CALIBRATION STEPS:

Notation:
PMT case first (**PMTs, SiPMs, PDFs**)
Commands: **green**
Paths: **pink**
Files: **lilac**

1) Run taking

Data for both type of sensors are taken. PMTs can be calibrated remotely because the LEDs are in the SiPMs dices and they are controlled with the java program.

PMTs:
runs with different LEDs are taken.

SiPMs:
electronics: run with SiPMs OFF and without LEDs.
dark current: run with SiPMs ON and without LEDs.
light: runs with SiPMs ON and with LEDs. There is a LED in every PMT that has to be manually turned ON/OFF. The level of light in controlled with the pulse generator.

2) Data decoder

The machines where we work are in Canfranc: **ssh shifter@195.77.159.50 -p 6030**
VNC: open a new terminal: **shifter@frontend1next**
Data are taken in binary format, so when the run finishes, we have to transform binary files into hdf5 format with the command:
runDecoDaemon (-t) run_no -t means test
hdf5 data are stored in: **/analysis/run_no/hdf5/**. Waveforms are in **data/**.

3) Make spectra

When the decoder is finished, jobs for spectra have to be launched.
The cities need a config file with the corresponding parameters. Every run has its own config, log and sh file.

PMTs: Command: **qsub -N job_name -v city=pmtgain,config=conf_name,log=log_name generic_job.sh**
SiPMs: Command: **qsub -N job_name -v city=sipmgain,config=conf_name,log=log_name generic_job.sh**
PDFs: Command: **qsub -N job_name -v city=sipmpdf,config=conf_name,log=log_name generic_job.sh**

In the spectra, what we do is to sum the light pulses of the waveforms. The pulse in our case is always in the same position (in the example is ~40 ms). So we sum this peak overall the events for every sensor.
Example of the config parameters for the SiPMs:
SiPMs: with only one pulse per event is enough.

```
# Histogram bins (ADCs)
```

```
    min_bin = -49.5
    max_bin = 300.5
    bin_width = 1.
    # Integral definitions. Start, width and period in
microseconds
    number_integrals =  1
    integral_start   = 40
    integral_width   =  3
    integrals_period = 50
```

In the case of the pdfs, we sum the whole waveform.

** The steps **2)** and **3**) can be done “simultaneously” with the command:

PMTs: **runSensorCal pmt run_no1 run_no2 …**
SiPMs: **runSensorCal sipm run_no1 run_no2 …**
PDFs: **runSensorCal pdf run_no1 run_no2 …**

In this case, the spectra are saved in:

PMTs: **/calibration/pmtCal**
SiPMs: **/calibration/sipmCal**
PDFs: **/calibration/pdfCal**

4) Fit spectra

From the fits of the spectra we obtain the important parameters of the sensors: the gain, the mu value, the sigma, the noise…
The files that make the fits are:

PMTs: **pmtCalFit_test.py**
python pmtCalFit_test.py spectra.h5 dfunc true 50
dfunc is the function we use for fitting
true means that we take the seeds from the gain values of the DB. If false, peaks are obtained from the curve.
50 is the number of bins

SiPMs: **sipmCalFit_test.py**
python sipmCalFit_test.py spectra.h5 dfunc true 10

A hdf5 file is obtained. Example: sipmCalParOut_R7699_Fdfunc.h5

IMPORTANT: Sometimes we need to change the gains of the sensors, so to make the fits, we cannot use the database gains for the seeds of the fits, so we have to take the peaks from the spectrum by using: **use_db_gain_seeds = False**!!!!!
Sometimes we will have to change also the parameters in the function **sensor_values** of IC!!!

SiPM voltage <= 27V:

```
if sensor_type is SensorType.SIPM:
        spectra         = spectrum
        peak_range      = np.arange(4, 20)
        min_bin_peak    = 8
        max_bin_peak    = 20
        half_peak_width = 4
        p1pe_seed       = 2
        lim_ped         = 10000
```

SiPM voltage > 27V:

```
if sensor_type is SensorType.SIPM:
        spectra         = spectrum
        peak_range      = np.arange(4, 20)
        min_bin_peak    = 10
        max_bin_peak    = 22
        half_peak_width = 5
        p1pe_seed       = 3
        lim_ped         = 10000
```

Also in the script, if the option use_db_gain_values is False, and the seeds cannot be found, we add them manually, so be careful with the approximate gain!!!

5) Extract fit values

A TXT file is created with the important parameters extracted from the fits for every sensor.

PMTs: **pmtGainTXT.py**
python pmtGainTXT.py pmtCalParOut_R7699_Fdfunc.h5
SiPMs: **sipmGainTXT.py**
python sipmGainTXT.py sipmCalParOut_R7699_Fdfunc.h5

6) Compare values from different runs

The script plots the comparison between the runs with different LEDs taking the txt file created in step **5).**

PMTs: **pmt_comparison.py** It plots the absolute gain value for each run
python pmt_comparison.py file0 file1 file2 …
SiPMs: **sipm_comparison.py** It plots a histogram with the difference of the values of each run with respect to the first one (file0).
python pmt_comparison.py file0 file1 file2 …

7) Select values to upload database

The script calculates the mean value overall the runs with different LEDs taking the txt file created in step **5)**.

PMTs: **pmtDatabaseTXT.py**
python pmtDatabaseTXT.py min_run file0 file1 file2 …
SiPMs: **sipmDatabaseTXT.py**
python sipmDatabaseTXT.py min_run file0 file1 file2 …

A txt file is created.

8) Upload database

Enter with the user nextwriter (same password as always)

a) ChannelGain —> Import —> select file created in step 7) —> format CSV —> Continue

b) We have to change the MaxRun for the previous dataset.

PMTs: Edit —> UPDATE `ChannelGain` SET MaxRun=XXXX WHERE MinRun=YYYY AND SensorID<100

SiPMs: Edit —> UPDATE `ChannelGain` SET MaxRun=XXXX WHERE MinRun=YYYY AND SensorID>100

9) Download the database and upload IC

a) Create a new branch in IC repository

b) Clonate the database:

source manage.sh work_in_python_version 3.7

source manage.sh download_test_db (NEWDB/DEMOPPDB if we don't add anything, it uploads all databases but takes long time)

c) Check that all tests pass: **source manage.sh work_in_python_version 3.7**

c) Add a commit with the file localdb.sqlite3

d) Make a PR explaining all the changes in the DB. Add some plots.

10) Upload DB in Canfranc

When the PR is accepted and merged, the database in Canfranc needs to be uploaded too.

Sign with the corresponding user in frontend1next. With the shifter account will not work (my alias is canfranc_cr).

a) Activate the environment variables: **source ic_setup.sh**

b) **cd /software/IC**

c) Download the new db: **bash manage.sh download_test_db NEWDB**

With the previous command DB is only downloaded in frontend1next, so we have to synchronize the 3 machines located in Canfranc:

d) **sudo /scripts/rsyncSoftware.sh**

e) Check that the values uploaded in Canfranc are correct. (example: python; from invisible_cities.database import load_db as DB; DB.DataSiPM('new', run_no))

11) Upload monitoring plots

/home/shifter/monitoring/calibration

12) Send an email to the Collaboration

next-physics@pegaso.ific.uv.es

Check SiPMs are alive:

Needed files: **sipmConnectionTest.py**
calutils.py

There are two possibilities:

1) Using the **waveforms** of the runs:

python sipmConnectionTest.py -w wf_file0 wf_file1 wf_file2 … run_no

In the run_no better to add the lowest run number because it is for the name of the txt file that is created

2) Using the electronics and dark current **spectra**. (The dark current spectrum

has been created with the standard way, not pdf!)

python sipmConnectionTest.py -e elec_spec -d dark_spec run_no

UTILS:

SensorID: assigned number to have the same values in MC and data.

PMTs: 0, 1, 2, …, 11

SiPMs: 1000, 10001, 10002, …, 28063

28063: 28 is the dice number and 63 is the sensor number

ChannelID: number of the sensors in order.

28063 would be the 1791

ElecID: number that comes from the electronics.