pyzigbee Documentation

Release 0.1.2

Pierre ROTH

June 10, 2015

Contents

Version: 0.1.2

CHAPTER 1

Introduction

pyzigbee is a python library that enables to talk to Legrand zigbee devices through gateways.

Supported gateways are:

- 0 883 28: USB dongle

This library works has been tested on **Windows** and **Linux** with **Python 2.7, 3.3 and 3.4**.

CHAPTER 2

Installation

If you only want to install the library:

```
python setup.py install
```

or with python installer:

```
pip install .
```

Use the python installer to install other dependencies that may be needed to run tests or build documentation:

```
pip install -r requirements.txt
```

CHAPTER 3

Specific setup instructions

Before playing with real hardware, you may need to run some manual setup. Depending on your hardware the steps may be different. This section maintains instructions for the supported gateways

3.1 088328 USB/Zigbee dongle

3.1.1 driver

This dongle embeds a CP2102 USB to UART transceiver. This driver is included in latest *Linux* kernels. For *Windows* users, install the driver first:

- for Windows 32 bits
- for Windows 64 bits

3.1.2 configuration

- Open or create a Zigbee network on a device (NETW led blinking)
- Press the button on the 088328 USB key (NETW led should start blinking slowly)
- Press the NETW button on the device which has open the network
- All the NETW leds should turn off excepted the one of the device which created the network

Note: For gateways relying on a serial line drive (such as 088328), you can list the available ports of your system by running: *python -m serial.tools.list_ports*

CHAPTER 4

Code example

Here is a simple example of pyzigbee library usage (see: *pyzigbee/examples/088328/scan.py*)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function
from contextlib import closing
from pyzigbee.gateways.factory import GatewayFactory


def scan():
    """
    Scan the network looking for zigbee devices and print the found IDs
    """
    # Note that you may need to pass a configuration file to
    # the GatewayFactory since your low level device may be different
    # from the default one. Please read the 'Configuration' section
    # of the documentation
    gateway = GatewayFactory().create_gateway(ref="088328")

    with closing(gateway.open()) as gateway:
        ids = gateway.scan()

    print("Zigbee devices on the network:", ids)

if __name__ == '__main__':
    scan()
```

You may need to pass a configuration file (see: *Configuration*) to the factory to be override some driver arguments. Change gateway creation as following:

```
gateway = GatewayFactory(conf_filename="my_conf.json").create_gateway(ref="088328")
```

The **gateway** is the abstraction you will deal with to talk to zigbee devices.

CHAPTER 5

Shell

When installing the pyzgbee library, a shell script called **pyzigbeesh** is also installed.

You can call it from command line:

```
pyzigbeesh
```

To display currenly supported arguments:

```
pyzigbeesh --help
```

For example, to activate logs:

```
pyzigbeesh -d 10
```

Warning: You may need to gain priviledges on your machine to access the underlying hardware (such as the serial com port). A *sudo pyzigbeesh* should do the trick.

5.1 Configuration

The pyzigbee library has a default configuration for each supported gateway. This configuration (mainly driver settings) may be overriden by a JSON config file that can be passed to the shell using the *–conf* option.

Here is an example of such a **conf.json** configuration file:

```
{
    "088328": {
        "driver": {
            "args": {
                "port": "/dev/ttyUSB0",
                "baudrate": "19200"
            }
        }
    }
}
```

This configuration file could be changed to the following to work on Windows machines:

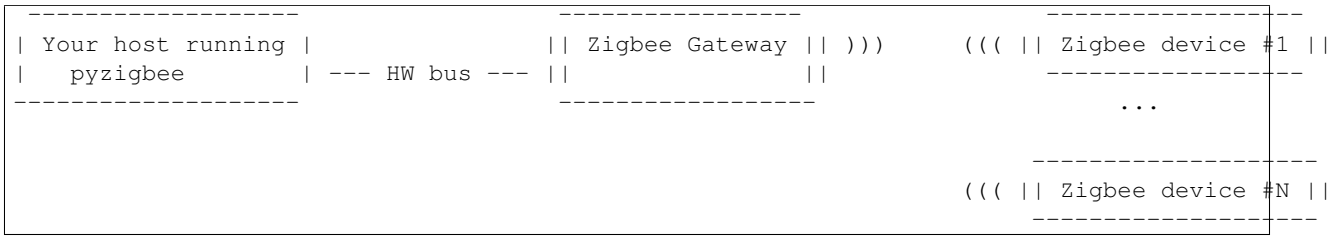
```
{
    "088328": {
        "driver": {
            "args": {
```

```
                "port": "COM1",
                "baudrate": "19200"
            }
        }
    }
}
```

CHAPTER 6

Architecture

6.1 The big picture



The HW (hardware) bus can be a serial line, a SPI line,... or whatever depending on the zigbee gateway. The same way, the protocol used to talk with this gateway over the HW bus can vary.

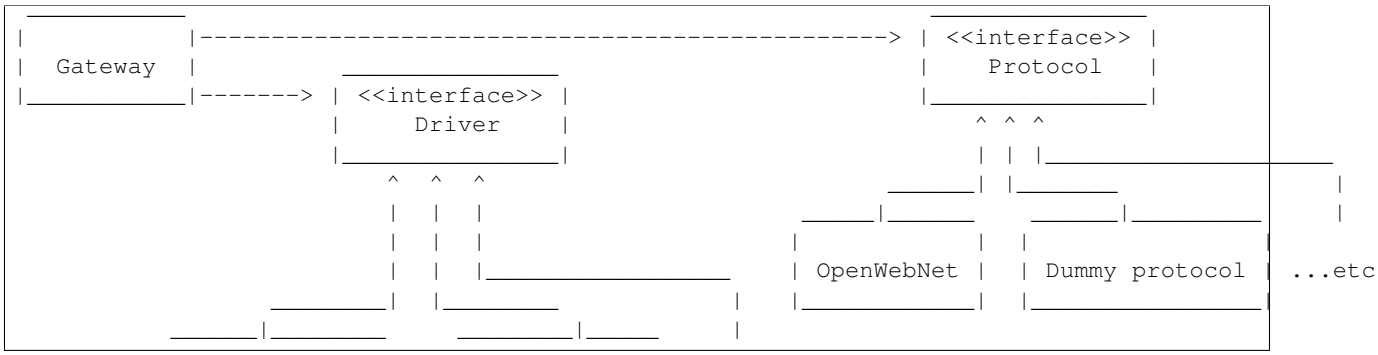
6.2 Some more details

To reflect reality, the library has the following classes:

- **Drivers**: deal with low level communication with the underlying hardware
- **Protocols**: deal with encoding and decoding frames for given protocols
- **Gateways**: relying on Drivers and Protocols, they provide an API to talk to Zigbee devices

6.2.1 UML diagram

The UML diagram should look like:



Note: The *dummy* classes are used for testing

The client code (application side) only relies on Gateway objects which are created by the *GatewayFactory*.

6.2.2 Interfaces

A *Gateway* is composed of two objects: a *Driver* and a *Protocol* which are interfaces. Real drivers and protocols must implement those interfaces.

class pyzigbee.drivers.basedriver.**BaseDriver**(***kwargs*)
Base driver inherited by all the drivers

class pyzigbee.protocols.baseprotocol.**BaseProtocol**
Base protocol inherited by all the protocols

The *Gateway* is the abstraction on which rely client code. You may need to inherit from and overload some methods to implement your own gateway.

class pyzigbee.gateways.gateway.**Gateway**(*driver*, *protocol*, *description=''*)
Gateways abstracts access to real devices

6.2.3 Some implementations

class pyzigbee.drivers.serialdriver.**SerialDriver**(***kwargs*)
Serial driver to communicate with underlying hardware

keyword args are: - port: the serial port such as /dev/tty3 or COM3 - baudrate: the serial line speed - parity: the serial line parity

class pyzigbee.protocols.openwebnet.**OWNProtocol**
OWN protocol is in charge of decoding/encoding OpenWebNet frames

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

p

B

G

O

P

S