
I suggest to use https://esphome.io/components/bluetooth_proxy.html instead. In my opinion its better solution now.

bt-mqtt-gateway

A simple Python script which provides a Bluetooth to MQTT gateway, easily extensible via custom workers.

See Wiki for more information.

Features

- Highly extensible via custom workers
- Data publication via MQTT
- Configurable topic and payload
- MQTT authentication support
- Systemd service
- Reliable and intuitive
- Tested on Raspberry Pi Zero W

Supported devices

- EQ3 Bluetooth smart thermostat via python-eq3bt
- Xiaomi Mi Scale
- Xiaomi Mi Scale v2 (Body Composition Scale)
- Linak Desk via linak_bt_desk
- MySensors
- Xiaomi Mi Flora plant sensor via miflora
- Xiaomi Aqara thermometer via mithermometer
- Bluetooth Low Power devices (BLE)
- Oral-B connected toothbrushes
- Switchbot
- Sensirion SmartGadget via python-smartgadget
- RuuviTag via ruuvitag-sensor
- Xiaomi Mijia 2nd gen, aka LYWSD02

Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes. See deployment for notes on how to deploy the project on a live system.

Requirements

- Python 3 installed
- Working bluetooth adapter (might be built in like in raspberry pi)
- Working mqtt server, to which you can connect

Testing mqtt: Use mosquitto_sub to print all messages. Change localhost to your mqtt server address.

```
1 # also helpful for testing MQTT messages
2 mosquitto_sub -h localhost -d -t '#'
3
4 # if user/password are defined on your mosquitto server, use
5 mosquitto_sub -h localhost -d -t '#' -u user -P password
6
7 # for more info, see mosquitto_sub --help
```

Installation

Docker

There are prebuilt docker images at <https://hub.docker.com/r/zewelor/bt-mqtt-gateway/tags>. Thanks @hobbypunk90 and @krasnoukhov for docker work.

Mount config.yaml as /application/config.yaml volume

Example exec

```
1 docker run -d --name bt-mqtt-gateway --network=host --cap-add=NET_ADMIN
  --cap-add=NET_RAW -v $PWD/config.yaml:/application/config.yaml
  zewelor/bt-mqtt-gateway
```

Docker-compose See docker-compose.yml file in repo. To run:

```
1 docker-compose run bt-mqtt-gateway
```

Virtualenv

On a modern Linux system, just a few steps are needed to get the gateway working. The following example shows the installation under Debian/Raspbian:

```
1 sudo apt-get install git python3 python3-pip python3-wheel bluetooth
   bluez libglib2.0-dev
2 sudo pip3 install virtualenv
3 git clone https://github.com/zewelor/bt-mqtt-gateway.git
4 cd bt-mqtt-gateway
5 virtualenv -p python3 .venv
6 source .venv/bin/activate
7 pip3 install -r requirements.txt
```

All needed python libs, per each worker, should be auto installed on run. If not you can install them manually:

```
1 pip3 install `./gateway.py -r configured`
```

Configuration

All worker configuration is done in the file `config.yaml`. Be sure to change all options for your needs. This file needs to be created first:

```
1 cp config.yaml.example config.yaml
2 nano config.yaml
3 source .venv/bin/activate
4 sudo ./gateway.py
```

Attention: You need to add at least one worker to your configuration. Scan for available Bluetooth devices in your proximity with the command:

```
1 sudo hcitool lscan
```

Execution

A test run is as easy as:

```
1 source .venv/bin/activate
2 sudo ./gateway.py
```

Debug output can be displayed using the `-d` argument:

```
1 sudo ./gateway.py -d
```

Deployment

Continuous background execution can be done using the example Systemd service unit provided.

```
1 sudo cp bt-mqtt-gateway.service /etc/systemd/system/
2 sudo nano /etc/systemd/system/bt-mqtt-gateway.service (modify path of
  bt-mqtt-gateway)
3 sudo systemctl daemon-reload
4 sudo systemctl start bt-mqtt-gateway
5 sudo systemctl status bt-mqtt-gateway
6 sudo systemctl enable bt-mqtt-gateway
```

Attention: You need to define the absolute path of `service.sh` in `bt-mqtt-gateway.service`.

Dynamically Changing the Update Interval To dynamically change the `update_interval` of a worker, publish a message containing the new interval in seconds at the `update_interval` topic. Note that the `update_interval` will revert back to the value in `config.yaml` when the gateway is restarted. I.E:

```
1 # Set a new update interval of 3 minutes
2 mosquitto_pub -h localhost -t 'miflora/update_interval' -m '150'
3 # Set a new update interval of 30 seconds
4 mosquitto_pub -h localhost -t 'mithermometer/update_interval' -m '30'
```

Custom worker development

Create custom worker in workers directory.

Example simple worker

```
1 from mqtt import MqttMessage
2 from workers.base import BaseWorker
3
4 REQUIREMENTS = ['pip_packages']
5
6 class TimeWorker(BaseWorker):
7     def _setup(self):
8         self._some = 'variable'
9
10    def status_update(self):
11        from datetime import datetime
12
13        return [MqttMessage(topic=self.format_topic('time'), payload=
            datetime.now())]
```

REQUIREMENTS add required pip packages, they will be installed on first run. Remember to import them in method, not on top of the file, because on initialization, that package won't exist. Unless installed outside of the gateway. Check `status_update` method

`_setup` method - add / declare needed variables.

`status_update` method - It will be called using specified `update_interval`

Example config entry

Add config to the example config:

```
1   timeworker:
2     args:
3       topic_prefix: cool_time_worker
4       update_interval: 1800
```

Variables set in `args` section will be set as object attributes in `BaseWorker.__init__`

`topic_prefix`, if specified, will be added to each mqtt message. Alongside with `global_prefix` set for gateway

Troubleshooting

See the Troubleshooting Wiki

Built With

- Python - The high-level programming language for general-purpose programming

Authors

- **zewelor** - *Initial work*
- **bbbenji** - *Minor contributions*
- **elviosebastianelli** - *BLEscanmulti*
- **jumping2000** - *BLEscan*
- **AS137430** - *Switchbot*

License

This project is licensed under the MIT License - see the LICENSE.md file for details