Department of Electronic & Telecommunications Engineering

University of Moratuwa

EN3270 Internet of Things Systems Engineering

Node-RED Session 1 - Exercises

2018 Batch _____ Semester 6

Exercise 01

Part 01: Familiarize with basics of Node-RED

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 node3: 9
   •Inject node 4: 13

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. 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,

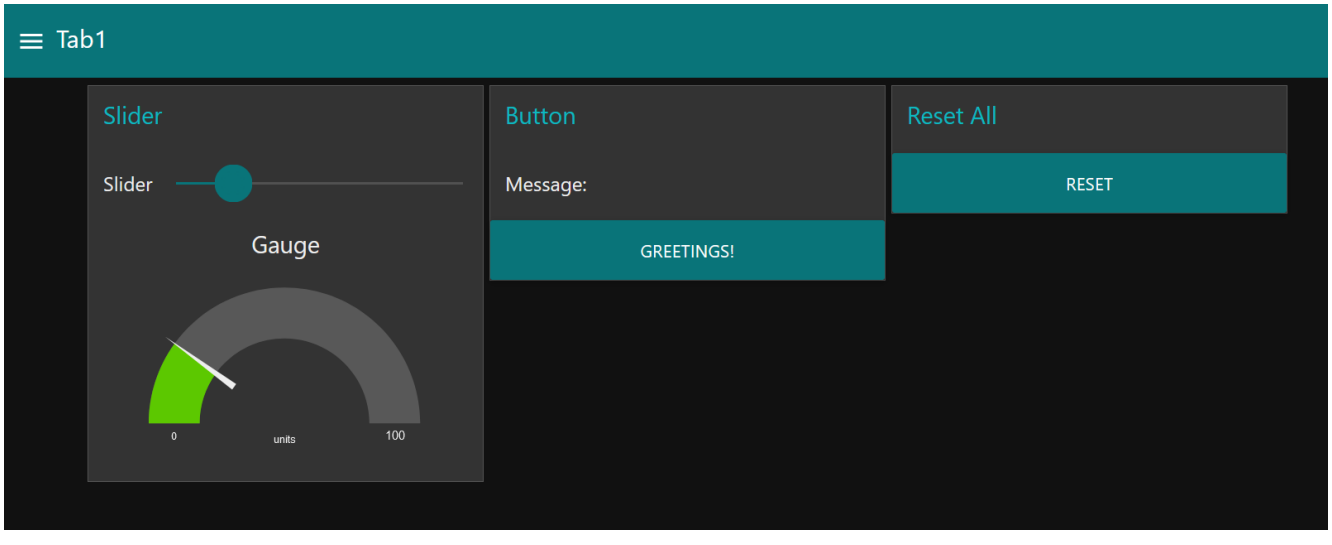
```
{
    "name": "<YOUR_NAME>",
    "index": "<YOUR_INDEX_NO>"
}
```

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. Continue to the part 02.

Part 02: User Interface Design



Tab 1:

- The ***Slider*** should produce data between 0 and 100. The ***Gauge*** should display the values as the *Slider* changes.
- When the **GREETINGS button** is pressed, the *Message* “Hello World!” should be displayed.
- When the **RESET button** is pressed, the *Slider* and the *Gauge* should return to zero and the *Message* should be cleared. An audio-visual indication “Reset button pressed” should be displayed, which can be cleared with an OK/Cancel button.

Tab 2 (Not shown):

- Should display a **chart** showing the slider data generated within the last 5 minutes.
- A ***Clear*** **button** should clear all data on the chart.Download the flow files (in JSON format) and save it as **<index_no>_session1_Ex1.json**

Exercise 02

Part 01: Acquiring and visualization of data from an API.

- Add a **drop-down List** and add **five different cities** in Sri Lanka (as String values) into the list as options.
- Add an **OpenWeatherMap API node** and,
    - Extract the property 'weather' and display the weather in a text box in the following format: **Weather< value in API response>**
    - Extract the property 'humidity' and display the humidity in a text box in the following format: **Humidity< value in API response>**
    - Extract the property 'tempc' and display the temperature (in **Celsius**) in a text box in the following format:
      Temperature :<value in API response> C
    - Convert the above temperature in Celsius to Fahrenheit and display them in a text box in the following format (hint: you may have to use *function*):
      Temperature :<value in API response> F
    - Extract the property 'pressure' (in **hPa**)
    - Convert the above temperature in **hPa to atm** and display in a text box in the following format:
      Pressure :<value in API response> atm
    - Show the *instantaneous temperature* (in Celsius) in the Gauge
        - Parameters for the Gauge:
          • Max value: 45
          • Min value: 15
    - Show the *instantaneous pressure* (in atm) in the Gauge
        - Parameters for the Gauge:
          • Max value: 2.5
          • Min value: 0.47
    - Use the chart (with a clear button) to show the last 10 temperature measurements (in Celsius).
    - Use the chart (with a clear button) to show the last 10 pressure measurements.
    - When the RESET button is pressed the Gauge should be set to its minimum value and all the text boxes should be cleared.

Part 02: The question is based on using and processing of responses from an API

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

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

- 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)
- Connect a function node after the **HTTP request node** to catch the response from the API and perform the following operations on the response;
    - If the status property of the response is “OK” then delete the “Status” property of the response. Otherwise leave it as it is.
    - 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**”.