Microhomie Documentation

Microhomie Team

Jan 28, 2019

Contents

A MicroPython implementation of Homie, a lightweight MQTT convention for the IoT.

Currently Microhomie implements Homie v3.0.1.

Latest release: v0.3.1

This project is in beta stage.

CHAPTER 1

Known issues

- SSL connection problems at least with ESP8266
- In lost of Wifi connection, there is a chance MQTT qos==1 will hang forever

CHAPTER 2

Install

You can get the detailed installation instructions here: http://microhomie.readthedocs.io/

CHAPTER 3

Local Development setup

You have to compile micropython with this guide https://github.com/micropython/micropython/wiki/Getting-Started

After that, you can install the required libraries.

```
micropython -m upip install micropython-umqtt.simple
micropython -m upip install micropython-logging
micropython -m upip install micropython-machine
```

CHAPTER 4

Base Installation

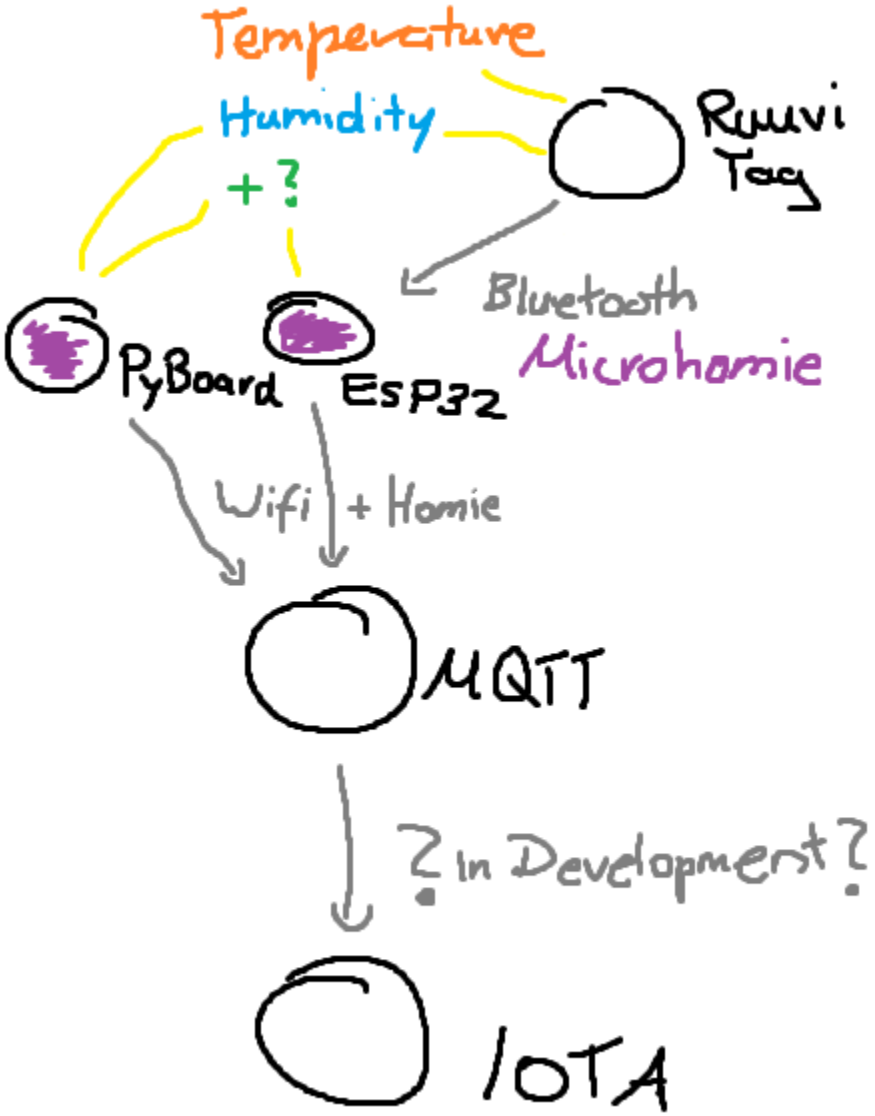
4.1 Quickstart

4.1.1 Overview

This project consists of 3 main components:

- the Microhomie sensor (hardware and software)
- the MQTT server (selfhosted or you can use a service, see below)
- the homie -> IOTA gateway (selfhosted)

First, take a look at this awesome architectural drawing:



This guide will take care of the purple ESP32 / Microhomie device with the requried hardware and software.

4.1.2 Install the MicroPython

For now, follow this tutorial (https://www.cnx-software.com/2017/10/16/esp32-micropython-tutorials/) to install MicroPython on the ESP. Please remember to use the latest version of the ESP software during the installation.

Important: On some boards, the installation of MicroPython will fail with an "connection timeout" if you have any wires attached to the board. This depends on your board and you have to detach all wires except power for the basic installation to work. This is only relevant for the installation of MicroPython and not Microhomie.

Once the installation is successful and you are able to execute python code on the ESP32, you can install Microhomie.

4.1.3 Install Microhomie

Microhomie is split in two parts, the base package and a node package with additional functions. We require the DHT22 node from the node package for this tutorial.

We provide some example nodes in the https://github.com/microhomie/micropython-homie-nodes repository. Most of these nodes can be used out of the box to publish data.

The base package and the node package can be installed via PyPi or manual. As of the time of this writing, the PyPi way of installing is prefered.

4.1.4 Setup WIFI for installation

Microhomie handles WIFI setup for you, but for installation from PyPi you have to manual setup WIFI once from REPL.

```
>>> import network
>>> wlan = network.WLAN(network.STA_IF)
>>> wlan.active(True)
>>> wlan.connect('wifi-name', 'wifi-secret')
# wait a few seconds
>>> wlan.isconnected()  # test if wlan is connected
True
>>> wlan.ifconfig()  # get wlan interface config
('192.168.42.2', '255.255.255.0', '192.168.42.1', '192.168.42.1')
```

4.1.5 Configuration

Microhomie use a `settings.py` file to configure the device. See `settings.example.py` as an example. Modify this file for your needs and copy it to your device root directory as `settings.py`.

4.1.6 Install with PyPi

We provide PyPi packages for easier installation on your device. Open the REPL from your device, make sure your device wlan is up and your device has access to the internet, import upip and install microhomie:

```
>>> import upip
>>> upip.install('microhomie')
```

4.1.7 Add files specific for your device

You now have Micropython adn Microhomie installed. The stage is set!

Please copy a new *main.py* file on the device:

```
import utime
import settings

from homie.node.dht22 import DHT22
from homie import HomieDevice


```

(continues on next page)

(continued from previous page)

```
# Homie device setup
homie_device = HomieDevice(settings)
homie_device.add_node(DHT22(interval=60, pin=2))
homie_device.publish_properties()

while True:
    # Push the new data to MQTT
    homie_device.publish_data()
    # Sleep a little bit
    utime.sleep(1)
```

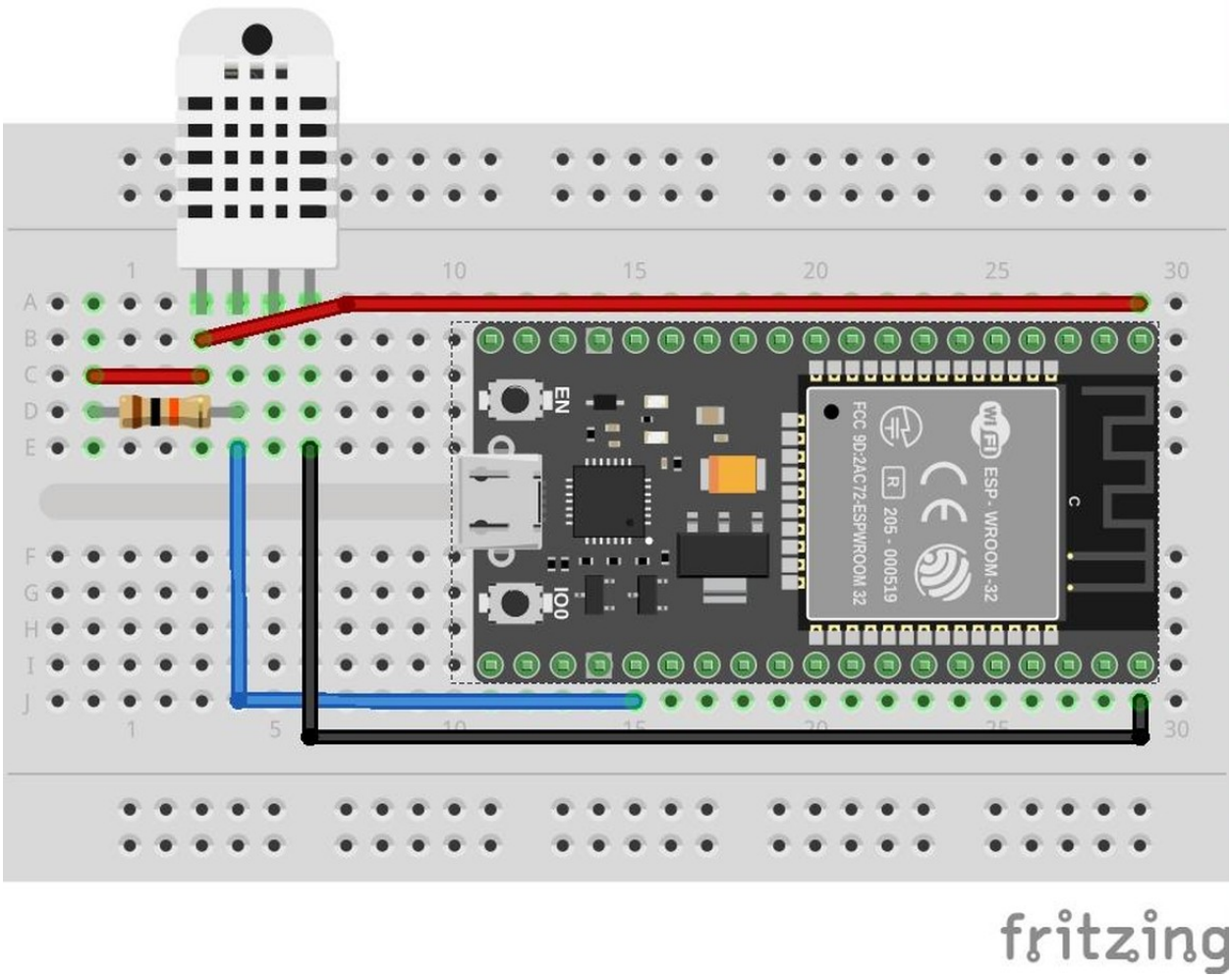
4.1.8 Soldering it

You need the following hardware:

- ESP32 NodeMCU (Amazon referral link de: http://amzn.to/2En655Q / en: http://amzn.to/2GUmdKz)
- 10 k Ohm resistor
- DHT22 (Amazon referral link de: http://amzn.to/2FU84f4 / en: http://amzn.to/2FUg0wY)
- Wires

You don't have to get the stuff on Amazon. Other places are cheaper, Amazon is just easier and for general reference.

Wire it all up:



The final result will look something like this. This includes the DHT22 for temperature and humidity AND the SDS011 for airquality which is not part of this quickstart but can be added later on as you can see:

4.1.9 Prepare MQTT

Checkout mosquitto MQTT Server (https://mosquitto.org/) if you want to host a server yourself or head over to IO Adafruit(https://io.adafruit.com/), createn an account and use their MQTT API(https://learn.adafruit.com/adafruit-io/mqtt-api).

If everything is setup correctly, the data will then be pushed to the MQTT server in the homie format. Take a look at the homie documentation(https://github.com/marvinroger/homie) to get an idea of the possibilities.

```
homie/686f6d6965/temperature/$type → temperature
homie/686f6d6965/temperature/$name → Bedroom Temperature Node
homie/686f6d6965/temperature/$properties → degrees
homie/686f6d6965/temperature/degrees/$settable → false
homie/686f6d6965/temperature/degrees/$unit → C
homie/686f6d6965/temperature/degrees/$datatype → float
homie/686f6d6965/temperature/degrees/$name → Bedroom Temperature
homie/686f6d6965/temperature/degrees/$format → -20.0:60
homie/686f6d6965/temperature/degrees → 12.07
```

(continues on next page)

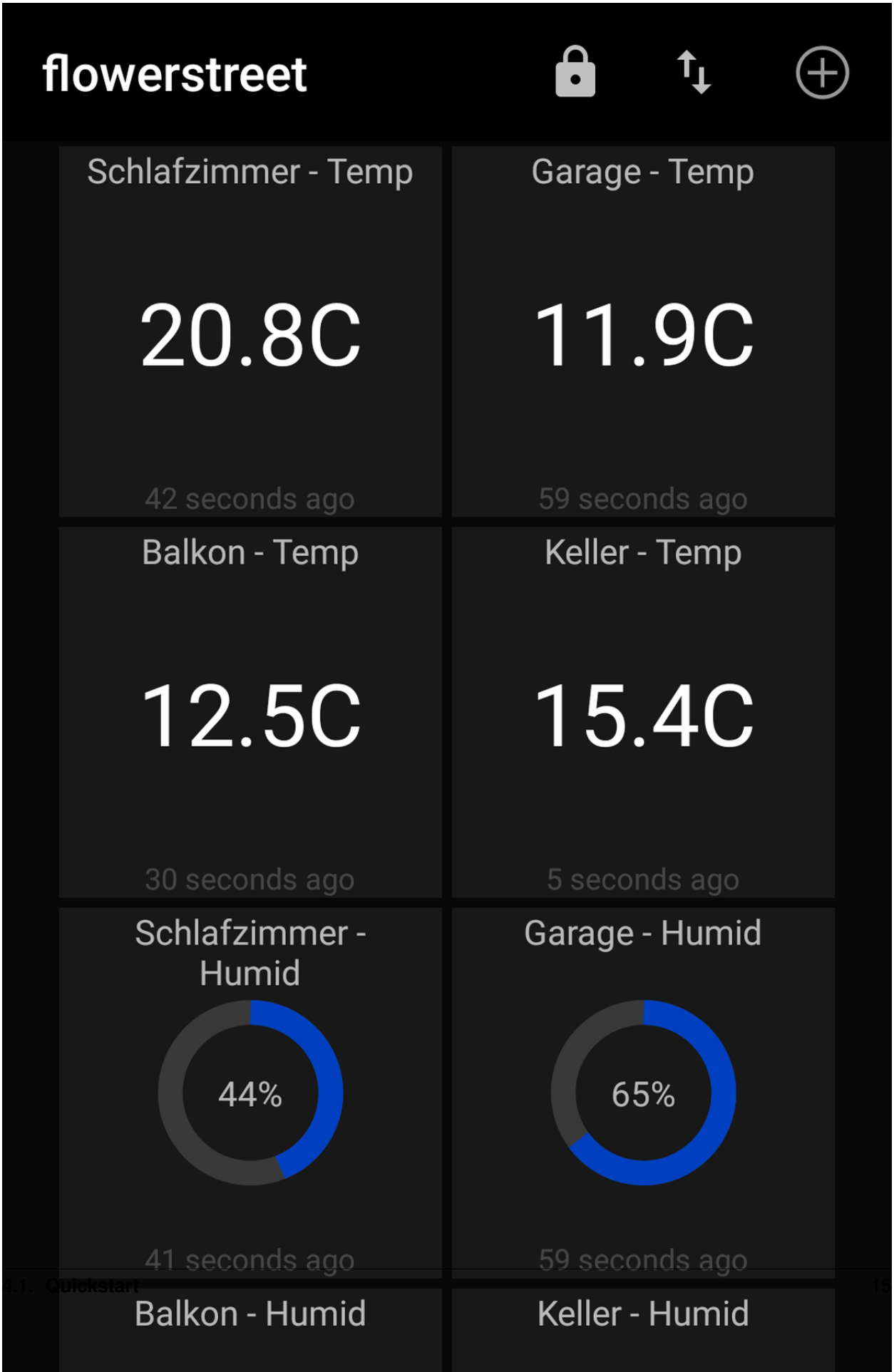
(continued from previous page)

```
homie/686f6d6965/humidity/$type → humidity
homie/686f6d6965/humidity/$name → Bedroom Humidity Node
homie/686f6d6965/humidity/$properties → percentage
homie/686f6d6965/humidity/percentage/$settable → false
homie/686f6d6965/humidity/percentage/$unit → %
homie/686f6d6965/humidity/percentage/$datatype → integer
homie/686f6d6965/humidity/percentage/$name → Bedroom Humidity
homie/686f6d6965/humidity/percentage/$format → 0:100
homie/686f6d6965/humidity/percentage → 79
```

4.1.10 Mobile App

You can chose from multiple Apps for Android / iOS / Desktop to display the sensor data. You have to subscribe to the topics you are interested in, for example *homie/686f6d6965/humidity/percentage* and *homie/686f6d6965/temperature/degrees*.

- MQTT Dash: https://play.google.com/store/apps/details?id=net.routix.mqttdash


flowerstreet
Schlafzimmer - Temp
20.8C
42 seconds ago
Garage - Temp
11.9C
59 seconds ago
Balkon - Temp
12.5C
30 seconds ago
Keller - Temp
15.4C
5 seconds ago
Schlafzimmer - Humid
44%
41 seconds ago
Garage - Humid
65%
59 seconds ago
Balkon - Humid
Keller - Humid

4.1.11 Send Data to IOTA with the homie-iota-gateway

This is still the great unknown. The good news is, the data can be gathered with Microhomie in the Homie format (or by other systems!) and its waiting in MQTT.

We already have boilerplate code in place, which subscribes in real time to updates of the data in MQTT and it can be configured. We just need the "push to IOTA" part when the data marketplace is public!

You can follow the final development on Github https://github.com/microhomie/homie-iota

4.2 Setup WIFI for installation

Microhomie handles WIFI setup for you, but for installation from PyPi you have to manual setup WIFI once from REPL.

```
>>> import network
>>> wlan = network.WLAN(network.STA_IF)
>>> wlan.active(True)
>>> wlan.connect('wifi-name', 'wifi-secret')
# wait a few seconds
>>> wlan.isconnected()  # test if wlan is connected
True
>>> wlan.ifconfig()  # get wlan interface config
('192.168.42.2', '255.255.255.0', '192.168.42.1', '192.168.42.1')
```

4.3 Configuration

Microhomie use a `settings.py` file to configure the device. See `settings.example.py` as an example. Modify this file for your needs and copy it to your device root directory as `settings.py`.

4.4 Install manually

Use your favorite MicroPython remote shell like rshell, mpfshell or ampy to copy Microhomie to your device.

Create a directory `homie` on your device `lib` directory and copy the file `__init__.py` from the `homie` directory. Then create a `node` directory in `homie` and copy `__init__.py`, `led.py`, `simple.py` from the `homie/node` directory to the device.

Your file system structure should now look similar like this:

```
├── lib
|   ├── homie
|   |   ├── __init__.py
|   |   ├── node
|   |   |   ├── __init__.py
|   |   |   ├── led.py
|   |   |   ├── simple.py
├── boot.py
├── main.py
```

4.5 Install with mpfshell (experimental)

With mpfshell you can execute our mpfshell-script `install.mpf` to install Microhomie on your device. Clone this repository and run:

```
mpfshell ttyUSB0 -s install.mpf
```

4.6 Install with PyPi

We provide PyPi packages for easier installation on your device. Open the REPL from your device, make sure your device wlan is up and your device has access to the internet, import upip and install microhomie:

```
>>> import upip
>>> upip.install('microhomie')
```

4.7 Install with Wizzard (experimental)

CHAPTER 5

Node Installation

5.1 Add a node

We provide some example nodes in the microhomie-nodes repository. Most of these nodes can be used out of the box to publish data. If you want to use a DHT22 sensor in example, copy the files `__init__.py` and `dht22.py` from `homie/node` to the `lib/homie/node` directory on your device. In the `dht22.py` file you see an example `main.py` as docstring. Copy this example to `main.py` on your device and on next reset it starts publishing temperature and humidity. In this example the DHT22 sensor is wired to GPIO PIN 4, on ESP8266 this is PIN D2.

You can also install nodes from PyPi:

```
>>> import upip
>>> upip.install('microhomie-nodes-dht22')
```

CHAPTER 6

Examples

6.1 Examples

Please find multiple examples in the `examples` folder.

6.1.1 Example: ESP8266 example device

In this example we use the on-board LED from the ESP8266. Copy the `example-led.py` file from the `examples` diretory to your device and rename it to `main.py`.

You can now connect a MQTT client to your MQTT Broker an listen to the `homie/#` topic, or whatever you set as base topic.

Reset your ESP8266 and watch incoming MQTT messages.

The on-board LED status is reversed to the pin status. On start the on-board LED is on. To turn it off send *on* or *toggle* via MQTT. Replace `<DEVICEID>` with the ID from the MQTT topic:

```
$ mosquitto_pub -t 'homie/<DEVICEID>/led/power/set' -m on
$ mosquitto_pub -t 'homie/<DEVICEID>/led/power/set' -m off
$ mosquitto_pub -t 'homie/<DEVICEID>/led/power/set' -m toggle
```

6.1.2 Example: Simple node

In most cases you write your own node classes. But if you just want to test publishing or have a simple use case, you can use the `SimpleHomieNode` class. The `SimpleHomieNode` does not provide all homie properties, but can be used as a fast start, when you don't want to write anything in a class:

```
import utime
import settings

from homie.node.simple import SimpleHomieNode
```

(continues on next page)

(continued from previous page)

```
from homie.device import HomieDevice


homie_device = HomieDevice(settings)

n = SimpleHomieNode(node_type=b'dummy', node_property=b'value', interval=5)
n.value = 17

homie_device.add_node(n)
homie_device.publish_properties()

while True:
    homie_device.publish_data()
    n.value = utime.time()
    print(n)
    utime.sleep(1)
```