INSTITUTO POLITÉCNICO NACIONAL
Escuela Superior de Cómputo

Tarea 01

CLIENTE MQTT

Materia || **Embedded Systems - IoT**
Profesor || **Miguel Ángel Alemán Arce**
Alumno || **Areli Alejandra Guevara Badillo**

VIERNES, ABRIL 04, 2025

INTRODUCCIÓN

En el contexto de la ingeniería en sistemas computacionales, el diseño de soluciones eficientes para el Internet de las Cosas (IoT) requiere protocolos de comunicación adaptados a entornos con recursos limitados. El protocolo **MQTT (Message Queuing Telemetry Transport)** destaca por su ligereza y bajo consumo energético, características que lo hacen ideal para dispositivos embebidos como microcontroladores. Este informe aborda la implementación de **MQTTX**, una herramienta cliente MQTT de código abierto, como recurso pedagógico en proyectos académicos. Su análisis se centra en cómo esta herramienta facilita la comprensión de conceptos clave de IoT, desde la comunicación máquina a máquina (M2M) hasta la gestión de brokers, en un entorno universitario.

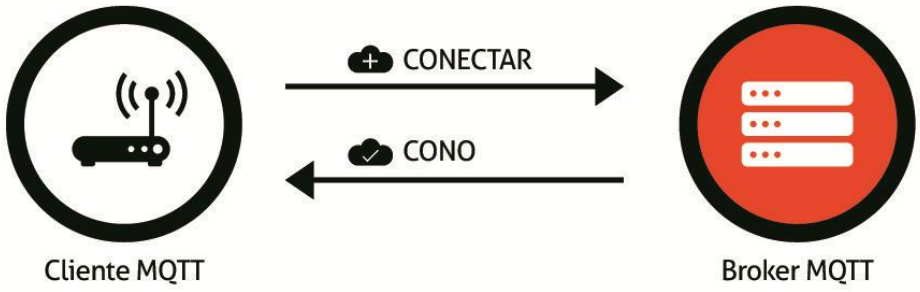
FUNDAMENTOS TÉCNICOS DEL PROTOCOLO MQTT

El protocolo MQTT opera bajo un modelo publicador-suscriptor, donde los dispositivos no se comunican directamente entre sí, sino a través de un intermediario denominado broker.

Arquitectura Básica

El protocolo MQTT sigue un modelo publicador-suscriptor con tres componentes clave:

- **Broker**: Actúa como servidor central, recibiendo mensajes de los clientes publicadores y distribuyéndolos a los suscriptores según los tópicos definidos. Ejemplos populares incluyen Mosquitto y HiveMQ.
- **Cliente Publicador**: Dispositivo o aplicación que envía datos a un tópico específico. Por ejemplo, un sensor de temperatura integrado en un ESP32.
- **Cliente Suscriptor**: Entidad que recibe información al suscribirse a un tópico. Aquí, MQTTX se posiciona como una herramienta clave para visualizar estos datos en tiempo real.



Mecanismos de Operación

- **Tópicos (Topics)**: Los mensajes se organizan en canales jerárquicos, como edificioA / sala1 / temperatura, permitiendo una gestión estructurada de la información.
- **Calidad de Servicio (QoS)**: MQTT ofrece tres niveles de entrega de mensajes:
    - **QoS 0**: Entrega "como máximo una vez" (sin confirmación).
    - **QoS 1**: Entrega "al menos una vez" (con confirmación de recepción).

- **QoS 2**: Entrega "exactamente una vez" (mecanismo de handshake para evitar duplicados).

- **Last Will and Testament (LWT)**: Notifica a otros clientes si un dispositivo se desconecta abruptamente, útil para monitorear fallos en sistemas críticos.

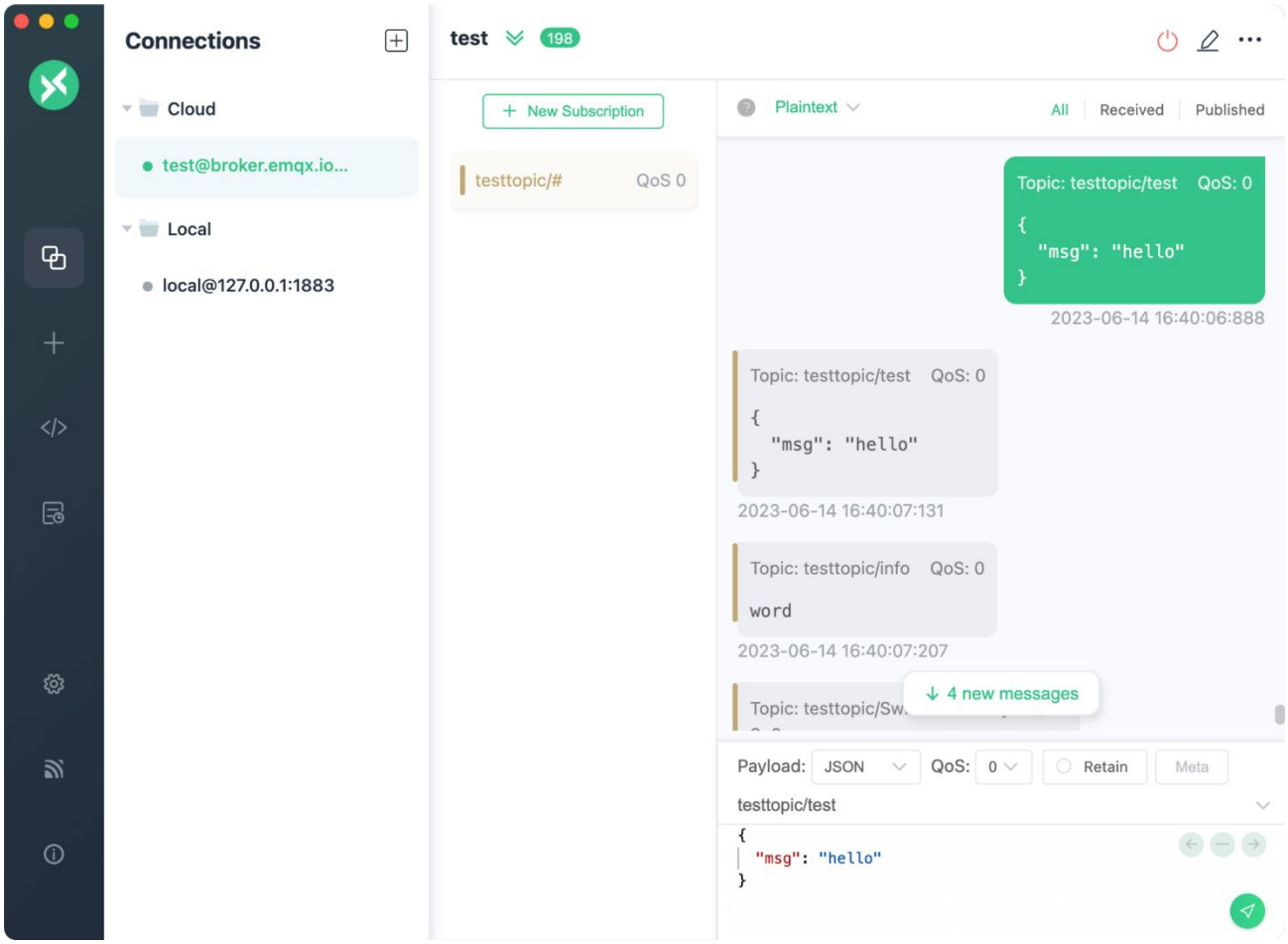
Ventajas para Sistemas Embebidos

Esta arquitectura es especialmente ventajosa en redes con latencia variable o ancho de banda reducido, común en aplicaciones IoT, por los siguientes rezones:

- **Bajo ancho de banda**: Cabeceras de mensaje mínimas (2 bytes).
- **Calidad de Servicio (QoS)**: Garantiza entrega de mensajes (niveles 0, 1 y 2).
- **Manejo de desconexiones**: Mecanismo *Last Will and Testament* (LWT) para notificar fallos.

ANÁLISIS DE MQTTX COMO HERRAMIENTA EDUCATIVA

MQTTX (https://mqttx.app/) es una aplicación cliente MQTT multiplataforma desarrollada por EMQ Technologies. Su diseño intuitivo y versatilidad la convierten en un recurso valioso para estudiantes de sistemas computacionales que buscan explorar IoT sin incurrir en complejidades técnicas innecesarias.



Funcionalidades Relevantes

La herramienta ofrece una interfaz gráfica de usuario (GUI) que simplifica la conexión a brokers MQTT, la suscripción a tópicos y la visualización de mensajes. Entre sus características destacadas se encuentran:

- **Interfaz gráfica intuitiva:** Permite gestionar múltiples conexiones simultáneas.
- **Soporte para MQTT 5.0 y 3.1.1**: Permite trabajar con las versiones más recientes del protocolo, así como con estándares legacy.
- **Formatos de Mensaje Flexibles**: Los datos pueden enviarse y recibirse en JSON, texto plano, Hex o Base64, adaptándose a diversos casos de uso.
- **Conexiones Seguras**: Integra soporte para TLS/SSL y autenticación mediante certificados X.509, esencial para proyectos que requieren seguridad robusta.
- **Herramientas de Diagnóstico**: Incluye un historial de mensajes, opciones para exportar datos en formatos CSV o JSON, y la capacidad de simular múltiples clientes simultáneamente.

Comparación con Otras Herramientas

Aunque existen alternativas como la línea de comandos de Mosquitto (mosquitto_pub y mosquitto_sub) o plataformas de flujo como Node-RED, **MQTTX** se distingue por su enfoque pedagógico. A diferencia de las herramientas CLI, que requieren familiaridad con terminales y scripts, MQTTX reduce la barrera de entrada para estudiantes principiantes. Por otro lado, mientras Node-RED ofrece integración visual avanzada, su curva de aprendizaje es más pronunciada, especialmente para quienes no tienen experiencia previa en programación de flujos.

Herramienta	Ventajas	Limitaciones
MQTTX	GUI amigable, ideal para depuración visual.	No integra capacidades de broker.
Mosquitto CLI	Ligero, útil para scripts automatizados.	Curva de aprendizaje para no expertos.
Node-RED	Integración con flujos visuales complejos.	Requiere configuración adicional.

EVALUACIÓN CRÍTICA DE MQTTX

Ventajas en el Aula

- **Accesibilidad:** Ideal para estudiantes sin experiencia previa en MQTT y fácil de conseguir.
- **Reducción de la Complejidad**: Al abstraer la configuración de conexiones MQTT en una interfaz gráfica, los estudiantes pueden enfocarse en conceptos fundamentales sin distraerse con detalles de implementación.
- **Prototipado Rápido**: Facilita la iteración ágil en proyectos, permitiendo probar ideas en minutos en lugar de horas.

- **Versatilidad en Casos de Uso**: Desde monitoreo de sensores hasta control de actuadores, MQTTX se adapta a múltiples disciplinas dentro de IoT, como domótica o agricultura de precisión.

Limitaciones y Desafíos

- **Dependencia de un Broker Externo**: Aunque brokers públicos como test.mosquitto.org son útiles para pruebas iniciales, su uso en proyectos críticos es desaconsejable debido a limitaciones de rendimiento y seguridad.
- **Curva de Aprendizaje en Seguridad**: Configurar TLS/SSL o certificados X.509 requiere conocimientos avanzados que pueden exceder el alcance de cursos introductorios.
- **Hardware Limitado**: Dispositivos sin conectividad Wi-Fi integrada, como Arduino Uno, necesitan módulos adicionales (ej: ESP8266) para usar MQTT, incrementando costos y complejidad.

RECOMENDACIONES PARA SU IMPLEMENTACIÓN CURRICULAR

1. **Fase de Introducción**:
    - Emplear brokers públicos gratuitos (test.mosquitto.org) para prácticas iniciales, enfocándose en la publicación y suscripción básica.
    - Utilizar QoS 0 para proyectos sencillos, evitando sobrecargar a los estudiantes con mecanismos de confirmación.
2. **Fase Intermedia**:
    - Implementar un broker local con Mosquitto en una máquina virtual o contenedor Docker, permitiendo a los alumnos gestionar su propia infraestructura.
    - Introducir autenticación básica (usuario/contraseña) y discutir buenas prácticas en seguridad.
3. **Fase Avanzada**:
    - Integrar MQTTX con Node-RED para crear dashboards interactivos, combinando visualización de datos y automatización.
    - Desarrollar proyectos multidisciplinarios, como un sistema de riego automático controlado mediante mensajes MQTT.
4. **Evaluación de Proyectos:**
    - Diseñar retos prácticos, como crear un sistema de alertas por correo electrónico usando MQTT + IFTTT.

CONCLUSIONES

La incorporación de **MQTTX** en el currículo de sistemas computacionales ofrece una oportunidad única para acercar a los estudiantes al ecosistema IoT de manera práctica y accesible. Su capacidad para simplificar la interacción con brokers MQTT, junto con su soporte para estándares modernos, la convierten en una herramienta indispensable para laboratorios universitarios. Aunque enfrenta desafíos en términos de seguridad y dependencia de hardware, estos pueden abordarse mediante una planificación curricular

estructurada que equilibre teoría y práctica. Al finalizar un curso que integre MQTTX, los alumnos no solo habrán dominado un protocolo clave en IoT, sino que también habrán desarrollado habilidades en depuración, integración de sistemas y gestión de datos, competencias críticas en la industria tecnológica actual.

REFERENCIAS

- **EMQ Technologies**. (2023). *MQTTX Documentation*. https://mqttx.app/docs
- **HiveMQ**. (2023). *MQTT Essentials*. https://www.hivemq.com/mqtt-essentials/
- **OASIS**. (2019). *MQTT Version 5.0*. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
- **Light, R. A.** (2017). Mosquitto: Server and client implementation of the MQTT protocol. *Journal of Open Source Software*, *2*(13), 265. https://doi.org/10.21105/joss.00265
- **Arduino**. (2023). *PubSubClient library*. GitHub repository. https://github.com/knolleary/pubsubclient