How to setup the Phenotiki sensor

*Massimo Minervini**
Pattern Recognition and Image Analysis (PRIAn)
IMT Institute for Advanced Studies Lucca, Italy
Version 1.0

October 23, 2015

Abstract

This document provides detailed instructions to setup and operate the Phenotiki affordable sensing solution for plant phenotyping. First, we describe the RaspiCam image sensor and the web-based interface that we developed to operate the sensor; next, we describe how to setup the Raspberry Pi device.

Contents

* massimo.minervini@imtlucca.it

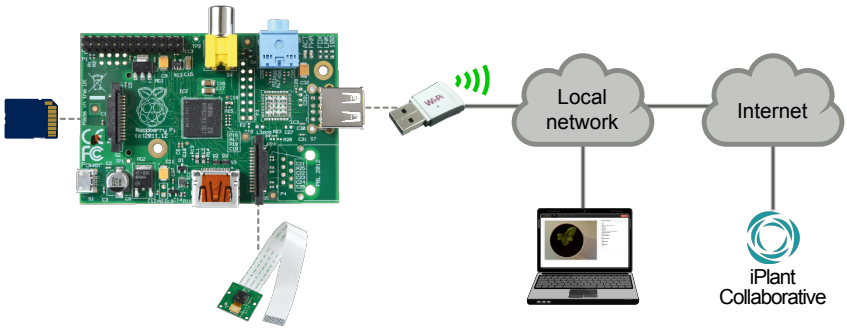


Figure 1: Schematic of our sensing solution based on the Raspberry Pi, showing the headless mode for image acquisition and transmission.

1 A smart sensor based on the Raspberry Pi

The Raspberry Pi [1, 2] is a credit-card sized single-board computer, designed and developed by the Raspberry Pi Foundation,[1] UK, as an educational tool for teaching computer science and programming [3, 4]. The Raspberry Pi is cheap (monetary cost is €35) and easy to use (it runs the Raspbian, a full-featured Linux operating system), therefore it is ideal to implement prototypes or customized systems.

As illustrated in Figure 1, we use a Raspberry Pi (Model B) single-board computer, shorthanded as *RPi*, equipped with a 5 megapixel 'RaspiCam' camera module, to capture static images (width×height: 2592×1944 pixels) of the scene. This solution offers great flexibility by having a complete yet small computer attached to the sensor. While the RaspiCam is capable of acquiring images of good quality that can satisfy a wide range of applications (the infrared camera module 'NoIR' is also available), in contexts where superior image quality is required, the Raspberry Pi can be used in combination with an SLR camera or other imaging sensors.

To control the camera and acquire images we adopt the *raspistill* application. The images acquired by the RPi setup are also automatically transmitted to the cloud for storage and analysis. Here we rely on the scientific cloud infrastructure offered by the iPlant Collaborative project [5], to deploy our plant image analysis software solution.

In a distributed sensing and analysis scenario, in which the acquired data needs to be transmitted via the Internet to centralized locations (e. g., a cloud service) for analysis, it becomes necessary to compress the images effectively [6]. In this context, a single-board computer such as the Raspberry Pi operating the sensor offers the possibility to: (a) perform (low-complexity) pre-processing operations on the acquired imagery (e. g., image enhancement, specialized compression, low level vision); and (b) autonomously transmit the images to a remote location.

To ease configuration and monitoring, we deploy a web-based interface to operate the sensor remotely (cf. Figure 1). In the following, we describe how to set up the hardware and software components of our affordable sensing solution based on the Raspberry Pi.

[1] `http://www.raspberrypi.org`

2 Imaging sensor

2.1 RaspiCam

The "RaspiCam" camera module is a fixed-focus 5 megapixel CMOS image sensor produced by OmniVision Technologies.[2]

raspistill: Two command line utilities are available on the Raspberry Pi to operate the camera: `raspistill`, to capture still photos, and `raspivid`, to record HD video. For example, to acquire a picture and save it in the PNG format [7] we use the following command line options:

```
raspistill -n -e png -awb fluorescent -rot 180 -o filename
```

The `raspistill` utility is easy to use and offers several options to configure image acquisition [8]. In particular, in our experimental setup we use neon lights for illumination, therefore we pass the option *fluorescent* to the automatic white balance algorithm embedded in `raspistill`.

Using a job scheduler (e. g., the Cron software utility), it is possible to run the `raspistill` periodically, to acquire a time-lapse sequence of images. However, any changes to the mode of operation require that the Raspberry Pi be connected to all input/output peripherals to provide the user with physical access. This procedure involves attaching and detaching cables, which may cause undesired displacements of the sensing device or the plants. Furthermore, in some scenarios, physical access to the device may not be possible or desirable after the initial setup, thus introducing the need for a solution that enables remote control.

2.2 Easy sensor setup with raspistillWeb

The Raspian runs by default an SSH (secure shell) server, allowing remote access to the command line of the Raspberry Pi using the following command:

```
ssh -p 22 pi@<IP>
```

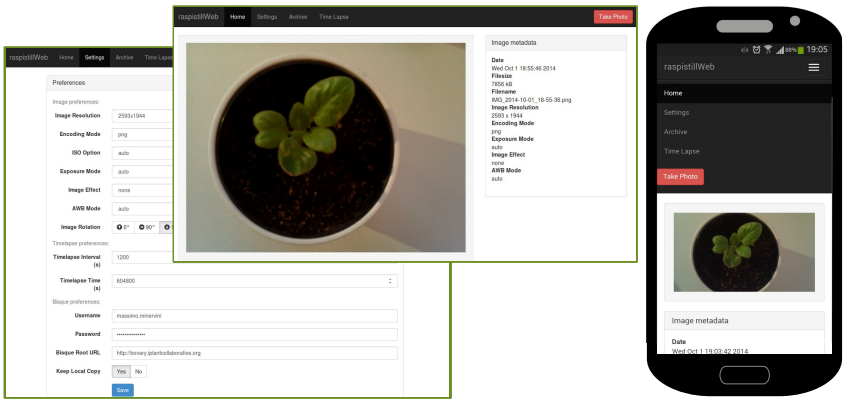
where `<IP>` is the local (or remote) IP address of the Raspberry Pi (executing the command `hostname -I` will display the IP address assigned to the Raspberry Pi, e. g., `192.168.1.4`). However, operating the camera module from the command line may reveal problematic for some users. Therefore, we devise a web-based interface to configure and operate the camera module of the Raspberry Pi easily from another computer (e. g., a laptop or even a smartphone, cf. Figure 2).

We implement our interface as a fork of the *raspistillWeb*[3] project (version 0.1), i. e. a web interface for the `raspistill` tool, implemented using the Python programming language and the Pyramid[4] web framework. We adapt the original software platform to the requirements of our application, adding the following key features:

(a) the user can select among different image file formats, including lossy (JPEG [9], GIF [10]) and lossless (BMP [11], PNG [7]) coding standards;

(b) the user can start and interrupt time-lapse image acquisitions;

[2] `http://www.ovt.com`
[3] `https://github.com/TimJuni/raspistillWeb`
[4] `http://www.pylonsproject.org`



(a) Desktop interface (b) Mobile interface

Figure 2: Screen captures of our web-based interface to operate the Raspberry Pi camera, based on the raspistillWeb project.

(c) more detailed information and metadata are displayed about acquired images; and

(d) acquired images can be transmitted to the iPlant Collaborative [5] cloud infrastructure for storage and analysis.

As shown in Figure 2, graphical control elements of the user interface are intuitive and self-explanatory, thus rendering the web application easy-to-use. The 'Settings' page allows to configure parameters regarding image acquisition, time-lapse photography, and transmission to the iPlant. 'Home' and 'Time Lapse' pages allow to capture single still images and initiate a time-lapse acquisition, respectively. Detailed information about acquired images is displayed in 'Home' and 'Archive' pages. In the 'Archive' page the user can browse previously acquired images and download time-lapse sequences as compressed archives. Our interface also adapts to small screens and can be displayed on mobile devices such as smartphones and tablets (cf. Figure 2).

Acquired images are transmitted to the iPlant using the Bisque Python APIs and the user's credentials (username and password) for iPlant. The user can also decide to delete the local copy of the image files after their transmission, to save storage space on the Raspberry Pi.

3 Setting up the Raspberry Pi

3.1 Hardware requirements

To implement our imaging sensor based on the Raspberry Pi we use the following hardware equipment:

- Raspberry Pi Model B;
- "RaspiCam" camera module;
- USB Micro power supply;
- 8 GB Secure Digital (SD) memory card;
- HDMI monitor and cable;
- USB keyboard and mouse;
- USB wireless dongle;
- self-powered USB hub.

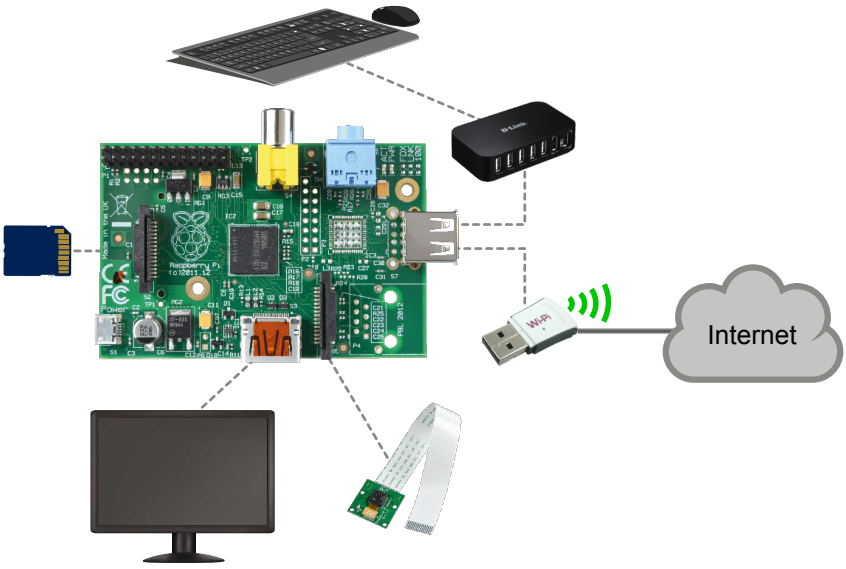


Figure 3: Schematic of our sensing solution at first configuration time, showing system setup and configuration. Note that after the initial setup, the system does not require input/output peripherals, and can be operated remotely (cf. Figure 1).

Setting up the Raspberry Pi requires only few simple steps, described in detail in the official documentation.[5] In order to install the necessary software and perform initial configurations, the device must be attached to input/output peripherals (monitor, keyboard, and mouse), as shown in Figure 3. To connect the camera module to the Raspberry Pi, we proceed as shown in the instructional video available at `http://www.youtube.com/watch?v=GImeVqHQzsE`. Subsequently, the Raspberry Pi can be started headless and operated remotely from a computer connected to the same local network (cf. Figure 1).

3.2 Operating system and software setup

Raspbian

Next, we download and install the latest version of the Raspbian operating system, available at: `http://downloads.raspberrypi.org/raspbian_latest`. We extract the system image from the ZIP archive and install it on the SD card (also used for local storage), following the instructions in the official documentation [8].[6] For example, on Linux, we run the following command with superuser privileges:

```
dd bs=4M if=2014-06-20-wheezy-raspbian.img of=/dev/sdc
```

to install the Raspbian image `2014-06-20-wheezy-raspbian.img` on the device named `/dev/sdc` associated with the SD card.

On first booting (default user name and password are, respectively, 'pi' and 'raspberry'), the Raspberry Pi configuration tool `raspi-config` is automatically started, allowing system-level configuration tasks. In particular, we edit: 'Internationalization Options', to change locale, timezone, and keyboard layout; 'Enable Camera', to enable the camera module; and

[5] `http://www.raspberrypi.org/documentation/setup/`
[6] `http://www.raspberrypi.org/documentation/installation/`

'Enable Boot to Desktop/Scratch', to enable automatic booting into a desktop environment (this option should be disabled after all configurations are done, to reduce overhead on the device when it is started headless).

Packages installation: With the USB wireless adapter plugged into the Raspberry Pi, we configure access to a local Wi-Fi connection using the WiFi Config tool available in the Raspbian (alternatively, wired connection is possible using an Ethernet network cable). Once the connection is established, we run in a terminal:

```
sudo rpi-update
sudo apt-get update
sudo apt-get -y dist-upgrade
```

to upgrade the device firmware and all installed packages to the latest available version (this operation may take some time, depending on network speed). Then, we install the software dependencies necessary for our operations:

```
sudo apt-get install python2.7-dev python-virtualenv \
       python-setuptools python-imaging python-picamera \
       python-lxml python-bs4
sudo pip install requests==2.4.1
```

Bisque support: We build and install the Python APIs for Bisque, with the following commands:

```
hg clone http://biodev.ece.ucsb.edu/hg/bisque/
cd bisque/bqapi/
python setup.py build_py
cp -r bqapi/RequestsMonkeyPatch/ \
       build/lib.linux-armv6l-2.7/bqapi/
sudo python setup.py install
```

raspistillWeb: Our fork of the raspistillWeb software can be installed with the following series of commands:

```
mkdir ~/Development
cd ~/Development
virtualenv env
cd env
wget http://.../raspistillWeb-PHIDIAS.tar.gz
tar -xvf raspistillWeb-PHIDIAS.tar.gz
```

To start the raspistillWeb service (by default listening on port number 6543) we execute on the Raspberry Pi:

```
cd Development/env/raspistillWeb
../bin/pserve development.ini
```

Then, from a web browser on a computer connected to the same local network, we access the raspistillWeb interface at the address `http://192.168.1.4:6543`. Note that by properly configuring the router of the local network, it is possible to enable remote access to the raspistillWeb via the Internet. In order to start the raspistillWeb automatically on system boot, we edit the `/etc/rc.local` file, adding the following lines:

```
cd ~/Development/env/raspistillWeb
../bin/pserve development.ini
```

before the line `exit 0` at the bottom of the file.

References

[1] C. Severance, "Eben Upton: Raspberry Pi," *Computer*, vol. 46, no. 10, pp. 14–16, 2013.

[2] E. Upton and G. Halfacree, *Raspberry Pi User Guide*. Wiley, 2014.

[3] C. Andrews, "Easy as Pi," *Engineering Technology*, vol. 8, no. 3, pp. 34–37, 2013.

[4] F. Cuomo, E. Mibuari, K. Weldemariam, and O. Stewart, "Leveraging Raspberry Pi for interactive education," in *Annual Symposium on Computing for Development*, ser. ACM DEV-4 '13, 2013.

[5] S. A. Goff, M. Vaughn, S. McKay, E. Lyons, A. E. Stapleton, D. Gessler, N. Matasci, L. Wang, M. Hanlon, A. Lenards, A. Muir, N. Merchant, S. Lowry, S. Mock, M. Helmke, A. Kubach, M. Narro, N. Hopkins, D. Micklos, U. Hilgert, M. Gonzales, C. Jordan, E. Skidmore, R. Dooley, J. Cazes, R. McLay, Z. Lu, S. Pasternak, L. Koesterke, W. H. Piel, R. Grene, C. Noutsos, K. Gendler, X. Feng, C. Tang, M. Lent, S.-J. Kim, K. Kvilekval, B. S. Manjunath, V. Tannen, A. Stamatakis, M. Sanderson, S. M. Welch, K. A. Cranston, P. Soltis, D. Soltis, B. O'Meara, C. Ane, T. Brutnell, D. J. Kleibenstein, J. W. White, J. Leebens-Mack, M. J. Donoghue, E. P. Spalding, T. J. Vision, C. R. Myers, D. Lowenthal, B. J. Enquist, B. Boyle, A. Akoglu, G. Andrews, S. Ram, D. Ware, L. Stein, and D. Stanzione, "The iPlant collaborative: Cyberinfrastructure for plant biology," *Frontiers in Plant Science*, vol. 2, no. 34, 2011.

[6] M. Minervini and S. A. Tsaftaris, "Application-aware image compression for low cost and distributed plant phenotyping," in *18th International Conference on Digital Signal Processing (DSP)*, Santorini, Greece, Jul. 2013, pp. 1–6.

[7] *Portable Network Graphics (PNG) Specification*, W3C Recommendation ISO/IEC 15 948:2003 (E), Rev. 2, 2003. [Online]. Available: http://www.w3.org/TR/PNG/

[8] Raspberry Pi documentation. Raspberry Pi Foundation. [Online]. Available: https://www.raspberrypi.org/documentation/

[9] *Information technology – Digital compression and coding of continuous-tone still images*, International Telecommunication Union (ITU) ITU-T Recommendation T.81, 1992.

[10] CompuServe Inc., "Graphics interchange format (GIF)," Jul. 1990, version 89a. [Online]. Available: http://www.w3.org/Graphics/GIF/spec-gif89a.txt

[11] J. D. Murray and W. VanRyper, *Encyclopedia of graphics file formats*. O'Reilly, 1994.