Experiment Number: 1	**Display Light Intensity using Node-RED and MQTT**	**Name:** Rahul Karthik S
Date: 04-01-2024		**Register Number:** 21BEC1851

Aim:

To Measure the light intensity in the room and display the output in Debug Monitor, UI and send the data through MQTT and display the output in HiveMQ.

Software Required:

Node-RED, HiveMQ (MQTT)

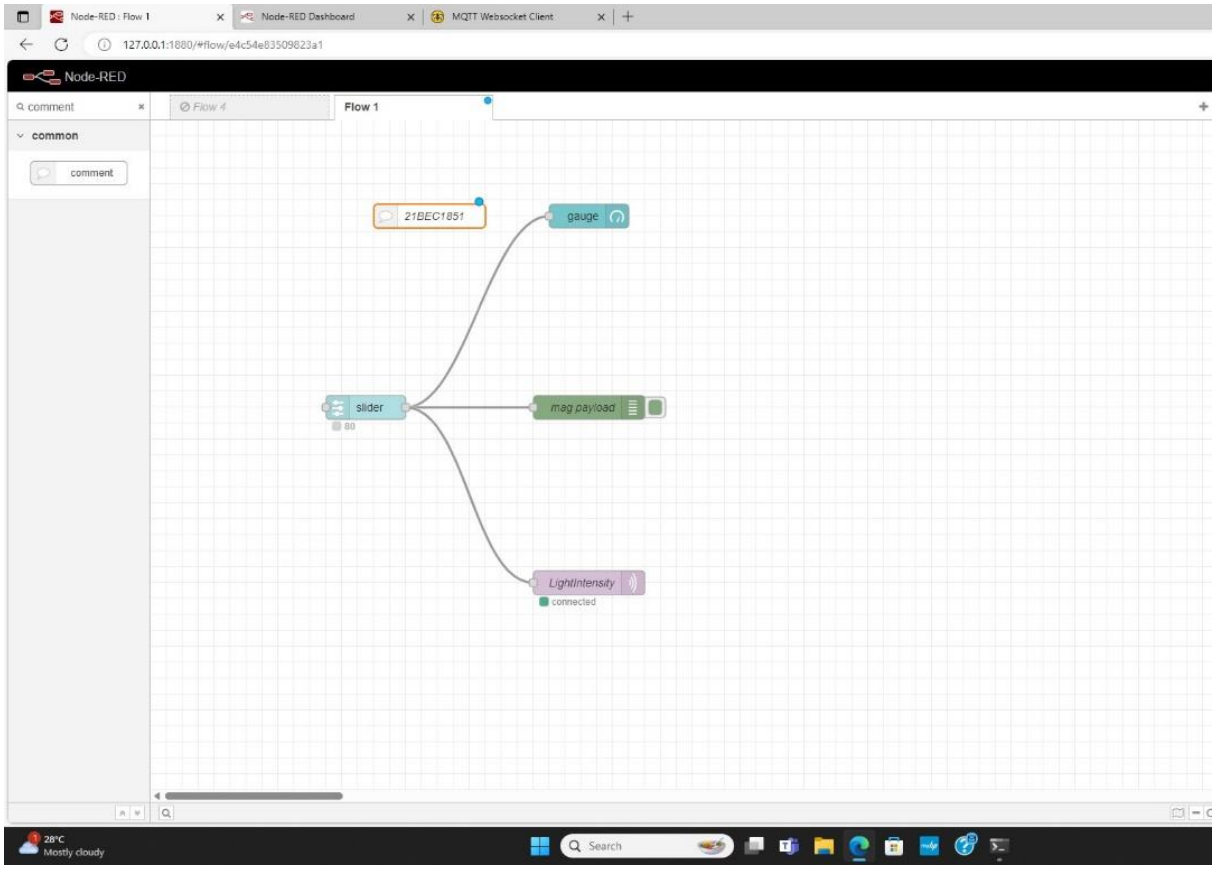
Theory:

- 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, bi-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:

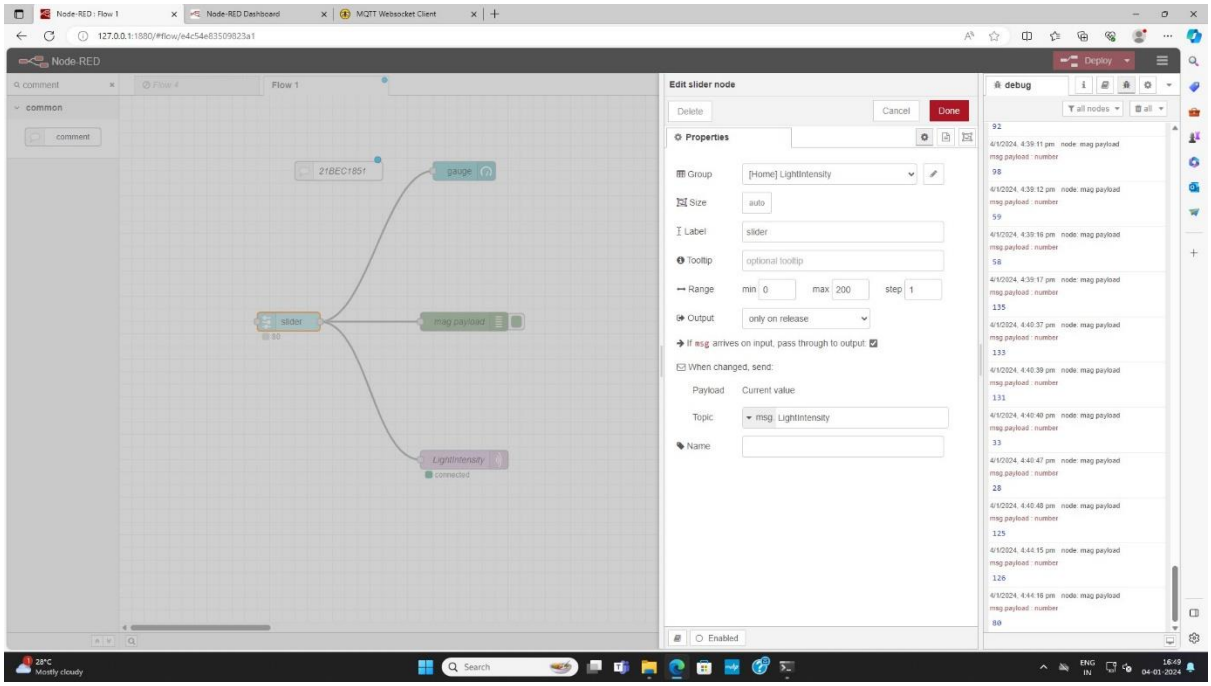
- Trigger Node-RED in Command Prompt using “node-red -v” command.
- Go to http://127.0.0.1:1880/
- Arrange the nodes according to the flow and assign properties for slider node, gauge node, debug node and MQTT out node.
- Open HiveMQ (http://www.hivemq.com/demos/websocket-client/).
- Add New Subscription and subscribe to the topic.
- Check the output from MQTT Dashboard and Node-RED debug window.

Flow:

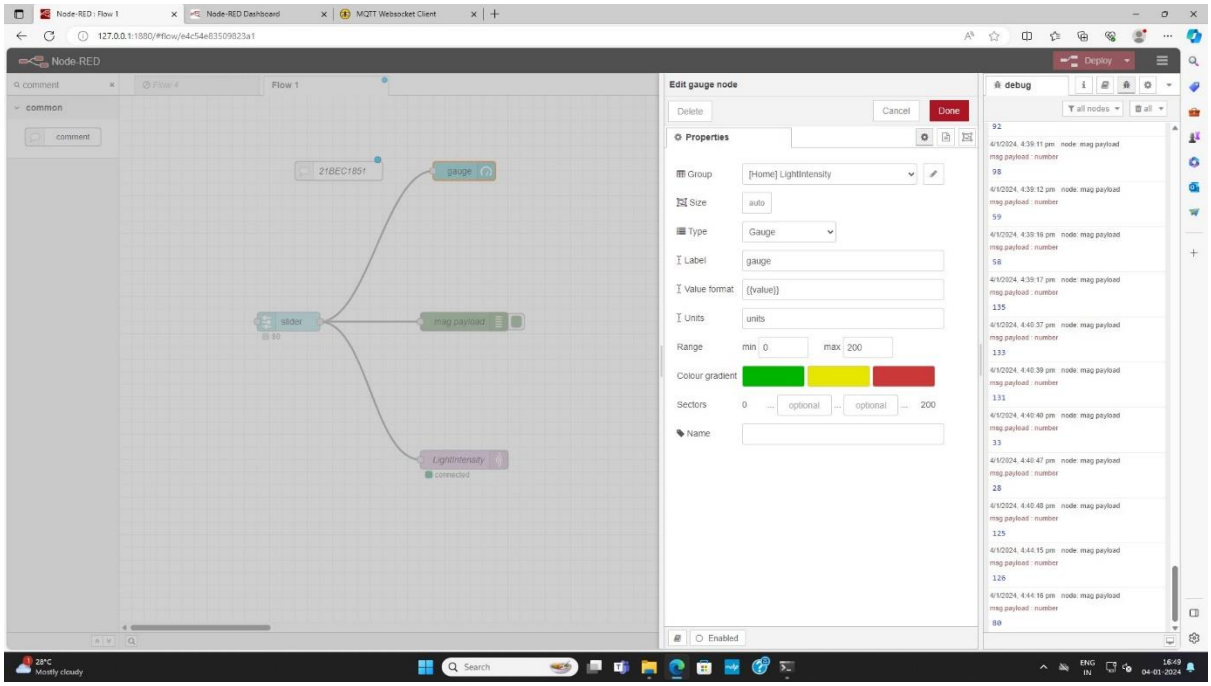


Properties:

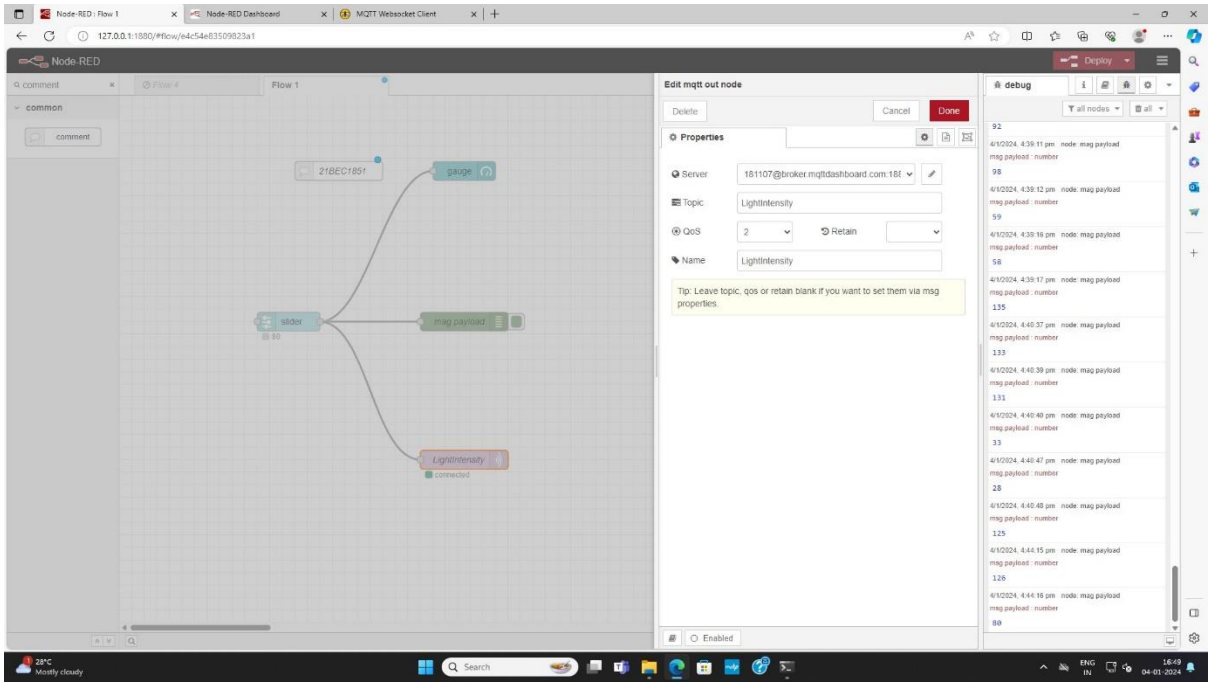
Slider Node:



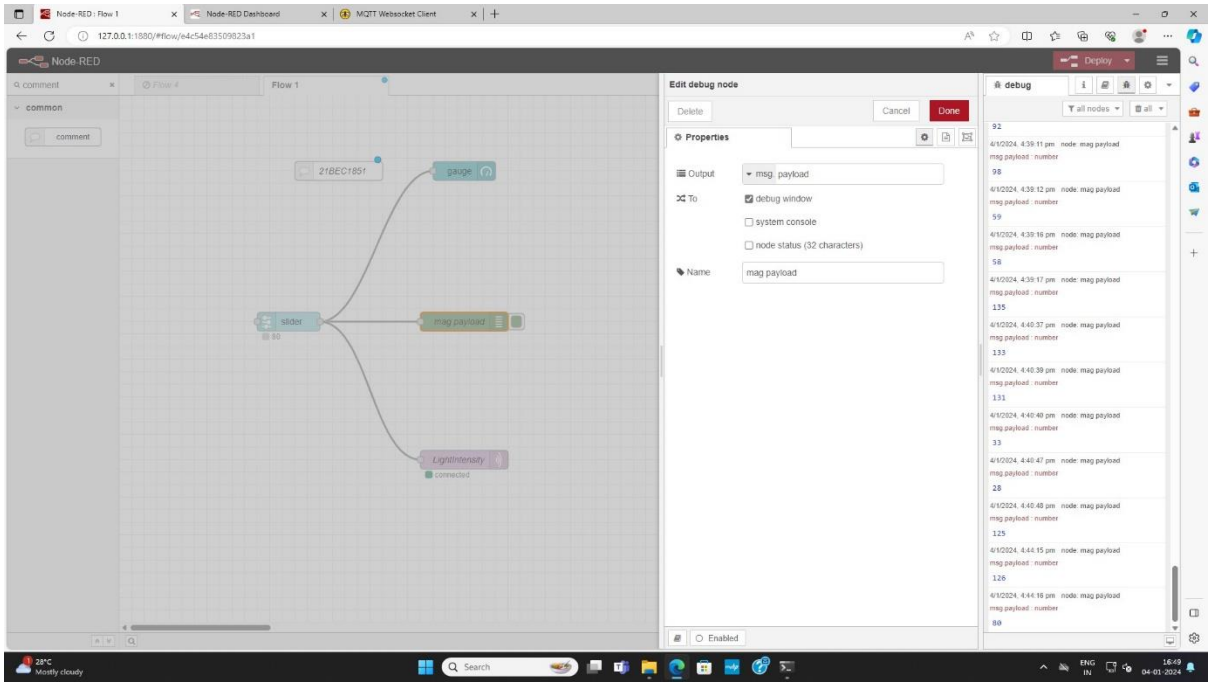
Gauge Node:



MQTT Out Node:

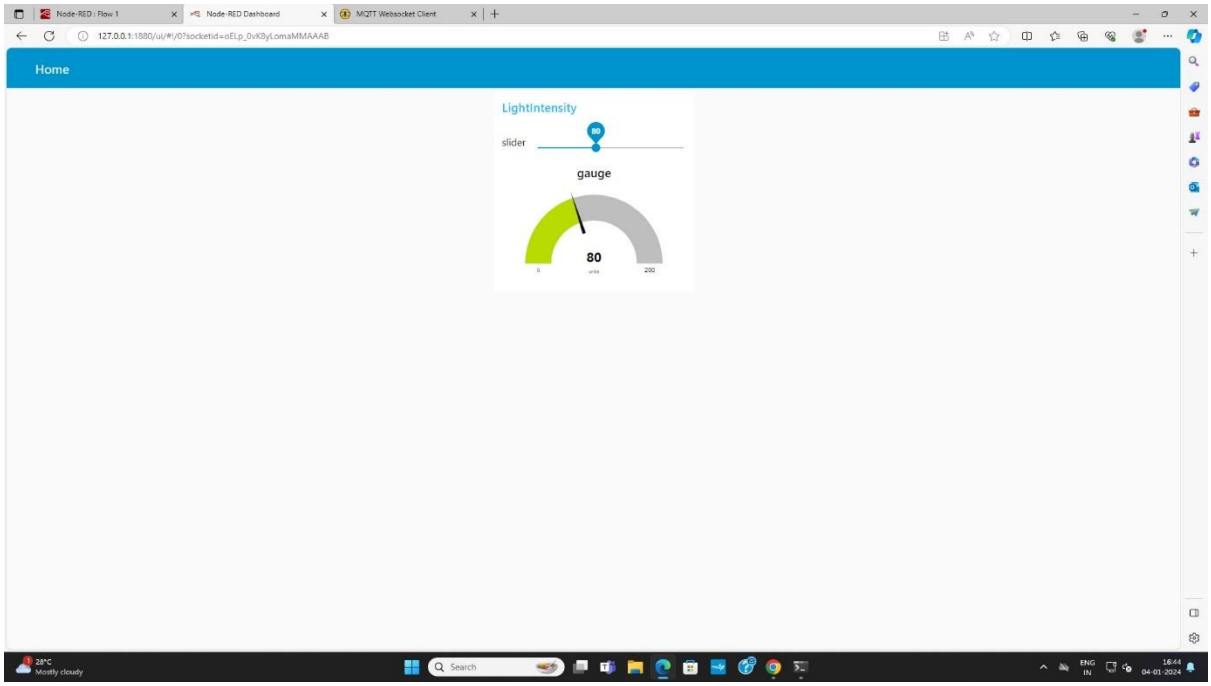


Debug Node:

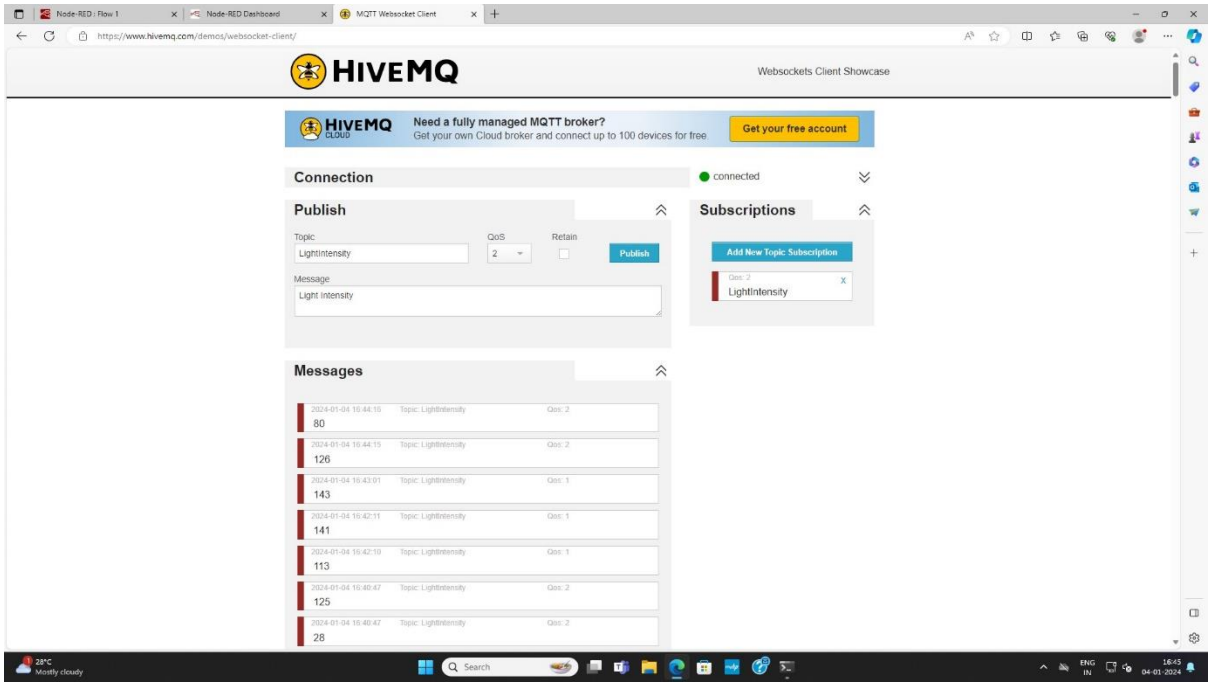


Output:

Node-RED Slider:



MQTT Dashboard:



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, the light intensity is displayed using the slider, in Node-Red and MQTT.

Experiment Number: 2	**Web based Application (HTML) Form Creation and Submission in Node-RED**	**Name:** Rahul Karthik S
Date: 18-01-2024		**Register Number:** 21BEC1851

Aim:

To create a Web based Application using form creation and submission with HTML in Node-RED.

Software Required:

Node-RED

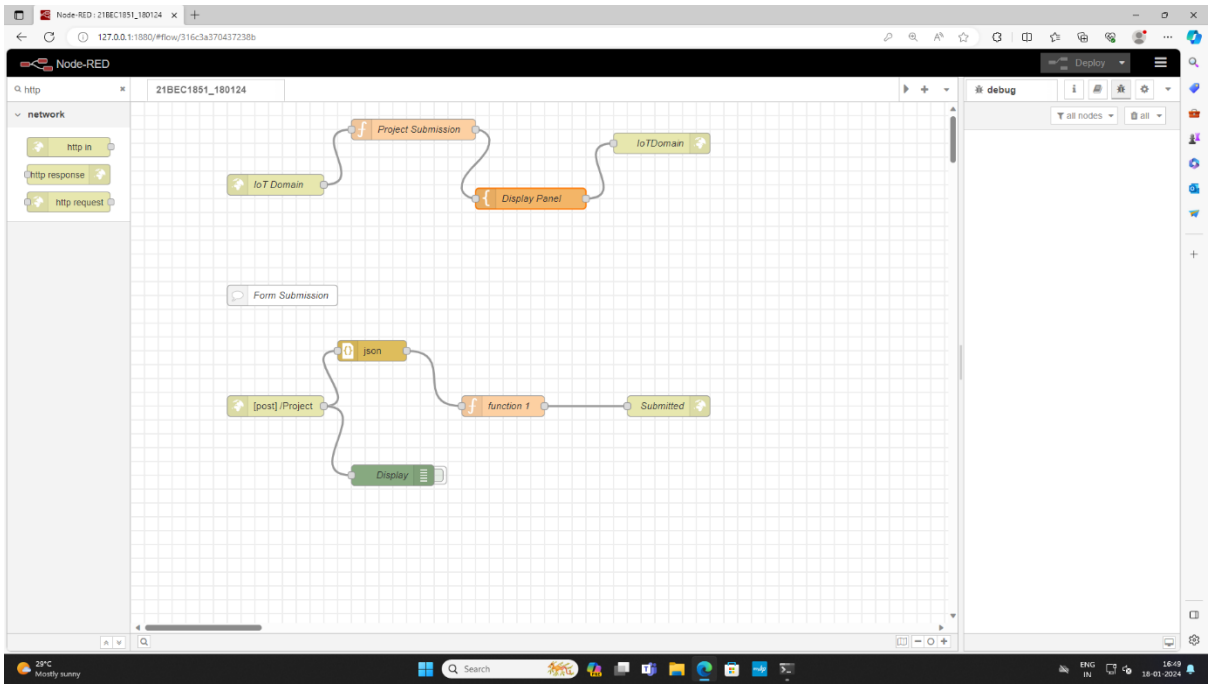
Theory:

- Web API: Web API as the name suggests, is an API over the web which can be accessed using HTTP protocol.

Procedure:

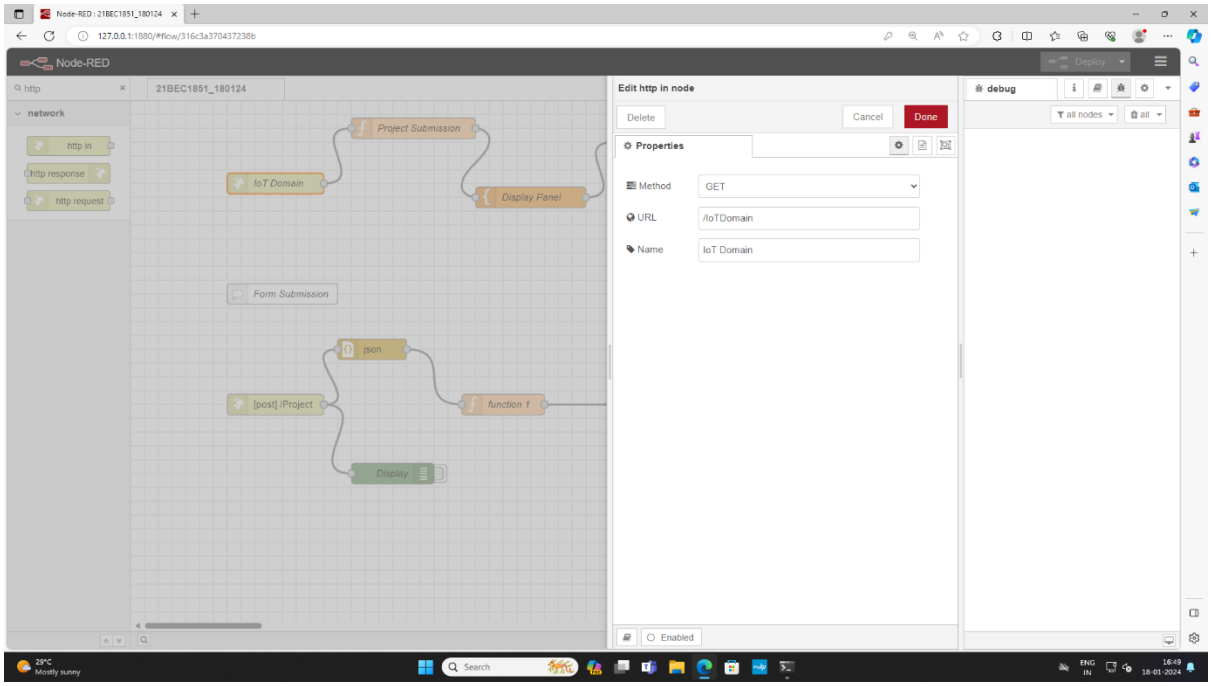
- Trigger node red in Command Prompt using “node-red -v” command.
- Arrange nodes as shown in the flow and configure properties
- Run the web app in the browser and insert values in the form
- Check for the output in the debug window.

Flow:

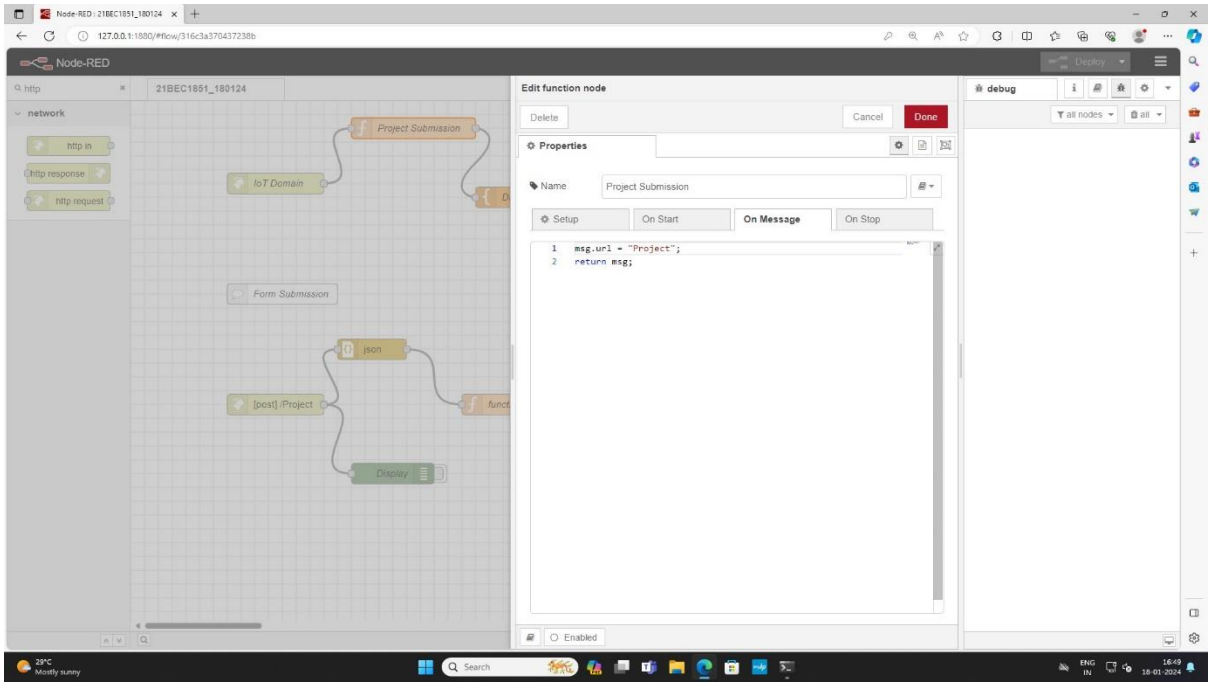


Properties:

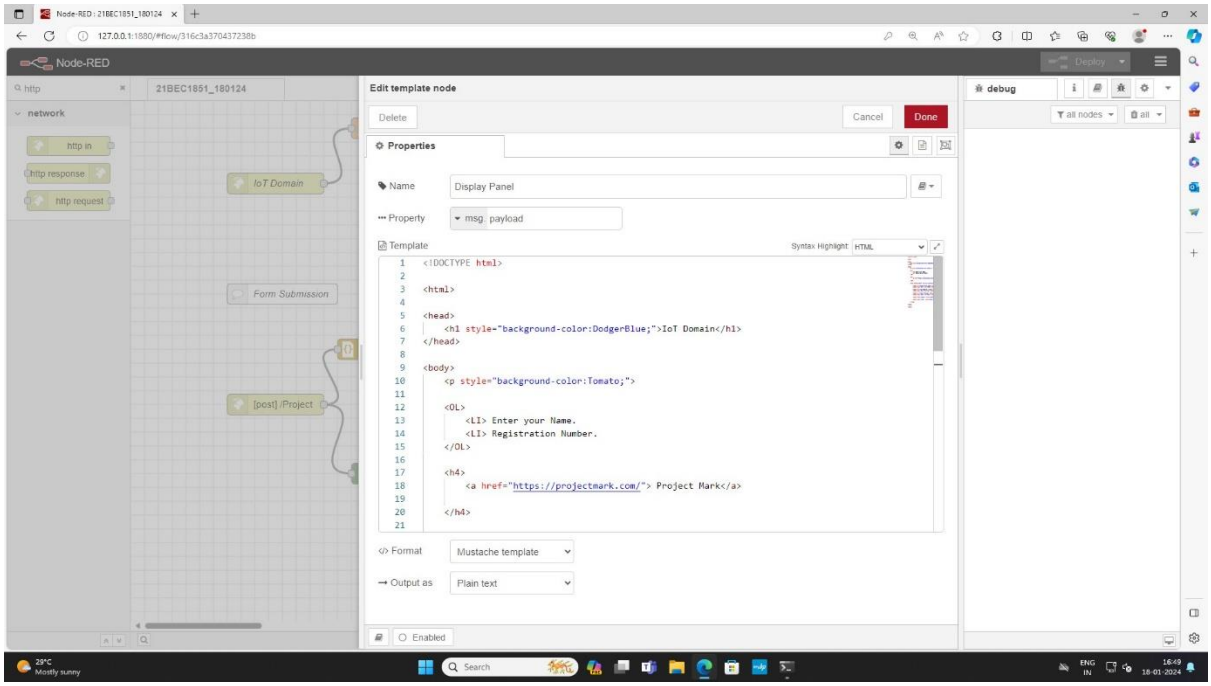
HTTP in Node:



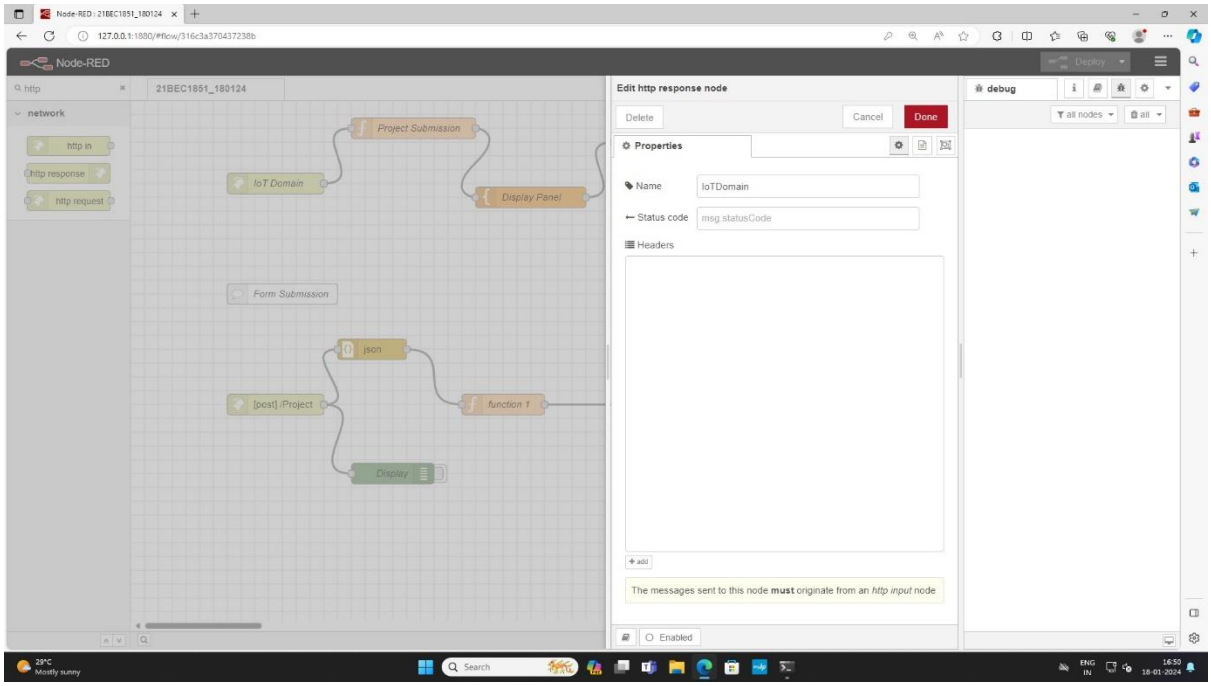
Function Node:



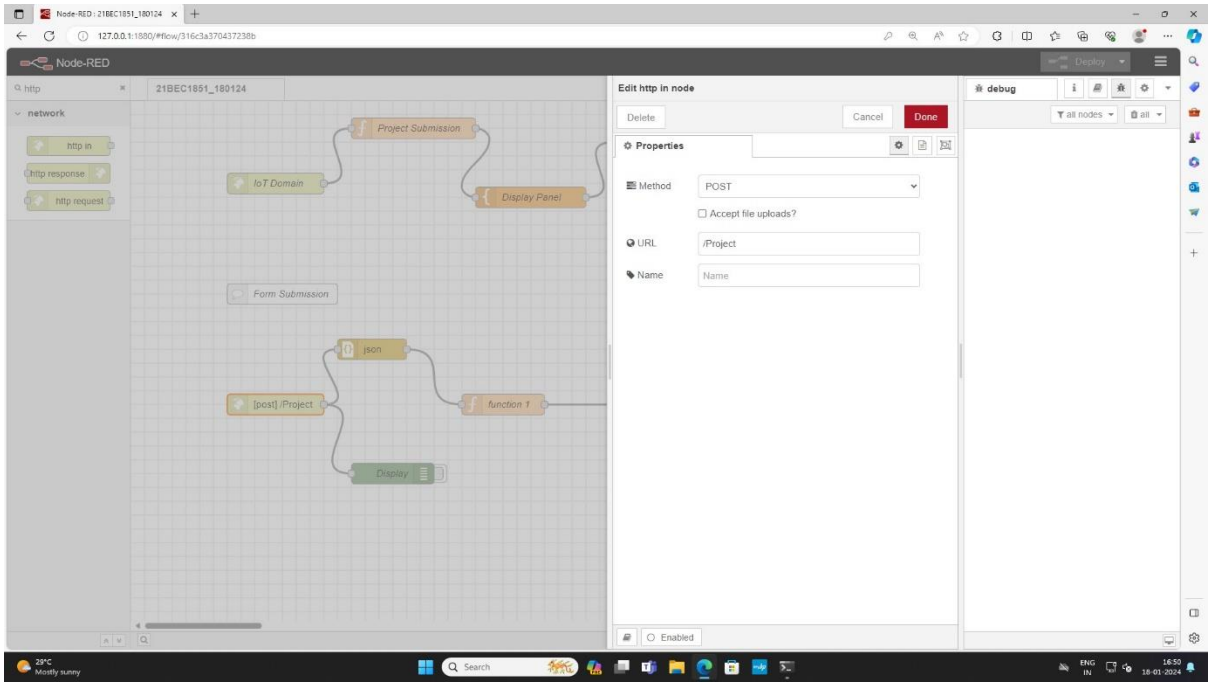
Template Node:



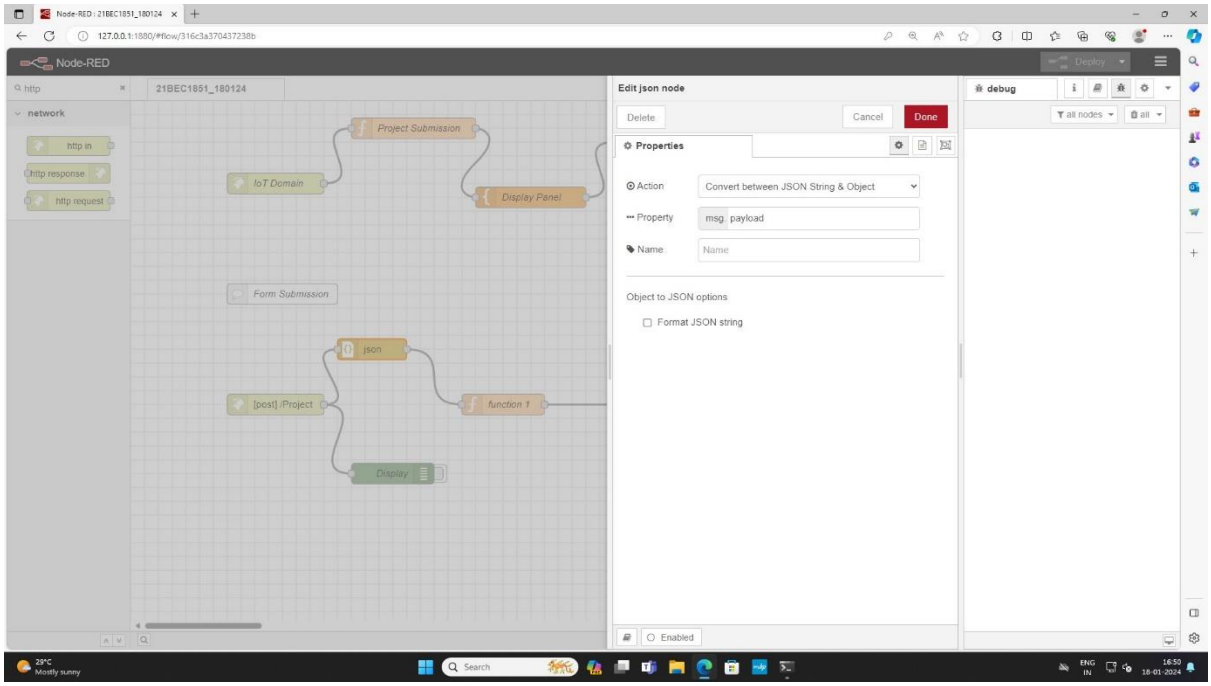
HTTP Response Node:



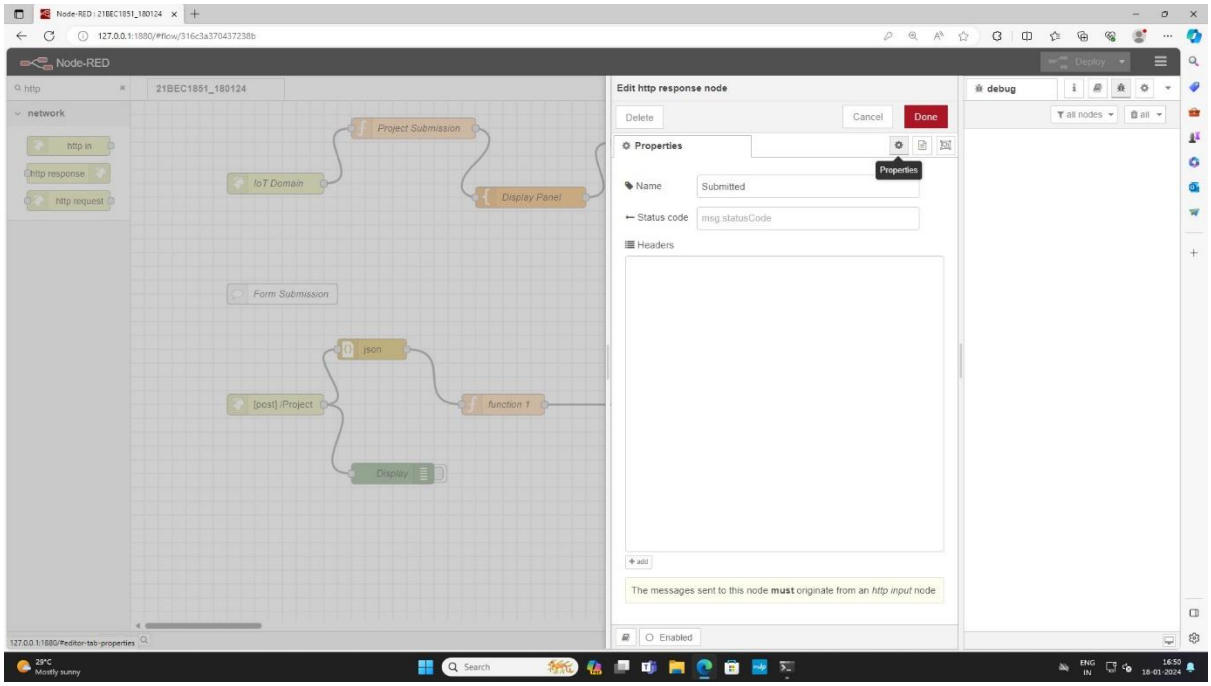
HTTP in Node:



JSON Node:



HTTP Response Node:

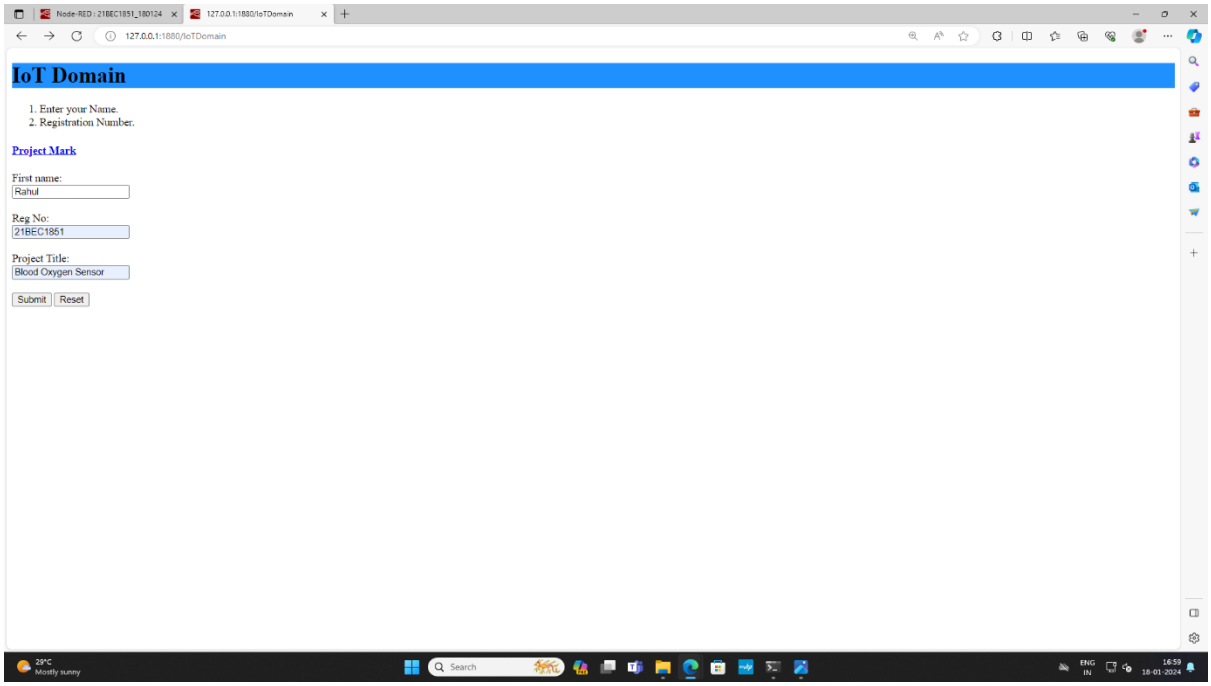


HTML Code:

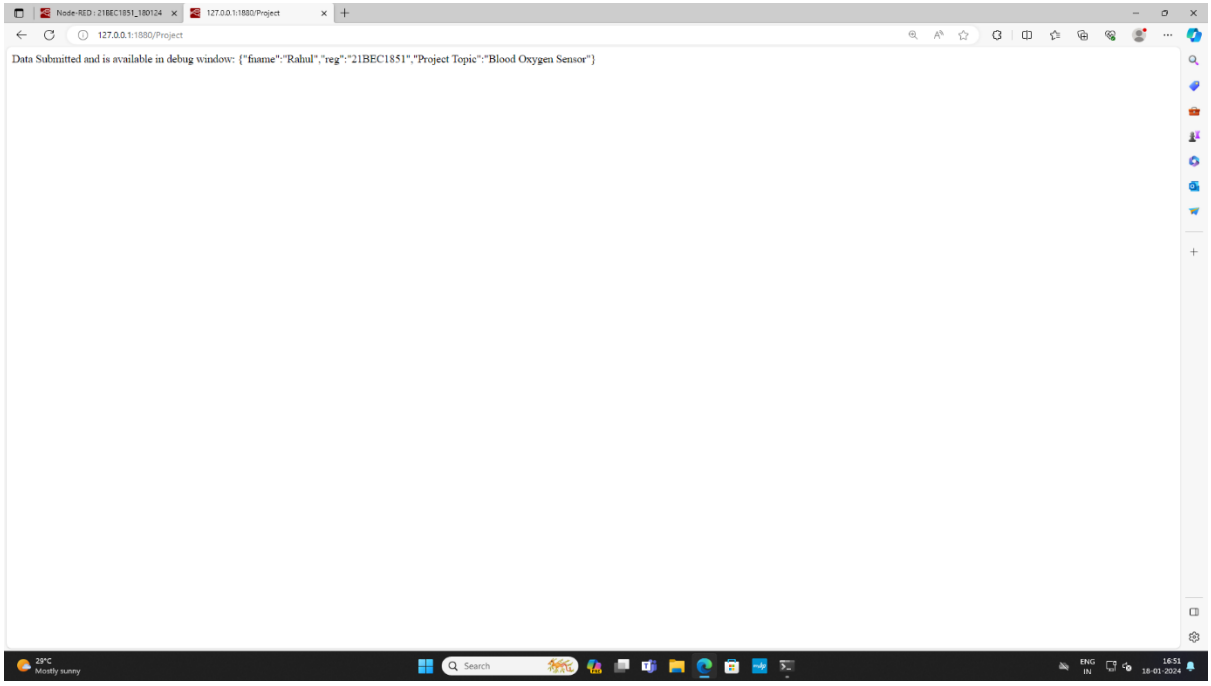
```
<!DOCTYPE html>
<html>
	<head>
		<h1 style="background-color:DodgerBlue;">IoT Domain</h1>
	</head>
	<body>
		<p style="background-color:Tomato;">
		<OL>
			<LI> Enter your Name.
			<LI> Registration Number.
		</OL>
		<h4>
			<a href="https://projectmark.com/"> Project Mark</a>
		</h4>
		<form method="post" 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>
```

Output:

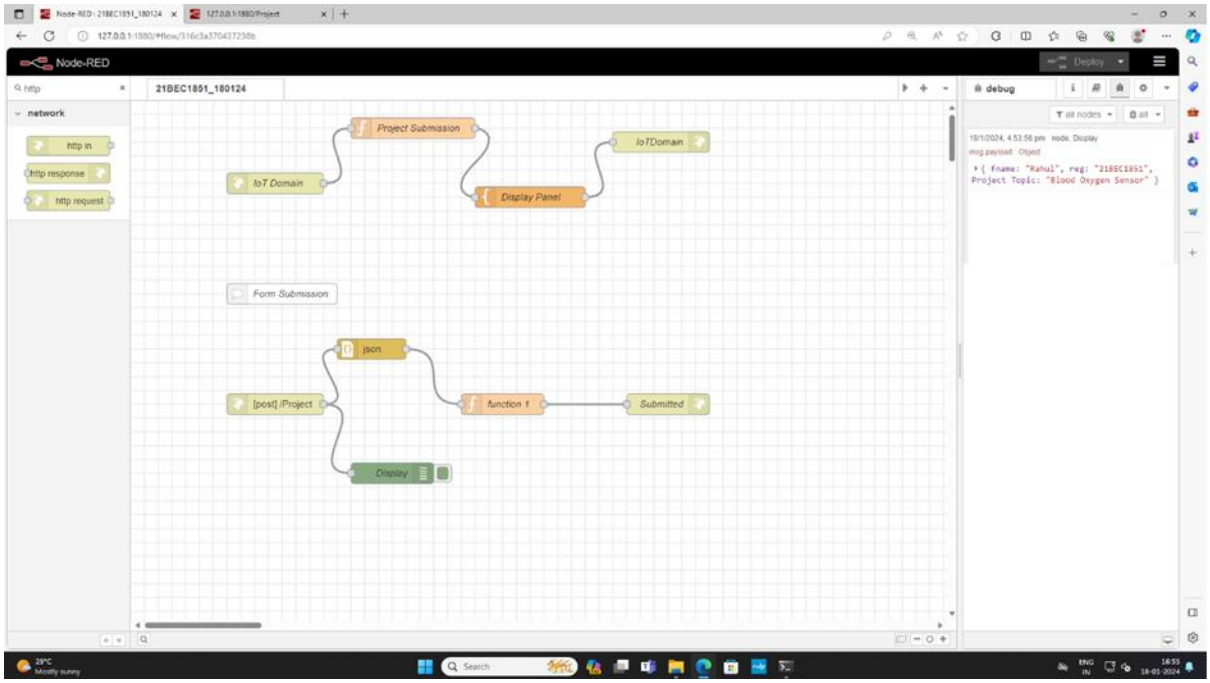
HTML Form:



Output in Project Page After Submitting the form:



Output in Node-RED Debug Window:



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 in Node-Red and have successfully obtained all the outputs for the same.

Name: Rahul Karthik S
Registration Number: 21BEC1851
Course: IoT Domain Analyst (BECE352E)
Slot: L51+L52

Experiment 3

Air Quality Monitoring System using NodeMCU ESP8266 and MQ135

Aim:

To monitor the Air Quality Index (AQI) on Thingspeak server using ESP8266 and MQ135 Air Quality sensor.

Software Required:

Arduino IDE, Thingspeak Software

Tools Required:

NodeMCU ESP8266 Development Board, MQ135 gas sensor, Jumper wires, Breadboard, USB cable.

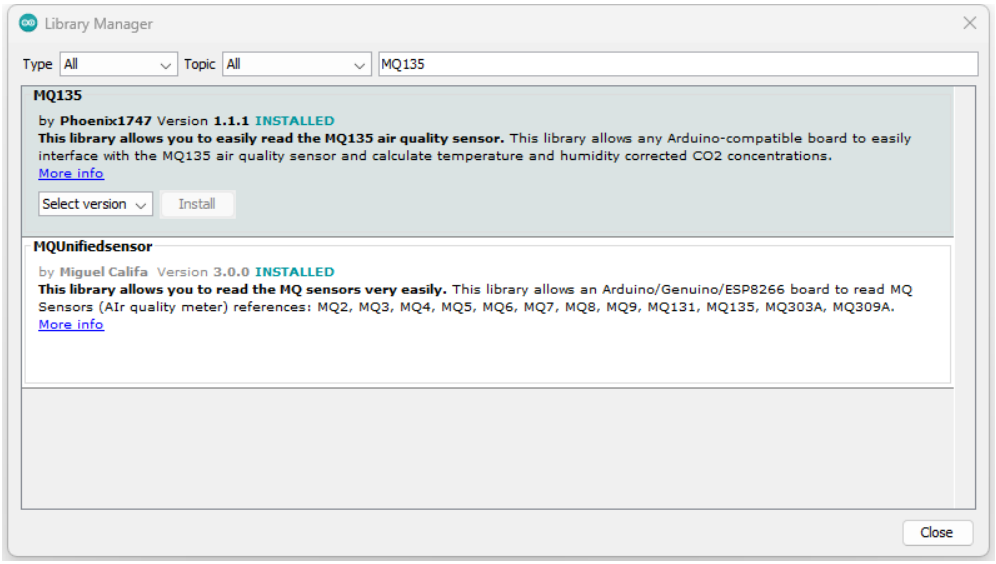
Theory:

- The AQI is an index that indicates how clean or polluted the air is, and what associated health effects might be a concern.
- The MQ135 gas sensor senses the gases like ammonia nitrogen, oxygen, alcohols, aromatic compounds, sulphide and smoke.
- The MQ135 sensor is a signal output indicator instruction. It has two outputs: Analog output and TTL output.
- The Analog output is a concentration, i.e. increasing voltage is directly proportional to increasing concentration. This sensor has a long life and reliable stability as well.
- The TTL output is low signal light which can be accessed through the IO ports on the Microcontroller.

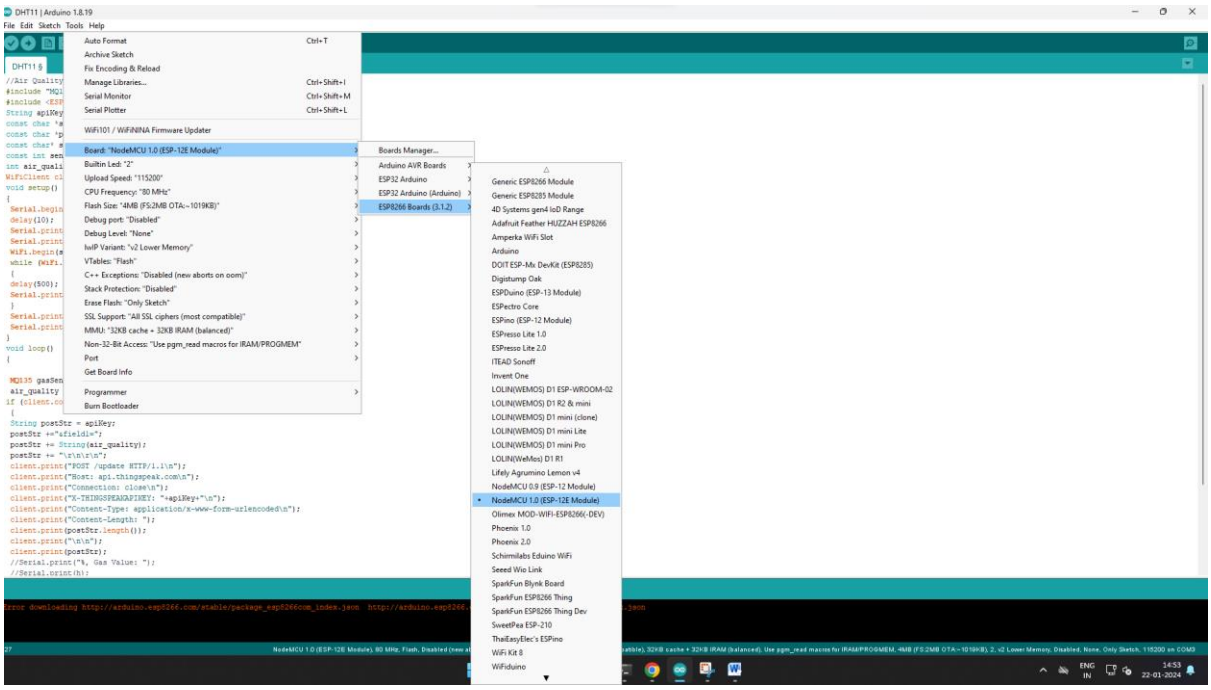
Procedure:

- Open Arduino IDE in the system.
- Go to File → New → Sketch → Code.
- Go to Tools → Manage Libraries → Install MQ135, DHT and ESP8266.

21BEC1851 – Rahul Karthik S



- For MQ135, Go to Search type MQ135, Version 1.1.1 → Install.
- For DHT, Go to DHT Sensor Library, Version 1.4.6 → Install.
- For ESP8266, Go to Preferences and type the URL Enter “http://arduino.esp8266.com/stable/package_esp8266com_index.json”in the Additional Boards Manager URLs.
- Connect the NodeMCU ESP8266 to your computer using a USB cable.
- Write a program in Arduino IDE to read data from the MQ135 sensor and transmit it the Thingspeak server.
- Select the correct board (NodeMCU ESP8266) and port from the Tools menu in Arduino IDE.



- Click on the Compile and Upload button to upload the code to the NodeMCU ESP8266.
- After uploading the code successfully, open the Serial Monitor in Arduino IDE to view the air quality readings being printed.
- You can also monitor the data on the Thingspeak server by checking the corresponding channel.

21BEC1851 – Rahul Karthik S

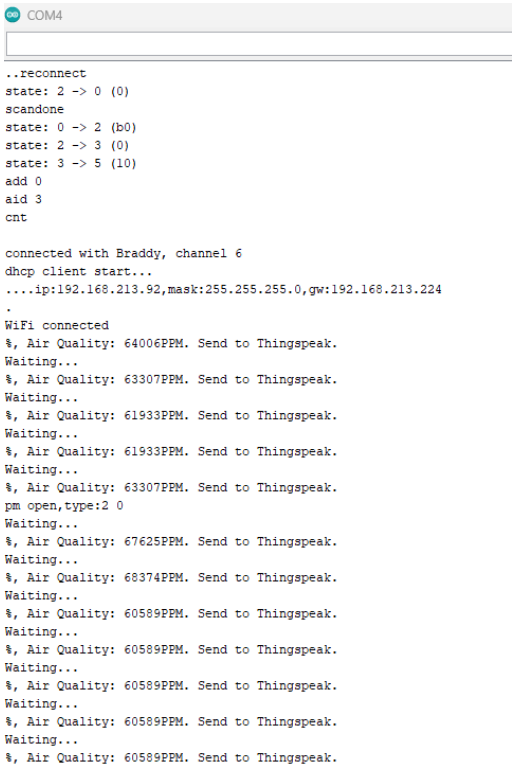
Arduino Code:

```
#include "MQ135.h"
#include <ESP8266WiFi.h>
String apiKey = " "; // Enter your Write API key from ThingSpeak
const char *ssid =" "; // replace with your wifi ssid and wpa2 key
const char *pass =" ";
const char* server = "api.thingspeak.com";
const int sensorPin= 0;
int air_quality;
WiFiClient client;
void setup()
{
	Serial.begin(115200);
	delay(10);
	Serial.println("Connecting to ");
	Serial.println(ssid);
	WiFi.begin(ssid, pass);
	while (WiFi.status() != WL_CONNECTED)
	{
		delay(500);
		Serial.print(".");
	}
	Serial.println("");
	Serial.println("WiFi connected");
}
void loop()
{
	MQ135 gasSensor = MQ135(A0);
	air_quality = gasSensor.getPPM();
	if (client.connect(server,80)) // "184.106.153.149" or api.thingspeak.com
	{
		String postStr = apiKey;
		postStr +="&field1=";
		postStr += String(air_quality);
		postStr += "\r\n\r\n";
		client.print("POST /update HTTP/1.1\n");
		client.print("Host: api.thingspeak.com\n");
		client.print("Connection: close\n");
		client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
		client.print("Content-Type: application/x-www-form-urlencoded\n");
		client.print("Content-Length: ");
		client.print(postStr.length());
		client.print("\n\n");
		client.print(postStr);
```

```
                //Serial.print("%, Gas Value: ");
                //Serial.print(h);
                Serial.print("%, Air Quality: ");
                Serial.print(air_quality);
                Serial.println("PPM. Send to Thingspeak.");
        }
        client.stop();
        Serial.println("Waiting...");
        // thingspeak needs minimum 15 sec delay between updates
        delay(1000);
}
```

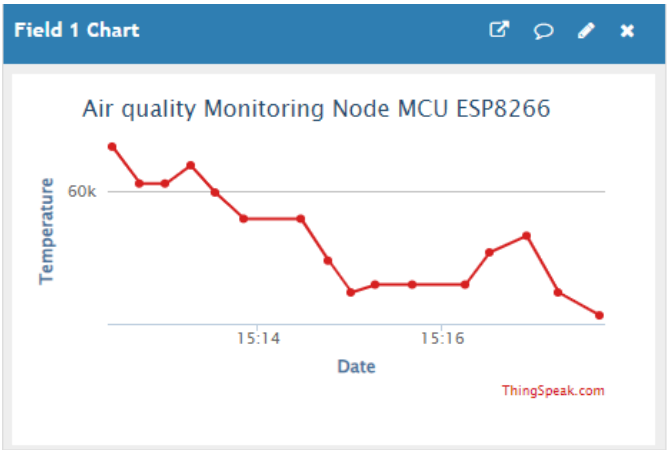
Output:

Arduino IDE:



Thingspeak:

21BEC1851 – Rahul Karthik S



Inference:

In this experiment, we employed the NodeMCU ESP8266 and MQ135 gas sensor to track air quality. By gauging gas concentrations, especially hazardous ones like carbon dioxide, we evaluated overall air quality. Analysing collected data over time unveiled fluctuations and trends, offering crucial insights into environmental health. This enables us to grasp the risks posed by air pollution and implement measures for better air quality and well-being.

Result:

Thus, the experiment has been conducted, and the connection between the Arduino board and the Thingspeak server has been successfully established. Sensor readings have been confirmed on both platforms.

Name: Rahul Karthik S
Registration Number: 21BEC1851
Course: IoT Domain Analyst (BECE352E)
Slot: L51+L52

Experiment 4

Smart Home Automation using Cisco Packet Tracer

Aim:

To create an IoT based Smart Home Automation environment in Cisco Packet Tracer.

Software Required:

Cisco Packet Tracer

Theory:

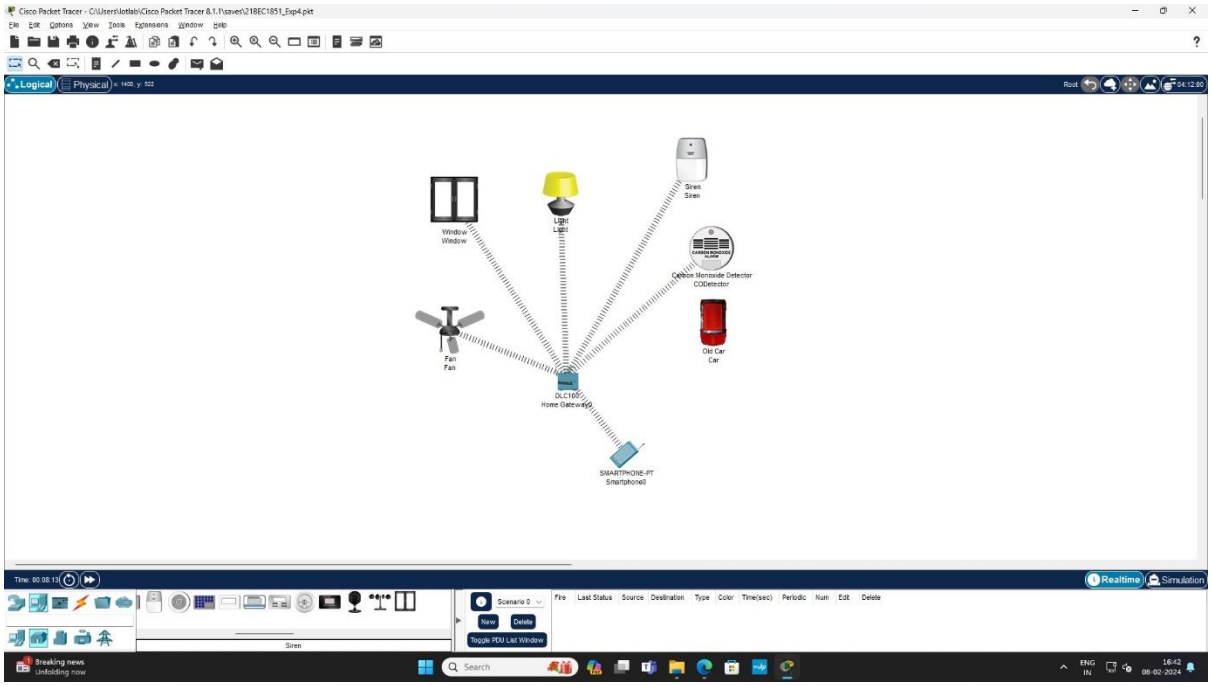
- Cisco Packet Tracer serves as a network simulation tool, allowing users to design, configure, and troubleshoot network setups.
- Supporting various networking devices like routers, switches, and end devices, it facilitates learning by simulating real-world network scenarios.
- This tool is widely utilized in educational settings for teaching networking concepts, enabling users to experiment with different configurations without affecting real networks.

Procedure:

- Launch the Cisco Packet Tracer application.
- Set up the devices according to the specified configuration. This likely involves adding a home gateway device, a smart phone, and other IoT Devices.
- Configure the SSID (Service Set Identifier) in the smart phone to match the SSID set in the home gateway device. This ensures seamless connectivity between the smart phone and the home network.
- Ensure that the IoT server of the home device is selected as the “Home Gateway”. This step is crucial for proper communication between the smart phone and the IoT devices within the home network.
- In the smart phone settings, select "Desktop" and then proceed to install or select the IoT monitor app. This app will allow you to monitor and control the IoT devices connected to the home network.
- Log in to the IoT monitor app using the provided credentials. Once logged in, you should see options to control the home devices, allowing you to manage and monitor them remotely from your smart phone.
- Then, go to configurations and configure for home automation.

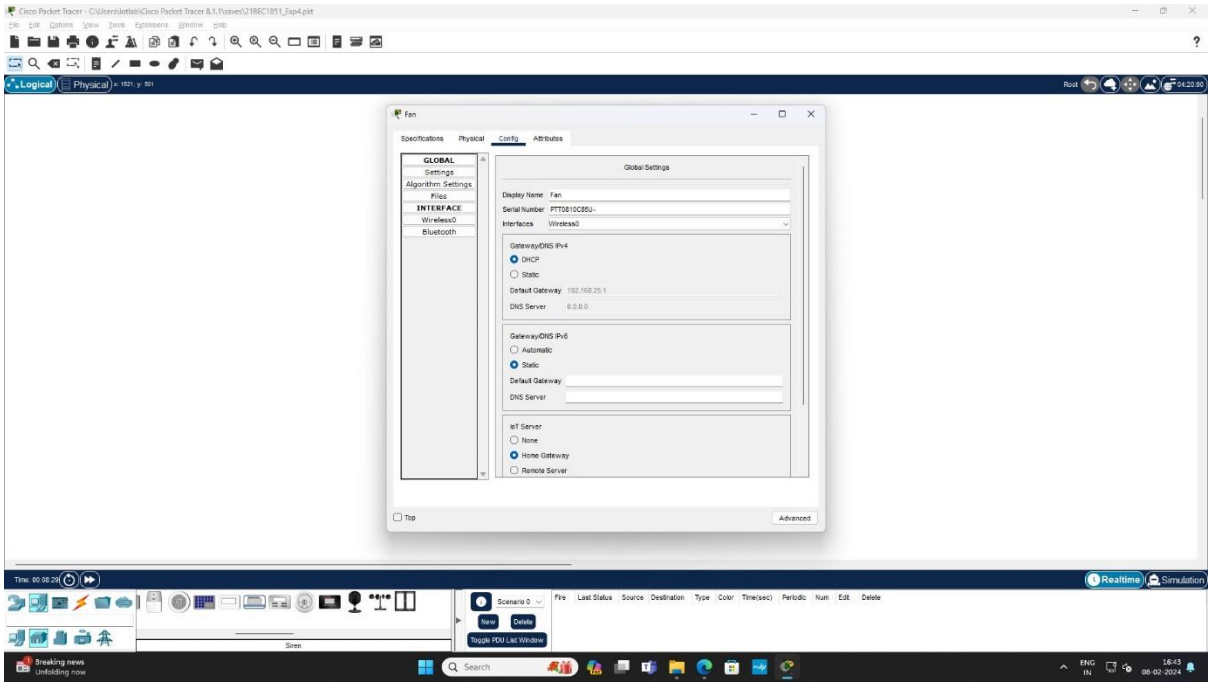
Circuit Diagram:

21BEC1851 – Rahul Karthik S



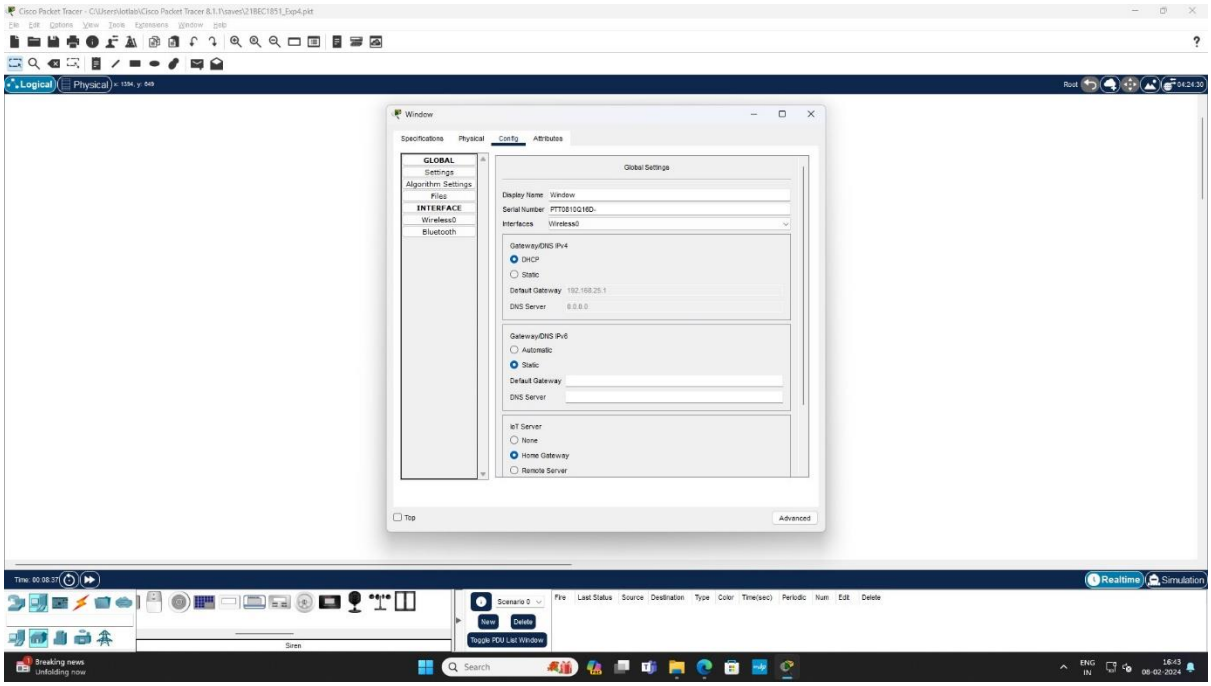
Properties:

Fan:

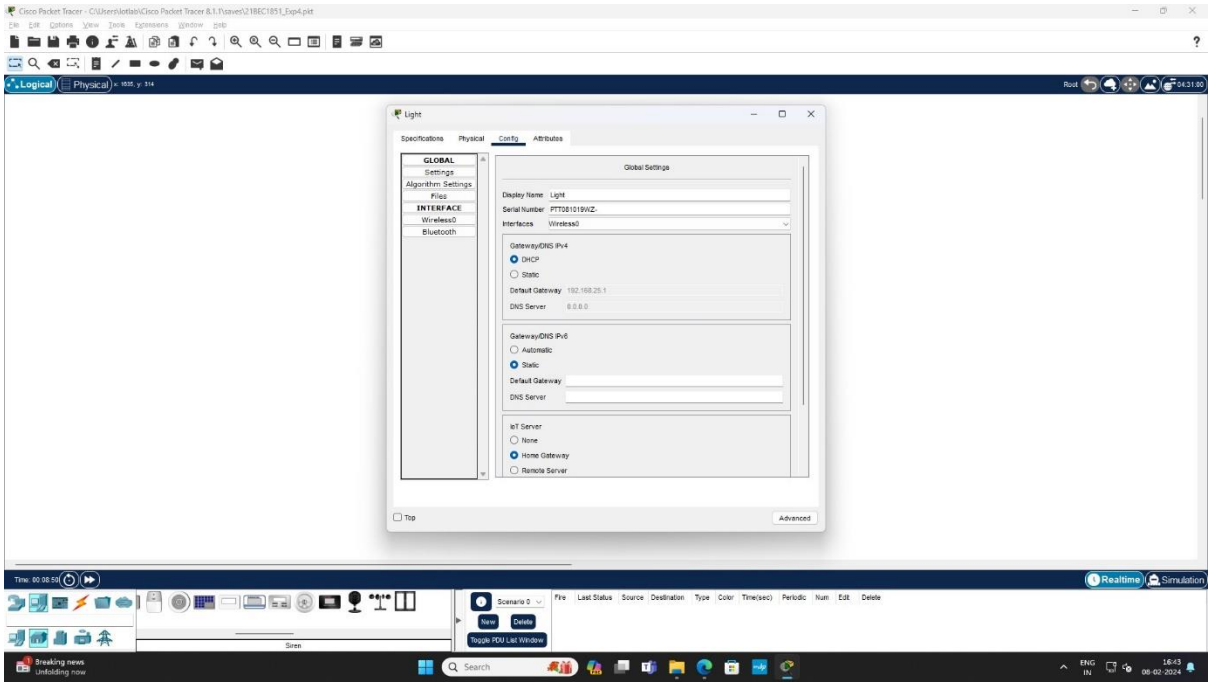


21BEC1851 – Rahul Karthik S

Window:

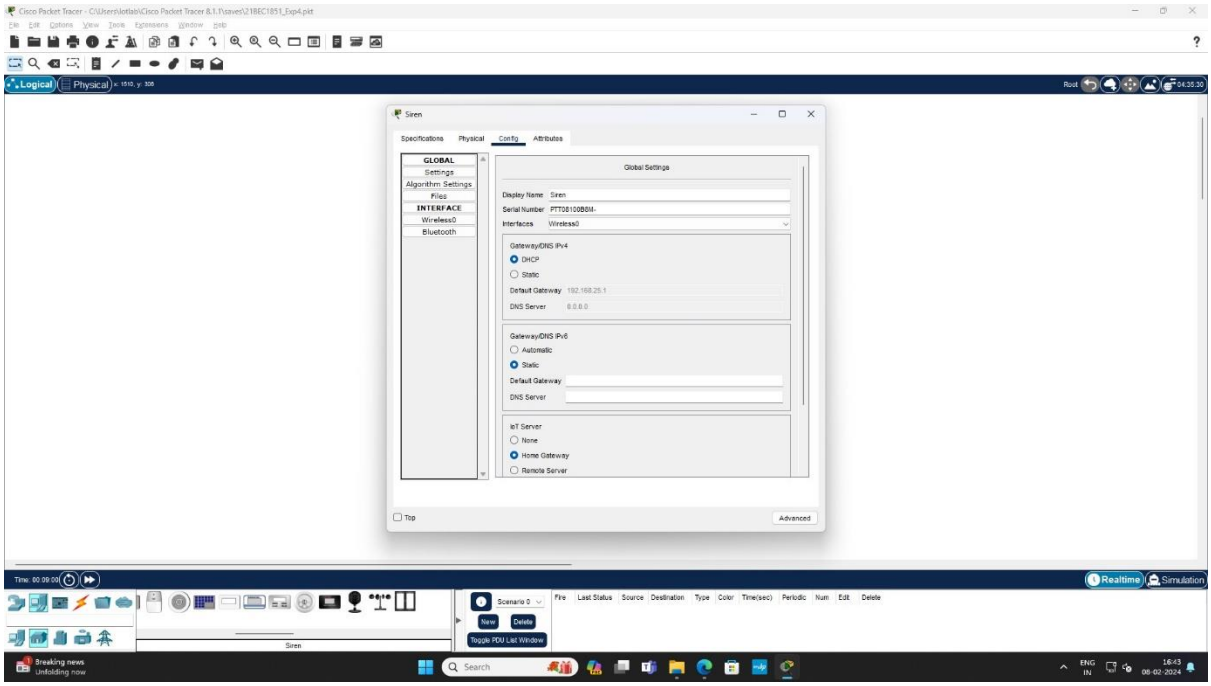


Light:

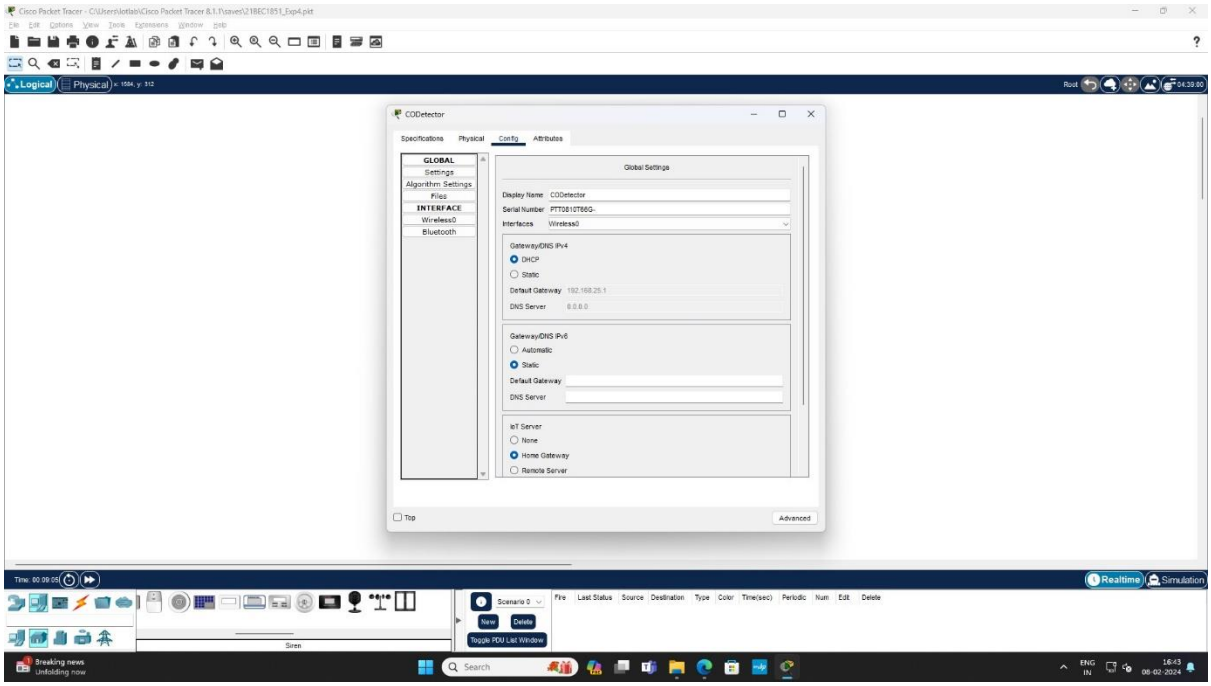


21BEC1851 – Rahul Karthik S

Siren:

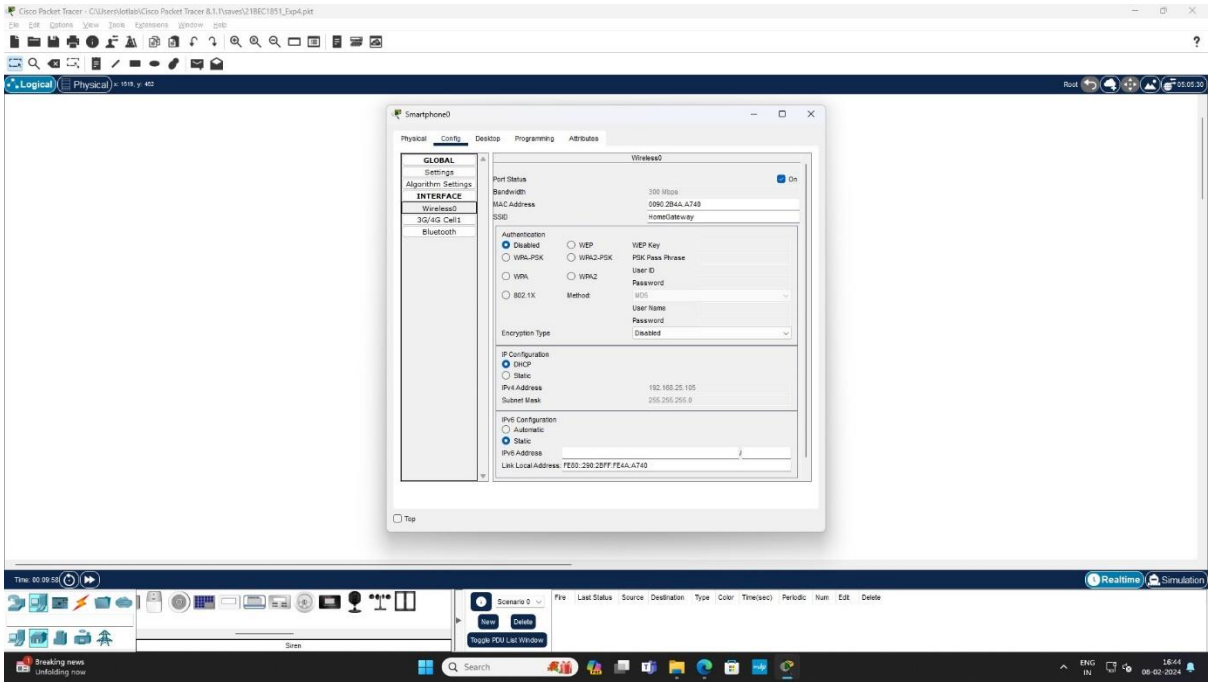


Carbon Monoxide Detector:

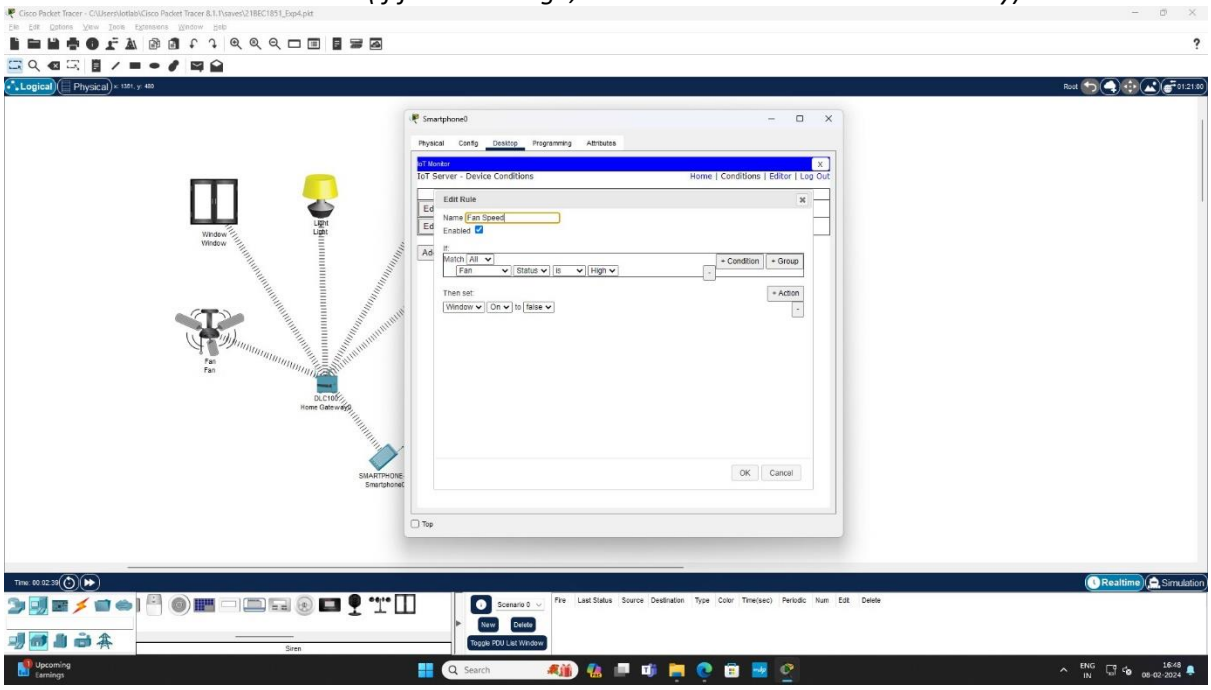


21BEC1851 – Rahul Karthik S

Smartphone:

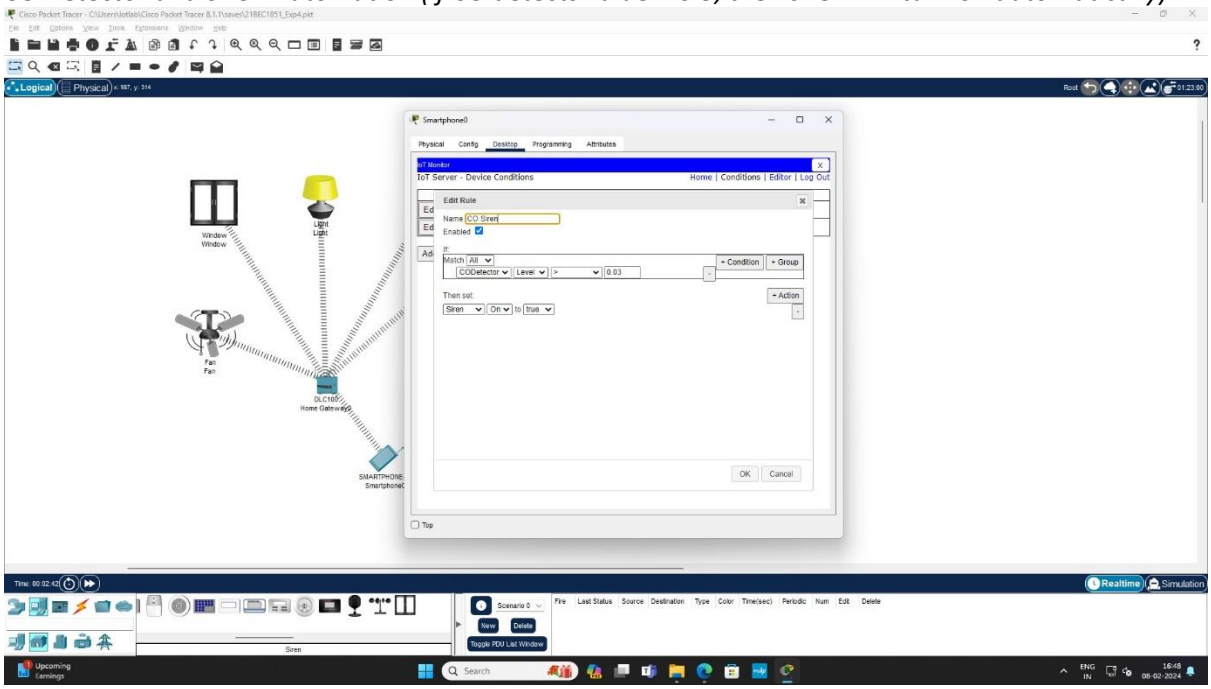


Fan and Window Automation: (if fan set to high, window will be closed automatically)



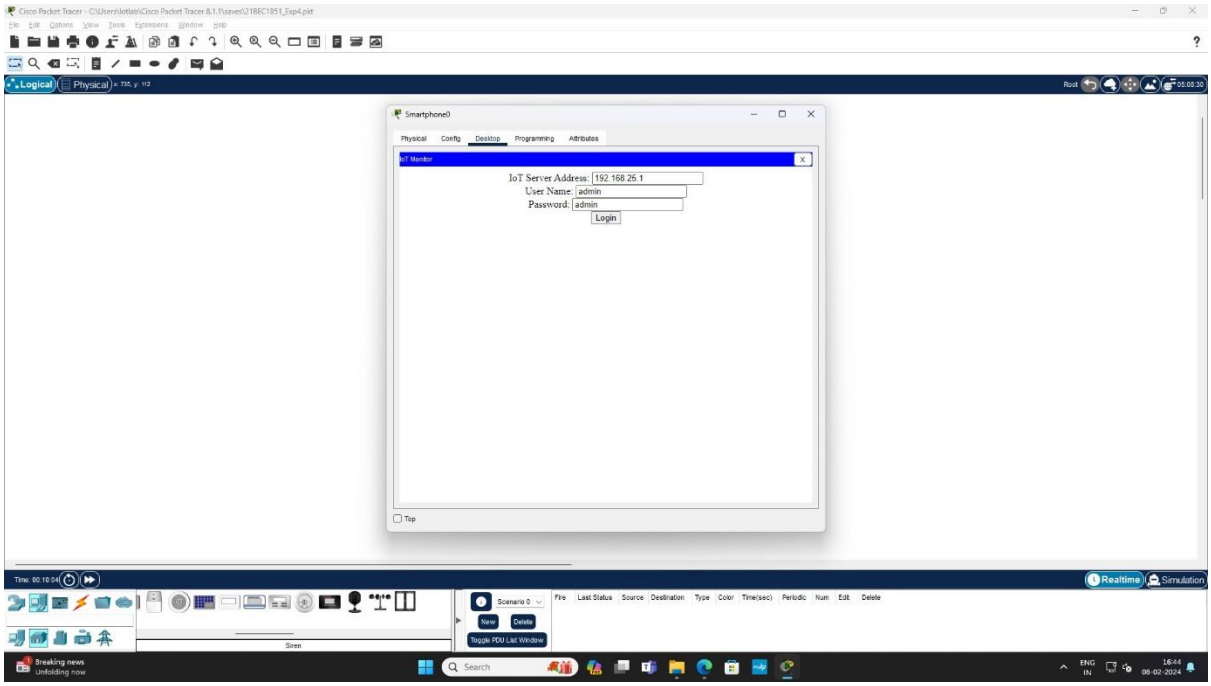
21BEC1851 – Rahul Karthik S

CO Detector and Siren Automation (if CO detects value > 0.3, then siren will turn on automatically)



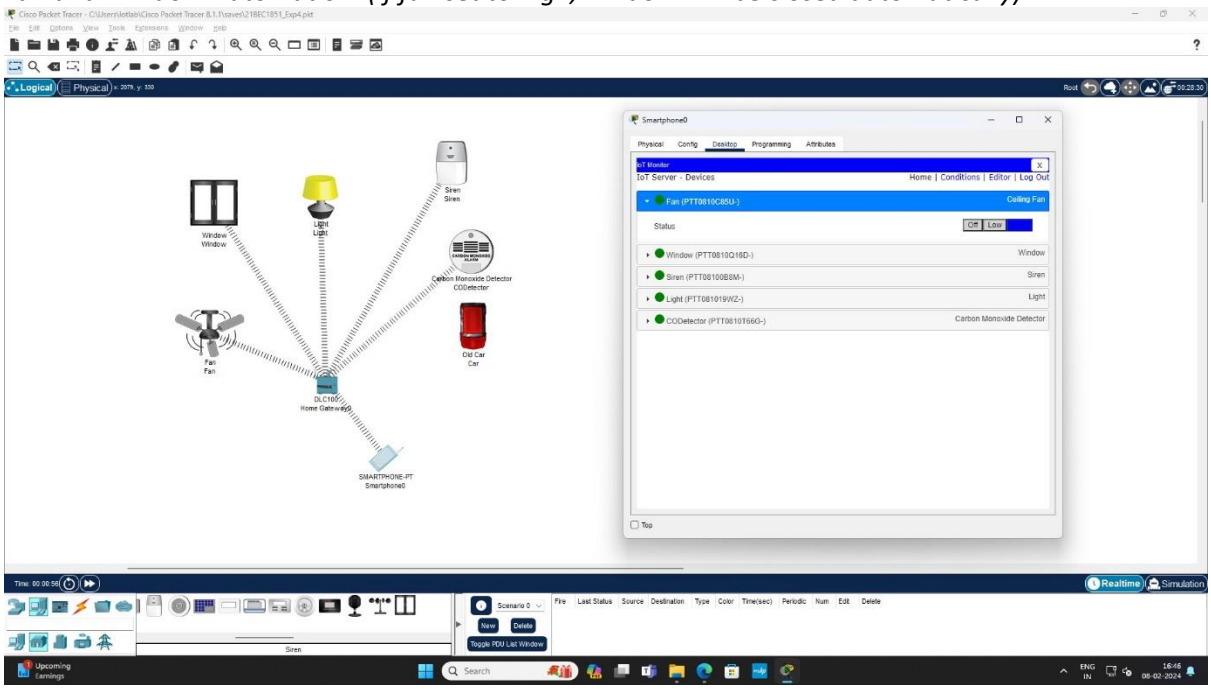
Output:

IoT Monitor:

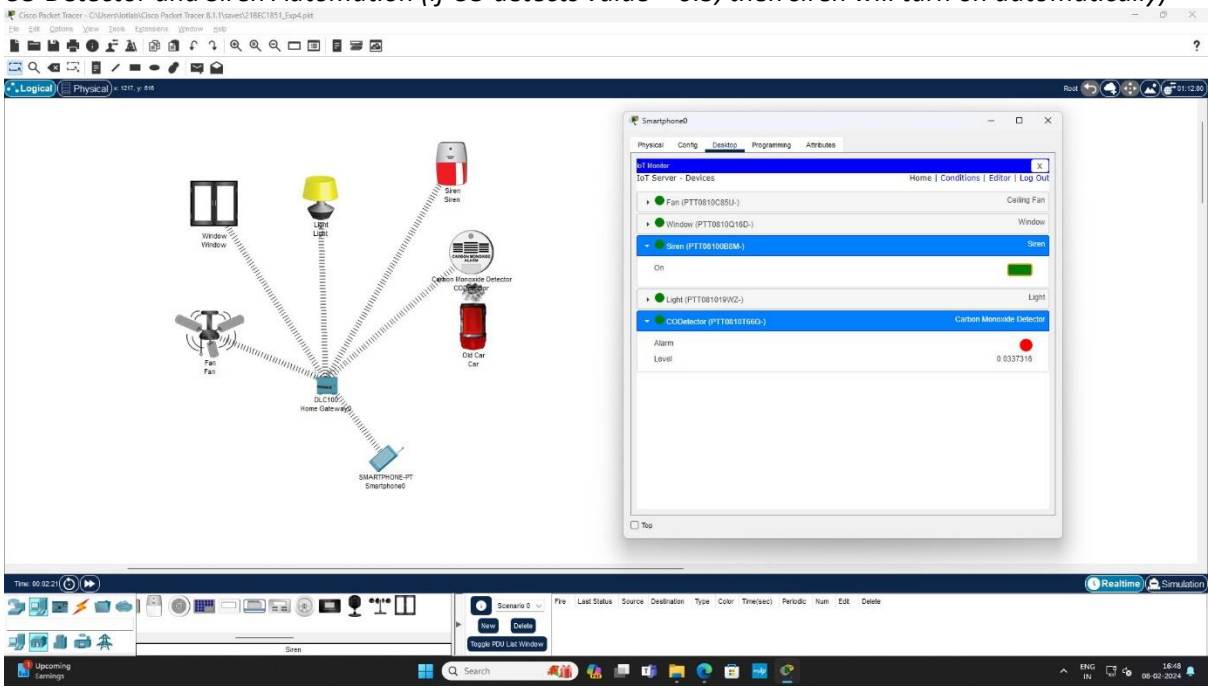


21BEC1851 – Rahul Karthik S

Fan and Window Automation: (if fan set to high, window will be closed automatically)



CO Detector and Siren Automation (if CO detects value > 0.3, then siren will turn on automatically)



Result:

Thus, the connection between the devices and smartphone is established and the experiment is executed successfully.

21BEC1851 – Rahul Karthik S

Name: Rahul Karthik S
Registration Number: 21BEC1851
Course: IoT Domain Analyst (BECE352E)
Slot: L51+L52

Experiment 5

Knime Analytics Platform

Aim:

To Perform the Data Manipulation in Knime like Row Filter, Column Filter and displaying Pie Chart and Area Chart.

Software Required:

Knime Analytics Platform

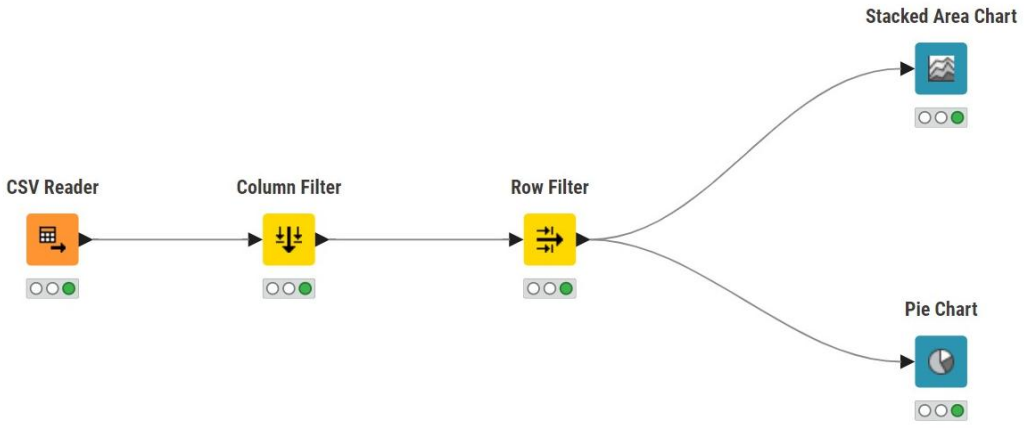
Theory:

- KNIME Analytics Platform is a free tool that's easy to use for analysing data.
- You can bring in data from lots of different places and clean it up easily with KNIME.
- It has tools to help you look at your data and do things like math or teach computers.
- You can show off what you find with KNIME in different ways.
- KNIME can handle really big sets of data and can work on your computer or on the internet.
- KNIME lets people work together on projects and talk about what they're doing with others.

Procedure:

- Install and open KNIME Analytics Platform on your computer.
- Import the CSV File into KNIME.
- Clean up your data by dealing with any missing info, unusual bits, or repeated entries.
- Here, the Column Filter and Row Filter Nodes are used to remove unwanted data.
- If needed, create new features to make your analysis better.
- Display the analysed data with the help of graphs and plots.
- Understand what the data is telling you and figure out what actions to take based on that.

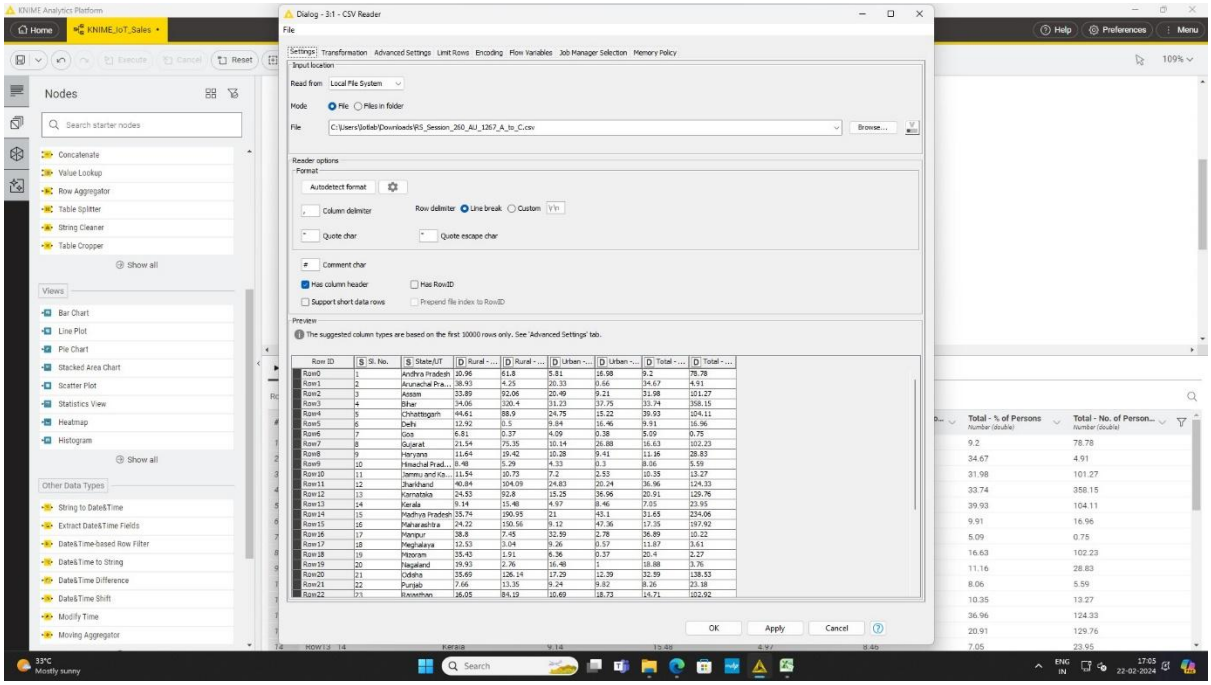
Flow Diagram:



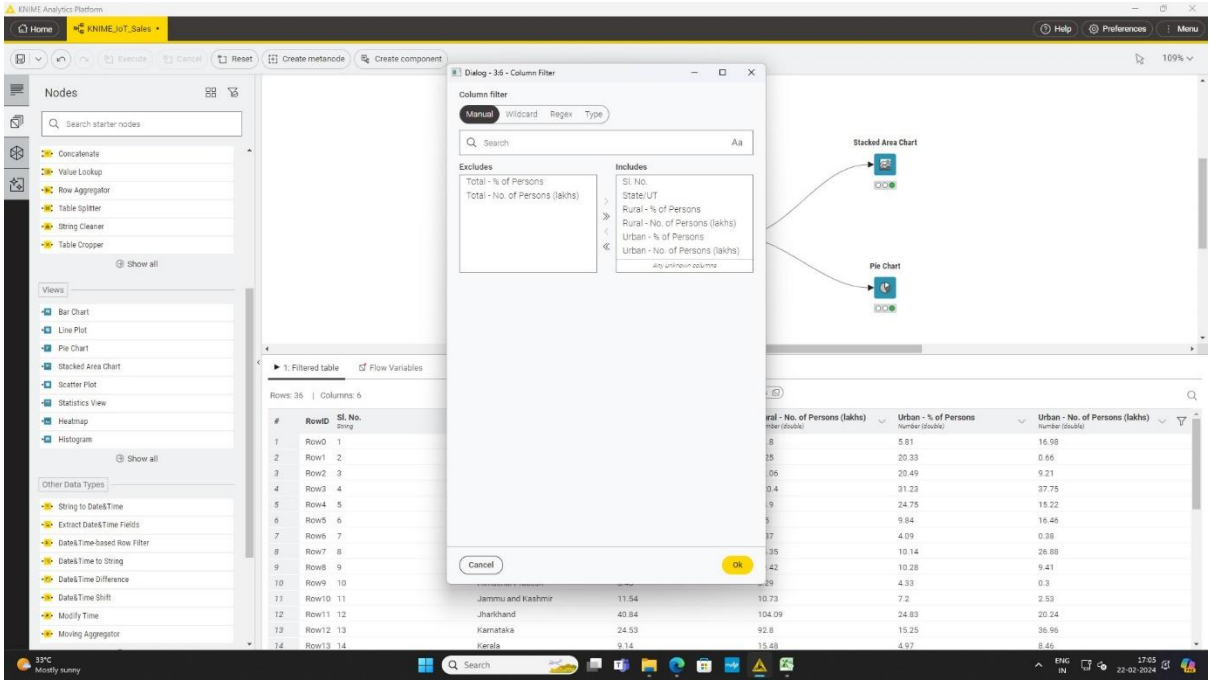
21BEC1851 – Rahul Karthik S

Properties:

CSV Reader:

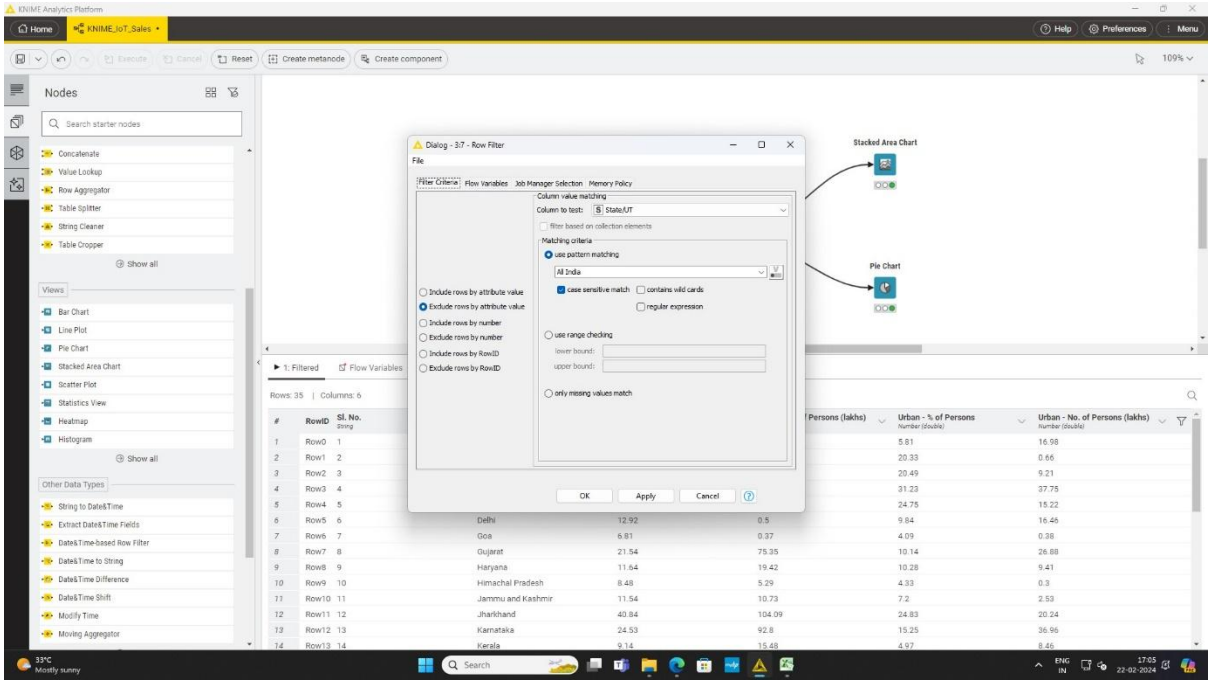


Column Filter:



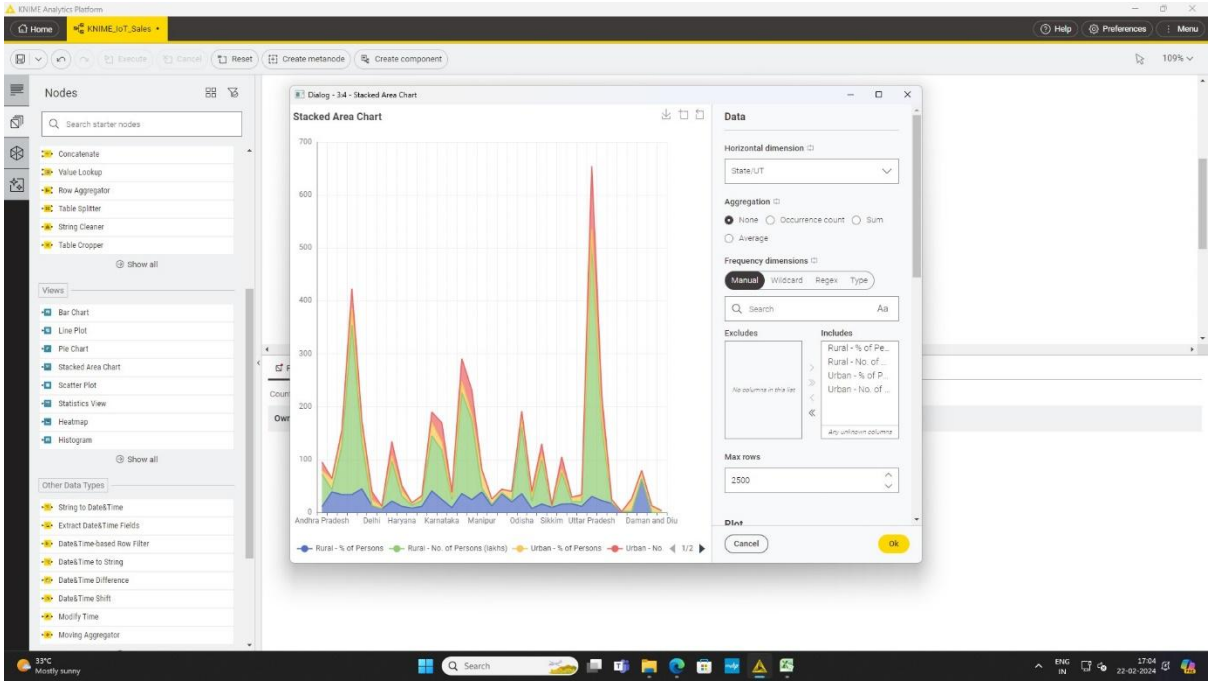
Row Filter:

21BEC1851 – Rahul Karthik S



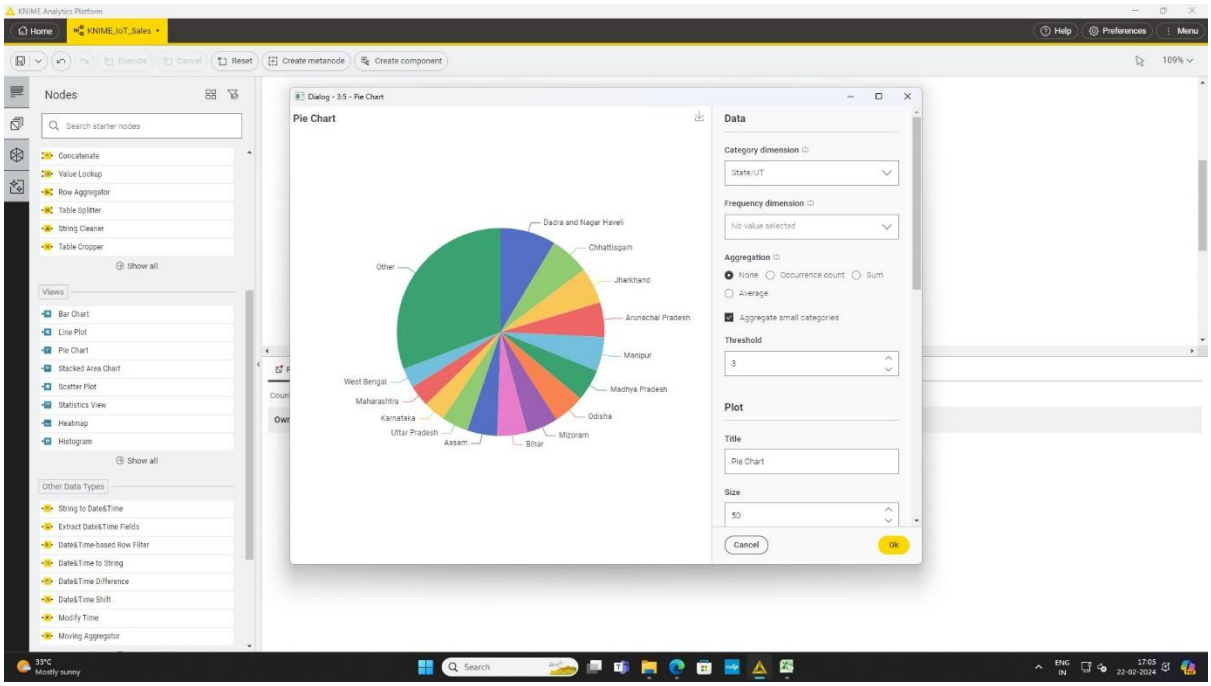
Output:

Stacked Area Chart:



Pie Chart:

21BEC1851 – Rahul Karthik S



Result:

The experiment was successfully simulated on KNIME Analytics Platform and the outputs were verified.

Inference:

To sum up, using KNIME Analytics Platform for sales analysis helps manage data well, understand it better, and share findings clearly. This shows how versatile, adaptable, and easy-to-use the platform is for making smart decisions based on data, improving sales plans, and making businesses grow.

21BEC1851 – Rahul Karthik S

Name: Rahul Karthik S
Registration Number: 21BEC1851
Course: IoT Domain Analyst (BECE352E)
Slot: L51+L52

Experiment 6

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

Aim:

To Perform the Data Manipulation in Knime like String Manipulation, Math Formula and Rule Engine.

Software Required:

Knime Analytics Platform

Theory:

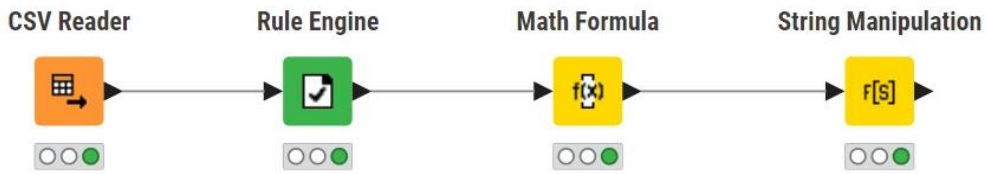
KNIME is a helpful tool for working with data. It has special parts called nodes that do different jobs.

- String Manipulation: This node helps change text in your data. You can cut out parts, switch letters, change the capitalization, and more.
- Math Formula: This node helps with numbers. You can make fancy calculations, like adding, subtracting, finding averages, and other math stuff on your data.
- Rule Engine: This node lets you set up rules for your data. You can decide what to do with the data based on certain conditions you set. For example, if a number is bigger than 10, do one thing, but if it's smaller, do another.

Procedure:

- Start by bringing in the CSV dataset with both words and numbers into KNIME.
- With the Rule Engine node, you can set up rules for your data. Decide what to do with the data if certain conditions are met, like filtering out some rows or changing them in some way.
- Use the String Manipulation node to change the text in different ways, like cutting parts out, changing letters, or making all the letters capital.
- Use the Math Formula node to do math stuff on the numbers in your dataset. You can make new numbers based on the ones you already have or do things like find averages.
- Run your plan in KNIME to make all these changes happen to your dataset.

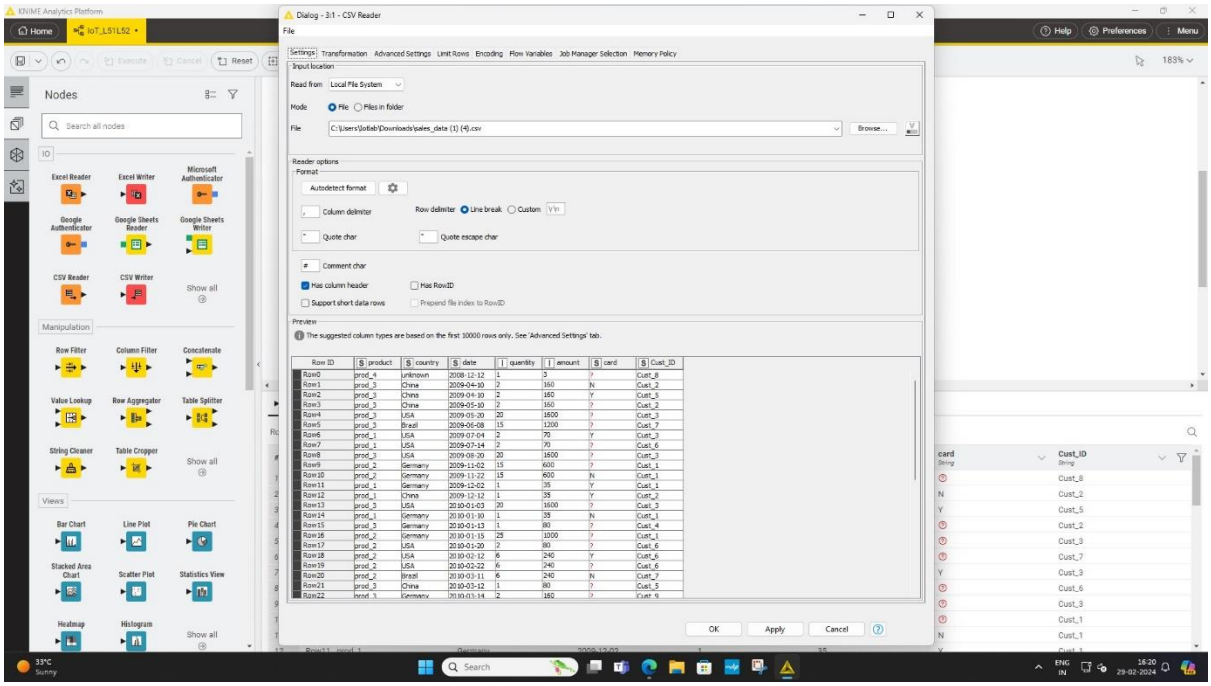
Flow:



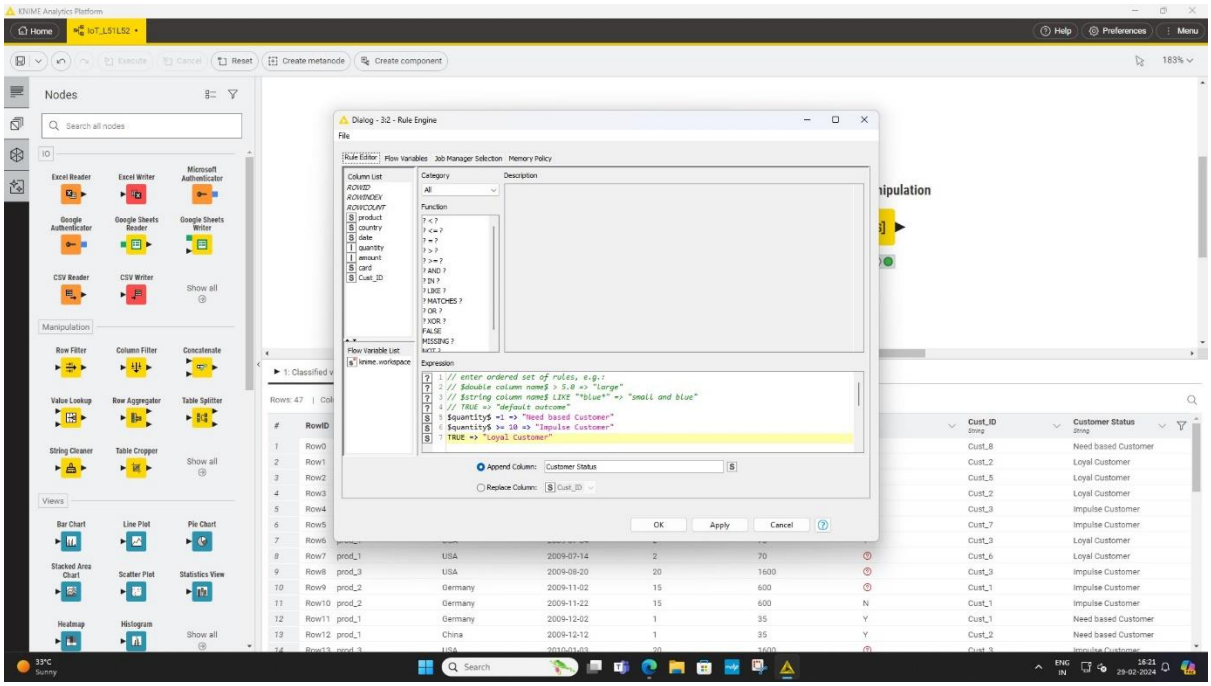
Properties:

CSV Reader:

21BEC1851 – Rahul Karthik S

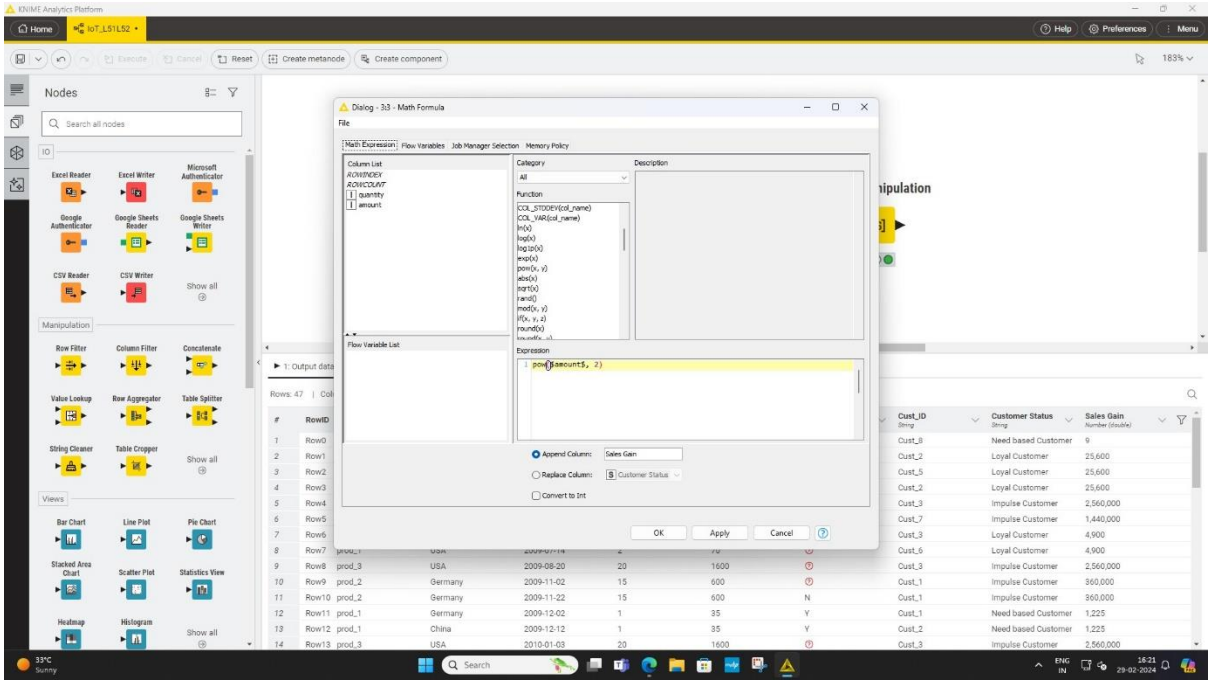


Rule Engine:

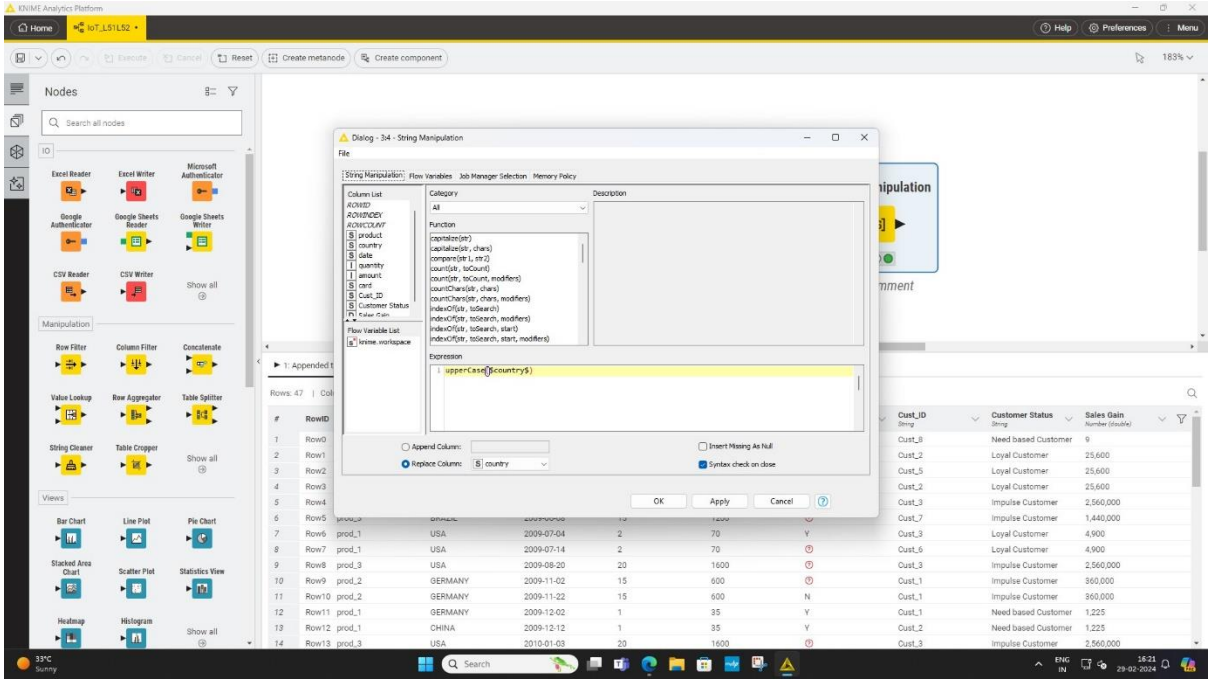


Math Formula:

21BEC1851 – Rahul Karthik S

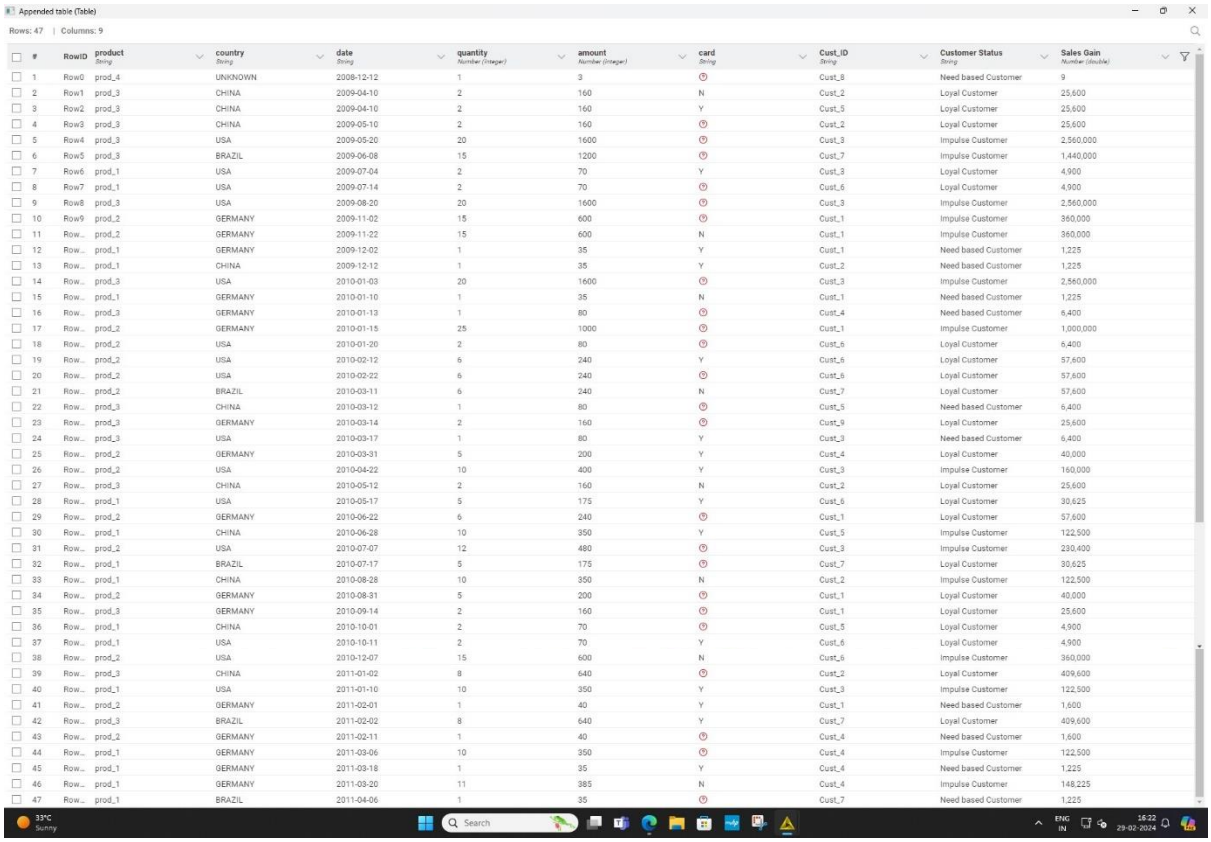


String Manipulation:



Output:

21BEC1851 – Rahul Karthik S



Appended table (Table)

Rows: 47 | Columns: 9

#	RowID	product	country	date	quantity	amount	card	Cust_ID	Customer Status	Sales Gain
1	Row0	prod_4	UNKNOWN	2008-12-12	1	3	?	Cust_8	Need based Customer	9
2	Row1	prod_3	CHINA	2009-04-10	2	160	N	Cust_2	Loyal Customer	25,600
3	Row2	prod_3	CHINA	2009-04-10	2	160	Y	Cust_5	Loyal Customer	25,600
4	Row3	prod_3	CHINA	2009-05-10	2	160	?	Cust_2	Loyal Customer	25,600
5	Row4	prod_3	USA	2009-05-20	20	1600	?	Cust_3	Impulse Customer	2,560,000
6	Row5	prod_3	BRAZIL	2009-06-08	15	1200	?	Cust_7	Impulse Customer	1,440,000
7	Row6	prod_1	USA	2009-07-04	2	70	Y	Cust_3	Loyal Customer	4,900
8	Row7	prod_1	USA	2009-07-14	2	70	?	Cust_6	Loyal Customer	4,900
9	Row8	prod_3	USA	2009-08-20	20	1600	?	Cust_3	Impulse Customer	2,560,000
10	Row9	prod_2	GERMANY	2009-11-02	15	600	?	Cust_1	Impulse Customer	360,000
11	Row...	prod_2	GERMANY	2009-11-22	15	600	N	Cust_1	Impulse Customer	360,000
12	Row...	prod_1	GERMANY	2009-12-02	1	35	Y	Cust_1	Need based Customer	1,225
13	Row...	prod_1	CHINA	2009-12-12	1	35	Y	Cust_2	Need based Customer	1,225
14	Row...	prod_3	USA	2010-01-03	20	1600	?	Cust_3	Impulse Customer	2,560,000
15	Row...	prod_1	GERMANY	2010-01-10	1	35	N	Cust_1	Need based Customer	1,225
16	Row...	prod_3	GERMANY	2010-01-13	1	80	?	Cust_4	Need based Customer	6,400
17	Row...	prod_2	GERMANY	2010-01-15	25	1000	?	Cust_1	Impulse Customer	1,000,000
18	Row...	prod_2	USA	2010-01-20	2	80	?	Cust_6	Loyal Customer	6,400
19	Row...	prod_2	USA	2010-02-12	6	240	Y	Cust_6	Loyal Customer	57,600
20	Row...	prod_2	USA	2010-02-22	6	240	?	Cust_6	Loyal Customer	57,600
21	Row...	prod_2	BRAZIL	2010-03-11	6	240	N	Cust_7	Loyal Customer	57,600
22	Row...	prod_3	CHINA	2010-03-12	1	80	?	Cust_5	Need based Customer	6,400
23	Row...	prod_3	GERMANY	2010-03-14	2	160	?	Cust_9	Loyal Customer	25,600
24	Row...	prod_3	USA	2010-03-17	1	80	Y	Cust_3	Need based Customer	6,400
25	Row...	prod_2	GERMANY	2010-03-31	5	200	Y	Cust_4	Loyal Customer	40,000
26	Row...	prod_2	USA	2010-04-22	10	400	Y	Cust_3	Impulse Customer	160,000
27	Row...	prod_3	CHINA	2010-05-12	2	160	N	Cust_2	Loyal Customer	25,600
28	Row...	prod_1	USA	2010-05-17	5	175	Y	Cust_6	Loyal Customer	30,625
29	Row...	prod_2	GERMANY	2010-06-22	6	240	?	Cust_1	Loyal Customer	57,600
30	Row...	prod_1	CHINA	2010-06-28	10	350	Y	Cust_5	Impulse Customer	122,500
31	Row...	prod_2	USA	2010-07-07	12	480	?	Cust_3	Impulse Customer	230,400
32	Row...	prod_1	BRAZIL	2010-07-17	5	175	?	Cust_7	Loyal Customer	30,625
33	Row...	prod_1	CHINA	2010-08-28	10	350	N	Cust_2	Impulse Customer	122,500
34	Row...	prod_2	GERMANY	2010-08-31	5	200	?	Cust_1	Loyal Customer	40,000
35	Row...	prod_3	GERMANY	2010-09-14	2	160	?	Cust_1	Loyal Customer	25,600
36	Row...	prod_1	CHINA	2010-10-01	2	70	?	Cust_5	Loyal Customer	4,900
37	Row...	prod_1	USA	2010-10-11	2	70	Y	Cust_6	Loyal Customer	4,900
38	Row...	prod_2	USA	2010-12-07	15	600	N	Cust_6	Impulse Customer	360,000
39	Row...	prod_3	CHINA	2011-01-02	8	640	?	Cust_2	Loyal Customer	409,600
40	Row...	prod_1	USA	2011-01-10	10	350	Y	Cust_3	Impulse Customer	122,500
41	Row...	prod_2	GERMANY	2011-02-01	1	40	Y	Cust_1	Need based Customer	1,600
42	Row...	prod_3	BRAZIL	2011-02-02	8	640	Y	Cust_7	Loyal Customer	409,600
43	Row...	prod_2	GERMANY	2011-02-11	1	40	?	Cust_4	Need based Customer	1,600
44	Row...	prod_1	GERMANY	2011-03-06	10	350	?	Cust_4	Impulse Customer	122,500
45	Row...	prod_1	GERMANY	2011-03-18	1	35	Y	Cust_4	Need based Customer	1,225
46	Row...	prod_1	GERMANY	2011-03-20	11	385	N	Cust_4	Impulse Customer	148,225
47	Row...	prod_1	BRAZIL	2011-04-06	1	35	?	Cust_7	Need based Customer	1,225

Result:

The experiment was simulated on KNIME Analytics Platform and the outputs were verified successful.

Inference:

By trying out these methods, we can see how well String Manipulation, Math Formula, and Rule Engine nodes work in KNIME for changing data. Using these tools helps users get their data ready for analysis, making it better and easier to use for further study.

21BEC1851 – Rahul Karthik S

Flow:

Properties:
Slider Node:

Gauge Node:

MQTT Out Node:

Debug Node:

Output:

Inference:

Result:
Hence, the Data Manipulation in Knime like String Manipulation, Math Formula and Rule Engine is performed.

21BEC1851 – Rahul Karthik S

Name: Rahul Karthik S
Registration Number: 21BEC1851
Course: IoT Domain Analyst (BECE352E)
Slot: L51+L52

Experiment 7

K – Means Clustering using Knime Analytics Platform

Aim:

To Perform the K – Means Clustering in Knime Analytics Platform.

Software Required:

Knime Analytics Platform

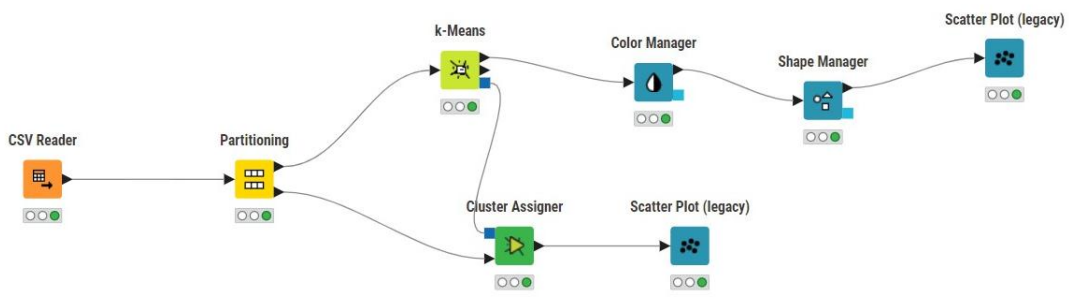
Theory:

- KNIME is a helpful tool for working with data. It has special parts called nodes that do different jobs.
- K Means Clustering is a widely-used machine learning method where data is split into K different groups, with no overlap.
- The algorithm repeatedly assigns each piece of data to the closest center point, then adjusts those points based on the average of the data in each group until it's done.
- The goal is to make the groups as similar as possible, minimizing the total differences within each group.

Procedure:

- Start by bringing your dataset into Knime, making sure it has the right features for clustering.
- Use the Partitioning node to split your dataset randomly into training and testing groups.
- Apply the K-Means node to the training set to do the clustering. Set it up with how many groups you want (K) and any other important settings.
- Use the Colour Manager and Shape Manager nodes to give each cluster a different color and shape. This makes it easier to see them on the scatter plot.
- Use the Cluster Assigner node to give the testing set cluster labels based on what we learned from the training set.
- Use the Scatter Plot node to make a picture of your data. Show the features on the axes and use the colors and shapes from before to show which group each point belongs to.

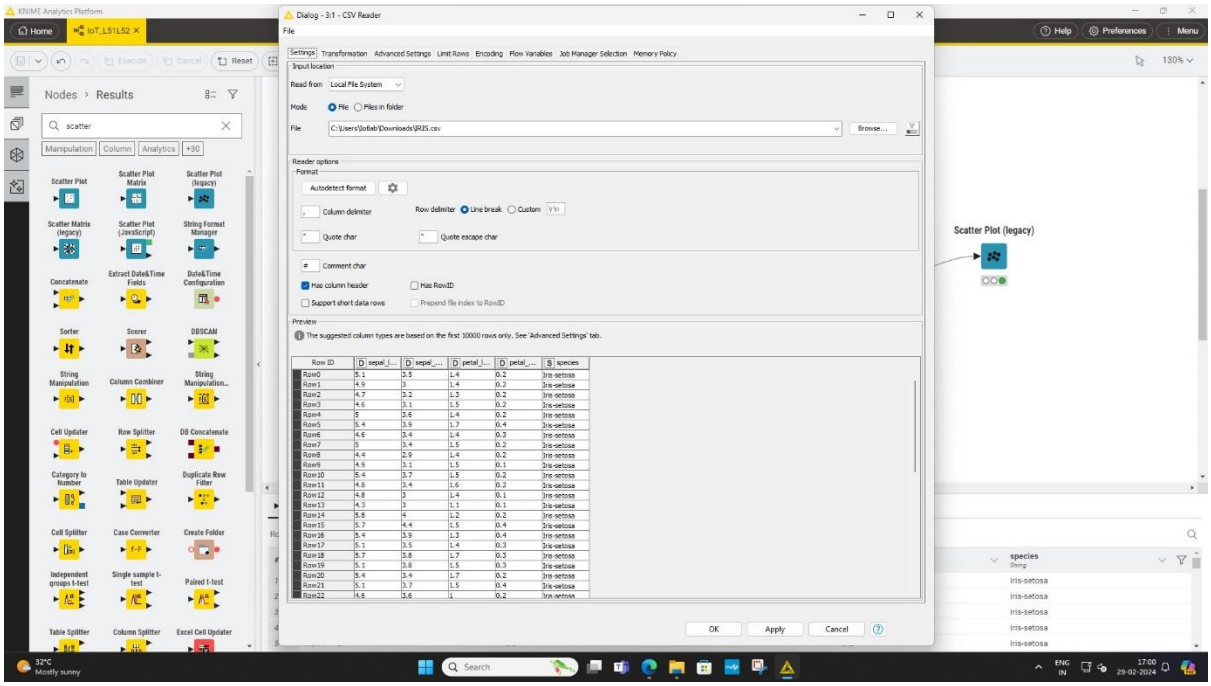
Flow:



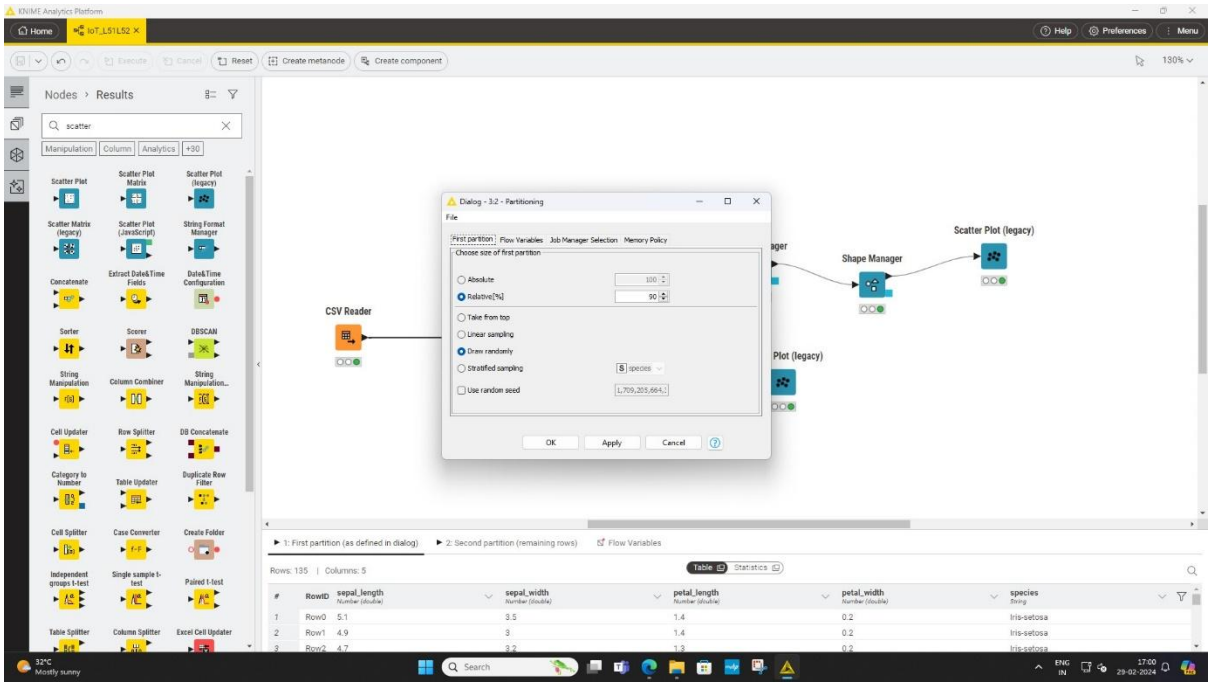
Properties:

21BEC1851 – Rahul Karthik S

CSV Reader:

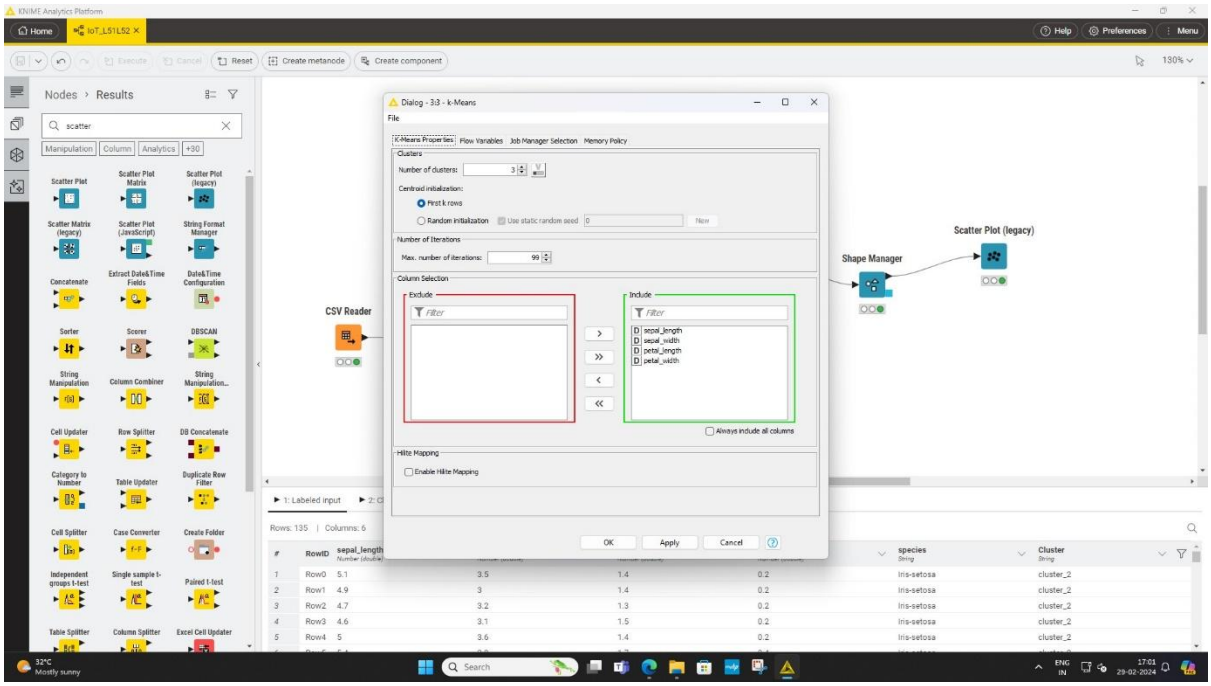


Partitioning:

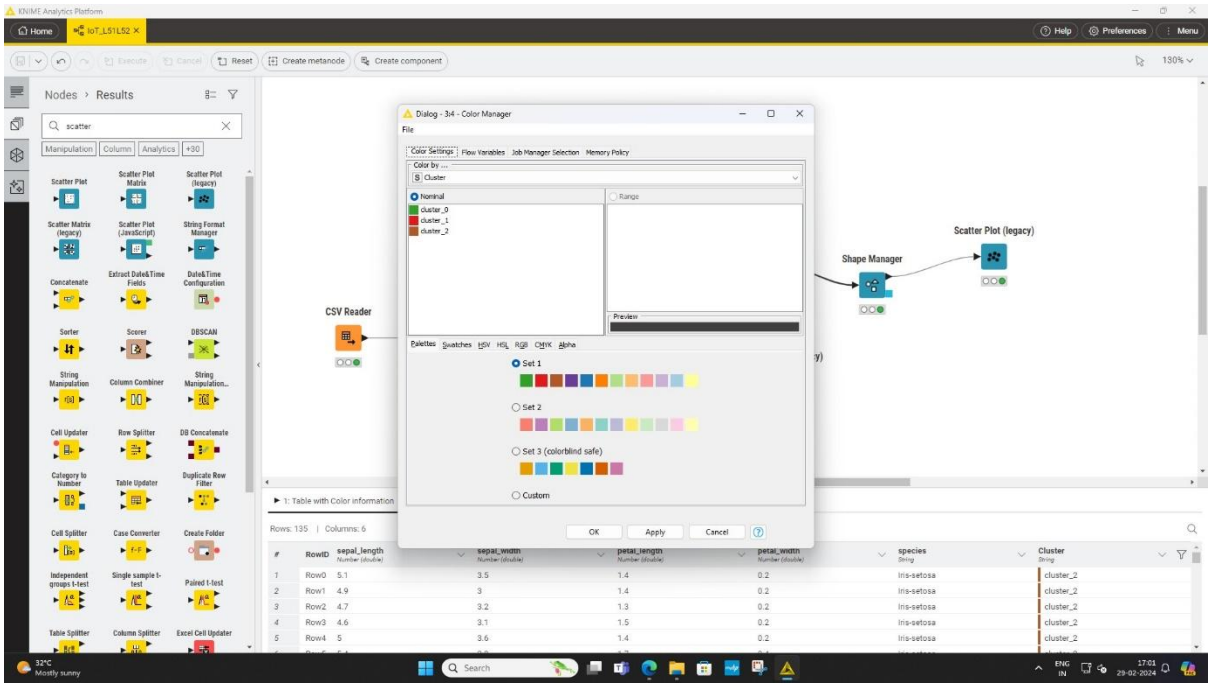


K-Means:

21BEC1851 – Rahul Karthik S

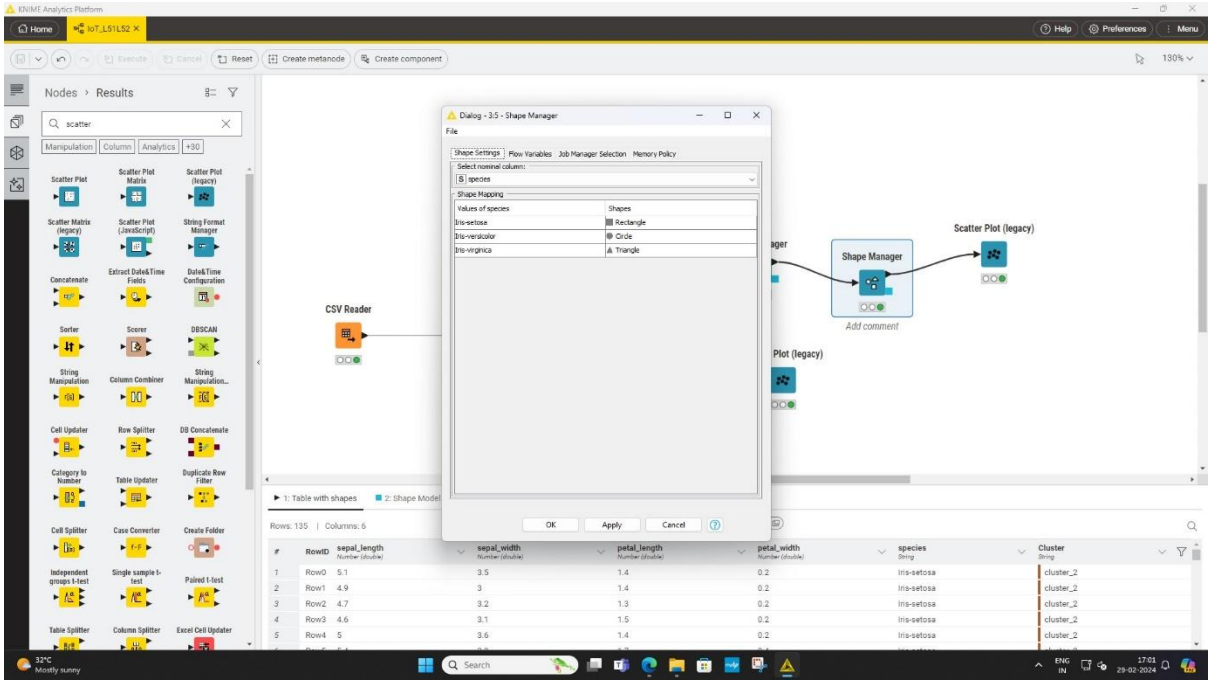


Color Manager:

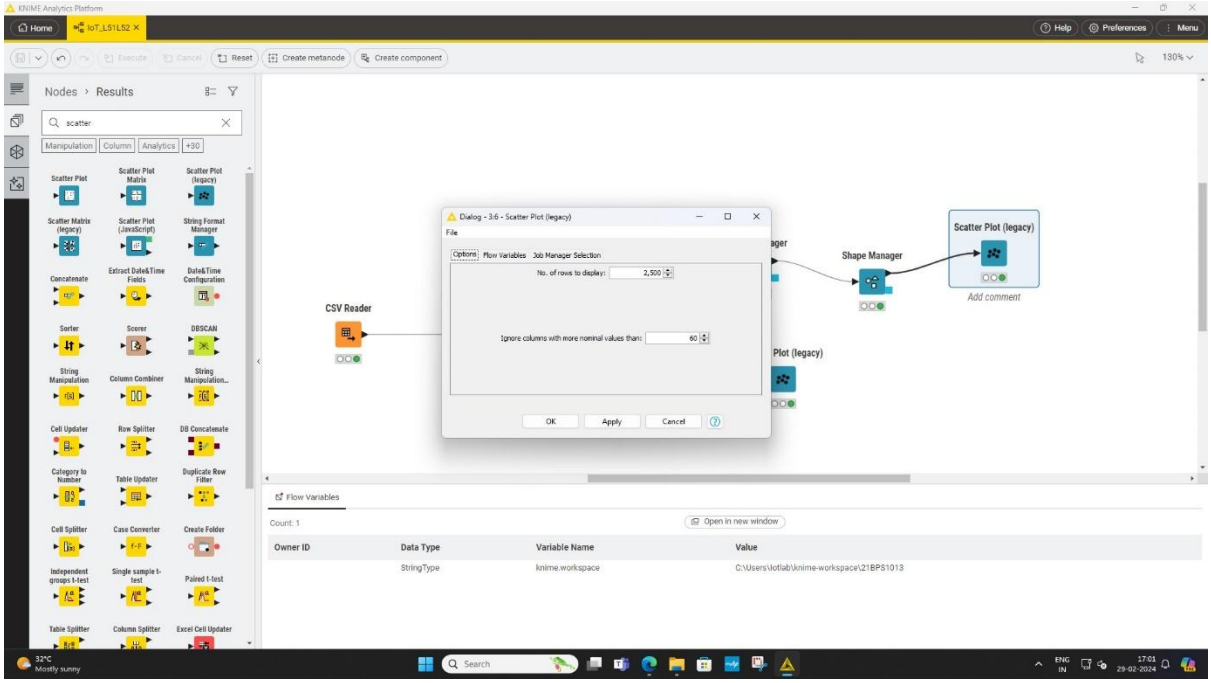


Shape Manager:

21BEC1851 – Rahul Karthik S

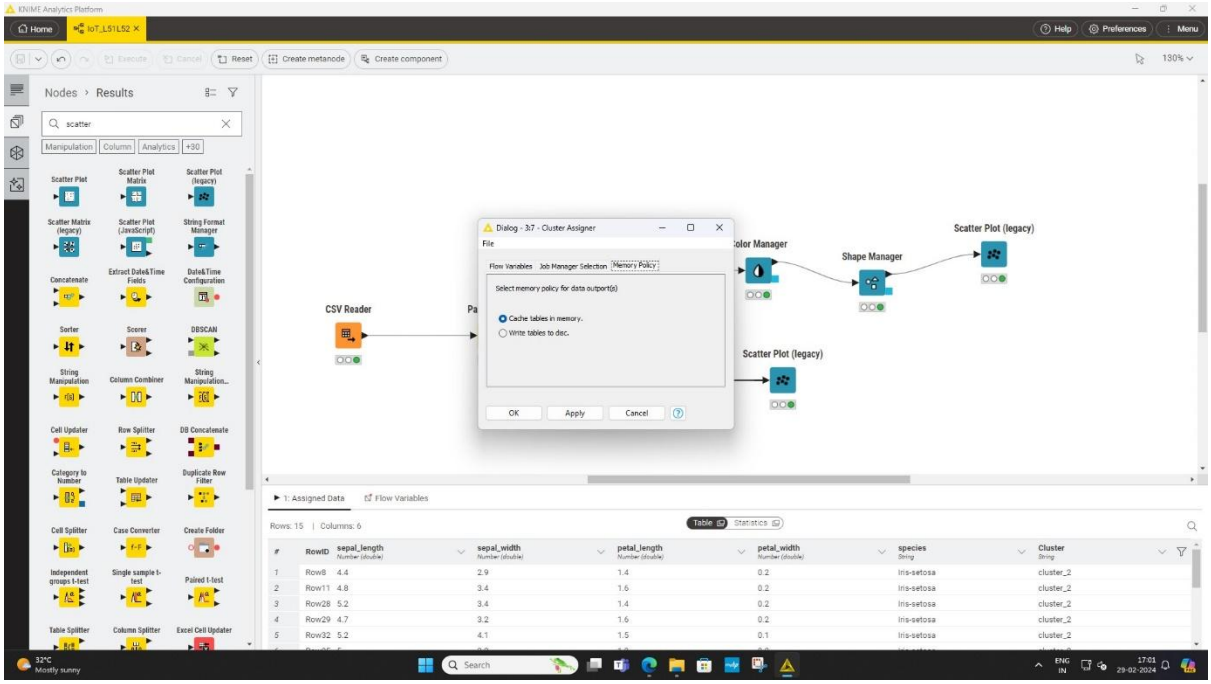


Scatter Plot (Legacy):

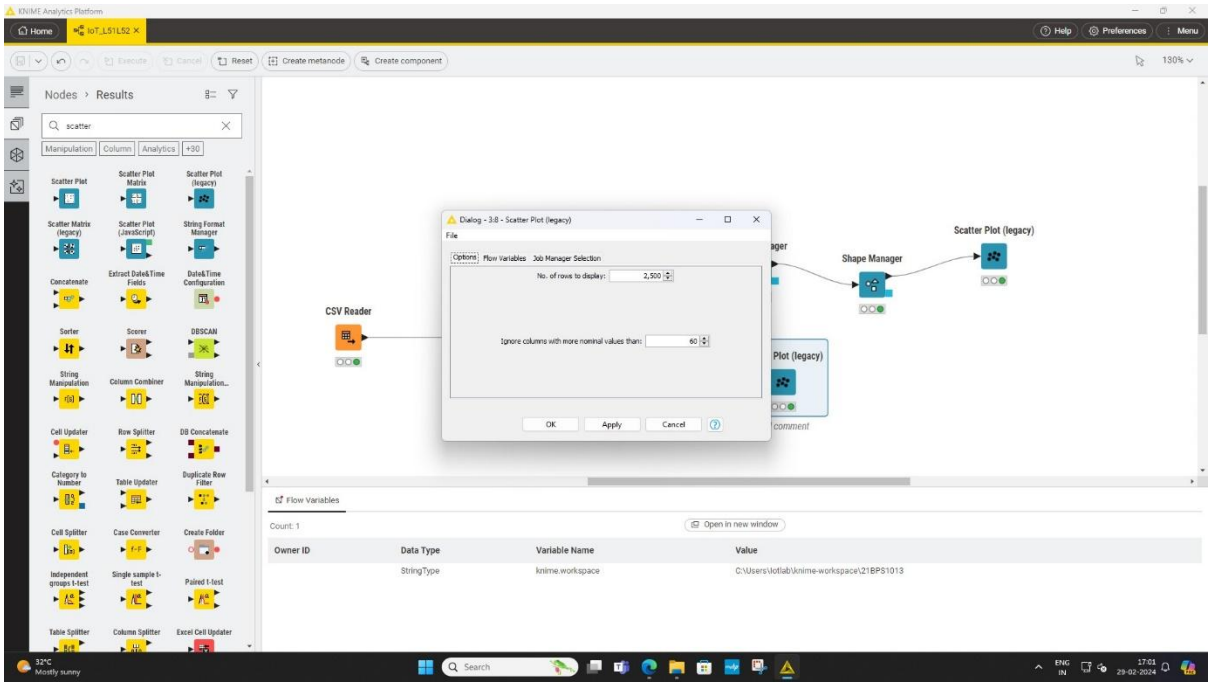


Cluster Assignment:

21BEC1851 – Rahul Karthik S



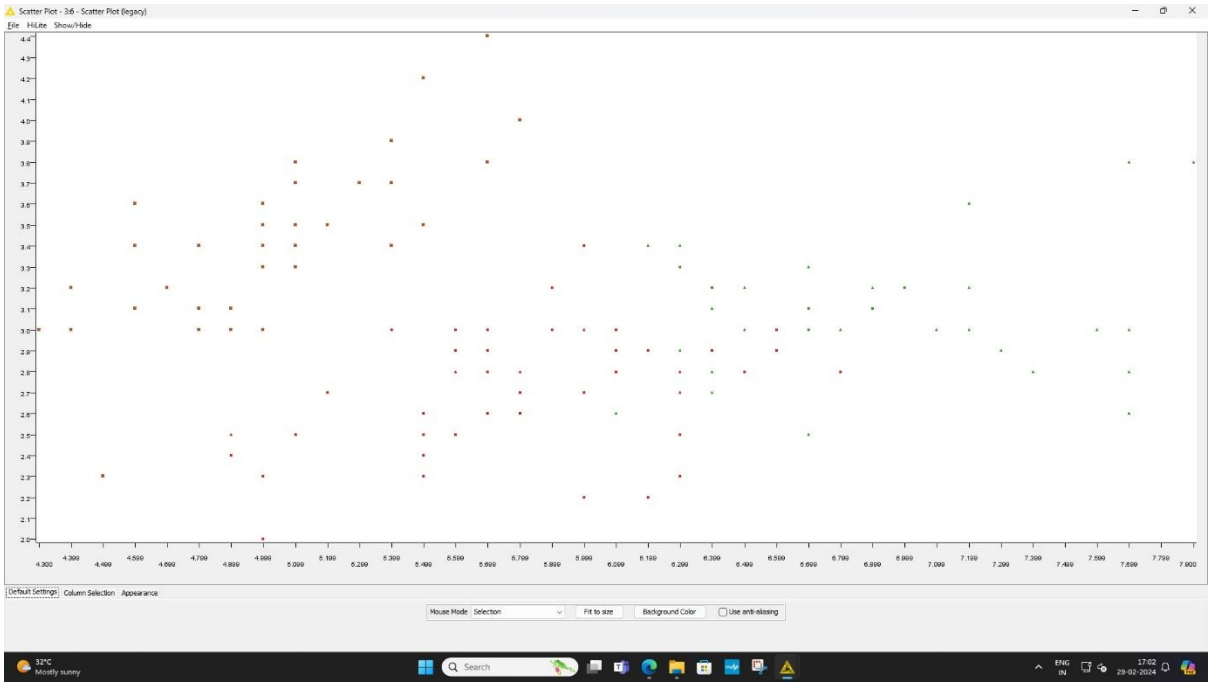
Scatter Plot (Legacy):



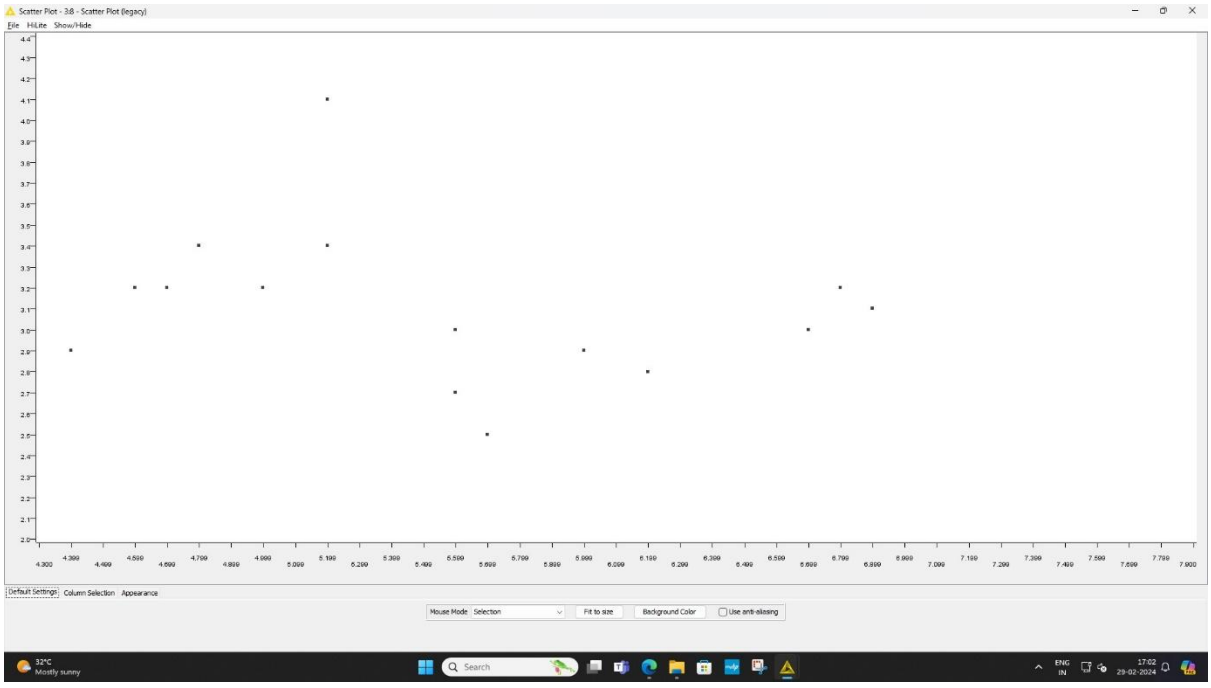
Output:

Scatter Plot 1:

21BEC1851 – Rahul Karthik S



Scatter Plot 2:



Result:

The experiment was simulated on KNIME Analytics Platform successfully and the outputs were verified.

Inference:

From this experiment, we can see how well K-Means Clustering works in Knime for grouping similar data together. The Scatter Plot node, along with colors and shapes assigned by the Colour Manager and Shape Manager nodes, helps us understand the clustering results better.

Name: Rahul Karthik S
Registration Number: 21BEC1851
Course: IoT Domain Analyst (BECE352E)
Slot: L51+L52

Experiment 8

Thingsboard – NodeRed Posting Data in Dashboard through HTTP

Aim:

To use ThingsBoard to post device data through HTTP.

Software Required:

NodeRed, ThingsBoard

Theory:

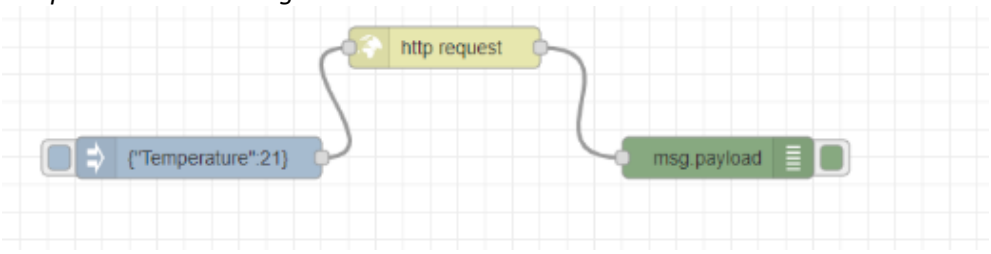
- An open-source IoT platform offering device management, data collection, processing, and visualisation for IoT solutions.
- It facilitates device connectivity through industry standards.
- IoT protocols such as MQTT, COAP, and HTTP support cloud and on-premises deployments.
- ThingsBoard ensures scalability, fault-tolerance, and performance, safeguarding against data loss.

Procedure:

- Open Node-Red in Command Prompt using node-red -v command.
- Configure the flow as per the properties on the Node Red.
- Now, open the Thingsboard website and sign in. Go to devices, create new device and paste the URL in the HTTP request node.

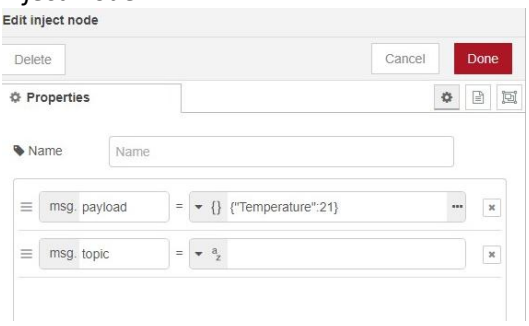
Flow:

Temperature Monitoring:



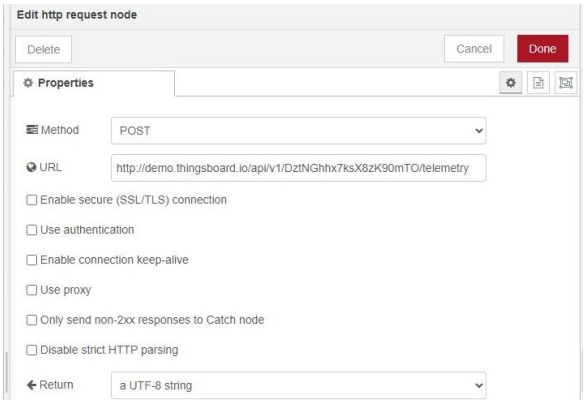
Properties:

Inject Node:



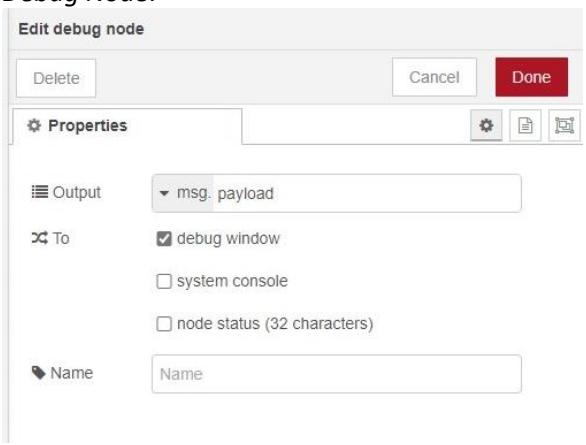
HTTP Request Node:

21BEC1851 – Rahul Karthik S



Key: DztNGhhx7ksX8zK90TO

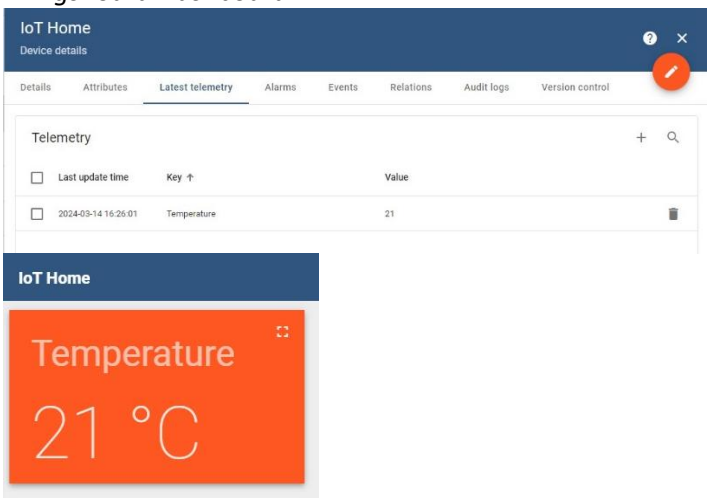
Debug Node:



Output:

Debug Window:

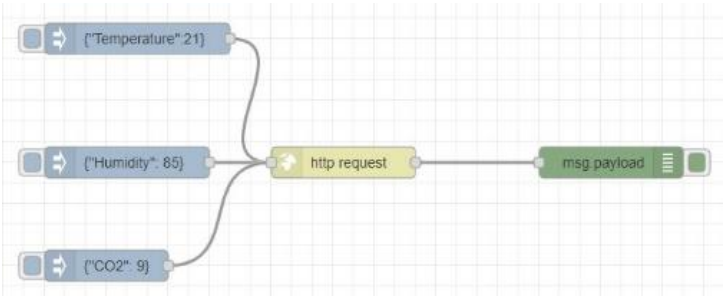
ThingsBoard Dashboard:



Task: Using node-red create a flow which performs automatic plant health monitoring. Using appropriate sensors simulate various parameters like air temperature, humidity, CO_2, etc. Display the visualization in ThingsBoard Dashboard Widgets.

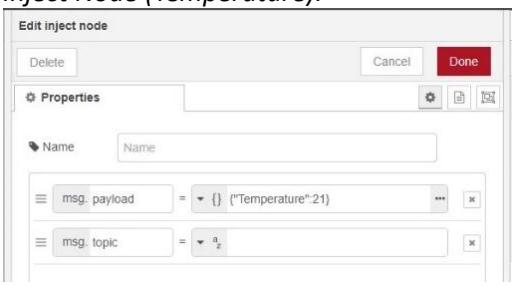
21BEC1851 – Rahul Karthik S

Flow:

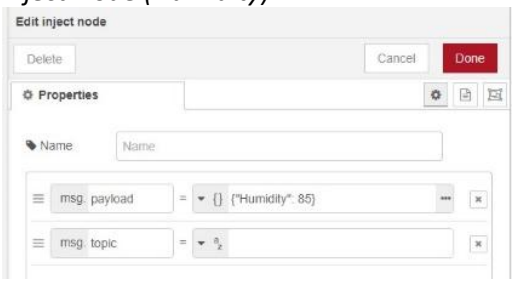


Properties:

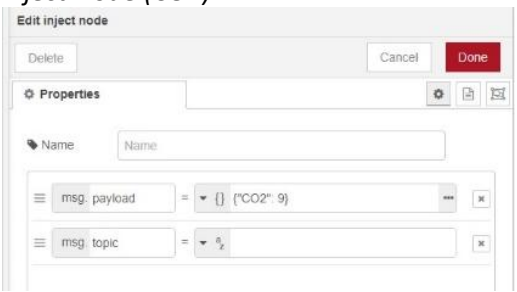
Inject Node (Temperature):



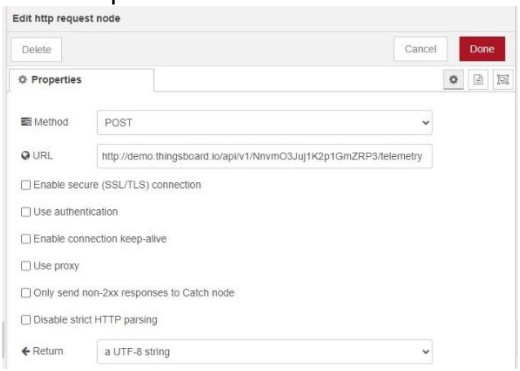
Inject Node (Humidity):



Inject Node (CO2):



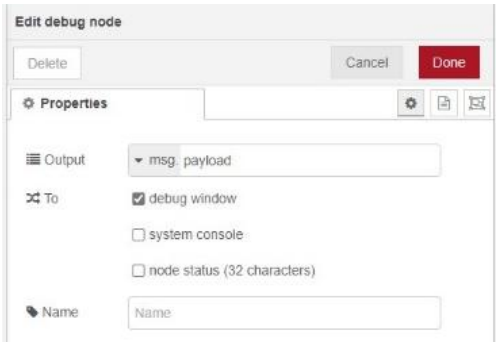
HTTP Request Node:



Key: NnvmO3Juj1K2p1GmZRP3

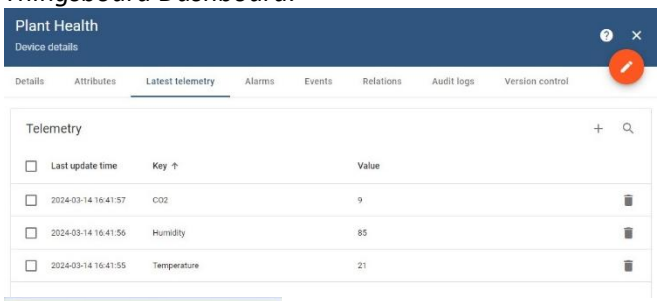
Debug Node:

21BEC1851 – Rahul Karthik S



Output:

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.

Result:

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

Name: Rahul Karthik S
Registration Number: 21BEC1851
Course: IoT Domain Analyst (BECE352E)
Slot: L51+L52

Experiment 9
Email Alert using Node-Red

Aim:
To send Email alerts using Node-Red.

Software Required:
Node-Red

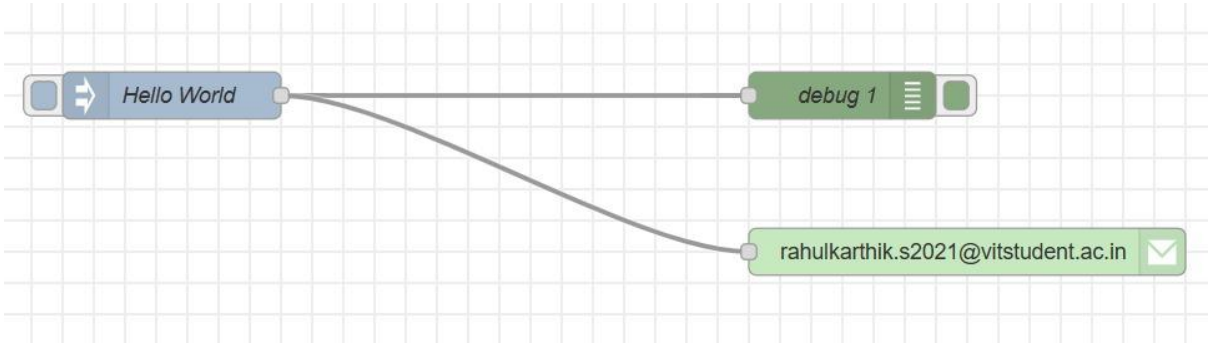
Theory:

- Setting up an email alert system using Node-RED involves leveraging event-driven and flow-based programming principles.
- Node-RED operates on event triggers and represents logic as interconnected nodes.
- Data from various sources is transformed, processed, and routed before triggering email alerts.
- Integration with external services like SMTP servers enables email notifications.
- Node-RED also offers tools for fault tolerance and monitoring to ensure reliability.

Procedure:

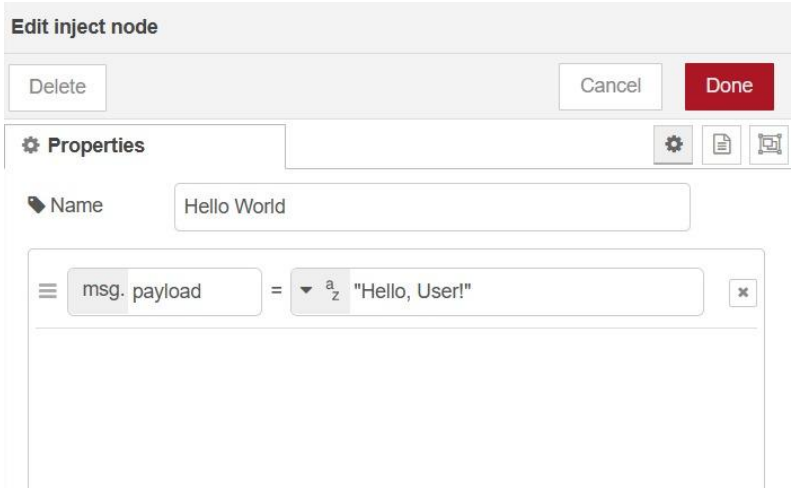
- Open Node-Red in Command Prompt using node-red -v command.
- To set up an email alert system using Node-RED, you'll first install and configure the email node with your SMTP server details.
- Configure the flow as per the properties on the Node Red.
- Click the Deploy button and inject from the inject node.
- Observe the Output from the debug window.

Flow:

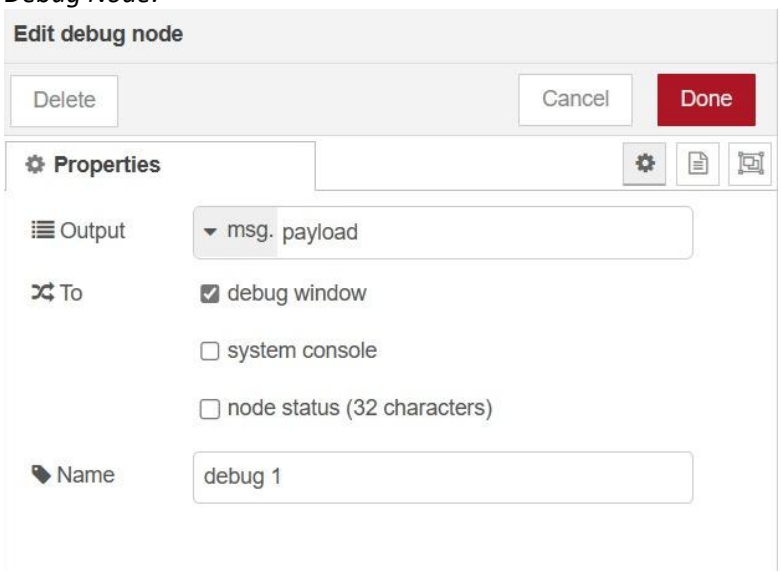


Properties:
Inject Node:

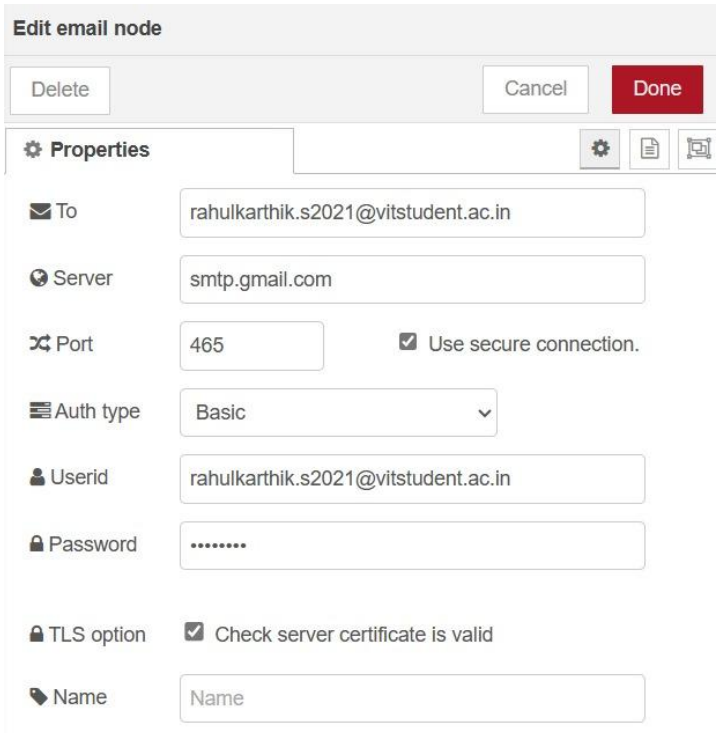
21BEC1851 – Rahul Karthik S



Debug Node:

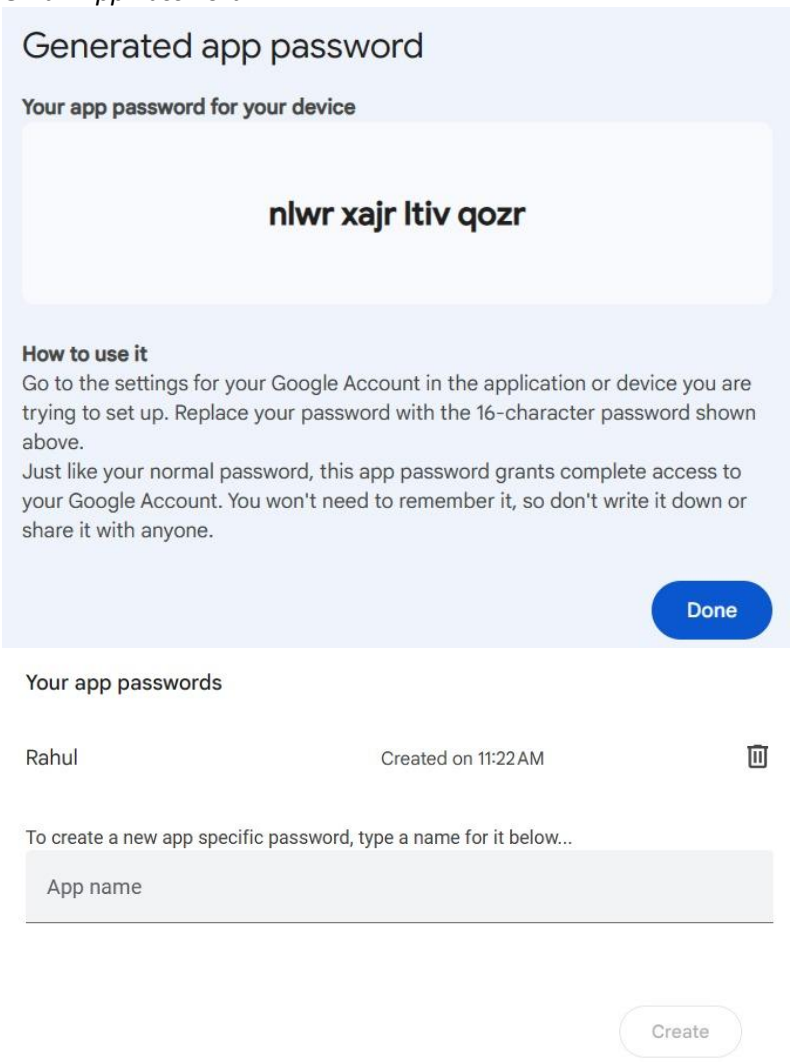


Email Node:



21BEC1851 – Rahul Karthik S

Gmail App Password:

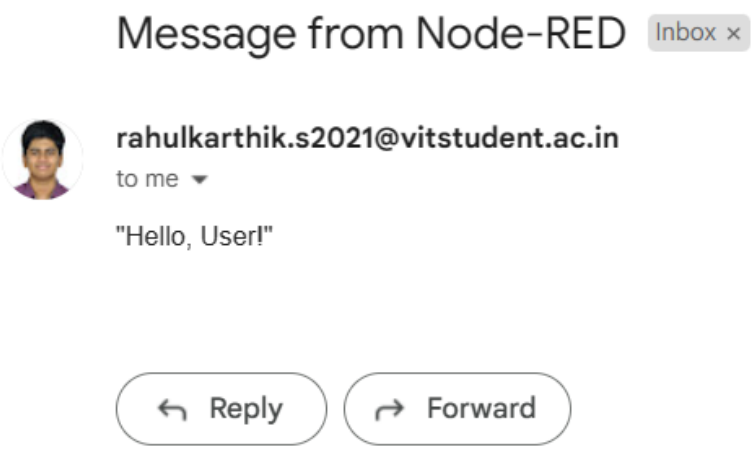


Output:

Debug Window:

Email Window:

21BEC1851 – Rahul Karthik S

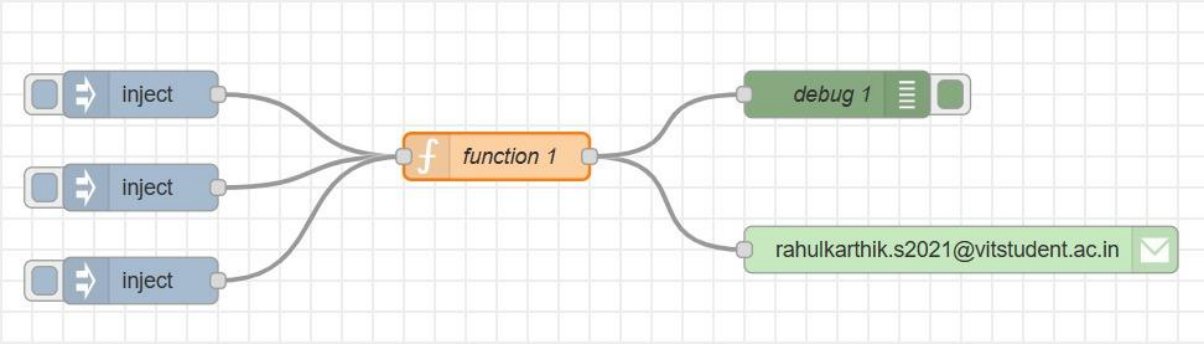


Task:

Drug consumption level notification through Email. The drug consumption is decided based on the patient's three parameters: HR, SpO2 and BP.

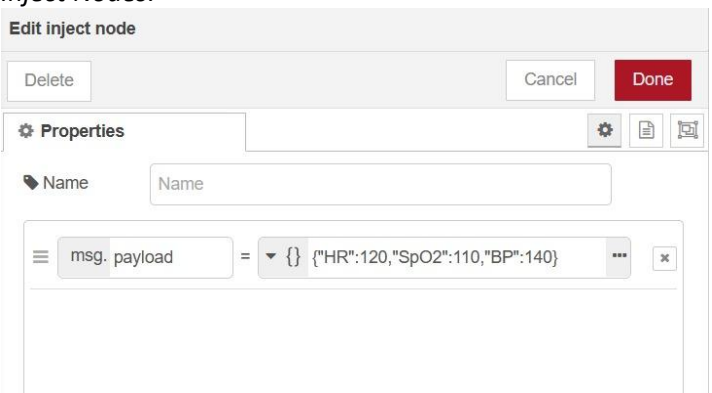
- If all the values are high alert the indication is "High".
- If all the values are Low alert the indication as "Low".
- In other cases it is "medium".

Flow:

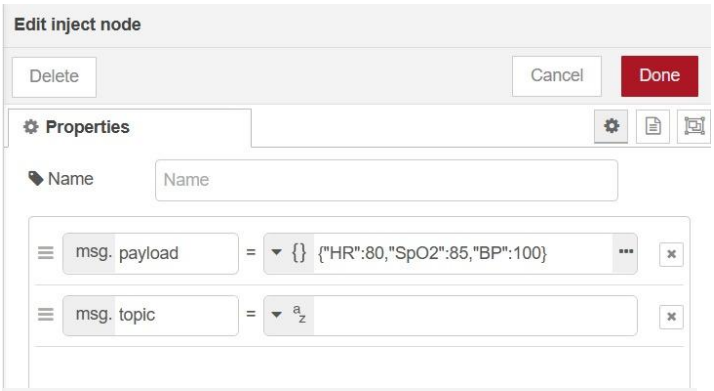


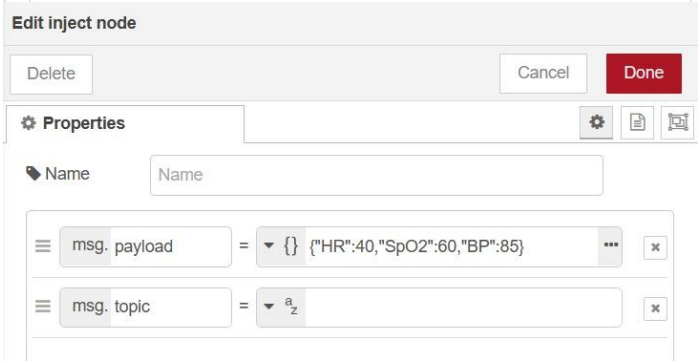
Properties:

Inject Nodes:

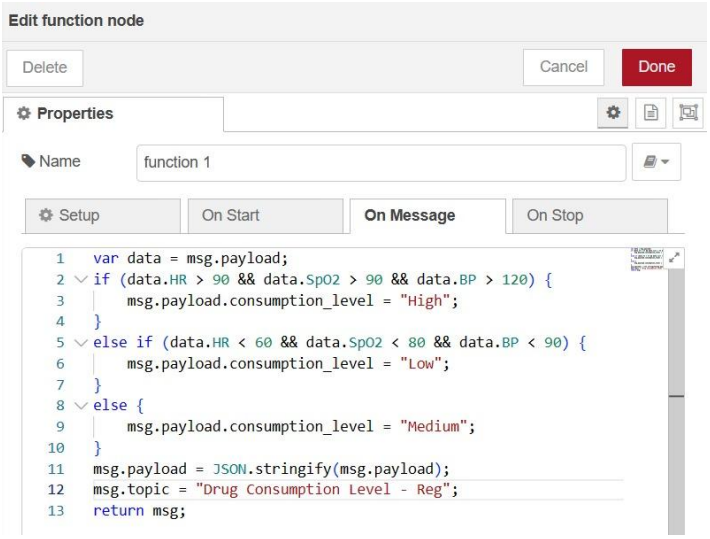


21BEC1851 – Rahul Karthik S

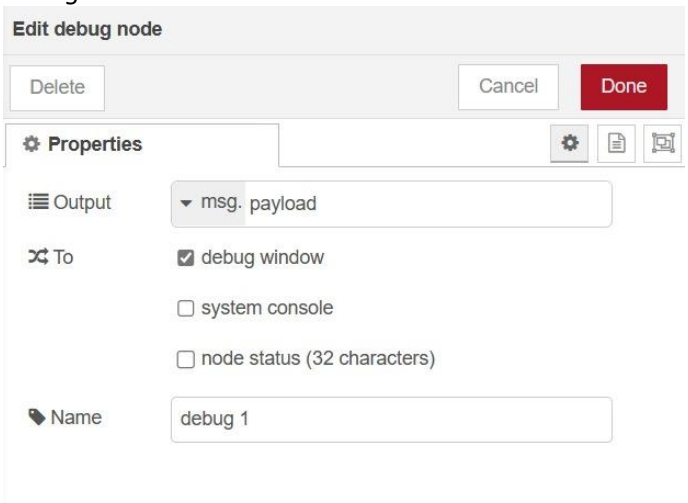




Function Node:

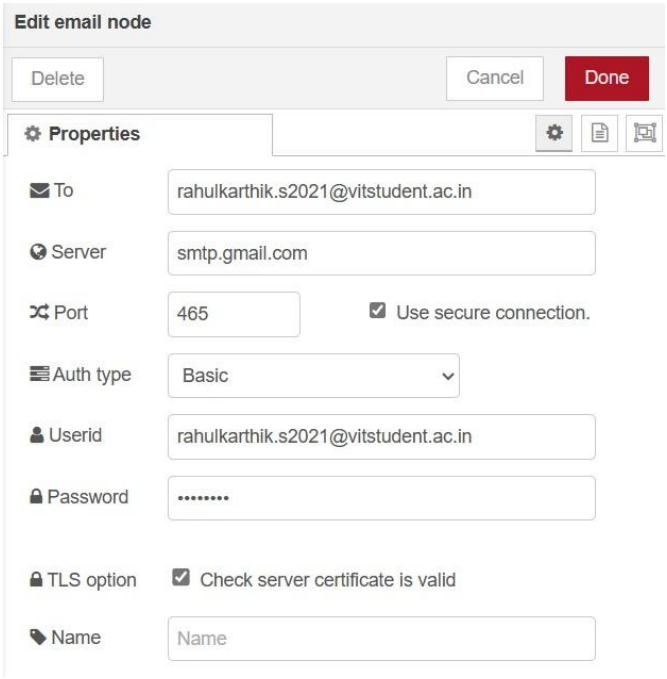


Debug Node:



21BEC1851 – Rahul Karthik S

Email Node:



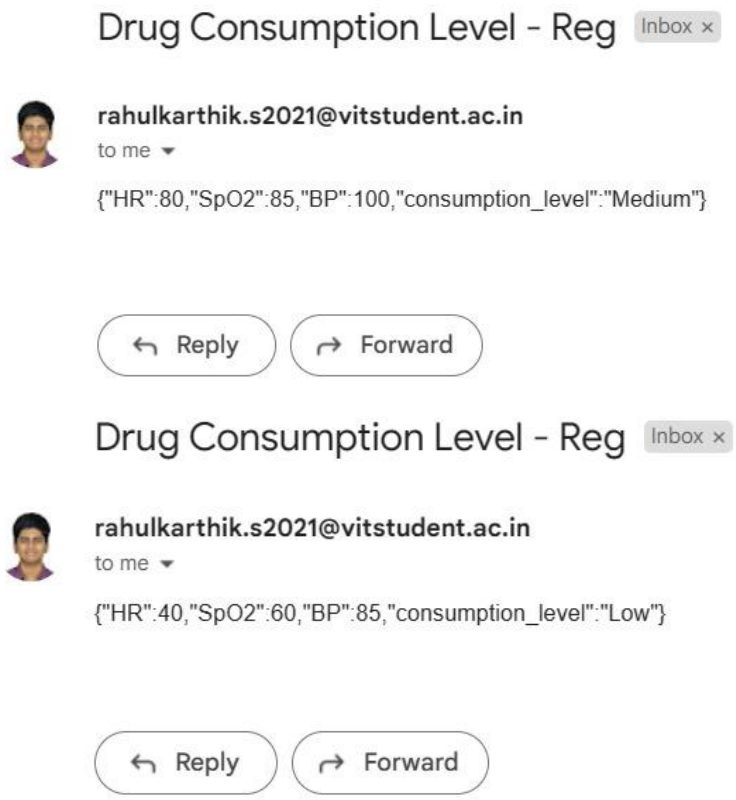
Code:

```
var data = msg.payload;
if (data.HR > 90 && data.SpO2 > 90 && data.BP > 120) {
    msg.payload.consumption_level = "High";
}
else if (data.HR < 60 && data.SpO2 < 80 && data.BP < 90) {
    msg.payload.consumption_level = "Low";
}
else {
    msg.payload.consumption_level = "Medium";
}
msg.payload = JSON.stringify(msg.payload);
msg.topic = "Drug Consumption Level - Reg";
return msg;
```

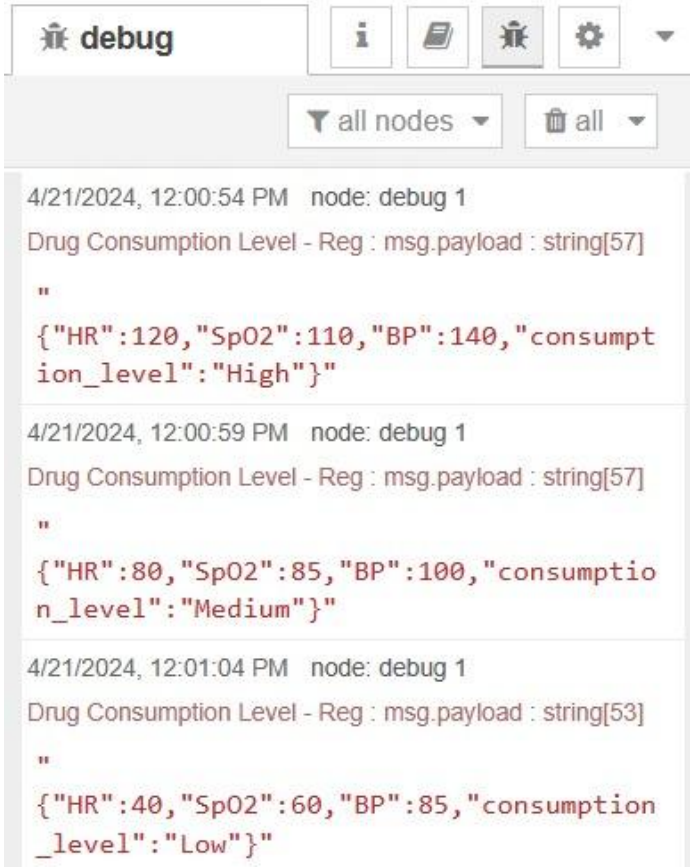
Output:

Email Window:

21BEC1851 – Rahul Karthik S



Debug Window:



Inference:

21BEC1851 – Rahul Karthik S

The experiment illustrates how Node-RED simplifies the creation of an email alert system through event-driven programming and visual flow-based design. By seamlessly integrating with external services like SMTP servers, Node-RED enables efficient communication and automation, underscoring its practicality for real-world applications requiring reliable event-based notifications.

Result:
Hence, using Node-Red the Email alerts are sent successfully.

21BEC1851 – Rahul Karthik S

Name: Rahul Karthik S
Registration Number: 21BEC1851
Course: IoT Domain Analyst (BECE352E)
Slot: L51+L52

Experiment 10

ESP8266 NodeMCU Web Server

Aim:

To create a web server with ESP8266 that controls LED using Arduino IDE.

Software Required:

Arduino IDE

Components Required:

- ESP8266 NodeMCU
- LEDs
- Resistors
- Jumper Wires
- Breadboard

Theory:

- The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability. It can connect to Wi-Fi networks and perform various tasks, making it an ideal choice for IoT projects.
- The ESP8266 can host a web server, allowing it to serve web pages to clients (such as web browsers or custom applications) over a Wi-Fi network. This capability is achieved through programming the ESP8266 to handle HTTP requests and responses.
- Hypertext Transfer Protocol (HTTP) is the standard protocol for communication on the World Wide Web. When a client sends an HTTP request to the ESP8266 web server, it can trigger actions such as turning an LED on or off based on the request parameters.
- The LED is connected to one of the GPIO pins of the ESP8266 microcontroller. By toggling the state of this GPIO pin, the LED can be turned on or off. Additionally, PWM (Pulse Width Modulation) can be used to control the brightness of the LED or to create color effects in RGB LEDs.
- The web server hosted on the ESP8266 provides a user interface (UI) through which users can interact with the LED. This UI can be a simple webpage with buttons for controlling the LED's state, or it can be more complex, allowing for features such as real-time updates or authentication.

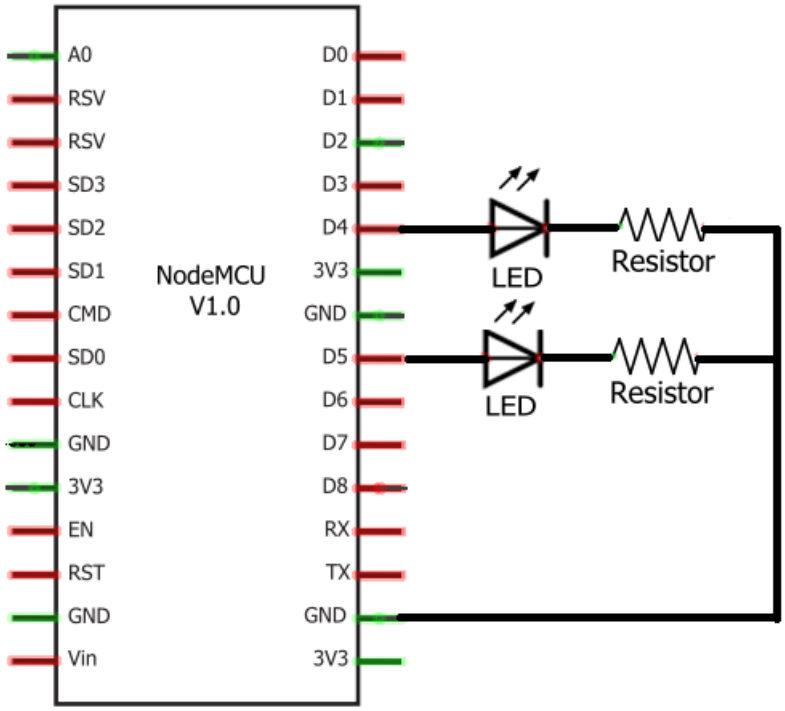
Procedure:

- Connect LED to GPIO pin of ESP8266.
- Write an Arduino sketch to configure ESP8266 for Wi-Fi and implement a web server.
- Define routes for LED control (e.g., "/on" to turn it on, "/off" to turn it off).
- Upload sketch to ESP8266.
- Test by accessing ESP8266's IP address in a web browser.

21BEC1851 – Rahul Karthik S

- Refine code as needed.
- Deploy ESP8266 for remote LED control.

Circuit Diagram:



Code:

```
#include <ESP8266WiFi.h>

const char* ssid     = "Rahul's Phone";

const char* password = "abcd@123";

WiFiServer server(80);

String header;

String output5State = "off";

String output4State = "off";

const int output5 = 5;

const int output4 = 4;

unsigned long currentTime = millis();

unsigned long previousTime = 0;

const long timeoutTime = 2000;

void setup() {

  Serial.begin(115200);
```

```
  pinMode(output5, OUTPUT);
  pinMode(output4, OUTPUT);
  digitalWrite(output5, LOW);
  digitalWrite(output4, LOW);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
void loop(){
  WiFiClient client = server.available();
  if (client) {
    Serial.println("New Client.");
    String currentLine = "";
    currentTime = millis();
    previousTime = currentTime;
    while (client.connected() && currentTime - previousTime <= timeoutTime) {
      currentTime = millis();
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        header += c;
        if (c == '\n') {
```

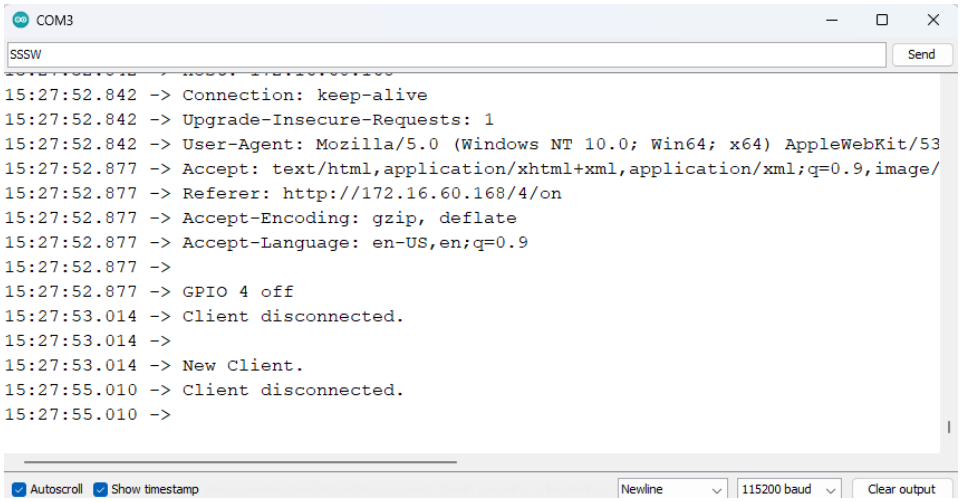
```
        if (currentLine.length() == 0) {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-type:text/html");
          client.println("Connection: close");
          client.println();
          if (header.indexOf("GET /5/on") >= 0) {
            Serial.println("GPIO 5 on");
            output5State = "on";
            digitalWrite(output5, HIGH);
          } else if (header.indexOf("GET /5/off") >= 0) {
            Serial.println("GPIO 5 off");
            output5State = "off";
            digitalWrite(output5, LOW);
        } else if (header.indexOf("GET /4/on") >= 0) {
          Serial.println("GPIO 4 on");
          output4State = "on";
           digitalWrite(output4, HIGH);
        } else if (header.indexOf("GET /4/off") >= 0) {
          Serial.println("GPIO 4 off");
          output4State = "off";
          digitalWrite(output4, LOW);
          }
          client.println("<!DOCTYPE html><html>");
          client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-
scale=1\">");
          client.println("<link rel=\"icon\" href=\"data:,\">");
          client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-
align: center;}");
          client.println(".button { background-color: #195B6A; border: none; color: white; padding: 16px
40px;");
          client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
          client.println(".button2 {background-color: #77878A;}</style></head>");
```

```
            client.println("<body><h1>ESP8266 Web Server</h1>");
            client.println("<p>GPIO 5 - State " + output5State + "</p>");
            // If the output5State is off, it displays the ON button
            if (output5State=="off") {
              client.println("<p><a href=\"/5/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/5/off\"><button class=\"button
button2\">OFF</button></a></p>");
            }
           client.println("<p>GPIO 4 - State " + output4State + "</p>");
            if (output4State=="off") {
              client.println("<p><a href=\"/4/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"a/4/off\"><button class=\"button
button2\">OFF</button></a></p>");
            }
            client.println("</body></html>");
            client.println();
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }
      }
    }
    header = "";
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
```

}

Output:

Arduino Serial Monitor:



Web Server:

Inference:

The ESP8266 web server for controlling an LED enables remote access and manipulation of the LED's state via a web interface, making it a versatile solution for IoT applications and home automation projects.

Result:

Hence, a web server with ESP8266 to control LED is created successfully.

Name: Rahul Karthik S
Registration Number: 21BEC1851
Course: IoT Domain Analyst (BECE352E)
Slot: L51+L52

Experiment 11

Monitoring Temperature and Humidity with DHT11 Sensor using Arduino and Node-Red

Aim:

To monitor temperature and humidity with a DHT11 sensor using Arduino and Node-Red.

Software Required:

NodeRed, Arduino IDE

Components Required:

1. Arduino Uno R3 – 1
2. DHT11 Sensor – 1
3. Connecting Wires – 3

Theory:

- *DHT11 Sensor:* A digital temperature and humidity sensor that measures environmental conditions.
- *Arduino:* Acts as the interface between the DHT11 sensor and Node-RED, reading data from the sensor and sending it to Node-RED.
- *Node-RED:* A visual programming tool used to process data received from Arduino, facilitating visualization and control.
- *Data Acquisition:* Arduino reads temperature and humidity values from the DHT11 sensor and sends them to Node-RED via serial communication.
- *Communication:* Arduino communicates with Node-RED using a serial protocol, sending the sensor data over the serial port.
- *Node-RED Flow:* A flow is created in Node-RED to receive, process, and visualize the data received from Arduino.
- *Visualization:* Node-RED provides tools for creating real-time visualizations of the temperature and humidity data.
- *User Interface:* A user interface is created to display the temperature and humidity readings, allowing users to monitor environmental conditions in real time.

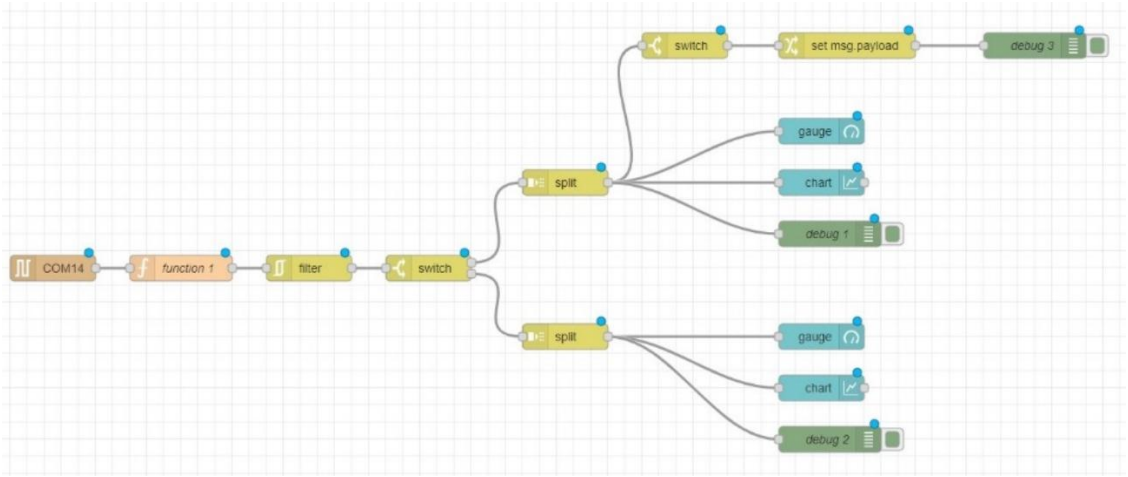
Procedure:

- Open Node-Red in Command Prompt using node-red -v command.
- Configure the flow as per the properties on the Node Red.
- Now, connect the Arduino Uno R3 board to the computer and then upload the code to the Arduino.
- Then connect the Temperature and Humidity sensors and measure the values from the Serial Monitor and Node-RED.

Flow:

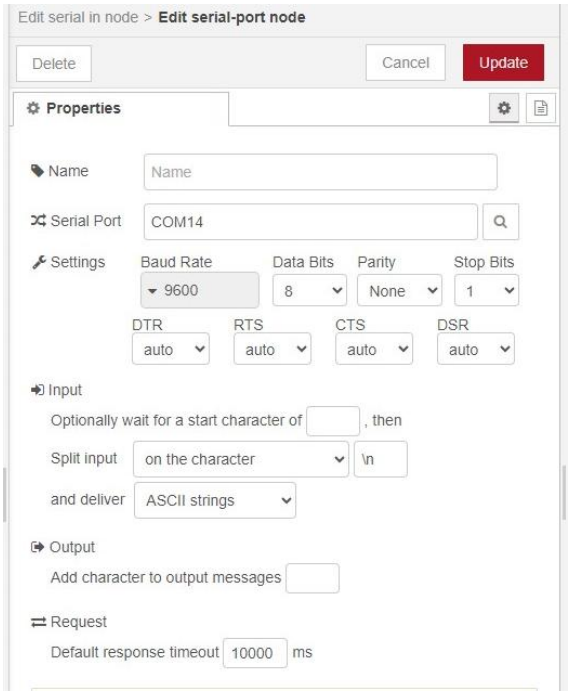
Temperature Monitoring:

21BEC1851 – Rahul Karthik S

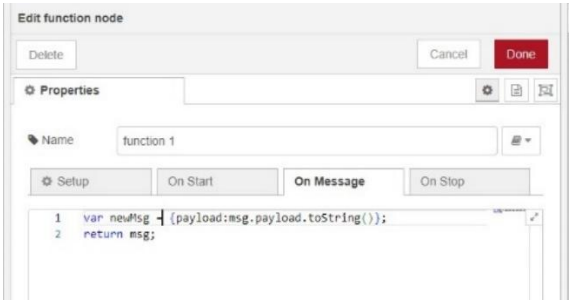


Properties:

Serial-Port Node:

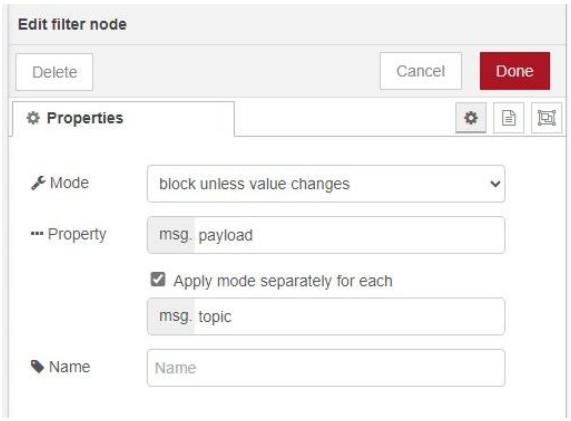


Function Node:

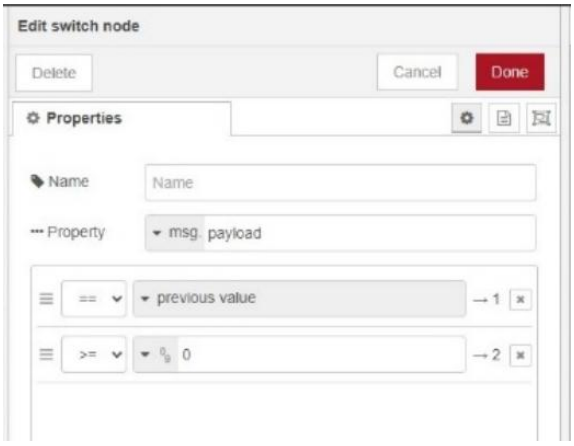


Filter Node:

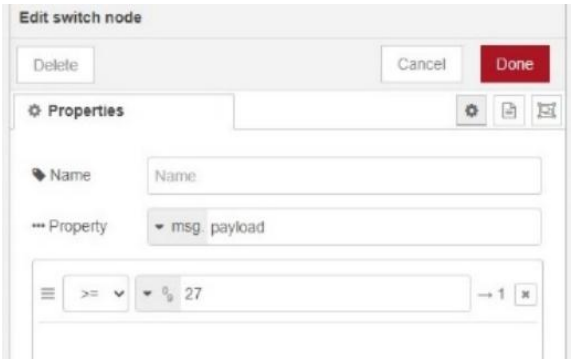
21BEC1851 – Rahul Karthik S



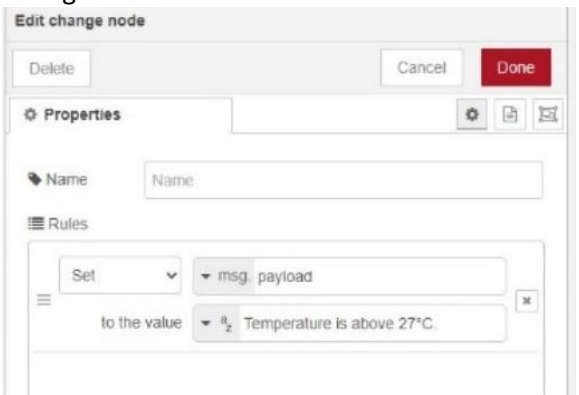
Switch Node:



Switch Node – 2:

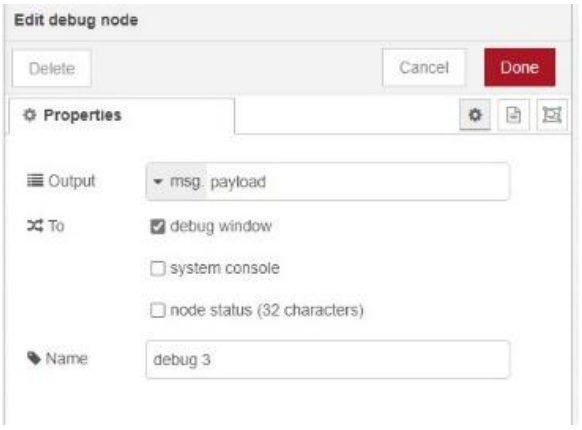


Change Node:



Debug Node:

21BEC1851 – Rahul Karthik S



Gauge Node:

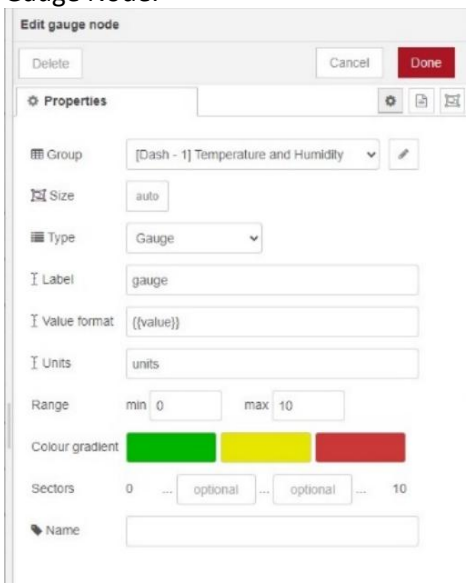
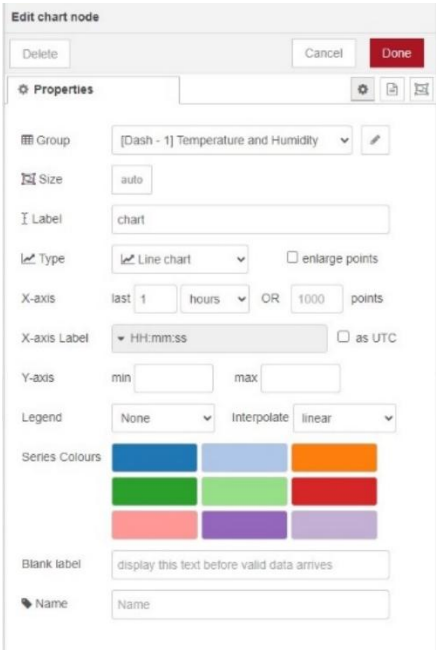
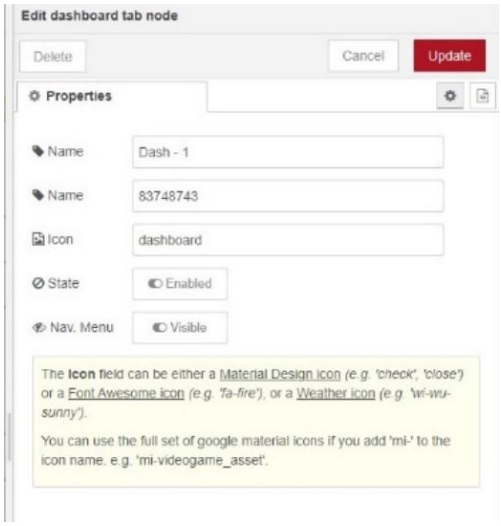


Chart Node:

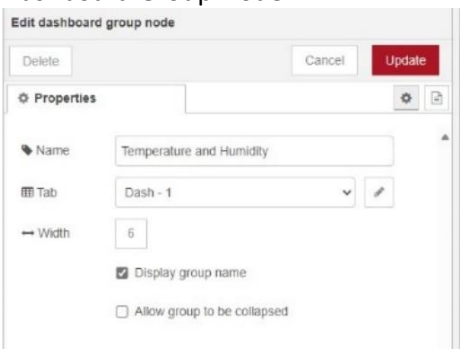


Dashboard Tab Node:

21BEC1851 – Rahul Karthik S



Dashboard Group Node:



Arduino Code:

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN 2     // Digital pin connected to the DHT sensor
// Uncomment the type of sensor in use:
#define DHTTYPE    DHT11     // DHT 11

DHT_Unified dht(DHTPIN, DHTTYPE);

uint32_t delayMS;

void setup() {
  Serial.begin(9600);
  // Initialize device.
  dht.begin();
  Serial.println(F("DHTxx Unified Sensor Example"));
  // Print temperature sensor details.
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  Serial.println(F("------------------------------------"));
  Serial.println(F("Temperature Sensor"));
  Serial.print  (F("Sensor Type: ")); Serial.println(sensor.name);
  Serial.print  (F("Driver Ver:  "));
Serial.println(sensor.version);
```

```
  Serial.print  (F("Unique ID:   "));
Serial.println(sensor.sensor_id);
  Serial.print  (F("Max Value:   "));
Serial.print(sensor.max_value); Serial.println(F("°C"));
  Serial.print  (F("Min Value:   "));
Serial.print(sensor.min_value); Serial.println(F("°C"));
  Serial.print  (F("Resolution:  "));
Serial.print(sensor.resolution); Serial.println(F("°C"));
  Serial.println(F("------------------------------------"));
  // Print humidity sensor details.
  dht.humidity().getSensor(&sensor);
  Serial.println(F("Humidity Sensor"));
  Serial.print  (F("Sensor Type: ")); Serial.println(sensor.name);
  Serial.print  (F("Driver Ver:  "));
Serial.println(sensor.version);
  Serial.print  (F("Unique ID:   "));
Serial.println(sensor.sensor_id);
  Serial.print  (F("Max Value:   "));
Serial.print(sensor.max_value); Serial.println(F("%"));
  Serial.print  (F("Min Value:   "));
Serial.print(sensor.min_value); Serial.println(F("%"));
  Serial.print  (F("Resolution:  "));
Serial.print(sensor.resolution); Serial.println(F("%"));
  Serial.println(F("------------------------------------"));
  // Set delay between sensor readings based on sensor details.
  delayMS = sensor.min_delay / 1000;
}

void loop() {
  // Delay between measurements.
  delay(delayMS);
  // Get temperature event and print its value.
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (isnan(event.temperature)) {
    Serial.println(F("Error reading temperature!"));
  }
  else {
    Serial.print(F("Temperature: "));
    Serial.print(event.temperature);
    Serial.println(F("°C"));
  }
  // Get humidity event and print its value.
  dht.humidity().getEvent(&event);
  if (isnan(event.relative_humidity)) {
    Serial.println(F("Error reading humidity!"));
  }
  else {
    Serial.print(F("Humidity: "));
    Serial.print(event.relative_humidity);
    Serial.println(F("%"));
  }
}
```

Output:

21BEC1851 – Rahul Karthik S

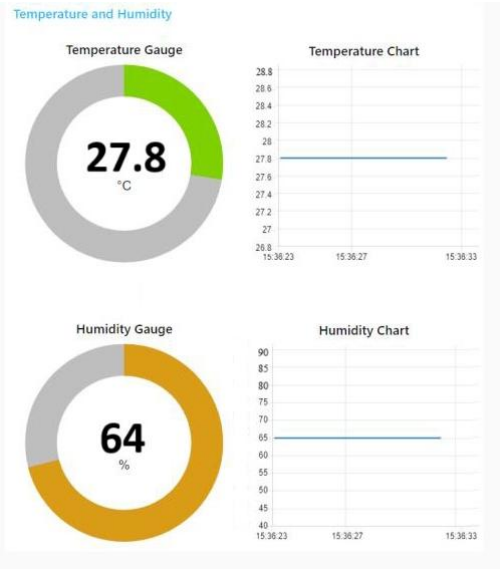
Arduino Serial Monitor:

```
Temperature Sensor
Sensor Type: DHT11
Driver Ver:  1
Unique ID:   -1
Max Value:   50.00°C
Min Value:   0.00°C
Resolution:  2.00°C
------------------------------------
Humidity Sensor
Sensor Type: DHT11
Driver Ver:  1
Unique ID:   -1
Max Value:   80.00%
Min Value:   20.00%
Resolution:  5.00%
------------------------------------
Temperature: 30.90°C
Humidity: 64.00%
Temperature: 30.90°C
Humidity: 64.00%
Temperature: 30.80°C
Humidity: 64.00%
Temperature: 30.80°C
Humidity: 64.00%
Temperature: 30.70°C
Humidity: 64.00%
```

Debug Window:

Node-Red Visualization:

21BEC1851 – Rahul Karthik S



Inference:

The experiment demonstrates the integration of a DHT11 sensor with Arduino and Node-RED for real-time monitoring of temperature and humidity. This setup enables efficient data acquisition, processing, visualization, and user interface creation, offering a practical solution for environmental monitoring applications.

Result:

Thus, we have successfully designed a temperature and humidity Monitoring System using DHT11 sensor, Arduino and Node-RED.