Picroscopy Documentation

Release 0.1

Dave Hughes

Oct 27, 2017

Contents

Picroscopy is a small Python web-application which is intended for usage with a Raspberry Pi as a microscopy solution. With the Raspberry Pi Camera mounted on a microscope, the Raspberry Pi provides a live video feed to its monitor via HDMI, while another machine can be used to control the setup via a web-based interface.

The project is written in Python 3 and is open-sourced under the GPL license. Packages can be downloaded from the project homepage. The source code can be obtained from GitHub. The documentation can be read on ReadTheDocs.

CHAPTER 1

Table of Contents

Installation

To install the application from PyPI into an existing Raspbian image, first ensure you install the pre-requisites:

```
$ sudo apt-get install build-essential python3-dev python-virtualenv \
  libjpeg8-dev libtiff5-dev libfreetype6-dev zlib1g-dev
```

Next, create a virtual Python 3 environment from which to run Picroscopy (this avoids having to install anything as root which makes it easier to remove later should you wish to start over):

```
$ virtualenv -p python3 picroscopyenv
$ source picroscopyenv/bin/activate
```

Finally, install Picroscopy within the environment:

```
$ easy_install picroscopy
```

If you wish to install the documentation building dependencies also:

```
$ easy_install picroscopy[doc]
```

Warning: If you install picroscopy with the optional `[doc]` specifier (which installs the dependencies required to build the documentation), the installation will take an *extremely* long time to build on the Pi.

Sphinx and docutils, which are used for the documentation, take nearly an hour to build; this seems to have something to do with their use of `2to3` to attain Python 3 compatibility.

You should now be able to run picroscopy with the included configuration file like so:

```
$ picroscopy -c picroscopy.ini
```

Development

The system relies on several third party libraries including Pillow, a fork of the Python Imaging Library (PIL) which requires some compilation. On Raspbian, first ensure you install all the pre-requisites:

```
$ sudo apt-get install build-essential exuberant-ctags python3-dev \
    python-virtualenv libjpeg8-dev libtiff5-dev libfreetype6-dev zlib1g-dev
```

Next, create a virtual python environment from which to run Picroscopy and activate it:

```
$ virtualenv -p python3 picroscopyenv
$ source picroscopyenv/bin/activate
```

Finally, grab a copy of the Picroscopy source from GitHub and install it into the virtualenv (the example below uses the `develop` target for development purposes; you can use `install` if you just wish to install Picroscopy without any of the extra development stuff like tags):

```
$ git clone https://github.com/waveform80/picroscopy.git
$ cd picroscopy
$ make develop
```

You should now be able to run picroscopy with the included configuration file like so:

```
$ picroscopy -c picroscopy.ini
```

If you wish to develop the documentation, please be aware of the warning above about long installation times. You can use the following make target to build the documentation in all available formats (output will be under `build/sphinx/{html,latex,man,...}`):

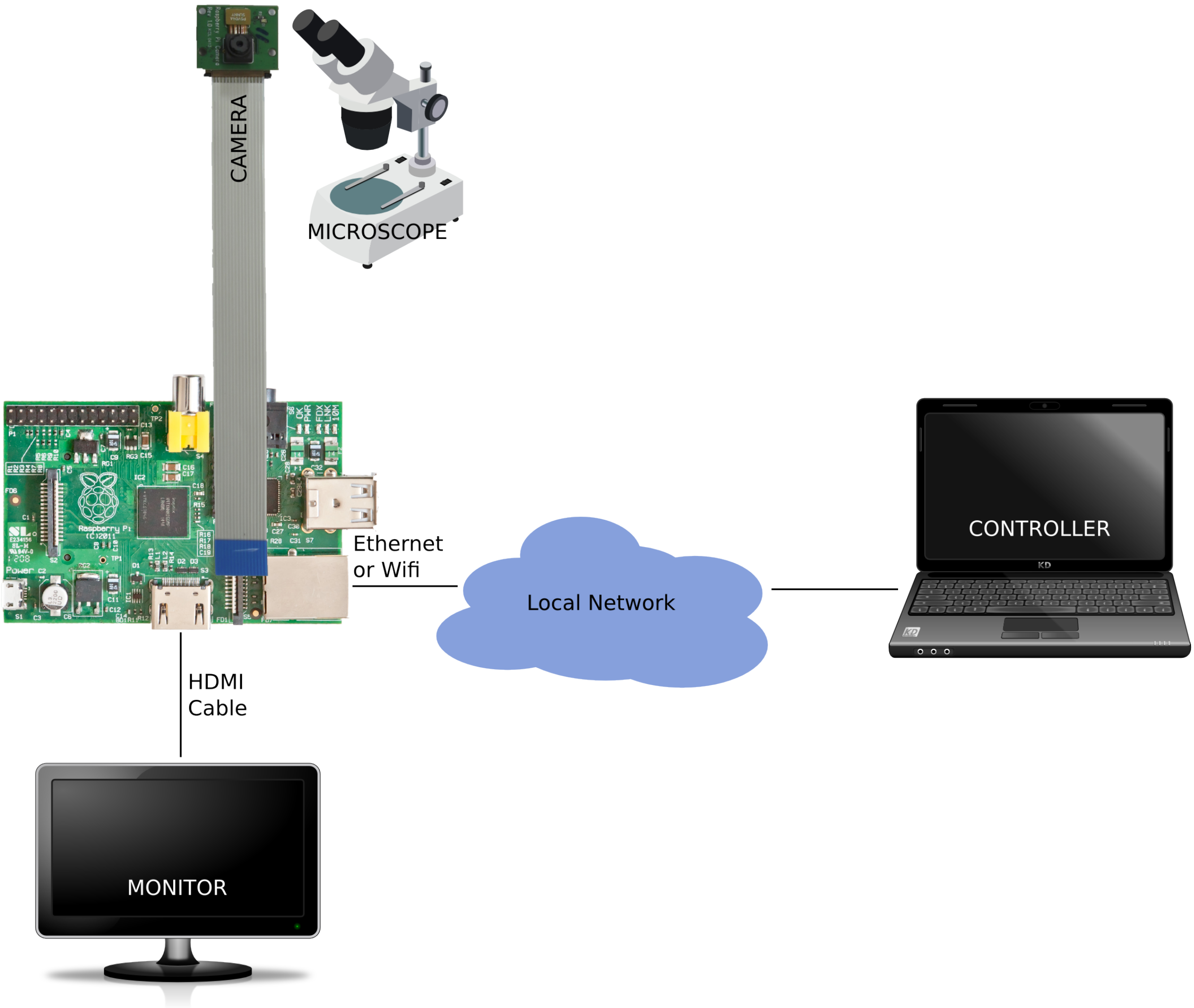
```
$ make doc
```

If you wish to run the test suite (currently non-existent!) use the following make target:

```
$ make test
```

Quick Start

A typical Picroscopy setup is illustrated in the image below:



The Raspberry Pi's camera is attached to the microscope. The exact manner in which the camera is mounted is left as an exercise for the reader. Simply taping the camera onto the eye-piece with electrical tape is usually sufficient for testing, but better results are generally obtained when the camera is held slightly away from the eye-piece.

The Raspberry Pi's HDMI port is connected to a monitor or TV to display the live feed from the camera (to allow for positioning and focusing of samples). The Raspberry Pi is connected to the network via the Ethernet port (or wifi if you have a compatible USB wifi dongle).

Finally, a second machine (or phone/tablet with web-browser) is used as the controller.

When Picroscopy is started, it should automatically start a video preview on the screen attached to the Pi. On a machine connected to the same network as the Pi, open a web browser and browse to the address and port that Picroscopy is listening on, e.g. `http://mypi.example.com:8000/`.

> **Note:** Picroscopy listens on port 80 by default when it is run by `root`, or on port 8000 when it is run by a non-root user.

The default page is the *Library Page* which displays the images taken by Picroscopy. Click on the *Settings* button and enter your name in the box at the top of the page, then click on *Apply* to return to the Library page.

> **Note:** By default, Picroscopy requires a name before it will capture images (the name, and other settings like copyright are added to the EXIF tags of captured images).

Click on the *Capture* button to capture the image currently display on the screen. The page should refresh and show a thumbnail of the captured image. At this point, the *Download* button should be enabled. Clicking on this button will download all captured images as a `.zip` archive.

If you entered your e-mail address on the *Settings Page*, the *Send* button should also be enabled. Clicking on this button will send an e-mail to your address with all captured images as attachments.

Finally, the *Logout* button will also be enabled. Clicking on this button will clear the Library of all captured images, and reset all fields on the *Settings Page* to their default values, ready for the next user.

User Guide

Library Page

Settings Page

Image Page

Administrators Guide

This section of the manual deals with configuring Picroscopy either via the command line switches or via a configuration file specified on the command line. It also provides rudimentary guidance on configuration of the Raspbian operating system under the assumption that this is the operating system most users will be using for Picroscopy.

Note: If any users wish to use Picroscopy with Arch Linux (the other major distro available for the Raspberry Pi platform), please feel free to submit documentation patches to the author. If you are reading this on ReadTheDocs, this can be done trivially with the `Edit on GitHub` link on the left of the page.

Network configuration

The `/etc/network/interfaces` file holds the network configuration under Raspbian. The default configuration which is shown below, attempts to obtain an IP address via DHCP:

```
iface eth0 inet dhcp
```

Should you wish your Pi to use a static address you will need to edit this file. A typical static address configuration for the private 10.0.0.0 network is shown below:

```
iface eth0 inet static
    address 10.0.0.150
    netmask 255.0.0.0
    gateway 10.0.0.1
```

Another static configuration is shown below, this time for the private 192.168.0.0 network typically used by home routers:

```
iface eth0 inet static
    address 192.168.1.3
    netmask 255.255.255.0
    gateway 192.168.1.1
```

After changing `/etc/network/interfaces` you will need to restart the networking service for the changes to take effect (note that if you are connected to your Pi via SSH this will very likely break your session):

```
$ sudo service networking restart
```

Alternatively you can simply reboot:

```
$ sudo reboot
```

Command Line

Picroscopy is started from the command line with the `picroscopy` command. Typically, options for the application are either passed as command line switches, or are specified as entries in a configuration file which is specified with the *`picroscopy -c`* option. If an option appears both in the configuration file and as a switch on the command line, the command line switch will take precedence. Specifically, the order of precedence for options is:

1. command line switch
2. configuration key
3. default

The Picroscopy application does not fork like a daemon once started. As it is a single user application that utilizes the Raspberry Pi's display there is little point in running as a system daemon, despite using a web interface. Furthermore, as the system monopolizes the Pi's display output, it is reasonable to expect that the application will be the only (visible) application running at any given time.

To this end, you may wish to configure the system to start Picroscopy at boot time, and shut down (or reboot) the Pi when the application terminates. A simple method for accomplishing this is to write a bash script similar to the following:

```
#!/bin/bash
picroscopy -c myconfig.ini
poweroff
```

Simply change the last line to `reboot` if you wish to reboot when picroscopy exits instead of powering off the computer. Save this script as `/root/run_picroscopy.sh` and make it executable by running the following command:

```
chmod +x /root/run_picroscopy.sh
```

To run this script on startup (without interfering with the rest of the boot sequence), add the following to the end of `/etc/rc.local`:

```
/root/run_picroscopy.sh &
```

The ampersand (&) at the end of the line ensures the script is started in the background, permitting the rest of the boot sequence to continue.

Command Line Reference

The picroscopy application launches a live view of the Raspberry Pi's camera and serves a web application providing control of the camera. Various options can be used to configure which network clients can access the web application, and the paths which the camera will use when writing images.

Synopsis

```
picroscopy [-h] [--version] [-c CONFIG] [-q] [-v] [-l FILE] [-P] [-d]
           [-L HOST[:PORT]] [-C NETWORK[/LEN]] [--images-dir DIR]
           [--thumbs-dir DIR] [--thumbs-size WIDTHxHEIGHT]
           [--email-from USER[@HOST]]
           [--sendmail EXEC | --smtp-server HOST[:PORT]]
```

Description

-h, --help
Show the application's help page, giving a brief description of each command line option, and exit.

--version
Show the program's version number and exit.

-c CONFIG**, --config** CONFIG
Specify the configuration file that the application should load. See the *Configuration* section for more information on the configuration file format.

-q, --quiet
Produce less console output. When this is specified, only error messages will be visible at the console. By default, warnings and error messages are displayed.

-v, --verbose
Produce more console output. When this is specified, information, warning, and error messages will be visible at the console. By default, warnings and error messages are displayed.

-l FILE**, --log-file** FILE
Log displayed messages to the specified FILE. The log file will be appended to if it already exists. Its format will include the timestamp that the message was displayed, and the severity of the message.

-P, --pdb
Run under PuDB (if available) or PDB. This launches Picroscopy within a Python debugger for development purposes.

-L HOST[:PORT]**, --listen** HOST[:PORT]
The address and port of the interface that Picroscopy will listen on. Defaults to `0.0.0.0:80` (when running as root) or `0.0.0.0:8000` (when running as a non-root user). The `0.0.0.0` address means "listen on all available network interfaces".

-C NETWORK[/LEN]**, --clients** NETWORK[/LEN]
The network that clients must belong to. Clients that do not belong to the specified network will be denied access to Picroscopy. Defaults to `0.0.0.0/0` (all valid addresses).

--images-dir DIR
The directory in which Picroscopy will store images captured by the camera. If not specified, defaults to a temporary directory which is destroyed upon exit. If the specified directory does not exist, it will be created.

--thumbs-dir DIR
The directory in which Picroscopy will store thumbnails generated from the images taken by the camera. If not specified, defaults to a temporary directory which is destroyed upon exit. If the specified directory does not exist, it will be created. The thumbnails directory *must* be different to the images directory.

--thumbs-size WIDTHxHEIGHT
The maximum size for generated thumbnails (the actual size may be smaller due to aspect ratio preservation). Defaults to 320x320.

--email-from USER[@HOST]
The address which Picroscopy will use as a From: address when sending e-mail. If HOST is not specified, the configuration of the sending SMTP server will determine the host associated with the USER.

--sendmail EXEC
Use the specified sendmail binary to send e-mail. This is the preferred option for sending e-mail as it

(usually) gracefully handles the case where the target SMTP server is unavailable. Defaults to `/usr/sbin/sendmail`.

`--smtp-server` `HOST[:PORT]`
Use the specified SMTP smarthost to send e-mail. This should only be used if you do not wish to configure a local sendmail binary. If this option is specified, it will always override any `--sendmail` specification.

Examples

Run Picroscopy, with the default configuration and verbose logging:

```
$ picroscopy -v
```

Run Picroscopy, listening for clients on port 8080 of the interface with the address `192.168.0.5`, and only accepting requests from the machine with IP address `192.168.0.6`:

```
$ picroscopy -L 192.168.0.5:8080 -C 192.168.0.6
```

Run Picroscopy, only accepting requests from the `192.168.0.0` private network:

```
$ picroscopy -C 192.168.0.0/16
```

Run Picroscopy, ensuring that e-mail is sent via the SMTP server running on `localhost`, and that e-mail appears to come from `noreply@example.com`:

```
$ picroscopy --smtp-server localhost --email-from noreply@example.com
```

Run Picroscopy, explicitly specifying the images directory and the thumbnails directory (which ensures both persist across runs; the default is to use ephemeral temporary directories):

```
$ mkdir -p picroscopy/images picroscopy/thumbs
$ picroscopy --images-dir picroscopy/images --thumbs-dir picroscopy/thumbs
```

Configuration

Picroscopy reads its configuration files from the following locations, in the order presented (i.e. values found in later files override values found in earlier files):

1. `/etc/picroscopy.ini`
2. `/usr/local/etc/picroscopy.ini`
3. `~/.picroscopy.ini` (where ~ represents the current user's home directory)

You can manually specify a configuration to load with the *`picroscopy -c`* option. In this case, the manually specified configuration will be read last, ensuring its values take precedence over any values read from the files listed above.

Picroscopy's configuration format is based on the familiar INI-file format. The configuration file must have a `[picroscopy]` section (Picroscopy will ignore other sections within the file), which contains `key=value` entries on separate lines. Key names are case insensitive. Key names and values may have leading or trailing whitespace which will be ignored. Blank lines are ignored, as are comments which are whole lines prefixed with either # or ;.

An example configuration file is shown below:

```
[picroscopy]

; Blank lines are ignored, as is this line, which is a comment
# This is also a comment
```

```
; Spaces surrounding keys and values are ignored...
  listen = 127.0.0.1:8000
clients = 127.0.0.0/8

; Case is ignored for key names
IMAGES_DIR=/tmp/picroscopy_images
Thumbs_Dir=/tmp/picroscopy_thumbs
```

The key names which can appear in the configuration file are the same as the available "long-style" command line options, with the caveat that leading dashes are stripped and any dashes within the option are replaced by underscore. Hence the `picroscopy --images-dir` option becomes the *images_dir* key within the configuration file.

Example Configurations

Two example configuration files are shipped with Picroscopy's source: `picroscopy.ini` which contains a configuration suitable for normal usage (all defaults), and `development.ini` which contains values suitable for development purposes. It is recommended that `picroscopy.ini` be placed in a suitable location where Picroscopy can find it automatically, e.g. `/etc` or `/usr/local/etc`.

Keys

The remainder of this document is a description of the available keys in a Picroscopy configuration file.

log_file

Log displayed messages to the given filename. The log file will be appended to if it already exists. Its format will include the timestamp that the message was displayed, and the severity of the message. Log files include all messages regardless of the verbosity of console output.

pdb

If `true`, run under PuDB (if available) or PDB. This launches Picroscopy within a Python debugger for development purposes.

listen

The address and port of the interface that Picroscopy will listen on. Defaults to `0.0.0.0:80` (when running as root) or `0.0.0.0:8000` (when running as a non-root user). `0.0.0.0` address means "listen on all available network interfaces".

clients

The network that clients must belong to. Clients that do not belong to the specified network will be denied access to Picroscopy. Defaults to all valid addresses (`0.0.0.0/0`).

images_dir

The directory in which Picroscopy will store images captured by the camera. If not specified, this defaults to a temporary directory which is destroyed upon exit. If the specified directory does not exist, it will be created.

thumbs_dir

The directory in which Picroscopy will store thumbnails generated from the images taken by the camera. If not specified, defaults to a temporary directory which is destroyed upon exit. If the specified directory does not exist, it will be created. The thumbnails directory *must* be different to the images directory.

thumbs_size

The maximum size for generated thumbnails (the actual size may be smaller due to aspect ratio preservation). Defaults to 320 pixels square.

email_from

The address which Picroscopy will use as a From: address when sending e-mail. The default is `picroscopy` with no specific host. If a host is not specified, the configuration of the sending SMTP server will determine the host associated with the address.

sendmail

Use the specified sendmail binary to send e-mail. This is the preferred option for sending e-mail as it (usually) gracefully handles the case where the target SMTP server is unavailable. Defaults to `/usr/sbin/sendmail`.

smtp_server

Use the specified SMTP smarthost to send e-mail. This should only be used if you do not wish to configure a local sendmail binary. If this option is specified, it will always override any `sendmail` specification.

Frequently Asked Questions (FAQ)

Why can't I view the preview in the browser?

There are numerous reasons for choosing to use HDMI for the video preview. The major one is simplicity: this was by far the quickest and simplest way of getting the project working. It's also trivial for the user to setup, provides perfect video quality and near-instant feedback of actions on the microscope.

Streaming over the LAN means dealing with encoding, client codecs, network bandwidth, latency, the list goes on - and of course the result will never be as good as the straight HDMI feed.

This is not to say that an in-browser preview isn't something that's being considered for the future but it simply wasn't a priority for the first version.

Why can't I control everything from the Pi?

Again, simplicity. Building an application on the Pi that has an interface and provides the live video interface means dealing with GPU coding (in order to get reasonable preview quality and latency). Relying on a separate machine running a web-browser to render the interface was the simplest method.

License

Copyright (c) 2013, Dave Hughes

libmmal Header License

libbcm_host Header License

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index