

Monitor de criptomonedas con MQTT y ESP8266

Gerardo Elián Martínez Ramírez

Ingeniería de Sistemas

Pontificia Universidad Javeriana

Bogotá, D. C.

gerardoemartinez@javeriana.edu.co

Abstract—This document presents the design, implementation, and validation of a prototype system for monitoring cryptocurrency prices using embedded devices and the MQTT protocol. The system consists of a sensor node that fetches real-time data from Binance and publishes it via MQTT, and a monitor node that displays selected cryptocurrency prices on a screen while providing visual and audible alerts based on configurable thresholds through an app. The MQTT protocol is used as the communication backbone, enabling lightweight, real-time message exchange between modules.

Keywords—MQTT, Broker, HTTP, TCP/IP, API, Client, Cryptocurrency, Publish/Subscribe, Microcontroller, IoT.

I. INTRODUCCIÓN

Una criptomoneda es un tipo de activo digital que utiliza criptografía para garantizar transacciones seguras y descentralizadas. Estas monedas no dependen de bancos centrales y permiten transferencias directas entre usuarios a nivel global. Su valor puede fluctuar ampliamente según la oferta, demanda y factores del mercado.

Un *trader* de criptomonedas se dedica a comprar y vender activos digitales con el objetivo de obtener ganancias aprovechando las variaciones de precio. Para su labor, utiliza, entre otras, herramientas como plataformas de intercambio, gráficos de análisis técnico, bots de *trading* automatizado y sistemas de alertas de precios.

El proyecto consiste en construir un dispositivo que permita visualizar en tiempo real el valor de tres criptomonedas distintas que el usuario pueda elegir, así como señalar la variación del valor e incluir una alarma sonora que se active cuando el precio de la criptomoneda seleccionada alcance un valor predeterminado por el usuario, facilitando la toma de decisiones rápidas ante cambios relevantes del mercado.

II. PROTOCOLO MQTT

El **internet de las cosas** (IoT) se ha estado usando ampliamente en el desarrollo de identificación por radiofrecuencia (RFID), tecnologías de comunicación, protocolos de internet y sensores inteligentes. El IoT explota las tecnologías subyacentes de estos objetos para transformarlos desde lo tradicional a lo inteligente. Considerándolo como un sistema de dispositivos y máquinas mecánicas interrelacionadas provistas de identificadores únicos, tiene la habilidad de transmitir datos en una red sin requerir de la interacción con otros objetos como humanos u otro computador [1].

En 1999, Andy Stanford-Clark de IBM y Arlen Nipper introdujeron un protocolo de mensajería llamado MQTT. En

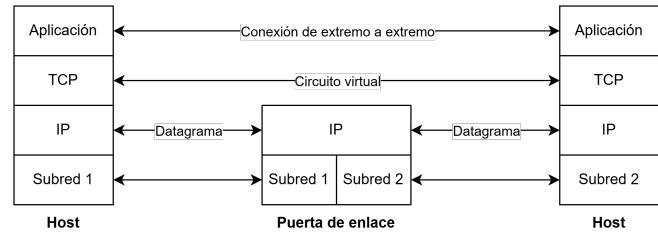


Fig. 1. Arquitectura del TCP/IP.

2013, el MQTT se convirtió en el protocolo estándar de la Organización para el Avance de Estándares de Información Estructurada (OASIS). El **transporte de telemetría de cola de mensajes** (MQTT) es un protocolo *push* estandarizado de mensajería ligero con publicación/suscripción. Fue planeado para enviar datos de manera precisa según el retraso y el ancho de banda bajo de la red [2]. El propósito del protocolo es usarlo en dispositivos con capacidades de memoria restrictivas y con poder de procesamiento limitado como en el IoT.

MQTT es un protocolo publicar-suscribir, y tras conectarse a un servidor MQTT (bróker), un cliente puede publicar y suscribirse a múltiples tópicos. El bróker actúa como un servidor de recursos (RS) responsable de distribuir los mensajes publicados por los publicadores a sus suscriptores [3].

A. Protocolos previos

1) **TCP/IP**: TCP es un protocolo de transmisión que permite la comunicación entre dos aplicaciones a través de una red. A diferencia de UDP, TCP establece una conexión entre el emisor y el receptor durante el proceso de transferencia, garantizando que todos los paquetes lleguen a su destino.

MQTT opera sobre **TCP/IP**, el cual es un conjunto de protocolos, aplicaciones y utilidades de red, como se ve en la figura 1. TCP provee una conexión confiable de circuito virtual entre dos hosts. IP provee un servicio de transporte de datagramas sobre cualesquier subredes. Ya que TCP e IP fueron originalmente imaginados para funcionar como un solo protocolo, este conjunto, que se refiere a una gran colección de protocolos y aplicaciones, normalmente se conoce simplemente como TCP/IP [4].

2) **HTTP**: El **protocolo de transferencia de hipertexto** (HTTP) es un protocolo de nivel de aplicación con la ligereza

y velocidad necesarias para los sistemas de información de hipermedia distribuidos y colaborativos. Está basado en un paradigma de petición-respuesta, en contraste con el modelo publicar-suscribir asíncrono de MQTT.

Un cliente establece una conexión con un servidor y envía una petición al servidor en forma de un método de petición, URI y versión de protocolo, seguido de un mensaje que contiene modificadores, información del cliente y posiblemente un contenido de cuerpo. El servidor responde con una línea de estado, incluyendo la versión del protocolo del mensaje y un código de error o éxito, seguido de un mensaje que contiene información del servidor, metainformación de la entidad y posiblemente un contenido de cuerpo. La mayoría de la comunicación HTTP es iniciada por un usuario agente y consiste en una petición para ser aplicada en un recurso en algún servidor de origen [5]. En MQTT los clientes suscritos reciben automáticamente los mensajes sin tener que consultar continuamente al servidor como en HTTP.

B. Conceptos básicos

El protocolo MQTT conecta las redes y dispositivos con *middleware* y aplicaciones utilizando patrones de comunicación máquina a servidor (M2S), servidor a servidor (S2S), máquina a máquina (M2M) y mecanismos de enrutamiento (uno a muchos, uno a uno y muchos a muchos) [1]. MQTT se ejecuta sobre TCP/IP utilizando un modelo *publish/suscribe* donde existen dos tipos de sistemas: clientes y brókeres.

1) *Tópicos y suscripciones*: Un publicador publica mensajes y los usuarios se suscriben a ciertos tópicos. Los suscriptores se suscriben a tópicos particulares que se relacionan con ellos y así reciben los mensajes que se publican en esos tópicos, mientras que los clientes pueden publicar mensajes a los tópicos de manera que permita a todos sus suscriptores acceder a ellos [2].

Los mensajes se publican bajo un nombre de tópico y los suscriptores se suscriben a dichos nombres para recibir los mensajes correspondientes. El bróker utiliza el nombre del tópico en un mensaje publicado para determinar a qué suscriptores transmitir los mensajes [3].

Existen dos tipos de suscripciones:

- **Suscripciones no compartidas**: está asociada solo con una sesión MQTT que la creó. Una sesión no puede tener más de una suscripción no compartida con el mismo filtro de tópico.
- **Suscripciones compartidas**: puede estar asociada con múltiples sesiones MQTT suscriptoras. Se identifica usando un estilo especial de filtro `$share/{nombreCompartido}/{filtro}`.

2) *Nombres y filtros de tópicos*: La barra “/” es usada para separar cada nivel en un árbol de tópicos y proveer una estructura jerárquica. Su uso es relevante junto con los caracteres de comodín. Pueden aparecer en cualquier parte del nombre [3].

- El signo de numeral “#” es un carácter de comodín que coincide con cualquier número de niveles en un tópico. Debe aparecer por su cuenta o tras un separador de nivel.

En cualquier caso es el último carácter especificado en el filtro.

- El signo de más “+” es un carácter de comodín que coincide solo con un nivel del tópico. Puede usarse en cualquier nivel en el filtro, donde debe ocupar un nivel entero de este. Puede usarse en más de un nivel y en conjunto con el comodín multinivel.

3) *Calidad del servicio*: En el protocolo MQTT se establecen tres diferentes niveles de **calidad del servicio** (QoS), los cuales se negocian entre dos partes de un mensaje con respecto de la garantía de la distribución de datos [6].

- **Nivel 0**: no considera la ejecución de retransmisiones de mensajes, por lo que no implica la garantía de entrega de los mismos. El mensaje puede ser entregado una única vez o no ser entregado. Por ejemplo, un sensor de temperatura que envíe datos cada segundo.
- **Nivel 1**: existe la garantía de que la entrega de los mensajes sucederá, pero no hay un control de entregas duplicadas. El mensaje es entregado al menos una vez y como máximo N veces, donde N es el número de retransmisiones realizadas. Por ejemplo, alarma que necesita confirmación y puede repetirse.
- **Nivel 2**: es el mayor nivel permitido por MQTT, donde los mensajes también tienen garantía de entrega y además la retransmisión duplicada se elimina. El mensaje es entregado solo una vez. Por ejemplo, una transacción financiera que debe procesarse exactamente una vez.

4) *Mensajes retenidos*: Un mensaje retenido es un mensaje MQTT normal con una bandera Retain en verdadero. El bróker guarda el último mensaje retenido y la QoS correspondiente para ese tópico. Cada cliente que se suscribe a un patrón de tópico que coincide con el tópico del mensaje retenido recibe el mensaje retenido inmediatamente después de que se suscriben. El bróker almacena solo un mensaje retenido por tópico.

Son cruciales para proveer a los clientes recién suscritos una actualización de estado inmediato al suscribirse a un tópico. Un mensaje retenido representa esencialmente el último valor válido conocido para un tópico particular. No necesariamente tiene que ser el valor más reciente, sino el mensaje más reciente con la bandera en verdadero [7].

5) *Puerto*: MQTT usa dos puertos primarios disponibles en los brókeres por defecto.

- Conexiones no cifradas: 1883.
- Conexiones cifradas con TLS/SSL: 8883.

6) *Colas de mensajes*: Existe confusión sobre si MQTT está implementado como colas de mensajes o no. Una cola de mensajes almacena mensajes hasta que son consumidos. Cada mensaje es guardado hasta que se recoja por un cliente, y si no es recogido, el mensaje queda atorado en la cola esperando ser consumido. No es posible que un mensaje no sea procesado por ningún cliente, como es en MQTT si nadie está suscrito al tópico [7].

Las colas deben ser creadas explícitamente. Solo después de que una cola es nombrada y creada, es posible publicar o

consumir mensajes. En contraste, los tópicos MQTT son muy flexibles y pueden ser creados sobre la marcha.

7) *Sesiones persistentes*: Las **sesiones persistentes** en MQTT permiten al cliente mantener su suscripción y estado del mensaje a través de múltiples conexiones. Cuando un cliente establece una sesión persistente con un bróker, el bróker guarda la información de la suscripción del cliente y cualquier mensaje no entregado. Así, si el cliente se desconecta y se reconecta más tarde, puede reanudar la comunicación [3].

En una sesión persistente el bróker almacena la siguiente información (incluso si el cliente no tiene conexión). La información queda disponible inmediatamente al cliente al reconnectarse:

- Existencia de la sesión.
- Todas las suscripciones del cliente.
- Flujo de todos los mensajes en una QoS 1 o 2 donde el cliente no ha confirmado aún.
- Todos los nuevos mensajes con QoS 1 o 2 que el cliente se perdió cuando estaba sin conexión.
- Todos los mensajes con QoS 2 recibidos desde el cliente que estaban esperando reconocimiento.

Al establecer una conexión con el bróker, los clientes pueden habilitar o deshabilitar una sesión persistente al poner el valor de la bandera `cleanSession`.

8) *Last Will and Testament*: El *Last Will and Testament* (JWT) es una característica que permite que los clientes especifiquen un mensaje que será publicado automáticamente por el bróker en su nombre o cuando ocurra una desconexión inesperada. De esta manera asegura que los clientes manejen las desconexiones de manera correcta sin dejar tópicos en un estado inconsistente. Esto es útil cuando los clientes deben notificar a otros su indisponibilidad o mandar información importante al desconectarse inesperadamente.

Cuando un cliente se conecta a un bróker, puede especificar un mensaje como su última voluntad, el cual posee la estructura regular de los mensajes MQTT. El bróker guarda el mensaje hasta que detecta una desconexión inesperada del cliente, y luego manda el mensaje en *broadcast* a todos los clientes suscritos al tópico correspondiente. El bróker descarta el mensaje LWT si el cliente se desconecta con un mensaje DISCONNECT [7]. Para especificarlo, se incluye en el mensaje CONNECT en `lastWillTopic`, `lastWillQos`, `lastWillMessage` y `lastWillRetain`.

C. Paquetes de control

MQTT envía varios **paquetes de control** a través de la conexión de red. Se describen los más relevantes a continuación [8]:

- **CONNECT**: El cliente solicita conectarse al bróker. Es el primer paquete enviado por el cliente..
- **CONNACK**: Reconocimiento de la conexión del bróker. Los paquetes contienen códigos de retorno que indican éxito o un estado de error en respuesta al paquete CONNECT del cliente.

- **AUTH**: Se envía del cliente al bróker o del bróker al cliente como parte de un intercambio extendido de autenticación.
- **PUBLISH**: Petición de publicación enviada de un cliente publicador al bróker o del bróker a un cliente suscriptor.
- **PUBACK**: Respuesta a una petición PUBLISH con nivel QoS 1. Puede ser enviado del bróker al cliente o del cliente al bróker.
- **PUBREC**: Respuesta a una petición PUBLISH con nivel QoS 2. Puede ser enviado del bróker al cliente o del cliente al bróker.
- **SUBSCRIBE**: Petición de suscripción enviada desde el cliente.
- **SUBACK**: Reconocimiento de la suscripción desde el bróker al cliente.
- **PINGREQ**: Petición de ping enviada desde el cliente al bróker. Indica al bróker que el cliente está ahí y es usado para confirmar que el bróker también sigue ahí. El periodo keepalive se establece en el paquete CONNECT.
- **PINGRESP**: Respuesta enviada por el bróker al cliente en respuesta a PINGREQ. Indica que el bróker está ahí.
- **DISCONNECT**: Paquete final enviado desde el cliente o el bróker. Indica el motivo por el cual la conexión de red está siendo cerrada.

Los paquetes de control son la unidad más pequeña de transferencia de datos en MQTT. Los clientes MQTT y servidores intercambian paquetes de control para hacer su trabajo, como conectarse, suscribirse y publicar a tópicos.

1) *Will*: Si la conexión de red no se cierra normalmente, el bróker envía un último mensaje Will para el cliente si el cliente proveyó uno en su paquete CONNECT. Situaciones en que un mensaje Will es publicado incluyen [3]:

- Error I/O o falla de la red detectada por el bróker.
- El cliente falla en comunicarse durante el periodo de keepalive.
- El cliente cierra la conexión sin primero haber enviado un paquete DISCONNECT con un código de motivo normal 0x00.
- El bróker cierra la conexión sin primero haber recibido un paquete DISCONNECT con un código de motivo normal 0x00.

2) *Formato*: Sin importar el tipo del paquete de control, todos consisten en tres partes según la figura 2:

- **Encabezado fijo**: contiene tres campos como se observa en la figura 3. El tipo está localizado en los 4 bits altos del primer byte. En la tabla I se muestra el significado de cada valor. Los restantes 4 bits bajos en el primer byte contienen banderas determinadas por el tipo. En MQTT 5.0, solo el paquete PUBLISH tiene asignado significados específicos:

- Bit 3: Si el paquete es retransmitido.
- Bit 2 y 1: QoS.
- Bit 0: Si el paquete está retenido.

El campo restante indica el número de bytes en la parte restante del paquete de control, que incluye el encabezado

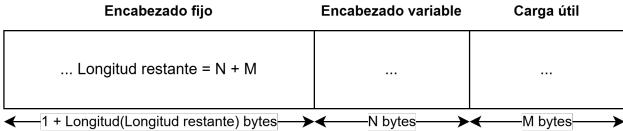


Fig. 2. Formato del paquete MQTT.



Fig. 3. Formato del encabezado fijo.

variable y la carga útil. Sin embargo, la longitud del encabezado fijo no es fija. Se utiliza el campo de longitud restante como un entero de bytes variables. Por ejemplo, para un mensaje de 2 MB, se necesitaría un entero de 4 bytes para indicar su longitud, pero el tamaño de ese entero sería excesivo para un mensaje de solo 2 bytes [9].

- **Encabezado variable:** el contenido del encabezado variable depende del tipo específico de paquete. Por ejemplo, para un paquete CONNECT se incluyen el nombre y nivel del protocolo, keepalive, propiedades, etc., mientras que uno PUBLISH lleva el nombre del tópico, identificador del paquete (si el QoS no es 0) y propiedades. Su forma se muestra en la figura 4.

Las propiedades, como se ve en la figura 5, introducidas con MQTT 5.0, son opcionales y suelen tener un valor por defecto. Cada una consisten en un identificador que define el propósito y el tipo de datos de la propiedad y un valor específico.

- **Carga útil (payload):** el encabezado variable puede considerarse como información adicional, mientras que la carga útil es el propósito principal del paquete. Por ejemplo, en un paquete PUBLISH serviría para llevar el mensaje de la aplicación real, y campos como el QoS en el encabezado variable proveen capacidades adicionales relacionadas [9].

D. Arquitectura

La arquitectura típica de MQTT puede ser dividida en dos componentes principales como se muestra en la figura 6. El

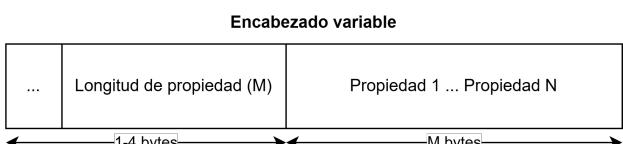


Fig. 4. Formato del encabezado variable.

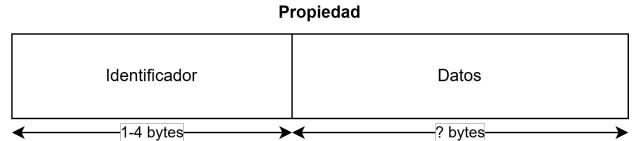


Fig. 5. Formato de la propiedad.

TABLA I
TIPOS DE PAQUETES DE CONTROL MQTT [11]

Nombre	Valor	Dirección	Carga útil
Reservado	0	Prohibido	
CONNECT	1	Cliente a servidor	Requerida
CONNACK	2	Servidor a cliente	Ninguna
PUBLISH	3	Cliente a servidor o servidor a cliente	Opcional
PUBACK	4	Cliente a servidor o servidor a cliente	Ninguna
PUBREC	5	Cliente a servidor o servidor a cliente	Ninguna
PUBREL	6	Cliente a servidor o servidor a cliente	Ninguna
PUBCOMP	7	Cliente a servidor o servidor a cliente	Ninguna
SUBSCRIBE	8	Cliente a servidor	Requerida
SUBACK	9	Servidor a cliente	Requerida
UNSUBSCRIBE	10	Cliente a servidor	Requerida
UNSUBACK	11	Servidor a cliente	Requerida
PINGREQ	12	Cliente a servidor	Ninguna
PINGRESP	13	Servidor a cliente	Ninguna
DISCONNECT	14	Cliente a servidor o servidor a cliente	Ninguna
AUTH	15	Cliente a servidor o servidor a cliente	Ninguna

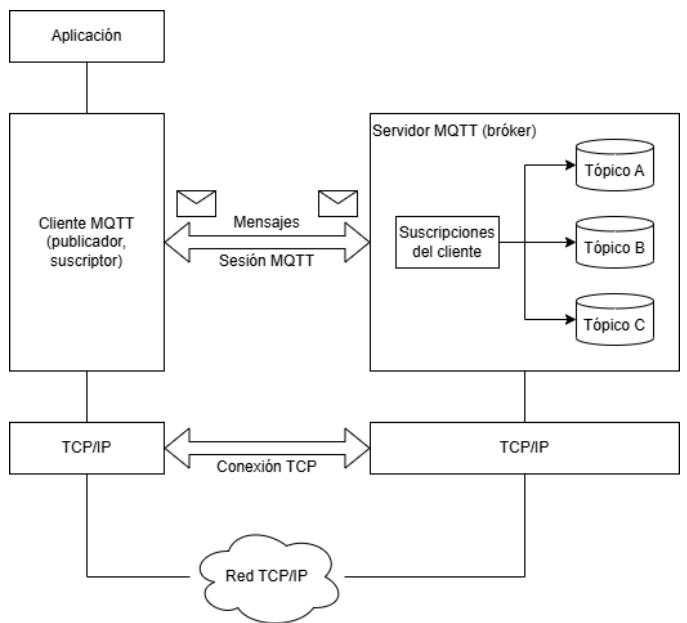


Fig. 6. Arquitectura de MQTT.

protocolo MQTT está basado en TCP/IP, lo que significa que el cliente y el bróker deben tener un *stack* TCP/IP. MQTT funciona con una arquitectura centralizada de tipo publicar-suscribir que desacopla el productor de datos (cliente publicador) del consumidor (cliente suscriptor) mediante un intermediario (bróker).

1) *Cliente*: El cliente puede ser un **publicador**, un **suscriptor** o ambos a la vez, el cual establece una conexión al servidor (bróker). Puede publicar mensajes a los usuarios interesados o suscribirse a un tópico de interés para recibir mensajes. Este puede ser cualquier dispositivo que corra una librería MQTT y se conecte a un bróker MQTT por una red [7].

Características clave del cliente:

- Puede operar como publicador, suscriptor o ambos.
- Se conecta al bróker mediante una conexión TCP/IP persistente.
- Gestiona las suscripciones y publica mensajes en tópicos específicos.
- Puede especificar el tipo de entrega de mensajes usando QoS 0, 1 o 2.

2) *Bróker*: El bróker controla la distribución de información y es principalmente responsable de recibir los mensajes del publicador, filtrarlos, decidir quiénes están interesados en ellos y luego enviarlos a todos los clientes suscritos. Puede aceptar peticiones de clientes, recibir los mensajes publicados por los usuarios, procesar diferentes peticiones como suscripciones y anulaciones de suscripciones de los usuarios y enviar mensajes a los usuarios interesados tras recibirlos del publicador [2].

Funciones principales del bróker:

- Aceptar conexiones entrantes de múltiples clientes.
- Recibir y validar los mensajes publicados por los clientes.
- Filtrar mensajes basándose en los tópicos y las suscripciones activas.
- Reenviar los mensajes recibidos a todos los clientes suscritos a los respectivos tópicos.
- Manejar peticiones de suscripción y cancelación de suscripción.
- Supervisar el estado de las conexiones para activar el mensaje Will en caso de desconexión inesperada.

E. Seguridad y conexiones

MQTT utiliza diferentes mecanismos de seguridad, pero la mayoría de ellos no está configurado o provisto por defecto como el cifrado de datos o la autenticación de entidades. Mecanismos de autenticación existen y son controlados por el bróker al registrar la información de un dispositivo una vez que trata de conectarse. La autorización de acceso puede lograrse utilizando una **lista de control de acceso** (ACL), que contiene registros de información de identificadores y contraseñas de los diferentes clientes que tienen permitido acceder a diferentes objetos y puede también especificar qué funciones puede hacer el cliente en ellos [10].

1) *Autorización de peticiones de conexión*: la autorización se refiere al proceso mediante el cual el bróker determina si

un cliente tiene permisos para realizar determinadas acciones como publicar o suscribirse a ciertos tópicos.

La figura 7 muestra el flujo de un protocolo básico durante la configuración de conexión. El primer paso en el flujo del protocolo es la obtención del token por el cliente desde el servidor de autorización (AS). El cliente y el AS deben hacer una autenticación mutua. El cliente solicita un token de acceso del AS y al recibir la solicitud, el AS lo verifica. Si el AS verifica exitosamente la solicitud del token de acceso, autoriza el cliente para la audiencia (RS) y los ámbitos indicados (permisos publicar-suscribir sobre los tópicos), el AS emite un token de acceso.

2) *Petición de conexión del cliente al bróker*: A menos que el cliente solo publique y se suscriba a tópicos públicos, el cliente y el bróker deben autenticarse mutuamente: el cliente se autentica al bróker sobre MQTT o TLS antes de hacer cualquier otra acción. Para MQTT, las opciones son “None” y “ace”. Para TLS, las opciones son “Anon” para un cliente anónimo y “Known(RPK/PSK)” para claves públicas sin procesar (RPK) y claves precompartidas (PSK) [3]. Por ejemplo, el cliente usa TLS:Anon,MQTT:None solo para los tópicos públicos que no requieren autenticación.

3) *Tópico de información de autorización*: En los casos donde el cliente debe transportar el token al bróker primero, el cliente se conecta al bróker para publicar su token al tópico “authz-info”. El tópico “authz-info” solo se publica (los clientes no pueden suscribirse) y no está protegido, así que el cliente usa la opción TLS:Anon,MQTT:None sobre una conexión TLS. Después de publicar el token, se espera que el cliente se desconecte del bróker y se reconecte usando la autenticación sobre TLS (TLS:Known(RPK/PSK),MQTT:none). El bróker almacena los índices y tókenes recibidos al tópico “authz-info”, verifica su validez y si no es válido, lo descarta [3].

F. Versiones

La última versión de MQTT es MQTT 5.0. La versión previa fue MQTT 3.1.1. MQTT 5.0 tiene mejoras en el rendimiento, confiabilidad y provee mayor control sobre la comunicación entre clientes y servidores. Los principios y características fundamentales de MQTT 3.1.1 se mantienen inalterados en MQTT 5.0. Entre los ajustes más destacados se incluyen ligeros cambios en funciones preexistentes y la inclusión de características como TTL y suscripciones compartidas. Se ha añadido un paquete de control adicional: AUTH, pero sigue conservando sus características reconocibles y mejorando sus capacidades [7]. Aquí se mencionan algunas mejoras principales en MQTT 5.0:

- Introduce un intervalo de expiración de sesión que permite a los clientes establecer un tiempo máximo permitido para una sesión.
- Ofrece un intervalo de expiración de mensaje para que los clientes administren el tiempo de vida de los mensajes.
- Los clientes pueden usar alias de tópicos en lugar del tópico completo.

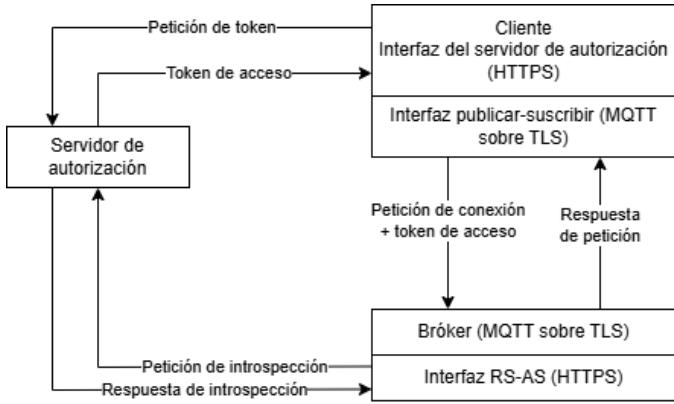


Fig. 7. Configuración de conexión.

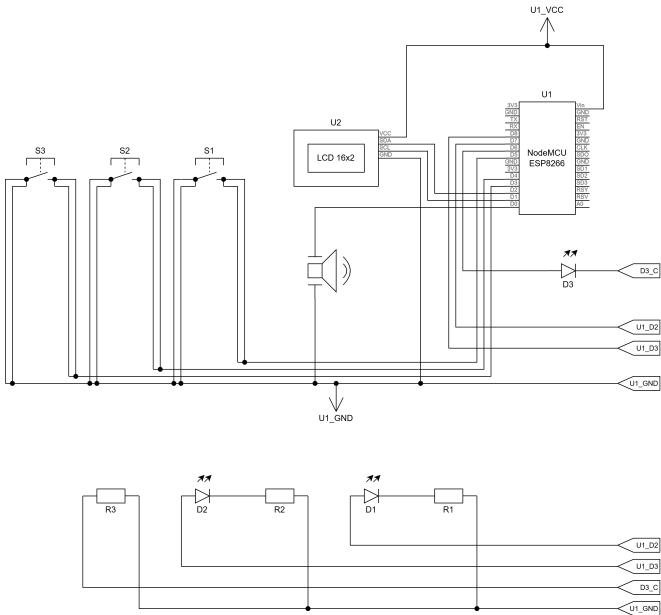


Fig. 8. Diagrama esquemático del monitor.

- Un identificador de suscripción tiene el propósito de permitir al servidor identificar y rastrear suscripciones.
- Permite el almacenamiento de valores de varias propiedades mediante llaves en propiedades del usuario.

III. COMPONENTES DEL PROYECTO

A. Bróker público

La plataforma utilizada para el bróker es HiveMQ, la cual ofrece HiveMQ Cloud, un servicio de mensajería de IoT gratuito. Las aplicaciones crean un cliente MQTT, establecen la conexión al *host* dado por la plataforma por el puerto 8883, que utiliza TLS para cifrado seguro de los datos transmitidos y se suscriben o publican en determinados tópicos según la tarea de cada una.

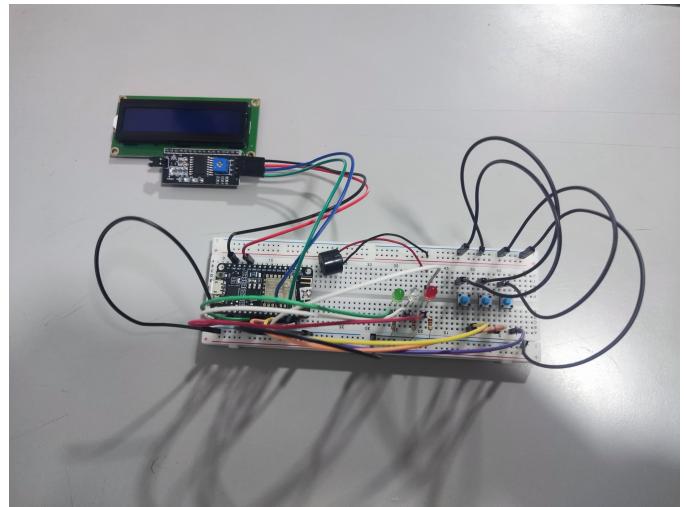


Fig. 9. Componentes ensamblados del monitor.

B. Monitor

Los componentes utilizados en el proyecto y el presupuesto se muestran en la tabla II. El diagrama esquemático se ilustra en la figura 8 y los componentes del monitor en la figura 9.

TABLA II
COMPONENTES UTILIZADOS

Componente	Cantidad	Precio (COP)
Protopboard	1	\$20 000
ESP8266 NodeMCU	1	\$18 000
LCD 16x2	1	\$12 000
Módulo I²C	1	\$7000
Buzzer	1	\$2000
Pulsador	3	\$1500
Resistencia 220 Ω	3	\$300
Led verde	1	\$200
Led blanco	1	\$200
Led rojo	1	\$200
Total		\$61 400

El monitor está programado para conectarse a Wi-Fi y suscribirse a los tópicos de `monitor01/#`, desde donde obtiene la información de la configuración y los datos de las criptomonedas y se encarga de controlar los demás componentes.

- Pulsadores: selecciona una de tres criptomonedas a rastrear.
- LCD: muestra el precio de la criptomoneda.
- Leds: indica el cambio en el precio de la criptomoneda: verde si sube, el blanco si se mantiene igual y el rojo si baja.
- Buzzer: hace sonar una alarma cuando la criptomoneda alcanza un valor límite.

1) *ESP8266*: Es un chip de bajo costo con un sistema que permite comunicarse con la red wifi utilizado en el desarrollo de aplicaciones en el IoT. NodeMCU es una placa de desarrollo que incluye el firmware que se ejecuta en el SoC Wi-Fi ESP8266 de Espressif. Puede ser programado con



Fig. 10. ESP8266 NodeMCU.



Fig. 13. Led.



Fig. 11. LCD y módulo I₂C.



Fig. 14. Buzzer.



Fig. 12. Pulsador.

Arduino IDE así como subir a internet datos desde varios sensores, o bien utilizar datos desde internet [12] como se hace en este proyecto. Su imagen se muestra en la figura 10.

2) *LCD (pantalla de cristal líquido)*: Es un dispositivo de visualización plano formado por pixeles que utiliza cristales líquidos para su funcionamiento. Debido a la cantidad limitada de pines, se utiliza además un módulo adaptador de interfaz serial I₂C que provee dos pines SDA (datos seriales) y SCL (reloj serial) para el control de la LCD. Su imagen se muestra en la figura 11.

3) *Pulsador*: Interruptor temporal que permite o interrumpe el paso de la corriente al momento de presionarlo. Su imagen se muestra en la figura 12.

4) *LED (diodo emisor de luz)*: Dispositivo electrónico que emite luz cuando por él circula corriente eléctrica. Su imagen se muestra en la figura 13.

5) *Buzzer (zumbador)*: Componente que genera sonido o un zumbido al recibir una señal eléctrica. Su imagen se

muestra en la imagen 14.

C. Configurador

El configurador es una aplicación Java que permite configurar remotamente el comportamiento del monitor ESP8266. Está construida utilizando Swing y se comunica con el bróker mediante la librería `hivemq-mqtt-client`.

Posee una interfaz gráfica para que el usuario pueda interactuar fácilmente e indicar por cada uno de los tres botones la moneda asignada y el valor límite a partir del cual emitirá la alarma, así como la duración de esta.

A continuación se muestran los tópicos a los cuales publica:

```
monitor01/
config/
button1/
    symbol
    threshold
button2/
    symbol
    threshold
button3/
    symbol
    threshold
alarmDuration
```

Un ejemplo de configuración podría ser:

- monitor01/config/button1/symbol: BTCUSDT.
- monitor01/config/button1/threshold: 103200.
- monitor01/config/button2/symbol: ETHUSDT.
- monitor01/config/button2/threshold: 2400.
- monitor01/config/button3/symbol: SOLUSDT.
- monitor01/config/button3/threshold: 160.
- monitor01/config/alarmDuration: 2.

Entonces el botón 1 sería responsable de cambiar la moneda visualizada a Bitcoin, el botón 2 a Ethereum y el botón 3 a Solana. Una alarma de dos segundos sonaría cuando el precio de Bitcoin alcance 103200, el de Ethereum 2400 o el de Solana 160 dependiendo de cuál esté seleccionada.

D. Sensor de monedas

El sensor es una aplicación Java que realiza solicitudes a la API de Binance para obtener información de las criptomonedas en tiempo real.

Antes de comenzar a publicar, primero necesita saber qué monedas debe rastrear, por lo que debe esperar a que el monitor sea configurado en un primer lugar. Una vez establecido esto desde el configurador, recibe entonces la configuración desde la suscripción al bróker y entonces publica en los tres tópicos correspondientes a las monedas indicadas cada cinco segundos.

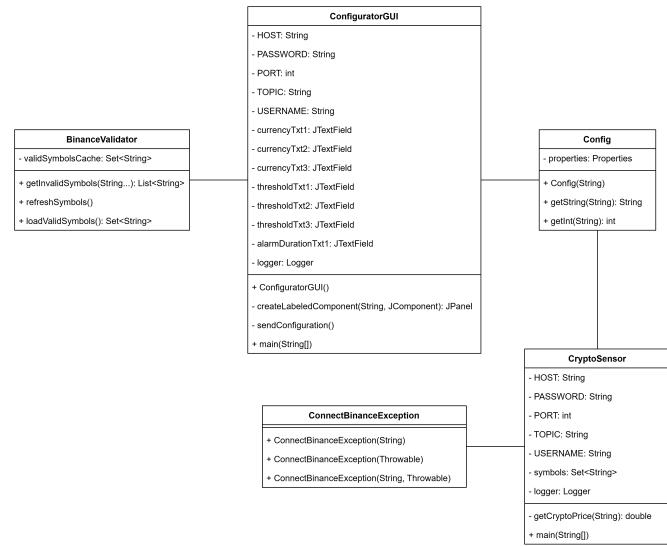


Fig. 15. Diagrama de clases.

A continuación se muestran los tópicos a los cuales publica:

```
monitor01/
crypto/
{SYMBOL1}
{SYMBOL2}
{SYMBOL3}
```

Por ejemplo, si desde el configurador se establecen los símbolos BTCUSDT, ETHUSDT y SOLUSDT, publicará los valores de estas monedas en monitor01/crypto/BTCUSDT, monitor01/crypto/ETHUSDT y monitor01/crypto/SOLUSDT. Si después se vuelven a configurar otras monedas diferentes, se dejará de publicar en estos para hacerlo en los nuevos tópicos.

En la figura 15 se ilustra el diagrama de clases de las aplicaciones Java que interactúan con el bróker a partir de una configuración `Config` dada que contiene los datos necesarios para conectarse. `ConfiguratorGUI` representa el configurador que mediante los datos ingresados por la interfaz gráfica publica los datos que luego los recibe el `CryptoSensor` para empezar a publicar los precios, los cuales recibe desde `BinanceValidator`, mientras que `ConnectBinanceException` se encarga de manejar los errores del lado del usuario y validando sus entradas.

IV. TOPOLOGÍA DE LA RED

En la figura 16 se ilustra la topología de red del proyecto organizada en una arquitectura de cliente-servidor con un bróker MQTT central (`HiveMQ`) que actúa como intermediario entre los clientes, los cuales se comunican entre sí siguiendo el patrón publicador/suscriptor.

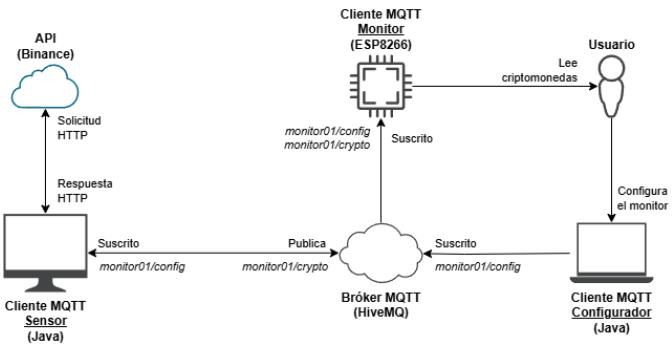


Fig. 16. Topología del proyecto.

A. Componentes de la red

1) *API (Binance)*: Fuente externa de los datos de las criptomonedas. Provee información mediante solicitudes HTTP al cliente Sensor.

2) *Cliente MQTT Sensor (Java)*: Aplicación que solicita los datos de las criptomonedas a la API de Binance vía HTTP y los publica al bróker MQTT en el tópico `monitor01/crypto/#`. Está suscrito al tópico `monitor01/config/#` para sincronizar las monedas publicadas a partir de las dadas en la configuración.

3) *Cliente MQTT Monitor (ESP8266)*: Microcontrolador que sirve de visualizador de datos. Está suscrito a los tópicos `monitor01/config/#` y `monitor01/crypto/#` para recibir las configuraciones y los valores de las criptomonedas para mostrarlos al usuario.

4) *Cliente MQTT Configurador (Java)*: Permite al usuario configurar el monitor publicando al bróker en el tópico `monitor01/config/#`.

5) *Bróker MQTT (HiveMQ)*: Componente central que enruta y distribuye los mensajes entre los clientes suscritos a los diferentes tópicos.

6) *Usuario final*: Interactúa con el sistema a través del monitor, observando los valores de las criptomonedas y configurándolo mediante el cliente configurador.

B. Principales protocolos utilizados

1) *HTTP*: Utilizado por el sensor para consultar la API de Binance.

2) *MQTT*: Utilizado para mensajería para IoT entre todos los clientes. Usa los tópicos `monitor01/config/#` y `monitor01/crypto/#` para enviar configuraciones y publicar los valores de las criptomonedas respectivamente.

C. Flujo de los datos

- 1) El usuario configura el monitor mediante el cliente configurador a través del bróker MQTT.
- 2) El cliente configurador publica la configuración al bróker.
- 3) El cliente sensor recibe la configuración y solicita los datos a Binance vía HTTP para publicarlos en el bróker MQTT.

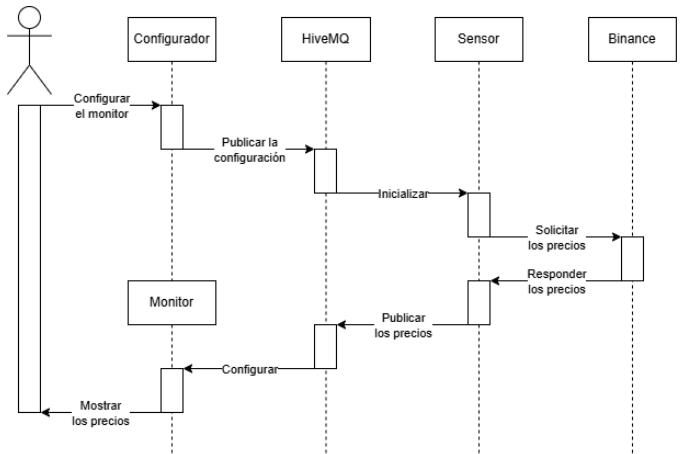


Fig. 17. Diagrama de secuencia.

```

Output  Serial Monitor x
Message (Enter to send message to 'Generic ESP8266 Module' on 'COM3')
23:00:29.247 -> 1: TERCEROS UCC_2.4G (RSSI: -89)
23:00:29.247 -> 2: HolyHome42 (RSSI: -56)
23:00:29.247 -> 3: Kevin (RSSI: -88)
23:00:29.247 -> 4: HolyHome42 (RSSI: -74)
23:00:29.247 -> 5: HolyHome42 (RSSI: -48)
23:00:29.247 -> Connecting to Wi-Fi.....
23:00:33.309 -> Connected to HolyHome42
23:00:33.309 -> IP 192.168.5.198
23:00:33.309 -> Connecting to MQTT broker as esp8266-client-EC:FA:BC:2E:95:B2...
23:00:35.141 -> Connected to MQTT broker

```

Fig. 18. Conexión al wifi y bróker.

- 4) El cliente monitor recibe los datos de las monedas desde el bróker y los muestra al usuario.
- 5) El usuario interactúa con el monitor para seleccionar las monedas.

En la figura 17 se indica el diagrama de secuencia para el proyecto.

V. PROTOCOLO DE PRUEBAS

Con el objetivo de garantizar el correcto funcionamiento del sistema, se proponen las siguientes pruebas, detallando los diferentes escenarios y casos que se evaluarán. Cada prueba incluye su propósito, procedimiento y criterio de aceptación.

A. Conexión inicial del monitor al wifi y bróker MQTT

- **Propósito:** Verificar que el monitor se conecta correctamente a la red wifi y al bróker MQTT.
- **Procedimiento:** Encender el dispositivo y observar los mensajes de depuración por puerto serial.
- **Criterio de aceptación:** El monitor debe mostrar un mensaje de conexión exitosa a la red wifi y al bróker.
- **Resultado:** El monitor establece correctamente la conexión al wifi y con el bróker de HiveMQ al momento de iniciar (figura 18).

B. Publicación de configuración desde el configurador

- **Propósito:** Verificar que el configurador publica correctamente la configuración en los tópicos MQTT.

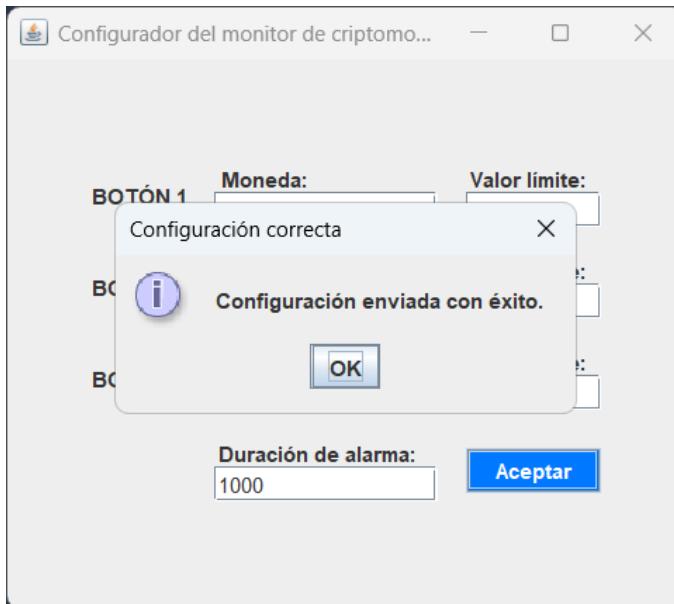


Fig. 19. Configuración correcta desde el configurador.

```
[2025-05-30 23:02:29] INFO: Publicado: monitor01/config/button1/symbol -> BTCUSDT.
[2025-05-30 23:02:29] INFO: Publicado: monitor01/config/button1/threshold -> 1.0.
[2025-05-30 23:02:29] INFO: Publicado: monitor01/config/button2/symbol -> ETHUSDT.
[2025-05-30 23:02:29] INFO: Publicado: monitor01/config/button2/threshold -> 1.0.
[2025-05-30 23:02:29] INFO: Publicado: monitor01/config/button3/symbol -> SOLUSDT.
[2025-05-30 23:02:29] INFO: Publicado: monitor01/config/button3/threshold -> 1.0.
[2025-05-30 23:02:29] INFO: Publicado: monitor01/config/alarmDuration -> 1000.
```

Fig. 20. Configuración publicada.

- Procedimiento:** Iniciar la aplicación, definir las monedas y demás configuración, y enviar los datos.
- Criterio de aceptación:** Los mensajes deben publicarse en los tópicos `monitor01/config/#` y ser recibidos por el monitor y el sensor.
- Resultado:** Al introducir los datos y presionar el botón para enviar, la configuración se aplica correctamente (figuras 19 y 20).

C. Publicación de precios desde el sensor

- Propósito:** Verificar que el sensor se conecta, recibe la configuración y publica correctamente los precios respectivos en los tópicos MQTT.
- Procedimiento:** Iniciar la aplicación, enviar la configuración, y esperar a que el sensor comience a publicar.
- Criterio de aceptación:** Los mensajes deben publicarse en los tópicos `monitor01/crypto/SYMBOL1` de acuerdo con la configuración.

```
[2025-05-30 22:50:32] INFO: Conectado al bróker MQTT.
[2025-05-30 23:02:25] INFO: Símbolo configurado para el botón 1: BTCUSDT.
[2025-05-30 23:02:25] INFO: Símbolo configurado para el botón 2: ETHUSDT.
[2025-05-30 23:02:25] INFO: Símbolo configurado para el botón 3: SOLUSDT.
[2025-05-30 23:02:27] INFO: Publicado: monitor01/crypto/ETHUSDT -> 2508.90.
[2025-05-30 23:02:28] INFO: Publicado: monitor01/crypto/SOLUSDT -> 153.69.
[2025-05-30 23:02:28] INFO: Publicado: monitor01/crypto/BTCUSDT -> 103586.05.
```

Fig. 21. Conexión al bróker y publicación de los precios.



Fig. 22. Pantalla con la moneda y su precio.

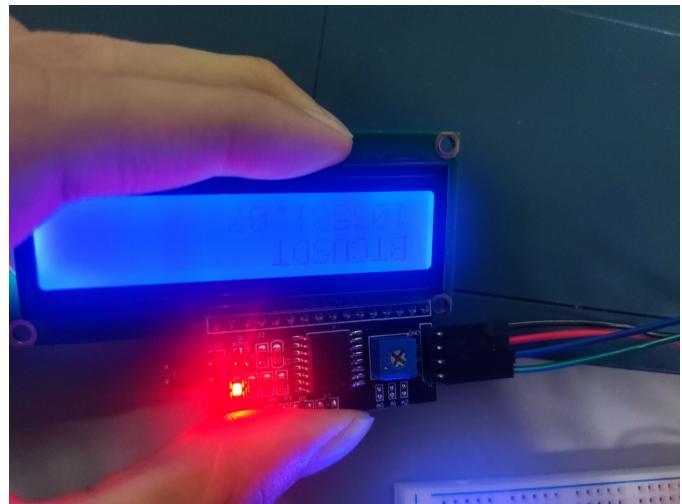


Fig. 23. Pantalla con una moneda diferente.

- Resultado:** El sensor se conecta correctamente al bróker, recibe la configuración desde el configurador y entonces publica las monedas (figura 21).

D. Visualización de precios en el monitor

- Propósito:** Verificar que el monitor recibe y muestra los precios de las criptomonedas configuradas.
- Procedimiento:** Configurar monedas en el configurador y esperar a que el sensor publique los precios.
- Criterio de aceptación:** El LCD del monitor debe mostrar el nombre de la criptomoneda seleccionada y su precio actual.
- Resultado:** La pantalla del monitor muestra correctamente el precio de la moneda configurada actualizándose cada cinco segundos (figura 22).

E. Cambio de moneda con los pulsadores

- Propósito:** Verificar que los botones cambian correctamente la criptomoneda mostrada.

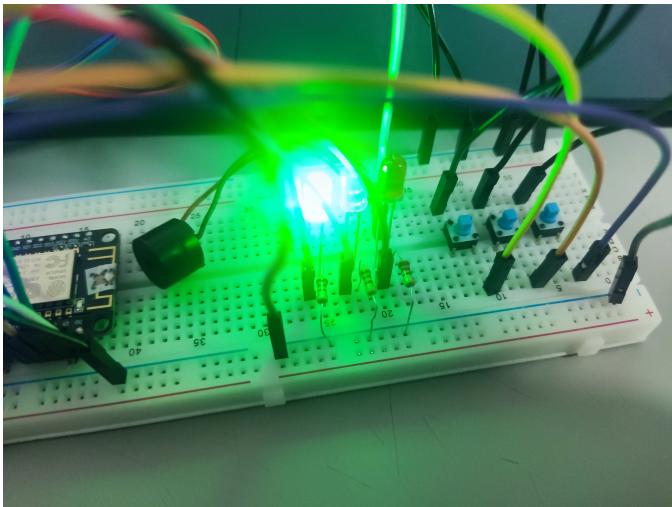


Fig. 24. Led verde encendido.

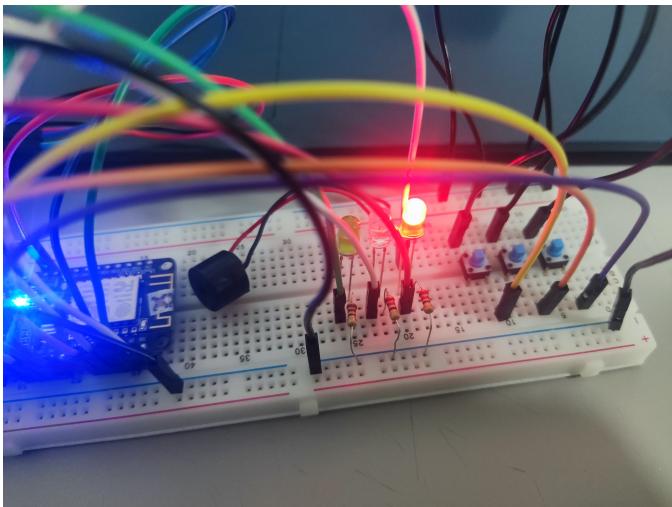


Fig. 25. Led rojo encendido.

- Procedimiento:** Presionar cada uno de los botones físicos del monitor.
- Criterio de aceptación:** El monitor debe actualizar el símbolo y mostrar el nuevo precio correspondiente al botón presionado.
- Resultado:** La pantalla del monitor cambia correctamente a la asignada al botón desde la configuración al presionarlo (figura 23).

F. Indicadores de cambio de precio con leds

- Propósito:** Confirmar que los leds reflejan la variación del precio.
- Procedimiento:** Observar los leds mientras se actualizan los precios publicados.
- Criterio de aceptación:** Enciende el led verde si el precio sube, el rojo si baja y el blanco si permanece igual.
- Resultado:** Se enciende el led verde cuando el precio es mayor que el anterior, el rojo si es menor y blanco si es

```
[2025-05-31 00:41:55] INFO: Publicado: monitor01/crypto/ETHUSDT -> 2520.80.
[2025-05-31 00:41:56] INFO: Publicado: monitor01/crypto/SOLUSDT -> 155.02.
[2025-05-31 00:41:56] INFO: Publicado: monitor01/crypto/BTCUSDT -> 103719.99.
[2025-05-31 00:41:59] INFO: Símbolo configurado para el botón 1: DOGEUSDT.
[2025-05-31 00:41:59] INFO: Símbolo configurado para el botón 2: ADAUSDT.
[2025-05-31 00:41:59] INFO: Símbolo configurado para el botón 3: ROSEUSDT.
[2025-05-31 00:42:00] INFO: Publicado: monitor01/crypto/ADAUSDT -> 0.67.
[2025-05-31 00:42:01] INFO: Publicado: monitor01/crypto/ROSEUSDT -> 0.03.
[2025-05-31 00:42:01] INFO: Publicado: monitor01/crypto/DOGEUSDT -> 0.19.
```

Fig. 26. Reconfiguración desde el sensor.

igual (figuras 24 y 25).

G. Activación de alarma sonora

- Propósito:** Verificar que la alarma sonora se activa al alcanzar el valor umbral configurado.
- Procedimiento:** Configurar un umbral por debajo del valor actual y esperar a que se supere.
- Criterio de aceptación:** El *buzzer* debe activarse por la duración configurada en el tópico `monitor01/config/alarmDuration`.
- Resultado:** El *buzzer* emite un sonido por la duración configurada cuando el precio alcanza el valor umbral.

H. Reconfiguración dinámica del sistema

- Propósito:** Comprobar que el sistema puede ser reconfigurado dinámicamente sin necesidad de reiniciar los dispositivos.
- Procedimiento:** Cambiar las monedas desde el configurador mientras el sistema está en funcionamiento.
- Criterio de aceptación:** El sensor debe dejar de publicar en los antiguos tópicos y comenzar en los nuevos; el monitor debe mostrar los valores actualizados.
- Resultado:** La configuración del monitor y los tópicos publicados se actualizan correctamente de acuerdo con la configuración (figura 26).

I. Tolerancia a fallos de red

- Propósito:** Evaluar el comportamiento del sistema ante interrupciones de red o pérdida de conexión MQTT.
- Procedimiento:** Desconectar temporalmente el acceso a internet y luego restaurarlo.
- Criterio de aceptación:** El sistema debe intentar reconectarse y recuperar el funcionamiento normal automáticamente.
- Resultado:** El sistema intenta conectarse a la red nuevamente y al bróker.

J. Tolerancia ante valores erróneos

- Propósito:** Validar que el sistema maneja correctamente valores mal formateados (p. ej., campos vacíos, monedas no válidas, precios negativos, texto en vez de números).
- Procedimiento:** Configurar valores incorrectos e intentar enviarlos en los tópicos esperados.
- Criterio de aceptación:** El sistema debe rechazar los valores inválidos y mantener su funcionamiento normal.
- Resultado:** El configurador posee las validaciones suficientes de las entradas, no permitiendo enviar los valores incorrectos descritos (figuras 27, 28 y 29).

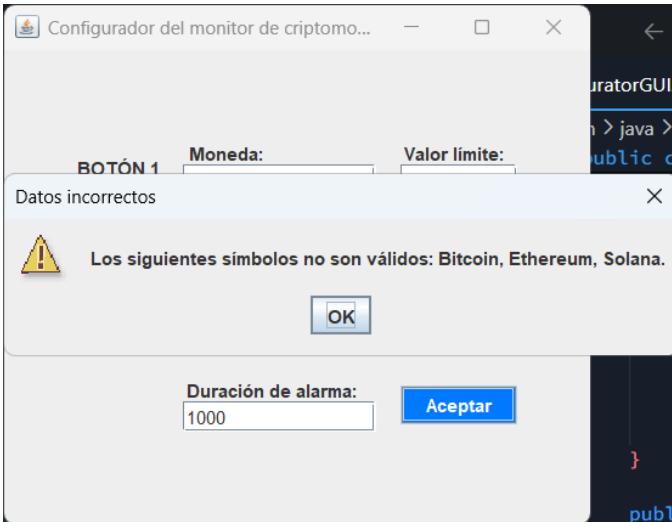


Fig. 27. Intento de envío de símbolos inválidos.

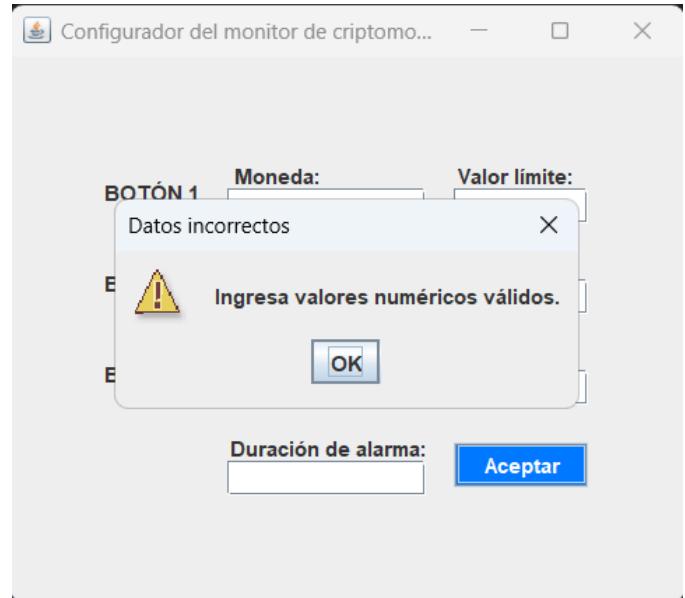


Fig. 29. Intento de envío de campos vacíos.

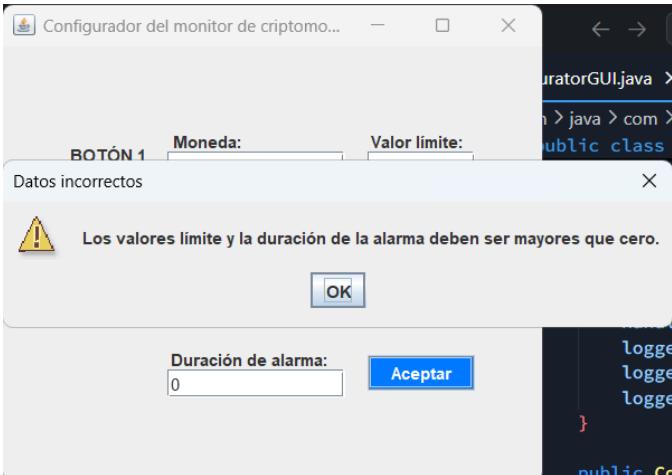


Fig. 28. Intento de envío de valores numéricos negativos o cero.

VI. CONCLUSIONES

REFERENCIAS

- [1] M. B. Yassein, M. Q. Shatnawi, S. Aljwaneh and R. Al-Hatmi, "Internet of Things: Survey and open issues of MQTT protocol," in 2017 International Conference on Engineering & MIS (ICEMIS), Monastir, Tunisia, 2017.
- [2] D. Soni, A. Makwana, "A Survey on MQTT: A Protocol of Internet of Things (IoT)," in International Conference on Telecommunication, Power Analysis and Computing Techniques (ICTPACT), vol. 20, Apr. 2017.
- [3] C. Sengul, A. Kriby, "Message Queuing Telemetry Transport (MQTT) and Transport Layer Security (TLS) Profile of Authentication and Authorization for Constrained Environments (ACE) Framework," RFC 9431, Jul. 2023.
- [4] G. C. Kessler, "An Overview of TCP/IP Protocols and the Internet," Nov. 2010. <https://www.garykessler.net/library/tcpip.html>.
- [5] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol," Network Working Group, May. 1996.
- [6] S. Quinoz, T. Emilio, J. Kazienko, "MQTT Protocol: Fundamentals, Tools and Future Directions," in IEEE Latin America Transactions, vol. 17, no. 9, Sep. 2019.
- [7] "MQTT Essentials The Ultimate Guide to the MQTT Protocol for IoT Messaging," HiveMQ.

- [8] "IoT Portal API Reference Guide," Tellit Cinterion, Sept. 2023.
- [9] "MQTT Control Packets: A Beginner's Guide," EMQX, Jul. 2023. <https://www.emqx.com/en/blog/introduction-to-mqtt-control-packets>.
- [10] D. Dinculeană, X. Cheng, "Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices," Applied Sciences, vol. 9, no. 5, p. 848, 2019.
- [11] A. Banks, E. Briggs, K. Borgendale, R. Gupta, "MQTT Version 5.0," OASIS Standard, Mar. 2019. <https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html>.
- [12] Z. Wan, Y. Song, Z. Cao, "Environment Dynamic Monitoring and Remote Control of Greenhouse with ESP8266 NodeMCU," School of Electronics and Information Engineering, Beijing, Jiaotong University, 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, 2019.