PiView Documentation

Release 2.0.3

Adrian Gould

Jun 02, 2021

CONTENTS:

PiView

![https://pypi.python.org/pypi/piview{}](https://img.shields.io/pypi/v/piview.svg) ![https://travis-ci.com/AdyGCode/piview{}](https://img.shields.io/travis/AdyGCode/piview.svg) ![https://piview.readthedocs.io/en/latest/?version=latest{}](https://readthedocs.org/projects/piview/badge/?version=latest)

A Raspberry Pi System Information Package

PiView provides the details of the Raspberry Pi currently being interrogated.

General Information

- Free software: Open Software License ("OSL") v. 3.0
- Documentation: https://piview.readthedocs.io.

Features

PiView provides system information including, but not limited to:

Group | Information |

|:--------:|———————————————————————————| | CPU | max load across cores, temperature, clock speed | | GPU | temperature | | HARDWARE | bluetooth, i2c, spi, camera statuses | | HOST | boot time, model, name, revision, serial number, uptime | | NETWORK | host name, interface names, ip addresses, mac addresses | | STORAGE | total disk capacity, free disk capacity, total RAM and free RAM |

Also includes a small utility library with:

- conversion of bytes into Kilobytes, Megabytes, Gigabytes and up
- create list with a quartet of integer numbers representing the IPv4 Address

Requirements

This project requires the following package(s):

- *psutils*

Remaining packages are Python 'built-ins'.

Credits

A very large thank you to Matt Hawkins upon whose code this package is based: https://www.raspberrypi-spy.co.uk/

The original code may be found at https://github.com/tdamdouni/Raspberry-Pi-DIY-Projects/blob/master/MattHawkinsUK-rpispy-misc/python/mypi.py

Thank you to Sander Huijsen for his contributions and guidance in all things Python.

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

[Cookiecutter: https://github.com/audreyr/cookiecutter{}](https://github.com/audreyr/cookiecutter) [Cookiecutter PyPackage: https://github.com/audreyr/cookiecutter-pypackage{}](https://github.com/audreyr/cookiecutter-pypackage)

Copyright

CHAPTER

ONE

INSTALLATION

1.1 Stable release

To install PiView, run this command in your terminal:

```
$ pip install piview
```

This is the preferred method to install PiView, 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.

1.2 From sources

The sources for PiView can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/AdyGCode/piview
```

Or download the tarball:

```
$ curl -OJL https://github.com/AdyGCode/piview/tarball/master
```

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

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

CHAPTER

TWO

USAGE

To use PiView in a project:

```
import piview
```

Import the required components:

```
from piview.CPU import CPU
from piview.GPU import GPU
from piview.Hardware import Hardware
from piview.Host import Host
from piview.Network import Network
from piview.Storage import Storage
```

Access the component methods as required:

```
# obtain the status of the i2c channel
hw_i2c = Hardware.i2c()
```

CHAPTER

THREE

PIVIEW

3.1 piview package

3.1.1 Submodules

3.1.2 piview.CPU module

class piview.CPU.**CPU**

Bases: object

static max_load(*random=False*)

This function returns the maximum "CPU load" across all CPU cores, or -1 if no value determined.

Providing a random=True parameter value will return a random value if the actual CPU load can't be determined.

Parameters **random** – boolean default False

Return type float

Returns The maximum CPU Load in range 0-100

static speed()

Get the CPU frequency using the vcgencmd on Linux based systems

If frequency cannot be determined, returns -1

Return type integer

Returns The CPU frequency in MHz

static temperature()

Requests the CPU temperature from the vcgencmd returning the result to the caller as a floating point value to 2DP

If no value can be determined, uses -999.99 as the returned "error" value.

Return type float

Returns The CPU temperature in degrees Celsius

static temperature_k()

Provide temperature of the CPU in degrees Kelvin

Returns CPU Temperature in Kelvin

3.1.3 piview.GPU module

class piview.GPU.**GPU**
Bases: object

static temperature()
Requests the GPU temperature from the thermal zone details

Return type string

Returns GPU temperature in degrees Celsius to 2DP

3.1.4 piview.Hardware module

class piview.Hardware.**Hardware**
Bases: object

static bt()
Check if Bluetooth module is enabled

Return type boolean

Returns True|False

static camera()
Check if camera is enabled and present

Return type dictionary

Returns Details in form {"supported": boolean, "detected": boolean}

static i2c()
Check if I2C bus is enabled by checking for i2c_bcm2 modules

Return type boolean

Returns True|False

static spi()
Check if SPI bus is enabled by checking for spi_bcm2 modules

Return type boolean

Returns True|False

3.1.5 piview.Host module

class piview.Host.**Host**
Bases: object

static boot_time()
Determines the time the device was started

Return type datetime

Returns How long ago the Pi was booted

static model()
Provide Pi Model Details

Extracts the details from the device tree model file

Return type string

Returns The model name and other identifying details

`static name()`
Provides the host name to the user

Return type string

Returns The host name of the Pi

`static python()`
Get current Python version

Return type string

Returns A string containing the version of python that is being used

`static revision()`
Provide board revision details

The details are extracted from the cpu info file

Return type string

Returns The revision number of the Pi motherboard

`static serial()`
Provide the Serial Number of the Pi CPU

The details are extracted from the cpu info file

Return type string

Returns The Pi's serial number

`static uptime()`
Determines the amount of time the device has been running for in seconds

Return type float

Returns The time that the Pi has been 'up' for

3.1.6 piview.Network module

`class piview.Network.Network`
Bases: `object`

`static eth_name(_type=None)`
Provide the Ethernet interface name

Parameters **_type** – string, possible options are: enx or eth

Return type string

Returns The network interface name

`static host_name()`
Provide the host name to the user

Return type string

Returns The host name of the Pi

static ip(*interface='eth0'*)
Provide IP Address from the named interface

Default is eth0

Uses the ifconfig command to create a text file, then processes this file to obtain the IP address.

Parameters **interface** – string, the interface to obtain the IP address for

Return type string

Returns IPv4 address of the selected interface

static mac(*interface='eth0'*)
Provides the hardware MAC address for the interface requested

Default is eth0

Parameters **interface** – string, the interface name to query

Return type string

Returns Mac address of the selected interface

3.1.7 piview.Storage module

class piview.Storage.**Storage**
Bases: object

static all_discs()
Provide the user with the storage space (Total, Free) for each disc attached to the Pi as a dictionary.

The dictionary will have: "disc name" : (total storage, free storage) for each disc.

Return disc stats dictionary of tuples

static all_disks()
Alias for all_discs()

see all_discs()

static disc()
Provide the total disc space and the disc space that is free

Return type tuple

Returns (Total, Free)

static disk()
Alias for disc()

see disc()

static ram()
Provide the total RAM and free RAM to the user as a tuple

Return type tuple

Returns (Total, Free)

3.1.8 piview.Utils module

class piview.Utils.**Utils**
Bases: object

static draw_line(*characters='-=-'*, *length=40*)
Draw a line of characters using a given character and length

Parameters

- **characters** – string, the character(s) to draw with
- **length** – integer), the length of the line

Return type string

Returns A string of exactly length characters

static format_bytes(*size=0*, *style=None*)
Formats the given value into Bytes, Kilobytes, Megabytes, ...

Using Byte shorthand by default - B, KB, MB, ...

The style may be:

- None | short | s – Short labels
- long | l – Long labels

If style is anything other than above, then defaults to long format.

Parameters

- **size** – integer, defaults to 0
- **style** – string, defaults to None

Return type tuple

Returns (float, string)

static random_percentage(*min_percentage=0*, *max_percentage=100*)
This function returns a random percentage. Useful for simulations when developing monitoring dashboards

Parameters

- **min_percentage** – float, minimum value to return, default 0.0
- **max_percentage** – float, maximum to return, default 100.0

Return type float

Returns A random CPU load value between 0% and 100% to 1DP

3.1.9 piview.piview module

Main module.

3.1.10 Module contents

Top-level package for PiView.

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/AdyGCode/piview/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/AdyGCode/piview/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 :)

Get Started!

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

1. Fork the *piview* repo on GitHub.
2. **Clone your fork locally:** `shell $ git clone git@github.com:your_name_here/piview.git `
3. **Install your local copy into a virtualenv. Assuming you have *virtualenvwrapper* installed, this is how you set up your fork fo**
   ```shell

   $ mkvirtualenv piview $ cd piview/ $ python setup.py develop ```
4. **Create a branch for local development:** `shell $ 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:

   `shell $ flake8 piview tests $ python setup.py test or pytest $ tox `

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

6. **Commit your changes and push your branch to GitHub:** `shell $ 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.

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.md.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/AdyGCode/piview/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

`shell $ python -m unittest tests.test_piview `

Deploying

A reminder for the maintainers on how to deploy.

Make sure all your changes are committed (including an entry in HISTORY.md).

`shell $ bump2version patch # possible:  major / minor / patch $ git push $ git push --tags `

Travis will then deploy to PyPI if tests pass.

Credits

Development Lead

- Adrian Gould <adrian.gould@nmtafe.wa.edu>

Contributors

- Sander Huijsen <...>

History

3.0.0 (TBA)

This will contain some 'breaking' changes.

- All results will be returned as dictionary with reading type and value
- Recreate the testing methods
- Update online documentation
- Deprecate the PiView_AG edition
- Code contains list of TODO elements - showing improvements / new features

2.0.3 (2021-05-30)

- Minor edits to code.
- CPU speed returns random value if parameter *random=True* used in call, or -1 when parameter *random=False* (or omitted)
- Temperatures return -273.16ºC when no reading given (absolute zero)
- Keeping documentation edits within this version until next minor update to code

2.0.2 (2021-05-28)

- Minor Fixes

2.0.1 (2021-05-27)

- Updated Package release.

1.0.0 Initial Release Some enhancements to come - such as return all the attached storage statistics

Added:

- python version (HOST)
- camera supported / detected (HARDWARE)

0.5.0 RAM, Storage, Host

Added the following:

- ram total and free
- storage total and free for 'all disks' in total
- name to Host, this is a temporary version until further investigation done, use the host name method in the network section to get host name

To do:

- statistics (total space, free) for each attached storage device

0.4.0 Network Features

The following are implemented in this version:

- host name
- interface names
- ip addresses
- mac addresses

Fixed missing self references in classes, removed *get* from function names Added missing file headers…

0.3.0 Host Features

The following have been implemented:

- boot time
- model
- serial number
- uptime
- revision

0.2.0 Hardware Features

Added Hardware checking for:

- SPI
- I2C
- BT

Updated [[README]] Design and added PiView Icon

0.1.1 GPU Features

Added:

- GPU temperature

##0.1.0 CPU Features

Added:

- max load across cores
- processor temperature
- processor clock speed

0.0.3 Setup fixes

Small fixes to setup.cfg, and a source reformat.

0.0.2 Utils

Added Utils to the package. Utils includes:

- format_bytes
- draw_line

0.0.1 Initial Version

Blank project, containing:

- starter folder structure
- [[README.md]]
- [[CHANGES.md]]
- [[LICENSE]]

CHAPTER

FOUR

API REFERENCE

This page contains auto-generated API reference documentation[1].

4.1 piview

Top-level package for PiView.

4.1.1 Submodules

piview.CPU

Module Contents

Classes

CPU

class piview.CPU.**CPU**

static speed()
Get the CPU frequency using the vcgencmd on Linux based systems

If frequency cannot be determined, returns -1

Return type integer

Returns The CPU frequency in MHz

static max_load(*random=False*)
This function returns the maximum "CPU load" across all CPU cores, or -1 if no value determined.

Providing a random=True parameter value will return a random value if the actual CPU load can't be determined.

Parameters **random** – boolean default False

Return type float

[1] Created with sphinx-autoapi

Returns The maximum CPU Load in range 0-100

static temperature()
Requests the CPU temperature from the vcgencmd returning the result to the caller as a floating point value to 2DP

If no value can be determined, uses -999.99 as the returned "error" value.

Return type float

Returns The CPU temperature in degrees Celsius

static temperature_k()
Provide temperature of the CPU in degrees Kelvin

Returns CPU Temperature in Kelvin

piview.GPU

Module Contents

Classes

GPU

class piview.GPU.**GPU**

static temperature()
Requests the GPU temperature from the thermal zone details

Return type string

Returns GPU temperature in degrees Celsius to 2DP

piview.Hardware

Module Contents

Classes

Hardware

class piview.Hardware.**Hardware**

static bt()
Check if Bluetooth module is enabled

Return type boolean

Returns True|False

static spi()
Check if SPI bus is enabled by checking for spi_bcm2 modules

Return type boolean

Returns True|False

static i2c()
Check if I2C bus is enabled by checking for i2c_bcm2 modules

Return type boolean

Returns True|False

static camera()
Check if camera is enabled and present

Return type dictionary

Returns Details in form {"supported": boolean, "detected": boolean}

piview.Host

Module Contents

Classes

Host

class piview.Host.Host

static boot_time()
Determines the time the device was started

Return type datetime

Returns How long ago the Pi was booted

static model()
Provide Pi Model Details

Extracts the details from the device tree model file

Return type string

Returns The model name and other identifying details

static name()
Provides the host name to the user

Return type string

Returns The host name of the Pi

static python()
Get current Python version

Return type string

Returns A string containing the version of python that is being used

static revision()
Provide board revision details

The details are extracted from the cpu info file

Return type string

Returns The revision number of the Pi motherboard

static serial()
Provide the Serial Number of the Pi CPU

The details are extracted from the cpu info file

Return type string

Returns The Pi's serial number

static uptime()
Determines the amount of time the device has been running for in seconds

Return type float

Returns The time that the Pi has been 'up' for

piview.Network

Module Contents

Classes

Network

class piview.Network.Network

static host_name()
Provide the host name to the user

Return type string

Returns The host name of the Pi

static eth_name(*_type=None*)
Provide the Ethernet interface name

Parameters **_type** – string, possible options are: enx or eth

Return type string

Returns The network interface name

static mac(*interface='eth0'*)
Provides the hardware MAC address for the interface requested

Default is eth0

Parameters **interface** – string, the interface name to query

Return type string

Returns Mac address of the selected interface

static ip(*interface='eth0'*)
Provide IP Address from the named interface

Default is eth0

Uses the ifconfig command to create a text file, then processes this file to obtain the IP address.

Parameters **interface** – string, the interface to obtain the IP address for

Return type string

Returns IPv4 address of the selected interface

piview.Storage

Module Contents

Classes

Storage

class piview.Storage.**Storage**

static ram()
Provide the total RAM and free RAM to the user as a tuple

Return type tuple

Returns (Total, Free)

static disc()
Provide the total disc space and the disc space that is free

Return type tuple

Returns (Total, Free)

static all_discs()
Provide the user with the storage space (Total, Free) for each disc attached to the Pi as a dictionary.

The dictionary will have: "disc name" : (total storage, free storage) for each disc.

Return disc stats dictionary of tuples

static disk()
Alias for disc()

see disc()

static all_disks()
Alias for all_discs()

see all_discs()

piview.Utils

Module Contents

Classes

Utils

class piview.Utils.**Utils**

static draw_line(*characters='-=-'*, *length=40*)
Draw a line of characters using a given character and length

Parameters

- **characters** – string, the character(s) to draw with
- **length** – integer), the length of the line

Return type string

Returns A string of exactly length characters

static format_bytes(*size=0*, *style=None*)
Formats the given value into Bytes, Kilobytes, Megabytes, ...

Using Byte shorthand by default - B, KB, MB, ...

The style may be:

- None | short | s – Short labels
- long | l – Long labels

If style is anything other than above, then defaults to long format.

Parameters

- **size** – integer, defaults to 0
- **style** – string, defaults to None

Return type tuple

Returns (float, string)

static random_percentage(*min_percentage=0*, *max_percentage=100*)
This function returns a random percentage. Useful for simulations when developing monitoring dashboards

Parameters

- **min_percentage** – float, minimum value to return, default 0.0
- **max_percentage** – float, maximum to return, default 100.0

Return type float

Returns A random CPU load value between 0% and 100% to 1DP

piview.piview

Main module.

4.1.2 Package Contents

piview.__author__ = Adrian Gould

piview.__email__ = adrian.gould@nmtafe.wa.edu.au

piview.__version__ = 2.0.3

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

Symbols

A

B

C

D

E

F

G

H

I

M

N

P

R

S

T

U