Results of the implementation of a sensor network based on Arduino devices and multiplatform applications using the standard OPC UA

Victor Vimos and Erwin J. Sacoto-Cabrera

***Abstract*— The Internet of Things is a technological paradigm that can be described as the set of elements that have small-embedded systems. This consisting of sensors, microprocessors, interconnection modules and actuators, which communicate with each other, to share useful information and improve processes. However, there are a large number of solutions from different vendors, both licensed and open source, that when deployed together may present problems, mainly due to a lack of standardization. In this paper we propose the use of OPC UA as middleware between MQTT and cross-platform Web applications. In addition, we develop a data management platform. Among the results obtained, it is possible to show the advantages of using MQTT and OPC UA, since it provides greater efficiency in the response time and security of the application.**

***Keywords*— Industry 4.0, Internet of Things, middleware, MQTT, OPC AC, sensor networks.**

I. INTRODUCCIÓN

El Internet de las Cosas (IoT por sus siglas en inglés) es un paradigma tecnológico que está ganando terreno, tanto en el campo de las telecomunicaciones como en el campo computacional [1]. Puede ser descrito como un término colectivo que agrupa diferentes elementos (maquinaria, vehículos, mercancías, electrodomésticos, ropa y "cosas" en general), que poseen pequeños sistemas embebidos conformados por sensores, microprocesadores, interfaces de conexión cableada/inalámbrica y actuadores, los cuales se comunican, entre sí y con diversas aplicaciones, a través de Internet.

Además, tienen la posibilidad de percibir su entorno, por lo que presentan un comportamiento situacional que ayuda a crear servicios inteligentes y autónomos. Esto permite el auge de nuevas tendencias, basadas en el almacenamiento de datos a gran escala, el análisis de cantidades robustas de información y la incorporación de sistemas cyber-físicos en comunicaciones máquina-a-máquina (M2M) [2].

Por un lado, con IoT nace el concepto de la industria 4.0 o "Internet de las Cosas Industriales (IIoT)", implicando una evolución de las máquinas hacia un contexto de interconexión mutua para formar una comunidad colaborativa, enfocada en mejorar la productividad, reducir costos y ofrecer servicios más eficientes [3]-[4]. Así, los sistemas industriales tradicionales, que no se encuentran conectados a Internet, se están transformando en "fábricas inteligentes" o automatizadas, que presentan diferentes niveles de control [5]. En un nivel más bajo se exponen un gran número de dispositivos de campo (por ejemplo, sensores y PLCs), los cuales recogen información de procedimientos industriales; mientras que, en un nivel superior, diferentes aplicaciones utilizan la información recolectada para tomar decisiones inteligentes [6]. Se requiere el empleo de herramientas avanzadas de predicción y monitoreo, que ayuden al tratamiento sistemático de diversos datos para convertirlos en información útil. De esta forma, se pueden explicar incertidumbres y optimizar procesos industriales [4].

Por otro lado, el despliegue de "Smart Cities"—una solución prometedora para suministrar servicios eficientes a la ciudadanía con el uso de tecnologías de la información y comunicación (TICs)—cada día gana sostenibilidad, apoyándose en IoT. Así, se han abierto las puertas para el desarrollo de aplicaciones eficientes, que buscan mejorar la vida de las personas, el uso de los recursos y el costo operacional de la administración pública [7]. Con el monitoreo del comportamiento de los habitantes de una ciudad, se puede crear conocimiento útil que se convierte en un factor crucial al momento de determinar hábitos de consumo [8]-[9]. Además, se busca garantizar la conservación del entorno y las condiciones salubres de la ciudadanía, por lo que la medición de variables medioambientales (temperatura, polución, humedad, entre otros), permite realizar los análisis y estudios adecuados, para tomar acciones correctivas en el momento oportuno [10]-[11].

Como se ha podido evidenciar, ya sea en la industria o en la vida cotidiana, el IoT puede resumirse como el procesamiento de datos de entrada para tomar decisiones eficientes sobre un proceso [12]. Uno de sus pilares son las redes de sensores, conformadas por varios elementos de medición y monitoreo. Ante esta situación, para el tratamiento de los datos obtenidos, es necesario la implementación de sistemas embebidos de bajo costo, por lo que varios fabricantes están invirtiendo grandes recursos investigativos en su diseño y producción, basándose en hardware y software, tanto licenciado como "open source" [13]. Entre las propuestas más relevantes, se pueden encontrar Arduino y Raspberry Pi, los cuales son dispositivos que cuentan con interfaces programables y altamente compatibles con diferentes tipos de sensores (temperatura, detección de poluciones, ruido, intensidad lumínica, entre otras) [14]. Sin embargo, aunque el uso de sistemas embebidos de bajo costo ha facilitado el despliegue de tendencias IoT, la amplia

Victor Vimos, Universidad Politécnica Salesiana, Cuenca, Ecuador, vvimos@est.ups.edu.ec

Erwin J. Sacoto Cabrera, Universidad Politécnica Salesiana, Cuenca, Ecuador, esacoto@ups.edu.ec.

Corresponding author: Erwin J. Sacoto Cabrera.

diversidad de proyectos existentes, ha provocado que la mayoría de soluciones diseñadas trabajen de manera aislada. Esto dificulta su integración en una plataforma centralizada, conllevando a la existencia de inconsistencias o falta de continuidad en los registros enviados por cada sensor.

A. Protocolos para Internet de las Cosas (IoT)

A medida que el IoT se expande en una gran cantidad de aplicaciones, gracias a la minimización de los sistemas embebidos actuales, la disponibilidad de sensores más versátiles y la creación de "objetos inteligentes", están surgiendo varios protocolos end-to-end para potenciar sus interconexiones máquina a máquina (M2M por sus siglas en inglés) [15]-[16]. Al momento de diseñar cada protocolo, se deben cumplir dos características principales: hacerle frente a conexiones poco fiables e intermitentes, y manejar anchos de banda relativamente bajos [17]. Entre los más importantes se encuentran:

1) CoAP (Constrained Application Protocol):

Es un protocolo especializado de transferencia web para su uso en nodos y redes con recursos limitados. Es utilizado principalmente para aplicaciones M2M, tales como servicios de energía inteligente y automatización de edificios. Proporciona un modelo de interacción solicitud/respuesta entre los end-points de las aplicaciones, soporta el descubrimiento integrado de servicios y recursos, e incluye conceptos clave de la Web como objetos de tipo Identificador de Recursos Uniforme (URI por sus siglas en ingles). Está diseñado para interactuar fácilmente con HTTP, a la vez que satisface requisitos especializados, entre los que se incluye multicast, bajo overhead y simplicidad para entornos limitados [18].

2) MQTT (Message Queue Telemetry Transport):

Es un protocolo de código abierto, basado en mensajería de publicación/suscripción. Fue diseñado para dispositivos restringidos y redes de bajo ancho de banda, alta latencia o poco confiables. El cliente MQTT publica mensajes a un broker MQTT, los cuales son suscritos—o retenidos para suscripciones futuras—por otros clientes. Cada mensaje se publica en una dirección conocida como tópico. Así, los clientes pueden suscribirse a múltiples tópicos y recibir los mensajes publicados en ellos. Utiliza TCP como protocolo de transporte y TLS/SSL para la seguridad, lo que conlleva a que la comunicación entre el cliente y el broker esté orientada a la conexión. Es adecuado para grandes redes de dispositivos pequeños que necesitan ser monitoreados o controlados desde un servidor back-end en Internet. No está diseñado para la transferencia de dispositivo a dispositivo ni para datos multicast a muchos receptores. Es un protocolo de mensajería muy básico que ofrece pocas opciones de control [18]-[20].

3) DDS (Data Distribution System):

Originalmente fue creado como un middleware de red para hacerle frente a las desventajas de una arquitectura centralizada en suscripciones. Se basa en TCP, presentando nodos descentralizados de clientes, los cuales se identifican como suscriptores o editores, a través de un servidor de localización. El uso de este sistema elimina la necesidad de que los usuarios identifiquen en dónde se encuentran otros nodos potenciales o qué temas les interesan, ya que existen procesos de auto-descubrimiento. Después de vincular editores y suscriptores, las conexiones dejan de utilizar las funciones del servidor, para convertirse en comunicaciones peer-to-peer [21].

Actualmente, se están desarrollando una gran cantidad de protocolos, a más de los mencionados, que ayudan a implementar diversas funciones, aprovechando de mejor manera los recursos disponibles en las tecnologías IoT [22]. No obstante, su falta de estandarización crea problemas de interoperabilidad entre distintos fabricantes, incluyendo protocolos Web [23]. Para esto, existen diversas soluciones, siendo una de las más populares el uso de un middleware basado en el estándar OPC UA.

B. Estándar OPC UA

Openness, Productivity and Connectivity Unified Architecture (OPC UA), es un estándar para la transmisión segura, fiable y neutra de datos sobre IP, que propone la implementación de una arquitectura para la integración de dispositivos de distintos fabricantes [24]-[25]. En su forma más general, es descrito como un middleware para el intercambio de datos entre los firmwares de dispositivos de campo (sensores), sistemas de control y aplicaciones [26]. Su principal objetivo es ofrecer información de alta disponibilidad sobre los procesos industriales/sensores, a cualquier aplicación o usuario autorizado, sin restricciones espacio-temporales. Esta funcionalidad es independiente del fabricante que haya desarrollado la aplicación, el lenguaje de programación seleccionado y el sistema operativo utilizado [23].

Desde un punto de vista técnico, OPC UA es la evolución de OPC clásico, el cuál desarrolló sus interfaces basándose en COM y DCOM, siendo compatible únicamente con Windows; esto es un problema importante debido al gran número de fabricantes existentes a la fecha. Entonces ha sido necesario romper esa dependencia, creando un estándar que utilice técnicas orientadas a objetos e incluya jerarquías de tipo y herencia para modelar la información a un Schema XML [25], [27]. Por ello, OPC-UA puede ser mapeado en una variedad de protocolos de comunicación y los datos pueden ser codificados de varias maneras, compensando portabilidad y eficiencia. Su funcionamiento puede explicarse en la creación de clientes y servidores OPC UA, contenidos en una misma arquitectura. Cada cliente puede interactuar simultáneamente con uno o más servidores, y cada servidor puede interactuar simultáneamente con uno o más clientes. Una aplicación puede combinar componentes de servidor y cliente para permitir la interacción con otros servidores y clientes, a varios niveles jerárquicos para una integración vertical perfecta [27].

En la Figura 1, se puede observar un esquema del funcionamiento de OPC-UA.

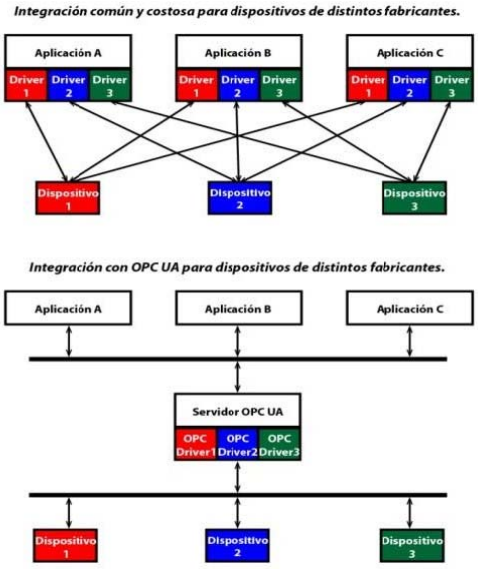

Figura 1. Comparación de la integración de dispositivos de diferentes fabricantes, de forma común y con OPC UA.

El uso de OPC es una gran ventaja al momento de unificar diferentes soluciones de hardware y software para IoT. Por tal motivo, en este trabajo se propone la implementación de una plataforma de medición y monitoreo de variables medioambientales (sensores de temperatura y humedad), basada en Arduino y Raspberry Pi. Se desplegará un middleware OPC entre el protocolo MQTT, empleado por Arduino para empaquetar los datos recibidos por cada sensor, y los diferentes protocolos de una aplicación Web multiplataforma. Además, se realizarán configuraciones en la herramienta Grafana para validar estadísticamente los datos de los sensores, ya que, al desarrollar un sistema apoyado en hardware y software open source, no existe el soporte adecuado de los fabricantes de cada dispositivo.

Este artículo se encuentra estructurado de la siguiente manera. En la Sección II, se describe la propuesta de diseño de la plataforma; mientras que, en la Sección III, se exponen sus principales configuraciones. En la Sección IV, se muestran los resultados obtenidos en diferentes pruebas, para corroborar el uso del estándar OPC UA como middleware entre protocolos IoT y la validación estadística de las mediciones de cada sensor. Finalmente, en la Sección V, se presentan las principales conclusiones y el trabajo futuro.

II. DESCRIPCIÓN DE LA PROPUESTA

La propuesta que se detallará a continuación se basa en una arquitectura conceptual que hace uso de dispositivos Arduino, Raspberry PI y OPC UA [25]. Se han realizado algunas modificaciones para conseguir un arquitectura más robusta y apegada a la realidad (ver Figura 2), utilizando un servidor OPC UA como un servicio seguro publicado en Internet desde una Zona Desmilitarizada (DMZ) para facilitar la recolección de datos. Las conexiones desde los clientes OPC AU han sido autenticadas autenticadas y cifradas, mediante certificados digitales. Los elementos principales de esta propuesta, se resumen en:

- Dispositivos o sensores de hardware limitado.
- OPC UA Gateway.
- OPC UA Server.
- Servidor de análisis de datos.

Como se ha mencionado, fue necesario agregar elementos secundarios que intervienen en la generación y transmisión de datos en una aplicación robusta. Por ello, se ha configurado un servidor NTP (Network Time Protocol), que permite sincronizar una marca de tiempo asignada a los datos transmitidos desde los elementos perimetrales; y un servidor DNS (Domain Name System), para explotar LDS (Local Discovery Server)—una de las ventajas presentes en OPC UA—y ayudar a que cada Gateway OPC UA se comunique con servidores OPC UA secundarios cuando el principal fallé. La arquitectura que respalda un valor de confiabilidad elevado se encuentra en la Figura 3. Finalmente, para la transmisión de datos entre el Gateway OPC UA y el Server OPC UA, se utilizaron librerías desarrolladas en JAVA por la fundación OPC.

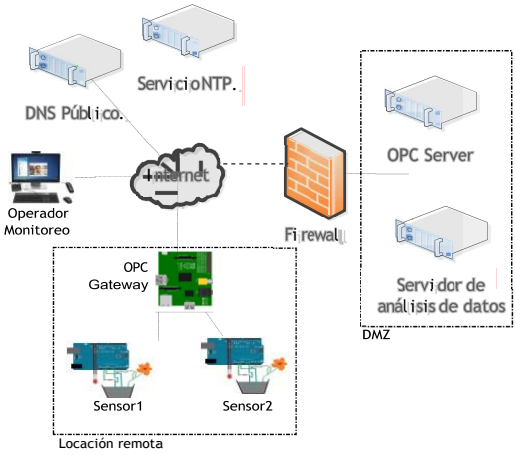

Figura 2. Diagrama de conexión de la arquitectura propuesta

En las siguientes subsecciones, se describirán los dispositivos físicos que se emplearon para cubrir las funciones de los elementos principales de la arquitectura propuesta.

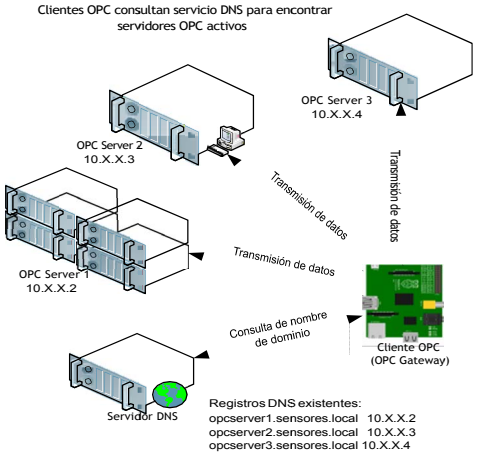

Figura 3. Arquitectura de respaldo en caso de fallas del servidor principal.

A. Dispositivos o Sensores de hardware limitado.

Representado por un Raspberry Pi 2, su principal objetivo es recibir y normalizar los datos MQTT, enviados por los sensores, a una estructura OPC. Para esto, cuenta con un servicio de broker MQTT, lo cual permite considerarlo como un nodo principal de la red de sensores.

El proceso de normalización inicia con el almacenamiento temporal de los datos en un archivo de texto plano; se les coloca una marca de tiempo en función de su hora de llegada; y, se los estructura en concordancia con la disposición XML, según las normativas de OPC UA. Finalmente, los datos XML, son enviados hacia el servidor OPC UA.

Entiéndase como el conjunto constituido por un módulo Arduino UNO equipado con un Ethernet Shield y dos sensores que registran mediciones de temperatura y humedad, conocido como "sensor/es" desde este punto. Algo importante a tener en consideración, es el hecho de que los sensores no interactúan directamente con el servidor OPC UA por la falta de librerías que habiliten el uso del estándar en Arduino. Por tal motivo, la transmisión se ha realizado desde cada sensor hacia el Gateway OPC UA (cuenta con un broker MQTT), por medio de una conexión TCP/IP y una interfaz de software basada en MQTT con los siguientes parámetros:

- Dirección IP del equipo con servicio broker MQTT activo.
- Servicio de publicación MQTT: Asunto de mensaje y mensaje.
- Servicio de subscripción MQTT: Asunto de mensaje.

Debido a que los sensores son considerados como elementos perimetrales, para el encaminamiento de los datos en formato MQTT hasta un nodo principal, que los convierte a formato OPC y los retransmite al servidor OPC UA, se han explotado algoritmos jerárquicos, siguiendo la recomendación expuesta en trabajos similares desarrollados con sistemas embebidos [26]. Además, para aprovechar la simplicidad del envío de mensajes MQTT, se ha agregado información adicional a los valores recolectados para ayudar en su posterior clasificación. En la Figura 4, se puede visualizar la traza MQTT utilizada, en donde: A es el ID del Arduino, B es el ID del sensor, C es el valor registrado de la variable, y D es la descripción de la variable.

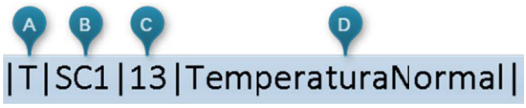

Figura 4. Traza MQTT generada en cada sensor.

B. Gateway OPC UA

Representado por un Raspberry Pi 2, su principal objetivo es recibir y normalizar los datos MQTT, enviados por los sensores, a una estructura OPC. Para esto, cuenta con un servicio de broker MQTT, lo cual permite considerarlo como un nodo principal de la red de sensores.

El proceso de normalización inicia con el almacenamiento temporal de los datos en un archivo de texto plano; se les coloca una marca de tiempo en función de su hora de llegada; y, se los estructura en concordancia con la disposición XML, según las normativas de OPC UA. Finalmente, los datos XML, son enviados hacia el servidor OPC UA.

C. Servidor OPC UA.

Este servicio ha sido desplegado en un servidor de aplicación Web con dirección IP pública y la configuración necesaria para recibir mensajes de los clientes OPC en el puerto 8666. Se ha generado una regla de NAT en un firewall perimetral para habilitar el tráfico de entrada en el puerto mencionado, tomando en consideración las recomendaciones de [29].

D. Servidor de análisis de datos.

Debido a la falta de soporte por parte de los fabricantes de dispositivos open source, cuando finalizan las comunicaciones sensor/servidor OPC UA, se deben analizar los datos recibidos, generando información útil para procesos de optimización. En esta propuesta, se emplea una herramienta de monitoreo y análisis llamada Grafana, la cual despliega funciones de tablero de mando para gestionar información de diferentes elementos. Cuenta con métodos que facilitan la detección de anomalías para crear alertas vía correo electrónico, en caso de que existan comportamientos inesperados.

Una vez descritos los dispositivos físicos que representan los elementos principales de la arquitectura propuesta, en la siguiente sección se describen sus características a nivel de hardware y software, y sus principales configuraciones.

III. IMPLEMENTACIÓN Y CONFIGURACIÓN DE LA ARQUITECTURA PROPUESTA.

El desarrollo de este trabajo puede ser descrito como un conjunto de sensores, apoyados en módulos Arduino, que registran valores de presión y temperatura, para empaquetarse en formato MQTT y enviarse a un servicio broker MQTT que está corriendo en un Raspberry. En esta Raspberry, además existe un Gateway OPC UA, el cual normaliza los datos que recibe siguiendo una estructura XML, según las normativas OPC. Los datos normalizados se transmiten a un servidor OPC UA para ser utilizados en una aplicación Web multiplataforma. Finalmente, se implementa un proceso de validación de la información registrada por cada sensor con la ayuda de una herramienta estadística, como lo es Grafana. Las características del hardware y software que se eligió para cada dispositivo de hardware son:

- Servidor de análisis de datos: Sistema operativo Ubuntu 16.4 LTS, panel de control para visualización de datos Grafana versión 4.3.1, base de datos MySQL versión Comunitaria 5.7.18.1.
- OPC Server: Ubuntu 16.4 LTS, JDK6, JRE6.

- OPC Client: Raspberry PI2, Modelo B, 1GB RAM, 3 GB de almacenamiento, Raspbian Jessie, JRE6, Mosquitto MQTT Broker versión 3.1
- Sensor o Dispositivo de Hardware Limitado: Arduino UNO, dos sensores de humedad para suelo FC-28, un sensor DTH11 para monitoreo de temperatura y humedad en el aire, un módulo de conexión Ethernet que habilita la capacidad de trasmitir datos haciendo uso de protocolos IP.

En las siguientes subsecciones se detallarán como se ha configurado cada elemento principal de la arquitectura propuesta.

A. Configuración en el sensor o dispositivo de hardware limitado.

Con las librerías generadas por "Nick O'Leary" , las cuales son distribuidas bajo "MIT License", se ha configurado en cada Arduino el rol de cliente MQTT. Desarrolladas con base en C, proporcionan las interfaces de software necesarias para transmitir los datos recolectados, haciendo uso de métodos programables muy simples.

Los módulos Arduino son dispositivos altamente programables, lo que facilita la personalización del formato de los registros trasmitidos. Por ello, aunque en este proyecto la estructura de datos se sustenta en barras verticales como separador de campo, otra alternativa podría ser usada es JSON, empleando etiquetas para facilitar la recolección y clasificación de información. Finalmente, se debe tener en cuenta que los sensores fueron diseñados exclusivamente para recolectar información, por lo que las marcas de tiempo se agregan en el Gateway OPC UA. De esta forma se ha logrado disminuir el consumo de energía e incrementar el volumen de transmisión de datos válidos.

Con las configuraciones que se han detallado, y en concordancia a las pruebas de transmisión reflejadas en [27], la velocidad de transmisión de datos entre 5 clientes y el broker MQTT Broker, usando una red LAN Fast Ethernet sin cifrado de conexión, es de 250 solicitudes por segundo con un QoS = 2.

B. Configuración del Gateway OPC UA y del servidor OPC UA.

A través de un proceso programado en Java, los datos recibidos por el broker MQTT son incluidos en la estructura XML requerida por OPC. Así, se retransmite la información recolectada por los sensores en un formato compatible con servidor OPC UA [30].

El servidor OPC UA recibe datos del Gateway OPC UA en el puerto 8666. Por defecto, el puerto empleado en aplicaciones Web es el 443, sin embargo, debido a que el servicio fue publicado en Internet, el puerto fue cambiado para incrementar la seguridad. Además, en este dispositivo se ha instalado una base de datos MySQL versión Comunitaria 5.7.18.1, para mejorar el rendimiento de la aplicación, permitiendo el uso de SQL y la integración con herramientas de monitoreo. Cabe destacar que, los costos del despliegue de un servidor OPC UA se ven disminuidos con el uso soluciones open source.

El proceso completo que sigue la implementación de la arquitectura propuesta y el flujo de la información, desde su origen hasta su análisis, se puede observar en la Figura 5.

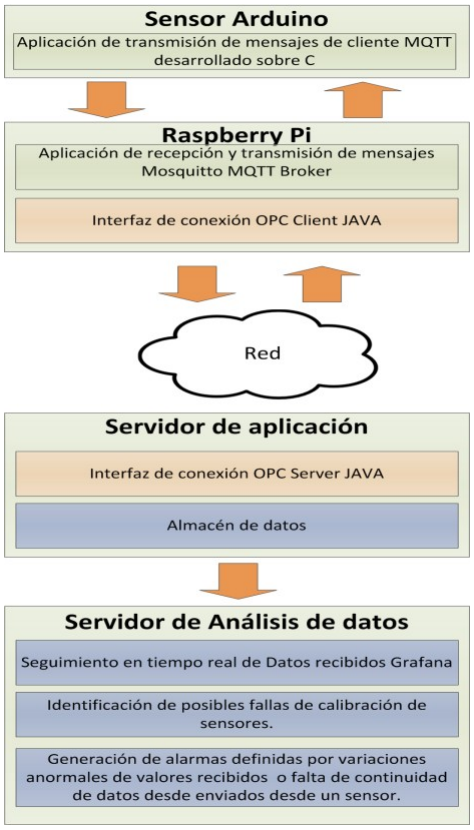

Figura 5. Flujo de procesamiento de registros desde su origen hasta su análisis.

C. Configuración del Servidor de análisis de datos.

Grafana, proporciona la posibilidad de realizar un monitoreo proactivo de las mediciones obtenidas en tiempo real, esto es posible por medio de la creación de una serie de reglas que combinan procesos estadísticos como la identificación de mínimos, máximos, media, varianza , entre otros, que al ser combinados con consultas SQL sobre la base de datos relacional en la que se almacenan los registros recibidos, permiten detectar anomalías dentro de un conjunto de datos agrupados por el ID de su sensor origen y su hora de almacenamiento. En el caso de detectar alguna inconsistencia, se genera inmediatamente una alarma que es enviada por correo electrónico al administrador de la plataforma y se muestra una notificación en el panel de control de la aplicación Web, como se puede observar en la Figura 7.

En la siguiente sección se detallarán las pruebas realizadas, analizando los resultados obtenidos.

IV. PRUEBAS Y ANÁLISIS DE RESULTADOS

Para evaluar el correcto funcionamiento de la aplicación de

monitoreo medioambiental, se ha definido una lista de valores predefinidos para ser transmitidos en intervalos de tiempo iguales. Esto significa que los valores de entrada en cada sensor deben ser iguales a los datos de salida registrados en el servidor OPC UA.

Lo primero que puede notarse es que, en la comunicación desde el cliente OPC UA hasta el servidor OPC UA existe una latencia promedio de red 108ms, transmitiendo 1 dato cada 20 segundos. En gran parte se debe al hecho de que se establece una conexión cifrada para cada dato, elevando considerablemente el procesamiento en el Raspberry PI. En comparación a algunas investigaciones, como por ejemplo en [28], estos valores podrían parecer altos, por lo que se debe considerar que, en esas investigaciones las pruebas fueron desplegadas en equipos con mejores prestaciones.

En las pruebas realizadas en este trabajo, ha sido posible verificar que las secuencias de datos enviadas no presentaron perdidas al utilizarse bloques de 100 registros, los cuales fueron receptados y registrados correctamente por el servidor de análisis de datos. Por tal motivo, se puede afirmar que el proceso de transmisión fue un éxito. En la Figura 6, se exponen los resultados obtenidos en la aplicación Web para el sensor de humedad. Las líneas entrecortadas verticales muestran posibles problemas de inconsistencia en el tiempo, que en este caso se debe al tiempo muerto como resultado del inicio del periodo de censado; mientras que, las líneas horizontales rojas marcan los límites establecidos como funcionamiento normal en cuanto a la magnitud de los datos.

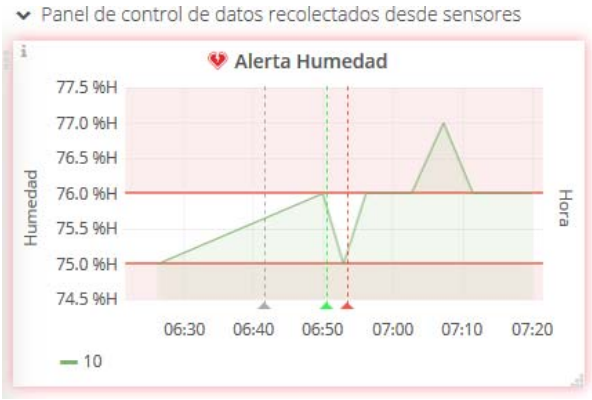

Figura 6. Visualización de los datos registrados por un sensor de humedad

Para comprobar el funcionamiento de las alarmas generadas por el servidor de análisis de datos, en otras pruebas se ha procedido a disminuir el intervalo de tiempo existente en el envío, para evaluar la consistencia de los datos. Así, se inició con un conjunto de 1000 registros con un intervalo de 60s, para posteriormente disminuirlo a 1 s. Este proceso fue repetido hasta que se encontraron inconsistencias con tiempos menores a 20s. Como se puede observar en la Figura 7, al encontrarse alguna inconsistencia en las lecturas de algún sensor, se genera una alarma en la aplicación Web.

Figura 7. Visualización de alarmas de sensores registradas.

V. CONCLUSIONES.

A partir del despliegue de la propuesta para la interconexión de sensores medioambientales (MQTT) con una aplicación Web multiplataforma, mediante un middleware OPC UA, se realizaron varias pruebas que permiten confirmar lo siguiente:

Las pruebas realizadas permitieron validar el uso de OPC UA como middleware, asegurando la integración de un protocolo IoT con una aplicación Web multiplataforma.

Se debe considerar que la arquitectura recomendada en este documento procura asegurar la transmisión de datos desde el Gateway OPC UA hacia el Servidor OPC UA, empleando certificados digitales para su cifrado y autenticación.

- En el caso de implementar tecnologías y protocolos inalámbricos de comunicación, desde los sensores hasta el Gateway OPC UA, el sistema podría estar expuesto a riesgos de seguridad.
- La eficacia del uso de Grafana como herramienta de análisis, monitoreo y evaluación de los datos recolectados, depende directamente de la claridad y detalle con que se especifiquen las reglas de generación de alarmas.
- Aunque OPC UA es un estándar muy difundido y desarrollado, el contar con el driver para el cliente OPC no garantiza la compatibilidad con el 100% de las aplicaciones de control y adquisición de datos, ya que se pueden requerir parámetros adicionales a los implementados dentro del middleware incorporado al Cliente
- Como trabajo futuro se propone ir un paso más allá e integrar varios protocolos IoT, como CoAP y DSS, por medio del estándar OPC UA, creando, de igual manera, una aplicación Web multiplataforma.

REFERENCIAS

[1] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] S. Forsström and U. Jennehag, “A performance and cost evaluation of combining opc-ua and microsoft azure iot hub into an industrial internet-of-things system,” in Global Internet of Things Summit (GIoTS), 2017. IEEE, 2017, pp. 1–6.

[3] M. Okuda, T. Mizuya, and T. Nagao, “Development of iot testbed using opc ua and database on cloud,” 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), 2017.

[4] P. L. Gallegos-Segovia, J. F. Bravo-Torres, J. J. Argudo-Parra, E. J. Sacoto-Cabrera and V. M. Larios-Rosillo, "Internet of Things as an attack vector to critical infrastructures of cities," 2017 International Caribbean Conference on Devices, Circuits and Systems (ICCDCS), Cozumel, 2017, pp. 117-120.

[5] J. Lee, H.-A. Kao, and S. Yang, “Service innovation and smart analytics for industry 4.0 and big data environment,” Procedia Cirp, vol. 16, pp. 3–8, 2014.

[6] S. Bhardwaj, P. Larbig, R. Khondoker, and K. Bayarou, “Survey of domain specific languages to build packet parsers for industrial protocols,” 2017 20th International Conference of Computer and Information Technology (ICCIT), 2017.

[7] S. Cavalieri, D. Di Stefano, M. G. Salafia, and M. S. Scroppo, “Opc ua integration into the web,” IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, 2017.

[8] E. Z. Tragos, V. Angelakis, A. Fragkiadakis, D. Gundlegard, C.-S. Nechifor, G. Oikonomou, H. C. Pohls, and A. Gavras, “Enabling reliable and secure iot-based smart city applications,” in Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on. IEEE, 2014, pp. 111–116.

[9] C. A. Medina, M. R. Perez, and L. C. Trujillo, “Iot paradigm into the smart city vision: A survey,” 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017.

[10] S. Halder and G. Sivakumar, “Embedded based remote monitoring station for live streaming of temperature and humidity,” 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), 2017.

[11] J. M. Del Alamo, R. Trapero, Y. S. Martin, J. C. Yelmo and N. Suri, "Assessing Privacy Capabilities of Cloud Service Providers," in IEEE Latin America Transactions, vol. 13, no. 11, pp. 3634-3641, Nov. 2015.

[12] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (iot): A vision, architectural elements, and future directions,” Future generation computer systems, vol. 29, no. 7, pp. 1645–1660, 2013.

[13] N. Vikram, K. Harish, M. Nihaal, R. Umesh, A. Shetty, and A. Kumar, “A low cost home automation system using wi-fi based wireless sensor network incorporating internet of things (iot),” 2017 IEEE 7th International Advance Computing Conference (IACC), 2017.

[14] N. Agrawal and S. Singhal, “Smart drip irrigation system using raspberry pi and arduino,” in Computing, Communication & Automation (ICCCA), 2015 International Conference on. IEEE, 2015, pp. 928–932.

[15] Y. Chen and T. Kunz, “Performance evaluation of iot protocols under a constrained wireless access network,” in Selected Topics in Mobile & Wireless Networking (MoWNeT), 2016 International Conference on. IEEE, 2016, pp. 1–7.

[16] J. E. Giral Sala, R. Morales Caporal, E. Bonilla Huerta, J. J. Rodriguez Rivas and J. d. J. Rangel Magdaleno, "A Smart Switch to Connect and Disconnect Electrical Devices at Home by Using Internet," in IEEE Latin America Transactions, vol. 14, no. 4, pp. 1575-1581, April 2016.

[17] M. B. Yassein, S. Aljawarneh, and W. Al-Sarayrah, “Mobility management of internet of things: Protocols, challenges and open issues,” 2017 International Conference on Engineering & MIS (ICEMIS), 2017.

[18] IETF, “Coap: Constrained application protocol,” 2018. [Online]. Available: https://-tools.ietf.org/¬html/¬rfc7252#page-10

[19] OASIS, “Mqtt: Message queue telemetry transport,” 2018. [Online]. Available: http://-mqtt.org/¬

[20] N. Naik, “Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http,” in Systems Engineering Symposium (ISSE), 2017 IEEE International. IEEE, 2017, pp. 1–7.

[21] OMG, “Dss: Data distribution services,” 2018. [Online]. Available: http://¬www.omg.org/-spec/¬DDS/¬

[22] P. Datta and B. Sharma, “A survey on iot architectures, protocols, security and smart city based applications,” in 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2017, pp. 1–5.

[23] P. Desai, A. Sheth, and P. Anantharam, “Semantic gateway as a service architecture for iot interoperability,” in Mobile Services (MS), 2015 IEEE International Conference on. IEEE, 2015, pp. 313–319.

[24] OPC-Foundation, “Opc unified architecture,” 2018. [Online]. Available: https://-opcfoundation.org/¬wp-content/¬uploads/¬2014/¬05/-OPC-UA_Overview_ES.pdf

[25] V. Vimos, E. Sacoto, and D. Morales, “Conceptual architecture definition: Implementation of a network sensor using arduino devices and multiplatform applications through opc ua,” in Automatica (ICA-ACCA), IEEE International Conference on. IEEE, 2016, pp. 1–5.

[26] M. Son and M.-J. Yi, “A study on opc specifications: Perspective and challenges,” in Strategic Technology (IFOST), 2010 International Forum on. IEEE, 2010, pp. 193–197.

[27] J. Imtiaz and J. Jasperneite, “Scalability of opc-ua down to the chip level enables â€œinternet of thingsâ€ ,” in Industrial Informatics (INDIN), 2013 11th IEEE International Conference on. IEEE, 2013, pp. 500–505.

[28] N. Sun, G. Han, H. Lu, and C. Lin, “A real-time monitoring and statistic system using hierarchical sensor network,” in Computational Intelligence Theory, Systems and Applications (CCITSA), 2015 First International Conference on. IEEE, 2015, pp. 136–141.

[29] T. Mizuya, M. Okuda, and T. Nagao, “A case study of data acquisition from field devices using opc ua and mqtt,” 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), 2017.

[30] W. Kim and M. Sung, “Opc-ua communication framework for plc-based industrial iot applications,” in Internet-of-Things Design and Implementation (IoTDI), 2017 IEEE/ACM Second International Conference on. IEEE, 2017, pp. 327–328.

Víctor Vimos. Nacido en Chunchi, Ecuador en 1986. Ingeniero de Sistamas de la Universidad Politécnica Salesiana, miembro del Grupo de Investigación en Cloud Computing Smart Cities & High Performance Computing de la Universidad Politécnica Salesiana. Tiene varias certificaciones Cisco y VMware como: CCNA Routing and Switching, UCCX Specialist Cisco Certification, CCNA Collaboration, CCNP Collaboration Cisco Certification, CCIE wr Collaboration Cisco Certification, VMware Certified Professional. Sus áreas de investigación de interés son: IoT, Ciudades Inteligentes, Redes de Sensores, Herramientas de Colaboración Email: vvimos@est.ups.edu.ec.

Erwin J. Sacoto Cabrera. Nacido en Azogues, Ecuador en 1976. Es Ingeniero Electrónico de la Universidad Politécnica Salesiana desde 2000; Especialista y Magister en Derecho y Gestión de las Telecomunicaciones de la Universidad Andina Simón Bolívar en 2010. En la actualidad es miembro del Grupo de Investigación en Cloud Computing Smart Cities & High Performance Computing de la Universidad Politécnica Salesiana y Docente Investigador de la Carrera de Ingeniería de Sistemas en la misma Universidad. Las áreas de Investigación de interés son: IoT, Ciudades Inteligentes, Redes 4G y 5G. Email: esacoto@ups.edu.ec.