Red Pitaya

On this page are all the information you need to set up your Red Pitaya and to work with all used Red Pitayas.

Quick-Start Tutorial

The homepage of Red Pitaya provides an overview of all the different STEMlab boxes and build-in hardware modules. As soon as you bought your Red Pitaya you need to flash an OS-Image to your SD-Card. The processor of e.g. STEMlab 125-14 (which is used in the Atom-Lab) is an armv7l 32-bit processor which is supposed to run with Ubuntu. You can find all new and stable OS-Images here.

As soon as your SD-Card is ready you can insert it into your STEMlab. Before connecting the power supply, you need to connect the board over an ethernet cable with one of the MG-Routers or directly with your computer. If you choose the direct connection be aware that as long as the IP-address of your Red Pitaya wasn't registered by a network administrator you don't have access to the internet.

Check the LED's to eventually detect some first problems:

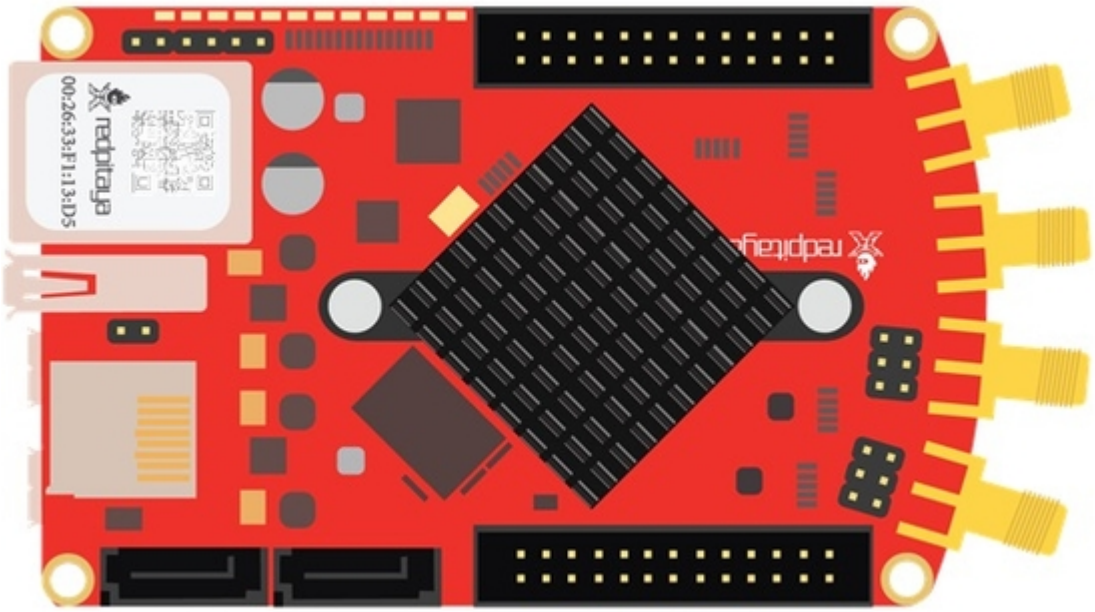
- <fc #00ffff>Blue LED:</fc> normally this LED is turned ON indicating fpga bitstream was successfully loaded
- <fc #00ff00>Green LED:</fc> normally this LED is turned ON indicating that all power supplies on Red Pitaya are working properly
- <fc #ff0000>Red LED:</fc> heartbeat blinking pattern should show CPU load (in normal operation this LED is blinking after about 10 seconds or earlier)
- <fc #ffa500>Orange LED:</fc> SD card access indicator (in normal operation this LED is blinking in slow intervals)

For the basic communication with the file system an SSH-connection works fine. You can use PuTTY to create it. To find out which IP-address your Red Pitaya has by checking all IP-addresses in your private network. Therefor you can enter the command `arp -a` in cmdlet. IP-addresses in private networks usually start with 192:168: and the physical address should match the address which is written on the board. Be aware that you need to be in the same network as your STEMlab. As soon as your connection works, you are asked to login. By default, the password is root. Check the disk space with `df -h`; if the disk seems to be smaller than it should be run the resize.sh script by entering the command `/opt/redpitaya/sbin/resize.sh`. Check the space again and if there is still space missing, you can try to manually run the command by entering `sudo resize2fs /dev/mmcblk0p2`.

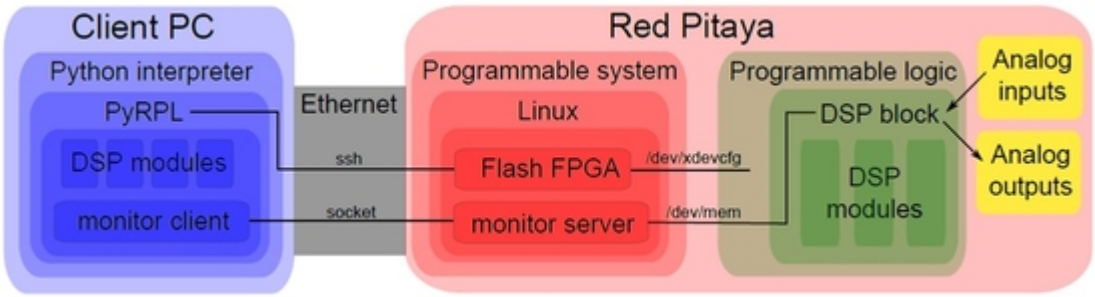
To work with the build-in hardware and software modules go to the website “http://rp-f07db1.local” (this is the physical address of the PID-Controller board - of course you need to insert the physical address of the board you want to connect to. Be aware that you need to be in the same network as your STEMlab. There you can find several applications, development environments, cailbration tools and updates. Here's the documentation for all those modules.

And that's it! Your Red Pitaya is ready! 😬

Basic information about Red Pitaya



The FPGA-Board has 4 fast analog inputs and outputs. You can define them as whatever you want, but only the SMA connectors which are marked as 'In1' and 'In2' possess jumpers which allow you to choose between a Low Voltage Range (LV, +-1V) and a High Voltage Range (HV, +-20V). The board provides several more connectors depending on which board you have.



This is an overview of the software architecture including the Python environment you need if you want to run your own scripts. To do so you can use the Python-package PyRPL on your PC which takes care of the communication with the board on it's own.

PyRPL

As described before, this is a Python-package which was designed to enlarge the possibilities of applications of Red Pitaya. After you finished the installation on your PC you can either use there graphical user interface, e.g. by running “pyrpl-windows” in windows or you can import their functions in your preferred python develop environment, which is very usefull for personalized applications, data recording, data verification or if you dont want to run their program all the time. Once executed, the script runs on your board until you execute a script that stops the process.

The package as well as Red Pitaya itself is still in development and therefore some applications are not fully available yet - so don't forget to regularly check for updates!

Step by step: basic communication with red pitaya (SSH)

A different way to work with the red pitaya is accessing it via ssh and typing in commands directly. To do this some steps are needed:

1. prepare the SD card as stated in the documentation.

2. connect to the Red Pitaya (documentation). This might be a problem depending on what you connect it to. If you connect it to the Lab internal network you need a pc connected to that network (if you work in the office but have the RP in the lab you need to connect to the network by e.g. WireGuard. You need the right file to create the connection to the Lab net). If you want to connect the RP in the office and also work in the office the red pitaya first needs to be added to the network by an admin (to do this: log in to LAM, go to “hosts” in the menu bar at the top of the window, click “hinzufügen”/“add”, then add the same way as adding a user; if needed read admin mails)

3. now you can connect to red pitaya. In any browser enter HOSTNAME/. The hostname is by default the upper address on the Sticker (e.g. “rp-f07db1.local”). Try connecting to your rp before taking any of the following steps to make sure the connection is there and you know the hostname.

4. SSH to rp:

- install putty e.g. from heise
- open putty, in the window marked “Hostname or IP adress” type in the Hostname of the rp.
- log in as root, pw root (default)
- change the log in info via: `$passwd` and `$hostnamectl set-hostname HOSTNAME` with HOSTNAME being the name of your choice, preferably “MG-LAB-XX” where XX ist the number of the red pitaya. (it is important you know what hostname you gave the rp because this is how you connect to it via ssh)
- set time/date via `$timedatectl set-timezone Europe/Berlin`.
- before doing anything else on the rp it's a good idea to add a new user. Do this with `$adduser` USERNAME where USERNAME is the chosen name of the user. You will then have to set a password as well.
- you also have to give the user root rights, in order to do that use the command `$usermod -aG sudo` USERNAME, whith USERNAME being the name of the user you want to give rights to.

All commands in the style of `$this is a command` should be typed in the console you get when connecting to your rp with putty. python version 2.7 is preinstalled. Here's how to install the wanted python 3 version:

First run `$sudo apt update && sudo apt upgrade` and`$sudo apt-get install wget build-essential check`

then install some packages: `$sudo apt-get install libncurses5 tk libc6 zlib1g libffi-dev libsodium-dev` `$ sudo apt-get install python3` on the rp via ssh (putty). If you want a specific python3 version, you have to specify, e.g.: `$sudo apt-get install python3.7` To make this version of python your default version, do the following: `$sudo update-`

`alternatives - -install /usr/bin/python python usr/local/bin/python3.7 10`. “`/usr/bin/python`” is the path of your old default python, this will be exactly this path. “`usr/local/bin/python3.7`” is the path of the new python you want as default. To check which path it is use `$whereis python`. The “`10`” at the end gives it priority 10 (highest). Typing `$python3 –version` should now give you your current and newly installed version of python, here python3.7.

Note that if you want to use PyRpl, you'll need Python 3.6, due to PySide. On Ubuntu 16.04.7 you can upgrade Python as described here.

To install pyrpl via pip first upgrade pip if you have a lower version than pip 21.0 with `$pip install –upgrade pip`. Then use `$sudo pip install pyrpl` (this takes a while). If installing PyNaCl takes too long, it's probably due to a bug in libsodium, which you can fix by installing PyNaCl manually with `$SODIUM_INSTALL=system pip install pynacl` before installing PyRpl. If you now type in `$python3` you can check whether the installation worked via `»>import pyrpl`. If it didn't work it will return something like `Traceback (most recent call last): File “<stdin>”, line 1, in <module> ImportError: No module named pyrpl`. Troubleshooting: if `&sudo -H pip install pyrpl` returned error: `sudo: unable to execute /usr/local/bin/pip3: no such file in directory`use `&sudo -H pip3.x install pyrpl` with 3.x being your installed python version. You can also check whether you have pip in the given directory with `$whereis pip`.

Used Boards in the Atom Lab

MAC	Hostname	Root Password	Location
rp-f07da9	mg-rp1	magnesium24!	
rp-f07db1	mg-rp2	magnesium24!	
rp-f096f2	mg-rp3	magnesium24!	
rp-f097be	mg-rp4	magnesium24!	
rp-f096c1	mg-rp5	magnesium24!	
rp-f096cb	mg-rp6	magnesium24!	

The first 6 have Ubuntu 16.04.7, Python 3.5.2 and pip 20.3.4 installed.

PID-Controller Board in the Atom Lab

- PID-Controller: The scripts are on the MG Laptop under C:\Documents
  - Turn_PID_On.py is starting a fast and a slow PI-Controller and it's also checking if the laser is still in lock; if it's not the controllers are turned off and the output produces a linear ramp function until the intensity is high enough. The voltage at which this is the case is setted as the new offset of the slow PI and both are turned on again. Their outputs are summed and amplified externally.
  - Turn_PID_Off.py is just setting the gains to zero.