VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

MAESTRÍA EN GERENCIA DE SISTEMAS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN: GERENCIA DE SISTEMAS

TEMA: ANÁLISIS COMPARATIVO DE DESEMPEÑO ENTRE PROTOCOLOS MQTT Y COAP PARA INTERNET DE LAS COSAS (IoT) CON RASPBERRY PI 3 EN AMBIENTES IEEE 802.11

AUTORA: GUAMÁN OÑA, YESENIA CECIBEL

DIRECTOR: ING. SALAZAR CHACÓN, GUSTAVO DAVID

SANGOLQUÍ

2019

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"*ANÁLISIS COMPARATIVO DE DESEMPEÑO ENTRE PROTOCOLOS MQTT Y COAP PARA INTERNET DE LAS COSAS (IOT) CON RASPBERRY PI 3 EN AMBIENTES IEEE 802.11*"**, fue realizado por la señorita ***Guamán Oña, Yesenia Cecibel***; el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 13 de agosto del 2019.

Firma

Ing. Salazar Chacón, Gustavo David

C.C.: 1716104797

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

AUTORÍA DE RESPONSABILIDAD

Yo, ***Guamán Oña, Yesenia Cecibel***, con cédula de ciudadanía N° 1722662457, declaro que el contenido, ideas y criterios del trabajo de titulación: ***Análisis comparativo de desempeño entre protocolos MQTT y CoAP para internet de las cosas (IoT) con raspberry pi 3 en ambientes IEEE 802.11***, es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de la Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 13 de agosto del 2019.

Firma

Guamán Oña, Yesenia Cecibel

C.C.: 1722662457

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

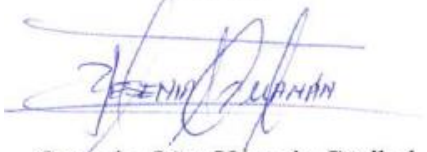
CENTRO DE POSGRADOS

AUTORIZACIÓN

Yo, ***Guamán Oña, Yesenia Cecibel***, autorizo a la Universidad de las Fuerzas Armadas – ESPE publicar el trabajo de titulación: ***Análisis comparativo de desempeño entre protocolos MQTT Y CoAP para internet de las cosas (IoT) con raspberry pi 3 en ambientes IEEE 802.11***, en el repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 13 de agosto del 2019.

Firma



Guamán Oña, Yesenia Cecibel

C.C.: 1722662457

DEDICATORIA

Dedico este trabajo a mis padres Marco y Eufemia por ser mi soporte en cada paso que decido emprender, por ser quienes me ayudan con su guía y consejos, gracias a su experiencia y amor. Sin ustedes no hubiese podido lograr obtener lo que ahora gracias a Dios y a ustedes lo obtengo. Con todo mi corazón, respeto y amor este trabajo es para ustedes mis Papás.

Dedico también a mis hermanos David y Camila, porque sin los momentos compartidos con ustedes, este camino sería mucho más difícil; porque entre risas y enojos son ustedes los que animaron en aquellos momentos de stress y agotamiento. Con el corazón lleno de amor por ser el regalo que Dios me dio, dedico este logro a ustedes mis hermanos. Tengo la convicción que mi familia es mi mayor tesoro y el impulso para lograr mis metas.

Finalmente dedico mi trabajo a quien estoy segura que, desde el cielo me guió e intercedió por mí y lo va seguir haciendo hasta que nos volvamos a encontrar; a ti querido primo que mientras más pasa el tiempo más te recuerdo. A ti Hugo Arturo nuestro “Chinito” te dedico mi trabajo, sé que siempre confiaste en mí y me quisiste tanto como yo a ti.

CON AMOR YESENIA.

AGRADECIMIENTO

Gracias mi Dios por hacer que cada acontecimiento de mi vida sea el correcto y en el tiempo adecuado, para mi bien. Gracias Mamita del Cielo, porque sé que con el inmenso amor de madre me ayudas e iluminas día con día. Gracias porque escucharon mis oraciones cuando pensé que no lo iba a lograr.

Gracias nuevamente a mis padres y hermanos por no dejarme sola, por apoyarme incondicional y desinteresadamente, inclusive soportando muchas mis cambios de carácter.

Gracias a la mejor tía, tía María, te agradezco inmensamente que hayas estado y sigas estando en mis momentos más importantes, gracias por tus consejos, tus palabras y tantas muestras de cariño que me han animado a seguir adelante.

Gracias a la persona que pese a las dificultades ha permanecido a mi lado incondicionalmente, que me ha brindado su apoyo y comprensión en aquellos momentos que hemos tenido dejar todo de lado, por nuestras obligaciones y así lograr nuestras metas. Gracias Tavito por tu amor y por tu comprensión, gracias por estar pendiente siempre de mí.

Finalmente, gracias un gran profesional que aportó de gran manera al desarrollo de este proyecto, que sin su ayuda y a la predisposición de ayudarme, este trabajo no tendría el resultado que hoy tiene. Gracias Ing. Gustavo Salazar.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

IoT (Internet of Things-Internet de las cosas), tecnología con la que los objetos serán fabricados, con ciertas cualidades que permitan la conexión a Internet, almacenamiento de datos, e interacción con el usuario. La problemática se genera con el crecimiento exponencial de objetos y de usuarios, ya que, al ser un tema en pleno desarrollo existen varios protocolos de comunicación en lanzamiento, para que los objetos se comprendan entre sí y a su vez intercambiar datos. Con ello, se evidencia que, mientras más protocolos se desarrollen mayor será la dificultad de establecer su estandarización. Por ello, nace la propuesta del presente proyecto que implica el estudio de dos protocolos semejantes entre sí y que más se usan. En este caso los protocolos son MQTT y CoAP, con el fin de determinar cuál tiene un mejor desempeño y las aplicaciones a las que se direccionan, las pruebas se realizaron en un ambiente Wireless 802.11, con un prototipo básico para el hogar como el control on / off de una lámpara. La metodología que se utilizó es de tipo experimental – cualitativa, se basa en mediciones, puesto que a partir de un mismo prototipo se realizan comparaciones de los factores de desempeño de cada protocolo, con herramientas, como wireshark y Jperf. Según el análisis de los resultados, se tiene que, MQTT tiene mejor un desempeño que CoAP, por su implementación sencilla, y se orienta a varios tipos de aplicaciones; mientras que CoAP mantiene una brecha de desactualización y sus aplicaciones son limitadas.

PALABRAS CLAVE:

- **INTERNET DE LAS COSAS**
- **ESTANDARIZACIÓN DE PROTOCOLOS IOT**
- **MQTT Y COAP**
- **DESAFÍOS DE IOT**

ABSTRACT

IoT (Internet of Things-Internet of things), technology with which objects will be manufactured, with certain qualities that allow Internet connection, data storage, and user interaction. The problem is generated with the exponential growth of objects and users, since, being a topic in full development there are several communication protocols in launch, so that the objects understand each other and in turn exchange data. With this, it is evident that, the more protocols are developed, the greater the difficulty of establishing its standardization. Therefore, the proposal for this project is born, which involves the study of two protocols that are similar to each other and that are most used. In this case the protocols are MQTT and CoAP, in order to determine which has a better performance and the applications to which they are addressed, the tests were performed in a Wireless 802.11 environment, with a basic prototype for the home as the control on / off of a lamp. The methodology used is experimental - qualitative, based on measurements, since from the same prototype comparisons are made of the performance factors of each protocol, with tools such as wireshark and Jperf. According to the analysis of the results, it is necessary that, MQTT has a better performance than CoAP, for its simple implementation, and it is oriented to several types of applications; while CoAP maintains an outdating gap and its applications are limited.

KEYWORDS:

- **INTERNET OF THINGS**
- **STANDARDIZATION OF PROTOCOLS IN IOT**
- **MQTT Y COAP**
- **IoT CHALLENGES**

GLOSARIO

A

ACK Acknowledgement o acuse de recibo, en el área de comunicaciones y tecnología.

AD-HOC Redes inalámbricas descentralizadas. Los nodos encaminan una ruta por el reenvío de datos. No depende de la infraestructura existente.

ALG (Application Layer Gateway - Software gestor de aplicaciones) Componente de software que gestiona protocolos y aplicaciones.

AMQP (Advanced Message Quering Protocol - Protocolo de estándar abierto en la capa de aplicaciones de un sistema de comunicación)

B

Big Data Grandes volúmenes de datos, término que describe la gran cantidad de datos extraídos para obtener información.

Bluetooth Protocolo de comunicaciones, para la trasmisión inalámbrica de datos, entre distintos dispositivos que se encuentran a una corta distancia aproximadamente 10 metros.

C

Cf Californium, proporciona un marco central con la implementación de protocolo para aplicaciones de IoT.

CoAP (Constrained Application Protocol - Protocolo de aplicación restringida) Protocolo a nivel de aplicación que se usa para dispositivos electrónicos simples para comunicarse a través de Internet.

CON Mensaje de confirmación, mensaje con el que requiere una respuesta, ya sea afirmativa o negativa hasta agotar los intentos.

COPPER Agente de usuario de Firefox, que sirve para manejar el esquema URI de CoAP.

D

DDS (Data Distribution Service - Servicio de distribución de datos) Se usa para comunicaciones machine to machine, en un grupo de gestión de objetos para sistemas de tiempo real.

Delay Tiempo de retraso en la transmisión de una cantidad de datos a través de una red.

DHCP (Dynamic Host Configuration Protocol - Protocolo de configuración dinámica de equipos) Protocolo de red de tipo cliente / servidor, en el que asigna de forma dinámica una dirección IP a un dispositivo.

DTLS (Datagram Transport Layer Security - Seguridad en capa de transporte) protocolo de comunicaciones diseñado para proteger la privacidad, prevenir intercepción y manipulación

DUP Bandera de duplicación en protocolo MQTT.

E

E-tag Respuesta que indica el valor actual de la entidad o tag, para la representación del destino.

G

GPIO (General Purpose Input Output - Propósitos generales de entradas y salidas) Pin genérico dentro de chip, su comportamiento se puede controlar por medio del usuario.

H

HDMI (High Definition Multimedia Interface - Interfaz multimedia de alta definición) Es un tipo de conexión muy común, ya que permite la conexión de diversos aparatos de alta definición.

HTTP (Hypertext Transfer Protocol - Protocolo de transferencia de Hipertexto) protocolo de transferencia de hipertextos, se usa en algunas direcciones de Internet.

I

IBM International Business Machines Corporation, reconocida empresa multinacional estadounidense de tecnología y servicios de infraestructura.

IoT (Internet of Things-Internet de las cosas) Sistemas que se compone varios dispositivos, máquinas y objetos para transferir datos a través de una red.

IP (Internet Protocol - Protocolo de Internet) es una etiqueta numérica que identifica lógicamente a un dispositivo conectado a una red.

J

Jitter Es la fluctuación del delay o retardo a la variabilidad temporal durante el envío de señales digitales.

JPERF Programa cliente / servidor que permite realizar mediciones de ancho de banda, delay y jitter de una red.

K

Kbps Velocidad en el orden de los kilobytes por segundo.

L

LAN (Local Area Network - Red de área local) red que conecta equipos en un área relativamente pequeña y predeterminada.

LPDDR 2 (Low Power Double Data Rate Synchronous Dynamic Random Access Memory - Memoria de acceso aleatorio síncrono de doble velocidad de datos y baja potencia) Tipo de memoria de acceso aleatorio tiene bajo consumo de energía y se destina a equipos móviles.

M

M2M (Machine to machine - máquina a máquina) Se refiere a la comunicación entre máquinas, y éstas se pueden comunicar a un servidor por internet.

Max-Age Tiempo de respuesta en caché del cliente para esperar la respuesta del servidor.

Mbps Velocidad en megabytes por segundos.

MQTT (Message Queue Telemetry Transport – Transporte de telemetría de cola de mensajes) Es un protocolo de telemetría que corresponde a la capa de aplicación en la arquitectura de IoT.

N

NFC (Near Field Communication - Comunicación de campo cercano) Tecnología de comunicación inalámbrica, corto alcance y alta frecuencia.

NODEJS	Software de código abierto de JavaScript, sirve para crear aplicaciones web altamente optimizadas.
NODE-RED	Herramienta de desarrollo que se basa en flujos para programación visual creada por IBM, para aplicaciones IoT, usa un navegador web y se crean funciones de JavaScript.
NON	Mensaje No - Confirmable, se envía este mensaje cuando no es obligatorio que el receptor reciba el mensaje y por eso no necesita de confirmación.

O

OPC-UA	(Unified Architecture – Arquitectura Unificada) proporciona una interfaz estándar para la comunicación con PLC
OSI	(Open System Interconnection - Interconexión abierta de sistemas) modelo de referencia para protocolos de la red, no es arquitectura.

P

PATH	Es el camino o ruta para referencias un archivo o directorio.
PLC	PLC (Programmable Logic Controller – Controlador Lógico Programable) Dispositivo electrónico que se programa para realizar acciones de control automático.
PoE	(Power over Ethernet - Energía sobre ethernet) Es una tecnología que incorpora la alimentación eléctrica a una infraestructura LAN estándar.

Q

QoS	(Quality of Service - Calidad de servicio) Permite dar prioridad al tráfico de datos.

R

RFID (Radio Frecuency Indentification - Indentificación por radiofrecuencia) Sistema de almacenamiento y recuperación de datos mediante etiquetas RFID.

RSA (Rivest, Shamir y Adleman) Sistema criptográfico de clave pública desarrollado en 1979.

RST Código usado para describir la calidad de transmisiones.

S

SDRAM (Synchronous Dynamic Random Access Memory - Memoria de acceso aleatorio, dinámica y sincrónica) Familia de memorias dinámicas de acceso aleatorio.

SSH (Secure Shell - Cubierta segura) Protocolo que facilita las comunicaciones seguras entre dos sistemas con arquitectura cliente / servidor.

T

TCP (Transmisión Control Protocol - Protocolo de transmisión de control) Protocolo fundamental

TOKEN Serie especial de bits, se envía a cada equipo de forma secuencial.

U

UDT (User Data Type -Tipos de datos de usuario) Estructuras de datos creados por el usuario, se pueden usar en todo el programa de usuario.

URI (Uniform Resource Identifier - Identificador uniforme de recursos) Sirve para identificar recursos en Internet.

URL	(Uniform Resource Locator - Localizador Uniforme de Recursos) Secuencia de caracteres que sigue un estándar que, permite denominar recursos dentro de Internet y así localizarlos.

W

WiFi	Tecnología que permite la conexión inalámbrica de dispositivos electrónicos.
WiMax	(Worlwide Interoperability for Microwave Access - Interoperabilidad mundial para acceso por microondas) Nueva tecnología de Internet inalámbrico de largo alcance.
WLAN	(Wireless Local Area Network - Red de Área Local Inalámbrica) Red de tipo local en la que sus equipos no necesitan estar vinculados por medio de cables.
WSN	(Wireless Sensor Network - Red de sensores inalámbricos) Red de sensores autónomos que se distribuyen para un monitoreo específico.

X

XMPP	(Extensible Messaging and Presence Protocol) Protocolo de mensajería instantánea para aplicaciones como multichat.

Z

Zigbee	Conjunto de protocolos de alto nivel de comunicación inalámbrica, bajo consumo para maximizar la vida útil de baterías. Usa el estándar IEEE 802.15.4
Z-wave	Protocolo de comunicaciones inalámbricas que se usa para domótica. Es una red en malla que usa ondas de radio de baja energía; de este modo permite el control de aparatos o equipos.

CAPÍTULO I

INTRODUCCIÓN

1.1. Antecedentes

El principal requisito para que la conexión entre dispositivos mediante internet es, la interoperabilidad de estos, es decir los objetos deben prácticamente entenderse entre sí, o en su defecto “hablar el mismo idioma”; lo que implican protocolos y codificaciones, es en este punto en que la estandarización entra en cuestionamiento para el desarrollo e implementación de IoT. Es importante mencionar que se tiene la necesidad de innovar, para ello se requiere compartir información; sin embargo, dicha interoperabilidad no se encuentra claramente definida, es decir, se deben considerar las capas de comunicación.

Con base en, el informe de McKinsey Global del año 2015, el que indica que la interoperabilidad es necesaria para generar el 40 % del potencial de IoT, lo que significa que tienen impacto social y económico, ya que con el uso de IoT en distintos sectores permitiría implementación de estándares existentes, incluso el desarrollo de nuevos con el fin de incrementar nuevos modelos de negocios, economías en escala y generar valor dentro de una empresa, es decir, competir con las nuevas tecnologías permite sostenerse en el mercado y obtener utilidades.

Existen organismos de normalización como IETF, ITU, IEEE, conjuntamente con nuevas propuestas como Industrial Internet Consortium, Open Interconetion, ZigBee entre otras, trabajan

en la evaluación, desarrollo y correcciones de los estándares y protocolos pertinentes. (Rose, Eldridge, & Chapin, 2015)

De acuerdo a lo que se puede evidenciar en investigaciones, existen estudios realizados en cuanto a análisis de desempeño de aplicaciones IoT con el uso de protocolos CoAP, MQTT incluso 6LoWPAN (Cañete, 2015), todo aquello que se refiera a comunicaciones machine to machine (M2M); sin embargo, lo que se busca es iniciar el análisis en cuanto al desempeño de los protocolos propiamente dichos.

1.2. Planteamiento del problema

En la actualidad el número de los dispositivos que se interconectan a través de internet, es cada vez superior se convierte en una expresión en auge. Desde el punto de vista de usuario, se evalúa que, a futuro para lograr el cumplimiento de necesidades, exigencias dentro del entorno laboral, tecnológico inclusive del hogar, es preciso el uso y el conocimiento de IoT, ya sea desde un smarthphone, tablet, reloj, auto, incluso desde prendas de vestir; esto implica que los objetos en general en lo posterior se fabricarán con el reto que implica la capacidad de interconexión entre sí mediante internet. Al considerar estos avances, IoT presenta la necesidad de estandarización de protocolos, ya que se debe garantizar la compatibilidad e interoperabilidad entre los objetos.

Por este motivo, la estandarización de protocolos en IoT se convierte en uno de los principales desafíos, hoy en día existen protocolos que han tomado gran impulso en implementaciones de internet de las cosas, entre ellos: MQTT y CoAP; sin embargo, surge la interrogante: como definir cuál de ellos es mejor, en cuanto a su desempeño y en características generales, a partir de este planteamiento se genera la propuesta para este proyecto.

El desempeño y estandarización de protocolos no son los únicos puntos de investigación que implica la tecnología IoT, por ello, se considera importante mencionar otro de los desafíos que es la seguridad que se proporciona en este tipo de entornos. Partiendo del punto que un usuario al proporcionar sus datos e información de cualquier tipo a la red implican riesgos de seguridad, privacidad y confidencialidad; para ello el usuario debe tener la confianza de que sus datos están protegidos y no se hará mal uso de ellos. (Rose et al., 2015) Dentro del proyecto el tema de seguridad será tomado como un análisis futuro que se debe realizar, puesto que, para comprender esta temática, es preciso estudiar el comportamiento general de cada protocolo.

1.3. Justificación

Es preciso el análisis que se planteó en el perfil de este proyecto, ya que se hace frente a uno de los desafíos de IoT, que es la interoperabilidad y estandarización, ya que IoT es una tecnología que crece de manera exponencial, ya que cada vez más dispositivos se fabricarán con el fin de su interconexión a Internet. Esto hace que dichos dispositivos intercambien datos e información, es decir deben comprenderse entre sí, para lo cual deben existir protocolos de comunicación estandarizados en sus distintas capas, según su arquitectura. Al prevenir este avance, ya existen varios protocolos, mismos que requieren de un análisis para comprender su funcionamiento y orientar su uso, de acuerdo a los requerimientos del usuario.

En este proyecto se plantea el análisis de los protocolos CoAP y MQTT, protocolos que se usan con más frecuencia, en especial en soluciones para el hogar, tomando en cuenta que para estas soluciones se usan ambientes Wireless IEEE 802.11.15.2. Para ello se desarrolló un prototipo con Raspberry PI 3 y Cloud comercial, mediante la elaboración de scripts en IDE de phyton llamado Node-Red o programación visual, para cada protocolo.

El prototipo comprende la implementación de un proceso común en el hogar como el control de una lámpara. Posterior a ello, mediante la observación y toma de información se analizó básicamente tres criterios de desempeño Throughput, jitter, y delay. Con la información pertinente se realizó la comparación propuesta y se definió el protocolo que mejor desempeño posee para estos entornos.

Además, es indispensable para el conocimiento de quienes se encuentran en vigilancia de la tecnología destacar lo fundamental que es la seguridad en IoT, ya que al ser una tecnología que avanza a pasos agigantados, crecen también los riesgos que implica tener nuestra información en tránsito en Internet; sin bien es cierto colabora de múltiples maneras al desarrollo de la sociedad, constituye un peligro si los sistemas no cuentan con la seguridad pertinente. En este proyecto se consideró que la seguridad es un referencial para trabajos futuros.

1.4. Objetivos

Objetivo general.

Determinar el mejor desempeño entre los protocolos MQTT y CoAP en aplicaciones IoT, a través de un prototipo implementado mediante Raspberry PI 3 en entornos Wireless 802.11 dentro del hogar.

Objetivos específicos.

- Analizar el estado del arte de aplicación, avance y deficiencias en desempeño de entornos IoT; y en consecuencia de sus protocolos de implementación.
- Analizar modelos de diseño para arquitecturas IoT, tomando en cuenta la futura estandarización de esta tecnología.

- Desarrollar un prototipo IoT con Raspberry PI 3 en entornos Wireless 802.11 dentro del hogar.
- Evaluar el desempeño de los protocolos MQTT y CoAP a partir de la implementación de los prototipos IoT.
- Analizar los resultados de los prototipos desarrollados.
- Emitir conclusiones y recomendaciones de desempeño sobre los protocolos IoT MQTT y CoAP.
- Definir el marco referencial y conceptual orientado a cyberseguridad, para protocolos y entornos IoT, con el fin del planteo de futuros trabajos en esta área.

1.5. Metodología de la investigación

La metodología que usó en este proyecto es cualitativa y experimental, ya que pone en marcha a partir de un análisis que no se puede cuantificar, es decir surge de la observación directa de un caso u objeto de estudio. ("Metodología de la investigación", 2016); y experimental ya que se desarrollaron prototipos cada uno con su proceso en programación visual, lo que permitió el análisis de dichos protocolos.

1.6. Hipótesis de investigación

La hipótesis que se plantea se basa en la siguiente pregunta: ¿Qué protocolo tiene el mejor desempeño CoAP o MQTT para aplicaciones IoT, en ambientes Wireless 802.11 dentro del hogar?

Se partió de esta pregunta, puesto que dichos protocolos son los que más se usan en aplicaciones IoT, lo que impulsa una oportuna investigación en la que se analice el funcionamiento de cada uno de ellos y determinar cuál de ellos puede ofrecer mejores servicios, con base en throughput, jitter, y delay.

CAPÍTULO II

ESTADO DEL ARTE DE IoT (INTERNET OF THINGS - INTERNET DE LAS COSAS)

2.1. Investigación de artículos

Se investigaron los artículos que tienen semejanza al tema que se planteó, se encontraron varios artículos, que fueron de gran ayuda para realizar sus referencias y a partir de ello encontrar información adecuada. Los artículos en los que se basó de manera inicial la investigación son los siguientes:

• (Jurado, Velásquez, & Vinueza, 2014)	Estado del Arte de las Arquitecturas de Internet de las Cosas (IoT)	Menciona los sectores y el avance de IoT, desde su aparición. Además, trata temas sobre su gestión, servicios, comunicación, seguridad y procesos en cuanto a IoT
• (Alcaraz, 2014)	Internet de las Cosas	Explica el concepto y definición de IoT, así como las áreas en que usa. Termina con una explicación sobre los desafíos para una implementación eficiente.
• (Camacho, Oropeza, & Lozoya, 2017)	Internet de las cosas y Realidad Aumentada: Una fusión del mundo con la tecnología	Trata IoT y realidad aumentada como tecnologías en pleno auge que se pueden explotar en varios ámbitos, con el aprovechamiento de un dispositivo. Finaliza con el desarrollo de una aplicación para demostrar lo que se puede realizar con esta fusión.
• (CSIRT-CV, 2017)	SEGURIDAD EN INTERNET DE LAS COSAS Estado del Arte	Realiza un a introducción, definición y riesgos de IoT con respecto a la seguridad; realiza una recopilación de incidentes y emite recomendaciones para prevenir ataques, así como del uso de aplicaciones.

• (Evans, 2011)	Internet de las cosas Internet de las cosas Cómo la próxima evolución de Internet lo cambia todo	Informe técnico de Cisco a partir de una investigación, menciona cifras de objetos que lograrán conectar a Internet, importancia, generación de información, desafíos y barreras en un futuro.
• (Luzuriaga, Zennaro, Cano, Calafate, & Manzoni, 2016)	Evaluando un escenario de pruebas para el IoT entre la emulación y el uso de dispositivos reales	Habla sobre las consecuencias de que existan grandes cantidades de objetos en Internet, ya que implica tráfico de datos a través de las redes. En este artículo se valida el uso del protocolo MQTT para mensajería dentro de una red con varios nodos suscriptores como publicadores, y avalúa su desempeño.
• (Osako, Josias, & Gimenez, 2016)	Internet das Coisas : o desafio de estabelecer um padrão único de protocolo	Menciona los resultados de investigaciones de IoT, también menciona la falta de estandarización y sus dificultades para generar alcance en más servicios y proyectos. Presenta las definiciones y funcionamiento de 4 protocolos de IoT entre ellos MQTT y CoAP.
• (Semle, 2016)	Protocolos IIoT para considerar	Artículo técnico que describe las principales características de varios protocolos de IoT.
• (Rodrigues & Zem, 2016)	Estudo dos Protocolos de Comunicação MQTT e CoAP para Aplicações Machine-to-Machine e Internet das Coisas	Habla sobre temas en cuanto a IoT y aplicaciones M2M, haciendo referencia a los protocolos CoAP y MQTT con aplicaciones de lectura de temperatura.
• (Vargas, Salvador, Yacchirema, & Palau, 2016)	Smart IoT Gateway For Heterogeneous Devices Interoperability	Menciona aplicaciones de IoT, como: sensores, electrodomésticos sector industrial, dispositivos. Menciona la heterogeneidad entre objetos y la capacidad de comunicación entre ellos. Adempas analiza lass ventajas que se generan en IoT así como los desafíos que se presentan.
• (Stansberry, 2015)	MQTT and CoAP: Underlying Protocols for the IoT	Realiza un análisis de los protocolos CoAP y MQTT, de acuerdo a los documentos técnicos y teóricos de cada uno de ellos.

• (D. Martínez & Guillen, 2015)	Diseño de un sistema en Cloud, para controlar dispositivos IoT vía web	Trata sobre la implementación de un sistema que usa la Web para el control de dispositivos inteligentes. Para ellos analiza estado del arte, definiciones, protocolos de comunicación y servicios de IoT.
• (Sengul & Kirby, 2017)	MQTT-TLS profile of ACE	Documento que detalla un perfil de autenticación, autorización para habilitar un sistema de mensajería MQTT.
• (Shelby, Hartke, & Bormann, 2014)	The Constrained Application Protocol (CoAP)	En este documento se detalla el funcionamiento de CoAP com protocolo, sus estructura, mensajes, bits, opciones, y métodos.
• (Oasis, 2018)	MQTT Version 5.0	Documento que explica el funcionamiento de MQTT como protocolo de mensajería, hace referencia las comunicaciones M2M. Detalla todasa sus características.
• (Melián & Anías, 2015)	Gestión de la comunicación máquina a máquina (M2M)	Detalla la comunicación de máquina a máquina, que es una de las tecnologías que se generan con IoT, es decir comunicación entre objetos, incluyendo los del hogar. Este documento se enfoca también en la cultura verde que es el cuidado del ambiente.
• (Yuan, 2017a)	Conozca MQTT	Artículo breve en que explica características de MQTT, un modelo de funcionamiento e instalación de mosquitto.

2.2. Selección de artículos

De la lista de artículos anterior, se seleccionaron aquellos que contienen mayor información para el desarrollo de este proyecto. Por lo que se realizó un análisis con el aporte de cada trabajo.

En (Alcaraz, 2014) se presenta un análisis de la tecnología Internet de las Cosas, da a conocer el concepto general, presenta ejemplos, y las áreas en que se puede desarrollar. Además, realiza una explicación de los desafíos que surgen con esta tecnología al momento de realizar cualquier

tipo de implementación. Se debe tomar en cuenta que de acuerdo a las áreas que se analizan, IoT se puede implementar desde el hogar hasta grandes industrias. Finalmente indica ejemplos de los servicios que se prestan con esta tecnología. Este trabajo sirvió de base para la investigación tanto de antecedentes como de marco teórico para este proyecto; ya que, presenta claramente los problemas que hoy en día se evidencian y sus conceptos expuestos de forma clara.

En (Evans, 2011) habla de la tecnología Internet de las Cosas como la evolución que cambiaría el comportamiento del mundo y de los seres humanos. Inicia con un análisis del comienzo y las tecnologías que dieron paso a IoT. Además, muestra cifras estadísticas del avance en el uso de dispositivos y la conexión a Internet. También, se presenta la relación que tienen las empresas y giros de negocio con la tecnología; finalmente incluye los pasos a seguir en cuanto IoT se imponga en las personas y los retos que junto con esta tecnología implican. El estudio estadístico de este artículo fue de gran ayuda para comprender el panorama del futuro inmediato en la tecnología especialmente en IoT, y los problemas que no se han resuelto y se deben hacer frente. Además, este documento fue de gran aporte para analizar los sectores e industrias en que IoT, se encuentra en auge y gran uso.

En (Osako et al., 2016) se muestran resultados de una investigación, inicia con conceptos y definiciones de IoT. Enlista y describe los desafíos que implica la implementación de esta tecnología. Analiza también una propuesta para solventar el problema de estandarización tanto para dispositivos como para servicios. Este artículo se centra en la investigación de cuatro protocolos, 6LoWPAN, ZigBee, MQTT y CoAP; con explicaciones de sus características, definiciones funcionalidades e implementaciones. Este trabajo aporta con el presente proyecto, ya que presenta comparaciones con respecto a cuatro protocolos que son los más frecuentes; sin embargo, este

artículo realiza una investigación teórica, por tanto, sus resultados también; mientras que el presente proyecto se basa en la comparación tanto teórica como experimental, de dos protocolos que trabajan bajo la misma capa en la arquitectura de IoT, cuyos resultados son más precisos.

En (Semle, 2016) analiza la interoperabilidad como el mayor desafío en IoT, menciona que la coexistencia de los protocolos actuales es importante, ya que algunos de ellos son abiertos mientras que otros son privados y se encuentran en constante lucha para mantenerse como el protocolo estándar. En este artículo detalla los protocolos abiertos entre ellos MQTT, CoAP y HTTP. Además, realiza un análisis de diferencias entre protocolos de tipo cliente/servidor y publicar/suscribir. Este trabajo aporta al presente proyecto, en cuanto a funcionalidades de protocolos, para la elección de aquellos que se someterían a una comparación de desempeño; ya que existen varios que mantienen similares características.

En (Rodrigues & Zem, 2016) mencionan aplicaciones de tipo M2M dentro de entornos IoT, contiene definiciones y conceptos de la investigación en otros artículos. Este trabajo realiza un estudio comparativo entre dos de los protocolos para este tipo de comunicaciones, que son: MQTT y CoAP. Se muestran aplicaciones que se desarrollan para cada protocolo, con base en la simulación, de transmisión de datos de temperatura. Se analizan los paquetes de datos con Wireshark. Dentro de este trabajo se indica que esta comparación no es suficiente, ya que es necesario realizar este mismo estudio dentro de ambientes reales. Este documento generó una línea base para la investigación de este proyecto, puesto que justamente realiza una comparación de MQTT junto con CoAP, a base de un trabajo orientado a la industria; la diferencia es que, la implementación no se la logró, ya que tanto las mediciones como conclusiones se forjan de acuerdo a simulaciones, y para aplicaciones de lectura de temperatura. Por este motivo, es preciso el análisis

en un entorno común, como es el hogar y bajo condiciones reales, como es la propuesta del presente proyecto.

En (Stansberry, 2015) presenta un análisis estadístico de los dispositivos con conexión a Internet en los próximos años; menciona las capas de aplicación, transporte e Internet y los requerimientos de los protocolos dentro de ellas. Este artículo explica brevemente las principales prestaciones de los protocolos MQTT y CoAP. Este documento aporta con literatura concreta para, comparar las arquitecturas de IoT, que se pueden usar. A partir de ello, emitir sugerencias e investigaciones para solventar el desafío de estandarización e interoperabilidad.

En (Sengul & Kirby, 2017) se halla un modelo ACE (Autenticación y Autorización), con base en MQTT. Para ello, realiza un estudio profundo de dicho protocolo, junto con todas sus funciones, mensajes, interacciones, establecimiento de conexiones, autorizaciones, estructura de mensajes, banderas, validaciones y opciones. Este artículo colabora al avance del presente proyecto, para considerar que la seguridad tanto de los objetos como de los sistemas que se implementen, deben poseer modelos de seguridad, de modo que, los usuarios tengan plena confianza en introducir su información. Además, otro de los aportes de este trabajo es el marco teórico entorno a MQTT.

En (Shelby et al., 2014) se muestra un estudio profundo del protocolo CoAP, por ello se destaca las funcionalidades, características, modelo de mensajería, formato, estructura de mensajes, solicitudes, respuestas, control, transmisión de parámetros y métodos. Este documento fue de gran aporte para el estudio y comprensión del protocolo CoAP, y de este modo trasladar lo teórico en experimental. En (Oasis, 2018) se encuentra la relación que existe entre MQTT y el NIST Cybersecurity Framework versión 1.0, a la que se denomina OASIS. Detalla la última versión de

MQTT. Este documento colabora con el presente proyecto, ya que se basó en el detalle técnico de MQTT, de modo que su implementación sea comprensible.

2.3. Conclusiones del estado del arte

- De acuerdo a los artículos de selección, se puede observar que, existen trabajos similares de comparación de protocolos de comunicación; sin embargo, ninguno de ellos realiza la propuesta de este proyecto en un ambiente de control en el hogar.
- Existen trabajos de comparación de los protocolos MQTT y CoAP, a nivel teórico, lo cual es de gran ayuda para iniciar la investigación de este proyecto. Por ello se concluye que los resultados son variables de acuerdo a la aplicación a la que se enfoque.
- Los objetivos de este proyecto nacen para hacer frente a la falta de estandarización de los protocolos de IoT; por tal motivo, se concluye que los resultados de este proyecto son de gran aporte a este desafío.
- El valor agregado del presente proyecto radica en una aplicación sencilla, cuya experimentación y comparación de los protocolos MQTT y CoAP, se basan en mediciones cuyos valores son reales, en un entorno frecuente; lo que no se realiza en otros trabajos.

2.4. Definición de IoT

IoT se puede definir como los sistemas conformados por dispositivos u objetos, capaces de conectarse a Internet que se encargan de recolectar información de distinto tipo, como: temperatura, sonido, luz, distancia, humedad. Los datos se procesan y se genera una gran cantidad de información, por eso la capacidad de Internet debe desarrollarse conjuntamente, ya que al hablar de IoT, se habla de un crecimiento exponencial de dispositivos u objetos conectados a la red de redes.

IoT desde la perspectiva empresarial, se prevé que será de gran utilidad para la mejora en los procesos tanto operativos como administrativos, ya que la información procesada a partir de los datos obtenidos de los clientes, proveedores y competencia, será de gran utilidad en la toma de decisiones para la implementación de nuevos sistemas o estrategias que permitan la sostenibilidad del negocio. De hecho, existen empresas que ya han puesto en marcha este desarrollo (Figura 1)

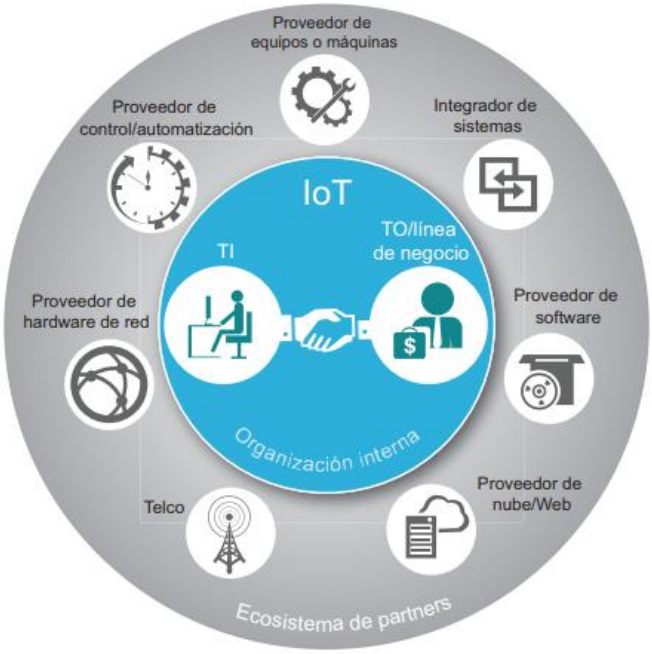


Figura 1. Entorno IoT interno-externo
Fuente. (Noronha, Moriarty, O'Conell, & Villa, 2014)

2.5. Historia y Evolución de IoT

El término IoT surgió en 1999 por Kevin Ashton en representación del Instituto de Tecnología de Massachusetts. En un principio se diseñaron infraestructuras RFID (Radio Frecuency Identification – Identificador de Radiofrecuencia) dichas infraestructuras se componían de un conjunto de objetos capaces de conectarse a Internet, mediante un RFID y posteriormente con WSN (Wireless Sensor Network – Red Inalámbrica de Sensores) con estos conceptos de establecían los

NFC (Near Field Communication – Comunicación de campo cercano), que se usan con frecuencia en soluciones domóticas y empresariales para el control de puertas, ventanas, interruptores, control de acceso entre otros, se puede decir que IoT nace también a partir de la domótica ya que en este ámbito se desarrolla el control y automatización de los objetos en el hogar.

El control en el hogar se establece desde redes inalámbricas que ya se conocen, iniciando con la tecnología de Infrarrojo, Wifi, Zigbee, Bluetooth, WLAN y WIMAX. (Esquivel, 2016). En el 2002 Ashton ya menciona que "se requiere que IoT se encuentre estandarizado para que los computadores se comprendan en el mundo real". (Hernández Rojas, Mazon Olivo, & Campoverde Marca, 2015)

De acuerdo a la IETF, grupo de trabajo de ingeniería se conoce a IoT, como "redes de objetos inteligentes" y al usar el término "objetos inteligentes" se entiende que dentro de este grupo se encuentran los dispositivos que se limitan de alguna forma en cuanto a energía, memoria o procesamiento; entonces surge la necesidad de que los dispositivos sean interoperables, sin importar el tipo de objeto o su fabricante; esto implica que se cumplan una lista de aspectos importantes, por ejemplo: la existencia de protocolos estandarizados con el fin de evitar conflictos entre fabricantes, seguridad y ahorro de energía. (Rose et al., 2015)

Hoy en día IoT se considera una tecnología en tendencia, en constante desarrollo e investigación; puesto que, si se analiza que en el mercado existen varios fabricantes de dispositivos, ya no sólo se comercializa sensores o dispositivos electrónicos, también se incluyen objetos comunes y de uso diario como autos, prendas de vestir, relojes, aretes, brazaletes, zapatos; es decir ya no es necesario que sea un ordenador, o a su vez que, posea un microprocesador. (Hernández Rojas et al., 2015)

De esta manera estos objetos serán capaces de intercambiar información, sin que sea necesario que el usuario intervenga. Por ello, los objetos se convierten en fuentes inagotables de información y se caracterizan por consumo bajo de energía, con el objetivo de automatizar o controlar un dispositivo o sistema. (Rose et al., 2015).

Debido a las funcionalidades que brinda IoT, causa impacto en varios sectores, no solo en el área de las telecomunicaciones sino también en áreas como: agrícola, producción en masa, salud, comunicaciones, medio ambiente, transporte, seguridad, petrolero. Estos sectores se encuentran desarrollando diversas aplicaciones IoT para que sus productos y procesos sean sustentables y efectivos; con el fin de integrar en un solo sistema maquinarias, equipos y personas con sus respectivos dispositivos personales, con esto se genera el desarrollo de otras tecnologías como por ejemplo Big Data, debido al volumen de datos y de información que se maneja actualmente; la inteligencia artificial, así como la relación Machine Learning o Aprendizaje de máquinas. Por lo tanto, las actividades dentro de aplicaciones industriales son automatizadas, lo que permite interacciones Hombre – Hombre, Máquina – Máquina y Hombre - Máquina. (Hernández Rojas et al., 2015)

Otra de las aplicaciones de gran interés con respecto a IoT con las ciudades inteligentes, es decir Smart Cities, en este ámbito serán los habitantes de cada ciudad quienes evalúen la aceptación de los nuevos servicios y tecnologías implementadas, por ejemplo Londres se destaca en proyección internacional y tecnología; sin embargo, en gestión pública sus indicadores no son óptimos (Jurado et al., 2014)

En cuanto a las aplicaciones a niveles industriales y por lo tanto empresariales surge la necesidad urgente de innovar, ya que es la única manera en que logra generar valor dentro las organizaciones

y en sus giros de negocios, esto implica prever transiciones que pueden surgir en los próximos años, así como nuevos modelos de negocio, de modo que la compañía sea sostenible y permanezca en el tiempo. Sin embargo, para desarrollar ideas innovadoras se requiere de la optimización de procesos tanto operativos como de infraestructura tecnológica; de esa forma se confrontan los desafíos implícitos en IoT.(Noronha et al., 2014)

Es importante, tomar en cuenta que, a partir de los datos se genera información valiosa, para la toma de decisiones en los diversos sectores; en este contexto se puede mencionar que, la información genera conocimiento, y a su vez el conocimiento junto con la experiencia genera sabiduría; lo que implica un gran beneficio para la humanidad. (Figura 2) En síntesis los datos como tal, no son de utilidad; mientras que, los mismos datos en grandes cantidades sirven para identificar tendencias y patrones.(Evans, 2011)

Figura 2. Conversión de datos en sabiduría
Fuente. (Evans, 2011)

En este sentido, los datos deben convertirse en información útil dentro de un sistema, para ello, se cuentan con las siguientes fases que son: percibir, recolectar, transmitir, analizar y distribuir datos; a partir de estas fases se puede continuar con el desarrollo de otro tipo de tendencias, como en la actualidad Business Intelligence o Analytics. Sin embargo, a pesar del uso de la tecnología es

necesaria la intervención de los seres humanos, por ello IoT significa el tratamiento de los datos a través de la información que se obtienen de los objetos. (Evans, 2011)

2.6. Sectores de desarrollo de IoT

De acuerdo a la investigación, se evidencia que IoT, realmente mejora los procesos de las empresas, es por ello que varios sectores explotan su potencial.

Hogar: En este sector se pueden visualizar varias aplicaciones, una de ellas electrodomésticos inteligentes, en la actualidad ya se cuentan con televisores inteligentes; sin embargo, se proyecta más allá. Como un referente se puede tomar una lavadora con autonomía, escoja en programa de lavado de acuerdo al tipo de prenda que contenga, peso y color. Otro de los proyectos en desarrollo es un refrigerador que monitorice el abastecimiento de productos que más se consume en el hogar y, en consecuencia, realizar un pedido de los productos faltantes al supermercado de preferencia. Al tomar en cuenta, la utilidad de los teléfonos celulares que también son inteligentes, se pueden generar otras aplicaciones como el control y la automatización del hogar, por ejemplo: control de interruptores, persianas, ventanas, luces, gas, encendido de electrodomésticos, y aire acondicionado, entre otras aplicaciones. (Alcaraz, 2014)

Ciudades Inteligentes: Se puede implementar el control y monitorización del tráfico de una ciudad en tiempo real, ya que se trata de un problema común en las grandes ciudades, ya que poseen gran flujo de automotores, que afecta tanto a la movilización como al ambiente. Con el desarrollo de IoT, se prevé tener un mejor desempeño de los semáforos debido a los sensores y cámaras que se ubicarían en sectores claves que indique el cambio de luz y el tiempo adecuado, de acuerdo a la cantidad de vehículos en el sitio. (Alcaraz, 2014) Otra de las aplicaciones dentro de una ciudad se

plantea la inspección de edificios, es decir mediante sensores se verifique cualquier tipo de daño, afección, grietas dentro de una construcción, de modo que exista generación de alertas tempranas indicando evacuación o atención según corresponda el caso. Además, dentro de un gobierno municipal el desarrollo en este campo se prevé indispensable, puesto que se controlaría de mejor manera el alumbrado público, sistemas de riego, condiciones del suelo, temperatura, recolección de basura en horarios que no causen conflicto. (Alcaraz, 2014) Sin embargo, para que una ciudad se encuentre dentro de la denominación Smart o Inteligente, se deben tomar en cuenta varias dimensiones, por ello es que el desarrollo de IoT en las ciudades constituyen un verdadero reto que tomaría un prolongado tiempo, esta dimensiones son: gobernanza, planificación urbana, gestión pública, tecnología, medio ambiente, proyección internacional, cohesión social, movilidad y transporte, capital humano y economía, es decir se convierte en una combinación de los sectores en que se desarrolla IoT. (Acciona, 2018)

Barcelona constituye un claro ejemplo se Smart City, puesto que se basa en dos ejes Smart Water y Smart Transportation; para ello se cuenta con una red de sensores en casi todos sus drenajes, tanques y líneas de suministro de agua, de modo que genera estrategias de forma centralizada para estimar la cantidad de uso, tanto en el hogar como en la industria. (González, 2017)

Vehículos: En la actualidad los vehículos ya cuentan con sensores para el control de su funcionamiento, como, por ejemplo: la temperatura, nivel de aceite, presión de aire en los neumáticos, incluyendo un canal de comunicación con los concesionarios para solicitar información específica, todo esto con el fin de realizar un mantenimiento adecuado al automotor y preservar su vida útil.

Salud: En este sector existen sensores para el hogar, ropa u objetos como brazaletes, zapatos y relojes para el monitoreo de niveles de azúcar, para tratamientos, ya que al recolectar este tipo de información diariamente, se pueden extender medicamentos adecuados que sean eficientes para el seguimiento de cada caso; sin embargo, se debe considerar un aspecto fundamental, ya que no debe existir margen de error y su seguridad debe ser totalmente garantizada, ya que una dosis mal aplicada podría generar la muerte del paciente. (Domínguez & Vargas-Lombardo, 2018). Además, estos beneficios pueden alcanzar a los adultos mayores, con objetos que monitoreen los signos vitales de personas de edad avanzada y que sean propensas a sufrir un accidente, de modo que en caso de emergencia la alerta se extienda desde un familiar cercano, así como una ambulancia que se dirija al centro de salud más cercano. Inclusive telemedicina, para atender cirugías de manera remota.(Alcaraz, 2014). Adicionalmente, se puede potencializar el desarrollo de los hospitales tanto en infraestructura como en tecnología, lo que optimizaría los procesos dentro de las casas de salud, cabe indicar que estas aplicaciones ya las implementan varios países.(Domínguez & Vargas-Lombardo, 2018)

Agricultura y ganadería: Para este sector el IoT ha sido de gran utilidad, puesto que se observan aplicaciones de monitoreo de variables ambientales como: supervisión de procesos de la producción, control de riego, monitoreo de oxígeno, temperatura del agua, niveles de PH, niveles de contaminación, protección de heladas, fertilización, presencia de químicos nocivos, consumo de agua, para cultivos de plantas, camarón y otras especies acuícolas.(Cázarez, 2010) (González, 2017) Para el caso de la ganadería las aplicaciones se orientan a la ubicación del ganado, estado de nutrición, enfermedades, eficiencia reproductiva, calidad de los pastos. (González, 2017)

Medio ambiente: Con la combinación de IoT y Blockchain que es una tecnología en tendencia, que permite la ejecución de transacciones sin la existencia de intermediarios, esta tecnología cambiaría los procesos de las empresas en especial del sector bancario y económico, con menor uso de papel moneda y en consecuencia menor contaminación del medio ambiente. (Calatayud, 2018) Los sensores también pueden obtener información de contaminación y calidad del aire, agua, suelo inclusive el sonido; de este modo se podrían emitir normativas en torno al ambiente para controlar y detectar incumplimiento por parte de las industrias y de la ciudadanía en general.(Alcaraz, 2014)

Industria: Dentro de este sector se refiere a la mejora de los procesos de producción, inventario y distribución de bienes y servicios, como puede ser detección temprana de fallos en máquinas industriales que se encargan de producción en masa. Otro de los desarrollos, que se evidencian en los últimos años es el comportamiento de los consumidores, a partir de los objetos que tiene cada consumidor con conexión a Internet, puesto que de esta forma se puede conocer las preferencias de los consumidores, esto se puede combinar con publicidad que permita interacción con los posibles clientes o en su defecto clientes existentes.(Alcaraz, 2014)

Como se puede observar en el sector de la industria, existen aplicaciones que pueden ser indispensables en esta era de revolución tecnológica; las mismas que mejoran la eficiencia de producción. En consecuencia, de ello se genera una nueva tecnología que abarca la industria y sus procesos, se la llama Industria 4.0.

2.7. Industria 4.0.

Es un término que hace referencia a un nuevo modelo de organización en la actual revolución que presentan los sistemas de información dentro de las industrias, ya que hoy en día estos sistemas juegan un papel importante en el desarrollo de las empresas; y de una u otra manera manejan gran cantidad de datos desde dispositivos, máquinas, celdas o productos, haciendo que este nuevo modelo forme parte de la cadena de valor en el ciclo de vida de un producto y se caracteriza por la digitalización de los procesos. (Catalán, Serna, & Blesa, 2015)(Román, 2016). El término nace en Alemania en un inicio como "Fábrica Inteligente" o a su vez como "Internet Industrial", que es básicamente IoT aplicado en el sector industrial, se considera que dicha revolución, se debe a la transformación digital que se genera debido al gran uso de las tecnologías de la información y comunicación.(Román, 2016). Industria 4.0, se basa en las siguientes tecnologías. (Román, 2016)

2.7.1. Tecnologías de Industria 4.0

Industria 4.0 se basa en una serie de tecnologías; cabe indicar que todas ellas también se generan a través de entornos IoT.

Comunicaciones móviles: Mediante M2M se logra un ambiente de producción en el que se establece comunicación entre los sistemas y los productos, para obtener la información en tiempo real.(Román, 2016)

La nube o Cloud Computing: Tecnología que sirve para acceder a la información desde cualquier lugar, en tiempo real; se requiere de Internet y de un servidor de alta disponibilidad.(Blanco, González, & Rodriguez, 2017)

Análisis de datos o Big Data: Sistemas que logran la integración de capacidades tanto físicas como computacionales, en ello también se involucra la nube.(Catalán et al., 2015) Los datos que se obtienen, sirven para el estudio de patrones, comportamientos y tendencia mediante su debido análisis, en este ámbito se desarrolla otra tecnología llamada minería de datos. (Blanco et al., 2017)

Comunicación Máquina a Máquina o comunicación M2M: Se refiere a los datos que se comparten entre las máquinas que se encuentran conectadas, mediante a un enlace punto a punto o una red.(González, 2017)

Plataformas sociales: Se refiere al uso de las redes sociales, lo que dentro de IoT, se llama social IoT, para formar redes sociales en las que los objetos trabajan como nodos de manera que se facilita la navegación en un conjunto de millones de dispositivos. De este modo el manejo de la información es efectiva.(González, 2017)

Impresión 3D: Corresponde a los modelos virtuales en tres dimensiones, genera la pronta fabricación o creación de prototipos, lo que implica fabricación descentralizada; se trabaja en conjunto con realidad aumentada para su diseño. Con esto se logra eficiencia en la producción, especialmente en la producción en masa, debido al ahorro de tiempo. Incluso permite la creación de objetos personalizados. (Román, 2016)

Realidad aumentada: Es una tecnología que se usa con frecuencia y se la combina con IoT, para aplicaciones educativas, industriales y de entretenimiento. Se basa en la creación de marcadores correspondiente a un objeto real, los marcadores son patrones impresos en blanco y negro. Una de las aplicaciones en cuanto a entretenimiento es el juego Pokemon GO; mientras que, en el sector agropecuario, también es de gran utilidad para determinar un método inteligente de activar los

sistemas de riego en cultivos, mediante una red de sensores, estos se encargan de sensar las condiciones del sitio y de acuerdo a los análisis de los datos activar o desactivar el sistema, cabe indicar que esto se lo realiza con autonomía.(Camacho et al., 2017)

Inteligencia artificial: La inteligencia artificial dentro del Internet hoy es indispensable, un claro ejemplo es Google la usa para personalizar las búsquedas, en sus listas de enlaces; Facebook la usa para las sugerencias y recomendación de noticias; Twitter para mostrar los contenidos de los usuarios de forma cronológica; LinKedln la usa para enlazar a las empresas con los posibles candidatos a un puesto de trabajo; incluso para los vehículos de conducción automática. Esto se debe a la cantidad de información que se maneja en la actualidad, ya que el tráfico es estima que superará el Zetabyte (10^{21}) en los próximos años. (Serrano-Cobos, 2016)

Machine learning o Aprendizaje automático: Dentro de inteligencia artificial se conjuga la tecnología machine learning, ya que los algoritmos son necesarios para que las máquinas obtengan un aprendizaje de patrones o comportamientos en diversas situaciones, de este modo tomar decisiones, tal es el caso de los chatbots, que son conversaciones en que los usuarios no perciben que no existe presencia humana; sin embargo existe un aprendizaje de una máquina que conoce que debe responder y que debe preguntar, en otras palabras se trata de una conversación inteligente e interactiva a través de un software.(Serrano-Cobos, 2016) Inclusive Google desarrolló su aplicación de mensajería ALLO, que contiene las mismas funcionalidades de otras aplicaciones similares, con la diferencia que posee un asistente que verifica los mensajes de entrada, los analiza y sugiere respuestas, sirve tanto para texto como para imágenes. Es decir, si se muestra una imagen que evidencia un suceso importante, la sugerencia para una respuesta será ¡Felicidades! O ¡Bien hecho! (Arantza, 2016)

Seguridad: Como ya mencionó anteriormente la falta de estandarización es lo que provoca que, el tema de seguridad no se solvente, y no permite que se genere una guía en la que consten las mejores prácticas para mermar dicha situación; mientras que las amenazas que atacan a la poca seguridad, cada vez son mayores, las mismas que no pueden ser combatidas con sistemas tradicionales de prevención o recuperación.(Zabala, 2016)

2.7.2. Pilares de inteligencia de Industria 4.0

Industria 4.0, corresponde a una tecnología que implica autonomía en cuanto a procesos a lo largo de la cadena de valor dentro de una empresa, por lo que se soporta en 4 pilares de inteligencia.

Productos inteligentes: Productos que poseen su equipo electrónico, software y conectividad embebidos. Esto hace que el producto tenga la capacidad de comunicación M2M (Machine to Machine – Máquina a Máquina) y con interacción humana para autogestionarse y tomar decisiones descentralizadas. (Román, 2016)

Servicios inteligentes: Se refiere a los servicios que componen un nuevo modelo de negocio, es decir resulta innovador su uso, por los servicios inteligentes se pueden analizar grandes cantidades de datos, de modo que, se puedan generar nuevos servicios, modelos o a su vez optimizar los existentes, inclusive se pueden automatizar la toma de decisiones de acuerdo a la ocurrencia de ciertos eventos, a esto se lo llama Big Data.(Blanco et al., 2017) (Román, 2016)

Innovación inteligente: El término innovación será de mayor relevancia en estos últimos tiempos, porque al incrementar la conectividad se deben aprovechar ciertas oportunidades que con la tecnología surgen, para ello algunas compañías se apoyan en herramientas colaborativas para los procesos de innovación, que directamente se involucran, tanto los clientes como los socios de la

industria; de esta forma se reducirán los tiempos de comercio. Adicionalmente, se debe considerar que el tema de innovación se lo debe implementar durante todo el ciclo de vida de un producto. (Blanco et al., 2017)(Román, 2016)

Cadenas de suministro inteligentes: En Industria 4.0 las cadenas de suministro se basan en redes de colaboración AD-HOC, con el fin de satisfacer necesidades de cada cliente; estas redes deben conformar sistemas de producción que cuenten con conectividad entre empresas; es decir cada empresa debe contar con una interface dentro de la red, la misma que debe contar con el software adecuado y que cumpla con los retos de la innovación. (Román, 2016)

2.8. Modelos de comunicación en IoT

IoT posee los siguientes modelos de comunicación.

Modelo de comunicación dispositivo a dispositivo: Este modelo que se basa en la comunicación directa entre dispositivos, no hace falta la intervención de un servidor de aplicaciones, para lo que se usan protocolos como Z-Wave, Bluetooth o ZigBee. Por lo general este modelo se usa para aplicaciones de automatización dentro del hogar, debido al bajo consumo de ancho de banda y uso de pequeños paquetes de datos, por ejemplo interruptores, iluminación y puertas; desde este tipo de aplicaciones se genera la necesidad de cumplir con requisitos de interoperabilidad y compatibilidad entre objetos, lo que involucra directamente a los fabricantes de los objetos, ya que es indispensable que los objetos interactúen y entiendan los datos que pueden compartir entre sí, lo que también se encamina directamente con el uso de formatos de datos estandarizados. (Rose et al., 2015)

Modelo de comunicación dispositivo a la nube: En este modelo el objeto se comunica con un servicio de la nube, de modo que se logre intercambiar datos y controlar el tráfico de los mismos. Cabe indicar que, pueden existir dificultades cuando los dispositivos son de distintos fabricantes, debido a la interoperabilidad, además se puede presentar cierta dependencia con el proveedor de tecnología, ya que es el proveedor quién tiene acceso a los datos y a la propiedad.(Rose et al., 2015)

Modelo de comunicación dispositivo a puerta de enlace: En este modelo los dispositivos se conectan mediante un servicio ALG o software de aplicación que se ejecuta en una puerta de enlace, para llegar a un destino que correspondería un servicio de la nube. Es decir, en este caso existe un intermediario entre el dispositivo u objeto y el servicio de la nube, se usa en aplicaciones dentro del hogar, para reducir el inconveniente de interoperabilidad, ya que logra integrar objetos dentro de un sistema; las dificultades que se presentan son en cuanto a costos y complejidad del desarrollo del software en mención. (Rose et al., 2015)

Modelos de intercambio de datos a través del back – end: Este modelo permite que los usuarios finales puedan compartir y analizar los datos, con ello se logra la interoperabilidad y la portabilidad de los datos. (Rose et al., 2015)

2.9. Desafíos de IoT

Todas las implementaciones de sistemas en entornos IoT implican varios riesgos y desafíos, los mismo que deben hacerse frente, estos son:

Transición de IPv4 a IPv6: En febrero del 2010, se conoce que se agotaron las direcciones IPv4 en el mundo, lo que genera una complicación, ya que cada objeto necesita una dirección IP.(Evans,

2011) Debido a la cantidad de dispositivos que se conectarán a Internet las direcciones no serán suficientes, con lo que será obligatorio el paso a IPv6.

Reducción de costos: Sobre todo se refiere al bajo consumo de energía, ya que se requieren sistemas y dispositivos destacados por su eficiencia energética. En otras palabras, todos los objetos que sean parte de un sistema IoT, debe ser autónomo en cuestión de energía, ya que no puede ser eficiente el cambio de baterías de millones de objetos en el mundo.(Evans, 2011)

Robustez: En hardware debido al procesamiento y condiciones de transmisión, el hardware debe ser compacto y a su vez resistente con su aplicativo o sistema embebido.

Seguridad y privacidad: En los entornos IoT, se dificulta la medición o calificación de seguridad, puesto que requiere de un análisis si el dispositivo se encuentra o no con protección. Cada vez se desarrollan más amenazas a las que se pueden someter diversos objetos. (Evans, 2011) Inclusive se deben determinar los riesgos, ya que al ser dispositivos que procesan información como la posición geográfica, la confidencialidad del usuario se vería vulnerada, y quienes usan medios tecnológicos para cometer crímenes podrían robar información para su conveniencia.(CSIRT-CV, 2017)

Interoperabilidad: Se debe trabajar junto a la optimización de energía, debido a las aplicaciones y servicios que prestarían los dispositivos dentro de un entorno IoT; este desafío abre paso al desarrollo de protocolos de comunicaciones.(R. Martínez, 2016). Además, es importante enfatizar que al buscar que los objetos operen entre sí, se requiere principalmente buscar soluciones a la falta de estandarización. Debido a la falta de estandarización en los protocolos, muchos desarrollos no pueden culminar, y a su vez no se puede hacer uso correcto de herramientas de seguridad. Para

ello, se requiere que los fabricantes no se centren en crear sus propios protocolos, sino que manejen un mismo lenguaje y protocolo para cada objeto, dependiendo de su aplicación. (Zabala, 2016)

2.10. Arquitectura en capas para IoT

Similar al modelo OSI, se pueden desarrollar o aplicar una referencia en capas, es decir se tienen varios protocolos, junto con distintas capas de arquitectura la misma que pueden ser de 3 o 5 niveles (Figura 3) y (Figura 4) (González, 2017). Cada capa utiliza protocolos específicos de acuerdo a su función.

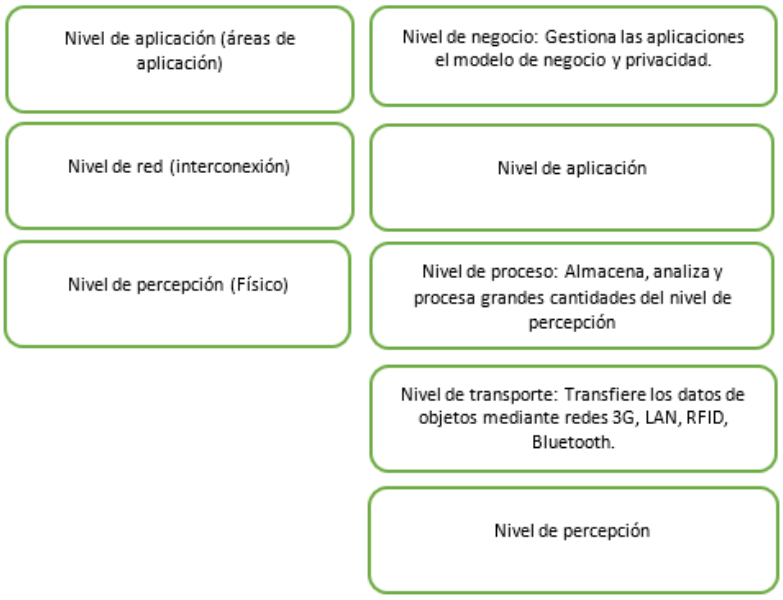


Figura 3. Arquitectura en capas de 3 y 5 niveles

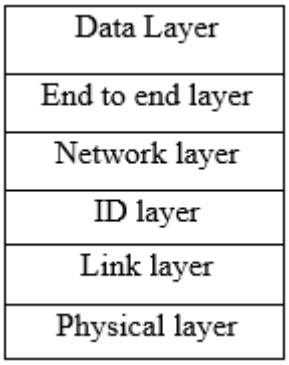


Figura 4. Modelo en capas para IoT
Fuente. (Romero, 2016)

La comparación de desempeño propuesto en este trabajo, se enfoca en protocolos de la capa de aplicación; por este motivo se realizó la investigación pertinente en cuanto a protocolos de dicha capa.

2.11. Protocolos de aplicación para entornos IoT

Estos protocolos permiten la interacción de las aplicaciones con los protocolos de la capa inferior para el envío de los datos en la red. Los datos de esta capa se codifican por el protocolo correspondiente y se encapsulan en la capa de transporte.(R. Martínez, 2016)

Los protocolos para entornos IoT se desarrollan con base en la necesidad de los objetos, y lograr el intercambio de información; se dividen en dos categorías: (Semle, 2016)

Cliente/servidor: Estos protocolos necesitan que el cliente establezca conexión con el servidor y realice solicitudes.

Publish/Suscribe – Publicar/Suscribir: En este tipo de protocolos se requiere que el dispositivo se conecte a un tópico de gestión intermediaria de modo que la información se publique.

De acuerdo a las categorías que se mencionan, se tiene varios protocolos que se desarrollaron para la capa de aplicación, entre los que se puede mencionar:

OPC UA: (Unified Architecture – Arquitectura Unificada) proporciona una interfaz estándar para la comunicación con PLC (Programmable Logic Controller – Controlador Lógico Programable). Es del tipo cliente/servidor, el término UA indica que la comunicación se origina desde la capa de aplicación hacia la capa de transporte.

HTTP: (Hypertext Transfer Protocol) permite la trasferencia de información en la web, sigue el modelo cliente – servidor. Este protocolo proporciona transporte, sin embargo, no logra la debida presentación de la información. Sus aplicaciones son para la lectura de temperaturas frecuentes.

XMPP: (Extensible Messaging and Presence Protocol) protocolo de mensajería instantánea para aplicaciones como multichat, videoconferencia, llamadas, se caracteriza por ser open source. Se usa para control de acceso con cifrado hop by hop. (González, 2017)

AMQP: (Advanced Message Queuing Protocol) se usa para el intercambio de mensajes en plataformas diferentes, asegura la información en entornos distribuidos en la Nube. (González, 2017)

DDS: (Data Distribution Service) se usa en el borde de la red, no requiere de gestión centralizada, se recomienda para aplicaciones M2M (Machine to machine)(Semle, 2016)

2.12. Elección de protocolos de aplicación para comparación

Los protocolos CoAP y MQTT son aquellos cuyo uso es frecuente en entornos IoT para servicios dentro del hogar. Por tal razón se eligieron para someterlos a comparación y definir el mejor desempeño. Estos son:

CoAP: (Constrained Application Protocol – Protocolo de aplicación limitado) Este protocolo mantiene sobrecarga baja de mensajes, de modo que no se requiere la fragmentación de los mismos. Usa UDP/multicast, es similar a HTTP, a diferencia que reduce su encabezado. Se usa en dispositivos de borde y para aplicaciones M2M, se recomienda usar CoAP cuando HTTP requiera de gran ancho de banda.(Semle, 2016). Su modelo de gestión de servicios se basa en que el cliente se interese por un recurso específico, para ello solicita al servidor dicho recurso, este responde con una representación del estado del recurso en el instante en que recibe la petición. Su categoría el cliente/servidor. Define cuatro tipos de mensajes: Confirmable (CON), Non-Confirmable (NON), Acknowledgement (ACK), Reset (RST). (Gimenez, 2013)

MQTT: (Message Queuing Telemetry Transport – Mensaje de cola de transporte telemétrico) se usa para redes remotas y sistemas SCADA, su encabezado es de 2 bytes. Se usa este protocolo cuando no se conozca a detalle la infraestructura y se asegura la comunicación con TLS (Transport Layer Security). (Semle, 2016) Si se realiza una analogía con el modelo OSI este protocolo se sitúa en las capas superiores (Figura 5)

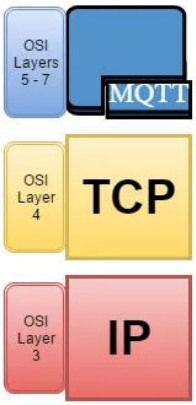


Figura 5. MQTT en modelo OSI
Fuente. (Semle, 2016)

Protocolo Publish/Suscribe, no requiere conocer el origen ni el destino al que se dirigen los mensajes, por esta razón su cabecera se reduce, cuya comunicación se la realiza bajo demanda (Figura 6).

Existe una implementación de tres niveles de QoS (Quality of Servicio-Calidad de servicio) con lo que se garantiza la entrega de los mensajes. Se usa para conexiones M2M con modos de gestión sencillos.(Ermesh, 2018)

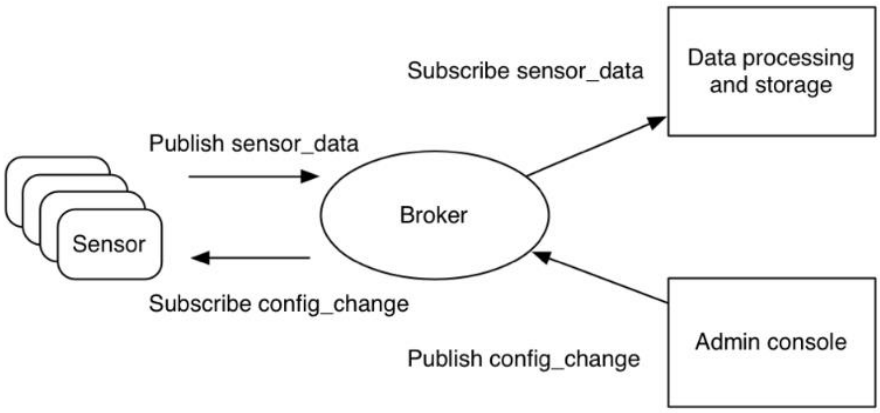


Figura 6. MQTT en IoT
Fuente. (Semle, 2016)

CAPÍTULO III

PROTOCOLOS DE APLICACIÓN IoT SUJETOS A COMPARACIÓN

IoT posee una arquitectura que se divide en capas, similar al modelo OSI, dicha arquitectura cuenta con 5 capas, como ya se revisó en el capítulo anterior. Y son:

1. Capa de borde
2. Capa de acceso
3. Capa de Internet
4. Capa middleware
5. Capa de aplicación

Las primeras 2 capas actúan para la adquisición de datos, mientras que las 2 últimas son para la utilización de estos datos para las aplicaciones. (D. Martínez & Guillen, 2015)

En el desarrollo de entornos IoT, sobretodo en el sector de la industria se ve obligada la implementación de comunicaciones M2M, que se convierten en un concepto y tecnología recurrente en investigación y desarrollo, puesto que compone una solución que implica reducción de costos y la inclusión de nuevos modelos de negocio. Este tipo de comunicaciones pueden servir para buscar la estandarización de protocolos de IoT (Melián & Anías, 2015).

3.1. Protocolo CoAP

CoAP (Constrained Application Protocol – Protocolo de aplicación restringida) este protocolo pertenece a la capa de aplicación, se orienta su uso para redes limitadas compuestas de sensores de baja potencia, interruptores o componentes que se deban controlar remotamente y con comunicación M2M. (Rodríguez, 2016) Se basa en el método REST (Representational State Transfer – Representación de estado de transferencia), es decir se compone de clientes y servidores; los clientes son los que envían peticiones y los servidores responden a ellas con una representación del recurso en solicitud, además usa DTLS (Datagram Transport Layer Security– Seguridad en datagramas de capa de transporte) para la protección de la comunicación y equivalente a claves RSA de 3072 bits. (Gimenez, 2013), (Semle, 2016), (Añez, 2018)

CoAP se asemeja a HTTP (Hypertext Transfer Protocol – Protocolo de trasferencia de hipertexto) sin embargo, usa protocolo de transporte UDP/multicast; incluso el encabezado de HTTP es de menor tamaño, así como sus requerimientos. (Semle, 2016) El cliente realiza una petición, con una opción que indica el método que se utilizará para la solicitud de cierto recurso, es decir este debe ser identificado por un URI (Uniform Resource Identifier – Identificador de recurso uniforme); el servidor correspondiente envía una respuesta con un código que respecta al recurso, esta comunicación entre el cliente y el servidor es asíncrona mediante el protocolo UDP.

3.1.1. Características

- Para dispositivos embebidos y poca memoria.
- Facilidad para mapear a HTTP.
- Transmisión de mensajes asíncronos.

- Fiabilidad por unicast y apoyo multicast de peticiones.
- Menor complejidad de análisis de mensaje por el tamaño reducido de las cabeceras.
- Apoyo de URI y contenido.
- Búsqueda de recursos y mecanismos de suscripción opcional.

3.1.2. Estructura de mensaje

Los mensajes se estructuran, por una cabecera de un tamaño fijo, un número de opciones las mismas que pueden ser variables (Figura 7) y su payload o carga útil. (Gimenez, 2013)

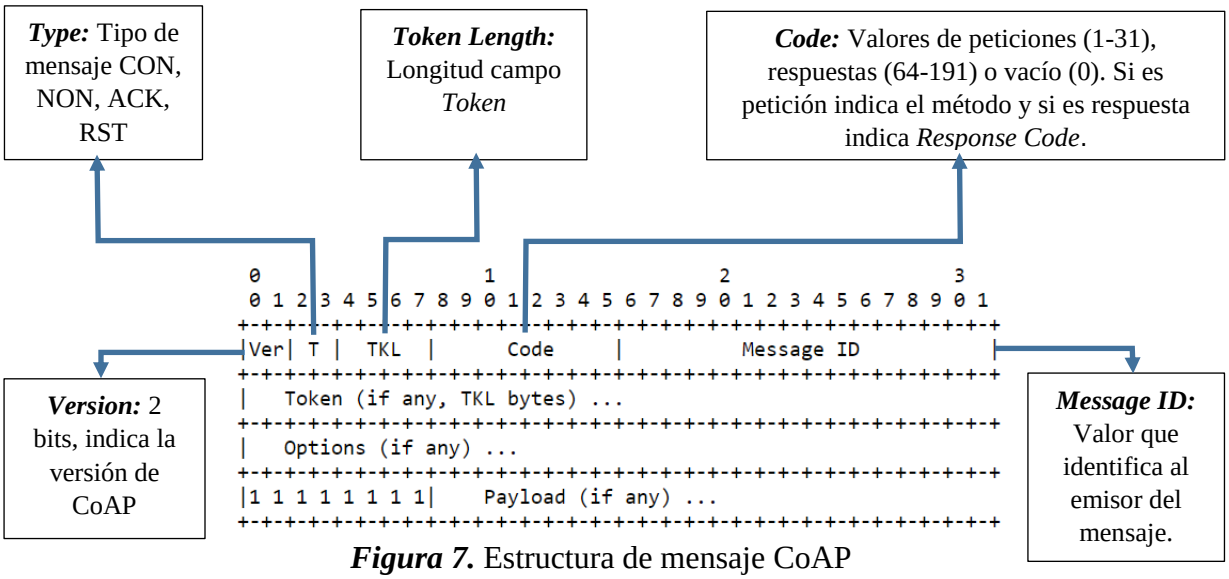


Figura 7. Estructura de mensaje CoAP
Fuente. (Shelby et al., 2014)

3.1.3. Mensajes

Los mensajes en este protocolo son:

Confirmable (CON): Estos mensajes requieren de confirmación de recepción es decir un *ACK*, el mismo que debe enviar el receptor. (Gimenez, 2013)

No-Confirmable (NON): Se envía este mensaje cuando no es obligatorio que el receptor reciba el mensaje, por ello no se requiere confirmación. Se usa por lo general para lecturas repetitivas del valor de un sensor. (Gimenez, 2013)

Asentimientos o Reconocimiento (ACK): Estos mensajes se envían para la confirmación de la recepción de un mensaje *CON*, o en su defecto para responder a una petición *GET*. (Gimenez, 2013)

Reset (RST): Es el mensaje que se responde por la recepción de un mensaje *CON* o *NON*, que el receptor no tiene la capacidad de procesarlo. Estos eventos se pueden presentar al reiniciar un equipo y éste pierde información, con respecto al estado que podría permitirle interpretar el mensaje correctamente. (Gimenez, 2013)

3.1.4. Opciones

Las opciones que contiene un paquete de CoAP están seguido de la cabecera; deben seguir un orden de acuerdo a los ON (Option Number – Números de Opciones), si corresponden a números impares se refiere a opciones críticas, mientras que si contiene números pares son opcionales (Figura 8). Las opcionales son aquellas que ignoran mensajes que no contienen opciones reconocidas; mientras que las críticas se dan cuando un mensaje de petición *CON* se regresa por un código de respuesta 402, o a su vez si un mensaje de respuesta de tipo *CON* y un *NON* se ignoran. (Castro, 2014)

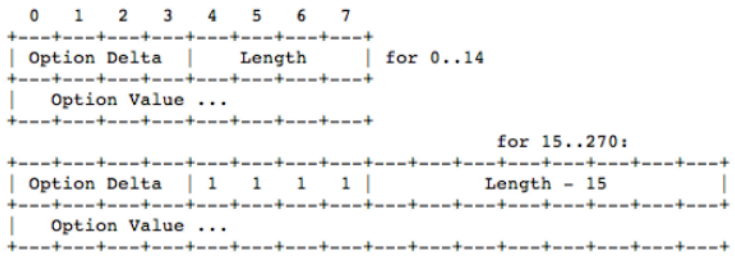


Figura 8. Estructura de opciones de CoAP
Fuente. (Shelby et al., 2014)

Se tienen algunas opciones, entre las que se pueden mencionar son:

Option Delta: Indica la diferencia entre el número de opción actual y anterior. En la primera opción este campo es 0, y en la recepción se usa este valor para recuperar las opciones siguientes que contiene el mensaje.

Length: Corresponde al tamaño de bytes de *Option Value*. Este campo tiene un tamaño de 4 bits, puede indicar un valor de 0 a 14 bytes, en caso de que sea 15 se añade un byte para modificar la longitud de 15 a 270 bytes.

Option Value: Es el tamaño y formato del campo, en un paquete de CoAP las opciones están seguido de la cabecera; deben seguir un orden de acuerdo a los (Option Number – Números de Opciones).

Token: Asocia peticiones con respuestas que tengas relación.

Uri – Host: Especifica solicitud de host de Internet del recurso.

Uri – Port: Número de puerto del recurso.

Uri – Path: Segmento de la ruta absoluta hacia el recurso.

Uri – Query: Es una cadena de consulta.

Proxy – Uri: Se usa para una solicitud a un proxy.

Content – Type: Indica el formato de representación de la carga útil, de mensaje, se representa en un valor.

Max – Age: Es el tiempo de respuesta en caché del cliente para esperar la respuesta de un servidor.

E – Tag: Es una respuesta que indica el valor actual de la entidad del tag, para la representación del recurso destino. La entidad del tag es un identificador de recursos a nivel local, se usa para diferenciar las representaciones de los recursos.

Location – Path y Location – Query: Representa la ubicación de un recurso, que se crea mediante un POST, y con la ruta URI absoluta.

If – Match: Se usa para solicitudes condicionales de una actual, existente o en su defecto de un valor *E – Tag* de una o varias representaciones del recurso destino.

If – None – Match: Se usa para peticiones de creación de recursos con métodos *PUT*, y también para proteger a *IDs* de sobre escrituras por parte de clientes cuando estos se ejecutan al mismo tiempo. Las opciones de este protocolo poseen formato y tamaño que se deben tomar en cuenta la momento de usarlas (Tabla 1). Además, para un mejor manejo se identifican mediante las siguientes letras:

- C (Critical – crítica)
- U (Usafe – insegura)
- N (NoCacheKey – sin clave en caché)
- R (Repeteable – repetible).

Tabla 1
Opciones de protocolo CoAP

N°	C	U	N	R	Nombre	Formato	Tamaño
1	x			x	If – Match	Opaque	0 – 8
3	x	x			Uri – Host	String	1 – 255
4				x	E – Tag	Opaque	1 – 8
5	x				If – None – Match	Empty	0
7	x	x			Uri – Port	Uint	0 – 2
8				x	Location – Path	String	0 – 255
11	x	x		x	Uri – Path	String	0 – 255
12					Content – Format	Uint	0 – 2
14		x			Max – Age	Uint	0 – 4
15	x	x		x	Uri – Query	String	0 – 255
17	x				Accept	Uint	0 – 2
20				x	Location – Query	String	0 – 255
35	x	x			Proxy – Uri	String	1 – 1034
39	x	x			Proxy – Scheme	String	1 – 255
60			x		Size1	Uint	0 – 4

Fuente: (Shelby et al., 2014)

3.1.5. Métodos

Método GET: Este método recupera la representación de información que corresponde al recurso identificado por el URI de una solicitud, si esta solicitud contiene una opción de aceptación, se refiere al formato del contenido de la respuesta. Por ejemplo, si una solicitud contiene la opción E – Tag, este método solicita que se valide E – Tag y que la representación se transfiera solo si la validación falla, sus códigos de respuesta son 2.05 (Content) o 2.03 (Valid).(Gimenez, 2013) (Shelby et al., 2014) Es un método seguro e ídem – potente, es decir que posee propiedades matemáticas e informáticas las mismas que se pueden usar varias veces sin cambiar su resultado en la aplicación inicial.

Método POST: La representación del recurso que se encuentra incluida con la petición debe ser procesada; de modo que el servidor responderá con un código 2.01 (Created) en caso de que se

encuentre creado un nuevo recurso en el servidor e incluirá en la respuesta el URI en la que se aloja dicho recurso. El URI se indica con una o varias opciones Location – Path y/o Location - Query. En caso de que el recurso no se pueda crear, debido a que ya se creó anteriormente, el servidor responderá con un código 2.04 (Changed), es decir será modificado con las especificaciones de la petición. No es seguro y no es ídem – potente.(Gimenez, 2013)

Método PUT: Este método solicita que el recurso que identificó el URI de la solicitud se actualice o se cree con la representación de la información incluida en la petición. En caso de existir un recurso en el URI de la solicitud, la representación adjunta debe contener una versión modificada de dicho recurso y adicionalmente debe devolver un código de respuesta 2.04 (Changed). En el caso de que no exista ningún recurso el servidor puede crear un nuevo recurso con el URI de la solicitud, para ello se usa el código 2.01 (Created). Si luego de este proceso no se logra obtener ninguno de los resultados anteriores, se debe enviar un código de respuesta de error adecuado. Este método no es seguro; sin embargo, sí es ídem – potente.(Shelby et al., 2014)

Método DELETE: Este método solicita que elimine un recurso identificado por el URI de la solicitud. Se usa el código de respuesta 2.02 (Deleted) en caso de que el recurso no exista antes de la solicitud. Este método no es seguro, pero sí es ídem – potente. (Shelby et al., 2014)

3.1.6. Solicitudes y respuestas

CoAP es similar al protocolo HTTP, ya que el modelo se basa en solicitudes y respuesta entre clientes y servidores; sin embargo, en CoAP no se envían las solicitudes y respuestas a una conexión anterior, sino que se intercambian entre sí de manera asíncrona mediante mensajes. (Shelby et al., 2014)

En las solicitudes y respuestas intervienen los métodos *GET*, *POST*, *PUT* y *DELETE* que se especificaron anteriormente, junto con los mensajes *CON* y *NON*. Como se indicó CoAP es un protocolo que se basa en el intercambio de mensajes asíncronos de un cliente a un servidor, el servidor responde a las peticiones, el mismo que debe indicar que recibió o no el mensaje. (Gimenez, 2013) En una petición *GET* con mensaje de tipo *CON*, se realiza la solicitud sobre el recurso, el servidor devuelve la respuesta 2.05 (Content) junto con el mensaje de acuse de recibo *ACK*, que se confirma la petición *CON*. (Figura 9). En este tipo de peticiones se pueden tener interacciones de mensajes NON (Figura 10).

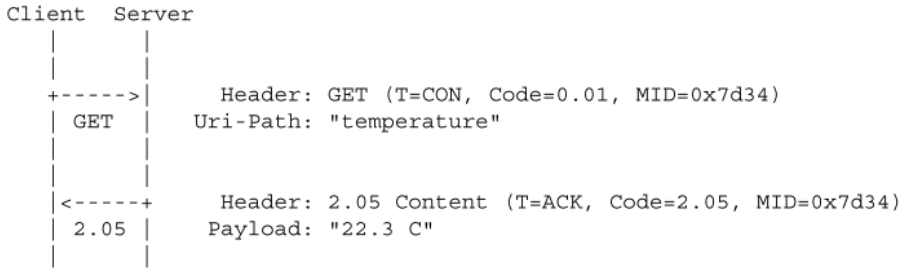


Figura 9. Petición GET y mensaje CON
Fuente. (Shelby et al., 2014)

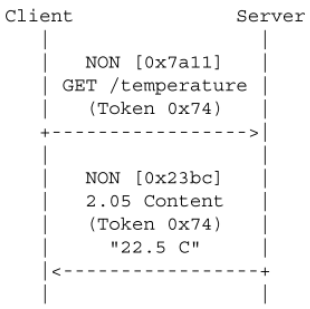


Figura 10. Interacción de mensajes de tipo NON
Fuente. (Shelby et al., 2014)

Existen también peticiones de tipo *GET* con mensajes no confirmables *NON*, se ejecuta la petición para solicitar la representación de información; sin embargo, no se recibe un acuse de

recibo *ACK*, sólo se obtiene un código de respuesta 2.05 (Content) que se encuentra en el contenido de un mensaje no confirmable NON. (Figura 11) (Añez, 2018)

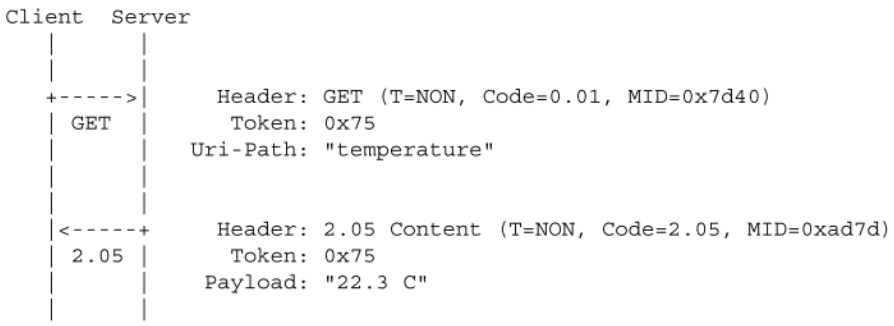


Figura 11. Petición GET y mensaje NON
Fuente. (Shelby et al., 2014)

Para la interacción asíncrona de mensajes se requiere de cierta coordinación entre los recursos, para ello se tienen técnicas que se usan de acuerdo a la aplicación y la necesidad.

Estas técnicas son las siguientes:

Piggy – backing: Se basa en la respuesta inmediata del servidor a una petición de tipo *CON*, y a su vez envía un *ACK* (Figura 12). (Castro, 2014)

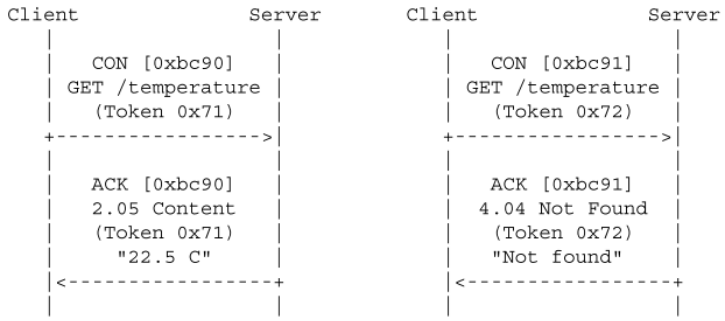


Figura 12. Peticiones GET confirmables y respuestas Piggy – backing
Fuente. (Shelby et al., 2014)

Separate: Se usa cuando se recibe una petición y el servidor no es capaz de responder de inmediato, ya sea porque no tiene acceso al recurso o por saturación. Las respuestas de tipo NON se envían

ya que no existen mensajes ACK para este tipo de peticiones. Si la petición se obtiene con un mensaje CON y el servidor no puede responder de inmediato, responde con un mensaje ACK, para confirmar la recepción de la petición y que se atenderá a la brevedad; tiempo después la respuesta se envía con el contenido del recurso con un mensaje CON, el mismo que debe confirmarse por el cliente; se debe tomar en cuenta que la confirmación y el acuse de recibo no llegan necesariamente al mismo tiempo (Figura 13) y (Figura 14).(Gimenez, 2013)

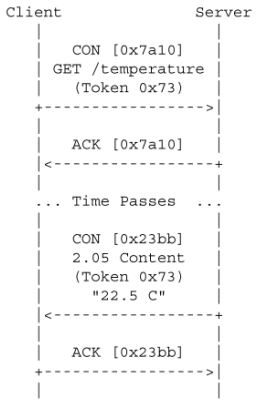


Figura 13. Petición GET y respuesta Separate
Fuente. (Shelby et al., 2014)

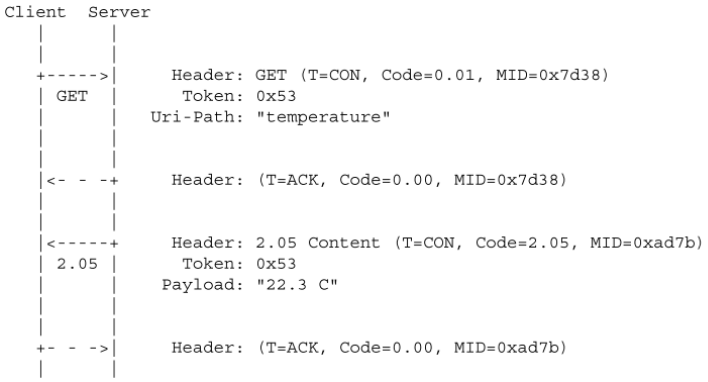


Figura 14. Respuesta confirmable y Separate
Fuente. (Shelby et al., 2014)

Token: Se trata de un valor que se gestiona localmente, con el fin de diferenciar frecuentemente las peticiones en curso. Este valor se actualiza, de manera que todas las peticiones pendientes de un cliente tomen un valor diferente. (Castro, 2014).

3.1.7. Almacenamiento en caché

La memoria caché almacena las respuestas de cada petición, las mismas que se pueden usar, siempre y cuando cumplan las siguientes condiciones:

- El método de petición y de respuesta deben ser iguales.
- Las opciones que contenga la petición debe ser igual a la original, a excepción de Max – Age o E – Tag.
- La respuesta que se almacena debe validarse o en su defecto no debe ser caduca.

Con estas condiciones se optimiza el uso de la red, de modo que no envíe paquetes redundantes. Además, las respuestas se pueden reusar cuando existan peticiones de iguales características a la inicial u original. Este tipo de almacenamiento en caché permite que el tiempo de respuesta disminuya, y a su vez el ancho de banda; sin embargo, existen dos modelos que permiten decidir si la caché se puede utilizar para atender una petición, y son:

Modelo Freshness: El término *Fresh* se refiere a que, cada vez que en la memoria caché exista una respuesta nueva, ésta se puede usar para atender peticiones posteriores. El servidor de origen proporciona un tiempo de caducidad, de modo que al usar la opción Max – Age que corresponde a la edad máxima, sea capaz de indicar que la respuesta se debe considerar desactualizada, una vez que su valor sea mayor, se debe especificar en segundos, cabe indicar que el valor predeterminado de la edad máxima es de 60 segundos; en caso de que este valor sea igual a cero, quiere decir que el servidor origen no desea el almacenamiento en caché. (Shelby et al., 2014)

Modelo de validación: Se usa este modelo en caso de que existan varias respuestas en almacenamiento para una solicitud *GET* y no se puedan utilizar porque no son actuales, en estos

casos se puede usar la opción *E – Tag* dentro de la petición *GET*, de modo que se especifique el origen, es decir el servidor puede elegir la respuesta en almacenamiento o a su vez puede actualizar dicha respuesta. La respuesta 2.03 (valid) indica que la respuesta en almacenamiento que se identificó a partir de *E – Tag* se puede reusar, una vez que cumpla con su actualización correspondiente. Otros códigos de respuesta pueden significar que ninguna de las respuestas en almacenamiento es la correcta. (Shelby et al., 2014)

3.2. Protocolo MQTT

MQTT (Message Queue Telemetry Transport – Transporte de telemetría de cola de mensajes) es un protocolo de telemetría que corresponde a la capa de aplicación en la arquitectura de IoT, es abierto, de fácil implementación y lo desarrolló la compañía IBM y Eurotech hoy en día se encuentra en estandarización por el grupo OASIS. Su arquitectura se basa en el modelo cliente y servidor, además se incluye un concepto que se denomina Publish/Subscribe – Publicar/Subscribir. Este protocolo es óptimo para aquellos dispositivos que cuentan con memoria y batería limitada, para redes, entornos restringidos e inseguros y redes con ancho de banda reducido y alta latencia. (Osako et al., 2016).

3.2.1. Modelo Publish/Subscribe

MQTT se basa en este modelo que necesita de la actuación de un agente central para la administración y enrutamiento de los mensajes entre los nodos de la red de MQTT. Usa protocolo de transporte TCP, además también se usan para comunicaciones M2M. (Jaffey, 2014)

Debido que MQTT usa el modelo cliente servidor, cada uno de los sensores de la red constituye un cliente los mismos que tienen conexión a un servidor al mismo que se lo llamará *broker* o

intermediario. Los mensajes corresponden a datos discretos los mismo que para el servidor son invisibles; cada mensaje se publica en un *topic* que no es más que una dirección o un tema, los clientes pueden suscribirse a varios *topics,* y cada cliente suscrito recibe mensajes publicados sobre aquel *topic* o dirección. (Jaffey, 2014) Por ejemplo, en una red de sensores de humedad se publicarán las mediciones que cada sensor realice, los mismos que se suscribirán al *topic* que se asigne a varias mediciones, mientras que, una consola de administración recibe los comandos que envía el administrador del sistema para realizar ajustes en las configuraciones de los sensores, parámetros como, la frecuencia de medición, sensibilidad y publicación. (Yuan, 2017b)

3.2.2. Términos de MQTT

A continuación, se describen algunos de los mensajes que se intercambian en el protocolo MQTT.

Cliente: Dispositivo que solicita la conexión con el servidor, publica los mensajes de aplicación con otros clientes que pueden estar interesados con dicho tema. Además, suscribe la solicitud en los mensajes de aplicación que pueden estar interesados en recibir. Otra de sus funciones es dar por terminada la suscripción para eliminar las solicitudes existentes y finalmente cierra la conexión con el servidor.

Servidor: Se trata de un sistema o dispositivo que actúa como intermediario entro los clientes que publican los mensajes de aplicación con los clientes que realizan su suscripción. Las funciones de un servidor se basan en aceptar la conexión que solicitan los clientes, aceptar las publicaciones de los mensajes de aplicación de los clientes, procesa y elimina las suscripciones de los clientes y reenvía los mensajes de aplicación que coinciden con las suscripciones de los clientes. (Oasis, 2018)

Session: Sesión e interacción entre un cliente y un servidor, unas sesiones pueden durar mientras dura la conexión de la red, mientras que otras pueden contener múltiples sesiones consecutivas entre el cliente y el servidor. (Oasis, 2018)

Topic Name: Es el nombre del tema, que se trata de una etiqueta que se adjunta a los mensajes de aplicación los mismos que deben suscribirse. (Sengul & Kirby, 2017)

Topic Filter: Se refiere al filtro del tema, que no es más que una expresión que indica que existe interés por uno o múltiples temas, éstos filtros puedes llevar *wildcards* o máscaras inversas. (Sengul & Kirby, 2017)

Subscription: Se refiere a la suscripción que posee el filtro del tema, y un valor máximo de calidad de servicio (QoS).(Sengul & Kirby, 2017)

Application Message: Es el mensaje de aplicación, es decir son los datos que transportan en el protocolo MQTT, tienen de igual forma un nivel de QoS y el nombre del tema. (Sengul & Kirby, 2017)

Shared Subscription: Se refiere a la suscripción compartida, que se compone de un *Filter Topic* y un valor de QoS máximo. Se puede asociar con más de una sesión de modo que permita un intercambio de mensajes considerable. Un mensaje de solicitud que pertenece a una suscripción compartida solamente se envía a un cliente asociado a una de esas sesiones. Una sesión puede suscribirse a más de una sesión compartida; además puede contener suscripciones compartidas y no compartidas. (Oasis, 2018)

Wildcard Subscription: Se trata de una suscripción con un *Filter Topic*, que posee una o varias máscaras inversas, lo que permite que la suscripción coincida con varios nombres de tema o *Name Topic*. (Oasis, 2018). MQTT envía varios paquetes de control (Tabla 2)

Tabla 2

Mensajes de protocolo MQTT

Mensaje	Numeración	Dirección de flujo	Descripción
RESERVED	0	No se ocupa	Reservado. (Martos, 2011)
CONNET	1	Cliente a servidor	Es la solicitud del cliente para conectarse al broker, luego de que la conexión se establece, este mensaje es el primer paquete enviado por un cliente. (Sengul & Kirby, 2017)
CONNACK	2	Servidor a cliente	Es el mensaje de acuse de recibo o ACK, es el paquete que envía el intermediario a un cliente; y contienen códigos de retorno que especifica el estado de error o de éxito al cliente. (Sengul & Kirby, 2017)
PUBLISH	3	Cliente a servidor o viceversa	Se publica el mensaje o paquete que se puede enviar desde el cliente al intermediario o viceversa. (Sengul & Kirby, 2017)
PUBACK	4	Cliente a servidor o viceversa	Es la respuesta a *Publish*, con nivel 1 de QoS, envía el intermediario al cliente o viceversa. (Sengul & Kirby, 2017)
PUBREC	5	Cliente a servidor o viceversa	Es la respuesta de *Publish* con nivel 2 de QoS, envía el intermediario al cliente o viceversa. (Sengul & Kirby, 2017)
PUBREL	6	Cliente a servidor o viceversa	Mensaje *Publish* en lanzamiento. (Martos, 2011)
PUBCOMP	7	Cliente a servidor o viceversa	Mensaje *Publish* completo. (Martos, 2011)
SUBSCRIBE	8	Cliente a servidor	Es la solicitud de suscripción del cliente. (Sengul & Kirby, 2017)
SUBACK	9	Servidor a cliente	Reconoce la suscripción solicitada por un cliente. (Sengul & Kirby, 2017)
UNSUBSCRIBE	10	Cliente a servidor	Solicitud de un cliente para eliminar la suscripción. (Martos, 2011)
UNSUBACK	11	Servidor a cliente	Reconocimiento de la eliminación de la suscripción. (Martos, 2011)
PINGREQ	12	Cliente a servidor	Ping de solicitud. (Martos, 2011)
PINGRESP	13	Servidor a cliente	Ping de respuesta. (Martos, 2011)
DISCONNECT	14	Cliente a servidor o viceversa	Mensaje de un cliente que se desconecta. (Martos, 2011)
AUTH	15	Cliente a servidor o viceversa	Autenticación Exchange.

Fuente: (Oasis, 2018)

3.2.3. Estructura de paquetes de control

Los paquetes de control tienen tres niveles en su estructura. (Figura 15)

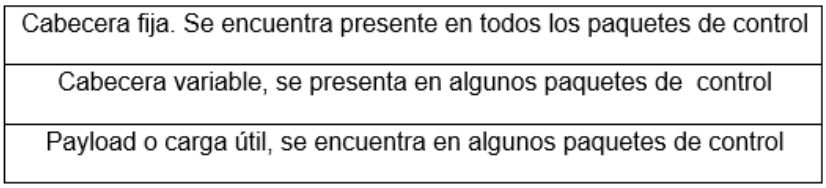


Figura 15. Estructura de paquetes de control de protocolo MQTT
Fuente. (Oasis, 2018)

Los mensajes de control contienen la misma cabecera fija. (Figura 16)

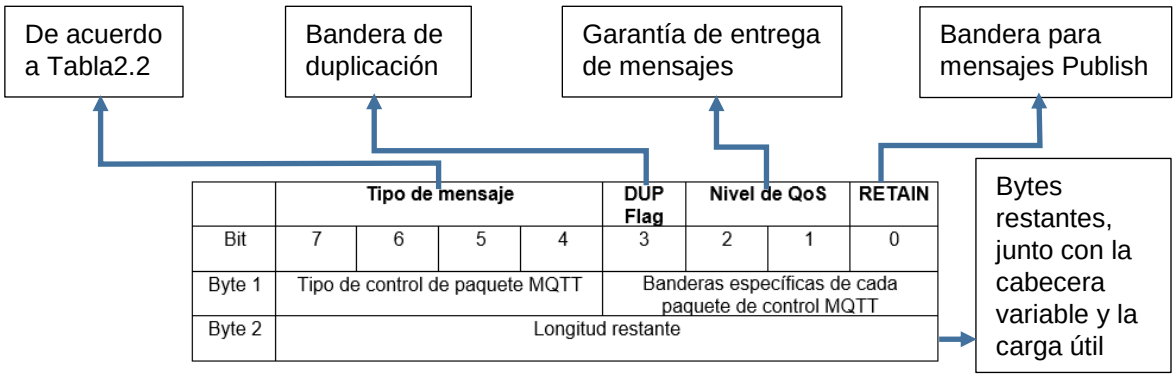


Figura 16. Cabecera de paquetes de control de MQTT
Fuente. (Oasis, 2018), (Martos, 2011)

3.2.4. DUP Flag

El término DUP se refiere a la duplicación, se usa esta bandera para cuando el cliente o el servidor intenta reenviar un mensaje *PUBLISH*, *PUBREL*, *SUBSCRIBE* o *UNSUBSCRIBE*; se asigna al mensaje con un nivel de QoS mayor a cero, o en su defecto necesita reconocer el mensaje. (Martos, 2011)

3.2.5. RETAIN

Se usa en caso de que un cliente envíe un mensaje de tipo *PUBLISH* a un servidor, con la bandera RETAIN activa, esta se almacena en el servidor, inmediatamente después de que se entregue la información a los suscriptores. En una nueva suscripción a un tópico, el último mensaje que se almacena para este tópico se envía al nuevo suscriptor siempre y cuando esta bandera se encuentre activa. (Rodrigues & Zem, 2016)

3.2.6. Calidad de servicio QoS

Dentro del protocolo MQTT, se habla de niveles de QoS, los mismos que garantizan la entrega de los mensajes *Publish*, además del encabezado y de la carga útil.

Nivel 0 de QoS: Se conoce a las acciones como Fire and Forget - Activar y Olvidar, máximo una vez, y comprende la transmisión de una ráfaga de mensajes sin que se garantice su llegada, su uso se centra para los mensajes repetitivos y sin criticidad.(Stansberry, 2015)

Nivel 1 de QoS: En este nivel garantiza que el receptor reciba el mensaje al menos una vez. Al recibir y comprender el destinatario dicho mensaje, éste envía un acuse de recibo PUBACK a quien realizó la publicación. Mientras el editor recibe el PUBACK, se almacena y retransmite el mensaje de forma periódica. Estos mensajes sirven para cerrar un nodo crítico. (Stansberry, 2015)

Nivel 2 de QoS: En este nivel se garantiza que el destinatario reciba y decodifique el mensaje. Este nivel es el más seguro, en el que el editor envía un mensaje, el destinatario recibe y decodifica, lo que quiere decir que está presto a recibir el mensaje, en ese momento el editor envía su mensaje. Posterior a ello, el destinatario debe comprender el mensaje para enviar un acuse de recibo. (Stansberry, 2015)

Los niveles de QoS, se manejan con los siguientes valores de bits. (Tabla 3)

Tabla 3
Niveles de QoS para protocolo MQTT

Valor de QoS	Bit 2	Bit 1	Descripción
0	0	0	< = 1
1	0	1	< = 1
2	1	0	= 1
3	1	1	No se usa

Fuente: (Oasis, 2018)

3.2.7. Autorización para establecimiento de conexión

En MQTT para que una conexión se establezca debe existir cierta interacción entre los diversos componentes que cumplen las siguientes funciones: publicador, suscriptor, intermediario o broker, y el servidor y su autorización. Se tienen 3 tipos de autorizaciones:

- Autorización para el establecimiento de la conexión entre clientes y el broker o intermediario.
- Autorización para la publicación de mensajes de los publicadores a un componente intermediario o broker, y del intermediario a los suscriptores.
- Autorización de los mensajes de suscripción desde los suscriptores al intermediario o broker.

Cada tópico se maneja a modo de recursos, es decir que los clientes tanto del publicador como de los suscriptores conocen de los tópicos de su interés. Se debe considerar que cada solicitud de conexión contiene un *TOKEN* el mismo que especifica los permisos del cliente (Figura 17). (Sengul & Kirby, 2017)

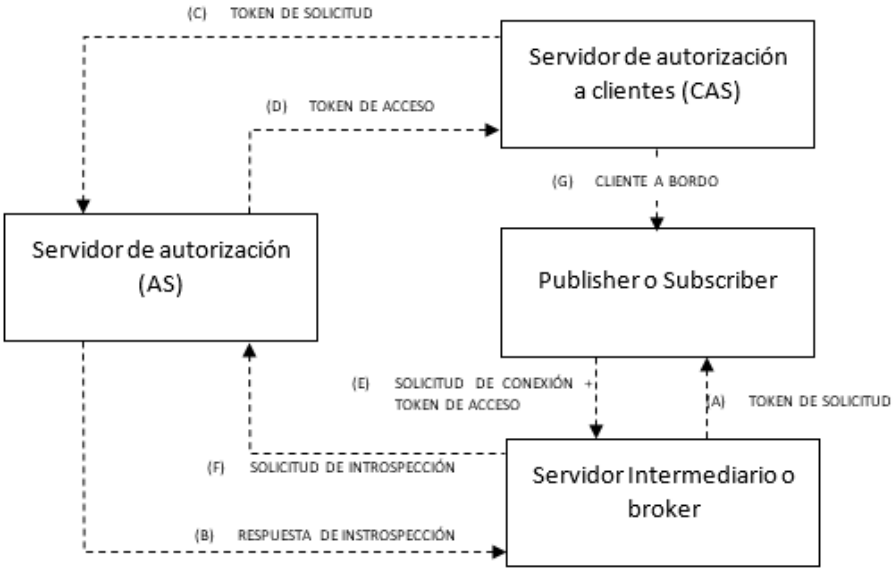


Figura 17. Establecimiento de conexión en MQTT

3.3. Comparativo entre CoAP y MQTT

Para una mejor visualización de las diferencias y semejanzas de estos protocolos de capa de aplicación para IoT, se detalla una tabla. (Tabla 4)

Tabla 4
Comparación de protocolos CoAP y MQTT

	CoAP	**MQTT**
Tipo de comunicaciones	M2M	M2M
Modelo de arquitectura	Cliente / Servidor	Publisher / Subscriber
Protocolo de transporte	UDP	TCP
Comunicación	Modelo REST (request / response)	Temas o tópicos en el bróker
Seguridad	DTLS	SSL / TLS
Tipo de Payload o carga útil	Binaria	Binaria
Escalabilidad	Compleja	Simple

CAPÍTULO IV

IMPLEMENTACIÓN DE PROTOCOLOS MQTT Y CoAP EN RASPBERRY PI

4.1. Materiales y herramientas

Para la implementación de este proyecto, se inició con la comparación de dos protocolos importantes para IoT; a partir de un prototipo básico que, consta en encender y/o apagar una lámpara doméstica que funcione a 110V, mediante el accionar de un switch o botón, y al hacerlo enviará un pulso o 5 voltios, para encenderlo; y 0 voltios para apagarlo.

Los materiales y herramientas a usar son los siguientes:

- **Servidor en la nube.** Se creó una cuenta en DigitalOcean, cuyo valor mensual es de $5.
- **Raspberry Pi 3 B+.** Microordenador que permite la implementación de varios entornos y aplicaciones de IoT.
- **Circuito opto acoplador.** Es un circuito constituido por un relé y transistores de modo que exista control del aislamiento eléctrico tanto en las entradas y salidas GPIO de la Raspberry.
- **Lámpara doméstica.** Su funcionamiento debe ser con 110V, junto con un foco convencional.
- **Node – RED.** Motor de flujos que sirve para generar comunicación entre servicios y hardware. Usa el puerto 1880 y se puede observar el flujo a través de una interfaz gráfica.
- **MobaXterm.** Herramienta para conexión entre el computador y el servidor virtual en la nube.
- **Wireshark.** Herramienta para captura y análisis de protocolos dentro de una red
- **Jperf.** Software para inyección de tráfico y verificación de ancho de banda.

Además de estas importantes herramientas también es preciso mencionar que se utilizaron cables de red, cables de poder, extensiones eléctricas, cables tipo jumper, cargador 5V de celular, monitor, teclado, mouse; así como la instalación de los debidos paquetes tanto en el servidor en la Nube como en la Raspberry.

4.2. Configuración de droplet en DigitalOcean

DigitalOcean es un proveedor de servidores virtuales en los Estados Unidos, los servidores se denominan Droplets, cada uno de ellos son privados, ya que DigitalOcean no interviene en ningún momento y bajo ningún concepto en las instalaciones que se realicen en los mismos. A continuación, se describe el proceso para crear un droplet de acuerdo a la necesidad de este proyecto.

Se creó una cuenta en DigitalOcean, en este proceso se ingresan datos como correo, ubicación, teléfono etc. Luego de crear la cuenta, se ingresan los datos de la tarjeta de crédito o débito con la que se realizarán los pagos.

Posteriormente se elige crear nuevo droplet, para este caso se eligió el sistema operativo Ubuntu con 1 GB de procesamiento, 25 GB en disco, y 1000 GB de transferencia, cuyo costo mensual es de 5 dólares (Figura 18).

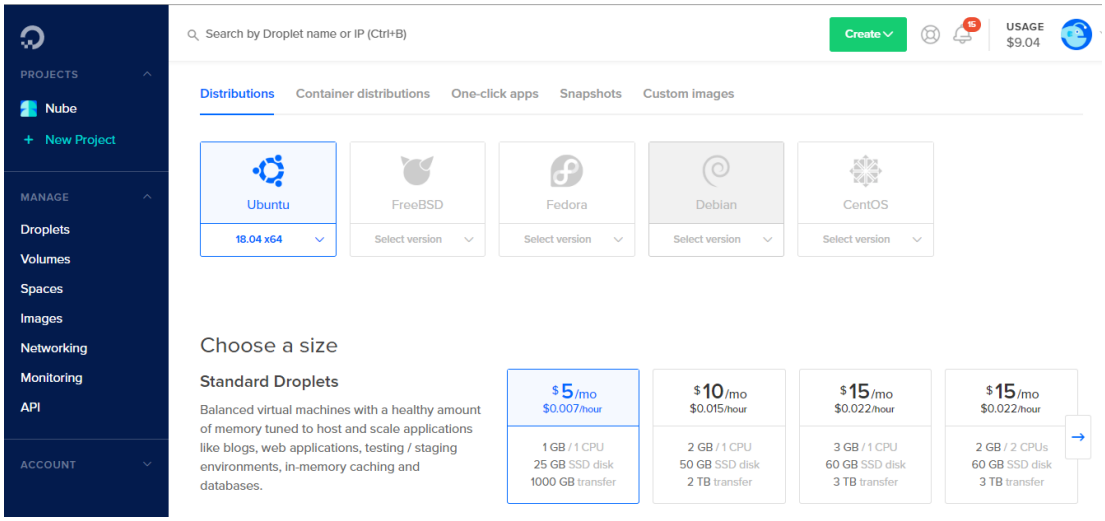


Figura 18. Creación de droplet en DigitalOcean

Se selecciona el sitio del que deseamos el servicio, para el caso de Ecuador lo más cercano es Nueva York – Estados Unidos. Además, se selecciona la versión de IP, en este proyecto se usó la opción de IPv6 (Figura 19).

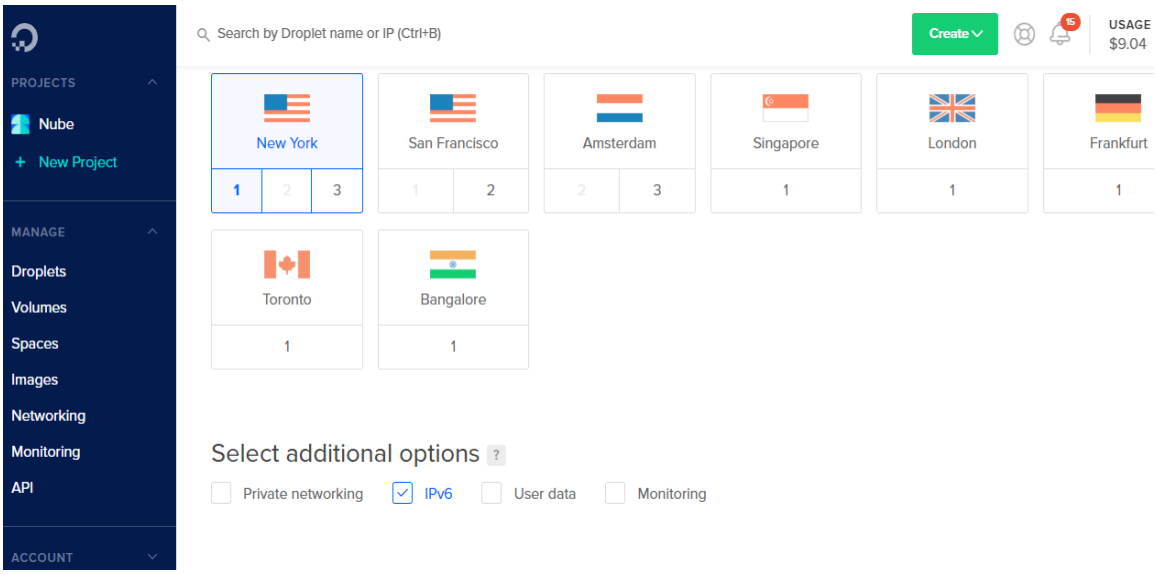


Figura 19. Opciones de configuración de droplet

Adicional a ello, se debe considerar que la conexión remota a dicho droplet se lo debe hacer por medio de una sesión con SSH, para ello se pueden utilizar varias herramientas; para este proyecto

se usó MobaXterm, la misma que permite generar una llave para inicio de sesión, desde el menú herramientas se elige la opción MobaKeyGen Generator (SSH Generator) (Figura 20).

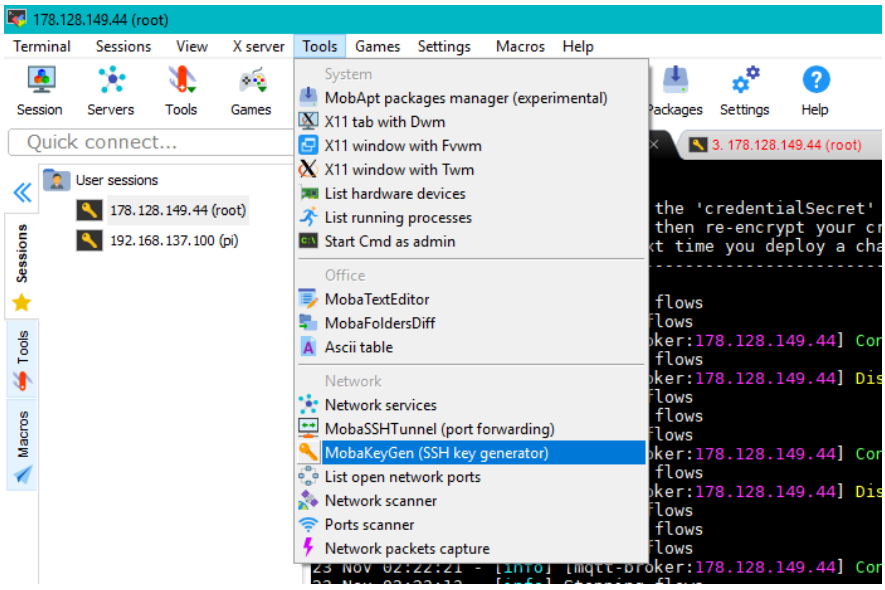


Figura 20. Generación de llave de sesión

Se generará un código en la ventana de creación, la misma que sirve para inicio de sesión en el servidor virtual por SSH (Figura 21). En este caso se llama llave.ppk.

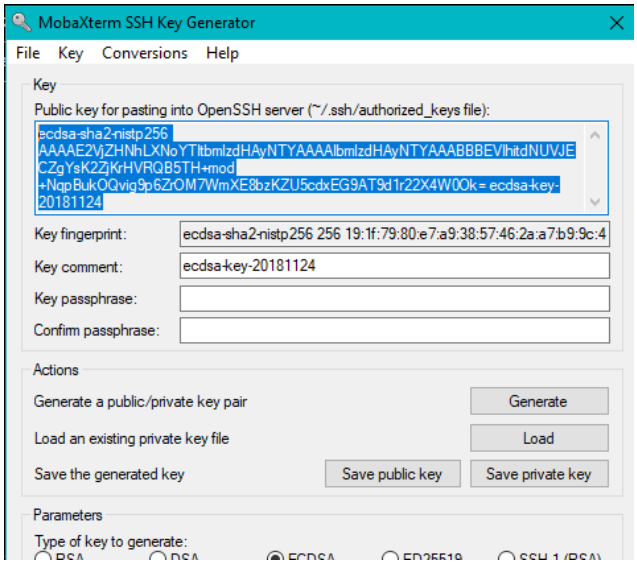


Figura 21. Código de llave SSH

El código de la llave, se copia en las opciones de creación del droplet en DigitalOcean, y se le asigna un nombre (Figura 22).

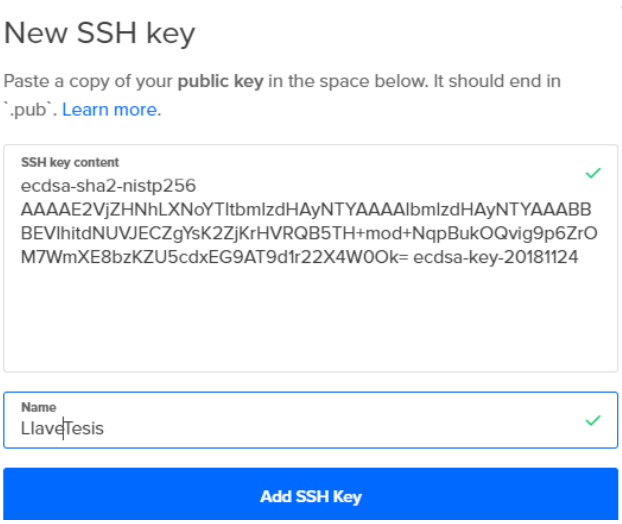


Figura 22. Adición de llave en droplet

Al finalizar la creación del droplet, se mostrará una ventana con su perfil (Figura 23).

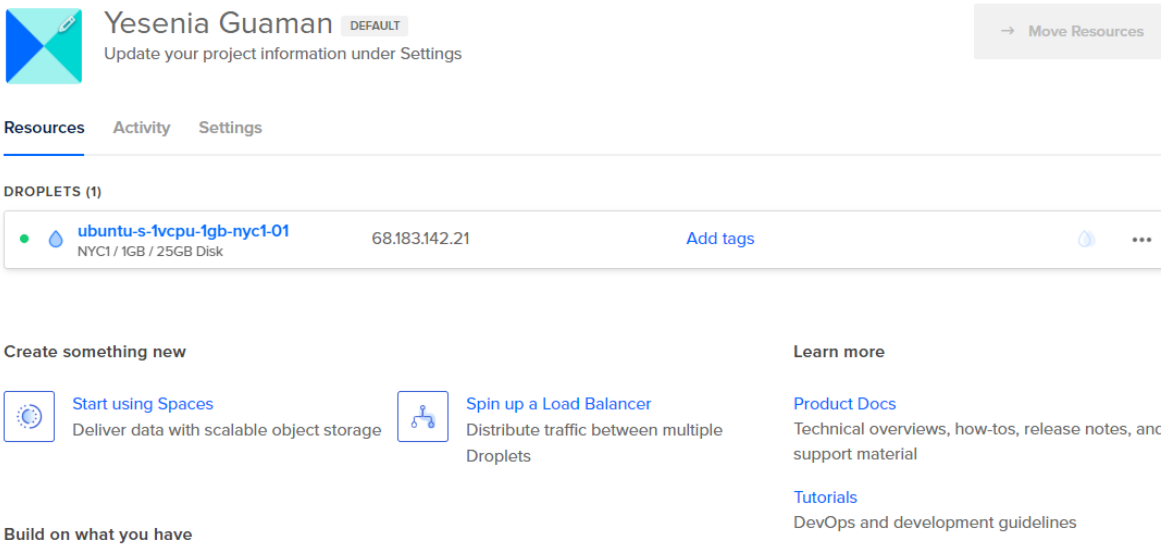


Figura 23. Perfil de droplet

A continuación, se configura la sesión desde MobaXterm, para lo cual se elige el modo SSH puerto 22, la dirección del droplet y el nombre de usuario en este caso root. Posterior a esto, se selecciona el archivo que contiene la llave dentro de las opciones avanzadas de SSH (Figura 24) finalmente ya se puede iniciar la sesión en el droplet (Figura 25).

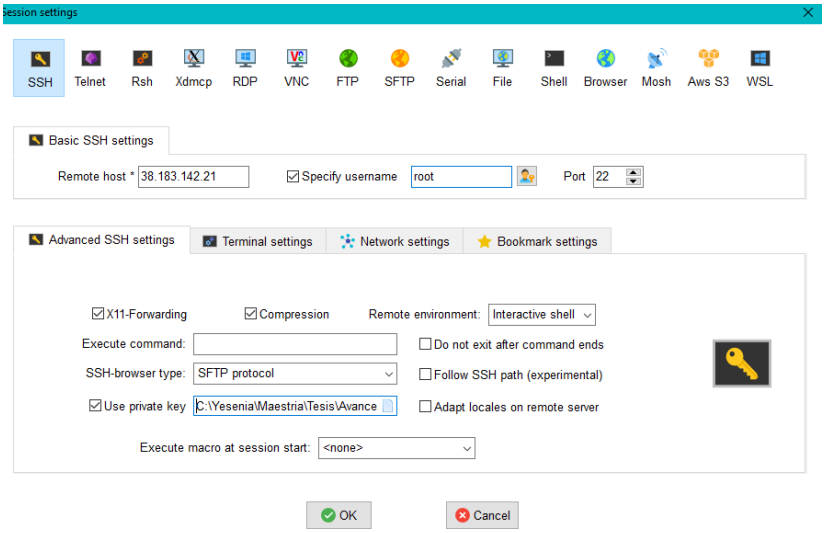


Figura 24. Sesión SSH con droplet

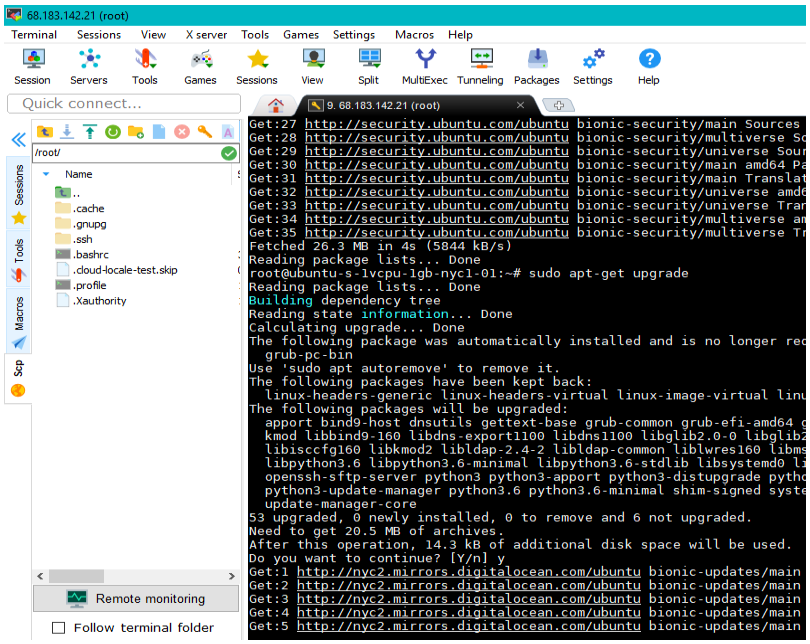


Figura 25. Inicio de sesión en servidor virtual o droplet

Debido a que se trata de la primera sesión en el droplet, es necesario ejecutar los comando Update y Up grade, para su actualización (Figura 26) y (Figura 27).

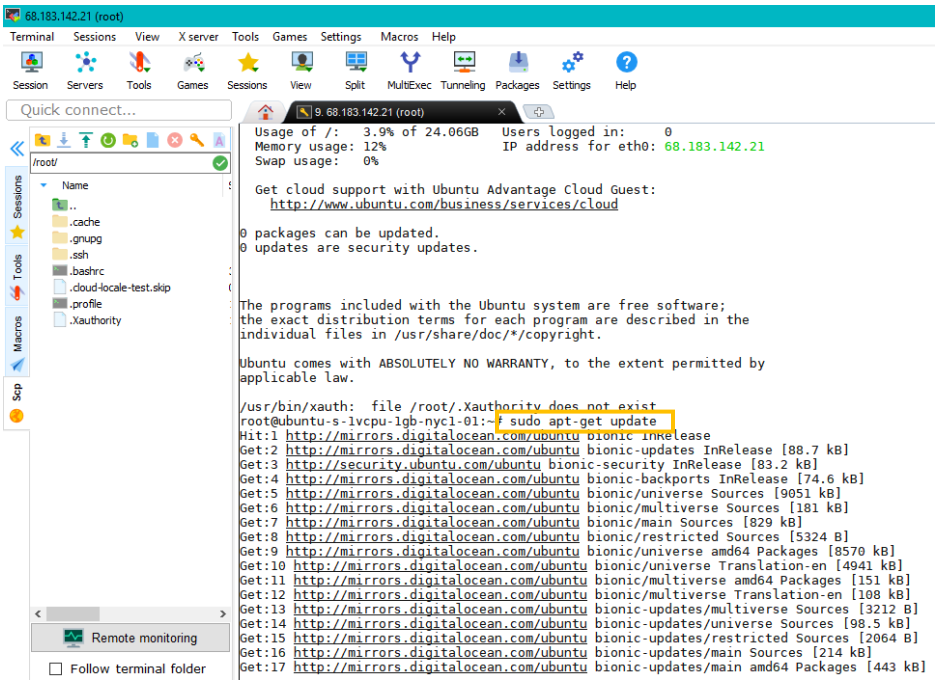


Figura 26. Update en droplet

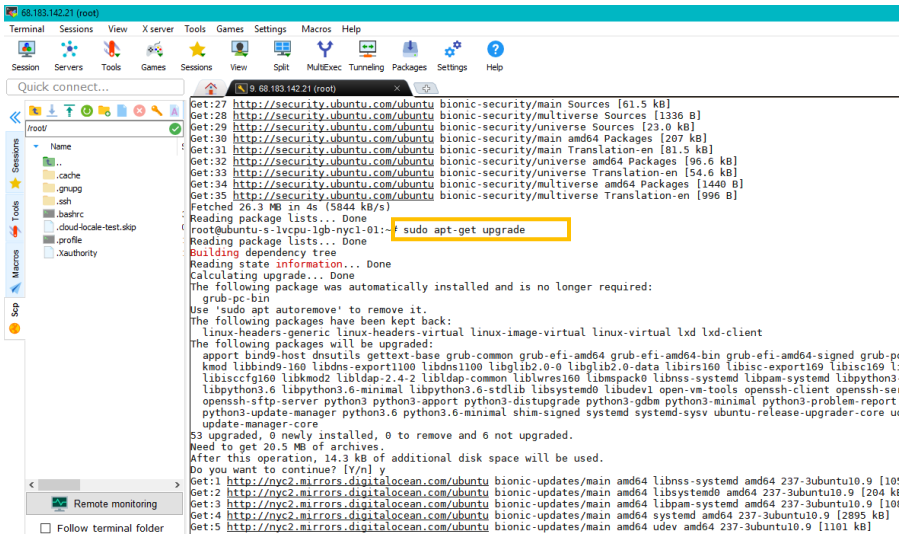


Figura 27. Upgrade en droplet

4.3. Configuración de Raspberry Pi 3 B+

Para la implementación de los protocolos se utilizará Raspberry pi 3 B+, el modelo más reciente el mismo que constituye un ordenador que gracias a su tamaño y versatilidad, se pueden crear proyectos de distintos tipos, en especial en el área que engloba el presente trabajo es decir IoT. (Figura 28), cuyas características se especifican en (**Tabla *5***).

Figura 28. Raspberry pi 3 B+
Fuente. (Raspberry Pi Foundation, s/f)

Tabla 5
Características técnicas de Raspberry pi

ESPECIFICACIONES DE RASPBERRY PI 3 B+

Procesador	Broadcom BXM2837B0, Cortex – A53 64 – bit SoC @ 1.4GHz
Memoria	1GB LPDDR2 SDRAM
Conectividad	2.4 GHz y 5Ghz IEEE 802.11.b/g/n/ac Wireless, LAN, Bluetooth 4.2.
Acceso	40 pines GPIO
Video y sonido	1 x Full size HDMI
Multimedia	H.264, MPEG4 decode (1080p30); H.264 encode (1080p30); Open GL ES 1.1, 2.0 graphics
Soporte de tarjeta SD	Micro SD para cargar el sistema operativo y almacenamiento.
Energía de entrada	5V/2.5V DC con conector micro USB-5V DC via GPIO.
	Energía mediante ethernet (PoE) – habilitado (requiere PoE HAT por separado)
Entorno	Opera a temperaturas de 0 – 50 °C

Fuente: (Raspberry Pi Foundation, s/f)

Como se puede observar en sus especificaciones técnicas, este microordenador se puede usar de varias formas, de acuerdo a las necesidades y creatividad de cada usuario. El sistema operativo que requiere se denomina Raspbian, el mismo que se basa en Linux y se puede descargar de la página oficial de Raspberry Pi Foundation. Éste se instala en una tarjeta micro SD luego de haberle dado un formato. Los softwares que se usaron para darle formato a la tarjeta SD y para grabar el sistema operativo es SD Card Fomatter, balenaEtcher respectivamente.

A partir de la instalación de su sistema operativo, se puede acceder al microordenador con el uso de un teclado, un monitor y mouse; sin embargo, para facilidad de manejo se puede activar el uso de SSH para ingresar por putty, o a su vez por escritorio remoto o VNC (Figura 30), para ello previamente se debe cargar e instalar el paquete conveniente, en este caso se instaló xrdp (Figura 29).

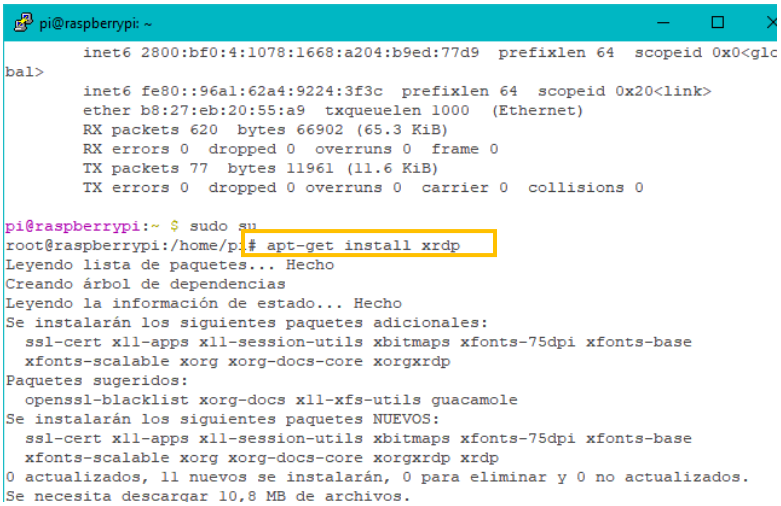


Figura 29. Instalación de xrdp

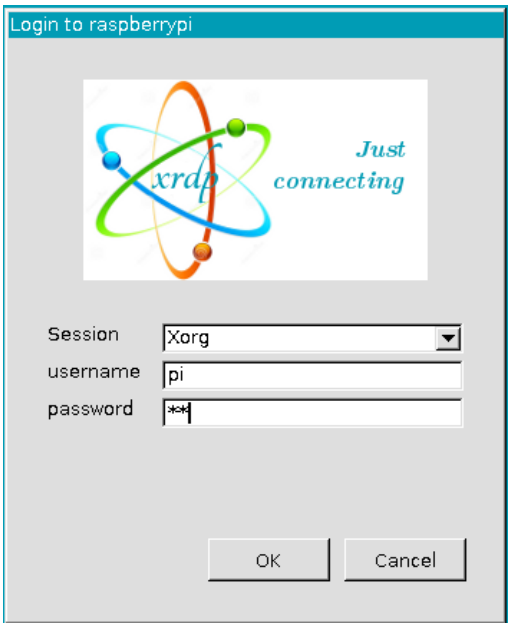


Figura 30. Conexión a Raspberry por escritorio remoto con xrdp

Posterior a ello, se añadió una dirección IP estática en la interfaz eth0, de modo que al conectarse mediante una red LAN entre el computador y el microcomputador o Raspberry, se pueda visualizar y operar por escritorio remoto (Figura 31) y (Figura 32).

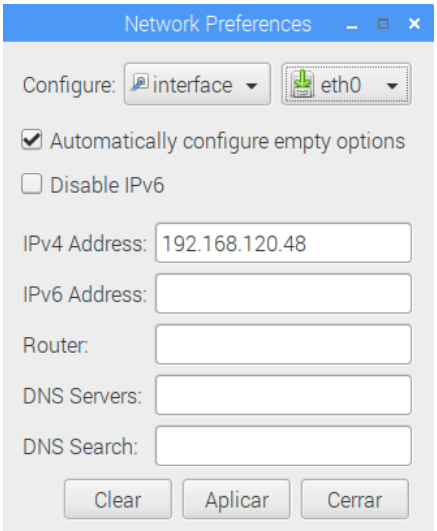


Figura 31. Dirección IP estática en Raspberry

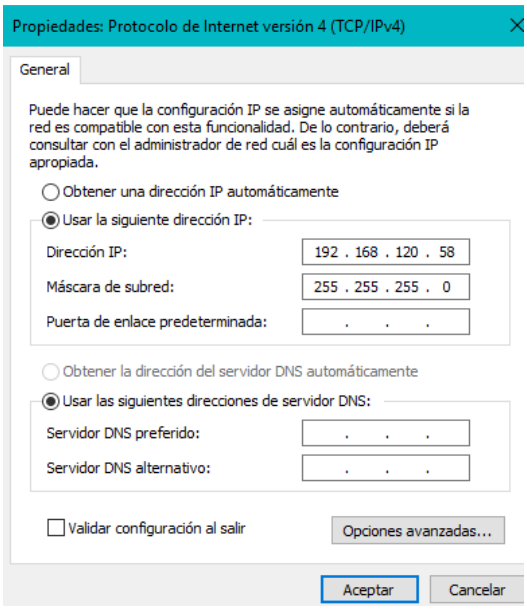


Figura 32. Dirección IP estática en computador

Cuando se enciende por primera vez el microordenador es necesario ejecutar los comandos Upgrade y Update, al igual que en el droplet, con el fin de actualizar todos los paquetes para su correcto funcionamiento (Figura 33).

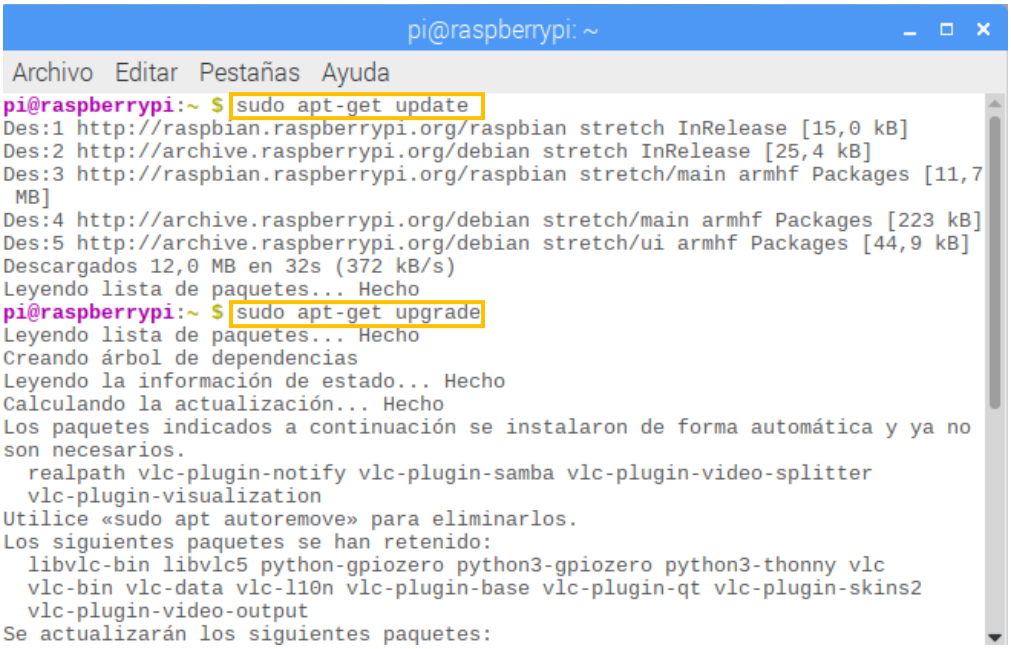


Figura 33. Update y Upgrade en Raspberry

Como se mencionó en el listado de herramientas y materiales, se requiere de un circuito opto acoplador para el microordenador. El circuito se basa en el aislamiento óptico, cuando se va a trabajar con dos voltajes diferentes, en este proyecto se trabaja con 110 Vac, ya que se trata de una lámpara doméstica, y con 3.3 Vdc que es el voltaje que soporta las entradas y salidas de Raspberry. Es decir, se garantiza el aislamiento entre la carga eléctrica y el control del microordenador, salvaguardando el buen uso del mismo. Cabe indicar que circuitos como estos se encuentran fácilmente en el mercado a bajo coste y vienen en módulos de 2, relés en adelante. Para este caso se usa el siguiente módulo (Figura 34).

Figura 34. Módulo opto acoplador

Luego de actualizar los paquetes tanto de Raspberry como del droplet, se realiza las instalaciones respectivas, es decir de Node-RED, Nodejs, Mosquitto y CoAP.

Se usan los siguientes comandos (Tabla 6). Luego de las instalaciones se deben verificar las versiones instaladas, en especial de Nodejs como de Node-RED.

Tabla 6
Comandos de instalación en Raspberry y droplet

Requerimiento	Comando
Mosquitto – MQTT	apt-get install mosquitto apt-get install mosquitto clients
Node – RED	apt-get install nodejs apt-get install npm node-red (ejecución) npm install --g unsafe node-red

Fuente: (Npm, 2018)

4.4. Implementación Protocolo MQTT

4.4.1. Flujo MQTT en droplet o servidor virtual

Para este caso se usa un switch para enviar la señal o mensaje de suscripción al tópico /foco/ hacia el broker, por ello se observa que se conecta con la salida de MQTT; una vez que el broker verifica el tópico /foco/ se publica el mensaje. Finalmente, el mensaje se muestra mediante un nodo debug, tanto de la salida como de la entrada MQTT. En (Figura 35) se observa que los nodos MQTT tienen en la parte inferior un estado “connected”, lo que obedece al inicio del flujo en Node-RED con los nodos MQTT, junto con Raspberry y droplet, en los que previamente se instaló mosquitto; es decir, al ejecutar el nodo se inicia el servicio.

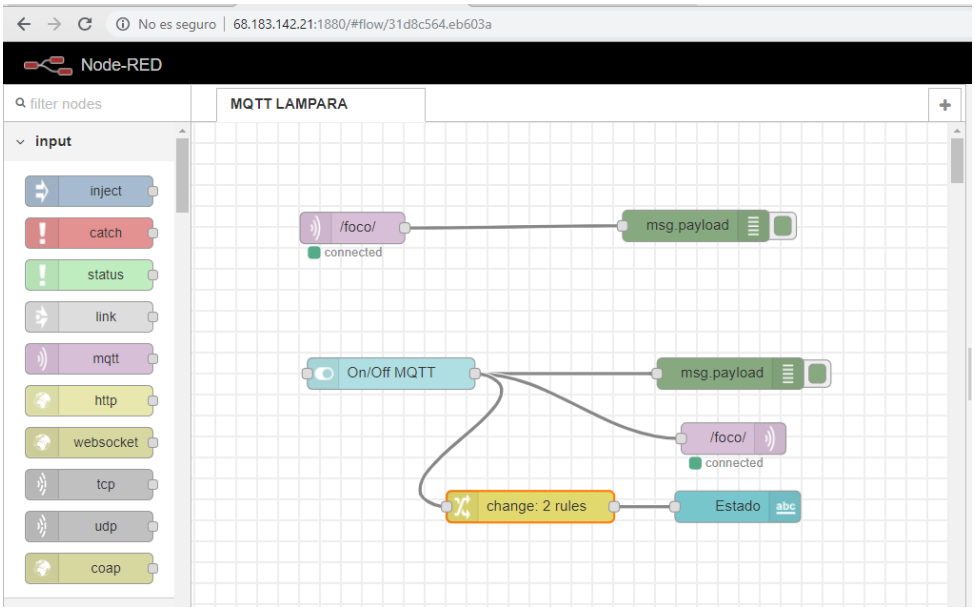


Figura 35. Flujo para protocolo MQTT en droplet

El flujo para este protocolo se compone de los siguientes nodos:

MQTT in: Permite la conexión entre el broker MQTT y suscribe los mensajes según el tópico que se indique. En este proyecto el tópico se denomina /foco/. Dentro de este nodo se especifica la dirección del servidor que es el broker de MQTT y el nivel de QoS (Figura 36).

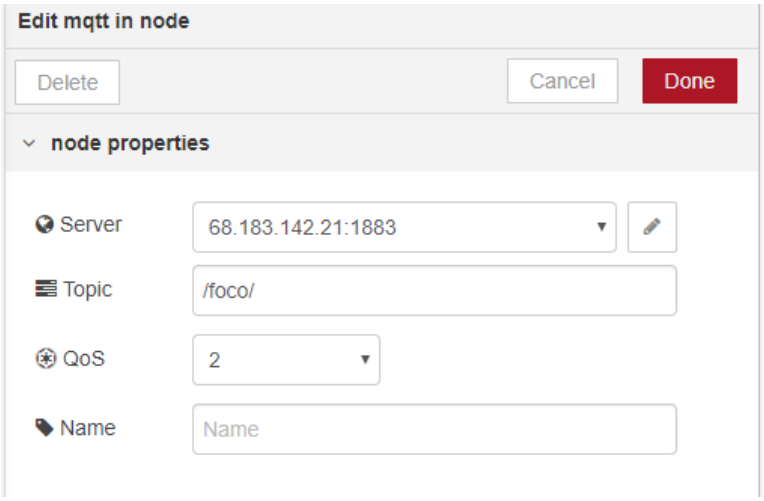


Figura 36. Nodo MQTT_in en droplet

MQTT out: Este nodo se conecta al broker MQTT y se encarga de publicar los mensajes. Al igual que en el nodo de entrada, se especifica la dirección de servidor en la Nube y el nivel de QoS (Figura 37).

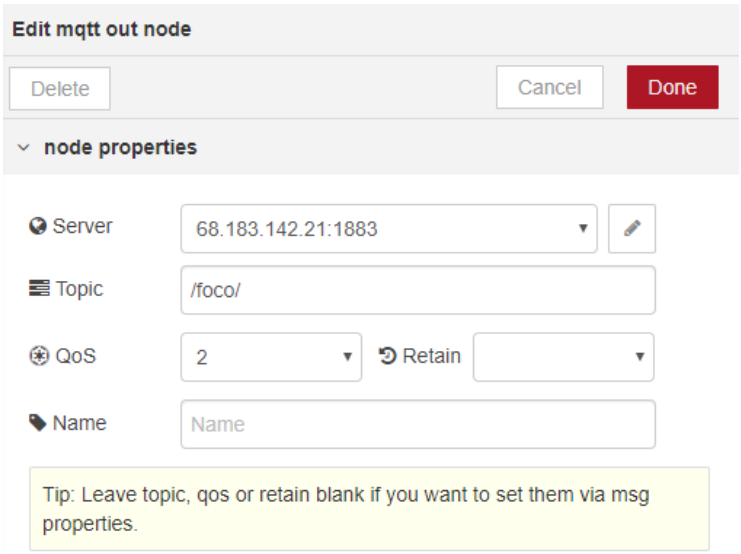


Figura 37. Nodo MQTT_out en droplet

Debug: Es un nodo que se encarga de mostrar en pantalla el valor del mensaje, ya sea este un dato numérico, booleano o cadena de caracteres (Figura 38).

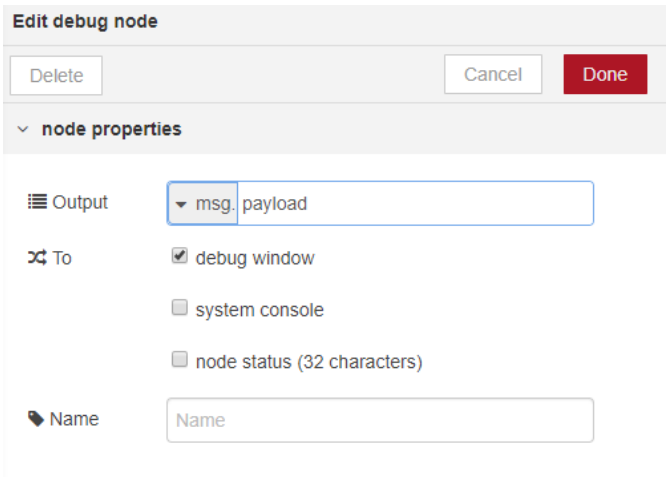


Figura 38. Nodo debug en droplet

En primera instancia, este nodo no se encuentra disponible en Node-RED, por lo que se requiere buscar dentro de *Manage Pallette* el grupo de nodos *node-red-Dashboard* e instalarlo (Figura 39).

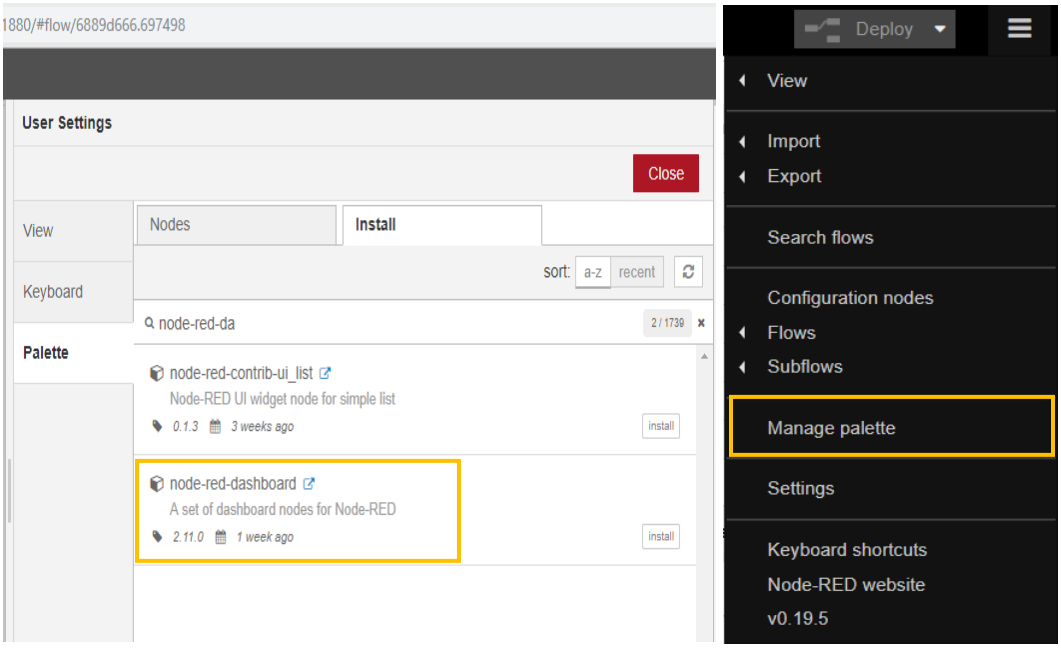


Figura 39. Instalación de Node-Red-dashboard

Change: Este nodo sirve para configurar, cambiar, eliminar o mover las propiedades de un mensaje, flujo o contexto. Dentro de este nodo se pueden especificar una serie de reglas que se verificarán en el orden en que se definan. Se utilizó este nodo para indicar si el foco se encuentra apagado o encendido, ya que el dato inicial es booleano y se desea mostrar una cadena de caracteres en la interfaz gráfica. De modo que es necesario realizar una conversión (Figura 40).

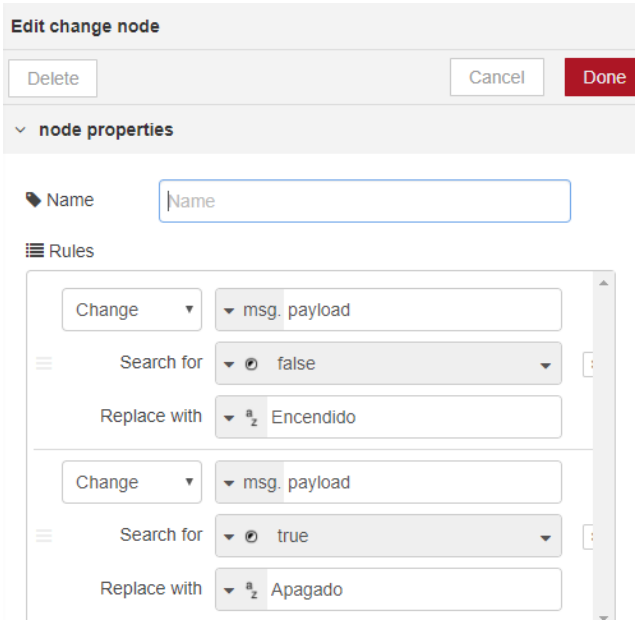


Figura 40. Nodo Change en droplet

Switch: Nodo para aplicación de on / off en la lámpara doméstica (Figura 41). Se usa datos booleanos, es decir true o false (verdadero o falso).

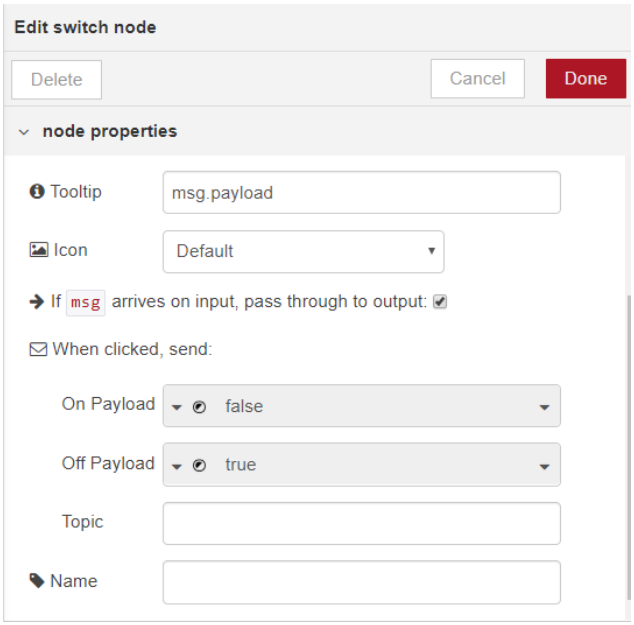


Figura 41. Nodo switch en droplet

UI_text: Nodo que muestra una cadena de caracteres, en este caso de acuerdo a la posición del switch, mostrará en la interfaz gráfica si el foco de la lámpara está encendido o apagado (Figura 42).

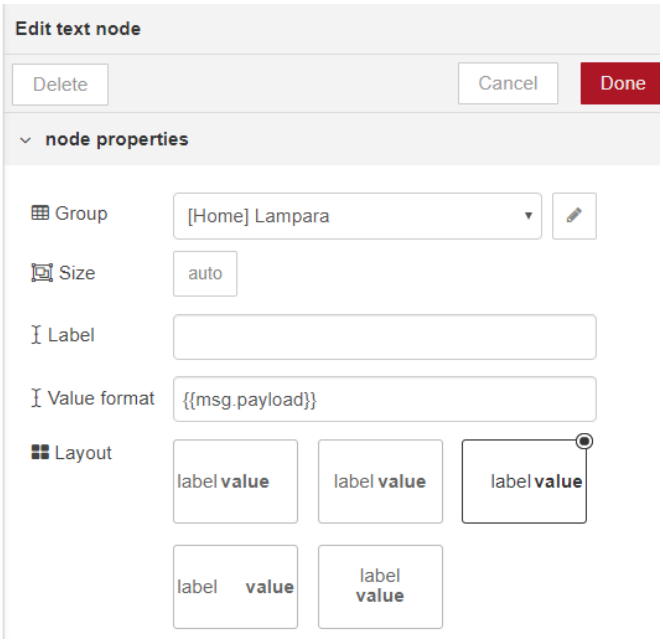


Figura 42. Nodo UI_text en droplet

4.4.2. Flujo en raspberry o localhost

Después del flujo en el droplet, se continua con el flujo en Raspberry; el mismo que se denomina como local o localhost, dentro del browser se digita http://localhost:1883 (Figura 43).

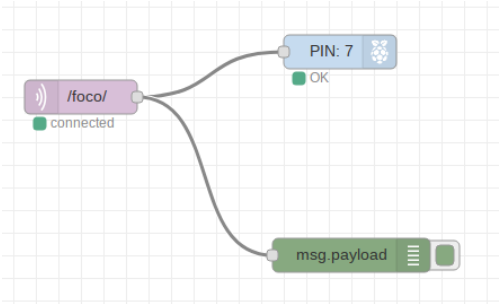


Figura 43. Flujo para protocolo MQTT en Raspberry o localhost

El flujo contiene los siguientes nodos:

MQTT in: Se especifica el tópico, nivel de QoS y la dirección del servidor o broker (Figura 44)

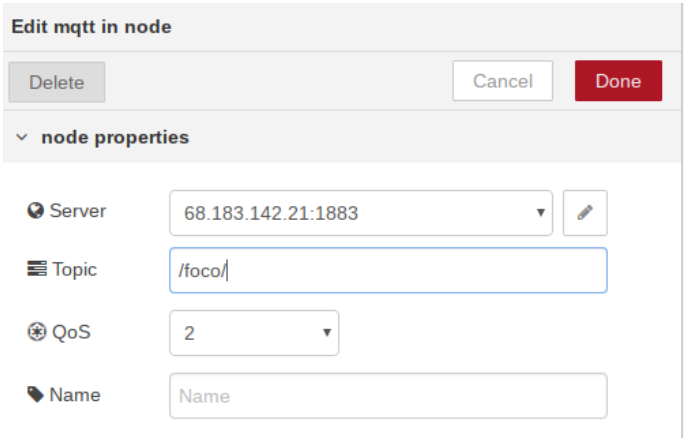


Figura 44. Nodo MQTT in en Raspberry

RPI-GPIO-OUT: Especifica el PIN al que se ingresará el valor o el contenido del mensaje y así publicarlo (Figura 45).

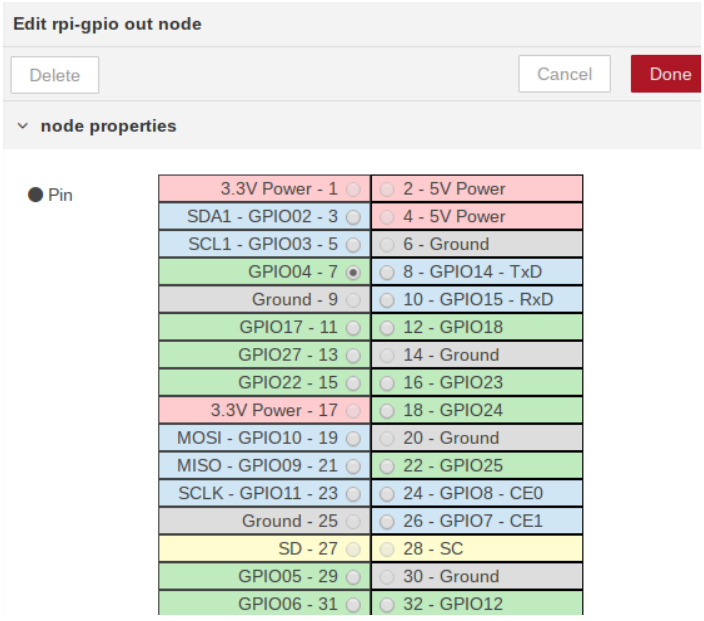


Figura 45. Nodo RPI-GPIO-OUT en Raspberry

Debug: Se mostrará en pantalla el mensaje que se publicó (Figura 46).

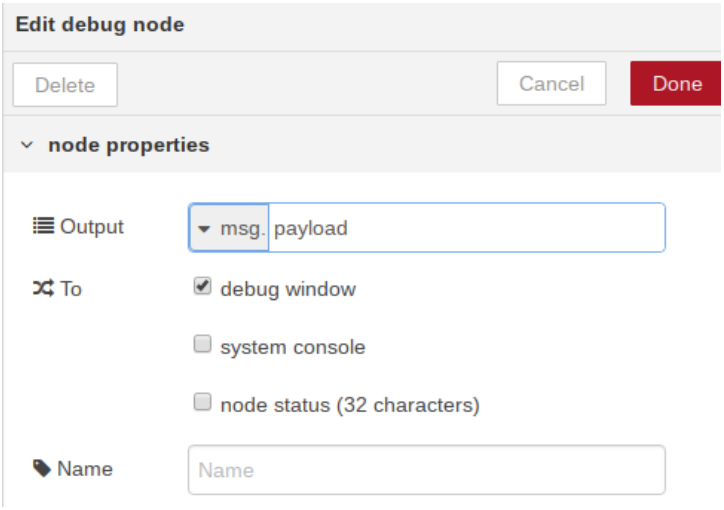


Figura 46. Nodo debug en Raspberry

Ahora bien, se puede verificar el proceso y funcionamiento del protocolo MQTT desde la interfaz gráfica de Node-RED, que se obtiene al digitar en la barra de un browser la dirección del droplet o broker en este caso, seguido de las letras minúsculas *ui*, de la siguiente manera: http://68.183.142.21:1880/ui (Figura 47).

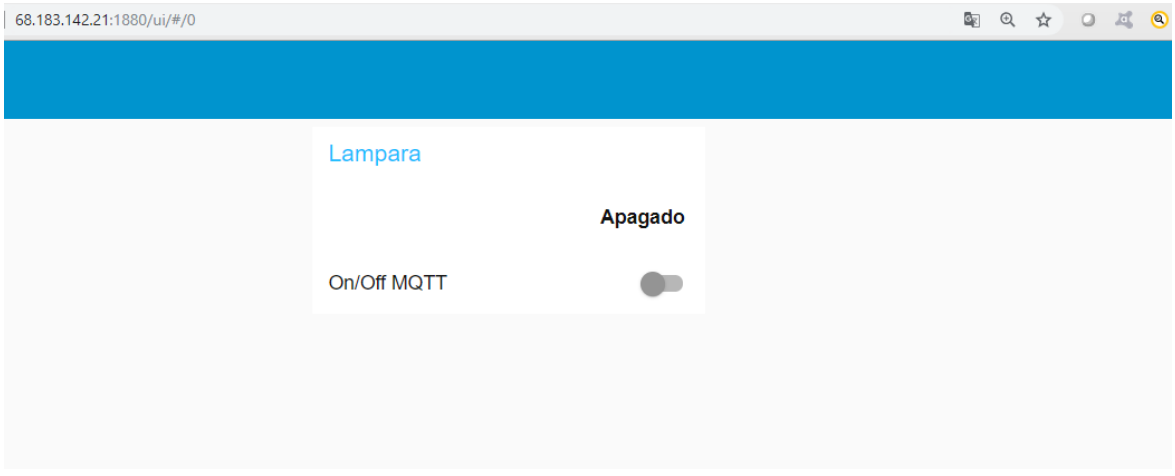


***Figura* 47.** Interfaz gráfica Node-RED en droplet

A continuación de muestra un diagrama de la conexión física de Raspberry, lámpara y circuito opto acoplador (Figura 48).

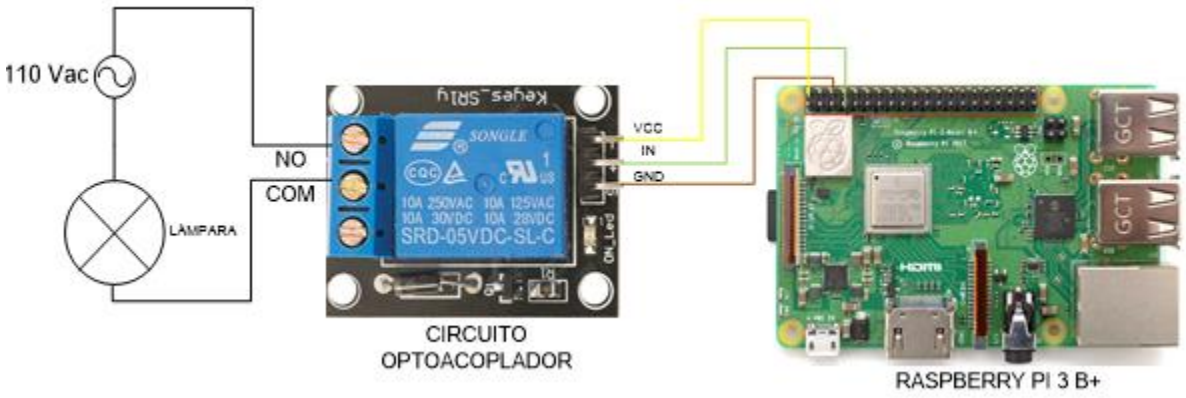


Figura 48. Conexión física de lámpara y Raspberry

Para la verificación del funcionamiento tanto del flujo, como se la conexión se ejecuta el proceso de la siguiente manera:

- Inicio de sesión en Raspberry, para este caso por escritorio remoto
- Inicio de sesión en droplet.
- Ejecución de Node-RED en Raspberry y droplet.
- Interfaz gráfica de droplet. Al encender la lámpara (Figura 49) se debe reflejar el mensaje en el debug de Raspberry o localhost (Figura 50), y se encenderá la lámpara (Figura 51).

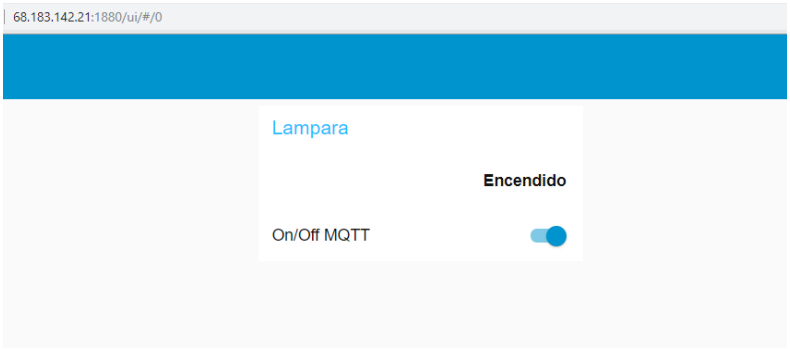


Figura 49. Encendido de lámpara por MQTT

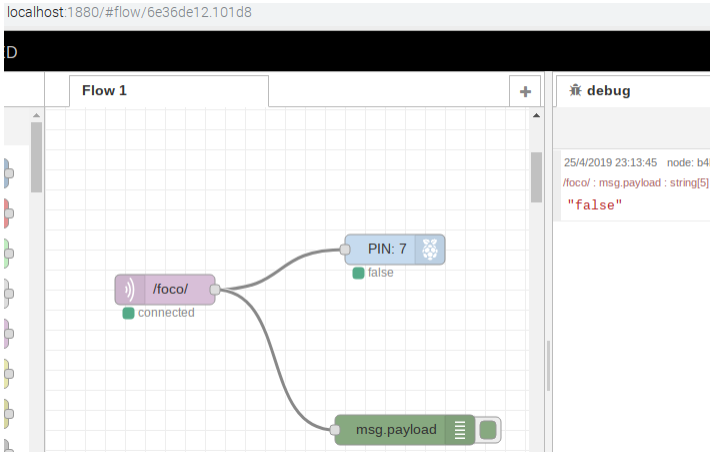


Figura 50. Debug en Raspberry (encendido)

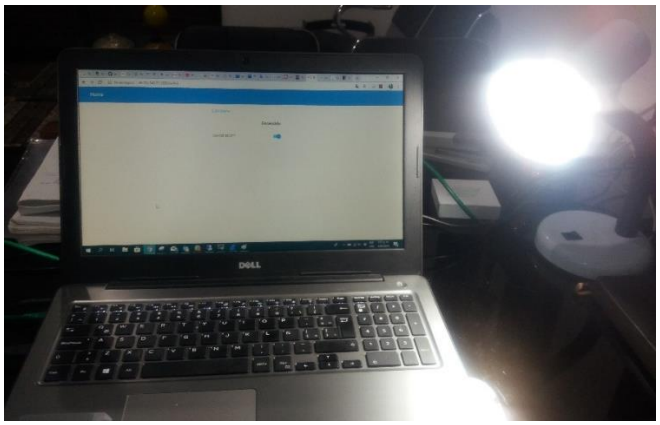


Figura 51. Lámpara encendida por MQTT

- Se realiza el mismo procedimiento desde la interfaz gráfica para apagar la lámpara (Figura 52). El debug de Raspberry cambiará el mensaje (Figura 53), y en consecuencia la lámpara se apagará (Figura 54).

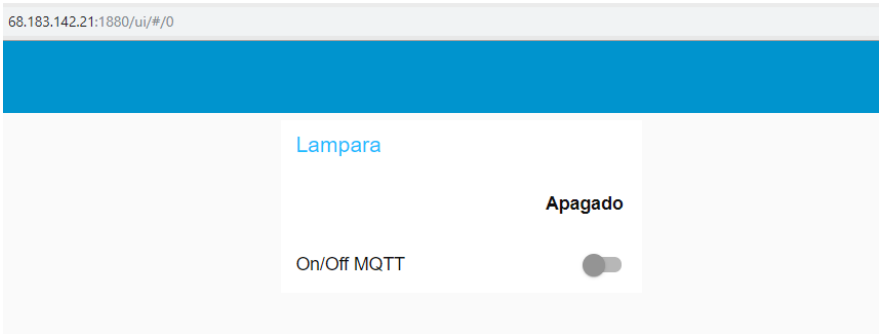


Figura 52. Apagado de lámpara por MQTT

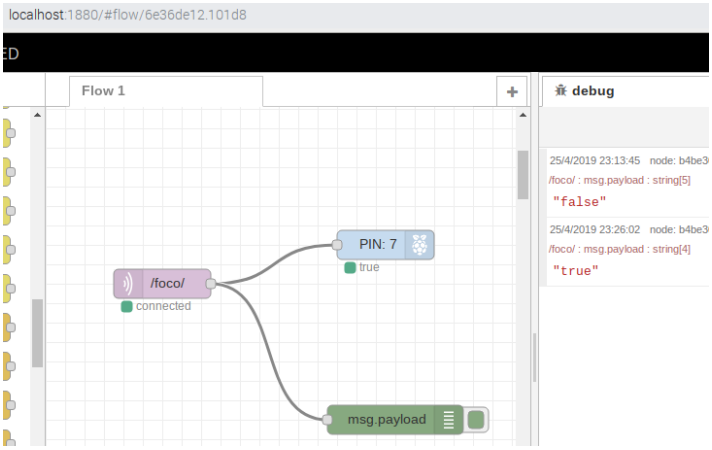


Figura 53. Debug en Raspberry (apagado)

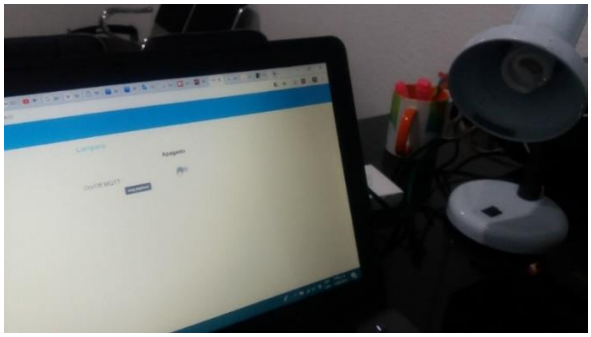


Figura 54. Lámpara apagada por MQTT

4.5. Implementación de CoAP

Para lo que corresponde a la implementación del protocolo CoAP, como prerrequisito, se necesita instalar las librerías necesarias.

4.5.1. Instalación de librerías CoAP en Node - RED

- CoAP-CLI, con el uso del comando: npm install coap-cli –g (NPM, s/f)
- Node-red-contrib-coap con el uso del comando: npm install node-red-contrib-coap (Node Red, s/f) (Figura 55)

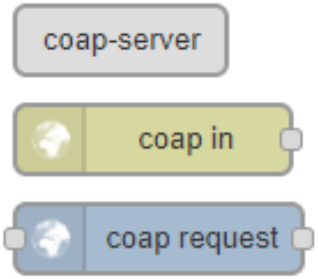


Figura 55. Nodos CoAP en Node Red

Cabe indicar que, si se encuentra otro servicio en ejecución en este caso MQTT, se debe inhabilitar el flujo dentro de Node-RED, o su vez, para aquel servicio dese un terminal.

4.5.2. Flujo CoAP para Node – RED en Raspberry

Debido a las funcionalidades y prestaciones que CoAP ofrece, su implementación se la realizó a nivel local, es decir desde el microordenador Raspberry; ya que CoAP al ser un protocolo REST, se basa en métodos y es por ésta estructura que NAT se limita. En consecuencia, si enviamos desde un droplet como en el caso de MQTT, este no tendría resultado, para solucionar este inconveniente se puede levantar un tunneling para el reenvío de puertos.

Todo el flujo y procedimiento que se describe corresponde al equipo local (Figura 56), el mismo que se compone tanto de un nodo coap in, como de un nodo coap request, nodo para función, Change y GPIO.

Su funcionamiento parte del control On/Off, el mismo que se genera la solicitud de suscripción al servidor; el servidor recibe la orden la procesa y permite o deniega la solicitud. Al aceptar la solicitud, la respuesta regresa al suscriptor, una vez que se recibe la respuesta se identifica si la orden es encender o apagar el foco, para esto se usa el nodo Change.

Además, se cuenta dos debug, uno de ellos muestra el objeto que se recibe y el otro indica el mensaje publicado.

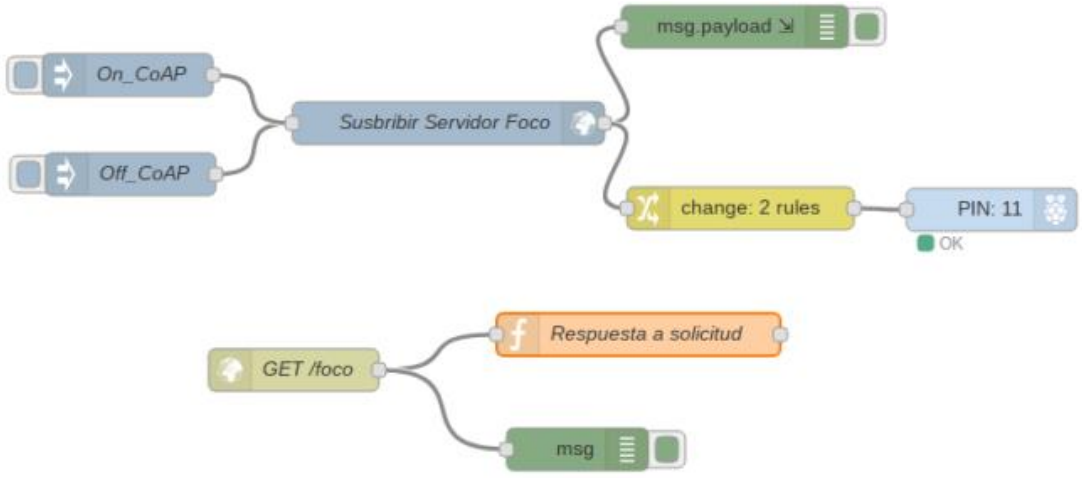


Figura 56. Flujo CoAP on-off en Node - Red

A continuación, se presenta las configuraciones que se realizaron en cada uno de los nodos.

Inject: Para el control On/Off, se usaron nodos de tipo Inject; los mismos que tienen como tipo de dato una cadena de String, que indica “Encendido” y “Apagado”, respectivamente. (Figura 57) y (Figura 58)

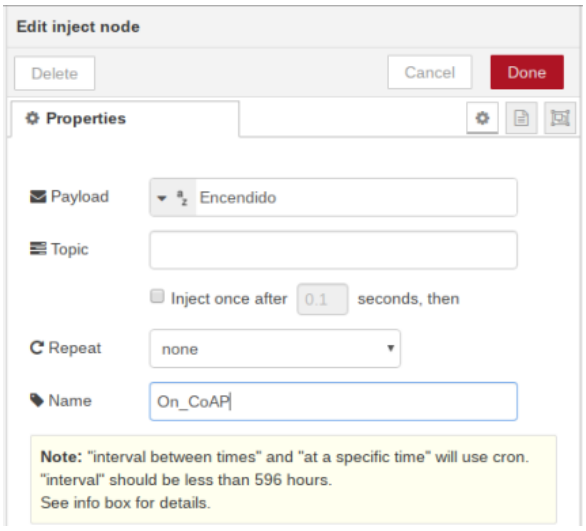


Figura 57. Nodo Inject On_CoAP

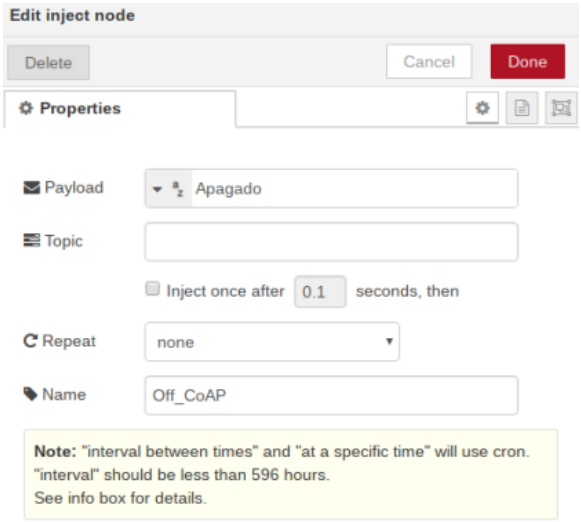


Figura 58. Nodo Inject Off_CoAP

Nodo coap request: En este nodo, se especifica la URL que se usará, para este caso coap://localhost/foco, puerto 5683 como se puede observar el tópico corresponde a foco, se usó el método GET en modo texto. (Figura 59). Este nodo tiene la función de solicitar la suscripción al servidor coap.

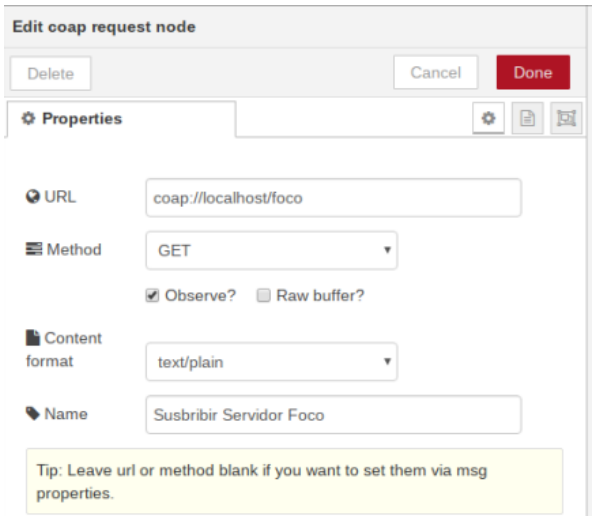


Figura 59. Nodo coap request

Nodo coap in: Corresponde al servidor con el que se va a comunicar, en el que se especifica nuevamente el método y la URL para este caso /foco. (Figura 60)

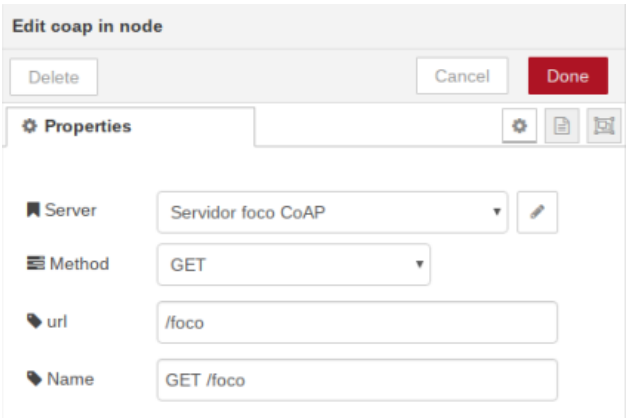


Figura 60. Nodo coap in

Nodo Change: Debido a que la inyección del dato se lo realiza de tipo String es decir una cadena de caracteres, re requiere que dicho dato se transforme en un pulso hacia uno de los GIPOs para el control del foco. Por ello, este nodo cumple con identificar la palabra “Encendido” o “Apagado” y transformarla en un 0 o 1. (Figura 61)

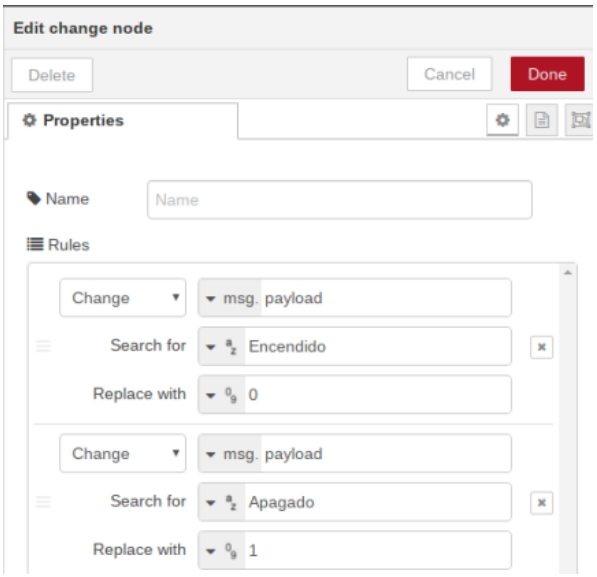


Figura 61. Nodo Change

Nodo Function: Dentro del nodo Function, se especifica el mensaje que retornará al debug que corresponde al suscriptor. Se aplicó el método msg.res.end en el objeto que se recibió como solicitud, lo que corresponde a msg.req.payload. Finalmente retorna el mensaje, el mismo que se visualizará en el debug correspondiente. (Figura 62)

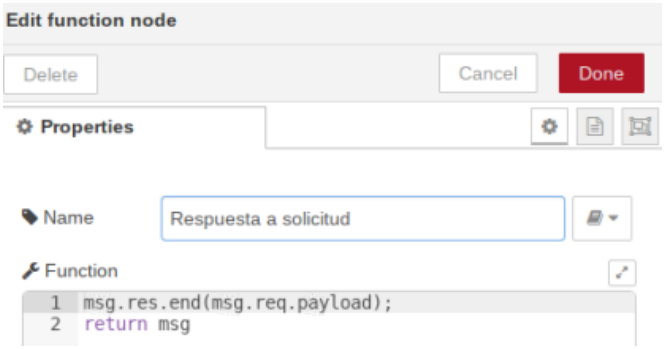


Figura 62. Nodo Function para respuesta de solicitud

Nodo rpi-gpio out: Se configuró el pin que recibe el pulso de control, para este caso se eligió el pin 11 o GPIO7. (Figura 63)

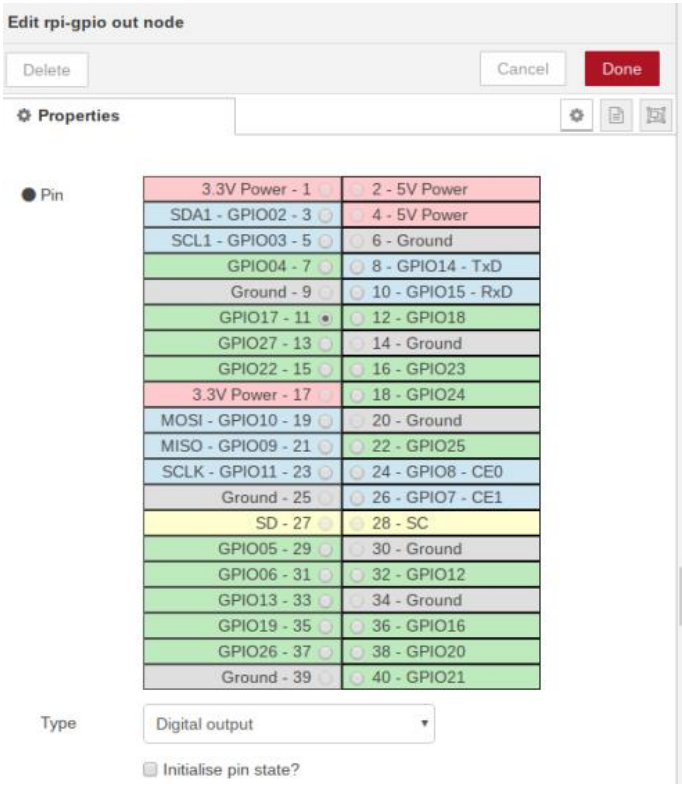


Figura 63. Nodo Rpi gpio out

Dentro del terminal en que se inició Node-RED, se observaron los mensajes publicados Encendido y Apagado. (Figura 64)

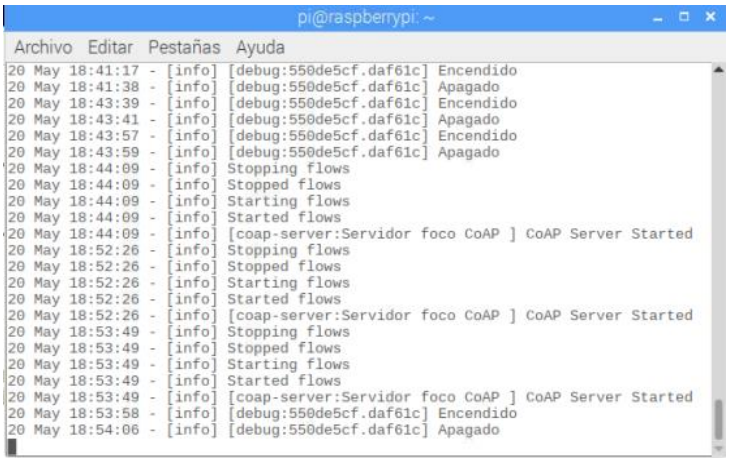


Figura 64. Terminal en Raspberry (localhost)

4.6. Otras formas de implementación de CoAP

En la actualidad, existen diversas formas para la implementación de este tipo de protocolos, sin embargo; es importante mencionar que, existen librerías que se no han continuado con su desarrollo. Dentro de las más destacadas se tiene:

- Node-RED, mediante flujos y correcta instalación de librerías. ("Open IOT Challenge, AttA plays well with Node-RED", 2016)
- Nodejs, con el uso de JavaScript.
- Copper (Cu), plugin de Mozilla. Se puede también añadir desde Google Chrome. ("CoAP Tutorial for Raspberry Pi", 2017), ("GitHub - mkovatsc_Copper4Cr_ Copper (Cu) CoAP user-agent for Chrome (JavaScript implementation)", 2018), (Kovatsch & Vermillard, s/f)
- Californium (Cf), librería de Java. (Kovatsch & Vermillard, s/f)
- Python, mediante la librería TxThings. (Github, 2014), (Garrido, 2016)
- Lenguaje C y C#, con Coap Net. (Github, 2014)

CAPÍTULO V

ANÁLISIS DE RESULTADOS

Como se mencionó en el capítulo anterior, el entorno en el que se basa el prototipo de comparación de los protocolos, constituye una actividad cotidiana dentro del hogar como el encendido y apagado de un foco o lámpara. Luego de cada implementación la comparación se la realizó mediante la captura de paquetes en wireshark, e inyección de tráfico en JPERF.

5.1. Resultados de protocolo MQTT

Para el análisis de los resultados se utilizaron dos herramientas tanto para la revisión de paquetes como de su ancho de banda y jitter. Por ello, se adjuntan los resultados que se obtuvieron con Wireshark que ayuda a distinguir los mensajes en cada protocolo y Jperf que permite la inyección de tráfico. Inicialmente, se verifica que el equipo en que se realizará la captura cuente con el software requerido. En este trabajo las dos herramientas se instalaron dentro del microordenador Raspberry, ya que en este dispositivo se reciben los mensajes.

Para instalar wireshark desde un terminal se digita los siguientes comandos (Kumar, 2017):

- sudo apt-get install wireshark –y
- sudo chmod +x /usr/bin/ dumpcap

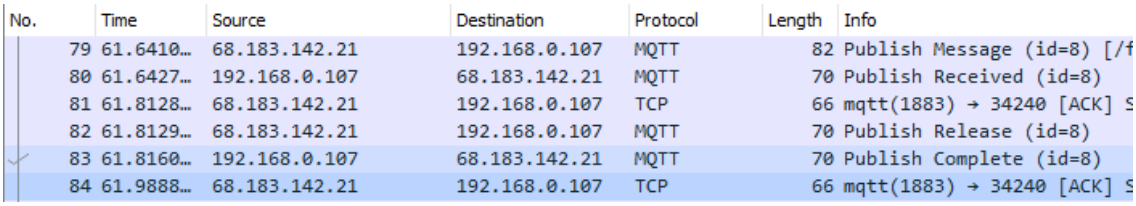
Para instalar Jperf dentro de un sistema operativo distribución de Linux, se siguen los siguientes pasos (Vouzis, 2018):

- Descargar la versión de Jperf requerida de la página SourceForge.Net.

- Descomprimir el archivo.
- Ingresar a la carpeta de Jperf.
- Se otorgan permisos con el comando: sudo chmod u+x jperf.sh
- Se ejecuta el archivo .sh: sudo ./jperf.sh

5.1.1. Resultados Wireshark

Con el uso de Wireshark, se capturaron los paquetes cuya secuencia se indica en (Figura 65). en la que se evidencia el establecimiento de conexión, mensaje de publicación, publicación receptada, envió de acuse de recibo, publicación en ejecución y completa y finalmente el acuse de recibo de dicha publicación.



No.	Time	Source	Destination	Protocol	Length	Info
79	61.6410…	68.183.142.21	192.168.0.107	MQTT	82	Publish Message (id=8) [/f
80	61.6427…	192.168.0.107	68.183.142.21	MQTT	70	Publish Received (id=8)
81	61.8128…	68.183.142.21	192.168.0.107	TCP	66	mqtt(1883) → 34240 [ACK] S
82	61.8129…	68.183.142.21	192.168.0.107	MQTT	70	Publish Release (id=8)
83	61.8160…	192.168.0.107	68.183.142.21	MQTT	70	Publish Complete (id=8)
84	61.9888…	68.183.142.21	192.168.0.107	TCP	66	mqtt(1883) → 34240 [ACK] S

Figura 65. Secuencia de paquetes MQTT

En el primer paquete, se observa el ping request (1100) que se ejecuta para establecer su conexión. (Figura 66) por lo que se realiza el ping pertinente para comprobar la existencia del bróker o intermediario; como se puede observar su tamaño es reducido y su duración corta.

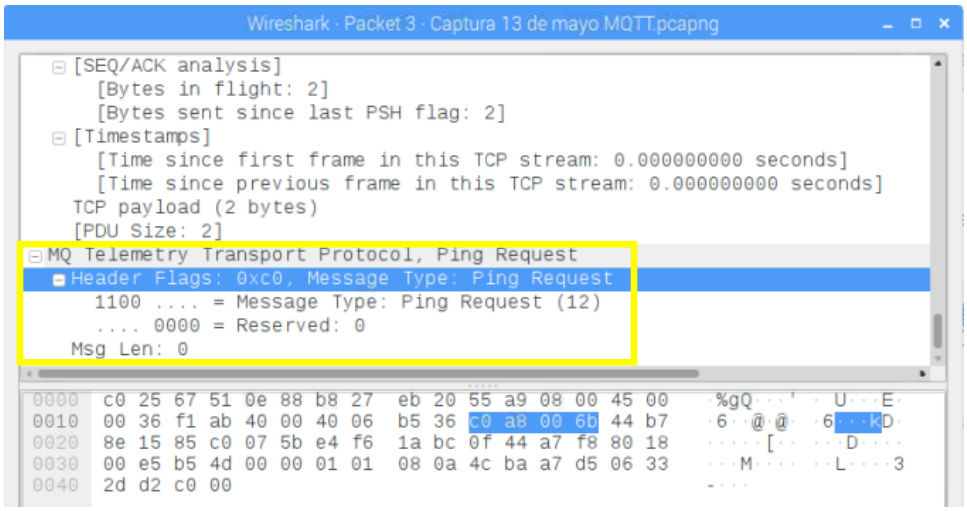


Figura 66. Ping request de MQTT

La longitud de un paquete ping request es de 68 bytes es decir de 544 bits, la interfaz en la que se refleja la captura de paquetes de MQTT es en la wlan0 (Figura 67). Puesto que, requiere comunicación entre el dispositivo final, el bróker y la Raspberry; es decir el usuario que envíe la orden de control On/Off debe contar con salida a Internet, de modo que, se comunique con el droplet el mismo que posee una IP pública, y que a partir de ella se puede conectar desde cualquier dispositivo.

Dentro del mismo paquete se puede observar el tamaño en bits de dicho ping, y la interfaz que se encuentra en uso, para este caso su tamaño es de 544 bits o su vez de 68 bytes lo cual es relativamente corto; mientras que la interfaz que se usa es la Wlan 0, ya que debe establece comunicación con el droplet de Digital Ocean, que está actuando como bróker. (Figura 67)

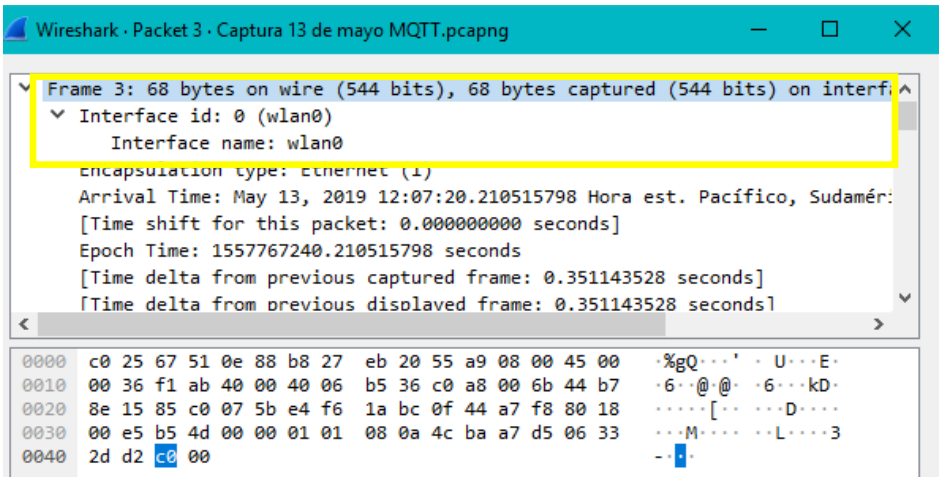


Figura 67. Longitud de ping request

El paquete de verificación de conexión, comprende un mensaje de tipo ping response, es decir la respuesta a la solicitud del ping request (1101) (Figura 68). Es decir, al ping de solicitud se le responde con otro ping desde el bróker, para verificar la existencia y procedencia del equipo que emitió la solicitud. Posterior a ello, se entabla la conexión. Es importante mencionar que, en este tipo de paquetes tanto los tamaños como PDUs, y tiempos de envío son bajos, ya que solo se trata de comprobación de conexión, mas no de envío de datos.

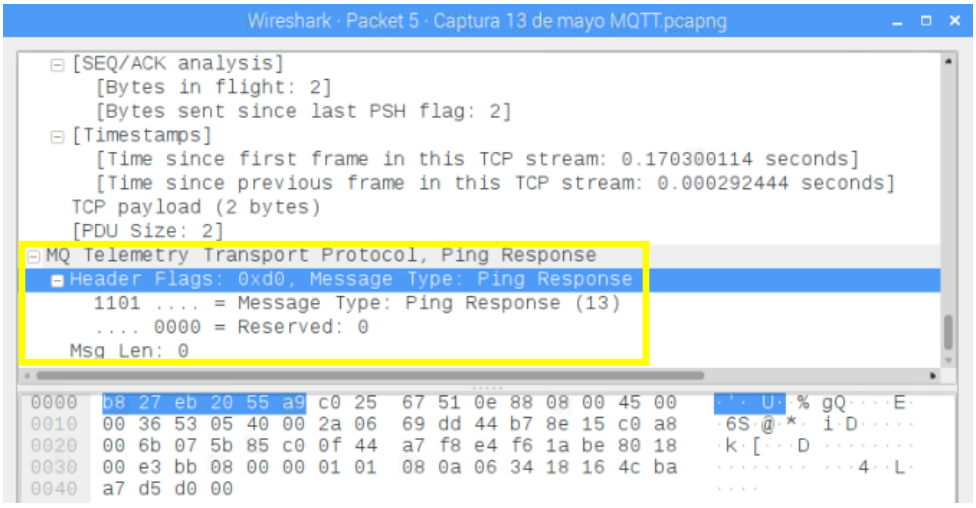


Figura 68. Ping response de MQTT

En (Figura 69) se muestra el tamaño e interfaz de uso, al igual que el ping de solicitud, se evidencia que tanto el ping request como el de response poseen el mismo tamaño de 544 bits y usan la misma interfaz Wlan 0. Por ello, su transmisión dura alrededor de milésimas de segundos.

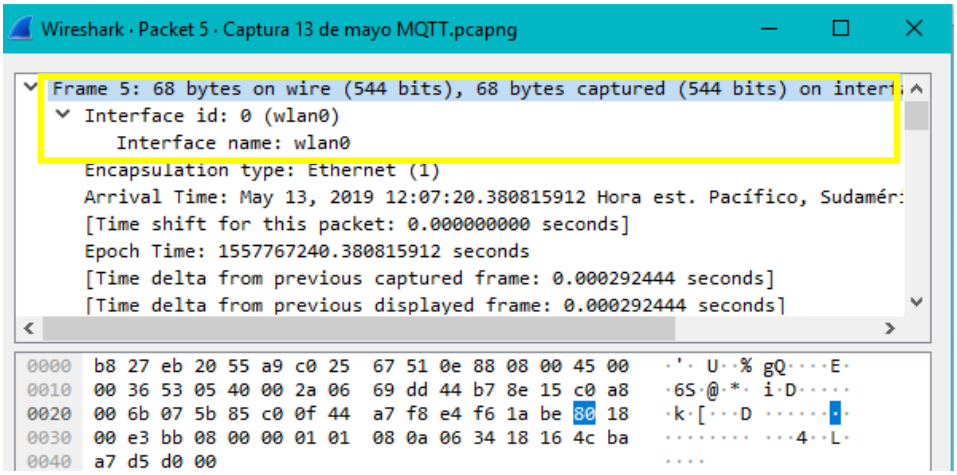


Figura 69. Longitud de ping response

Luego de verificar la conexión entre los equipos involucrados, se inicia con la aplicación del flujo en Node-Red. Para ello, se inicia al conectar a Internet cualquier dispositivo ya sea móvil o fijo o a su vez al abrir un browser, se digita la dirección pública del droplet en modo interfaz de usuario, para este caso: htpp://68.183.142.21:1883/ui. Al interactuar con el switch de control, se capturaron los siguientes paquetes.

Inicialmente, la solicitud de publicación (Figura 70). Como se mencionó en el capítulo anterior el nivel de QoS es de nivel 2, se denota que el mensaje corresponde a un dato booleano (false), que es el que indica encender el foco; el tópico al que desea suscribirse es /foco/. Todo aquello se observa dentro de este paquete.

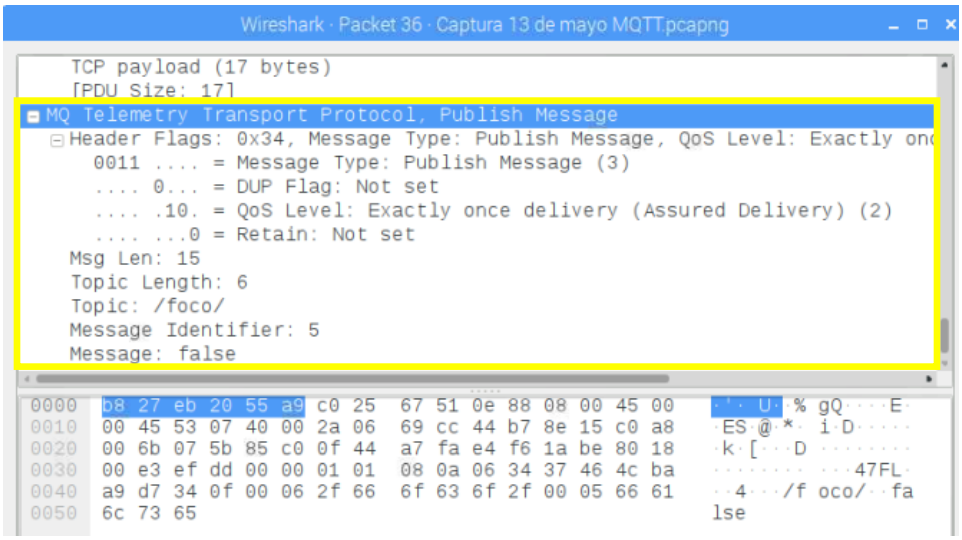


Figura 70. Solicitud de publicación de mensaje

En (Figura 71) se muestra el paquete que indica que la recepción de la solicitud, cuyo tipo de mensaje corresponde a (0101), con carga útil de 4 bytes al igual que su PDU; Además, se observa que, esta solicitud presenta en el primer timestamp un valor de 8.08 segundos.

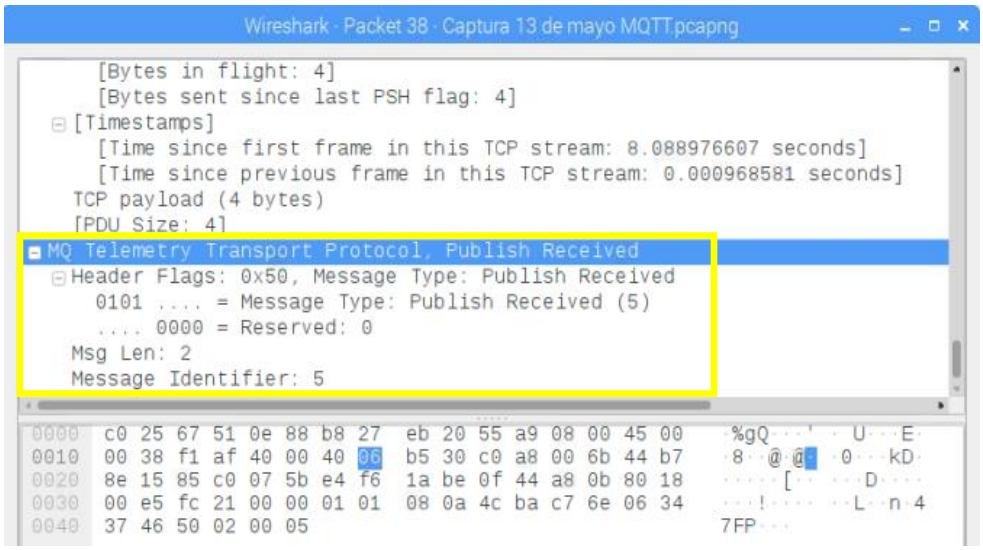


Figura 71. Solicitud de publicación receptada

En (Figura 72) se observa el paquete que confirma la publicación completa del mensaje. Posee tanto la longitud como su identificador; su código corresponde a (0111), el tamaño y PDU no difiere del anterior ya que es el mismo valor de 4 bytes. Sin embargo, el primer timestamp si toma un tiempo de 24.54 segundos, es decir tarda más por el mismo hecho de realizar una publicación.

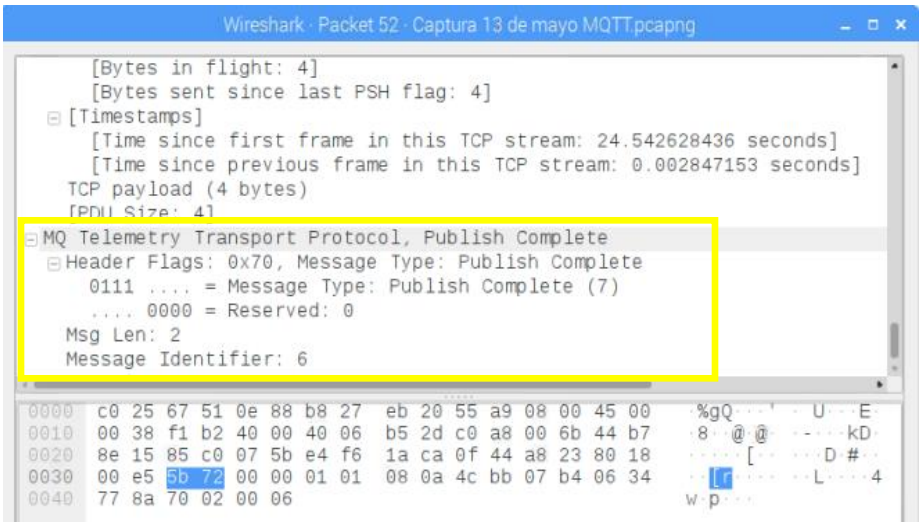


Figura 72. Publicación completa de mensaje

En uno de los paquetes MQTT, se observan los puertos y direcciones IP involucrados en la conexión. En este caso, el origen es el droplet con IP 68.183.142.21, y el destino con IP 192.168.0.107, que corresponde a la dirección de la Raspberry que se encuentra conectada mediante DHCP a un router con salida a Internet. Mientras que, los puertos responden a 1883 y 34240 respectivamente. (Figura 73) Además, se muestran las banderas y longitudes de los mensajes que indican 32 bytes.

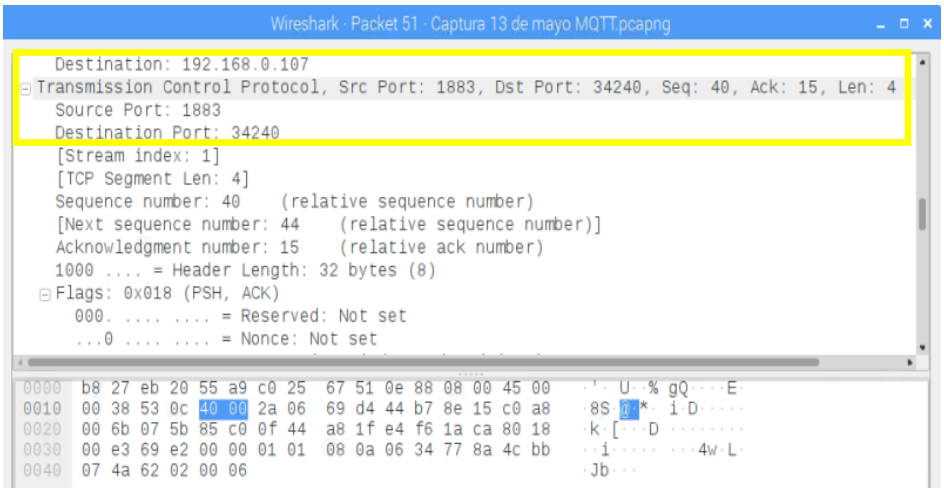


Figura 73. Puertos de origen y destino

Se realizó el mismo ejercicio varias veces, de modo que wireshark capture la mayor cantidad de paquetes, y con ellos realizar un análisis gráfico del throughput entre la Raspberry y el droplet. Para este trabajo, se obtiene un pico alto de 64 bits/s en un solo flujo de datos. Con lo que se comprueba que este protocolo es de bajo costo, no consume mayores recursos y sobretodo el ancho de banda que usa es mínimo. Lo que indica que dentro de una red en la que se involucren mayor número de sensores y actuadores, esta no se verá afectada en su rendimiento. Cabe indicar que dicha gráfica se puede capturar en Wireshark, ya que se tratar de un protocolo que se basa en TCP. (Figura 74)

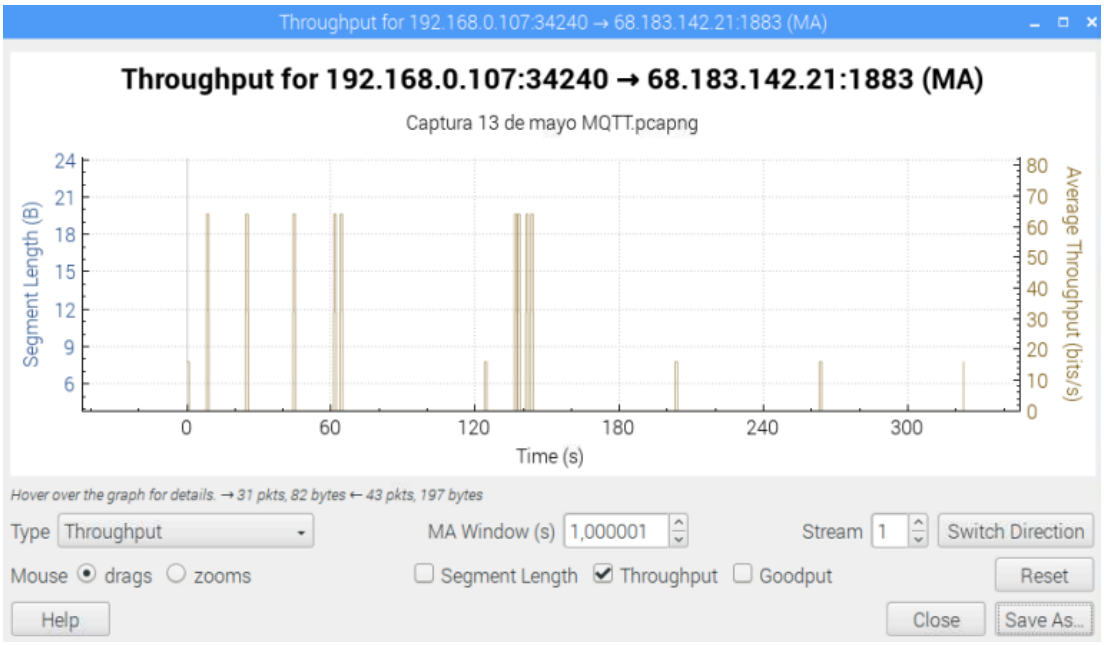


Figura 74. Throughput entre el ordenador y droplet

5.1.2. Resultados de Jperf

Se utilizó esta herramienta ya que se trata de un sofware que trabaja con base en la esructura cliente – servidor; sirve para realizar mediciones sobre el rendimiento de la red, es decir el ancho de banda, jitter. Además, tiene la ventaja de trabajar con flujos de datos tanto TCP como UDP, es multiplataforma, es decir que se puede usar tanto en Linux como en Windows; solo divergen en la forma de instalación.

Para el caso de MQTT, se realizaron varias pruebas con diferentes valores de ancho de banda, con 1, 5 y 10 Kbyte/s, y en todos los casos se usan 2 streams para una mayor comparación, durante un tiempo de 20 segundos. En (Figura 75) se observa que los dos flujos de tráfico mantienen constantes tanto el ancho de banda como su trasnferencia, no posee delay ni jitter, debido que se mantiene en cero. El tráfico de inyección de 1Kbyte/s para MQTT, es óptimo ya que se trata de un valor bajo y aporvecha las características de este protocolo.

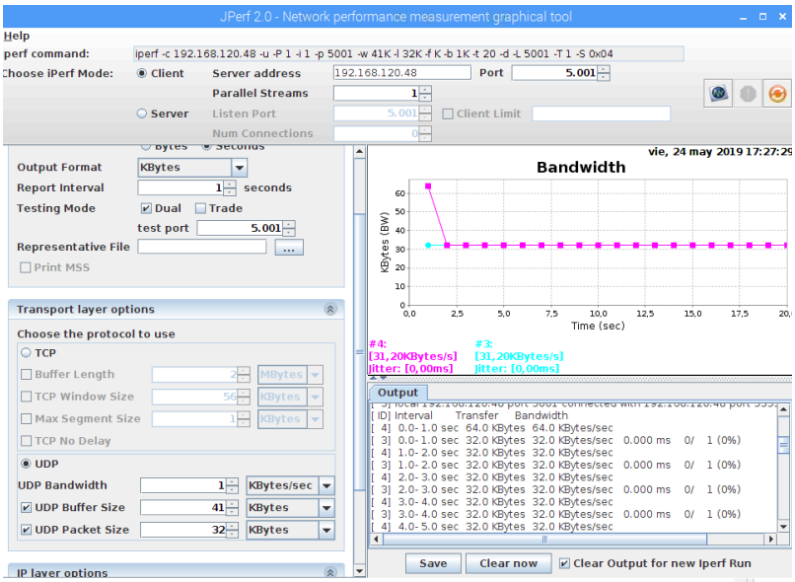


Figura 75. Inyección de tráfico de 1 KByte/s

En (Figura 76) se muestran los flujos correspondientes a 5 Kbytes/s, cuyos valores de trasnferencia, jitter y delay son los mismos que, en el caso de 1 Kbyte/s, lo que indica que, en el rango de los Kbytes al ser un rango bajo los parámetros de desempeño se mantienen constantes, sin importar su variación.

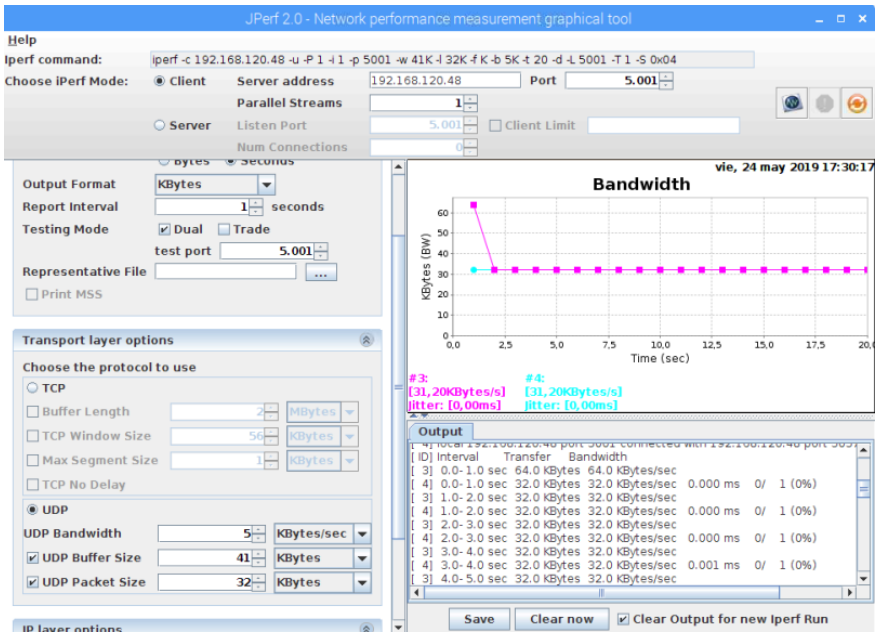


Figura 76. Inyección de tráfico 5 KByte/s

En (Figura 77), se ratifica la conjetura anterior, puesto que los valores de desempeño en este rango para MQTT, no varían, es clara su constancia, a pesar de la variación de inyección de tráfico. Por ello es que, se decidió no realizar la prueba de 10 Kbytes, debido a que los resultados anteriores

son repetitivos. Este evento muestra que, en inyecciones de Kbps el desempeño no varía en lo absoluto, por el mismo hecho de que se trata de trasnferencias mínimas.

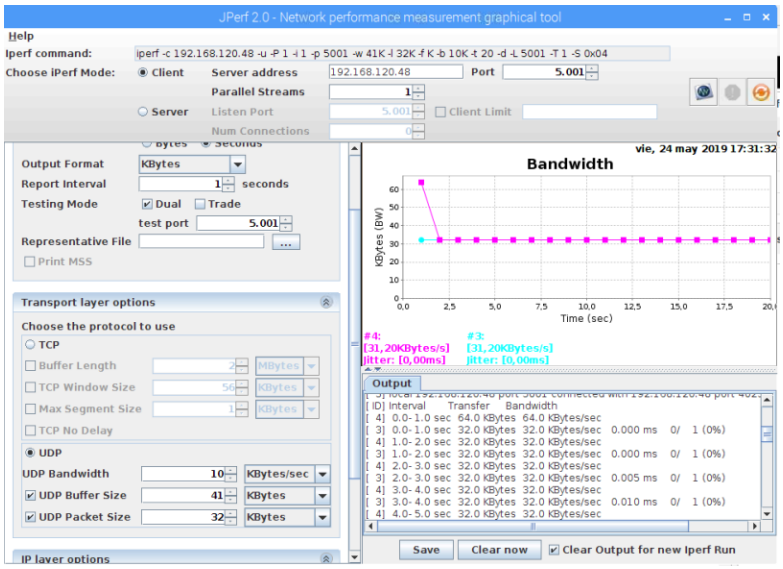


Figura 77. Inyección de tráfico 10 KBytes/s

Posteriormente se realizaron las inyecciones de tráfico con valores de 1, 5, 10, 15 y 20 MByte/s, ya que se consideró necesario verificar el comportamiento de MQTT, al elevar el tráfico. Puesto que, si bien es cierto no es un rango alto se deben notar ciertas diferencias con respecto a las pruebas anteriores. Las pruebas finalizan con este rango, ya que al tomar índices de los Gigabytes/s, se estaría sobredimiensionando el tráfico, de acuerdo a las características propias de los protocolos. En (Figura 78) se observa la inyección de 1Mbyte/s, con dos flujos en la que se evidencia la variacion de los tres parámetros de desempeño, transferencia, delay y jitter. Sin embargo, dicha variación no es significativa, no obstante es necesaria una tabulación para un mejor análisis. Se dice que no es significativa ya que se observan milésimas de segundos como diferencia.

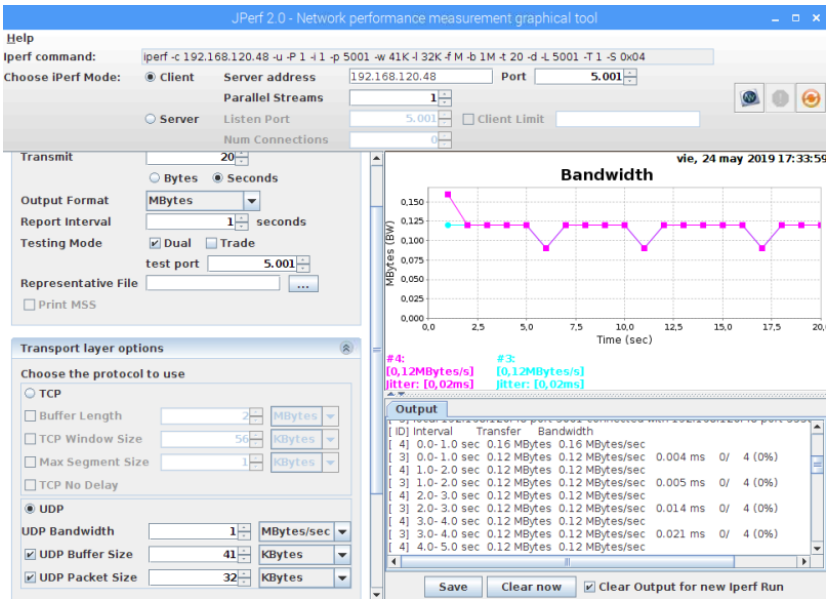


Figura 78. Inyección de tráfico 1 MBytes/s

En (Figura 79), se observa la inyección de 5 MBytes/s, en la que su transferencia varía alrededor de 0.01 Mbyte/s, mientras que el delay se denota que incrementa con respecto al caso anterior, y en consecuencia el jitter aumenta debido a la relación que existe entre estos valores.

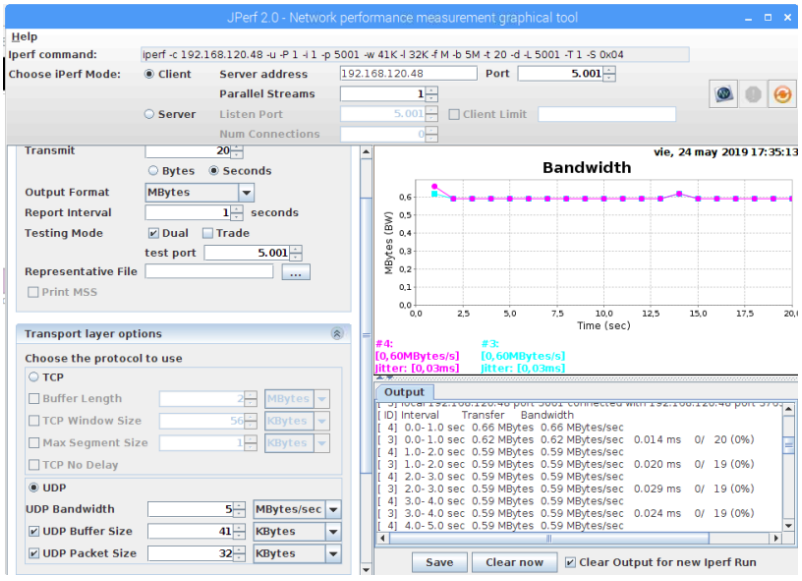


Figura 79. Inyección de tráfico 5 MBytes/s

En (Figura 80), se representa la inyección de 10 MBytes/s, cuyos valores de transferencia se ven afentados en los primeros segundos ya que tienen una pequeña variación; no obstante al poco tiempo se mantiene constante. El delay incrementa conforme el tiempo, y su jitter al parecer se mantiene en los primeros segundos de inyección.

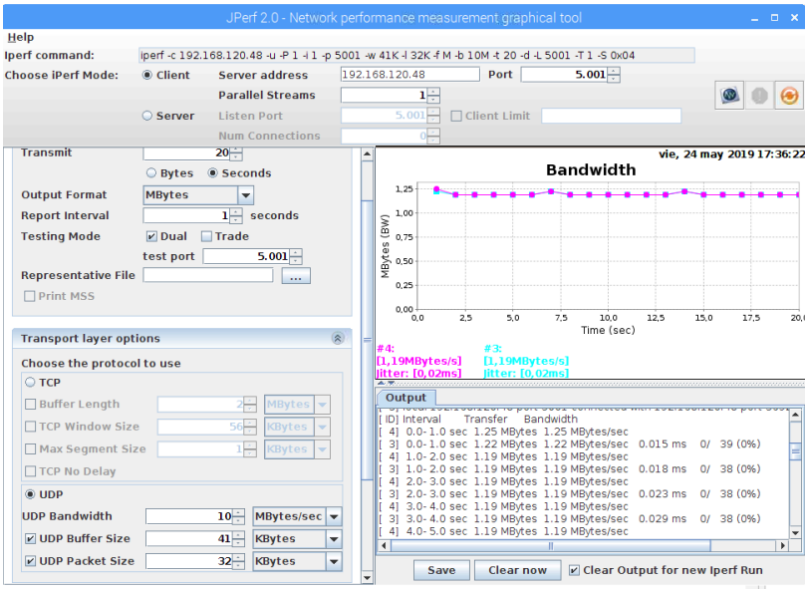


Figura 80. Inyección de tráfico 10 MBytes/s

En (Figura 81), se observan los valores de inyección de 15 MBytes/s, su transferencia se mantiene estable, su delay incrementa durante el tiempo; mientras que el jitter en el primer segundo mantiene un valor superior y después mantiene constancia.

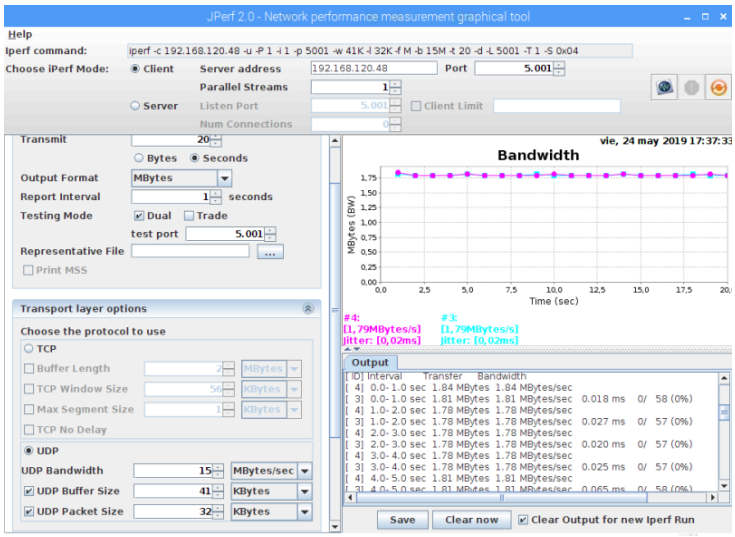


Figura 81. Inyección de tráfico 15 MBytes/s

En (Figura 82) se representan los valores correspondiente a la última inyección, en la que se observa que los valores de los factores de desempeño varían en algunos puntos, para verificar si constancia o estabilidad es necesaria la tabulación de todos los valores.

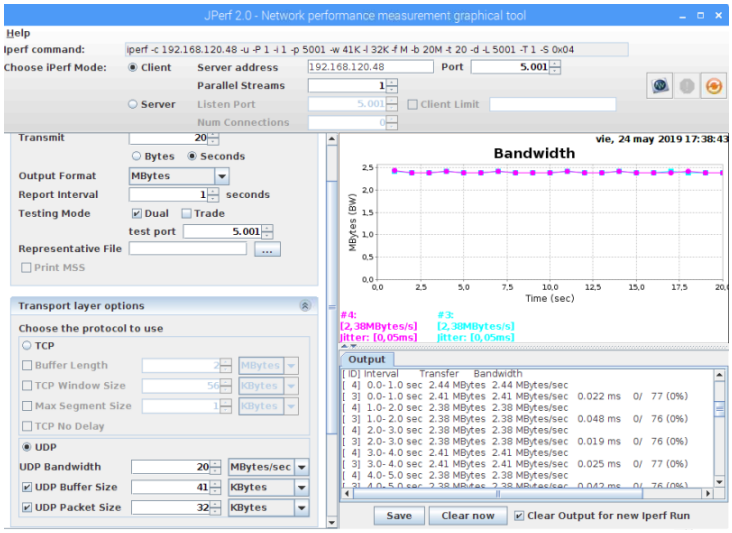


Figura 82. Inyección de tráfico 20 MBytes/s

5.2. Resultados de protocolo CoAP

Al igual que en el protocolo anterior, se realizó el mismo procedimiento para la obtención de resultados, con las herramientas wireshark y Jperf.

5.2.1. Resultados de Wireshark

Para el caso de CoAP, se obtuvo una secuencia de dos paquetes, tanto para el envío de la orden de encender como de apagar. Además, se muestran los ID de cada paquete, así como el código 2.05 CONTENT, y el método que se usó. (Figura 83)

No.	Time	Source	Destination	Protocol	Length	Info
10	0.02182…	localhost	localhost	HTTP	304	HTTP/1.1 200 OK (text/pl
11	0.02185…	localhost	localhost	TCP	66	49534 → vsat-control(1880
12	0.02267…	localhost	localhost	CoAP	68	CON, MID:18891, GET, TKN:
13	0.02915…	localhost	localhost	CoAP	62	ACK, MID:18891, 2.05 Cont
14	0.08028…	localhost	localhost	TCP	5688	vsat-control(1880) → 4869

Figura 83. Secuencia de paquetes CoAP

El primer paquete que se observó es el mensaje CON, es decir se conecta al servidor, se envía la orden para el control On/Off. Dentro del mismo paquete se evidenció el método que se usó es decir GET. (Figura 84)

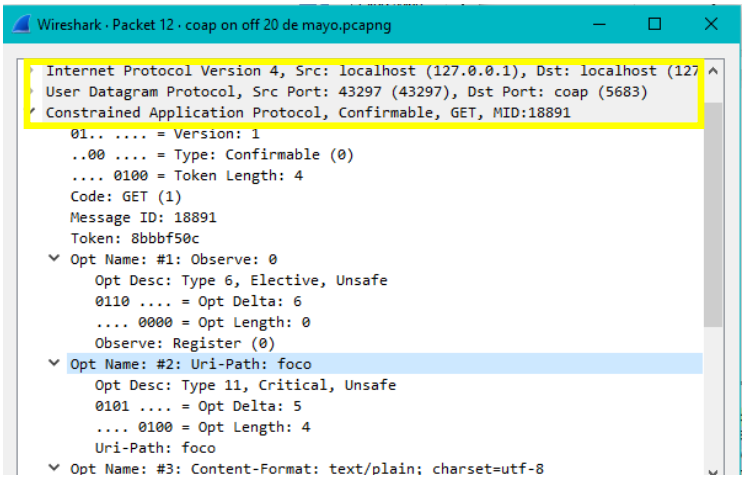


Figura 84. CON, GET de CoAP

Además, se observa el path que se nombró /foco, y el mensaje que en este caso corresponde a Encendido. Su carga útil se evidencia como texto, en código UTF-8, el mismo que dentro del flujo en Node – Red fue necesario decodificar para comprobar su envío. (Figura 85)

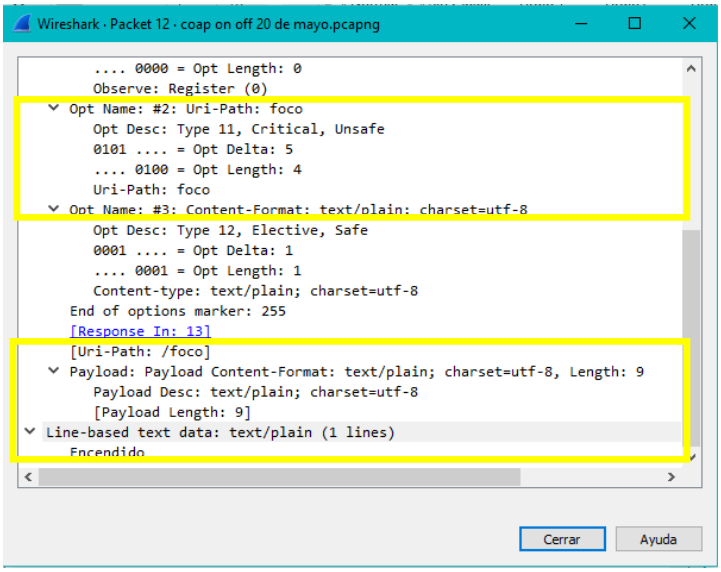


Figura 85. Uri-Path y payload del objeto

El segundo paquete que se capturó, es el ACK o acuse de recibo que corresponde al mensaje de confirmación, que se envía por parte del destinatario. Se observa tanto el puerto de destino como de origen, así como del código Content que significa satisfactorio 2.05 (Figura 86)

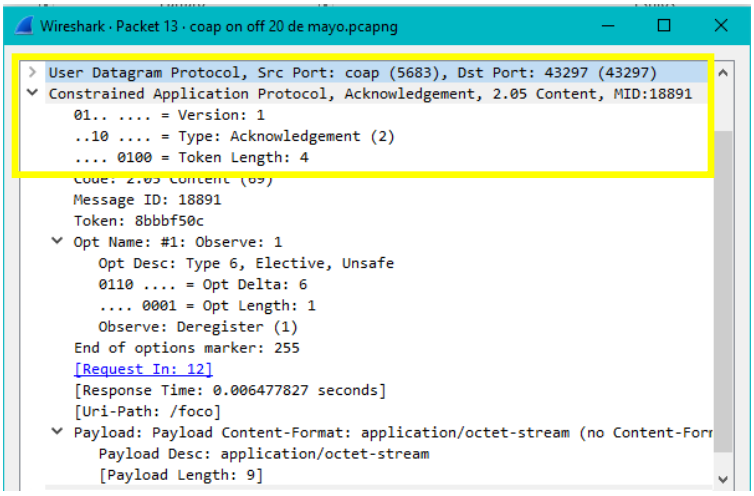


Figura 86. ACK de CoAP

En los dos paquetes se observan tanto los puertos de origen como de destino y el puerto de CoAP; se recuerda que se realizaron pruebas que involucraron a la dirección de loopback 127.0.0.1, a nivel local, debido a las características de CoAP en este tipo de entornos. (Figura 87).

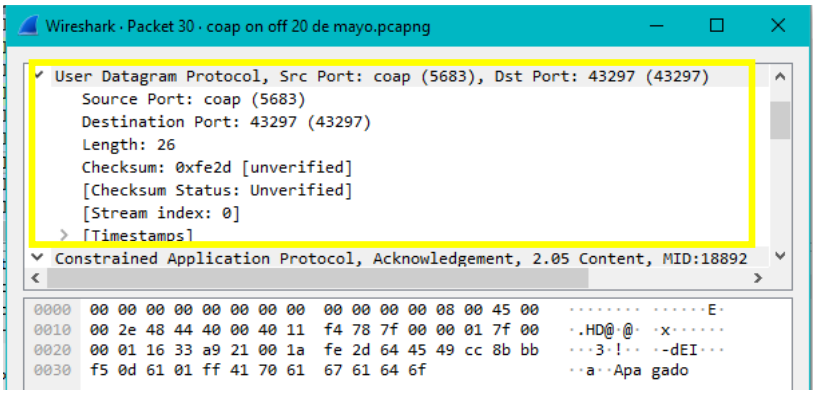


Figura 87. Puertos de origen y destino

Dentro del mismo paquete se observa el tamaño del Frame que corresponde a 60 bytes, la dirección de localhost que es la del microcomputador 127.0.0.1, y el puerto CoAP que se especificó dentro del flujo de programación 5683 (Figura 88).

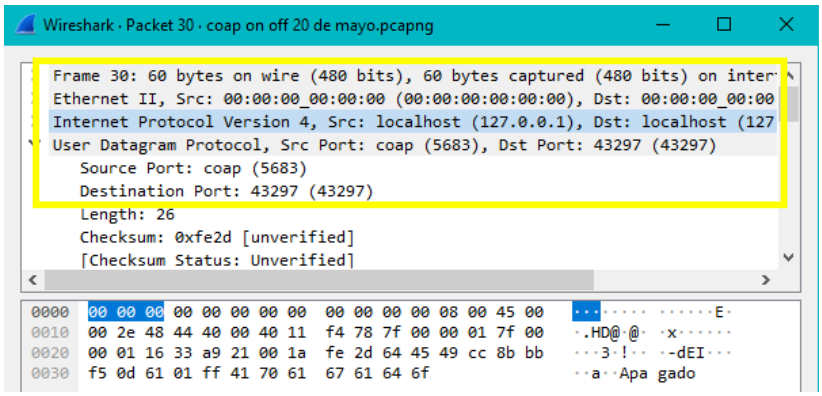


Figura 88. Dirección IP de localhost

Para verificar los mensajes que se trasmitieron, se tomó de la opción UDP Streams, se encontraron todos los mensajes de control On/Off del foco. (Figura 89)

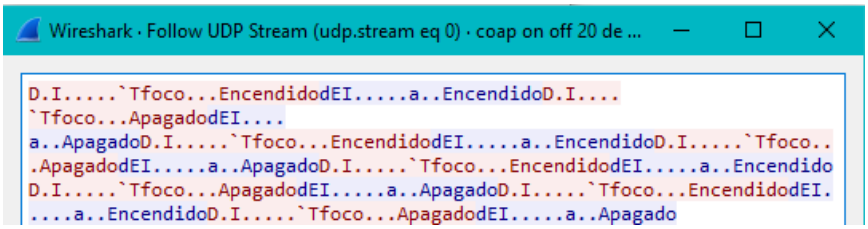


Figura 89. String de payload

Al finalizar la secuencia de paquetes se observó el código de CoAP (2.05 CONTENT), que indica que se recibió correctamente la solicitud y conexión al servidor. Además, se observa el código del Token y el ID del mensaje (Figura 90).

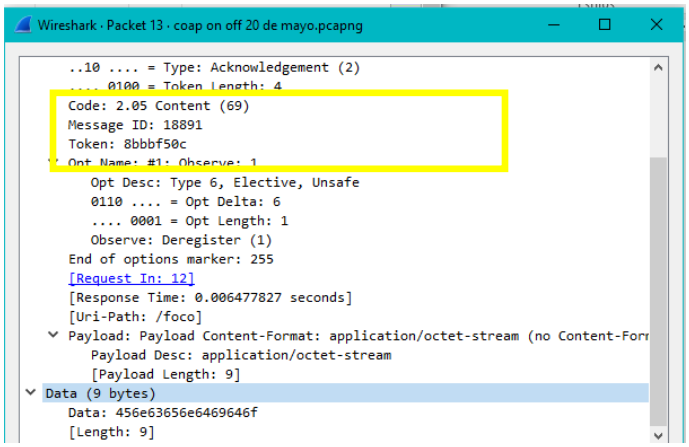


Figura 90. Código 2.05 de CoAP

5.2.2. Resultados de Jperf

Una vez que se realizaron las capturas de paquetes en Wireshark, se ejecutó la inyección de tráfico en el rango de los KBytes/s y MBytes/s, como en protocolo anterior, debido a su bajo costo.

Dentro de la inyección de 1 Kbyte/s, se observa que su transferencia varía considerablemente, además el delay varía en milésimas de segundos, lo cual explica que el jitter se mantenga estable (Figura 91). Sucede el mismo evento que se observó en MQTT.

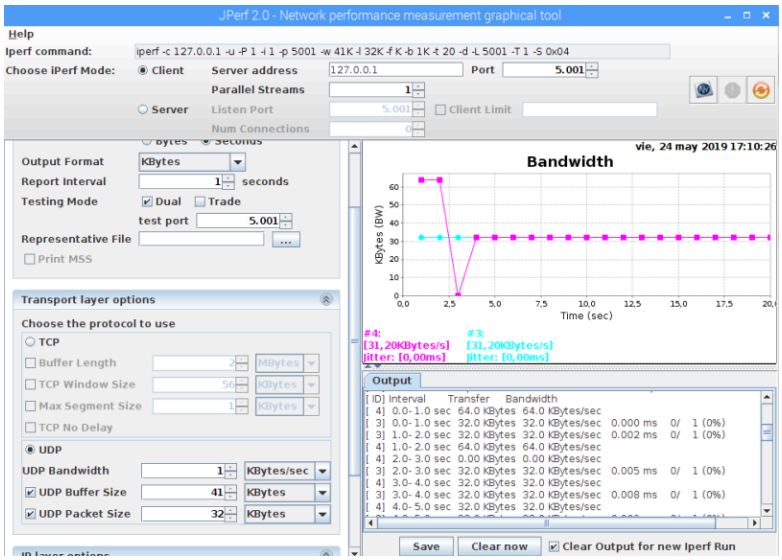


Figura 91. Inyección de tráfico 1 KByte/s

Los valores de inyección de 5 KBytes/s mantienen valores similares al caso anterior, por lo que se puede decir que en el rango de los KBytes/s no existen diferencias mayores (Figura 92).

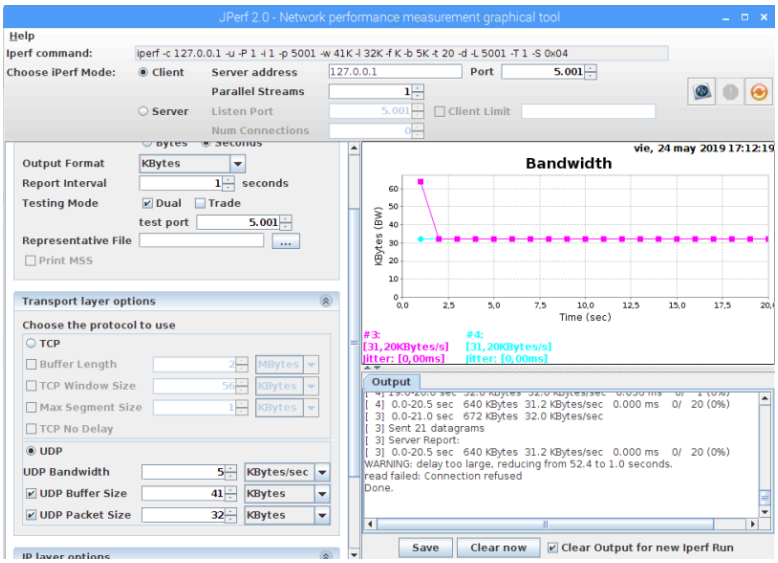


Figura 92. Inyección de tráfico 5 Kbyte/s

En (Figura 93) se observa la inyección de 10 KBytes/s, en la que se presenta constancia de todos los parámetros de desempeño, no existen variaciones.

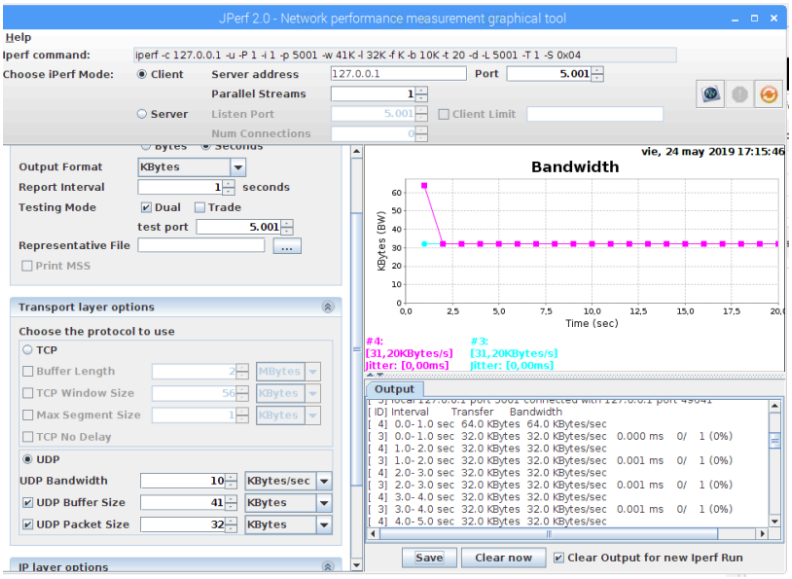


Figura 93. Inyección de tráfico 10 Kbyte/s

En el valor de inyección de 15 Kbyte/s (Figura 94) se observa estabilidad en la transferencia y jitter; además, una pequeña variación en su delay de 0.001 segundo.

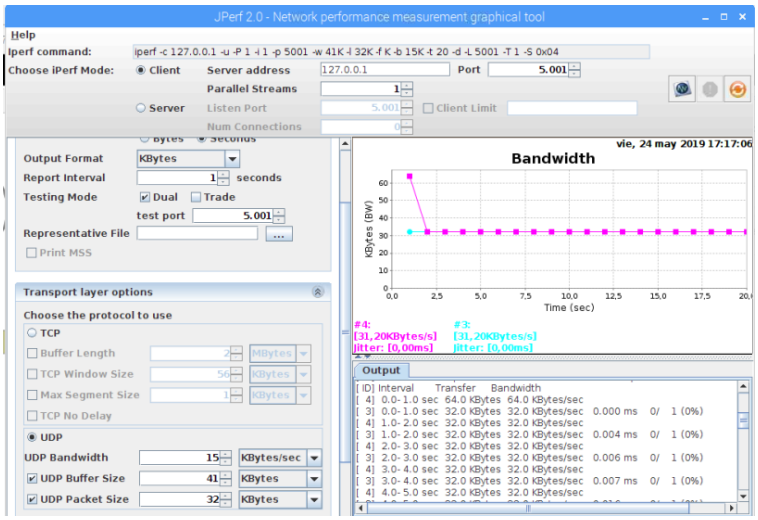


Figura 94. Inyección de tráfico 15 Kbyte/s

En (Figura 95) se observa valores parecidos a la inyección anterior. Y los parámetros se mantienen para el caso de 25 KBytes/s. Es decir que, al aumentar el ancho de banda, sus parámetros se mantendrán iguales.

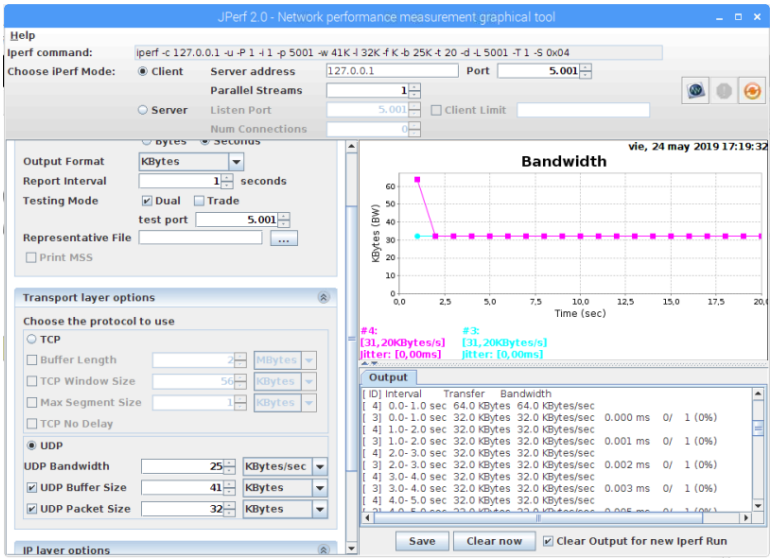


Figura 95. Inyección de tráfico 20 y 25 Kbyte/s

El mismo procedimiento se realizó para el rango de los MBytes/s. Por ejemplo, para el caso de (Figura 96), se observa estabilidad en la transferencia y delay; mientras que, el jitter varía.

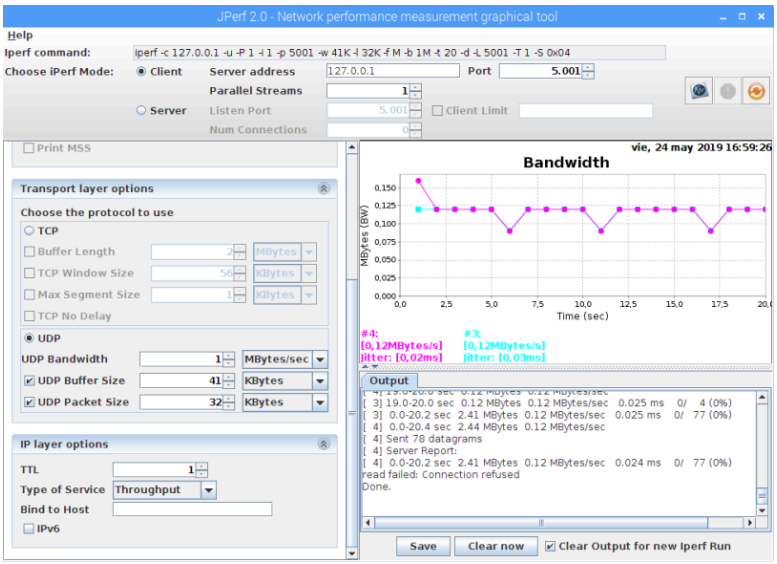


Figura 96. Inyección de tráfico 1 Mbyte/s

Para el caso de 5MBytes/s (Figura 97), se evidencia constancia en su transferencia y jitter. Además, existe un delay que varía notablemente.

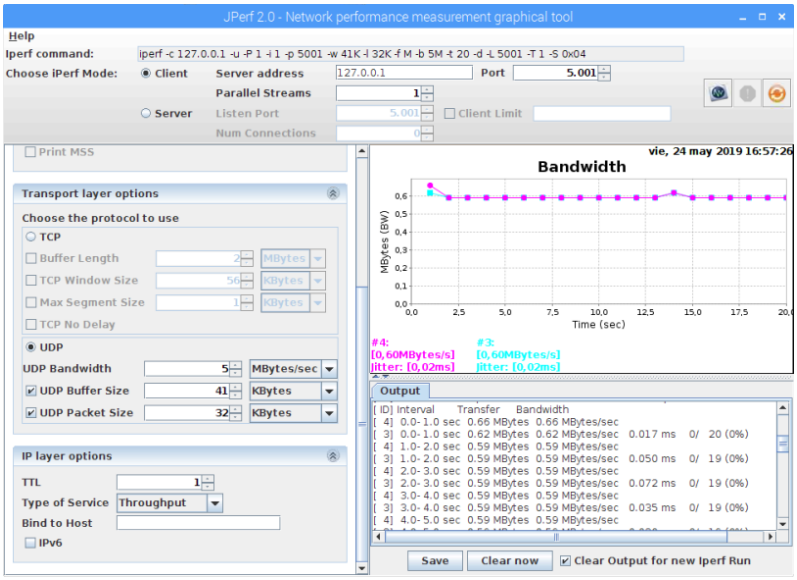


Figura 97. Inyección de tráfico 5 Mbyte/s

En la inyección de 10 MBytes/s, se verifica constancia de su transferencia y jitter; mientras que el delay mantiene una ligera variación (Figura 98).

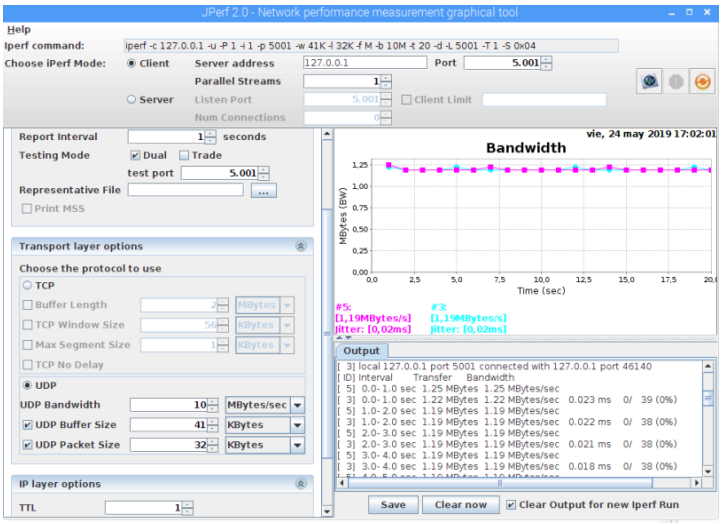


Figura 98. Inyección de tráfico 10 Mbyte/s

En (Figura 99) se observa los valores de inyección de 15MBytes/s, en los que mantienen estabilidad tanto en su transferencia y jitter; mientras que, su delay presenta variaciones a partir del segundo 2 de inyección.

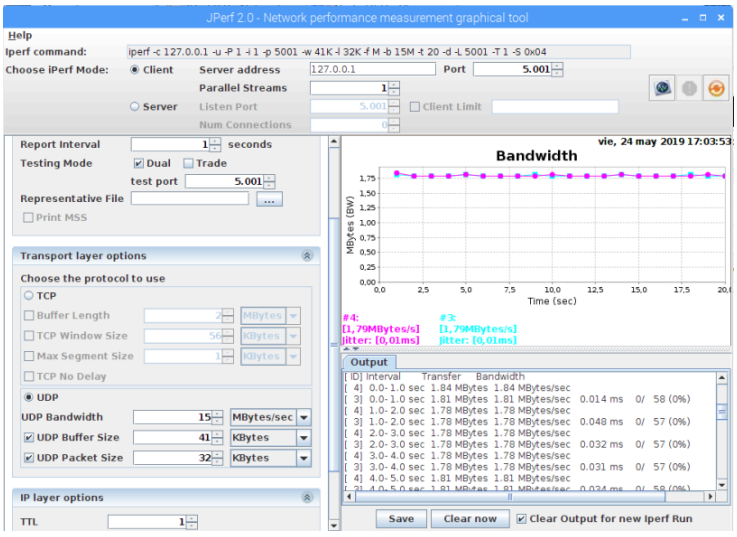


Figura 99. Inyección de tráfico 15 Mbyte/s

En el último factor de inyección de 20 MBytes/s, se presentan valores que varían en los tres parámetros de desempeño, para lo cual es indispensable tabular los datos para un mejor análisis (Figura 100).

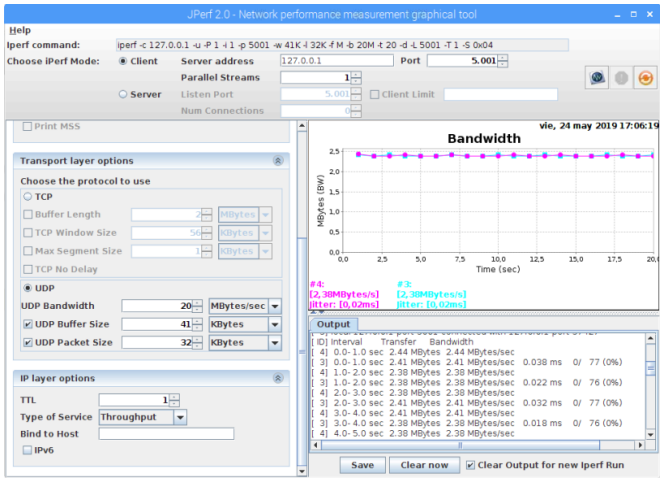


Figura 100. Inyección de tráfico 20 Mbyte/s

Se tabularon los datos que se obtuvieron mediante las inyecciones de tráfico para cada caso, se consideraron los parámetros de ancho de banda, delay y jitter, para el análisis del rendimiento en cada protocolo. En (Tabla 7) se encuentran los resultados con correspondientes a MQTT en Kbps, existe constancia en valores de retardo, mientras que su jitter es nulo en todos los casos. Lo que indica que, sin importar el valor de inyección en el rango de los Kbps, sus parámetros no se verán afectados, con un ancho de banda reducido y sin afectar la red en que se use.

Tabla 7
Resultados Jperf para MQTT en el rango de los Kbps

Parámetro de inyección - UDP Bandwidth [Kbps]	Transferencia [KBytes]	Ancho de banda [Kbps]	Delay [s]	Jitter [ms]
1	640	32	0.487	0.00
5	640	31.2	0.487	0.00
10	640	31.2	0.487	0.00
15	640	31.2	0.487	0.00
20	640	31.2	0.487	0.00

En (Tabla 8) los valores en Mbps, se observa que MQTT tiene tendencia a elevar su delay cuando el ancho de banda incrementa, a partir de los 5Mbps, su ancho de banda se incrementa de acuerdo al tráfico de inyección al igual que su transferencia. Cabe indicar que el jitter para estos casos es mínimo. Mientras que su delay sufre variaciones mínimas y se observa un ligero incremento. Se puede indicar que, tanto para el rango de Kbps como de Mbps, se tienen respuestas favorables, es decir no existiría cambios bruscos en el comportamiento esperado de una red; sin embargo, los cambios se evidencian solo en los casos de inyecciones en Mbps, lo cual es preciso analizar de forma gráfica.

Tabla 8
Resultados Jperf para MQTT en el rango de los Mbps

Parámetro de inyección - UDP Bandwidth [Mbps]	Transferencia [MBytes]	Ancho de banda [Mbps]	Delay [s]	Jitter [ms]
1	2.41	0.12	0,0005	0.021
5	11.9	0.60	0,0504	0.027
10	23.8	1.19	0,05	0.015
15	35.8	1.79	0,05	0.018
20	47.7	2.38	0,049	0.047

En (Tabla 9) se observa que el jitter es nulo; mientras que, los valores de transferencia, ancho de banda y delay son constantes y similares a los resultados de MQTT en el mismo rango de tráfico. Se ratifica el comportamiento de los dos protocolos en el rango de los Mbps.

Tabla 9
Resultados Jperf para CoAP en el rango de los Kbps

Parámetro de inyección - UDP Bandwidth [Kbps]	Transferencia [KBytes]	Ancho de banda [Kbps]	Delay [s]	Jitter [ms]
1	640	31.2	0,0487	0.00
5	640	31.2	0,0487	0.00
10	640	31.2	0,0487	0.00
15	640	31.2	0,0487	0.00
20	640	31.2	0,0487	0.00

En el rango de los Mbps, se tiene que CoAP posee un delay y jitter muy bajo, con un consumo de ancho de banda reducido, al igual que MQTT. Además, es evidente el incremento de transferencia a partir de 5Mbps, y en consecuencia de su ancho de banda (Tabla 10).

Tabla 10
Resultados Jperf para CoAP en el rango de los Mbps

Parámetro de inyección - UDP Bandwidth [Mbps]	Transferencia [MBytes]	Ancho de banda [Mbps]	Delay [s]	Jitter [ms]
1	2.41	0.12	0,0498	0.014
5	11.9	0.60	0,0504	0.024
10	23.9	1.19	0,049	0.023
15	35.8	1.79	0,05	0.014
20	47.7	2.38	0,0487	0.023

Como se observa en las tablas, el rendimiento puede estar por una misma línea; sin embargo, se graficaron los datos de modo que se puede evidenciar de mejor manera dichos parámetros por cada protocolo.

De esta forma se denota que, el delay en MQTT asciende junto con el ancho de banda, su pico más alto en este parámetro se observa en 2.4 Mbps aproximadamente; y a partir de los 3 Mbps se denota que el delay toma estabilidad, debido al nivel de QoS para este protocolo, que puede causar cierta demora ya que requiere garantizar la entrega del mensaje al destinatario. La variación de delay para este caso es de milésimas de segundos, lo cual no afectaría al desempeño de los protocolos dentro del entorno. Mientras que, para CoAP los valores de delay permanecen constantes, se observa un ligero cambio en 3 Mbps, finalmente los dos protocolos hallan una convergencia a partir de este mismo valor. (Figura 101)

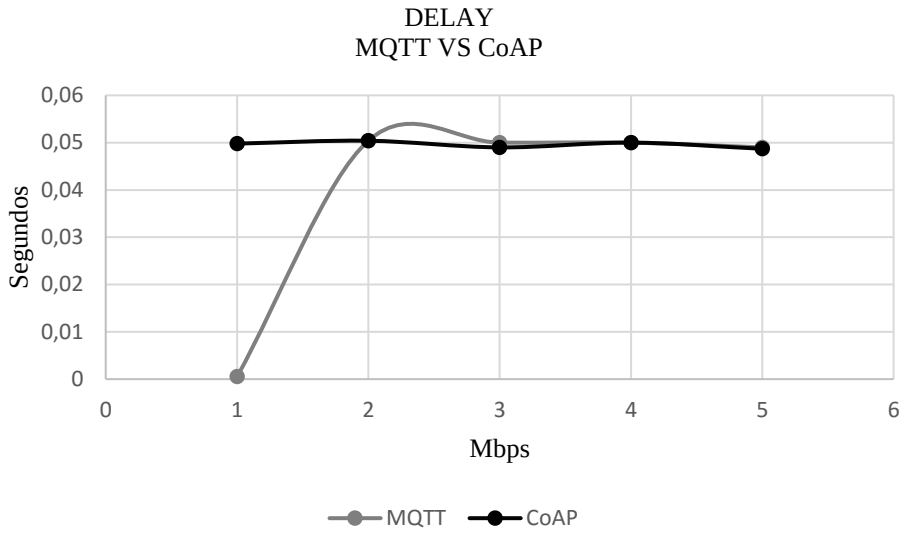


Figura 101. Delay en MQTT y CoAP

En cuanto al jitter de estos protocolos, mantienen una misma forma en ambos casos, con una leve variación, esta variación corresponde al rango en que el delay de MQTT y CoAP todavía no logran su convergencia. Se observa también que en MQTT el delay tiende a incrementar si el ancho de banda aumenta a partir de los 5 Mbps, en este caso específico. (Figura 102)

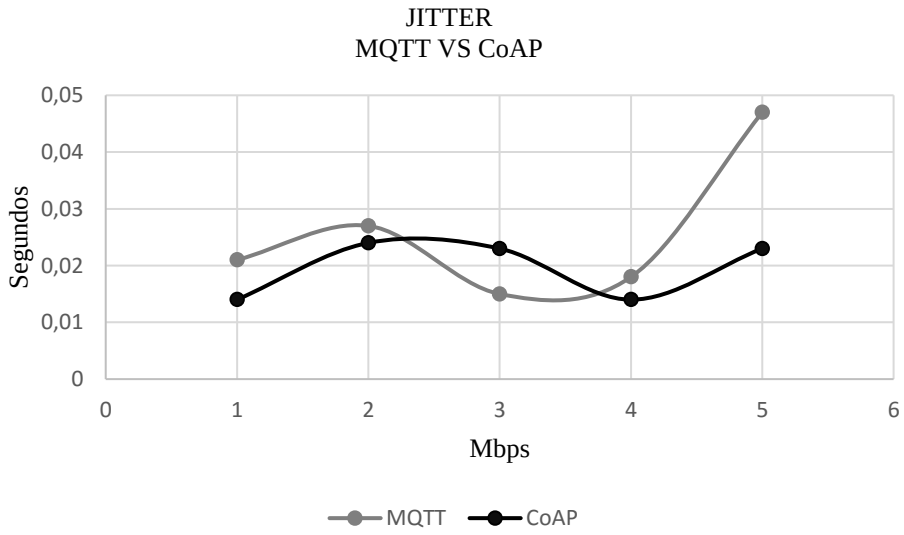


Figura 102. Jitter en MQTT y CoAP

Además, los pronunciamientos de las curvas se deben a la variación del delay, por ello es que se observa la curva de MQTT más pronunciada que la curva de CoAP, debido al comportamiento del delay en MQTT.

En las siguientes gráficas se realizó un análisis independiente de cada protocolo con sus parámetros de rendimiento. Para lo que corresponde a MQTT, es evidente la variación del delay en el jitter, ya que este logra estabilizarse a partir de los 3 Mbps de velocidad, es por ello que la curva de jitter se nota más pronunciada, si se considera que el nivel de calidad de servicio es 2. (Figura 103)

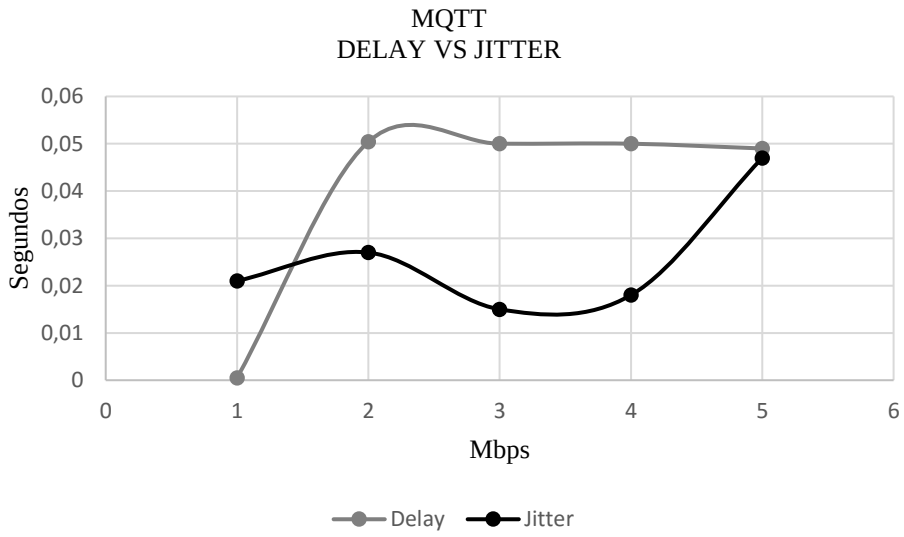


Figura 103. Delay y Jitter en MQTT

En cuanto al protocolo CoAP, los valores de retardo o delay, son constantes a pesar de la variación de velocidad, por ello la curva de jitter posee un leve pronunciamiento; sin embargo, tanto el delay como jitter son aceptables. (Figura 104)

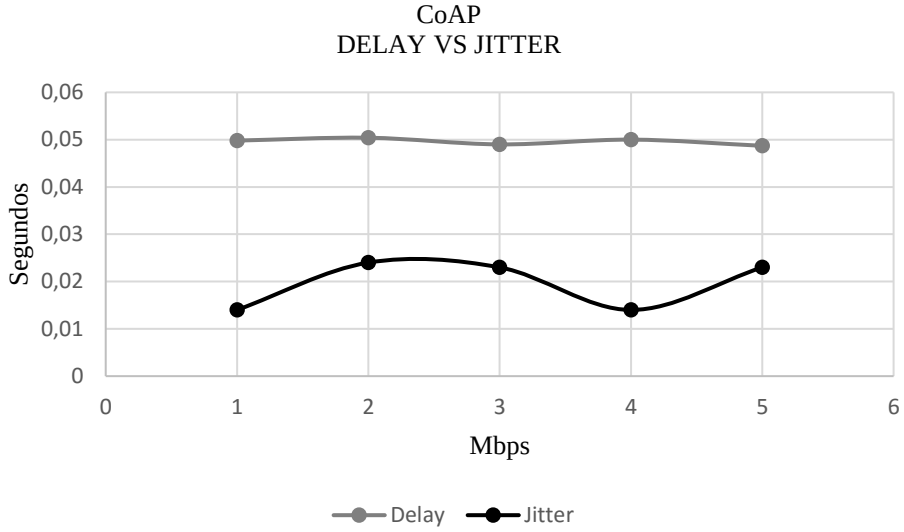


Figura 104. Delay y Jitter en CoAP

En caso de que el jitter tome valores superiores a 100 ms, se puede decir que la red con aplicación IoT se vería afectada considerablemente, ya que existiría congestión, pérdidas de paquetes y dificultad para la entrega de mensajes a su destino. Sin embargo, para este proyecto se observa que, la variación máxima para CoAP y MQTT en cuanto al jitter es de 0.024 ms y 0.047 ms respectivamente, son valores que pueden ser compensados sin dificultad, y que a su vez no representan complicaciones en este tipo de entornos y en sus entregas de paquetes.

Con respecto al delay de los protocolos, se observa que para ambos casos la variación máxima es de 50 ms, tiempo que es imperceptible al momento de la ejecución de un control on/off de una lámpara. No obstante, cada aplicación tiene sus propios requerimientos, ya que puede tratarse de eventos altamente críticos en los que, se requieran menor tiempo de retardo, por ejemplo, la aplicación de un medicamento en un paciente con peligro de muerte.

Finalmente, tanto MQTT como CoAP tienen un buen desempeño, con valores de alta aceptación, por ello es pertinente el estudio del ambiente y la aplicación en que se desarrollaría cada uno de ellos, ya que, de acuerdo los resultados dentro de este proyecto se observó más dificultad en la fase de implementación en cuanto a CoAP, para obtener resultados similares a MQTT. Por lo que se afirma, que MQTT no solo tiene un buen desempeño, sino que ofrece facilidad de implementación, lo que ayudaría a definir la estandarización que hace falta.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

- Con base a los resultados tanto en la fase de implementación como de análisis, se concluye que MQTT es el protocolo con mejor desempeño debido a los valores de jitter y delay que se obtuvieron y a su facilidad para implementación en un entorno doméstico.
- Se analizó el estado del arte, arquitecturas, desarrollos y protocolos de IoT en la capa de aplicación, tanto para MQTT como de CoAP, y se concluye que la capa apta para entornos IoT es la de cinco capas; de este modo se definen de mejor manera sus funciones y protocolos adecuados. Con ello se puede superar la falta de estandarización y el tema de interoperabilidad de dispositivos, en consecuencia, incrementar los desarrollos y sectores de implementación.
- Se utilizó el mismo prototipo para la implementación de CoAP y MQTT, el mismo que responde sin ninguna complicación y consta básicamente de una lámpara doméstica y un microcomputador. La única diferencia que existe es que, para CoAP se usa en una red local y para MQTT un servidor en la nube, esto debido a que CoAP, tiene como característica propia la transparencia a NAT.
- En la fase de implementación el protocolo que resultó más sencillo de poner en marcha es MQTT, puesto que este mantiene actualización oportuna y sus librerías no causan

inconvenientes en cuanto a su instalación; mientras que, CoAP posee su última actualización en el año 2014.

- Se usaron alternativas para la implementación de CoAP debido a la dificultad que se presentó, entre ellas Nodejs, Copper, Californium, TxThings, y Node-RED; luego de correcciones y reinstalación de algunas de sus librerías, se concluye que Node – Red es la mejor opción puesto que, brinda la posibilidad de programación visual con un ambiente amigable, sin necesidad de conocimientos avanzados en lenguaje Phyton.
- Se concluye que CoAP en Node-RED, es limitado en cuanto al tipo de dato de entrada, ya que no es versátil y únicamente acepta datos de tipo String; es por ello que, limita el uso de interfaces de usuario y se requiere añadir funciones para la conversión de datos.
- MQTT y CoAP ofrecen varias presentaciones útiles, y de acuerdo a lo observado en este trabajo, se puede concluir que CoAP se direcciona a aplicaciones de lectura de datos y MQTT a ejecución de órdenes en actuadores.

6.2. Recomendaciones

- Se recomienda un profundo análisis del entorno a implementar, puesto que cada necesidad es diferente y los protocolos se pueden adaptar de acuerdo al ambiente en que se ejecuten, considerando la red de sensores y actuadores.
- En caso de requerir la implementación de CoAP con una plataforma Cloud, se recomienda levantar una infraestructura tunneling y así contrarrestar el reenvío de puertos con NAT.
- Se recomienda tomar como referencia MQTT y CoAP en cuanto a protocolos IoT, para la generación de nuevas investigaciones y desarrollos, de este modo contribuir a la solución de estandarización

- A pesar de las prestaciones y funcionalidades que tanto CoAP como MQTT ofrecen, y en especial MQTT, es importante mencionar que la seguridad es un tema muy trascendental que se debe potencializar, ya que en el caso de MQTT al estar comunicado con un intermediario, este puede convertirse en un punto vulnerable de ataques, por ello se recomienda fortalecer y el ámbito de seguridad.
- En cuanto al uso de Raspberry, se recomienda realizar backups periódicamente; puesto que, al tener problemas de instalación de librerías de protocolos, puede ocasionar conflictos con archivos del sistema operativo del microcomputador, lo que puede forzar a un formateo y en consecuencia pérdida de información.

REFERENCIAS

Acciona. (2018). ¿Qué es una Smart City? Top 5 ciudades inteligentes. Recuperado a partir de http://www.sostenibilidad.com/que-es-una-smartcity-top-5-ciudades-inteligentes

Alcaraz, M. (2014). Internet de las Cosas. *PMQuality*, 1–27. https://doi.org/10.1016/j.cirp.2011.03.145

Añez, P. (2018). *Diseño e implementación de un "smart object" con conexión "wireless"*. https://doi.org/10.1007/s10956-014-9514-8

Arantza, A. (2016). Google presenta una nueva aplicación de chat inteligente. Recuperado a partir de https://es.blastingnews.com/tecnologia/2016/05/google-presenta-una-nueva-aplicacion-de-chat-inteligente-00928383.html

Blanco, M., González, K., & Rodriguez, J. (2017). *Propuesta de una arquitectura de la industria 4.0 en la cadena de suministro desde la perspectiva de la Ingeniería Industrial*. *Ingeniería Solidaria* (Vol. 13(23)). https://doi.org/https://doi.org/10.16925/in.v23i13.2007

Calatayud, M. M. (2018). ¿Cómo las empresas pueden hacer rentable la ecología gracias a blockchain?

Camacho, J., Oropeza, E., & Lozoya, O. (2017). Internet de las cosas y Realidad Aumentada: Una fusión del mundo con la tecnología. *Año*, *6*(1), 139–150.

Cañete, J. (2015). *Análisis de seguridad del protocolo 6LoWPAN*.

Castro, J. (2014). *Uso del protocolo CoAP para la implementación de una aplicación domótica con redes de sensores inalámbricas*. Recuperado a partir de

http://repositorio.upct.es/xmlui/handle/10317/4163

Catalán, C., Serna, F., & Blesa, A. (2015). Industria 4.0 en el Grado de Ingeniería Electrónica y Automática. *Actas de las XXI Jornadas de la Enseñanza Universitaria de Informática*, 327–332. Recuperado a partir de https://upcommons.upc.edu/bitstream/handle/2117/78299/JENUI2015_337-342.pdf

Cázarez, G. (2010). Diseño de un prototipo didáctico para la implementación de redes de sensores inalámbricos basados en el protocolo ZIGBEE. *Agriculture*, *6*, 199–219.

CoAP Tutorial for Raspberry Pi. (2017).

CSIRT-CV. (2017). Seguridad en Internet de las cosas Estado del Arte.

Domínguez, A., & Vargas-Lombardo, M. (2018). El estado del arte: Salud inteligente y el internet de las cosas, *14*, 14–17.

Ermesh. (2018). El protocolo MQTT explicado de forma sencilla.

Esquivel, E. (2016). IoT y su impacto social.

Evans, D. (2011). Internet de las cosas Internet de las cosas Cómo la próxima evolución de Internet lo cambia todo.

Garrido, J. (2016). *Estudio y análisis de escenarios de transmisión de datos orientados a Smart C ities*.

Gimenez, X. (2013). *Desarrollo y Estudio del Protocolo Observe para CoAP*.

Github. (2014). CoAP — Constrained Application Protocol | Implementations. Recuperado a

partir de http://coap.technology/impls.html (letzter Abruf am 2018-02-27)

GitHub - mkovatsc_Copper4Cr_ Copper (Cu) CoAP user-agent for Chrome (JavaScript implementation). (2018). Recuperado a partir de https://github.com/mkovatsc/Copper4Cr/commit/9a015d62da0aa9a2cdbe756ed54d11252bc7cb82

González, A. (2017). IoT : Dispositivos , tecnologías de transporte y aplicaciones.

Hernández Rojas, D. L., Mazon Olivo, B. E., & Campoverde Marca, A. M. (2015). Cloud Computing Para El Internet De Las Cosas. Caso De Estudio Orientado a La Agricultura De Precisión. *I Congreso Internacional De Ciencia Y Tecnología Utmach 2015*. Recuperado a partir de http://repositorio.utmachala.edu.ec/bitstream/48000/4972/1/0040- I Congreso Internacional de Ciencia y Tecnología UTMACH 2015

Jaffey, T. (2014). MQTT and CoAP, IoT Protocols. Recuperado a partir de https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php

Jurado, L., Velásquez, W., & Vinueza, N. (2014). Estado del Arte de las Arquitecturas de Internet de las Cosas (IoT).

Kovatsch, M., & Vermillard, J. (s/f). Hands-on with CoAP - Google Slides.

Kumar, P. (2017). How to Install and Use Wireshark on Debian 9 _ Ubuntu 16. Recuperado a partir de https://www.linuxtechi.com/install-use-wireshark-debian-9-ubuntu/

Luzuriaga, J. E., Zennaro, M., Cano, J. C., Calafate, C., & Manzoni, P. (2016). Evaluando un escenario de pruebas para el IoT entre la emulación y el uso de dispositivos reales. *Actas*

Jornadas Sarteco 2016, (December), 441–445. https://doi.org/978-84-9012-626-4

Martínez, D., & Guillen, E. (2015). Diseño de un sistema en Cloud, para controlar dispositivos IoT vía web. https://doi.org/10.1016/j.jde.2004.10.006

Martínez, R. (2016). Retos tecnológicos en la IoT en el ámbito de las redes de sensores, (Plan 3052).

Martos, V. (2011). *Implementación de una plataforma de juegos multijugador para Android*. *Arantxaiiuames Arantxaiiuames*. Recuperado a partir de http://www.uniovi.net/calidad/procesos/Difusion/Guias/pdf/1011/epi/4_planes_antiguos.pdf#page=76

Melián, A., & Anías, C. (2015). Gestión de la comunicación máquina a máquina (M2M). *Revista Cubana de Ingeniería*, *VI VI*(2), 49–56. https://doi.org/2223 -1781

Metodología de la investigación. (2016). Recuperado a partir de https://www.significados.com/metodologia-de-la-investigacion/

Node Red. (s/f). node-red-contrib-coap - Node-RED.

Noronha, A., Moriarty, R., O'Conell, K., & Villa, N. (2014). El valor de IoT: cómo pasar de conectar cosas a obtener información. *Cisco*, *1*(1), 20.

Npm. (2018). Npm - Build amazing things. Recuperado a partir de https://www.npmjs.com/

NPM. (s/f). coap-cli - npm.

Oasis. (2018). MQTT Version 5.0. https://doi.org/10.3350/kjhep.2010.16.1.66

Open IOT Challenge, AttA plays well with Node-RED. (2016).

Osako, H., Josias, E., & Gimenez, C. (2016). Internet das Coisas : o desafio de estabelecer um padrão único de protocolo.

Raspberry Pi Foundation. (s/f). Raspberry Pi 3 Model B+ - Raspberry Pi. Recuperado a partir de https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

Rodrigues, I., & Zem, J. L. (2016). Estudo dos Protocolos de Comunicação MQTT e CoAP para Aplicações Machine-to-Machine e Internet das Coisas. *Revista Tecnológica da Fatec Americana*, *3*(1), 24. Recuperado a partir de http://www.fatec.edu.br/revista_ojs/index.php/RTecFatecAM/article/view/41/50

Rodríguez, C. (2016). *Desarrollo de una pasarela doméstica para comunicaciones seguras en entornos IoT*.

Román, J. L. del V. (2016). *Industria 4.0: La Transformación Digital de la Industria Española*. *coddiinforme*.

Romero, J. (2016). Internet of Things, 1–17.

Rose, K., Eldridge, S., & Chapin, L. (2015). La Internet de las cosas - Una breve reseña. *Internet Society*.

Semle, A. (2016). Protocolos IIoT para considerar, 32–35.

Sengul, C., & Kirby, A. (2017). MQTT-TLS profile of ACE draft-sengul-ace-mqtt-tls-profile-00 This. *ACE Working Group*, *134*(4), 635–646.

Serrano-Cobos, J. (2016). Tendencias tecnológicas en internet: hacia un cambio de paradigma. *El*

Profesional de la Información, *25*(6), 843. https://doi.org/10.3145/epi.2016.nov.01

Shelby, Z., Hartke, K., & Bormann, C. (2014). *The Constrained Application Protocol (CoAP)*. https://doi.org/10.17487/rfc7252

Stansberry, J. (2015). MQTT and CoAP: Underlying Protocols for the IoT. *Electronic Design*, 1–8.

Vargas, D. C. Y., Salvador, C. E. P., Yacchirema, D. C., & Palau, C. (2016). Smart IoT Gateway For Heterogeneous Devices Interoperability. *IEEE Latin America Transactions*, *14*(8), 3900–3906. https://doi.org/10.1109/TLA.2016.7786378

Vouzis, P. (2018). How to use Jperf. Recuperado a partir de https://netbeez.net/blog/how-to-use-jperf/

Yuan, M. (2017a). Conozca MQTT. Recuperado a partir de https://www.ibm.com/support/knowledgecenter/es/SSFKSJ_7.5.0/com.ibm.mm.tc.doc/tc00150_.htm

Yuan, M. (2017b). Conozca MQTT. *04-10*. Recuperado a partir de https://www.ibm.com/developerworks/ssa/library/iot-mqtt-why-good-for-iot/index.html

Zabala, L. (2016). Gestión de la seguridad en el internet de las cosas. *Globb Security*. Recuperado a partir de http://globbsecurity.com/seguridad-iot-y-mundo-3-0-37652/