Project Report: Challenge 2 - IOT

Alberto Latino - **10600138**, *Lorenzo Mainetti* - **10622242**

Goal:
Generate 100 messages from a CSV file to be sent to the ThingSpeak channel using MQTT through Node-RED

1. **ThingSpeak setup (https://thingspeak.com/channels/1712048)**
   We first set up the ThingSpeak channel, making it public and adding the requested dashboards: one chart and one lamp per field (field1, field2, field5).
   The ThingSpeak channel stores data sent from the Node-RED application through MQTT. Hence, we created a MQTT Device with our credentials.

2. **Node-RED application**
   We built a flow that meets the challenge goals and requirements. Below the nodes that we used and a description on how we used them. We created 2 flows starting from the same CSV. One sends messages through MQTT and the other creates the RSSI chart.

   CSV reading
   - **Inject Node**: it is used to manually trigger the flow, no repetition is set
   - **File Node**: here we selected the CSV file from which we generate the 100 messages
   - **CSV Node**: we inserted the csv columns and we set the output to “a message per row”, this way a message is generated for each row of the CSV.

   Message selection - 1st flow
   - **Function Node**: here we wrote the javascript code in order to filter the CSV. In particular we returned all the messages with msg.payload.code in the interval [138,237]. Also, we created the msg.payload and msg.topic by using the fields 1, 2 and 5 that were extracted from the csv msg.

   MQTT publishing - 1st flow
   - **Delay Node**: this node allowed us to output 2 messages per minute
   - **MQTT out Node**: this node allowed us to connect and send data through MQTT to the ThingSpeak platform.

   Node-RED Chart generation - 2nd flow
   - **Function Node**: this function node filtered the messages coming from the csv node returning the ones with code in [138,237] and created a message that only had field5 in the payload. This way the Node-RED chart received it as a number and created the chart of RSSI.
   - **Delay Node**: the delay node allowed to generate 2 messages per minute
   - **Chart Node**: this node is the one responsible for creating the chart given the msg as input.

We also added a **Debug Node** to print the msg.payload and to check that everything works as expected. They were useful to understand what was going on in the flow and helped to fix initial errors.