UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN

TEMA:

DISEÑO E IMPLEMENTACIÓN DE UNA RED DE SENSORES INALÁMBRICOS DEFINIDOS POR SOFTWARE PARA EL MONITOREO DE UN SISTEMA HIDROPÓNICO NFT.

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN ELECTRÓNICA Y REDES DE TELECOMUNICACIÓN

AUTOR: JUAN PABLO BAUTISTA SINCHICO

DIRECTOR: Ing. EDGAR ALBERTO MAYA OLALLA, MSc

IBARRA- ECUADOR

2022

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1003639497		
APELLIDOS Y NOMBRES:	Bautista Sinchico Juan Pablo		
DIRECCIÓN:	Luis Alberto de la Torre y Quiroga		
EMAIL:	jpbautistas@utn.edu.ec		
TELÉFONO FIJO:	062923-872	**TELÉFONO MÓVIL:**	0979983866

DATOS DE LA OBRA	
TÍTULO:	Diseño e implementación de una red de sensores inalámbricos definidos por software para el monitoreo de un sistema hidropónico NFT
AUTOR (ES):	Bautista Sinchico Juan Pablo
FECHA: DD/MM/AAAA	25-02-2022
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniería en Electrónica y Redes de Comunicación
ASESOR /DIRECTOR:	Ing. Edgar Alberto Maya Olalla, MSc

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 25 días del mes de febrero de 2022

AUTOR:

Firma .....
Nombre: Juan Pablo Bautista
Cédula: 100363949-7

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

Ing. EDGAR MAYA, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN

CERTIFICA

Que, el presente Trabajo de Titulación "Diseño e implementación de una red de sensores inalámbricos definidos por software para el monitoreo de un sistema hidropónico NFT" Ha sido desarrollado por el señor Bautista Sinchico Juan Pablo bajo mi supervisión.

Es todo en cuanto puedo certificar en honor a la verdad.

.....

Ing. Edgar Alberto Maya Olalla, MSc

DIRECTOR

DEDICATORIA

El desarrollo de esta tesis dedico a Dios por el apoyo incondicional en los momentos difíciles y permitirme compartir cada día con mis seres queridos, además dedico este trabajo a mis padres José Bautista y Margarita Sinchico por todo el sacrificio que hicieron para que mis metas se cumplan, por el apoyo incondicional y consejos que ayudaron a ser una buena persona.

A mis hermanas: Alicia, Vero y Cristina que siempre estuvieron pendientes y apoyándome en todo el transcurso de mi carrera universitaria

A Edith Peñafiel, gracias por cada uno de los momentos que hemos pasado a lo largo de nuestra vida estudiantil, por ser el incentivo para seguir adelante

Juan Pablo Bautista

AGRADECIMIENTO

Agradezco,

A mi director de tesis Ing. Edgar Maya que con su experiencia y conocimientos aportó su apoyo durante el desarrollo del presente trabajo,

Así también a los ingenieros Mauricio Domínguez y Fabián Cuzme que actuaron como asesores aportando importantes consejos para el desarrollo del presente proyecto.

Juan Pablo Bautista

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

En la actualidad las redes de sensores inalámbricos forman parte del internet de las cosas, cuya tecnología es ampliamente investigada por diferentes empresas tecnologías con el propósito de mejorar el funcionamiento de las redes WSN, sin embargo, estas redes presentan limitaciones de hardware como la capacidad de procesamiento, ancho de banda reducido y consumo de energía, estos factores afectan al procesamiento y envío de la información.

Ahora con base a este contexto, las redes definidas por software es una alternativa para mejorar las funcionalidades de las wsn debido a su principal característica, es que separar el plano de datos y control, por consiguiente, mejoraría el envío de datos, la escalabilidad y la administración de la red WSN debido a que mantiene un control centralizado a través del controlado SDN.

Por medio, investigación bibliográfica se obtuvieron parámetros tales como: protocolos de comunicación, dispositivos, métricas de evaluación, tipo de arquitectura, etc. Estos parámetros ayudaron a determinar una propuesta que integre las dos tecnologías SDN y WSN, que mediante el apoyo de la metodología en cascada permite el despliegue y ejecución del sistema propuesta.

La solución que se desarrolló en este trabajo está enfocada a la gestión de flujo de datos, donde se intenta aprovechar el direccionamiento IP local de las redes 6lowpan y el enrutamiento centralizado de las redes SDN, de manera que permita habilitar el flujo de datos por medio de políticas envidas por el controlador SDN.

La implementación de las SDN dentro de una red WSN tiene sus beneficios, y así que, de la evaluación realizada se concluye que la gestión de datos a través de la aplicación permitió mejorar el tiempo de respuesta y la entrega de datos entre el servidor y nodo sensor como también contribuye de forma general a la seguridad de la red WSN.

ABSTRACT

Currently, wireless sensor networks are part of the internet of things, whose technology is widely investigated by different technology companies with the purpose of improving the performance of WSN networks, however, these networks have hardware limitations such as processing capacity, reduced bandwidth and power consumption, these factors affect the processing and sending of information.

Now based on this context, software defined networking is an alternative to improve the functionalities of wsn due to its main characteristic, is that separating the data and control plane, therefore, it would improve the data delivery, scalability and management of the WSN network because it maintains a centralized control through the SDN controlled.

Through bibliographic research, parameters such as: communication protocols, devices, evaluation metrics, type of architecture, etc. were obtained. These parameters helped to determine a proposal that integrates the two technologies SDN and WSN, which through the support of the cascade methodology allows the deployment and implementation of the proposed system.

The solution developed in this work is focused on data flow management, where we try to take advantage of the local IP addressing of 6lowpan networks and the centralized routing of SDN networks, in order to enable data flow by means of policies sent by the SDN controller.

The implementation of SDN within a WSN network has its benefits, and so, from the evaluation performed it is concluded that the data management through the application allowed improving the response time and data delivery between the server and sensor node as well as contributing in a general way to the security of the WSN network.

CAPÍTULO I

ANTECEDENTES

Tema

Diseño e implementación de una red de sensores inalámbricos definidos por software para el monitoreo de un sistema hidropónico NFT

Problema

En las últimas décadas las tecnologías de computación y comunicación han tenido una enorme evolución, lo que ha llevado a un desarrollo y despliegue continúo de las infraestructuras de redes informáticas en términos de dimensión y complejidad. Una de las tecnologías que se ha convertido en parte integral de la vida cotidiana son las WSN, (El-Mougy, Ibnkahla, & Hegazy, 2015), que se han utilizado en diferentes escenarios como vigilancia, detección de desastres, monitoreo ambiental, atención médica y agricultura.

En la actualidad las WSN requiere de un enorme despliegue de nodos para cubrir grandes extensiones, ya sea para el monitoreo ambiental o de condiciones físicas, con la finalidad de censar información relevante de un fenómeno o acontecimiento. Sin embargo, esto ha causado una gran cantidad de tráfico, el cual demanda un alto grado de consumo de energía, generando que en algún momento los nodos puedan agotarse completamente, que provoca altas tasas de pérdida de datos y esto a su vez degrada el rendimiento de la red, por otro lado, la cantidad de nodos a gestionar dificulta la administración y configuración de la red, lo cual hace que sea compleja y poco escalable, (Alves, Oliveira, Pereira, Albertini, & Margi, 2018). Por ende, se requiere de una red de sensores inalámbricos que puedan soportar la administración en

grandes niveles de escalabilidad, tráfico de datos, movilidad y seguridad con el propósito de mantener la red operativa (Alves, 2018).

Una alternativa para solventar este desafío es aprovechar las características importantes de una red definida por software como desacoplar el plano de datos y de control, el cual permitirá gestionar el flujo de datos de una manera adecuada mediante la implementación de un controlador SDN.

En resumen, lo que se pretende desarrollar en este proyecto es una red de sensores definas por software, la cual será implementada dentro de un cultivo hidropónico NFT dedicado al sembrío de Fragaria Vesca con el objetivo de evaluar su funcionamiento.

Objetivos

Objetivo General

Integrar las redes de sensores inalámbricos y las redes definidas por software mediante la implementación de un controlador SDN enfocado al cultivo hidropónico NFT.

Objetivos Específicos

- Estudiar los conceptos fundamentales sobre las redes de sensores inalámbricos definidas por software y de los cultivos hidropónicos NFT.
- Diseñar una red de sensores inalámbricos que permita la interoperabilidad con la SDN para el monitoreo de cultivos Hidropónicos NFT.
- Implementar un sistema de monitoreo para cultivos Hidropónicos NFT dedicado al sembrío de la Fragaria Vesca.
- Realizar las pruebas de funcionamiento para determinar el desempeño de la red de sensores inalámbricos definidos por software

Alcance

El presente proyecto consiste en integrar las redes de sensores inalámbricos y la SDN, con el propósito de administrar la red mediante la implementación de un controlador SDN, para la demostración se desarrollará un sistema de monitoreo de un cultivo hidropónico NFT dedicado al cultivo de fragaria Vesca.

Para el desarrollo del proyecto se iniciará con el estudio de la parte teórica, el cual comprenderá los conceptos básicos las redes de sensores inalámbricos definidos por software, tales como arquitectura, protocolos, topología y controladores SDN, con la finalidad de determinar los requerimientos necesarios que permita su integración, además, se analizará los parámetros ambientales involucrados en cultivo hidropónico NFT enfocado al sembrío de fragaria Vesca.

En el diseño de la red, se implementará una topología que permita el despliegue de los sensores en el cultivo hidropónico NFT. Además, la comunicación inalámbrica entre los nodos sensores y la estación base se lo realizará mediante el estándar de red 802.15.4, así mismo cada nodo sensor será responsable de censar la temperatura, humedad relativa, y niveles de pH. Con respecto a la selección de los equipos tantos de nodos inalámbricos como del dispositivo que tendrá configurado el controlador SDN se utilizará el proceso de benchmarket.

Una parte importante de las WSN basadas en SDN es el software, que facilita la gestión del flujo de datos generados por los sensores, por lo cual se estudiará el tipo de sistema operativo y controlador SDN, que comunique el nodo central con el controlador SDN, para esto es necesario aplicar la ISO/IEC/IEEE 29148 para elegir el sistema operativo que cumpla los requerimientos del sistema.

En base al diseño, se llevará a cabo la implementación del sistema de monitoreo del cultivo hidropónico NFT el cual, constará de una base de datos para tener un registro de las variables ambientales a monitorear, dicha información será visualizada en una plataforma web.

Finalmente se ejecutarán las pruebas de funcionamiento que permitan validar el funcionamiento de la red de sensores definidas por software para comprobar la correcta operatividad de todos los elementos que van a estar involucrados en la red de sensores, especialmente el controlador SDN

Justificación

La tecnología juega un papel fundamental en la formación de los estudiantes universitarios, sobre todo si se toma en cuenta que hoy nos encontramos en la era de la sociedad del conocimiento y de las TIC (Tecnologías de la Información y la Comunicación). Mediante el Plan Nacional del Buen Vivir vigente entre el 2017-2021, se impulsan nuevas estrategias que permitan el despliegue y desarrollo de infraestructura en Telecomunicaciones con el propósito de promover el desarrollo en áreas como salud, educación y actividad agrícola, mismas que poseen un alto impacto social para el desarrollo económico y social del país. Además, permita eliminar las brechas tanto tecnológicas como sociales de los ciudadanos, promoviendo la generación de conocimiento para crear nuevos productos innovadores que puedan competir en los mercados internacionales (Plan de buen vivir, 2017).

El crecimiento exponencial en el uso de sensores inteligentes conectados a la red de internet en diferentes escenarios, han proporcionado nuevas oportunidades de negocios. Sin embargo, al mismo tiempo, ha traído enormes desafíos a los profesionales de telecomunicaciones y tecnología de información. principalmente la modernización de las redes de telecomunicaciones, (Triawan, Hindersah, Yolanda, & Hadiatna, 2017).

Hoy en día, la arquitectura actual se basaba en conceptos que nacieron hace décadas. Desde ahí, diversos protocolos y componentes fueron agregados de acuerdo a las necesidades que surgieron. Sin embargo, a pesar de atender a las necesidades actuales, se volvió muy compleja y de cierta manera poco flexible, en otras palabras, a pesar de que la arquitectura de redes actual es bastante resistente, aparentemente, no es lo suficientemente escalable para las futuras necesidades (Tecnology & Operadoras, n.d.).

Como respuesta a este desafío, diversos campos del sector de telecomunicaciones y tecnología de la información, como fabricantes e investigadores empezaron a trabajar en las propiedades de las redes definidas por software, lo cual permitiría proporcionar una red que puedan soportar la administración en grandes niveles de escalabilidad, tráfico de datos, movilidad y seguridad con el propósito de mantener la red operativa debido que permite desacoplar el plano de datos y control.

CAPÍTULO II

FUNDAMENTO TEÓRICO

Para iniciar el desarrollo del presente trabajo de titulación, se llevará a cabo la revisión bibliográfica con el propósito de conceptualizar los elementos y protocolos que intervienen en el diseño de una red de sensores inalámbricos y una red definida por software. Por último, se define lo que consiste en un cultivo hidropónico NFT (por sus siglas en ingles *Nutrient Film Technicque*) y las variables ambientales a censar.

2.1. Redes de sensores inalámbricos (WSN)

Se define a las redes de sensores inalámbricos WSN (por sus siglas en inglés *Wireless Sensor Networks*) como un conjunto de sensores interconectados y distribuidos en un área determinada para el monitoreo de acontecimientos naturales o físicos. Cada uno de estos nodos sensores se comunican entre sí, de manera inalámbrica o cableada, con el propósito de transmitir datos hacia el sistema de control para su procesamiento y actuación (Modieginyane et al. 2018).

La figura 1 muestra un ejemplo de una red WSN que está compuesta por nodos sensores, gateway y estación base, cada uno de estos dispositivos cumplen con las siguientes funciones, los nodos sensores son encargados de recoger datos, el gateway sincroniza cada nodo para establecer una topología mesh y, por último, llega a la estación base donde los datos de los sensores son almacenados y procesados.

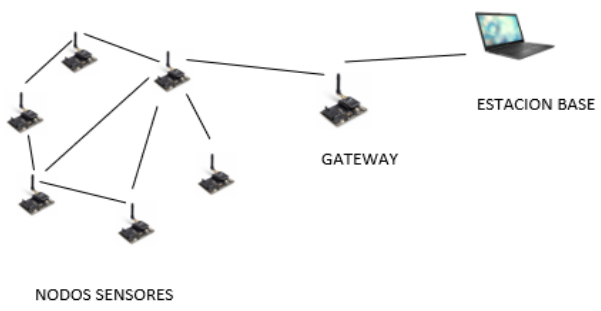


Figura 1. Elementos de una red WSN

Fuente: elaborado por el autor

Los Nodos Sensores o también denominados Motas, son dispositivos computacionales con una limitada capacidad de procesamiento y almacenamiento de datos (Ruiz Canales & Molina Martínez, 2016). La figura 2 representa el conjunto de subsistemas que forman parte de un nodo sensor, cada uno de estos subsistemas cumplen con diferentes funciones específicas tales como:

- Subsistemas de energía: es la principal fuente de energía para que los nodos sensores cumplan con sus funciones correctamente. Estos subsistemas utilizan principalmente células solares y baterías recargables.
- Subsistema de sensores: son dispositivos electrónicos que miden diversos parámetros ambientales como: presión barométrica, humedad del suelo, temperatura ambiente, etc.
- Subsistema computacional: está formado por microcontroladores que se encargan de ejecutar un conjunto de instrucciones para procesar y enviar información hacia los subsistemas de comunicación
- Subsistema de comunicación: son dispositivos semi-duplex que son encargados de enviar y receptar información por un medio inalámbrico que trabaja en la banda de frecuencia de 2.4 GHz.

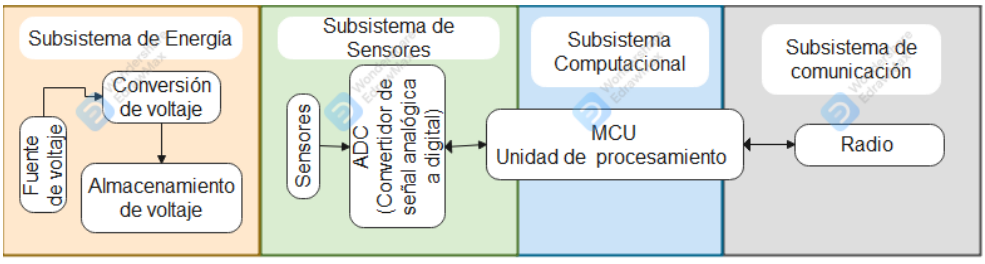


Figura 2. Estructura de una mota.

Fuente:(Ruiz Canales & Molina Martínez, 2016)

Con lo que respecta al gateway es un dispositivo que actúa como puerta de enlace, permitiendo interconectar las redes de sensores hacia una infraestructura de red TCP/IP existente. Por lo general el gateway, en la red WSN actúan como coordinador para organizar y

establecer protocolos de enrutamiento entre los nodos sensores, como es el caso de las redes 6lowpan (Lee et al., 2016).

Otra característica del nodo gateway es que funciona como servidor, es decir permite almacenar datos en archivos de texto plano basados en el formato CSV (por sus siglas en inglés, coma seperated value). Una vez que, los datos se alojen en el servidor es necesario visualizar la información de un modo entendible para el usuario, por ende es necesario aplicar una arquitectura orientada al servicio por medio de los middleware que son encargados de enviar datos hacia las diferentes plataformas web (Ortega-Corral et al., 2015).

2.2. Estándar 802.15.4

El estándar 802.15.4 tiene como objetivo proporcionar una comunicación inalámbrica de bajo costo y velocidades de transmisión de datos hasta 250 Kbps, compatibles con dispositivos de bajos recursos como son las WSN. El grupo de trabajo del IEEE 802.15.4 se centró en el estudio de las capas inferiores del modelo OSI, con el propósito de mejorar el funcionamiento de la capa física (por sus siglas en ingles PHY) y la capa de control de acceso al medio (por sus siglas en ingles MAC), además define el tipo de nodos y topologías que podrían formar la infraestructura para redes basadas en este estándar (Yang, 2013).

Por este motivo, el estándar 802.15.4 presenta diferentes versiones que se detallan en tabla 1, todas las recomendaciones sugeridas a partir del año 2011 han permitido interconectar la capa física y MAC, mientras que para año 2012 se agregaron nuevas funcionalidades a la capa MAC para apoyar a los mercados industriales.

Tabla 1.
Familia del estándar 802.15.4

Año	Versión del Estándar 802.15.4	Descripción General
2011	802.15.4 2003	Operaciones en las bandas de baja frecuencia de 868 y 915 MHz y 2.4 Ghz.
	802.15.4 2006	Mejorar la velocidad de datos asignando los siguientes esquemas de modulación: • Desplazamiento de fase en cuadratura offset. • Modulación por desplazamiento de amplitud (ASK). • Técnica del espectro ensanchado por secuencia directa (DSSS).
	802.15. 4ª	Operación en la banda ultra ancha basada en radio de impulso (IR-UWB).
2012-2013	802.15.4c	Asignación de la banda de operación 779 y 787 Mhz para que opere en China.
	802.15.4d	Asignación de la banda japonesa 950 - 956 MHz.
	802.15.4e	Mejoramiento del estándar ISA SP100. 11a.
	802.15.4f	Define las bandas 2.4Ghz y 433 Mhz para UWB
	802.15.4g	Permite desarrollar aplicaciones redes inteligentes destinadas a industrias energéticas basadas en la banda 902-9228 Mhz.

Fuente: (Yang, 2013)

2.2.1. Capa Física (PHY)

La capa física es encargada de transmitir y enviar información a través de un medio físico, en este caso mediante una conexión inalámbrica que utilizan módulos que trabajan en la banda de frecuencia 2.4 GHz (Sasián et al., 2016).

La información es enviada por una trama denominada unidad de datos de protocolos de capa física (por sus siglas por inglés, PPDU) que proporciona dos servicios: datos y gestión. El servicio de datos es encargado de encapsular toda la información para enviarla hacia las capas

superiores; mientras que el servicio de control provee a la capa física las siguientes funcionalidades:

- Encendido y apagado del radio de comunicación
- Evaluación del canal de comunicación mediante los procesos de detección de energías (ED) y asistente de canal limpio (CCA) para seleccionar el canal adecuado para transmitir la trama PHY (Ramírez, 2012).

En la figura 3 se representan los campos de la trama de la unidad de protocolos de capa física que incluye los campos: sincronismo (SHR), encabezado (PHR) y carga útil; el campo de sincronismo cumple con la función de sincronizar la comunicación entre los dispositivos a través de los transceptores además, permite determinar el inicio y final de una trama; mientras que, el encabezado PHY (por sus palabras en ingles Heard PHY) indica el tamaño de la trama; y por último se tiene el campo de carga útil que representa la unidad de datos de servicios que es encapsulado en la capa MAC con una longitud de trama que varía entre 0 a 127 bytes (Bhat, 2011).

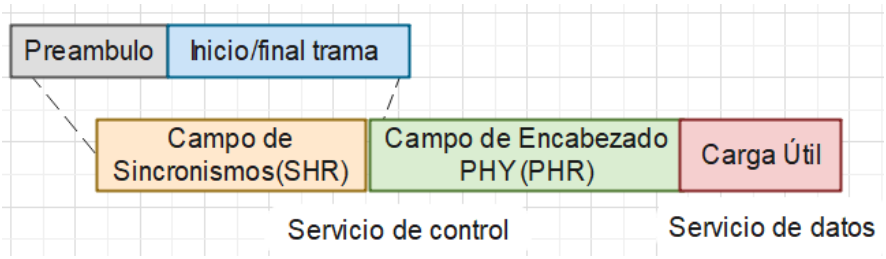


Figura 3. Trama de Unidad de datos de Protocolo de PHY

Fuente:(Caprile, 2009)

2.2.2. Capa de Control de Acceso al Medio (MAC)

Los nodos sensores en la red WSN al tener un reducido ancho de banda comparten el mismo medio de acceso para transmitir los datos, por lo que requieren del protocolo de capa de control de acceso al medio (por sus siglas en ingles MAC) para gestionar los canales físicos de una forma eficiente y evitar la colisión entre paquetes (Caprile, 2009).

En la figura 4 se detalla el formato de la trama MAC, que también se le denomina unidad de datos de protocolo MAC (MPDU) y está formado por los campos de encabezado, carga útil y secuencia de comprobación de trama (FCS).

El campo encabezado de la trama MAC se subdivide en los campos trama de control y direccionamiento; mientras que la unidad de datos de servicios MAC (por sus siglas en ingles MSDU) o también denominado carga útil contiene la información a transmitir hacia las capas superiores; finalmente se tiene el campo de secuencia de comprobación de la trama que incluye la información de cheque de errores (Caprile, 2009).

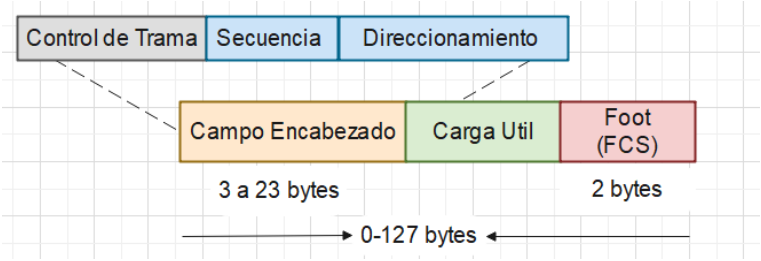


Figura 4. Formato de la capa MAC

Fuente: (Caprile, 2009)

Ahora bien, se puede identificar que la trama de encabezado define dos campos importantes como trama de control y campo de direccionamiento. Con lo que respecta a la trama de control consiste en identificar los diferentes mensajes que permiten la comunicación entre nodos; tales como: datos, comandos, reconocimiento o acuse de recibido (ACK) y beacon, cada uno de estos mensajes cumplen con una función específica:

- **Dato**: contiene la información proveniente de capas superiores.
- **Acuse recibido** (ACK): son mensajes trasmitidos por la trama FCS que permiten la verificación de la recepción y validación de datos en los dispositivos.
- **Comandos**: permiten realizar funciones administrativas en la red y verifican la disponibilidad de los nodos sensores. Un ejemplo de este tipo de mensajes es el *association request* que permiten asociar los dispositivos hacia la red WSN.

- **Beacon:** son mensajes emitidos por el coordinador de la red para descubrir y asociar nuevos dispositivos a la red.

Con relación al campo direccionamiento define dos tipos de direccionamiento: corto y extendido con el objetivo de reducir el espacio en la trama MAC, en la tabla 2 describe las características de los tipos de direccionamientos como también los identificadores de red que son ID PAN y Broadcast (Caprile, 2009).

Tabla 2.
Tipos de Direccionamiento en la Trama MAC

Tipo de Direccionamiento	**Características**
Extendido	Es una dirección única de 64 bits que permite identificar los dispositivos.
Corta	Es una dirección única de 16 bits que permite identificar los dispositivos dentro de una red. Por lo general esta dirección es asignada por el coordinador de la red.
Identificador de red (PAN ID)	Permite identificar el tipo de tráfico dentro de una misma red.
Broadcasts	Es un direccionamiento corto y es utilizado cuando la información es útil para muchos dispositivos finales. El formato de dirección de destino es 0xFFFF.

Fuente: (Caprile, 2009)

2.2.3. Topología de red

La topología de red se refiere a la forma en que se conectan los nodos sensores para trasmitir la información. El estándar 802.15.4 define tres tipos de topologías: estrella, *mesh* y árbol, cada una de estas topologías se hallan conformadas por dispositivos de función completa (por sus siglas en ingles FFD) denominados coordinador PAN o routers que cumplen con la función de sincronizar los elementos de la red mediante los mensajes beacon, mientras que los dispositivos de función reducida (por sus siglas en ingles RFD), son los sensores y actuadores de la red (Ghosh, 2017)

La figura 5 muestra la topología en estrella formada por un solo coordinador, cuya función es gestionar y controlar el flujo de datos de cada nodo. Las ventajas que ofrece la topología en estrella, es que al tener una falla en un único nodo no afecta al resto de la red. Sin embargo, la distancia entre los nodos y el coordinador debe ser menor de 10 metros.

Mientras que, la topología *mesh* está formada por un coordinador y varios FFD router, que permite extender la cobertura de la red WSN, es decir cada nodo sensor se convierte en un nodo enrutador con la finalidad de crear rutas alternas hacia el coordinador, de esta forma se tiene una red tolerante a fallas. No obstante, el consumo de energía en este tipo de topologías aumenta debido a la cantidad de procesos de enrutamiento que realizan los dispositivos de red.

Finalmente, se tiene la topología en árbol que es similar a la topología en *mesh* con la diferencia que los dispositivos RFD y FFD *routers* son agrupados dentro del clúster, como se detalla en figura 4, cada clúster es identificado mediante un *head clúster*, el cual realiza el proceso de control en la transmisión de datos y apagado del radio físico del nodo, lo que hace eficiente el consumo de energía

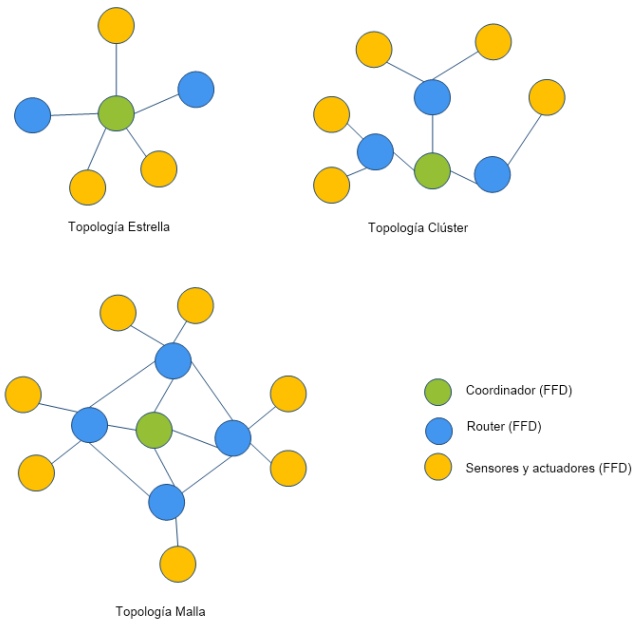


Figura 5, Tipos de topologías del estándar 802.15.4

Fuente: (Ghosh, 2017)

2.3. 6LOWPAN

El estándar 6LoWPAN (por sus siglas en inglés *IPv6 over Low Power Wireless Personal Area Networks*) está definida por el IETF (por sus iniciales en inglés Grupo de Trabajo de ingeniería de Internet) en la RFC (*Request For Comment*) 4919, donde se especifica el proceso de optimización para transmitir el datagrama del protocolo de internet versión 6 (IPv6) a través de enlaces que soporten el estándar 802.15.4 (Kumar et al., 2014).

Este proceso de optimización permite relacionar el protocolo IPv6 dentro de una trama 802.15.4 mediante los mecanismos de fragmentación, reensamblaje y compresión de cabecera, los mismos que se encuentran agrupados en la capa de adaptación. Cabe mencionar que estos mecanismos se hallan especificados con mayor detalle en el RFC 4944 (Shelby & Bormann, 2011)

En la figura 6, expone la arquitectura 6Lowpan con sus respectivos protocolos correspondientes a cada capa. La capa inferior comprenden el estándar 802.15.4 que se mencionó en la sección 2.2, mientras que el resto de capas como red, transporte y aplicación se detallará en las siguientes secciones incluyendo el protocolo IPv6 y los protocolos de enrutamiento, denominados router-over y mesh-under (Al-Kashoash, 2019b).

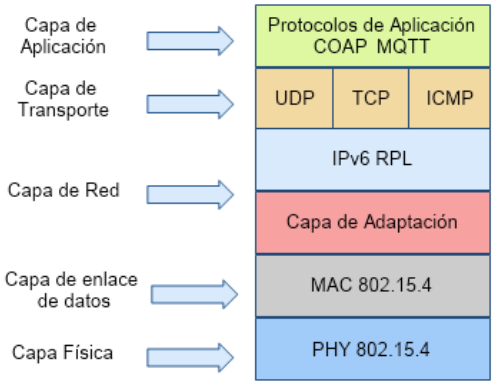


Figura 6, Arquitectura 6lowpan

Fuente: (Ruzzelli, 2011)

6lowpan utiliza el proceso de comprensión de cabecera con la finalidad de reducir la sobrecarga del encabezamiento de IPv6 y UDP debido a que sobrepasa el tamaño máximo de transferencia establecida en el estándar 802.15.4 que es de 127 bytes.

Según la RFC 4944 existen dos tipos de comprensión: encabezado IPv6 denominado HC1 y encabezado UDP denominado HC2. La comprensión HC1 consiste en eliminar la información redundante, como es el caso del ID de la interfaz de IPv6 y la dirección MAC, que se genera cuando los hosts envían mensajes de solicitud a los hosts vecinos, además permite comprimir de forma independiente las direcciones locales de destino y origen, los mismos, que son agrupados en los campos SAE (codificación de dirección de origen) y DAE (destino de codificación de dirección), que ocupan 2 bits de la cabecera HC1, Además la comprensión HC1 incluye un campo denominado dispatch que permiten identificar el tipo de comprensión, mientras que el campo c permite omitir las etiquetas de flujo y clase de tráfico cuando su valor es cero (Daneels et al., 2021).

Por otro lado, la comprensión de la cabecera UDP denominado también HC2, consiste en comprimir los puertos UDP de destino y origen, el resto de los campos UDP no comprimidos son enviados al campo NC (campos no comprimidos) de HC1. La figura 7 indica los campos que forman parte la comprensión de cabeceras IPv6 y UDP.

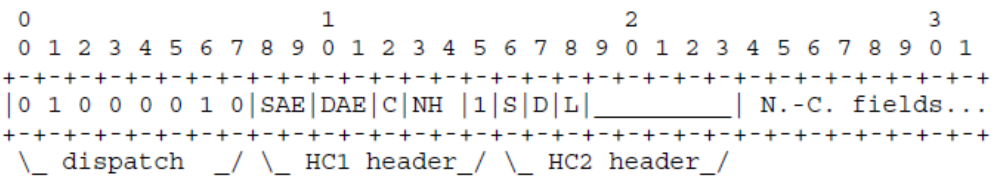


Figura 7. Paquete IPv6 comprimido HC1: sin y con HC2

Fuente: (Graziani, 2017)

2.3.1. Fragmentación y Reensamblado

La fragmentación es un mecanismo importante que sucede en la capa adaptación y divide la trama IPv6 en múltiples fragmentos, cada uno de estos fragmentos tienen un tamaño

de 8 bytes. Este proceso es debido a que, la trama IPv6 tiene un tamaño de 1280 bytes y excede la trama IEEE 802.15.4, que solo es de 127 bytes (Vasseur, 2012), (Shelby & Bormann, 2011).

Mientras que, el proceso de reensamblado tiene el propósito de reconstruir exactamente el paquete original en el extremo receptor, por el cual 6lowpan asigna un buffer equivalente del tamaño del encabezado IPv6 incluyendo la carga útil, de tal manera que permita almacenar los diferentes datagramas de fragmentación,

En la figura 8 se muestra los campos que forman parte del encabezado de fragmentación que consta de tres datagramas denominados: *size, tag y offset*; el datagrama *size* tiene un tamaño de 11 bits y soportar hasta 2047 bytes, suficiente para alojar los 1280 bytes requeridos por la trama Ipv6; mientras que el datagrama *tag*, aloja la dirección de la capa de enlace del destino que junto con datagrama size permiten distinguir los diferentes paquetes que se van a reensamblar; por último se tiene el datagrama *offset* con una capacidad de 8 bits que indica la posición del fragmento del paquete reensamblado

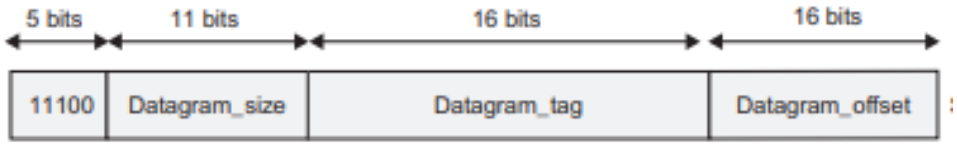


Figura 8. Formato de la cabecera de fragmentación
Fuente:(Vasseur, 2012)

2.3.2. Protocolo de Internet versión 6 (Ipv6)

Ipv6 denominado también IP de nueva generación es un protocolo de capa de red que fue desarrollada por IETF no solo para resolver el agotamiento de direcciones Ipv4, sino también para añadir nuevas funcionalidades que permitan al protocolo Ipv6 tener mayor escalabilidad frente al protocolo Ipv4, entre las ventajas más importantes se tiene (Grgic et al., 2016):

- Mayor número de direccionamiento aproximadamente Ipv6 otorga 4.3 billones de direcciones.
- Ofrece la facilidad de implementar calidad de servicio (QoS) debido a que agrega en su cabecera los campos: clase de tráfico y etiqueta de flujo, los cuales permiten dar prioridad a los paquetes dependiendo del tipo de servicio.
- Incluye mecanismos de seguridad basados en Ipsec.

De acuerdo con la RFC 2460, el encabezamiento IPv6 tiene un tamaño fijo de 40 bytes distribuidos en 8 campos. En la tabla 3 se describe la longitud y el funcionamiento de cada uno de los campos que forman para de la cabecera IPv6 (Jareño, 2016).

Tabla 3.
Encabezado IPv6

Campo de la cabecera IPv6	**Longitud (bit)**	**Función**
Versión	4	Indica el tipo de versión IPv4 o IPv6.
Clase de tráfico	8	Permite identificar y distinguir el tipo de clases o prioridades de paquetes IPv6.
Etiqueta de flujo	20	Permite identificar los paquetes que requieren tener un manejo especial.
Tamaño de datos	16	Indica la cantidad de carga útil del encabezado IPv6. Además, dentro de este campo se incluye el encabezado de extensión y la unidad de PDU de las capas superiores
Próxima cabecera	8	Permite identificar la siguiente trama IPv6, por ejemplo: • O1= ICMPv4 • 06= TCP • 17 = UDP • 58 = ICMPv6
Límite de saltos	8	Indica el número de saltos máximo que puede dar un paquete antes de ser eliminado
Dirección de origen	128	Indica la dirección IPv6 de origen.

Dirección de destino	128	Indica la dirección IPv6 de origen.

Fuente: recopilado de (Minoli, 2013)

Por otra parte, IPv6 incluye el protocolo de descubrimientos de vecinos (por sus siglas en ingles NDP) y el protocolo de control de mensajes de internet versión 6 (por sus siglas en ingles ICMPv6), ambos protocolos realizan las funciones de descubrimientos de dispositivos y determinan errores que suceden dentro de una red local.

Con referencia al protocolo de descubrimiento de vecinos (por sus siglas en ingles ND), está definido en el RFC 2461 y permite establecer un diálogo con los dispositivos que están dentro de una misma red a través de los mensajes de ICMPv6, los cuales son encargados de resolver las direcciones IPv6 en direcciones MAC válidas, cuya información son almacenados en *neighbor* cache, de esta forma permiten verificar la disponibilidad de un dispositivo dentro de una red local (Graziani, 2017). Otras de las características de este protocolo son:

- Permite una configuración automática de las direcciones IPv6.
- Permite verificar la duplicidad de las direcciones IPv6 locales.
- Provee de información de los dispositivos vecinos si se encuentran disponibles

En cuanto al protocolo de control de mensajes de internet versión 6 (ICMPv6) es encargado de detectar y reportar errores de paquetes que no fueron procesados correctamente dentro de la red. Este proceso de verificación es posible con la utilización de las herramientas del ping y tracerouter que permiten dar un diagnóstico del estado de la red (Loshin, 2004).

En la figura 9, se indica el formato del paquete ICMPv6 que consta de tres campos: tipo, código y checksum; el campo tipo permite identificar si es un mensaje de error o informativo; mientras que el campo código informa el tipo de mensaje ICMPv6 y por último el checksum detecta errores en los mensajes ICMPv6 (Graziani, 2017).

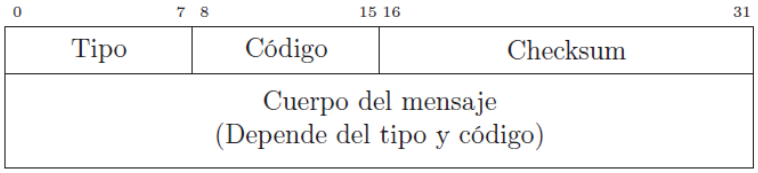


Figura 9. Formato del paquete ICMPv6

Fuente: (Al-Kashoash, 2019b)

2.3.3. Protocolos de enrutamiento de 6LOWPAN

Los protocolos de enrutamiento de una red 6LOWPAN juegan un papel importante porque permiten establecer rutas adecuadas para enviar información entre un nodo origen y destino, además existen dos tipos de enrutamiento: *mesh under y router-over* que se ejecutan en distintas capas de la arquitectura 6LOWPAN, es decir el enrutamiento mesh aunder enruta paquetes a nivel de enlace, en cambio, el router-over a nivel de red (Noordin, & Ali, 2010).

Mesh-under consiste enviar paquetes al destino en múltiples saltos utilizando la trama 802.15.4 o en un encabezado 6lowpan. Con el objetivo de lograr exitosamente este proceso de enrutamiento la IETF define un encabezado denominado *Mesh-Header* que ayuda a determinar el límite de saltos que tiene el paquete antes de ser descartado por el *router*. Mientras que el *router-over* convierte a cada nodo en un enrutador IP, para utilizar sus tablas de enrutamiento IP y la opción salto a salto del encabezado IPv6 para enrutar paquetes con los diferentes nodos vecinos (Ćulibrk., 2013).

Ahora bien, existen otros protocolos que fueron desarrollados con la finalidad de mejorar el proceso de enrutamiento en las redes de baja potencia como: enrutamiento jerárquico (Hilow), LOAD y RPL. Sin embargo, Bohloulzadeh & Rajaei, señala que RPL tiene una mayor estabilidad en la construcción de rutas debido a que separa el procesamiento y el reenvió de paquetes, obteniendo como resultados el mínimo consumo de energía y latencia en la red (Bohloulzadeh & Rajaei, 2020).

Finalmente, en la figura 10, representa el procedimiento para ejecutar un enrutamiento RPL que según la RFC 6550, consiste en construir una topología con base en grafos denominados *Destination Oriented Directed Acyclic Graph* (por sus siglas en ingles DOGDA) que a su vez se expande y se divide en subgrafos denominados Directed Acyclic Graph (por sus siglas en ingles DAG), los cuales se comunican con los nodos raíces o root por medio de los mensajes DIO (Objeto de información DODAG) que son emitidos periódicamente por el temporalizador trickle. Además, el funcionamiento del DOGDA depende de la función objetiva que realiza las funciones de determinar la mejor ruta basadas en métricas que garantizan el consumo de energía y memoria en los nodos (Shelby & Bormann, 2011).

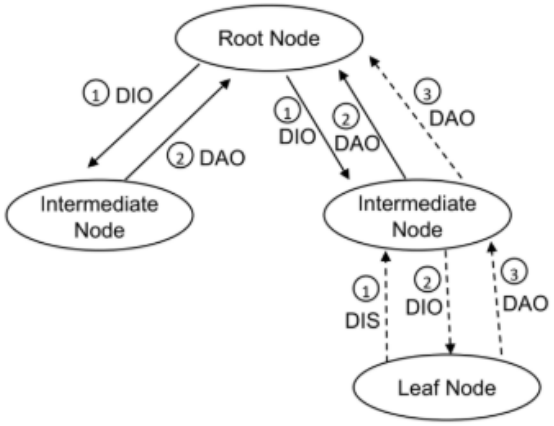


Figura 10. Proceso para construir un DOGDA

Fuente: (Jung et al., 2013)

2.3.4. Protocolos de capa Aplicación

Los protocolos de capa aplicación para las redes WSN son encargados de transferir la información recolectada por los nodos hacia aplicaciones o servicios que se encuentran disponibles en el internet. Sin embargo, estos protocolos deben adaptarse a las limitaciones de los dispositivos finales de las WSN para reducir la pérdida de información. Entre los protocolos que en la actualidad han obtenido gran aceptación para las redes WSN son COAP y MQTT (Naik, 2017).

El protocolo de Aplicación Restringida (CoAP) es un protocolo de transferencia web desarrollado por la IETF para dispositivos y redes de bajos recursos, que utiliza una arquitectura REST para establecer una comunicación bidireccional entre nodo y servidor. Este tipo de comunicación emplea el protocolo UDP que garantiza que la transferencia de información llegue sin sobre carga.

Por otro lado, CoAP se divide en dos subcapas principales; mensajería y solicitud/respuesta. La subcapa de mensajería cumple con dos funciones verificar la fiabilidad y duplicidad del mensaje; a diferencia de la subcapa solicitud/respuesta que es encargada de establecer una comunicación con los nodos finales a través de los métodos GET, PUT, POST y DELETE.

Por último, se describe el protocolo de transporte de telemetría de la cola de mensajes (MQTT) que es un protocolo de mensajería basado en publicaciones y suscripciones que fue desarrollado para garantizar la entrega de mensajes en dispositivos que tienen un ancho de banda reducido y una capacidad de procesamiento bajo.

Por lo general, MQTT está formado por tres componentes denominados editor, suscriptor y broken que utilizan el protocolo TCP para transmitir la información; el editor es encargado de trasmitir la información sobre un determinado tema o tópico hacia el broken que, a su vez, lo retransmite a los suscriptores que estén registrados con el mismo tópico.

2.4. Computación en la Nube

La computación en la nube también llamada "*cloud*" es una arquitectura que proporciona a las empresas una variedad de recursos en hardware y software con la finalidad de obtener los servicios de almacenamiento de archivos y procesamiento de datos en el internet (Sinaeepourfard et al., 2017).

Ahora bien, el *cloud* proporciona diferentes modelos de servicios que permiten a las redes WSN tener una variedad de funcionalidades como: conectividad de dispositivos IoT, administración de datos y gestión de seguridad, desarrollo de aplicaciones etc. Estos modelos de servicios se dividen en tres categorías: Software como Servicio (SaaS), Plataforma como Servicio (PaaS) e Infraestructura como Servicio (IaaS).

- Software como Servicio: es una infraestructura que ofrece a los usuarios aplicaciones ya desarrollas como por ejemplo los servicios de correo, páginas web, almacenamiento de datos, etc.

- Plataforma como servicios: proporciona al usuario los recursos necesarios para que desarrolle y administre aplicaciones que se encuentran disponibles en la red, por lo general los recursos que el usuario puede acceder son: sistemas operativos, lenguaje de programación y base de datos.

- Infraestructura como Servicio: este modelo permite a los usuarios tener un control completo de los recursos de hardware y software incluyendo virtualización, servidores, redes, backup y firewalls. De esta forma, el usuario puede desarrollar una infraestructura de red completa con una variedad de aplicaciones y servicio de una forma rápida y económica.

En la tabla 4, se mencionan las características de las principales plataformas disponibles en el *cloud*, por lo general estas plataformas proveen al usuario una versión gratuita de un año, pero con limitaciones en almacenamiento y memoria RAM.

Tabla 4.
Comparación de plataformas IoT

Características	AWS	IoT Google	Azure Microsoft
Soporte sistemas operativos (OS)	Windows Linux	Windows Linux	Windows Linux
Base de Datos	MySQL	MongoDB	MySQL

Soporte de protocolos	MQTT HTTP	MySQL PostgreSQL MQTT HTTP	MQTT HTTP
Lenguaje de Soporte	PHP	PHP Java Python NodeJS	Java Python NodeJS
Soporte IoT	Gestión y administración de dispositivos IoT	Gestión y administración de dispositivos IoT	Gestión y administración de dispositivos IoT
Paquetes de inicio gratis el primer mes	UPC: 1 núcleo Memoria: 512 SSD: 15	UPC: 1 núcleo Memoria: 512 SSD: 20	UPC: 1 núcleo Memoria: 512 SSD: 20

Fuente: (O. Q. Muñoz, 2019)

2.5. Redes Definidas por Software (SDN)

Según la *Open Nerworking Foundation* (ONF), la SDN es una arquitectura emergente, en la que control de la red se desacopla de las funciones de reenvió por medio de la lógica de programación. Es decir la SDN permite que los dispositivos de red (routers y conmutadores) se convierten en elementos de reenvíos de datos mientras que el comportamiento e información de la red se halla centralizada en un software de control denominado controlador SDN (Benzekki et al. 2016). Sin embargo, Duan y toy, define a la SDN como una plataforma de control que permite administrar y gestionar los recursos de la red por medio del software, que normalmente se denominan aplicaciones SDN. Este software define el comportamiento de la red que consiste en construir la topología de la red y especificarlas operaciones de reenvío y procesamiento de datos (Duan & Toy, 2017)

Según la ITU-T (Unión Internacional de Telecomunicaciones) la arquitectura SDN tienen como propósito facilitar la comprensión del funcionamiento de los componentes e interfaces de conexión, que forman parte de la red SDN, por lo cual lo divide en tres capas denominadas: datos, control y aplicación, mismos que se detallan en la figura 11 (Duan & Toy, 2017).

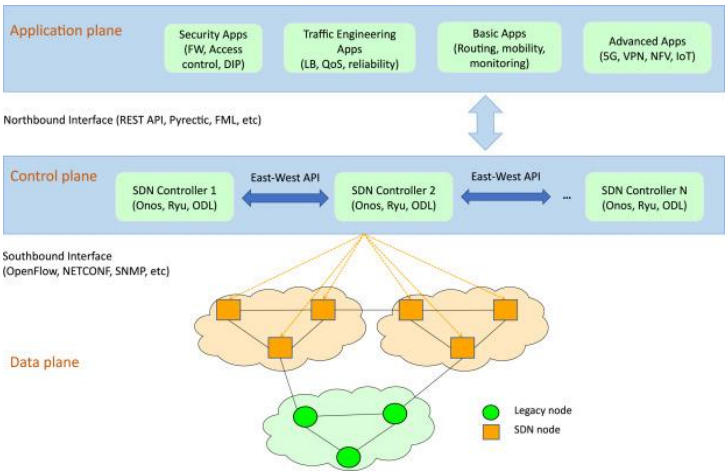


Figura 11. Arquitectura básica de las redes SDN

Fuente: (Khorsandroo et al., 2021)

La capa de aplicación: es un conjunto de aplicaciones y servicios que permiten determinar las funcionalidades y el comportamiento de la red, es decir estas aplicaciones emiten políticas a la capa de control, que luego traducen las políticas en reglas que soporten los dispositivos de la capa de datos.

La capa de aplicación establece comunicación con la capa de control a través de la interfaz hacia el norte (Northbound) y cumple con la función de desarrollar aplicaciones para una infraestructura de red como ingeniería de tráfico, calidad de servicio y seguridad,

La capa de control: es el cerebro de la red SDN y es responsable de gestionar y administrar los dispositivos que se encuentran en la capa de datos de acuerdo con las instrucciones que se establecen en la capa de aplicación, de tal forma permita tener una arquitectura inteligente y configurable a través de las interfaces de programación norte y sur.

Por otro lado, el controlador SDN está estructurado por módulos que mantienen una base de datos local para contener la topología y estadísticas de la red. En figura 12 se muestra cada uno de los módulos de del controlador mientras que en la tabla 5 se resumen el funcionamiento de cada uno de los módulos.

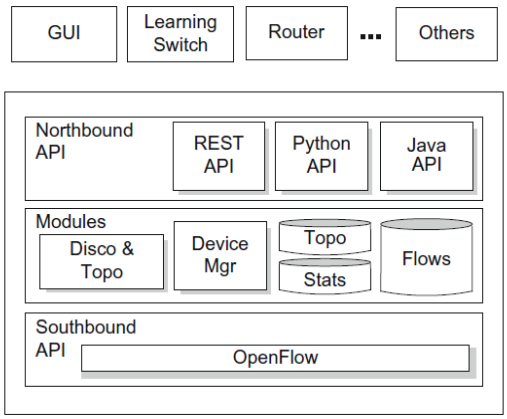


Figura 12. Arquitectura del controlador SDN.

Fuente (Göransson, 2014)

Tabla 5.

Descripción de las funciones de los módulos del controlador SDN

Función	Descripción	Interfaces
Desabrimientos de topología	Permite construir y obtener información de las conexiones de la red	Interfaz del sur
Gestión de flujos	permite establecer una base de datos sobre los flujos de datos y el sincronismo del controlador SDN	
Mecanismo de Seguridad	Proporciona seguridad a los dispositivos y aplicaciones que forman parte de la red SDN	Interfaz del norte

Fuente: (Duan & Toy, 2017)

Capa de datos: está compuesta por dispositivos virtuales y físicos como el switch SDN, que cumplen con la función de procesar paquetes, dependiendo de a las instrucciones que ejecute el controlador SDN. Además, permite establecer un canal de comunicación en el plano de control y el plano de datos por medio del protocolo openflow (Rothenberg et al., 2015)

2.5.1. Openflow (OF)

En un principio openflow fue desarrollada por la universidad de Stanford con el propósito de experimentar nuevos protocolos que permitan sustituir las funcionalidades de la capa 2 y 3 con el propósito mejorar la gestión de redes cotidianas (Morreale & Anderson, 2015).

En la actualidad, Openflow es un protocolo de código abierto que cumple con dos funciones: prioridad y ruteo; la primera define el flujo de paquetes que va a circular y la prioridad que tendrá dentro de la red por medio de las tablas de flujos que se encuentran alojadas en el conmutador; mientras que la función de ruteo determina la mejor ruta que tendrá el paquete, esto es posible debido al software de control que añade, actualiza y elimina instrucciones en las tablas de flujo, de esta manera se tiene una red flexible y optimizada (El-Mougy et al., 2015).

2.5.2. Switch openflow

Son dispositivos físicos o virtuales que están formados por tablas, canal seguro y controlador lógico SDN; las tablas están formadas por dos tipos de tablas: las tablas de flujo y tablas de grupo; las tablas de flujo son encargadas de procesar los paquetes entrantes; mientras que las tablas de grupo es un conjunto de identificadores e instrucciones que se aplican aun grupo específico de paquetes entrantes, por último, se tiene un canal seguro que permite una comunicación TCP/IP entre el controlador lógico SDN y switch openflow. En la figura 13 se muestra los componentes que forman parte de un conmutador openflow de acuerdo a la ONF (Fernández et al., 2018)

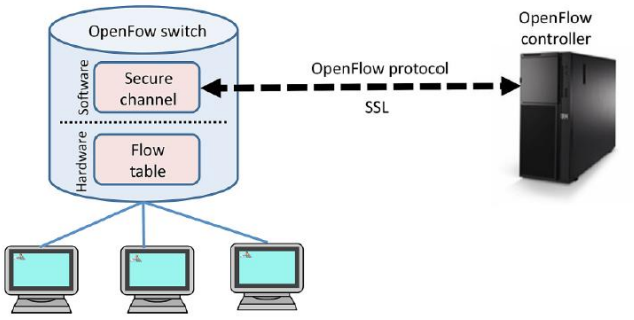


Figura 13. Componentes de un conmutador openflow 1.3 de acuerdo la ONF

Fuente:(Morreale & Anderson, 2015)

2.5.3. Tabla de Flujo

La tabla de flujo es estructura de datos que permiten a los switches evaluar ciertos campos de los paquetes entrantes, para luego tomar acciones apropiadas si los campos coinciden. Estas acciones pueden incluir reenvío de paquetes a un puerto específico o inundar paquetes en todos los puertos.

En la figura 14, se indica la estructura de una tabla de flujos que está formado por seis campos: coincidente, prioridad, contadores, instrucciones, tiempo de espera y cooke; el campo de coincidente está constituido por doce parámetros que se detallan en la figura 14, cada uno de estos parámetros son comparados con los paquetes entrantes de acuerdo a un orden de prioridad

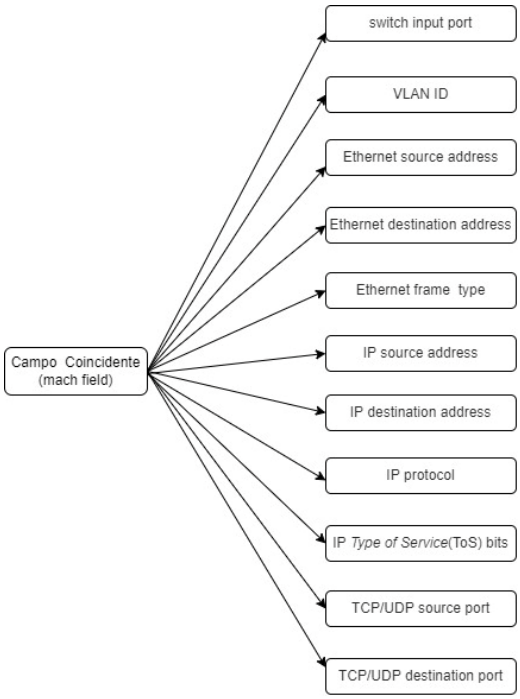


Figura 14. Parámetros del campo coincidente

Fuente:

El campo prioridad es encargado de especificar la preferencia que tiene un flujo de datos dentro del conmutador openflow; por otro lado, el contador es único para cada flujo entrante y recopila información sobre el estado del flujo de datos, esta información se basa en: almacenar paquetes aceptados y rechazados por el conmutador; el campo instrucciones es un conjunto de acciones que se ejecutan cuando existen paquetes coincidentes en el conmutador, sin embargo si no existe acciones o si la lista de instrucciones no son procesadas de acuerdo al orden específico el conmutador automáticamente rechaza el paquete; el campo de tiempo de espera es el tiempo que permanece un flujo de datos en el conmutado antes de ser descartado; por último se tiene el campo cookie que es utilizado por el controlador para enviar estadísticas, modificaciones o eliminar flujo entrante, pero no es utilizado en el procesamiento de los paquetes (Luo et al., 2012).

Campo Coincidente(Match)	Priorioridad	Contador	Instrucciones	Tiempo de espera	Cooke

Figura 15. Tabla de openflow versión 1.1

Fuente: recopilado de (Abdullah Al Atawi, 2018)

2.5.4. Canal seguro

Es una ruta donde circula los diferentes tipos mensajes de comunicación entre el controlador y el switch openflow a través de una conexión asimétrica basada en TLS *(Transport Layer Security)* aunque también soporta conexión TCP (protocolo de control de transmisión) sin cifrar (Hu, 2014). Los tipos de mensajes que soportan son:

- **Mensajes del controlador a switch**: estos mensajes son iniciados por el controlador para administrar y solicitar información del estado del *switch* openflow, alguno de las funciones que realiza son: verificar el estado de los puertos, insertar, actualizar y eliminar instrucciones dentro de las tablas de flujo (Rothenberg et al., 2015).
- **Mensajes asincrónicos**: son utilizados por el *switch* openflow para actualizar al controlador sobre el estado de la red como también los cambios que suceden en el *switch* como, por ejemplo: la llegada y eliminación de los flujos de datos entrantes en el conmutador.
- **Mensajes sincrónicos**: son mensajes que pueden iniciar el controlador o switch openflow y generalmente son utilizados para verificar la disponibilidad de un dispositivo mediante los mensajes *hello,* como también permite medir la latencia del canal mediante los mensajes *echo request* y *echo reply*.

Una vez comprendido el funcionamiento y los elementos que componen el protocolo openflow se puede entender el comportamiento del software open vswitch que es un dispositivo virtual multicapas basados en código libre escrito en lenguaje C, que está especialmente diseñado para gestionar y administrar una gran cantidad de flujos de datos con ayuda del controlador lógico SDN. Por lo general este componente es utilizado en entorno de virtualización como: Xen, KVM, Docker y VirtualBox. También se ha integrado en sistemas de gestión virtual, incluidos OpenStack, openQRM y oVirt. (Kobo et al., 2017).

Un conmutador *open vswith* incluye tres componentes: *ovs-vswitchd*, *ovsdb-server* y *data path*, los mismos que se hallan distribuidos en diferentes segmentos del sistema operativos por ejemplo el *ovs-vswitchd* y *ovsdb-server* se aloja en el espacio del usuario mientras que el *datapath* se aloja en el espacio del *kernel*. En la figura 16 se detalla los componentes del *open vswitch.*

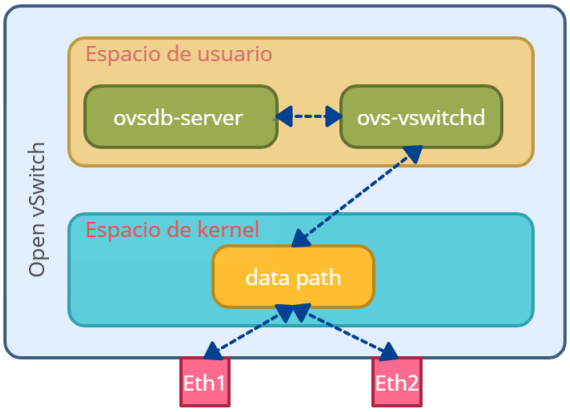


Figura 16. Elementos de un conmutador open vswitch

Fuente: Elaborado por el autor en referencia a (Jararweh et al., 2015)

Básicamente, el funcionamiento del *open vswitch* se basa en el *ovs-vswitchd* es un demonio que controla e implementa el protocolo openflow que con ayuda del módulo *ovsdb-server* almacena la configuración inicial del conmutador, mientras que el *datapath* es encargado del reenvío de paquetes (Gonzalez et al., 2018).

Un paquete dentro del conmutador *open vswitch* puede elegir una ruta rápida o lenta; la ruta rápida consiste en procesar primero el paquete localmente en el módulo kernel del sistema operativo sin la intervención de las reglas de openflow, a base de un conjunto de reglas que están formadas por tablas de flujo que asocian un flujo de datos con una acción. Este proceso es mucho más rápido de ejecutar con respecto a las reglas de openflow, en la figura 16, se visualiza la línea tomate como representación de la ruta más rápida (Emmerich et al., 2018).

Si las reglas o tablas de flujo no proporcionan la información necesaria de un paquete, este es enviado a la biblioteca denominada dpif que es una tabla simple que admite solo flujo de coincidencias exactas. Si no hay una coincidencia el paquete pasa a la librería ofproto donde se aloja la tabla de flujo de openflow completa, si existe una coincidencia se realiza las acciones asociadas, de lo contrario el ofproto pasa el paquete hacia el controlador lógico SDN, este proceso es la ruta lenta que es representado por la linea azul punteada de la figura 17 (Emmerich et al., 2018).

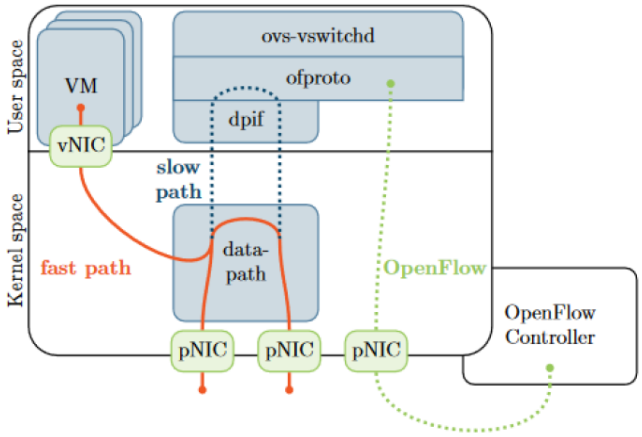


Figura 17. Procesamiento de paquetes dentro del open vswitch

Fuente: (Gonzalez et al., 2018)

2.5.5. Software de Control SDN

El controlador SDN representa el núcleo de las redes definidas por software, que proporciona el servicio de administración y gestión de flujo de datos de acuerdo con las políticas que un administrador de red implementa (Nadeau & Gray, 2013).

El controlador mantiene una vista general de la red y tiene el control de todos los dispositivos que conforman la estructura de la SDN mediante los APIs. El API en dirección *northbound* conectan un conjunto de aplicaciones, mientras el API de dirección al *southbound* con ayuda del protocolo openflow administrar y gestionar los dispositivos como también elegir la mejor ruta para el tráfico de los dispositivos (Göransson, 2014).

En la actualidad existen una variedad de controladores que se pueden implementar en una red, tomando en cuenta ciertos parámetros como, por ejemplo: soporte de openflow, funcionalidad de la red, estabilidad y rendimiento, etc. En la tabla 6, se detalla controladores SDN basado en código abierto.

Tabla 6.
Lista de software de control SDN basados en código abierto

Nombre de Software SDN	Lenguaje de desarrollo	Licencia
Beacon	Java	Apache
Floodlight	Java	Eclipse
Opendaylight	Java	Eclipse
Ryu	Python	Apache
Onos	Python	Apache

Fuente: (Centeno ,2017)

2.6. Los cultivos Hidropónicos

Los cultivos hidropónicos forman parte del sistema agrícola inteligente que requieren de una alta atención por parte del agricultor, por ende se requiere del monitoreo constante de los parámetros ambiente donde se desarrolla el cultivo, de esta forma se obtiene un mejor rendimiento en la producción y optimización de recursos (Umamaheswari et al., 2017).

La palabra hidroponía deriva del griego Hidro (agua) y Ponos (labor), lo que significa trabajos en agua. Los cultivos hidropónicos es una técnica de cultivo que permite el desarrollo de plantas o vegetales sin suelo, pero utiliza soluciones nutritivas a través del agua. Sin embargo, si se desea tener éxito en este tipo de cultivos se requiere de un control riguroso de las soluciones nutritivas, con la finalidad de proveer los nutrientes necesarios para que la planta se desarrolle normalmente (Beltrano & Gimenez, 2015).

Con la hidroponía se puede cultivar una variedad de verduras, frutas y plantas aromáticas entre las cuales se tiene: lechuga, espinaca, perejil, ajo, arándanos, fresas, frambuesas, zarzamoras, orégano y menta, que según la organización de *Hydroponic Food Production (*Productos Alimenticios Hidropónicos) se obtuvieron cultivos con mejor sanidad y calidad. Las ventajas que se obtiene con este tipo de cultivos son los siguientes: (Beltrano & Gimenez, 2015).

- Cultivos libres de parásitos, bacterias, hongos y contaminación.
- Reducción de costos de producción.
- Independencia de los fenómenos meteorológicos.
- Permite producir cosechas en contra estación.
- Ahorro de agua, que se puede reciclar.
- Ahorro de fertilizantes e insecticidas.
- Se evita la maquinaria agrícola (tractores, rastras, etcétera).
- Mejor y mayor calidad del producto.

En la actualidad existen varios tipos de sistema Hidropónicos como: NFT, Raíz flotante, Aeroponía, etc. Sin embargo, la de mayor implementación ha sido la hidroponía de técnica de película de nutrientes, debido a la facilidad de la instalación y no requiere de una inversión alta para su realización. (Wilson & Suárez, 2018).

2.6.1. Sistema Hidropónico NFT.

Este tipo de sistema NFT consiste en la re-circulacion de las soluciones nutritivas que circula a través de varios canales de PVC, que permiten la alimentación de la planta, estos sistemas requieren de un cuidado especial para su correcto funcionamiento, lo cual necesitan un sistema de monitoreo para medir las condiciones ambientales como pH, temperatura y humedad, factores importantes para el crecimiento de la planta. (Beltrano & Gimenez, 2015).

- **pH:** es la medida de concentración de iones de Hidrógeno (H+) que permiten determinar el grado de acidez y basicidad de una solución. Dentro de la hidroponía la medida del pH es un factor importante para determinar la absorción de los nutrientes por parte de las raíces. Los niveles de pH óptimos para la planta se encuentra en el rango de 5.5 - 6.5 (Vargas, 2010).
- **Temperatura:** es un factor importante que influye significativamente en la absorción mineral, el desarrollo y crecimiento de las raíces. Las condiciones normales de temperatura son aproximadamente 20 ªC (Wilson & Suárez, 2018).
- **Humedad:** es un factor muy difícil de controlar dentro de un invernadero **y** al no ser monitoreado trae enfermedades de las raíces y las hojas. El rango óptimo oscila entre los 50 a 70 % de humedad relativa (Wilson & Suárez, 2018).

2.7. Trabajos Relacionados

El enfoque de las redes definidas por software en redes de sensores inalámbricos (SDWSN), consisten en introducir las características de una red SDN en redes WSN con el propósito de mejorar: la gestión y administración de los nodos sensores, consumo eficiente de energía y mejoramiento de los protocolos de enrutamiento, factores que impulsaron relacionar las SDN y WSN.(Fernández, 2018).

La primera iniciativa para aplicar SDWSN fue publicada 2002, que consiste en habilitar el protocolo de openflow dentro de un Gateway-IoT, el cual permite manejar grandes flujos de datos mediante los controladores SDN, El principal objetivo era obtener nodos sensores más confiables, que sean fácil de gestionar y mejorar el consumo de energía de la red WSN (Mostafaei & Menth, 2018).

La segunda solución consiste en una arquitectura basada en múltiples controladores, distribuidos en cada punto de acceso de la red WSN con propósito de disminuir la latencia de

la red, a través de un protocolo de prueba denominado sensor openflow (SOF), que era responsable de crear tablas de flujos dentro de los nodos sensores para tener una visión general de los dispositivos conectados a la red (Mostafaei & Menth, 2018).

Para finalizar, se tiene la arquitectura SDN-WISE (Software definido para redes de sensores inalámbricos), basado en el controlador ONOS (Open Networking Operating System), el cual está formado por nodo sensor, nodo sink y un controlador. Su funcionamiento se basa en la tabla de flujos denominada WISE-Flow-table que son encargados de gestionar el tráfico de los nodos sensores con el objetivo de proporcionar a la red mayor eficiencia en el flujo de datos y consumo de energía. Sin embargo, no es óptimo en la implementación de escenario debido al alto consumo de la memoria en los nodos sensores (Ezziyyani, 2019).

CAPÍTULO III

DISEÑO DE LA RED DE SENSORES INALÁMBRICOS DEFINIDOS POR SOFTWARE

En esta sección, se desarrolla el diseño de un modelo de arquitectura de red que permita integrar las redes de sensores inalámbricos y la red definida por software destinada al monitoreo de un sistema hidropónico NFT para cultivo de fresas por medio de la implementación de la metodología en cascada, el cual incluye un análisis de la situación actual de las dos tecnologías SDN y WSN con el propósito de definir los requerimientos y limitaciones del sistema propuesto en relación con el estándar IEEE-29148.

3.1. Metodología de Diseño

La metodología es un factor clave que permite desarrollar de forma adecuada este trabajo de titulación, que consistió en primera instancia en recolectar información a partir de la investigación bibliográfica donde se analizó los proyectos en ejecución o artículos científicos de los principales repositorios digitales.

Como resultado de la investigación bibliográfica se obtuvieron parámetros tales como: protocolos de comunicación, dispositivos, métricas de evaluación, tipo de arquitectura, etc. Estos parámetros ayudaron a determinar una propuesta que integre las dos tecnologías SDN y WSN, que mediante el apoyo de la metodología en cascada permite el despliegue y ejecución del sistema propuesta.

Por otro lado, la metodología en cascada se define como un conjunto de etapas relacionadas entre sí de forma secuencial, cada una de estas etapas tienen diferentes enfoques que ayudan a determinar requerimientos necesarios para el diseño del sistema, sin embargo, al ser un modelo lineal se debe completar cada fase antes de avanzar al siguiente nivel, debido a

que cada fase se superpone y proporciona información a la siguiente fase (Sommerville & Galipienso, 2005).

En la figura 18, se mencionan las 5 fases que comprende la metodología en cascada y las actividades que se van a realizar a lo largo del desarrollo de este trabajo de titulación.

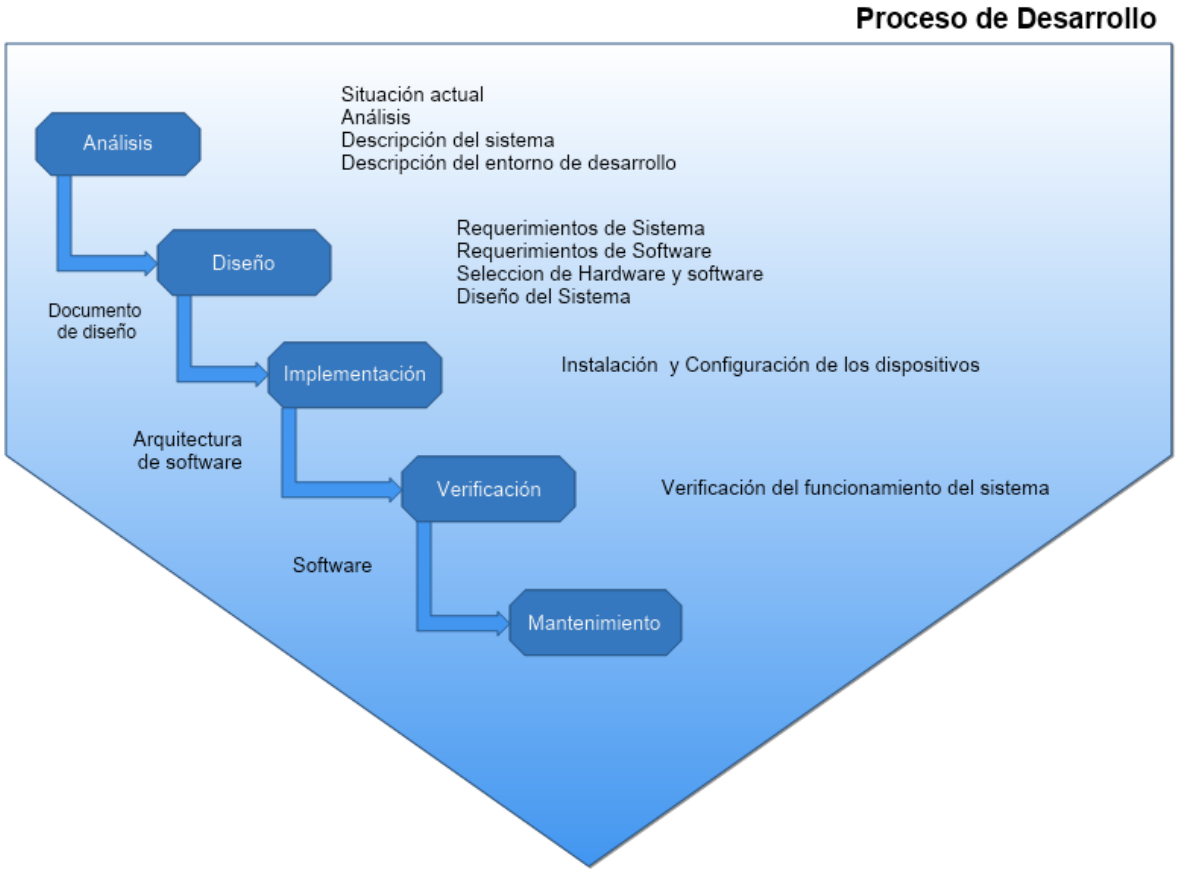


Figura 18. Diagrama de desarrollo de la metodología en cascada

Fuente: (Sommerville & Galipienso, 2005)

3.2. Fase de Análisis

Esta sección consiste en exponer la situación actual que hace referencia a las motivaciones y desafíos que representa integrar las redes de sensores inalámbricos y las redes definidas por software, como también se incluye el análisis que se llevó a cabo para determinar la solución propuesta para lo cual, se utilizó el proceso de revisión sistemática.

Con respecto a la revisión sistemática (RS) consiste en obtener información a partir de los estudios o de experimentos publicados en artículos científicos, donde se valoriza el tipo propuestas, limitaciones, protocolos, servicios y aplicaciones para integrar las dos tecnologías.

Para llevar correctamente el proceso de revisión sistemática se establece tres fases: planeación, ejecución y presentación de resultado, de este modo se obtienen resultados cualitativos y cuantitativos que permitan determinar el peso de relación entre los diferentes elementos que forman parte de una arquitectura red. Cabe mencionar que este proceso de análisis se detalla en el anexo A.

3.2.1. Situación Actual

Hoy en día, las redes de sensores inalámbricos son consideras como la tecnología más exitosa en el mercado empresarial, debido a que permite mejorar y optimizar las actividades cotidianas de los seres humanos a través de aplicaciones y servicios en línea, tales como: transporte inteligente, Smart City, Smart Home, Smart Health, agricultura de precisión etc.

Por lo general, las WSN son dispositivos inteligentes autoorganizados, con recursos limitados que aprovechan la infraestructura de las redes convencionales para transmitir y procesar información hacia el internet, y como resultado se obtiene una gran cantidad de dispositivos inteligente conectados a la red que afectan, no solo al rendimiento sino también, en el diseño y en la gestión de la infraestructura de red.

Ahora, el origen de estos problemas radica en que las redes WSN presenta limitaciones en la capacidad computacional, energía, almacenamiento, protocolos de enrutamiento y ancho de banda, estas limitaciones impiden adoptar característica de las redes definidas por software. En base a este contexto las investigaciones recientes han extendido el concepto de las redes definidas por software a las redes de sensores inalámbricas de baja potencia IEEE 802.15.4.

Las redes definidas por software es una nueva tendencia que separa el plano de control del plano de datos en los dispositivos de red a través de la lógica de programación, esta lógica de programación por lo general son aplicaciones, que ejecuta una gran cantidad de instrucciones con la finalidad de configurar el comportamiento del tráfico de datos y de los dispositivos de red de una forma automática. De esta manera la SDN se convertiría en una de las tecnologías que podría superar las limitaciones y mejorar el rendimiento de las redes WSN, entre los beneficios que proporcionaría la SDN son;

El controlador SDN es una entidad central que proporciona una visión global de la red, lo que facilita la implementación de estrategias para controlar el flujo de datos y el equilibrio de carga en los nodos sensores de una forma centralizada y, por lo tanto, reduciría consideradamente en el consumo de energía y la latencia de la red.

Además, la SDN al tener una gestión centralizada mejoría el proceso de comunicación entre los nodos sensores, al permitir el desarrollo de algoritmos que administren de manera efectiva el proceso de acceso al medio y la programación del ciclo de trabajo de los nodos, además, permitiría ejecutar y configurar mecanismo de seguridad para la red WSN y, como resultado se tendría una infraestructura de red WSN eficiente y tolerante a fallos.

Como se mencionó anteriormente, las limitaciones de las redes WSN implican un gran desafío al momento de incluir las características de la SDN, por ejemplo, la mayoría de los dispositivos de la red WSN se hallan en un estado de suspensión para ahorrar el consumo de energía, lo que dificulta establecer un canal de comunicación constante entre los nodos y el controlador SDN que es necesario para obtener información de la red.

Ahora bien, ejecutar el protocolo openflow en cada uno de los nodos sensores representaría la saturación del ancho de banda de las redes WSN, ya que los nodos utilizan un mismo medio de comunicación para transmitir información, por otro lado, los sistemas

operativos destinados a la configuración de los nodos sensores no soportan la instalación de tablas y reglas de flujo que se requieren en las redes SDN (Davoli et al., 2019).

3.2.2. Análisis

A partir de la situación actual y análisis bibliográfico descrito en el anexo A, se deduce que la adopción de las redes definidas por software en dispositivos de baja potencia ha representado un gran desafío para los investigadores, sin embargo, se pude distinguir dos tipos de propuestas: las redes de sensores programables y las redes WSN enfocadas en SDN, ambas propuestas presentan diferentes enfoques y funcionalidades.

Con lo que se refiere a las redes de sensores programables su funcionamiento se centra en configurar la capa física y MAC para que los nodos sensores adopten ciertas características del protocolo openflow por medio de la lógica programación. Los nodos sensores recrean las tablas de flujo donde almacena instrucciones enviadas por el controlador SDN que juntamente con las aplicaciones permiten crear reglas de flujo, de modo que ayuden a administrar el funcionamiento de los nodos sensores y de la red. Sin embargo, las propuestas que tiene en este enfoquen presenta sobrecarga en los protocolos de descubrimiento de topología y procesamiento de datos. Por otro lado, las redes WSN enfocadas en SDN intentan aprovechar las características de diferentes tecnologías como SDN, WSN y NFV para proporcionar una red inteligente, escalable e interoperable.

Ahora bien, se puede observar que la mayoría de las soluciones utilizan las funcionalidades del protocolo openflow para obtener características de una red SDN en una red de bajo recursos como son las WSN. Sin embargo, el protocolo openflow tiene una capacidad limitada para admitir nuevos estándares o protocolos debido a que openflow define un formato de paquetes que solo puede ser reconocidos y procesado por dispositivos que soporte openflow

Tomando en cuenta este contexto la ONF(*Open Networking Foundation*) propone un modelo de arquitectura abstracta para obtener tener un plano de datos parcialmente programable por un controlador SDN. Este modelo abstracto define tres parámetros:

- Este modelo abstracto debe permitir que el controlador SDN programe los dispositivos de plano de datos sin estar vinculado el protocolo openflow, garantizando la coexistencia de redes heterogéneas
- Debe definirse una estructura de paquetes que pueda ser interpretado por el protocolo openflow para su correcto procesamiento
- El controlador SDN deber permitir por lo menos modificar o habilitar las rutas de los flujos de datos.

3.2.3. Descripción general del sistema

Teniendo en cuento las observaciones anteriores se propone una solución enfocada a la gestión de flujo de datos, donde se intenta aprovechar el direccionamiento IP local de las redes 6lowpan y el enrutamiento centralizado de las redes SDN, de manera que permita habilitar el flujo de datos por medio de políticas envidas por el controlador SDN. Dentro de la solución se contempló que la red WSN es independiente a la red SDN, por lo que facilita la programación y ejecución de protocolos que soporten los nodos sensores, sin afectar su rendimiento en relación con el procesamiento y consumo de energía. Además, en esta propuesta se añade la virtualización de funciones de red para ejecutar el protocolo openflow de modo que permita crear interfaces virtuales para conectar múltiples redes, de tal forma que compartan la misma infraestructura, lo que permitiría desarrollar nuevos servicios y aplicaciones. En la figura 19, se presenta el diseño de la infraestructura general que está compuesto por cuatro capas conocidas como: adquisición, datos, control y aplicación.

La capa de adquisición está formada por dispositivos de diferentes tecnologías, en este caso se utiliza dispositivos que soporte el estándar 802.15.4 con el objetivo de habilitar el protocolo 6LOWPAN, ya que permiten proporcionar a cada uno de los nodos sensores una dirección IPv6, facilitando la interoperabilidad entre la red WSN y la red de internet.

Los nodos sensores inalámbricos son encargados de recolectar la información del ambiente dentro del invernadero, esta información se basa en los tres parámetros mencionados en el capítulo II que son: temperatura ambiente, humedad del sustrato y niveles de pH de la sustancia nutritiva, cada uno de estos nodos utiliza el protocolo RPL para establecer una topología en malla y enviar la información comunicarse inalámbricamente a través del estándar 802.15.4.

En cuanto, a la capa de datos está compuesta por el Gateway y el conmutador open vswitch, este último elemento es un punto de convergencia entre las dos redes WSN y SDN, es decir todo el flujo de datos generado por cada sensor se concentra en el conmutador a la espera que el controlador SDN habilite el flujo de datos hacía el servidor.

El nodo Gateway realiza dos funciones; primero coordina la comunicación entre los nodos a través del protocolo RPL; y el segundo permite el tráfico UDP de cada uno de los sensores hacia el servidor UDP que se aloja en la capa de aplicación.

Con respecto a la capa de control, está formado por el controlador SDN que junto al *switch open vswitch* realizan el proceso para gestionar el paso del tráfico de datos. Este proceso inicia en el momento que se introducen reglas de flujo en el *switch open vswitch*. Al recibir un paquete, el *switch* lo procesa y compara con la tabla de flujos, al encontrar una coincidencia reenvía el que el paquete hacia su destino, en este caso al servidor UDP.

La última, etapa corresponde a la capa de aplicación y servicios donde se establece la comunicación entre el cliente y servidor UDP para luego transmitir los datos hacia el internet por medio del protocolo MQTT, donde son almacenados en una base de datos y luego visualizados en una página web.

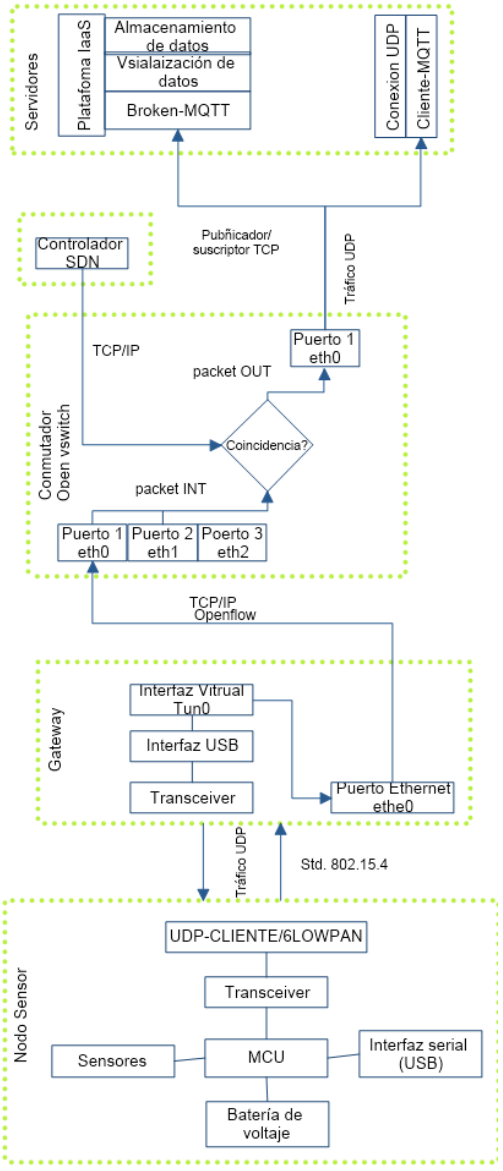


Figura 19. Esquema general de la arquitectura propuesta

3.2.4. Descripción del Entorno de desarrollo

En la figura 20 se muestra el escenario donde se llevó a cabo el proyecto, el cual consiste en un invernadero hidropónico dedicado al cultivo de fresas que se halla ubicado en el barrio Domingo Savio perteneciente a la parroquia de Quiroga, cantón Cotacachi, cuyas características

son las siguientes; tiene un diámetro de 4 metros, ancho de 4 metros y de altura 3 metros, la mayoría de la infraestructura se halla cubierta de plástico y, en el interior del invernadero se ubica cuatro tubos de PVC de 4 pulgadas por 3 metros, que son el soporte para colocar la semilla de la frutilla.

Figura 20. Invernadero de cultivo de fresas en base sistema hidropónico NFT

3.3. Fase de diseño

Ahora bien, para dar cumplimiento con el objetivo de implementar y desarrollar la solución planteada en la sección 3.2.3, es necesario determinar las características y atributos de las dos tecnologías, por lo que, se utiliza el estándar ISO/IEC/IEEE 29148 que define estrategias relacionadas con la ingeniería de requerimientos de software y hardware.

3.3.1. Requerimientos

Los requerimientos es la base fundamental para determinar los requisitos necesarios para el diseño y la funcionalidad del sistema, por ende, el estándar ISO/IEC/IEEE 29148 define los requerimientos stakeholders, sistema y arquitectura que ayudan a especificar y evaluar los

requerimientos en relación con las necesidades de la solución planteada. Sin embargo, al considerarse una investigación de carácter investigativo, los requerimientos operacionales, sistema y arquitectura son planteados acorde a la información recolectada de la investigación bibliográfica. Considerando este contexto en la tabla 7 se muestra la lista de stakeholder que contribuyeron y supervisaron el desarrollo del proyecto

Tabla 7.
Lista de Stakeholder

Lista de Stakeholders
1. Universidad Técnica del Norte
2. Facultad de Ciencias Aplicada (FICA)
3. Ing. Edgar Maya
4. Ing. Mauricio Dominguez
5. Ing. Fabian Cuzme
6. Sr. Juan Pablo Bautista

Fuente: Elaborado por el autor.

Por otra parte, el estándar ISO/IEC/IEEE/29148 hace uso de acrónimos y abreviaturas con la finalidad de identificar el tipo de requerimiento y tener un mejor manejo de la información. En la tabla 8 se define la abreviatura que se va a utilizar en el transcurso de este proyecto.

Tabla 8.
Requerimientos estipulados en el estándar ISOC/IEC/IEEE-29148

Abreviatura	**Significado**
StRS	Requerimientos de Stakeholder
SyRS	Requerimiento de Sistema
SRSH	Requerimientos Funcionales

Fuente: Elaborado por el autor

3.3.1.1. Requerimientos Operacionales

Los requerimientos operación no solo reflejan el funcionamiento final del sistema, sino también especifican los requisitos generales que permitan relacionar las funciones de las redes WSN y SDN. En la tabla 9 se describe una lista de requerimientos que junto con el análisis descrito en el anexo A, marcan el punto de partida para determinar los requerimientos del sistema y arquitectura

Tabla 9.
Lista de requerimientos operacionales

Abreviatura	**Requerimientos**	**Prioridad**		
		Alta	Medio	Baja
Requerimiento: StRS Operacionales				
StRS1	La plataforma debe visualización de los datos censados por los sensores		x	
StRS2	Almacenamiento de los datos generados por los nodos sensores		x	
StRS3	Los dispositivos deben tener accesibilidad al internet			x
StRS4	Los nodos sensores deben soportar direccionamiento IPv6	**x**		
StRS5	El controlador SDN maneja la comunicación entre nodo sensor y servidor, como también el acceso hacia el internet	**x**		
StRS6	El switch openflow debe permitir procesar direcciones IPv6	**x**		
StRS7	Switch openflow debe conectar el Gateway y el servidor	**x**		

Fuente: Elaborado por el autor

3.3.1.2. Requerimiento de Sistema (SyRS)

De acuerdo con el estándar ISO/IEC/IEEE/29148 y los Requerimientos de Ingeniería de Software (RES), los requerimientos del sistema son factores importantes en el diseño de la red, ya que permite definir las funciones específicas que van a realizar los dispositivos dentro del sistema. En la tabla 10, se plantea una lista de requerimientos que debe soportar los dispositivos.

Tabla 10.
Lista de requerimientos Sistema

Abreviatura	Requerimiento de Sistema	Prioridad		
		Alta	**Medio**	**Baja**
SyRS 1	Los nodos sensores medirán la temperatura del ambiente del invernadero, humedad del sustrato y pH de la solución nutritiva	x		
SyRS 2	Conversión analógicos/digitales		x	
SyRS 3	Comunicación por medio del puerto UART	x		
SyRS 4	Los nodos sensores forman una en topología en malla		x	
SyRS5	Se establece una comunicación cliente-cliente servidor por medio del protocolo UDP	x		
SyRS 5	El Gateway sincronizara la comunicación entre los nodos por medios del protocolo RPL		x	
SyRS 7	El conmutador open vswitch analizara los campos de cada paquete UDP hasta encontrar una coincidencia	x		
SyRS 8	El conmutador open vswitch tendrá acceso al internet		x	

SyRS 9	Controlador SDN permite habilitar el tráfico de datos a través de las reglas de flujo	x
SyRS 10	Facilidad para crear redes y evaluar escenarios	x

Fuente: Elaborado por el autor

3.3.1.3. Requerimientos de Arquitectura

Está sección se describe las propiedades que debe cumplir el hardware y software para que el sistema funcione de una manera correcta. En la tabla 11, se especifica las características necesarias para los dispositivos que van a formar parte del sistema.

Tabla 11.
Requerimientos de arquitectura

Abreviatura	**Requerimiento Arquitectura**	**Prioridad**		
		Alta	**Medio**	**Bajo**
Requerimiento: SRSH- Hardware				
SRSH1	Disponibilidad de pines analógicos y digitales	x		
SRSH2	Incluye o facilita la integración de módulos USB (UART)	x		
SRSH3	Disponibilidad de puertos Ethernet	x		
SRSH4	Permite incluir adaptadores USB-Ethernet	x		
SRSH5	Compatible con arduino		x	
SRSH6	Disponibilidad	x		
Requerimiento: SRSH- Software				
SRSH7	Soporte a las diferentes distribuciones de sistemas operativos		x	
SRSH8	Soporte a php, mysql y apache,		x	
SRSH9	Soporte a las librerías de Python			
SRSH10	Soporte open vswitch	x		
Requerimiento: SRSH- lógicos				
SRSH11	Protocolo UDP/6Lowpan	x		
SRSH12	Protocolo 802.15.4	x		

SRSH13	Protocolo MQTT		x
SRSH14	Protocolo RPL	x	
SRSH15	Soporte para configurar interfaces virtuales	x	
SRSH16	Soporte del protocolo Openflow versión 3		
Requerimiento: SRSH- Eléctrico			
SRSH17	Se requiere una fuente voltaje de 5 voltios a 3 amperios para alimentar los dos raspberrypi Baterías de 5 y 3,3 voltios para los nodos sensores	x	

Fuente: Elaborado por el autor

3.3.2. Elección de Software y Hardware para el sistema

En este capítulo se centra en evaluar los componentes de hardware y software de acuerdo con los requerimientos que se especificaron anteriormente, para ello se realiza un proceso de comparación entre los diferentes dispositivos. Este proceso de comparación consiste en asignar un valor de 1 y 0, donde 1 = si cumple, y 0 no cumple. Al final se contabiliza los valores y selecciona el dispositivo que obtenga el mayor valor.

3.3.2. Elección de Hardware

Este apartado describe el proceso de selección de los dispositivos que van a cumplir, las funciones de nodo sensor, gateway y switch openflow, que son requeridos en la solución propuesta. Con respecto a los nodos sensores incluyen la selección del hardware de comunicación inalámbrica y los sensores para medir como ph, humedad y temperatura.

Selección del hardware de comunicación inalámbrica

En la actualidad existen diferentes módulos de comunicación destinados a redes de sensores inalámbricos de bajo potencia, que en algunos casos se encuentran integrados en la placa de procesamiento. En la tabla 12 se exponen el análisis de módulos de comunicación a partir de los requerimientos.

Tabla 12.
Selección de los módulos de comunicación inalámbrica

Módulo de comunicación	SyRS2	SyRS4	SRSH1	SRSH6	SRSH11	SRSH13	Total
Bluetooth	0	1	0	1	0	0	1
Xbee S2	1	0	0	1	1	0	2
Telosb	1	1	1	1	1	1	6

Fuente: Elaborado por el autor

De acuerdo con los resultados obtenidos en la tabla 12, se escoge el módulo tmote sky (telos B), que es un módulo *system on chip* (SoC) que integra en una misma placa sensores, microcontrolador y módulo de comunicación como se indica en la figura 21.

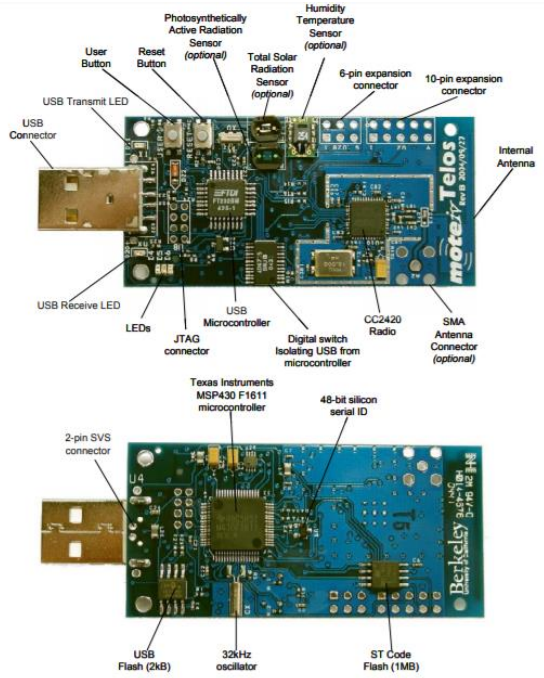


Figura 21. Mota Tmote sky (Telosb)

Fuente: (Polastre et al., 2005)

Estos dispositivos pertenecen a la familia de los microcontroladores *Texas Instruments MSP430* y el chip CC2440, y permiten establecer una comunicación inalámbrica basada en el estándar 802.15.4, además estos módulos operan con software de código libre como son TinyOS y Contiki, que soportan la pila de protocolos TCP/IP y 6LOWPAN requeridos StRS4.

Otras de las características que lo hace factible para su utilización es que integra sus propios sensores como de humedad, temperatura relativa y de luminosidad.

Cabe mencionar que a pesar de que el módulo tmote sky (telos B) cuenta con sus propias entradas analógicas/digitales para conectar sensores externos no cuenta con la información necesaria para agregar sensores de humedad de suelos y pH que se requieren en este proyecto, por ende, se ve obligado a seleccionar un hardware compatible con el módulo telos B.

Este hardware debe permitir conectar los sensores requeridos SyRS 1 y sobre todo debe ser compatible con el módulo telos B para transmitir la información. En la tabla 13, se muestra un listado de dispositivos que se tomaron en cuenta para el análisis.

Tabla 13.
Selección de la placa de procesamiento

Placas de procesamiento	SRSH1	SRSH2	SRSH6	Total
Ardino uno	1	1	1	3
Raspberry pi 3b	1	0	1	2

Fuente: Elaborado por el autor

Según los resultados que se muestran en la tabla 12 se elige la placa de arduino uno, ya que dispone de 14 conectores de entrada/salidas para señales y seis conectores para conexión de señales analógicas, lo que facilita conexión de varios sensores, otra de las características que lo hace al arduino uno, el adecuado para este proyecto es que permite la integración de diferentes módulos como por ejemplo modulo ethernet, xbee, shield USB, etc. En la figura 22 se indica la placa Arduino uno.

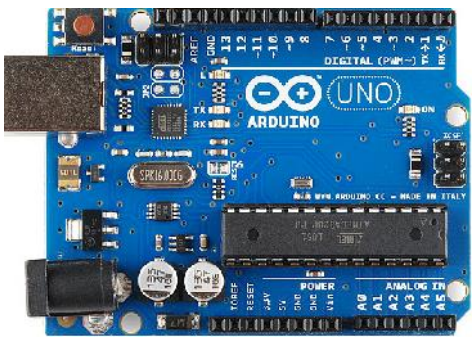


Figura 22. Tarjeta arduino uno

Fuente: (Aldea, n.d.)

El Arduino cumplirá con la función de enviar la información de los sensores de humedad de suelo y pH, a través de una comunicación serial-UART que es posible mediante la utilización de un shield host USB que se muestra en la figura 23, y tendrá el desempeño de un dispositivo host, es decir, la placa Arduino solo enviará los datos de los sensores y de suministrar energía necesaria para que funcione correctamente módulo telosB que actuará como un transceiver (A. M. Muñoz, n.d.).

Figura 23. Shield host USB

Fuente: (Electronics, 2018.)

Sensores

En este apartado se selecciona los sensores de temperatura, humedad del suelo y de Ph, que ayudan a tomar datos del cultivo hidropónico, con respecto al sensor de temperatura relativa se utilizará el sensor integrado en el módulo telosB denominado SHT11 con la finalidad de ahorrar recursos económicos y hardware; en la tabla 14 se exponen las características del sensor SHT11.

Tabla 14.
Características técnicas del sensor SHT11

Sensor	**SHT11**
Tipo	Humedad relativa

	Temperatura relativa
Parámetros	Características
Señal de salida	Digital
Rango de operación	-40 - 123.8 °C
Resolución	Min 0.04 °C Max 0.01 °C
Tiempo de respuesta	5 seg
Fuente de alimentación	Min 2.4 v Max 5.5 v

Fuente:(Sensor & Kit, n.d.)

Con respecto al sensor de pH, este cumple con la función de medir el nivel de acidez de la sustancia nutritiva que circula en los tubos de sistema hidropónico. En la tabla 15, se expone los sensores que pueden ser usados en este proyecto.

Tabla 15.
lista de sensores de pH

Tipo	SRSH1	SRSH5	SRSH6	Total
Sensor pH 4502c	1	1	1	3
Sensor pH-genérico	1	1	1	3

Fuente: Elaborado por el autor

Según el análisis de la tabla 14 ambos sensores cumplen con los requerimientos, sin embargo, para el desarrollo de este proyecto se elige Sensor pH-genérico, ya que representa un costo económico bajo en el mercado con valor de 45 USD (dólar estadounidense), a diferencia del Sensor pH 4502c que tiene un valor de 72 USD. En la tabla 16 se indica las características técnicas del sensor de pH genérico.

Tabla 16.
Características del sensor de pH

Sensor	Sensor pH-genérico
Parámetros	Características
Voltaje de alimentación	5 voltios

Corriente	10 miliamperios
Rango de Medición	0 a 14 pH

Fuente: (Orgone Tecnology, 2015.)

Mientras que, para medir el nivel de humedad del sustrato ubicado en el interior de los tubos PVC, se considera los sensores de humedad disponibles en mercado que se diferencian por dos características por su material y resistencia a la corrosión. En la tabla 17 se indica el proceso de evaluación y selección del sensor idóneo para este proyecto.

Tabla 17.
Lista de requerimientos de sensores humedad

Tipo	SRSH1	SRSH5	SRSH6	Total
Sensor de Humedad FC-28	1	1	1	3
Sensor de Humedad Hd-38	1	1	0	2

Fuente: Elaborado por el autor

De acuerdo con la tabla 17, se escoge el sensor de humedad modelo FC-28 por su disponibilidad en mercado como también, su valor económico que representa 3.35 USD (dólar estadounidense) a diferencia del modelo Hd-38 que su valor económico es de 30 USD. La variación de precios de estos dispositivos es debido al tamaño y al material que están construidos los electrodos del sensor. En la tabla 18, se muestra las características técnicas del sensor de humedad modelo FC-28.

Tabla 18.
Características técnicas del sensor de humedad FC-38

Sensor	FC-28
Tipo	Humedad de suelo
Parámetros	Características
Señal de salida	Digital/analógica
Voltaje de alimentación operativa	3.3v – 5.5v
Corriente	35 mA

Dimensiones	60 x 20 x 5 mm

Fuente: (A. M. Muñoz, n.d.)

Selección hardware para el gateway y switch openflow

Por otro lado, para el funcionamiento del gateway y switch openflow se consideró los dispositivos raspeberry PI en sus diferentes modelos 3b y 4 como también la placa rock PI 4. En la tabla 19 se compara las características significativas de cada uno de los dispositivos con la finalidad de seleccionar el más idóneo para el desarrollo del sistema.

Tabla 19.
Características técnicas de la placa raspberry PI

Parámetros	**Raspberry pi 3B**	**Raspberry pi 4**	**Rock PI 4**
Procesador	Broadcom BCM2837B0, Cortex-A53	Broadcom BCM2711, quad-core Cortex-A72	hexa core processor Dual Cortex-A72, quad Cortex-A53
Frecuencia	1.4 Ghz	1.5 Ghz	1,4 Ghz 1.85 Ghz
Tipo de memoria RAM	LP-DDR2	LP-DDR4	LP-DDR4
Puertos Ethernet	Gigabit Ethernet	Gigabit Ethernet	Gigabit Ethernet con soporte de PoE
WIFI	2.4GHz y 5GHz IEEE 802.11.b/g/n/ac	2.4 GHz y 5 GHz IEEE 802.11b/g/n/ac	2.4 GHz y 5 GHz IEEE 802.11 ac
Puertos USB	4 puertos USB 2.0	2 puertos USB 2.0 2 puertos USB 3,0	2 puertos USB 3.0 1 puerto USB 2.0
Tarjeta SD	Tipo- microSD	Tipo- microSD	Tipo- microSD
Pines GPIO	40 pines	40 pines	40 pines
Fuente de alimentación	5 voltios Tipo micro USB	5 voltios USB Tipo-C	5 voltios USB Tipo-C
Compatibilidad sistema operativos	Raspbian, Ubuntu-IOT	Raspbian, Ubuntu-IOT	Windows 10 Ubuntu 20.4 server

Costos	85	96.50	120

Fuente: Elaborado por el autor en base a los datasheet de cada uno de los dispositivos

Gateway

Para dar cumplimiento con el requerimiento SyRS5 el Gateway debe sincronizar la comunicación entre los nodos sensores para facilitar la comunicación y el reenvío del tráfico UDP hacia la red LAN (*Local Area Networking*) donde se aloja el servidor UDP. En la tabla 19 se evalúa cada uno de los dispositivos descritos en la tabla 20 de acuerdo con los requerimientos establecidos.

Tabla 20.
Selección del hardware para el Gateway

Tipo	SRSH3	SRSH4	SRSH6	SRSH7	SRSH9	Total
Raspberry pi 4	1	1	1	1	1	5
Rapberry pi 3b	1	1	1	1	1	5
Rock pi4	1	1	0	0	1	3

Fuente: Elaborado por el autor

Según la tabla 19 se puede decir que los dispositivos raspberary PI en sus modelos 3B y 4 son idóneos para el desarrollo del sistema descartando totalmente al rock PI 4, sin embargo, si se observa la tabla 18, placa rock PI 4 tiene un procesador de seis núcleos que puede alcanzar una frecuencia de 1.4 a 1.85 GHz (Gigahertz), lo que le permite ejecutar múltiples tareas de manera eficiente a diferencia de los procesadores de los raspberry pi que son quad core y alcanzan una frecuencia de 1.4 o 1.5 GHz.

Otra característica que hace al rock PI 4 mejor que las placas raspberry PI es el tipo de memoria, ya que utiliza una memoria LP-DDR4(Low-Power Double Data Rate) que tiene la capacidad de transmitir datos a una velocidad de 3200 Mbps permitiéndole al rock PI 4 cargar información en menor tiempo y ejecutar programas de manera temporal y simultánea, en cambio, la placa raspberry

Pi modelo 3B tiene una memoria LD-DDR2 que alcanza a transmitir datos a una velocidad de 800 Mbps

Considerando esta comparación se puede decir que el rock PI 4 es factible para el desarrollo de este proyecto, pero su adquisición representa contratiempos ya que la compra se lo realiza bajo pedido en tiendas electrónicas debido a que no es conocido en el mercado nacional como son las placas raspberry PI, además su costo económico es 120 USD, mucho más costoso que las placas raspberry pi que no superan los 100 USD.

Para finalizar este proceso de evaluación, se selecciona el raspberry PI modelo 3B debido a que el gateway no se va a ejecutar una gran cantidad de procesos que requieran el máximo uso de la memoria y del procesador del dispositivo, ya que contiki-ng solo requiere 10 kilobytes de memoria RAM para ejecutarse correctamente en la placa raspberry PI.

Switch Openflow

Para la selección del dispositivo que va a actuar como switch virtual se considera los requerimientos SyRS6, SyRS7 y SyRS8 que definen el funcionamiento switch openflow y habilita el protocolo openflow en su versión 1.3. En la tabla 21, se muestran los dispositivos que se analizaron para la instalación del open vswitch.

Tabla 21.
Selección del conmutador open vswitch

Tipo	SRSH3	SRSH4	SRSH6	SRSH7	SRSH10	Total
Rapberry pi 4	1	1	1	1	1	5
Rapberry pi 3b	1	1	1	1	1	5
Rock pi4	1	1	0	0	1	3

Fuente: Elaborado por el autor

Ahora, como se indica en la tabla 21, los dos modelos de raspberry pi cumplen con los requerimientos del sistema, por el cual, se vio en la obligación de considerar los requerimientos mínimos de instalación del open vswitch que se detallan en la tabla 22, como también se recuerda que el software open vswitch demanda un alto consumo de los recursos del dispositivo debido a

que se ejecuta en el kernel del sistema operativo anfitrión. Tomando estos atributos se selecciona la placa raspberry pi más idónea para este sistema

Considerando el anterior contexto y la tabla 21, se selecciona la placa raspberry pi modelo 4, la cual tiene mejores prestaciones frente respberry pi modelo 3B, debido a que tiene un procesador quad core de 1.5 Ghz, lo que permite procesar información un poco más rápido en relación con el raspberrypi 3B que, cuenta con un quad core de 1.4 Ghz. Otra ventaja que tiene el raspberry PI 4 con relación al modelo 3B, es la memoria LP-DDR4, que admite una transferencia de datos a 3200 Mbps lo que facilitaría el funcionamiento del open vswitch correctamente

Tabla 22.
Requerimientos mínimos para la instalación de open vswitch

Hardware	**Especificación**
Procesador	Dual core Frecuencia mayor 1.4 GHz
Almacenamiento	16 Gb
Memoria	1 Gb

Fuente: (Emmerich et al., 2018)

3.3.2.2. Selección de software

Esta sección corresponde a seleccionar el software que va a formar parte del sistema propuesto, el cual inicia seleccionando el software de control SDN encargado de administrar el flujo de datos de los sensores, además se incluye la selección de la plataforma de almacenamiento en la nube y la plataforma web que permita mostrar los datos censados.

Selección del controlador SDN

El rol que va desempeña el controlador SDN dentro de las redes WSN es limitado, sin embargo, su utilización permitirá manejar eficientemente las peticiones de un cliente UDP hacia el servidor mediante la dirección IP del nodo, de esta manera se tendría una red eficiente en la entrega de datos y en el acceso a la red WSN. En la tabla 23 se exponen una lista de software

de control SDN basado en código abierto que van a ser analizados con relación a los requerimientos del sistema.

Tabla 23.
Selección del controlador SDN

Controlador	StRS5	SyRS8	SyRS9	SyRS10	SRSH110	SRSH15	SRSH16	Total
ONOS	1	1	1	0	1	1	1	5
Opendaylight	1	1	1	0	1	1	1	5
Ryu	1	1	1	1	1	1	1	6
Floodlight	1	1	1	0	1	0	1	4

Fuente: Elaborado por el autor

De acuerdo con la tabla 23, se selecciona el controlador Ryu porque presenta una gran cantidad de aplicaciones que utilizan atributos que incluyen direcciones IP de origen y destino manejo del protocolo ICMPv6, TCP/IP, etc. Estos atributos son soportados por las redes 6lowpan por lo que facilita la ejecución de las aplicaciones que ofrece el controlador Ryu, entre las aplicaciones que proporciona el controlador ryu y que pueden representar un beneficio para las redes WSN son simple_switch, traffic monitor, router, firewall, túnel, etc.

Otro aspecto a considerar en la selección del controlador Ryu, es el rendimiento del controlador, que de acuerdo con un estudio realizado por Mamushiane Lusani en el 2018 sobre "A comparative evaluation of the performance of popular SDN controllers", el controlador Ryu tiene mejores prestaciones en redes de prueba, a comparación de los controladores ONOS, Opendaylight y Floodlight que son aplicados en redes extensas (Mamushiane et al., 2018).

Servicio de almacenamiento en la nube

Considerando los requerimientos StRS1, StRS2 y StRS3 referentes con el almacenamiento y el desarrollo de una plataforma web para la visualización de los datos censados; en la tabla 24 se realiza una comparación entre los principales proveedores de almacenamiento en la nube destacándose Microsoft Azure, Amazon Web y Google Cloud.

Tabla 24.
Selección de la Infraestructura como Servicio

	StRS1	StRS2	SRSH10	SRSH7	SRSH8	Aprecia
Microsoft Azure	1	1	1	1	1	5
Amazon Web Services	1	1	1	1	1	5
Google Cloud Platform	1	1	1	1	1	5

Fuente: Elaborado por el autor

De acuerdo con el análisis de la tabla 24, las tres plataformas ofrecen los mismos servicios para almacenar datos en el internet y creación de máquinas virtuales. Sin embargo, se elige Google cloud, ya que proporciona mayor información sobre su instalación y configuración para el desarrollo de aplicaciones y servicios en la nube.

Ahora bien, el desarrollo de la aplicación web para la visualización de datos representa un trabajo laborioso, no obstante, existe plataformas de código abierto que permiten interactuar al usuario con los datos obtenidos de los sensores de una manera eficiente. Tomando en cuenta esta disposición, en la tabla 25, se muestran algunas plataformas web que pueden ayudar al desarrollo de este proyecto.

Tabla 25.
lista de plataformas web basado en código abierto

	StRS1	StRS2	SRSH7	SRSH8	Valoración
Zabbix	0	0	1	1	3
Node red	0	1	1	1	3
Grafana	1	1	1	1	4
XAMPP	1	1	1	1	4

Fuente: Elaborado por el autor

De la tabla 25 se puede decir que la plataforma zabbix no permite la visualización y el almacenamiento de los datos censados por los sensores, además no cuenta con documentos o

repositorios guías que permitan realizar la instalación y configuración correctamente, mientras que la plataforma node-red permite el almacenamiento de datos, y la visualización de los elementos de la red WSN, pero no visualización de datos censados por los sensores.

Ahora con respecto a grafana y XAMPP ambos cumplen con los requerimientos del sistema, pero se elige XAMPP por su simplicidad de instalación y por qué es compatible con la mayoría de los sistemas operativos Linux y Windows, además XAMPP reúne las principales herramientas para el desarrollo de páginas web como es apache, MYSQL y PHP. Sin embargo, si se desea tener mejor presentación de los datos es recomendable instalar Grafana, ya que permite trabajar con diferentes gestores de base de datos como también permite la facilidad de representar gráficas de los datos censados.

3.3.3 Diseño de Sistema

Una vez, que se ha seleccionado el hardware y software que van a formar parte de la arquitectura propuesta, el siguiente paso es el diseño del sistema que consiste en representar gráficamente el proceso de funcionamiento de los principales componentes del sistema, que incluye el diagrama de conexión, direccionamiento IP y topología del sistema.

Diagrama de bloques del sistema

En la figura 24 se representa la relación entre los diferentes componentes del sistema, el cual, inicia conectando cada sensor a un puerto analógico de la placa Arduino, mismo que al recibir un dato es procesado en el microprocesador para luego ser enviado hacia al módulo tmote sky (telosB) que está conectado al puerto USB de la placa shield host USB. Una vez que los datos llegan al módulo tmote sky (telosB) son almacenados temporalmente para luego ser encapsulados en una trama UDP y enviarlos hacia el nodo coordinador por medio de una comunicación inalámbrica que soporte el estándar 802.15.4.

Este nodo coordinador está conectado a un puerto USB del raspberry pi-gateway que crea una interfaz virtual denominada tun0 y que se asocia con la interfaz ethernet del raspberry para permitir la comunicación de la red de sensores con una red LAN externa.

Por otro lado, la interfaz física ethernet (eth0) del raspberry Pi-gateway está conectado a un puerto del switch openflow, el cual está formado por el software open vswitch que proporciona a la placa raspberry Pi las funcionalidades de un switch administrable, es decir permite gestionar el flujo de datos que ingresan a los puertos del switch por medio del protocolo openflow.

El open vSwitch permite crear una interfaz virtual denominado bridge que agrupa las interfaces eth0, eth1 y eth2 de placa raspberry Pi, cada una de las interfaces se convierten en puertos del switch openflow que conectan las máquinas virtuales donde se ejecutan el controlador Ryu y el servidor local.

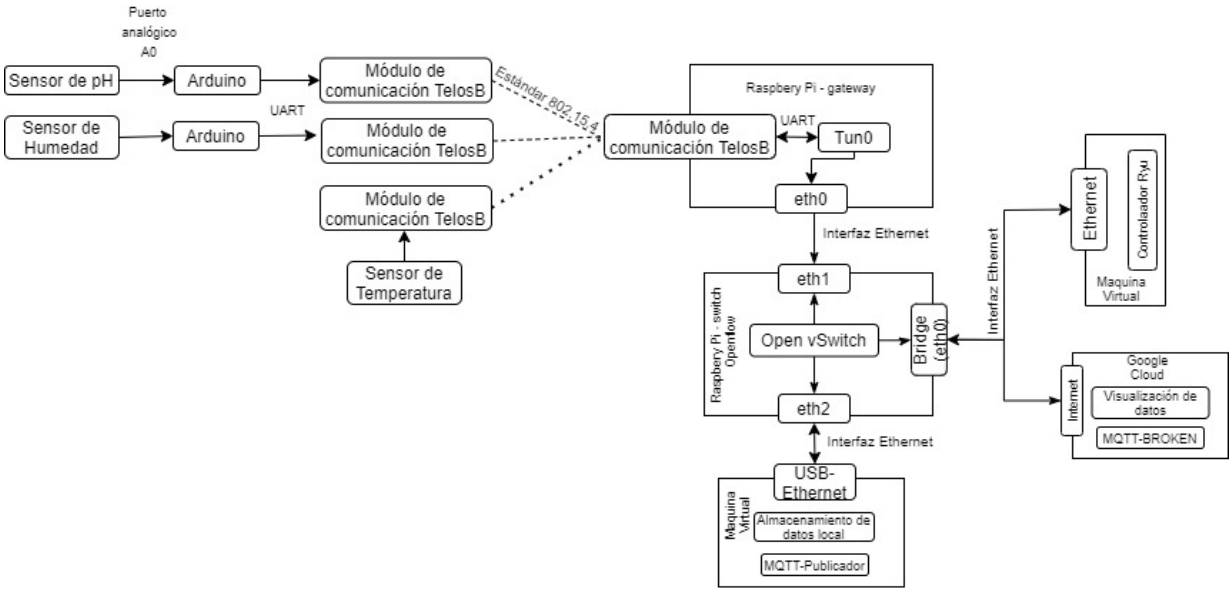


Figura 24. Diagrama de bloques del sistema

Diagrama de flujos del Sistema

En esta sección consiste en definir el funcionamiento de los dispositivos del sistema y de la aplicación SDN y es así que, en la figura 25se detalla el proceso de inicio de los nodos sensores, que consiste en habilitar el puerto UART para recolectar la información de los

sensores conectados al arduino, al mismo tiempo el nodo envía mensajes de sincronismo por medio de protocolo ICMPv3, estos mensajes de sincronismo cumplen con dos funciones: la primera es establecer comunicación con el nodo coordinador para habilitar el protocolo RPL y la segunda es mantener una comunicación con el servidor UDP para la transmisión de datos. En el caso que no exista una conexión con el servidor, el módulo de comunicación del nodo sensor se apagara durante un instante, de esta manera se pretende reducir el consumo de energía.

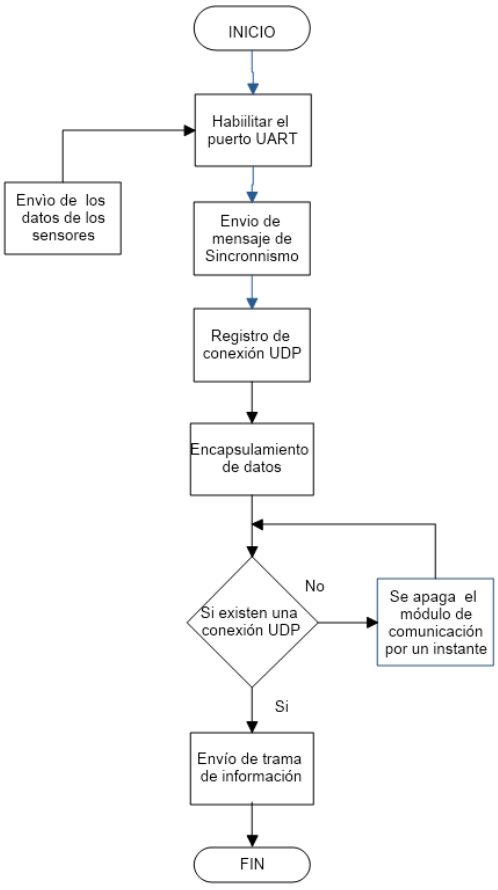


Figura 25. Diagrama de flujos de los nodos sensores

Por otro lado, en figura 26, se evidencia el proceso de funcionamiento que realiza el nodo gateway que es responsable de habilitar un canal para sincronizar los nodos a través del protocolo RPL, como también es encargado de reenviar los paquetes UDP hacia la red LAN para su almacenamiento y procesamiento.

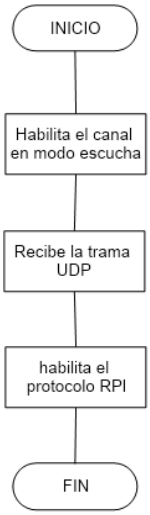


Figura 26. Diagrama de funcionamiento del nodo Gateway

Por último, en la figura 27, se describe el funcionamiento de la aplicación SDN, que tiene la función de habilitar él envió de datos hacia al servidor local y a la nube por medio de direcciones IP. Este funcionamiento inicia verificando la comunicación entre el controlador ryu y el switch openflow, en el caso de existir una conexión el usuario puede registrar la dirección IP del nodo y el protocolo UDP o ICMPv3 que desee habilitar por medio de reglas de flujo que son insertadas en la tabla de flujos del open vswitch.

Cuando un paquete proveniente del gateway llega al switch openflow es analizado con cada uno de los parámetros del campo coincidente que se indicaron en la figura 14, con la finalidad de determinar una coincidencia, al existir una coincidencia el paquete es procesado y enviado al puerto destino en este caso hacia el servidor UDP donde los datos censados son almacenados en una base de datos.

Con respecto al servidor UDP no solo procesa los datos, sino que es encargado de enviar constantemente solicitudes hacia la plataforma Google cloud con la finalidad de verificar su disponibilidad, si las solicitudes son rechazadas se debe registrar la dirección IPv4 del servidor y protocolo TCP en el controlador SDN, de esta manera permite habilitar el protocolo MQTT

para la transmisión de los datos hacia la plataforma Google cloud, donde se almacenara y visualizara los datos

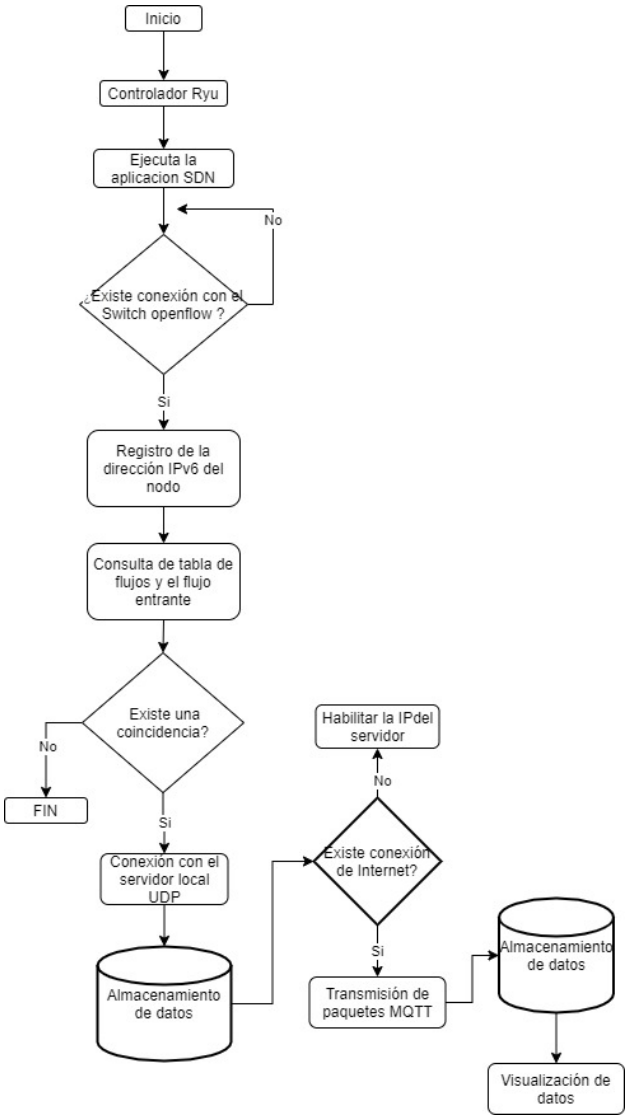


Figura 27. Diagrama de flujos del open vswitch y controlador Ryu

Diagrama de la base de datos

Para el almacenamiento de datos se considera un servidor local y uno en la nube, mismo que van a estar constituido por tres bases de datos que corresponden a los datos censados por cada uno de los nodos, cada tabla de la base de datos tendrá los siguientes parámetros: ID, fecha, IPv6 del nodo y dato del nodo, En la figura 28 se describe el proceso de almacenamiento de la información en los servidores

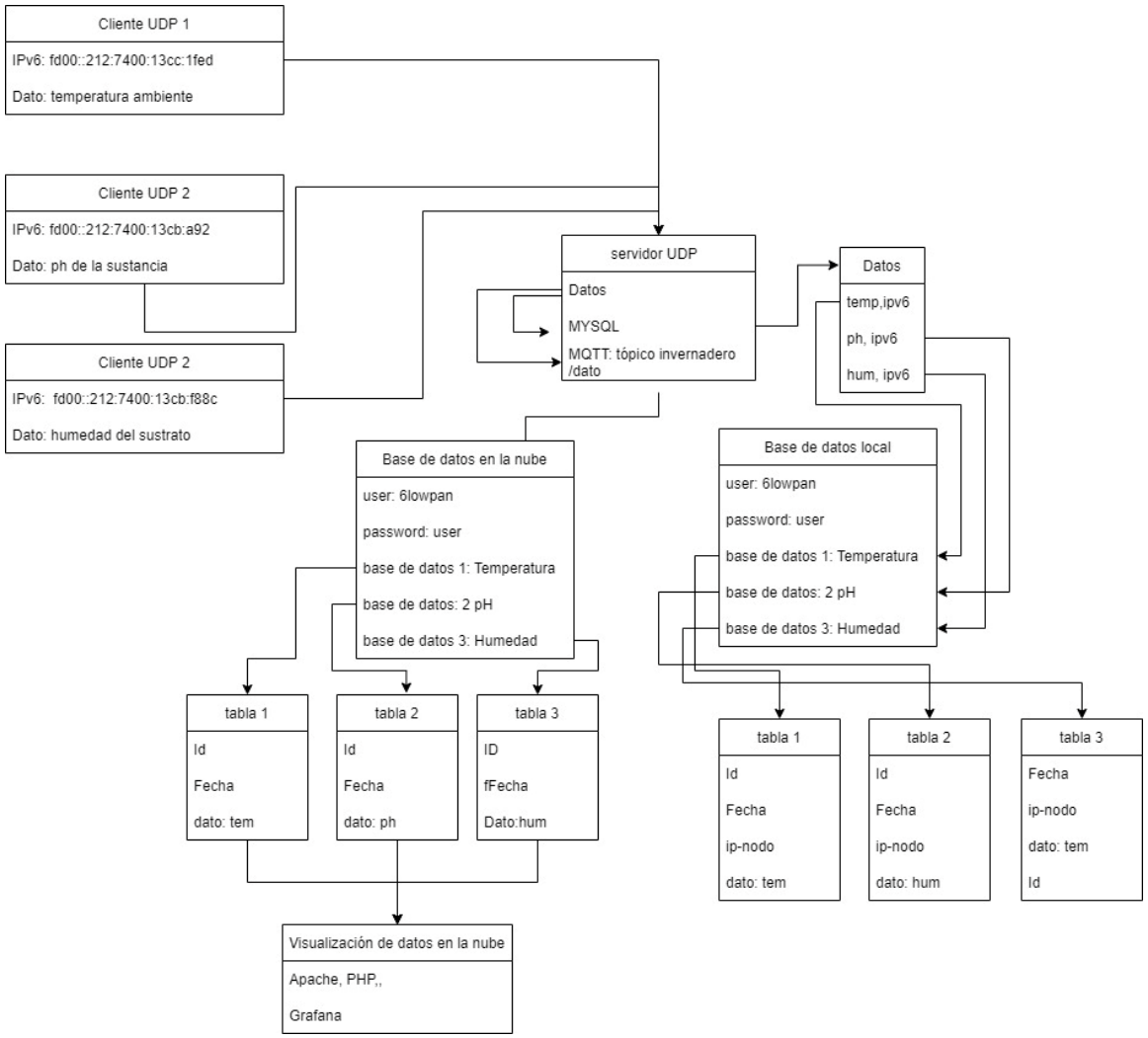


Figura 28. Diagrama de la base de datos

Diagrama de señalización

En la figura 29 se detalla el proceso de envió de los paquetes dentro del sistema, el cual consiste en establecer una comunicación cliente-servidor, es decir los nodos sensores envían mensajes de sincronismo para acceder a la red WSN por medio del nodo gateway, Al registrarse los nodos pueden enviar paquetes ICMPv6 y UDP hacia la red externa donde se halla el switch openflow.

Cuando los paquetes ICMPv6 y UDP llegan a los puertos del open vswitch son analizados de acuerdo con la tabla de flujos, Este análisis consiste en comparar cada paquete entrante con los siguientes campos: puerto de entrada del switch, dirección de origen IP,

dirección de destino IP, protocolo IP, puerto de origen TCP / UDP y puerto de destino TCP / UDP.

Si en el proceso de comparación no exista una coincidencia el switch open vSwitch envía mensaje packet-INT hacia al controlador Ryu anunciando una nueva entrada de flujo, y además cumple con la función de enviar detalladamente las instrucciones que debe realizar el switch para procesar los paquetes entrantes por medios de los mensajes packet-OUT.

Por último, si existe una coincidencia entre los paquetes entrantes y tablas de flujos se utiliza los mensajes packet-INT y packet-OUT, para procesar y enviar los paquetes hacia su puerto de destino en este caso, hacia el puerto que conecta el servidor UDP, de esta manera se establece una comunicación entre los nodos sensores y el servidor, el mismo procedimiento se repite para habilitar la comunicación entre el servidor UDP y Google cloud por medio del protocolo MQTT, donde se almacena y se publica los datos obtenidos por los sensores.

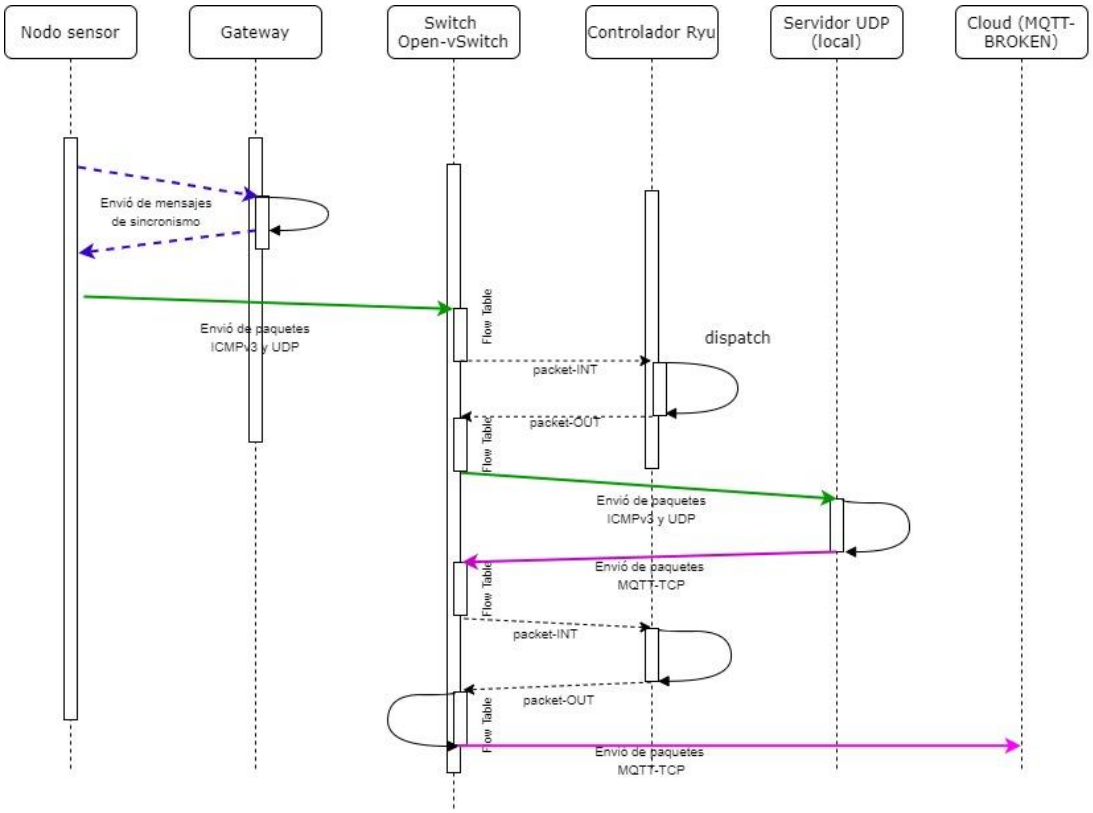


Figura 29. Diagrama de envió de información

Diagrama conexión del Sistema de arquitectura SD-WSN

En la figura 30 se muestra el diagrama de conexión de todo el sistema propuesto que incluye: dispositivos, protocolos de comunicación y las fuentes de alimentación. Para empezar, el nodo 1 integra una fuente de energía de 3 voltios a través de dos pilas, mientras que, el resto de los nodos su principal fuente de energía es una batería portátil de 9 voltios, mientras que, el gateway y switch openflow están conectados directamente a una fuente de voltaje de 5 voltios a 3.2 amperios. Además, cada nodo está asociado a un sensor para medir la humedad del sustrato, el pH de la sustancia nutritiva y la temperatura del ambiente, estos datos son encapsulados en una trama UDP para transmitirlos inalámbricamente por medio del estándar 802.15.4, este proceso es representado por la línea de color verde de la figura 26,

Por otro lado, el switch openflow es el mediador entre las dos tecnologías WSN y SDN que se encuentra equipado por dos interfaces USB-Ethernet que conectan el gateway y el servidor local, esta conexión es representada por la línea negra que al mismo tiempo refleja la ruta que a traviesa la trama UDP para llegar a su destino.

Por último, la línea amarilla indica la ruta de conexión entre el servidor UDP y la nube Google cloud a través del protocolo MQTT, los datos son enviados mediante los mecanismos de mensajerías o tópicos que deben coincidir en el bróker-MQTT para almacenar y visualizar los datos.

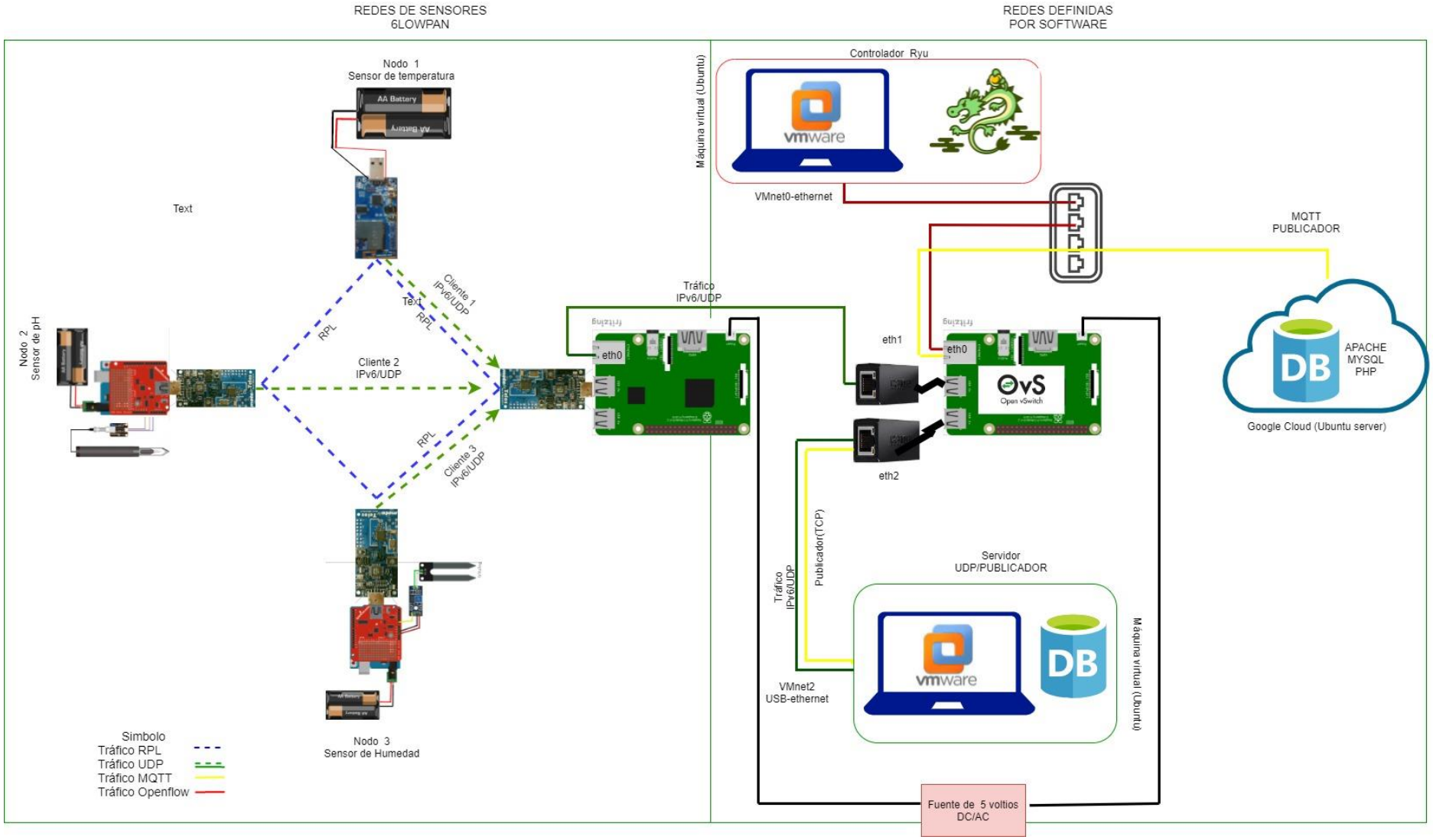


Figura 30. Diagrama de conexión del sistema

Direccionamiento y topología

Como se indica la figura 31, la red de sensores utiliza el protocolo RPL para establecer una topología en malla de forma que permita la comunicación entre nodos sensores, mientras que, para él envió de datos hacia el servidor se utiliza una comunicación cliente-servidor.

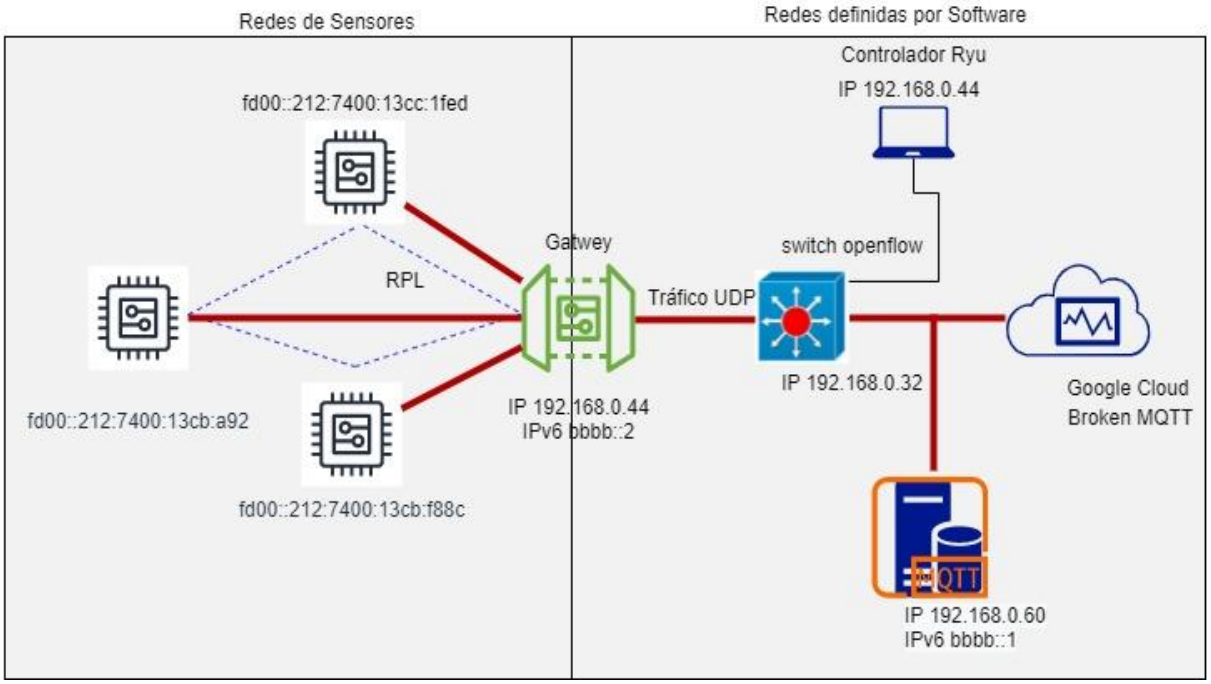


Figura 31. Representación gráfica de la topología de la red

De acuerdo con la figura 31 el sistema está formado por dos redes: redes definidas por software y redes sensores inalámbricos; el direccionamiento para los dispositivos de la red SDN está asignada por el siguiente pool de dirección IP 192.168.0.0 con una máscara 255.255.255.0. En la tabla 26 se muestran los dispositivos que forman parte de la red con sus respectivas direcciones IP

Tabla 26.
Direccionamiento para la red definida por software

Dispositivo	Direccionamiento IPv4	máscara	Direccionamiento IPv6	Prefijo
Controlador SDN	192.168.0.44	/24		
Switch Openflow	192.168.0.32	/24		
Gateway de la red WSN	192.168.0.45	/24	bbbb::2	/64
Servidor local	192.168.0.60	/24	Bbbb::1	/64

Fuente: Elaborado por el autor.

En cuanto al direccionamiento de la red WSN está definido por el protocolo 6lowpan que permite asignar a dispositivos de bajo recurso una dirección IPv6 y transmitirlo a través de un medio de comunicación inalámbrico basado en el estándar 802.15.4, el proceso de asignación del direccionamiento IPv6 se denomina autoconfiguración de direcciones sin estado, el cual consiste en dos parámetros: una dirección MAC propio del dispositivo y el identificador de PAN (ID-PAN), que es una dirección de 16 bits fijada aleatoriamente por el software que se ejecuta en el dispositivo. En la tabla 27, se muestra los dispositivos con sus respectivas direcciones IPv6 como también su dirección MAC, prefijos y el ID de la red.

Tabla 27.
Direccionamiento de la red WSN

Dispositivo	Dirección MAC	Dirección Link-Local	Prefijo de direccionamiento IPv6
Nodo 1	**00:12:74:00:13:cc:1f:ed**	**fe80::212:7400:13cc:1fed**	**fd00::212:7400:13cc:1fed**
Nodo 2	**00:12:74:00:13:cb:0a:92**	**fe80::212:7400:13cb:a92**	**fd00::212:7400:13cb:a92**
Nodo 3	**00:12:74:00:13:cb:f8:8c**	**fe80::212:7400:13cb:f88c**	**fd00::212:7400:13cb:f88c**
Servidor	**00:12:74:00:13:cc:01:70**	**fe80::212:7400:13cc:170**	**fd00::212:7400:13cc:170**

Fuente: Elaborado por el autor.

TÍTULO IV

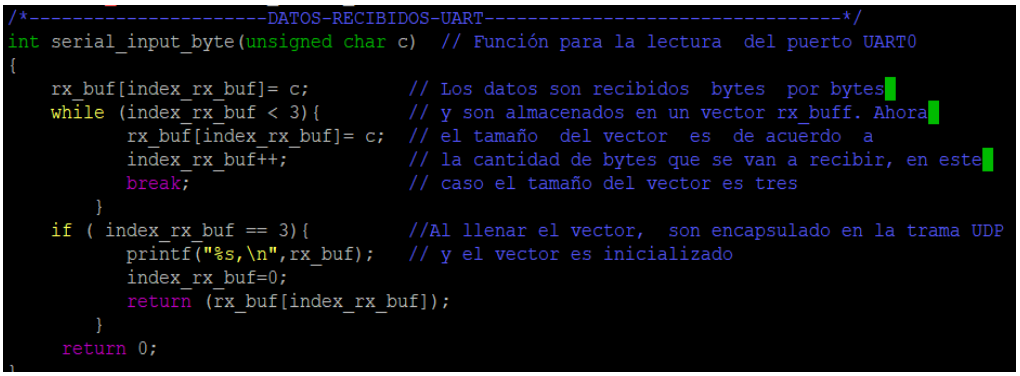
IMPLEMENTACIÓN Y PRUEBAS

Dentro de esta sección se reúnen los procesos de implementación y verificación del funcionamiento la red planteada. Es decir, este apartado abarca el procedimiento de instalación y configuración de cada uno de los dispositivos que forman parte de la red, como también, se describe el proceso de evaluación del funcionamiento de la red, en base a las métricas estipuladas en la RFC 2544.

4.1. Fase de implementación

Básicamente en esta sección consiste en describir las instrucciones que se llevó a cabo para la instalación y configuración de cada uno los dispositivos de la red tales como, nodos sensores, controlador Ryu, switch openflow y servidores, todo este proceso se puede observar detalladamente en el anexo B.

Ahora bien, para configurar una red de sensores se emplea el sistema operativo contiki-ng que propone un modelo de comunicación entre nodos, basado en el protocolo RPL-lite y UDP. Este modelo de comunicación se detalla en un archivo denominado udp-client.c, mismo que es modificado para habilitar la lectura del puerto UART0 del módulo tmote sky (telosB), de manera que permita recibir los datos censados por la placa arduino. En la figura 32 se indica la configuración para la adquisición de los datos provenientes de la placa arduino.



```
/*----------------------DATOS-RECIBIDOS-UART---------------------------------*/
int serial_input_byte(unsigned char c)  // Función para la lectura  del puerto UART0
{
    rx_buf[index_rx_buf]= c;          // Los datos son recibidos  bytes  por bytes
    while (index_rx_buf < 3){         // y son almacenados en un vector rx_buff. Ahora
          rx_buf[index_rx_buf]= c;    // el tamaño  del vector  es  de acuerdo  a
          index_rx_buf++;             // la cantidad de bytes que se van a recibir, en este
          break;                      // caso el tamaño del vector es tres
       }
    if ( index_rx_buf == 3){          //Al llenar el vector,  son encapsulado en la trama UDP
          printf("%s,\n",rx_buf);     // y el vector es inicializado
          index_rx_buf=0;
          return (rx_buf[index_rx_buf]);
       }
     return 0;
}
```

Figura 32. Configuración del puerto UART0 para a adquisición de datos

Con respecto a la transmisión de datos, consiste en registrar una conexión UDP por medio de la función *simple_udp_register*, que incluye dos parámetros: los puertos UDP del cliente y del servidor y la función *udp_rx_callback*, que cumple con la función de verificarla comunicación y obtener la dirección IPv6 del servidor. En la figura 33 se muestra la función simple_udp_register que permite iniciar la comunicación UDP.

```
/* Initialize UDP connection */
simple_udp_register(&udp_conn, UDP_CLIENT_PORT, NULL,
                    UDP_SERVER_PORT, udp_rx_callback);
```

Figura 33. Función que permite iniciar la comunicación UDP

Una vez que se registre la comunicación UDP, el siguiente paso es configurar la dirección IPv6 del servidor en cada uno de los nodos, en este caso es bbbb::1, de esta forma los datos se encapsulan y son enviados al servidor por medio de la función *simple_udp_sendto*. Mientras que, para obtener una red de sensores autoconfigurable y que se comuniquen entre sí, se hace uso del algoritmo de enrutamiento RPL-LITE que a través del método *NETSTACK_ROUTER* define la función *nodo_is_reachable*, que cumple con la capacidad de reparar la topología y determinar si un nodo está disponible en la red WSN. El proceso de encapsulamiento y la transmisión de los datos hacia el servidor se presenta en lenguaje de programación C++ en la figura 34.

```
simple_udp_register(&udp_conn, UDP_CLIENT_PORT, NULL,
                    UDP_SERVER_PORT, udp_rx_callback);
LibreOffice Writer
etimer_set(&periodic_timer, random_rand() % SEND_INTERVAL);
hum = (char*)rx_buf;
while(1) {

  PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&periodic_timer));
   if(NETSTACK_ROUTING.node_is_reachable()){
/*-------------------------------------------------------------------------*/
    uip_ip6addr(&dest_ipaddr, 0xbbbb,0,0,0,0,0,0,1);
    LOG_INFO("Enviando trama de sincronismo con el servidor %u to ", count);
    hum = (char*)rx_buf;
    LOG_INFO_("\n");
    LOG_INFO("humedad %s,\n",hum );
    LOG_INFO_6ADDR(&dest_ipaddr);
    LOG_INFO_("\n");
    snprintf(str, sizeof(str), "|%d|h|%s|", count,hum);
    simple_udp_sendto(&udp_conn, str, strlen(str), &dest_ipaddr);
    count++;
```

Figura 34. Proceso de envío de datos hacia el servidor

Por último, para corroborar con todo el proceso de configuración de la red de sensores se ejecuta el servidor UDP, con la finalidad de comprobar la recepción de los datos censados por los nodos. En la tabla 28 se indican los parámetros a ser monitoreados por cada uno de los nodos sensores

Tabla 28.
Representación del funcionamiento de los nodos sensores

Sensor	Mota	Pruebas
Temperatura		{'humsuelo': '242'} {'ph': '7.36'} {'temp': '43'} {'humsuelo': '242'} {'ph': '7.55'} {'temp': '43'} {'ph': '7.87'} {'humsuelo': '246'} {'temp': '43'} {'ph': '6.06'} {'humsuelo': '246'} {'temp': '43'} {'ph': '7.42'} {'humsuelo': '246'} {'ph': '7.50'} {'temp': '43'} {'humsuelo': '246'} {'temp': '43'} {'ph': '6.91'} {'humsuelo': '244'} {'ph': '6.21'} {'temp': '43'} {'humsuelo': '244'}
pH		
Humedad de suelo		

Fuente: Elaborado por el autor

Otra forma de comprobar la transmisión de los datos es realizar una captura de tráfico utilizando la herramienta wireshark donde se identifica los principales parámetros de la trama UDP como son puerto origen y destino, dirección IP del servidor y cliente, etc. En la figura 35 se indica la captura de tráfico UDP generado por los nodos sensores

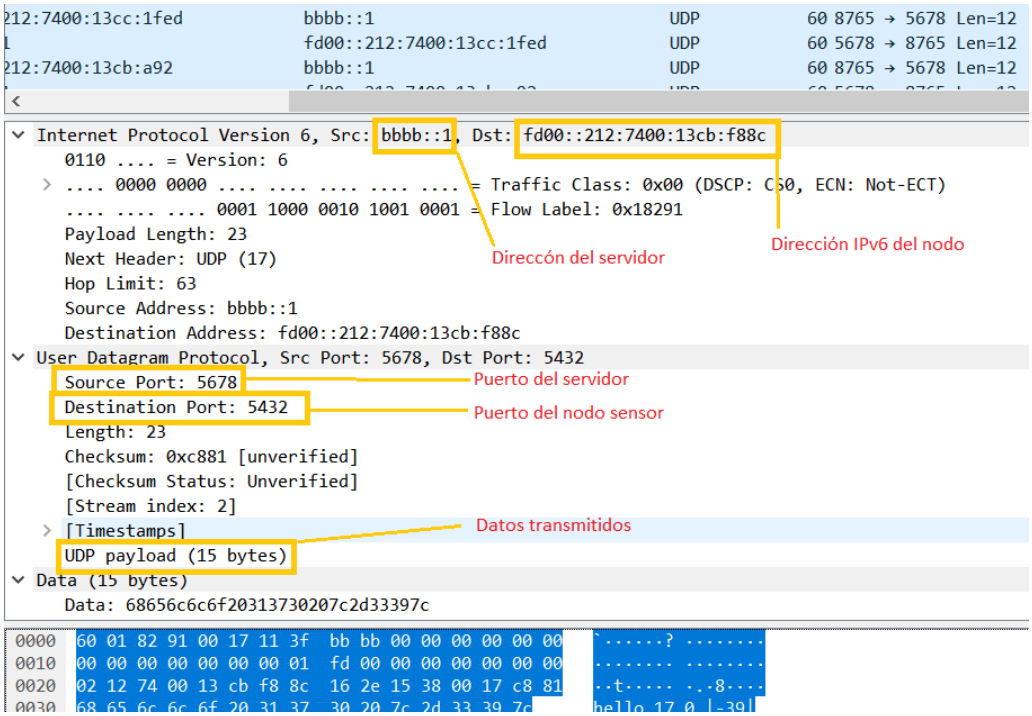


Figura 35. Captura del tráfico UDP de los sensores

Instalación y configuración de controlador Ryu y open vswitch

Antes de empezar con la instalación del open vswitch en una placa raspberry pi se requiere la instalación de las librerías de Python y el módulo de encabezado de kernel que son necesario, para que, el funcionamiento del open vswitch sea el correcto. La figura 36 indica la versión instalada del open vswitch como también la instalación de los principales módulos que definen el funcionamiento de raspberry pi como un switch openflow como son: osv-vswitchd y ovsdb-server.

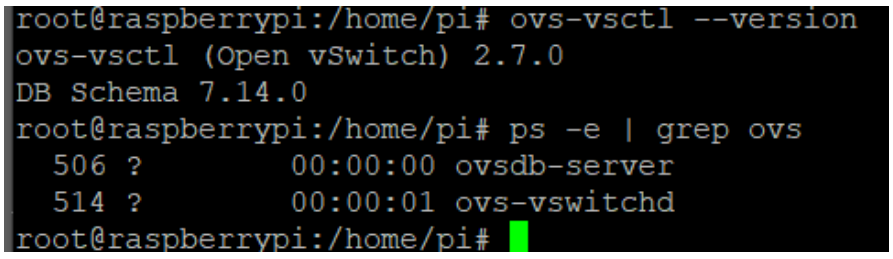


Figura 36. Verificación de la instalación de open vswitch

El siguiente paso es la configuración del switch openflow, que consiste en crear un puente entre el open vswitch y el raspberry pi por medio de la creación de una interfaz virtual denominada

bridge, que es asociado a la interfaz ethernet del raspberry pi, en este caso en la interfaz eth0, con el objetivo de que todo el tráfico que ingresa por las interfaces USB-Ethernet sean direccionadas hacia el open vswitch. Además, para poder agregar las interfaces USB-Ethernet como puertos del switch openflow, estas interfaces deben tener una dirección IP nula.

Es fundamental mencionar que el switch requiere del controlador ryu para establecer un control en los flujos de datos, por tal razón se configura un canal de comunicación TCP/IP a través del puerto 6633 donde se ejecuta el protocolo openflow en su versión 1.3. En figura 37 se resume las principales configuraciones que se realizaron en switch openflow.

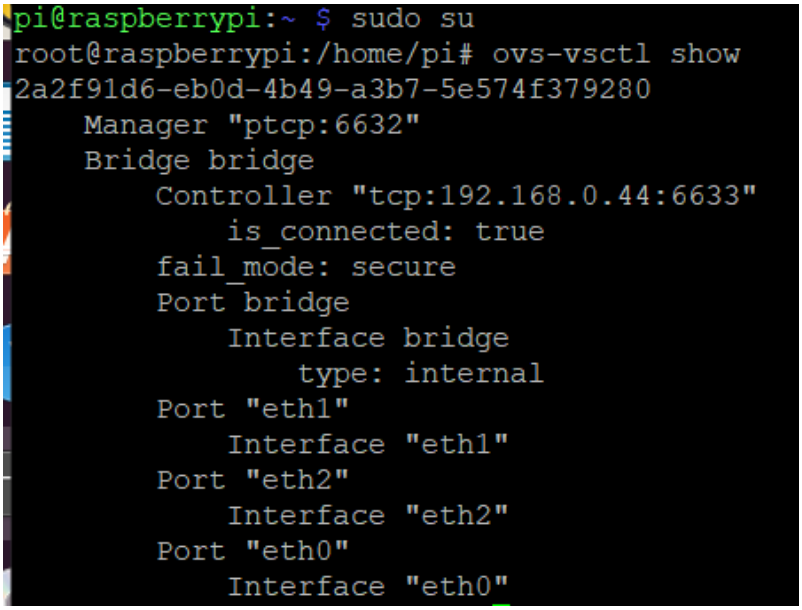


Figura 37. Configuración del switch openflow

Mientras que, para la configuración del controlador Ryu requiere la instalación de las librerías de python, para luego, instalar el repositorio del controlador ryu, al finalizar el proceso de instalación se debe asignar una dirección IP al adaptador ethernet de sistema operativos Ubuntu de acuerdo con la tabla 26, de esta manera, el controlador intercambia mensajes de control con el switch openflow cada vez que se ejecute una aplicación.

Antes de ejecutar la aplicación SDN es necesario verificar que el switch openflow tenga desactivado el modo de pruebas de fallas, de esta forma se garantice que al iniciar no existe ninguna regla de flujos de datos preinstalada en el switch openflow.

Como se mencionó anteriormente la comunicación entre el controlador ryu y switch openflow consiste en el intercambio de ciertos mensajes como: *hello, feature, Multipat y Flow-modify* que cumple las siguientes funciones:

- **Hello:** estos mensajes son enviados por el switch openflow y controlador ryu con la finalidad de establecer una conexión entre los dos.
- **Feature:** este tipo de mensaje es enviado por el switch openflow al controlador para establecer una comunicación por medio del protocolo TLS (*Transport Layer Security*)
- **Multipart:** este tipo de mensajes son ejecutados por el controlador para solicitar información sobre el funcionamiento del conmutador

Cada uno de estos mensajes se pueden evidenciar al realizar una captura de tráfico en el controlador Ryu por ejemplo en la figura 38, se muestra los mensajes *HELLO* emitidos por los dispositivos, además se puede apreciar la dirección IP de los dispositivos y la versión del protocolo openflow.

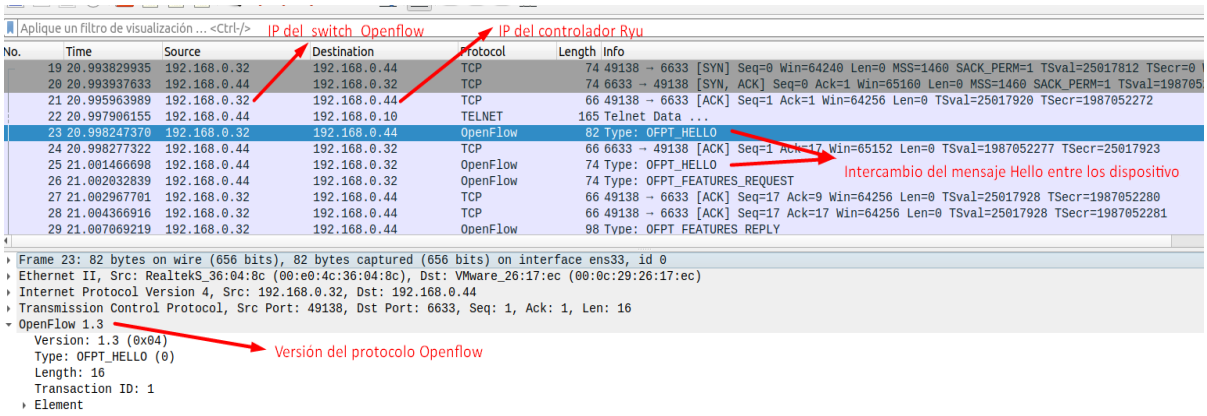


Figura 38. Comunicación entre el switch openflow y controlador por medio de los mensajes hello

Una vez establecida la comunicación entre los dos dispositivos el siguiente paso es establecer una conexión segura por medio del protocolo TLS (Transport Layer Security), para lo cual, se envía los mensajes de tipo FEATURES_REQUES y FEATURES_REPLY, mismos que se evidencia en la figura 39.

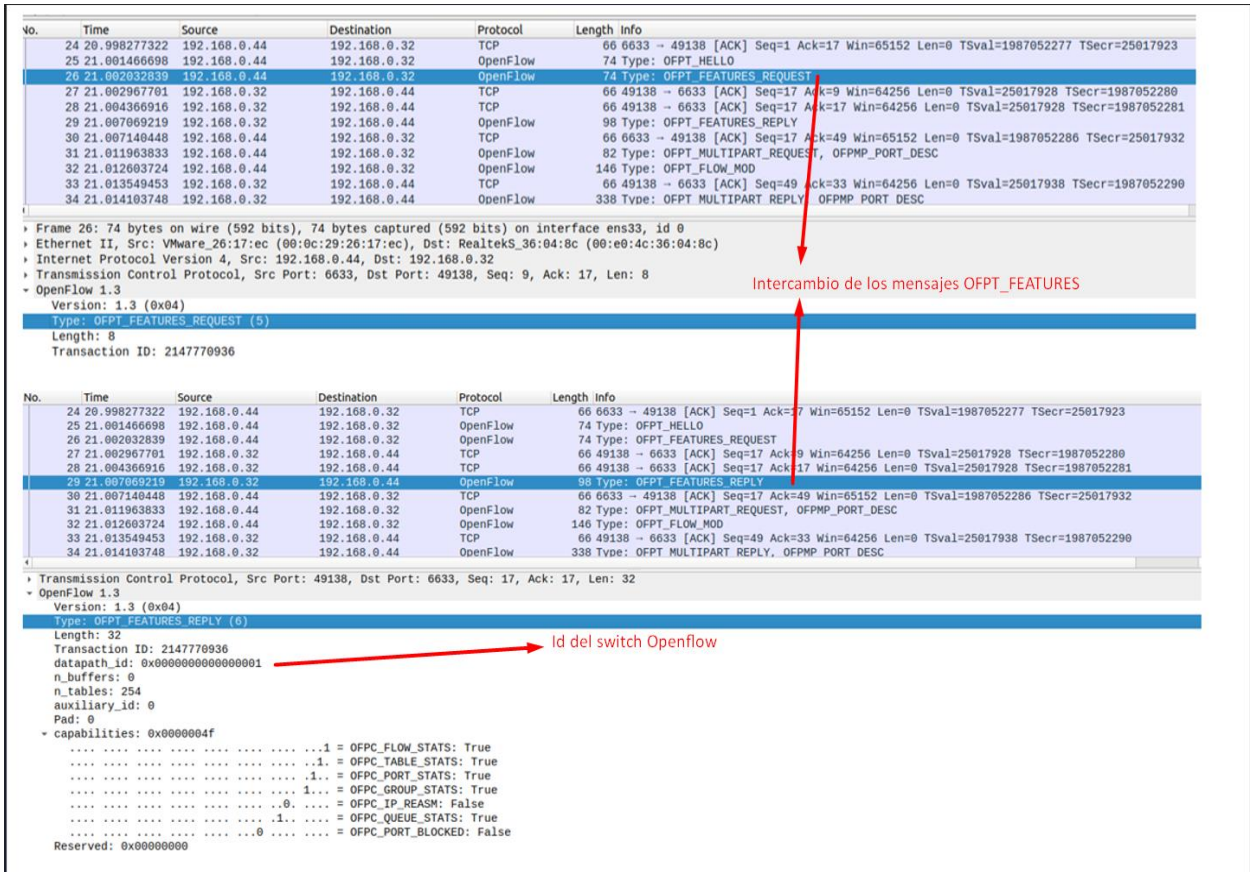


Figura 39. Captura de los mensajes FEATURES REQUEST y REPLAY

Por último, el controlador utiliza el mensaje MULTIPART_REQUEST para solicitar al switch openflow información detallada de cada uno de sus puertos, este responde por medio del mensaje MULTIPART_REPLAY el cual, contienen información como, el nombre del puerto, direcciones MAC y estadísticas del flujo que atraviesan por el puerto. En la figura 40 se puede observar las características de los puertos del switch openflow.

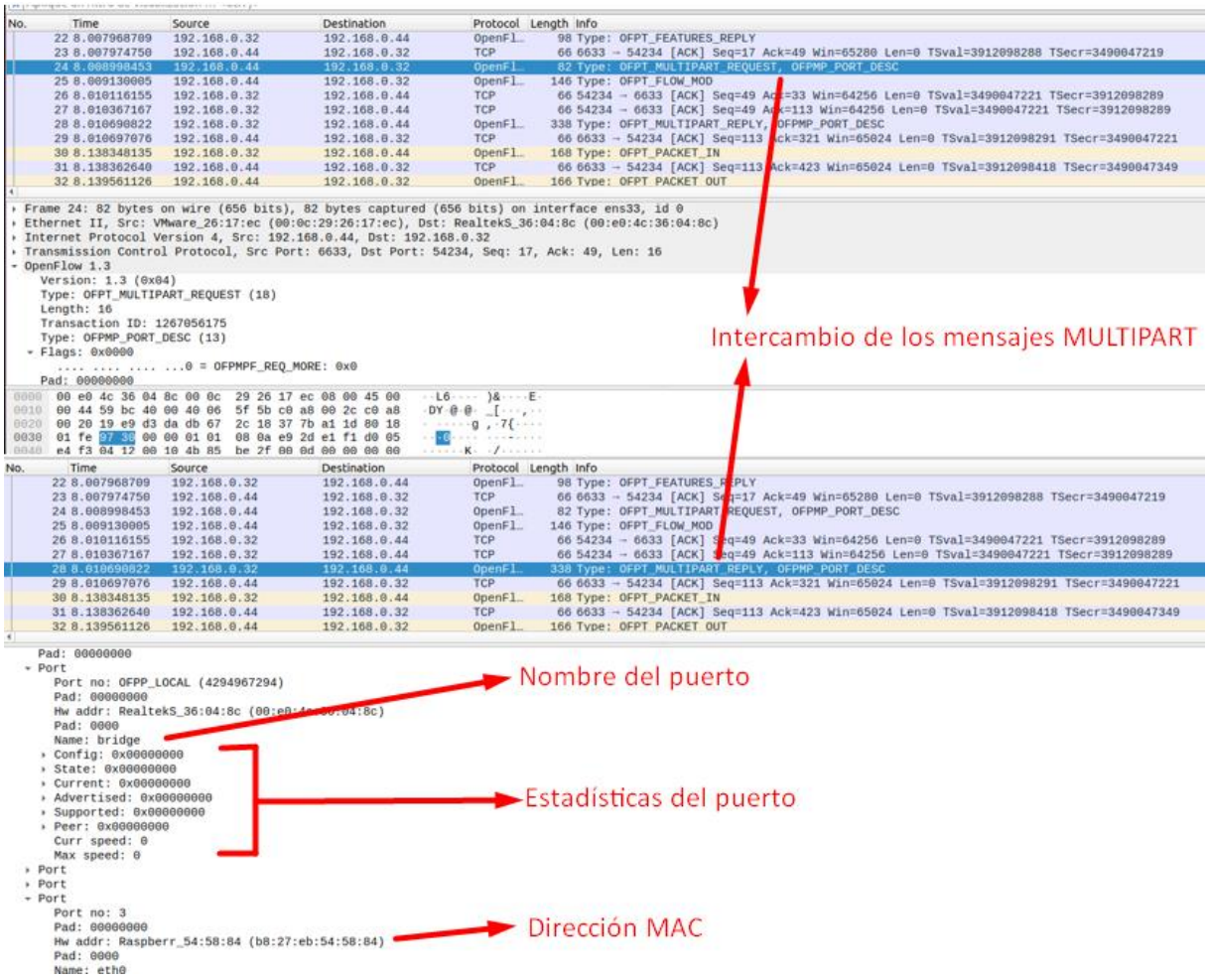


Figura 40. Captura de los mensajes MULTIPART

Implementación de la aplicación SDN

Existen una variedad de aplicaciones SDN que se pueden aplicar en una red WSN para mejorar sus funcionalidades, entre dichas aplicaciones se encuentra el firewall que permite relacionar los nodos sensores en una infraestructura SDN por medio de las direcciones IP, de tal manera que se pueda controlar el flujo de datos y el acceso de los dispositivos de los sensores.

Esta aplicación denominada firewall se halla disponible en el repositorio del controlador ryu y hace uso de los principales métodos del protocolo http (Hypertext Transfer Protocol) como son: get, post y delete, estos, permiten insertar o eliminar flujo de datos dentro del switch openflow.

Ahora bien, la estructura para insertar una regla de flujo radica en, ejecuta el comando *curls* seguido de las peticiones http, que indican el tipo de acción que se va a realizar en el controlador ryu en referencia a los campos: Id switch, prioridad, dirección de IP de destino y origen, protocolo y acción. En la figura 41, se muestra las principales reglas de flujo que se implementaron para ejecutar la aplicación SDN y habilitar el flujo de datos de los nodos sensores.

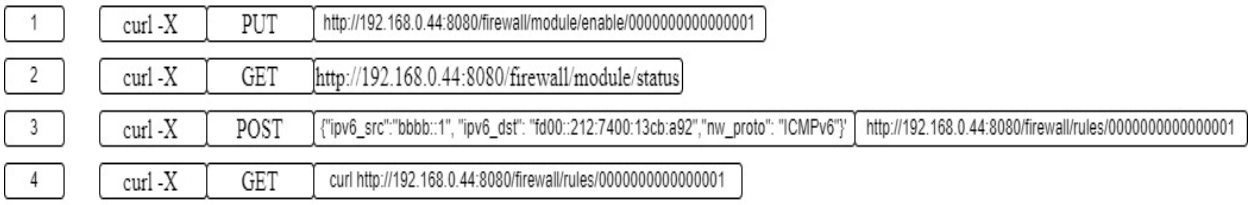


Figura 41. Reglas de flujo utilizadas por el controlador Ryu

Por otro lado, para ejecutar la aplicación, se debe insertar los siguientes comandos *ryu-manager* y *ryu.app.rest_firewall*, mismos que permiten iniciar la comunicación entre el controlador y el switch, En la figura 42, se indica el proceso de ejecución de la aplicación.

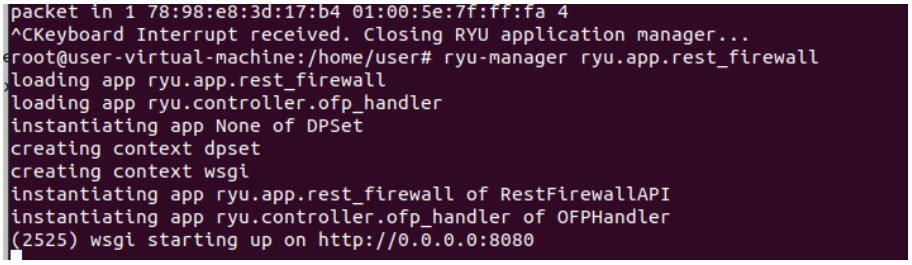


Figura 42. Ejecución de la aplicación firewall

A continuación, para verificar el funcionamiento de la aplicación es necesario ejecutar los comandos 1 y 2 que se encuentran detallados en la figura 40, estos permiten comprobar, si la aplicación se ejecuta correctamente como también permite que el controlador escuche el flujo de datos que ingresan por los puertos del switch, estos flujos de datos no son procesados debido a que no existe ninguna instrucción insertada en la tabla de datos del switch, este proceso se evidencia en la figura 43.

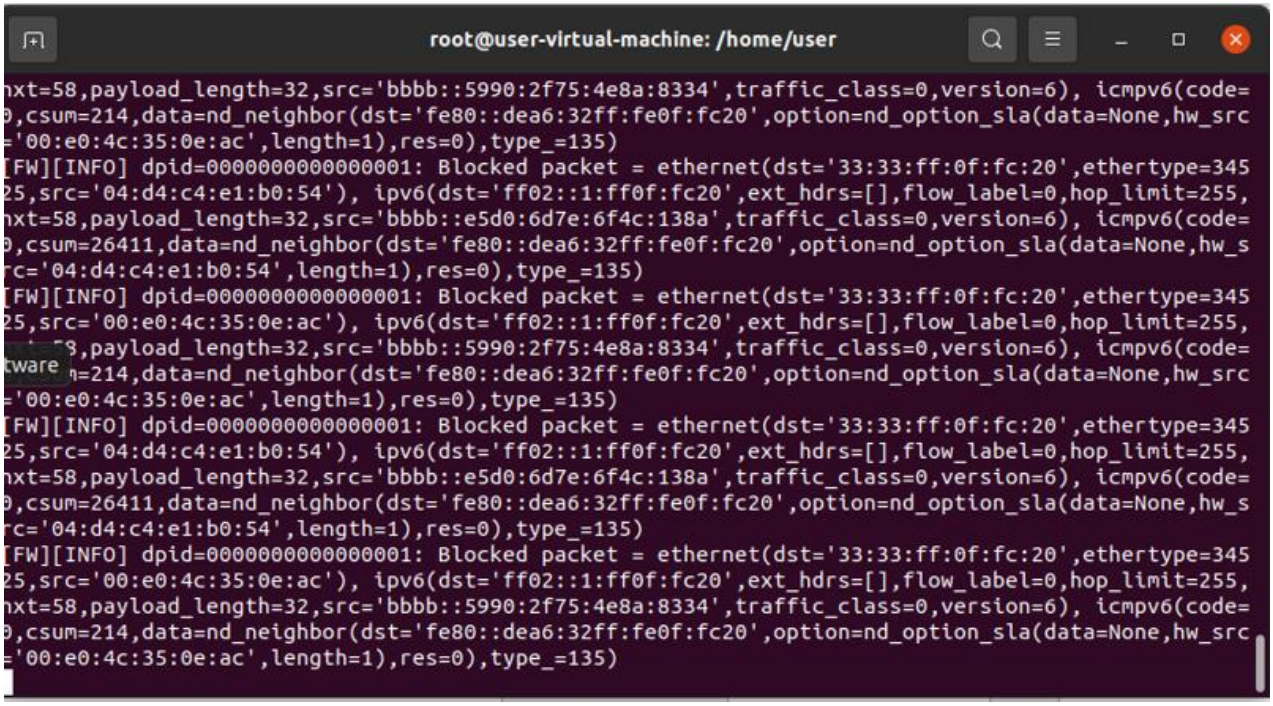


Figura 43. Tráfico bloqueado por el open vswitch

Implementación de los servidores

La instalación del servidor UDP se llevó a cabo en una máquina virtual, el cual permite el procesamiento de la información de los nodos sensores, es decir una vez que los datos lleguen al servidor son almacenados en un sistema de gestión de datos denominado MySQL. Este sistema tiene configurado tres bases de datos, cuyas tablas contiene la información de cada uno de los nodos como, por ejemplo, la dirección IPv6 y los datos censados por los sensores. En la figura 44 se puede visualizar la configuración de la base de dato para el proyecto.

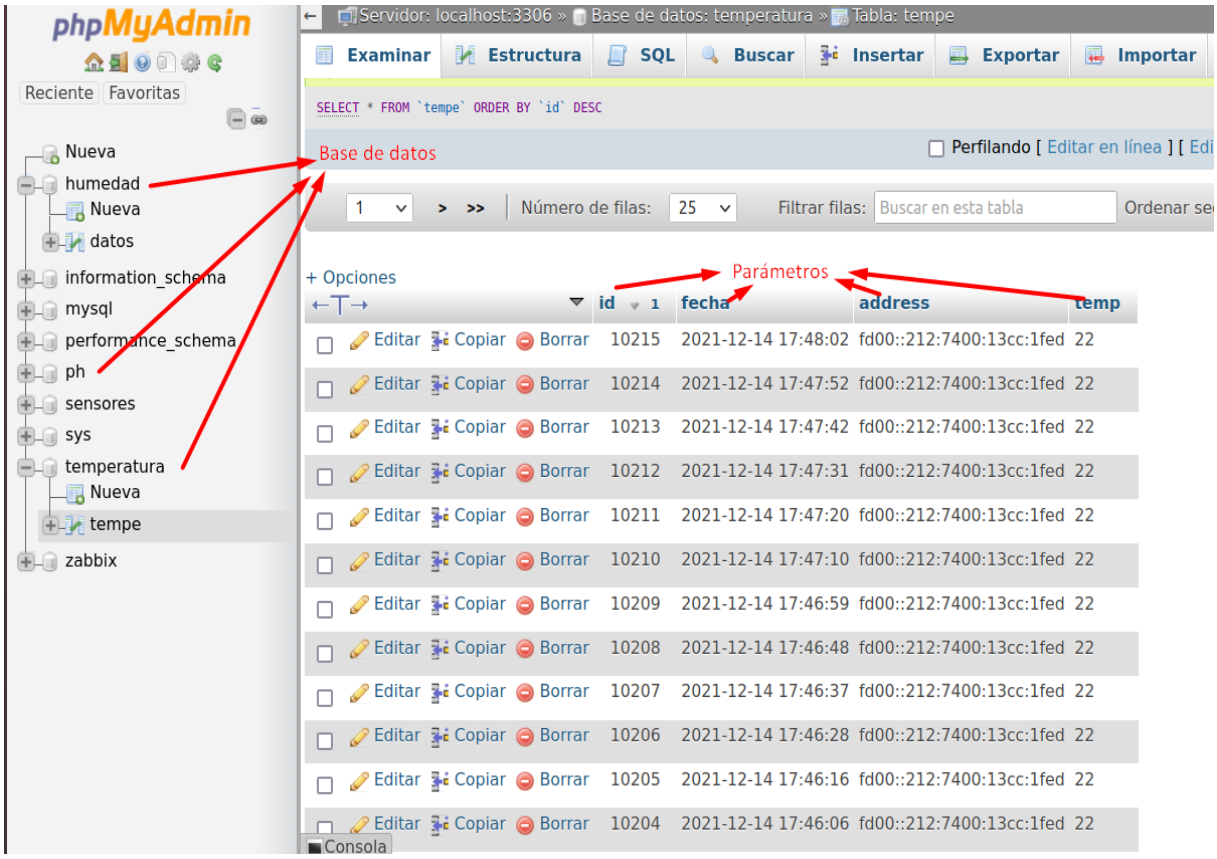


Figura 44. . Interfaz de la base datos para los sensores

Esta misma configuración se lo realiza en la nube Google cloud quien es encargado de establecer una comunicación directa con el servidor UDP por medio del protocolo MQTT que requiere de tres parámetros, la dirección IP de la plataforma Google cloud, habilitar el puerto 1883 y finalmente, se debe especifica un tópico que tiene la función de filtrar mensajes para cada el cliente. En la figura 45 se muestra la configuración realizada en el servidor UDP.

Figura 45. Parámetros para iniciar una comunicación por medio del protocolo MQTT

Ahora, para que el flujo de datos llegue a la plataforma Google por medio del protocolo MQTT, se debe crear una regla de firewall que habilite el tráfico TCP y el puerto 1883 este proceso se puede evidencia en la figura 46.

☐	grafana	Entrada	Aplicar a todi	Intervalos de	tcp:3000	Permitir	1000	default	Desactivado
☐	mqtt	Entrada	Aplicar a todi	Intervalos de	tcp:1883	Permitir	1000	default	Desactivado
☐	mysql	Entrada	Aplicar a todi	Intervalos de	tcp:3306	Permitir	1000	default	Desactivado
☐	todo	Entrada	Aplicar a todi	Intervalos de	tcp:22	Permitir	1000	default	Desactivado

Habilitar el puerto 1883 para la recepción de dato

Figura 46. Configuración del firewall de la plataforma google cloud

Por último, se instala el paquete XAMPP que permite desarrollar una interfaz web con el propósito de visualizar los parámetros de humedad del suelo, temperatura del invernadero y los niveles de pH de la sustancia nutritiva. En la figura 47 se muestra el diseño de la página web. Sin embargo, como se mencionó en la fase de diseño se hizo uso del software grafana ya que, permite crear registros y visualizar los datos censados de una manera sencilla como se puede apreciar en la figura 48.

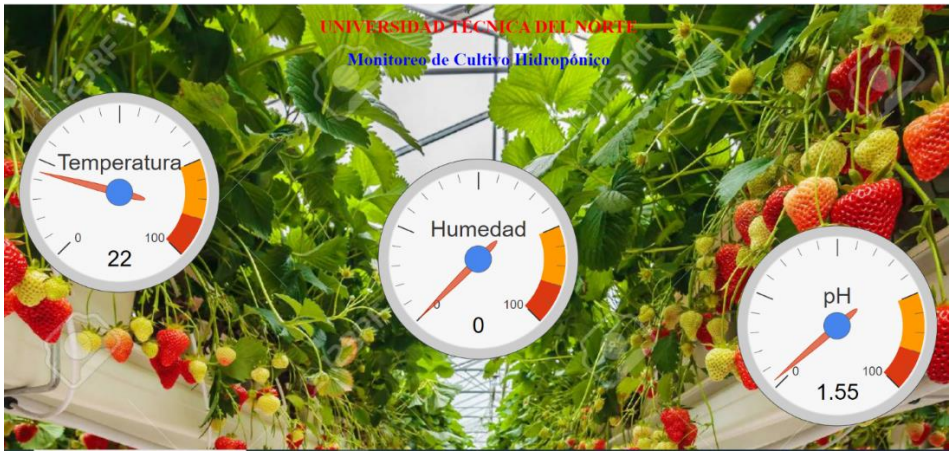


Figura 47. visualización de datos en la nube

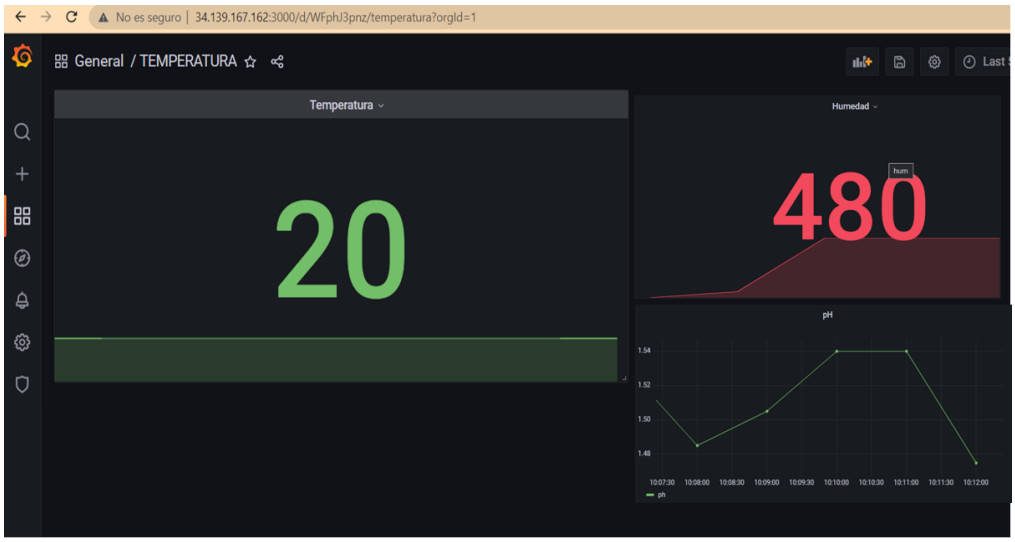


Figura 48. Visualización de los datos en el software grafana

4.2. Fase de Pruebas

Con respecto al proceso de evaluación de la red se plantea dos escenarios. El primer escenario consta de sensores que se conectan al nodo gateway para enviar información hacia el servidor local en cambio, en el segundo escenario se introducen el switch openflow y el controlador ryu, que juntos permiten controlar el tráfico de los flujos de datos y el acceso a la red a cada uno de los nodos sensores.

Para evaluar cada uno de los escenarios se considera las sigueintes métricas, latencia y la taza de pérdida de paquetes, los cuales fueron obtenidos al ejecutar el comando ping que utiliza el protocolo ICMPv6, que envía mensajes ECHO_REQUEST y ECHO_RESPONSE con diferentes tamaños de carga útil. En el siguiente ítem se describe de forma resumida cada una de las métricas a utilizar.

- Latencia: permite evaluar el tiempo total que demora un paquete en viajar desde el servidor hacia al nodo origen, este proceso se realiza por medio del comando ping.
- Taza de pérdidas de paquete: permite conocer la capacidad de procesamiento de los paquetes dentro de una red. El valor de la taza de pérdidas de paquetes resulta de la relación entre los mensajes trasmitidos menos los paquetes recibidos sobre los mensajes transmitidos.

Dentro de las pruebas se considera que la unidad máxima de transferencia (MTU) de las redes 6lowpan es de 127 bytes, lo que significa que el máximo tamaño de carga útil que se puede enviar a través del comando ping es de 84 bytes. Cabe aclarar que, el paquete ICMPv6 para múltiples saltos tiene un tamaño de 56 bytes de dedicados a los campos de capa física, MAC y encabezado IPv6, dejando solo 72 bytes para transmisión de datos.

Tomando en cuenta el análisis anterior, en la tabla 29, se detalla el rango de tamaño de carga útil que admite los nodos sensores, el tiempo de ejecución de la prueba y la distancia entre el sensor y el gateway, estos parámetros permitirán determinar el rendimiento de cada uno de los escenarios.

Tabla 29.
Parámetros iniciales

Parámetros	
Carga útil	20, 40, 60,72, 84 (bytes)
Tiempo de ejecución de la prueba	60s
Distancia entre nodo central y nodos sensores	3,15 m
Nodo 1	fd00::212:7400:13cb:a92
Nodo 2	fd00::212:7400:13cb:f88c
Nodo 3	fd00::212:7400:13cc:1fed

Fuente: Elaborado por el autor

4.2.1. Primer escenario

Para iniciar con la evaluación del primer escenario se verifica la comunicación entre el servidor y los nodos, utilizando el software wireshark que nos permite visualizar el proceso del ping, que inicia desde el servidor enviando un mensaje ECHO REQUEST hacia la red para determinar la disponibilidad del nodo, en el caso que nodo de destino este habilitado dentro de la red responde con un mensaje ECHO REPLAY. En la figura 49 se muestra el proceso de comunicación entre dos nodos con el servidor local por medio de los mensajes ECHO REQUEST y REPLAY.

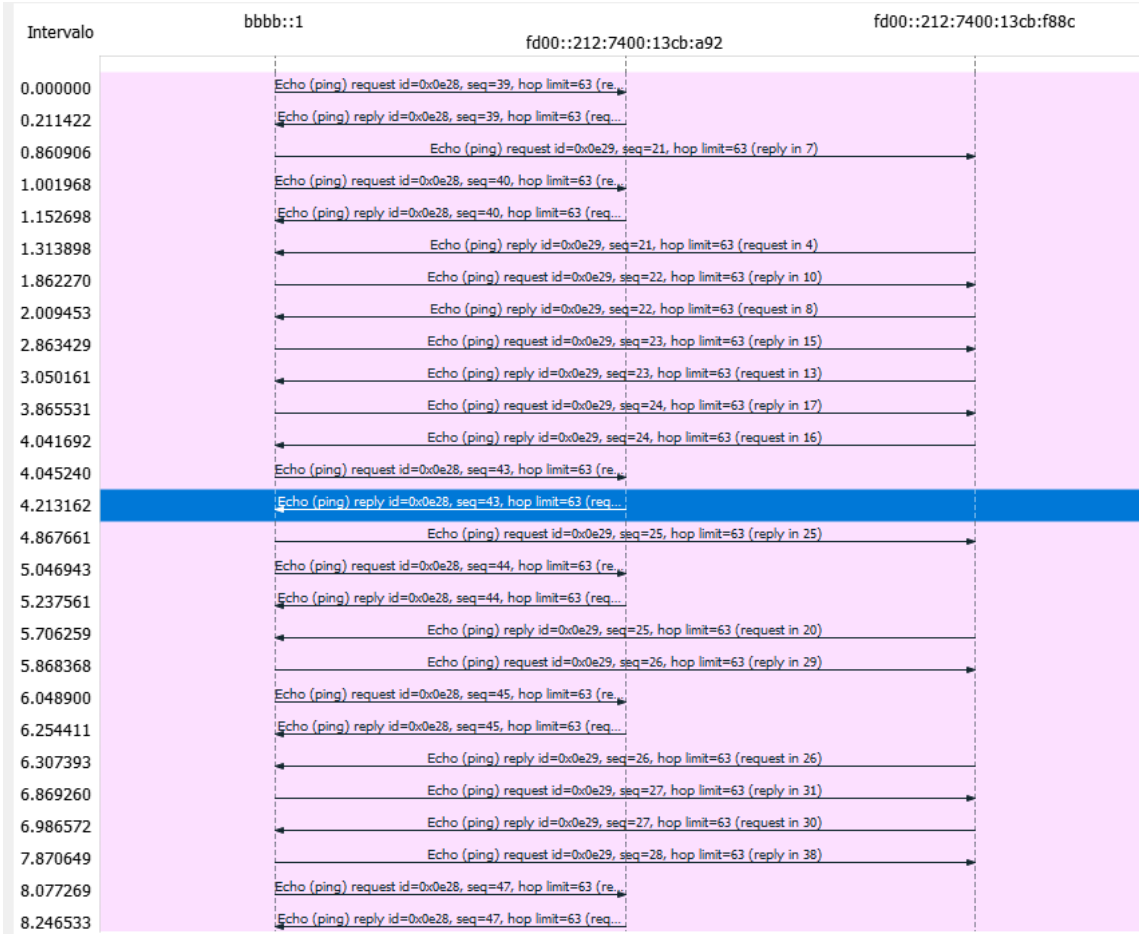


Figura 49. Diagrama de secuencia del comando ping entre el nodo y servidor

4.2.2. Segundo escenario

El segundo escenario consiste en verificar el comportamiento de la red, incluyendo la funcionalidad de la aplicación SDN, que consiste en insertar reglas de flujo para habilitar los protocolos ICMPv3 y UDP, de esta forma, permite establecer una conexión entre nodo y el servidor. Es importante mencionar que, al momento de ejecutar la aplicación por primera vez, no se va, a procesar ningún paquete ya sea ICMPv6 o UDP, debido a que no se encuentra ninguna regla de flujo instalado en el switch openflow como se visualiza en la figura 50.

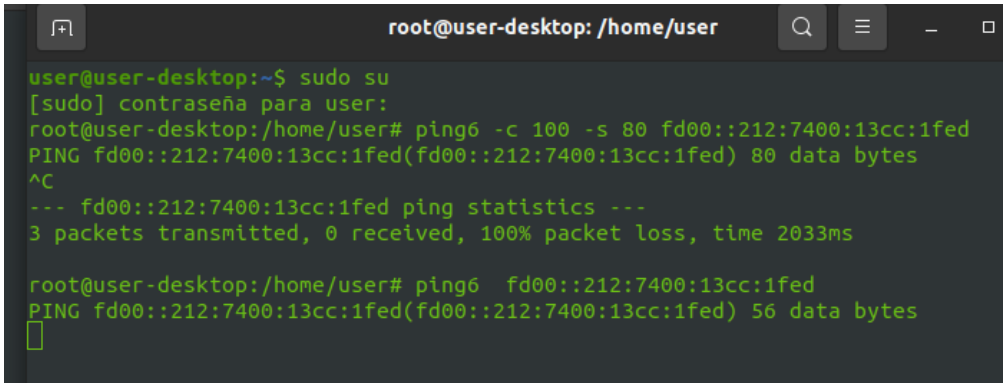


Figura 50. Comprobación de la comunicación entre nodo y servidor

Ahora, con el fin de verificar la disponibilidad del nodo sensor dentro de la red, se habilita el tráfico ICMPv6, por medio de las reglas que se describen en la figura 51, los parámetros más significativos de la regla son la dirección IP del nodo y el tipo de protocolo que se va a, habilitar si la regla fue insertada correctamente los mensajes ICMPv6 pueden ser procesados como se muestran en la figura 52.

```
curl -X POST -d '{"ipv6_src": "bbbb::1", "ipv6_dst": "fd00::212:7400:13cc:1fed",
 "nw_proto": "ICMPv6"}' http://localhost:8080/firewall/rules/0000000000000001
curl -X POST -d '{"ipv6_src": "fd00::212:7400:13cc:1fed", "ipv6_dst": "bbbb::1",
 "nw_proto": "ICMPv6"}' http://localhost:8080/firewall/rules/0000000000000001

curl -X POST -d '{"ipv6_src": "fd00::212:7400:13cc:1fed", "ipv6_dst": "ff02::1:f
f00:2", "nw_proto": "ICMPv6"}' http://localhost:8080/firewall/rules/000000000000
0001
curl -X POST -d '{"ipv6_src": "bbbb::1", "ipv6_dst": "ff02::1:ff00:1", "nw_proto
": "ICMPv6"}' http://localhost:8080/firewall/rules/0000000000000001
```

Figura 51. Reglas insertadas para habilitar tráfico ICMPv6

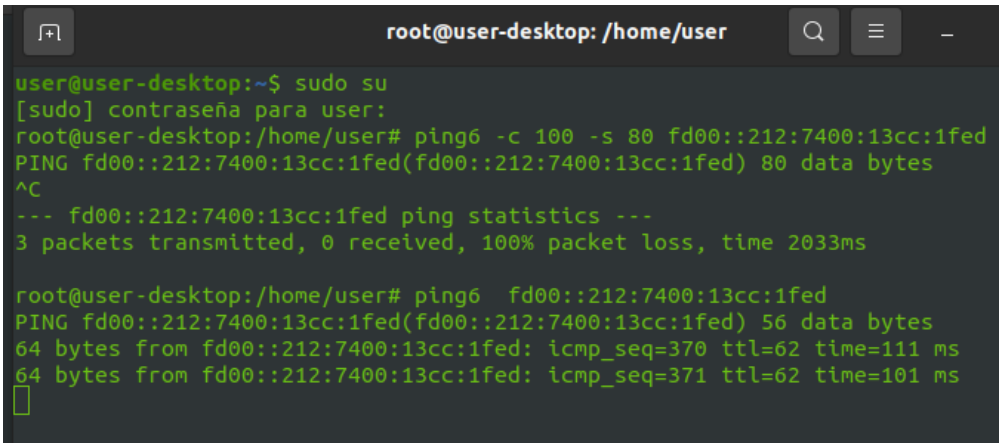


Figura 52. Flujo ICMPv6 habilitado

El mismo procedimiento descrito anteriormente, se repite para habilitar el tráfico UDP de cada uno de los nodos con el propósito de que la información de los nodos llegue al servidor UDP. En la figura 53 se describe las reglas necesarias para que el nodo 1 envie los datos censados de la humedad del sustrato hacia el servidor.

Figura 53. Reglas para habilitar el tráfico UDP del nodo 1

Una vez, insertada la regla de flujo en el switch openflow se puede visualizar los datos del nodo en el servidor UDP, como se indica en la figura 54. Estos datos son almacenado en el servidor local y reenviados hacia la plataforma Google a través del protocolo MQTT.

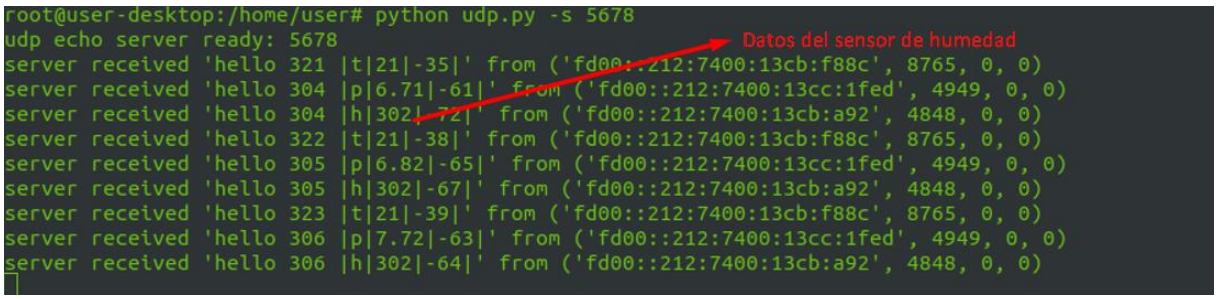


Figura 54. Datos recibidos por el servidor

Para verificar las reglas instaladas en firewall del controlador Ryu se ejecuta el comando curl http://localhost:8080/firewall/rules/0000000000000001 donde se puede observar el identificador de open vswitch como también el número de reglas que se divide en varios campos los importantes son las acciones que permiten habilitar el tipo de tráfico como se indica en la figura 55

Figura 55. Reglas insertadas en controlador SDN

Otra forma de verificar el flujo de datos es mediante el switch openflow, que registra el flujo entrante en su base de datos. La figura 56 indica las estadísticas del flujo de datos que a traviesan switch los parámetros que se destacan las direcciones IP de origen y destino como también el número de paquetes que ha procesado el switch.

Figura 56. Verificación de las reglas insertadas en el openvswitch

Para finalizar se verifica la comunicación entre el servidor local y la plataforma Google cloud que se comunican con el protocolo MQTT para esto, utilizamos la herramienta wireshark para capturar el tráfico de datos en la red. La figura 57 indica la dirección IP del servidor instalado en Google cloud

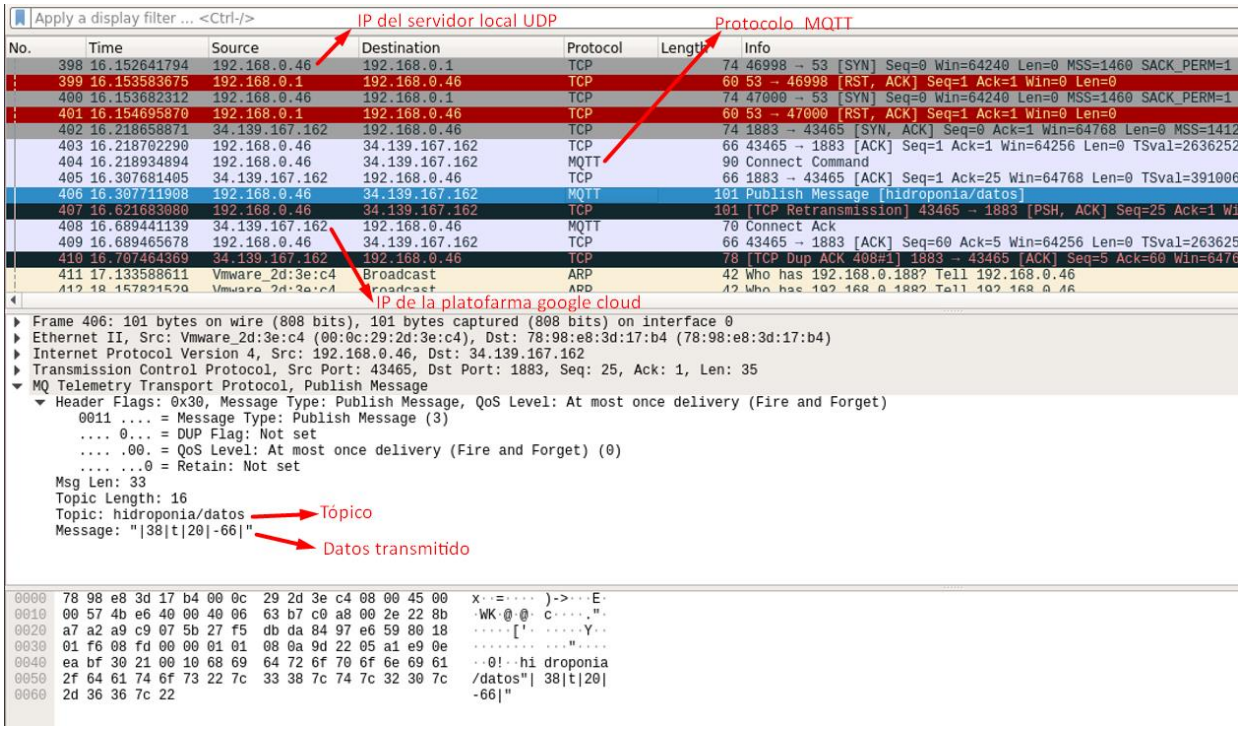


Figura 57. Captura del tráfico MQTT

4.3. Resultados

En esta sección se plantea evaluar la capacidad entrega de los datos de los dos escenarios que consistió enviar diferentes paquetes entre el servidor y los nodos sensores. En la tabla 30 se muestran los resultados que se obtuvieron en este proceso.

Tabla 30.
Resultados obtenidos de la taza de pérdida de paquetes de cada uno de los nodos sensores

Tamaño de Trama (bytes)	**Paquetes transmitidos**	**Taza de pérdidas de paquetes (%)**					
		Nodo 1		Nodo 2		Nodo 3	
		SDN	SIN SDN	SDN	SIN SDN	SDN	SIN SDN

20	20	0	5	0	5	0	0
20	40	0	0	0	0	0	0
20	60	0	0	0	1,667	0	0
20	80	0	0	0	0	0	0
20	100	0	1	0	2	0	0
40	20	0	0	0	0	0	0
40	40	0	0	0	0	0	0
40	60	0	0	0	1,667	0	3,333
40	80	0	0	0	1,25	0	0
40	100	0	0	0	0	0	3
60	20	0	0	0	0	0	5
60	40	0	2,5	2,5	2,5	0	5
60	60	0	1,667	0	0	0	0
60	80	0	2,5	0	0	0	1,25
60	100	0	0	0	0	3	1
72	20	0	5	0	0	0	0
72	40	0	0	0	7,5	0	5
72	60	0	3,333	0	10	0	0
72	80	0	1,25	0	2,5	5	2,5
72	100	2	10	0	1	0	0
84	20	30	40	25	30	25	35
84	40	42,5	30	10	20	27,5	30
84	60	26,667	5	18,333	31,667	28,33	30
84	80	26,25	18,75	26,25	28,75	13,75	26,25
84	100	29	24	18	33	30	34

Fuente: Elaborado por el autor.

En la figura 58 se compara los resultados obtenidos en relación con taza de pérdida de paquetes de los dos escenarios, donde se demuestra que en ambos escenarios la taza de pérdidas es aceptable al no superar el 10 %. Por otro lado, se puede observar que la pérdida de paquetes en la red SDN es menor cuando se envía una trama con una carga útil de 20 a 72 bytes, mientras que, en la red sin SDN el menor número de paquetes perdidos es cuando se envía una trama con una carga útil de 20 a 40 bytes.

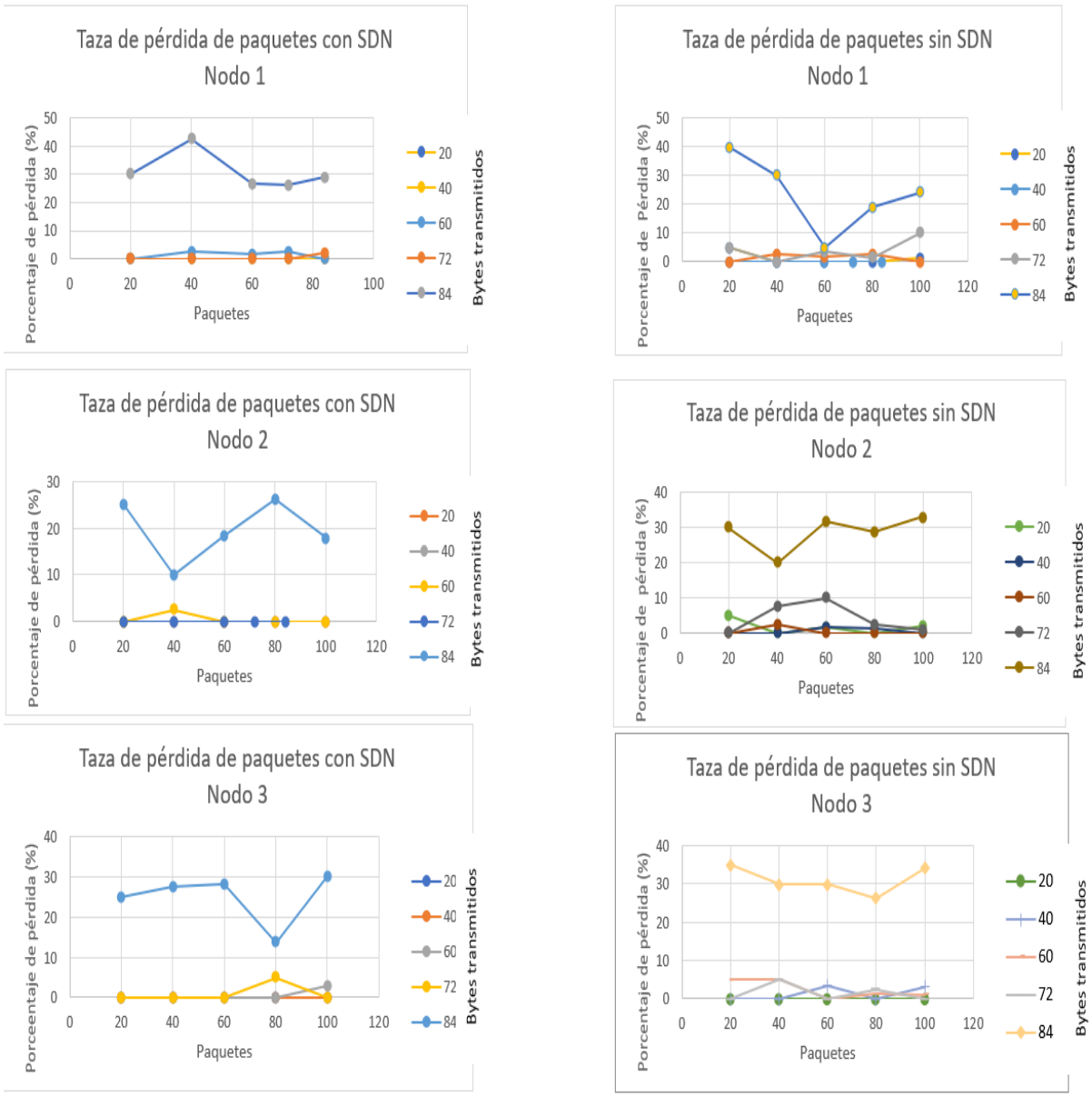


Figura 58. Comparación de la taza de pérdida de paquetes entre SDN y sin SDN

Para evaluar el promedio del retardo entre los nodos y el servidor se envían diferentes paquetes ICMPv3 con distintos tamaños de carga útil, para luego registrar los tiempos que se demora en procesar los paquetes ICMPv3. En la tabla 31 se registra los valores RTT que se expresan en milisegundos (ms).

Tabla 31.
Comparación de los valores RTT de los dos escenarios

Tamaño de trama	Resultados de los tiempos de retardo entre los nodos y el servidor local expresado en milisegundo					
	Nodo 1		Nodo 2		Nodo 3	
	SDN	Sin SDN	SDN	Sin SDN	SDN	Sin SDN
20	105.919	113.030	172.090	260.829	119.137	123.457
40	110.795	120.343	227.211	293.852	106.137	131.455
60	121.451	132.449	255.532	281.585	119.821	138.426
72	127.354	199.230	366.278	702.837	204.665	249.536
84	209.278	230.506	368.601	480.945	196.232	234.249

Fuente: Elaborado por el autor.

En la figura 59 se representa dos curvas obtenidas con diferentes valores de tamaños de carga útil (20,40,60,72 y 84 bytes). Si nos fijamos en la línea tomate que representa el primer escenario se puede observar que el tiempo RTT aumenta a medida que se aumenta la carga a diferencia de la línea azul que representa el segundo escenario, se nota una ligera reducción en el tiempo RTT, esto es debido a la red SDN permite ordenar y priorizar el flujo de datos.

Es importante destacar que en el momento de realizar las pruebas los nodos sensores se ubicaban una distancia de 3.15 metros del nodo gateway además para obtener datos fiables los nodos sensores estarán conectados a una fuente de energía constante.

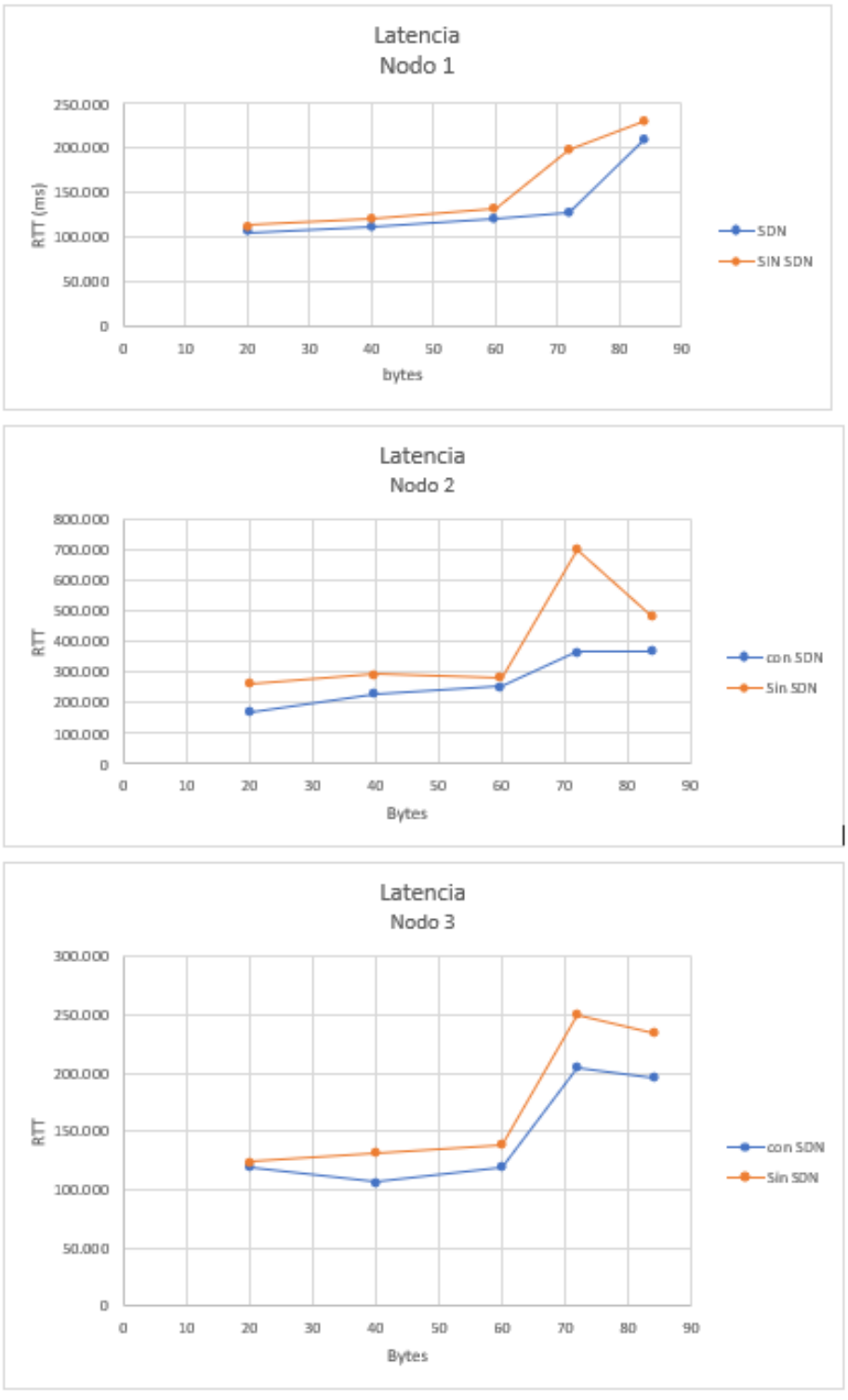


Figura 59. Representación gráfica del RTT de cada uno de los nodos

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- En este documento, se presenta un diseño de arquitectura híbrida que permite relacionar el funcionamiento de las redes WSN y SDN, es decir proporciona la capacidad de gestionar de forma centralizada el flujo de datos a través de la dirección IPv6 de los nodos juntamente con aplicación SDN. Sin embargo, al incluir una red de baja potencia aún se depende de un nodo central que es encargado de sincronizar la comunicación con el resto de los nodos por medio del protocolo RPL.
- El tiempo que tarda en establecer una comunicación entre el nodo sensor y el servidor local después de instalar una regla de flujo es aproximadamente entre uno a dos minutos, eso es debido, a que la trama UDP es comparada con la tabla de flujo con la finalidad de determinar si existe una coincidencia, otro factor que afecta a establecer la comunicación es la distancia entre el nodo sensor y gateway. Sin embargo, al encontrar una coincidencia este proceso no se repite para los demás flujos de datos provenientes del nodo habilitado.
- La solución planteada utiliza la aplicación SDN denominado firewall que permite habilitar el flujo de datos, sin que los nodos estén vínculos al dominio del controlador ryu o soporten openflow de esta manera no se afecta el rendimiento de los dispositivos que conforman la red WSN.
- Para habilitar el tráfico de datos de los de los nodos sensores es importante habilitar el direccionamiento multidifusión el cual, permite determinar la disponibilidad del servidor local y verificar la duplicidad de la dirección IPv6 dentro de la red.
- La implementación de las SDN dentro de una red WSN tiene sus beneficios, y así que, de la evaluación realizada se concluye que la gestión de datos a través de la aplicación

permitió mejorar el tiempo de respuesta y la entrega de datos entre el servidor y nodo sensor como también contribuye de forma general a la seguridad de la red WSN.

- La coexistencia entre las dos redes es debido a que, las redes 6lowpan proporcionan una dirección IP y define una estructura de paquetes UDP que pueda ser interpretado por el switch openflow para ser gestionados por la aplicación SDN.

Recomendaciones

- Para obtener un control de los dispositivos conectados al switch openflow se recomienda redirigir todo el tráfico de la interfaz física del raspberry pi hacia la interfaz virtual del open vswitch denomina bridge.
- Es necesario que se verifique la compatibilidad entre el controlador y la versión del sistema operativo, debido a que en el caso de RYU no tiene soporte de actualización.
- Cuando se inserta reglas de flujo es necesario que se identifique de forma correcta las direcciones IPv6 de cada uno de los nodos y de esta forma evitar confusiones.
- Las memorias microSD de las Raspberry Pi deben ser de clase 10 evitar problemas en la ejecución del sistema operativo.
- La configuración de nodo gateway debe tener una configuración transparente para que permita el paso del flujo de datos al openvswitch.
- Es importante suministrar el voltaje y corriente recomendada en las placas raspberry pi con el propósito de que, todos dispositivos periféricos conectados cumplan con su función correctamente.

BIBLIOGRAFÍA

Al-Kashoash, H. (2019a). *Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things*. Springer International Publishing. https://books.google.com.ec/books?id=sRCTDwAAQBAJ

Al-Kashoash, H. (2019b). *Congestion Control for 6LoWPAN Wireless Sensor Networks*. Springer. https://books.google.com.ec/books?id=sRCTDwAAQBAJ

Aldea, E. L. (n.d.). *Arduino. Gu{\'\i}a práctica de fundamentos y simulación*. https://books.google.com.ec/books?id=Wo6fDwAAQBAJ

Beltrano, J., & Gimenez, D. O. (2015). Introducción al cultivo hidropónico. *Cultivo En Hidroponía*, *1*(978-950-34-1258–9), 181. http://sedici.unlp.edu.ar/bitstream/handle/10915/46752/Documento_completo.pdf?sequence=1

Benzekki, K., El Fergougui, A., & Elbelrhiti Elalaoui, A. (2016). Software-defined networking (SDN): a survey. *Security and Communication Networks*, *9*(18), 5803–5833. https://doi.org/10.1002/sec.1737

Bhat, N. S. (2011). *Design and Implementation of IEEE 802 . 15 . 4 Mac Protocol on FPGA*. 4–8.

Bohloulzadeh, A., & Rajaei, M. (2020). A Survey on Congestion Control Protocols in Wireless Sensor Networks. *International Journal of Wireless Information Networks*, *27*(3), 365–384. https://doi.org/10.1007/s10776-020-00479-3

Caprile, S. R. (2009). *Equisbi: Desarrollo de aplicaciones con comunicación remota basadas en módulos ZigBee y 802.15.4*. GAE. https://books.google.com.ec/books?id=xTXv5-AhOhMC

Centeno, A. G., Manuel, C., Vergel, R., & Calderón, C. A. (2017). *Controladores SDN, elementos para su selección y evaluación*. *October*.

Ćulibrk, D., Vukobratovic, D., Minic, V., Fernandez, M. A., Osuna, J. A., & Crnojevic, V. (2013). *Sensing Technologies For Precision Irrigation*. Springer New York. https://books.google.com.ec/books?id=vOK2BAAAQBAJ

Daneels, G., Van Leemput, D., Delgado, C., De Poorter, E., Latré, S., & Famaey, J. (2021). Parent and PHY selection in slot bonding IEEE 802.15.4e TSCH networks. *Sensors*, *21*(15), 1–21. https://doi.org/10.3390/s21155150

Davoli, G., Cerroni, W., Tomovic, S., Buratti, C., Contoli, C., & Callegati, F. (2019). Intent-based service management for heterogeneous software-defined infrastructure domains. *International Journal of Network Management*, *29*(1), 1–22. https://doi.org/10.1002/nem.2051

Duan, Q., & Toy, M. (2017). *Virtualized Software-Defined Networks and Services*.

Ee, G. K., Ng, C. K., Noordin, N. K., & Ali, B. M. (2010). A Review of 6LoWPAN Routing Protocols. *Proceedings of the Asia-Pacific Advanced Network*, *30*(0), 71. https://doi.org/10.7125/APAN.30.11

El-Mougy, A., Ibnkahla, M., & Hegazy, L. (2015). Software-defined wireless network architectures for the Internet-of-Things. *Proceedings - Conference on Local Computer Networks, LCN*, *2015-Decem*, 804–811. https://doi.org/10.1109/LCNW.2015.7365931

Electronics, R. (n.d.). *USB Host Shield 2.0 for Arduino « Circuits@Home*. http://www.circuitsathome.com/products-page/arduino-shields/usb-host-shield-2-0-for-arduino

Emmerich, P., Raumer, D., Gallenmüller, S., Wohlfart, F., & Carle, G. (2018). Throughput and

Latency of Virtual Switching with Open vSwitch: A Quantitative Analysis. *Journal of Network and Systems Management*, *26*(2), 314–338. https://doi.org/10.1007/s10922-017-9417-0

Ezziyyani, M. (2019). *Advanced Intelligent Systems for Sustainable Development (AI2SD'2019): Volume 6 - Advanced Intelligent Systems for Networks and Systems*. Springer International Publishing. https://books.google.com.ec/books?id=Hcm1DwAAQBAJ

Fernández, J. A. P., Villalba, L. J. G., & Kim, T. H. (2018). Software defined networks in wireless sensor architectures. *Entropy*, *20*(4), 1–23. https://doi.org/10.3390/e20040225

Ghosh, R. K. (2017). Wireless networking and mobile data management. In *Wireless Networking and Mobile Data Management*. https://doi.org/10.1007/9789811039416

Gonzalez, C., Flauzac, O., & Nolot, F. (2018). Evolución y Contribución para el Internet de las Cosas por las emergentes Redes Definidas por Software Evolution and Contribution for the Internet of Things by the Emerging Software-defined networking. *Revistas.Utp.Ac.Pa*, 28–33. https://revistas.utp.ac.pa/index.php/memoutp/article/view/1842

Göransson, P. (2014). Software Defined Networks A Comprehensive Approach [2014]. In *Elsevier*. https://doi.org/10.1017/CBO9781107415324.004

Graziani, R. (2017). *IPv6 Fundamentals: A Straightforward Approach to Understanding IPv6 Second Edition* (R. (2017). Ip. F. A. S. A. to U. Ip. S. E. http://www. nodis-cisco. com/wp-content/download/IPv6_Fundamentals_A_Straightforward. pd. Graziani (ed.)). http://www.nodis-cisco.com/wp-content/download/IPv6_Fundamentals_A_Straightforward.pdf

Grgic, K., Zagar, D., & Krizanovic Cik, V. (2016). System for Malicious Node Detection in

IPv6-Based Wireless Sensor Networks. *Journal of Sensors*, *2016*. https://doi.org/10.1155/2016/6206353

Hu, F. E. I. (2014). Network Innovation through OpenFlow and SDN. In *Network Innovation through OpenFlow and SDN*. https://doi.org/10.1201/b16521

Jararweh, Y., Al-Ayyoub, M., Doulat, A., Abed Al Aziz, A. Al, Haythem, A. B. S., & Abdallah, A. K. (2015). Software defned cognitive radio network framework: Design and evaluation. *International Journal of Grid and High Performance Computing*, *7*(1), 15–31. https://doi.org/10.4018/ijghpc.2015010102

Jareño, R. G. (2016). *IP v6*. 5.

Jung, H. K., Kim, J. T., Sahama, T., & Yang, C. H. (2013). *Future Information Communication Technology and Applications: ICFICE 2013*. Springer Netherlands. https://books.google.com.ec/books?id=ydlEAAAAQBAJ

Khorsandroo, S., Sánchez, A. G., Tosun, A. S., Arco, J. M., & Doriguzzi-Corin, R. (2021). Hybrid SDN evolution: A comprehensive survey of the state-of-the-art. *Computer Networks*, *192*, 107981. https://doi.org/10.1016/j.comnet.2021.107981

Kobo, H. I., Abu-Mahfouz, A. M., & Hancke, G. P. (2017). A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. *IEEE Access*, *5*, 1872–1899. https://doi.org/10.1109/ACCESS.2017.2666200

Kumar, V., Oikonomou, G., Tryfonas, T., Page, D., & Phillips, I. (2014). Digital investigations for IPv6-based wireless sensor networks. *Proceedings of the Digital Forensic Research Conference, DFRWS 2014 USA*, *11*, S66–S75. https://doi.org/10.1016/j.diin.2014.05.005

Lee, W., Nam, K., Roh, H. G., & Kim, S. H. (2016). A gateway based fog computing architecture for wireless sensors and actuator networks. *International Conference on*

Advanced Communication Technology, ICACT, *2016-March*, 210–213. https://doi.org/10.1109/ICACT.2016.7423332

Loshin, P. (2004). *IPv6: Theory, Protocol, and Practice*. Elsevier Science. https://books.google.com.ec/books?id=6JDuPUzMU4AC

Luo, T., Tan, H. P., & Quek, T. Q. S. (2012). Sensor openflow: Enabling software-defined wireless sensor networks. *IEEE Communications Letters*, *16*(11), 1896–1899. https://doi.org/10.1109/LCOMM.2012.092812.121712

Mamushiane, L., Lysko, A., & Dlamini, S. (2018). A comparative evaluation of the performance of popular SDN controllers. *IFIP Wireless Days*, *2018-April*(2), 54–59. https://doi.org/10.1109/WD.2018.8361694

Minoli, D. (2013). Building the Internet of Things with IPv6 and MIPv6. In *Building the Internet of Things with IPv6 and MIPv6*. https://doi.org/10.1002/9781118647059

Modieginyane, K. M., Letswamotse, B. B., Malekian, R., & Abu-Mahfouz, A. M. (2018). Software defined wireless sensor networks application opportunities for efficient network management: A survey. *Computers & Electrical Engineering*, *66*, 274–287. https://doi.org/10.1016/J.COMPELECENG.2017.02.026

Morreale, P. A., & Anderson, J. M. (2015). *Software defined networking: design and deployment*. https://doi.org/10.1145/2980258.2980294

Mostafaei, H., & Menth, M. (2018). Software-defined wireless sensor networks: A survey. *Journal of Network and Computer Applications*, *119*, 42–56. https://doi.org/10.1016/j.jnca.2018.06.016

Muñoz, A. M. (n.d.). *Arduino. Edición 2018 Curso práctico*. https://books.google.com.ec/books?id=yo6fDwAAQBAJ

Muñoz, O. Q. (2019). *Internet de las Cosas (IoT)*. Ibukku, LLC. https://books.google.com.ec/books?id=vnnEDwAAQBAJ

Nadeau, T. D., & Gray, K. (2013). *SDN : software defined networks*. O'Reilly Media.

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings*. https://doi.org/10.1109/SysEng.2017.8088251

ORGONE TECNOLOGY. (n.d.). *Sensor de Ph con módulo (PH-4502C) | orgontec*. Retrieved January 12, 2022, from https://www.orgontec.com/product-page/sensor-de-ph-con-módulo-ph-4502c

Ortega-Corral, C., Acosta, O. R., Campo, D., Enrique, J., Montoya, L., Jaime, J., Elizondo, E., Palafox, L. E., Guerra Frausto, R., Reyes, R. A., & López Cruz, F. (2015). Desarrollo E Implementación De Una Estación Base Para Una Red Inalámbrica De Sensores De Largo Alcance Con Conexión a La Nube. *Congreso Internacional En Ingeniería Electrónica. Mem. Electro*, *37*(October), 92–97. https://doi.org/10.13140/RG.2.1.2951.2404

Polastre, J., Szewczyk, R., & Culler, D. (2005). Telos: Enabling ultra-low power wireless research. *2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005*, *2005*, 364–369. https://doi.org/10.1109/IPSN.2005.1440950

Rothenberg, C. E., Ieee, M., Azodolmolky, S., Ieee, S. M., Uhlig, S., & Ieee, M. (2015). Software-Defined Networking : A Comprehensive Survey. *Proceedings of the IEEE*, *103*(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

Ruiz Canales, A., & Molina Martínez, J. M. (2016). *Automatización y telecontrol de sistemas de riego*.

Ruzzelli, A. (2011). Signal and Communication. In *Encyclopedia of Mobile Computing and*

Commerce. https://doi.org/10.4018/978-1-59904-002-8.ch173

Sasián, F., Theron, R., & Gachet, D. (2016). Protocolo para comunicación inalámbrica en instalaciones de energías renovables. *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, *13*(3), 310–321. https://doi.org/10.1016/j.riai.2016.05.003

Sensor, T., & Kit, E. (n.d.). *SHT1x / SHT7x*. 1–9.

Shelby, Z., & Bormann, C. (2011). *6LoWPAN: The Wireless Embedded Internet - Shelby - Wiley Online Library*. http://onlinelibrary.wiley.com/book/10.1002/9780470686218;jsessionid=1BDEF8F5F70E795897585F984C9D5ECA.f03t03

Sinaeepourfard, A., Garcia, J., Masip-Bruin, X., & Marin-Tordera, E. (2017). A Novel Architecture for Efficient Fog to Cloud Data Management in Smart Cities. *Proceedings - International Conference on Distributed Computing Systems*, 2622–2623. https://doi.org/10.1109/ICDCS.2017.202

Sommerville, I., & Galipienso, M. I. A. (2005). *Ingenier{\'\i}a del software*. Pearson Educación. https://books.google.com.ec/books?id=gQWd49zSut4C

Umamaheswari, S., Preethi, A., Pravin, E., & Dhanusha, R. (2017). Integrating scheduled hydroponic system. *2016 IEEE International Conference on Advances in Computer Applications, ICACA 2016*, 333–337. https://doi.org/10.1109/ICACA.2016.7887976

Vargas, J. H. B. (2010). *Curso basico de Hidroponia*. Bosques Hidropónicos. https://books.google.com.ec/books?id=GV_XAQAAQBAJ

Vasseur, J.-P. (2012). Interconnecting Smart Objects with IP. In *הנוטע עלון* (Vol. 66).

Wilson, A., & Suárez, Á. A. M. (2018). *Los hidropónicos: La guia suprema de los hidroponicos para salvar tiempo y dinero*. Adidas Wilson.

https://books.google.com.ec/books?id=l2B7DwAAQBAJ

Yang, S. H. (2013). *Wireless Sensor Networks: Principles, Design and Applications*. Springer London. https://books.google.com.ec/books?id=zhG4BAAAQBAJ

Al-Kashoash, H. (2019a). *Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things*. Springer International Publishing. https://books.google.com.ec/books?id=sRCTDwAAQBAJ

Al-Kashoash, H. (2019b). *Congestion Control for 6LoWPAN Wireless Sensor Networks*. Springer. https://books.google.com.ec/books?id=sRCTDwAAQBAJ

Aldea, E. L. (n.d.). *Arduino. Gu{\'\i}a práctica de fundamentos y simulación*. https://books.google.com.ec/books?id=Wo6fDwAAQBAJ

Beltrano, J., & Gimenez, D. O. (2015). Introducción al cultivo hidropónico. *Cultivo En Hidroponía*, *1*(978-950-34-1258–9), 181. http://sedici.unlp.edu.ar/bitstream/handle/10915/46752/Documento_completo.pdf?sequence=1

Benzekki, K., El Fergougui, A., & Elbelrhiti Elalaoui, A. (2016). Software-defined networking (SDN): a survey. *Security and Communication Networks*, *9*(18), 5803–5833. https://doi.org/10.1002/sec.1737

Bhat, N. S. (2011). *Design and Implementation of IEEE 802 . 15 . 4 Mac Protocol on FPGA*. 4–8.

Bohloulzadeh, A., & Rajaei, M. (2020). A Survey on Congestion Control Protocols in Wireless Sensor Networks. *International Journal of Wireless Information Networks*, *27*(3), 365–384. https://doi.org/10.1007/s10776-020-00479-3

Caprile, S. R. (2009). *Equisbi: Desarrollo de aplicaciones con comunicación remota basadas*

en módulos ZigBee y 802.15.4. GAE. https://books.google.com.ec/books?id=xTXv5-AhOhMC

Centeno, A. G., Manuel, C., Vergel, R., & Calderón, C. A. (2017). *Controladores SDN, elementos para su selección y evaluación*. *October*.

Ćulibrk, D., Vukobratovic, D., Minic, V., Fernandez, M. A., Osuna, J. A., & Crnojevic, V. (2013). *Sensing Technologies For Precision Irrigation*. Springer New York. https://books.google.com.ec/books?id=vOK2BAAAQBAJ

Daneels, G., Van Leemput, D., Delgado, C., De Poorter, E., Latré, S., & Famaey, J. (2021). Parent and PHY selection in slot bonding IEEE 802.15.4e TSCH networks. *Sensors*, *21*(15), 1–21. https://doi.org/10.3390/s21155150

Davoli, G., Cerroni, W., Tomovic, S., Buratti, C., Contoli, C., & Callegati, F. (2019). Intent-based service management for heterogeneous software-defined infrastructure domains. *International Journal of Network Management*, *29*(1), 1–22. https://doi.org/10.1002/nem.2051

Duan, Q., & Toy, M. (2017). *Virtualized Software-Defined Networks and Services*.

Ee, G. K., Ng, C. K., Noordin, N. K., & Ali, B. M. (2010). A Review of 6LoWPAN Routing Protocols. *Proceedings of the Asia-Pacific Advanced Network*, *30*(0), 71. https://doi.org/10.7125/APAN.30.11

El-Mougy, A., Ibnkahla, M., & Hegazy, L. (2015). Software-defined wireless network architectures for the Internet-of-Things. *Proceedings - Conference on Local Computer Networks, LCN*, *2015-Decem*, 804–811. https://doi.org/10.1109/LCNW.2015.7365931

Electronics, R. (n.d.). *USB Host Shield 2.0 for Arduino « Circuits@Home*. http://www.circuitsathome.com/products-page/arduino-shields/usb-host-shield-2-0-for-

arduino

Emmerich, P., Raumer, D., Gallenmüller, S., Wohlfart, F., & Carle, G. (2018). Throughput and Latency of Virtual Switching with Open vSwitch: A Quantitative Analysis. *Journal of Network and Systems Management*, *26*(2), 314–338. https://doi.org/10.1007/s10922-017-9417-0

Ezziyyani, M. (2019). *Advanced Intelligent Systems for Sustainable Development (AI2SD'2019): Volume 6 - Advanced Intelligent Systems for Networks and Systems*. Springer International Publishing. https://books.google.com.ec/books?id=Hcm1DwAAQBAJ

Fernández, J. A. P., Villalba, L. J. G., & Kim, T. H. (2018). Software defined networks in wireless sensor architectures. *Entropy*, *20*(4), 1–23. https://doi.org/10.3390/e20040225

Ghosh, R. K. (2017). Wireless networking and mobile data management. In *Wireless Networking and Mobile Data Management*. https://doi.org/10.1007/9789811039416

Gonzalez, C., Flauzac, O., & Nolot, F. (2018). Evolución y Contribución para el Internet de las Cosas por las emergentes Redes Definidas por Software Evolution and Contribution for the Internet of Things by the Emerging Software-defined networking. *Revistas.Utp.Ac.Pa*, 28–33. https://revistas.utp.ac.pa/index.php/memoutp/article/view/1842

Göransson, P. (2014). Software Defined Networks A Comprehensive Approach [2014]. In *Elsevier*. https://doi.org/10.1017/CBO9781107415324.004

Graziani, R. (2017). *IPv6 Fundamentals: A Straightforward Approach to Understanding IPv6 Second Edition* (R. (2017). Ip. F. A. S. A. to U. Ip. S. E. http://www. nodis-cisco. com/wp-content/download/IPv6_Fundamentals_A_Straightforward. pd. Graziani (ed.)). http://www.nodis-cisco.com/wp-

content/download/IPv6_Fundamentals_A_Straightforward.pdf

Grgic, K., Zagar, D., & Krizanovic Cik, V. (2016). System for Malicious Node Detection in IPv6-Based Wireless Sensor Networks. *Journal of Sensors*, *2016*. https://doi.org/10.1155/2016/6206353

Hu, F. E. I. (2014). Network Innovation through OpenFlow and SDN. In *Network Innovation through OpenFlow and SDN*. https://doi.org/10.1201/b16521

Jararweh, Y., Al-Ayyoub, M., Doulat, A., Abed Al Aziz, A. Al, Haythem, A. B. S., & Abdallah, A. K. (2015). Software defned cognitive radio network framework: Design and evaluation. *International Journal of Grid and High Performance Computing*, *7*(1), 15–31. https://doi.org/10.4018/ijghpc.2015010102

Jareño, R. G. (2016). *IP v6*. 5.

Jung, H. K., Kim, J. T., Sahama, T., & Yang, C. H. (2013). *Future Information Communication Technology and Applications: ICFICE 2013*. Springer Netherlands. https://books.google.com.ec/books?id=ydlEAAAAQBAJ

Khorsandroo, S., Sánchez, A. G., Tosun, A. S., Arco, J. M., & Doriguzzi-Corin, R. (2021). Hybrid SDN evolution: A comprehensive survey of the state-of-the-art. *Computer Networks*, *192*, 107981. https://doi.org/10.1016/j.comnet.2021.107981

Kobo, H. I., Abu-Mahfouz, A. M., & Hancke, G. P. (2017). A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. *IEEE Access*, *5*, 1872–1899. https://doi.org/10.1109/ACCESS.2017.2666200

Kumar, V., Oikonomou, G., Tryfonas, T., Page, D., & Phillips, I. (2014). Digital investigations for IPv6-based wireless sensor networks. *Proceedings of the Digital Forensic Research Conference, DFRWS 2014 USA*, *11*, S66–S75. https://doi.org/10.1016/j.diin.2014.05.005

Lee, W., Nam, K., Roh, H. G., & Kim, S. H. (2016). A gateway based fog computing architecture for wireless sensors and actuator networks. *International Conference on Advanced Communication Technology, ICACT*, *2016-March*, 210–213. https://doi.org/10.1109/ICACT.2016.7423332

Loshin, P. (2004). *IPv6: Theory, Protocol, and Practice*. Elsevier Science. https://books.google.com.ec/books?id=6JDuPUzMU4AC

Luo, T., Tan, H. P., & Quek, T. Q. S. (2012). Sensor openflow: Enabling software-defined wireless sensor networks. *IEEE Communications Letters*, *16*(11), 1896–1899. https://doi.org/10.1109/LCOMM.2012.092812.121712

Mamushiane, L., Lysko, A., & Dlamini, S. (2018). A comparative evaluation of the performance of popular SDN controllers. *IFIP Wireless Days*, *2018-April*(2), 54–59. https://doi.org/10.1109/WD.2018.8361694

Minoli, D. (2013). Building the Internet of Things with IPv6 and MIPv6. In *Building the Internet of Things with IPv6 and MIPv6*. https://doi.org/10.1002/9781118647059

Modieginyane, K. M., Letswamotse, B. B., Malekian, R., & Abu-Mahfouz, A. M. (2018). Software defined wireless sensor networks application opportunities for efficient network management: A survey. *Computers & Electrical Engineering*, *66*, 274–287. https://doi.org/10.1016/J.COMPELECENG.2017.02.026

Morreale, P. A., & Anderson, J. M. (2015). *Software defined networking: design and deployment*. https://doi.org/10.1145/2980258.2980294

Mostafaei, H., & Menth, M. (2018). Software-defined wireless sensor networks: A survey. *Journal of Network and Computer Applications*, *119*, 42–56. https://doi.org/10.1016/j.jnca.2018.06.016

Muñoz, A. M. (n.d.). *Arduino. Edición 2018 Curso práctico*. https://books.google.com.ec/books?id=yo6fDwAAQBAJ

Muñoz, O. Q. (2019). *Internet de las Cosas (IoT)*. Ibukku, LLC. https://books.google.com.ec/books?id=vnnEDwAAQBAJ

Nadeau, T. D., & Gray, K. (2013). *SDN : software defined networks*. O'Reilly Media.

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings*. https://doi.org/10.1109/SysEng.2017.8088251

ORGONE TECNOLOGY. (n.d.). *Sensor de Ph con módulo (PH-4502C) | orgontec*. Retrieved January 12, 2022, from https://www.orgontec.com/product-page/sensor-de-ph-con-módulo-ph-4502c

Ortega-Corral, C., Acosta, O. R., Campo, D., Enrique, J., Montoya, L., Jaime, J., Elizondo, E., Palafox, L. E., Guerra Frausto, R., Reyes, R. A., & López Cruz, F. (2015). Desarrollo E Implementación De Una Estación Base Para Una Red Inalámbrica De Sensores De Largo Alcance Con Conexión a La Nube. *Congreso Internacional En Ingeniería Electrónica. Mem. Electro*, *37*(October), 92–97. https://doi.org/10.13140/RG.2.1.2951.2404

Polastre, J., Szewczyk, R., & Culler, D. (2005). Telos: Enabling ultra-low power wireless research. *2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005*, *2005*, 364–369. https://doi.org/10.1109/IPSN.2005.1440950

Rothenberg, C. E., Ieee, M., Azodolmolky, S., Ieee, S. M., Uhlig, S., & Ieee, M. (2015). Software-Defined Networking : A Comprehensive Survey. *Proceedings of the IEEE*, *103*(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

Ruiz Canales, A., & Molina Martínez, J. M. (2016). *Automatización y telecontrol de sistemas*

de riego.

Ruzzelli, A. (2011). Signal and Communication. In *Encyclopedia of Mobile Computing and Commerce*. https://doi.org/10.4018/978-1-59904-002-8.ch173

Sasián, F., Theron, R., & Gachet, D. (2016). Protocolo para comunicación inalámbrica en instalaciones de energías renovables. *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, *13*(3), 310–321. https://doi.org/10.1016/j.riai.2016.05.003

Sensor, T., & Kit, E. (n.d.). *SHT1x / SHT7x*. 1–9.

Shelby, Z., & Bormann, C. (2011). *6LoWPAN: The Wireless Embedded Internet - Shelby - Wiley Online Library*. http://onlinelibrary.wiley.com/book/10.1002/9780470686218;jsessionid=1BDEF8F5F70E795897585F984C9D5ECA.f03t03

Sinaeepourfard, A., Garcia, J., Masip-Bruin, X., & Marin-Tordera, E. (2017). A Novel Architecture for Efficient Fog to Cloud Data Management in Smart Cities. *Proceedings - International Conference on Distributed Computing Systems*, 2622–2623. https://doi.org/10.1109/ICDCS.2017.202

Sommerville, I., & Galipienso, M. I. A. (2005). *Ingenier{\'\i}a del software*. Pearson Educación. https://books.google.com.ec/books?id=gQWd49zSut4C

Umamaheswari, S., Preethi, A., Pravin, E., & Dhanusha, R. (2017). Integrating scheduled hydroponic system. *2016 IEEE International Conference on Advances in Computer Applications, ICACA 2016*, 333–337. https://doi.org/10.1109/ICACA.2016.7887976

Vargas, J. H. B. (2010). *Curso basico de Hidroponia*. Bosques Hidropónicos. https://books.google.com.ec/books?id=GV_XAQAAQBAJ

Vasseur, J.-P. (2012). Interconnecting Smart Objects with IP. In *הנוטע עלון* (Vol. 66).

Wilson, A., & Suárez, Á. A. M. (2018). *Los hidropónicos: La guia suprema de los hidroponicos para salvar tiempo y dinero*. Adidas Wilson. https://books.google.com.ec/books?id=l2B7DwAAQBAJ

Yang, S. H. (2013). *Wireless Sensor Networks: Principles, Design and Applications*. Springer London. https://books.google.com.ec/books?id=zhG4BAAAQBAJ

ANEXOS

Anexo A. Análisis Bibliográfico

A.1. Revisión sistemática

Con el propósito de validar y determinar los requerimientos de este proyecto de titulación se realizó una investigación bibliográfica de manera que, permita obtener datos que aporten al diseño de una arquitectura de red de sensores definidos por software. En la figura 60 detalla el proceso de búsqueda que consisten en tres etapas: planificación de la búsqueda, realización de búsqueda y presentación de resultados.

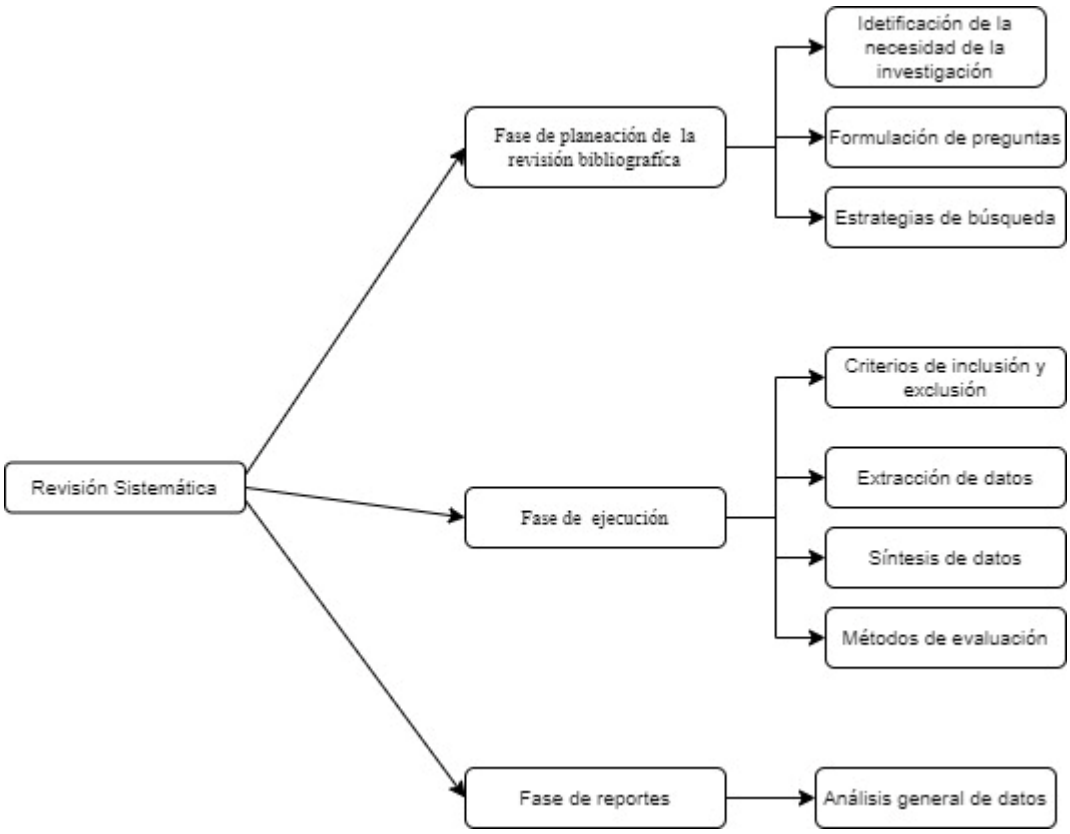


Figura 60. Proceso de la revisión sistemática

A.2. Fase planeamiento de la revisión

Antes de empezar con el proceso de investigación es importante planificar y delimitar la revisión sistemática para lo cual, se define tópicos o preguntas relacionados con el tema con el propósito de extraer información que ayude al desarrollo del proyecto.

A.2.1. Identificación de la necesidad de la revisión

Al realizar una primera búsqueda con relación al tema planteado, se puede identificar que la integración de las dos tecnologías representa un gran desafío, ya que existen diferentes soluciones como, por ejemplo, soluciones basadas en problemas específicos sobre las redes WSN o soluciones que abarquen diferentes tecnologías que no afecten las limitaciones de los dispositivos de las redes WSN.

Por este motivo se procedió un proceso de búsqueda de documentos científicos que permita establecer criterios significativos y que ayuden a diseñar e implementar una arquitectura operativa entre las dos tecnologías.

A.2.2. Formulación de preguntas

En este apartado consiste definir los factores o características que se van a tomar en cuenta al momento de seleccionar un artículo como, por ejemplo: soluciones planteadas, protocolos utilizados y, aplicaciones. Estas características son expresadas en forma de preguntas para extraer información específica de los artículos científicos. La tabla 32 se muestra un listado de preguntas que se van a aplicar para extraer información.

Tabla 32.
Lista de preguntas para extraer información de los artículos científicos

Nº Pregunta	Preguntas de Investigación
1	¿Qué soluciones son implementadas para la integración de redes WSN y SDN?
2	¿Cuál es el objetivo de integrar las redes WSN y SDN?
3	¿Qué tendencias o tecnologías son incluidas en las soluciones propuestas en los artículos?

4	¿En qué escenarios han sido implementados las soluciones propuestas en los artículos?
5	¿Qué protocolos son incluidos en las soluciones?
6	¿Qué métricas se contemplan para validar la solución presentada en los artículos?
7	¿Qué limitaciones son incluidas al momento de integrar las dos tecnologías?

Fuente: Elaborado por el autor

A.2.3. Estrategia de búsqueda

En esta sección consiste en extraer artículos de los principales repositorios digitales como: Google *Academic, sensors, ScienceDirect, ResearchGate, MDPI (Multidisciplinary Digital Publishing Institute*) y IEEE Explore. De los artículos obtenidos se realiza un proceso de selección dependiendo los criterios de inclusión y exclusión. Por otra parte, con la finalidad de agilizar el proceso de búsqueda dentro de los repositorios digitales se establecen palabras claves. Los cuales se presenta a continuación:

- 6LBR en redes IOT Y SDN
- WSN and SDN
- Integration WSN and SDN
- 6lowpan and SDN
- Applications developed for WSN and SDN
- Computación en la nube, WSN y SDN
- Gateway y SDN

A.2.4. Criterios de Inclusión y Exclusión

Al encontrar una gran cantidad de artículos relacionados con el tema de investigación se establecieron criterios de exclusión e inclusión, de manera que se puede seleccionar la información más idónea y respalde el presente trabajo, para excluir un artículo científico se considera los siguientes criterios:

- Documentos duplicados.

- Artículos cortos e introductorio.
- Artículos que no superen los 5 años de publicación.

Mientras que los artículos aceptados deben tener criterios que ayuden a determinar requerimientos y limitaciones. Entre los criterios que se toman en cuenta son las siguientes:

- Se eligen los artículos que presenten metodología clara y precisa con la finalidad de proporcionar información de calidad para el desarrollo del proyecto
- Los artículos deben proporcionar los recursos necesarios para recrear la solución propuesta
- Los artículos deben presentan resultado sobre la solución planteada

A.3. Fase de Ejecución de la Investigación

Esta fase consiste en ejecutar el plan búsqueda para obtener la documentación necesaria, que luego será analiza de acuerdo con las preguntas y los criterios que se mencionaron en las secciones anteriores, como resultado de este proceso se obtuvo una totalidad de 59 artículos de los diferentes repositorios digitales. En la figura 61 se evidencia los resultados obtenidos del proceso de búsqueda inicial de la información y el proceso de análisis donde se aplican los criterios de exclusión e inclusión para evitar información errónea.

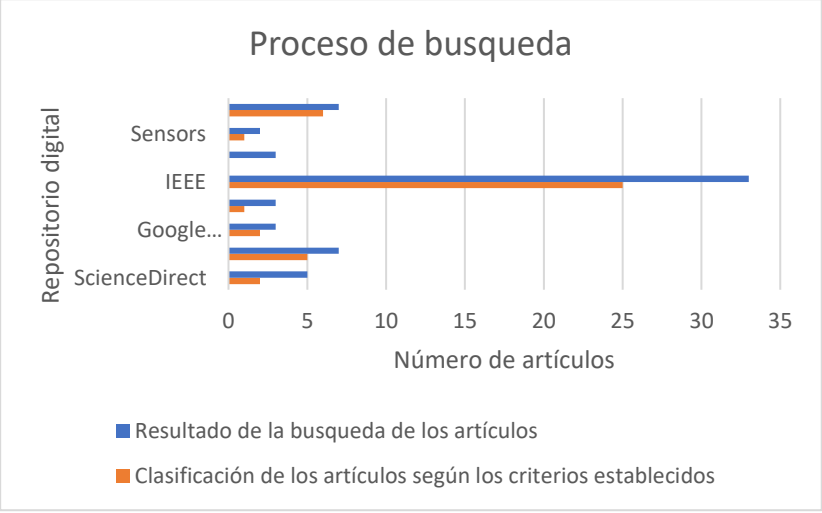


Figura 61. Resultado del proceso de búsqueda en los diferentes repositorios digitales

A.3.1. Extracción de datos

Al finalizar el proceso de búsqueda se procede contestar cada uno las preguntas planteadas en la tabla 32.

Preg1: Qué soluciones son implementadas para la integración de Redes WSN y SDN

Para integrar las redes de sensores inalámbricos y SDN se han identificado diferentes propuestas que dependen de la funcionalidad y de los problemas que se desea mitigar de la red WSN, Sin embargo, se puede identificar dos soluciones; la primera red de sensores programables y la segunda red de sensores basados en SDN. Entre las soluciones más significativas relacionadas con las redes de sensores programables se tienen: SDWN, SDN-WISE, uSDN, TINY-SDN, SDWSN, WISHPER

Preg2: ¿Cuál es el objetivo de integrar las redes WSN y SDN?

La integración de una red de sensores inalámbricos dentro de la red SDN tiene dos enfoques; la primera consiste en que un nodo sensores cumplan con las funciones del protocolo openflow con la finalidad de solucionar los problemas que tiene las redes WSN, por ejemplo: optimizar el consumo de energía, mejorar proceso de enrutamiento, mejorar la configuración y actualización de los nodos sensores, etc. Por otro lado, el segundo enfoque consiste en que el nodo principal de red WSN sea administrada por elemento de una red SDN con el propósito de aumentar la heterogeneidad, interoperabilidad, gestión de flujo de datos y análisis de bigdata.

Preg3: ¿Qué tecnologías son incluidas en las soluciones propuestas en los artículos?

Dentro de este análisis se identificaron varias tecnologías que aportan para que una red WSN sea mucho más eficiente en el manejo de recursos y en el tráfico de datos. En algunos artículos incluyen soluciones basadas en computación en la nube o la virtualización de funciones de la red.

Preg4: ¿En qué escenarios han sido implementados las soluciones propuestas en los artículos?

La mayoría de las redes de sensores programables son analizadas de una forma simulada, ya que según las soluciones planteadas representarían un alto consumo procesamiento y energía de los nodos sensores. Mientras que los WSN basadas en SDN es implementado en entorno físico, el cual analizan el comportamiento del flujo de datos generados por los nodos

Preg5: ¿Qué protocolos son incluidos en las soluciones?

Se identifican los siguientes protocolos: sensor openflow que es utilizado para arquitecturas centralizadas es decir un nodo sensor que actúa como un switch que simula las funciones del protocolo openflow; mientras que, en una solución WSN-SDN permiten incluir una variedad de protocolos entre ellos openflow, 6lowpan, COAP y MQTT.

Preg6: ¿Qué métricas se contemplan para validar la solución presentada en los artículos?

De los artículos analizados se pudo identificas las siguientes métricas que son utilizadas para evaluar las soluciones propuestas: latencia, throughput, ancho de banda, consumo de energía y control de flujo

Preg7: ¿Qué limitaciones existen al momento de integrar las dos tecnologías?

Al análisis las soluciones que presentan los diferentes artículos para integrar las dos tecnologías se pudieron identificar las siguientes limitaciones:

- Las redes WSN no admiten direcciones IP
- No se define un estándar de comunicación entre el plano de datos y control, para redes de sensores inalámbricos (Carece de detalles sobre el protocolo en dirección sur)

- Los protocolos de enrutamiento de las WSN son diferentes con respecto a una red IP que soporten el protocolo openflow
- Los campos de coincidencia dentro de una red SDN se lo hace en base en direcciones IP destino u origen
- Los dispositivos WSN al tener una reducida capacidad de procesamiento, no aceptan el protocolo openflow
- Los sistemas operativos destinados a las WSN no soportan la creación e instalación de reglas de flujo.

A.3.2. Síntesis de datos

En esta sección consiste en establecer parámetros que permitan evaluar y simplificar las respuestas obtenidas en las preguntas de investigación, para ello, se acude a las recomendaciones emitidas por la *International Telecommunication Union* (ITU) que define los elementos que compone una arquitectura de red tales como: dispositivos, protocolos, tipo de arquitectura y servicios.

A.3.3. Método de Evaluación

Es importante determinar los parámetros que permitan evaluar los datos obtenidos, en base a métodos cualitativos y cuantitativos, cada uno de estos parámetros se le asigna un valor de uno si cumple con los requisitos y cero si no cumple.

Para la evaluación cuantitativa se consideró los siguientes procesos:

- Se realizó un conteo de los artículos obtenidos mismos que fueron clasificados de acuerdo con los criterios establecidos.
- Se realizan gráficos estadísticos que permitan determinar la relación con los diferentes elementos que componente una red

Mientras que, para la evaluación cualitativa se determina la calidad de los artículos científicos obtenidos, para lo cual se establecen los siguientes parámetros, año de publicación y el número de citas. Además, se agregó preguntas subjetivas acordes al criterio de lector. La tabla 33 muestra un listado de pregunta que debe tomarse en cuenta para determinar la calidad del artículo.

Tabla 33.
Preguntas para determinar la calidad del contenido del artículo

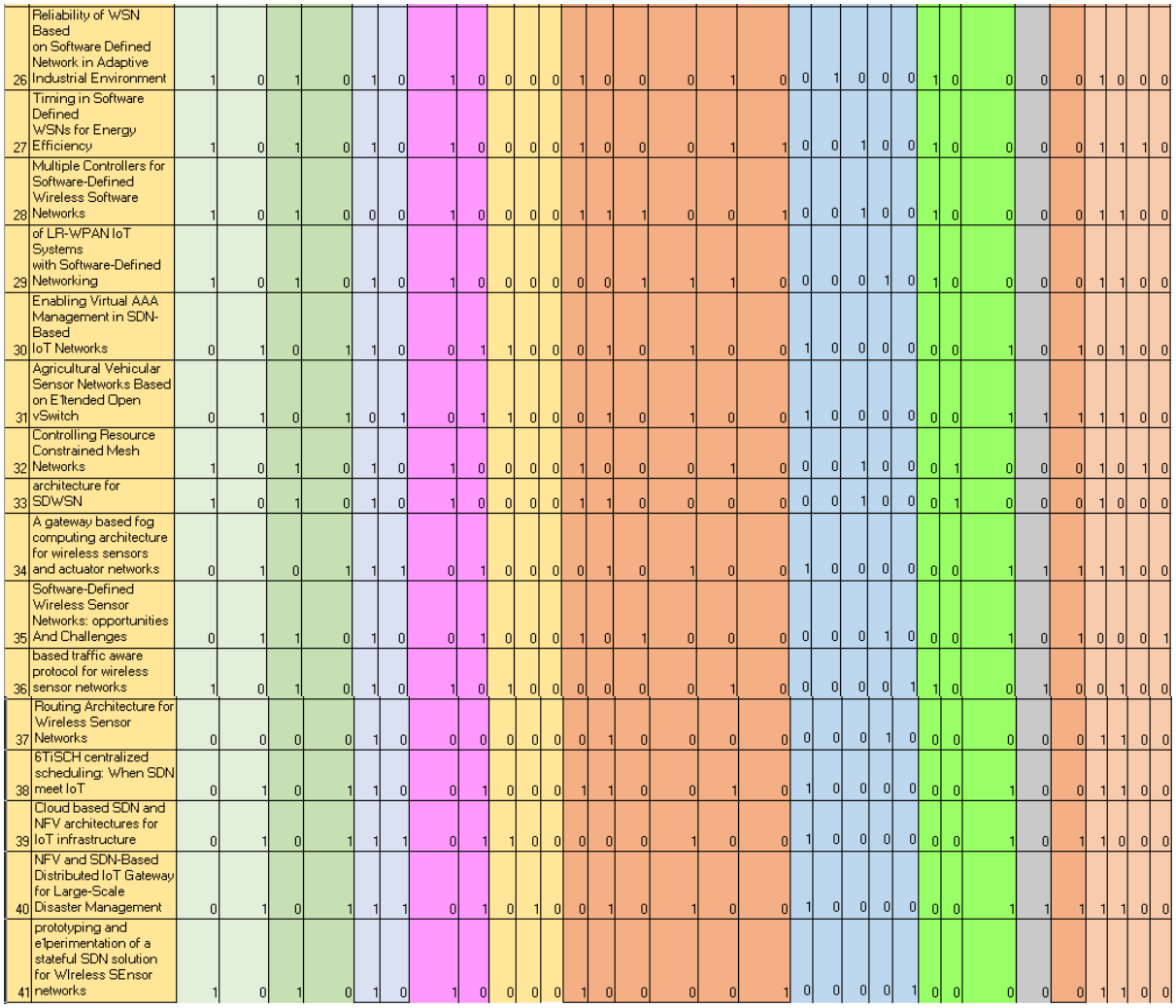
Nª	Preguntas
a.)	Las soluciones establecidas por los artículos contemplan las limitaciones de los dispositivos WSN
b.)	El artículo ofrece información clara sobre los requerimientos para integrar las redes de sensores inalámbricos y las redes definidas por software
c.)	Los artículos científicos obtenidos son citados en otros trabajos

Fuente: Elaborado por el autor

Tomando en cuenta el contexto anterior, en la figura 62 se realiza el proceso de síntesis de datos a partir de las preguntas expuestas en tabla 32.

	arquitectura según la ITU	Equipos		Protocolos									Servicios																			
		Nodos SDN Gateway controlador	Nodo simple Gateway-openswitch controlador snd	Capa Fisica/MAC	RPL IPv6 (6lowpan)	UDP	TCP	Middleware sensor openflow	Openflow	MQTT	COAP	Websocket	Enrutamiento	QoS		Firewall		Eficiencia en el uso de Recurso	SDWSN	Whisper	TINY-SDN	uSDN	SDN-WISE	Centralizada	Distribuida	Centralizado pero distribuido fisicamente)	Infraestructura como servicio	Virtualización de funciones de red	Eficiencia	Interoperabilidad	Escalabilidad	Seguridad
														Retardos	Perdidas de paquetes	Control de Flujo de datos	Control de topologia	Eficiencia en el consumo														
1	Software defined wireless sensor networks application opportunities for efficient network	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	0	0
2	SDN-Based Management Framework for IoT	0	1	0	1	0	1	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1	0	0	0
3	A System Architecture for Software-Defined Industrial Internet of Things	0	1	0	1	1	0	0	1	0	1	1	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	1	0	0	1	1
4	Publish/Subscribe-enabled Software Defined Networking for Efficient and Scalable IoT Communications	0	1	0	1	0	1	0	1	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0
5	Forwarding in 6LoWPANs through Software-Defined Networking	1	0	1	0	1	0	1	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0
6	A SOFTWARE-DEFINED SOLUTION FOR MANAGING FOG COMPUTING RESOURCES IN SENSOR NETWORKS	1	1	0	1	1	0	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	1	1	0

7	Protocols for full End-to-End Softwarization in IoT Networks	0	1	0	1	1	0	1	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	1	0	0	1	1	1	0
8	Architecture for Horizontal Internet of Things Services	1	1	0	0	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
9	Improving Security in Internet of Things with Software Defined Networking	0	1	0	1	1	0	0	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0	1	1	0
10	Transport SDN and Edge/Cloud computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network	0	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	1	1	0	1	0	1	0
11	Evolving SDN for Low-Power IoT Networks	1	0	0	1	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0
12	Wireless Sensor Networks: A Survey	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	1	1	1	1
13	Design and Implementation of SDN-based 6LBR with QoS Mechanism over Heterogeneous WSN and Internet	0	0	0	1	1	0	0	1	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	1	1	0	1	1	0	0
14	Networks and Software Defined Networks Integration	0	1	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0
15	IoT Devices using an SDN Gateway	1	0	1	0	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0	0	1	1	0
16	Topology Control Strategies for the Internet of Things	1	0	1	0	0	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	1	1	1	0
17	End-to-End Delay Enhancement in 6LoWPAN Testbed Using Programmable Network Concepts	1	0	0	1	1	0	0	1	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	1	1	0	1	0	0	0
18	Wireless Networking for Efficient Communication in Smart Cities	0	1	0	1	0	1	0	1	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0
19	SD-NFV as an Energy Efficient Approach for M2M Networks Using Cloud-Based 6LoWPAN Testbed	1	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	1	0	1	0	1	1	1	0
20	Services - SDN-enabled IMS Architecture for IoT Integration	0	1	0	1	0	1	0	1	1	1	0	1	1	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	1	1	0
21	Interfaces Virtualization: a Feasibility Study	0	1	0	1	1	0	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	1	1	0	0
22	IoT Framework with NVF Implementation	0	1	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	1	0	0
23	Software Defined Wireless Networks: Unbridling SDNs	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	0	0
24	IoT Ecosystem over programmable SDN infrastructure for smart city applications	0	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	0	0
25	Performance Analysis of Software-Defined Multihop Wireless Sensor Networks	1	0	1	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0



26	Reliability of WSN Based on Software Defined Network in Adaptive Industrial Environment	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0
27	Timing in Software Defined WSNs for Energy Efficiency	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	1	1	0
28	Multiple Controllers for Software-Defined Wireless Software Networks	1	0	1	0	0	0	1	0	0	0	0	1	1	1	0	0	1	0	0	1	0	0	1	0	0	0	0	1	1	0	0
29	of LR-WPAN IoT Systems with Software-Defined Networking	1	0	1	0	1	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0
30	Enabling Virtual AAA Management in SDN-Based IoT Networks	0	1	0	1	1	0	0	1	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0
31	Agricultural Vehicular Sensor Networks Based on E1tended Open vSwitch	0	1	0	1	0	1	0	1	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1	1	0	0
32	Controlling Resource Constrained Mesh Networks	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
33	architecture for SDWSN	1	0	1	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0
34	A gateway based fog computing architecture for wireless sensors and actuator networks	0	1	0	1	1	1	0	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1	1	0	0
35	Software-Defined Wireless Sensor Networks: opportunities And Challenges	0	1	1	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	1
36	based traffic aware protocol for wireless sensor networks	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0	1	0
37	Routing Architecture for Wireless Sensor Networks	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0
38	6TiSCH centralized scheduling: When SDN meet IoT	0	1	0	1	1	0	0	1	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0
39	Cloud based SDN and NFV architectures for IoT infrastructure	0	1	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0	0
40	NFV and SDN-Based Distributed IoT Gateway for Large-Scale Disaster Management	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1	1	0	0
41	prototyping and e1perimentation of a stateful SDN solution for WIreless SEnsor networks	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	0

Figura 62. Proceso de síntesis de datos

A.4. Fase de reportes

Luego de finalizar el proceso de ejecución que incluyo la valorización de los artículos y la extracción de datos, se realizan cuadros estadísticos con el propósito de analizar los resultados obtenidos sobre la integración de las redes de sensores inalámbricos y las redes definidas por software.

Este reporte va a mostrar los resultados obtenidos de la búsqueda de los artículos, para luego presentar la valorización de los artículos seleccionados y por último se muestran la síntesis de datos que se obtuvieron de las preguntas de investigación.

Dentro del proceso de extracción de datos también se contempló el año de publicación de los artículos con la finalidad obtener información actual y que aporte al desarrollo del proyecto por ende se selecciona los artículos que se publicaron entre el 2014 al 2020. La figura 63 indica la cantidad de estudios obtenidos luego de aplicar los criterios de exclusión e inclusión, además se observa que, el mayor número de publicaciones relacionadas sobre esta tendencia es a partir del año 2015 a 2016, entre el año 2016 y 2017, se percibe una ligera declinación para luego mantenerse estable en el año 2018 a 2019.

Figura 63. Número de artículos obtenidos de los diferentes repositorios

Otro punto que se consideró para valorizar el contenido de los artículos científicos son los resultados obtenidos de la tabla 33, según la figura 64 la primera pregunta evidencia que el 63% de soluciones incluyen de forma general las limitaciones de los dispositivos de las redes WSN, Un 20% realizan un análisis profundo sobre las limitaciones de las WSN, siendo la más renombrada el consumo de energía y el procesamiento de datos, por otro lado, 17% no consideran las limitaciones de las WSN en las soluciones.

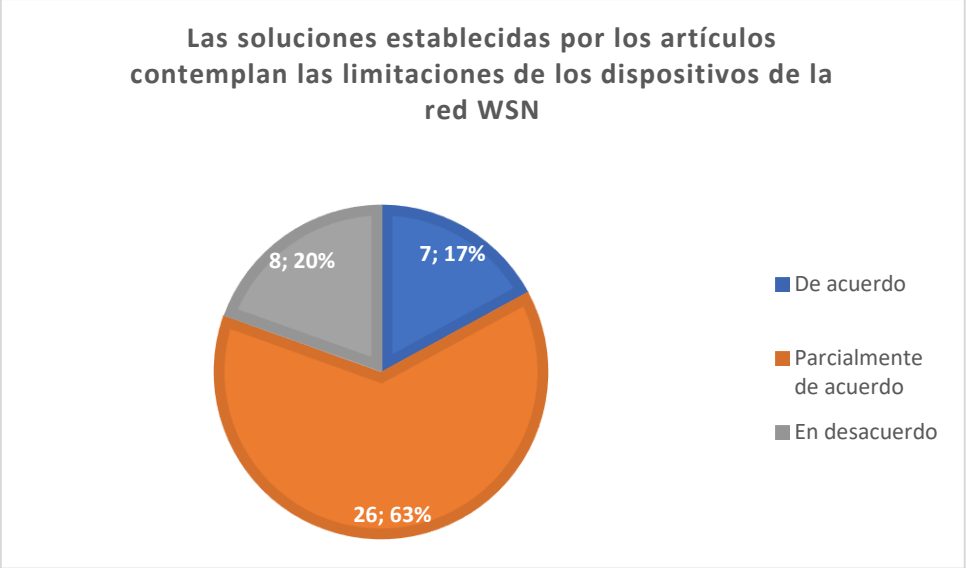


Figura 64. Resultados de la primera pregunta sobre las limitaciones de las WSN

La segunda pregunta de la tabla 33, permite analizar si los documentos proporcionan la información precisa para determinar los requerimientos, como resultado se obtuvo que un 64,70% de los artículos establecen requerimientos para el diseño de la WSN y SDN, mientras que el 20,58% representan a los artículos que aportan con teoría sobre las dos tecnologías y por último el 14,70% representa encuestas entre diferentes soluciones para integrar las dos tecnologías. En la figura 65 se resumen los resultados que se obtuvieron a cerca de la pregunta de la tabla 33.

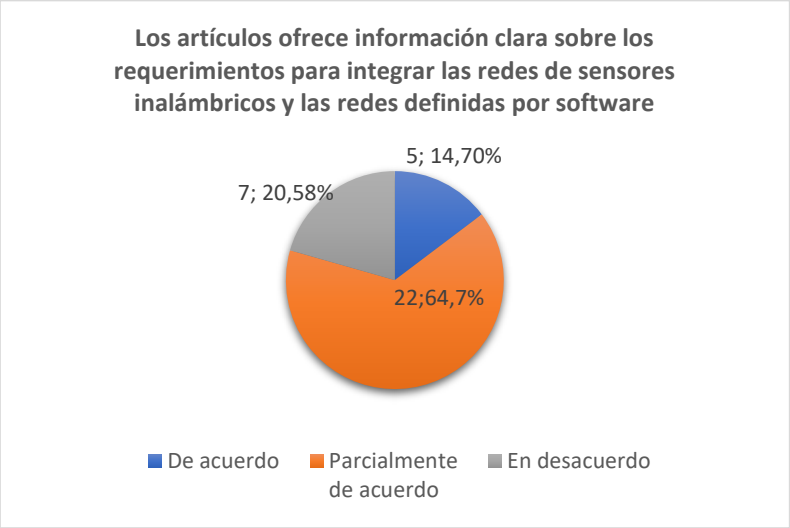


Figura 65. Resultados sobre los requerimientos para integrar las redes WSN y SDN

Finalmente, en la figura 66 se expresa los resultados de la pregunta “c.” que indica que existen más de 25 artículos (69,44%) citados en diferentes estudios, mientras que, una cantidad de 7 artículos (19,44%) son citados entre 20 a 50 ocasiones y únicamente 4 de los artículos (11,11%) son citados entre 1 a 20 artículos.

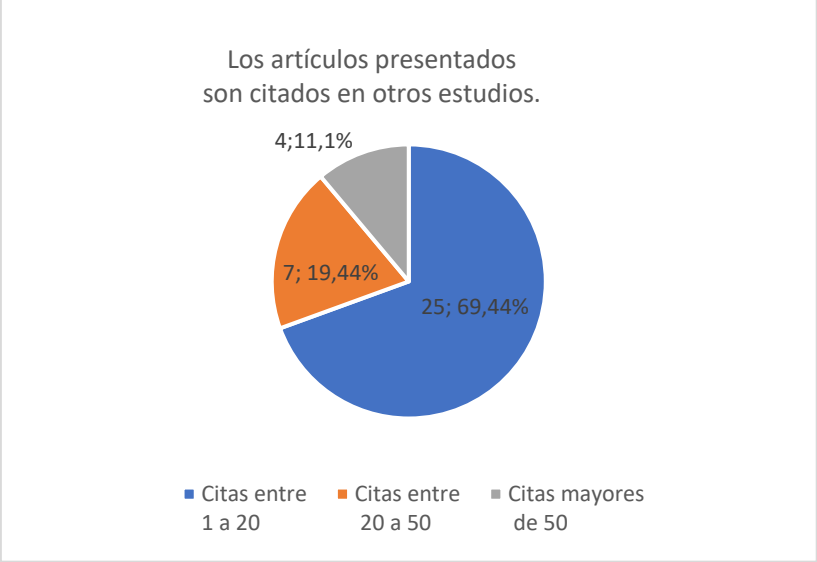


Figura 66. Resultado sobre los artículos presentados son citados en otros estudios

Análisis General

De los datos extraídos de las preguntas y la simplificación de datos se llega a la conclusión que existen dos soluciones: las rede de sensores programables y redes de sensores basadas en SDN. La figura 67 muestra las soluciones encontradas en los diferentes artículos en relación con las redes de sensores programables que representan valores bajos que van desde un 3% a 18%, mientras que el 56% adopta soluciones de redes de sensores enfocadas a la SDN.

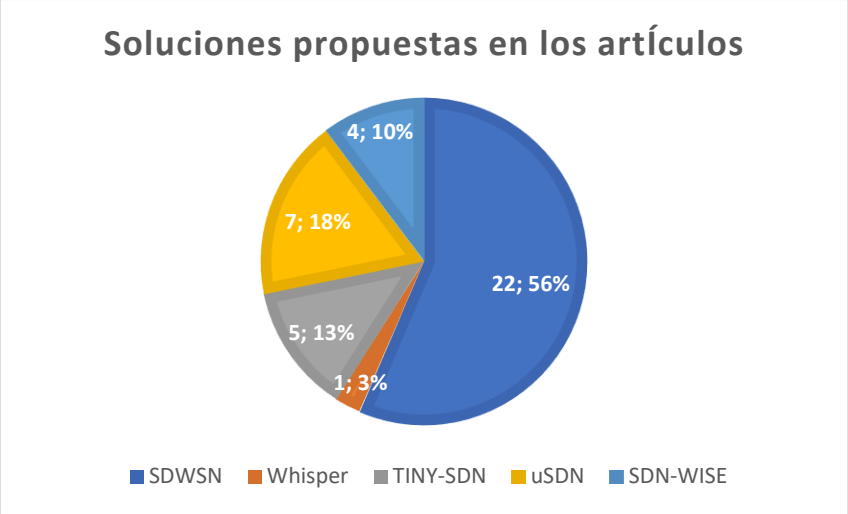


Figura 67. Tipo de propuesta para integrar las redes WSN y SDN

Al analizar cada una de las soluciones propuestas se puede identificar que las soluciones basadas en redes de sensores programables que se enfocan en modificar la capa física y MAC para adoptar las características del protocolo openflow, a diferencia de las redes WSN basadas en SDN que se concentran en controlar el flujo de datos, a través de las direcciones IP. La figura 68 representa la relación los dos enfoques, siendo que el 53,6% presenta soluciones basadas en direcciones IP y un 41,5% corresponden a soluciones de capa física y MAC

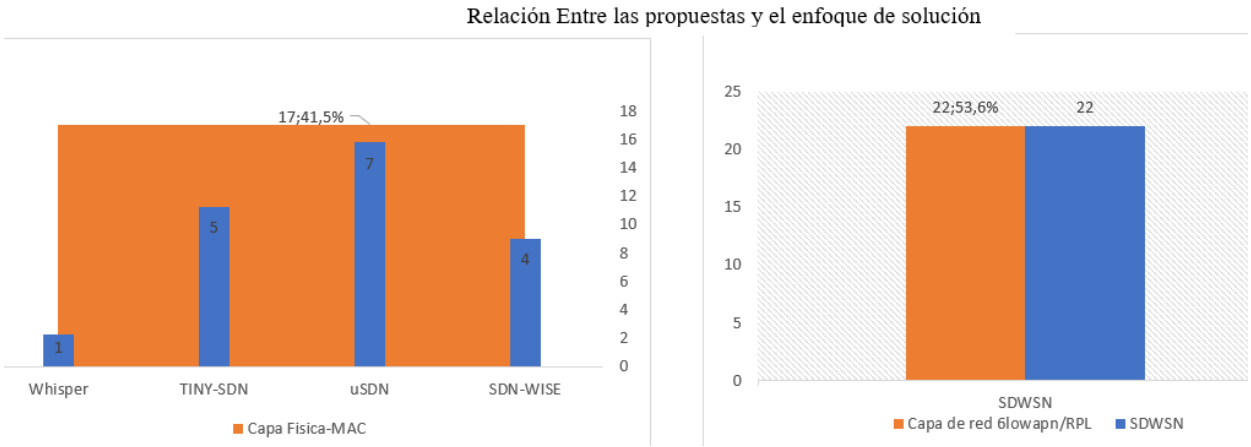


Figura 68. Comparación entre las propuestas y su enfoque de solución

De los resultados obtenidos, se puede identificar el tipo de arquitectura que se han, implementado en las diferentes soluciones, siendo la más utiliza la arquitectura hibrida que

alcanza un valor de 56%, debido a que relación el protocolo openflow con las redes WSN, sin afectar su limitada capacidad de procesamiento y memoria. Mientras que, la arquitectura centralizada alcanza un 22% el cual consiste en recrear las funciones del protocolo openflow en los nodos por medio del middleware denominado sensor openflow de esta manera se intenta tener un control centralizado de los nodos dentro de la red WSN. Por otro lado, la arquitectura distribuida intenta gestionar los nodos de forma general por medio de los protocolos de enrutamiento como protocolo RPL. En la figura 69 se muestra la comparación entre las diferentes modelos de arquitecturas

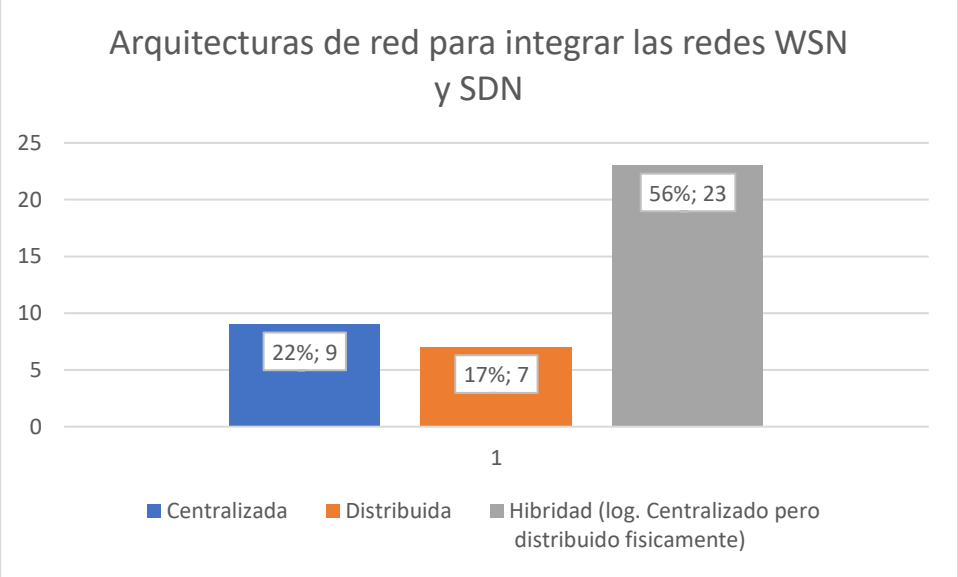


Figura 69. Modelos de arquitecturas implementados en las diferentes soluciones

Otro punto relevante que se puede identificar son los diferentes protocolos que se incluyen dentro de una arquitectura como: MQTT, COAP y Websocket. De la figura 70, se obtiene que la mayoría de los protocolos son aplicados en arquitecturas híbridas siendo el protocolo MQTT el más utilizado con el 34,14% seguido del protocolo COAP con 9.75% y por último, el protocolo Websocket con el 2,43%, mientras que el 53,65% representa soluciones que no incluye protocolos adicionales.

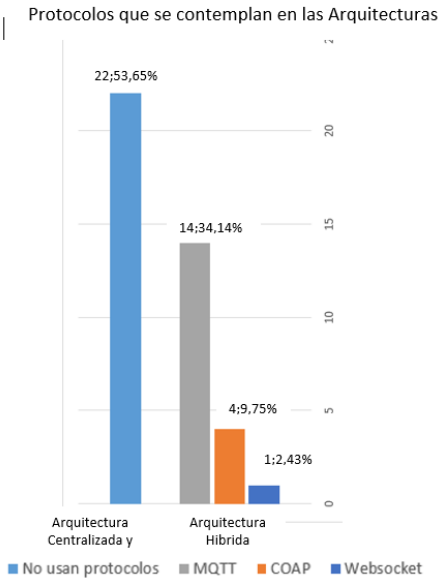


Figura 70. Protocolos que se incluyen en las arquitecturas

De la figura 71, se exponen los diferentes aplicaciones que se utilizaron para validar las diferentes soluciones donde el 51,29% de las soluciones se enfocan en mejorar el enrutamiento entre nodos, el 46,34% se concentró en controlar el flujo de datos por medio de las direcciones IP para medir el tiempo de respuesta entre nodos y servidor o para calcular las pérdidas de paquetes, por último, el 24,39% se enfoca desarrollar aplicaciones para optimizar el consumo de energía y control de topología.

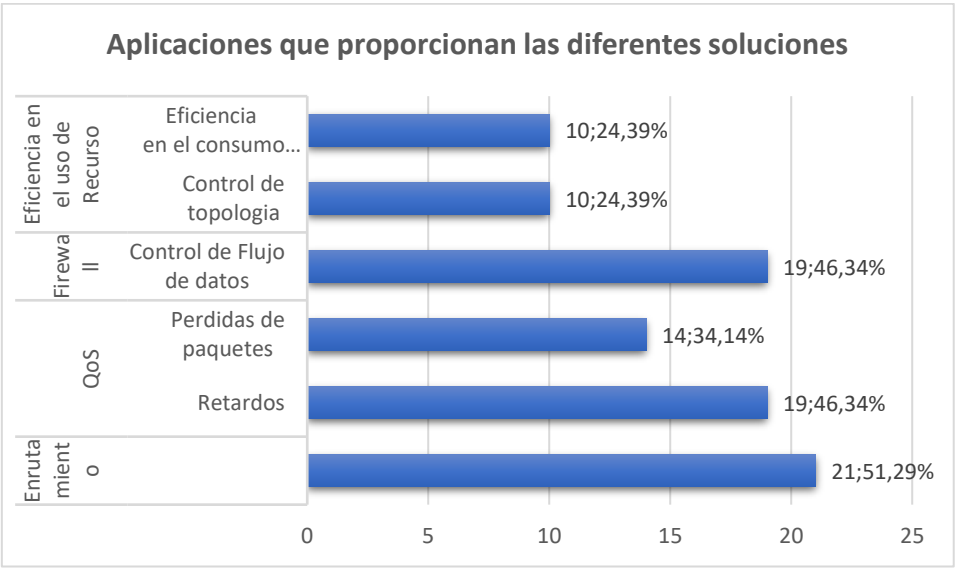


Figura 71. Aplicaciones que proporcionan las diferentes soluciones

Fuente: Elaborado por el autor

Anexo B. Instalación y configuración de los componentes de red

B.1. Implementación y configuración del controlador RYU

El controlador RYU se encuentra disponible en el repositorio github, para su instalación se escoge el sistema operativo ubuntu server en su 18.04.4 LTS que se alojara en una máquina virtual. A continuación, se muestra los pasos de instalación.

1. Se inicia con la actualización de los repositorios ubuntu server 18.04.4 LTS con los siguientes comandos.

   apt-get update

   apt-get upgrade

2. Es necesario instalar todas las dependencias Python con el propósito de evitar errores al momento de ejecutar el controlador RYU, para lo cual debe ejecutar los siguientes comandos:

   sudo apt-get install python-setuptools python-pip -y

   sudo apt-get install python-eventlet python-routes python-webob python-paramiko -y

3. Una vez instalado los requerimientos se procese a clonar el controlador Ryu directamente de los repositorios de github mediante el comando: *git clone git://github.com/osrg/ryu.git*

4. al terminar la descarga se ingresa al directorio ryu y ejecutamos el siguiente comando *python ./setup.py* install para finalizar la instalación.

5. Por último, se verifica el funcionamiento del controlador como se detalla en la figura 72

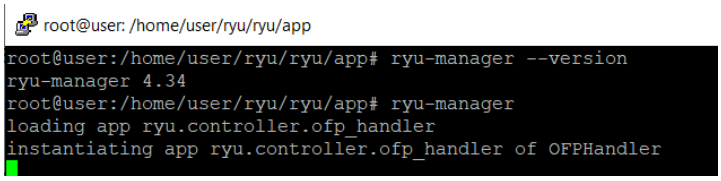


Figura 72. Funcionamiento del controlador Ryu

Autor: Elaborado por el autor

B.2. Configuración de la interfaz de red del controlador Ryu

Para asignar la dirección IP estática al controlador Ryu de acuerdo a las especificaciones de la tabla 26 se debe configurar el fichero 50-cloud-init.yaml ubicado cd /etc/netplan como se muestra en la figura 73.

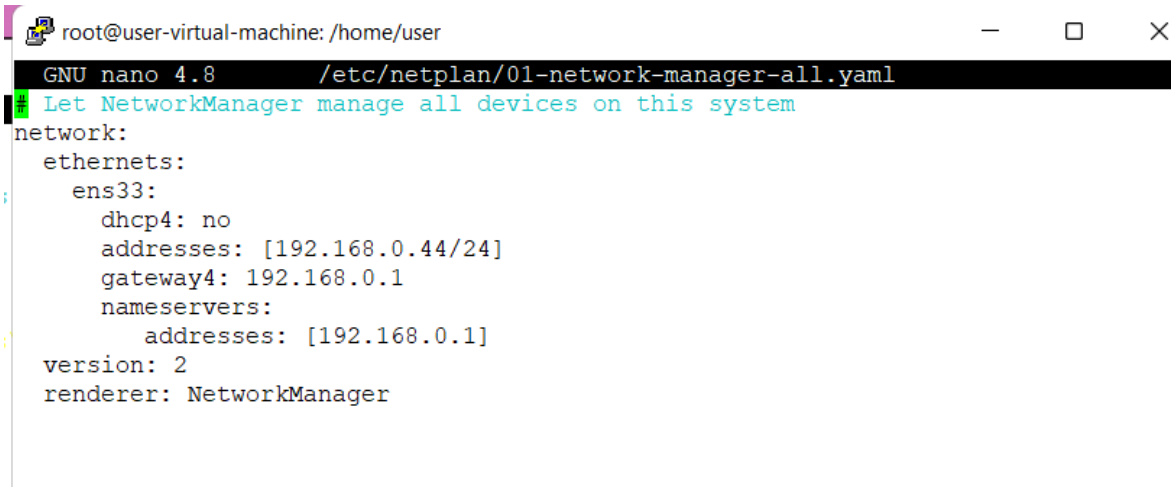


Figura 73. Configuración de la interfaz de red del controlador Ryu

Autor: Elaborado por el autor

B.3. Implementación del conmutador open Vswitch en raspberrry pi model B+

Para empezar, se inicia descargando la versión lite del sistema operativo (OS) Raspbian del siguiente link https://www.raspberrypi.org/software/operating-systems/, una vez descargado se descomprime el archivo para obtener la imagen de OS, luego se procederá a la escritura de la tarjeta microSD por medio del software balenaEtcher que se muestra en la figura 74. Cabe mencionar que la tarjeta microSD debe tener las siguientes características: clase 10 y una capacidad de 32 GB, para evitar problemas en el arranque del OS dentro del dispositivo raspberry pi.

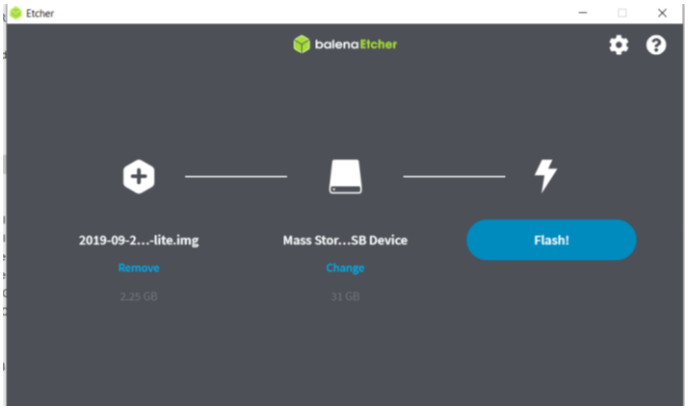


Figura 74. Proceso de grabado del SO en microSD

Autor: Elaborado por el autor.

Al terminar el proceso de grabado se inserta la tarjeta microSD en la placa raspberry pi para luego conecta los dispositivos periféricos y la fuente de alimentación. Al momento de encender el dispositivo debe ingresar las credenciales por defecto los mismos que son: *username = pi, password = raspberry.* A continuación, se detalla el proceso a instalación de open vswitch en la placa.

1. Para iniciar con la instalación de open vswitch en raspbian es necesario ingresar como superusuario para realizar las actualizaciones de los repositorios del OS con los siguientes comandos:

   apt-get update
   apt-get upgrade

2. Al terminar las actualizaciones, nos dirigimos al siguiente link https://www.openvswitch.org/download/ con la finalidad de descargar los paquetes de instalación de open vswitch a través del comando que se observa en la figura 75.

   Wget https://www.openvswitch.org/releases/openvswitch-2.7.0.tar.gz

Figura 75. Instalación de open vswitch

Autor: Elaborado por el autor

Al tener el archivo en formato tar.gz se descomprime mediante el comando *tar -xvzf openvswitch-2.7.0.tar.gz.*

3. Para que se ejecute correctamente open vswitch es necesario la instalación de dependencias de Python y es posible con el siguiente comando:

   apt-get install python-simplejson python-qt4 python-twisted-conch automake autoconf gcc uml-utilities libtool build-essential pkg-config.

4. Para que el conmutador virtual interactúe con el kernel del sistema operativo de la placa raspberry pi es necesario buscar encabezado del kernel correcto mediante el comando para posteriormente ser instalado como se detalla figura 76.

   apt-cache search linux-headers

   apt-get install linux-headers-4.9.0-6-rpi

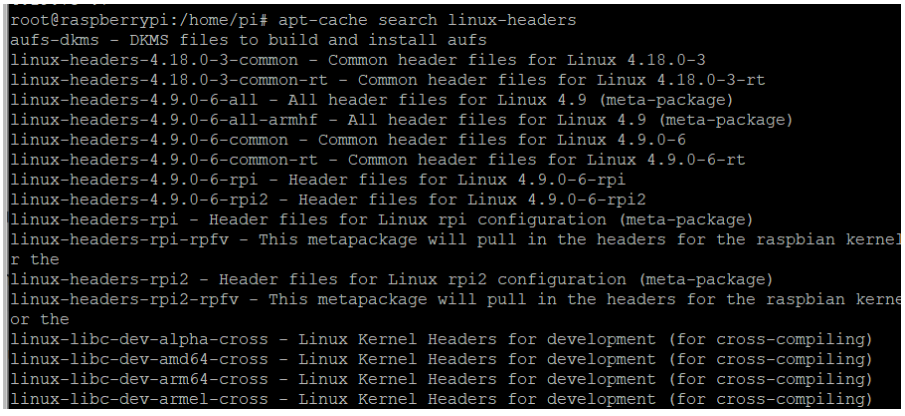


Figura 76. Instalación del encabezado del kernel

Autor: Elaborado por el autor

a. En la figura 77 se resumen el proceso para construir el open vswitch el cual consiste en ingresar al directorio cd opemvswitch-2.7.0 y ejecutar los comandos:

./boot.sh

./configure --with-linux=/lib/modules/4.9.0-6-rpi/build

Make

make install

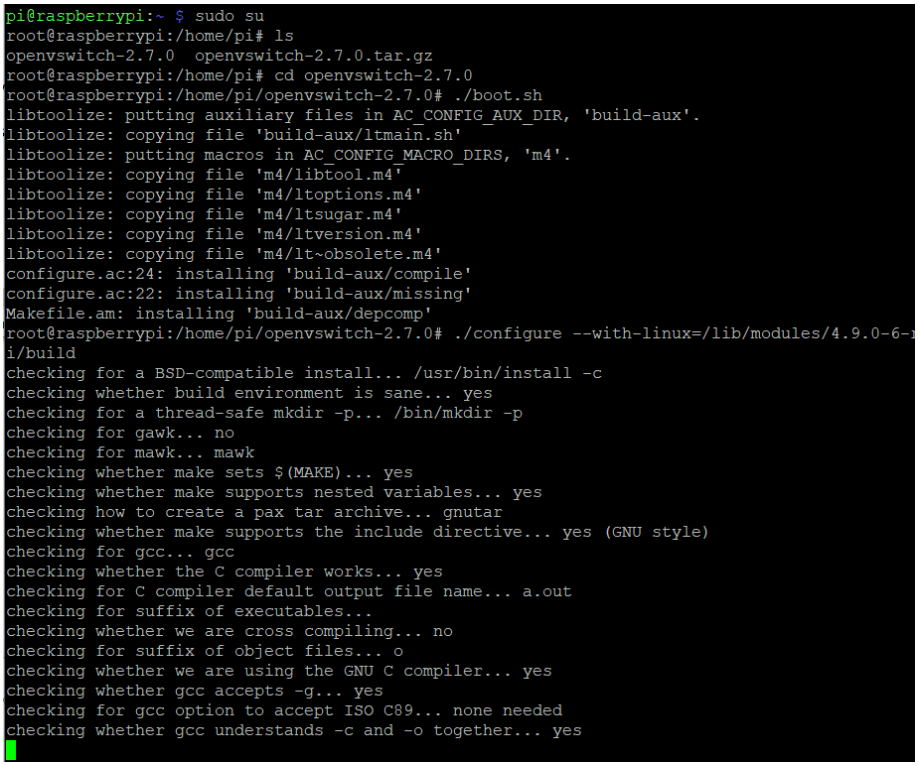


Figura 77. Proceso de construcción del conmutador open Vswitch

Autor: Elaborado por el autor.

5. Si el proceso de construcción del open vswitch no presento ningún error diríjase a la ruta datapath como muestra en la figura 78, para cargar los módulos. Esto se realiza con los comandos

cd datapath/Linux

modprobe openvswitch

cat /etc/modules

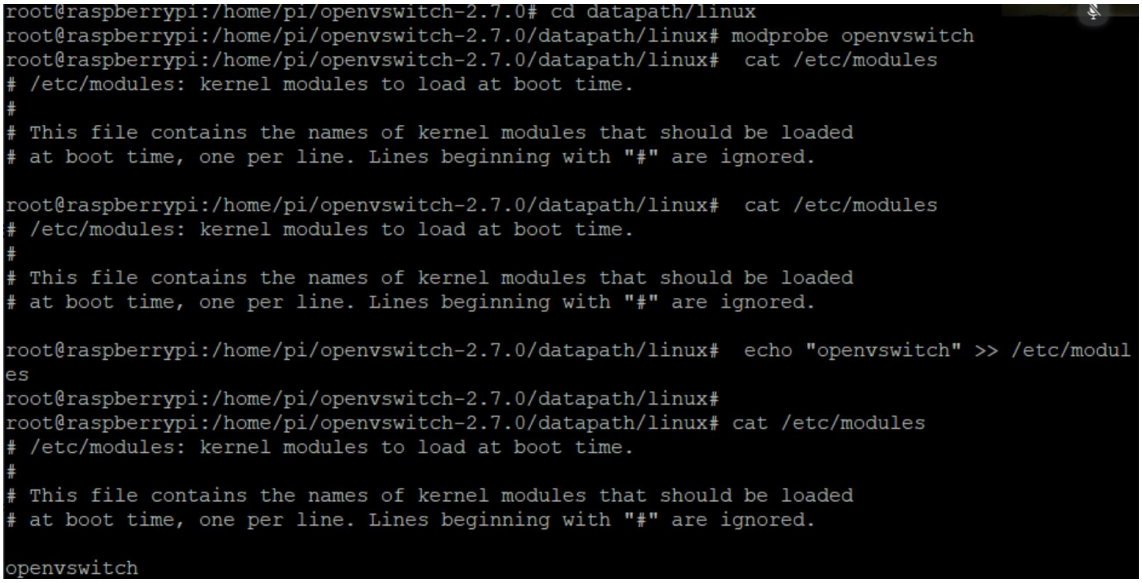


Figura 78. Instalación de los módulos de open vswitch al kernel del Linux

Autor: Elaborado por el Autor

6. Para finalizar el proceso de instalación se crea dos archivos: *osv-vswitchd y ovsdb-server*; el *ovs-vswitch* es un proceso que se ejecuta en segundo plano y permite que el dispositivo actúe con las funcionalidades de un conmutador SDN; *ovsdb-server* guarda toda la información de configuración del open Vswitch. En la figura 79, se muestra el proceso de creación de los archivos con los siguientes comandos:

   touch /usr/local/etc/ovs-vswitchd.conf

   mkdir -p /usr/local/etc/openvswitch

   ovsdb-tool create /usr/local/etc/openvswitch/conf.db vswitchd/vswitch.ovsschema

Figura 79. Creación de los archivos de configuración del open Vswitch

Autor: Elaborado por el autor.

7. Para establecer comunicaciones entre OVS y con cualquier administrador a través de SSL es necesario copiar el código que se expresa en script sw_start.sh de la figura 80.

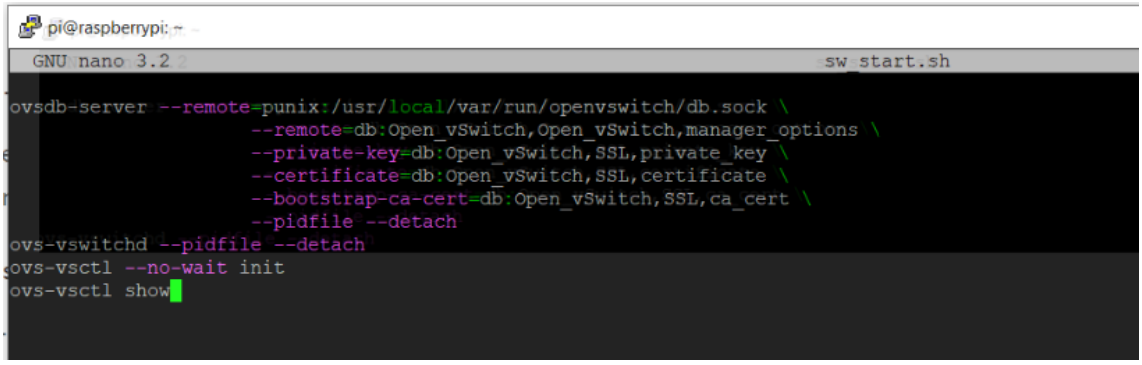


Figura 80. Script de inicio del conmutador OVS

Autor: Elaborado por el autor.

Par que el script pueda ejecutarse correctamente debe darse permisos mediante el comando *chmod +x sw_start.sh*, por otro lado, para que el script arranque junto con el raspberry se debe añadir la dirección del script en el archivo de configuración */etc/rc.local/* como se muestra en la figura 81.

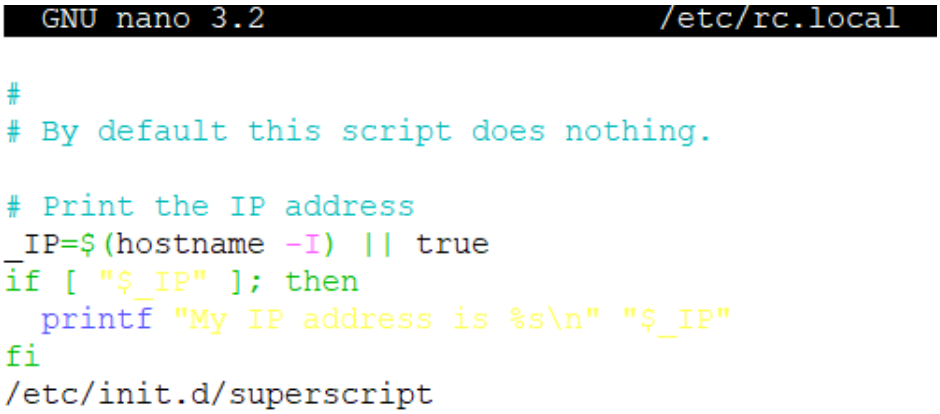


Figura 81. Configuración del archivo "rc.local"

Autor: elaborado por el autor.

8. Por último, en la figura 82 se verifica la versión y si la instalación fue correcta del conmutador open Vswitch.

   ovs-vsctl –versión

   ps -e | grep ovs

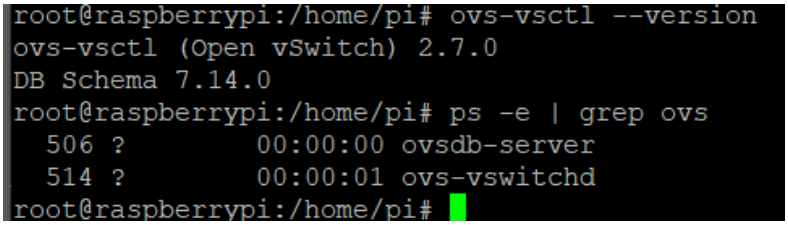


Figura 82. Verificación de instalación de open vswitch

Autor: elaborado por el autor

B.4. Configuración de los adaptadores de red en raspberry pi model B+.

De acuerdo con la figura 86, se requieren conectar tres adaptadores USB-ethernet en el dispositivo raspberry pi, para verificar si las interfaces fueron reconocidas por el dispositivo se ejecuta el comando *ifconfig* y debería desplegarse una lista de interfaces habilitadas como se muestra en la figura 83.

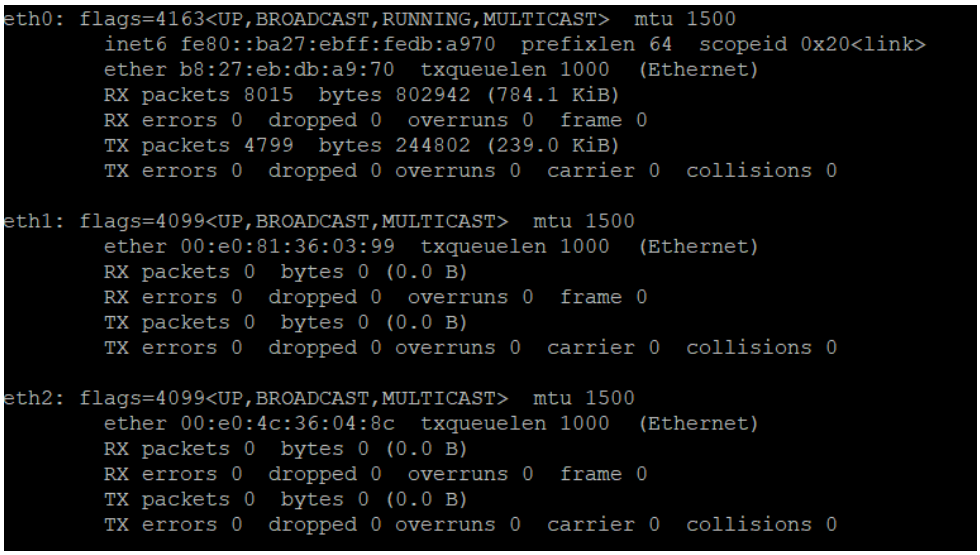


Figura 83. Lista de interfaces habilitadas en el OVS

Autor: elaborado por el autor

El siguiente paso de configuración es que las interfaces eth1 y eth2 deben tener una dirección IP nula para poder ser agregados como puertos del conmutador open Vswitch para esto debe introducir los siguientes comando en el archivo de configuración *rc.local*, *sudo ifconfig eth1 0*

Sudo ifconfig eth2 0

B.4.1. Configuración del open vswitch en la placa raspberry pi model B+

Para configurar el open Vswitch se debe crear una interfaz virtual para posteriormente asociarle a la interfaz eth0.

1. Se crea una interfaz virtual que cumple con la función de puente entre el open Vswitch y el dispositivo a través del comando *sudo ovs-vsctl add-br bridge*, por otro lado, para que la interfaz bridge tenga acceso a internet se debe asignar la dirección IP, subred y el Gateway de la interfaz eth0 como se muestra en la figura 84.

```
sudo ifconfig bridge 192.168.0.32 netmask 255.255.255.0
sudo route add default gw 192.168.0.1 bridge
```

Figura 84. Configuración de la interfaz bridge.

Autor: Elaborado por el autor.

2. Mediante el comando sudo ovs-vsctl add-port bridge se agregan las interfaces eth1 y eth2 al comuntador openflow (bridge)
3. Se debe configura un identificador para el Data Path del conmutador y la versión del protocolo opemflow que se va a implementar esto es posible mediante los comandos:

   sudo ovs-vsctl set bridge bridge other-config:datapath-id=00000000001

   sudo ovs-vsctl set bridge bridge protocols= Openflow1
4. Por último, se establece una comunicación TCP/IP en el puerto 6633 entre el controlador y el conmutador además se desactiva el modo de pruebas de fallas donde el conmutador openflow no puede enviar ningún paquete a menos que lo autorice el controlador.