Node-RED as an IoT Application Development Tool

Difficulty Level: Easy

Objective

Gain basic understanding of Node-RED as a visual flow programming tool that can support IoT application design and development.

Achievements

The skills to be acquired at the end of this module:

- Getting familiar with Node-RED as a visual IoT prototype development tool
- Ability to prototype and practice IoT applications on a PC without relying on any specific IoT hardware
- Installing and running Node-RED as a docker container on your computer
- The flow concept and how to create data flows in Node-RED

Background

Flow-based programming (FBP) is a programming paradigm that focuses on the information flow within a system and defines how data should travel and be processed through different entities in the system.

Node-RED (https://nodered.org/) is a popular FBP tool, originally developed by IBM and now a part of the OpenJS Foundation. It consists of a runtime based on Node.js and a web-based flow editor.

The "flow editor" in Node-RED allows you to create your application within the browser by dragging nodes from your palette into a workspace and wiring them together. With a single click, the application can then be deployed back to the runtime for execution. The palette of nodes in Node-RED can be easily extended by installing new nodes that have been created by the community, and you can easily share your flows (applications) with others as JSON files.

Installing and Running Node-RED

We will first install Node-Red on our PC. We can either do this directly on our operating system (by downloading and executing the installation file) or we can install it as a Docker container, as we will follow here.

Co-funded by the Erasmus+ Programme of the European Union

In order to run Node-RED in a Docker container, we need to have Docker application running on our system. Docker is a widely used virtualization technology that allows us to run multiple "*virtual machines*" (in the form of "*containers*") on a single hardware and operating system.

If Docker is not already installed on your system, you can download and install the correct version for your operating system (Windows, Linux, or Mac): https://docs.docker.com/get-docker/.

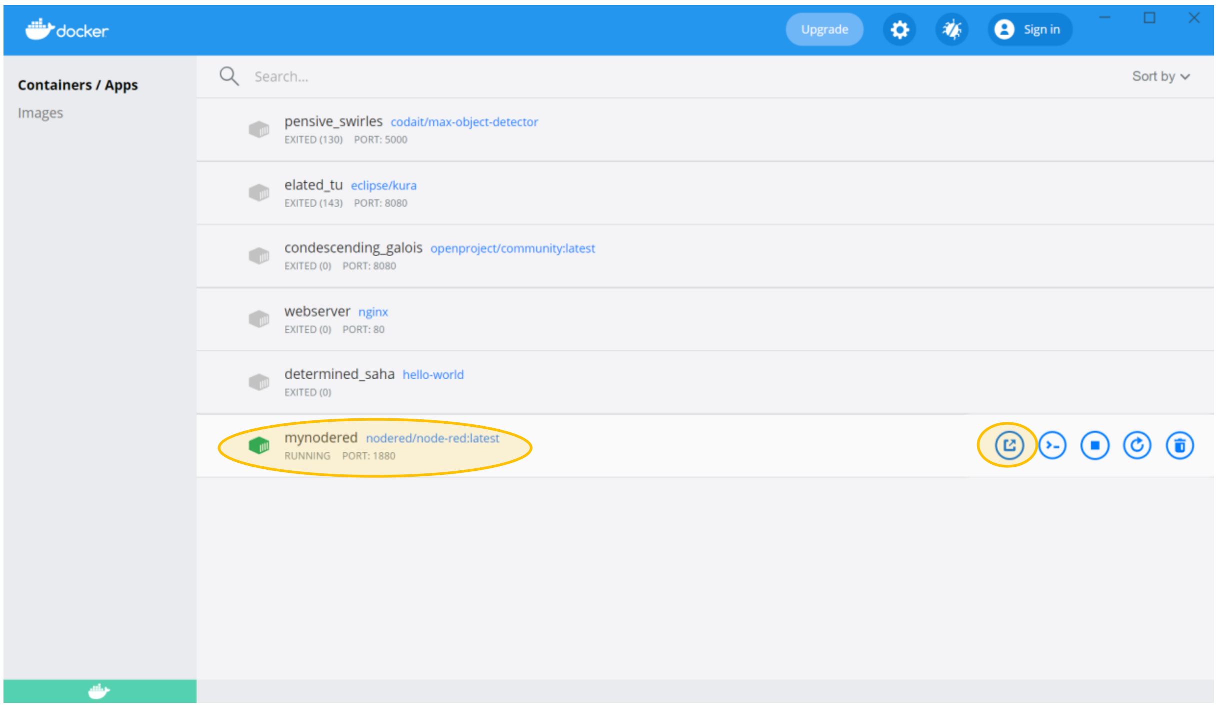
After Docker is installed and is running, open a command prompt. (On Windows, you can open the start menu and start typing "cmd". Alternatively, you can type "Powershell", which provides a better Linux-like command shell.)

In order to install Node-RED container, type the following on the command line and press enter:

```
docker run -it -p 1880:1880 -v node_red_data:/data --name mynodered nodered/node-red
```

This command pulls the Node-RED system image, installs, and runs it as a container in Docker. In case you have any issues with the installation, you can see further details and instructions here: https://nodered.org/docs/getting-started/docker.

After successful installation, you should be able to see the Node-RED instance (container) running on your Docker, as shown in Figure 1.



pensive_swirles codait/max-object-detector
EXITED (130) PORT: 5000

elated_tu eclipse/kura
EXITED (143) PORT: 8080

condescending_galois openproject/community:latest
EXITED (0) PORT: 8080

webserver nginx
EXITED (0) PORT: 80

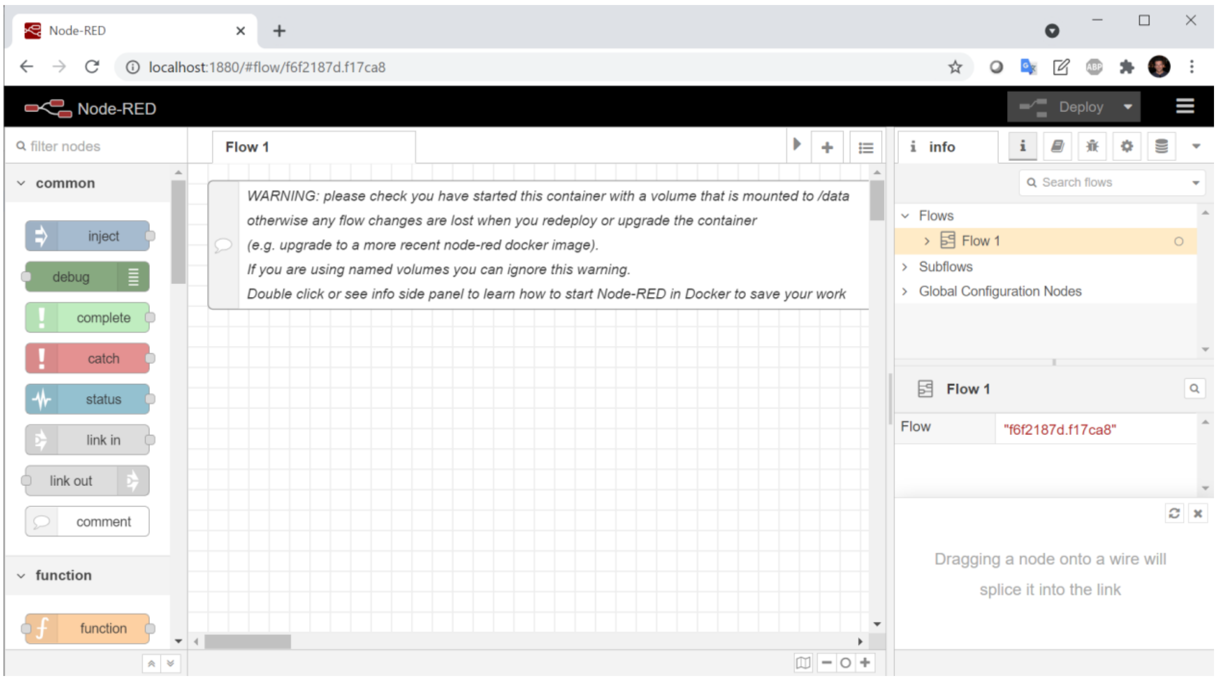
determined_saha hello-world
EXITED (0)

mynodered nodered/node-red:latest
RUNNING PORT: 1880

Figure 1. Docker application, showing the running Node-RED container instance

Clicking on the button next to the "mynodered" container will open up the Node-RED web interface in your browser. Note that this page points to http://localhost:1880/, which is a web page that runs locally on your PC. You can also open your browser and enter the same URL (http://localhost:1880/) to manually open the Node-RED user interface (UI), as long as the Node-RED Docker container is running.

After the previous step, you should be able to see the initial Node-RED web UI, depicted in Figure 2.



WARNING: please check you have started this container with a volume that is mounted to /data otherwise any flow changes are lost when you redeploy or upgrade the container (e.g. upgrade to a more recent node-red docker image). If you are using named volumes you can ignore this warning. Double click or see info side panel to learn how to start Node-RED in Docker to save your work

Dragging a node onto a wire will splice it into the link

Figure 2. Node-RED web interface

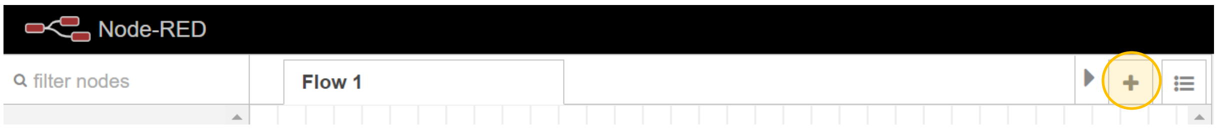
"Nodes" and "Flows" in Node-RED

The core elements in Node-Red are the "**nodes**", which are functional components with a visual representation, as shown on the left-side panel ("node palette"). In Figure 2, we see some of these nodes; namely "inject" node, "debug" node, "function" node, etc. Also note that the large set of nodes in Node-RED are typically organized in groups or clusters. In this case, we see the "common" group at the top and a "function" group underneath, each containing a set of nodes.

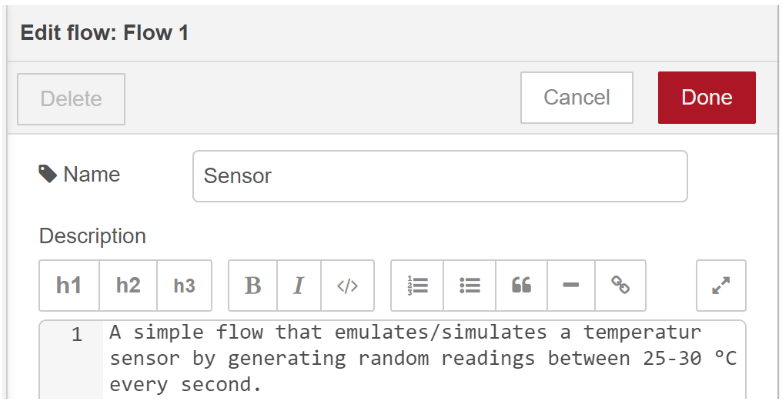
At the center of the screen is an empty area, with a tab title on top: "Flow 1". A "**flow**" in Node-RED represents a set of connected nodes. We can drag and drop nodes from the left panel onto this area and then interconnect those nodes to start composing an application with a well-defined data flow across several nodes.

Co-funded by the
Erasmus+ Programme
of the European Union

We can create multiple flows in a single instance of Node-RED to logically combine different parts or functionalities of our application. In order to create a new flow, use the "+" button located at the far right of the flow tab, as depicted below.



Once we have multiple flows, it is a good idea to give a descriptive name to each flow and even provide a brief description of its intended purpose. To do that, you can double-click on the flow name and a pop-up menu will come up:



Name: Sensor

```
A simple flow that emulates/simulates a temperatur
sensor by generating random readings between 25-30 °C
every second.
```

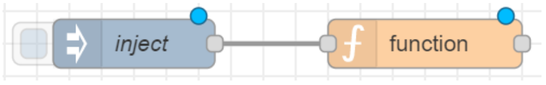
Creating your first flow / application in Node-RED

It is best to describe the general process of application development with an example. We start by opening the Node-RED web interface. In this simple example, we will compose a virtual program that emulates a temperature sensor.

Our emulated sensor will generate a random number between 25 and 30 at every second, which represents a temperature reading in °C (Celsius). Node-RED's "function" node is a very versatile node that fits this purpose perfectly.

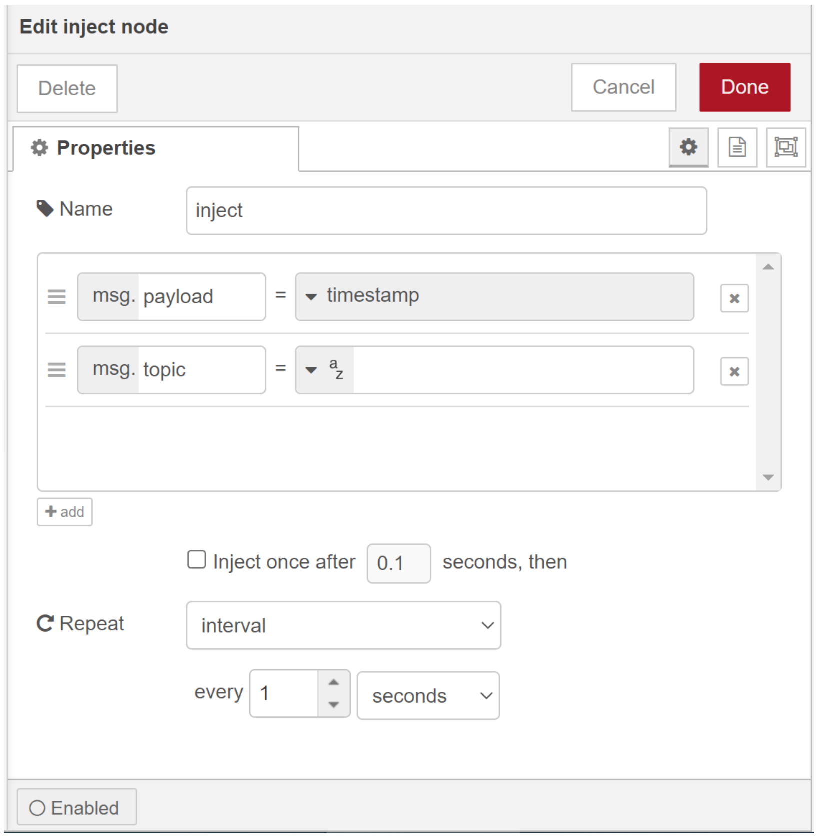
We will also use the so-called "inject" node to trigger the generation of a new random number periodically (once every second).

- Drag a "**function**" node from the node palette and drop it onto the flow area.
- Add an **inject** node into the flow; connect the output of the **inject** node to the input of the **function** node, as depicted below.



Co-funded by the
Erasmus+ Programme
of the European Union

Now let's configure the **inject** node. Open the node's properties by double clicking on it. Set the repeat option to "Interval" and set the interval value to 1 sec, as shown next.



Name: inject
msg. payload = timestamp
msg. topic =
Inject once after 0.1 seconds, then
Repeat: interval
every 1 seconds

Figure 3. Settings of the *inject* node

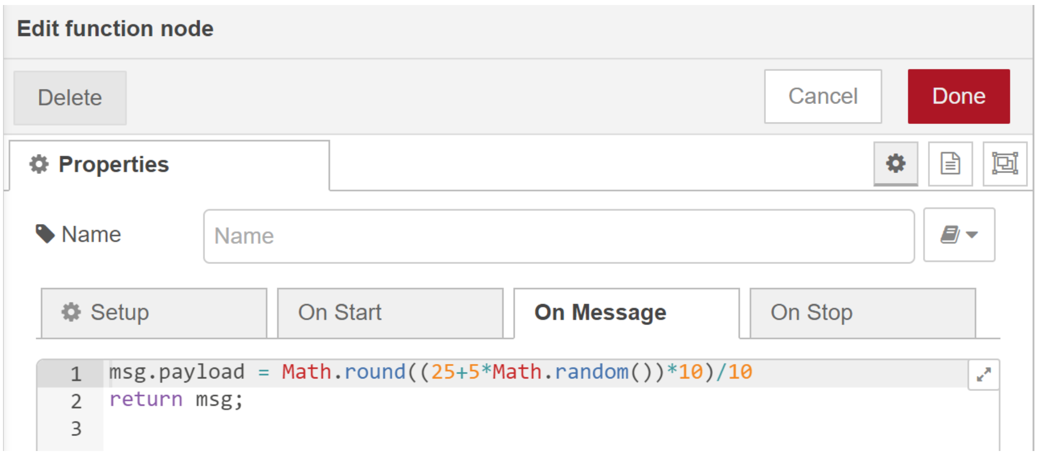
The inject node can also be used to create a timestamp and some data, but for this example we just need its ability to generate a periodic trigger. Now we have a function that gets called at defined intervals. We are ready to configure the sensor by programming the **function** node so that it generates random values in our target range.

To do this, double click on the **function** node. The window that opens allows us to add JavaScript codes once at the start/stop of the function or whenever there is an incoming message to this function node. In our case, we use an inject node to trigger this function, so we add our code under the "On Message" tab.

We want to generate a number between 25 and 30, with up to one decimal, to represent our temperature value, and add it to our message payload, as depicted in Figure 4:

```
msg.payload = Math.round((25+5*Math.random())*10)/10
return msg
```

Co-funded by the
Erasmus+ Programme
of the European Union



Edit function node

Delete | Cancel | Done

Properties

Name: Name

Setup | On Start | On Message | On Stop

```
msg.payload = Math.round((25+5*Math.random())*10)/10
return msg;
```

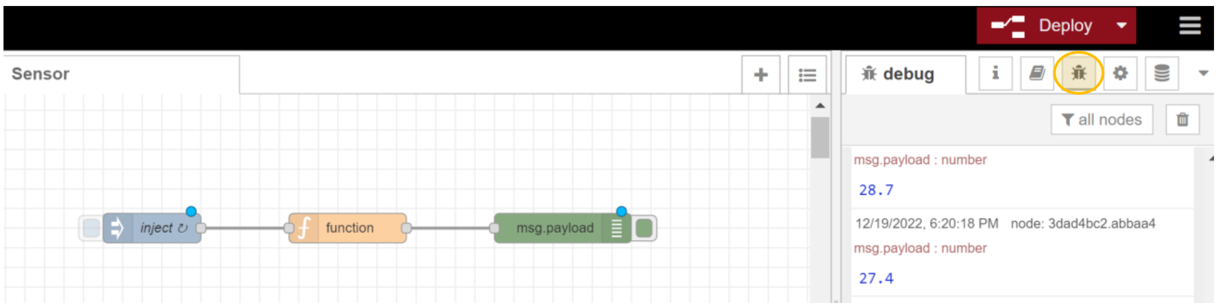
Figure 4. Editing the settings of the *function* node

Once we are ready, we can click on "Done" button to close this editing dialog.

Now, our simulated sensor is actually ready, but in order to be able to see the generated sensor readings, we will add another node, called "**Debug**". As its name implies, the debug node allows us to analyze, or debug, our application functionality.

- Go ahead and add a **debug** node from the node palette onto the flow area.
- Connect the output of the **function** node to the input of the **debug** node.

We can now deploy/run our application, simply by clicking on the red "**Deploy**" button on the upper right of the window.



```
msg.payload : number
28.7
12/19/2022, 6:20:18 PM   node: 3dad4bc2.abbaa4
msg.payload : number
27.4
```

Figure 5. Deployed flow with the generated sensor readings shown in the debug panel

While the application is running, we should be able to see the outputs of our sensor in the "**debug window**" of the **right side panel**. This is accessible by clicking on the **'bug' icon**, as shown in Figure 5. We see that the sensor readings are coming in every second.