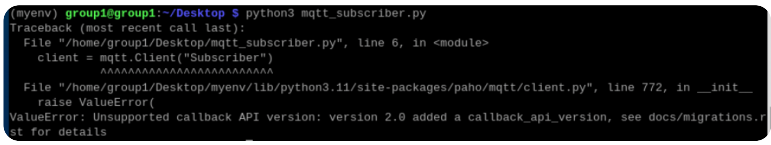


what error is this?

my code:
import paho.mqtt.client as mqtt

def on_message(client, userdata, message):
    print(f"Received message '{message.payload.decode()}' on topic '{message.topic}'")

client = mqtt.Client("Subscriber")
client.on_message = on_message
client.connect("localhost", 1883)
client.subscribe("test/topic")
client.loop_forever()

The error message:

pgsql Copy Edit

```
ValueError: Unsupported callback API version: version 2.0 added a callback_api_version, see docs/migrations.rst for details
```

indicates that the `paho-mqtt` library has introduced a change in its API requiring the specification of a `callback_api_version` in the `mqtt.Client` constructor.

Fix: Specify `callback_api_version`

Modify the `mqtt.Client` instantiation in your script as follows:

python Copy Edit

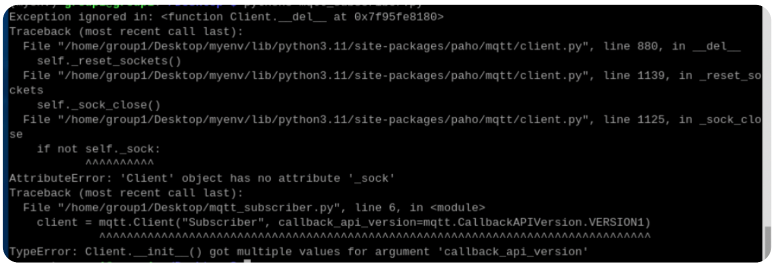
```python
import paho.mqtt.client as mqtt

def on_message(client, userdata, message):
    print(f"Received message '{message.payload.decode()}' on topic '{message.topic}'")

# Add `callback_api_version=mqtt.CallbackAPIVersion.VERSION1`
```

```
client = mqtt.Client("Subscriber",
callback_api_version=mqtt.CallbackAPIVersion.VERSION1)
client.on_message = on_message
client.connect("localhost", 1883)
client.subscribe("test/topic")
client.loop_forever()
```

Explanation

- The recent versions of `paho-mqtt` require explicitly specifying the `callback_api_version` because they introduced a new API version (Version 2.0).
- By setting `callback_api_version=mqtt.CallbackAPIVersion.VERSION1`, you ensure compatibility with the older callback API.

If you're using a newer version and want to migrate to API version 2.0, you will need to update your callbacks accordingly. Let me know if you need help with that! 🚀



The error message:

css Copy Edit

```css
TypeError: Client.__init__() got multiple values for argument
'callback_api_version'
```

indicates that the `mqtt.Client` constructor does not accept `callback_api_version` as a named argument in the way you specified.

Fix: Use `mqtt.Client` Without the Name Argument

Try modifying your code as follows:

python Copy Edit

```python
import paho.mqtt.client as mqtt

def on_message(client, userdata, message):
    print(f"Received message '{message.payload.decode()}' on topic
```

```
'{message.topic}'")

# Use named argument explicitly
client = mqtt.Client(client_id="Subscriber",
callback_api_version=mqtt.CallbackAPIVersion.VERSION1)
client.on_message = on_message
client.connect("localhost", 1883)
client.subscribe("test/topic")
client.loop_forever()
```

Explanation:

1. The `mqtt.Client` constructor expects `callback_api_version` as a **keyword argument**.
2. The first positional argument in `mqtt.Client("Subscriber", ...)` was treated as `client_id`, causing a conflict when `callback_api_version` was also added.
3. To fix this, explicitly specify `client_id="Subscriber"` instead of passing it positionally.

Alternative: Check Installed Version

If the error persists, your installed version of `paho-mqtt` might be outdated or incompatible. You can check your version by running:

sh Copy Edit

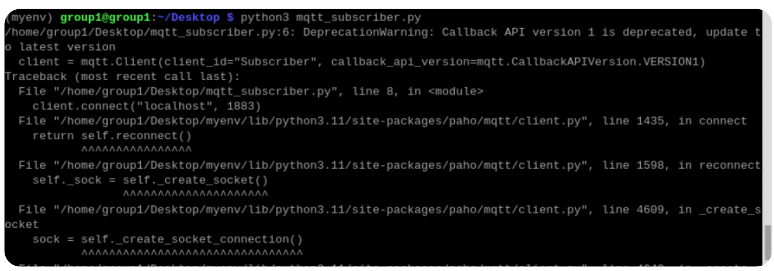
```
pip show paho-mqtt
```

If it's outdated, upgrade it:

sh Copy Edit

```
pip install --upgrade paho-mqtt
```

Try running the updated script again and see if the issue is resolved! 🚀



The error message shows:

1. **Warning:**

```pgsql
DeprecationWarning: Callback API version 1 is deprecated, update to latest version
```

This means that `mqtt.CallbackAPIVersion.VERSION1` still works but is deprecated. You should update your code to use the newer API version.

2. **Error in** `client.connect("localhost", 1883)`:

```arduino
File "/home/group1/Desktop/mqyen/lib/python3.11/site-packages/paho/mqtt/client.py", line 1435, in connect
return self.reconnect()
...
self._sock = self._create_socket()
```

This suggests a connection failure, likely due to:

- The MQTT broker **not running** on `localhost:1883`
- Network issues preventing the connection
- Firewall or security settings blocking the port

Fixes:

1. Check if the MQTT broker is running

Run the following command to check if a service is listening on port **1883**:

```sh
netstat -tulnp | grep 1883
```

or

```sh
ss -tulnp | grep 1883
```

If nothing is listening, start your MQTT broker, e.g., for **Mosquitto**:

```sh
sudo systemctl start mosquitto
sudo systemctl enable mosquitto
```

2. Try Connecting to a Public MQTT Broker

If your local broker isn't working, try a public MQTT broker (like `test.mosquitto.org`) by changing:

```python
client.connect("test.mosquitto.org", 1883)
```

3. Use the Latest MQTT API Version

Since **Version 1 is deprecated**, switch to **Version 2**:

```python
client = mqtt.Client(client_id="Subscriber",
callback_api_version=mqtt.CallbackAPIVersion.VERSION2)
```

Final Updated Code

```python
import paho.mqtt.client as mqtt

def on_message(client, userdata, message):
    print(f"Received message '{message.payload.decode()}' on topic
'{message.topic}'")

client = mqtt.Client(client_id="Subscriber",
callback_api_version=mqtt.CallbackAPIVersion.VERSION2)
client.on_message = on_message

# Change to test.mosquitto.org if local broker isn't working
client.connect("localhost", 1883)
client.subscribe("test/topic")
client.loop_forever()
```

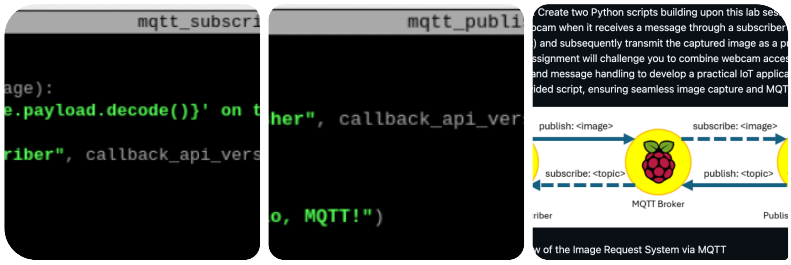
Next Steps

- If the local broker is down, restart it.
- If a firewall is blocking MQTT traffic, allow it with:

  ```sh
  sudo ufw allow 1883/tcp
  ```

- Test with a public MQTT broker to isolate the issue.

Try these fixes and let me know what happens! 🚀



i have 2 python files attached in the first 2 images. now i have to do this lab assignment attached in the 3rd image. how can i do the assignment.

To complete this lab assignment, you need to modify your MQTT publisher and subscriber scripts to handle **image capture from a webcam** and **image transmission over MQTT**. Below is a breakdown of how you can achieve this.

Steps to Implement the Lab Assignment

1. **Modify the Subscriber (** `mqtt_subscriber.py` **)**
   - Listen for an **image request** topic.
   - When a message is received, capture an image from the webcam.
   - Publish the image to another MQTT topic.
2. **Modify the Publisher (** `mqtt_publisher.py` **)**
   - Subscribe to the image topic.
   - Receive and decode the image.
   - Display or save the received image.

1️⃣ Updated `mqtt_subscriber.py` (Captures Image & Publishes)

This script acts as a **camera module** that listens for a request to take a picture. Once it gets the request, it captures an image and sends it over MQTT.

python

Copy Edit

```python
import paho.mqtt.client as mqtt
import cv2
import base64
import numpy as np

# MQTT Configuration
BROKER = "localhost"
REQUEST_TOPIC = "image/request"
IMAGE_TOPIC = "image/data"

def on_message(client, userdata, message):
    print(f"Received request: {message.payload.decode()}")

    # Capture image from webcam
    cap = cv2.VideoCapture(0)  # 0 is the default webcam
    ret, frame = cap.read()
    cap.release()

    if ret:
        # Encode image as Base64
        _, buffer = cv2.imencode('.jpg', frame)
        img_base64 = base64.b64encode(buffer).decode()

        # Publish image data
        client.publish(IMAGE_TOPIC, img_base64)
        print("Image captured and published.")
    else:
        print("Failed to capture image.")

# Initialize MQTT Client
client = mqtt.Client(client_id="Camera",
callback_api_version=mqtt.CallbackAPIVersion.VERSION2)
client.on_message = on_message
client.connect(BROKER, 1883)

# Subscribe to the request topic
client.subscribe(REQUEST_TOPIC)
client.loop_forever()
```

2️⃣ Updated `mqtt_publisher.py` (Sends Image Request & Displays Image)

This script acts as a **requesting module** that asks for an image and then receives and displays the image from MQTT.

python Copy Edit

```python
import paho.mqtt.client as mqtt
import time
import cv2
import base64
import numpy as np

# MQTT Configuration
BROKER = "localhost"
REQUEST_TOPIC = "image/request"
IMAGE_TOPIC = "image/data"

def on_message(client, userdata, message):
    print("Image received.")

    # Decode Base64 image
    img_data = base64.b64decode(message.payload)
    np_arr = np.frombuffer(img_data, np.uint8)
    frame = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)

    # Show image
    cv2.imshow("Received Image", frame)
    cv2.waitKey(5000)  # Display for 5 seconds
    cv2.destroyAllWindows()

# Initialize MQTT Client
client = mqtt.Client(client_id="Viewer",
callback_api_version=mqtt.CallbackAPIVersion.VERSION2)
client.on_message = on_message
client.connect(BROKER, 1883)

# Subscribe to the image data topic
client.subscribe(IMAGE_TOPIC)

# Send an image request
client.publish(REQUEST_TOPIC, "capture")
```

```
# Start loop to receive the image
client.loop_forever()
```

How It Works

1. The **Publisher (** `mqtt_publisher.py` **)** sends a request on `image/request` .
2. The **Subscriber (** `mqtt_subscriber.py` **)** listens on `image/request` , captures an image, and publishes it on `image/data` .
3. The **Publisher (** `mqtt_publisher.py` **)** receives the image from `image/data` , decodes it, and displays it.

Testing the System

1. **Run the MQTT broker** (if you haven't already):

   sh Copy Edit

   ```
   sudo systemctl start mosquitto
   ```

2. **Run the subscriber (camera module) first:**

   sh Copy Edit

   ```
   python3 mqtt_subscriber.py
   ```

3. **Run the publisher (request module) next:**

   sh Copy Edit

   ```
   python3 mqtt_publisher.py
   ```

4. **Result:**
   - The subscriber captures an image when a request is received.
   - The publisher receives and displays the image.

Next Steps

- If you need to **save** the received image, modify `mqtt_publisher.py`:

```python
cv2.imwrite("received_image.jpg", frame)
```

- If running on Raspberry Pi, ensure **OpenCV (`cv2`) is installed**:

```sh
pip install opencv-python numpy paho-mqtt
```

- If `cv2.VideoCapture(0)` doesn't work, **try `1` or `2` instead of `0`**, depending on your webcam.

This setup ensures smooth **image capturing, transmission, and display over MQTT**. 🚀 Let me know if you need any modifications!

The error message:

```arduino
qt.qpa.plugin: Could not find the Qt platform plugin "wayland"
```

indicates that OpenCV's GUI functions (such as `cv2.imshow`) require the **Qt** backend, but it is missing or misconfigured in your environment.

Solution 1: Force OpenCV to Use X11 Instead of Wayland

Try running your script with this environment variable:

```sh
```

```
export QT_QPA_PLATFORM=xcb
python3 mqtt_publisher.py
```

Or modify your script before `cv2.imshow()`:

python Copy Edit

```
import os
os.environ["QT_QPA_PLATFORM"] = "xcb"
```

Solution 2: Install Missing Qt Plugins

If you are using Ubuntu or Debian-based Linux, install the missing package:

sh Copy Edit

```
sudo apt update
sudo apt install qt6-wayland
```

For Fedora:

sh Copy Edit

```
sudo dnf install qt6-qtwayland
```

For Arch Linux:

sh Copy Edit

```
sudo pacman -S qt6-wayland
```

Solution 3: Run OpenCV Without `cv2.imshow()`

If you **do not need to display the image** but only want to save it, **replace**:

python Copy Edit

```
cv2.imshow("Received Image", frame)
cv2.waitKey(5000)
```

```
cv2.destroyAllWindows()
```

with:

```python
cv2.imwrite("received_image.jpg", frame)
print("Image saved as received_image.jpg")
```

Final Check

If `cv2.imshow()` is still not working, verify OpenCV was installed with GUI support:

```sh
python3 -c "import cv2; print(cv2.getBuildInformation())" | grep GUI
```

If **Qt is missing**, reinstall OpenCV with GUI support:

```sh
pip install --upgrade opencv-python-headless
```

This will install OpenCV **without GUI dependencies**, avoiding Qt errors.
