



**University of  
Nottingham**  
UK | CHINA | MALAYSIA

# **Department of Electrical & Electronic Engineering**

## **EEEE1002: Session 4 Episode 1 Vehicle to Anything (V2X) Communication**

# EEEE1002: Session 04 Episode 01

## Vehicle to Anything (V2X) Communication

### Overview

As covered in the introduction to session 4 you will be setting up a wireless communication system to enable data to be shared between your EEEBot, a central controller and ultimately anything else configured to work on the same network. You will be provided with a R-Pi 3 Model A+ (R-Pi) that will act as the central controller, which will host the Mosquitto MQTT broker and Node-RED dashboard.

This guide will walk you through the installation of the Mosquito MQTT broker and the Node-RED on your R-Pi. This will enable you to develop a web based control panel to send/receive data to/from your R-Pi via a Node-RED dashboard.

The development of a Node-RED dashboard will be demonstrated through a simple example of a Node-RED application that will be used to control a remote ESP32 and receive data from it. In this example the ESP32 is interfaced to a temperature and humidity sensor (BME280) and data from this sensor is collected by the ESP32 and sent to the R-Pi and displayed on the Node-RED dashboard.

In addition to acting as a tutorial for publishing and subscribing messages using the ESP32 and MQTT, this document also further demonstrates **key technical report writing elements** such as a system overview, wiring schematics, realised circuits, key code snippets and serial monitor output displays.

**Note:** the example presented uses a different sensor to those available on the EEEBot, so you will need to adapt the example code to accommodate your sensors. **This is a key engineering skill as, typically, information online will not entirely match your specific application, and so the key appropriate information and/or code snippets need to be identified and used.**

## Table of Contents

Overview .....	2
Table of Contents .....	3
1. Mosquitto MQTT Broker & Node-RED on the R-Pi .....	4
1.1 Installing the Mosquitto Broker on the R-Pi OS .....	4
1.2 Setting Up the Mosquitto MQTT Broker .....	5
1.3 Testing the Mosquitto MQTT Broker Installation .....	6
1.4 Installing Node-RED .....	7
1.5 Installing the Node-RED Dashboard .....	7
1.6 Testing the Node-RED Installations .....	7
2. ESP32 MQTT Example – Publish & Subscribe with the Arduino IDE .....	9
2.1 Constructing the Circuit .....	9
2.2 Setting Up the Arduino IDE .....	10
2.3 ESP32 Code Breakdown .....	11
3. Design Task – Vehicle to Anything (V2X) Communication .....	15

## 1. Mosquitto MQTT Broker & Node-RED on the R-Pi

This presumes you have a R-Pi set up that has access to the internet. In your case, this will be the R-Pi 3A+ board and can be done by following the two set-up guides on Moodle under 'Session 4: Vehicle to Anything (V2X) Communication':

1. EEEE1002 From zero to Pi 3 A+ with zero additional hardware in about 60 minutes
2. EEEE1002 Connect R-Pi to Eduroam

The following is a sucking guide covering everything you need to get you started, but for further detail please refer to the following:

<https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/>

<https://randomnerdtutorials.com/getting-started-with-node-red-dashboard/>

<https://randomnerdtutorials.com/getting-started-with-node-red-on-raspberry-pi/>

**Linux Tech note:** Your R-Pi will be running the Linux operating system which is **case sensitive**. If things don't seem to be working always check this first along with any required spacing between elements of a long string of commands and arguments

### 1.1 Installing the Mosquitto Broker on the R-Pi OS

Start by opening a new R-Pi terminal (sometimes called command prompt) window. This will look identical to Figure 1 below, with the exception of the 'pi@raspberrypi' label being your specific hostname.

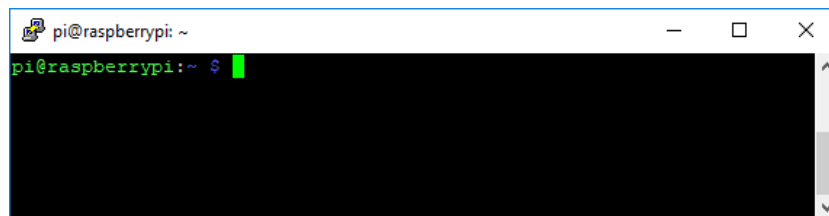


Figure 1: Blank R-Pi Terminal Window

Again, we shall ensure the R-Pi OS is up to date using the following command:

```
sudo apt update && sudo apt upgrade
```

**Linux Tech note:** this command is comprised of three separate commands **sudo**, **apt update** and **apt upgrade**

**sudo** this command allows you to run a command (or application) with the privileges of another user, in this case the **superuser**. This is necessary as the next two commands are used to modify system files

**apt update** updates the list of available packages and their versions, but it does not install or upgrade any packages

**apt upgrade** installs newer versions of the packages have already installed in your specific OS. After updating the lists, the package manager knows about available updates for the software you have installed

You could run **sudo apt update** and then run **sudo apt upgrade**, but here they are run in sequence from one line using the **&&** operator

Press 'Y' and 'Enter' when prompted to do so – note this process may take some time depending on when the the system was last updated.

The Mosquitto broker is then, enter the following command:

```
sudo apt install -y mosquitto mosquitto-clients
```

**Linux Tech note:** **apt install** is used to install software, where that software exists as a package configured and pre-built for your specific OS. Packages are not always available and would require building from source which can be complex and time consuming. Packages have done the hard work for you.

## 1.2 Setting Up the Mosquitto MQTT Broker

Having successfully installed the Mosquitto broker package, we will now set up some additional 'quality of life' features.

The first is to make Mosquitto automatically start up when the R-Pi boots by running the following command:

```
sudo systemctl enable mosquitto.service
```

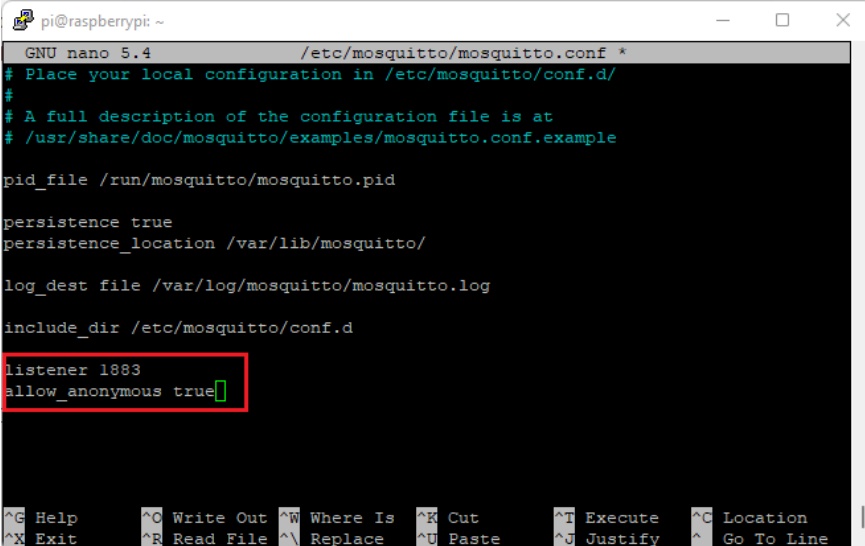
Secondly, we will enable remote access so that we can communicate with other IoT devices. Run the following command in order to open the 'mosquitto.conf' configuration file:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

Move to the end of the file using the keyboard arrow keys and add the following two lines:

```
listener 1883  
allow_anonymous true
```

Your file should now look as shown in Figure 2:



```
pi@raspberrypi: ~  
GNU nano 5.4 /etc/mosquitto/mosquitto.conf *  
# Place your local configuration in /etc/mosquitto/conf.d/  
#  
# A full description of the configuration file is at  
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example  
  
pid_file /run/mosquitto/mosquitto.pid  
  
persistence true  
persistence_location /var/lib/mosquitto/  
  
log_dest file /var/log/mosquitto/mosquitto.log  
  
include_dir /etc/mosquitto/conf.d  
  
listener 1883  
allow_anonymous true  
  
^G Help      ^O Write Out ^W Where Is  ^K Cut      ^T Execute  ^C Location  
^X Exit      ^R Read File ^\ Replace   ^U Paste    ^J Justify  ^_ Go To Line
```

Figure 2: Modified 'mosquitto.conf' Configuration File

Press 'Ctrl-X' to exit the file and then press 'Y' and 'Enter' to save the file.

Finally, in order to use the Mosquitto broker for this session, you will need to know your group's R-Pi IP address and one way is by typing the following command in the terminal window:

**hostname -I**

Once you have clicked 'Enter', your IP address should be shown on the next line, as shown in Figure 3. Note that your IP address will vary from that shown below and from other groups, this is expected as the IP address is unique to each R-Pi. Also the IP address is assigned dynamically and so will change following a reboot:

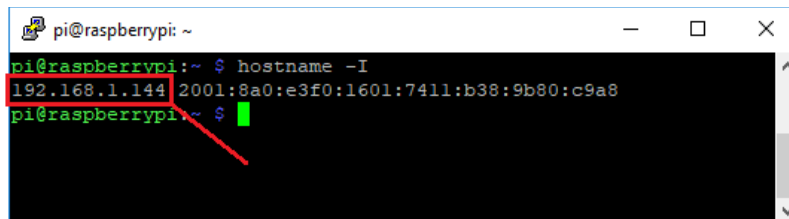
A terminal window titled 'pi@raspberrypi: ~' showing the command 'hostname -I' being executed. The output is '192.168.1.144 2001:8a0:e3f0:1601:7411:b38:9b80:c9a8'. A red box highlights the IP address '192.168.1.144' and a red arrow points to it from the bottom right.

Figure 3: Example R-Pi IP Address

### 1.3 Testing the Mosquitto MQTT Broker Installation

Now that we have set up Mosquitto, we will now perform a simple test to check that the installation and set-up were successful.

First, reboot your R-Pi for the previous changes to take effect, using the following command:

**sudo reboot**

Test the installation by running the following command:

**mosquitto -v**

Although this is not a complete test, if done correctly this command returns the Mosquitto version that is currently running on your Pi. Note this version may vary depending on the time of installation but it will be version 2.0.11 or above – as shown in Figure 4. Ignore any errors below this line.

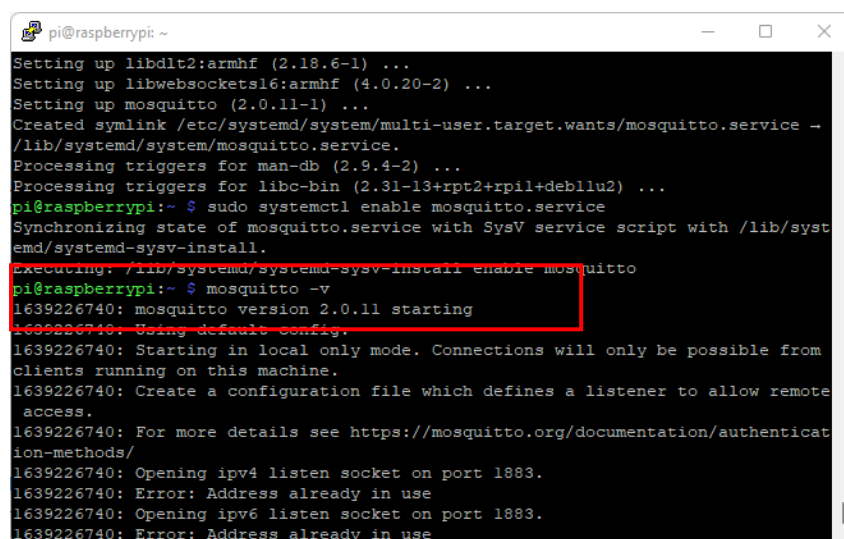
A terminal window titled 'pi@raspberrypi: ~' showing the output of the 'mosquitto -v' command. The output includes system messages about installing libdlt2, libwebsockets, and mosquitto, followed by 'pi@raspberrypi:~ \$ sudo systemctl enable mosquitto.service', 'Synchronizing state of mosquitto.service with SysV service script with /lib/systemd/systemd-sysv-install.', and 'Executing: /lib/systemd/systemd-sysv-install enable mosquitto'. The command 'pi@raspberrypi:~ \$ mosquitto -v' is executed, resulting in '1639226740: mosquitto version 2.0.11 starting' and '1639226740: Using default config.'. Below this, there are messages about starting in local only mode and opening listen sockets on port 1883, with errors for 'Address already in use'. A red box highlights the line '1639226740: mosquitto version 2.0.11 starting'.

Figure 4: Example Command Prompt Showing a Successful Mosquitto Installation



If your installation was not successful and your command prompt does not resemble that shown above, carefully work through all of the previous steps again before asking for assistance.

### 1.4 Installing Node-RED

Node-RED is a powerful open source tool for building Internet of Things (IoT) applications with the goal of simplifying the programming component. It uses a visual programming approach that allows you to connect code blocks, known as nodes, together to perform a task. These nodes, when wired together, are called flows.

To install Node-RED, enter the following command – note this may take several minutes:

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
```

To automatically run Node-RED when the R-Pi boots up, enter the following command:

```
sudo systemctl enable nodered.service
```

### 1.5 Installing the Node-RED Dashboard

Node-RED Dashboard is a module that provides a set of nodes in Node-RED to quickly create a live data dashboard.

To install the Node-RED dashboard, run the following commands:

```
node-red-stop  
cd ~/.node-red  
npm install node-red-dashboard
```

In order for the Node-RED and Node-RED dashboard changes to take effect, reboot your Pi using:

```
sudo reboot
```

### 1.6 Testing the Node-RED Installations

In order to check your Node-RED installations, we will use your R-Pi IP address. When your R-Pi is back on, you can first test the Node-RED installation by entering the IP address of your Pi in a web browser followed by the 1880 port number:

```
http://YOUR_RPi_IP_ADDRESS:1880
```

For this example, this would be:

```
http://192.168.1.98:1880
```

When clicking 'Enter', a page as shown in Figure 5 will appear. Note this will be blank as we have not yet added any nodes.

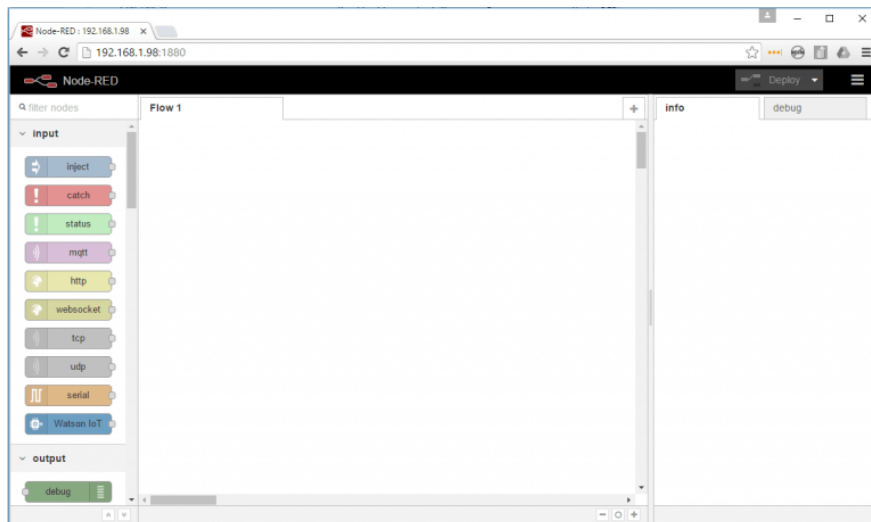


Figure 5: Blank Node-RED Webpage

To open the Node-RED UI, type your R-Pi IP address in a web browser followed by ':1880/ui' as shown below:

**`http://Your_RPi_IP_address:1880/ui`**

When clicking 'Enter, a page as shown in Figure 6 will appear. Again, this will be blank as we haven't added anything to the dashboard yet.

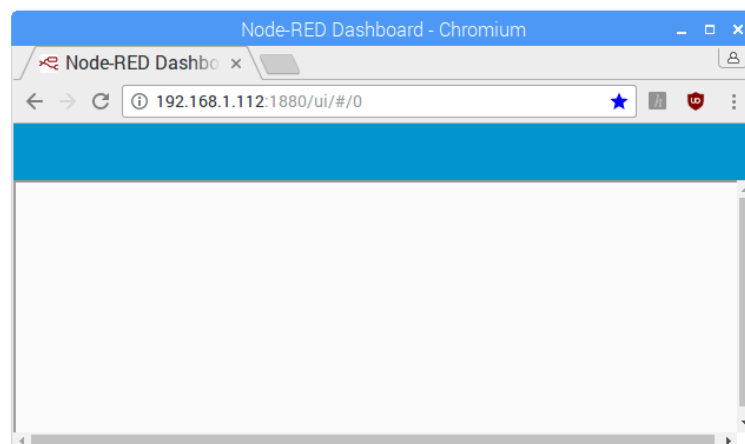


Figure 6: Blank Node-RED Dashboard



## 2. ESP32 MQTT Example – Publish & Subscribe with the Arduino IDE

Now you have successfully set up your R-Pi and installed the Mosquitto MQTT broker, you will now be presented with a guide to a simple example. Note that this example uses a different sensor to those available on the EEEBot, so **you will need to adapt the example code** to accommodate your sensors. **This is a key engineering skill as, typically, information online will not entirely match your specific application, and so the key appropriate information and/or code snippets need to be identified and used.**

By way of example, the following project shows how to use the MQTT communication protocol with the ESP32 to publish messages and subscribe to topics. The example described below will publish BME280 sensor readings to the Node-RED Dashboard, and control an ESP32 output. The ESP32 will be programmed using Arduino IDE. For further information, see:

<https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/>

The following figure shows an overview of the example described in this tutorial:

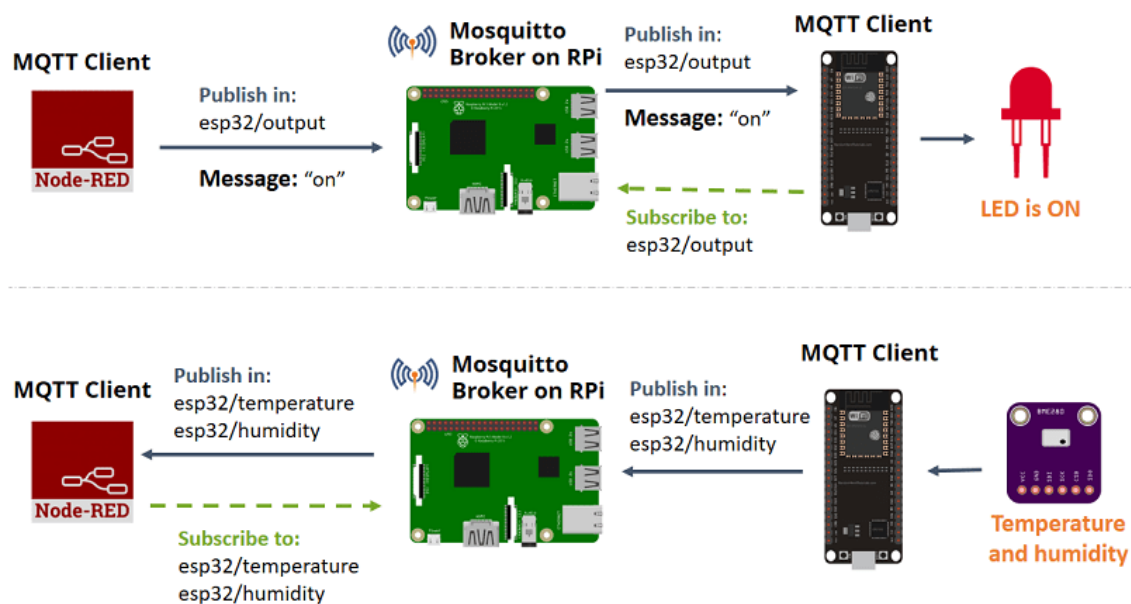


Figure 7: ESP32 MQTT Example Project Overview

The Node-RED application publishes messages (“on” or “off”) in the topic esp32/output. The ESP32 is subscribed to that topic. So, it receives the message with “on” or “off” to turn the LED on or off. The ESP32 publishes temperature on the esp32/temperature topic and the humidity on the esp32/humidity topic. The Node-RED application is subscribed to those topics. So, it receives temperature and humidity readings that can be displayed on a chart or gauge, for example.

### 2.1 Constructing the Circuit

In this example, the chosen sensor to send data to the MQTT, and therefore to the Node-RED dashboard, is the BME280 sensor module. This measures temperature, humidity and pressure – although in this tutorial only the temperature and humidity readings will be used. Further research can be performed into this sensor module, but the following connections were made using I2C, as per the wiring schematic and the realised circuit as shown in Figure 8. Note we will also make use of the ‘subscribe’ functionality by controlling an ESP32 output i.e. the LED connected to GPIO 4.

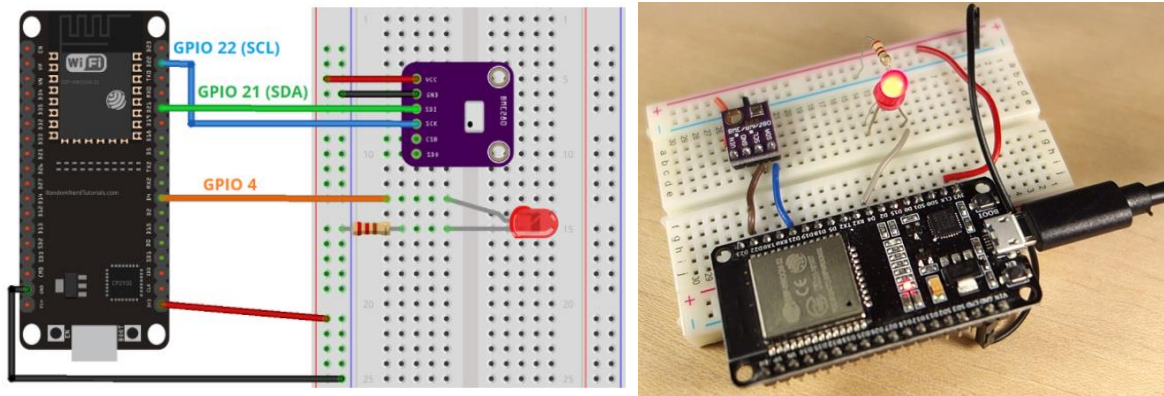


Figure 8: ESP32 MQTT Example Wiring Schematic & ESP32 MQTT Example Realised Circuit

## 2.2 Setting Up the Arduino IDE

Ensure that you have installed the ESP32 board into the Arduino IDE. If you have not done this already, refer back to the process detailed at the end of 'EEEE1002 S01 E03 The EEEBot' in 'Session 1: Test & Measurement and the EEEBot' on Moodle.

### Installing the PubSubClient Library

The PubSubClient library provides a client for doing simple publish/subscribe messaging with a server that supports MQTT i.e. this allows your ESP32 to talk with Node-RED. In order to install the PubSubClient library, follow the below steps:

- 1 Download the PubSubClient library from <https://github.com/knolleary/pubsubclient/archive/master.zip>. You should have a .zip folder in your Downloads folder.
- 2 *Optional:* Rename your folder from 'pubsubclient-master' to 'pubsubclient'.
- 3 Include the library by clicking 'Sketch → Include Library → Add .ZIP Library...' inside the Arduino IDE and select the downloaded .zip file.

The library comes with a number of example sketches. See 'File → Examples → PubSubClient' within the Arduino IDE software.

Note: PubSubClient is not fully compatible with the ESP32, but the example provided in this tutorial is working very reliably during our tests.

*As mentioned earlier, your EEEBot does not contain the BME280 sensor. For completeness, the details of installing the appropriate libraries are detailed but replace the following sections with the installation of the relevant libraries relating to the sensors/components on the EEEBot that you want to use. Googling '\*SENSOR NAME\* Arduino/ESP32 library' is typically a good starting point.*

### Installing the BME280 Library

To take readings from the BME280 sensor module we'll use the Adafruit\_BME280 library. Follow the next steps to install the library in your Arduino IDE:

- 1 Download the Adafruit-BME280 library from [https://github.com/adafruit/Adafruit\\_BME280\\_Library/archive/master.zip](https://github.com/adafruit/Adafruit_BME280_Library/archive/master.zip). You should have a .zip folder in your Downloads folder.

- 2 *Optional:* Rename your folder from 'Adafruit-BME280-Library-master' to 'Adafruit\_BME280\_Library'.
- 3 Include the library by clicking 'Sketch → Include Library → Add .ZIP Library...' inside the Arduino IDE and select the downloaded .zip file.

### *Installing the Adafruit\_Sensor Library*

To use the BME280 library, you also need to install the Adafruit\_Sensor library. Follow the next steps to install the library:

- 1 Download the Adafruit\_Sensor library from [https://github.com/adafruit/Adafruit\\_Sensor/archive/master.zip](https://github.com/adafruit/Adafruit_Sensor/archive/master.zip). You should have a .zip folder in your Downloads folder.
- 2 *Optional:* Rename your folder from 'Adafruit\_Sensor-master' to 'Adafruit\_Sensor'.
- 3 Include the library by clicking 'Sketch → Include Library → Add .ZIP Library...' inside the Arduino IDE and select the downloaded .zip file.

## 2.3 ESP32 Code Breakdown

Once you have installed all of the relevant libraries, we will program the ESP32 using the code provided on Moodle. Note that two codes are provided on Moodle as zip files, 'ESP32\_MQTT\_Example\_Code' that contains complete code using the BME280 sensor, and 'ESP32\_MQTT\_EEBot\_Template' that contains the necessary **blank** publish/subscribe functions i.e. the skeleton code where you will need to add your own sensor/component related code. You may find it useful to use this code for Session 4 although this is not mandatory.

To complete this tutorial, extract and upload 'ESP32\_MQTT\_Example\_Code' to the ESP32 board. This code publishes temperature and humidity readings on the 'esp32/temperature' and 'esp32/humidity' topics through the MQTT protocol. The ESP32 is subscribed to the 'esp32/output' topic to receive the messages published on that topic by the Node-RED application. Then, according to the received message, it turns the LED on or off. Below, key code snippets from the code are described and explained.

### *Subscribing to MQTT Topics*

In the 'reconnect()' function, you can subscribe to MQTT topics. In this case, the ESP32 is only subscribed to the 'esp32/output':

```
client.subscribe("esp32/output");
```

In the callback() function, the ESP32 receives the MQTT messages of the subscribed topics. According to the MQTT topic and message, it turns the LED on or off:

```
if (String(topic) == "esp32/output") {  
  ...  
  if(messageTemp == "on"){  
    ...  
    digitalWrite(ledPin, HIGH);  
  }  
  else if(messageTemp == "off"){  
    ...  
    digitalWrite(ledPin, LOW);  
  }...}
```

### Publishing MQTT Messages

In the loop(), new readings are being published every 5 seconds:

```
if (now - lastMsg > 5000) {...}
```

By default, the ESP32 is sending the temperature in Celsius, but you can uncomment the last line to send the temperature in Fahrenheit:

```
temperature = bme.readTemperature();  
//temperature = 1.8 * bme.readTemperature() + 32;
```

You need to convert the temperature float variable to a char array, so that you can publish the temperature reading in the 'esp32/temperature' topic:

```
char tempString[8];  
dtostrf(temperature, 1, 2, tempString);  
Serial.print("Temperature: ");  
Serial.println(tempString);  
client.publish("esp32/temperature", tempString);
```

The same process is repeated to publish the humidity reading in the 'esp32/humidity' topic:

```
char humString[8];  
dtostrf(humidity, 1, 2, humString);  
Serial.print("Humidity: ");  
Serial.println(humString);  
client.publish("esp32/humidity", humString);
```

### Creating the Node-RED Flow

Once you have uploaded the above sketch to your ESP32, we will now create the Node-RED flows that display the BME280 sensor readings. Typically this will be done from scratch, using the GUI elements available from Node-RED. However, for this tutorial, import prewritten Node-RED flow available from:

[https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/master/Projects/ESP32-MQTT/Node RED Flow ESP32 MQTT Publish Subscribe.txt](https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/master/Projects/ESP32-MQTT/Node%20RED%20Flow%20ESP32%20MQTT%20Publish%20Subscribe.txt)

Open the Node-RED window, by entering the following into your browser:

`http://Your_RPi_IP_address:1880`

In the top right corner, select the menu and click 'Import → Clipboard' as shown below in Figure 9.

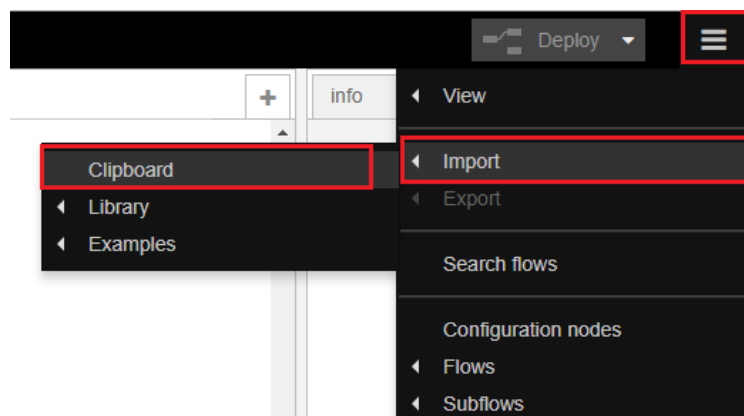


Figure 9: Import Menu Inside the Node-RED Window

Then copy and paste the code provided and click 'Import' – as per Figure 10 below.

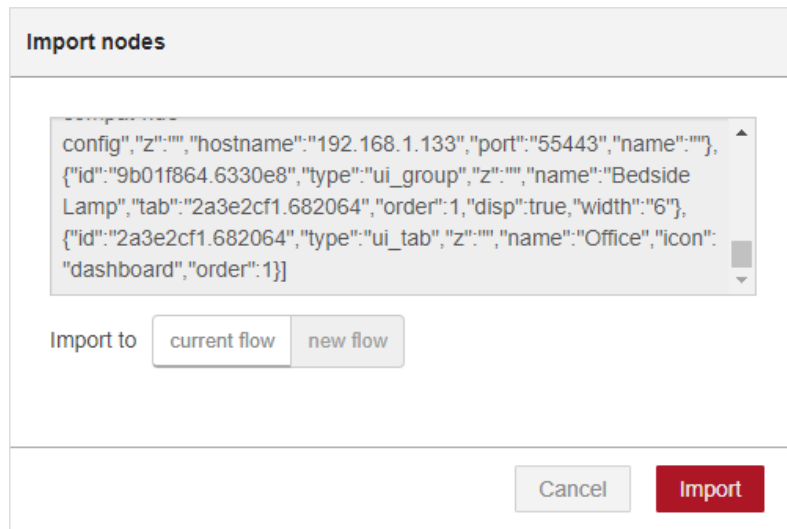


Figure 10: Import Dialogue Box

The following nodes should then load in the Node-RED window:

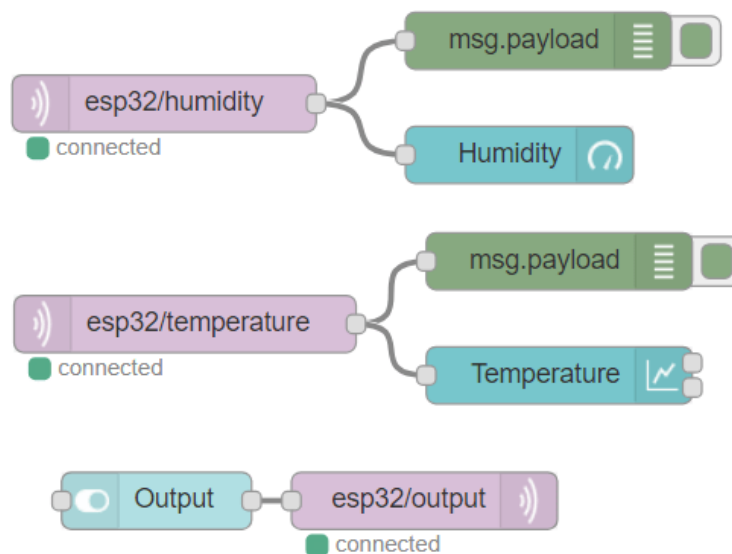


Figure 11: Loaded Node-RED Nodes

After adding, removing or modifying the nodes, click the 'Deploy' button, as shown in Figure 12, to save all changes.

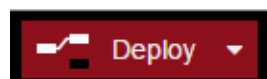


Figure 12: Deploy Button

### Node-RED UI

Now, your Node-RED application is ready. To access the Node-RED UI and see how your application looks, access any browser in your local network and type:

**http://Your\_RPi\_IP\_address:1880/ui**

Your application should look as shown below in Figure 13. You can control the LED on and off with the switch or you can view temperature readings in a chart and the humidity values in a gauge.

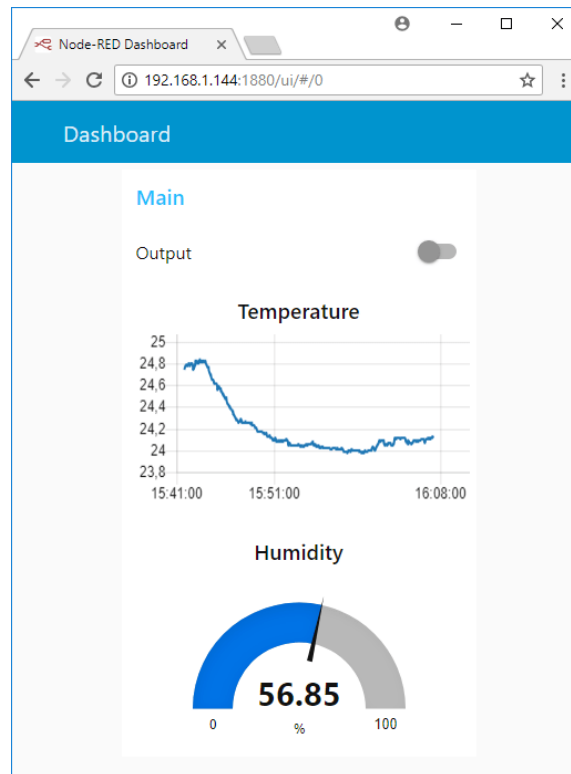


Figure 13: Example Node-RED Dashboard

We can also view the MQTT messages being received and published by opening the serial monitor in the Arduino IDE. This should resemble Figure 14.

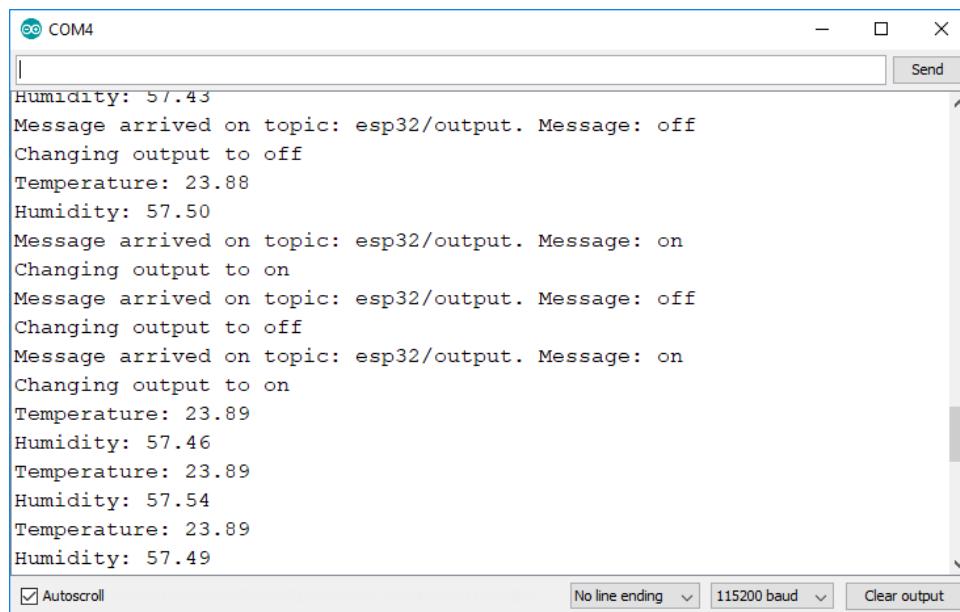


Figure 14: Example Serial Monitor Displaying the MQTT Messages

### 3. Design Task – Vehicle to Anything (V2X) Communication

In summary, the simple tutorial you have worked through has shown you the basic concepts that allow you to turn on lights and monitor sensors with your ESP32 using Node-RED and the MQTT communication protocol. You can use this example to integrate into your own home automation system, control more outputs, or monitor other sensors.

For this session, you must work as a group so that each EEEBot in your group sends and receives data to/from one central Node-RED page using one R-Pi 3A+ running the Mosquitto MQTT broker. Each member of the group will be responsible for:

- equipping their own vehicle with a sensor that can send data to the central controller to be displayed via a GUI. Each vehicle in the group must use a different sensor.
- equipping their vehicle with a device that can be controlled from the central controller's GUI. The device can be the same for all vehicles in the group.

Completion of the above is worth 70% of Session 4. In addition to this, in order to obtain a further 30% if at the end of the session the group presents a single vehicle equipped with a number of sensors and is capable of line following that sends/receives data to/from the central controller that can:

- display status of each IR sensor.
- display the weighted average.
- vary the PID constants.

Note that each of the above is worth 10% for Lab Report 2.

You may also find it useful to refer to 'Suggested MQTT Tasks.pdf' on Moodle for example publish and subscribe topics to implement. There will be a higher amount of marks available for Poster 2 with the more data published and subscribed to/from your group's EEEBots using Node-RED.