## What to do? (1)

- Create a flow to periodically publish MQTT messages to the local mosquitto broker **(localhost, port 1884)**, to the topic ***challenge3/id_generator***
  (be sure to start the mosquitto broker locally with the correct port)

- Messages should be sent with a rate of **1 message every 5 seconds.**
- Each message should contain in the payload a string of JSON format with a **random** number (**id**) between **0 and 50000**, and the time in which the msg is generated (**UNIX timestamp**)

**Message payload example:** {"id": 7781, "timestamp":1710930219}

When sending the message, also save it in a CSV (***id_log.csv***) with the form:

No.,ID,TIMESTAMP where No. is the row number (incremental)

**Include this CSV in your delivery**

## What to do? (2)

In another branch of the flow (same flow):

- Subscribe to the topic ***challenge3/id_generator*** in the local broker **(localhost, port 1884)**

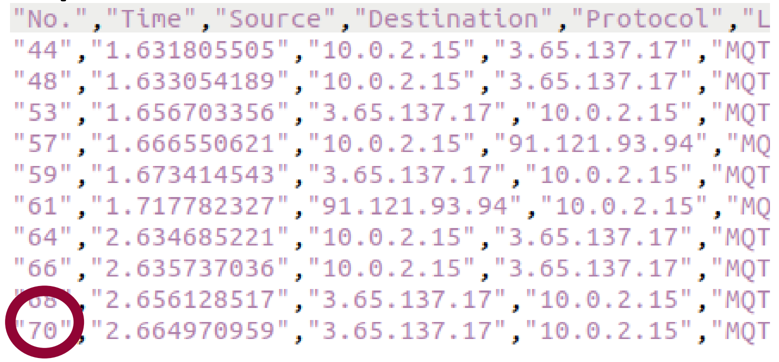
- After receiving a message from the subscription, take the ID and compute the **remainder of the division by 7711** to get **N**:

  **N = ID modulo 7711**

- At every message you receive, process the **challenge3.csv** file and take the message with frame number equal to the received identifier **N**

e.g. id=7781

**N** = 7781 modulo 7711 **= 70** →



```
"No.","Time","Source","Destination","Protocol","L
"44","1.631805505","10.0.2.15","3.65.137.17","MQT
"48","1.633054189","10.0.2.15","3.65.137.17","MQT
"53","1.656703356","3.65.137.17","10.0.2.15","MQT
"57","1.666550621","10.0.2.15","91.121.93.94","MQ
"59","1.673414543","3.65.137.17","10.0.2.15","MQT
"61","1.717782327","91.121.93.94","10.0.2.15","MQ
"64","2.634685221","10.0.2.15","3.65.137.17","MQT
"66","2.635737036","10.0.2.15","3.65.137.17","MQT
"68","2.656128517","3.65.137.17","10.0.2.15","MQT
"70","2.664970959","3.65.137.17","10.0.2.15","MQT
```

## What to do? (3)

- If the message with **Frame No. = N** in the file contains an **MQTT Publish** then, send a publish message to the local broker to the same topic found in the MQTT Publish The message you publish should have as payload the following string: {"timestamp":"CURRENT_TIMESTAMP","id":"SUB_ID","payload":"MQTT_PUBLISH_PAYLOAD"}

  Where:
  - CURRENT_TIMESTAMP **Current time** at the moment of the sending of the pub
  - SUB_ID Message **ID** received from the Subscription i.e. **7781**
  - MQTT_PUBLISH_PAYLOAD Payload from the CSV of the Publish message with frame number **N**

- Limit the msg published in this step with a rate of **four messages per minute***

*Use the rate limiter node

## What to do? (4)

- In addition, after publishing the publish message, if the Publish Message contains **in the payload a temperature in Fahrenheit** (check for the Type=Temperature and Unit=F attributes in the payload), take this message and plot its value in a Node-Red chart:

- For the Chart:
  - Take only publish messages having payload with temperature in Fahrenheit
  - Produce a **chart** in Node-Red plotting the temperature value**, taking the mean value in the "range" attribute as a number (min + max divided by two)**

When plotting, save the payload of these msgs (only those with Temp in Fahrenheit) in a CSV *(**filtered_pubs.csv**)* containing one msg Payload for each row:

**filtered_publish.csv format:**

*No.,LONG, RANGE, LAT, TYPE, UNIT, DESCRIPTION* where No. is row number (incremental)

**Include this CSV in your delivery**

## Publish example (1)

**No.,Time,Source,Destination,Protocol,Length,Source Port,Destination Port,Info,Payload**

**36,5.70180035,3.65.137.17,10.0.2.15,MQTT,322,1883,34039**,"Publish Message [hospital/facility1], Publish Message [hospital/room1]","{""long"": 80, ""range"": [0, 59], ""lat"": 86, ""type"": ""temperature"", ""unit"": ""C"", ""description"": ""Room Temperature""},{""long"": 92, ""range"": [8, 37], ""lat"": 80, ""type"": ""temperature"", ""unit"": ""F"", ""description"": ""Room Temperature""}"

**PUBLISH WARNING:**

**If packet contains multiple Publish, send them as separate publish messages (and plot in the chart separately if match the filtering)**
**If some of the payload do not appear, consider it as empty payload**

# Publish example (2)

**No.,Time,Source,Destination,Protocol,Length,Source Port,Destination Port,Info,Payload**

**36,5.70180035,3.65.137.17,10.0.2.15,MQTT,322,1883,34039**,"Publish Message [hospital/facility1], Publish Message [hospital/room1]","{""long"": 80, ""range"": [0, 59], ""lat"": 86, ""type"": ""temperature"", ""unit"": ""C"", ""description"": ""Room Temperature""},{""long"": 92, ""range"": [8, 37], ""lat"": 80, ""type"": ""temperature"", ""unit"": ""F"", ""description"": ""Room Temperature""}"

Parse the Payload in a JSON **when possible**.

We know, you have to deal with ""

Publish Message to topic: hospital/facility1

```
{
  "timestamp": "1712561821",
  "id": "7747",
  "payload":{"long":80,"range":[0,59],"lat":86,"type":"temperature","unit":"C","description":"Room Temperature"}
}
```

Publish Message to topic: hospital/room1

```
{
  "timestamp": "1712561821",
  "id": "7747",
  "payload": {"long": 92,"range": [8,37],"lat": 80,"type": "temperature","unit": "F","description": "Room Temperature"}
}
```

## What to do? (5)

- If the message with **Frame No. = N** instead contains an MQTT ACK message **(Publish Ack, Connect Ack, Sub/Unsub Ack)**, increment a global ACK counter, then save the message into a CSV file named "**ack_log.csv**" with the form:

  **TIMESTAMP, SUB_ID, MSG_TYPE**

  Where:
  - TIMESTAMP is the current time when the msg is saved in the CSV
  - The SUB_ID is the Message **id** received from the Subscription i.e. **7781**
  - MSG_TYPE is the message type: **e.g Connect Ack**

  **Include this CSV in your delivery**

- After you find an ACK and you save it in the CSV: **SEND** the value of the global ACK counter to **your thingspeak channel**, passing in the **field1** of the channel the value of the global ACK counter. **SEND USING HTTP API**

  **Include the channel link in the report and make it public!!**

## What to do? (6)

- In all the other cases (frame No = **N** not containing an ACK or a publish) → Ignore the message!

Program your flow to stop working after **receiving exactly 80 id** messages from the subscription:

**do not process more than ID 80 messages**

(discarded msgs still counted in the 80 messages limit)