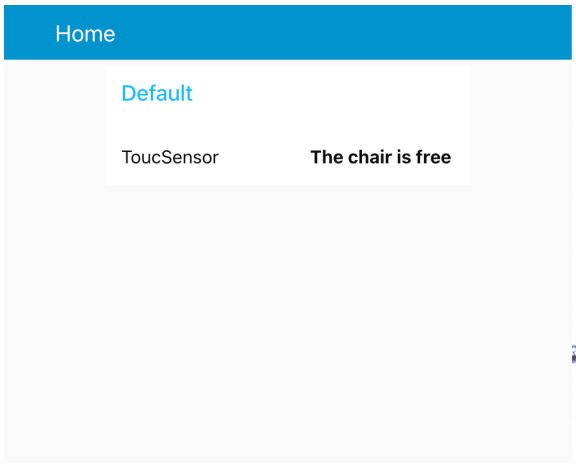
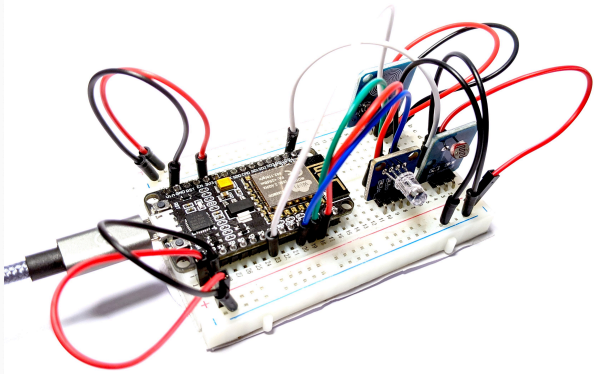
Node-RED

Reading data from the serial port

Project description:

In this project, you will learn how third-party applications can read data sent through serial port communication. Precisely, you will setup Node-RED, an open source tool for visual programming developed by IMB, to read incoming serial data from the Arduino IDE. This will show you an example of how you can use the Arduino Serial Monitor to send data to other applications. There is no need for you to build a separate circuit for this exercise. You will use the smart chair circuit that you have built during the course and upload a different code sketch.



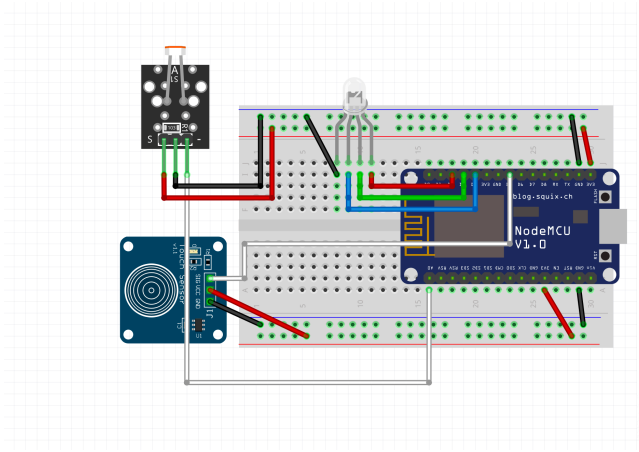


Project objectives:

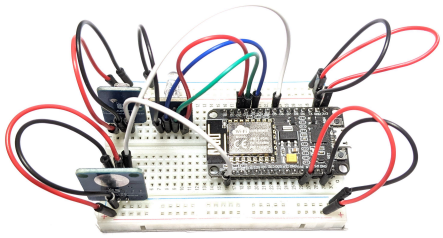
- Modify the smart chair circuit behaviour with alternative code
- Use the Serial Monitor to print data
- Use Node-RED to read incoming data sent from the Arduino Serial Monitor

Project components:

Component Reference	Component Quantity	Component Name	Component Description
A	1	Smart Chair circuit	The smart chair circuit which features the photoresistor, the RGB LED, and the touch sensor
B	1	Micro USB Cable	A USB cable to power and upload instructions to a microcontroller



A



B

Step One
Project Introduction

The idea behind this project is for you to explore how serial communication can be used as a protocol to share data across different applications. You will modify the smart chair code to only use the touch sensor to identify if the chair is free or taken.

Writing the touch sensor output to the Arduino Serial Monitor, will make such data available for other applications to listen to via the serial port.

You program should mimic an available-chair detector and simply check the status of the touch sensor. If the sensor is “HIGH”, meaning that you are touching it, the program should print “The chair is busy” to the Serial Monitor. On the other hand, if the sensor is “LOW”, the program should print “The chair is free” to the Serial Monitor.

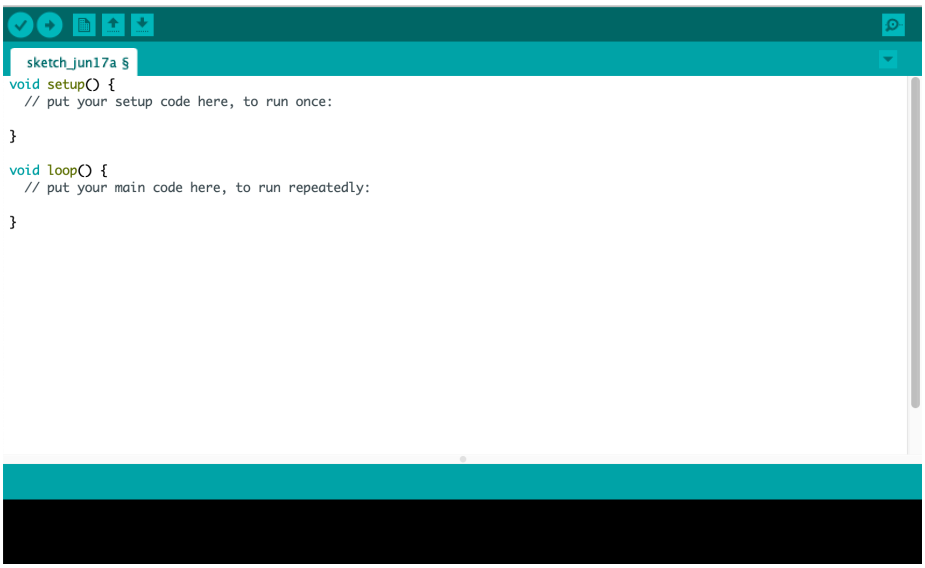
Once such data is sent to a specific COM port via the Serial Monitor, other applications such as Node-RED can listen on the same COM port and retrieve the data. You will then use Node-RED to read the data message and display it on a simple dashboard.

Node-RED is an open-source programming tool, for connecting hardware devices, APIs and online services creatively and easily. Primarily, it is a visual tool designed for the Internet of Things. Node-RED runs on Windows, Mac, and Linux computers. It uses visual blocks/nodes to construct the logic flow and it is relatively easy to use to create complex data flow.

Step Two
Writing the Code

Now that you are familiar with the task, it is time to write the code for the chair detector. Go ahead and create a new empty sketch from the Arduino IDE.

You should see an empty sketch like the following:



Feel free to save the sketch and rename it to something sensible: **nodered_chair_detector** for instance.

There is one core functionality that we want to program here:

- Print a message to the Serial Monitor according to the touch sensor reading

Let's begin by initialising some variables at the to of the sketch:

```
// Initialise the touch sensor pin
// The chair value coming from the touch sensor
const int touch_sensor = D5;
int chair_busy;
```

touch_sensor is a reference to pin D5, the pin where the touch sensor is wired up on you smart chair circuit. **chair_busy** is an integer variable to store the reading value coming from the touch sensor.

Next we want to add some code to the setup() function:

```
// Put your setup code here, to run once:
void setup() {

  // Touch sensor as INPUT
  pinMode(touch_sensor, INPUT);

  // Start the serial to debug the values
  Serial.begin(9600);

}
```

Here, we simply set the touch sensor pin mode to INPUT as we want to read the incoming signal from the sensor. Furthermore, we start the Serial Monitor with the data rate of 9600 bits per second.

Now, we need to create the logic for the chair detector.

The **chairSignal()** utility function:

```
//  Utility function to control the chair availability
void chairSignal(){

  chair_busy = digitalRead(touch_sensor); // Read the value of the touch sensor (0--1)

  if(chair_busy == HIGH){
    Serial.println("The chair is busy"); // The chair is busy
  }else{
    Serial.println("The chair is free"); // The chair is free
  }

}
```

Here, we first read the digital incoming value from the touch sensor (0 or 1) , and then we write either “The chair is busy” or “The chair is free” data to the Serial Monitor.

Finally, we need to call the **chairSignal()** function inside the **loop()** function for the Serial Monitor to print our data.

The **loop()** function:

```
// put your main code here, to run repeatedly:
void loop() {

   // Check for chair availability
   chairSignal();

}
```

Hurray, you have successfully completed the code section of the circuit.
Go ahead, compile your code, and upload it to your ESP board.

Once the code has been uploaded, open the Serial Monitor and you should see “The chair is busy” when you touch the sensor and “The chair is free” when the sensor is not touched.

Step Three
Node-RED installation

Now that you have your ESP board sending data to the serial port, we can use other applications such as Node-RED to listen to the same port and access such data. Let’s first install and run Node-RED.

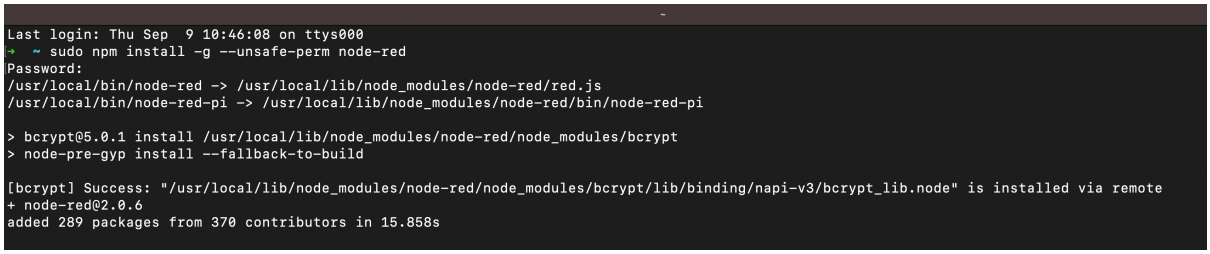
Here you can find information on how to install Node-RED locally on your machine (follow the NPM instructions):

https://nodered.org/docs/getting-started/local

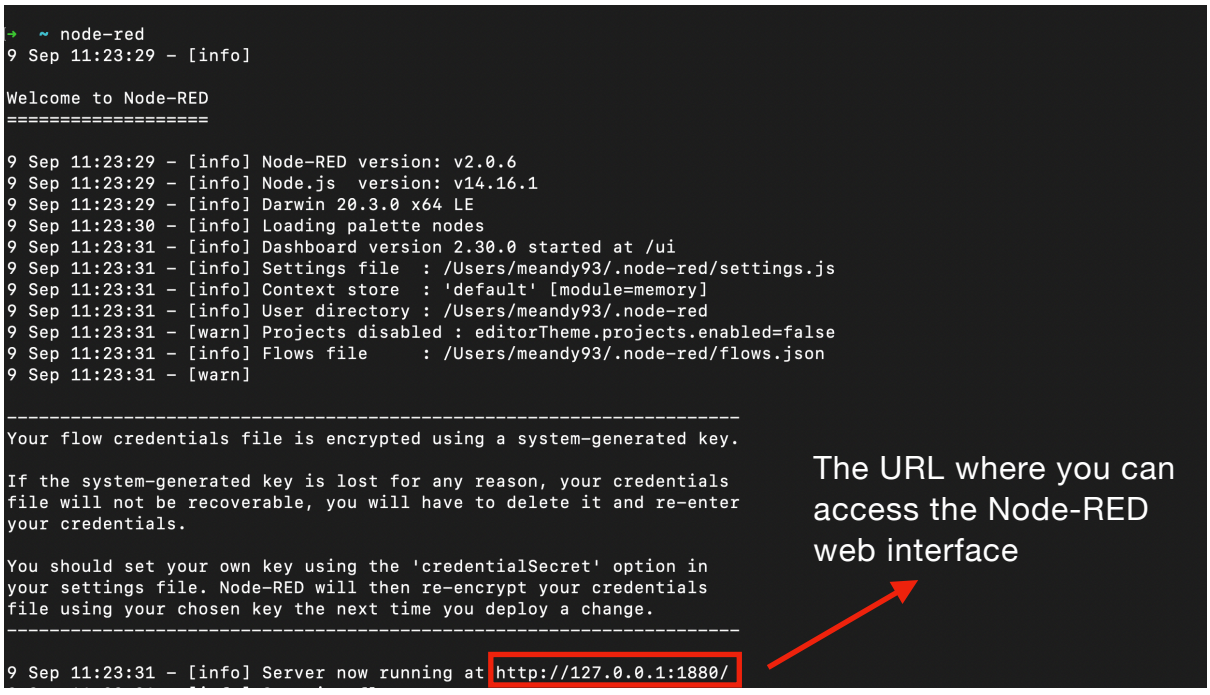
There are slightly different instructions for **Windows** users:

https://nodered.org/docs/getting-started/windows

Open your computer Terminal and run your OS suggested commands for the installation (sudo npm install -g —unsafe-perm node-red):

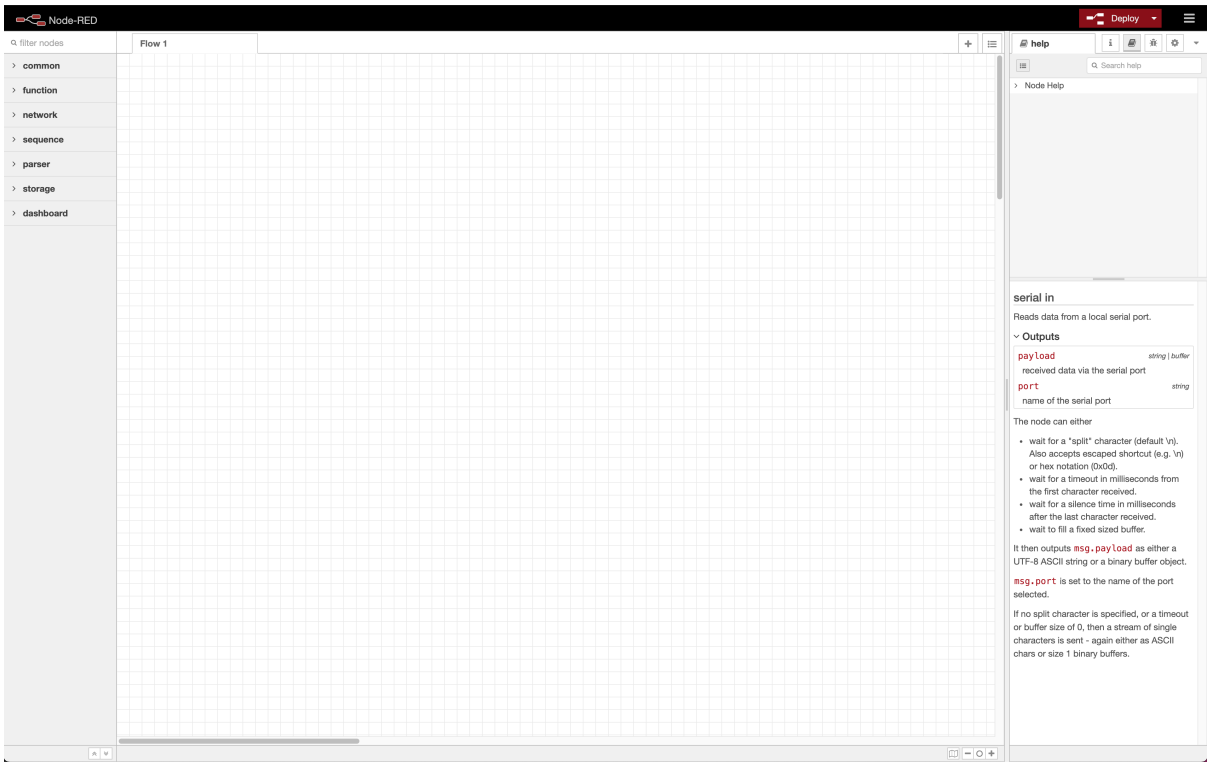


Once installed as a global module you can use the node-red command to start Node-RED in your terminal (node-red):



Once you run the node-red command, you can access the application at your given URL as shown on the image above. Open your Web Browser and access the given Node-RED URL.

You should see the following dashboard:



Great, you have now installed and run Node-RED.
In the next section you will see how you can setup Node-RED to read the serial data that your ESP board is sending the COM port.

Step Four
Node-RED: Reading data from serial port

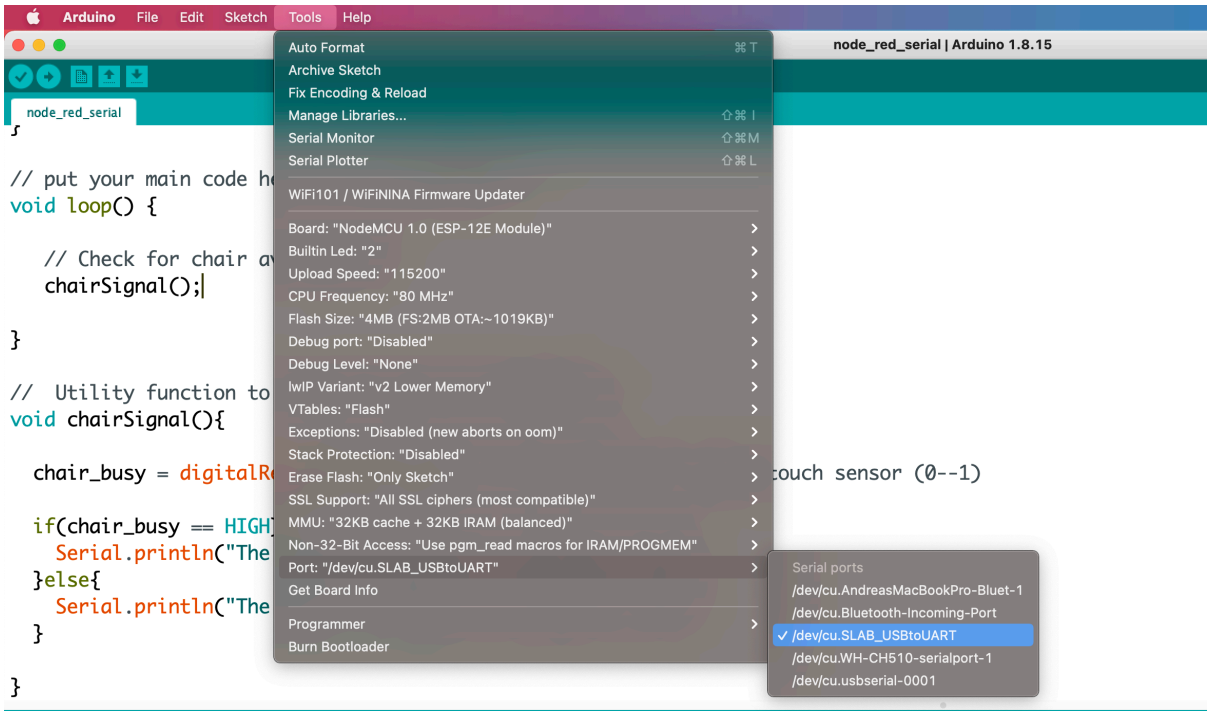
Now that you have Node-RED working, it is time to configure it to listen to the same COM port as where your ESP board is sending data to. This will allow Node-RED to retrieve the data and display them accordingly.

Close the Node-Red process from the Terminal for now.

First thing first, make sure that your circuit is working fine and that you can see the messages on the Serial Monitor. Leave your circuit powered and the Serial Monitor opened for the moment.

Now, let's find the name of the serial port that you ESP board is sending data to. You will need the name of the port to then configure Node-RED to listen to the same port as your ESP board.

With your sketch open, you want to navigate to **Tools** > **Port:** as shown on the image below:



In my case, the PORT name is **“/dev/cu.SLAB_USBtoUART”**.
Make a note of that as you will need it to configure Node-RED.

Now that you have the serial port name, it is time to configure Node-RED to listen to such port, read the data messages, and display the data on a web dashboard.

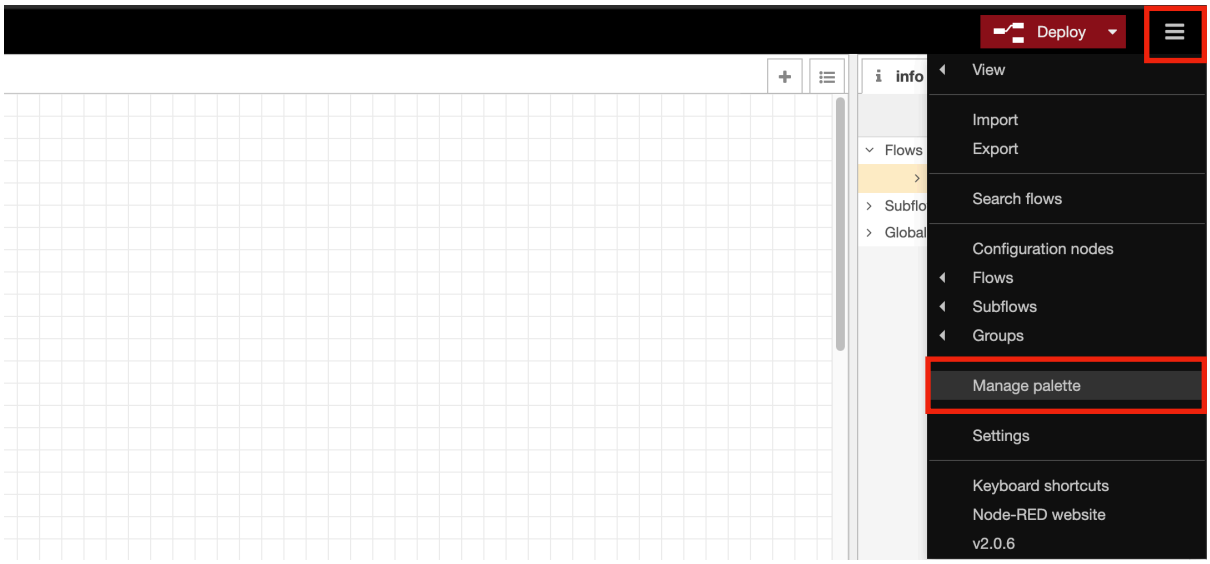
Go ahead and close the Serial Monitor from the Arduino IDE if it is open but leave the circuit plugged in via USB. Use the terminal to launch Node-RED once more and open the application dashboard on your Web Browser.

There are two nodes that we need to add to the Node-RED dashboard.
One serial node to listen incoming data from the serial port and one text node to display such data on a simple web dashboard.

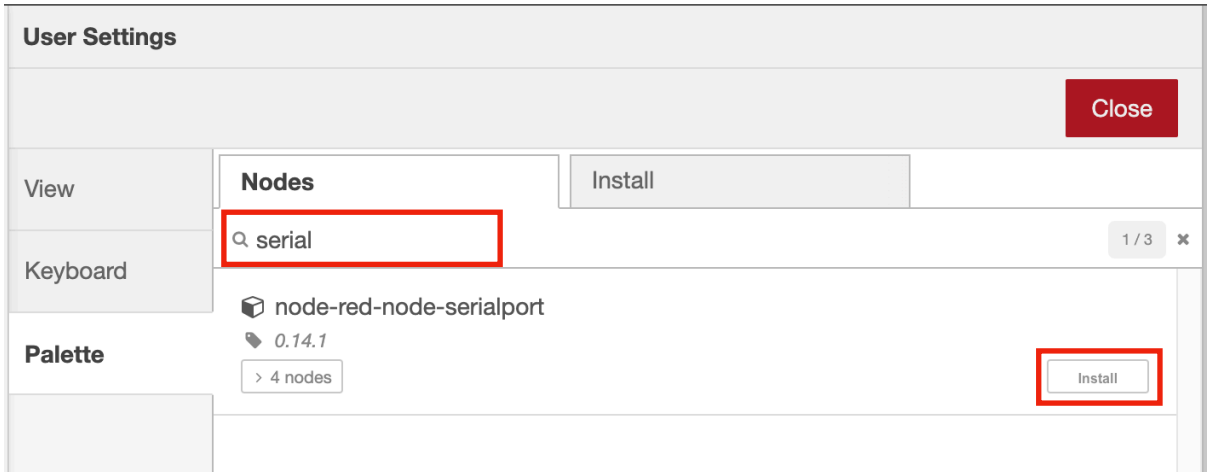
By default, Node-RED does not have any nodes for serial connection but you can add them similarly to how you add libraries inside the Arduino IDE.

You can see a list of available nodes on the left hand side of the dashboard.

Let's add the serial connection node. On the top right hand side click on the hamburger menu first and then click on the "Manage palette" option:

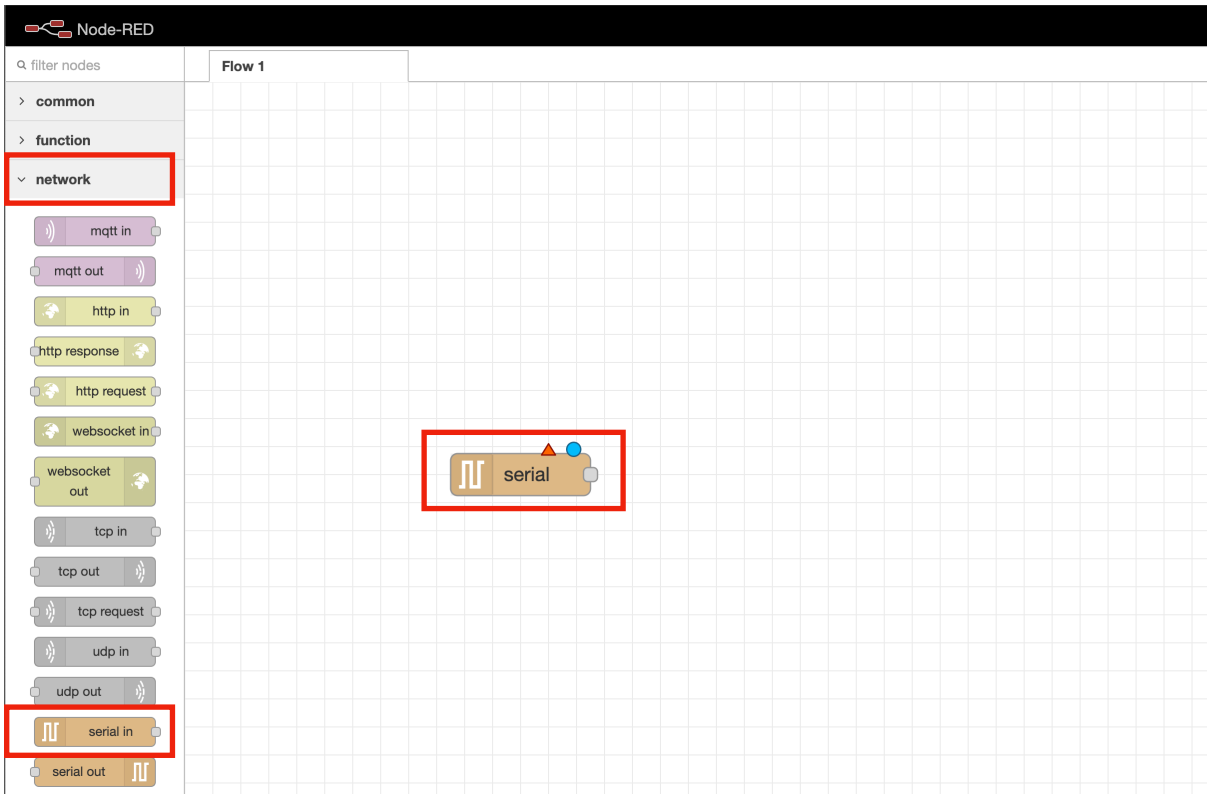


Now, type "serial" on the search bar and install **node-red-node-serialport** as show below:



Now that you have the serial communication node added to the library, it is time to add it to the Node-RED dashboard.

Select the “serial in” node from the **network** section on the left hand side of the Node-RED dashboard and drag it to the design area as show below:

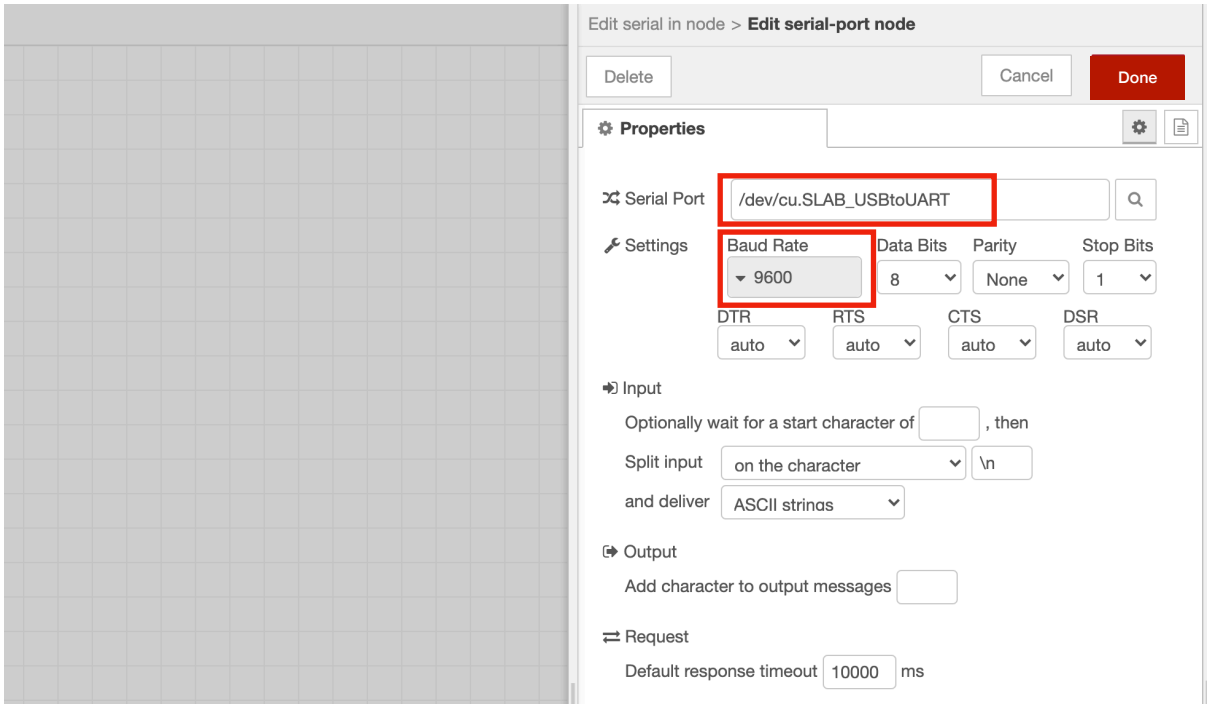


The **serial in** node is useful to listen for incoming data on a specific serial port. You now have to configure it to listen to your ESP board serial port, the port name that you checked earlier.

Double click on the serial node to configure it and add the following properties:

- **Serial Port:** The serial port that your ESP board is communicating to
- **Baud Rate:** 9600

See below my configuration of the serial node:

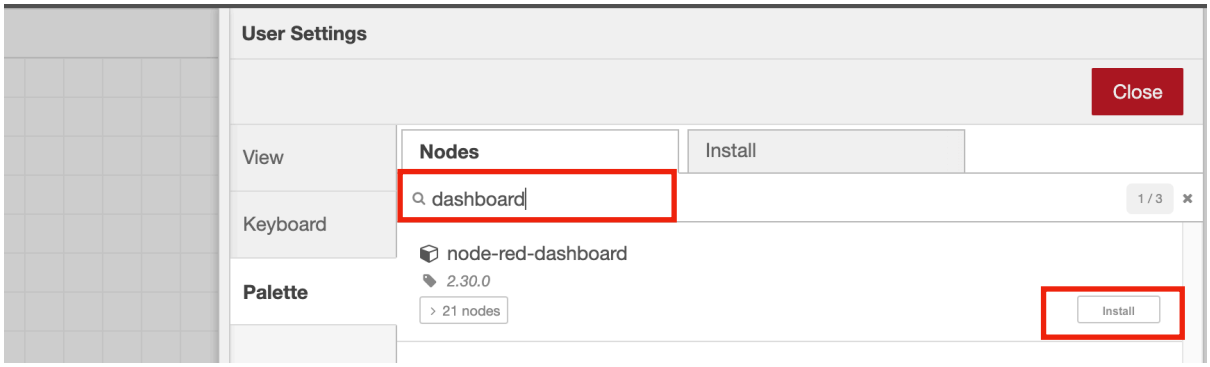


Click on the “Done” button on the top right hand side of the properties section once finished.

At this point, we have finished configuring the serial read node. We now need to add a text node to display the received data on a dashboard. Similarly to the serial node, Node-RED does not have any nodes for such behaviour and you will need to install them.

Once more, click on the hamburger menu located on the top right hand side of the dashboard first and then click on the “Manage palette” option.

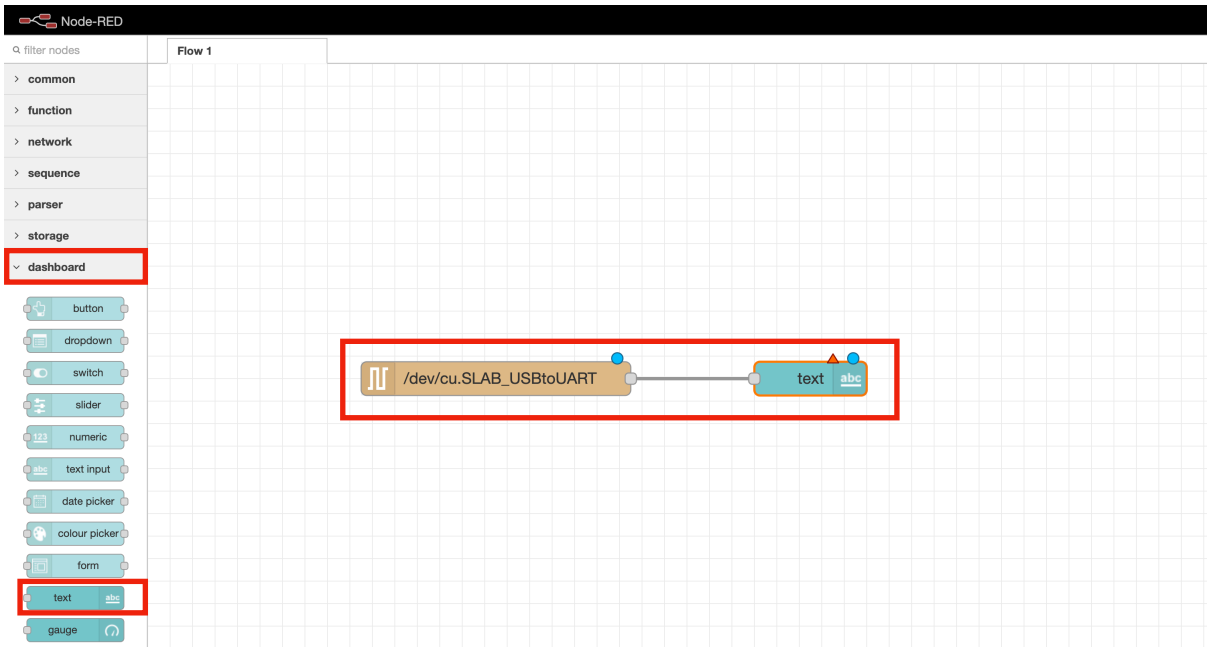
This time you want to search for “dashboard” and install **node-red-dashbaord** as shown below:



Now that you have the dashboard text node added to the library, it is time to add it to the Node-RED dashboard.

Select the “text” node from the **network** section on the left hand side of the Node-RED dashboard, drag it to the design area, and connect it to the “serial” node as show below:

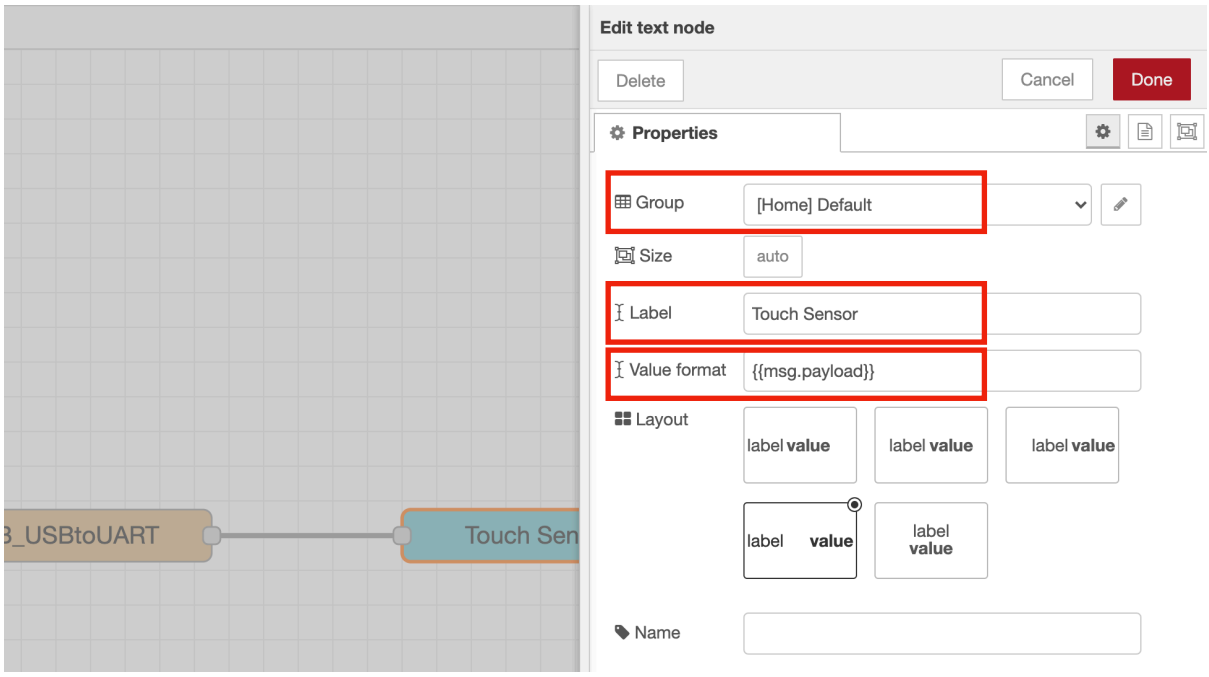
The **text** node is useful to display data on a simple dashboard.
You now have to configure it to display the data from the serial node.



Double click on the text node to configure it and add the following properties:

- **Group:** [Home] Default
- **Value format:** {{msg.payload}}
- **Label:** {{Touch Sensor}}

See below my configuration of the text node:

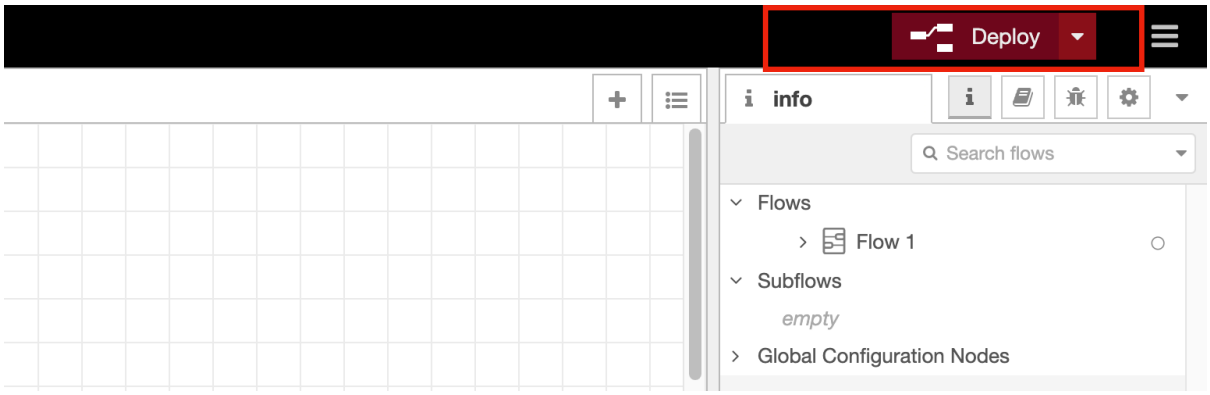


Click on the “Done” button on the top right hand side of the properties section once finished.

At this point, we have finished configuring the text node.

Now you want to deploy this configuration.
Make sure the Serial Monitor on the Arduino IDE is closed and the circuit is plugged in via USB to your computer.

Click on the “Deploy” button on the Node-RED application dashboard. The button is located on the top right hand side:



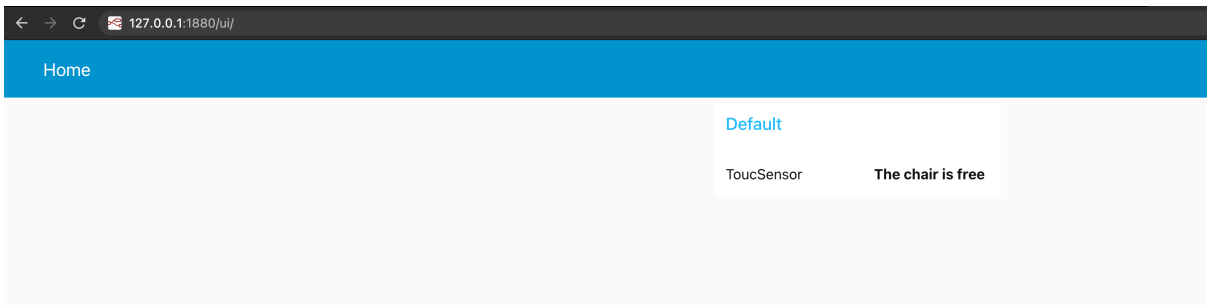
Well done, the configuration is now finished. Your ESP board will send a message through the serial port. Such message will be picked up by Node-RED serial node and displayed on a dashboard by the text node.

In order to see the data dashboard, you need to append the following route to your Node-RED application URL:

http://127.0.0.1:1880/ui/

Remember that the first part of the URL might differ from the one above.

When you access the above URL, you should see the following:



Hurray, you have successfully complete this exercise and setup a communication between your ESP board and Node-RED. Go ahead and touch the sensor on your circuit, you should the data dashboard update with the “The chair is busy” message.

You can learn more about Node-RED here:

https://nodered.org/

Feel free to add any modifications to your circuit and display different data on the Node-RED dashboard.