Python Dynamixel Documentation

Release 2019.03.19

Vincent Poulailleau

May 30, 2022

CONTENTS:

CHAPTER

ONE

PYTHON DYNAMIXEL

Python package to use Dynamixel servos through a serial interface (unofficial, easier SDK).

1.1 Documentation

The full documentation can be read at https://pydxl.readthedocs.io.

Example code:

```
import time

from pydxl import Mx28, SerialLink

link = SerialLink(
    device="/dev/ttyUSB0", baudrate=1_000_000, protocol_version=1.0
)

servo = Mx28(identifier=1, serial_link=link)
servo.ping()
servo.led = True

servo.torque_enable = True
servo.goal_position = 2000
print(servo.goal_position)
time.sleep(3)
servo.goal_position = 1500
time.sleep(3)
servo.torque_enable = False
```

(continues on next page)

(continued from previous page)

```
link.close()
```

1.2 Features

- Use protocol 1.0 through a serial link, known to work with:
  - U2D2
  - USB2Dynamixel
  - DARwIn-OP (after having installed a more recent Python, tested with Python 3.7.2)
- Support servos:
  - MX-28
- TODO: implement protocol 2.0
- TODO: add more servo types

1.3 License

BSD 3-Clause license, feel free to contribute: https://pydxl.readthedocs.io/en/latest/contributing.html.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install Python Dynamixel, run this command in your terminal:

```
$ pip install pydxl
```

This is the preferred method to install Python Dynamixel, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Python Dynamixel can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/vpoulailleau/pydxl
```

Or download the tarball:

```
$ curl  -OL https://github.com/vpoulailleau/pydxl/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use Python Dynamixel in a project:

```
import pydxl
```

CHAPTER

FOUR

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/vpoulailleau/pydxl/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

Python Dynamixel could always use more documentation, whether as part of the official Python Dynamixel docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/vpoulailleau/pydxl/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *pydxl* for local development.

1. Fork the *pydxl* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pydxl.git
```

3. Install your local copy into a virtualenv. Set up your fork for local development:

```
$ python3 -m venv venv
$ source venv/bin/activate
$ cd pydxl/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass whatalinter and the tests, including testing other Python versions with tox:

```
$ flake8 pydxl tests (TODO use whatalinter from python-dev-tools)
$ py.test
$ tox
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, and 3.7. Check https://travis-ci.org/vpoulailleau/pydxl/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_pydxl
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Run:

```
$ ./deploy.sh
```

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Vincent Poulailleau <vpoulailleau@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

HISTORY

6.1 2019.3.19

- Full support of MX-28 servo with protocol 1.0

6.2 2019.3.17 (2019-03-18)

- First release on PyPI.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search