CENTRO UNIVERSITARIO TECNOLÓGICO
CEUTEC

FACULTAD DE INGENIERÍA

PROYECTO DE GRADUACIÓN

APLICACIÓN DEL PROTOCOLO MQTT (Message Queuing Telemetry Transport) PARA MONITOREO Y RECOLECCIÓN DE DATOS PARA APLICACIONES IoT (Internet of Things) BASADO EN ESP32.

SUSTENTADO POR

CESAR ALBERTO FERNANDEZ MORAZAN, 31721226

PREVIA INVESTIDURA AL TITULO INGENIERIA EN ELECTRONICA

TEGUCIGALPA **HONDURAS, C.A.**

MARZO 2022

CENTRO UNIVERSITARIO TECNOLÓGICO
CEUTEC

INGENIERÍA EN ELECTRÓNICA

AUTORIDADES UNIVERSITARIAS

RECTOR
MARLON ANTONIO BREVÉ REYES

SECRETARIO GENERAL
ROGER MARTÍNEZ MIRALDA

VICERRECTORA ACADÉMICA CEUTEC
DINA ELIZABETH VENTURA DÍAZ

DIRECTORA ACADÉMICA CEUTEC
IRIS GABRIELA GONZALES ORTEGA

TEGUCIGALPA **HONDURAS, C.A.**

MARZO 2022

APLICACIÓN DEL PROTOCOLO MQTT (Message Queuing Telemetry Transport) PARA MONITOREO Y RECOLECCIÓN DE DATOS PARA APLICACIONES IoT (Internet of Things) BASADO EN ESP32.

TRABAJO PRESENTADO EN EL CUMPLIMIENTO DE LOS REQUISITOS
EXIGIDOS PARA OPTAR AL TÍTULO DE:

INGENIERÍA EN ELECTRÓNICA

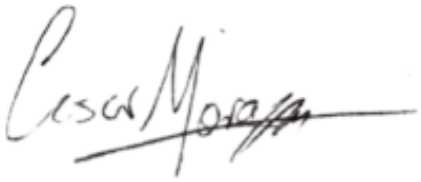
ASESOR:
KARIO ALEXANDRO VILLAFRANCA REYES

TERNA EXAMINADORA:
LUCY ALEXANDRA LÓPEZ QUINTANILLA
MANUEL ALEJANDRO ELVIR OSORIO

TEGUCIGALPA **HONDURAS, C.A.**

MARZO 2022

DERECHOS DE AUTOR



Dedicatoria

Dedicado a mi Madre Johanna Valladares.

Agradecimientos

A las personas que me apoyaron e instruyeron en estos años de carrera. A mi familia, mi abuela Sandra Valladares, a mi novia Susy Rodríguez, mi padre Cesar Fernandez, a mi hermano Josué Miguel y mi abuelo Jorge Morazán. A mis amigos, compañeros y mis maestros que me han instruido ayudándome a adquirir nuevos conocimientos y habilidades en la rama de la ingeniería en electrónica.

Resumen ejecutivo

La conexión a internet de todo tipo de dispositivo electrónicos a día de hoy avanza a pasos agigantados, a esta conexión de dispositivos a internet la conocemos como el internet de las cosas, gracias a esto se pueden tener dispositivos de control y monitorización tales como sensores, actuadores, dispositivos electromecánicos, hasta dispositivos para la industria y dispositivos personales, estas tecnologías han creado nuevas áreas de desarrollo e implementación como la domótica, ha mejorado la automatización de procesos a distancia, y es utilizada en sectores como industrias de alimentos, industrias textiles, en automóviles, en dispositivos de telecomunicaciones, etc.

En este proyecto se detalla como montar y utilizar un dispositivo para el internet de las cosas, comenzando con un dispositivo donde se puede monitorear la humedad y temperatura a distancia o en el hogar, hasta un dispositivo de monitoreo de gas natural y de fugas pudiendo ser aplicado a una industria, como también la aplicación de un sensor electrónico que se activa al detectar una llama por medio de un fotodiodo que puede servir en caso detectarse un incendio en cualquier área ya que al detectar dicho fenómeno estos enviaran por medio del patrón de publicación/suscripción del protocolo MQTT un tópico con una carga útil, un mensaje, el cual al estar debidamente programado donde otros sistemas pueden recibir estos datos al estar suscritos al tópico y así poder activar alarmas o dispositivos de protección para evitar daños.

Para el desarrollo de este sistema de monitoreo y recolección de datos se implementó la placa de desarrollo ESP32 desarrollada por Espressif la cual es ideal para el montaje de este tipo de sistemas debido a que cuenta con la posibilidad de conexión a internet gracias a su modulo WiFi integrado y por su bajo costo en comparación con otras placas de desarrollo.

El protocolo MQTT se implementó en este proyecto gracias a que es un protocolo fácil de utilizar e implementar, cabe destacar que debido a esto se ha convertido en un estándar ISO para la comunicación entre maquinas mediante el internet de las cosas, este utiliza la conexión mediante TCP/IP, por lo cual cada dispositivo conectado a un broker MQTT podrá ser identificado debido a una dirección IP.

OASIS lo describe como el protocolo MQTT diseñado como un transporte de mensajería de publicación/suscripción extremadamente ligera que es ideal para conectar dispositivos remotos con un espacio de código pequeño y un ancho de banda de red mínimo. Este protocolo de comunicación para IoT actualmente es utilizado por casi todos los sectores comerciales e industriales, algunos de los más reconocidos por esto son IBM, Amazon, Facebook y Microsoft.

El proyecto realizado cuenta con una gran flexibilidad de implementación debido a que se pueden agregar cualquier tipo de sensores para el monitoreo o dispositivos de activación, todo esto gracias a la programación en la placa ESP32 y la configuración del mismo, el broker tomado para este proyecto es CloudMQTT pudiendo ser utilizado cualquier otro, pero por su precio mensual de 5$ se cuenta con los mismo elementos que cualquier otro broker, donde la única ventaja en otros con pagos más elevados es la integración de paneles de monitoreo personalizados, para esto se utilizó a la aplicación gratuita para smartphones "IoT MQTT panel" la cual permite poder crear paneles de monitoreo de los sensores y datos recibidos.

Palabras clave: IoT, MQTT, OASIS, ISO, TCP/IP, WiFi, Telemetría.

Abstract

The internet connection of all kinds of electronic devices today is advancing by leaps and bounds, we know this connection of devices to the internet as the internet of things, thanks to this you can have control and monitoring devices such as sensors, actuators, electromechanical devices, even devices for industry and personal devices, these technologies have created new areas of development and implementation such as home automation, have improved remote process automation, and are used in sectors such as food industries, textile industries, in automobiles, in telecommunication devices, etc.

This project details how to assemble and use a device for the internet of things, starting with a device where humidity and temperature can be monitored remotely or at home, to a natural gas and leak monitoring device that can be applied to an industry, as well as the application of an electronic sensor that is activated by detecting a flame by means of a photodiode that can be used in case a fire is detected in any area since upon detecting said phenomenon they will send through the pattern of publication/subscription of the MQTT protocol a topic with a payload, a message, which, when properly programmed, where other systems can receive this data when subscribed to the topic and thus be able to activate alarms or protection devices to avoid damage.

For the development of this monitoring and data collection system, the ESP32 development board developed by Espressif was implemented, which is ideal for assembling this type of system because it has the possibility of connecting to the internet thanks to its WiFi module. integrated and for its low cost compared to other development boards.

The MQTT protocol was implemented in this project thanks to the fact that it is an easy to use and implement protocol, it should be noted that due to this it has become an ISO standard for communication between machines through the Internet of Things, it uses the connection through TCP/IP, whereby each device connected to an MQTT broker can be identified by an IP address.

OASIS describes it as the MQTT protocol designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small

code space and minimal network bandwidth. This communication protocol for IoT is currently used by almost all commercial and industrial sectors, some of the most recognized for this are IBM, Amazon, Facebook and Microsoft.

The project carried out has a great flexibility of implementation because any type of sensors for monitoring or activation devices can be added, all this thanks to the programming on the ESP32 board and its configuration, the broker taken for this project is CloudMQTT and any other broker can be used, but for its monthly price of $5 it has the same elements as any other broker, where the only advantage in others with higher payments is the integration of custom monitoring panels, for this it was used to the free application for smartphones "IoT MQTT panel" which allows to create monitoring panels of the sensors and data received.

Keywords: IoT, MQTT, OASIS, ISO, TCP/IP, WiFi, Telemetry.

Índice de contenido

Índice de Figuras

Índice de tablas

Índice de gráficos

Glosario

IoT: Internet of Things (Internet de las cosas).

MQTT: Message Queuing Telemetry Transport, a día de hoy MQTT es el nombre del protocolo y sus siglas no significan nada.

SoC: System on a chip (Sistema en un chip).

SCADA: Supervisory Control and Data Acquisition (Supervisión, control y adquisición de datos).

RFID: Radio Frequency Identification (Identificación por radiofrecuencia).

Estándar Oasis: Organización para el Avance de Estándares de Información Estructurada

GPIO: General Purpose Input/Output, Entrada/Salida de Propósito General.

Ping: Unidad de medida que sirve para medir la latencia.

Latencia: Tiempo exacto que tarda un paquete de datos en ser transmitido por una red

Websocket: Tecnología avanzada que hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor

Broker: Entidad intermediaria que permite que los clientes de MQTT se comuniquen.

TCP: Protocolo de Control de Transmisión que permite establecer una conexión y el intercambio de datos entre dos anfitriones.

IP: Protocolo de internet, utiliza direcciones series de cuatro octetos con formato de punto decimal.

Capítulo I: Introducción

Hoy en día en pleno siglo XXI la integración del internet, las tecnologías electrónicas y digitales son una necesidad para cada una persona de las personas en la sociedad, estas tecnologías se han ido desarrollando para ser cada vez más autónomas y más avanzadas, la cuarta revolución industrial o la industria 4.0 como muchos investigadores y expertos en el tema lo llaman está marcada por la aparición de nuevas tecnologías como la robótica, la analítica, la inteligencia artificial, las tecnologías cognitivas, la nanotecnología y el Internet of Things (IoT).

El Internet de las Cosas, IoT (Internet of Things), hace referencia a todas las tecnologías basada en la conexión de objetos cotidianos a Internet que intercambian, agregan y procesan información sobre su entorno físico para proporcionar servicios de valor añadido a los usuarios finales, la frase fue acuñada por Kevin Ashton en el año 1999, donde la definió como la conexión de las máquinas a internet para poder compartir información entre ellas y con los seres humanos. Para que ocurra una interacción de datos entre los dispositivos conectados a internet era necesario contar con un protocolo de comunicación, los medios para conectarse a dispositivos y controlarlos se denominaron telemetría. La incompatibilidad entre protocolos en el inicio fue una limitación para poder conectarse entre varios sistemas, debido a esto dos ingenieros de IBM crearon el protocolo de comunicación MQTT, el cual se desarrolló para ser el protocolo ideal para la comunicación por el internet de las cosas.

El protocolo MQTT (Message Queue Telemetry Transport) o en español Transporte de Telemetría de Cola de Mensajes fue creado por Stanford-Clark y Nipper ambos ingenieros de IBM donde su objetivo era crear el protocolo cumpliendo con 3 objetivos principales los cuales era ser un protocolo para la comunicación por telemetría liviano, fácil de entender y fácil de implementar, los cuales se mantuvieron desde el día de su creación en 1998 para la monitorización de oleoductos en ubicaciones remotas vía satélite, con el tiempo se siguió desarrollando y mejorando este protocolo hasta que alrededor del año 2008 se empezó a utilizar ampliamente en todo tipo de sistemas conectados a IoT, gracias a esto y a su 3 principales características este protocolo en el año 2014 se convirtió en un estándar OASIS (Organization for the Advancement of Structured Information Standards) y en 2016 paso a ser un estándar ISO (International Organization for Standardization).

Este proyecto tiene el objetivo de realizar una aplicación del protocolo MQTT para el monitoreo y recolección de datos para aplicaciones IoT y telemetría basada en el microcontrolador ESP32, así como enumerar sus múltiples usos, aplicaciones y posibilidades y así desarrollar una serie de conclusiones.

Una vez que se realiza un sistema como el planteado en este proyecto se puede observar que gracias al crecimiento exponencial del IoT casi todo dispositivo puede estar conectado a internet para el monitoreo y transmisión de datos, donde la única limitación de un sistema de este tipo es que requiere la conexión a internet de manera constante en caso de requerir un monitoreo en tiempo real.

Capitulo II: Planteamiento del problema

2.1. Antecedentes

Sistemas inteligentes para monitoreo y recolección de datos a distancia: Telemetría, MQTT e IoT

2.1.1. Telemetría

(The Editors of Encyclopaedia Britannica, s. f.) afirma: “La telemetría se puede describir como un proceso de comunicaciones altamente automatizado que implica la recopilación de mediciones y otros datos en puntos remotos o inaccesibles, antes de la transmisión al equipo receptor con fines de seguimiento y control.”

La propia telemetría puede rastrear su propio pasado hasta el siglo XIX. Este es el período en el que la información de telemedida por cable tiene sus orígenes, siendo 1845 el año en el que se desarrolló uno de los primeros circuitos de transmisión de datos, entre el Palacio de Invierno del Zar Ruso y el cuartel general del ejército. En 1874, los ingenieros franceses hicieron construir un sistema de sensores meteorológicos y de profundidad de nieve en el Mont Blanc para la transmisión de información en tiempo real a París (Orielsystems, 2022).

Luego, en 1901, el inventor estadounidense C. Michalke patentó un circuito, conocido como selsyn, que permitía el envío a distancia de información de rotación sincronizada. Los desarrollos que culminaron gradualmente en los sistemas de adquisición de datos de hoy en día llegaron con fuerza y rapidez después de eso, con el año 1906 viendo la construcción de un conjunto de estaciones sísmicas con telemedida al Observatorio Pulkovo en Rusia. Luego, Commonwealth Edison desarrolló un sistema de telemetría en 1912, para monitorear las cargas eléctricas de su red eléctrica. En 1914, se completó el Canal de Panamá y se utilizaron extensos sistemas de telemetría para monitorear las esclusas y los niveles de agua (Orielsystems, 2022).

El comienzo de la década de 1930 vio el desarrollo simultáneo de la radiosonda por parte del ruso Pavel Molchanov y Robert Bureau en Francia, y la telemetría inalámbrica hizo sus primeras apariciones. En el sistema de Molchanov, la modulación de las mediciones de temperatura y presión fue posible con su conversión a código Morse inalámbrico. Mientras tanto, otro sistema, "Messina", que consiste en señales de radio primitivas multiplexadas, fue utilizado por el cohete alemán V-2 para informar de cuatro parámetros del cohete, aunque su falta de fiabilidad era tal que Wernher von Braun afirmó una vez que mirar el cohete a través de binoculares fue una mejor idea (Orielsystems, 2022).

Los sistemas de telemetría originales se denominaron de supervisión porque se utilizaban para monitorear la distribución de energía eléctrica. Dichos sistemas se extendieron a otros campos además de las redes eléctricas y sufrieron amplias mejoras, que culminaron con la introducción en 1960 de los llamados principio de interrogación-respuesta, un arreglo altamente automatizado en el que la instalación de transmisor-receptor en el punto de medición transmite automáticamente los datos necesarios solo cuando se le indica que lo haga. La técnica se aplica ampliamente en todo el mundo en campos tales como sistemas de monitoreo y control de oleoductos y oceanografía, en los que una red de boyas transmite información a pedido a una estación maestra (The Editors of Encyclopaedia Britannica, s. f.).

Ciertamente, la gama de aplicaciones de la telemetría se ha ampliado significativamente, siendo la ciencia espacial, la agricultura, el automovilismo, las pruebas de vuelo, la inteligencia militar, la medicina y la distribución de recursos solo algunos de los campos que se han beneficiado de ella.

2.1.2. Historia del IoT

La idea de agregar sensores e inteligencia a los objetos básicos se discutió durante las décadas de 1980 y 1990 (y posiblemente hay algunos antepasados mucho más antiguos), pero aparte de algunos proyectos iniciales, incluida una máquina expendedora conectada a Internet, el progreso fue lento simplemente porque la tecnología no estaba lista. Los chips eran demasiado grandes y voluminosos y no había forma de que los objetos se comunicaran eficazmente (Ranger, 2020).

Se necesitaban procesadores que eran lo suficientemente baratos y con bajo consumo de energía como para ser casi desechables antes de que finalmente se volviera rentable para conectar miles de millones de dispositivos. La adopción de etiquetas RFID (chips de bajo consumo que pueden comunicarse de forma inalámbrica) resolvió parte de este problema, junto con la creciente disponibilidad de Internet de banda ancha y redes móviles e inalámbricas. La adopción de IPv6, que, entre otras cosas, debería proporcionar suficientes direcciones IP para todos los dispositivos que el mundo (o de hecho esta galaxia) pueda necesitar, también fue un paso necesario para que IoT escale (Ranger, 2020).

Kevin Ashton acuñó la frase "Internet de las cosas" en 1999, aunque la tecnología tardó al menos otra década en ponerse al día con la visión.

Agregar etiquetas RFID a costosos equipos para ayudar a rastrear su ubicación fue una de las primeras aplicaciones de IoT. Pero desde entonces, el costo de agregar sensores y una conexión a Internet a los objetos ha seguido cayendo, y los expertos predicen que esta funcionalidad básica algún día podría costar tan solo 10 centavos, lo que haría posible conectar casi todo a Internet. (Ranger, 2020).

El IoT fue inicialmente más interesante para los negocios y la fabricación, donde su aplicación a veces se conoce como máquina a máquina (M2M), pero ahora el énfasis está en llenar hogares y oficinas con dispositivos inteligentes, transformándolo en algo que es relevante para casi todos. Las primeras sugerencias para los dispositivos conectados a Internet incluyeron 'blogjects' (objetos que escriben en blogs y registran datos sobre sí mismos en Internet), computación ubicua (o 'ubicomp'), computación invisible y computación omnipresente. Sin embargo, fue Internet de las cosas e IoT lo que se mantuvo (Ranger, 2020).

2.1.3. El protocolo MQTT

A finales de los noventa, los sistemas de Control de Supervisión y Adquisición de Datos se usaban para administrar grandes infraestructuras industriales, como oleoductos o redes eléctricas (de hecho, SCADA todavía se usa mucho en la actualidad). Esencialmente, los sistemas SCADA permitieron a los operadores controlar varios aspectos de la infraestructura de forma remota, como encender o apagar un generador en una red eléctrica.

Por cierto, debido a que los sistemas SCADA pueden operar maquinaria y dispositivos conectándose a ellos de forma remota, en gran medida se consideran precursores de los sistemas IoT actuales (Behrens, 2019).

Los medios para conectarse a dispositivos remotos y controlarlos se denominaron "telemetría". El problema con la telemetría en esos días era que había varios protocolos, generalmente desarrollados por los fabricantes de los dispositivos o maquinaria y, por lo tanto, propietarios. Esto resultó en todo tipo de problemas de compatibilidad. Dos ingenieros de IBM decidieron que intentarían resolver esto desarrollando un único protocolo de código abierto para gobernarlos a todos. Estos ingenieros eran Andy Stanford-Clark y Arlen Nipper, juntos comenzaron a trabajar en algo que denominaron "Message Queuing Telemetry Transport" o MQTT (Behrens, 2019).

Aunque inicialmente solo estaba destinado a administrar componentes de oleoductos de forma remota vía satélite, pronto se hizo evidente que MQTT también podría tener aplicaciones fuera de los sistemas SCADA. Andy Stanford-Clark (IBM) y Arlen Nipper (que entonces trabajaba para Eurotech, Inc.) fueron los autores de la primera versión del protocolo en 1999. Se utilizó para monitorear oleoductos dentro del sistema de control industrial SCADA. El objetivo era tener un protocolo que fuera eficiente en el ancho de banda, liviano y que usara poca energía de la batería, porque los dispositivos estaban conectados a través de un enlace satelital que, en ese momento, era extremadamente costoso (Behrens, 2019).

A medida que IoT florecía en la primera década del siglo XXI, los ingenieros y fabricantes necesitaban una forma de conectar sus dispositivos novedosos. La sofisticación simple y la utilidad de MQTT se adaptaban muy bien a sus propósitos, y poco a poco se fue generalizando. Luego, alrededor de 2008, llegó un punto de inflexión (como lo llama Stanford-Clark) cuando se lanzó el broker MQTT de código abierto Mosquitto (hoy llamado Eclipse Mosquitto). De repente, MQTT era más accesible y su tasa de uso aumentó drásticamente (Behrens, 2019).

En 2014, MQTT era un estándar OASIS y en 2016 se convirtió en un estándar ISO. En este punto, ya conectaba millones de dispositivos en todo el mundo en todo tipo de aplicaciones e industrias. IBM, Amazon y Microsoft son solo algunos ejemplos de grandes empresas que han adoptado MQTT como su protocolo principal (HiveMQ, 2022).

Históricamente, el "MQ" en "MQTT" provenía de la línea de productos IBM MQ (entonces 'MQSeries') MQ, donde significa "Message Queue". Sin embargo, el protocolo proporciona mensajes de publicación y suscripción (sin colas, a pesar del nombre). En la especificación abierta por IBM como versión 3.1, el protocolo se denominó "Transporte de telemetría MQ". Las versiones posteriores publicadas por OASIS se refieren estrictamente al protocolo simplemente como "MQTT", aunque el comité técnico en sí se denomina "Comité Técnico de Transporte de Telemetría de Cola de Mensajes de OASIS". Desde 2013, "MQTT" no significa nada (HiveMQ, 2020).

2.2. Enunciado del problema

Dado al auge y crecimiento exponencial en el uso de dispositivos y protocolos IoT y la creciente creación de dispositivos que se conectan a internet para automatización es importante para futuros ingenieros conocer todo lo que conlleva esta tecnología y así aprovechar las oportunidades para conocer e incluso poder desarrollar un producto de uso personal o comercial basado en ello, como se observa en la actualidad desde industrias gigantes hasta empresas en pleno desarrollo, han estado apostando por la automatización y el intercambio rápido y receptivo de grandes cantidades de pequeños datos.

Esta proliferación ha sido posible gracias a los siguientes hechos: precio y tamaño reducidos y mayor rendimiento de los microchips como en este proyecto donde se utilizará un ESP32 SoC. (System on Chip), el cual es de bajo costo. El sector del transporte y la cadena de suministro estará entre los primeros en ver ganancias significativas de productividad de las tecnologías IoT y los datos de telemetría de vehículos. La instalación generalizada de sensores, el despliegue de sistemas basados en redes neuronales y el análisis de big data permitirán la recopilación de conocimiento oculto de objetos del mundo real cuyos parámetros a primera vista parecen desestructurados e inconexos.

Con la ayuda de estos sistemas de monitoreo y recolección de datos el uso del protocolo de red MQTT en transporte y logística para transmitir datos de telemetría, la automatización será mucho más eficiente y de manera más económica dado a las características con las que cuenta el protocolo MQTT. Gracias al estudio y aplicación que se

realizará se podrá observar de primera mano la utilidad y las muchas tareas que se pueden realizar y monitorear mediante IoT.

2.3. Preguntas de investigación

- De forma general ¿Qué es y cómo funciona un sistema de monitoreo utilizando MQTT como protocolo de comunicación?
- ¿Para qué sirve comprender la construcción, el funcionamiento y programación de un sistema de monitoreo y recolección de datos?
- ¿En qué sectores y para qué tipo de fines se puede aplicar un sistema de monitoreo y recolección de datos con MQTT?

2.4. Hipótesis y variables de investigación

***Hi*i** La creación de aplicaciones para el monitoreo y recolección de datos para el internet de las cosas utilizando la placa de desarrollo ESP32 con el protocolo MQTT son una manera fácil y eficaz para conocer sobre estos entornos electrónicos e informáticos y sus innumerables aplicaciones. **Comprobada.**

***Ha*a** Una de las razones importantes para comprender el desarrollo y posibilidades de IoT es el crecimiento exponencial de las aplicaciones y su uso en el mundo de la tecnología al día de hoy para poder automatizar procesos. **Comprobada**

***Ho*o** La creación de aplicaciones para el monitoreo y recolección de datos para el internet de las cosas utilizando la placa de desarrollo ESP32 con el protocolo MQTT no son una manera fácil para comprender sobre estos entornos electrónicos e informáticos**. Rechazada**

2.4.1. Operacionalización de las variables

Variables	Definición Conceptual	Definición Operacional	Dimensiones	Indicadores
Protocolos para IoT	MQTT es un protocolo de mensajería estándar de OASIS para Internet de las cosas (IoT). Está diseñado como un transporte de mensajería de publicación / suscripción extremadamente liviana que es ideal para conectar dispositivos remotos con un código pequeño y un ancho de banda de red mínimo.	MQTT se utiliza en una amplia variedad de industrias, como la automotriz, la fabricación, las telecomunicaciones, el petróleo, etc.	Protocolos bidireccionales	Mensajería remota y envió de datos a distancia
Conexión con IoT	Agrupación e interconexión de dispositivos y objetos a través de una red (bien sea privada o Internet, la red de redes), dónde todos ellos podrían ser visibles e interaccionar	Es un sistema de dispositivos informáticos que tiene la capacidad de transferir datos a través de una red sin requerir interacción humana o de persona a computadora.	Conexión a internet de dispositivos	Comunicación con la placa ESP32 y los sensores a usar.
Broker MQTT	El Broker es el intermediario que hará de función de comunicación entre el protocolo MQTT y el dispositivo a usar, ESP32, Arduino, celular, etc…	El Broker o intermediario es responsable de recibir todos los mensajes, filtrar los mensajes, determinar quién está suscrito a cada mensaje y enviar el mensaje a estos clientes suscritos	Intermediario para el protocolo MQTT	Comunicación Pub/Sub, envió de mensajes al sistema.
Latencia en comunicación	La latencia es la distancia entre dos puntos que están intentando comunicarse y los obstáculos que se pueden presentar en esta comunicación.	Ping su unidad de medida que representa en milisegundos la demora en la transmisión de los paquetes de información de un punto a otro.	Mensajes en tiempo real	Eficiencia de la comunicación y rapidez en transferencia de datos.

Arquitecturas de publicación/suscripción	Es un patrón de mensajería en el que los remitentes de mensajes, denominados editores, no programan los mensajes para que se envíen directamente a receptores específicos, denominados suscriptores, sino que clasifican los mensajes publicados en clases sin saber qué suscriptores, si los hay o puede haber.	Permite que los servicios se comuniquen de forma asíncrona, con latencias de alrededor de 100 milisegundos.	Comunicaciones bidireccionales	Publicación, suscripción, conexión, desconexión mediante el sistema a crear.
Conocimientos básicos de la electrónica y programación	Son todos aquellos conocimientos básicos ya sea sobre el funcionamiento de lenguaje de programación, microcontroladores, microprocesadores diagramas electrónicos, fenómenos físicos, comportamientos en un sistema electrónico.	Estos serían aplicados y mejorados en el transcurso de la elaboración del proyecto fase 1.	Electrónica y programación	Trazabilidad en el proyecto, facilidad de creación
Obtención de datos en tiempo real	Son los datos que serán enviados a través desde el dispositivo monitoread a través del protocolo hasta el panel MQTT	Estos datos serán enviados según la configuración y tiempo programado, donde se nos estará enviando mediante un Callback la información y datos en tiempo real	Internet de las cosas	Datos recolectados, datos para estadísticas.
Telemetría en el proyecto	La telemetría es una tecnología que permite medir y rastrear magnitudes físicas de forma remota para que un operador pueda obtener, generalmente de manera inalámbrica.	Esto será gracias a la conexión a internet que se dispone para realizar al proyecto, gracias a esto se podrán recibir y enviar datos desde largas distancias o de manera remonta	Recolección de datos de manera remota	Pruebas de conexión al broker y datos existentes sobre la latencia promedio por conexión.

Tabla 2.1, *Operacionalización de variables*

2.5. Justificación de la Investigación

Con esta investigación y aplicación se da el propósito de dar a conocer y explicar la comunicación por el IoT mediante el protocolo de comunicación MQTT, la placa de desarrollo ESP32 para la demostración del envió de datos hacia un panel MQTT donde se visualizarán que los datos se están recibiendo

Siempre se ha hablado de IoT como la tecnología que iba cambiar la vida de las personas, la tecnología y el mundo, se decía que para el año 2020-2025 aproximadamente 20 mil millones de dispositivos IoT estarían aplicados, estos datos basados en analistas de mercado como IDC, cual predicción puede ser real ya que la tecnología 5G está casi próxima a su implementación, también porque se estima que las industrias como la manufactura, servicio, transporte, logística, entre otras gasten más de 40 mil millones de dólares cada una en plataformas y servicios IoT para la automatización, comunicación, conexión, almacenamiento de datos, generar estadísticas utilizando esto, por lo cual se necesita un protocolo de comunicación liviano y eficiente para esta labor, MQTT es un protocolo transporte de datos y mensajería del tipo publicación/suscripción y es simple como fácil de implementar, lo cual es óptimo para el uso en el IoT, el cual fue inventado en 1999, y fue creado para la telemetría de oleoductos por medio de satélites, esto debido a que los sensores de los oleoductos se encontraban en lugares remotos y también se necesitaba monitorear las cargas de baterías de los mismos.

Debido a todas las oportunidades que estos sistemas ofrecen para automatización es importante conocer cómo desarrollar un sistema que puede servir para monitorear cosas pequeñas como un sensor, hasta grandes maquinarias o industrias. Esta investigación no sólo ayudara a adquirir conocimientos sobre el funcionamiento del protocolo MQTT y aplicaciones de IoT, si no también abarcar temas como funciones que se realizan y como es la estructura de dicho protocolo.

Capitulo III: Objetivos

3.1. Objetivo General:

- Realizar una aplicación de alarma de incendio y de fugas de gas implementando el protocolo MQTT para el internet de las cosas (IoT) basada en el microcontrolador ESP32 para el monitoreo y recolección de datos en tiempo real por medio del broker Cloud MQTT y el panel de monitoreo IoTMQTTPanel.

3.1.1. Objetivos Específicos:

- Crear un programa para la conexión y transferencia de datos desde de la placa de desarrollo ESP32 para la comunicación por telemetría mediante el protocolo MQTT para el monitoreo dispositivos como sensores.

- Explicar el funcionamiento sobre la monitorización en tiempo real de los valores de módulos y sensores mediante este protocolo.

- Demostrar la flexibilidad de un sistema basado en ESP32 y MQTT así las múltiples áreas donde puede ser implementado.

Capítulo IV: Marco Teórico

4.1 Microcontrolador ESP32

Un microcomputador integrado o microcontrolador (en inglés "Embedded Microcontroller"), abreviado como μC o MCU, está constituido por un único circuito integrado que contiene la unidad central de proceso, memoria activa (RAM), memoria pasiva (EPROM) y algunas unidades de acoplamiento de periféricos. Estas interfases pueden ser funciones de control de entrada o salida, tan comunes como encender o apagar dispositivos y/o monitorear determinadas condiciones. Entre algunos de los ejemplos del empleo de este popular dispositivo destacan: los sistemas de control de procesos automotrices (bolsas de aire, control del motor, frenos ABS, etc.); los medidores de servicios (de energía eléctrica, agua o gas); los productos de línea blanca (lavadoras, hornos de microondas, procesadores de comida, secadoras, etc.); algunos artículos de consumo común (monitores de glucosa, dispositivos de presión de sangre, teléfonos, radios, relojes despertadores, etc.); algunos juguetes y aplicaciones de Internet, entre muchos otros (Mandado Pérez et al., 1995; Mijarez Castro, 2014).

El ESP32 es un microcontrolador de bajo costo y bajo consumo con Wi-Fi y Bluetooth integrados. Es el sucesor del ESP8266, que también es un microchip Wi-Fi de bajo costo, aunque con una funcionalidad muy limitada (Amazon Web Services, s. f.).

Los fabricantes actuales del micotrontrolador ESP32 Espressif (2021), nos indican desde la hoja de datos o datasheet que el ESP32 es un único chip combinado de Wi-Fi y Bluetooth de 2,4 GHz diseñado con el TSMC de 40 nm de potencia ultrabaja tecnología. Está diseñado para lograr el mejor rendimiento de potencia y RF, mostrando robustez, versatilidad y confiabilidad en una amplia variedad de aplicaciones y escenarios. El ESP32 está diseñado para aplicaciones móviles, dispositivos electrónicos portátiles e Internet de las cosas (IoT). Cuenta con todas las características de vanguardia de los chips de bajo consumo, incluida la activación de reloj de granularidad fina, múltiples modos de alimentación y escalado dinámico de alimentación. Por ejemplo, en un escenario de aplicación de concentrador de sensores IoT de baja potencia, ESP32 se activa periódicamente solo cuando se detecta una condición específica. El ciclo de trabajo bajo se usa para minimizar la cantidad de energía

que gasta el chip. La salida del amplificador de potencia también es ajustable, lo que contribuye a un compromiso óptimo entre rango de comunicación, velocidad de datos y consumo de energía.

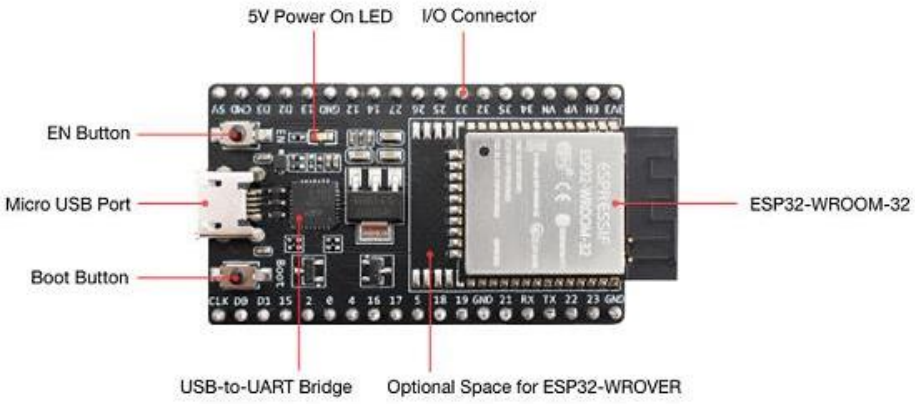


Figura 4.1 Esquema de placa microcontrolador ESP32

4.1.1. Lista de especificaciones ESP32

Las características del ESP32 incluyen:

Procesador:

- CPU: microprocesador de 32-bit Xtensa LX6 de doble núcleo (o de un solo núcleo), operando a 160 o 240 MHz y rindiendo hasta 600 DMIPS
- Coprocesador de ultra baja energía (ULP)

Memoria:

- 520 KiB SRAM

Conectividad inalámbrica:

- Wi-Fi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR y BLE

Interfaces periféricas:

- 12-bit SAR ADC de hasta 18 canales

- 2 × 8-bit DACs
- 10 × sensores de tacto (sensores capacitivos GPIOs)
- 4 × SPI
- 2 × interfaces I²S
- 2 × interfaces I²C
- 3 × UART
- Controlador host SD/SDIO/CE-ATA/MMC/eMMC
- Controlador esclavo SDIO/SPI
- Interfaz Ethernet MAC con DMA dedicado y soporte para el protocolo IEEE 1588 Precision Time Protocol
- Bus CAN 2.0
- Controlador remoto infrarrojo (TX/RX, hasta 8 canales)
- Motor PWM
- LED PWM (hasta 16 canales)
- Sensor de efecto Hall
- Pre-amplificador analógico de ultra baja potencia

Seguridad:

- Soporta todas las características de seguridad estándar de IEEE 802.11, incluyendo WFA, WPA/WPA2 y WAPI
- Arranque seguro
- Cifrado flash
- 1024-bit OTP, hasta 768-bit para clientes
- Criptografía acelerada por hardware: AES, SHA-2, RSA, criptografía de curva elíptica (ECC), generador de números aleatorios (RNG)

Administración de energía:

- Regulador interno de baja caída
- Dominio de poder individual para RTC
- Corriente de 5μA en modo de suspensión profundo
- Despierta por interrupción de GPIO, temporizador, medidas de ADC, interrupción por sensor de tacto capacitivo

4.2. IoT (Internet de las cosas)

El «Internet de las Cosas» (IoT) hace referencia, a una tecnología basada en la conexión de objetos cotidianos a Internet que intercambian, agregan y procesan información sobre su entorno físico para proporcionar servicios de valor añadido a los usuarios finales. También reconoce eventos o cambios, y tales sistemas pueden reaccionar de forma autónoma y adecuada. Su finalidad es, por tanto, brindar una infraestructura que supere la barrera entre los objetos en el mundo físico y su representación en los sistemas de información. Esta integración de sensores y dispositivos en objetos cotidianos que quedan conectados a Internet a través de redes alámbricas e inalámbricas ha alumbrado, un nuevo modo de interacción en el mundo físico, inspirado en la idea de ubicuidad y facilitado por el desarrollo de las TIC y la industria electrónica. Se crea así una malla de conexiones en el planeta que establecería una suerte de «sistema nervioso mundial», donde la aldea global alcanzará a los objetos cotidianos (Barrio Andrés, 2018, p. 19-20).

Por lo anterior se puede tomar una idea de que casi cualquier cosa de uso cotidiano o electrónico que esté conectado a internet ya es parte de lo que es internet de las cosas. Ranger, (2020) afirma que desde una bombilla que se puede encender con una aplicación de teléfono inteligente es un dispositivo de IoT, al igual que un sensor de movimiento o un termostato inteligente en su oficina o una farola conectada. Un dispositivo de IoT podría ser tan esponjoso como un juguete para niños o tan serio como un camión sin conductor. Algunos objetos más grandes pueden estar llenos de muchos componentes de IoT más pequeños, como un motor a reacción que ahora está lleno de miles de sensores que recopilan y transmiten datos para asegurarse de que esté funcionando de manera eficiente. A una escala aún mayor, los proyectos de ciudades inteligentes están llenando regiones enteras con sensores para ayudarnos a comprender y controlar el medio ambiente.

4.2.1. Tamaño del IoT

La empresa de analistas tecnológicos IDC predice que en total habrá 41.600 millones de dispositivos IoT conectados para 2025, o "cosas". También sugiere que los equipos industriales y automotrices representan la mayor oportunidad de "cosas" conectadas, pero

también prevé una fuerte adopción de dispositivos portátiles y domésticos inteligentes a corto plazo.

Otro analista tecnológico, Gartner, predice que los sectores empresarial y automotriz representarán 5.800 millones de dispositivos este año, casi un cuarto más que en 2019. Las empresas de servicios públicos serán el mayor usuario de IoT, gracias al continuo despliegue de medidores inteligentes. Los dispositivos de seguridad, en forma de detección de intrusos y cámaras web, serán el segundo uso más importante de los dispositivos de IoT. La automatización de edificios, como la iluminación conectada, será el sector de más rápido crecimiento, seguido de la automoción (automóviles conectados) y la atención médica (monitoreo de condiciones crónicas).

IoT Endpoint Market by Segment, 2018-2020, Worldwide (Installed Base, Billions of Units)

Segment	2018	2019	2020
Utilities	0.98	1.17	1.37
Government	0.40	0.53	0.70
Building Automation	0.23	0.31	0.44
Physical Security	0.83	0.95	1.09
Manufacturing & Natural Resources	0.33	0.40	0.49
Automotive	0.27	0.36	0.47
Healthcare Providers	0.21	0.28	0.36
Retail & Wholesale Trade	0.29	0.36	0.44
Information	0.37	0.37	0.37
Transportation	0.06	0.07	0.08
Total	**3.96**	**4.81**	**5.81**

Source: Gartner (August 2019)

Tabla 4.1, *Principal mercado inversor en IoT (2018-2020)*

4.2.2. Beneficios de IoT para la industria

Los beneficios de IoT para las empresas dependen de la implementación particular; la agilidad y la eficiencia suelen ser las principales consideraciones. La idea es que las empresas tengan acceso a más datos sobre sus propios productos y sus propios sistemas internos, y como resultado, una mayor capacidad para realizar cambios.

Los fabricantes están agregando sensores a los componentes de sus productos para que puedan transmitir datos sobre su desempeño. Esto puede ayudar a las empresas a detectar cuándo es probable que falle un componente y a cambiarlo antes de que cause daños. Las empresas también pueden usar los datos generados por estos sensores para hacer que sus sistemas y sus cadenas de suministro sean más eficientes, porque tendrán datos mucho más precisos sobre lo que realmente está sucediendo.

El uso empresarial de IoT se puede dividir en dos segmentos: ofertas específicas de la industria, como sensores en una planta generadora o dispositivos de ubicación en tiempo real para el cuidado de la salud; y dispositivos de IoT que se pueden utilizar en todas las industrias, como aire acondicionado inteligente o sistemas de seguridad (Ranger, 2020).

4.3. Protocolos para IoT

Los protocolos de IoT son una parte crucial de la pila de tecnología de IoT; sin ellos, el hardware se volvería inútil ya que los protocolos de IoT le permiten intercambiar datos de una manera estructurada y significativa. De estos datos transferidos, se puede extraer información útil para el usuario final y, gracias a ello, toda la implementación se vuelve económicamente rentable, especialmente en términos de gestión de dispositivos IoT.

Cuando hablamos de Internet de las cosas, siempre pensamos en la comunicación. La interacción entre sensores, dispositivos, puertas de enlace, servidores y aplicaciones de usuario es la característica esencial que hace que Internet de las cosas sea lo que es. Pero lo que permite que todas estas cosas inteligentes hablen e interactúen son los protocolos de IoT, que pueden verse como lenguajes que utiliza el equipo de IoT para comunicarse.

4.3.1. Protocolo de aplicación restringida (CoAP)

Si bien la infraestructura de Internet existente está disponible gratuitamente y se puede utilizar para cualquier dispositivo de IoT, a menudo resulta demasiado pesada y consume mucha energía para la mayoría de los casos de uso de IoT. Creado por el grupo de trabajo de Entornos RESTful Restringidos de IETF y lanzado en 2013, el Protocolo de Aplicación Restringida (CoAP) fue diseñado para traducir el modelo HTTP de modo que pudiera usarse en entornos de red y dispositivos restrictivos.

Diseñado para abordar las necesidades de los sistemas de IoT basados en HTTP, CoAP se basa en el Protocolo de datagramas de usuario (UDP) para establecer una comunicación segura entre los puntos finales. Al permitir la transmisión y la multidifusión, UDP puede transmitir datos a múltiples hosts mientras conserva la velocidad de comunicación y el uso de ancho de banda bajo, lo que lo convierte en una buena combinación para las redes inalámbricas que se emplean normalmente en entornos M2M con recursos limitados. Otra cosa que CoAP comparte con HTTP es la arquitectura RESTful que admite un modelo de interacción de solicitud / respuesta entre los puntos finales de la aplicación. Además, CoAP adopta los métodos básicos HTTP get, post, put y delete, gracias a los cuales se puede evitar la ambigüedad en el momento de la interacción entre clientes.

4.3.2. MQTT (Message Queue Telemetry Transport)

Este protocolo de transporte por telemetría MQ es el que se usara para este proyecto fase 1, siendo un estándar ampliamente adoptado en el IoT o internet de las cosas hasta la fecha, como Bernstein, (2021) nos explica, MQTT (MQ Telemetry Transport) es un protocolo de mensajería abierto y liviano que brinda a los clientes de red con recursos limitados una forma simple de distribuir información de telemetría en entornos de bajo ancho de banda. El protocolo, que emplea un patrón de comunicación de publicación/suscripción, se utiliza para la comunicación de máquina a máquina (M2M). Esto quiere decir que no es necesaria la interacción humana ya que debido a su tipo de patrón de comunicación los datos se enviaran cada vez que se necesiten, esto se programa según la necesidad o aplicación a cubrir.

MQTT se basa en el modelo de suscriptor, editor y broker. Dentro del modelo, la tarea del editor es recopilar los datos y enviar información a los suscriptores a través de la capa de mediación, que es el broker. El papel del broker, por otro lado, es garantizar la seguridad mediante la verificación cruzada de la autorización de los editores y suscriptores. MQTT ofrece tres modos de conseguirlo (Quality of Service), gracias a los cuales el editor tiene la posibilidad de definir la calidad de su mensaje:

- QoS0 (como máximo una vez): el modo menos confiable pero también el más rápido. La publicación se envía, pero no se recibe confirmación.

- QoS1 (al menos una vez): garantiza que el mensaje se entregue al menos una vez, pero se pueden recibir duplicados.
- QoS2 (exactamente una vez): el modo más confiable y el que consume más ancho de banda. Los duplicados se controlan para garantizar que el mensaje se entregue solo una vez.

Habiendo encontrado una amplia aplicación en dispositivos IoT como medidores eléctricos, vehículos, detectores y equipos industriales o sanitarios, MQTT responde bien a las siguientes necesidades:

- Uso mínimo de ancho de banda
- Operación a través de redes inalámbricas
- Bajo consumo de energía
- Buena fiabilidad si es necesario
- Escasos recursos de memoria y procesamiento

A pesar de sus características, MQTT puede resultar problemático para algunos dispositivos muy restrictivos, debido al hecho de la transmisión de mensajes sobre TCP y la gestión de nombres largos de temas. Esto se resuelve con la variante MQTT-SN que usa UDP y admite la indexación de nombres de temas. Sin embargo, a pesar de su amplia adopción, MQTT no admite un modelo bien definido de representación de datos y estructura de administración de dispositivos, lo que hace que la implementación de sus capacidades de administración de datos y administración de dispositivos sea completamente específica de la plataforma o del proveedor (AVSystem, 2019).

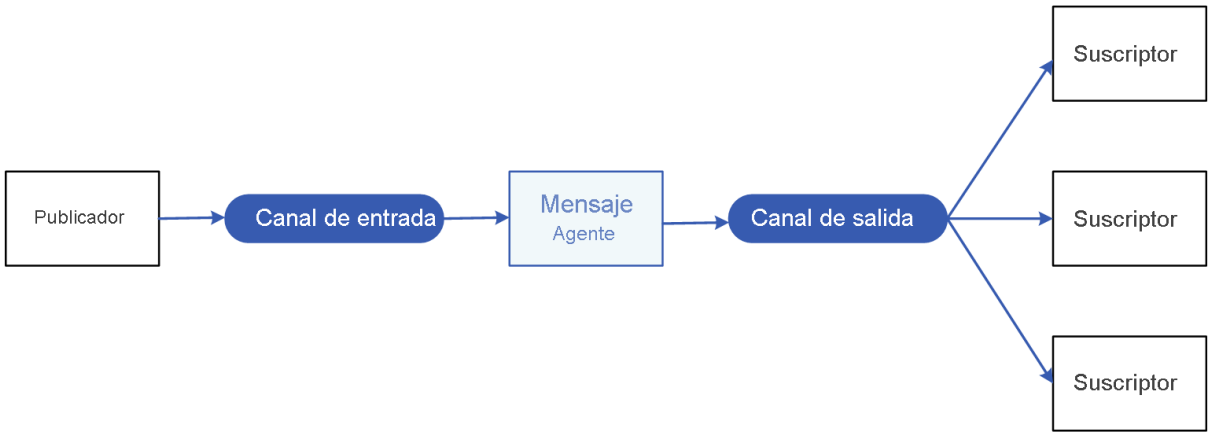


Figura 4.2 Esquema de comunicación Pub/Sub

4.3.3. Wifi

La creación de una red Wi-Fi requiere dispositivos que puedan enviar señales inalámbricas, es decir, dispositivos como teléfonos, computadoras o enrutadores, por nombrar algunos. En casa, se utiliza un enrutador para transferir la conexión a Internet desde una red pública a una red privada de casa u oficina. Wifi proporciona una conexión a Internet a dispositivos cercanos que se encuentran dentro de un cierto rango. Otra forma de utilizar Wifi es crear un punto de acceso Wifi, es decir, los teléfonos o las computadoras pueden compartir una conexión a Internet inalámbrica o por cable con otros dispositivos mediante la transmisión de una señal.

Wifi utiliza ondas de radio que transmiten información en frecuencias específicas, como canales de 2,4 GHz o 5 GHz. Ambos rangos de frecuencia tienen una serie de canales a través de los cuales pueden trabajar diferentes dispositivos inalámbricos, lo que ayuda a distribuir la carga para que las conexiones individuales de los dispositivos no se interrumpan. Esto evita en gran medida el desbordamiento de las redes inalámbricas.

4.3.4. ZigBee

Las redes basadas en ZigBee se caracterizan por un bajo consumo de energía, bajos rendimientos (hasta 250 kbps) y un rango de conectividad de 100 metros entre nodos. Las aplicaciones típicas incluyen redes de sensores, redes personales (WPAN), domótica, sistemas de alarma y sistemas de monitoreo.

ZigBee fue desarrollado como un estándar para redes de radio de corto alcance auto configurables, destinadas a su uso en sistemas de telemetría, para la comunicación entre varios tipos de sensores, dispositivos de monitoreo, así como para la lectura inalámbrica de resultados de medición de medidores de energía y calor. etc. El estándar ZigBee es un protocolo de intercambio de paquetes de datos relativamente simple, resistente a errores de comunicación y lecturas no autorizadas, que a menudo se implementa en dispositivos con requisitos relativamente pequeños, como microcontroladores, sensores, etc.

4.3.5. Bluetooth

Bluetooth es una tecnología que permite la conexión inalámbrica de varios dispositivos electrónicos, como un teléfono, teclado, computadora, computadora portátil, mouse, computadora de mano, impresora, auricular o altavoz, y más. Si desea una definición más similar a una wiki, este es un estándar abierto descrito en la especificación IEEE 802.15.1 y su especificación técnica incluye tres clases de potencia de transmisión ERP 1-3 con un rango de, respectivamente, 100, 10 y 1 metro de espacio abierto. La clase más común es la segunda (10 m) que le permite conectar dispositivos que se encuentran en diferentes habitaciones e incluso en diferentes pisos.

El estándar utiliza ondas de radio en la banda de frecuencia ISM de 2,4 GHz y el dispositivo que permite el uso de este estándar es un adaptador Bluetooth.

4.4. Transporte por Telemetría MQTT

Figura 4.3 Logo del protocolo MQTT

4.4.1. El protocolo MQTT

El protocolo MQTT es el estándar de facto para la mensajería de IoT. Estandarizado por OASIS e ISO, el protocolo de publicación / suscripción MQTT proporciona una forma escalable y confiable de conectar dispositivos a través de Internet. Hoy en día, muchas empresas utilizan MQTT para conectar millones de dispositivos a Internet (HiveMQ, 2022).

En esta sección de este proyecto se utilizara la información brindada por uno de los principales brokers y plataformas del protocolo MQTT el cual es HiveMQ que desde su sitio web nos brinda acceso a todo lo necesario para conocer y aprender cómo funciona este protocolo, también el libro de texto “MQTT & MQTT 5 Essentials A comprehensive overview of MQTT facts and features for beginners and experts alike" que es un libro y manual escrito por la empresa HiveMQ que está enfocado desde principiantes hasta expertos.

La mayoría de empresas hoy en día usan el protocolo MQTT para enviar información a través de múltiples dispositivos en la red, Facebook Messenger actualmente implementa este protocolo para su aplicación de mensajería, y esto se debe a que tiene muchas ventajas que permiten usar el protocolo en casi cualquier dispositivo, algunas son:

- Requiere recursos mínimos ya que es liviano y eficiente
- Admite mensajería bidireccional entre el dispositivo y la nube
- Puede escalar a millones de dispositivos conectados
- Funciona bien en redes poco fiables
- Seguridad habilitada, por lo que funciona con TLS y protocolos de autenticación comunes

4.4.2. Características clave de MQTT

Clientes MQTT: Los clientes MQTT publican un mensaje para un intermediario MQTT y otros clientes MQTT se suscriben a los mensajes que desean recibir. Las implementaciones de clientes MQTT generalmente requieren un espacio mínimo, por lo que son adecuadas para la implementación en dispositivos pequeños restringidos y son muy eficientes en sus requisitos de ancho de banda.

Broker MQTT: Los intermediarios MQTT reciben mensajes publicados y envían el mensaje a los clientes MQTT que se suscriben. Un mensaje MQTT contiene un tema de mensaje al que los clientes MQTT se suscriben y los intermediarios MQTT utilizan estas listas de suscripción para determinar los clientes MQTT que recibirán el mensaje.

Calidad de servicio: MQTT implementa 3 niveles de calidad de servicio para el acuerdo entre el remitente y el receptor: 1) Como máximo una vez (0), 2) Al menos una vez (1) y 3) Exactamente una vez (2). Estos niveles de QoS permiten aplicaciones de IoT más confiables desde la infraestructura de mensajería subyacente y se adaptan a condiciones de red no confiables.

Sesiones persistentes: MQTT permite una sesión persistente entre el cliente y el broker. Esto permite que las sesiones persistan incluso si la red está desconectada. Una vez que se vuelve a conectar la red, la información para volver a conectar al cliente con el broker aún existe. Esta es una de las características clave que hace que el protocolo MQTT sea más eficiente que HTTP para su uso en redes celulares poco confiables.

Mensajes retenidos: Los clientes de MQTT que se suscriben a un nuevo tema no saben cuándo esperar el primer mensaje que recibirán. Sin embargo, un broker de MQTT puede almacenar un mensaje retenido que puede enviarse inmediatamente después de una nueva suscripción a MQTT.

Última voluntad y testamento: Un cliente MQTT puede especificarle a un broker MQTT un mensaje, llamado último testamento, que se enviará si el cliente MQTT se desconecta sin gracia. Esto permite una notificación más elegante en todo el sistema de que un cliente se ha desconectado.

4.4.3. Funcionamiento: Conexión, Publicación/Suscripción y desconexión

El patrón de publicación/suscripción (también conocido como pub/sub) proporciona una alternativa a la arquitectura cliente-servidor tradicional. En el modelo cliente-servidor, un cliente se comunica directamente con un punto final. El modelo pub / sub desacopla al cliente que envía un mensaje (el editor) del cliente o clientes que reciben los mensajes (los suscriptores. Los editores y suscriptores nunca se contactan directamente entre sí. De hecho, ni siquiera son conscientes de que existe el otro. La conexión entre ellos es manejada por un tercer componente (el broker, en este caso CloudMQTT que estaré explicando en la siguiente sección de este proyecto). El trabajo del broker es filtrar todos los mensajes entrantes y distribuirlos correctamente a los suscriptores (HiveMQ, 2020, p. 7).

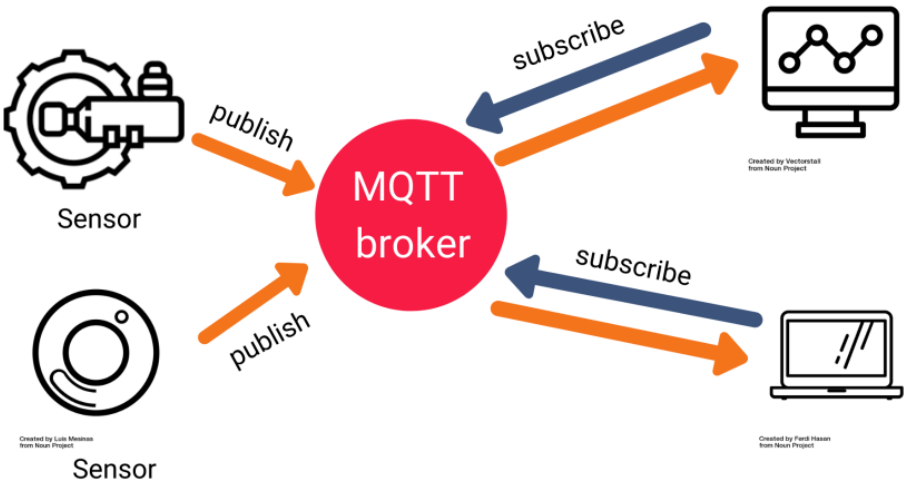


Figura 4.4 Esquema de envió de datos por suscripción y publicación.

El aspecto más importante de pub / sub es el desacoplamiento del editor del mensaje del destinatario (suscriptor). Este desacoplamiento tiene varias dimensiones:

- **Desacoplamiento de espacio:** el editor y el suscriptor no necesitan conocerse (por ejemplo, no hay intercambio de dirección IP y puerto).
- **Desacoplamiento de tiempo:** el editor y el suscriptor no necesitan ejecutarse al mismo tiempo.

- **Desacoplamiento de sincronización:** no es necesario interrumpir las operaciones en ambos componentes durante la publicación o la recepción.

En resumen, el modelo pub/sub elimina la comunicación directa entre el editor del mensaje y el destinatario / suscriptor. La actividad de filtrado del broker permite controlar qué cliente / suscriptor recibe qué mensaje. El desacoplamiento tiene tres dimensiones: espacio, tiempo y sincronización.

4.4.4. Cliente MQTT

Tanto los editores como los suscriptores son clientes de MQTT. Las etiquetas de editor y suscriptor se refieren a si el cliente está publicando mensajes actualmente o suscrito para recibir mensajes (la funcionalidad de publicación y suscripción también se puede implementar en el mismo cliente MQTT). Un cliente MQTT es cualquier dispositivo (desde un microcontrolador hasta un servidor completo) que ejecuta una biblioteca MQTT y se conecta a un agente MQTT a través de una red. Por ejemplo, el cliente MQTT puede ser un dispositivo muy pequeño con recursos limitados que se conecta a través de una red inalámbrica y tiene una biblioteca mínima. El cliente MQTT también puede ser una computadora típica que ejecuta un cliente MQTT gráfico con fines de prueba. Básicamente, cualquier dispositivo que habla MQTT sobre una pila TCP / IP puede llamarse cliente MQTT. La implementación del cliente del protocolo MQTT es muy sencilla y optimizada. La facilidad de implementación es una de las razones por las que MQTT es ideal para dispositivos pequeños. Las bibliotecas cliente MQTT están disponibles para una gran variedad de lenguajes de programación. Por ejemplo, Android, Arduino, C, C ++, C #, Go, iOS, Java, JavaScript y .NET.

En este proyecto se utiliza el lenguaje de programación .IO de Arduino, pero potenciado con el editor de código Atom con el complemento de PlatformIO que incorpora las herramientas necesarias para poder implementar dispositivos al internet de las cosas.

4.4.5. Broker

El broker o en español intermediario será el encargado de lo que se transmite y recolecta hacia el cliente, como mencione anteriormente para este proyecto se implementara el broker Cloud MQTT debido a su bajo precio de suscripción, o de otra más técnica como nos comenta HiveMQ, (2020) la contraparte del cliente MQTT es el broker MQTT. El broker es el núcleo de cualquier protocolo de publicación / suscripción. Dependiendo de la implementación, un broker puede manejar hasta millones de clientes MQTT conectados simultáneamente (p. 11).

El broker es responsable de recibir todos los mensajes, filtrar los mensajes, determinar quién está suscrito a cada mensaje y enviar el mensaje a estos clientes suscritos. El broker también guarda los datos de sesión de todos los clientes que tienen sesiones persistentes, incluidas las suscripciones y los mensajes perdidos. Otra responsabilidad del broker es la autenticación y autorización de los clientes. Por lo general, el intermediario es extensible, lo que facilita la autenticación, la autorización y la integración personalizadas en los sistemas backend. La integración es particularmente importante porque el intermediario es con frecuencia el componente que está expuesto directamente en Internet, maneja muchos clientes y necesita pasar mensajes a los sistemas de análisis y procesamiento posteriores. En resumen, el broker es el eje central a través del cual debe pasar cada mensaje. Por lo tanto, es importante que el broker sea altamente escalable, integrable en sistemas backend, fácil de monitorear y (por supuesto) resistente a fallas.

4.5. Conexión a MQTT

El protocolo MQTT se basa en TCP / IP. Tanto el cliente como el intermediario deben tener una pila de TCP / IP.

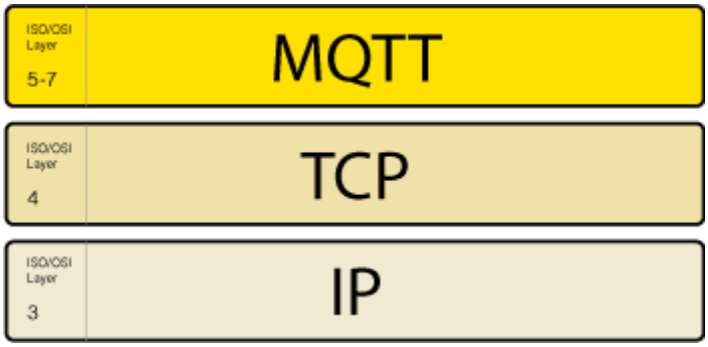


Figura 4.5 Capa de protocolos en dispositivos MQTT

La conexión MQTT siempre es entre un cliente y el intermediario. Los clientes nunca se conectan entre sí directamente. Para iniciar una conexión, el cliente envía un mensaje CONNECT al broker. El broker responde con un mensaje CONNACK y un código de estado. Una vez que se establece la conexión, el broker la mantiene abierta hasta que el cliente envía un comando de desconexión o se interrumpe la conexión.

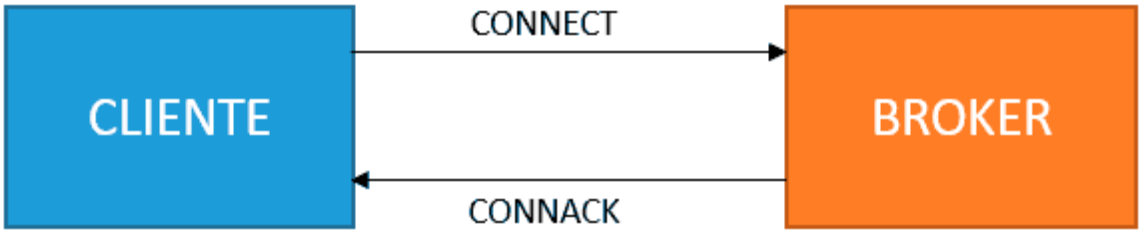


Figura 4.6 Representación del inicio de conexión a MQTT.

4.5.1. Conexión MQTT a través de un NAT

En muchos casos de uso común, el cliente MQTT está ubicado detrás de un enrutador que usa traducción de direcciones de red (NAT) para traducir de una dirección de red privada (como 192.168.xx, 10.0.xx) a una dirección pública. Como ya mencionamos, el cliente MQTT inicia la conexión enviando un mensaje CONNECT al broker. Debido a que el broker tiene una dirección pública y mantiene la conexión abierta para permitir el envío y la

recepción bidireccional de mensajes (después del CONNECT inicial), no hay ningún problema con los clientes que se encuentran detrás de un NAT.

4.5.2. Inicio de conexión

Ahora veamos el mensaje del comando MQTT CONNECT. Para iniciar una conexión, el cliente envía un mensaje de comando al broker. Si este mensaje CONNECT tiene un formato incorrecto (de acuerdo con la especificación MQTT) o pasa demasiado tiempo entre la apertura de una toma de red y el envío del mensaje de conexión, el broker cierra la conexión. Este comportamiento disuade a los clientes malintencionados que pueden ralentizar al broker. Un cliente MQTT envía un mensaje de conexión con el siguiente contenido:

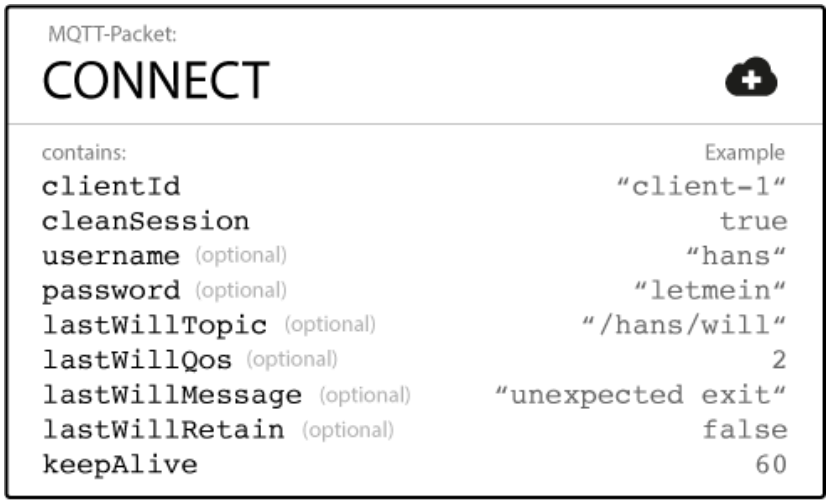


Figura 4.7 Credenciales enviadas del Cliente al Broker.

Alguna información incluida en un mensaje CONNECT es probablemente más interesante para los implementadores de una biblioteca MQTT que para los usuarios de esa biblioteca.

4.5.3. Identificación del cliente

El identificador de cliente (ClientId) identifica a cada cliente MQTT que se conecta a un broker MQTT. El broker usa el ClientId para identificar al cliente y el estado actual del cliente, por lo tanto, este ID debe ser único por cliente y broker.

4.5.4. Sesión limpia

El indicador de sesión limpia le dice al broker si el cliente quiere establecer una sesión persistente o no. En una sesión persistente (CleanSession = false), el broker almacena todas las suscripciones para el cliente y todos los mensajes perdidos para el cliente que se suscribió con un nivel de calidad de servicio (QoS) 1 o 2. Si la sesión no es persistente (CleanSession = true), el broker no almacena nada para el cliente y purga toda la información de cualquier sesión persistente anterior.

4.5.5. Usuario Contraseña

MQTT puede enviar un nombre de usuario y una contraseña para la autenticación y autorización del cliente. Sin embargo, si esta información no está cifrada o codificada (ya sea por implementación o TLS), la contraseña se envía en texto sin formato.

4.5.6. Mensaje de voluntad

El mensaje de última voluntad es parte de la función Última voluntad y testamento (LWT) de MQTT. Este mensaje notifica a otros clientes cuando un cliente se desconecta sin gracia. Cuando un cliente se conecta, puede proporcionar al broker una última voluntad en forma de un mensaje MQTT y un tema dentro del mensaje CONNECT. Si el cliente se desconecta sin gracia, el broker envía el mensaje LWT en nombre del cliente.

4.5.7. KeepAlive

KeepAlive es un intervalo de tiempo en segundos que el cliente especifica y comunica al broker cuando se establece la conexión. Este intervalo define el período de tiempo más largo que el broker y el cliente pueden soportar sin enviar un mensaje. El cliente se compromete a enviar mensajes de solicitud PING regulares al broker. El broker responde con una respuesta PING. Este método permite que ambos lados determinen si el otro todavía está disponible.

Básicamente, esa es toda la información que es todo lo que necesita para conectarse a un bróker MQTT desde un cliente MQTT

4.5.8. Conectar código de retorno

La segunda bandera en el mensaje CONNACK es la bandera de reconocimiento de conexión. Esta bandera contiene un código de retorno que le dice al cliente si el intento de conexión fue exitoso o no.

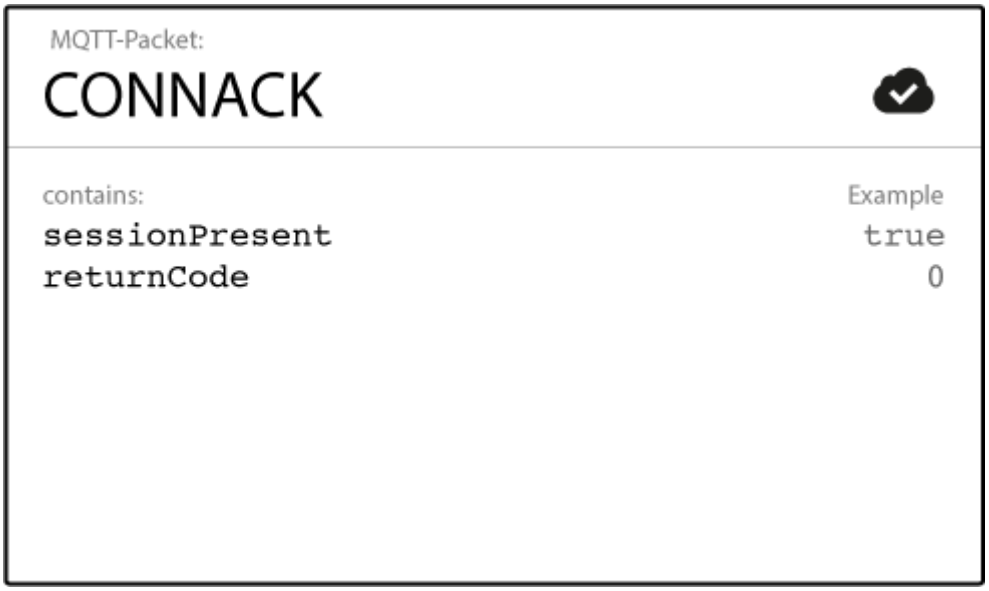


Figura 4.8 Respuesta de conexión del broker al cliente

Código de retorno	Respuesta de código de retorno
0	Conexión aceptada
1	Conexión rechazada, versión de protocolo inaceptable
2	Conexión rechazada, identificador rechazado
*3	Conexión rechazada, servidor no disponible

Código de retorno	Respuesta de código de retorno
4	Conexión rechazada, nombre de usuario o contraseña incorrectos
5	Conexión rechazada, no autorizada

Tabla 4.2, *Posibles respuestas del broker y su significado.*

4.6. Publicar y suscribirse a un tópico

4.6.1. Publicar

Un cliente MQTT puede publicar mensajes tan pronto como se conecte a un intermediario. Cada mensaje debe contener un tema que el broker pueda usar para reenviar el mensaje a los clientes interesados. Normalmente, cada mensaje tiene una carga útil que contiene los datos a transmitir en formato de bytes. MQTT es independiente de los datos. El caso de uso del cliente determina cómo se estructura la carga útil. El cliente remitente (editor) decide si desea enviar datos binarios, datos de texto o incluso XML o JSON completo.

Un mensaje PUBLICAR en MQTT tiene varios atributos que queremos discutir en detalle:

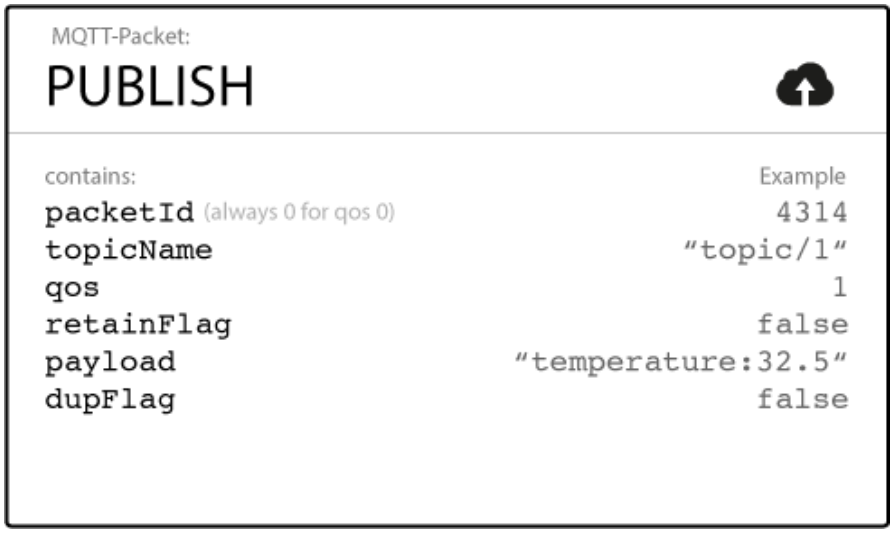


Figura 4.9 Identificadores principales contenidos en un mensaje publicado.

Tópico: El tópico nombre del tema es una cadena simple que está estructurada jerárquicamente con barras diagonales como delimitadores. Por exempla, “myhome / livingroom / temperature” o “Germany / Munich / Octoberfest / people”.

QoS: Este número indica el nivel de calidad de servicio (QoS) del mensaje. Hay tres niveles: 0, 1 y 2. El nivel de servicio determina qué tipo de garantía tiene un mensaje para llegar al destinatario previsto (cliente o broker).

Indicador de retención: Este indicador define si el intermediario guarda el mensaje como el último valor válido conocido para un tema específico. Cuando un nuevo cliente se suscribe a un tema, recibe el último mensaje que se conserva sobre ese tema.

Carga útil (Payload): Este es el contenido real del mensaje. MQTT es independiente de los datos. Es posible enviar imágenes, texto en cualquier codificación, datos cifrados y prácticamente todos los datos en binario.

Identificador de paquete: El identificador de paquete identifica de forma única un mensaje a medida que fluye entre el cliente y el intermediario. El identificador de paquete solo es relevante para niveles de QoS mayores que cero. La biblioteca cliente y / o el intermediario son responsables de establecer este identificador MQTT interno.

Indicador DUP: El indicador indica que el mensaje es un duplicado y se reenvió porque el destinatario previsto (cliente o broker) no reconoció el mensaje original. Esto solo es relevante para QoS mayor que 0. Por lo general, el mecanismo de reenvío / duplicación lo maneja la biblioteca cliente MQTT o el intermediario como un detalle de implementación.

4.6.2. Suscribir

Publicar un mensaje no tiene sentido si nadie lo recibe nunca. Es decir, si no hay clientes para suscribirse a los temas de los mensajes. Para recibir mensajes sobre temas de interés, el cliente envía un mensaje subscribe al broker de MQTT. Este mensaje de suscripción es muy simple, contiene un identificador de paquete único y una lista de suscripciones.

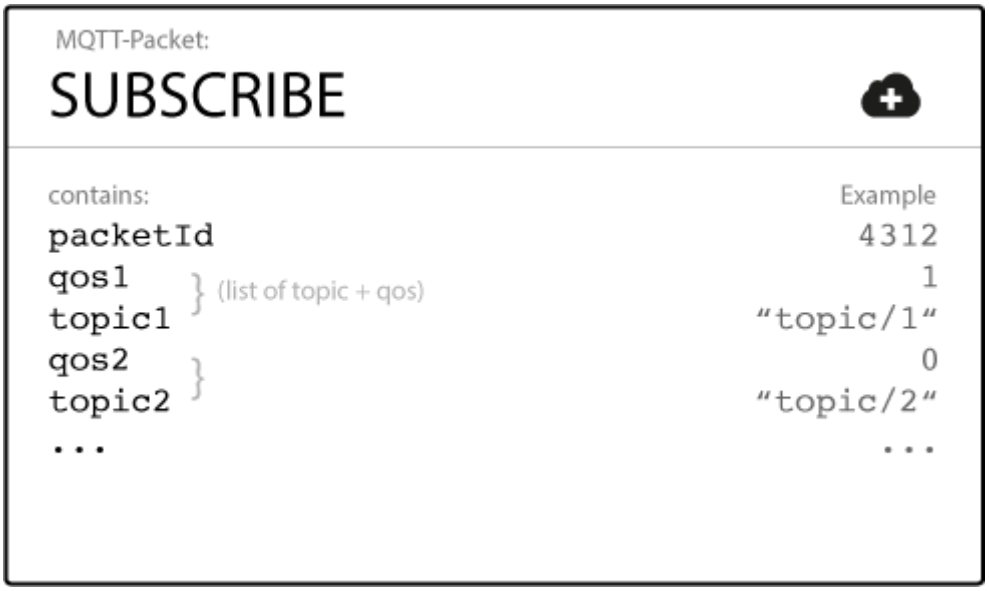


Figura 4.10 Identificadores principales recibidos al suscribirse a un tópico.

Identificador de paquete: El identificador de paquete identifica de forma única un mensaje a medida que fluye entre el cliente y el intermediario. La biblioteca cliente y / o el intermediario son responsables de establecer este identificador MQTT interno.

Lista de suscripciones: Un mensaje SUBSCRIBE puede contener varias suscripciones para un cliente. Cada suscripción se compone de un tema y un nivel de QoS. El tema del mensaje de suscripción puede contener comodines que permiten suscribirse a un patrón de tema en lugar de a un tema específico. Si hay suscripciones superpuestas para un cliente, el broker entrega el mensaje que tiene el nivel más alto de QoS para ese tema.

4.7. Cloud MQTT

CloudMQTT son servidores Mosquitto administrados en la nube. Mosquitto implementa el protocolo de transporte de telemetría MQ, MQTT, que proporciona métodos ligeros para llevar a cabo la mensajería utilizando un modelo de cola de mensajes de publicación / suscripción (CloudMQTT, 2022).

4.7.1 Broker Cloud MQTT

Cloud MQTT es el broker que estamos usando para este tópico/aplicación por su sencilla interfaz y por tener un precio razonable de 5 USD por mes que nos permite tener hasta 25 dispositivos conectados a él, los cuales pueden ser suscriptores o publicadores. En el siguiente capitulo estaré configurando paso a paso esto desde el registro hasta su puesta en marcha.

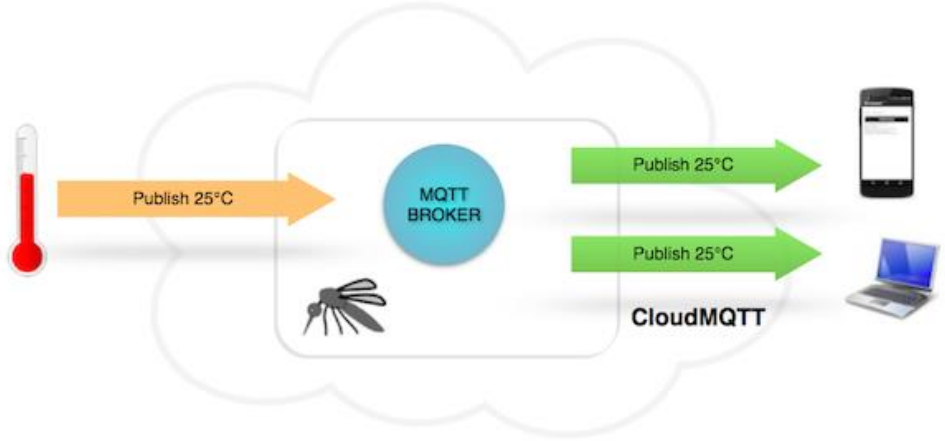


Figura 4.11 Esquema de función del broker Cloud MQTT basado en mosquitto.

4.8. IoT: ESP32 usando el protocolo MQTT

En esta sección se configura la placa ESP32 y también se explica la creación de un usuario en CloudMQTT para poder poner en marcha el entorno para la recolección de datos.

4.8.1. Preparación del entorno virtual

Lo primero a realizar es el ingreso a la página de CloudMQTT, una vez completado el registro se procede a la selección de un plan de pago, para tener una menor latencia de comunicación se debe seleccionar el servidor más cercano a la ubicación geográfica donde se utilizará el sistema a montar. Para este caso se seleccionó Virginia del Norte como servidor, este manejado por Amazon Web Services.

4.8.1.1. Registro en MQTT

Una vez que se ha completado el registro en la página se procede a crear la instancia necesaria acorde al plan, en este caso se pagó 5$ por el tiempo de un mes, de este apartado se obtienen las siguientes credenciales que servirán para poder conectar la placa de desarrollo a internet mediante el broker.

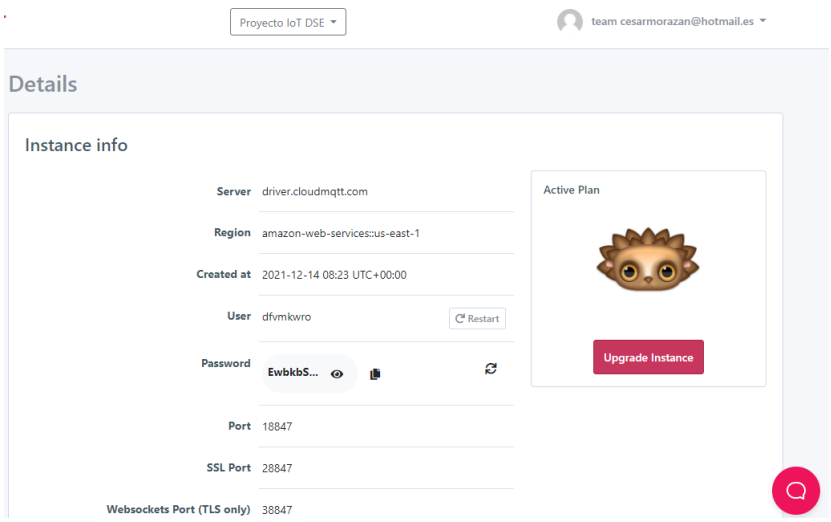


Figura 4.12 Credenciales obtenidas por pago de servicio.

Aparte de haber obtenido los datos necesarios para conectar distintos dispositivos también se creó la instancia donde se muestran los clientes conectados, información de pago, Websocket entre otros.

4.8.2. Preparación del ESP32

En este apartado se muestra la configuración del ESP32 junto con su programación para poder enviar los primeros datos al broker.

4.8.2.1. Librería PubSubClient

Esta biblioteca permite enviar y recibir mensajes MQTT. Es compatible con el protocolo MQTT 3.1.1 más reciente y se puede configurar para usar el MQTT 3.1 anterior si es necesario. Es compatible con todo el hardware compatible con Arduino Ethernet Client, incluidos Intel Galileo / Edison, ESP8266 y TI CC3000.

Creador de la librería: Nick O'Leary

Los pasos para instalarlos son sencillos, en el editor de texto que se esté usando ir a herramientas y luego a librerías, en este caso PlatformIO, en el inicio se muestra el apartado de librerías, basta con buscar PubSubClient y se mostrara de primero. Como se muestra en la siguiente figura solo basta con dar clic y luego dar en instalar, con esto se obtendrán los comandos internos necesario para la comunicación con MQTT

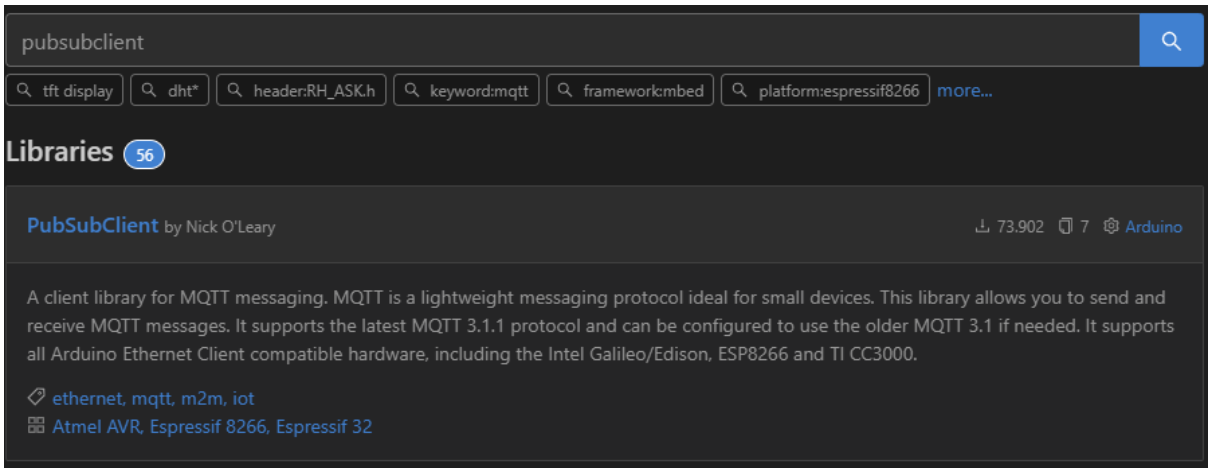


Figura 4.13 Librería PubSubClient necesaria para MQTT.

4.8.2.2. Creación de estancias para conexión local (Wifi)

Una vez realizado lo anterior se procede con la programación del código, se encontrará en anexos. Como se aprecia en la siguiente imagen la programación inicial y necesaria para la conexión con el broker depende de los datos que se asignen en el broker MQTT una vez suscritos y de la conexión a internet a usar.

```
#include <Arduino.h>
#include <Wifi.h>
#include <PubSubClient.h>//libreria para uso del mqtt

const char* ssid = "Insane"; // Nombre de red(router)
const char* password = "VirusG15030*"; //contraseña

//Credenciales del servidor MQTT
const char *mqtt_server = "driver.cloudmqtt.com";
const int   mqtt_port = 18847;
const char *mqtt_user = "dfvmkwro";
const char *mqtt_pass = "EwbkbS2Dd";
```

Figura 4.14 Introducción de credenciales obtenidas para utilizar el dispositivo ESP32.

Las librerías inicializadas son para poder usar la programación base de Arduino con ESP32, la de conexión mediante wifi de la placa ESP32 y la librería PubSubClient que se explicó anteriormente.

4.8.2.3. Testeo de conexión local

Una vez que se ha realizado toda la programación en el sistema se procede a realizar la comprobación de que se conecta localmente el ESP32 a la red wifi.

Como se puede observar el ESP32 realizo la conexión correctamente a la red wifi local y está enviando un mensaje programado vacío solo para comprobar de que si está bien su configuración.

```
Conectando a...
Insane
.....
Conectado a red WiFi!
Dirección IP:
192.168.0.[redacted]
Intentando Conexión MQTT
Conectado a MQTT
Mensaje enviado -> 1
```

Figura 4.15 Conexión exitosa a red local.

La información, de la figura anterior es mostrada mediante el monitor serial del sistema. Cada punto después del nombre del router (Insane) representa medio segundo de espera para conectarse con los servidores mosquitto del broker

4.8.2.4. Callback y conexión de ESP32 a Cloud MQTT

Es necesario programar una función Callback la cual estará recibiendo datos y mostrando información de los suscriptores y publicadores sin tener que preguntar si desea o no.

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Mensaje recibido bajo el tópico -> ");
  Serial.print(topic);
  Serial.print("\n");

  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
    Serial.println();
}
```

Figura 4.16 Función que permitirá la comunicación y recepción de datos.

Como se aprecia la función recibe 3 parámetros los cuales son el tópico o tema con el que se envía, el Payload o carga útil que es el dato que se enviara y por último un entero que servirá para realizar una condición para que se muestren los datos enviados mediante una escritura por el monitor serial.

4.8.3. Testeo de conexión al bróker

Una vez todo está programado se procede a revisar si el broker ya tiene establecida la comunicación con la placa de desarrollo.

Se logra apreciar en la figura 4.17 que la conexión se realizó de manera exitosa y está todo listo para comenzar a hacer pruebas de envíos y recolección de dato.

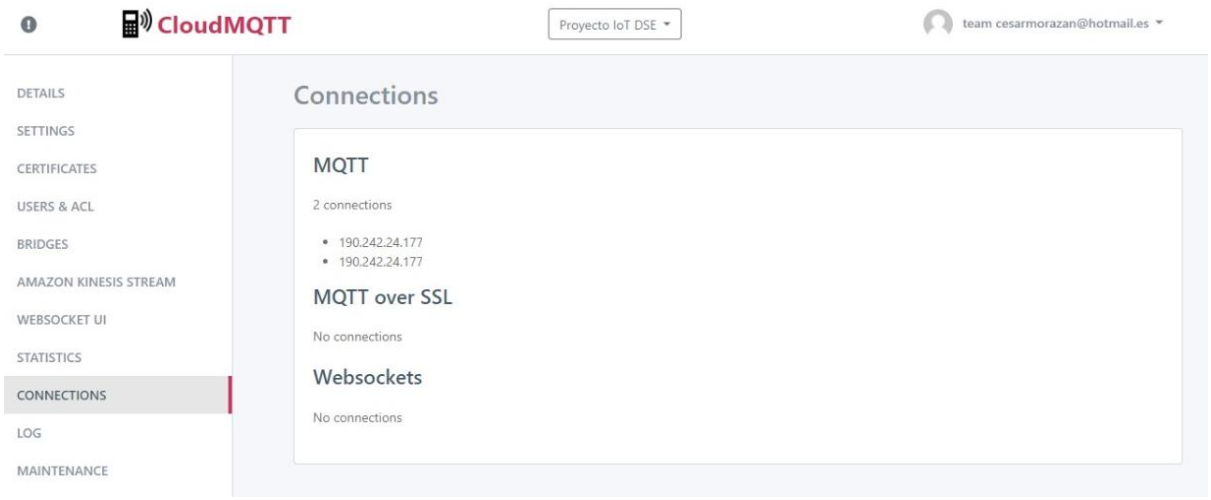


Figura 4.17 Conexión realizada al broker.

4.8.4. IoT MQTT Panel

En la figura anterior se mostraban 2 conexiones al broker como se puede observar, pero esto se debe a que previamente había configurado con las mismas credenciales del broker para la conexión a al celular mediante la aplicación móvil IoT MQTT Panel, este será el panel de monitoreo de datos.

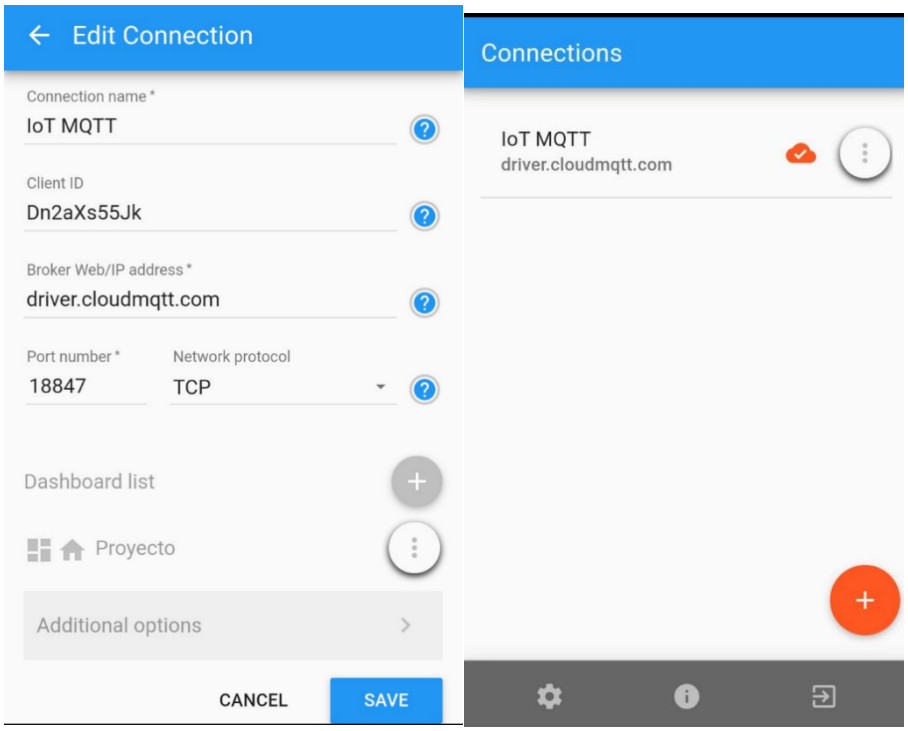


Figura 4.18 Configuración general del IoT MQTT Panel.

4.9. Testeo básico usando un sensor DHT11

Para la demostración del envío y recepción de datos pub/sub se usa un único sensor para validar el uso para así poder tener la base del proceso del sistema de monitoreo y recolección de datos que será desarrollado y mostrado en la sección de proceso y resultados de este informe.

4.9.1. Sensor DHT11

El DHT11 es un sensor digital de temperatura y humedad relativa de bajo costo y fácil uso. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Utilizado en aplicaciones académicas relacionadas al control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más.

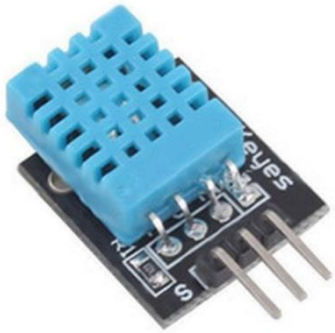


Figura 4.19 Modulo de sensor DHT11.

4.9.2. Programación para los sensores

Una de las variables importantes es tener conocimientos sobre programación en Arduino ya que para el ESP32 se programará de la misma manera lo que son los bucles y declaración de variables para obtener el funcionamiento correcto y salidas del o los sensores que se deseen implementar.

La única diferencia es que no se mostraran directamente en pantalla los datos leídos por los sensores o dispositivos conectados, en esta parte entra en juego lo que es la publicación y suscripción del protocolo.

Una vez que se ha programado el sensor la parte para la salida seria la siguiente como en el bloque de código

```
String mes = String(t);
mes.toCharArray(msg, 50);
client.publish("temperatura", msg);
Serial.println("La temperatura actual es -> " + String(t));

String mes2 = String(h);
mes2.toCharArray(msg2, 50);
client.publish("humedad", msg2);
Serial.println("La humedad actual es -> " + String(h));
```

Figura 4.20 Bloques de código encargado de publicar datos y enviarlos al broker para los distintos suscriptores del tópico.

Las variables t y h son donde se almacenan los datos de las temperaturas y la humedad detectada en el lugar donde este instalado el sensor, luego este se convierte en un mensaje que será publicado y enviado hacia el broker, como se aprecia en el código "client.publish("temperatura", msg); este le dice al sistema que se publique un mensaje bajo el tópico temperatura y que la carga útil será msg en donde se almaceno la temperatura leída, igual que para la humedad, esto hará que se le envié la información al broker y este reenvié todos los datos recolectados a los distintos dispositivos que estén suscritos a estos tópicos, igualmente se pueden mostrar en el monitor serial los valores enviados.

4.9.3. Lectura y Datos Recolectados

Ya con todo montado estamos listos para hacer primeras pruebas. Una vez subido el programa a la placa se muestra esto el monitor de la placa, buenas noticias.

```
Conectando a...
Insane
.....
C[NUL]Intentando Conexión MQTT
Conexión a MQTT exitosa
La temperatura actual es -> 27.50
La humedad actual es -> 57.00
La temperatura actual es -> 27.60
La humedad actual es -> 57.00
La temperatura actual es -> 27.50
La humedad actual es -> 57.00
```

Figura 4.21 Primer funcionamiento y observación mediante puerto serial del ESP32.

Lo segundo es revisar que el Broker este recibiendo esta información
Una vez ingresado al Websocket se observan los primeros mensajes recibidos lo que indica que hasta el momento no hay problemas, lo siguiente será revisar el panel IoT

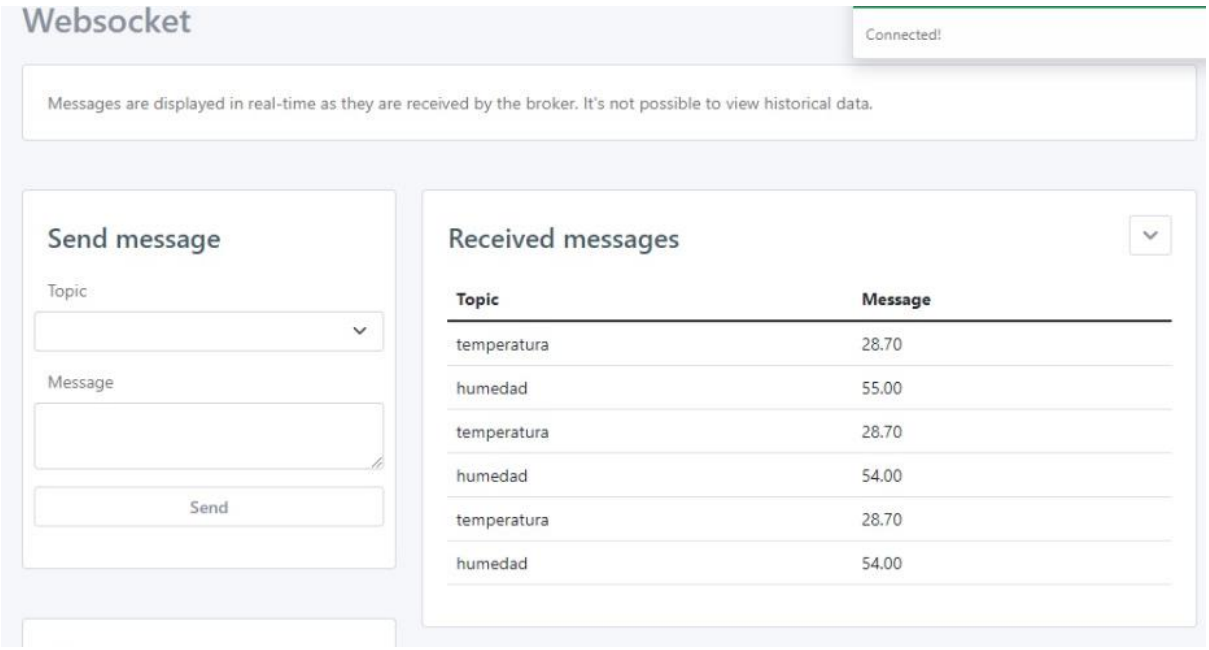


Figura 4.22 Mensajes recibidos con su tópico y dato en el broker.

Ahora que todo está conectado gracias a IoT es casi seguros que toda la parte experimental de esta investigación se realizó de manera exitosa, recordar que estos datos no se reciben en la misma red local, estos primeros son leídos, luego enviados al servidor mosquitto ubicado en Virginia del norte, y luego reenviados a los suscriptores, estos datos viajan tan rápido que es casi imperceptible si hay latencia en la comunicación ya que los datos viajan a través de fibra óptica submarina.

Figura 4.23 Demostración de que se están recibiendo y transmitiendo los datos desde el broker hacia el panel en dispositivo móvil.

Como se puede observar paneles para el sensor doble se reciben los datos ya que se está suscritos a tópicos individuales con cada uno.

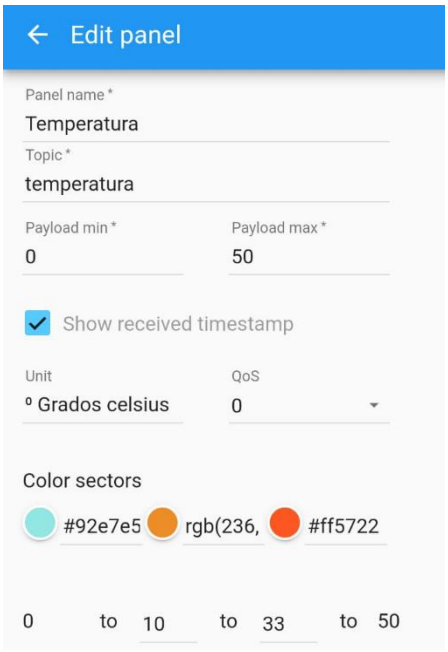


Figura 4.24 Configuración del panel de temperatura.

Como se puede observar el programa de testeo para comprobar que existiera la conexión mediante la placa ESP32, MQTT y el Broker ha funcionado correctamente, por lo que la base para el sistema esta lista. En la próxima sección de anexos se observará el funcionamiento de un sistema más complejo que el del testeo, incluyendo el listado de los materiales y equipo a utilizar y al mismo tiempo el código fuente del programa que se realice.

Capítulo V: Metodología

5.1. Enfoque y métodos

Esta investigación tiene un enfoque cualitativo debido a que no genera datos estadísticos, los datos se obtienen por observación, medición y documentación y tiene como meta describir, explicar, comprobar y predecir los fenómenos.

La información recopilada en el marco teórico de esta investigación está basada en la definición, explicación y segmentación de términos y conceptos característico, directamente relacionados con el tema principal del trabajo, con alcance Descriptivo. El alcance de esta investigación es respaldado por el tipo de resultado que se ha esperado obtener, los cuales serán adquiridos a través de un proceso de recolección de datos que pretende comprender el funcionamiento de lo que es un sistema de monitoreo y recolección de datos utilizando MQTT, ESP32 y el IoT. El alcance descriptivo permite comprender las dimensiones del fenómeno de análisis, permite determinar rasgos y puntos clave para el desarrollo deseado.

5.2. Unidad de análisis y respuesta

La unidad de análisis principal para este proyecto de graduación son investigaciones previamente realizadas por otros investigadores sobre las variables principales que son el protocolo MQTT, IoT y ESP32.

La unidad de respuesta es cómo se comporta el protocolo MQTT según las investigaciones realizadas para aplicaciones IoT, como la latencia en la comunicación, la perdida de datos a larga distancia, resultados de entregas de paquetes, su fiabilidad frente a otros protocolos para IoT, entre otros.

5.3. Técnicas e instrumentos aplicados

Los instrumentos de medición para esta investigación serán el análisis de contenido cuantitativo, gráficos porcentuales para comprobación de confiabilidad del protocolo y

programa de recolección de datos, panel de monitoreo MQTT como equipo de recolección de datos, también algunos datos secundarios o investigaciones realizadas previamente por otros investigadores (incluyendo mediciones realizadas con proyectos ya elaborados) e investigación documental.

5.4. Fuentes de información

5.4.1. Fuentes primarias

- Behrens, S. (2019, agosto 9). From Oil Pipelines to the IoT: A Brief History of MQTT [Online]. *PAESSLER*. https://blog.paessler.com/a-brief-history-of-mqtt

- Bernstein, C. (2021). MQTT (MQ Telemetry Transport) [Revista]. *IoT Agenda. TechTarget*. https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport
- HiveMQ. (2020). *MQTT & MQTT 5 Essentials A comprehensive overview of MQTT facts and features for beginners and experts alike*. E-Book.

- HiveMQ. (2022). MQTT Protocol [Online]. *The Messaging and Data Exchange Protocol of the IoT*. https://www.hivemq.com/mqtt/mqtt-protocol/

- Ranger, S. (2020, Febrero 3). What is the IoT? Everything you need to know about the Internet of Things right now [Online]. *ZDNet*. https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

- Sánchez Stahlschmidt, P. L. (2019). *MASTER EN ARDUINO ¡INCLUYE IoT! INTERNET OF THINGS!* [Curso Online]. Udemy. https://www.udemy.com/course/master-en-arduino/

5.4.2. Fuentes secundarias

- Barrio Andrés, M. (2018). *Internet de las cosas*. Reus.

- Hervás Parra, C. (2018). *Análisis de rendimiento de protocolos de publicación/subscripción en comunicación con una red de sensores inalámbricos Zigbee* [Magister en Redes de Datos, Universidad Nacional de La Plata]. https://doi.org/10.35537/10915/69435

- AVSystem. (2019, mayo 24). IoT Standards and protocols guide—Protocols of the Internet of Things [Online]. *AVSYSTEM*. https://www.avsystem.com/blog/iot-protocols-and-standards/

- IDC. (2014, mayo). *Worldwide and Regional Internet of Things (IoT) 2014–2020 Forecast: A Virtuous Circle of Proven Value and Demand* [Market Analysis]. http://branden.biz/wp-content/uploads/2017/06/IoT-worldwide_regional_2014-2020-forecast.pdf

- IDC. (2021, Junio 9). European IoT Spending to Exceed $200 Billion in 2021 [Market Analysis]. *IDC Media Center*. https://www.idc.com/getdoc.jsp?containerId=prEUR147929621

- Orielsystems. (2022). *A SHORT HISTORY OF TELEMETRY* [Online]. https://orielsystems.com/short-history-telemetry/

- Mandado Pérez, E., Mario Espiñeira, P., & Lago Ferreiro, A. (1995). *Instrumentación electrónica*. Marcombo. http://site.ebrary.com/id/10357205

- Mijarez Castro, R. (2014). *Electrónica*. http://site.ebrary.com/id/11013154

- Mishra, B., Mishra, B., & Kertesz, A. (2021). Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements. *Energies*, *14*(18), 5817. https://doi.org/10.3390/en14185817

- The Editors of Encyclopaedia Britannica. (s. f.). *Telemetry*. https://www.britannica.com/technology/telemetry

5.4.3. Fuentes terciarias

- Amazon Web Services. (s. f.). What Is ESP32? [Online]. *Workshop: ESP32 with AWS IoT*. https://catalog.us-east-1.prod.workshops.aws/workshops/5b127b2f-f879-48b9-9dd0-35aff98c7bbc/en-US/module1/esp32

- OASIS. (2019). *MQTT Version 5.0 Oasis Standard*. http://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

- CloudMQTT. (2022). *CloudMQTT Documentation* [Online]. https://www.cloudmqtt.com/docs/index.html

- Espressif. (2021). *ESP32 Series Datasheet*. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

- IoTicos. (2019, julio 30). *Desentrañando MQTT - ¿Cómo funciona?* [Video Online].

5.5. Cronología del trabajo

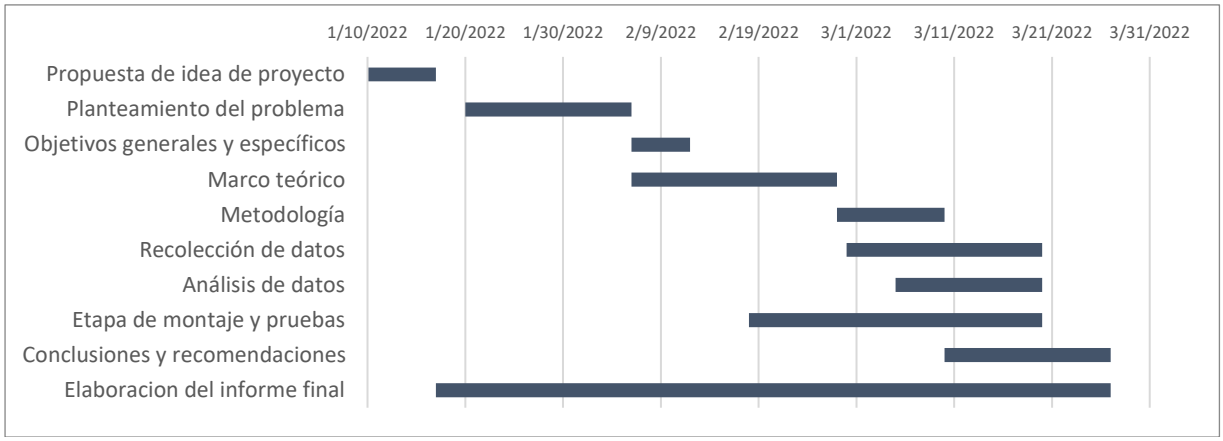


Gráfico 5.1 Cronología del trabajo.

Capítulo VI: Resultado y Analisis

6.1. Fase experimental

En esta fase se utilizó como instrumento el montaje realizado para la verificación de la eficiencia y eficacia del protocolo MQTT mediante el editor de código PlatformIO basado en el IDE Arduino pero con más prestaciones y más cómodo para su programación, se manipularon variables desde un punto inicial el cual fue el tiempo de conexión o latencia de conexión con el broker, en el cual se realizaron 10 conexiones desde 0 para ver el tiempo de respuesta a la conexión dándonos como resultados distintos tiempos debido a la carga de los servidores y la distancia de los servidores. También se utilizó Excel como medio para la creación de los gráficos.

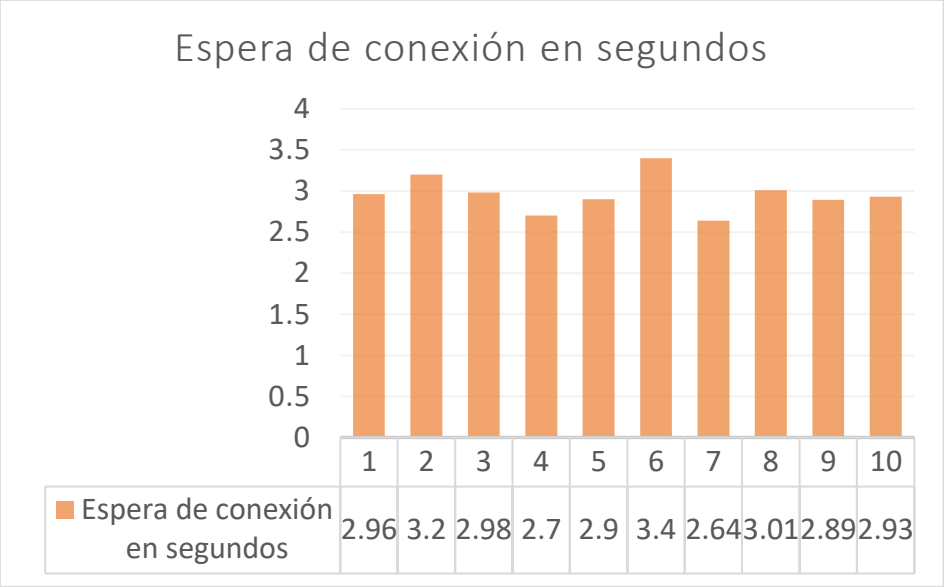


Gráfico 6.1 Latencia de comunicación inicial en segundos, 10 pruebas realizadas.

Esta latencia de conexión se presenta como única vez cuando se conecta un sistema por primera vez, luego de esto los datos son enviados de manera casi instantánea. Esto se explica en la recolección de datos basados en investigación documental debido a que se necesitan programas especializados para medir tiempos menores a 0.01 segundos y también el consumo de ancho de banda.

6.2. Recolección de datos por investigación documental

Esta recolección y análisis de datos es gracias a investigaciones realizadas anteriormente en el área de ciencias de la computación, una realizada por el master en redes de datos Carlos A. Hervás de la universidad de la plata, Argentina, la cual lleva por nombre Análisis de rendimiento de protocolos de Publicación/Subscripción en comunicación con una Red de Sensores Inalámbricos Zigbee, en donde se comparan rendimientos del protocolo MQTT (entre otros) en redes de sensores y por medio de conexión TCP/IP. La otra tesis que se tomó para la recolección de datos es del departamento de ingeniería de Software de la universidad Szeged de Hungría, donde sus autores son Biswajeeban Mishra, Biswaranjan Mishra, y Attila Kertesz, con el título de Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements, en el cual se sometieron con cargas grandes información a distintos brokers MQTT para analizar y medir su rendimiento.

Brokers	Average Latency in ms.		
	QoS0	QoS1	QoS2
Mosquitto 2.0.7	0.47	0.50	0.98
Bevywise MQTT Route 3.1- build 0221-017	0.89	0.69	1.30
ActiveMQ 5.16.1	0.83	1.09	0.64
HiveMQ CE 2020.2	2.07	4.48	3.38
VerneMQ 1.11.0	0.79	0.90	1.10
EMQ X 4.2.7	0.59	1.35	1.22

Tabla 6.1, *Latencia de comunicación promedio de distintos brokers.*

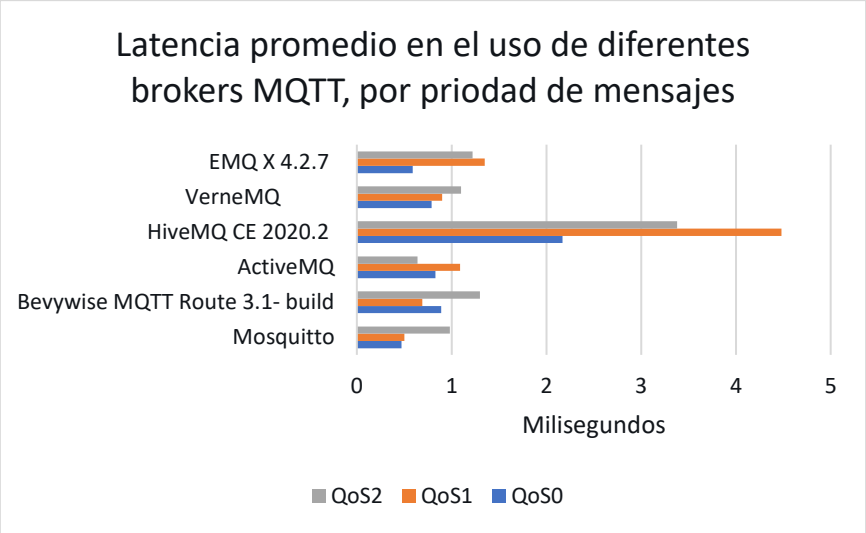


Gráfico 6.2 Latencia de comunicación comprobada por el equipo húngaro.

Tasa de entrega y pérdida de publicaciones MQTT-SN QoS 0												
Nodos WSN			Nodo Gateway	Nodo Suscriptor		Tasa de Entrega (%)			Total Pub. Perdidas	Tasa de Pérdida (%)		
# de nodos	Pub. Realizadas	Pub. Perdidas	Pub. Enviadas	Pub. Perdidas	Pub. Recibidas	WSN	Red TCP/IP	TOTAL		WSN	Red TCP/IP	TOTAL
10	3000	1300	1700	22	**1678**	56,67	98,71	**55,93**	**1322**	43,33	1,29	**44,07**
20	6000	2260	3740	61	**3679**	62,33	98,37	**61,32**	**2321**	37,67	1,63	**38,68**
30	9000	3401	5599	85	**5514**	62,21	98,48	**61,27**	**3486**	37,79	1,52	**38,73**
40	12000	4659	7341	122	**7219**	61,18	98,34	**60,16**	**4781**	38,83	1,66	**39,84**
		Tasas Promedio de Entrega y Pérdida (%):				**60,60**	**98,47**	**59,67**		**39,40**	**1,53**	**40,33**

Tabla 6.2, *Resultados de Tasa de entrega y pérdida de publicaciones MQTT QoS0.*

Se puede observar los resultados de la tasa de entrega y pérdida de publicaciones del mecanismo de confiabilidad QoS 0 en función al número de nodos que generan publicaciones. Teniendo una tasa de entrega promedio del 59,67% y una tasa de pérdida promedio del 40,33% en la recepción de publicaciones.

Tasa de entrega, retransmisión y pérdida de publicaciones MQTT-SN QoS 1													
Nodos WSN		Nodo Gateway			Tasa de Retrans-misión (%)	Nodo Suscriptor		Tasa de Entrega (%)			Total Pub. Perdidas	Tasa de Pérdida (%)		
# de nodos	Pub. Realizadas	Pub. Perdidas	Pub. Enviadas	Pub. Retrans-mitidas		Pub. Recibidas	Pub. Perdidas	WSN	Red TCP/IP	TOTAL		WSN	Red TCP/IP	TOTAL
10	3000	186	2814	**75**	**2,67**	2805	**9**	93,80	99,67	**93,49**	**195**	6,20	0,33	**6,51**
20	6000	408	5592	**154**	**2,75**	5578	**14**	93,20	99,75	**92,97**	**422**	6,80	0,25	**7,03**
30	9000	1164	7836	**202**	**2,58**	7803	**33**	87,07	99,58	**86,70**	**1197**	12,93	0,42	**13,30**
40	12000	718	11282	**305**	**2,70**	11249	**33**	94,02	99,70	**93,74**	**751**	5,98	0,30	**6,26**
Tasas Promedio de Entrega, Retransmisión y Pérdida (%):					**2,68**			**92,02**	**99,68**	**91,72**		**7,98**	**0,32**	**8,28**

Tabla 6.3, *Resultados de entrega, retransmisión y pérdida de publicaciones MQTT QoS1.*

Se puede observar los resultados de la tasa de entrega, retransmisión y pérdida de publicaciones del mecanismo de confiabilidad QoS 1 en función al número de nodos que generan publicaciones. Teniendo una tasa de entrega promedio del 91,72%, tasa de retransmisión del 2,68% y una tasa de pérdida promedio del 8,28% en la recepción de publicaciones.

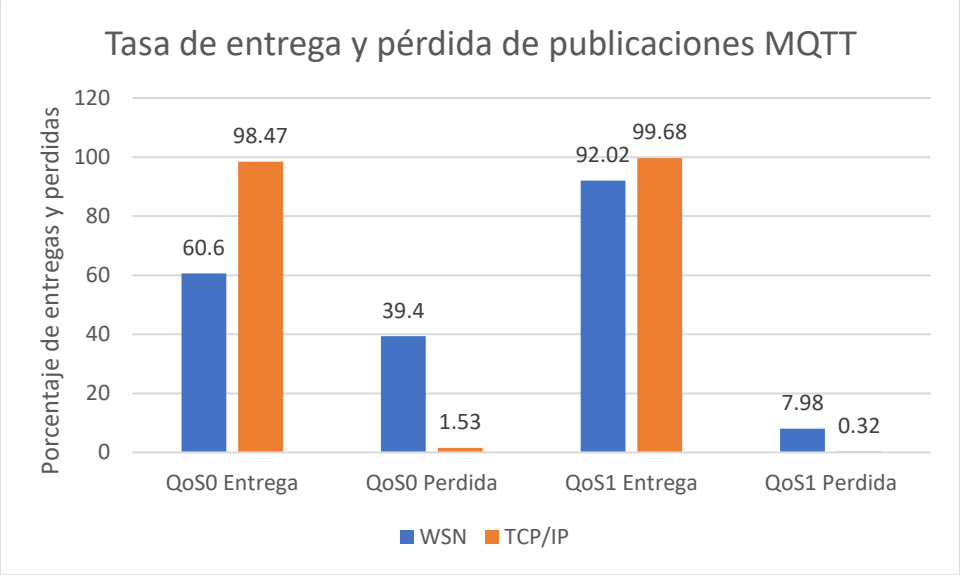


Gráfico 6.3 Resultados en porcentaje de entrega y perdida de datos.

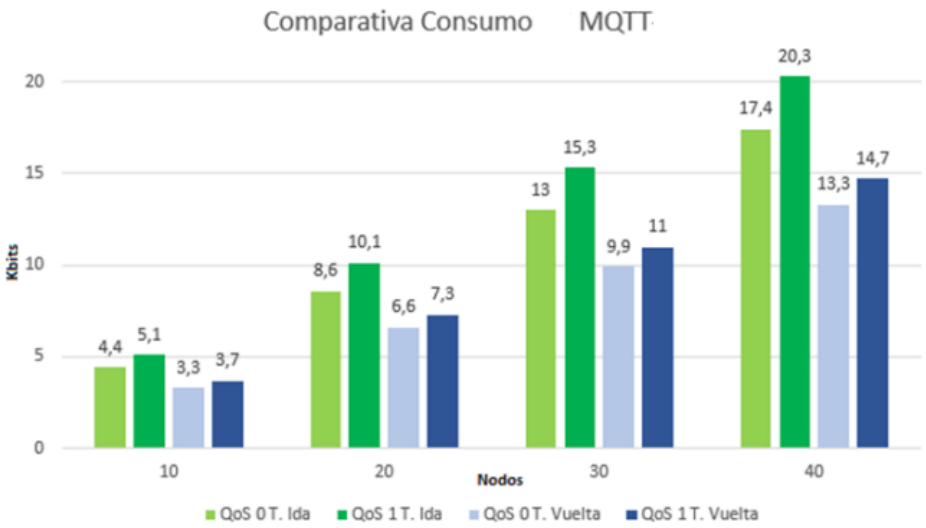


Gráfico 6.4 Resultados de consumo de Kbit por cantidad de nodos en redes de sensores MQTT.

6.3. Futuro de las aplicaciones IoT

Las aplicaciones en el internet de las cosas están abarcando todos los sectores comerciales e industriales a día de hoy, desde juguetes hasta monitoreo de misiles, algo interesante es que ya existen prótesis de extremidades que se conectan al IoT, también sensores corporales que envían datos para monitoreo en pacientes con enfermedades, como tacómetros, glucómetros e implantes cerebrales como es el caso de Neuralink los cuales están desarrollando una interfaz cerebro-computadora para personas con discapacidades neurológicas.

6.3.1. Hogares inteligentes

Para los consumidores, el hogar inteligente es probablemente el lugar donde es probable que entren en contacto con cosas habilitadas para Internet, y es un área en la que las grandes empresas de tecnología (en particular, Amazon, Google y Apple) compiten duro.

Los más obvios son los parlantes inteligentes como el Echo de Amazon, pero también hay enchufes inteligentes, bombillas, cámaras, termostatos y el frigorífico inteligente tan burlado. Pero además de mostrar su entusiasmo por los nuevos y brillantes dispositivos, hay un lado más serio de las aplicaciones para el hogar inteligente. Pueden ayudar a mantener a las personas mayores independientes y en sus propios hogares por más tiempo al facilitar que la familia y los cuidadores se comuniquen con ellos y controlen cómo les va. Una mejor comprensión de cómo funcionan en hogares y la capacidad de modificar esas configuraciones podrían ayudar a ahorrar energía, al reducir los costos de calefacción, por ejemplo.

6.3.2. Internet de las cosas y ciudades inteligentes

Al distribuir una gran cantidad de sensores en un pueblo o ciudad, los planificadores pueden tener una mejor idea de lo que realmente está sucediendo, en tiempo real. Como resultado, los proyectos de ciudades inteligentes son una característica clave del IoT. Las ciudades ya generan grandes cantidades de datos (de cámaras de seguridad y sensores ambientales) y ya contienen grandes redes de infraestructura (como las que controlan los

semáforos). Los proyectos de IoT tienen como objetivo conectarlos y luego agregar más inteligencia al sistema.

6.3.3. IoT y 5G

Un área de crecimiento en los próximos años será sin duda el uso de redes 5G para respaldar proyectos de IoT. 5G ofrece la capacidad de colocar hasta un millón de dispositivos 5G en un kilómetro cuadrado, lo que significa que será posible utilizar una gran cantidad de sensores en un área muy pequeña, lo que hará más posibles las implementaciones de IoT industrial a gran escala. El Reino Unido acaba de comenzar una prueba de 5G e IoT en dos 'fábricas inteligentes'. Sin embargo, podría pasar algún tiempo antes de que las implementaciones de 5G se generalicen: se predice que habrá alrededor de cinco mil millones de dispositivos IoT conectados a redes celulares para 2025, pero solo alrededor de una cuarta parte de ellos serán IoT de banda ancha, con 4G conectando la mayoría de los dispositivos. aquellos.

6.3.4. Datos de IoT e inteligencia artificial

Los dispositivos de IoT generan grandes cantidades de datos; que podría ser información sobre la temperatura de un motor o si una puerta está abierta o cerrada o la lectura de un medidor inteligente. Todos estos datos de IoT deben recopilarse, almacenarse y analizarse. Una forma en que las empresas aprovechan al máximo estos datos es alimentarlos a sistemas de inteligencia artificial (IA) que tomarán esos datos de IoT y los usarán para hacer predicciones.

En realidad, para esta área tecnológica la imaginación es el límite.

6.4. Limitaciones

Las limitaciones encontradas durante el desarrollo de este proyecto fueron varias, una de las principales es que al trabajar con aplicaciones IoT se debe mantener la conexión a internet y una suscripción de pago a un broker MQTT u otro protocolo para IoT, si no se cuenta con una conexión no se enviaran los datos a los dispositivos correspondientes, estos

quedaran almacenados en cola hasta que el dispositivo vuelta a conectarse, por lo cual no se perdería información pero si se conecta por ejemplo un sensor que envié datos cada 2 segundos y este llega a desconectarse de internet por una hora, al conectarse de nuevo se recibirán demasiados datos en el broker y también los dispositivos suscritos a los tópicos que perdieron la conexión, para esto se debe programar muy bien un sistema, así previniendo estos inconvenientes como la acumulación excesiva de datos, un posible arreglo para esto sería programar que en caso de perder la comunicación via internet el dispositivo entre en modo Deep sleep o modo espera y programar los sensores o dispositivos conectados para que al suceder esto sus mediciones se prolonguen a tiempos más largos, otra solución es implementar un módulo SIM800L que sirva como modem en caso de perder conexión a red, el inconveniente seria que el SIM deberá contar con datos para que la placa ESP32 pueda conectarse a internet de nuevo.

Otra de las limitaciones al realizar un sistema de monitoreo y recolección es al momento de usar una batería como fuente de alimentación externar en caso de perder conexión por USB, si el sistema ESP32 es cargado con distintos módulos y una única batería de respaldo podría durar poco su carga útil, debido a que el ESP32 al conectarse a un broker está enviando cargas útiles cada cierto tiempo según lo programado, algunos dispositivos necesitan voltajes distintos a los 3.3v y 5v que proporciona el ESP32, para esto se usan fuentes externas, en este caso particular me encontré con que el módulo SIM800L que utilice requiere un voltaje de 3.7v a 4.3v por lo cual tenía que implementar una batería de 3.7v como fuente externa o utilizar un diodo en polarización directa para disminuir el voltaje en 0.7v manteniendo una corriente de 1A, algo que sucedió es que en esta placa ESP32 es que los pines GND (3 en total) no funcionan de manera individual al conectar un dispositivo con un voltaje mayor a 3.3v, por lo cual es necesario conectar los 3 pines a una tierra común para poder tener respuesta de la placa y también utilizar divisores de voltaje.

Mas que una limitación lo siguiente es un dato importante y es que al montar un dispositivo así en un lugar remoto, que no se pueda tener conectado siempre a un fuente de alimentación mediante USB, que su uso sea móvil o con fallas de energía eléctrica constantes es que se puede utilizar una alimentación externa como respaldo como una batería de litio, para esto es necesario analizar el datasheet de la placa a utilizar en mi caso ESP-WROOM-32 la cual tiene un diodo Schottky en la entrada por USB que impide que el ESP32 sea alimentado al mismo tiempo por 2 fuentes (en este caso una batería y el puerto de carga USB)

al mismo tiempo, pero solo en una dirección, si la fuente externa supera la barrera del diodo este podría llegar a causar serios daños en el sistema, para esto se deben de implementar otros dispositivos electrónicos para evitar estos daños, como diodos o fusibles, también sería necesario la implementación de un transistor MOSFET para permitir el paso de corriente desde el USB hacia la batería para poder contar con su carga completa al momento de perder conexión eléctrica y contar con la de respaldo, también otros dispositivos para controlar la carga de la batería y no llegar a sobrecargara.

6.5. Hallazgos

Entre los hallazgos principales destaca la facilidad con la que se pueden realizar aplicaciones IoT mediante la placa de desarrollo ESP32 y el protocolo MQTT, una persona familiarizada con el entorno de programación de Arduino podría sacar mucho provecho al pasarse a esta placa de desarrollo ya que este permite implementar una librería para poder programar como si fuera el Arduino. Son casi infinitas las posibilidades de desarrollo y aplicación de estos elementos, se puede realizar desde un sencillo prendido y apagado de un led hasta realizar una interconexión de distintos dispositivos que se comunican entre si como en este caso que estará en anexos de este proyecto, donde se envían y reciben datos desde distintos lugares y todo en un orden concreto programado.

Debido a las prestaciones, costo y facilidad de programación de la placa ESP32 y los atributos de fiabilidad del protocolo MQTT brinda la oportunidad de incursionar en la elaboración de dispositivos ya sea por entretenimiento, uso personal o para uso en la industria siendo este último uno de los campos más prometedores ya que se estiman inversiones muy grandes para la automatización y la conexión de dispositivos a internet.

Capítulo VII: Conclusiones

Con la realización del sistema de monitoreo y recolección de datos el cual se logró utilizando el sensor KY-026 como dispositivo para alarma de incendios y el sensor MQ-05 como sensor para detección de fugas de gas, se aprendió que el protocolo MQTT es un protocolo de transporte de mensajería entre máquinas donde se pudo observar su funcionamiento gracias a lo realizado y el cual permite la creación de sistemas de monitoreo para IoT recolectando datos, pudiendo transmitir dichos datos de manera remota, esto gracias a la telemetría, donde los sensores y dispositivos envían los datos a una placa programada, donde luego son transmitidos a un broker MQTT para así llegar a los usuarios finales y paneles de monitoreo.

Con los sistemas de monitoreo y recolección de datos se puede monitorear desde algo tan pequeño como un llavero inteligente, hasta industrias enteras, esto debido a la construcción que puede ser flexible al implementarse y la fácil programación del protocolo MQTT la cual permite que el funcionamiento pueda variar según las necesidades que se quieran cubrir, pudiendo monitorear valores en tiempo real de múltiples dispositivos instalados a larga distancia gracias a un broker MQTT.

Y por último concluir que debido a las predicciones realizadas por analistas de mercado es importante conocer lo que es IoT incluyendo saber cómo crear un dispositivo basado en esta tecnología, teniendo la posibilidad de desarrollar un producto personal o comercial, donde al utilizar placas como la ESP32 se pueden realizar proyectos de bajo costo y grandes prestaciones que pueden implementarse en áreas como la domótica, automatización de procesos y otros sectores comerciales que utilizan las tecnologías del IoT para la monitorización de dispositivos.

Capítulo VIII: Recomendaciones

Para futuros proyectos basados en este mismo sistema de recolección, monitoreo y conexión basado en el protocolo MQTT utilizando la placa de desarrollo ESP32 se debería implementar una fuente de alimentación externa con la opción de poder cargar la batería (fuente externa) al momento de conectar la placa a la fuente de energía por USB, si esto se realiza se podrá mejorar y crear un prototipo real que funcione de manera independiente cuando no esté conectado a un cargador o una computadora.

Leer la documentación del fabricante de los dispositivos obtenidos, ya que muchos pueden tener el mismo nombre, pero diferentes valores nominales, como en este caso que se encontró algo curioso pero molesto al tener que conectar las 3 tierras a una tierra común para poder utilizar ciertos dispositivos conectados a la placa ESP32.

Contar con el equipo adecuado para realizar pruebas de voltajes y corrientes por si no está seguro lo que hace, indagar en los límites para poder sacar provecho de este sistema. Otra recomendación es revisar foros especializados y respuestas de expertos con puntuaciones reales como expertos cuando se tengan dudas sobre los sistemas con ESP32 y MQTT, evitar tomar información de sitios informales, debido a que el tema es relativamente nuevo para el hispanohablante la información muchas veces es muy reducida, para esto si cuenta con un nivel promedio de ingles le será mucho más fácil este tipo de búsqueda ya que se encuentra mucha más información. También para futuras investigaciones y proyectos de monitoreo, recomiendo a los investigadores enfocarse en encontrar información estructurada lo más clara y comprensible para su evaluación. Por otro parte comprender bien el patrón publicación/suscripción para poder enviar datos de los sensores y dispositivos al broker a utilizar.

Capitulo IX: Bibliografía

Amazon Web Services. (s. f.). What Is ESP32? [Online]. *Workshop: ESP32 with AWS IoT*. https://catalog.us-east-1.prod.workshops.aws/workshops/5b127b2f-f879-48b9-9dd0-35aff98c7bbc/en-US/module1/esp32

AVSystem. (2019, mayo 24). IoT Standards and protocols guide—Protocols of the Internet of Things [Online]. *AVSYSTEM*. https://www.avsystem.com/blog/iot-protocols-and-standards/

Barrio Andr??s, M. (2018). *Internet de las cosas*. Reus.

Behrens, S. (2019, agosto 9). From Oil Pipelines to the IoT: A Brief History of MQTT [Online]. *PAESSLER*. https://blog.paessler.com/a-brief-history-of-mqtt

Bernstein, C. (2021). MQTT (MQ Telemetry Transport) [Revista]. *IoT Agenda. TechTarget*. https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport

CloudMQTT. (2022). *CloudMQTT Documentation* [Online]. https://www.cloudmqtt.com/docs/index.html

Espressif. (2021). *ESP32 Series Datasheet*. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Hervas Parra, C. (2018). *Análisis de rendimiento de protocolos de publicación/subscripción en comunicación con una red de sensores inalámbricos Zigbee* [Magister en Redes de Datos, Universidad Nacional de La Plata]. https://doi.org/10.35537/10915/69435

HiveMQ. (2020). *MQTT & MQTT 5 Essentials A comprehensive overview of MQTT facts and features for beginners and experts alike*. E-Book.

HiveMQ. (2022). MQTT Protocol [Online]. *The Messaging and Data Exchange Protocol of the IoT*. https://www.hivemq.com/mqtt/mqtt-protocol/

IDC. (2014, mayo). *Worldwide and Regional Internet of Things (IoT) 2014–2020 Forecast: A Virtuous Circle of Proven Value and Demand* [Market Analysis].

http://branden.biz/wp-content/uploads/2017/06/IoT-worldwide_regional_2014-2020-forecast.pdf

IDC. (2021, junio 9). European IoT Spending to Exceed $200 Billion in 2021 [Market Analysis]. *IDC Media Center*. https://www.idc.com/getdoc.jsp?containerId=prEUR147929621

Mandado P??rez, E., Mari??o Espi??eira, P., & Lago Ferreiro, A. (1995). *Instrumentación electrónica*. Marcombo. http://site.ebrary.com/id/10357205

Mijarez Castro, R. (2014). *Electrónica*. http://site.ebrary.com/id/11013154

Mishra, B., Mishra, B., & Kertesz, A. (2021). Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements. *Energies*, *14*(18), 5817. https://doi.org/10.3390/en14185817

OASIS. (2019). *MQTT Version 5.0 Oasis Standard*. http://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

Orielsystems. (2022). *A SHORT HISTORY OF TELEMETRY* [Online]. https://orielsystems.com/short-history-telemetry/

Ranger, S. (2020, febrero 3). What is the IoT? Everything you need to know about the Internet of Things right now [Online]. *ZDNet*. https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

Sánchez Stahlschmidt, P. L. (2019). *MASTER EN ARDUINO ¡INCLUYE IoT! INTERNET OF THINGS!* [Curso Online]. Udemy. https://www.udemy.com/course/master-en-arduino/

The Editors of Encyclopaedia Britannica. (s. f.). *Telemetry*. https://www.britannica.com/technology/telemetry

Capitulo X: Anexos

En esta sección se demostrarán los montajes realizados, así como sus diagramas y programación utilizada, para esto es necesario haber leído previamente el capítulo 4, donde se explican y detallan cada una de las funciones del protocolo MQTT y la placa ESP32, también para evitar confusiones al interpretar el código programado y poder entender el funcionamiento del broker y el panel MQTT.

Antes de comenzar es necesario conocer los pines de conexión y su ubicación en la placa, como se mencionó en las recomendaciones se debe revisar el datasheet del fabricante de la placa, ya que hay distintos fabricantes de las placas de desarrollo pudiendo variar la ubicación de los pines.

Figura 10.1 Diagrama de conexión de pines de placa de desarrollo ESP-WROOM-32.

10.1. Montaje realizado y demostración de funcionamiento

El primer montaje que se realizó para este proyecto se trata de un sistema de transmisión y recolección de datos en tiempo real utilizando lo que son un sensor de gas MQ-5, este sensor se utiliza en equipos de detección de fugas de Gas LPG, gas natural, gas de carbón, este se usa tanto en aplicaciones domesticas como industriales, y también lo que es un sensor de flama KY-026, este módulo es útil para sistema de detección de incendios, como

una medida de seguridad, cabe mencionar que ambos módulos utilizados cuentan con un potenciómetro para ajustar su sensibilidad en caso de usarse los pines de comunicación digital, en este caso el sensor MQ-5 se comunicó con la placa ESP32 de manera análoga mientras que el sensor de flama se conectó de manera digital para demostrar ambos funcionamientos.

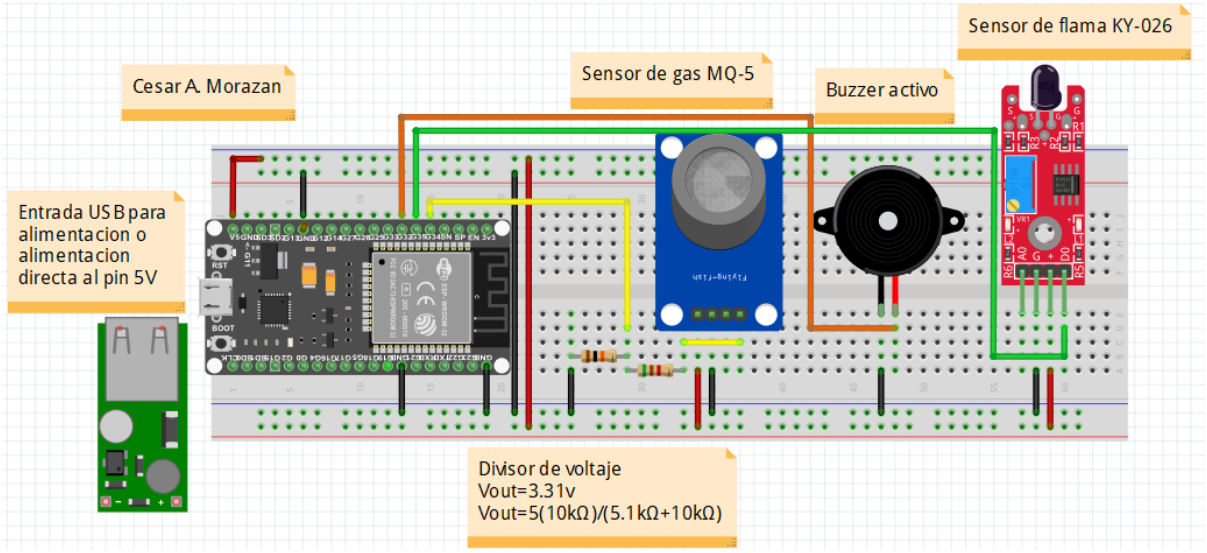


Figura 10.2 Diagrama del primer montaje del sistema de transmisión y recolección de datos.

KY-026 a GPIO35

MQ-5 a GPIO34

Buzzer a GPIO32

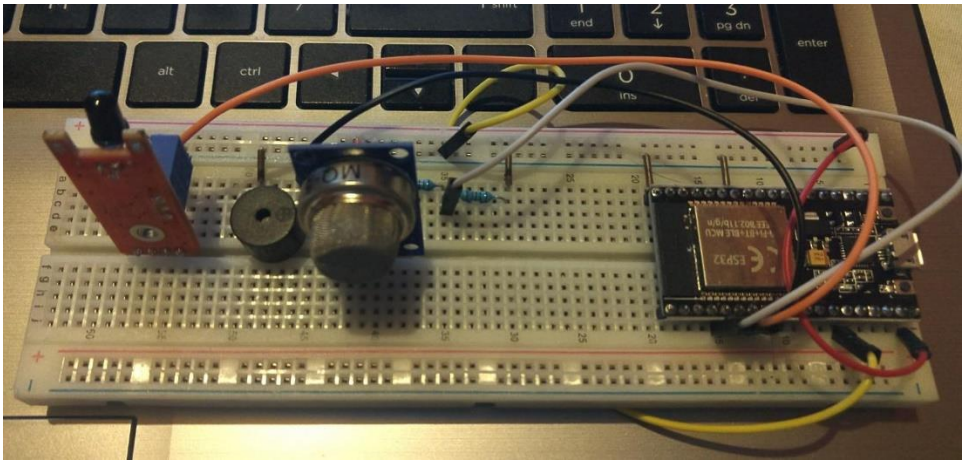


Figura 10.3 Montaje físico del sistema de transmisión y recolección de datos.

Antes de explicar el funcionamiento de este se revisó el datasheet del fabricante el cual indica que los pines ADC2 solo funcionan de manera digital y no de manera análoga al utilizar la librería wifi la que es necesaria para que los sensores envíen los datos a la placa y luego se publicados en el broker. También se conectaron los 3 pines GND a una tierra común y se utilizó un divisor de voltaje debido a que el sensor MQ-5 utiliza 5v y los pines de salida y entrada no son tolerantes a 5 voltios, en caso de no utilizar un divisor de tensión y conectar directamente los cables de comunicación se recibirán datos erróneos o muy elevados de los sensores.

Al final de esta sección se adjuntará el código de programación del sistema, una vez iniciado el programa se puede comprobar en la siguiente figura que los datos de los sensores se están recibiendo por el monitor serial y que hemos conectado al broker.

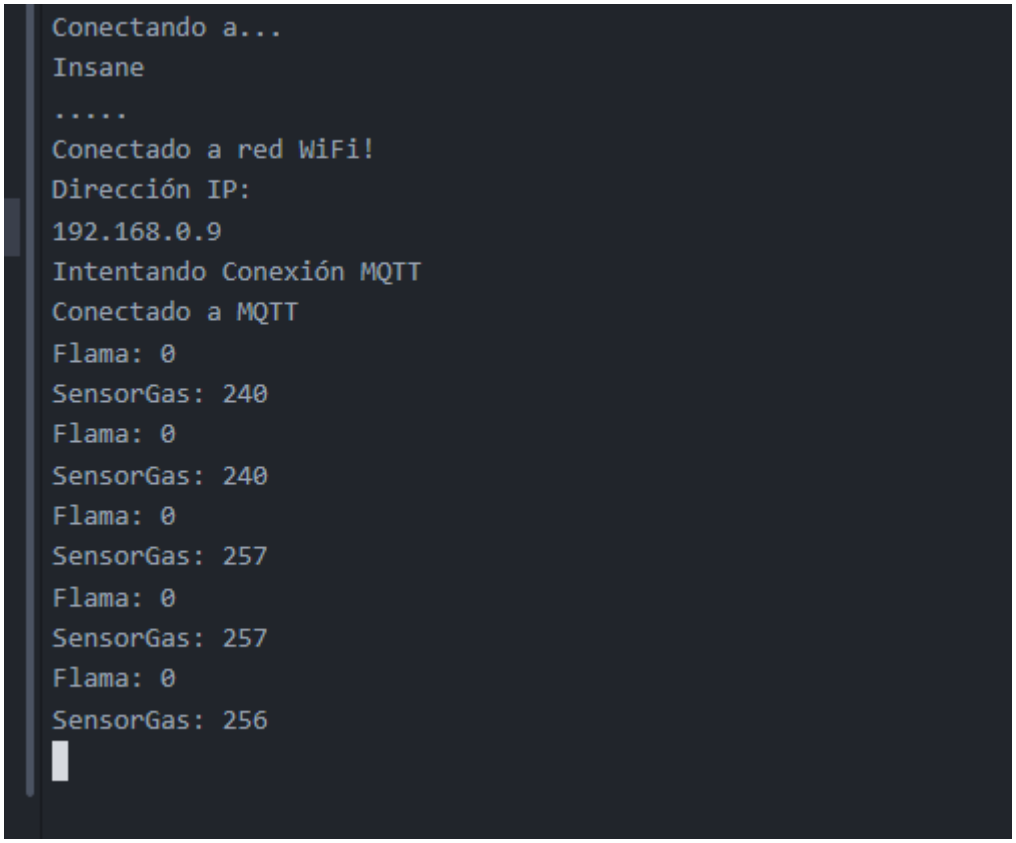


Figura 10.4 Datos mostrados en el monitor serial.

Si recordamos el sensor de gas nos envía una señal análoga entre 0 y 1024 según el fabricante, donde valores superiores a 400 muestran la detección de gas en el sensor, por lo cual se programó para que si en caso de detectar una fuga envié una alerta que se mostrara en otras figuras más adelante.

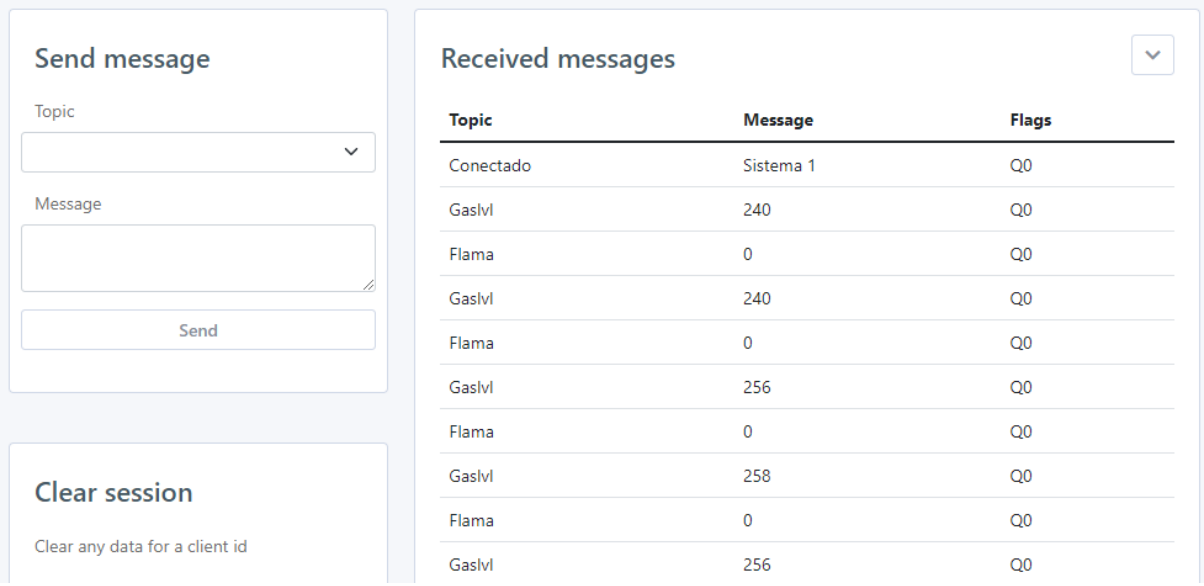


Figura 10.5 Datos enviados por el ESP32 al broker MQTT.

Como se observa en la imagen anterior al conectarse el sistema este le envía un mensaje al broker bajo el tópico conectado y con la carga útil que es “Sistema 1” lo cual sirve como identificador en caso de tener múltiples sistemas interconectados.

Los valores mostrados por el sensor de gas son publicados bajo en el broker bajo el tópico Gaslvl, el cual sirve para hacer un monitoreo a distancia de esto, en la siguiente figura se observa un ejemplo desde el panel MQTT en un celular.

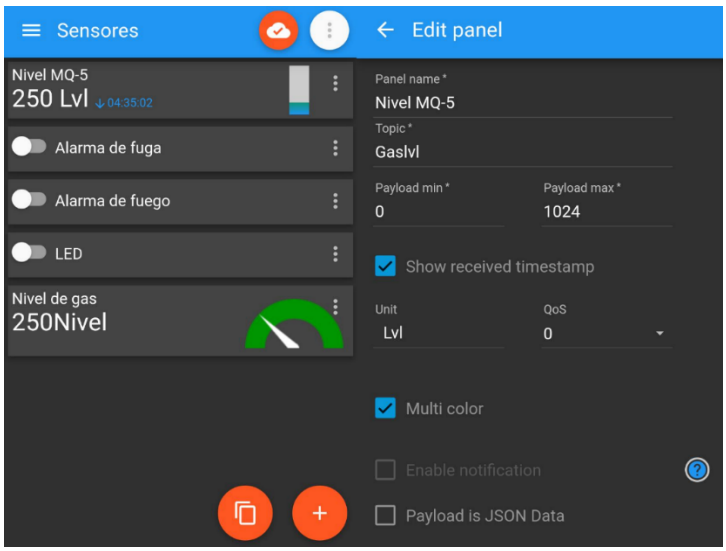


Figura 10.6 Panel MQTT suscrito al tópico Gaslvl para el monitoreo de datos.

De la imagen anterior contamos también con una alarma de fuga y una de incendio, en el caso del sensor de flama al estar conectado por comunicación digital siempre mostrara un 0 y un 1, este último cuando detecta la presencia de fuego.

Si dicho sistema detecta una fuga de gas o fuego, estos enviaran los datos al broker e inmediatamente se activara el buzzer mientras sigan detectando esto, para evitar que se apague la alarma y replicarla a otros sistema de alarma conectado se realizó lo siguiente, los paneles en forma de switch en la aplicación móvil están suscritos al tópico FUGA_S1 y FUEGO_S1 estos son los tópicos que publicaran en caso de detectar una falla, estos con carga útil como Sistema 1 para dar a conocer dónde están los fallos, pero los paneles MQTT (alarmas switch) publicaran al mismo tiempo bajo un tópico una activación de todas las alarmas que estén suscritas a esos tópicos, teniendo que hacer una desactivación manual de alarmas, esto debido a que si sucede un incendio por ejemplo es muy probable que los sistemas que detecto el fuego se estropee y se desactive su alarma instalada, por eso es necesario que se activen otras más para tener presente en que sistema y donde ocurrió el fallo.

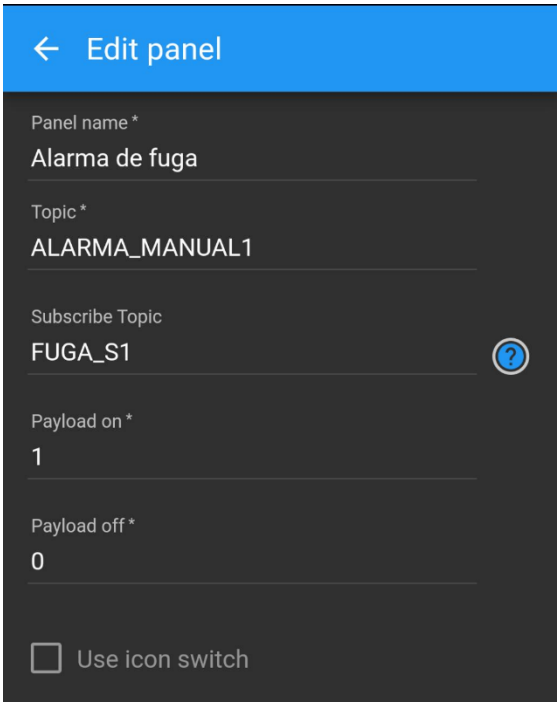


Figura 10.7 Configuración de alarma de fuga.

Como se comentó, una alarma recibe el dato en caso haber fuga, al recibir el dato se activa el switch y luego se publica un tópico con una carga útil, en este caso llamado ALARMA_MANUAL1 enviando una carga útil 1 para activar las demás alarmas que estén

configuradas como suscriptores de ese tópico, y al mismo tiempo pudiendo desde la misma app desactivar las alarmas al mover el switch a su posición inicial haciendo que se publique un 0 bajo el tópico ALARMA_MANUAL1. Para esto debe estar debidamente programado el comando.

A continuación, se hará una prueba en el sensor de gas y el sensor de flama utilizando un encendedor o mechero desechable.

También se configuro que al activarse cada una de los sensores y estos publiquen los mensajes de alarma del panel MQTT móvil reciba una marca de tiempo para saber a qué hora se recibió esta alarma, en caso de estropearse en un accidente el sensor dejara de enviar los datos y quedaremos únicamente con la última marca de tiempo, pero en caso contrario se estará enviando cada 2 segundos.

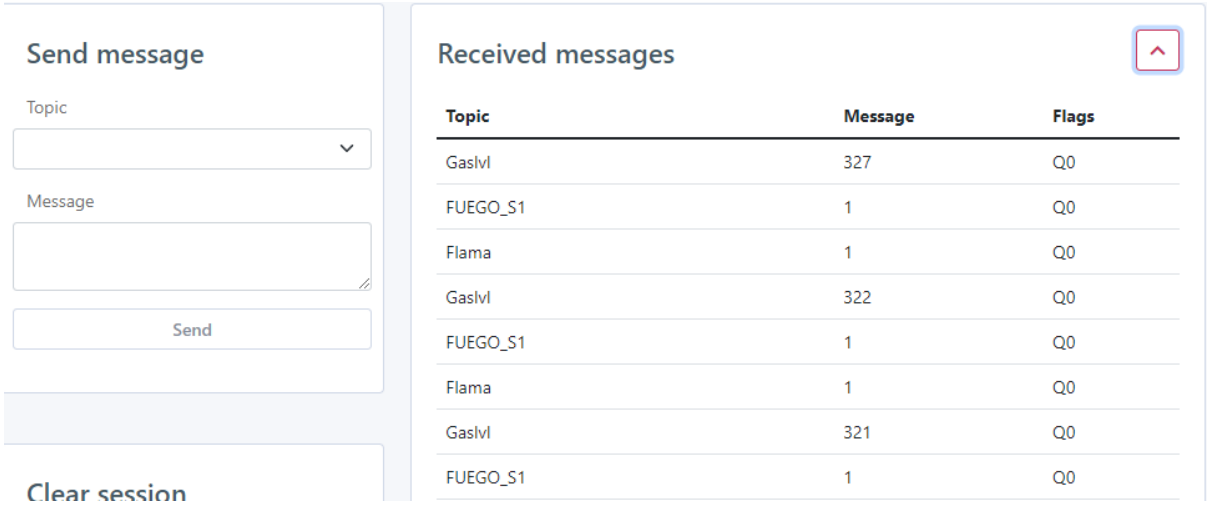


Figura 10.8 Datos enviados por la placa ESP32.

FUGA_S1	1	Q0
Gaslvl	682	Q0
Flama	0	Q0
FUGA_S1	1	Q0
Gaslvl	754	Q0
Flama	0	Q0
FUGA_S1	1	Q0
Gaslvl	651	Q0

Figura 10.9 Datos enviados por la placa ESP32 pt2.

Como observamos al acercar una flama o al presionar el encendedor para liberar el gas butano los sensores publican según lo programado, en caso del sensor de flama pasa a un Alto donde se envía un 1 y al mismo tiempo publicando bajo el tópico FUEGO_S1 y el sensor de gas muestra sus lecturas análogas y publica FUGA_S1 mostrado la carga útil que es la necesaria para activar las alarmas.

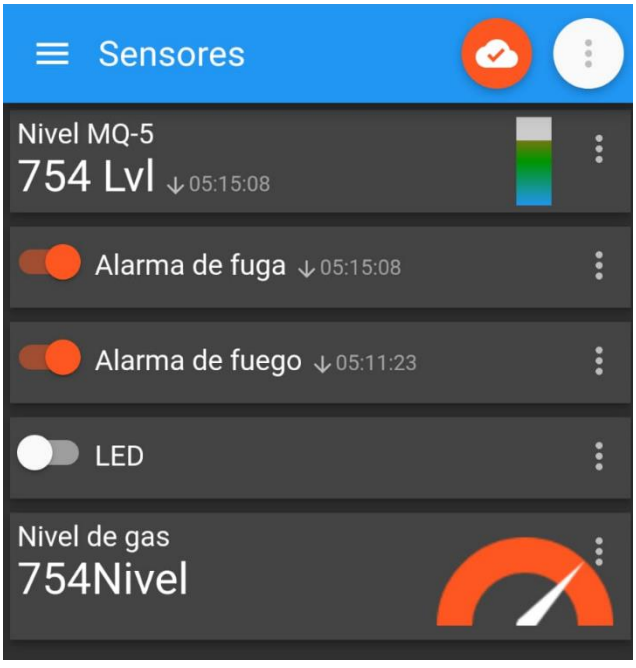


Figura 10.10 Datos recibidos y activación de las alarmas.

En la figura anterior se demuestra que todo el sistema de conexión, recolección y monitoreo está funcionando correctamente, se puede observar que se activaron las alarmas

programadas con sus respectivas marcas de tiempo así como un nivel que muestra lo detectado por el sensor de gas MQ-5, en caso de no contar con una aplicación en el celular también se pueden enviar comandos para conexión y desconexión mediante el Websocket del broker, este permite publicar bajo un tópico e ingresar la carga útil que queremos enviar, sirve como un apagado manual extra en caso de fallos.

Este es un sistema sencillo únicamente utilizado para demostración, pero también tiene múltiples funciones, el sensor MQ-5 se puede montar en tuberías para detección de gases como también en la cocina de una casa como dispositivo de alarma en caso de olvidar cerrar una perilla de una estufa a gas, el sensor de flama de igual manera se puede montar en una casa para como dispositivo de alarma de incendio como también en una industria o como dispositivo de control forestal utilizando un fotodiodo más potente y con mayor rango. También cabe la posibilidad de montar los 2 de manera conjunta como sistema de alarma para domótica, como una mejora se podría montar un dispositivo electromecánico como un relé que al detectar fuego en una casa se active localmente o mediante el protocolo MQTT un dispositivo contra incendios.

10.2. Codigo de programa utilizado

```
#include <Arduino.h>// libreria para utilizar esp32 con código de arduino en platformio
#include <Wifi.h> // librería para wifi
#include <PubSubClient.h>//libreria para uso del mqtt

const char* ssid = " "; // Nombre de red(router)
const char* password = " "; //contraseña

//Credenciales del servidor MQTT
const char *mqtt_server = "driver.cloudmqtt.com"; //servidor broker
const int   mqtt_port = 18648; //Puerto del broker
const char *mqtt_user = ""; //usuario del boker
const char *mqtt_pass = ""; //contraseña del broker

WiFiClient espClient; // creacion del objeto wifi
PubSubClient client(espClient);

//declaracion de variables
char msg[50];
long lastMsg = 0;
int buzzer = 32;
const int SensorFlama = 35;
const int SensorGas = 34;

//function callback
void callback(String topic, byte* payload, unsigned int length) {
  Serial.print("Mensaje recibido: ");
  Serial.print(topic);
  Serial.print(", ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }

  //condicion para activado general de alarmas en caso de fuga
  if ((topic=="ALARMA_MANUAL1") && ((char)payload[0] == '0')) {
    digitalWrite(buzzer, LOW);
  } else if ((topic=="ALARMA_MANUAL1") && ((char)payload[0] == '1')) {
    digitalWrite(buzzer, HIGH);
  }
//condicion para activado general de alarmas en caso de incendio
  if ((topic=="ALARMA_MANUAL2") && ((char)payload[0] == '0')) {
    digitalWrite(buzzer, LOW);
  } else if ((topic=="ALARMA_MANUAL2") && ((char)payload[0] == '1')) {
    digitalWrite(buzzer, HIGH);
  }

  Serial.println();
}
//función para forzar conexion a internet
void reconnect(){

  while (!client.connected()) {
    Serial.println("Intentando Conexión MQTT");

    String clientId = "Proyecto Final";
    clientId = clientId + String(random(0xffff), HEX);

    if (client.connect(clientId.c_str(),mqtt_user,mqtt_pass)) {
      Serial.println("Conectado a MQTT");
      client.publish("Conectado", "Sistema 1");
      client.subscribe("LED");// suscripción a los tópicos individuales
      client.subscribe("ALARMA_MANUAL1");
      client.subscribe("ALARMA_MANUAL2");

    }
```

```
    else{
      Serial.println("Falló la conexión ");
      Serial.print(client.state());
      Serial.print(" Se intentará de nuevo en 5 segundos");
      delay(5000);
    }
  }
}
//función para conexión a red wifi
void setup_wifi(){
  // Conexión a la red Wifi
  Serial.println();
  Serial.println("Conectando a...");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("Conectado a red WiFi!");
  Serial.println("Dirección IP: ");
  Serial.println(WiFi.localIP());

}
//función realizada para el sensor KY-026 y MQ-5
void FLAMAS() {
  int ValorGas = analogRead(SensorGas);
  int ValorFlama = digitalRead(SensorFlama);

  Serial.print("Flama: ");
  Serial.println(ValorFlama);
  Serial.print("SensorGas: ");
  Serial.println(ValorGas);

    // activa el buzzer
  if (ValorFlama==1 or ValorGas > 500) {
    if(ValorFlama==1){
      client.publish("FUEGO_S1", "1");
    }
    else {
      client.publish("FUGA_S1", "1");

    }
    for (size_t i=0; i < 10; i++) {
      digitalWrite(buzzer, HIGH);
      delay(200);
      // Desactiva el buzzer
      digitalWrite(buzzer, LOW);
      delay(100);
    }
  }
}
//activación de los pines como entrada o salida
void setup() {
  pinMode(buzzer, OUTPUT);
  pinMode(SensorFlama, INPUT);
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}
```

```
void loop() {
  if(client.connected()==false){
    reconnect();
  }

  client.loop();

  if (millis() - lastMsg > 2000){
    lastMsg = millis();

    FLAMAS();
    int ValorGas = analogRead(SensorGas);
    String gaspayload = String(ValorGas);
    gaspayload.toCharArray(msg, 50);
    client.publish("Gaslvl", msg);
    int ValorFlama = digitalRead(SensorFlama);
    String flamapayload = String(ValorFlama);
    flamapayload.toCharArray(msg, 50);
    client.publish("Flama", msg);
  }

}
```

10.3. Rentabilidad y flexibilidad de aplicación

Los costos que abarcaron este proyecto son los siguientes:

Componente	Unidades	Costo en dólares
Placa ESP-WROOM-32	1	6.50
Sensor MQ-5	1	3
Sensor KY-026	1	5.59
Protoboard	1	4
Cables y Jumpers	Varias	5
Suscripción a MQTT Cloud	Mensual	4,99
Buzzer activo	2	5
Diodos y resistencias para divisor de voltaje	Varias	3 aprox.

Tabla 10.1, *Gastos realizados para demostración de sistema 1.*

Los costos anteriormente detallados se manejaron en dólares debido a que los componentes se compraron en el extranjero ya que incluyendo el costo de envió de 7$ estos salían más económicos que adquirirlos en una tienda local.

Al final el gasto total asciende a los 45$ aproximadamente, con el modelo realizado se puede hacer una comparación de rentabilidad con elementos existentes pero debido a que el sistema desarrollado en este proyecto no tiene el fin de ser un prototipo se aleja de lo que podría ser un estudio de mercado debido a que este sistema puede adaptarse e implementarse a múltiples tipos de sensores y dispositivos y dado a esto no habrá un buen punto de referencia.

Donde se puede tomar una comparación mas precisa es en el montaje que se realizo con efecto de prueba y rastreo utilizando un módulo GPS, el módulo sim800l y la placa esp32 sin ningún otro dispositivo conectado donde el modulo tenia un precio de 11$, la antena para el módulo neo 7m un costo de 4$, un módulo sim800l con el precio de 5$, la placa ESP32 el costo de 6.5$ y el pago del broker MQTT con un costo de 4.99$, donde el costo total era de 31.5$, en el mercado actual existen diferentes tipos de rastreadores satelitales que utilizan estos módulos rondando precios entre los 30$ y 100$ donde igualmente que en este caso se

necesita la conexión constante a internet o tener conexión a internet mediante la tarjeta sim del módulo sim800l que hace de modem en el sistema.

Debido a que este proyecto implementaba la placa de desarrollo ESP32 y no únicamente el módulo ESP32 es difícil hacer una estimación pero este funciona de igual manera, se puede decir por medio de montajes y pruebas realizadas que el una placa de desarrollo ESP32 puede superar a cualquier producto actual debido a que en este se pueden agregar cualquier tipo de sensores a petición de un cliente o preferencia personal pudiendo después ser modificado en la marcha, al mismo tiempo la característica de que se puede programar a preferencia pudiendo realizar ciertas acciones de interés.

La flexibilidad y capacidad de integración del sistema realizado se limita únicamente los pines de comunicación con los que cuenta, pudiendo conectarse una cantidad considerable de dispositivos a la placa ESP32.