This tutorial contains a shortcut to use the Red Pitaya exactly as we do in this paper, a brief introduction to the Red Pitaya hardware, getting started with Red Pitatya and how to use it for a negative feedback controller. It is worth mentioning that Red Pitaya and its software are evolving pretty fast, the instructions and open source programs on official website might be slightly different from what we have here, as we stopped following the official updates since August, 2016.

Shortcut:

1. Connect to Red Pitaya via SSH.
2. Replace the '*controllerhf.so*' in /opt/redpitaya/www/apps/scope+pid/ with the file attached in the guide.
3. Modify the '*fpga.conf*' in /opt/redpitaya/www/apps/scope+pid/ so that it reads: opt/redpitaya/fpga/delay00.bit
4. Copy the delay00.bit attached in the guide to Red Pitaya's directory opt/redpitaya/fpga/

Above all, Red Pitaya has an ARM processor (like in Raspberry Pi) and a FPGA SoC. A Linux system runs on the ARM processor, while the FPGA runs on its own. Both of them have access to the board RAM, which enables real-time communication between them. When we use Red Pitaya as the feedback controller of our imaging system, the FPGA chip performs data acquisition, feedback computation and so on; the Linux systems runs a web service, which we can access through WIFI or Ethernet. By simply operating the webpage, we can modify the feedback parameter and visualize the input signal. To use the Red Pitaya as we do, you will need to do several things including programming the FPGA, reconfigure the FPGA and modifying the web service.

To begin with, we need to perform a few starter steps as illustrated here: http://blog.redpitaya.com/quick-start/. These steps will initialize the Red Pitaya board and get a Debian OS running on the board. For the type of connection, you could use console connection through USB, but I strongly recommend using the Ethernet connection, because you will have to use it later to access the FPGA chip in real time.
Next, with the IP address, establish a SSH connection between the Linux system and your lab computer. To do this, you can use OpenSSH on Linux, or install a Cygwin with OpenSSH on Windows and type *ssh: root@(Red Pitaya's IP)*, then use *root* as password. Please refer to:http://wiki.redpitaya.com/index.php?title=SSH_connection.

Then, we will use Xilinx Vivado 2015.4 on Windows to generate the binary file for re-programming FPGA. The software is freely available on Xilinx website as long as you registered a student account. Then, download a WebPack License. With the Vivado software installed, you can start with our Vivado projects:
https://www.dropbox.com/s/018tb5ox68oym8d/project.7z?dl=0
After modifying the code, you can hit "generating bit stream" on the bottom left corner. The newly generated bit stream file, called "red_pitaya_top.bit", can be

find in $(project path)\redpitaya.runs\impl_1. Later this bit file will be used to re-program the FPGA on Red Pitaya.
If you wish to start with code from the official github, please follow the steps below:

1. Clone files from https://github.com/RedPitaya/RedPitaya/tree/master/fpga
2. Launch Vivado, at the bottom of your launcher, you should have TCL Console. Go to the fpga directory you downloaded previously with the command cd as on a Linux OS.
3. Type *source red_pitaya_vivado_project.tcl*
4. Now, you have a project directory inside your fpga directory, open this .xpr file in Vivado.

Next step is to use the new binary file on Red Pitaya. After logging in onto the Red Pitaya's Debian system using SSH, enable writing by typing *rw*. After that, terminate the SSH connection and copy the bit stream file to */home/redpitaya/fpga* using SCP. Then, log on to Red Pitaya agai and try to locate a series of files called "fpga_0.9x.bit". The number indicate the version of FPGA configuration file. Both the file path and the version number might have been updated. Let's say the highest version number is 0.96 and the path is */opt/redpitaya/fpga*. Backup the original "fpga_0.96.bit" and replace it with the new bit stream file.

By now, the FPGA re-configuration is completed. You could skip the last step, and use the web page as it is. To do this, you need first to 'activate' the new configuration. Open a Chrome or Firefox browser (IE or Edge won't work) and type in Red Pitaya's IP address. You will be directed to Red Pitaya's main page. Click PID & Scope app. On the left, there is an oscilloscope window which can show you the input signal; on the right, there are a bunch of options. Among them, unfold the bottom one called "PID Controller". To use the PID function, please check the correspondence between the name shown in the web page, and the real function below.

1. For PID11, all the parameters are used as indicated.

2. Set point in PID 12 is input 1 offset. Input1 = ADC output – input1 offset.

3. Kp in PID 12 is input 2 offset. Input2 = ADC output – input2 offset.

4. Ki in PID 12 is the switch to bypass the feedback. If it is set to a number > 2000, feedback computation will be bypassed and the DAC channel 1 will send out pid12_kd / 8191 * 1(v).

5. Kd in PID 12 is the output threshold, the maximum DAC output value is setpoint / 8191 * 1(v)

6. PID 21 are test ports

7. PID 22 are reversed for future use.

The last step is to modify the web page to make is consistent with the new FPGA configuration. We could do this by simply replace the old webpage front end by 'controllerhf.so' attached here. First, find the webpage file, called "controllerhf.so" on your Red Pitaya. In my case, it's located in */opt/redpitaya/www/apps/assets*. Then, replace the "controllerhf.so" in scope+pid folder with the new one.
If you want to change the web page or the web service, please check this link. https://github.com/RedPitaya/RedPitaya/blob/master/apps-free/README.md. An Ubuntu 1404 LTS OS is required. To setup the environment, run the following commands:

1. sudo
wget https://releases.linaro.org/14.11/components/toolchain/binaries/arm-linux-gnueabihf/gcc-linaro-4.9-2014.11-x86_64_arm-linux-gnueabihf.tar.xz

2. tar xvf gcc-linaro-4.9-2014.11-x86_64_arm-linux-gnueabihf.tar.xz

3. export PATH=$PATH:$(*your path here*)/gcc-linaro-4.9-2014.11-x86_64_arm-linux-gnueabihf/bin

4. export CROSS_COMPILE=arm-linux-gnueabihf-

5. make clean all
Basically, The compilation generate a controllerhf.so which is a dynamic link file and a fpga.conf.

After this long and tedious procedure, it should be ready to use. You can type in Red Pitaya's IP address in a browser and click scope + pid app. If you do not see the app, you need first to go to Red Pitaya's app store (called Bazaar) and hit install.

In this email I did not include detailed explanations of my code, as the program is developed for our customized imaging system and might not be suitable when applied to other setups. Please let me know if you want to use the exact same code (or with minor modification), I can write you a document about the code and how the PID feedback is implemented.

Finally, as a reminder, do not forget to change the root password of Red Pitaya. I was using the default password when someone hacked into my board and used it as a spam email zombie.

Best,
Ron