
AxiBot Documentation

Release 0.0.3.dev

Scott Torborg

Mar 02, 2019

Contents

1	Contents	3
1.1	Quick Start	3
1.2	Command Line Usage	3
1.3	API Reference	4
1.4	Contributing	5
2	Indices and Tables	7
	Python Module Index	9

AxiBot is an alternative control interface for the [AxiDraw pen plotting robot](#) from Evil Mad Scientist Labs.

The design goals are:

- Maintainable, hackable, codebase with light dependencies and the ability to run on a wide variety of embedded platforms.
- Excellent motion control, tuned to eliminate undesirable plotting artifacts.
- Easy command-line and programmatic use.

1.1 Quick Start

1.1.1 Install

The recommended installation method is `pip`:

```
$ pip install axibot
```

You can also download a source package here [directly from the Python Package Index](#).

AxiBot can be used via the command line or Python code. Tasks like real-time control or plotting from sources other than SVG will generally require writing Python code.

1.1.2 Plot a File

A number of example files are included in the source distribution. With a typical rollerball pen and a sheet of US Letter paper (8.5" x 11") you can plot the world map example:

```
$ axibot plot examples/worldmap.svg
```

1.2 Command Line Usage

The main way to use AxiBot is the `axibot` command-line utility. Some examples are shown here. You can also see the utility itself for more info:

```
$ axibot -h
```

1.2.1 Manual Control

Issue single manual commands:

```
$ axibot manual pen_up 1000
$ axibot manual disable_motors
```

Enter a shell to use the same commands:

```
$ axibot manual
(axibot) pen_down 1000
(axibot) xy_move 400 400 100
```

1.2.2 File Estimation

Print info about the motion plan that would be used to plot an SVG file:

```
$ axibot info examples/worldmap.svg
```

1.2.3 Plotting

Actually plot an SVG file:

```
$ axibot plot examples/worldmap.svg
```

By default, this will use an interface interface to prompt certain user actions.

1.2.4 Web Server

To start a webserver for remote control of the AxiDraw:

```
$ axibot server
```

1.3 API Reference

```
class axibot.action.ABMove (da, db, duration)
```

```
    name = u'ab_move'
```

```
    time()
```

```
class axibot.action.Move
```

```
class axibot.action.PenDownMove (delay)
```

```
    name = u'pen_down'
```

```
    time()
```

```
class axibot.action.PenUpMove (delay)
```



```
    name = u'pen_up'
    time()
class axibot.action.XYAccelMove(dx, dy, v_initial, v_final)

    name = u'xy_accel_move'
    time()
class axibot.action.XYMove(m1, m2, duration)

    name = u'xy_move'
    time()
```

1.4 Contributing

Patches and suggestions are strongly encouraged! GitHub pull requests are preferred, but other mechanisms of feedback are welcome.

A test suite is included. To run the tests, simply run in the top level of the repo:

```
$ tox
```

This will also ensure that the Sphinx documentation builds correctly, and that there are no [PEP8](#) or [Pyflakes](#) warnings in the codebase.

Any pull requests should preserve all of these things.

CHAPTER 2

Indices and Tables

- `genindex`
- `modindex`

a

`axibot.action`, 4
`axibot.config`, 4

A

ABMove (class in axibot.action), 4
axibot.action (module), 4
axibot.config (module), 4

M

Move (class in axibot.action), 4

N

name (axibot.action.ABMove attribute), 4
name (axibot.action.PenDownMove attribute), 4
name (axibot.action.PenUpMove attribute), 4
name (axibot.action.XYAccelMove attribute), 5
name (axibot.action.XYMove attribute), 5

P

PenDownMove (class in axibot.action), 4
PenUpMove (class in axibot.action), 4

T

time() (axibot.action.ABMove method), 4
time() (axibot.action.PenDownMove method), 4
time() (axibot.action.PenUpMove method), 5
time() (axibot.action.XYAccelMove method), 5
time() (axibot.action.XYMove method), 5

X

XYAccelMove (class in axibot.action), 5
XYMove (class in axibot.action), 5