# IoT Challenge 3

Adriano Totaro 10701581
Francesco Maria Tranquillo 10674562
Jacopo Vismara 10699231

April 2024

## Contents

## 1 Flow

Complete node-red flow:

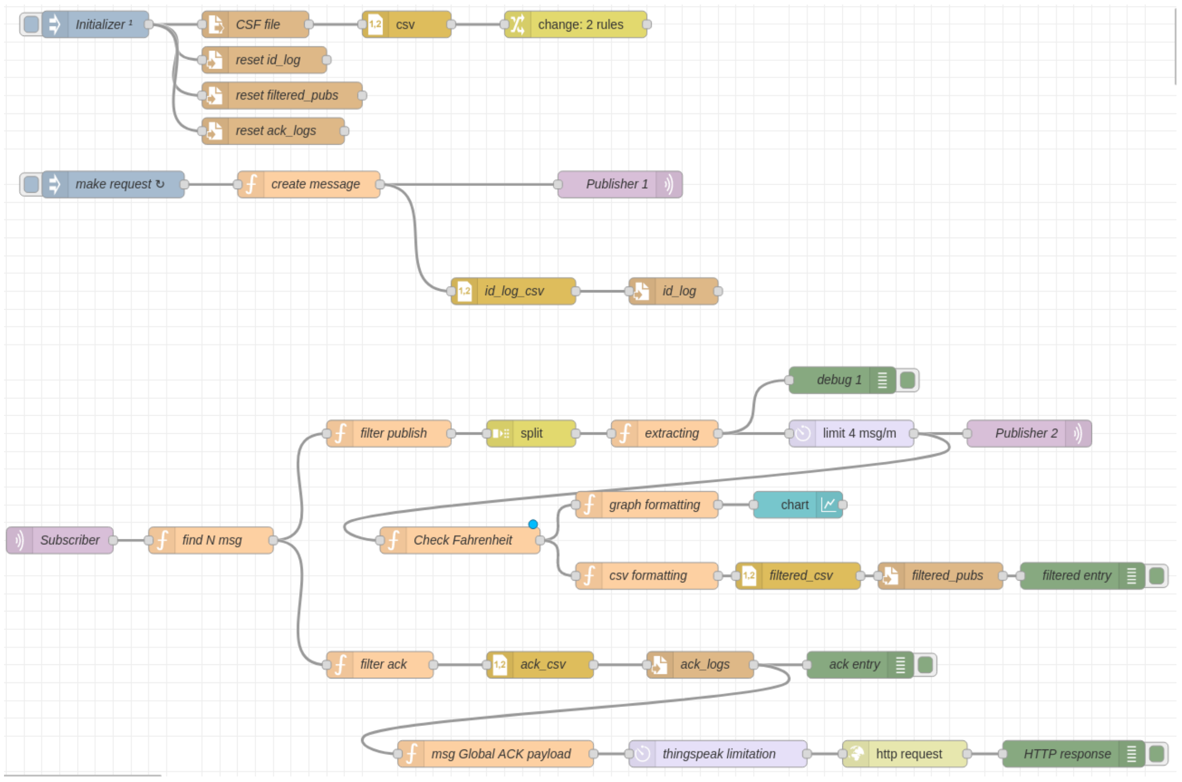

Figure 1: Complete Flow

To better see the image *click here*

In order to explain the complete flow we are going to explain how each node works.

## 2 Initialization

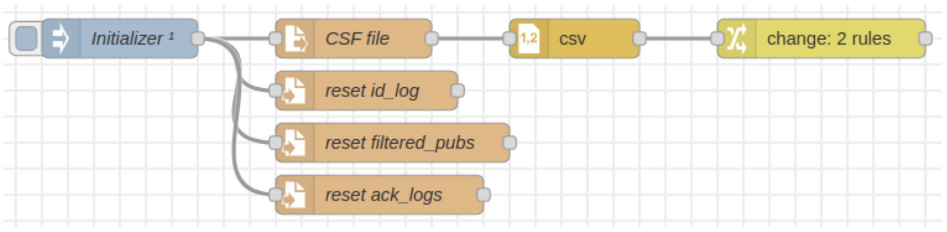

Figure 2: Initialization

After deploying the node red project an ”Initializer” block send a message to 4 blocks. The first block reads the file ”challenge3.csv”, while the three others remove, if existing, the files: ”id_log.csv”, ”filtered_pubs.csv”, ”ack_log.csv”.
After reading the file ”challenge3.csv”, using the first row for the name of the columns and using the comma as a separator for the fields, the output is saved in a global variable named ”csv_file”.

## 3 Publish

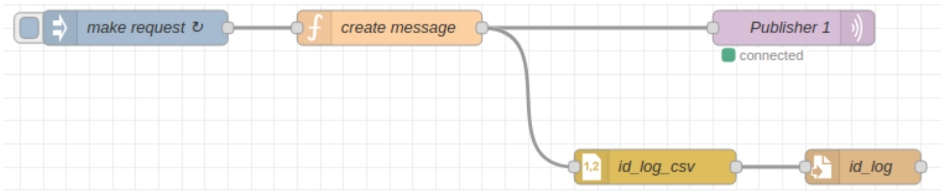

Figure 3: Publish

The second group of blocks has the role to make a publish every 5 seconds at the topic ”challenge3/id_generator” and saving the publish message in a csv file.
In order to do so we firstly created a ”make request” block to trigger a message once every 5 seconds. Then a function block ”create message” will construct the fields of the message using a random number (between 0 and 50000) for the ID field, in the payload of the message. A context variable ”counter” will be used to fill the field ”No.” in the payload of the message and at each message sent the variable will be increased by 1.

In order to stop the flow after receiving 80 messages we also check in this block if a global variable ”rec_counter” is greater or equal to 80.

After the creation of the message, a pair of nodes ”id_log_csv” and ”id_log” is used to save the publish message in a csv file named ”id_log.csv”.
The message is also sent to a ”Publisher 1” block, which is connected to the local blocker in

the same machine (127.0.0.1) on the port 1884, that will publish the message on the topic specified in the filed ”topic”, created by the function block ”create message”.

## 4 Subscribe

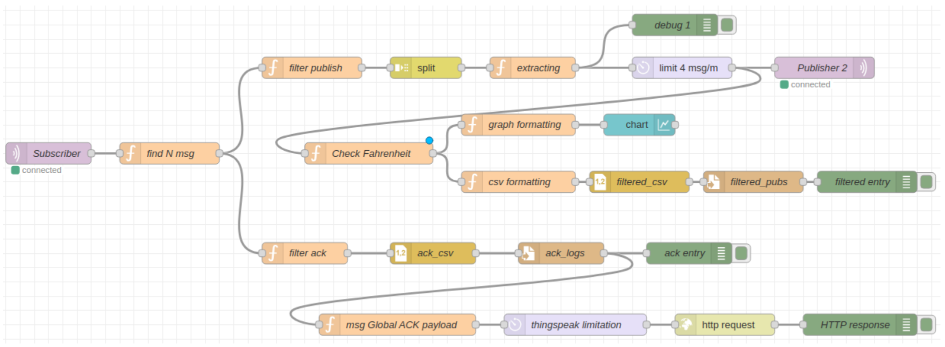

Figure 4: Subscribe

The last group begins with a ”Subscriber” block, used to subscribe to the topic ”challenge3/id_generator” on the local broker.
The messages received by the ”Subscriber” block are then sent to the function block ”find N msg” that firstly checks if the ”rec_counter” is greater or equal to 80, in which case returns an empty message NaN.
Then increases the global variable ”rec_counter” and retrieves the message from the global variable ”csv_file” that has the frame number equals to the remainder of the division between the message ID and 7711.
Since the frame numbers begin from 1, if the reminder of the division is equal to 0 we took the message with frame number 7711.
In the end the ”find N msg” returns a message which contains the newly found message concatenated to the ID used to find it.

After this the flow splits in 2 groups.

## 4.1 Filtered Publish

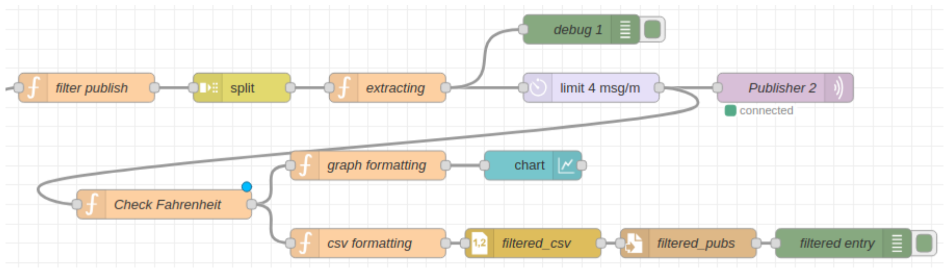

Figure 5: FilteredPublish

In this group of blocks we filter the publish messages, then check if the number of payloads coincide with the number of publishes contained in the message (else we create an empty payload).
Then we create an array containing all the publish messages and their respective payloads, adding the field "timestamp" and using for the "topic" field their own.
With the blocks "split" and "extracting" the array of publish messages is splitted and treated as a single message.
In the block extracting we also parse the payload and check if it is malformed, in which case it is substituted with an empty one.

A rate limiter block is then used to limit the publish messages at 4 per minute and the exceeding messages are saved in a queue to be sent later. Even if the global counter "rec_counter" is 80, the queue keeps flushing the saved messages.

The publish messages are then published to the local broker with the block "Publisher 2" and also further filtered with the block "Check Fahrenheit" that checks if the publish message is a temperature type with "F" as unit of measure. In this case the message is also formatted and plotted in a graph with the blocks "graph formatting", "chart" and saved in a csv file "filtered_pubs.csv" with the blocks "csv formatting" used to set the parameter of each element in the csv file in a certain order, "filtered_csv" and "filtered_pubs" used to save the entry in the csv file.

### 4.2 Ack Blocks

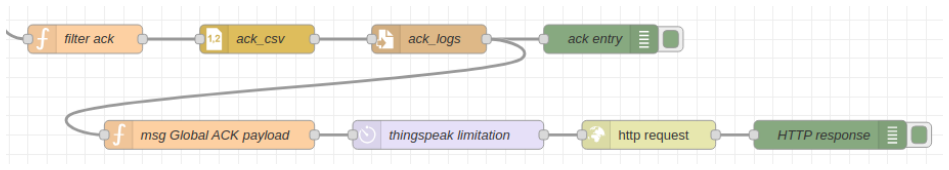

Figure 6: AckBlocks

In the second block only the publish messages representing an ack pass the filter (also the one containing the scring "[Malformed Packet]" are considered, they are not discarded). A global variable "ACK_counter" is also increased.
Then the ack messages are saved in a csv with the blocks "ack_csv" and "ack_logs".
With the last 3 blocks "msg Global ACK payload", "thingspeak limitation", "http request" the ack counter is sent to the thingspeak channel at the field1 (filled in the function block "msg Global ACK payload") with a rate limiter of 1 message every 16 seconds in order to comply with the requests limitation of thingspeak.

## 5 Simulation

After running a simulation we obtained the following chart, representing only the publish messages with a temperature type and Fahrenheit as unit of measure.

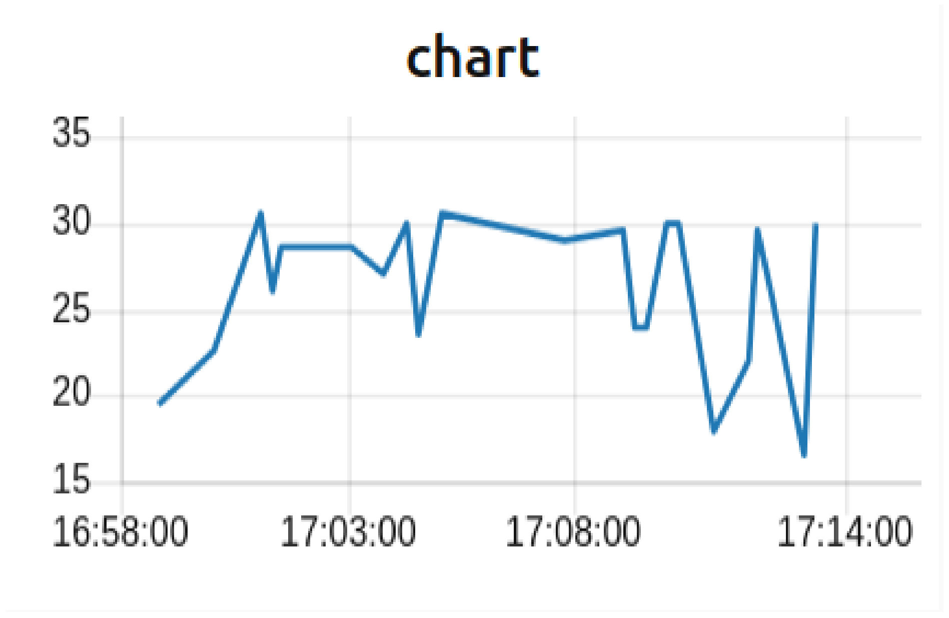

Figure 7: chart

## 6 Channel ID Thingspeak

*This* is the FREE thingspeak channel used for the project.