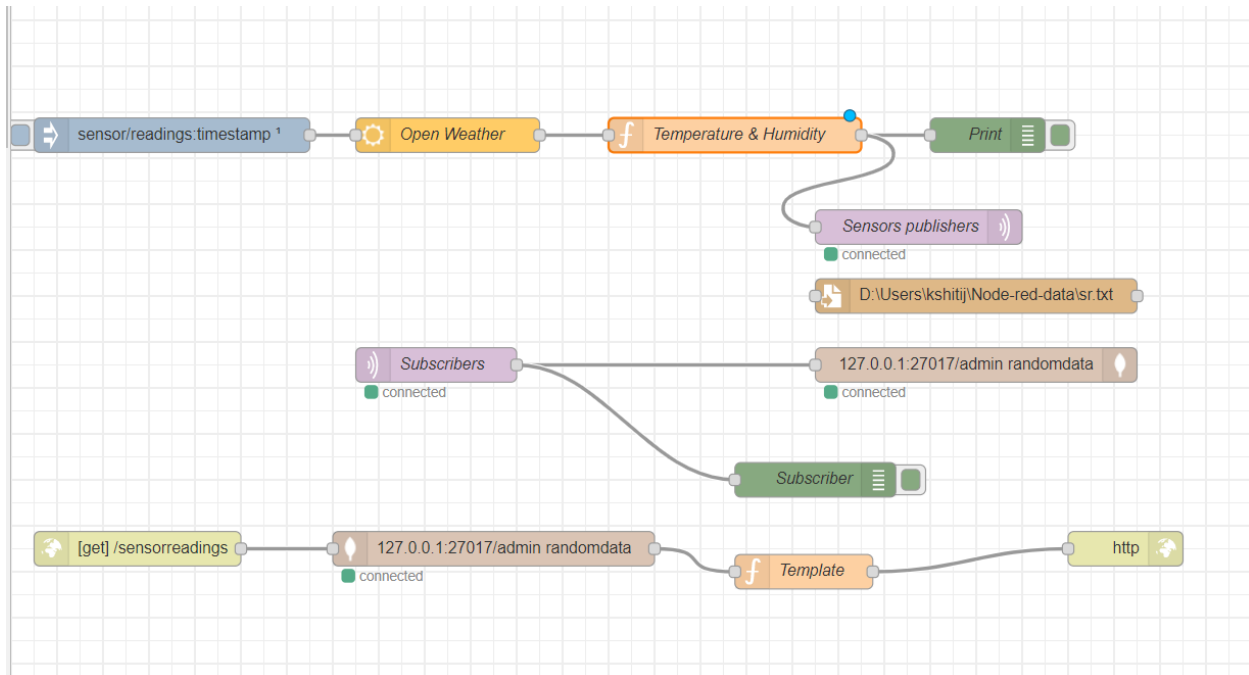


NodeRed based MQTT Client Server DB simulation

i) The simulation looks as follows.



ii) Here, Openweather API is used to fetch the readings for a given location.

The API key has to be fed which is provided when you create an account.

iii) The timestamp (inject node) is provided to get values at every interval. It is configured as shown below.

Edit inject node

Delete Cancel Done

Properties

Payload ▼ timestamp

Topic sensor/readings

☒ Inject once after 3 seconds, then

Repeat none ▼

Name Name

Note: "interval between times" and "at a specific time" will use cron.
"interval" should be 596 hours or less.
See info box for details.

Figure 2: Inject node

- iv) The topic given in inject node is sensor/readings from which the subscriber will fetch the readings of temperature and humidity.

- v) Next in the function node, we set the value of payload obtained from openweather API such that we obtain only the temperature and humidity values.

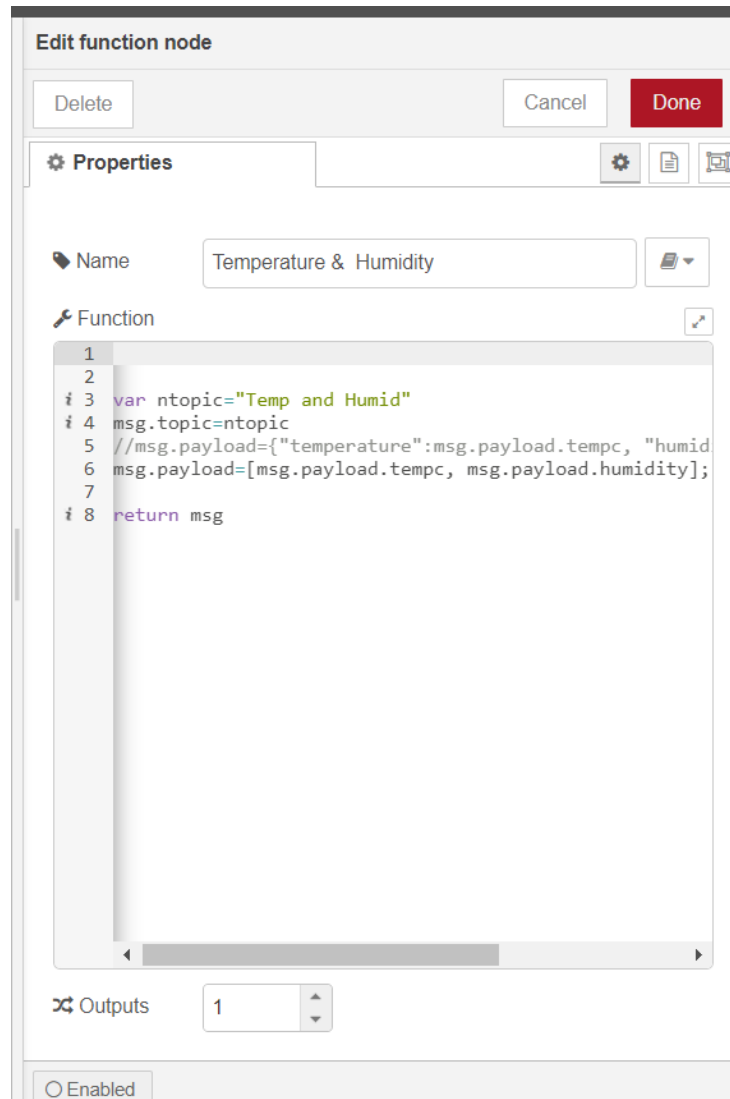





Figure 3: Function node

- vi) The publishers are configured as shown below in figure 4 and 5.


Edit mqtt out node > **Edit mqtt-broker node**

Delete Cancel Update


Properties  


 Name

Connection Security Messages

 Server Port

☐ Enable secure (SSL/TLS) connection

 Client ID

 Keep alive time (s) ☒ Use clean session

☒ Use legacy MQTT 3.1 support


☐ Enabled  2 nodes use this config ▼

Figure 4: MQTT-broker node

The screenshot shows a configuration window titled "Edit mqtt out node > Edit mqtt-broker node". At the top, there are three buttons: "Delete", "Cancel", and "Update". Below these is a tabbed interface with a "Properties" tab selected. The "Properties" tab contains a "Name" field with the placeholder text "Name". Below this is a tabbed interface with three tabs: "Connection", "Security", and "Messages". The "Connection" tab is active and contains the following settings: "Server" set to "localhost", "Port" set to "1883", an unchecked checkbox for "Enable secure (SSL/TLS) connection", "Client ID" set to "Leave blank for auto generated", "Keep alive time (s)" set to "60", a checked checkbox for "Use clean session", and a checked checkbox for "Use legacy MQTT 3.1 support".

Figure 5: MQTT-out node (publisher)

- vii) We made use of Mosquitto broker running on local machine on default port 1883.
- viii) Similarly, the subscribers were configured as shown below.

Edit mqtt in node

Delete Cancel Done

Properties

Server localhost:1883

Topic sensor/readings

QoS 2

Output auto-detect (string or buffer)

Name Subscribers

☐ Enabled

Figure 6: MQTT in node

- ix) The subscribers were now subscribed to topic called as sensors/readings and the publishers published the data (temperature and humidity) and was available to all the users subscribed to the topic.
- x) The data at the subscriber's side was also stored on a mongoDB server. The database name in this experiment was "admin" and collection name was "randomdata". The mongoDB node configurations are shown in figure 7 and 8.

Edit mongodb out node

Delete Cancel Done

Properties

Server 127.0.0.1:27017/admin

Collection randomdata

Operation insert

☒ Only store msg.payload object

Name Name

☐ Enabled

Figure 7: MongoDB out node(1)

Edit mongodb out node > Edit mongodb node

Delete Cancel Update

Properties

Host 127.0.0.1 Port 2701

Database admin

Username

Password

Name Name




Enabled 2 nodes use this config On all flows


Figure 8: MongoDB out node(2)


- xi) The username and password fields were left blank as no authorization credentials was given for database admin.
- xii) HTTP endpoints like input node (GET request) and response node were used.
- xiii) The get input node was given an URL where a user could see the temperature and humidity readings which were actually fetched from mongodb database. The response node is responsible for sending back the responses to requests received from HTTP input node. The configurations for input and response displayed back to user at the URL “http://127.0.0.1:1880/sensorreadings” are shown in figure 9 and 10 respectively.


Edit http in node

Delete Cancel Done

Properties   

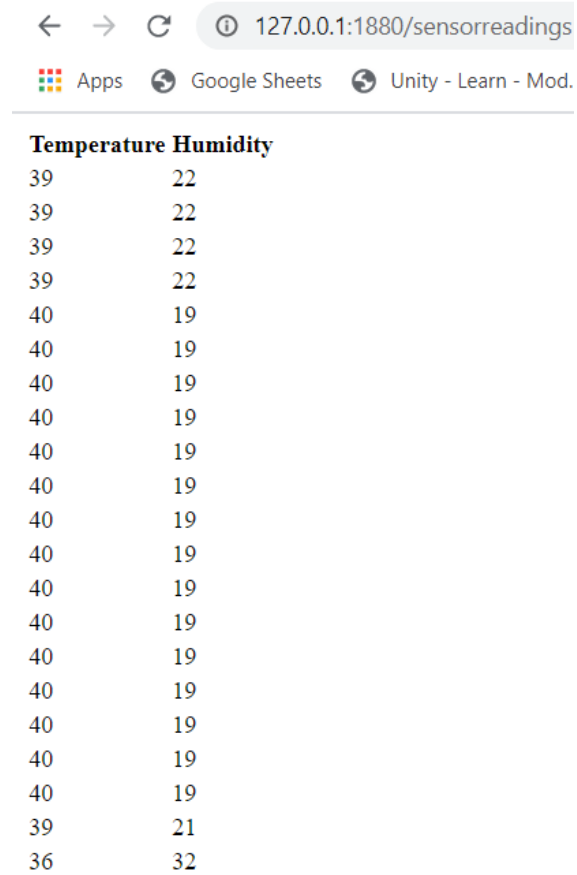
 Method GET ▼

 URL /sensorreadings

 Name Name

☐ Enabled

Figure 9: HTTP input node



← → ↻ ⓘ 127.0.0.1:1880/sensorreadings

Apps Google Sheets Unity - Learn - Mod.

Temperature	Humidity
39	22
39	22
39	22
39	22
40	19
40	19
40	19
40	19
40	19
40	19
40	19
40	19
40	19
40	19
40	19
40	19
40	19
40	19
40	19
39	21
36	32

Figure 10: Client request to the server