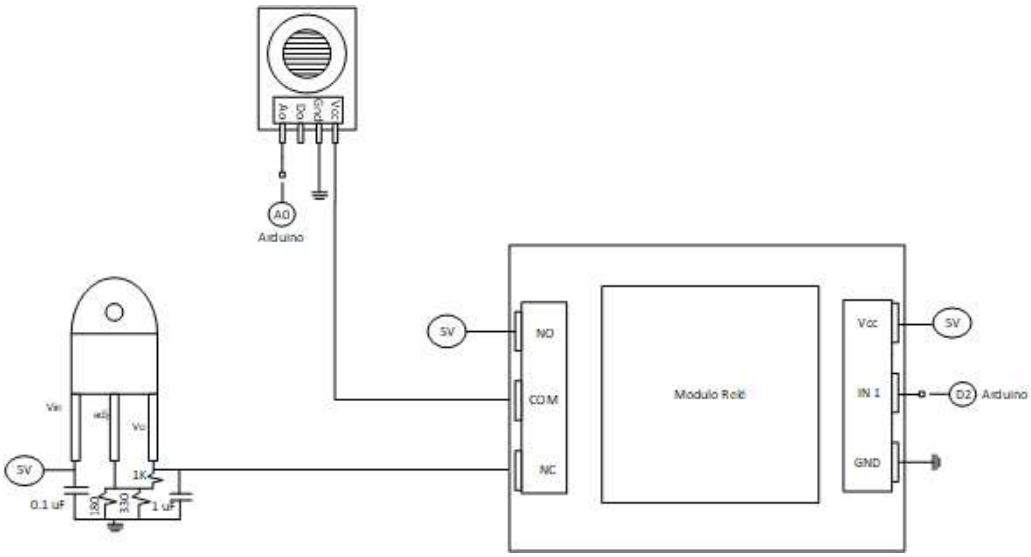


Figura 2.22. Diagrama de conexión sensor MQ-7

La implementación de este diagrama puede visualizarse en la Figura 2.23. El alternado únicamente se efectúa dentro del período de calentamiento, y cuando se encuentra en el período de lectura únicamente se utiliza la alimentación a 5V.

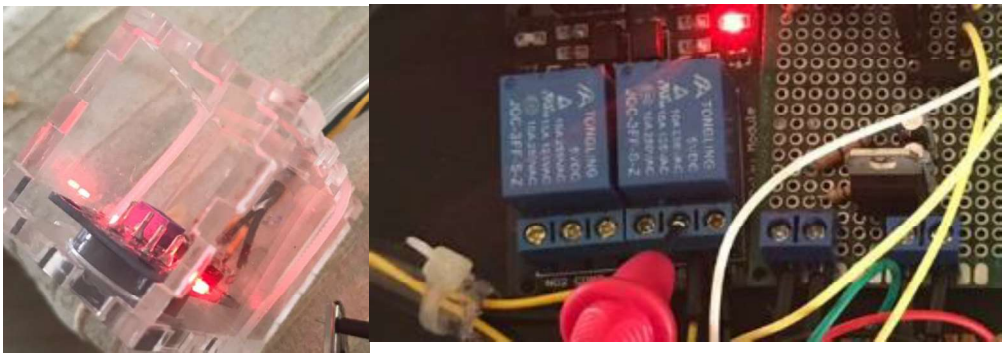


Figura 2.23. Implementación del circuito de conexión del sensor MQ-7

2.2.2.2.2. Calibración del sensor MQ-7

Una vez lista la conexión se procede a realizar el proceso de calibración del sensor. Cabe indicar que la aplicación de la fórmula de calibración será en el script en Python a ejecutarse en el RPi, esto fue resuelto debido a la intención de mantener datos simples al momento de elaborar la trama de entrega de datos desde el Arduino hacia el RPi.

La calibración es necesaria para asegurar que los valores entregados sean lo más reales posibles. Dado que la lectura que se realiza del sensor es conectada a un pin analógico es necesario realizar la conversión del valor de voltaje al valor de la magnitud medida. La resolución del conversor analógico - digital propio del Arduino Uno es de 10 bits, esto nos indica que el valor máximo de lectura será $2^{10} - 1$ valores o 1023 y esto equivale al voltaje máximo de 5 [V].

El método para realizar la calibración es por comparación con un equipo dedicado. El equipo por comparar es el medidor de monóxido de carbono Kidde KN-COPP [45]. Puede ser visualizado en la Figura 2.24.

Figura 2.24. Monitor de CO Kidde *[45]*

Este equipo tiene un rango de detección de 30 a 1000 ppm. El proceso por seguir es someter a ambos dispositivos: tanto el equipo dedicado como al sensor a ciertas concentraciones de contaminación de CO y realizar apuntes sobre la señal leída desde el Arduino. Para obtener concentraciones elevadas del gas se utilizó los gases emitidos desde un auto, los cuales fueron insertados en un contenedor con poco o nada de aire en su interior, para que de esta manera la concentración de dicho gas sea elevado al interior. Ambos equipos ocuparon posiciones semejantes en tanques sellados para la obtención de los valores. En la Figura 2.25 puede observarse lo indicado.

Figura 2.25. Proceso de calibración MQ-7

A continuación, en la Tabla 2.5 se presentan algunos de los datos recogidos y su equivalente leído desde el sensor MQ-7.

Tabla 2.5. Valores de muestra en comparación MQ-7

Equipo Kidde (ppm)	MQ-7
35	79
81	90
125	94
154	116
221	122
299	133

En la línea de tendencia presente en la comparación de valores se evidenció dos comportamientos distintos a partir de un punto de intersección entre los valores reales y los medidos por el sensor, por lo cual se emplea una fórmula de calibración por cada rango separado por dicho punto de intersección. La Tabla 2.6 muestra las fórmulas de calibración para el sensor MQ-7.

Tabla 2.6. Calibración MQ-7

Rangos (cuentas)	Fórmula
0 - 102	ppm = 5.164 X – 377.11
103 +	ppm = 4.0827 X – 290.74

Siendo X el valor leído desde el MQ-7.

2.2.2.2.3. Conexión del sensor de $PM_{2.5}$.

El esquema de conexión por utilizar con este sensor es el mostrado en la Figura 2.26. Este esquema ha sido reproducido según la recomendación de la hoja de datos del sensor.

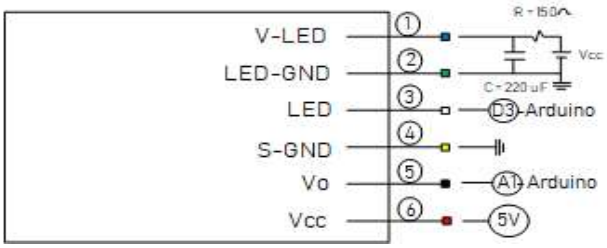


Figura 2.26. Esquema de conexión sensor Sharp

Este sensor depende de un disparo de acción ejecutado desde un pin digital del Arduino que debe conectarse al pin 3 del sensor. Una vez realizado esto se procede a leer desde el pin analógico del Arduino que debe conectarse al pin 5 del sensor. La conexión del módulo puede observarse en la Figura 2.27.

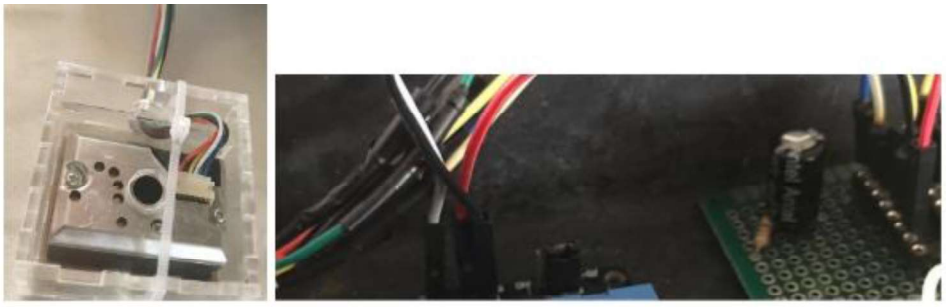


Figura 2.27. Implementación del circuito del sensor PM2,5

El pin digital D3 del Arduino es el encargado de realizar el disparo y este es conectado al pin 3 del módulo.

2.2.2.2.4. Calibración del sensor de $PM_{2.5}$.

Cabe indicar que la aplicación de la fórmula de calibración será en el script en Python a ejecutarse en el RPi, esto fue resuelto debido a la intención de mantener datos simples al momento de elaborar la trama de entrega de datos desde el Arduino. La lectura que se realiza del sensor es conectada a un pin analógico por lo que también es necesario realizar la conversión del valor de voltaje al valor de la magnitud medida, el valor máximo de lectura es 2^{10}-1 valores o 1023.

El método escogido para realizar la calibración es por comparación con un equipo dedicado. El equipo por comparar es el medidor de material particulado Temtop P600 el cual puede ser observado en la Figura 2.28.

Figura 2.28. Monitor de CO Temtop P600 *[46]*

Este equipo permite medir la concentración de material particulado fino y compuesto. El proceso por seguir es someter a ambos dispositivos: tanto el equipo dedicado como al sensor a ciertas concentraciones de contaminación de $PM_{2.5}$ y realizar apuntes sobre la señal leída desde el Arduino. En la Figura 2.29 puede observarse lo indicado.

Figura 2.29. Calibración sensor PM2.5

La comparación se realiza para afinar los valores obtenidos a través del proceso de calibración propio del sensor Sharp [47]. A continuación, en la Tabla 2.6 se presenta una muestra de los valores recogidos por ambos dispositivos.

Tabla 2.7. Valores comparación sensor Sharp

Equipo Temtop ($^{ug}/_{m^3}$)	Sensor Sharp GP
637.2	281
651.6	162
1000	438
730.3	322
256	100
385.4	60

Este sensor presentó medidas inestables pues cuando los valores sobrepasaban ciertos umbrales entonces se marcaba una tendencia similar en las medidas, pero cuando los valores se localizaban debajo de esas medidas entonces el sensor y el dispositivo de medición marcaban casi lo mismo. Por lo cual también se definió la utilización de 3 rangos con aplicación de fórmulas para cada uno. De esta manera se controló mejor el comportamiento de los valores y se evitó que al marcar una sola ecuación de tendencia se exageren medidas.

Tabla 2.8. Calibración PM2.5

Rango	Fórmula
0 - 35	$^{ug}/_{m^3} = 0.1377X + 17.984$
36 – 100	$^{ug}/_{m^3} = -1.4513X + 406.21$
101 +	$^{ug}/_{m^3} = 0.3457X + 571.9$

Siendo Y el valor de $^{ug}/_{m^3}$, mientras que X corresponde al valor leído desde el Sharp.

2.2.2.2.5. Codificación del script

En el Arduino se manejarán y enviarán únicamente los valores enteros leídos por los puertos analógicos, mientras que en los scripts de Python en el Raspberry Pi se realizarán los cálculos de calibración de los valores.

El Arduino obtiene los valores de los sensores y los debe enviar hacia el Raspberry Pi a través del puerto serial, por lo que se debe indicar en el lazo de configuración la velocidad de la comunicación a utilizar. El espacio de Código 2.1 muestra la definición de lo mencionado.

```
void setup()
  {
    Serial.begin(9600);
  }
```

Código 2.1. Comunicación serial Arduino

La comunicación serial tiene una característica negativa y es que sufre pérdidas, por lo que es necesario implementar un mecanismo para de asegurar que los datos recibidos sean los enviados. Para esto se definió la utilización de una trama que albergue los datos. La trama por utilizar puede observarse en la Figura 2.30

1 byte	2 bytes	2 bytes	2 bytes	1 byte
Bandera	DatoMQ	DatoPM	FCS	Bandera

Figura 2.30. Trama comunicación Arduino – RPi

En la trama, se define la utilización de banderas que definan el inicio y fin de una trama, a excepción de las banderas, todos los campos son de 2 bytes, para las banderas se utiliza el código ASCII que tiene por entero 126 o 01111110 en bits. Cada campo, al interior de las banderas el segundo y tercer campo corresponden a aquellos que contienen los

valores de los sensores, mientras que el penúltimo campo corresponde a un FCS que contiene la suma de todos los campos de valores de la trama, este es el mecanismo de protección de los datos.

Cada campo de datos es de 2 bytes, debido a que la resolución del conversor analógico-digital del Arduino es 10 bits, por lo que es necesario utilizar 2 bytes para poder representar cada lectura. En el espacio de Código 2.2 puede observarse la lógica del arreglo de datos a acomodar en la trama.

```
com = analogRead(MQ_PIN);
arreglo[1] = com&255;
arreglo[2] = com>>8;
```

Código 2.2. Lógica de guardado de datos en trama

El método *analogRead(MQ_PIN)* realiza la lectura analógica del pin, esto se guarda en la variable *com*, se utiliza un arreglo para guardar los datos, por lo que los 2 primeros campos de este arreglo serán utilizados en este momento. Los primeros 8 bits deben ser alojados en el primer campo, por lo que se realiza un & lógico con 255 de la variable com, mientras que los restantes bits son desplazados 8 campos hacia la derecha para que ocupen 1 byte, esto se logra con el operador lógico >>, este proceso se realiza para cada valor leído, así como para el FCS. El Código 2.3 muestra la generación del FCS una vez recogidos todos los valores de los sensores.

```
for (i=1; i<5; i++)
  tok = tok + arreglo[i];

arreglo[5] = tok&255;
arreglo[6] = tok>>8;
```

Código 2.3. Cálculo FCS de trama

Para enviar el arreglo por el puerto serial, se debe utilizar el método *Serial.write(),* este permite enviar los datos como bits, el Código 2.4 muestra la declaración utilizada.

```
Serial.write(arreglo,sizeof(arreglo));
```

Código 2.4. Código envío de datos por puerto serial

Para el código correspondiente al manejo del sensor MQ-7 se tiene dos estados, el primero es el estado de lectura en el cual se realiza lecturas del puerto analógico a una alimentación constante de 5 [V], y el otro estado es el correspondiente al período de

calentamiento para lo cual se consideró el patrón definido en la hoja de datos sobre la variación en la alimentación y los tiempos. El patrón puede observarse en la Figura 2.31.

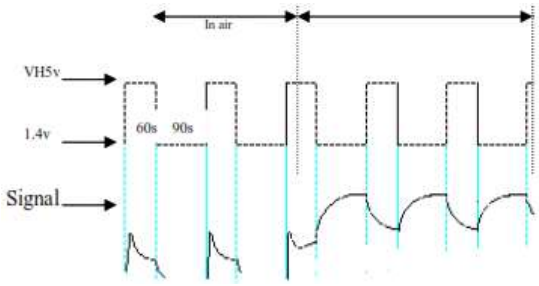


Figura 2.31. Utilización MQ-7 *[36]*

Un conmutador o switch será el responsable de definir el estado en el que se encuentre el sensor, para obtener el valor se realizará una lectura en un pin digital y dependiendo del valor se ejecutará cada acción. El código correspondiente queda de la siguiente manera:

```
com = digitalRead(LECT);
if(com == HIGH)
{

  //Serial.println("fue high");
  digitalWrite(IN1, LOW);
  PM25(6017);
  digitalWrite(IN1, HIGH);
  PM25(9025);

}
else
{
  //Serial.println("fue low");
  digitalWrite(IN1, LOW);
  PM25(1001);
  makenSend();
}
```

Código 2.5. Método para manejo del sensor MQ-7

Sección $PM_{2.5}$

Para el código correspondiente al manejo del sensor $PM_{2.5}$ se consideró que previo a cada lectura se debe seguir el patrón definido en la hoja de datos. Dicho patrón puede observarse en la Figura 2.32.

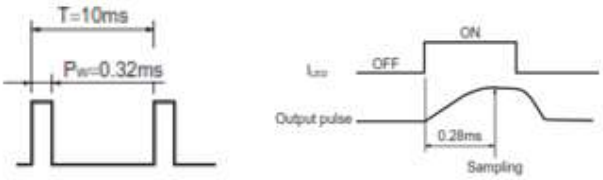


Figura 2.32. Período de funcionamiento sensor MQ-7 *[36]*

Es decir, mientras se está enviando el pulso positivo se debe esperar el tiempo indicado y realizar la lectura. Considerando estas recomendaciones, el código queda de la siguiente manera:

```
void PM25(int lazo){

  for(int i=1; i<lazo; i++)
  {
    digitalWrite(LED_PM, LOW);
    delayMicroseconds(160);//280
    voPM = analogRead(voPM_PIN);
    delayMicroseconds(40);

    digitalWrite(LED_PM, HIGH);
    delayMicroseconds(9596); //997360 a
  }

}
```

Código 2.6. Método para manejo de sensor de PM2,5

Se definió el mecanismo del algoritmo para que este dependa su número de iteraciones de una variable y de esta manera este pueda ser utilizado mientras se está realizando el calentamiento del sensor MQ-7.

2.2.2.3. Obtención de datos desde Raspberry Pi

Concluida la etapa de envío de datos desde el Arduino, se prosigue con la obtención de dichos datos en el RPi, esto se logra mediante la utilización de un script que constantemente lea el puerto serial a la espera de las tramas que contienen la información deseada. El primer paso es la preparación del ambiente de desarrollo de dicho script.

2.2.2.3.1. Codificación del script

Durante la codificación de este script se utilizarán librerías en aspectos importantes del funcionamiento, como son el leer el puerto de comunicación y el manejo de dichos datos únicamente cuando son recibidos. Para dicho propósito son utilizados las librerías mostradas en el espacio de Código 2.7.

```python
import serial
import threading

#ser = serial.Serial('COM3')
#doc_list = ['valorSensor1.txt','valorSensor2.txt']
try:
    ser = serial.Serial('/dev/ttyACM0')
except serial.SerialException as e:
    print("arduino desconcetado")
```

Código 2.7. Librerías utilizadas script python

El método serial.Serial(), permite definir el puerto de comunicación que se utilizará. Para debian el puerto de comunicación por defecto es aquel con ruta en `/dev/ttyACM0`.

La sincronización a la hora de ejecutar tareas considerando la interconexión de plataformas puede ser problemático si se considera que una de esas plataformas puede presentar retrasos por uso de recursos de procesamiento, por tal motivo la utilización de la ejecución a través del uso de cron se acompañó con la constante lectura de dicho puerto serial y el uso de hilos. De esta manera se asegura la obtención de datos desde el puerto. A continuación, en el Código 2.8 se puede observar la lógica empleada para la constante lectura del puerto serial.

```python
#   METODO PARA IDENTIFICAR LA TRAMA EN BUFFER
while True:
    #obtengo un entero del byte recibido.
    x = int.from_bytes(ser.read(), byteorder = "little")
    print(x, end='')
    #compruebo si el valor es una bandera
    if x == 126 and flag == False:
        #al ser una bandera entonces cambio el valor de la variable para
        #despues verificar si es dato u otra bandera
        flag = True
    elif flag == True:
            #verifico que  hayan llegado hasta los bytes del crc
        if x == 126 and len(trama) > 4:
            #verifico  si llegaron mas datos de lo debido por lo que reiniciaria el conteo.
            if len(trama) > 6:
                trama = list();
                print("trama erronea, esperando por otra")
                flag = True
            else:
                print(trama)
                flag = False
                #procedo a apuntar el hilo a una operación enviando como argumento
                #el arreglo de la trama.
                t = threading.Thread(target=save_to_file, args=(trama,))
                trama = list()
                #defino la trama como una lista.
                t.start()

        else:
            #no es bandera, entonces lo coloco en un arreglo.
            #en arduino fije si un dato es leido en 126 entonces cargar como 127, falta revisar cuanto afec
            trama.append(x)
```

Código 2.8. Lógica lectura de puerto serial

La operación llamada contiene todos los pasos por seguir para guardar los datos requeridos en los archivos de texto, al terminar esta operación se configura a 0 la variable trama y se cambia el valor de la variable flag. Esto es útil para que el proceso anterior definido en el Código 2.8 pueda desarrollarse con normalidad. Esto puede verificarse en el Código 2.9.

```
def save_to_file(argu):
    print("stf")
    t = checkTrama(argu)
    if(t==1):
        arregloInt = hexToInt(argu)
        param = getReal(arregloInt)
        #param = [0,0]
        #saveInFiles(doc_list,param)
        #print("saved to file", param)
    else:
        param = [0,0]
        print("bad values, saved 0")

    saveInFiles(doc_list, param)
    print("saved to file", param)
```

Código 2.9. Función principal

El entregable del Código 2.8 fue la composición de la trama llegada al puerto, sin embargo, aún no ha sido verificado si los datos recibidos fueron los enviados, para dicho propósito se procede a verificar si el FCS de la trama es correcto, en el Código 2.10 se puede observar el algoritmo para la verificación.

```
def checkTrama(tr):
    print(tr)
    global fcs
    global suma
    global v
#se obtiene el valor del CRC a traves de la suma del ultimo y penultimo
    #valor de la trama, en este punto las banderas ya no forman parte de la
    #trama.
    fcs = (tr.pop() << 8) + tr.pop()

    suma = 0

    for x in range(0,len(tr)-1):
        if x%2 == 0:
            #defino el punto de operacion como la primera posiciom, recordando
            #que dos bytes conforman una lectura de cada sensor.
            suma = suma + tr[x] + (tr[x+1] << 8)
    if suma == fcs:
        #ambos valores han coincidido por lo que procedo a dar el visto bueno.
        print("\ntrama valida suma={} y fcs={}".format(suma,fcs))
        v = 1
        return v

    else:
        print("\nerror suma={} y fcs={}".format(suma,fcs))
        v = 0
        return v
```

Código 2.10. Código de verificación de FCS

El FCS actúa únicamente sobre los datos enviados, en su formato de envío, una vez verificado que ha llegado en la trama aquello que originalmente ha sido enviado, se procede a obtener los valores reales enviados, esto se logra sumando los dos bytes separados de cada sensor en un solo valor, y acomodados en una lista vacía. Recordando que se utilizó Little Endian para el orden de la trama, puede observarse dicha lógica en el Código 2.11.

```
def hexToInt(arreglo):
    print("hextoint")
    print(arreglo[0])
    #defino una lista nueva.
    arregloF = list()
    #recorro en un lazo los items del arreglo original
    for x in range(0,len(arreglo)-1):
        if x%2 == 0:
            #procedo a añadir los valores totales a la nueva lista.
            arregloF.append(arreglo[x] + (arreglo[x+1] << 8))
            #y devuelvo dicha nueva lista
    print(arregloF)
    return arregloF
```

Código 2.11. Conversión de 2 bytes a 1

En este momento se tiene la lista de valores arrojados por la lectura de los pines analógicos del Arduino, entonces es hora de aplicar la fórmula de calibración en cada uno de los sensores. En el Código 2.12 se puede observar ambas fórmulas, y también se puede observar que la función retorna dos valores, cada uno representa cada magnitud.

```
# METODO QUE EMPLEA FORMULAS DE CALIBRACION
# ENTREGA ARREGLO CON DOS VALORES, UNO DE CADA SENSOR
def getReal(arreglo):
    print("getReal")
    #CO = ((arreglo[0]*4.0827)-290.74)
    #PM = ((arreglo[1]*1.7319)+68.649)
    if(arreglo[0] == 0 or arreglo[0] <=75):
        CO = 0.1
    if(arreglo[0] > 75 and arreglo[0] <= 102):
        CO = ((5.134*arreglo[0])-377.11)
    if(arreglo[0]>102):
        CO = ((4.0827*arreglo[0])-290.74)

    if(arreglo[1] == 0):
        PM = 0.1
    if(arreglo[1] > 0 and arreglo[1] <= 35):
        PM = ((0.1377*arreglo[1])+17.984)
    if(arreglo[1]>35 and arreglo[1] <= 100):
        PM = ((-1.4513*arreglo[1])+406.21)
    if(arreglo[1]>100):
        PM = ((0.3457*arreglo[1])+571.9)

    return CO,PM
```

Código 2.12. Aplicación de fórmula de calibración de MQ-7

Finalmente, al obtener los valores reales se procede a guardar dichos valores en los archivos de texto que serán utilizados como fuente de información para el siguiente módulo

del proyecto. Estos archivos cada uno tiene una ruta específica que han sido declaradas previamente en un arreglo al inicio del script. En el Código 2.13 puede observarse la función utilizada para guardar los valores obtenidos en los archivos respectivos.

```
def saveInFiles(files_names,values_list):
    """este metodo me permite iterar la lista de archivos y escribir iterando
    los valores de sensores."""
    print("save")
    i=0
    for doc in files_names:
        data = open(doc, "w")
        x = str(values_list[i])  #guardo como texto
        data.write('%s' % x)
        data.close()
        i+=1
```

Código 2.13. Función de guardado de datos en archivos.

2.2.2.3.2. Tarea en crontab

La ejecución de cada 10 segundos para la inserción de datos en la base de datos se realizó añadiendo la ejecución del script de inserción en el crontab a ejecutarse cada minuto, y añadiendo una condición dentro del script para que cada 10 segundos realice la inserción. El Código 2.14 muestra el comando escrito en el archivo crontab.

```
* * * * *   root    sh /home/pi/Downloads/insert_in_tables.sh
```

Código 2.14. Comando para crontab de inserción de datos en tablas

2.2.3. IMPLEMENTACIÓN DEL MÓDULO AGENTE

2.2.3.1. Codificación del archivo MIB

El kit de herramientas entregado por WebNMS incluye varias aplicaciones necesarias para desarrollar un agente y probar su funcionamiento, el agente JMX necesita de 2 elementos para su generación, primero un archivo MIB y segundo una fuente de donde obtener la información. En este apartado se presenta una vía de codificación de la MIB, se dará uso de la aplicación MibEditor perteneciente a WebNMS, a continuación, se presenta los pasos a seguir para la elaboración del archivo MIB.

Para lanzar la aplicación se procede a ejecutar el siguiente comando en el terminal ubicándose en la carpeta bin.

```
root@raspberrypi:/home/pi/Downloads/WebNMS/JavaAgent/bin# ./MibEditor.sh
```

Código 2.15. Comando para lanzar aplicación de creación de MIB

Se debe seleccionar “Crear Mib”, en el menú inicio, inmediato a eso se crea el árbol MIB y el primer nodo a modificar es aquel marcado bajo el título “MODULE-IDENTITY”, en este punto se debe llenar en este caso con la información de la empresa. La Figura 2.33 presenta el formulario lleno con los datos correspondientes.

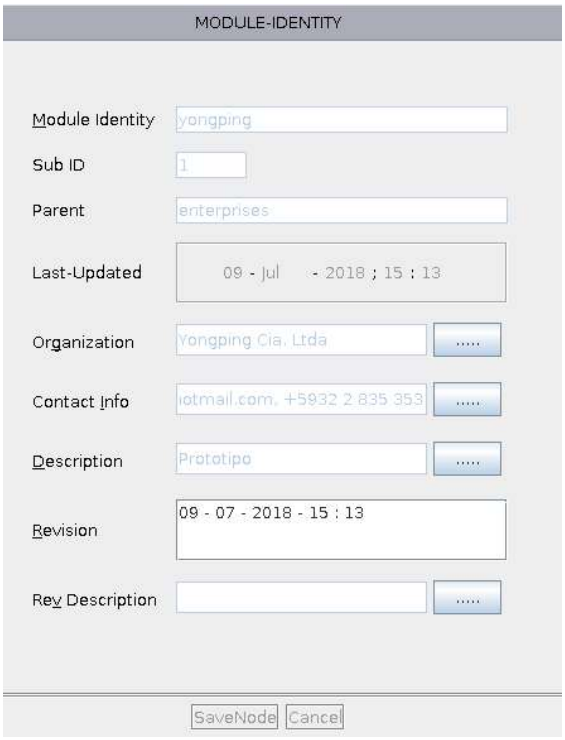


Figura 2.33. Llenado de etiquete MODULE-IDENTITY

Según el diseño de la MIB presentada en el capítulo anterior, la MIB se situaría por debajo del nodo enterprises, esta MIB tendría un nodo hijo denominado moduloProduccion, para la elaboración de este nodo únicamente se debe añadir un item OID en la Figura 2.34 se presenta esto.

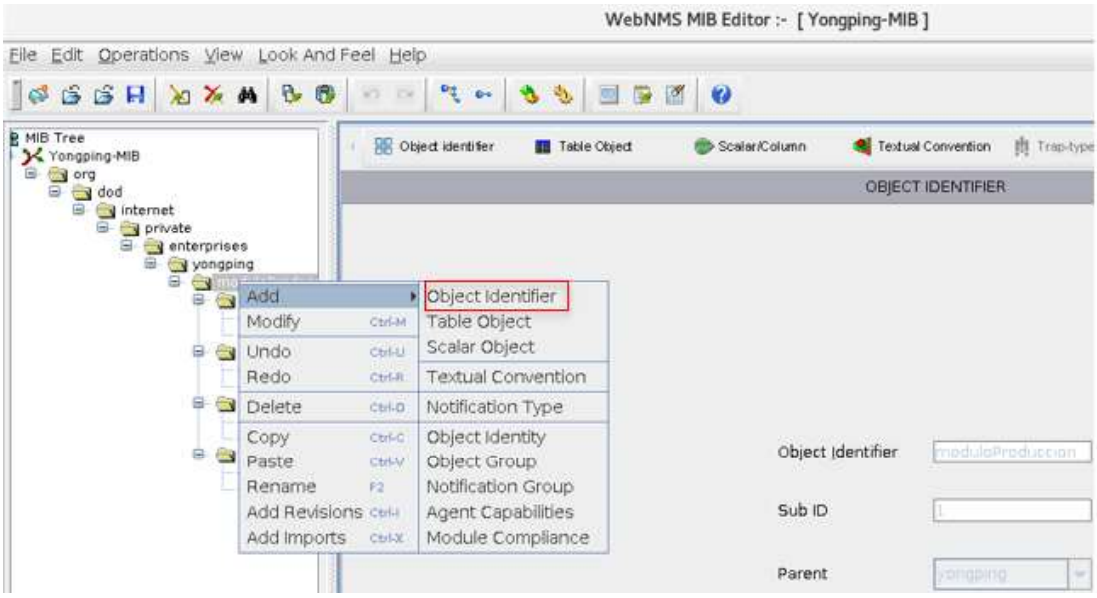


Figura 2.34. Inserción nodo oid

Al interior del nodo "moduloProduccion" se encontrarán 4 nodos hijos que albergarán objetos que entregan información similar de cada sensor, por ejemplo, se encontrará el nodo nombres dentro del cual se encontrarán objetos que entregan justamente los nombres específicos de cada sensor. Para la especificación de un objeto se debe seleccionar el tipo en primera instancia, por ejemplo, para la definición del objeto correspondiente al sensor de CO se deben llenar al menos los siguientes campos: en Object Type se coloca el nombre del objeto, se llamará sensCO, en Sub ID se define el índice numérico del nodo, en Syntax se define la sintaxis del objeto, este será OCTET STRING, en Max Access se define el permiso máximo que soportará, este tendrá permiso de lectura y escritura, en Status se presentará la versión máxima actual, por ejemplo puede ser "`obsolet`" o "`current`", en este caso se colocará Current. En Reference se establece si tiene alguna dependencia con valores de otros objetos, en este caso no posee ninguna dependencia. En el apartado DefVal se puede establecer un valor por defecto, en este caso aquí se inicializa con el valor sensCO. En la Figura 2.35 se presenta la plantilla para la especificación del objeto descrito.

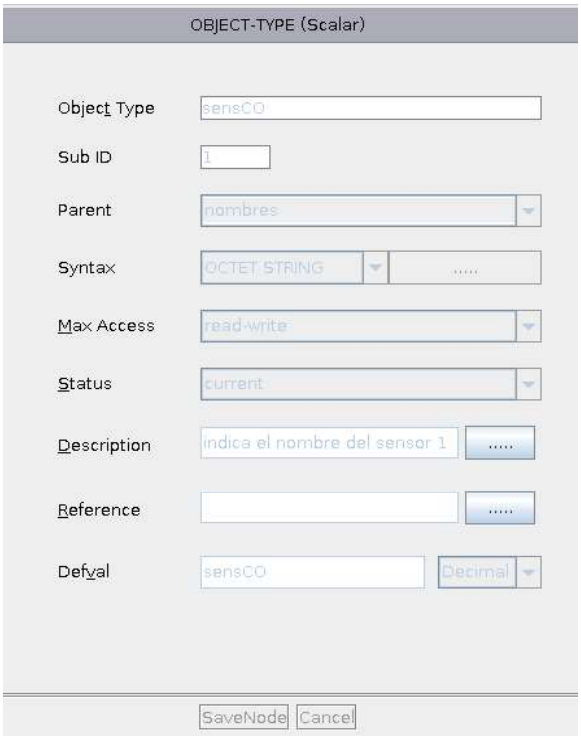


Figura 2.35. Plantilla Objeto sensCO

Este proceso se debe realizar para cada objeto del árbol, según el diseño de este, los objetos pertenecientes a los nodos “nombres” y “estado” tendrán permiso de lectura y escritura, mientras que los nodos de “valores” y “tiempo” tendrán permiso únicamente de lectura. Visualmente en el árbol situado en la sección de exploración la diferencia puede visualizarse a través de una X roja sobre la hoja que representa los objetos escalares. A continuación, la Figura 2.36 muestra el árbol con todos los objetos declarados en él.

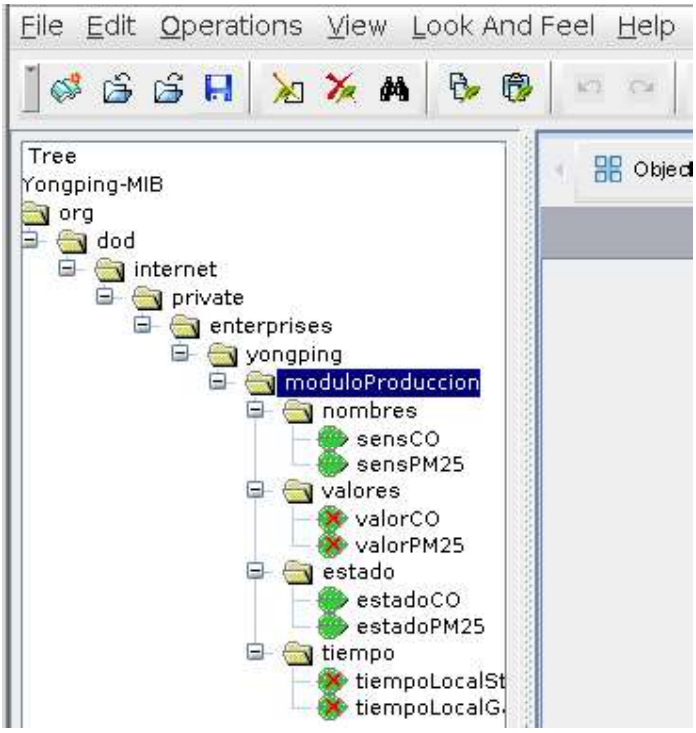


Figura 2.36. Árbol en MIB Editor.

Este editor entrega el archivo MIB el cual no posee extensión alguna. Este también puede codificarse en cualquier editor de texto, teniendo precaución en cumplir las reglas de sintaxis. La ubicación del archivo es: `WebNMS/JavaAgent/mibs/Yongping-MIB`.

La Figura 2.37 muestra las primeras líneas del archivo MIB, en estas se identifican la etiqueta de inicio “BEGIN”, así como la declaración de importación “IMPORTS” que define etiquetas especiales declaradas previamente en otros archivos MIB las cuales serán utilizadas en el archivo.

```
 GNU nano 2.7.4                                  Fichero: WebNMS/JavaAg

Yongping-MIB    DEFINITIONS ::= BEGIN
    IMPORTS
        enterprises, MODULE-IDENTITY, OBJECT-TYPE, Integer32, Gauge32
            FROM SNMPv2-SMI;
```

Figura 2.37. Encabezado Archivo MIB

En la declaración de identidad del módulo, se declaran datos informativos acerca de la organización autora del archivo, seguido de la referencia de ubicación dentro del árbol MIB, como fue mencionado en el pasado capítulo, el nodo principal se ubicará por debajo

del nodo .1.3.6.1.4.1 perteneciente a “enterprises”. A continuación, en la Figura 2.38 muestra la sección del archivo MIB discutida previamente.

```
yongping        MODULE-IDENTITY
        LAST-UPDATED    "201807091513Z"
        ORGANIZATION    "Yongping Cia. Ltda"
        CONTACT-INFO    "Juan de Leon Cordero N12-137 y Antonio Castelo$
        DESCRIPTION             "Prototipo"
        REVISION                "201807091513Z"
        DESCRIPTION             ""
        ::=  {  enterprises  1  }
```

Figura 2.38. Etiqueta MODULE-IDENTITY

Después de esto se debe continuar con la declaración de cada uno de los nodos definidos en la MIB, hasta llegar a la declaración de objetos, esto puede comprobarse en la Figura 2.39 la cual muestra como el editor genero correctamente todas las declaraciones de los nodos. En la declaración de un nodo, entre corchetes separados por espacios se debe definir el nombre del nodo padre seguido de un espacio y a continuación definir el índice del nodo especificado. Esta norma debe ser seguida para cada nodo hijo por ser definido en un archivo MIB, esto incluye a los objetos que entregan información.

```
org     OBJECT IDENTIFIER
        ::=  {  iso  3  }

dod     OBJECT IDENTIFIER
        ::=  {  org  6  }

internet        OBJECT IDENTIFIER
        ::=  {  dod  1  }

private OBJECT IDENTIFIER
        ::=  {  internet  4  }

enterprises     OBJECT IDENTIFIER
        ::=  {  private  1  }

moduloProduccion        OBJECT IDENTIFIER
        ::=  {  yongping  1  }

nombres OBJECT IDENTIFIER
        ::=  {  moduloProduccion  1  }

valores OBJECT IDENTIFIER
        ::=  {  moduloProduccion  2  }

estado  OBJECT IDENTIFIER
        ::=  {  moduloProduccion  3  }

tiempo  OBJECT IDENTIFIER
        ::=  {  moduloProduccion  4  }
```

Figura 2.39. Declaraciones de nodos

En los nodos correspondientes a objetos informativos se visualiza en una primera instancia la declaración del nombre del objeto, seguido por las 4 etiquetas: `syntax, max-access,`

`status y description`, seguido de la declaración de la ubicación en el árbol MIB. En la Figura 2.40 se puede observar lo mencionado.

```
sensPM25        OBJECT-TYPE
        SYNTAX                  OCTET STRING
        MAX-ACCESS              read-write
        STATUS                  current
        DESCRIPTION             "indica el nombre del sensor 2"
        ::=  {  nombres  2  }
```

Figura 2.40. Objeto sensPM25

La información de todo el archivo MIB puede encontrarse en el ANEXO II adjunto.

2.2.3.2. Configuración del agente

En esta sección se presenta los pasos a seguir para la implementación del agente JMX, iniciando por la instalación del kit, y el cumplimiento de los requisitos para la generación del agente. Se anticipa que se accedió al RPi de 2 maneras, la primera es la manipulación principal a través de un teclado y utilizando un monitor, y la segunda es dando uso del kit "XManager" a través de la herramienta XShell la cual permite conectarse a través de un escritorio remoto al RPi, por tal motivo cambian los colores en las imágenes.

2.2.3.2.1. Instalación del agente net-snmp

El agente JMX permite conexiones a través de SNMP y esto lo logra utilizando la característica de los agentes extendidos o AgentX. Esto indica que se extiende las funcionalidades del agente SNMP para la interacción con agentes propietarios que requieran su servicio, por lo tanto, es necesario la instalación y configuración del agente SNMP en el RPi [48].

Para la instalación sobre sistemas operativos Linux se utiliza el agente net-snmp, este posee soporte para Raspberry Pi, el comando por utilizar para la instalación se puede observar en el Código 2.16.

```
> apt-get install snmp snmpd
```

Código 2.16. Comando instalación snmp en RPi

Se debe modificar el contenido del archivo de configuración snmpd.conf localizado en el path `/etc/snmp/`, el Código 2.17 muestra lo que debe ser añadido en dicho archivo.

```
#
#  AgentX Sub-agents
#

 master          agentx


proxy -c public -v 2c localhost:8001 .1.3.6.1.4.1
```

Código 2.17. Modificación de archivo snmpd.conf.

La primera modificación es quitar el numeral la declaración del uso de “master agentx”, y lo siguiente es añadir el código de uso de proxy para la rama OID perteneciente al nodo “enterprises”. La dirección de respuesta es apuntada al puerto 8001, este puerto deberá ser especificado después en la configuración del agente. El proxy permite que todo aquello que el agente principal no conozca pueda ser respondido por subagentes si estos lo conocieran.

2.2.3.2.2. Instalación de la plataforma

Para obtener el kit se debe acceder a la página oficial de webnms en la sección javaagent [26],. Se debe seleccionar el sistema operativo sobre el cual se va a trabajar, en este caso es Linux. También se debe seleccionar si se desea descargar con jre o sin jre, se recomienda descargar sin jre debido a que con la otra opción se presentan problemas para la configuración de java.

Una vez descargado el archivo a través del terminal de Linux se debe acceder a dicho directorio y descomprimir el archivo. En el Código 2.18 se encuentra el comando a utilizar para descomprimirlo y después se enlista la carpeta recién extraída.

```
pi@raspberrypi:~/Downloads $ ls | grep WebNMS_Java_AgentToolkit.zip
WebNMS_Java_AgentToolkit.zip
pi@raspberrypi:~/Downloads $ unzip WebNMS_Java_AgentToolkit.zip
```

```
pi@raspberrypi:~/Downloads $ ls | grep WebNMS
WebNMS
WebNMS Java AgentToolkit.zip
```

Código 2.18. Comando descompresión de archivos zip

De aquí en adelante es necesario ser usuario root para terminar el resto de la configuración. Para colocarse como usuario *root*, es necesario utilizar el comando “sudo antes de cada comando.

El siguiente paso es la comprobación de la versión existente de java, según los requerimientos, el único requisito para el funcionamiento del kit WebNMS es la presencia de java en la máquina. Sin embargo, cuando se realiza la instalación recomendada y se intenta utilizar cualquier aplicación se obtiene el siguiente mensaje como el de la Figura 2.41.

```
root@debian9:/home/debian9/Downloads/WebNMS_Java_AgentToolkit/WebNMS/JavaAgent/b
in# ./JmxCompiler.sh

########################    !   WARNING   !  ################################
##############
### java source compilation  needs JAVA_HOME/lib/tools.jar
            ##
### jdk only contains this jar and not jre.
            ##
### It seems that JRE is set as JAVA_HOME in this console now.
            ##
### Please correct this. If not so, NoClassDefFoundError may occur on source com
pilation.   ##
### This is only a warning message...
            ##
################################################################################
##############
```

Figura 2.41. Advertencia librería faltante.

La librería faltante es “tools.jar” y también se indica que dicha librería se encuentra en un JDK. El RPi al ser una plataforma de desarrollo, posee por defecto instalado una versión de JDK, sin embargo, debe esta ser la seleccionada por defecto a través de la publicación en JAVA_HOME. Como primer paso se debe localizar la carpeta del JDK que por defecto suele estar dentro del directorio `/usr/lib`. Al tener el path se debe escribir dicho path en las variables ambientales, el comando para abrir el archivo de declaración de las variables locales se muestra en el Código 2.19.

```
pi@raspberrypi:~/Downloads/WebNMS/JavaAgent/bin $ sudo nano /etc/environment
```

Código 2.19. Archivo configuración variables ambientales.

La forma de escribir la ruta se muestra en el Código 2.20.

```
 GNU nano 2.7.4                                  Fichero: /etc/environment

JAVA_HOME=/usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt
```

Código 2.20. *Declaración JAVA_HOME*

Una vez listo esto se debe exportar dichas variables, para esto se deben ejecutar dos comandos, seguido de un reinicio del sistema. En el Código 2.21 puede visualizarse dichos comandos.

```
pi@raspberrypi:~/Downloads/WebNMS/JavaAgent/bin $ export PATH=$PATH:$JAVA_HOME/bin
pi@raspberrypi:~/Downloads/WebNMS/JavaAgent/bin $ export
pi@raspberrypi:~/Downloads/WebNMS/JavaAgent/bin $ sudo reboot
```

Código 2.21. Actualización de variables ambientales.

Una vez listo el requisito del uso del JDK se procede a lanzar la herramienta JMXCompiler.

2.2.3.2.3. Configuración de parámetros del agente

Una vez ingresado a la herramienta, después de la creación de un nuevo proyecto el paso principal es cargar una MIB, la MIB creada puede encontrarse en la carpeta mibs dentro del directorio de WebNMS. En la Figura 2.42 se muestra la apariencia al cargarse la MIB.

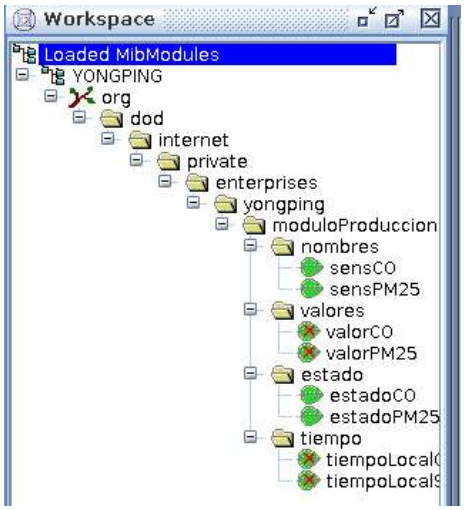


Figura 2.42. MIB cargada en JMXCompiler

A continuación, se deben definir los parámetros para la generación del código esqueleto del agente, para esto se debe ingresar a la pestaña de configuraciones. En el apartado de generación de código fuente se selecciona el uso de archivos de texto como modelo de almacenamiento de escalares. En la Figura 2.43 se muestra dichas opciones de configuración.

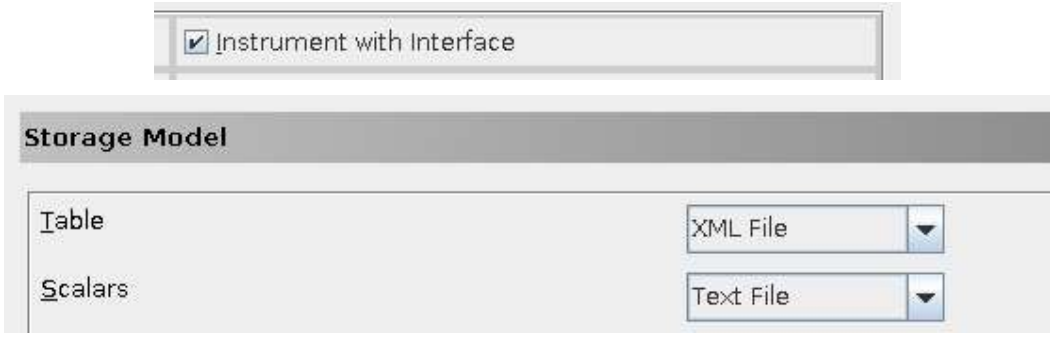


Figura 2.43. Parámetros configuración

Lo siguiente por hacer es la elección de los adaptadores y conectores a utilizar, en primer lugar, en la sección SNMP, se debe registrar el uso del dicho adaptador, así como seleccionar el puerto a utilizar y su versión. En la Figura 2.44 se muestra dicha configuración.

Figura 2.44. Selección de adaptador SNMP

A continuación, se procede con la elección del conector RMI, en este también debe elegirse el número de puerto y seleccionar su uso. En la Figura 2.45 se muestra su aspecto visual.

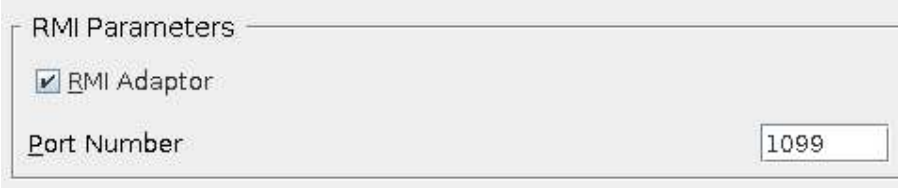


Figura 2.45. Selección de conector RMI

Todos los demás adaptadores y conectores deben ser desactivados para que la generación del agente no tenga código innecesario y demande mayor recurso para la máquina. De igual forma los servicios del agente JMX o propios de la herramienta.
En este punto se continúa con la generación del código fuente. Al ser elegida la generación, se presentan 4 interfaces y 7 clases Java, de las cuales 4 corresponden a los nodos OID y las 3 restantes a la configuración del agente. La Figura 2.46 muestra dichos archivos MIB.

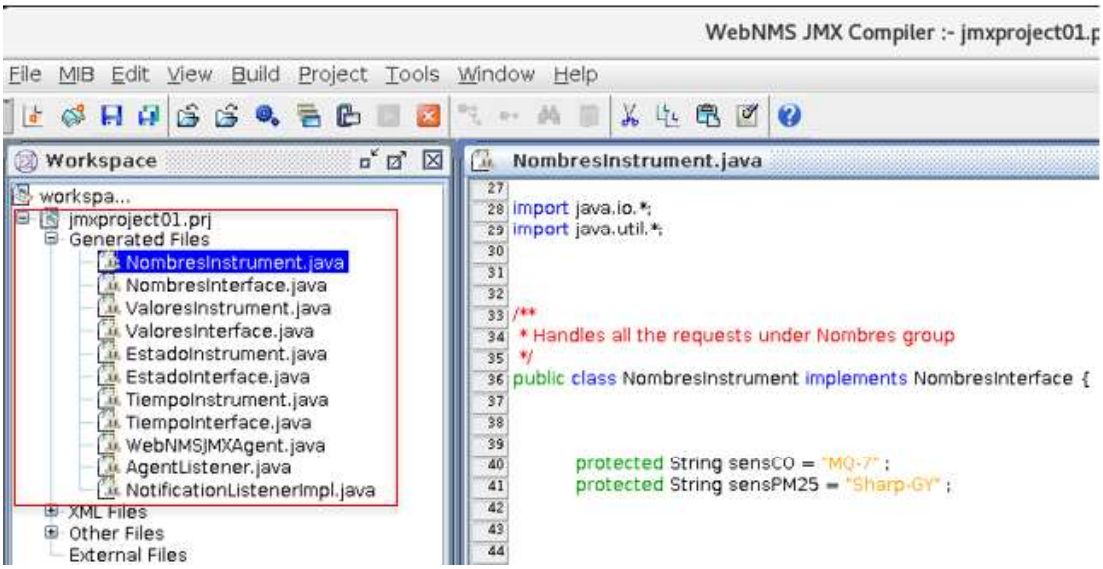


Figura 2.46. Archivos JAVA generados

Por cada grupo definido en la MIB, se visualiza una clase y su interfaz, en estos se definen los métodos getters y setters correspondientes a cada objeto de dichos grupos. Por ejemplo la interfaz *NombresInterface.java* contiene los métodos getSensCO() y setSensCO(), considerando que en el OID Nombres en el archivo MIB se tiene el objeto sensCO.

El archivo con terminación Instrument contiene el código que define el comportamiento de los métodos de las interfaces, dependiendo si, los permisos de acceso de dichos objetos MIB son de escritura o no, los métodos setNombre, especificarán acceso o no.

Cuando un objeto MIB se define con acceso de lectura y escritura, el código esqueleto generado posee métodos para acceder a los archivos de los cuales se leerá la información sobre escritura. Sin embargo, cuando los objetos son únicamente de lectura, dicho código

no existe. Por lo cual en este momento la herramienta JMX Compiler puede ser utilizada como un IDE de desarrollo del agente.

Tomando en consideración que, según el diseño definido en el literal a.3 de la sección 2.1.4 los objetos pertenecientes a los nodos Valores y Tiempo tienen permiso únicamente de lectura entonces se debe añadir código que defina el procedimiento a seguir cuando se acceda a los métodos get de cada uno.

Para añadir código extra al esqueleto generado por default, se deben utilizar etiquetas que definen el inicio y fin del espacio que contendrá dicho código. En la Figura 2.47 se muestran dichas etiquetas.

```
/* User code starts here */
/* User code ends here */
```

Figura 2.47. Etiquetas inicio y fin código usuario

A continuación, se presenta el Código implementado para el manejo de los valores correspondiente al objeto valorCO de la MIB, el nombre del archivo java a manipular es ValoresInstrument.java y este implementa la interfaz ValoresInterface.java.

En primera instancia se define un objeto del tipo String que será el valor retornado del método get, esto se muestra en el Código 2.22.

```
protected String cero = "0.0";
```

Código 2.22. Declaración objeto Long

El siguiente paso es implementar código necesario para el getter, este método será utilizado en cada ocasión que se desee acceder a dichos valores, tanto a través de RMI como a través de SNMP. El Código 2.23 muestra dicha implementación. La devolución del valor depende del valor perteneciente al objeto estadoCO, pues si se tiene como inhabilitado al sensor entonces no debe retornar valores que pueden ser erróneos.

```
public String  getValorCO()
                        throws AgentException
{
        // Fill up the stub with required processing
        try{
                BufferedReader bf = new BufferedReader(new FileReader("/home/pi/Downloads/WebNMS/JavaAgent/fileToScalar/valorCO.txt"));
                String inV = bf.readLine();


                bf =   new BufferedReader(new FileReader("/home/pi/Downloads/WebNMS/JavaAgent/fileToScalar/estadoCO.txt"));
                String inE = bf.readLine();
                Integer ent = new Integer(inE);
                if(ent.intValue()==1)
                        valorCO = inV;
                else
                        valorCO = cero;
                        CommonUtils.writeIntoFile("fileToScalar/", "valorCO.txt", cero);
        }catch(Exception e){
            e.getMessage();
        }
        return valorCO;
}
```

Código 2.23. Código correspondiente a getValorCO( )

Debido a que al abrir un archivo puede dar errores por diferentes motivos, todo el código se encapsula dentro de una declaración try-catch, el archivo del cual obtener los datos es ubicado dentro del path del agente en la subcarpeta fileToScalar, este mismo archivo es el utilizado para el guardado de los valores en el módulo de adquisición. El archivo contendrá únicamente un valor por lo que se utiliza el método readLine() para la extracción, se guarda en un dato de tipo String para después asignarlo a la variable de retorno del método.

Otro nodo que contiene objetos con acceso de lectura es el llamado Tiempos, este presenta un objeto del tipo String y otro del tipo Long. El archivo por modificar es el llamado TiempoInstrument.java. El primer paso es la declaración de los objetos a retornar en cada método, así como la declaración del formato a utilizar en el objeto de tipo String. Esto se visualiza en el Código 2.24

```
public static DateFormat ndf = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

protected Long  tiempoLocalGauge = new Long(), //f ;
protected String tiempoLocalString = new String(); //ndf.format(now) ;
```

Código 2.24. Objetos declarados archivo TiempoInstrument.java

En el espacio de Código 2.25 se muestra la codificación implementada para el manejo de peticiones get del objeto MIB TiempoLocalGauge, el primer paso es la creación de una instancia del tipo Date, el cual obtiene la fecha y hora del sistema, debido a que el interés es únicamente la hora se utiliza el método getTime() y este se lo guarda en una variable del tipo long, conociendo que el método retorna un objeto del tipo Long, se realiza la conversión de variable long a objeto Long y se igualan los nombres de los objetos.

```
public Long  getTiempoLocalGauge()
                        throws AgentException
{
        // Fill up the stub with required processing
        now = new Date();
        long i = now.getTime();
        tiempoLocalGauge = new Long(i);
        return tiempoLocalGauge;
}
```

Código 2.25. Método getTiempoLocal()

Las modificaciones necesarias únicamente deben ser efectuadas en los archivos de instrumentación de los objetos, el archivo que contiene toda la lógica del funcionamiento del agente es WebNMSJMXAgent.java, incluyendo el registro de los adaptadores y el registro de los MBeans.

El siguiente y último paso es la compilación de la fuente, es decir a través de este paso se crea el agente en un solo proyecto independiente, la Figura 2.48 muestra la ventana de elección cualquier cambio requerido a partir de aquí, deberá ser regenerado un nuevo agente pues los cambios no se reflejarán.

Figura 2.48. Selección compilar fuente

Cuando el agente fue correctamente compilado, en el recuadro inferior se indicará, el éxito o fracaso también de su compilación, si todo fue exitoso se habilitará la opción de iniciar el agente.

2.2.4. IMPLEMENTACIÓN DEL MÓDULO DE VISUALIZACIÓN

2.2.4.1. Configuración de la base de datos

La base de datos es el primer componente importante de este módulo, como se indicó en la sección 2.1 correspondiente al diseño, se utilizará el motor postgresql, este motor es orientado al trabajo con herramientas open source y esto indica una compatibilidad con la plataforma RPi.

2.2.4.1.1. Instalación del motor

Para la instalación del motor, basta con un comando para instalarlo, así como componentes extras necesarios para su correcto funcionamiento. El Código 2.26 muestra el comando ejecutado.

```
pi@raspberrypi:~ $ sudo apt install postgresql libpq-dev postgresql-client postgresql-client-common -y
```

Código 2.26. Comando instalación postgresql

En este momento el servidor de la base de datos se encuentra listo para ser usado, se debe añadir el permiso correspondiente para acceso remoto, esto se lo realiza editando el archivo `/etc/postgresql/9.4/main/pg_hba.conf` y añadiendo la línea de texto correspondiente al Código 2.27:

```
host  all  all  192.168.1.0/24 trust
```

Código 2.27. Comando por añadir en pg_hba.conf

Considerando esa dirección definida como la de la red, se debe además añadir en el archivo /etc/postgresql/9.4/main/postgresql.conf una nueva línea que debe situarse en la sección connection and authentication, esto es mostrado en el Código 2.28.

```
#------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------

# - Connection Settings -
listen_addresses = '*'
```

Código 2.28. Comando por añadir en postgresql.conf

Una vez concluidas estas acciones se procede a reiniciar el servicio utilizando el comando mostrado en el Código 2.29, así como comprobar que dicho servicio se encuentre activo.

```
pi@raspberrypi:~ $ sudo systemctl restart postgresql
pi@raspberrypi:~ $ systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled)
   Active: active (exited) since Mon 2018-09-24 03:42:45 UTC; 12s ago
  Process: 5734 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 5734 (code=exited, status=0/SUCCESS)
```

Código 2.29. Comando de reinicio y revisión de servicio.

La siguiente acción por realizar es la creación de la contraseña del usuario por defecto llamada postgres. Para realizar dicha acción se debe ingresar como usuario postgres, el Código 2.30 muestra el comando para realizar esto.

```
pi@raspberrypi:~ $ sudo su postgres
```

Código 2.30. Creación de usuario postgres

Una vez que se fija el usuario postgres, se prosigue con el comando mostrado en el Código 2.31 para abrir la línea de comandos y posterior a ello ingresar el comando de modificación de la contraseña.

```
npostgres@raspberrypi:/home/pi$ psql
psql (9.4.19)
Type "help" for help.

postgres=# alter user postgres with password 'postgres';
ALTER ROLE
```

Código 2.31. Terminal psql y creación de usuarios

2.2.4.1.2. Instalación de la herramienta de administración

La versión que será utilizada es la 4, esta herramienta permite la conexión remota si la gestión se realiza desde la misma red, por lo cual es multiplataforma y puede ser gestionada tanto desde una máquina Linux como Windows. Para distribuciones Debian, se debe añadir el repositorio que contiene dicha herramienta, mientras que para Windows únicamente se debe descargar e instalar desde la página oficial de pgadmin.
El Código 2.32 muestra el comando para la instalación en Raspbian.

```
pi@raspberrypi:~ $ sudo apt-get install pgadmin4
```

Código 2.32. Comando instalación pgadmin

El primer paso es la creación de una base de datos desde el explorador lanzado por la herramienta, la Figura 2.49 muestra la respuesta después de dar un click izquierdo sobre la etiqueta de Database.

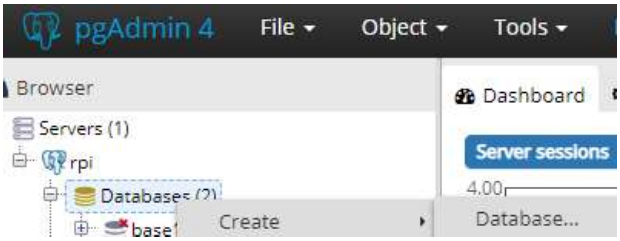


Figura 2.49. Creación de base de datos en pgadmin

Con la base de datos creada se procede a la creación de las tablas, para crear una nueva tabla se debe dar un click izquierdo sobre la etiqueta de “Tables” y seleccionar Create> “Table”. Esto se visualiza en la Figura 2.50.

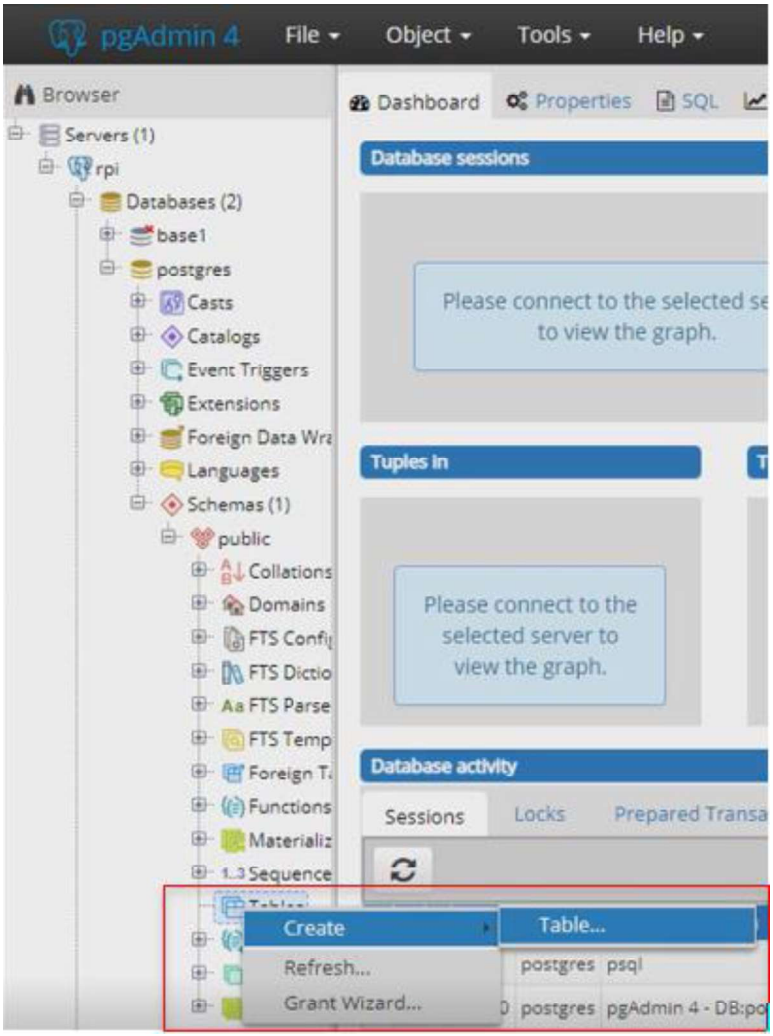


Figura 2.50. Creación de tabla en pgadmin

El nuevo recuadro que aparece muestra una tabla sin contenido, para añadir una nueva columna a la tabla se selecciona el botón “+” localizado en el vértice superior derecho, mediante este se puede añadir una nueva columna de la tabla. El ejemplo presentado en la Figura 2.51 corresponde a la tabla “RPi”, esta contiene 3 columnas, en cada una de las columnas se define el tipo de dato soportado, puede definirse su longitud, así como si el campo puede ser nulo o no y si ese atributo puede considerarse como clave primaria.

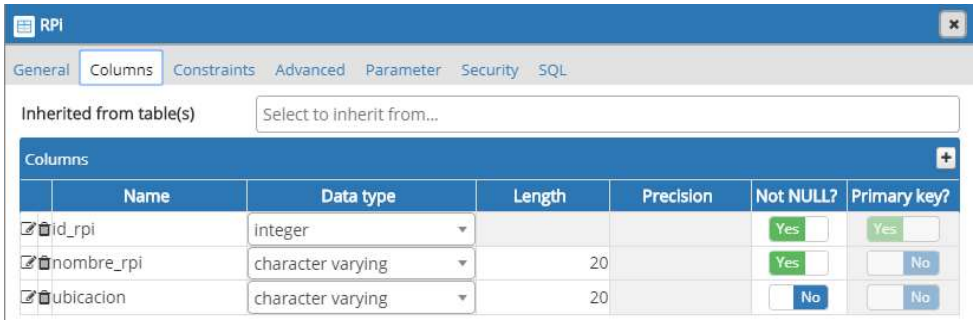


Figura 2.51. Creación columnas de tabla RPi en pgadmin.

Una vez realizadas las tablas se puede definir elementos extras como por ejemplo la secuencia de incrementación de las columnas de identificación de cada tabla, o las declaraciones de claves foráneas. Concluidas esas tareas se procede a llenar datos conocidos en las tablas existentes. La Figura 2.52 muestra los datos de los tipos de sensores por utilizar.

Figura 2.52. Datos Tabla tipo sensores

2.2.4.1.3. Creación de scripts para añadir datos a la tabla

La adición de un nuevo dato correspondiente a cada lectura de los sensores debe ser accionado a partir de la ejecución de un determinado script, el período de ejecución se define en el archivo crontab. Conociendo que la base de datos es alojada en un RPi distinto a aquel que contiene al agente, se debe utilizar direcciones ip fijas.

Para la inserción de datos se utilizará un script sql que contiene una consulta de inserción y que será ejecutado a partir del script de bash que es accionado por el crontab.
Una parte del script de bash puede visualizarse en el Código 2.33, en este se definen las variables necesarias para realizar la consulta.

```
url=192.168.1.27
user=postgres
database=rpi
password=postgres
fileName1="/home/pi/Downloads/WebNMS/JavaAgent/fileToScalar/valorCO.txt"
fileName2="/home/pi/Downloads/WebNMS/JavaAgent/fileToScalar/valorPM25.txt"

export PGPASSWORD=$password

#minuto=`date +%M`
#rest=`expr $minuto % 5`
#echo $rest

for i in 1 2 3 4 5 6
do
    valor1=`cat $fileName1`
    valor2=`cat $fileName2`
    fecha=`date +"%Y-%m-%d %H:%M:%S"`
    echo $fecha
```

Código 2.33. Script de inserción en base de datos

Para la inserción de los datos en la tabla, se depende del valor guardado en los archivos de texto, pues únicamente existirá un cero si es que se desconectó o se inhabilitó un sensor, de tal manera que cuando el valor contenido sea cero, no se guardará un nuevo dato. Esta condición se presenta en el Código 2.34. El comando psql permite ingresar al terminal de comandos de postgresql.

```
if [ $valor1 != '0.0' ]
then
   echo "insert into lectura(valor,fecha,id_sensor,estado) values($valor1,'$fecha', 1,true);" > /home/pi/insert.sql
   psql -h $url -U $user -d $database -f /home/pi/insert.sql
fi
```

Código 2.34. Porción de código script insert_in_tables.sh

Para cada sensor se utiliza un lazo diferente, pues la condición es independiente por sensor.

2.2.4.2. Codificación de la aplicación

El IDE NetBeans 8.2 se utilizó para el desarrollo de la aplicación, y Java es el lenguaje de programación.

2.2.4.2.1. Clase MainProgram

Esta clase contiene la lógica de conexión y obtención de los MBeans, en la parte del Código Fuente, en el Código 2.35 se muestra el método readConfiguration(), el cual permite definir las propiedades a utilizar en la conexión RMI. Como son la dirección del servidor y el puerto.

```
private boolean readConfiguration() {
    try {
        Properties properties = new Properties();
        File file = new File(this.getClass().getProtectionDomain().getCodeSource().getLocation().toURI());
        InputStream inputStream = new FileInputStream(file.getParent() + System.getProperty("file.separator")
                + "rmi.properties");
        properties.load(inputStream);
        url = properties.getProperty("url");
        host = properties.getProperty("ip_server");
        port = Integer.valueOf(properties.getProperty("port_rmi"));
        return true;
    } catch (FileNotFoundException ex) {
        System.out.println(ex.getMessage());
        JOptionPane.showMessageDialog(this, ex.getMessage());
    } catch (IOException | URISyntaxException ex) {
        System.out.println(ex.getMessage());
        JOptionPane.showMessageDialog(this, ex.getMessage());
    }
    return false;
}
```

Código 2.35. Método readConfiguration() de MainProgram.java

Una vez que las propiedades de conexión fueron establecidas, se prosigue a realizar la conexión con el agente, el Código 2.36 muestra el método connectAgent() el cual utiliza la clase “ClientFactory” y, el método “createClient” utilizando RMI como argumento para definir la variable “client”. Así como la habilitación del método “Heartbeat” útil para la verificación de la conexión.

```
private boolean connectAgent() {
    try {
        client = ClientFactory.createClient(protocol);
        client.enableHeartBeat(true);
        client.setHeartBeatRate(10000);
        client.addHeartBeatListener(this);

        if(client != null)
            client.connect(host, port, null);

        return true;
    }catch (Exception e) {
        System.out.println(e.getMessage());
        JOptionPane.showMessageDialog(this, e.getMessage());
    }
    return false;
}
```

Código 2.36. Método connectAgent() de MainProgram.java

Una vez que el método connectAgent() se haya presentado como satisfactorio y efectivamente un cliente se encuentre disponible, se procede a recuperar todos los MBeans de dicho cliente, el Código 2.37 muestra el método utilizado para recuperar dichos MBeans.

```
private void assignMBeans() {
    try {
        QueryExp query = null;
        Set names = client.queryNames(new ObjectName(mbeanName), query);
        Object[] mbeans = names.toArray();

        MBeanInfo info;

        if (mbeans != null && mbeans.length > 0) {
            for (Object mbean : mbeans) {
                info = client.getMBeanInfo((ObjectName)mbean);
                MBeanAttributeInfo[] attributes = info.getAttributes();

                if (info.getClassName().contains("NombresInstrument"))
                    MBeanNombres = mbean;
                else if (info.getClassName().contains("ValoresInstrument")) {
                    MBeanValores = mbean;
                   for (MBeanAttributeInfo attribute : attributes) {
                       if (attribute.getName().contains("CO")) {
                           coAttribute = attribute;
                       }
                       else {
                           pm25Attribute = attribute;
                       }
                   }
                }
                else if (info.getClassName().contains("EstadoInstrument"))
                    MBeanEstados = mbean;
                else if (info.getClassName().contains("TiempoInstrument")) {
                    MBeanTiempo = mbean;
                    timeAttribute = attributes[0];
                }
            }
        }
    }catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
```

Código 2.37. Método assignMBeans() de MainProgram.java

Dependiendo de cual MBean sea, es asignado al objeto creado para cada uno de ellos, y su posterior uso cuando sea requerido. Son dos las interfaces de interacción con el usuario, la interfaz o pantalla monitor y la interfaz o pantalla de reportes.

2.2.4.2.2. Clase Monitor

En la pantalla monitor se muestra en la Figura 2.53, ésta despliega las gráficas de los valores recuperados de cada sensor, también indica el grado de alerta actual del valor recogido a través de la bandera, así como presentar una etiqueta de texto que indica si el

sensor se encuentra en funcionamiento, junto a esta etiqueta se encuentra un botón que permite desactivar al sensor y no graficar las lecturas de dicho sensor.

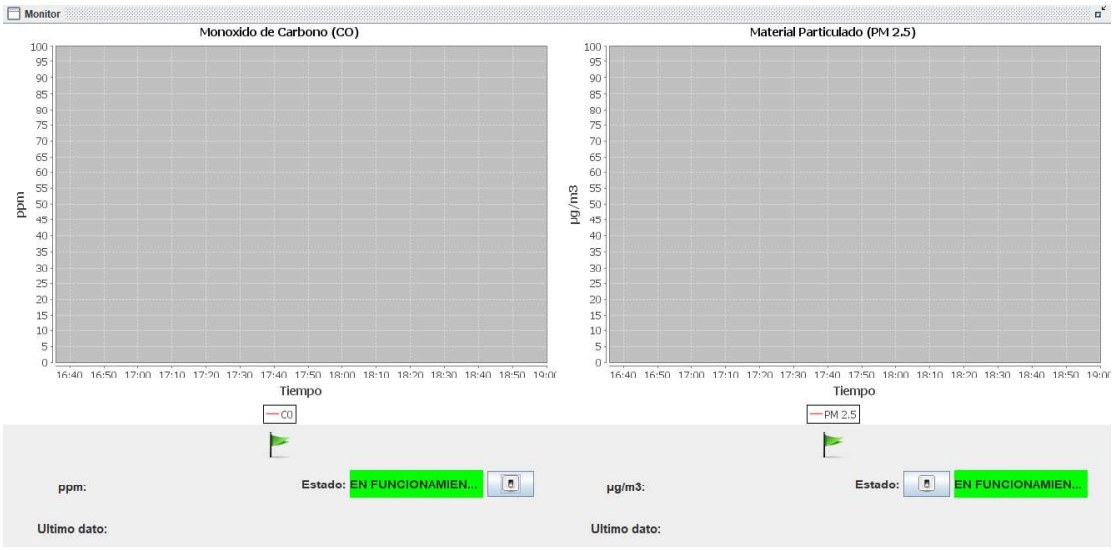


Figura 2.53. Pantalla Monitor

El Código 2.38 muestra el método initExtraComponents() el cual contiene la creación de las instancias de la colección de series de tiempo de cada sensor, mismas que sirven para la creación de las instancias de los gráficos por crear y cargar en los paneles correspondientes.

```
public void initExtraComponents() {
    timeseries1 = new TimeSeries("C0", Second.class);
    timeseries2 = new TimeSeries("PM 2.5",Second.class);
    timeseriesCollection1 = new TimeSeriesCollection();
    timeseriesCollection1.addSeries(timeseries1);

    timeseriesCollection2 = new TimeSeriesCollection();
    timeseriesCollection2.addSeries(timeseries2);

    jfreeChart1 = createChart(timeseriesCollection1,
        "Monoxido de Carbono (CO)","Tiempo","ppm",100,0);

    jfreeChart2 = createChart(timeseriesCollection2,
        "Material Particulado (PM 2.5)","Tiempo","µg/m3",100,0);

    ChartPanel chartPanel1 = new ChartPanel(jfreeChart1);
    ChartPanel chartPanel2 = new ChartPanel(jfreeChart2);
    chartPanel1.setVisible(true);
    chartPanel2.setVisible(true);
    jPanel1.add(chartPanel1);
    jPanel1.add(chartPanel2);
}
```

Código 2.38. Método initExtraComponents() en clase Monitor.java

Un método importante en esta clase es setReadingCO el cual se muestra en el Código 2.39 y permite al recibir un objeto del tipo Lectura añadir el nuevo dato de CO para graficarlo.

```
public void setReadingCO(Lectura lectura, String color) {
    jLabel13.setIcon(new javax.swing.ImageIcon(
            getClass().getResource("/images/"+color+".png")));
    timeseries1.add(new Second(), lectura.getValor());
    jLabel6.setText(String.valueOf(lectura.getValor()));
}
```

Código 2.39. Método setReadingCO en clase Monitor.java

2.2.4.2.3. Clase Reportes

La pantalla reportes se muestra en la Figura 2.54, ésta permite realizar consultas sobre datos históricos de valores de cada sensor, el cual debe ser seleccionado mediante el combo-box, además del rango de fechas de consulta.

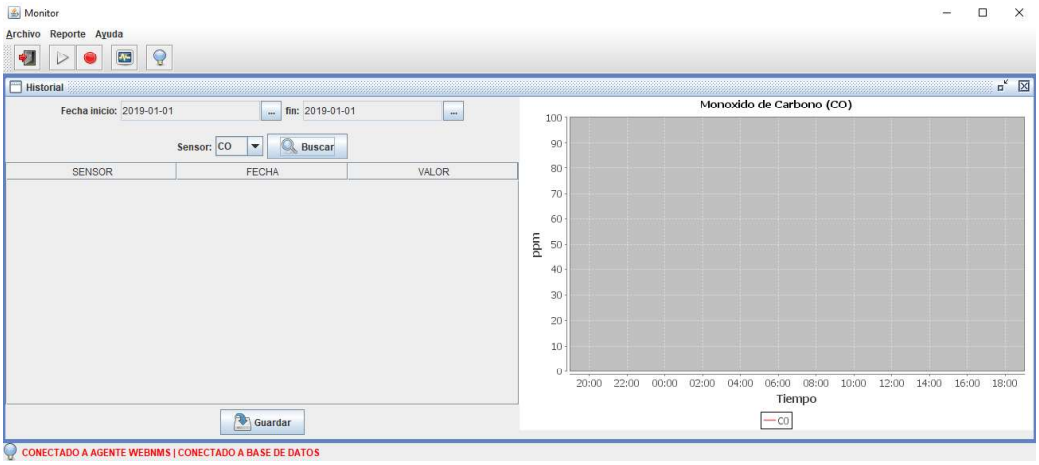


Figura 2.54. Pantalla de Reporte

El espacio del código correspondiente a esta pantalla se despliega en el Código 2.40, en este se muestra el constructor de la clase el cual recibe un objeto del tipo EntityManager y lo empata con el objeto instanciado en la clase, además se definen propiedades por utilizar en la gráfica a dibujar.

```
public Reporte(EntityManager entityManager) {
    this.entityManager = entityManager;
    Properties properties = new Properties();
    properties.put("text.today", "Today");
    properties.put("text.month", "Month");
    properties.put("text.year", "Year");
    utilDateModel1 = new UtilDateModel();
    utilDateModel1.setDate(2010, 0, 1);
    utilDateModel1.setSelected(true);
    utilDateModel2 = new UtilDateModel();
    utilDateModel2.setDate(2018, 0, 1);
    utilDateModel2.setSelected(true);
    jDatePanel1 = new JDatePanelImpl(utilDateModel1, properties);
    jDatePanel2 = new JDatePanelImpl(utilDateModel2, properties);
    jDatePicker1 = new JDatePickerImpl(jDatePanel1, new DateLabelFormatter());
    jDatePicker2 = new JDatePickerImpl(jDatePanel2, new DateLabelFormatter());
    simpleDateFormat1 = new SimpleDateFormat("yyyy-MM-dd");
    simpleDateFormat2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    initComponents();
    initExtraComponents();
    loadSensors();
}
```

Código 2.40. Constructor clase Reporte

Una parte del método correspondiente a la acción de presión del botón de consulta del reporte se despliega en el Código 2.41, este permite realizar la consulta según el sensor seleccionado y el rango de fechas. Para la realización de la consulta se utiliza la clase Query y el objeto entityManager instanciado en la clase actual.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        StringBuilder consulta = new StringBuilder();

        Query query = entityManager.createNamedQuery("Sensor.findByNombreSensor");
        query.setParameter("nombreSensor", jComboBox1.getSelectedItem().toString());
        Sensor sensor = (Sensor)query.getSingleResult();

        consulta.append("select l from Lectura l where l.fecha between "
                + ":fechaInicio and :fechaFin and l.idSensor = :idSensor "
                + "order by l.idSensor, l.fecha");

        query = entityManager.createQuery(consulta.toString());
        query.setParameter("fechaInicio", simpleDateFormat1.parse(jDatePicker1.getJFormattedTextField().getText()));
        query.setParameter("fechaFin", simpleDateFormat1.parse(jDatePicker2.getJFormattedTextField().getText()));
        query.setParameter("idSensor", sensor.getIdSensor());

        List<Lectura> listaLecturas = query.getResultList();
```

Código 2.41. Método JButtonActionPerformed() en pantalla Reporte

3. RESULTADOS Y DISCUSIÓN

En este apartado se presentan los resultados de cada uno de los módulos implementados y de la funcionalidad del prototipo en sí.

3.1. TABLERO KANBAN

El tablero Kanban presentado en la Figura 3.1 describe los últimos pasos a seguir para completar el desarrollo del presente proyecto, restan actividades de pruebas de funcionamiento, así como la finalización de detalles de la sección previa.

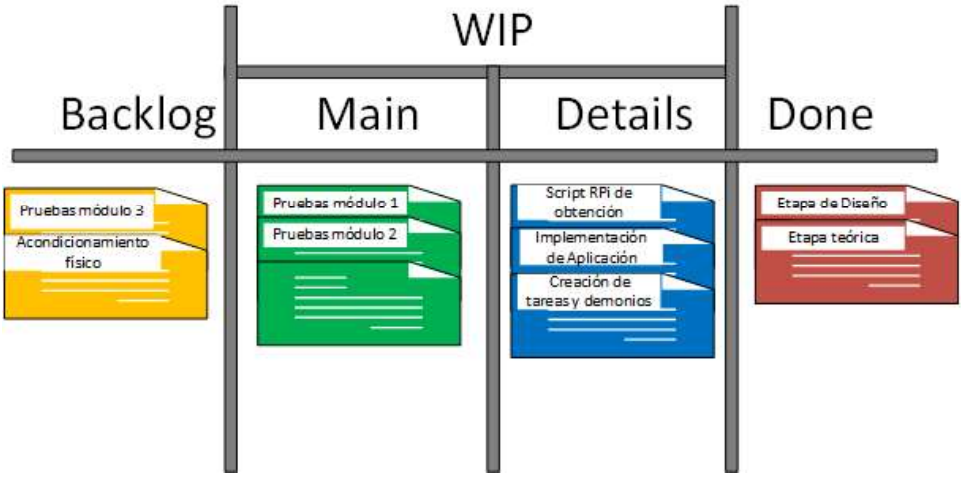


Figura 3.1. Tablero Kanban de fase de pruebas

3.2. PRUEBAS DE FUNCIONAMIENTO

3.2.1. PRUEBA DE FUNCIONAMIENTO DE MÓDULO DE ADQUISICIÓN

En este apartado se presentan las pruebas de funcionamiento según el camino que siguen los datos desde que son obtenidos de los sensores hasta cuando son guardados en los archivos de texto.

3.2.1.1. Envío de datos desde Arduino

El funcionamiento de los sensores significa que se ha seguido las recomendaciones de los fabricantes y que funcionan adecuadamente, se comprobó que el sensor MQ-7

conmuta en el tiempo determinado y el voltaje de alimentación también lo hace. Esta conmutación corresponde al proceso de calentamiento que debe ser sometido el sensor. La Figura 3.2 muestra las condiciones del sensor en el segundo 1, que corresponde al voltaje en alto a ~5 [V].

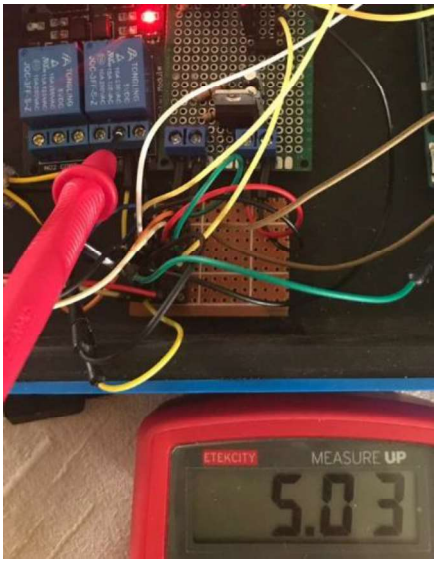


Figura 3.2. Voltaje en alto con relé

Mientras que la Figura 3.3 muestra las condiciones del sensor en el segundo 61, y corresponde al voltaje en bajo a ~1.4 [V].

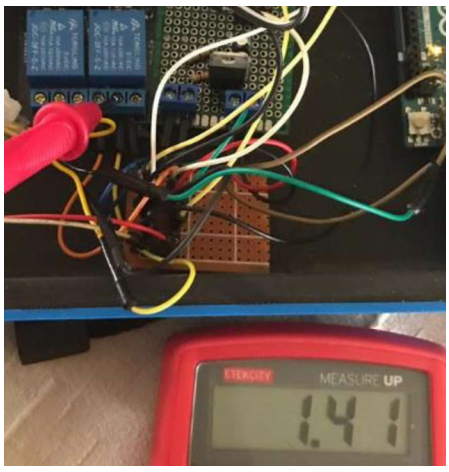


Figura 3.3 Voltaje en bajo con relé

La recolección de datos desde los sensores y el armado de la trama para el envío por el puerto serial fueron realizadas, el primer paso para la verificación es obtener la trama

enviada a través del puerto serial. Esto se logra verificando que se transmite a través del puerto serial propio de Arduino. A continuación, en la Figura 3.4 se presenta el resultado obtenido en dicho monitor serial.

Figura 3.4. Monitor Serial Arduino visualización de tramas

Se modificó el script original por propósito de mostrar ejemplo de uso para que pueda entregarse el valor numérico previo a la trama por facilidad de lectura debido a que a través del monitor serial se imprime la trama en caracteres ASCII, algunos de ellos son malinterpretados y aparenta pérdida de información, pero, en el siguiente apartado puede identificarse toda la trama completa en valor decimal.

3.2.1.2. Obtención y guardado en archivos de texto

El siguiente paso en el flujo de información es la transformación y guardado de los datos obtenidos desde el puerto serial en los archivos de texto. Debido a que un puerto serial no puede trabajar o ser leído en simultaneidad, se debe realizar la verificación con otra trama de ejemplo y esta vez utilizando el script de Python. Para propósitos de verificación se insertó instrucciones de impresión en cada función del script para poder obtener los valores a través del terminal de Linux.

La Figura 3.5 muestra los valores recibidos hasta que la función identifique una trama válida.

```
root@raspberrypi:/home/pi/Downloads# python3 readNsave2File.py
778132112114105110116583232505055108077321121141051101165832325351101262270530241126
```

Figura 3.5. Identificación de trama válida

La trama identificada se desglosa, se identifica cada campo y se presenta en la Figura 3.6.

126	227	0	53	0	24	1	126
bandera	LSB(CO)	MSB(CO)	LSB(PM)	MSB(PM)	LSB(CRC)	MSB(CRC)	Bandera

Figura 3.6. Trama identificada en puerto serial

Siendo los términos:

- LSB(CO): Byte menos significativo de CO.
- MSB(CO): Byte más significativo de CO.
- LSB(PM): Byte menos significativo de PM.
- MSB(PM): Byte más significativo de PM.
- LSB(FCS): Byte menos significativo del FCS.
- MSB(FCS): Byte más significativo del FCS.

Siguiendo con el proceso del script, la Figura 3.7 muestra un ejemplo de la transformación de los campos de la trama a valores reales considerando un sensor. El primer paso es la comprobación del FCS, como es 2 bytes entonces se transforma a un solo valor decimal, para lo que se realiza la conversión del byte más significativo a su valor original y se suma el valor del byte menos significativo. Este valor es el comparado con el valor del campo FCS.

$$FCS\ Suma = 24 + \ (1 \ll 8) = 24 + 256 = 280$$

$$Campo\ FCS\ = 280$$

Figura 3.7. FCS transmitido

Lo explicado puede comprobarse en la Figura 3.8 que presenta otra trama recibida, en esta se imprime en pantalla las salidas de comprobación de trama identificada y de FCS.

```
pi@raspberrypi:~/Downloads $ sudo python3 readNsave2File.py

352368711065874126227053024 1126
1262270530241126[227, 0, 53, 0, 24, 1]
stf

trama valida suma=280 y fcs=280
```

Figura 3.8. FCS verificado según datos recibidos

Al coincidir ambos valores se continúa con las demás funciones del script hasta guardar los valores reales en los archivos de texto. Por motivos de ejemplo para facilidad del lector,

se reemplaza las fórmulas de calibración por una división sobre 3 de estos valores utilizados, de tal manera que debe guardarse 75.66 y 17.66 en los archivos valorCO.txt y valorPM25.txt respectivamente. Esto puede comprobarse en la Figura 3.9.

Figura 3.9. Valores ejemplo guardados en archivos de texto

Estos archivos de texto son utilizados por el agente en el módulo agente y también utilizados para su inserción en la base de datos correspondiente al módulo visualización.

3.2.2. PRUEBA DE FUNCIONAMIENTO DE MÓDULO AGENTE

En este apartado se presentan las pruebas de funcionamiento del módulo 2 correspondiente a la puesta a punto del agente, y el uso de la MIB definida.

3.2.2.1. Funcionamiento agente por SNMP

Para comprobar el funcionamiento del agente a través del protocolo SNMP se utilizarán comandos de consulta SNMP a la dirección IP del agente, en este caso del RPi-agente. Se inicia con el comando snmpget en versión 2c, la Figura 3.10 muestra el comando pidiendo información sobre el OID correspondiente al sensor de monóxido de carbono llamado “sensCO”.

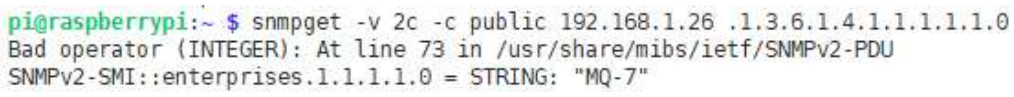


Figura 3.10. Consulta snmpget a la MIB

Considerando la sección del diseño de la MIB, el objeto sensCO es del tipo string y define un acceso de lectura y escritura, es decir es apto para emplear una consulta set en dicho OID, se debe indicar los permisos de escritura para los grupos de acceso, esto se logra modificando el archivo de configuración `/etc/snmp/snmpd.conf`. La Figura 3.11 muestra el comando y la respuesta de dicha consulta.

```
pi@raspberrypi:~ $ snmpset -v 2c -c public 192.168.1.26 .1.3.6.1.4.1.1.1.1.1.0 s MQ-7_on
Bad operator (INTEGER): At line 73 in /usr/share/mibs/ietf/SNMPv2-PDU
SNMPv2-SMI::enterprises.1.1.1.1.0 = STRING: "MQ-7_on"
```

Figura 3.11. Consulta snmpset a la MIB

Considerando obtener respuesta de todos los objetos definidos en la MIB, se procede a realizar una consulta snmpwalk al nodo padre de la MIB, la Figura 3.12 muestra el comando y la respuesta de dicha consulta.

```
pi@raspberrypi:~ $ snmpwalk -v 2c -c public 192.168.1.26 .1.3.6.1.4.1.1
Bad operator (INTEGER): At line 73 in /usr/share/mibs/ietf/SNMPv2-PDU
SNMPv2-SMI::enterprises.1.1.1.1.0 = STRING: "MQ-7_on"
SNMPv2-SMI::enterprises.1.1.1.2.0 = STRING: "Sharp GY"
SNMPv2-SMI::enterprises.1.1.2.1.0 = STRING: "67.33333333333333"
SNMPv2-SMI::enterprises.1.1.2.2.0 = STRING: "0.0"
SNMPv2-SMI::enterprises.1.1.3.1.0 = INTEGER: 1
SNMPv2-SMI::enterprises.1.1.3.2.0 = INTEGER: 1
SNMPv2-SMI::enterprises.1.1.4.1.0 = STRING: "2019/02/08 00:01:21"
SNMPv2-SMI::enterprises.1.1.4.2.0 = Gauge32: 3413854913
```

Figura 3.12. Consulta snmpwalk a la MIB

El siguiente paso de verificación es a través de la utilización de un Mib Browser para la comprobación del correcto funcionamiento de las MIB. Se da uso de la versión de prueba de la herramienta MG Soft MibBrowser instalada en un sistema operativo Windows en la misma red del prototipo [49].

La Figura 3.13 muestra la consulta snmpget realizada al objeto sensCO el cual se ubica en el OID numérico 1.3.6.1.4.1.1.1.1.1.

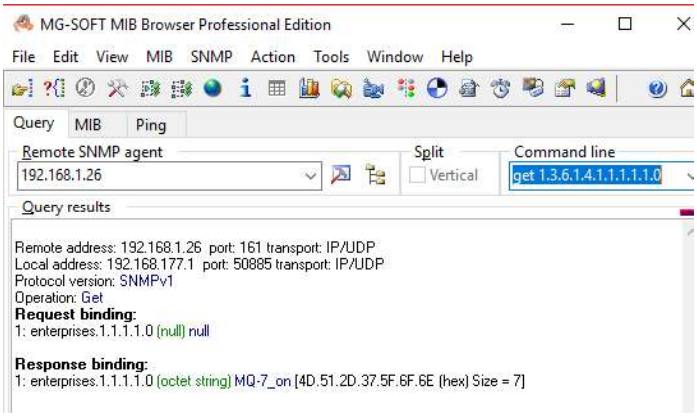


Figura 3.13. Consulta snmpget por MIB Browser

Una vez obtenida la información requerida, se procede con la ejecución de una consulta snmpwalk al OID enterprises, el cual contiene en su primer nodo al OID Yongping, la Figura 3.14 muestra la respuesta obtenida.

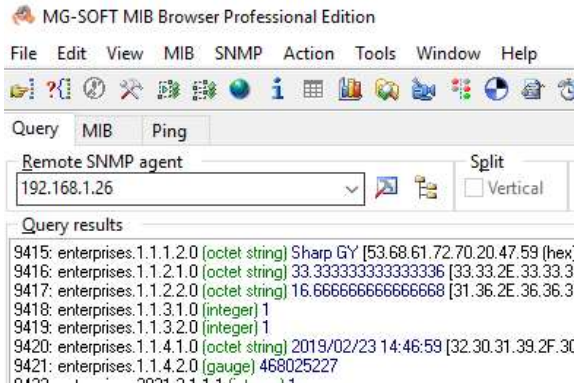


Figura 3.14. Consulta snmpwalk por MIB Browser

Comprobado el funcionamiento de la MIB propia, se procede a verificar las tramas de los mensajes snmp tanto de consulta como de respuesta, esto se logra capturando el tráfico de la red al realizar las consultas.

Finalmente se realiza una captura del tráfico para obtener la trama tanto de la consulta como de la respuesta, la Figura 3.15 muestra una consulta snmpget al objeto que indica el nombre del sensor de monóxido de carbono.

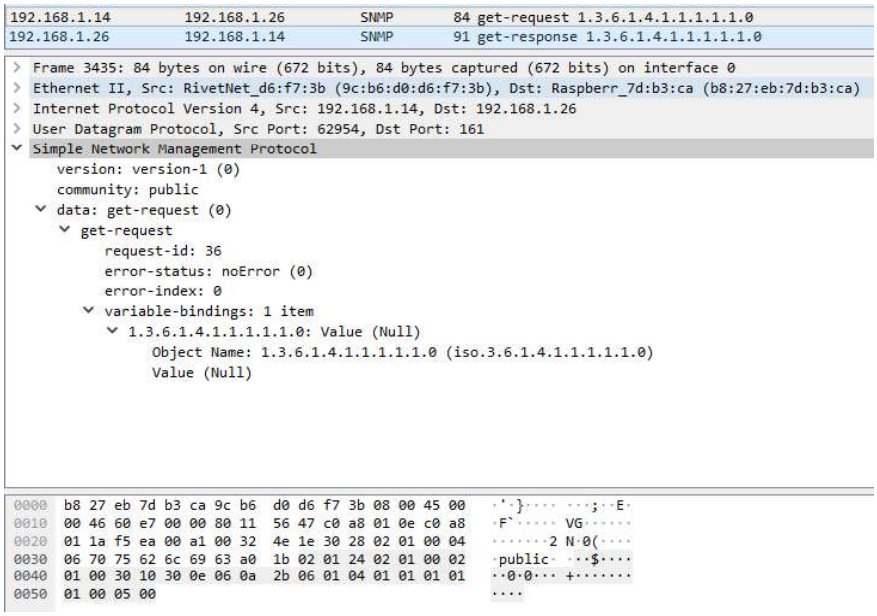


Figura 3.15. Trama snmpget capturada por Wireshark

La trama mostrada indica el nombre del objeto consultado, en este caso el OID numérico, en el valor indica "null", esto debido a que corresponde a la trama de consulta. A continuación, en la Figura 3.16 se muestra la trama de respuesta obtenida.

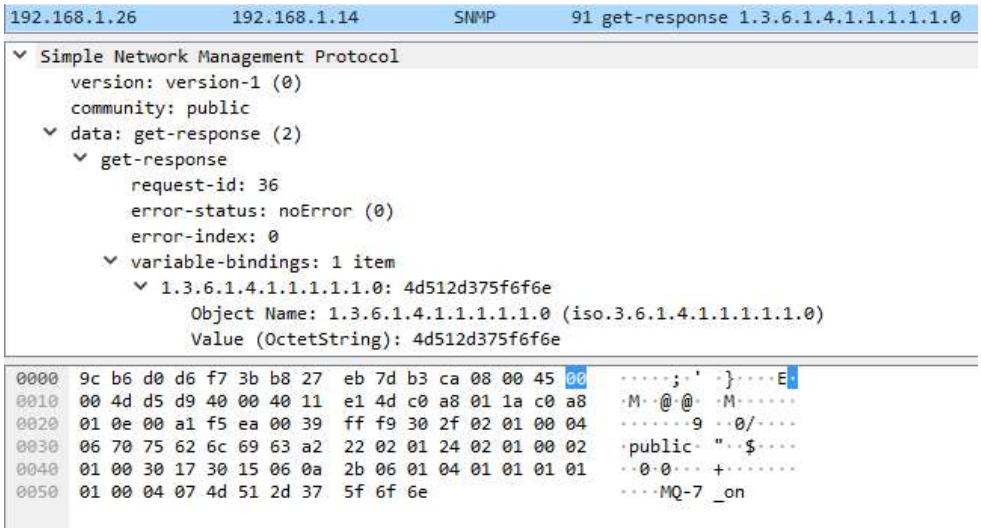


Figura 3.16. Trama get-response capturada por Wireshark

A diferencia de la Figura 3.15, la Figura 3.16 ya tiene un valor devuelto, este corresponde al arreglo de números hexadecimales. A continuación, en la Tabla 3.1 se muestra la conversión de los valores hexadecimales a ASCII.

Tabla 3.1. Conversión Hexadecimal a Ascii

Hex	**4d**	**51**	**2d**	**37**	**5f**	**6f**	**6e**
ascii	M	Q	-	7	_	o	n

3.2.2.2. Funcionamiento agente por RMI

Para comprobar el funcionamiento del agente a través de RMI, se utilizará la herramienta MBean Browser propia del kit de herramientas de WebNMS. Esta herramienta permite establecer conexión con el agente por RMI y desplegar todos los MBeans encontrados, la Figura 3.17 muestra los MBeans correspondientes al nodo valores.

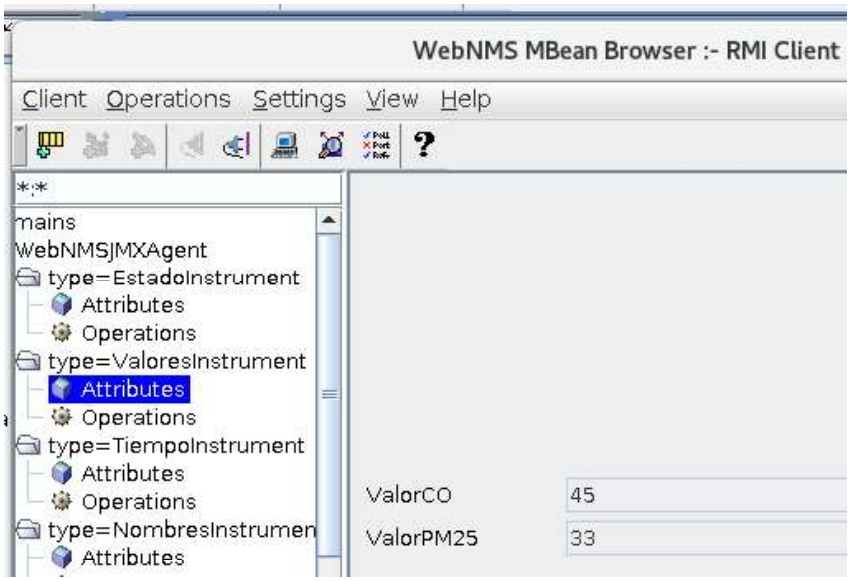


Figura 3.17. Objetos de nodo valores

A través de esta herramienta se puede obtener y colocar nuevos valores, en aquellos objetos que en un inicio fueron definidos como de lectura y escritura.

3.2.3. PRUEBA DE FUNCIONAMIENTO DE MÓDULO DE VISUALIZACIÓN

El módulo 3 comprende la visualización y almacenamiento en base de datos de los valores obtenidos desde los archivos de texto usados por el agente.

3.2.3.1. Llenado de base de datos

Los archivos de texto localizados en el RPi-Agente, que albergan los datos de los parámetros, son utilizados para extraer los valores e insertarlos en la tabla “Lectura” de la base de datos localizada en el RPi-Database, la Figura 3.18 muestra datos de la tabla lectura filtrando los correspondientes al sensor Sharp.

	id_lectura integer	valor numeric (6,2)	fecha timestamp without time zone	id_sensor integer
215	151794	22.12	2019-04-25 11:31:11	2
216	151792	18.12	2019-04-25 11:31:01	2
217	151790	0.10	2019-04-25 11:30:51	2
218	151788	0.10	2019-04-25 11:30:41	2
219	151786	0.10	2019-04-25 11:30:31	2
220	151784	0.10	2019-04-25 11:30:21	2
221	151782	0.10	2019-04-25 11:30:11	2
222	151780	22.39	2019-04-25 11:30:01	2
223	151778	0.10	2019-04-25 11:29:51	2
224	151776	18.40	2019-04-25 11:29:41	2
225	151774	0.10	2019-04-25 11:29:31	2
226	151772	349.61	2019-04-25 11:29:21	2
227	151770	18.67	2019-04-25 11:29:11	2
228	151768	19.64	2019-04-25 11:29:01	2

Figura 3.18. Datos en tabla lectura de Sharp

3.2.3.2. Pantalla Monitor

En la aplicación de escritorio se establece la conexión con el agente a través de RMI, y la forma de comprobar el funcionamiento es comparando la gráfica del sensor mostrada en la Figura 3.19 junto con los valores guardados en la base de datos mostrados en la Figura 3.18.

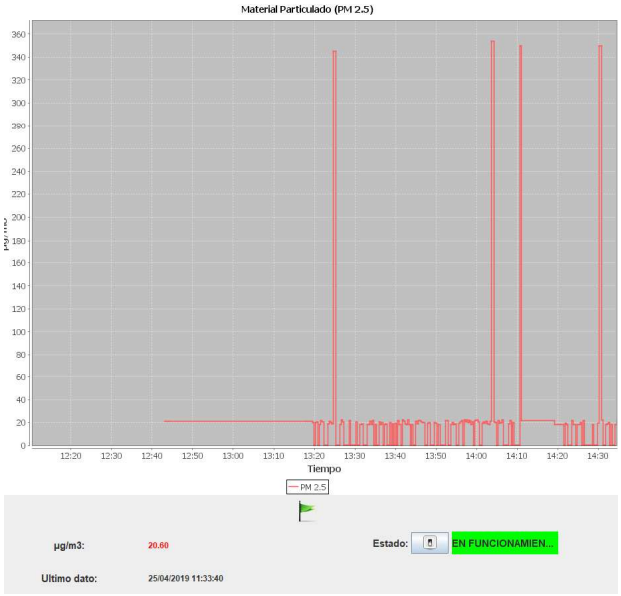


Figura 3.19. Valores PM2.5 graficados

Se puede apreciar que la mayoría de las lecturas fluctúan entre 0 y 20, esto corresponde al tiempo en el que no se presenta actividades de corte de material en la empresa, sin embargo, el sensor logra percibir pequeñas cantidades de material en el aire por lo cual sitúa los valores alrededor de $20\,{}^{ug}/_{m^3}$, y cuando se realizan actividades de corte se presentan lecturas que bordean los $350\,{}^{ug}/_{m^3}$, el cual es un valor que supera los límites recomendados según el IQCA.

Se vio la necesidad de implementar un sistema de notificación dependiendo del rango alcanzado por los sensores, considerando la tabla de índices de calidad ambiental de Quito (IQCA)[2]. Los 3 rangos más altos: amarillo, naranja y rojo representan niveles de alerta en los que empieza la afectación de la salud de individuos sensibles.

La Tabla 3.2 muestra los rangos IQCA, su color y afectación.

Tabla 3.2. Rangos IQCA y significado.

Rangos	Condición para la salud	Color
0-50	Ideal.	Blanco
50-100	Buena.	Verde
100-200	No saludable para personas muy sensibles (enfermos crónicos, adultos mayores, mujeres embarazadas y niños).	Gris
200-300	No saludable para individuos sensibles.	Amarillo
300-400	No saludable para la mayoría de la población y peligrosa para individuos sensibles.	Naranja
400-500	Peligrosa para toda la población.	Rojo

Considerando la concentración graficada en la pantalla, se debe ubicar el rango en el que cae cada lectura, para lo cual se utiliza la Tabla 3.3 para transformar la concentración y obtener el rango de afectación.

Tabla 3.3 Cálculo de índices IQCA desde concentración.

Contaminante	Cálculo de Rango de Concentración.			
CO	$0 < Ci < 10$	$10 < Ci \leq 15$	$15 < Ci \leq 30$	$30 < Ci$
	$IQCA = 10\,Ci$	$IQCA = 20\,Ci - 100$	$IQCA = 6.67\,Ci + 100$	$IQCA = 10\,Ci$
$PM_{2.5}$	$0 < Ci \leq 50$	$50 < Ci \leq 250$	$250 < Ci$	
	$IQCA = 2\,Ci$	$IQCA = Ci + 50$	$IQCA = Ci + 50$	

Ci: *concentración de cada elemento.* (CO: mg/m^3 ; $PM_{2.5}$:ug/m^3)

Se utilizó el ícono visual de una bandera para mostrar el rango actual de cada lectura, la Figura 3.20 muestra un ejemplo cuando una lectura cae en el rango de alerta gris.

Figura 3.20. Bandera indicadora de rango IQCA

La alerta implementada entrega información a través de un correo electrónico dirigida a la persona encargada del control de los niveles de calidad ambiental, a continuación, en la Figura 3.21 se muestra el correo recibido cuando una lectura ingresa en un nivel de alarma.

Figura 3.21. Correo de alerta recibido

Adicional a las notificaciones por email, se realizó la implementación de la notificación por envió de mensajes por la plataforma Whatsapp, para lograr dicho objetivo se utilizó la librería Twilio [51]. Twilio es una plataforma que entrega varias API de distintos lenguajes de programación para el envío de mensajes personalizados. Es un servicio pagado y el mensaje promedio para la región es de 5 ctvs. La Figura 3.22 muestra una captura de pantalla del chat con el número asignado a la cuenta de Twilio.

Figura 3.22. Alerta recibida por Whatsapp

La cuenta utilizada representa una caja de pruebas en la que se verifica el funcionamiento de la aplicación, al ser un paquete de prueba tiene un número limitado de mensajes, para la activación se requiere enviar al número entregado por la plataforma el mensaje “join slightly-brave” y de esta manera la implementación queda activada para el envío de los mensajes personalizados.

3.2.3.1. Pantalla Registro

La ventana utilizada para consulta de valores históricos permite mostrar en valores y gráfico los datos recogidos por cada sensor de manera individual, la Figura 3.23 muestra dicho gráfico correspondiente al sensor MQ-7.

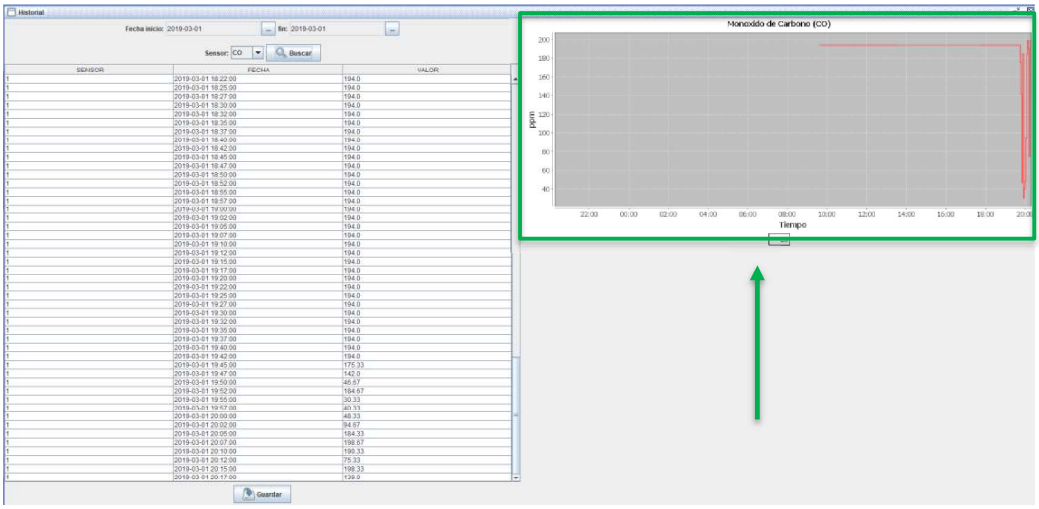


Figura 3.23. Valores MQ-7 según fecha

Obtenido el gráfico también se muestra en una tabla aledaña los valores recuperados de la consulta. La tabla recuperada se muestra en la Figura 3.24.

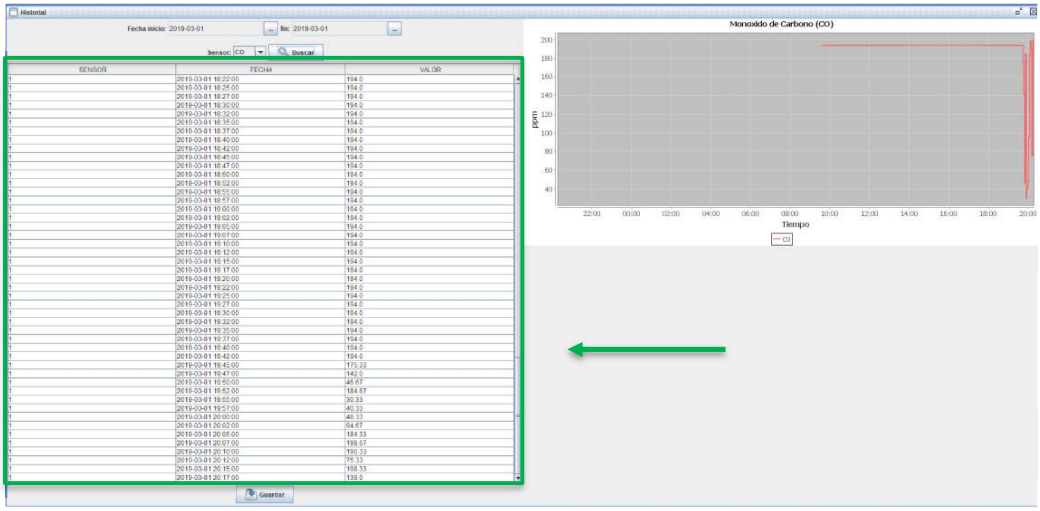
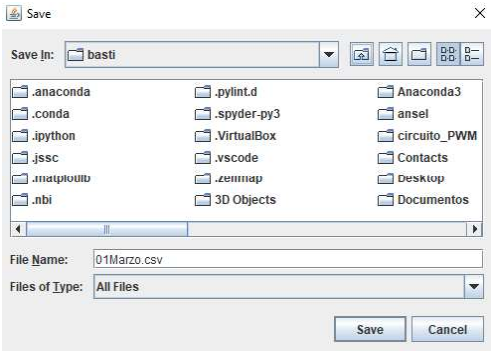


Figura 3.24. Tabla de valores recuperada

Los datos recuperados y presentados en la tabla pueden ser de utilidad para la persona administradora, por tal motivo se implementó la posibilidad de descargar un archivo para su apertura en Excel. La Figura 3.25 muestra el archivo descargado.

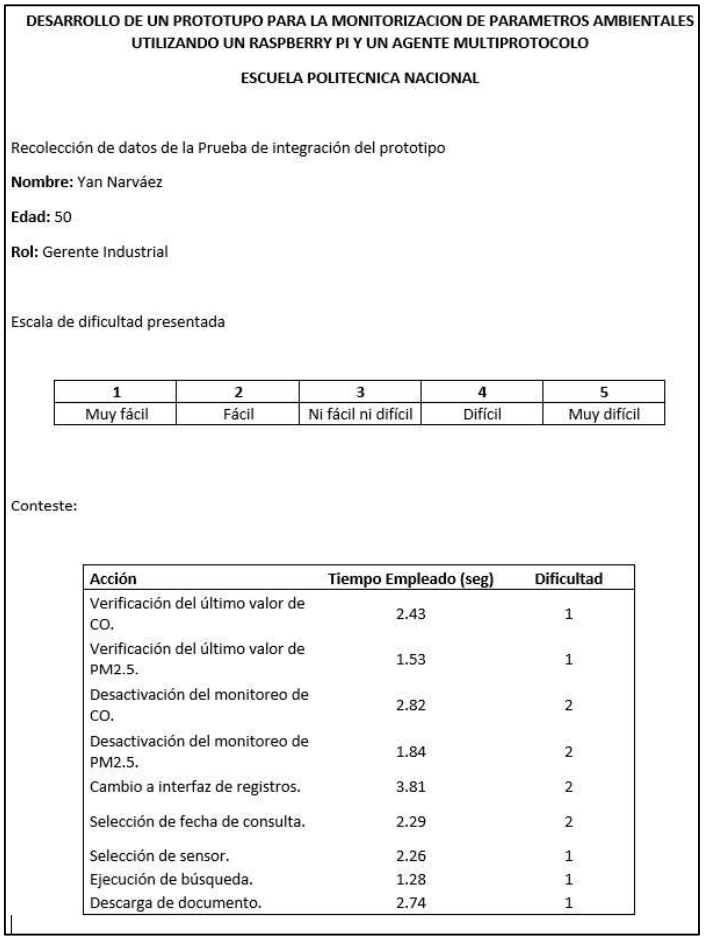


	fecha	sensor	valor	D
142	3/1/2019 19:37	1	194	
143	3/1/2019 19:40	1	194	
144	3/1/2019 19:42	1	194	
145	3/1/2019 19:45	1	175.33	
146	3/1/2019 19:47	1	142	
147	3/1/2019 19:50	1	46.67	
148	3/1/2019 19:52	1	184.67	
149	3/1/2019 19:55	1	30.33	
150	3/1/2019 19:57	1	40.33	
151	3/1/2019 20:00	1	48.33	
152	3/1/2019 20:02	1	94.67	
153	3/1/2019 20:05	1	184.33	
154	3/1/2019 20:07	1	198.67	
155	3/1/2019 20:10	1	190.33	
156	3/1/2019 20:12	1	75.33	
157	3/1/2019 20:15	1	198.33	
158	3/1/2019 20:17	1	139	

Figura 3.25. Archivo descargado de consulta de valores

3.3. PRUEBA DE VALIDACIÓN DE REQUERIMIENTOS

Para corroborar el cumplimiento de los requerimientos establecidos en este proyecto se realizaron encuestas a personas que forman parte del segmento de usuarios del prototipo. En primer lugar, se presenta en la Figura 3.26 el formato empleado para la encuesta de utilización de la aplicación de escritorio que indica el grado de dificultad de cada acción seguida del tiempo aproximado empleado en cada una.



DESARROLLO DE UN PROTOTUPO PARA LA MONITORIZACION DE PARAMETROS AMBIENTALES UTILIZANDO UN RASPBERRY PI Y UN AGENTE MULTIPROTOCOLO

ESCUELA POLITECNICA NACIONAL

Recolección de datos de la Prueba de integración del prototipo

Nombre: Yan Narváez

Edad: 50

Rol: Gerente Industrial

Escala de dificultad presentada

1	2	3	4	5
Muy fácil	Fácil	Ni fácil ni difícil	Difícil	Muy difícil

Conteste:

Acción	Tiempo Empleado (seg)	Dificultad
Verificación del último valor de CO.	2.43	1
Verificación del último valor de PM2.5.	1.53	1
Desactivación del monitoreo de CO.	2.82	2
Desactivación del monitoreo de PM2.5.	1.84	2
Cambio a interfaz de registros.	3.81	2
Selección de fecha de consulta.	2.29	2
Selección de sensor.	2.26	1
Ejecución de búsqueda.	1.28	1
Descarga de documento.	2.74	1

Figura 3.26. Encuesta de utilización de la aplicación de escritorio.

El segmento de usuarios del prototipo puede ser o no del área técnica de una empresa, por lo cual, la aplicación de escritorio debe ser intuitiva, y por tal motivo se evalúa la dificultad al realizar las acciones. Esta prueba se realizó a 10 personas, las pruebas se encuentran en el ANEXO I.

Los resultados obtenidos correspondientes a los tiempos de ejecución promedio de las acciones sobre la aplicación de escritorio se presentan en la Tabla 3.4, en esta se puede observar que la acción en la que se emplea más tiempo es en acceder a la interfaz de registros de históricos, y que es ésta la única acción cuya dificultad se cataloga como Fácil.

Tabla 3.4 Tiempos promedio de acciones en aplicación de escritorio.

	Promedio Tiempo	Promedio Dificultad
Verificación del último valor de CO.	1.751	1
Verificación del último valor de PM2.5.	1.353	1
Desactivación del monitoreo de CO.	2.781	1.8
Desactivación del monitoreo de PM2.5.	1.704	1.7
Cambio a interfaz de registros.	3.054	2.1
Selección de fecha de consulta.	2.019	1.9
Selección de sensor.	1.666	1.1
Ejecución de búsqueda.	1.413	1
Descarga de documento.	1.892	1

4. CONCLUSIONES Y RECOMENDACIONES

En la presente sección se presentan las conclusiones y las recomendaciones obtenidas al desarrollar el presente proyecto, estas fueron definidas siguiendo el flujo del manejo de la información del proyecto.

4.1. CONCLUSIONES

- Varias han sido las ocasiones en el Ecuador que se han producido muertes trágicas por intoxicación por gases, por situaciones desafortunadas como dormir cerca de carbón o dentro de un auto prendido para obtener calor en la noche, o situaciones peligrosas como ingresar y permanecer en cloacas en las que no exista una adecuada ventilación o circulación de aire. Es necesario poseer alternativas de fácil acceso para la población y así poder prevenir estas lamentables pérdidas, y si estas alternativas además no presentan dificultades para el monitoreo, su factibilidad de implementación se incrementa.
- El objetivo general de este Trabajo de Titulación es desarrollar un prototipo para la monitorización de parámetros ambientales utilizando un Raspberry Pi y un agente multiprotocolo, mismo que ha sido realizado con satisfacción, el prototipo realizado permite visualizar valores obtenidos por sensores mediante una aplicación de escritorio que utiliza la tecnología RMI para la conexión con un agente situado en un Raspberry Pi que a su vez es el encargado de entregar la información recogida por sensores conectados a un Arduino Uno, este agente puede entregar la información a través de conexiones RMI o consultas SNMP.
- Mediante el análisis realizado de los conceptos claves para el desarrollo del prototipo se concluye que la arquitectura JMX es el pilar fundamental para la generación del agente multiprotocolo, y que el Raspberry Pi presenta recursos suficientes para albergar a dicho agente. El agente generado con la plataforma WebNMS permite seleccionar los protocolos compatibles y entregar la información requerida en simultaneo desde la aplicación de monitoreo a través de JMX o por consola a través de consultas SNMP. El limitante de este proyecto es la adquisición de la licencia de la plataforma WebNMS pues el costo de la licencia depende del número de agentes por desarrollar. Un punto factible es la implementación de un agente multiprotocolo utilizando las librerías correspondientes a la tecnología JMX y se debe considerar que la precisión de la lectura depende de las características de los sensores por utilizar.
- Se concluyó que el lenguaje de programación Python posee mucha versatilidad pues para cada área de desarrollo se tiene acceso a librerías especializadas, y que el lenguaje

bash de Linux siempre puede ser utilizado para programación de tareas en las que el control del tiempo de ejecución es importante.

- El prototipo actúa sobre diferentes campos de desarrollo, se considera la utilización de hardware con funciones específicas como son los módulos de sensores, unidades de almacenamiento, así como plataformas de desarrollo, y la utilización de IDE de desarrollo para la codificación. Por lo tanto, para lograr considerar todos los requisitos y lograr el objetivo principal se realizó un diagrama de despliegue, el mismo que permitió abstraer cada elemento o parte fundamental de cada nodo o entidad principal, y poder relacionarlos adecuadamente. Además, dependiendo de la etapa del proyecto, se realizaron también de acuerdo con las necesidades: diagramas de flujo, diagrama relacional y diagrama de clases.
- El prototipo desarrollado se centró en 3 áreas que constituyen los componentes principales del proyecto, el primer módulo corresponde a la adquisición de valores y su almacenamiento, el siguiente corresponde a la gestión y exposición de estos valores a través del agente multiprotocolo situado en un Raspberry Pi, y el tercer módulo corresponde al almacenamiento y a la visualización de estos valores mediante el acceso de una aplicación de escritorio al agente para obtener valores actuales utilizando la arquitectura JMX así como también a valores históricos consultados a una base de datos localizada en otro Raspberry Pi, esta aplicación también entrega notificaciones de valores que sobrepasaron los umbrales definidos a través de correo electrónico y a través de mensajes de WhatsApp a un número previamente definido, así como la obtención del registro consultado en formato CSV.
- Durante la codificación de la interfaz gráfica de la aplicación de escritorio se priorizó la facilidad de uso del usuario final, la adquisición y graficación de los valores de los sensores son controlados mediante un timer que cada determinado tiempo efectúa una nueva impresión, a través de la ventana principal además del gráfico de cada sensor, se puede ejecutar un cambio en el “estado” actual del sensor, pues se consideró que un sensor puede estar en un período de mantenimiento y que durante ese tiempo no es necesario su graficación. Este estado también posee su propio objeto en la MIB de la empresa y puede ser consultado mediante SNMP al agente multiprotocolo. Cualquier acción realizada sobre los objetos de la MIB que tienen permisos de escritura son reflejados a través de cualquier vía de comunicación disponible del agente.
- El término material particulado fino hace referencia al conjunto de elementos presentes en cualquier área en la que una persona se encuentre, pues únicamente define el diámetro de las partículas en el aire, basta que cualquier partícula entre en ese rango y que la

concentración sobrepase los umbrales definidos para que el riesgo sobre la salud de las personas se incremente.

- Uno de los procesos por seguir para la calibración de sensores es someter al sensor a concentraciones definidas del parámetro medido, en la provincia de Pichincha puede realizarse este proceso en la Secretaría del Ambiente de Quito. Otra opción es realizar la calibración comparando los valores arrojados por un sensor junto con un dispositivo de monitoreo dedicado, relacionar ambas muestras y obtener una ecuación de tendencia para la transformación. Finalmente, la opción óptima es adquirir sensores que no necesiten de una calibración.
- Los sensores empleados son de bajo costo por lo tanto también de baja precisión, al calibrarlos junto a los equipos de monitoreo dedicados se pudo mejorar las mediciones de estos, sin embargo, los rangos de detección más altos o muy bajos no pudieron ser alcanzados por las características de los sensores y del monitor pues este no marcaba valores superiores.
- El archivo crontab te permite ejecutar periódicamente tareas como scripts, sin embargo, si se requiere un desfase de segundos entre ejecuciones, se puede optar por ingresar lazos de condición en los scripts a ejecutar para cumplir con el objetivo. Para la ejecución de scripts considerando el inicio de Raspbian puede utilizarse el script `/etc/rc.local`, en este basta con definir el script o comando por ejecutar en él para su ejecución automática.
- Existe una extensa cantidad de librerías en lenguaje Java para la implementación de proyectos en los que se desea proveer de funcionalidades específicas, como por ejemplo la librería JFreeChart para la graficación, la librería javax.mail para el envío o recepción de correo, y otras librerías pagas pero muy útiles como Twilio la cuál, permite enviar mensajes por Whatsapp por $0.0042 dólares.
- Considerando las redes de sensores, el RPi contenedor del agente puede ser utilizado como nodo coordinador para gestionar la información proveniente de varios nodos recopiladores, los cuales pueden contener a los sensores, de esta manera el área de cobertura puede incrementarse, toda la gestión de la información se centraría en el RPi a través del agente. Cabe indicar que el área de aplicación de este proyecto es muy amplia, no se encuentra limitado a los parámetros ambientales descritos ni al área industrial, este puede ser utilizado en cualquier escenario que lo requiera.
- El RPi además de todos los puertos de comunicación serial que dispone, trae incorporado Bluetooth, esto le otorga más opciones para la recepción de datos por parte de otros dispositivos además del Arduino, así como la opción de incorporar adaptadores como es el Raspbee el cual le permite comunicarse con dispositivos Zigbee. Todas estas

variantes son fuentes de datos cuya información puede también ser gestionada por parte del agente multiprotocolo.

- El presente proyecto puede ser complementado utilizando diversas plataformas para la entrega de la información, por ejemplo, la utilización de aplicaciones móviles que puedan acceder a la información de la base de datos y que permitan entregar a los interesados las alertas directamente en su celular, o si se dispone de asistentes virtuales como lo es Alexa, el cuál viene incorporado en los parlantes Echo de Amazon, puede implementarse un ”Skill” o habilidad nueva, que permita acceder a los mismos datos y entregar dicha información mediante comandos de voz.

4.2. RECOMENDACIONES

- La utilización del relé para la conmutación de los voltajes posee un tiempo de vida útil, usualmente considerando una frecuencia baja de conmutaciones la vida del componente se estima en diez millones de conmutaciones, sin embargo, cuando la frecuencia de conmutaciones se eleva, la vida del componente se reduce a un millón de conmutaciones. Un componente de reemplazo es un transistor por cada línea de voltaje que alimente al sensor MQ-7.
- El rango de detección de los sensores empleados es muy bajo e impreciso, si se considera el área de aplicación del proyecto es recomendable el uso de sensores con un rango de detección más alto para una monitorización más acertada, de igual manera que estos sensores sean calibrados de fábrica para mejorar la precisión de las lecturas.
- Al codificar el script a colocar en el Arduino se recomienda obtener el tiempo que cada función y declaración se demora en ejecutarse, esto puede obtenerse utilizando la función por defecto “`micros()`”, la cual entrega la cantidad de microsegundos desde que inició el script. Puede utilizarse 2 veces esta función entre inicio y fin de una declaración o función, para después al restar ambos resultados encontrar el tiempo que toma en ejecutarse. Este mismo concepto es aplicable en scripts de bash cuando se desea manejar tiempos exactos de espera.
- Cuando se codifique los métodos extras en el esqueleto del agente JMX, se recomienda revisar la versión de java disponible con el IDE del agente, pues con versiones antiguas se presentan más limitantes con la manipulación de datos y al momento de probar las funciones empleadas podría darse varios errores.
- La licencia del paquete WebNMS dura 45 días, cuando culmina se restringe el uso de cualquier herramienta. Se recomienda reinstalar el paquete y generar un nuevo proyecto

y espacio de trabajo desde cero pues al copiar los archivos modificados se generan errores de registro de licencia y el agente no inicia adecuadamente.

- Es importante utilizar los archivos MIB creados por la herramienta MIBCreator del paquete WebNMS pues cuando se utiliza archivos generados con cualquier editor de texto, se detectan errores por espacios mal ubicados.
- La detección de valores elevados de los parámetros analizados es característica solamente a la aplicación de escritorio, una mejora importante es la implementación de umbrales de detección para generación de traps correspondiente al adaptador SNMP.
- Cuando se trabaja con scripts es importante trabajar con logs en los que se direccione no solo problemas sino pruebas de un buen funcionamiento del programa, de esta manera se tiene un registro exacto de cuál es la función que no trabaja bien.
- Una mejora importante para el presente prototipo es el incremento de puntos de detección para amplificar el área de impacto, una opción es utilizar módulos Arduino Nano o similares para reducir el costo, teniendo varios puntos de detección se podría enviar los datos a un punto de recolección para procesarlos, este punto puede ser el mismo agente o un dispositivo intermedio.
- Una limitante del prototipo es la aplicación de escritorio pues obliga a ser monitoreado desde un escritorio de trabajo, esta limitante puede ser solventada con la implementación de una aplicación web que permita acceder desde ubicaciones remotas.
- Otra de las limitantes a considerar es que los Raspberry necesitan tener conexión a una red con salida al internet para cumplir con la entrega de las notificaciones por WhatsApp o correo.
- Un componente físico que no dispone el Raspberry Pi es un reloj de tiempo real, esto afecta en que si únicamente se conecta a una red sin salida a internet el tiempo no se actualiza, esto puede ser solventado si es que es conectado a un servidor NTP en la red local, si a través de la red local tenga acceso al internet o también puede solventarse si se conecta un componente RTC al RPi.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] El Comercio, «Dos policías aparecen muertos en un patrullero en el norte de Quito,» [En línea]. Available: http://www.elcomercio.com/actualidad/policias-muertos-interior-patrullero-quito.html. [Último acceso: 01 Marzo 2019].

[2] Secretaría del medio ambiente de Quito, «Documento IQCA,» [En línea]. Available: http://www.quitoambiente.gob.ec/ambiente/images/Secretaria_Ambiente/red_monitoreo/informacion/iqca.pdf. [Último acceso: 01 Noviembre 2018].

[3] The Raspberry Pi Foundation, «Review 2017,» [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Último acceso: 01 Noviembre 2018].

[4] The Raspberry Pi foundation, «Modelo 3B+,» [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Último acceso: 01 Noviembre 2018].

[5] Elinux, «Distribuciones Linux para RPi,» [En línea]. Available: https://elinux.org/RPi_Distributions. [Último acceso: 10 Noviembre 2018].

[6] Ubuntu-Mate, «Ubuntu Mate for the Raspberry Pi,» [En línea]. Available: https://ubuntu-mate.org/raspberry-pi/. [Último acceso: 01 Noviembre 2018].

[7] Raspbian, «Raspbian main page,» [En línea]. Available: https://raspbian.org/. [Último acceso: 01 Noviembre 2018].

[8] The Raspberry Pi foundation, «GPIO usage,» [En línea]. Available: https://www.raspberrypi.org/documentation/usage/gpio/. [Último acceso: 01 Noviembre 2018].

[9] «I2C Bus, Interface and Protocol,» [En línea]. Available: http://i2c.info/. [Último acceso: 10 Diciembre 2018].

[10] Sparkfun, «Serial Peripheral Interface (SPI),» [En línea]. Available: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi. [Último acceso: 01 Noviembre 2018].

[11] Stanford, «UART Basics,» [En línea]. Available: https://web.stanford.edu/class/cs140e/notes/lec4/uart-basics.pdf. [Último acceso: 01 Noviembre 2018].

[12] MIT, «Scratch,» [En línea]. Available: https://scratch.mit.edu/. [Último acceso: 01 Noviembre 2018].

[13] Java, «Java about,» [En línea]. Available: https://www.java.com/es/about/. [Último acceso: 01 Noviembre 2018].

[14] Python Software Foundation, «Python about,» [En línea]. Available: https://www.python.org/about/. [Último acceso: 17 Noviembre 2018].

[15] arduino.cc, «Arduino about,» [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction. [Último acceso: 20 Noviembre 2018].

[16] arduino.cc, «Arduino Uno Rev3,» [En línea]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Último acceso: 20 Noviembre 2018].

[17] IETF, «RFC 1157: SNMP,» [En línea]. Available: https://tools.ietf.org/html/rfc1157. [Último acceso: 27 Noviembre 2018].

[18] «Essential SNMP,» de *Essential SNMP 2nd Edition*, O´Reilly, 2005, pp. 19-75.

[19] Wikipedia, «ASN.1,» [En línea]. Available: https://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One. [Último acceso: 23 Noviembre 2018].

[20] IETF, «RFC 1156: MIB,» [En línea]. Available: https://tools.ietf.org/html/rfc1156. [Último acceso: 27 Noviembre 2018].

[21] ITU-T, «BER,» [En línea]. Available: https://www.itu.int/rec/T-REC-X.690-201508-I/en. [Último acceso: 02 Diciembre 2018].

[22] Oracle, «Especificación JMX,» [En línea]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/jmx/JMX_1_4_specification.pdf. [Último acceso: 05 Diciembre 2018].

[23] Oracle, «RMI Overview,» [En línea]. Available: https://docs.oracle.com/javase/tutorial/rmi/overview.html. [Último acceso: 10 Diciembre 2018].

[24] ireasoning, «Plataforma iReasoining,» [En línea]. Available: http://www.ireasoning.com/ . [Último acceso: 10 Diciembre 2018].

[25] Zoho Corporation, «WebNMS,» [En línea]. Available: https://www.webnms.com/javaagent/help/mp_agent/index.html.. [Último acceso: 10 Diciembre 2018].

[26] DB-Engines, «DBMS,» [En línea]. Available: https://db-engines.com/en/article/Database+Management+System. [Último acceso: 10 Diciembre 2018].

[27] DB-Engines, «Databases,» [En línea]. Available: https://db-engines.com/en/article/Database. [Último acceso: 10 Diciembre 2018].

[28] DB-Engines, «Ranking,» [En línea]. Available: https://db-engines.com/en/ranking. [Último acceso: 10 Diciembre 2018].

[29] DB-Engines, «Ranking Definition,» [En línea]. Available: https://db-engines.com/en/ranking_definition. [Último acceso: 10 Diciembre 2018].

[30] DB-Engines, «PostgreSQL vs MySQL,» [En línea]. Available: https://db-engines.com/en/system/MySQL%3BOracle%3BPostgreSQL. [Último acceso: 11 Diciembre 2018].

[31] mysql, «Workbench,» [En línea]. Available: https://www.mysql.com/products/workbench/. [Último acceso: 12 Diciembre 2018].

[32] The pgadmin Development Team, «pgadmin,» [En línea]. Available: https://www.pgadmin.org/features/. [Último acceso: 12 Diciembre 2018].

[33] archilinux, «Daemons,» [En línea]. Available: https://wiki.archlinux.org/index.php/Daemons_(Espa%C3%B1ol). [Último acceso: 13 Diciembre 2018].

[34] SparxSystems, «UML Deployment,» [En línea]. Available: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html. [Último acceso: 15 Diciembre 2018].

[35] Kanbantool, «Metodología Kanban,» [En línea]. Available: https://kanbantool.com/es/metodologia-kanban. [Último acceso: 14 Diciembre 2018].

[36] Sparkfun, «MQ-7 Datasheet,» [En línea]. Available: https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf. [Último acceso: 01 Noviembre 2018].

[37] Amazon, «Sensor MQ-7 image,» [En línea]. Available: https://www.amazon.com/Solu-Monoxide-Detection-Arduino-Compatible/dp/B00UYSOUL8. [Último acceso: 01 Noviembre 2018].

[38] dfrobot, «Sharp GP2Y,» [En línea]. Available: http://image.dfrobot.com/image/data/SEN0144/gp2y1010au_e.pdf. [Último acceso: 01 Noviembre 2018].

[39] Amazon, «Sharp GP2Y imágen,» [En línea]. Available: https://www.amazon.es/DIYmall-GP2Y1014AU0F-GP2Y1010AU0F-condensador-resistencia/dp/B01823APBK. [Último acceso: 10 Diciembre 2018].

[40] The Spyder Website Contributors, «IDE Spyder,» [En línea]. Available: : https://www.spyder-ide.org/. [Último acceso: 01 Noviembre 2018].

[41] Netbeans, «IDE Netbeans,» [En línea]. Available: https://netbeans.org/. [Último acceso: 01 Noviembre 2018].

[42] JFree, «JFreeChart,» [En línea]. Available: http://www.jfree.org/jfreechart/. [Último acceso: 15 Diciembre 2018].

[43] Balena, «Software Etcher,» [En línea]. Available: https://www.balena.io/etcher/. [Último acceso: 01 Noviembre 2018].

[44] NetSarang Computer, «Xmanager Power Siute,» [En línea]. Available: https://www.netsarang.com/en/. [Último acceso: 01 Diciembre 2018].

[45] Kidde, «Kidde: Monitor CO,» [En línea]. Available: https://www.kidde.com/home-safety/en/us/products/fire-safety/co-alarms/kn-copp-b-lpm/. [Último acceso: 15 Diciembre 2018].

[46] Amazon, «Particle Detector P600,» [En línea]. Available: https://www.amazon.com/Temtop-Particle-Detector-Professional-Accurate/dp/B0787Z5DK9. [Último acceso: 20 Diciembre 2018].

[47] SharpSensorUser, «Sharp GP2Y Dust Density,» [En línea]. Available: https://github.com/sharpsensoruser/sharp-sensor-demos/blob/master/docs/Sharp%20GP2Y1010AU0F%20-%20Dust%20Density%20Conversion.pdf. [Último acceso: 15 Diciembre 2018].

[48] Net-SNMP, «Agente net-snmp,» [En línea]. Available: http://www.net-snmp.org/. [Último acceso: 21 Diciembre 2018].

[49] MG-SOFT, «MIB Browser,» [En línea]. Available: https://www.mg-soft.com/download.html?product=mibbrowser. [Último acceso: 27 Diciembre 2018].

[50] IETF, «RFC 2580: SMI v2,» [En línea]. Available: https://tools.ietf.org/html/rfc2580. [Último acceso: 30 Noviembre 2018].

[51] Página Oficial Raspberry Pi, "Raspberry Pi Revisión 2017", [En línea]. Available: https://static.raspberrypi.org/files/about/RaspberryPiFoundationReview2017.pdf. [Último acceso: 1-Nov-2018].

ANEXOS

Los anexos se encuentran en el CD adjunto al presente documento.

ANEXO A. Respuestas de encuesta para obtención de requerimientos del prototipo y respuestas de encuesta de utilización de aplicación de escritorio.

ANEXO B. Código de los scripts y MIB.

ANEXO C. Código esqueleto del agente JMX y código extra implementado.

ANEXO D. Código de aplicación de escritorio y script de base de datos.

ANEXO E. Hojas de datos de sensores y elementos utilizados.

ANEXO F. Diagramas de circuitos implementados.

ORDEN DE EMPASTADO