IoT Mandatory Assignment 02 - MQTT

This repository contains a solution for the IKT520 MQTT Mandatory Assignment (MA-02), implementing various aspects of the MQTT protocol using the `paho-mqtt` library in Python.

Source code for this assignment can be found here: https://github.com/yngvemag/ikt520-mqtt-assignment

The README.md file in this repository provides a detailed description of the solution, and can be used as a guide for how to setup and test the solution.

1. Creating MQTT Clients

From `ma-02-solution.py`, the key arguments used are:

```
# Create publisher
publisher = mqtt.Client(
    client_id=f"publisher-{uuid4().hex[:8]}",  # Generate unique ID
    clean_session=True                         # New session each time
)

# Create subscriber
subscriber = mqtt.Client(
    client_id=f"subscriber-{uuid4().hex[:8]}",  # Generate unique ID
    clean_session=True                          # New session each time
)
```

- **client_id**: Unique identifier (e.g., "publisher-1", "subscriber-1")
- **broker_host**: MQTT broker address (localhost for EMQX)
- **broker_port**: MQTT port (1883 for unencrypted MQTT)
- **clean_session**: Controls session persistence
- **keep_alive**: Maximum time (seconds) between messages

---

2. Publisher Connection

From `ma-02-solution.py`, the on_connect callback receives:

```
def on_connect(client: mqtt.Client,
               userdata: Any,
               flags: Dict[str, bool],
               rc: int) -> None:
    """
    Callback when client connects to broker.
    """
    if rc == 0:
        print(f"Connected to broker with result code {rc}")
        session_present = flags.get('session_present', False)
        print(f"Session present flag: {session_present}")
    else:
        print(f"Failed to connect: {rc}")
```

Arguments:

- **client**: Client instance that connected
- **userdata**: Custom user data (if set)
- **flags**: Dictionary containing session_present flag
- **rc**: Connection result (0 = success)

---

3. Subscriber Connection

From `ma-02-solution.py`:

```
def on_subscribe(client: mqtt.Client,
                 userdata: Any,
                 mid: int,
                 granted_qos: List[int]) -> None:
    """
    Callback when broker confirms subscription.
    """
    print(f"Subscribed with message ID {mid}, granted QoS: {granted_qos}")
```

The subscriber subscribes to "CyberSec/IKT520" and receives SUBACK confirmation through these parameters.

---

4. Publishing Messages

From `ma-02-solution.py`:

```
info = publisher.publish(
    topic=topic,
    payload=payload,
    qos=qos,
    retain=retain
)
```

Arguments:

- **topic**: Message destination
- **payload**: Message content
- **qos**: Quality of Service (0, 1, or 2)
- **retain**: Whether broker should store message

---

5. Wildcard Subscriptions

From `ma-02-solution.py`:

```
# Single-level wildcard
client.subscribe("Sensors/+/Temperature", qos=1)
print("Subscribed to: Sensors/+/Temperature")

# Multi-level wildcard
client.subscribe("Sensors/#", qos=1)
print("Subscribed to: Sensors/#")
```

- `+` matches exactly one level (e.g., "Sensors/Living/Temperature")
- `#` matches multiple levels (e.g., "Sensors/Living/Temperature/Indoor")

---

6. Persistent Session (QoS 1)

From `ma-02-solution.py`:

```
# Create subscriber with persistent session
subscriber = mqtt.Client(
    client_id=client_id,
    clean_session=False  # Persistent session
)

# Subscribe with QoS 1
client.subscribe("Sensor/Temp", qos=1)
```

Observation: Subscriber received all 20 messages after reconnection.

Reason:

- clean_session=False creates persistent session
- QoS 1 messages are stored for offline clients
- Broker delivers stored messages upon reconnection

---

7. Non-persistent Session (QoS 1)

From `ma-02-solution.py`:

```
# Create subscriber with clean session
subscriber = mqtt.Client(
    client_id=client_id,
    clean_session=True  # Non-persistent session
)

# Always subscribe since it's a clean session
client.subscribe("Sensor/Temp", qos=1)
```

Observation: No messages received after reconnection.

Reason:

- clean_session=True discards session state
- No message storage for offline clients
- New session starts clean on reconnection

---

8. Mixed QoS with Persistent Session

From `ma-02-solution.py`:

```
# Create subscriber with persistent session
subscriber = mqtt.Client(
    client_id=client_id,
    clean_session=False  # Persistent session
)

# Subscribe with QoS 0
client.subscribe("Sensor/Temp", qos=0)

# Publish with QoS 2
publisher.publish("Sensor/Temp", f"Temperature reading {i}", qos=2)
```

Observation: No messages received despite QoS 2 publishing.

Reason:

- QoS 0 subscription doesn't support message storage
- Subscription QoS (0) overrides publish QoS (2)
- Even with persistent session, QoS 0 messages aren't stored