Universidad Autónoma de Querétaro
Facultad de Informática
Maestría en Sistemas Computacionales

Propuesta y evaluación de un middleware tipo gateway que permita la interoperabilidad entre dispositivos LoRaWAN y sistemas OPC-UA Pub-Sub para sistemas IoT.

Tesis

Que como parte de los requisitos para obtener el Grado de

Maestro en Sistemas Computacionales

Presenta

Ing. Wilbert Magdiel Caballero López

Dirigido por:

M.C. José Arturo Gaona Cuadra

Co-dirigido por:

M.C Alejandro Madariaga Navarrete

M.C José Arturo Gaona Cuadra
Presidente

M.C Alejandro Madariaga Navarrete
Secretario

MISD Carlos Alberto Olmos Trejo
Vocal

M.C. Alberto Lamadrid Álvarez
Suplente

MSI José Alfredo Acuña García
Suplente

Centro Universitario, Querétaro, Qro.
Agosto 2019
México

A Dios por su amor en mi vida y permitirme realizar esta investigación.

A mi esposa por todo su apoyo y paciencia durante estos meses.

A mi familia por todas sus enseñanzas que han forjado mi vida.

En memoria de nuestro compañero Juan Pablo Gutiérrez Oliva.

AGRADECIMIENTOS

Agradezco a todos mis maestros por su alto profesionalismo, experiencia y empeño en cada una de sus clases.

A mis asesores y tutores de tesis por todas las horas que me obsequiaron, las enseñanzas, correcciones y los consejos durante esta investigación.

A mis compañeros por todo su apoyo, palabras de ánimo, amistad y momentos que compartimos.

Agradezco al CONACYT por el apoyo brindado en la realización de esta investigación.

También agradezco a la UAQ por el apoyo brindado mediante el programa FOPER para la realización del prototipo.

ÍNDICE

1. INTRODUCCIÓN 9

2. ESTADO DEL ARTE 11

2.1 RETOS EN SISTEMAS *MIDDLEWARE* IOT 14

3. JUSTIFICACIÓN 17

4. DESCRIPCION DEL PROBLEMA 18

5. HIPÓTESIS 20

5.1 DELIMITACIÓN Y ALCANCES 20

5.2 LA HIPÓTESIS 20

5.3 PREGUNTAS DE INVESTIGACIÓN 21

6. OBJETIVOS 22

6.1 OBJETIVO GENERAL: 22

6.2 OBJETIVOS PARTICULARES: 22

7. TRABAJOS RELACIONADOS 23

8. METODOLOGÍA 24

8.1 METODOLOGÍA 24

9. MARCO TEÓRICO 28

9.1 INTERNET DE LAS COSAS (*IOT*) 28

9.2 SISTEMAS DISTRIBUIDOS 29

9.3 COMPUTACIÓN MÓVIL Y UBICUA 33

9.4 SISTEMAS DISTRIBUIDOS MASIVOS Y REDES DE MONITOREO 34

9.5 MIDDLEWARE 34

9.5.1 Middleware orientado a mensajes (MOM) 37

9.6 PROTOCOLOS ORIENTADOS A MENSAJES 39

9.6.1 Cola de Mensajes 39

9.6.2 Modelo Punto a Punto 39

9.6.3 Modelo Publicador – Subscriptor 39

9.6.4 MQTT 40

9.7 DISPOSITIVOS *LTN*, *LPWAN* Y *LORAWAN* 42

9.8 DATAGRAMAS UDP 50

9.9 OPC-UA 51

9.9.1 Terminología 53

9.9.2 Alcance de la Arquitectura Unificada 56

9.9.3 El Espacio de Direcciones 57

9.9.4 Sesiones 58

9.9.5 Descubrir 59

9.9.6 Canal seguro 59

9.9.7 Vistas 59

9.9.8 Consulta 60

9.9.9 Métodos 60

9.9.10 Subscripción 60
9.9.11 Modelo Objeto 60
9.10 Sistemas Ciber Físicos (CPS) 61

10. ARQUITECTURA PROPUESTA 62
10.1 Análisis de la arquitectura de referencia OPC-UA 63
10.2 Análisis del protocolo LoRaWAN 64
10.3 Análisis del *Gateway* y Reenviador de Paquetes LoRaWAN 67
10.3.1 Paquete de datos de carga tipo PUSH 68
10.3.2 Paquete de datos de descarga tipo PULL 71
10.4 Descripción de la Arquitectura Propuesta 72

11. DISEÑO DE EXPERIMENTO 79
11.1 Configuración *Gateway* RK831 en ambiente Linux Raspbian 81
11.2 Instalación de servidor LoRa Server en ambiente Linux Raspbian 81
11.3 Programación de Nodos LoRaWAN 85
11.4 Interacción entre paquetes 87
- Iniciar el controlador *Gateway* LoRa 87
- Iniciar Intermediario Pub/Sub (*Broker*) Mosquitto. 87
- Iniciar el Servidor OPC-UA 87
- Iniciar la Extensión OPC-UA Pub. 87
- Iniciar la Extensión OPC-UA Sub. 87
- Iniciar el Servidor LoRaWAN. 87
- Iniciar Clientes OPC-UA. 87
- Activar Nodos LoRaWAN. 88

11.5 Evaluación de la arquitectura propuesta 95
11.5.1 Prueba de carga sobre el módulo Puente LoRaWAN 97
11.5.2 Prueba de carga sobre el módulo Servidor OPC 98
11.5.3 Prueba de carga sobre el módulo MQTT Sub 99
11.5.4 Prueba de carga sobre el módulo Servidor LoRa.IO 100

12. RESULTADOS Y DISCUSIÓN 102

13. CONCLUSIONES 108

14. REFERENCIAS 110

ÍNDICE DE TABLAS

Tabla 1 Tipos de Paquete de Control 41
Tabla 2 Objeto tipo *PUSH_DATA* 69
Tabla 3 Objeto tipo *PUSH_ACK* 69
Tabla 4 Dato JSON del objeto *PUSH_DATA* 69
Tabla 5 Objeto tipo *PULL_DATA* 71
Tabla 6 Objeto tipo *PULL_ACK* 71
Tabla 7 Objeto tipo *PULL_RESP* 71
Tabla 8 Dato JSON del objeto *PULL_RESP* 72
Tabla 9 Resultados de la prueba de carga por módulos 107

ÍNDICE DE FIGURAS

Figura 1 Proyección del crecimiento IoT de acuerdo a Johnson (2016) 13
Figura 2. Marco de referencia de investigación 24
Figura 3 Plan metodológico 27
Figura 4 Byte de Cabecera Fija MQTT, bits de datos. 41
Figura 5 Rango comparativo de las tecnologías LTN vs redes celulares y de bajo alcance. 43
Figura 6 Diferencias tecnológicas de LTN (LPWAN) contra tecnologías rivales. 44
Figura 7 Arquitectura LoRaWAN 45
Figura 8 Componentes de la investigación 62
Figura 9 Clases de abstracción Servidor OPC-UA 64
Figura 10 Trama de datos del protocolo LoRaWAN rev1.0 65
Figura 11 Algoritmo de cifrado LoRaWAN 67
Figura 12 Análisis del Gateway LoRaWAN 68
Figura 13 Casos de Uso 74
Figura 14 Arquitectura propuesta 75
Figura 15 Despliegue de la arquitectura propuesta 77
Figura 16 Dominios de la arquitectura propuesta. 78
Figura 17 LoRa App Server 83
Figura 18 Configuración de Aplicación LoRaWAN 84
Figura 19 Alta de Nodos LoRaWAN 84
Figura 20 Configuración Típica de los Nodos LoRaWAN (*ABP*) 85
Figura 21 Reprogramación de Nodos LoRaWAN 86
Figura 22 Comprobación del funcionamiento Nodo LoRaWAN. 87
Figura 23 Script para automatizar arranque de tareas 88
Figura 24 Extensión Pub/Sub Publicador 89
Figura 25 Extensión Pub/Sub Subscriptor 89
Figura 26 Servidor OPC-UA 90
Figura 27 Cliente OPC-UA 90
Figura 28 Espacio de Direcciones del Sistema LoRaWAN 91
Figura 29 Servicio de Acceso de Dato (DA) en tiempo real 91
Figura 30 Recepción de dato en el Servidor LoRaWAN 92
Figura 31 Recepción de mensajes en el Sistema LoRaWAN 93

Figura 32 Nodo LoRaWAN TTN. 94
Figura 33 Gateway LoRa RAK Wireless. 94
Figura 34 Configuración de Thread Group de JMeter....95
Figura 35 Configuración del retardo en milisegundos. 96
Figura 36 Configuración del objeto UDP Request con dato simulado de la capa física. 96
Figura 37 Visor de resultados de los mensajes simulados por JMeter. 97
Figura 38 Publicación de contador de paquetes del Puente LoRaWAN. 98
Figura 39 Cliente OPC de datos Históricos con un contador de paquetes. 99
Figura 40 Contador de mensajes consumidos desde Mosquitto. 100
Figura 41 Datos LoRaWAN recibidos en el Servidor LoRaWAN. 101
Figura 42 Contador de mensajes del Servidor LoRaWAN. 101

RESUMEN

El constante crecimiento de los dispositivos conectados a Internet de las Cosas requiere de un mecanismo para poder intercomunicar dispositivos y tecnologías heterogéneas a través de métodos estandarizados. En el presente trabajo se realiza una propuesta de arquitectura middleware, tipo gateway, basado en la tecnología OPC-UA Pub/Sub, el cual dota de servicios estandarizados a un espacio de direcciones de un sistema típico LoRaWAN. Como parte de la metodología aplicada se ha desarrollado un prototipo en el cual se han utilizado diversos elementos de hardware y software, como: (i) los Nodos LoRaWAN operados a baterías que incorporan un chip Microchip RN2903 clase A; (ii) un Gateway LoRa basado en un chip Semtech SX1301 con de recepción de paquetes simultáneos alojado en una Raspberry Pi 3B; (iii) el sistema LoRaWAN fue realizado mediante la arquitectura de referencia del proyecto LoRaServer.IO alojado en un servidor local; (iv) tanto el Servidor como el Cliente OPC-UA fueron implementados mediante proyectos de referencia OPC-UA .NET Estándar proporcionados por la Fundación OPC; y (v) el Intermediario Pub/Sub con modelo de comunicación middleware orientado a mensajes fue implementado con el proyecto Mosquitto de la Fundación Eclipse en ambos casos alojados en un servidor local. Bajo estas delimitaciones fue posible la correcta ejecución del sistema LoRaWAN en el espacio de direcciones OPC-UA, el uso de sus datos en cualquier cliente OPC-UA y el consumo, mediante OPC-UA, por el Servidor LoRaWAN, demostrando la factibilidad de la arquitectura. Las posibles aplicaciones de esta arquitectura están enfocadas en la integración de tecnologías *IoT* en la creación de nuevos sistemas *IoT*, *IIoT* y de la Industria 4.0 permitiendo una integración sencilla, robusta y basada en modelos estandarizados.

(**Palabras clave**: Internet de las Cosas, OPC-UA, LoRaWAN, *Gateway*)

ABSTRACT

The rapidly increase of Internet of Things interconnected devices is demanding a mechanism to allow interoperability between heterogeneous technologies by means of standardized methods. This thesis presents a middleware software architecture in form of a gateway device based on the standard OPC-UA Publish-Subscribe (Pub/Sub) to provide a set of standardized services for Internet of Things devices focused on LoRaWAN technology. The used methodology aims the creation of a viable prototype to deliver a proof of concept of the architecture. A prototype was constructed using various packages and technologies as: (i) battery operated LoRaWAN Class A Nodes including Microchip RN2903 integrated circuits; (ii) a physical LoRa gateway containing a Semtech SX1301 chip hosted in a Raspberry PI 3; (iii) a LoRaWAN system build from LoRaServer.io project locally hosted in an Intel architecture; (iv) the OPC-UA implementation from the OPC Foundation provided Net Standard Stack; (v) the Pub/Sub broker implemented by the Mosquitto project hosted in an Intel architecture. The proposed architecture was developed and validated permitting a LoRaWAN system be modeled as part of the Address Space in a OPC-UA Server, which allow the interoperability to any OPC-UA Clients and OPC-UA Pub/Sub Client, while keeping the LoRaWAN application server integrity. This software architecture provides a mechanism for an easy and robust integration of Internet of Things technologies into IoT, IIoT and Industry 4.0 systems.

(**Key words**: Internet of Things, OPC-UA, LoRaWAN, Gateway)

1. INTRODUCCIÓN

El advenimiento del Internet de las Cosas (también conocido como *IoT*, por sus siglas del inglés *Internet of Things*) se ha visto favorecida por el desarrollo de nuevas tecnologías, el abaratamiento de los dispositivos de intercomunicación y el aumento de las actividades de negocio que hacen uso de este hito que se traduce en millones de dispositivos conectado alrededor del mundo, con un ritmo de crecimiento que aumenta cada año y para el año 2018 se estima que el número de dispositivos IoT sobrepasó el número de teléfonos celulares (Johnson, 2016).

Con la ayuda del *IoT* es posible la construcción de ciudades inteligentes, monitoreo y alertas en caso de altos niveles de contaminación, congestionamiento de tráfico, falta de iluminación pública, espacios disponible de estacionamientos, estado del sistema de agua y alcantarillado, prevención de desastres naturales; verificación de la seguridad estructural de puentes y edificios viejos; la transformación de la manufactura para hacerla más inteligente y adaptable a los cambios en el mercado, entre muchas otras aplicaciones (Boman, Taylor, y Ngu, 2014).

Sin embargo, este crecimiento acelerado ha introducido diversos retos como la falta de interoperabilidad y estandarización entre tecnologías heterogéneas y la carencia de algunos sistemas de aplicar capacidad de cómputo localmente. Una solución a esta problemática es el uso de tecnologías basadas de arquitecturas software de capa abstracción, conocidos como *middleware* (Boman et al., 2014). Existen *middleware* cuya función es conectar dispositivos *IoT* con aplicaciones de usuario y son conocidos como *gateway*.

En esta tesis se propone un modelo de arquitectura de software para sistemas *IoT* tipo *gateway* que permita la interoperabilidad de sistemas heterogéneos haciendo uso de una plataforma de estandarización de modelos de información a través de OPC-UA (Comunicaciones de Plataforma Abierta Arquitectura Unificada por sus siglas en inglés *Open Platform Communications*

Unified Architecture) con el uso de un patrón de intercambio de datos tipo Publicador-Subscriptor. De la población de dispositivos IoT se ha elegido a la recientemente publicada tecnología LoRaWAN (Red de Area Ampliada de Alto Alcance o Long Range Wide Area Network por sus siglas en inglés) por ser una tecnología abierta basadas en componentes de alta eficiencia energética, para redes de comunicacación de alto alcance, alta ubiquidad y que sirve como ejemplo de la heterogeneidad que aqueja este campo.

En la realización de esta tesis se ha utilizado una metodología basada en la generación de prototipos, el cual se ha desarrollado haciendo uso de diversos paquetes de software de código libre como LoraServer.IO, OPC-UA .NET Standard y Mosquitto, con licenciamiento adecuados, que permiten la comprobación con fines académicos y potencialmente comerciales.

2. ESTADO DEL ARTE

Es tal la diversidad de los sistemas *IoT* que Ngu et al. (2016) hacen referencia a esta cuestión realizando una analogía entre el estado del arte actual contra el estado de la tecnología de los navegadores web varios años atrás: “el desarrollo de aplicaciones *IoT* necesita de soporte específico por tipo de dispositivos, lo cual es equivalente al escenario cuando se requería utilizar cierto navegador web para poder conectarse a recursos de internet” por lo que los sistemas *middleware* son de vital inmportancia en este campo y permiten resolver ciertas carencias de interoperabilidad.

Un *middleware* es un componente de software diseñado para ser el intermediario entre dispositivos *IoT* y aplicaciones de usuario (Ngu et al., 2016). Existen *middleware* cuya función es conectar dispositivos de campo con aplicaciones de mayor nivel, éstos son conocidos como *gateway*. Éstos son sistemas capaces de trasladar los datos desde una red de comunicación de bajo nivel hasta un sistema de aplicación de internet y son parte esencial para la popularización y adopción de la tecnología permitiendo el uso transparente desde el dispositivo hasta la aplicación tal como lo hacen el Wifi, Bluetooth y redes celulares. Estas tecnologías están basadas en estándares establecidos y su popularidad permite acceder a ellas desde casi cualquier lugar y en cualquier momento, Zachariah et al. (2015). Sin embargo, con los dispositivos *IoT*, especialmente, los que están basados en sistemas embebidos, o sistemas con recursos de hardware limitados, el panorama actual es diferente ya que existe una gran diversidad de protocolos de comunicación y no todos permiten una conectividad transparante y segura a internet; por tal razón, para conectarse a internet estos dispositivos requieren de un *gateway*.

En el área de sistemas de control de procesos una forma de comunicar dispositios heterogéos de forma segura y estandarizada es OPC, tal como lo mencionan Imtiaz y Jasperneite (2013): “OPC funciona de forma abstracta a la tecnología de redes y a las aplicaciones de software, ofreciendo una interfaz

genérica de comunicación". OPC es parte importante en la generación de soluciones que integran diferentes tecnologías en aplicaciones, sobre todo, de automatización industrial. Recientemente, Cupek y Ziebinski (2017) han propuesto un *gateway* utilizando OPC-UA para conectar una red de dispositivos CAN (Red de Área de Controllador por sus siglas en ingles *Controller Area Network*) con sistemas ciber físicos[1] (CPS, por sus siglas en inglés *Ciber Phisicall Systems*) logrando realizar una comunicación entre ambos dominios de forma exitosa. Por otro lado, Astarloa et al. (2016) han desarrollado un *gateway* inteligente enfocado a conectar CPS con aplicaciones de tecnología de la información (TI) mediante sistemas redundantes doble anillo como una solución a necesidades de la Industria 4.0[2]. Por su parte, Oksanen, Linkolehto, y Seilonen (2016) evaluaron el uso de la tecnología OPC-UA como un actor en la estandarización de las redes de comunicación para sistemas *IoT* mediante la conexión de un sistema industrial agrícola basado en una red CAN.

En febrero del 2018 la Fundación OPC publicó la extensión Pub/Sub (Publicación-Subscripción), el cual es el mecanismo que permite la operación de OPC-UA en un modelo no cliente-servidor sino en un modelo orientado a mensajes tipo publicador-subscriptor que cuenta con elementos como configuración, conexiones y grupo de datos (utilizando protocolos orientados a mensajes como AMQP - *Advanced Message Queuing Protocol* por sus siglas en inglés - y MQTT - *Message Queuing Telemetry Transport* por sus siglas en inglés -) permitiendo la generación de aplicaciones más complejas, escalables, y con restricciones de tiempo (baja latencia). La adición de este paradigma en el estándar OPC permite la

[1] Un Sistema Ciber Físico consiste en una colección de dispositivos de cómputo comunicándose entre sí e interactuando con el mundo físico mediante sensores y actuadores en un lazo de retroalimentado (Alur, 2015).

[2] Industria 4.0: Es un concepto nacido en Alemania como referencia a la revolución industrial que genera el uso de sistemas ciber físicos (CPS), sistemas IoT y computación en la nube para la generación de fábricas y manufactura inteligente.

adopción de ambientes basados en *middleware* orientado a mensajes muy adecuado para uso en sistemas *IoT*.

El desarrollo de sistemas *middleware,* que faciliten la integración de capas físicas (sensores y actuadores) con el mundo virtual de soluciones de tecnologías de la información (TI) con una forma transparente y estandarizada, es un reto importante y tema de interés en este campo de conocimiento actualmente (Mineraud, Mazhelis, Su, y Tarkoma, 2016).

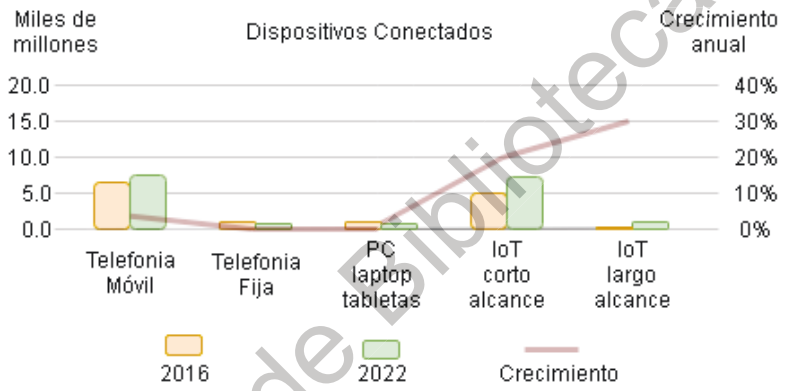

Figura 1 Proyección del crecimiento IoT de acuerdo a Johnson (2016)

De acuerdo al estudio realizado por Ericson, en su reporte de movilidad noviembre 2016, se estima que entre 2016 y 2022 los dispositivos IoT conectados en redes inalámbricas de corto alcance (WiFi, Bluetooth, Zigbee3) crezcan a un ritmo de 20% anual pasando de 5.2 mil millones de dispositivos conectados a 16 mil millones. Mientras que los dispositivos de largo alcance (3GPP IoT4, Sigfox5,

[3] Zigbee es una tecnología basado en la especificación IEEE 802.15.4 para comunicación de alto nivel destinado a redes de área personal (PAN) con bajo consumo de energía, útil para sistemas como automatización de casas inteligentes.

[4] 3GPP o *3rd Generation Partnetship Project* es la asociación que agrupa diversos miembros quienes mantienen el estándar de tecnologías celulares como GSM, UMTS, LTE. En el contexto de IoT en el año 2016 han terminado el estándar conocido como Narrow Band IoT NB-IoT.

[5] Sigfox empresa de origen francés que posee una tecnología de redes inalámbricas basado en ancho de banda ultra estrecho destinado a sistemas LPWAN. Operando en la banda 868 MHZ y 902 MHz.

LoRaWAN6, Ingenu7) se estima lo hagan a un ritmo de 30% crecimiento anual compuesto, pasando de 0.4 mil millones a 2.1 mil millones de dispositivos en el mismo periodo (Johnson, 2016). La proyección se puede observar en la Figura 1 (página anterior).

Considerando el escenario planteado por Jonhson las redes *IoT* de corto alcance han dominado el mercado en los últimos años. Sin embargo, hay aplicaciones donde se requiere una mayor cobertura y en donde aplicar alguna de dichas tecnologías no es práctico o resulta en costes mayores, tanto a nivel dispositivo como en consumo energético; de ahí que han surgido las redes de largo alcance de bajo consumo conocidas como LPWAN (Red de Área Ampliada de Bajo Consumo, por sus siglas en inglés *Low Power Wide Area Network*), a esta tecnología también se le conoce como LTN (Red de Bajo Consumo de Datos, por sus siglas en inglés *Low Throughput Network*). Por ejemplo, LoRaWAN permite la comunicación de dispositivos en rangos considerablemente altos (hasta 40 km en condiciones ideales), con bajo consumo de energía, operados a batería con duración de varios años con una carga (dependiendo de las condiciones de uso) y velocidades de transmisión bajos. Además se trata de una tecnología disponible sin la necesidad de cubrir una cuota por utilizar el servicio, como en el caso de las redes de comunicación celular itinerantes u otras tecnologías *IoT*.

2.1 Retos en sistemas *middleware* IoT

Ngu et al. (2016) hicieron un recuento de los *middleware* que se han venido desarrollando y enumeran diversos retos y problemas abiertos que aún necesitan resolverse. También categoriza los *middleware* de acuerdo a su funcionamiento y forma de aplicación: (i) las arquitecturas orientadas a servicio son aquellas que

[6] LoRaWAN es una tecnología para redes de área ampliada de telecomunicaciones inalámbricas basada en la tecnología LoRa que utiliza una modulación propia. Destinado a sistemas LPWAN. Operando en la banda 868 MHZ y 902 MHz. https://www.lora-alliance.org/.

[7] Ingenu (también conocido como On Ramp Wireless) es una tecnología para redes de área amplia de telecomunicaciones inalámbricas basada en la tecnología de modulación RPMA. Destinado a sistemas M2M (Machine to Machine).

permiten a los desarrolladores o a los usuarios añadir o desplegar diferentes dispositivos *IoT* como si fueran servicios; (ii) el segundo tipo de arquitectura es conocida como basada en la nube, la cual limita a los usuarios en el tipo y número de dispositivos que se pueden desplegar pero facilita la conectividad, recolección e interpretación de datos; (iii) el tercer tipo es conocido de como basado en actores el cual remarca la necesidad de una arquitectura abierta y transparente para dispositivos y aplicaciones finales, el concepto principal de esta arquitectura es flexibilidad de poder colocarse en cualquier nivel dentro del modelo OSI y mantener una latencia y escalabilidad sobresaliente, se trata de arquitecturas de servicio completamente funcionales. De igual manera presentaron sus hallazgos exponiendo diversas problemáticas de los sistemas *middleware* analizados, mismas que se resumen a continuación.

- *Primer reto:* falta de un *middleware* que permita la interacción con datos en la nube así como computación al borde (o edge computing) en donde se pueda realizar ciertos niveles de analítica de forma descentralizada, con el objetivo de mejorar la privacidad y la latencia.

- *Segundo reto*: Dar las herramientas a los consumidores para crear aplicaciones *IoT* en el contexto que necesiten. Con el objetivo de no limitarlos a utilizar sólo dispositivos *IoT* con funcionalidades pre-programadas, sino con características que ellos puedan crear.

- *Tercer reto*: Proveer servicios de búsqueda de semántica, tanto de dispositivos como de nuevos servicios, ya que éstos pueden aparecer en cualquier momento, por lo que de no poseer este servicio, dispositivos podrían dejar de funcionar.

El proyecto *Gatica* es otro ejemplo de *middleware* desarrollado para solucionar el problema de la falta de semántica en cadenas de datos recolectados por sensores que pueden proveer datos útiles en temas relacionados con la salud. Esta arquitectura corre dentro de un *gateway* que permite la inserción de semántica

para poder ejecutar algoritmos de análisis y extracción de datos. Utiliza un sistema operativo compacto denominado BusiBox e implementado con un *framework* llamado Sedona, que permite el envío de peticiones y gestión de todo el ciclo de vida de las conexión de forma compacta, muy adecuado para sistemas embebidos. Su arquitetura ha sido implementada con éxito, permitiendo desplegar el *gateway* en dispositivos de bajo recursos permitiendo la inyección de semántica y el análisis de datos (Qanbari, Behinaein, Rahimzadeh, & Dustdar, 2015).

De igual manera, avances importantes los sigue dando Microsoft con su plataforma para servicios de conectividad *IoT* (Microsoft Azure IoT) en la nube, recientemente ha incluido el soporte de OPC-UA a su ecosistema. Se trata de un Kit de Desarrollo de Software denominado MS Azure IOT OPC-UA, un *gateway* de código abierto que permite la conectividad con su servicio en la nube MS Azure IOT Suite Hub. Esto recalca la importancia de un modelo unificado que permita la integración a gran escala de dispositivos del internet de las cosas en entornos industriales y no industriales. “En este entorno, *IoT*, vemos a OPC-UA como un estándar crítico que asegura la interoperabilidad entre un gran número de equipos y procesos de manufactura conservando décadas de inversión realizadas por muchas empresas” George (2016). Al momento de escribir este documento, Microsoft Azure IoT Gateway SDK se refiere al *gateway* cliente e incluye las librerías con los componentes necesarios para conectar dispositivos de campo hasta una capa a nivel aplicación e infraestructura, multiplataforma, código libre y escrito en ANSI C (C99). Basado en un modelo publicador-suscriptor, soporta protocolos AMQP, MQTT y HTTP (*Hypertext Transfer Protocol* por sus siglas en inglés), persistencia y procesamiento de múltiples dispositivos al mismo tiempo.

3. JUSTIFICACIÓN

De lo anterior, se hace notable la necesidad de encontrar una solución óptima que permita el desarrollo de aplicaciones finales sin importar la tecnología *IoT* base que se emplee y es un tema pendiente fundamental para incursionar en el segmento del Internet de las Cosas Industrial *IIoT* (también conocido como Industria 4.0). Generar una solución TI particular, integrada, a un sistema de manufactura o servicio, para cada protocolo o tecnología *IoT* sería una desventaja considerable comparada con generar una arquitectura que permita la integración de cada tecnología hacia un estándar probado, robusto y factible.

Una de las características más significativas de la tecnología *IoT* es la necesidad de integrar o hacer interoperable sistemas heterogéneos, por ejemplo conectar un sistema de sensores de un sistema de estacionamiento a una red de un edificio de oficinas o a una plataforma para teléfonos inteligentes. Esta necesidad fundamental de los sistemas *IoT* es también un reto debido a la variedad de tecnologías que se pueden utilizar para generar un dispositivo *IoT*, sin mencionar la forma en que cada dispositivo haya sido programado.

En la industria de la automatización de líneas de manufactura el problema de interoperabilidad de sistemas heterogéneos ha sido resuelto mediante el uso de modelos de comunicación estandarizado utilizando OPC como lo han propuesto Imtiaz y Jasperneite (2013), por lo que tratar aportar en este tema en pro de resolver esta problemática mediante una arquitectura *middleware* basado en OPC que pueda contribuir en el esfuerzo de utilizar modelos estandarizados en el campo *IoT* es la principal razón de la realización del presente trabajo.

4. DESCRIPCION DEL PROBLEMA

En los capítulos anteriores se han expuesto diferentes conceptos, escenarios, el estado del arte y temas de estudio relevantes. A continuación se consolidan y plantea la problemática que se pretende resolver en este trabajo de investigación.

La problemática detectada en los sistemas *middleware* para aplicaciones *IoT* se puede exponer de forma resumida de la siguiente manera:

1. Falta de estandarización en los protocolos de comunicación para la interoperabilidad de dispositivos heterogéneos (Mineraud et al., 2016).
2. Falta de servicios para descubrir y re-conectar nodos dentro de las arquitecturas *middleware* (Ngu et al., 2016; Qanbari et al., 2015).
3. Falta o deficiencias de seguridad, confiabilidad y privacidad de datos (Ngu et al., 2016).
4. Problemas con alta latencia ocasionado o asociados a los *middleware* o derivados de la tecnología en particular (Oksanen et al., 2016).
5. Problemas de escalabilidad al saturar la capacidad del *middleware* o aquellos asociados a tecnología en particular (Ngu et al., 2016).
6. Dificultad para integrar los datos obtenidos en aplicaciones finales (Mineraud et al., 2016).
7. Falta de capacidad de cumplir el paradigma de computación al borde o perímetro (*edge computing*) (Mineraud et al., 2016).

Por lo tanto, bajo el contexto en el que se ha venido desarrollando esta exposición, se considera que la **problemática general** es la siguiente:

La falta de una arquitectura middleware tipo gateway que permita la integración de protocolos de Internet de las Cosas con aplicaciones establecidas para facilitar la interoperabilidad entre tecnologías de forma estandarizada.

Al ser ésta una problemática muy general es necesario acotar el alcance de este trabajo y realizar algunas premisas sustentadas con la información expuesta hasta el momento:

La tecnología LoRaWAN es una tecnología emergente utilizada en el Internet de las Cosas.

El estándar OPC-UA permite la interoperabilidad entre sistemas heterogéneos que ha ayudado en el campo del control de procesos y automatización industrial.

Considerando estas dos premisas se puede avanzar en el planteamiento del problema delimitado a sistemas específicos y que la problemática particula expone la necesidad de contar con un *gateway middleware* para LoRaWAN que pueda interoperar sistemas heterogéneos mediante OPC-UA.

Teniendo como **problemas específicos** los siguientes:

- Se desconoce el modelo de arquitectura *gateway* de aplicación que sería más eficiente para poder trabajar con tecnologías heterogéneas que permita la interoperabilidad con sistemas basados en tecnología existentes para *IoT*.
- Se desconoce si las tecnologías LoRaWAN y OPC-UA pueden interoperar de forma eficiente en una aplicación CPS e *IoT*.
- Se desconoce las características de hardware y el sistema operativo necesario para ejecutar adecuadamente la arquitectura *gateway* propuesta.

5. HIPÓTESIS

Se ha planteado la necesidad por una arquitectura *middleware,* o *gateway* de aplicación, que permita la interoperabilidad de tecnologías heterogéneas bajo un modelo estandarizado y dentro de un ambiente *IoT*, por lo tanto se propone realizar una arquitectura de software que satisfaga tales demandas utilizando un modelo de comunicación estandarizado OPC-UA. Existen algunos antecedentes, como el *gateway* propuesto por Cupek y Ziebinski (2017), sin embargo no involucran el acceso de tecnologías *IoT* ni un modelo Pub/Sub. Para cumplir con este planteamiento es necesario fijar un escenario que pueda servir como referencia y permita acotar la investigación y poder fijar metas, objetivos y plazos de entrega.

5.1 Delimitación y alcances

El escenario de este trabajo contempla el uso de la tecnología *IoT* de radio frecuencia de largo alcance, bajo consumo de energía, sin reestricciones de tiempo y de acceso libre tipo LoRaWAN operando en un sistema típico mediante un gateway LoRa (capa física) y un Servidor LoRaWAN. Además, se considera al estándar OPC-UA como la tecnología proveniente de los CPS que permiten la interoperabilidad de sistemas heterogéneos en sistemas de control distribuidos y cuya extensión Pub/Sub permiten su uso en aplicaciones altamente escalables.

5.2 La Hipótesis

A continuación se define la hipótesis de esta tesis:

Es posible implementar un modelo de información OPC-UA y desarrollar una arquitectura de software que permita el traslado de dominio desde LoRaWAN hasta OPC-UA para interoperar la información de los Nodos LoRaWAN con cualquier cliente OPC-UA.

Realizar esto conlleva resolver la siguiente problemática: (i) aumentar la interoperabilidad, mediante la disponibilidad de la información de los Nodos LoRaWAN en el espacio de direcciones de OPC-UA lo cual ayudaría,

potencialmente, su acceso de forma estandariza por cualquier cliente OPC-UA; (ii) posibilitar la computación al borde ya que el Servidor OPC-UA, anfitrión de la información de los Nodos LoRaWAN, puede a su vez ser un cliente que realice acciones o genere meta-datos sobre esta información; (iii) tener acceso a servicios para descubrir y reconectar nodos ya que el Servidor OPC-UA cuenta con éstos; (iv) aumentar las medidas de seguridad y autenticación ya que el Servidor OPC-UA soportar mecanismos de autenticación y encriptación; y (v) aumentar la escalibilidad del sistema ya que OPC-UA permite operar con el modelo Pub/Sub.

5.3 Preguntas de investigación

Para poder cumplir con cada uno de los requisitos, las actividades establecidas y la generación de un prototipo que permita su evaluación se deberán resolver las siguientes interrogantes:

- ¿Qué es la tecnología OPC-UA Pub Sub y cuáles son sus características?
- ¿Cuál es Campo de Dominio/ Modelo de Información OPC-UA Pub-Sub que mejor describe la capa física de los dispositivos que serán conectados?
- ¿Cómo funciona la tecnología LoRaWAN y cuáles son sus características?
- ¿Cuál es el modelo, patrón o arquitectura que permita crear una solución lo suficientemente flexible, extensible y adaptable a cualquier modelo de información proveniente de los sensores y actuadores?
- ¿Qué características de hardware se requiere y cuáles son sus restricciones para poder alojar la solución propuesta?

6. OBJETIVOS

6.1 Objetivo General:

Proponer y evaluar una arquitectura de software que permita interoperar dispositivos LoRaWAN y sistemas ciber físicos OPC-UA Pub Sub.

6.2 Objetivos Particulares:

Identificar las características del estándar OPC-UA Pub-Sub como middleware orientado a mensajes en un ambiente de sistemas ciber físicos.

Describir la tecnología LoRaWAN y sus componentes que permitan adaptar sus tramas de mensajes a un modelo con semántica agregada.

Diseñar la arquitectura de software que permita la creación de un middleware de aplicación tipo Gateway que incorpore la detección de señales LoRaWAN y permita su uso con tecnología OPC-UA.

Diseñar los componentes de software que permitan el desarrollo del prototipo del objetivo anterior.

Realizar un prototipo que permita evaluar la arquitectura propuesta.

Evaluar la funcionalidad del sistema teniendo en cuenta los siguientes puntos: (i) funcionalidad correcta del sistema y (ii) robustez del modelo Pub/Sub implementado mediante pruebas de carga en las diversas capas de software del modelo.

7. TRABAJOS RELACIONADOS

Forsstrom y Jennehag (2017) han publicado los resultados de la evaluación de la factibilidad de uso de OPC-UA en aplicaciones *IoT* Industriales considerando sensores, conectadas bajo el paradigma de WSN (Redes de Sensores Inalámbricos, por sus siglas en inglés *Wireless Sensor Networks*). El trabajo que realizaron estuvo centrado en la evaluación de la tecnología OPC-UA en un entorno empresarial mediante Microsoft Azure IoT Hub un servicio provisto por Microsoft para soluciones *IoT* y cómputo en la Nube. En su reporte hacen un recuento del tiempo de respuesta y el costo estimado para una cantidad de sensores. Realizaron estas pruebas utilizando sensores simulados, sin embargo, no describen ninguna implementación (el *middleware* o modelos de abstracción asociados) para adaptar la Red de Sensores Inalámbricos a la plataforma usando OPC-UA.

Otro trabajo relacionado lo realizaron Cupek y Ziebinski (2017) en el que propusieron una arquitectura de software que permite compartir datos de protocolo de campo de una red CAN a una red de comunicaciones en un ambiente de Sistemas Ciber Físicos a través del uso de OPC-UA. La selección de la tecnología OPC-UA se debió a la gran estandarización que permite generar soluciones que pueden ser rápidamente integradas en dispositivos heterogéneos. Uno de los mayores retos presentados en este documento de investigación fue la asociación de semántica de los paquetes de datos intercambiados en los mensajes de CAN, esta labor fue resuelta mediante la incorporación de un puente que permite el uso de la información existente en la base de datos de CAN y su interpretación en el espacio de direcciones del servidor OPC-UA. Con esta arquitectura es posible la comunicación de redes de campo CAN con Sistemas Ciber Físico mediante OPC-UA.

8. METODOLOGÍA

8.1 Metodología

Se utilizará la metodología de investigación propuesta por Hevner, Park, Sudha y March(2004) en donde se parte del concepto de investigación teórica enfocada a la resolución e implementación de artefactos que resuelvan una necesidad relevante al entorno a través de un proceso de investigación como se observa en la Figura 2 y cuyos elementos se describen a continuación:

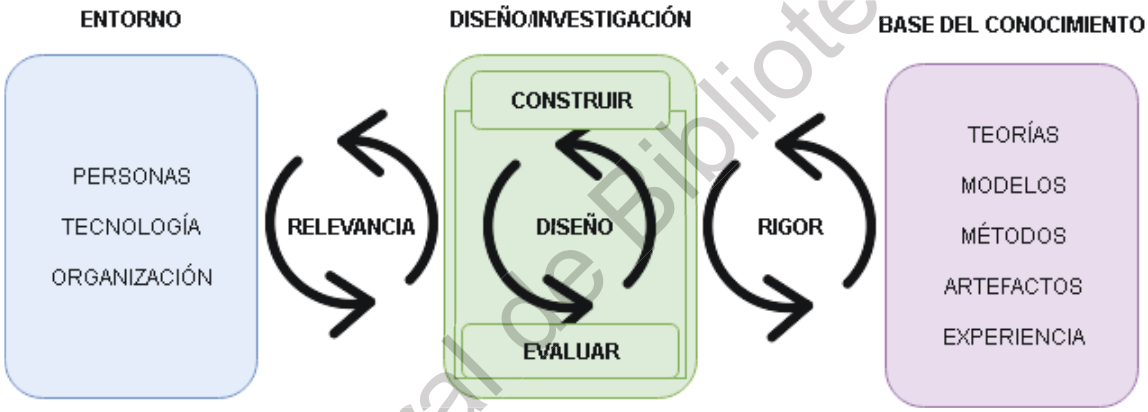

Figura 2. Marco de referencia de investigación

1) Relevancia del problema. A lo largo de este documento se ha mostrado la relevancia económica de las tecnologías investigadas. Los problemas citados son parte de investigaciones de relevancia actuales lo cual demuestra la escala del tema. Se estima que para el 2022 los dispositivos LTN crezcan a una tasa de 30% llegando a un total de 2.1 mil millones de dispositivos conectados.

2) Contribuciones de la investigación. Se propone realizar aportaciones en la generación de una arquitectura de software que incluya componentes necesarios para resolver la problemática planteada.

   a) Definir el modelo de Información OPC-UA para el manejo de LoRaWAN.

b) Probar el anexo OPC-UA Pub-Sub con la tecnología LoRaWAN.

c) Proponer una arquitectura middleware que permita la comprobación de la hipótesis.

d) Evaluar la arquitectura propuesta realizando un prototipo de aplicación.

3) Investigación rigurosa. Se realizará una investigación formal siguiendo el método científico que involucra la búsqueda de artículos científicos, de divulgación y la lectura de estándares que permitan la generación de nuevo conocimiento.

a) Siguiendo y profundizando en los antecedentes mencionados por Cupek y Astarloa y sus colegas (Astarloa et al., 2016; Cupek & Ziebinski, 2017).

b) Analizando el anexo del estándar OPC-UA Pub-Sub y del estandar LoRaWAN 1.0.

4) Diseñar como un artefacto. El objetivo es generar un prototipo funcional que incluya la adopción del estándar OPC-UA Pub Sub e involucra el desarrollo e integración de productos de trabajo en las áreas de software y hardware.

a) Se ha propuesto como objetivo la evaluación de la arquitectura de aplicación propuesta.

b) Se desarrollará un prototipo funcional que permita cumplir con el punto anterior.

5) Diseñar como un proceso de búsqueda. Se trata de investigar y mejorar el conocimiento adquirido mediante el refinamiento e iteraciones a lo largo del proceso planteado.

a) Investigado a diversos autores y trabajos académicos y de investigación. Se proseguirá teniendo en cuenta los resultados actuales y los futuros para tener un panorama y mayor conocimiento al respecto de este tema,

adoptando cualquier hallazgo que se considere relevante para esta investigación.

6) Evaluación del diseño. Se evaluará formalmente considerando la interoperabilidad entre LoRaWAN y OPC-UA.

   a) Se evaluará la funcionalidad de la arquitectura.

   b) Se reportarán las métricas de software de los diversos paquetes de software involucrados.

7) Comunicación de la investigación. Se publicará la tesis para la obtención de grado teniendo en cuenta una audiencia científica e industrial. Además, se tiene como objetivo publicar un artículo de divulgación en congreso académico.

Se ha propuesto el uso del marco de referencia bajo un proceso de diseño de software iterativo incremental típico cuyo diagrama de flujo se puede observa en la Figura 3. El objetivo es refinar el artefacto final mediante la sinergia del entorno, la investigación, el diseño y la base del conocimiento en procesos iterativos.

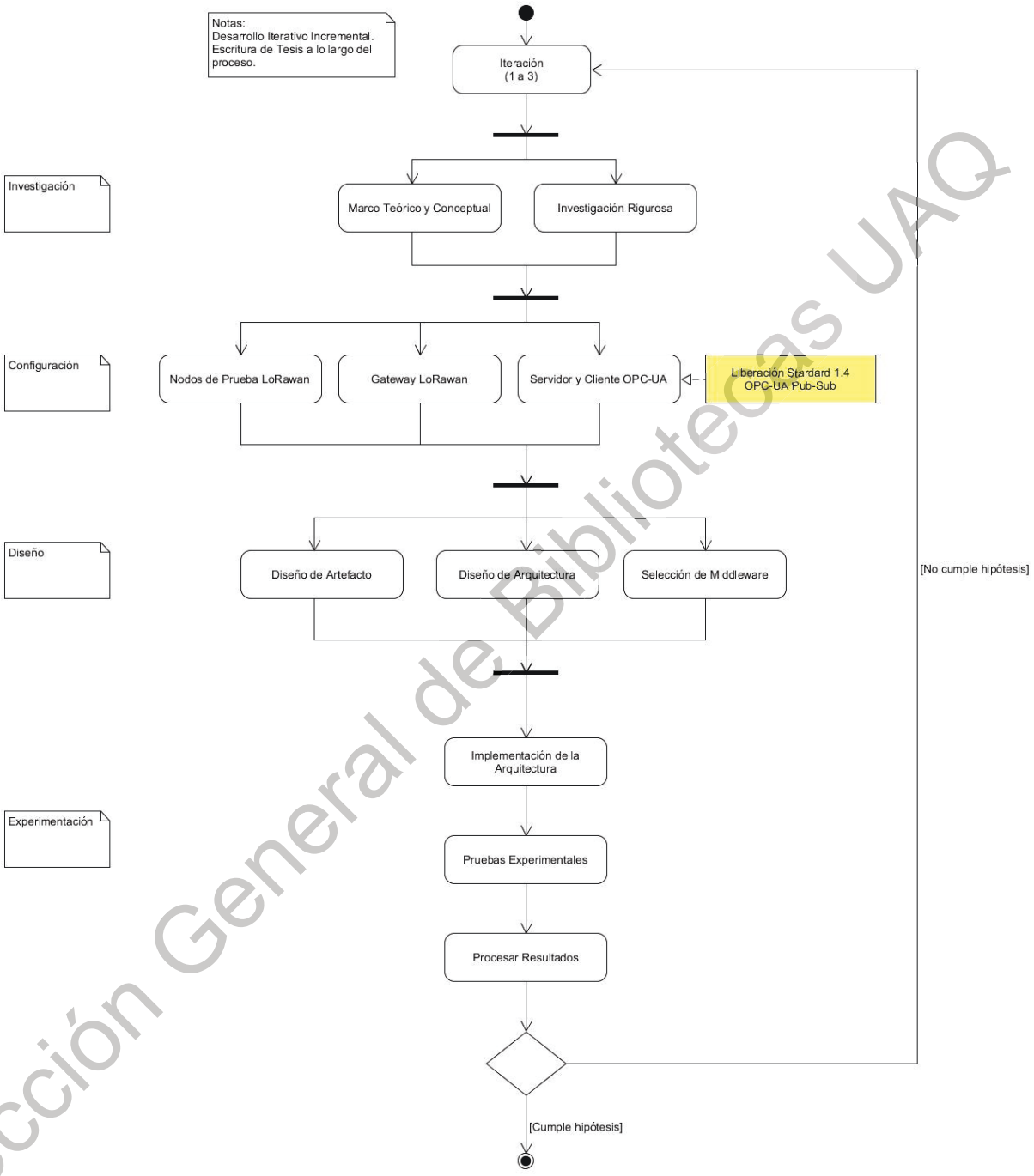

Figura 3 Plan metodológico

9. MARCO TEÓRICO

9.1 Internet de las Cosas (*IoT*)

IoT (Internet de las Cosas o *Internet of Things* por sus siglas en inglés) se refiere al grupo de dispositivos conectados a internet que generan o consumen información y no requieren, necesariamente, de la intervención del ser humano para su funcionamiento. Estos dispositivos pueden ser equipos de cómputo, teléfonos inteligentes, pulseras inteligentes, sensores, actuadores, aparatos industriales, entre muchos otros. La IEEE en la publicación destinada a proponer una definición para el término *IoT* escrita por Minerva, Biru, y Rotondi (2015) exponen las razones que han causado ambigüedad en las diferentes definiciones de *IoT* y también proporciona algunas pautas a seguir en la búsqueda de acuñar una definición neutral enfatizando todos los componentes que un sistema *IoT* puede incorporar. De esta manera, definen *IoT* como: "Una red de Cosas (dispositivos, nodos) únicamente identificable conectadas a internet. Las Cosas tienen capacidad de sensar, actuar, y potencialmente de ser reprogramados. A través del uso de la identifiación única y sensórica, la información obtenida puede ser recolectada, mientras que su estado puede ser cambiado, en ambos casos, desde cualquier lugar, a cualquier hora y por cualquier medio".

IoT tiene su origen en la industria de la tecnología RFID (Identificación por Radio Frecuencia, *Radio Frecuency Identification* por sus siglas en inglés*),* cuyos antecedentes se remontan a la segunda guerra mundial mediante el uso de radares para identificación de aviones. Los trabajos de investigación y desarrollo continuaron y fue en 1973 cuando Mario W. Cardullo y Charles Walton, cada uno, obtuvieron patentes relacionadas con *trasponders* (tags o etiquetas RFID). En 1999 Uniform Code Council, EAN International, Procter and Gamble y Gillete reunieron los fondos para crear el Auto-ID Center en el Instituto Tecnológico de Massachusetts (MIT) con la idea de crear sistemas RFID lo suficientemente viables tecnológica y económicamente para poder ser utilizados en cadenas de suministro. Fue en el Auto-ID Center entre los años 1999-2000 donde Kevin Ashton asegura haber

acuñado el término Internet de las Cosas, sin embargo de acuerdo al profesor Daniel Engels, dicho término había sido usado un una publicación de la Unión Internacional de Telecomunicaciones en 1997 (Minerva et al., 2015).

Existen términos que se han derivado del *IoT* y que describen y especifican el campo o sector donde es utilizado, un ejemplo es el término Internet de las Cosas Industrial (*IIoT* por su acrónimo en inglés *Industrial Internet of Things)* el cual se centra en el escenario industrial de la implementación de sistemas *IoT*. Otro ejemplo es *MIoT* (*Multimedia Internet of Things*) enfocado a los sistemas *IoT* con capacidades y aplicaciones multimedia.

9.2 Sistemas Distribuidos

Un sistema distribuido se define como “aquel sistema en el que los componentes de hardware o software se encuentran localizados en computadoras unidas mediante red y comunican y coordinan sus acciones sólo mediante el paso de mensajes” (Coulouris, Dollimore, y Kindberg, 2001, p. 2).

Uno de los principales objetivos para conectar equipos en un sistema distribuido es la necesidad de compartir recursos como: impresoras, servidores de base de datos, hardware o software especializados o cualquier otro recurso que sea interés en la aplicación (Coulouris et al., 2001). Los sistemas distribuidos poseen ciertas características las cuales se describen a continuación:

Recursos compartidos. Uno de los principales objetivos de los sistemas distribuidos es la capacidad de poder compartir recursos mediante la identificación y la coordinación de éstos a lo largo del entorno distribuidos en el que se encuentran (Domínguez, 2005, p. 56).

Apertura. Un sistema distribuido abierto es un sistema que ofrece servicios de acuerdo a las reglas estándar que describe la sintaxis y semántica de estos. De esta manera se obtienen sistemas concurrentes, independientes extensibles y que

permite a componentes heterogéneos conectarse o desconectarse del sistema (Domínguez, 2005, p. 56).

Heterogeneidad. Es decir la variedad y diversidad de un sistema distribuido tanto de hardware, protocolos de red, sistemas operativos, lenguajes de programación y aplicativos de diferentes proveedores. Para poder interoperar por lo regular es necesario transformar formatos o protocolos de un sistema dado a formatos o protocolos estandarizados que permitan una comunicación escalable. De esta necesidad surgen los sistemas *middleware* o capas de comunicación abstracta que permite a un programa comunicarse con otro mediante llamadas a procedimiento, métodos o peticiones, independientemente del sistema operativo, lenguaje de programación o hardware de los agentes involucrados (Domínguez, 2005, p. 57). El término heterogeneidad se aplica a todos los elementos que a continuación de enumeran (Coulouris et al., 2001, p. 15):

- Redes.
- Hardware de computadoras.
- Sistemas operativos.
- Lenguajes de programación.
- Aplicativos de diferentes desarrolladores.

Interoperabilidad. Es la característica por la cual dos sistemas o componentes de diferentes fabricantes pueden coexistir y trabajar de forma coordinada valiéndose de los servicios que cada uno ha implementado siguiendo un estándar o especificación en común (Domínguez, 2005, p. 57).

Concurrencia. Se menciona que los sistemas distribuidos deben ser capaz de gestionar el intento de acceder a un mismo recurso al mismo tiempo por parte de uno varios usuarios sin que esta situación produzca una condición de error en el sistema (Domínguez, 2005, p. 57).

Portabilidad. Es la característica que posee un sistema distribuido por la cual puede ser ejecutado en un equipo A y un equipo B diferente, sin la necesidad de realizar ninguna modificación (Domínguez, 2005, p. 57).

Flexibilidad. Es una característica que se deberá tener en cuenta al momento de diseñar un sistema distribuido ya que éste deberá ser los suficientemente moldeable para poder incorporar nuevas características o modificar algunas existentes de forma sencilla (Domínguez, 2005, p. 57).

Escalabilidad o extensibilidad. Se dice que un equipo es escalable si conserva e incrementa su efectividad cuando hay un aumento en el número de recursos y de usuarios conectados al sistema distribuido. Un sistema escalable deberá soportar dicho aumento de usuarios y recursos sin presentar pérdida de prestaciones, rendimiento o aumento de cuello de botella (Domínguez, 2005, p. 58).

Tolerancia a fallos. Cualquier equipo es susceptible a generar o verse afectado por una falla por lo que los diseñadores del sistema deberán prever este tipo de situaciones (Coulouris et al., 2001, p. 2). Domínguez menciona que los sistemas distribuidos son más confiables que los centralizados porque se dice que sus componentes son parciales, por lo que algunos componentes pueden fallar mientras los otros siguen funcionando. De igual manera indica algunas técnicas a tener presentes para el tratamiento de fallos (Domínguez, 2005, p. 58):

- Detección de fallos.
- Enmascaramiento de fallos.
- Tolerancia de fallos.
- Recuperación frente a fallos.
- Redundancia.

Disponibilidad. Un buen sistema distribuido tiene la capacidad de estar disponible en el momento que se necesita. Es una propiedad recíproca a la tolerancia de fallos, ya que si un sistema no falla o falla muy poco entonces estará disponible el mayor tiempo posible (Domínguez, 2005, p. 58).

Transparencia. Es una propiedad muy importante, ya que un buen sistema distribuido deberá ser capaz de ocultar de la mejor forma , tanto a los usuarios como a los programadores, las separaciones entre los componentes o recursos a los que se accede y funcionar como una sola unidad, la transparencia de puede sub-clasificar de la siguiente forma: (Domínguez, 2005, pp. 58-59).

- Transparencia de acceso.
- Transparencia de ubicación/localización.
- Transparencia de concurrencia.
- Transparencia de replicación.
- Transparencia frente a fallos.
- Transparencia de movilidad.
- Transparencia de rendimiento.
- Transparencia de escalado.
- Transparencia de persistencia.
- Transparencia de migración.

Inexistencia de un reloj global. La cooperación entre componentes de un sistema distribuido permite el intercambio de mensajes, sin embargo la naturaleza de separación inherente entre componentes no permite una sincronización precisa por lo que no existe una noción de un reloj global, por lo que la comunicación se

realiza por mensajes que están coordinados entre los participantes y no por un reloj de sincronización (Coulouris et al., 2001, p. 2).

Como parte de los sistemas distribuidos se hace notable la necesidad de definir algunos términos que son frecuentemente utilizados al hacer referencia a las diferentes estructuras de comunicación y recursos compartidos:

Servicios. Se utiliza este término para referirse a la parte de un componente de un sistema distribuido que gestiona una colección de recursos relacionados y presenta su funcionalidad a los usuarios y aplicaciones. Un servicio ofrece un conjunto de operaciones con las cuales se puede interactuar con los recursos dados y sólo a través de éstas (Coulouris et al., 2001, p. 8).

Modelo Cliente - Servidor. El término Servidor se refiere a un proceso o programa en ejecución en una computadora conectada en red que acepta peticiones desde otro programa que se ejecuta en otra computadora con el objetivo de realizar un servicio y responder de forma adecuada. Por otra parte, los procesos que provienen del solicitante son llamados Clientes. En una operación dada un componente puede tener un rol de cliente o servidor, por lo que la figura cliente-servidor es válida por la operación en específico (Coulouris et al., 2001, p. 8).

Interfaces. Por lo general en los sistemas distribuidos los servicios se encuentran definidos a través de interfaces los cuales se suelen definir en un lenguaje de definición de interfaz. Las cuales sólo suelen capturar la sintaxis de los servicios, esto es, el nombre de la función, los parámetros, valores de retorno y posibles excepciones (Tanenbaum y Van-Steen, 2008, p. 8).

9.3 Computación móvil y ubicua

Se le llama computación móvil o nómada a la realización de tareas de cómputo al tiempo que el usuario se encuentra en movimiento o cambiando de ubicación. Este tipo de tecnología se encuentra muy extendida por el uso de los teléfonos inteligentes.

Por su parte, la computación ubicua es la utilización concertada de una gran variedad de dispositivos de computación, por lo regular pequeños, de bajo costo y bajos recursos presentes en el entorno físico de los usuarios, ejemplo de ellos son los dispositivos de consumo conectados a internet como lavadoras y otros dispositivos (Coulouris et al., 2001, p. 6).

9.4 Sistemas distribuidos masivos y redes de monitoreo

Los sistemas distribuidos masivos se caracterizan por ser dispositivos pequeños, de baterías, portátiles y tienen una conexión a la red (por lo general inalámbrica). Otra característica es la carencia del control humano administrativo sobre el dispositivo. Las redes de monitoreo, una forma de sistema distribuido masivo, son utilizadas por lo general para obtener información. Por ejemplo, una red de sensores típica está constituida por decenas o miles de nodos relativamente pequeños y cada nodo está equipado con sensores, la mayoría de las redes de monitoreo utiliza la comunicación inalámbrica y los nodos son generalmente operados a batería, por lo que demandan un alto grado de eficiencia en las comunicaciones. Una red de monitoreo puede verse como una base de datos distribuida (Tanenbaum y Van-Steen, 2008, pp. 24-30).

9.5 Middleware

El término *middleware* se refiere a la capa de software que provee una abstracción de programación y el enmascaramiento de los componentes de más bajo nivel sobre el cual se encuentra construida una aplicación, incluyendo hardware, sistemas operativos y lenguajes de programación. Además, provee un modelo o arquitectura computacional disponible para los programadores de servicios y aplicaciones distribuidas. Puede incorporar elementos de abstracción como: procedimientos de invocación remota, comunicación dentro grupos de procesos, notificación de eventos, replicación de datos compartidos y transmisión de datos multimedia en tiempo real.(Coulouris et al., 2001, pp. 16,30).

Otra definición de un *middleware* es: aquella capa de software colocado de manera lógica entre una capa de alto nivel, que puede incluir usuarios y aplicaciones, y una capa de inferior construida por sistemas operativos y recursos básicos de comunicación (Tanenbaum y Van-Steen, 2008, p. 3).

Domínguez realiza una notable definición de un sistema *middleware:* "Para conseguir esta heterogeneidad (una de las características de los sistemas distribuidos), es necesario cumplir alguno de los siguientes requisitos: transformar los formatos o representaciones de datos o protocolos de un sistema a otro o cumplir con formatos o protocolos estandarizados que permitan la comunicación. Así por ejemplo, Internet permite heterogeneidad a nivel de sistemas operativos, redes y hardware debido a que está basado en un protocolo de comunicación estandarizado y ampliamente aceptado como es TCP/IP. De tal forma que todo dispositivo conectado a Internet, tiene en común que se comunica utilizando el protocolo TCP/IP. Sobre este protocolo hay un conjunto de servicios soportados por otros protocolos que permiten realizar un gran número de tareas como son HTTP para la Web, FTP para transferencia de archivos, TELNET para terminales remotos, H323 para video conferencias, POP3, MAPI y SMTP para correo electrónico, etc. Estos protocolos están ampliamente aceptados y estandarizados. Todos estos protocolos han sido ideados para una comunicación humano/computador y computador/humano, si un programa necesita comunicarse con otro debe utilizar una capa, protocolo intermedio proporcionado por lo que se denominan *middleware*. Un *middleware* es una abstracción de comunicación a nivel de lenguaje de programación de tal forma que un programa puede comunicarse con otro, realizando llamadas a procedimientos, funciones, métodos de otro a través de él, independientemente del sistema operativo, del lenguaje de programación destino, del hardware y de los protocolos de red subyacentes. Existen varias plataformas *middleware* que pretenden solucionar el problema de la abstracción de la programación en los sistemas abiertos distribuidos, como son: RPC, CORBA, DCOM, RMI, SOAP. A diferencia de Internet, que se a estandarizado su protocolo

de comunicación TCP/IP, a nivel *middleware* no hay aún un consenso y nos ha surgido una torre de babel a nivel *middleware*, es decir hay muchos sistemas implementados en diferentes plataformas que no pueden interoperar entre sí. Realmente existen un conjunto de especificaciones y sistemas desarrollados en el mercado denominados puentes – o *gateways* como es denominado en este documento - que convierten a los sistemas abiertos distribuidos con *middleware* en un conjunto de islas conectadas con estos puentes."(Domínguez, 2005, p. 60).

En la práctica es usual referirse a los sistemas *middleware* de acuerdo al modelo o estilo arquitectónico de software que siguen. La arquitectura de software describe abstractamente cómo se encuentran organizados los componentes de software y la forma en la que se relacionan. La implementación real de la arquitectura de software se conoce como arquitectura de sistema o aplicación. Una arquitectura de software se define con base en ciertos conceptos base: los componentes que son las unidades modulares definidas junto con sus interfaces, teniendo en cuenta que los componentes puedan ser redefinidos o reemplazos siempre que mantengan la interfaz compatible con lo cual no debería afectar el comportamiento general del sistema; y los conectores, que son aquellos mecanismos que permiten la interacción en diversos componentes (Tanenbaum y Van-Steen, 2008, pp. 33-34). Los estilos arquitectónicos más relevantes a los sistemas distribuidos, los cuales se caracterizan por la forma en que sus componentes y conectores se configuran son:

Arquitectura de capas. Esta arquitectura es muy sencilla de entender ya que utiliza diferentes capas o niveles (L). De tal forma que cierta capa L_n puede comunicarse con la capa L_{n-1} pero no con las otras capas. Por lo que las peticiones fluyen de una capa a otra, por lo regular generando peticiones de una capa superior a una inferior y recibiendo las respuesta de forma invertida.

Arquitectura basada en objetos. Es un estilo arquitectónico en el cual el arreglo o disposición de los elementos u objetos, en su papel de componentes, se

encuentran conectados de cierta forma mediante conectores para peticiones remotas.

Arquitectura centrada en datos**.** Surgieron bajo la idea de que una parte de los procesos distribuidos son gestionados en base al intercambio de datos, centrados bajo un repositorio común.

Arquitectura basada en eventos. Es un modelo arquitectónico que se basa en un mecanismo en el cual sus componentes se comunican entre sí mediante la propagación de eventos, los cuales a su vez, pueden transportar datos. Un ejemplo de este tipo de arquitectura es el sistema de publicación-suscripción. Una ventaja de este tipo de sistemas es que los procesos se asocian libremente (referencialmente acoplados).

9.5.1 *Middleware* orientado a mensajes (MOM)

Un *middleware* orientado a mensaje (MOM) es una clase importante y creciente de los sistemas distribuidos y tiene las características de no requerir de sincronía en la comunicación. Fundamentado en un estilo arquitectónico basado en eventos; funciona cuando las aplicaciones envían mensajes a punto lógicos de contacto, los cuales por lo general están descritos por un sujeto (publicando), otras aplicaciones pueden estar interesadas por algún cierto tipo de mensajes (subscritas). El *middleware* se encarga que los mensajes lleguen a sus destinos sin importar si la aplicación subscriptora está activa en el justo instante cuando el publicador emitió el mensaje o no (Tanenbaum y Van-Steen, 2008).

Dependiendo de la funcionalidad requerida es posible agrupar los *middleware* MOM en dos grupos distintos, aunque en aplicaciones prácticas se suelen combinar ambos modelos con el objetivo de satisfacer el requerimiento de la aplicación en cuestión:

Modelo Punto a Punto: Se trata de un modelo MOM en el cual el mensaje enviado por el emisor es entregado al consumidor. Esto es, el mensaje se origina y

se consume por una sola entidad, respectivamente. Este modelo permite controlar y saber cómo fluye la información.

Modelo Publicador y Suscriptor (Pub-Sub): Se trata de un modelo MOM muy poderoso el cual permite la comunicación de "uno a muchos" y de "muchos a muchos", a diferencia del modelo Punto a Punto el cual se centra en "uno a uno". La entidad que publica se encarga de suministrar mensajes, mientras, que la entidad subscrita se encarga de consumir los mensajes a los cuales está subscrito. La máquina Pub-Sub intermediaria se encarga de recibir y de repartir los mensajes de acuerdo al rol de cada entidad. Una entidad puede estar subscrita y ser un publicador al mismo tiempo. Existen dos formas básicas en las que un modelo Pub-Sub puede consumir los mensajes: (a) mediante el mecanismo *Pull* en donde la entidad consumidora constantemente pregunta por una respuesta; y (b) mediante el mecanismo *Push* en donde el consumidor se registra en la máquina Pub-Sub la cual al recibir el mensaje la reenvía al consumidor. Este modelo no puede conocer de antemano quienes recibirán la información ya que cualquier suscrito a un mensaje puede escucharlo.

Un *middleware* MOM define un modelo de comunicación por herencias o agrupados en ramas. De tal forma que debe existir un modelo bien definido y conocido por las entidades participantes, en donde existe un modelo vertical descendente comenzando por conceptos generales, divididos en categorías consecutivamente de acuerdo al modelo de información. De esta manera, una entidad puede estar subscrito al nivel de información al que necesite acceder.

Los *middleware* MOM pueden incluir las *transacciones de mensajes* que se refieren a la situación en donde una entidad necesita transmitir una serie de mensajes. Este mecanismo funciona de la siguiente manera: el publicador de mensaje envía la serie de mensajes y al final confirma la petición (*commit* para aceptar o *rollback* para descartar) lo cual permite que la cola de mensajes se vuelva utilizable o se elimine de la misma.

9.6 Protocolos Orientados a Mensajes

Los sistemas *middleware* orientado a mensajes dependen de protocolos de comunicación que han sido desarrollados pensando en satificer las necesidades de la comunicación asíncrona pudiendo seguir distintos modelos de mensajería como: los modelos punto a punto y publicador – subscriptor. Se requiere de un intermediario que se encargue del manejo del intercambio de las peticiones, y el intermediario es por lo regular un manejador de colas de mensajes (Mahmoud, 2004).

9.6.1 Cola de Mensajes

Las colas de mensajes son una parte primordial de los *middleware* MOM ya que proveen de capacidades de almacenamiento temporal de mensajes provenientes de los productores de mensajes para ser enrutados hacia los consumidores. El tipo de cola de mensajes utilizados en los sistemas asíncronos middleware MOM son las tipo *FIFO* (Primero en Entrar Primero en Salir, *First In First Out* por sus siglas en inglés) que permiten que el primer mensaje recibido sea el primer mensaje en salir (Curry, 2004).

9.6.2 Modelo Punto a Punto

Este modelo comunicación permite que los mensajes publicados hacia una cola de mensajes sean entregados una sóla vez a un solo consumidor, si éste no se encuentra disponible al momento de su publicación entonces es almacenado hasta que sea entregado.

9.6.3 Modelo Publicador – Subscriptor

El modelo publicador – subscriptor extiende las capacidades del modelo punto a punto ya que permite la diseminación de mensajes de forma anónima siguiendo el patrón de uno-a-muchos o de muchos-a-muchos el cual permite que un productor de mensaje publique información hacia muchos consumidores. El productor no conoce la identidad ni el número de consumidores subscritos. En este modelo tanto

el publicador como el subscriptor realizan sus transacciones con respecto a un *tópico* o *canal* que sirve como referencia o identificador para ambas entidades, es resposabilidad del manejador de colas identificar el tópico al cual enrutar de acuerdo al perfil del publicador-subscriptor disponible. El tópico pueder ser estructurado de forma jerárquica para permitir distintos niveles de granularidad.

La confiabilidad de las transacciones llevadas acabo por los protocolos orientados a mensajes contienen un conjunto mínimo de métricas y parámetros de configuración que permiten ajustar aquellos factores de confiabilidad de acuerdo a las necesidades del sistema.

La calidad del servicio (*QoS* por su acrónimo en inglés) que permite configurar la calidad del aseguramiento de transmisión y recepción de mensajes típicamente configurado como: tan-sólo-una-vez *(at-most-once*), por-lo-menos-una-vez (*at-least-once*) y una-sóla-vez (*once-and-once-only*). Adicionalmente es posible configurar otros parámetros que delimitan la ejecución de las transacciones como: el Tiempo de Vida (*TTL*, por su acrónimo en inglés) para contemplar el tiempo de expiración del mensaje.

9.6.4 MQTT

Transporte de Telemetría de Colas de Mensajes o *MQTT (*del inglés *Message Queuing Telemetry Transport)* es un protocolo que permite una implementación muy ligera de publicación y subscripción de mensajes con el *overhead* de transmisión muy bajo. La siguiente información ha sido recabada del estándar 3.1.1 de Oasis *MQTT* publicado por OASIS MQTT Technical Committee, 2014.

MQTT es un protocolo de transporte de colas de mensaje con modelo cliente-servidor para publicador-subscriptor; ligero, simple de implementar y abierto. Corre sobre TCP/IP o cualquier otro medio de transporte que permita comunicación bi-direccional de forma confiable y ordenada. La cola de mensajes se realiza en modo FIFO.

9.6.4.1 Paquetes de datos

MQTT define tres distintos tipos de paquetes: Cabecera Fija, Cabecara Variable y Dato. La Cabecera Fija está presente en todos los Paquetes de Control *MQTT*. La Cabecera Variable está presente en algunos Paquetes de Control *MQTT* y el Dato también está presente sólo en algunos Paquetes de Control *MQTT*.

La Cabecera Fija contiene la información del Tipo de Paquete de Control y su longitud y definición se puede observar en la Figura 4.

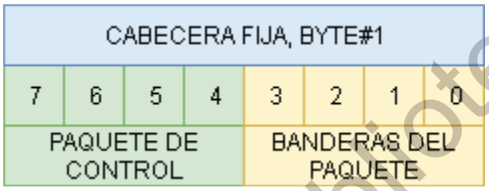

Figura 4 Byte de Cabecera Fija MQTT, bits de datos.

Los Tipos de Paquetes de Control se pueden observar en la Tabla 1. Cada uno de ellos representa el tipo de mensaje que se pueden intercambiar como parte del protocolo.

La Cabecera Variable es parte del mensaje para ciertos Tipos de Paquetes de Control y su longitud es variable. Los Tipos de Paquete que se forman de una longitud variable son PUBLISH, PUBREC, PUBACK, PUBREL, PUBCOMP, SUSBCRIBE, SUBACK, UNSUBSCRIBE y UNSUBACK. Por ejemplo el Tipo PUBLISH contiene una cabecera variable cuyo contenido indicar el Nombre del Tópico al que se desea subscribir y un Identificador del Paquete como referencia única.

El segmento Dato contiene la información o dato que se requiere transmitir a la capa de aplicación y es indispensable para ciertos Tipos de Paquetes.

Tabla 1 Tipos de Paquete de Control

Nombre	Valor	Descripción
Reservado	0	Reservado
CONNECT	1	Petición de conexión del cliente al servidor.

CONNACK	2	Reconocimiento de conexión
PUBLISH	3	Publicar mensaje
PUBACK	4	Reconocimiento de publicación
PUBREC	5	Publicación recibida (aseguramiento parte 1)
PUBREL	6	Publicación liberada (aseguramiento parte 2)
PUBCOMP	7	Publicación complete (aseguramiento parte 3)
SUBSCRIBE	8	Petición de subscripción
SUBACK	9	Reconocimiento de subscripción.
UNSUBSCRIBE	10	Petición para de-subscribirse.
UNSUBACK	11	Reconocimiento de de-subscripción.
PINGREQ	12	Petición de PING
PINGRES	13	Respuesta de PING
DISCONNECT	14	Desconexión de cliente.
Reservado	15	Reservado

9.7 Dispositivos *LTN*, *LPWAN* y *LoRaWAN*

LTN (Redes de Bajo Consumo de Datos, *Low Throughput Networks* por sus siglas en inglés) es una nomenclatura generada por el Instituto Europeo para Estándares de Telecomunicaciones quienes han trabajado en la generación de un estándar que permita sentar las bases para las comunicaciones inalámbricas de bajo consumo, bajo ancho de banda y largo alcance para aplicaciones *IoT*. LTN se define como una tecnología para redes inalámbricas de acceso amplio que permite el intercambio de información de largo alcance usando mínima cantidad de energía (“Low Throughput Networks,” 2017). Anteriormente, se ha definido a este tipo de tecnología con el nombre de LPWAN (Red de Acceso Ampliado de Bajo Consumo, *Low Power Wide Access Network* por sus siglas en inglés) una tecnología que se especializa en la interconectividad de dispositivos de largo alcance, con bajo ancho de banda y bajo consumo de energía; recomendadas para situaciones donde se requiera de monitoreo periódico aunque no para señales constantes o sistemas críticos (Margelis, Piechocki, Kaleshi, & Thomas, 2016). Existen diversas tecnologías que se pueden considerar LTN que son utilizadas en soluciones *IoT* tales como LoRaWAN[MR], Sigfox[MR], Ingenu[MR], NB-OIT[MR], etc en donde las diferencias más significativas radican en la modulación, protocolo de

comunicaciones, velocidad de transmisión y modelo de negocios de las empresas que las patrocinan (Mikhaylov, Petäjäjärvi, & Hänninen, 2016).

LoRaWAN (Red de Area Ampliada de Alto Alcance o *Long Range Wide Area Network* por sus siglas en inglés) es un protocolo LPWAN, a nivel de control de acceso al medio, MAC, del modelo OSI, desarrollado para aplicaciones de dispositivos operados a batería, de bajo consumo, bajo costo, alta escalabilidad, largo alcance y alta autonomía de batería que permite a dispositivos comunicarse con aplicaciones conectadas a internet de forma inalámbrica. La tecnología LoRaWAN se distingue de sus competidores como Sigfox, Ingenu y NB-IoT en el hecho de que cualquier desarrollador puede generar su propia solución mediante la instalación de una red privada; contrario al modelo manejado por sus competidores donde es necesario realizar un contrato para utilizar el servicio de sus redes, como en el caso de las redes celulares. Esta característica hacen de LoRaWAN una buena alternativa para aplicaciones que van desde nivel residencial hasta industriales (Wixted et al., 2017). En la Figura 5 se puede observar la arquitectura típica de una aplicación con LoRaWAN y en la Figura 6 es posible visualizar una comparación de la tecnología LPWAN contra otras tecnologías *IoT* existentes tomando en cuenta diversas características tanto funcionales como no funcionales.

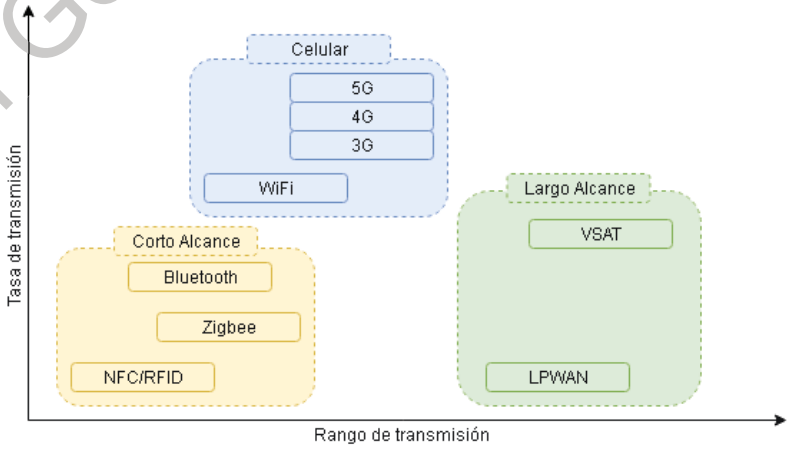

Figura 5 Rango comparativo de las tecnologías LTN vs redes celulares y de bajo alcance.

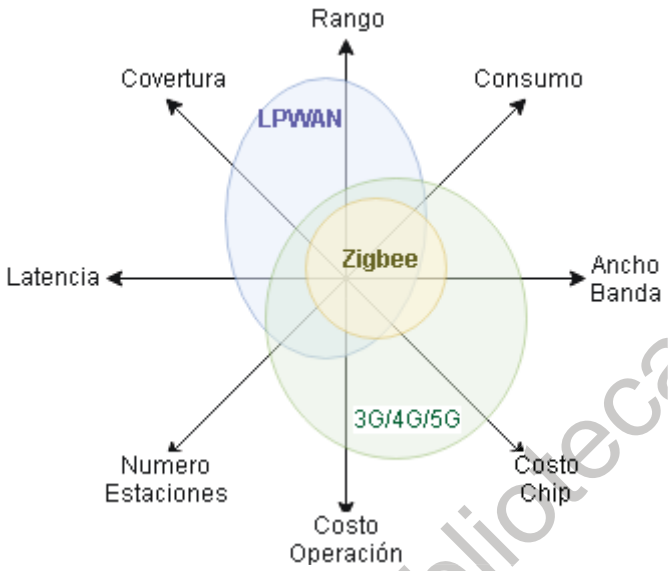

Figura 6 Diferencias tecnológicas de LTN (LPWAN) contra tecnologías rivales.

LoRaWAN está basado en una topología estrella donde cada nodo debe comunicarse con un *gateway* que actúa como concentrador y re-direccionador de datos hacia un servidor de red - aplicación (en la nube), en la Figura 7 se puede observar la distribución de sus componentes. La frecuencia de transmisión está fijada en la banda libre de uso científico industrial y médico ISM de 900MHz, para México. La modulación está basado en el protocolo *LoRa*[MR] (propiedad de Semtech), una forma de modulación por frecuencia que genera una oscilación estable y permitiendo la detección extremadamente sensible con bajos requerimientos de energía y velocidades de transmisión de entre 0.3kbps hasta 50 kbps.

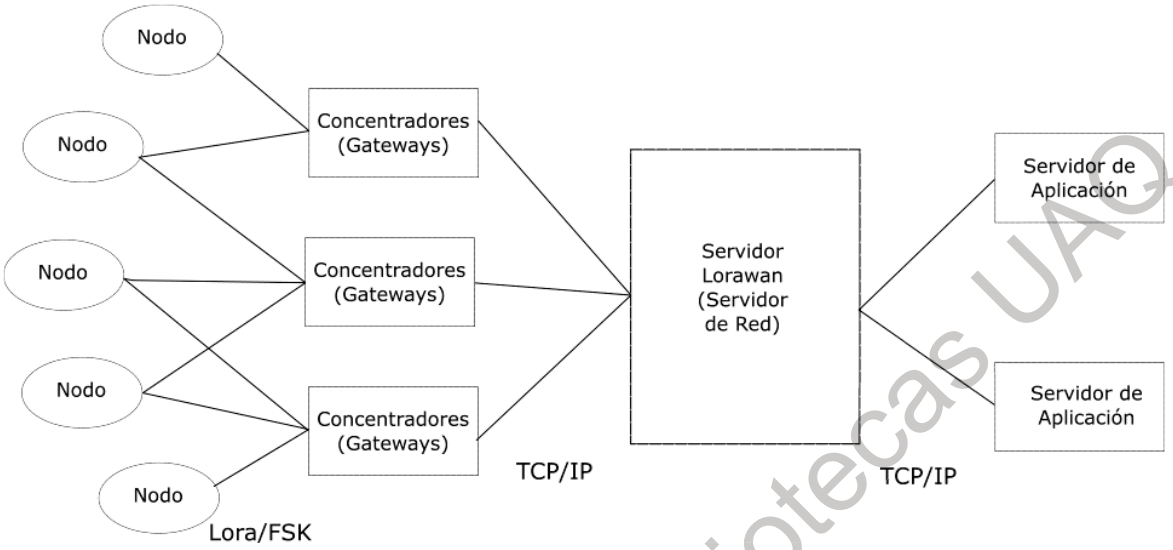

Figura 7 Arquitectura LoRaWAN

El estándar LoRaWAN considera el uso de canales de transmisión, en su versión 1.0, se implementan 8 canales, por lo que cada *gateway* puede recibir múltiples peticiones en diferentes canales concurrentemente. De igual manera, en esta versión del protocolo, se definen tres diferentes clases de dispositivos, diferenciados primordialmente por su capacidad de transmitir y recibir información periódicamente.

Clase A: Totalmente asíncrono; básicamente el nodo, al necesitarlo, envía información haciendo una transmisión desde el nodo hacia el servidor (*Uplink*). Y si es necesario realizar una transmisión desde el servidor hacia el nodo (*Downlink*) entonces genera una petición para que en las siguientes ventanas de comunicación se reciba la descarga juntamente con la confirmación de recepción de datos (*Acknowledge*). Estos son dispositivos de bajo consumo de energía.

Clase B: El *gateway* envía un "*Bacon*", o cadena de sincronización, periódicamente por lo que existe una certeza de comunicación bidireccional constante. Estos son dispositivos de consumo de energía intermedio.

Clase C: Principalmente para dispositivos sin restricciones de energía, en los cuales el nodo y *gateway* pueden intercambiar datos en todo momento. Estos son dispositivos de mayor consumo de energía.

La seguridad y privacidad de datos es un tema muy importante para los sistemas *IoT,* por tal motivo LoRaWAN incorpora encriptación en dos capas diferentes, la primera mediante una llave de red la cual permite que sólo nodos y *gateways* de una red definida puedan intercambiar datos; y por su parte, la llave de aplicación encripta la información para que sólo aquel servidor de red configurado pueda procesar dicha información utilizando encriptación AES[8] con claves de 128 bits (denominado *AppKey*) en diferentes modos de operación, además de incorporar un identificador global único EUI-64-based.

La configuración de una aplicación utilizando la tecnología LoRaWAN requiere de la realización, de por lo menos, las siguientes actividades:

1. Contar con nodos, sensores o actuadores, con capacidad LoRaWAN.
2. Cada nodo deberá contener las llaves de acceso a la red y a la aplicación.
3. Contar con un *gateway* con capacidad LoRaWAN.
4. Configurar el *gateway* con la llave de acceso a red, lo cual permite interceptar los mensajes de los nodos configurados con la misma llave.
5. Configurar el *gateway* con la dirección URL del servidor de red.
6. Servidor de red compatible con LoRaWAN, en el cual se configuran los dispositivos y se genera una llave de acceso a la aplicación.

[8] AES – Estándar de Encriptación Avanzada, *Advanced Encryption Standard* por sus siglas en inglés. Es un algoritmo de encriptación de llave pública basado en llaves secretas simétricas. Utilización en encriptación de mensajes y autenticación.

7. Desarrollar una aplicación final que permite recabar, interactuar y analizar los datos enviados y recibidos de los diferentes nodos.

Según el estudio realizado por Mikhaylov et al. (2016) la tecnología LoRaWAN está capacitada para proporcionar transmisión de datos (*Uplink*) a baja velocidad de entre 100 bit/s a 2 Kbit/s dependiendo de la distancia con respecto a la antena receptora *gateway*. Además, por el protocolo que utiliza (el cual no posee un mecanismo de petición de acceso al medio o disponibilidad de la antena receptora para enviar un paquete) se prioriza la baja latencia y bajo consumo energético en redes de baja densidad; esta misma caracterísitica ocasiona que en sistemas altamente saturados exista mayor riego de colisiones, y por consecuencia, aumentando la latencia. Por lo tanto: "LoRaWAN puede ser utilizada en redes de densidad moderada y de bajo tráfico de datos, en los cuales no se impone restricciones estrictas de latencia o requerimientos de confiabilidad, como sistemas no críticos" Mekki, Bajic, Chaxel, y Meyer (2018).

Una de las grandes ventajas de LoRaWAN, que lo hacen un fuerte competidor dentro de las redes LTN, es su especificación de comunicación libre y puede ser descargado desde la página web de la Alianza LoRa, esto permite que cualquier empresa pueda crear su propio hardware de comunicación para crear soluciones operadas por ellas mismas, sin la necesidad de pagar planes de datos (como pasa en el caso de Sigfox). Es importante notar que al ser una tecnología patentada y adquirida por Semtech, actualmente sólo esta empresa manufactura los chips LoRa decodificadores de radiofrecuencia (*Lora-PHY* – capa física). Pese a esto, cualquier empresa o consultor puede integrar estos chips *LoRa* para poder crear alguna solución LoRaWAN que cumpla con la especificación, desplegarla en una red privada y montar el sistema de acuerdo a sus necesidades (Wixted et al., 2017). Esta característica es la razón por la cual se utilizará LoRaWAN como parte del presente trabajo de investigación y proponer una arquitectura *middleware* que permita la interoperabilidad de sistemas heterogéneos.

Además de los modos de transmisión es importante mencionar los ajustes comunes a nivel capa física que incorporan los módulos de radio típicos de la tecnología LoRa:

Potencia de transmisión. Regularmente se puede esperar un rango de ajuste de entre -4 dBm a 20 dBm. El rango final está limitado a las restricciones de hardware en particular.

Frecuencia portadora. Es la frecuencia central misma que puede ser ajustada en rango de entre 137 MHz a 1020 MHz en incrementos de 61 Hz. El rango final está limitado a las restricciones de hardware en particular.

Factor de difusión (*Spreading Factor, SF*). Es la relación entre la frecuencia por símbolo y la frecuencia de pulso. Al incrementar el SF también se incrementa la Razón Señal Ruido (*Signal to Noise Ratio, SNR*), por lo tanto la sensibilidad y el rango de transmisión, de igual manera incrementando el tiempo de transmisión y la energía consumida. Por ejemplo, un SF de 12 tiene una razón de 2^{12} = 4096 pulsos/símbolo.

Ancho de banda (*Band Width, BW*). Es el ancho de las frecuencias en la banda de transmisión. A mayor BW se tiene mayor tasa de transmisión (menor tiempo en aire y menor consumo de energía), pero también menor sensibilidad. La transmisión de datos o frecuencia de pulso se da a la misma frecuencia que el BW. Para un BW de 125 KHz se tiene una frecuencia de pulso de 125 kcps.

Tasa de codificación (*Coding Rate, CR*). Es la Corrección de Error hacia Adelante (*Foward Error Correction, FEC*) utilizada para afrontar la interferencia de ráfaga y se puede fijar a 4/5, 4/6, 4/7 o 4/8. A mayor CR se tiene mayor protección pero también mayor tiempo en el aire. (Bor, Roedig, Voigt, & Alonso, 2016).

De acuerdo al documento "*Regional Parameters*" de la especificación LoRaWAN 1.0 el rango de frecuencia para uso de la tecnología LoRa en México se encuentra incluida en el rango de 902 a 928 MHz el cual se encuentra disponible en

los modos “US902-928, AU915-928” del estándar o también conocido con un nombre común de US915 y AU915 respectivamente. De tal forma, que US915 engloba a los estándares aplicados a Estados Unidos de América; mientras que AU915 a aquellos correspondientes a Australia.

De acuerdo al documento “*LoRaWAN Backend Interfaces*” de la especificación LoRaWAN 1.0 se tienen los siguientes tipos de actores del proceso de comunicación:

Nodos o “End-Device”. Son aquellos elementos, sensores y/o actuadores, que generan o consumen datos y se comunican de forma inalámbrica con los *gateways*. Los datos contenidos en la Capa de Aplicación del Nodo es transmitida al Servidor de Aplicación.

Gateways. Reenvía todos los paquetes recibidos hacia el Servidor de Red LoRaWAN, éste se conecta a una red de internet. El *gateway* opera totalmente a nivel de Capa Física.

Servidor de Red LoRaWAN (NS) o Servidor LoRaWAN. Está centrada en el funcionamiento de la capa 2 del modelo OSI con el Control de Acceso al Medio (MAC) de los Nodos. Algunas funciones que realiza son:

- Verificación de la dirección del Nodo.
- Autenticación de tramas y verificación de contador de transmisiones.
- Confirmación de transmisiones.
- Adaptación de Razón de Datos.
- Responder a todos los mensajes de nivel MAC provenientes de todos los Nodos.
- Reenvío de los datos de Aplicación hacia los Servidores de Aplicación configurados.

- Encolado de tramas de bajada (respuesta) hacia los Nodos provenientes de los servidores de Aplicación.

- Renvío de peticiones de Autenticación hacia los Servidores de Activación.

Servidor de Activación. Este servidor gestiona el proceso de Activación a Través del Aire (*OTA*, por sus siglas en inglés) mediante la autenticación de las diversas llaves necesarias para realizar la conexión.

Servidor de Aplicación. Permite que la información proveniente de los Nodos sea utilizada de forma útil de acuerdo a la lógica de negocio. De igual manera contiene las reglas necesarias para comunicar de vuelta información hacia los Nodos.

Los Nodos pueden activarse para la participación en la red mediante alguna de las siguientes dos maneras:

Activación Personalizada (ABP). En este modo de activación los Nodos cuentan con las llaves del Servidor *LoRaWAN* (dirección del dispositivo *dev_addr*, llave de sesión de red *nwks_key* y llave de sesión de aplicación *apps_key*) y por lo tanto no requieren de un proceso de Activación adicional.

Activación A través del Aire (OTA). La activación se hace mediante un conjunto de llaves (identificador única de dispositivo *dev_eui*, identificador única de aplicación *app_eui* y la llave de aplicación *app_key*) que permiten generan la validación por parte del servidor de Activación para que éste devuelva las llaves de conexión del Servidor de Red.

9.8 Datagramas UDP

Los datagramas son paquetes de datos utilizados para transmitir información en una red no orientada a conexión por lo que los datagramas incluyen una cabecera con la información del destinatario y de origen. Es un método muy ligero de transmitir información de internet en operaciones enfocadas en realizar transacciones. Tiene

el inconveniente de no garantizar la entrega de mensajes como en el caso de TCP/IP.

La estructura de los datagramas UDP es muy sencilla, conteniendo una cabecera compuesta por: dos bytes para indicar el puerto origen, dos bytes para el puerto destino, dos para longitud, n bytes del dato incluyendo la cabecera, dos bytes más para el checksum complemento a uno y n bytes del dato transmitido (Postel, 1980).

9.9 OPC-UA

El estándar OPC (Comunicaciones de Plataforma Abierta, *Open Platform Communications* por sus siglas en inglés) es parte importante en la generación de soluciones que integran diferentes tecnologías en aplicaciones, sobre todo, de automatización industrial. tal como lo mencionan Cupek y Ziebinski (2017) quienes han propuesto un *gateway* utilizando OPC-UA para conectar una red de dispositivos CAN y sistemas ciber físicos logrando realizar una comunicación entre ambos dominios de forma exitosa. Debido a lo relevante de este tema para esta investigación a continuación se describen los antecedentes y las características más importantes de OPC.

OPC deriva su nombre de su primera versión de Control de Procesos Embebido y Linqueo de Objetos *(Object Linking and Embedding for Process Control por sus nombre en inglés*) originalmente creado para permitir la interoperabilidad de equipos, máquinas y herramientas en entornos de telecomunicación industrial, inicialmente haciendo uso del ambiente .NET de Microsoft. Actualmente la Fundación OPC[9], define a OPC como Plataforma de Comunicaciones Abierta (*Open Platform Communications* por su nombre en inglés) ya que su actual estándar OPC Arquitectura Unificada (UA, *Unified Architecture* por su nombre en inglés) va más

[9] https://opcfoundation.org/

allá del entorno .NET y está disponible en diferentes lenguajes de programación para diferentes entornos de desarrollo. OPC-UA fue lanzada en el año 2008 por la Fundación OPC, es compatible con la versión original, ahora denominada *Classics.* Implementa mecanismos de seguridad y es extensible para agregar nuevas funcionalidades y comprensible a través de modelados simples o complejos ("Unified Architecture. OPC Unified Architecture Specification," n.d.).

El estándar OPC permite la interoperabilidad de equipos como controladores lógicos programables (PLC), interfaces hombre-máquina (HMI), sistemas SCADA, Robots, estaciones de prueba, entre muchos otros dispositivos. OPC ha sido formalmente estandarizado en el estándar IEC-62541 (Grüner et al., 2016). OPC define un conjunto de métodos, objetos e interfaces que permiten la interoperabilidad entre estos dispositivos gracias a que todos los miembros involucrados se comunican utilizando el mismo lenguaje sin importar el medio o protocolo de comunicación. La gran ventaja es que al diseñar y producir un nuevo dispositivo compatible con OPC automáticamente los hace compatible con la inmensa mayoría de dispositivos actualmente instalados en las industrias.

OPC UA está basado en un modelo de comunicación cliente-servidor con servicios como búsqueda, escritura/lectura, llamado de métodos. Provee mecanismos de seguridad como autenticación, autorización, encriptación e integridad de datos basados en los estándares criptográficos como lo son los estándares PKI, AES y SHA. En el mes de febrero del año 2018 la fundación OPC publicó la extensión Pub-Sub (Publicación-Subscripción), el cual es el mecanismo que permite la operación de OPC UA en un modelo no cliente-servidor sino publicador-subscriptor que cuenta con elementos como configuración, conexiones y grupo de datos (utilizando protocoles como AMQP, MQTT) que permiten generar aplicaciones más complejas, escalables, y con restricciones de tiempo (baja latencia).

OPC Pub Sub está diseñado para suplir las necesidades del paradigma de integración de alta escala, como el del Internet de las Cosas, en donde una gran cantidad de nodos o dispositivos pueden conectarse de manera constante o intermitente. El modelo tradicional, servidor-cliente, necesita contar con una conexión establecida para poder realizar el intercambio de datos a petición del cliente o servidor. Por el contrario el modelo publicador-suscriptor se basa en dos actores: los publicadores quienes generan datos y los suscriptores quienes consumen la información. Este tipo de modelo permite las condiciones para modelos de comunicación de uno a muchos y de muchos a muchos (Ray, 2010).

A continuación se presenta un breve resumen de las especificaciones del estándar OPC UA obtenido de la versión 1.4 disponible en la Fundación OPC complementado con algunos conceptos importantes mencionado por Postol (2016).

9.9.1 Terminología

Espacio de Direcciones (*AddressSpace*). Colección de información que un servidor OPC UA hace visible a sus clientes.

Conjunto (*Aggregate*). Una función que calcula los valores derivados de una trama de datos en crudo.

Alarma. Tipo de evento asociado con una condición de estado que típicamente requiere de reconocimiento.

Atributo. Característica primitiva de un nodo.

Intermediario (*Broker*). Programa intermediario que se encarga de encaminar mensajes de red desde publicadores a suscriptores.

Certificado. Estructura de dato firmada digitalmente que contiene una llave pública y la identificación del cliente o servidor.

Cliente. Aplicación de software que manda mensajes a un servidor OPC UA de acuerdo a los servicios especificados en este conjunto de especificaciones.

Stack *de Comunicación*. Conjunto de módulos acomodados en capas entre la capa de aplicación y la capa de hardware, que provee de varias funciones para codificar, encriptar y formatear un mensaje para envio o recepción.

Dato Complejo. Es un dato que se compone por elementos de contienen más de un tipo de dato primitivo, como las estructuras.

Conjunto de Datos (*DataSet*). Listado de valores de datos. Típicamente consiste de campos de eventos y valores de variables.

Mensaje de Conjunto de Datos (*DataSetMessage*). Dato transmitido de un mensaje de red creado por un conjunto de datos.

Descubrir. Proceso por el cual un Cliente OPC UA obtiene la información acerca de un servidor OPC UA incluyendo su información de enlace y seguridad.

Evento. Término genérico usado para describir la ocurrencia de algo significativo concerniente al sistema o componente del sistema.

Notificador de Evento. Se trata de un atributo de un nodo cuyo significado dice que un cliente puede suscribirse al nodo para recibir notificaciones de ocurrencias de un evento.

Modelo de Información. Marco organizacional que define, caracteriza y relaciona fuentes de información de un sistema dado o conjunto de sistemas. Se realiza mediante la asignación de espacio de direcciones.

Mensaje. Unidad de dato referenciado entre el Cliente y Servidor y que representa una petición o respuesta a un servicio en particular.

Mensaje de Red. Mensajes de conjunto de datos y sus cabeceras que facilitan su entrega, enrutado, seguridad y filtrado.

Método. Función de software que puede llamarse y es un componente de un objeto.

Artículo Monitoreado. Entidad definida por un cliente en el servidor utilizado para monitorear atributos o notificadores de eventos y así detectar nuevos valores o la ocurrencia de eventos y la generación de sus correspondientes notificaciones.

Nodo. Componente fundamental de un espacio de direcciones.

Clase Nodo. Clase de un nodo en el espacio de direcciones.

Notificación. Término genérico para datos que anuncian la detección de un evento o el cambio del valor de un atributo. Las notificaciones son enviadas en mensajes de notificaciones.

Objeto. Nodo que representa un elemento físico o abstracto de un sistema.

Instancia de Objeto. Sinónimo de objeto.

Tipo Objeto. Nodo que representa la definición del tipo de un objeto.

Aplicación UPC-UA. Se puede referir a un Cliente OPC-UA que llama los servicios de un Servidor OPC-UA, quien realiza esos servicios. O puede referirse a un suscriptor o publicador OPC-UA.

Publicador. Entidad que envía mensajes de red a un *middleware* orientado a mensajes.

Pub/Sub. Variante de OPC-UA con el patrón de mensajería publicador-suscriptor.

Perfil. Conjunto específico de capacidades al cual un servidor puede argumentar cumplir.

Programa. Objeto ejecutable que, al momento de ser invocado, inmediatamente regresa una respuesta para indicar que la ejecución ha comenzado y luego regresa los resultados intermedios o finales a través de suscripciones inidentificadas por el cliente durante la invocación.

Referencia. Relación explícita (puntero con nombre) de un nodo a otro.

Tipo Referencia. Nodo que representa la definición a una referencia.

Servidor. Aplicación de software que implementa y expone los servicios especificados en su conjunto de especificaciones.

Servicio. Operación del cliente que puede ser llamado en un servidor OPC-UA.

Conjunto de Servicios. Grupo de servicios relacionados.

Sesión. Conexión durable lógica que se realiza entre un cliente y un servidor.

Subscriptor. Entidad que recibe mensajes de conjunto de datos en un *middleware* orientado a mensajes.

Subscripción. Enlace definido por el cliente en el servidor utilizado para regresar notificaciones al cliente.

Sistema en Cuestión (*Underlying System*). Plataforma de hardware o software que existe como una entidad independiente. Las aplicaciones UA dependen de la existencia de una entidad para poder realizar los servicios UA. Sin embargo, la entidad no es dependiente de una aplicación UA.

Variable. Nodo que contiene un valor.

Vista. Subconjunto específico de un espacio de direcciones que es del interés del cliente.

9.9.2 Alcance de la Arquitectura Unificada

OPC UA se puede aplicar a los componentes de todos los dominios industriales tales como sensores y actuadores, sistemas de control, sistemas de ejecución de manufactura, sistema de planeación de recursos empresariales, incluyendo Internet de las Cosas Industriales (Industria 4.0), comunicación Máquina a Máquina. OPC-

UA define un modelo común de infraestructura que facilita el intercambio de información, especificando lo siguiente:

- Modelos de información que se pueden utilizar para representar estructuras, comportamientos, semántica y describir la arquitectura sistémica de un proceso de tiempo real.
- Modelos de mensajes que permiten interactuar entre aplicaciones.
- Modelos de comunicación que hacen posible intercambiar datos entre puntos finales, nodos.
- Modelos de conformidad que son la base para la interoperabilidad entre sistemas heterogéneos.
- Modelos de seguridad que reflejan la creciente demanda de sistemas preparados para incorporar protecciones de ciber seguridad.

9.9.3 El Espacio de Direcciones

El Espacio de Direcciones tiene el propósito de soportar la integración entre sistemas heterogéneos mediante la construcción de significado entendible entre sí (semántica y sintaxis) basado en un lenguaje lo suficientemente robusto como para intercambiar mensajes sin importar la jerarquía del sistema en cuestión. En otras palabras, el sistema servidor necesita describirse y el sistema cliente necesita conocer la descripción del servidor mediante un canal de intercambio de datos, la estructura que permite este proceso es el Espacio de Direcciones. Para poder realizarlo es necesario contar con: (i) el Modelo de Información con el objetivo de representar información sin ambigüedades. y (ii) el Modelo de Servicio para establecer las bases de la transportación de datos.

A través del Espacio de Direcciones es posible consultar el servidor de metadatos, permitiendo que clientes que no conocen el formato de los datos, puedan determinarlos en tiempo de ejecución y utilizarlos correctamente. Además,

OPC UA da soporte a la relaciones entre nodos en vez de limitarse a sólo una jerarquía. De esta manera se presentan los datos de distintas jerarquías adaptados a la forma que el cliente normalmente los quiere ver. El Espacio de Direcciones de OPC UA representa una serie de nodos conectados por referencia, similar, pero no limitado, a una estructura tipo árbol.

Para promover la interoperabilidad OPC UA define perfiles, que son un conjunto de servicios que deben ser implementados de acuerdo a las capacidades del hardware. Se cuenta con los siguientes perfiles: Embebido Nano, Embebido Micro, Embebido, Estándar y Global, dependiendo del perfil ejecutado el Servidor OPC soportará una cierto conjunto de servicios y métodos.

Las características primitivas de los nodos se describen por atributos definidos por OPC, estos atributos son los elementos de un servidor que tienen valores, y los tipos de datos pueden ser simples o complejos.

En el espacio de trabajo, los nodos se clasifican de acuerdo a su uso y significado, las clases de nodos definen los metadatos para el espacio de direcciones OPC UA.

Por ejemplo, la clase NodoBase define los atributos comunes a todos los nodos, para definir la identificación, nomenclatura y clasificación. Cada clase de nodo hereda estos atributos, y además se pueden sumar otros propios. La clase NodoBase es heredada por otras sub-clases permitiendo composiciones de datos complejos.

9.9.4 Sesiones

Las sesiones se definen como las conexiones lógicas entre clientes y servidores que permiten el intercambio de información. Cada sesión es independiente de los protocolos de comunicación, por lo que si falla el protocolo la sesión no se interrumpe automáticamente; la sesión termina cuando el cliente o el servidor lo solicitan.

9.9.5 Descubrir

Este grupo de servicios se usan para descubrir los servidores OPC UA que están disponibles en el sistema, también da a los clientes la posibilidad de leer la configuración de seguridad para la conexión con el servidor. Esta serie de servicios es implementada por servidores individuales y que proporcionan a los clientes la manera de descubrir todos los servidores OPC UA registrados.

9.9.6 Canal seguro

Este grupo de servicios comprende los que son usados para abrir un canal de comunicación que garantice la confidencialidad e integridad de los mensajes intercambiados con el servidor.

Un canal seguro conecta un único cliente a un servidor, y mantiene una serie de claves que sólo conocen ambos, utilizada para autenticar y encriptar los mensajes enviados. Los servicios de canal seguro permiten negociar de forma segura las claves para usar entre el cliente y el servidor. Los algoritmos exactos usados para autenticar y encriptar los mensajes están descritos en las políticas de seguridad de un servidor, y se pueden consultar en los servicios de Descubrir descritos anteriormente.

9.9.7 Vistas

Las vistas están definidas públicamente; el espacio de direcciones completo es la vista predeterminada, por esto los servicios de vistas son capaces de operar en todo el espacio de direcciones.

Los clientes pueden detectar nodos fácilmente en una vista, pueden desplazarse hacia arriba o abajo en las jerarquías.

9.9.8 Consulta

Los servicios de consulta permiten a los usuarios acceder al espacio de direcciones sin navegar y sin conocimiento del esquema lógico utilizado para el almacenamiento interno de los datos. Se pueden seleccionar conjuntos de nodos en una vista basados en algún criterio del cliente para filtrarlos, estos nodos seleccionados se denominan conjunto de resultados.

9.9.9 Métodos

Son servicios de llamados a procedimientos remotos que devuelven valores dependiendo del resultado de la operación. Un método está asociado siempre con un objeto. Los clientes pueden buscar, para los objetos, los métodos soportados por un servidor.

9.9.10 Subscripción

Los servicios de subscripción los utiliza el cliente para crear y mantener la conexión constante de los valores de un cierto nodo. El formato de estas notificaciones es específico para el tipo de artículo que monitorea. Una vez que se crea la subscripción, ésta existe independientemente de la sesión del cliente con el servidor. La duración de la subscripción se configura, y el cliente lo debe renovar periódicamente, de no hacerlo la subscripción para el servidor se cancela; en este caso, los artículos monitoreados asignados a ella son borrados.

Las subscripciones incluyen funciones que detectan y recuperan los mensajes perdidos. Cada mensaje de notificación tiene una secuencia de números que permite que los clientes detecten si se pierde un mensaje, así los clientes pueden pedir al servidor que se reenvíe uno o más mensajes.

9.9.11 Modelo Objeto

El principal objetivo de la OPC UA es proveer un estándar que permita a los servidores representar los estados de los procesos en tiempo real y su entorno a

los clientes, en una forma única e independiente de la plataforma. El modelo objeto ha sido diseñado precisamente para este fin.

El modelo objeto define los objetos en términos de variables y métodos, y también permite expresar las relaciones con más objetos como referencias. Al usarlo, los clientes tienen acceso a una parte específica del proceso. Por tanto, los objetos deben haber establecido conexión con los dispositivos del Sistema en Cuestión.

9.10 Sistemas Ciber Físicos (CPS)

El concepto de Sistemas Ciber Físicos (*CPS, Cyber Physical Systems* por su siglas en inglés) ha cobrado relevancia debido a la gran cantidad de aplicaciones *IoT* y se refiere a las tecnologías basadas en el campo de la mecatrónica cuya aplicación se basa en dispositivos inteligentes que controlan componentes e interactuan por algún medio computacional. Los *CPS* incluyen sistemas compuestos por sensores, actuadores, sistemas de control, redes distribuidas e interfaces hombre máquina que producen y consumen información, y su comportamiento se adapta con respecto a la información recabada. La integración de los *CPS* en procesos productivos resulta en lo que se conoce como Industria 4.0 un nuevo paradigma en la industría manufacturera (Lee, Bagheri, y Kao, 2014).

10. ARQUITECTURA PROPUESTA

Para generar una solución a la hipótesis planteada, a través de los objetivos mencionados, es necesario ir resolviendo cada una de las interrogantes. En primer lugar es necesario entender los distintos elementos fundamentales de las tecnologías involucradas, las cuales se pueden observar en la Figura 8. A continuación se realiza el análisis de cada una de las tecnologías utilizadas.

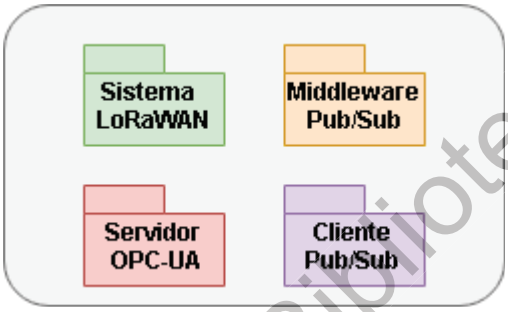

Figura 8 Componentes de la investigación

Por un lado la tecnología LoRaWAN está formada por tres elementos principales en donde los Nodos LoRaWAN son aquellos dispositivos que producen o consumen información valiosa para los usuarios, normalmente son operados a batería y se comunican de forma inalámbrica; los *Gateways* LoRa cuya principal función es captar la información codificada desde los Nodos y reenviarlos, sin ser tratados, mediante protocolos de comunicación basados en internet hacia Servidores LoRaWAN cuyo papel es identificar, asociar y filtrar cada uno de los mensajes para ser utilizados en aplicaciones finales donde a la información recibida se transforma en la regla de negocio correspondiente.

El Servidor OPC-UA cuya principal tarea consiste en proveedor de servicios estandarizados al sistema en cuestión permitiendo la interoperabilidad con cualquier cliente que siga las reglas del estándar OPC. El Espacio de Direcciones es el nombre dado al sistema modelado (o sistema en cuestión) y los elementos de la estructura jerárquica son representados por Nodos. Existen una cantidad de

Servicios que se encuentran disponible para ser utilizados por los clientes. El Servidor que se requiere para esta tesis debe considerar la operacíón en dos modos de operación simultaneas: Cliente-Servidor y Publicador-Subscriptor.

El Cliente OPC-UA puede acceder a la información disponible de cualquier Servidor OPC-UA, siempre y cuando se acceda mediante los mecanismos de seguridad configurados en el Servidor. Debe soportar como mínimo el modo Cliente-Servidor para poder probar las capacidades de computación del borde, o servicios disponibles a nivel Servidor.

El *Middleware* Pub/Sub provee el bajo acoplamiento entre los actores involucrados mediante la aplicación del patrón de interacción de mensajes Publicador-Subscriptor de un *middleware* MOM el cual proporciona la capacidad de envío de datos de uno-a-muchos o de muchos-a-muchos permitiendo un sistema altamente escalable. El Intermediario se encarga del manejo de la cola de mensajes. La Capa de Transporte debe ser provista por un protocolo de paso por mensajes.

10.1 Análisis de la arquitectura de referencia OPC-UA

Se realizó el análisis del código fuente del proyecto de referencia OPC-UA Pub/Sub .Net Estándar versión 1.04 disponible gracias a la Fundación OPC y en el que se pudieron identificar tres distintas capas de asbtracción mostradas en la Figura 9: (i) OPC.Core en el cual se encuentran definidas todas las clases bases que cumplen con la estructura del estándar OPC-UA, incluyendo definición de Tipos de Datos, Esquemas, mecanismos de Autenticación, abstracción de capa de Transporte, Nodos, Servidor, Cliente, Estados, Constantes y Configuraciones; (ii) el SDK para el lenguaje .Net Standard que implementa las característica de la capa Core adaptada en el funcionamiento del lenguaje de programación .Net 4.5; y (iii) la capa de aplicación que instancía un servidor totalmente funcional con las características provistas en las capas de bajo nivel. Para el caso particular del paquete Servidor, se parte de una clase interface *iServerBase* que es realizada por la clase *ServerBase* que a su vez es realizada por *SessionServerBase* que es heredada por

la clase *StandardServer* la cual realiza cada una de las interfaces externas definidas en las clases bases utilizando las características particulares de sintaxis y estructura definidas por C# .Net Standard. A su vez la clase *StandardServer* es la plataforma del Servidor de aplicación en donde se implementa el sistema en cuestión.

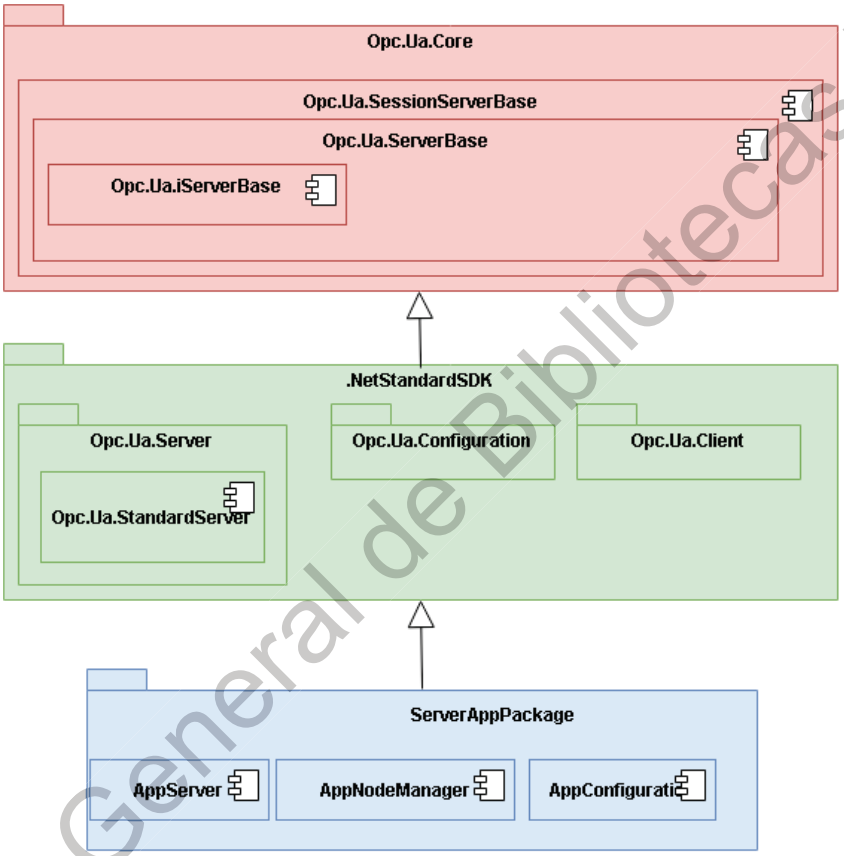

Figura 9 Clases de abstracción Servidor OPC-UA

10.2 Análisis del protocolo LoRaWAN

La especificación LoRaWAN versión 1.0 define dos tipos de tramas de datos. El primer modo, *Downlink o* descarga, para transmitir mensajes desde la aplicación hasta el dispositivo final (Nodo) mediante un *gateway*: y el segundo modo, *Uplink* o carga, para la operación inversa en donde el Nodo transmite la información hacia capas altas a traves de uno o varios gateways.

Para tramas *Uplink* la información PHDR, PHDR_CRC y CRC es insertada por el trasceptor de comunicación. PHDR es el encabezado de trama correspondiente al protocolo LoRa y el CRC o mecanismos de comprobación de la validez de la información mediante la detección de errores por verificación de redundancia cíclica. La descomposición de los elementos del mensaje LoRaWAN se puede observar en la Figura 10.

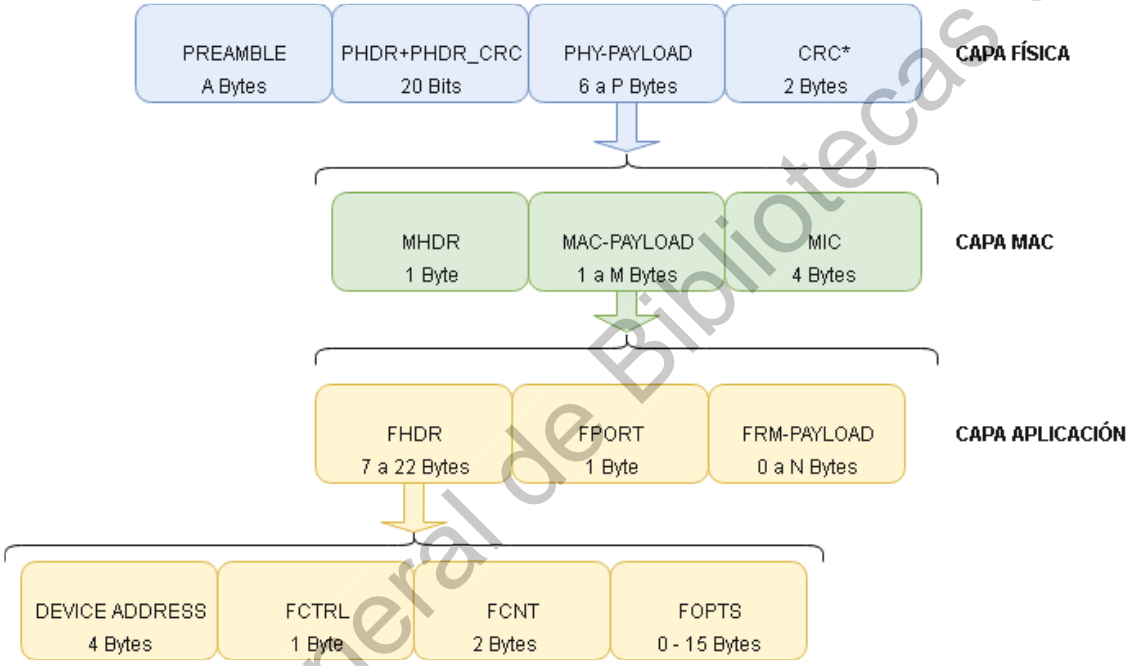

Figura 10 Trama de datos del protocolo LoRaWAN rev1.0

El campo Preamble es insertado por el *gateway* y contiene la dirección física del *gateway* (Dirección MAC) y algunos valores aleatorios.

El campo PHYPayload contiene la información codificada en LoRa que puede contener información cifrada LoRaWAN.

Ambos tipos de tramas descritos conforman el protocolo de datos de la capa física de los transceptores de radio modulados en LoRa y son invisibles para las sub-capas de enlace de datos ya que sólo el segmento PHYPayload es decodificado en las capas superiores.

PHYPayload contiene la información de la trama LoRaWAN comenzando con un encabezado MHDR de un byte que identifica el tipo de mensaje y la versión del protocolo utilizado para codificar el mensaje. MACPayload contiene la trama de datos del mensaje transmitido y MIC que contiene un código de integridad de la trama PHYPayload a cuatro bytes cifrado mediante AES CMAC de 128 bits RFC4493 (FRMPayload deberá estar cifrado antes de realizar este cálculo).

El segmento MACPayload está compuesto de un encabezado del mensaje LoRaWAN, FHDR, cuya información incluye cuatro bytes de la dirección del Nodo (DevAddr); un byte de control de mensajes (FCtrl) donde se especifica modos de operación para los estados de transmisión, recepción de mensajes y la longitud de los comandos FOpts en caso de existir; dos bytes para contador de mensajes (FCnt) que sirve como elemento verificador para cada actor, que da a conocer la calidad del servicio de los mensajes recibidos/enviados; y hasta quince bytes con comandos espaciales (FOpts) para transportar comandos de la capa MAC.

FPort puede estar compuesto de una cantidad de cero o un byte. Si la trama MACPayload contiene FRMPayload entonces FPort debe estar presente. Un valor de cero indica que FRMPayload contiene sólo comandos para la capa de MAC. Valores distintos de cero pueden ser utilizados según se requiera.

FRMPayload contiene el dato con la información transmitida por o para el Nodo y se encuentra cifrada por algoritmo simétrico AES 128 bits descrito en el IEEE802154.

El cifrado de FRMPayload tiene una llave dada por el valor de Fport, si el valor es cero la llave que se utiliza es NwkSKey (llave de configuración del Nodo para acceso a la red LoRaWAN) y si su valor es diferente, entonces se utiliza AppSKey (llave de configuración del Nodo para acceso a la aplicación configurada en el Servidor LoRaWAN). El vector de incialización de 16 bytes inicializados a cero y cuyo vector de encriptación viene dado por S. El valor cifrado de FRMPayload

viene dado por la XOR de sí misma y S a 16 bytes (pld se refiere a FRMPayload y k viene dado por Ceil(len(pld)/16)) tal como se muestra en la Figura 11.

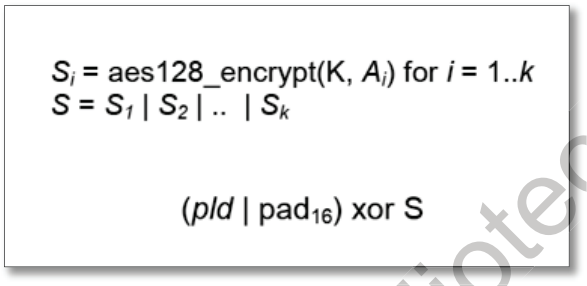

Figura 11 Algoritmo de cifrado LoRaWAN

10.3 Análisis del *Gateway* y Reenviador de Paquetes LoRaWAN.

El *gateway* típico LoRaWAN está compuesto por una capa de bajo nivel que se encarga de demodular las señales LoRa provenientes de los transceptores LoRa, el software controlador y de un módulo denominado Reenviador de Paquetes, o *Packets Forwarder*, (por sus nombre en inglés) que reenvía la información codificada hacia la capa MAC y superiores, se puede observar en la Figura 12. Para poder conocer los elementos de un *gateway* típico se realizó el análisis del paquete disponible en código abierto por Semtech.

El siguiente análisis se realiza sobre el módulo Packet Forwarder:

- Nombre del paquete: packet_forwarder
- Versión: 1.2
- Repositorio: https://github.com/Lora-net/packet_forwarder.git

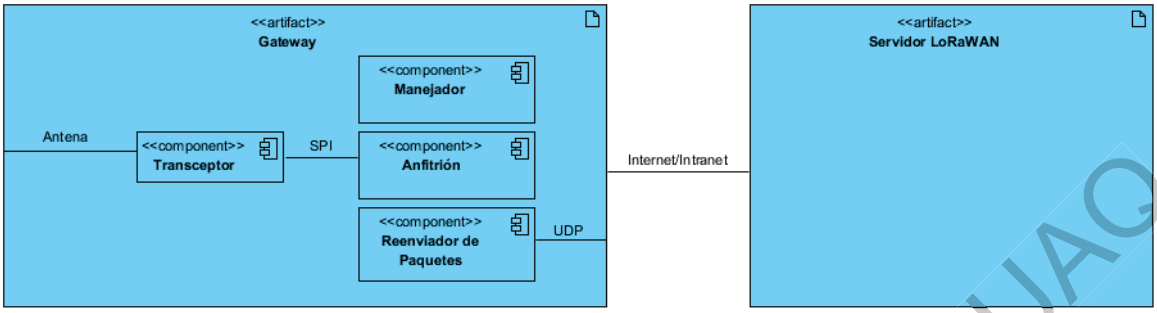

Figura 12 Análisis del Gateway LoRaWAN

El *Gateway* se refiere al dispositivo que puede leer y enviar datos desde y hasta nodos LoRaWAN y conectarlo a un Servidor LoRaWAN.

El módulo Transceptor es un dispositivo electrónico que permite la recepción y trasmisión de paquetes en el protocolo LoRa, por el ejemplo los transceptores SX1301 o SX1276 cuya principal función es (de)modular en protocolo LoRa. Este módulo actúa a nivel capa física.

El Anfitrión es el recurso de hardware y software necesario para recibir los paquetes desde el módulo transceptor a través de un puerto de comunicación serial síncrono SPI, interpretarlo mediante el software manejador y reenviarlos hacia el servidor mediante una conexión de red. La transmisión de datos se realiza mediante datagramas UDP.

El Servidor LoRaWAN es el dispositivo que interpreta las señales de LoRa a nivel Capa 2 (LL y MAC) y permite decodificar la señal recibida y además publicar la información para Servidores de Aplicación LoRaWAN.

10.3.1 Paquete de datos de carga tipo *PUSH*

Cuando el *gateway* recibe uno o más paquetes, o cuando transmite su estatus realiza un envío de mensaje tipo *PUSH_DATA* que incluye la versión del protocolo, un token aleatorio, la dirección MAC y el paquete de datos y metadatos en formato

JSON. La composición de los elementos de este tipo de datos se puede observar en la Tabla 2, Tabla 3 y Tabla 4 (a partir de la siguiente página).

Cuando el Servidor recibe el dato, deberá responder con un *PUSH_ACK* reconociendo que ha recibido el dato, el frame de respuesta deberá contener el mismo token recibido.

El Servidor LoRaWAN es responsable de tratar y procesar los paquetes recibidos.

Tabla 2 Objeto tipo *PUSH_DATA*

Bytes	**Función**
0	Versión del protocolo, 0x01 para código analizado.
1-2	*Token* aleatorio generado por el *gateway*.
3	Identificador tipo PUSH_DATA, 0x00
4-11	Identificador único del *gateway* o dirección MAC
12-final	**DATO_JSON.** Dato transmitido en formato JSON conteniendo los corchetes delimitadores {}

Tabla 3 Objeto tipo *PUSH_ACK*

Bytes	**Función**
0	Versión del protocolo, 0x01 para código analizado.
1-2	Mismo *Token* aleatorio generado por el gateway en el objeto *PUSH_DATA*
3	Identificador del *PUSH_ACK*, 0x01

Tabla 4 Dato JSON del objeto *PUSH_DATA*

Campo	**Elementos**	**Descripción**
rxpk		Arreglo de objetos que contiene por lo menos un objeto json con la trama LoRa recibida y metadato asociado.
	time	Cadena. Tiempo en formato UTC ISO-8601 expandido
	tmst	Número. Marca de tiempo interno *RX_Finished* (32b no signado)

	freq	Número. Frecuencia central de la transmisión (flotante, in Hz)
	chan	Número. Canal “IF “usado por el Concentrador (entero no signado)
	rfch	Número. Cadena “RF” usado por el Concentrador (entero no signado)
	stat	Número. Estado del CRC, 1=OK, -1=Fallo, 0=No CRC
	modu	Cadena. Identificador de modulación (“LORA”, “FSK”).
	datr	Cadena. Identificador de *DataRate* ejemplo “SF12BW500”
	codr	Cadena. Identificador de corrección de error ECC LoRa
	rssi	Número. Valor de RSSI en dBm (entero signado, 1db)
	lsnr	Número. Valor SNR en dB(flotante, 0.1 dB)
	size	Número. Cantidad de bytes del Dato (entero signado)
	data	Cadena. Dato LoRaWAN en formato Base64. PHYPayload.
stat		Objeto que contiene el estado del Gateway.
	time	Cadena. Tiempo en formato UTC ISO-8691 expandido.
	lati	Número. Latitud GPS en grados (flotante, N es positivo)
	long	Número. Longitud GPS en grados (flotante, E es positivo)
	alti	Número. Altitud GPS en metros (entero)
	rxnb	Número. Numero de paquetes de RF recibidos (entero no signado).
	rxok	Número. Numero de paquetes de RF recibidos con CRC válido (entero no signado).
	rxfw	Número. Numero de paquetes de RF reenviados (entero no signado).
	ackr	Número. Porcentaje de datagramas de subidas reconocidos.
	dwnb	Número. Número de datagramas de bajada recibos (entero no signado).
	txnb	Número. Número de paquetes emitidos (entero no signado).

10.3.2 Paquete de datos de descarga tipo *PULL*

El *Gateway* debe mandar periódicamente una petición tipo *PULL_DATA* con el objetivo de mantener un canal de comunicación vivo (disponible) con el Servidor y que éste pueda enviar información de regreso. Por cada *PULL_DATA* enviado por el *gateway* el servidor responde con un *PULL_ACK*. La composición de los elementos de este tipo de datos se puede observar en la Tabla 5, Tabla 6, Tabla 7 y Tabla 8 (a partir de la siguiente página).

Una vez que el servidor pueda regresar información de regreso al *Gateway* lo hace realizando la transmisión de vuelta con el protocolo *PULL_RESP*.

Tabla 5 Objeto tipo *PULL_DATA*

Bytes Orden	Función
0	Versión del protocolo, 0x01 para código analizado.
1-2	*Token* aleatorio generado por el *gateway*.
3	Identificador del *PULL_DATA*, 0x02
4-11	Identificador único del *gateway* o dirección MAC

Tabla 6 Objeto tipo *PULL_ACK*

Bytes Orden	Función
0	Versión del protocolo, 0x01 para código analizado.
1-2	Mismo *Token* aleatorio generado por el *gateway* en *PULL_DATA*.
3	Identificador del *PULL_ACK*, 0x04

Tabla 7 Objeto tipo *PULL_RESP*

Bytes Orden	Función
0	Versión del protocolo, 0x01 para código analizado.
1-2	Sin usar
3	Identificador del *PULL_RESP*, 0x03

4-final	DATO_JSON. Dato transmitido en formato JSON conteniendo los corchetes delimitadores {}

Tabla 8 Dato JSON del objeto *PULL_RESP*

campo	Elementos	Descripción
txpk		Arreglo de objetos que contiene por lo menos un objeto json con el paquete RF recibido y metadato asociado.
	imme	Boleano. Envía el paquete inmediatamente. Igora tmst y time.
	tmst	Número. Enviar el paquete en cierto valor de *timestamp*. Ignora time
	time	Cadena. Enviar el paquete a cierto tiempo.
	freq	Número. Frecuencia central de la transmisión (Hz)
	chan	Número. Canal "IF "usado por el Concentrador
	rfch	Número. Cadena "RF" usado por el Concentrador
	powe	Número. Potencia de salida Tx en dBm (dBm)
	modu	Cadena. Identificador de modulación ("LORA", "FSK").
	datr	Cadena. Identificador de *DataRate* ejemplo "SF12BW500"
	codr	Cadena. Identificador de corrección de error ECC LoRa
	fdec	Número. Desviación frecuencia FSK (Hz)
	ipol	Boleano. Inversión de polaridad en modulación LoRa.
	prea	Número. Tamaño del Preámbulo RF
	size	Número. Tamaño en bytes del Dato
	data	Cadena. Dato LoRaWAN en formato Base64. Padeo es opcional.
	ncrc	Boleano. Si es cierto deshabilita CRC de la capa física (opcional)

10.4 Descripción de la Arquitectura Propuesta

Considerando los alcances, delimitaciones del presente trabajo y tras haber analizado la información recabada de las diferentes tecnologías, se han realizado

varias iteracciones incrementales sobre el diseño, se propone una arquitectura de software que permita configurar un sistema LoRaWAN como el Sistema en Cuestión del Espacio de Direcciones de un Servidor OPC-UA. Utilizar la extensión Pub/Sub para publicar la información siguiendo un patrón publicador-susbcritor *middleware* MOM mediante un Intermediario de cola de mensajes MQTT. Los datos LoRaWAN son consumidos por elementos subscritos al Intermediario Pub/Sub que alimentan al Servidor LoRaWAN y su correspondiente capa de Aplicación.

Esta manera permite que el *Gateway* cuente con Servidor OPC-UA que potencialmente puede ser utilizado por cualquier Cliente OPC-UA localmente y cualquier Cliente OPC-UA Pub/Sub de forma remota.

Siguiendo las buenas prácticas de la ingeniería de software es necesario realizar un diseño arquitectónico que muestre el funcionamiento del sistema describiéndolo desde diferentes perspectivas a través de varios niveles de granularidad.

Casos de Uso. Comenzando con la descripción del sistema desde el punto de vista de los actores involucrados y las acciones del sistema con las que pueden interactuar; los Nodos LoRaWAN pueden publicar y consumir información a través de peticiones *Uplink* y *Downlink* respectivamente. La información de los Nodos del espacio de direcciones está disponible para proveer servicios OPC-UA hacia Clientes OPC-UA para modos cliente-servidor y publicador-subscriptor. Finalmente el sistema puede obtener metadatos de los nodos LoRaWAN mediante peticiones al Servidor LoRaWAN, este caso, sin embargo por cuestión de tiempo y simplicidad, no se implementará en el prototipo actual pero es viable por los servicios REST y gRPC de los servidores LoRaWAN, los metadatos estarán simulados directamente en el prototipo de pruebas. El diagrama de casos de uso se puede ver en la Figura 13.

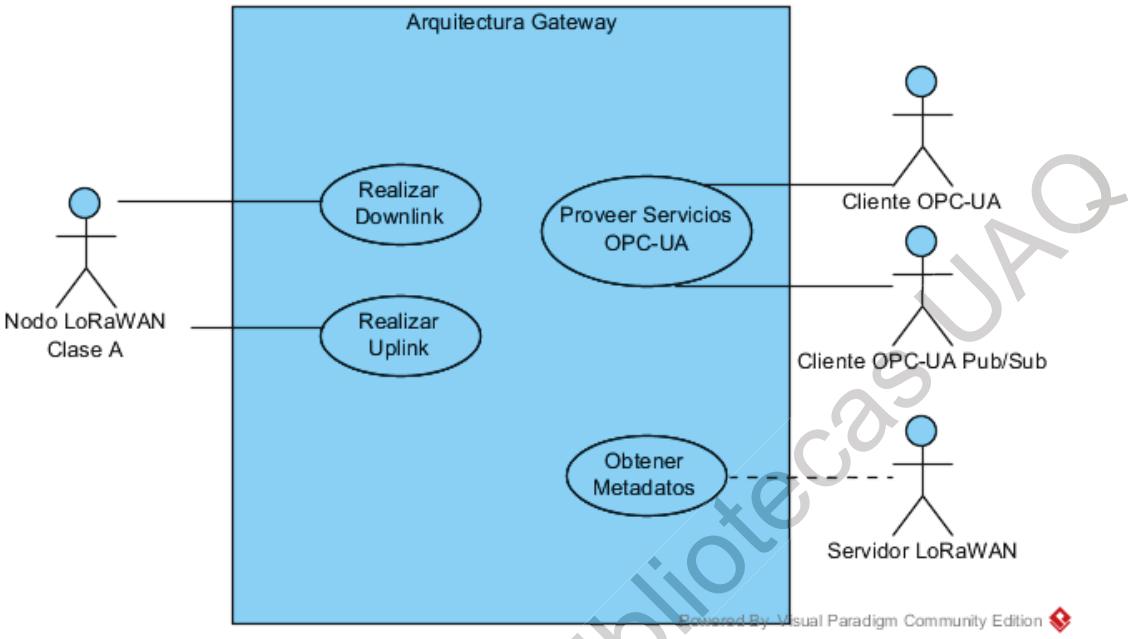

Figura 13 Casos de Uso

Arquitectura de software (componentes). La arquitectura propuesta está basado en un *middleware* MOM MQTT que permite dotar de un patrón de intercambio de mensajes tipo publicador-subscriptor. Por una parte, el Publicador compuesto por el *gateway* LoRaWAN, su intérprete y puente del espacio de direcciones de un servidor OPC-UA que implementa la extensión Pub/Sub para interactuar con el *middleware* MOM; en el caso del Subscriptor la información publicada se consume mediante la extensión Pub/Sub o cliente MQTT que permite utilizar la información tanto en cliente OPC-UA como devuelta al sistema LoRaWAN, mismo que continua con el flujo de datos hacia el modelo de negocio requerido (esto para soportar sistemas legados existentes). La información disponible en OPC-UA mediante Clientes permitirían su uso en modelos de negocios basados en OPC-UA tanto CPS como Web. Consulte en la Figura 14.

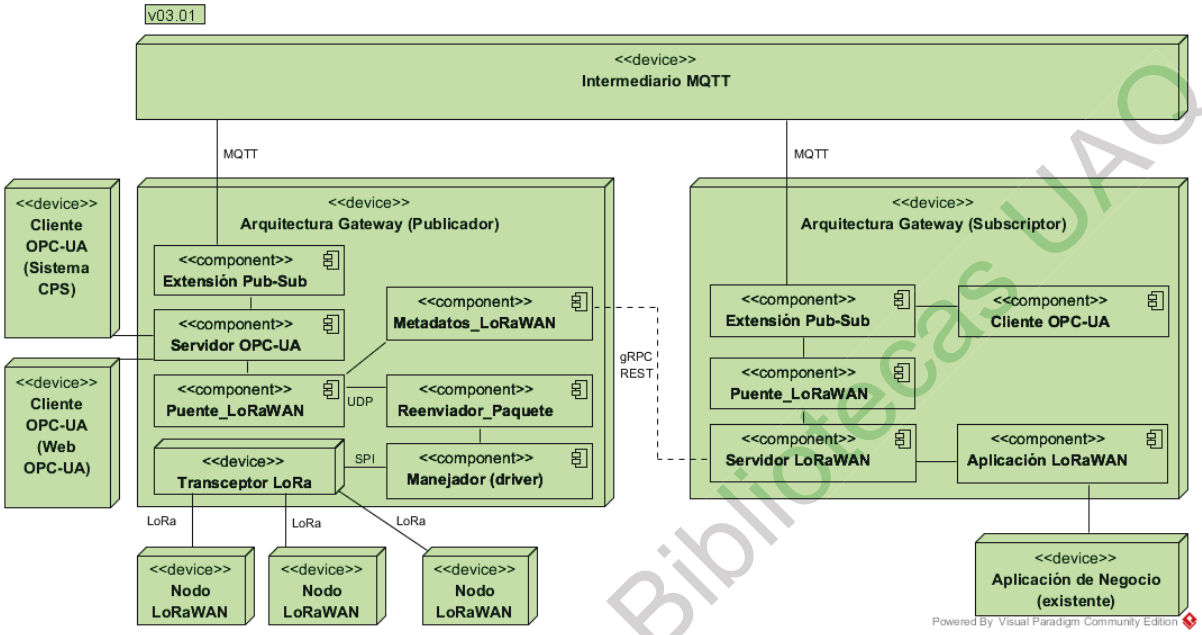

Figura 14 Arquitectura propuesta

La arquitectura propuesta permite la interoperabilidad entre LoRaWAN y sistemas ciber físicos a través de OPC-UA y se compone de los siguientes elementos:

- Nodos LoRaWAN. Son los sensores o actuadores que incorporan un radio transceptor LoRaWAN para publicar o consumir información mediante modulación LoRa.
- Tranceptor LoRA. Es el dispositivo que permite decodificar múltiples paquetes LoRa a modo de *gateway* físico. Su función únicamente es la de concentrar y reenviar la información a capas más altas y seguras. Realiza únicamente la interpretación a nivel de capa física del protocolo LoRa.

- Manejador LoRa. (Librería del fabricante) Permite la obtención de información del Radio LoRa mediante la implementación de los comandos específicos del hardware.

- Reenviador de Paquetes LoRa. Es un módulo de software típico en los *gateways* físicos LoRaWAN y permiten el reenvío de los paquetes hacia los servidores de red para la interpretación del protocolo LoRaWAN hacia capas más altas implementando comunicación basada en trasacciones.

- Puente LoRaWAN. Es un componente de software que permite la adaptación y extensibilidad de metadatos del protocolo para poder ser transformado al modelo de información de OPC-UA. Además el puente permite la decodificación del mensaje LoRa depediendo de la versión del protocolo en tiempo de ejecucción.

- Conector Meta-datos LoRaWAN. Es un componente de software que puede extraer información del Servidor LoRaWAN y hacerlo visible en el Espacio de Direcciones del Servidor OPC-UA.

- Cliente y Servidor OPC-UA. Provee de un modelo estandarizado de modelo de información en el que el sistema LoRaWAN es accesible a través del Espacio de Direcciones OPC-UA.

- Extensión OPC-UA Pub/Sub. Añade la funcionalidad del modelo de *Middleware* Orientado a Mensajes sobre el servidor OPC-UA.

- Intermediario Pub/Sub. Es el *middleware* orientado a mensajes que permite el desacoplamiento y la interoperabilidad entre diferentes actores mediante colas de mensajes tipo publicador-subscriptor.

- Servidor de Red y Aplicación LoRaWAN. Son elementos de la distribución de la tecnología LoRaWAN que gestionan, filtran y aplican las reglas del negocio de la información de los Nodos.

Arquitectura de aplicación (despliegue). De acuerdo al diseño de experimento los componentes están distribuidos de acuerdo a una aplicación típica de los subsistemas OPC-UA, un modelo de intercambio de datos basado en mensajes tipo *Middleware* MOM y la estructura de la tecnología LoRaWAN. El diagrama de despliegue se puede observar en la Figura 15.

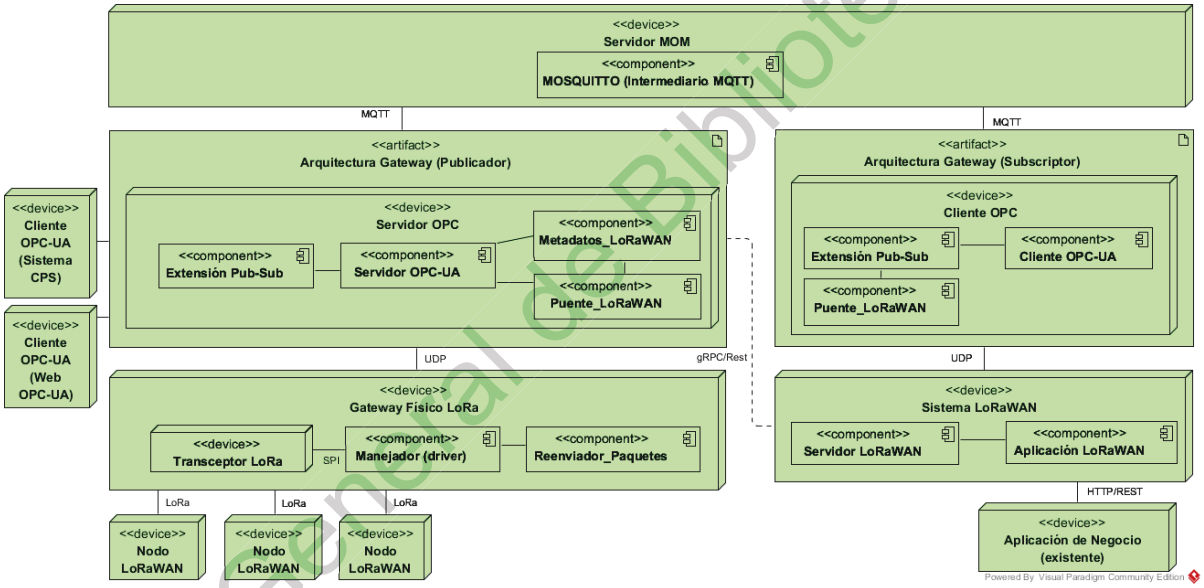

Figura 15 Despliegue de la arquitectura propuesta

Mediante el uso de la arquitectura propuesta se contempla la ejecución del modelo de información en cuatro diferentes dominios: 1) el dominio LoRa en el cuál el sistema de Nodos y el Gateway LoRa funcionan de manera normal. 2) El dominio OPC-UA en el cuál el sistema LoRaWAN es modelado en el Espacio de Direcciones. 3) El dominio de los Sistemas Ciber Físicos o Industria 4.0 en el que la información es distribuida a un proceso inteligente que permite adaptar su salida dependiendo de la información recabada. Y 4) El dominio *IoT* mediante el paradigma Pub/Sub

permitiendo el uso de la información LoRa en aplicaciones masivas con beneficio a la sociedad. Los dominios se pueden observar en la Figura 16.

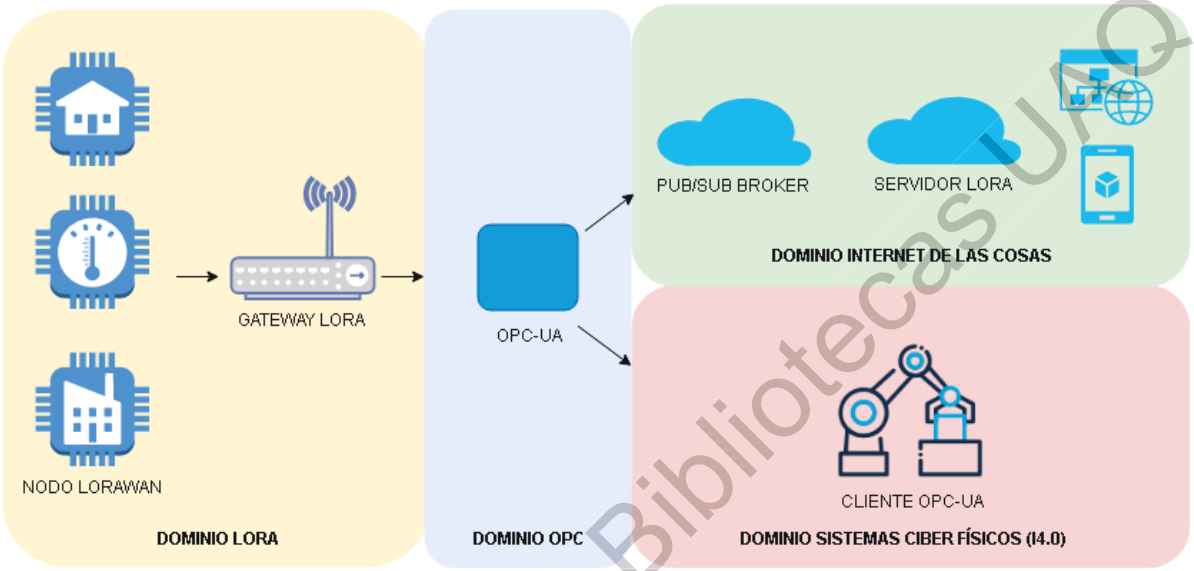

Figura 16 Dominios de la arquitectura propuesta.

11. DISEÑO DE EXPERIMENTO

Para realizar la evaluación de la arquitectura propuesta es necesario construir un sistema que cuente con los elementos tecnológicos planteados en la hipótesis, tanto OPC-UA como LoRaWAN en un ambiente para desarrollo de Internet de las Cosas como se ha presentado en la arquitectura propuesta en el capítulo anterior.

Para realizar el prototipo fue posible acceder a los siguientes recursos de hardware:

- *Gateway* LoRaWAN RK831 que integra un chip Semtech SX1331.
- Nodos TTN que integran un chip Microchip RN2903.
- Raspberry PI para alojar el *Gateway* y los Servidores LoRaWAN.
- Equipo de cómputo Intel Core i5 para alojar el Servidor OPC-UA, Clientes OPC-UA y Intermediario MOM.

Se utilizaron diversos recursos de software libre que permiten implementar algunos de los componentes de la arquitectura propuesta:

- Librería de código abierto OPC .NET Standard version.
    - Repositorio: github.com/OPCFoundation/UA-.NETStandard.git
    - Licencia: GPL 2.0 y EPL 1.0.
- Librería de código abierto Loraserver.io LoRa Server.
    - Repositorio: github.com/brocaar/loraserver.git
    - Licencia: MIT
- Librería de código abierto Loraserver.io LoRa App Server.
    - Repositorio: github.com/brocaar/lora-app-server.git
    - Licencia: MIT

- Librería de código abierto Loraserver.io LoRa Gateway Bridge.
    - Repositorio: github.com/brocaar/lora-gateway-bridge.git
    - Licencia: MIT
- Manejador Gateway LoRa
    - Repositorio: github.com/ttn-zh/ic880a-gateway.git
    - Licencia: GPL3.0
- Librería de código abierto para *gateway* físico TTN-ZH Packet Forwarder
    - Repositorio: github.com/ttn-zh/packet_forwarder.git
    - Licencia: MIT Modificada
- Librería de código abierto para nodos LoRaWAN TTN Nodes
    - Repositorio: github.com/TheThingsNetwork/arduino-device-lib.git
    - Licencia: MIT
- Intermediario (middleware MOM) Mosquitto.
    - Repositorio: github.com/eclipse/mosquitto.git
    - Licencia: EPL1.0 y EDL1.0
- Cliente Mosquitto para .Net.
    - Repositorio: github.com/eclipse/paho.mqtt.m2mqtt.git
    - Licencia: EPL1.0.

11.1 Configuración *Gateway* RK831 en ambiente Linux Raspbian

Se debe utilizar el paquete Manejador *Gateway* LoRa para realizar la instalación del *gateway* RK831 en una tarjeta Raspberry PI3 siguiendo los pasos que a continuación se muestran:

```
git clone -b spi https://github.com/ttn-zh/ic880a-gateway.git
~/ic880a-gateway
cd ~/ic880a-gateway
sudo ./install.sh spi
Elegir N (no) en "remote configuration"
scp /media/sf_wk/06_WK/git/packet_fwd/global_conf.json
pi@192.168.1.87:tmp/
scp /media/sf_wk/06_WK/git/packet_fwd/local_conf.json
pi@192.168.1.87:tmp/
Luego copiar de ~/tmp a /opt/ttn-gateway/bin/
```

En donde pi es el nombre de usuario y 192.168.1.87 es la dirección del RPI en la red local (deberá ser reemplazado por la dirección al momento de la instalación).

La rutina de comandos "install.sh spi" instala los paquetes de software necesarios para utilizar el transceptor SX1331 mediante una comunicación serial SPI directamente del bus de periféricos de la Raspberry. Además installa los paquetes de software para el módulo Reenviador de Paquetes cuya configuración de red, autenticación y parámetros está disponible en "global_conf.json" y "local_conf.json".

11.2 Instalación de servidor LoRa Server en ambiente Linux Raspbian

El proyecto "LoRaServer.io" está compuesto de tres paquetes: (i) LoRa *Server* que implementa los servicios de un servidor LoRaWAN típico; (ii) LoRa App *Server* mediante una interfaz web sobre los servicios de LoRa Server; y (iii) LoRa Gateway *Bridge* una capa de abstracción que permite recibir los datos desde cualquier *Gateway* LoRa mediante MQTT.

Para la instalación de LoRaServer.io en una RPI3 con sistema operativo Raspbian es necesario realizar la siguiente instalación:

```
sudo apt install mosquitto
sudo apt install mosquitto-client
sudo apt install redis-server
sudo apt install redis-tools
sudo apt install postgresql
sudo apt install postgresql-contrib
#setup postgress db
sudo apt install dirmngr
sudo apt install apt-transport-https
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
1CE2AFD36DBCCA00
sudo echo "deb
https://artifacts.loraserver.io/packages/2.x/deb stable main"
| sudo tee /etc/apt/sources.list.d/loraserver.list
sudo apt update
sudo apt install lora-gateway-bridge
sudo apt install loraserver
sudo apt install lora-app-server
```

Se instalan los tres paquetes del proyecto además de los sub-paquetes de software necesarios para el funcionamiento de la base de datos y el cliente para conectarse manejador de cola de mensajes Mosquitto. Si la instalación se ejecutó de manera correcta se podrá acceder al puerto 8080 de la red local y ver el servidor mediante un navegador web como se observa en la Figura 17.

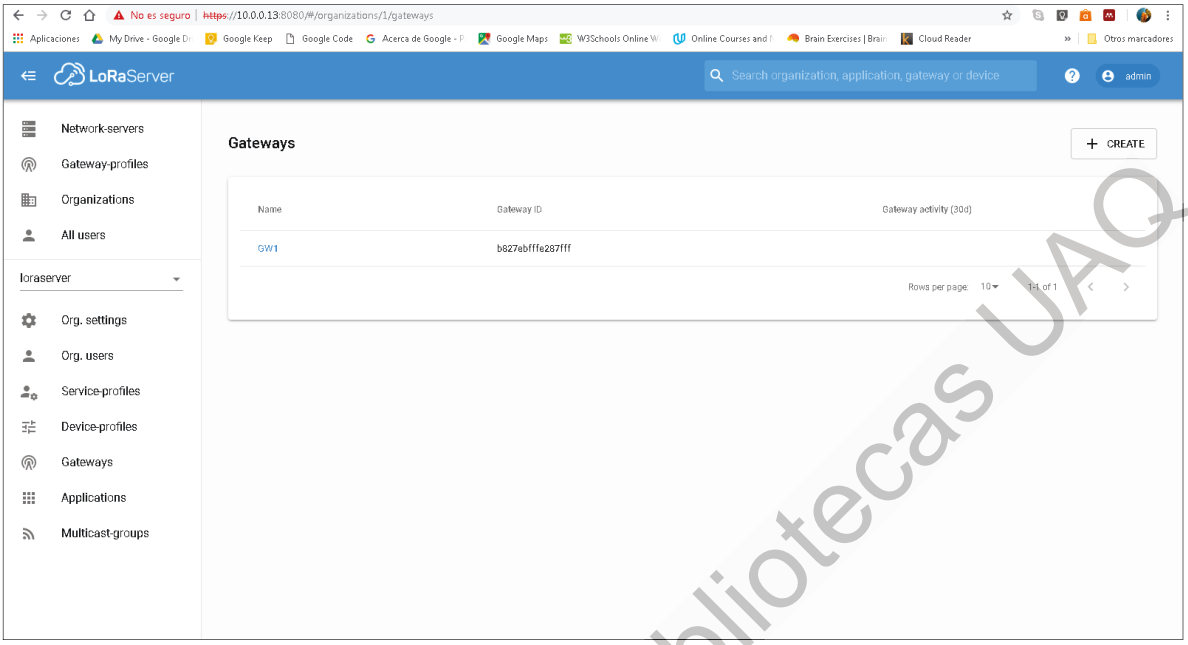

Figura 17 LoRa App Server

Como parte de la configuración del Servidor LoRaWAN se deberá administrar el sitio asignando perfiles de usuario, nombre de *Gateways*, tipo de Nodos y la aplicación en donde se agregan los Nodos. Ejemplos se pueden ver en la Figura 18, Figura 19 y Figura 20.

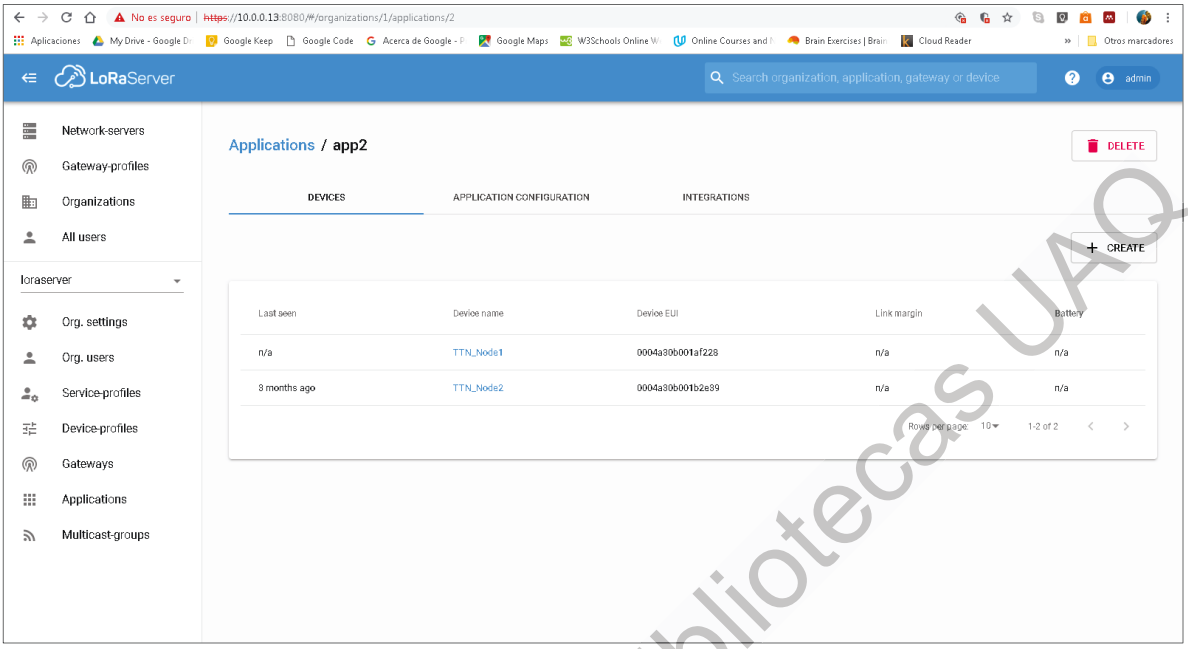

Figura 18 Configuración de Aplicación LoRaWAN

Figura 19 Alta de Nodos LoRaWAN

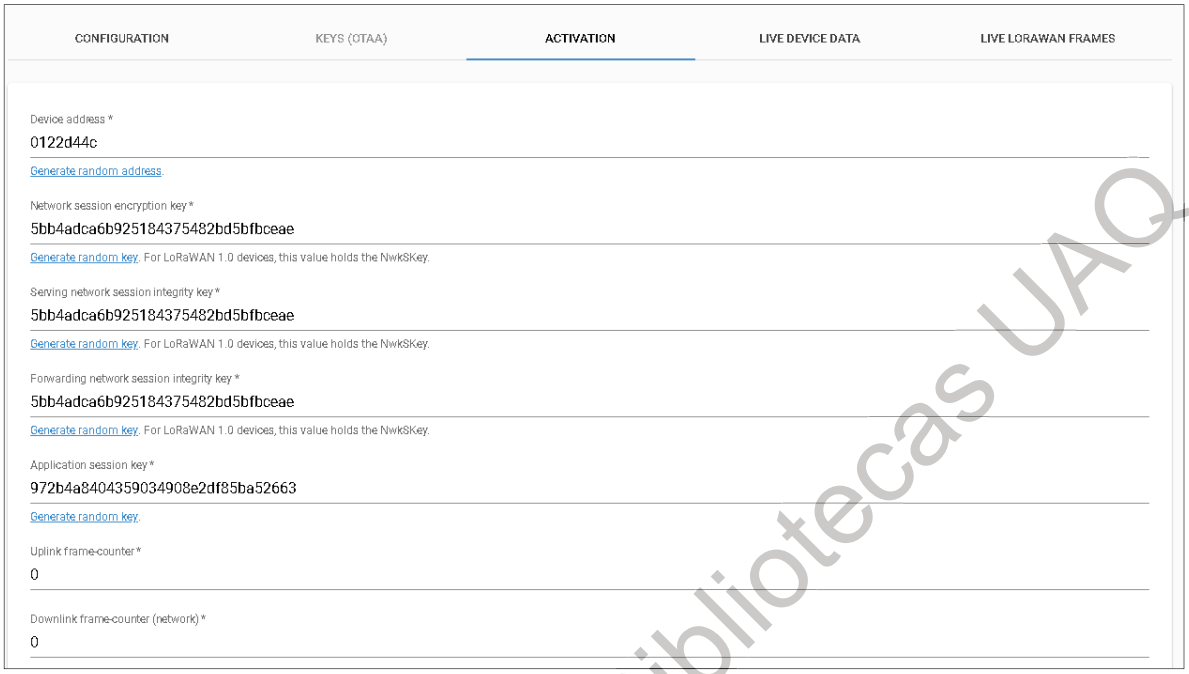

Figura 20 Configuración Típica de los Nodos LoRaWAN (*ABP*)

11.3 Programación de Nodos LoRaWAN

El funcionamiento del Sistema LoRaWAN requiere de la configuración de los Nodos. Es posible la configuración remota (*OTA*) en la que el Nodo y el Servidor intercambian una serie de mensajes para adquirir llaves de sesión o mediante una configuración pre-cargada (*ABP*) en el cual el Servidor y el Nodo son configurados con un conjunto de llaves. Para el desarrollo de este experimento se ha elegido la configuración *ABP*. Los Nodos se han dado de alta en el Servidor LoRaWAN, como se muestró en la sección anterior; ahora se deberá configurar cada dispositivo con las mismas llaves provistas en el pérfil *ABP* y reprogramar cada unidad.

Para realizar la actualización de los Nodos es necesario utilizar la Librería código abierto de nodos LoRaWAN TTN Nodes tal como se ejemplifica en la Figura 21 y Figura 22 (a partir de la siguiente página).

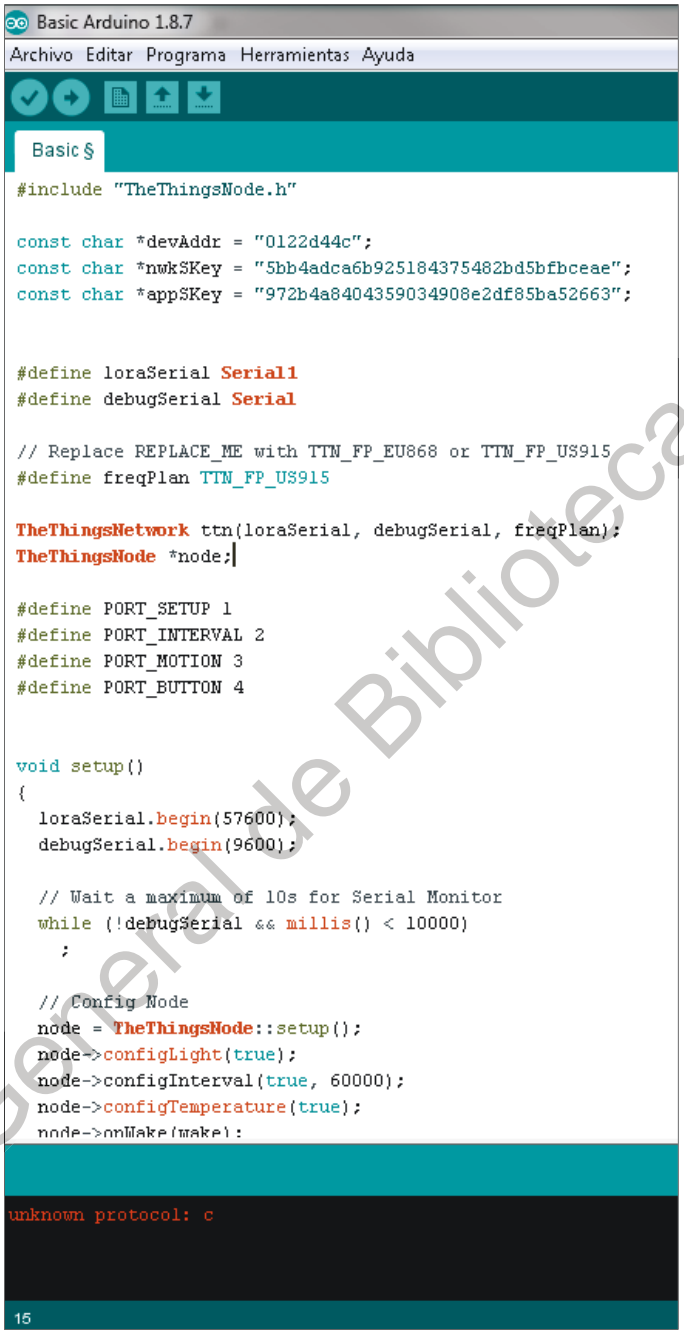

Figura 21 Reprogramación de Nodos LoRaWAN

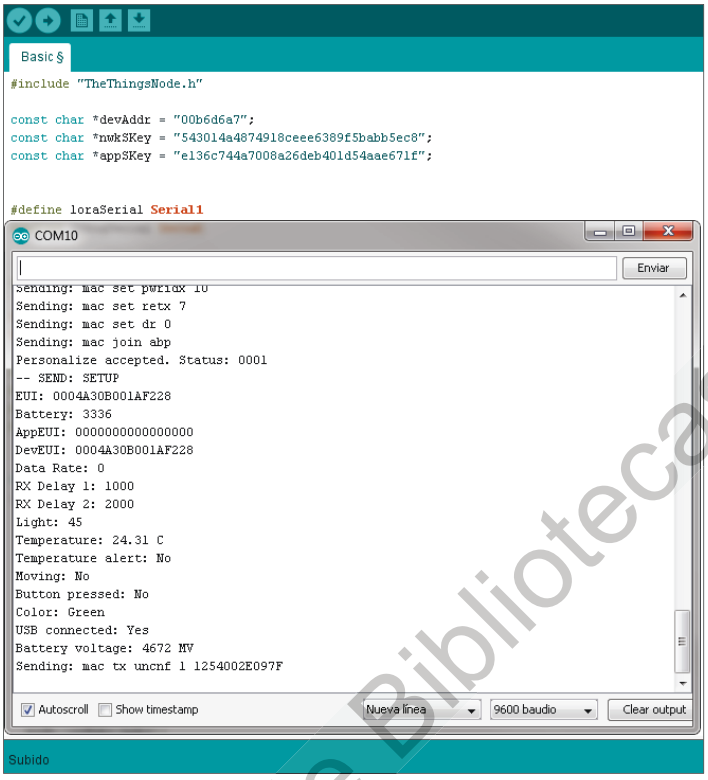

Figura 22 Comprobación del funcionamiento Nodo LoRaWAN.

11.4 Interacción entre paquetes

Una vez configurados de los distintos elementos se deberá poner en marcha cada uno de ellos en el siguiente orden:

- Iniciar el controlador *Gateway* LoRa.
- Iniciar Intermediario Pub/Sub (*Broker*) Mosquitto.
- Iniciar el Servidor OPC-UA
- Iniciar la Extensión OPC-UA Pub.
- Iniciar la Extensión OPC-UA Sub.
- Iniciar el Servidor LoRaWAN.
- Iniciar Clientes OPC-UA.

- Activar Nodos LoRaWAN.

Para el experimento se realizó un programa de arranque que realiza la ejecución automatizada de estas tareas. Tanto el programa como la ejecución de cada uno se puede observar en la Figura 23, Figura 24, Figura 25, Figura 26, Figura 27, Figura 28 y Figura 29 (a partir de la siguiente página).

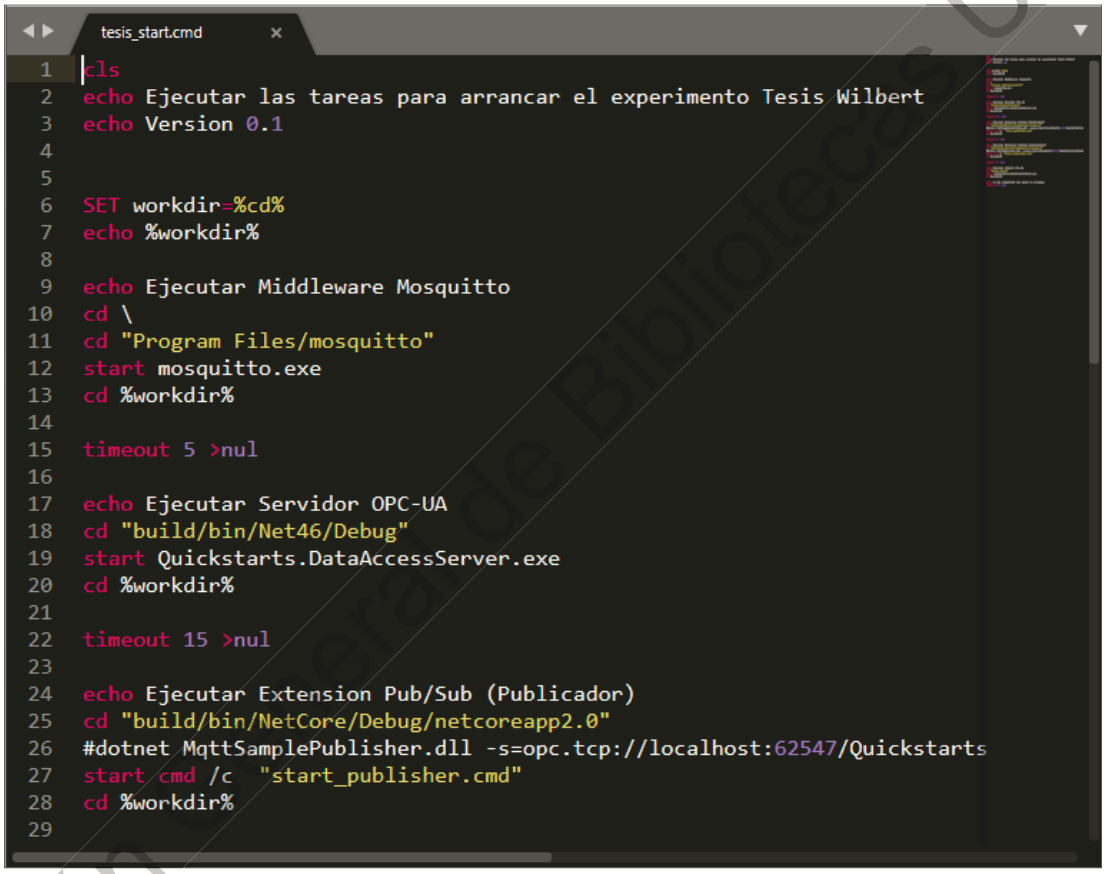

Figura 23 Script para automatizar arranque de tareas

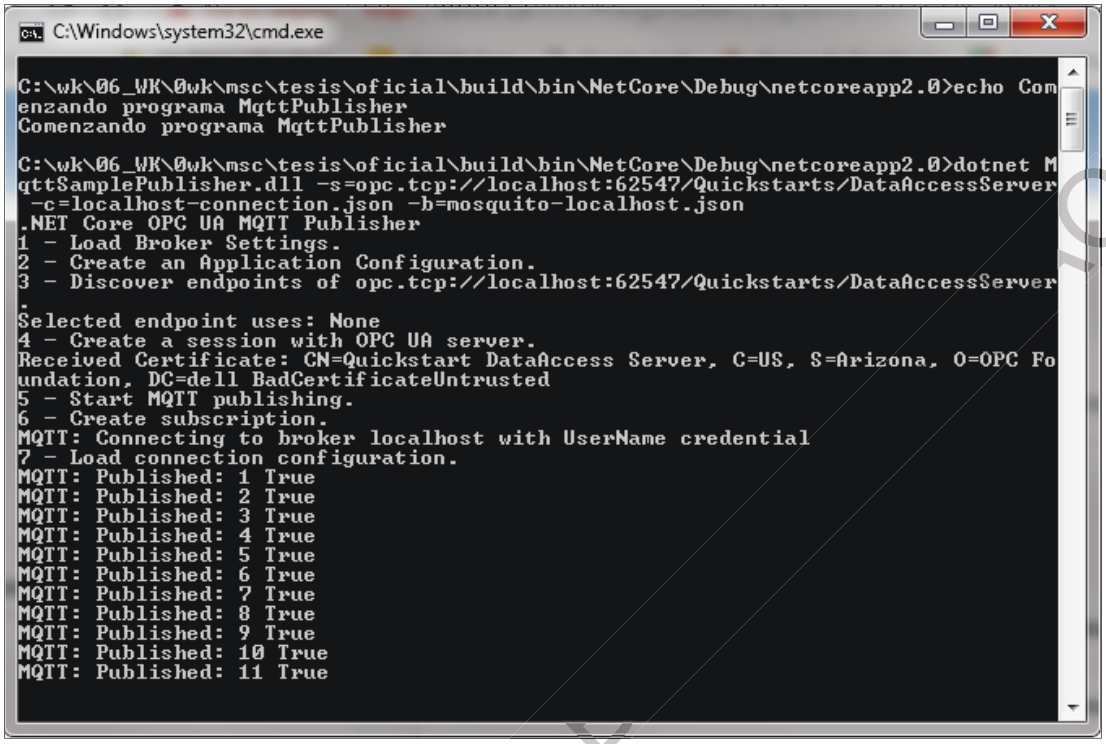

Figura 24 Extensión Pub/Sub Publicador

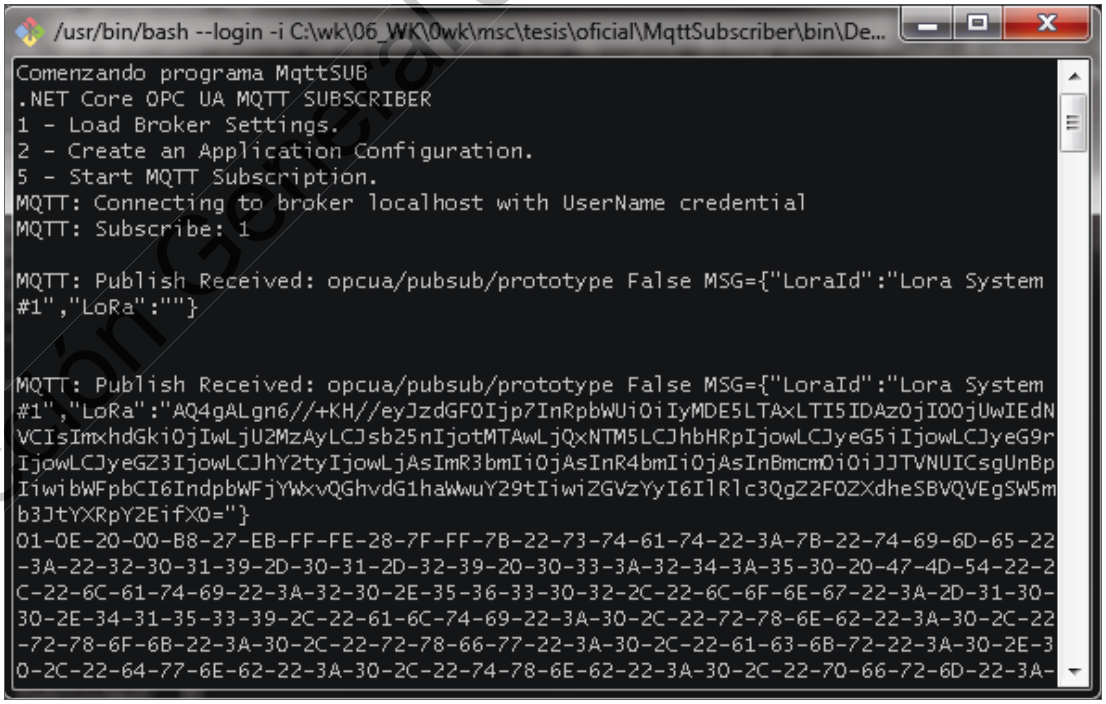

Figura 25 Extensión Pub/Sub Subscriptor

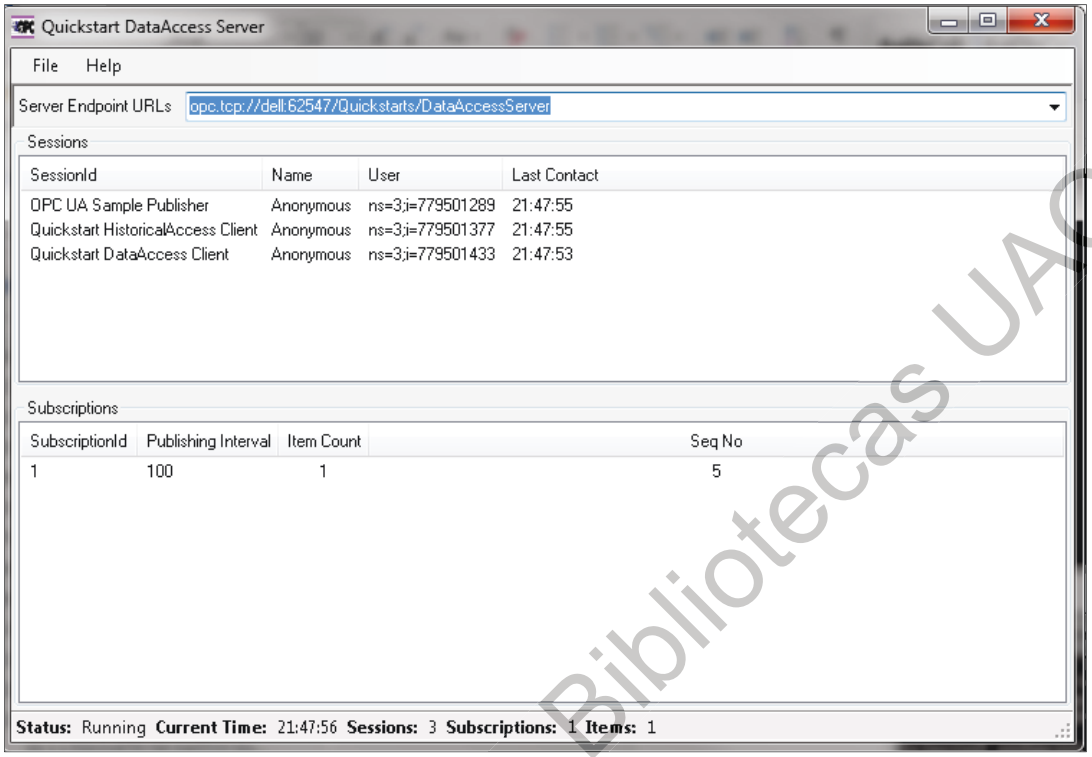

Figura 26 Servidor OPC-UA

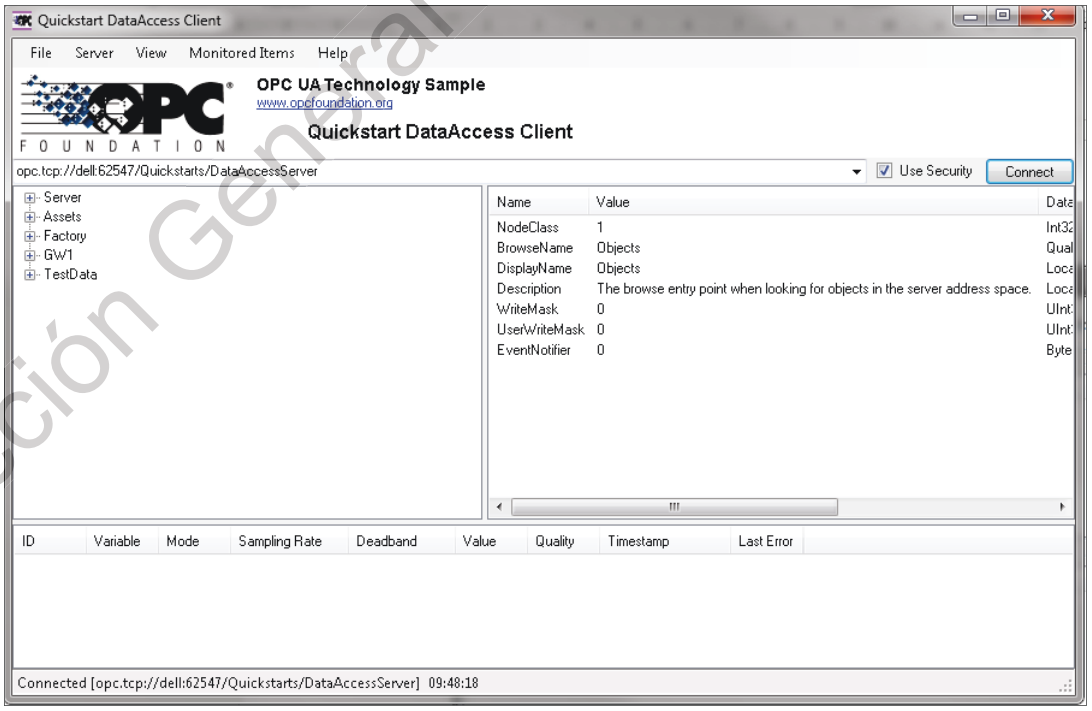

Figura 27 Cliente OPC-UA

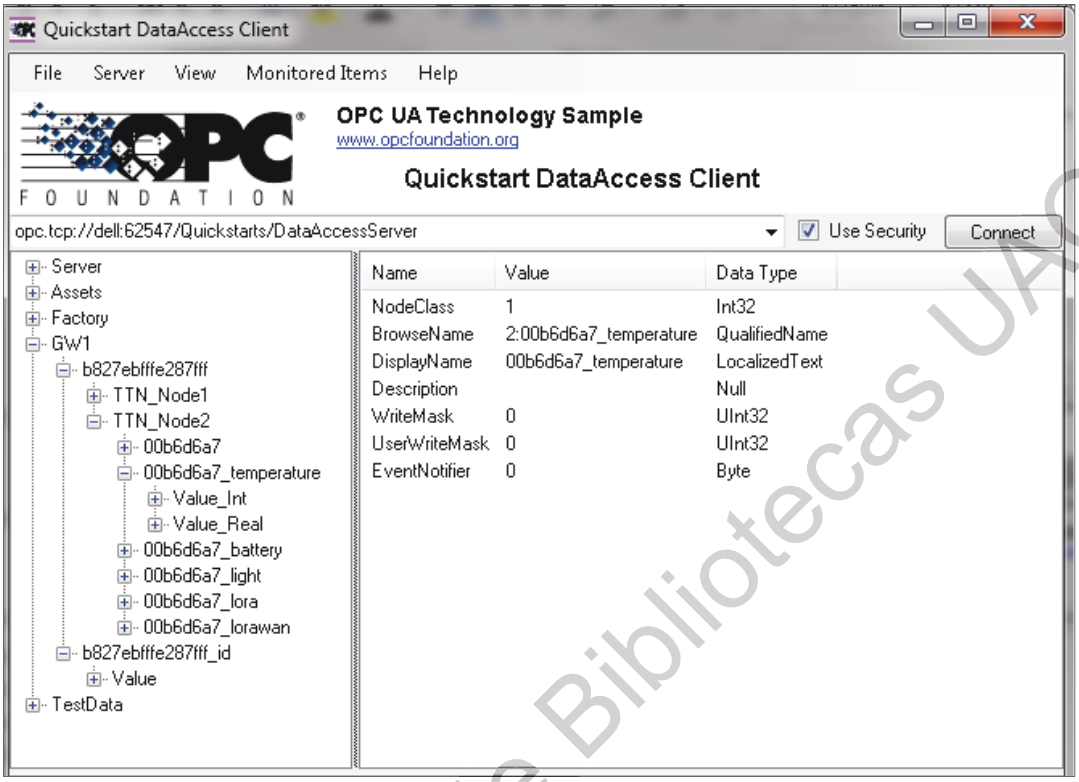

Figura 28 Espacio de Direcciones del Sistema LoRaWAN

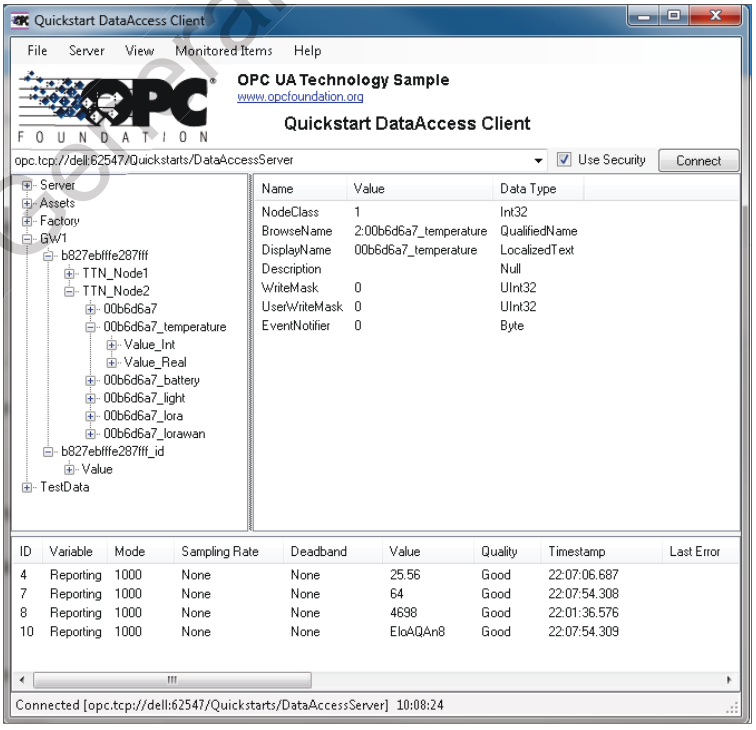

Figura 29 Servicio de Acceso de Dato (DA) en tiempo real

Cuando los Nodos LoRaWAN se encuentran transmitiendo es posible acceder a su información tanto mediante el sistema OPC-UA, o mediante el servidor LoRaWAN como se observa en la Figura 30 y Figura 31.

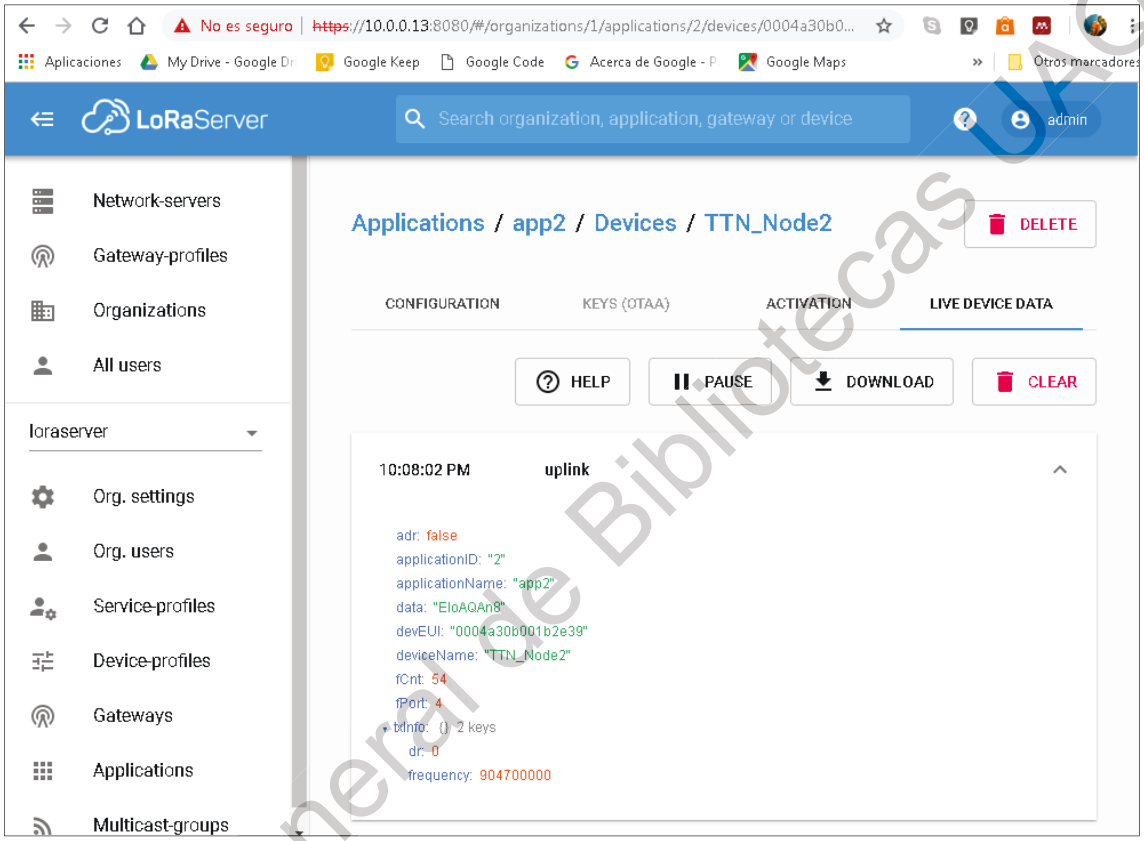

Figura 30 Recepción de dato en el Servidor LoRaWAN

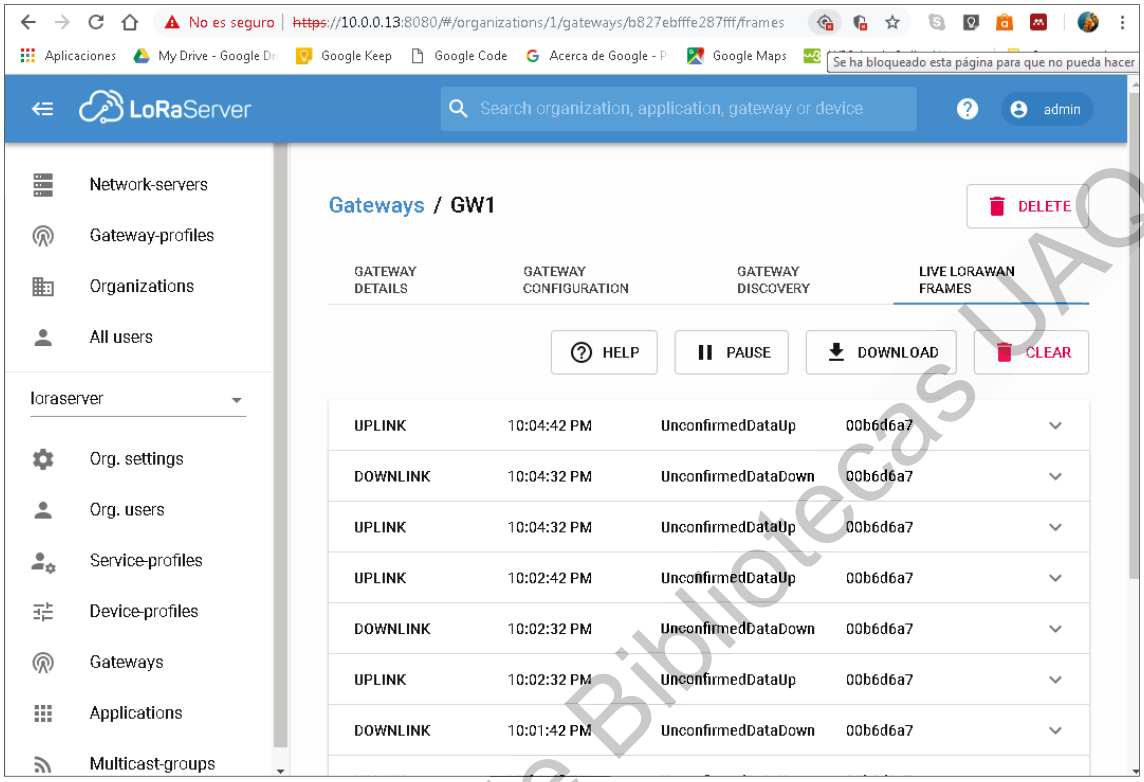

Figura 31 Recepción de mensajes en el Sistema LoRaWAN

Con el propósito de mostrar físicamente el aspecto del Nodo y el *Gateway* Físico utilizados se han incluido y se pueden observar en la Figura 32 y Figura 33.

Figura 32 Nodo LoRaWAN TTN.

Figura 33 Gateway LoRa RAK Wireless.

11.5 Evaluación de la arquitectura propuesta

Para determinar la robustez de la arquitectura propuesta, bajo la delimitación impuesta por el diseño de experimento, se realizaron pruebas de carga sobre los diversos módulos de software.

Se utilizó la herramienta denominada JMeter de Apache la cual está disponible gratuitamente mediante Licencia Apache 2.0 por la Fundación de Software Apache.

El escenario de prueba de JMeter simula la entrada de datagramas UDP desde la capa física de un *gateway* LoRa, la cantidad de datagramas simulados se fijó a diez mil y la unidad que se varió durante las diversas corridas fue el retardo entre el envío de cada mensaje. En JMeter se logró mediante la configuración de un objeto *Thread Group*, un objeto *Request UDP* y un objeto tipo *Constant Timer*, para mayor detalles y ejemplificación de uso vea la Figura 34, Figura 35, Figura 36 y Figura 37.

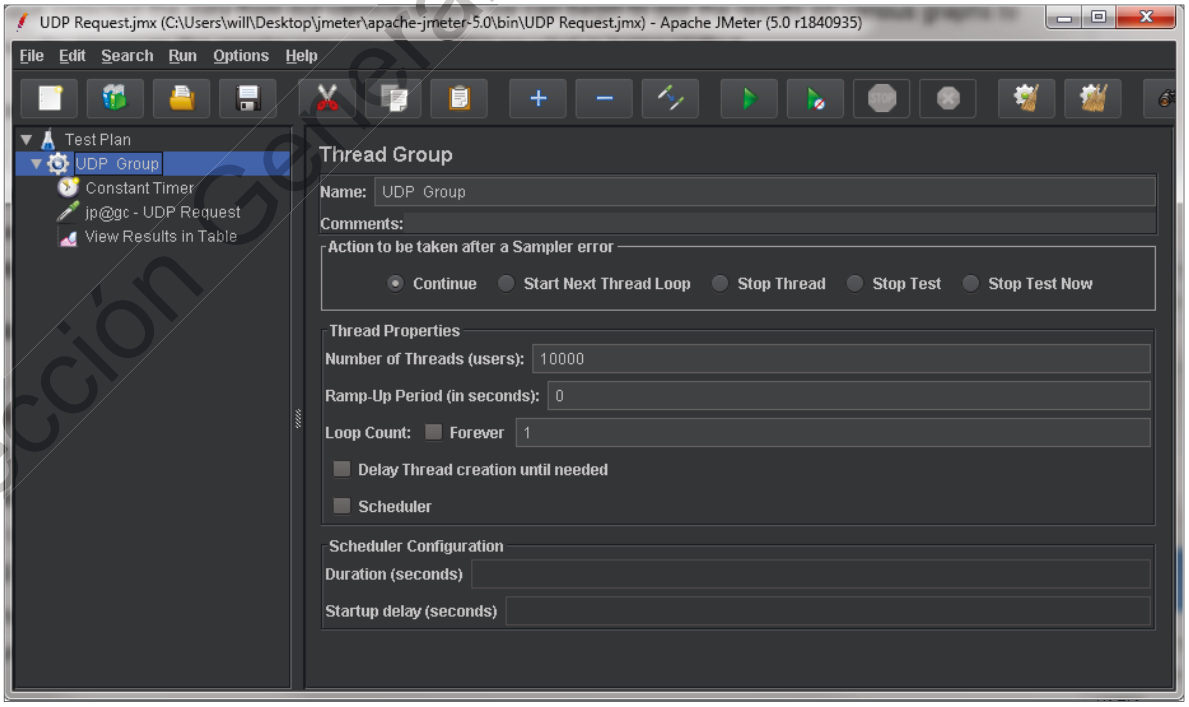

Figura 34 Configuración de Thread Group de JMeter

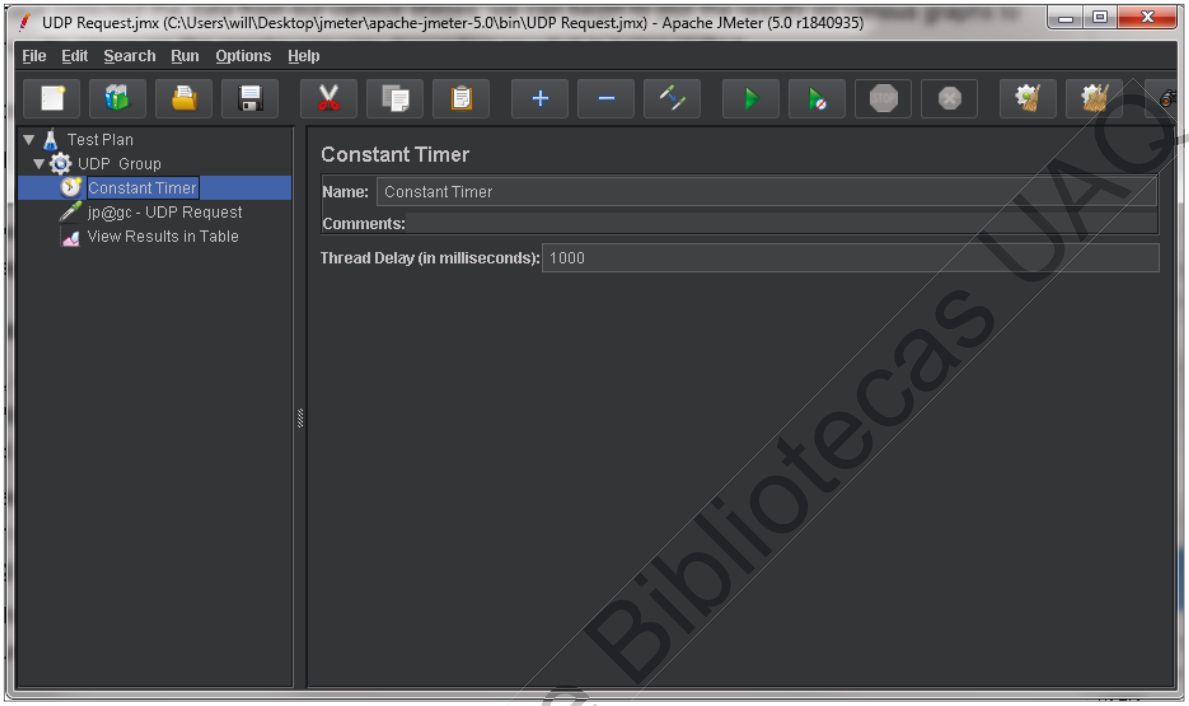

Figura 35 Configuración del retardo en milisegundos.

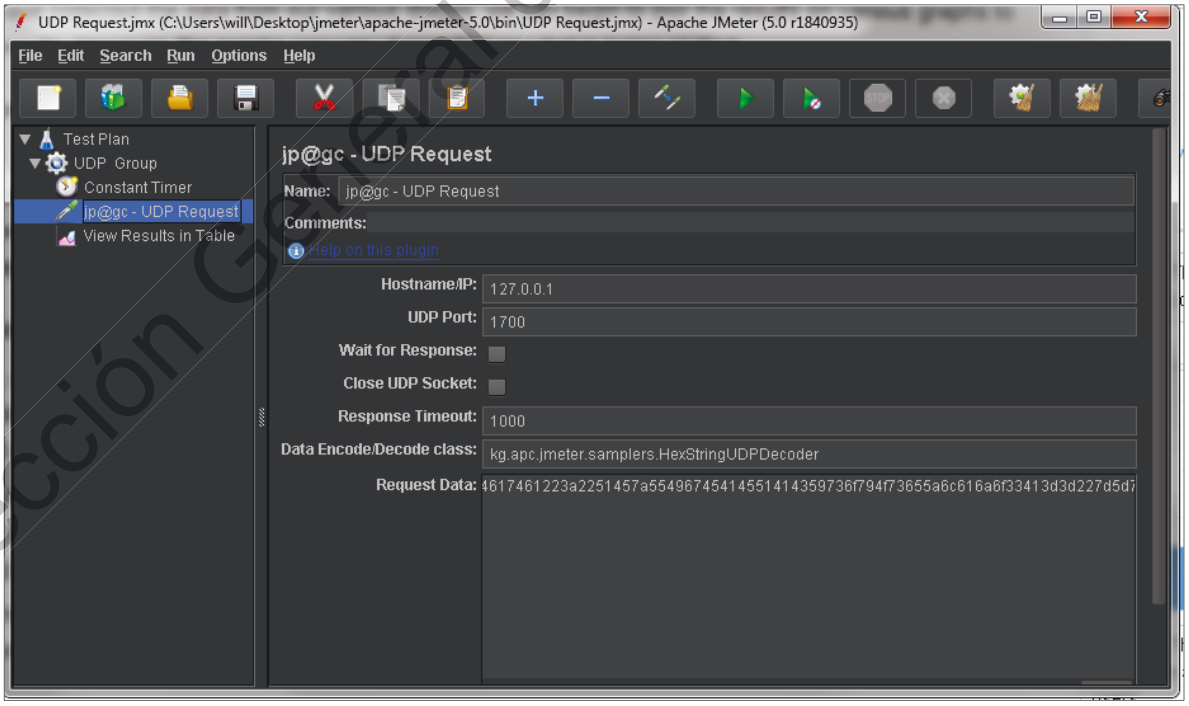

Figura 36 Configuración del objeto UDP Request con dato simulado de la capa física.

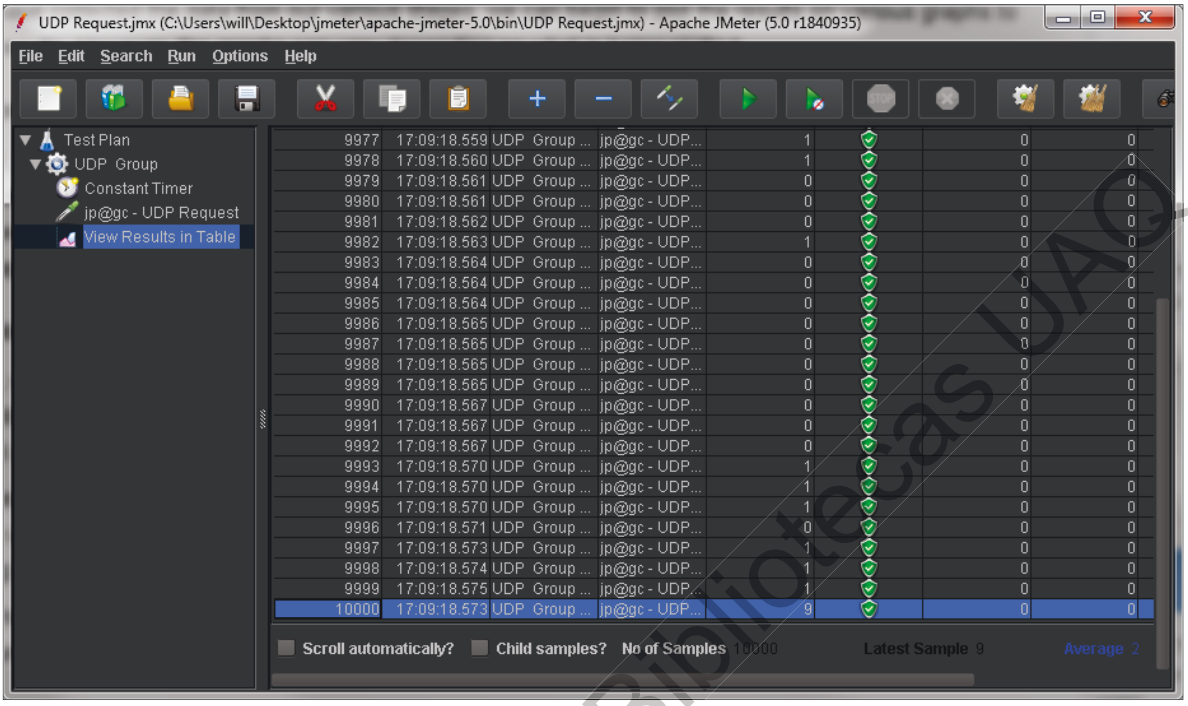

Figura 37 Visor de resultados de los mensajes simulados por JMeter.

11.5.1 Prueba de carga sobre el módulo Puente LoRaWAN

El objetivo de esta prueba es determinar la robustez del módulo Puente LoRaWAN el cual se encarga de leer cada datagrama recibido desde el Reenviador de Paquetes UDP mismo que contiene la información recibida por la capa física del gateway LoRa. La función del Puente LoRaWAN es leer el datagrama en el puerto configurado, decodificar el paquete LoRa, descifrar cada sub-paquete LoRaWAN y publicar el valor hacia el Servidor OPC. La prueba en ejecución se puede ver en la Figura 38.

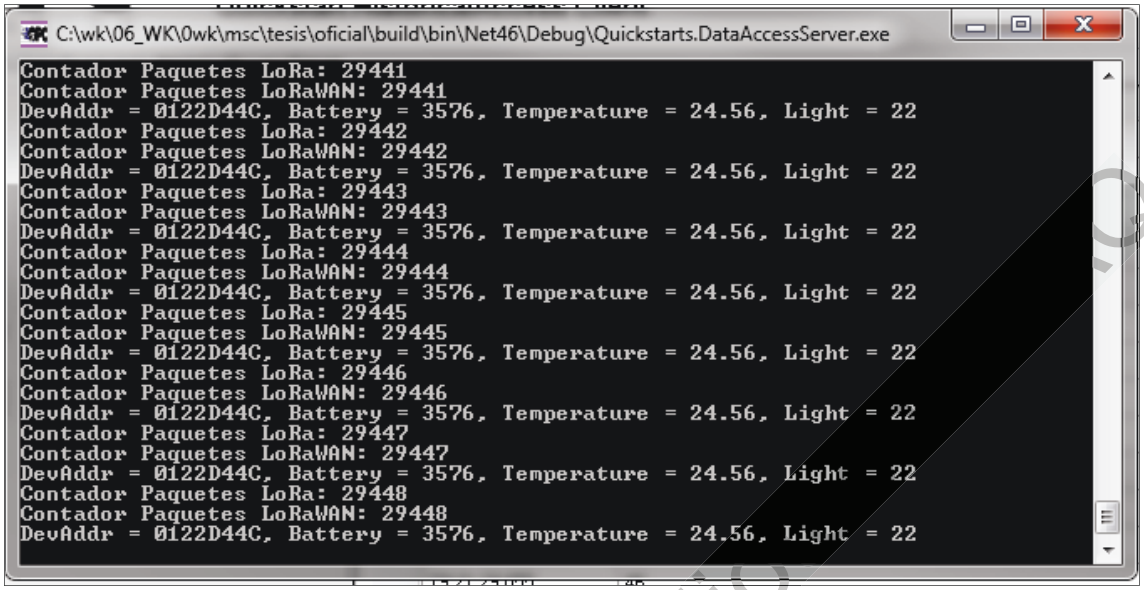

Figura 38 Publicación de contador de paquetes del Puente LoRaWAN.

11.5.2 Prueba de carga sobre el módulo Servidor OPC

Cada paquete de dato recibido desde el Puente LoRaWAN es ingresado al Espacio de Direcciones del Servidor OPC, el objetivo de esta prueba es conocer la capacidad del Servidor OPC de interpretar cada paquete y utilizarlo en su Espacio de Direcciones. Para realizar esta prueba se utilizó un Cliente de Datos Históricos de OPC actuando sobre el periodo de prueba en modo Contador. La prueba en ejecución se puede ver en la Figura 39.

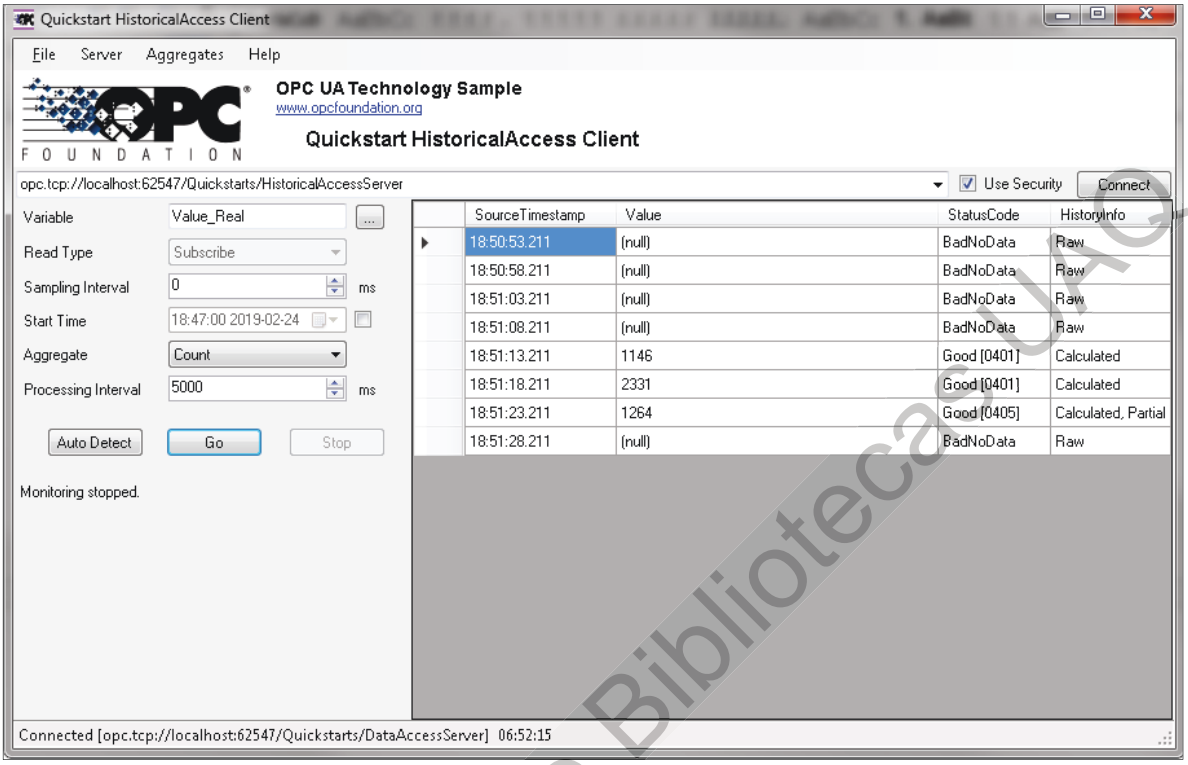

Figura 39 Cliente OPC de datos Históricos con un contador de paquetes.

11.5.3 Prueba de carga sobre el módulo MQTT Sub

El objetivo de esta prueba es determinar la confiabilidad del Intermediario Mosquitto, ya que cada nuevo paquete recibido en el Servidor OPC es publicado al Intermediario y posteriormente consumido por el paquete MQTT Sub. La prueba en ejecución se puede ver en la Figura 40.

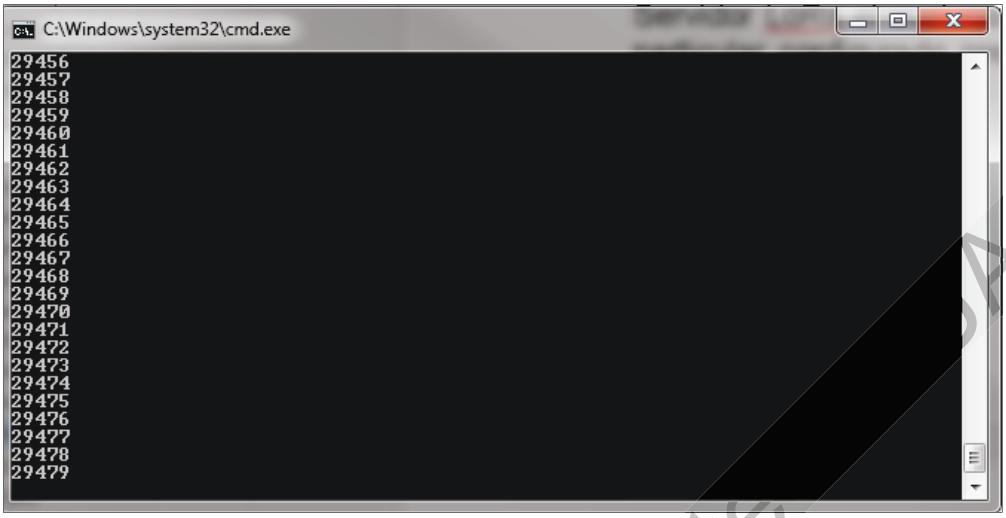

Figura 40 Contador de mensajes consumidos desde Mosquitto.

11.5.4 Prueba de carga sobre el módulo Servidor LoRa.IO

Finalmente cada dato consumido por el paquete MQTT Sub es utilizado por el Servidor LoRa el cual muestra cada nuevo dato sobre un Nodo LoRaWAN en particular configurado previamente. La prueba en ejecución se puede ver en la Figura 41 y Figura 42.

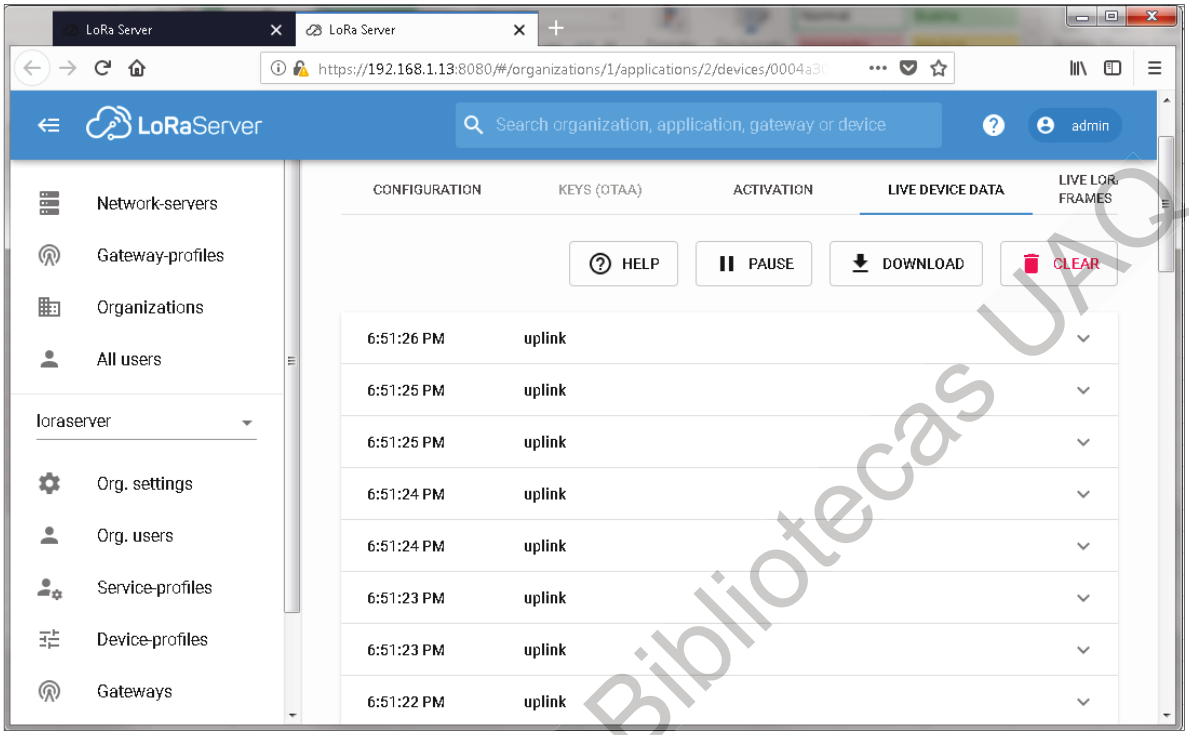

Figura 41 Datos LoRaWAN recibidos en el Servidor LoRaWAN.

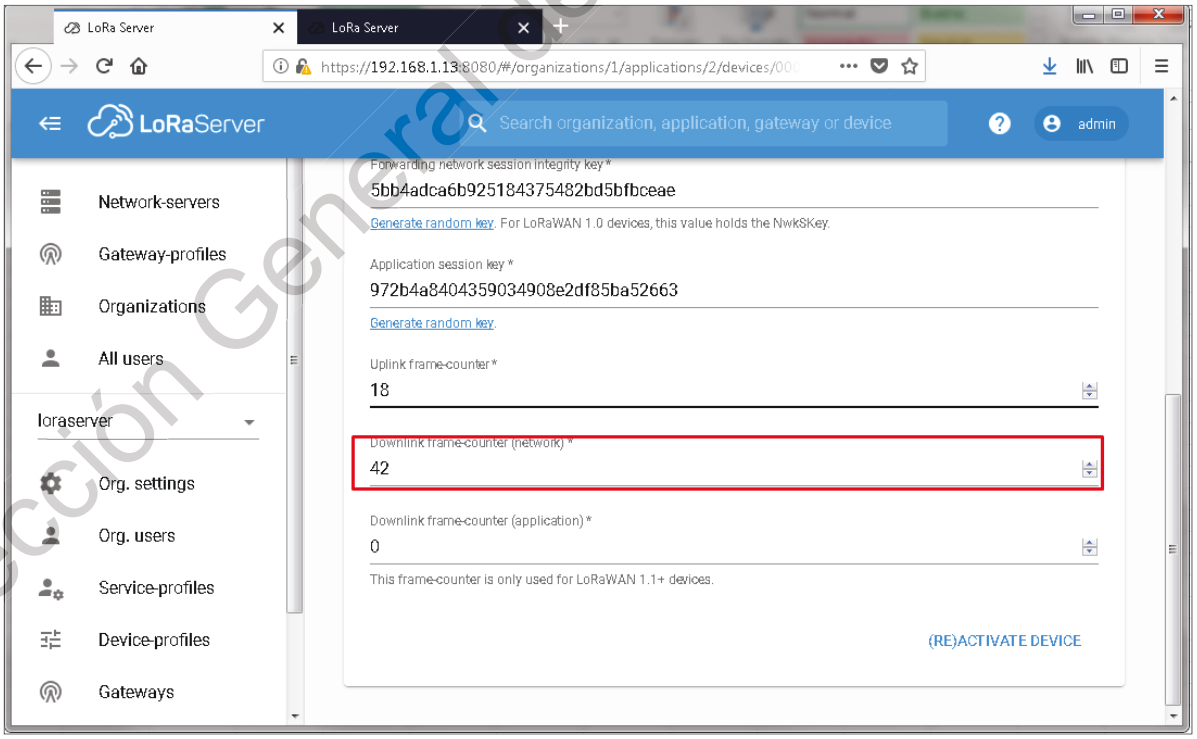

Figura 42 Contador de mensajes del Servidor LoRaWAN.

12. RESULTADOS Y DISCUSIÓN

En este capítulo se muestran los resultados de la implementación de la arquitectura de software propuesta y el resultado de su evaluación teniendo en cuenta diversas métricas de calidad de software.

Recapitulando, es necesario recordar que la tecnología para internet de las cosas que se utiliza en el presente trabajo tiene una arquitectura de aplicación basada en una topología tipo estrella cuyo elemento concentrador consiste en un *Gateway* Físico LoRa y los puntos de información denominados Nodos LoRaWAN son dispositivos de capacidades limitadas que proporcionan información o ejecutan acciones mediante el uso de módulos de radiofrecuencia tipo LoRa acoplados a microcontroladores operados mediante batería. En una aplicación típica de esta tecnología, el *Gateway* es el encargado de recibir los paquetes de mensajes desde los Nodos y reenviarlos hacia un Servidor LoRaWAN el cual se encarga de gestionar la autenticación de mensajes, filtración de mensajes repetidos provenientes de *Gateways* vecinos, decodificación de la trama LoRaWAN y su reenvío a Servidores de Aplicación.

Por otra parte, los sistemas ciber físicos que utilizan OPC-UA tradicional funcionan bajo un esquema Cliente-Servidor a través de diferentes niveles de redes distribuidas. El Servidor es cualquier dispositivo o sistema que cuenta con información disponible para ser compartida o modificada a través de diferentes servicios como: acceso a los valores actuales, valores históricos, subscripción para monitoreo continuo, intercambio de información bajo el paradigma publicador-suscriptor, descubrir dispositivos disponibles, manejo del modelo de información en vista binaria, ejecución remota de métodos, acceso a datos mediante encriptación y manejo de sesiones. El Cliente es aquel dispositivo o sistema que puede acceder o suministrar información desde o al Servidor con la finalidad de ser utilizada para la toma de decisiones, reportes, ajustes, inventario, control de proceso, mantenimiento o como parte de la línea de producción.

La arquitectura propuesta como parte de esta tesis está enfocada para ser un elemento intermedio entre el *Gateway Físico* y el Servidor LoRa. Se puede describir como un *Gateway* OPC-UA para LoRaWAN el cual permite la interpretación y manejo de datos proveniente de cualquier *Gateway Físico* y su uso de forma inmediata, proporcionando la capacidad de cómputo periférico, proporcionando la habilidad de pre-procesar la información proveniente de Nodos/*Gateways* LoRa antes que alcancen un Servidor *IoT*.

Considerando la arquitectura propuesta, el *Gateway* LoRa está compuesto principalmente de tres elementos: la Radio LoRa, el Controlador y el Reenviador de Paquetes. La Radio LoRa es un dispositivo de hardware capaz de interpretar las señales de radio provenientes de cualquier nodo LoRa. El Controlador es un paquete de software que se encarga de orquestar las diversas funciones de la Radio LoRa para ser utilizado dentro de un sistema de comunicación LoRaWAN. Finalmente, el Reenviador de Paquetes consiste en un set de configuraciones que son aplicados a los paquetes provenientes de la Radio para poder ser distribuidos a través de medios de conexiones de red, la utilidad de este paquete radica en que cualquier Radio/Controlador pueda suministrar la información LoRaWAN hacia un Servidor LoRa siguiendo una serie de reglas previamente definidas entre ambas entidades.

El Puente LoRaWAN consiste en un paquete de software que se encarga de interpretar la información proveniente de un *gateway* LoRa mediante la detección e interpretación de la información de los paquetes LoRa y LoRaWAN permitiendo que la información de los nodos sea visible al momento de la lectura. Para poder interpretar los paquetes LoRa y LoRaWAN es necesario el uso del paquete Metadatos LoRaWAN el cual se encarga de recopilar los parámetros necesarios para la interpretación desde un Servidor LoRa mediante llamadas remotas a procedimientos. El Puente LoRaWAN también permite fungir como el Reenviador de Paquetes LoRa integralmente, cuando es utilizado por el Servidor OPC-UA Pub/Sub, al momento de recibir una actualización a un elemento Subscrito, por lo

que el Paquete original es suministrado al Sevidor LoRa tal como lo hace proveniente del Reenviador.

El Servidor OPC-UA permite el uso de la información de *Gateways* y sus Nodos mediante un modelo de información dado por el Espacio de Direcciones, el cual consiste en una estructura tipo Árbol sobre los diferentes *Gateways* y Nodos cuya información pueden ser representada como Sub-Árboles y cada uno de los nodos de todo Espacio de Direcciones puede ser tratado como parte del servidor OPC, logrando una gran estandarización en el manejo de la información.

La Extension OPC-UA Pub/Sub soportada a partir de la versión 1.04 de la especificación del estándar OPC-UA consiste en la incorporación del patrón de intercambio de datos distribuidos orientado a mensajes el cual permite la extensibilidad de cualquier sistema. La Extensión permite el publicar o consumir la información de cualquier servidor OPC-UA bajo el esquema de *Middleware* Orientado a Mensaje (MOM).

El Intermediario Pub/Sub es parte fundamental del esquema MOM ya que posibilita la existencia de elementos publicadores o subscritos bajamente acoplados entre ellos, es deciri un elemento publicador no conoce si hay algún elemento subscrito o miles de ellos, es responsabilidad del Intermediario realizar esta gestión, liberando a ambos elementos de la carga de procesamiento que se tendría en algún otro esquema, como el cliente-servidor. El Intermediario es el responsable de mantener el vínculo entre emisores y consumidores de información y de la seguridad y confiabilidad de la información.

El prototipo se diseñó utilizando Nodos LoRaWAN de la marca The Things Network (TTN Nodes) que proporcionan información de temperatura, iluminación y voltaje de operación. Estos dispositivos también cuentan con un sensor de aceleración y un botón que son utilizados como generadores de eventos para despertar al microcontrolador de un estado de ultra bajo consumo de energía y realizar una transmisión de datos. El microcontrolador está basado en un SparkFun

Pro Micro 3.3 V a 8 MHz, el sensor de temperatura montado en la placa es un Microchip MCP9804, el acelerómetro es un NXP MMA8452Q de tres ejes tipo capacitivo y 12 bits de resolución, y el transceptor LoRa es un Microchip RN2903 modem LoRa para el rango de frecuencia ISM 915MHz, stack integrado LoRaWAN, gestionado mediante comando AT sobre UART, potencia ajustable de hasta 18.5dBm y sensibilidad de hasta -146dBm, certificado FCC y ROHS. La antena viene integrada en la tableta PCB con 1/8 de largo de onda. Contiene un conector USB 2.0 Micro-B para reprogramación o alimentación y espacio para tres baterías AAA.

El *Gateway* LoRa está basado en un RAK Wireless RAK831 acoplado a un Raspberry PI 3B mediante puerto SPI a través la tableta adaptadora *RPI Converter Board*. El dispositivo RAK831 es capaz de la recepción concurrente de hasta 8 paquetes LoRa gracias a la incorporación de un transceptor Semtech SX1301. La sensibilidad de recepción es de -142.5 dBm, la potencia máxima es de 23dBm, cuenta con conector SMA y antena externa de ½ longitud de onda. RAK Wireless proporciona los ejecutables y ejemplos de configuraciones para poder realizar el montaje sobre una RPI 3, éstos son componentes de software adecuados y personalizados para funcionar con el hardware proporcionado, aunque es fácilmente detectable que están basados en el código abierto proporcionado por The Thinks Network, por ejemplo el paquete *Packet_Forwarder* es una derivación del proyecto del mismo nombre realizado por The Things Network que a su vez está derivado del proyecto de LoRa Network de la Alianza LoRa. Éste se configura para que los paquetes LoRa sean enviados a una dirección IPV4 y Puerto especificado mediante datagramas UDP.

El Servidor LoRaWAN está basado en el proyecto Loraserver.io de código abierto bajo licencia MIT patrocinado por CableLabs, SIDNFonds y Acklio. Se trata de una solución para redes LoRaWAN que permite a los desarrolladores contar con un sistema amigable y listo para usar, sin embargo, es necesario realizar una serie de configuraciones para lograr el correcto direccionamiento entre las entidades

involucradas. Tiene la posibilidad de usar una *API* mediante *gRPC* y comandos *REST*. Está compuesto por tres paquetes distintos: LoRa *Gateway Bridge* (Puente *Gateway* LoRa), LoRa *Server* (Servidor LoRaWAN) y LoRa *App Server* (Servidor de Aplicación). El Puente *Gateway* LoRa tiene la función de abstraer el reenvío de información desde el *Gateway* mediante el uso de un Intermediario MQTT hacia el Servidor LoRa; el Servidor LoRaWAN es el encargado de gestionar los mensajes LoRaWAN mediante la identificación, gestión, decodificación y publicación hacia servidores de usuario; el Servidor de Aplicación es el cliente HTML que provee de una interfaz gráfica del Servidor LoRaWAN al usuario. El Servidor LoRaWAN en conjunto ha sido desplegado en una Raspberry PI.

El Servidor OPC-UA está basado en el proyecto de código abierto disponible bajo el patrocinio de la Fundación OPC a través de la licencia GPL 2.0. Existen varias versiones del stack disponible tanto en .Net Standard, Java y ANSI C. Se ha elegido .NET Standard ya que es el lenguaje de conveniencia adecuado al momento de escribir este trabajo, sin embargo, cualquiera de los tres disponible se puede utilizar y el resultado sería el mismo. Se utilizó la version 1.04/prototyping-mqtt del *stack* en su versión de desarrollo MQTT (Pub/Sub) ya que el stack 1.04, tal cual, no incluye la versión final de la extensión Pub/Sub que la Fundación formalmente soportará, sin embargo, la versión experimental se aplica perfectamente para los fines de esta investigación y poder demostrar que es posible la interoperabilidad entre los sistemas LoRaWAN en entornos OPC-UA bajo un modelo IoT de Pub/Sub. El Servidor OPC-UA en conjunto fue desplegado en un equipo Intel Core i5 de 64 bit con 3.6GHz, 500GB en disco duro y 14GB RAM.

La Extensión OPC-UA Pub/Sub, utilizada en este trabajo, hace uso de la librería M2MQTT un cliente para .Net y WinRT para peticiones MQTT disponible mediante código abierto patrocinado por la comunidad de Eclipse bajo licencia EPL 1.0.

El Intermediario MQTT Pub Sub está basado en Mosquitto que soporta la versión 3.1 y 3.1.1 del protocolo MQTT disponible mediante código abierto patrocinado por la comunidad de Eclipse bajo licencia EPL 1.0 y EDL1.0. En el presente trabajo mosquitto.exe es ejecutado en el mismo equipo de cómputo en el cual se ejecuta el Servidor OPC-UA, sin embargo es posible ejecutarlo en cualquier otro equipo dentro o fuera de la red local ya que soporta mecanismo de autenticación y cifrado.

Como parte de la evaluación del prototipo y de la arquitectura se llevó acabo una prueba de robustez sometiendo a pruebas de carga los módulos de software que la componen, el escenario de prueba consisitó en generar datagramas simulando el paquete de dato de un Nodo. La prueba se planteó para inyectar el datagrama diez mil veces con un retardo entre cada uno que fue variado desde 1ms hasta 1000ms. Los resultados de la **¡Error! No se encuentra el origen de la eferencia.** muestran evidencia que los componentes de la arquitectura de software pueden soportar una carga de trabajo de hasta 10ms entre cada paquete recibidos, sin embargo el Servidor LoRaWAN (Loraserver.io) sólo fue capaz de ser confiable para una carga de trabajo de 1000ms entre cada paquete.

JMeter Usuarios – Loop [datagrama]	Retardo entre datagrama (ms)	Paquetes en Puente LoRaWAN UDP	Paquetes en OPC-UA Cliente DA Histórico	Paquetes en MQTT Sub	Paquetes en LoRa Server.io
1 - 10000	1000	10000	10000	10000	10000
1 - 10000	100	10000	10000	10000	1996
1 - 10000	10	10000	10000	10000	305
1 - 10000	1	4741	4741	4741	42

Tabla 9 Resultados de la prueba de carga por módulos

13. CONCLUSIONES

El presente trabajo de investigación fue realizado con el objetivo de proponer una arquitectura de software que permita la interoperabilidad de sistemas heterogeneos en un ambiente de Internet de las Cosas a través de un medio estandarizado.

Fue un reto importante poder analizar los diferentes dominios tecnológicos y proponer una arquitectura que pudiera ser implementada en un prototipo funcional. Como parte de la delimitación y alcances, se escogió trabajar con la tecnología LoRaWAN como muestra de la población de los dispositivos *IoT* ya que es un claro ejemplo de tecnologías que han surgido en los últimos años para tratar de satisfacer la demanda por tecnologías más eficientes y especialmente diseñado para entornos *IoT* LPWAN de alta eficiencia energética y alto alcance de transmisión; se estima que el crecimiento de este tipo de comunicaciones se presente a una tasa de crecimiento mayor que las tecnologías tradicionales de corto alcance. Además, su licencia de funcionamiento lo hace ideal para desplegarlo de manera privada sin necesidad de pagar cuota por acceso a la red.

Por su parte, OPC-UA es un modelo de intercambio de datos muy utilizado para mejorar la interoperabilidad en sistemas de control de procesos. De hecho OPC ha permitido que una gran cantidad de dispositivos heterogeneos coexistan en entornos como manufactura y control; desde redes con sensores y actuadores de campo, celdas de manufactura y robots, hasta sistemas de planificación de recursos empresariales.

Mediante la arquitectura propuesta se integra la semántica de un sistema LoRaWAN en un ambiente OPC-UA permitiendo su uso con tecnologías heterogéneas a través de modelos de comunicación estandarizados que OPC provee. Con servicios como búsqueda de nodos, comunicación en capas de seguridad y procesamiento perimetral hacen que esta arquitectura *gateway* sea ideal para un ambiente de Sistemas Ciber Físicos en la Industria 4.0 y aplicarla en

solucionar problemas en varios modelos de negocios, por ejemplo: en fábricas donde se requiere de un monitoreo constante de niveles; mejorar la logística, movilidad y predicción de mantenimiento en entornos productivos; en general es posible aplicarla en entornos con sistemas no críticos, de baja latencia o que no requieren de restricciones de tiempo. Es posible confirmar que se cumplieron los objetivos planteados al inicio de la investigación y se corroboró la hipótesis planteada.

Mediante el uso del *gateway* propuesto se podría facilitar la incorporación, integración y aplicación de la tecnología LoRaWAN de Internet de las Cosas dentro de los sistemas OPC-UA actualmente en funcionamiento permitiendo la incorporación del paradigma *IIoT* (Industria 4.0) más asequible.

La siguiente fase de esta investigación, para futuros trabajos, consistiría en desarrollar una plataforma de servicios web que permita el consumo directo desde el servidor OPC-UA Pub/Sub hacia cualquier Servidor de Aplicación, de forma equivalente a lo que realiza Loraserver.io pero aplicado al propio servidor OPC.

14. REFERENCIAS

Alur, R. (2015). *Principles of Ciber-Phisical Systems*. Cambridge, Massachusetts: The MIT Press.

Astarloa, A., Bidarte, U., Jimenez, J., Zuloaga, A., & Lazaro, J. (2016). Intelligent gateway for industry 4.0-compliant production lines. *IECON Proceedings (Industrial Electronics Conference)*, 4902–4907. https://doi.org/10.1109/IECON.2016.7793890

Boman, J., Taylor, J., & Ngu, A. (2014). Flexible IoT Middleware for Integration of Things and Applications. *Proceedings of the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, (CollaborateCom), 481–488. https://doi.org/10.4108/icst.collaboratecom.2014.257533

Bor, M. C., Roedig, U., Voigt, T., & Alonso, J. M. (2016). Do LoRa Low-Power Wide-Area Networks Scale? In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems - MSWiM '16* (pp. 59–67). New York, New York, USA: ACM Press. https://doi.org/10.1145/2988287.2989163

Committee, O. M. Q. T. T. (MQTT) T. (2014). MQTT Version 3.1.1. OASIS STANDARD. Retrieved from http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf

Coulouris, G., Dollimore, J., & Kindberg, T. (2001). *Sistemas Distribuidos. Conceptos y diseño.* (A. Otero, Ed.) (Tercera). Madrid: Pearson Educación S.A.

Cupek, R., & Ziebinski, A. (2017). An OPC UA server as a gateway that shares CAN network data and engineering knowledge. In *Industrial Technology (ICIT), 2017 IEEE International Conference on* (pp. 1424–1429). Toronto, ON, Canada: IEEE. https://doi.org/10.1109/ICIT.2017.7915574

Curry, E. (2004). Message-Oriented Middleware. In Q. Mahmoud (Ed.), *Middleware For Communications* (pp. 1–26). West Sussex, England, UK: Wiley.

Domínguez, F. (2005). *LIIBUS : Arquitectura de Sistemas Interoperables entre Middlewares Heterogéneos usando Máquinas Abstractas Reflectivas Orientadas a Objetos .* Universidad de Oviedo.

Forsstrom, S., & Jennehag, U. (2017). A performance and cost evaluation of combining OPC-UA and Microsoft Azure IoT Hub into an industrial Internet-of-Things system. *2017 Global Internet of Things Summit (GIoTS)*, 1–6. https://doi.org/10.1109/GIOTS.2017.8016265

George, S. (2016). Microsoft introduces new open-source cross-platform OPC UA support for the industrial Internet of Things. Retrieved June 1, 2017, from https://blogs.microsoft.com/iot/2016/06/23/microsoft-introduces-new-open-source-cross-platform-opc-ua-support-for-the-industrial-internet-of-things/

Hevner, A. R., Park, J., Sudha, R., & March, S. T. (2004). Design Science in information systems research. *MIS Quarterly*, *28*(1), 75–105.

Imtiaz, J., & Jasperneite, J. (2013). Scalability of OPC-UA down to the chip level enables "internet of Things." *IEEE International Conference on Industrial*

Informatics (INDIN), 500–505. https://doi.org/10.1109/INDIN.2013.6622935
Johnson, P. (2016). *Ericsson Mobility Report. IoT Section*. Retrieved from https://www.ericsson.com/assets/local/mobility-report/documents/2016/ericsson-mobility-report-november-2016.pdf
Lee, J., Bagheri, B., & Kao, H.-A. (2014). A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*.
Low Throughput Networks. (2017). Retrieved June 1, 2017, from http://www.etsi.org/technologies-clusters/technologies/low-throughput-networks
Mahmoud, Q. (Ed.). (2004). *Middleware For Communications*. Inglaterra: John Wiley & Sons.
Margelis, G., Piechocki, R., Kaleshi, D., & Thomas, P. (2016). Low Throughput Networks for the IoT: Lessons learned from industrial implementations. *IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings*, 181–186. https://doi.org/10.1109/WF-IoT.2015.7389049
Mekki, K., Bajic, E., Chaxel, F., & Meyer, F. (2018). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express*. https://doi.org/10.1016/j.icte.2017.12.005
Mikhaylov, K., Petäjäjärvi, J., & Hänninen, T. (2016). Analysis of Capacity and Scalability of the LoRa Low Power Wide Area Network Technology. *European Wireless 2016*, 119–124.
Mineraud, J., Mazhelis, O., Su, X., & Tarkoma, S. (2016). A gap analysis of Internet-of-Things platforms. *Computer Communications*, *89–90*, 5–16. https://doi.org/10.1016/j.comcom.2016.03.015
Minerva, R., Biru, A., & Rotondi, D. (2015). Towards a definition of the Internet of Things (IoT). *IEEE#Internet#Initiative*, 1–86.
Ngu, A., Guitierrez, M., Metsis, V., & Nepal, S. (2016). IoT middleware: a survey on issue and enabling technologies. *IEEE Internet of Things Journal (in Print)*, *4*(1), 1–20. https://doi.org/10.1109/JIOT.2016.2615180
Oksanen, T., Linkolehto, R., & Seilonen, I. (2016). Adapting an industrial automation protocol to remote monitoring of mobile agricultural machinery: a combine harvester with IoT. *IFAC-PapersOnLine*, *49*(16), 127–131. https://doi.org/10.1016/j.ifacol.2016.10.024
Postel, J. (1980). *User Datagram Protocol*.
Postol, M. (2016). *OPC UA INFORMATION MODEL DEPLOYMENT*.
Qanbari, S., Behinaein, N., Rahimzadeh, R., & Dustdar, S. (2015). Gatica: Linked Sensed Data Enrichment and Analytics Middleware for IoT Gateways. *Proceedings - 2015 International Conference on Future Internet of Things and Cloud, FiCloud 2015 and 2015 International Conference on Open and Big Data, OBD 2015*, 38–43. https://doi.org/10.1109/FiCloud.2015.37
Tanenbaum, A., & Van-Steen, M. (2008). *Sistemas Distribuidos. Principios y Paradigmas.* (L. Cruz, Ed.) (Segunda). México: Pearson Educación.
Unified Architecture. OPC Unified Architecture Specification. (n.d.). Retrieved November 10, 2017, from https://opcfoundation.org/developer-tools/specifications-unified-architecture

Wixted, A. J., Kinnaird, P., Larijani, H., Tait, A., Ahmadinia, A., & Strachan, N. (2017). Evaluation of LoRa and LoRaWAN for wireless sensor networks. *Proceedings of IEEE Sensors*, *0*, 5–7. https://doi.org/10.1109/ICSENS.2016.7808712

Zachariah, T., Klugman, N., Campbell, B., Adkins, J., Jackson, N., & Dutta, P. (2015). The Internet of Things Has a Gateway Problem. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile '15)* (pp. 27–32). https://doi.org/10.1145/2699343.2699344