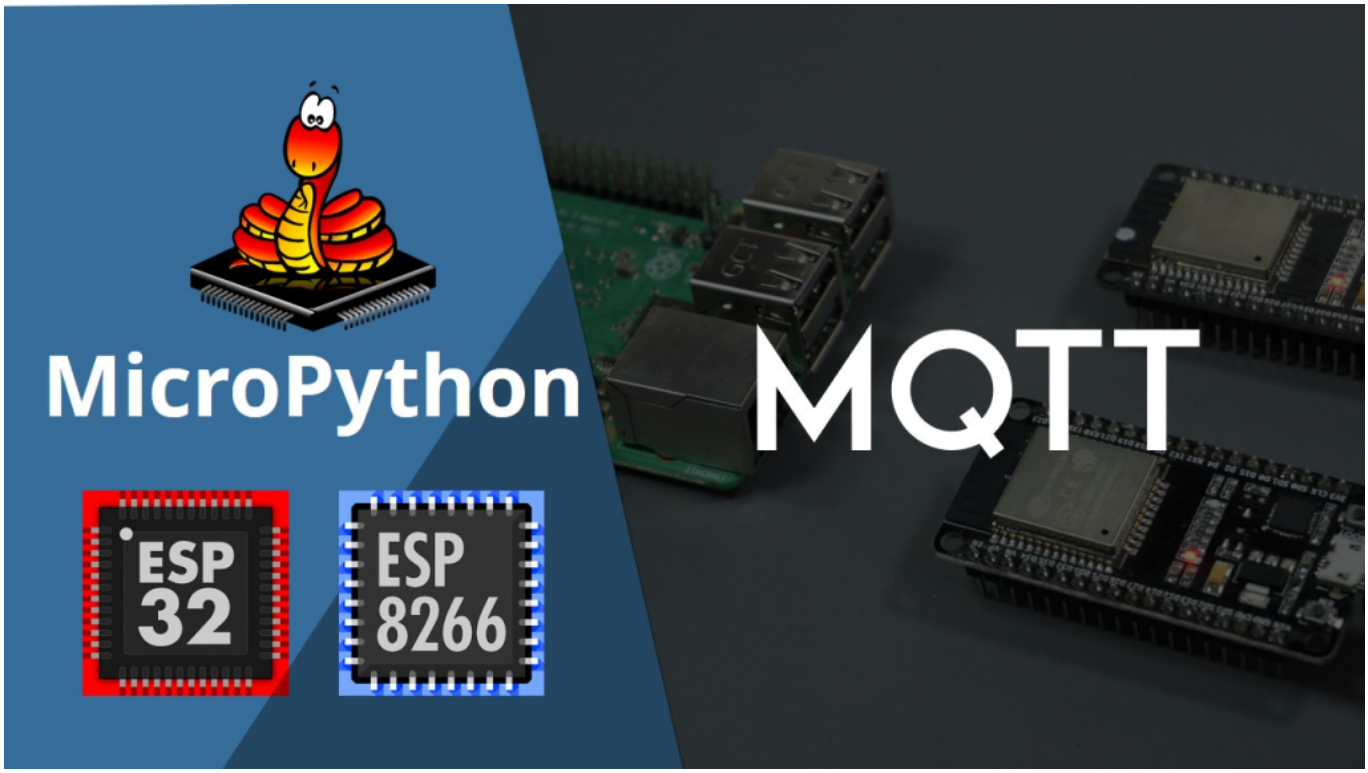
MicroPython – Getting Started with MQTT on ESP32/ESP8266

In this tutorial, we'll show you how to use MQTT to exchange data between two ESP32/ESP8266 boards using MicroPython firmware. As an example, we'll exchange simple text messages between two ESP boards. The idea is to use the concepts learned here to exchange sensor readings, or commands.



Note: *this tutorial is compatible with both the ESP32 and ESP8266 development boards.*

Prerequisites

Before continuing with this tutorial, make sure you complete the following prerequisites:

MicroPython firmware

To program the ESP32 and ESP8266 with MicroPython, we use uPyCraft IDE as a programming environment. Follow the next tutorials to install uPyCraft IDE and flash MicroPython firmware on your board:

- Install uPyCraft IDE: Windows PC, MacOS X, or Linux Ubuntu
- Flash/Upload MicroPython Firmware to ESP32 and ESP8266

MQTT Broker

To use MQTT, you need a broker. We'll be using Mosquitto broker installed on a Raspberry Pi. Read How to Install Mosquitto Broker on Raspberry Pi.

If you're not familiar with MQTT make sure you read our introductory tutorial: What is MQTT and How It Works

Parts Required

For this tutorial you need two ESP32 or two ESP8266 boards:

- 2x ESP32 DEVKIT DOIT board – read ESP32 Development Boards Review and Comparison
- (alternative) 2x ESP8266-12E NodeMCU Kit – read Best ESP8266 Wi-Fi Development Board

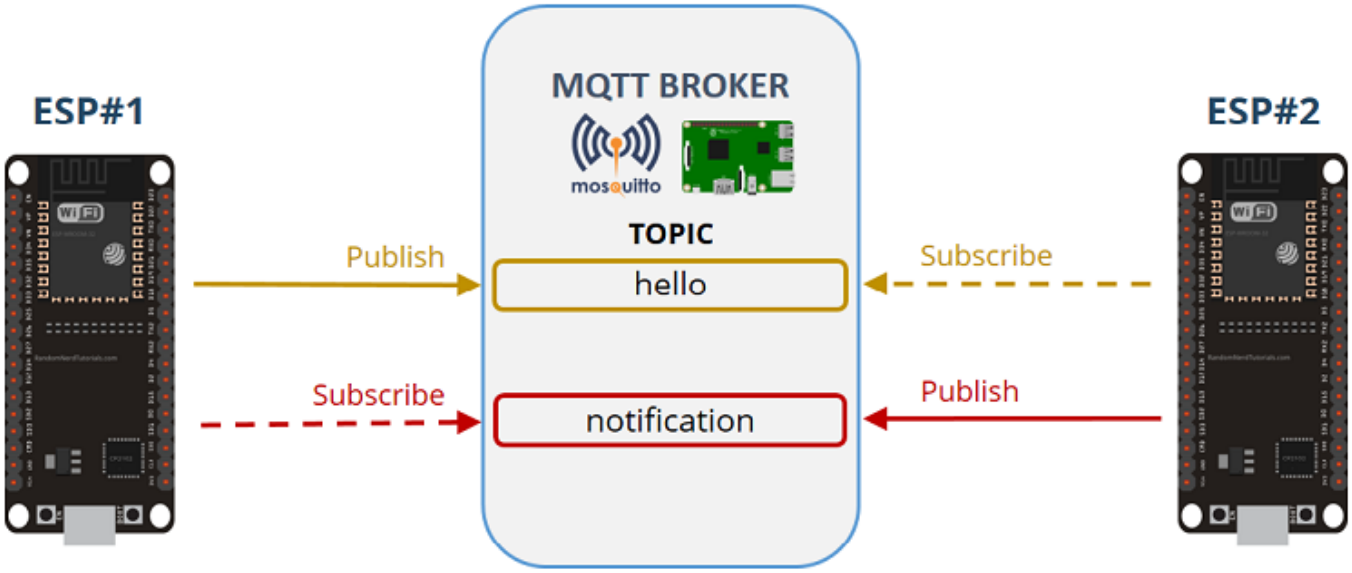
You also need a Raspberry Pi and the following accessories:

- Raspberry Pi board – read Best Raspberry Pi Starter Kits
- MicroSD Card – 16GB Class10
- Raspberry Pi Power Supply (5V 2.5A)

You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Project Overview

Here's a high-level overview of the project we'll build:



- ESP#1 publishes messages on the ***hello*** topic. It publishes a "Hello" message followed by a counter (Hello 1, Hello 2, Hello 3, …). It publishes a new message every 5 seconds.
- ESP#1 is subscribed to the ***notification*** topic to receive notifications from the ESP#2 board.
- ESP#2 is subscribed to the ***hello*** topic. ESP #1 is publishing in this topic. Therefore, ESP#2 receives ESP#1 messages.
- When ESP#2 receives the messages, it sends a message saying 'received'. This message is published on the ***notification*** topic. ESP#1 is subscribed to that topic, so it receives the message.

Preparing ESP#1

Let's start by preparing ESP#1:

- It is subscribed to the ***notification*** topic
- It publishes on the ***hello*** topic

Importing umqttsimple library

To use MQTT with the ESP32/ESP8266 and MicroPython, you need to install the *umqttsimple* library.

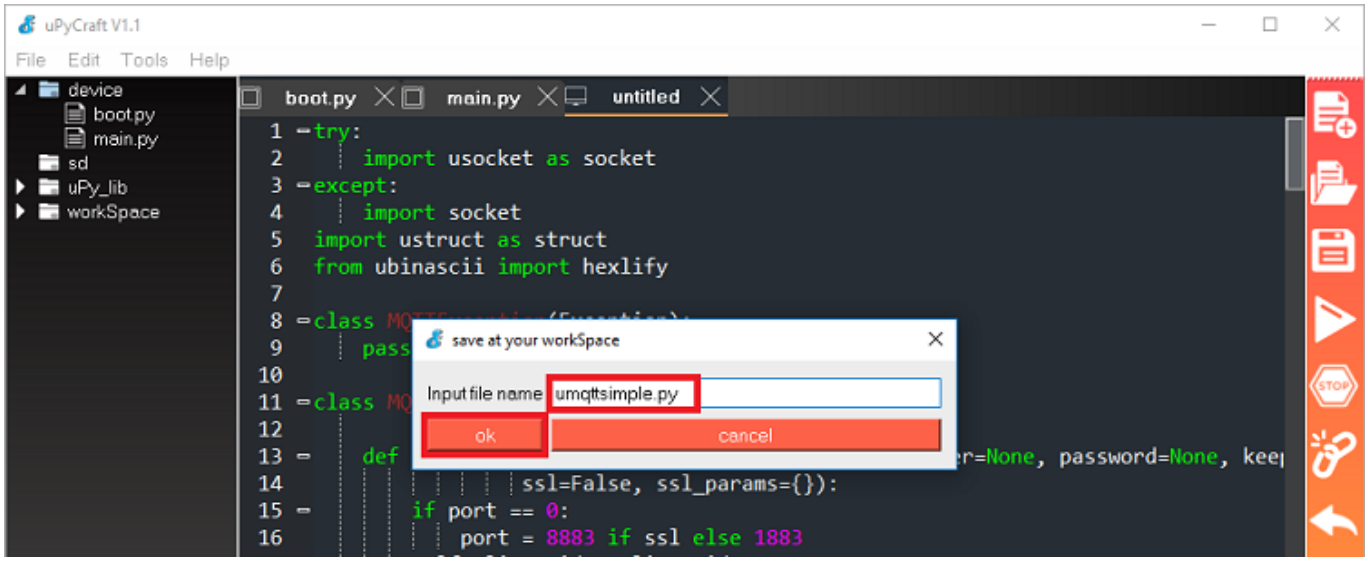
1. Create a new file by pressing the **New File** button.

2. Copy the *umqttsimple* library code into it. You can access the *umqttsimple* library code in the following link:

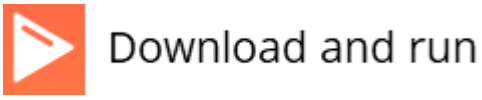
- https://raw.githubusercontent.com/RuiSantosdotme/ESP-MicroPython/master/code/MQTT/umqttsimple.py

3. Save the file by pressing the **Save** button.

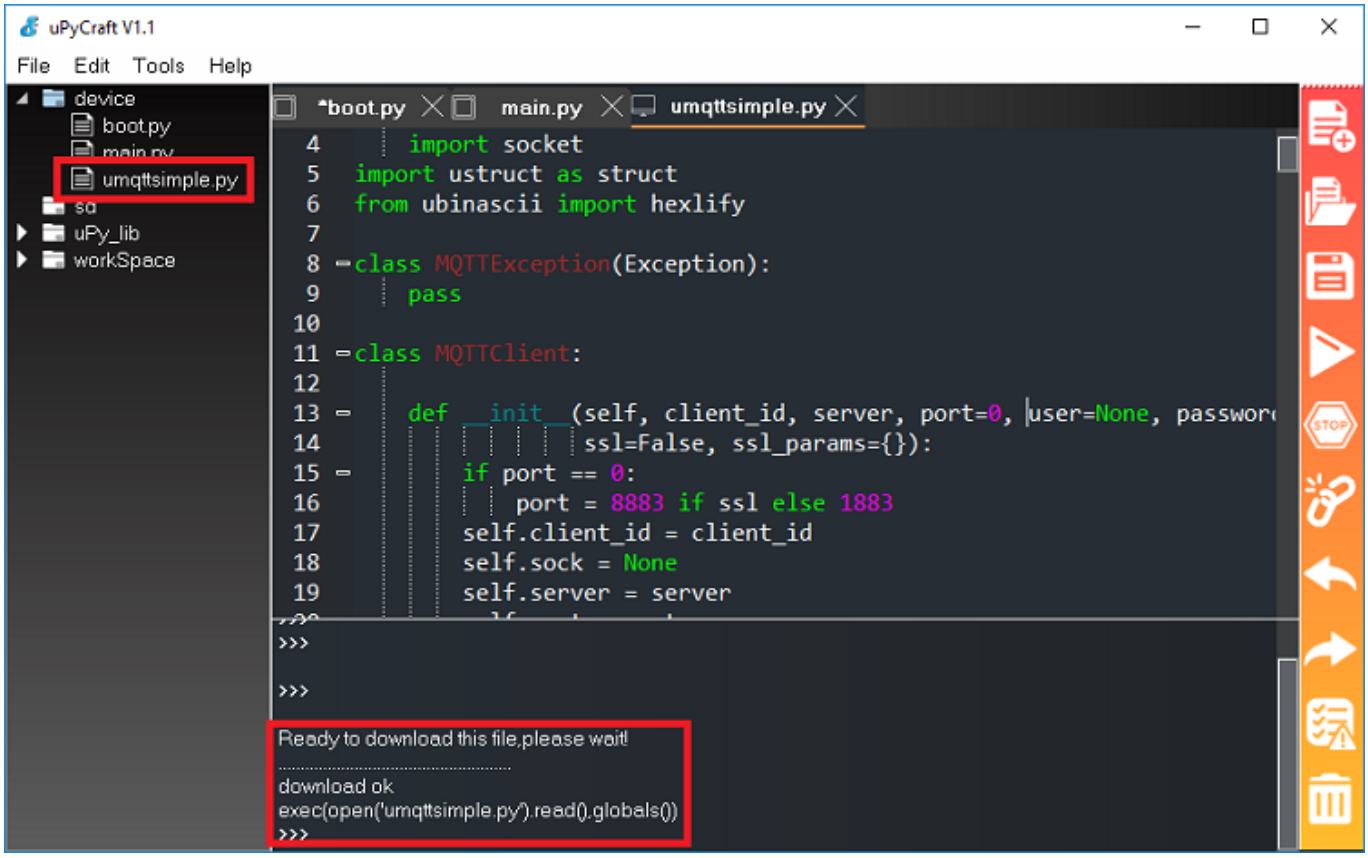
4. Call this new file "**umqttsimple.py**" and press **ok**.



5. Click the **Download and Run** button.



6. The file should be saved on the *device* folder with the name "**umqttsimple.py**" as highlighted in the figure below.



Now, you can use the library functionalities in your code by importing the library.

boot.py

Open the *boot.py* file and copy the following code to ESP#1.

```
mqtt_server = 'REPLACE_WITH_YOUR_MQTT_BROKER_IP'
#EXAMPLE IP ADDRESS
#mqtt_server = '192.168.1.144'
client_id = ubinascii.hexlify(machine.unique_id())
topic_sub = b'notification'
topic_pub = b'hello'

last_message = 0
message_interval = 5
counter = 0

station = network.WLAN(network.STA_IF)

station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
  pass

print('Connection successful')
print(station.ifconfig())
```

View raw code

How the Code Works

You need to import all the following libraries:

```
import time
from umqttsimple import MQTTClient
import ubinascii
import machine
import micropython
import network
import esp
```

Set the debug to None and activate the garbage collector.

```
esp.osdebug(None)
import gc
gc.collect()
```

In the following variables, you need to enter your network credentials and your broker IP address.

```
ssid = 'REPLACE_WITH_YOUR_SSID'
password = 'REPLACE_WITH_YOUR_PASSWORD'
mqtt_server = 'REPLACE_WITH_YOUR_MQTT_BROKER_IP'
```

For example, our broker IP address is: 192.168.1.144.

Note: read this tutorial to see how to get your broker IP address.

To create an MQTT client, we need to get the ESP unique ID. That's what we do in the following line (it is saved on the `client_id` variable).

```
client_id = ubinascii.hexlify(machine.unique_id())
```

Next, write the topic the ESP#1 is subscribed to, and the topic it will be publishing messages:

```
topic_sub = b'notification'
topic_pub = b'hello'
```

Then, create the following variables:

```
last_message = 0
message_interval = 5
```

```
counter = 0
```

The `last_message` variable will hold the last time a message was sent. The `message_interval` is the time between each message sent. Here, we're setting it to 5 seconds (this means a new message will be sent every 5 seconds). The `counter` variable is simply a counter to be added to the message.

After that, we make the procedures to connect to the network.

```
station = network.WLAN(network.STA_IF)

station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
  pass

print('Connection successful')
print(station.ifconfig())
```

main.py

In the *main.py* file is where we'll write the code to publish and receive the messages. Copy the following code to your *main.py* file.

```
client = MQTTClient(client_id, mqtt_server)
```

```
try:
  client = connect_and_subscribe()
except OSError as e:
  restart_and_reconnect()

while True:
  try:
    client.check_msg()
    if (time.time() - last_message) > message_interval:
      msg = b'Hello #%d' % counter
      client.publish(topic_pub, msg)
      last_message = time.time()
      counter += 1
  except OSError as e:
    restart_and_reconnect()
```

View raw code

How the code works

The first thing you should do is creating a callback function that will run whenever a message is published on a topic the ESP is subscribed to.

Callback function

The callback function should accept as parameters the topic and the message.

```
def sub_cb(topic, msg):
  print((topic, msg))
  if topic == b'notification' and msg == b'received':
    print('ESP received hello message')
```

In our callback function, we start by printing the topic and the message. Then, we check if the message was published on the ***notification*** topic, and if the content of

the message is 'received'. If this if statement is True, it means that ESP#2 received the 'hello' message sent by ESP#1.

Basically, this callback function handles what happens when a certain message is received on a certain topic.

Connect and subscribe

Then, we have the `connect_and_subscribe()` function. This function is responsible for connecting to the broker as well as to subscribe to a topic.

```
def connect_and_subscribe():
```

Start by declaring the `client_id`, `mqtt_server` and `topic_sub` variables as global variables. This way, we can access these variables throughout the code.

```
global client_id, mqtt_server, topic_sub
```

Then, create a `MQTTClient` object called `client`. We need to pass as parameters the `cliend_id`, and the IP address of the MQTT broker (`mqtt_server`). These variables were set on the *boot.py* file.

```
client = MQTTClient(client_id, mqtt_server)
```

After that, set the callback function to the client (`sub_cb`).

```
client.set_callback(sub_cb)
```

Next, connect the client to the broker using the `connect()` method on the `MQTTClient` object.

```
client.connect()
```

After connecting, we subscribe to the `topic_sub` topic. Set the `topic_sub` on the *boot.py* file (`notification`).

```
client.subscribe(topic_sub)
```

Finally, print a message and return the client:

```
print('Connected to %s MQTT broker, subscribed to %s topic' % (mqtt
return client
```

Restart and reconnect

We create a function called `restart_and_reconnect()` . This function will be called in case the ESP32 or ESP8266 fails to connect to the broker.

This function prints a message to inform that the connection was not successful. We wait 10 seconds. Then, we reset the ESP using the `reset()` method.

```
def restart_and_reconnect():
  print('Failed to connect to MQTT broker. Reconnecting...')
  time.sleep(10)
  machine.reset()
```

Receive and publish messages

Until now, we've created functions to handle tasks related with the MQTT communication. From now on, the code will call those functions to make things happen.

The first thing we need to do is to connect to the MQTT broker and subscribe to a topic. So, we create a client by calling the `connect_and_subscribe()` function.

```
try:
  client = connect_and_subscribe()
```

In case we're not able to connect to the MQTTT broker, we'll restart the ESP by calling the `restart_and_reconnect()` function

```
except OSError as e:
  restart_and_reconnect()
```

In the while loop is where we'll be receiving and publishing the messages. We use `try` and `except` statements to prevent the ESP from crashing in case something goes wrong.

Inside the `try` block, we start by applying the `check_msg()` method on the `client`.

```
try:
  client.check_msg()
```

The `check_msg()` method checks whether a pending message from the server is available. It waits for a single incoming MQTT message and process it. The subscribed messages are delivered to the callback function we've defined earlier (the `sub_cb()` function). If there isn't a pending message, it returns with `None`.

Then, we add an `if` statement to checker whether 5 seconds (`message_interval`) have passed since the last message was sent.

```
if (time.time() - last_message) > message_interval:
```

If it is time to send a new message, we create a `msg` variable with the "Hello" text followed by a counter.

```
  msg = b'Hello #%d' % counter
```

To publish a message on a certain topic, you just need to apply the `publish()` method on the `client` and pass as arguments, the topic and the message. The `topic_pub` variable was set to `hello` in the *boot.py* file.

```
  client.publish(topic_pub, msg)
```

After sending the message, we update the last time a message was received by setting the `last_message` variable to the current time.

```
  last_message = time.time()
```

Finally, we increase the `counter` variable in every loop.

```
  counter += 1
```

If something unexpected happens, we call the `restart_and_reconnect()` function.

```
  except OSError as e:
    restart_and_reconnect()
```

That's it for ESP#1. Remember that you need to upload all the next files to make the project work (you should upload the files in order):

1. *umqttsimple.py*;
2. *boot.py*;
3. *main.py*.

After uploading all files, you should get success messages on: establishing a network connection; connecting to the broker; and subscribing to the topic.

ESP #2

Let's now prepare ESP#2:

- It is subscribed to the ***hello*** topic
- It publishes on the ***notification*** topic

Like the ESP#1, you also need to upload the *umqttsimple.py*, *boot.py*, and *main.py* files.

Importing umqttsimple

To use MQTT with the ESP32/ESP8266 and MicroPython, you need to install the *umqttsimple* library. Follow the steps described earlier to install the *umqttsimple* library in ESP#2.

You can access the umqttsimple library code in the following link:

- https://raw.githubusercontent.com/RuiSantosdotme/ESP-MicroPython/master/code/MQTT/umqttsimple.py

boot.py

Copy the following code to the ESP#2 *boot.py* file.

```
import machine
import micropython
import network
import esp
esp.osdebug(None)
import gc
gc.collect()
```

```
client_id = ubinascii.hexlify(machine.unique_id())
topic_sub = b'hello'
topic_pub = b'notification'

station = network.WLAN(network.STA_IF)

station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
  pass

print('Connection successful')
```

View raw code

This code is very similar with the previous *boot.py* file. You need to replace the following variables with your network credentials and the broker IP address.

```
ssid = 'REPLACE_WITH_YOUR_SSID'
password = 'REPLACE_WITH_YOUR_PASSWORD'
mqtt_server = 'REPLACE_WITH_YOUR_MQTT_BROKER_IP'
```

The only difference here is that we subscribe to the ***hello*** topic and publish on the ***notification*** topic.

```
topic_sub = b'hello'
topic_pub = b'notification'
```

main.py

Copy the following code to the ESP#2 *main.py* file.

```
# Complete project details at https://RandomNerdTutorials.com
```

```
def sub_cb(topic, msg):
  print((topic, msg))

def connect_and_subscribe():
  global client_id, mqtt_server, topic_sub
  client = MQTTClient(client_id, mqtt_server)
  client.set_callback(sub_cb)
  client.connect()
  client.subscribe(topic_sub)
  print('Connected to %s MQTT broker, subscribed to %s topic' % (
  return client

def restart_and_reconnect():
  print('Failed to connect to MQTT broker. Reconnecting...')
  time.sleep(10)
  machine.reset()

try:
  client = connect_and_subscribe()
except OSError as e:
  restart_and_reconnect()

while True:
  try:
```

View raw code

This code is very similar with the *main.py* from ESP#1. We create the `sub_cb()`, the `connect_and_subscribe()` and the `restart_and_reconnect()` functions. This time, the `sub_cb()` function just prints information about the topic and received message.

```
def sub_cb(topic, msg):
  print((topic, msg))
```

In the while loop, we check if we got a new message and save it in the `new_message` variable.

```
  new_message = client.check_msg()
```

If we receive a new message, we publish a message saying 'received' on the `topic_sub` topic (in this case we set it to `notification` in the *boot.py* file).

```
  if new_message != 'None':
    client.publish(topic_pub, b'received')
```

That's it for ESP#2. Remember that you need to upload all the next files to make the project work (you should upload the files in order):

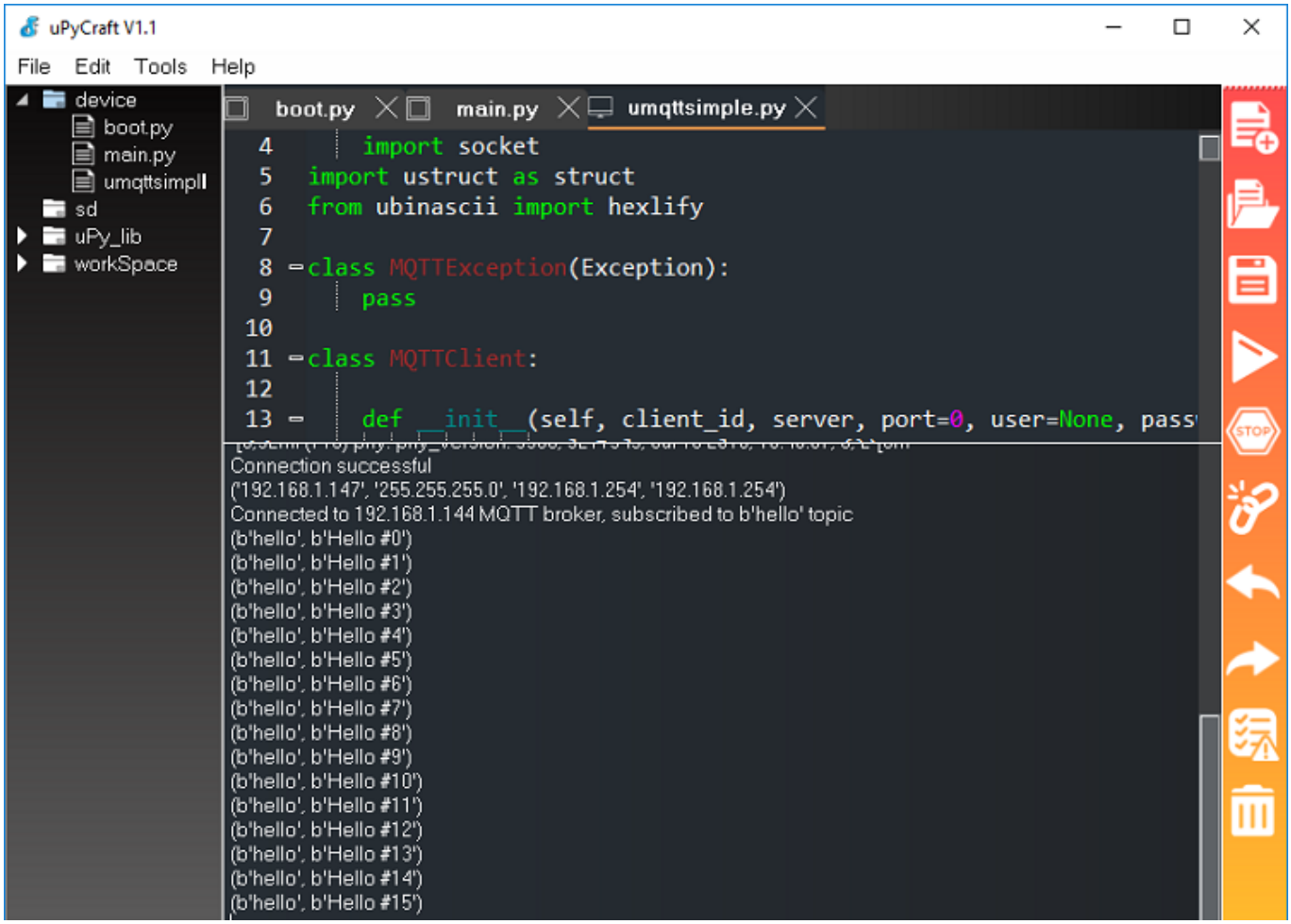
1. *umqttsimple.py*;
2. *boot.py*;
3. *main.py*.

The ESP32/ESP8266 should establish a network connection and connect to the broker successfully.

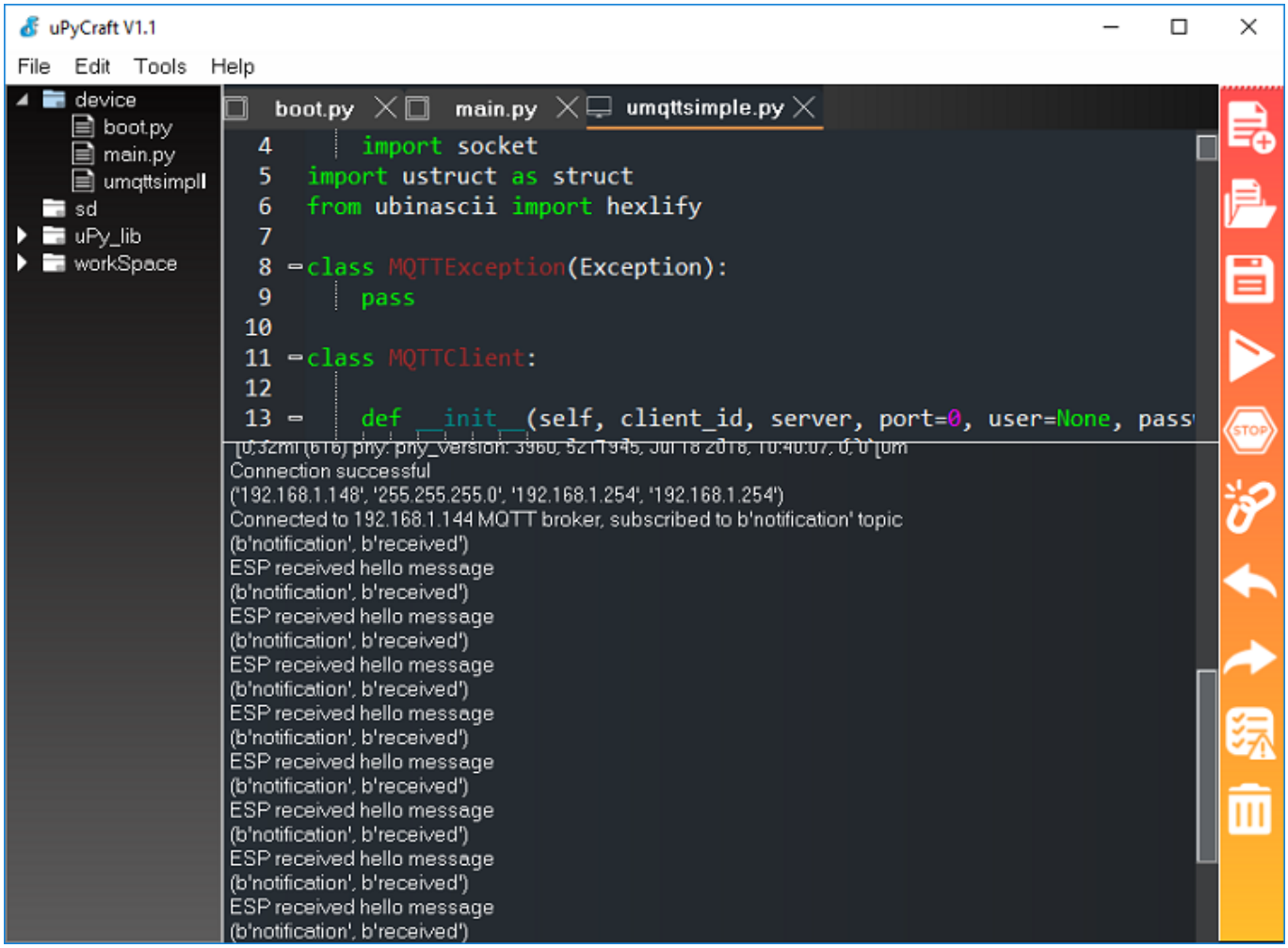
Demonstration

After uploading all the necessary scripts to both ESP boards and having both boards and the Raspberry Pi with the Mosquitto broker running, you are ready to test the setup.

The ESP#2 should be receiving the "Hello" messages from ESP#1, as shown in the figure below.



On the other side, ESP#1 board should receive the “received” message. The “received” message is published by ESP#2 on the ***notification*** topic. ESP#1 is subscribed to that topic, so it receives the message.



Wrapping Up

In this simple example, you've learned how to exchange text between two ESP32/ESP8266 boards using MQTT communication protocol. The idea is to use the concepts learned here to exchange useful data like sensor readings or commands to control outputs.

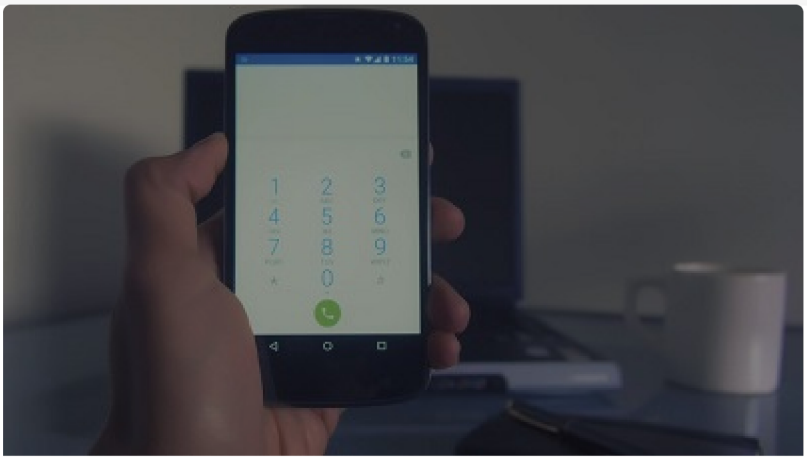
If you like MicroPython with the ESP32/ESP8266, you may also like:

- **MicroPython Programming with ESP32 and ESP8266 eBook**
- Getting Started with MicroPython on ESP32 and ESP8266
- MicroPython with ESP32 and ESP8266: Interacting with GPIOs
- ESP32/ESP8266 MicroPython Web Server – Control Outputs

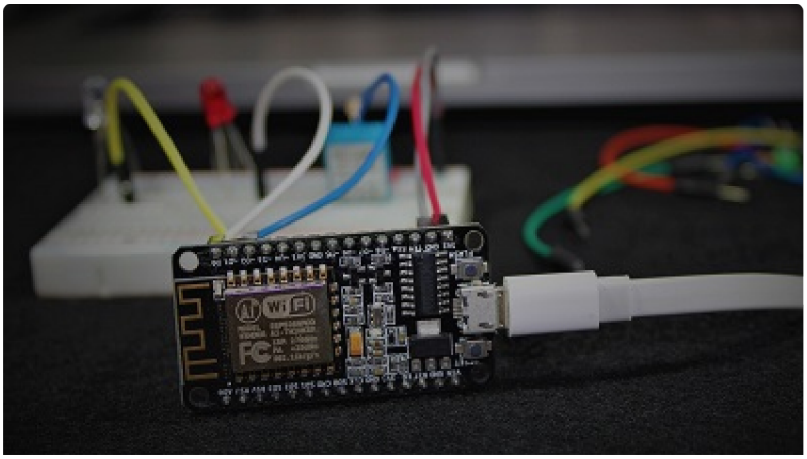
[eBook] Build Web Servers with ESP32 and ESP8266 (2nd Edition)

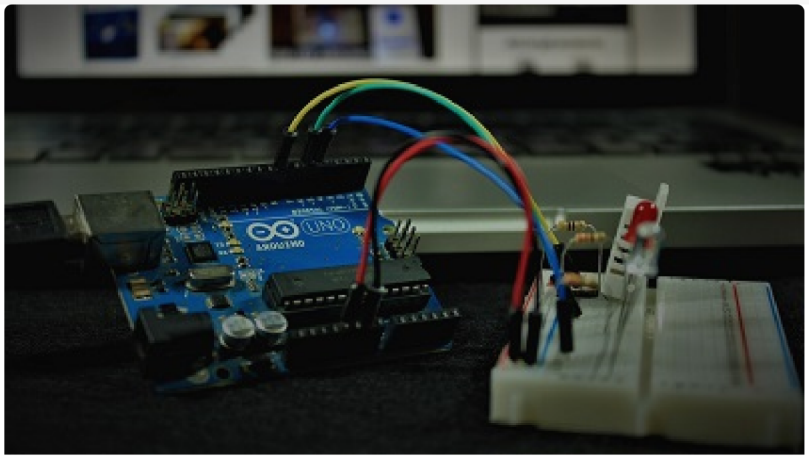
Build Web Server projects with the ESP32 and ESP8266 boards to control outputs and monitor sensors remotely. Learn HTML, CSS, JavaScript and client-server communication protocols **DOWNLOAD »**

Recommended Resources



Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED.





What to Read Next…

Better Debugging for Arduino IDE: SerialDebug Library (Part 1)

ESP8266 NodeMCU with MPU-6050 Accelerometer, Gyroscope and Temperature Sensor (Arduino)

53 thoughts on “MicroPython – Getting Started with MQTT on ESP32/ESP8266”

Miguel Wisintainer
December 7, 2018 at 12:17 pm

How is going mypython to ESP32 (BLE) ? (BLUETOOTH)

Reply

Sara Santos
December 9, 2018 at 2:31 pm

Hi Miguel.
As far as I know, it is not yet implemented in the official MicroPython distribution.
Regards,
Sara

Reply

Dušan LEZO
December 7, 2018 at 10:25 pm

Prakticke, jednoduche perfekt!
Priama komunikacia ESP32 s ESP32 bez Pi s micropython ?

Reply

Sara Santos
December 8, 2018 at 5:14 pm

Hi Dusan.
The most reliable way for establishing a two-way communication between two ESP boards is using MQTT. For that you either need a local broker (for example, installed on a Raspberry Pi), or you can use a cloud MQTT broker and you don’t need a Pi.
Otherwise, you can establish a Wi-Fi connection between the two boards. But it will be a bit tricky to use something like that for a two-way communication.
Regards,
Sara 🙂

Reply

anand
February 11, 2021 at 3:34 pm

how to configure raspi as a broker

Reply

Sara Santos

February 11, 2021 at 4:11 pm

Follow this tutorial. https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/

Reply

Ian

December 12, 2018 at 3:58 pm

Hi

Does the mqtt library in this tutorial support mqtt over websocketS? If not do you know an ESP32 Arduino or Micropython library that does?

Regards

Ian

Reply

Rui Santos

December 17, 2018 at 4:05 pm

With this MicroPython MQTT library all Subscribed messages are delivered via a callback function.

Reply

Bill

February 13, 2019 at 2:36 am

code reports a syntax error in line 82. Code copied directly from webpage with zero changes made to it. Has anyone actually gotten this code to run?

https://raw.githubusercontent.com/RuiSantosdotme/ESP-MicroPython/master/code/MQTT/umqttsimple.py

Reply

Sara Santos

February 13, 2019 at 9:42 am

Hi Bill.
That is the umqtt simple library code. I've tested uploading the library to my ESP32 and it is working fine.
You probably have an indentation error.Please check that you have copied the code correctly and that all the indentation is ok.

With some editors, sometimes it is very easy to accidentally mess up with indentation copying code.
Regards,
Sara

Reply

Allen
June 28, 2023 at 12:18 pm

I know this is old but I had this issue too. Here's the fix: set keepalive to a nonzero number in the constructor in umqttsimple.py.

def **init**(self, client_id, server, port=0, user=None, password=None, keepalive=1 <<– THIS

github.com/micropython/micropython-lib/issues/466

Reply

Sara Santos
June 28, 2023 at 6:03 pm

Thanks for the tip.
Regards,
Sara

Reply

Jan Hurban

May 19, 2019 at 6:18 pm

one of the best mqtt instructions, thank you very much!

Reply

Sara Santos

May 22, 2019 at 11:41 am

Thank you for your comment.
Regards,
Sara

Reply

Peter Ashford

April 24, 2020 at 6:23 pm

please could you explain how you put in username and password.
I want to use my home assistant mqtt server.
Peter

Reply

Sara Santos

April 24, 2020 at 10:30 pm

Hi Peter.
If your MQTT broker requires username and password, you should use the following line to pass your broker username and password as arguments when defining the MQTTClient
client = MQTTClient(client_id, mqtt_server, user=your_username, password=your_password)
Regards,
Sara

Reply

Mauro Paschoini
February 21, 2021 at 10:54 am

You need use:

client = MQTTClient(client_id, mqtt_server, user=b'your_username', password=b'your_password')

Reply

Tarjei
June 13, 2019 at 11:21 am

Great article! I'm trying to connect to a MQTT server with username/password though – do you have any hints on how I can achieve this?

Reply

Tarjei

June 13, 2019 at 11:29 am

Nevermind – read the code and it was obvious. Thanks!

Reply

Nicke

October 16, 2019 at 9:16 am

Very nice article, I tried to figure out how to use keep alive and will so that I get a message when the device loses the connection to the broker. I now have a device that does not always detect that it is offline and does not restart. Thanks again for your effort!

Reply

Sara Santos

October 16, 2019 at 10:12 am

Hi.
We added the restart_and_reconnect() that is responsible for restarting your board in case it disconnects from the broker.
Isn’t that happening in your case?

Regards,
Sara

Reply

Nicke

October 16, 2019 at 11:23 am

Hi Sara and thanks for the fast answer. I've added stuff to the code (running leds based on mqtt messages) but after running for 4-5 hours the boards does not do anything anymore (I have 5 boards and today 2 of them stopped). The reconnect/restart happens if I shutdown the broker when it is still connected but it does not work if it has been on for hours and has lost connection. The only way to know if they are working is to check if they respond to the commands I send. That is why the keepalive is important because otherwise it will take hours before the will is broadcasted.

Reply

LMtx

April 1, 2020 at 5:02 am

If you are using ESP8266 I recommend that instead of calling `client.check_msg()` you use `client.wait_msg()`. Executing `client.check_msg()` in the while loop caused my device to constantly restart after a few minutes.

Reply

Rui Santos

April 3, 2020 at 10:21 am

Thanks for the tip!

Reply

Frank

January 26, 2021 at 9:19 pm

Late coming in…
Great post!
one comment: The umqttsimple lib in the top is spelled with 3 ‘t’s
one question: instead of a second ESP32 how could I use the raspberry wthout the mosquito installed?

Reply

Sara Santos

January 27, 2021 at 11:03 am

Hi.
I’m sorry, but I didn’t understand you question. You need the Raspberry Pi with the mosquitto because MQTT needs a broker. So, you’ll always need the RPi, unless you want to install the broker on your computer or on the

cloud: https://randomnerdtutorials.com/cloud-mqtt-mosquitto-broker-access-anywhere-digital-ocean/

Thanks for telling me about the typo, it is fixed now.

Regards,
Sara

Reply

Leo B

July 22, 2021 at 2:05 am

Also late finding this…
I'm running my MQTT broker on a Win7 machine and trying to connect a NodeMCU running micro python on the same network. I've tried connecting to 'localhost', '127.0.0.1', and '192.168.12.229' (Win7 IP), but nothing works. It works fine when I connect to a cloud MQTT broker. The service is running and listening on port 1883. Any advice?
Thanks.

Reply

Zoyam

September 29, 2021 at 2:55 pm

Could I use mqtt.eclipseprojects.io as MQTT server?
For example, modify mqtt_server =

‘REPLACE_WITH_YOUR_MQTT_BROKER_IP’
become mqtt_server = ‘mqtt.eclipseprojects.io’

Reply

Sara Santos
September 29, 2021 at 4:28 pm

Hi.
Yes. You can use a cloud MQTT broker.
REgards,
Sara

Reply

Zoyam
September 30, 2021 at 3:40 am

I try the demo code but it’s fail on client.connect() by connect_and_subscribe() functions.
It seem that I didn’t connect to the broker successfully.
Traceback (most recent call last):
File “main.py”, line 21, in (this is point to try: client = connect_and_subscribe())
File “main.py”, line 10, in connect_and_subscribe () (this is point to client.connect())
File “umqttsimple.py”, line 103, in connect
MQTTException: 2

Reply

Peter-Frank Spierenburg

October 1, 2021 at 1:30 am

I am trying to connect to test.mosquitto.org using umqttsimple and micropython but I keep getting an exception:

```
$ ./micropython
MicroPython v1.17-74-gd42cba0d2 on 2021-09-30; linux version
Use Ctrl-D to exit, Ctrl-E for paste mode
>>> from umqttsimple import MQTTClient
>>> client = MQTTClient("test_client", "test.mosquitto.org")
>>> client.connect()
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "umqttsimple.py", line 102, in connect
MQTTException: 2
>>>
```

This is running on the linux version of micropython, but I've also tried on an ESP8266 running micropython and get the same error.

Reply

Sara Santos

October 1, 2021 at 10:49 am

Hi.
Do you get the same error when using the complete script?

Regards,
Sara

Reply

Peter-Frank Spierenburg
October 1, 2021 at 3:16 pm

So I've combined the boot.py and main.py together and commented out the parts that require an ESP:

```
# Complete project details at https://RandomNerdTutorials.com

import time
from umqttsimple import MQTTClient
import ubinascii
import machine
import micropython
#import network
#import esp
#esp.osdebug(None)
import gc
gc.collect()

ssid = 'REPLACE_WITH_YOUR_SSID'
password = 'REPLACE_WITH_YOUR_PASSWORD'
mqtt_server = 'mqtt.eclipseprojects.io'
#EXAMPLE IP ADDRESS
#mqtt_server = '192.168.1.144'
client_id = "client_id"
topic_sub = b'notification'
topic_pub = b'hello'

last_message = 0
message_interval = 5
counter = 0

#station = network.WLAN(network.STA_IF)

#station.active(True)
#station.connect(ssid, password)
```

```
#while station.isconnected() == False:
# pass

#print('Connection successful')
#print(station.ifconfig())

# Complete project details at https://RandomNerdTutorials.com

def sub_cb(topic, msg):
print((topic, msg))
if topic == b'notification' and msg == b'received':
print('ESP received hello message')

def connect_and_subscribe():
global client_id, mqtt_server, topic_sub
client = MQTTClient(client_id, mqtt_server)
client.set_callback(sub_cb)
client.connect()
client.subscribe(topic_sub)
print('Connected to %s MQTT broker, subscribed to %s topic' %
(mqtt_server, topic_sub))
return client

def restart_and_reconnect():
print('Failed to connect to MQTT broker. Reconnecting...')
time.sleep(10)
machine.reset()

try:
client = connect_and_subscribe()
except OSError as e:
print(e)
restart_and_reconnect()
while True:
try:
client.check_msg()
if (time.time() - last_message) > message_interval:
msg = b'Hello #%d' % counter
client.publish(topic_pub, msg)
last_message = time.time()
counter += 1
except OSError as e:
restart_and_reconnect()
```

and I can confirm that I get the same error:

```
$ ./micropython tmp.py
Traceback (most recent call last):
File "tmp.py", line 60, in <module>
File "tmp.py", line 49, in connect_and_subscribe
File "umqttsimple.py", line 102, in connect
MQTTException: 2
```

Reply

Marco Vallejos

October 3, 2021 at 2:34 am

MQTTException: 2 in $SYS/log says:”Bad socket read/write on client . Invalid arguments provided.”

https://github.com/micropython/micropython-lib/issues/445
https://github.com/hiveeyes/terkin-datalogger/pull/97

##
client = MQTTClient(client_id, mqtt_server, keepalive=60)
##

Reply

DevNerdChic

January 31, 2022 at 12:12 pm

I had to do the same thing to get it to work with my TinyPico esp 32, any changes to the boot.py made it unusable. I did learn how to reflash it though lol… I’m now an expert at that 😉

Reply

DevNerdChic

January 31, 2022 at 12:14 pm

I didn't have this issue though, mine is working.

Emilio

October 24, 2021 at 8:18 pm

Hi,
Thanks for sharing.
Why using umqttsimple library and not the built in umqtt.robust? What's the difference? I'm using an ESP32.
Thank you.

Reply

marc

November 11, 2021 at 8:34 pm

hi,
Is there an easy way to store the subscribed message in a variable on the esp ?
(i tried to store the message in a global var in the de "def sub_cb(topic, msg):" but that is not so stable)

```
def sub_cb(topic, msg):
print((topic, msg))
test = (topic, msg)
data = test[1].decode(“utf-8”)
global command
command = data
```

Reply

James

November 14, 2022 at 10:55 pm

I would like this answer as well 😀 !

Reply

Dave Festing

January 29, 2022 at 7:29 am

Thanks for providing this example code and special thanks to Macro for the keepalive fix.

Reply

DevNerdChic

January 31, 2022 at 12:10 pm

This is a great tutorial thank you!! best one I found while researching AND got my esp32 up and working with mqtt….my only question is… and I'm new to micro python so forgive me if this is really basic… why do the pub/sub topics have a 'b' in front of them?

Reply

Lucas

February 12, 2022 at 5:04 am

Hi,
I am implement the code to MQTT and the conecction with the WIFI is successful however it trying conect with the broker is 'Failed to connect to MQTT broker. Reconnecting…' i am used the Node Red.
someone can help me.
Thank you.
Lucas

Reply

Sara Santos

February 14, 2022 at 12:03 pm

Hi.
double-check that your broker is connected to the internet and that you've inserted the right credentials for the broker.
Regards,
Sara

Reply

Nicolas

July 15, 2022 at 2:58 am

Hola, muy buen tutorial. Quisiera saber como puedo enviar multiples datos con un solo publish. Por ejemplo, si tengo dos sensores, como deberia escribir la linea de código para enviar dos datos al tiempo?.

Gracias

Reply

Sara Santos

July 16, 2022 at 5:18 pm

Hi.
You can send a JSON string containing all the data, for example.
Regards,
Sara

Reply

Alexander

September 15, 2022 at 8:21 pm

Hey! It can’t handle special characters like ääö, how can I solve this? UTF-8?

Reply

James

November 14, 2022 at 10:54 pm

Hello. Thanks for the post

“new_message = client.check_msg()” does not save the variable for me.

I can see the text coming over just fine , but does not save to the variable:

```
while True:
try:
a = client.check_msg()
print(“a is”, a)
time.sleep(5)
except OSError as e:
restart_and_reconnect()
```

I just get:

a is None

Thanks!

Reply

Gary Kuipers

February 4, 2023 at 4:32 pm

Please provide guidance: ERROR: [Errno 104] ECONNRESET

I can ping the ESP32 from the laptop

The mosquitto broker is running (standard, no changes):
ps ax | grep mosqu
1043 ? Ss 0:00 /bin/sh /snap/mosquitto/776/launcher.sh
1320 ? S 0:01 /snap/mosquitto/776/usr/sbin/mosquitto -c /snap/mosquitto/776/default_config.conf
8230 pts/2 S+ 0:00 /snap/mosquitto/776/usr/bin/mosquitto_sub -h localhost -t to_sensors -v

Subscriptions work on the laptop:
mosquitto_sub -h localhost -t ‘to_sensors’ -v
to_sensors Hello from mosquitto_pub
to_sensors Hello from mosquitto_pub

obviously the publisher must work because the subscriber is receiving:
mosquitto_pub -h localhost -t ‘to_sensors’ -m ‘Hello from mosquitto_pub’

I modified the example code in this post to read:
try:
client.connect()
except Exception as e:
print(f”MQTT: connect_and_subscribe: client.connect(): ERROR: {e}”)
which is where we find the error:
MQTT: connect_and_subscribe: client.connect(): ERROR: [Errno 104] ECONNRESET

My phone can ping both the ESP32 and my laptop.

I don’t think it is the firewall:
sudo ufw status
Status: inactive

Help?

Reply

Sara Santos
February 5, 2023 at 7:09 pm

Hi.
Check if you allowed remote access to the broker:
https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/#mosquitto-no-authentication

Reply

Gary Kuipers
February 5, 2023 at 4:28 pm

HI Sara: I have been looking for an asyncio mqtt client. Any pointers to the best one?

Reply

Darwin von Corax

April 9, 2023 at 7:02 pm

Hi, Sara:
I've found a couple of bugs in umqttsimple.py. The first is that .check_msg() returns the return value of .wait_msg(), but .wait_msg never returns the payload message. The second is that, for whatever reason, when I call .publish() my broker never receives the payload.

I've patched .wait_msg() by adding 'return msg' at line 202, but I can't see why .publish() isn't working.

Reply

Sara Santos

April 13, 2023 at 8:24 pm

Thanks for providing that information.
Regards,
Sara

Reply

Ozgur Karaaslan

May 1, 2023 at 3:56 pm

If you take error with esp8266, you can add parameter “keepalive=60” while initilization of MQTTClient() class.

Like this: client = MQTTClient([..other parameters..] , keepalive=60)

Reply

Leave a Comment

Name *

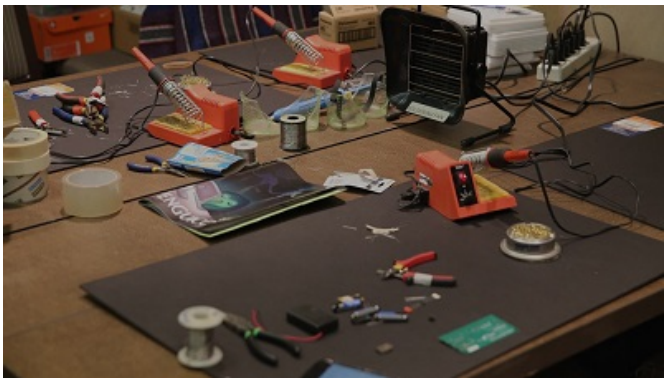
Email *

Website

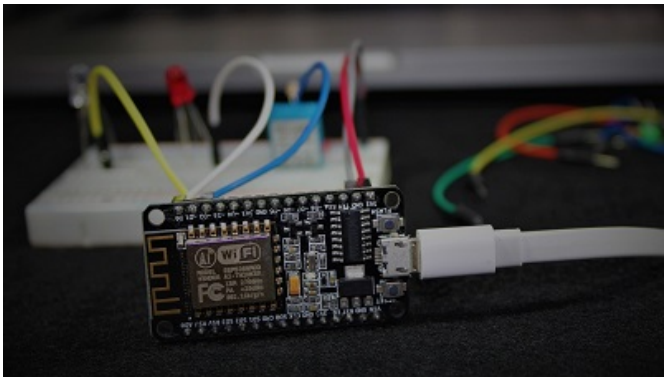
☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

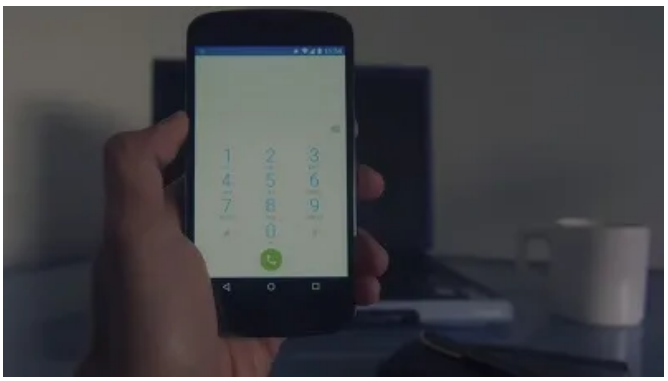
Post Comment



Visit Maker Advisor – Tools and Gear for makers, hobbyists and DIYers »



Home Automation using ESP8266 eBook and video course » Build IoT and home automation projects.



Build Web Servers with ESP32 and ESP8266 » boards to control outputs and monitor sensors remotely.