Guida comunicazione SSL/TLS Raspberry – ESP8266 con MQTT

1. Installazione di Mosquitto Broker

|  |
| --- |
| Importare il repository e la rispettiva signing key |
| pi@raspberry:~ $ **wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key**  pi@raspberry:~ $ **sudo apt-key add mosquitto-repo.gpg.key** |
| Rendere il repository disponibile all’installazione con apt |
| pi@raspberry:~ $ **cd /etc/apt/sources.list.d/**  pi@raspberrypi:/etc/apt/sources.list.d $ **sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list** |
| Tornare alla root directory |
| pi@raspberrypi:/etc/apt/sources.list.d $ **cd** |
| Installare Mosquitto |
| pi@raspberry:~ $ **sudo apt-get update**  pi@raspberry:~ $ **sudo apt-get install mosquitto**  pi@raspberry:~ $ **sudo systemctl enable mosquitto.service** |
| Per verificare che l’installazione sia stata eseguita correttamente digitare pi@raspberry:~ $ **mosquitto -v .**  Nota: se Mosquitto ritorna un errore del tipo “Error: address already in use” non ci sono problemi. |

1. Installazione di Node-RED

|  |
| --- |
| Da terminale digitare il seguente comando |
| pi@raspberry:~ $ **bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)** |

1. Installazione di Node-RED Dashboard

|  |
| --- |
| Importare il repository e la rispettiva signing key |
| pi@raspberry:~ $ **wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key**  pi@raspberry:~ $ **sudo apt-key add mosquitto-repo.gpg.key** |
| Rendere il repository disponibile all’installazione con apt |
| pi@raspberry:~ $ **cd /etc/apt/sources.list.d/**  pi@raspberrypi:/etc/apt/sources.list.d $ **sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list** |
| Tornare alla root directory |
| pi@raspberrypi:/etc/apt/sources.list.d $ **cd** |
| Installare Mosquitto |
| pi@raspberry:~ $ **sudo apt-get update**  pi@raspberry:~ $ **sudo apt-get install mosquitto**  pi@raspberry:~ $ **sudo systemctl enable mosquitto.service** |
| Per verificare che l’installazione sia stata eseguita correttamente digitare pi@raspberry:~ $ **mosquitto -v .**  Nota: se Mosquitto ritorna un errore del tipo “Error: address already in use” non ci sono problemi. |

1. Creazione flow-chart in Node-RED per comunicazione MQTT con Node-RED

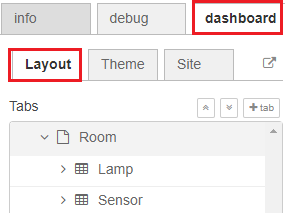
Establishing an MQTT communication with Node-RED

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

Dashboard Layout

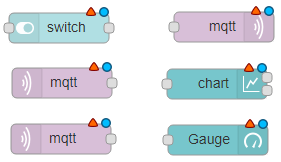
The first step is to create the dashboard layout. In this example, we’ll have a button to control an ESP8266 output; a chart and a gauge to display temperature and humidity readings from the DHT11 sensor.

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.



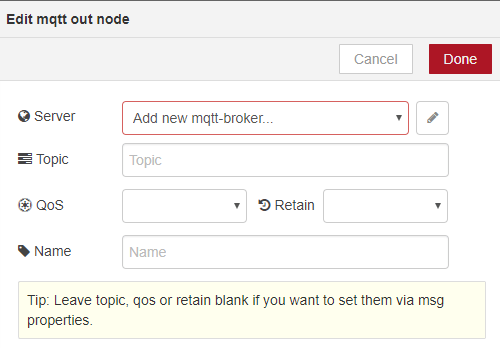
Creating the Flow

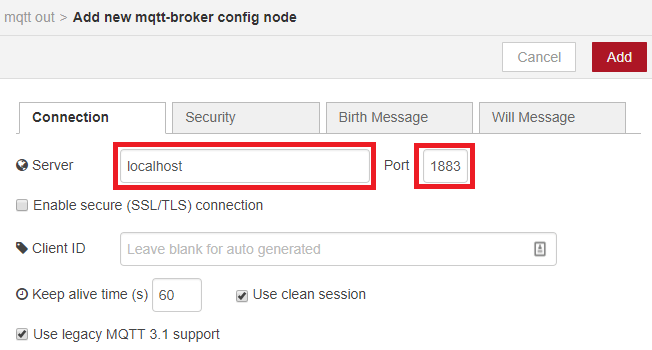
Drag the following nodes to the flow – see figure below:



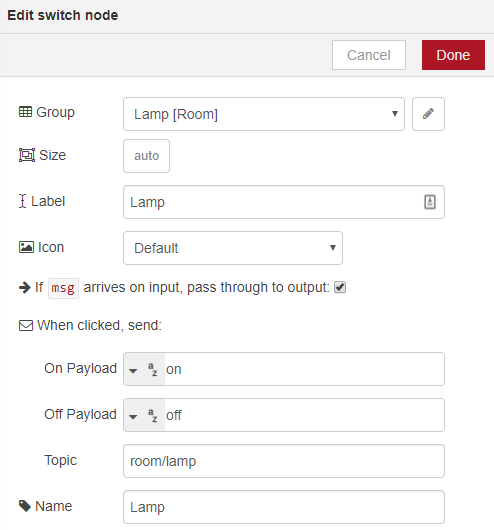
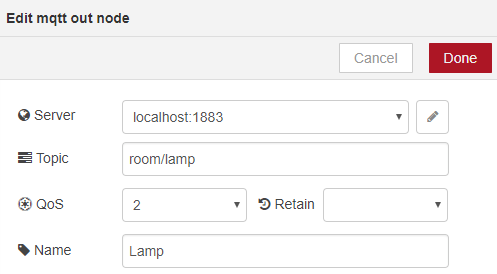
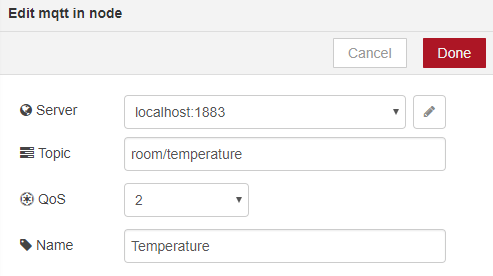
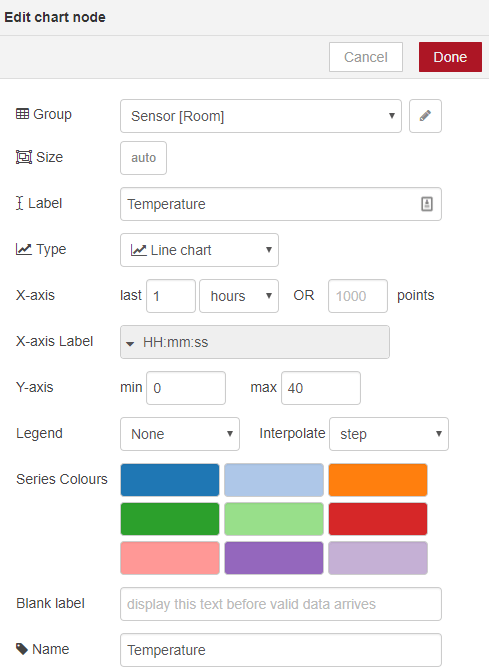
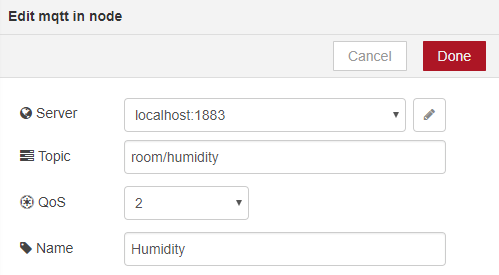
* **switch** – this will control the ESP8266 output
* **mqtt output node** – this will publish a message to the ESP8266 accordingly to the switch state
* **2x** **mqtt input nodes** – this nodes will be subscribed to the temperature and humidity topics to receive sensor data from the ESP
* **chart**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

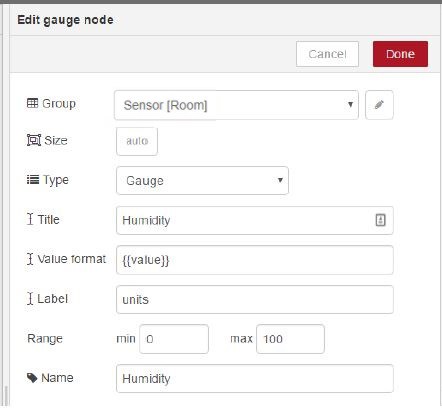
Node-RED and the MQTT broker need to be connected. To connect the MQTT broker to Node-REd, double-click the MQTT output node. A new window pops up – as shown in figure below.



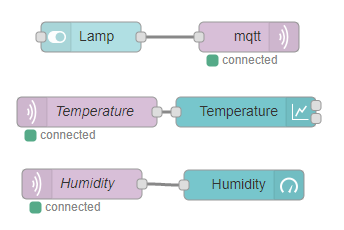
1. Click the **Add new mqtt-broker** option.
2. Type **localhost** in the server field
3. All the other settings are configured properly by default.
4. Press **Add** and the MQTT output node automatically connects to your broker.

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.
* **mqtt output node**. This node is connected to the mosquitto broker and it will publish in the **room/lamp** topic.
* **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.
* **chart.**The chart will display the readings received on the r**oom/temperature** topic.
* **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.
* **gauge.**The gauge will display the readings received on the **room/humidity** topic.



Wire your nodes as shown in the figure below.



Your Node-RED application is ready. Click the **Deploy** button on the top right corner.https://i2.wp.com/randomnerdtutorials.com/wp-content/uploads/2016/08/7-Deploy-Your-Application.png?resize=145%2C43&ssl=1

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](http://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) 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](http://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)

1. CODE

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include "DHT.h"

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

#define DHTPIN D2

const char\* ssid = "nome wifi";

const char\* password = "password wifi";

const char\* mqtt\_server = "192.168.1.43";

// 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);

const int DHTPin = DHTPIN;

// Initialize DHT sensor.

DHT dht(DHTPin, DHTTYPE);

// Timers auxiliar variables

long now = millis();

long lastMeasure = 0;

// Don't change the function below. This functions connects your 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());

}

// Riconnessione dell'ESP8266 al broker MQTT

void reconnect() {

// Loop fino a connessione eseguita

while (!client.connected()) {

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

// Attempt to connect

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

Serial.println("connected");

} 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

void setup() {

dht.begin();

Serial.begin(115200);

setup\_wifi();

client.setServer(mqtt\_server, 1883);

}

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 > 10000) {

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(false);

// Verifica letture. Se una lettura fallisce, si esce e si ritenta

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

Serial.println("Lettura fallita dal DHT22");

return;

}

// Computes temperature values in Celsius

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

static char temperatureTemp[7];

dtostrf(hic, 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.println(h);

Serial.print("Temperature: ");

Serial.print(t);

Serial.println("\*C");

Serial.print("Uploaded: ");

Serial.print(hic);

Serial.println("\*C");

}

}