**Raspberry Pi Spectrometer**

**Les Wright**

**Old**

[**https://github.com/leswright1977/PySpectrometer**](https://github.com/leswright1977/PySpectrometer)

**New**

[**https://github.com/leswright1977/PySpectrometer2**](https://github.com/leswright1977/PySpectrometer2)

|  |  |
| --- | --- |
| **Large Size** | **Miniature Size** |
| A commercial Diffraction grating Spectroscope <https://www.patonhawksley.com/product-page/benchtop-spectroscope> | A commercial Diffraction grating Pocket Spectroscope: <https://www.patonhawksley.com/product-page/pocket-spectroscope> |
| **Me: Buy**  3B Scientific U19500 Diffraction Grating Spectroscope  [https://www.amazon.in/gp/product/B005LY493O/ ref=ox\_sc\_act\_title\_7?smid=A1CR2SJC09TYY8&psc=1](https://www.amazon.in/gp/product/B005LY493O/%20ref=ox_sc_act_title_7?smid=A1CR2SJC09TYY8&psc=1) | **Me: Buy**  Pocket Diffraction Grating Gemological Spectroscope for Quick Gemstone Identification and GIA Use  <https://www.amazon.in/Diffraction-Gemological-Spectroscope-Gemstone-Identification/dp/B00D49E8XE> |
| A Raspberry Pi Camera (with an M12 Thread) <https://thepihut.com/products/raspberry-pi-camera-adjustable-focus-5mp>  **Me: Buy**  [https://evelta.com/raspberry-pi-camera-module-adjustable-focus-waveshare/?campaignid=19631771445&adgroupid=&network=x&device=c&utm\_source =19631771445&utm\_medium=&gclid=Cj0KCQjwqs6lBhCxARIsAG8YcDhtnEF1Mmzsu PjFSoBOlEuX1isH\_x-jYZYxlrqW8gUL10z4L6Xncu4aAlALEALw\_wcB](https://evelta.com/raspberry-pi-camera-module-adjustable-focus-waveshare/?campaignid=19631771445&adgroupid=&network=x&device=c&utm_source%20=19631771445&utm_medium=&gclid=Cj0KCQjwqs6lBhCxARIsAG8YcDhtnEF1Mmzsu%20PjFSoBOlEuX1isH_x-jYZYxlrqW8gUL10z4L6Xncu4aAlALEALw_wcB) | A Raspberry Pi Camera (with an M12 Thread): <https://thepihut.com/products/raspberry-pi-camera-adjustable-focus-5mp>  **Me: Buy**  [https://evelta.com/raspberry-pi-camera-module-adjustable-focus-waveshare/?campaignid=19631771445&adgroupid=&network=x&device=c&utm\_source =19631771445&utm\_medium=&gclid=Cj0KCQjwqs6lBhCxARIsAG8YcDhtnEF1Mmzsu PjFSoBOlEuX1isH\_x-jYZYxlrqW8gUL10z4L6Xncu4aAlALEALw\_wcB](https://evelta.com/raspberry-pi-camera-module-adjustable-focus-waveshare/?campaignid=19631771445&adgroupid=&network=x&device=c&utm_source%20=19631771445&utm_medium=&gclid=Cj0KCQjwqs6lBhCxARIsAG8YcDhtnEF1Mmzsu%20PjFSoBOlEuX1isH_x-jYZYxlrqW8gUL10z4L6Xncu4aAlALEALw_wcB) |
| ***A CCTV Lens with Zoom (M12 Thread)*** (Search eBay for F1.6 zoom lens) **NO IR FILTER**  **Les Wright Suggestion: Buy**  [https://www.amazon.in/KAZRU-Megapixel-Varifocal-Security-Cameras/dp/B0CBMY6LW9/ref=sr\_1\_5?keywords=zoom+ lens+m12&qid=1689450170&sr=8-5](https://www.amazon.in/KAZRU-Megapixel-Varifocal-Security-Cameras/dp/B0CBMY6LW9/ref=sr_1_5?keywords=zoom+%20lens+m12&qid=1689450170&sr=8-5)  KAZRU 5.0 Megapixel Varifocal 6-22mm CCTV Camera Lens Manual Focus Zoom 1/2.5" Lens CCTV MTV IR Lens for Security Cameras HD IP Cameras F1.6 M12 Mount (No IR Filter), Opens in a new tab  **Me: Buy**  <https://www.amazon.in/gp/product/B06XFXV6PP/ref=ox_sc_act_title_1?smid=AMCK6A78OQ8A9&psc=1>  Xenocam 9-22mm 1/3" IR F1. 4 CCTV Video Vari-focal Zoom Lens for CCTV Security Camera, Opens in a new tab | M12x0.5 F2.0 Fixed 12mm Focal length Lens  **Me: Buy**  <https://www.amazon.in/gp/product/B07D3RD4BM/ref=ox_sc_act_title_1?smid=A3DRYI1U78K7M8&psc=1> |
| Screenshot  Screenshot  Screenshot | Screenshot  **Pimoroni PIM370 PIM370 Development Board Hyperpixel 4.0 Hi-Res Display Raspberry Pi 4" TFT 800x480 Pixels**  <https://www.tanotis.com/products/pimoroni-pim370-pim370-development-board-hyperpixel-4-0-hi-res-display-raspberry-pi-4-tft-800x480-pixels?gclid=Cj0KCQjwqs6lBhCxARIsAG8YcDj1cEcBV5E0Pbpvre1dQP8K_WjGZ4NLQyjeXk_FbENkJMhiXtrL9d0aAnn0EALw_wcB>  Screenshot |

Check for RPi camera lens

<https://rees52.com/22-raspberry-pi-camera>

Calibration Laser Pen Tools

**Laser Pointer Pen Beam Light 5Mw Lazer**

Laser pen 405nm, 532nm, 650nm.

|  |  |
| --- | --- |
| Laser pen 405nm | <https://www.amazon.in/gp/product/B079BHVT6D/ref=ox_sc_act_title_5?smid=A3TFCS5VGMP6EE&psc=1> |
| Laser pen 532nm | <https://www.amazon.in/gp/product/B08YWXV7NL/ref=ox_sc_act_title_1?smid=A2939L6GG2SUE3&psc=1> |
| Laser pen 650nm | <https://www.amazon.in/gp/product/B0B2W99D2X/ref=ox_sc_act_title_2?smid=A3CYMJZJHWN7A0&psc=1> |
| Helium Nion (He Li) Laser 632.8nm |  |
| DPSS 532 nm |  |
| Fluorescent Light Tube |  |
| UV Light  High Risk Typographic Stamp. Typographic Sign, Badge Or Icon ... |  |
| Mercury Vapor Lamp  High Risk Typographic Stamp. Typographic Sign, Badge Or Icon ... | <https://www.amazon.in/LAB-Lighting160-Mercury-Vapor-Lamp/dp/B07MJJWBMM> |
| Deuterium Lamp  High Risk Typographic Stamp. Typographic Sign, Badge Or Icon ... |  |
| Xenon flash lamp |  |
| White paper (coumarin dye) |  |
| Multi-Spectrum LED Grow Lights | * Red Light: Around 620-700nm wavelength. Red light is crucial for promoting flowering and fruiting in plants. * Blue Light: Around 400-500nm wavelength. Blue light is essential for vegetative growth and overall plant development. * Green Light: Around 500-600nm wavelength. Although plants do not heavily rely on green light for photosynthesis, it can play a role in enhancing plant growth and canopy penetration. * Ultraviolet (UV) Light: Around 280-400nm wavelength. UV light, in small doses, can stimulate plant defenses and increase essential oil production. |
| Fluorescent Lights | * Cool White: Typically emits a broad spectrum of light with peaks in the blue and green range, covering a range of approximately 400-700nm. Cool white fluorescent lights are often used for general plant growth, but they may lack specific wavelengths required for optimal results. * Warm White: Similar to cool white, warm white fluorescent lights emit a broad spectrum covering a range of approximately 400-700nm, with a slightly more pronounced peak in the red-orange region. They are also used for general plant growth but may have limitations in delivering targeted light wavelengths for specific growth stages. |

Top of Form

Bottom of Form

# leswright1977/PySpectrometer2

## Latest commit

[@leswright1977](https://github.com/leswright1977)

[leswright1977](https://github.com/leswright1977/PySpectrometer2/commits?author=leswright1977) [Update README.md](https://github.com/leswright1977/PySpectrometer2/commit/14ca18754911ecf8a92240b655568eadac86c371)

[14ca187on Nov 28, 2022](https://github.com/leswright1977/PySpectrometer2/commit/14ca18754911ecf8a92240b655568eadac86c371)

## Git stats

* [**128** commits](https://github.com/leswright1977/PySpectrometer2/commits/main)

## Files

Type

Name

Latest commit message

Commit time

[media](https://github.com/leswright1977/PySpectrometer2/tree/main/media)

[Add files via upload](https://github.com/leswright1977/PySpectrometer2/commit/47984c8e20139f6d37e95060ce5331cbc41bc9c0)

9 months ago

[src](https://github.com/leswright1977/PySpectrometer2/tree/main/src)

[Update PySpectrometer2-USB-v1.0.py](https://github.com/leswright1977/PySpectrometer2/commit/e10ddb7f0ec9c8292cbc7757a12d3b09022367ba)

9 months ago

[LICENCE](https://github.com/leswright1977/PySpectrometer2/blob/main/LICENCE)

[Create LICENCE](https://github.com/leswright1977/PySpectrometer2/commit/727c99d2899d9b2e5b5581383a13cce54336d4ae)

9 months ago

[README.md](https://github.com/leswright1977/PySpectrometer2/blob/main/README.md)

[Update README.md](https://github.com/leswright1977/PySpectrometer2/commit/14ca18754911ecf8a92240b655568eadac86c371)

8 months ago

## [README.md](https://github.com/leswright1977/PySpectrometer2#readme)

# PySpectrometer2

The second incarnation of the Spectrometer project!

This is a more advanced, but more flexible version of the original program. It changes the spectrometer from educational 'toy' to serious instrument, which can easily compete with commercial units costing thousands of dollars!

This program, hardware design, and associated information is Open Source (see Licence), but if you have gotten value from these kinds of projects and think they are worth something, please consider donating: <https://paypal.me/leslaboratory?locale.x=en_GB> This project is a follow on from: <https://github.com/leswright1977/PySpectrometer>

This readme is accompanied by youtube videos, showing how to build and use the spectrometer! Visit my Youtube Channel at: <https://www.youtube.com/leslaboratory>

There is a video on this project here: <https://youtu.be/SCp9T8NKfnM>

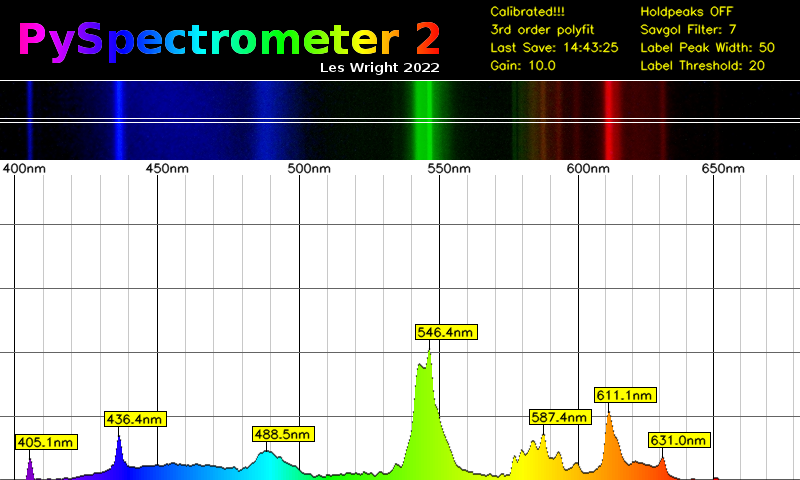
## Rationale for the new build

Recent changes in the OS (Bullseye) broke the old version e.g broken video, broken dependencies and so on. PySpectrometer v3.0 was hacked and fixed as of 3.1), however I have been thinking about a rewrite of this software for a while, so here it is!

Accuracy has been significantly improved by implementing multi-wavelength calibration and a Polynomial regression data fit to compute wavelengths as precisely as possible across the measured range.

Tk has been dropped as the GUI to allow easier maintainability, extendability and flexibility. The new interface is coded entirely in OpenCV, and whilst things like trackbars and buttons are now dropped in favour of keybindings, this frees up screen real estate, and allows the support of resizing without fuss, and fullscreen mode is now supported!

In Fullscreen mode on Raspberry Pi systems, the interface is designed to fit 800 x 480 screens, which seem to be a common resolution for RPi LCD's, paving the way for the creation of a stand alone benchtop instrument.

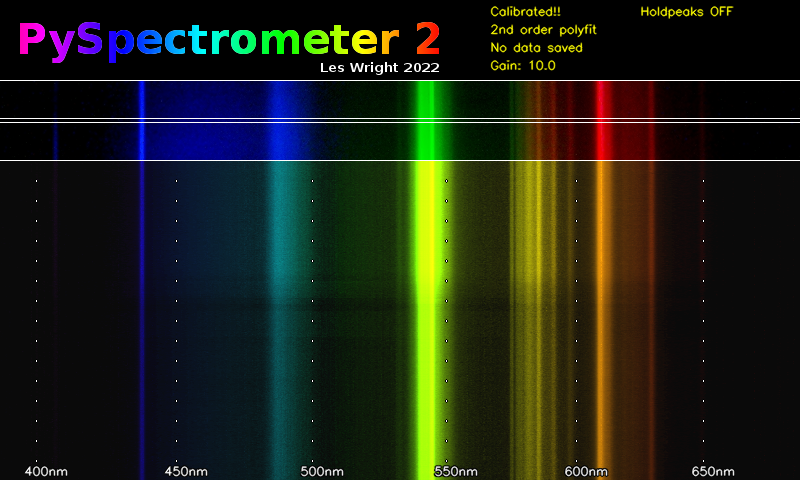
[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/calibrated.png)

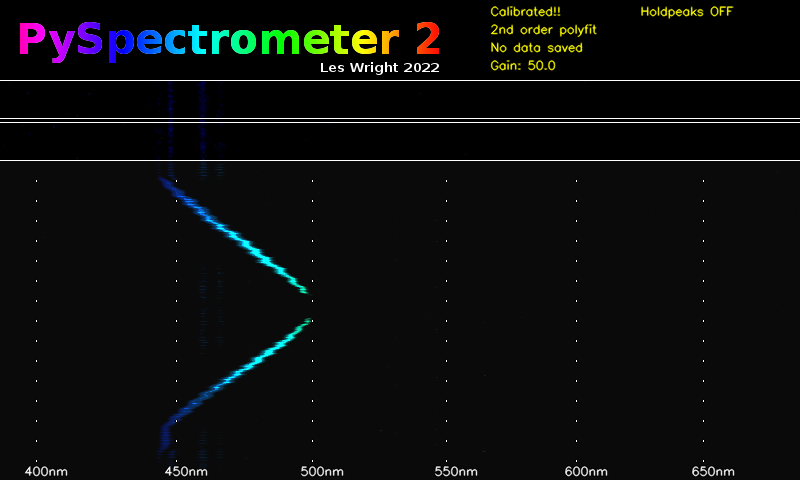
### Whats new:

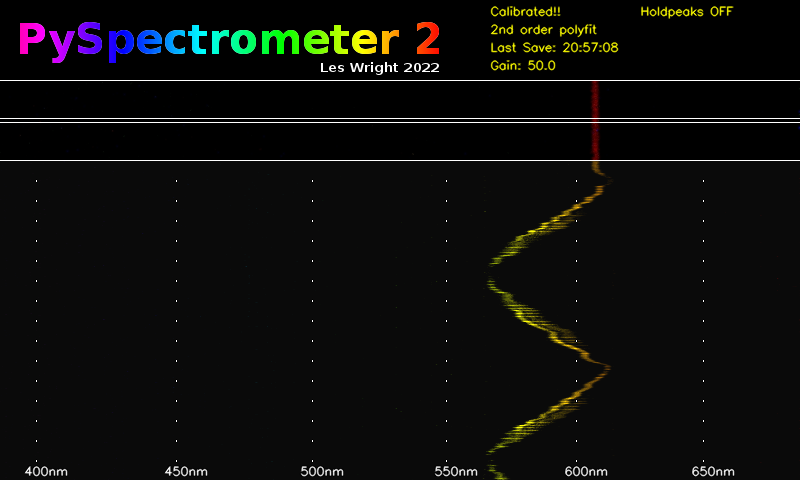
* Higher resolution (800px wide graph).
* 3 row pixel averaging of sensor data.
* Fullscreen option for the Spectrometer graph.
* 3rd order polynomial fit of calibration data for accurate measurement.
* Improved graph labelling.
* Labelled measurement cursors.
* Optional waterfall display for recording spectra changes over time.
* Key Bindings for all operations.
* Analogue Gain control for the Picam

The functionality of the previous version has been retained, including peak hold, peak detect, Savitsky Golay filter, and the ability to save graphs as png and data as CSV.

A very cool addition to this project is a Waterfall display! This allows the recording of change in in wavelength over time. For my purposes this is especially useful for Dye Lasers, however this may be of use to those observing the spectra of light sources that may change in intensity or wavelength.

Example waterfall display of a fluorescent lamp: [](https://github.com/leswright1977/PySpectrometer2/blob/main/media/waterfall-20221013--210412.png)

Waterfall display of a Dye laser being tuned (Coumarin-1). The jagged tuning curve is because this laser was hand-tuned! [](https://github.com/leswright1977/PySpectrometer2/blob/main/media/waterfall-20221013--205708.png)

Below the tuning curve of Rhodamine 6G [](https://github.com/leswright1977/PySpectrometer2/blob/main/media/waterfall-20221013--205840.png)

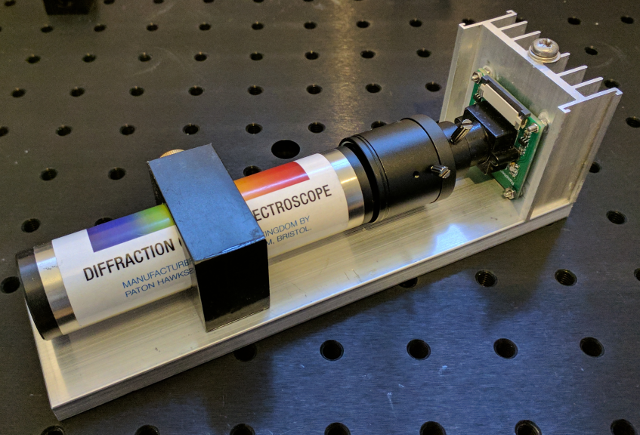
# Hardware

The hardware is simple and widely available and so should be easily to duplicate without critical alignment or difficult construction. The hard work was developing the software.

Resolution/accuracy seems to be down to the nanometre with a well built and calibrated setup, which is excellent for the price of the hardware, especially when you consider the price of commercial components such as the Hamamatsu C12880MA breakout boards which run north of 300 bucks, and has a resolution of 15nm. Of course, this build is physically much larger, but not enormous!

## Standard Spectroscope

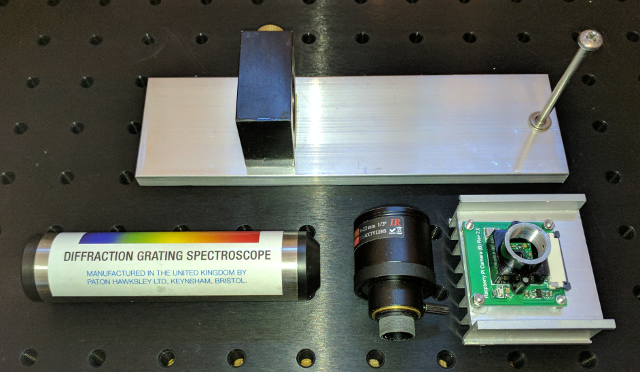
For the standard build, I used a pocket spectroscope(link below) coupled into a picamera by means of a zoom lens. The job is simple: Mount the zoom lens on the picam, and mount the spectroscope in front, and focus the camera on the spectrum, until it is sharp and clear. Use either daylight (which has pronounced Fraunhoffer lines) or a fluorescent lamp, which has pronounced emission lines. The following command will help you: **libcamera-hello -t 0**

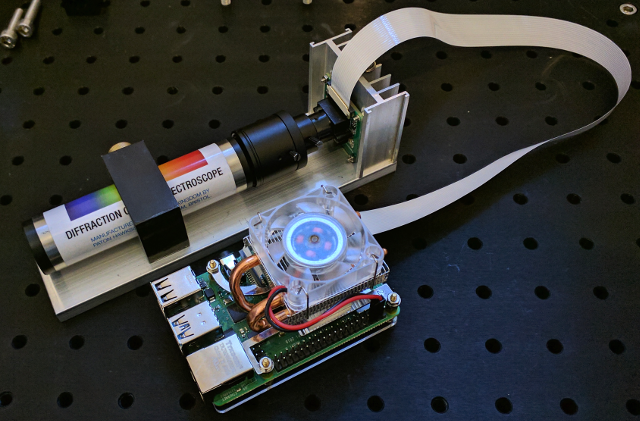
[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/scope.png)

The hardware consists of:

* A commercial Diffraction grating Spectroscope <https://www.patonhawksley.com/product-page/benchtop-spectroscope>
* A Raspberry Pi Camera (with an M12 Thread) <https://thepihut.com/products/raspberry-pi-camera-adjustable-focus-5mp>
* A CCTV Lens with Zoom (M12 Thread) (Search eBay for F1.6 zoom lens)

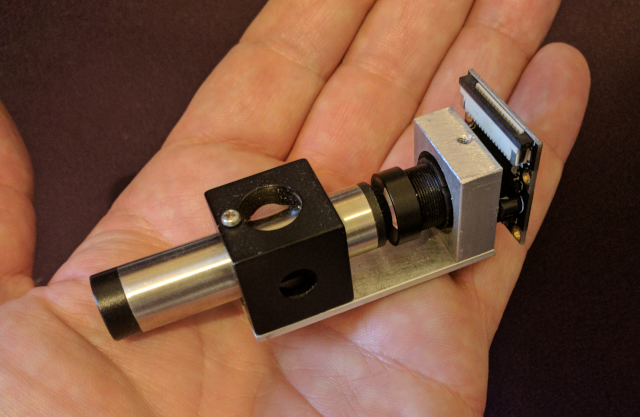
Everything is assembled on an aluminium base (note the Camera is not cooled, the heatsink was a conveniently sized piece of aluminium.)

[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/parts.png)

[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/pi.png)

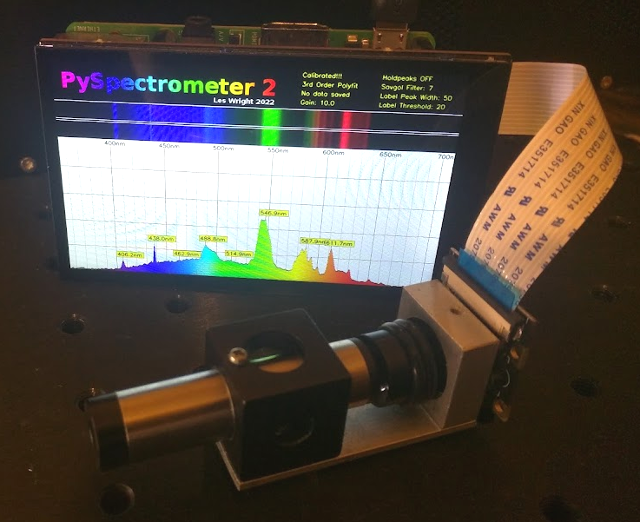
## Miniture Spectroscope

The build is as simple as the standard version, however, uses a miniature pocket spectrometer.

[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/minispect.png)

* A commercial Diffraction grating Pocket Spectroscope: <https://www.patonhawksley.com/product-page/pocket-spectroscope>
* A Raspberry Pi Camera (with an M12 Thread): <https://thepihut.com/products/raspberry-pi-camera-adjustable-focus-5mp>
* M12x0.5 F2.0 Fixed 12mm Focal length Lens: (search eBay)

## Stand alone unit

[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/mini.png)

Above, a compact unit built with a Hyperpixel 4 inch screen, running on fullscreen mode (800 x 480)

<https://shop.pimoroni.com/products/hyperpixel-4?variant=12569539706963>

## Custom units

There is nothing to stop you building a spectrometer head with a couple of razor blades, a diffraction grating, a couple of lenses and a Picam! The software should work just the same as shown in this readme!

# User Guide

## Key Bindings:

### Graph Display Controls

* t/g = Analogue Gain up/down (not available on USB version, see below for alternative camera controls)
* o/l = savpoly up/down
* i/k = peak width up/down
* u/j = Label threshold up/down
* h = hold peaks

### Calibration and General Software

* m = measure (Toggles measure function. In this mode a crosshairs is displayed on the Spectrogram that allows the measurement of wavelength)
* p = record pixels (Toggles pixel function (Part of the calibration procedure) allows the selection of multiple pixel positions on the graph)
* x = clear points (Clear selected pixel points above)
* c = calibrate (Enter the calibration routine, requires console input)
* s = save data (Saves Spectrograph as png and CSV data. Saves waterfall as png.
* q = quit (Quit Program)

## Starting the program

First, clone this repo!

In /src you will find:

* PySpectrometer2-Picam2-v1.0.py (PySpectrometer for Raspberry Pi)
* PySpectrometer2-USB-v1.0.py (USB version of this program (This is for USB Cameras See end of Readme)).
* specFunctions.py (A library of functions including: Wavelength to RGB, SavGol filter from Scipy, Peak detect from peakutils, readcal and writecal.

## Dependencies

Run: **sudo apt-get install python3-opencv**

**Also note, this build is designed for Raspberry Pi OS Bullseye, and will only work with the new libcamera based python library (picamera2)** It will **not** work with older versions of Raspbery Pi OS. You **will** however be able to use PySpectrometer2-USB-v1.0.py with an external USB camera with other Operating Systems.

To run the program, first make it executable by running: **chmod +x PySpectrometer2-Picam2-v1.0.py**

Run by typing: **./PySpectrometer2-Picam2-v1.0.py**

Note to also display the waterfall display, run with the option: **./PySpectrometer2-Picam2-v1.0.py --waterfall**

To run in fullscreen mode (perform calibration in standard mode first), run with the option: **./PySpectrometer2-Picam2-v1.0.py --fullscreen**

When first started, the spectrometer is in an uncalibrated state! You must therefore perform the calibration procedure, but at this stage you should be able to focus and align your camera with your spectroscope using the preview window. Is is expected that red light is on the right, and blue-violet on the left. An excellent choice for this is daylight, as well defined Fraunhoffer lines are indicative of good camera focus.

## Calibration

This version of the PySpectrometer performs Polynomial curve fitting of the user provided calibration wavelengths. This procedure if done with care with result in a precision instrument!

When light from a diffraction grating falls upon a flat sensor the dispersion of light is not linear, and so calibration with just two data points (as in the old version of this software) will result in inaccurate readings. This nonlinearity is likely compounded by additional nonlinearities introduced by the camera lenses. To address the nonlinearity, the user must provide the pixel positions of at least 3 known wavelengths (4 to 6 is highly recommended for high accuracy!). This information is then used by the program to compute the wavelengths of every single pixel position of the sensor.

Where 3 wavelengths are used for calibration, the software will perform a 2nd order polynomial fit (Reasonably accurate)

Where 4 or more wavelengths are used, the software will perform a 3rd order polynomial fit (Very accurate)

Assuming your physical spectrometer setup is rigid and robust (not held together with gravity, tape or hot glue!), calibration will only need to be done once (Data is saved to a file called: caldata.txt), and therafter when any change is made to the physical setup.

Direct your Spectrometer at a light source with many discrete emission lines. A target illuminated by Lasers would be an excellent (though very expensive!) choice! An inexpensive alternative is a Fluorescent tube.

You should be able to identify several peaks in your graph, now you need to match them up with known good data. For serious work I would recommend an academic resource such as: <https://physics.nist.gov/PhysRefData/Handbook/Tables/mercurytable2.htm> however in the spirit of citizen science (and because fluorescent lamps are somewhat variable in manufacture), I would recommend this wikipedia article to get you started: <https://en.wikipedia.org/wiki/Fluorescent_lamp> have a read, and scroll down to the section called: Phosphor composition. In here you will find emission spectra of a variety of fluorescent lamps!

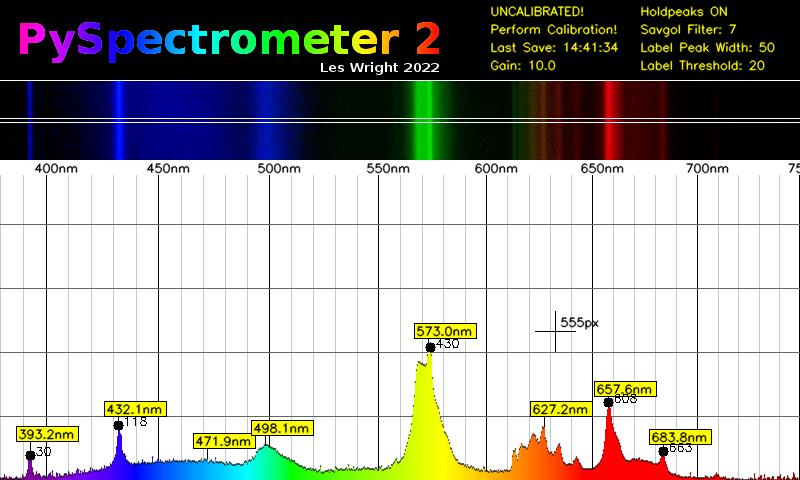
Likely the most useful is this graph: <https://commons.wikimedia.org/wiki/File:Fluorescent_lighting_spectrum_peaks_labeled_with_colored_peaks_added.png>

These are the notable visible peaks:

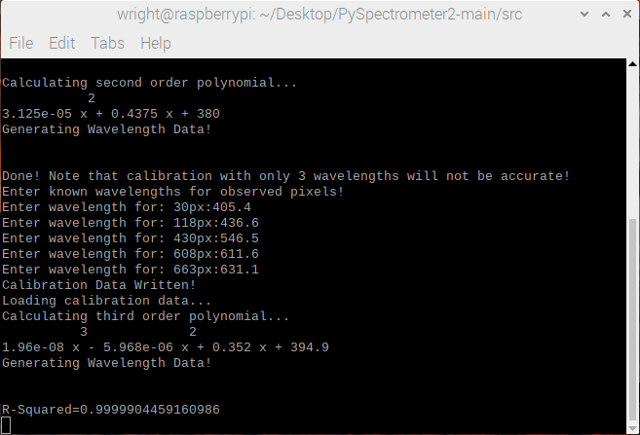
* 1 405.4 nm (Mercury)
* 2 436.6 nm (Mercury)
* 3 487.7 (Terbium)
* 4 542.4 (Terbium)
* 5 546.5 (Mercury)
* 12 611.6 (Europium)
* 14 631.1 (Europium)

Once you have identified some peaks, at least 3, but even better 4 to 6, first press 'h' to toggle on peak hold, this will stabilize the graph, and even allow you to switch off the light source!

Press the 'p' key on the keyboard. This will toggle on the pixel measuring crosshairs, move the crosshairs to each of your peaks, and click once the crosshairs are aligned with the 'flagpole' of the wavelength marker. Rinse and repeat for your identified peaks. (Note it makes sense to do this from left to right!)

[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/pointsadded.png)

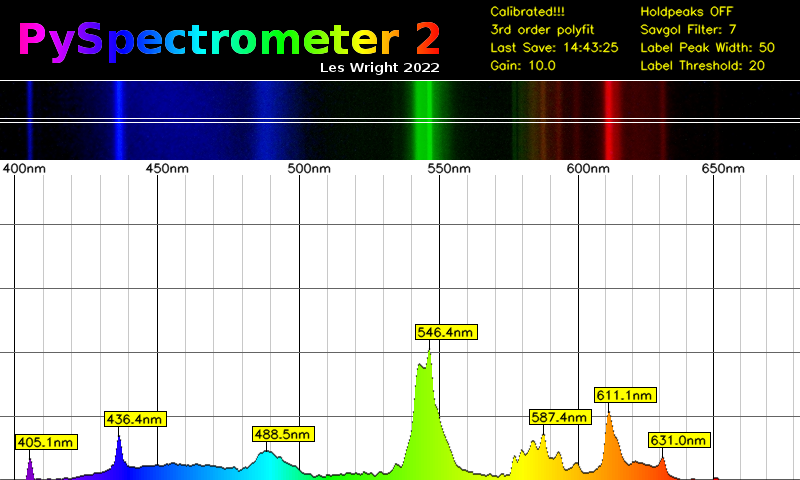
Once you have selected all of your peaks, press 'c' and turn your attention to the terminal window. For each pixel number, enter the identified wavelength.

[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/console.png)

Once you have entered the wavelengths for each data point, the software will recalibrate the graticule and its internal representation of all the wavelength data. In the console, it will print out the value of R-Squared. This value will give an indication of how well the calculated data matches your input data. The closer this value is to 1, the more accurately you recorded your wavelengths! for example a six nines fit (0.999999xxxx) is excellent, and 5 nines is good. If it is a way off, one or more of your identified wavelengths may be incorrect, and you should repeat the calibaration procedure! (Press 'x' to clear the points, and repeat the calibration procedure)

### Check your work

Refer back to the graph from the wiki, can you identify with a reasonable degree of accuracy other peaks? (bearing in mind your fluorescent lamp may differ from the one on the wiki!).

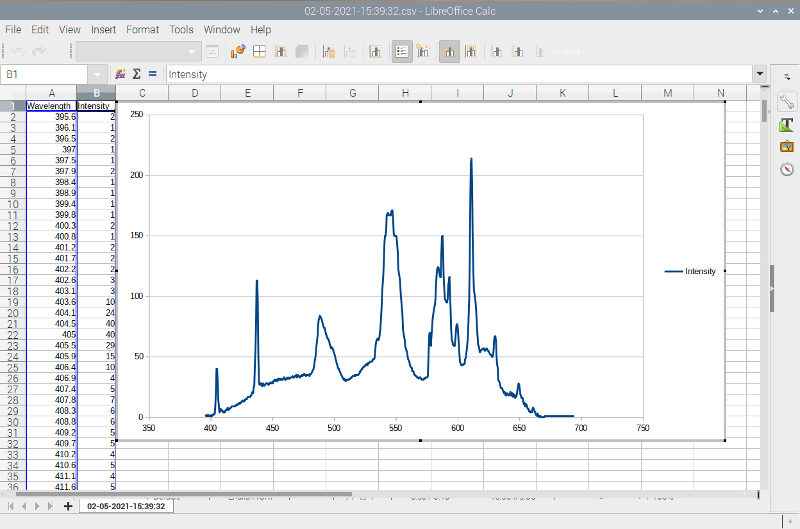
[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/calibrated.png)

In the screenshot above above, a well defined peak (not used as a caibration value) at 587.4nm has been detected. Referring to the Wiki this is listed at 587.6nm, only 0.2nm off with a five nines calibration! :-) For unlabelled peaks, pressing 'm' will toggle on the measurement crosshairs, that display wavelength for any given position.

Calibration data is written to a file (caldata.txt), so calibration is retained when the program is restarted.

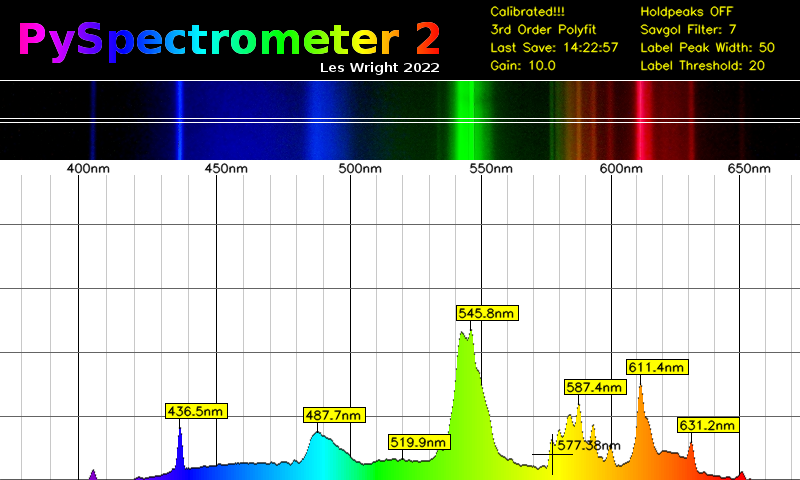
# Saving Data

Pressing 's' will save all data. It saves graph and waterfall data as PNG images, with date and time as part of the filename. Additionally it saves graph data as CSV that can then be opened in other programs such as OpenOffice on the Pi.

[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/csv.png)

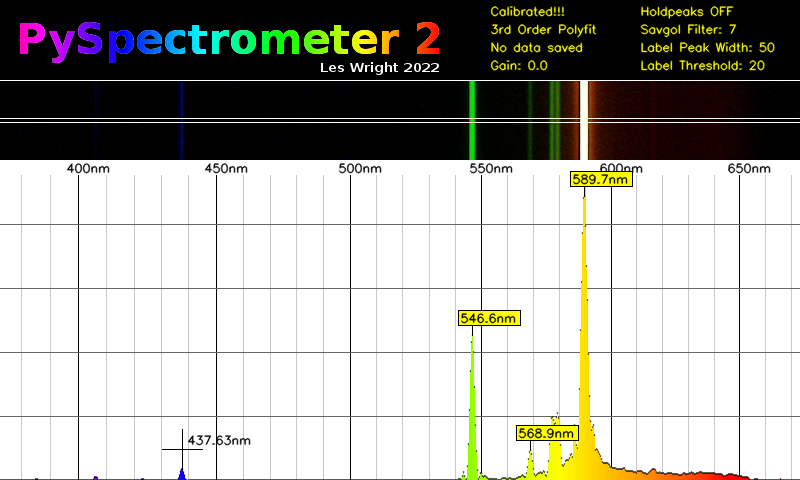
# Arbitrary measurement

Pressing the 'm' key will toggle a measurement cursor. This can be used (once the intrument has been calibrated) to arbitratily measure any point on the graph. The following screenshot shows the measurement of a possible Terbium or Mercury line at 577nm

[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/terbium-measure.png)

Below: Another example of measurement cursors on a weak line at 437nm. This is from a high pressure Sodium lamp as is in fact likely Mercury (HP Sodium lamps contain Xenon, Mecury and Sodium).

The prominent peak at 589nm is Sodium. Of the other peaks: 546.6nm is Mercury, 568.9nm is Sodium, 577nm and 579nm (unlabelled, but just to the left of the 589nm Sodium peak) are both Mercury.

[](https://github.com/leswright1977/PySpectrometer2/blob/main/media/high-pressure-sodium.png)

# USB Camera Version

A version of the software is provided for those who wish to use third party USB cameras with the Pi, or even a USB camera with any other Linux box!

The following command line options must be considered:

* Video device number
* Framerate

**Note: the expected resolution from USB cameras is 800x600, other resolutions will cause the software to crash!**

For an external USB camera, first find the device by issuing: **v4l2-ctl --list-devices**

Once you have determined this, you can run the program. For example if your camera is /dev/video3 and you require a framerate of 15fps you would issue:

**./PySpectrometer2-USB-v1.0.py --device 3 --fps 15**

If you want fine control over camera settings use guvcview: **sudo apt-get install guvcview**

You can run guvcview at the same time as the spectrometer software, so long as you disable guvcview preview, like this:

(assuming your device is /dev/video3)

**guvcview --device /dev/video3 --control\_panel**

This will allow you to control:

* Brightness
* Contrast
* Saturation
* Gain
* Other settings (depending on camera, see note below)

Note: Guvcview is a more sensible choice for camera control, than trying to shoehorn in USB camera functionality into this code. The Python OpenCV libary has limited and oftentimes broken support for the huge variety of USB cameras out there, and so direct control with a tried and tested utility makes sense!

# Future work:

It is planned to add inputs of some description, to allow the use of buttons and knobs to control the Spectrometer.

The type of inputs will depend on oddly the type of screen!

The hyperpixel displays consume all of the GPIO on the Pi, however buttons could easily be provided if they talk HID.

DSI displays could be used, however seemingly that might require the user roll back to legacy camera support!

HDMI displays can be used, and this would free up all the GPIO.

A one size fits all approach would be a HID device, there is plenty of choice, including using a Teensy or an Arduino and buttons, or even a number pad with custom keycaps.

I am thinking of implementing something approaching autocalibration, though this might be difficult implement for all use-cases.