

CLUSTER DE MONITOREO DE TEMPERATURA Y HUMEDAD USANDO INTERNET DE LAS COSAS Y EL PROTOCOLO MQTT

Área técnica: Sistemas Digitales y Computación

José Ignacio Vega Luna¹, José Francisco Cosme Aceves¹, Gerardo Salgado Guzmán¹, Francisco Javier Sánchez Rangel¹, Víctor Noé Tapia Vargas¹, Víctor Gonzalo Rodríguez Tapia¹, Edgar Alejandro Andrade González²

¹Área de Sistemas Digitales, ²Área de Comunicaciones, Departamento de Electrónica, Universidad Autónoma Metropolitana-Azcapotzalco.

Av. San Pablo 420, Col. Nueva El Rosario, Alcaldía Azcapotzalco, C.P. 02128, Cd. de México.

Tel. 52-55-53189030

vlji@azc.uam.mx

RESUMEN.

Se presenta un cluster o red de clientes MQTT que permite monitorear la temperatura y humedad a través de la Internet de las Cosas. El objetivo fue diseñar un cluster para medir la temperatura y humedad en un laboratorio de cómputo y publiquen los valores a un agente privado MQTT. Estos parámetros pueden visualizarse en la interfaz de usuario que se ejecuta en un teléfono móvil. Los clientes MQTT constan de un sistema embebido ESP32 y un sensor DHT22. El agente MQTT se implementó usando el software Mosquitto, el cual se ejecuta en una computadora con sistema operativo Linux-Ubuntu. Las pruebas realizadas mostraron que la precisión del sistema en la medida de la temperatura es menor a 0.5°C y en la medida de la humedad es menor a 2%.

Palabras Clave: ESP32, Internet de las Cosas, Mosquitto, MQTT

ABSTRACT.

This paper presents a cluster or network of MQTT clients that allows monitoring temperature and humidity through the Internet of Things. The objective was to design a cluster or network of clients that measure temperature and humidity in a computer laboratory and publish the values to a private MQTT agent. These parameters can be displayed in the user interface running on a mobile phone. MQTT clients are composed of an ESP32 embedded system and a DHT22 sensor. The MQTT agent was implemented using Mosquitto software, which runs on a computer with Linux-Ubuntu operating system. The tests carried out showed that the precision of the system in measuring temperature is less than 0.5°C and in measuring humidity it is less than 2%.

Keywords: ESP32, Internet Of Things, Mosquitto, MQTT

1. INTRODUCTION

La Internet de las Cosas (IoT) ha cambiado la forma mediante la cual las personas interactúan con diversos dispositivos en el hogar, en oficinas, en procesos industriales, en centros de salud, en escuelas y en ciudades inteligentes, entre otros ambientes. En la actualidad, una cantidad importante de aplicaciones y dispositivos de IoT usan los servicios que proporcionan los servidores de IoT en la nube a través del protocolo HTTP. Este protocolo se denomina sin estado ya que los dispositivos cliente deben establecer una conexión con el servidor cada vez que necesiten enviar o recibir datos. El protocolo HTTP puede transferir información multimedia para transmitir grandes cantidades de datos, sin embargo, también se usa en aplicaciones de IoT con sensores y actuadores donde no se necesita un ancho de banda grande. Éstos, intercambian información con el servidor periódica o eventualmente y las

conexiones del protocolo HTTP consumen cantidades importantes de tiempo, ancho de banda de la red y alimentación eléctrica. Una solución para hacer más eficiente este tipo de conexiones es usar el protocolo MQTT [1].

El protocolo de Transporte y Telemetría MQ (MQTT-MQ Telemetry Transport) está basado en el modelo de mensajes cliente-servidor y la funcionalidad publicar/suscribir. Es un protocolo ligero, abierto, simple y fácil de implementar. Se utiliza en aplicaciones máquina a máquina (M2M), de IoT y de IoT Industriales (IIoT) ya que usa un ancho de banda reducido y pocos recursos de procesamiento. El MQTT emplea un formato de mensajes binarios para la comunicación entre los clientes y un agente, lo opuesto a los protocolos basados en texto como el HTTP y el SMTP [2]. El protocolo MQTT tiene su origen en 1999 cuando IBM lo utilizó para monitorear tuberías de petróleo vía satélite y estaba basado en el sistema MQ de IBM, de ahí el nombre. Ha sido tal su éxito por las ventajas que presenta para el monitoreo remoto de variables de ambiente y procesos en diversos campos, que en 2010 IBM hizo disponible y libre la versión 3.1 de MQTT.

Los principales componentes funcionales del protocolo MQTT son los clientes y el agente. Un cliente es un dispositivo que ejecuta una biblioteca o rutina de software MQTT para conectarse a un agente a través de la red. El modelo de mensajería de MQTT se fundamenta en las acciones de publicar y suscribir. Un cliente puede publicar mensajes a un agente o recibir mensajes a los que está suscrito desde un agente. El agente es el dispositivo intermediario responsable de recibir los mensajes publicados por los clientes, filtrarlos y distribuirlos a los clientes suscriptores. El agente controla además la autenticación, la autorización y la sesión de los clientes. El cliente puede ser un dispositivo basado en un microcontrolador, un sistema embebido, una computadora, un dispositivo móvil o un servidor que envía publicaciones al agente o recibe, por medio de publicaciones del agente, la información a mostrar en la interfaz y aplicaciones de usuario que se ejecutan en computadoras o dispositivos móviles de monitoreo y control [3]. Las bibliotecas de MQTT del cliente son módulos de software que se ejecutan sobre los protocolos de red TCP e IP [4]. Pueden usar redes de tipo celular, ethernet, WiFi, Bluetooth LE, LoRa y Zigbee, estas tres últimas necesitan una puerta de enlace para la conexión a la Internet. Los mensajes de MQTT son de formato binario y de menor tamaño que los del protocolo HTTP que son de texto plano.

Por ejemplo, un cliente puede ser un sistema embebido que integre un sensor de temperatura y humedad y publique los mensajes con el valor de estos parámetros ambientales al agente. Otro cliente puede ser una computadora o un dispositivo móvil suscrito al agente y que reciba mensajes publicados de la temperatura y la humedad para mostrarlos en la interfaz de usuario, como se muestra en la Figura 1.

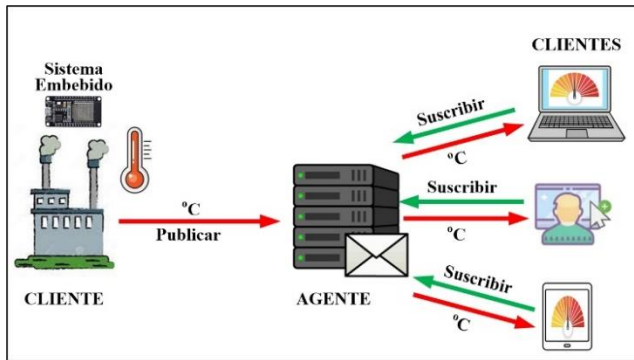


Fig. 1. Intercambio de mensajes con el protocolo MQTT

El objetivo de este trabajo fue diseñar una red de cinco clientes que midan la temperatura y humedad en un laboratorio de cómputo, publiquen los valores a un agente privado MQTT y el valor de estos parámetros pueda visualizarse en la interfaz de usuario que se ejecute en un teléfono móvil. La justificación del trabajo fue la necesidad de contar con un sistema de monitoreo remoto, de dos variables de ambiente usando un protocolo de comunicación más eficiente en el uso del ancho de banda de la red que el HTTP. A continuación, se presenta una revisión de la literatura y estado del arte de trabajos relacionados con el que aquí se presenta, posteriormente, se explica el desarrollo de la red de monitoreo, las pruebas realizadas y los resultados obtenidos y finalmente, se exponen las conclusiones obtenidas. El protocolo MQTT se ha convertido en el estándar de facto para aplicaciones de IoT. Durante los últimos años, se ha realizado una cantidad importante de investigaciones usando este protocolo en soluciones de IoT. Estos trabajos incluyen aplicaciones en: el campo de ciudades inteligentes y en ambientes de difícil acceso [5], en la agricultura [6], en el cuidado de la salud usando la llamada Internet de las Cosas Médicas (IoMT-Internet of Medical Things) [7], en el control de vehículos aéreos no tripulados [8], en la automatización de hogares y oficinas [9] y en el monitoreo y control de procesos industriales a través de la llamada IoT de las Cosas Robóticas [10], entre otros usos. Se han desarrollado también investigaciones que integran clusters de agentes [11], [12], [13]. En otros trabajos se han implementado algoritmos de búsqueda dinámica de rutas de calidad para la comunicación entre dispositivos MQTT [14]. El objetivo de las soluciones anteriores fue el monitoreo remoto de variables y procesos, así como el balanceo y la distribución de la carga de los clientes en los agentes.

La contribución de este trabajo es contar con sistema de monitoreo remoto de temperatura y humedad que permita

reaccionar oportunamente ante alguna eventualidad cuando el valor de estos parámetros alcance un valor crítico. Otra aportación es que los nodos del cluster de monitoreo son dispositivos compactos y de bajo costo cuya instalación no es intrusiva y no es necesario modificar la infraestructura de lugar.

2. DESARROLLO

El cluster de monitoreo se instaló en un laboratorio de cómputo. La metodología seguida consistió en implementarlo a través de cinco clientes MQTT basados en la misma arquitectura. Cuatro de ellos se instalaron, cada uno, en las esquinas del laboratorio y el otro en la parte central, como se muestra en la Figura 2.

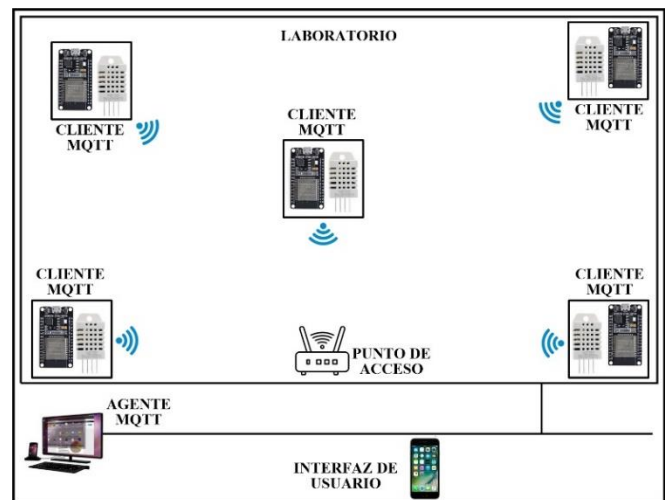


Fig. 2. Ubicación de la red de monitoreo

Los clientes se encargan de leer periódicamente los valores de temperatura y humedad y publicarlos al servidor MQTT. El periodo de monitoreo es configurable y por defecto es 60 segundos.

2.1. Los clientes MQTT.

La arquitectura de los clientes MQTT se desarrolló usando el sistema embebido ESP32 y el sensor de temperatura y humedad DHT22, como se indica en el diagrama de la Figura 3.

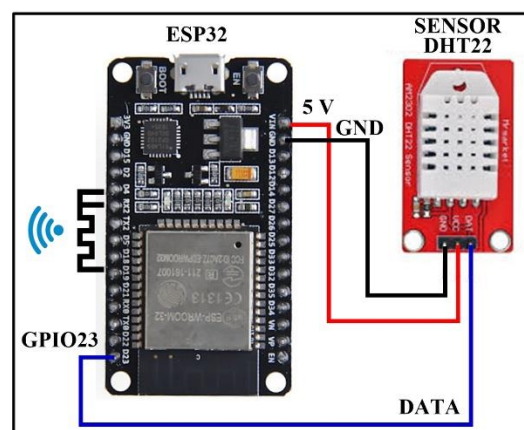


Fig. 3. Arquitectura de los nodos esclavos

El sistema ESP32 es un dispositivo de bajo consumo de energía. Se usa para ejecutar programas realizados en varias plataformas de desarrollo, como por ejemplo el entorno de desarrollo integrado (IDE-Integrated Development Environment) de Arduino, o usando alguno de los lenguajes de programación para microcontroladores, como MycroPython. Se eligió el ESP32 para conformar la arquitectura de los clientes, debido a que incorpora los recursos requeridos y suficientes para el monitoreo realizado, lo cual permitió reducir el tamaño y costo del sistema, obteniendo una buena relación costo-beneficio. Los principales recursos con los que cuenta este sistema embebido son los siguientes: un microcontrolador Tensilica LX6 de dos núcleos a 240 MHz, memoria flash de 16 MB, memoria RAM de 520 KB, interfaces WiFi 802.11b/g/n, Bluetooth clásico y LE con antena integrada, 3 puertos UART, 4 puertos SPI, 2 puertos I2C, 34 entradas/salidas de propósito general (GPIO-General Purpose Input/Output), un convertidor analógico-digital de 12 bits y 18 canales, soporte para interfaz de memoria SD y un cargador para batería de ion de litio.

Para medir el valor de la temperatura y la humedad se utilizó el sensor DHT22. Este dispositivo, de bajo costo y tamaño compacto, incorpora un sensor capacitivo de humedad y un termistor para medir la temperatura ambiente. Incluye un convertidor analógico-digital, un microcontrolador y una interface 1-wire para proporcionar el valor de temperatura y humedad relativa. El voltaje de alimentación del DHT22 es de 3.3V a 5.5V, el rango de temperatura que puede medir es de -40°C a 80°C, con una precisión de $\pm 0.5^{\circ}\text{C}$ y una resolución de 0.1°C . Respecto a la humedad, el DHT22 trabaja en el rango de 0 a 100% de humedad relativa (RH), ofrece una precisión de 2% RH y una resolución de 0.1% RH. Cuenta con 4 terminales, dos son tierra, otra es para alimentación y la última es la línea de datos de salida serie 1-wire mediante la cual entrega los valores digitales de temperatura y humedad en una palabra de 40 bits. Esta última terminal se conectó a la entrada GPIO23 del sistema embebido ESP32. De los 40 bits, los primeros 16 corresponden al valor de humedad relativa, los siguientes 16 son el valor de temperatura en grados Celsius, 8 para la parte entera y 8 para la parte decimal, y los últimos 8 bits indican el checksum. La señal del bus 1-wire del DHT22 tiene un alcance de 20 metros nominalmente.

2.2. La programación de los clientes.

Se desarrollaron los algoritmos para crear el canal de comunicación para el intercambio de información entre los clientes y el agente privado usando el protocolo MQTT y se implementaron estos algoritmos para convertirlos a rutinas o bibliotecas de funciones invocadas en la programación realizada. El intercambio de información usa dos mensajes, uno para el tópico de la temperatura y el otro para el de la humedad. La programación se ubicó en la capa de aplicación del modelo OSI, situada arriba de la pila TCP/IP como se indica en la Figura 4.

La programación de los clientes MQTT se realizó usando el IDE Arduino, ya que es un software libre, de código abierto y

multiplataforma que permite el desarrollo de programas para sistemas embebidos ESP32. Se pueden usar los lenguajes C y C++ para la creación de programas y una serie de bibliotecas incorporadas al IDE o de terceros que pueden integrarse para el desarrollo de una variedad de aplicaciones.

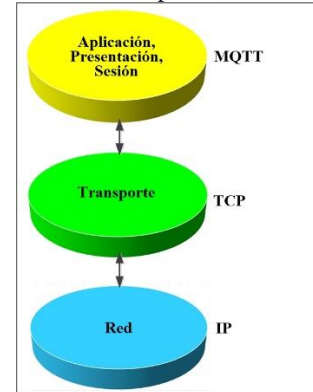


Fig. 4. Ubicación del protocolo MQTT en el modelo OSI

Otra razón por la cual se empleó el IDE de Arduino fue por la facilidad de construir la programación, la cual se elaboró basada en el diagrama de flujo mostrado en la Figura 5.

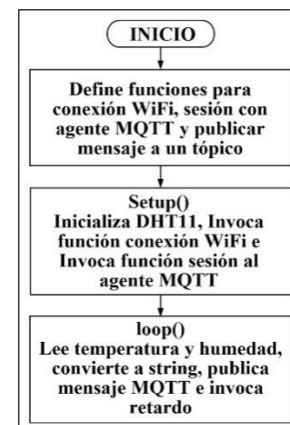


Fig. 5. Diagrama de flujo de la programación de los clientes MQTT

Al iniciar el programa, realiza las siguientes tareas: 1-Importa las rutinas o bibliotecas para: la conexión y comunicación WiFi a la Internet, el intercambio de mensajes entre el cliente y el agente MQTT y para la configuración y comunicación con el sensor DHT22 y 2-Inicializa las variables que definen: las credenciales de la red WiFi, la dirección IP y el puerto del agente MQTT, el nombre de los tópicos donde serán publicados los valores de temperatura y humedad, la terminal GPIO donde se encuentra conectado el sensor DHT22 y el periodo de tiempo de las publicaciones. Posteriormente, define las funciones para: la conexión al punto de acceso WiFi, establecer la sesión con el agente MQTT y para publicar un mensaje a un tópico MQTT. A continuación, define la primera función básica del programa, la de *setup()*. Esta función lleva a cabo las acciones siguientes: A-Inicializa el sensor DHT22, B-Establece la conexión WiFi y

C-Realiza la conexión e inicio de la sesión con el agente MQTT. Finalmente, en la segunda función básica del programa, el *loop()*, entra a un ciclo continuo, donde ejecuta tres tareas: lee los valores de temperatura y humedad, convierte los valores anteriores a una variable tipo string y publica estos valores al agente MQTT cada 30 segundos.

2.3. El agente MQTT.

El agente MQTT se instaló y configuró en una computadora con el sistema operativo Linux-Ubuntu 22.04.1 LTS. Esta versión es la última estable que tiene soporte hasta abril de 2027. Fue seleccionado este sistema operativo por las siguientes razones: a-Es una distribución de Linux de software libre y código abierto basada en Debian, b-Es una de las distribuciones de Linux más populares y reconocidas a nivel mundial, c-Puede ejecutarse en una computadora de escritorio y d-Es una distribución predecible cuyas versiones actualizadas son lanzadas periódicamente, comúnmente, cada seis meses. El agente MQTT se implementó usando el software de Mosquitto 2.0.18. En este trabajo se denominó al agente como privado, ya que, aun cuando está conectado a la Internet, solo puede ser accedido por los clientes del cluster implantados en este sistema. Adicionalmente, su administración está bajo el control de los participantes de este trabajo y no es para acceso público. El objetivo de lo anterior es tener el dominio completo del agente en lo referente a configuración, uso y mantenimiento.

Se usó Eclipse Mosquitto dado que es un software de licencia pública que permite implementar agentes de mensajes del protocolo MQTT versiones 5.0, 3.1.1 y 3.1. Es ligero e ideal para usarse en dispositivos de bajo consumo de energía y en servidores, lo cual es útil en aplicaciones de IoT. Mosquitto es parte de la Fundación Eclipse y pertenece al proyecto iot.eclipse.org. Proporciona una biblioteca de funciones en C que permiten el desarrollo de clientes MQTT a través de una línea de comandos.

Con el agente instalado, se creó un programa en Shell o Shell script que se ejecuta permanentemente en la computadora y realiza las tareas siguientes: 1-Arranca al agente MQTT, 2-Publica en los tópicos MQTT de la temperatura y humedad los valores recibidos cada 30 segundos por los clientes y almacena en un archivo de texto estos valores. En la computadora donde se ejecuta el agente, se crearon cinco directorios, uno para cada cliente, y en cada directorio dos subdirectorios, uno para la temperatura y otro para la humedad. En cada subdirectorio existe un archivo para cada día de la semana donde se almacenan los valores de temperatura y humedad recibidos. Esto sirve para mantener un registro histórico de ambos parámetros.

2.4. La interfaz de usuario.

La interfaz de usuario permite visualizar los valores de temperatura y humedad leídos y transmitidos por los clientes al agente MQTT. La interfaz es un cliente MQTT suscrito al tópico usado para cada uno de los parámetros anteriores que publica el agente. Este cliente es la aplicación MyMQTT, descargada de la Internet y configurada en un teléfono móvil

Android, como se muestra Figura 6. No tiene costo y es sencilla de configurar y usar.



Fig. 6. La interfaz de usuario

3. RESULTADOS

Se realizaron dos grupos de pruebas. El primero tuvo como objetivo determinar la precisión de la medida de temperatura reportada por el sistema en la interfaz de usuario. En la realización de estas pruebas se usó un termómetro profesional infrarrojo Dwyer Omega Serie OS423 [15] del tipo usado en los centros de datos. El rango de medida del termómetro es de -60 a 860°C y ofrece una precisión de $\pm 1^{\circ}\text{C}$. El valor de la temperatura indicada por el termómetro se usó como referencia en estas pruebas. Se realizaron 22 pruebas en las cuales se varió la temperatura del laboratorio, por medio del sistema de calefacción y aire acondicionado, en el rango de 13°C a 65°C . En cada prueba se determinó la precisión comparando el valor de referencia de la temperatura con la media de temperatura reportada por los cinco clientes MQTT en la interfaz de usuario. El fabricante del DHT22 indica que, el sensor ofrece una precisión nominal de temperatura de $\pm 0.5^{\circ}\text{C}$. Los resultados obtenidos en estas pruebas mostraron que la precisión real es menor a 0.5°C , la cual se mantuvo hasta los 50°C . Después de ese valor, al aumentar la temperatura, la diferencia entre el valor indicado en el termómetro respecto al reportado en la interfaz aumentó proporcionalmente, variando la precisión, como se muestra en la gráfica de la Figura 7.

El objetivo del segundo grupo de pruebas fue similar al del primero, consistió en determinar la precisión de la medida de humedad relativa reportada por el sistema en la interfaz de usuario. En estas pruebas se utilizó como medida de referencia la obtenida con un medidor de humedad absoluta PCE Instruments modelo PCE-PMI 4 [16].

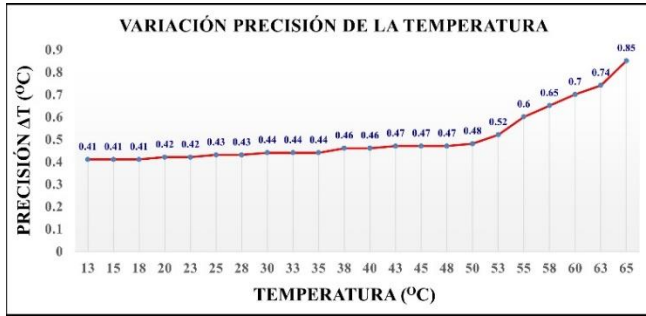


Fig. 7. Precisión de la medida de temperatura

Este dispositivo usa 8 electrodos para medir el valor de la humedad relativa en el rango de 0 a 100% y presenta una precisión de $\pm 0.5\%$. Se llevaron a cabo 22 pruebas realizadas a la par con las del primer grupo. Es decir, cuando se varió la temperatura del laboratorio se midió la humedad relativa. En cada prueba se determinó la precisión comparando el valor de referencia de la humedad con el de la media reportada por los clientes MQTT en la interfaz de usuario. El fabricante del DHT22 indica que el sensor ofrece una precisión nominal de la humedad de 2%. Los resultados obtenidos en estas pruebas mostraron que la precisión real es menor a 2%, la cual se mantuvo hasta que la humedad relativa del ambiente fue superior a los 28°C. Después de ese valor, al aumentar la humedad, el valor de la precisión varió proporcionalmente como se muestra en la gráfica de la Figura 8.

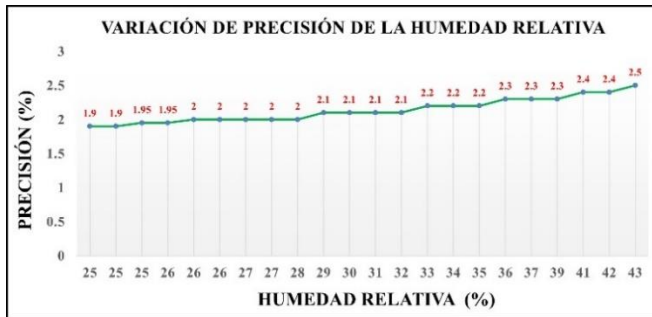


Fig. 8. Precisión de la medida de humedad

En la Tabla 1 se presenta un comparativo de la velocidad, latencia a la plataforma de IoT y el alcance logrado en cluster aquí presentado respecto a las redes de monitoreo de otros trabajos consultados. El tiempo de latencia es el existente entre el sistema de monitoreo y la plataforma. El alcance es logrado entre el nodo central de la red y los sensores. Cuando en el trabajo no se especifica (NE) un parámetro, se indica el valor nominal del transceptor usado.

Tabla 1. Comparativo del cluster con otras redes de monitoreo

Red	Protocolo	Velocidad	Latencia (ms)	Alcance (m)
WiFi, este trabajo	MQTT	200 Mbps	5	35

XBee [17]	HTTP	250 kbps	10	60
XBee [18]	HTTP	10 a 200 Kbps	75	4,000
LoRa [19]	MQTT	NE, 300 Kbps	20	NE, 10,000

4. CONCLUSIONES

Se obtuvo un sistema de monitoreo remoto de dos variables de ambiente, la temperatura y la humedad, usando el protocolo MQTT y la IoT. El sistema está integrado por un agente y cinco clientes MQTT. El agente es privado y se ejecuta en una computadora con sistema operativo Linux-Ubuntu y el software de Mosquitto. Los clientes se basan en un sistema embebido ESP32. Por medio de la interfaz de usuario, la cual se ejecuta en un teléfono móvil, pueden visualizarse los valores de las variables de ambiente monitoreadas. Esto permite al usuario tomar las decisiones necesarias en caso de alguna eventualidad. El uso del sistema ESP32 facilitó el desarrollo de las rutinas de la programación de los clientes encargados de realizar el intercambio de mensajes MQTT entre los clientes y el agente. Con el ESP32 se desarrollaron los clientes cuyo tamaño compacto y bajo costo facilita su instalación y mantenimiento. Se monitorean dos variables, no obstante, el sistema puede monitorear otros parámetros, integrando los sensores adecuados y las rutinas de software necesarias al sistema.

Los mecanismos de seguridad con los que cuenta el MQTT para la comunicación de los mensajes son mínimos. Sin embargo, se puede integrar a las rutinas de la programación de este sistema el protocolo criptográfico de Seguridad en la Capa de Transporte (TLS-Transport Layer Security). Con esto, se protegerá la información transmitida entre los clientes y el agente MQTT en redes públicas y la Internet. El TLS protege las comunicaciones MQTT contra las escuchas y accesos no autorizados cifrando los datos de la transmisión. Esta mejora al sistema es un trabajo a futuro, con el cual se tiene planeado desarrollar la segunda versión del cluster presentado en este trabajo.

REFERENCIAS

- [1] B. Mishra and A. Kertesz, "The Use of MQTT in M2M and IoT Systems: A Survey," in *IEEE Access*, vol. 8, pp. 201071-201086, 2020, doi: 10.1109/ACCESS.2020.3035849.
- [2] C. -S. Park and H. -M. Nam, "Security Architecture and Protocols for Secure MQTT-SN," in *IEEE Access*, vol. 8, pp. 226422-226436, 2020, doi: 10.1109/ACCESS.2020.3045441.
- [3] MQTT & MQTT 5 Essentials, HiveMQ, Ergoldingerstr. 2a 84030 Landshut/Germany, © HiveMQ GmbH 2020
- [4] N. -Z. Wang and H. -Y. Chien, "Design and Implementation of MQTT-Based Over The Air Updating Against Curious Brokers," in *IEEE Internet of Things Journal* (Early Access), pp. 1-1, doi: 10.1109/IIOT.2023.3327447.
- [5] M. Pradhan, "Federation Based on MQTT for Urban Humanitarian Assistance and Disaster Recovery Operations," in *IEEE Communications Magazine*, vol. 59, no. 2, pp. 43-49, February 2021, doi: 10.1109/MCOM.001.2000937.
- [6] D. T. Tran, H. S. Le and J. -H. Huh, "Building an Automatic Irrigation Fertilization System for Smart Farm in Greenhouse," in *IEEE*

- Transactions on Consumer Electronics (Early Access), pp. 1-1, doi: 10.1109/TCE.2023.3304554.
- [7] N. K. Dewangan and P. Chandrakar, "Patient-Centric Token-Based Healthcare Blockchain Implementation Using Secure Internet of Medical Things," in IEEE Transactions on Computational Social Systems, pp. 3109-3119, 2022, doi: 10.1109/TCSS.2022.3194872.
 - [8] F. Xiong, A. Li, H. Wang and L. Tang, "An SDN-MQTT Based Communication System for Battlefield UAV Swarms," in IEEE Communications Magazine, vol. 57, no. 8, pp. 41-47, August 2019, doi: 10.1109/MCOM.2019.1900291.
 - [9] J. A. Cujilema Paguay, G. A. Hidalgo Brito, D. L. Hernandez Rojas and J. J. Cartuche Calva, "Secure home automation system based on ESP-NOW mesh network, MQTT and Home Assistant platform," in IEEE Latin America Transactions, vol. 21, no. 7, pp. 829-838, July 2023, doi: 10.1109/TLA.2023.10244182.
 - [10] D. Yoshino, Y. Watanobe and K. Naruse, "A Highly Reliable Communication System for Internet of Robotic Things and Implementation in RT-Middleware With AMQP Communication Interfaces," in IEEE Access, vol. 9, pp. 167229-167241, 2021, doi: 10.1109/ACCESS.2021.3136855.
 - [11] K. Kosaka, Y. Noda, T. Yokotani and K. Ishibashi, "Implementation and Evaluation of the Control Mechanism Among Distributed MQTT Brokers," in IEEE Access, vol. 11, pp. 134211-134216, 2023, doi: 10.1109/ACCESS.2023.3335273.
 - [12] E. Longo, A. E. C. Redondi, M. Cesana and P. Manzoni, "BORDER: A Benchmarking Framework for Distributed MQTT Brokers," in IEEE Internet of Things Journal, vol. 9, no. 18, pp. 17728-17740, 15 Sept.15, 2022, doi: 10.1109/JIOT.2022.3155872.
 - [13] A. Detti, L. Funari and N. Blefari-Melazzi, "Sub-Linear Scalability of MQTT Clusters in Topic-Based Publish-Subscribe Applications," in IEEE Transactions on Network and Service Management, vol. 17, no. 3, pp. 1954-1968, Sept. 2020, doi: 10.1109/TNSM.2020.3003535.
 - [14] C. Zunino, G. Cena, S. Scanzio and A. Valenzano, "Adaptive Seamless Redundancy to Achieve Highly Dependable MQTT Communication," in IEEE Transactions on Industrial Informatics, vol. 20, no. 1, pp. 984-994, Jan. 2024, doi: 10.1109/TII.2023.3271708.
 - [15] OMEGA Engineering Inc., "Termómetro profesional infrarrojo Serie OS423", [en línea], disponible en: <https://mx.omega.com/temperature/pdf/OS423-LS.pdf>, 23 de mayo de 2024.
 - [16] PCE Instruments, "Medidor de humedad absoluta PCE-PMI 4", [en línea], disponible en: https://www.pce-instruments.com/mexico/api/getartfile?_fmr=2312711&_dsp=inline, 23 de mayo de 2024.
 - [17] S. I. Ossa, "Monitoreo y control de variables ambientales mediante una red inalámbrica para agricultura de precisión en invernaderos", Revista Vector, vol. 12, pp. 51-60, 2017, doi: 10.17151/vect.2017.12.6.
 - [18] S. N. Muñoz y A. F. Toquica, "Red de sensores inalámbricos para el monitoreo de variables microclimáticas en el Relicto Vegetal Cedro Rosado", Scientia Et Technica, vol. 23, no. 4, pp. 501-510, 2018.
 - [19] J. Herrera-Rubio y V. Ortiz-E, "Implementación de un sistema de monitoreo y control con tecnología IoT para determinar el comportamiento de las variables ambientales en la avicultura", Investigación e Innovación en Ingenierías, vol. 10, no. 1, pp. 30-41, 2022, doi: <https://doi.org/10.17081/invinno.10.1.5016>.