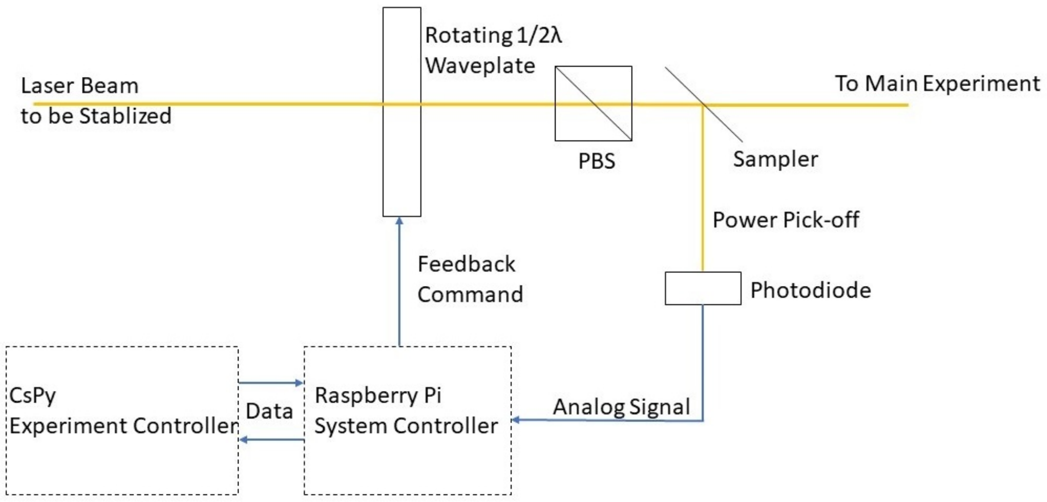
User Manual for the Power Stablization Box

Xiaoyu Jiang

08/15/2017

1 Introduction

The Rotating-waveplate-based DC Noise Eater is a system designed to deal with slow power fluctuations of laser beams in experiments. The basic diagram of the feedback system can be shown as follow:



In this system, a linealy polarized laser beam goes through a $1/2\lambda$ mounted on a rotating stage. Then the power of the beam gets measured by a photodiode, and the measurement is sent to a Raspberry Pi(RPi). The RPi compares the measured laser power to the desinated value, calculates the feedback signal, and sends it to the rotating stage. The stage will rotate accordingly to change the polarization of the beam, which also changes the power of the beam that goes through the PBS. The loop will be executed repeatedly to eventually stablize the laser power to the setpoint. At the same time, the RPi can also be identified as a TCP/IP instrument on the CsPy experiment controller, which enables data exchange between them. CsPy can receive measurements from the RPi, and dynamically set experiment parameters.

2 Installation

Before setting up the system, we need to initialize the components used in the system.

2.1 Configure the Raspberry Pi

Installing Operation System: We are using Raspbian Jessie with desktop as operation systems on the RPis. A micro SD card and image writing tools are needed. The image file can be downloaded from https://www.raspberrypi.org/downloads/raspbian/.

First Boot: Once Raspbian is installed, the RPi is ready to be powered up. Connect the RPi to monitors via HDMI and to mouse and keyboard via USB, then turn on the power supply. During the first boot, some settings are needed, so that all the following operations can be done with SSH(Secure Shell).

Setting interfacing option:
Open the terminal window and type the following command:
```
$ sudo raspi-config
```
Then in 'Interfacing options', choose to enable SSH and SPI.

Identify IP address:
First have the RPi connected to the internect. For now we are using wireless network UWNet. Then in command window, use:
```
$ sudo ifconfig
```
Next to the `wlan0` entry you will see the IP address of the RPi, indicated by `inet addr`
Record the IP address.

Further Settings: From here all other configurations can be done remotely with SSH. So all the following operations will be done in the command terminal.

Download wiringpi python package:
Wiringpi is a package designed to make the communication between the RPi and its GPIO pins more convenient and effective. In command window, enter the following commands:
```
$ sudo apt-get install python-dev python-setuptools swig
$ git clone –recursive https://github.com/WiringPi/WiringPi-Python.git
$ cd WiringPi-Python
$ git submodule update –init
$ cd WiringPi
$ sudo ./build
$ cd ..
$ swig 2.0 -python wiringpi.i
$ sudo python setup.py install
```
Wiringpi package should be installed by now.

Download the controller project:
The controller project is a python project developed by XJ to performs all the power stablizing fuctionality of the system.
In terminal window enter the following commands:
```
$ git clone https://github.com/QuantumQuadrate/PowerStablizer
$ mv /home/pi/PowerStablizer/project1 /home/pi
```
These commands downloads the repository from Github and moves the /project1 folder under /home/pi. This can be checked by whether or not one can successfully enter the project1 folder:
```
$ cd /home/pi/project1
```

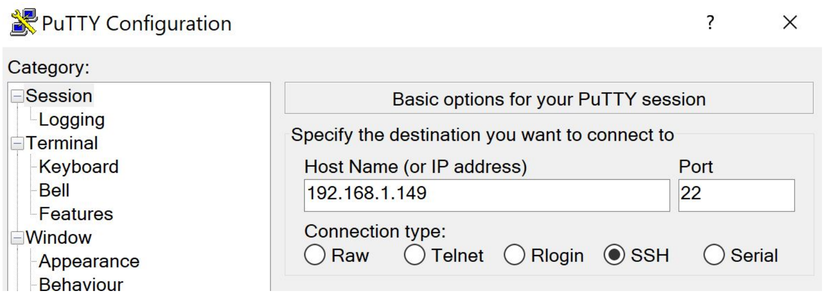
Bumping the system to version 4.49:
The wiringpi python package has compatiblity problems with newest version of the firmware kernel, so we need to bump it to a more stable version:
```
$ sudo apt-get install rpi update
$ sudo rpi-update 2ca627126e49c152beb1bf7abd7122ce076dcc65
```
Reboot the RPi after installation:
```
$ sudo reboot now
```
RPis can also be shut down remotely:
```
$ sudo shutdown now
```

2.2 Remote login via PuTTY

Putty is a open source SSH software that can enable access to RPis remotely. For now we are using it as the communication tool in our project.

PuTTY can be downloaded freely from: http://www.putty.org/.

To use PuTTY for remote login, first switch on the power of the RPi. Then in configuration window of PuTTY, enter the IP address of the target RPi. Then enter 22 as Port and choose SSH as connection type. Click Open to start the session.


PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.149

Port: 22

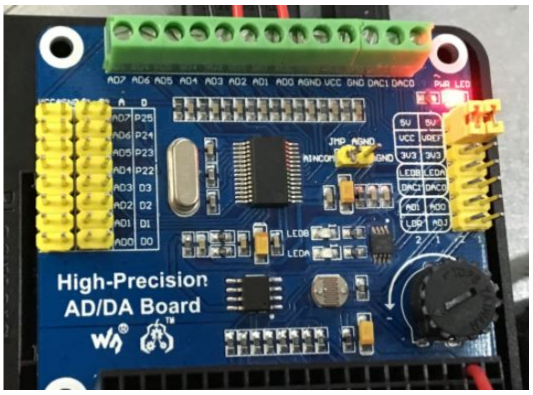
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

To login, user name and password are needed. A list of active RPis are being kept and updated on the wiki, so the username and passwords can be found in that document.

2.3 Set up the analog shield and the box

We use the Waveshare High Precision AD/DA Board as shield on the RPi to read analog inputs.

Before putting the shield into the box, remember to take off all the yellow jumpers on the shield with only two left: one across '5V' and 'Vcc', the other across '5V' and 'Vref'. The connection is shown as follow:



The circuit diagram and front panel design of the box can be found on the wiki. Assemble the box according to the circuit diagram.

2.4 Initialize the rotators

We are using the Thorlabs K10CR1 rotating mounts as the unit to control the intensity of the laser beam.

The introduction, control software, C-language API and communication protocols can be found online

Before using the rotators in the experiments, please install Kinesis(motion control software) on a PC, have the device connected, and set the backlash parameter of the device to zero.

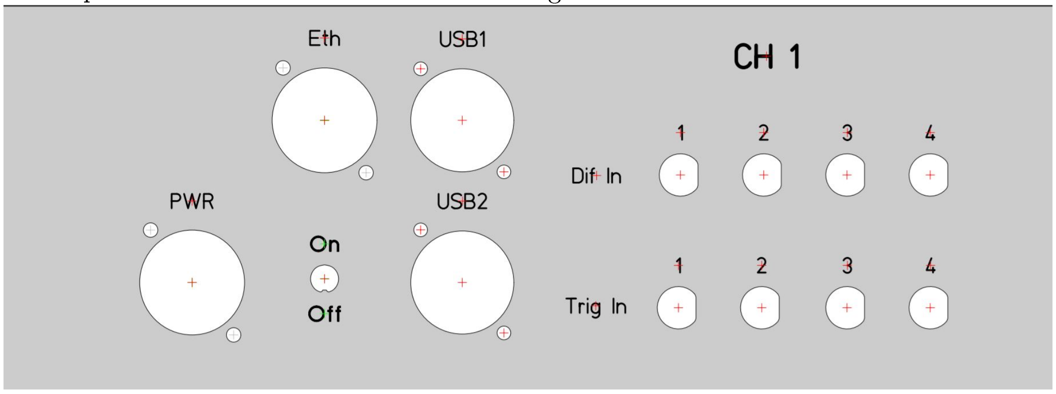
To achieve functionality of the system, a $1/2\lambda$ waveplate should be mounted in the rotator.

2.5 Optical setup

After 2.1-2.4 is done, one can start setting up the optical system as the basic diagram shows.

3 Introduction to the box

The front panel of each box looks like the following:



For each RPi, there are 2 USB ports, 4 analog inputs(Diff In), 4 digital inputs/outputs(Trig In), and 1 ethernet port on the box connected to it.

3.1 USB ports

There are 2 USB ports for each RPi in the box, labeled by 1 and 2. USB devices can be connected to the RPi through these ports. However the port name of the connected device inside the RPi is randomly assigned. One can check the port by entering:

```
$ ls /dev/ttyUSB*
```

Generally when connecting multiple K10CR1 rotators to the box, we normally don't connect them directly to the USB ports, for the RPi is not able to provide enough current to drive multiple motors. Instead, we should plug them into a self-powered USB hub, and then plug the hub into one of the USB ports. The number of rotators should not exceed 4, and the hub itself should be able to provide enough current for each K10CR1.

3.2 Input/Output

The 4 analog inputs('Diff In') are labeled as 1-4 on the box. All of them are connected to the analog shield(Waveshare High Precision AD/DA Board) on the RPi. Input i is connected to AD$2i-2$ and AD$2i-1$ on the shield as a differential input. Generally we are using these ports to read analog voltage inputs from the photodiode, which is proportional to the intensity of the laser beam.

The 4 digital input/ourputs('Trig In') are also labeled as 1-4 on the box. Each of the Diff In inputs are connected directly to two GPIO(general purpose input and output) pins on the RPi. The ground of the input signal are all connected to Pin 6 on the RPi, and the signals from port 1-4 are hooked to Pin 3,5,7,8 respectively. Generally we use these ports to receive the triggers from the experiment controller.

The 'Trig In' signals are independent of the 'Diff In' signals. One can have multiple rotators sharing the same trigger.

On Raspberry Pi, all GPIO banks are supplied from 3.3V. Connection of a GPIO to a voltage higher than 3.3V will likely destroy the GPIO block.

In principle the GPIO pins on a RPi are all digital ones, so we cannot send/receive analog signals directly to/from a RPi. As introduced before, an AD/DA shield can be used to realize these fuctions. It's also useful to note that the GPIO pins can be set to output a pulse-width modulated(PWM) signal, which can be converted to analog outputs with some external circuitry.

4 Running Stablization

Remote login to the RPi, then enter the following path:

```
$ cd /home/pi/project1
```

4.1 Start the program

Use command:

```
$ sudo python s3.0.py
```

4.2 Change stablization parameters

In the main program, a lot of parameters can be set to fit the system into the main experiment. Most of the parameters are arrays with length of 4. This is because we have 4 'Diff In' channels on the box panel, and each of those channels correspond to a particular laser beam and a specific rotator. The parameter for each laser-stablization system should be different, so we have many 4-component arrays to set up each channels individualy.

Use the following command to remote-edit the program and change parameters:

```
$ nano s3.0.py
```

The following parameters can be set:

(bool)**enable**: 1 for using,0 for not using the channel.Here the channels refer to the 'Diff In' ports on the box panel:
enable=[1,1,1,1]

(string)**SN**: Serial number for the rotator connected to each channel, if in use:
SN=['55000491','55000389','55000392','55000604']

(int/volts)**target**: Stablization target for the beam in each channel, if in use:
target=[3,3,3,3]

(float)**KP**: Proportional part of feedback parameters for each channel:
KP=[1,1,1,1]

(float)**KI**: Integral part of feedback parameters for each channel(cannot be used for now, the new version is still being tested):
KI=[0,0,0,0]

(int)**triglist**: For the specific 'Diff In' channel, specify which trig port it is using(please note that 0,1,2,3 in triglist corresponds to trigger channel 1,2,3,4 on the box):
triglist=[0,0,0,0]

(float/seconds)**delaytime**: Set the time delay after trigger before measurement for each channel:
delaytime=[0,0,0,0]

(float)**tintegral**: Set the time duration for measurements in each channel. Multiple measurements will be done within this time limit and the average will be used for feedback:
tintegral=[0.01,0.01,0.01,0.01]

(int)**integralwindow**: Set the length of the integral window in feedback calculation for each channel(cannot be used for now, the new version is still being tested):
integralwindow=[10,10,10,10]

(bool)**readonly**: 1 for only reading out the signal,0 for running stablization:
readonly=0

(bool)**enablebuffer**: 1 for enabling the input buffer(increase input inpedance),0 for disable: enablebuffer=0