MQTT Topic/Payload

Publishers and Subscribers

- MQTT works on the principle of a central hub called a **broker**
- One or more **Publishers** and **Subscribers**.
- **One** or **more** entities connected to the service can **Publish** data onto it, and
- **One** or **more** of those interested in receiving the data can **Subscribe** to receive notifications when the data arrives.

Publishing to MQTT

When publishing to MQTT you can specify two things

- Topic - a string indicating a named **slot** or **channel** into which data is sent
- Payload - the **content** of the data to be put into that **slot** or **channel**. This is often a **JSON** data structure, but it can be whatever you want.

Topics

- Topics typically take the form of a series of names, separated by slashes
  - Data from thermometer with the name

```
tm-1
```

  - Thermometer is installed in a room numbered r101

```
r101/tm-1
```

  - Thermometer, light sensor, switch and window shader in the same room

```
r101/tm-1
r101/ls-1
r101/sw-1
r101/ws-1
```

Topic Subscribing

When subscribing to a channel

- If you were interested in **ONLY** temperature sensor in room r101

```
r101/tm-1
```

- If you want to be notified of ALL temperature sensors

```
*/tm-1
```

- If you were interested in All devices in room r101

```
r101/*
```

Topic Subscribing

When subscribing to a channel

- If room r101 is in building b2, you may want to use a topic hierarchy that allows for more granularity

```
b2/f8/r101/tm-1
```

Think of this as a hierarchy:

```
  b2 +
     |
     + f8 +
          |
          + r101 +
                 |
                 + tm -1
```

Topic Subscribing

Single and multiple levels of wildcards with # and *

- From **building/floor/room/device** topics
- subscribe to #/f8/#/#
- subscribe to **all devices** from **any room** and device on the 8th floor of any building

Payload

- The payload is the data written into that **named slot or channel**
- Could be anything you want but is often structured into a **JSON** packet for **easier debugging** and legibility

Payload

- A temperature sensor may want to send the current temperature reading into its own channel:

```
Topic: b2/f8/r101/tm-1
Payload: {
         "temp": 78.3,
         "units": "F",
         "timestamp": 3221188
         }
```

Payload

you could just as well put the data that you might otherwise put in the topic in the payload itself

-
```
Topic: b2/f8/r101/tm-1
Payload: {
         "temp": 78.3,
         "units": "F",
         "timestamp": 3221188
         }
```

-
```
Topic: mydata
Payload: {
         "building": "b2",
         "floor": "f8",
         "room": "r101",
         "device": "tm-1",
         "temp": 78.3,
         "units": "F",
         "timestamp": 3221188
         }
```

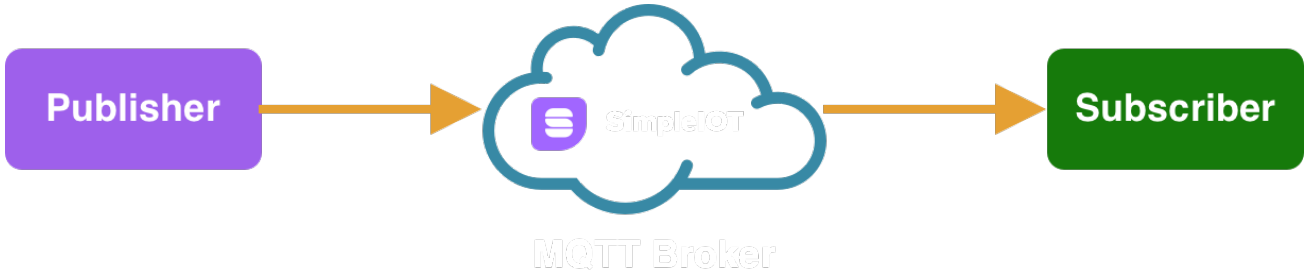
Payload

```
Topic: mydata
Payload: {
         "building": "b2",
         "floor": "f8",
         "room": "r101",
         "device": "tm-1",
         "temp": 78.3,
         "units": "F",
         "timestamp": 3221188
         }
```

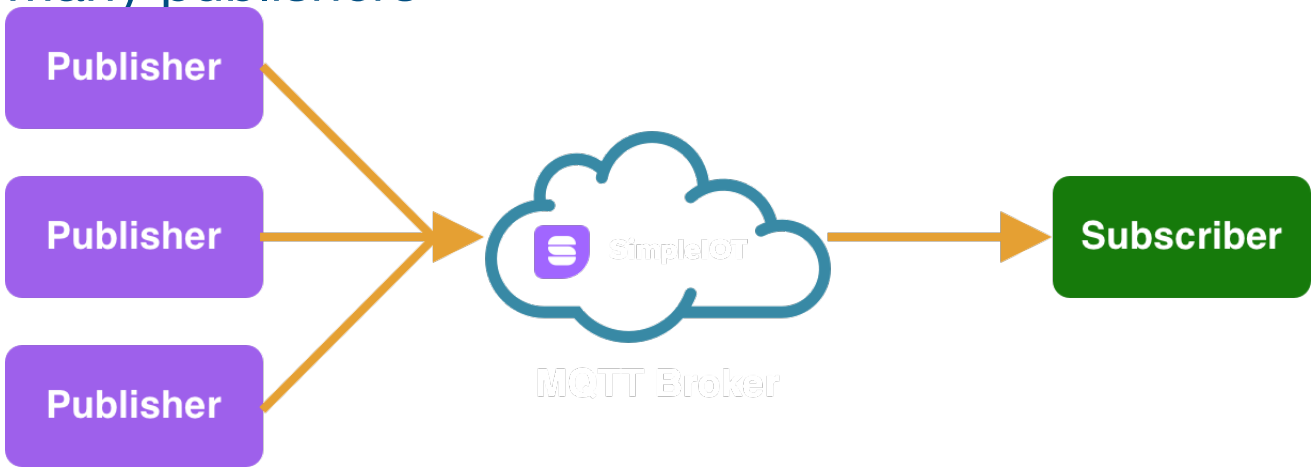
- you would no longer be able to do wildcard filtering based on the topic hierarchy

PubSub modes

- One-to-One

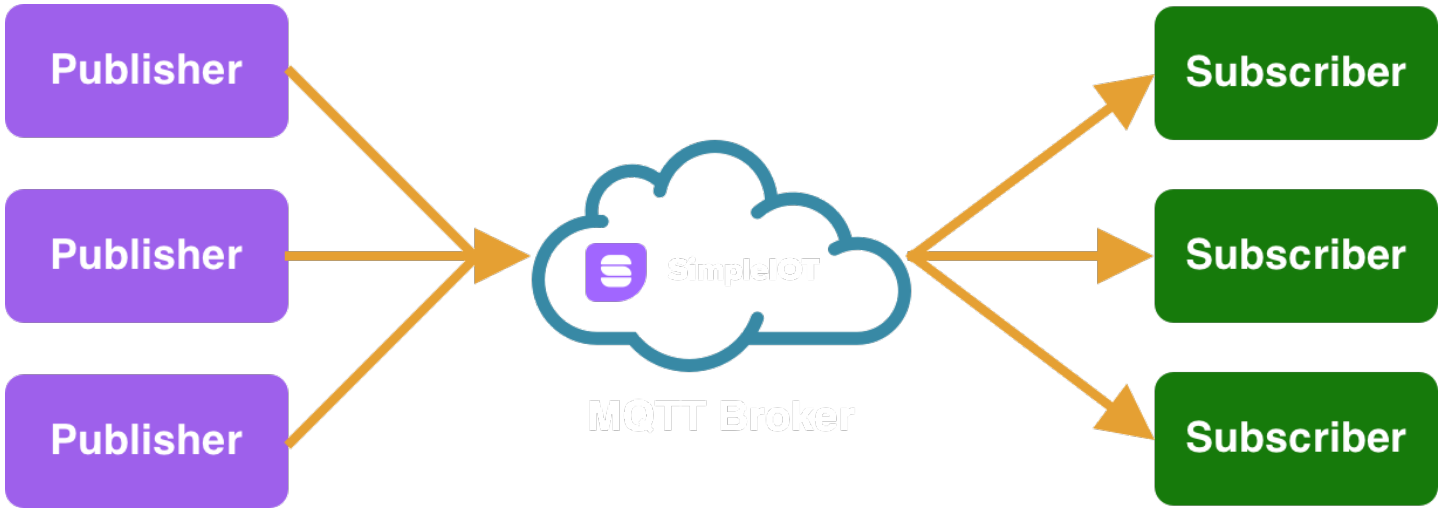


- Many publishers



PubSub modes

- Many publishers and many subscribers

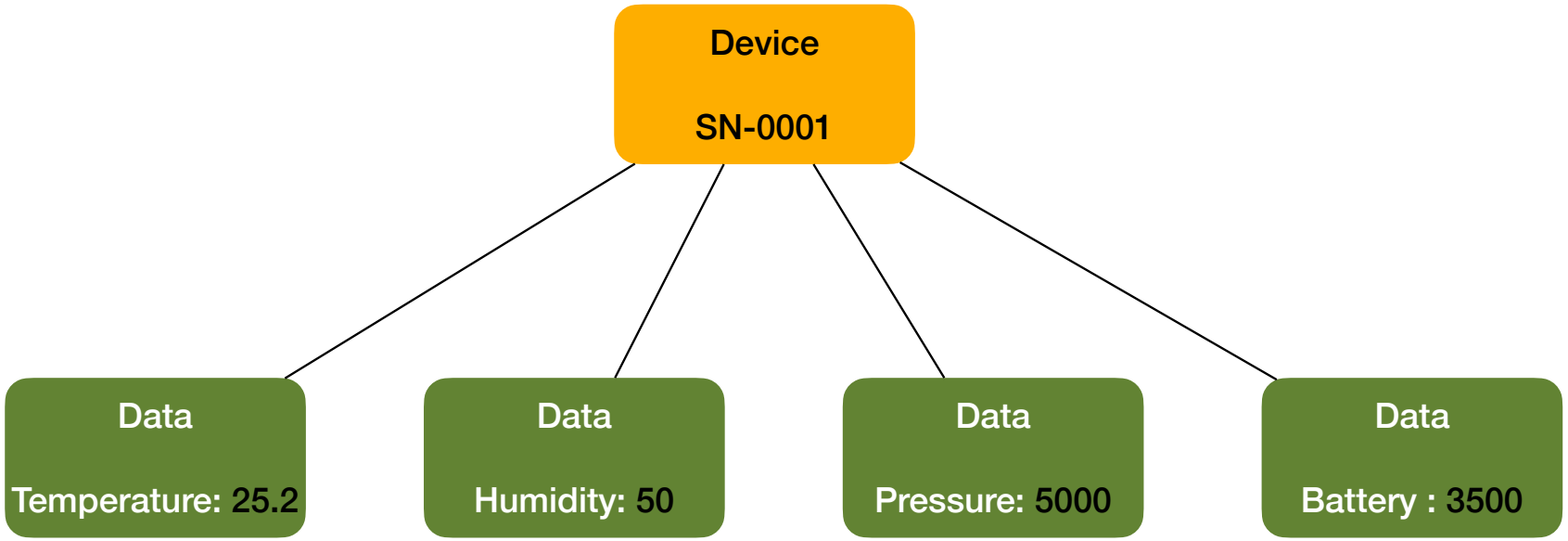


MQTT design pattern

1-device

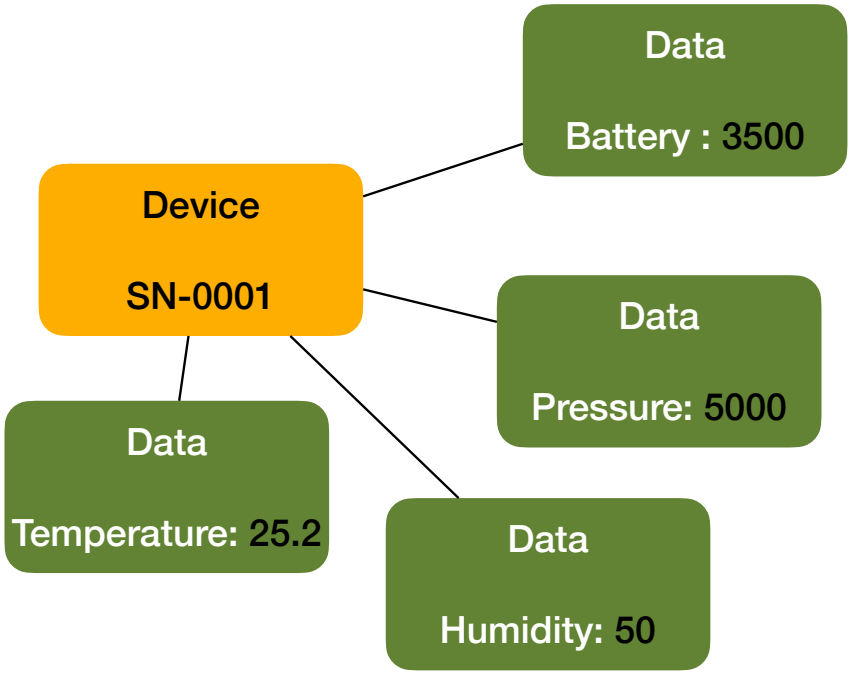
Topic?

Payload?



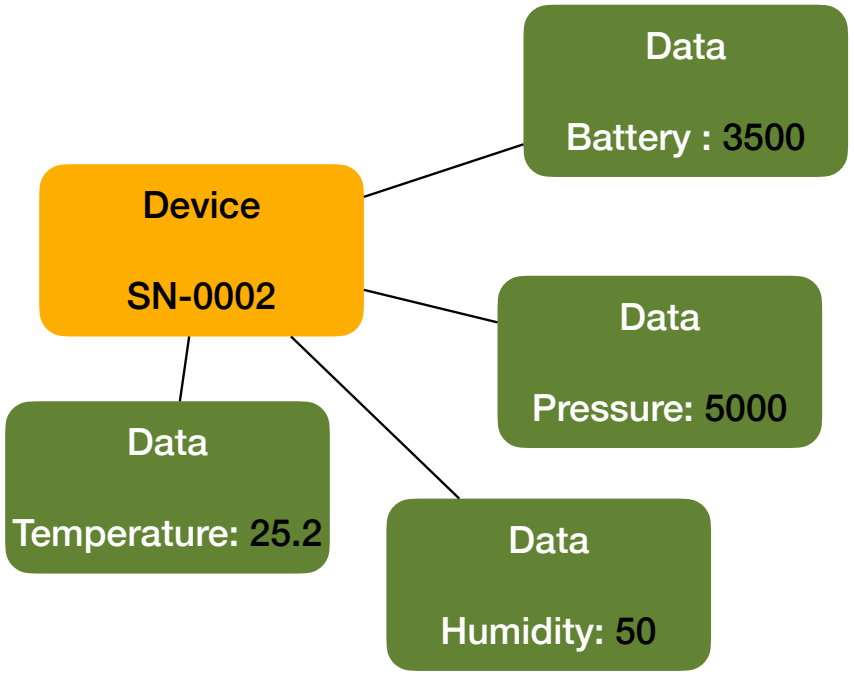
MQTT design pattern

2-device



Topic?

Payload?

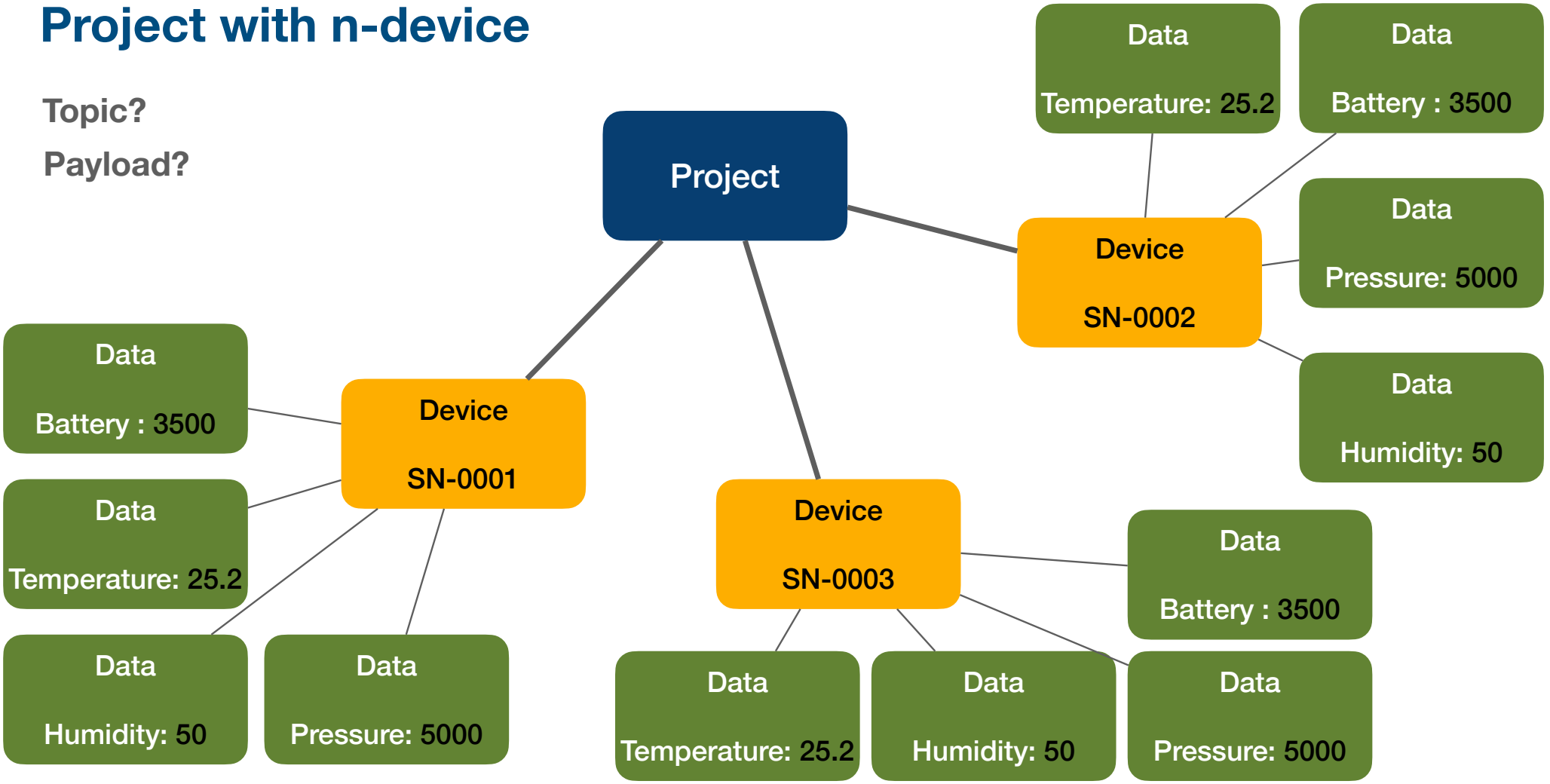


MQTT design pattern

Project with n-device

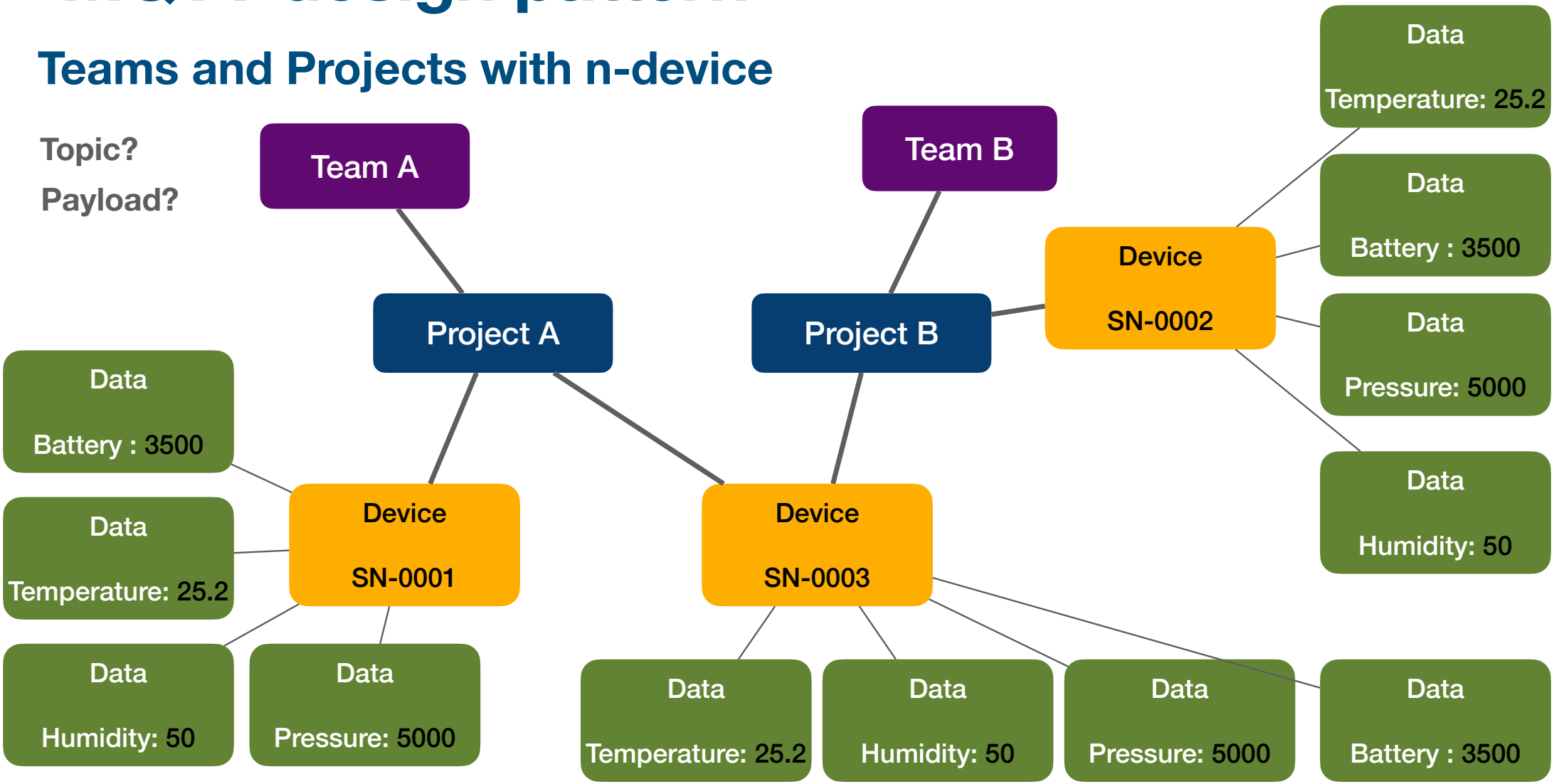
Topic?

Payload?

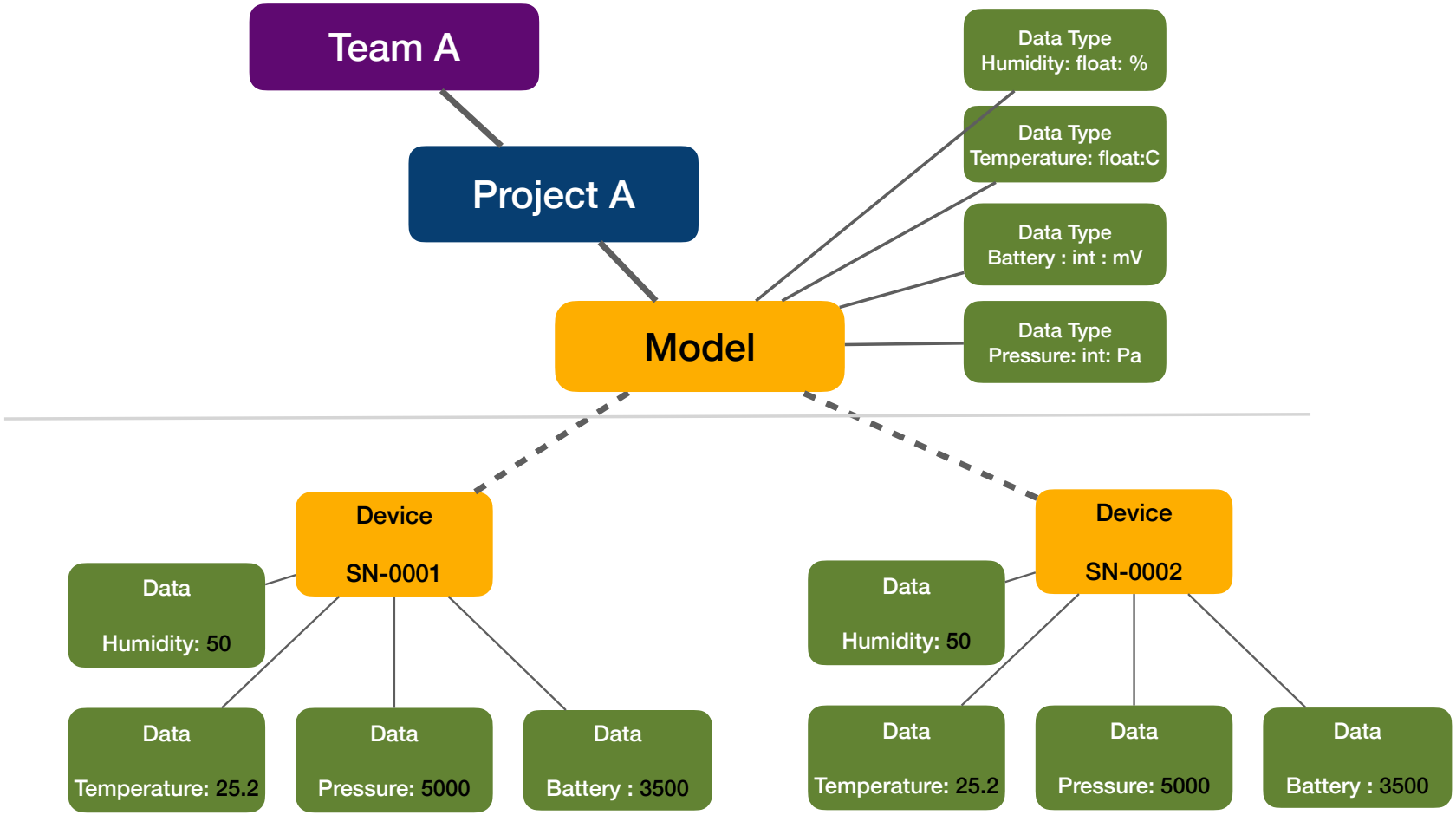


MQTT design pattern

Teams and Projects with n-device

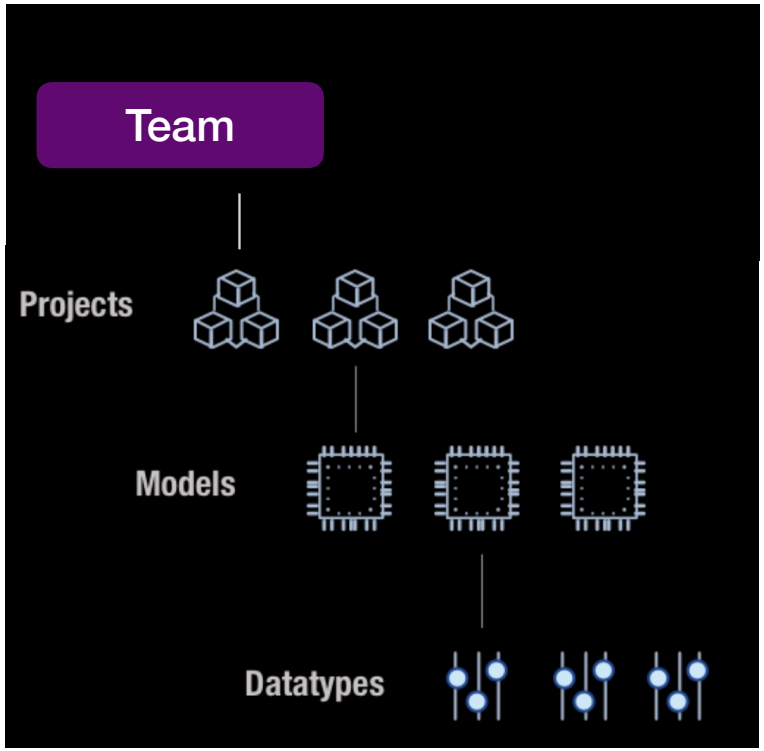


MQTT design pattern concept



Taxonomy

- Each **Team** can have one or more **Projects.**
- A **Project** allows you to define **one** or **more Models.**
- **Each** Model describes a single type of device.
- A model can have one or more **Datatypes.** A Datatype is a named value for each piece of information you want to exchange with the device.



Design Example

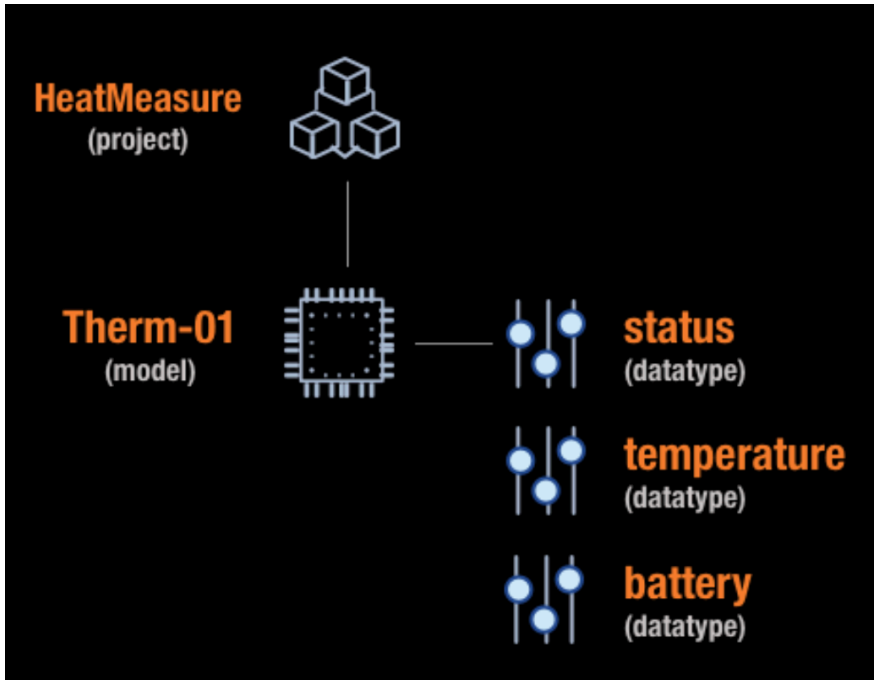
Define Project then Model

- a Project to build a variety of hand-held **thermometers**.
  For the first one, you define a **Model** and give it a name.
  That model will be exchanging three **attributes**

Design Example

Define Project then Model

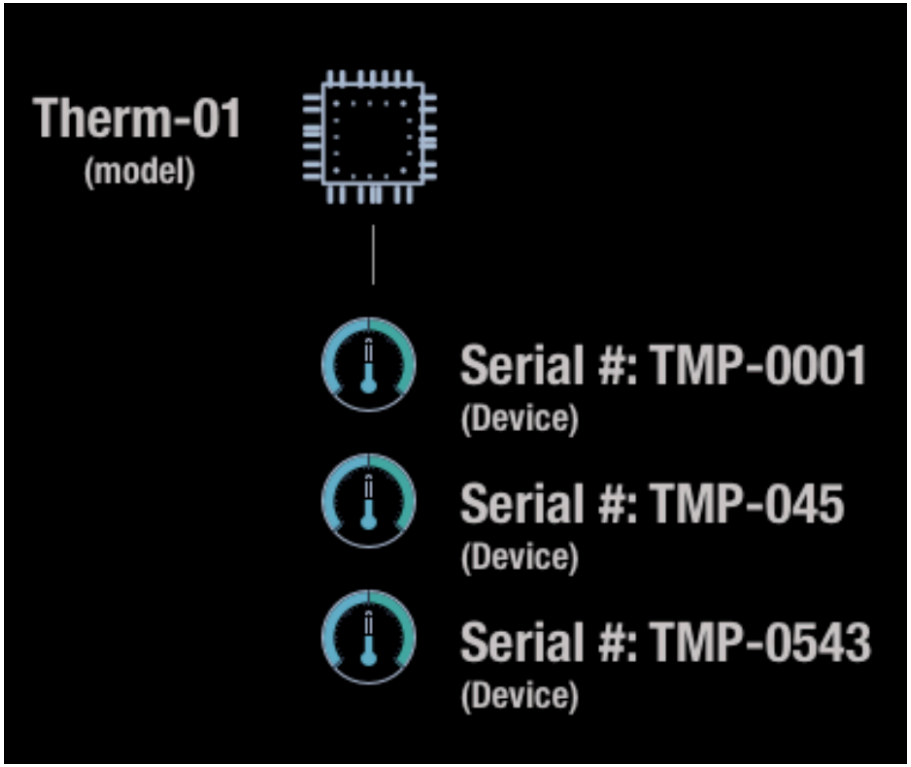
- A project is named **HeatMeasure**, and a model in that project is called **Therm-01.** The device wants to exchange **three values** with the cloud: **status**, **temperature**, and **battery.** In the future, you can have a **Therm-02**, **Therm-03**



Design Example

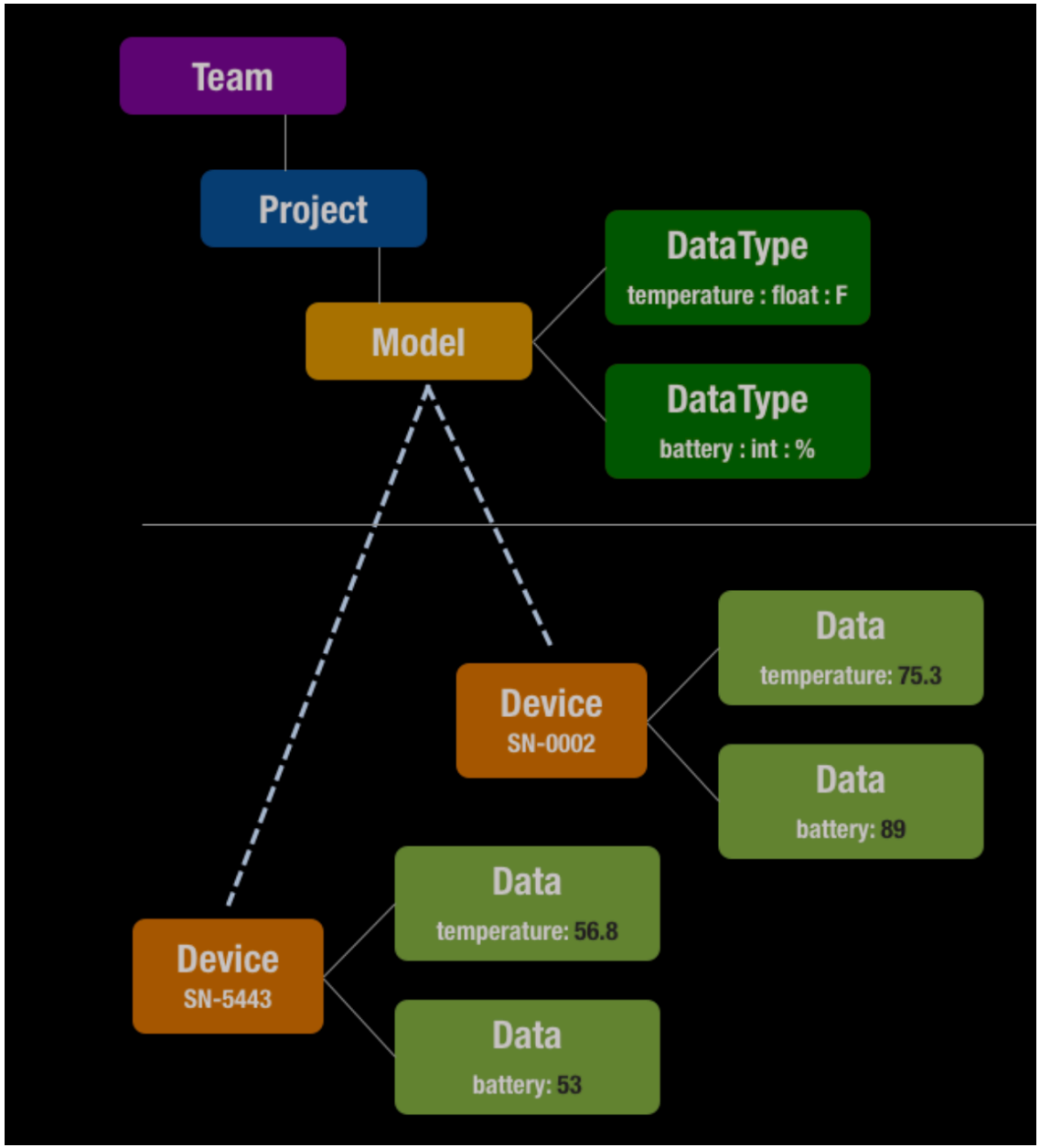
An instance of a model is called a “Device”

- Once you've defined your **Model** and **Datatype**, you can now virtually stamp out one or more instances. An **instance** of a **Model** is called a **Device.** Each Device needs to have its own **unique serial** number so you can tell multiple Devices apart



Design Example

Putting it all together.



IoT and MQTT

Typical topic and payload

- Topic: v1/app/set/project-01/model-ABC/SN-0001

- Payload: {
  “action” : “set”,
  “project” : “project-01”,
  “model” : “model-ABC”,
  “serial” : “SN-001”,
  “name” : “temperature”,
  “value” : “78.2”
  }

IoT and MQTT

If data has geographic/GPS data associated

- Topic: v1/app/set/project-01/model-ABC/SN-0001

- Payload: {
  “action” : “set”,
  “project” : “project-01”,
  “model” : “model-ABC”,
  “serial” : “SN-001”,
  “name” : “temperature”,
  “value” : “78.2”,
  “geo_lat” : “22.1234”
  “geo_lng”: “-112.221”
  }

Topic hierarchy

v1/app/set/project-01/model-ABC/SN-1234

- **v1**: This is a prefix with the protocol version embedded in it. This means your application topics should never interfere with IoT data. It also allows you to filter messages sent by IoT from other types of applications.
- **app**: separates data sent to/from a device into serval stream
  - /**hw**: hardware **level operations**, like **voltages**, calibrations, **RAM** level
  - /**sys**: **operating system** or **firmware** level operation, like **versions**, manufacturer IDs
  - /**app**: application level values, e.g. whether and **application** has run, what the **sensor values** are, what button has **been pressed**

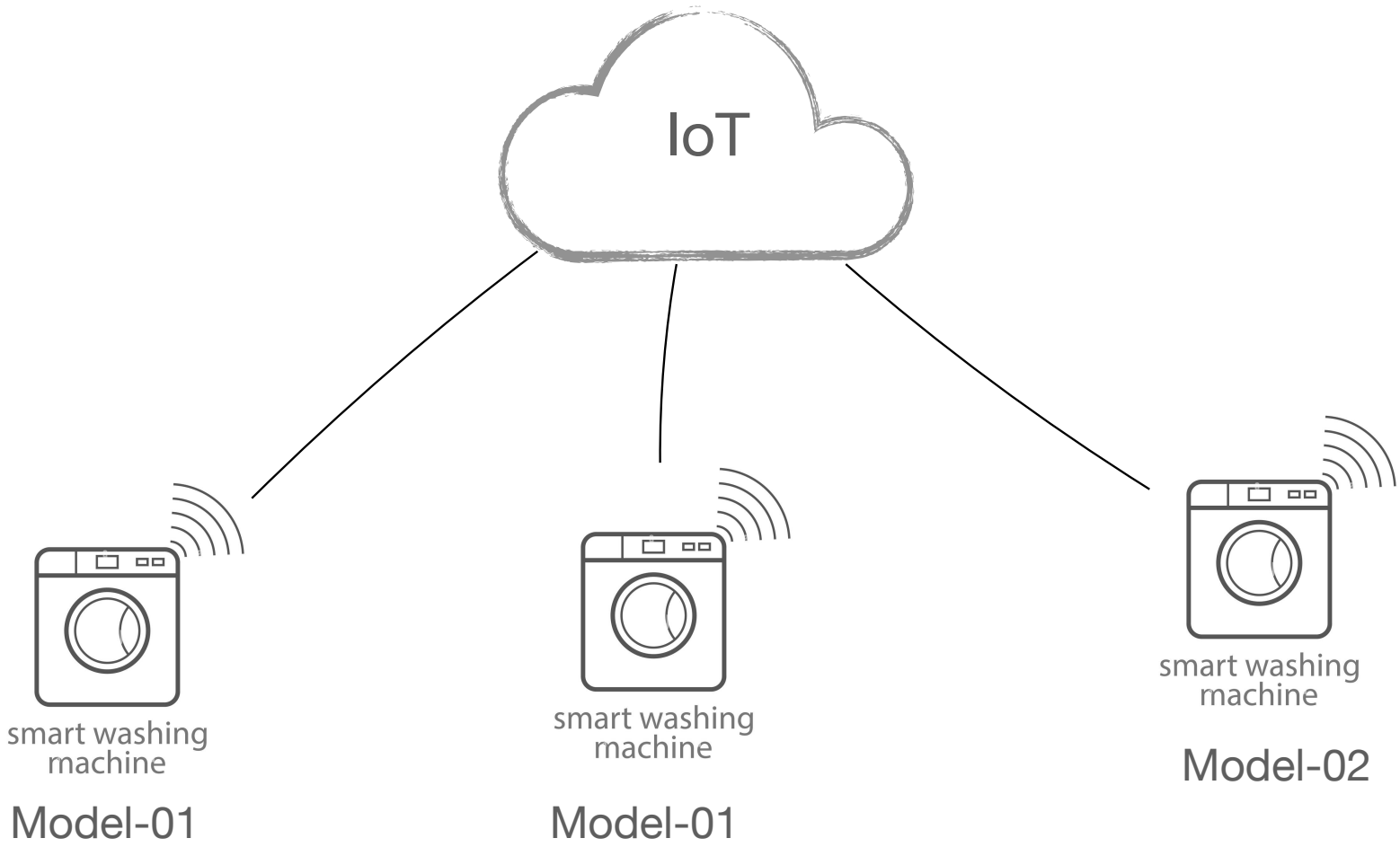
Topic hierarchy

v1/app/set/project-01/model-ABC/SN-1234

- **action**: An operation to be performed on individual data elements**.** The most common are **set**, **get**, and **monitor.**
- **project name**: Name of the Iot Project
- **model name**: Model name
- **serial number**: Device serial number of this specific data
- **name**: Name of the Datatype for which we send a value
- **Vvalue:** Value of the Data element for this specific device associated with that name

Exercise

Smart Washing Machine



Exercise design topic and payload

Smart Washing Machine

- Project Name ?
- Model Name ?
- Device Serial Number ?
- Sensors ?
  - status : string : “on”, “off”, “restart”, “wash”, “spin”, “drained”
  - temperature : float: C
  - wash_count : int: 1-200
  - ????

Exercise design topic and payload

Smart Washing Machine - for this operation

- Get hardware level operations e.g. wash_count
- Get firmware version, manufacturer id
- Set geo-location or location placement
- Monitor value of all sensors
- Set status to “maint” to indicate this machine need to be maintenance.