

Node-RED: Case Study 1

Basic IoT Based Smart Agriculture with Remote
Monitoring System using Arduino Uno & Soil
Moisture Sensor

v1 mar2021 : ver1(2) mar2021

iezan74@gmail.com



Scenario: Develop a real time basic IoT system that will plot the soil moisture value on a dashboard (the condition are standard: higher value represent higher moisture level due to less resistance).

- when the soil was dry (~850)
- when the soil was completely wet (~400)



Status: Dry
Test Reading: ~850



Status: Completely wet
Test Reading: ~400

v1(2)-mar-21

<https://lastminuteengineers.com/soil-moisture-sensor-arduino-tutorial/>

Requirement:

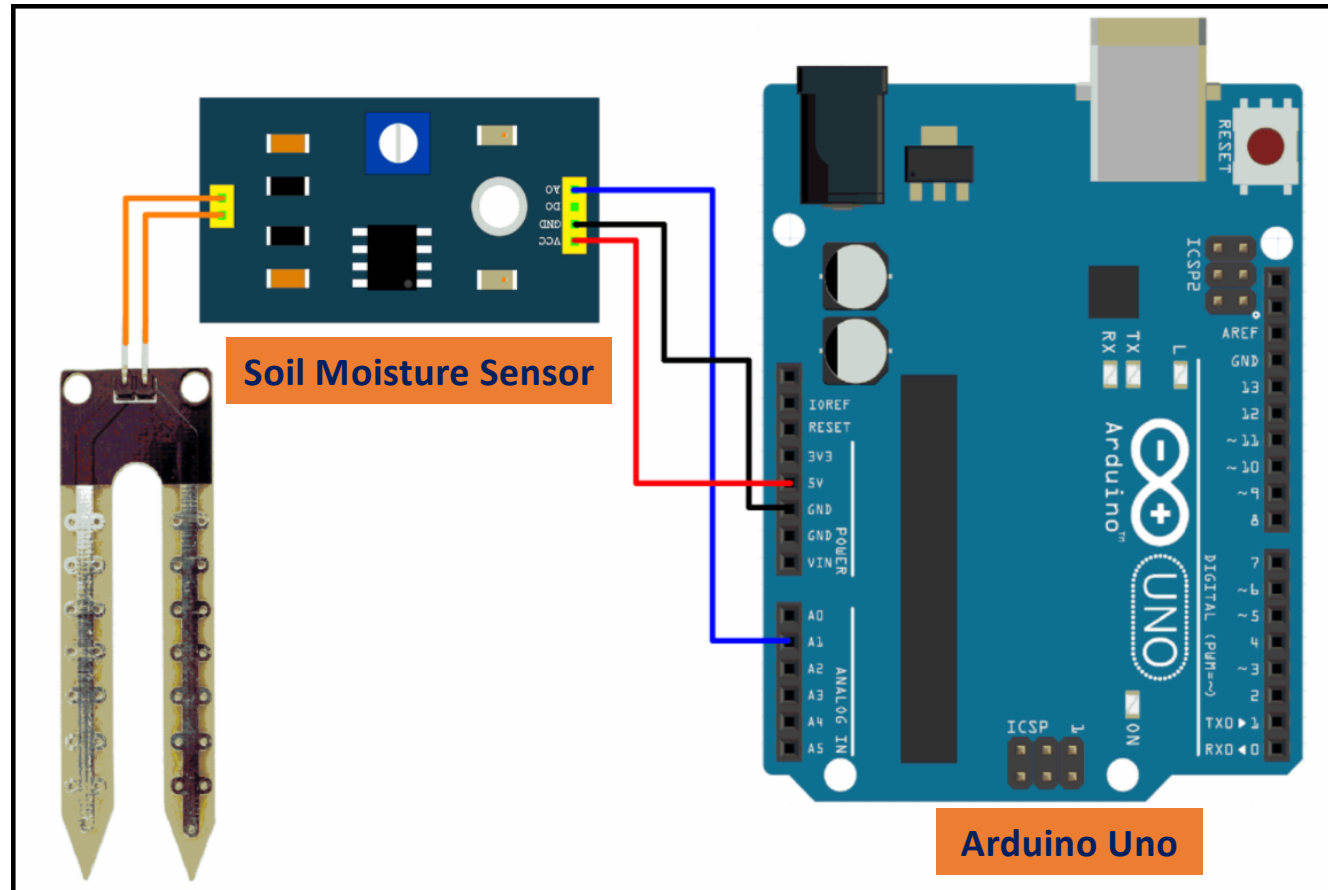
- i. Microcontroller x 1 (Uno / Mega / NodeMCU / ESP32 / Nano...
- ii. Soil Moisture Sensor x1
- iii. NodeRED – PC or Pi
- iv. Sketch: **nodeRed-02mac21-argiculture-v1.ino**

Methods:

- i. Do wiring connection & upload the sketch into microcontroller.
Troubleshoot any errors.
- ii. NodeRED configuration: Layout, Nodes & nodes properties
- iii. Test the system

Microcontroller: a. The Schematic Diagram.

- >Connect your board to PC / laptop.
- >Make sure correct board name & port is selected.
- >Always check your wiring especially the power supply. This might save your money from replacing a burnt device.



Microcontroller: b. The sketch.

```
1 //IoT Based Smart Agriculture with Remote Monitoring System
2 //v1-mac2021
3 void setup() {
4   // initialize serial communication at 9600 bits per second:
5   Serial.begin(9600);
6 }
7
8 void loop() {
9   String moist; //set moist as string
10  int sensorValue = analogRead(A0); //read incoming value from analog pin 1 & put at variable named sensorValue
11  moist = String(sensorValue); //convert sensorValue from int to string -> only to display at nodeRED dashboard
12
13  Serial.print("Moisture Level: "); //remark this line when displaying the value at nodeRED dashboard
14  Serial.print(moist); //will print at Serial Monitor & nodeRED: debug node & serial in node
15  Serial.println(","); //delimiter -> for nodeRED -> to differentiate new data
16  delay(1000); //pause for 1 sec
17 }
```

source: nodeRed-02mac21-argiculture-v1.ino

Note:

Put a remark at line number 9 when using nodeRED dashboard. The chart node cannot read the serial value. Refer to next slide.

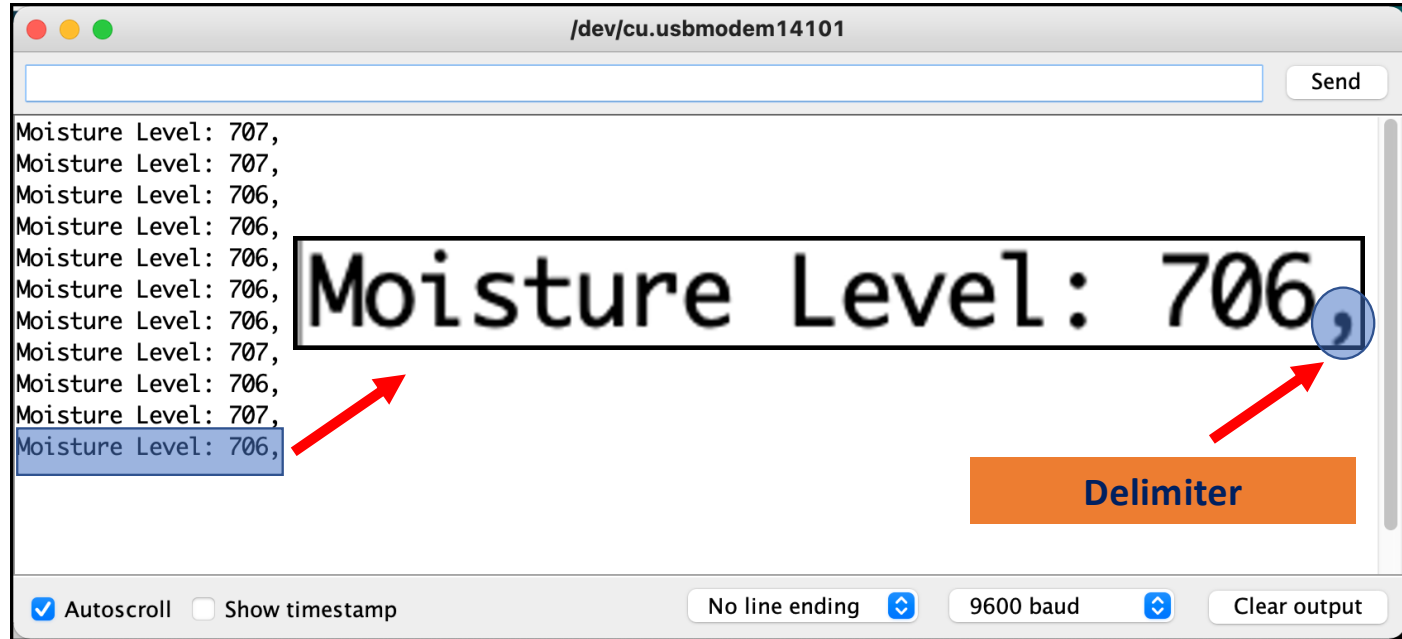
Microcontroller: c. Expected Output.

>Upload the sketch.

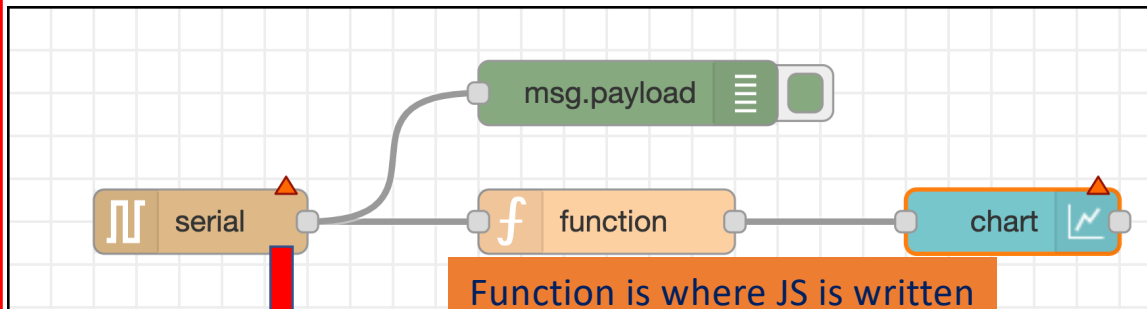
>**Error are expected** when you copy the sketch from previous page. Check the **double quotes** symbols. (“ ”). Delete and replace new **double quotes**.

(Always happens when
CnP text form net or pdf)

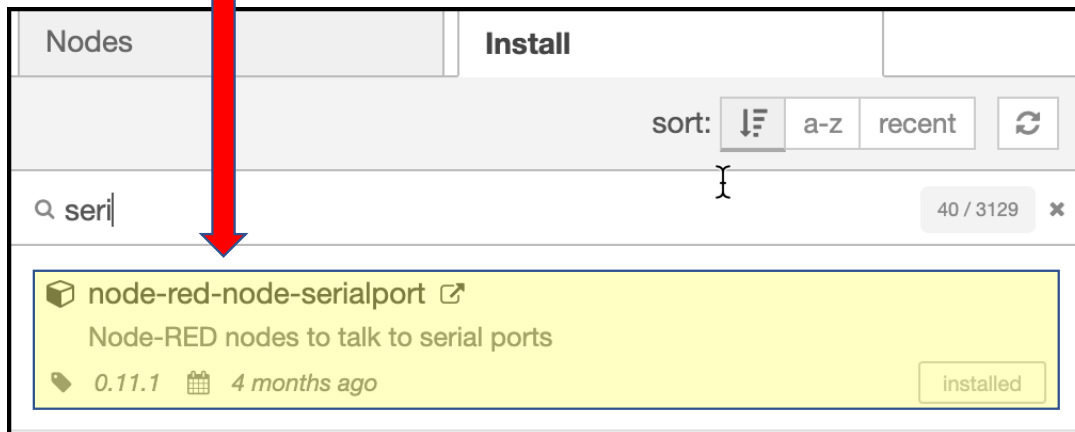
>**Delimiter** in line 15 is an indication for nodeRED to split data into array ([0],[1]...[nth]).



Node-RED: d. Layout & Installation.



Go to **Manage Palette** menu to install the Serial palette.



v1(2)-mar-21

> The main idea is to display the moisture value over nodeRED's chart dashboard.

> **Serial in** node is used in for the microcontroller to talk (communicate) with the computer (with nodeRED installed).

> **Serial widget** is not installed by default. Go to Manage Palette to install **node-red-node-serialport**.

>  indicate nodes not properly setup.

Node-RED: e. Setting the Layout.

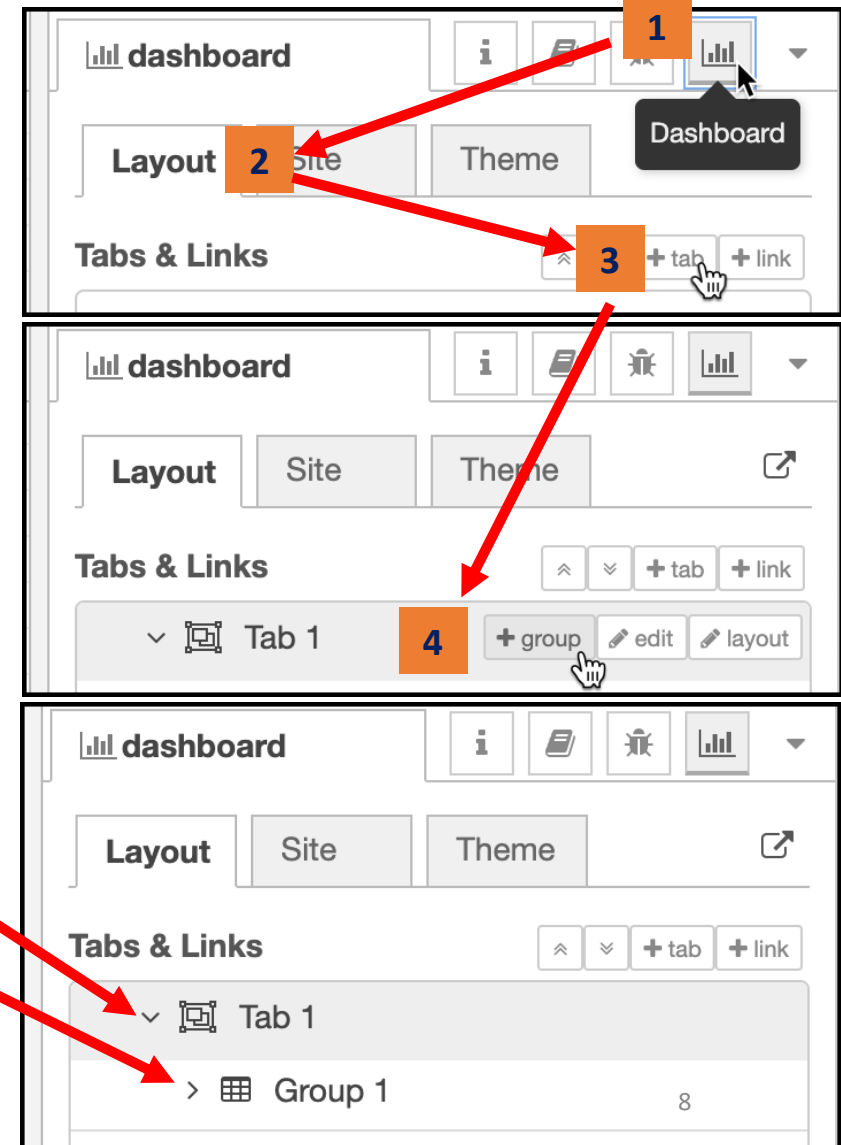
> Since the project requires dashboard UI, it is advisable to start the task by configuring the **Layout** properties located at right side panel.

> Every dashboard UI must be in a **tab** field & **group** field.

3 Click **+tab** once to create **Tab 1** field.

4 Click **+group** once to create **Group 1** field.

> Next step is to rename **Tab 1** & **Group 2** that reflect the project.

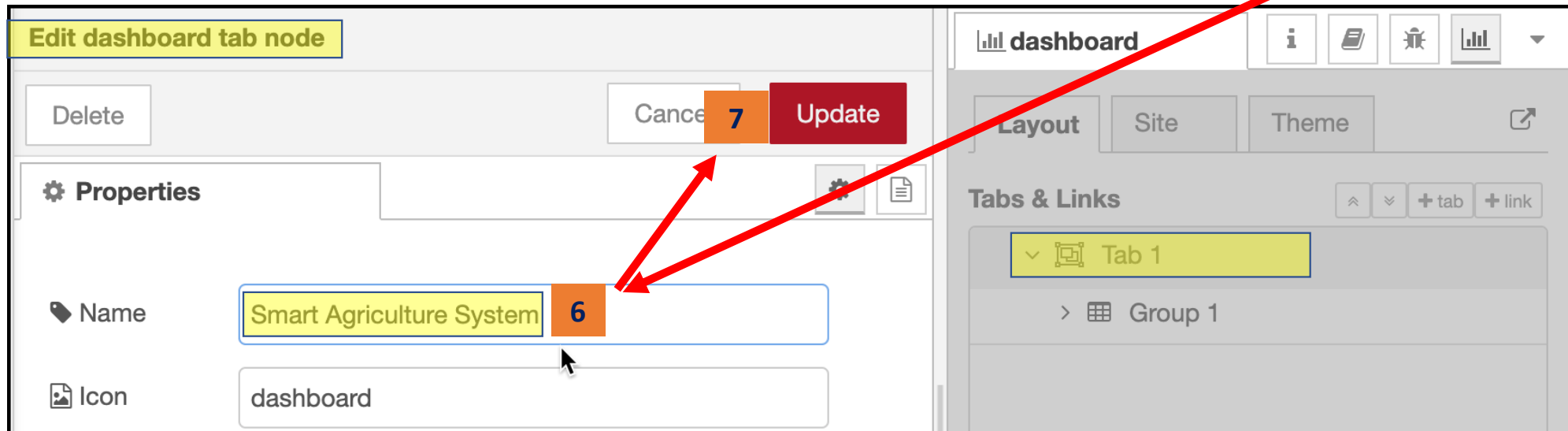
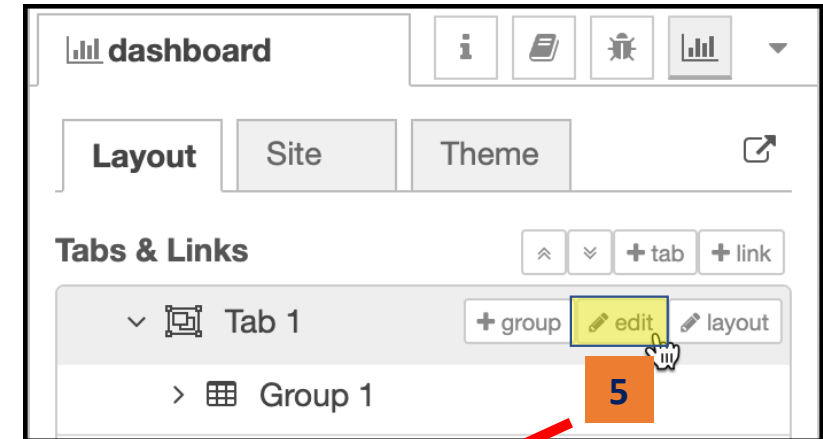


Node-RED: e. Setting the Layout.

5 Click **edit** link at **Tab 1** field.

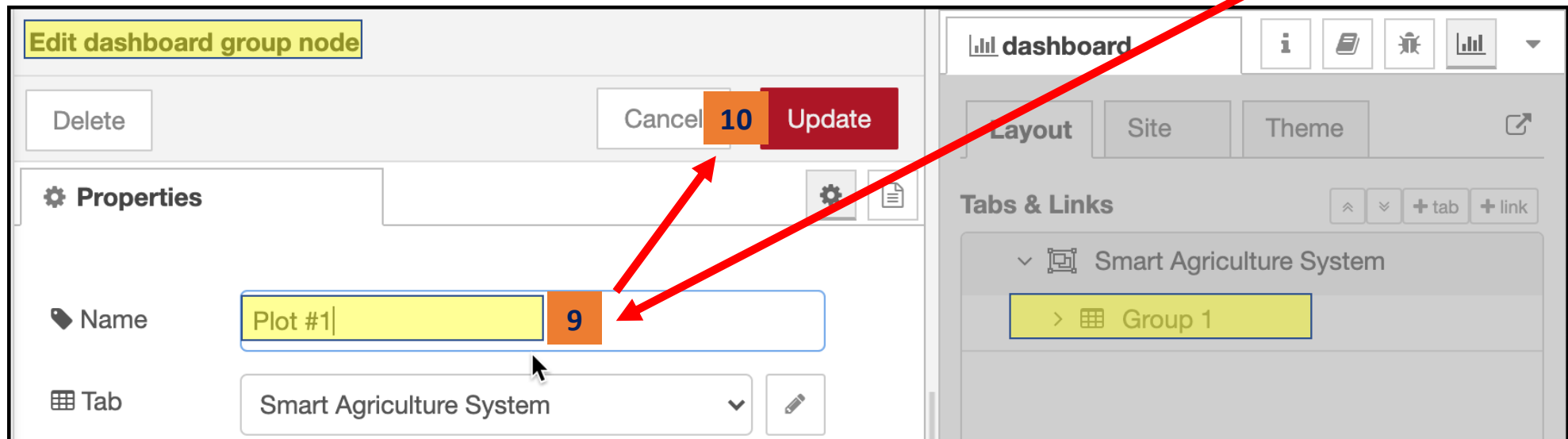
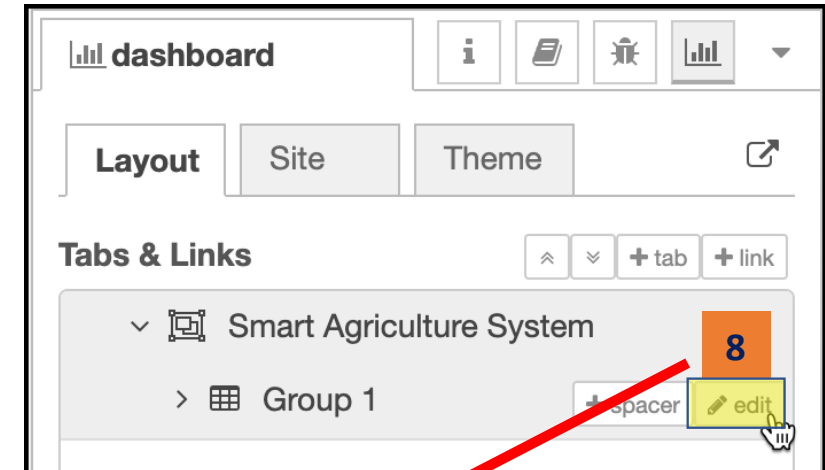
6 Change **Tab 1** to **Smart Agriculture System**.

7 Click **Update** upon completion.



Node-RED: e. Setting the Layout

- 8 Click **edit** link at **Group 1** field.
- 9 Change **Group 1** to **Plot #1**.
- 10 Click **Update** upon completion.

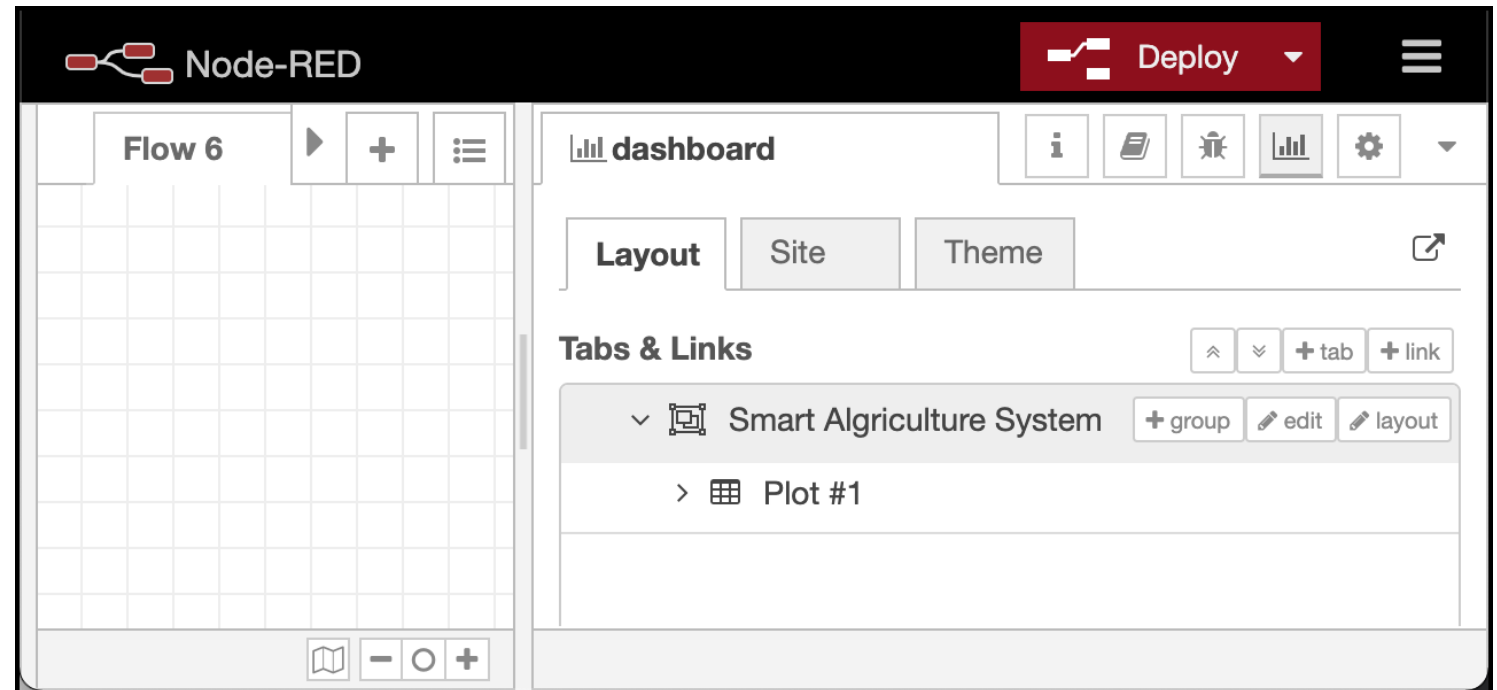


Node-RED: e. Setting the Layout.

> If you have multiple sensors attached at different **Plot**, just add **+group** and name it according the **Plot** number.

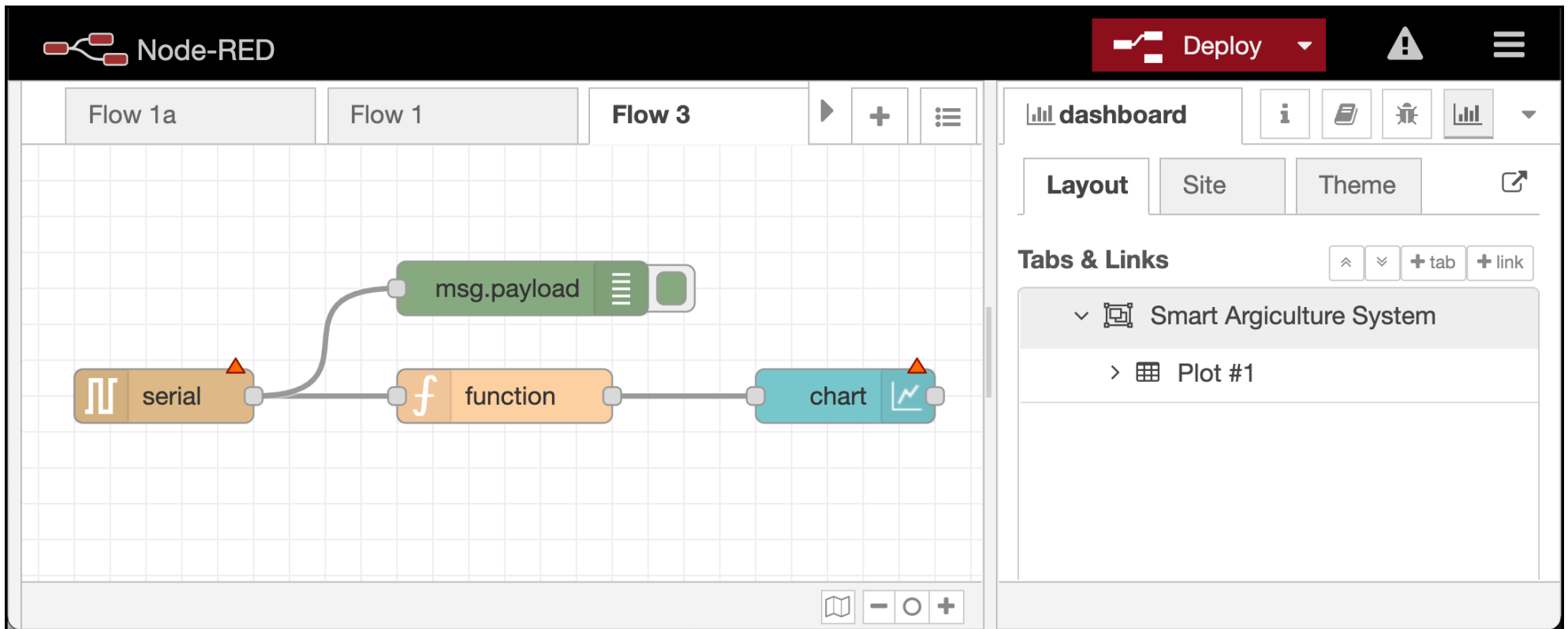
> To delete the layout, go to **edit** section and click **Delete**.

> Don't forget to click **Deploy** after every activity, otherwise, your will lost your work.



Node-RED: f. Workspace Setup.

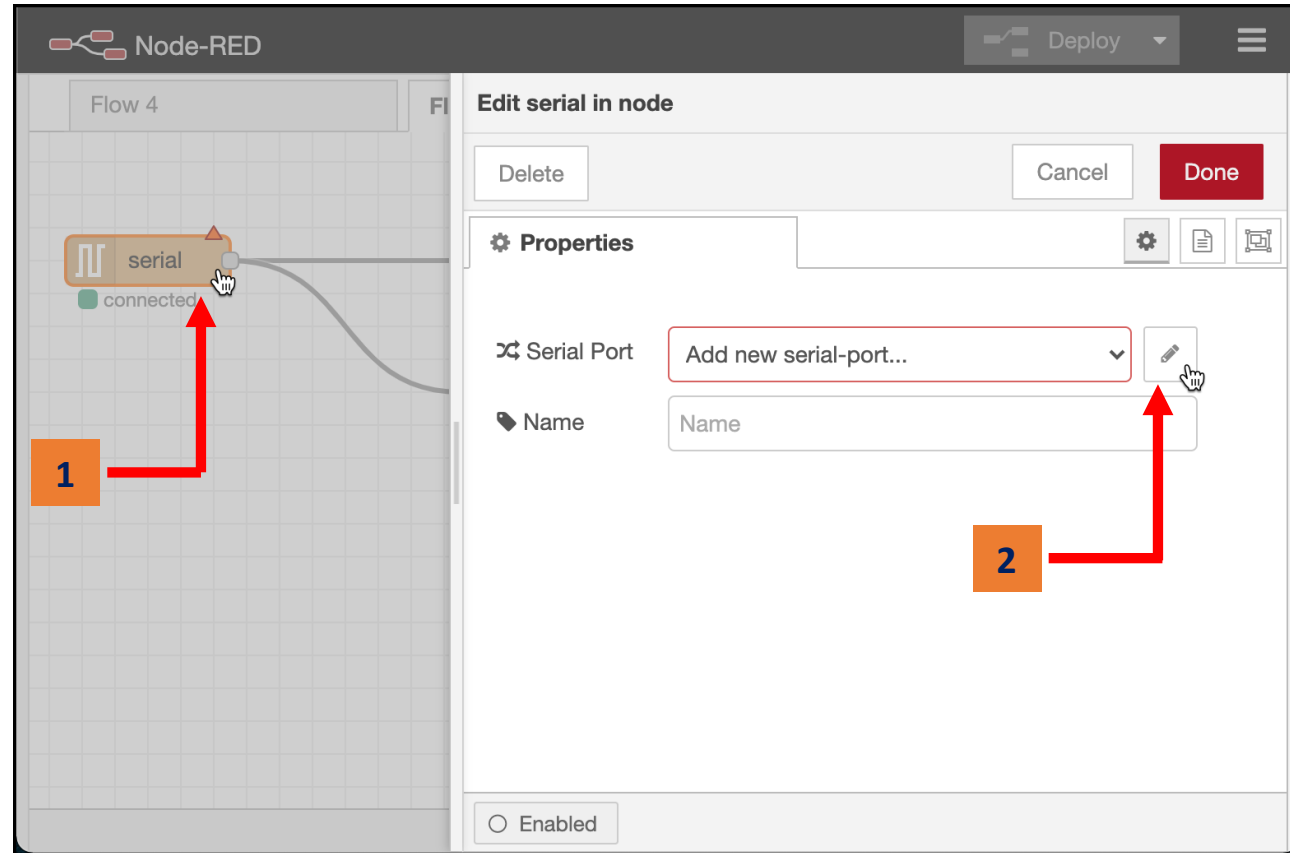
> Next process is to setup the **Serial in** node & assign the properties to **Chart** node.



Node-RED: g. Serial in Config.

- 1 Double click **Serial in** node.
- 2 Click the pencil icon to **Add new serial-port...** config node.

>The port number is the same with the microcontroller's (refer to **Arduino IDE** or **Device Manager>Port**)



Node-RED: g. Serial in Config.

3 Click the browse port icon & select the correct ***serial port***. Make sure the microcontroller is connected to your system.

4 Change the **Baud Rate** accordingly. Refer to **Slide 5 line#5 Serial.begin(9600)**.

5 Click **Add** & you will be diverted to previous page

Edit serial in node > Add new serial-port config node

Cancel Add

Properties

Serial Port

Settings

Baud Rate

Data Bits

Parity

Stop Bits

DTR

RTS

CTS

DSR

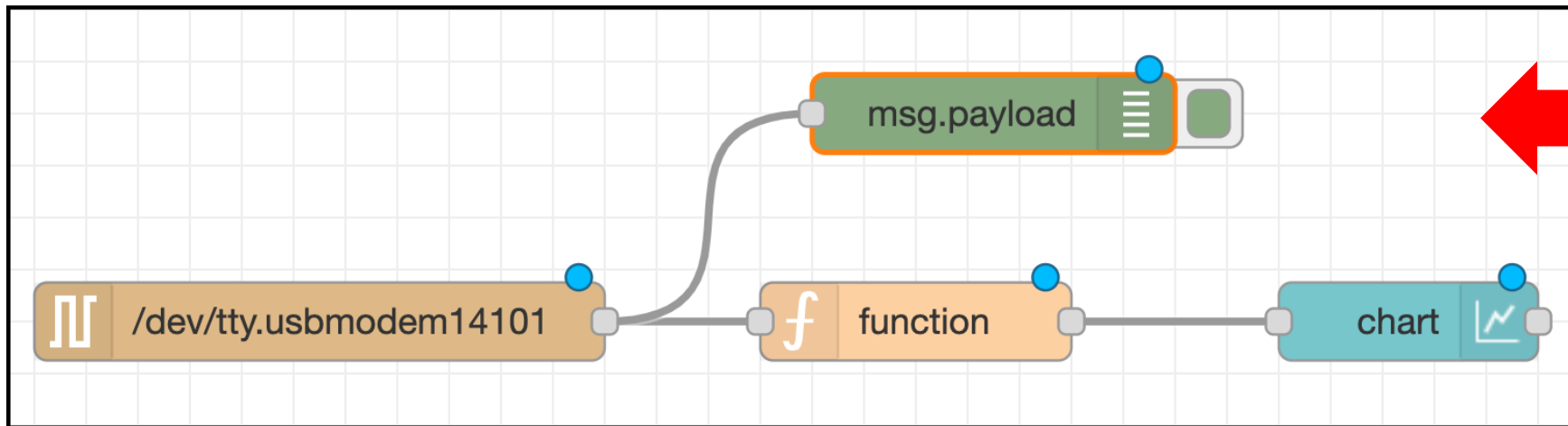
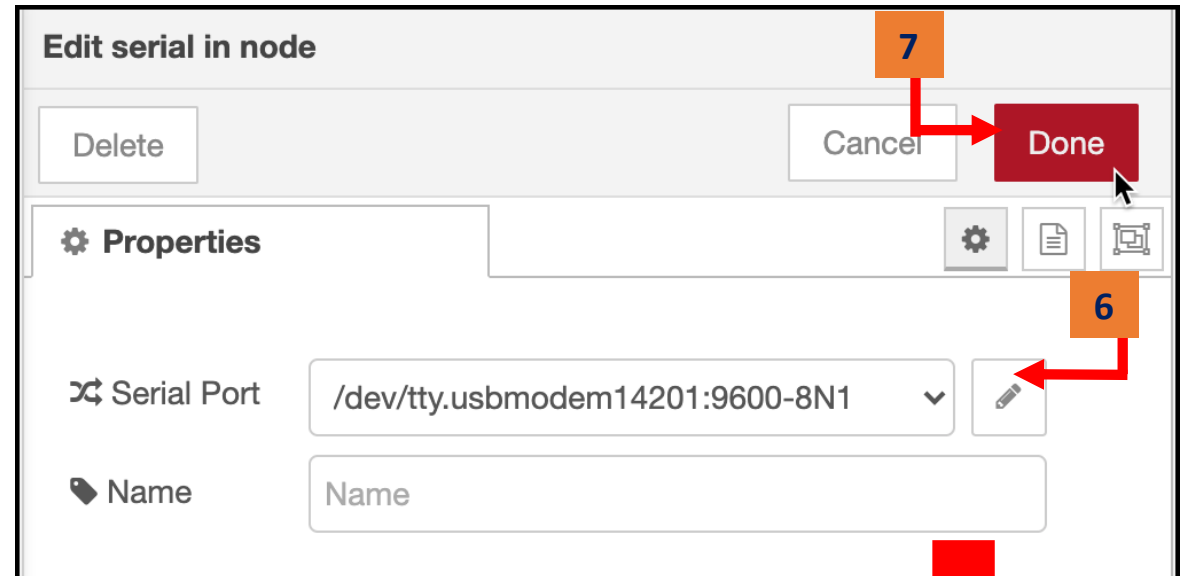
Enabled 0 nodes use this config On all flows

Other than Windows OS: /dev/tty.usbmodem14201
In Windows OS: Port xx

Node-RED: g. Serial in Config.

6 Confirm the setting? If not, click the pencil icon to edit.

7 Click Done upon completion.



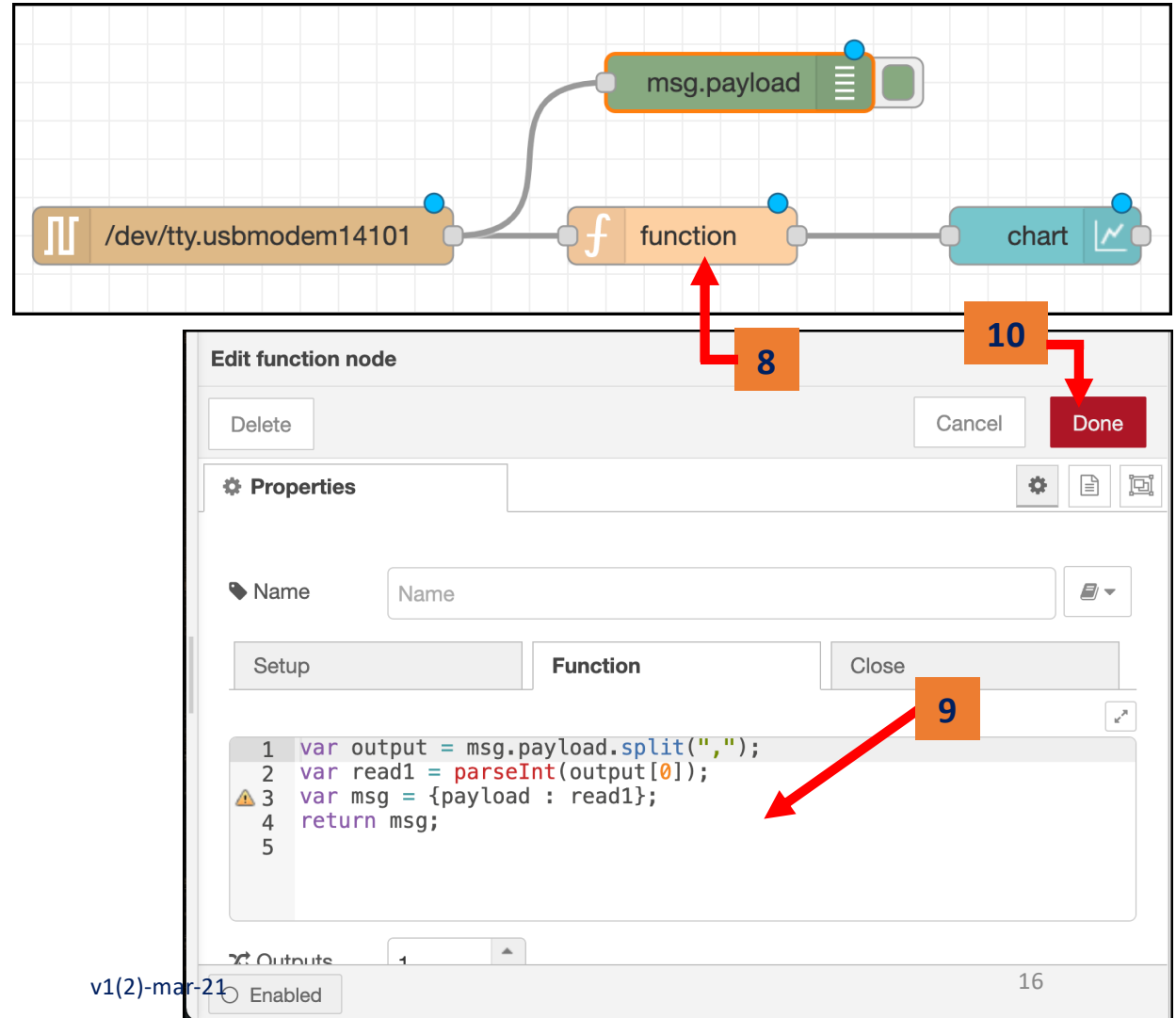
Node-RED: h. Function Script.

8 The information received from microcontroller need to be extracted before send to chart node. To do that double click **Function** node.

9 Type the following at **JS built in code editor**.

```
var output = msg.payload.split(",");  
var read1 = parseInt(output[0]);  
var msg = {payload : read1};  
return msg;
```

10 Click **Done** upon completion.



The image shows a Node-RED workflow and its function editor. The workflow at the top consists of three nodes: a serial port node labeled '/dev/tty.usbmodem14101', a function node labeled 'function', and a chart node. A message box labeled 'msg.payload' is connected to the function node. Red arrows with numbers indicate the steps: arrow 8 points to the function node, arrow 9 points to the code editor, and arrow 10 points to the 'Done' button.

Edit function node

Buttons: Delete, Cancel, Done

Properties

Name: Name

Setup | **Function** | Close

```
1 var output = msg.payload.split(",");  
2 var read1 = parseInt(output[0]);  
3 var msg = {payload : read1};  
4 return msg;  
5
```

Outputs: 1

Enabled

v1(2)-mar-21 16

Node-RED: h. Function Script.

> nodeRED will not communicate with the microcontroller if the same port is open by other program.

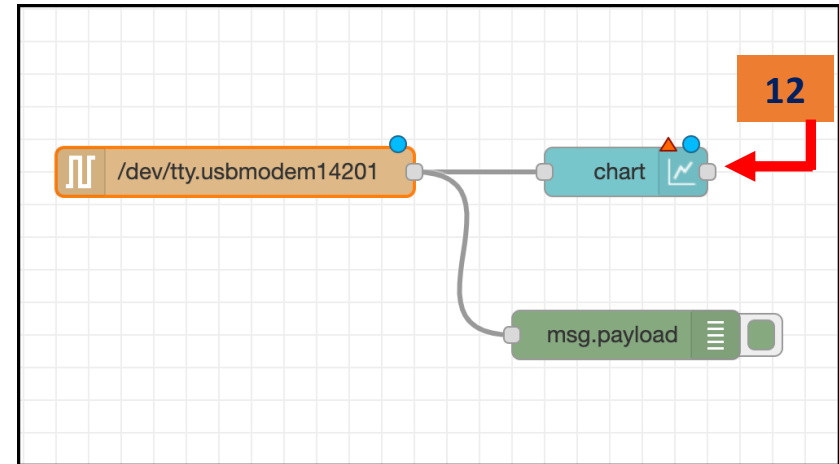
11 Click ***Debug*** tab to view the output from serial port.

The screenshot shows the Node-RED web interface. The main workspace displays a flow named 'Flow 3' on a grid. The flow consists of three nodes: a serial port node labeled '/dev/tty.usbmodem14201' (status: connected), a function node labeled 'function', and a chart node. A red dashed arrow points from a text box to the function node. The text box contains the instruction: 'Change debug wire to end of function node, observe the debug window.' The right sidebar shows the 'debug' window with a list of messages. The messages are: 'msg.payload : string[4]' followed by '"5,\r"', '16/03/2021, 13:11:00 node: bacdf265.91ca7' followed by 'msg.payload : string[4]' followed by '"4,\r"', and '16/03/2021, 13:11:00 feac40d5.8f8a7' followed by 'msg.payload : string[4]' followed by '"4,\r"'. A red arrow points from the 'debug' window to the function node in the flow.

Node-RED: i. Chart Config.

> Next step is to set up Chart's properties.

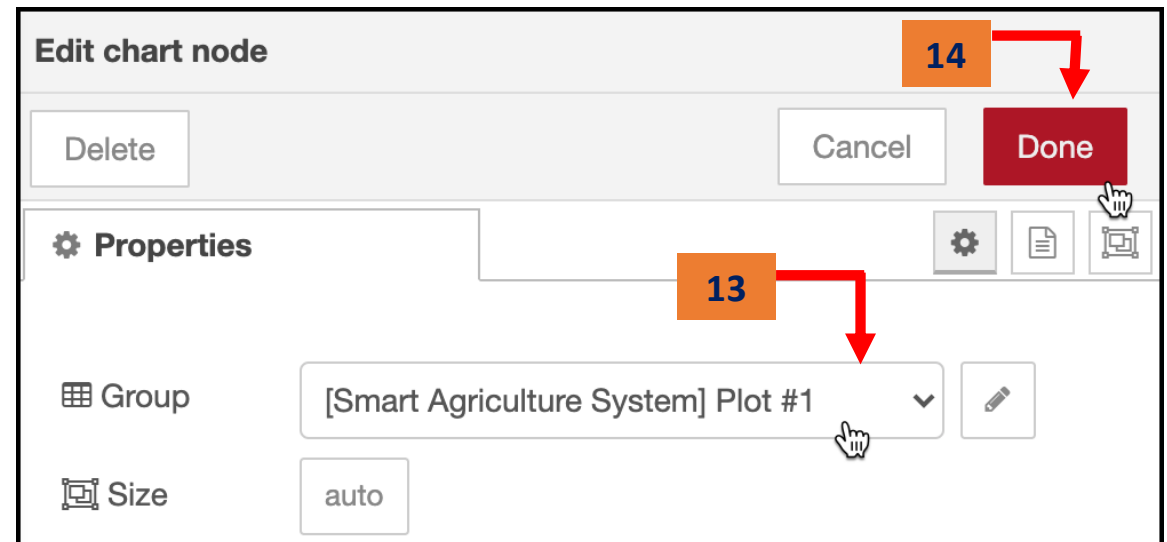
12 Double click Chart node.



13 Select [Smart Agriculture System] Plot #1.

14 Click **Done** upon completion.

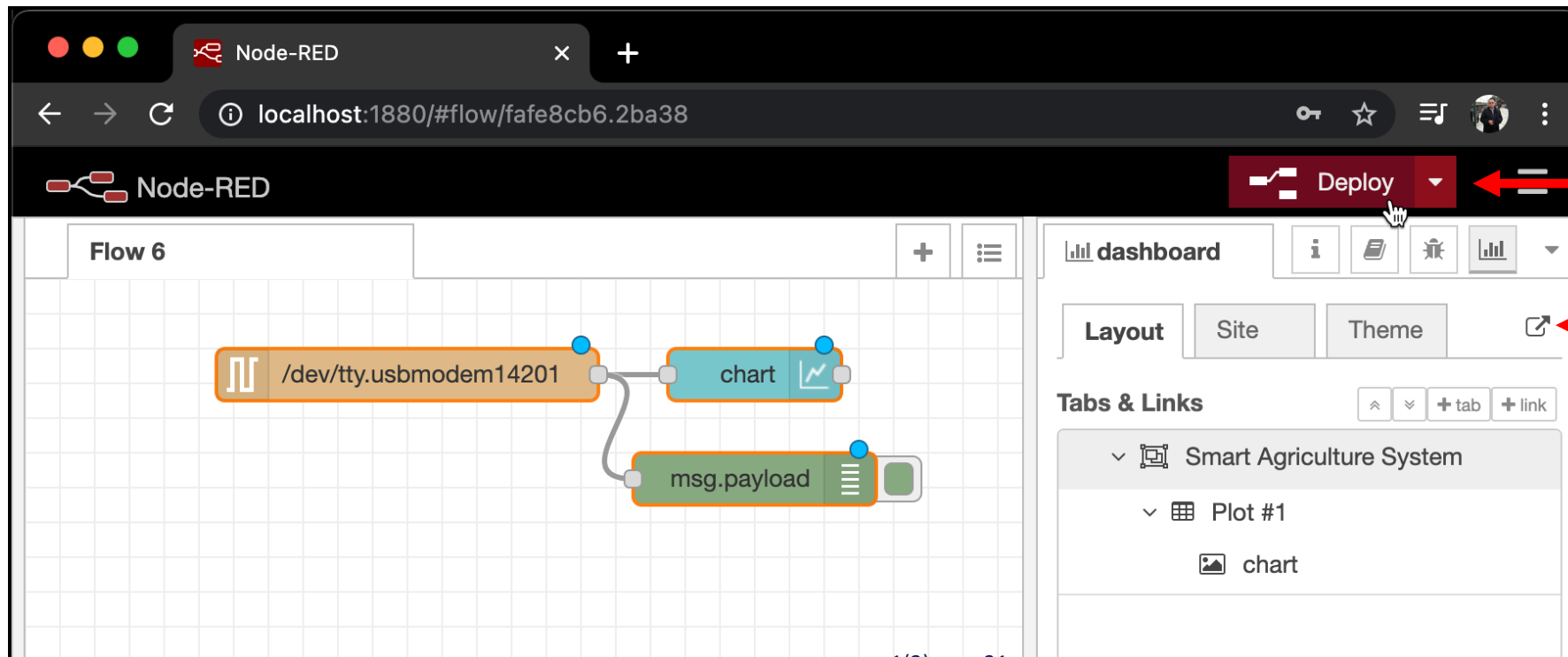
Note: Total element in group will increase if you have created series of dashboard tasks.



Node-RED: j. Prepare to Execute.

15 Click Deploy to compile the flow's update, setup and config.

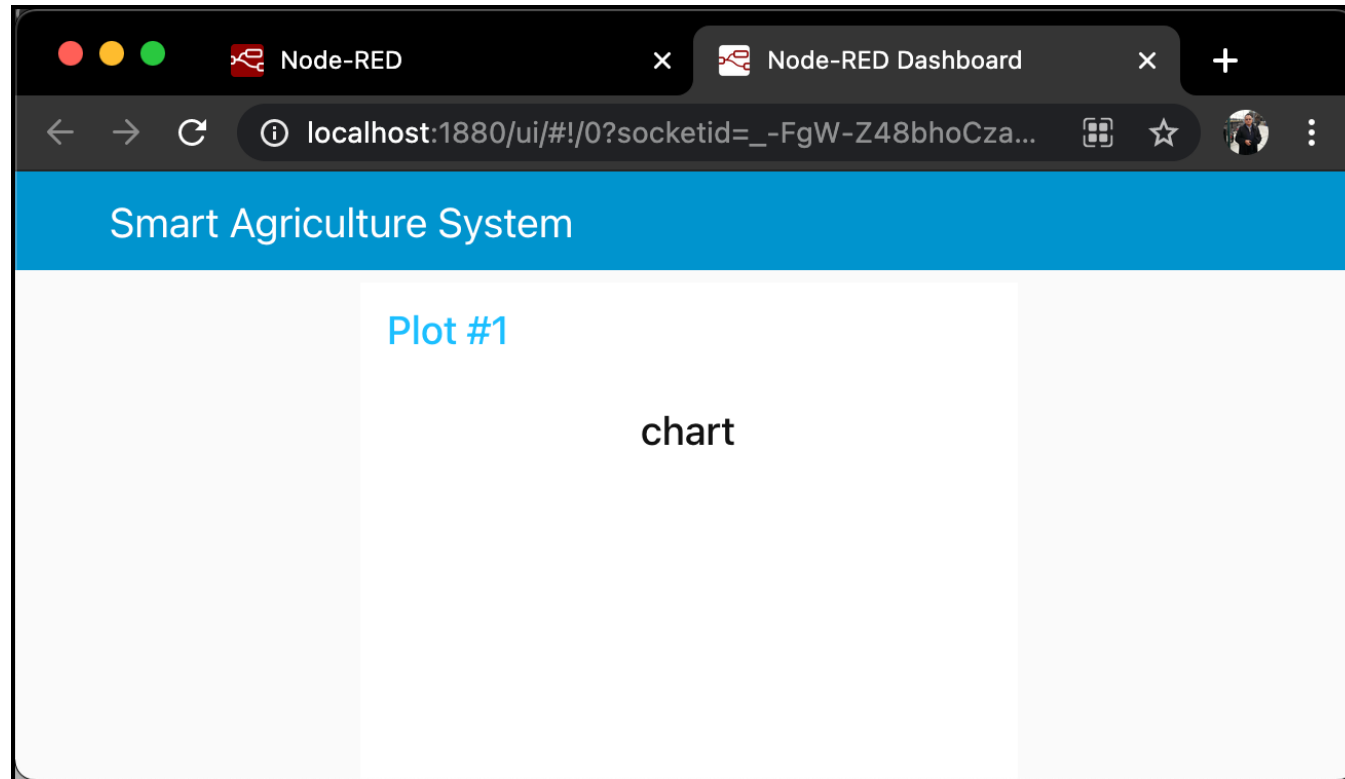
16 Click  to view the UI interface aka dashboard. (<http://localhost:1880/ui/>)



v1(2)-mar-21

Node-RED: k. The Output & Troubleshoot.

> No output produced. Go back to **nodeRED workspace editor** & click debug  message to see the output tapped at **debug** node.



Node-RED: k. The Output & Troubleshoot.

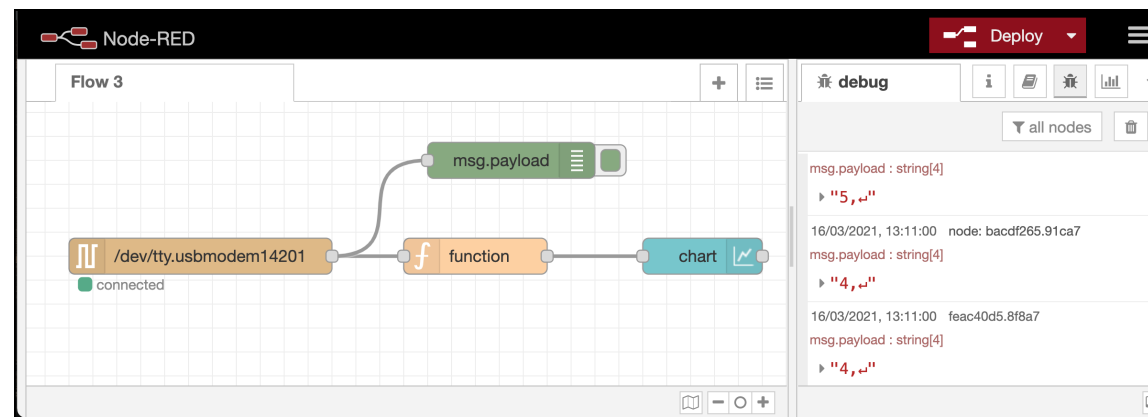
> Need to omit the '**Moisture Level:** ' text. Remark line number 13 (refer slide 5). Upload the sketch again.

The screenshot shows the Node-RED web interface. On the left, a flow named 'Flow 6' is visible. It contains three nodes: a serial input node with the text '/dev/tty.usbmodem14201' and a 'connected' status, a 'chart' node, and a 'msg.payload' node. The 'chart' node is connected to the 'msg.payload' node. On the right, the 'debug' console is open, displaying four log entries. Each entry shows a timestamp, a node ID, and a message payload. The messages are: 'Moisture Level: 678,', 'Moisture Level: 848,', 'Moisture Level: 877,', and 'Moisture Level: 764,'. The 'msg.payload' field in each entry is highlighted in yellow.

Note: You need to disconnect the board from nodeRED first, by double click the **serial in** node text & change port parameter  & click **Deploy**. When connect to nodeRED, close the **Serial Monitor**.

Node-RED: j. The Output & Troubleshoot.

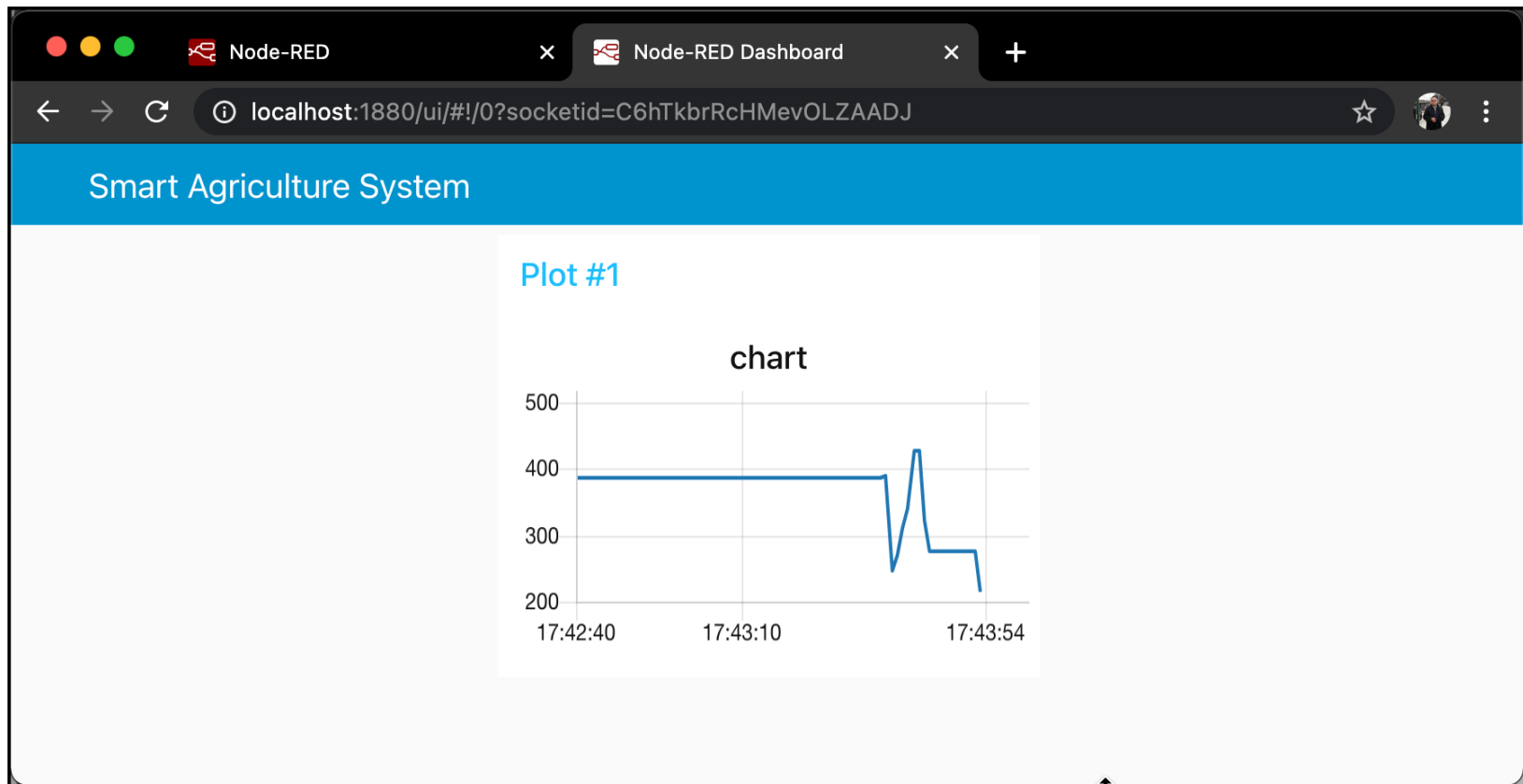
- > Output at Serial Monitor shows the reading from the sensor.
- > Similar with at **nodeRED**, don't forget to exit **Serial Monitor** before triggering the **nodeRED** serial port again. Only one application per serial port.



flows-mar-gmi-bda-cs1-1sensor.json

Node-RED: i. The Output & Troubleshoot.

> The final output.



EXERCISE:

Add one more sensor, modify the function node and also the sketch to suit your needs.

Answer: -github

flows-mar-gmi-bda-cs1-2sensors.json

nodeRED

```
/Variables
String data1, data2;
int d1, d2;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  //Read data from port 0 & 2, and store it to variables d1, d2 in integer mode
  d1 = analogRead(0);
  d2 = analogRead(2);
  data1 = String(d1);
  data2 = String(d2);
  //Print d1 and d2 values to serial monitor

  Serial.print(d1);
  Serial.print(",");
  Serial.print(d2);

  delay(2000); //Delay 2 sec.
}
```

arduino

QnA

END