

TUTORIAL 3: Setting up Zigbee2Mqtt

Last revision: March 28th, 2022

1 INTRODUCTION & MOTIVATION

Many pervasive computing project must be able to monitor sensors and control actuators. These include movement sensors (PIR) and control devices such as LED light strips and power plugs. In order to communicate with these devices, you need to choose a communication standard which will allow you access to thousands of sensors. Zigbee is one of the most popular platforms for wireless sensor networks. This can be controlled in various ways, but one of them is with the Zigbee2Mqtt (Z2M) [1], which is an open-source application that interfaces Zigbee based networks for home automation and translates its events to MQTT.

In this tutorial you will learn how to setup Z2M in a Raspberry Pi and interact with it.

2 WHAT TO READ BEFORE THE TUTORIAL?

It is expected that you have a basic knowledge of the following topics:

- What is the Zigbee standard [2, 3].

3 MATERIALS

You'll need the following materials:

- Zigbee USB adapter
- Raspberry Pi 4
- HDMI to mini-HDMI cable
- Power supply
- Keyboard, mouse and monitor (any wired keyboard and mouse should work)

4 INSTALLING Z2M IN THE RASPBERRY PI

The installation of Z2M in Linux is made in two stages: 1) install the Zigbee USB adapter and 2) install Z2M. The second stage can be done before the second stage, but Z2M will not start, because the adapter is not installed. If you have any problem with stage 1, then refer to [1] and [4] for more detailed information and support.

4.1 Installing the Zigbee USB adapter

The installation of Z2M will be done using the terminal of the Raspberry Pi OS (RPiOS). Start by connecting the Zigbee USB adapter to the Raspberry Pi and check if it is detected by executing the command `lsusb`. This command lists all USB devices connected to the

Raspberry. One of the listed devices should be a “Texas Instruments, Inc.”, as depicted in Figure 1.

```
pi@cep2-rpi4:/opt/zigbee2mqtt $ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 0451:16a8 Texas Instruments, Inc.
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@cep2-rpi4:/opt/zigbee2mqtt $
```

Figure 1. List of USB devices connected to the Raspberry Pi. Marked is the Zigbee USB adapter.

This device is detected as a serial device, so Linux will create a serial port where communication with the device will be performed. To know which serial port the device is, execute the command `ls -l /dev/serial/by-id`. A device with the prefix `usb-Texas_Instruments_TI_CC2531_USB_CDC` should be listed as depicted in Figure 2. This is a symbolic link (or shortcut) for the actual serial device. In the end of the line (in yellow) you can see that the device `ttyACM0` is listed. This is the serial port that refers to the USB adapter and, if you try to list it (`ls -l /dev/ttyACM0`), it should be listed in the `/dev` directory.

```
pi@cep2-rpi4:/opt/zigbee2mqtt $ ls -l /dev/serial/by-id
total 0
lrwxrwxrwx 1 root root 13 Feb  2 12:55 usb-Texas_Instruments_TI_CC2531_USB_CDC___0X00124B001CCE304E-if00 -> ../../ttyACM0
pi@cep2-rpi4:/opt/zigbee2mqtt $ ls -l /dev/ttyACM0
crw-rw---- 1 root dialout 166, 0 Feb  2 13:25 /dev/ttyACM0
pi@cep2-rpi4:/opt/zigbee2mqtt $
```

Figure 2. List of serial ports detected by RPiOS.

4.2 Installing and running Z2M

Z2M is a Node.js application. Therefore, Z2M installation will be done in two stages: 1) install Node.js and 2) actually install Z2M.

To install Node.js, execute the following commands (comments are prefixed with a ‘#’ in grey):

```
# Setup Node.js repository in the packet manager
sudo curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
# Install Node.js and its dependencies
sudo apt-get install -y nodejs npm git make g++ gcc

# Verify that the correct nodejs and npm are installed
node --version # Should output v12.X or v10.X
npm --version  # Should output 7.X
```

In Figure 3 it is depicted the output of the node and npm commands.

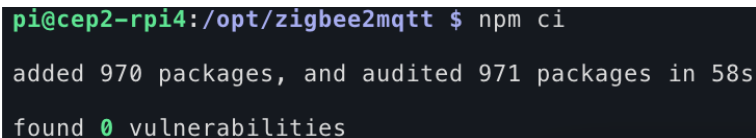
```
pi@cep2-rpi4:~ $ node --version
v12.22.5
pi@cep2-rpi4:~ $ npm --version
7.5.2
```

Figure 3. output of the node and npm commands.

If Node.js is successfully installed, then you can proceed to install Z2M using the following commands:

```
# Clone Z2M repository to the directory /opt/zigbee2mqtt. This is the
directory where it will be installed.
sudo git clone https://github.com/Koenkk/zigbee2mqtt.git /opt/zigbee2mqtt
# Change ownership of the directory /opt/zigbee2mqtt to user pi (user and
group).
sudo chown -R pi:pi /opt/zigbee2mqtt
# Install dependencies (as user "pi", which was changed in the previous
step).
cd /opt/zigbee2mqtt
# Build the application using Node.js building tools.
npm ci
```

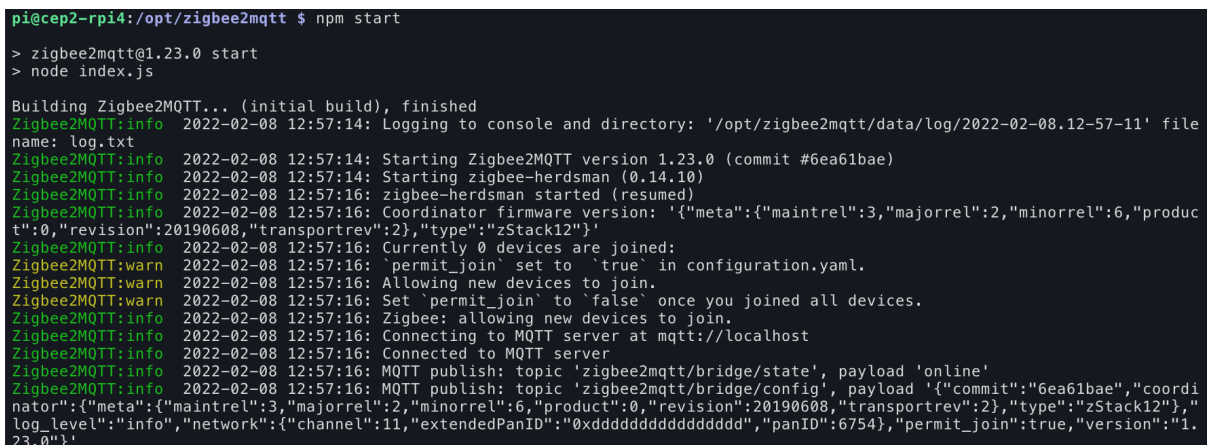
The output of the last command should be similar to what is depicted in Figure 4. This also produces a lot of warnings which can be ignored (they are related with building Node.js' serialport package).



```
pi@cep2-rpi4:/opt/zigbee2mqtt $ npm ci
added 970 packages, and audited 971 packages in 58s
found 0 vulnerabilities
```

Figure 4. Output of the command `npm ci -production`

Once Z2M is installed, you can run it. To do so, you must be in the directory where Z2M is installed (if you followed the given steps correctly, you should already be there). To start it, execute the command `npm start`. Figure 5 depicts the truncated output of Z2M once it is started. Note that when you run it the first time, it will take a bit more time to start, because it will build the application (see the line "Building Zigbee2MQTT... (initial build)").



```
pi@cep2-rpi4:/opt/zigbee2mqtt $ npm start
> zigbee2mqtt@1.23.0 start
> node index.js

Building Zigbee2MQTT... (initial build), finished
Zigbee2MQTT:info 2022-02-08 12:57:14: Logging to console and directory: '/opt/zigbee2mqtt/data/log/2022-02-08.12-57-11' file
name: log.txt
Zigbee2MQTT:info 2022-02-08 12:57:14: Starting Zigbee2MQTT version 1.23.0 (commit #6ea61bae)
Zigbee2MQTT:info 2022-02-08 12:57:14: Starting zigbee-herdsman (0.14.10)
Zigbee2MQTT:info 2022-02-08 12:57:16: zigbee-herdsman started (resumed)
Zigbee2MQTT:info 2022-02-08 12:57:16: Coordinator firmware version: '{"meta":{"maintrel":3,"majorrel":2,"minorrel":6,"product":0,"revision":20190608,"transportrev":2},"type":"zStack12"}'
Zigbee2MQTT:info 2022-02-08 12:57:16: Currently 0 devices are joined:
Zigbee2MQTT:warn 2022-02-08 12:57:16: 'permit_join' set to 'true' in configuration.yaml.
Zigbee2MQTT:warn 2022-02-08 12:57:16: Allowing new devices to join.
Zigbee2MQTT:warn 2022-02-08 12:57:16: Set 'permit_join' to 'false' once you joined all devices.
Zigbee2MQTT:info 2022-02-08 12:57:16: Zigbee: allowing new devices to join.
Zigbee2MQTT:info 2022-02-08 12:57:16: Connecting to MQTT server at mqtt://localhost
Zigbee2MQTT:info 2022-02-08 12:57:16: Connected to MQTT server
Zigbee2MQTT:info 2022-02-08 12:57:16: MQTT publish: topic 'zigbee2mqtt/bridge/state', payload 'online'
Zigbee2MQTT:info 2022-02-08 12:57:16: MQTT publish: topic 'zigbee2mqtt/bridge/config', payload '{"commit":"6ea61bae","coordinator":{"meta":{"maintrel":3,"majorrel":2,"minorrel":6,"product":0,"revision":20190608,"transportrev":2},"type":"zStack12"},"log_level":"info","network":{"channel":11,"extendedPanID":"0xddddddd","panID":6754},"permit_join":true,"version":"1.23.0"}'
```

Figure 5. Output of Z2M on start.

To stop the process (or kill it) you can press `Ctrl+C`. Figure 6 depicts Z2M's log output when it stops.

```

^CZigbee2MQTT:info 2022-02-09 10:03:58: MQTT publish: topic 'zigbee2mqtt/bridge/state', payload 'offline'
Zigbee2MQTT:info 2022-02-09 10:03:58: Disconnecting from MQTT server
Zigbee2MQTT:info 2022-02-09 10:03:58: Stopping zigbee-herdsman...
Zigbee2MQTT:info 2022-02-09 10:03:59: Stopped zigbee-herdsman
Zigbee2MQTT:info 2022-02-09 10:03:59: Stopped Zigbee2MQTT
pi@cep2-rpi4:/opt/zigbee2mqtt $

```

Figure 6. Output of Z2M when it is stopped.

4.3 Interact with Z2M

In the last tutorial you learned what MQTT is and how to use the Mosquitto clients to publish and subscribe to/from other MQTT clients. Here you will use them to interact with Z2M, which is a publisher and subscriber.

The default Z2M's top level topic is zigbee2mqtt. You can use mosquitto_sub to subscribe to all events with this topic to see the events published by Z2M. You can use the following command to subscribe¹:

```
mosquitto_sub -h localhost -t "zigbee2mqtt/#" -F "[%t] %p"
```

Note that the subscribed topic is zigbee2mqtt/#, which indicates that all topics with the top level zigbee2mqtt are subscribed. A sample (and truncated) output of running the subscriber command above is depicted in Figure 7. As you can see, the subscriber received messages with the topics zigbee2mqtt/bridge/state and zigbee2mqtt/bridge/info.

```

pi@cep2-rpi4:~ $ mosquitto_sub -h localhost -t "zigbee2mqtt/#" -F "[%t] %p"
[zigbee2mqtt/bridge/state] online ← Sample topic and message
[zigbee2mqtt/bridge/info] {"commit":"99274c7","config":{"advanced":{"adapter_conc
e,"ext_pan_id":[221,221,221,221,221,221,221,221],"homeassistant_discovery_topic":"
,"log_syslog":{},"pan_id":6754,"report":false,"soft_reset_timeout":0,"timestamp_for
a3e0"},"font":{"coordinator":"#ffffff","enddevice":"#000000","router":"#ffffff"},"
config_schema":{"definitions":{"device":{"properties":{"debounce":{"description":"
type":"array"},"filtered_attributes":{"description":"Allows to prevent certain attr
name","type":"string"},"optimistic":{"description":"Publish optimistic state after
:"Sets the MQTT Message Expiry in seconds, Make sure to set mqtt.version to 5","ti
p":{"properties":{"devices":{"items":{"type":"string"},"type":"array"},"filtered_a
"description":"Adapter concurrency (e.g. 2 for CC2531 or 16 for CC26X2R1) (default
y_blocklist":{"description":"Prevent devices from being checked for availability",
ription":"Availability timeout in seconds When enabled, devices will be checked if
date for serial port, default: 115200 for Z-Stack, 38400 for Deconz"},"examples":[3
t":{"default":true,"description":"Persist cached state, only used when Cache state

```

Figure 7. Output of subscribing to the top level topic zigbee2mqtt, using mosquitto_sub.

As previously stated, Z2M is also a subscriber, i.e. it subscribes to topics that can be used to monitor or configure it. This feature is what enables other applications to interact with Z2M. One example is the topic zigbee2mqtt/bridge/request/health_check, which can be used to check the current state of Z2M. The response to this request is published to the topic zigbee2mqtt/bridge/response/health_check².


¹ The -F argument is used to format the output of the Z2M messages. In this case, it will print a string with the topic in square brackets, followed by the message payload. This argument can be omitted, but it can be helpful to aid read the received messages.

² The response to the topics zigbee2mqtt/bridge/request/+ is always sent to the topic zigbee2mqtt/bridge/response/+ (see [5]).

The following commands can be used to, respectively, publish and subscribe to Z2M's health_check topics:

```
mosquitto_pub -h localhost -t zigbee2mqtt/bridge/request/health_check -m ""
mosquitto_sub -h localhost -t zigbee2mqtt/bridge/response/health_check
```

In Figure 8 it is depicted the use of the commands above, specifically publish and subscribe to the health_check topics. It is also depicted Z2M's log output.



The screenshot shows a terminal window with three distinct sections. The top section shows the command `mosquitto_pub -h localhost -t zigbee2mqtt/bridge/request/health_check -m ""` being executed, with an orange arrow pointing to it from the label "Publisher". The middle section shows the command `mosquitto_sub -h localhost -t zigbee2mqtt/bridge/response/health_check` being executed, with an orange arrow pointing to it from the label "Subscriber". The bottom section shows the Z2M log output: `Zigbee2MQTT:info 2021-02-02 17:44:18: MQTT publish: topic 'zigbee2mqtt/bridge/response/health_check', payload '{"data":{"healthy":true},"status":"ok"}'`, with an orange arrow pointing to it from the label "Log output of Z2M for the health_check topic".

Figure 8. Using the Mosquitto clients to interact with Z2M's health_check topics. On top, the publisher, middle the subscriber and on the bottom the log output of Z2M.

5 NEXT STEPS

You should know have Z2M installed and running in your Raspberry Pi. A lot of aspects were not covered here, but you should consider them for your project according to your requisites. All of the topics described next can be consulted in Z2M's documentation page [1].

5.1 Running Z2M in other platforms

Z2M also run on other platforms such as Windows and Docker. You can find more information on how to install them in the following links:

- Windows: <https://www.zigbee2mqtt.io/information/windows.html>
- Docker: <https://www.zigbee2mqtt.io/information/docker.html>

5.2 Configuring Z2M

As shown, the default configuration of Z2M is enough for running it. But other options are available, which might have to be changed in order to adapt Z2M to the requirements of your project. Some examples that you might look into are MQTT configurations (username and password for connecting to the MQTT broker), change the serial port where the USB adapter is connected, or change the Zigbee network name and key. These options are documented in [6].

5.3 Running Z2M as a Linux service

With the current installation, you'll have to start Z2M manually every time you boot the Raspberry Pi. This can be avoided if you run Z2M as a Linux service. Information on how to do it can be consulted in [7].

5.4 Topics

In section 4.3 some examples of Z2M topics were presented. A list of all the topics where Z2M publishes/subscribes is presented in [8], as well as its messages content (tip: all communications are in JSON). This page will be especially helpful when Zigbee devices are paired and you must check which topics you must subscribe.

5.5 Python clients

In section 4.3 you used the Mosquitto clients to interact with Z2M. Using the Python examples of tutorial 2, how would you modify them in order to communicate with Z2M? You can start by creating two separate clients (a publisher and a subscriber) and then try to merge them into a single application. Tip: start by the subscriber and see if you can receive messages from Z2M.

6 FURTHER READING

You should further explore the Zigbee2Mqtt's documentation [1] in order to gain a deeper understanding of its capabilities and how they fit your project requirements.

You can also start exploring what devices are supported (this will be address in detail in the next tutorial) and see how they report their events [9].

Finally, you can see a practical implementation of the Zigbee protocol in the IKEA Trådfri system and its integration with the Google Home [10]. This reference should give you a better understanding of the application of the Zigbee for home automation (or domotics) and may help inspire you to design your system.

7 REFERENCES

[1] Zigbee2Mqtt documentation: <https://www.zigbee2mqtt.io>

[2] Zigbee: A complete beginners breakdown: <https://www.smarthomebit.com/a-beginners-guide-to-zigbee/>

[3] Zigbee2Mqtt's Zigbee network: https://www.zigbee2mqtt.io/information/zigbee_network.html

[4] Zigbee2Mqtt Github: <https://github.com/koenkk/zigbee2mqtt>

[5] MQTT bridge requests: https://www.zigbee2mqtt.io/guide/usage/mqtt_topics_and_messages.html#zigbee2mqtt-bridge-request

[6] Zigbee2Mqtt's configuration: <https://www.zigbee2mqtt.io/guide/configuration/>

[7] Running Zigbee2Mqtt as a Linux service: https://www.zigbee2mqtt.io/guide/installation/01_linux.html#optional-running-as-a-daemon-with-systemctl

[8] Zigbee2Mqtt's MQTT topics and message structure:

https://www.zigbee2mqtt.io/guide/usage/mqtt_topics_and_messages.html

[9] Zigbee2Mqtt's supported devices: <https://www.zigbee2mqtt.io/supported-devices/>

[10] Complete Ikea Tradfri + Google Home Guide:

<https://www.smarthomebit.com/complete-ikea-tradfri-google-home-guide/>