**ESP8266 and Node-RED with MQTT (Publish and Subscribe)**

In this post we’re going to show you how to control ESP8266 outputs and display sensor data from the ESP8266 on Node-RED. The Node-RED software is running on a Raspberry Pi, and the communication between the ESP8266 and the Node-RED software is achieved with the MQTT communication protocol.

The following figure shows an overview of what we’re going to do in this tutorial.

[Graphical user interface, application

Description automatically generated](https://i1.wp.com/randomnerdtutorials.com/wp-content/uploads/2017/08/MQTT-ESP8266-publish-and-subscribe-Node-RED.png?quality=100&strip=all&ssl=1)

**Node-RED and Node-RED Dashboard**

You need to have Node-RED and Node-RED Dashboard installed in your [Raspberry Pi](https://makeradvisor.com/raspberry-pi-board/). The following blog posts are useful for getting started with Node-RED and Node-RED dashboard:

* [Getting started with Node-RED on Raspberry Pi](https://randomnerdtutorials.com/getting-started-with-node-red-on-raspberry-pi/)
* [Getting Started with Node-RED Dashboard](https://randomnerdtutorials.com/getting-started-with-node-red-dashboard/)

**MQTT Protocol**

In this tutorial we’re going to establish a communication between a Raspberry Pi running the Node-RED software and an ESP8266 using MQTT.

MQTT stands for **MQ T**elemetry **T**ransport and it is a nice lightweight publish and subscribe system where you can publish and receive messages as a client. It is a simple messaging protocol, designed for constrained devices and with low-bandwidth. So, it’s the perfect solution for Internet of Things applications.

If you want to learn more about MQTT, watch the video below.

For a written version of this video and additional resources, read this blog post [What is MQTT and How It Works](https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/).

**Installing Mosquitto Broker**

In MQTT, the broker is primarily responsible for **receiving** all messages, **filtering** the messages, **decide** who is interested in it and then **publishing** the message to all subscribed clients.

There are several brokers you can use. In this tutorial we’re going to use the **Mosquitto Broker** which needs to be installed on Raspberry Pi.

To install the Mosquitto broker on Raspberry Pi follow this tutorial: [How to Install Mosquitto Broker on Raspberry Pi](https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/).

**Testing**

To see if Mosquitto broker was successfully installed, run the next command:

pi@raspberry:~ $ **mosquitto -v**

This returns the Mosquitto version that is currently running in your Raspberry Pi. It should be 1.4 or above.

Text

Description automatically generated

**Note:** the Mosquitto command returns the Mosquitto version that is currently installed, but it also tries to initialize Mosquitto again. Since Mosquitto is already running it prompts an error message. Don’t worry Mosquitto is properly installed and running if you see a similar message.

**Establishing an MQTT communication with Node-RED**

In this section we’re going to establish an MQTT communication using the Node-RED nodes.

**Dashboard Layout**

The first step is to create the dashboard layout. In this example, we’ll have a button to control an ESP8266 output; a chart and a gauge to display temperature and humidity readings from the DHT11 sensor.

On the top right corner of the Node-RED window, select the **Layout** tab under the **dashboard**tab. Create a tab called **Room** and inside the Room tab, create two groups:**Lamp** and **Sensor** as shown in figure below.

Graphical user interface, text, application

Description automatically generated

**Creating the Flow**

Drag the following nodes to the flow – see figure below:

Text, chat or text message

Description automatically generated with medium confidence

* **switch** – this will control the ESP8266 output
* **mqtt output node** – this will publish a message to the ESP8266 accordingly to the switch state
* **2x** **mqtt input nodes** – this nodes will be subscribed to the temperature and humidity topics to receive sensor data from the ESP
* **chart** – will display the temperature sensor readings
* **gauge** – will display the humidity sensor readings

Node-RED and the MQTT broker need to be connected. To connect the MQTT broker to Node-REd, double-click the MQTT output node. A new window pops up – as shown in figure below.

Graphical user interface, text, application, email

Description automatically generated

1. Click the **Add new mqtt-broker** option.
2. Type **localhost** in the server fieldGraphical user interface, text, application

   Description automatically generated
3. All the other settings are configured properly by default.
4. Press **Add** and the MQTT output node automatically connects to your broker.

Edit all the other nodes properties as shown in the following figures:

* **switch –**the switch sends an **on** string messagewhen it’s on; and sends an **off**string message when it’s off. This node will publish on the **room/lamp** topic. Your ESP will then be subscribed to this topic, to receive its messages
* Graphical user interface, application, email

  Description automatically generated
* **mqtt output node**. This node is connected to the mosquitto broker and it will publish in the **room/lamp** topic.
* Graphical user interface, text, application, email

  Description automatically generated
* **mqtt input node**. This node is subscribed to the **room/temperature** topic to receive temperature sensor data from the ESP8266. The ESP8266 will be pusblishing the temperature readings on this topic.

Graphical user interface, text, application

Description automatically generated

* **chart.**The chart will display the readings received on the r**oom/temperature** topic
* .Graphical user interface

  Description automatically generated
* **mqtt input node.**This node is subscribed to the **room/humidity** topic to receive humidity sensor data from the ESP8266. The ESP8266 will be pusblishing the humidity readings on this same topic.
* Graphical user interface, text, application, email

  Description automatically generated
* **gauge.**The gauge will display the readings received on the **room/humidity** topic.

Graphical user interface, website

Description automatically generated

Wire your nodes as shown in the figure below.

Diagram

Description automatically generated

Your Node-RED application is ready. Click the **Deploy** button on the top right corner.



The Node-RED application is ready. To see how your dashboard looks go to  ***http://your-pi-ip-address/ui***.

Now, follow the next sections to prepare your ESP8266.

**Preparing your Arduino IDE**

We’ll program the ESP8266 using the Arduino IDE. In order to upload code to your ESP8266 using the Arduino IDE, you need to install the ESP8266 add-on ([How to Install the ESP8266 Board in Arduino IDE](https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/)). You’ll also need to install two additional libraries to have everything ready for your ESP8266.

**Installing the PubSubClient Library**

The [PubSubClient](https://github.com/knolleary/pubsubclient" \t "_blank) library provides a client for doing simple publish/subscribe messaging with a server that supports MQTT (basically allows your ESP8266 to talk with Node-RED).

**1)** [Click here to download the PubSubClient library](https://github.com/knolleary/pubsubclient/archive/master.zip). You should have a *.zip* folder in your Downloads folder

**2)** Unzip the *.zip* folder and you should get **pubsubclient-master** folder

**3)** Rename your folder from to **pubsubclient**

**4)** Move the **pubsubclient** folder to your Arduino IDE installation **libraries** folder

**5)** Then, re-open your Arduino IDE

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

**Installing the DHT Sensor Library**

The [DHT sensor library](https://github.com/adafruit/DHT-sensor-library) provides an easy way of using any DHT sensor to read temperature and humidity with your ESP8266 or Arduino boards.

**1)** [Click here to download the DHT sensor library](https://github.com/adafruit/DHT-sensor-library/archive/master.zip). You should have a *.zip* folder in your Downloads

**2)** Unzip the *.zip* folder and you should get **DHT-sensor-library-master**folder

**3)** Rename your folder from  to**DHT**

**4)** Move the **DHT**folder to your Arduino IDE installation **libraries**folder

**5)** Then re-open your Arduino IDE

For more information about the DHT11 sensor and the ESP8266, read [ESP8266 DHT11/DHT22 Temperature and Humidity Web Server with Arduino IDE](https://randomnerdtutorials.com/esp8266-dht11dht22-temperature-and-humidity-web-server-with-arduino-ide/).

**Selecting the right board on Arduino IDE**

You also need to select the right board on Arduino IDE:

**1)** Go to Tools and select “NodeMCU 1.0 (ESP-12E Module)”.

**2)** Select your ESP port number under the Tools > Port > COM4 (in my case)

**Uploading code**

Now, you can upload the following code to your ESP8266. This code publishes messages of the temperature and humidity from the DHT11 sensor on the **room/temperature** and **room/humidity** topics trough MQTT protocol.

The ESP is subscribed to the **room/lamp** topic to receive the messages published on that topic by the Node-RED application, to turn the lamp on or off.

The code is well commented on where you need to make changes. **You need to edit the code with your own SSID, password and RPi IP address.**

This code is also compatible with other DHT sensors – you just need to uncomment and comment the right lines of code to chose your sensor.

/\*\*\*\*\*

All the resources for this project:

https://randomnerdtutorials.com/

\*\*\*\*\*/

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include "DHT.h"

// Uncomment which DHT sensor you are using

#define DHTTYPE DHT11 // DHT 11

//#define DHTTYPE DHT21 // DHT 21 (AM2301)

//#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

// Change

const char\* ssid = "SSID";

const char\* password = "PASSWORD";

const char\* mqtt\_server = "RPI\_IP\_ADDRESS";

// Initializes the espClient. You should change the espClient name if you have multiple ESPs running in your home automation system

WiFiClient espClient;

PubSubClient client(espClient);

// DHT Sensor - GPIO 5 = D1 on ESP-12E NodeMCU board

const int DHTPin = 5;

// Lamp - LED - GPIO 4 = D2 on ESP-12E NodeMCU board

const int lamp = 4;

// Initialize DHT sensor.

DHT dht(DHTPin, DHTTYPE);

// Timers auxiliar variables

long now = millis();

long lastMeasure = 0;

// Connects ESP8266 to your router

void setup\_wifi() {

delay(10);

// We start by connecting to a WiFi network

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("WiFi connected - ESP IP address: ");

Serial.println(WiFi.localIP());

}

// This functions is executed when some device publishes a message to a topic that your ESP8266 is subscribed to

// Change the function below to add logic to your program, so when a device publishes a message to a topic that

// your ESP8266 is subscribed you can actually do something

void callback(String topic, byte\* message, unsigned int length) {

Serial.print("Message arrived on topic: ");

Serial.print(topic);

Serial.print(". Message: ");

String messageTemp;

for (int i = 0; i < length; i++) {

Serial.print((char)message[i]);

messageTemp += (char)message[i];

}

Serial.println();

// Feel free to add more if statements to control more GPIOs with MQTT

// If a message is received on the topic room/lamp, you check if the message is either on or off. Turns the lamp GPIO according to the message

if(topic=="room/lamp"){

Serial.print("Changing Room lamp to ");

if(messageTemp == "on"){

digitalWrite(lamp, HIGH);

Serial.print("On");

}

else if(messageTemp == "off"){

digitalWrite(lamp, LOW);

Serial.print("Off");

}

}

Serial.println();

}

// This functions reconnects your ESP8266 to your MQTT broker

// Change the function below if you want to subscribe to more topics with your ESP8266

void reconnect() {

// Loop until we're reconnected

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

// Attempt to connect

/\*

YOU MIGHT NEED TO CHANGE THIS LINE, IF YOU'RE HAVING PROBLEMS WITH MQTT MULTIPLE CONNECTIONS

To change the ESP device ID, you will have to give a new name to the ESP8266.

Here's how it looks:

if (client.connect("ESP8266Client")) {

You can do it like this:

if (client.connect("ESP1\_Office")) {

Then, for the other ESP:

if (client.connect("ESP2\_Garage")) {

That should solve your MQTT multiple connections problem

\*/

if (client.connect("ESP8266Client")) {

Serial.println("connected");

// Subscribe or resubscribe to a topic

// You can subscribe to more topics (to control more LEDs in this example)

client.subscribe("room/lamp");

} else {

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Wait 5 seconds before retrying

delay(5000);

}

}

}

// The setup function sets your ESP GPIOs to Outputs, starts the serial communication at a baud rate of 115200

// Sets your mqtt broker and sets the callback function

// The callback function is what receives messages and actually controls the LEDs

void setup() {

pinMode(lamp, OUTPUT);

dht.begin();

Serial.begin(115200);

setup\_wifi();

client.setServer(mqtt\_server, 1883);

client.setCallback(callback);

}

// For this project, you don't need to change anything in the loop function. Basically it ensures that you ESP is connected to your broker

void loop() {

if (!client.connected()) {

reconnect();

}

if(!client.loop())

client.connect("ESP8266Client");

now = millis();

// Publishes new temperature and humidity every 30 seconds

if (now - lastMeasure > 30000) {

lastMeasure = now;

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

float h = dht.readHumidity();

// Read temperature as Celsius (the default)

float t = dht.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

float f = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).

if (isnan(h) || isnan(t) || isnan(f)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

// Computes temperature values in Celsius

float hic = dht.computeHeatIndex(t, h, false);

static char temperatureTemp[7];

dtostrf(hic, 6, 2, temperatureTemp);

// Uncomment to compute temperature values in Fahrenheit

// float hif = dht.computeHeatIndex(f, h);

// static char temperatureTemp[7];

// dtostrf(hif, 6, 2, temperatureTemp);

static char humidityTemp[7];

dtostrf(h, 6, 2, humidityTemp);

// Publishes Temperature and Humidity values

client.publish("room/temperature", temperatureTemp);

client.publish("room/humidity", humidityTemp);

Serial.print("Humidity: ");

Serial.print(h);

Serial.print(" %\t Temperature: ");

Serial.print(t);

Serial.print(" \*C ");

Serial.print(f);

Serial.print(" \*F\t Heat index: ");

Serial.print(hic);

Serial.println(" \*C ");

// Serial.print(hif);

// Serial.println(" \*F");

}

}

After uploading the code, and with the Raspberry Pi running your Node-RED application and the Mosquitto broker, you can open the Arduino IDE serial monitor at a baud rate of 115200 and see what’s happening in real time.

This is helpful to check if the ESP has established a successful connection to your router and to the Mosquitto broker. You can also see the messages the ESP is receiving and publishing.

Graphical user interface, text, application

Description automatically generated

**Building the Circuit**

The following sections show you the needed parts and schematics to build the circuit for this project.

* [Raspberry Pi](https://makeradvisor.com/tools/raspberry-pi-board/) – [read Best Raspberry Pi 3 Starter Kits](https://makeradvisor.com/best-raspberry-pi-3-starter-kits/)
* [ESP8266 (ESP-12E nodemcu)](https://makeradvisor.com/tools/esp8266-esp-12e-nodemcu-wi-fi-development-board/) – [read Best ESP8266 Wi-Fi Development Boards](https://makeradvisor.com/best-esp8266-wi-fi-development-board/)
* [DHT11 temperature and humidity sensor](https://makeradvisor.com/tools/dht11-temperature-humidity-sensor/)
* [Breadboard](https://makeradvisor.com/tools/mb-102-solderless-breadboard-830-points/)
* [330 Ω resistor](https://makeradvisor.com/tools/resistors-kits/)
* [LED](https://makeradvisor.com/tools/3mm-5mm-leds-kit-storage-box/)
* [4700 Ω resistor](https://makeradvisor.com/tools/resistors-kits/)

.

A screenshot of a computer

Description automatically generated with medium confidence

**Demonstration**

Go to ***http://your-pi-ip-address/ui***to control the ESP with the  Node-RED application.

The application should look something the figure below.

Graphical user interface

Description automatically generated

<https://www.youtube.com/watch?v=OFaow_TPv6s>

https://www.youtube.com/watch?v=7gjJfJgi6uE

Diagram

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated