CircuitPython Kernel Documentation

Release 0.2.0.dev

Carol Willing

Jun 14, 2018

Contents

Contents:

CHAPTER 1

CircuitPython Kernel

CircuitPython Kernel enables CircuitPython to be used in Jupyter Notebooks to teach Python coding with microcontrollers.

- Free software: BSD license
- Documentation: https://circuitpython-kernel.readthedocs.io.

Warning: This package is a pre-alpha version. Use at your own peril. It is missing functionality, has hard coding of variables, and has only been used with MacOS 10.12 and Atmel SAMD.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

This package is inspired by Thomas Kluyver's ubit_kernel for the BBC Micro:bit and Project Jupyter's wrapper kernel.

CHAPTER 2

Installation

2.1 Stable release

To install CircuitPython Kernel, run this command in your terminal:

```
$ pip3 install circuitpython_kernel
$ python3 -m circuitpython_kernel.install
```

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

If you don’t have **‘pip3‘_** installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for CircuitPython Kernel can be downloaded from the GitHub repo.

You can either clone the public repository:

```
$ git clone git://github.com/willingc/circuitpython_kernel
```

Or download the tarball:

```
$ curl  -OL https://github.com/willingc/circuitpython_kernel/tarball/master
```

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

```
$ python3 setup.py install
$ python3 -m circuitpython_kernel.install
```

CHAPTER 3

Usage

To use CircuitPython Kernel in a Jupyter Notebook, you must install and activate the kernel specification (kernelspec) for Jupyter:

```
$ python3 circuitpython_kernel.install
```

You can check that CircuitPython has been activated for Jupyter:

```
$ jupyter kernelspec list
```

Removing CircuitPython from Jupyter's kernel specification list:

```
$ jupyter kernelspec remove circuitpython_kernel
```

CHAPTER 4

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/willingc/circuitpython_kernel/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

CircuitPython Kernel could always use more documentation, whether as part of the official CircuitPython Kernel 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/willingc/circuitpython_kernel/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 *circuitpython_kernel* for local development.

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

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

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv circuitpython_kernel
   $ cd circuitpython_kernel/
   $ 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 flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 circuitpython_kernel tests
   $ python setup.py test or py.test
   $ tox
   ```

   To get flake8 and tox, just pip install them into your virtualenv.

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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/willingc/circuitpython_kernel/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_circuitpython_kernel
```

CHAPTER 5

Credits

5.1 Development Lead

- Carol Willing <carolcode@willingconsulting.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2017-03-23)

- First release on PyPI.

CHAPTER 7

Board Preparation

7.1 Adafruit Feather M0 Express

7.1.1 Add CircuitPython firmware

- Download CircuitPython firmware.
- Plug in board and double click the **reset** button.
- Drag and drop the *.uf2 CircuitPython file to the USB drive
- You should see the `CIRCUITPY` as the new name of the USB drive

7.2 Adafruit Feather M0 SAMD21

7.2.1 Add CircuitPython firmware

- Download CircuitPython firmware.
- Go to the BOSSA GitHub releases page and download the latest release.
- Plug in board and double click the **reset** button.
- Copy firmware into the same directory as BOSSA tool.
- Enter `bossac -e -w -v -R -p PORT_NAME firmware.bin`

7.2.2 Access the REPL

Use `screen` program:

```
screen /dev/cu.usb1421 115200
```

7.3 Adafruit Feather Huzzah ESP8266

7.3.1 Add CircuitPython firmware

- python2 -m pip install esptool
- Download CircuitPython firmware
- Install the SiLabs CP210x driver
- Erase flash `python2 esptool.py --port /path/to/ESP8266 erase_flash`
- Load firmware: `esptool.py --port /path/to/ESP8266 --baud 460800 write_flash --flash_size=detect 0 firmware.bin`
- Press reset or unplug/plug the board.

7.3.2 Access the REPL

Use `screen` program:

```
screen <device> 115200
```

7.4 ampy

- Install ampy `python3 -m pip install adafruit-ampy`
- To get options for listing files and moving files: `ampy --help`

CHAPTER 8

Indices and tables

- genindex
- modindex
- search