Piccolo User Guide

Release 2.0.0

Magnus Hagdorn, Iain Robinson, Alasdair Mac Arthur

May 28, 2018

Contents

Contents:

CHAPTER 1

Introducing

The Piccolo is a hardware instrument for recording the optical spectrum of light. The hardware includes a spectrometer (or two spectrometers on the *Piccolo Doppio*) and an optical fibre. Light input to the instrument is controlled by electronic shutters and fore-optics provide a weather-proof housing.

The Piccolo also includes a small computer, the Raspberry Pi, which controls the Piccolo hardware and acquires spectra. The Piccolo software on the Raspberry Pi can record spectra. It also allows the Piccolo to be controlled remotely from a remote laptop (or desktop computer) over a network link.

1.1 Online resources

- This documentation is available online, hosted by *Read the Docs*.
- Bitbucket is the online home of the Piccolo's Python source code.

1.2 About the User Guide

The Piccolo User Guide covers the essential features of the Piccolo software:

- Installing the Piccolo software
- Plotting data acquired on the Piccolo
- Programming the Piccolo with Python

1.3 Hardware

The Piccolo hardware supports one or two optical spectrometers. The following models are supported.

Manufacturer	Series	Supported models
Ocean Optics	Flame	Flame-S and Flame-T
Ocean Optics	HR	HR4000
Ocean Optics	Maya2000 Pro	All in series
Ocean Optics	NIRQuest	NIRQuest512
Ocean Optics	QEPro	All in series
Ocean Optics	USB	USB2000+

The Raspberry Pi is a miniature computer which runs an operating system, *Raspbian*, which is a variant of Linux.

CHAPTER 2

Getting started

2.1 Unpack

A Piccolo optical spectrometer instrument is required to use the firmware and software described here.

A laptop computer is required to run *Picoplayer*, the software for controlling and interacting with the Piccolo. The laptop is not included with Piccolo.

Note: The Piccolo can be controlled from any computer, laptop, desktop, server. In these instructions the computer controlling the Piccolo will be referred to as the *laptop*.

The computer must meet the following specifications:

- A laptop or personal computer (PC)
- The Ubuntu 16.04 LTS operating system
- Python 2.7 (included by default with Ubuntu)

Warning: Although Piccolo Player is known to run on other operating systems which support Python, Ubuntu is recommended and is the only system for which support is provided. Other operating systems, such as Mac OS and Windows differ considerably in the way they run Python.

2.2 Power

Power the Piccolo with a 12 volt direct current supply or a battery. The current must be sufficient to meet the requirements (approx. 1.2 amps).

2.3 Firmware

This step can usually be skipped. The Piccolo firmware is supplied with the instrument on the Raspberry Pi's SD card. Therefroe, skip this step and go to Turn On.

The Piccolo instrument runs firmware which is held on a Secure Digital (SD) memory card. The firmware controls the Piccolo's hardware (including the spectrometers, and shutters) and handles the recording of spectra.

Note: A USB flash drive is supplied with the Piccolo to store the spectra it records. The firmware is not on this USB flash drive; it is on the SD memory card. This card also contains a piccolo.config file reqired by the system. If the config file is not present the piccolo will not run.

Picoplayer is software with a graphical user interface which enables you to acquire spectra and adjust instrument settings. It communicates with the instrument using a radio or network connection. In addition it can display plots of spectra as they are being acquired.

Note: The Piccolo software is written entirely in version 2.7 of the Python programming language.

This section describes how to install the firmware. The firmware is stored on the SD card and runs on the Raspberry Pi.

2.3.1 Set up the SD card

The first step in setting up the SD card is to copy *Raspian Jessie Lite* on it. This process must be performed on a linux laptop (or other computer) preferably with Ubuntu 16.04 LTS and with an SD card slot.

1. Download *Raspbian Jessie* zip file from the downloads page at the Raspberry Pi Foundation (make sure you use 2017-07-05 version)
2. Follow the Raspberry Pi Foundation's instructions to copy it onto the memory (SD) card.

Warning: Copying *Raspbian* onto the SD card will erase all other data on the card.

Warning: If using the `dd` tool to copy *Raspbian* onto the SD card, remember that this tool can overwrite other data on the laptop if used incorrectly.

3. Connect the Raspberry Pi to the laptop with an Ethernet cable (or alternatively to a display and keyboard).

When the pi is booted up for the first time the Raspberry Pi 'Pixel' desktop environment will be displayed. Click on the 'terminal' icon to open a command line terminal.

4. Type `sudo raspi-config` to open a dialogue.
5. From this dialogue select `Change User Password` and follow instruction to change the password to a password you wish to use. Do not leave the username as 'pi' and password as 'raspberry' as these defaults would allow anyone on the Web or your network to access your system when you go online.
6. Select `Boot Options` and select the option to boot to the terminal and to require a pasworh at startup.
7. Select `Advanced Options` then `SSH` and enable the pi so that you can connect via ssh.

8. When the picolo was provided to you it was matched to a raspberry Pi and given a unique name i.e. piccolo1. The Piccolo Team keep a record of this information so that if the system is returned to UofE for calibration it can be logged in. Please reuse the same piccolo name if tyou are rebuilding the SD card.

2.3.2 Package list

The `apt-get` tool allows software to be installed onto the Raspberry Pi directly from its internet connection. To do this it maintains a local package list which needs occasionally to be updated. Type:

```
sudo apt update
```

this will update the raspian Jessie operatig system

then:

```
sudo apt dist-upgrade
```

to upgrade the system to the latest Jessie packages

Enter the username and password when prompted. (If the The Raspberry Pi username and password have not been changed they will still be the default `pi` and `raspberry`.)

Note it may take some time to update and upgrade the operating system.

2.3.3 Install Mercurial

This step is optional.

Mercurial is a source code management tool. It is useful to developers of the Piccolo software to quickly upload and download source code from Bitbucket, the web site which holds the development version of the software.

To install Mercurial type the following command on the Raspberrry Pi:

```
sudo apt-get install mercurial
```

If asked `Do you want to continue? [Y/n]` answer `Y` for yes.

2.3.4 Installing dependencies

First of all you need to install some dependencies. On the Raspberry Pi, type.:

```
sudo apt-get update
sudo apt-get install devscripts debhelper cython
```

2.3.5 Installing the virtual environment

This is needed to creat an environement for the piccolo python project. Do not use apt-get as this will install an older version

The lastest version can be installed by typing:

```
sudo apt-get install python-virtualenv git equivs
git clone https://github.com/spotify/dh-virtualenv.git
cd dh-virtualenv
sudo mk-build-deps -ri
dpkg-buildpackage -us -uc -b
sudo dpkg -i ../dh-virtualenv_1.0-1.deb
```

If an old version of dh-virtualenv was installed it will be upgraded.

2.3.6 Install debs (*server-bundle*.deb and/or *player-bundle*.deb)

If you have been provided with a link to the `*-bundle.deb` files download them and unzip if necessary. For the piccolo create a piccolo2-packages directory on the RPi

```
pi@rasppberrypi:~$ mkdir piccolo2-packages/
```

and copy the *server-bundle*.deb into it . This may require you poweroff the RPi and put the SD card into a loptop to copy the `*.deb` over. N.B. the debs nmust be built on the computer thta they will be used on i.e. a RPi with Raspian Jexssie lite (2017-04-10 version)

You only need to download and install *player-bundle*.deb to the RPi you wish to be able to run the Player GUI on the RPi. If it is not installed on the RPi you will still be able to run the Player GUI on a computer.

For a laptop, create a piccolo2-packages directory and copy the *player-bundle*.deb to there

These -*bundle*.deb files are then installed by:: sudo dpkg -i -*bundle*.deb

If there are dependancy problems run:

```
sudo apt-get install -f
```

and then try:: sudo dpkg -i -*bundle*.deb

again.

2.3.7 To download and Install individual packages if *server-bundle*.deb or *player-bundle*.deb are not avialable

With Mercurial installed, the Piccolo repository can be cloned onto the Raspberry Pi.

Download each of the Picolo firmware packages from Bitbucket with:

```
hg clone https://username@bitbucket.org/teampiccolo/piccolo2-common
hg clone https://username@bitbucket.org/teampiccolo/piccolo2-packaging
hg clone https://username@bitbucket.org/teampiccolo/piccolo2-client
hg clone https://username@bitbucket.org/teampiccolo/piccolo-hardware
hg clone https://username@bitbucket.org/teampiccolo/piccolo2-player
```

You only need to download and install piccolo2-player you wish to be able to run the Player GUI on the RPi. If it is not installed on the RPi you will still be able to run the Player GUI on a computer.

2.3.8 Build packages

The package bundles are built by running the `` `dpkg-buildpackage` `` command in the corresponding subdirectory. For example:

```
cd /home/pi/somewhere/piccolo2-packaging/client
dpkg-buildpackage -us -uc
```

This command may take some minutes to complete on the Raspberry Pi for each of the packages.

2.3.9 Upgrading the Piccolo OS

Upgrading the packages in similar. cd to the package directory for each of the Piccolo firmware packages, for example:

```
cd piccolo2-common
```

then:

```
hg pull
hg update
```

2.4 Turning on the Piccolo controller

Switch on the Piccolo by connecting a battery or powering up the supply.

There are several light-emitting diodes (LEDs) that indicate the status of the device.

The red LED labelled `PWR` on the Raspberry Pi should be illuminated. This indicates that the Raspberry Pi is powered up.

The green LED labelled `ACT` should flash intermittently whilst the Raspberry Pi is booting. It indicates that the Raspberry Pi is loading the operating system and the Piccolo firmware from its Secure Digital (SD) card. It should stop flashing once loaded.

Optional: A monitor and keyboard can be connected to the Raspberry Pi to monitor the Piccolo. This is not essential, but can be useful for getting started. The monitor should have a High Definition Multimedia Interface (HDMI) connector and the keybaord a Universal Serial Bus (USB) connector.

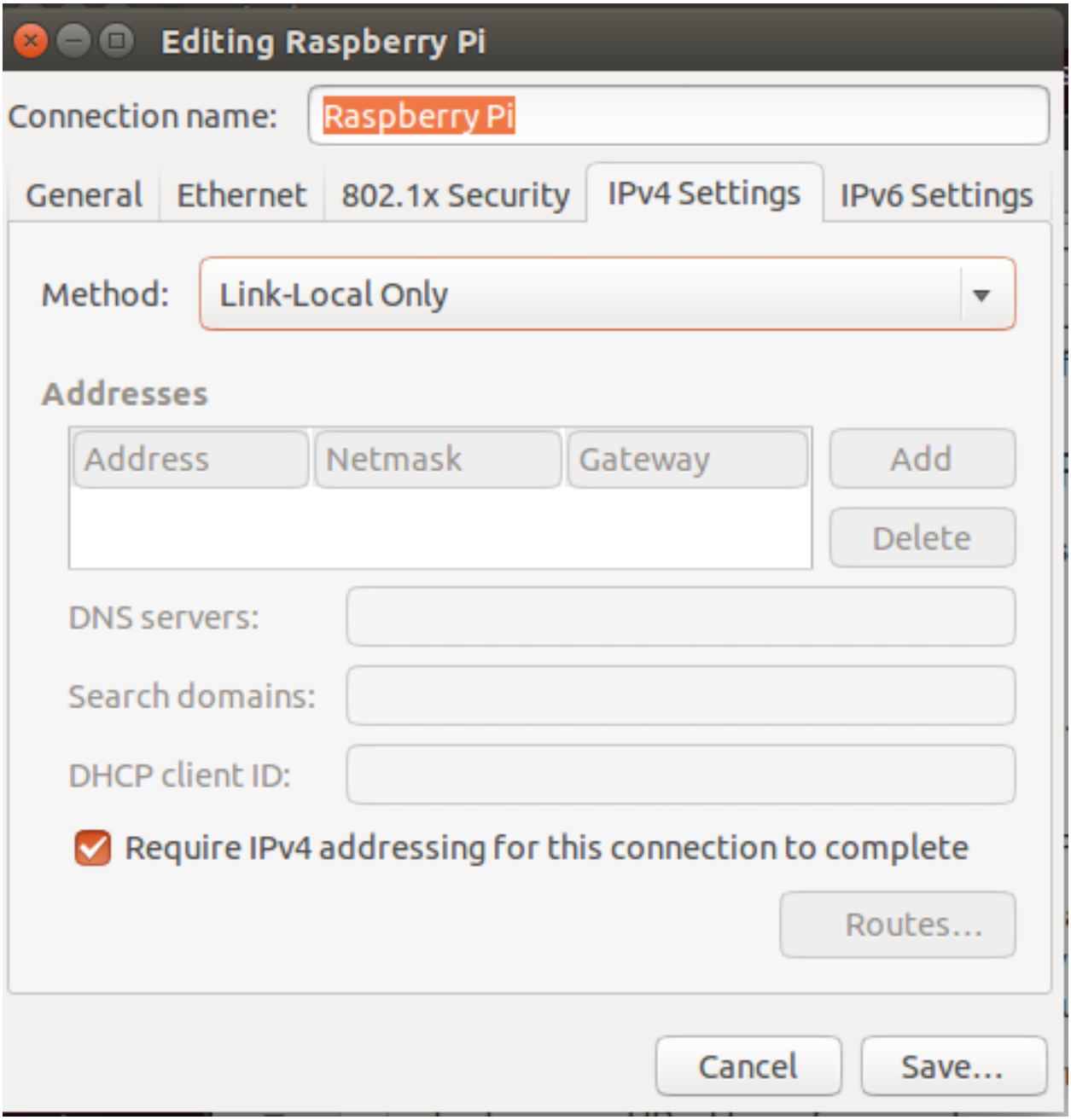
2.5 Connect

The Piccolo is usually controlled using a laptop. There are two ways to connect the laptop to the Piccolo: wired and wireless. A wired connection uses an Ethernet cable. A wireless connection uses a digital radio module.

2.5.1 Wired

A wired connection can be made to the Piccolo using an Ethernet cable. To do this, use anEthernet cable to connect the Ethernet port of the Raspebrry Pi to the Ethernet port of the laptop. The cable should be a standard (Category 5e) cable and not a cross-over cable.

Ubuntu must be configured to make a `Link-local` connection. Select `Require IPv4 addressing for this conneciton to complete`.



At this point it may be necessary to enable ssh as it is not enabled by default.

To connect to the Raspberry Pi with *Secure Shell* (*ssh*), type:

```
ssh pi@raspberrypi.local
```

Warning: Plugging the Raspberry Pi into a network may make it available to other computers on the network or the internet. It is a good idea to change the default password for the `pi` user from the default (`raspberry`) to something more secure. The password can be changed with the Raspberry Pi configuration tool described in the next subsection.

2.5.2 Wireless

Set up a wireless connection using the radio modules supplied.

2.6 Select the IP address and port

Whilst Piccolo Server is running on (*binding to*) the `localhost` address (usually `127.0.0.1`) it will not (at least not always) be accessible to other machines on the network. To remedy this, Piccolo Server needs to run on the Internet Protocol (IP) address of the Raspberry Pi.

To find out the IP address of the Raspberry Pi type:

```
ip addr
```

The result of this command is a list of network interfaces. The IP address should be in this list. This will be different for different networks. As an example, if the Raspberry Pi is connected to the network wirelessly, the IP address will be given under the wireless adapters list `wlan0`:

```
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default␣
↪qlen 1000
    link/ether a0:f3:c1:1d:ff:b0 brd ff:ff:ff:ff:ff:ff
    inet 172.16.1.113/24 brd 172.16.1.255 scope global wlan0
       valid_lft forever preferred_lft forever
    inet6 fe80::e4ec:c07:79ff:92d7/64 scope link
       valid_lft forever preferred_lft forever
```

In the above case, the IP address of the Raspberry Pi is `172.16.1.113`.

To verify that the correct IP address has been identify, try using it to connect to the Raspberry Pi with secure shell:

```
ssh pi@172.16.1.113
```

If this opens a connection to the Raspberry Pi this confirms that `172.16.1.113` is the IP address of the Raspberry Pi (and not some other device on the network).

2.7 Dialout

This step applies only to Piccolo instruments equipped with XBee wireless digital radio modules.

Add the user to the `dialout` group for using USB radio modules:

```
sudo usermod -a -G dialout $USER
```

+============== Download Player ===============

Download the Piccolo Player Debian package archive (*deb* file).

2.8 Install Piccolo Player

Install it:

```
sudo dpkg --install piccolo-player-x.x.deb
```

In the above command, replace `x.x` with the version number of the downloaded Debian package archive file.

If there are dependency problems these can be fixed with:

```
sudo apt-get update
sudo apt-get install python-numpy python-matplotlib python-qt4
```

Sometimes persistent dependency problems can be fixed with:

```
sudo apt-get -f install
```

2.9 Start Piccolo Player

Wired only: Check that the connection to the Piccolo works with:

```
ping raspberrypi.local
```

If the Piccolo is not directly connected to the laptop sometimes the Internet Protocol (IP) address can be discovered with:

```
avahi-browse -ar
```

Alternatively, `nmap` may help.

Once the host name or IP address is known, start Piccolo Player.

2.10 View log file

To view the log file, type the following command on the Raspberry Pi:

```
tail -f /var/log/piccolo.log
```

2.11 Download data

Open an Ubuntu File Manager window. Click `Connect to server`. Then type `sftp://pi@raspberrypi.local/mnt`.

This will open a window with the data files.

2.12 Backup

Copy data regularly. A power failure during acquisition may corrupt the USB flash drive.

Warning: A power failiure or other electrical fault could corrupt data. Back it up regularly!

2.12.1 Wired

Use the download data above, or `rsync`.

2.12.2 Wirelsss

Copy directly from the USB flash drive, or use the `download` feature.

2.12.3 Image the SD card

Plug the SD card into the Ubuntu laptop. (Use the adapter.) It should mount (two partitions) automatically. Type:

```
mount
```

This will inform you of the device name. It will be something like `mmcblk0p0`

Type:

```
sudo dd if=/dev/mmcblk? of=sdcard.img
```

Change `/dev/mmcblk0` to whatever is required by the device.

Warning: Type the above very careully! Don't overwrite the laptop's hard drive! Note the device is called `mmcblk0` (and not `mmcblk0p0`).

2.13 Useful terminal commands

`ip addr` gives the IP address of the Raspberry Pi.

`ssh-copy-id` can be used for paswordless logins.

Once Raspbian is installed on the SD card, plug it into the Raspberry Pi and connect the Piccolo's power cable to a battery or power supply. The red *PWR* light-emitting diode (LED) on the Raspberry Pi should be illuminated and the green *ACT* LED should flash as the Raspberry Pi boots.

2.14 Configuring

The Piccolo has two configuration files.

2.14.1 Server configuration file

This holds some information that helps the Piccolo to start up. It is located on the Raspberry Pi's Secure Digital (SD) memory card. To edit this file use a terminal (keyboard or Secure Shell):

```
sudo nano /etc/piccolo.cfg
```

Note: The server configuration file is on the Raspberry Pi's SD card, not the USB flash drive.

JSON RPC

Piccolo Server uses Javascript Object Notation Remote Procedure Calls (JSON RPC) for communication. Configure the uniform resource locator (URL) at which the Piccolo will listen for incoming commands.

Port 8080 is the default.

> **Warning:** No password required to control the Piccolo.

Daemon

Determines whether to run Piccolo Server as a daemon.

Data directory

All acquired data along with the instrument configuration file are stored in this directory.

2.14.2 Instrument configuration file

This holds information about customizable configurations of the Piccolo. It is in the data directory on the USB Flash drive. The name of the data directory is specified in the Server Configuration File. The filename of the instrument configuraiton file is:

```
piccolo.config
```

Channels

The Piccolo has two channels: upwelling and downwelling. This section associates a shutter (1, 2 or 3) and a fibre with each channel. This information is required to ensure that the correct shutter (1, 2 or 3) is openend to record each spectrum. This section also includes information about he optical fibre used for each channel. This information is not used by the Piccolo, but it is saved in the meatadata (header) section of data (Pico) files.

Reverse polarity on a shutter.

If the polarity of a shutter connection has been reversed the shutter will be open when it should be closed; and closed when it should be open. If this happens, try changing Reverse from false to true.

Channel	Shutter	Reverse?
downwelling		no (false)
upwelling		no (false)

Note: The terms *downwelling* and *upwelling* refer to the direction of the optical radiation, not the direction in which the input is pointing. The downwelling radiation is measured by the upward-pointing input whilst the upwelling radiation is measured by the downward-pointing input.

Spectrometers

Some spectrometers, including the Ocean Optics NIRQuest and QE Pro series, are equipped with a thermoelectric cooler to control the temperature of the detector.

> **Warning:** The thermoelectric cooler on QE Pro series spectrometers can heat the detector as well as cooling it. To ensure the detector is cooled (and not heated) choose a detector set temperature below the ambient temperature.

The temperature to which the detector will be cooled can be adjusted. A cooler temperature will reduce the electronic noise in the signal but will increase the (electrical) power consumption. The spectrometer's fan can also be turned on or off. These settings will be applied when the Piccolo's hardware is initialized. The settings will be applied only to the spectrometer with the given serial number.

Calibrations

The instrument configuraiton file has a place to store calibration data.

[[[calibrations]]] [[QEP01651]] # Spectrometer serial number [[[upwelling]]] # Channel wavelengthCalibrationCoefficientsPiccolo = [0.1, 2, 3.1, 0.5, 0.1, 0.5] [[[downwelling]]] wavelengthCalibrationCoefficientsPiccolo = [-0.1, -2, -3.1, -0.5, -0.1] }}

CHAPTER 3

Plotting

This chapter of the Piccolo spectrometer documentation describes how to work with data recorded on the Piccolo. It explains how to go from the *Pico files* recorded by the instrument to a plot of the spectra. It also describes how to work with the metadata (also known as the *headers*) that are included in the data files.

3.1 Pico files

3.1.1 Overview

Every Pico file contains a `Spectra` object which is a list of spectra in the order that they were recorded.

3.1.2 Batches and sequences

The names of files saved in the *data directory* by *Piccolo Server* have the format:

```
title_batB_seqS_type.pico
```

where `title` is a description of the data, *B* is the *batch* number, *S* is the *sequence* number and *type* is the type of spectra in the file. `pico` is the file extension which identifies the file as a *Pico file*.

There are two spectrum types:

- `dark` spectra are recorded with both shutters closed
- `light` spectra are recorded with one shutter open and the other closed

The batch number (`b`) is zero padded to a width of six digits. Similarly, the set number (`s`) is padded to a width of six digits.

A batch of measurements consists of a number of sets. Each set contains of one upwelling spectrum for each spectrometer and one downwelling spectrum for each spectrometer. There are therefore four spectra in each set. Sets are numbered sequentially, starting from 1.

The first and last set in each batch will be dark. These sets are recorded with both shutters closed. These sets should be used to ensure that the dark current has not changed significantly during the recording of the batch (for example due to temperature changes). An additional check can be made by using optical black pixels.

The number of spectra in the batch will be the number of light sets requested plus the two dark sets.

For example, a the first batch in the data named plant with 100 sets will consist of 102 files with the names:

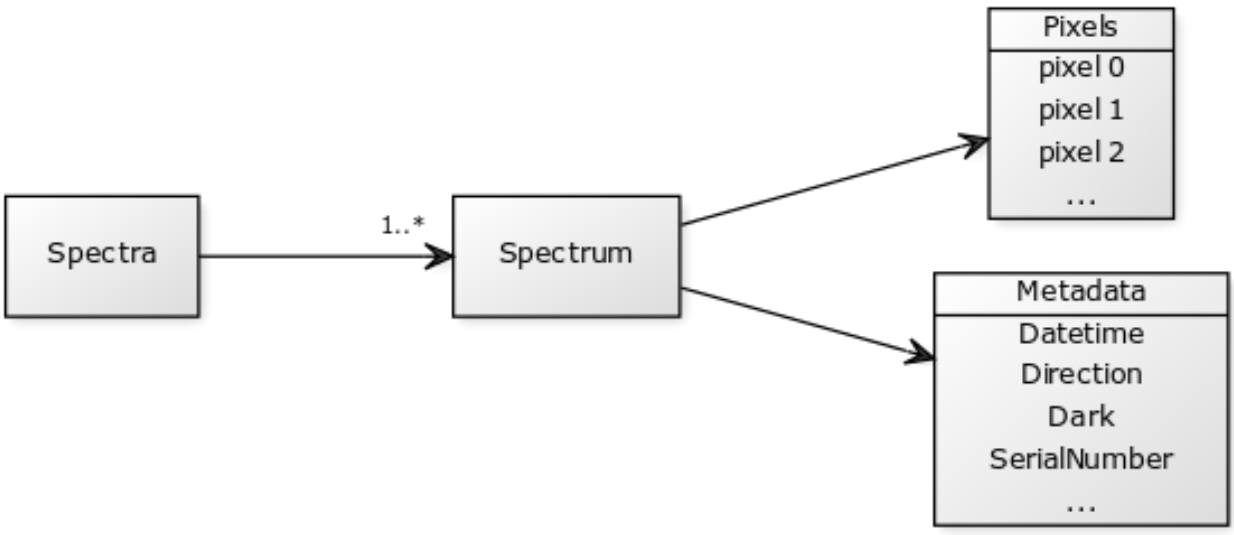
```
plant_000001_000001_dark.pico
plant_000001_000002_light.pico
plant_000001_000003_light.pico
...
plant_000001_000099_light.pico
plant_000001_000100_light.pico
plant_000001_000101_light.pico
plant_000001_000102_dark.pico
```

The next batch of 100 sets will have the names:

```
plant_000002_000001_dark.pico
plant_000002_000002_light.pico
plant_000002_000003_light.pico
...
plant_000002_000099_light.pico
plant_000002_000100_light.pico
plant_000002_000101_light.pico
plant_000002_000102_dark.pico
```

3.1.3 File structure

Each Pico file contains a single set of spectra within a batch. A set will usually consist of two spectra, one upwelling and one downwelling. If more than one spectrometer is connected, then the set contains upwelling and downwelling spectra recorded on each spectrometer.



File format

The format of the text within Pico files is JavaScript Object Notation (JSON).

Note: Despite its name, JavaScript Object Notation is not specific to the JavaScript programming language. It is actually a popular format for structured data which is supported by most programming languages, including Python.

The file contains an object called `Spectra`. This object contains an ordered list. Each specturm has both metadata and a list of pixel values.

The example below shows a Pico file containing a single spectrum. (The list of pixel values has been shortened.)


```
{
 "Spectra": [
  {
   "Metadata": {
    "Dark": true,
    "Datetime": "2015-08-16T17:58:53.048330Z",
    "Direction": "Upwelling",
    "FilterWavelength": "Missing metadata",
    "Grating": "Missing metadata",
    "IntegrationTime": 20.0,
    "IntegrationTimeUnits": "milliseconds",
    "NonlinearityCorrectionCoefficients": [
     0.849144,
     7.94044e-06,
     -7.61693e-11,
     -9.56982e-15,
     6.07747e-19,
     -1.68136e-23,
     2.2759e-28,
     -1.22882e-33
    ],
    "OpticalConfiguration": "B32 0",
    "PolynomialOrderOfNonlinearityCalibration": 7,
    "SaturationLevel": 29000,
    "SerialNumber": "USB2+H16355",
    "SlitSize": "Missing metadata",
    "SlitSizeUnits": "Missing metadata",
    "StrayLightConstant": 0.0,
    "WavelengthCalibrationCoefficients": [
     339.88843,
     0.37405726,
     -1.6419715e-05,
     -1.9436905e-09
    ]
   },
   "Pixels": [
    7,
    10408,
    339,
    302,
    440,
    440,
    440
   ]
  }
 ]
}
```

3.1.4 Metadata

Overview

The `Metadata` object is a (unordered) list of name/value pairs that describe the spectrum. The metadata fields are described in detail below. The fields which are included for each of the Ocean Optics' series of spectrometers are shown in the table. In the table, the symbol ✓ means that this metadata field is available. A means that the spectrometer series or model does not provide this metadata field. Fields which are unavailable will have the value `null` in Pico files.

Field	Flame	HR	Maya2000 Pro	NIRQuest	QEPro	USB
Batch	✓	✓	✓	✓	✓	✓
Datetime	✓	✓	✓	✓	✓	✓
Channel	✓	✓	✓	✓	✓	✓
FiberDiameter	✓	✓	✓	✓	✓	✓
FiberDiameterUnits	✓	✓	✓	✓	✓	✓
Filename	✓	✓	✓	✓	✓	✓
IntegrationTime	✓	✓	✓	✓	✓	✓
IntegrationTimeMaximum	✓	✓	✓	✓	✓	✓
IntegrationTimeMaximumUseful	✓	✓	✓	✓	✓	✓
IntegrationTimeMinimum	✓	✓	✓	✓	✓	✓
IntegrationTimeMinimumUseful	✓	✓	✓	✓	✓	✓
IntegrationTimeUnits	✓	✓	✓	✓	✓	✓
NonlinearityCorrectionCoefficients	✓	✓	✓	✓	✓	✓
PiccoloModel	✓	✓	✓	✓	✓	✓
PiccoloSerialNumber	✓	✓	✓	✓	✓	✓
PiccoloSoftwareVersion	✓	✓	✓	✓	✓	✓
OpticalPixelRange	✓	✓	✓		✓	✓
Run	✓	✓	✓	✓	✓	✓
SaturationLevel	✓	✓	✓	✓		✓
Sequence	✓	✓	✓	✓	✓	✓
SequenceType	✓	✓	✓	✓	✓	✓
SpectrometerManufacturer	✓	✓	✓	✓	✓	✓
SpectrometerModel	✓	✓	✓	✓	✓	✓
SpectrometerSerialNumber	✓	✓	✓	✓	✓	✓
TemperatureHeatsink				✓		
TemperaturePCB	✓	✓		✓		✓
TemperatureMicrocontroller					✓	
TemperatureDetectorActual				✓	✓	
TemperatureDetectorSet				✓	✓	
TemperatureUnits	✓	✓		✓	✓	✓
ThermoelectricCoolerEnabled				✓	✓	
ThermoelectricCoolerStable					✓	
WavelengthCalibrationCoefficientsPiccolo	✓	✓	✓	✓	✓	✓
WavelengthCalibrationCoefficientsSpectrometer	✓	✓	✓	✓	✓	✓
WavelengthUnits	✓	✓	✓	✓	✓	✓

Fields

Batch

Field	Data type	Example	Can be null?
`Batch`	number	`0`	no

The `Batch` metadata field gives the batch number that the spectrum was recorded in. A batch consists of multiple sequences (indicated by the `Sequence` metadata field) and a sequence consists of multiple spectra. The batch number is additionally indicated by the `Filename` metadata field and the filanem itself. It is an integer in the range 0 to 999999. Batches use Python numbering, so the first batch is batch 0.

..note:: When reading a *Pico* file into Python it is generally simpler to derive the batch number from the `Batch` metadata field and not the `Filename`.

Dates and times

Field	Data type	Example	Can be null?
`Datetime`	string	`2015-08-16T17:58:53.048330Z`	no

The `Datetime` metadata field gives the date and time at which the spectrum was recorded, according to the Raspberry Pi's clock.

Note: The timezone for the combined date and time in the `Datetime` field is Coordinated Universal Time (UTC). UTC is also known as *Zulu time* and the letter `Z` is the *UTC designator*. It appears at the end of every combined date and time to indicate that the time zone is UTC.

Warning: The Raspberry Pi does not have a real-time clock, so it does not remember the date and time between interruptions of power. It is possible that the value in this field is incorrect if the clock has not been set.

Channel

Field	Data type	Example	Can be null?
`Channel`	string	`upwelling`	no

Direction is one of the following:

- `upwelling`
- `downwelling`

Note: Upwelling means radiation travelling in an upward direction towards a downward-looking sensor. Downwelling means radiation travelling in an upward direction towards a downward-looking sensor. So the terms upwelling and downwelling refer to the direction the light radiation is travelling in, not the direction the sensors are looking in.

Fiber

Field	Data type	Example	Can be null?
`FiberDiameter`	numeric	`400`	yes
`FiberDiameterUnits`	string	`micrometre`	no

The optical inputs are connected to the Piccolo instrument by fiber optic cables. The `FiberDiameter` metadata field gives the size of the core of the optical fiber which was used in acquiring the spectrum. The units are micrometres (µm), as indicated by the `FiberDiameterUnits` metadata field.

The value of the fiber diameter for each channel (upwelling, downwelling) and each spectrometer is recorded in the instrument configuration file.

Note: The fiber diameter refers to the diameter of the *core* of the optical fiber and excludes the cladding and jacket.

Filename

Field	Data type	Varies	Example	Can be null?
`Filename`	string	yes	`s0003_b000032_dark.pico`	no

The `Filename` metadata field contains the name of the file that the sequence was saved to. The filename has the format shown in the example in the table above. The sequence number is included in the filename, prefixed by the letter `s`. The batch number is prefixed by the letter `b`. This is follwed by a single word to indicate the spectrum type. The type can be `light` or `dark`.

Integration time

Field	Data type	Example	Can be null?	Configurable?
`IntegrationTime`	number	`102.000`	no	yes
`IntegrationTimeMaximum`	number	`3600000.000`	no	no
`IntegrationTimeMaximumUseful`	number	`10000.000`	no	yes
`IntegrationTimeMinimum`	number	`2.000`	no	no
`IntegrationTimeUnits`	string	`milliseconds`	no	no

The integration time used for the measurement. The `IntegrationTimeUnits` field always has the value `milliseconds`.

The integration time is a number greater than zero. It is rounded to to a precision of 0.001 ms (1 µs). This is the supported spectrometers allow setting of the integration time to a precision of 0.01 ms or 0.001 ms.

Note: For most real measurement scenarios it is not necessary to set the integration time as precisely as microsecond-level.

Warning: The precision of the integration time depends on the spectrometer's hardware, specifically the resolution of the integration time counter. Setting an integration time with microsecond does not imply that the actual integration time is so precise.

Nonlinearity correction

Field	Data type	Example	Can be null?
`NonlinearityCorrectionCoefficients`	array	`[0.993...]`	no

This array of numbers contains a number of elements (typically eight) which allow application of a nonlinearity correction formula. They are read from the spectrometer's memory when the spectrometer is initialized.

Optical pixel range

Field	Data type	Example	Can be null?
`OpticalPixelRange`	array	`[22, 3669]`	yes

All spectrometers read a one-dimensional array of pixels from the detector. A subset of these pixels, the optical pixel range, contain spectral information. Other pixels are either unused or masked to use for electronic dark current correction.

Note: The Piccolo is equipped with shutters to measure dark currents and therefore does not use electronic dark current correction.

When plotting spectra, only pixels within the optical pixel range should be plotted.

The table below gives the optical pixel range for each spectrometer. Pixels are numbered such that the first pixel is pixel 0. The ranges specified (first and last) are inclusive.

Spectrometer		All pixels			Optical pixels		
Manufacturer	Model	First	Last	Total	First	Last	Total
Ocean Optics	Flame-S	0	2047	2048	20	2047	2028
Ocean Optics	Flame-T	0	3681	3682	22	3669	3648
Ocean Optics	QE Pro	0	1043	1044	10	1033	1024

Warning: The ranges given in the table above are inclusive. This convention differs from Python array slice notation.

Warning: The Ocean Optics NIRQuest spectrometer does not provide information about the optical pixel range. The `OpticalPixelRange` metadata field will therefore have the value `null`.

Piccolo

Field	Data type	Example	Can be null?
`PiccoloModel`	string	`Doppio`	yes
`PiccoloSerialNumber`	string	`P2_0001`	yes
`PiccoloSoftwareVersion`	string	`2.0.0`	yes

The three metadata fields listed in the table above describe the Piccolo, giving its model name, serial number and the version of the Piccolo software that is running on the instrument.

The model name and serial number of the Piccolo instrument which recorded the spectrum. These values are read from a memory chip on the Piccolo's printed circuit board. Some older Piccolo instruments are not equipped with a memory chip, so these values may be `null`.

Warning: The software version indicated by this field is the version of the Piccolo firmware that is running on the Raspberry Pi. If the instrument is being controlled with Piccolo Player or Piccolo Client, the version numbers must all be identical.

Saturation

Field	Data type	Varies	Example	Can be null?
`SaturationLevel`	number	no	`29000`	no

The saturation levels for the spectrometers are

Model	Saturation	Read from spectrometer?
Flame-S	28,000	yes
USB2000+	29,000	yes
Flame-T	32,000	yes
NIRQuest	63,161	yes
QEPro	200,000	no

Ocean Optics spectrometers use a procedure called autonulling to scale pixel values.

Pixel values read from a spectrometer have a range between zero counts and the saturation level. The saturation level is different for each spectrometer model and cannot be inferred from the resolution of the analogue-to-digital converter.

Sequence

Field	Data type	Example	Can be null?
`Sequence`	number	`5`	no

Gives the sequence number of the spectrum. Sequences are grouped into batches (`Batch`) and numbered sequentially, starting from zero. The sequence number is also given in the `Filename` metadata field, and the filename itself. In the filename it is prefixed with the letter `s` to distinguish it from the batch number. The sequence number is an integer in the range 0 to 9999.

Spectrometer

Field	Data type	Example	Can be null?
`SpectrometerManufacturer`	string	`Ocean Optics`	no
`SpectrometerModel`	string	`USB2000+`	no
`SpectrometerSerial`	string	`USB2+H16355`	no

The spectrometer's saturation level is the maximum pixel value that can occur in a spectrum. A value equal to the saturation level usually indicates that the integration time is set too high.

..note:: The saturation level is related to auto-nulling and pixel scaling.

Run

Field	Data type	Example	Can be null?
`Run`	string	`trees`	no

Each group of batches is given a `Run` name. The files for each `Run` are stored in separate subdirectories of the Piccolo data directory. This is usually on a USB flash drive.

Temperatures

Field	Data type	Example	Can be null?
`TemperatureHeatsink`	number		yes
`TemperaturePCB`	number	`15.892791748046875`	yes
`TemperatureDetectorActual`	number	`-9.887267112731934`	yes
`TemperatureDetectorSet`	number	`-10.0`	yes
`TemperatureMicrocontroller`	number	`34.384765625`	yes
`TemepratureUnits`	string	`degrees Celcius`	no

Gives the temperatures of various components of the spectrometer. Different spectrometers have different temperature sensors, so values may be `null`.

Warning: The `TemperatureDetectorSet` value is read from the spectrometer, not from the Piccolo instrument configuration file. Changing the value in the instrument configuration file should change the value of the `` `TemperatureDetectorSet `` metadata field, though only if the serial number is specified correctly.

Note: The numerical values of the temperatures are not rounded by the Piccolo firmware, they are simply recorded exactly as read from the spectrometer. The number of decimal places does not imply that the temperature readings that precise.

Thermoelectric cooler

Field	Data type	Example	Can be null?
`ThermoelectricCoolerEnabled`	boolean	`false`	yes
`ThermoelectricCoolerStable`	boolean	`true`	yes

Information about the state of the thermoelectric cooler which controls the temperature of the spectrometer's detector. For spectrometers without a thermoelectric cooler this field will have the value `null`.

SequenceType

Field	Data type	Example	Can be null?
`SequenceType`	string	`light`	no

The sequence type can be `light` or `dark`.

Wavelength

Field	Data type	Example	Can be null?	Configurable?
`WavelengthCalibrationCoefficientsPiccolo`	array		yes	yes
`WavelengthCalibrationCoefficients`	array	`[344.99734, 0.218061, -6.331727e-06, -4.1425971e-10]`	no	no

The wavelength calibration coefficients is a list (array) of four values which are used to calculate a the wavelength for every pixel on the spectrometer. These values are determined by calibration, often using a mercury lamp. All Ocean Optics spectrometers are programmed with a set of four wavlength coefficients at the factory. These values are read from the spectrometer by the Piccolo software and saved into data files in the `SpectrometerWavelengthCalibrationCoefficients` metadata field.

In addition to the calibration values programmed into the spectrometer, the Piccolo can be calibrated independently using a mercury lamp. The calibration corrects for any long-term drift in the spectrometers, and any difference between the upwelling and downwelling inputs.

Note: Pixels are numbered from zero, and include non-optical pixels. So a spectrometer with 2048 pixels will have pixels numbered from 0 to 2047 inclusive.

The calirbation equation.

The first element of the list is the zeroth coefficient (symbol *I* or *C0*). It is the wavelength of the first *optically-active* pixel (pixel 0). The first optically-active pixel can be determined using the information in the `OpticalPixelRange` field.

The second element of the list is the first coefficient (*C1*), the third element is the second coefficient (*C2*) and the fourth element is the third coefficient (*C3*).

The information in this field, together with `OpticalPixelRange` and `NumberOfPixels` can be used to calculate the wavelength of each optically-active pixel.

The units of the wavelength calibration coefficients are nanometres.

3.1.5 Pixels

Overview

Method

The set of raw pixel values U read from the spectrometer can be converted to scaled pixel values V if the saturation level S is known:

Saturation equation.png

The range of scaled pixel values (V) is always from 0 counts to 65,535 counts.

$$V = \frac{65535}{S}U$$

3.2 Importing

Pico files can readily be imported into Python, MATLAB or other software.

3.2.1 MATLAB

Using importpico

The MATLAB function `importpico` can read Pico files recorded on the Piccolo into MATLAB. It can read a single file, or multiple files, into a MATLAB structure (`struct`).

Installing importpico

The function is in a MATLAB *m file* called `importpico.m`. It can be downloaded from the `MATLAB` function in the Source tab on Bitbucket. (The Source tab is on the left-hand side of the Bitbucket web page. It looks like a document icon.)

The file `importpico.m` must be saved to a directory which is in the MATLAB search path.

To find out which directories are in the MATLAB ("user") search path type:

```
userpath
```

in MATLAB. Save `importpico.m` to this directory.

To check that MATLAB can find the file type:

```
help importpico
```

If MATLAB can find `importpico.m` this will show a summary of how to use it.

Using importpico

To import a single Pico file type:

```
P = importpico('00001.xml')
```

This will import the file `00001.xml` into a structure called `P` in the MATLAB workspace. The output will be:

```
Importing 00001.xml...

P =

1x4 struct array with fields:

    direction
    dark
    spectrometer
    header
    data
    wavelength
```

In this case `P` is a `1x4 struct`, i.e. it contains 4 spectra. The spectra are saved in the file in the order they are recorded. This is (commit e2d5a12):

1. Upwelling dark
2. Upwelling
3. Downwelling dark
4. Downwelling

For example the second element of `P` should be the upwelling spectrum. Typing `P(2)` gives:

```
ans =

     direction: 'upwelling'
          dark: 'no'
  spectrometer: 'USB2000+'
        header: [1x1 struct]
          data: [2048x1 double]
    wavelength: [2048x1 double]
```

To plot all four spectra in `P` type:

```
plot([P.wavelength], [P.data])
```

Tip: The syntax `[P.data]` is useful for combining multiple vectors into a single matrix. The equivalent in Python is a *list comprehension*.

To import multiple files use wildcards (e.g. `*`) in the filename. For example, typing:

```
Q = importpico('*.xml')
```

will import all XML (now JSON) files in the current working folder into a `struct` called `Q`.

Since each file contains 4 spectra (upwelling dark, upwelling, . . .), if *n* files are imported `Q` will be a nx4 structure.

For example, if 11 files are imported MATLAB outputs:

```
Q =

11x4 struct array with fields:

  direction
  dark
  spectrometer
  header
  data
  wavelength
```

To plot the first spectrum of the fifth file type:

```
plot( Q(5,1).wavelength, Q(5,1).data )
```

3.3 Python

Pico files use a file format based on *Javascript Object Notation* (see *File format*). can be imported into Python using the `json` modules.

See Issue 56.

CHAPTER 4

Programming

The Piccolo has an application programming interface (API).

The API uses CherryPy.

4.1 Bitbucket

This section is optional.

The Bitbucket web site is used to manage the open source Piccolo code. The Piccolo software is written entirely in the Python programming language. The Python modules which control and operate the Piccolo are open source. The hardware drivers for USB spectrometers, radios and shutters are closed source.

As the Piccolo is distributed with open source software users may wish to modify or customize the included software to suit their own applications. *Bitbucket* is a web site that maintains a copy of the Piccolo source code and tracks any changes users' make to the code. It enables users who wish to to share their changes to the Piccolo source code, or additional software they have written, with other users. It is though not essential to use *Bitbucket* in order to operate or program the Piccolo.

Version control software is required to use *Bitbucket*. The software synchronizes a copy of the Piccolo source code on a computer with the Bitbucket server. *Mercurial* (abbreviated `hg` after the chemical symbol for mercury) is the version control system used for the Piccolo. To copy code changes to the Piccolo server it must be installed on the computer. It can also be installed on the Raspberry Pi, if the Pi has a network connection. For instructions on installing Mercurial on the Raspberry Pi see the *Install Mercurial* section of the installation guide.

A Bitbucket account is required.

To commit a change to Bitbucket type:

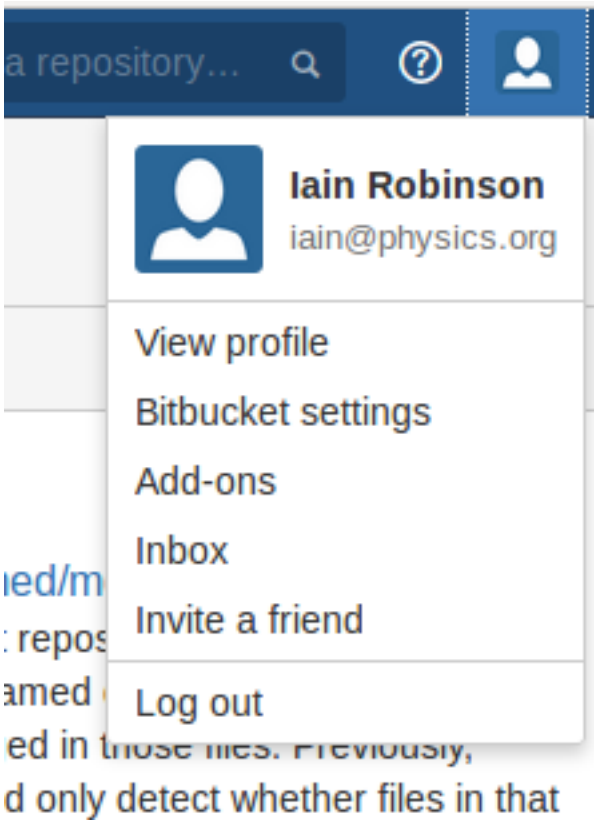
```
hg commit -u "Iain Robinson <iain@physics.org>"
```

replacing `Iain Robinson <iain@physics.org>` with the name and email address used in your Bitbucket account.

This will open a text editor such as *Windows Notepad* or *nano*. Enter a commit message that describes the changes.

Attention: Ensure your name and email address in above `commit` command are identical to those used for your *Bitbucket* account.

The name and email address are used by Bitbucket to connect the commit message with the user account. To find out your usename and email address log into Bitbucket and click the user icon as shown below.



Hint: You edit your Mercurial settings file (`.hgrc`) to include your name, email address, default text editor and other settings.

4.2 Updating Piccolo Server

To clone the Piccolo repository:

```
hg clone https://yourusername@bitbucket.org/teampiccolo/piccolo
```

replacing `yourusername` with the username of your Bitbucket account.

To update a cloned repository:

```
hg pull
hg update
```

To install (or update) Piccolo Server:

```
cd piccolo/Source/Server
sudo python setup.py install
```

4.3 Spectrometers

Spectrometers have a name that is based on their serial numbers.

4.4 Channels

The Piccolo is a dual-field-of-view spectrometer system and therefore has two (or possibly more) light inputs. These are referred to as *channels* and in most configurations there are two channels defined:

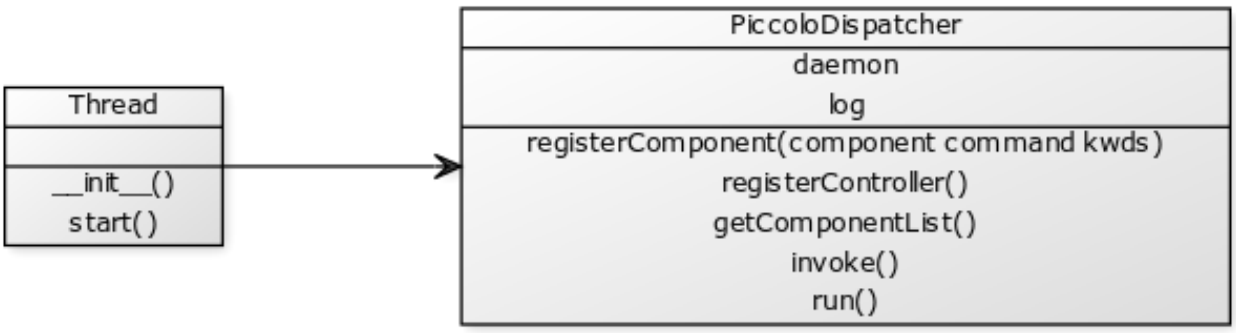
- upwelling
- downwelling

Channel names may be case sensitive.

4.5 Dispatcher

The dispatcher is a Python object (class name `PiccoloDispatcher`) that receives instructions from the laptop or other sources and passes them on the instruments. There can only be one dispatcher running at any one time. It is possible to run the dispatcher in a background thread.

A class diagram for `PiccoloDispatcher` is shown below. As `PiccoloDispatcher` is designed to be run in the background as a separate thread, it inherits from Python's built-in `Thread` class from the `threading` module.



The dispatcher by itself does not handle any hardware. Instead a number of *components* must be registered with it before they can be used. In principle any components can be registered with the dispatcher. Below is an example of the:

1. An upwelling shutter
2. A downwelling shutter
3. An Ocean Optics USB2000+ spectrometer
4. The Piccolo Component
5. The Controller

4.6 Controller

The controller contains:

1. The data directory
2. shutters
3. spectrometers

The interface to the data directory is provided by an object of class `PiccoloDataDir`.

The controller receives instructions and puts them into a queue.

A subclass of `PiccoloController` is `PiccoloControllerCherryPy` which provides an interface for Remote Procedure Calls (RPC) written in Javascript Object Notation (JSON). This provides a web-like API for the Piccolo.

4.7 Scheduler

In logging applications it is required to acquire data at defined times. The scheduler provides an object to do this. Items (jobs) can be added to the scheduler with a specified start time.

4.8 Starting the server

Starting *Piccolo Server* goes through the following setup steps:

1. Reads the *server configuration file*.
2. Sets up a log file.
3. Creates the *data directory*.
4. Initializes (but doesn't start) the dispatcher.
5. Reads the *instrument configuration file*.
6. Initializes the shutters.
7. Registers the shutters with the dispatcher.
8. Finds and initializes all (USB-connected) spectrometers, then registers them with the dispatcher.
9. Sets up the *Piccolo Component* and registers it with the dispatcher.
10. Initializes the controller and registers it with the dispatcher.
11. Starts the dispatcher.
12. Starts the *CherryPy* server with the conroller.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search