**PyCam manual**

**1. INTRODUCTION**

PyCam is a software written by Dr Thomas Wilkes to control the PiCam instrument developed in The University of Sheffield; see [Wilkes et al. (2016)](https://www.mdpi.com/1424-8220/16/10/1649) and [Wilkes et al. (2017)](https://www.mdpi.com/2072-4292/9/1/27) for more details on the early prototype development and deployment of this instrument. *Note: this software will not work with earlier versions of the PiCam instrument (pre-2022), which have their own PyCamUV software.* The software is written in Python 3 and provides a GUI (tkinter) interface in the hope of creating a user-friendly environment for acquiring and processing UV SO2 camera data. Any issues/bugs or suggestions can be submitted to the github project (<https://github.com/twVolc/PyCamPermanent>) or enquiries can be made directly to [tcwilkes1@sheffield.ac.uk](mailto:tcwilkes1@sheffield.ac.uk); I will do my best to respond as quickly as possible. Please note, at the current time the instrumentation and software are still in the early stages of rigorous field-deployment testing, therefore unexpected behaviour may be encountered. Also, GUI formatting seems to be a real pain and takes time that has been better spent on software functionality, so please excuse how things look – they may not transfer to other screen resolutions perfectly…

Much, but not all, of PyCam implements the pyplis (<https://www.mdpi.com/2076-3263/7/4/134>) package as a backend, since a lot of time has already been put into developing this package and it is quite a powerful tool for SO2 camera data analysis. [iFit](https://www.sciencedirect.com/science/article/pii/S037702731930647X) is also used for processing of spectra for SO2 column density retrievals – this follows similar principles to differential optical absorption spectroscopy (DOAS) but does not require acquisition of a Fraunhofer reference spectrum, since it models down-welling radiation.

**2. HARDWARE SETUP**

If using a 128GB microSD must expand filesystem after copying disk image (unless 128GB disk image is used). sudo raspi-config > advanced options > expand filesystem. All space should then be available to the pi.

Jumper cable on GPIOs for Pi start up should be placed from GPIO23 (physical pin 16) on the Witty Pi (Pi 1) to GPIO3 (physical pin 5) on Pi 2. Jumper cable on GPIOs for Pi shut-down should be placed from GPIO16 (physical pin 36) on Pi 1 to GPIO21 (physical pin 40) on Pi 2.

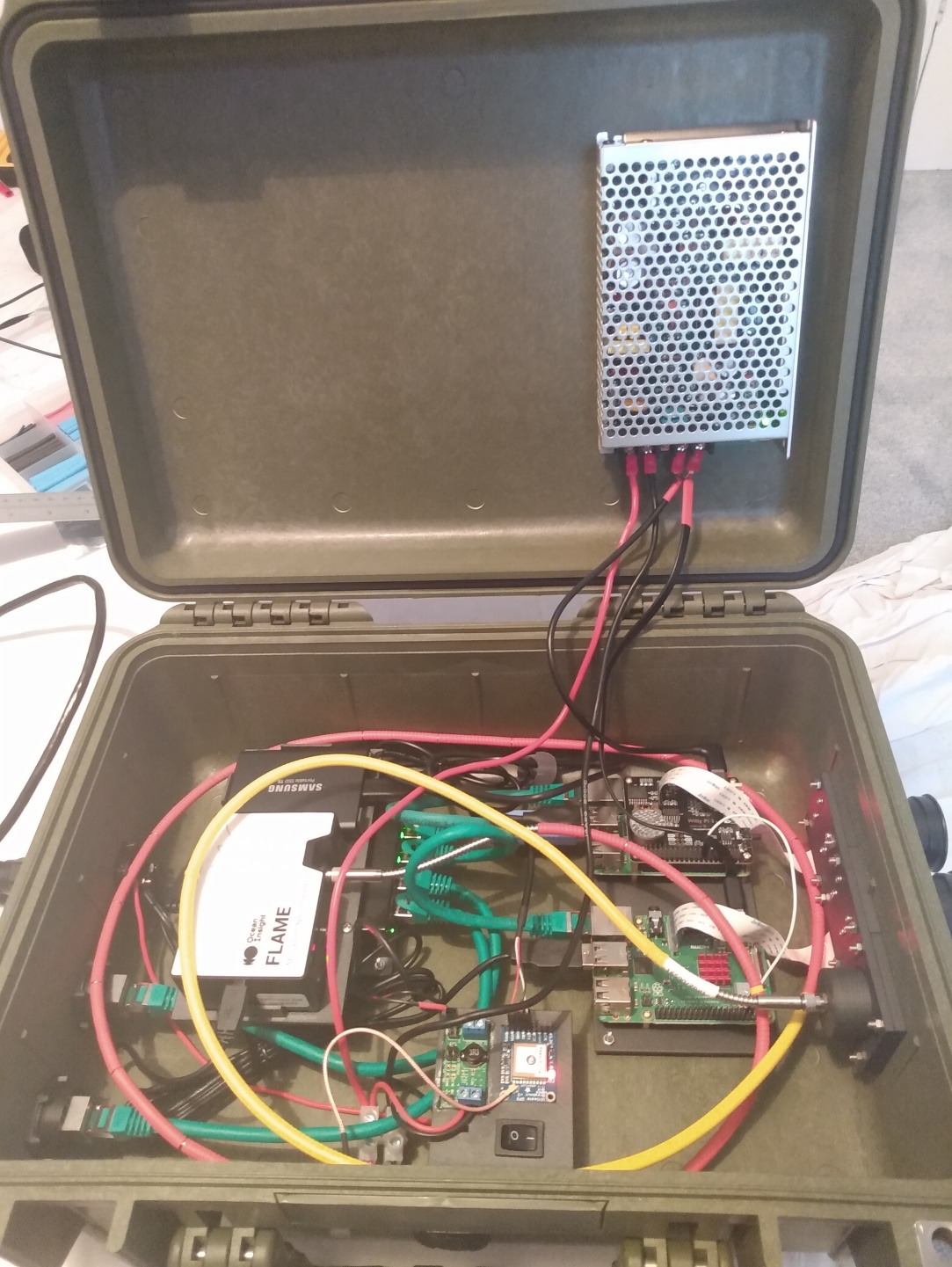
***GPS connections***

Red - VIN

Black - GND

Green - RX

White – TX

***Instrument layout***

**SSD (1TB)**

Data storage. Connected to Pi 1.

**Voltage regulator**

(12V input – 5V output)

Powers 2 R-Pis

**Flame spectrometer**

Camera calibration. Expensive!!

Connected to Pi 2.

**Pi 1 (Master Pi)**

Main controller of instrument. Has a WittyPi HAT to control on-off sequence of Pis.

Controls acquisition from Filter A camera.

Careful!

**External power connector**

2-pin Bulgin connector

**Network switch**

Internal Pi communication and external communication

**WittyPi Off/On button**

Pressing this will turn off both Pis – can be useful for removing SSD safely.

The Pis remain powered, so will turn back on with the next scheduled on time.

**External ethernet connections**

For laptop control or data transmission

**Pi 2**

Controls acquisition from Filter B camera and spectrometer. All data is immediately transmitted to Pi 1 (no data saved on this Pi).

**Voltage regulator**

(12V input – 5V output)

Powers network switch. Somewhat unnecessary as I now have the big regulator, but I have kept this one just for the network switch for now.

**Optical Fibre**

Couples light to spectrometer from lens. Fragile – don’t bend too tightly. Also the metal outer layer conducts electricity, so if it comes into contact with electronics it could short things and kill them. The heat shrink tubing is to insulate, so stops this, but be careful!

**Master power switch**

Turns off all power to the instrument. It is best to shutdown the R-Pis before turning this off.

**GPS**

For syncing time. Provides time in UTC, not local time! Connected to Pi 1.

**IMPORTANT!!! Please be careful when closing the Peli Case, wires from the power regulator could become trapped.**

***Instrument Optics***

****

**Spectrometer**

Connected to Flame spectrometer via optical fibre. Acquires spectra (≈250-400 nm). Used to calibrate camera apparent absorption using DOAS/iFit.

**Filter B**

Connected to Pi 2. Off-band images where SO2 absorption is negligible (330 nm).

**Filter A**

Connected to Pi 1. On-band images where SO2 absorbs strongly (310 nm).

**3. SOFTWARE INSTALLATION**

***Pycharm***

I use Pycharm for my python projects and would recommend you do the same. It is quite a useful IDE which for example will help pull new versions of the PyCam software from the GitHub repository. The free (choose Community on the download page) version is available here: <https://www.jetbrains.com/pycharm/download/#section=windows>

Then run the installer.

***Git***

Download and install git from <https://git-scm.com/download/win>

***Conda***

Install conda – makes it easier to install other python packages. I start by installing Miniconda <https://docs.conda.io/en/latest/miniconda.html>.

Then find the anaconda prompt (search in windows menu “Anaconda”), open it and create a new environment (Python 3.8.2 definitely works):

**conda create --name pycam python=3.8.2**

Activate this new environment (so you are working in it):

**conda activate pycam**

Then install conda version 4.8.3, which definitely works:

**conda install conda=4.8.3**

Some versions of conda seem to fail when installing pyplis. They give a malformed “~” error. Can use “conda install conda=4.8.3” to install specific version. 4.8.3 definitely works.

***Pyplis installation***

Next install pyplis as this will also install a large number of dependencies, such as numpy and cv2. Pyplis can be installed with conda (from conda command prompt):

**conda install -c conda-forge pyplis**

IMPORTANT: May need to manually go into pyplis and add volcano source information file to pyplis package. Need to add it to the ./pyplis/data/my\_sources.txt file, which should be relatively straightforward. Source\_id is caps sensitive, so just add two source IDs, one with capital letter to start and one without. ***UPDATE: volcano source information can be saved through the GUI – go into geometry configuration and edit the volcano info, this will save directly to the pyplis my\_sources.txt location.***

Pyplis currently has a bug meaning it can’t download volcano data, as the URL it used to use not longer exists. Because of this we need to manually update a file. Navigate to your Miniconda folder, then go to:

*envs > pycam > Lib > site-packages > pyplis > data*

In the file “my\_sources.txt” add the following lines to the bottom of the file:

source\_ids:Villarrica,villarrica

name:Villarrica

country:Chile

lat:-39.42

lon:-71.93

altitude:2847.0

END

Currently pyplis has an error which stops cell calibration from working if using an empty cell as part of the calibration (in my experience). To get this to work, after installing pyplis go to line 467 in calib\_base.py and change min(cds) to max(cds).

***Other module installs***

Some pip installs - use command **pip install x** where x is the package name (write it exactly as displayed below). Make sure this is done within the pycam conda environment:

* ttkthemes
* shapely
* paramiko
* scikit-image
* pyproj==2.6.1.post1 (PyProj is installed with pyplis, but the version currently installed (3.3.0) doesn’t work for some functions. Reverting to this version seems to fix issues!

***PyCam installation***

Clone pycam project from github: <https://github.com/twVolc/PyCamPermanent.git>. This is currently a private repository but can be made available on request – you will need to sign into your github account and I can then make this available. Clone the repository with:

**git clone** [**https://github.com/twVolc/PyCamPermanent.git**](https://github.com/twVolc/PyCamPermanent.git)

This will create a new directory (in whatever directory you write the command), called PyCamPermanent. Within this there is a pycam folder which contains all of the code.

Copy ifit and ifit\_ld folders from memory stick into pycam folder – these are not downloaded with the git clone.

Filenames in pycam/tests/test\_data/test\_spectra/ must be changed – for some reason github has them as lower case (I can’t get this to change…) but should have first letter upper case e.g. plume must be changed to Plume and dark to Dark.

In the pycam\_gui.py file (“./pycam/gui/pycam\_gui.py”) line 6 gives an absolute path to “pyproj”. I have recently tested removing this and I’m not sure I lost any functionality, so it may be fine, but if an error does occur, this path simply needs to be changed to the path on your local machine. This is within your miniconda folder then relative to the miniconda folder it will be located:

*.\envs\pycam\Lib\site-packages\pyproj*

Simply add your miniconda path to the start of this path.

***Running PyCam***

The PyCam software should now be ready to run. Open PyCharm. Select Open Project and navigate to the PyCamPermanent folder. This should open the project. IT may try to create a new virtual environment when you open this – click cancel. We need to set the project interpreter to the conda environment we have already created.

*File > Settings > Project: PyCamPermanent > Python Interpreter*

Select the cog in the top right corner and click “Add”. On the left hand side select “Conda Environment”. Select “Existing Environment”. In the interpreter section click the three dot on the right hand side to search for an interpreter. Now navigate to your Miniconda folder, go to “envs” then “pycam”. Scroll down this folder and there should be a python.exe file – select this. Click OK and this should all be loaded into the project as the interpreter.

Open the “run\_pycam.py” file. Right click inside the file and select “Run File in Python Console”. Once you have done this the first time, subsequently you will be able to run PyCam GUI from the Green triangle button in the toolbar.

**Setup**

After starting PyCam GUI, it will probably be necessary to adjust settings so that figures fit the screen better. Go to:

*File > Settings > GUI settings*

Play around with this. Firstly I recommend changing the resolution DPI, which adjusts all of the figures simultaneously. Then individual figures could be adjusted afterwards if this doesn’t fix everything. Font size my also need to be changed too. ***NOTE: Changes aren’t made until the software is restarted.***

**4. INSTRUMENT CONTROL**

First, it is important to avoid confusion related to the instrument and software, so here initially some important information is defined. There are 2 distinct python softwares, both of which may be referred to as PyCam in some form:

* The first is the software run on the instrument itself, on the Raspberry Pis. Primarily this is used for control of data acquisition and also for setting up sockets for interfacing with the second (GUI) software. NOTE: This software can run independently of the second software, therefore allowing the instrument to run in a “headless” state. The main script of this software is *pycam\_masterpi.py*. In this document this may be referred to either as pycam (lower case) or the script name itself *pycam\_masterpi.py.*
* The second software provides a GUI for interfacing with the instrument and processing data. It allows the user to control instrument settings, download data from the instrument, process data etc… In this document we refer to this as PyCam GUI.
* “PyCam” is then used to refer to the software as a whole, encompassing pycam and PyCam GUI.

**4.1 Starting the instrument**

Once the instrument has been turned on at the main switch the Raspberry Pi will be running. However, this will not start the instrument software needed for data acquisition (script name: *pycam\_masterpi.py* and associated subprocess scripts). This software can be scheduled to start at a specific time each day (needed for permanent/automated installations), or it can manually be started through the PyCam GUI. For scheduled runs see *section 4.5*. To manually run the instrument software, first ensure you have an Ethernet connection to the instrument then from the PyCam GUI:

*Instrument > Commands > Run pycam*

There are two options for *Run pycam,* defining whether or not *automated capture* should be initiated immediately on software start-up, i.e. if *(with automated capture)* is chosen, the instrument will immediately begin acquiring data (with its most recent acquisition settings) once the software has started. Note: if you chose *(without automated capture)*, automated capture can still be started later through the PyCam GUI (see *section 4.3*).

For **debugging** it may be useful to run pycam\_masterpi.py through SSH, so any errors it throws can be read from the shell. To do this, open an SSH session (see *section 4.2*) on Pi 1, move to the pycam scripts directory “/home/pi/pycam/scripts/” then run *pycam\_masterpi.py* by typing the command: **python3 ./pycam\_masterpi.py**. If you wish to run instrument with automated capture simply add the flag 1 to the end of the command: **python3 ./pycam\_masterpi.py 1**

**4.2 Connecting to the instrument**

***4.2.1 Pi networking***

Pi 1 IP: 169.254.10.180 Controls on-band camera, GPS, external SSD storage, external communications.

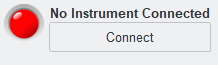
Pi 2 IP: 169.254.10.178 Controls off-band camera and spectrometer

Username: pi

Password: raspberry

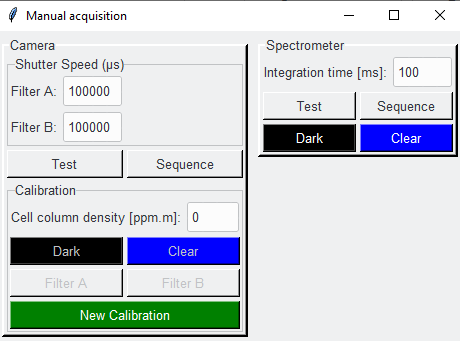
For debugging and some other functions it can sometimes be useful to SSH into the instrument Raspberry Pis using the above IP addresses. The IP address should not change unless specific ad-hoc edits to the instrument have been made (not recommended – may break the instrument). A software such as **Putty** ([www.putty.org](http://www.putty.org)) can be used to facilitate SSHing. This access is available whether or not the *pycam*\_masterpi.py software is running on the instrument, as long as the instrument is switched on and you have a network (ethernet) connection with the instrument.

***4.2.2 Standard GUI connection***

To connect to the instrument use the “Connect” button in the top left region of the PyCam GUI. ***NOTE: This connection will only work at times when the pycam software (pycam\_masterpi.py) is running on the instrument, since this connection is made directly through the pycam sockets – this is not the same as standard SSH connections to the instrument***. If connection is successful the indicator will turn **green**. If it is not successful, check the port and IP address of the connection; these can be edited through: File > Settings > Connection. The port may change - if the instrument has been unsuccessful creating a server on one port it will try the next in the list. As stated above, the IP address should not change.

**4.3 Data acquisition**

There are a few ways to control image acquisition. First, make sure you are connected to the instrument through the PyCam GUI (see *section 4.2*), with a green indicator light.

*****4.3.1 Manual acquisition***

*Instrument > Acquisition settings > Manual acquisition*

For quick and simple acquisitions, or on discrete field campaigns, it may be useful to use manual acquisitions. Here the user can simply control shutter speed/integration time and acquire other data types, such as gas cell calibrations.

***4.3.2 Automated acquisition***

*Instrument > Acquisition settings > Start Automated Acquisition*

This will begin automated acquisition on the instrument, running on the settings that are currently uploaded to the instrument. With this function, unlike *Manual acquisition*, data is not automatically uploaded to the GUI computer (acquisition begins in a “headless” mode). To upload images simply start data transfer (either before or after acquisition has started):

*Instrument > Data Transfer > Start transfer*

This just sets up an FTP directory watcher that will grab an Images/Spectra that are saved on the instrument (see also *section* *4.4.3*). ***Note: this will start by uploading any pre-existing images that have not yet been pulled off the instrument.***

Automated acquisition can then simply be stopped using:

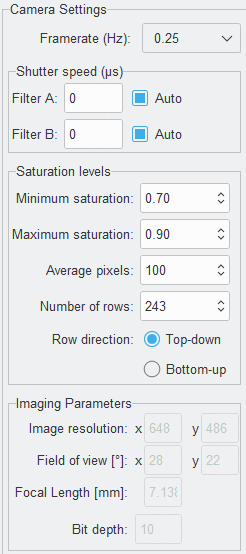
*Instrument > Acquisition settings > Stop Automated Acquisition*

***4.3.3 Automated acquisition settings***

To update the instruments current settings use:

*Instrument > Acquisition settings > Update all instrument settings*

This will update all (camera and spectrometer) acquisition settings, which are outlined in the figures below. Whilst automated acquisition is underway it is possible to update these settings “on-the-fly”.

*Camera window*

Camera acquisition rate (Hz). e.g. 0.25 Hz = 1 image per 4 seconds

Shutter speed of camera. **Note: it is defined in microseconds not milliseconds, following the camera API.** If “Auto” is ticked this shutter speed will be ignored.

If ticked, the camera will automatically adjust its shutter speed as light conditions change. Adjustments are made based on the parameters in the “Saturation levels” section.

Minimum saturation level of analysed pixels before shutter speed is increased – to increase signal.

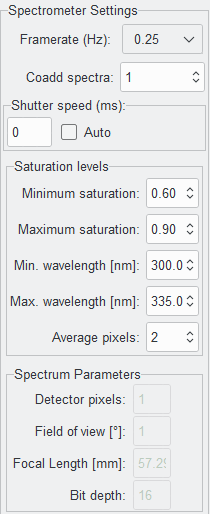
Maximum saturation level of analysed pixels before shutter speed is decreased – to decrease signal.

Number of pixels to take average of for saturation determination. This finds the average of the *X* most saturated pixels in the image and determines the saturation level relative to the 10-bit (1024 levels) maximum of the detector. This value is compared to the user-defined minimum and maximum saturation levels to determine whether shutter speed should be adjusted.

Number of rows to take pixels from in saturation analysis. This may be useful in areas where volcano flanks contain snow, therefore producing very bright pixels that we would want to exclude from the saturation analysis – typically we are only interested in sky/plume saturation. To analyse all rows set this value to the y-resolution of the detector.

Direction from which to take the “Number of rows” extraction. In most cases this would be from Top-down to extract only sky pixels and omit ground pixels.

*DOAS window*



Number of pixels to take the average of for saturation analysis. Analysis takes the average of the *X* most saturated pixels in the spectrum (within the min.-max. wavelength range) and determines the saturation level relative to the 16-bit (65554) maximum level of the detector. This value is compared to the user-defined minimum and maximum saturation levels to determine whether shutter speed should be adjusted.

Defines the maximum wavelength for saturation analysis to take place – above this wavelength pixels are not analysed and therefore could become saturated.

Defines the minimum wavelength for saturation analysis to take place – below this wavelength pixels are not analysed and therefore could become saturated.

Maximum saturation level of analysed pixels before shutter speed is decreased – to decrease signal.

Minimum saturation level of analysed pixels before shutter speed is increased – to increase signal.

If ticked, the spectrometer will automatically adjust its shutter speed as light conditions change. Adjustments are made based on the parameters in the “Saturation levels” section.

Spectrometer shutter speed (also called integration time. **Note: unlike the cameras, this is defined in milliseconds not microseconds.**

Number of spectra to add together for each data point – can be used to improve signal to noise ratio.

Spectrometer acquisition rate (Hz). e.g. 0.25 Hz = 1 image per 4 seconds

***NOTE: for both camera and spectrometer, shutter speeds will not change if auto shutter speed is enabled. You also may need to*** *Update all instrument settings* ***twice to move from auto shutter speed to a manual shutter speed. This is because on the first update if the instrument attempts to set the new shutter speed before auto shutter speed has been turned off, the shutter speed set will fail. So the first ping will turn off the auto shutter speed whilst the second will update to the new user-defined shutter speed. (This is basically a bug that I haven’t got around to fixing yet…)***

***4.3.4 Retrieving current instrument settings***

*Instrument > Acquisition settings > Retrieve current settings*

To find the current settings on the instrument, use the above path in the GUI menu. This will only work when connected to the instrument. Once clicked, the PyCam GUI will query all settings on the instrument and then update the Camera Settings and Spectrometer Settings panes.

**4.4 Data and data transfer**

Data on the Pi is located in two places, the SSD (1TB) and the microSD card that contains the operating system of the Pi (128GB – probably ≈100GB available for data).

***4.4.1 SSD***

To access the SSD data the SSD must be mounted to the Raspberry Pi; *pycam\_masterpi.py* performs this mount procedure automatically on start-up, so if this script is running the SSD will be accessible. The SSD can also be mounted with the PyCam GUI:

*Instrument > Data Transfer > Mount SSD*

This essentially runs the linux command:

**sudo mount -o uid=pi,gid=pi <dev\_path> <mount\_path>**

where <dev\_path> is the path to the device, which is typically (but this could change) “/dev/sda1”, and <mount\_path> is the location to mount the drive, which we set to “/mnt/pycam”. To manually mount the SSD this command can therefore be written in an SSH session.

***NOTE: When run from the GUI a notification will pop-up afterwards saying the SSD is successfully mounted. This is currently just an indicator that the command was successfully sent to the Pi, but does not actually check that the mount was successful, sorry… So if downloading is then unsuccessful it may be because the command failed to mount the SSD for some reason.***

Data on the SSD is located at the path “/mnt/pycam/data/” where it is divided into subdirectories for each day. One way of checking that the device is correctly mounted is to list what exists in the “/mnt/pycam” directory from an SSH session: **ls /mnt/pycam**. If this command returns nothing, the SSD device may not be mounted, whereas if it lists directories (most importantly a Data directory), then it is currently mounted.

***4.4.2 MicroSD***

Data on the microSD card is located a “/home/pi/pycam/Images/”; this is the initial save location of all data. This directory is always available (no mounting required) and data here is not divided into subdirectories. Note that this storage is considerably smaller than the SSD storage, so this directory may not contain all data acquired by the instrument since the last download.

There are a few ways that data can be transferred from the instrument to the local machine. Without the use of the PyCam GUI, FTP software (e.g. winSCP) can be used to connect to the Master Pi via its IP address and download the data.

***4.4.3 Transfer and process new data***

Data is downloaded into the ./pycam/Data folder on the local machine, separated into Images and Spectra.

Using the GUI, data can be transferred and processed simultaneously using:

*Instrument > Data Transfer > Start data transfer*

This command starts an FTP client that watches the microSD data storage and downloads any images/spectra that are available there. ***NOTE: this method will delete the files from the microSD storage once they have been downloaded, to ensure space is freed up. The SSD data is not touched using this method.***

***4.4.4 Full data download***

Alternatively, bulk download of data from the SSD can be performed with:

*Instrument > Data Transfer > SSD full download*

Ensure the SSD is mounted before attempting this, otherwise it will fail to find the data. This download method does not delete data after download, to ensure that large amounts of data aren’t inadvertently destroyed without a back-up. After checking that the download has been successful, and perhaps creating another back-up, data on the SSD can be deleted using one of two commands:

*Instrument > Data Transfer > Clear SSD data*

*Instrument > Data Transfer > Free space on SSD*

The latter generates a prompt to define how much space on the SSD the user wishes to create. Data will be deleted (oldest first) until the user-defined amount of space is available on the SSD.

A manual data download could be performed using an FTP client and connecting to the Pi 1 (169.254.10.180).

***4.4.5 Removing the SSD for data download***

***CAUTION!!! Please only use this method if you are certain you know what you are doing – incorrect removal of the SSD could lead to corruption/loss of data***

***Note: The SSD is actually automatically unmounted when pycam (pycam\_masterpi.py) stops running, so if you are accessing the data at a time when pycam is not on, you can unplug the SSD straight away (although it is still best to check the SSD has been unmounted).***

The SSD USB connection must only be removed if it has first been unmounted from the Raspberry Pi – otherwise data may be corrupted/lost.

The SSD can be unmounted with the PyCam GUI:

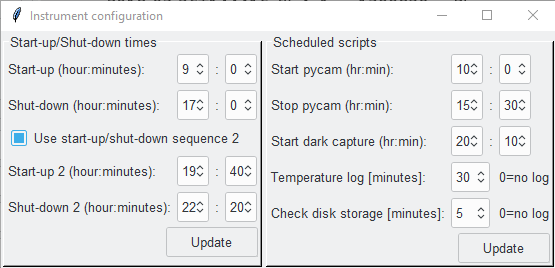
*Instrument > Data Transfer > Unmount SSD*

If this is run whilst pycam\_masterpi.py is running, the unmount process may fail because the SSD device is in use. However, the GUI will (incorrectly) still say that the SSD has been unmounted successfully, so it is important to check if this is true. The best way to do this is use an SSH session (e.g. Putty) and list the mount directory to see if anything is inside: **ls /mnt/pycam**. If this command returns nothing then the SSD should be safe to remove from the Pi.

If the device remains mounted, first stop any automated acquisition and then stop the instrument software:

*Instrument > Commands > Stop pycam*

The SSD should be unmounted automatically when the instrument pycam software is stopped. If it is still mounted, use the GUI unmount command, which should now work. If the SSD still remains mounted, the instrument can be shutdown entirely, which would allow safe removal of the SSD device. Instrument shutdown can be performed on the WittyPi HAT by pressing the black button – this will shut-down both Pis, but won’t power off the entire system.

**4.5 Scheduling**

*Instrument > Configure*

There are a few functions of the instrument that can be run on a schedule, to be used in permanent installations. All settings are contained within the GUI window found through the above pathway.

**ALL TIMES ARE IN UTC**

***4.5.1 Start-up/Shut-down times***

It is possible to create 1 or 2 start-up/shut-down cycle(s) on the instrument per day. In general, the second start-up may be useful if wanting to start the instrument briefly at night to acquire a set of dark images. To use the second start-up/shut-down sequence simply tick the box *Use start-up/shut-down sequence 2*. If the instrument is required to be running 24 hours a day (for instance, for continuous data transfer if the transmission band width isn’t high enough to transfer in real-time with acquisition) simply make the start-up and shut-down times of the first sequence the same time, and ensure the *Use start-up/shut-down sequence 2* is not ticked. **However, I would recommend always having one start-up/shut-down sequence per day, even if it is only turning the instrument off for a very short period – this will help reset the system so that any bugs that may be encountered are hopefully corrected.**

**IMPORTANT! It looks like the witty pi somehow takes into consideration daylight saving time. There may come a point in the year where suddenly things become an hour out of sync – this may stop pycam from running if the camera instrument isn’t turned on before the scheduled pycam start. Keep an eye on this! If this does become an issue, to reconfigure simply update the schedule time after the clocks change and the pi should be set back onto the correct schedule. Note, the crontab timing for script scheduling (*section 4.5.2*) doesn’t seem to have this same issue, so won’t adjust in the same way.**

***4.5.2 Scheduled scripts***

*Start pycam*

This will start *pycam\_masterpi.py*  in automated capture mode, such that data will be acquired as soon as this script has started up.

*Stop pycam*

Stops *pycam\_masterpi.py*. Note: When *pycam\_masterpi.py* is not running it is not possible to communicate with the instrument through the PyCam GUI, therefore only SSH communication (see *section 4.2*) and FTP data transfer (see *section 4.4*) will be possible.

*Start dark capture*

Start a dark capture sequence, acquiring an image at every shutter speed available to the instrument in automated capture mode. ***NOTE: in manual capture mode more shutter speeds are available to the camera via manual setting rather than the list of speeds available to automated capture.*** To avoid running dark capture simply set it to run at a time when the instrument is not turned on.

*Temperature log*

Uses the temperature sensor on the WittyPi (pi HAT) to log temperature every *x* minutes. At the moment it’s not entirely clear whether this will provide ambient temperature in the box or the temperature the pi is running at, but it may prove useful if issues are encountered with the instrument.

*Check disk storage*

Schedules the run frequency of *check\_disk\_space.py*. This script checks the disk space on the **microSD card** on the master Pi. This script allows a maximum storage of 100GB by default. Once this storage is exceeded the script will start deleting data (starting with the oldest), until the space is below the threshold. ***NOTE: this checks SD card space, not SSD space. Data deleted from SD card will still be available on the SSD; however, it is advised that data is downloaded frequently so that the microSD doesn’t fill up too frequently, leaving no backup for the SSD dataset.***

**5. DATA PROCESSING**

**5.1 Display only mode**

*Processing > Display only mode*

This option allows you to transfer data across to your computer/laptop without automatically processing all of the data as it arrives. Images and spectra will still be displayed as they arrive, so you can see what the data looks like as it is downloaded. There are a few advantages to having this option ticked:

1. The software is less likely to encounter bugs as it deals with the incoming data, so data download and display should continue to work smoothly.
2. If processing settings are computationally expensive, the GUI may freeze up or simply not be able to keep up with the speed at which data is being downloaded. If you are wanting to display the imagery in near-real-time then this could become an issue (in particular light dilution corrections take a long time and are likely to take longer than the gaps between image pairs, at least if acquiring at 0.2 Hz or faster).

**5.2 Background modelling**

*Processing > Background Model*

The background intensity is used to find the optical depth of each band, and subsequently, the SO2 differential optical depth. It is therefore critical that a good estimate of the background intensity is

There are a range of options available for background intensity modelling in pycam. Options 0-6 and 99 are all pyplis methods as described in Table 2 therein. Option 7 is a very basic method which uses vignette correction and then finds the average intensity in the ambient region (defined by the rectangle on the SO2 image of the Analysis tab), for each band separately. This intensity is taken as *I0* and tau is calculated from this.

**5.3 Light Dilution**

Options are available for both spectrometer and camera light dilution corrections. Each can be toggled on and off separately. IMPORTANT NOTE: The ifit (spectrometer) light dilution correct is extremely slow – it will not be able to keep up with real-time processing. It is therefore strongly suggested that any real-time monitoring setup of pycam is run without spectrometer light dilution correction; in circumstances where the correction is required for the data it could then be reanalysed at a later date – for most instrument setups neglecting light dilution can still provide an adequate first approximation of emission rates, with later refinement most critical for high-accuracy applications.

**6. TROUBLESHOOTING**

As mentioned in the introduction, there may be times when the instrument does not work perfectly. There may be a few ways that this could be investigated/solved.

***Turning the instrument off and on again is always a good place to start! For this reason I suggest always have an on-off sequence scheduled every day, even if only for 5 minutes. This should reset everything so that if something has gone wrong the instrument can start up again and run correctly. If the instrument remains on permanently it may encounter issues that persist.***

Other possible ways to investigate/solve problems:

* Check that the pycam software is running on the instrument. If errors are encountered the pycam software may close unexpectedly. Try opening an SSH client into the Pi 1 then running the command: **ps axg**. This lists all processes currently running on the Pi. It should have both *pycam\_masterpi.py* and *pycam\_camera.py* listed – if it doesn’t, pycam is not running on the instrument. Do the same with Pi 2. This Pi should have *pycam\_camera.py* and *pycam\_spectrometer.py* running.
* If the scripts seem to be running, but not as anticipated, it may be useful to run the *pycam\_masterpi.py* script from an SSH session. This makes it possible to see the output of the script in the session, and investigate what is happening. Open a SSH session, change directory to *~/pycam/scripts* and then run one of the commands:

**python3 pycam\_masterpi.py** (without automated capture on start-up)

**python3 pycam\_masterpi.py 1** (with automated capture on start-up)

You can then play with settings, performing acquisitions etc, by connecting to the instrument as normal through the PyCam GUI. If there is a specific feature that seems to be causing the program to fail each time, test this and see what happens. If an error message is thrown in the SSH session, take a screenshot if possible and send it to me, it may help with debugging.

***Draw geometry bug***

*Instrument > Geometry configuration > Draw geometry*

If you wish to draw the geometry of the camera setup, there is currently a bug in the pyplis/geonum package which needs to be corrected. Navigate to the geosetup.py file in the geonum package in your miniconda environment. Relative to your miniconda directory it will be located:

./envs/pycam/Lib/site-packages/geonum/geosetup.py

Here you can either remove lines 521 and 522, which cause the bug:

if draw\_coastline:

m.drawcoastlines()

this will prevent coastlines from ever being drawn. Or you can edit the lines to catch an exception if there are no coastlines to be drawn:

if draw\_coastline:  
 try:  
 m.drawcoastlines()  
 except ValueError:  
 pass

This has not been tested on regions where coastline should be drawn, so it may still not draw them perfectly, but it does prevent the draw procedure from failing, so is worthwhile.