**Getting Started with MQTT on Arduino Using NodeMCU ESP8266**

Written by Kudzai Manditereza

https://www.hivemq.com/blog/mqtt-on-arduino-nodemcu-esp8266-hivemq-cloud/

Category: [HiveMQ Cloud](https://www.hivemq.com/tags/hivemq-cloud) [IoT](https://www.hivemq.com/tags/iot)

Published: February 3, 2023

In this tutorial, I will show you how to use the Arduino IDE to program an ESP8266 microcontroller to read sensor data and publish it using MQTT to a HiveMQ Cloud MQTT broker.

Further, I’ll show you how to receive MQTT messages as commands from the cloud and activate the microcontroller output to switch ON and OFF a Light Emitting Diode (LED).

**What are ESP8266 NodeMCU Unit and DHT11 Sensor?**

Let us look briefly at the hardware components we will use for this demo, the NodeMCU Micontroller Unit, and the DHT11 Temperature and Humidity Sensor.

Diagram

Description automatically generatedBased on the ESP8266 WLAN chip, the NodeMCU is a small, fast, and Wi-Fi-capable microcontroller board. You can easily program it in the popular Arduino Integrated Development Environment (IDE). It has many Input/Output ports and various serial interfaces such as SPI, I2C, or RS232.

*NodeMCU Microcontroller Board and DHT11 Sensor*

The DHT11 is a sensor capable of measuring the relative humidity and temperature of the surrounding environment. It consists of three pins, two for power, and one is a signal line for transmitting the sensor data to the NodeMCU microcontroller.

**Hardware Wiring and Circuit Diagram**

Next, let’s look at how to wire the hardware components for this demonstration.

A picture containing text, electronics

Description automatically generated

The image above shows the setup for this demonstration, and the one below shows the visualization of the specific connection using a circuit diagram.

As shown, the connections are as follows:

* Vcc of the DHT11 goes to +3v of the NodeMCU,
* Signal pin of the DHT11 goes into the Digital Pin GPIO2 of the NodeMCC, and the
* A picture containing diagram

  Description automatically generatedGround pin of the DHT11 goes into the Ground Pin of the NodeMCU.
* A resistor connects the LED to the NodeMCU digital pin, GPIO5.

**Setting up a HiveMQ Cloud MQTT Broker**

Since I’ll be publishing MQTT messages from the NodeMCU board to an MQTT Broker, I must create the MQTT Broker on HiveMQ Cloud and retrieve the broker details to use in the Arduino code.

Graphical user interface, website

Description automatically generated

I’ll go to the [HiveMQ website](https://www.hivemq.com/) and select ‘*Cloud’* on the menu. I’ll get the [HiveMQ Cloud](https://www.hivemq.com/mqtt-cloud-broker/) page and then click the ‘*Try out for free’* button. If you’d like, you can [sign up for HiveMQ Cloud directly here](https://console.hivemq.cloud/?utm_source=MQTT+on+Arduino+NodeMCU+ESP8266+Blog&utm_medium=CTA+Button&utm_campaign=HiveMQ+Cloud&__hstc=184124345.da6b1dc05694d0cee1e417c536b5e2ce.1677057302006.1677057302006.1677057302006.1&__hssc=184124345.1.1677057302006&__hsfp=3034756077).

We are taken to a sign-up page and prompted to create a free account that allows us to connect up to 100 devices.

If you are accessing the HiveMQ Cloud portal for the first time, you’ll need to provide an email address and password and follow the simple steps to confirm your email and create your account.

Graphical user interface, application

Description automatically generated

An MQTT broker cluster will automatically be created when you’ve successfully created your account. Start by setting up credentials to allow devices to connect to your MQTT broker.

Graphical user interface, text, application

Description automatically generated

So, I’ll define the username and password and click on *ADD*.

Clicking on ‘*Clusters’* will provide the list of my MQTT broker clusters. Since I just created this account, I only have one. So, my MQTT broker setup is ready. Now, I need to get the cluster URL, port number, and access credentials for use in my Arduino code.

Graphical user interface, application

Description automatically generated

If I wish to update my connection credentials, I can always log in to my portal, click on ‘MANAGE CLUSTER,’ and select ‘ACCESS MANAGEMENT’.

Graphical user interface, text, application

Description automatically generated

**Installing and Configuring Arduino IDE**

**Downloading and Installing**

Next, we program the ESP8266 NodeMCU board using Arduino IDE. You can download Arduino IDE by going to the Software [downloads](https://www.arduino.cc/en/software) page of their website.

Graphical user interface, website

Description automatically generated

For this demonstration, I’ll download Arduino for Windows and install it.

**Enabling Support for the ESP8266 chip**

After installing the Arduino IDE, you can program various Arduino boards. However, to program a NodeMCU board instead of an Arduino board, you will need to add a package to the IDE’s board management system.

To do this, go to the ‘*File’* menu and select ‘*Preferences.’*

Then under the ‘*Additional Boards Manager’* field, I’ll enter this URL:

[*https://arduino.esp8266.com/stable/package\_esp8266com\_index.json*](https://arduino.esp8266.com/stable/package_esp8266com_index.json)*.*

*Graphical user interface, text, application

Description automatically generated*

Then click OK.

Next,  go to the ‘*Tools’* menu, and select ‘*Board’*, followed by ‘*Boards Manager’*.

Graphical user interface, application

Description automatically generated

Look for ESP8266, select it, and click on the Install button.

Graphical user interface, text, application

Description automatically generated

**Connecting the NodeMCU board to the Arduino IDE via USB**

Next, I need to specify the ESP8266 board I want to program in the Arduino IDE. So, I’ll connect my NodeMCU board to my PC using a micro USB cable with my Windows Device Manager open.

Graphical user interface, text, application, Word

Description automatically generated

As soon as I plug in my board, a new COM port appears on the Device Manager.

If you experience problems with your USB drivers, Windows Device Manager can help you troubleshoot. However, generally, it doesn’t need to be open. You can usually access the COM port by going to the *Port* menu item under tools in the Arduino IDE.

Graphical user interface, application

Description automatically generated

To select the board for programming,  go to the ‘*Tools’* menu item, followed by ‘*Board,’’* and then NodeMCU 12 module.

 Now we are ready to start programming my NodeMCU using the Arduino IDE.

**Installing Required Software Libraries**

Before moving directly to coding, let’s install the required libraries.

**Installing DHT Sensor Library**

You must install the DHT sensor library available from the Arduino library manager to read sensor data.

So navigate to Sketch > Include Library > Manage Libraries.

Graphical user interface, application

Description automatically generated

Search for DHT and choose one from a number of DHT libraries.

Graphical user interface, text, application

Description automatically generated

**Installing MQTT Client Library**

We need an MQTT Client library to publish MQTT messages to my HiveMQ Cloud MQTT Broker. For this demonstration, we will use a library called PubSubClient. To add it, I’ll go to Sketch > Include Library > Manage Libraries and search for PubSubClient.

Graphical user interface, text, application

Description automatically generated

There are many to choose from. I’ll add this one from Nick O’Leary.

**Programming NodeMCU to Send and Receive MQTT Messages**

Now that we’ve installed the required libraries, let’s go through the code for sending and receiving MQTT messages using NodeMCU on the Arduino platform.

First, I import the required libraries into my Arduino code.

#ifdef ESP8266

#include <ESP8266WiFi.h>

#else

#include <WiFi.h>

#endif

#include "DHTesp.h"

#include <ArduinoJson.h>

#include <PubSubClient.h>

#include <WiFiClientSecure.h>

Next,  declare GPIO2 as the pin to which my DHT sensor is connected. Also declare GPIO5 as the pin to which the LED is connected.

/\*\*\*\* DHT11 sensor Settings \*\*\*\*\*\*\*/

#define DHTpin 2 //Set DHT pin as GPIO2

DHTesp dht;

/\*\*\*\* LED Settings \*\*\*\*\*\*\*/

const int led = 5; //Set LED pin as GPIO5

After that, we declare variables for holding the Wi-Fi connection and HiveMQ Cloud MQTT Broker details

/\*\*\*\*\*\* WiFi Connection Details \*\*\*\*\*\*\*/

const char\* ssid = "your\_wifi\_ssid";

const char\* password = "your\_wifi\_password";

/\*\*\*\*\*\*\* MQTT Broker Connection Details \*\*\*\*\*\*\*/

const char\* mqtt\_server = "34bab4bb63014dce9e71f4ad8fb6ffc2.s2.eu.hivemq.cloud";

const char\* mqtt\_username = "your\_mqtt\_client\_username";

const char\* mqtt\_password = "your\_mqtt\_client\_password";

const int mqtt\_port =8883;

Then I make sure to initialize a secure Wi-Fi client connection, create an MQTT client using PubSubClient library and attach it to the secure Wi-Fi connectivity. Also, here I declare MQTT message buffer variables.

/\*\*\*\* Secure WiFi Connectivity Initialisation \*\*\*\*\*/

WiFiClientSecure espClient;

/\*\*\*\* MQTT Client Initialisation Using WiFi Connection \*\*\*\*\*/

PubSubClient client(espClient);

unsigned long lastMsg = 0;

#define MSG\_BUFFER\_SIZE (50)

char msg[MSG\_BUFFER\_SIZE];

Now that we’ve set our connectivity, we can declare a variable for holding a root certificate that allows the client to connect securely to the HiveMQ Cloud MQTT broker.

/\*\*\*\*\*\* root certificate \*\*\*\*\*\*\*\*\*/

static const char \*root\_ca PROGMEM = R"EOF(

-----BEGIN CERTIFICATE-----

MIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw

TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2Vh

cmNoIEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwHhcNMTUwNjA0MTEwNDM4

WhcNMzUwNjA0MTEwNDM4WjBPMQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJu

ZXQgU2VjdXJpdHkgUmVzZWFyY2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBY

MTCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAK3oJHP0FDfzm54rVygc

h77ct984kIxuPOZXoHj3dcKi/vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+

0TM8ukj13Xnfs7j/EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U

A5/TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW

T8KOEUt+zwvo/7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/ELNKjD+Jo2FR3qyH

B5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC

B5iPNgiV5+I3lg02dZ77DnKxHZu8A/lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv

KBds0pjBqAlkd25HN7rOrFleaJ1/ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn

OlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3SzynTn

jh8BCNAw1FtxNrQHusEwMFxIt4I7mKZ9YIqioymCzLq9gwQbooMDQaHWBfEbwrbw

qHyGO0aoSCqI3Haadr8faqU9GY/rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI

rU7m2Ys6xt0nUW7/vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV

HRMBAf8EBTADAQH/MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq

hkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1XIpu+ILlaS/V9lZL

ubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ

3BebYhtF8GaV0nxvwuo77x/Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/KK

NFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5

ORAzI4JMPJ+GslWYHb4phowim57iaztXOoJwTdwJx4nLCgdNbOhdjsnvzqvHu7Ur

TkXWStAmzOVyyghqpZXjFaH3pO3JLF+l+/+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC

jNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc

oyi3B43njTOQ5yOf+1CceWxG1bQVs5ZufpsMljq4Ui0/1lvh+wjChP4kqKOJ2qxq

4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/9yi0GE44Za4rF2LN9d11TPA

mRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d

emyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/+jh5Xos1AnX5iItreGCc=

-----END CERTIFICATE-----

)EOF";

Then  declare the function that   we will use to connect to the WiFi network.

/\*\*\*\*\*\*\*\*\*\*\*\*\* Connect to WiFi \*\*\*\*\*\*\*\*\*\*\*/

void setup\_wifi() {

delay(10);

Serial.print("\nConnecting to ");

Serial.println(ssid);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

randomSeed(micros());

Serial.println("\nWiFi connected\nIP address: ");

Serial.println(WiFi.localIP());

}

Next, we set  a function for esp8266 connecting to the HiveMQ Cloud MQTT broker.

/\*\*\*\*\*\*\*\*\*\*\*\*\* Connect to MQTT Broker \*\*\*\*\*\*\*\*\*\*\*/

void reconnect() {

// Loop until we're reconnected

while (!client.connected()) {

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

String clientId = "ESP8266Client-"; // Create a random client ID

clientId += String(random(0xffff), HEX);

// Attempt to connect

if (client.connect(clientId.c\_str(), mqtt\_username, mqtt\_password)) {

Serial.println("connected");

client.subscribe("led\_state"); // subscribe the topics here

} else {

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

Serial.print(client.state());

Serial.println(" try again in 5 seconds"); // Wait 5 seconds before retrying

delay(5000);

}

}

}

Then let’s add  a callback method for receiving MQTT messages and switching the LED ON and OFF.

/\*\*\*\*\* Call back Method for Receiving MQTT messages and Switching LED \*\*\*\*/

void callback(char\* topic, byte\* payload, unsigned int length) {

String incommingMessage = "";

for (int i = 0; i < length; i++) incommingMessage+=(char)payload[i];

Serial.println("Message arrived ["+String(topic)+"]"+incommingMessage);

//--- check the incomming message

if( strcmp(topic,"led\_state") == 0){

if (incommingMessage.equals("1")) digitalWrite(led, HIGH); // Turn the LED on

else digitalWrite(led, LOW); // Turn the LED off

}

}

After that, I implement  a method for publishing MQTT messages to my HiveMQ Cloud MQTT broker.

/\*\*\*\* Method for Publishing MQTT Messages \*\*\*\*\*\*\*\*\*\*/

void publishMessage(const char\* topic, String payload , boolean retained){

if (client.publish(topic, payload.c\_str(), true))

Serial.println("Message publised ["+String(topic)+"]: "+payload);

}

We move to the default Arduino setup method, where we set up the DHT 11 sensor and LED, enable the certificate for a secure connection, setup up Wi-Fi, and create instances of the MQTT server and callback methods.

/\*\*\*\* Application Initialisation Function\*\*\*\*\*\*/

void setup() {

dht.setup(DHTpin, DHTesp::DHT11); //Set up DHT11 sensor

pinMode(led, OUTPUT); //set up LED

Serial.begin(9600);

while (!Serial) delay(1);

setup\_wifi();

#ifdef ESP8266

espClient.setInsecure();

#else

espClient.setCACert(root\_ca); // enable this line and the the "certificate" code for secure connection

#endif

client.setServer(mqtt\_server, mqtt\_port);

client.setCallback(callback);

}

Finally, we are ready to add the main function for executing the application logic. If you are not connected, connect the MQTT client to the HiveMQ Cloud MQTT broker. Then read the temperature and humidity values from the DHT11 sensor, and serialize them, including ***deviceId*** and ***siteId*** properties, into a JSON object.

After serialization, I publish the JSON string to my HiveMQ Cloud MQTT Broker under the topic “esp8266\_data”, and repeat the process every 5 seconds.

/\*\*\*\*\*\*\*\* Main Function \*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop() {

if (!client.connected()) reconnect(); // check if client is connected

client.loop();

//read DHT11 temperature and humidity reading

delay(dht.getMinimumSamplingPeriod());

float humidity = dht.getHumidity();

float temperature = dht.getTemperature();

DynamicJsonDocument doc(1024);

doc["deviceId"] = "NodeMCU";

doc["siteId"] = "My Demo Lab";

doc["humidity"] = humidity;

doc["temperature"] = temperature;

char mqtt\_message[128];

serializeJson(doc, mqtt\_message);

publishMessage("esp8266\_data", mqtt\_message, true);

delay(5000);

}

Next, deploy this code onto the microcontroller board.

Now that my code is running and I’m publishing data to the HiveMQ Cloud MQTT Broker, I can use an MQTT Client application such as MQTTfx to subscribe to the broker and view the data.

When I open my MQTT.fx application, which is already connected to the HiveMQ Cloud MQTT broker and subscribed to the same topic, I can see the data coming.

Graphical user interface, text, application

Description automatically generated

Seeing data flowing means you are successfully publishing data from the NodeMCU using MQTT.

Now, I will publish an MQTT command to switch on the LED light from MQTT.fx by publishing the value of “1” under the topic “led\_state”. When I do that, you can see the LED light switch ON.

Graphical user interface, application

Description automatically generated

To switch the light off, publish a value of “0”.

**Conclusion**

Together, we have experienced using the Arduino IDE to program an ESP8266 microcontroller to send and receive MQTT messages from a HiveMQ Cloud MQTT broker. How was the process for you? Can you see other use cases?

Try out the above demonstration and get started with building MQTT applications by signing up for a free HiveMQ Cloud account.

[Sign-up for HiveMQ Cloud Now](https://console.hivemq.cloud/?utm_source=MQTT+on+Arduino+NodeMCU+ESP8266+Blog&utm_medium=CTA+Button&utm_campaign=HiveMQ+Cloud&__hstc=184124345.da6b1dc05694d0cee1e417c536b5e2ce.1677057302006.1677057302006.1677057302006.1&__hssc=184124345.1.1677057302006&__hsfp=3034756077)