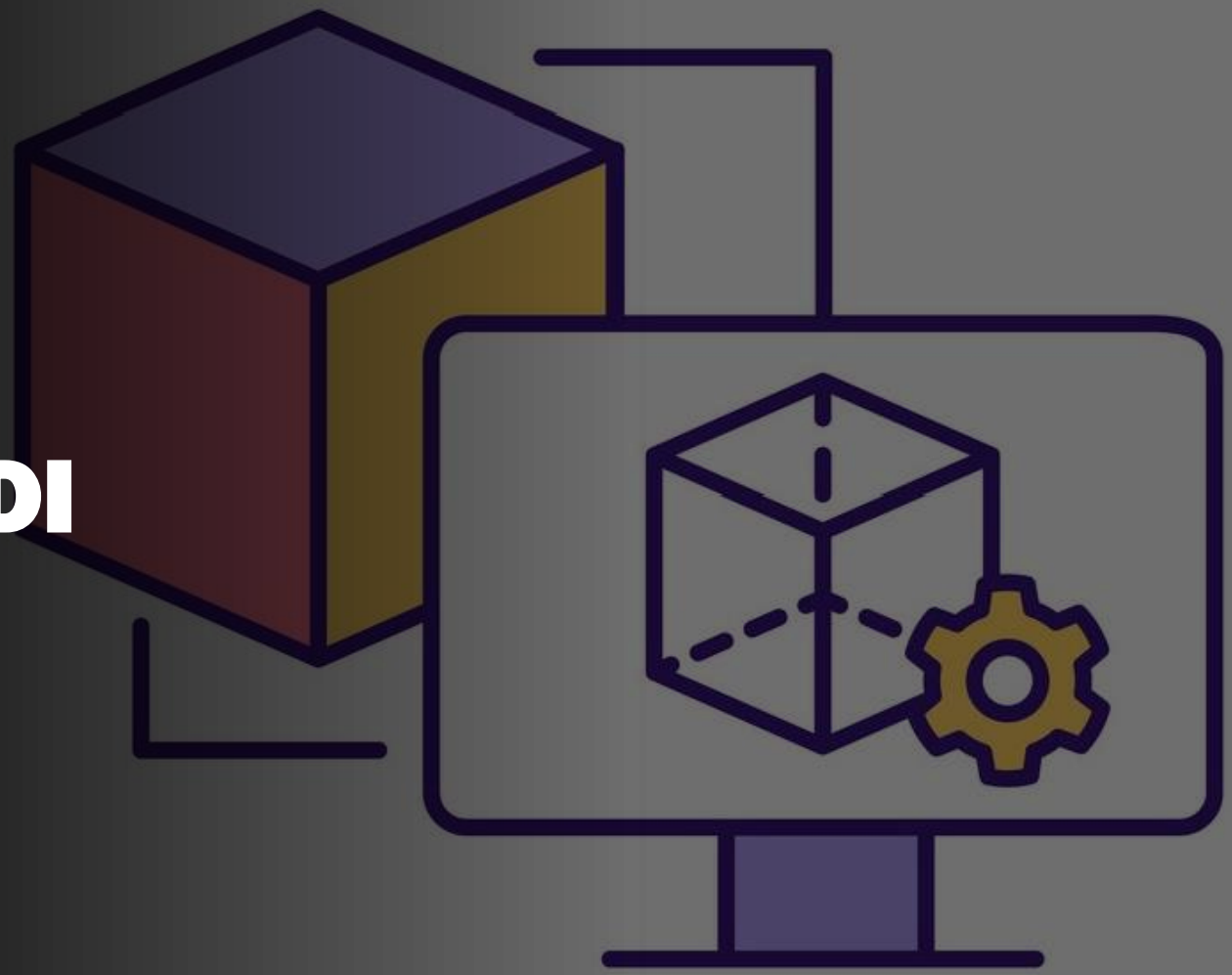
PROGETTAZIONE E REALIZZAZIONE DI DIGITAL TWIN INTEROPERABILI PER SCENARI APPLICATIVI DI INTERNET OF THINGS



Monika Ghidotti

Relatore: Prof. Ing. Marco Picone

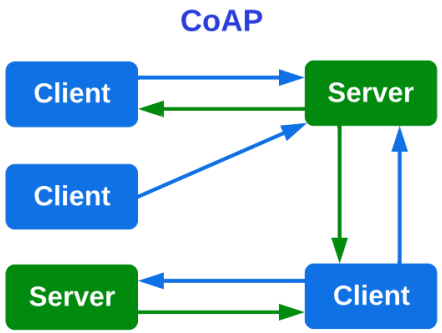
20 Ottobre 2022

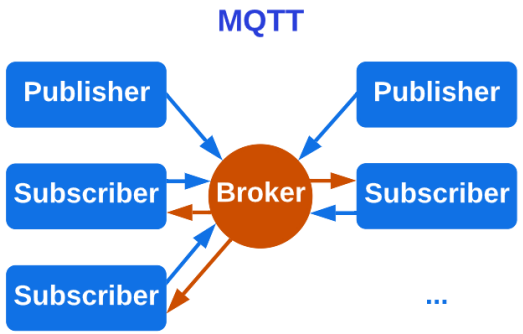
Introduzione - IoT

L' idea alla base dell'**Internet of Things** è quella di portare nel mondo digitale gli oggetti di uso comune.

Dato che molti degli oggetti IoT sono dispositivi con bassa capacità computazionale e di memoria,

- a livello fisico vengono usate tecnologie di comunicazione sviluppate appositamente per ridurre i consumi di energia dei dispositivi,
- tutti i dispositivi utilizzano il *protocollo IP* a livello di rete,
- a livello applicativo esistono protocolli progettati specificatamente per dispositivi con bassa capacità computazionale. I protocolli più usati nell'IoT sono:
  - **CoAP** – protocollo basato sul modello Richiesta/Risposta e sul paradigma REST
  - **MQTT** – protocollo basato sul modello Pub/Sub



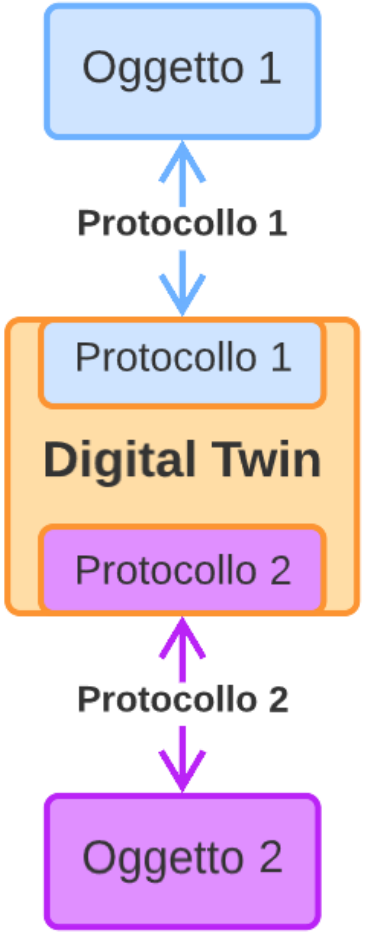


Introduzione – Digital Twin

Data l'eterogeneità del mondo IoT è possibile **sfruttare i Digital Twins** per mitigare questa varietà, portando ***interoperabilità*** anche a livello degli oggetti fisici.

Un **Digital Twin** è una **rappresentazione virtuale di un oggetto o di un sistema.**

- Il Digital Twin si *connette* all'oggetto fisico e rimane la sua controparte virtuale per tutto il ciclo di vita dell'oggetto
- Il Digital Twin si *sincronizza* continuamente con l'oggetto fisico.
- È possibile sfruttare le caratteristiche software dei Digital Twin per *estendere e migliorare le funzionalità della controparte fisica – proprietà di* ***augmentation***.

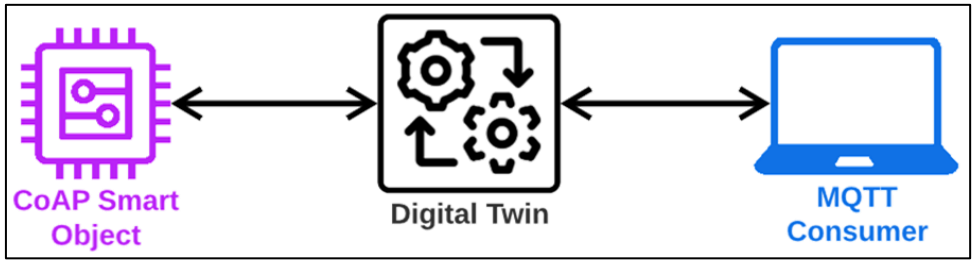


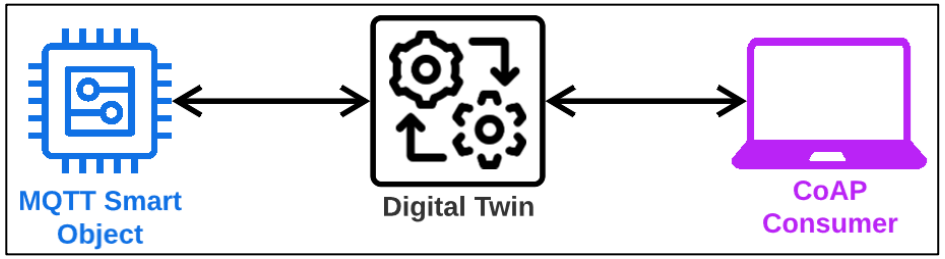
Problema

Considerando endpoint che utilizzano i protocolli più noti nell'Internet of Things, cioè **CoAP e MQTT**, è quindi possibile adoperare il Digital Twin per consentire comunicazioni tra oggetti CoAP e consumatori MQTT e, viceversa, tra oggetti MQTT e consumatori CoAP.

Il Digital Twin sarà un componente in grado di:

- **Ricevere** i messaggi dall'oggetto (CoAP o MQTT)
- **Mappare** i messaggi nel **protocollo** del consumer
- **Inviare** messaggi al consumer utilizzando il protocollo utilizzato dal consumatore



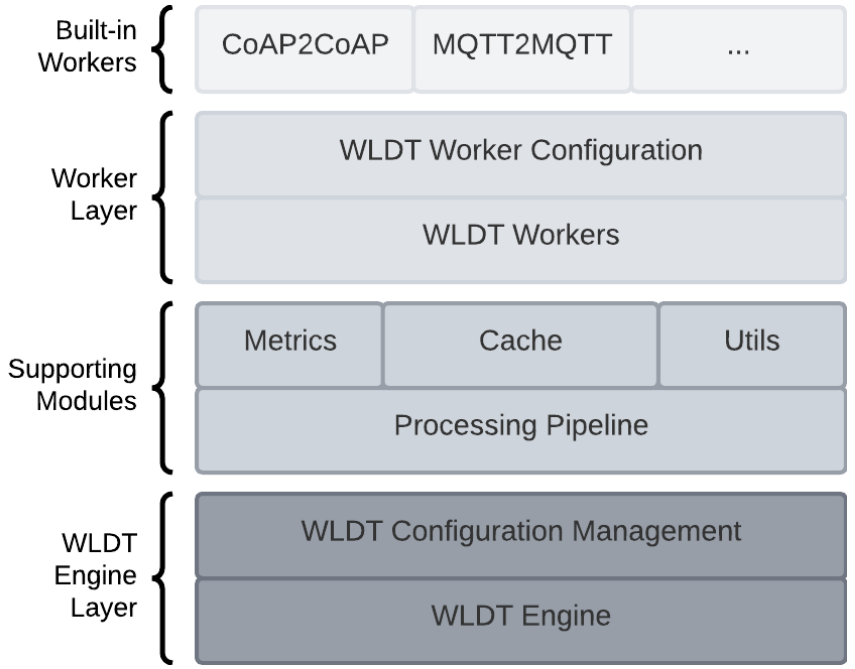


***<u>Come</u>** utilizzare i Digital Twin per consentire la comunicazione tra partecipanti capaci di parlare solamente il **protocollo CoAP** o il **protocollo MQTT**?*

White Lablel Digital Twin

Esistono vari framework per lo sviluppo di Digital Twin. Fra questi, la libreria **White Label Digital Twin (WLDT)** offre la possibilità di creare Digital Twins modulari e flessibili per scenari applicativi di Internet of Things.

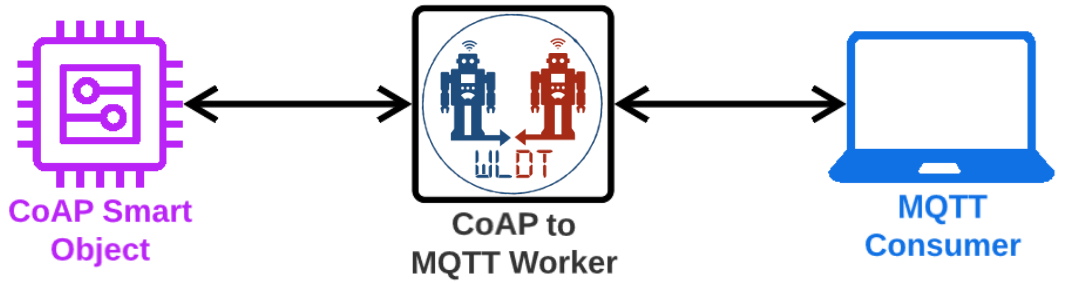
- **WLDT Engine** → gestisce e orchestra i vari Worker
- **Worker** → modulo attivo del framework che modella funzionalità specifiche del Digital Twin
  - Workers built-in → *CoAP2CoAP*, *MQTT2MQTT*
- **Moduli di supporto** → Processing pipeline, Cache, Metriche



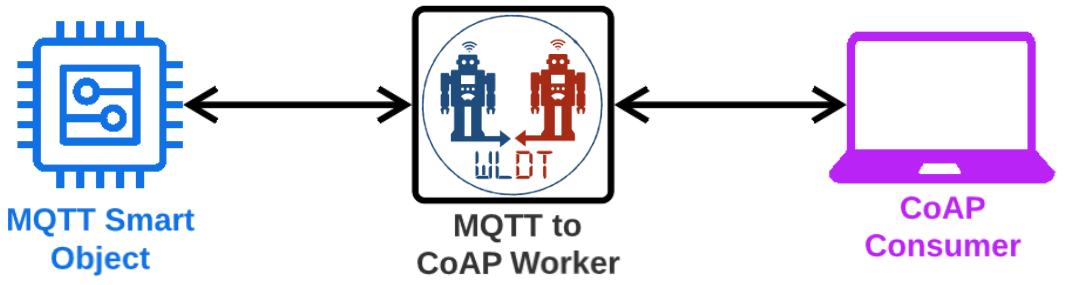
Soluzione proposta

Sono stati creati due Worker aggiuntivi del framework **White Label Digital Twin**:

- **CoAP to MQTT Worker**: funziona da intermediario tra un oggetto CoAP e un consumer MQTT

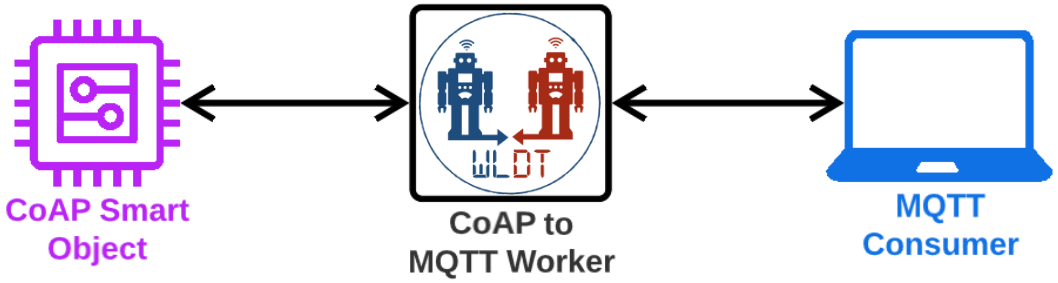


- **MQTT to CoAP Worker**: funge da intermediario tra un oggetto MQTT e un consumer CoAP

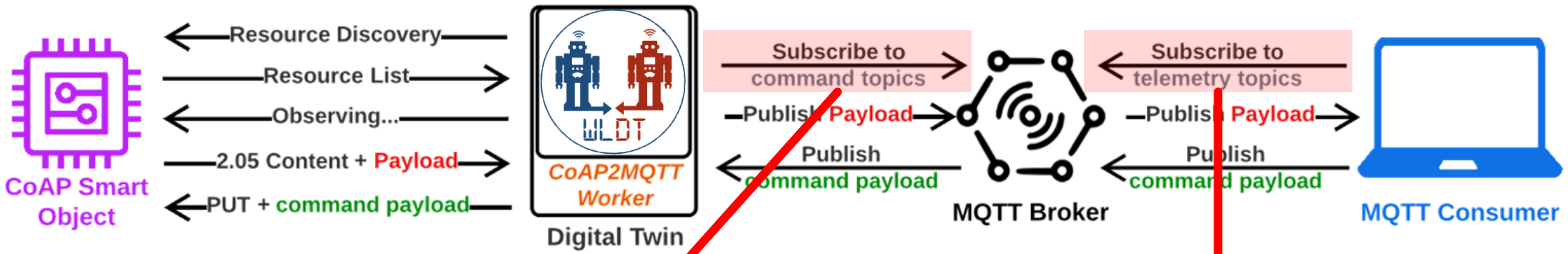


CoAP to MQTT Worker

Il Worker CoAP to MQTT è un modulo configurabile che consente di creare Digital Twins in grado di fare da *intermediari tra un oggetto che parla il protocollo CoAP e un consumer che parla il protocollo MQTT*.



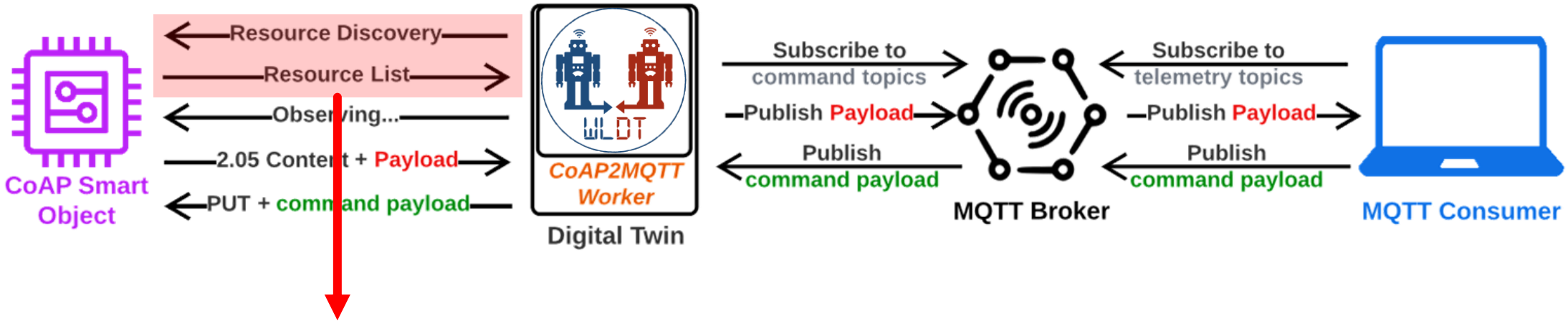
CoAP to MQTT Worker



1. Il **Digital Twin** si **sottoscrive ai topic di comando** sul Broker MQTT

Il **Consumer MQTT** si **sottoscrive** ai **topic di telemetria** sul Broker MQTT

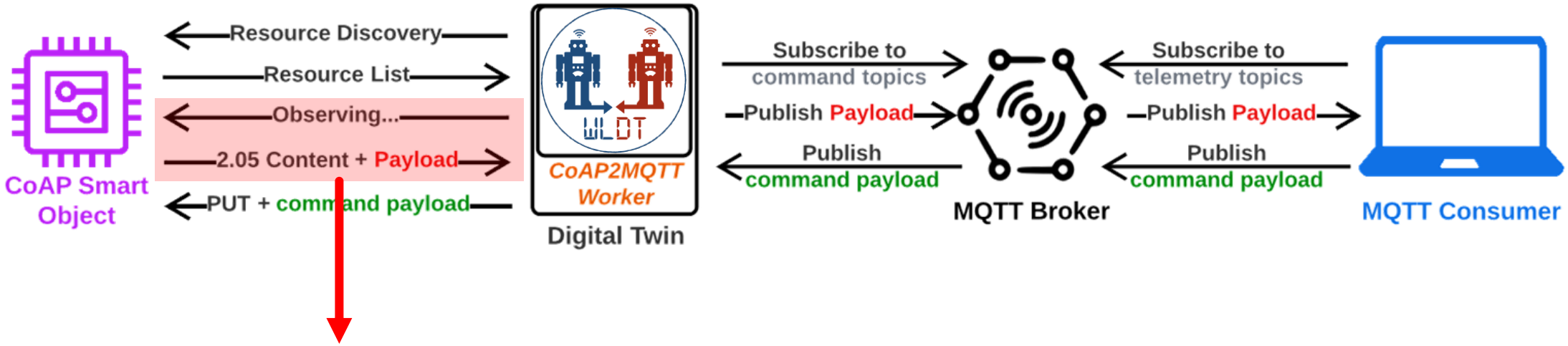
CoAP to MQTT Worker



Ogni oggetto presenta delle risorse (es. sensore di temperatura, switch, …)

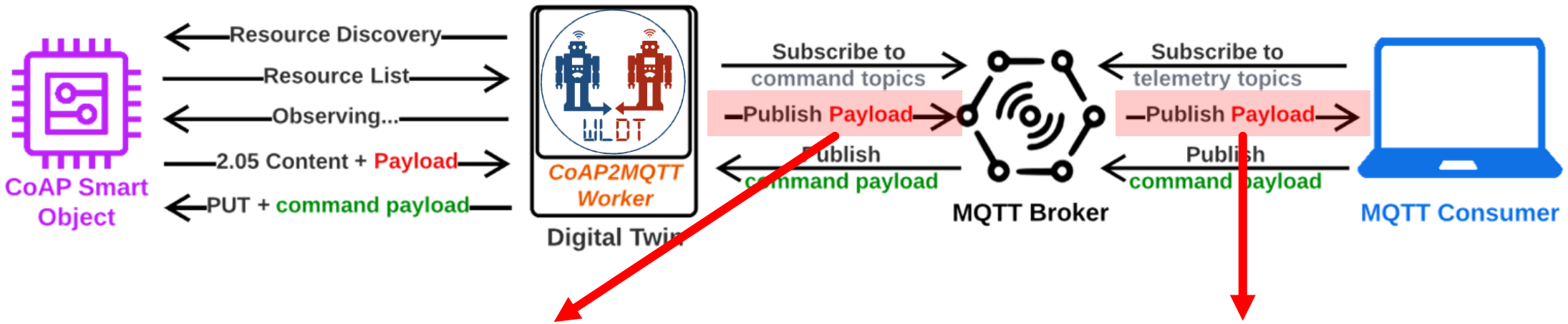
2. Il **Digital Twin chiede** all'oggetto la **lista delle risorse** da esso ospitate. **L'oggetto risponde** al Digital Twin **con la lista delle risorse** contenente l'URI della risorsa e alcuni attributi (tipo di risorsa, tipo di interfaccia, …)

CoAP to MQTT Worker



3. Data la lista delle risorse, il **Digital Twin inizia a osservare** tutte le risorse osservabili.
In questo modo, **l'oggetto manderà un messaggio al Digital Twin** ogni volta che lo stato di una risorsa cambia

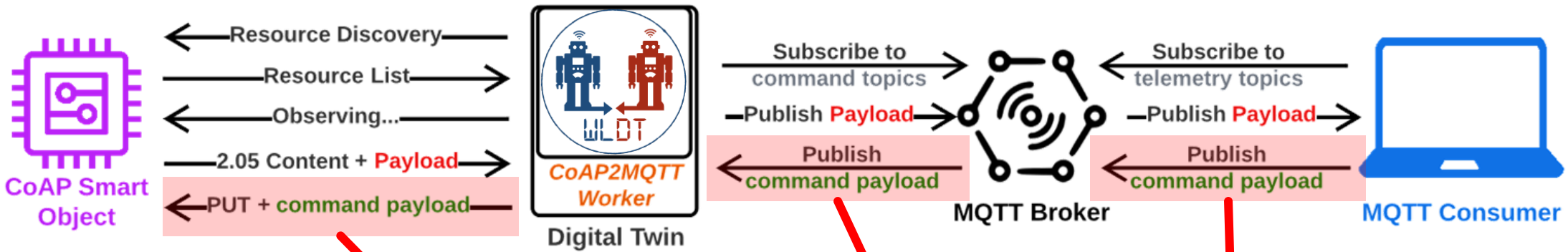
CoAP to MQTT Worker



4. Ogni volta che arriva un messaggio dall'oggetto, il **Digital Twin pubblicherà il payload del messaggio sul Broker MQTT**

Il **Broker MQTT recapiterà il messaggio al consumatore MQTT**

CoAP to MQTT Worker



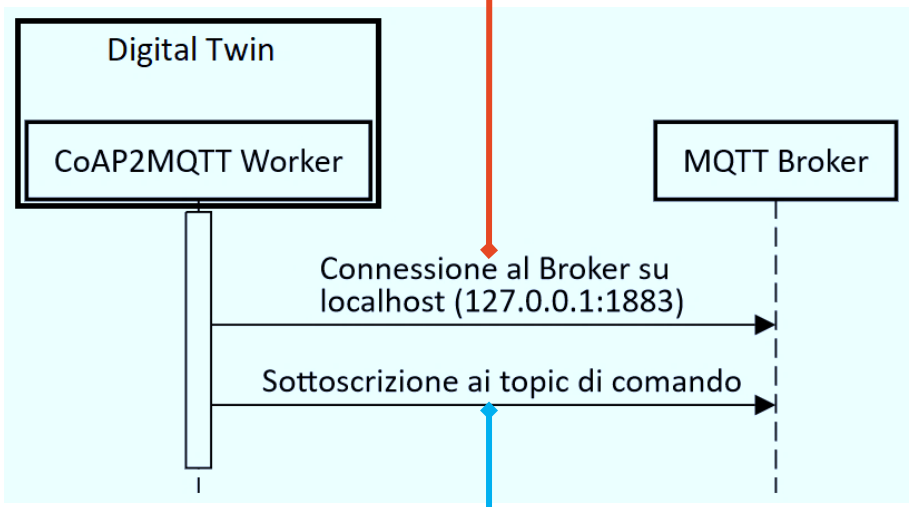
6. Il **Digital Twin manderà** il **payload del messaggio di comando all'oggetto CoAP**

5. Il **consumer MQTT** può pubblicare un **messaggio di comando** sul Broker MQTT

Il **Broker MQTT recapiterà il messaggio al Digital Twin**

1. Connessione al Broker e sottoscrizione ai topic di comando

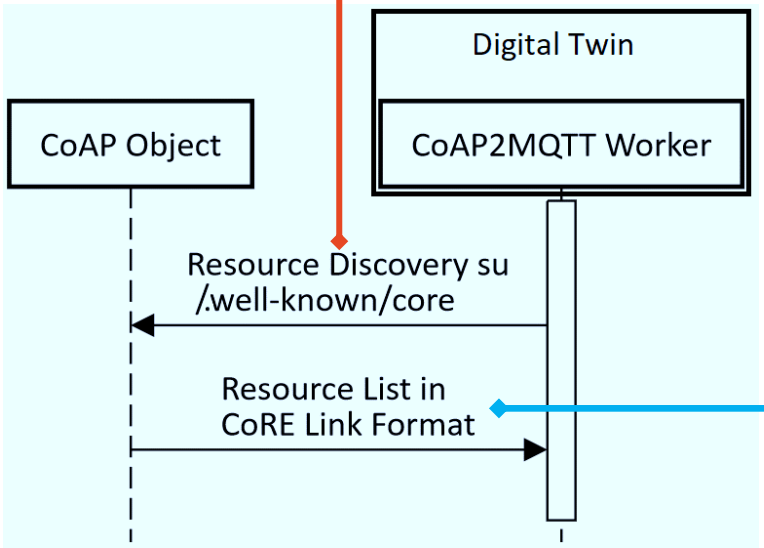

```
09:40:34.934 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - [WLDT-CoAP2MQTT-Manager]
Initializing outgoing MQTT Client ...
09:40:34.937 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - [WLDT-CoAP2MQTT-Manager] Client (
digitaltwin0001) - Connecting to tcp://127.0.0.1:1883
 ...
09:40:35.349 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - [WLDT-CoAP2MQTT-Manager] MQTT Client
 Connected to tcp://127.0.0.1:1883
```




```
09:40:35.355 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - Registered to MQTT Command Topic /
device/test/irrigation/status/change
09:40:35.357 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - Registered to MQTT Command Topic /
device/test/irrigation/status/action
09:40:35.357 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - Registered to MQTT Command Topic /
device/test/irrigation/level/change
09:40:35.358 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - Registered to MQTT Command Topic /
device/test/irrigation/level/action
```

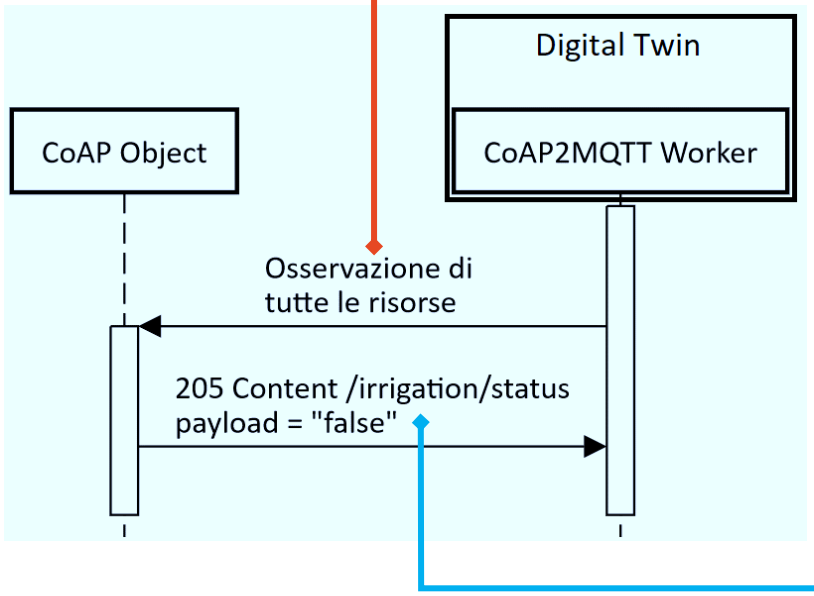
2. Resource Discovery


```
09:40:35.358 [pool-1-thread-1] DEBUG i.u.d.i.w.w.c.
Coap2MqttManager - [WLDT-CoAP2MQTT-Manager] Starting
CoAP Resource Discovery ...
09:40:35.358 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - Sending Discovery Requesto to: coap
://127.0.0.1:5683/.well-known/core
```




```
Coap2MqttManager - Resource Discovery Response:
 ==[ CoAP Response
 ]=============================================
MID     : 60033
Token   : 426A714E755A5541
Type    : ACK
Status  : 2.05 - CONTENT
Options: {"Content-Format":"application/link-format"}
RTT     : 19 ms
Payload: 250 Bytes
------------------------------------------------------
--------
</irrigation>,</irrigation/level>;ct="110 0";if="core.a
";obs;rt="it.unimore.device.actuator.level";title="
LevelActuator",</irrigation/status>;ct="110 0";if="core
.a";obs;rt="it.unimore.device.actuator.status";title="
StatusActuator",</.well-known/core>
======================================================
```

3. Osservazione delle risorse e ricezione messaggio di telemetria



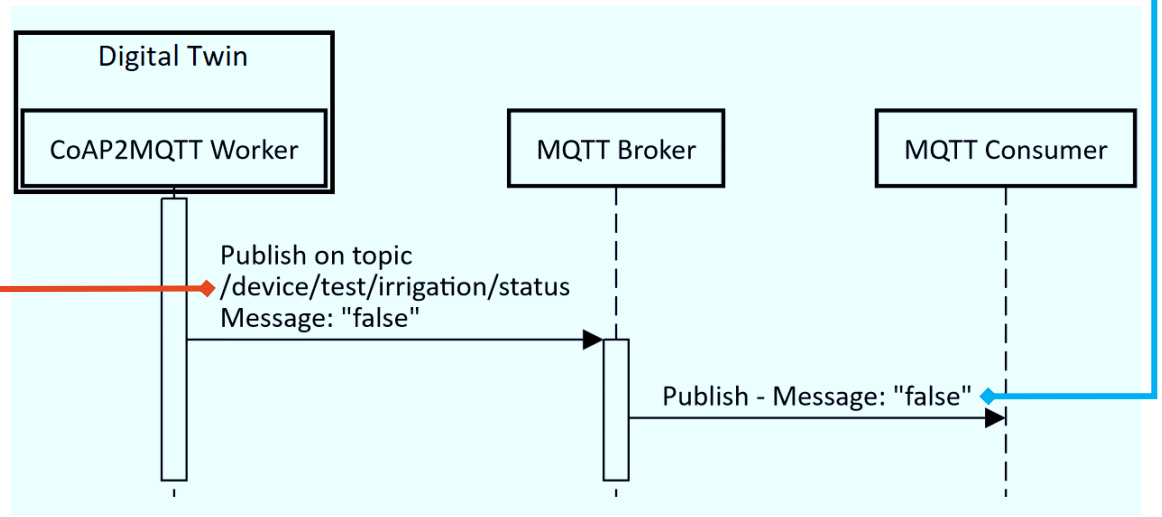

```
09:40:35.461 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - OBSERVING RESOURCE coap://127.0.0.1:
5683/irrigation/level ...
09:40:35.461 [pool-1-thread-1] INFO  i.u.d.i.w.w.c.
Coap2MqttManager - OBSERVING RESOURCE coap://127.0.0.1:
5683/irrigation/status ...
```

```
09:40:43.865 [:CoapEndpoint-UDP-0.0.0.0/0.0.0.0:0#1]
INFO  i.u.d.i.w.w.c.Coap2MqttManager - NOTIFICATION ->
From Resource : coap://127.0.0.1:5683/irrigation/status
 -> Body: false
09:40:43.865 [:CoapEndpoint-UDP-0.0.0.0/0.0.0.0:0#1]
INFO  i.u.d.i.w.w.c.Coap2MqttManager - ==[ CoAP
Response ]==============================================
MID     : 61202
Token   : 8F380F33C375270E
Type    : CON
Status : 2.05 - CONTENT
Options: {"Observe":3, "Content-Format":"text/plain"}
RTT     : 1 ms
Payload: 5 Bytes
---------------------------------------------------------
false
=========================================================
```

4. Pubblicazione del messaggio di telemetria sul Broker MQTT


```
09:40:43.865 [:CoapEndpoint-UDP-0.0.0.0/0.0.0.0:0#1]
DEBUG i.u.d.i.w.w.c.Coap2MqttManager - [WLDT-CoAP2MQTT-
Manager] Forwarding to TOPIC: /device/test/irrigation/
status - Message: false
09:40:43.867 [:CoapEndpoint-UDP-0.0.0.0/0.0.0.0:0#1]
DEBUG i.u.d.i.w.w.c.Coap2MqttManager - Data Correctly
Published !
```

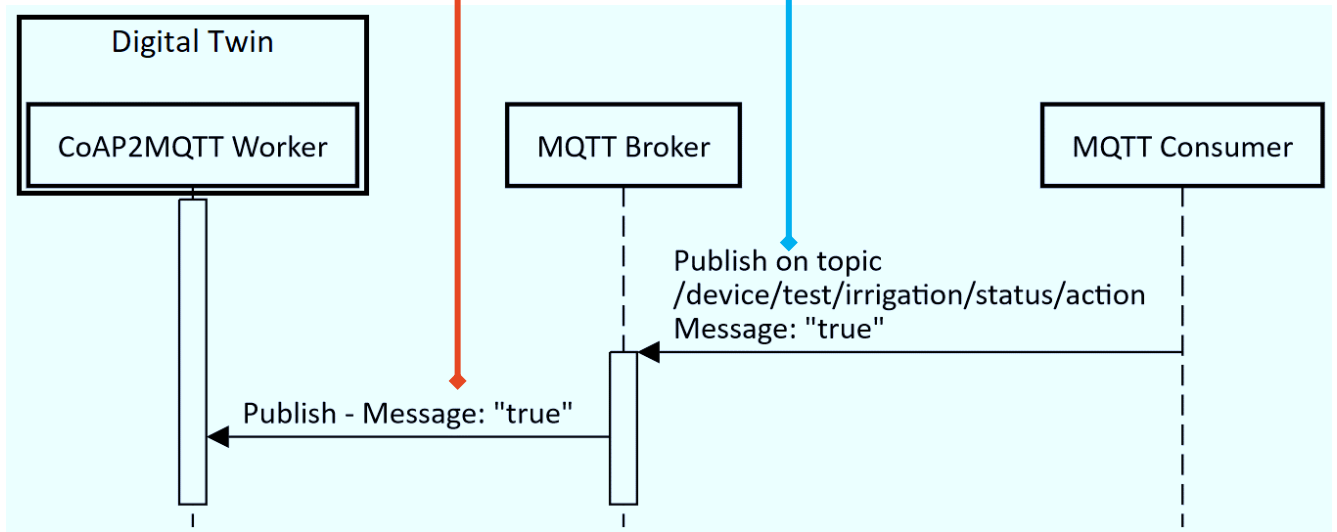
```
1664955643: Received PUBLISH from digitaltwin0001 (d0, q2, r1, m7,
 '/device/test/irrigation/status', ... (5 bytes))
1664955643: Sending PUBREC to digitaltwin0001 (m7, rc0)
1664955643: Received PUBREL from digitaltwin0001 (Mid: 7)
1664955643: Sending PUBLISH to SimpleClient-001 (d0, q1, r0, m5, '
/device/test/irrigation/status', ... (5 bytes))
```



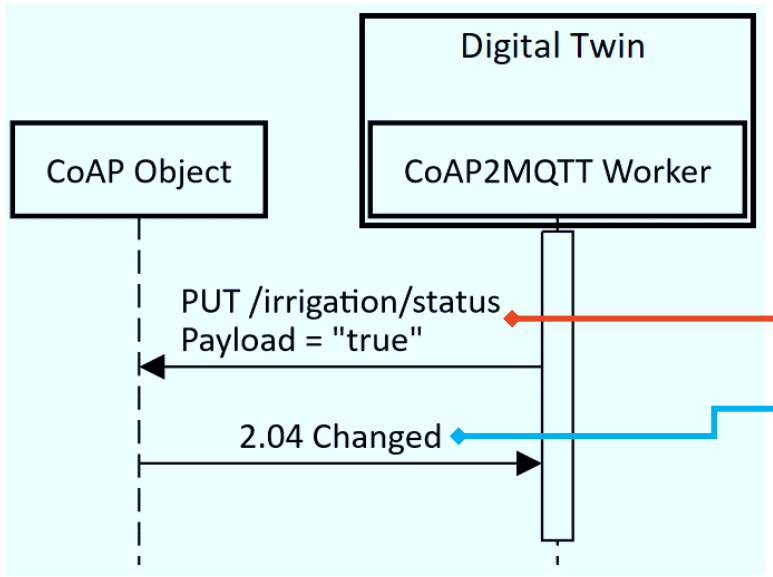
5. Ricezione del messaggio di comando dal Consumer MQTT


```
1664955683: Received PUBLISH from SimpleClient-001 (d0, q0, r0, m0
, '/device/test/irrigation/status/action', ... (4 bytes))
1664955683: Sending PUBLISH to digitaltwin0001 (d0, q0, r0, m0, '/
device/test/irrigation/status/action', ... (4 bytes))
```

```
09:43:23.903 [MQTT Call: SimpleClient-001] INFO  i.u.
wldt.test.coap2mqtt.MqttConsumer - Message received
 --> Topic: /device/test/irrigation/status -- Payload:
false
09:43:23.904 [Thread-9] INFO  i.u.wldt.test.coap2mqtt.
MqttConsumer - Sending to topic: /device/test/
irrigation/status/action -> Data: true
09:43:23.904 [Thread-9] INFO  i.u.wldt.test.coap2mqtt.
MqttConsumer - Data Correctly Published to topic: /
device/test/irrigation/status/action
```



6. Invio del messaggio di comando all'oggetto CoAP

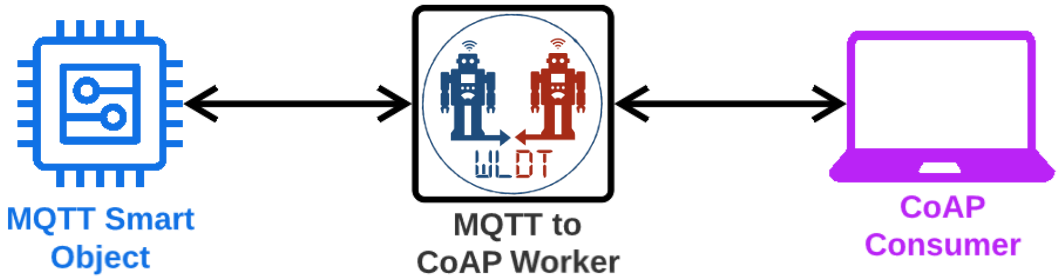



```
09:40:43.869 [MQTT Call: digitaltwin0001] INFO  i.u.d.i
.w.w.c.Coap2MqttManager - Forwarding Message to CoAP
object
09:40:43.871 [MQTT Call: digitaltwin0001] INFO  i.u.d.i
.w.w.c.Coap2MqttManager - Sending PUT Request to CoAP
Object
```

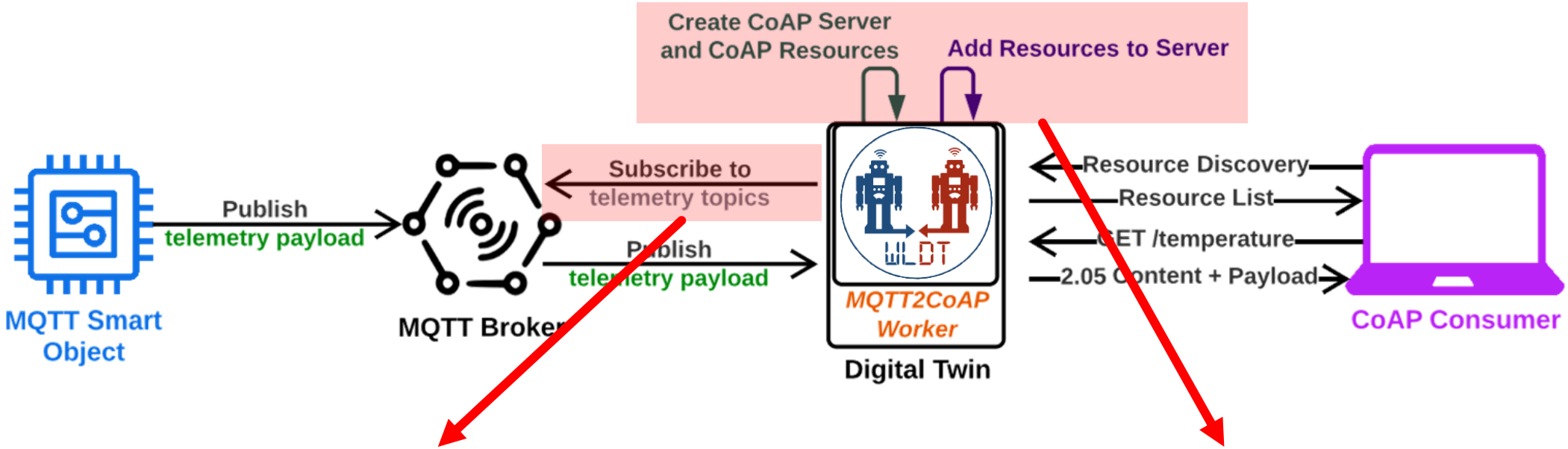
```
09:40:43.875 [MQTT Call: digitaltwin0001] INFO  i.u.d.i
.w.w.c.Coap2MqttManager - Response :
==[ CoAP Response
 ]=============================================
MID     : 11941
Token   : 645D35C773923C7D
Type    : ACK
Status  : 2.04 - CHANGED
Options: {}
RTT     : 3 ms
Payload: 0 Bytes
============================================================
```

MQTT to CoAP Worker

Il Worker MQTT to CoAP è un modulo configurabile che consente di creare Digital Twins in grado di fare da ***intermediari*** *tra un oggetto che parla il protocollo MQTT e un consumer che parla il protocollo CoAP*.



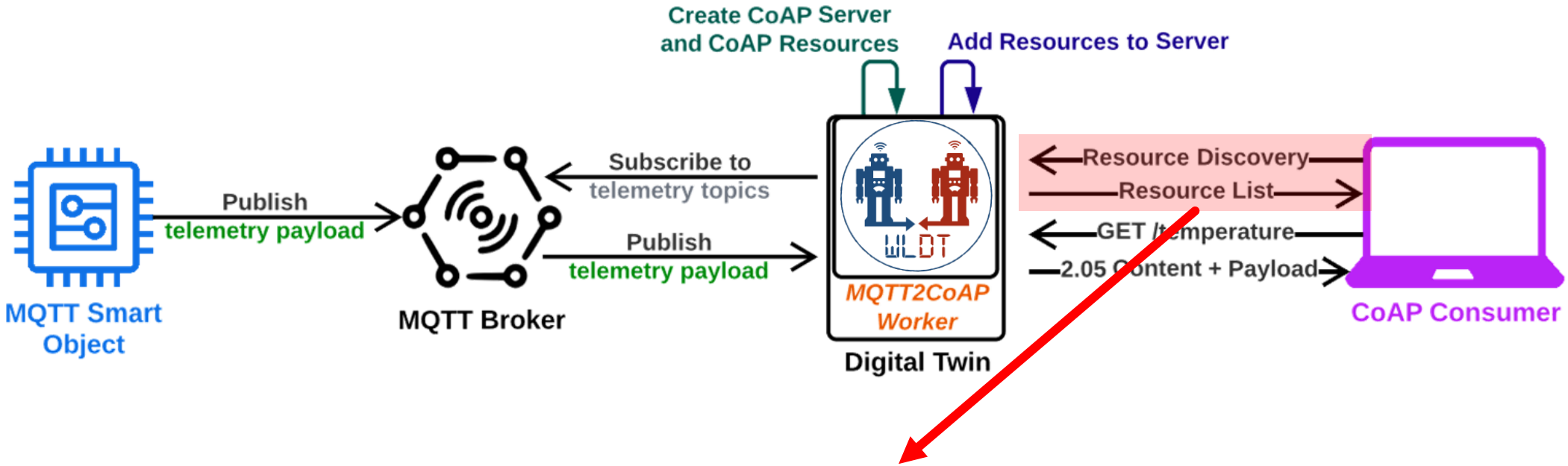
MQTT to CoAP Worker



1. Inizialmente il **Digital Twin** si **connette al Broker** e si **sottoscrive ai topic di telemetria**

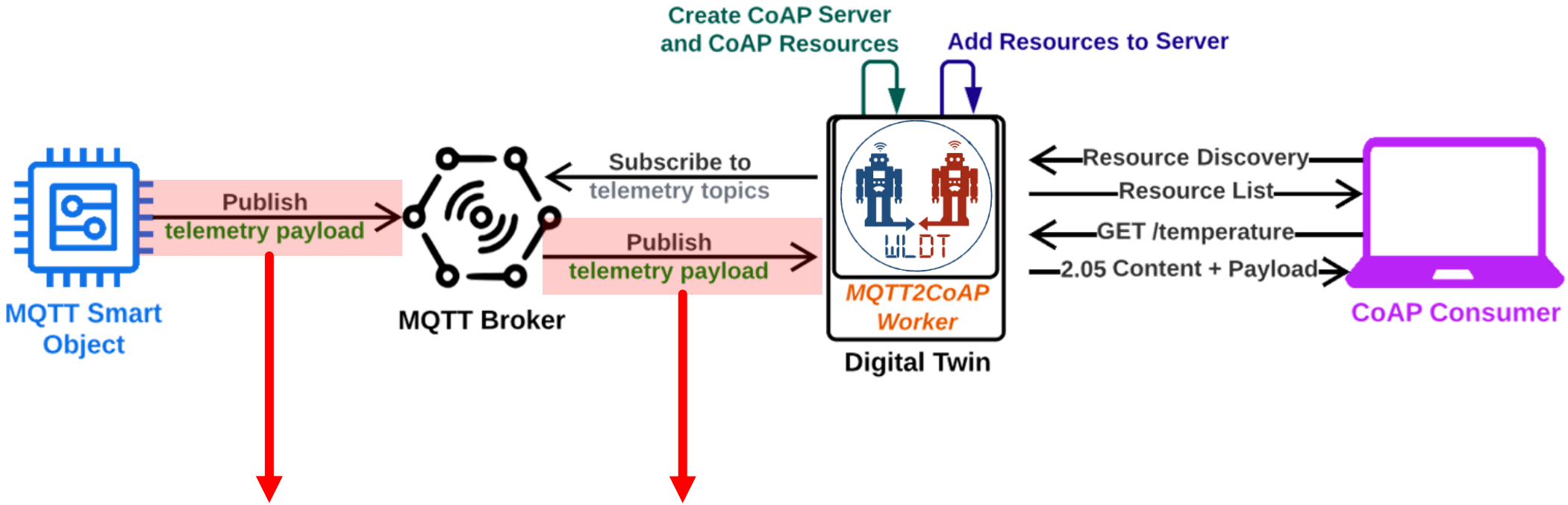
2. Il **Digital Twin crea e avvia** un **CoAP Server**. In seguito, **crea e aggiunge al Server** le **risorse CoAP** da esporre

MQTT to CoAP Worker



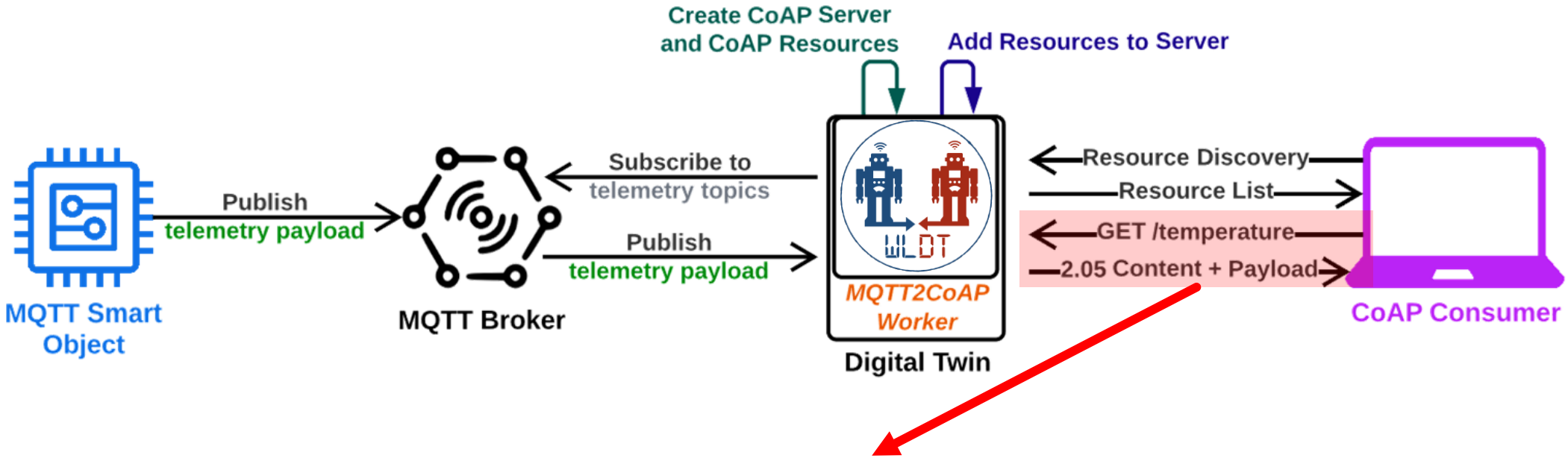
3. Quando il **consumer CoAP fa una Resource Discovery** per conoscere le risorse ospitate dal **Digital Twin**, quest'ultimo **risponderà con la lista delle risorse**.

MQTT to CoAP Worker



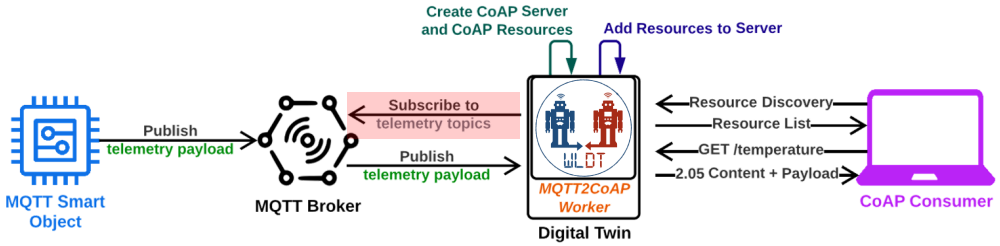
4. Ogni volta che **l'oggetto MQTT pubblica il messaggio sul Broker**, quest'ultimo lo **recapita al Digital Twin**, il quale salva il topic e il payload del messaggio in una mappa

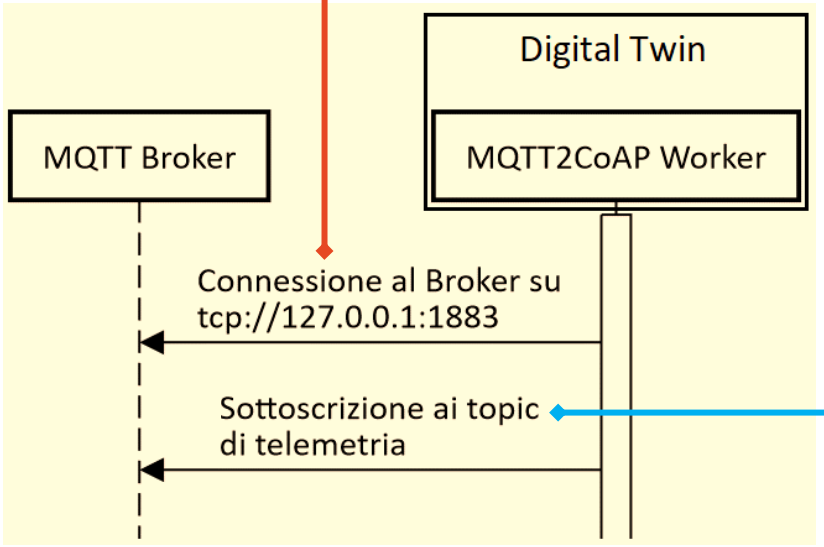
MQTT to CoAP Worker



5. Quando il consumer fa una **richiesta di tipo GET** per ottenere lo stato di una risorsa dal **Digital Twin**, quest'ultimo **risponderà con l'ultimo payload di telemetria ricevuto dall'oggetto MQTT** e associato a quella risorsa

1. Connessione al Broker e sottoscrizione ai topic di telemetria

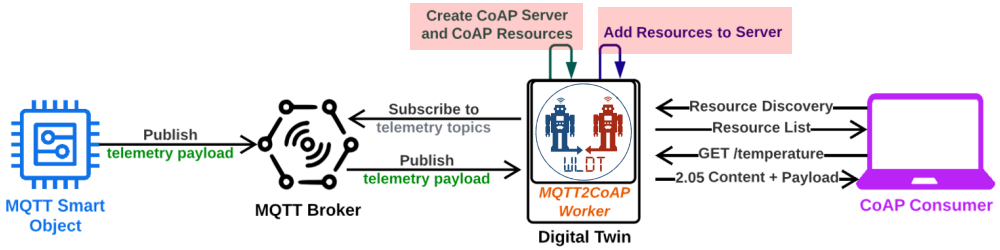


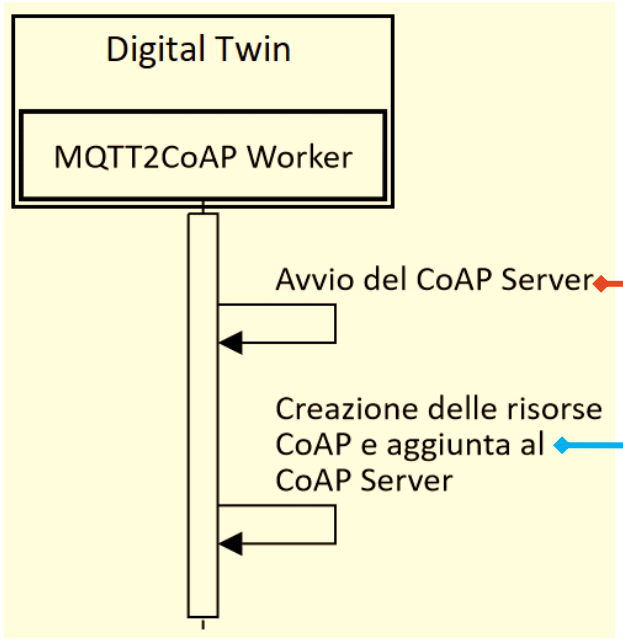



```
11:07:16.694 [pool-1-thread-1] INFO  i.u.d.i.w.w.m.
Mqtt2CoapManager - [WLDT-MQTT2CoAPManager] Initializing
 Incoming MQTT Client ...
11:07:16.697 [pool-1-thread-1] INFO  i.u.d.i.w.w.m.
Mqtt2CoapManager - [WLDT-MQTT2CoAPManager] Client (
digitaltwin0002) - Connecting to tcp://127.0.0.1:1883
 ...
11:07:17.098 [pool-1-thread-1] INFO  i.u.d.i.w.w.m.
Mqtt2CoapManager - [WLDT-MQTT2CoAPManager] INCOMING
MQTT Client Connected to tcp://127.0.0.1:1883
```

```
11:07:17.103 [pool-1-thread-1] INFO  i.u.d.i.w.w.m.
Mqtt2CoapManager - Subscribed on topic: /device/test/
sensor/battery
11:07:17.107 [pool-1-thread-1] INFO  i.u.d.i.w.w.m.
Mqtt2CoapManager - Subscribed on topic: /device/test/
sensor/temperature
```

2. Creazione CoAP Server e aggiunta delle risorse CoAP

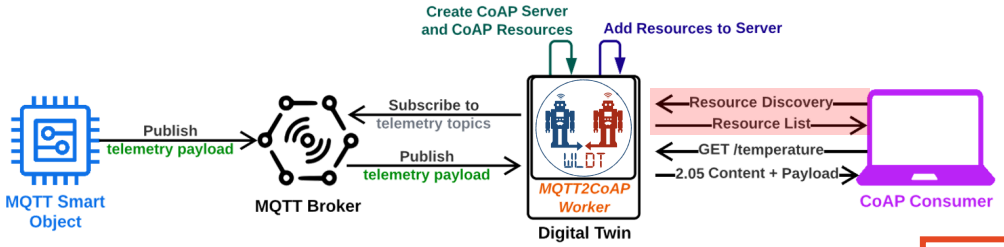


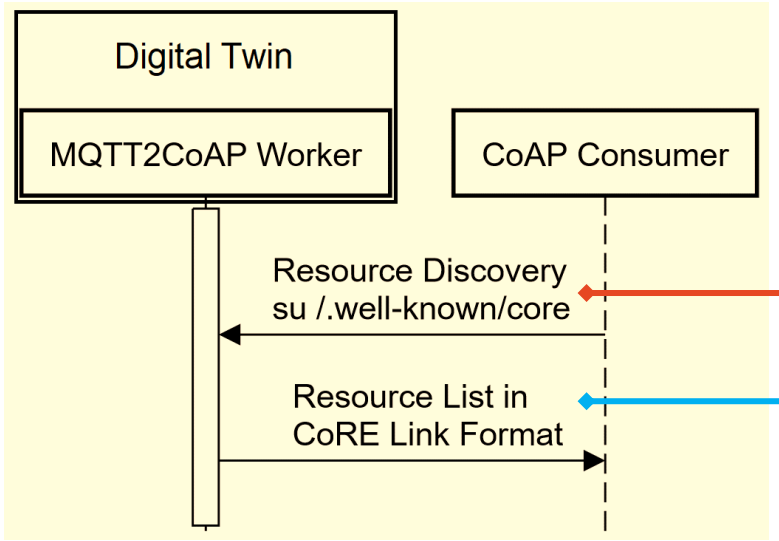



```
11:07:17.161 [pool-1-thread-1] INFO  o.e.californium.
core.CoapServer - Starting server
11:07:17.165 [pool-1-thread-1] DEBUG o.e.c.core.network
.CoapEndpoint - Starting endpoint at coap://0.0.0.0:
5783
```

```
11:07:17.160 [pool-1-thread-1] DEBUG i.u.d.i.w.w.m.
Mqtt2CoapResource - [WLDT-Mqtt2CoAP-Resource] Resource
Descriptor Configured ! Endpoint: coap://127.0.0.1:5783
/battery
11:07:17.161 [pool-1-thread-1] INFO  i.u.d.i.w.w.m.
Mqtt2CoapManager - Resource battery -> URI: /battery
   (Observable: false)
11:07:17.161 [pool-1-thread-1] DEBUG i.u.d.i.w.w.m.
Mqtt2CoapResource - [WLDT-Mqtt2CoAP-Resource] Resource
Descriptor Configured ! Endpoint: coap://127.0.0.1:5783
/temperature
11:07:17.161 [pool-1-thread-1] INFO  i.u.d.i.w.w.m.
Mqtt2CoapManager - Resource temperature -> URI: /
temperature   (Observable: false)
```

3. Resource Discovery dal Consumer CoAP



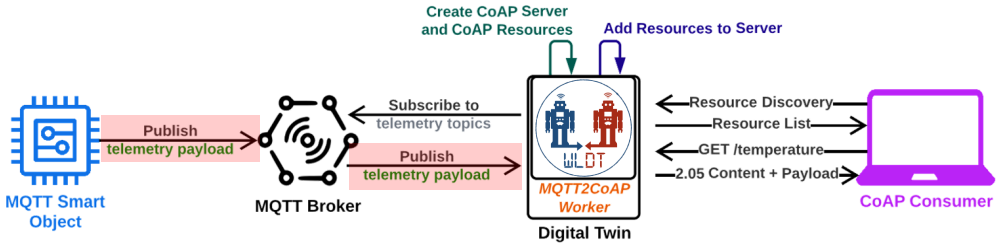


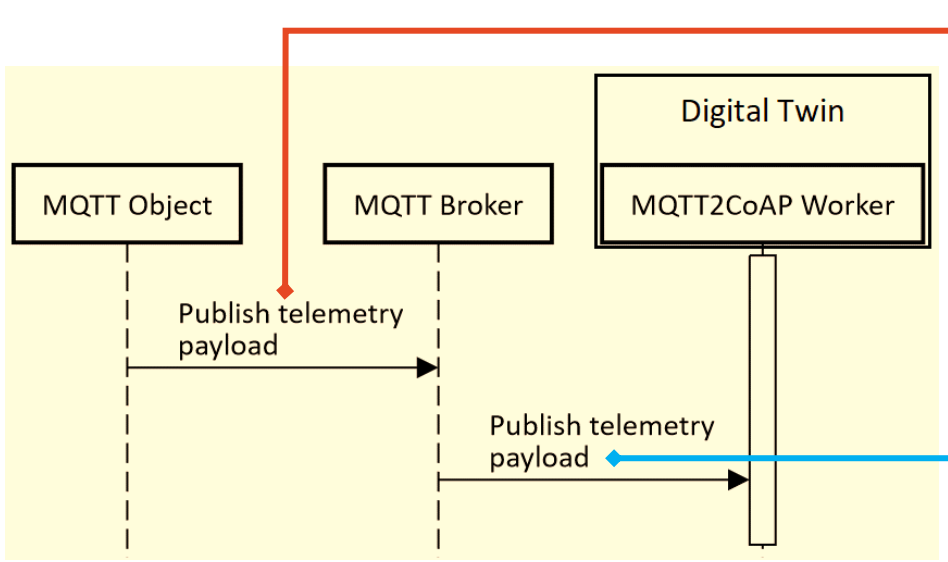

```
11:07:29.767 [main] INFO  i.u.w.t.m.c.
CoapResourceDiscoveryClientProcess - Resource Discovery
 Request to coap://127.0.0.1:5783/.well-known/core:
==[ CoAP Request  ]===================================
MID     : -1
Token   : null
Type    : CON
Method  : GET
Options: {}
Payload: 0 Bytes
=======================================================
```

```
11:07:29.929 [main] INFO  i.u.w.t.m.c.
CoapResourceDiscoveryClientProcess - Response :
==[ CoAP Response ]===================================

MID     : 24819
Token   : 6C3FD3799D89AF02
Type    : ACK
Status  : 2.05 - CONTENT
Options: {"Content-Format":"application/link-format"}
RTT     : 40 ms
Payload: 183 Bytes
-------------------------------------------------------
</temperature>;ct="0 110";if="core.s";rt="
temperatureSensor";title="temperature_sensor",</battery
>;ct="0 110";if="core.s";rt="batterySensor";title="
battery_sensor",</.well-known/core>
=======================================================
```

4. Pubblicazione del messaggio di telemetria

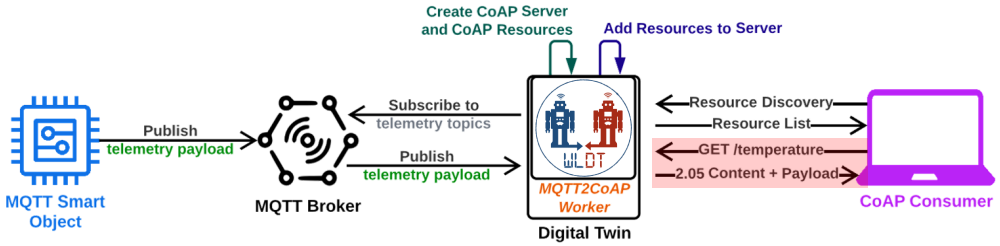


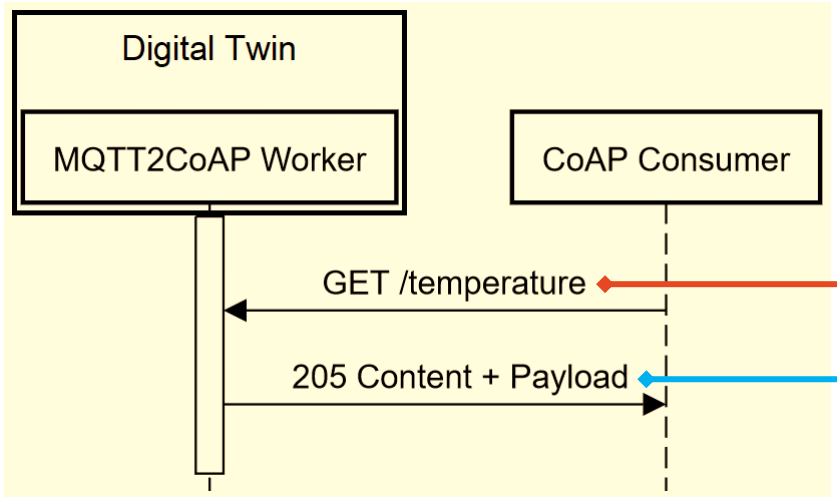



```
11:07:47.049 [Timer-0] INFO  i.u.w.t.m.m.o.
ThermometerMqttSmartObj - Sending to topic: /device/
test/sensor/temperature --> Data: TelemetryMessage{
timestamp=1665047267049, type='iot.sensor.temperature
', dataValue=26.954350537153328}
```

```
1665047267: Received PUBLISH from thermometer-001 (d0, q0, r0,
m0, '/device/test/sensor/temperature', ... (85 bytes))
1665047267: Sending PUBLISH to digitaltwin0002 (d0, q0, r0, m0,
 '/device/test/sensor/temperature', ... (85 bytes))
```

5. Richiesta di tipo GET dal Consumer CoAP





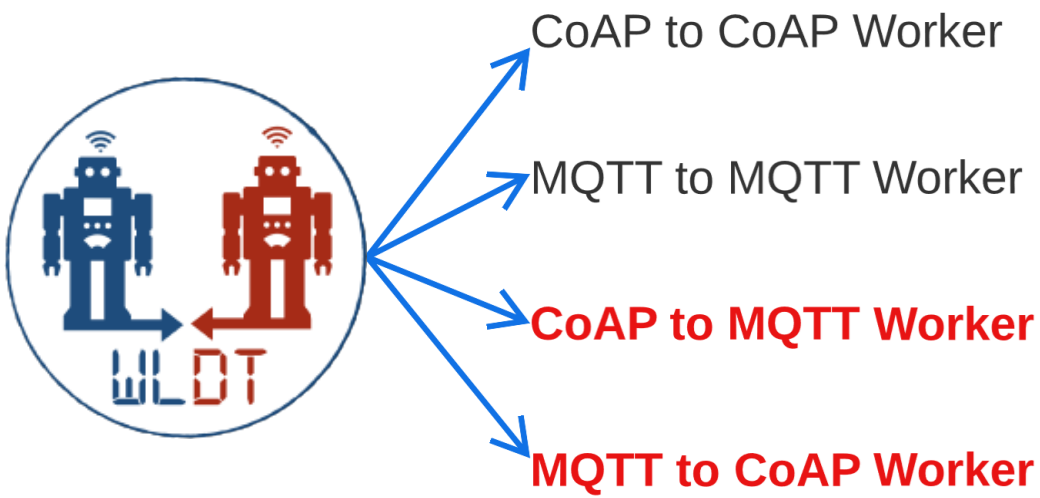

```
11:07:51.592 [main] INFO  i.u.w.t.m.c.
CoapGetClientProcess - Request to enpoint coap://127.0.
0.1:5783/temperature:
==[ CoAP Request  ]===================================
MID     : -1
Token   : null
Type    : CON
Method  : GET
Options: {"Accept":"application/senml+json"}
Payload: 0 Bytes
=======================================================
```

```
11:07:51.726 [main] INFO  i.u.w.t.m.c.
CoapGetClientProcess - Response :
==[ CoAP Response ]===================================
MID     : 21645
Token   : 88616A9FDE0F5BD6
Type    : ACK
Status  : 2.05 - CONTENT
Options: {}
RTT     : 16 ms
Payload: 85 Bytes
-------------------------------------------------------
{"timestamp":1665047267049,"type":"iot.sensor.
temperature","data":26.954350537153328}
=======================================================
```

Conclusioni

Sono stati **creati due moduli aggiuntivi della libreria WLDT che mappano i protocolli CoAP e MQTT**.

Vengono superate le limitazioni e le caratteristiche dei singoli protocolli consentendo un'**interazione più ampia tra oggetti IoT**, in modo da fornire la possibilità a oggetti differenti e consumer differenti di comunicare correttamente e in modo interoperabile.



PROGETTAZIONE E RELIZZAZIONE DI DIGITAL TWIN INTEROPERABILI PER SCENARI APPLICATIVI DI INTERNET OF THINGS

Monika Ghidotti

Relatore: Prof. Ing. Marco Picone

20 Ottobre 2022

CoAP to MQTT Worker - Configurazione

Affinchè il Worker funzioni, è necessaria una ***configurazione***, creata dall'utilizzatore della libreria prima di avviare il Worker.
La configurazione contiene informazioni quali:

```
{
    destinationBrokerAddress='127.0.0.1',
    destinationBrokerPort=1883,
    destinationBrokerClientId='digitaltwin0001',
    deviceId='com:iot:coap2mqtt:coap2mqttDevice001',
    brokerLocal=true,
    coapResource2MqttTopic={
        /irrigation/status={resource_id=status_resource,
                            get_topic=/device/test/irrigation/status,
                            post_topic=/device/test/irrigation/status/change,
                            put_topic=/device/test/irrigation/status/action},
        /irrigation/level={resource_id=level_resource,
                            get_topic=/device/test/irrigation/level,
                            post_topic=/device/test/irrigation/level/change
                            put_topic=/device/test/irrigation/level/action}
    },
    resourceDiscovery=true,
    deviceAddress='127.0.0.1',
    physicalDevicePort=5683
}
```

Configurazione MQTT

Mapping da CoAP a MQTT

Configurazione CoAP

- Indirizzo e porta del Borker
- Client ID e Device ID
- Una **mappa** che *associa ad ogni risora*
  - *i topic su cui pubblicare il messaggio di telemetria* (*get_topic*)
  - *i topic ai quali si deve sottoscrivere il Worker per ricevere i messaggi di controllo (post_topic, put_topic)*
- Indirizzo e porta dell'oggetto CoAP

MQTT to CoAP Worker - Configurazione

```
{
    brokerAddress='127.0.0.1',
    brokerPort=1883,
    brokerClientId='digitaltwin0002',
    deviceId='com:iot:mqtt2coap:mqtt2coapDT',
    topicList=[
        {id='battery0001',
        resourceId='battery',
        topic='/device/test/sensor/battery',
        publishQosLevel=MQTT_QOS_2},
        {id='temperature0001',
        resourceId='temperature',
        topic='/device/test/sensor/temperature',
        publishQosLevel=MQTT_QOS_2}
    ],
    configurationDescriptorList=[
        {deviceAddress='127.0.0.1',
        devicePort=5783,
        id='battery',
        isObservable=false,
        title='battery_sensor',
        resourceType='batterySensor',
        coreInterface='core.s',
        telemetryTopic='/device/test/sensor/battery'},
        {deviceAddress='127.0.0.1',
        devicePort=5783,
        id='temperature',
        isObservable=false,
        title='temperature_sensor',
        resourceType='temperatureSensor',
        coreInterface='core.s',
        telemetryTopic='/device/test/sensor/temperature'}
    ]
}
```

Configurazione MQTT

Topic List

Mappa dei descrittori delle risorse CoAP

Affinchè il Worker funzioni, è necessaria una ***configurazione***, creata dall'utilizzatore della libreria prima di avviare il Worker.

La configurazione contiene informazioni quali:

- Indirizzo e porta del Broker MQTT
- ID del client e ID del dispositivo
- Lista di Topic
- Lista dei **descrittori delle risorse CoAP**, all'interno della quale sono presenti:
    - Inidirizzo e porta della risorsa
    - Id della risorsa
    - Attributi CoRE Link Format: *resource type, core interface, title* e *obs*.
    - Topic di telemetria associato alla risorsa