Department of Electronic & Telecommunications Engineering

University of Moratuwa

EN3250 Internet of Things / EN2560 Internet of Things Design and Competition

Node-RED Session 1 – Exercises

2019 Batch **Semester 4**

Prerequisites:

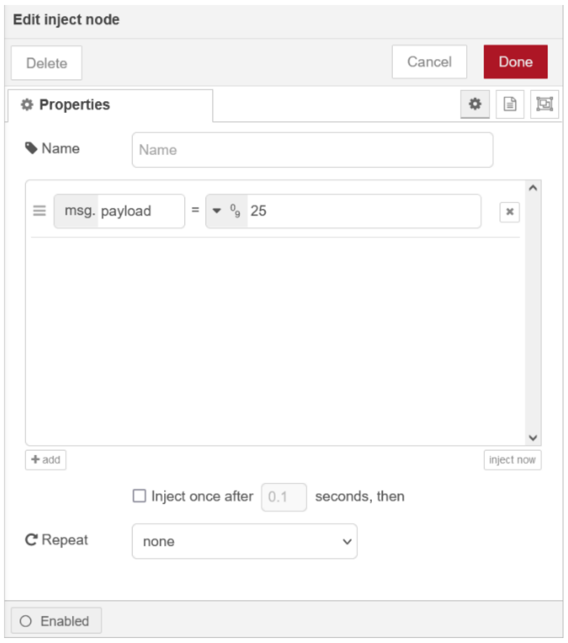
- Node-RED

Exercise 1: Familiarize with basics of Node-RED

In this exercise, we will be using the following basic nodes to familiarize with the Node-RED development environment:

- Inject Node
- Debug Node
- Filter Node
- Switch Node
- Change Node
- Function Node

1. Take four *Inject* nodes on to the flow design area.
2. Double click each inject node to open its properties tab
   a. Remove the item for *msg.topic* from the item list
   b. Set *msg.payload* type to number and set the following numbers to each node:
      - Inject node 1: 25
      - Inject node 2: 3
      - Inject node 3: 9
      - Inject node 4: 13



Name: Name

msg. payload = 25

☐ Inject once after 0.1 seconds, then

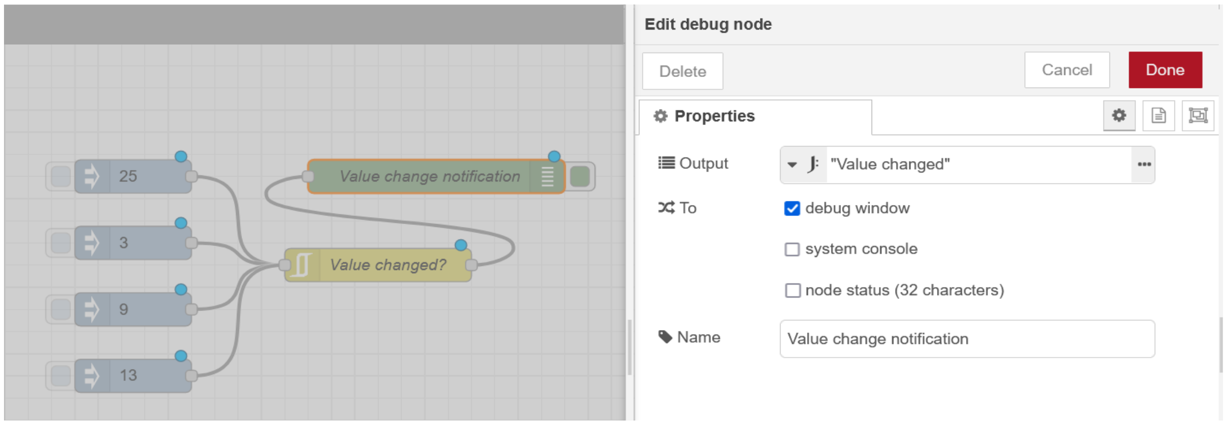
Repeat: none

Figure 1: Inject node configuration example

3. Insert a *Filter* node and connect the outputs of the inject nodes to the input of the filter node. Now, using the *mode* property, configure the filter node such that it will only pass the message forward only the input value changes.

Note: You may edit the *Name* property of nodes to give them meaningful display labels so that the flow is easier to understand/debug

4. Insert a *Debug* node to the output of the filter and configure it as follows,
   - Set output type to expression and put the string “value changed” as the expression.



Output: "Value changed"
To: [✓ debug window] system console, node status (32 characters)
Name: Value change notification

Figure 2: The flow and the configuration of the debug node after step 4

Deploy the flow and verify its functionality.

5. Insert *Switch* node to the output of the filter node and set the following paths,
   - Path 1: for the set of numbers greater than 20
   - Path 2: for the set of numbers that are divisible by 3
     (**Hint:** Use a JSONata expression)
   - Path 3: for the set of numbers that do not belong to the above sets of numbers

   If done correctly, you should now see three outputs in the Switch node.

6. Connect two debug nodes to paths 1 and 2 to receive meaningful text notifications in the debug window if the message goes into those paths
   - Path 1: “Greater than 20”
   - Path 2: “Divisible by 3”

7. Insert a Change node to the output of path 3 of the switch node and set the following rules,
   - Move the number in *msg.payload* to a new property *msg.received_number*.
   - Set msg.payload to the following JSON object containing your name and index number,

```
{
    "name": "<YOUR_NAME>",
    "index": "<YOUR_INDEX_NO>"
}
```

8. Connect a debug node to the output of the change node and set the output as "*complete message object*"

Deploy and verify the functionality of the flow.

9. Insert a *Function* node to the output of the change node and write a code in JavaScript to perform the following operation:
    - Extract the 6-digit index number from the message payload, add the number at *msg.received_number* to it and set the answer as *msg.payload.answer* (**Hint:** check the *parseInt* method)
    - Return the modified message

10. Connect a debug node to the output of the function node

Deploy the flow and verify the functionality.

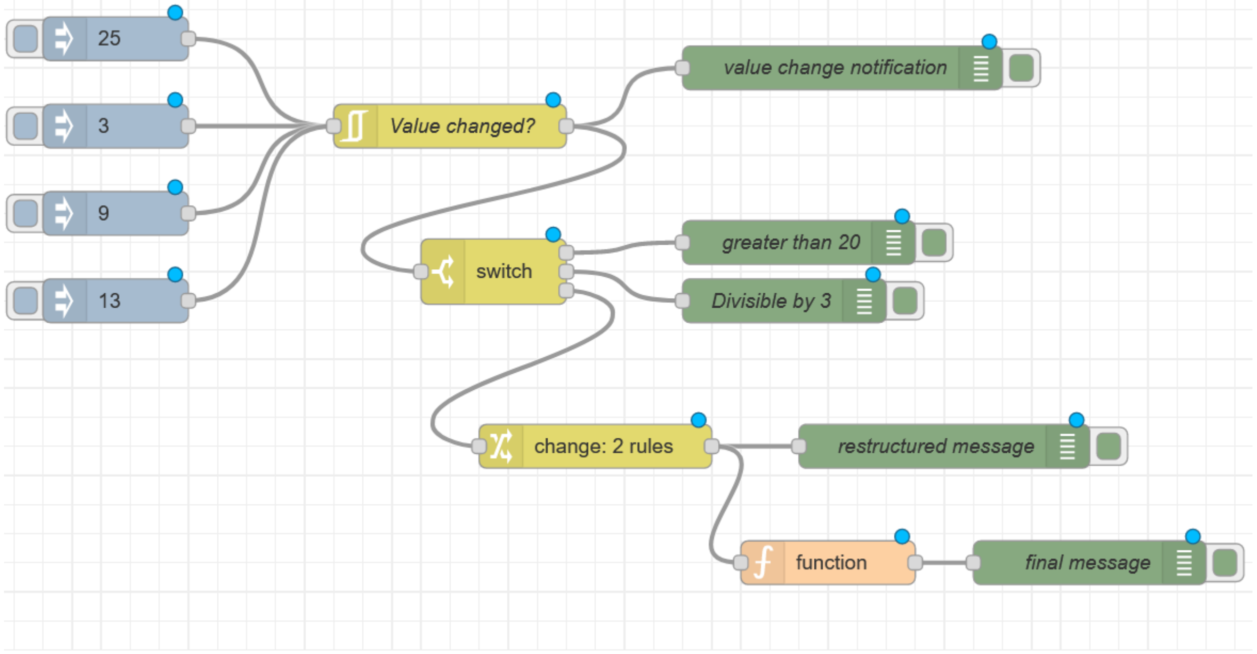


Figure 3: Completed Node-RED flow for Exercise 1

Exercise 2: Using APIs and processing API responses in Node-RED

Sunrise-Sunset API Documentation: https://sunrise-sunset.org/api

1. Create an *Inject* node capable of injecting a message with latitude, longitude, and date in JSON format

2. Use a *function* node to process the injected message, construct the request URL according to the requirements of the Sunrise-Sunset API and set the constructed request URL as a message property: *msg.url*

3. Use an *HTTP Request* node to and call the Sunrise-Sunset API using the request URL set in the message. (**Note:** Since we are passing the request URL as a message property, we must keep the *URL* field of the HTTP Request node empty)

4. Connect a *function* node after the HTTP request node to catch the response from the API and perform the following operations on the response;
   a) If the status property of the response is "OK" then delete the "Status" property of the response. Otherwise leave it as it is.
   b) Parse the value of the property "day_length", round it to the closest number of hours and append the rounded value to *msg.payload.results* as a new property named "*day_length_rounded".*
   (**Hint:** try out these JavaScript methods: *split, parseInt*)

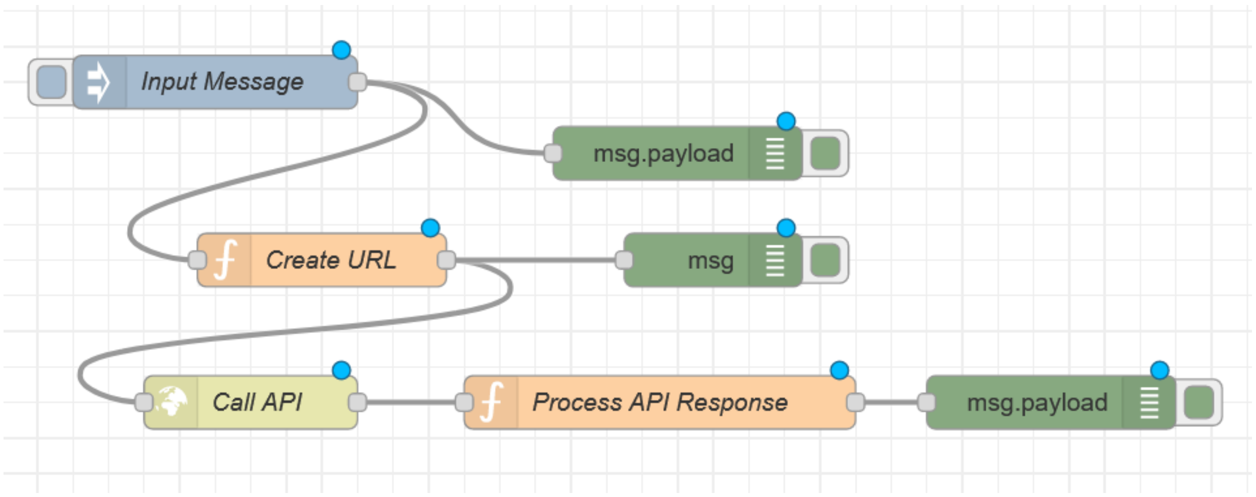


Figure 4: Completed Node-RED flow for Exercise 2