Ubiquitous Computing - Lab 2: Node-RED

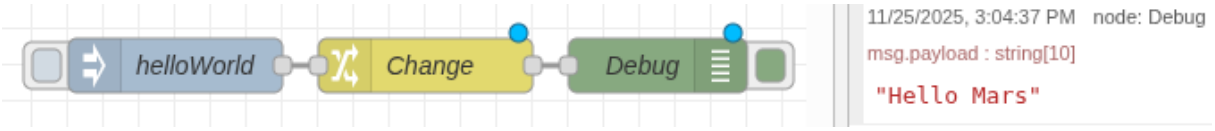
Vianney Hervy

All the work done here (along with the flows as JSON) is available on my GitHub[1].

1. Exercise 1

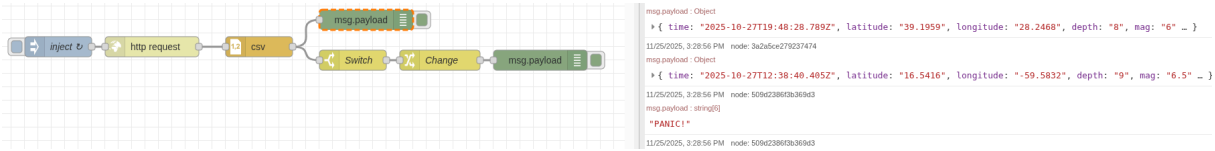
This flow contains two nodes: *helloWorld* and *Debug*. *helloWorld* is an inject node that has a string payload of value `Hello World`. *Debug* is a debug node that has an output set to `msg.payload` in order to output the payload received from *helloWorld*.

2. Exercise 2



This flow contains an additional *Change* node to the ones of the Exercise 1. *Change* is a change node which has a Change rule with `msg.payload` as input, searches for `World` and replaces with `Mars`.

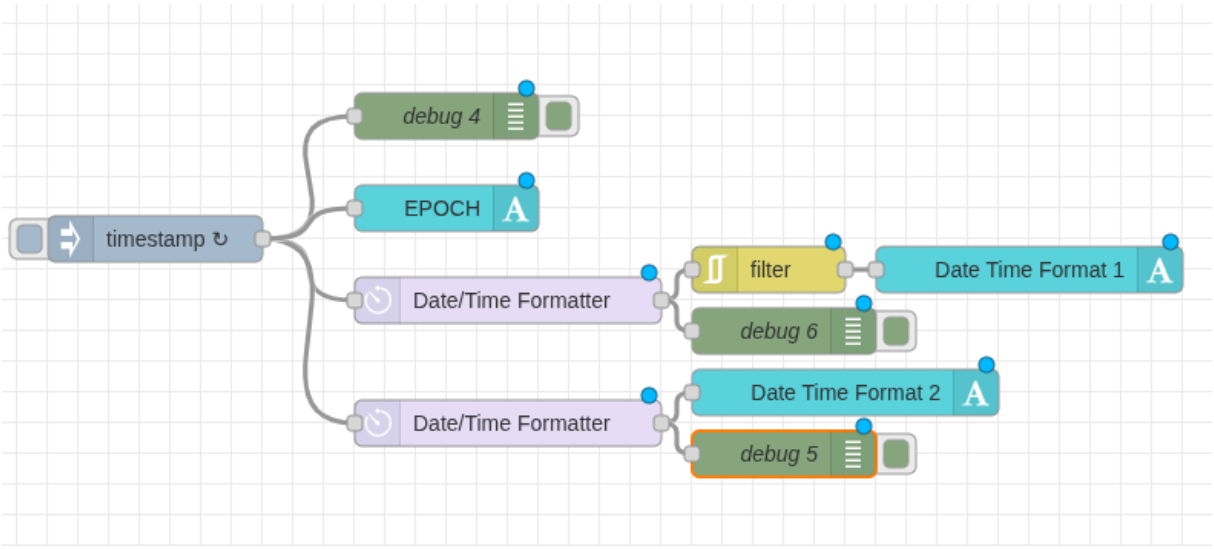
3. Exercise 3



This flow is made with the `>= 6` rule in the *Switch* node. The top debug node outputs the earthquake data (line per line) at every injection. The bottom debug node outputs `PANIC!` for every earthquake that has a mag(nitude) superior or equal to 6.

[1] https://github.com/sully-vian/HTWG/tree/main/ubiquitous-computing/lab-2

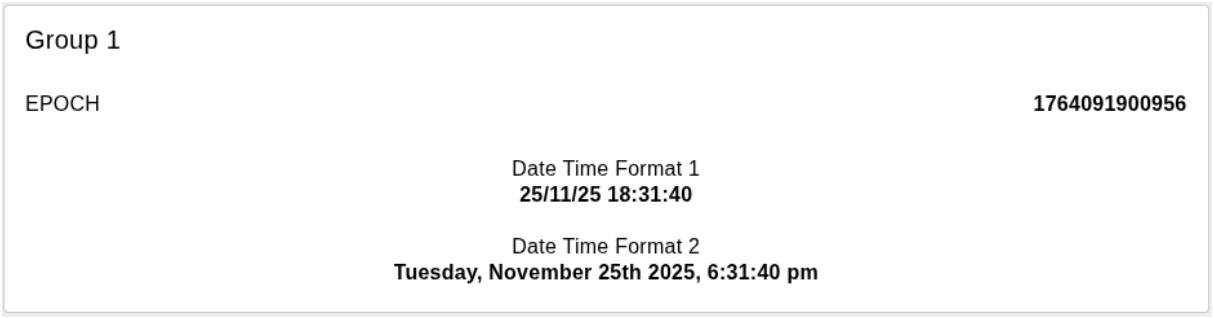
4. Exercise 4



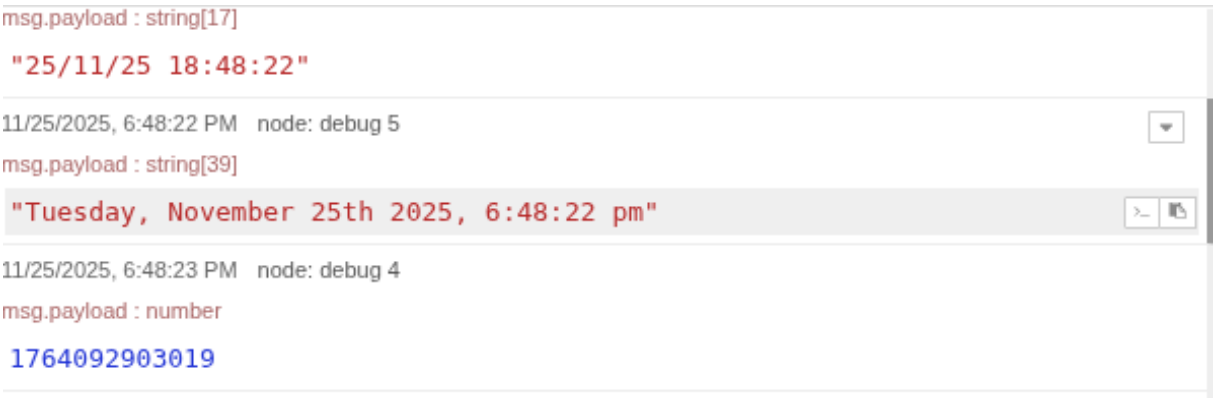
This flow uses two "Date/Time Formatter" to format the raw epoch of the inject node. To get the short format of "Date Time Format 1", we should specify `DD/MM/YY HH:mm:ss` as output format. To get the longer format of "Date Time Format 2", we shoud specify `dddd, MMMM Do YYYY, h:mm:ss a` as output format.

I'm not sure I get what the "filter" node is used for in this flow, so I just used the default configuration.

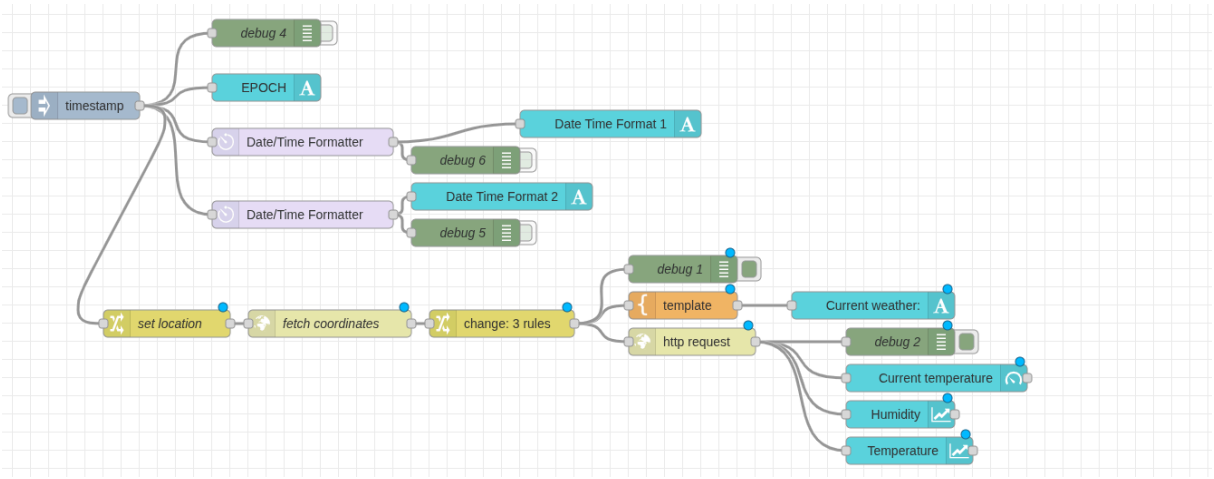
Inputing the dates into "text" nodes of the @flowfuse/node-red-dashboard collection allows to us to display the results in the following dashboard:



Of course, the debug nodes also print the displayed content in the debug window:



5. Exercise 5



This flow is sensibly more complex asthe displayed information comes from two different sources: the timestamp and the Open-Meteo weather API[2].

The weather flow goes as such:

- **Change node:** Set `msg.payload` to "Konstanz" (or the city of your choice)
- **HTTP request node:** Fetch the city GPS coordinates with the following url: `https://geocoding-api.open-meteo.com/v1/search?name={{payload}}&count=1`
- **Change node:** Pick up the fetched values and bring them to the message's payload's root for easier access.
- **HTTP request node:** Fetch the weather data with the acquired GPS coordinates: `https://api.open-meteo.com/v1/forecast?latitude={{payload.latitude}}&longitude={{payload.longitude}}&current=temperature_2m,relative_humidity_2m`
- **Dashboard nodes:** Displatch the fetched information to the different dashboard nodes.

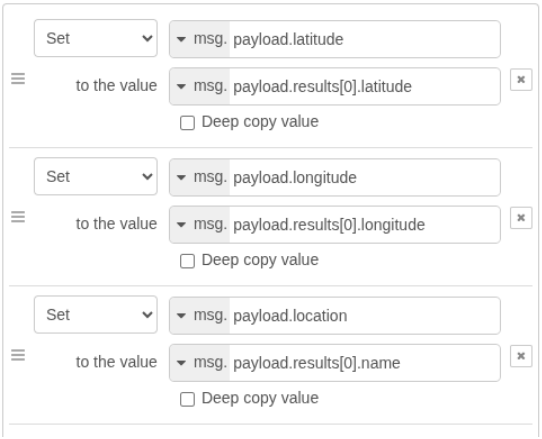


Figure 1: Configuration of the second change node

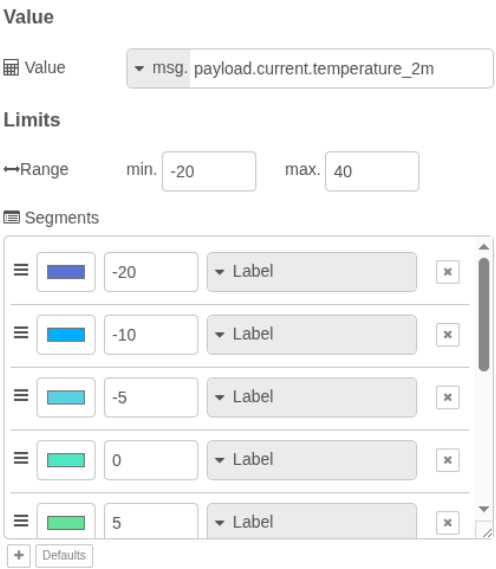
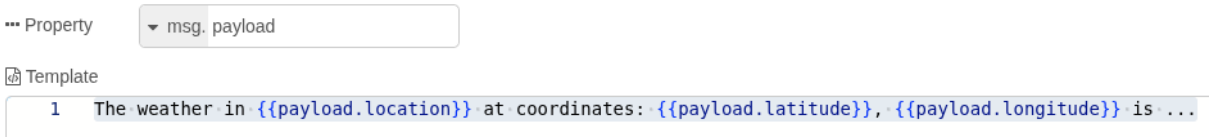


Figure 2: Configuration of the gauge node

Concerning the "The weather in ..." sentence, I used a template node with the following template which I then plugged into a regular dashboard text node.

[2]https://github.com/sully-vian/HTWG/tree/main/ubiquitous-computing/lab-2



I didn't manage to find a service that provides textual information such as the "cloudiness" shown in the subject example.

The resulting dashboard after a few hours of gathering data (weather API kicked me out and hid the temperature gauge value):