INTERNET OF THINGS FOR EUROPEAN SMALL AND MEDIUM ENTERPRISES

DEMONSTRATOR:

Home I/O to ThingSpeak

Developer institution:

Université Nice Sophia Antipolis

Membre de UNIVERSITÉ CÔTE D'AZUR

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+

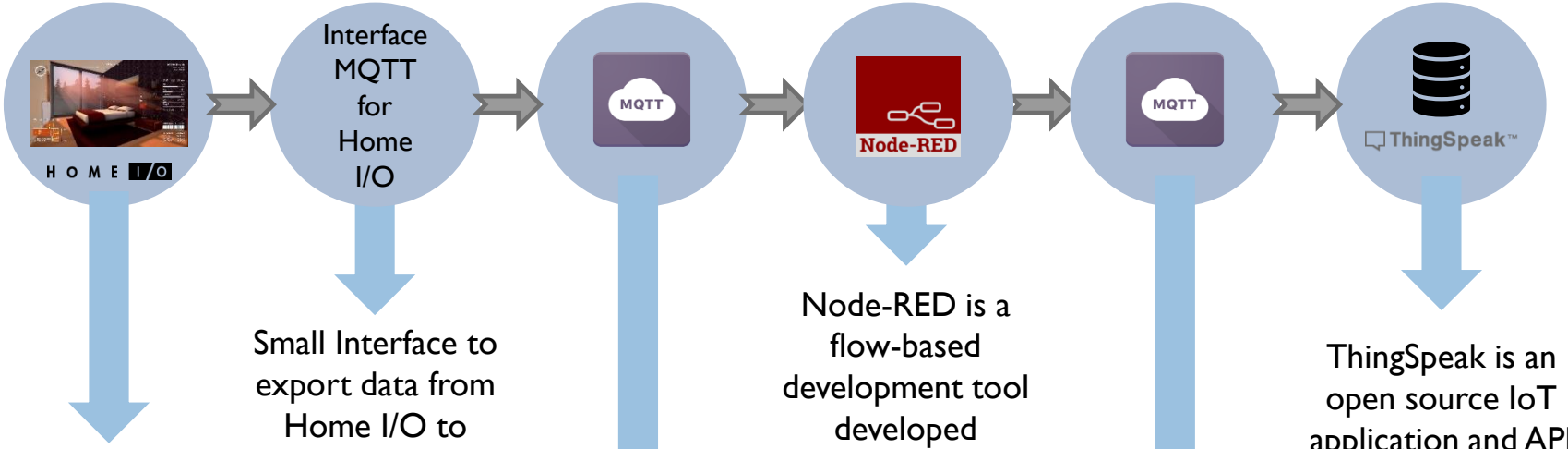
CONTEXTUALIZATION

- This demonstrator shows the processing of the IOT chain from the sensors (simulated by a virtual house environment named Home I/O) to the data processing (ThingSpeak).

- The routing of the data will be via the MQTT protocol

- The IoT workflow orchestration of the application will be made graphically within a Red-Node container

CONTEXTUALIZATION



HOME I/O has everything you need to create and monitor a real-time smart home simulation. You can monitor, control and secure a virtual home by creating smart home scenarios

Small Interface to export data from Home I/O to MQTT

MQTT is an ISO standard publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol

Node-RED is a flow-based development tool developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things

MQTT is an ISO standard publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol

ThingSpeak is an open source IoT application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network.

ORIENTATION

Follow these instructions to use the demonstrator:

1. Download & Install Home I/O

- We will use Home I/O to simulate a home equipped with many connected sensors and actuators. Through this software, it will be possible to have access to simulated data (temperature, user's presence, …) but also to act on the environment (turn on a light, control the heating, close the blinds, …).

- To install the simulation environment, you can retrieve the software from one of the following addresse :
  - https://teachathomeio.com/telecharger-demo/

ORIENTATION

2. Download & Install Node-Js

- You can go to the following page to download it for your system: https://nodejs.org/en/download/

3. Download & Install Node-Red

- Now that you have the sensors and actuators of the simulator, you will be interested in the Node-RED environment that will allow us to exploit this information.
- Node-RED is based on Node. Once you have the Node.js environment on your machine, you can install Node-RED simply as an additional pack-age. You will follow the documentation available at the following address: https://nodered.org/docs/getting-started/installation

ORIENTATION

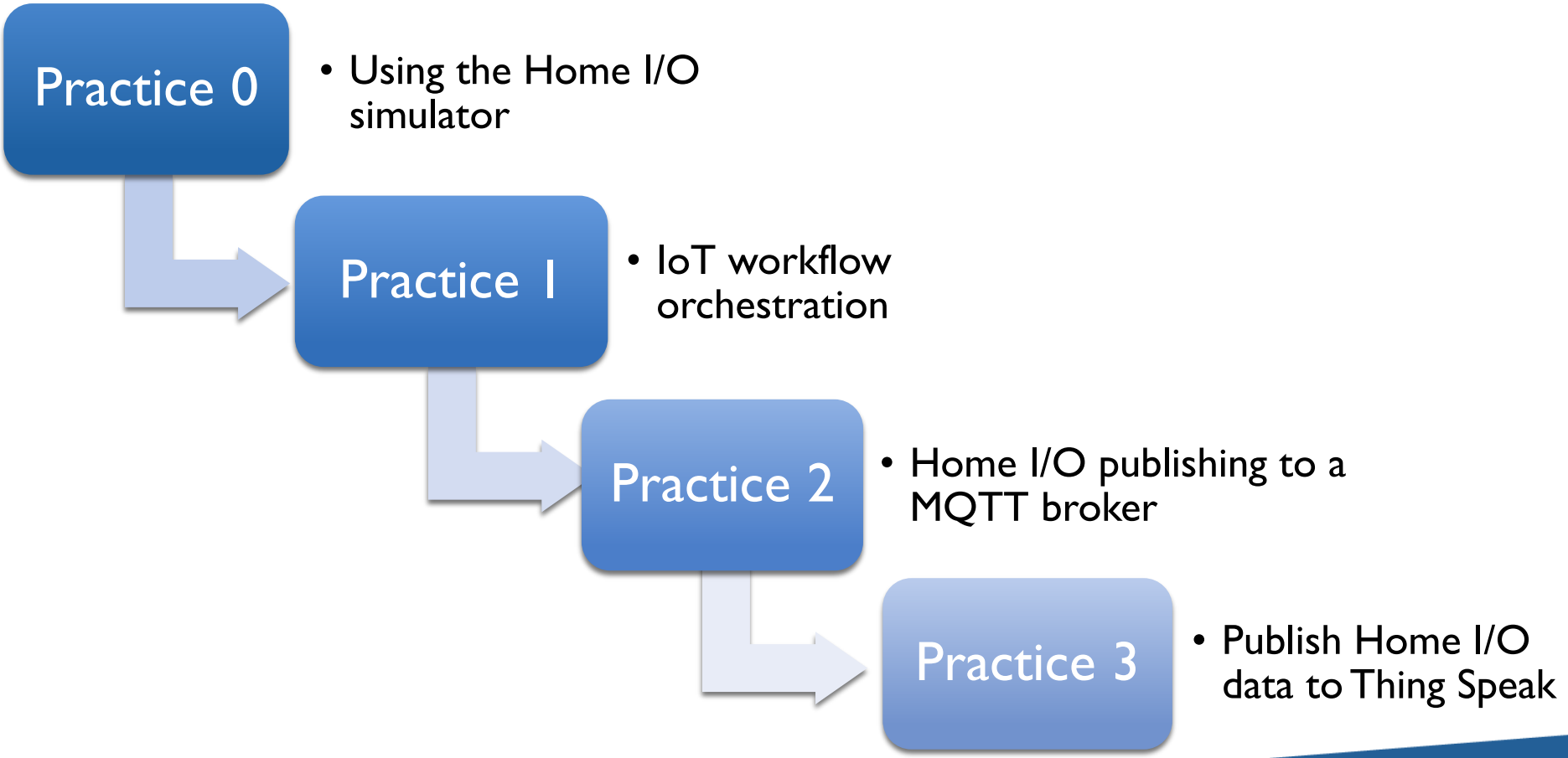
4. Download & Install Broker MQTT in Node-RED environment

- MQTT is a messaging protocol used in the Internet of Things. This one is based on a Publisher-Broker-Subscriber template. A Publisher sends the information to a Broker. A Subscriber will subscribe to a topic from a Broker.
- To make it easier to implement, you can start a Broker in Node-RED. But for that you must install an additional package. You can do this directly from the command line (but you will have to restart your Node-RED instance)
  - `npm install node-red-contrib-mqtt-broker`
- or from the Node-RED interface, go to the right menu and then "Manage Palette / Install". You will then type:
  - `node-red-contrib-mqtt-broker`
- By instantiating a node "mosca", for clicking on "Deploy" you will launch a MQTT Broker which will be available at the address 127.0.0.1:1883.

ORIENTATION

5. Download HomeIO_MQTT.exe program
   - http://trolen.polytech.unice.fr/cours/projet-si3/HomeIO_MQTT.zip
6. Create a Thing Speak account
   - The default ThingSpeak service limit you to publish 1 data every 15 seconds. So, we installed a ThingSpeak server without that limitation.
     - Navigate to http://sparks-vm24.i3s.unice.fr:3000/users/sign_up
     - Create an account
     - Once logged in, navigate to http://sparks-vm24.i3s.unice.fr:3000/account
     - Write down the API key seen on this page

EXPERIMENTATION



Erasmus+ Strategic Partnership n° 2016-1-IT01-KA202-005561
IoT4SMES - Internet of Things for European Small and Medium Enterprises

PRACTICE 0. USING THE HOME I/O SIMULATOR

- **Aim**: Take control of the environment and configure it to export its data

1. First, please configure the Home I/O software to use English. You can configure this by selecting "Options / Localization / Language = English". This will avoid problems with devices names using special characters and bad interaction with other programs.
2. To get a handle on the simulation environment, we invite you to consult the documentation: https://realgames.co/docs/homeio/en/
3. We invite you to consult:
   - The sensors and actuators map: https://realgames.co/docs/homeio/en/devices-map/
   - The device control mode: https://realgames.co/docs/homeio/en/device-modes/

PRACTICE 0. USING THE HOME I/O SIMULATOR

4. Regarding the control mode of the devices, they must be set in the "External" mode to be controlled by a third-party application (and not by clicking in the simulator on all devices).
5. You can click on all the small symbols to change them to blue (external mode) ... but it may take you a half day and you may forget one. The easiest way is to save the simulation:

Red (wired) item: incorrect | Blue (external) item: correct

- Go to folder "Documents / Home IO / Saves"
- Edit the xml file containing the sensor and actuator configuration with your favourite text editor
- Replace the word "Wired" with the word "External".
- Save the document
- Load the simulation in Home I/O and check if devices turned “blue”.

PRACTICE 1. IOT WORKFLOW ORCHESTRATION

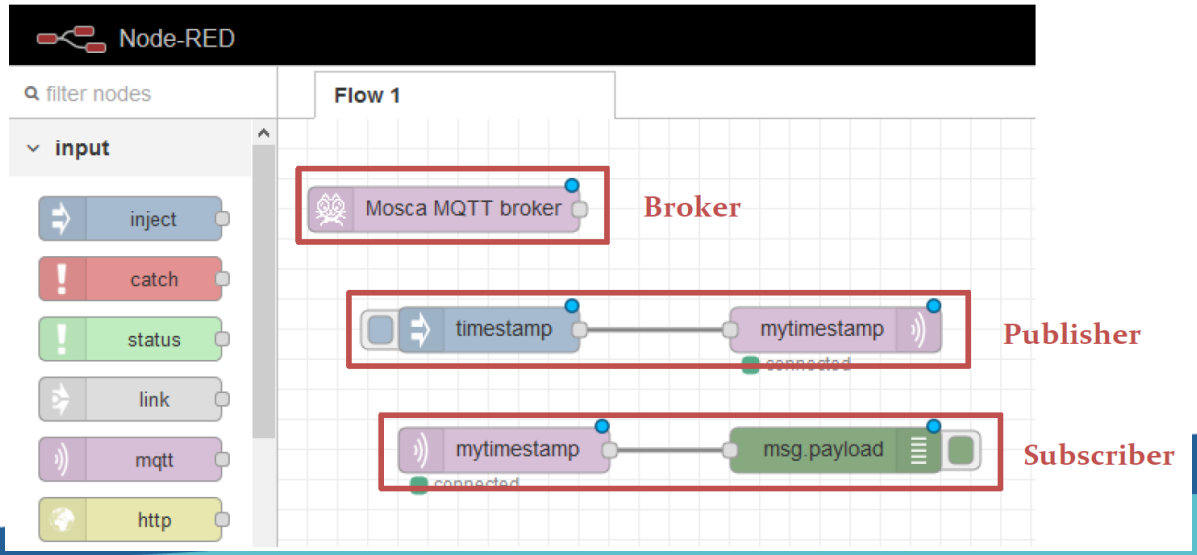
- **Aim**: Check the installation of Node-RED and test the MQTT protocol

1. Test your Node-RED installation
   - You will then open a command interpreter (PowerShell if you are on Windows) and run the node-red command. The environmental control interface is then accessible on your machine at the following address: http://127.0.0.1:1880
2. MQTT Publisher inside Node-RED
   - Posting information about an MQTT topic is trivial. It's all about naming the topic (naming based on the notion of path as to find a file).
   - You will start by setting an "inject" node (which will send a default timestamp) in the INPUT category and a "mqtt" node in the OUTPUT category. You will link the two nodes in question so that the information provided by the timestamp is published on a topic named "mytimestamp". The configuration of a node is done by double clicking on it. You have just set up an MQTT Publisher.

PRACTICE 1. IOT WORKFLOW ORCHESTRATION

3. MQTT Subscriber inside Node-RED

- To subscribe to an MQTT topic, simply add a node "mqtt" type INPUT and a node "debug" (to see that the infor-mation has been sent (you can view it in the debug console right After connecting the two nodes and configuring the topic of the node “mqtt” to the value "mytimestamp", each time you click on the injector, you will receive the information via MQTT.You should have the following schema.



PRACTICE 1. IOT WORKFLOW ORCHESTRATION

4. MQTT Subscriber inside Node-RED

- To convince yourself that these communications go through the network, have fun running only the Broker on one computer, have a Publisher on another machine, and the Subscriber on a third.
- You can also instantiate the previous sample with the following code. To create such a flow, you must copy/paste the following code after selecting top-right menu "Import / Clipboard":


```
[{"id":"45cc7449.5a195c","type":"tab","label":"Flow 3"},{"id":"19a467ac.a8eba8","type":"mosca
in","z":"45cc7449.5a195c","mqtt_port":1883,"mqtt_ws_port":8080,"name":"","username":"","password":"","dburl":"","x"
:149,"y":67,"wires":[[]]},{"id":"811eb811.d17888","type":"mqtt
in","z":"45cc7449.5a195c","name":"","topic":"test","qos":"2","broker":"aa872460.7dff18","x":109,"y":148,"wires":[["
352cbba6.641a64"]]},{"id":"d0ffe662.31c688","type":"mqtt
out","z":"45cc7449.5a195c","name":"","topic":"test","qos":"","retain":"","broker":"aa872460.7dff18","x":311,"y":225
,"wires":[]},{"id":"d4e5a13a.ad9a2","type":"inject","z":"45cc7449.5a195c","name":"","topic":"","payload":"","payloa
dType":"date","repeat":"","crontab":"","once":false,"x":134,"y":225,"wires":[["d0ffe662.31c688"]]},{"id":"352cbba6.
641a64","type":"debug","z":"45cc7449.5a195c","name":"","active":true,"console":"false","complete":"false","x":295,"
y":149,"wires":[]},{"id":"aa872460.7dff18","type":"mqtt-
broker","z":"","broker":"localhost","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"60","
cleansession":true,"willTopic":"","willQos":"0","willPayload":"","birthTopic":"","birthQos":"0","birthPayload":""}]
```


- Once the flow is deployed and both MQTT nodes are connected, injecting the timestamp on the blue node will result in a debug message in the debug tab. If the flow is confirmed working, you can continue this demonstrator.

PRACTICE 2. HOME I/O PUBLISHING TO A MQTT BROKER

- **Aim**: publish the Home I/O data to the local broken and try to visualize the data reception

1. Home I/O to MQTT protocol: use a bridge
   - When Home I/O is running, start the HomeIO_MQTT.exe program. This program will publish the home I/O data to a broker. The default address of the broker used is localhost:1883. You can change the address of the broker by specifying it as a command line option.

PRACTICE 2. HOME I/O PUBLISHING TO A MQTT BROKER

- If the tool detects Home I/O properly it will dump the list name of data sources from the simulator into a CSV formatted list (file sensors.csv in the program directory) with the following format: `MemoryType,Address,DSName,Room,DataType,Topic`
  - MemoryType: type of Home I/O memory
    - Input: from sim to broker, event-driven. e.g. switches or sensors.
    - Output: from broker to sim. e.g. lights or heater.
    - Memory: from sim to broker, continuous update rate. e.g. temperatures.
  - Address: Home I/O address
  - DSName: Description of the data source
  - Room: Room the data source is attached to
  - DataType: value type
    - Topic: MQTT topic used to communicate with the data source
    - Input/Memory: bridge publishes to the topic with the updated data
    - Output: bridge subscribes to the topic and updates the sim with data published to the topic

PRACTICE 2. HOME I/O PUBLISHING TO A MQTT BROKER

2. Subscribe to the published Home I/O values

- If the tool successfully connected to the broker it will display a message "Connected to broker at localhost" Then "Connected".
- To test the data publication to the broker, we will do a subscribe with a node in Node-RED.
- First, get the name of a topic for the Room A Temperature ("/home/general/Memory/float/Tempera-ture_Zone_A_[K]"). Be careful, the topic name depends on your computer language. So, it could be different if you use a French, an Italian, a Spanish or a German one. If you want to use the English names, set the appropriate locale in Home I/O with the menu "Options / Localization / Language".

PRACTICE 2. HOME I/O PUBLISHING TO A MQTT BROKER

- You can also create the flow with the following code. We added a node for limiting the subscribing to one message every 10 seconds to avoid collecting to much data.


```
[{"id":"907d1380.2af72","type":"mosca
in","z":"f59c6764.97c658","mqtt_port":1883,"mqtt_ws_port":8080,"name":"","username":"",
"password":"","dburl":"","x":150,"y":40,"wires":[[]]},{"id":"c92ffef8.6cbec","type":"de
bug","z":"f59c6764.97c658","name":"","active":true,"tosidebar":true,"console":false,"to
status":false,"complete":"false","x":690,"y":100,"wires":[]},{"id":"cc4d3412.5f1318","t
ype":"mqtt
in","z":"f59c6764.97c658","name":"","topic":"/home/general/Memory/float/Temperature_Zon
e_A_[K]","qos":"2","broker":"62b4dd3e.6162cc","x":260,"y":100,"wires":[["a37cd033.3776b
8"]]},{"id":"a37cd033.3776b8","type":"delay","z":"f59c6764.97c658","name":"","pauseType
":"rate","timeout":"5","timeoutUnits":"seconds","rate":"1","nbRateUnits":"10","rateUnit
s":"second","randomFirst":"1","randomLast":"5","randomUnits":"seconds","drop":true,"x":
480,"y":180,"wires":[["c92ffef8.6cbec"]]},{"id":"62b4dd3e.6162cc","type":"mqtt-
broker","z":"","name":"","broker":"127.0.0.1","port":"1883","clientid":"","usetls":fals
e,"compatmode":true,"keepalive":"60","cleansession":true,"willTopic":"","willQos":"0","
willPayload":"","birthTopic":"","birthQos":"0","birthPayload":""}]
```

PRACTICE 3. PUBLISH HOME I/O DATA TO THING SPEAK

- **Aim**: Connect our data with Thingspeak through Node-RED

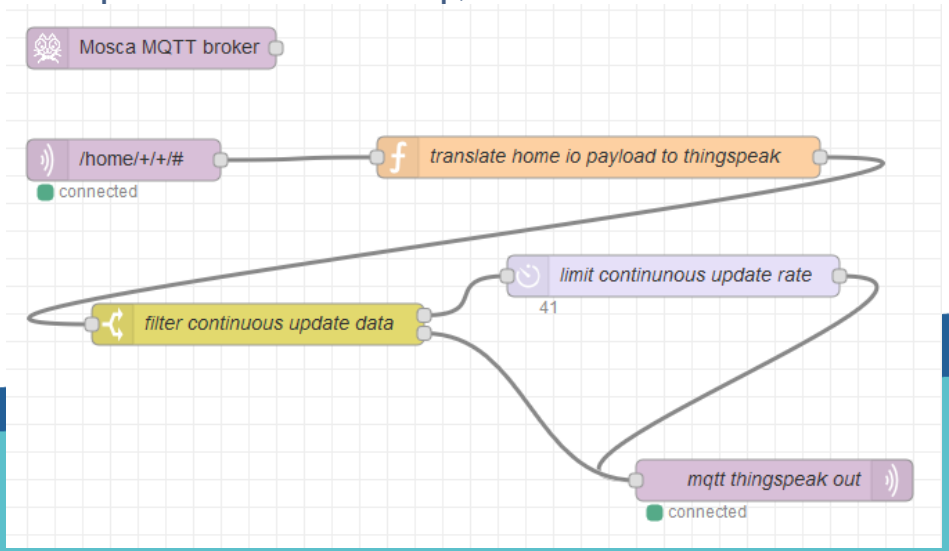
1. Node-RED flow linking Home I/O and ThingSpeak

   - To connect the Home I/O sensors to ThingSpeak, import the following flow:


```
[{"id":"45cc7449.5a195c","type":"tab","label":"Flow 3"},{"id":"19a467ac.a8eba8","type":"mosca
in","z":"45cc7449.5a195c","mqtt_port":1883,"mqtt_ws_port":8080,"name":"","username":"","password":"","dburl":"","x":149,"y":67,"wire
s":[[]]},{"id":"811eb811.d17888","type":"mqtt
in","z":"45cc7449.5a195c","name":"","topic":"/home/+/+/#","qos":"2","broker":"aa872460.7dff18","x":129,"y":148,"wires":[["f5868538.c
4e148"]]},{"id":"d0ffe662.31c688","type":"mqtt out","z":"45cc7449.5a195c","name":"mqtt thingspeak
out","topic":"","qos":"","retain":"","broker":"18d4ce79.021f92","x":599,"y":380,"wires":[]},{"id":"f5868538.c4e148","type":"function
","z":"45cc7449.5a195c","name":"translate home io payload to thingspeak","func":"var API_KEY = \"xxxx\";\n\nvar topicSplit =
msg.topic.split(/[/]+/);\n\n// Flag continuous messages\nmsg.shouldBeThrottled = false;\nif(\n topicSplit[3] == \"Memory\" ||\n
topicSplit[5].startsWith(\"Thermostat\") ||\n topicSplit[5].startsWith(\"Date_and_Time\") ||\n
topicSplit[5].startsWith(\"Brightness\")){\n msg.shouldBeThrottled = true;\n}\n\n// Build ThingSpeak payload\nvar value =
msg.payload;\nif(topicSplit[3] == \"bool\"){\n value = msg.payload==\"True\"?1:0;\n}\nmsg.payload = Date.now() + \",\" +
topicSplit[topicSplit.length - 1] + \",\" + value + \",\" + API_KEY;\nmsg.topic = \"devices\" + msg.topic;\n\n// Don't proceed if
the API key isn't set properly\nif(API_KEY == \"xxxx\") return null;\n\nreturn
msg;","outputs":1,"noerr":0,"x":472,"y":146,"wires":[["3d1be41d.edff7c"]]},{"id":"c71aa5da.5f9328","type":"delay","z":"45cc7449.5a19
5c","name":"limit continunous update
rate","pauseType":"timed","timeout":"5","timeoutUnits":"seconds","rate":"1","nbRateUnits":"30","rateUnits":"second","randomFirst":"1
","randomLast":"5","randomUnits":"seconds","drop":true,"x":526,"y":232,"wires":[["d0ffe662.31c688"]]},{"id":"3d1be41d.edff7c","type"
:"switch","z":"45cc7449.5a195c","name":"filter continuous update
data","property":"shouldBeThrottled","propertyType":"msg","rules":[{"t":"eq","v":"true","vt":"jsonata"},{"t":"eq","v":"false","vt":"
jsonata"}],"checkall":"true","outputs":2,"x":226,"y":267,"wires":[["c71aa5da.5f9328"],["d0ffe662.31c688"]]},{"id":"aa872460.7dff18",
"type":"mqtt-
broker","z":"","broker":"localhost","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"60","cleansession":tru
e,"willTopic":"","willQos":"0","willPayload":"","birthTopic":"","birthQos":"0","birthPayload":""},{"id":"18d4ce79.021f92","type":"mq
tt-broker","z":"","broker":"sparks-
vm24.i3s.unice.fr","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"60","cleansession":true,"willTopic":"",
"willQos":"0","willPayload":"","birthTopic":"","birthQos":"0","birthPayload":""}]
```

PRACTICE 3. PUBLISH HOME I/O DATA TO THING SPEAK

- Things to note about this new flow:
  - It listens to all data sources (for this, it uses regular expression in the MQTT topic)
  - Update the API_KEY variable in the flow to the API key retrieved when creating your ThingSpeak account
  - Filters continuous update data (simulation date/time, thermostats, brightness, memory typed inputs) to reduce sample rate to 1 data point per 30 seconds
  - The function node in the middle performs the translation of Home IO MQTT data to MQTT to ThingSpeak bridge data
    - For reference, the format of the output data is: TimeStamp,ItemName Value,API_KEY
  - Deploy the flow
- If the MQTT input and outputs show as connected the flow is up and running, data is being sent to ThingSpeak.

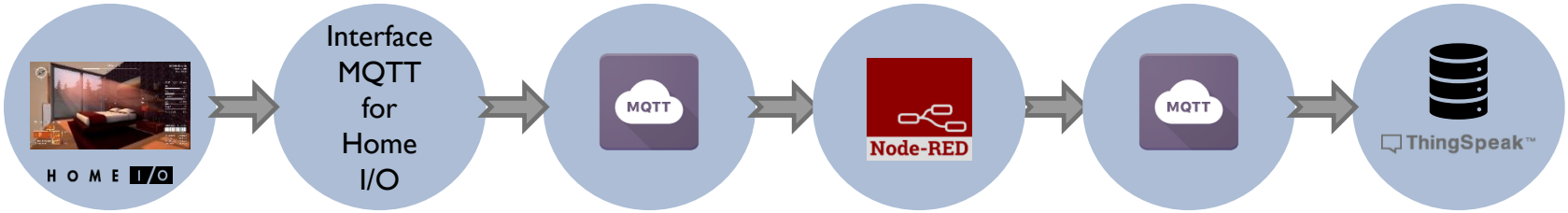


PRACTICE 3. PUBLISH HOME I/O DATA TO THING SPEAK

2. Data visualization
   - Navigate to http://sparks-vm24.i3s.unice.fr:3000/channels
   - Click on any channel to visualize data
   - You can also export the data as CSV format.

CONCLUSIONS

- ✓ You setted up a simulator for a "Smart-Home" environment, and you can now easily develop more complex things for the implementation of a Home Automation. You have finally sent the publication of your data to a ThingSpeak server, allowing you to store and view the data



- ✓ Finally, we can of course replace the Home I / O simulation system with real sensors and actuators by always using Node-RED and MQTT on the same model as the one we just presented.

INTERNET OF THINGS FOR EUROPEAN SMALL AND MEDIUM ENTERPRISES

Erasmus+ Strategic Partnership nº 2016-1-IT01-KA202-005561

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+

Erasmus+ Strategic Partnership n° 2016-1-IT01-KA202-005561 IoT4SMES - Internet of Things for European Small and Medium Enterprises