Guide to running Silicon Lab Software

1) Introduction

TL;DR:

1) sudo python flash_GPIB_USB.py
2) sudo python GUI_SiPM_lab.py
3) Click on what you need

For developer: skip to section 4 (Software stuffs)

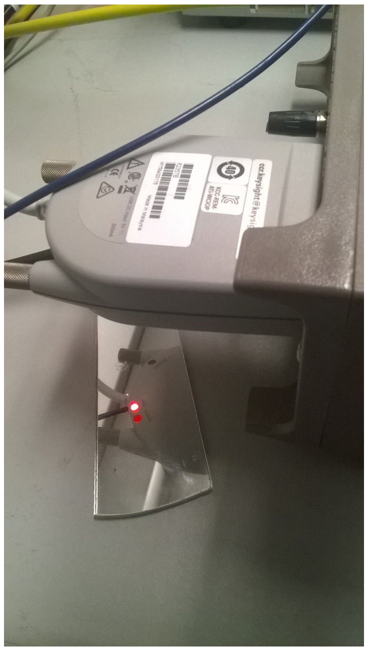
The Silicon Lab Software suite consists of 2 independent components. One that interfaces with the picoammeter to readout the current from the calibrated PIN diode. One that interfaces with the oscilloscope and power supply that power the SiPM and readout the voltage of the SiPM. Additionally, there is a small script that analyzes the SiPM peak height spectrum.

2) PIN diode script

Step 1: Flashing custom firmware to the GPIB adapter

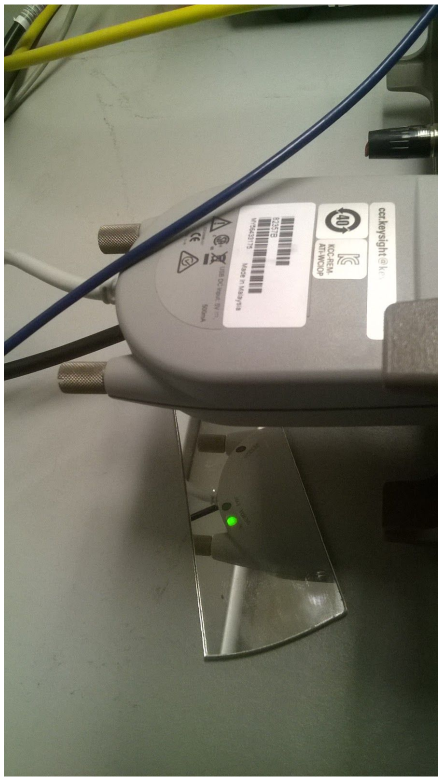
In order to run the PIN diode script, the GPIB adapter (GPIB -> USB) needs to be flashed with a customized firmware so that it can run in Linux environment. You have to run this **ONLY** **ONCE** when powering up the device.

When the USB was first plugged into the computer, all the LEDs underneath the GPIB are red.



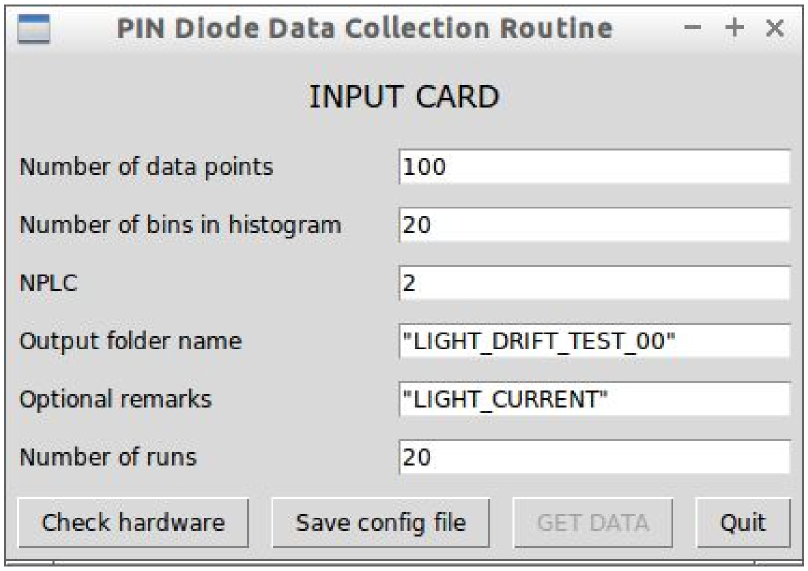
Run “sudo python flash_GPIB_USB.py”. Password for jhoang laptop account is “*w*e *o*perate *w*ith *2 17* *m*eter *t*elescopes”, same as one of the old MAGIC passwords, but with a non-capital w at the start.

All the LEDs are now green and we can communicate with the GPIB-USB adapter.



Step 2: Editing config file

Run the GUI PIN script by typing “sudo python run_PIN.py”. A windows will pop up.



PIN Diode Data Collection Routine

INPUT CARD

Number of data points	100
Number of bins in histogram	20
NPLC	2
Output folder name	"LIGHT_DRIFT_TEST_00"
Optional remarks	"LIGHT_CURRENT"
Number of runs	20

You can check communication with the Picoammeter by clicking on “Check hardware” button. If everything is fine, you can hear a click from the picoammeter.

Key in the config parameters, the <u>“Output folder name” should be unique</u>. I normally put 1000 for Number of data points, 100 for number of bins, 10 for Number of runs. Leave NPLC as 2 (technical details). When done, press click “Save config file” or press Enter. If things are okay, the GET DATA button is now clickable. Check terminal output for debug messages.

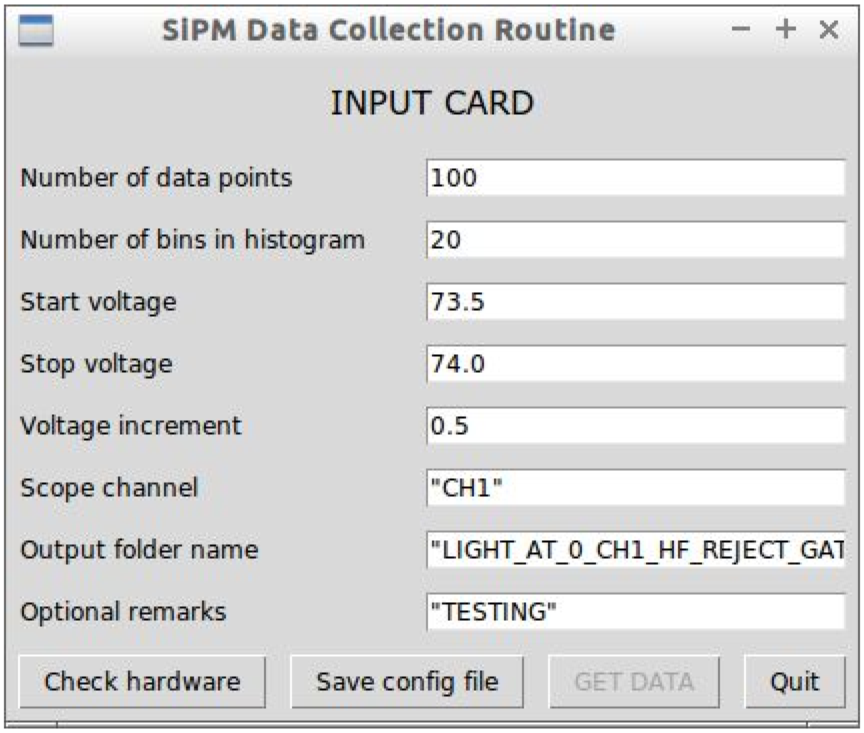
Step 3: Collecting data

Click GET DATA and go for a long coffee. During data taking, this windows will be frozen and you can’t click on anything. For panic emergency stop, go to the terminal and press Ctrl + C to abort. When the script is finished, the GET DATA button will become unclickable. The results will be saved to the folder named that you indicated in the input card. To make a new run, you have to save the input card again, with a different output folder name.

Note: since the folder was created with sudo command, if you want to delete it, you have chown it. Open a terminal and run “sudo chown jhoang *”

3) <u>SiPM script</u>

Start by opening a new terminal and run the python script: “sudo python run_SiPM.py“. Similar to the PIN diode script, it will pop up a new windows.


SiPM Data Collection Routine

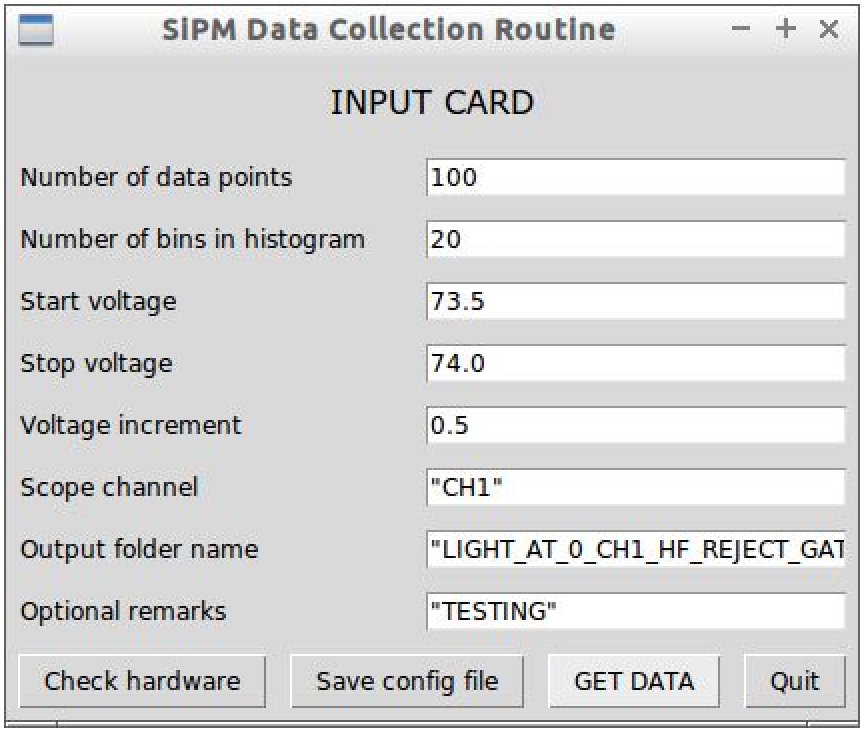
INPUT CARD

Number of data points	100
Number of bins in histogram	20
Start voltage	73.5
Stop voltage	74.0
Voltage increment	0.5
Scope channel	"CH1"
Output folder name	"LIGHT_AT_0_CH1_HF_REJECT_GAT
Optional remarks	"TESTING"

You can click on “Check hardware” to test the communication with the hardware. If the communication is ok, you should hear a beep from the scope.

Step 1: Editing config file

Key in the config parameters, pay particular attention to the “Start voltage” and “Stop voltage”, make sure that they don’t exceed the recommended value from the manufacturer, otherwise you might damage the SiPM. Also, the “Output folder name” should not exist yet. I normally put 10000 for Number of data points, 100 for number of bins, 0.2 for Voltage increment. When done, press click “Save config file” or press Enter. If things are okay, the GET DATA button is now clickable. Otherwise check terminal output for debug messages.

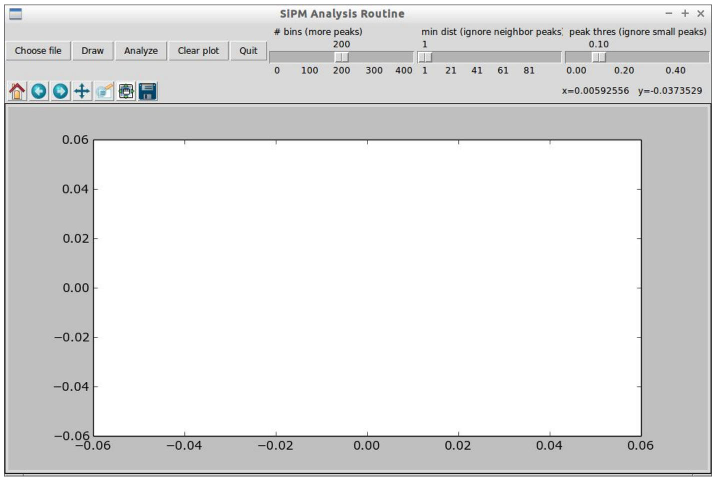


Number of data points 100
Number of bins in histogram 20
Start voltage 73.5
Stop voltage 74.0
Voltage increment 0.5
Scope channel "CH1"
Output folder name "LIGHT_AT_0_CH1_HF_REJECT_GAT
Optional remarks "TESTING"

Step 2: Collecting data
Click GET DATA and grab a cup of coffee. During data taking, this windows will be frozen and you can't click on anything. For emergency stop, go to the terminal and press Ctrl + C to abort. When the script is finished, the power supply will be reset to 0V, the GET DATA button will become unclickable. To make a new run, you have to save the input card again, with a different output folder name.

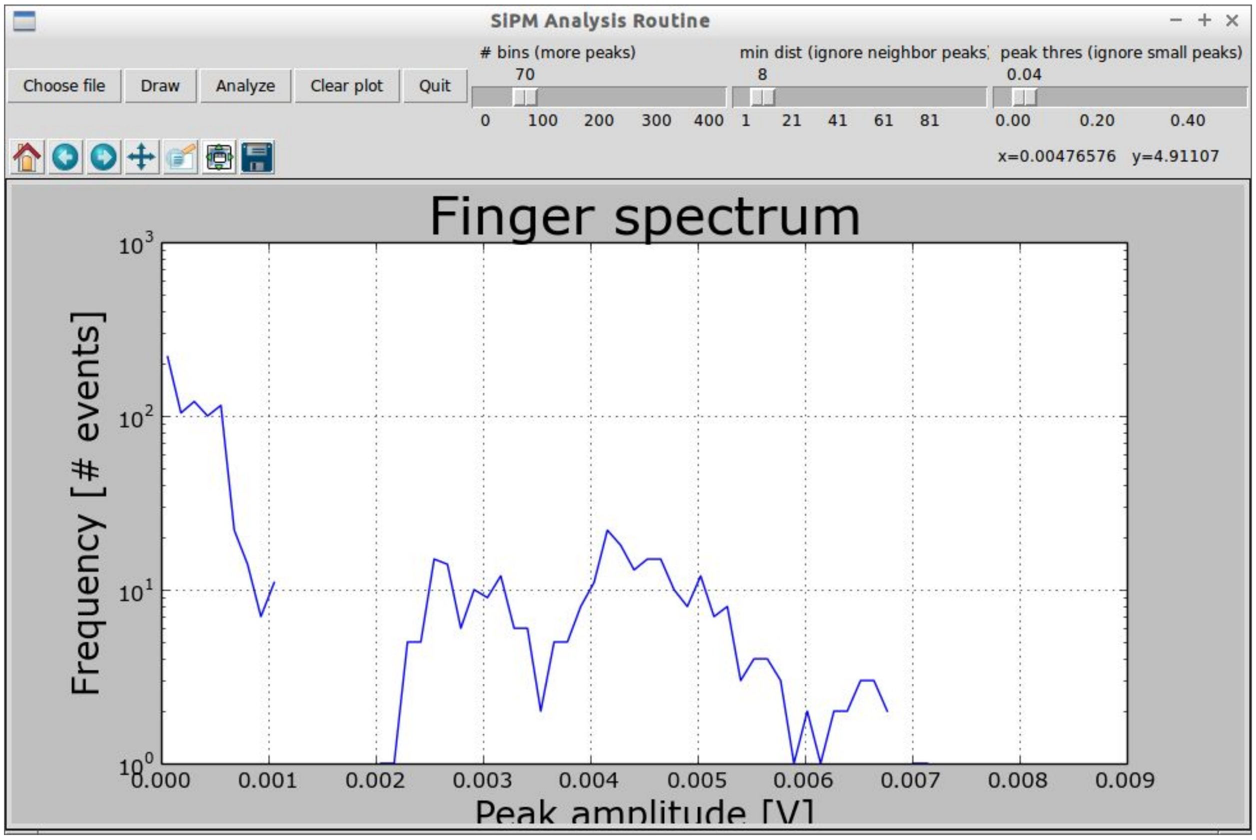
Step 3: Analyzing SiPM data

After collecting data, one you can run the script "python anayzer_SiPM.py" to start analyzing SiPM data. A window will pop up:



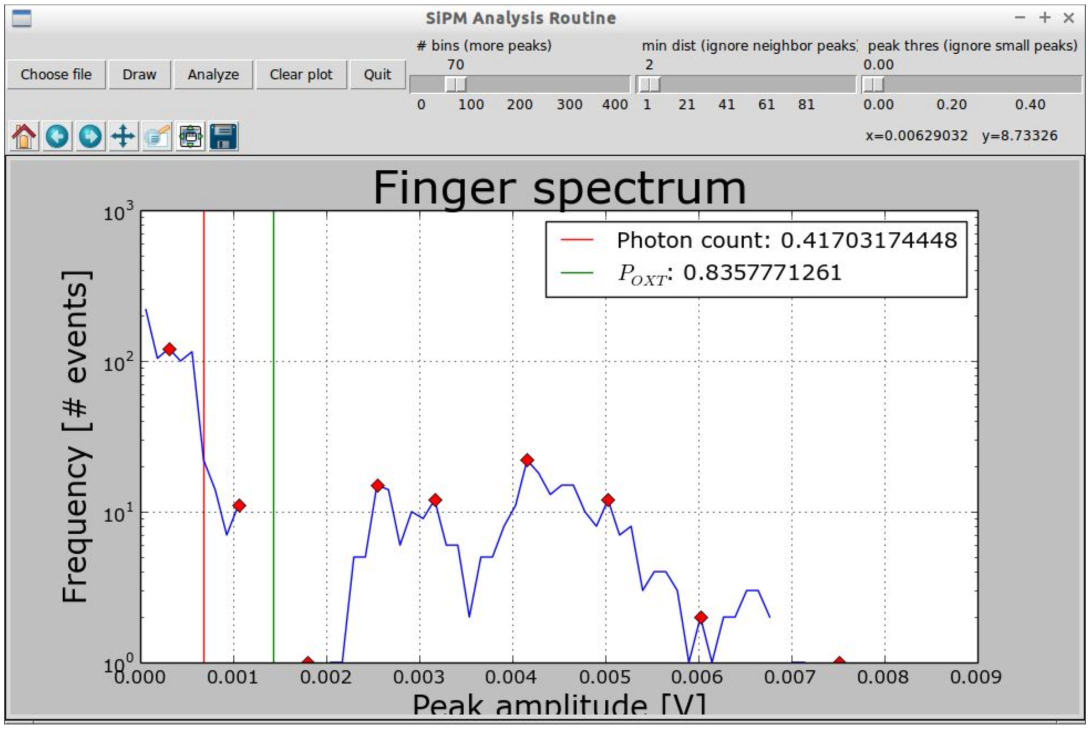
bins (more peaks) 200
min dist (ignore neighbor peaks) 1
peak thres (ignore small peaks) 0.10
x=0.00592556 y=-0.0373529

Click on “Choose file” button to load data, and then click on “Draw” button to draw the peak height spectrum:

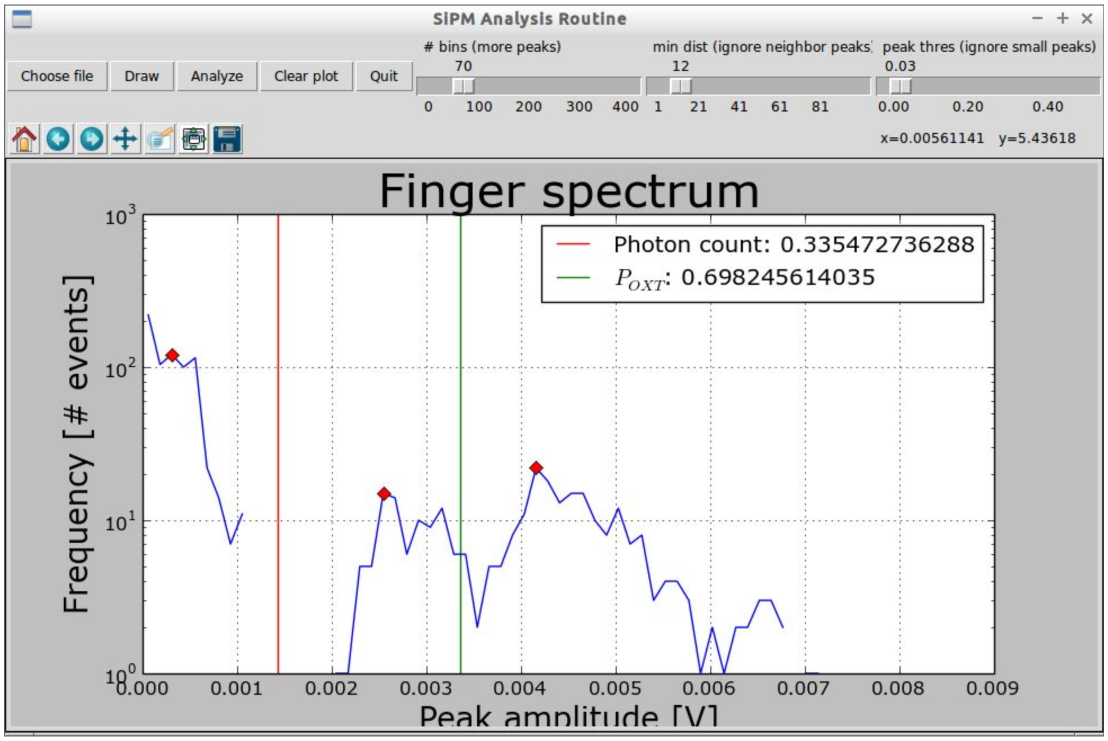


Adjust the first slider (# bins (more peaks)) so that the histogram looks reasonable enough with visible peaks. Remember to click “Draw” button to update the histogram.

The goal is to find where the pedestal, 1phe and 2phe peaks. To do so, adjust the 2 sliders on the right to small values and then click on “Analyze” to get the feel of where the peaks are.



Obviously there are too many peaks, so you need to increase the minimum distance between peaks (middle slider) and get rid of small peaks (right slider):



That's better. Whatever reading below the red vertical cut will be counted as pedestal event. Those that are between the red and green vertical lines will be the 1phe events. Those that are greater than the green vertical cut will be 2 or more phe. With these, the software calculates automatically the number of photoelectrons and optical crosstalk (if we're taking dark measurement). More detailed results can be obtained from the terminal output.

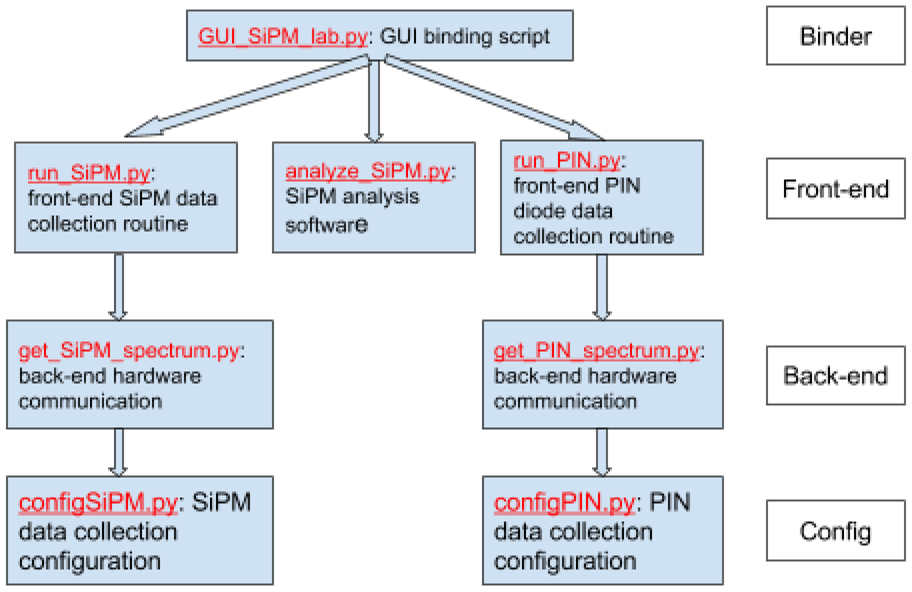
4) <u>Software stuffs</u>

There is a script called "GUI_SiPM_lab.py" sitting at the highest level, directing all the interface GUI. This enables run_SiPM.py, analyze_SiPM.py, run_PIN.py to run in parallel, each with its own newly spawned output terminal. I have only tested this option in Linux environment. Multiprocessing with Python can be tricky, so use this binding script with care.

Next, the 2 python scripts "*run_PIN.py*" and "*run_SiPM.py*" are actually 2 front-end GUI programs. They are used to modify the config files according to user inputs, and to control 2 main work horses "*get_PIN_spectrum.py*" and "*get_SiPM_spectrum.py*".

The 2 back-end scripts handle the communication with the hardware devices. They contain several device classes, each with functions that do useful things built into the class definitions. You can add more functions to the classes simply by reading the device's manual library. In principles, just these 2 backend scripts are already sufficient to collect data, importing automatically configurations from the pre-saved input card files "*configSiPM.py*" and "*configPIN.py*". However, it's not advisable to start at the backend level for speed and safety reasons.

The software architecture is as follows:


GUI_SiPM_lab.py: GUI binding script
Binder
run_SiPM.py: front-end SiPM data collection routine
analyze_SiPM.py: SiPM analysis software
run_PIN.py: front-end PIN diode data collection routine
Front-end
get_SiPM_spectrum.py: back-end hardware communication
get_PIN_spectrum.py: back-end hardware communication
Back-end
configSiPM.py: SiPM data collection configuration
configPIN.py: PIN data collection configuration
Config

Important for (future) developer:

- In the get_SiPM_spectrum.py, one has to specify the IP address of the scope and the serial communication port of the power supply. These are defined in the backend, at the initialization of the class def__init__(self, host='147.96.10.14', timeout = 10): and self.ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1). Normally you only have to check this once you make a physical change to the setup. The picocammeter don't suffer from this problem, given that you have to flash it every time it powers up.
- This software suite requires 3 external python dependencies to communicate with hardwares: visa, serial, vxi11. Please make sure that you have them when using another computer. Also, it's possible just to use one, but I don't have time to figure out how.