Using Red Pitaya + PyRPL for digital flux feedback

Logan Bishop-Van Horn, logan.bvh@gmail.com, Updated 2018-10-23

External Resources:

- Red Pitaya Documentation
- PyRPL Documentation

Overview:

Red Pitaya is a data acquisition and signal processing board with high speed (125 MHz) 14 bit ADCs and DACs, and an FPGA for signal processing, all run by a small Linux OS. It is intended for use as a low-cost digital oscilloscope, function generator, etc. Out of the box, Red Pitaya's basic functions can be run through a web interface.

PyRPL is an open source software package that replaces the out-of-the-box Red Pitaya software to turn the board into a very capable digital signal processing tool. PyRPL includes FPGA bit files that define several signal processing modules, including multiple PI controllers, a 2-channel 50 MHz oscilloscope, multiple 50 MHz function generators, IQ demodulators, network analyzers, and more. PyRPL also includes a Python interface to these hardware module, including a GUI.

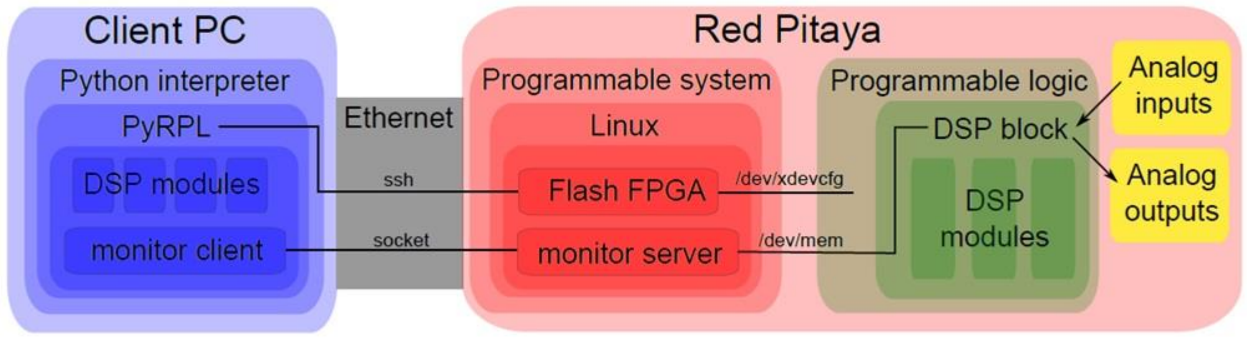


Figure 1. *Schematic of the Red Pitaya + PyRPL architecture.*

Connecting to the Red Pitaya:

After setting up your Red Pitaya board following the instructions in the Red Pitaya Documentation, connect the Red Pitaya to the computer via Ethernet and a LAN to USB adapter. Connecting the board using a LAN to USB adapter keeps the Red Pitaya off the university network, eliminating security hassles. At this point, you should be able to control the Red Pitaya using the off-the-shelf software. To do this, open a web browser like Chrome and in the address bar enter the Red Pitaya's MAC address, e.g. **rp-F05678.LOCAL/** (the MAC address can be found on a sticker on the Red Pitaya board, and it's not case-sensitive). Note that the Red Pitaya is not connected to the internet and your computer need not be connected to the internet to use this web interface.

If you can see the off-the-shelf Red Pitaya software in your browser (see below), the board is connected correctly!

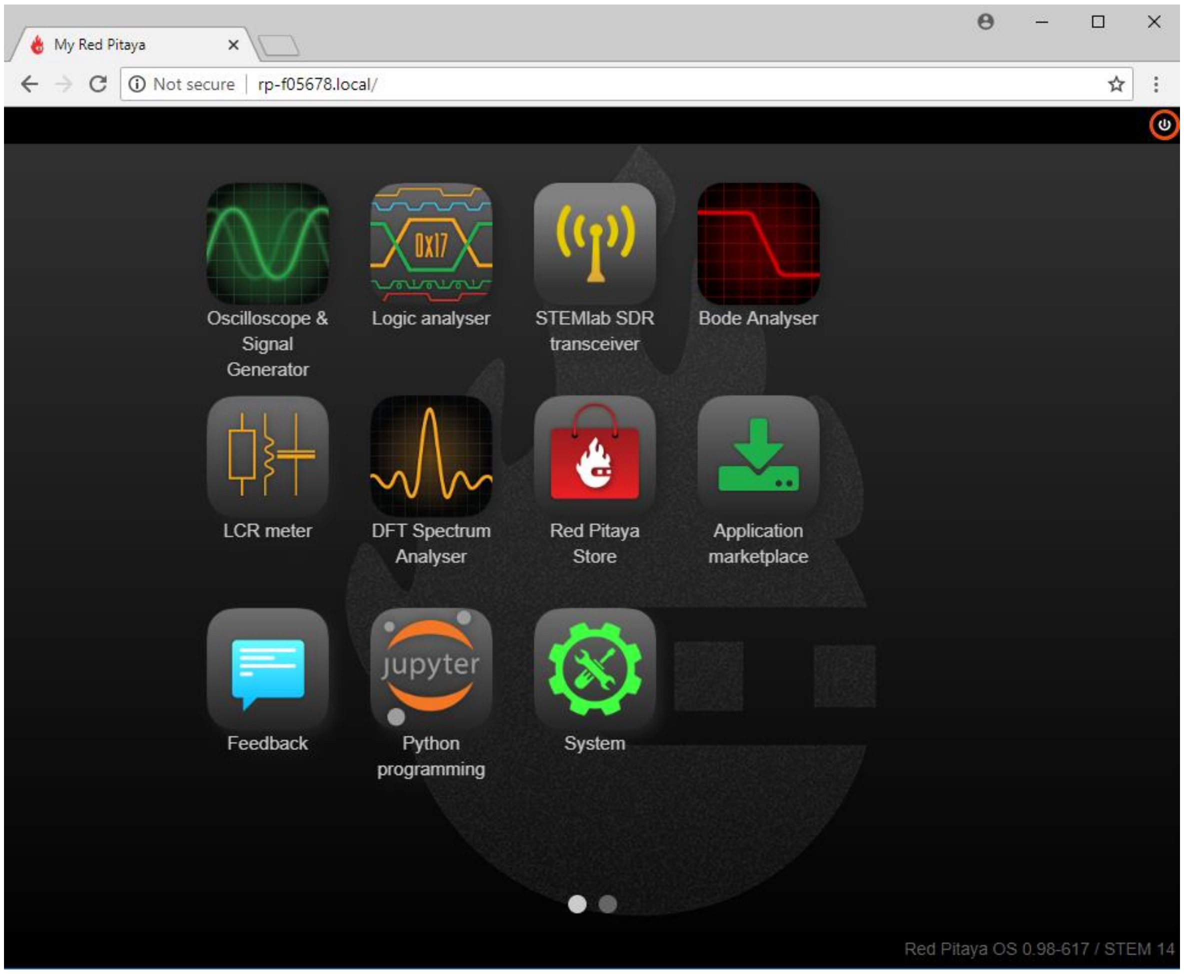


Figure 2. *Off-the-shelf Red Pitaya software, running in Chrome.*

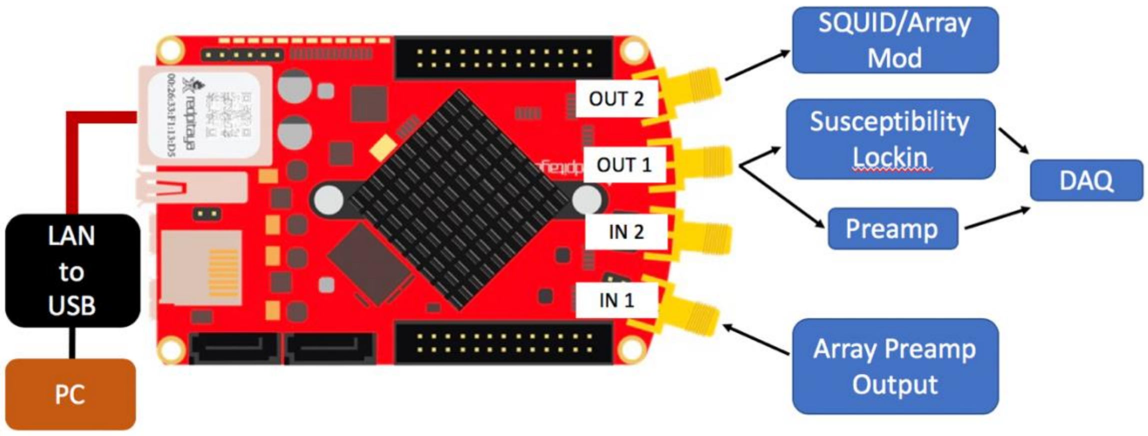
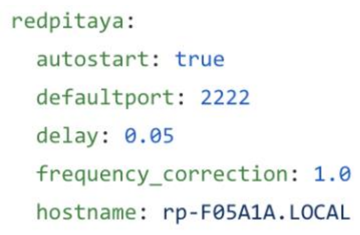


Figure 3. *Connecting the Red Pitaya board for SQUID tuning/flux feedback.*

Configuration Files:

Different PyRPL GUI layouts can be defined in YAML configuration files located at **C:\Users\<your-user>\pyrpl_user_dir\config**. A useful layout for SQUID tuning and monitoring is squid_lockbox.yml. In order to use this configuration, download the .yml file from Github, open it in a text editor, enter the MAC address of your Red Pitaya board under “hostname”, and save to **C:\Users\<your-user>\pyrpl_user_dir\config**.



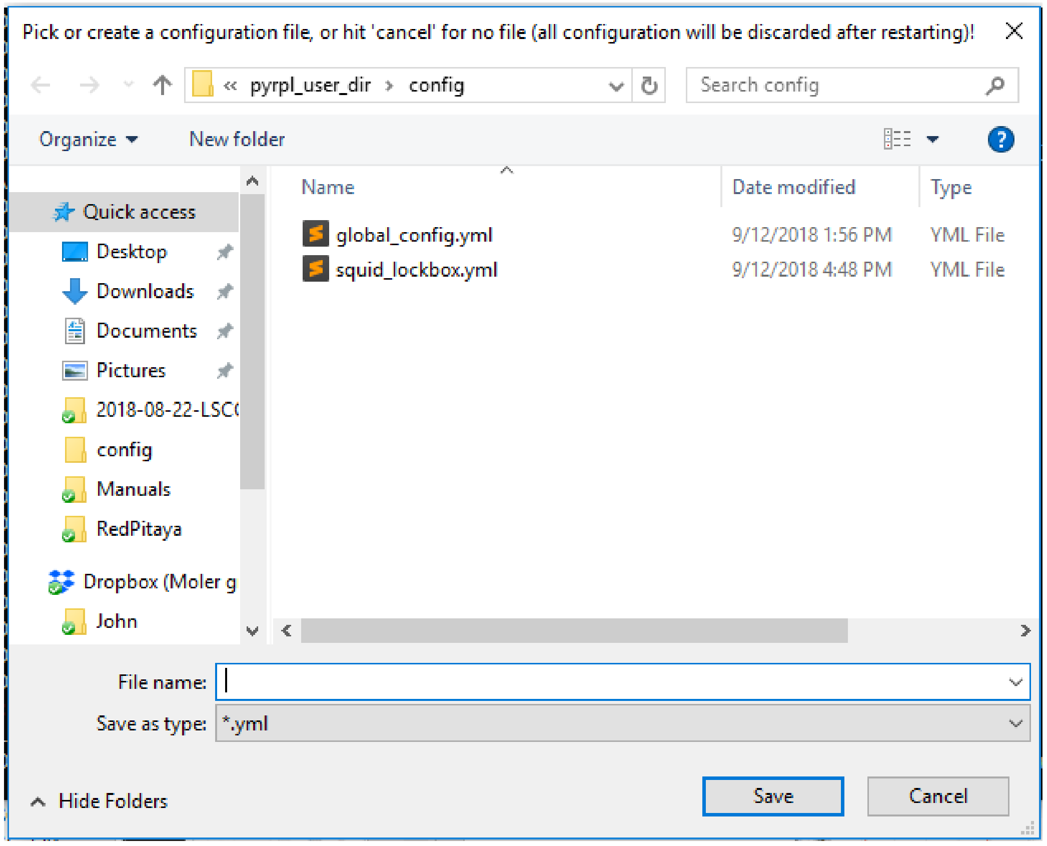
```
redpitaya:
  autostart: true
  defaultport: 2222
  delay: 0.05
  frequency_correction: 1.0
  hostname: rp-F05A1A.LOCAL
```

PyRPL Quickstart:

NOTE: This method for starting PyRPL does not allow you to run Python scripts to control the Red Pitaya (e.g. to automatically relock the SQUID). For that, you need to start PyRPL through the Python command line.

The easiest way to get started with PyRPL is to download the pre-packaged binary for your operating system (e.g. **pyrpl-windows.exe**). After installing the binary, open the pyrpl-windows desktop application (see icon to the right). This will open up two windows, a black and white command line prompt (you don’t have to do anything with this, just don’t close it), and a file selection dialog box pointing to **C:\Users\<your-user>\pyrpl_user_dir\config** (see Figure 4 below). This directory contains files that define the GUI and Red Pitaya configuration. A GUI for SQUID tuning and feedback is defined in squid_lockbox.yml. If you want to use this one, select it in the dialog box and click Save.

pyrpl-win...


Pick or create a configuration file, or hit 'cancel' for no file (all configuration will be discarded after restarting)!

pyrpl_user_dir › config

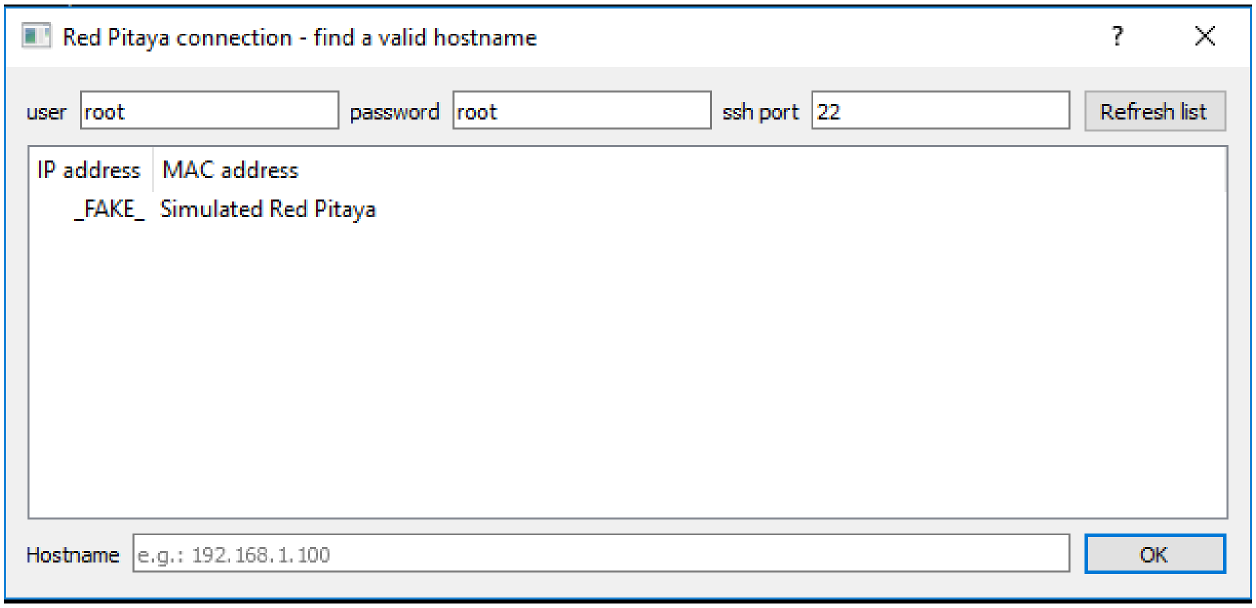
Name	Date modified	Type
global_config.yml	9/12/2018 1:56 PM	YML File
squid_lockbox.yml	9/12/2018 4:48 PM	YML File

File name:

Save as type: *.yml

Figure 4. *Configuration selection dialog box.*

If you want to start from scratch and build a new GUI configuration, enter a name for the configuration in the dialog box and click Save. This will bring up the window shown in Figure 6 below. Enter your Red Pitaya MAC address in the box labeled Hostname and click OK.


Red Pitaya connection - find a valid hostname

user root   password root   ssh port 22

IP address	MAC address
FAKE	Simulated Red Pitaya

Hostname e.g.: 192.168.1.100

Figure 6. *Red Pitaya connection dialog box.*

This will bring up a blank GUI that can be populated by selecting modules to activate.

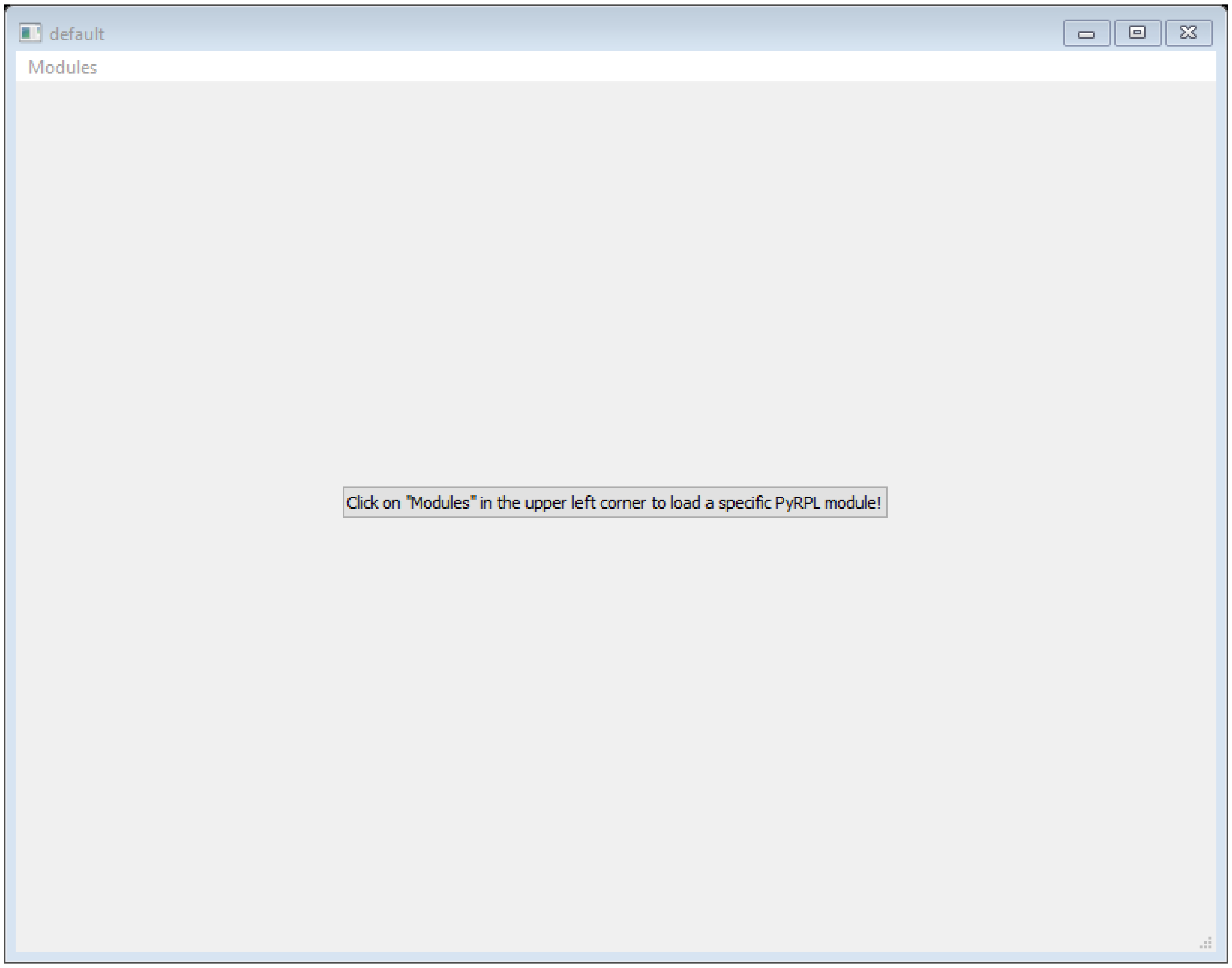

default
Modules
Click on "Modules" in the upper left corner to load a specific PyRPL module!

Figure 7. *Unpopulated PyRPL GUI.*

After you populate the GUI, the layout will be saved as a .yml file in **C:\Users\<your-user>\pyrpl_user_dir\config**.

Running PyRPL through Python:

To run PyRPL through Python, make sure that there is a conda environment called **pyrpl-env** located at C:\Users\<your-user>\Anaconda3\envs. If not, create it by opening an Anaconda Prompt and running

```
conda create -y -n pyrpl-env numpy scipy paramiko pandas nose pip pyqt qtpy pyqtgraph pyyaml
```

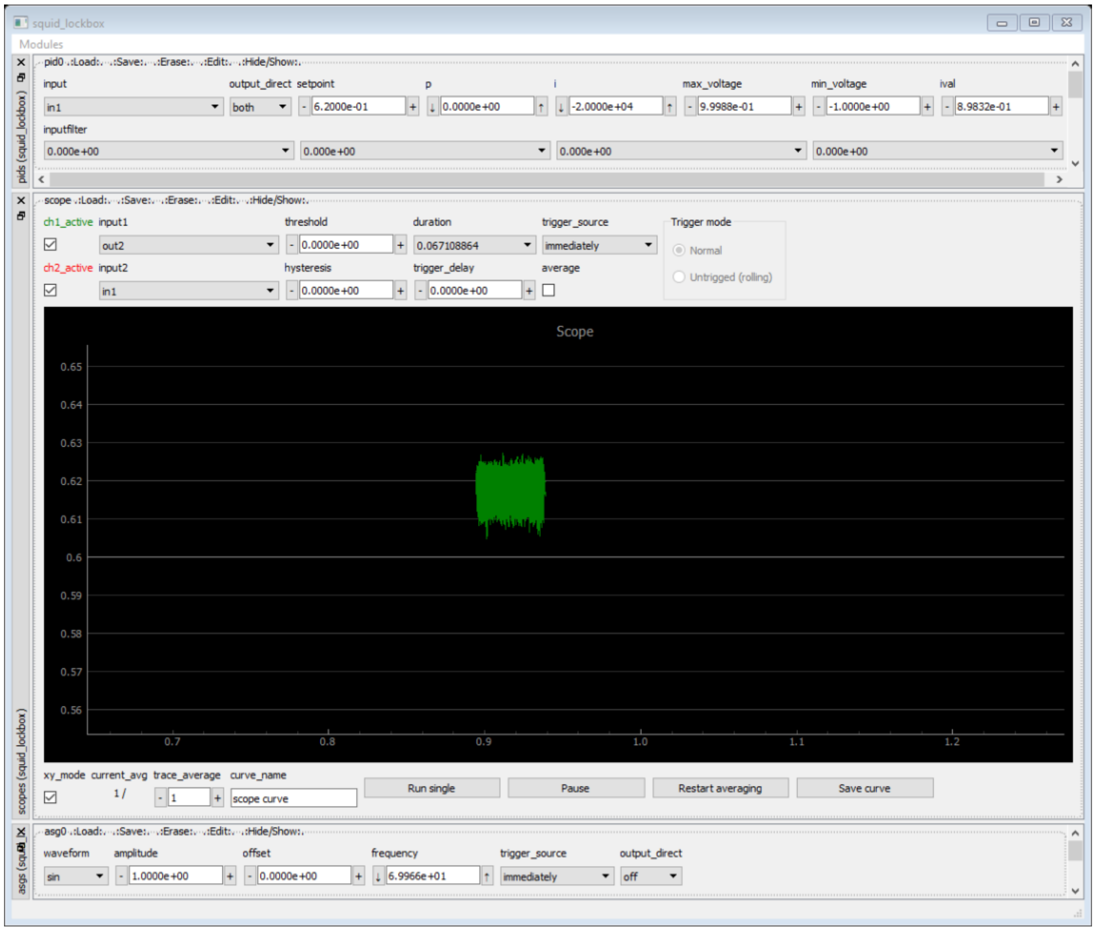
After creating this environment (or verifying that it already exists), run

```
activate pyrpl-env

pip install quamash ipython pyrpl
```

At this point, PyRPL and all of its dependencies should be installed in a conda environment called pyrpl-env.

To run PyRPL, activate pyrpl-env, start the Python interpreter, import the Pyrpl class, and create a new instance of it. This should open up the PyRPL GUI with the pid, scope, and asg (signal generator) modules displayed.

```
>> activate pyrpl-env

>> ipython

In [1]: from pyrpl import Pyrpl

In [2]: p = Pyrpl(‘squid_lockbox’)
```



pid0 .:Load:. .:Save:. .:Erase:. .:Edit:. .:Hide/Show:.

input	output_direct	setpoint	p	i	max_voltage	min_voltage	ival
in1	both	6.2000e-01	0.0000e+00	-2.0000e+04	9.9988e-01	-1.0000e+00	8.9832e-01

inputfilter: 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00

scope .:Load:. .:Save:. .:Erase:. .:Edit:. .:Hide/Show:.

ch1_active	input1	threshold	duration	trigger_source
✓	out2	0.0000e+00	0.067108864	immediately

ch2_active	input2	hysteresis	trigger_delay	average
✓	in1	0.0000e+00	0.0000e+00	☐

Trigger mode: [✓ Normal] Untrigged (rolling)

xy_mode	current_avg	trace_average	curve_name
✓	1 /	1	scope curve

asg0 .:Load:. .:Save:. .:Erase:. .:Edit:. .:Hide/Show:.

waveform	amplitude	offset	frequency	trigger_source	output_direct
sin	1.0000e+00	0.0000e+00	6.9966e+01	immediately	off

Figure 8. *PyRPL GUI configured for SQUID tuning and flux feedback.*

Next, create a handle for the Red Pitaya object:

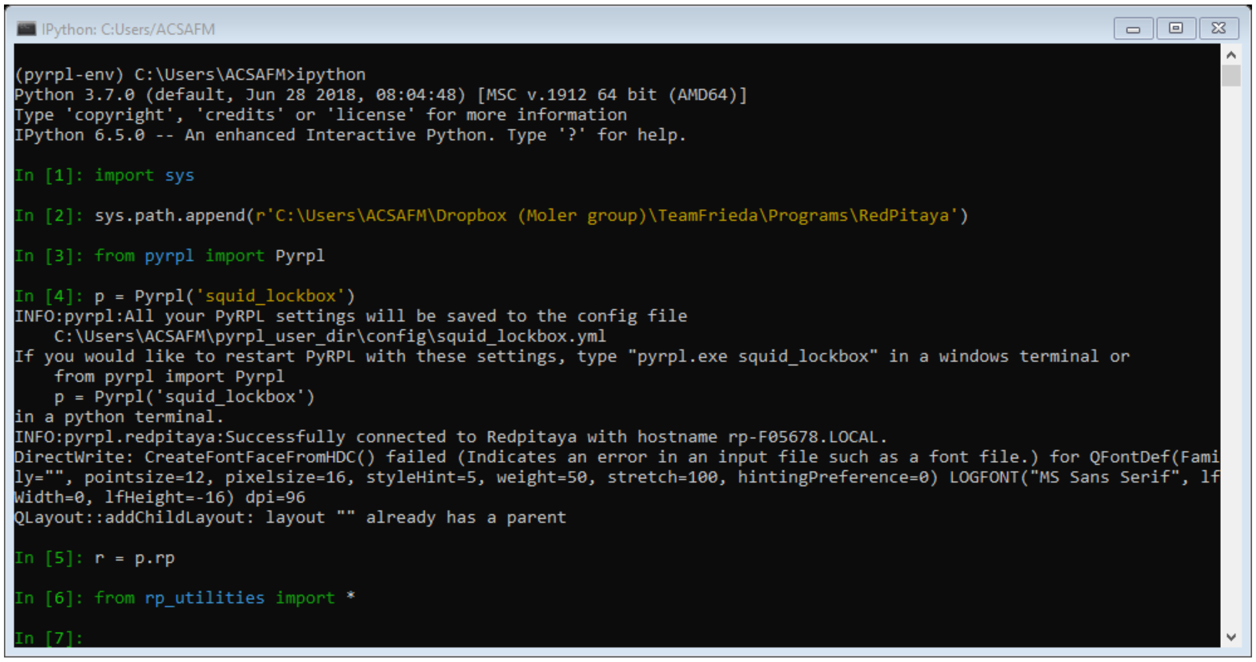
```
In [3]: r = p.rp
```

Now in addition to being able to configure the Red Pitaya modules using the GUI, you can also query and set them at the command line. For example, to query the current value of the trace displayed on scope channel 1, enter

```
In [4]: r.scope.voltage_in1
```

Or, if you want to zero the integrator of pid0, enter

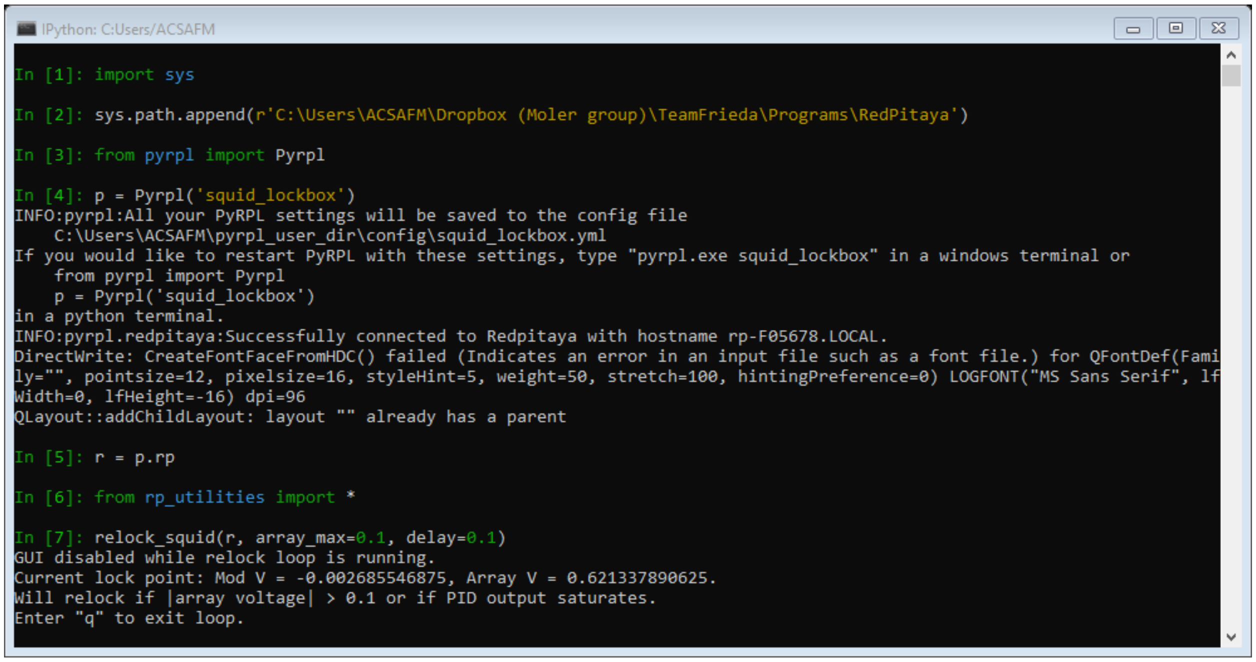
```
In [5]: r.pid0.ival = 0
```



```
(pyrpl-env) C:\Users\ACSAFM>ipython
Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.5.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import sys

In [2]: sys.path.append(r'C:\Users\ACSAFM\Dropbox (Moler group)\TeamFrieda\Programs\RedPitaya')

In [3]: from pyrpl import Pyrpl

In [4]: p = Pyrpl('squid_lockbox')
INFO:pyrpl:All your PyRPL settings will be saved to the config file
    C:\Users\ACSAFM\pyrpl_user_dir\config\squid_lockbox.yml
If you would like to restart PyRPL with these settings, type "pyrpl.exe squid_lockbox" in a windows terminal or
    from pyrpl import Pyrpl
    p = Pyrpl('squid_lockbox')
in a python terminal.
INFO:pyrpl.redpitaya:Successfully connected to Redpitaya with hostname rp-F05678.LOCAL.
DirectWrite: CreateFontFaceFromHDC() failed (Indicates an error in an input file such as a font file.) for QFontDef(Fami
ly="", pointsize=12, pixelsize=16, styleHint=5, weight=50, stretch=100, hintingPreference=0) LOGFONT("MS Sans Serif", lf
Width=0, lfHeight=-16) dpi=96
QLayout::addChildLayout: layout "" already has a parent

In [5]: r = p.rp

In [6]: from rp_utilities import *

In [7]:
```

Figure 9. *Anaconda Prompt showing how to start PyRPL using Python.*

We can now run functions to control the Red Pitaya by passing the handle `r` as a function argument. For example, there is a function called `relock_squid` that periodically checks if the SQUID is unlocked and, if so, zeros the integrator to relock.



```
In [1]: import sys

In [2]: sys.path.append(r'C:\Users\ACSAFM\Dropbox (Moler group)\TeamFrieda\Programs\RedPitaya')

In [3]: from pyrpl import Pyrpl

In [4]: p = Pyrpl('squid_lockbox')
INFO:pyrpl:All your PyRPL settings will be saved to the config file
    C:\Users\ACSAFM\pyrpl_user_dir\config\squid_lockbox.yml
If you would like to restart PyRPL with these settings, type "pyrpl.exe squid_lockbox" in a windows terminal or
    from pyrpl import Pyrpl
    p = Pyrpl('squid_lockbox')
in a python terminal.
INFO:pyrpl.redpitaya:Successfully connected to Redpitaya with hostname rp-F05678.LOCAL.
DirectWrite: CreateFontFaceFromHDC() failed (Indicates an error in an input file such as a font file.) for QFontDef(Fami
ly="", pointsize=12, pixelsize=16, styleHint=5, weight=50, stretch=100, hintingPreference=0) LOGFONT("MS Sans Serif", lf
Width=0, lfHeight=-16) dpi=96
QLayout::addChildLayout: layout "" already has a parent

In [5]: r = p.rp

In [6]: from rp_utilities import *

In [7]: relock_squid(r, array_max=0.1, delay=0.1)
GUI disabled while relock loop is running.
Current lock point: Mod V = -0.002685546875, Array V = 0.621337890625.
Will relock if |array voltage| > 0.1 or if PID output saturates.
Enter "q" to exit loop.
```

Figure 10. *Anaconda Prompt showing how to start PyRPL using Python and run the relocking script.*

Note that, as above in the **PyRPL Quickstart** section, you can create GUI configuration from scratch at the command line:

```
>> activate pyrpl-env

>> ipython

In [1]: from pyrpl import Pyrpl

In [2]: p = Pyrpl('new_config_name')
```

This will open up the Red Pitaya connection dialog box (Figure 6), into which you can enter the Red Pitaya MAC address, which will open up an unpopulated PyRPL GUI (Figure 7). Once you populate this GUI, the configuration will be saved as **C:\Users\<your-user>\pyrpl_user_dir\config\new_config_name.yml**.