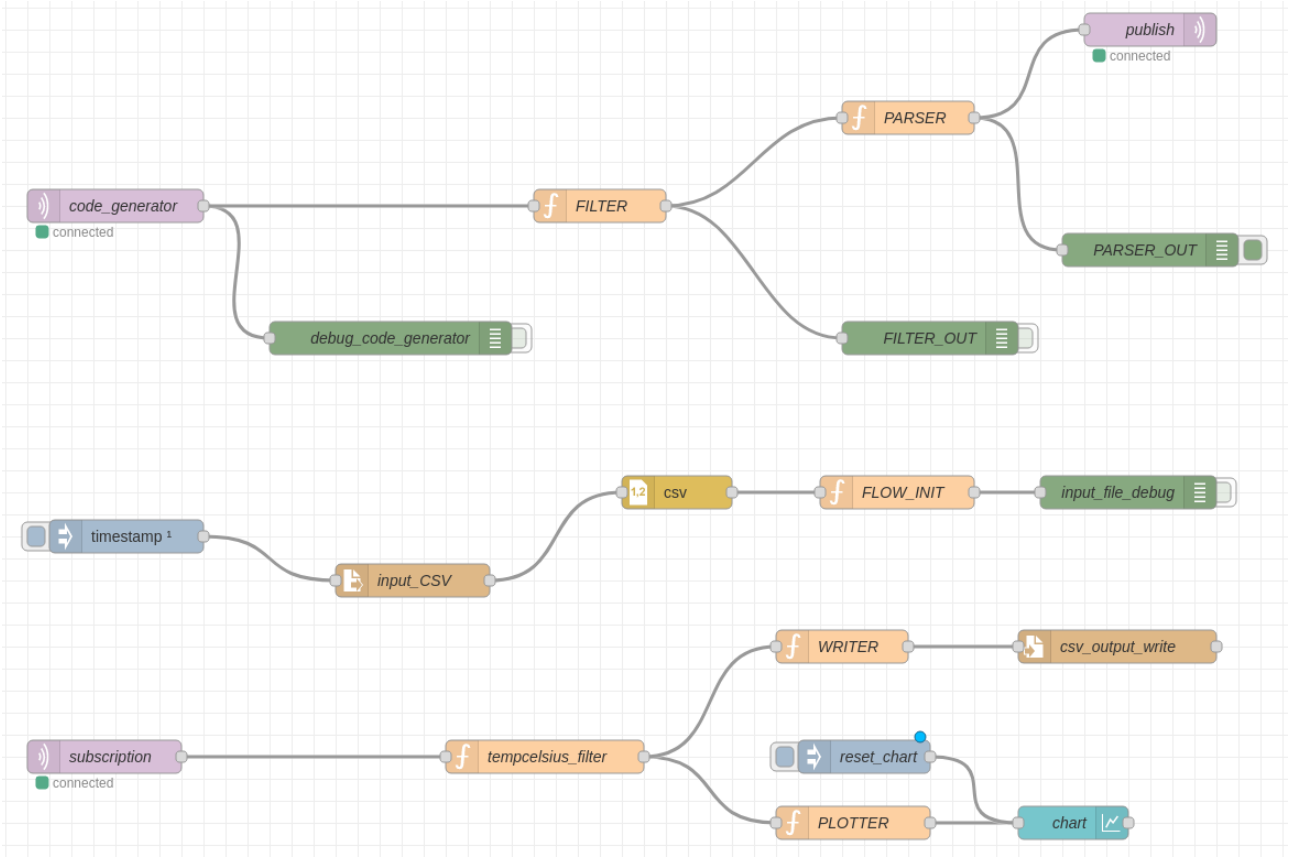
IoT 2023 CHALLENGE2

(1) Name: Giovanni De Lucia Person Code: 10700658

(2) Name: Lorenzo Battiston Person Code: 10618906

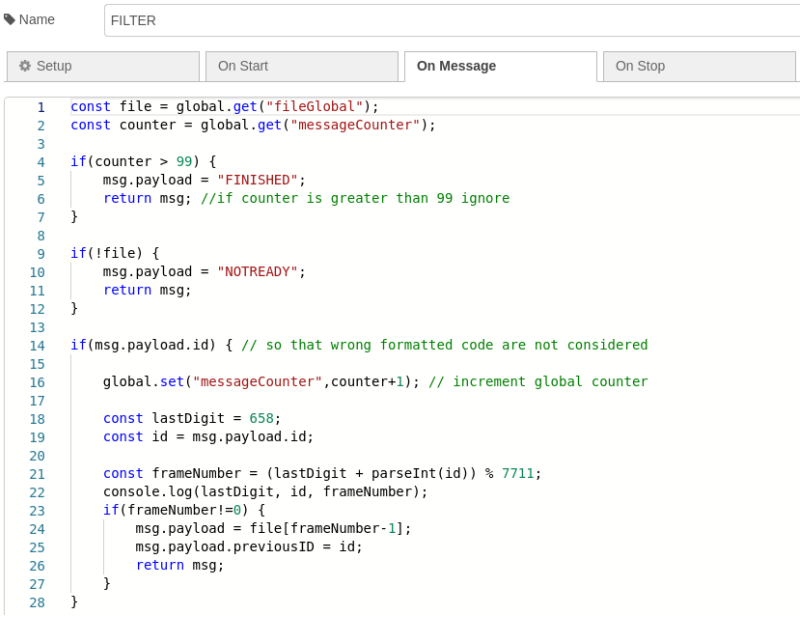
NOTE: All the files in the flow are referred with a local machine path.

Node-red Flow



- **code_generator**: subscribes to the topic “polimi/challenge_2/2023/id_code_generator”
- **FILTER**: detailed below
- **PARSER**: detailed below
- **Publish**: publish the various Publish Messages Payload to the topic “/polimi/iot2023/challenge2/10700658”
- **Timestamp**: at the beginning of the deploy dictates the input_CSV to read the file
- **input_scv**: read the input file
- **csv**: parse the input csv file
- **FLOW_INIT**: detailed below
- **Subscription**: subscribes to the topic “/polimi/iot2023/challenge2/10700658” to receive the messages that are published
- **tempcelsius_filter**: Filters the payloads of type temperature and unit “C”
- **WRITER**: prepare the payload to be appended to the .csv output file.
- **csv_output_write**: write to the output file called “10700658_1061806.csv”
- **Plotter**: extract the upper bound from the range array and gives it to the **chart**

Filter Function Node




```
const file = global.get("fileGlobal");
const counter = global.get("messageCounter");

if(counter > 99) {
    msg.payload = "FINISHED";
    return msg; //if counter is greater than 99 ignore
}

if(!file) {
    msg.payload = "NOTREADY";
    return msg;
}

if(msg.payload.id) { // so that wrong formatted code are not considered

    global.set("messageCounter",counter+1); // increment global counter

    const lastDigit = 658;
    const id = msg.payload.id;

    const frameNumber = (lastDigit + parseInt(id)) % 7711;
    console.log(lastDigit, id, frameNumber);
    if(frameNumber!=0) {
        msg.payload = file[frameNumber-1];
        msg.payload.previousID = id;
        return msg;
    }
}
```

It reads the file value and the counter. If the counter exceeds the limit, then it returns a FINISHED message so that the Parser knows to not process the message. Similarly in the case that the file has not been read. Otherwise, it processes the message, increments the counter, extract the packets with the calculated frame number and pass it to the parser.

Parser Function Node

```
if (msg.payload != "NOTREADY" && msg.payload != "FINISHED") {
    //todo: should parse the msg payload and return couples publish, payload

    const messages = msg.payload["Info"]; // messages
    let payloads = msg.payload["Message"]; // payloads

    const publishMessages = messages.split(",").filter(str => str.startsWith("Publish Message"))

    let emptyPublishPayloads = publishMessages.length;

    const previousID = msg.payload.previousID;

    if(payloads) {
        // check for malformed packets
        const lastClosedBrackets = payloads.lastIndexOf("}");
        const lastOpenBrackets = payloads.lastIndexOf("{");
        let malformed = "";

        if (lastClosedBrackets < lastOpenBrackets) {
            malformed = payloads.slice(lastOpenBrackets, payloads.length);
            payloads = payloads.slice(0,lastOpenBrackets-1);
        }

        const parsedPayloads = JSON.parse('[' + payloads +']');
        emptyPublishPayloads -= parsedPayloads.length;
        parsedPayloads.forEach(payload => {
            msg.payload = {
                "timestamp": Date.now().toString(), "id": previousID, "payload": JSON.stringify(payload)
                };
            node.send(msg);
        })
        if(malformed.length > 0) {
            emptyPublishPayloads -= 1; // it does count as a non-empty payload
            msg.payload = {"timestamp": Date.now().toString(), "id": previousID, "payload": malformed};
            node.send(msg);
        }
    }

    for(let i=0;i<emptyPublishPayloads;i++) {
        msg.payload = {
            "timestamp": Date.now().toString(), "id": previousID, "payload": ""
        };
        node.send(msg);
    }
}
```

It counts the Publish Messages (let's say N), the Payloads (let's say M) and the malformed payload (at most 1). Then send M payloads, N-M (-1 if a malformed payload is present) empty payloads and the malformed payload if present.

FLOW_INIT Node

Name: FLOW_INIT

Setup | On Start | **On Message** | On Stop

```
1  global.set("fileGlobal",msg.payload); // save the parsed input file in a global variable
2  global.set("messageCounter",0); // init the counter
3  return msg;
```

Final Run