UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Carrera de Electrónica y Telecomunicaciones

Sistema de Gestión de Dispositivos IoT enfocados en Escenarios de Smart Cities

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones.

Autores:

Renato Sebastián Alvarado Illescas
C.I: 010599979-1
renato.alvarado55@gmail.com

Edisson Javier Muñoz Quizhpi
C.I: 010642485-6
edisoniitss@gmail.com

Director:

Ing. Santiago Renán González Martínez, PhD.
C.I: 010389593-4

Cuenca, 05 octubre de 2020

Resumen

En este documento se presenta el desarrollo de una aplicación para la gestión y monitorización remota de nodos con capacidad para integrarse en escenarios diseñados para *Smart Cities*. En concreto, la aplicación permite monitorear variables tales como la carga de la CPU, el uso de la memoria, la temperatura del dispositivo y el consumo de energía. En cuanto al dispositivo a gestionar, este consiste en una plataforma Raspberry Pi 4 con el sistema operativo Raspberry Pi OS 1.4, sobre la cual se instalaron sensores para la adquisición de datos de geolocalización y variables ambientales como contaminación, temperatura y luz, así como el uso de una batería para energizar el nodo.

En la tarea de gestión de los nodos se empleó el protocolo SNMPv3 (Simple Network Management Protocol), de esta manera el usuario tiene la posibilidad de optimizar los recursos del sistema (sensores y consumo energético).

La aplicación se desarrolló utilizando el *framework* Node-RED, en un ordenador con el sistema operativo Ubuntu 20.04. Esta aplicación cuenta con una interfaz de usuario y la capacidad de conectarse a la nube de IBM para su monitoreo desde Internet.

La conexión entre los nodos se realizó mediante una red Ad Hoc multi-salto. Adicionalmente, en el ordenador se configuró un *gateway* por defecto que permite la conexión de los nodos Ad Hoc hacia Internet.

Con la finalidad de emular una red tipo Ad Hoc multi-salto con cuatro nodos, se utilizó la herramienta NS3, la misma que permite adecuar el escenario a uno real al realizar las simulaciones utilizando hardware sobre contenedores Linux. De esta forma fue posible inyectar tráfico SNMP real en la red simulada.

Para la evaluación de la red se empleó un escenario multi-salto conformado por cuatro nodos fijos. Se realizaron experimentos empleando enrutamiento estático y dinámico mediante OLSR (Optimized Link Route State). Y finalmente se planteó un experimento adicional considerando un quinto nodo con capacidad de desplazamiento.

En la red emulada se realizó el análisis de métricas de red como: el tráfico, el porcentaje de recepción de paquetes, el retardo de la red y por último se midió la capacidad de ancho de banda.

Palabras Clave: SNMP. CPU. Gateway. Delay. Throughput. Smart City. Framework. Ad Hoc. NS3. Node-RED.

Abstract

In this document the development of an application for the remote management and monitoring, is presented, with capabilities to integrate into scenarios designed for Smart Cities. Specifically, the application enables monitoring variables as: CPU load, memory usage, device temperature and power consumption. On a Raspberry Pi 4 with Raspberry Pi OS 1.4 as operating system, the installation of environmental and geolocation sensors as: pollution, temperature and light, furthermore, the use of a battery to power the node.

In the node management task, the SNMPv3 (Simple Network Management Protocol) is implemented for resource management of the node. Thus, the user has the possibility to optimize system resources (sensors and power consumption).

The application is developed in the framework Node-RED in a computer with the operating system Ubuntu 20.04. This application has a user interface and the capacity to connect to the IBM cloud for monitoring from the internet.

The connection between the nodes is made through an Ad Hoc network, in the computer a default gateway is configured that allows the network connection to the internet from the Raspberry Pi.

In order to emulate a multi-hop Ad Hoc network with four nodes, the NS3 tool is used, which allows to adapt the scenario to a real one by performing simulations using hardware on Linux containers, through real SNMP traffic is injected into the network in static and mobile scenarios, using static routing and the OLSR (Optimized Link Route State) protocol. Finally, an additional experiment was proposed considering a fifth node with displacement capacity.

In the emulated network, the analysis of network metrics such as: throughput, percentage of packet reception, delay and the network bandwidth capacity are measured.

Keywords: SNMP. CPU. Gateway. Delay. Throughput. Smart City. Framework. Ad Hoc. NS3. Node-RED.

Índice general

Índice de figuras

Índice de tablas

Cláusula de Propiedad Intelectual

Renato Sebastián Alvarado Illescas, autor/a del trabajo de titulación "Sistema de Gestión de Dispositivos IoT enfocados en Escenarios de Smart Cities", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

Cuenca, 05 de octubre de 2020

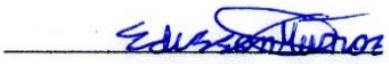
Renato Sebastián Alvarado Illescas

C.I: 010599979-1

Cláusula de Propiedad Intelectual

Edisson Javier Muñoz Quizhpi, autor/a del trabajo de titulación "Sistema de Gestión de Dispositivos IoT enfocados en Escenarios de Smart Cities", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

Cuenca, 05 de octubre de 2020



Edisson Javier Muñoz Quizhpi

C.I: 010642485-6

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Renato Sebastián Alvarado Illescas en calidad de autor/a y titular de los derechos *morales y patrimoniales del trabajo de titulación* "Sistema de Gestión de Dispositivos IoT enfocados en Escenarios de Smart Cities", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 05 de octubre de 2020

Renato Sebastián Alvarado Illescas

C.I: 010599979-1

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Edisson Javier Muñoz Quizhpi en calidad de autor/a y titular de los derechos morales y patrimoniales del trabajo de titulación "Sistema de Gestión de Dispositivos IoT enfocados en Escenarios de Smart Cities", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 05 de octubre de 2020

Edisson Javier Muñoz Quizhpi

C.I: 010642485-6

Dedicatoria

A mis padres y abuelos, que a lo largo de estos años me brindaron su apoyo incondicional y con su guía han logrado que pueda realizar mis estudios.

A mis hermanas Cristina, Paula e Isabella, que siempre han sido la motivación para seguir adelante.

A Estefanía, por su amor, compañía, apoyo en los momentos difíciles de la carrera y la motivación para superarme.

A mi familia y amigos, por el apoyo de cada día y por estar conmigo en todo momento.

Renato Alvarado Illescas

Dedicatoria

A mis padres Franco y Nora, por todo su amor, trabajo, sacrificio y apoyo que me han brindado en todos estos años y ya que son la mayor motivación de mi vida. Siempre me han dado la fuerza para seguir adelante y nunca rendirme.

A mi hermano Sebastián, por estar siempre presente las veces que lo necesite, además de brindarme apoyo en todos los momentos posibles.

A mi tía Genoveva y mi abuela Rosa, ya que sin ellas nada de esto hubiera sido posible.

A María José, quien me apoyo y siempre me alentó a seguir adelante incluso en los momentos en los que todo parecía perdido y es una persona especial en mi vida.

A todos mis demás familiares, que de una u otra forma me ayudaron en algún momento de mi vida para concluir este proceso.

Edisson Muñoz Quizhpi

Agradecimientos

Al Ing. Santiago González, nuestro director del trabajo de titulación y maestro a lo largo de la carrera, por su guía y dedicación a este trabajo, así como sus consejos y conocimientos brindados.

A Edisson, un gran amigo y compañero en este trabajo, gracias a quién hemos podido culminar exitosamente.

A mis compañeros y maestros, por su apoyo durante la carrera y en el día a día. Por las enseñanzas recibidas que serán la base de la vida profesional.

Renato Alvarado Illescas

Agradecimientos

A mis padres y familiares, por estar siempre al pendiente de mí y brindarme muchas lecciones a lo largo de mi vida, además de tener mucha paciencia conmigo.

A mis maestros, por los conocimientos que me han otorgado, en especial al **Ingeniero Santiago González,** por estar siempre atento y disponible para la consulta de inquietudes, y por toda la ayuda brindada en la elaboración de este trabajo.

A Renato, por aportar y compartir todo su conocimiento en este trabajo, pero sobre todo ser un buen compañero y un gran amigo.

A mis compañeros y amigos, que me ayudaron en la etapa de vida universitaria ya sea por medio de un consejo o motivación, pero especialmente a María José por haber sido comprensiva y paciente, pero en especial por haber compartido muchos momentos tanto buenos como malos y apoyarme siempre en cada decisión.

Edisson Muñoz Quizhpi

Abreviaciones y Acrónimos

6

6LoWPAN
IPv6 over Low-Power Wireless Personal Area Network, 47

A

AES
Advanced Encryption Standard, 51, 52, 124
ANS.1
Abstract Syntax Notation One, 33, 34, 35
AODV
Ad Hoc On-Demand Distance Vector, 42
API
Application Programming Interface, 23, 46, 152

B

BER
Basic Encoding Rules, 34

C

CoAP
Constrained Application Protocol, 47
CoMI
CoAP Management Interface, 47
CPU
Central Processing Unit, 2, 3, 9, 12, 24, 49, 65, 70, 80, 81, 82, 85

D

DARPA
US Defense Advanced Research Projects Agency, 42
DDS
Data Distribution Service, 47
DES
Data Encryption Standard, 38, 51
DoS
Denial of Service, 43
DSR
Dynamic Source Routing, 42

E

eCO2
Carbon Dioxide Equivalent, 65, 68, 121

F

FCAPS
Fault, Configuration, Accounting, Performance, Security, 40

G

GND
Ground, 113
GPIO
General Purpose Input/Output, 10, 23, 114, 115, 118, 119, 120, 121, 126, 127
GPLv2
General Public License version 2, 43
GPS
Global Positioning System, 6, 10, 23, 59, 64, 65, 69, 76, 116, 117, 120

H

HDMI
High Definition Multimedia Interface, 54

I

I2C
Inter-Integrated Circuit, 54
IBM
International Business Machines, 3, 6, 7, 9, 10, 11, 25, 60, 111, 131, 132, 133, 134, 135, 138, 139
IDS
Intrusion Detection System, 42
IETF
Internet Engineering Task Force, 32, 42
IoT
Internet of Things, 22
IP
Internet Protocol, 31, 43, 47, 48, 61, 62, 63, 85, 88, 89, 95, 96, 112, 124, 143, 151
IPS
Intrusion Prevention System, 42
IPv4
Internet Protocol version 4, 46
IPv6
Internet Protocol version 6, 46
IR
Infrarred, 23
ISO
International Organization for Standardization, 5, 33, 40, 41
ITU
International Telecommunication Union, 41

L

LDR
Light Dependent Resistor, 6, 10, 64, 65, 76, 115, 116, 120

LED
Light Emitting Diode, 10, 53, 118, 119, 127
LoRaWAN
Low Power Wide Area Network, 47
LTE
Long Term Evolution, 43
LwM2M
Lightweight M2M, 47
LXC
Linux Containers, 9, 90, 91

M

MAC
Media Access Control, 47, 93
MANET
Mobile Ad-Hoc Network, 48
MD5
Message-Digest Algorithm 5, 38, 50, 51, 52, 124
MIB
Management Information Based, 6, 32, 33, 34, 36, 38, 46, 49, 51, 52, 63, 152
MIT
Managment Information Tree, 46
MPR
Multi Point Relay, 96
MQTT
MQ Telemetry Transport, 24, 47, 112, 152
MSC
Modulation and Coding Scheme, 94

N

NAT
Network Address Translation, 62
NMS
Network Management System, 6, 32, 50
NS3
Network Simulator 3, 3, 7, 8, 25, 26, 27, 43, 44, 45, 88, 89, 94, 95, 96, 110, 111, 112, 145

O

OID
Object Identifier, 34, 47, 48, 52, 53, 58, 85, 86, 87, 119, 120, 121, 123, 124, 128, 152
OLSR
Optimized Link Route State, 3

P

PDU
Protocol Data Unit, 11, 12, 35, 36, 37, 39, 87

R

RAM
Random Access Memory, 6, 9, 11, 12, 47, 65, 70, 76, 77, 78, 85, 131, 136, 138, 139, 141, 142, 143
RFC
Request For Comments, 32, 33, 34, 38, 151
RFID
Radio Frequency Identification, 23
ROM
Read Only Memory, 47

S

SCL
System Clock, 113, 118
SDA
System Data, 113, 118
SDN
Software Defined Networking, 45, 46, 112
SHA
Secure Hash Algorithm, 38, 50
SHM
Structural Health Monitoring, 23, 150
SMI
Structure Management Information, 8, 33, 34
SNMP
Simple Network Management Protocol, 3, 5, 6, 7, 8, 11, 24, 25, 26, 27, 31, 32, 33, 35, 38, 39, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 56, 57, 63, 85, 86, 88, 89, 96, 100, 107, 112, 122, 123, 124, 125, 148, 151, 152
SNMPv1
Simple Network Management Protocol version 1, 32
SNMPv2c
Simple Network Management Protocol version 2 community, 32
SNMPv3
Simple Network Management Protocol version 3, 3, 12, 24, 26, 32, 38, 39, 48, 50, 63, 111, 112, 122, 130, 151
SPI
Serial Peripheral Interface, 54

T

TAP
Terminal Access Point, 9, 89, 90
TBRPF
Topology Broadcast Based on Reverse Path Forwarding, 42
TCP
Transmission Control Protocol, 29, 43, 151
TMN3
Telecommunications Management 3, 41
TUN

Tunnel Network, 9, 89, 90
TVOC
Total Volatile Organic Compounds, 65, 68, 121

U

UART
Universal Asynchronous Receiver-Transmitter, 54, 116, 118
UDP
User Datagram Protocol, 24, 32, 85, 148
USB
Universal Serial Bus, 54, 114
USM
User-based Security Model, 38, 86, 151
UTC
Universal Time Coordinated, 128

V

VACM
View-based Access Control Mode, 38, 151
VANET
Vehicular Ad-Hoc Network, 48

W

WBAN
Wireless Body Area Network, 23, 150
WMN
Wireless Mesh Netwok, 48
WSN
Wireless Sensor Network, 48, 150

Capítulo 1: Introducción y objetivos

Los sistemas de gestión basados en la tecnología del IoT (*Internet of Things*) constituyen una solución tecnológica actual enfocada en la captura, procesamiento de eventos y variables físicas. Su operación es resultado de la interconexión y cooperación entre diversos tipos de agentes, tal como usuarios, sensores y actuadores.

En concreto, aplicaciones como por ejemplo el control del tráfico vehicular, la gestión de movilidad, el análisis de variables climáticas, la caracterización de escenarios urbanos, los sistemas de monitorización para alerta ante emergencias, la agricultura de precisión, entre otras, son algunas de las opciones posibles. Dichas aplicaciones representan la base tecnológica para un desarrollo urbano basado en la sostenibilidad con eficiencia en infraestructura, seguridad y prestación de servicios.

En este capítulo se presenta la definición del problema que se trata en el trabajo de titulación, así como la justificación y sus objetivos, y finalmente se detalla la estructura general del documento.

1.1 Definición del problema

El término *Smart Cities*, se utiliza para referirse a ciudades que tiene por características incluir el uso de las Tecnologías de la Información y Comunicación (TIC por sus siglas en inglés) para la recopilación, transmisión y procesamiento de la información de manera que se puedan resolver problemas de desarrollo urbano relacionados con la gestión de transporte público, la gestión energética, la sostenibilidad, entre otros [1].

En cuanto a la recopilación y generación de datos, estos se obtienen principalmente gracias al despliegue de redes inalámbricas de sensores, las cuales se pueden implementar en muchas aplicaciones industriales y de consumo. En estas aplicaciones se utiliza el IoT, tecnología que consiste en instalar diferentes tipos de sensores (RFID, IR, GPS, entre otros) y conectarlos a Internet a través de diversos protocolos para el intercambio de información. De esta forma es posible implementar aplicaciones como por ejemplo sistemas de reconocimiento inteligente, sistemas de ubicación, seguimiento, monitoreo y gestión [2][3].

En particular, las redes de sensores se utilizan en varias aplicaciones relacionadas a las *Smart Cities* tales como el monitoreo de fenómenos ambientales a través de diferentes plataformas sensorizadas en un campus universitario [4], el control de tráfico en las intersecciones basado en una red Ad Hoc para realizar transporte inteligente [5], el monitoreo de gases nocivos, así como la temperatura y humedad para un sistema de automatización industrial inalámbrico y seguro [6], la atención médica por medio de datos recolectados del cuerpo humano y subidos a la nube para su posterior análisis [7], el SHM (*Structural Health Monitoring*) para el control de seguridad y costos de mantenimiento [8].

Por otra parte, una configuración de especial interés en cuanto a las redes de sensores consiste en la interconexión de tipo multi-salto, donde cada nodo depende del nodo vecino para reenviar sus paquetes [9]. Este tipo de red puede ampliar su cobertura fácilmente y mejorar la conectividad sin el despliegue de una infraestructura cableada. Además, múltiples rutas pueden estar disponibles para aumentar la robustez de la red. Sin embargo, debido al uso de baterías una desventaja es su limitada capacidad energética que afecta al tiempo de vida de la red, la tasa de envío de paquetes y el *throughput* [10].

Además de las aplicaciones antes mencionadas existen también soluciones como las redes vehiculares destinadas a reducir la congestión del tráfico y disminuir el número de accidentes. De igual forma, aplicaciones que usan la adquisición de datos de usuarios mediante teléfonos móviles para compartir, interpretar y verificar información sobre el comportamiento de los usuarios y el entorno social [11]. También se tienen aplicaciones como las WBAN (*Wireless Body Area Network*) que se utilizan para servicios médicos; estas redes reducen el número de mensajes de control entre los dispositivos para tratar de contrarrestar el problema de la eficiencia energética [12].

Para la visualización y monitorización de los resultados obtenidos por las aplicaciones, una solución común es el uso del *framework* Node-RED, ya que permite el desarrollo de aplicaciones IoT por medio de programación basada en flujo. Adicionalmente, proporciona diferentes funciones para la integración de dispositivos de hardware IoT; esta herramienta permite definir gráficamente flujos de servicios a través de protocolos estándar, usar funciones API (*Application Programming Interface*), leer y escribir pines GPIO (*General Purpose Input/Output*) de Raspberry Pi [11]. En lo que respecta al uso de esta herramienta se tienen varias aplicaciones, por ejemplo, un sistema de apagado automático de luz y aire acondicionado para eficiencia energética en edificios [12], de manera similar un sistema de supervisión y control para un centro de transmisión de datos ambientales [13].

Con referencia a las aplicaciones en donde se cuenta con una red de sensores, la principal característica que se busca implementar es el permitir gestionar tanto los dispositivos como la información de forma remota. Para este objetivo se utilizan protocolos de administración de red teniendo en cuenta principalmente las limitaciones de memoria de los dispositivos, así como también la eficiencia energética de los mismos. Las soluciones implementadas comúnmente se basan en los protocolos SNMP, Zabbix y MQTT. Entre estas opciones se ha seleccionado el protocolo SNMP debido a que se trata de un mecanismo estándar, robusto, escalable y compatible con una diversidad de dispositivos.

Es importante mencionar una ventaja del protocolo seleccionado, ya que SNMP presenta un bajo consumo de energía respecto a otras alternativas, debido a que utiliza el protocolo UDP (*User Datagram Protocol*) junto con el menor tamaño del agente al momento de leer datos [14]. En relación a SNMP, existe una variedad de aplicaciones orientadas a *Smart Cities*, por ejemplo, en sistemas de telemetría en ambientes urbanos [15] y monitoreo de procesos en la producción de alimentos [16].

1.2 Justificación

Con el avance de la tecnología las aplicaciones enfocadas en *Smart Cities* van tomando cada vez una mayor importancia. El crecimiento de la población y la migración de personas de las áreas rurales hacia las ciudades. En tal sentido, se requiere administrar entornos orientados al IoT de manera simple, sin importar el enfoque, los equipos, ni las variables a ser administradas. Para esto se utiliza el protocolo SNMP que es un estándar robusto en este cometido.

En tal contexto, es fundamental contar con una interfaz de usuario que permita monitorear las variables y el estado de la red. En este cometido, el *framework* Node-RED permite la creación y diseño de una interfaz utilizando programación basada en flujos, lo que es una ventaja frente a otras alternativas.

En cuanto a las redes de sensores no estructuradas, estas no requieren de infraestructura como enrutadores o estaciones base para su funcionamiento. Por lo que cabe destacar que pueden ser desplegadas en prácticamente cualquier escenario de *Smart Cities*.

1.3 Alcance de la propuesta

El trabajo experimental consta de dos partes, la primera es el desarrollo de una aplicación para la gestión y monitorización remota de nodos. En concreto la aplicación permite el monitoreo de variables de rendimiento del nodo como la carga del CPU, el uso de la memoria, la temperatura del nodo y el consumo de energía. Con tal finalidad, se implementaron dos nodos mediante la plataforma Raspberry Pi. Sobre dicha plataforma, se instalaron sensores de geolocalización y de variables ambientales como contaminación, temperatura y luz; así como el uso de una batería para la alimentación.

En la tarea de monitoreo y gestión se emplea el protocolo SNMPv3 que otorga capacidades de autenticación y control de acceso. En particular, a través de los mensajes generados por el protocolo SNMP, la aplicación es capaz de administrar los recursos de los nodos, como por ejemplo seleccionar un perfil de consumo de energía o establecer un control sobre los sensores (activar o desactivar dispositivos).

Para administrar recursos del nodo, el sistema tiene la capacidad de optimizar los recursos del sistema, como la selección de un perfil energético predefinido, la capacidad de encender/apagar sensores y establecer un modo de ahorro de energía.

Para el desarrollo de la aplicación se utilizó el *framework* Node-RED integrándolo con el protocolo SNMP en un ordenador con el sistema operativo Ubuntu 20.04. Adicionalmente, la aplicación cuenta con una interfaz de usuario y la capacidad de conectarse a la nube de IBM para su monitoreo desde Internet.

La conexión entre los nodos se realizó mediante una red Ad Hoc con un enrutamiento estático y en uno de los nodos se configuró un *gateway* por defecto que permite la salida de la red hacia internet.

Para la segunda parte del trabajo, se empleó la herramienta NS3 con la finalidad de emular una red tipo Ad Hoc multi-salto con cuatro nodos. Para adecuar el escenario a uno real, se utilizan contenedores Linux lo que permite emular los nodos sobre hardware. De la misma manera los contenedores permiten inyectar tráfico SNMP real a la red.

Por medio de esta simulación se obtienen métricas de red como: *throughput*, porcentaje de recepción de paquetes y retardo de la red. La simulación se realiza en distintos escenarios:

- Cuatro nodos fijos y enrutamiento estático.
- Cuatro nodos fijos y enrutamiento dinámico OLSR.
- Cuatro nodos fijos con un quinto nodo móvil que se desplaza hacia el primero con enrutamiento dinámico OLSR.
- Medición del ancho de banda de la red con los cuatro nodos fijos.

1.4 Objetivos

1.4.1 Objetivo General

Implementar un sistema de gestión para la administración y monitorización de nodos con capacidades de integrarse en aplicaciones enfocadas en *Smart Cities*.

1.4.2 Objetivos Específicos

1. Implementar una red Ad Hoc entre dos nodos para la aplicación de gestión y monitoreo.
2. Utilizar el protocolo SNMPv3.
3. Integrar el protocolo SNMPv3 con el *framework* Node-RED.
4. Utilizar sensores ambientales y de geolocalización.
5. Implementar una interfaz con conexión a Internet en uno de los nodos.

6. Emular una red multi-salto Ad Hoc de cuatro nodos utilizando NS3 para inyectar tráfico SNMP y evaluar métricas de red.

1.5 Estructura del documento

El documento se encuentra estructurado de la siguiente manera. En el Capítulo 2, se presenta el marco teórico en donde se detallan los conceptos de *Smart Cities* e IoT, los *framework* más conocidos para el desarrollo de aplicaciones IoT. Por otra parte, se detallan conceptos relacionados al funcionamiento del protocolo SNMP, necesarios para la implementación de la aplicación; así como la gestión de dispositivos y seguridad que se incorpora en la tercera versión de SNMP. Finalmente, se presenta una descripción de las redes Ad Hoc y conceptos sobre el simulador NS3.

A continuación, en el Capítulo 3, se presentan los principales trabajos relacionados disponibles en la literatura respecto a aplicaciones para *Smart Cities* mediante redes no estructuradas y los protocolos que se utilizan. En el Capítulo 4, se detalla la arquitectura del sistema de gestión, en especial la configuración de SNMP, el proceso de desarrollo de la aplicación y la implementación de la red Ad Hoc.

Luego en el Capítulo 5, se exponen los resultados de las pruebas realizadas en el software de gestión. En el Capítulo 6, se describe el procedimiento para la emulación de la red Ad Hoc multi-salto en distintos escenarios y se realiza el análisis de cada uno. Finalmente, en el Capítulo 7 se exponen las conclusiones y recomendaciones del trabajo de titulación.

Capítulo 2: Marco teórico

En este capítulo se describen los principales conceptos relacionados con el ámbito de las *Smart City* e IoT, los mismos que permiten comprender hacia qué escenarios está destinado el trabajo de titulación. Se hace un repaso entre los principales *frameworks* para el desarrollo de aplicaciones IoT.

Se detalla el protocolo SNMP, el cual está formado de varios componentes y una estructura definida de mensajes ya que es de importancia para el desarrollo de la aplicación. De la misma manera, se describe la gestión de los dispositivos y la seguridad que ofrece la tercera versión de SNMP.

Se realiza una descripción de las redes Ad Hoc donde se presentan las principales ventajas y desventajas respecto a las redes estructuradas. Finalmente, se repasan conceptos básicos sobre el simulador NS3 y contenedores Linux.

2.1 Smart Cities

El desarrollo de las ciudades durante el Siglo XXI y la investigación de actividades para ciudades inteligentes se ha vuelto una prioridad, a la vez que tecnologías de la información y comunicación han avanzado de manera exponencial. La migración de la población de áreas rurales a las ciudades requiere nuevos métodos para administrar la complejidad de la vida urbana [17].

2.1.1 Definición

No existe una definición universalmente aceptada de *Smart City*, ya que depende del nivel de desarrollo, recursos y aspiraciones de los residentes de la ciudad, sin embargo, algunos especialistas [17] han sugerido algunas definiciones como:

- Una *Smart City* utiliza las tecnologías de la información y comunicación para mejorar su habitabilidad y sustentabilidad.

- Una ciudad que monitorea e integra las condiciones de su infraestructura crítica como las obras civiles, las comunicaciones, los sistemas de distribución de agua y energía. Con el fin de optimizar los recursos, la planificación de actividades de mantenimiento y monitoreo de seguridad, mejorando los servicios para los ciudadanos.

- Es un espacio geográfico determinado capaz de manejar recursos naturales, humanos, edificios e infraestructura, que genera el estilo de vida del lugar de manera sustentable y que no sea perjudicial para el medio ambiente.

2.1.2 Características y herramientas

Una *Smart City* tiene dos componentes principales: características y herramientas. Entre las características inteligentes se encuentran: la innovación, su habitabilidad, la eficiencia, que sea resiliente, amigable con el medio ambiente y energéticamente sostenible. En cuanto a las herramientas que se utilizan para lograr estas características se tienen las tecnologías de la información y comunicación, manejo de datos, enfoque participativo por parte de los integrantes de la ciudad, y financiación ya sea pública o privada [18] [19].

2.2 Tecnología IoT y frameworks

2.2.1 IoT (Internet of Things)

El IoT es una infraestructura global para la sociedad de la información, que propicia la prestación de servicios avanzados mediante la interconexión de objetos físicos y virtuales. En particular, esto se da gracias a la interoperatividad de las tecnologías de la información y comunicación, implementadas en el presente y las que serán desplegadas en el futuro [19].

En este sentido se aplica el uso de herramientas de adquisición de datos, procesamiento y comunicación, con el fin de obtener objetos para ofrecer servicios como se presenta en la Figura 1, los cuales están destinados a todo tipo de aplicaciones, garantizando requisitos de seguridad y privacidad.

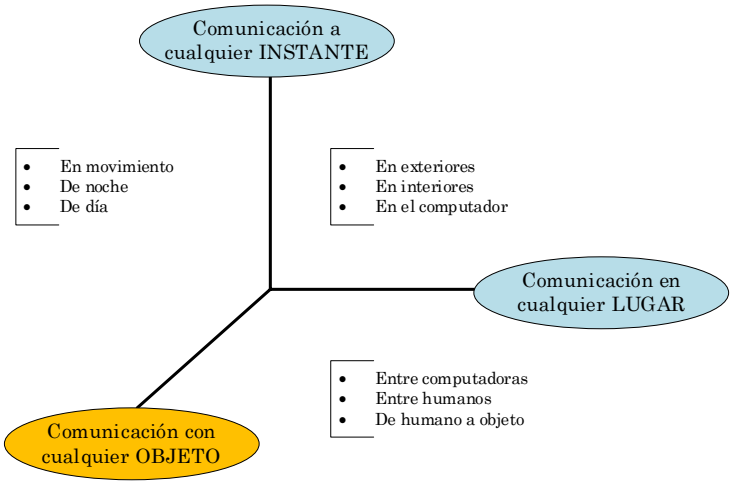


Figura 1: Posibilidad de comunicación que añade IoT

Los objetos mencionados pueden pertenecer al mundo físico (equipos eléctricos, bienes, entre otros) o al mundo virtual (software, contenido multimedia, entre otros) y estos se pueden integrar en redes de comunicación. En la Figura 2, se observa que estos objetos tienen información conexa, que puede ser estática o dinámica ya sea por medio de un *gateway* o directamente entre ellos.

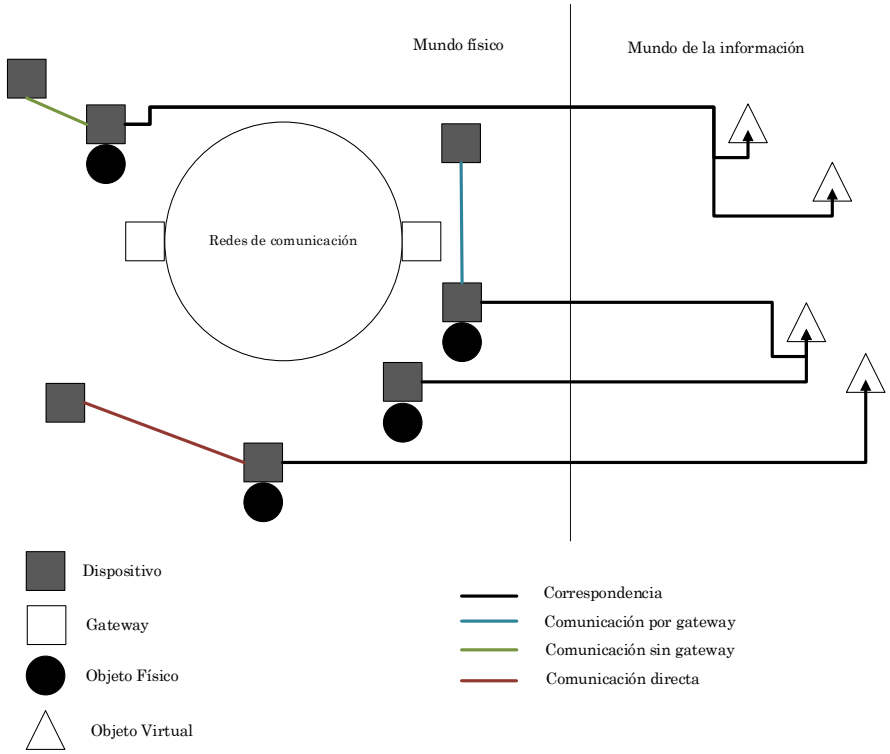


Figura 2: Descripción técnica de un entorno IoT

Para cada objeto, existe un dispositivo que es una pieza con capacidades de comunicación y en algunos casos de detección, accionamiento, adquisición, almacenamiento y procesamiento de datos. En cuanto a las aplicaciones y servicios IoT, generalmente ofrecen capacidades genéricas como pueden ser autenticación, gestión de dispositivos, adquisición de variables, entre otros.

La infraestructura en la que se encuentran tanto objetos como dispositivos, puede crearse mediante redes existentes basadas en TCP (*Transmission Control Protocol*) en redes evolutivas. En la Figura 3, se relacionan los tipos de dispositivo y la relación que tienen con objetos físicos.

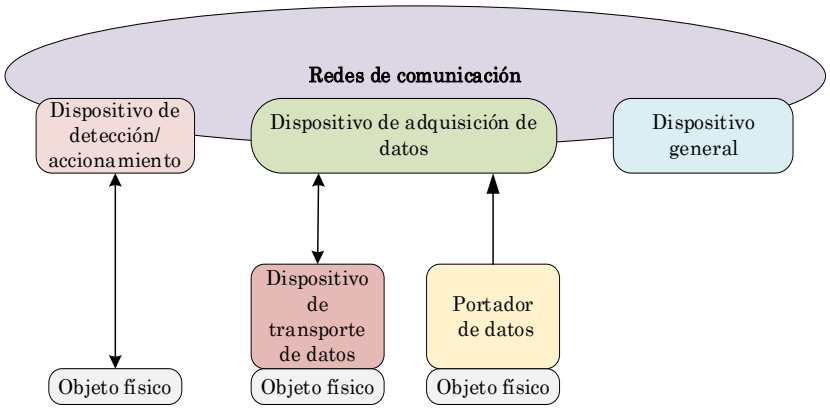


Figura 3: Tipos de dispositivos y su relación con objetos físicos

2.2.2 Tipos de dispositivos

Los dispositivos IoT deben cumplir un requisito mínimo que es disponer capacidades de comunicación, en base a esto se dividen en los siguientes tipos [19]:

- Dispositivos de transporte de datos: Es un dispositivo que va con un objeto físico y sirve para la conexión con las redes de comunicación.

- Dispositivo de adquisición de datos: Dispositivo con capacidad para interactuar con objetos físicos de manera directa o indirecta, a través de dispositivos de transporte de datos.

- Dispositivo de detección y accionamiento: Dispositivo que mide información de su entorno y la convierte en señales electrónicas digitales. Por lo general, forman redes locales que luego se conectan a redes de comunicación más grandes.

- Dispositivo genérico: Dispositivo que cuenta con capacidades de procesamiento y comunicación. Los dispositivos generales incluyen equipos y aplicaciones para diferentes dominios de aplicación IoT, como pueden ser máquinas industriales, electrodomésticos y teléfonos inteligentes.

2.2.2.1 Características fundamentales

Como características principales del IoT se destacan las siguientes:

- Interconectividad: Hacia una infraestructura de información y comunicación.

- Heterogeneidad: Es fundamental ya que los dispositivos IoT vienen de distintas plataformas y fabricantes. Además, se considera que estos deben interactuar entre sí.

- Cambios dinámicos: En el estado de los dispositivos debido a su disponibilidad, ubicación, velocidad y el número de dispositivos puede ser variable.

- Escalabilidad: Debido a que en el futuro el número de dispositivos a gestionarse puede ser mayor al número de dispositivos conectados hoy en día a Internet.

2.2.3 Frameworks

Existe una variedad de *frameworks* orientados al desarrollo de IoT, algunas propuestas son de *software* libre y otras de propietario. Cada uno de estos *frameworks* permiten la integración de dispositivos y la gestión tanto por medio de la nube como a través de redes locales. En la Figura 4, se destacan los más importantes con sus principales características.

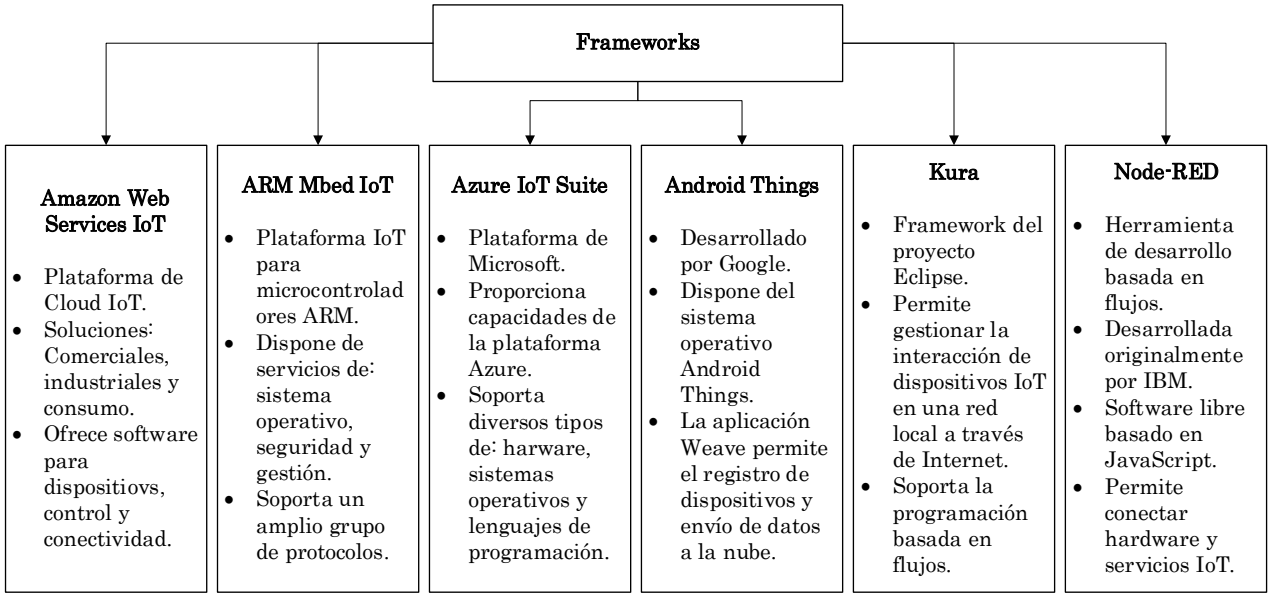


Figura 4: Principales framework de desarrollo IoT

2.3 El protocolo SNMP

El protocolo SNMP, es un estándar para administrar dispositivos en redes IP [20]. Este protocolo brinda la capacidad de configurar, monitorizar y corregir un sistema de manera remota mediante un conjunto de operaciones simples. SNMP puede usarse para administrar una gran variedad de tipos de dispositivos.

El protocolo SNMP ha sido desarrollado por el IETF (*Internet Engineering Task Force*) y está definido en la RFC 1157 [21]. El protocolo SNMP cuenta con tres versiones SNMPv1, SNMPv2c y SNMPv3, las cuales se diferencian por las mejoras y ventajas que se integran entre las versiones, relacionadas principalmente a la seguridad. En la pila de protocolos se encuentra en la capa de aplicación. Los mensajes SNMP, estos se encapsulan en datagramas UDP.

Los mensajes que se generan de una consulta o respuesta se envían hacia el puerto 161 y los mensajes que se originan de una notificación se envían hacia el puerto 162. En la Figura 5, se observan los protocolos que intervienen en el transporte de un mensaje SNMP.

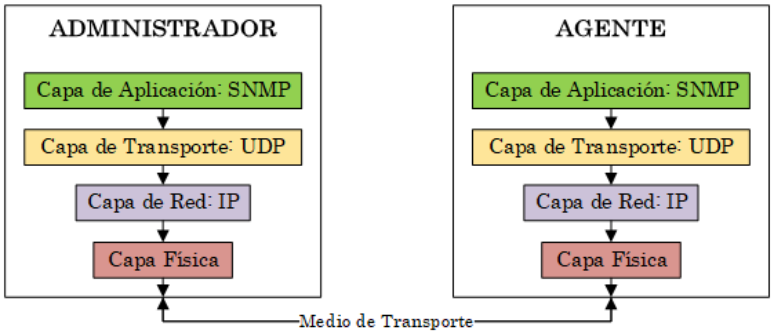


Figura 5: Transporte de un mensaje SNMP

2.3.1 Componentes del protocolo SNMP

Los componentes fundamentales para el funcionamiento de SNMP se describen a continuación [21]:

- Administrador de red (NMS): Es el encargado de controlar y monitorear a los dispositivos que componen la red. Además, se encarga de recibir las notificaciones y respuestas enviadas por los agentes, en las cuales se obtiene información de los elementos gestionados.

- Agente: Es un programa que se encuentra en los dispositivos gestionados y se encarga de responder las peticiones realizadas por el administrador. Así como también el envío de notificaciones cuando sucede un evento con el elemento gestionado.

- Base de información administrada (MIB): Es la base de datos que contiene la información de los objetos gestionados, la cual es manejada por el agente que puede consultar o alterar esta información.

- Protocolo de administración: Es el protocolo que establece normas, para la comunicación entre un agente y un administrador, así como el manejo de la MIB. Con

SNMP, la comunicación se da por medio de tres procesos: las consultas, las respuestas y las notificaciones.

Los componentes de la arquitectura se pueden ver en la Figura 6, donde se resaltan cada uno de los elementos fundamentales del protocolo SNMP. Además, se observan los procesos que se generan en la comunicación entre el agente y el administrador.

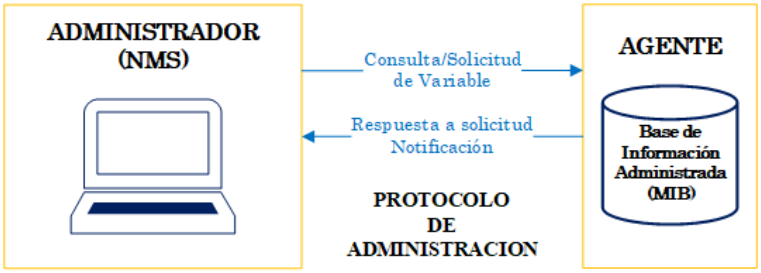


Figura 6: Arquitectura de SNMP

2.3.2 Representación de la información gestionada

Para acceder a la información de los objetos gestionados, se usa la MIB. Los elementos que componen esta base son definidos mediante la SMI (*Management Information Base*). Este estándar se emplea para definir como se nombran los objetos gestionados y especificar el tipo de datos que contiene. El MIB se organiza en forma de un árbol, en el que cada nodo se asocia un número entero positivo [22]. En la Figura 7, se observa el árbol definido por la ISO, donde se puede apreciar la jerarquía de la MIB y en la que se verifican los niveles correspondientes a las distintas organizaciones.

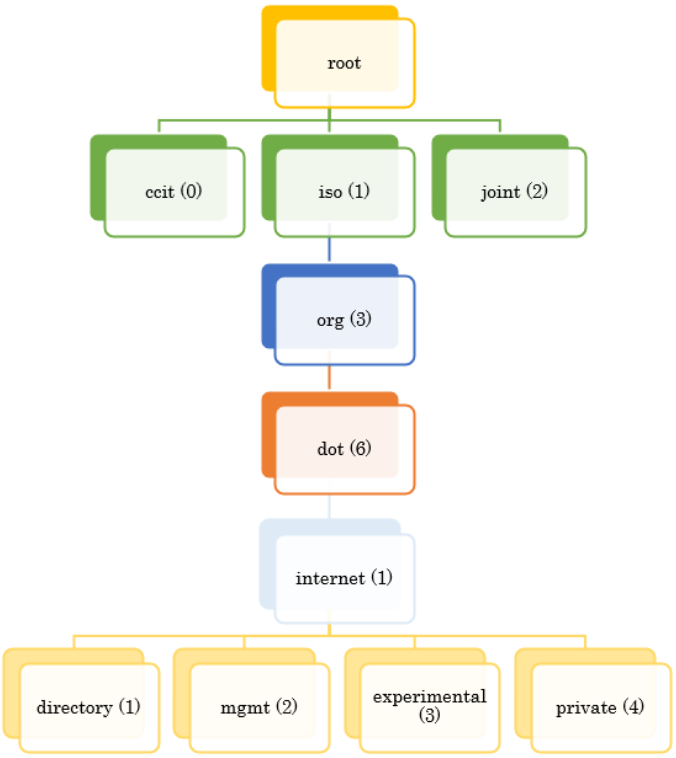


Figura 7: Árbol de Objetos SMI

Existen dos versiones de la MIB, la MIB-I o concisa y la MIB-II, las cuales se encuentran definidas por los RFC 1212 y 1213 respectivamente. Estas versiones se diferencian entre sí por las mejoras que se incluyen, tales como el agregar y depurar objetos ya definidos [23]. Cada objeto que está definido en la MIB-I tiene un nombre o OID (*Object Identifier*), un tipo-sintaxis, y una codificación [22], como se lista a continuación:

- OID: Identifica unívocamente a un objeto, es una secuencia de número enteros positivos, que están separados por un punto. Para identificar un objeto se debe recorrer desde la raíz hacia el nodo de interés.
- Sintaxis: Define la estructura de datos del tipo de objeto.
- Codificación: Define las reglas de codificación básica.

La versión MIB-II agrega campos opcionales a un objeto, lo que otorga más control sobre el acceso hacia un objeto. Los nuevos campos son [20]:

- *UnitParts:* Descripción de las unidades utilizadas para representar al objeto.
- *MAX-ACCESS:* El acceso que se otorga a un objeto, puede ser lectura, escritura, no accesible y accesible para notificación.

- *STATUS:* Campo que se amplía, para poder utilizar las palabras clave *obsolete* y *deprecated.*

- *ARGUMENTS:* Permite agregar una o más columnas a una tabla.

2.3.3 Mensajes SNMP

La PDU (*Protocol Data Unit*), es el formato de mensaje que usa el administrador y los agentes para el envío y recepción de información. Las operaciones mencionadas a continuación, tienen un formato PDU estándar: *Get, GetNext, Set, GetResponse, Trap*.

Un mensaje del protocolo SNMP, está compuesto por un identificador de versión, un nombre de comunidad y un PDU, y se define mediante el estándar ANS.1 (*Abstract Syntax Notation One*) como se indica a continuación [20]:

Message ::= SEQUENCE {
version INTEGER
community OCTET STRING,
data ANY }

En la Tabla 1, se observa la estructura de los formatos de los PDUs del protocolo, entre los que se encuentra en primer lugar la versión de SNMP utilizada, luego, la comunidad que es la relación que existe entre un agente y un grupo de aplicaciones SNMP y, por último, el campo relacionador a SNMP PDU que se detalla a continuación.

Versión	Comunidad	SNMP PDU

Tabla 1: Mensaje SNMP

El campo relacionador a SNMP PDU puede tomar uno de los siguientes cinco valores, *GetRequest, GetNextRequest, GetResponse, SetRequest y Trap* [23]:

- La PDU del tipo *GetRequest* realiza la consulta del valor de una o varias variables. En respuesta a este PDU, se obtiene un PDU de tipo *GetResponse,* con los valores de las variables solicitadas.

- La PDU *GetNextRequest* se utiliza para consultar el valor de la siguiente variable a la indicada o indicadas.

- En el tipo *GetResponse*, se origina como respuesta a las PDUs de tipo *GetRequest*, *GetNextResquest* y *SetRequest*.

- La PDU *SetRequest* se utiliza para cambiar o establecer el valor de una o más variables de la MIB. Para cambiar un valor se debe tener un permiso de lectura-escritura.

La Tabla 2, muestra la estructura de un mensaje generado por las operaciones mencionadas anteriormente, donde se destaca el Tipo PDU que indica el tipo de PDU que va en el mensaje, el cual puede ser de tipo *Request* o *Response*. De la misma manera se encuentra la *ID Petition* que se utiliza para distinguir entre las solicitudes, cada solicitud tiene una identificación única. Además, permite identificar respuestas repetidas.

Tipo PDU	*ID Petition*	0	0	Campos Variables

Tabla 2: *Get Request, GetNextRequest y SetRequest* PDU

La estructura de un mensaje específicamente del tipo *Get Response* PDU, se pueden observar en la Tabla 3.

Tipo PDU	*ID Petition*	*Error-Status*	*Error-Index*	Campos Variables

Tabla 3: Get Response PDU

Entre los parámetros presentados de en la Tabla 3, se detallan los tipos de errores que se pueden presentar.

- *Error-Status:* Indica el tipo de error que se genera, este puede tomar los valores de la Tabla 4.

Error-Status	Descripción
noError	Sin error.
tooBig	La respuesta es demasiado grande.
noSuchName	La variable consultada no existe.
badValues	El valor asignado por *SetRequest* no es el correcto.
readOnly	La variable a modificar, es de solo lectura.
genErres	Cualquier tipo de error.

Tabla 4: Errores generados en un *Trap*-PDU

- *Error-Index:* Muestra la posición de la variable responsable del error.

- En caso de no haber errores el receptor devuelve una PDU de tipo *Response* con el campo *Error-Status* establecido a *NoError* y *Error-Index* a 0.

- Mediante la PDU del tipo *Trap* los agentes informan que se ha dado un suceso inusual en el dispositivo gestionado. Este mensaje se genera de manera asíncrona ya que debe

ser reportado sin la necesidad de una solicitud. En la Tabla 5, se observa la estructura de un mensaje de tipo *Trap*.

Tipo PDU	Empresa	Dirección agente	*Trap* genérico	*Trap* específico	*Time-stamp*	Campos Variables

Tabla 5: *Trap* PDU

En esta estructura de mensaje se pueden encontrar los siguientes parámetros:

- Empresa: Representa el tipo de objeto que genera un *Trap*.
- Dirección agente: Es la dirección de red del objeto generador del *Trap*.
- *Trap* genérico: En la Tabla 6, se observan los *Trap* genéricos que se pueden generar.

Trap	**Descripción**
Cold Start	Cuando un agente se reinicia y su configuración cambia.
Warm Start	Cuando un agente se reinicia y no se dan cambios en la configuración.
Link Down	Cuando se producen problemas en los enlaces de comunicación.
Authentication Failure	Fallas en la autenticación.
Egp Neighbor Loss	Si un vecino EGP no está disponible.
EnterprisseSpecific	Cuando no se puede clasificar un *trap.*

Tabla 6: Trap genéricos

- *Trap* especifico: Es un código específico del *Trap*.
- *Time-stamp*: Es el tiempo transcurrido entre la última vez que se reinició el dispositivo de red y la generación del *Trap*.
- Campos variables: Es una lista de nombre de variables con sus correspondientes valores, normalmente contiene datos solicitados por una operación *Get* o *Trap*.

2.3.4 SNMPv3 (Versión 3)

La versión 3 de SNMP está definida en la RFC 3410, y proporciona seguridad a través de mecanismos de autenticación y encriptación de los paquetes a través de la red [24]. Las principales características que se integran en esta versión son:

- Autenticación: Determina que los mensajes provengan de una fuente válida, proporciona una identidad al usuario desde el cual se generan los mensajes.

- Encriptación: Evitan que el contenido de un mensaje sea observado por una fuente externa. Realiza el proceso para que los mensajes generados entre el agente y administrador sean confidenciales.

El protocolo SNMPv3 implementa el modelo de seguridad de usuario (USM por sus siglas en inglés) y un modelo de control de acceso (VACM por sus siglas en inglés), los cuales se definen en la RFC 3414 y RFC 3415. Debido a esta funcionalidad, la arquitectura para esta versión presenta cambios.

- Modelo de seguridad basado en usuarios: Se encarga de verificar que los mensajes recibidos, no hayan sido modificados. Además, controla que el contenido de cada uno de los mensajes esté protegido contra la divulgación [25].

- Modelo de control de acceso basado en vistas: Realiza el control de tipo de acceso el cual puede ser de lectura, escritura o notificación [26].

2.3.4.1 Arquitectura SNMPv3

De acuerdo a la RFC 2271, los elementos que componen la versión 3 de SNMP se conocen como entidades. Cada entidad se encarga de realizar una función específica y contiene un motor SNMP que se encarga del envío y recepción de mensajes SNMP [27]. Los elementos que conforman el motor SNMP son:

- Despachador: Realiza el trabajo de enviar y recibir mensajes, hacia otras entidades.

- Subsistema de procesamiento de mensajes: Prepara los mensajes que se van a enviar y extrae la información de los mensajes que se reciben.

- Subsistema de seguridad: Provee de servicios de autenticación y privacidad, para lo cual se utilizan algoritmos MD5 o SHA para la autenticación y los algoritmos DES y AES para realizar el cifrado y descifrado de los mensajes.

- Subsistema de control de acceso: Controla el acceso y las operaciones que un usuario puede realizar en los objetos de la MIB.

La entidad SNMP también está compuesta de aplicaciones SNMP, los cuales son: el generador de comandos, un contestador de comandos, un generador de notificaciones, un receptor de notificaciones y un despachador proxy [27].

- Generador de comandos: Se encarga de supervisar y manipular la administración de los datos.
- Contestador de comandos: Este proporciona acceso a los datos de gestión.
- Originadores de notificación: Inician mensajes asíncronos para indicar un evento inusual.
- Receptor de notificaciones: Procesan los mensajes asíncronos recibidos.
- Despachador *proxy:* Realiza el reenvío de mensajes entre entidades.

2.3.4.2 Mensajes SNMPv3

La versión 3, posee un formato de mensaje específico para la transferencia de información entre entidades. El mensaje contiene un encabezado y la PDU correspondiente a las operaciones de SNMP [24]. En la Tabla 7, se presenta el formato de mensaje.

Formato	Descripción
msgVersion	Versión del mensaje SNMP, establecida en 3.
msgID	Identificador de mensajes de solicitud y respuesta.
msgMaxSize	Tamaño máximo de un mensaje admitido por SNMP.
msgFlags	Cadena de ocho bits, que informa si un mensaje debe generar una PDU de informe.
msgSecurityModel	Indica el modelo de seguridad utilizado.
msgAuthoritativeEngineID	Cadena que contiene la información generada por el subsistema de seguridad.
msgAuthoritativeEngineBoots	
msgAuthoritativeEngineTime	
msgUserName	
msgAuthenticationParameters	
msgPrivacyParameters	
contextEngineID	Identifica de forma única a una entidad SNMP.
contextName	Identifica un contexto del motor SNMP.
PDU	Son las PDU generadas por las operaciones SNMP.

Tabla 7: Formato de mensajes SNMPv3

2.4 Gestión de dispositivos y seguridad

La gestión de red o dispositivos, se define como el conjunto de acciones con las cuales se realiza el control, organización y supervisión de los recursos de los dispositivos o la red. De manera que se garantice una adecuada calidad de servicio con el mínimo costo. Logrando así mejorar la disponibilidad, rendimiento y efectividad de los sistemas [28].

2.4.1 Procesos de la gestión de dispositivos

Las principales operaciones que se realizan en la gestión son la monitorización y el control [28].

- Monitorización: Obtiene la información de los dispositivos de la red. La información monitorizada puede ser estática cuando no cambia con el tiempo o dinámica cuando cambia frecuentemente.

- Control: Mejora el rendimiento de los servicios, mediante la definición de la información de configuración, el cambio de atributos y la modificación de relaciones.

2.4.2 Modelos de administración ISO

El modelo ISO, define cinco áreas correspondientes a la gestión conocidas como FCAPS (*Fault, Configuration, Accounting, Performance, Security)* que se describen a continuación [20]:

- Gestión de Fallos (*Fault*): Monitoriza un dispositivo o red gestionada, con el fin de detectar, registrar y notificar al usuario de problemas suscitados. De esta manera, se busca cumplir con componentes: fiabilidad, disponibilidad y la supervivencia.

- Gestión de la Configuración (*Configuration*): Realiza la monitorización de la información, modificación de la configuración de los dispositivos y el almacenamiento de la información.

- Gestión de la Contabilidad (*Accountating*): Verifica que los recursos de la red sean utilizados de manera eficiente.

- Gestión de las Prestaciones (*Performance*): Evalúa el comportamiento de los elementos de la red, y su efectividad. Para lo cual realiza el análisis estadístico de los datos obtenidos.

- Gestión de la Seguridad (*Security*): Controla el acceso a los recursos, además de ayudar a detectar y prevenir ataque que ponen en riesgo el funcionamiento de los dispositivos o la red.

2.4.3 Arquitectura de gestión de red

Existen tres modelos básicos que definen como se realiza la gestión de una red [29]:

- Administración ISO: Definido por la ISO para gestionar los recursos de una red siguiendo el modelo de la ISO presentada anteriormente.

- Administración de Internet: Esta basado en el modelo del protocolo SNMPv2 y utiliza una arquitectura del tipo administrador-agente.

- Arquitectura TMN3: Está definida por la ITU (International Telecommunication Union) para gestionar las redes de telecomunicaciones.

2.4.4 Gestión de seguridad

La gestión de seguridad debe garantizar que solo los usuarios autorizados tengan acceso físico a los dispositivos de la red. De igual manera, se debe proveer autenticación, control de acceso, integridad y confidencialidad de los datos [30].

Las funciones que cumple esta gestión, se dividen en las categorías detalladas a continuación:

- De prevención: Función que se utiliza para evitar la intrusión.

- De detección: Esta función detecta las intrusiones.

- De contenencia y recuperación: Para evitar el acceso de un intruso, y reparar los fallos causados por un intruso, también se encarga de recuperar las pérdidas en los datos.

- De administración de la seguridad: Utilizada para planificar y administrar las políticas de seguridad, así como también la información relacionada a la seguridad.

La gestión de seguridad debe cumplir los siguientes requisitos [30]:

- Integridad: Evitar que los recursos sean modificados por usuarios no autorizados, además que la información tenga protección contra la divulgación.

- Confidencialidad: Controlar que la información pueda ser accedida por quienes tienen los permisos de lectura, escritura.

- Disponibilidad: Se necesita que la información esté disponible en el momento exacto que se requiera acceder a ella.

Los requisitos de seguridad se logran mediante el uso de herramientas y sistemas creados para cumplir con los mismos: Estas herramientas son [20]:

- Firewall.
- Sistemas de Detección de Intrusos (IDS).
- Sistemas de Prevención de Intrusos (IPS).
- Antivirus.
- Políticas de administración.

2.5 Redes no estructuradas o Ad Hoc

Una red Ad Hoc es un conjunto de nodos inalámbricos, que pueden ser móviles y dinámicos que forman una red sin el uso de infraestructura o administración centralizada. En sus inicios la idea de este tipo de redes fue concebida por DARPA (*US Defense Advanced Research Projects Agency*) en los años 1970s [31].

Los nodos o dispositivos son libres de moverse en la red y de organizarse de manera arbitraria con la posibilidad de operar de manera independiente o conectado a Internet. Entre los retos que presenta una red Ad Hoc está el que se pueda implementar el multi-salto entre los nodos, la movilidad y la heterogeneidad. Además, este tipo de red debe considerar el ancho de banda y uso eficiente de la energía. En lo que respecta a la estandarización de las redes Ad Hoc, se han realizado investigaciones para desarrollar un grupo de protocolos bajo un grupo formado en la IETF.

Es necesaria una forma de enrutar los nodos de la red, para esto existen varios protocolos, entre los más destacados se encuentran:

- Enrutamiento estático.
- AODV (*Ad Hoc On-Demand Distance Vector*).
- OLSR (*Optimized Link State Routing*).
- DSR (*Dynamic Source Routing*).
- TBRPF (*Topology Broadcast Based on Reverse Path Forwarding*).

Cada uno de estos protocolos tienen sus ventajas y desventajas, ninguno es superior a otro y debe ser seleccionado según las necesidades específicas de la red y la cantidad de nodos [32]. En contraparte de una red Ad Hoc están las redes estructuradas que requieren de una infraestructura central para enrutar los dispositivos, también existen las redes celulares que requieren una planificación detallada y altos costos de instalación de su infraestructura. Cabe señalar que se puede implementar una red Ad Hoc en el caso de que una red estructurada o una red celular no esté disponible o no sea económicamente viable.

Las redes inalámbricas ya sean estructuradas o no, tienen problemas inherentes que se presentan en el canal inalámbrico ya que este es débil, sensible al ruido y no es robusto. El medio inalámbrico no tiene fronteras bien definidas por lo que no se puede determinar si un nodo está fuera del alcance de otro. Otro punto importante que se debe considerar es que el canal es variante en el tiempo y tiene propiedades de propagación asimétricas, de la misma manera los problemas del nodo oculto y nodo expuesto pueden darse.

En cuanto a la seguridad en los dispositivos, esta característica se debe tener en cuenta ya que es un tema complejo. Esto se debe a que la red por lo general es heterogénea y está soportada por cada uno de sus nodos, por lo tanto, cada uno de ellos es vulnerable [33].

Entre las áreas a investigar y desarrollar en una red no estructurada las más importantes son:

- Disponibilidad: Asegura que los servicios estén activos siempre que sean necesarios. Esta característica es incluso más importante que en una red estructurada porque todos los dispositivos dependen entre sí, por lo que un ataque de denegación de servicio (DoS) es simple de ejecutar.

- Manejo de claves y autenticación: Para aplicaciones de terceros ya que identificar a un nodo externo que ingresa a la red es una tarea a considerar.

- Confidencialidad de los datos: Debido que al ser un canal inalámbrico un atacante puede colocar un *sniffer* y capturar los paquetes.

- Integridad de los datos: Que los mensajes enviados de un nodo a otro no sean modificados en el trayecto por un nodo malicioso.

2.6 Simulador NS3

La herramienta NS3 es un simulador de eventos discretos, enfocado a la investigación de redes. Esta desarrollado en C++, de código abierto y distribución gratuita, además que se puede utilizar en ámbitos educativos y de investigación. NS3 cuenta con una licencia GNU GPLv2 y tiene soporte para funcionar en la plataforma Linux y Windows.

Las funcionalidades que se pueden implementar con este simulador pueden ser redes inalámbricas como Wi-Fi o LTE, redes de sensores, redes Ad Hoc, redes TCP. Todas las funcionalidades, permiten que NS3 se use como un emulador de red en tiempo real ya que también se puede implementar distintas aplicaciones y protocolos. Para implementar un escenario de simulación, se puede utilizar tanto el lenguaje C++ como Python.

2.6.1 Arquitectura básica de un escenario en NS3

Los principales elementos que posee una simulación de red son:

- Nodo: Es la abstracción de un dispositivo como una computadora o un *router*. A este elemento se le puede agregar funcionalidades como protocolos, dispositivos de red, aplicaciones, etc.

- Canal: Es la representación del medio físico por el que se transmiten los datos en los nodos.

- Dispositivo de red: Es la abstracción tanto de software como de hardware, de una tarjeta de red. Al añadir este elemento a un nodo, este puede comunicarse con otros por medio de un canal. En un dispositivo, se puede tener uno o varios dispositivos de red.

- Pila de protocolos: Se encarga de implementar en un nodo, una pila de protocolos de Internet.

- Aplicaciones: Es un programa, que se ejecuta en los nodos para realizar diferentes tareas. De esta manera, se puede generar diferente tráfico en la red.

En la Figura 8, se puede ver los diferentes elementos que componen una estructura básica de red en NS3.

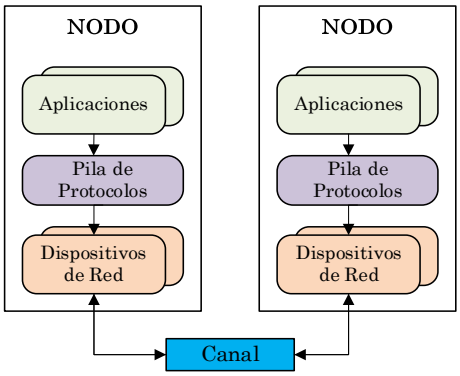


Figura 8: Estructura básica de una red en NS3

2.6.2 Contenedores Linux en NS3

Un LXC (*Linux Container*) es una tecnología de virtualización que permite ejecutar sistemas operativos aislados en un hardware físico, además, tiene la capacidad de configurar su red y posee su propio espacio para aplicaciones y procesos.

La herramienta NS3 permite realizar la emulación de una red utilizando hardware real mediante el uso de contenedores. El proceso de virtualización a nivel de red se realiza utilizando dispositivos TUN/TAP que son interfaces de red virtuales de *kernel*, Estos dispositivos de red están soportados totalmente en software y difieren de los dispositivos de red ordinarios en que se encuentran soportados por adaptadores de hardware.

- TUN (*Tunnel Network*): Simula un dispositivo de capa de red y lleva los paquetes IP.
- TAP (*Terminal Access Point*): Simula un dispositivo de capa de enlace y lleva los *frames ethernet*.

Los dispositivos TUN se usan para *routing* y TAP para crear un puente de red, en la Figura 9, se resalta una representación de la ubicación de los dispositivos TUN/TAP en la pila protocolos. Como se puede apreciar, los paquetes se envían al sistema operativo a través de los dispositivos los TUN/TAP, ya que se inyectan en la pila de protocolos del sistema emulando la recepción de una fuente externa.

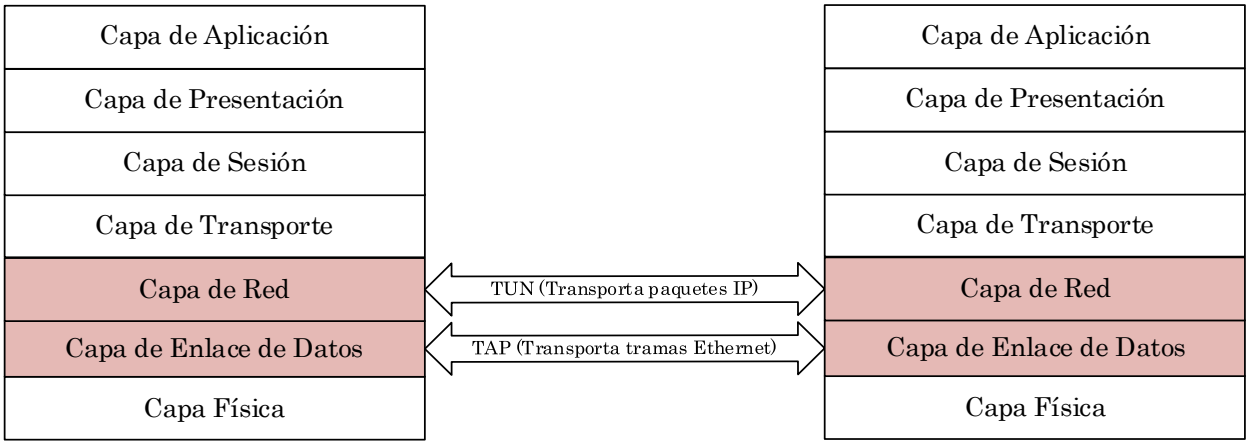


Figura 9: Ubicación de los dispositivos TUN/TAP en la pila de protocolos

2.7 Conclusiones

En este capítulo se revisaron los conceptos relacionados para la comprensión y desarrollo del trabajo de titulación. En un inicio, contextualizando que es una *Smart City* y el IoT. En conceptos más técnicos se profundizó en el protocolo SNMP haciendo énfasis en la versión 3 que se requieren para el desarrollo de la aplicación.

A continuación, se repasaron conceptos de las redes Ad Hoc mostrando las ventajas que tienen respecto a las redes estructuradas, así como las deficiencias que estas presentan.

Finalmente, se revisaron conceptos introductorios al software de simulación de redes NS3 con sus principales funcionalidades y el uso de contenedores Linux. Lo que permite comprender la estructura de las simulaciones del trabajo de titulación.

Capítulo 3: Estado del arte

En este capítulo se describen trabajos relacionados disponibles en la literatura, se comienza por tratar el tema de las *Smart Cities* y el IoT en general. Se describen propuestas para la gestión de dispositivos en la red que utilizan el protocolo SNMP. A continuación, se hace una revisión de aplicaciones que utilizan alternativas similares a SNMP. Para finalizar se analiza los tipos de redes Ad Hoc multi-salto y sus aplicaciones.

El término *Smart City* es utilizado para referirse a las ciudades que utilizan la tecnología en conjunto con otros recursos para mejorar la calidad de vida de sus ciudadanos [34]. En una ciudad inteligente se hace uso de las TIC para fomentar una mejora en diferentes aspectos como su economía, población y ecosistema. Esto se realiza a través de aplicaciones que analizan distintos ámbitos como son la satisfacción personal, la prosperidad económica, entre otros. Estas aplicaciones se basan en la recopilación de la información para lo cual se utilizan redes de sensores, redes Ad Hoc vehiculares, IoT, redes definidas por software (SDN), entre otras.

Las aplicaciones que se desarrollan mediante el uso de las tecnologías mencionadas anteriormente, se pueden implementar en diferentes campos como la energía sostenible y el medio ambiente, en sistemas de vida asistida, sistemas de tráfico inteligente, en sistemas de transporte inteligente, entre otros [35]. Entre estas aplicaciones se destaca el uso de IoT, ya que permite simplificar los procesos realizados en diferentes áreas; de esta manera se logra una mejor eficiencia en los sistemas en los que se implementa.

El uso del IoT, ha permitido llevar a cabo diferentes aplicaciones enfocadas a *Smart Cities.* Por ejemplo, en [36] se analiza el caso de la gestión de residuos mediante el uso de sensores en los contenedores, esto con el objetivo de optimizar el recorrido de los recolectores, de esta manera se ahorra tiempo y se reduce el consumo de combustible.

Así también, en [37] se realiza una investigación sobre su uso en la industria de la minería, para lo cual se utilizan como base las redes Ad Hoc vehiculares con un protocolo de enrutamiento basado en la optimización de colonia de hormigas. Mediante este proceso, se demuestra que se mejora el rendimiento en relación a otros protocolos, así también se muestra un cambio e innovación en esta industria.

De igual forma en [38], se estudia la implementación de un sistema de riego automático para plantaciones, en el cual se usa sensores que ayuden a controlar la humedad del suelo. La información generada se puede monitorear tanto de forma local como de forma remota. Con la implementación de este sistema, se muestra que se puede mejorar la producción en la agricultura.

Otro sistema para IoT se trata en [39], donde se utiliza una combinación de *hardware* y *software* para realizar el monitoreo en línea de un entrenamiento de ciclismo. El sistema utiliza sensores que recolectan información acerca de la velocidad, cadencia, potencia y

ubicación del ciclista de manera que se puedan mostrar y administrar los datos en una aplicación móvil.

En el marco de las aplicaciones de SNMP, en [40], se analiza este protocolo como un estándar para la gestión de sensores en redes IoT domésticas, para que en cada momento se permita visualizar y gestionar la información recolectada por los sensores. Adicionalmente se implementa el control de fallas de los sensores, para así proporcionar una gestión del sistema confiable.

Se presenta una solución basada en SNMP para el monitoreo de infraestructura de redes dinámicas y virtualizadas [41]. Esta solución enfocada al cliente de la red permite acceder directamente a la infraestructura como si esta fuera un dispositivo genérico SNMP. Lo que hace posible detectar problemas en el desempeño de la red.

En [42] se resalta la importancia del monitoreo de dispositivos de red para detectar elementos que no estén funcionando correctamente. En concreto, el estudio propone una herramienta basada en el protocolo SNMP para obtener variables de los equipos las mismas que son almacenadas en una base de datos.

Se plantea el desarrollo de una API basada en el protocolo SNMP, debido a la necesidad de integrar y gestionar dispositivos IoT con dispositivos Android y Windows [43]. Esta API permite registrar dispositivos y obtener información a través de la red. Adicionalmente, se hace un análisis del retardo al momento de gestionar un dispositivo en distintos escenarios.

El desarrollo de IoT ha acelerado el despliegue de IPv6, sin embargo, existen dificultades como se presentan en [44], debido a que se necesitan técnicas para integrar los protocolos IPv4 e IPv6 ya que son incompatibles en el proceso de autenticación. Por lo tanto, mediante el uso de SNMP se obtienen las direcciones IP y MAC de dispositivos, logrando una mejoría en la compatibilidad para el sistema de autenticación.

En el ámbito de las redes de sensores masivas existen retos como se describe en [45] al desarrollar un sistema para identificar cada sensor usando SNMP. Esto se realiza al otorgar un OID a cada sensor, esta red se implementa en el sistema operativo Contiki y con la pila de protocolo de 6LoWPAN (*IPv6 over Low-Power Wireless Personal Area Network*). El nodo de administración se implementa en una distribución Linux, como resultado se muestra un manejo eficiente de los identificadores en este entorno dinámico.

Respecto al uso de alternativas al protocolo SNMP. Por la disponibilidad de diversas tecnologías o distintos protocolos de gestión en aplicaciones IoT tales como SNMP, Zabbix, MQTT, LoRaWAN, en [14], se realiza un análisis comparativo ente los distintos protocolos de gestión, en concreto MQTT, SNMP y Zabbix. Específicamente se analizan características como el consumo de la memoria y el consumo de energía. Entre los resultados del estudio se destaca que SNMP utiliza la menor cantidad de memoria tanto de RAM como de ROM, y en cuanto al consumo de energía no se presenta una diferencia significativa.

En [46] se utiliza la tecnología LoRaWAN (*Low Power Wide Area Network*) con el objetivo de realizar el monitoreo de la calidad del aire. En el estudio se destaca que el sistema es capaz

de obtener una tasa significativa de recepción de paquetes (72.4%). Esto muestra que este protocolo es una tecnología de comunicación aplicable al monitoreo de la calidad de aire, así como a otras aplicaciones y además resulta útil en áreas en las que se cuenta con conectividad limitada o Internet de baja velocidad de conexión.

Por otra parte, en [47] se realiza un análisis de los protocolos utilizados en la capa de aplicación de un sistema enfocado en una *Smart City*, para esto se examinan los siguientes protocolos: MQTT, CoAP y DDS. Como resultado, se muestra que de acuerdo a los requerimientos de cada aplicación se puede elegir un protocolo u otro, también se muestra que se debe tomar en cuenta el ancho de banda, latencia y la pérdida de paquetes. Así, por ejemplo, el protocolo MQTT resulta útil en un sistema de iluminación inteligente, ya que la pérdida de datos no afecta al rendimiento, el protocolo CoAP sería una mejor opción para las aplicaciones de agricultura en donde se requiere de precisión, ya que no presenta un retardo significativo que pueda poner en peligro los campos. Por último, el protocolo DDS sería una mejor opción en la construcción inteligente debido a que la información está en varios formatos, como binario, audio, video, texto.

En [48], se presenta una comparación de protocolos de configuración y administración para dispositivos IoT. El cual analiza características como energía, procesamiento y latencia. De esta manera, se muestra que a pesar de la antigüedad del protocolo SNMP, es una solución madura y estandarizada que puede ayudar a una mayor interoperabilidad. Por otra parte, también cuenta con control de seguridad y privacidad lo que le hace competitivo con protocolos nuevos como CoMI y LwM2M que están surgiendo como nuevas opciones de protocolos de gestión para dispositivos IoT.

En cuanto al estado actual de las redes Ad Hoc multi-salto, el concepto ha estado en el medio durante los últimos veinte años según [49], en la última década se ha dedicado mucha investigación a esta área. El concepto de multi-salto se aplica cuando no existe un camino directo de comunicación entre nodos, por lo tanto, los nodos intermedios actúan direccionando el tráfico. Este concepto ha sido aplicado en una variedad de tipos de redes como se presenta a continuación.

Las *Wireless Ad Hoc Network* se despliegan en casos que los nodos centrales no son fiables, por ejemplo, en casos de desastres naturales o aplicaciones militares. En otra categoría las MANET (*Mobile Ad Hoc Network*) son redes que necesitan del proceso de autoorganización debido a la movilidad de los nodos. Estas redes se aplican en entornos que carezcan de infraestructura, entre los que se encuentran: los servicios de emergencia, en ambientes comerciales, en redes vehiculares, aéreas y peatonales. En cuanto a las WMN (*Wireless Mesh Network*) son redes que se interconectan con una topología de malla, es decir, hay varias conexiones entre los nodos. Suelen ser estáticas para que la red pueda converger y enviar datos, estas tienen aplicación en Voz sobre IP (VoIP), aplicaciones militares, domótica y medidores eléctricos. A continuación, con las WSN (*Wireless Sensor Network*) se refieren a un grupo de sensores dispersos dedicados al monitoreo y recolección de datos como temperatura, sonido, humedad, entre otros. Cooperativamente estos pasan los datos entre los nodos y las más modernas son bidireccionales, es decir, tienen la capacidad de deshabilitar

los sensores. Además, pueden aplicarse en ámbitos como la salud, localización de personas, control industrial y automatización, seguridad, agricultura, entre otros. Por último, las VANET (*Vehicular Ad Hoc Network*) son un caso particular de las MANET, se crean redes espontáneas en el dominio vehicular, estas se utilizan para informar de accidentes, tráfico y seguridad del usuario.

A continuación, en el Capítulo 4, se detalla la arquitectura del sistema de gestión implementado en el presente trabajo de titulación.

Capítulo 4: Arquitectura del sistema de gestión

En este capítulo, en primer lugar, se describe el sistema de gestión implementado el cual se basa en el protocolo de capa de aplicación SNMPv3. A continuación, se detalla el desarrollo de la aplicación con el *framework* Node-RED. Dicha herramienta emplea un esquema de programación basada en flujos y permite crear la interfaz de usuario, la misma que en este caso tiene por objetivo realizar el monitoreo de las variables y sensores de un nodo. Por último, se describe la configuración de la red Ad Hoc que permite la conexión entre los nodos y de un *gateway* por defecto que permite la conexión de la red hacia a Internet.

4.1 Descripción de la Arquitectura

El sistema de gestión cuenta con dos elementos principales: el dispositivo administrador o gestor y el agente o dispositivo gestionado. La comunicación entre dichos elementos se realiza mediante un enlace inalámbrico en una red configurada en modo Ad Hoc. En tal sentido, para permitir el acceso a Internet, el dispositivo administrador cuenta con una interfaz adicional configurada como *gateway* por defecto. En la Figura 10, se presenta un diagrama funcional de la arquitectura de gestión. En particular, el sistema consta de cuatro nodos, de los cuales se dispone de dos nodos físicos implementados y dos nodos virtualizados los cuales se detallarán en el Capítulo 6, dejando este capítulo a la implementación física.

- Una plataforma Raspberry Pi 4, la cual actúa como agente que se encarga de administrar los objetos de la MIB tales como los recursos del nodo (el uso de la memoria, la temperatura del nodo y la carga del CPU) y las mediciones obtenidas por los sensores (geolocalización, temperatura, humedad, contaminación y luz). Este nodo se encarga de realizar el proceso de lectura o escritura en la MIB de acuerdo a lo requerido por el gestor.

- El ordenador, el cual consiste en el nodo administrador que realiza el proceso de enviar consultas hacia el agente y recibir las respuestas o notificaciones generadas por el mismo. En este elemento se encuentra la interfaz de usuario para el monitoreo y control.

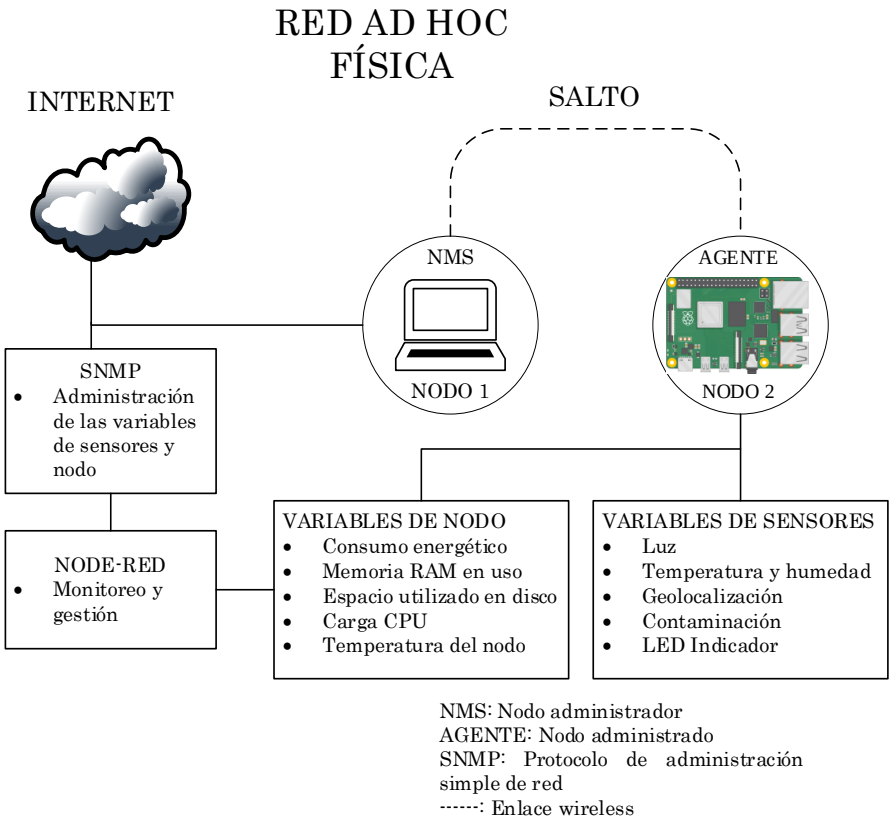


Figura 10: Arquitectura del sistema de gestión

4.2 Instalación y configuración del protocolo SNMP

En la Figura 11, se presenta un diagrama esquemático para la instalación del protocolo SNMP sobre el nodo administrador y sobre la plataforma Raspberry Pi, como se puede apreciar la principal diferencia es el paquete o librería disponible ya que en el caso del administrador permite realizar consultas y en el caso del agente permite administrar la MIB. A continuación, se describe el proceso de configuración del agente SNMP.

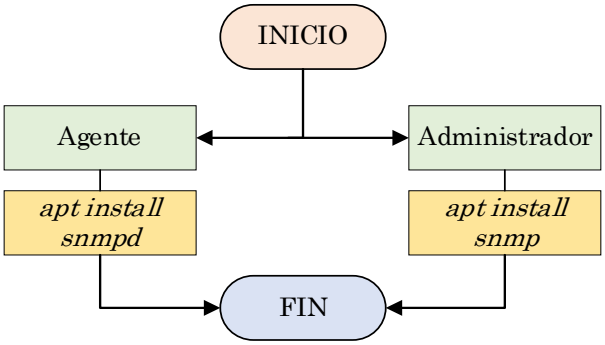


Figura 11: Instalación de SNMP en el nodo administrador y agente

4.2.1 Configuración del Agente

Una vez instalado el demonio SNMP en la Raspberry Pi se ingresa al directorio */etc/snmp* donde se encuentra el archivo para su configuración llamado *snmpd.conf.*.

Este archivo contiene varias funcionalidades ya que SNMP es un protocolo extenso, por lo tanto, a continuación, se detallan sólo las configuraciones necesarias para el funcionamiento de la aplicación. En la Figura 12, se presentan los pasos para realizar esta configuración, que a continuación se describen.

Se pueden recibir conexiones únicamente en el *localhost* o a través de las interfaces del agente, en este caso se requiere conexión por la interfaz Ad Hoc por lo que se debe configurar la opción que permite a través de las interfaces.

Por otra parte, es importante indicar que una de las mejoras de SNMPv3 se enfoca en la capa de seguridad, ya que añade autenticación y encriptación al tráfico. Específicamente permite tres opciones de seguridad:

- *noAuthNoPriv* (no autenticación y no privacidad).
- *authNoPriv* (autenticación y no privacidad).
- *authPriv* (autenticación y privacidad).

Adicionalmente, es posible configurar los siguientes mecanismos de autenticación:

- MD5: Es un algoritmo criptográfico de reducción de 128 bits.
- SHA: Es una familia de funciones que permiten realizar autenticación.

SHA es más robusto que MD5 y más rápido computacionalmente.

En cuanto a la encriptación también se tiene la opción de elegir dos algoritmos:

- DES: Es un algoritmo de clave simétrica.
- AES: Es el sucesor de DES, es al menos seis veces más rápido.

AES es la opción que debe utilizarse ya que otorga más seguridad. Estos mecanismos de encriptación y autenticación son la diferencia más notable con las versiones 1 y 2 en las que únicamente se requiere un nombre de comunidad para hacer solicitudes. Otro punto a considerar es que en estas versiones no se realiza encriptación del tráfico, por lo que son más vulnerables a ataques.

Por lo tanto, en función de estas capacidades de SNMPv3, se crea el usuario *pi* con clave de autenticación *123345678* y clave de encriptación *12345678.*

Para los usuarios creados se tiene la posibilidad de dar permisos de control de acceso de lectura mediante *rouser* o de lectura/escritura *rwuser.* En este caso se proporcionan permisos de lectura/escritura ya que también se realizan modificaciones en el agente. Para esto, se coloca el nombre de usuario *pi* y el nivel de seguridad.

Finalmente, para que las configuraciones realizadas se apliquen se debe reiniciar el demonio, para esto se utiliza el comando *sudo service snmpd restart.*

Para eliminar un usuario se debe detener el demonio, eliminar las directivas *CreateUser* del demonio y finalmente eliminar la línea correspondiente al usuario que se desea eliminar del archivo */var/lib/snmp/snmpd.conf*.

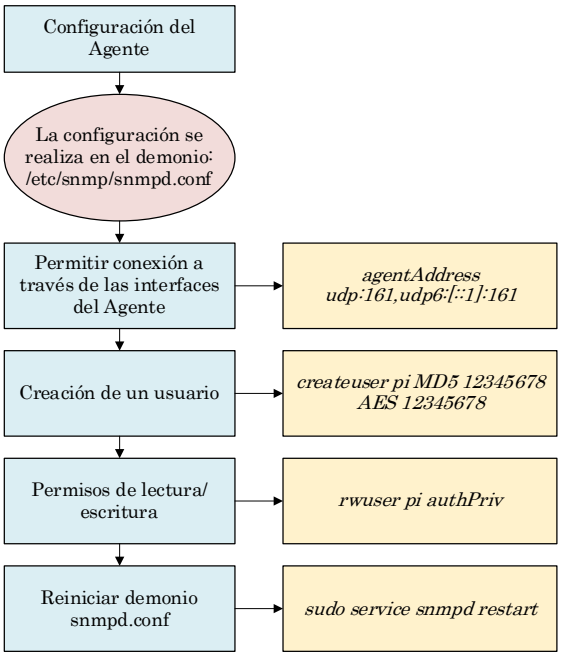


Figura 12: Configuración del demonio del agente

4.2.2 Configuración de mensajes

Una de las funciones de *snmpd* es la extensión del agente que permite ejecutar comandos externos o *scripts* de la *shell* de Linux, independientes de la operación de SNMP. Existen dos métodos para la ejecución de *scripts* con SNMP la primera con la ejecución arbitraria de un *script* y la segunda mediante la extensión de la MIB, ambos métodos se presentan en las secciones 4.2.2.1 y 4.2.2.2. Cada uno de estos métodos se detalla en el Apéndice H.

4.2.2.1 Ejecución arbitraria de un script

En la Figura 13, se muestra el proceso para la ejecución arbitraria de un *script*, el cual se encuentra representado por medio de un diagrama.

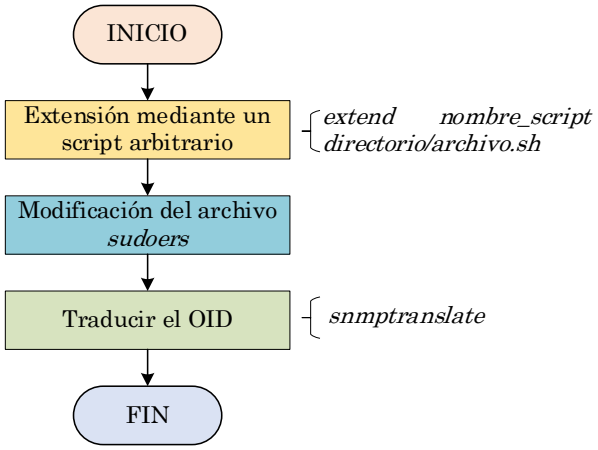


Figura 13: Extensión del agente mediante la ejecución arbitraria de un script

4.2.2.2 Extensión de la MIB

El diagrama mostrado en la Figura 14, presenta las etapas para realizar la extensión del agente SNMP por medio de la MIB.

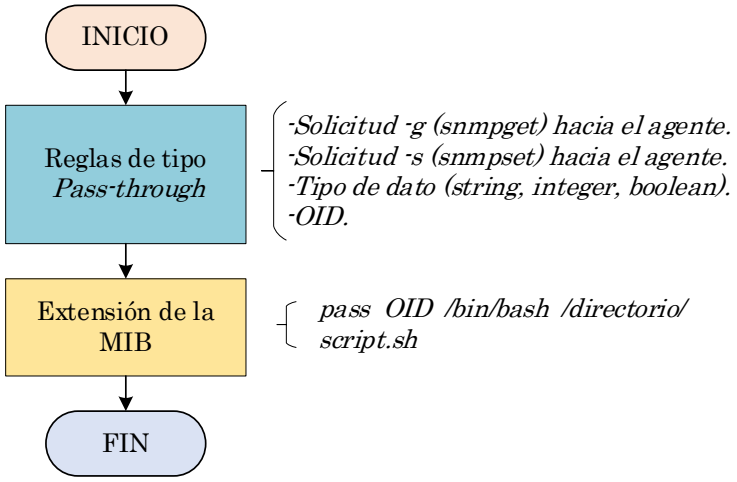


Figura 14: Extensión del agente mediante extensión de la MIB

En el Apéndice A se encuentran los *scripts* de configuración de las solicitudes: *snmpget* para la medición de temperatura del nodo, *snmpset* para el control de un LED indicador y la obtención de las variables de sensores.

4.2.3 Scripts de gestión

Los *scripts* de gestión implementados se dividen en dos partes: los *scripts* de medición y los *scripts* de gestión, en los de gestión se encuentran los de control de sensores y los de ahorro de energía.

Con el objetivo de obtener las mediciones de los sensores se implementaron *scripts* en Python, donde se configura el tiempo de muestreo, así como las configuraciones generales dependiente de cada sensor. Finalmente, los valores medidos son almacenados en archivos para posteriormente ser leídos por la aplicación.

Para todos los sensores se sigue la secuencia que se presenta en la Figura 15. En el Apéndice B se presenta la instalación de los sensores en el hardware Raspberry Pi.

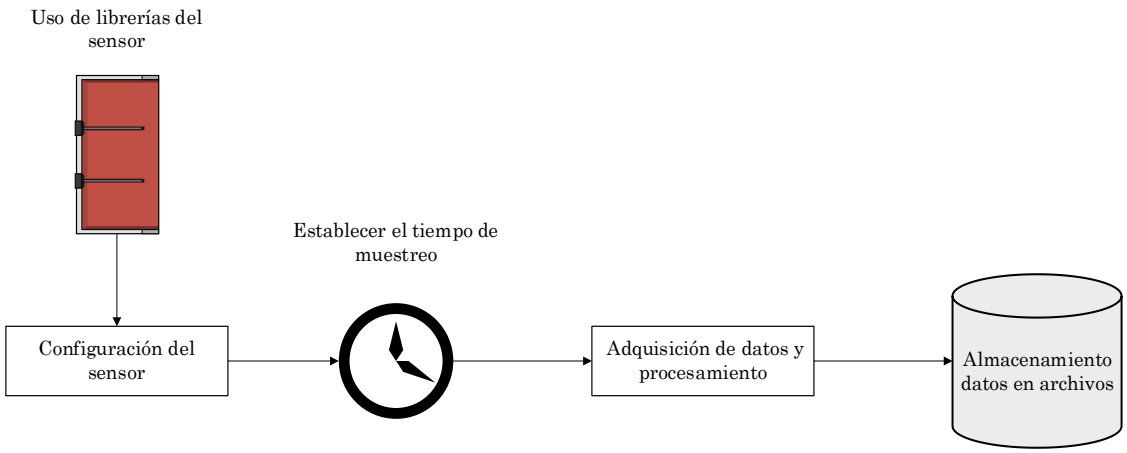


Figura 15: Flujo de configuración y medición de variables para los sensores

Los *scripts* de medición se ejecutan automáticamente al iniciar el nodo. Por lo tanto, para la gestión de los sensores se desarrollaron *scripts* para detener/reestablecer el proceso de medición de los sensores en caso de que no se requiera su uso. Para esto se identifica el proceso que está realizando la medición y se utiliza el comando *kill -STOP* para detener el proceso y *kill -CONT* para que se reanude. Estos *scripts* son ejecutados mediante SNMP por la aplicación.

En el caso que el nodo requiera ahorrar energía se puede activar el ahorro de energía mediante un *script* que consiste en cortar el suministro eléctrico de las interfaces de la plataforma como son: I2C, SPI, UART, HUB de puertos USB, módulo Bluetooth, puertos HDMI. Es importante resaltar, que dichas acciones se deben realizar sin perder conectividad para que la red siga activa con la capacidad de transmitir datos. Para realizar esta configuración se modifica el archivo */boot/config.txt* de Raspberry Pi que permite activar o desactivar dichas interfaces.

Este modo requiere reiniciar la plataforma al ser activado. Para reestablecer el funcionamiento normal se utiliza un *script* que enciende estas interfaces, también requiere un reinicio. De la misma manera se ejecuta a través de SNMP.

4.4 Desarrollo de la Aplicación de Gestión

4.4.1 El framework Node-RED

La herramienta en la que se desarrolla la aplicación de gestión es Node-RED, la cual está basada en Node.js y escrita en JavaScript. Posee módulos de entrada y salida para crear aplicaciones, en este caso la gestión de los nodos.

Existen varias maneras de instalar Node-RED, en la Figura 16, se muestra el proceso de instalación que puede ser de forma manual o automatizada mediante un *script*. Un análisis más detallado del proceso de instalación se presenta en el Apéndice C.

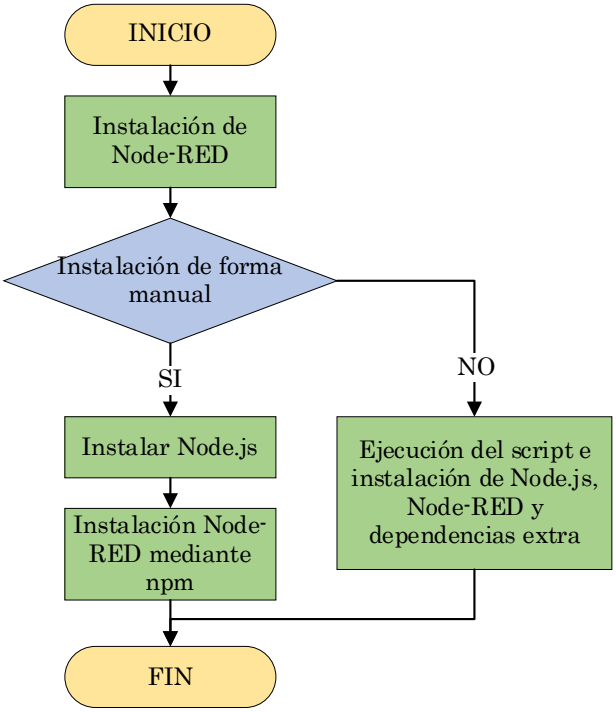


Figura 16: Proceso de Instalación de Node-RED

4.4.1.1 Ejecución de Node-RED

Con la instalación realizada se procede a ejecutar Node-RED, para lo cual se realizan los pasos mostrados en la Figura 17.

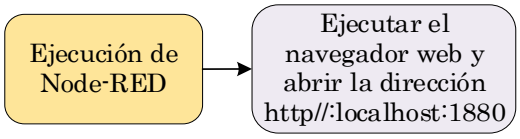


Figura 17: Pasos para la ejecución de Node-RED

Una vez que se lleva a cabo el proceso de ejecución, se observa la interfaz que permite trabajar con varios módulos para la programación, tal como se muestra en la Figura 18.

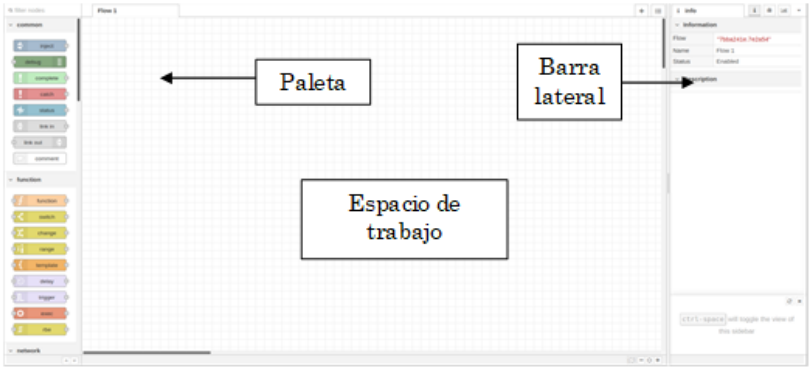


Figura 18: Interfaz de Node-RED

En la interfaz de programación se encuentran varios elementos, como son:

- Paleta: Contiene los nodos, cada nodo realiza una función específica.
- Espacio de trabajo: Sitio en donde se arrastran los nodos para realizar un flujo.
- Barra lateral: Presenta información relacionada al nodo, o al flujo realizado.

4.4.1.2 Integración de SNMP en Node-RED

A continuación, se resumen los principales pasos de todas las configuraciones de Node-RED y los detalles se encuentran en el Apéndice C. La aplicación Node-RED permite integrar varios módulos entre ellos el protocolo SNMP para lo cual se utiliza el gestor de paquetes de Node-RED y se llevan a cabo los pasos mostrados en la Figura 19.

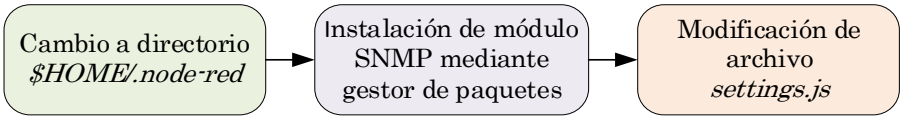


Figura 19: Procedimiento para la integración de SNMP con Node-RED

En la Figura 20, se observan las etapas para implementar un flujo que permita realizar consultas SNMP.

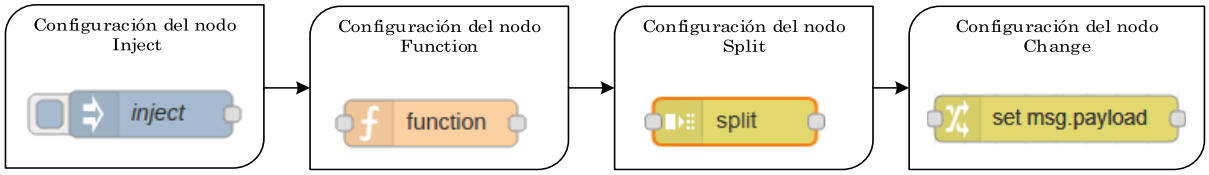


Figura 20: Diagrama de flujo general para consultas SNMP

De manera similar con el mismo flujo, se puede reemplazar el nodo *Function* y utilizar el nodo *Exec.* Este nodo permite ejecutar comandos de la *Shell* de Linux, de esta manera también es posible ejecutar directamente las peticiones del protocolo.

Para realizar la interfaz gráfica se debe instalar el módulo *Dashboard* que permite crear un diseño en varias pestañas con distintos tipos de gráficas como: en tiempo real, barras, gráficos circulares, entre otras maneras de presentar los datos.

Entre las principales partes de la interfaz existe:

- *Layout:* Es la pantalla donde se coloca la interfaz gráfica y su división se realiza en forma de malla.
- Barra lateral: Permite organizar las gráficas en pestañas, grupos y zonas.
- *Widgets*: Son las gráficas a presentarse, en las que ingresan y son mostradas en pantalla, además, posee varias características de personalización.

La instalación y configuración del módulo *Dashboard* consta de varias etapas. Una vez realizado este proceso, y al ejecutar un despliegue de la interfaz gráfica se obtiene el resultado de la Figura 21, donde se muestra el orden de las pestañas en la interfaz gráfica.

Figura 21: Pestañas de la interfaz gráfica de las funciones de la aplicación

En el módulo *Dashboard*, se cuenta con el nodo *Chart* que se utiliza para presentar el valor obtenido del OID en un gráfico cartesiano en tiempo real. En la Figura 23, se muestra el flujo para presentar gráficas cartesianas en la interfaz. Esto consiste en añadir este nodo al flujo de la Figura 22.

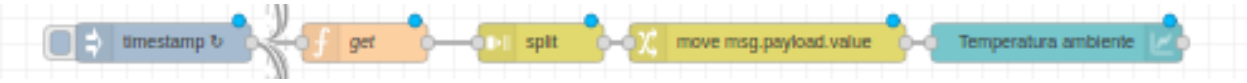


Figura 22: Diagrama de flujo con el nodo Chart

4.4.1.3 Configuración de botones para perfil energético y control de sensores

Se configuraron botones en la interfaz que permiten seleccionar un perfil energético en función a la necesidad del usuario, de la siguiente manera:

- Perfil 1: Todos los sensores encendidos.
- Perfil 2: El sensor de temperatura y luz encendidos.
- Perfil 3: Todos los sensores apagados.

En la Figura 23, se presenta la interfaz final para los botones. Así como en la Figura 24, se observa un mensaje que se presenta en pantalla al seleccionar el perfil con el tiempo estimado de duración de la batería.

Figura 23: Botones de perfil energético y ahorro de energía

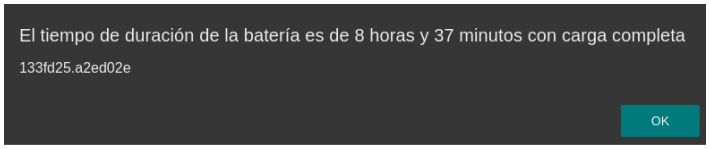


Figura 24: Mensaje con el tiempo estimado de duración de batería

De manera similar se realiza la configuración de botones para controlar el encendido y apagado de los sensores con los siguientes botones como se presenta en la Figura 25.

- Corriente Encendido/Apagado: Controla el sensor de corriente INA219.
- Luz Encendido/Apagado: Controla el sensor de luz.
- GPS Encendido/Apagado: Controla el sensor GPS *Breakout* v3.
- Temperatura Encendido/Apagado: Controla el sensor de temperatura y humedad.

- Contaminación Encendido/Apagado: Controla el sensor de contaminación.

Figura 25: Botones para el control de sensores

4.4.2 Conexión con IBM Cloud

La conexión entre la plataforma IBM Cloud y Node-RED se realiza mediante el módulo Watson IoT. Para lo cual se siguen las etapas mostradas en la Figura 26.

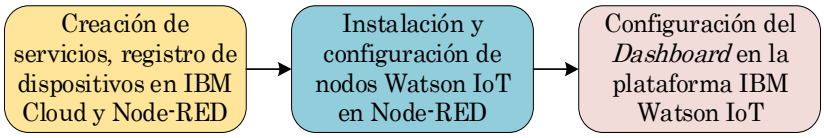


Figura 26: Etapas para la conexión de IBM Cloud con Node-RED

Cada una de las etapas mostradas anteriormente, se encuentra detallada en el Apéndice D.

Por medio de este proceso, se puede visualizar información proveniente del software Node-RED. Para la monitorización de las variables, se separa a las mismas en dos grupos relacionados a variables obtenidas por sensores y variables del nodo.

En consecuencia, se crea un panel de datos en el cual se especifique la agrupación y para cada una de las variables se crea un gráfico de datos y una tarjeta que muestre el valor recibido, como se puede observar en las Figuras 27 y 28, donde se presentan las gráficas y los valores.

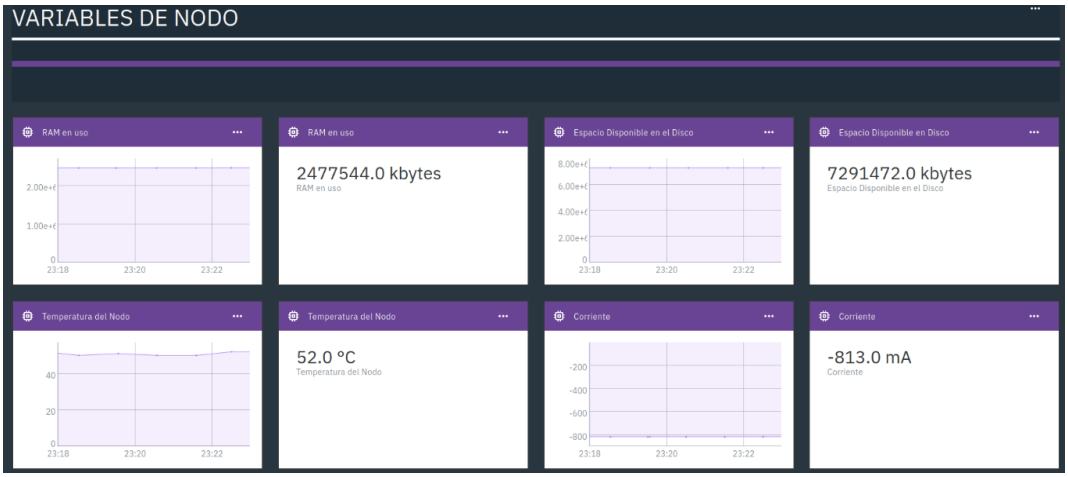


Figura 27: Panel de datos correspondiente a las variables del nodo

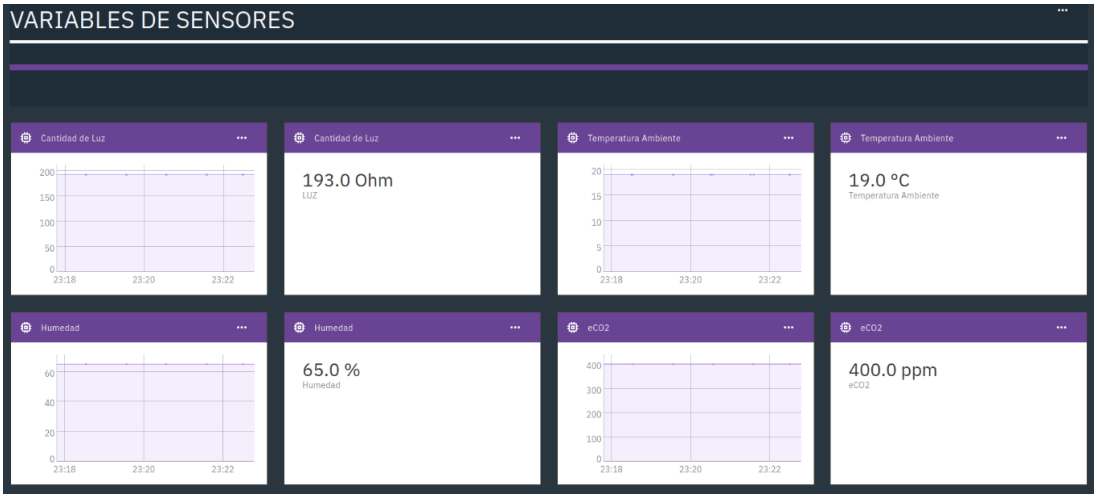


Figura 28: Panel de datos correspondiente a las variables de sensores

4.5 Configuración de la red Ad Hoc

En este punto se presentan los procesos para la configuración de la red Ad Hoc entre el ordenador y la Raspberry Pi. De la misma manera la configuración del *gateway* por defecto para la conexión a Internet. En el Apéndice I se presenta el código que permite estas funcionalidades.

4.5.1 Configuración del modo Ad Hoc en una distribución basada en Debian

La configuración de la red Ad Hoc física se lleva a cabo entre el ordenador con dirección IP *10.10.10.2* y la Raspberry Pi con dirección IP *10.10.10.3*, en el archivo */etc/network/interfaces.*

Para realizar la configuración de la red Ad Hoc se debe considerar el nombre de la interfaz en la que se quiere utilizar este modo. A continuación, se seleccionar una dirección IP y la máscara de red. Es necesario elegir un canal en ambos dispositivos que coincidan en la misma frecuencia y se recomienda elegir uno en el que no exista interferencia de otras redes cercanas. Es importante colocar un ESSID para la red y una contraseña para que no se conecten dispositivos externos. Finalmente, una vez que se guardan los cambios se debe reiniciar el servicio *Networking*.

Con el comando *iwconfig* se puede observar que la interfaz está en modo Ad Hoc y es importante corroborar que ambos deben estar en el mismo canal y frecuencia para que exista conexión. En la Figura 29, se presentan los pasos para realizar la conexión.

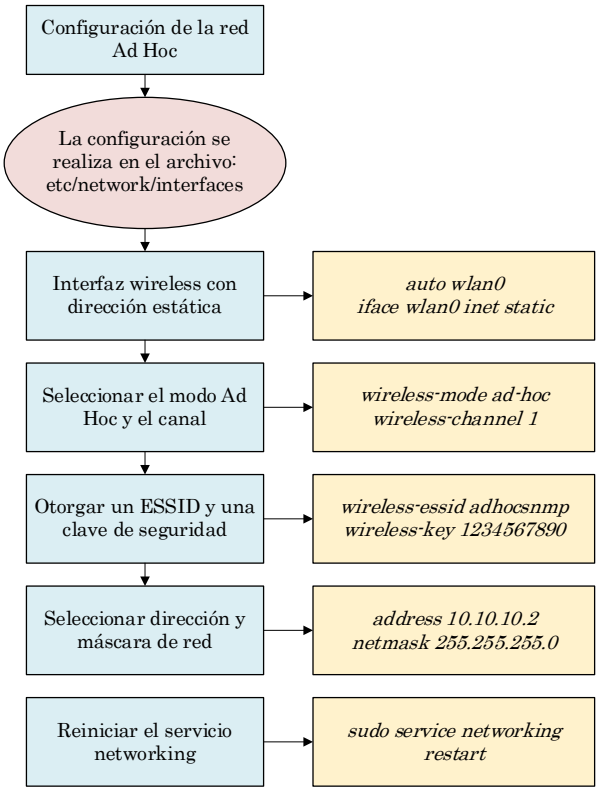


Figura 29: Configuración de la red Ad Hoc

4.5.2 Acceso a Internet mediante un gateway por defecto

La red permite la conexión a Internet mediante un *gateway* por defecto [50]. El objetivo es enviar el tráfico por la dirección 0.0.0.0 para esto se utiliza la herramienta *iptables* con el fin de establecer reglas para el ingreso y salida de paquetes en el nodo con salida a Internet.

En un primer paso se debe habilitar el *IP forwarding* para que los paquetes pasen a través de las interfaces de red.

La herramienta *iptables* permite al usuario configurar entre sus opciones el NAT (*Network Address Translation*) que es el que permite la salida a Internet, específicamente se configura *NAT MASQUERADE* cuya función es la de sustituir la IP origen con una IP pública dinámica, como es el caso de la interfaz eth0 del nodo. En el caso que esta sea estática se sustituye por *SNAT*.

NAT posee tres cadenas como son: *PREROUTING*, *OUTPUT* y *POSTROUTING*.

- PREROUTING permite modificar los paquetes entrantes antes del enrutamiento.
- OUTPUT permite modificar paquetes generados en el mismo dispositivo después de ser enrutados.
- POSTROUTING permite modificar los paquetes antes de que salgan del dispositivo.

En un primer paso se debe habilitar el *IP forwarding* para que los paquetes pasen a través de las interfaces de red. Para la configuración se aplican los siguientes comandos, en el caso de que sea una interfaz cableada con el nombre *eth0* se aplica una regla *POSTROUTING*: Consecuentemente, se realiza el *forwarding* de la interfaz cableada a la interfaz inalámbrica de la red Ad Hoc.

En el caso de que se utilice una red inalámbrica en lugar de una cableada para el *gateway* se debe cambiar el nombre de *eth0* al nombre de la tarjeta de red. Finalmente, se puede guardar las reglas creadas para que la configuración sea permanente. En la Figura 30, se presentan los pasos para realizar la configuración.

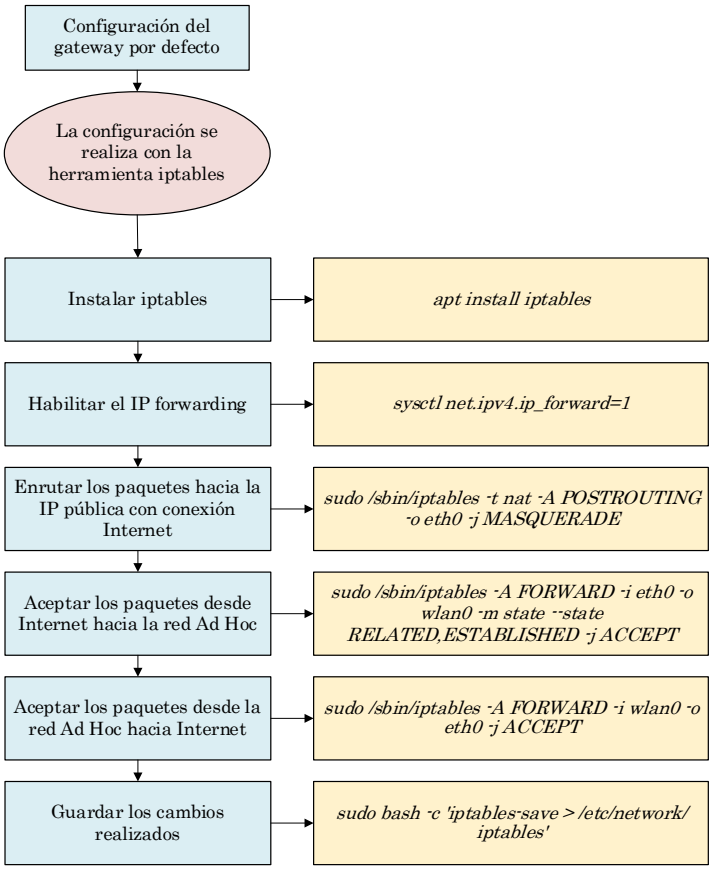


Figura 30: Configuración del gateway por defecto

4.6 Conclusiones

Tal como se trató en este capítulo la implementación del sistema de gestión la arquitectura tiene dos entidades principales: el Agente (Raspberry Pi) encargado de manejar los datos de la MIB y el Administrador (Ordenador) encargado de realizar consultas hacia el agente.

La instalación del *framework* Node-RED es relativamente sencilla si se utiliza el *script* de instalación, además se detalla la forma de integrar el protocolo SNMPv3 mediante funciones ya que no viene soportado por defecto. Esto es de suma importancia ya que este proceso es el núcleo de la aplicación.

La interfaz diseñada brinda muchas posibilidades de diseño y se adapta a los distintos dispositivos en los que se visualizan como: ordenadores y dispositivos móviles, ya que se lo realiza a través del navegador.

La arquitectura que consta de cuatro nodos, de los cuales se dispone de dos nodos físicos y dos virtualizados que se presentan en el Capítulo 6. Para la red Ad Hoc se debe tener en

cuenta el *IP forwarding* para que los paquetes pasen entre las interfaces de un mismo nodo y pueda salir por el *gateway* por defecto configurado con *iptables*.

Capítulo 5: Evaluación del software de gestión

En este capítulo se presentan las pruebas realizadas con el objetivo de verificar las funcionalidades del *software* de gestión implementado. Adicionalmente, se realiza una caracterización del rendimiento de los nodos en lo que respecta al consumo de energía, recursos computacionales y la información adquirida por los sensores. Finalmente, se verifica la funcionalidad del protocolo SNMP mediante la captura de tráfico empleando un analizador de red.

5.1 Escenario de pruebas

Las funcionalidades del sistema de gestión se verificaron sobre los escenarios indicados en las Figuras 31 y 32, como se puede observar estos se diferencian por el uso de un sensor de contaminación y de geolocalización respectivamente. Como se puede apreciar, en cada caso se analiza la aplicación con respecto a un nodo sensor. Cabe destacar que la comunicación entre el dispositivo administrador (ordenador) y el nodo se realiza mediante una red inalámbrica en modo Ad Hoc. En cuanto a las características de cada nodo, estas se especifican en la Tabla 8.

Nodo agente del escenario 1	Nodo agente del escenario 2
Sensor de luz (LDR)	Sensor de luz (LDR)
Sensor de temperatura y humedad (DHT11)	Sensor de temperatura y humedad (DHT11)
Sensor de corriente y voltaje (INA219)	Sensor de corriente y voltaje (INA219)
Sensor de contaminación (SGP30)	Sensor de geolocalización (GPS Breakout v3)

Tabla 8: Sensores instalados en las Raspberry Pi para cada uno de los escenarios

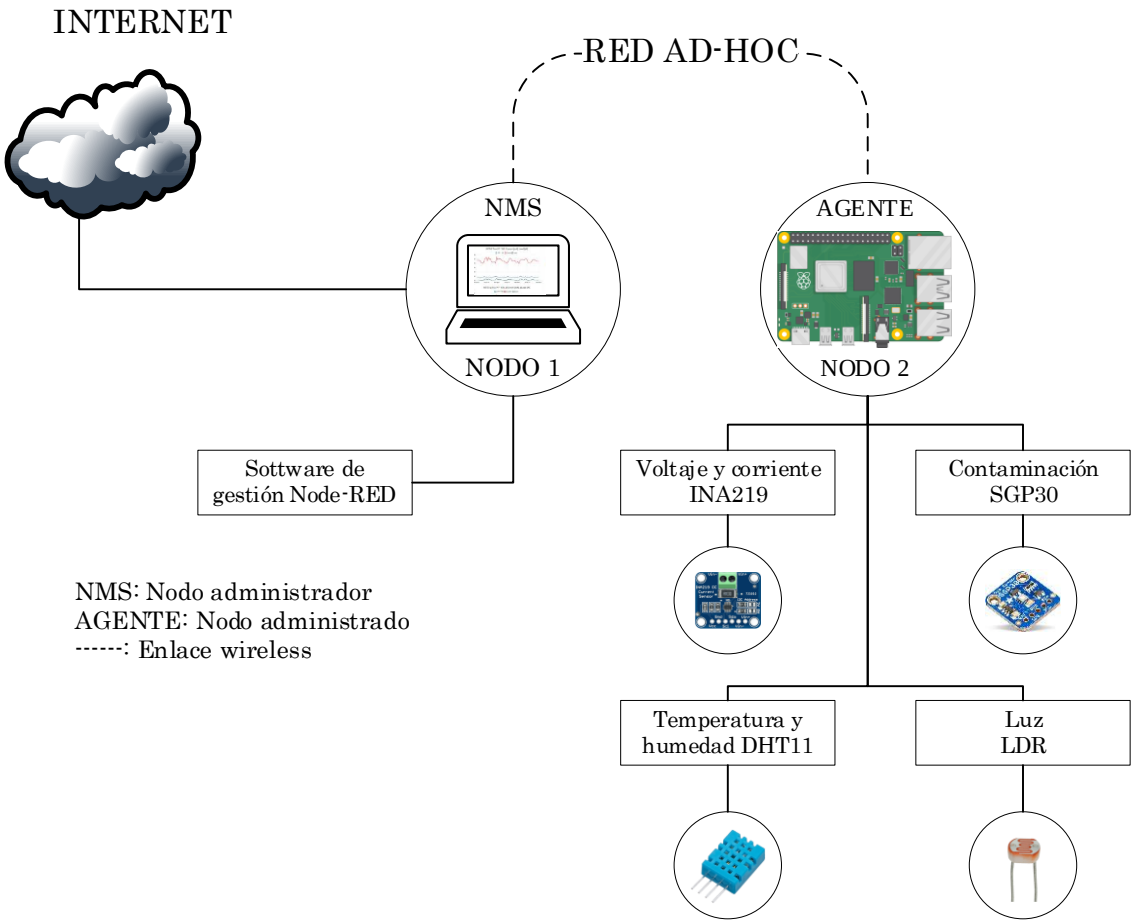


Figura 31: Escenario de pruebas 1 (Incluye sensor de contaminación)

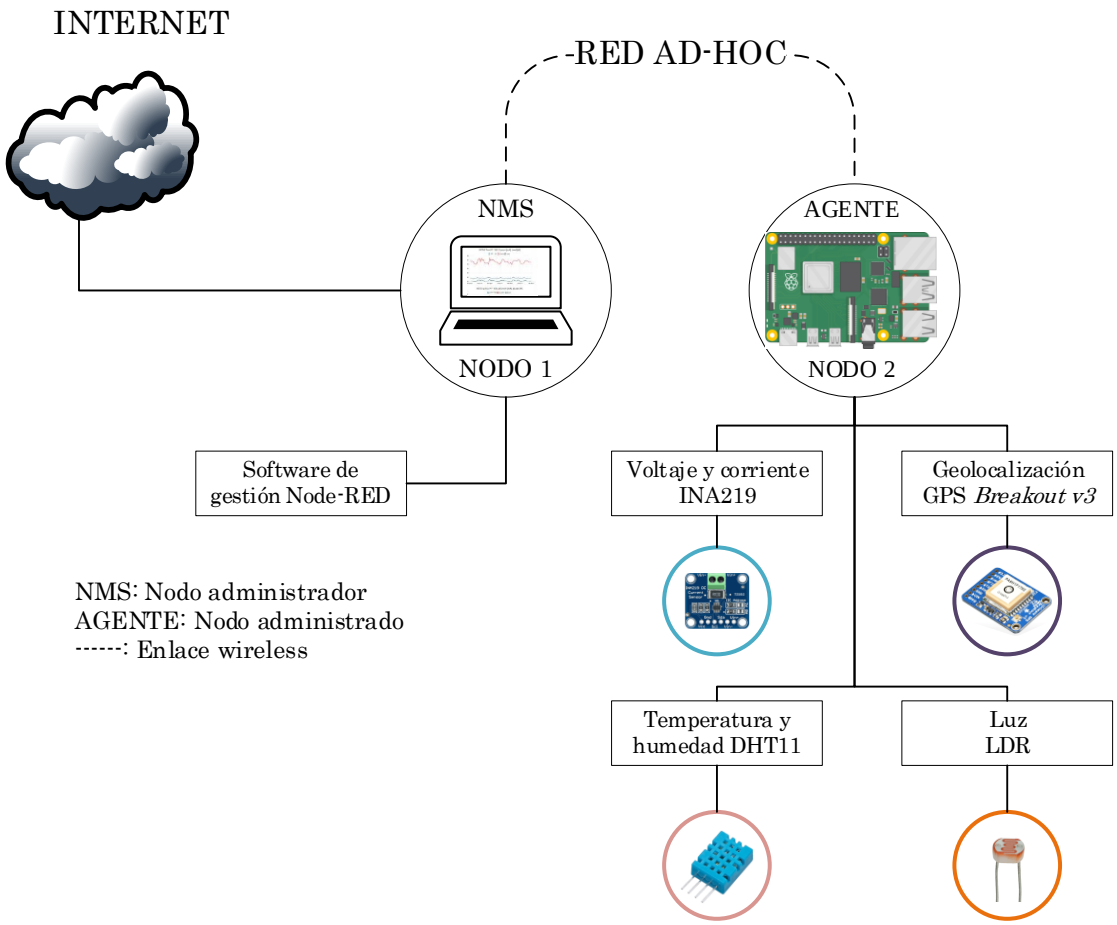


Figura 32: Escenario de pruebas 2 (Incluye sensor de geolocalización)

5.2 Evaluación de software de gestión

El software de gestión se muestra en un *Dashboard* implementado en Node-RED, al cual se accede a través de un navegador web ingresando a la dirección: *http://localhost:8080/ui*. Este software posee dos funcionalidades principales que son monitorización de variables y la gestión remota.

5.2.1 Monitorización de variables

La función de monitorización, se encuentra disponible en las pestañas de monitoreo de variables y geolocalización. En concreto, para mostrar la información relacionada a las variables de los sensores o del nodo se utilizan gráficos cartesianos, indicadores y un mapa.

La información presentada se actualiza cada 60 segundos ya que es el tiempo de adquisición de datos que se configuró en el diseño de la aplicación.

En cada gráfico, o indicador se presentan los datos obtenidos de la última hora antes que estos sean actualizados. Tanto el tiempo de adquisición de datos, como el tiempo de actualización de las gráficas es configurable acorde a la necesidad de un usuario.

En cuanto a los valores a ser monitorizados, estos se clasificaron en variables de los sensores, geolocalización y variables del nodo. En la primera agrupación correspondiente a las variables de sensores se encuentran:

- Nivel de luz.
- Temperatura ambiente.
- Humedad ambiente,
- Compuesto orgánico volátil total (TVOC).
- Dióxido de carbono equivalente (eCO2).

En la pestaña de geolocalización se presentan los valores obtenidos por el sensor GPS.

- Latitud.
- Longitud.
- Altitud.

El segundo grupo corresponde a las variables del nodo, en concreto:

- RAM en uso.
- Espacio utilizado en el disco.
- Carga del CPU.
- Temperatura del nodo.
- Corriente de la batería.
- Voltaje de la batería.

A continuación, en los siguientes apartados, se presentan los resultados obtenidos durante el funcionamiento de la aplicación, así como también se indica las características de los sensores disponibles en los nodos.

5.2.1.1 Sensor de luz (LDR)

El valor varía en función de la cantidad de luz que incide sobre el fotorresistor; las unidades en las que se mide esta variación son los Ohmios (Ω). Por tanto, es posible caracterizar la iluminación presente en un ambiente, ya que si el valor está en el orden de los megaohmnios el sensor está en la oscuridad, mientras que si está por debajo de los 4.000 hay presencia de luz. En la Figura 33, se muestra la monitorización de la variación de la resistencia en el sensor de luz.

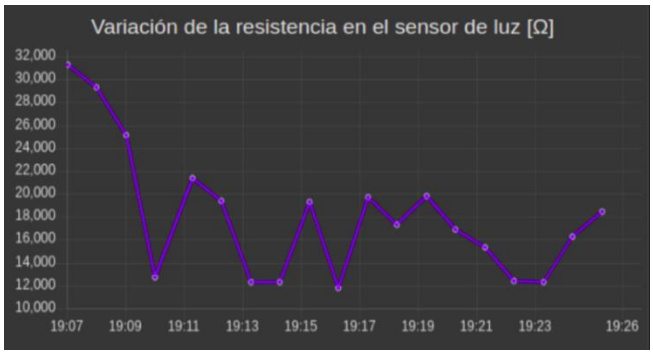


Figura 33: Resultado de medición sensor de luz

Adicionalmente, en la Tabla 9, se presenta la caracterización del sensor en distintos ambientes con las mediciones respectivas.

Rango de medición (Ω)	Ambiente
0 - 500	Día soleado
500 - 4000	Día nublado
4000 – 20.000	Atardecer
1.500 – 4.000	Luz artificial
200.000 en adelante	Oscuridad

Tabla 9: Caracterización del sensor de luz en diferentes ambientes

5.2.1.2 Sensor de temperatura y humedad (DHT11)

La variable de temperatura se muestra en grados Celsius (°C). Los valores que puede tomar esta variable en la ciudad de Cuenca van desde los 7°C hasta los 23 °C según la época del año y la hora de medición. En la Figura 34, se observa el resultado de la monitorización en un lapso de tiempo correspondiente a una hora, y se puede verificar que no existe un cambio brusco en los valores mostrados.

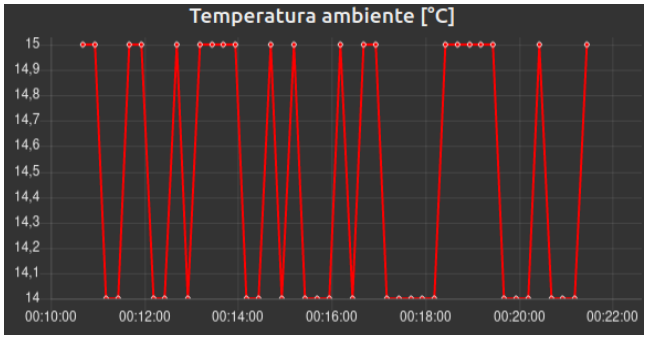


Figura 34: Resultado de monitorización de temperatura ambiente

El sensor DHT11 también permite adquirir información respecto a la presencia de vapor de agua en el aire, es decir la humedad relativa. Este valor corresponde a la humedad de una masa de aire con relación a la máxima humedad que podría soportar la misma sin producirse condensación y se presenta como porcentaje (%). Para el caso de la ciudad de Cuenca, varía entre 40 % y 70 %. La Figura 35, muestra el proceso de monitorización en una hora del día en la que se comprueba una variación existente en el lapso de tiempo medido.

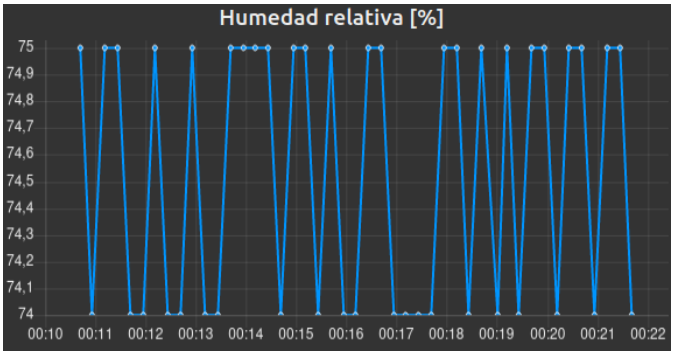


Figura 35: Resultado de monitorización de humedad ambiente

5.2.1.3 Sensor de contaminación (SGP30)

El sensor de contaminación SGP30 permite detectar un compuesto orgánico volátil o TVOC en el ambiente. Los resultados se expresan en unidades de partes por billón (ppb). Para observar un cambio en la variable, en este caso el sensor fue expuesto a la presencia de alcohol, con lo que se presenta el resultado mostrado en la Figura 36, donde se puede apreciar una variación en el rango de 0 a 60.000.

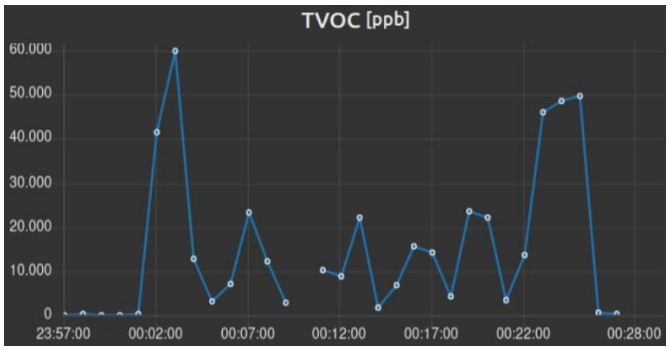


Figura 36: Resultado de monitorización de total de compuestos orgánicos volátiles

Adicionalmente, el sensor SGP30 puede ser empleado para detectar la concentración de eCO2 en el ambiente. De forma similar la información se obtiene en partes por millón (ppm). Los valores que puede tomar la variable y que se muestran en la gráfica de datos van desde los 400 a 60.000 ppm. Al igual que la variable de TVOC, si se desea obtener una variación se puede utilizar alcohol cerca al sensor, el resultado de esta experimentación se observa en la Figura 37.

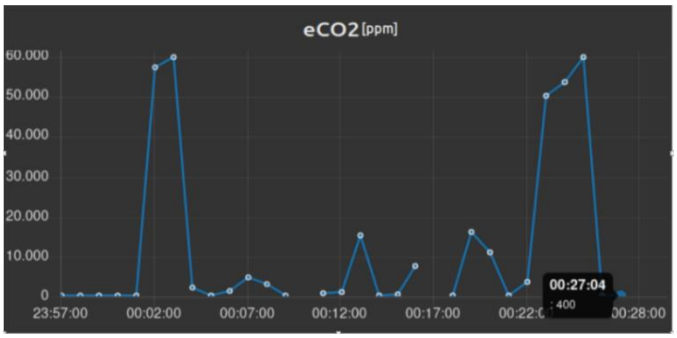


Figura 37: Resultado de monitorización del equivalente de dióxido de carbono

5.2.1.4 Sensor de geolocalización (GPS Breakout v3)

El sensor de geolocalización tiene la capacidad de hacer un seguimiento de hasta 22 satélites con una precisión menor a 3 metros. Además, posee una antena parche integrada y también permite la conexión de una antena externa. En la Tabla 10, se presenta los resultados obtenidos acorde a la ubicación del nodo durante la prueba.

Latitud	-2.905892
Longitud	-79.050725
Altitud	2654,7 metros sobre el nivel del mar

Tabla 10: Datos del sensor de geolocalización

En cuanto a la visualizacion en el software de gestión, en este caso los datos se ingresan a OpenStreetMap para ser presentados gráficamente, lo que resalta la precisión del sensor ya que muestra donde está exactamente el nodo. De igual manera, se obtiene la altitud de 2654,7 m.s.n.m que es la altura en el sector de Misicata, parroquia Baños. (Cuenca – Ecuador).

Finalmente, al seleccionar sobre el nodo se presenta información que puede ser mostrada al usuario como la memoria RAM en uso y la temperatura interna del nodo gestionado. En la Figura 38, se muestra el resultado correspondiente al mapa y en la Figura 39, la altitud a la que se encuentra el nodo.

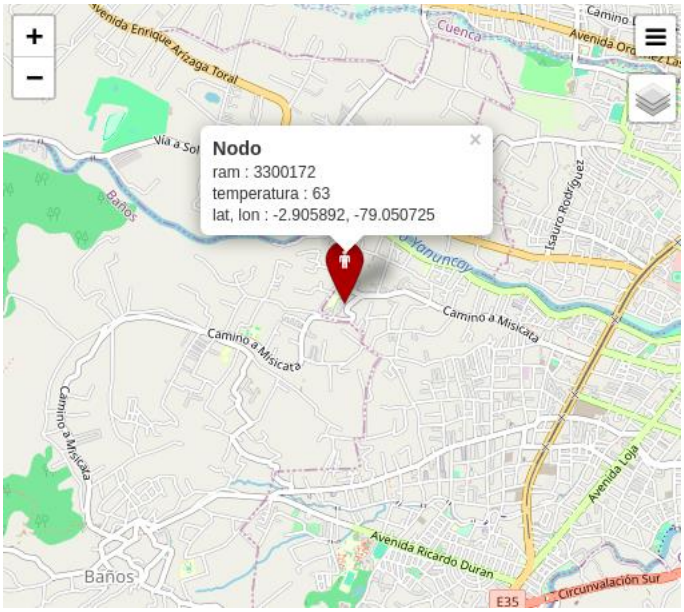


Figura 38: Mapa de geolocalización con información del nodo

Figura 39: Altitud del nodo en metros sobre el nivel del mar

5.2.2 Gestión y configuración remota

Las funcionalidades de gestión y configuración remota de los nodos, se encuentran disponibles en las pestañas de Selección de Perfil y Control de Sensores. En particular, se implementaron diferentes opciones como el encendido y apagado de sensores, así como la elección del perfil de consumo energético. La decisión de elegir un perfil u otro se considera en base a la monitorización del consumo del CPU, el uso de la memoria, la temperatura del nodo y el consumo energético. Cabe indicar que los nodos implementados disponen de baterías de litio de 5000 mAh.

5.2.2.1 Perfiles de energía y consumo energético (corriente y voltaje)

Los perfiles que se pueden seleccionar poseen diferentes características en las que se busca una disminución en el consumo de corriente, como se presenta en la Figura 40.

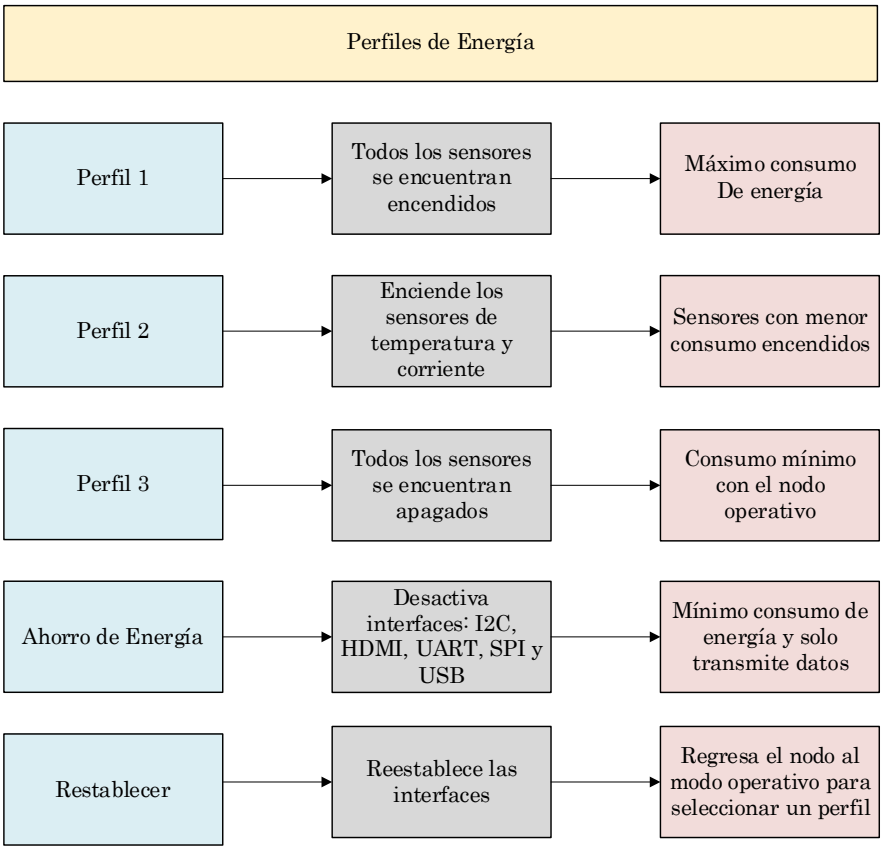
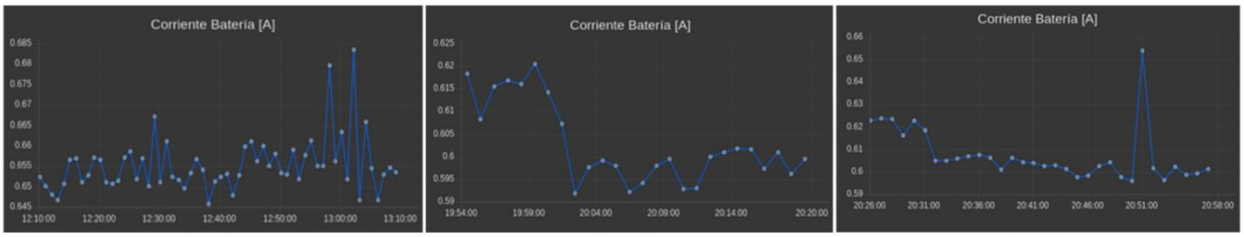


Figura 40: Perfiles de energía y modo de ahorro

La selección de un perfil u otro influye directamente en la duración de la batería, ya que para un menor consumo de energía se tendrá un mayor tiempo de duración, mientras que un alto consumo energético reducirá significativamente el tiempo de uso de la batería.

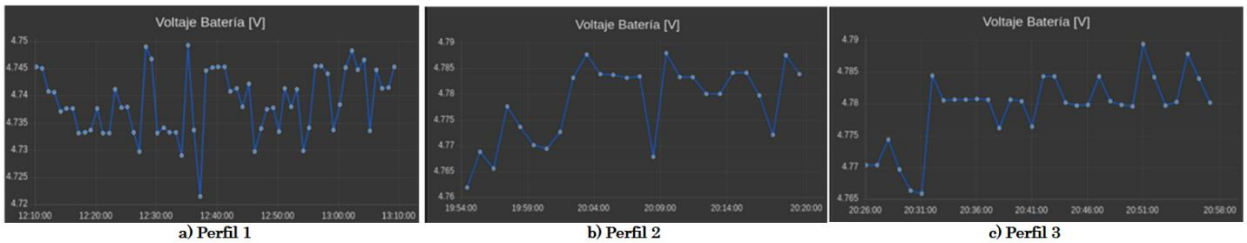
El consumo de corriente medido en amperios (A) depende del consumo generado tanto por el propio nodo como por los sensores conectados al dispositivo. Este valor se relaciona con la duración de la batería. En cada perfil se tiene un consumo diferente, así para el perfil 1 Figura 41.a, los datos toman valores entre los 680 mA y 640 mA. Para el perfil 2 Figura 41.b el consumo toma valores entre 620 mA y 590 mA. Finalmente, para el perfil 3 Figura 41.c, existen variaciones 620 mA y 580 mA. Esto demuestra que el consume de corriente varía de acuerdo a las funcionalidades activas en cada perfil.



a) Perfil 1 b) Perfil 2 c) Perfil 3

Figura 41: Resultado de consumo de corriente a) perfil 1 b) perfil 2 c) perfil 3

El voltaje se mantiene relativamente constante en las baterías de Litio hasta que disminuye cuando está cerca a agotar su carga. La Figura 42, muestra el proceso de monitorización de esta variable. Como se puede apreciar las variaciones de voltaje son mínimas obteniendo valores entre 4,72 V hasta 4,79 V para los tres perfiles.



a) Perfil 1 b) Perfil 2 c) Perfil 3

Figura 42: Resultado de medición del voltaje a) perfil 1 b) perfil 2 c) perfil 3

En cuanto al consumo energético, se realizó un proceso de censado de corriente durante un tiempo de 60 minutos, en cada uno de los perfiles. De esta manera se obtiene un consumo promedio mediante el cual se puede estimar la duración de la batería. Teniendo en cuenta que la capacidad de la batería es 5000 mAh, se utiliza la Ecuación 1 y se calcula el tiempo estimado de duración de la batería.

$$Tiempo\ estimado = Carga\ Batería\ /\ Consumo$$

Ecuación 1: Tiempo estimado de duración de la batería

Por medio de este análisis, se puede verificar que el consumo energético al realizar un cambio entre perfiles existe una variación del consumo de corriente promedio y el tiempo de batería estimado. Este resultado se presenta en la Tabla 11.

Perfiles	Consumo de corriente promedio (mA)	Tiempo de batería estimado (Horas)
Perfil 1	660	7.57 (7 horas y 34 minutos)
Perfil 2	607	8.23 (8 horas y 14 minutos)
Perfil 3	600	8.33 (8 horas y 20 minutos)
Ahorro de Energía	515	9.70 (9 horas y 42 minutos)

Tabla 11: Consumo de corriente promedio y tiempo de duración de batería para cada perfil

Para realizar la medición de consumo de energía de cada sensor se configuró el tiempo de muestreo cada 10 segundos. A continuación, se mide el consumo que genera, en la Tabla 12 se presentan los resultados.

Sensor	Consumo de corriente (mA)
LDR	4mA
DHT11	5mA
GPS Breakout v3	50mA
INA219	10mA
SGP30	50mA

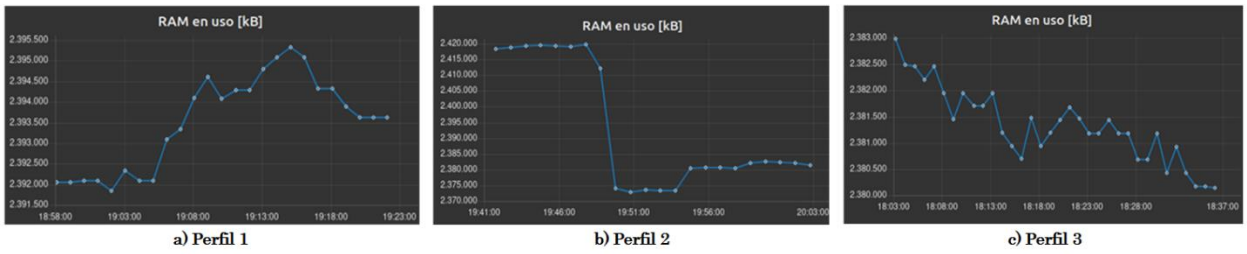
Tabla 12: Consumo de corriente en la adquisición de datos de cada sensor

5.2.3 Análisis del rendimiento de los nodos

5.2.3.1 Uso de memoria RAM

Se obtiene la cantidad de memoria RAM utilizada en un nodo, la cual se presenta en kilobytes (kB) y depende de los procesos o tareas que se encuentren en ejecución. Este valor depende de la cantidad total de memoria que se tenga disponible en el dispositivo, para el caso de la Raspberry Pi 4 se tiene un máximo de 4 GB.

La Figura 43, muestra el resultado de la monitorización para los distintos perfiles. Cada perfil muestra un comportamiento distinto debido a que cuentan con diferentes características de funcionamiento.



a) Perfil 1 b) Perfil 2 c) Perfil 3

Figura 43: Monitorización de la RAM en uso a) perfil 1 b) perfil 2 c) perfil 3

Por medio de las gráficas anteriores, se promedian los valores obtenidos en la monitorización y los resultados se muestran en la Tabla 13. De esta manera se puede verificar que mientras más procesos de censado estén ejecutándose, se tiene un mayor consumo de la memoria RAM, y cuando no existen procesos relacionados a los sensores se tiene un consumo mínimo de la memoria.

Perfiles	Memoria RAM en uso (kB)	Memoria RAM total (kB)
Perfil 1	2.393.530	3.884.368
Perfil 2	2.391.950	3 884.368
Perfil 3	2.381.740	3 884.368

Tabla 13: Memoria RAM en uso y total para cada perfil

5.2.3.2 Espacio utilizado en disco

El valor que se obtiene al monitorizar esta variable muestra información acerca del espacio utilizado en el disco medido en kilobytes (kB). De forma similar, este valor depende de la cantidad de almacenamiento disponible en el nodo que en el caso analizado es de aproximadamente 30 GB.

En la Figura 44, se muestra la monitorización para los diferentes perfiles. De cada grafica obtenida, se verifica que la variable tiene un comportamiento en el que el espacio utilizado en el disco aumenta si se utilizan más sensores.

a) Perfil 1 b) Perfil 2 c) Perfil 3

Figura 44: Monitorización del espacio utilizado en el disco para a) perfil 1 b) perfil 2 c) perfil 3

Por medio de los valores obtenidos en las gráficas anteriores se obtiene un promedio y se verifica que el uso de un perfil con más sensores encendidos implica un mayor uso del disco ya que los valores de las variables se almacenan en archivos, como se presenta en la Tabla 14.

Perfiles	Espacio utilizado en el disco (kB)	Tamaño total en el disco (kB)
Perfil 1	7.295.127	30.447.300
Perfil 2	7.295.124	30.447.300
Perfil 3	7.295.120	30.447.300

Tabla 14: Espacio utilizado en el disco y tamaño total para cada perfil

5.2.3.3 Carga del CPU

La monitorización de la carga del CPU indica la carga promedio que se ha tenido durante los últimos 15 minutos. El valor obtenido depende principalmente de los núcleos con los que cuente el dispositivo, se debe tener en cuenta que este valor es adimensional. En el caso de la Raspberry Pi 4 tiene cuatro núcleos por lo que el valor máximo que puede tomar esta variable es 4. Esto quiere decir que sus cuatro núcleos trabajan al 100%, un valor de 1 quiere decir que un núcleo está trabajando al 100%

La Figura 45, muestra la monitorización de esta variable para los distintos perfiles. Por medio de cada grafica obtenida se verifica que de acuerdo a la selección de un perfil se tiene una distinta carga del CPU lo que se debe a la cantidad de procesos activos en un perfil.

a) Perfil 1 b) Perfil 2 c) Perfil 3

Figura 45: Monitorización de la carga del CPU a) perfil 1 b) perfil 2 c) perfil 3

Al final con todos los valores obtenidos en cada perfil se realiza un promedio y se presenta el resultado en la Tabla 15, en la cual se visualiza que el cambio entre un perfil y otro afecta directamente al valor obtenido. Este resultado se debe principalmente, a que de acuerdo a la selección de perfil se tiene un número de procesos relacionados a la adquisición de datos de los sensores.

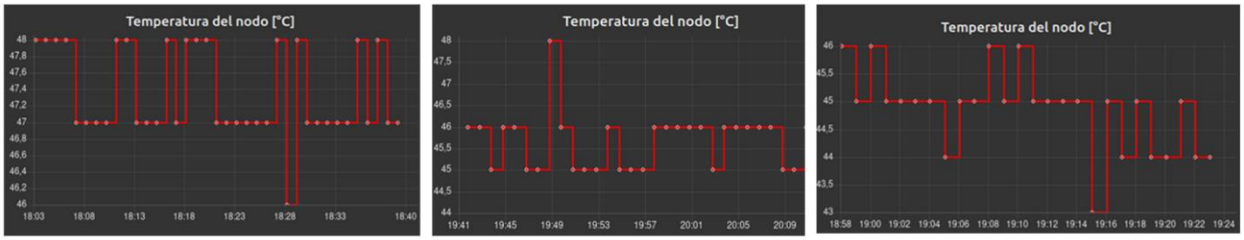
Perfiles	Carga CPU
Perfil 1	0,15
Perfil 2	0,0272
Perfil 3	0,0161

Tabla 15: Carga del CPU para cada perfil

Por medio del análisis de la carga del CPU, se puede concluir que el uso del software de gestión no implica un trabajo alto para los núcleos del procesador, ya que los valores obtenidos son inferiores a 1 (100% para un núcleo). Por lo tanto, al observar el máximo valor de carga se puede indicar que uno de los núcleos del CPU trabaja al 15 %.

5.2.3.4 Temperatura del nodo

Al monitorizar esta variable se obtiene la temperatura de la placa de la Raspberry Pi 4 la cual se mide en grados Celsius (°C) y varía de acuerdo a la utilización del dispositivo. Los valores que puede tomar esta variable oscilan entre los 40 °C y 65 °C. Se verifica que para el perfil 1 de la Figura 46.a, los valores obtenidos se encuentran en el rango de 46 °C a 48 °C. Por otra parte, el resultado de la monitorización para el perfil 2 de Figura 46.b, los valores están entre 45 °C y 48 °C. Por último, para el perfil 3 Figura 46.c, se obtienen valores entre 43 °C y 46 °C.



a) Perfil 1 b) Perfil 2 c) Perfil 3

Figura 46: Monitorización de la temperatura del nodo a) perfil 1 b) perfil 2 c) perfil 3

Se realizó un promedio de los resultados con el objetivo de establecer si los cambios entre perfiles intervienen en la temperatura de la placa. Los resultados se indican en la Tabla 16.

Perfiles	Temperatura del nodo (°C)
Perfil 1	47,27
Perfil 2	45,34
Perfil 3	44,75

Tabla 16: Temperatura del nodo para cada perfil

El resultado indica que el cambio entre un perfil u otro si ocasiona un cambio en la temperatura interna del nodo. Esto se debe principalmente, al uso de memoria y la carga del CPU ya que estas variables dependen de los procesos en ejecución.

5.3 Análisis de mensajes SNMP

Para realizar el análisis de mensajes SNMP, se realizó una captura de tráfico de la red utilizando la herramienta *Wireshark*. de esta manera se obtienen las tramas de los mensajes correspondientes a las consultas y respuestas generadas entre el agente y el administrador. En la Figura 47, se muestra una consulta *snmpget* realizada por la aplicación para obtener el valor de un OID específico.

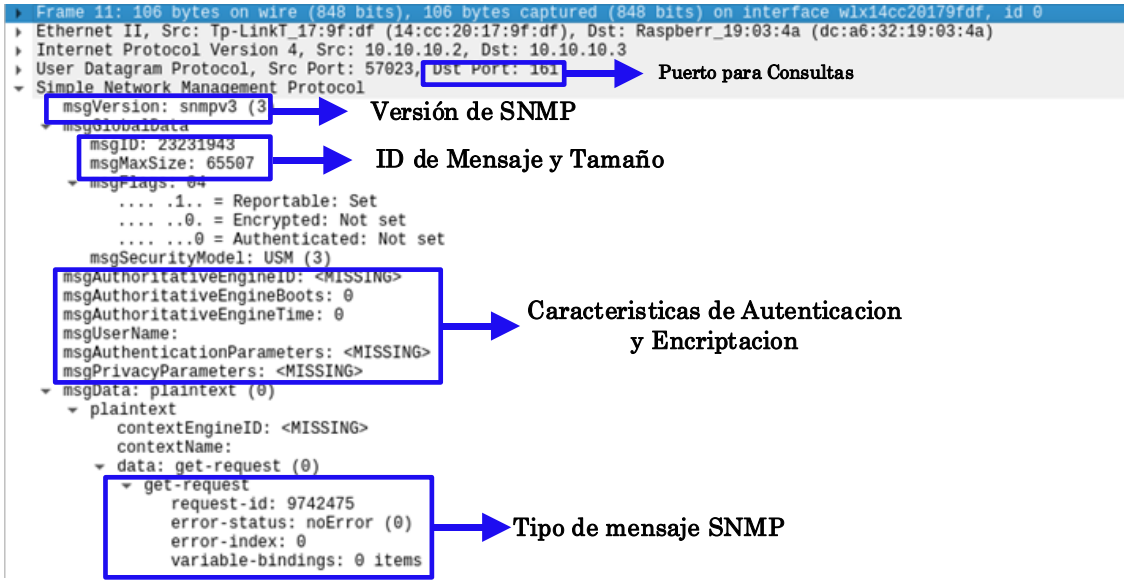


Figura 47: Mensaje capturado correspondiente una trama de consulta (get-request)

En esta se puede verificar por medio del protocolo IP que la solicitud se ha iniciado desde la dirección IP del administrador y tienen como destino la IP del agente. Además, por medio del protocolo de transporte UDP se puede ver que el puerto usado para recibir una consulta de tipo GET es el 161.

En cuanto al protocolo SNMP, se observa que se usa la versión 3. En este apartado también se indica el ID del mensaje, el tamaño máximo y las características de la trama de mensaje. Entre las características de la versión utilizada se encuentra la autenticación y encriptación. Para el mensaje analizado no se tiene un valor ya que se trata de una consulta.

Una característica importante es el tipo de mensaje que en este caso se trata de un *get-request*, y que no posee OID debido a las características de seguridad implementadas en esta versión.

Para cada consulta realizada desde el administrador hacia el agente, se genera una trama de tipo *report*, en la que se indica el número de paquetes recibidos por el motor SNMP. La trama que se captura se muestra en la Figura 48.

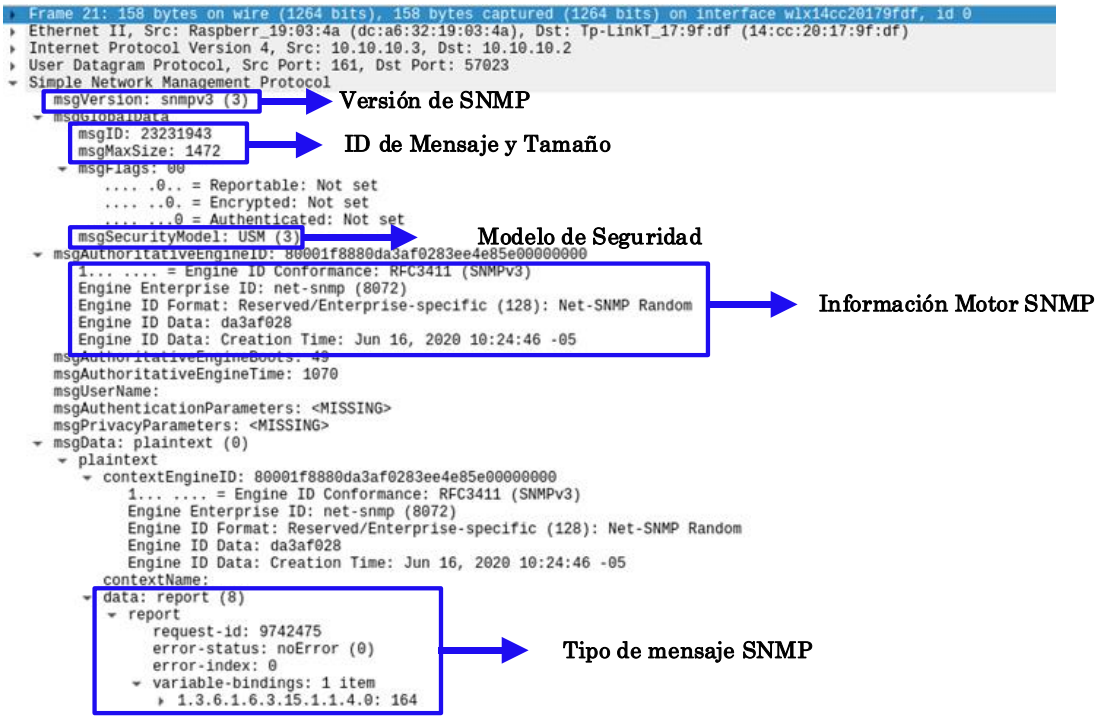


Figura 48: Mensaje capturado correspondiente una trama de reporte (report)

En esta trama al igual que en la del tipo *request*, se identifica la versión SNMPv3, además del ID y tamaño de mensaje. De la misma forma, se encuentra el modelo de seguridad utilizado que en este caso es USM. También se verifica la información del motor SNMP como son su ID, la fecha de creación, el formato, la empresa a la que pertenece, entre otros.

A continuación, muestra el tipo de mensaje que corresponde a un *report* el cual tiene su propio *request-ID*, OID y el valor que en este caso corresponde a 164 mensajes recibidos por el motor SNMP.

Finalmente, se obtiene la trama de respuesta que se envía desde el agente hacia el administrador en este caso la mayor parte de la información está encriptada. En esta trama se puede verificar que se utiliza la versión SNMPv3. De la misma manera que en las anteriores tramas se muestra el ID y tamaño máximo de mensaje, así como también el modelo de seguridad utilizado que en este caso corresponde a USM.

Al igual que en los anteriores paquetes capturados, en esta trama se obtiene información relacionada al motor SNMP en el cual se indica principalmente, el ID, la empresa, el formato, fecha de creación, etc. A diferencia de las otras tramas, en esta se indica el usuario que está en forma legible y las claves de autenticación y privacidad en forma encriptada. Al final, se tiene el tipo de mensaje que es el PDU encriptado y contiene la información del OID consultado. Todas estas características se pueden observar en la Figura 49.

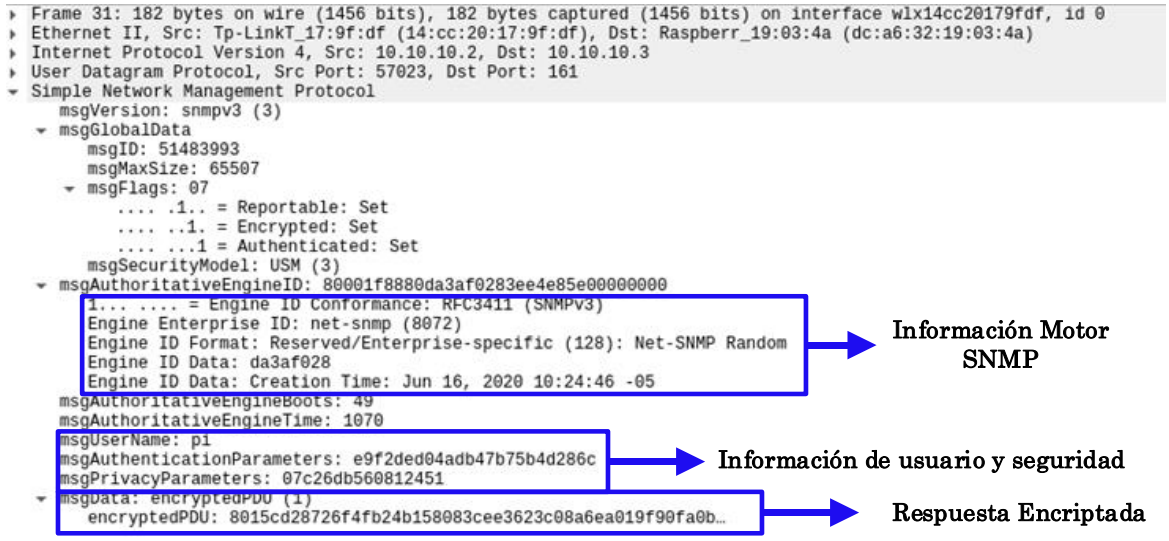


Figura 49: Mensaje capturado correspondiente una trama de respuesta encriptada

5.4 Conclusiones

En este capítulo se presentaron los resultados del uso del software en el que se verificó su correcto funcionamiento respecto al monitoreo y la gestión.

La aplicación al presentar las variables del nodo permite conocer el estado del hardware. De esta forma un usuario tiene la capacidad de efectuar tareas de gestión en caso de que se requiera además de seleccionar un perfil de energía, desactivar un sensor específico e incluso el activar el modo de ahorro de energía que permite únicamente transmitir datos, según sean las necesidades del usuario.

De la misma manera con las variables de los sensores se logró caracterizar el entorno en donde está colocado el nodo, esto se realizó de manera simple ya que el uso de gráficas en tiempo real facilitó dicha tarea.

Finalmente, se analizaron los mensajes de red utilizando *Wireshark* para comprobar la estructura de la versión 3 y que estos paquetes sean enviados por la interfaz Ad Hoc configurada. En la que se detalló todas sus partes corroborando la estructura de los mensajes presentadas en el marco teórico, en el Capítulo 2. A continuación, en el Capítulo 6, se presenta la evaluación del sistema de gestión operando sobre una red inalámbrica multi-salto, con tal finalidad se empleó la herramienta de emulación disponible en el simulador NS3.

Capítulo 6: Emulación de la red Ad Hoc multi-salto con NS3

En este capítulo se presenta la emulación de una red Ad Hoc multi-salto en el simulador NS3. Con la característica del uso de hardware virtualizado sobre contenedores Linux. En particular, se definen las arquitecturas sobre las cuales se realizó la emulación detallando el proceso para la instalación y configuración de los contenedores Linux sobre los que se implementa cada nodo.

Además, se describe el procedimiento para habilitar el hardware virtualizado sobre el simulador NS3. En cuanto a los escenarios para los experimentos, se trabajó sobre una topología lineal multi-salto con enrutamiento estático y posteriormente con enrutamiento dinámico mediante el protocolo OLSR (*Optimized Link State Routing*). Finalmente, se presenta un escenario que incluye un nodo móvil con enrutamiento OLSR.

Para finalizar se analizan las variables de red correspondientes a métricas de *throughput* o tráfico de la red, porcentaje de recepción de paquetes y el *delay* o retardo de la red. Estos resultados se presentan mediante gráficos de barras. Además de la medición del ancho de banda en la red multi-salto para lo cual se empleó la herramienta *iperf*.

6.1 Arquitectura de emulación

El proceso de emulación y configuración de cada uno de los nodos se realiza de la misma manera que en el hardware real, como fue descrito en el Capítulo 4. Se trabajó con dos arquitecturas presentadas en las Figuras 50 y 51, con el objetivo de demostrar las capacidades y opciones disponibles en el módulo de emulación NS3. Cabe recalcar que no se realiza una comparación entre ambas arquitecturas ya a nivel de funcionamiento del simulador no se hace ningún cambio en la capa física como se presenta en 6.3.

En la primera arquitectura se utiliza el ordenador como administrador en el Nodo 1 y se emplea la aplicación desarrollada en el Capítulo 4 para realizar las consultas a cada nodo gestionado. Los nodos 2 y 3 sirven de salto para llegar al Nodo 4 que se configura como agente la implementación se hace de manera virtualizada utilizando contenedores Linux. En la Figura 64, se presenta la arquitectura planteada.

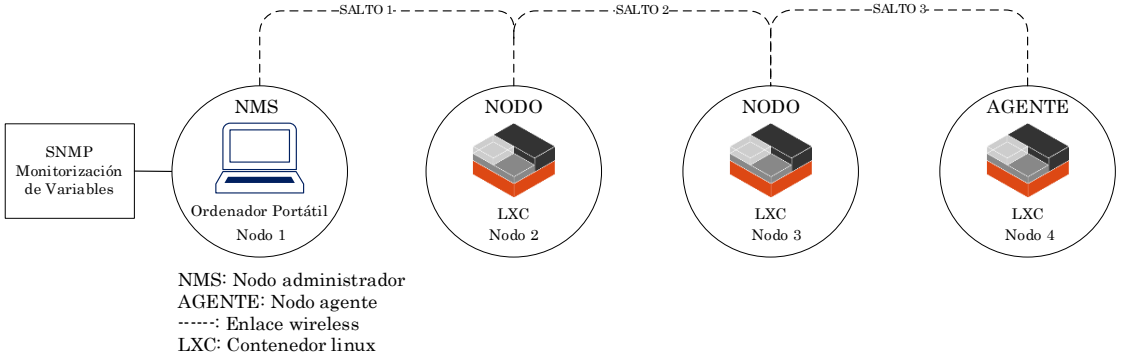


Figura 50: Arquitectura 1 de emulación

La segunda arquitectura se realizó utilizando cuatro nodos virtualizados en contendores Linux, de la misma manera el Nodo 1 se configuró como administrador SNMP simulando en este caso la aplicación desarrollada para hacer consultas SNMP. Los nodos 2 y 3 sirven de salto para llegar al Nodo 4 que se configura como agente. Como se observa en la Figura 65, los cuatro nodos conforman una red multi-salto.

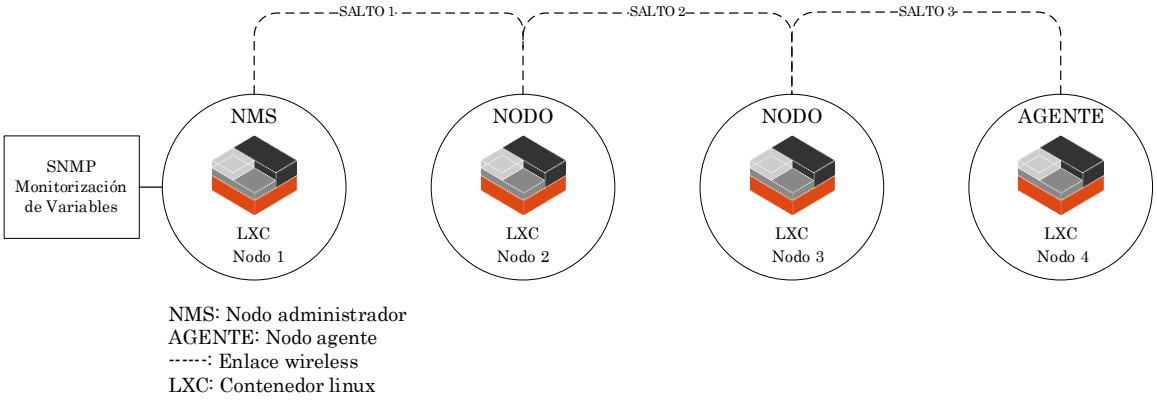


Figura 51: Arquitectura 2 de emulación

6.2 Configuración de contenedores Linux

Para la red multi-salto se utilizaron contendores Linux, la cual es una tecnología que permite empaquetar un sistema operativo y sus aplicaciones. En la Figura 52, se muestran las partes principales del proceso de instalación y configuración de contenedores Linux, en el Apéndice K se presenta detalladamente el proceso de: instalación, creación y habilitación de un contenedor Linux.

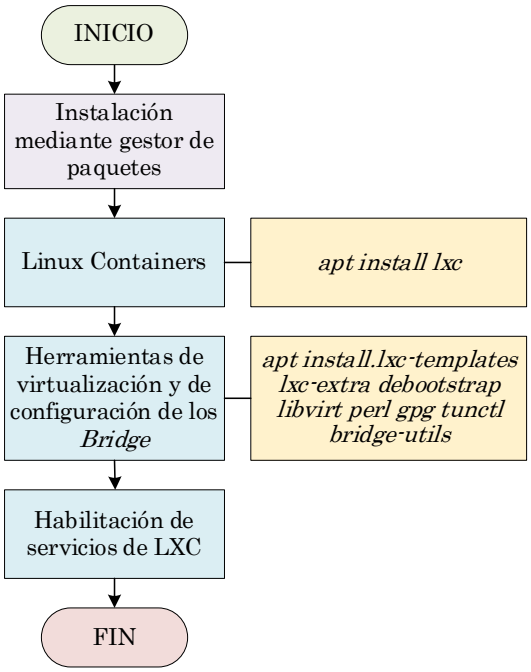


Figura 52: Proceso para la instalación y uso de los contenedores Linux

En la Figura 53, se observa el proceso para la creación de un contenedor, en el cual se debe seleccionar el sistema operativo a ser instalado, con su respectiva versión y arquitectura. Se seleccionó la distribución Debian Buster debido a que los sistemas operativos Ubuntu y Raspberry Pi OS que fueron utilizados en el Capítulo 4 están basados en la misma.

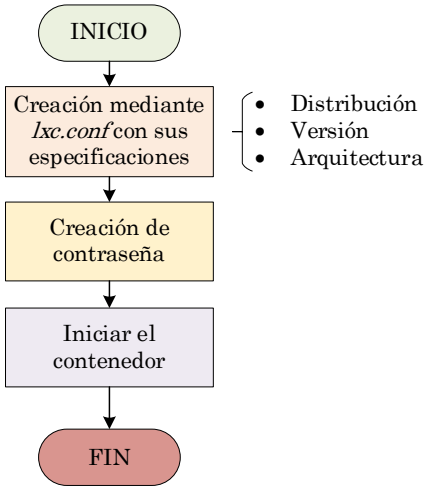


Figura 53: Proceso de creación de contenedor

Una vez creado el contendor, se procede a configurar los dispositivos *Bridge* y *TAP* como se esquematiza en la Figura 54, con la creación de un puente de red para el contenedor y la asignación de una dirección IP.

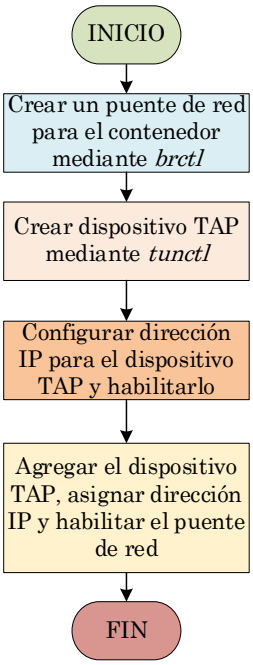


Figura 54: Proceso de configuración de dispositivos Bridge y TAP

6.3 Configuración de la herramienta de emulación

La herramienta NS3 permite definir una serie de parámetros, cuya configuración se presenta detalladamente en el Apéndice L. Inicialmente se configura el estándar 802.11 para la conexión inalámbrica entre nodos. Se utiliza el estándar 802.11n en la banda de 2.4 GHz con MSC (*Modulation Coding Scheme*) 7, los estándares 802.11ac y 802.11ax están todavía en desarrollo para la herramienta emulación con hardware.

Este estándar se caracteriza de la siguiente manera:

- *Stream* Espacial: 1
- Modulación: 64-QAM
- Tasa de codificación: 5/6

El siguiente parámetro de configuración es la movilidad de los nodos, estos se posicionan a una distancia de 23 metros entre sí, que es la máxima distancia obtenido antes de que la red empiece a ser inestable, es decir, exista pérdida de conectividad entre los nodos. Es importante indicar que el máximo rango de cobertura de 23 metros fue determinado mediante un experimento previo realizado en el simulador. En particular, se utilizó el modelo de posición constante ya que los nodos se encuentran estáticos en la topología lineal multi-salto.

Para utilizar los contenedores Linux con NS3 se aplica el modelo *tapBrigde,* el cual permite integrar hosts reales en las simulaciones como: los contenedores Linux, máquinas virtuales previamente configuradas o utilizar directamente el ordenador como si fuera un nodo.

Al ejecutar la simulación se puede variar el canal físico, utilizando el valor *$NS_GLOBAL_VALUE=RngRun=#Simulación*, si bien el canal ya es aleatorio por defecto, este parámetro asegura este comportamiento, es decir que cada experimento desarrollado tendrá características distintas con respecto al comportamiento del medio inalámbrico lo que permite una evaluación mucho más cercana a un entorno real.

6.4 Escenarios de simulación

6.4.1 Escenario 1: Red multi-salto con enrutamiento estático

En la Figura 55, se detalla la topología lineal de la red que está conformada de cuatro nodos y tres saltos. Se observa el administrador en el Nodo 1 y el agente en el Nodo 4. Además, se utilizó enrutamiento estático para la conexión entre los nodos.

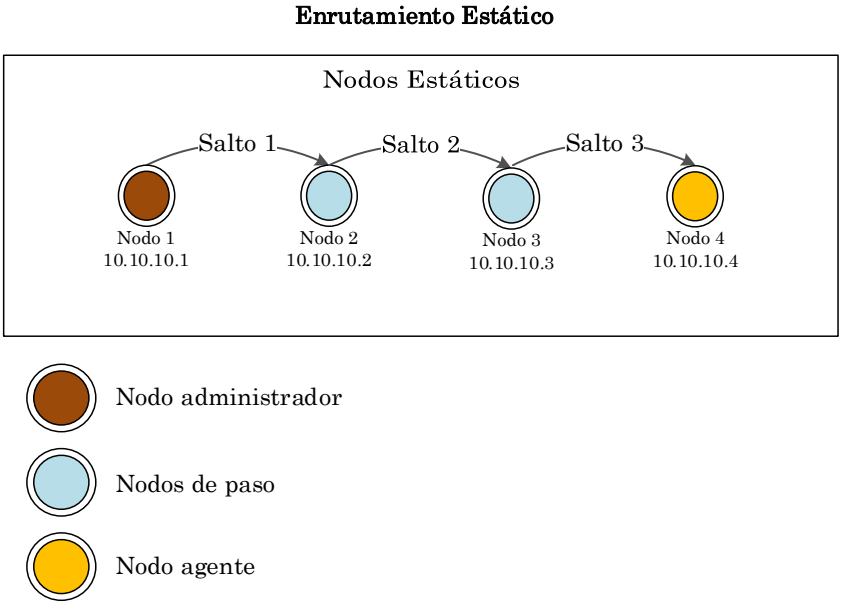


Figura 55: Escenario 1: Red multi-salto con enrutamiento estático

Para realizar la configuración de una red Ad Hoc multi-salto empleando rutas estáticas, en primer lugar, es necesario eliminar en cada nodo las configuraciones previas disponibles en las tablas de enrutamiento, para ello se puede emplear la instrucción *ip route flush dev eth1.*

Adicionalmente, es necesario activar el *flag IP forward*, ya que de esta manera los paquetes pueden pasar a través de las interfaces y así se pueden dar los saltos en la red. La configuración del enrutamiento para cada nodo se detalla en el Apéndice E. Finalmente, las rutas configuradas se pueden verificar al ejecutar la instrucción, *ip route show* o de forma alternativa utilizando *route –n.*

6.4.2 Escenario 2: Red multi-salto con enrutamiento OLSR

Para el caso de este escenario se mantiene la topología lineal como se observa en la Figura 56, sin embargo, en este caso se empleó un protocolo de enrutamiento OLSR (*Optimized Link State Routing*), en lugar de las rutas estáticas.

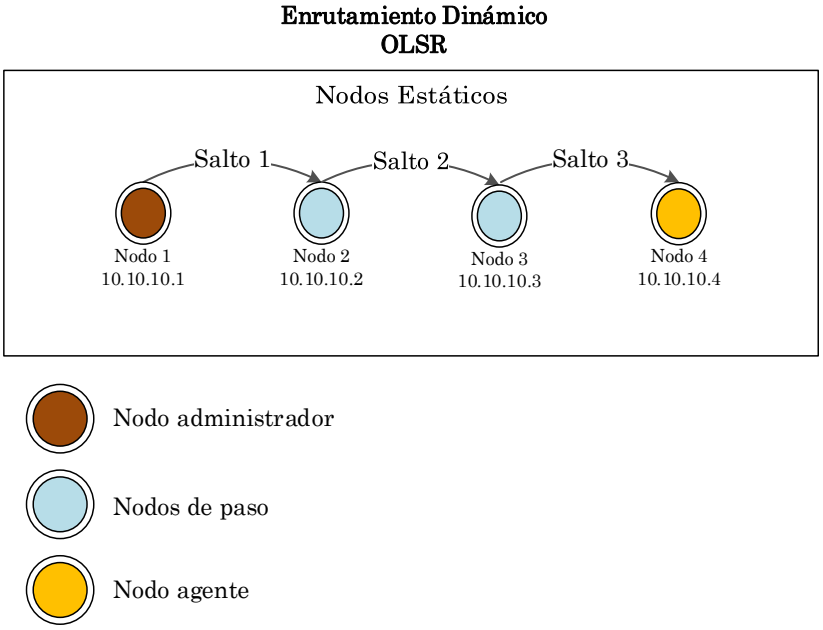


Figura 56: Escenario 2: Red multi-salto con enrutamiento OLSR

El protocolo de enrutamiento OLSR es un protocolo proactivo que permite mantener tablas actualizadas de enrutamiento en todo momento, sin embargo, requiere una carga mayor en la red debido a que envía paquetes de control.

En concreto utiliza mensajes denominados *Hello* mediante los cuales se hace el descubrimiento de nodos vecinos. Dichos mensajes se intercambian de manera periódica para los nodos que están a un salto, es decir no se retransmiten más allá de este salto.

También se utilizan mensajes de control de topología, estos se difunden por toda la red y se envían únicamente por nodos especiales denominados MPR (*Multi Point Relay)* lo que permite reducir el tráfico de enrutamiento en la red.

A nivel de implementación, se encuentra disponible el demonio *olsrd*, el cual cuenta con un archivo llamado *olsr.conf* donde se definen sus distintas opciones de configuración. El proceso detallado para el enrutamiento OLSR de nodos se presenta en el Apéndice E.

6.4.3 Escenario 3: Red multi-salto con un nodo móvil y enrutamiento OLSR

En este escenario se añade un quinto nodo móvil llamado Nodo 5 a la red como se observa en la Figura 57. Con la dirección IP 10.10.10.5, que se configura como agente SNMP. Dicho experimento se planificó con el objetivo de evaluar la arquitectura propuesta bajo condiciones de movilidad.

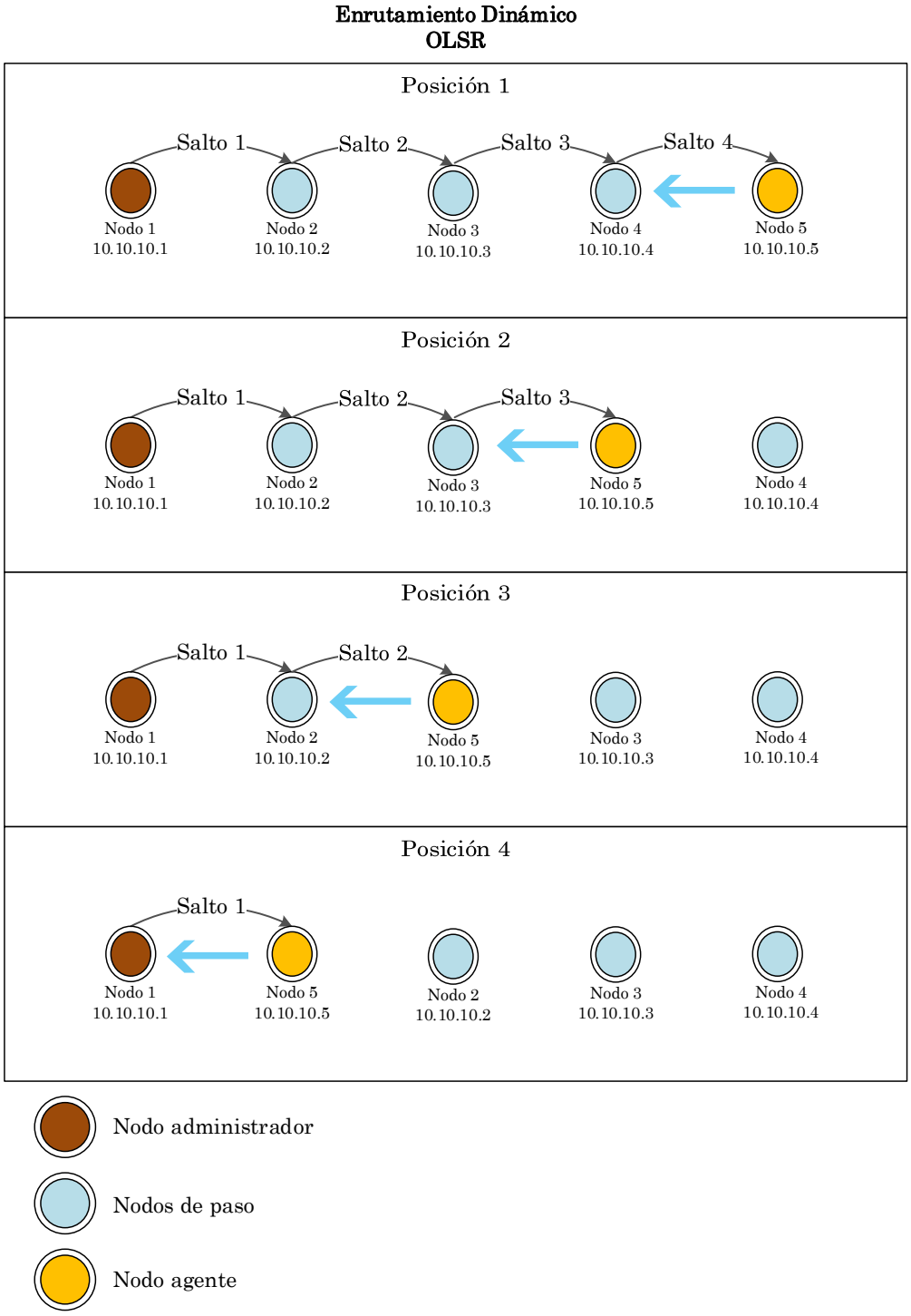


Figura 57: Escenario 3: Red multi-salto con un nodo móvil y enrutamiento OLSR

En concreto el nodo móvil inicia a 23 metros del Nodo 4 que es el extremo de la red, se mueve en dirección hacia el Nodo 1 con el que tiene 92 metros de distancia. Pasa por todos los nodos de la red hasta terminar en la posición del Nodo 1. La velocidad de movimiento del nodo se configura a 0,153 (metros/segundo) para cubrir toda la trayectoria en un tiempo de diez minutos.

6.5 Resultados de la simulación

Se realizaron diez simulaciones para cada escenario con una duración de 10 minutos cada una, con el objetivo de que estos experimentos sean estadísticamente válidos. En particular, se efectuaron cinco simulaciones con la arquitectura que contiene un nodo como el ordenador y cinco simulaciones que contienen todos los nodos virtualizados. Las simulaciones promedio se realizaron utilizando un intervalo de confianza del 95%.

En el caso del tráfico se obtienen los datos cada 60 segundos de simulación debido a que la aplicación desarrollada en el Capítulo 4 hace las solicitudes SNMP cada 60 segundos. Los paquetes capturados, el porcentaje de recepción de paquetes y el retardo de realiza para todo el tráfico en la red con el objetivo de caracterizar la red multi-salto. Las tablas con los resultados se presentan en el Apéndice F, en esta sección se muestran los gráficos de barras de valores promedio de las diez simulaciones que resumen dichos resultados.

Para la métrica del tráfico de red tal como se presenta en la Tabla 17, en el caso del enrutamiento estático se tiene un valor de 10,36 kbps, se muestra un aumento de 1,06 kbps al utilizar el enrutamiento OLSR y para el caso móvil el valor promedio se mantiene cercano a los 10 kbps. Sin embargo, el nivel de tráfico generado continúa siendo bajo en comparación con la capacidad disponible en el estándar IEEE 802.11n. En la Figura 58, se observa este comportamiento con el intervalo de confianza respectivo de cada medición.

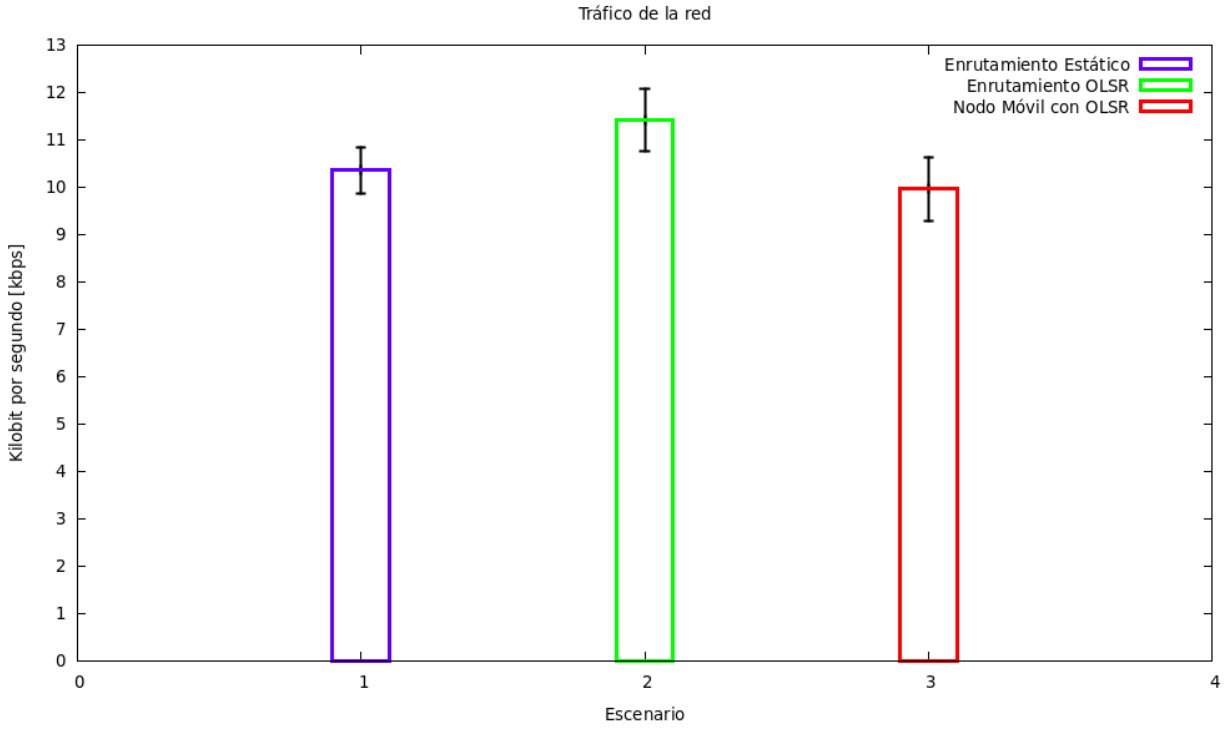


Figura 58: Tráfico promedio de la red para cada escenario

Escenario	Tráfico promedio de la red [kbps]
1	10,36
2	11,42
3	9,96

Tabla 17: Tráfico promedio de la red en cada escenario

En la Tabla 18, se presenta el porcentaje de recepción de paquetes, para los escenarios estáticos se comportan de manera similar con un valor cercano al 95% logrando que la red sea estable y la aplicación funcione correctamente, es importante destacar que nunca se llega al 100% ya que el medio inalámbrico es muy variable y no siempre se dan las condiciones óptimas. En el caso del escenario móvil se puede apreciar una clara disminución de esta métrica a un valor del 45% aproximadamente, el proceso de enrutamiento junto con movilidad del nodo dificulta la recepción de los paquetes. Esto resulta en que algunas solicitudes SNMP no son respondidas por el agente y la red se vuelve inestable. En la Figura 59, se observa este comportamiento con el intervalo de confianza respectivo de cada medición.

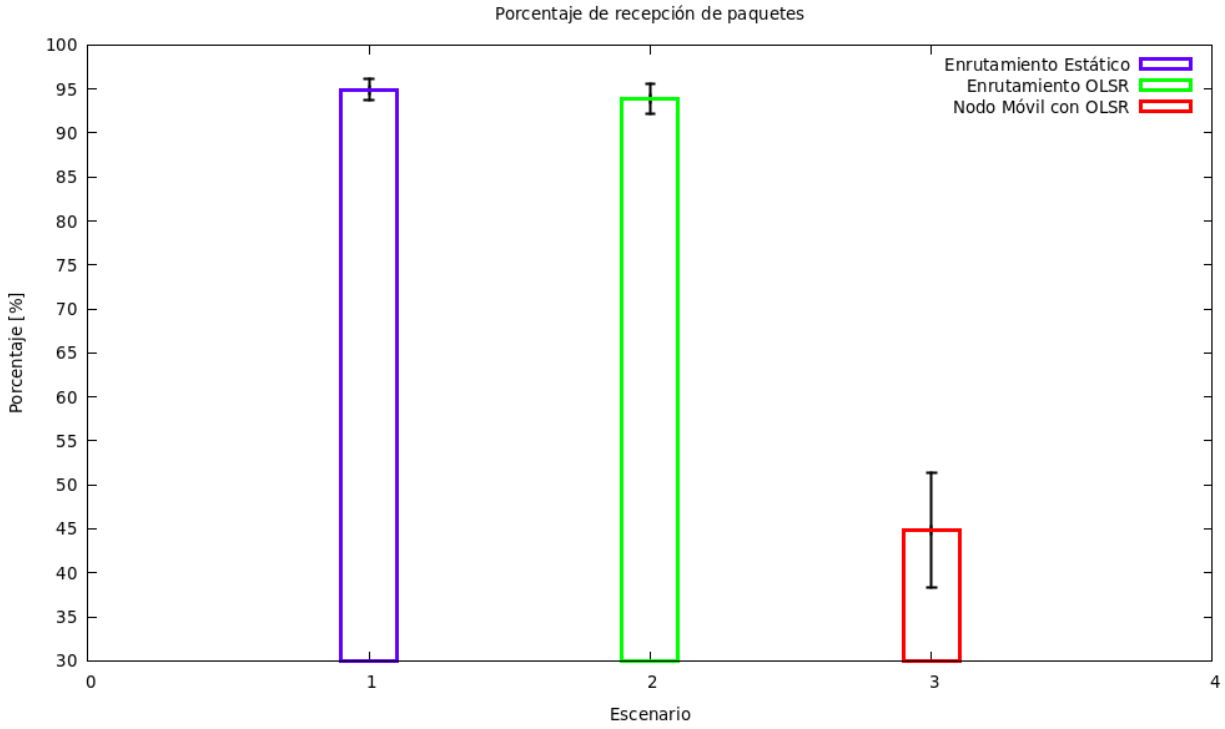


Figura 59: Porcentaje de recepción de paquetes promedio para cada escenario

Escenario	Porcentaje promedio de recepción de paquetes [%]
1	94,91
2	93,90
3	44,88

Tabla 18: Porcentaje de recepción de paquetes promedio para cada escenario

En el caso de métrica del retardo que se presenta en la Tabla 19, se mantiene en valores promedio similares para los escenarios estáticos cercanos a los 60 ms y en el escenario móvil de puede apreciar una disminución hasta los 35 ms, esto se debe al progresivo acercamiento del nodo hacia el administrador. En la Figura 60, se observa este comportamiento con el intervalo de confianza respectivo de cada medición.

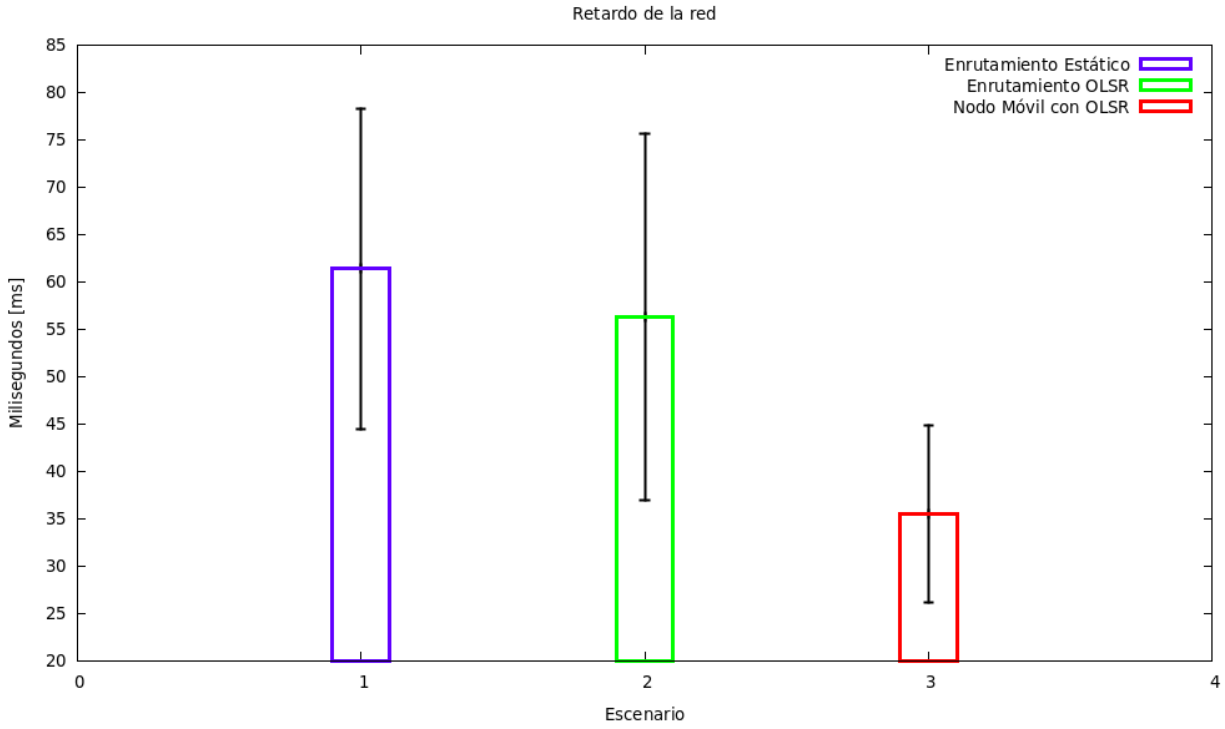


Figura 60: Retardo promedio en la red para cada escenario

Escenario	Retardo promedio de la red [ms]
1	61,4
2	56,3
3	35

Tabla 19: Retardo promedio en la red para cada escenario

6.5.1 Ancho de banda de la red multi-salto

Para dimensionar el ancho de banda máximo de la red se utilizó la herramienta *iperf*, dicho procedimiento se presenta en el Apéndice G.

Iperf permite enviar tráfico desde el Nodo 1 que actúa como cliente a los nodos 2, 3 y 4, que actúan como servidores. En la Figura 61, se observa que el ancho de banda disminuye en cada salto.

En la Tabla 20, se presenta los valores de cada arquitectura y con un valor promedio de 2225 kbps en el primer salto, 849 kbps en el segundo y 529 kbps en el tercero. En [51] se encuentra este mismo patrón de comportamiento de una red Ad Hoc para cada salto entre nodos donde

se resalta que este comportamiento, se debe principalmente a las interferencias intra-flujos (interferencia radio y contención del canal) entre los nodos. Sin embargo, cabe destacar que el ancho de banda determinado es muy superior a los requerimientos de tráfico demandados por la aplicación.

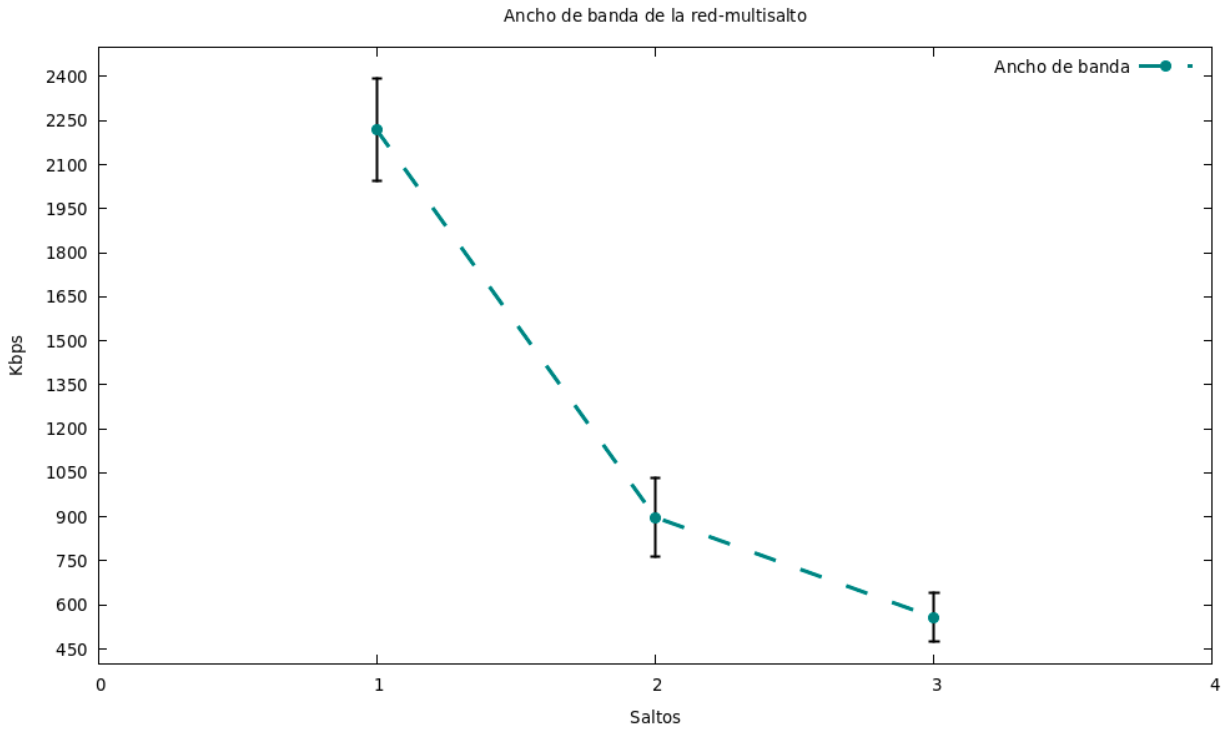


Figura 61: Ancho de banda promedio para cada uno de los saltos

Ancho de Banda [kbps]	Arquitectura 1	Arquitectura 2	Promedio
1 (Nodo 1 - Nodo 2)	2240	2210	2225
2 (Nodo 1 - Nodo 2- Nodo 3)	812	886	849
3 (Nodo 1 - Nodo 2 - Nodo 3 - Nodo 4)	505	553	529

Tabla 20: Ancho de banda promedio de la red

6.6 Conclusiones

El uso del software NS3 y contenedores de Linux posibilitan la interacción entre el mundo real y el virtualizado, lo que permite llevar a cabo el análisis de las métricas de *throughput*,

porcentaje de recepción de paquetes y retardo en la red Ad Hoc multi-salto permitiendo hacerlo de manera realista con el tráfico SNMP generado.

Esto facilitó el estudio de distintos escenarios porque se puede variar parámetros sin la necesidad de desplegar la red de manera física ya que esto limitaría mucho el desarrollo de los experimentos.

Según los resultados encontrados correspondientes al tráfico en la red, se puede verificar que existe un mayor tráfico en el enrutamiento OLSR ya que se producen mensajes propios del protocolo lo que no sucede en un escenario con enrutamiento estático. De igual manera, se observa que existen pérdidas ocasionadas por la distancia de separación entre los nodos y a la variación del canal simulado, esto no afecta significativamente a la red ya que se tiene una tasa de recepción superiores al 93% en los casos estáticos y el proceso de gestión no sufre mayor afectación.

Por otro lado, al verificar los resultados de retardo en la red esta métrica depende principalmente de las características del medio y el tráfico de la red, además, de la posición y movimiento de los nodos dependiendo de cada escenario.

En el escenario del nodo móvil es donde se encuentran problemas, este al moverse disminuye considerablemente la estabilidad que obtiene la red en ambos casos estáticos, así como no siempre se reciben las consultas SNMP realizadas, por lo que se considera el caso con los peores resultados.

Como trabajos futuros en el escenario móvil se recomienda realizar más experimentos ya que existen más variables a considerar como: la velocidad del nodo, la distancia entre nodos, el patrón de desplazamiento, el tiempo de actualización de los mensajes de enrutamiento y el tipo de protocolo que se usa, ya sea proactivo o reactivo.

Al verificar el ancho de banda en la red, se puede concluir que este disminuye con cada salto producido, se obtiene el mayor ancho de banda en el primer salto y el menor entre los nodos 1 al 4. Entre cada salto, el ancho de banda disminuye considerablemente, sin embargo, este es un comportamiento inherente a las redes Ad Hoc.

Capítulo 7: Conclusiones y Recomendaciones

7.1 Conclusiones

El trabajo de titulación realizado es un desarrollo versátil porque puede adaptarse en una gran cantidad de escenarios de *Smart Cities* que requieran monitoreo y gestión de nodos. En el escenario concreto de este trabajo, con los nodos desplegados se puede monitorizar variables ambientales y de geolocalización de un sitio específico lo que permite al usuario conocer el entorno de manera remota.

El factor diferenciador de este desarrollo es que se puede implementar sin necesidad de una red estructurada, ya que en algunos entornos no existe dicha infraestructura y la red Ad Hoc sustituye esta carencia. De esta manera, el despliegue se vuelve independiente del lugar geográfico. El uso de baterías de la misma manera, hace que el sistema no necesite de la red eléctrica para funcionar.

El problema principal está en la recarga de las baterías porque tienen una capacidad energética definida, esta limitación debe ser solucionada para mejorar la adaptabilidad del sistema a prácticamente cualquier entorno.

Por otra parte, el uso del protocolo SNMPv3 cumple satisfactoriamente con los objetivos planteados haciendo que el monitoreo y gestión sea simple para los dispositivos y sensores. Lo que resulta en que sea factible integrar otras plataformas distintas a Raspberry Pi al sistema, siempre que soporten el modo Ad Hoc y el protocolo SNMP. De la misma manera con cualquier tipo de sensor, lo que hace que las posibilidades sean muy amplias.

El *framework* Node-RED se integra directamente con SNMP gracias a las librerías y capacidades que ofrece, por lo que es una herramienta útil en este cometido. El uso de la interfaz gráfica brinda prestaciones para que el monitoreo se realice en tiempo real, el uso de un mapa para ver la localización geográfica y que se puedan programar las funciones de control de los sensores, perfiles y ahorro de energía.

El análisis de rendimiento de los nodos permite al usuario conocer la carga computacional del nodo ya que no siempre se requiere tener todos los sensores activos y como se hizo énfasis el uso de la batería hace que sea necesario conocer el consumo energético. De la misma manera, el uso de memoria debe tenerse en cuenta ya que no todos los dispositivos tienen la misma capacidad.

Con esta información el usuario según sus necesidades tiene la capacidad de desactivar sensores e incluso dejar el nodo únicamente para que transmita datos hasta cargar la batería.

La aplicación web implementada con IBM Cloud permite hacer el monitoreo desde Internet, ya que los datos se envían por el *gateway* a la nube, lo que amplía la capacidad de monitorización.

En lo que respecta a la segunda parte de este trabajo, el uso de la herramienta de emulación utilizando hardware real en NS3 brinda capacidades como el uso del propio ordenador como un nodo y la más importante el uso de contenedores Linux lo que permite elegir el sistema operativo y las aplicaciones que se ejecutan en los contenedores.

A diferencia de una simulación de NS3 convencional, en la que únicamente se configura el nodo en el código que se ejecuta y, el envío de tráfico es limitado y complejo. Con la herramienta toda la configuración se traslada a los contendores, esto se traduce que el tráfico enviado es real y es generado por los mismos, comportándose como si el nodo fuera hardware real.

En el caso de este trabajo esto hace que la configuración del administrador y agente en los nodos se realice con el mismo proceso del ordenador y la Raspberry Pi, enviando tráfico real generado por la aplicación que se desarrolló.

Por consiguiente, el enrutamiento de los nodos de la manera estática o dinámica se realiza directamente en el nodo, con lo que el proceso al pasar a un hardware real sea exactamente igual. Lo que se deja al código en NS3 es únicamente la simulación del canal Ad Hoc, el posicionamiento de los nodos y la movilidad en caso de que se requiera.

Con estas bases, se plantearon los distintos escenarios para una red multi-salto para evaluar las variables de red. Los resultados muestran que la red es estable y eficiente en el caso estático ya sea utilizando enrutamiento estático y con un ligero aumento del tráfico con enrutamiento OLSR.

Por último, en el caso de la red móvil con enrutamiento OLSR, el nodo móvil cambia este comportamiento y la red disminuye su eficiencia, por lo que se requiere considerar una solución para acercarse a los valores obtenido en el caso estático.

7.2 Recomendaciones

Como recomendaciones para trabajos futuros principalmente se plantea el despliegue del sistema con la red multi-salto para analizar su respuesta respecto a las variables de red. Esto con el fin de contrastar los resultados obtenidos en el simulador con la realidad. Además, esta red puede ser heterogénea, es decir, que utilice plataformas que soporten el modo Ad Hoc y una distribución Linux.

En cuanto a las desventajas planteadas por el uso de baterías se puede plantear la recarga mediante una solución preferentemente de energías renovables o en los escenarios donde esto no se permita con energías no renovables. Esto con el fin de incrementar la autonomía de la red y prolongar su uso.

Respecto al protocolo utilizado SNMPv3 se recomienda la comparación con otros protocolos, como por ejemplo el protocolo MQTT que es muy popular en la actualidad. De la misma

manera con una SDN con el objetivo de comparar las ventajas y desventajas al usar dichas tecnologías.

En el desarrollo de la aplicación como tal, el protocolo SNMP es extenso y brinda muchas capacidades por lo que se recomienda profundizar aún más en su uso para hacer un mejor uso del sistema. Así como también añadir nuevas funciones a la aplicación en futuras versiones.

Una de estas mejoras estaría enfocada en añadir una capa de seguridad a la aplicación ya que si se enfoca en el manejo de datos sensibles la red podría ser vulnerable a un ataque ya que esto es un problema inherente a las redes Ad Hoc y es tema de investigación actual.

Con un enfoque a nivel comercial, se puede mejorar el encapsulado de los nodos para que resistan distintas condiciones climáticas.

Finalmente, a nivel de monitoreo, se pueden conectar el sistema a Internet mediante el uso de una IP pública y el *gateway* por defecto.

Anexos

Apéndice A: Scripts que utilizan la función Pass-Through

Las variables que se pueden monitorizar son tanto del propio nodo, así como también valores obtenidos mediante sensores. La monitorización puede ser a través de consultas utilizando *snmpget* o también utilizando *snmpset* que sirve para establecer un valor.

A.1 Script de temperatura del nodo

A continuación, se describe el proceso para monitorizar la temperatura de la placa del dispositivo realizando la conversión a grados Celsius con el comando:

echo $(($(cat /sys/class/thermal/thermal_zone0/temp) / 1000))

Crear un *script* que tenga el OID definido y muestre la información requerida:

```
#!/bin/bash
if [“$1” = “-g” ]
then
echo .1.3.6.1.4.1.8072.2.1
echo integer
echo $(($(cat /sys/class/thermal/thermal_zone0/temp) / 1000))
fi
exit 0
```

Para poder ejecutar se debe hacer el *script* ejecutable con el siguiente comando:

sudo chmod +x “Nombre del Archivo”

A.2 Script para control de pin GPIO

Para controlar la GPIO se utiliza el *script* presentado a continuación, en este caso el Pin 25 en el que se define un OID y un valor de 1 para encendido y 0 para apagado.

```
#!/bin/bash
if [ "$1" = "-g" ]
```

```
then
echo .1.3.6.1.4.1.8072.2.2
echo integer
pin=25
gpio mode $pin output
gpio read $pin
fi

if [ "$1" = "-s" ]
then
pin=25
gpio write $pin $4
fi
exit 0
```

A.3 Scripts para los sensores

Para todos los sensores se sigue la misma estructura que permite definir un OID y presentar la información:

```
#!/bin/bash
if [ "$1" = "-g" ]
then
echo .1.3.6.1.4.1.8072.2.3
echo integer
echo $(cat /home/pi/ldr.txt)
fi
exit 0
```

A.4 OIDs definidos para las variables y por defecto

En la Tabla 21, se presenta los OID definidos para las variables. Además, se presentan los OID que viene por defecto en el paquete *snmp* que son utilizados en la aplicación.

Variables definidas	OID
Temperatura del nodo	.1.3.6.1.4.1.8072.2.1
Control Pin GPIO	.1.3.6.1.4.1.8072.2.2
Luz	.1.3.6.1.4.1.8072.2.3
Temperatura ambiente	.1.3.6.1.4.1.8072.2.4
Humedad ambiente	.1.3.6.1.4.1.8072.2.5
Voltaje	.1.3.6.1.4.1.8072.2.6
Corriente	.1.3.6.1.4.1.8072.2.7
Altitud	.1.3.6.1.4.1.8072.2.8
Latitud y Longitud	.1.3.6.1.4.1.8072.2.9
TVOC	.1.3.6.1.4.1.8072.2.10
eCO2	.1.3.6.1.4.1.8072.2.11
Variables por defecto	
RAM en uso	.1.3.6.1.4.1.2021.4.6.0
Espacio utilizado en el disco	.1.3.6.1.4.1.2021.9.1.8.1
Carga del CPU	.1.3.6.1.4.1.2021.10.1.3.3

Tabla 21: OIDs definidos para cada sensor

Apéndice B: Instalación de los sensores del nodo

Todos los sensores se ejecutan sobre *CircuitPython* que es un lenguaje basado en Python. Este *software* sirve para la programación de microprocesadores y es en el que se ejecutan librerías para el uso de cada sensor.

Los pines de la Raspberry Pi de este anexo se refieren a la numeración física de los mismos. La alimentación de todos los sensores se realiza a 5V en el Pin 2 y la referencia GND en el Pin 6.

B.1 Instalación de sensor de corriente y voltaje (INA219)

El sensor requiere la instalación de la librería *sudo pip3 install adafruit-circuitpython-ina219.* La conexión de los pines se realiza de la siguiente manera en la Figura 62, en esta se observa la conexión:

- Vcc a la alimentación de 5V o 3.3V (Soporta ambos voltajes) (Pin 2)
- Gnd a al pin Pi GND (Pin 6)
- Scl a al pin Pi SCL (Pin 5)
- Sda a al pin Pi SDA (Pin 3)

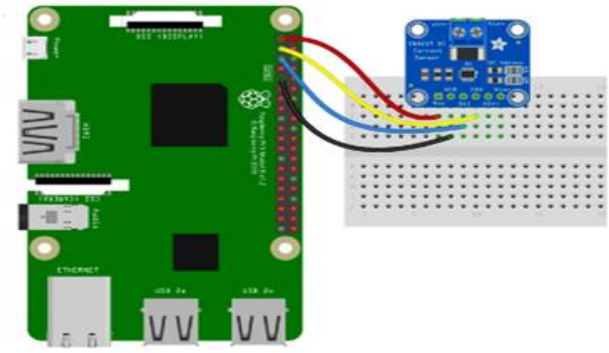


Figura 62: Conexión del sensor INA219

Para realizar la medición se utiliza un cable USB Tipo A a USB Tipo C, cuyas conexiones van a la batería y a la Raspberry Pi respectivamente. Se realiza un corte en el cable de alimentación V+ para colocar el sensor de la siguiente manera.

- V+ a la batería (USB A)

- V- a la entrada de alimentación de la Raspberry Pi (USB C)

B.2 Instalación de sensor de temperatura y humedad (DHT11)

El sensor de humedad requiere la librería *sudo pip3 install Adafruit_DHT* cuya conexión se realiza como en la Figura 63 para la versión de 3 pines y como la Figura 64, para la versión de 4 pines.

- Vcc a la alimentación de 5V (Pin 2)
- Gnd a al pin Pi GND (Pin 6)
- Señal al pin GPIO 4 (Pin 7)

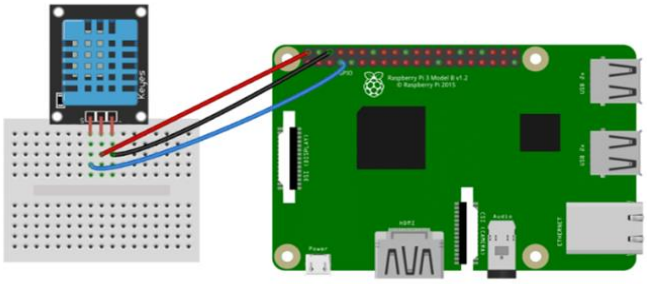


Figura 63: Conexión de sensor DHT11 de tres pines

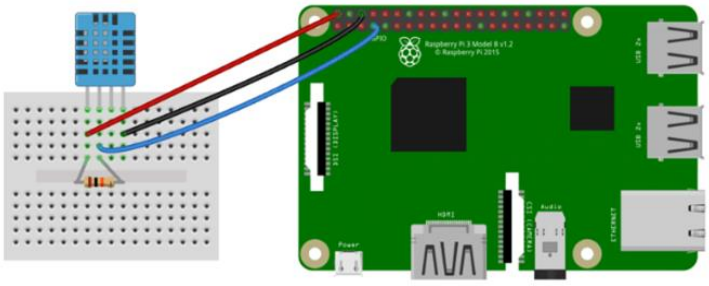


Figura 64: Conexión de sensor DHT11 de cuatro pines

B.3 Instalación de sensor de luz (LDR)

El sensor de luz (LDR) al ser un sensor analógico no requiere de ninguna librería, por lo que únicamente requiere conectarse a un capacitor de 1uF formando un circuito RC que hace posible la medición.

La conexión se realiza como se presenta en la Figura 65:

- Vcc a la alimentación de 5V (Pin 2)
- Gnd a al pin Pi GND (Pin 6)
- Señal al pin GPIO 18 (Pin 12)

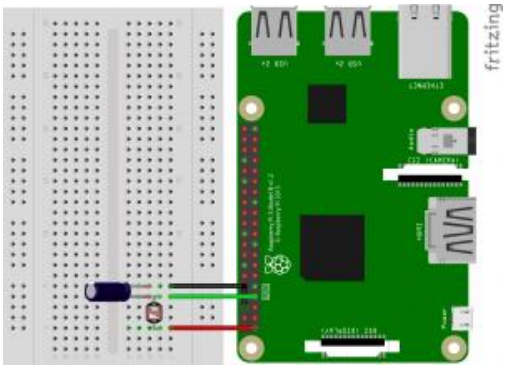


Figura 65: Conexión del sensor de LDR

B.4 Instalación de sensor de geolocalización (GPS Breakout v3)

La conexión del sensor GPS se realiza utilizando la interfaz UART de la Raspberry Pi lo que requiere desactivar la interfaz serial del mismo.

Para este proceso se ejecuta el comando *sudo raspi-config* en las opciones de interfaz se selecciona en Serial y a continuación, desactivar la consola serial como en las Figuras 66 y 67.

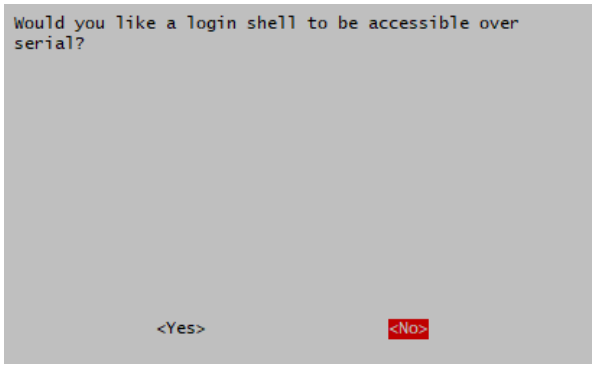


Figura 66: Desactivar el login mediante acceso serial

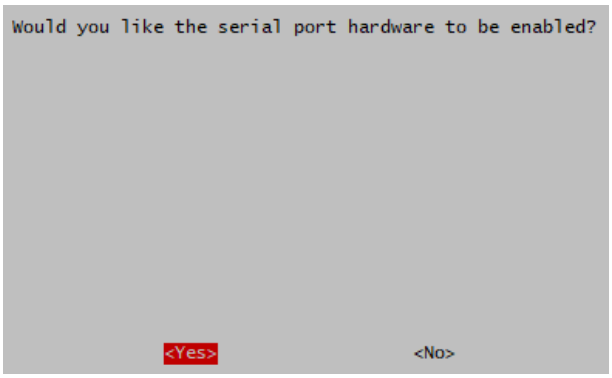


Figura 67: Habilitar el puerto serial

Una vez realizado se reinicia la plataforma y se instala las siguientes librerías *sudo pip3 install adafruit-circuitpython-gps* y se recomienda actualizar la librería *sudo pip3 install --upgrade adafruit_blinka.*

La conexión del sensor se realiza como en la Figura 68:

- Vcc a la alimentación de 5V o 3.3V (Soporta ambos voltajes) (Pin 2)
- Gnd a al pin Pi GND (Pin 6)
- Tx a al pin Pi UART0 RX (Pin 10)
- Rx a al pin Pi UART0 TX (Pin 8)

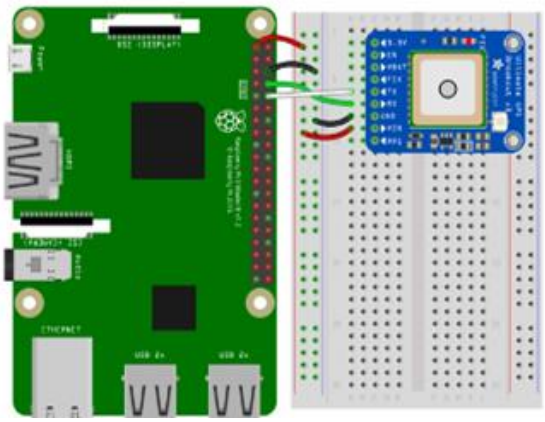


Figura 68: Conexión del sensor GPS

En el código en Python se debe activar el acceso mediante la UART, por lo general se utiliza la interfaz ttyS0, el *timeout* de actualización se configura en función de la necesidad del usuario.

```
import serial
uart = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=10)
```

B.5 Instalación de sensor de contaminación (SGP30)

El sensor de contaminación requiere la instalación de la librería *sudo pip3 install adafruit-circuitpython-sgp30* la conexión se realiza como en la Figura 69:

- Vcc a la alimentación de 5V (Pin 2)
- Gnd a al pin Pi GND (Pin 6)
- Scl a al pin Pi SCL (Pin 5)
- Sda a al pin Pi SDA (Pin 3)

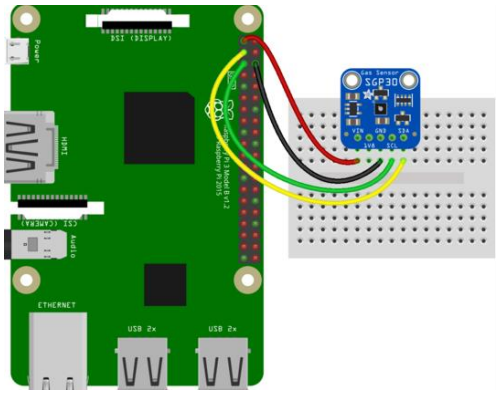


Figura 69: Configuración del sensor SGP30

B.6 Instalación de LED indicador

Para el funcionamiento del LED indicador se requiere la librería *Wiring Pi* para controlar los pines de la GPIO que viene pre instalada. En caso contrario se utiliza el comando *apt-get install wiringpi.*

Con el comando *sudo gpio readall* se puede revisar los pines y sus distintos usos. En este caso se hace la conexión al Pin 25 GPIO que entrega el voltaje y corriente necesarios para controlar un LED, la conexión se realiza de la forma que presenta la Figura 70:

- Ánodo al Pin 25 GPIO (Pin 37)
- Cátodo a al pin Pi GND (Pin 6)

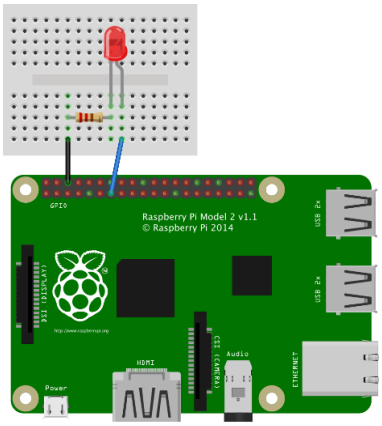


Figura 70: Configuración del LED indicador

Apéndice C: Instalación, ejecución y configuración funciones de Node-RED

C.1 Instalación manual mediante comandos

La instalación manual de Node.js que es el entorno de ejecución para JavaScript y contiene el administrador de paquetes *npm*. Se realiza mediante el comando *apt install nodejs.* A continuación, se debe instalar Node-RED, por medio del gestor de paquetes npm.

npm install -g –unsafe-perm node-red node-red-admin

- -g: Permite instalar el paquete de manera global.
- -*unsafe-perm*: Para suprimir cualquier error que se produzca en la instalación.

Para evitar problemas con el firewall se debe permitir el uso del puerto 1880 para lo cual se utiliza la herramienta ufw con el comando *ufw allow 1880.*

C.2 Instalación mediante un script

Los desarrolladores ofrecen la posibilidad de automatizar la instalación de todos los paquetes mediante un *script* para una distribución basada en Debian como Raspberry Pi OS y Ubuntu, que se puede descargar y ejecutar con el siguiente comando:

bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

Lo que instala Node.js, Node-RED y todas las dependencias necesarias para su ejecución.

C.3 Integración de SNMP en Node-RED

La aplicación Node-RED permite integrar el protocolo SNMP mediante módulos, los cuales se instalan con el gestor de paquetes *npm*. Para la instalación en primer lugar se debe cambiar de directorio, hacia la carpeta correspondiente a Node-RED con el comando *cd $HOME/.node-red* en este directorio se deben instalar todas las librerías.

Node-RED tiene nodos para SNMP en el módulo *node-red-node-snmp* que vienen por defecto o pueden ser instalados desde la aplicación. Sin embargo, no son de utilidad en este caso debido a que solo soportan las versiones 1 y 2. Por esta razón se debe instalar librerías para

poder utilizar SNMPv3 mediante funciones en JavaScript, para la instalación se utiliza el comando: *npm install net-snmp*

Para que se pueda utilizar este módulo, se debe realizar modificar el archivo de configuración de *settings.js.* Este archivo se encuentra en el directorio *$HOME/.node-red* y se modifica la línea *snmpModule:require('net-snmp')* como en la Figura 71.

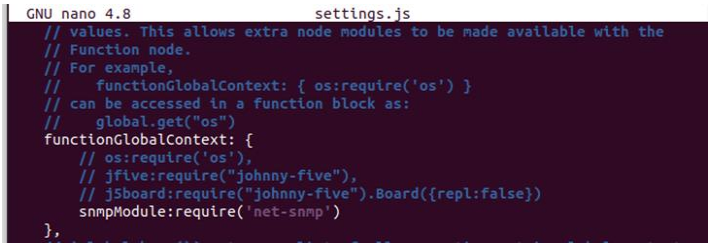


Figura 71: Modificación realizada en el archivo Setting.js

C.4 Configuración del nodo Inject

El nodo temporizador de la Figura 72, corresponde al nodo en que se puede configurar un intervalo de tiempo para hacer consultas SNMP. En este caso se configura para hacer consultas cada 60 segundos. En la Figura 73, se presenta las propiedades que debe tener el nodo para esta configuración.

Figura 72: Nodo Inject

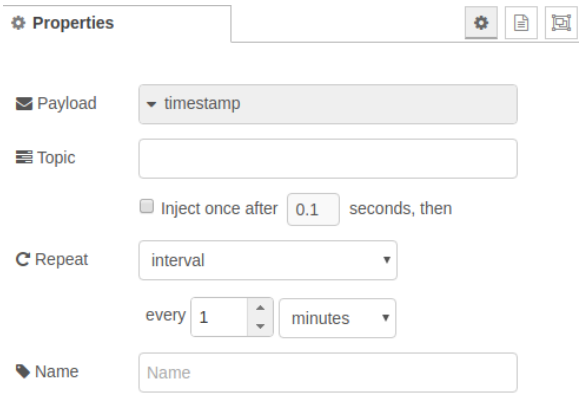


Figura 73: Configuración del nodo Inject

C.5 Configuración del nodo Function

El nodo *Function* de la Figura 74, correspondiente a la consulta SNMP escrita en *JavaScript,* mediante este nodo. Para la aplicación se realizan dos tipos de consulta:

- *Snmpget*: Permite leer variables al obtener información de uno o varios OID.
- *Snmpset*: Permite escribir o modificar valores de una variable en uno o varios OID.

Figura 74: Nodo Function

Para cargar la función se utiliza la siguiente instrucción:

```
var snmp = global.get('snmpModule');
```

Una vez que se carga el módulo, se especifica el puerto, reintentos, el protocolo de transporte y la versión. El código utilizado se muestra a continuación:

```
var options = {
        port: 161,
        retries: 1,
        timeout: 5000,
        transport: "udp4",
        trapPort: 162,
        version: snmp.Version3,
idBitsSize: 32,
context: ""
};
```

De igual manera se especifica el usuario y la seguridad implementada en la autenticación y encriptación, en este caso se utiliza MD5 y AES. Además, se especifica la dirección IP a la cual se realiza la consulta. En este caso se coloca la dirección *10.10.10.3* de la red Ad Hoc del agente que es la Raspberry Pi.

```
var user = {
name: "authOnlyUser",
        level: snmp.SecurityLevel.authPriv,
        authProtocol: snmp.AuthProtocols.md5,
        authKey: "12345678",
        privProtocol: snmp.PrivProtocols.aes,
        privKey: "12345678"
};
var sesión = snmp.createV3Session("10.10.10.3", user, options);
```

Para las consultas snmpget se agrega el o los OID que se desean consultar.

var oids = ["1.3.6.1.4.1.2021.4.6.0", "1.3.6.1.4.1.2021.4.6.1"];
sesión.get (oids, function (error, varbinds)

C.6 Configuración de los nodos Split y Change

Para el nodo *Split* de la Figura 75, los valores se dejan por defecto como se muestra en la Figura 76, debido a que se encarga de dividir el *payload* obtenido por la función SNMP. Para el nodo *Change* se debe mover el valor *msg.payload.value* hacia la salida del bloque, para lo cual se realiza la configuración presentada en la Figura 77.

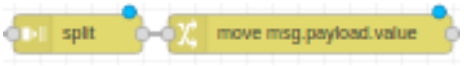


Figura 75: Nodos Split y Change

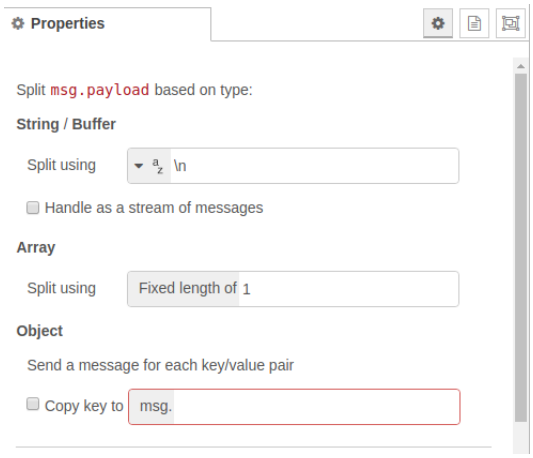


Figura 76: Configuración del nodo Split

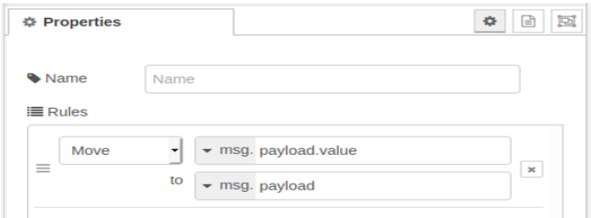


Figura 77: Configuración del bloque Change

C.7 Configuración del nodo Exec

Como alternativa a la librería *snmp-net* con la que se configuraron las consultas se puede utilizar el nodo *Exec* para ejecutar comandos de la terminal, por lo que también es posible ejecutar directamente las consultas. Con la finalidad de demostrar las capacidades de Node-RED se configura mediante este método, el control de encendido/apagado un pin de la GPIO. En la Figura 78, se observa la configuración del nodo *Exec*.

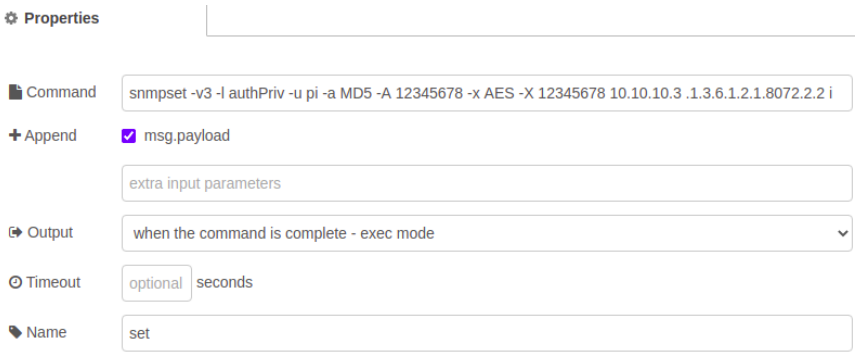


Figura 78: Configuración del nodo Exec

En la Figura 79, se muestra el flujo para el control del pin de la GPIO, como se puede observar ejecuta directamente el comando al enviar el dato desde un botón.

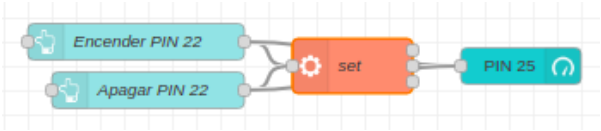


Figura 79: Flujo para el control de pin de GPIO

En la interfaz se muestra el control del LED indicador con el nodo *Gauge* que viene incluido en el *Dashboard* en el que se utiliza la función *snmpset* para colocar una variable para el encendido y apagado. La interfaz presentada en el *Dashboard* se muestra en la Figura 80.

Figura 80: Control del LED indicador en el Dashboard

C.8 Instalación y configuración del módulo Dashboard

El primer paso para realizar este procedimiento, es ingresar en el directorio *$HOME/.node-red* y para instalar se debe ejecutar *npm install node-red-dashboard.* Luego para ingresar a la interfaz se debe ejecutar *localhost:1880/ui.*

En la Figura 81, observa la configuración de la barra lateral para dividir en pestañas cada una de las funciones de las aplicaciones con sus respectivos grupos para organizar los widgets.

Figura 81: Configuración de la barra lateral

C.9 Configuración del nodo Chart

Se utiliza el nodo Chart para presentar el valor obtenido del OID en un gráfico cartesiano en tiempo real, para su configuración se debe ingresar el grupo en el que se va a presentar el gráfico. Además, se puede colocar una etiqueta para identificar el gráfico en el diagrama y contiene opciones para personalizar el gráfico tal como se observa en la Figura 82, se elige presentar el tiempo real en horario UTC en el eje X y en el eje Y el valor obtenido con la consulta, así como resaltar los puntos de medición.

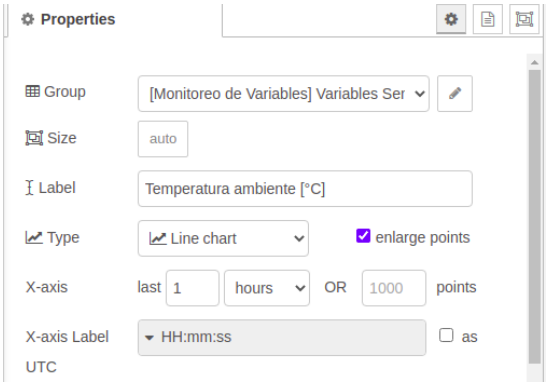


Figura 82: Configuración del nodo Chart

C.10 Visualización de mapa para el sensor de geolocalización

El sensor de geolocalización requiere una librería que se instala con el comando *npm install node-red-contrib-web-worldmap* lo que permite visualizar en un mapa las coordenadas obtenidas. Mediante el nodo *function* se añade la opción de presentar datos del nodo al seleccionar el nodo en el mapa.

La configuración del flujo presentado en la Figura 83, muestra que sigue la estructura general detalla en los puntos anteriores, en la que se añade el *payload* de los otros sensores para la función de mostrar los datos al seleccionar el nodo en el mapa.

Los nodos *Inject, template y worldmap* sirven para configurar la interfaz gráfica en la que se debe colocar el código:

* *

Con el objetivo de que se pueda mostrar el payload. Finalmente, en el nodo *Function* se coloca el código que se presenta más adelante que permite obtener los valores de los nodos *Move.*

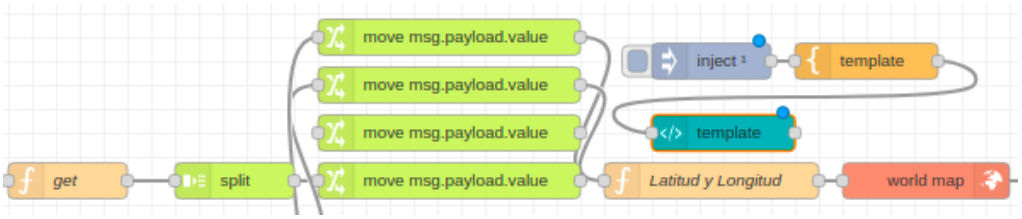


Figura 83: Flujo para visualización de mapa para el sensor de geolocalización

Los datos de geolocalización se obtienen con el formato {Latitud, Longitud} por lo que es necesario utilizar la función *Split* que permite segmentar valores separados por comas. Para presentar los valores de: temperatura, humedad y el uso de memoria RAM se coloca en el *payload*, para que al momento de seleccionar el nodo estos valores se presenten en pantalla.

```
var pay1 = "" + msg.payload;
var res = pay1.split(" ");
var lat = res[0];
var lon = res[1];
var temperatura = msg.payload_temperatura;
var hum = msg.payload_hum;
var ram = msg.payload_ram

msg.payload={
    lat:lat,
    lon:lon,
    Temperatura:temperatura,
    Humedad:hum,
    RAM:ram,
    name:"Nodo",
    icon:"male",
```

C.11 Función snmpget para la obtención de variables

La función *snmpget* ejecutada con la librería *snmp-net*, se configura utilizando las credenciales configuradas en el archivo *snmpd.conf*, se presenta el código completo para su ejecución:

```
var snmp = global.get('snmpModule');
// Configuración para SNMPv3
var options = {
    port: 161,
    retries: 1,
    timeout: 5000,
    transport: "udp4",
    trapPort: 162,
    version: snmp.Version3,
    idBitsSize: 32,
    context: ""
};

// Configuración de usuario
var user = {
    name: "pi",
    level: snmp.SecurityLevel.authPriv,
    authProtocol: snmp.AuthProtocols.md5,
    authKey: "12345678",
    privProtocol: snmp.PrivProtocols.aes,
    privKey: "12345678"
};
```

```
var session = snmp.createV3Session ("10.10.10.3", user, options);
var oids = ["1.3.6.1.2.1.8072.2.9"];

session.get (oids, function (error, varbinds) {
    if (error) {
        console.error (error.toString ());
    } else {
        for (var i = 0; i < varbinds.length; i++) {
            // for version 1 we can assume all OIDs were successful
            console.log (varbinds[i].oid + "|" + varbinds[i].value);

            if (snmp.isVarbindError (varbinds[i]))
                console.error (snmp.varbindError (varbinds[i]));
            else
                console.log (varbinds[i].oid + "|" + varbinds[i].value);
                msg.oids=oids;
                msg.payload=varbinds;
                node.send(msg);
        }
    }
});
```

Apéndice D: Plataforma IBM Cloud

D.1 Creación de servicio, registro de dispositivos y generación de claves

Para trabajar de forma conjunta con el software de Node-RED e IBM Cloud se debe crear un servicio en la segunda plataforma. Para lo que se debe contar con una cuenta de IBM y acceder a la página *cloud.ibm.com* en la que se debe dar de alta en el sistema para el inicio de sesión, tal como lo muestra la Figura 84.

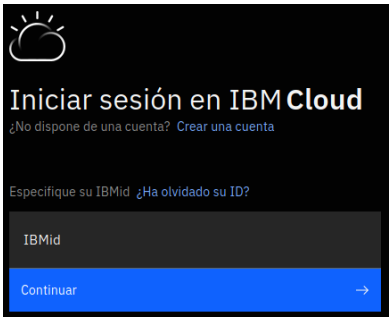


Figura 84: Inicio de sesión en IBM Cloud

Luego de iniciar sesión, se muestra el panel de control de IBM Cloud, en el que se presenta un resumen de los recursos utilizados como: aplicaciones, servicios, herramientas de desarrollo, soporte al cliente, entre otros. En la Figura 85, se observa el panel de control de la plataforma IBM.

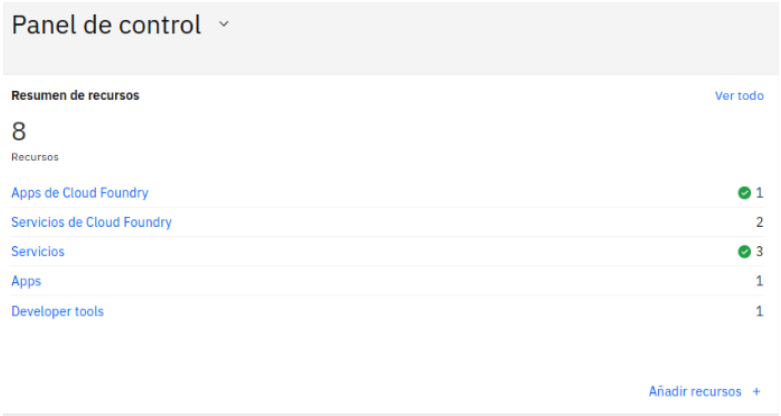


Figura 85: Panel de control IBM Cloud

En la parte superior se encuentra un botón de acceso llamado Catálogo, mediante el cual se pueden buscar los servicios disponibles como por ejemplo plataformas IoT, bases de datos, APIs, *cloud*, entre otros. La Figura 86, muestra el catálogo de opciones.

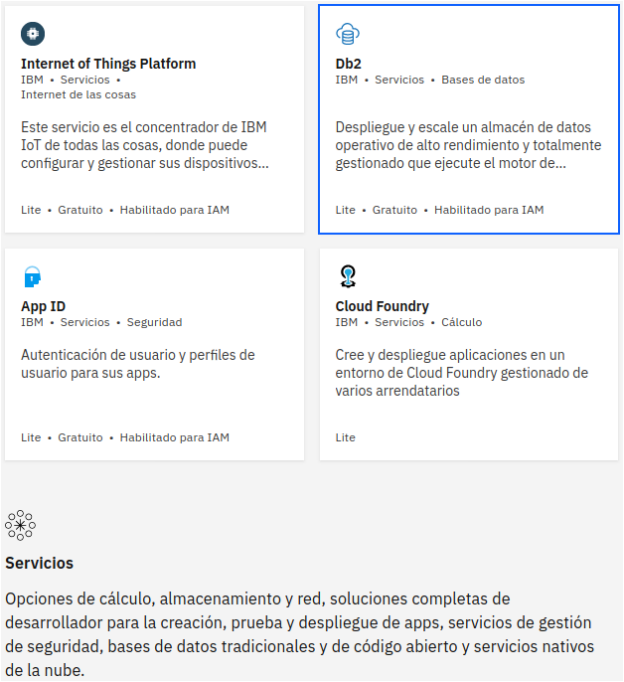


Figura 86: Catálogo de IBM Cloud

El servicio que se utiliza es *Internet of Things Platform,* y se encuentra en el catálogo mencionado anteriormente; este servicio corresponde a la plataforma *Watson IoT*. Una vez que se ha iniciado, se debe configurarlo, en primer lugar, se elige la región en la que se encuentra el sistema, luego se define un nombre y grupo de recursos, como lo muestra la Figura 87.

Figura 87: Servicio de Internet of Things Platform

Después de crear el servicio, se abre el panel de control y en la lista de recursos como la que se muestra en la Figura 88, se selecciona *Internet of Things Platform.*

Una vez seleccionada esta opción se muestra una ventana como la de la Figura 89, en la que se presenta un resumen de las características de este servicio, así como también las opciones de gestión del servicio, modificación de planes y conexión de otros servicios, en esta ventana

también se cuenta con la opción de Lanzar por medio de la cual se abre el panel de la plataforma Watson IoT.

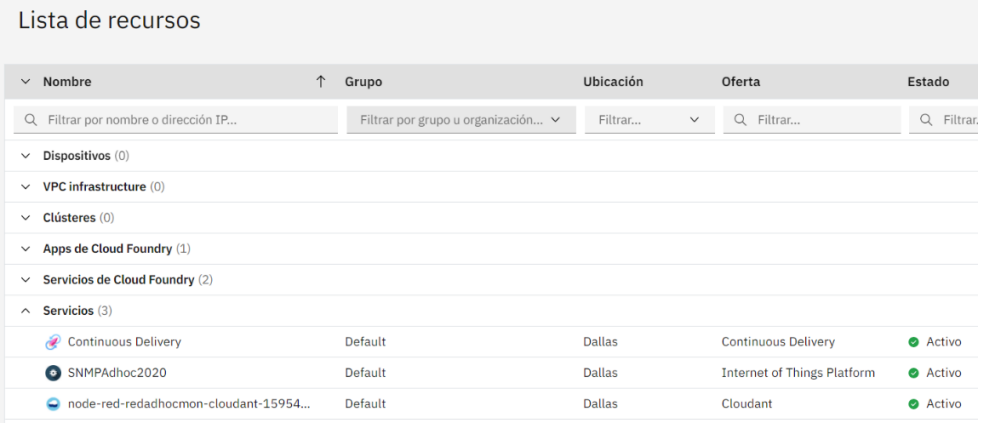


Figura 88: Lista de recursos de IBM Cloud, con servicio IoT

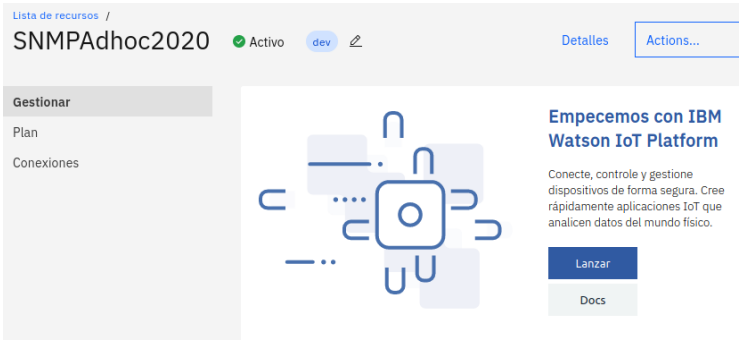


Figura 89: Opciones de gestión IBM Cloud

En la plataforma IBM Watson IoT, se encuentran varias opciones que permiten administrar los dispositivos, de igual manera permiten gestionar el almacenamiento, el acceso a los datos entre otros. Una parte importante de esta plataforma muestra el ID de la organización, el cual se utiliza para realizar la conexión con la aplicación diseñada en Node-RED.

En la Figura 90, se observa la ventana de la plataforma, en la que se encuentran las características mencionadas anteriormente.

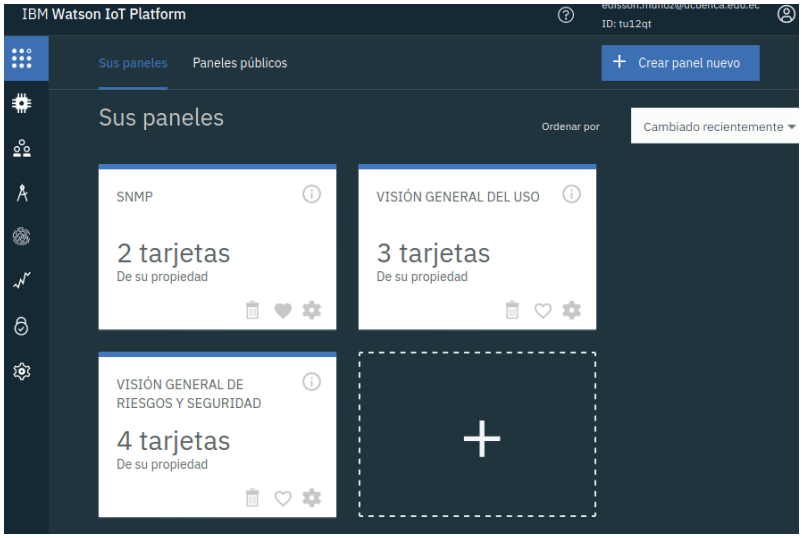


Figura 90: Panel de control

Para realizar el registro de dispositivos en esta plataforma, primero se debe acceder al menú de configuración y activar la función la memoria cache de últimos sucesos, de manera que se almacene la información sobre el último suceso que un dispositivo conectado haya enviado a la plataforma. Esta configuración se presenta en la Figura 91.

Figura 91: Activación de memoria caché

Luego que se realiza esta configuración previa, en la opción de Dispositivos se pueden registrar dispositivos en la organización; para lo cual al seleccionar Tipo de dispositivo y luego Anadir tipo de dispositivo, se muestra una ventana como la Figura 92, en la que se ingresa el Nombre y la Descripción del tipo, y si es necesaria información extra que en este caso se omite.

Figura 92: Registro de dispositivos

Cuando se ha creado el tipo de dispositivo, se puede agregar un dispositivo al mismo, para ello se selecciona la opción examinar y se pulsa sobre el botón añadir dispositivo. A continuación, en una nueva ventana tal como lo muestra la Figura 93, se ingresa la información de Tipo de Dispositivo que corresponde al creado anteriormente y un ID de dispositivo que en este caso es RAM.

Figura 93: Configuración para añadir dispositivo

Al igual que en el tipo de dispositivo se omiten los datos de información, y se continua con la opción de Seguridad, en la que se introduce una señal de autenticación que es una combinación de números y letras, con una extensión de 8 a 36 caracteres, proceso que presenta la Figura 94.

Figura 94: Configuración de seguridad

Al final, se muestra un resumen de los detalles del dispositivo se puede ver en la Figura 95. Esta información se debe recordada, ya que se utiliza en el proceso de integración de Node-RED y Watson IoT, y no se puede acceder a ella otra vez.

Figura 95: Resumen del dispositivo

D.2 Instalación y configuración del módulo Watson IoT en Node-RED

El módulo Watson IoT se debe instalar en las librerías del programa Node-RED para lo cual se utiliza el administrador de paquetes de node (npm) mediante el siguiente comando:

npm install node-red-contrib-ibm-watson-iot

En la interfaz de Node-RED se debe configurar el nodo correspondiente a Watson IoT llamado *Wiotp out*. Este nodo se encarga de enviar la información hacia *IBM Cloud* y de esta manera se pueda almacenar, visualizar, gestionar y controlar la misma. En la Figura 96, se observa una gráfica del nodo.

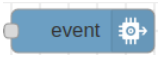


Figura 96: Nodo Watson IoT Platform

El proceso para realizar la configuración de este nodo, se detalla a continuación:

En las propiedades del nodo, se debe especificar que el tipo de conexión se realiza mediante un dispositivo registrado, además se debe especificar el tipo de evento y el formato de salida que es del tipo JSON. Así también, se puede ingresar un nombre para el nodo en caso de ser necesario. La propiedad correspondiente a las credenciales se indica en el siguiente punto. La Figura 97, muestra la configuración de las propiedades de este nodo.

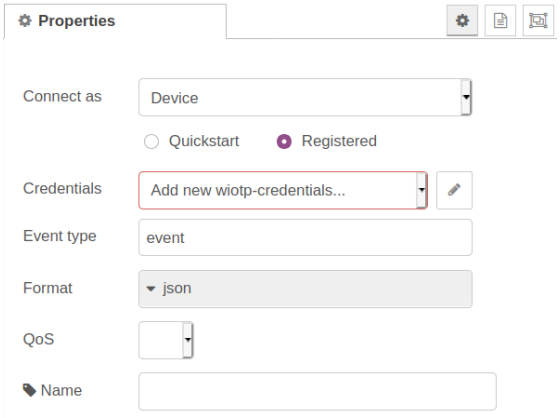


Figura 97: Configuración nodo wiotp

En la propiedad de credenciales, se debe añadir una nueva. En esta se debe ingresar la organización, el tipo de dispositivo, ID de dispositivo y el *token* de autenticación, estos valores se obtienen de la plataforma IBM Watson IoT en el momento de registro de dispositivo. La propiedad correspondiente al nombre del servidor, se omite ya que tiene un valor por defecto y no es necesario su ingreso. La edición de las propiedades de la credencial se puede verificar mediante la Figura 98, en este caso corresponde al valor de la RAM en uso.

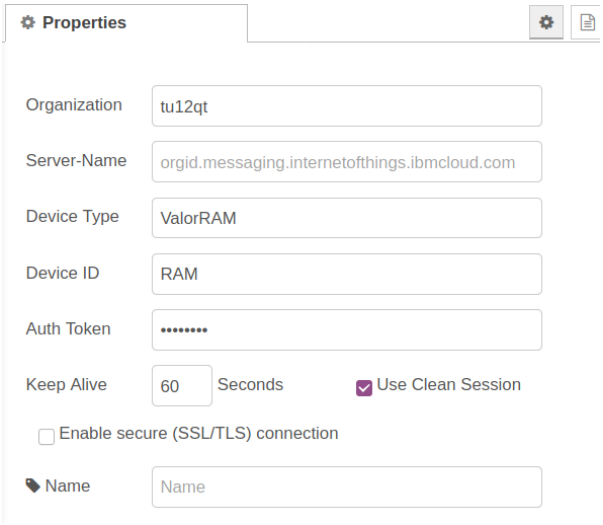


Figura 98: Edición de credenciales Watson IoT

Una vez que se ha configurado el nodo *Wiotp ibm*, se conecta hacia el nodo *Change* encargado de enviar el valor monitorizado hacia las gráficas. De esta manera el flujo para el envío de datos hacia la plataforma de IBM Cloud se encuentra listo. Para verificar la conexión se muestra un indicador en conjunto con el nodo, tal como se observa en la Figura 99.

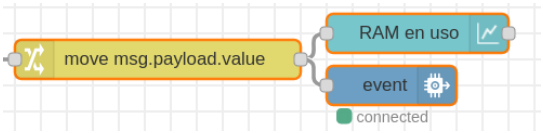


Figura 99: Flujo para la integración de IBM Cloud

D.3 Configuración de dashboard plataforma IBM Watson IoT

Cuando el dispositivo se ha conectado, se puede revisar la información del mismo en la plataforma IBM Watson IoT. Además, se puede observar el Estado del dispositivo para verificar si los datos enviados desde el software Node-RED se están recibiendo. Este resultado se observa en la Figura 100, en la que se comprueba el dispositivo correspondiente a la RAM en uso y donde se obtiene un valor de 2.497.892 kB.

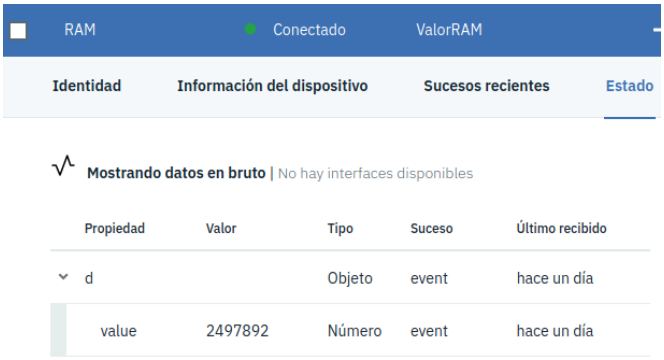


Figura 100: Visualización del estado de un dispositivo

Este proceso se realiza para cada una de las variables a monitorizar, y una vez que se conectan y envían la información hacia la plataforma, se puede implementar un Panel de datos. Para lo cual se debe crear un nuevo Panel, asignarle un Nombre y proporcionar una descripción.

Además, se necesita ingresar la información de miembros en caso de requerir la edición o visualización compartida. Una vez creado el panel se añaden tarjetas al mismo, las que pueden ser: gráfico de líneas, gráfico de barras, diagrama de anillo, valor, entre otros. Los tipos de tarjetas disponibles se observan en la Figura 101.

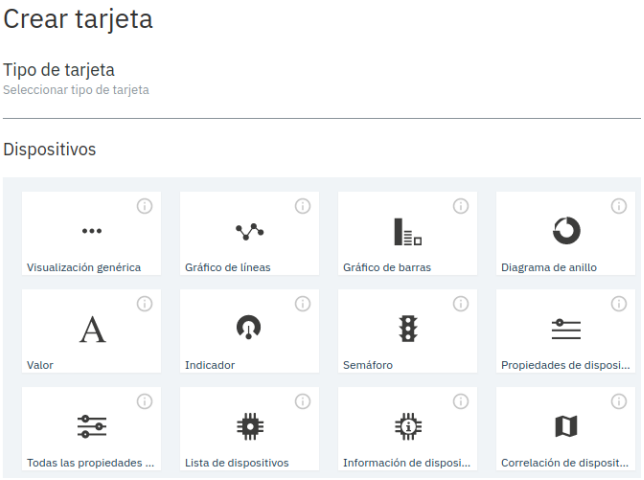


Figura 101: Tipos de tarjetas disponibles

Para presentar la información en el panel, se seleccionan las tarjetas de gráfico de líneas y Valor. A continuación, en la Figura 102, se muestra la configuración de la tarjeta de diagrama de líneas, en la que en primer lugar se debe especificar el origen de datos seleccionando el dispositivo desde el que se obtienen los valores, en este caso se analiza la RAM en uso.

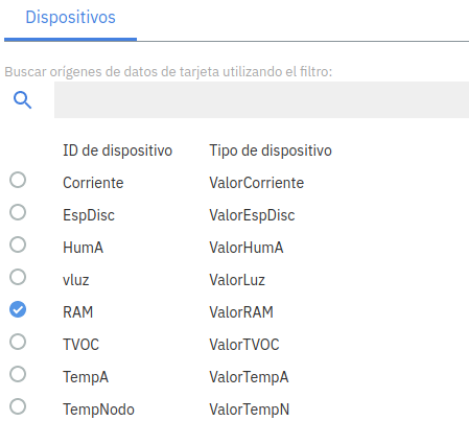


Figura 102: Configuración de tarjeta tipo Gráfico de líneas

Cuando se ha seleccionado el origen de datos, se debe conectar con un nuevo conjunto de datos en el que se debe especificar el suceso, propiedad, nombre, tipo y unidad en este caso se tiene *event, value*, RAM en uso, Número y kilobytes respectivamente tal como se observa en la Figura 103. El valor mínimo y máximo se ajustan automáticamente, por lo que se deja en los valores por defecto.

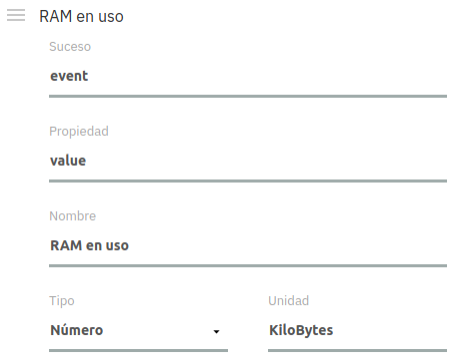


Figura 103: Configuración de conjunto de datos

Una vez que se ha conectado a un conjunto de datos, se puede elegir el tamaño que tendrá la tarjeta en el panel, las opciones que se pueden elegir son S, M, L y XL. En este caso, se selecciona un tamaño de tipo M, ya que permite visualizar mayor información. Por último, se especifica un título al gráfico y se finaliza el proceso, así esta tarjeta se agrega al panel de datos y se puede monitorizar la variable de RAM en uso por medio del diagrama de líneas configurado, como se muestra en la Figura 104.

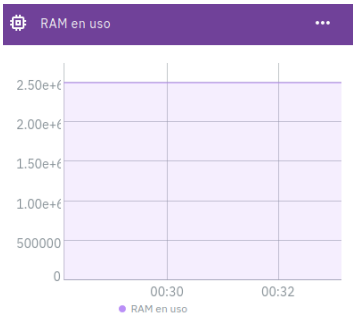


Figura 104: Tarjeta tipo Gráfico de líneas correspondiente a uso de la RAM

Para la tarjeta de tipo valor se realiza un proceso similar al realizado para el gráfico de líneas, ya que se debe registrar un origen de datos, seleccionar el tamaño de la tarjeta y asignarle un título. En la Figura 105, se puede observar la tarjeta en el panel de datos.

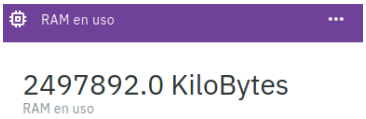


Figura 105: Tarjeta tipo Valor correspondiente a uso de la RAM

Al final el panel de datos, mostrara información relacionada a la RAM en uso por medio de un gráfico de líneas y una tarjeta que muestra el valor obtenido con las unidades de medición. La Figura 106, muestra el panel implementado para la monitorización de la variable antes mencionada.

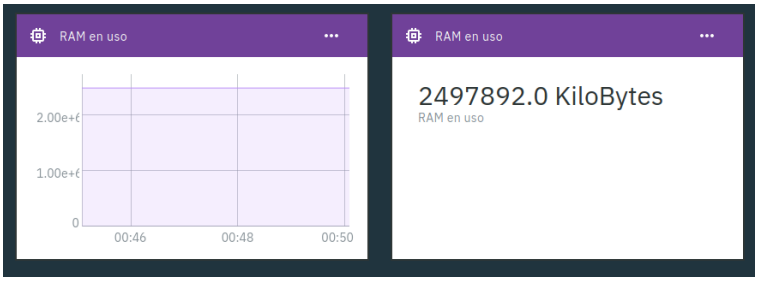


Figura 106: Panel para monitorización de la RAM en uso

Apéndice E: Enrutamiento estático y OLSR de una red multi-salto

E.1 Enrutamiento Estático

Se presenta el código de enrutamiento para cada nodo añadiendo con el comando *route* y la configuración del *IP forwarding* para permitir el paso de los paquetes a través de las interfaces

#Nodo 1

```
#! /bin/bash
ip route flush dev eth1
route add 10.10.10.1 dev eth1
route add 10.10.10.2 gw 10.10.10.1 metric 1 dev eth1
route add 10.10.10.3 gw 10.10.10.2 metric 2 dev eth1
route add 10.10.10.4 gw 10.10.10.2 metric 3 dev eth1
sysctl net.ipv4.ip_forward=1
```

#Nodo 2

```
#! /bin/bash
ip route flush dev eth1
route add 10.10.10.2 dev eth1
route add 10.10.10.1 gw 10.10.10.2 metric 1 dev eth1
route add 10.10.10.3 gw 10.10.10.2 metric 1 dev eth1
route add 10.10.10.4 gw 10.10.10.3 metric 2 dev eth1
sysctl net.ipv4.ip_forward=1
```

#Nodo 3

```
#! /bin/bash
ip route flush dev eth1
route add 10.10.10.3 dev eth1
route add 10.10.10.2 gw 10.10.10.3 metric 1 dev eth1
route add 10.10.10.4 gw 10.10.10.3 metric 1 dev eth1
route add 10.10.10.1 gw 10.10.10.2 metric 2 dev eth1
sysctl net.ipv4.ip_forward=1
```

#Nodo 4

```
#! /bin/bash
ip route flush dev eth1
route add 10.10.10.4 dev eth1
```

route add 10.10.10.3 gw 10.10.10.4 metric 1 dev eth1
route add 10.10.10.2 gw 10.10.10.3 metric 2 dev eth1
route add 10.10.10.1 gw 10.10.10.3 metric 3 dev eth1
sysctl net.ipv4.ip_forward=1

E.2 Enrutamiento OLSR

Para el enrutamiento OLSR se requiere la instalación de algunos paquetes de software y configurar el demonio.

git clone https://github.com/OLSR/olsrd.git
apt install build-essential flex bison
cd olsrd
make
make install
cp olsrd.conf etc/olsrd
cd /olsrd/lib/txtinfo/
make
make install
Cambiar en el olsrd.conf lo siguiente LoadPlugin "olsrd_txtinfo.so.0.1" por LoadPlugin "olsrd_txtinfo.so.1.1"
olsrd -i INTERFACE

Para ver las tablas de enrutamiento se utiliza:

apt install netcat
echo /all | nc localhost 2006

Apéndice F: Tablas de resultados de simulaciones en NS3

F.1 Resultados del escenario 1: Red multi-salto con enrutamiento estático

Throughput [kbps]	Simulaciones												
Tiempo [segundos]	1	2	3	4	5	6	7	8	9	10	Throughput Promedio	Desviación Estándar	Intervalo Confianza
30	5,1990	4,1510	6,4290	0,2030	4,0250	8,9040	2,8500	9,4020	9,2570	8,8900	5,9310	3,1727	1,9664
90	11,8650	11,6620	10,2690	10,6680	11,8700	9,3130	9,3220	10,0110	9,0990	9,9720	10,4051	1,0729	0,6650
150	10,5780	10,9100	12,9010	11,3720	11,1980	10,4020	9,4440	10,1480	10,7420	9,6500	10,7345	0,9813	0,6082
210	10,7490	10,5340	11,2680	10,9880	11,5800	9,6250	9,8590	9,9820	9,5980	9,6430	10,3826	0,7388	0,4579
270	10,8130	12,1330	11,1850	11,1480	11,1160	9,2270	9,4560	12,4980	10,5940	9,2090	10,7379	1,1472	0,7110
330	10,5620	10,6450	12,4550	11,5530	11,8100	9,4370	11,0930	9,9180	9,0720	9,5230	10,6068	1,1241	0,6967
390	11,0270	10,4720	9,9690	11,9390	11,1860	10,0330	9,4840	9,6260	9,6140	9,2500	10,2600	0,8748	0,5422
450	12,6960	11,5650	11,2170	10,8650	11,2880	10,1040	9,5220	8,9450	10,6480	8,9000	10,5750	1,2148	0,7529
510	9,9160	10,8900	11,1290	12,5540	11,2160	6,3480	10,3960	9,6120	6,2280	10,6530	9,8942	2,0619	1,2779
570	12,5460	11,6700	12,2220	11,4460	11,5590	4,1760	9,8540	9,5420	4,0370	10,0860	9,7138	3,1203	1,9339
Promedio Total											10,3678	0,7549	0,4932

Tabla 22: Throughput del envío de paquetes en la red

	Simulaciones										
Pérdida de paquetes	1	2	3	4	5	6	7	8	9	10	Promedio
Capturados Nodo 1	3690	3712	3843	3551	3827	2637	2738	3031	2709	2903	3264,10
Capturados Nodo 4	3535	3597	3651	3431	3571	2500	2601	2770	2508	2828	3099,20
Perdidos	155	115	192	120	256	137	128	261	201	75	164,00
Porcentaje de Recepción [%]	95,80	96,90	95,00	96,62	93,31	94,80	95,32	91,38	92,58	97,41	94,91

Tabla 23: Pérdida de paquetes entre los extremos de la red (Nodo 1 y Nodo 4)

	Simulaciones										
Delay [segundos]	1	2	3	4	5	6	7	8	9	10	Promedio
Mínimo	0,0021	0,0021	0,0026	0,0019	0,0025	0,0027	0,0031	0,0030	0,0030	0,0030	0,0026
Máximo	0,2453	0,2495	0,2461	0,2417	0,2353	0,2592	0,1168	0,2295	0,1989	0,2785	0,2301
Promedio	0,0893	0,0953	0,0815	0,0830	0,0855	0,0375	0,0347	0,0350	0,0358	0,0361	0,0614

Tabla 24: Retardo de paquetes entre los extremos de la red (Nodo 1 y Nodo 4)

F.2 Resultados del escenario 2: Red multi-salto con enrutamiento OLSR

Throughput [kbps]	Simulaciones												
Tiempo [segundos]	1	2	3	4	5	6	7	8	9	10	Throughput Promedio	Desviación Estándar	Intervalo Confianza
30	1,7000	4,6150	1,7690	3,1030	1,6640	3,4280	12,7620	8,5410	7,4260	1,8530	4,6861	3,7456	2,3215
90	12,2760	12,0740	12,6180	3,7300	12,3970	12,8860	10,2160	11,7630	9,3640	19,3380	11,6662	3,8401	2,3801
150	11,9470	12,3140	12,6040	8,3750	12,3900	10,9520	5,9560	11,3870	10,8110	10,3630	10,7099	2,0873	1,2937
210	12,2720	11,8380	12,6470	12,8330	12,5050	12,3440	6,4250	10,8260	11,6990	11,5220	11,4911	1,8790	1,1646
270	13,0180	12,0670	11,8910	11,8510	12,3460	10,4930	10,7680	10,4670	10,4050	11,5530	11,4859	0,9089	0,5633
330	11,8090	11,6900	11,8170	13,0960	11,8010	10,6400	11,2720	11,0070	11,3870	5,2880	10,9807	2,1032	1,3035
390	12,2650	13,1180	12,2590	10,2850	12,1700	12,2150	10,4020	11,3080	11,4890	7,6330	11,3144	1,5659	0,9705
450	12,0090	12,6890	13,7410	13,9900	12,5200	10,9670	11,5160	11,2560	10,6640	10,2900	11,9642	1,2600	0,7809
510	11,7460	12,3280	12,3180	11,5770	11,9720	10,3130	11,3900	11,1830	10,7500	10,6430	11,4220	0,6993	0,4334
570	12,3230	12,0090	11,8400	12,6390	11,8210	11,4950	11,7120	10,9290	11,0470	12,1340	11,7949	0,5334	0,3306
Promedio Total											11,4255	1,0065	0,6576

Tabla 25: Throughput del envío de paquetes en la red

	Simulaciones										
Pérdida de paquetes	1	2	3	4	5	6	7	8	9	10	Promedio
Capturados Nodo 1	3082	3567	3377	3108	3103	2917	2711	3003	2857	2795	2983,88
Capturados Nodo 4	3051	3526	3168	2967	3027	2709	2509	2780	2695	2575	2803,75
Perdidos	31	41	209	141	76	208	202	223	162	220	180,13
Porcentaje de Recepción [%]	98,99	98,85	93,81	95,46	97,55	92,86	92,54	92,57	94,32	92,12	93,90

Tabla 26: Pérdida de paquetes entre los extremos de la red (Nodo 1 y Nodo 4)

	Simulaciones										
Delay [segundos]	1	2	3	4	5	6	7	8	9	10	Promedio
Mínimo	0,0011	0,0011	0,0010	0,0010	0,0013	0,0031	0,0026	0,0026	0,0027	0,0033	0,0020
Máximo	0,2475	0,2499	0,2477	0,2410	0,2441	0,2192	0,11802	0,2216	0,2085	0,2003	0,2198
Promedio	0,0801	0,0871	0,0850	0,0879	0,0899	0,0284	0,0259	0,0270	0,0260	0,0261	0,0563

Tabla 27: Retardo de paquetes entre los extremos de la red (Nodo 1 y Nodo 4)

F.3 Resultados del escenario 3: Red multi-salto con nodo móvil y enrutamiento OLSR

Throughput [kbps]	Simulaciones												
Tiempo [segundos]	1	2	3	4	5	6	7	8	9	10	Throughput Promedio	Desviación Estándar	Intervalo Confianza
30	12,8000	13,5480	13,0690	12,4070	12,4900	18,3950	1,8530	17,0540	5,1920	15,9330	12,2741	5,1105	3,1674
90	8,8070	8,5900	8,2300	8,4550	8,7010	4,5600	4,2780	2,2360	13,7000	6,3750	7,3932	3,1991	1,9828
150	12,2930	13,0950	13,5160	12,6790	12,7700	19,9500	14,1190	18,9710	11,9620	15,4120	14,4767	2,8132	1,7436
210	12,4410	12,4740	12,5220	13,4600	13,5380	3,2160	12,9120	4,8230	13,6540	9,9420	10,8982	3,7936	2,3512
270	13,3690	9,6750	16,6670	9,4160	15,5840	10,4220	10,3440	10,8780	10,0140	10,3140	11,6683	2,5983	1,6104
330	12,7540	17,7590	14,7680	16,4730	13,6370	14,1440	2,6090	12,1880	2,6120	8,7450	11,5689	5,3101	3,2912
390	12,4620	12,6110	12,4570	12,2770	12,4010	6,7230	15,4520	8,4570	16,0770	11,7780	12,0695	2,7883	1,7282
450	12,4400	12,6190	12,6170	12,5810	12,5280	13,3900	4,2280	12,2630	4,0070	10,5740	10,7247	3,5534	2,2023
510	2,7350	2,7540	2,7150	3,4210	2,7350	2,3610	10,6800	2,3550	10,9840	2,3610	4,3101	3,4521	2,1396
570	5,9020	6,2350	5,9940	6,1240	5,9480	2,5590	2,5800	2,5800	2,5970	2,6020	4,3121	1,8243	1,1307
Promedio Total											9,9696	1,0863	0,6733

Tabla 28: Throughput del envío de paquetes en la red

	Simulaciones										
Pérdida de paquetes	1	2	3	4	5	6	7	8	9	10	Promedio
Capturados Nodo 1	2681	2861	2918	2841	2889	2740	2144	2669	2778	2622	2724,56
Capturados Nodo 5	1125	1418	1544	1395	1510	731	1087	1409	777	1602	1221,78
Perdidos	1556	1443	1374	1446	1379	2009	1057	1260	2001	1020	1502,78
Porcentaje de Recepción [%]	41,96	49,56	52,91	49,10	52,27	26,67	50,69	52,79	27,96	61,09	44,88

Tabla 29: Pérdida de paquetes entre los extremos de la red (Nodo 1 y Nodo 5)

	Simulaciones										
Delay [segundos]	1	2	3	4	5	6	7	8	9	10	Promedio
Mínimo	0,0006	0,0006	0,0010	0,0010	0,0010	0,0018	0,0016	0,0010	0,0014	0,0012	0,0011
Máximo	0,1247	0,1246	0,1176	0,1164	0,1245	0,1261	0,0634	0,1340	0,0852	0,0536	0,1070
Promedio	0.0446	0,0550	0,0487	0,0382	0,0576	0,0260	0,0220	0,0249	0,0230	0,0200	0,0350

Tabla 30: Retardo de paquetes entre los extremos de la red (Nodo 1 y Nodo 5

Apéndice G: Medición del ancho de banda de la red multi-salto

La medición del ancho de la red multi-salto se realiza utilizando la herramienta iperf. Para su instalación se utiliza el comando *apt install iperf* que se utiliza como un modelo cliente-servidor. La herramienta permite crear flujos de tráfico, para los fines requeridos se utiliza tráfico UDP que es el protocolo que utiliza SNMP.

El nodo 1 envía tráfico seleccionado por el usuario como cliente con el parámetro *-c*, el parámetro -i muestra el tráfico cada cierto tiempo en este caso cada segundo y finalmente el *-b* que es el tráfico que se va a enviar *(K kilobits, M megabits)*, como se muestra con el comando:

Iperf -u -c 10.10.10.2 -i 1 -b 10M

Los nodos 2, 3 y 4 actúan como servidores del tráfico con el parámetro *-s*, el parámetro *-u* permite enviar tráfico UDP, con el comando:

Iperf -s -u

Se realiza el envío aumentando la carga para cada experimento, hasta llegar a un valor umbral en el que la capacidad es la máxima para cada salto de la red. Como se presenta en la Figura 107, el tráfico se muestra el tiempo en la primera columna en este caso cada segundo, en la segunda el tráfico enviado y en la tercera el ancho de banda enviado. En la última línea se muestra el reporte del servidor junto con el número de paquetes que se han recibido.

```
root@nodo1:/# iperf -u -c 10.10.10.2 -i 1 -b 10M
------------------------------------------------------------
Client connecting to 10.10.10.2, UDP port 5001
Sending 1470 byte datagrams, IPG target: 1121.52 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.10.10.1 port 49951 connected with 10.10.10.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.25 MBytes  10.5 Mbits/sec
[  3]  1.0- 2.0 sec  1.25 MBytes  10.5 Mbits/sec
[  3]  2.0- 3.0 sec  1.25 MBytes  10.5 Mbits/sec
[  3]  3.0- 4.0 sec  1.25 MBytes  10.5 Mbits/sec
[  3]  4.0- 5.0 sec  1.25 MBytes  10.5 Mbits/sec
[  3]  5.0- 6.0 sec  1.25 MBytes  10.5 Mbits/sec
[  3]  6.0- 7.0 sec  1.25 MBytes  10.5 Mbits/sec
[  3]  7.0- 8.0 sec  1.25 MBytes  10.5 Mbits/sec
[  3]  8.0- 9.0 sec  1.25 MBytes  10.5 Mbits/sec
[  3]  0.0-10.0 sec  12.5 MBytes  10.5 Mbits/sec
[  3] Sent 8917 datagrams
[  3] Server Report:
[  3]  0.0-10.5 sec  2.78 MBytes  2.22 Mbits/sec   1.266 ms 6936/ 8916
```

Figura 107: Resultado del experimento con iperf para el cliente

En la Figura 108, se muestra el servidor de la misma manera con el orden de las columnas, el tiempo en la primera columna en este caso cada segundo, en la segunda el tráfico recibido

y en la tercera el ancho de banda recibido. Además, el *jitter*, los paquetes perdidos y el porcentaje de datagramas recibidos.

```
^Croot@nodo2:/# iperf -u -s -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.10.10.2 port 5001 connected with 10.10.10.1 port 49951
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec   175 KBytes  1.43 Mbits/sec   1.333 ms  332/  454 (73%)
[  3]  1.0- 2.0 sec   270 KBytes  2.21 Mbits/sec   1.014 ms  706/  894 (79%)
[  3]  2.0- 3.0 sec   290 KBytes  2.38 Mbits/sec   1.660 ms  685/  887 (77%)
[  3]  3.0- 4.0 sec   289 KBytes  2.36 Mbits/sec   0.936 ms  697/  898 (78%)
[  3]  4.0- 5.0 sec   270 KBytes  2.21 Mbits/sec   0.864 ms  695/  883 (79%)
[  3]  5.0- 6.0 sec   313 KBytes  2.56 Mbits/sec   1.107 ms  696/  914 (76%)
[  3]  6.0- 7.0 sec   266 KBytes  2.18 Mbits/sec   1.012 ms  690/  875 (79%)
[  3]  7.0- 8.0 sec   286 KBytes  2.34 Mbits/sec   0.769 ms  689/  888 (78%)
[  3]  8.0- 9.0 sec   260 KBytes  2.13 Mbits/sec   1.240 ms  704/  885 (80%)
[  3]  9.0-10.0 sec   278 KBytes  2.28 Mbits/sec   1.068 ms  715/  909 (79%)
[  3]  0.0-10.5 sec  2.78 MBytes  2.22 Mbits/sec   1.267 ms 6936/ 8916 (78%)
```

Figura 108: Resultado del experimento con iperf para el servidor

Este experimento se realiza para cada uno de los saltos con los resultados que se presentaron en el Capítulo 6.

Apéndice H: Extensión del agente SNMP

En este apéndice se muestra el proceso para la extensión del agente SNMP, que puede ser de dos maneras como son la ejecución arbitraria de un *script* y la extensión de la MIB.

H.1 Ejecución arbitraria de un script

Para extender el agente SNMP, usando la ejecución arbitraria de un *script* se modifica el archivo de configuración /etc/snmp/snmpd.conf y se añade la sentencia *extend*, en conjunto con un nombre para identificar al *script* y su ubicación, por ejemplo, para encender un sensor se da el nombre *sensor_on* y la instrucción resultante es la siguiente:

extend sensor_on /home/pi/sensor_on.sh

De esta manera se realiza esta configuración para cada uno de los *scripts* de gestión del nodo y de los sensores.

Adicionalmente, se deben dar permisos para ejecutar en el archivo /etc/*sudoers.* A este archivo se puede acceder con el comando *sudo visudo*, para luego añadir la línea *Debian-snmp ALL=(ALL) NOPASSWD:ALL.* Luego de realizar las modificaciones el archivo queda como se presenta en la Figura 109.

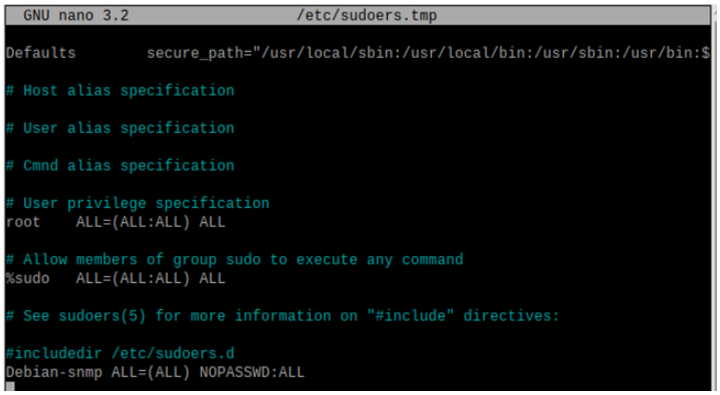


Figura 109: Configuración del archivo sudoers

El *script* se puede ejecutar utilizando la sentencia:

'NET-SNMP-EXTEND-MIB::nsExtendOutLine."sensor_on".1'

Mediante el comando:

snmpget -u pi -l authPriv -a MD5 -x AES -A 12345678 -X 12345678 localhost 'NET-SNMP-EXTEND-MIB::nsExtendOutLine."sensor_on".1'

Se puede obtener el OID para ejecutar el *script* utilizando *snmptranslate* con el siguiente comando:

snmptranslate -On 'NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."sensor_on".1'

Con el OID obtenido, no se debe tomar en cuenta el último dígito ya que no pertenece al OID, por ejemplo, si obtenemos:

.1.3.6.1.4.1.8072.1.3.2.3.1.1.7.112.114.111.99.101.115.111.1

El OID del *script* sería: *.1.3.6.1.4.1.8072.1.3.2.3.1.1.7.112.114.111.99.101.115.111*

Otra opción es ejecutar *snmpwalk* y ver que OID pertenece al *script* creado.

H.2 Extensión de la MIB

La segunda posibilidad es mediante la extensión de la MIB al otorgar un OID por el usuario a un *script*. Esto se realiza mediante la función *pass-through* del archivo *snmpd*, a diferencia del anterior método que permite ejecutar cualquier tipo de script, los archivos de tipo *pass-through* cumplen con reglas como:

- Una solicitud snmpget, pasa un parámetro "-g" *get* hacia el agente.
- Una solicitud snmpset pasa un parámetro "-s" *set* hacia el agente.
- La solicitud también debe contener el OID y el tipo de dato (*string, integer, boolean).*
- Como respuesta se obtiene el valor de la variable.

Una vez realizado el *script*, se modifica el archivo de configuración del agente en el archivo de configuración *snmpd.conf* se debe añadir la extensión del OID, mediante el código:

pass .1.3.6.1.4.1.8072.2.1 /bin/bash /home/pi/temp_nodo.sh

En este caso para la medición de temperatura del nodo, como se observa el OID se otorga por el usuario, por último, es necesario reiniciar el demonio mediante *sudo service snmpd restart.*

Apéndice I: Configuración de la red Ad Hoc y el gateway por defecto

En este apéndice se muestran los códigos para realizar la configuración de la red Ad Hoc, asi como también el proceso de configuración de un *gateway* por defecto que permite la conexión hacia el Internet.

I.1 Configuración de la red Ad Hoc física mediante el archivo interfaces

```
#Configuración en el ordenador

auto wlo1
iface wlo1 inet static
wireless-mode ad-hoc
wireless-channel 1
wireless-essid adhocsnmp
wireless-key 1234567890
address 10.10.10.2
netmask 255.255.255.0

#Configuración en la Raspberry Pi

auto wlan0
iface wlan0 inet static
wireless-mode ad-hoc
wireless-channel 1
wireless-essid adhocsnmp
#Encriptación WEP
wireless-key 1234567890
address 10.10.10.3
netmask 255.255.255.0
```

I.2 Configuración del gateway por defecto en el ordenador con la herramienta iptables

```
#Configuración en el ordenador

sysctl net.ipv4.ip_forward=1
sudo /sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo /sbin/iptables -A FORWARD -i eth0 -o wlo1 -m state --state
RELATED,ESTABLISHED -j ACCEPT
sudo /sbin/iptables -A FORWARD -i wlo1 -o eth0 -j ACCEPT
sudo bash -c 'iptables-save > /etc/network/iptables'
```

Apéndice J: Diseño del encapsulado para el nodo agente

Se diseño un encapsulado para proteger el nodo Agente. Este se conforma de la Raspberry Pi, una placa donde se colocan los sensores, la batería y los cables de conexión.

El diseño está impreso en 3D como se observa en la Figura 110, con las siguientes dimensiones.

- Largo: 118 mm
- Ancho: 120 mm
- Alto: 60 mm

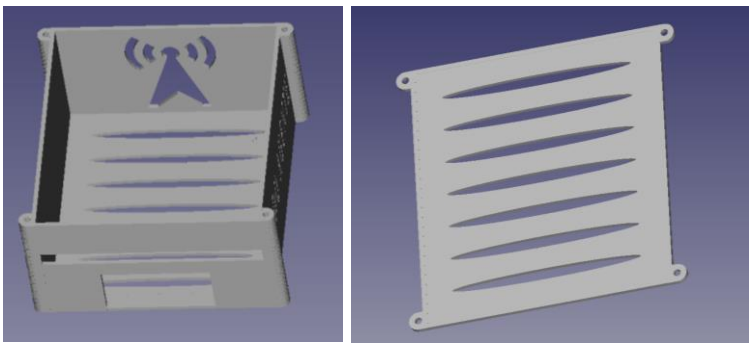


Figura 110: Diseño en 3D del encapsulado

En la Figura 111, se muestra el resultado final del encapsulado ensamblado con la Raspberry Pi, la placa con los sensores y los cables de conexión.

Figura 111: Encapsulado del nodo agente

Apéndice K: Instalación y configuración de los contenedores Linux

La instalación de las herramientas para el uso de LXC, se realiza con el paquete *lxc.* El resto de paquetes necesarios vienen por defecto en las distribuciones Linux actuales, sin embargo, en caso de requerirse se listan a continuación y deben ser instalados con el comando:

apt install.lxc-templates lxc-extra debootstrap libvirt perl gpg tunctl bridge-utils

Una vez instaladas las herramientas se deben iniciar los servicios para los contenedores, para lo cual se emplean las siguientes instrucciones desde una terminal:

sudo systemctl start libvirtd.service
sudo systemctl start lxc.service
sudo systemctl enable lxc.service

La configuración de un contenedor se realiza con un archivo .conf, por ejemplo, un fichero nombrado como *nodo1.conf*, este proceso se realiza para cada nodo con las siguientes características.

lxc.net.1.type = veth
lxc.net.1.flags = up
lxc.net.1.link = br-nodo1
lxc.net.1.ipv4.address = 10.10.10.1/24

En el directorio donde se encuentran los archivos *.conf* de cada nodo se debe instalar el contenedor con el sistema operativo para este caso se utiliza el sistema operativo *Debian Buster*.

cd src/tap-bridge/adhocmulti-salto
sudo lxc-create -f lxc-nodo1.conf -t download -n nodo1 -- -d debian -r buster -a amd64

Es posible definir una contraseña para el contenedor con el comando.

sudo chroot /var/lib/lxc/nodo1/rootfs/passwd

A continuación, se configuran los *Bridge* y dispositivos TAP de la siguiente manera:

sudo brctl addbr br-nodo1
sudo tunctl -t tap-nodo1
sudo ifconfig tap-nodo1 0.0.0.0 promisc up
sudo brctl addif br-nodo1 tap-nodo1
sudo ifconfig br-nodo1 up
sudo lxc-start -n nodo1

Para ingresar a un contenedor desde la terminar se utiliza el comando *sudo lxc-attach –n nodo1* para el caso del Nodo 1. Adicionalmente, se puede configurar un acceso a Internet para

el contenedor con el que se pueden descargar aplicaciones, existen dos maneras para realizar esta configuración.

En el archivo *.conf* al momento de crear el contenedor se debe configurar la interfaz, en el campo *hwaddr y* se coloca la dirección MAC de la tarjeta de red del ordenador.

lxc.net.0.type = veth
lxc.net.0.flags = up
lxc.net.0.link = lxcbr0
lxc.net.0.hwaddr = 50:3e:aa:2e:7b:df

De forma alternativa una vez que ha sido creado el contenedor en el directorio del contenedor *var/lib/lxc* se modifica el archivo *config* para crear la interfaz con el código anteriormente mostrado.

Una vez ha sido modificado, se procede al archivo *interfaces* que se encuentra en el directorio */etc/network/interfaces* en el que se debe añadir las siguientes líneas.

auto eth0
iface eth0 inet dhcp
auto eth1
iface eth1 inet dhcp

Esta configuración se realiza para todos los nodos de la red multi-salto.

Para finalizar un contenedor y el puente de red se utilizan los siguientes comandos:

sudo lxc-stop -n nodo1 –k
sudo ifconfig br-nodo1 down
sudo brctl delif br-nodo1 tap-nodo1
sudo brctl delbr br-nodo1
sudo ifconfig tap-nodo1 down
sudo tunctl -d tap-nodo1

Apéndice L: Configuración para las simulaciones en NS3

L.1 Configuración de la capa física en NS3

En el archivo de simulación se debe configurar de la siguiente manera para el estándar mencionado en el Capítulo 6:

```
WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211n_2_4GHZ);
wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager","DataMode", StringValue ("HtMcs7"));
```

L.2 Uso de un contendor Linux o el ordenador en la simulación en NS3

Para utilizar un nodo creado, en este caso mediante contenedores, se inicializa de la siguiente manera:

```
tapBridge.SetAttribute ("DeviceName", StringValue ("tap-nodo1"));
tapBridge.Install (nodes.Get (0), devices.Get (0));
```

De manera similar para utilizar el ordenador como uno de los nodos se utiliza el siguiente código:

```
TapBridgeHelper tapBridge;
tapBridge.SetAttribute ("Mode", StringValue ("UseLocal"));
tapBridge.Install (nodes.Get (0), devices.Get (0));
```

L.3 Posicionamiento de los nodos

La configuración del posicionamiento de los nodos se realiza en el eje X ya que la topología que se sigue es lineal, con una separación de 23 metros entre nodos tal como se presentó en el Capítulo 6, de la siguiente manera:

```
MobilityHelper mobility;
 Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
  positionAlloc->Add (Vector (0.0, 0.0, 0.0));
  positionAlloc->Add (Vector (23.0, 0.0, 0.0));
  positionAlloc->Add (Vector (46.0, 0.0, 0.0));
  positionAlloc->Add (Vector (69.0, 0.0, 0.0));
  mobility.SetPositionAllocator (positionAlloc);
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
```

L.4 Movilidad del nodo

Para esto se utiliza la herramienta de movilidad de NS3 *mobility.SetMobilityModel ("ns3::ConstantVelocityMobilityModel");*

En el código de NS3 se definen las variables de movilidad:

```
{
 Ptr<ConstantVelocityMobilityModel>mobility=node->GetObject<ConstantVelocityMobilityModel> ();
   mobility->SetVelocity (vel);
}
```

Finalmente se configura, la movilidad del nodo 5:

Simulator::Schedule (Seconds (0.0), &SetVelocity, nodes.Get (4),Vector (-0.153,0.0,0.0));

Bibliografía y referencias

[1] D. Jiang, "The construction of smart city information system based on the Internet of Things and cloud computing," *Comput. Commun.*, vol. 150, no. November 2019, pp. 158–166, 2020.

[2] T. hoon Kim, C. Ramos, and S. Mohammed, "Smart City and IoT," *Futur. Gener. Comput. Syst.*, vol. 76, no. July 2014, pp. 159–162, 2017.

[3] C. Badii, P. Bellini, A. Difino, P. Nesi, G. Pantaleo, and M. Paolucci, "Microservices suite for smart city applications," *Sensors (Switzerland)*, vol. 19, no. 21, 2019.

[4] S. Trilles, A. Calia, Ó. Belmonte, J. Torres-Sospedra, R. Montoliu, and J. Huerta, "Deployment of an open sensorized platform in a smart city context," *Futur. Gener. Comput. Syst.*, vol. 76, pp. 221–233, 2017.

[5] W. Ni, W. Wu, and K. Li, "A message efficient intersection control algorithm for intelligent transportation in smart cities," *Futur. Gener. Comput. Syst.*, vol. 76, pp. 339–349, 2017.

[6] A. Mir and A. Khachane, "Sensing Harmful Gases in Industries Using IOT and WSN," *Proc. - 2018 4th Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2018*, pp. 1–3, 2018.

[7] J. V. Alamelu and A. Mythili, "Design of IoT based generic health care system," *2017 Int. Conf. Microelectron. Devices, Circuits Syst. ICMDCS 2017*, vol. 2017-Janua, pp. 1–4, 2017.

[8] M. A. Mahmud, K. Bates, T. Wood, A. Abdelgawad, and K. Yelamarthi, "A complete Internet of Things (IoT) platform for Structural Health Monitoring (SHM)," *IEEE World Forum Internet Things, WF-IoT 2018 - Proc.*, vol. 2018-Janua, pp. 275–279, 2018.

[9] M. Lescisin and Q. H. Mahmoud, "Ad-hoc messaging infrastructure for peer-to-peer communication," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 1, pp. 60–73, 2019.

[10] S. Iqbal, K. N. Qureshi, N. Kanwal, and G. Jeon, "Collaborative energy efficient zone-based routing protocol for multihop Internet of Things," *Trans. Emerg. Telecommun. Technol.*, no. November 2019, pp. 1–16, 2020.

[11] M. Lekić and G. Gardašević, "IoT sensor integration to Node-RED platform," *2018 17th Int. Symp. INFOTEH-JAHORINA, INFOTEH 2018 - Proc.*, vol. 2018-Janua, no. March, pp. 1–5, 2018.

[12] A. E. Alves, H. V. Barreto Câmara, J. V. Da Silva, and R. Araújo De Souza, "Development of an automatic shutdown system for lighting and air conditioning by using Esp8266 to meet energy efficiency requirements in buildings," *2019 IEEE PES Conf. Innov. Smart Grid Technol. ISGT Lat. Am. 2019*, 2019.

[13] R. Benhamadi, M. Bouhedda, B. Bengherbia, H. Benyezza, and O. Benzineb, "IoT-Based System for Supervision and Control of a Transmission Center," *Proc. - 2019 3rd*

Int. Conf. Appl. Autom. Ind. Diagnostics, ICAAID 2019, vol. 1, no. September, pp. 1–5, 2019.

[14] L. C. Mota, E. D. Moreno, A. L. Ribeiro, and R. J. P. B. Salgueiro, "A comparative analysis of network management protocols in IoT applications," *J. Comput. Sci.*, vol. 14, no. 9, pp. 1238–1246, 2018.

[15] G. Suciu, C. Butca, A. Vulpe, and V. Suciu, "Simple Network Management Protocol for Remote Telemetry Systems in Urban Environments," *Springer Int. Publ. AG*, vol. 569, pp. 257–266, 2017.

[16] J. W. Abijaude, L. Santiago, P. De Lima Sobreira, O. A. Wahab, and F. Greve, "IoTCocoa - An IoT platform to assist gourmet cocoa production," *Proc. - 2019 IEEE Latin-American Conf. Commun. LATINCOM 2019*, 2019.

[17] M. Eremia, L. Toma, and M. Sanduleac, "The Smart City Concept in the 21st Century," *Procedia Eng.*, vol. 181, pp. 12–19, 2017.

[18] European investment bank insitute, "Smart cities : Concept , challenges and projects ascimer document," *Assess. smart city Initiat. Mediterr. Reg.*, p. 31, 2017.

[19] I. Unión Internacional de Telecomunicaciones, "Descripción general de Internet de los objetos," p. 20, 2012.

[20] D. Mauro and K. Schmidt, "Essential SNMP, 2nd Edition," O'Reilly Media, pp. 1–81, 2009.

[21] M. Fedor, J. Case, and M. Schoffstall, "RFC 1157 - Simple Network Management Protocol SNMP," 1990.

[22] M. Rose and K. McCloghrie, "RFC 1155 - Structure and identification of management information for TCP/IP-based internets," 1990.

[23] L. Molero, D. Gráfico, E. Aguirre Diagramación, and A. Martínez, "Evolución del protocolo de gestión internet," no. Rfc, p. 20,25, 2014.

[24] J. Case, R. Mundy, D. Partanin, and B. Stewart, "RFC 3410 - Introduction and Applicability Statements for Internet-Standard Management Framework," 2002.

[25] U. Blumenthal and B. Wijnen, "RFC 3414 - User-based Security Model …USM— for version 3 of the Simple Network Management Protocol …SNMPv3—," 2002.

[26] B. Wijnen, R. Presuhn, and K. McCloghrie, "RFC 3415 - View-based Access Control Model …VACM— for the Simple Network Management Protocol …SNMP—," 2002.

[27] D. Harrington, "RFC 2271 - An Architecture for Describing SNMP Management Frameworks," 1998.

[28] L. Molero, D. Gráfico, E. Aguirre Diagramación, and A. Martínez, "Introducción a la gestión de redes," 2014.

[29] S. K. Narvaez, I.-E. Olivo, S. K. Narváez, and C. M. E. Inuca, "Administración y gestión de la red de área local con el protocolo SNMP."

[30] I. Jaime, A. González, M. Carlos, and A. Vanegas, "La Seguridad en las Redes de Comunicaciones."

[31] K. S. S., B. T.G, and P. C., *Ad Hoc Wireless Networks*, no. June. 2013.

[32] M. Conti and S. Giordano, "Mobile ad hoc networking: Milestones, challenges, and new research directions," *IEEE Commun. Mag.*, vol. 52, no. 1, pp. 85–96, 2014.

[33] C. de M. Cordeiro and D. P. Agrawal, *Ad hoc and sensor networks: Theory and applications*. 2011.

[34] D. Van Dinh, B. N. Yoon, H. N. Le, U. Q. Nguyen, K. D. Phan, and L. Di. Pham, "ICT Enabling Technologies for Smart Cities," *Int. Conf. Adv. Commun. Technol. ICACT*, vol. 2020, pp. 1180–1192, 2020.

[35] S. Nižetić, P. Šolić, D. López-de-Ipiña González-de-Artaza, and L. Patrono, "Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future," *J. Clean. Prod.*, vol. 274, p. 122877, 2020.

[36] M. Ghosal, A. Bobade, and P. Verma, "Digitized City with IoT," *Proc. 2nd Int. Conf. Comput. Methodol. Commun. ICCMC 2018*, no. Iccmc, pp. 877–880, 2018.

[37] A. Chehri, T. El Ouahmani, and N. Hakem, "Mining and IoT-based Vehicle Ad-hoc NETwork: Industry opportunities and innovation," *Internet of Things*, p. 100117, 2019.

[38] I. K. A. A. Aryanto, R. R. Huizen, and K. Y. E. Aryanto, "Design of Soil Humidity Monitoring System Using the Internet of Things Concept and MQTT," *Proceeding - ICoSTA 2020 2020 Int. Conf. Smart Technol. Appl. Empower. Ind. IoT by Implement. Green Technol. Sustain. Dev.*, 2020.

[39] G. Catargiu, E. H. Dulf, and L. C. Miclea, "Connected bike-smart IoT-based cycling training solution," *Sensors (Switzerland)*, vol. 20, no. 5, 2020.

[40] J. Han and S. Oh, "A Study of IoT Home Network Management System Using SNMP," *Int. J. Control Autom.*, vol. 11, no. 5, pp. 163–172, 2018.

[41] M. Savic, M. Ljubojevic, and S. Gajin, "A Novel Approach to Client-Side Monitoring of Shared Infrastructures," *IEEE Access*, vol. 8, pp. 44175–44189, 2020.

[42] Namrata, S. Ravi Shankar, and S. B. Kandukuri, "Network management using SNMP," *Int. J. Adv. Res. Eng. Technol.*, vol. 10, no. 3, pp. 81–86, 2019.

[43] M. Zeeshan, M. Z. Siddiqui, and F. Bin Rashid, "Design and Testing of SNMP/MIB based IoT Control API," *HONET-ICT 2019 - IEEE 16th Int. Conf. Smart Cities Improv. Qual. Life using ICT, IoT AI*, pp. 54–58, 2019.

[44] J. C. Liu, Y. Q. Ke, Y. C. Kao, S. C. Tsai, and Y. B. Lin, "A dual-stack authentication mechanism through SNMP," *J. Wirel. Mob. Networks, Ubiquitous Comput. Dependable Appl.*, vol. 10, no. 4, pp. 31–45, 2019.

[45] J. Chen, P. Shangguan, and X. Zhang, "A novel method of IoT wireless sensor nodes management based on OID technology," *J. Phys. Conf. Ser.*, vol. 1449, no. 1, 2020.

[46] P. J. Basford, F. M. J. Bulot, M. Apetroaie-Cristea, S. J. Cox, and S. J. J. Ossont, "LoRaWan for smart city IoT deployments: A long term evaluation," *Sensors (Switzerland)*, vol. 20, no. 3, 2020.

[47] S. Bansal and Di. Kumar, "IoT Application Layer Protocols: Performance Analysis and Significance in Smart City," *2019 10th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2019*, pp. 1–6, 2019.

[48] S. Sinche *et al.*, "A Survey of IoT Management Protocols and Frameworks," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 2, pp. 1168–1190, 2020.

[49] P. G. Pillai and S. P, "Significance of Wireless Multi-Hop AD-HOC Networks," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 8, pp. 161–167, 2018.

[50] I. Palacios, "Tecnologías IoT y Redes Inalámbricas de sensores aplicados a la monitorización de salud estructural en los edificios esenciales de la ciudad de Cuenca," 2020.

[51] I. W. H. Ho, P. P. Lam, P. H. J. Chong, and S. C. Liew, "Harnessing the high bandwidth of multiradio multichannel 802.11n mesh networks," *IEEE Trans. Mob. Comput.*, vol. 13, no. 2, pp. 448–456, 2014.