IoT 2023 CHALLENGE 2

(1) Name: Fontana Nicolo' Person Code: 10581197
(2) Name: Gerosa Andrea Person Code: 10583298

Node-Red nodes and which is their function (all parsing operations are performed with JSON):

Subscribe to our own topic, filter messages by temperature unit, plot values and save payloads.

- **self pub** (MQTT input node): subscribes to the topic *"polimi/iot2023/challenge2/10581197"* of the broker *"broker.hivemq.com:1883"*.
- **filter Celsius** (function node): parses the payload of the messages received by the *"self pub"* node and checks if the unit of the message is Celsius; if so, it sends to the next nodes an array containing the upper bound of the value *"range"* of the message and the entire payload of the message.
- plot (function node): extracts from the received array the value of the temperature to be plotted.
- **Temperature** (chart node): plots the value received on a plot with Temperature over the y-axis and time over the x-axis.
  NB: the plot is not included in any attached file because we forgot to save it and then it was lost.
- save (function node): extracts the payload from the received array, parses it and propagates it.
- csv node: parses again the received payload to match the CSV format.
- **10581197_10583298.csv** (file-out node): save the received parsed payload into the *10581197_10583298.csv* file.
  NB: in the end, the headers of the columns (*description, type, unit, range, lat, long*) will be manually added to the CSV file.

Reset flow variables and chart.

- RESET (inject node): used to start the flow execution
- reset flow vars (function node): sets flow constants and resets all other flow variables (both useful for execution and debugging)
- reset chart (function node): at the beginning of the execution sends an empty array to the chart to clear it and remove previous plottings.

Loading the CSV file, handling random identifiers and re-publish matched messages.

- **id_code_generator** (MQTT input node): subscribed to *"polimi/challenge_2/2023/id_code_generator"* of the broker *"broker.hivemq.com:1883"* to receive the random identifier every 5 seconds
- load csv (function node): behaves as a gate to execute the branch which loads and parses the original CSV file
- **challenge2023_2.csv** (file-in node): actually load the *challenge2023_2.csv* file
- csv node: parses the loaded CSV file ad sends 1 message for each packet.
- **csv parsing** (function node): parses each packet and extracts the useful information in it (i.e. number, info and message) and stores the number and the message only for the packets containing at least

one *“Publish Message”*. Those values are stored into flow variables (to be accessed more easily by other nodes): *”nums”* (array with 1 packet identifier for each Publish Message), *”pub_msgs”* (array with the message of each Publish Message).

- **rnd id processing** (function node): extracts the random identifier from the message sent by the broker, adds the last digits of our person-code (i.e. 1197), computes the module operation with respect to the total number of packets in the CSV file (i.e. 7711).
  Handles the creation of the payload for the starting message (“START”) and the ending message (“END”).
- **re-pubs creation** (function node): for each random identifier received from *“rnd id processing”* check if in the flow array *”nums”* there is a matching packet number and for each matching packet number create the object with the values to be re-published (i.e. timestamp captured when creating the object, original random identifier received from the broker, payload of the matching message, taken from the flow array *”pub_msgs”*). All the objects created are sent to the next node into an array.
  The initial and the final messages (“START” and “END”) are just propagated.
- switch node: handles the distinction between initial/final messages and matching-packet messages. It sends the initial/final messages directly to the *re-pubs* node to be published, while the messages containing the packets matching the random identifier are sent to a split node.
- split multiple publish (split node): splits the array containing all the info about matched MQTT messages into multiple messages, each one containing one MQTT message info.
- **re-pubs** (MQTT output node): publishes each message received to the topic *“polimi/iot2023/challenge2/10581197”* of the broker *“broker.hivemq.com:1883”*.

Other nodes are present just for debugging purposes.