Disclaimer

This is not a commercial product, this is a research tool so there are some “rough edges” and limited documentation, if you need support you can contact me at (jean-daniel.deschenes@gel.ulaval.ca) – although I cannot guarantee that I will answer quickly all the questions, but I will do my best to do so.

This software was built starting from two existing code bases: the NIST digital control box software for most of the FPGA firmware and all the Python code, and a part of the Red Pitaya software for the Zynq embedded software and FPGA firmware. Both the NIST and the Red Pitaya code were released in the public domain with a specific license, both placed in the subfolder "Licenses".

This digital PLL software is also released under a license (BSD) placed in the same subfolder.

Additional information

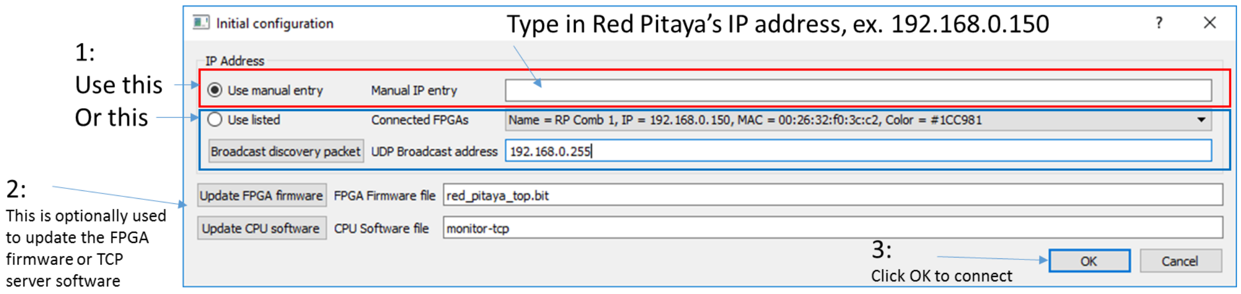
Additional information can be obtained from the instructions manual of the NIST digital control box (https://www.nist.gov/services-resources/software/fpga-based-digital-control-box-phase-stabilization-frequency-comb). The Red Pitaya digital phase-locked loop is based on this software and thus shares many of the same functionality with some differences.

Installation instructions for Windows (for MacOSX, see below)

1. Install WinPython-64bit-3.6.1.0Qt5. Other versions of WinPython or Python + numpy + PyQt5 + pyqtgraph might also work, but have never been tested.
2. Use a SD card image writing tool such as WinDiskImager32 to write the SD card image file "red_pitaya_dpll_2017-05-31.img" to a 4 GB SD card that came with the Red Pitaya. Note that sometimes, SD cards have very slightly differing capacities, and WinDiskImager32 might give an error message saying that the image file does not fit the SD card. If the size difference is only a very small fraction of 4 GB, you can try truncating the .img file to the SD card exact capacity and try again. The .img file contains a sizeable amount of zeros at the end of the file and thus truncating it slightly shouldn't affect the operation.
3. Put the SD card in the Red Pitaya, connect an Ethernet cable to the Red Pitaya and connect it either to a router with a DHCP server, or directly in an Ethernet port of a PC.
   a. If using a router with a DHCP server, the Red Pitaya will be assigned an IP address in the router's subnet.
   b. If connecting directly to a PC running Windows, the Red Pitaya's DHCP client will time out after 20 seconds and start using IP address 192.168.0.150 by default. This default IP address is configurable by editing the file "/etc/dhcp/dhclient.conf" inside the Linux distribution running on the Zynq. You also need to set your PC's network configuration to be a fixed IP address in the same subnet (192.168.0.x, where x is any integer from 1 to 254).
4. Apply power to the Red Pitaya using the micro-usb port and the power supply that came with the Red Pitaya. Note that the Red Pitaya's supply is capable of providing 2 A and thus a standard PC USB port might not be able to supply enough current for the Red Pitaya and thus using the

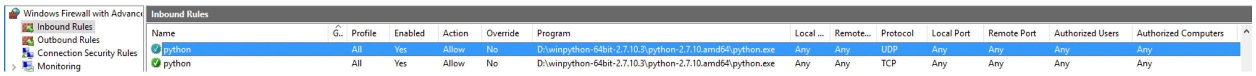
furnished power supply is recommended. The Red Pitaya then boots Linux; the process takes roughly 1-2 min before you can connect.

5. Open Spyder (Python IDE that comes with WinPython) and run XEM_GUI3.py, which is the top-level Python script to open the Digital PLL user interface. If Python was installed correctly, you should see a window open that looks like figure 1 (it might not open at the topmost of the desktop so look in your taskbar if you cannot find it).
6. In this first window, you have two choices:
    a. You can either type in directly the Red Pitaya's IP address if you know it already and check the "Use manual entry" radio button,
    b. or you can check the "Use listed" radio button if you can see the Red Pitaya in the list next to "Connected FPGAs". If this list is not populated, here are a few pointers that might help debugging this:
        i. you might have an issue with your network's IP address assignations (they all need to be on the same subnet),
        ii. If your network uses a different subnet than 192.168.0.XXX, for example 192.168.1.XXX, you need to change the content in the textbox next to "UDP Broadcast address" to the correct broadcast address. This is always equal to an address that has the first part equal to your network's subnet, and the last number(s) to "255". This broadcast mechanism is used to automatically detect all Red Pitaya's on the network by sending a broadcast packet, to which a program inside Linux (udp_discovery) will answer, allowing to figure out the Red Pitaya's IP and MAC addresses (used as a unique serial number for storing configuration files).
        iii. You might have an issue with your Windows Firewall configuration. You need to make sure to allow the python executable (python.exe inside "WinPython-64bit-2.7.10.3\python-2.7.10.amd64") to receive all UDP packets from any port (see figure 2). Make sure that this exception applies to all "profiles" to avoid having to determine which profile was applied to the network that you have created for the Red Pitaya.
7. Once you have used either of the two options and pointed to the correct IP address for the Red Pitaya, click OK to connect.
8. At this point, the window shown in figure 3 will appear if the connection succeeded. Check "Auto-refresh" to display the diagnostics signals in the interface.


Type in Red Pitaya's IP address, ex. 192.168.0.150
1:
Use this
Or this
2:
This is optionally used to update the FPGA firmware or TCP server software
3:
Click OK to connect
Initial configuration
IP Address
Broadcast discovery packet
Update FPGA firmware
Update CPU software
OK
Cancel

(●) Use manual entry — Manual IP entry:
( ) Use listed — Connected FPGAs: Name = RP Comb 1, IP = 192.168.0.150, MAC = 00:26:32:f0:3c:c2, Color = #1CC981
UDP Broadcast address: 192.168.0.255
FPGA Firmware file: red_pitaya_top.bit
CPU Software file: monitor-tcp

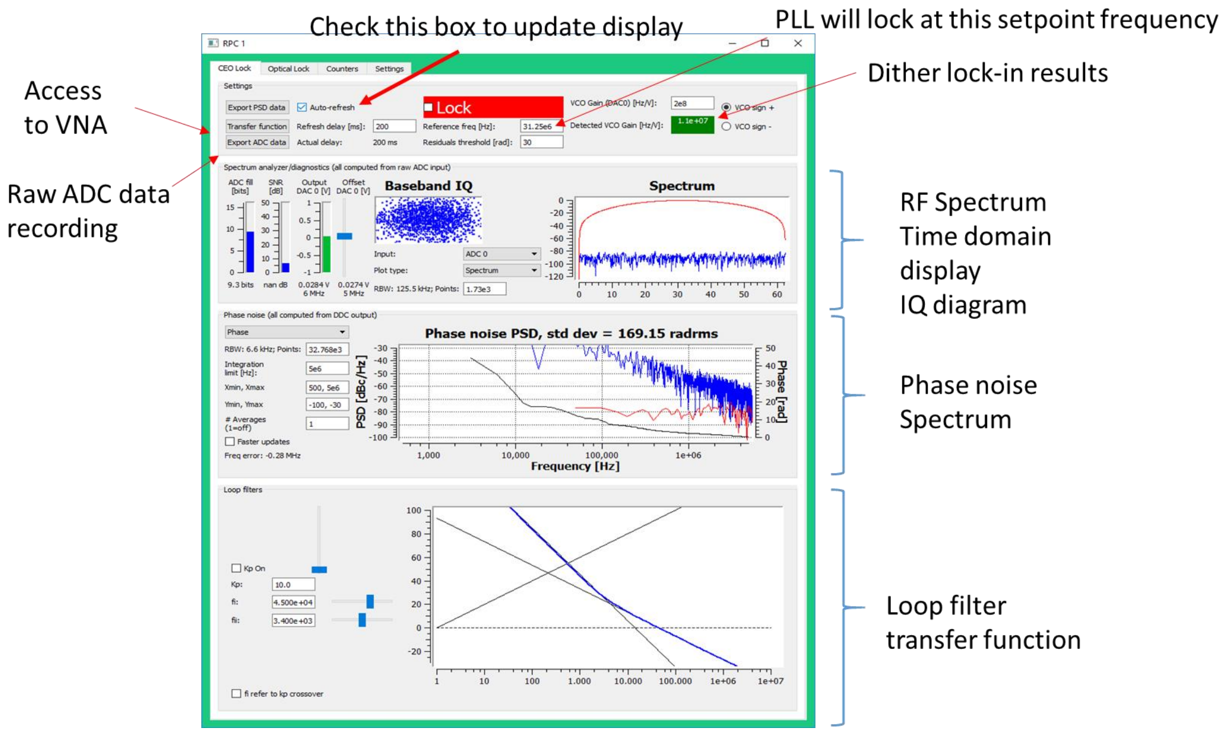
Figure 1 - First window that opens after running XEM_GUI3.py, which allows connecting to a Red Pitaya on the network, and optionally updating its FGPA firmware or the TCP server running inside Linux on the Zynq's processor.



Inbound Rules

Name	G..	Profile	Enabled	Action	Override	Program	Local ...	Remote...	Protocol	Local Port	Remote Port	Authorized Users	Authorized Computers
python		All	Yes	Allow	No	D:\winpython-64bit-2.7.10.3\python-2.7.10.amd64\python.exe	Any	Any	UDP	Any	Any	Any	Any
python		All	Yes	Allow	No	D:\winpython-64bit-2.7.10.3\python-2.7.10.amd64\python.exe	Any	Any	TCP	Any	Any	Any	Any

Figure 2 - Example entry in Windows Firewall with advanced security (tested on Windows 10) to allow the udp-discovery mechanism to work.



Settings: [✓ Auto-refresh]; Refresh delay [ms]: 200; Actual delay: 200 ms; Reference freq [Hz]: 31.25e6; Residuals threshold [rad]: 30; VCO Gain (DAC0) [Hz/V]: 2e8; Detected VCO Gain [Hz/V]: 1.1e+07; [✓ VCO sign +] VCO sign -

Spectrum analyzer: ADC fill [bits]: 9.3 bits; SNR [dB]: nan dB; Output DAC 0 [V]: 0.0284 V 6 MHz; Offset DAC 0 [V]: 0.0274 V 5 MHz; Input: ADC 0; Plot type: Spectrum; RBW: 125.5 kHz; Points: 1.73e3

Phase noise: Phase; RBW: 6.6 kHz; Points: 32.768e3; Integration limit [Hz]: 5e6; Xmin, Xmax: 500, 5e6; Ymin, Ymax: -100, -30; # Averages (1=off): 1; Faster updates; Freq error: -0.28 MHz

Loop filters: Kp On; Kp: 10.0; fi: 4.500e+04; fii: 3.400e+03; fi refer to kp crossover

Figure 3 - Main user interface. Refer to the NIST digital lockbox manual or the PowerPoint presentation for more details on each subsection.

Differences between the Red Pitaya version and the NIST Digital Phase-lock box

- The Red Pitaya has 2x125 MS/s DACs (+/- 1V range) while the NIST lock-box has 2x100 MS/s DACs with various output ranges up to +/- 8V and 1x 1 MS/s DAC with 0-55V output range at higher resolution.

- The Red Pitaya software can only record a limit of 32768 points for the phase noise measurement (and raw ADC data measurement), meaning that the lower frequency limit on the phase noise measurement is limited to a much higher frequency than on the NIST lockbox, which could record up to 50 Msamples (although a much lower number of points should be used for phase noise estimation unless the processing time will be very long – the raw data export can easily work up to that limit).
- Using an external clock for the ADC on the Red Pitaya requires a small hardware modification of the board (see http://redpitaya.readthedocs.io/en/latest/doc/developerGuide/125-14/extADC.html). I have never tested this personally but it should work if the correct signal amplitude, frequency and duty cycle are respected.

Installation instruction for MaxOSX for Python+packages - Quick version

Install python 3.6 from: https://www.python.org/downloads/mac-osx/

In a terminal: /Library/Frameworks/Python.framework/Versions/3.6/bin/pip3 install scipy matplotlib PyQt5 pyqtgraph

This should install all the required packages and dependencies.

You can then checkout or simply download the code base at:

https://github.com/jddes/Frequency-comb-DPLL/tree/DPLL-python3

Follow the other instructions in the Windows installation instructions, starting at item 2. Replace the Spyder IDE by IDLE within the /Applications/Python3.6/ folder.

Installation instructions Python+packages for Mac OS X – Detailed version

Installing python 3.x for Mac OS X
(tested with 3.5 and 3.6, please replace all the 3.x below with appropriate version number)

https://www.python.org/downloads/mac-osx/

This installs a packaged version of python, which is independent from any other system version that you might or might not have installed.

In your applications folder, you'll have a Python3.X folder containing a rather simple integrated development environment (IDLE) that you can use to write an run python scripts/programs.

This encapsulated python version looks for its packages and related files in Library/Frameworks/Python.framework/Versions/3.X

Each installed version of this packaged MacOS python distribution will look for files in the correspondingly numbered version folder, meaning that you can easily have several non interfering versions simultaneously installed.

It is important to understand that once a python module is opened with the IDLE, the search path for custom libraries and packages is:

/Library/Frameworks/Python.framework/Versions/3.x/lib/python3.x/site-packages

one can confirm this by opening the IDLE and selecting "path browser" from the file menu, making sure this path is listed in the window.

Installing libraries

This is most easily done using "pip", but one has to make sure the proper version of pip is used.

In this case, we want to use:

/Library/Frameworks/Python.framework/Versions/3.x/bin/pip3

Open a terminal window, The command below should install commonly needed libraries:

/Library/Frameworks/Python.framework/Versions/3.6/bin/pip3 install numpy scipy matplotlib ipython jupyter pandas sympy nose

You might be asked to install the apple developper tools, just accept

You might want to check if your packages are indeed in:

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.x/site-packages

If you can see folders named "numpy" and "mathplotlib", then the installation worked

Installing pyQt and pyqt graph

/Library/Frameworks/Python.framework/Versions/3.x/bin/pip3 install PyQt5 pyqtgraph

/Library/Frameworks/Python.framework/Versions/3.6/bin/pip3 install pyqtgraph

This should install PyQt5, SIP and pyqtgraph

Running the python code to connect to the RedPitaya PLL

You can then checkout or simply download the code base at:

https://github.com/jddes/Frequency-comb-DPLL/tree/DPLL-python3

Open and run the file “XEM_GUI3.py” using the IDLE within the /Applications/Python3.x/ folder