Device Readings -

```
BROKER = "localhost"
PORT = 1883

# MQTT topic format: domain/sector/device
TOPIC_TEMPLATE = "{domain}/{sector}/{device}"

# Define devices with specific payload structures
devices = {
    "crowd": {
        "device": "camera1",
        "payload": lambda: {"people_count": random.randint(50, 500), "density": round(random.uniform(0.1, 1.0), 2)}
    },
    "sanitation": {
        "device": "sensor1",
        "payload": lambda: {"garbage_level": random.randint(0, 100), "odor_index": random.randint(1, 10)}
    },
    "traffic": {
        "device": "traffic_light1",
        "payload": lambda: {"vehicle_count": random.randint(10, 300), "signal_status": random.choice(["Red", "Yellow", "Green"])}
    },
    "healthcare": {
        "device": "heart_monitor1",
        "payload": lambda: {"heart_rate": random.randint(60, 120), "oxygen_level": random.randint(90, 100)}
    },
    "energy": {
        "device": "smart_meter1",
        "payload": lambda: {"power_usage": random.randint(100, 5000), "voltage": random.randint(200, 240)}
    },
    "announcements": {
        "device": "speaker1",
        "payload": lambda: {"message": random.choice(["Stay Hydrated!", "Emergency Exit Route Updated", "Lost Child Assistance Available"])}
    }
}

sectors = ["sector1", "sector2", "sector3", "sector4"]
```

Publisher.py-

```
def publish_all_domains():
    client = mqtt.Client()
    client.connect(BROKER, PORT, 60)

    for sector in sectors:  # Publish for each sector
        for domain, info in devices.items():
            device = info["device"]
            payload = info["payload"]()

            topic = TOPIC_TEMPLATE.format(domain=domain, sector=sector, device=device)
            message = json.dumps(payload)  # Convert payload to JSON format

            client.publish(topic, message)
            print(f"Published: {message} to {topic}")

            time.sleep(1)

    client.disconnect()
    print("All messages published.")

# Run the publisher once
publish_all_domains()
```

[7] ⊗ 6.0s

All messages published –

```
client = mqtt.Client()
Published: {"people_count": 460, "density": 0.35} to crowd/sector1/camera1
Published: {"garbage_level": 13, "odor_index": 7} to sanitation/sector1/sensor1
Published: {"vehicle_count": 25, "signal_status": "Red"} to traffic/sector1/traffic_light1
Published: {"heart_rate": 115, "oxygen_level": 98} to healthcare/sector1/heart_monitor1
Published: {"power_usage": 1778, "voltage": 235} to energy/sector1/smart_meter1
Published: {"message": "Emergency Exit Route Updated"} to announcements/sector1/speaker1
Published: {"people_count": 317, "density": 0.52} to crowd/sector2/camera1
Published: {"garbage_level": 55, "odor_index": 10} to sanitation/sector2/sensor1
Published: {"vehicle_count": 111, "signal_status": "Green"} to traffic/sector2/traffic_light1
Published: {"heart_rate": 78, "oxygen_level": 99} to healthcare/sector2/heart_monitor1
Published: {"power_usage": 2191, "voltage": 232} to energy/sector2/smart_meter1
Published: {"message": "Emergency Exit Route Updated"} to announcements/sector2/speaker1
Published: {"people_count": 67, "density": 0.99} to crowd/sector3/camera1
Published: {"garbage_level": 95, "odor_index": 1} to sanitation/sector3/sensor1
Published: {"vehicle_count": 154, "signal_status": "Green"} to traffic/sector3/traffic_light1
Published: {"heart_rate": 95, "oxygen_level": 99} to healthcare/sector3/heart_monitor1
Published: {"power_usage": 4411, "voltage": 201} to energy/sector3/smart_meter1
Published: {"message": "Lost Child Assistance Available"} to announcements/sector3/speaker1
Published: {"people_count": 103, "density": 0.13} to crowd/sector4/camera1
Published: {"garbage_level": 62, "odor_index": 10} to sanitation/sector4/sensor1
Published: {"vehicle_count": 124, "signal_status": "Yellow"} to traffic/sector4/traffic_light1
Published: {"heart_rate": 86, "oxygen_level": 97} to healthcare/sector4/heart_monitor1
Published: {"power_usage": 4428, "voltage": 234} to energy/sector4/smart_meter1
Published: {"message": "Lost Child Assistance Available"} to announcements/sector4/speaker1
All messages published.
```

Overall subscriber.py –

```
BROKER = "localhost"
TOPIC = "crowd/#"

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.subscribe("#")  # Subscribe to all messages

def on_message(client, userdata, msg):
    print(f"Received: {msg.payload.decode()} from {msg.topic}")

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect(BROKER, 1883, 60)
client.loop_forever()
```

[2] 29.5s

```
C:\Users\anves\AppData\Local\Temp\ipykernel_16768\2438233750.py:13: DeprecationWarning: Callback API
  client = mqtt.Client()
Connected with result code 0
Received: {"people_count": 162, "density": 0.26} from crowd/sector1/camera1
Received: {"garbage_level": 66, "odor_index": 10} from sanitation/sector1/sensor1
Received: {"vehicle_count": 271, "signal_status": "Green"} from traffic/sector1/traffic_light1
Received: {"heart_rate": 61, "oxygen_level": 95} from healthcare/sector1/heart_monitor1
Received: {"power_usage": 4517, "voltage": 201} from energy/sector1/smart_meter1
Received: {"message": "Lost Child Assistance Available"} from announcements/sector1/speaker1
```

Camera only subscriber

```
import paho.mqtt.client as mqtt

BROKER = "localhost"
PORT = 1883
TOPIC = "crowd/sector1/camera1"  # Specific device

def on_message(client, userdata, message):
    print(f"Device-Level | {message.topic}: {message.payload.decode()}")

client = mqtt.Client(client_id="DeviceSubscriber", protocol=mqtt.MQTTv311)
client.on_message = on_message
client.connect(BROKER, PORT, 60)
client.subscribe(TOPIC)
print(f"Subscribed to {TOPIC}")
client.loop_forever()
```

[4] 17.2s

```
C:\Users\anves\AppData\Local\Temp\ipykernel_16768\3213386342.py:10: DeprecationWarning:
  client = mqtt.Client(client_id="DeviceSubscriber", protocol=mqtt.MQTTv311)
Subscribed to crowd/sector1/camera1
Device-Level | crowd/sector1/camera1: {"people_count": 140, "density": 0.84}
```

Sanitation only subscriber

```
BROKER = "localhost"
PORT = 1883
TOPIC = "sanitation/#"

def on_message(client, userdata, message):
    print(f"Sector-Level | {message.topic}: {message.payload.decode()}")

client = mqtt.Client(client_id="SectorSubscriber", protocol=mqtt.MQTTv311)
client.on_message = on_message
client.connect(BROKER, PORT, 60)
client.subscribe(TOPIC)
print(f"Subscribed to {TOPIC}")
client.loop_forever()
```

[6] 26.2s

```
C:\Users\anves\AppData\Local\Temp\ipykernel_16344\845952362.py:10: DeprecationWarning:
  client = mqtt.Client(client_id="SectorSubscriber", protocol=mqtt.MQTTv311)
Subscribed to sanitation/#
Sector-Level | sanitation/sector1/sensor1: {"garbage_level": 53, "odor_index": 4}
Sector-Level | sanitation/sector2/sensor1: {"garbage_level": 5, "odor_index": 1}
Sector-Level | sanitation/sector3/sensor1: {"garbage_level": 32, "odor_index": 4}
Sector-Level | sanitation/sector4/sensor1: {"garbage_level": 62, "odor_index": 8}
```