UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN TELEMATICA

MENCIÓN: Calidad en el Servicio

Resolución: RPC-SO-19-No.300-2016

PROYECTO DE TITULACIÓN EN OPCIÓN AL GRADO DE MAGISTER

Título del proyecto:
Evaluación de rendimiento entre el estándar de mensajería MQTT y la plataforma Firebase a través de un prototipo, modelo de comunicación IoT.
Línea de Investigación:
Telecomunicaciones
Campo amplio de conocimiento:
Telecomunicaciones y Sistemas Informáticos aplicados a la producción y la sociedad
Autor/a:
Carlos Alberto Rivadeneira Proaño
Tutor/a:
Mg. Renato Mauricio Toasa Guachi

Quito – Ecuador

2020

APROBACIÓN DEL TUTOR

Yo, Renato Mauricio Toasa Guachi con C.I: 1804724167 en mi calidad de Tutor del proyecto de investigación titulado: Evaluación de rendimiento entre el estándar de mensajería MQTT y la plataforma Firebase a través de un prototipo, modelo de comunicación IoT.

Elaborado por: Carlos Alberto Rivadeneira Proaño, de C.I: 1003315981, estudiante de la Maestría: Telemática, mención: Calidad en el Servicio de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL)**, como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar que luego de haber orientado, analizado y revisado el trabajo de titulación, lo apruebo en todas sus partes.

Quito D.M., _____ de 2020

_____

Firma

Tabla de contenidos

Índice de tablas

Índice de figuras

INFORMACIÓN GENERAL

Contextualización del tema

El Internet de las cosas (IoT) proporciona la capacidad de conectar un gran número de cosas o dispositivos a través de Internet. Estas cosas o dispositivos tienen identidades únicas. El IoT crea un entorno inteligente conectando dispositivos con Internet y equipándolos con la capacidad de reunir e intercambiar datos. Estos dispositivos o aparatos suelen estar conectados con micro controladores, sensores y una conectividad a Internet. El hecho de que Internet esté presente al mismo tiempo en todas partes permite que la adopción masiva de esta tecnología sea más factible. Dado su tamaño y coste, los sensores son fácilmente integrables en hogares, entornos de trabajo y lugares públicos. De esta manera, cualquier objeto se puede conectar y manifestarse en la Red. Además, IoT implica que todo objeto puede ser una fuente de datos. Esto está empezando a transformar la forma de hacer negocios, la organización del sector público y el día a día de millones de personas.

En el IoT se utilizan protocolos ligeros como MQTT para la transmisión de datos. Message Queuing Telemetry Transport (MQTT) es un protocolo de mensajería extremadamente liviano basado en la publicación y suscripción que puede ser usado con conexiones que tienen un ancho de banda muy bajo o conexiones que no son confiables. MQTT utiliza el método de publicación y suscripción para transferir la información a diferencia del patrón cliente-servidor. (Martin Niño, 2017)

Por otro lado, Firebase es un conjunto de herramientas orientadas a la creación de aplicaciones de alta calidad, dedicada al crecimiento de los usuarios y a generar mayores beneficios económicos. Su principal objetivo, es mejorar el rendimiento de las apps mediante la implementación de diversas funcionalidades que van a hacer de la aplicación en cuestión, mucho más manejable, segura y de fácil acceso para los usuarios. (Ruiz, 2017)

La principal ventaja de utilizar MQTT sobre Firebase es su ligereza. Es así como en una red muy lenta, el uso de Firebase puede ser una carga porque utiliza HTTP y un ancho de banda relativamente alto. De igual manera si una red es poco fiable, incluso una pequeña caída de señal puede interrumpir la comunicación con el dispositivo IoT. Pero

como MQTT es tan liviano, puede comunicarse en la mayoría de estos escenarios y ser usado muy fácilmente para transmitir las lecturas del sensor u dispositivo IoT. (Microsoft Azure, 2020)

Mientras tanto plataformas como Arduino, OpenMote o Raspberry Pi permiten conectar sensores para obtener información, cada vez es mayor la cantidad de datos a procesar y analizar. Es aquí donde nace el propósito para el presente proyecto de usar una base de datos en tiempo real para la comunicación entre las dos aplicaciones, permitiendo que estén conectadas a la base y puedan ser informadas de los cambios ejecutados, utilizando las API que facilita dicha base, permitiendo un control y análisis de la información dada por estos sensores, aportando además la flexibilidad y elasticidad de un sistema Cloud y funcionalidades como analítica en tiempo real, conexiones de cualquier ubicación incluido dispositivos móviles, integración con sistemas corporativos, entre otros. (Gharssellaoui, 2019)

Pregunta Problémica

¿Cómo determinar las herramientas más eficientes para el desarrollo de aplicativos IoT, con un enfoque en almacenamiento de datos y comunicación en tiempo real?

Objetivo general

Determinar las herramientas más eficientes para almacenamiento de datos y comunicación en tiempo real en el desarrollo de aplicativos IoT.

Objetivos específicos

- Contextualizar los fundamentos teóricos sobre los diferentes protocolos de comunicación IOT existentes para la industria 4.0.
- Desarrollar un aplicativo móvil integrando un sistema de comunicación IoT, prototipo para el control de encendido de luminaria y registro de temperatura/humedad.
- Determinar el escenario óptimo al utilizar Firebase y MQTT en aplicaciones IoT.

- Validar los resultados obtenidos mediante pruebas de rendimiento entre el estándar MQTT y Firebase.

Beneficiarios directos:

Empresas de Telecomunicaciones e IOT, Instituciones Educativas y Lectores de revistas de tecnología e innovación.

CAPÍTULO I: DESCRIPCIÓN DEL PROYECTO

1.1. Contextualización de fundamentos teóricos

Cuando se dialoga sobre IoT se hace referencia a los sensores u objetos de nuestro día a día que, a través de redes conectadas entre ellas, envían en tiempo real millones de datos a centros de interpretación y análisis para su posterior estudio y toma de decisiones. El enriquecimiento de los datos, la conectividad y la comunicación en tiempo real están cambiando la forma en que las empresas trabajan.

El Internet de las cosas hace uso de diferentes tecnologías y, aunque no haya una definición unificada del término, por norma general se le atribuyen las siguientes características según (Ionos, 2020):

- Recopilación, almacenamiento y procesamiento de datos.
- Comunicación con otros objetos.
- Interconexión.
- Ubicuidad.
- Automatización.
- Capacidad de aprendizaje.

Actualmente, existen algunos trabajos e investigaciones que involucran el análisis de rendimiento, evaluaciones de desempeño, tiempos de respuesta y carga de sistemas integrados de comunicación en tiempo real a través de los diferentes protocolos de comunicación IoT y plataformas Cloud existentes. A continuación, se presentan algunas de las investigaciones de mayor relevancia dentro del contexto del presente trabajo de titulación:

(Chimarro Amaguaña, 2020) menciona en su trabajo titulado *"Sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles"* cuyo trabajo tuvo como objetivo gestionar y establecer la teleoperación de un brazo robótico a través de una arquitectura de red Cloud computing con Firebase para la optimización de recursos y minimización de procesos de integración. En este trabajo de investigación se analiza en especial las

plataformas que integran los sistemas robóticos teleoperados en tiempo real, ya que actualmente estos sistemas requieren altos recursos económicos en desarrollo porque deben cumplir criterios como la interacción con dispositivos móviles, escalabilidad, versatilidad, baja latencia, robustez, control remoto, compatibilidad de hardware y por ello resulta complejo integrarlas. La teleoperación del brazo robótico se basó sobre la arquitectura Cloud Computing usando las API y servicios de Google Cloud Platform, permitiendo una interacción directa con Firebase. Estas plataformas tienen enlaces directos con dispositivos móviles y uso de protocolos de comunicación IoT, lo que permitió tener el control de todas las variables del robot de forma remota en tiempo real, carga y descarga de información de la base de datos, seguridad de la información y obtención de estadísticas sobre el uso de la aplicación. Como resultados se valoró el funcionamiento del sistema teleoperado obteniendo información sobre el tráfico tele-transmitido, tiempos de latencia, errores y eventos cuyos parámetros permitieron mejorar el rendimiento y la calidad del servicio de la aplicación.

El aporte para el presente trabajo es una guía para la elaboración de la arquitectura de comunicación tanto física como lógica entre el microcontrolador y el framework de desarrollo móvil, así como también una breve descripción de las plataformas Cloud de mayor popularidad en la actualidad, que Google junto con sus servicios ofrece.

(Alhomsi & Alsalemi, 2017) menciona en su trabajo titulado *"Real-Time Communication Network Using Firebase Cloud IoT Platform for ECMO Simulation"*, plantea implementar un sistema de simulación para médicos en tiempo real, el procedimiento demandó robustez y coordinación en la arquitectura de red por el cual se hizo uso del Internet de las cosas (IoT), el servicio Firebase como método de comunicación y se discutió las características como baja latencia, gestión de datos y compatibilidad de hardware. Firebase demostró ser una solución adecuada de comunicación que satisface el entorno urgente de entrenamiento médico porque cada unidad pudo acceder en tiempo real a la base de datos para almacenar y leer cualquier parámetro una vez que sea necesario y poder comprobar cambios que ocurrieron, tal técnica pudo aumentar la eficacia del sistema de comunicación. Se estableció una topología en estrella entre el dispositivo móvil (unidad) y Firebase (donde Firebase es el Hub) permitiendo que el sistema de comunicación se mantenga en operación incluso si alguna unidad en la red no

funcionara o tenga una mala conexión y además puedan ser fácilmente integrados con el sistema actual y poder establecer condiciones de la vida real.

Esta investigación apoya al presente proyecto ya que permite considerar las características que brinda el servicio Firebase junto a Realtime Database y la interacción que tiene con los dispositivos móviles como baja latencia, robustez, control remoto y compatibilidad de hardware.

Adicionalmente (Wu-Jeng & Chiaming , 2018) menciona en su trabajo titulado "JustIoT Internet of Things based on the Firebase real-time database" diseña un sistema de Internet de las cosas (llamado JustIoT) cuyo proyecto se divide principalmente en cuatro partes: back-end base de datos en tiempo real de Google Firebase, front-end SPA (Aplicación de una sola página) programa de monitoreo en la web (incluyendo aplicación de monitoreo móvil), software y hardware de control, y servidor de inteligencia que soporta conexiones MQTT y control de condiciones. En JustIoT, la página web de gestión basada en Angular está conectada a Firebase Realtime Database. El evento de modificación de datos de Firebase database puede provocar un enlace de datos Angular bidireccional para lograr un efecto de vínculo en tres direcciones e implementar una arquitectura sin servidor fácilmente. Los datos de Firebase database son leídos y escritos por los dispositivos de entrada (aplicaciones web, aplicaciones móviles y controladores) directamente. El servidor de inteligencia es un servidor MQTT que soporta conexiones de controladores incorporados relativamente débiles como el controlador de Arduino. El servidor inteligente puede ser considerado como un intermediario entre Firebase Realtime Database y controladores débiles, que realizan la transferencia de datos y comandos remotos.

Esta investigación contribuye al actual proyecto para considerar los desarrollos recientes de la revolución tecnológica actual, y tomar en cuenta las características que permiten integrar el software como servicio, usando la arquitectura de computación en la nube e implementación en plataformas de programación de lenguaje de código abierto y de bajo costo.

1.2. Problema a resolver

Al desarrollar aplicaciones concernientes al Internet de las cosas (IoT), el mayor problema que el programador está llamado a enfrentar es la comunicación entre el microcontrolador y el aplicativo. Existen varios métodos de comunicación directa que se pueden implementar tales como Bluetooth o NFC, pero si se quiere brindar mejor acceso remoto a la aplicación es necesario usar Internet para la comunicación entre ambos dispositivos. Cuando se inicia una comunicación a través de internet, se puede efectuar métodos comunes como HTTP y MQTT para comunicarse entre el dispositivo y el software, pero aun resultan un poco complicados de implementar para una aplicación simple. Para este inconveniente se pretende solucionar con Google Firebase ya que resulta ser una herramienta de fácil uso como medio de comunicación intermedio para dispositivos IoT, utilizando su poderosa base de datos en tiempo real y APIs.

El propósito principal e innovador de dicha aplicación es usar Firebase como intermediario de comunicación para dispositivos IoT, el cual ofrece muchos servicios en la nube desde la autenticación hasta el alojamiento de una aplicación web. Este proyecto se centrará en usar una base de datos en tiempo real para la comunicación entre las dos aplicaciones, permitiendo que las mismas que están conectadas a la base puedan ser notificadas de los cambios ejecutados, utilizando las API que proporciona dicha base, contribuyendo un control y análisis de la información dada por estos sensores, como además la flexibilidad y elasticidad de un sistema Cloud y funcionalidades como analítica en tiempo real, conexiones de cualquier ubicación incluido dispositivos móviles, integración con sistemas corporativos, entre otros.

1.3. Proceso de investigación

Con el objetivo de solventar la problemática planteada en primera instancia, esta investigación se estructura en base a un enfoque de investigación cuantitativa, teniendo en cuenta que el desempeño tecnológico de las plataformas de comunicación a examinarse puede ser evaluado empíricamente, mediante factores como la calidad del servicio, tolerancia a fallos, flexibilidad del modelo de servicio, entre otros. De acuerdo con (Hernández Escobar & Ramos Rodríguez, 2018) la investigación cuantitativa acoge una estrategia sistemática, objetiva y rigurosa para generar y afinar conocimientos. Las

variables que se tomaran en cuenta para el análisis, están definidas en los indicadores de resultado, apartado 1.5.

1.3.1. Metodología de Trabajo

De acuerdo al enfoque investigativo, es fundamental adoptar una metodología que se ajuste a la naturaleza del proyecto, por lo que es necesario utilizar el método experimental, que tiene como objetivo tratar de predecir eventos mediante la formulación de pruebas o hipótesis. (Gomez Bastar, 2019). La evaluación de rendimiento de plataformas de comunicación operará como variable independiente el sistema integrado o modelo IoT y como variable dependiente el estándar o plataforma utilizado para la comunicación. La técnica que permitirá poner en práctica la metodología seleccionada es el estudio de los datos obtenidos. La interpretación y el análisis revelarán la importancia de los mismos de acuerdo a los resultados obtenidos para establecer las conclusiones.

1.3.2. Técnica de recolección de datos

La técnica que se utiliza es la observación experimental, ya que es una técnica planificada, controlada, sujeta a comprobaciones y controles de validez y fiabilidad. De acuerdo con (Klaus , 2003) La observación se efectúa a través de un análisis atento que el investigador realiza sobre determinados objetos y hechos para llegar al conocimiento profundo de los mismos mediante la obtención de una serie de datos, que son imposibles alcanzar por otros medios. Es por esta razón que la observación brinda información permanente acerca de lo que ocurre en su entorno. Para esta investigación, se hará uso de ciertos instrumentos que se utilizan para la técnica de observación:

- Observación no sistematizada o registro anecdótico.
- Registro de actividad o seguimiento.
- Informe de observación.

1.3.3. Población y muestra

Ya que no se conoce con exactitud cuál es el valor de la población involucrada, se considerará los resultados obtenidos de las pruebas, aplicando la fórmula para el cálculo de la muestra de población infinita.

$$n = \frac{Z^2\, p * q}{d^2}$$

En donde:

p = probabilidad de éxito, en este caso es del 50% o 0.5.

q = probabilidad de fracaso (1 -p).

d = nivel de precisión absoluta (error máximo admisible en términos de proporción).

Z = nivel de confianza, en este caso se consideró el 95% correspondiente a 1.96 de acuerdo al anexo 1.

$$n = \frac{1.96^2 * \ 0.5 * 0.5}{0.05^2}$$

$$n = 384$$

Con un nivel de confianza del 95% y un margen de error del 5% el tamaño de la muestra para la investigación es de 384.

1.4. Vinculación con la sociedad

Desde un punto de vista de vinculación con la sociedad, la contribución que tendrá el proyecto será tanto para empresas de telecomunicaciones e IoT, instituciones educativas y lectores de revistas de ciencias aplicadas e innovación, haciéndoles ver como esta tecnología va encaminada hacia una gran variedad de ámbitos, tales como la industria, la salud, la educación y la energía, así como en el desarrollo de nuevas aplicaciones y la mejora de las mismas.

El proyecto permitirá conocer las ventajas, desventajas y diferencias que tienen ambas plataformas de comunicación relacionadas, a través de pruebas de rendimiento para recomendar mediante resultados y conclusiones cual herramienta es más eficiente y optima en diferentes escenarios IoT. De esta manera proyectos IoT venideros se beneficiarán al conocer por cual herramienta optar dependiendo.

1.5. Indicadores de resultados

Para evaluar el rendimiento entre las dos plataformas de comunicación señaladas se realizará una comparación de las mismas bajo condiciones similares, tomando como referencia los siguientes indicadores que (Nuñez & Toasa, 2020)señala:

- Tiempos de respuesta y carga.
- Pruebas de rendimiento.
- Tiempos de latencia.

CAPÍTULO II: PROPUESTA

Evaluación de rendimiento entre el estándar de mensajería MQTT y la plataforma Firebase a través de un prototipo, modelo de comunicación IoT.

Carlos Rivadeneira

Universidad Tecnológica Israel
Francisco Pizarro E4-142 y Marieta de Veintimilla Quito, Ecuador
carivadeneiraproano@gmail.com

Resumen

Al construir aplicaciones que se relacionan con el Internet de las cosas (IoT), se produce principalmente un problema que el desarrollador está llamado a enfrentar, se trata de una adecuada configuración de la comunicación entre el microcontrolador y la aplicación de software. Existen varios métodos de comunicación directa que se puede implementar ante dicho problema como Bluetooth/NFC, pero no permite un mejor acceso remoto a la aplicación, por lo que es necesario usar Internet para la comunicación. En este contexto, este trabajo propone realizar una evaluación de desempeño de dos poderosas herramientas de comunicación utilizadas actualmente en IoT, como son MQTT y Firebase; se tomaron criterios de rendimiento y la velocidad del tiempo de respuesta para la evaluación y comparativa entre ambas plataformas a través de herramientas para pruebas de stress y servicios de monitoreo de desempeño propios de Firebase y Arduino. Como resultado final se dedujo que ambos garantizan fiabilidad y calidad en la transmisión, administración y uso de la data.

Palabras claves: MQTT, Firebase, IoT, Realtime database.

Abstract

When building applications that are related to the Internet of Things (IoT), there is mainly a problem that the developer is called to face, it is a proper configuration of the communication between the microcontroller and the software application. There are several methods of direct communication that can be implemented to this problem as Bluetooth/NFC, but it does not allow a better remote access to the application, so it is necessary to use the Internet for communication. In this context, this work proposes to make a performance evaluation of two powerful communication tools currently used in IoT, such as MQTT and Firebase; performance criteria and response time speed were taken for the evaluation and comparison between both platforms through tools for stress testing and performance monitoring services of Firebase and Arduino. As a final result, it was deduced that both guarantee reliability and quality in the transmission, administration and use of the data.

Keywords: MQTT, Firebase, IoT, Realtime database.

2.1. Introducción

En IoT se usan protocolos livianos como MQTT para la transmisión de datos. Message Queuing Telemetry Transport (MQTT) es un protocolo de mensajería extremadamente ligero basado en la publicación y suscripción que puede ser usado con conexiones que tienen un ancho de banda muy bajo o conexiones que no son confiables (Carpio Santos, 2018). En lugar de utilizar el patrón común de cliente-servidor, MQTT utiliza el método de publicación y suscripción para transferir la información. En este método hay dos entidades principales: MQTT Bróker y el Cliente de MQTT, el bróker puede ser nombrado como la entidad central del sistema, el cual solo puede recibir mensajes de los clientes y mas no permitir que los clientes envíen mensajes entre ellos. El envío de mensajes se conoce como “publicación”, estos mensajes están compuestos de un tema y cuerpo. El bróker filtra los mensajes enviados por los clientes y los reenvía. Para recibir un determinado mensaje hay que suscribirse a un determinado tema y así recibir todos los mensajes publicados bajo ese tema en específico, independientemente del cliente que los haya enviado (GoldenMace IT Solutions, 2019).

La principal ventaja de usar MQTT sobre Firebase es que es muy ligero. Así que, en una red muy lenta, el uso de Firebase puede ser una carga porque utiliza HTTP y un ancho de banda relativamente alto. Si una red es muy poco fiable, incluso una pequeña caída de señal puede interrumpir la comunicación con el dispositivo IoT. Pero debido a que MQTT es tan liviano, puede comunicarse en la mayoría de estos escenarios y puede ser usado muy fácilmente para transmitir las lecturas del sensor u dispositivo IoT (Sparkfun, 2020). Mientras que plataformas como Arduino, OpenMote o Raspberry Pi permiten conectar sensores para adquirir información, cada vez es mayor la cantidad de datos a procesar y analizar. De este punto de partida nace el propósito para esta aplicación de usar una base de datos en tiempo real para la comunicación entre las dos aplicaciones, permitiendo que las aplicaciones que están conectadas a la base puedan ser notificadas de los cambios ejecutados, utilizando las API que proporciona dicha base, permitiendo un control y análisis de la información dada por estos sensores, aportando además la flexibilidad y elasticidad de un sistema Cloud y funcionalidades como analítica en tiempo real, conexiones de cualquier ubicación incluido dispositivos móviles, integración con sistemas corporativos, entre otros.

Debido a los problemas de comunicación que existe entre un dispositivo y su aplicación es necesario buscar una solución que optimice y facilite el acceso a través de Internet. Para este proyecto se pretende simular un prototipo modelo de comunicación IoT usando Android Studio como ambiente de desarrollo móvil y como plataforma de hardware la placa NodeMCU microchip ESP 8266, la cual está programada a través del IDE de Arduino. El propósito principal e innovador de dicha aplicación es usar MQTT y Firebase como intermediario de comunicación para dispositivos IoT, el cual ofrece muchos servicios en la nube desde la autenticación hasta el alojamiento de una aplicación web. La investigación se centrará en la base de datos en tiempo real, el cual es un sistema de base de datos que utiliza el procesamiento en tiempo real para manejar cargas de trabajo cuyo estado cambia constantemente. Los clientes que se conectan a la base de datos mantienen una conexión bidireccional abierta a través de los websockets, por lo tanto, si un cliente empuja datos a la base, esta se activara informando al resto de clientes conectados los nuevos datos guardados. Esto es muy parecido al corredor MQTT y cómo reacciona cuando recibe un mensaje de un editor y lo envía a todos los suscriptores, con la diferencia de la adición de la parte persistente

de los datos. Por lo que se puede ver, no necesita enrutar los mensajes usando otros protocolos, sino que Firebase se encarga de eso, además de realizar su función normal de base de datos (Fiware Tutorials, 2020).

2.2. Fundamentos teóricos

2.2.1. Protocolos de Comunicación IOT existentes para la industria 4.0.

El Internet de las Cosas es una realidad que se está viviendo, pero su efecto se intensificará en los próximos años. Sus posibilidades son infinitas, tanto como alcance la imaginación. Según el portal de estadísticas (Statista, 2016), se prevé que en 2025 habrá más de 75.000 millones de dispositivos conectados en todo el mundo. Más de 25.000 millones en 2019, casi 31.000 millones en 2020 y más de 50.000 millones de dispositivos en 2023.

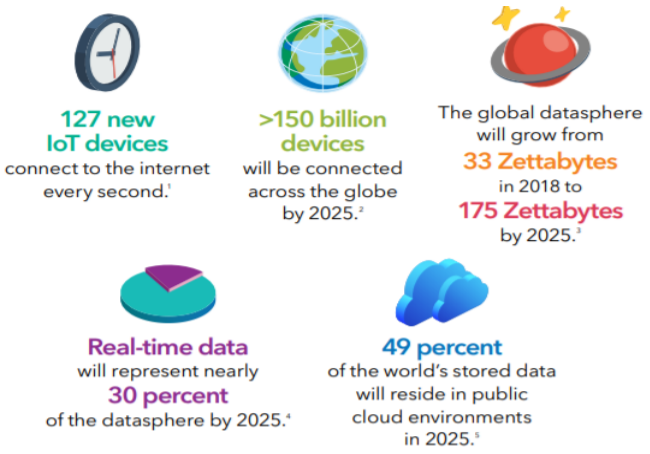

Figura 1: IoT Hoy y mañana. Fuente: (SaS IoT Solutions, 2019).

En la figura 1, se detalla algunos valores estadísticos referentes a como se proyecta el IoT para el mañana, como se observa cada segundo 127 nuevos dispositivos IoT se conectan al Internet, además el 49% de los datos almacenados a nivel mundial, residirían en entornos cloud públicos para el 2025 y los datos en tiempo real representaran casi el 30% de la esfera de datos.

Con la aparición de los dispositivos IoT, surge el concepto de industria 4.0. aquella que (Sáez, 2019) la define como la digitalización completa a través de la integración de

tecnologías de procesamiento de datos, software inteligente y sensores, desde los proveedores hasta los clientes. Para este análisis se considerará los protocolos de la capa de aplicación que son aquellos en los que los dispositivos pueden dialogar entre sí independientemente de cómo sean las infraestructuras sobre las que se apoyen siempre y cuando usen el mismo lenguaje en la sesión de comunicación que establezcan. Las ventajas derivadas de la aplicación de algunos de ellos son tan grandes que han pasado de ser prácticamente desconocidos, a ser junto al HTTP los más usados por los desarrolladores de todo el mundo.

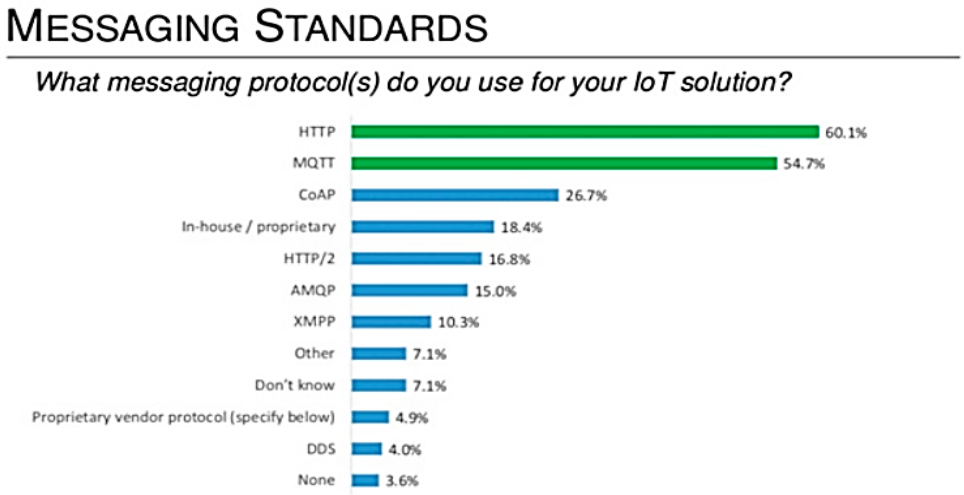

Figura 2: Protocolos de mensajería IoT. Fuente: (Warex, 2019).

En la figura 2, se puede evidenciar algunos de los estándares de mensajería de mayor demanda para proyectos o soluciones IoT, como se observa HTTP y MQTT resultan ser los protocolos preferidos en el mercado industrial.

(Rashmin, 2018), menciona ciertas características relevantes de los protocolos de comunicación de mayor demanda para el desarrollo IoT:

- AMQP (Advanced Message Queuing Protocol): Al igual que MQTT, este protocolo consiste en un estándar abierto para el intercambio de mensajes entre aplicaciones. Es un protocolo ampliamente utilizado en el mundo de las transacciones financieras pues proporciona interesantes características como son; su elevada seguridad, su fiabilidad, la inspección de paquetes para el enrutamiento (soporta comunicación

punto-a-punto y publicación-subscripción) e incorpora gestión de colas. Ofrece seguridad a través de la autenticación y cifrado mediante SASL o TLS.

- DDS (Servicio de Distribución de Datos): Permite el intercambio de datos de forma escalable, concebido para sistemas de tiempo real, es confiable y de alto rendimiento e interoperable con la metodología de publicación-suscripción, a diferencia de MQTT utiliza una arquitectura sin intermediarios, esto permite que la multidifusión para QoS sea de alta calidad en las aplicaciones. DDS puede llegar a realizar transacciones por debajo del milisegundo, lo que lo hace ideal para procesos en tiempo real.
- CoAP (Constrained Application Protocol): Es un protocolo cliente/servidor que difiere en muchos aspectos con MQTT, pero muy útil porque ha sido diseñado para poder traducirse a HTTP y así poder integrarse fácilmente en la Web. A pesar de su gran rapidez, extrema ligereza y su simplicidad, CoAP dispone de una capa de seguridad para encriptación de datos y funciones de alto nivel como multicast (uno publica y muchos escuchan). Trabaja sobre transporte UDP y también está orientado al uso en dispositivos conectados de bajos recursos como las redes de sensores inalámbricos.

Protocol	*Transport*	*Messaging*	*2G,3G,4G (1000's)*	*LowPower and Lossy (1000's)*	*Compute Resources*	*Security*	*Success Stories*	*Arch*
CoAP	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium - Optional	Utility field area ntwks	Tree
Continua HDP	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	None	Medical	Star
DDS	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Poor	100Ks/RAM Flash +++	High-Optional	Military	Bus
DPWS	TCP		Good	Fair	100Ks/RAM Flash ++	High-Optional	Web Servers	Client Server
HTTP/ REST	TCP	Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	Low-Optional	Smart Energy Phase 2	Client Server
MQTT	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium - Optional	IoT Msging	Tree
SNMP	UDP	Rqst/Response	Excellent	Fair	10Ks/RAM Flash	High-Optional	Network Monitoring	Client-Server
UPnP		Pub/Subscrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	None	Consumer	P2P Client Server
XMPP	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	High-Manditory	Rmt Mgmt White Gds	Client Server
ZeroMQ	UDP	Pub/Subscrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	High-Optional	CERN	P2P

Tabla 1: Diferencia entre protocolos de comunicación IoT. Fuente: (Warex, 2019).

En la tabla1, se diferencia las características principales de algunos de estos protocolos a conocer.

2.2.2. Firebase y MQTT a la vanguardia IoT

Según (Mayur Tanna, 2018), Firebase es una combinación de los muchos servicios de Google en la nube, incluyendo la mensajería instantánea, la autentificación de usuarios, base de datos en tiempo real, almacenamiento, hospedaje, entre otros. Firebase proporciona acceso a su base de datos en tiempo real que almacena y sincroniza los datos alojándolos en la nube (internet), estos datos son almacenados en JSON y se pueden agregar reglas para permitir solicitudes con token o solo desde una URL. Los datos de la base se sincronizan con todos los clientes en tiempo real, esto ayuda mucho cuando la app no tiene conexión a Internet. Cuando la conexión regresa el dispositivo lo reconoce y lo guarda en el servidor actual.

La diferencia y ventaja de Firebase sobre MQTT esta vez es la adición de la parte persistente de datos, que es la base de datos, es decir no se necesita enrutar los mensajes uno mismo usando otros protocolos, sino que Firebase Realtime Database se encargará de eso además de realizar su función normal de base de datos.

Entre las diferentes áreas de aplicación de los productos que ofrece Firebase, el proyecto se centra únicamente en las siguientes:

- Firebase Hosting: Es un servicio de hosting de contenido web orientado a programadores en una red de distribución de contenido global. A través de este servicio se pretende alojar el aplicativo web IoT del proyecto, haciendo uso de las librerías SDK que ofrece esta herramienta; lo que se intenta es que la aplicación no sea solamente local, sino que sea accesible y operativa desde afuera.
- Realtime Database: Base de datos NoSQL alojada en la nube que almacena datos y sincroniza en tiempo real con cada cliente conectado. Considerando a esta herramienta como la más importante a destacar del proyecto, cabe señalar que gracias a su procesamiento en tiempo real es posible manejar cargas de datos que cambian constantemente. Esto difiere de las bases de datos tradicionales que contienen datos persistentes, en su mayoría no afectados por el tiempo, es decir que

los clientes estarán conectados a la base de datos y mantendrán una conexión bidireccional abierta a través de websockets.

- Performance Monitoring: Servicio que obtiene estadísticas sobre el rendimiento de la aplicación en dispositivos iOS, Android y recientemente en Web con su versión beta. Aplicando este servicio al proyecto, se intenta medir tiempos de inicio de la app, solicitudes de red HTTP/S, tiempos de carga, recursos de la app, entre otros; recopilando datos de rendimiento para su posterior revisión, análisis y conclusiones.

Entre las ventajas de Firebase (Martínez, 2019) señala:

- Ofrece seguridad al usuario: El sistema proporciona datos de encriptación SSL.
- Con Firebase no se tiene que solicitar apoyo para la construcción de servicios web, API Rest, configuración del servidor, entre otros.
- Hosting sin necesidad de FTP ni servidores.
- Gestión de ficheros en la nube sin necesidad de desplegar ni montar en un servidor.

MQTT es un protocolo de mensajes publish-suscribe, muy simple y ligero, diseñado para sistemas con recursos muy limitados, bajo ancho de banda, alta latencia o redes poco fiables. Sus principios de diseño son minimizar el ancho de banda y los requerimientos de recursos mientras garantiza fiabilidad y calidad en la entrega del mensaje. Estas características lo convierten en un protocolo ideal para dispositivos móviles o bien dispositivos conectados M2M o IoT. La arquitectura de MQTT sigue una topología de estrella, donde existe un bróker o nodo principal con hasta 10.000 clientes conectados. El bróker se encarga de enrutar, filtrar y distribuir los mensajes a los clientes apropiados, y los clientes publican y se suscriben a estos mensajes. Esta comunicación puede ser cifrada. La comunicación se basa en unos temas (topics), de manera que el cliente publica el mensaje y los nodos que desean recibirlos han de suscribirse a él, de manera que esta comunicación puede ser uno a uno o uno a varios. (Mínguez, 2019)

Entre sus ventajas cabe resaltar:

- Facilita el cifrado de mensajes mediante TLS y la autenticación de clientes mediante protocolos de autenticación modernos, como OAuth.

- Sencillez y ligereza para no consumir demasiados recursos (aunque con seguridad TLS/SSL sube).
- Requiere un ancho de banda mínimo, lo cual es importante en redes inalámbricas, o conexiones con posibles problemas de calidad.
- Eficiencia energética para dispositivos que dependen de batería o funcionan 24/7, no necesita de un gran ancho de banda (ideal para conexiones lentas, como algunas inalámbricas).

La seguridad siempre debe ser un factor importante a considerar en cualquier sistema de comunicación. El protocolo MQTT dispone de distintas medidas de seguridad que podemos adoptar para proteger las comunicaciones. Esto incluye transporte SSL/TLS y autentificación por usuario y contraseña o mediante certificado. Sin embargo, hay que tener en cuenta que muchos de los dispositivos IoT disponen de escasa capacidad, por lo que el SLL/TLS puede suponer una carga de proceso importante. En muchos casos, la autentificación consiste en una contraseña y usuario que son enviados como texto plano. Por último, también es posible configurar el bróker para aceptar conexiones anónimas. Todo esto debe ser tenido en cuenta a la hora de configurar un sistema MQTT, y entender los riesgos de cada uno de ellos, así como su impacto en la eficiencia del sistema (Llamas, 2019).

Existe una gran variedad de placas electrónicas en el mercado, una de ellas es Arduino y otras que son compatibles con esta marca. Para esta investigación se usa la famosa NodeMCU, una placa compatible con el IDE de Arduino, basada en el chipset ESP8266, producido por el fabricante chino de Shanghai, Espressif Systems, revisar su datasheet en el anexo 2. Esta placa es atractiva para los desarrolladores, ya que se pueden comprar por unidades individuales y ser programadas de la misma forma que un Arduino. El NodeMCU se puede usar para aplicaciones IoT ya que tiene acceso a Internet de manera muy fácil, solo basta con una red Wifi y unas dependencias para conseguirlo. (Chimarro Amaguaña, 2020)

Para conectar correctamente la tarjeta Wifi ESP8266 mediante Arduino y Firebase se requiere lo siguiente:

- Descargar la librería Firebase Arduino extended.

- Descargar la librería Arduino JSON versión 5.13.4.
- Tener cuenta de Google creada para acceder a Firebase, de lo contrario, no se podrá usar esta poderosa herramienta.
- Tener instalado el entorno de desarrollo de Arduino, disponible en la página oficial.
- Tener instalada y configurada la librería ESP8266.

2.3. Propuesta

En el presente trabajo de investigación, se propone realizar una comparativa entre el estándar MQTT y la plataforma Google Firebase a través de dos aplicativos IoT, prototipos básicos que permiten el almacenamiento de datos y la comunicación en tiempo real. El propósito de dicha investigación consiste en realizar un análisis y evaluación de rendimiento que determine la eficiencia entre ambas plataformas señaladas. Para esta investigación se tomará en cuenta:

- Tiempos de respuesta y carga.
- Pruebas de rendimiento.
- Tiempos de latencia.

2.4. Caso de estudio

El proyecto de investigación consiste en realizar dos sistemas: uno que controle el encendido y apagado de luminaria y otro que capte valores del medio físico, cada uno independiente del otro. Los valores adquiridos de cada sistema son enviados a un servicio de base de datos para almacenarlos y posteriormente mostrarlos dentro de la aplicación móvil. Para este trabajo se implementará dos soluciones distintas, por un lado, se configurará el bróker MQTT que actuará como Hub y redirigirá todos los mensajes entrantes publicados desde el dispositivo a todos los clientes suscritos, en este caso la aplicación móvil. Y, por otro lado, a través de Firebase combinar para que la aplicación haga ambas cosas: mostrar los datos que vienen del bróker en tiempo real y obtener los datos de la base.

2.5. Implementación de las aplicaciones IoT

2.5.1. Aplicación IoT - Registro de temperatura

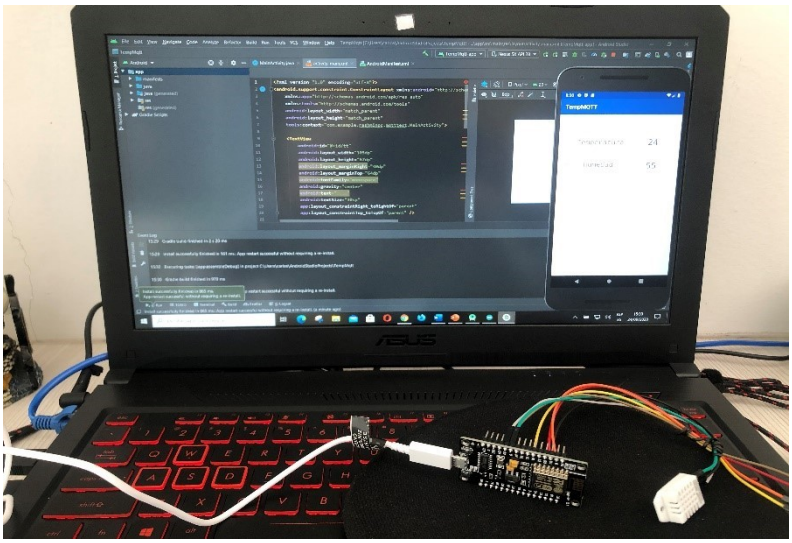

Figura 3: Aplicación IoT - Registro de temperatura. Fuente: Del autor.

En la figura 3, se observa como en esta aplicación, el dispositivo envía o publica los datos al bróker MQTT, y éste posteriormente redirige el mensaje a todos los suscriptores conectados, en este caso el aplicativo móvil. Pero con la diferencia de que existe un suscriptor conectado que representa un motor API, el cual acepta esos datos y los envía al servicio web de la base de datos para ser almacenados.

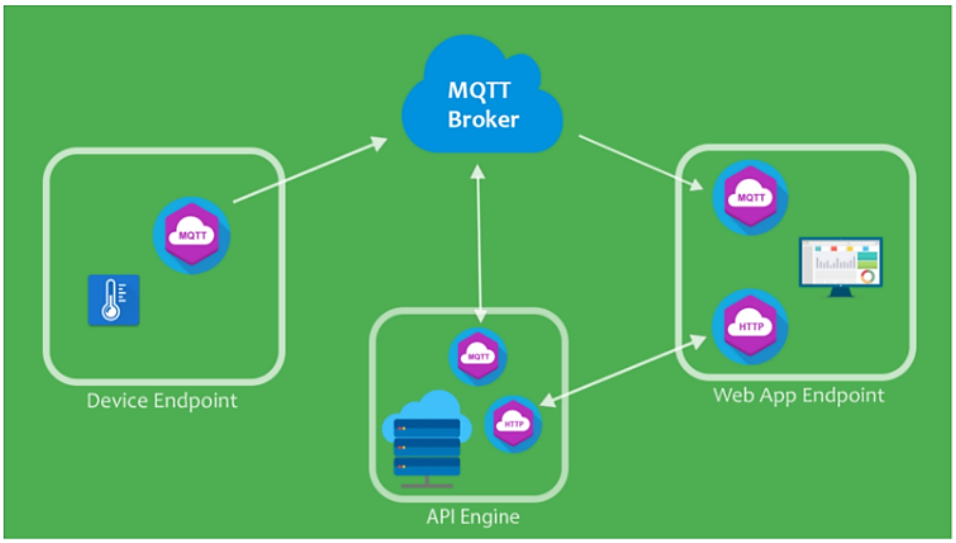

Figura 4: Modelo de comunicación MQTT con API engine. Fuente: (Gharssellaoui, 2019).

Como se habrá notado en esta solución, el cliente HTTP se desacopla del dispositivo y se implementa como un servicio backend. De esta manera se hace el programa del dispositivo mucho más ligero. Esto es algo importante a tener en cuenta cuando se desarrolla en dispositivos IoT restringidos donde los recursos como la CPU y la memoria son limitados. Aun así, esta solución requirió un trabajo adicional en el desarrollo del servicio de fondo que actuó como una capa de persistencia.

Con respecto a la configuración del aplicativo se hace énfasis señalando las dependencias, librerías y servicios utilizados durante su desarrollo e implementación:

- Como bróker, se utilizó el servicio Cloud-MQTT por su rápida configuración y administración en soluciones IoT, cuyos planes son únicamente de pago, pero cuenta con un plan asequible dedicado a investigadores y estudiantes.

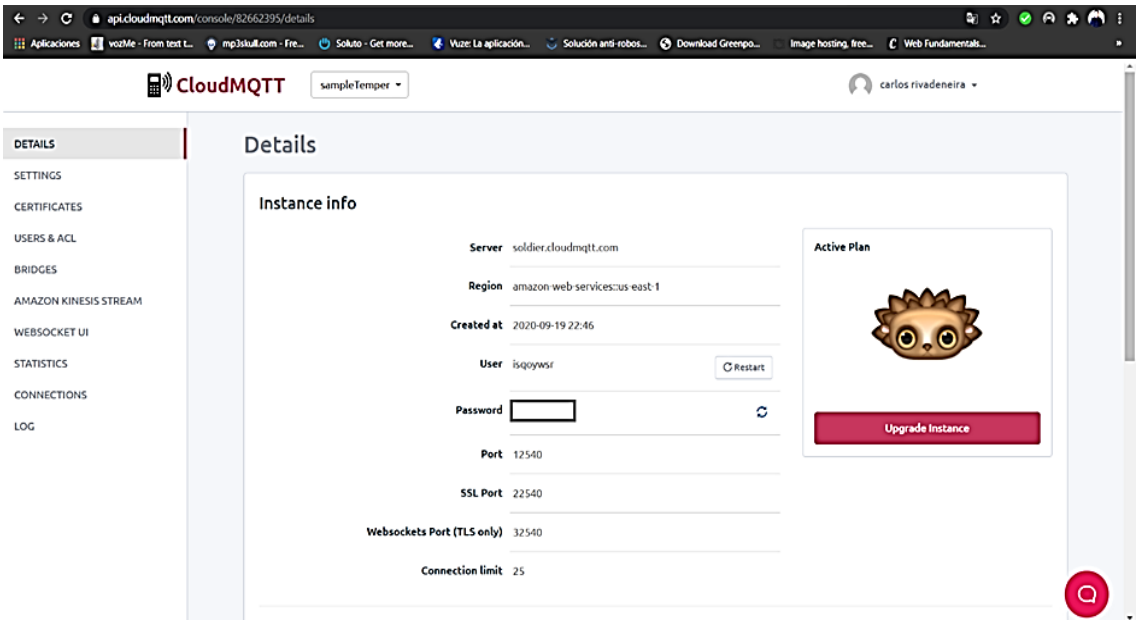

Figura 5: Detalles del servicio CloudMQTT. Fuente: Del autor.

- Se utilizó Pub-Sub-client como biblioteca MQTT para la comunicación entre el IDE de Arduino y la placa Node-MCU. En la figura 6, se señala y se define las bibliotecas que se debe importar, así como las constantes que son necesarias para las configuraciones MQTT y Wifi (credenciales).

Figura 6: Importación y definición de bibliotecas y constantes en Arduino. Fuente: Del autor.

- Para la parte de Android Studio, se utilizó Paho-MQTT como API. En la figura 7 y 8, se señala como se incluye este complemento con sus dependencias en el archivo app.gradle, así como en el mainActivity.

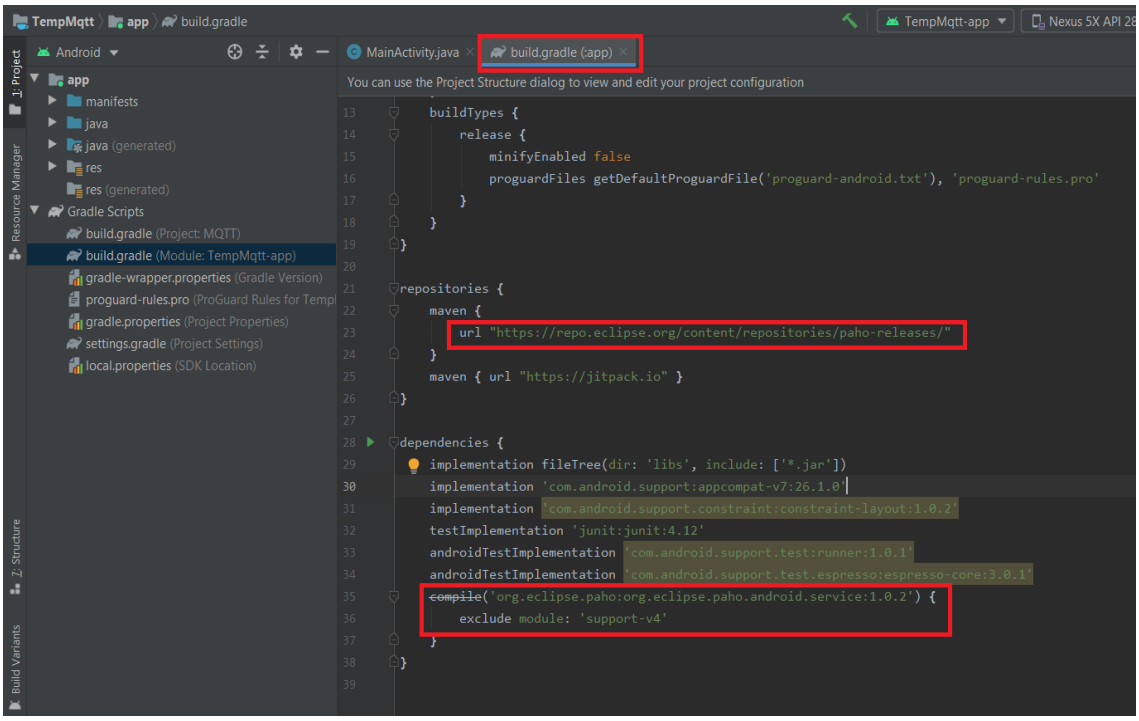

Figura 7: Inserción de repositorios y dependencias del API Paho-MQTT. Fuente: Del autor.

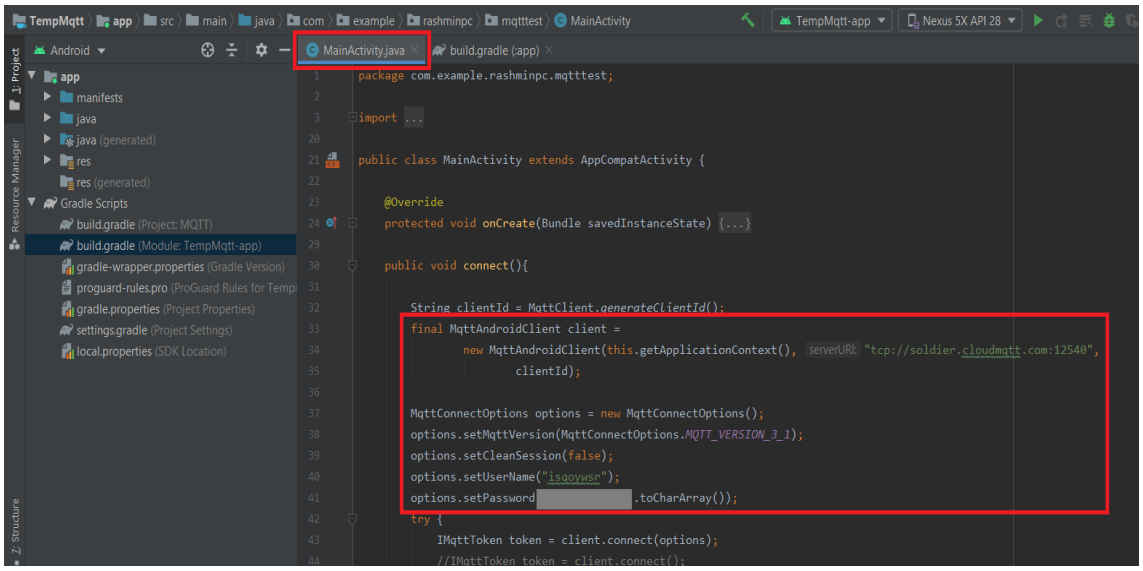

Figura 8: Definición de las variables de conexión al bróker MQTT. Fuente: Del autor.

2.5.2. Aplicación IoT - Control de encendido de luminaria

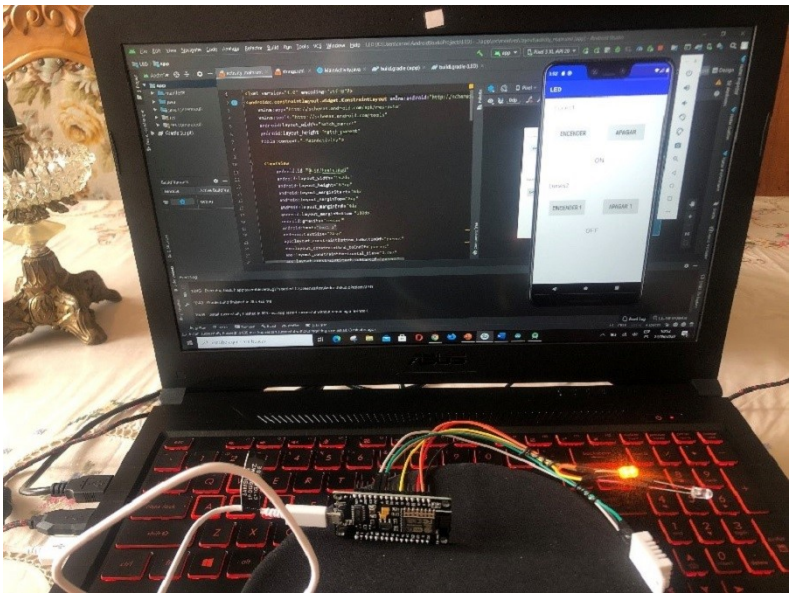

Figura 9: Aplicación IoT - Control de encendido de luminaria. Fuente: Del autor.

En la figura 9, se observa cómo se implementó la plataforma de Google Firebase, a través de la cual se conectó el dispositivo a la base de datos en tiempo real y permitió el envío de información periódicamente. Por otro lado, al sistema se le agrego una app móvil que se conecta al mismo servicio que el dispositivo y recibe nuevos datos cada vez que haya un cambio en la base, lo que se conoce como persistencia de datos.

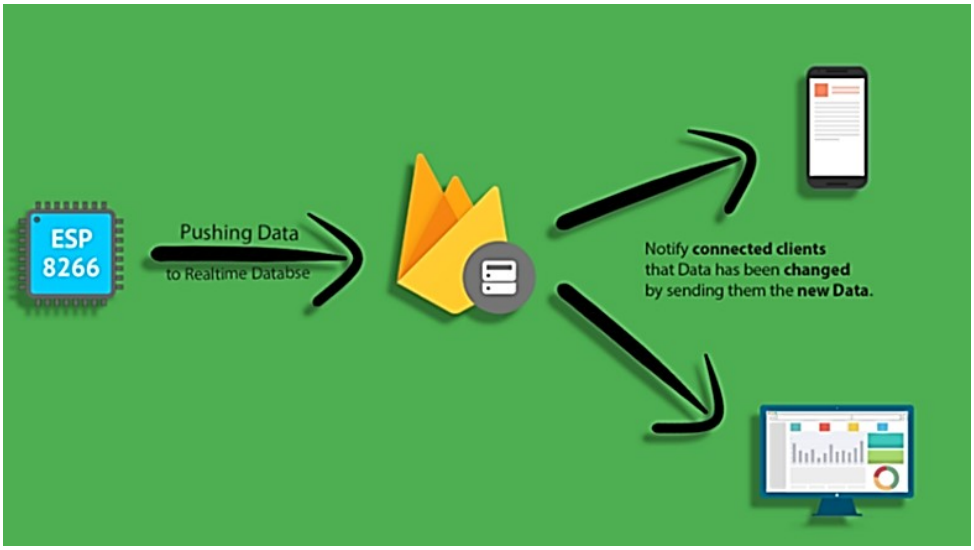

Figura 10: Modelo de comunicación Firebase. Fuente: (Gharssellaoui, 2019)

Como se puede apreciar en la figura anterior, tan pronto como se actualiza un campo en la base de datos, las aplicaciones que están conectadas a esta son notificadas de esos cambios utilizando las API que proporciona dicha base. En este caso, el aplicativo móvil se encuentra conectado a la base de datos de Firebase a través de servicios y librerías que ofrece la API, el mismo que al efectuar un cambio o recibir un dato, es notificado y almacenado instantáneamente en la base.

Con respecto a la configuración e implementación del aplicativo se consideró las siguientes herramientas a utilizar:

- Instalación de la BDD Firebase: se ingresó a la consola de Firebase una vez registrado en Google y se creó un nuevo proyecto, posteriormente se procedió a crear una base de datos en tiempo real y a declarar sus variables, finalmente se definieron las reglas de seguridad, este paso es muy importante ya que si no se modifican ningún dispositivo podrá acceder a la base de datos.

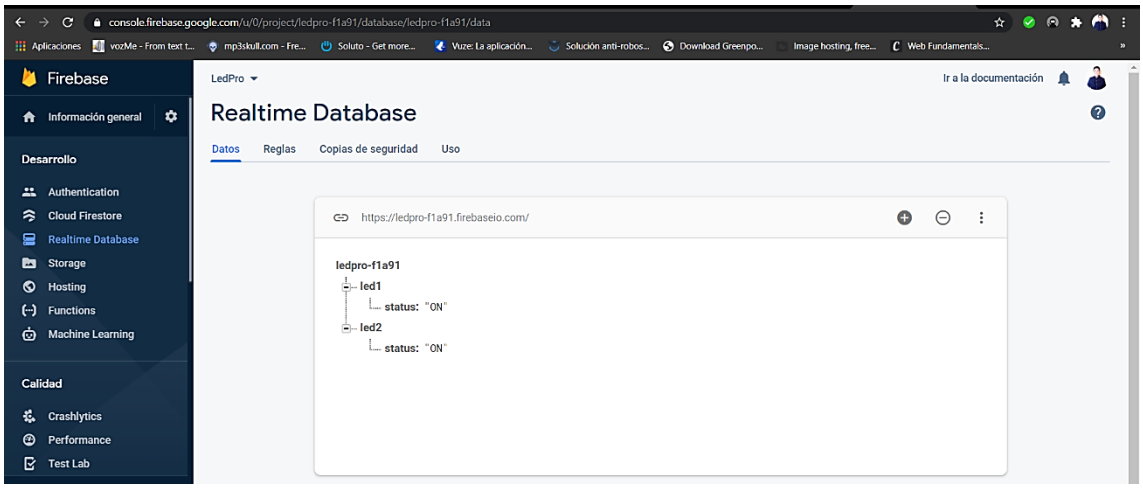

Figura 11: Configuración de la BDD en Firebase. Fuente: Del autor.

- Creación de una aplicación y conexión a la BDD: para este proyecto se creó un aplicativo únicamente en Android, considerando que Firebase es compatible con cualquier entorno de desarrollo. En la figura 12 y 13 se observa cómo se conecta Android Studio al servicio de la base de datos de Firebase a través de un complemento propio de la herramienta, así como también se crea dos métodos o instancias que sirven para tomar los datos de la base, mostrarlos y empujarlos a la misma. Estos segmentos de código crean una conexión con la raíz de la base de datos.

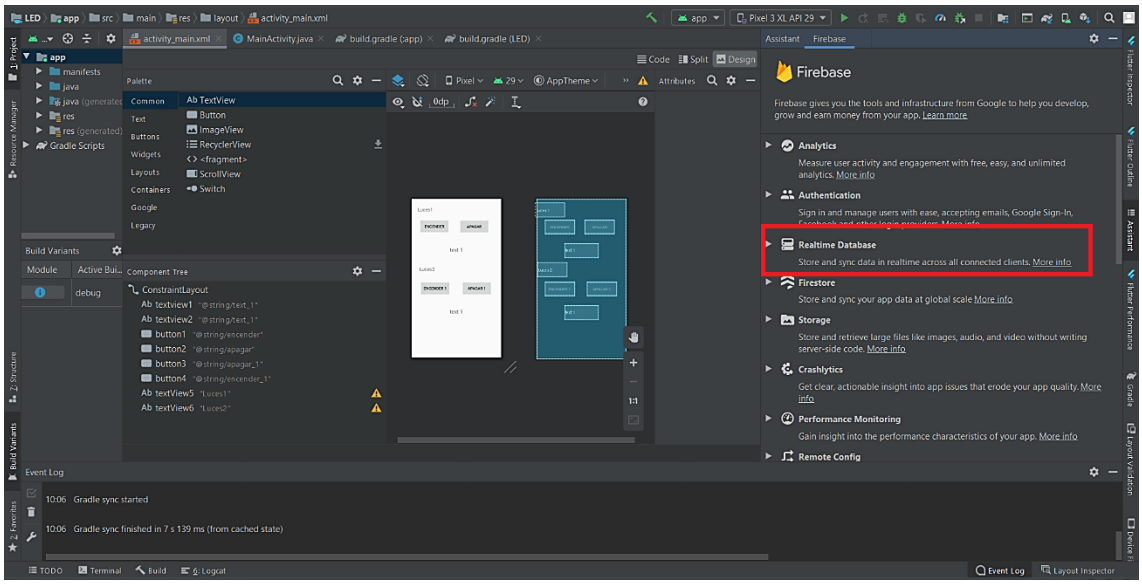

Figura 12: Conexión de Android Studio a la BDD Firebase. Fuente: Del autor.

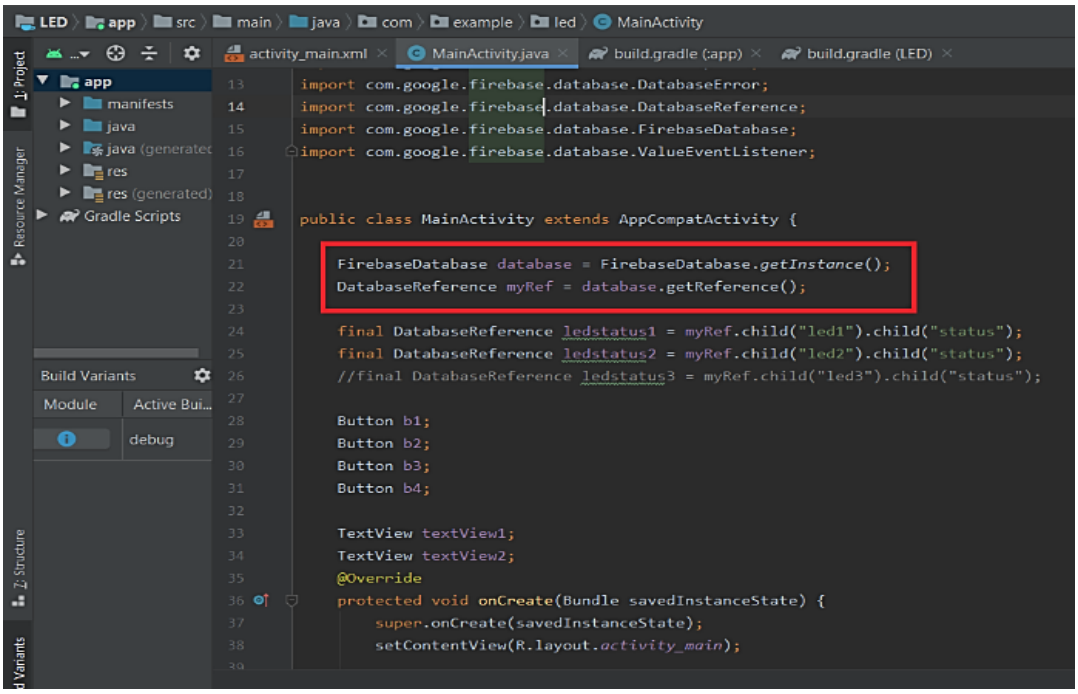

Figura 13: Conexión con la raíz de la BDD. Fuente: Del autor.

- Conexión de la placa con la BDD: la conexión del chipset NodeMCU con la base de datos se lo realizo a través de la biblioteca de Arduino Firebase que no es más que una biblioteca muy fácil de usar y con una buena documentación. En la figura 14 y 15 se señala la declaración en Arduino de la clave de autenticación y la URL de la API de la BDD, esta información se obtiene de la página de configuración del proyecto Firebase. Finalmente, dentro de la función de configuración del proyecto en Arduino se agrega el llamado a la base de datos.

Figura 14: Declaración de credenciales Firebase en Arduino. Fuente: Del autor.

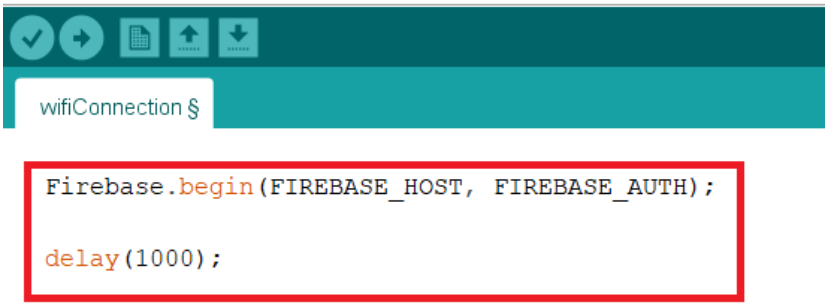

Figura 15: Petición a la BDD. Fuente: Del autor.

2.6. Test y Resultados

En esta última fase, con el propósito de obtener los resultados finales se integran los dispositivos con sus respectivos protocolos de comunicación y los programas que lo compilan, para llevar a cabo el funcionamiento general del sistema, y realizar las pruebas de rendimiento, latencia e integridad de la información.

2.6.1. Pruebas de rendimiento y latencia

Una vez que se ha culminado con el desarrollo de los aplicativos IoT y la implementación de MQTT y Firebase respectivamente, se analizó en cada uno de ellos, el tiempo promedio de retardo que existe entre los atributos y el tiempo promedio de retardo de cada atributo cuando se genera un evento. Para extraer los tiempos se usó la comunicación serial entre la tarjeta ESP8266 y la PC, a una velocidad de 9600 Baudios, visualizando los datos en la interfaz serial del IDE de Arduino, como se observa en las siguientes figuras:

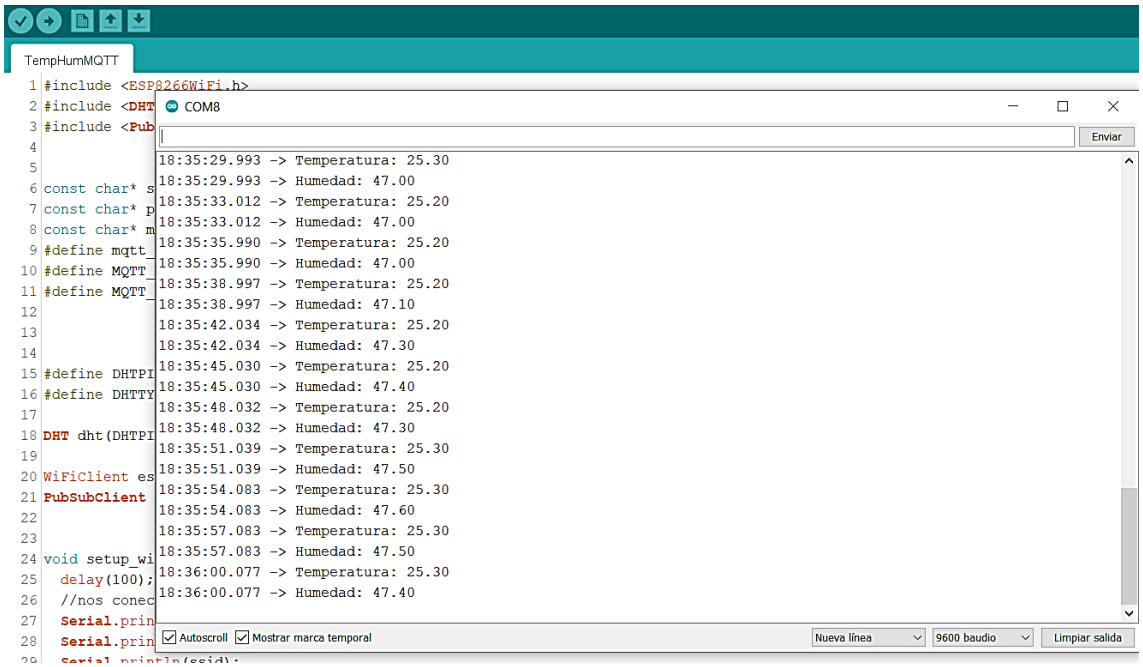

Figura 16: Salida de datos por el serial de Arduino - Aplicación MQTT. Fuente: Del autor.

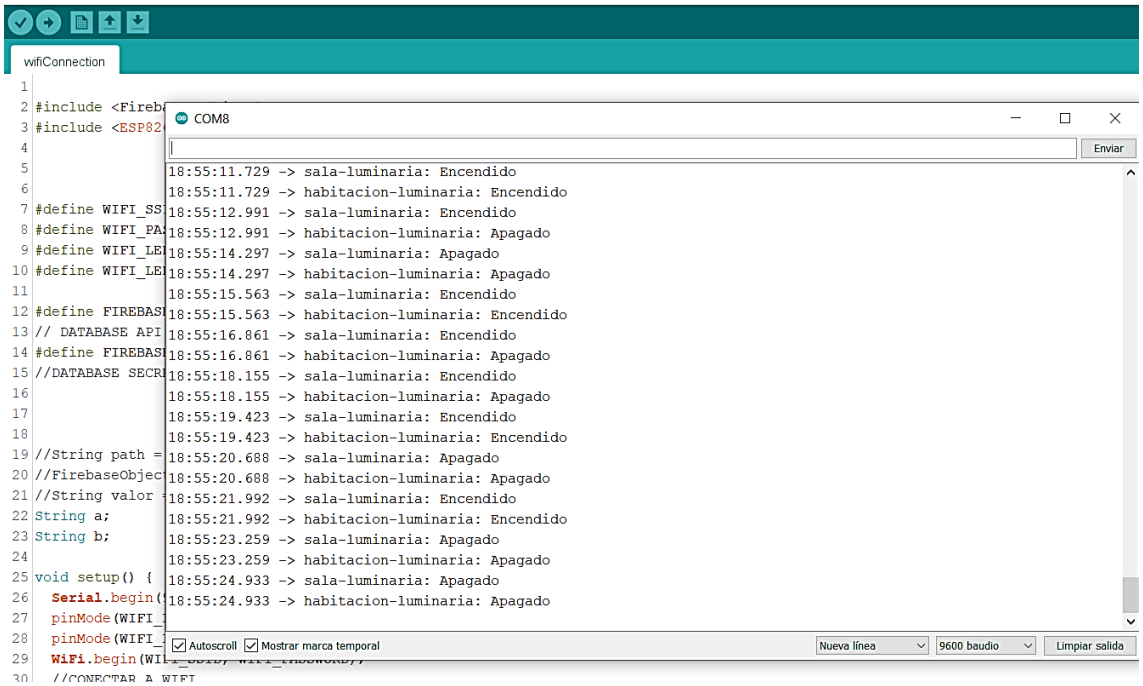

Figura 17: Salida de datos por el serial de Arduino - Aplicación Firebase. Fuente: Del autor.

Protocolo / Plataforma	Promedio total de retardo entre cada atributo (ms)
MQTT	727.25
FIREBASE REALTIME DATABASE	569.6

Tabla 2: Promedio total de retardo por atributo. Fuente: Del autor.

Los valores señalados, representan el resultado de un análisis de tiempos de latencia generado por cada atributo. Ver anexo 3 y 4.

Como se puede observar en la tabla 2, los resultados obtenidos tras la implementación del protocolo MQTT como la plataforma Firebase en los aplicativos IoT referidos en el apartado 2.5, detallan una notable diferencia de tiempo entre ambos: Firebase con 157.65 milisegundos menos en retardo, lo cual representa un tiempo de latencia aceptable y moderado al comparar ambas herramientas de comunicación. Ya que se debe tomar en cuenta además que se trabajó con una red Wifi de 8.3 Mbps de ancho de banda y 48.5 milisegundos de tiempo promedio entre carga y descarga, ver figura 18.

Figura 18: Velocidad de red inalámbrica empleada en la placa NodeMCU. Fuente: Fast. Obtenido de: https://fast.com/es/

Protocolo / Plataforma	Promedio total de retardo por evento (ms)
MQTT	2527.5
FIREBASE REALTIME DATABASE	2102.5

Tabla 3: Promedio total de retardo por evento. Fuente: Del autor.

Los valores señalados, representan el resultado de un análisis de tiempos de latencia generado por cada evento. Ver anexo 5.

Como se puede observar en la tabla 3, el tiempo promedio de retardo por evento que se produjo en cada atributo, tanto para la comunicación MQTT como la de Firebase fue de 2.53 y 2.10 segundos respectivamente; eso indica que el tiempo de latencia es considerable por cada atributo pero puede representar una desventaja con respecto a la precisión, ya que debido a la velocidad con la que navega el dispositivo móvil este se puede conectar a una red 3G, 4G o Wifi dependiendo de la zona. Para las aplicaciones móviles se usó una red móvil 3G con un ancho de banda de 6.5Mbps y un tiempo promedio de carga y descarga de 38 milisegundos con el fin de mantener la comunicación lo más óptima posible, ver figura 19.

Figura 19: Velocidad de la red 3G Claro que uso en el dispositivo móvil. Fuente: SpeedTest. Obtenido de: https://www.speedtest.net/apps/mobile

2.6.2. Uso de la BDD Firebase Realtime y el Bróker MQTT

En este apartado se muestra los resultados sobre el uso de la base de datos en tiempo real de Firebase así como del bróker, en la cual se detallan el número de conexiones simultáneas, almacenamiento de la base de datos, carga y descarga de datos.

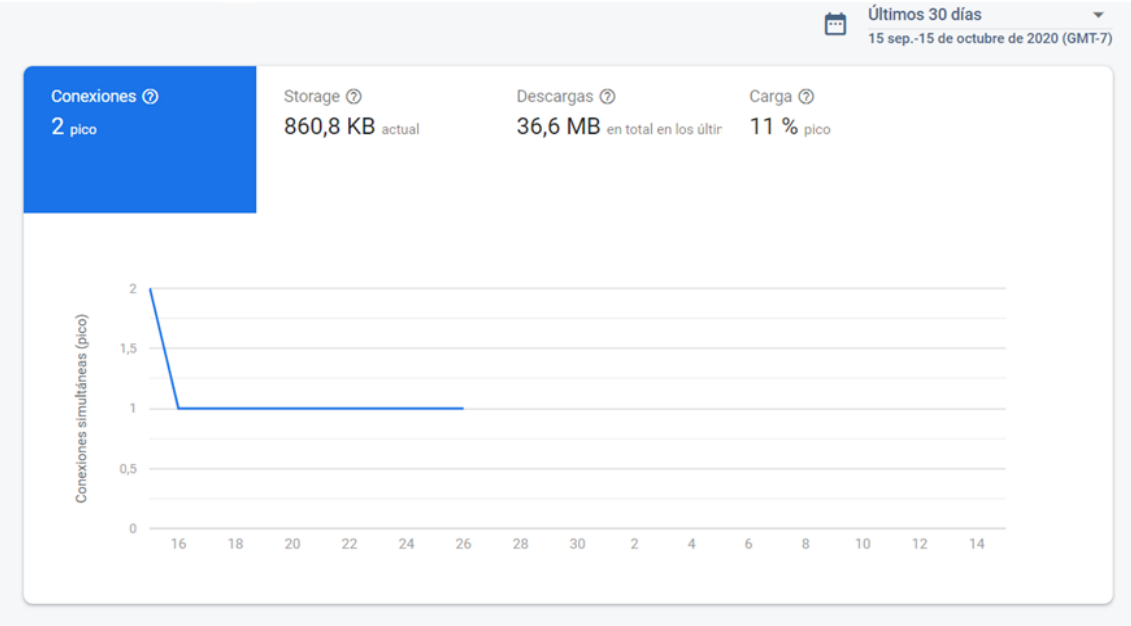

Figura 20: Conexiones simultaneas. Fuente: Firebase Realtime Database.

En la figura 20, se define la cantidad de conexiones simultáneas en tiempo real a la base de datos generado durante los últimos 30 días (15 septiembre - 15 octubre).

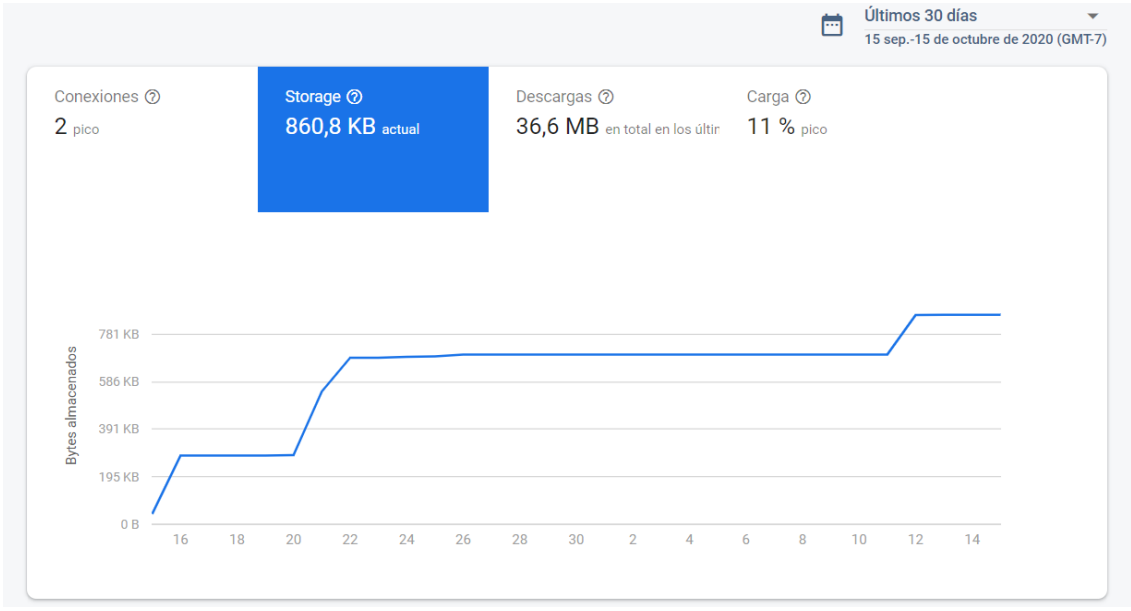

Figura 21: Almacenamiento en la base de datos. Fuente: Firebase Realtime Database.

En la figura 21, se especifica la cantidad de datos que se almacenaron en la base de datos durante 30 días, sin incluir el hosting de Firebase ni los datos almacenados de otras funciones de Firebase.

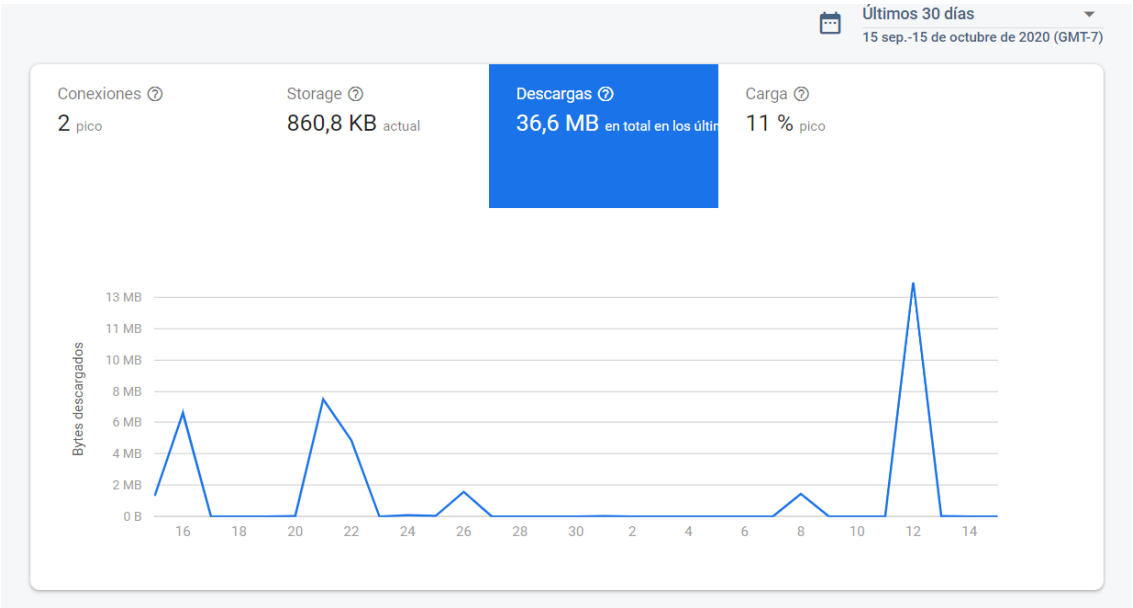

Figura 22: Descarga de datos de la BDD. Fuente: Firebase Realtime Database.

En la figura 22, se señala la cantidad de datos descargados de la BDD, incluido la sobrecarga del cifrado SSL y el protocolo de conexión durante intervalos de 1 minuto por los 30 días.

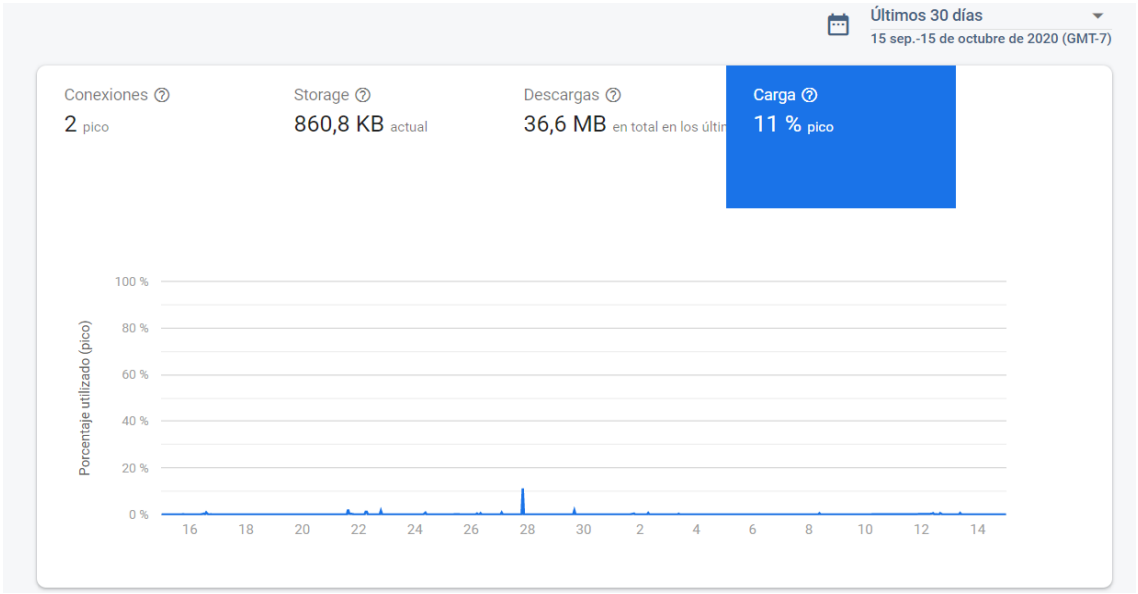

Figura 23: Carga de datos desde la BDD. Fuente: Firebase Realtime Database.

En la figura 23, se marca el porcentaje de la base de datos que está ocupada procesando peticiones, ya sea conexiones en tiempo real o solicitudes de la API de RESTful, durante intervalos de 1 minuto por 30 días. Es necesario resaltar que ocurran problemas de rendimiento a medida que se ocupe el 100%, ya que usar variables innecesarias o irrelevantes puede incrementar tiempos de latencia lo cual seria una desventaja en este tipo de sistemas que funcionan en tiempo real.

Con respecto al bróker, se realizó una prueba de stress a través de la herramienta MQTT stresser, software de licencia libre, la cual se configuro para probar con 20 clientes y 700 mensajes, produciendo una cantidad de 14000 mensajes por cliente. Estos valores corresponden a los datos enviados y recibidos durante un periodo de 30 minutos.

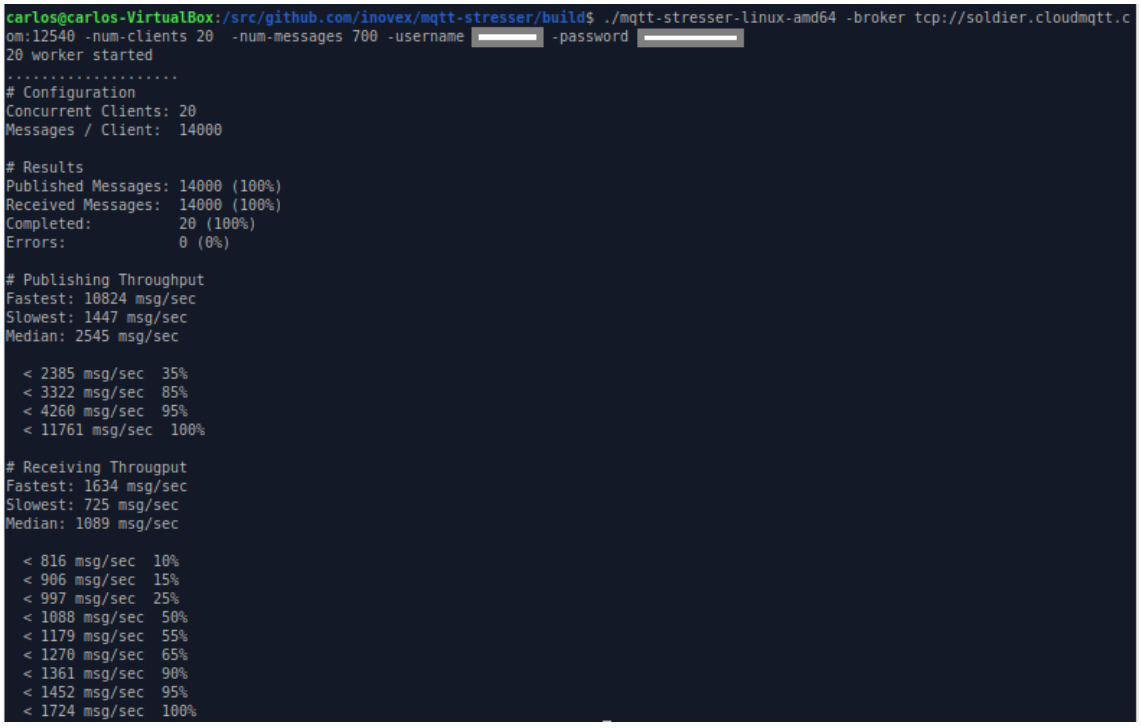

Figura 24: Prueba de stress al bróker. Fuente: MQTT stresser.

En la figura 24, se considera el rendimiento de publicación y recepción como resultados significativos al momento de evaluar la escalabilidad del servidor, como se puede observar se alcanzó los 2545 mensajes/segundo en lo que respecta al rendimiento de publicación y 1089 mensajes/segundo en lo que corresponde al rendimiento de recepción. Estos valores representan la velocidad mediana con la que se enviaron y recibieron los mensajes del bróker al cliente y viceversa.

Finalmente, a través del reporte general de resultados de Google Cloud Platform se pudo obtener el tráfico transmitido, porcentaje de errores de ingreso a los controles y los tiempos de latencia generado por las APIs de Firebase.

Nombre	↓ Solicitudes	Errores (%)	Latencia, mediana (ms)	Latencia del 95 % (ms)
Cloud Functions API	6	0	734	1.017
Firebase Management API	5	0	196	1.835
Firebase Remote Config API	2	0	196	255
Firebase Extensions API	1	0	196	255
Firebase Installations API	1	0		

Figura 25: Reporte de resultados y servicios. Fuente: Google Cloud Platform.

Como se observa en la figura 25, Google Cloud Platform entregó tiempos de latencia aproximados a los que se midió en la interfaz serial de Arduino, con respecto al tráfico de datos obtenidos del Cloud Functions API con Firebase se obtuvo tiempos de latencia entre cada atributo correspondientes a 734 milisegundos y respecto al tiempo de latencia por evento de cada atributo se obtuvo 1017 milisegundos.

Una vez que ambas plataformas de comunicación fueron sometidas a pruebas de rendimiento, latencia y uso de la base de datos, se obtuvieron resultados finales de los cuales se puede deducir que ambas garantizan fiabilidad y calidad en la transmisión, administración y uso de la data. Por un lado, Firebase junto con los varios servicios que ofrece en la nube, es recomendable usarlo por su fácil configuración cuando se tratan de proyectos IoT pilotos o prototipos que se desean poner en marcha rápidamente y no requieran de una base de datos extensa y compleja ya que esto conduciría al uso de un ancho de banda relativamente más alto y a la contratación de un plan por consumo, opción que en este caso es gratuita y ofrece todos los servicios sin restricción. Por otro lado, MQTT es recomendable usarlo ya en proyectos de producción debido a que es un protocolo ligero diseñado para sistemas con recursos limitados, cuyo consumo de ancho de banda, CPU y memoria en los dispositivos IoT son mínimos.

CONCLUSIONES

En base a los objetivos específicos planteados, se determinan las siguientes conclusiones:

- La contextualización de fundamentos teóricos permitió conocer y considerar las características, diferencias, ventajas y desventajas entre los diferentes protocolos de comunicación IoT existentes, conocer los requerimientos necesarios para su uso permitió determinar que protocolos se utilizarían como referencia para la investigación y pruebas.
- La arquitectura Cloud Computing aportó en la reducción de recursos y costos ya que la infraestructura del prototipo IoT se la implementó usando los APIs y Services de Google Firebase, dicha plataforma solo requiere tener una cuenta de Gmail y actúa como bróker lo que permitió una interacción directa por medio del aplicativo móvil, permitiendo conectarse de forma remota a tarjetas de control que se conectan a la red de Internet.
- La aplicación móvil que se generó, tiene la capacidad de conectarse a la red de Internet lo cual permite enlazar a los sensores de temperatura y controles leds desde casi cualquier punto del planeta en tiempo real, la simplicidad de la aplicación permite una gran usabilidad y control de todas las variables del sistema, así como la carga y descarga de datos, seguridad de la información, obtención de estadísticas sobre el uso y compatibilidad de hardware.
- Se valoró el rendimiento del sistema integrado de comunicación IoT a través de los resultados estadísticos que ofrecen las herramientas Firebase Analytics, Firebase Performance y CloudMQTT, donde se midió el tráfico tele-transmitido, tiempos de latencia, errores producidos, número de eventos, y se verificó el cumplimiento de los criterios de calidad, servicio, escalabilidad, confiabilidad y versatilidad del sistema.

RECOMENDACIONES

En base a las conclusiones determinadas, se sugiere las siguientes recomendaciones:

- Se recomienda realizar un trabajo de investigación donde se combinen estas dos plataformas de comunicación estudiadas y se generen nuevas pruebas de evaluación de rendimiento, para que junto con la investigación realizada se pueda determinar si existen diferencias y mejorías.
- Implementar este modelo de comunicación en proyectos IoT de mayor escala y a un nivel de producción para poder evaluar su desempeño.
- MQTT es un protocolo sumamente versátil, que presenta una muy buena opción a la hora de realizar comunicación entre dispositivos enviando mensajes sumamente pequeños, pero es necesario implementar herramientas de seguridad adicional, ya que cualquier usuario que conozca la IP del host puede suscribirse a un tópico y recibir mensajes.
- Experimentar este tipo de investigaciones sobre la red móvil 5G, de tal manera que se pueda mejorar la calidad de la comunicación, así como la implementación en aplicaciones de ámbito sanitario e industria.
- Al momento de crear un proyecto IoT se recomienda usar o establecer variables únicamente necesarias, ya que la información irrelevante puede incrementar tiempos de latencia y carga, provocando errores lo cual sería una desventaja en este tipo de sistemas que funcionan en tiempo real.

BIBLIOGRAFÍA

Alhomsi, Y., & Alsalemi, A. (2017). Real-Time Communication Network Using Firebase Cloud IoT Platform for ECMO Simulation.

Carpio Santos, L. K. (2018). *Sistema móvil para controlar la posición en tiempo real del ganado de la Finca Sartenejal del Cantón Baba.* Babahoyo.

Chimarro Amaguaña, J. D. (2020). *Sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles.*

Fiware Tutorials. (2020). *IOT AGENTS & ROBOTS.* Obtenido de IOT OVER MQTT: https://fiware-tutorials.readthedocs.io/en/latest/iot-over-mqtt/index.html

Gharssellaoui, B. (19 de 01 de 2019). *freeCodeCamp*. Obtenido de freeCodeCamp: https://www.freecodecamp.org/news/iot-prototyping-with-firebase-doing-more-with-less-2f5c746dac8b/

GoldenMace IT Solutions. (31 de 05 de 2019). *GoldenMace IT Solutions.* Obtenido de Node-MCU with MQTT and Firebase: https://goldenmace.com/blog/labs/node-mcuesp8266-with-mqtt-and-firebase/

Gomez Bastar, S. (2019). *Metodología de la Investigación.* España: Red Tercer Milenio.

Hernández Escobar, A. A., & Ramos Rodríguez, M. P. (2018). *Metodología de la Investigación Científica.* Mexico: 3Ciencias.

Ionos. (18 de 05 de 2020). *Ionos.* Obtenido de Ionos: https://www.ionos.es/digitalguide/paginas-web/derecho-digital/iot-internet-de-las-cosas/

Klaus , H. (2003). *Introducción a la Metodología de la Investigación Empírica.* España: Paidotribo.

Llamas, L. (17 de 04 de 2019). *LUIS LLAMAS Ingeniería, informática y diseño*. Obtenido de ¿QUÉ ES MQTT? SU IMPORTANCIA COMO PROTOCOLO IOT: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

Martin Niño, D. R. (2017). Sistema de alarma y monitoreo móvil para automóviles implementando el protocolo de transmisión MQTT y mensajería instantánea.

Martínez, D. (04 de Junio de 2019). *OpenWebinars.* Obtenido de Ventajas de usar Firebase: https://openwebinars.net/blog/ventajas-de-usar-firebase/

Mayur Tanna, H. (2018). *Serverless Web Applications with React and Firebase: Develop real-time applications for web and mobile platforms.* Birmingham: Packt Publishing.

Microsoft Azure. (2020). *Microsoft Azure*. Obtenido de Protocolos y tecnologías de IoT: https://azure.microsoft.com/es-es/overview/internet-of-things-iot/iot-technology-protocols/

Mínguez, C. (28 de 08 de 2019). *Interempresas.* Obtenido de Interempresas: https://www.interempresas.net/TIC/Articulos/252927-IoT-sacando-partido-al-mundo-hiperconectado.html

Nuñez, L., & Toasa, R. (2020). Performance evaluation of RTMP, RTSP and HLS protocols for IPVT in mobile networks. *IEEE Xplore*.

Rashmin, R. (22 de 07 de 2018). *Coinmonks.* Obtenido de Arduino to Android , Real Time Communication For IoT with Firebase: https://medium.com/coinmonks/arduino-to-android-real-time-communication-for-iot-with-firebase-60df579f962

Ruiz, M. (09 de 08 de 2017). *OpenWebinars.* Obtenido de https://openwebinars.net/blog/que-es-firebase-de-google/?cv=1

Sáez, I. P. (07 de 02 de 2019). *Incibecert.* Obtenido de IoT: protocolos de comunicación, ataques y recomendaciones: https://www.incibe-cert.es/blog/iot-protocolos-comunicacion-ataques-y-recomendaciones

SaS IoT Solutions. (2019). *SaS IoT Solutions*. Obtenido de SaS IoT Solutions: https://www.sas.com/content/dam/SAS/documents/infographics/2019/internet-of-things-109082.pdf

Sparkfun, E. (2020). *MAKER.IO.* Obtenido de Send and Receive Messages to your IoT Devices using MQTT: https://www.digikey.com/en/maker/projects/send-and-receive-messages-to-your-iot-devices-using-mqtt/39ed5690cc46473abe8904c8f960341f

Statista. (27 de 11 de 2016). *Statista.* Obtenido de Statista: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

Warex. (04 de 03 de 2019). *Warex.* Obtenido de Protocolos de IoT: https://warex.com.co/protocolos-de-iot/

Wu-Jeng , L., & Chiaming , Y. (2018). JustIoT Internet of Things based on the Firebase real-time database.

ANEXOS

ANEXO 1

Niveles de Confianza

Niveles de Confianza Comunes

Los niveles de confianza comúnmente usados son 90%, 95% y 99%

Nivel de Confianza	*Valor Crítico, z*
80%	1.28
90%	1.65
95%	1.96
98%	2.33
99%	2.58
99.8%	3.08
99.9%	3.27

ANEXO 2

Datasheet del NodeMCU v3 - ESP8266

INFO

NodeMCU es una tarjeta de desarrollo similar a Arduino, especialmente orientada al Internet de las cosas (IoT). Está basado en el SoC (System on Chip) **ESP8266,** un chip altamente integrado, diseñado para las necesidades de un mundo conectado. Integra un potente procesador con Arquitectura de 32 bits (más potente que el Arduino Due) y conectividad Wifi.

Para el desarrollo de aplicaciones se puede elegir entre los lenguajes Arduino y Lua. Al trabajar dentro del entorno Arduino podremos utilizar un lenguaje que ya conocemos y hacer uso de un IDE sencillo de utilizar, ademas de hacer uso de toda la información sobre proyectos y librerías disponibles en internet. La comunidad de usuarios de Arduino es muy activa y da soporte a plataformas como el ESP8266.

NodeMCU viene con un firmware pre-instalado el cual nos permite trabajar con el lenguaje interpretado LUA, enviandole comandos mediante el puerto serial (CP2102). Las tarjetas NodeMCU y Wemos D1 mini son las plataformas mas usadas en proyectos de Internet de las cosas (IoT). No compite con Arduino, pues cubren objetivos distintos, incluso es posible programar NodeMCU desde el IDE de Arduino.

La tarjeta NodeMCU está diseñada especialmente para trabajar en protoboard. Posee un regulador de voltaje en placa que le permite alimentarse directamente del puerto USB. Los pines de entradas/salidas trabajan a 3.3V. El chip CP2102 se encarga de la comunicación USB-Serial.

ESPECIFICACIONES TÉCNICAS

- Voltaje de Alimentación (USB): 5V DC
- Voltaje de Entradas/Salidas: 3.3V DC
- SoC: ESP8266 (Módulo ESP-12)
- CPU: Tensilica Xtensa LX3 (32 bit)
- Frecuencia de Reloj: 80MHz/160MHz
- Instruction RAM: 32KB
- Data RAM: 96KB
- Memoria Flash Externa: 4MB
- Pines Digitales GPIO: 17 (pueden configurarse como PWM a 3.3V)
- Pin Analógico ADC: 1 (0-1V)
- UART: 2
- Chip USB-Serial: CP2102
- Certificación FCC
- Antena en PCB
- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Stack de Protocolo TCP/IP integrado
- PLLs, reguladores, DCXO y manejo de poder integrados
- Potencia de salida de +19.5dBm en modo 802.11b
- Corriente de fuga menor a 10uA
- STBC, 1×1 MIMO, 2×1 MIMO

ANEXO 3

Tiempos de retardo entre atributos. Aplicativo IoT - MQTT

Iteración de datos por nodo	Hora de llegada del dato (Bróker – ESP8266)	Atributo	Valor actual	Tiempo de retardo entre cada atributo (ms)	Promedio del tiempo de retardo entre cada atributo (ms)
1	19:33:44.014	Temperatura	23.30	-	474
	19:33:44.488	Humedad	52.90	474	
2	19:33:45.508	Temperatura	23.30	1020	764
	19:33:46.016	Humedad	52.90	508	
3	19:33:47.025	Temperatura	23.30	1009	743.5
	19:33:47.503	Humedad	52.90	478	
4	19:33:48.516	Temperatura	23.30	1013	761
	19:33:49.025	Humedad	52.90	509	
5	19:33:50.038	Temperatura	23.30	1013	758
	19:33:50.541	Humedad	52.90	503	
6	19:33:51.554	Temperatura	23.30	1013	748
	19:33:52.037	Humedad	52.90	483	
7	19:33:53.059	Temperatura	23.30	1022	749
	19:33:53.535	Humedad	52.90	476	
8	19:33:54.556	Temperatura	23.30	1021	766.5
	19:33:55.068	Humedad	52.90	512	
9	19:33:56.087	Temperatura	23.30	1019	745.5
	19:33:56.559	Humedad	52.90	472	
10	19:33:57.574	Temperatura	23.30	1015	763
	19:33:58.085	Humedad	52.90	511	
Promedio total del tiempo de retardo entre cada atributo (ms)					**727.25**

Tabla 4: Resultados de las pruebas de latencia por atributo IoT - MQTT. Fuente: Del autor.

En la tabla 4 se describen las 10 primeras iteraciones de datos por nodo entre atributos, resultados de las pruebas realizadas con el protocolo MQTT.

ANEXO 4

Tiempos de retardo entre atributos. Aplicativo IoT - Firebase

Iteración de datos por nodo	Hora de llegada del dato (Firebase – ESP8266)	Atributo	Valor actual	Tiempo de retardo entre cada atributo (ms)	Promedio del tiempo de retardo entre cada atributo (ms)
1	21:02:37.943	S. Luminaria	Apagado	-	217
	21:02:38.160	H. Luminaria	Apagado	217	
2	21:02:39.258	S. Luminaria	Apagado	1098	655.5
	21:02:39.471	H. Luminaria	Apagado	213	
3	21:02:40.504	S. Luminaria	Apagado	1033	623
	21:02:40.717	H. Luminaria	Apagado	213	
4	21:02:41.717	S. Luminaria	Apagado	1000	603.5
	21:02:41.924	H. Luminaria	Apagado	207	
5	21:02:42.916	S. Luminaria	Apagado	992	602.5
	21:02:43.129	H. Luminaria	Apagado	213	
6	21:02:44.111	S. Luminaria	Apagado	982	596.5
	21:02:44.322	H. Luminaria	Apagado	211	
7	21:02:45.283	S. Luminaria	Apagado	961	588.5
	21:02:45.499	H. Luminaria	Apagado	216	
8	21:02:46.493	S. Luminaria	Apagado	994	603
	21:02:46.705	H. Luminaria	Apagado	212	
9	21:02:47.697	S. Luminaria	Apagado	992	603
	21:02:47.911	H. Luminaria	Apagado	214	
10	21:02:48.907	S. Luminaria	Apagado	996	603.5
	21:02:49.118	H. Luminaria	Apagado	211	
Promedio total del tiempo de retardo entre cada atributo (ms)					**569.6**

Tabla 5: Resultados de las pruebas de latencia por atributo IoT - Firebase. Fuente: Del autor.

En la tabla 5 se describen las 10 primeras iteraciones de datos por nodo entre atributos, resultados de las pruebas realizadas con la plataforma Firebase Realtime Database.

ANEXO 5

Tiempos de retardo por evento

Atributo	Hora de llegada del primer dato (Bróker - ESP8266)	Valor inicio	Hora de llegada del siguiente dato (Bróker - ESP8266)	Valor fin	Tiempo de retardo por cada atributo (ms)
Temperatura	21:42:25.148	23.00	21:42:27.686	23.10	2538
Humedad	21:42:26.157	60.10	21:42:28.674	59.40	2517
Promedio total del tiempo de retardo por evento					**2527.5**

Tabla 6: Resultados de las pruebas de latencia por evento IoT - MQTT. Fuente: Del autor.

Atributo	Hora de llegada del primer dato (Firebase - ESP8266)	Valor inicio	Hora de llegada del siguiente dato (Bróker - ESP8266)	Valor fin	Tiempo de retardo por cada atributo (ms)
Sala luminaria	21:49:21.790	Apagado	21:49:23.878	Encendido	2088
Habitación luminaria	21:49:21.996	Apagado	21:49:24.113	Encendido	2117
Promedio total del tiempo de retardo por evento					**2102.5**

Tabla 7: Resultados de las pruebas de latencia por evento IoT - Firebase. Fuente: Del autor.

En la tabla 6 como 7 se describen las iteraciones de datos por evento o suceso desde que llega el primer dato hasta la llegada del otro, resultados de las pruebas realizadas con el protocolo MQTT como la plataforma Firebase Realtime Database.