Sensorwave

Applicazione IoT per il monitoraggio e il controllo in tempo reale di sensori

Raffaele Di Maso
N97000411

Vincenzo Tramo
N97000433

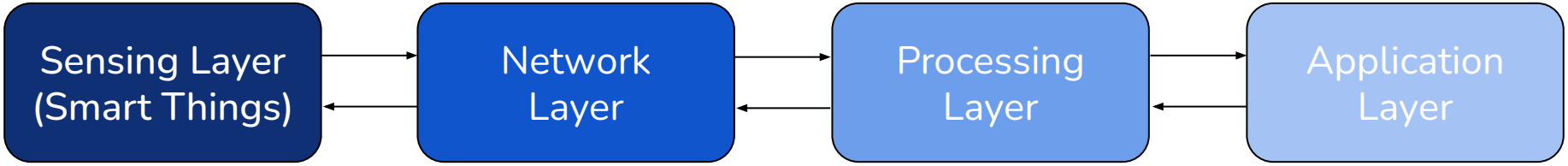
22 Settembre, 2023
Università di Napoli, Federico II

L’idea di Sensorwave

- Piattaforma IoT per la *raccolta*, *l'elaborazione*, la *visualizzazione* e la *gestione* dei dispositivi (o **smart objects**) in *tempo reale*

- Semplificare il *monitoraggio* e il *controllo* dei sensori in una *varietà di contesti*, dai settori industriali all'ambito domestico

Key Building Blocks



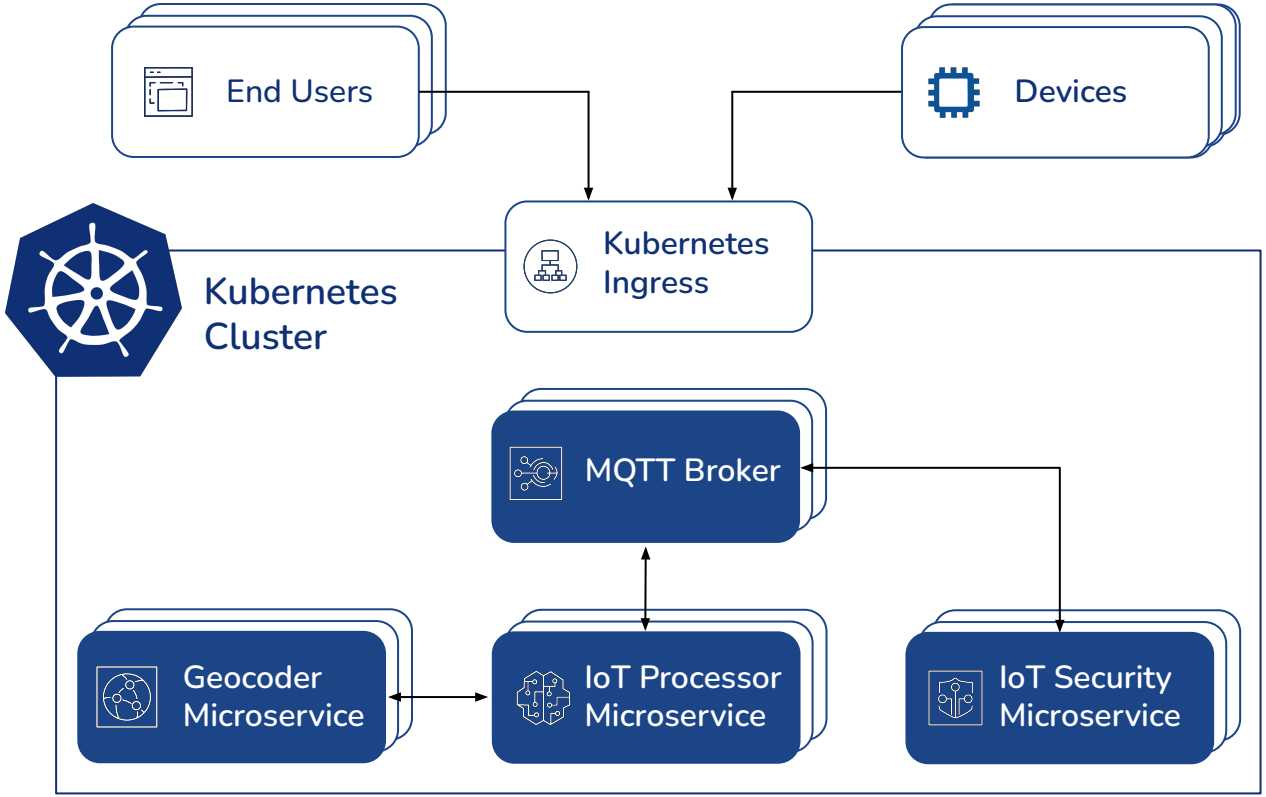
- ❏ Temperature
- ❏ Humidity
- ❏ GPS
- ❏ Raspberry Pi
- ❏ …

- ❏ Ethernet
- ❏ Wireless
- ❏ 4G, 5G
- ❏ Bluetooth
- ❏ …

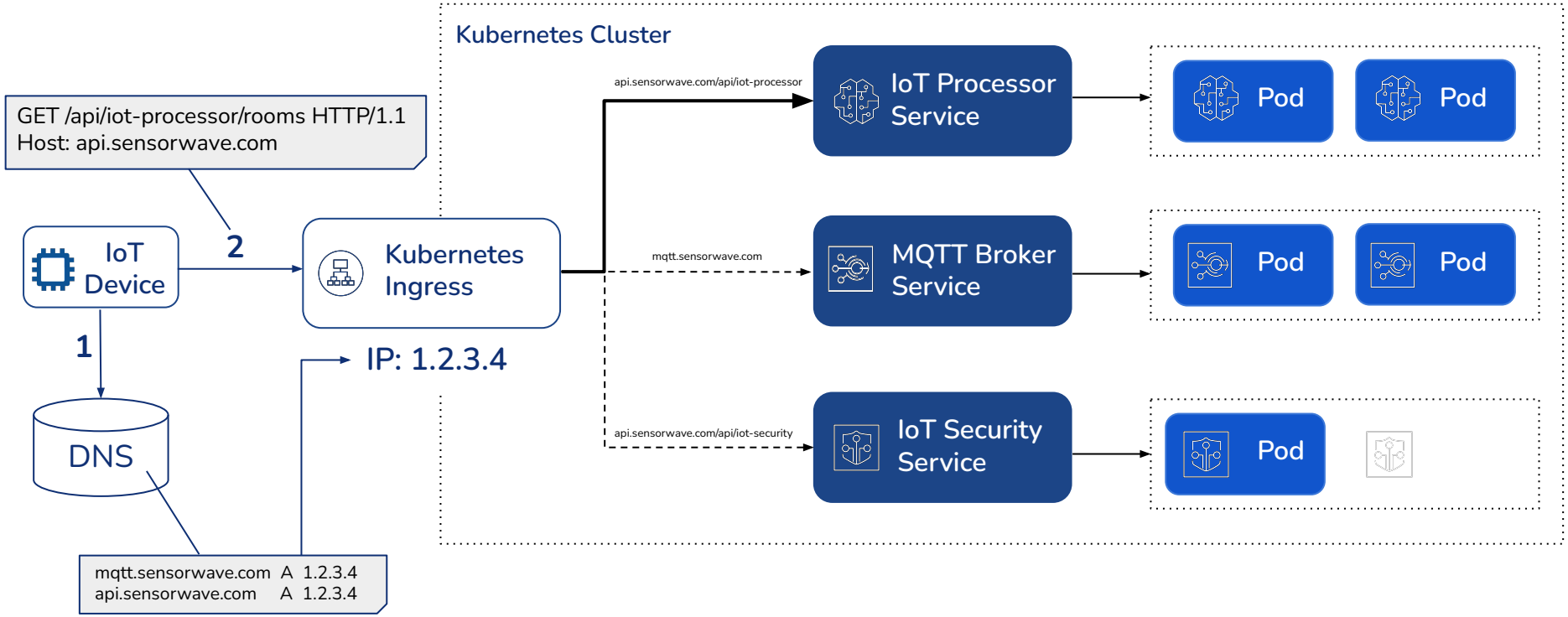
- ❏ Algoritmi
- ❏ Database
- ❏ Server
- ❏ …

- ❏ Web
- ❏ Mobile
- ❏ Smartwatch
- ❏ …

Sensorwave Architecture

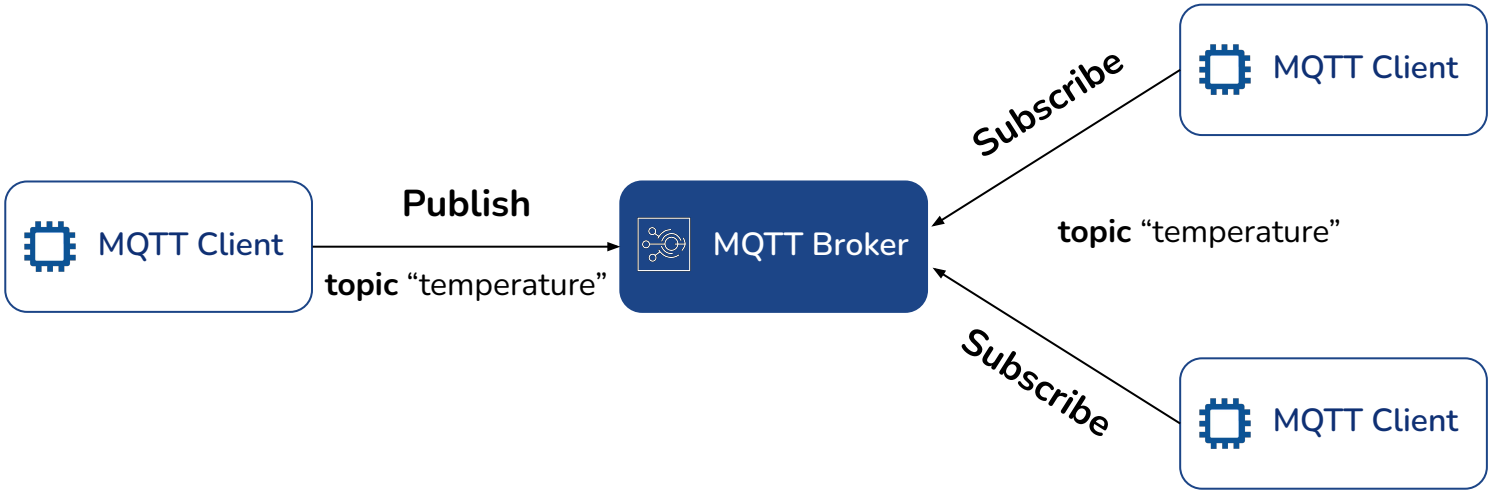


Kubernetes Ingress

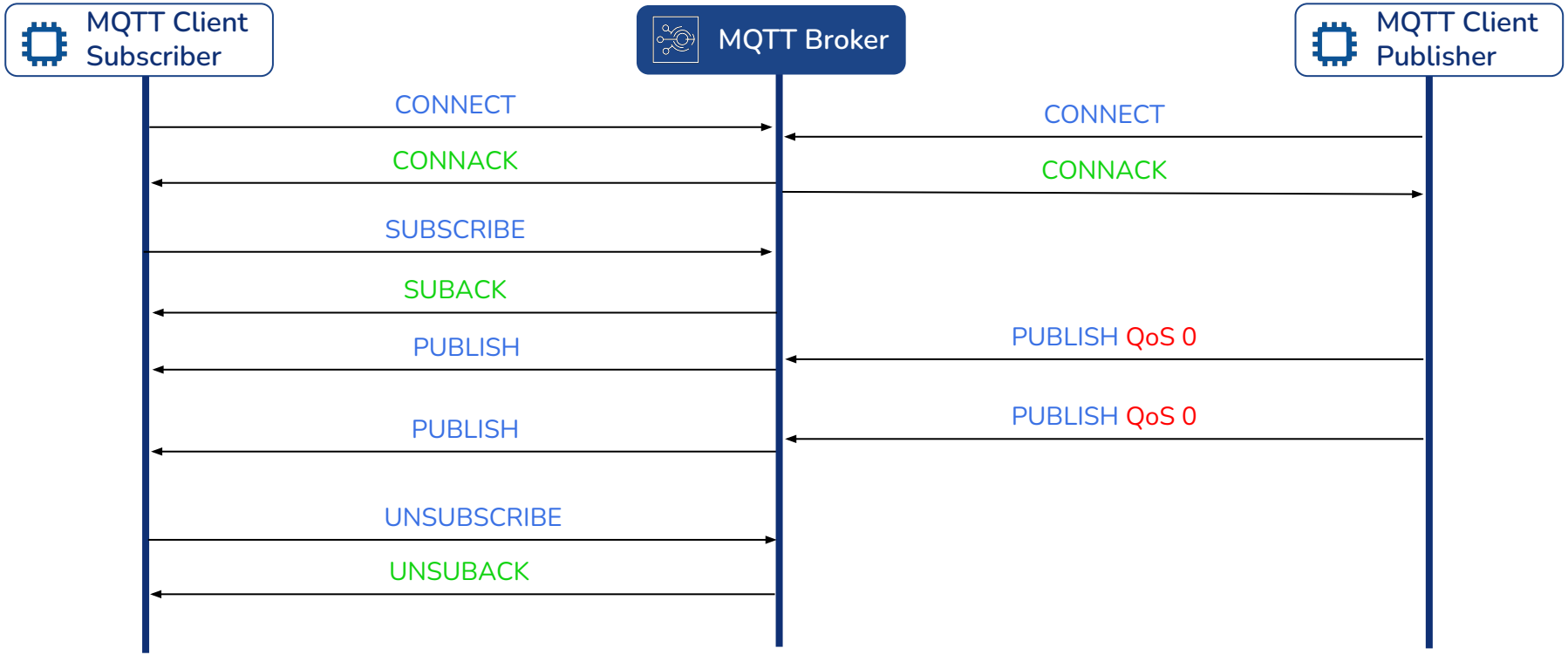


Protocollo MQTT

- È il **protocollo di messaggistica** più usato nell'ambito IoT
- Basato sul design pattern **publish/subscribe**
- Basato su **TCP/IP**



MQTT Flow

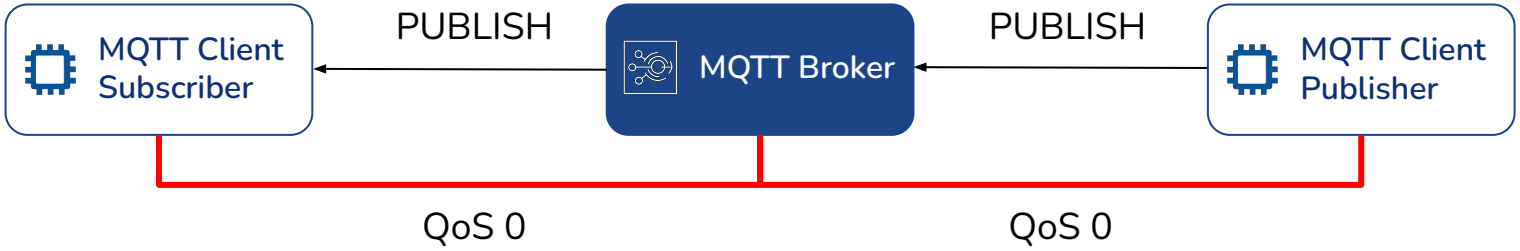

MQTT Client Subscriber
MQTT Broker
MQTT Client Publisher
CONNECT
CONNECT
CONNACK
CONNACK
SUBSCRIBE
SUBACK
PUBLISH
PUBLISH QoS 0
PUBLISH
PUBLISH QoS 0
UNSUBSCRIBE
UNSUBACK

MQTT QoS

- Il Quality of Service (QoS) è una misura che indica il **livello di garanzia di consegna dei messaggi** tra il *mittente* e il *destinatario*
- Esistono tre livelli di garanzia:
  - *At Most Once* (QoS 0)
  - *At Least One* (QoS 1)
  - *Exactly One* (QoS 2)

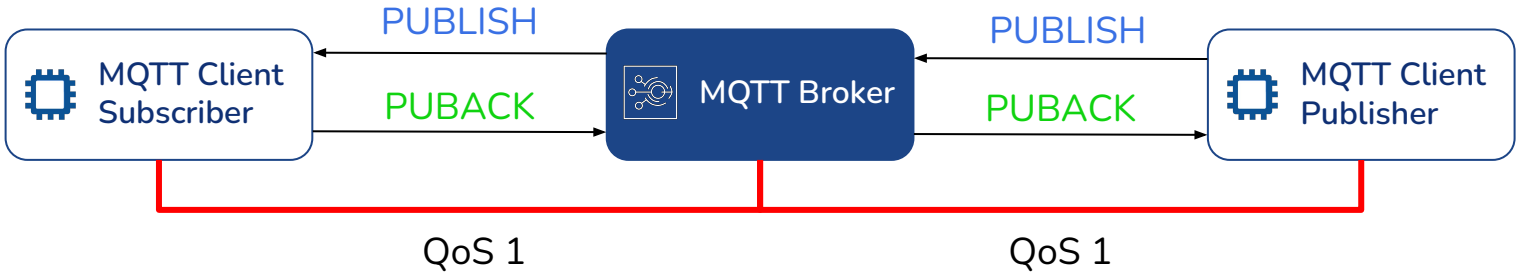
MQTT QoS 0 - *At Most Once*

- Nessuna garanzia di consegna
- Il destinatario non conferma la ricevuta del messaggio
- "*Fire AND Forget*" (stessa garanzia del protocollo TCP)
- Molto veloce

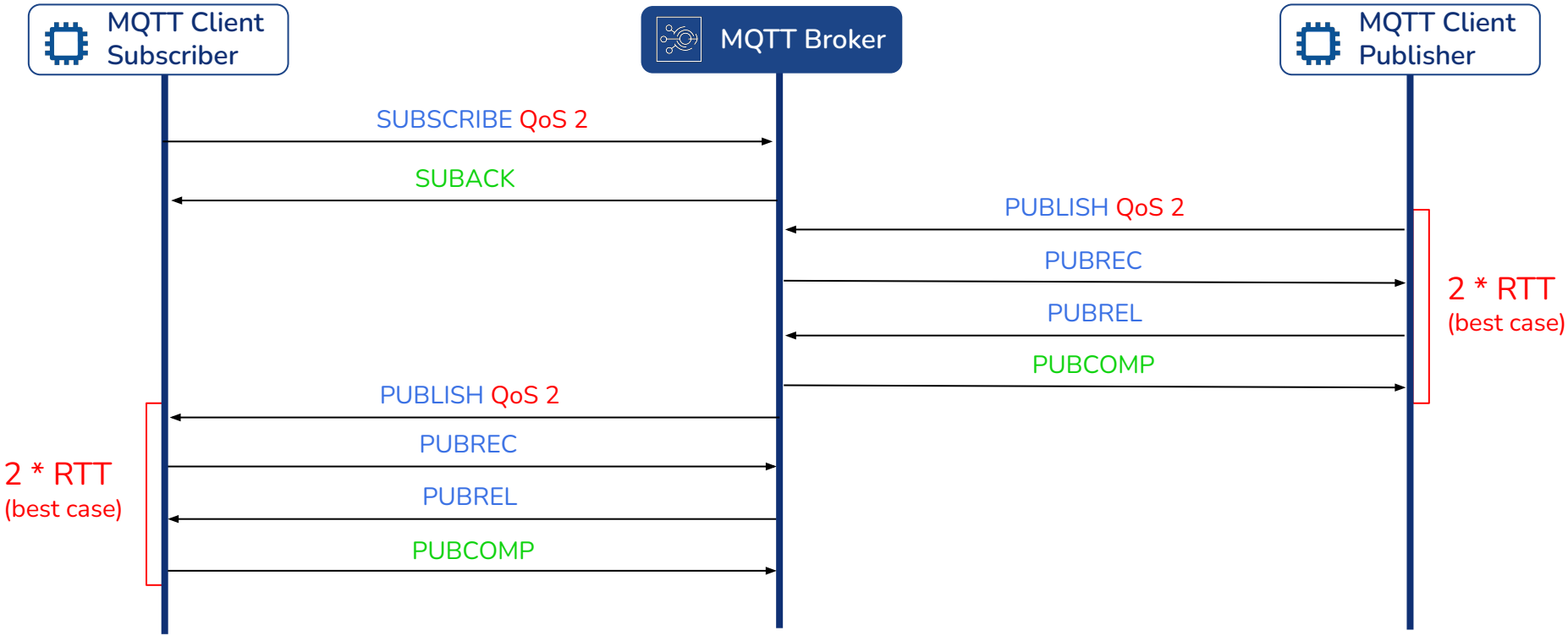


MQTT QoS 1 - *At Least Once*

- Il messaggio inviato dal mittente viene ricevuto **almeno una volta** dal destinatario
- Il mittente memorizza il messaggio fino a quando non riceve un **PUBACK packet**
- È possibile che un messaggio venga inviato più di una volta
- È il livello di garanzia più utilizzato

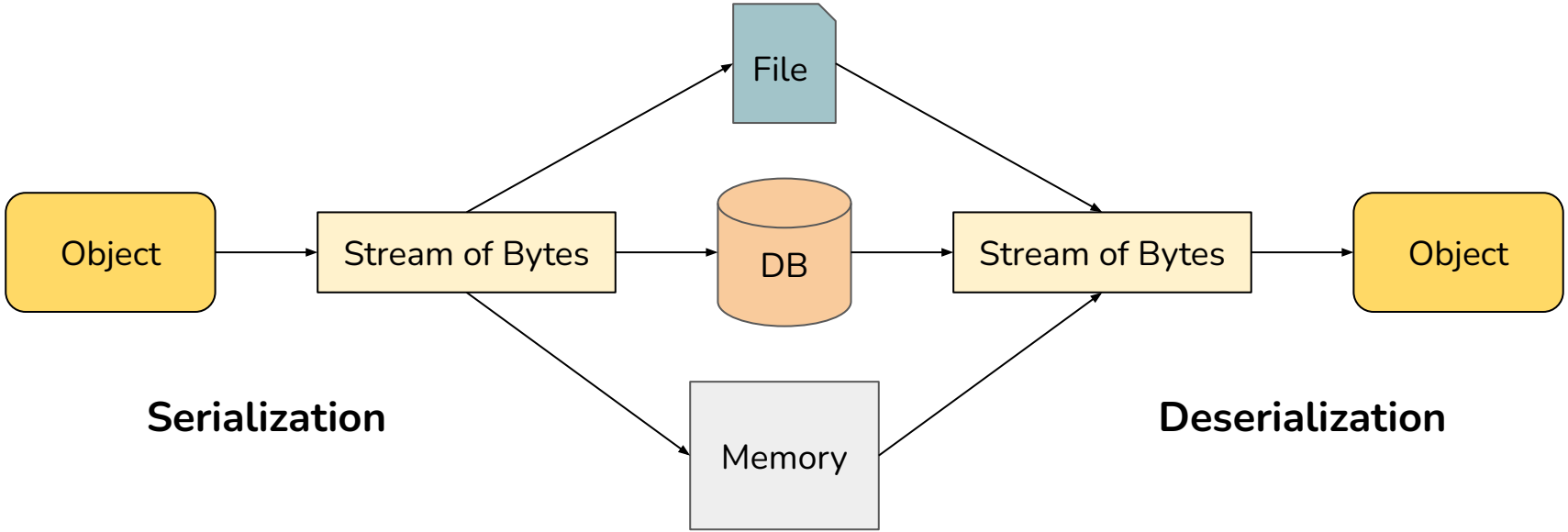


MQTT QoS 2 - *Exactly Once*



Protocol Buffers

- È un meccanismo veloce per la **serializzazione/deserializzazione dei dati**
- Utilizza un **formato dei messaggi binario**
- Permette di definire uno **schema dei dati**



Protocol Buffers

```
syntax = "proto3";

option java_multiple_files = true;
option java package = "com.sensorwave.iot processor";
option java_outer classname = "Protobuf";

import "google/protobuf/any.proto";
import "google/protobuf/timestamp.proto";

message SmartObjectMessage {
 string smartObjectId = 1;
 string roomId = 2;
 optional google.protobuf.Timestamp timestamp = 3;
 repeated Data data = 4;
}
...
```

```
...

enum DataType {
 TEMPERATURE = 0;
 POSITION = 1;
 HUMIDITY = 2;
 STATUS = 3;
}

message Data {
 DataType type = 1;
 google.protobuf.Any data = 2;
}

message Temperature { double temperature = 1; }

message Position {
 double longitude = 1;
 double latitude = 2;
}

message Humidity { double humidity = 1; }

message Status { bool isOnline = 1; }
```

Protocol Buffers

```
syntax = "proto3";

option java_multiple_files = true;
option java package = "com.sensorwave.iot processor";
option java_outer classname = "Protobuf";

import "google/protobuf/any.proto";
import "google/protobuf/timestamp.proto";

message SmartObjectMessage {
 string smartObjectId = 1;
 string roomId = 2;
 optional google.protobuf.Timestamp timestamp = 3;
 repeated Data data = 4;
}
...
```

```
...

enum DataType {
 TEMPERATURE = 0;
 POSITION = 1;
 HUMIDITY = 2;
 STATUS = 3;
}

message Data {
 DataType type = 1;
 google.protobuf.Any data = 2;
}

message Temperature { double temperature = 1; }

message Position {
 double longitude = 1;
 double latitude = 2;
}

message Humidity { double humidity = 1; }

message Status { bool isOnline = 1; }
```

Protocol Buffers

```
syntax = "proto3";

option java_multiple_files = true;
option java package = "com.sensorwave.iot processor";
option java_outer classname = "Protobuf";

import "google/protobuf/any.proto";
import "google/protobuf/timestamp.proto";

message SmartObjectMessage {
 string smartObjectId = 1;
 string roomId = 2;
 optional google.protobuf.Timestamp timestamp = 3;
 repeated Data data = 4;
}
...
```

```
...

enum DataType {
 TEMPERATURE = 0;
 POSITION = 1;
 HUMIDITY = 2;
 STATUS = 3;
}

message Data {
 DataType type = 1;
 google.protobuf.Any data = 2;
}

message Temperature { double temperature = 1; }

message Position {
 double longitude = 1;
 double latitude = 2;
}

message Humidity { double humidity = 1; }

message Status { bool isOnline = 1; }
```

Protocol Buffers

```
syntax = "proto3";

option java_multiple_files = true;
option java package = "com.sensorwave.iot processor";
option java_outer classname = "Protobuf";

import "google/protobuf/any.proto";
import "google/protobuf/timestamp.proto";

message SmartObjectMessage {
 string smartObjectId = 1;
 string roomId = 2;
 optional google.protobuf.Timestamp timestamp = 3;
 repeated Data data = 4;
}
...
```

```
...

enum DataType {
 TEMPERATURE = 0;
 POSITION = 1;
 HUMIDITY = 2;
 STATUS = 3;
}

message Data {
 DataType type = 1;
 google.protobuf.Any data = 2;
}

message Temperature { double temperature = 1; }

message Position {
 double longitude = 1;
 double latitude = 2;
}

message Humidity { double humidity = 1; }

message Status { bool isOnline = 1; }
```

Protocol Buffers

```
syntax = "proto3";

option java_multiple_files = true;
option java package = "com.sensorwave.iot processor";
option java_outer classname = "Protobuf";

import "google/protobuf/any.proto";
import "google/protobuf/timestamp.proto";

message SmartObjectMessage {
 string smartObjectId = 1;
 string roomId = 2;
 optional google.protobuf.Timestamp timestamp = 3;
 repeated Data data = 4;
}
...
```

```
...

enum DataType {
 TEMPERATURE = 0;
 POSITION = 1;
 HUMIDITY = 2;
 STATUS = 3;
}

message Data {
 DataType type = 1;
 google.protobuf.Any data = 2;
}

message Temperature { double temperature = 1; }

message Position {
 double longitude = 1;
 double latitude = 2;
}

message Humidity { double humidity = 1; }

message Status { bool isOnline = 1; }
```

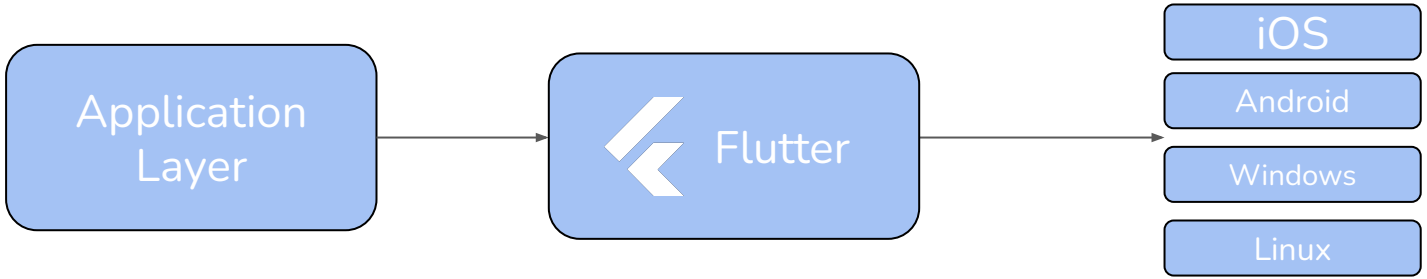
Protocol Buffers

```
syntax = "proto3";

option java_multiple_files = true;
option java package = "com.sensorwave.iot processor";
option java_outer classname = "Protobuf";

import "google/protobuf/any.proto";
import "google/protobuf/timestamp.proto";

message SmartObjectMessage {
 string smartObjectId = 1;
 string roomId = 2;
 optional google.protobuf.Timestamp timestamp = 3;
 repeated Data data = 4;
}
...
```

```
...

enum DataType {
 TEMPERATURE = 0;
 POSITION = 1;
 HUMIDITY = 2;
 STATUS = 3;
}

message Data {
 DataType type = 1;
 google.protobuf.Any data = 2;
}

message Temperature { double temperature = 1; }

message Position {
 double longitude = 1;
 double latitude = 2;
}

message Humidity { double humidity = 1; }

message Status { bool isOnline = 1; }
```

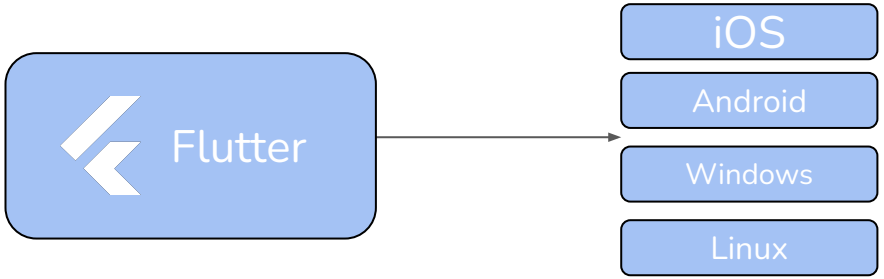
Application Layer

- Autenticazione utente
- Gestione delle stanze e *smart objects*
- Monitoraggio dei sensori (client MQTT)



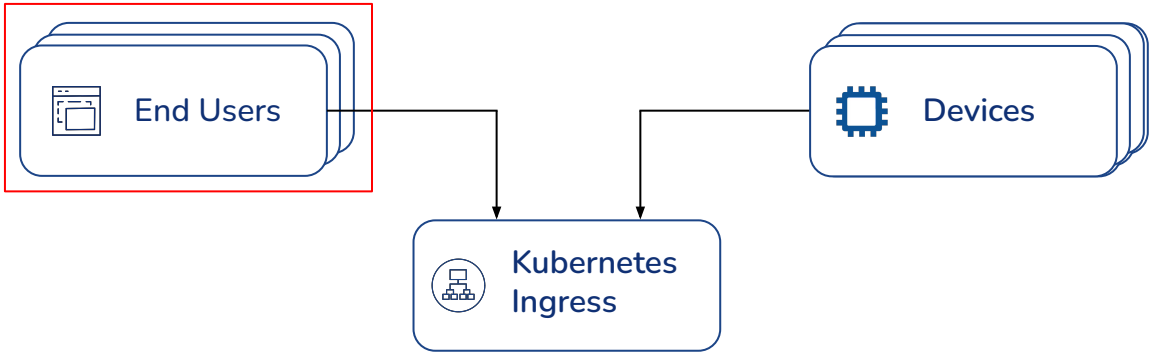
Application Layer

- **Flutter** è un framework open-source per UI
- E’ impiegato per lo sviluppo di app native

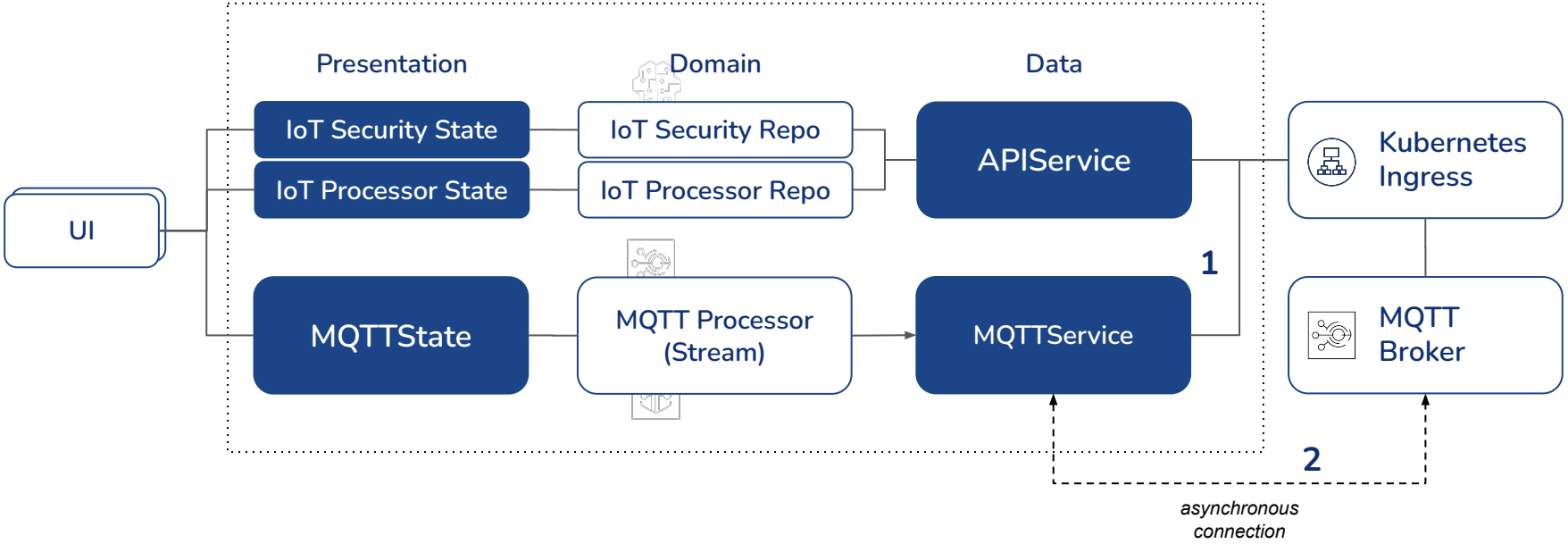


Dashboard

- La **dashboard** è l'interfaccia utente per la gestione e il monitoraggio dei sensori.

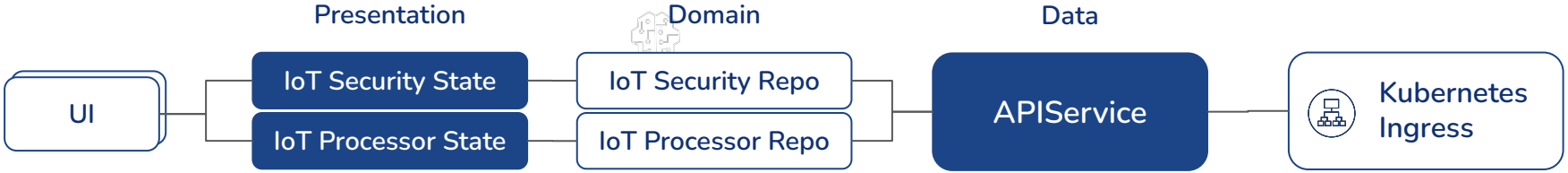


Dashboard Architecture



Dashboard Architecture

- Autenticazione utente
- Gestione delle stanze e *smart objects*
- Rest API



Dashboard: iot-security

```
POST
http://{{keycloak-host}}/realms/quarkus/protocol/open
id-connect/token
Content-Type: application/x-www-form-urlencoded

client_id = CLIENT_ID &
client_secret = CLIENT_SECRET &
grant_type = password &
username = USERNAME &
password = PASSWORD &
scope = openid
```

generate_user_access_token

- Keycloak API
- Login e registrazione
- Ottenere un ***token***

Dashboard: iot-processor

```
POST http://{{iot-processor-host}}/api/iot-processor/rooms
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "name": NOME_STANZA
}
```

create_room

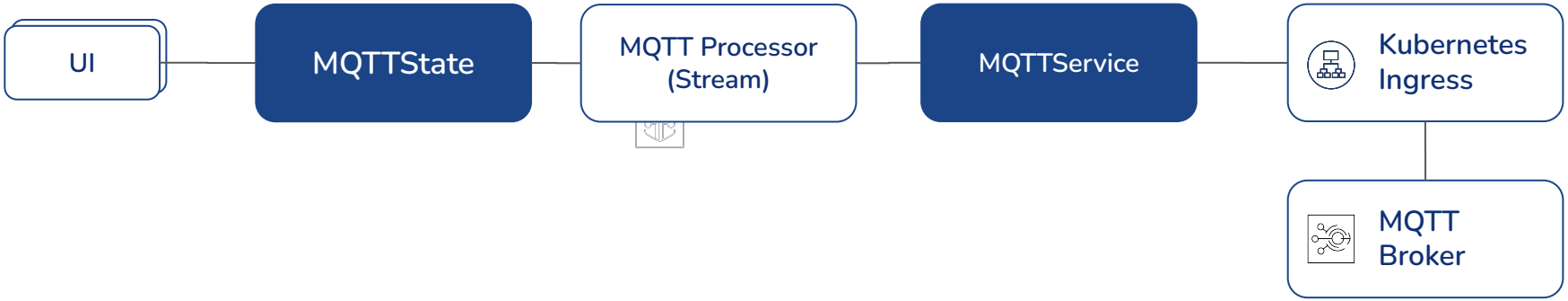
```
POST
http://{{iot-processor-host}}/api/iot-processor/rooms/room/smartobjects
Content-Type: application/json
Authorization: Bearer TOKEN

{
  "name": NOME_SMART_OBJECT,
  "roomOwnerUsername": USERNAME
}
```

create_smart_object

- Creazione stanza
- Creazione smart object
- Ottenere stanze

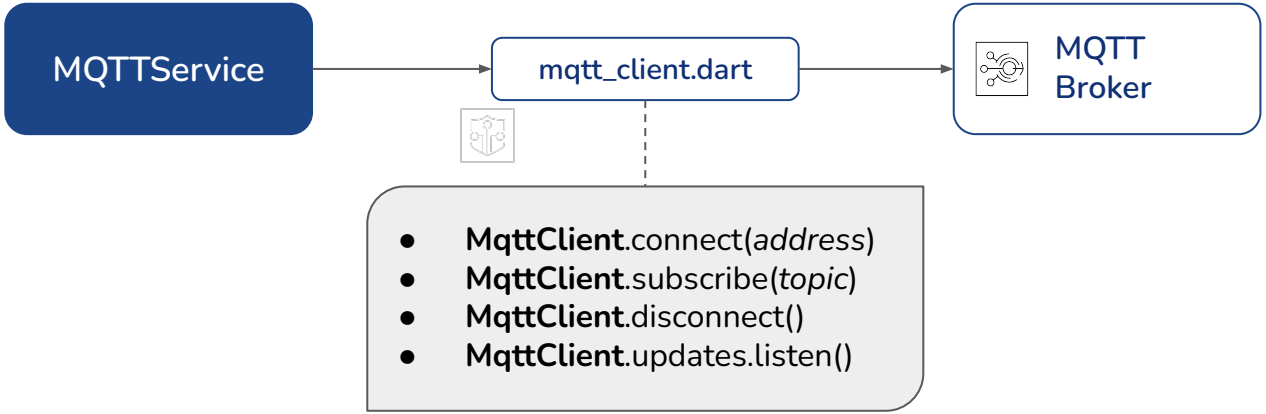
Dashboard Architecture

- Connessione al broker MQTT
- Elaborazione dei messaggi *raw*
- Monitoraggio dei *smart objects*



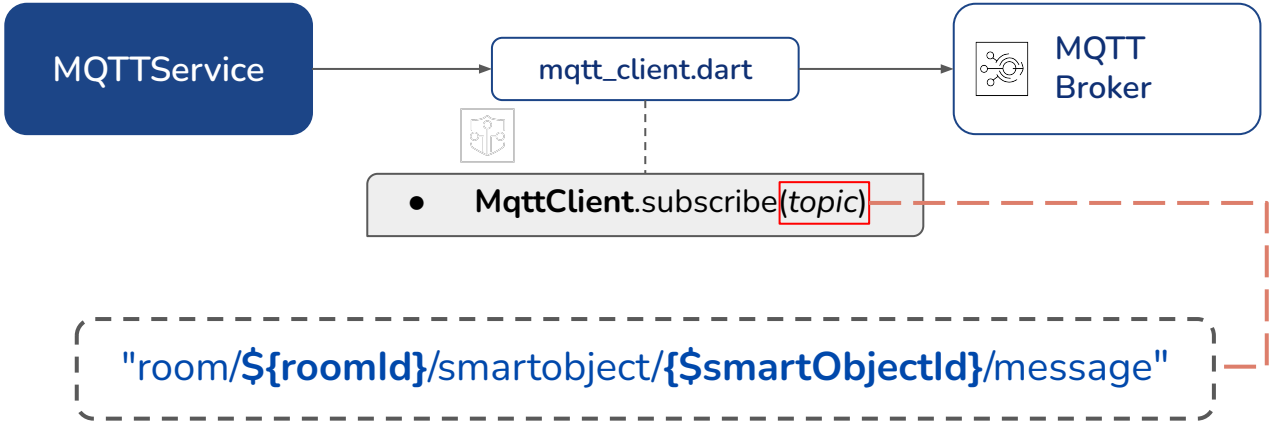
Dashboard: connessione MQTT

- Connessione al broker MQTT
- Libreria **mqtt_client** per *Dart*



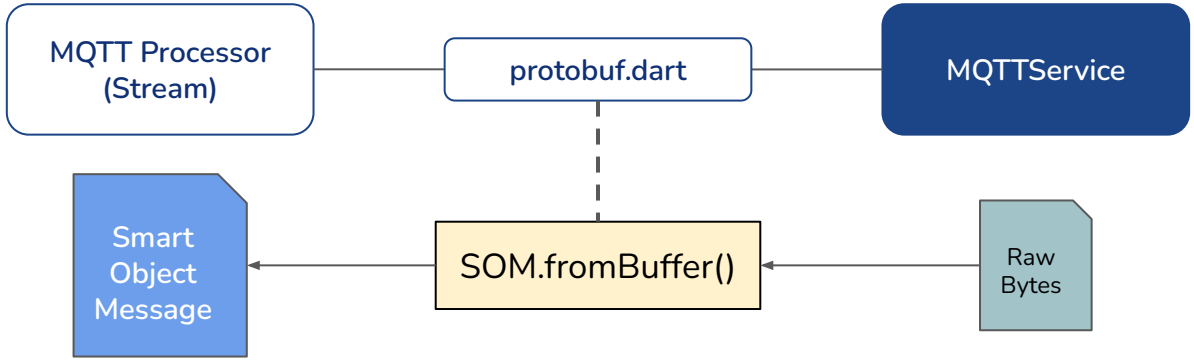
Dashboard: connessione MQTT

- Se il client si connette al broker, sarà necessaria la *sottoscrizione* al topic.



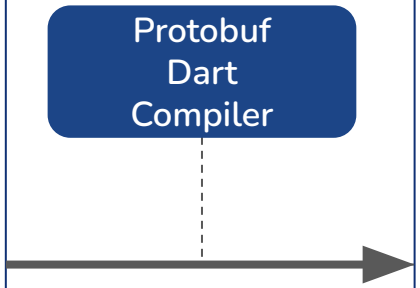
Dashboard: elaborazione messaggi

- I buffer di bytes vengono trasformati in **SmartObjectMessage**.
- L'elaborazione è effettuata con l'impiego della libreria ufficiale di *protobuf*.



Dashboard: elaborazione messaggi

```
...

enum DataType {
 TEMPERATURE = 0;
 POSITION = 1;
 HUMIDITY = 2;
 STATUS = 3;
}

message Data {
 DataType type = 1;
 google.protobuf.Any data = 2;
}

message Temperature { double temperature = 1; }

message Position {
 double longitude = 1;
 double latitude = 2;
}

message Humidity { double humidity = 1; }

message Status { bool isOnline = 1; }
```

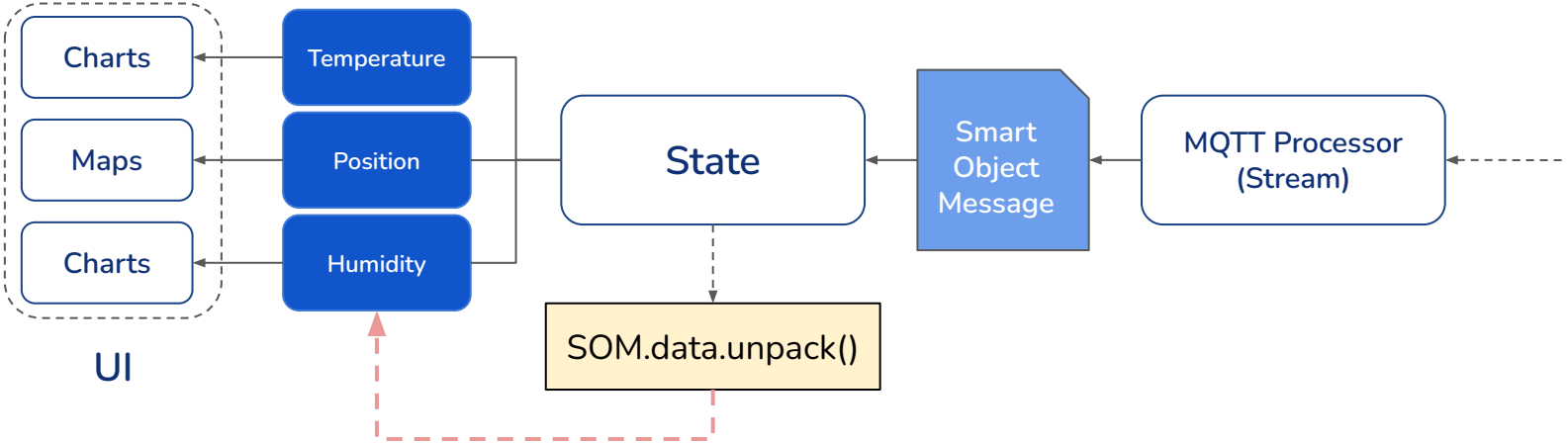


- Il compilatore Protobuf converte file .proto generici in modelli del linguaggio richiesto (*.dart* nel caso corrente)

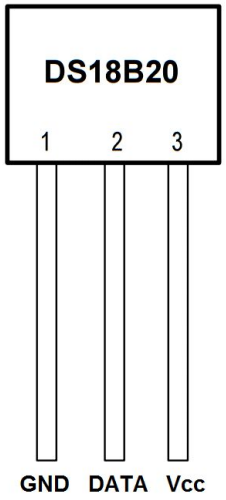
```
class SmartObjectMessage extends
$pb.GeneratedMessage {
  factory SmartObjectMessage({
    $core.String? smartObjectId,
    $core.String? roomId,
    $0.Timestamp? timestamp,
    $core.Iterable<Data>? data,
  }) {
    ...
  }
  SmartObjectMessage._() : super();
  factory
SmartObjectMessage.fromBuffer($core.List<$core.i
nt> i, [$pb.ExtensionRegistry r =
$pb.ExtensionRegistry.EMPTY]) =>
create()..mergeFromBuffer(i, r);
  factory
SmartObjectMessage.fromJson($core.String i,
[$pb.ExtensionRegistry r =
$pb.ExtensionRegistry.EMPTY]) =>
create()..mergeFromJson(i, r);
...
```

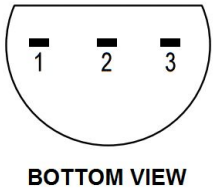
Dashboard: elaborazione messaggi

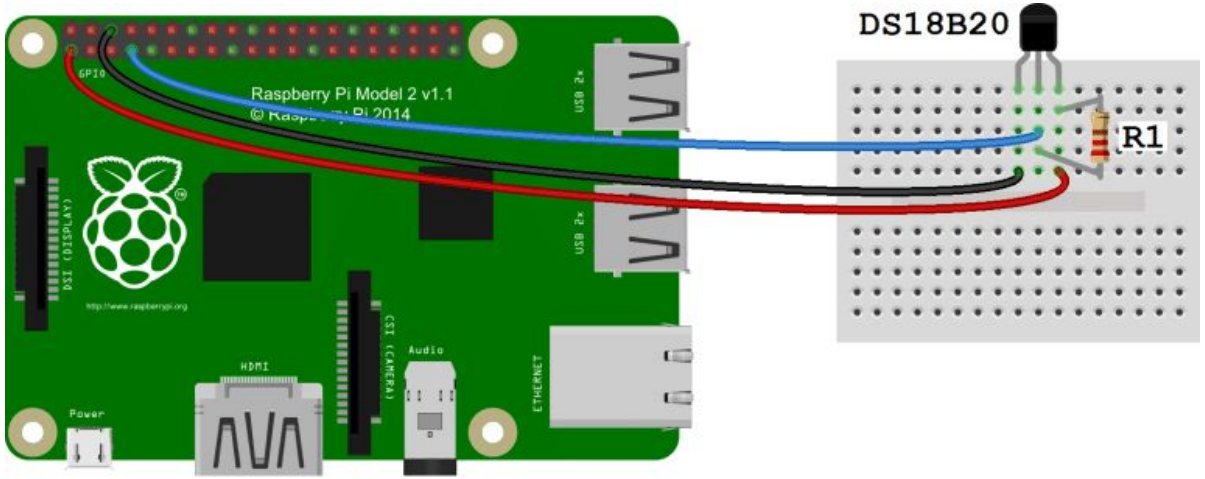
- Lo stato dei widget è in ascolto sullo stream (**Stream<SmartObjectMessage>**)
- I widget (grafici, mappe...) vengono aggiornati.



Sensorwave Demo - Raspberry Pi

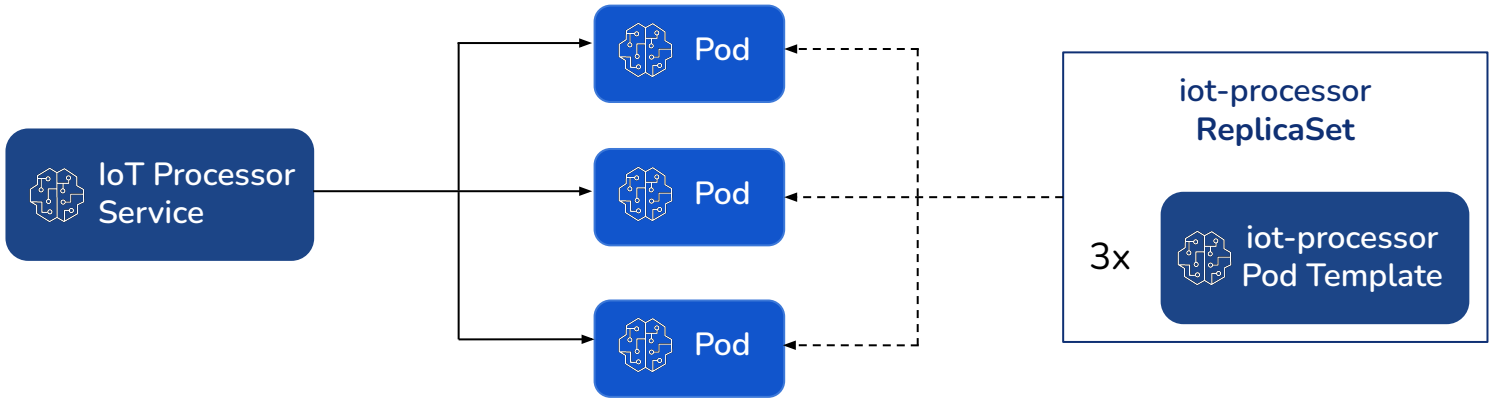






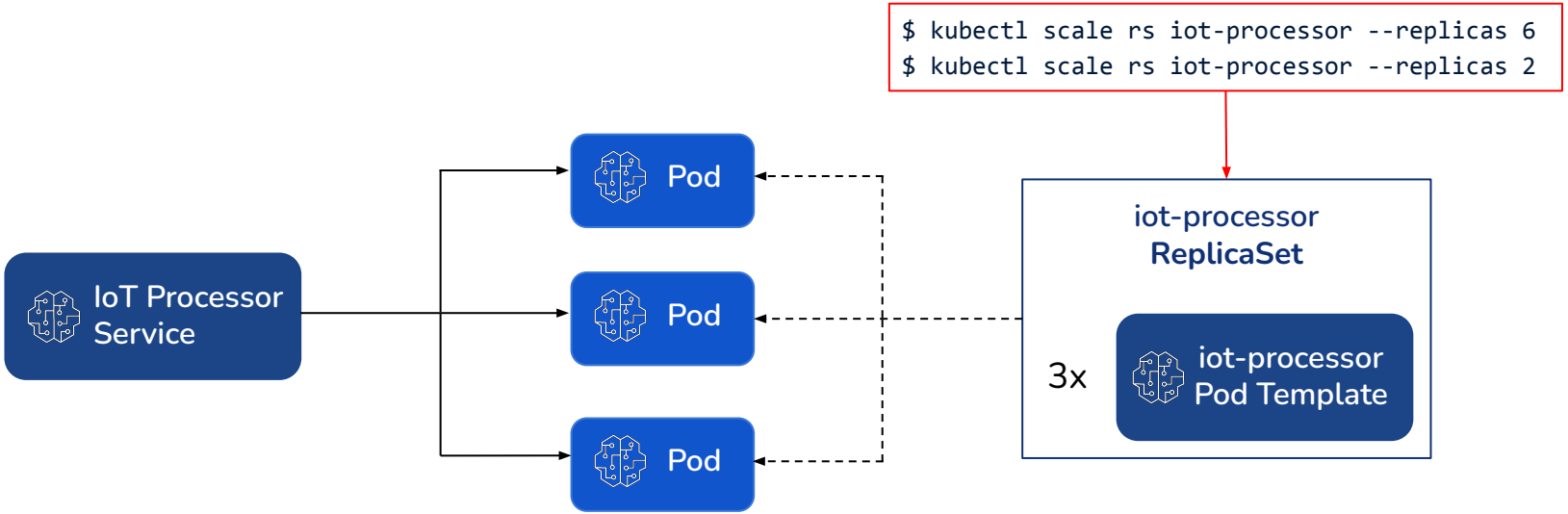
Kubernetes Pod Scaling

- L'oggetto **ReplicaSet** rappresenta un gruppo di repliche di Pod (copie esatte di Pod)



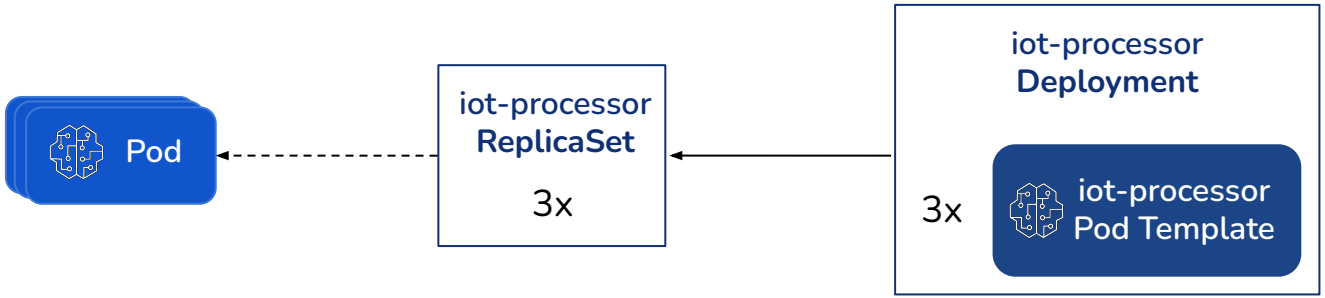
Kubernetes Pod Scaling

- L'oggetto **ReplicaSet** rappresenta un gruppo di repliche di Pod (copie esatte di Pod)



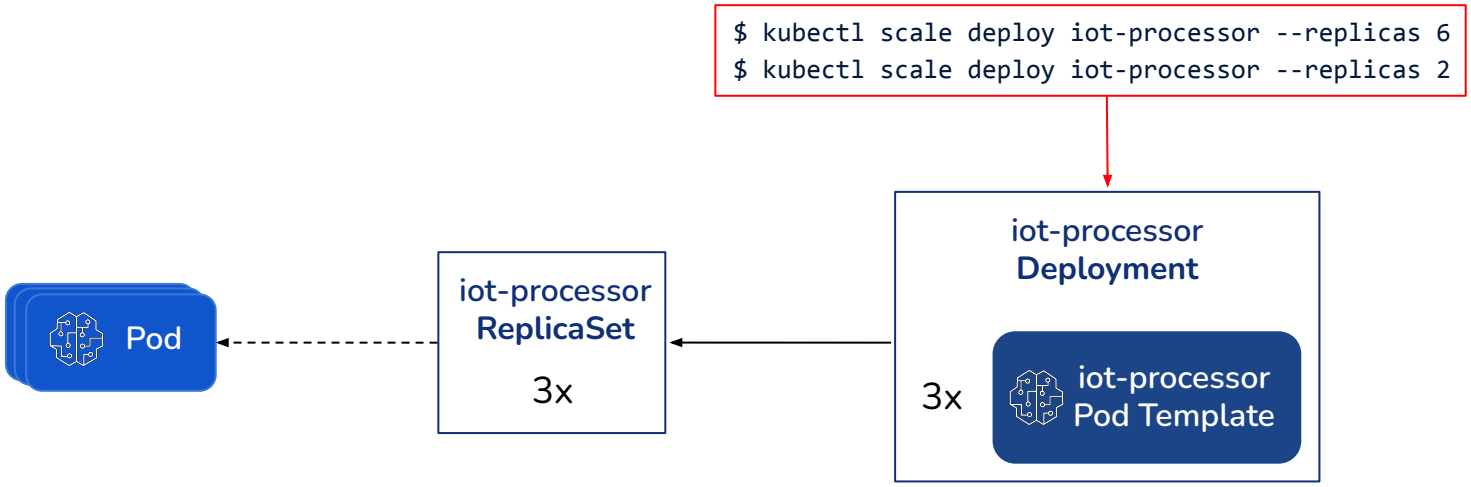
Kubernetes Pod Scaling

- L’oggetto **Deployment** è un costrutto di più alto livello che estende le funzionalità dell'oggetto ReplicaSet



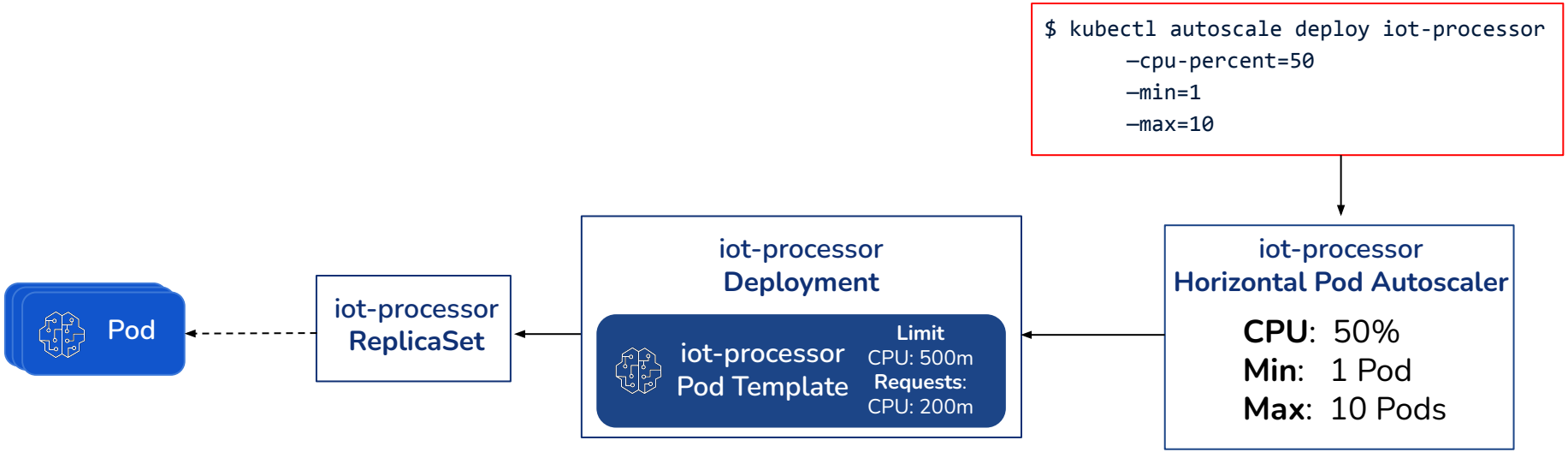
Kubernetes Pod Scaling

- L'oggetto **Deployment** è un costrutto di più alto livello che estende le funzionalità dell'oggetto ReplicaSet



Kubernetes Pod Auto-Scaling

- Un **HorizontalPodAutoscaler** regola periodicamente il numero di Pod ad esempio di un Deployment per adattarlo alle metriche osservate, come l'utilizzo medio della CPU e l'utilizzo medio della memoria



Grazie per l’attenzione

https://github.com/vtramo/sensorwave

https://github.com/ralphthegod/sensorwave_flutter