SLab Linux Installation

This document describes how to install the SLab system in a Linux PC. The installation has been tested on Debian but it should be similar in any other Linux version.

V1.0(23/5/2019) License information is at the end of the document

In order to have use the **SLab system**, you need a **Python 3.x** environment that includes the required modules:

- Numpy
- Matplotlib
- PySerial
- Jupyter

If you have access to a Windows system, although you can successfully use SLab it Linux, it is easier to work with SLab in Windows. This is due to two reasons. In the first place, SLab code and notebooks are developed in Windows, so, most potential problems that relate to Windows have been corrected as part of the development process.
In the second place, the SLab Hardware Board communicates with the PC using a serial link. Serial connections are always available on the user space on Windows, but they are much more restricted in Linux, so making the serial connections work in the user space is more difficult in Linux than in Windows.

Installing Anaconda

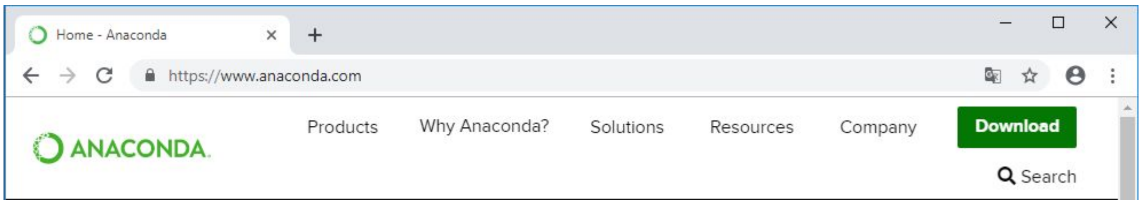
Although you can manually install Python and add the needed packages afterwards, we will only cover in this tutorial the simpler approach of using the **Anaconda** Python platform. You can get information about it on:

https://www.anaconda.com/

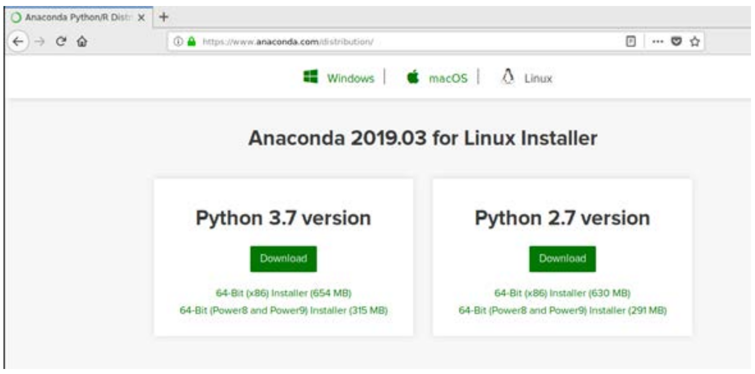
Installation can be performed system wide (requires root access) or on a user folder. We will follow this second case. In all examples in this document, username is ***tecnic*** and its user folder is `/home/tecnic`, also available as `~/tecnic`

Note, however, that in order to access to the serial port, needed for SLab to operate, you will need at certain point root access to execute some commands.

First, we will go to the Anaconda web page:



Then go to the download tab:



Anaconda 2019.03 for Linux Installer

Python 3.7 version

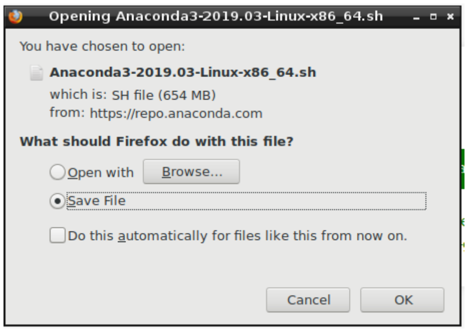
64-Bit (x86) Installer (654 MB)
64-Bit (Power8 and Power9) Installer (315 MB)

Python 2.7 version

64-Bit (x86) Installer (630 MB)
64-Bit (Power8 and Power9) Installer (291 MB)

As this manual only covers Linux installations, select 64bit or 32bit version of the Python 3.x installer that match your CPU.

Anaconda Linux installers are usually associated to shell files (.sh).



Opening Anaconda3-2019.03-Linux-x86_64.sh

You have chosen to open:

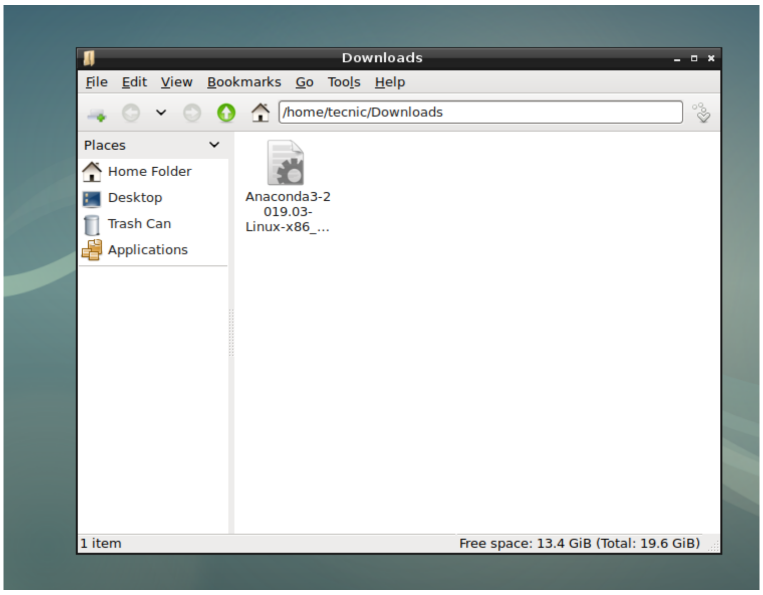
Anaconda3-2019.03-Linux-x86_64.sh
which is: SH file (654 MB)
from: https://repo.anaconda.com

What should Firefox do with this file?

Open with Browse...
[✓ Save File]

Do this automatically for files like this from now on.

Download the selected shell file to any place inside your user folder. Typically, it will be downloaded to `~/Downloads`.



/home/tecnic/Downloads

Anaconda3-2019.03-Linux-x86_...

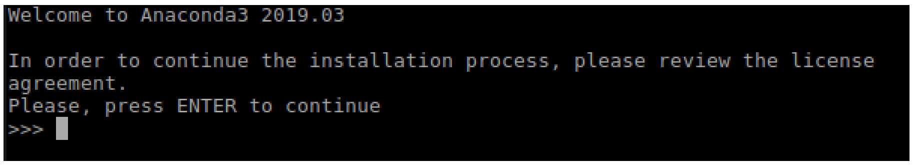
1 item   Free space: 13.4 GiB (Total: 19.6 GiB)

After downloading the installer, run it executing the shell file:

```
$ cd Downloads
$ cd bash Anaconda_Installer.sh
```

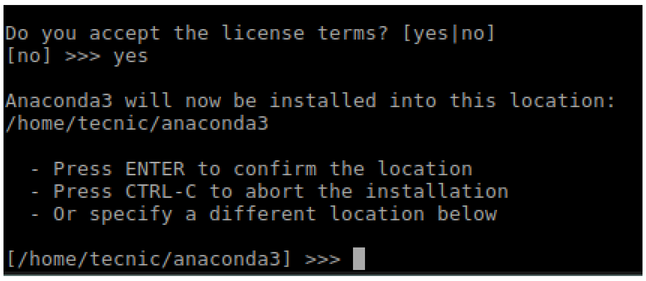
Note that the above ones are not the exact commands you should execute. The download folder can be different from the proposed "*Downloads*" and the shell file will be sure different from the proposed name.

The installer will ask you to agree with the license agreement.



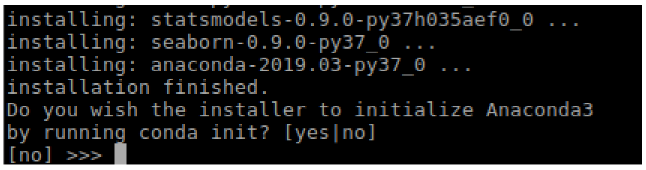
```
Welcome to Anaconda3 2019.03

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>> 
```

After accepting the license, you should confirm the install location, using `~/anaconda3` is fine for most purposes.



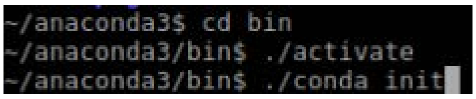
```
Do you accept the license terms? [yes|no]
[no] >>> yes

Anaconda3 will now be installed into this location:
/home/tecnic/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/tecnic/anaconda3] >>> 
```

When the installer ends, it recommends you to initialize Anaconda3. This initialization sets the paths for Anaconda so that you can run Python3 applications from any folder.



```
installing: statsmodels-0.9.0-py37h035aef0_0 ...
installing: seaborn-0.9.0-py37_0 ...
installing: anaconda-2019.03-py37_0 ...
installation finished.
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[no] >>> 
```

It is recommended to respond "yes", but if you don't do that, you can perform this process later by going into the `anaconda3/bin` folder and executing the `activate` and the `conda init` commands.



```
~/anaconda3$ cd bin
~/anaconda3/bin$ ./activate
~/anaconda3/bin$ ./conda init
```

After completing the Anaconda installation, it is a good idea to check that you have all the needed modules. In order to do that, execute `python3`. If you have done the activation, it will work from any folder. If you don't, you will need to execute it from the `anaconda/bin` folder. The same applies to any other anaconda related command so we won't explain that again in this document.

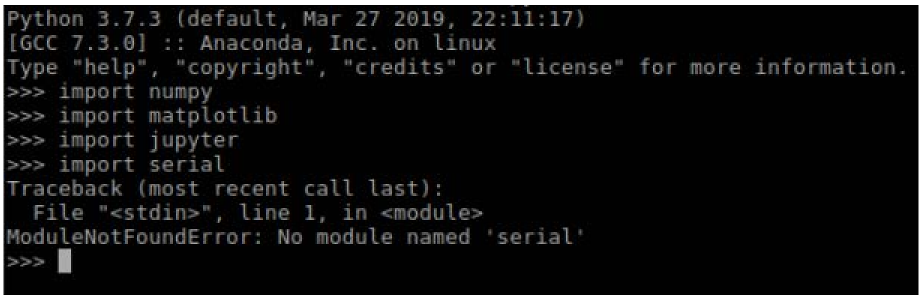
The modules to test are:

- Numpy
- Matplotlib
- Jupyter
- serial

To test the modules we will try to import all of them, the Python syntax is:

```
import module
```

If the module exits, Python won't say anything. It will only complain if it fails to load.
A typical execution will be:

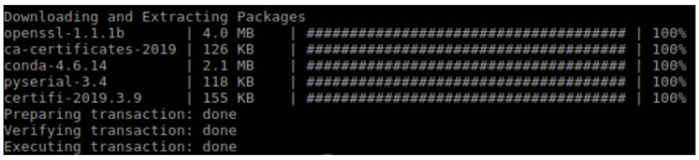


```
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> import matplotlib
>>> import jupyter
>>> import serial
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'serial'
>>> 
```

Adding PySerial

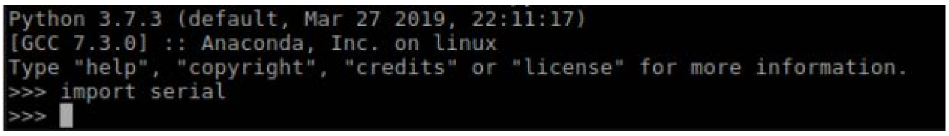
It is quite normal for ***serial*** to fail as it is not usually installed automatically in Anaconda. To install this module, you will need to execute the code:

```
conda install -c anaconda pyserial
```



```
Downloading and Extracting Packages
openssl-1.1.1b       | 4.0 MB    | ##################################### | 100%
ca-certificates-2019 | 126 KB    | ##################################### | 100%
conda-4.6.14         | 2.1 MB    | ##################################### | 100%
pyserial-3.4         | 118 KB    | ##################################### | 100%
certifi-2019.3.9     | 155 KB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

After installing **pyserial** we can verify that we can import this module:

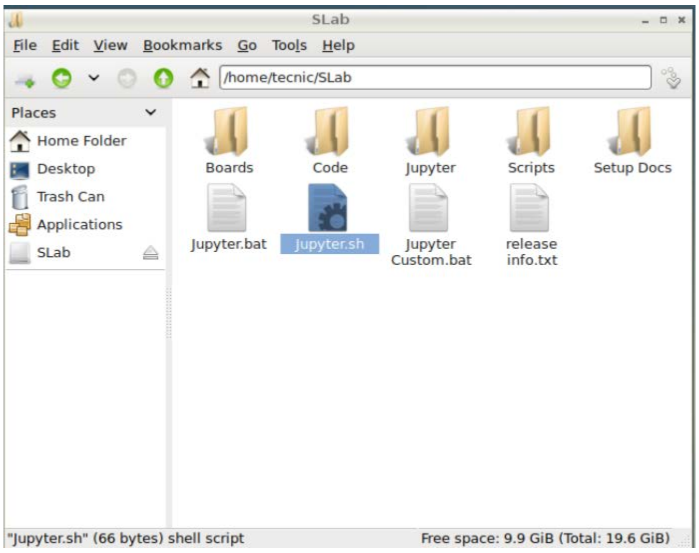


```
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import serial
>>> 
```

Once all needed modules are working, we have completed the Anaconda installation. Now it is type to deal with the SLab specifics.

Adding the SLab files

Locate the SLab installation zip file and uncompress its main **SLab** folder on the root of your user folder. The SLab folder will be then be accessible as `~/SLab`



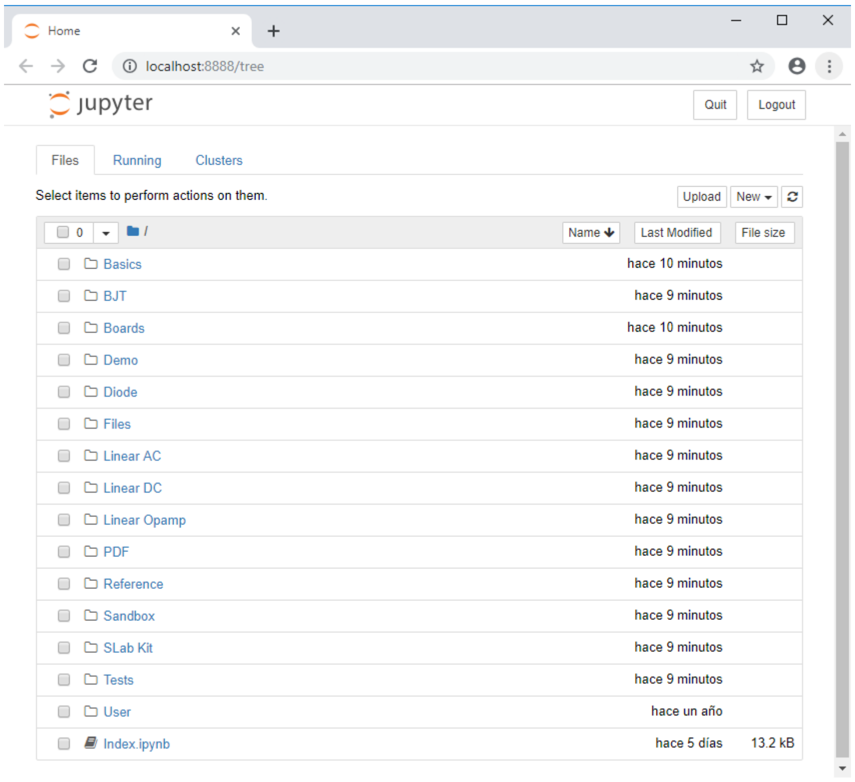
To start the **SLab system**, just enter the ***SLab*** folder and execute the ***Jupyter.sh*** shell file.

```
cd ~/SLab
./Jupyter.sh
```

Note that Jupyter.sh needs to be executable to do that. If this is not the case, make it executable by executing the command:

```
chmod +x Jupyter.sh
```

Jupyter is a programming environment that uses web pages for the user interface. After a successful start, your default navigator should open and show a navigator for the files on the `~/SLab/Jupyter` folder.

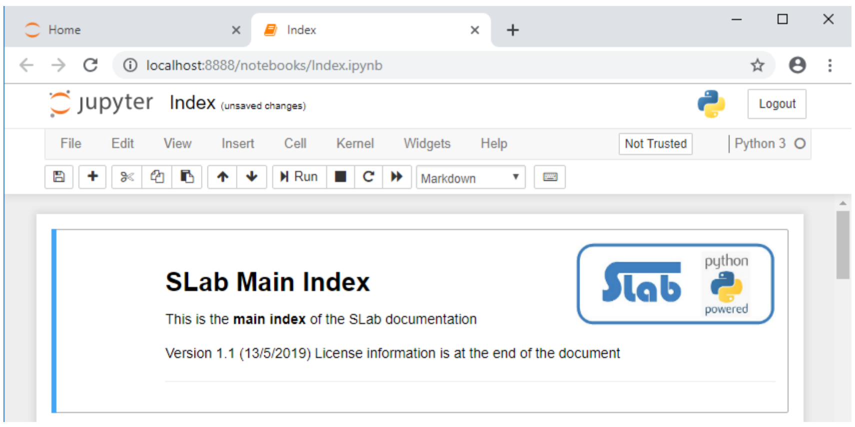


localhost:8888/tree

Select items to perform actions on them.

Name	Last Modified	File size
Basics	hace 10 minutos	
BJT	hace 9 minutos	
Boards	hace 10 minutos	
Demo	hace 9 minutos	
Diode	hace 9 minutos	
Files	hace 9 minutos	
Linear AC	hace 9 minutos	
Linear DC	hace 9 minutos	
Linear Opamp	hace 9 minutos	
PDF	hace 9 minutos	
Reference	hace 9 minutos	
Sandbox	hace 9 minutos	
SLab Kit	hace 9 minutos	
Tests	hace 9 minutos	
User	hace un año	
Index.ipynb	hace 5 días	13.2 kB

You can now click on the **Index.pynb** file to open the main SLab index file.



localhost:8888/notebooks/Index.ipynb

SLab Main Index

This is the **main index** of the SLab documentation

Version 1.1 (13/5/2019) License information is at the end of the document

From this file you can access to all SLab contents. We recommend that you go through the documentation till the point where you need to connect with the **Hardware Board**. At this point continue reading this document to have a successful connection with the board.

Making serial work

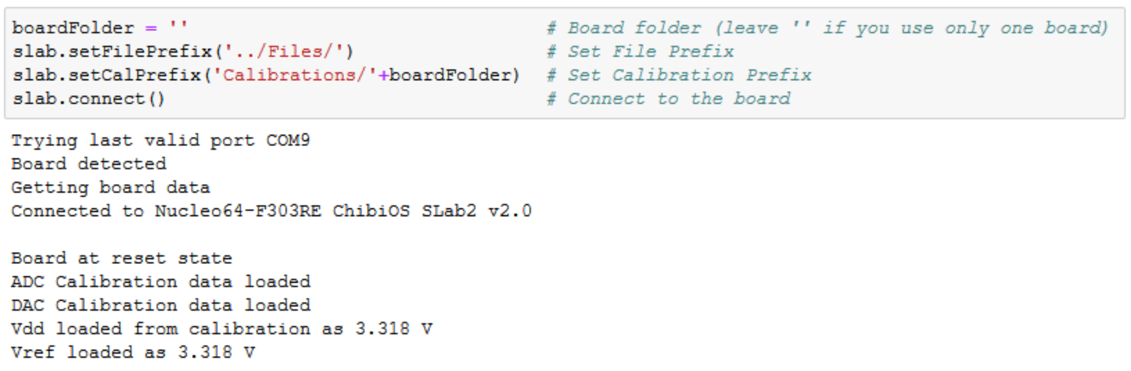
Making the serial communication work in Linux is much complex than in a Windows system. Linux prevents, by default, user level access to devices and that includes the serial ports.

When you connect the SLab hardware board it shows up in the `/dev` folder as a serial terminal **`tty`** device. Typically the `/dev/ttyACM0` device for serial over USB connections. You can check that if you go to the `/dev` folder and checking the **`tty`** devices before and after connecting the **Hardware Board** to the computer.

To test the communication, you need to connect the Hardware Board to the computer and check that it can be seen on the `/dev` folder. Don't continue if this is not the case.
The board should also contain a proper **SLab firmware**, check the documentation from the main SLab index notebook about this subject.

After we have a board **connected** that hosts the proper **firmware**, to test the communication you can use any of the SLab Jupyter notebooks that contain a `connect()` command. For instance, the ***SLab Test*** notebook that is inside the `Tests` folder and is also accessible at the top of the main SLab index notebook.

A successful connection will show something like:

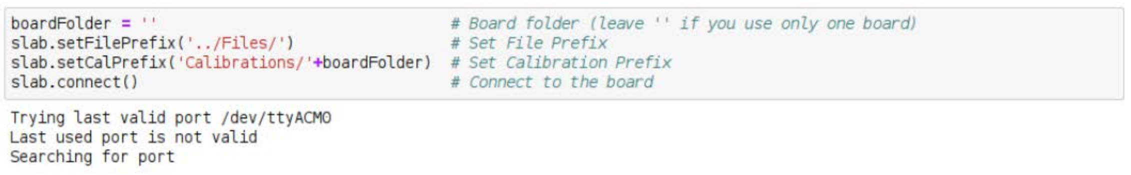


```
boardFolder = ''                                  # Board folder (leave '' if you use only one board)
slab.setFilePrefix('../Files/')                   # Set File Prefix
slab.setCalPrefix('Calibrations/'+boardFolder)    # Set Calibration Prefix
slab.connect()                                    # Connect to the board
```

```
Trying last valid port COM9
Board detected
Getting board data
Connected to Nucleo64-F303RE ChibiOS SLab2 v2.0

Board at reset state
ADC Calibration data loaded
DAC Calibration data loaded
Vdd loaded from calibration as 3.318 V
Vref loaded as 3.318 V
```

The calibration data could be different depending if the SLab files you are using contain a calibration for the board.

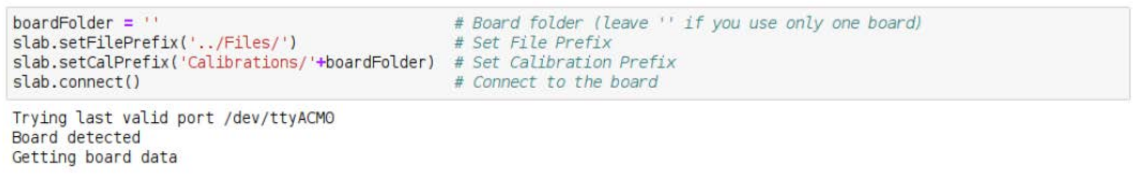
A connection can lock during the search for the serial port:



```
boardFolder = ''                                  # Board folder (leave '' if you use only one board)
slab.setFilePrefix('../Files/')                   # Set File Prefix
slab.setCalPrefix('Calibrations/'+boardFolder)    # Set Calibration Prefix
slab.connect()                                    # Connect to the board
```

```
Trying last valid port /dev/ttyACM0
Last used port is not valid
Searching for port
```

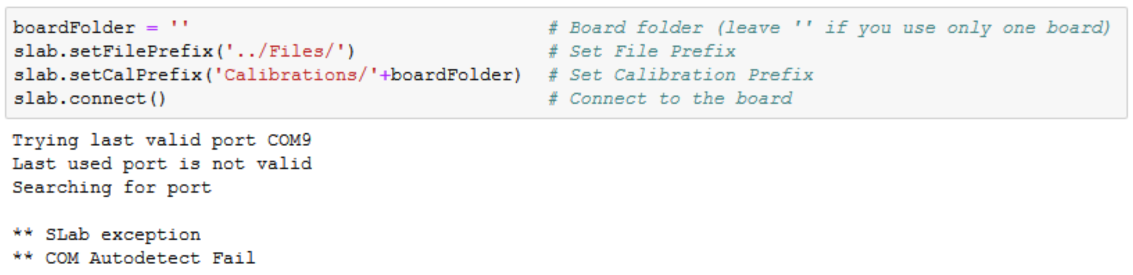
In can also fail during after the communication has started:



```
boardFolder = ''                                 # Board folder (leave '' if you use only one board)
slab.setFilePrefix('../Files/')                  # Set File Prefix
slab.setCalPrefix('Calibrations/'+boardFolder)   # Set Calibration Prefix
slab.connect()                                   # Connect to the board
```

```
Trying last valid port /dev/ttyACM0
Board detected
Getting board data
```

Also, it can fail because it cannot detect the board.



```
boardFolder = ''                                 # Board folder (leave '' if you use only one board)
slab.setFilePrefix('../Files/')                  # Set File Prefix
slab.setCalPrefix('Calibrations/'+boardFolder)   # Set Calibration Prefix
slab.connect()                                   # Connect to the board
```

```
Trying last valid port COM9
Last used port is not valid
Searching for port

** SLab exception
** COM Autodetect Fail
```

An alternative way to check the most basic board communication is not to use Python at all. Every time you reset the board, it will generate a message on the serial port. You can check it using any terminal program like ***miniterm***. This is a good check because it separates the needed rights to access to the serial port of any other error related to the python installation.

If you don't have this program you can install it with:

```
sudo apt-get install minicom
```

Now, if the **Hardware Board** shows its serial port as `/dev/ttyACM0`, you can open a terminal on this port on the standard 38400 baudrate with:

```
minicom -D /dev/ttyACM0 -b 38400
```

The above command will most probably fail if you are not root. It is much easier to work on SLab if you are root as you are granted access to all the computer hardware, but, as this is not safe, we will need to find how we can make serial work on the normal user space. Note, however that, as in the case of installing minicom, you need to be **root** or have access to **sudo** to perform some of the operations.

First thing to do is to get rid of the modem manager if it is installed in the system:

```
sudo apt-get autoremove modemmanager
```

Second thing to do is to add the user that wants to use SLab to the dialup group:

```
sudo adduser username dialout
```

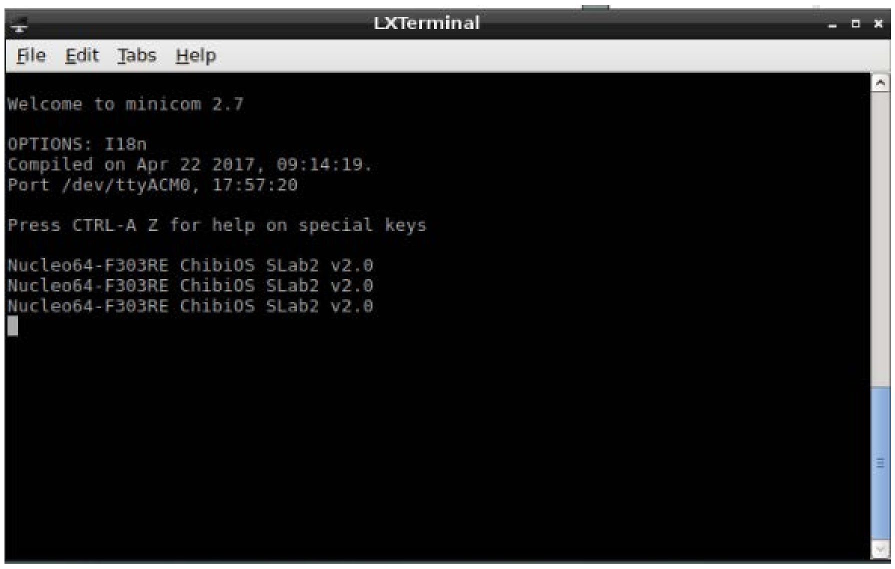
Off course you need to substitute ***username*** with the username that will use SLab.

Sometimes the above commands are enough to be able to access to the terminal on the serial port, as this was the case in one **Ubuntu** installation. But in some installations, as was the case in the **Debian** system used to document this manual, we also need to make the `/dev/ttyACM0` accessible in a more basic way:

```
sudo chmod -R 777 /dev/ttyACM0
```

The problem if you need to perform this third step is that both getting rid of the modem manager and adding the user to the dialup group are permanent in the system and they only need to be performed once. However, the `/dev/ttyACM0` folder is generated when you connect the board to the computer, so the **chmod** command, if needed, needs to be performed, with root access, every time you connect the board to the computer.

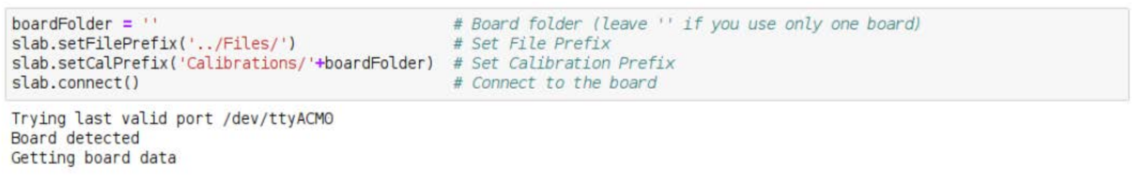
If all goes ok, minicom, running with normal user rights, will get access to the Hardware Board serial port. Each time you reset the board, it will respond with a message with the version of the installed firmware. In the example below, the board has been reset three times.



```
Welcome to minicom 2.7

OPTIONS: I18n
Compiled on Apr 22 2017, 09:14:19.
Port /dev/ttyACM0, 17:57:20

Press CTRL-A Z for help on special keys

Nucleo64-F303RE ChibiOS SLab2 v2.0
Nucleo64-F303RE ChibiOS SLab2 v2.0
Nucleo64-F303RE ChibiOS SLab2 v2.0
```

To exit minicom just use CTRL-A followed by Z, the X and confirm with return.

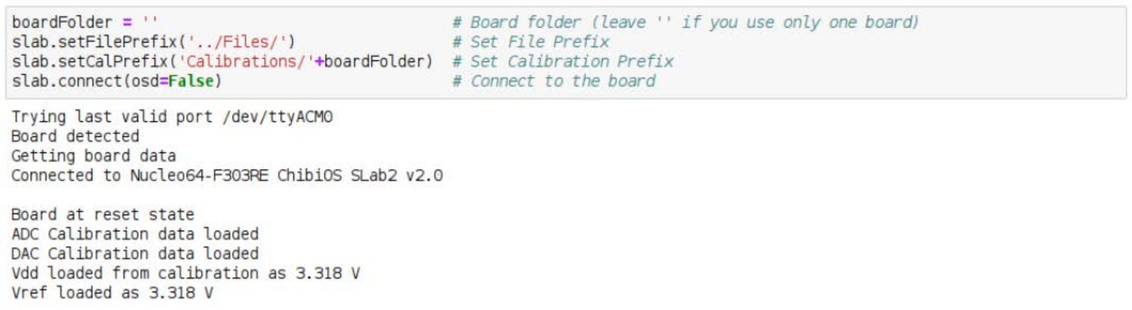
The above procedures will hopefully make the SLab notebooks able to connect with the board. Sometimes, however, you can see the firmware string on **minicom** but SLab will lock when connecting to the board. Typically, during the gathering of board data:



```
boardFolder = ''                              # Board folder (leave '' if you use only one board)
slab.setFilePrefix('../Files/')               # Set File Prefix
slab.setCalPrefix('Calibrations/'+boardFolder)  # Set Calibration Prefix
slab.connect()                                # Connect to the board
```

```
Trying last valid port /dev/ttyACM0
Board detected
Getting board data
```

This is quite specific to some Linux versions like **Debian** as was the case with the need to use the **chmod** command. For those cases, one solution that has been proven to work is to disable the optional operating system detection in the connection using the parameter **osd** with a **False** value.



```
boardFolder = ''                              # Board folder (leave '' if you use only one board)
slab.setFilePrefix('../Files/')               # Set File Prefix
slab.setCalPrefix('Calibrations/'+boardFolder)  # Set Calibration Prefix
slab.connect(osd=False)                       # Connect to the board
```

```
Trying last valid port /dev/ttyACM0
Board detected
Getting board data
Connected to Nucleo64-F303RE ChibiOS SLab2 v2.0

Board at reset state
ADC Calibration data loaded
DAC Calibration data loaded
Vdd loaded from calibration as 3.318 V
Vref loaded as 3.318 V
```

If you need to use the ***osd=False*** setting to be able to connect to the board in your system, you will need to add this setting to the connect command on all notebooks you work with. Fortunately, changes made to the notebooks became permanent if you save them after the modification.

That all for this document, if you have been able to connect with the board using a notebook you have a fully working Linux SLab system.