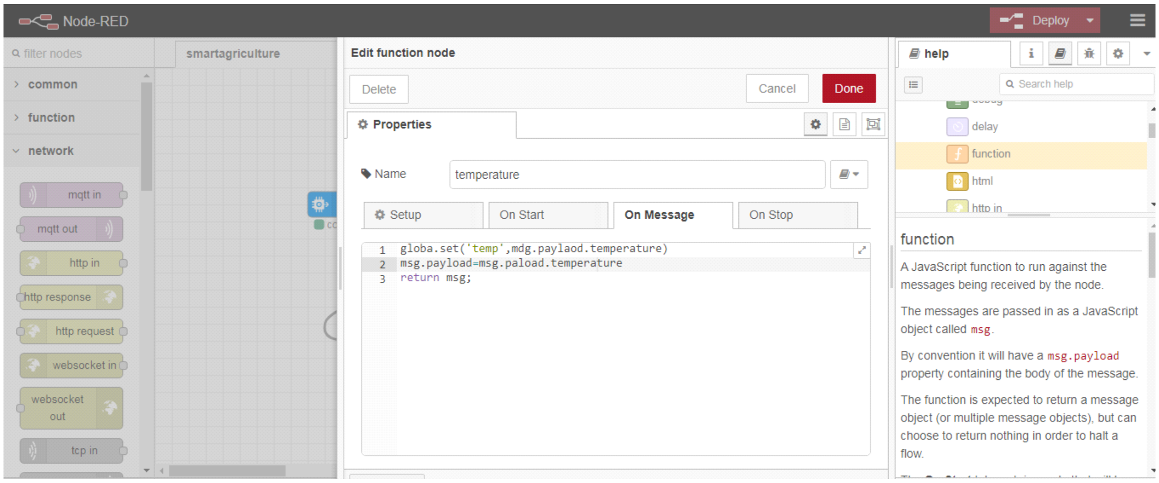
IOT ENABLED SMART FARMING APPLICATION.

Build A Web Application Using Node-RED

TEAM ID : PNT2022TMID04037

Configuration of Node-Red to collect IBM cloud data

The node IBM IOT App In is added to Node-Red workflow. Then the appropriate device credentials obtained earlier are entered into the node to connectand fetch device telemetry to Node-Red



Edit function node

Delete | Cancel | Done

Properties

Name: temperature

Setup | On Start | On Message | On Stop

```
globa.set('temp',mdg.paylaod.temperature)
msg.payload=msg.paload.temperature
return msg;
```

function

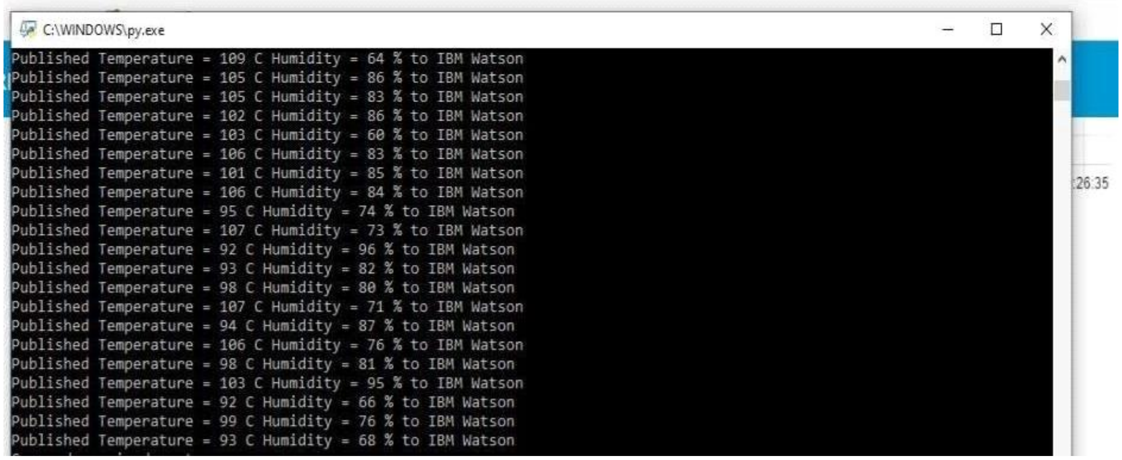
A JavaScript function to run against the messages being received by the node.

The messages are passed in as a JavaScript object called msg.

By convention it will have a msg.payload property containing the body of the message.

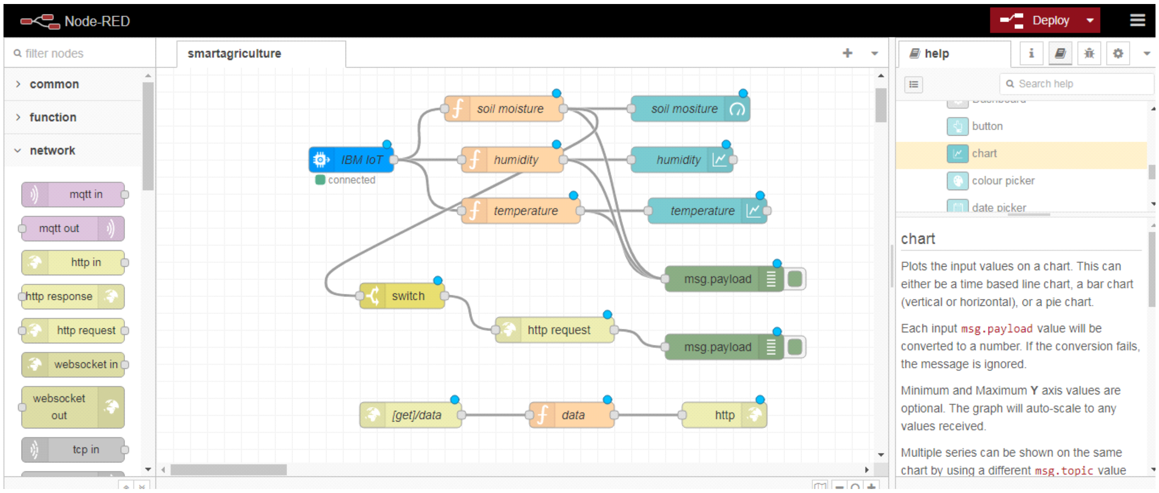
The function is expected to return a message object (or multiple message objects), but can choose to return nothing in order to halt a flow.

- Once it is connected Node-Red receives data from the device
- Display the data using debug node for verification
- Connect function node and write the Java script code to get each readingseparately.
- The Java script code for the function node is:msg.payload=msg.payload.d.temperature return msg;
- Finally connect Gauge nodes from dashboard to see the data in UI



```
Published Temperature = 109 C Humidity = 64 % to IBM Watson
Published Temperature = 105 C Humidity = 86 % to IBM Watson
Published Temperature = 105 C Humidity = 83 % to IBM Watson
Published Temperature = 102 C Humidity = 86 % to IBM Watson
Published Temperature = 103 C Humidity = 60 % to IBM Watson
Published Temperature = 106 C Humidity = 83 % to IBM Watson
Published Temperature = 101 C Humidity = 85 % to IBM Watson
Published Temperature = 106 C Humidity = 84 % to IBM Watson
Published Temperature = 95 C Humidity = 74 % to IBM Watson
Published Temperature = 107 C Humidity = 73 % to IBM Watson
Published Temperature = 92 C Humidity = 96 % to IBM Watson
Published Temperature = 93 C Humidity = 82 % to IBM Watson
Published Temperature = 98 C Humidity = 80 % to IBM Watson
Published Temperature = 107 C Humidity = 71 % to IBM Watson
Published Temperature = 94 C Humidity = 87 % to IBM Watson
Published Temperature = 106 C Humidity = 76 % to IBM Watson
Published Temperature = 98 C Humidity = 81 % to IBM Watson
Published Temperature = 103 C Humidity = 95 % to IBM Watson
Published Temperature = 92 C Humidity = 66 % to IBM Watson
Published Temperature = 99 C Humidity = 76 % to IBM Watson
Published Temperature = 93 C Humidity = 68 % to IBM Watson
```

Data received from the cloud in Node-Red console



chart

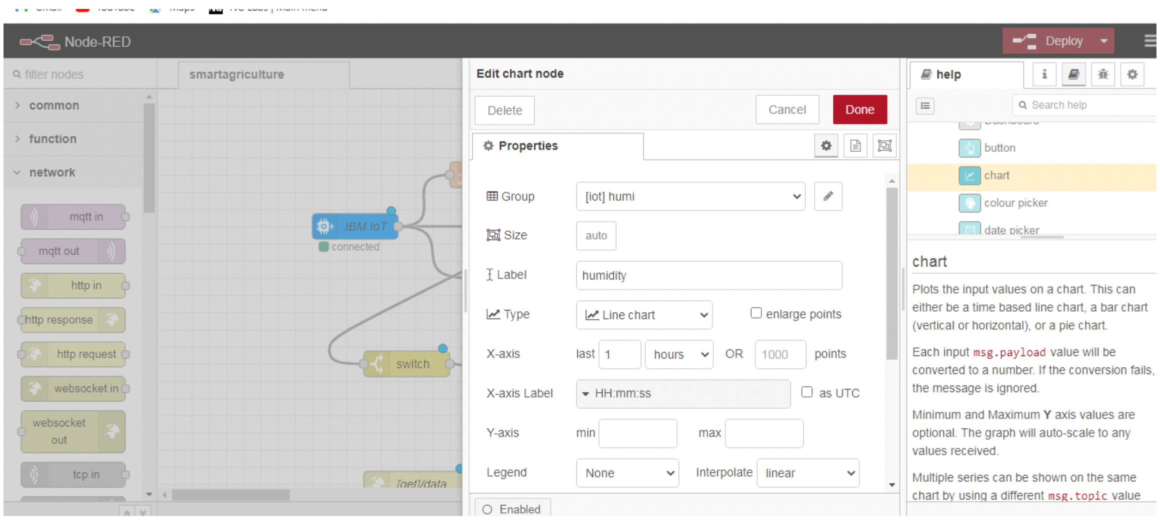
Plots the input values on a chart. This can either be a time based line chart, a bar chart (vertical or horizontal), or a pie chart.

Each input `msg.payload` value will be converted to a number. If the conversion fails, the message is ignored.

Minimum and Maximum **Y** axis values are optional. The graph will auto-scale to any values received.

Multiple series can be shown on the same chart by using a different `msg.topic` value

Nodes connected in following manner to get each reading separately



Group [iot] humi

Size auto

Label humidity

Type Line chart □ enlarge points

X-axis last 1 hours OR 1000 points

X-axis Label HH:mm:ss □ as UTC

Y-axis min max

Legend None Interpolate linear

○ Enabled

chart

Plots the input values on a chart. This can either be a time based line chart, a bar chart (vertical or horizontal), or a pie chart.

Each input `msg.payload` value will be converted to a number. If the conversion fails, the message is ignored.

Minimum and Maximum **Y** axis values are optional. The graph will auto-scale to any values received.

Multiple series can be shown on the same chart by using a different `msg.topic` value

This is the Java script code I written for the function node to get Temperature separately.

Configuration of Node-Red to collect data from Open Weather

- The Node-Red also receive data from the Open Weather API by HTTP GET request.An inject trigger is added to perform HTTP request for every certain interval.
- HTTP request node is configured with URL
- Thedata we receive from Open Weather after request is in belowJSON

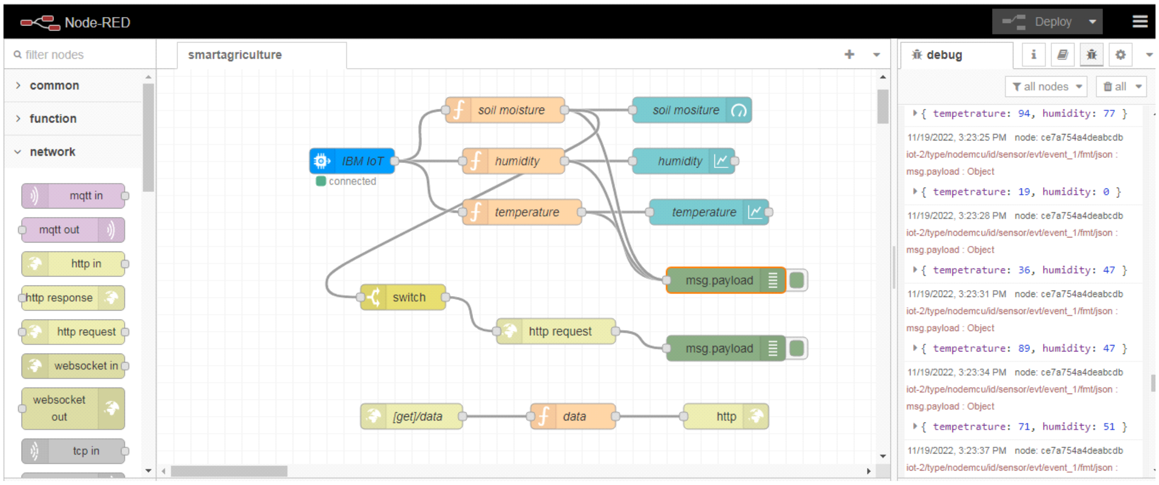
format:{"coord":{"lon":79.85,"lat":14.13},"weather":[{"id":803,"main":"Clouds","description":"brokenclouds","icon":"04n"}],"base":"stations","main":{"temp":30759,"feels_like":305.5,"temp_min": 307.59,"temp_max":307.59,"pressure":1002,"humidity":35,"sea_level":1002,"grnd_level":1000},"wind":{"speed":6.23,"deg":170},"clouds":{"all":68},"dt":1589991979,"sys":{"country":"IN","sunrise":1589933553,"sunset":1589979720},"timezone":19800,"id":1270791,"name": "Gūdūr","cod":20 0}

In order to parse the JSON string we use Java script functions and geteach parameters

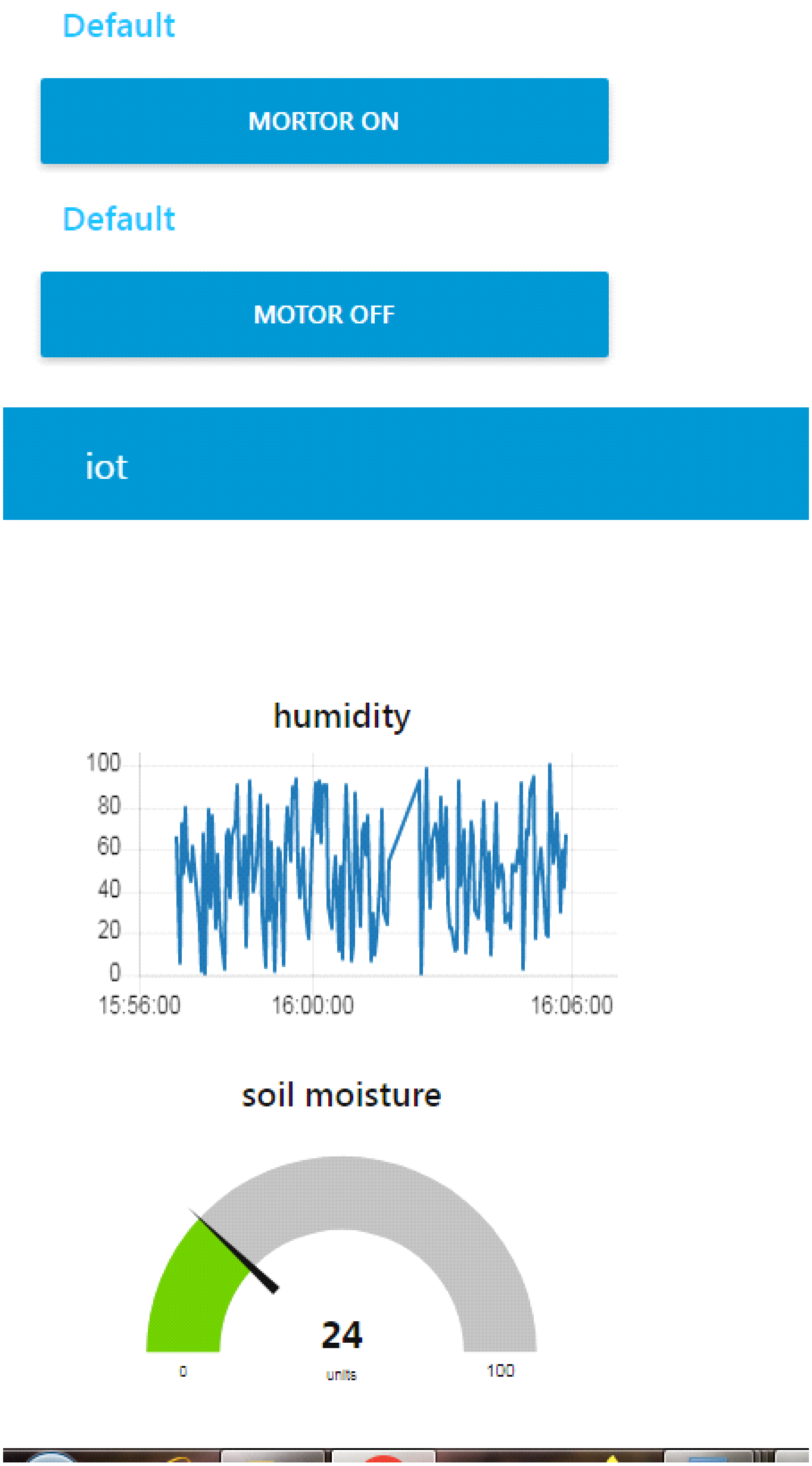
```
var temperature =
 msg.payload.main.temp;
 temperature = temperature-
 273.15;
return {payload : temperature.toFixed(2)};
```

In the above Java script code we take temperature parameter into anew variable and convert it from kelvin to Celsius

Then we add Gauge and text nodes to represent data visually in UI.



```
▸{ tempetrature: 94, humidity: 77 }
11/19/2022, 3:23:25 PM  node: ce7a754a4deabcdb
iot-2/type/nodemcu/id/sensor/evt/event_1/fmt/json :
msg.payload : Object
▸{ tempetrature: 19, humidity: 0 }
11/19/2022, 3:23:28 PM  node: ce7a754a4deabcdb
iot-2/type/nodemcu/id/sensor/evt/event_1/fmt/json :
msg.payload : Object
▸{ tempetrature: 36, humidity: 47 }
11/19/2022, 3:23:31 PM  node: ce7a754a4deabcdb
iot-2/type/nodemcu/id/sensor/evt/event_1/fmt/json :
msg.payload : Object
▸{ tempetrature: 89, humidity: 47 }
11/19/2022, 3:23:34 PM  node: ce7a754a4deabcdb
iot-2/type/nodemcu/id/sensor/evt/event_1/fmt/json :
msg.payload : Object
▸{ tempetrature: 71, humidity: 51 }
11/19/2022, 3:23:37 PM  node: ce7a754a4deabcdb
iot-2/type/nodemcu/id/sensor/evt/event_1/fmt/json :
```


Default
MORTOR ON
Default
MOTOR OFF
iot
humidity
100
80
60
40
20
0
15:56:00
16:00:00
16:06:00
soil moisture
24
units
0
100