GUÍA PASO A PASO DE LABORATORIO — MÓDULO 05

> **Despliegue de Broker Eclipse Mosquitto en Raspberry Pi, Endurecimiento de Seguridad y Puente Bidireccional Modbus RTU ↔ MQTT**

1. Objetivo General del Laboratorio

Implementar una arquitectura completa de mensajería industrial **Publicador/Suscriptor (MQTT v3.1.1 / v5.0)** sobre una **Raspberry Pi 4 Model B / 5**, integrando el nodo de campo **Arduino UNO R3 (Esclavo Modbus RTU ID=1)** construido en el Módulo 04 hacia un **Broker Eclipse Mosquitto 2.x** asegurado con autenticación por contraseña cifrada (`SHA-512`), listas de control de acceso por tópico (**ACL**) y un servicio demonio en Python 3 (`puente_modbus_a_mqtt_edge.py`) con exclusión mutua (`threading.Lock`), mensajes retenidos (**Retained**) y testamento de última voluntad (**LWT**).

2. Arquitectura de Red y Mapa de Tópicos del Laboratorio

Parámetro de Red / Serial	Valor de Producción en Laboratorio
Interfaz Serial RS-485 en Raspberry Pi	`/dev/ttyUSB0` (9600 bps, formato `8N1` , Half-Duplex)
Dirección Esclavo Modbus RTU (Arduino UNO)	`ID = 1` (Registros Holding `40001` a `40005`)
Dirección IP del Broker Mosquitto (RPi)	`127.0.0.1` (local) / `192.168.1.100` (LAN Planta)
Puertos TCP del Broker	`1883` (LAN autenticada) / `8883` (TLS 1.3 con X.509)
Raíz Jerárquica de Tópicos (ISA-95)	`invernadero/zona_norte/nodo_01/`
Tópico de Estado y Testamento (LWT)	`invernadero/zona_norte/nodo_01/estado/conexion` (`QoS 1, Retain=True`)
Tópico de Telemetría Agregada JSON	`invernadero/zona_norte/nodo_01/estado/resumen` (`QoS 1, Retain=False`)
Tópico de Mando Descendente (Actuadores)	`invernadero/zona_norte/nodo_01/actuadores/comando` (`QoS 1, Retain=False`)
Tópico de Confirmación de Escritura `0x06`	`invernadero/zona_norte/nodo_01/actuadores/confirmacion` (`QoS 1, Retain=False`)

3. Fase 1: Instalación del Broker Eclipse Mosquitto 2.x en Raspberry Pi OS

Conéctese por SSH a la Raspberry Pi y ejecute la actualización de repositorios e instalación tanto del servidor demonizado (`mosquitto`) como de las herramientas de diagnóstico por línea de comandos (`mosquitto-clients`):

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y mosquitto mosquitto-clients python3-pip python3-venv
sudo systemctl enable mosquitto
```

Verifique que el binario del servidor corresponde a la serie **Eclipse Mosquitto 2.x**, la cual por diseño de ciberseguridad **restringe el tráfico no autenticado a la interfaz `loopback` (`127.0.0.1`)** hasta que se configure explícitamente el archivo de seguridad:

```
mosquitto -h | head -n 3
sudo systemctl status mosquitto
```

---

4. Fase 2: Creación de Cuentas Cifradas (`mosquitto_passwd`) y Listas ACL

Nunca opere un broker industrial con `allow_anonymous true`. Crearemos tres identidades con privilegios segregados según el principio de mínimo privilegio:

- `puente_edge`: Cuenta exclusiva del demonio Python en la Raspberry Pi.
- `scada_central`: Cuenta del sistema supervisor / Node-RED con permiso de mando.
- `monitor_operador`: Cuenta de solo lectura para paneles de visualización.

```
# 1. Crear archivo de contraseñas cifradas con hash SHA-512 + Salt (-c crea el archivo)
sudo mosquitto_passwd -c /etc/mosquitto/passwd puente_edge
sudo mosquitto_passwd /etc/mosquitto/passwd scada_central
sudo mosquitto_passwd /etc/mosquitto/passwd monitor_operador

# 2. Copiar los archivos de configuración industrial y ACL del material complementario
sudo cp mosquitto_industrial.conf /etc/mosquitto/conf.d/mosquitto_industrial.conf
sudo cp acl_invernadero.conf /etc/mosquitto/acl_invernadero.conf

# 3. Asegurar permisos estrictos de propietario para que Mosquitto pueda leerlos
sudo chown mosquitto:mosquitto /etc/mosquitto/passwd /etc/mosquitto/acl_invernadero.conf
sudo chmod 0600 /etc/mosquitto/passwd /etc/mosquitto/acl_invernadero.conf

# 4. Reiniciar el broker y verificar que escucha en 0.0.0.0:1883
sudo systemctl restart mosquitto
sudo ss -tulpn | grep 1883
```

---

5. Fase 3: Verificación del Árbol de Tópicos, QoS y Retención por CLI

Abra **dos terminales SSH** en la Raspberry Pi para comprobar el enrutamiento y los permisos ACL antes de conectar el bus físico:

Terminal 1 (Suscriptor Supervisor con comodín multinivel `#`):

```
mosquitto_sub -h localhost -p 1883 \
  -u "scada_central" -P "TuClaveSeguraSCADA" \
  -t "invernadero/zona_norte/nodo_01/#" -q 1 -v
```

Terminal 2 (Prueba de publicación retenida y prueba de bloqueo ACL):

```
# Publicar configuración retenida (-r) con QoS 1 (-q 1)
mosquitto_pub -h localhost -p 1883 \
  -u "puente_edge" -P "ClaveSeguraIndustrial2026" \
  -t "invernadero/zona_norte/nodo_01/configuracion/limites" \
  -m '{"consigna_temp_c": 27.5, "histeresis_c": 1.0}' -q 1 -r

# Verificar que monitor_operador NO puede publicar comandos (ACL deniega escritura)
mosquitto_pub -h localhost -p 1883 \
  -u "monitor_operador" -P "ClaveOperador" \
  -t "invernadero/zona_norte/nodo_01/actuadores/comando" \
  -m '{"ventilador": 1}' -q 1
```

---

6. Fase 4: Despliegue del Puente Bidireccional Python (`Modbus RTU ⇔ MQTT`)

- Agregue el usuario `pi` al grupo `dialout` para acceso directo al conversor USB-RS485 (`/dev/ttyUSB0`):

  ```bash
  sudo usermod -a -G dialout pi
  ```

- Instale las bibliotecas de comunicación industrial en un entorno virtual o del sistema:

  ```bash
  pip3 install minimalmodbus pyserial paho-mqtt
  ```

- Copie `puente_modbus_a_mqtt_edge.py` a `/opt/iot_invernadero/` y ejecútelo inicialmente en primer plano para observar el ciclo de sondeo cada $2.0\ s$ (`Función 0x03`) y el mensaje de nacimiento (**Birth Message**):

  ```bash
  sudo mkdir -p /opt/iot_invernadero
  sudo cp puente_modbus_a_mqtt_edge.py /opt/iot_invernadero/
  sudo chown -R pi:pi /opt/iot_invernadero
  python3 /opt/iot_invernadero/puente_modbus_a_mqtt_edge.py
  ```

---

7. Fase 5: Prueba de Mando Bidireccional (`MQTT → Mutex → Modbus 0x06 → Relé`)

Con el puente en ejecución y el suscriptor abierto en la Terminal 1, envíe desde otra consola una orden JSON para encender el extractor forzado y actualizar la consigna térmica a $26.0\ ^\circ C$:

```
mosquitto_pub -h localhost -p 1883 \
  -u "scada_central" -P "TuClaveSeguraSCADA" \
  -t "invernadero/zona_norte/nodo_01/actuadores/comando" \
  -m '{"ventilador": 1, "bomba_riego": 0, "consigna_c": 26.0}' -q 1
```

Comportamiento esperado:

- El callback asíncrono `al_recibir_mensaje` adquiere `candado_serial` (`threading.Lock()`), esperando a que finalice cualquier lectura `0x03` en curso (~$32\ ms$).
- Envía la trama **Modbus** `0x06` **(** `Write Single Register` **)** al registro `40003` (valor `1`) y al registro `40005` (valor `260` , coma fija $\times\ 10$).
- El Arduino UNO R3 conmuta físicamente el **Relé K1** en la protoboard, encendiendo el **motor DC del ventilador extractor**, y devuelve el eco Modbus `0x06` .
- El puente libera `candado_serial` y publica en `.../actuadores/confirmacion` el objeto JSON con `"resultado": "OK_MODBUS_0x06"` .

---

8. Fase 6: Demonización 24/7 con `systemd` y Prueba de Testamento (LWT)

Instale la unidad `modbus-mqtt-bridge.service` para arranque automático y recuperación ante fallos:

```
sudo cp modbus-mqtt-bridge.service /etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable modbus-mqtt-bridge.service
sudo systemctl start modbus-mqtt-bridge.service
sudo journalctl -u modbus-mqtt-bridge.service -f
```

Para validar el **Testamento de Última Voluntad (LWT)**, simule una caída abrupta del proceso enviando la señal `SIGKILL` (`kill -9`), la cual impide enviar el paquete `DISCONNECT` :

```
sudo kill -9 $(pgrep -f puente_modbus_a_mqtt_edge.py)
```

Tras $T_{LWT} = 1.5 \times T_{keepalive}$ (o inmediatamente al cerrarse el descriptor TCP del kernel), el Broker Mosquitto publicará automáticamente `"DESCONECTADO_INESPERADO"` con bandera retenida en `invernadero/zona_norte/nodo_01/estado/conexion`, y `systemd` relanzará el puente a los $5\ s$.