VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING

B.TECH. ECE/ECM

ECE3502 – IoT DOMAIN ANALYST

LAB MANUAL

List of Experiments

Experiment 1

Display Light Intensity using Node-Red and MQTT

Aim: Using appropriate nodes in Node Red measure the light intensity in the room and output data to the web API. Display the output in Debug Monitor, UI and send the data through MQTT and display the output in HiveMQ.

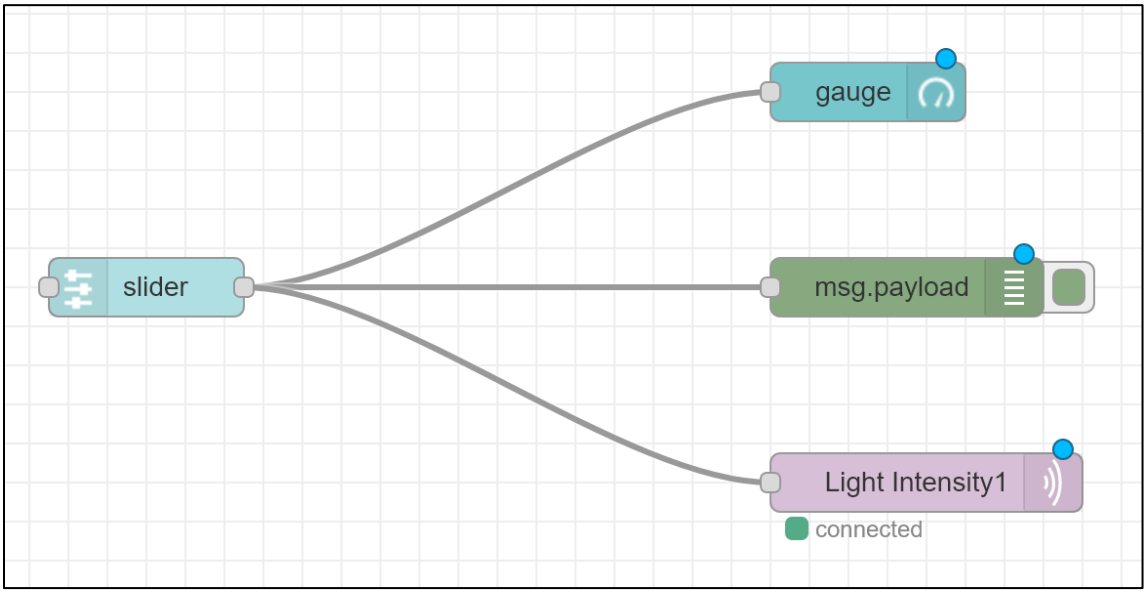
Theory:

- MQTT (Message Queuing Telemetry Transport) is an open OASIS and ISO standard (ISO/IEC 20922) lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP; however, any network protocol that provides ordered, lossless, b i-directional connections can support MQTT. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited.
- MQTT is a publish-subscribe-based messaging protocol used in Internet of Things. The goal is to provide a protocol, which is bandwidth-efficient and uses little battery power.

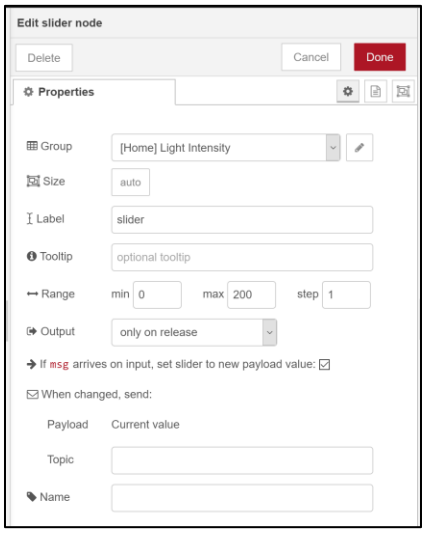
Procedure:

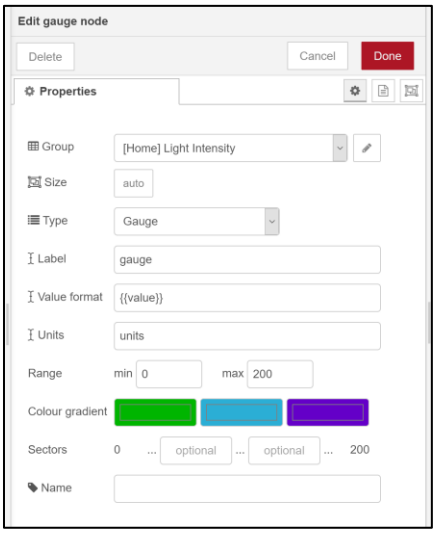
- Trigger node red in cmd using node-red -v
- http://127.0.0.1:1880/
- Arrange nodes according to the flow and assign properties
- Open HiveMQ and get the output
- http://www.hivemq.com/demos/websocket-client/

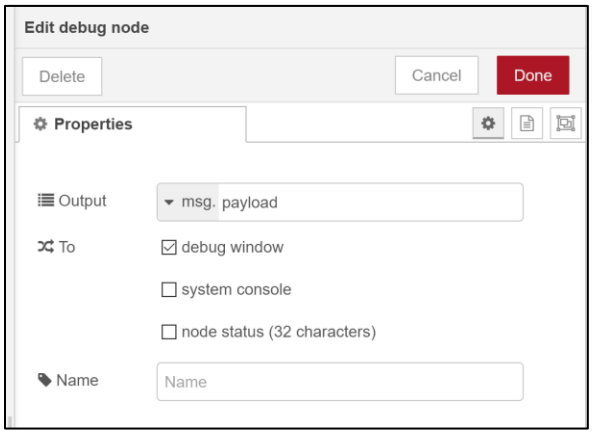
Flow:

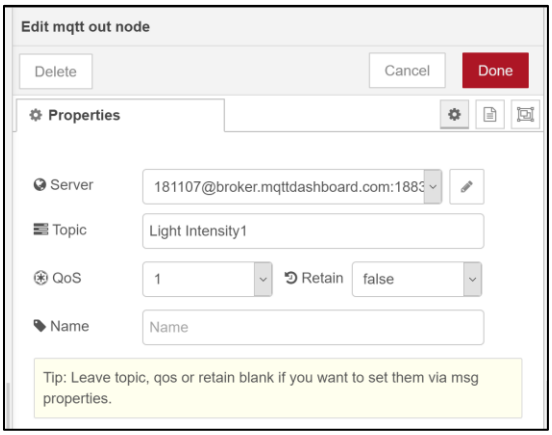


Properties:



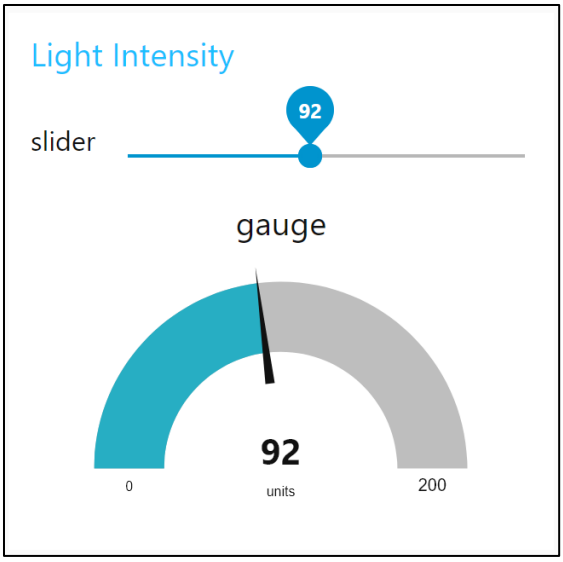




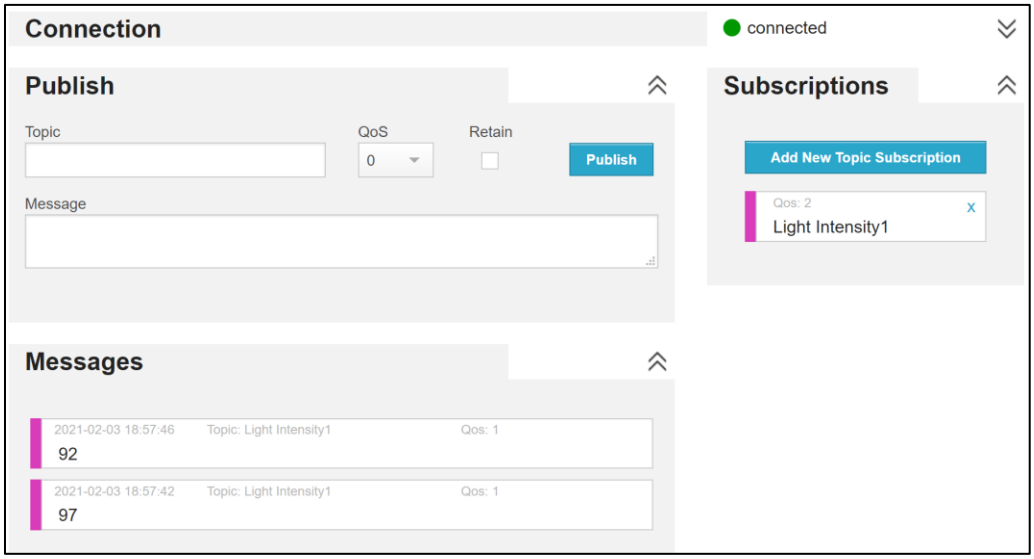


Output:

Dashboard:



MQTT:



Inference: A slider node is used to control the light intensity by the user. This is measured and published to the MQTT server using MQTT out node

Result: Hence, Light Intensity has been measured, displayed and published using MQTT and Dashboard

Experiment 2

Web Based Application (HTML) Form Creation & Submission in Node-Red

Aim: To create a web-based application using form creation and submission with HTML in node-red

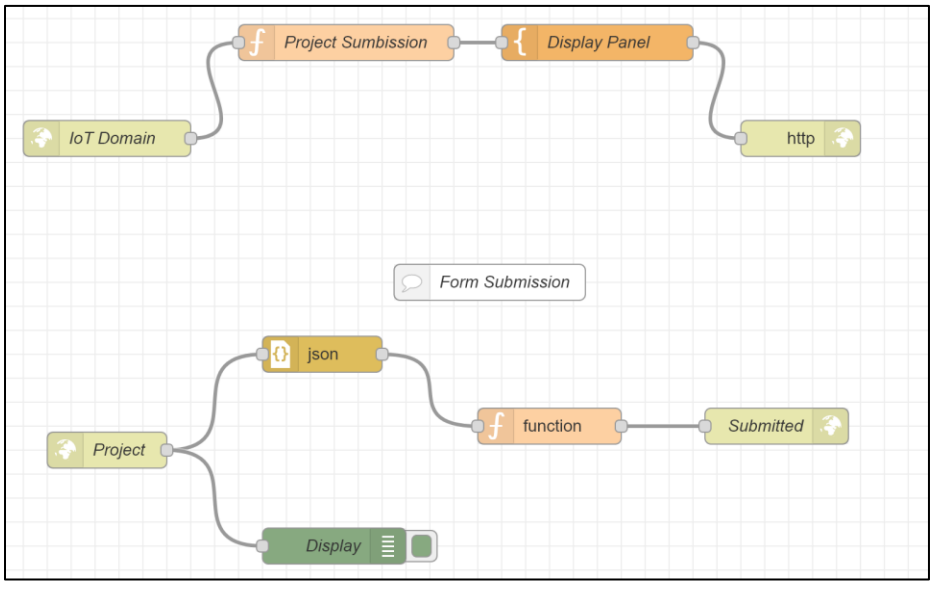
Theory:

- Web API: Web API as the name suggests, is an API over the web which can be accessed using HTTP protocol.
- Mustache: Mustache is a simple web template system. Mustache is described as a logic-less template engine because it does not have any explicit control flow statements, such as if and else conditionals or for loops. Looping and conditional evaluation can be achieved using section tags processing lists and lambdas.

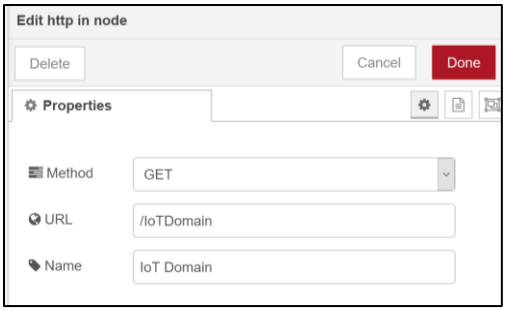
Procedure:

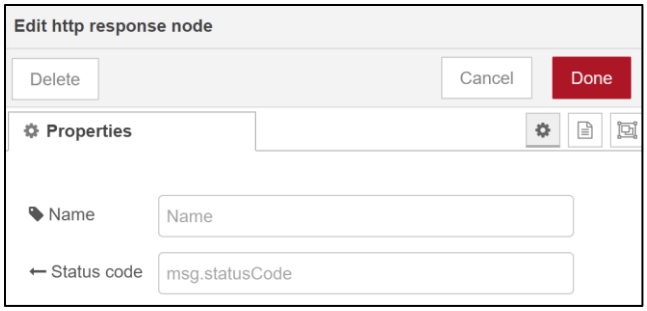
- Trigger node red in cmd using node-red -v
- Arrange nodes as shown in the flow and configure properties
- Run the web app in the browser and insert values in the form

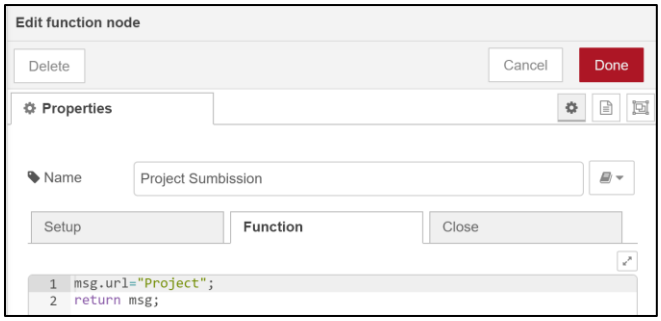
Flow:

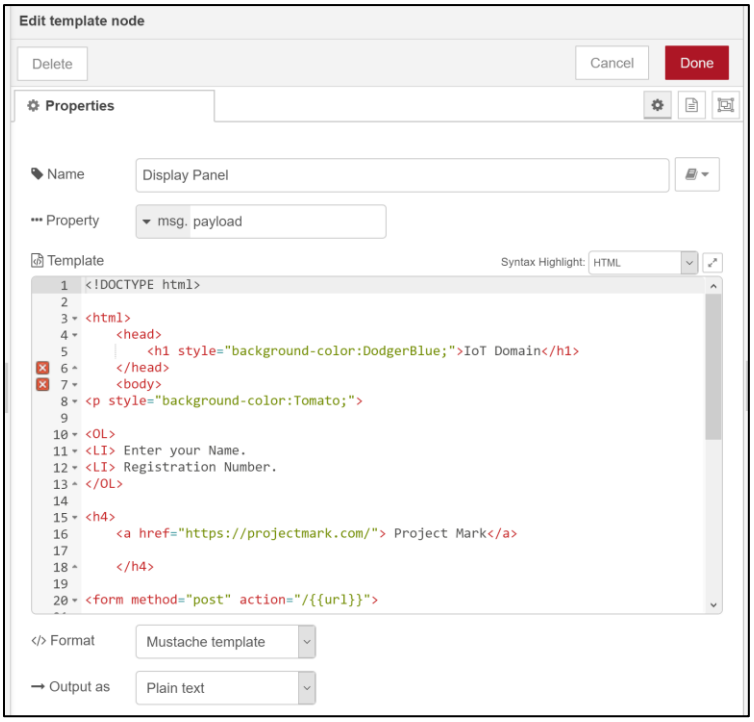


Properties:







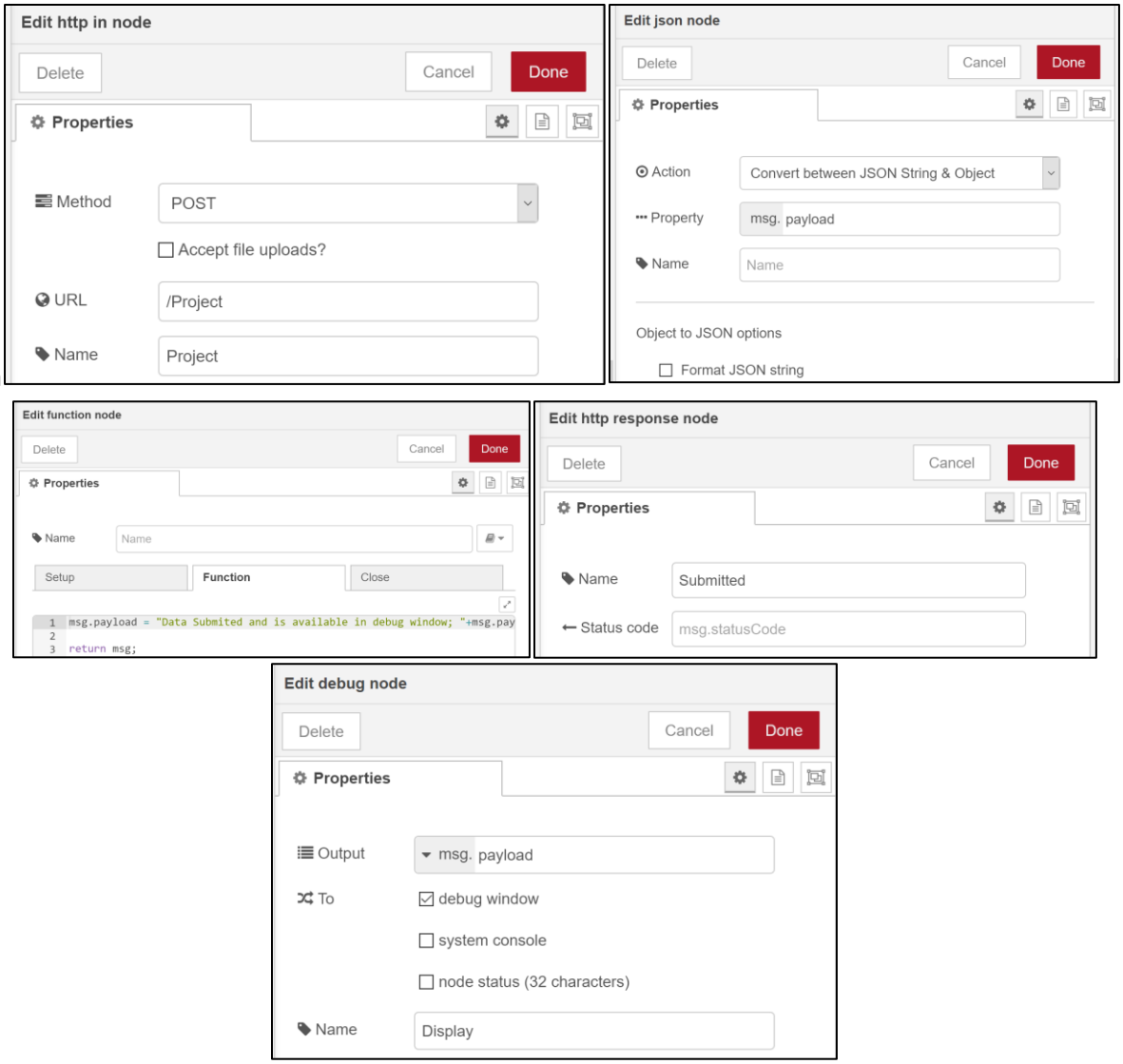


HTML Code:

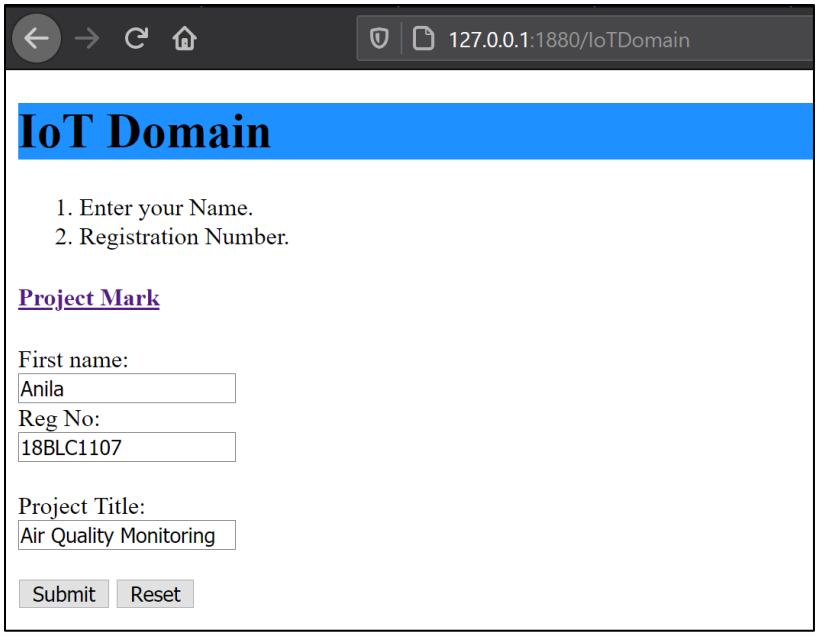
```
<!DOCTYPE html>

<html>
    <head>
        <h1 style="background-color:DodgerBlue;">IoT Domain</h1>
    </head>
    <body>
```

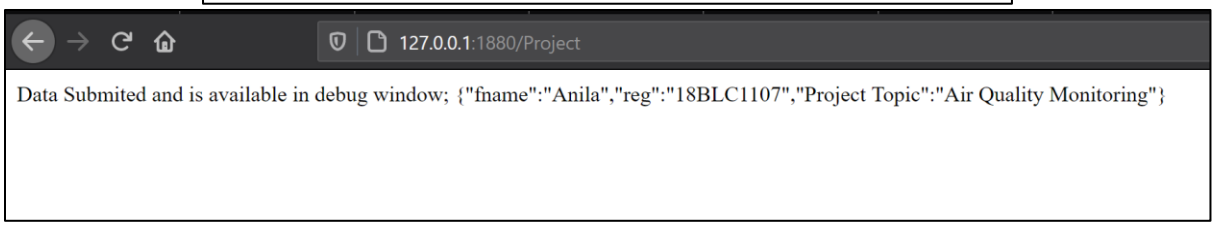
```
<p style="background-color:Tomato;">

<OL>
<LI> Enter your Name.
<LI> Registration Number.
</OL>
<h4>
    <a href="https://projectmark.com/"> Project Mark</a>

    </h4>
<form method="post" action="/{{url}}">
  <label for="name">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>

  <label for="reg">Reg No:</label><br>
  <input type="text" id="reg" name="reg" ><br><br>

  <label for="topic">Project Title:</label><br>
  <input type="text" id="topic" name="Project Topic" ><br><br>

  <input type="submit" value="Submit">

  <input type="reset" value="Reset" >
</form>

</body>
</html>
```

a



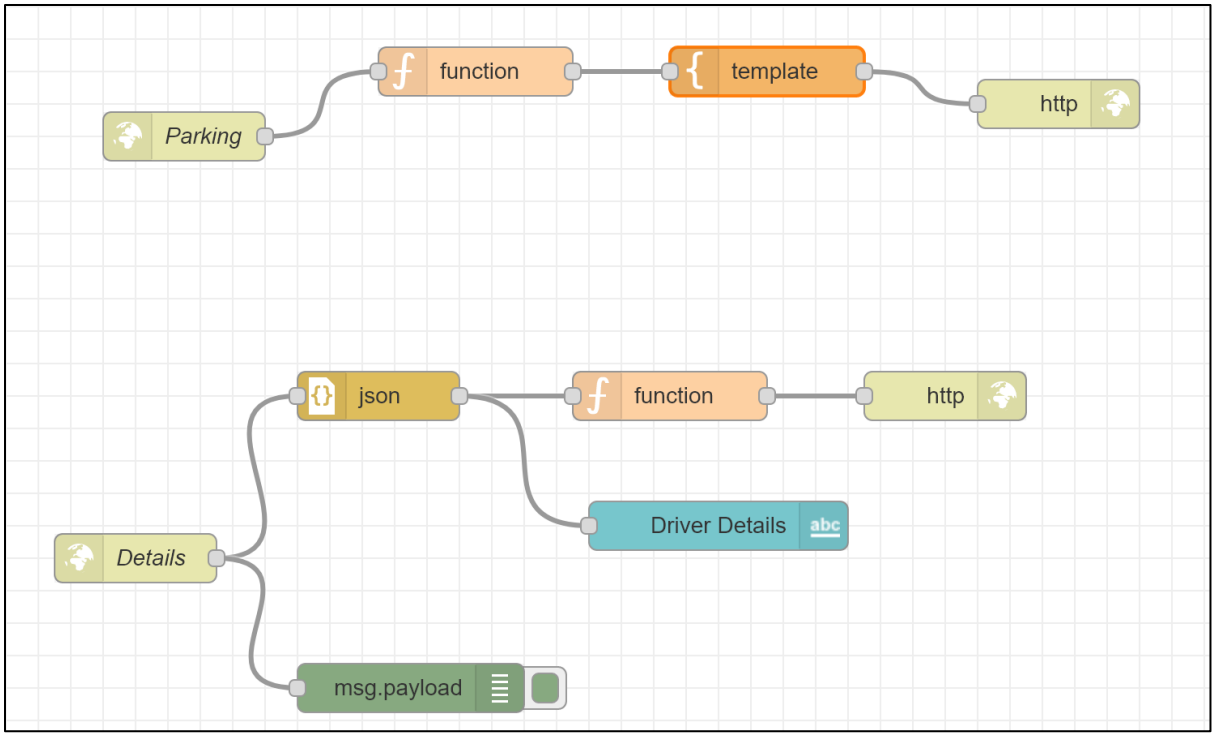
Web Application (Form)





Task: Using Node-Red build a web-based application to automate a smart parking system in a metro station. Create a form through HTML and CSS to get car owner details to automate the parking system. Display the received details in the debug monitor and UI. Create a hyperlink to show the parking rates of the metro station

Flow:



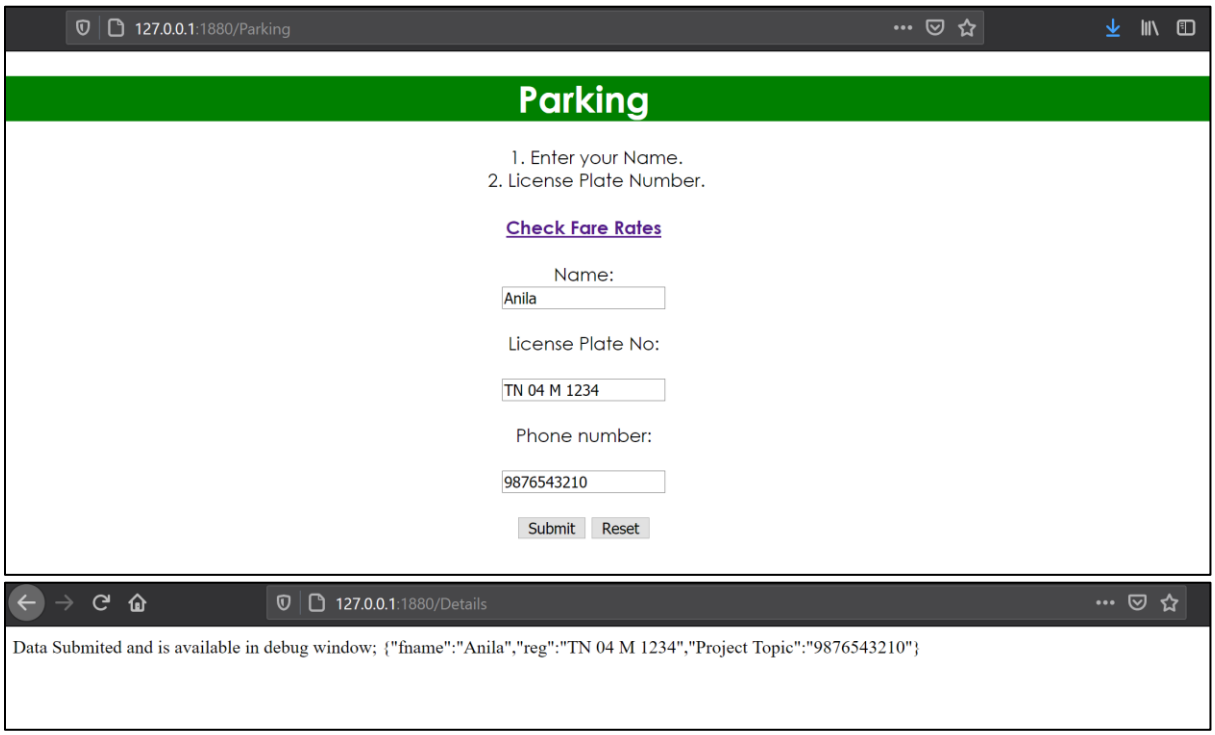
HTML Code:

```
<!DOCTYPE html>

<html>
    <head>
        <h1 style="background-color:Green;text-align:center;font-family:century
gothic;color:white">Parking</h1>

    <style>
    h4 {text-align: center; font-family:century gothic}
    form {text-align: center; font-family:century gothic}
    body {text-align: center; font-family:century gothic}

</style>
    </head>
    <body>
<p style="background-color:Tomato;">

<OL>
<LI> Enter your Name.
<LI> License Plate Number.
</OL>

<h4>
    <a href="https://chennaimetrorail.org/wp-content/uploads/2018/05/Parking-Tariff-
stations.pdf"> Check Fare Rates</a>

    </h4>
```

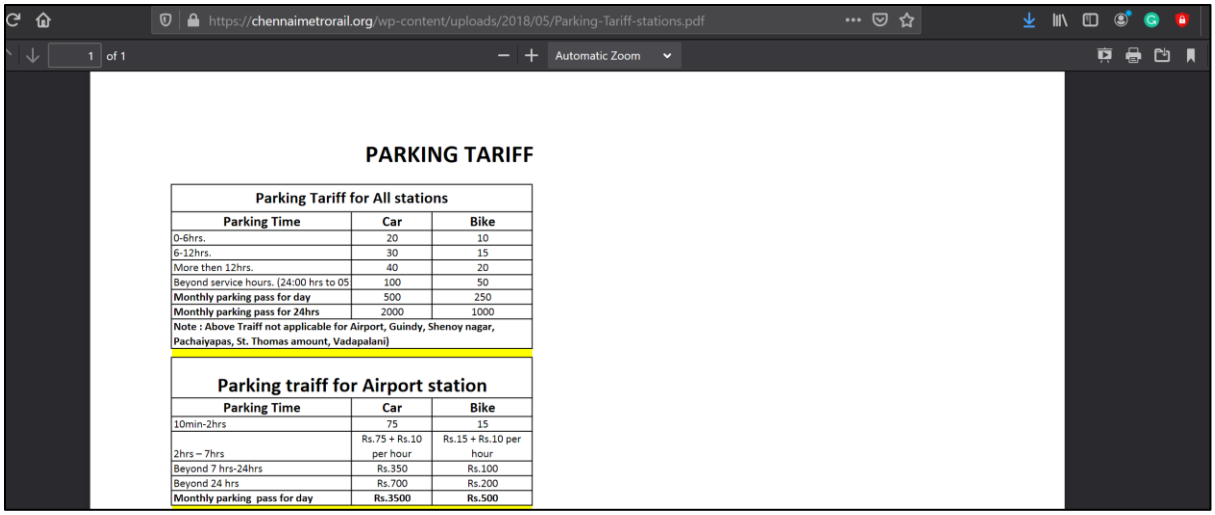
```
<form method="post" action="/{{url}}">

  <label for="name">Name:</label><br>
  <input type="text" id="fname" name="fname"><br><br>

  <label for="reg">License Plate No:</label><br><br>
  <input type="text" id="reg" name="reg" ><br><br>

  <label for="topic">Phone number:</label><br><br>
  <input type="text" id="topic" name="Project Topic" ><br><br>

  <input type="submit" value="Submit">

  <input type="reset" value="Reset" >

</form>

</body>
</html>
```

Output:

Form

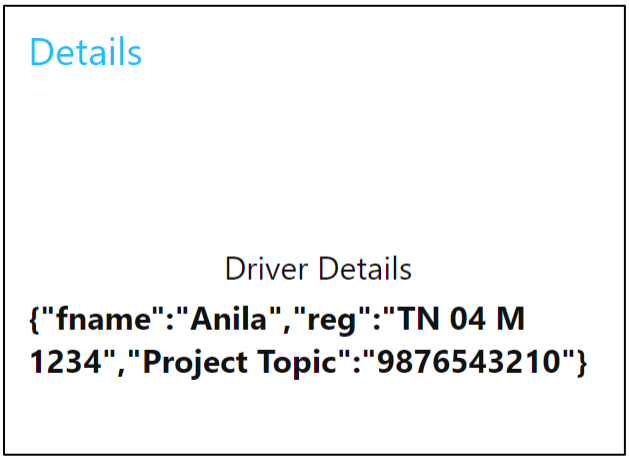


Hyperlink



Debug Window

Dashboard



Inference: HTTP in node has been used to do GET and POST operations. The template node is used to write the HTML code for the form submission and provide hyperlinks. The input information is then posted in the debug and dashboard windows

Result: Thus, we have performed and observed a Web Based Application (HTML) Form Creation and Submission for a Project Title Submission and a Smart Parking System (Task) in Node-Red and have successfully obtained all the outputs f or the same.

Experiment 3

Online Media Monitoring Using Node-Red | Feedparser Node

Aim: To monitor online news feeds and display headlines using Feedparser node in Node-Red

Theory:

RSS (RDF Site Summary or Really Simple Syndication)

- It refers to files easily read by a computer called XML files that automatically update information. An RSS feed takes the headlines, summaries, and update notices, and then links back to articles on the website's page.
- RSS is a web feed that allows users and applications to access updates to websites in a standardized, computer-readable format.

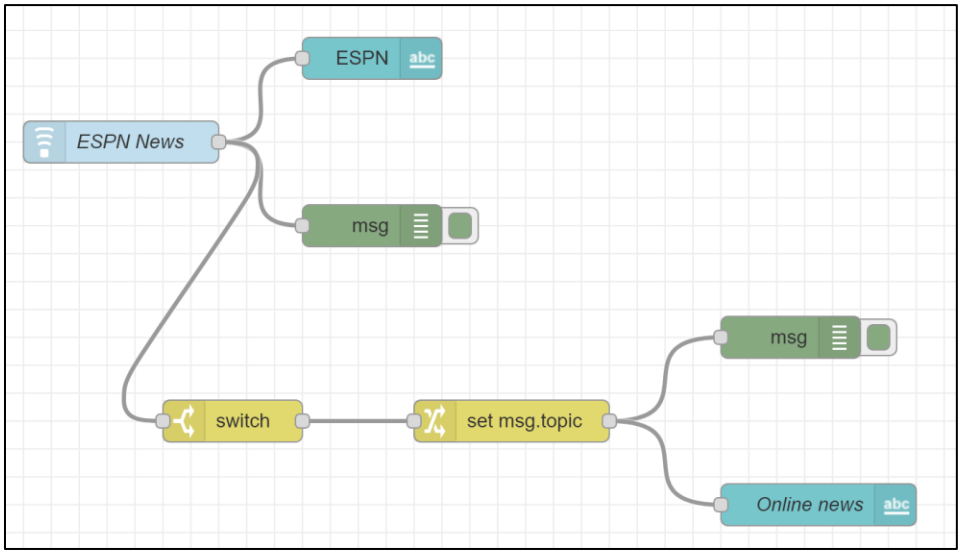
RSS Feed:

- In an RSS feed, these updates and notifications are gathered, organized and updated in real-time into one convenient dashboard.
- his information is fetched by a user's RSS feed reader that converts the files and the latest updates from websites into an easy-to-read format.

Procedure:

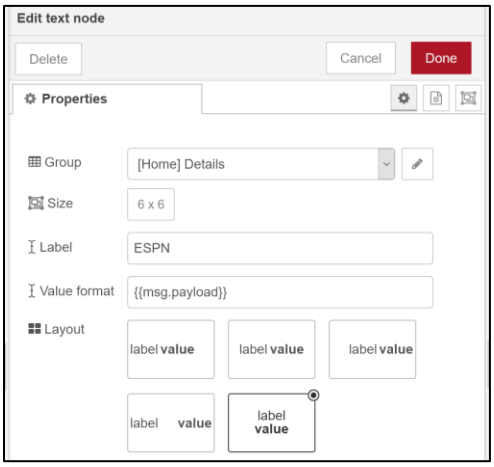
- Trigger node-red in cmd using node-red -v
- Connect the flow and configure their properties as shown
- Copy the RSS feed link to the Feedparser node to monitor news feeds
- Deploy onto dashboard and debug window

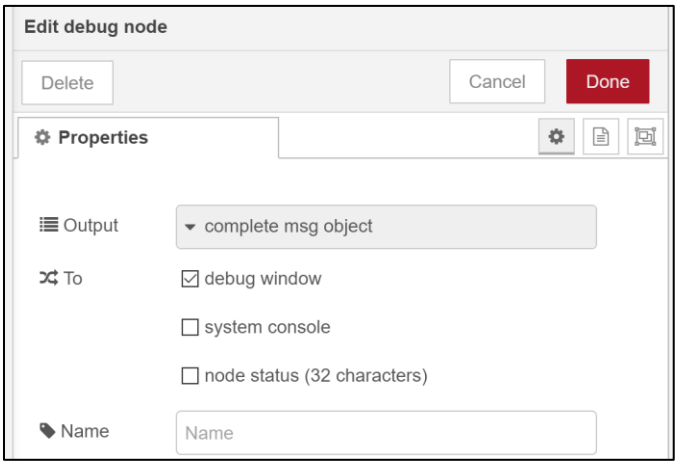
Flow:



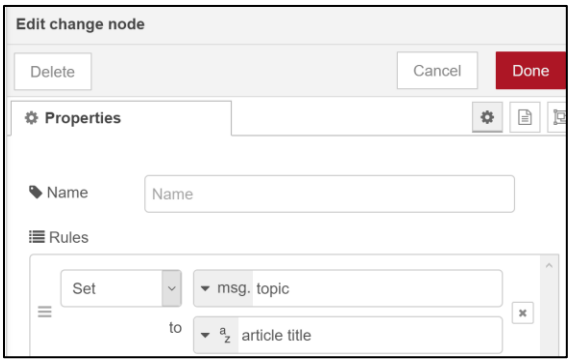
Properties:

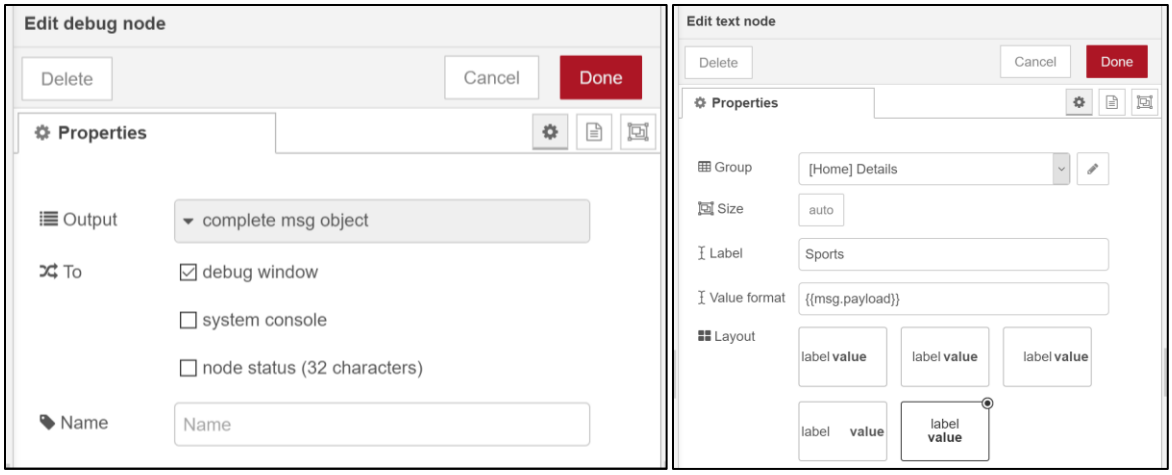

Edit feedparser node
Delete
Cancel
Done
Properties
Feed url
https://www.espn.com/espn/rss/news
Refresh
15
minutes
Name
ESPN News


Edit text node
Delete
Cancel
Done
Properties
Group
[Home] Details
Size
6 x 6
Label
ESPN
Value format
{{msg.payload}}
Layout
label value
label value
label value
label value
label value


Edit debug node
Delete
Cancel
Done
Properties
Output
complete msg object
To
debug window
system console
node status (32 characters)
Name
Name


Edit switch node
Delete
Cancel
Done
Properties
Name
Name
Property
msg.
article.title
matches regex
prince
→ 1
ignore case

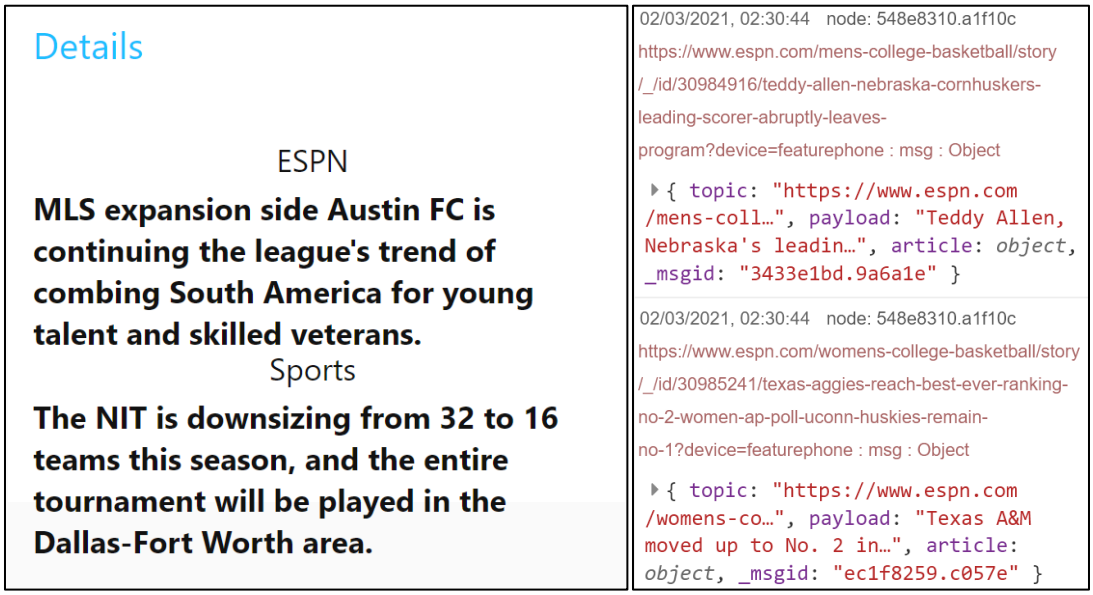

Edit change node
Delete
Cancel
Done
Properties
Name
Name
Rules
Set
msg.
topic
to
article title



Output:

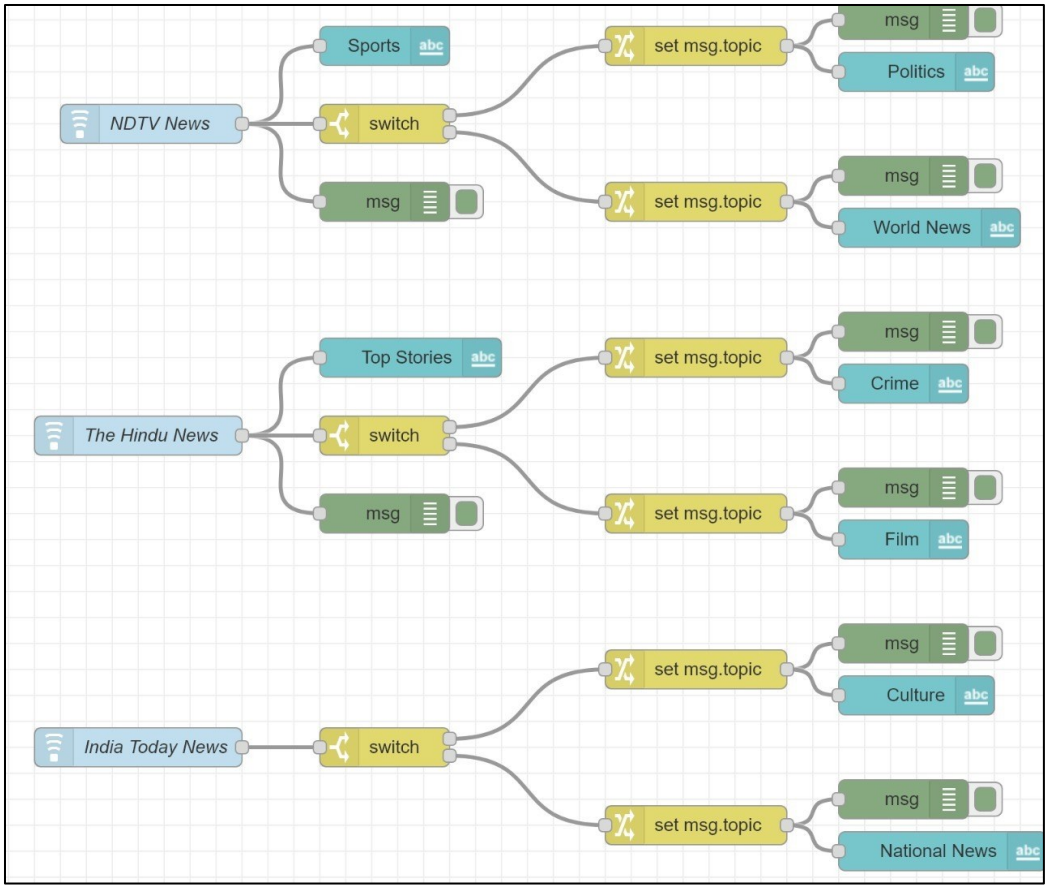
Dashboard

Debug window

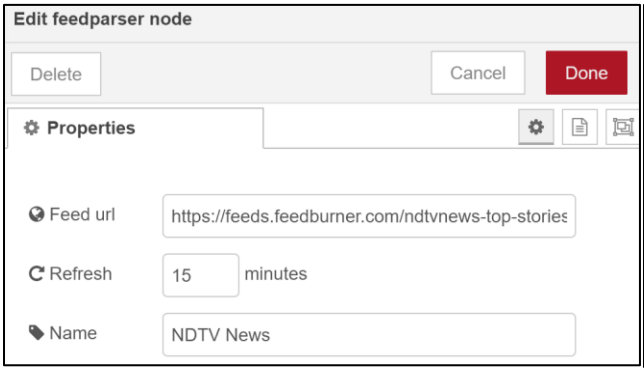


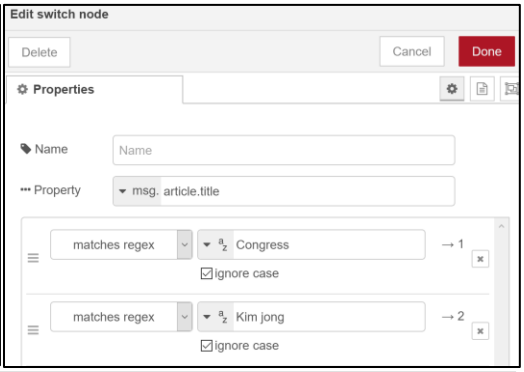
Task: Create a web API using template node and design a form to get reader details. On submitting reader details, make the reader read the article. Using feedparser node, create an online media monitoring system from 3 RSSFEEDS. Filter the news category with at least 3 verticals. Display the output in dashboard and debug monitor.

News Reader Flow:

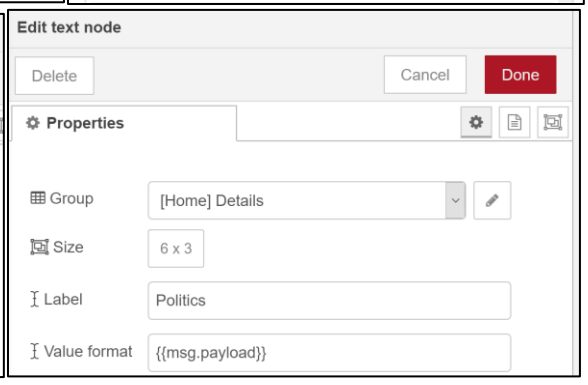

NDTV News
Sports
switch
msg
set msg.topic
set msg.topic
msg
Politics
msg
World News
The Hindu News
Top Stories
switch
msg
set msg.topic
set msg.topic
msg
Crime
msg
Film
India Today News
switch
set msg.topic
set msg.topic
msg
Culture
msg
National News

Properties:

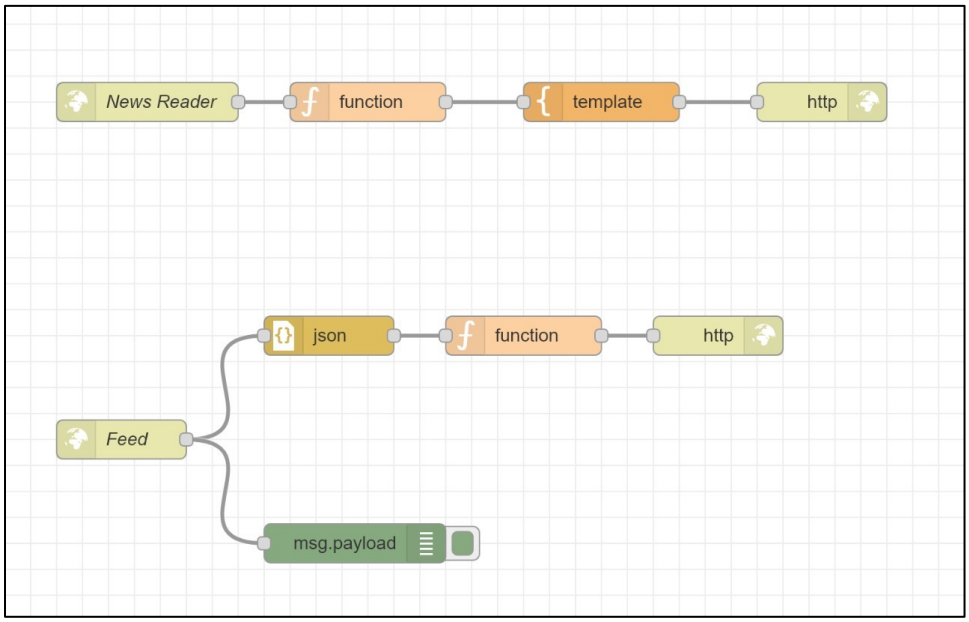

Edit feedparser node
Delete
Cancel
Done
Properties
Feed url
https://feeds.feedburner.com/ndtvnews-top-stories
Refresh
15
minutes
Name
NDTV News


Edit switch node
Delete
Cancel
Done
Properties
Name
Name
Property
msg. article.title
matches regex
Congress
→ 1
ignore case
matches regex
Kim jong
→ 2
ignore case

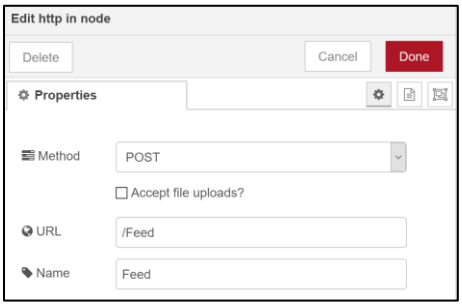

Edit change node
Delete
Cancel
Done
Properties
Name
Name
Rules
Set
msg. topic
to
article title


Edit text node
Delete
Cancel
Done
Properties
Group
[Home] Details
Size
6 x 3
Label
Politics
Value format
{{msg.payload}}

Form Flow:



Properties:



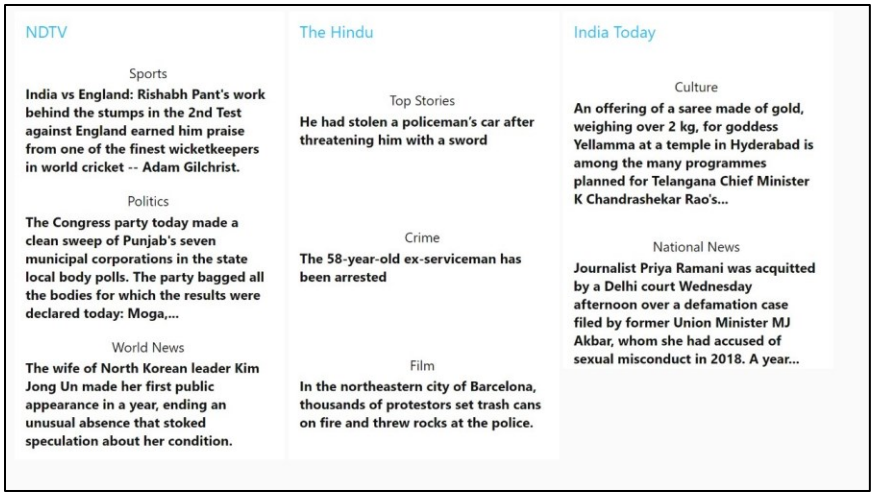
HTML code for template node:

```
<!DOCTYPE html>

<html>
    <head>
        <h1 style="background-color:Tomato;font-family:century gothic">News Reader</h1>
```

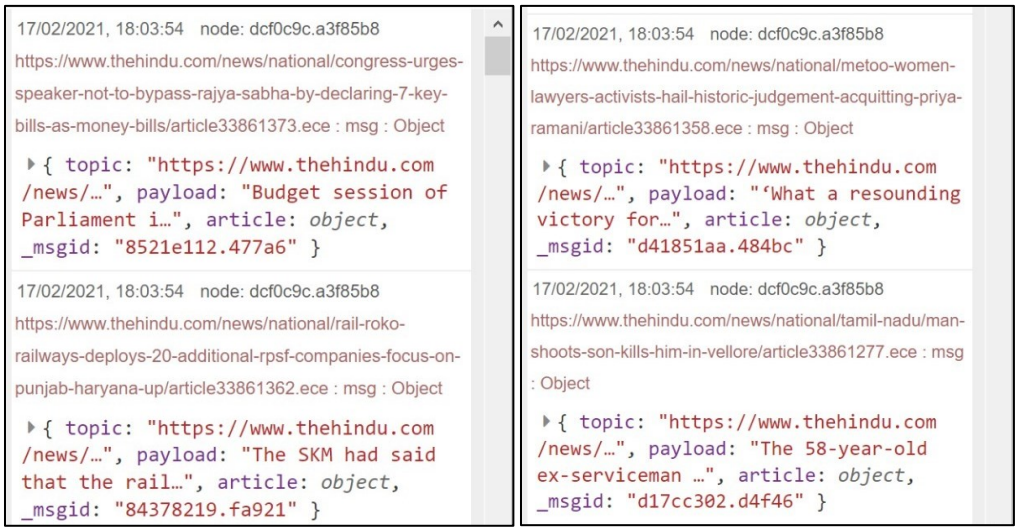
```
    </head>
    <body>
<p style="background-color:Tomato;">

<OL>
<LI> Enter your Name.
<LI> Enter Phone number and Email.
</OL>

<h4>
    <a href="https://feeds.feedburner.com/ndtvsports-latest"> NDTV</a>
    <a href="https://www.thehindu.com/business/Economy/feeder/default.rss"> The Hindu</a>
    <a href="https://www.indiatoday.in/rss/1206504"> India Today</a>

    </h4>

<form method="post" action="/{{url}}">

  <label for="name">Name:</label><br>
  <input type="text" id="fname" name="fname"><br><br>

  <label for="reg">Phone no:</label><br>
  <input type="text" id="reg" name="reg" ><br><br>

  <label for="topic">Email:</label><br>
  <input type="text" id="email" name="email" ><br><br>

  <input type="submit" value="Submit">

  <input type="reset" value="Reset" >

</form>

</body>
</html>
```

News Reader Output:

Dashboard:

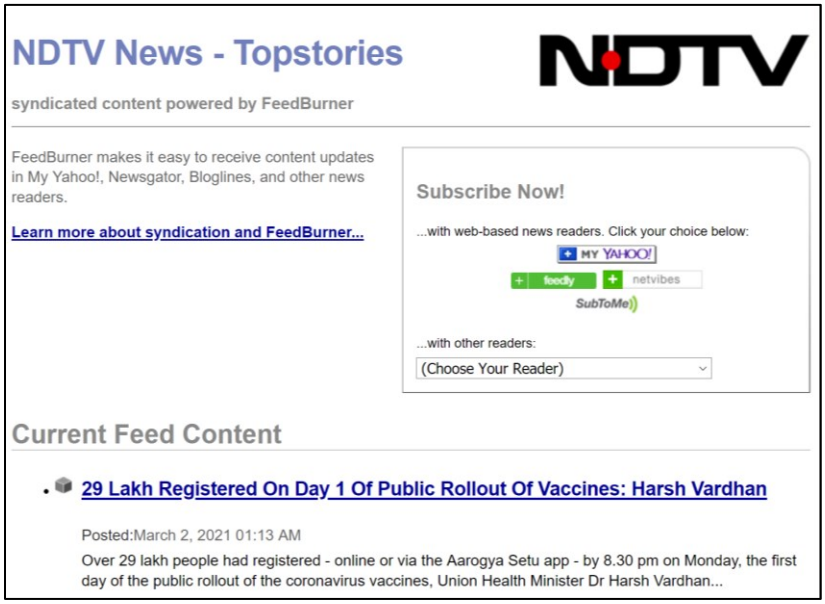


Debug Window:



News Form:

Hyperlinks:



Inference: Feedparser node is used to get RSS feeds of news websites like ESPN, NDTV, the Hindu, etc. The results are displayed in the dashboard using text node, and in the debug window using debug node.

Results: Hence the given tasks to monitor online news feeds using feedparser node have been carried out in node-red.

Experiment 4

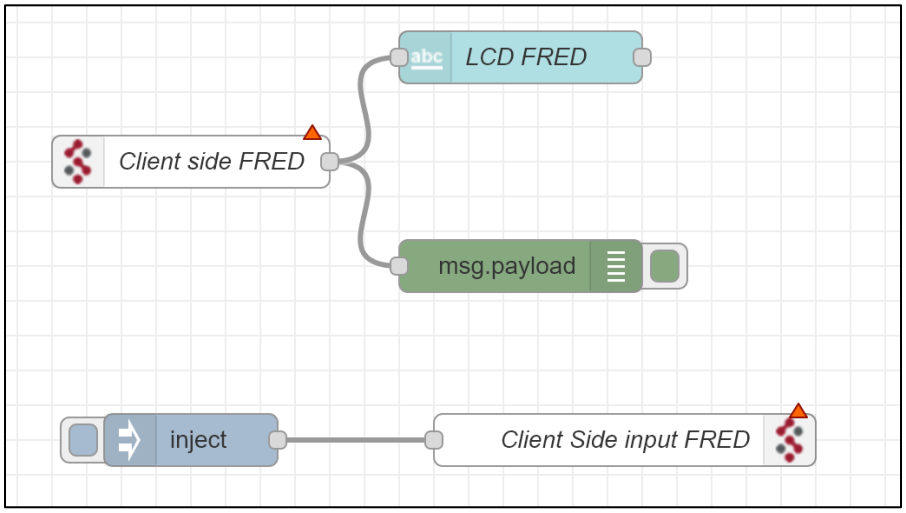
Front End for Node-RED (FRED) a cloud-based Node-RED

Aim: To use FRED, a cloud-based node-red to communicate between client and admin side.

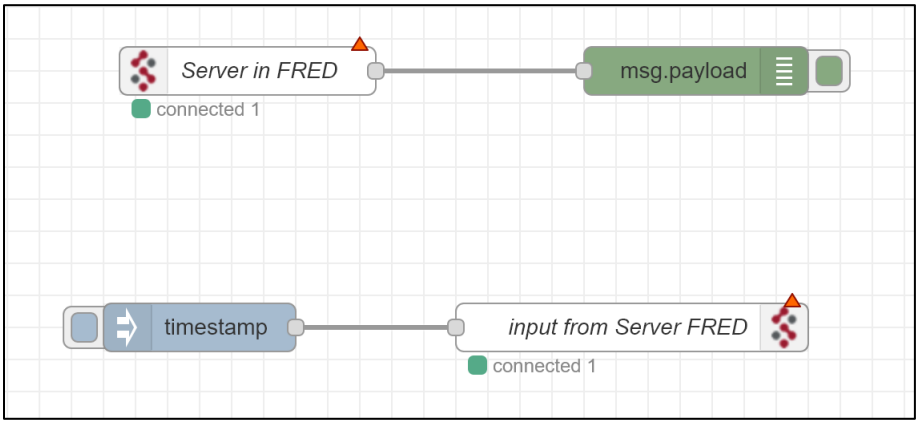
Procedure:

- Trigger node red in cmd using node-red -v
- Configure the flow as shown according to the properties on node-red and in FRED
- Deploy to see results in debug window

Node-red Flow:

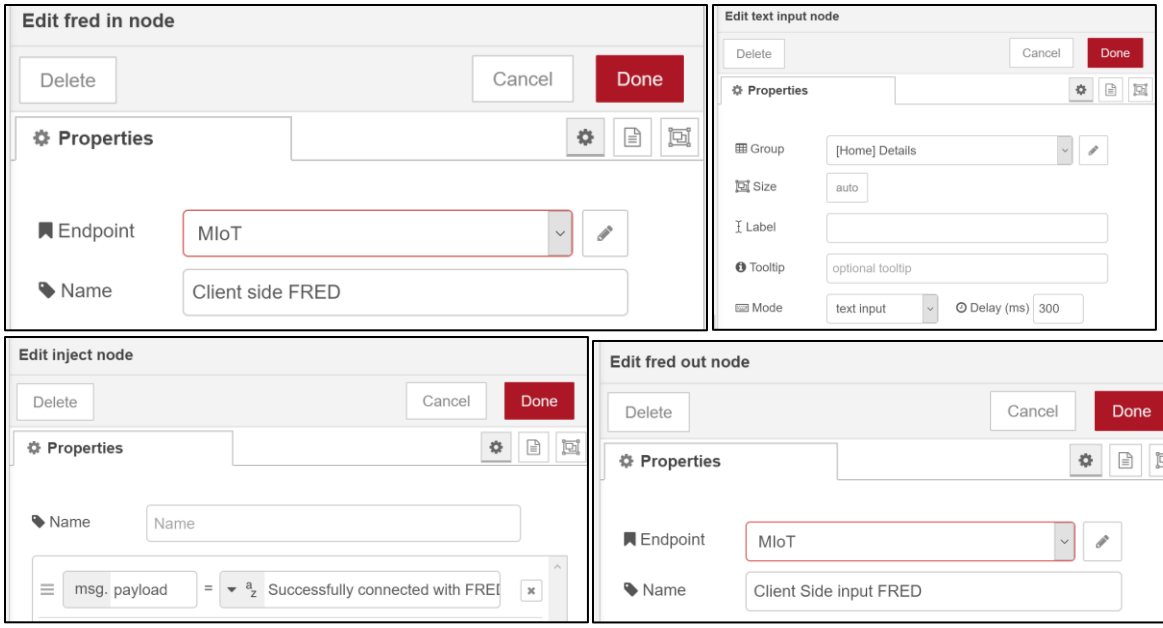


FRED flow:

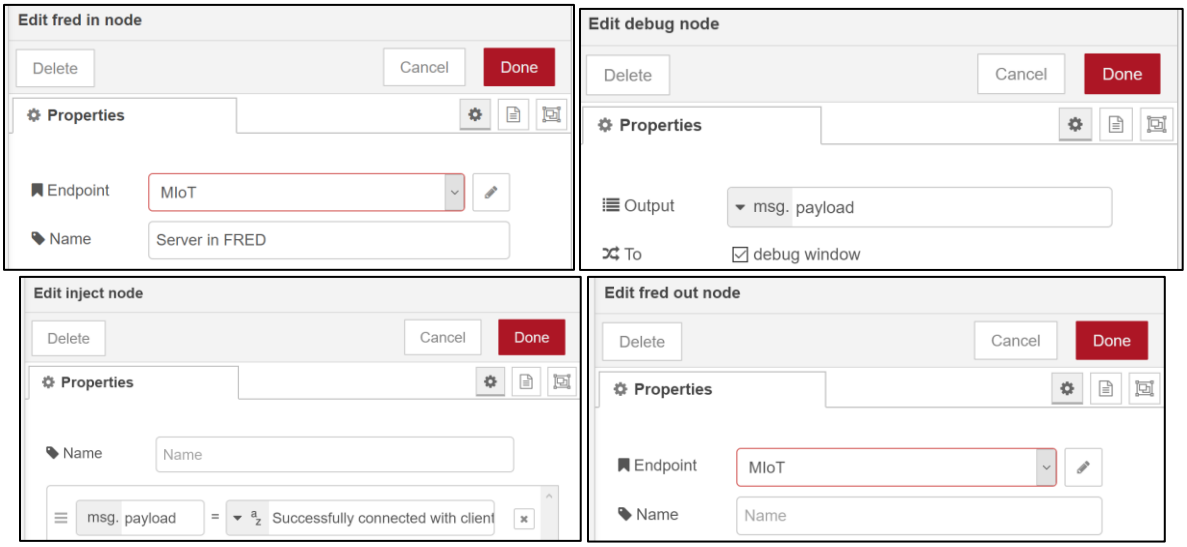


Properties:

Client side/Node-red:



Server side/FRED:

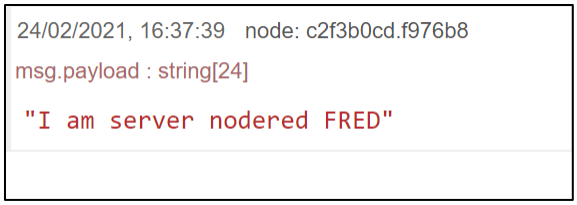


Client-side output:

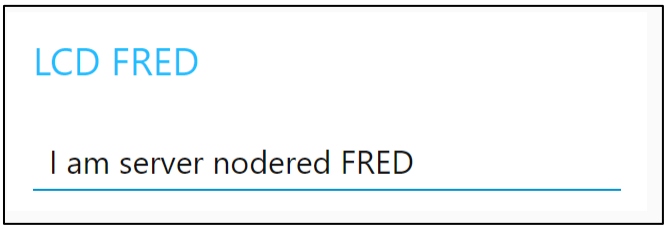
```
02/03/2021, 03:46:07   node: 63fa7ce7.3c1b24
msg.payload : string[32]
"Successfully connected with FRED"
```

Server-side Output:

Debug Window

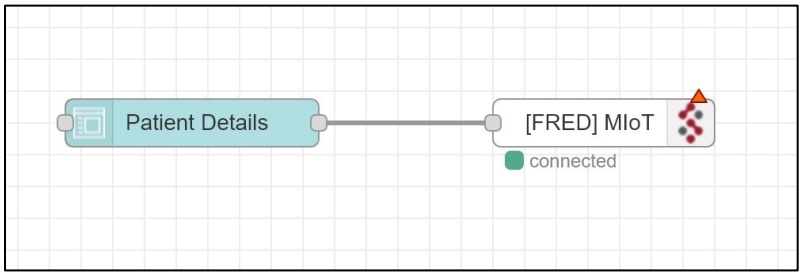


Dashboard

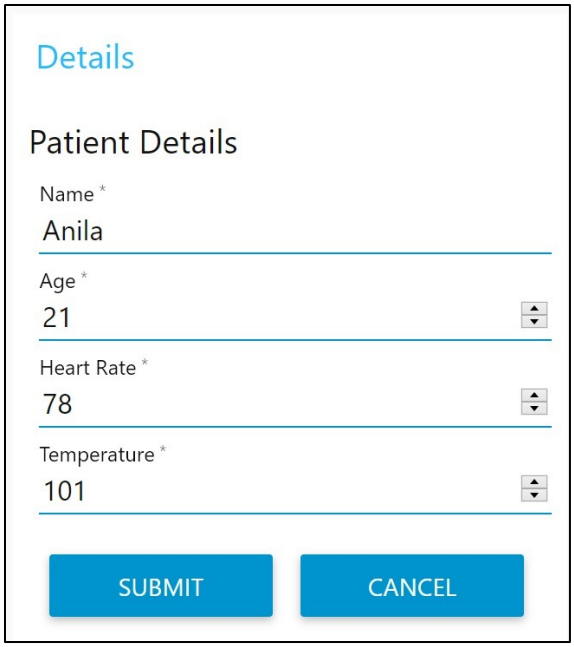


Task: Create a hospital-patient service using cloud-based FRED and node-red to patient details like name, temperature, heart-rate, etc. and suggest appropriate treatment.

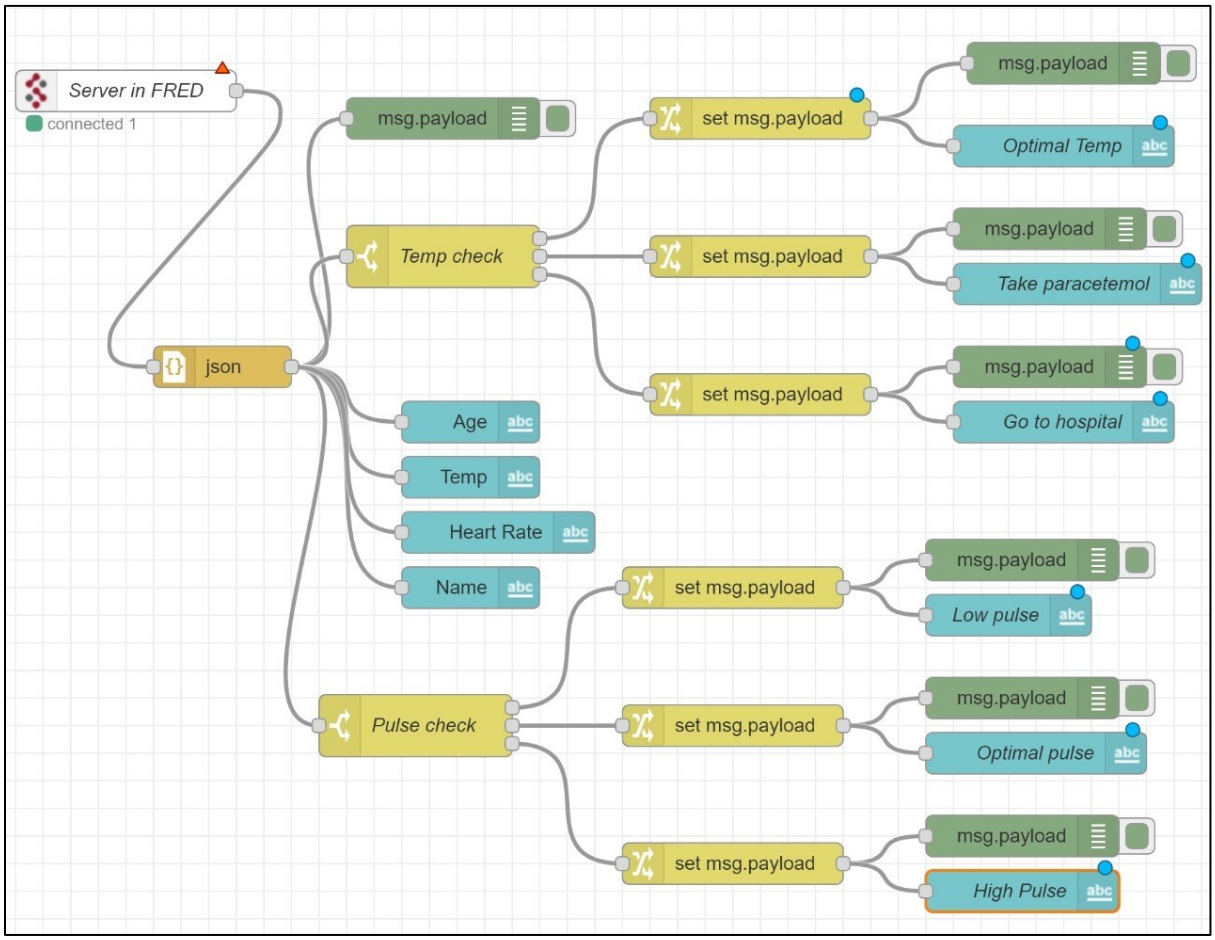
Client/Patient Side:



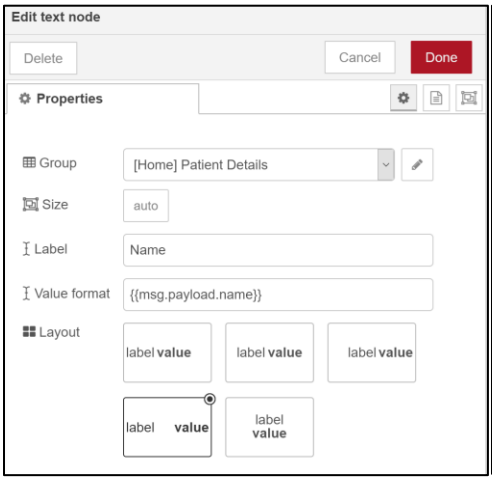
Dashboard (form Input):

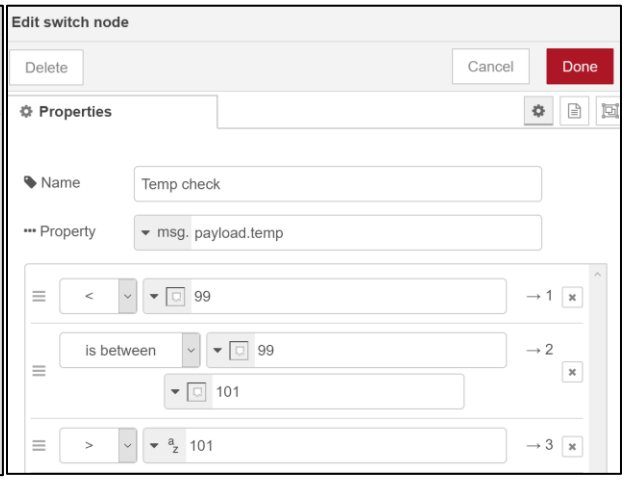


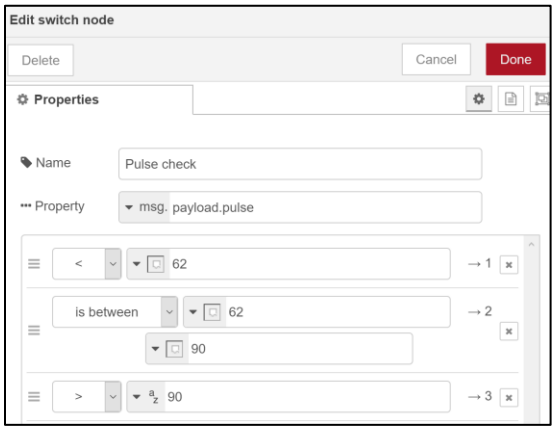
Server/Hospital Admin side:

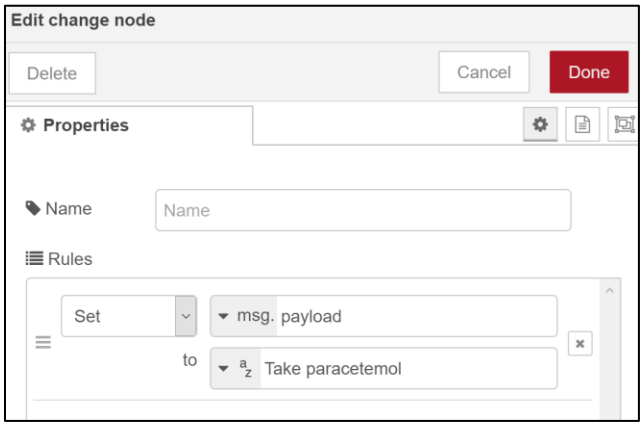


Properties:





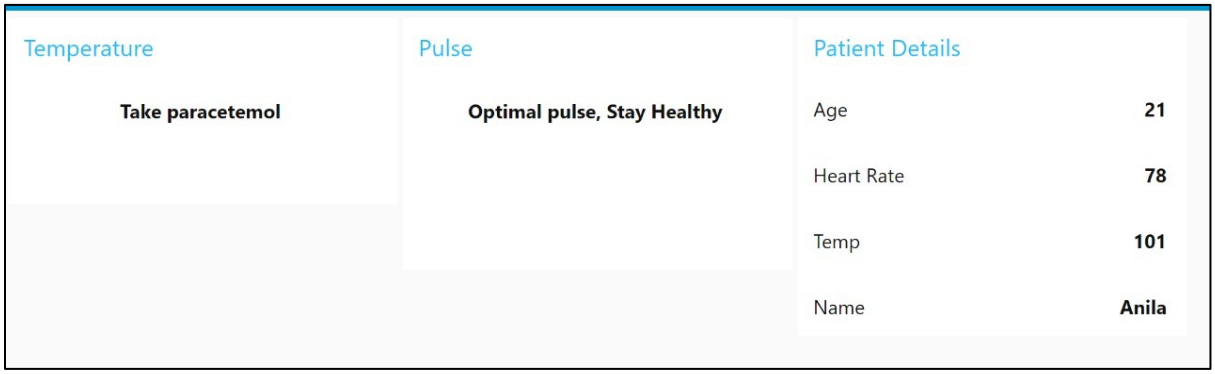




Output:

Debug Window

Dashboard



Inference: FRED in node is used to get input from the other side via cloud and FRED out is used to send data. In the task, form node (form in dashboard) is used to get patient details on the client side. This info is sent in json format to the server/hospital admin side, where appropriate treatment is suggested based on the symptoms using switch and change nodes and displayed in the debug and dashboard windows.

Result: Hence, cloud-based FRED and node-red have been used to perform server-client operations and display outputs.

Experiment 5

Weather Report Analysis using OpenWeatherMap node in Node-Red and FRED

Aim: To create a weather report analysis system using OpenWeatherMap node in Node-Red and FRED

Theory:

OpenWeatherMap Nodes – Two nodes that get the weather report from OpenWeatherMap.

- Input Node
- Query node

Input Node:

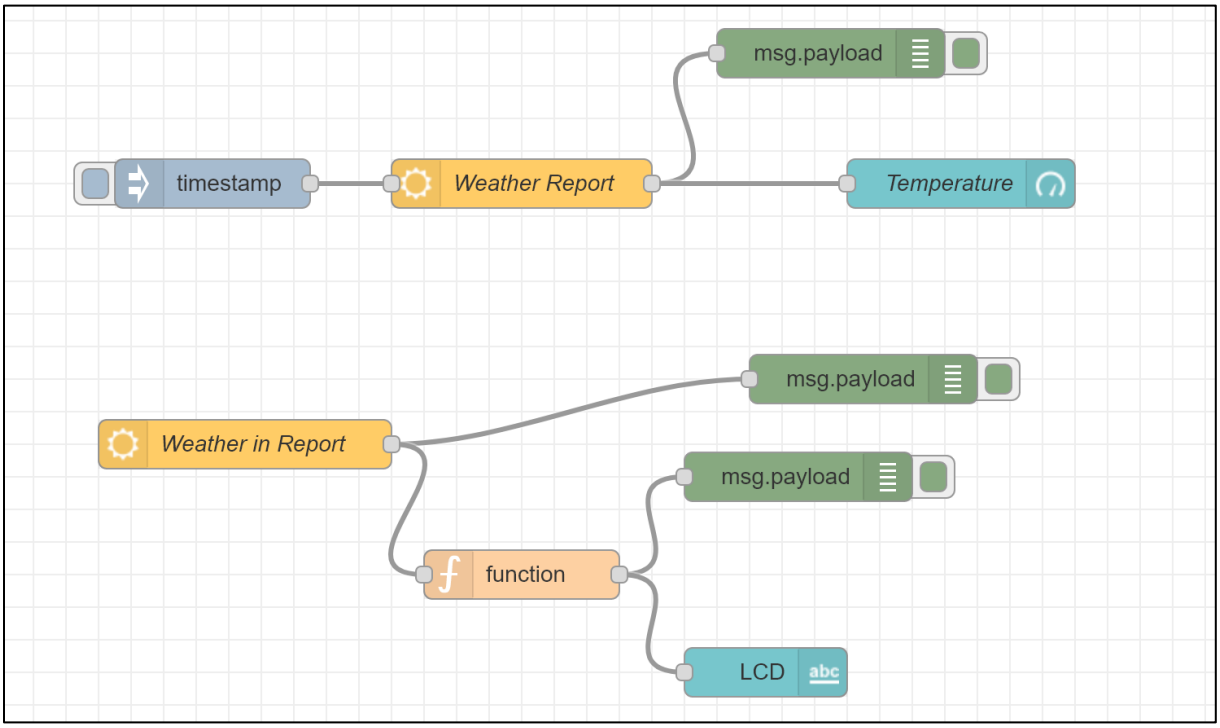
- Fetches the current weather or day forecast at a location specified by city and country or latitude and longitude every 10 minutes.
- Outputs a message if something has changed.

Query Node: Accepts an input to trigger fetching the current weather either from a specific city and country or latitude and longitude or passed in on.

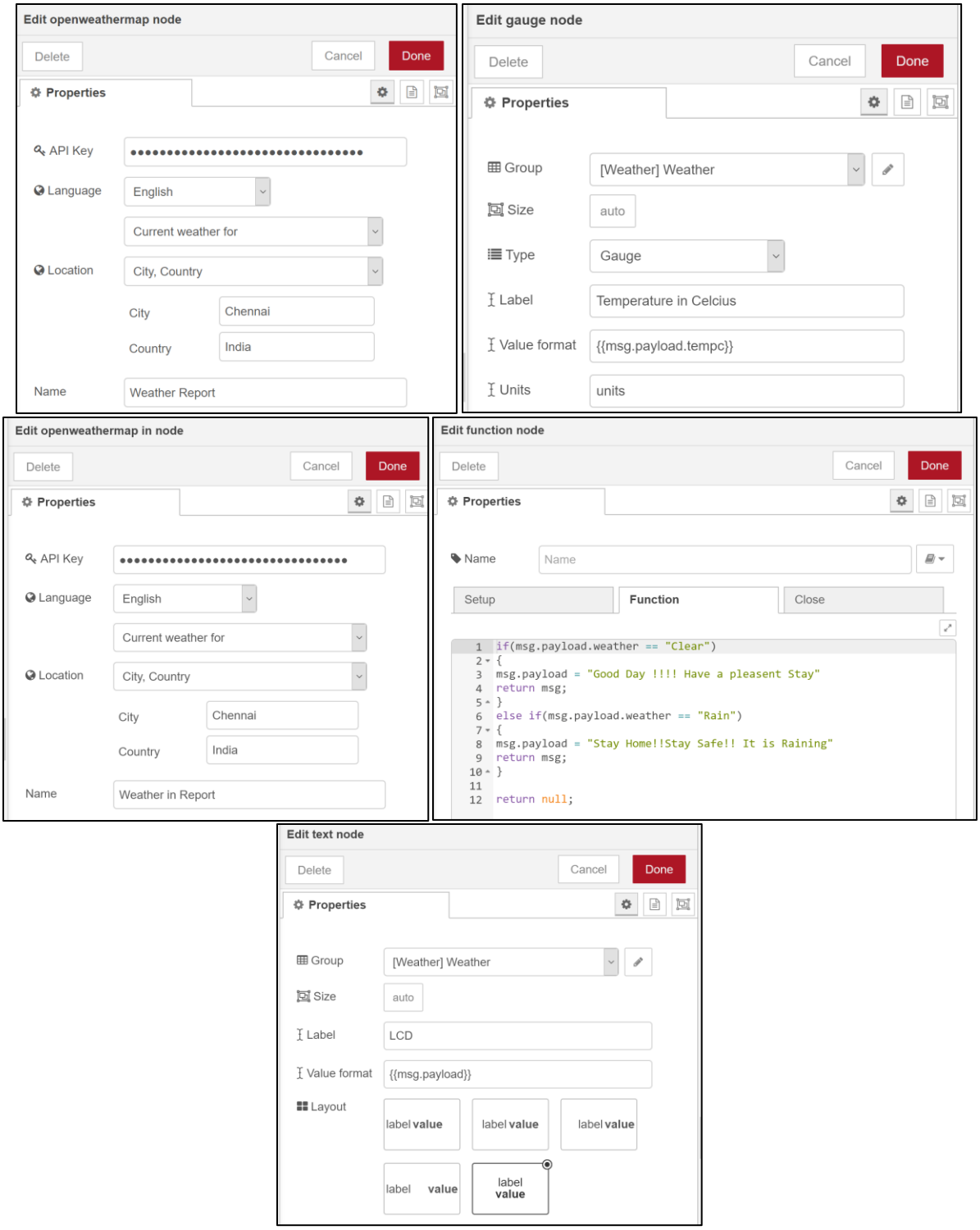
Procedure:

- Trigger node red in cmd using node-red -v
- Configure the flow as shown according to the properties on node-red and in FRED
- Deploy to see results in debug window

Flow:

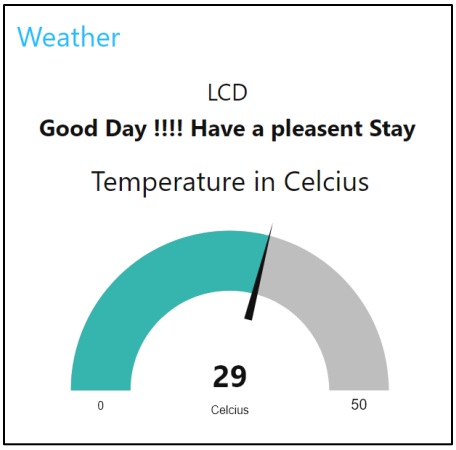


Properties:



Output:

Dashboard



Debug Window

03/03/2021, 16:32:48 node: 7184ff9.1e42c8
msg.payload : Object

```
object
  id: 801
  weather: "Clouds"
  detail: "few clouds"
  icon: "02d"
  tempk: 304.15
  tempc: 31
  temp_maxc: 31
  temp_minc: 31
  humidity: 35
  pressure: 1011
  maxtemp: 304.15
  mintemp: 304.15
  windspeed: 3.6
  winddirection: 160
  location: "Chennai"
  sunrise: 1614732838
  sunset: 1614775684
  clouds: 20
  description: "The weather in Chennai at
  coordinates: 13.0878, 80.2785 is Clouds (few
  clouds)."
```

03/03/2021, 16:32:48 node: 4324cb7b.5612ec
msg.payload : string[34]

```
"Good Day !!!! Have a pleasent Stay"
```

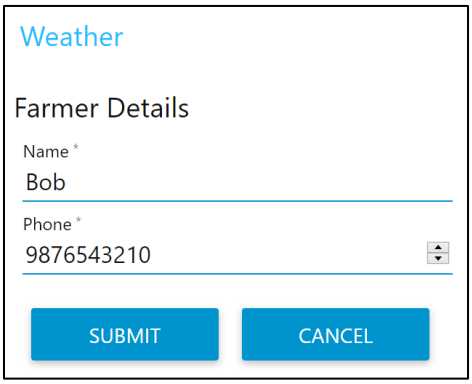

Task: Design a cloud-based weather analysis and forecasting automation through FRED. Retrieve details from https://openweathermap.org, perform analysis which helps the farmer for their agriculture and send appropriating forecast details

Client Device: Get farmer details in local Node-Red and send to the server

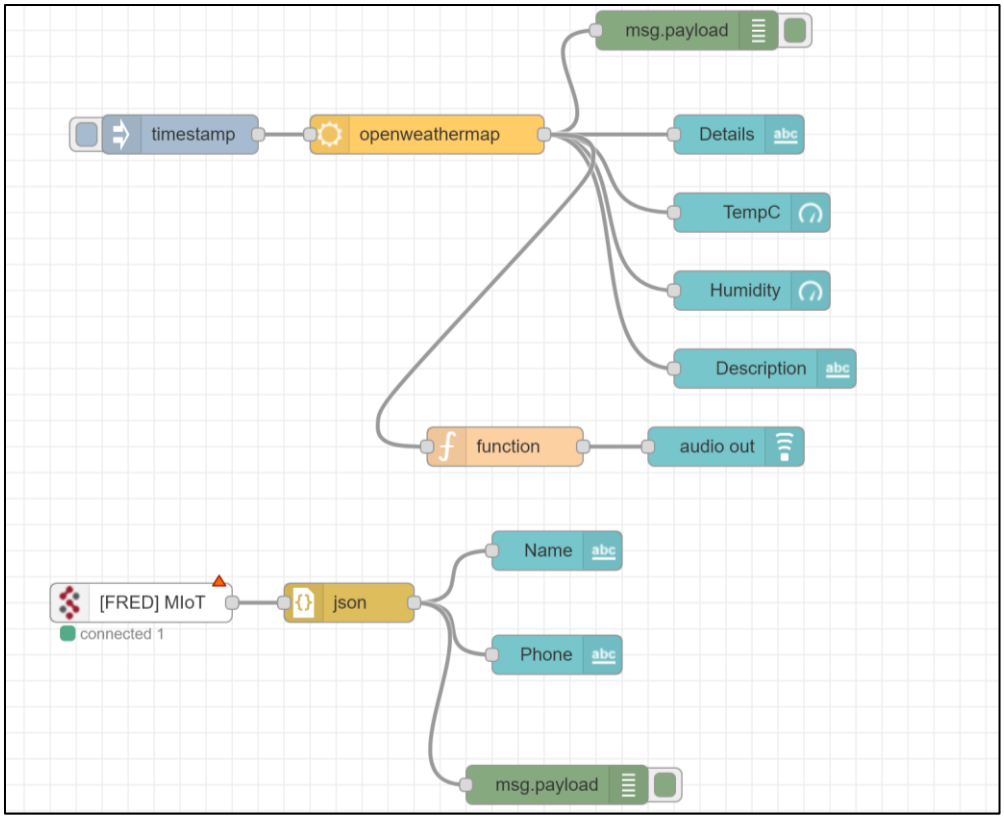
Server Device: in FRED, display all weather details and produce a visual representation of all parameters retrieved from OpenWeatherMap which may help farmers visualize the current state of weather. Enable audio output so that farmers with poor eyesight can hear the details. Display the output in UI and Debug window

Client-side Flow:

Dashboard Input



Server-Side Flow:

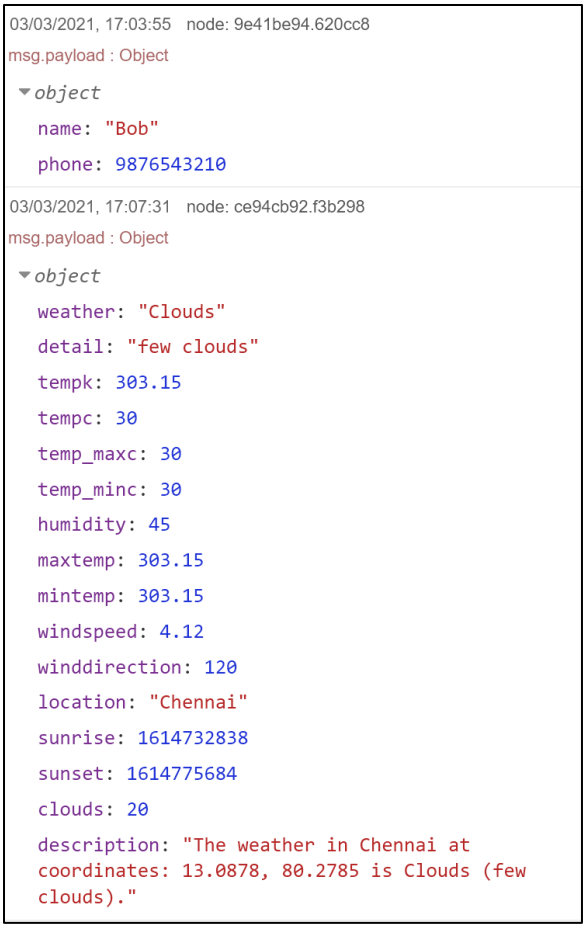


Function for audio:

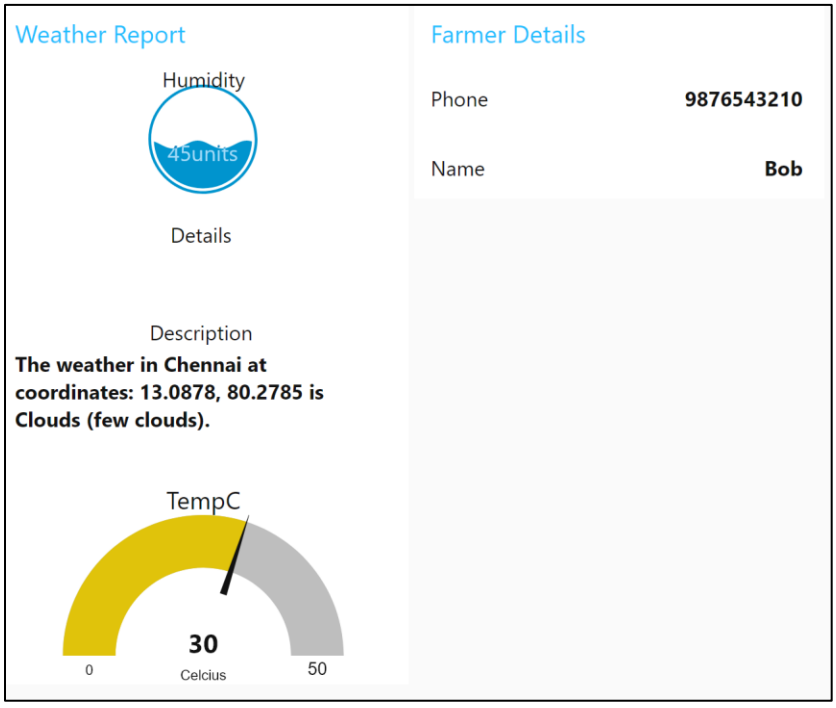
```
msg.payload="Temperature: " + msg.payload.tempc + "Humidity: " + msg.payload.humidity +
"Description: " + msg.payload.description
return msg;
```

Output:

Debug Window


03/03/2021, 17:03:55 node: 9e41be94.620cc8
msg.payload : Object
object
name: "Bob"
phone: 9876543210
03/03/2021, 17:07:31 node: ce94cb92.f3b298
msg.payload : Object
object
weather: "Clouds"
detail: "few clouds"
tempk: 303.15
tempc: 30
temp_maxc: 30
temp_minc: 30
humidity: 45
maxtemp: 303.15
mintemp: 303.15
windspeed: 4.12
winddirection: 120
location: "Chennai"
sunrise: 1614732838
sunset: 1614775684
clouds: 20
description: "The weather in Chennai at coordinates: 13.0878, 80.2785 is Clouds (few clouds)."

Dashboard


Weather Report
Humidity
45units
Details
Description
The weather in Chennai at coordinates: 13.0878, 80.2785 is Clouds (few clouds).
TempC
30
Celcius
0
50
Farmer Details
Phone
9876543210
Name
Bob

Inference: The openweathermap node is used to retrieve weather details of a city like temperature, humidity, etc. from https://openweathermap.org. These details are then displayed in the dashboard using text and gauge nodes. In the task, farmer details are retrieved using the form node through dashboard and sent to the server side through FRED out node. In the server side, weather details are retrieved from the website and displayed along with the farmer details in the debug window and dashboard. Audio output is also enabled using the audio output node.

Results: Hence, OpenWeatherMap node has been successfully used to perform weather analysis in Node-Red and FRED.

Experiment 6

ThingsBoard – NodeRed Posting Data in Dashboard through HTTP

Aim: To use ThingsBoard to post device data through HTTP

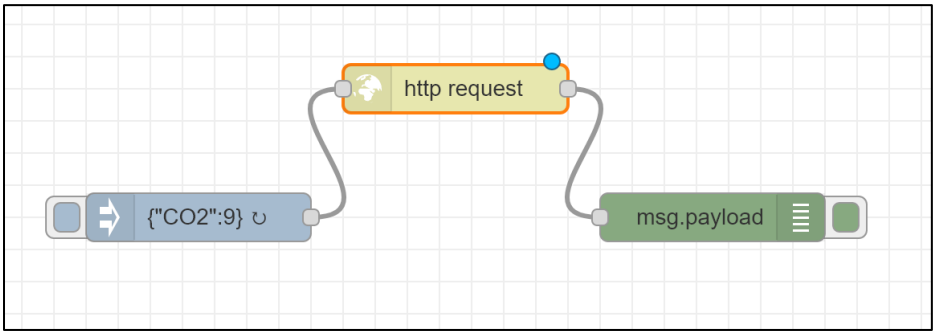
Theory: ThingsBoard

- Open-source IoT Platform
- Device management, data collection, processing and visualization for IoT solutions.
- It enables device connectivity via industry standard
- IoT protocols - MQTT, COAP and HTTP and supports both cloud and on-premises deployments.
- ThingsBoard combines scalability, fault-tolerance and performance so we will never lose our data.

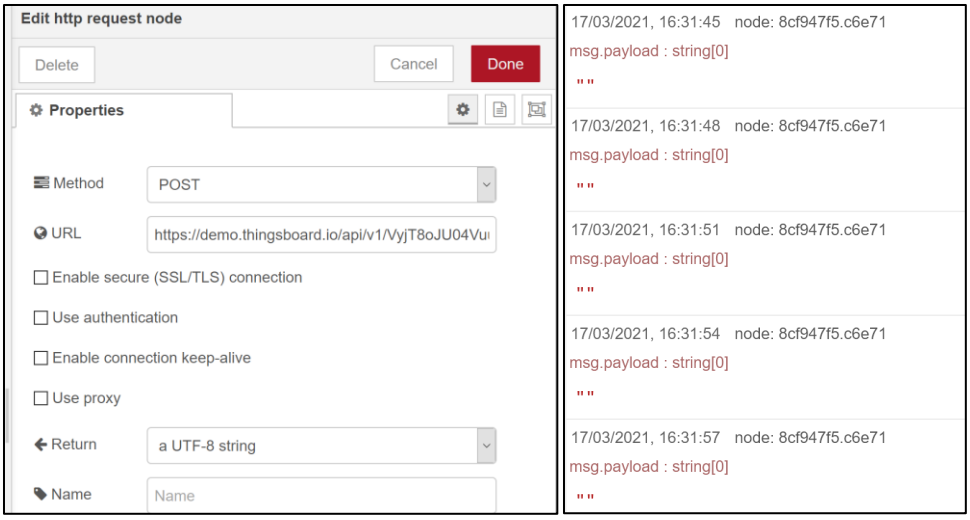
Procedure:

- Trigger node red in cmd using node-red -v
- Configure the flow as shown according to the properties on node-red and in FRED
- Deploy to see results in debug window

Flow:



Properties:

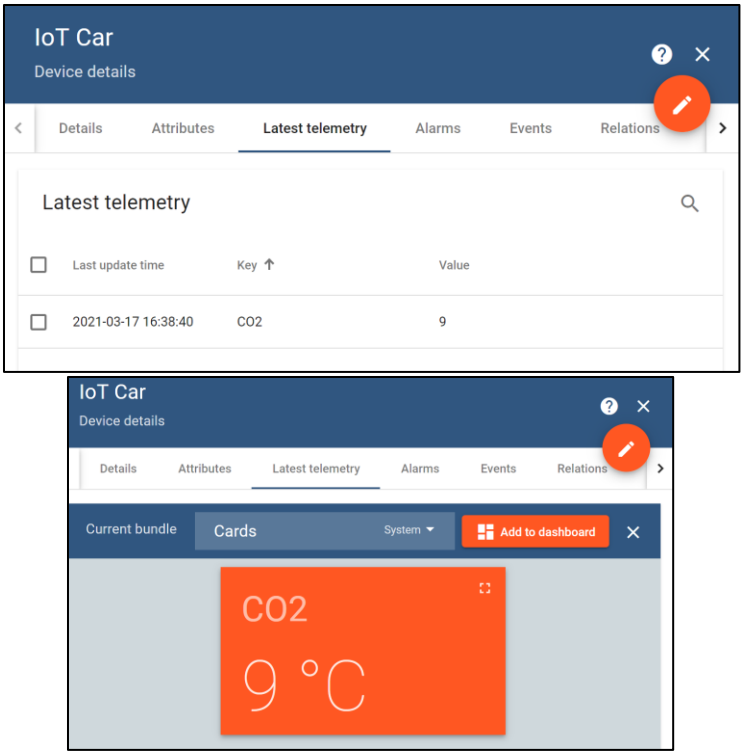


Key: VyjT8oJU04Vuu1BGNToh

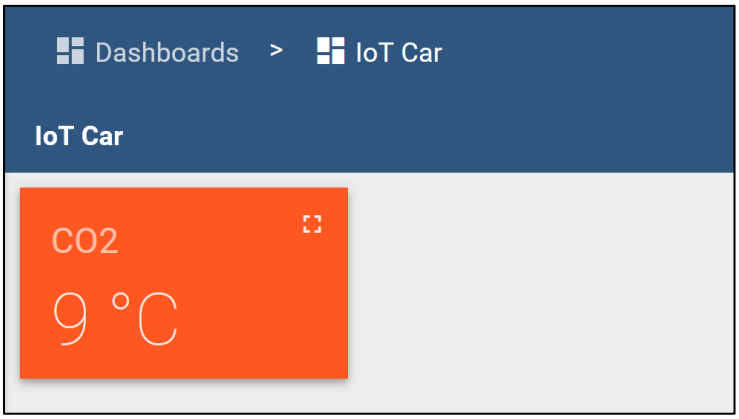
Output:

Debug Window

ThingsBoard

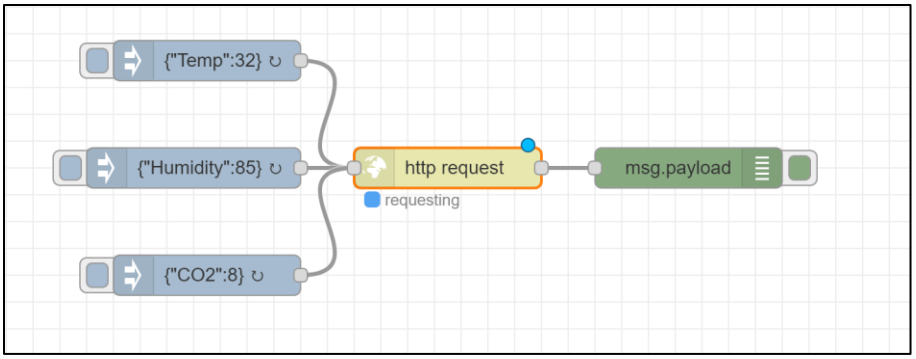


ThingsBoard Dashboard



Task: Using node-red create a flow which performs automatic plant health monitoring. Using appropriate sensors simulate various parameters like air temperature, humidity, CO2, etc. Display the visualization in ThingsBoard Dashboard Widgets.

Flow:

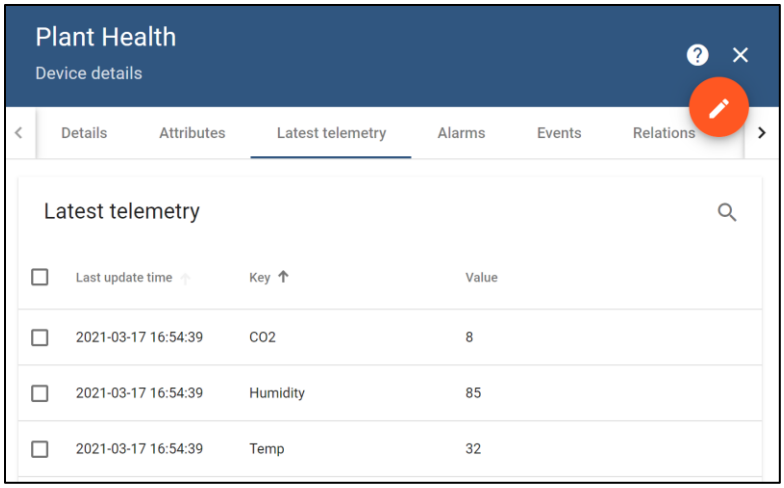


Debug

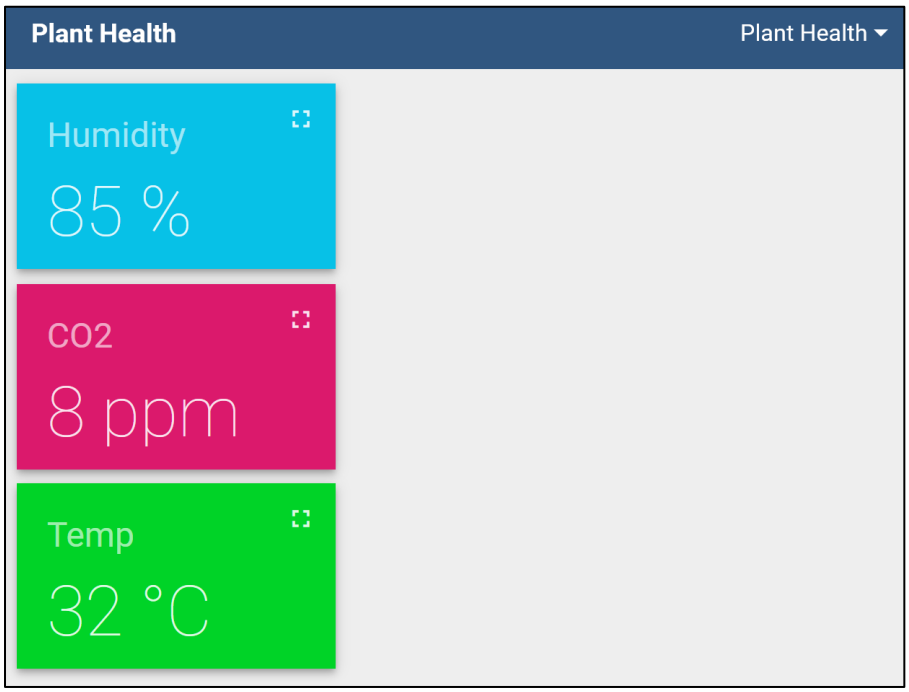
Key for Task: rX9ZwBM3pLECgUxafSxI

Output:

ThingsBoard



ThingsBoard Dashboard



Inference: ThingsBoard is used to monitor sensor values. The sensor values are provided through the inject node. Then a device profile is created in ThingsBoard and the sensor values are displayed through widgets.

Results: Hence sensor data monitoring through http has been done in Node-Red and ThingsBoard and displayed in ThingsBoard Dashboard

Experiment 7

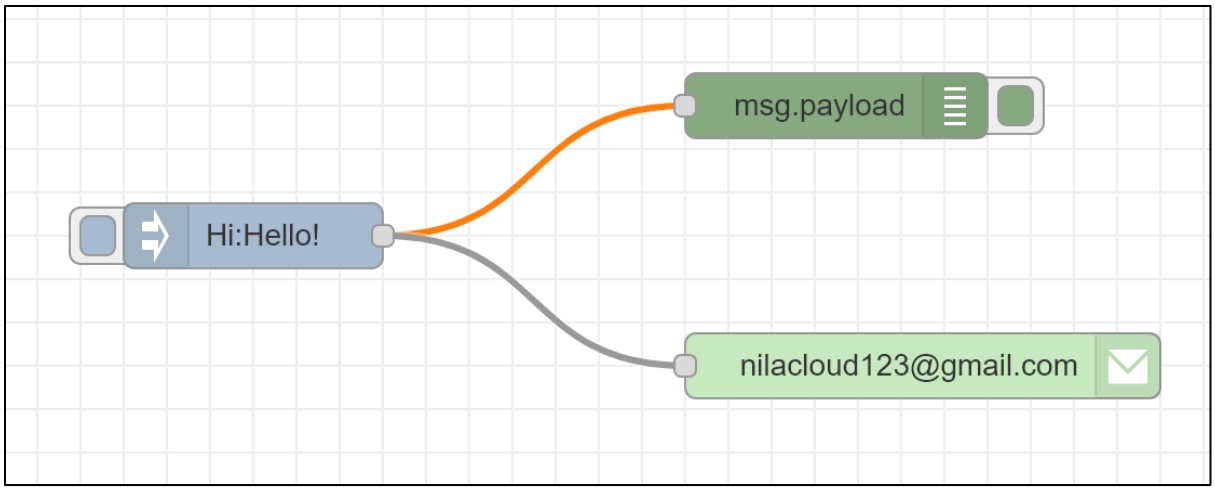
Email using Node-Red

Aim: To send emails via email node in Node-Red

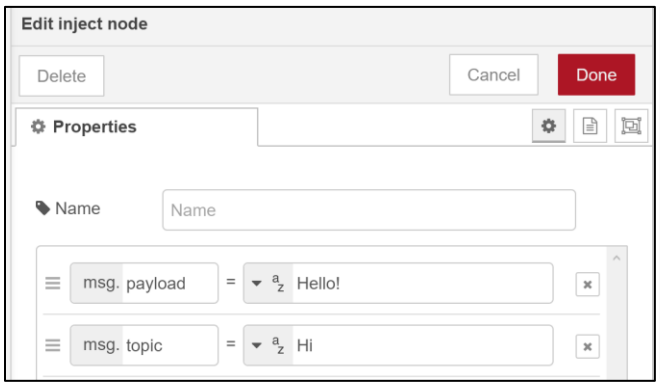
Procedure:

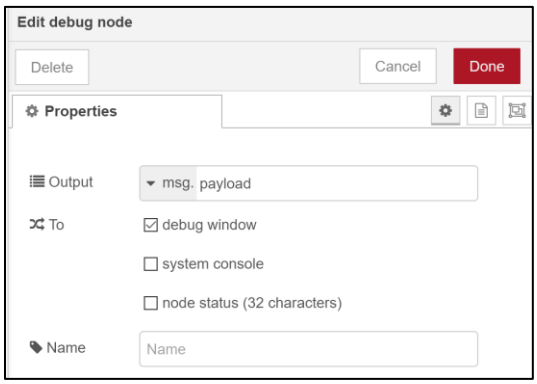
- Trigger Node-Red in cmd using node-red -v
- Open Node-Red and arrange flow as shown and configure properties
- Enter email and password. Then run

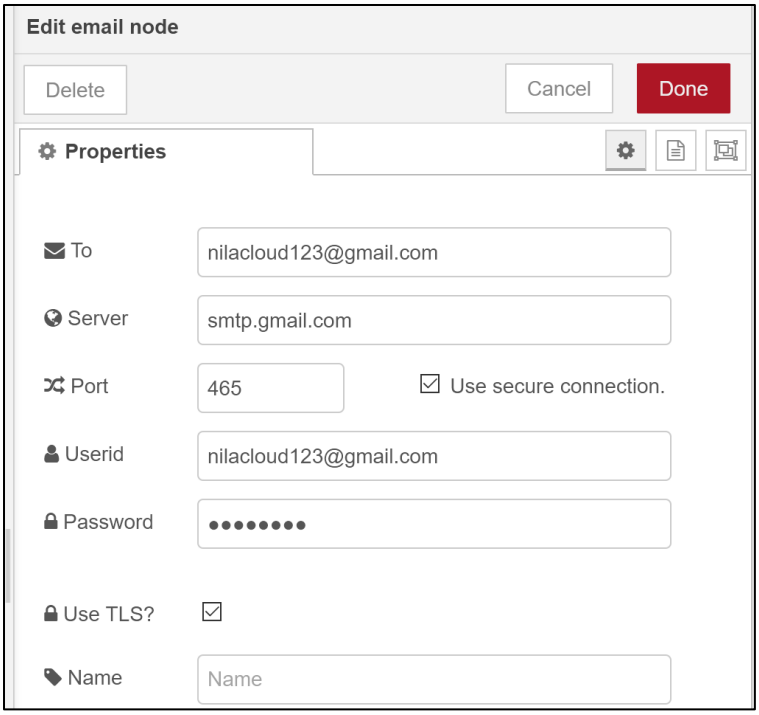
Flow:



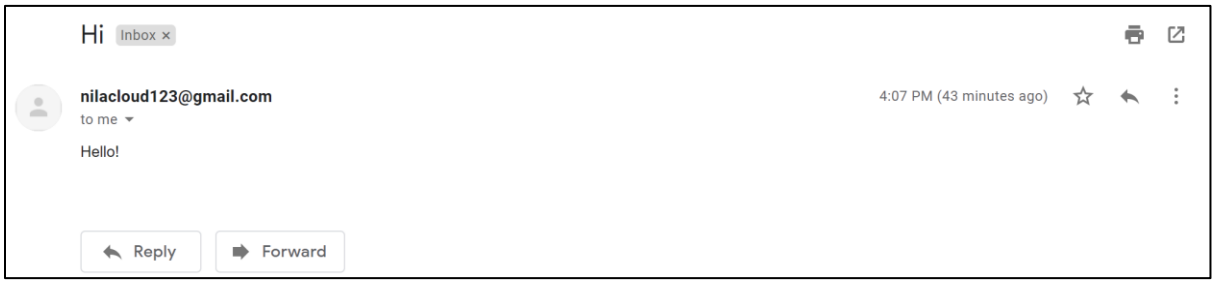
Properties:





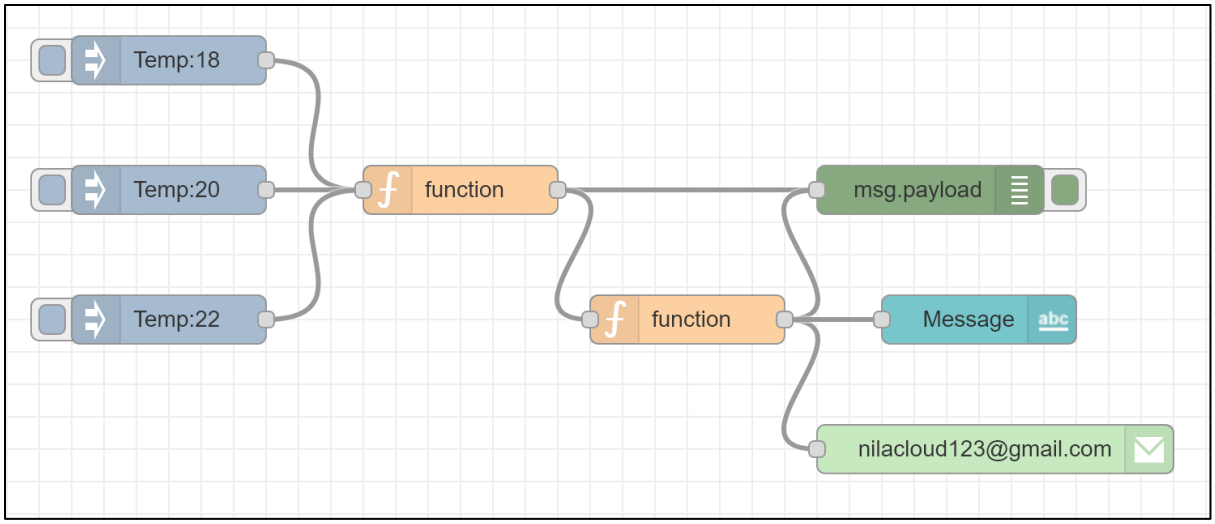


Output:

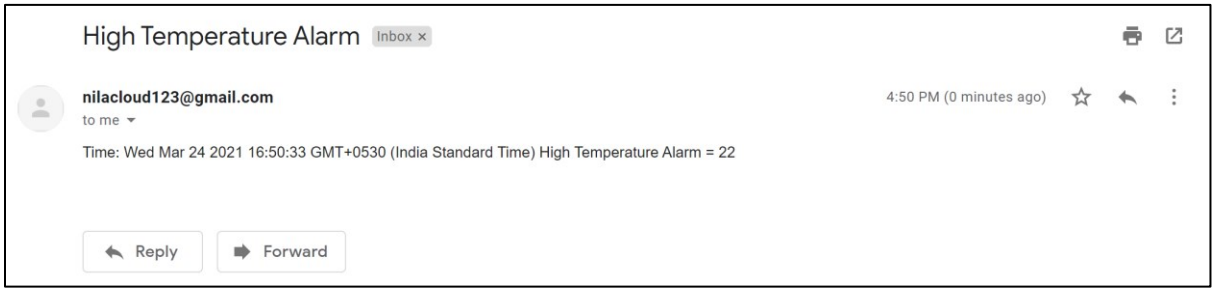


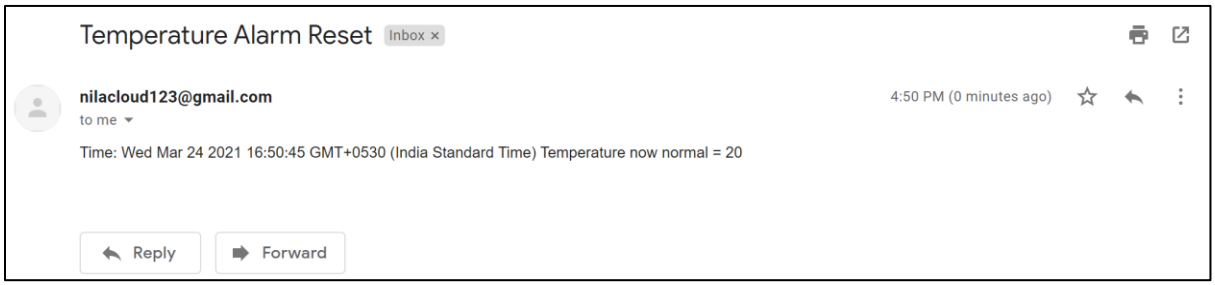
Task 1:

Flow:

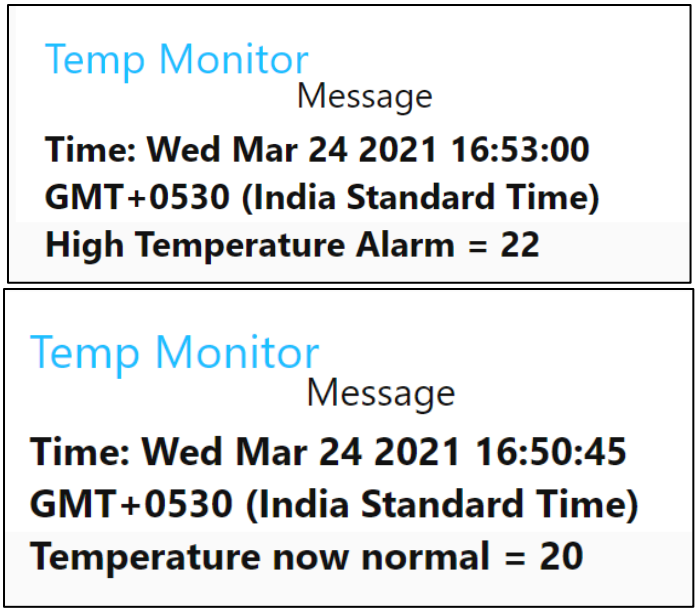


Properties:



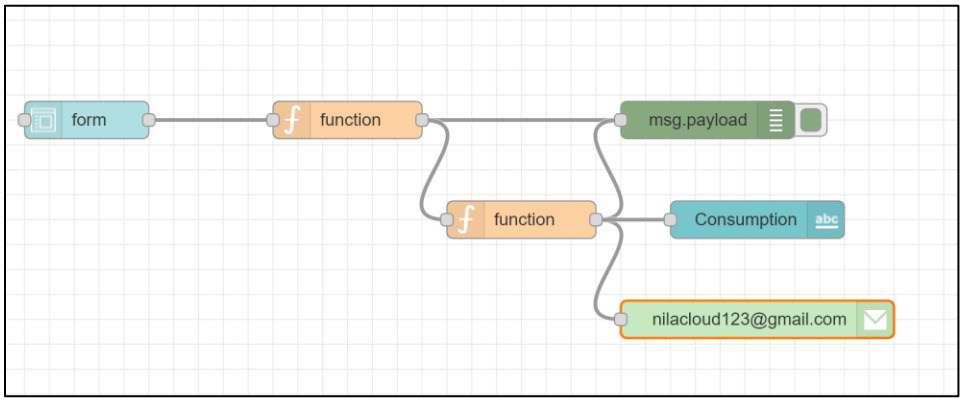


Output:

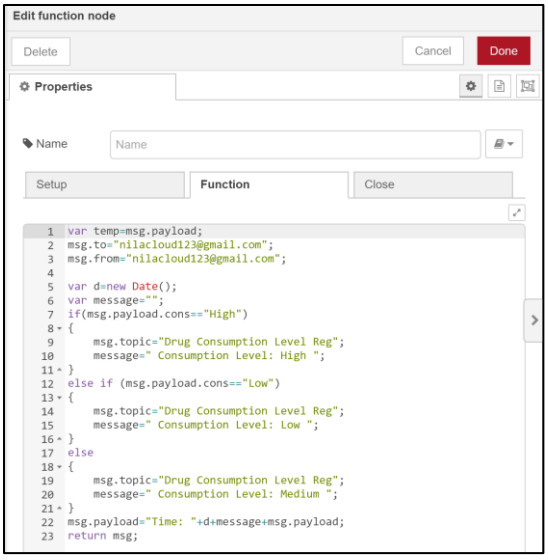


Task 2:

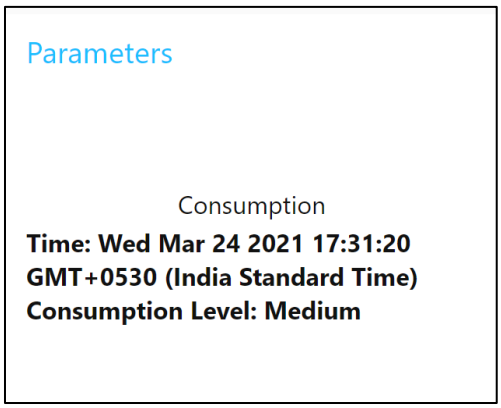
Flow:



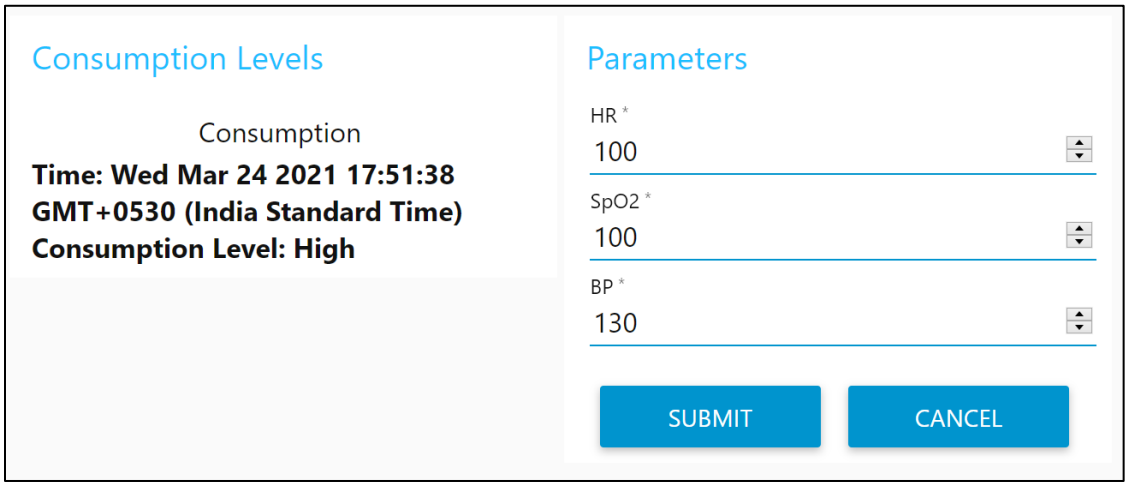
Properties:



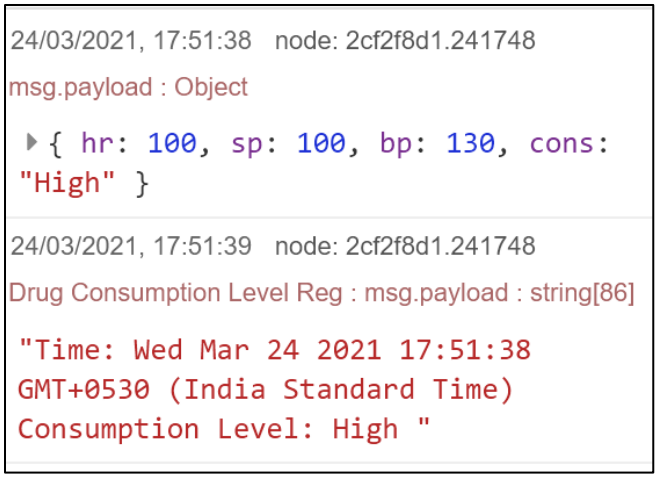
Output:



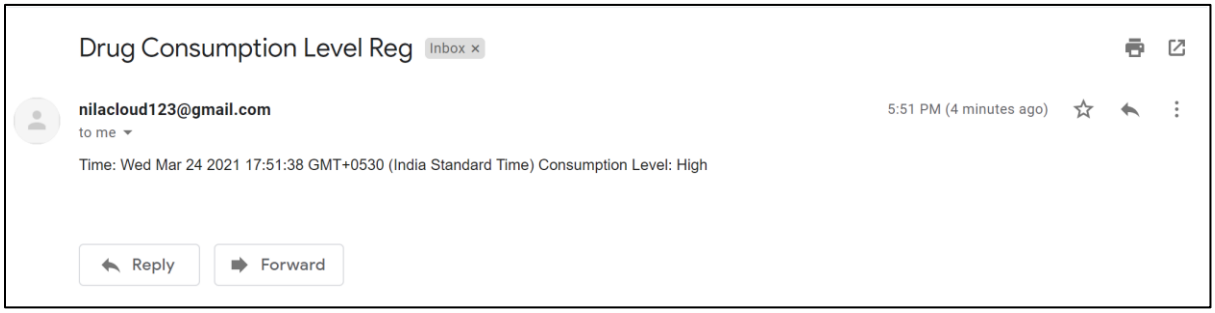
Output with Form Node in Dashboard:



Debug Window:



Email:



Inference: The email node was configured in node-red to send emails to the required email address. The emails sent were according to the constraints set by the function node.

Result: Hence, sending emails via Node-red was successfully executed

Experiment 8

Telemedicine using Node-Red and FRED

Aim: To collect the patient's symptoms for COVID-19 and display the doctor's decision in dashboard and debug window of node-red.

Task: The Patients who have corona will have the following symptoms

Most common symptoms:

- Fever
- dry cough
- tiredness

Less common symptoms:

- headache
- aches and pains
- sore throat
- diarrhoea
- conjunctivitis
- loss of taste or smell
- a rash on skin, or discolouration of fingers or toes

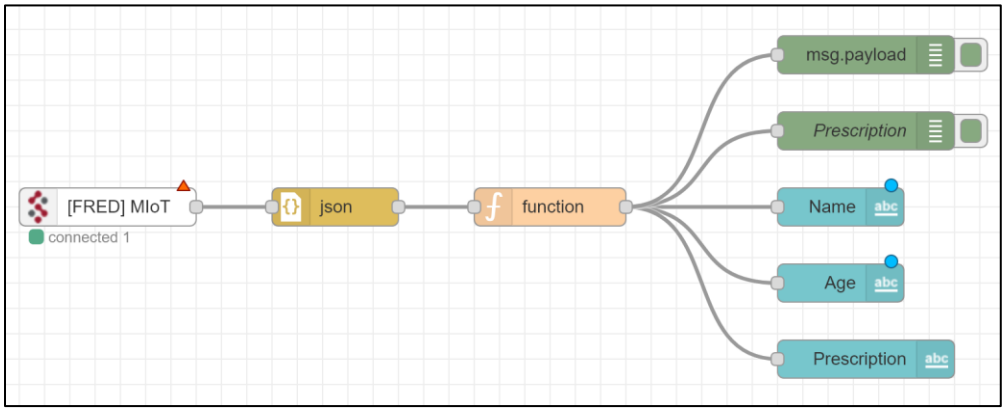
Serious symptoms:

- difficulty breathing or shortness of breath
- chest pain or pressure
- loss of speech or movement

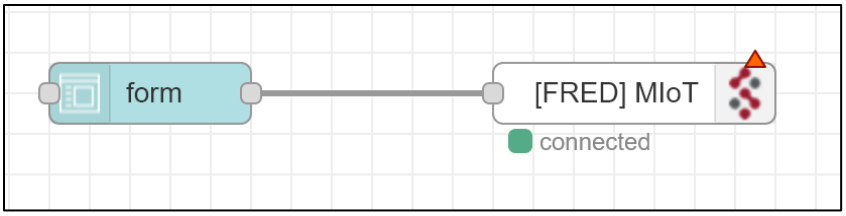
Assume the patient is seeking for telemedicine, For the serious symptoms doctor may suggest the patient to be admitted in the hospital, For the other symptoms doctor may prescribe medicine. Display the doctor's decision in dashboard and debug window of Node red.

Flow:

Server-side

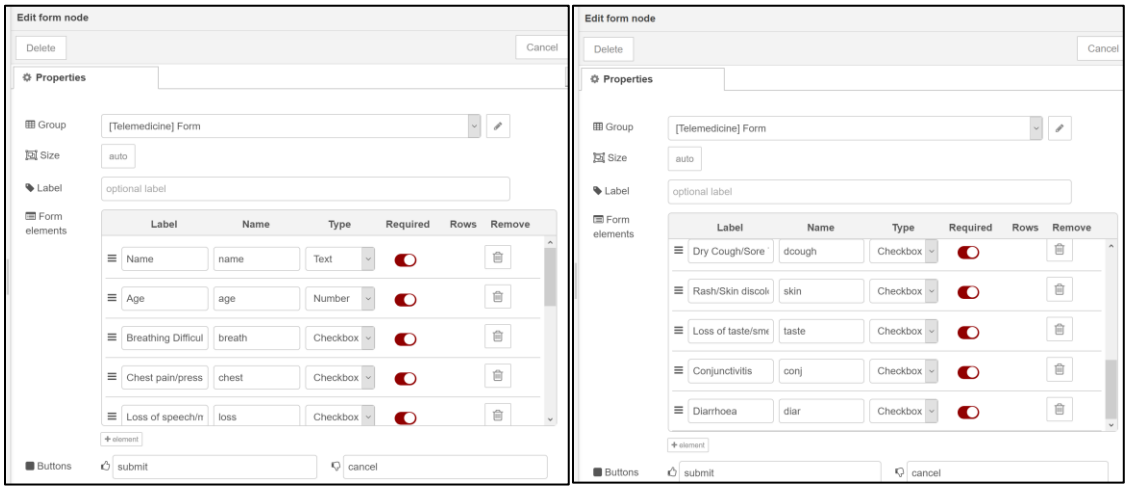


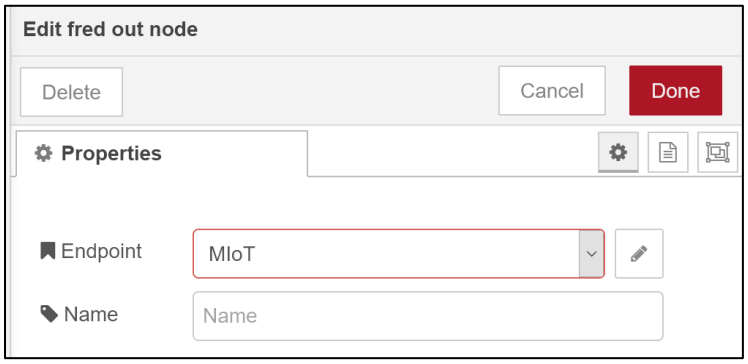
Client-side



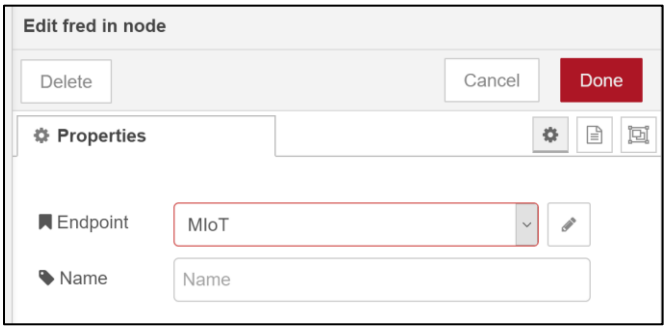
Properties:

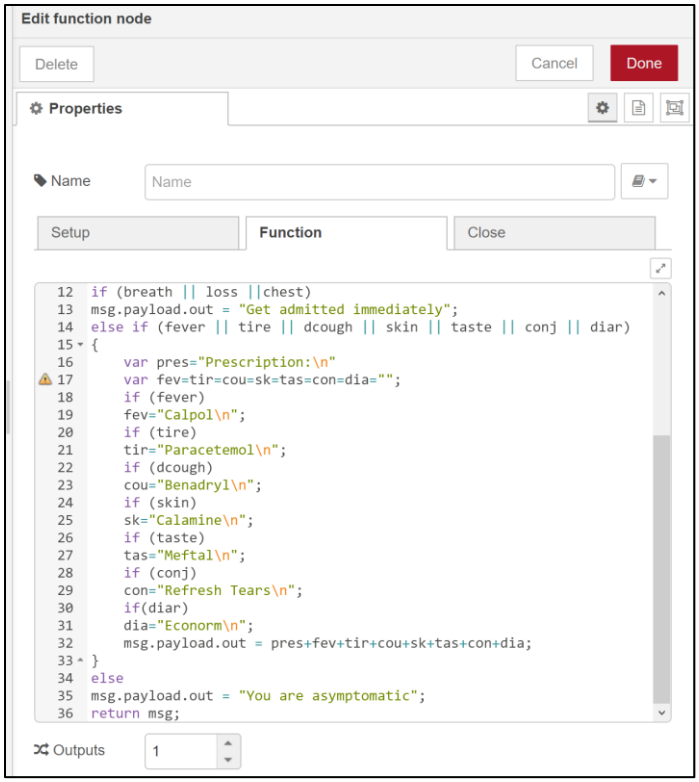
Client-side





Server-side

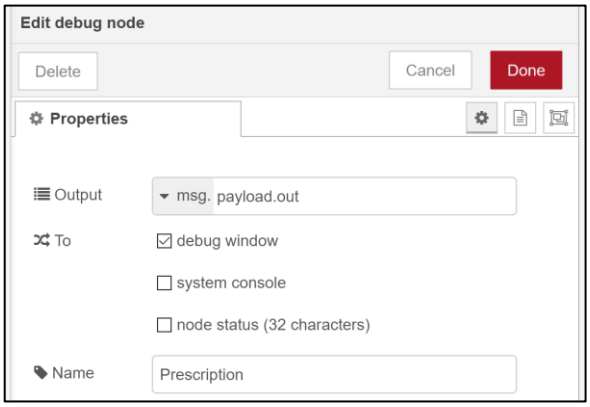


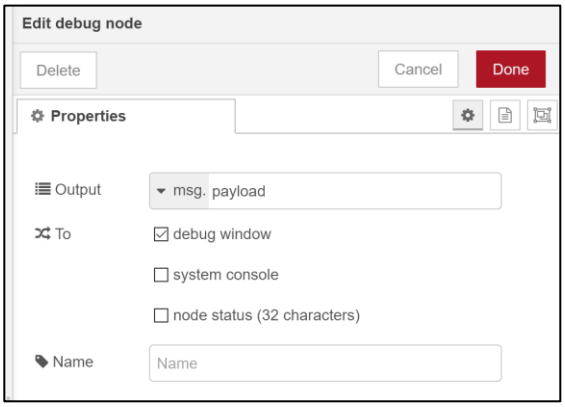


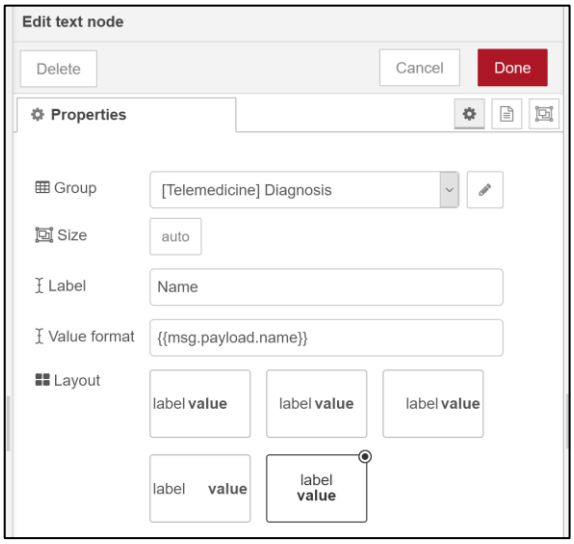
Code:

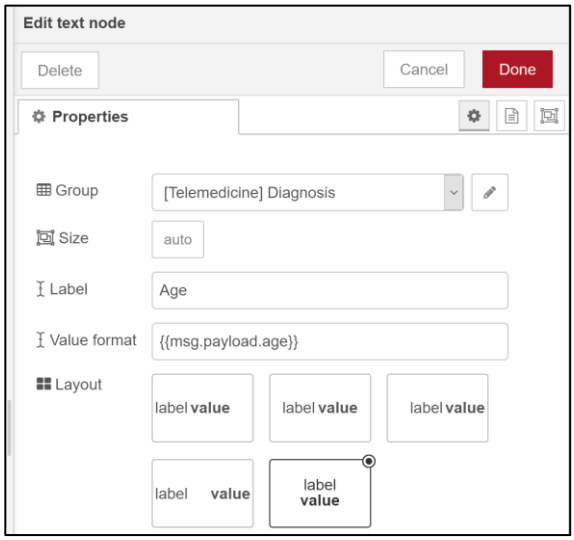
```
var breath=msg.payload.breath;
var loss=msg.payload.loss;
var chest=msg.payload.chest;
var fever=msg.payload.fever;
var tire=msg.payload.tire;
var dcough=msg.payload.dcough;
var skin=msg.payload.skin;
var taste=msg.payload.taste;
var conj=msg.payload.conj;
var diar=msg.payload.diar;
var out;
if (breath || loss ||chest)
msg.payload.out = "Get admitted immediately";
else if (fever || tire || dcough || skin || taste || conj || diar)
{
    var fev=tir=cou=sk=tas=con=dia="";
    if (fever)
    fev="Calpol\n";
    if (tire)
    tir="Paracetemol\n";
    if (dcough)
    cou="Benadryl\n";
    if (skin)
```

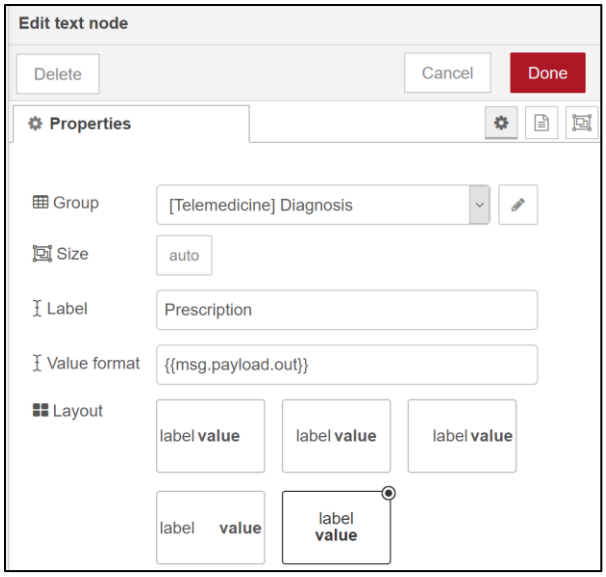
```
        sk="Calamine\n";
        if (taste)
        tas="Meftal\n";
        if (conj)
        con="Refresh Tears\n";
        if(diar)
        dia="Econorm\n";
        msg.payload.out = fev+tir+cou+sk+tas+con+dia;
    }
    else
    msg.payload.out = "You are asymptomatic / Do not have COVID";
return msg;
```









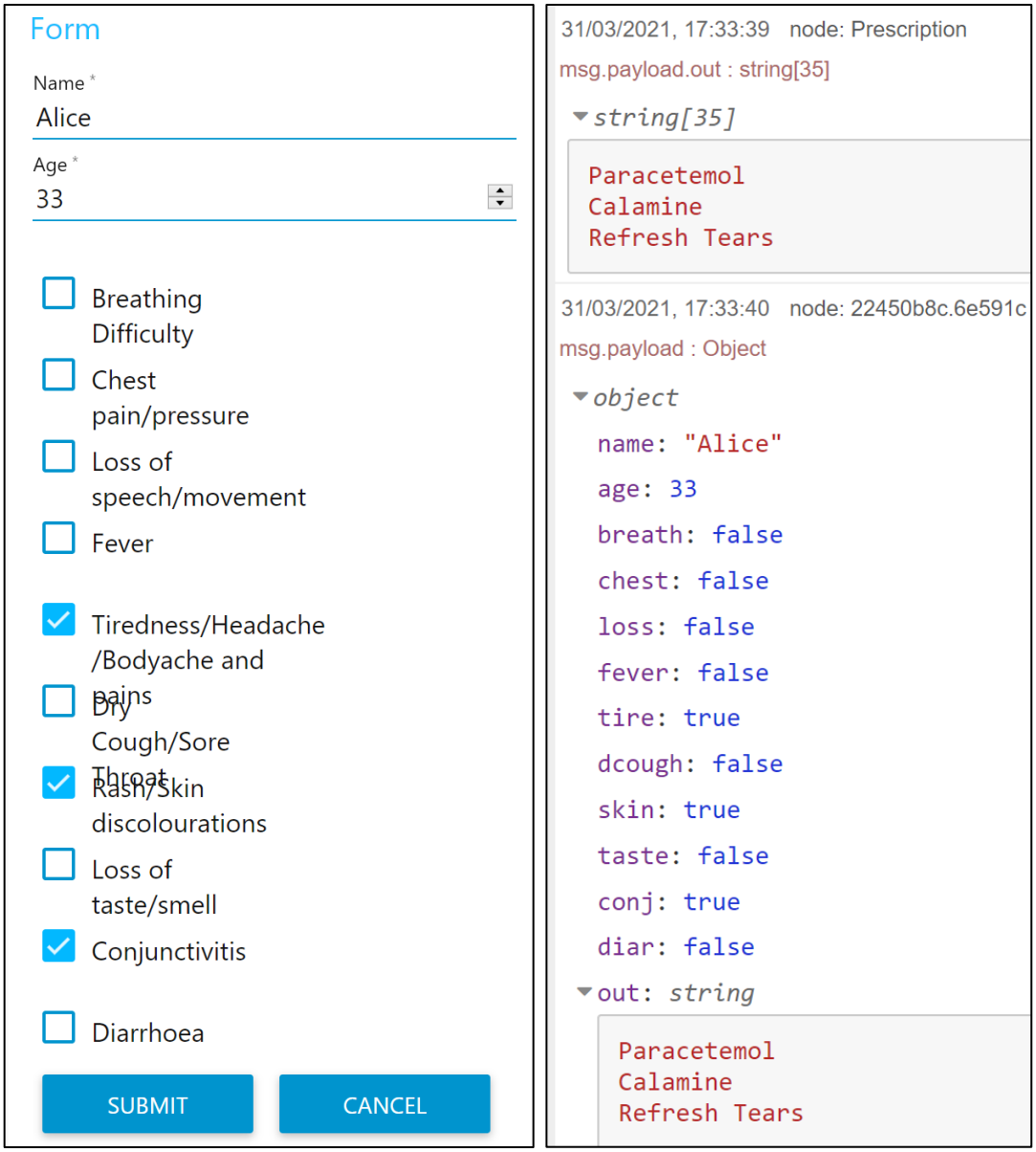


Output:

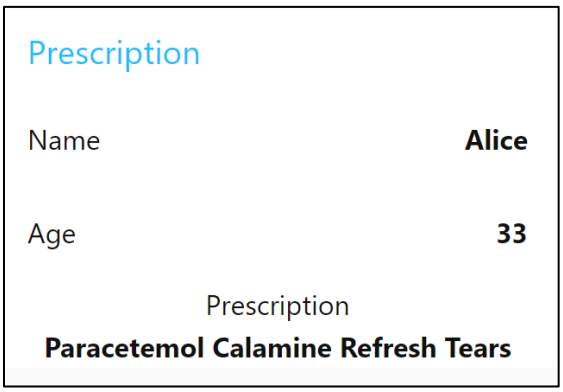
1. Less severe symptoms

Client-side Patient Info

Server- side Debug Window

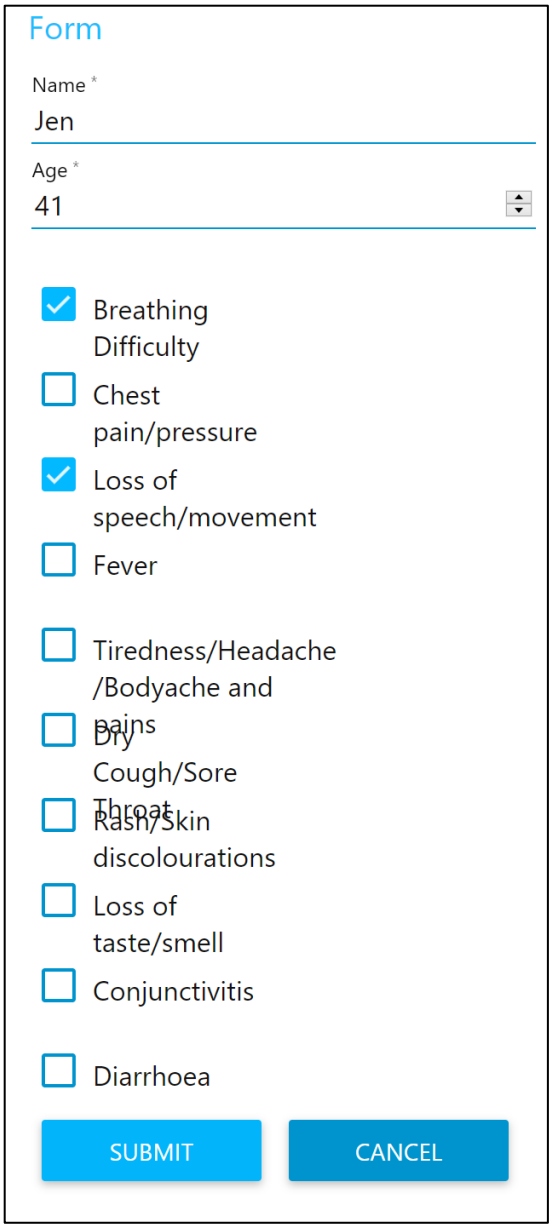


Server-side Doctor's decision

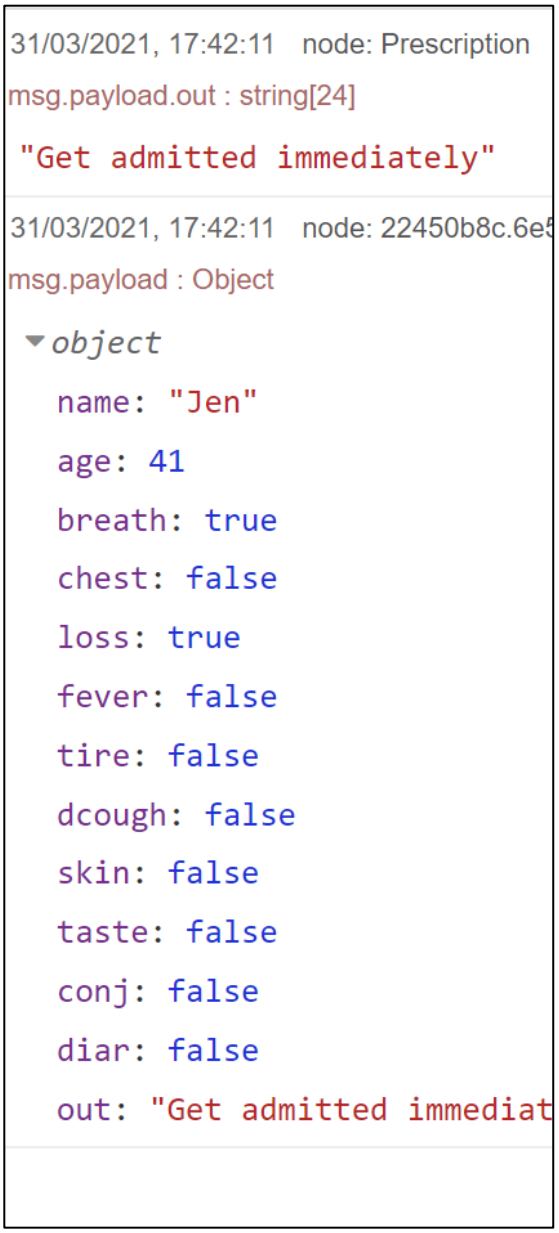


2. Severe Symptoms

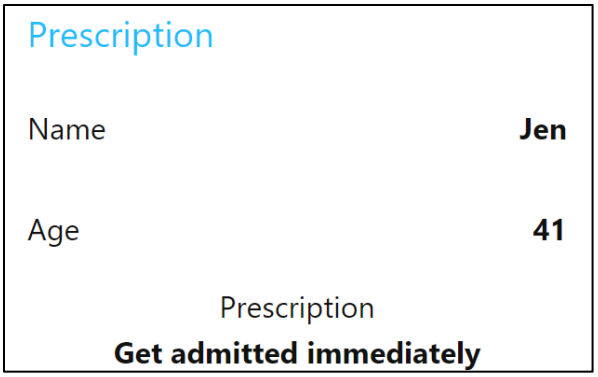
Client-side Patient Info



Server- side Debug Window



Server-side Doctor's decision



Inference: The form node is used in the client side to get patient details and symptoms, which are sent to the server-side using FRED out. Then, on the server-side, the symptoms' information is extracted through the FRED in node and a prescription is formed. For less severe symptoms, medicines are prescribed by the doctor. For the serious symptoms, the patient is advised to get admitted.

Results: Hence patient symptom info is collected and doctor's decision is displayed in dashboard and debug window using node-red and FRED.

Experiment 9

KNIME Analytics Platform – Sales Analytics

Aim: To do Sales Analysis on KNIME Analytics Platform and visualize the data with Stacked Area Chart and Pie Chart.

Software Used: KNIME, the Konstanz Information Miner (https://www.knime.com/).

Theory:

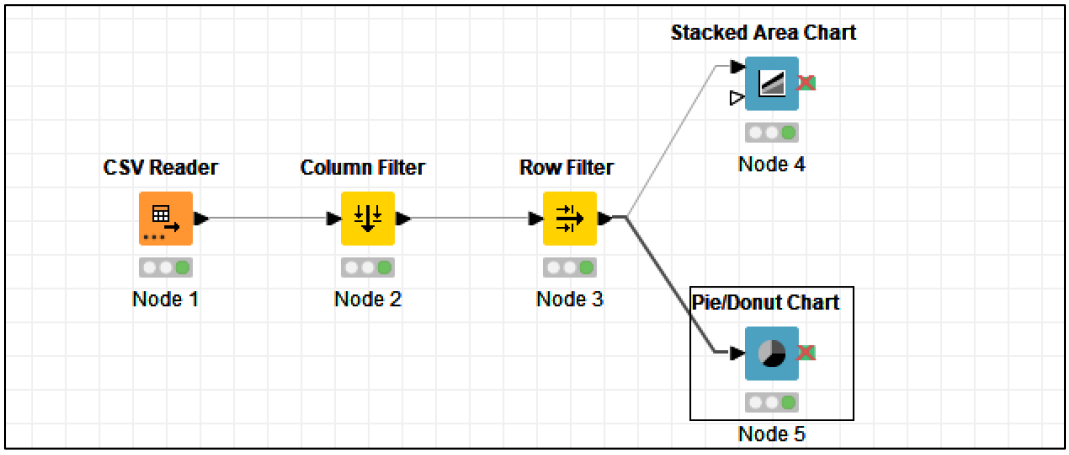
About KNIME:

- KNIME Analytics Platform is a free, open-source software used in data science.
- It is helpful in discovering potentials hidden in the data, mining for fresh insights, or predicting new features.

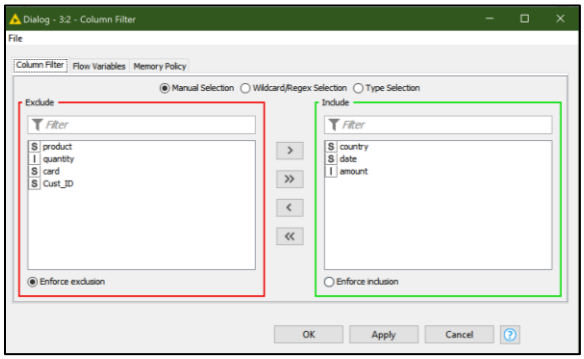
Dataset Used:

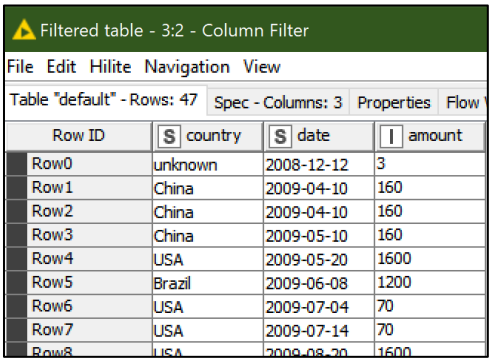
- sales_data.csv file used for Sales Analytics
- (https://files.knime.com/sites/default/files/sales_data.csv)

Workflow:



Configurations and Outputs:



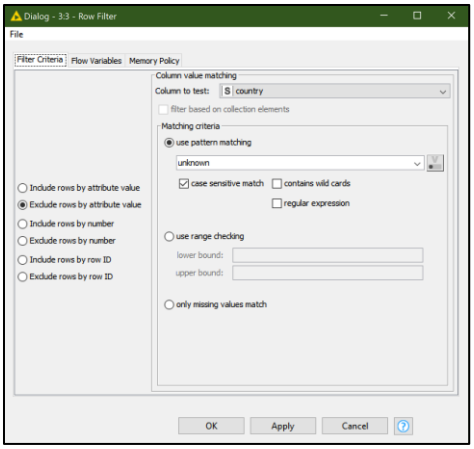


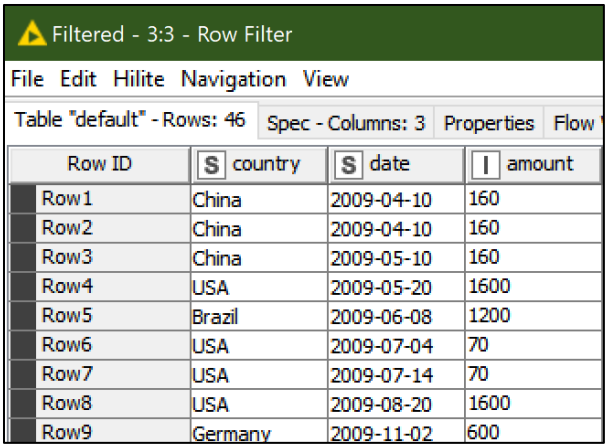
Filtered table - 3:2 - Column Filter

File Edit Hilite Navigation View

Table "default" - Rows: 47 | Spec - Columns: 3 | Properties | Flow

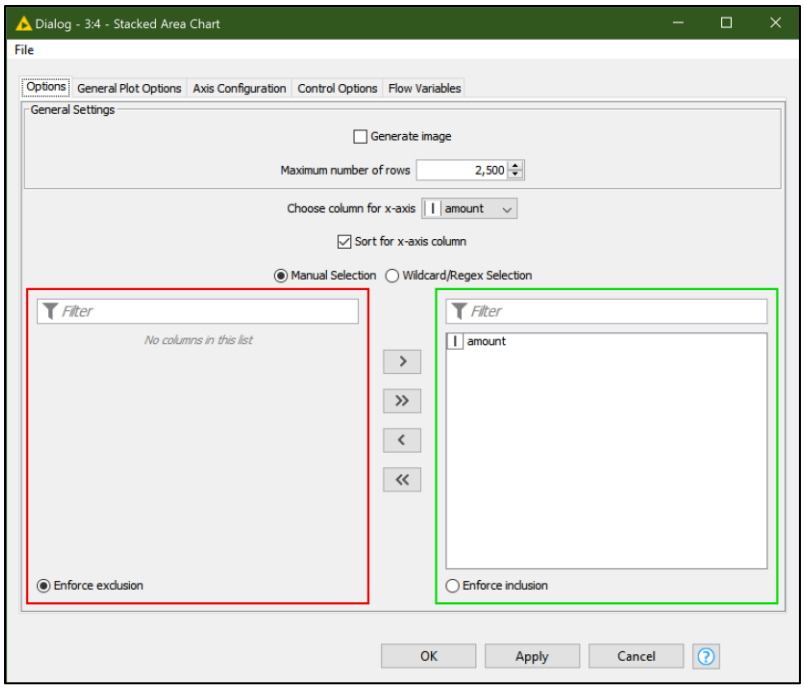
Row ID	country	date	amount
Row0	unknown	2008-12-12	3
Row1	China	2009-04-10	160
Row2	China	2009-04-10	160
Row3	China	2009-05-10	160
Row4	USA	2009-05-20	1600
Row5	Brazil	2009-06-08	1200
Row6	USA	2009-07-04	70
Row7	USA	2009-07-14	70
Row8	USA	2009-08-20	1600

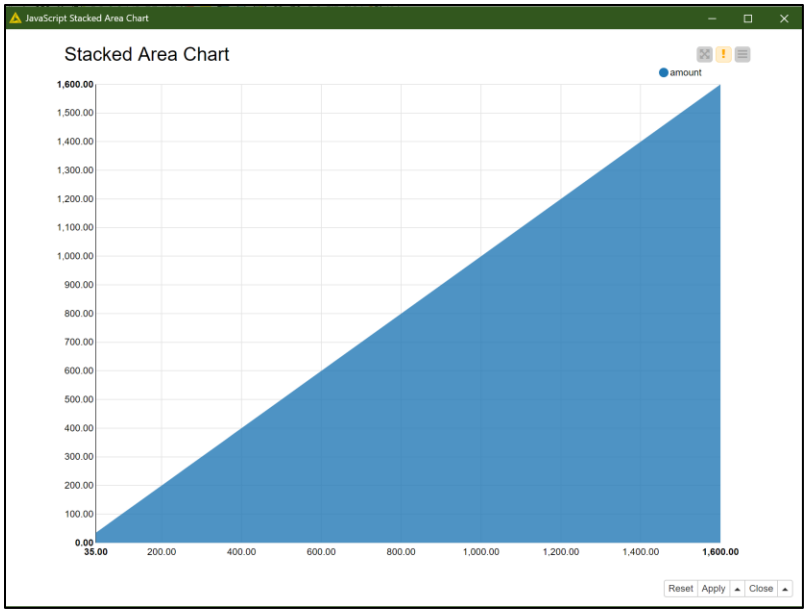




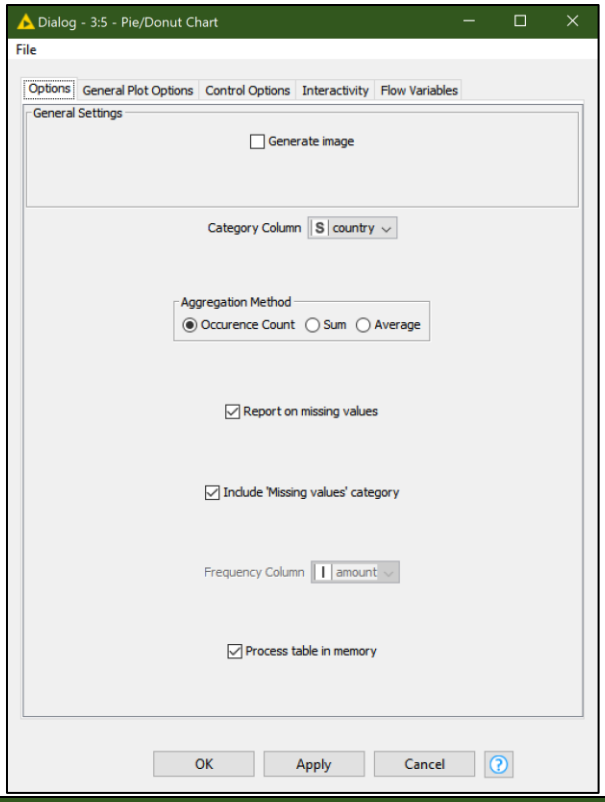
Row ID	country	date	amount
Row1	China	2009-04-10	160
Row2	China	2009-04-10	160
Row3	China	2009-05-10	160
Row4	USA	2009-05-20	1600
Row5	Brazil	2009-06-08	1200
Row6	USA	2009-07-04	70
Row7	USA	2009-07-14	70
Row8	USA	2009-08-20	1600
Row9	Germany	2009-11-02	600

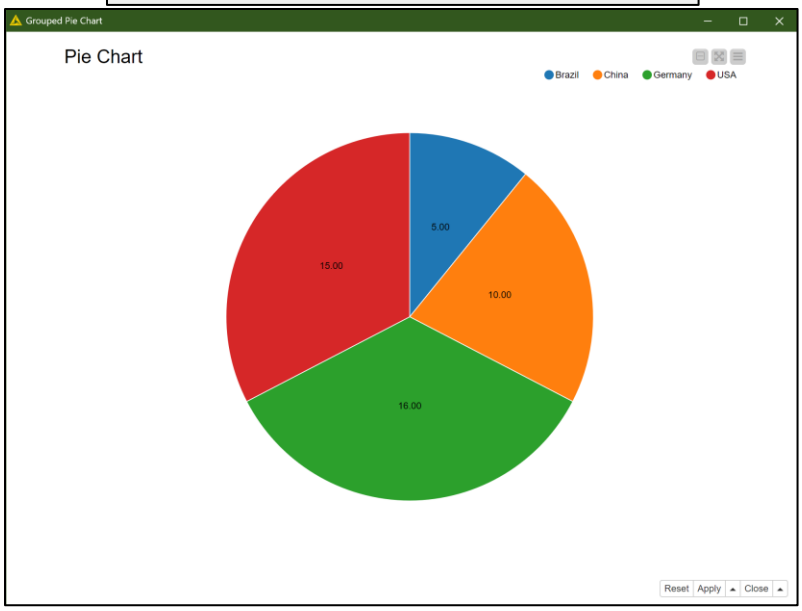
Stacked Area Chart:




JavaScript Stacked Area Chart
Stacked Area Chart
amount
1,600.00
1,500.00
1,400.00
1,300.00
1,200.00
1,100.00
1,000.00
900.00
800.00
700.00
600.00
500.00
400.00
300.00
200.00
100.00
0.00
35.00
200.00
400.00
600.00
800.00
1,000.00
1,200.00
1,400.00
1,600.00
Reset
Apply
Close

Pie Chart:





Inference: Knime aids in pre-processing the data using column and row filters, removing redundant or unwanted data (data cleaning) and visualizing the data via graphs

Result: Thus, we have performed Sales Analysis on Knime Analytics Platform and visualized the data with Stacked Area Chart and Pie Chart and observed the outcome successfully.

Experiment 10

Data Manipulation in Knime: String Manipulation, Math Formula and Rule Engine

Aim: To do Sales Analysis and Data Manipulation on Knime Analytics Platform and analyze the data with Rule Engine, Math Formula and String Manipulation.

Software Used: KNIME, the Konstanz Information Miner (https://www.knime.com/).

Theory:

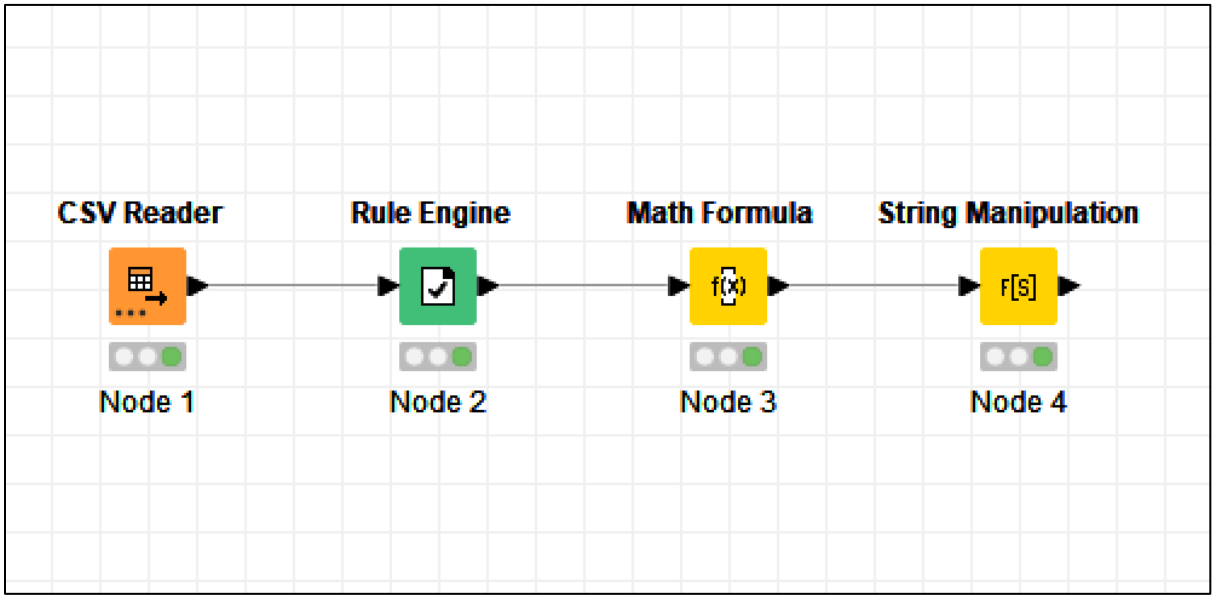
About KNIME: KNIME Analytics Platform is a free, open-source software used in data science. It is helpful in discovering potentials hidden in the data, mining for fresh insights, or predicting new features.

Data Manipulation Nodes:

- Rule Engine Node: This node takes a list of user-defined rules and tries to match them to each row in the input table. If a rule matches, its outcome value is added into a new column. The first matching rule in order of definition determines the outcome.
- Math Formula Node: This node evaluates a mathematical expression based on the values in a row. The computed results can be either appended as a new column or be used to replace an input column. This node uses JEP, the Java Math Expression Parser.
- String Manipulation Node: Manipulates strings like search and replace, capitalize or remove leading and trailing white spaces.

Dataset used: sales_data.csv file used for Sales Analytics

Flow:



Configurations:

CSV Reader:

File Table - 3:1 - CSV Reader

File Edit Hilite Navigation View

Table "default" - Rows: 47 | Spec - Columns: 7 | Properties | Flow Variables

Row ID	product	country	date	quantity	amount	card	Cust_ID
Row0	prod_4	unknown	2008-12-12	1	3	?	Cust_8
Row1	prod_3	China	2009-04-10	2	160	N	Cust_2
Row2	prod_3	China	2009-04-10	2	160	Y	Cust_5
Row3	prod_3	China	2009-05-10	2	160	?	Cust_2
Row4	prod_3	USA	2009-05-20	20	1600	?	Cust_3
Row5	prod_3	Brazil	2009-06-08	15	1200	?	Cust_7
Row6	prod_1	USA	2009-07-04	2	70	Y	Cust_3
Row7	prod_1	USA	2009-07-14	2	70	?	Cust_6
Row8	prod_3	USA	2009-08-20	20	1600	?	Cust_3
Row9	prod_2	Germany	2009-11-02	15	600	?	Cust_1
Row10	prod_2	Germany	2009-11-22	15	600	N	Cust_1
Row11	prod_1	Germany	2009-12-02	1	35	Y	Cust_1
Row12	prod_1	China	2009-12-12	1	35	Y	Cust_2
Row13	prod_3	USA	2010-01-03	20	1600	?	Cust_3

Rule Engine:

Expression

```
1 // enter ordered set of rules, e.g.:
2 // $double column name$ > 5.0 => "large"
3 // $string column name$ LIKE "*blue*" => "small and blue"
4 // TRUE => "default outcome"
5 $quantity$ =1 => "Need based customer"
6 $quantity$ >=10 => "Impulse customer"
7 TRUE => "Loyal Customer"
```

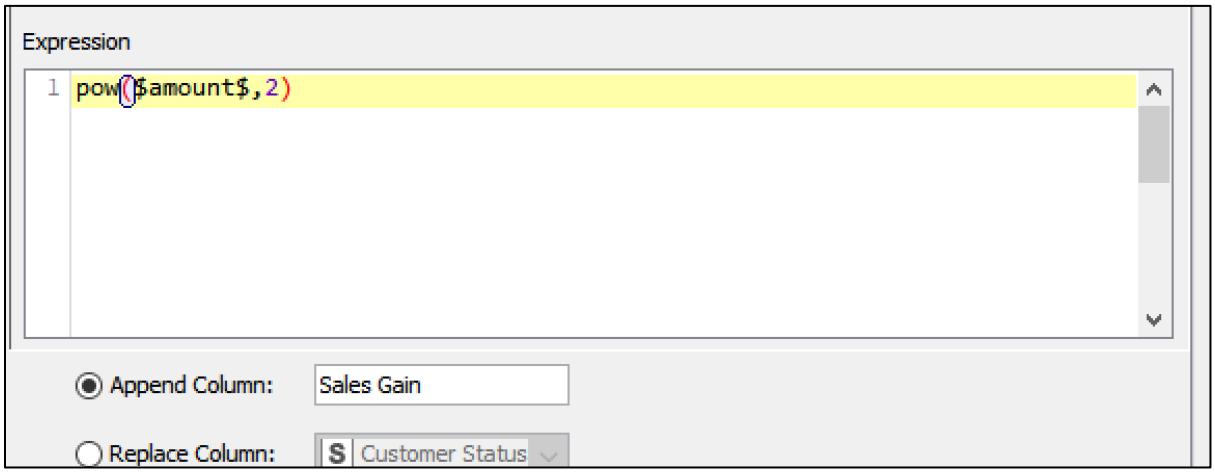
Classified values - 3:2 - Rule Engine

File Edit Hilite Navigation View

Table "default" - Rows: 47 | Spec - Columns: 8 | Properties | Flow Variables

Row ID	product	country	date	quantity	amount	card	Cust_ID	Customer Status
Row0	prod_4	unknown	2008-12-12	1	3	?	Cust_8	Need based customer
Row1	prod_3	China	2009-04-10	2	160	N	Cust_2	Loyal Customer
Row2	prod_3	China	2009-04-10	2	160	Y	Cust_5	Loyal Customer
Row3	prod_3	China	2009-05-10	2	160	?	Cust_2	Loyal Customer
Row4	prod_3	USA	2009-05-20	20	1600	?	Cust_3	Impulse customer
Row5	prod_3	Brazil	2009-06-08	15	1200	?	Cust_7	Impulse customer
Row6	prod_1	USA	2009-07-04	2	70	Y	Cust_3	Loyal Customer
Row7	prod_1	USA	2009-07-14	2	70	?	Cust_6	Loyal Customer
Row8	prod_3	USA	2009-08-20	20	1600	?	Cust_3	Impulse customer
Row9	prod_2	Germany	2009-11-02	15	600	?	Cust_1	Impulse customer
Row10	prod_2	Germany	2009-11-22	15	600	N	Cust_1	Impulse customer
Row11	prod_1	Germany	2009-12-02	1	35	Y	Cust_1	Need based customer
Row12	prod_1	China	2009-12-12	1	35	Y	Cust_2	Need based customer
Row13	prod_3	USA	2010-01-03	20	1600	?	Cust_3	Impulse customer
Row14	prod_1	Germany	2010-01-10	1	35	N	Cust_1	Need based customer
Row15	prod_3	Germany	2010-01-13	1	80	?	Cust_4	Need based customer
Row16	prod_2	Germany	2010-01-15	25	1000	?	Cust_1	Impulse customer

Math Formula Node:



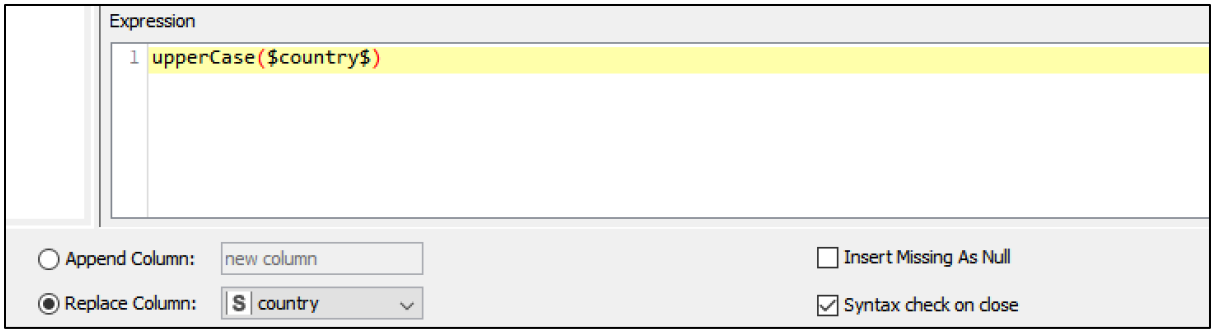
Output data - 3:3 - Math Formula

File Edit Hilite Navigation View

Table "default" - Rows: 47 | Spec - Columns: 9 | Properties | Flow Variables

Row ID	product	country	date	quantity	amount	card	Cust_ID	Customer Status	Sales Gain
Row0	prod_4	unknown	2008-12-12	1	3	?	Cust_8	Need based customer	9
Row1	prod_3	China	2009-04-10	2	160	N	Cust_2	Loyal Customer	25,600
Row2	prod_3	China	2009-04-10	2	160	Y	Cust_5	Loyal Customer	25,600
Row3	prod_3	China	2009-05-10	2	160	?	Cust_2	Loyal Customer	25,600
Row4	prod_3	USA	2009-05-20	20	1600	?	Cust_3	Impulse customer	2,560,000
Row5	prod_3	Brazil	2009-06-08	15	1200	?	Cust_7	Impulse customer	1,440,000
Row6	prod_1	USA	2009-07-04	2	70	Y	Cust_3	Loyal Customer	4,900
Row7	prod_1	USA	2009-07-14	2	70	?	Cust_6	Loyal Customer	4,900
Row8	prod_3	USA	2009-08-20	20	1600	?	Cust_3	Impulse customer	2,560,000
Row9	prod_2	Germany	2009-11-02	15	600	?	Cust_1	Impulse customer	360,000
Row10	prod_2	Germany	2009-11-22	15	600	N	Cust_1	Impulse customer	360,000
Row11	prod_1	Germany	2009-12-02	1	35	Y	Cust_1	Need based customer	1,225

String Manipulation Node:



Appended table - 3:4 - String Manipulation

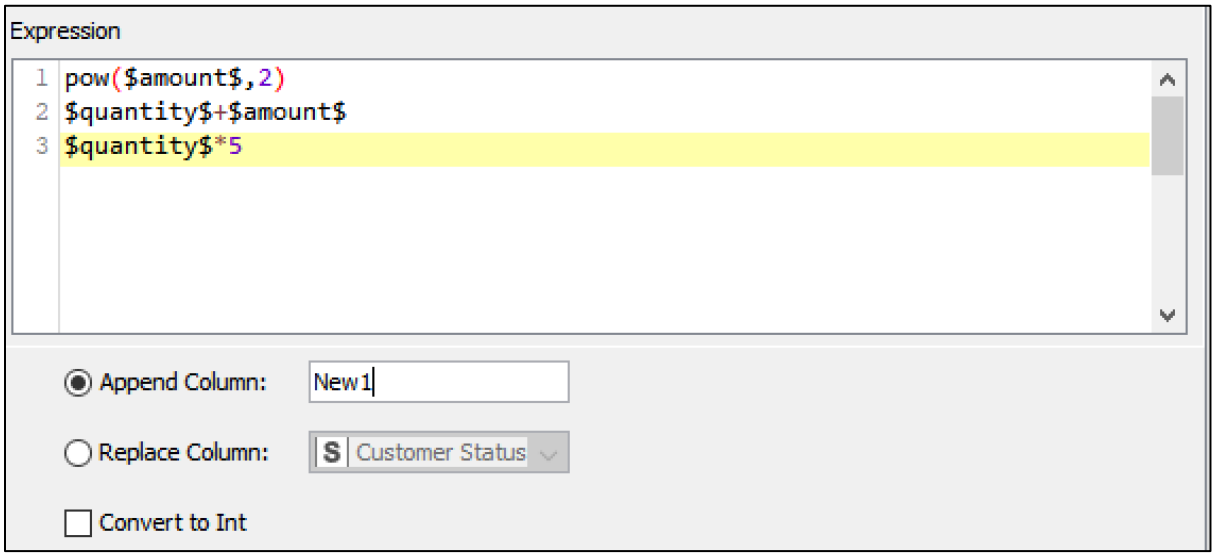
File Edit Hilite Navigation View

Table "default" - Rows: 47 | Spec - Columns: 9 | Properties | Flow Variables

Row ID	product	country	date	quantity	amount	card	Cust_ID	Customer Status	Sales Gain
Row0	prod_4	UNKNOWN	2008-12-12	1	3	?	Cust_8	Need based customer	9
Row1	prod_3	CHINA	2009-04-10	2	160	N	Cust_2	Loyal Customer	25,600
Row2	prod_3	CHINA	2009-04-10	2	160	Y	Cust_5	Loyal Customer	25,600
Row3	prod_3	CHINA	2009-05-10	2	160	?	Cust_2	Loyal Customer	25,600
Row4	prod_3	USA	2009-05-20	20	1600	?	Cust_3	Impulse customer	2,560,000
Row5	prod_3	BRAZIL	2009-06-08	15	1200	?	Cust_7	Impulse customer	1,440,000
Row6	prod_1	USA	2009-07-04	2	70	Y	Cust_3	Loyal Customer	4,900
Row7	prod_1	USA	2009-07-14	2	70	?	Cust_6	Loyal Customer	4,900
Row8	prod_3	USA	2009-08-20	20	1600	?	Cust_3	Impulse customer	2,560,000
Row9	prod_2	GERMANY	2009-11-02	15	600	?	Cust_1	Impulse customer	360,000
Row10	prod_2	GERMANY	2009-11-22	15	600	N	Cust_1	Impulse customer	360,000
Row11	prod_1	GERMANY	2009-12-02	1	35	Y	Cust_1	Need based customer	1,225
Row12	prod_1	CHINA	2009-12-12	1	35	Y	Cust_2	Need based customer	1,225

Task:

Math Formula:



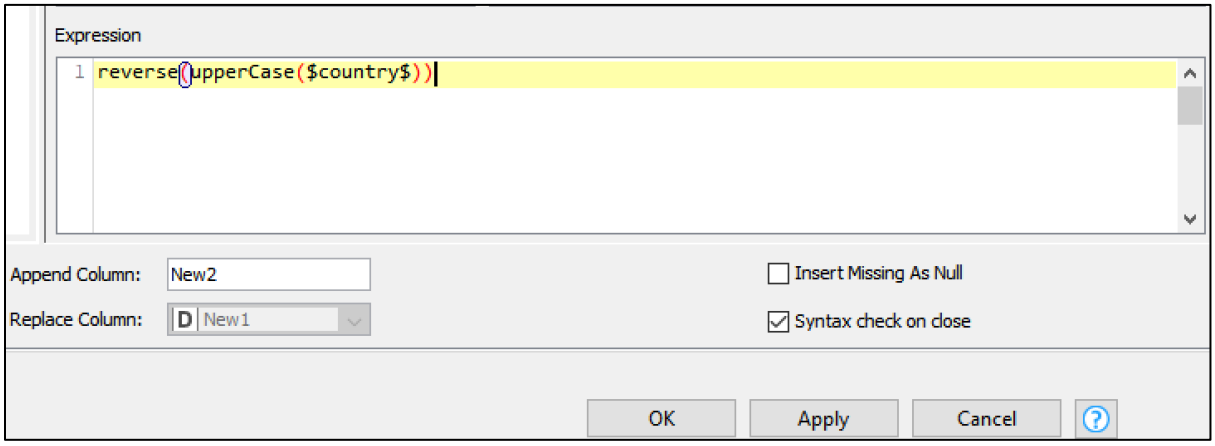
Output data - 3:3 - Math Formula

File Edit Hilite Navigation View

Table "default" - Rows: 47 | Spec - Columns: 9 | Properties | Flow Variables

Row ID	product	country	date	quantity	amount	card	Cust_ID	Customer Status	New1
Row0	prod_4	unknown	2008-12-12	1	3	?	Cust_8	Need based customer	24
Row1	prod_3	China	2009-04-10	2	160	N	Cust_2	Loyal Customer	52,800
Row2	prod_3	China	2009-04-10	2	160	Y	Cust_5	Loyal Customer	52,800
Row3	prod_3	China	2009-05-10	2	160	?	Cust_2	Loyal Customer	52,800
Row4	prod_3	USA	2009-05-20	20	1600	?	Cust_3	Impulse customer	51,360,000
Row5	prod_3	Brazil	2009-06-08	15	1200	?	Cust_7	Impulse customer	21,690,000
Row6	prod_1	USA	2009-07-04	2	70	Y	Cust_3	Loyal Customer	10,500
Row7	prod_1	USA	2009-07-14	2	70	?	Cust_6	Loyal Customer	10,500
Row8	prod_3	USA	2009-08-20	20	1600	?	Cust_3	Impulse customer	51,360,000
Row9	prod_2	Germany	2009-11-02	15	600	?	Cust_1	Impulse customer	5,445,000
Row10	prod_2	Germany	2009-11-22	15	600	N	Cust_1	Impulse customer	5,445,000
Row11	prod_1	Germany	2009-12-02	1	35	Y	Cust_1	Need based customer	1,400
Row12	prod_1	China	2009-12-12	1	35	Y	Cust_2	Need based customer	1,400
Row13	prod_3	USA	2010-01-03	20	1600	?	Cust_3	Impulse customer	51,360,000
Row14	prod_1	Germany	2010-01-10	1	35	N	Cust_1	Need based customer	1,400

String Manipulation:



Appended table - 3:4 - String Manipulation

File Edit Hilite Navigation View

Table "default" - Rows: 47 | Spec - Columns: 10 | Properties | Flow Variables

Row ID	product	country	date	quantity	amount	card	Cust_ID	Customer Status	New1	New2
Row0	prod_4	unknown	2008-12-12	1	3	?	Cust_8	Need based customer	24	NWONKNU
Row1	prod_3	China	2009-04-10	2	160	N	Cust_2	Loyal Customer	52,800	ANIHC
Row2	prod_3	China	2009-04-10	2	160	Y	Cust_5	Loyal Customer	52,800	ANIHC
Row3	prod_3	China	2009-05-10	2	160	?	Cust_2	Loyal Customer	52,800	ANIHC
Row4	prod_3	USA	2009-05-20	20	1600	?	Cust_3	Impulse customer	51,360,000	ASU
Row5	prod_3	Brazil	2009-06-08	15	1200	?	Cust_7	Impulse customer	21,690,000	LIZARB
Row6	prod_1	USA	2009-07-04	2	70	Y	Cust_3	Loyal Customer	10,500	ASU
Row7	prod_1	USA	2009-07-14	2	70	?	Cust_6	Loyal Customer	10,500	ASU
Row8	prod_3	USA	2009-08-20	20	1600	?	Cust_3	Impulse customer	51,360,000	ASU
Row9	prod_2	Germany	2009-11-02	15	600	?	Cust_1	Impulse customer	5,445,000	YNAMREG
Row10	prod_2	Germany	2009-11-22	15	600	N	Cust_1	Impulse customer	5,445,000	YNAMREG
Row11	prod_1	Germany	2009-12-02	1	35	Y	Cust_1	Need based customer	1,400	YNAMREG
Row12	prod_1	China	2009-12-12	1	35	Y	Cust_2	Need based customer	1,400	ANIHC
Row13	prod_3	USA	2010-01-03	20	1600	?	Cust_3	Impulse customer	51,360,000	ASU
Row14	prod_1	Germany	2010-01-10	1	35	N	Cust_1	Need based customer	1,400	YNAMREG
Row15	prod_3	Germany	2010-01-13	1	80	?	Cust_4	Need based customer	6,800	YNAMREG
Row16	prod_2	Germany	2010-01-15	25	1000	?	Cust_1	Impulse customer	25,125,000	YNAMREG

Inference: Rule Engine node is used to add columns based on user-defined rules. Math Formula node is used to execute mathematical expressions on the data, and String Manipulation node is used to manipulate string values in the dataset.

Result: Thus, we have performed sales analysis and data manipulation on Knime Analytics Platform and analysed the data with Rule Engine, Math Formula and String Manipulation successfully.

Experiment 11

K Means Clustering in Knime: Unsupervised Machine Learning

Aim: To cluster data and classify them under respective features on Knime Analytics Platform with the help of Color Manger & Shape Manager and analyze the data using k-Means cluster node and Cluster assigner node.

Software Used: KNIME, the Konstanz Information Miner (https://www.knime.com/).

Theory:

About KNIME: KNIME Analytics Platform is a free, open-source software used in data science. It is helpful in discovering potentials hidden in the data, mining for fresh insights, or predicting new features.

Clustering:

- Unsupervised Learning.
- Requires data, but no labels.
- Detect patterns e.g., in
    - Group emails or search results.
    - Customer shopping patterns.
    - Region of images.
- Useful when you don't know what you are looking for.
- Basic Idea: Group Together similar instances.
- Example: 2D point patterns.

k-Means Algorithm:

- An iterative clustering algorithm.
- Initialize: Pick k random points as cluster centers.
- Alternate: (1) Assign data points to the closest cluster center. (2) Change the cluster
- center to the average of its assigned points.
- Stop when no points assignment changes.

Nodes:

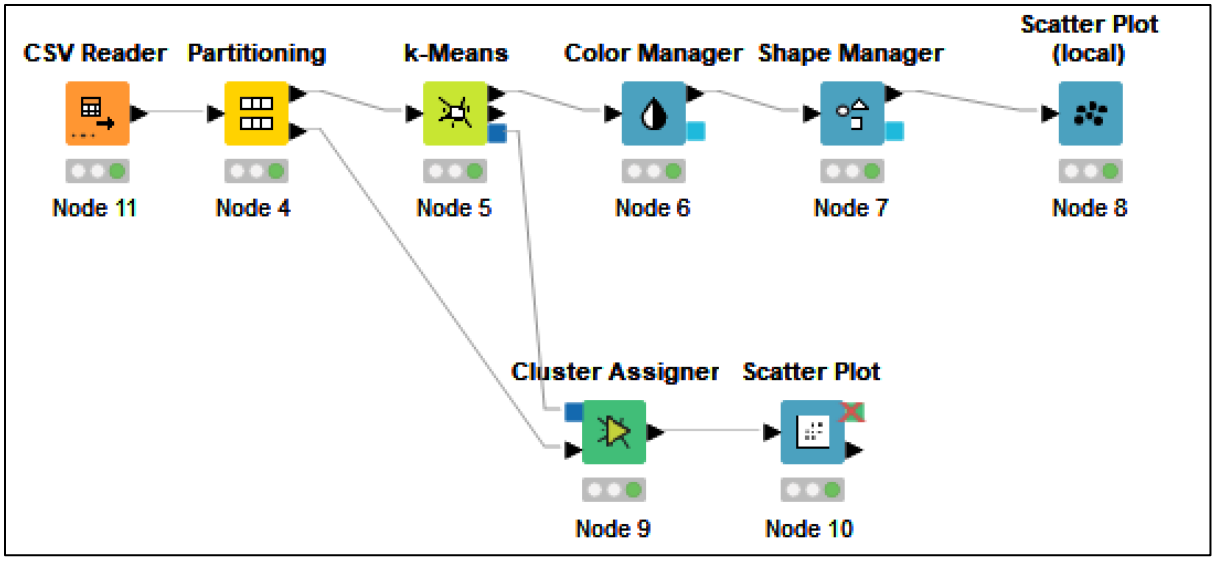
- k-Means Node:
    - This node outputs the cluster centers for a predefined number of clusters (no
    - dynamic number of clusters).
    - K-means performs a crisp clustering that assigns a data vector to exactly one cluster.
    - The Algorithm terminates when the cluster assignments do not change anymore.
    - The clustering algorithm uses the Euclidean distance on the selected attributes.
    - The data is not normalized by the node.

- Cluster Assigner Node:
    - This node assigns new data to an existing set of prototypes, which are obtained e.g., by a k-means clustering.
    - Each data point is assigned to its nearest prototype.
- Color Manager Node:
    - Colors can be assigned for either nominal (possible values have to be available) or numeric columns (with lower and upper bounds).
- Shape Manager Node:
    - Assigns (different) shapes for each attribute value of one nominal column, i.e., for each possible value. Supporting views then render data points with the shape associated with the corresponding attribute value.
- Scatter Plot Node (local):
    - Creates a scatterplot of two selectable attributes. Then each datapoint is displayed as a dot at its corresponding place, dependent on its values of the selected attributes.
    - The dots are displayed in the color defined by the Color Manager, the size defined by the Size Manager, and the shape defined by the Shape Manager.
- Scatter Plot:
    - A scatter plot using a JavaScript based charting library. The view can be accessed either via the "interactive view" action on the executed node or in KNIME Server web portal page.
    - The configuration of the node lets you choose the size of a sample to display and to enable certain controls, which are then available in the view.

Dataset Used: The Iris dataset consists of 50 samples from each of three classes of iris flowers. There are five attributes in the dataset:

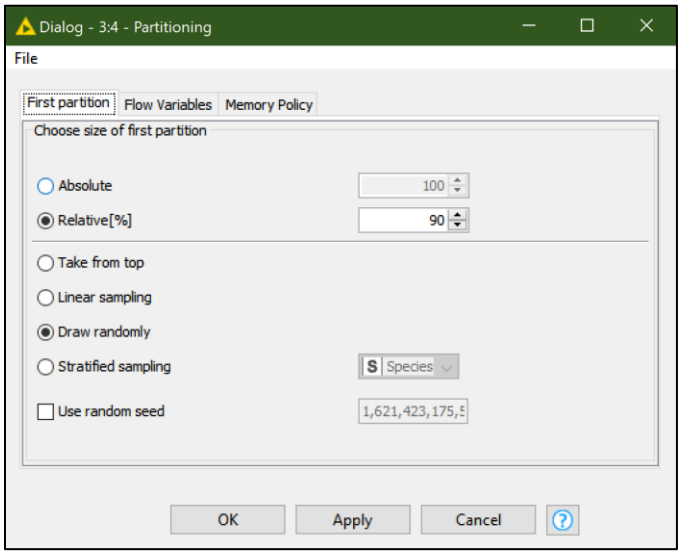
- sepal length in cm,
- sepal width in cm,
- petal length in cm,
- petal width in cm, and
- class: Iris Setosa, Iris Versicolour, and Iris Virginica.

Flow:

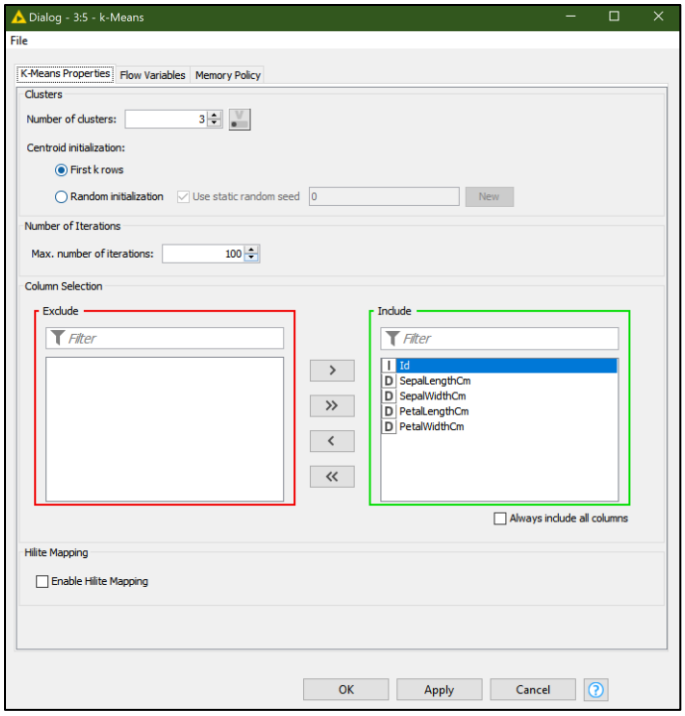


Configurations:

Partitioning:



K-Means:



Clusters - 3:5 - k-Means

File Edit Hilite Navigation View

Table "default" - Rows: 3 | Spec - Columns: 5 | Properties | Flow Variables

Row ID	D Id	D SepalLe...	D SepalW...	D PetalLe...	D PetalWi...
cluster_0	24.674	5	3.405	1.467	0.253
cluster_1	74.864	5.961	2.768	4.23	1.295
cluster_2	124.646	6.552	2.981	5.517	2.004

Labeled input - 3:5 - k-Means

File Edit Hilite Navigation View

Table "default" - Rows: 135 | Spec - Columns: 7 | Properties | Flow Variables

Row ID	I Id	D SepalLe...	D SepalW...	D PetalLe...	D PetalWi...	S Species	S Cluster
Row0	1	5.1	3.5	1.4	0.2	Iris-setosa	cluster_0
Row1	2	4.9	3	1.4	0.2	Iris-setosa	cluster_0
Row2	3	4.7	3.2	1.3	0.2	Iris-setosa	cluster_0
Row3	4	4.6	3.1	1.5	0.2	Iris-setosa	cluster_0
Row4	5	5	3.6	1.4	0.2	Iris-setosa	cluster_0
Row5	6	5.4	3.9	1.7	0.4	Iris-setosa	cluster_0
Row6	7	4.6	3.4	1.4	0.3	Iris-setosa	cluster_0
Row7	8	5	3.4	1.5	0.2	Iris-setosa	cluster_0
Row9	10	4.9	3.1	1.5	0.1	Iris-setosa	cluster_0
Row10	11	5.4	3.7	1.5	0.2	Iris-setosa	cluster_0
Row11	12	4.8	3.4	1.6	0.2	Iris-setosa	cluster_0

Color Manager:

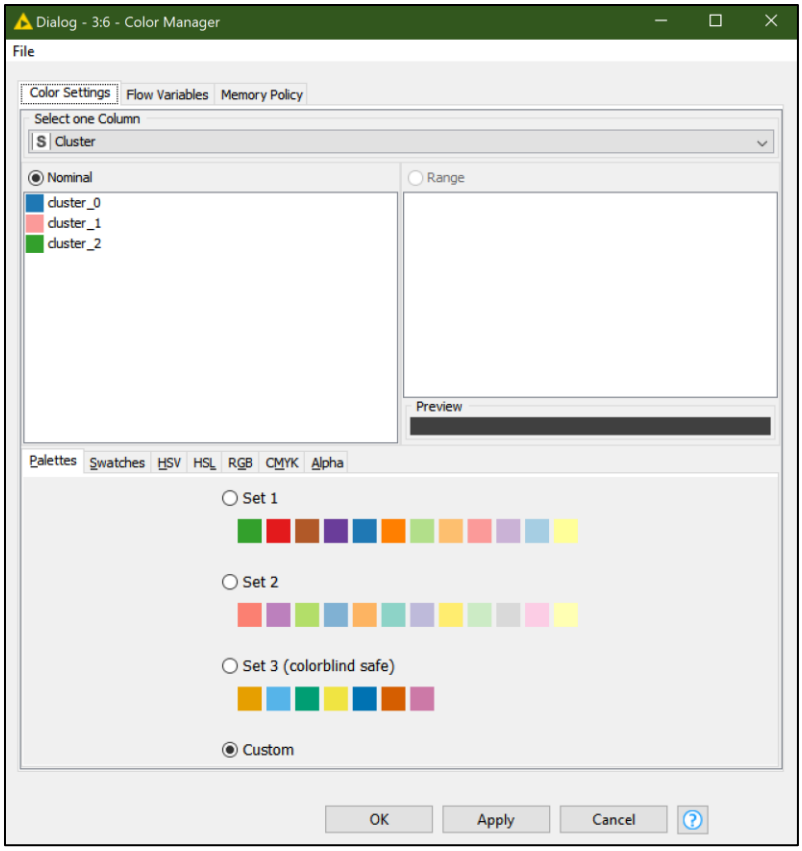


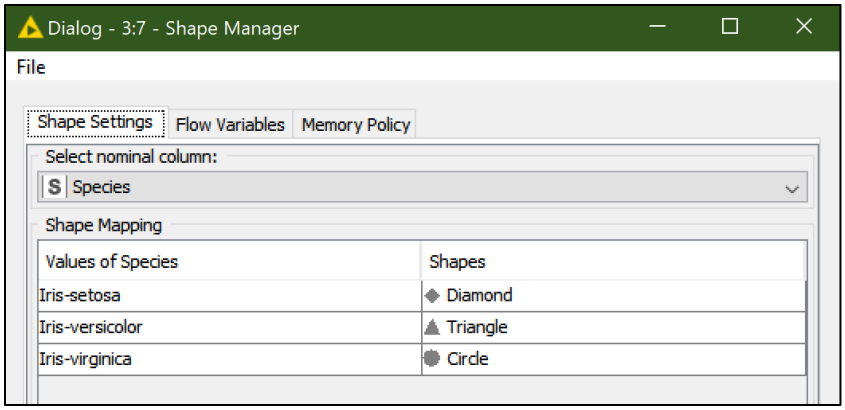
Table with Colors - 3:6 - Color Manager

File Edit Hilite Navigation View

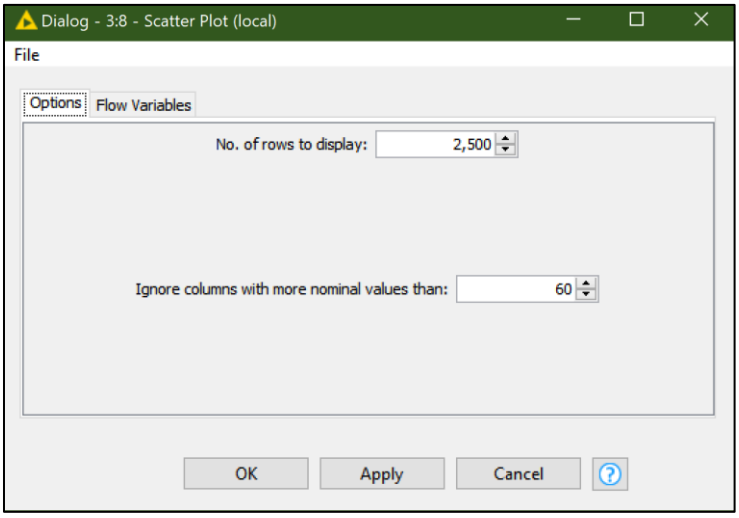
Table "default" - Rows: 135 | Spec - Columns: 7 | Properties | Flow Variables

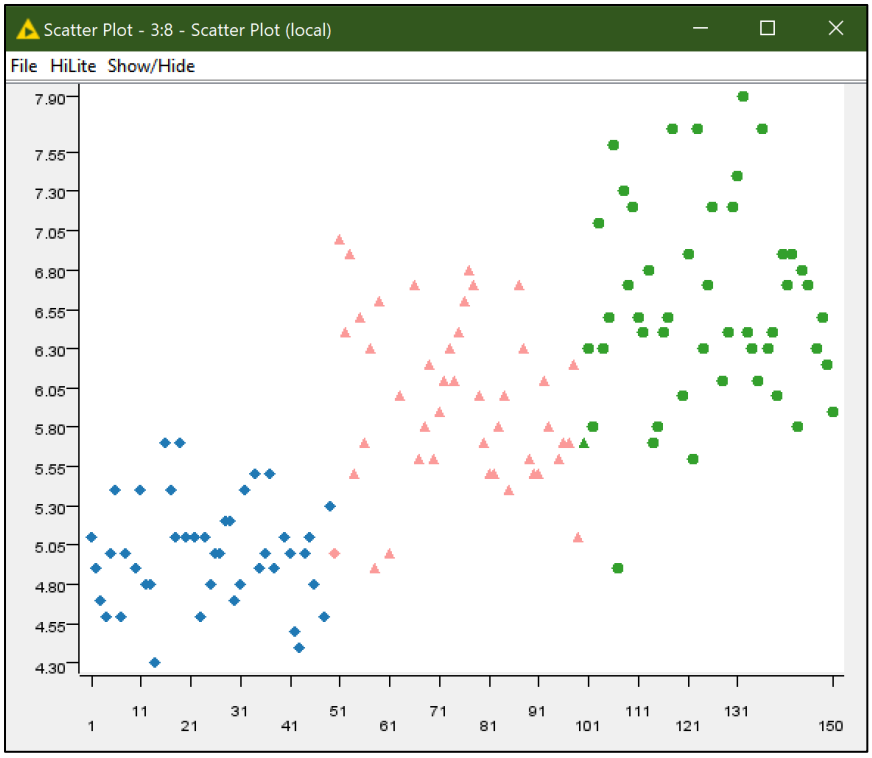
Row ID	Id	SepalLe...	SepalW...	PetalLe...	PetalWi...	Species	Cluster
Row41	42	4.5	2.3	1.3	0.3	Iris-setosa	cluster_0
Row42	43	4.4	3.2	1.3	0.2	Iris-setosa	cluster_0
Row43	44	5	3.5	1.6	0.6	Iris-setosa	cluster_0
Row44	45	5.1	3.8	1.9	0.4	Iris-setosa	cluster_0
Row45	46	4.8	3	1.4	0.3	Iris-setosa	cluster_0
Row47	48	4.6	3.2	1.4	0.2	Iris-setosa	cluster_0
Row48	49	5.3	3.7	1.5	0.2	Iris-setosa	cluster_0
Row49	50	5	3.3	1.4	0.2	Iris-setosa	cluster_1
Row50	51	7	3.2	4.7	1.4	Iris-versicolor	cluster_1
Row51	52	6.4	3.2	4.5	1.5	Iris-versicolor	cluster_1
Row52	53	6.9	3.1	4.9	1.5	Iris-versicolor	cluster_1
Row53	54	5.5	2.3	4	1.3	Iris-versicolor	cluster_1

Shape Manager:

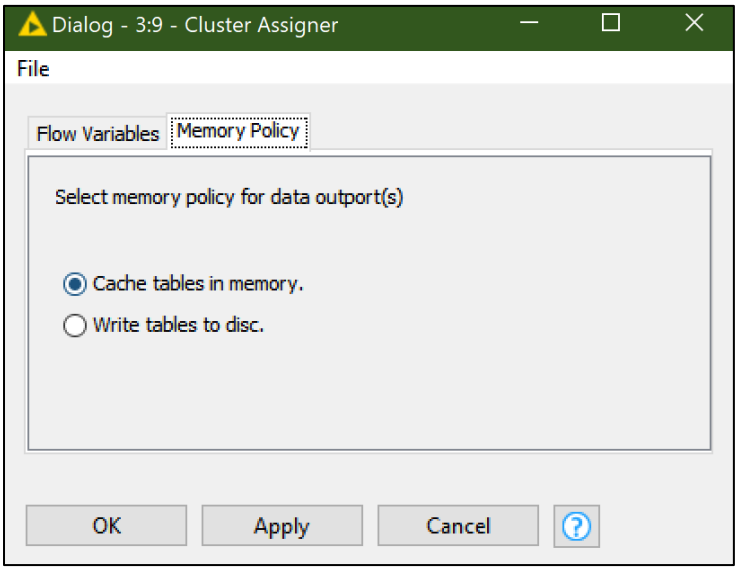


Scatter Plot (Local):





Cluster Assigner:



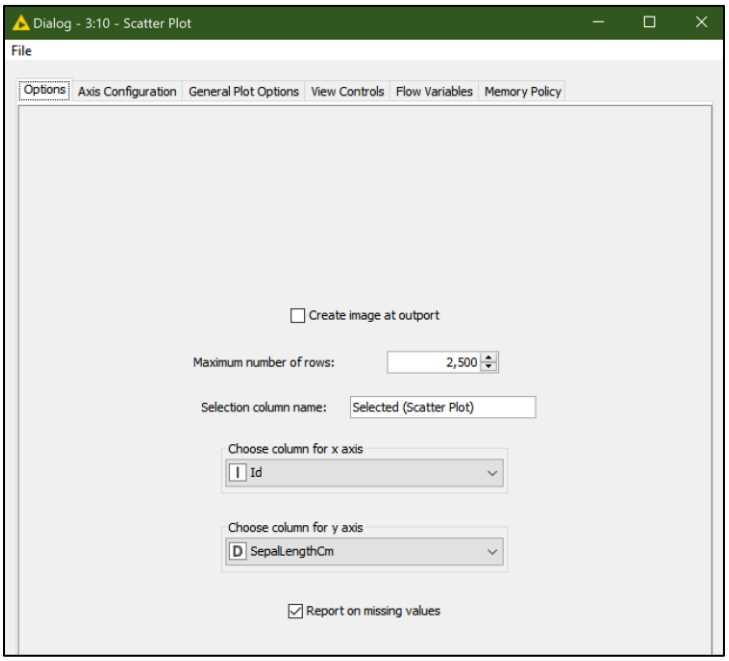
Assigned Data - 3:9 - Cluster Assigner

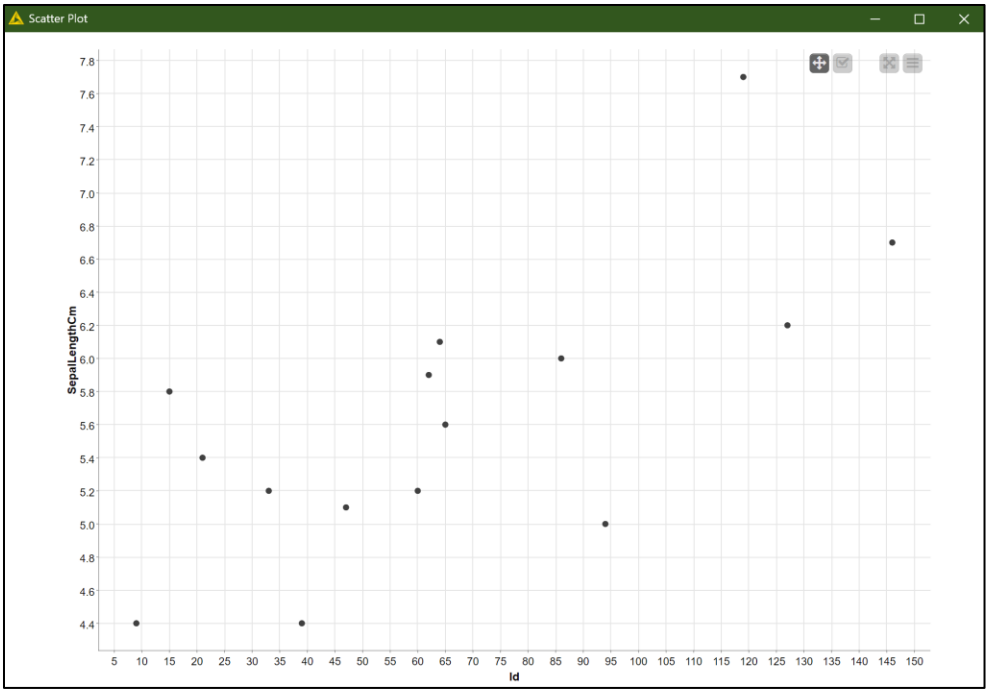
File Edit Hilite Navigation View

Table "default" - Rows: 15 | Spec - Columns: 7 | Properties | Flow Variables

Row ID	Id	SepalLe...	SepalW...	PetalLe...	PetalWi...	Species	Cluster
Row8	9	4.4	2.9	1.4	0.2	Iris-setosa	cluster_0
Row14	15	5.8	4	1.2	0.2	Iris-setosa	cluster_0
Row20	21	5.4	3.4	1.7	0.2	Iris-setosa	cluster_0
Row32	33	5.2	4.1	1.5	0.1	Iris-setosa	cluster_0
Row38	39	4.4	3	1.3	0.2	Iris-setosa	cluster_0
Row46	47	5.1	3.8	1.6	0.2	Iris-setosa	cluster_0
Row59	60	5.2	2.7	3.9	1.4	Iris-versicolor	cluster_1
Row61	62	5.9	3	4.2	1.5	Iris-versicolor	cluster_1
Row63	64	6.1	2.9	4.7	1.4	Iris-versicolor	cluster_1
Row64	65	5.6	2.9	3.6	1.3	Iris-versicolor	cluster_1
Row85	86	6	3.4	4.5	1.6	Iris-versicolor	cluster_1
Row93	94	5	2.3	3.3	1	Iris-versicolor	cluster_1
Row118	119	7.7	2.6	6.9	2.3	Iris-virginica	cluster_2

Scatter Plot:

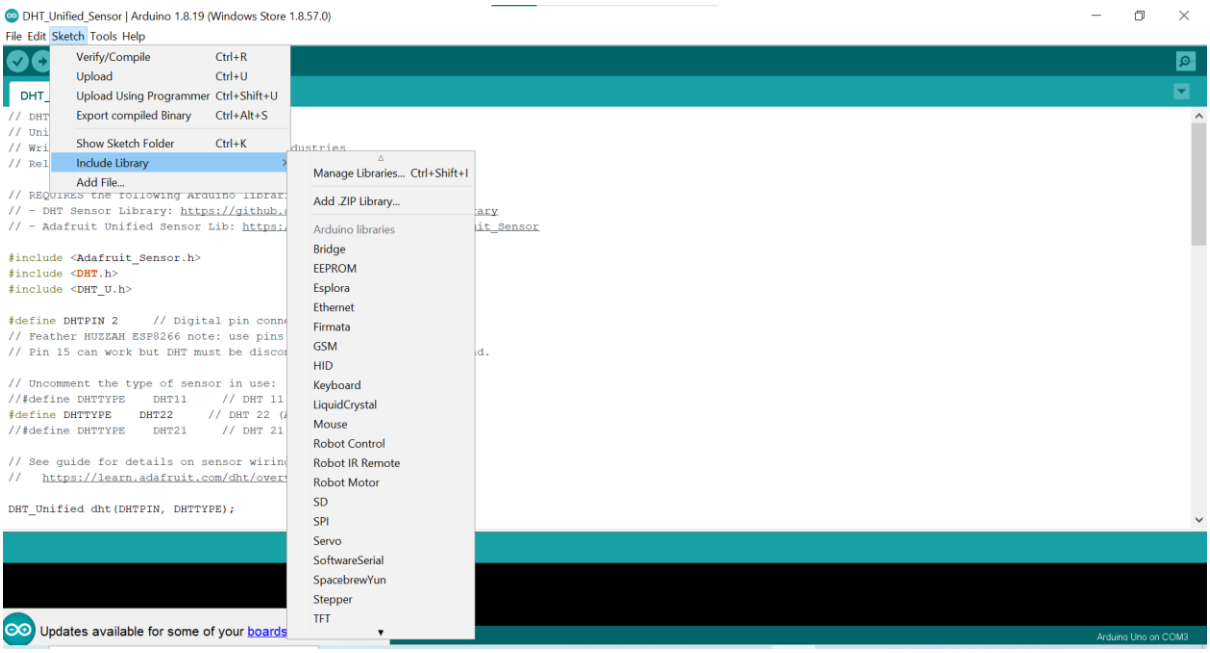




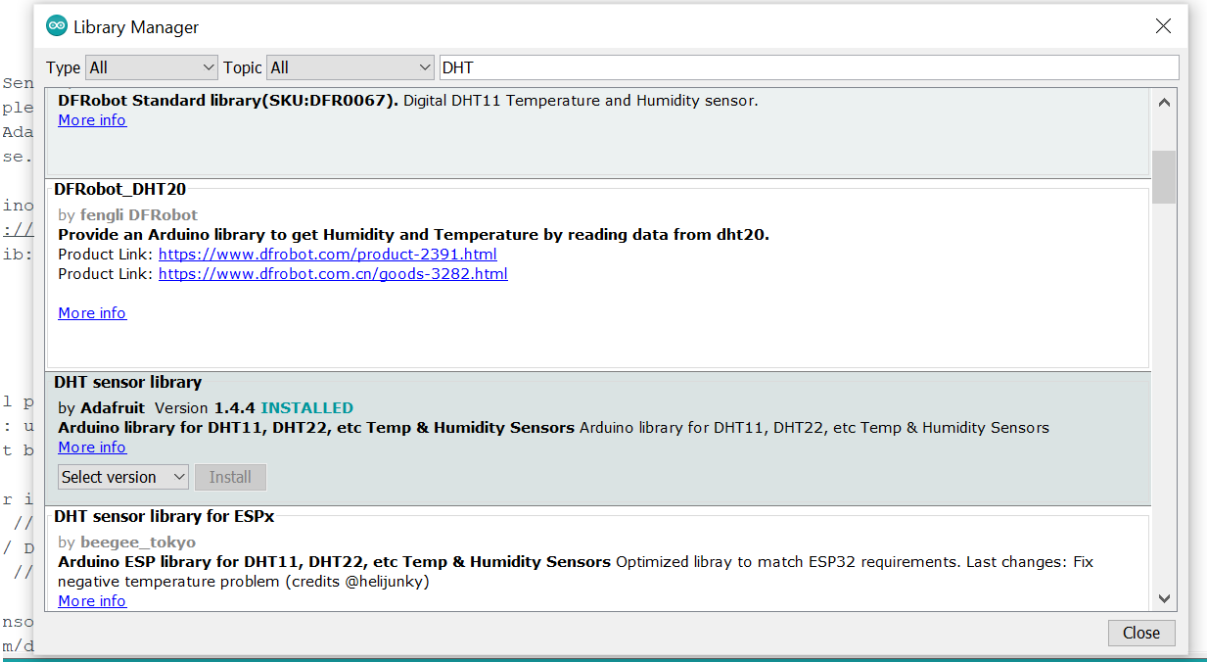
Inference: The K Means node is used to perform k-means clustering on the dataset. The Color Manager and Shape Manager nodes are used to assign colours and shapes to the clusters. The Scatter Plot node is used to view the clusters as scatter plots.

Result: Thus, we have clustered data and classified them under respective features on Knime Analytics Platform with the help of Color Manger & Shape Manager and analyzed the data using k-Means cluster node and Cluster assigner node.

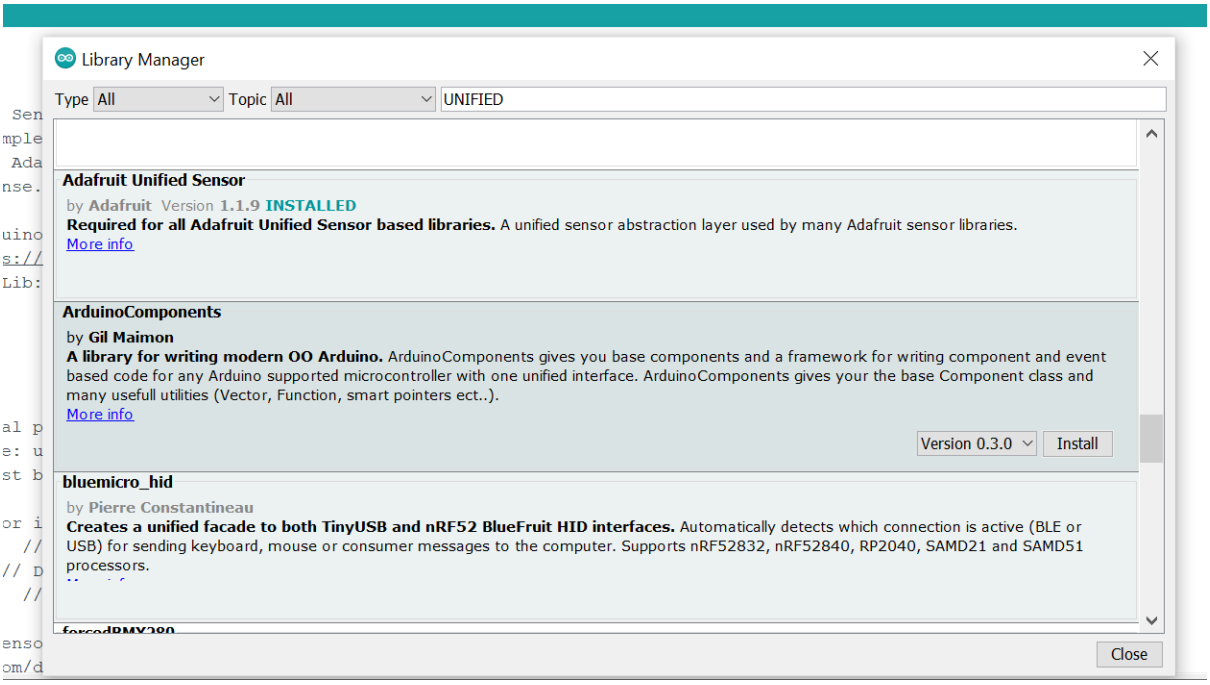
Temperature, humidity sensing with Arduino and node red dashboard



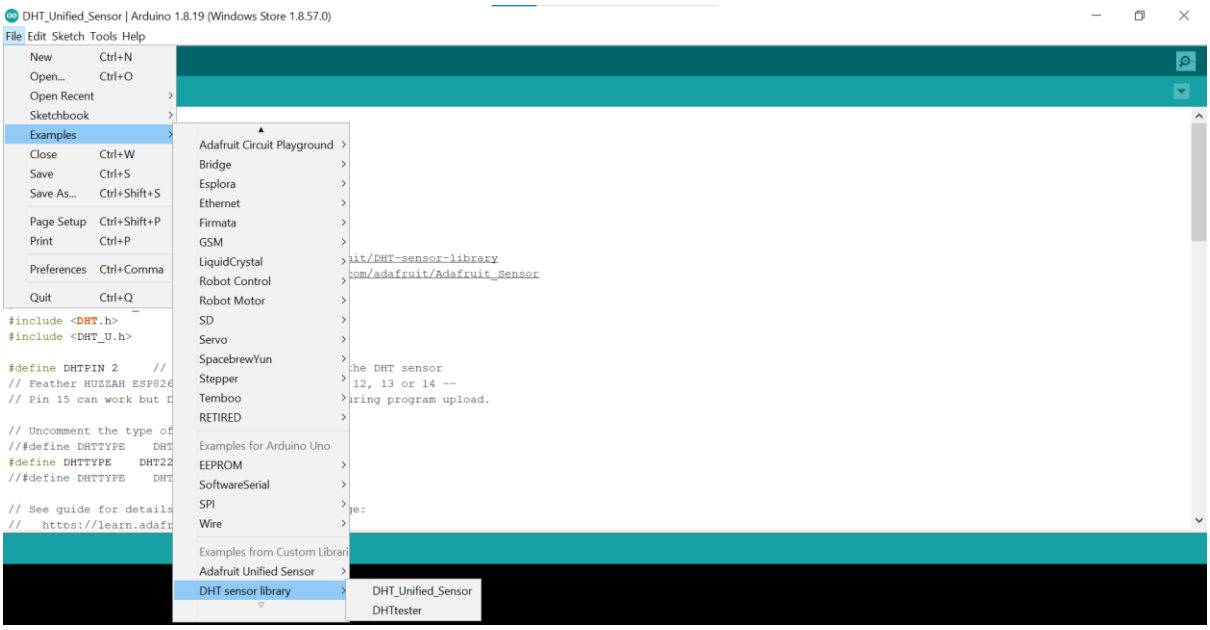
Click manage libraries



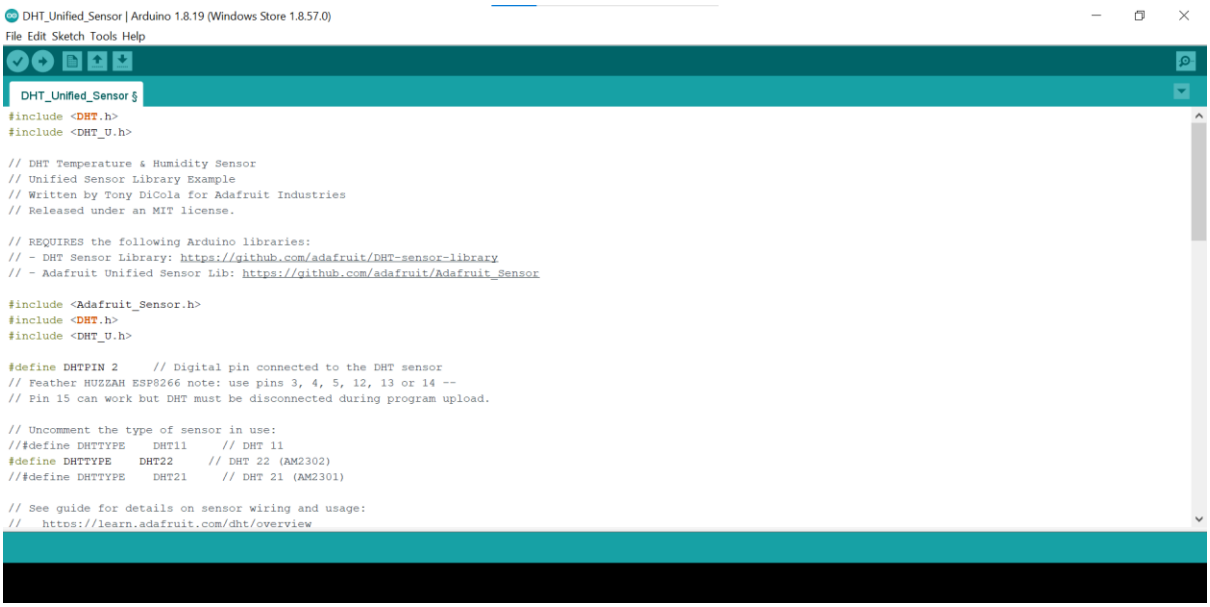
INSTALL DHT SENSOR LIBRARY



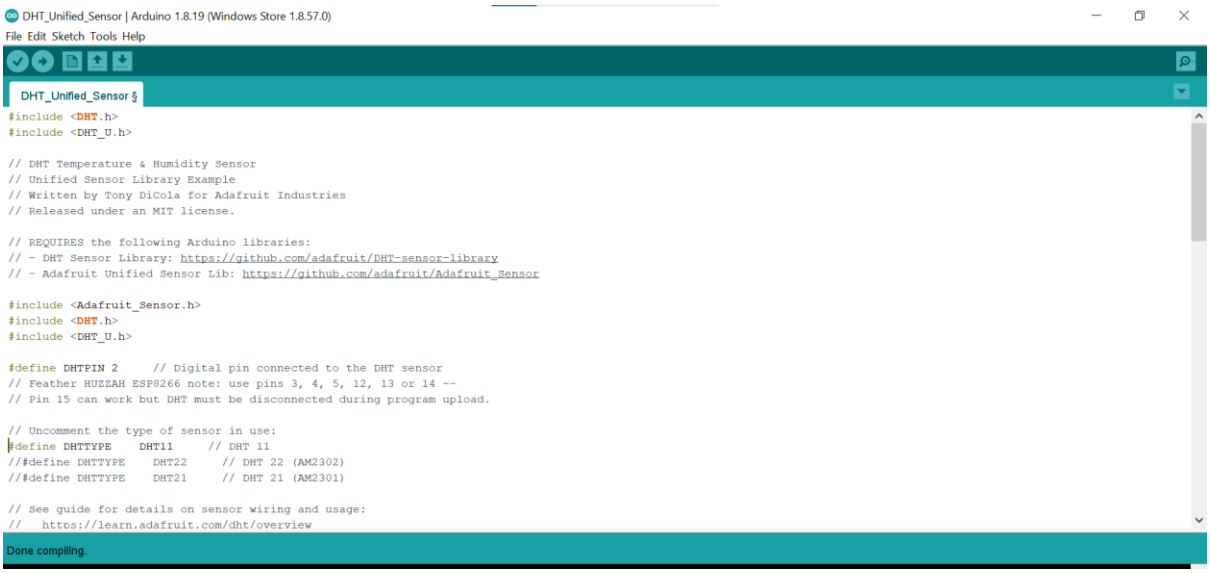
Make sure adafruit unified sensor is installed



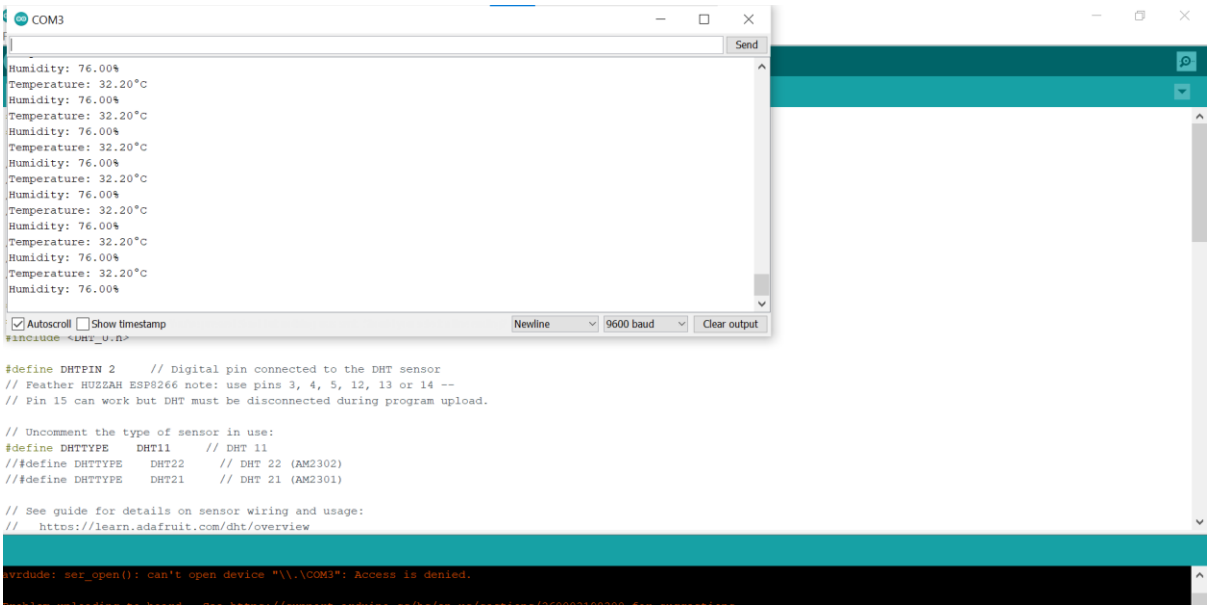
Choose DHT unified sensor



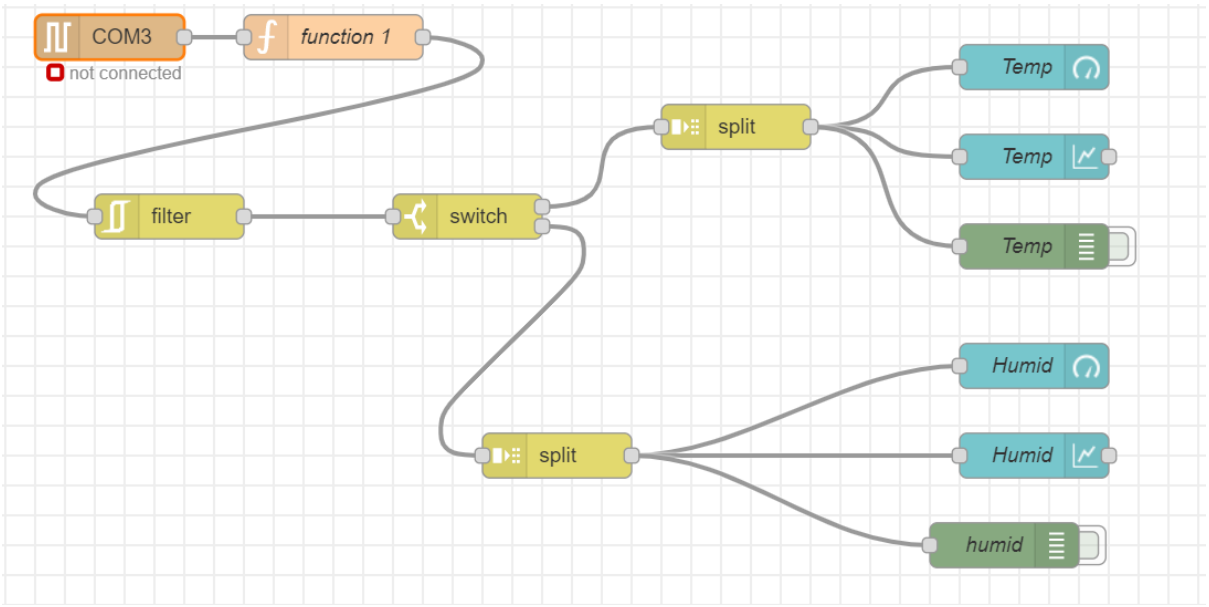
I am using DHT11 so

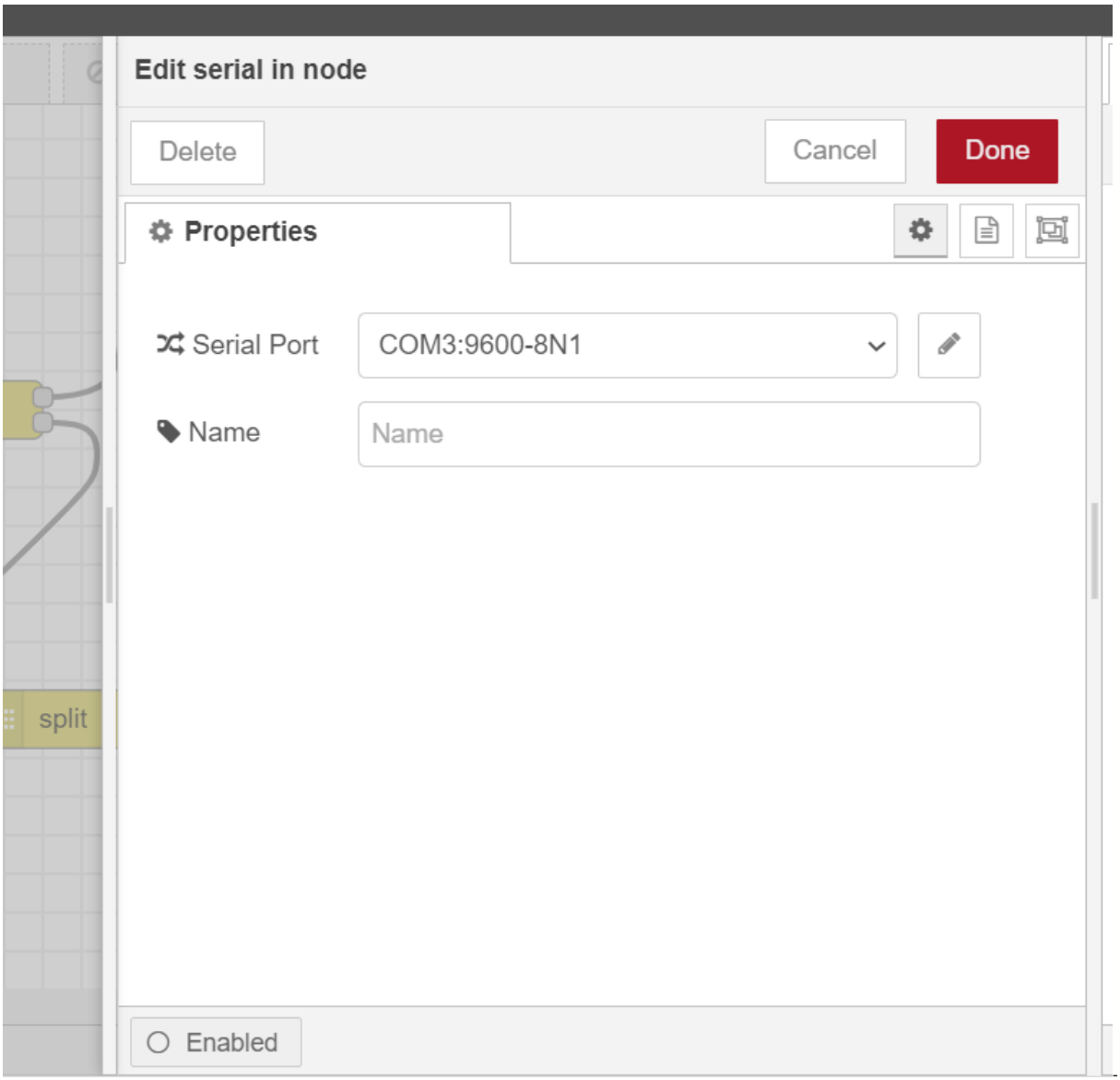


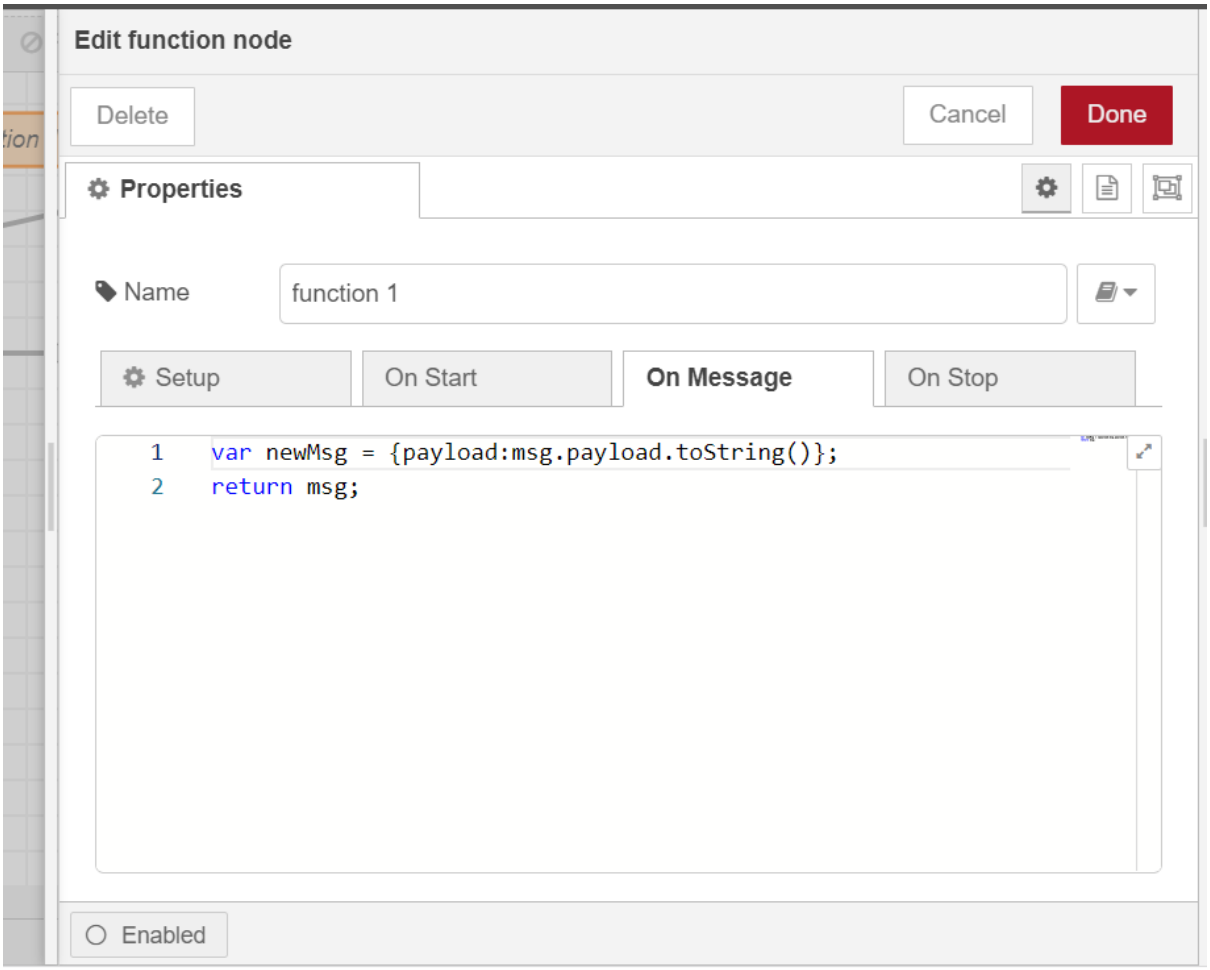
Click serial monitor



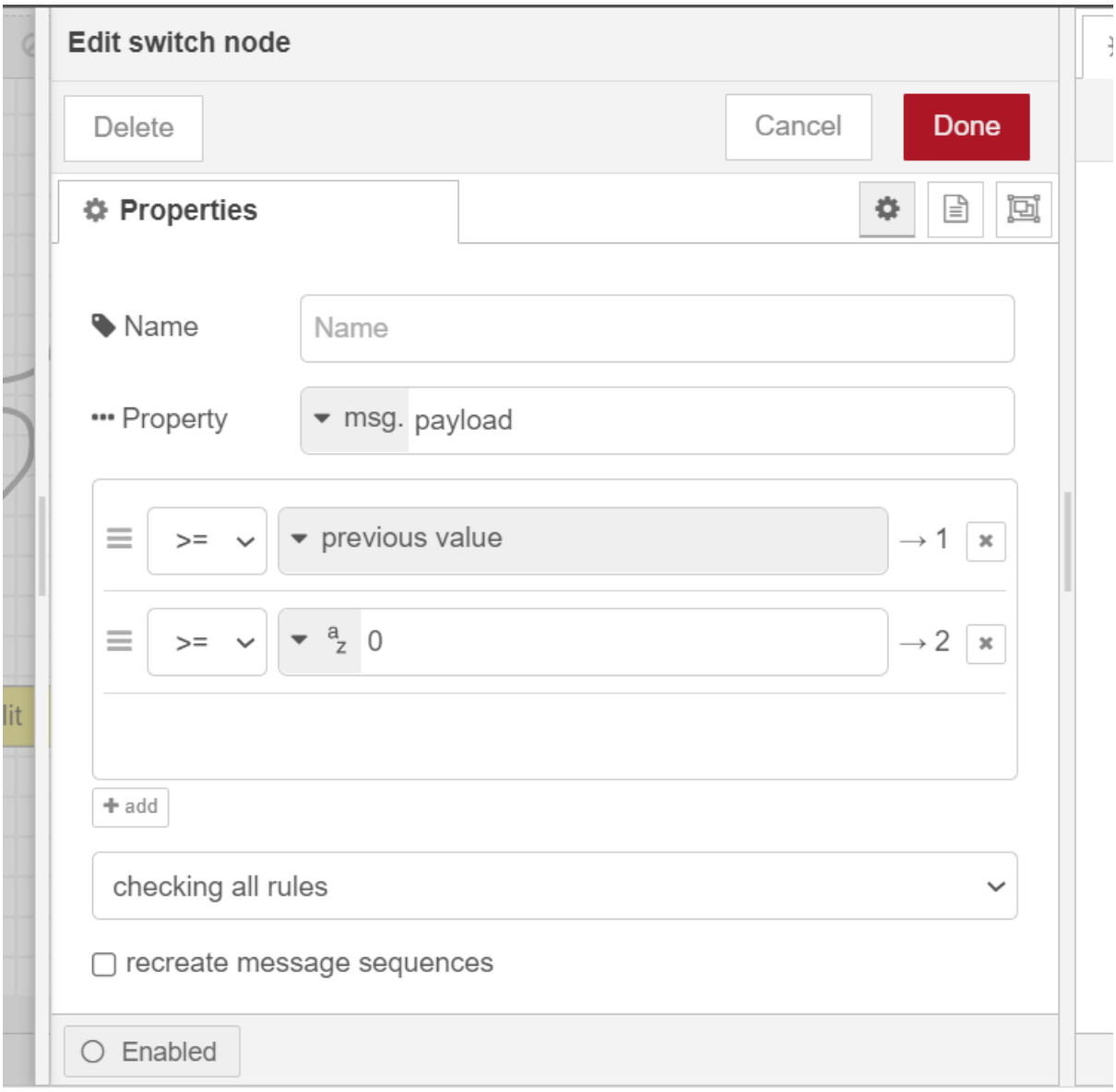
Go to node red

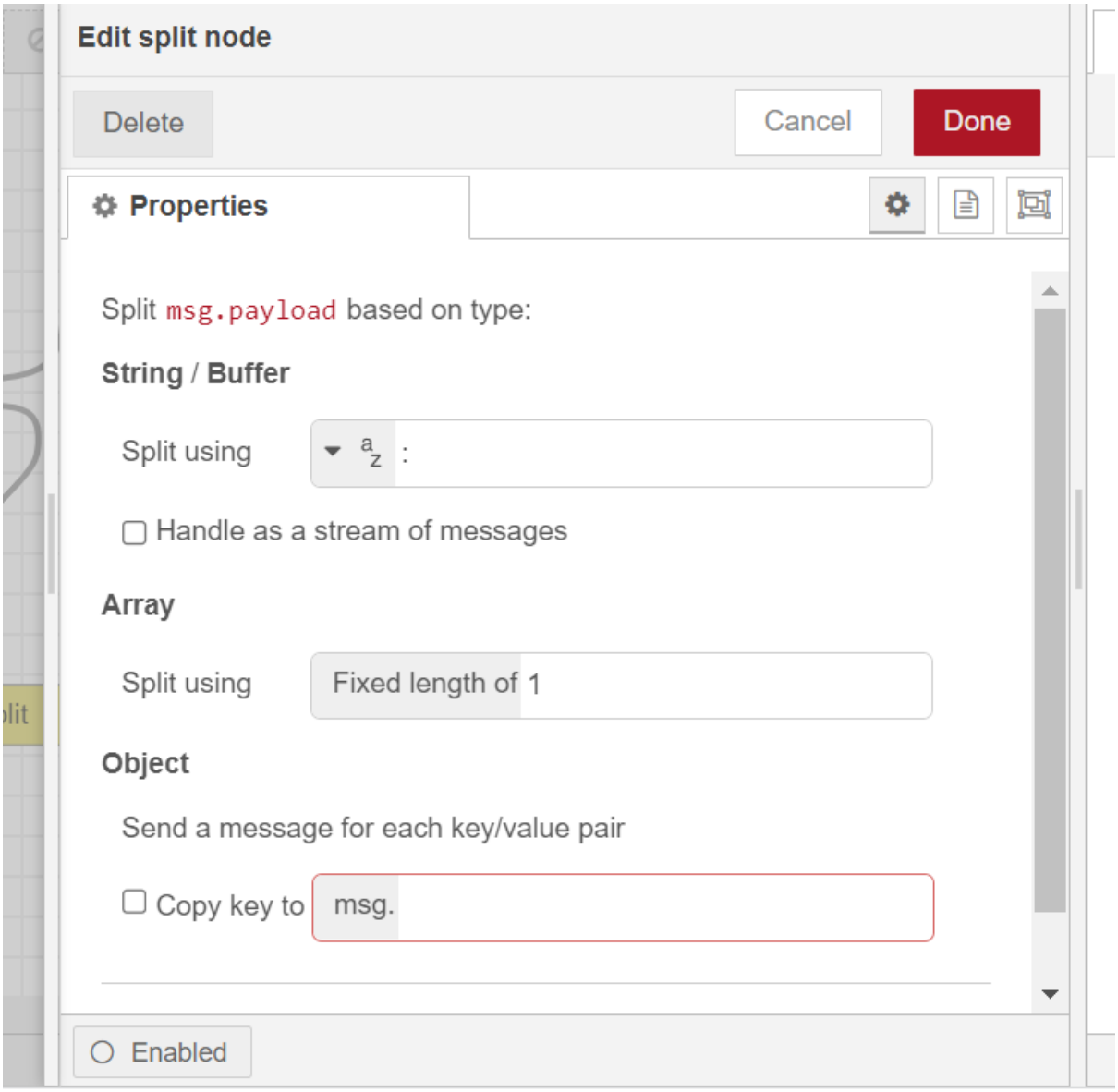

COM3
not connected
function 1
filter
switch
split
split
Temp
Temp
Temp
Humid
Humid
humid

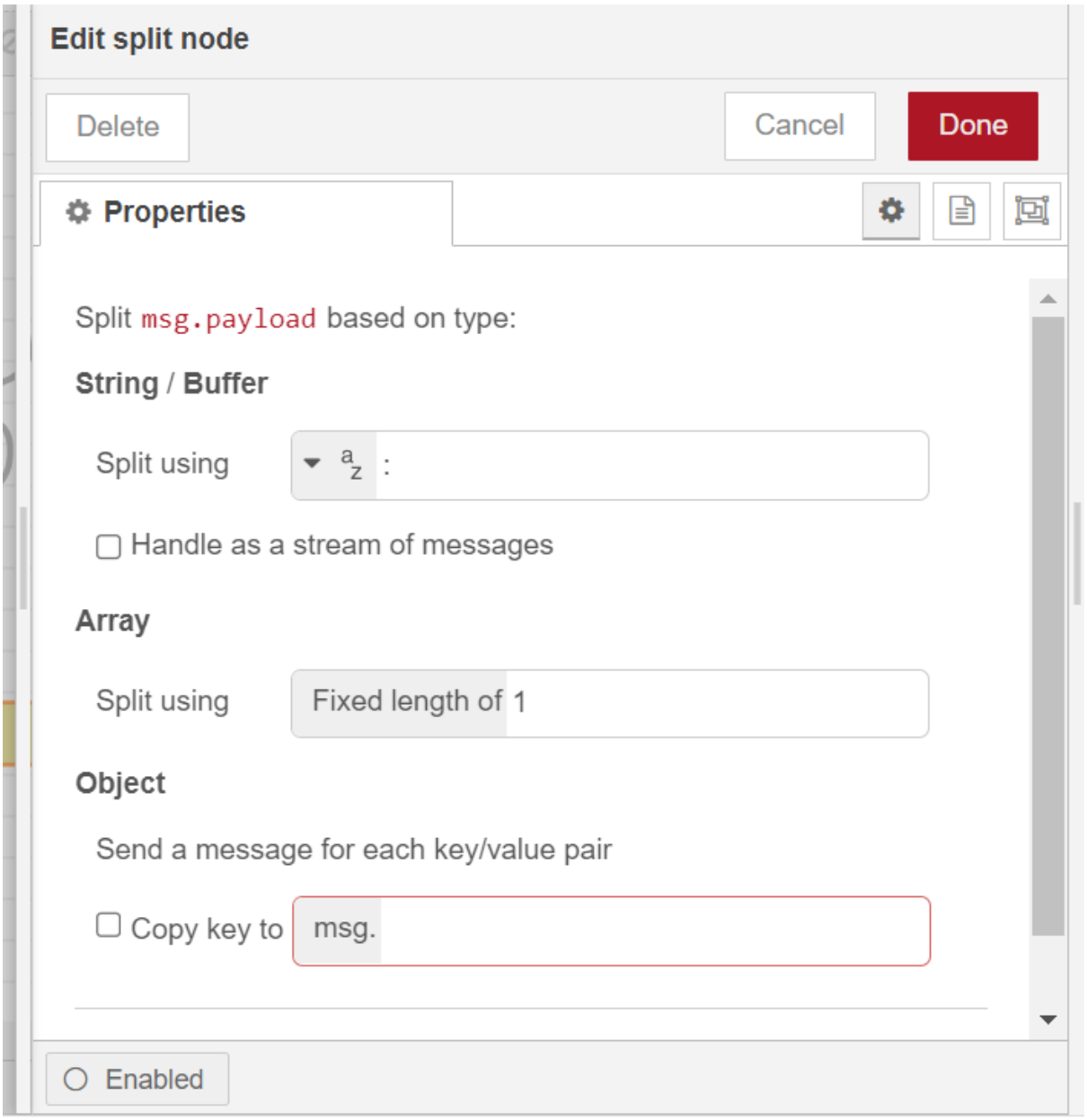

Edit serial in node
Delete
Cancel
Done
Properties
Serial Port
COM3:9600-8N1
Name
Name
split
Enabled

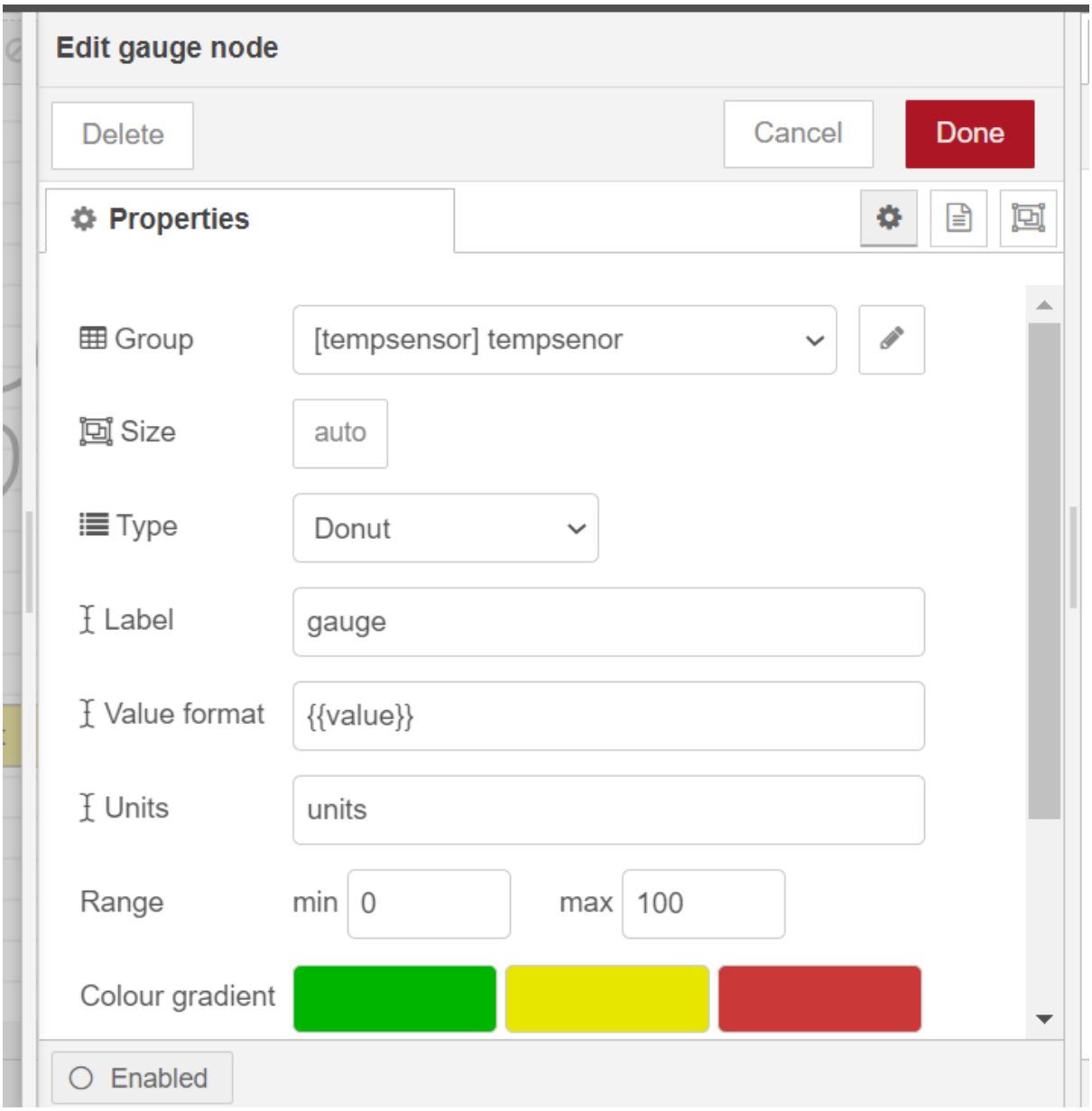


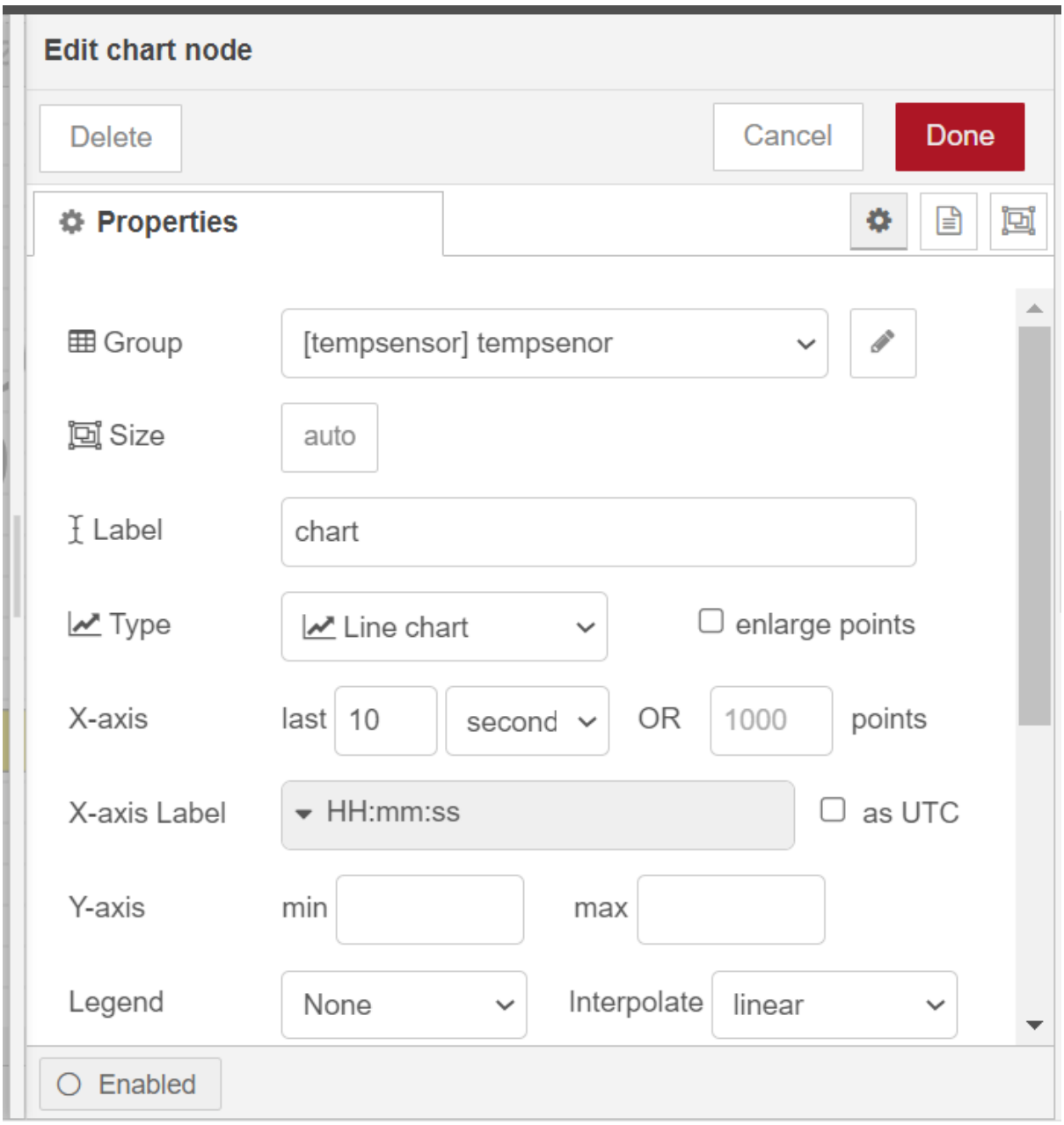

```
var newMsg = {payload:msg.payload.toString()};

return msg;
```

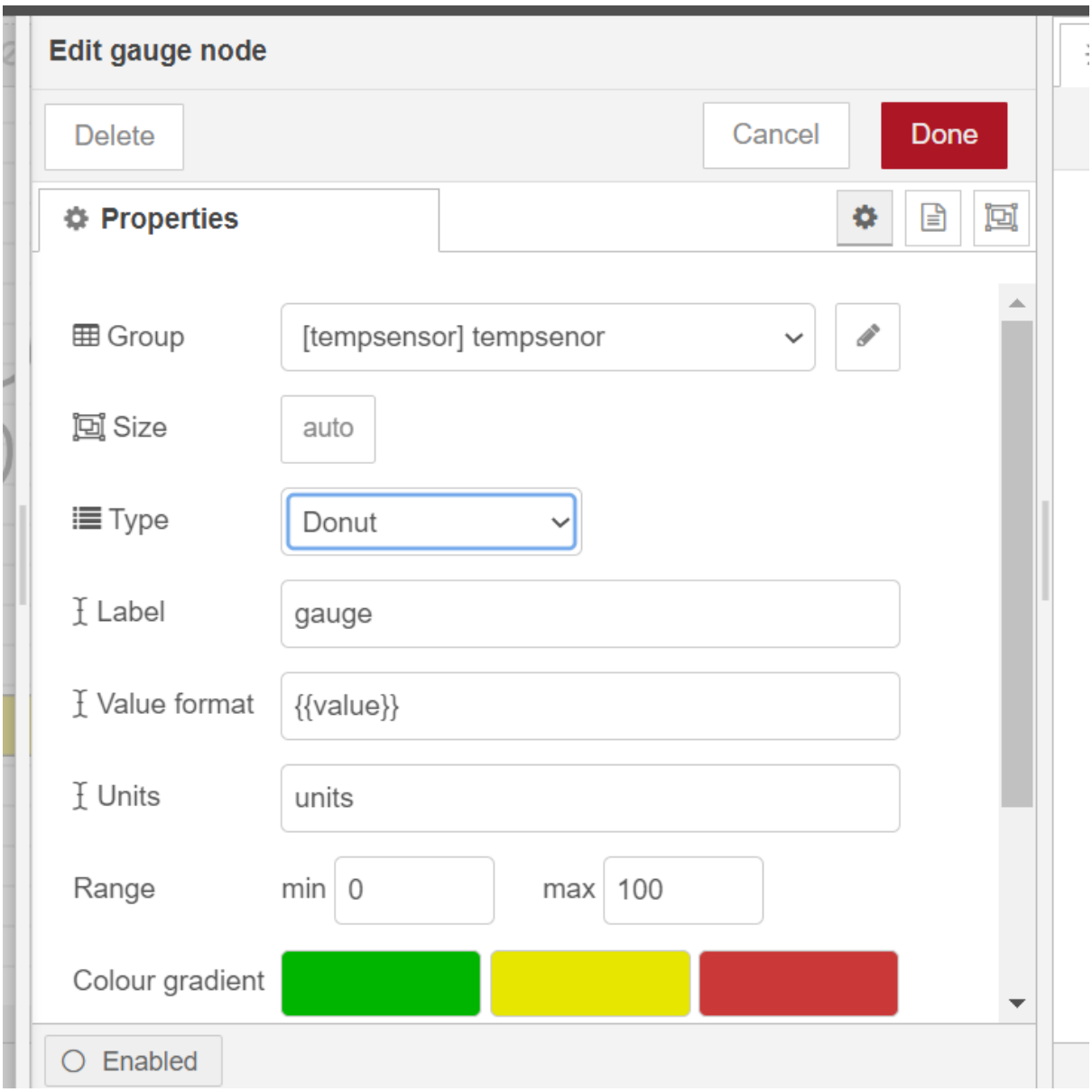

Edit switch node
Delete
Cancel
Done
Properties
Name
Name
Property
msg. payload
>=
previous value
→ 1
>=
0
→ 2
add
checking all rules
recreate message sequences
Enabled

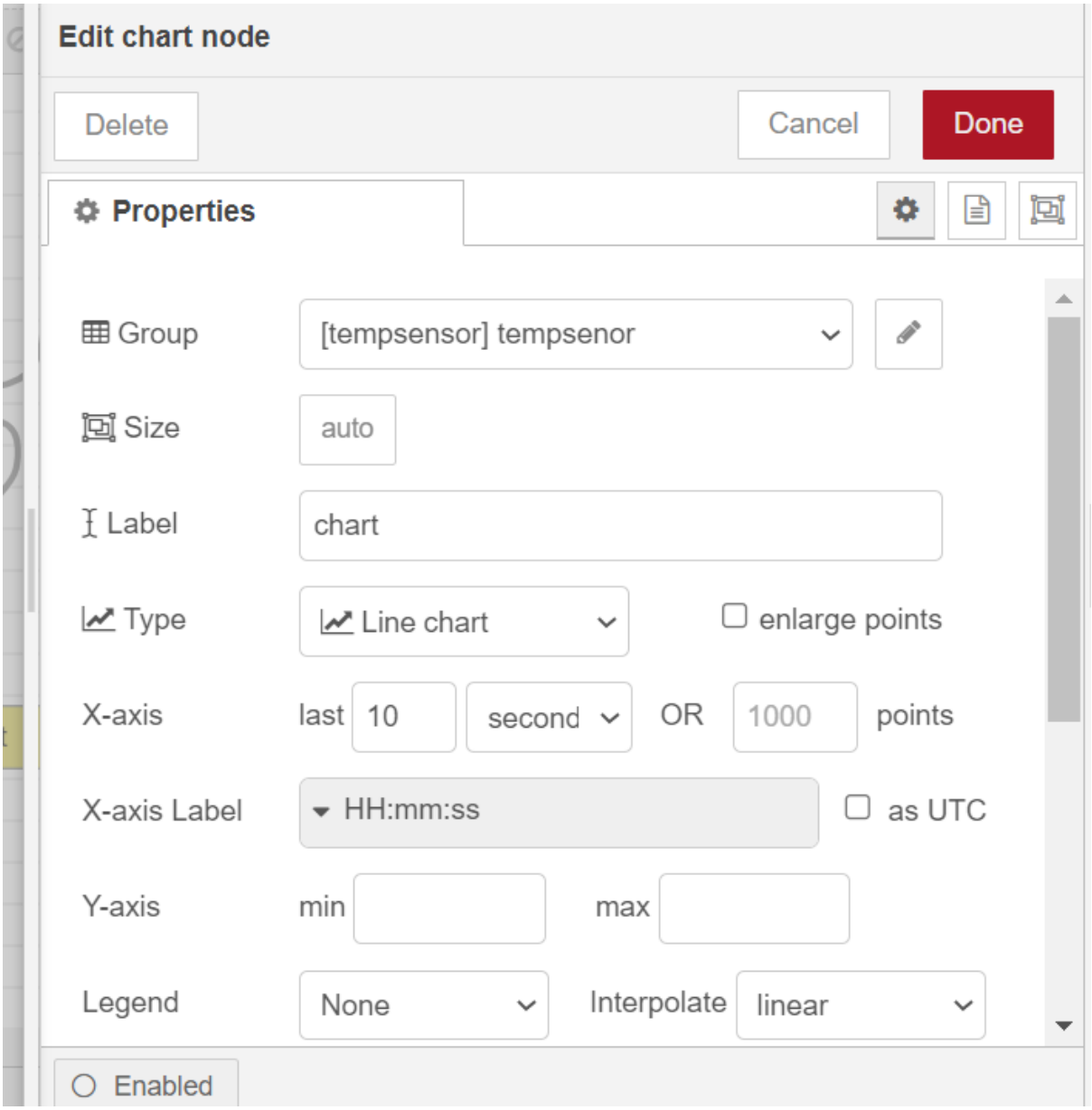

Edit split node
Delete
Cancel
Done
Properties
Split msg.payload based on type:
String / Buffer
Split using
a z
:
Handle as a stream of messages
Array
Split using
Fixed length of 1
Object
Send a message for each key/value pair
Copy key to
msg.
Enabled


Edit split node
Delete
Cancel
Done
Properties
Split msg.payload based on type:
String / Buffer
Split using
a z
:
Handle as a stream of messages
Array
Split using
Fixed length of 1
Object
Send a message for each key/value pair
Copy key to
msg.
Enabled


Edit gauge node
Delete
Cancel
Done
Properties
Group
[tempsensor] tempsenor
Size
auto
Type
Donut
Label
gauge
Value format
{{value}}
Units
units
Range
min
0
max
100
Colour gradient
Enabled

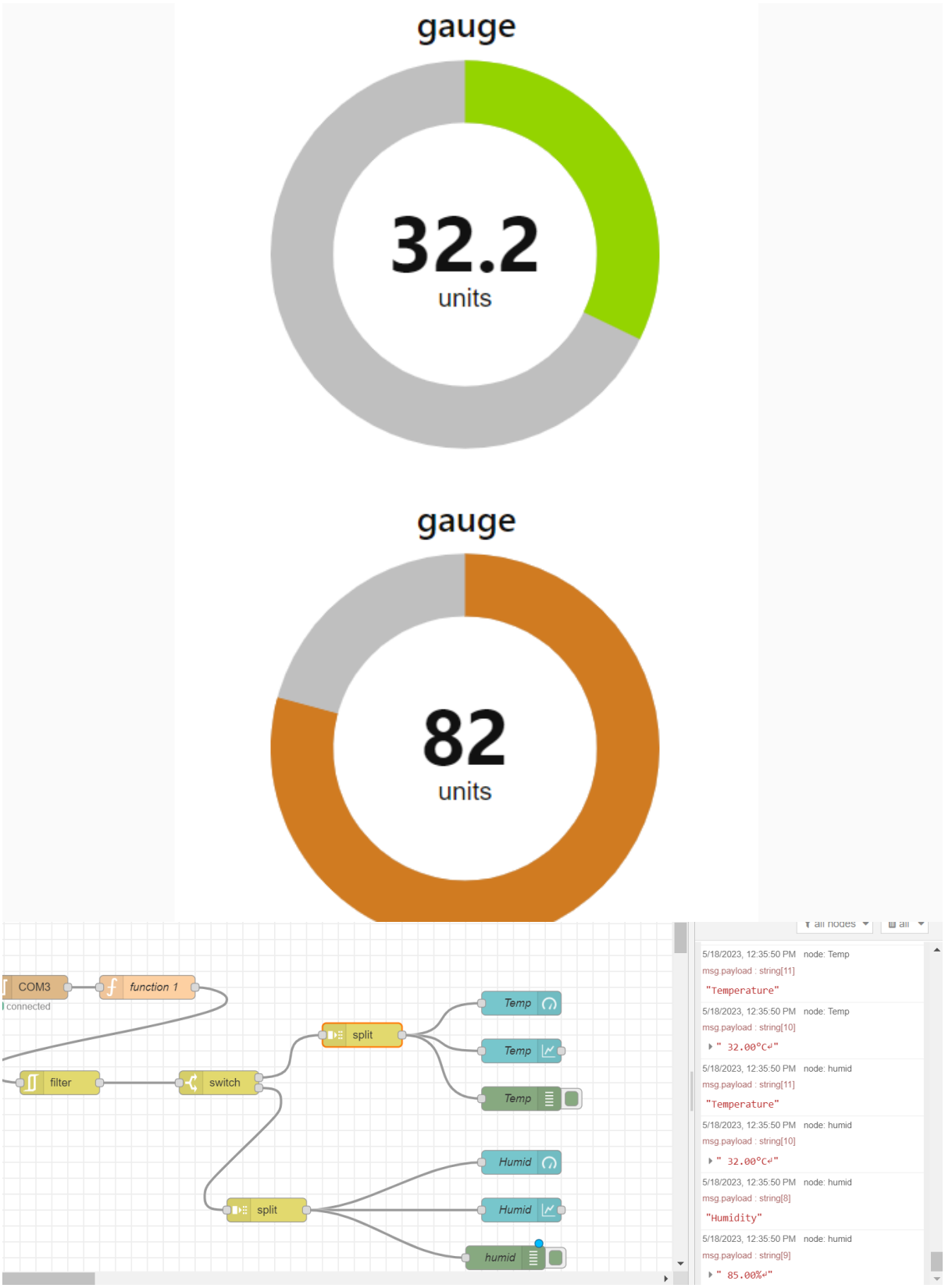

Edit chart node
Delete
Cancel
Done
Properties
Group
[tempsensor] tempsenor
Size
auto
Label
chart
Type
Line chart
enlarge points
X-axis
last
10
second
OR
1000
points
X-axis Label
HH:mm:ss
as UTC
Y-axis
min
max
Legend
None
Interpolate
linear
Enabled


Edit gauge node
Delete
Cancel
Done
Properties
Group
[tempsensor] tempsenor
Size
auto
Type
Donut
Label
gauge
Value format
{{value}}
Units
units
Range
min
0
max
100
Colour gradient
Enabled



Close the com output in arduino

Deploy and ui


gauge
32.2
units
gauge
82
units
COM3
connected
function 1
filter
switch
split
split
Temp
Temp
Temp
Humid
Humid
humid
all nodes
all
5/18/2023, 12:35:50 PM node: Temp
msg.payload : string[11]
"Temperature"
5/18/2023, 12:35:50 PM node: Temp
msg.payload : string[10]
" 32.00°C"
5/18/2023, 12:35:50 PM node: humid
msg.payload : string[11]
"Temperature"
5/18/2023, 12:35:50 PM node: humid
msg.payload : string[10]
" 32.00°C"
5/18/2023, 12:35:50 PM node: humid
msg.payload : string[8]
"Humidity"
5/18/2023, 12:35:50 PM node: humid
msg.payload : string[9]
" 85.00%"

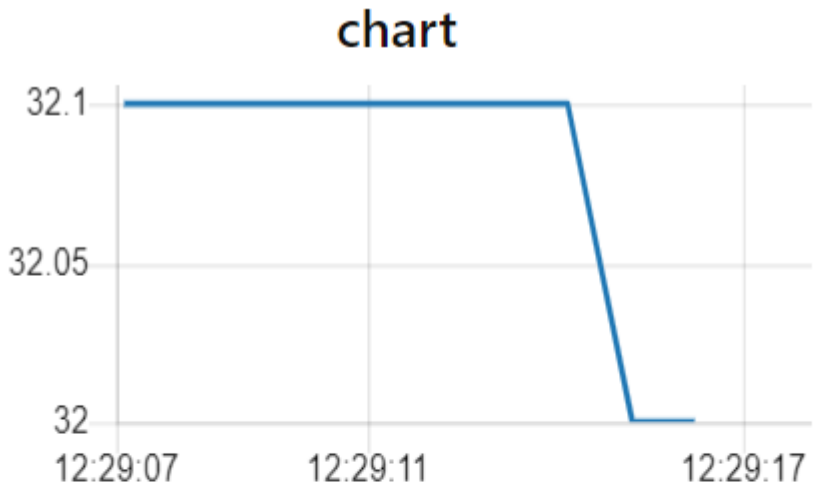


chart
32.1
32.05
32
12:29:07
12:29:11
12:29:17

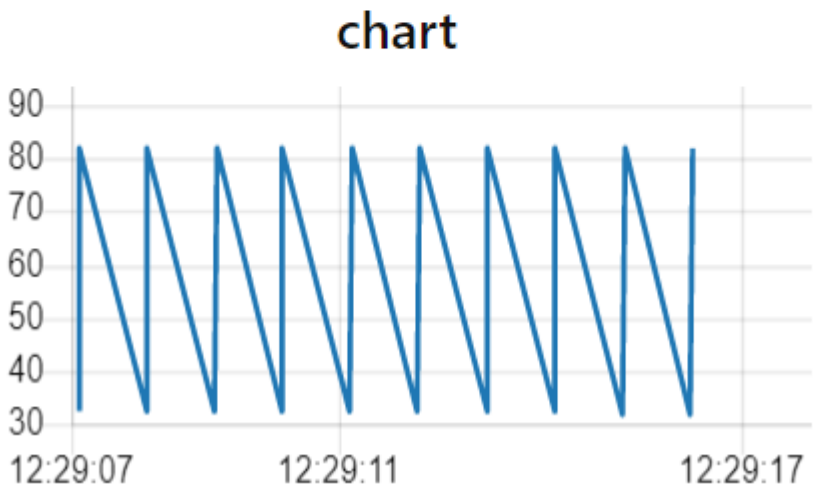


chart
90
80
70
60
50
40
30
12:29:07
12:29:11
12:29:17