October 16, 2025

Basic System Setup for Measurement

Charlie G. Tolley[1, 2]

[1] *Department of Astronomy, University of California Berkeley, Berkeley, CA 94720, USA*
[2] *Radio Astronomy Laboratory, University of California Berkeley, Berkeley, CA 94720, USA*

(Dated: Oct 1, 2025)

1. SETUP AND INSTRUCTIONS

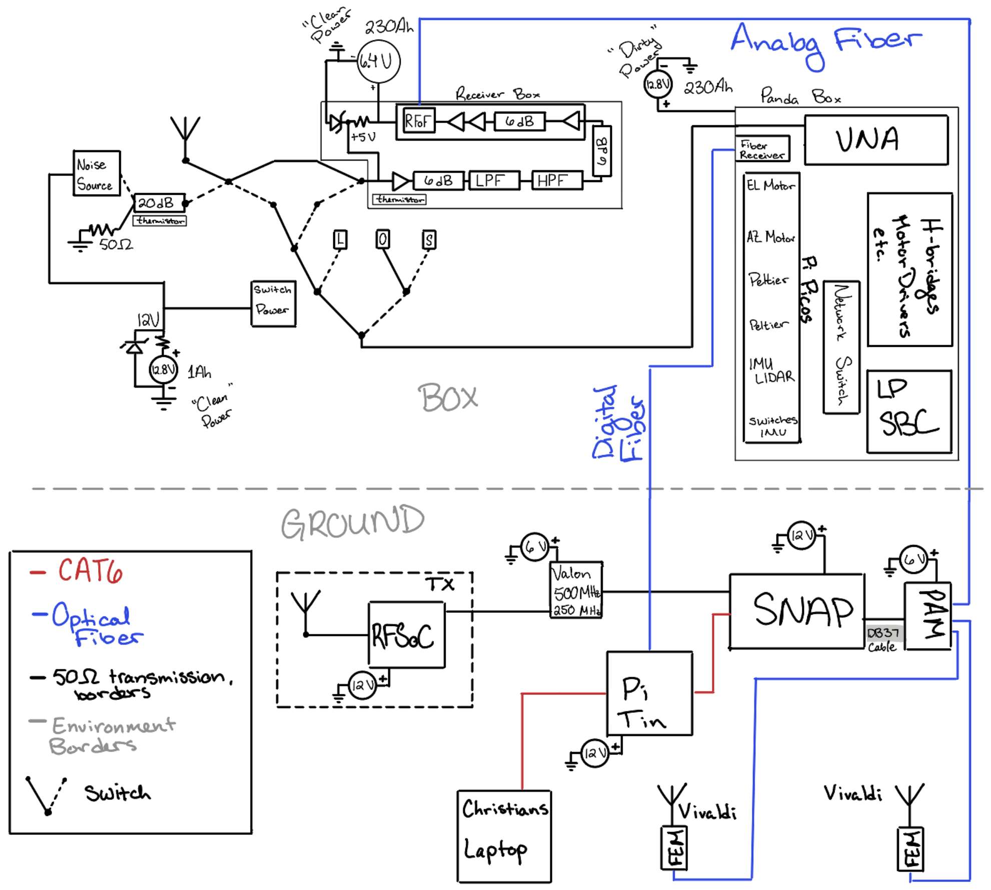

Figure 1. The configuration of the most basic components of the telescope. The only elements that are not included are the IMU, LIDAR, and Peltier devices (thermal control). Note that the DB9 cables are not pictured here - all cables in the box are labeled with their destination DB9 connector. This being said, the only cables needed for basic switching/calibration cycle within the box are the RF Switch DB9, power to the switches, noise source, receiver and panda box. Note also that the Vivaldi feeds are only included for the sake of completion, as they are only used for RFI flagging and unnecessary for the internal calibration cycle. Likewise, the RFSoC external transmitter (TX box) is not required and not setup in the lab currently.

Corresponding author: Charlie G. Tolley
tolley412@berkeley.edu

1.1. *Power*

In the lab, the systems shown can be powered by either battery or power supplies. Generally, I use power supplies for the backend (everything in the "GROUND" section), and batteries for the box. There are three battery packs that power the box:

1. 1Ah 12V clean power (attached to the roof of the box)
2. 230Ah 12V dirty power (the big battery with the N-type cable, has an auto-reset circuit breaker)
3. 230Ah 6V clean power (big battery with SMA cable, has an auto-reset circuit breaker)

The clean box components all have zener diodes connected to regulate voltage as our batteries are not truly 3V per cell, but closer to 3.2V per cell. To ensure that the voltage levels of the LNAs (which have voltage and temp dependencies) remain consistent between the power supplies and batteries, make sure to set the voltage levels accordingly.

The thermistors, which provide thermal monitoring, are powered through connection to the Pi Pico microcontrollers and do not require an external power source.

1.2. *Hardware*

You will have to be aware of the SNAP that you are using. In the current setup, the HERA number is `C000069`, and the other SNAP we have available is `C000122`. This has implications for which `corr_config.yaml` you use when initializing the FPGA.

For the PAMs, or Post-Amplification Module, if the optical fiber port is dirty then it will not be able to receive our analog signal. It doesn't matter which fiber input/SMA output pair you use.

To connect to the Pi tin (which talks to the LattePanda SBC), one can connect via wireless connection or a wired connection, which is shown in the figure above.

1.3. *Software*

In its current configuration, the Raspberry Pi has the IP address `10.10.10.10`, and the Panda can be ssh'ed into from the Pi at the IP address `10.10.10.11`.

The *internal* calibration cyle can be run manually as follows on a system set up as shown above:

1.3.1. *Terminal 1*

Initialize the FPGA.

```
ssh eigsep@10.10.10.10 (requires password)
python ~/eigsep/eigsep_observing/scripts/fpga_init.py -pasf
```

You may need to specify the config, as mentioned above. The default file is `corr_config.yaml` and has the correct information for SNAP `C000069`. If you need to switch to SNAP `C000122`, you will have to specify the correct config file with argument `--config_file ~/eigsep/eigsep_observing/src/eigsep_observing/config/corr_config_snap122.yaml`.

All you need control of are switches and VNA, but thermistor control is highly recommended.

1.3.2. *Terminal 2*

Live plotter for real time monitoring. Optional, but useful, and can be run from your local server.

```
python <path to eigsep_observing>/scripts/live_plotter.py --pairs 1
```

The `pairs` flag is optional, but without it, it will plot all the outputs, including cross-correlations, which is slow. As the system is currently set up, the data we care about is coming out of "pair" 1.

1.3.3. *Terminal 3*

Connect to the box.

```
ssh eigsep@10.10.10.10 (RPi, requires password)
ssh eigsep@10.10.10.11 (LattePanda, requires password)
```

1.3.4. *Terminal 4*

For capturing spectra, this can also be run from local server. The following command will produce an npz file. Can be run at any point.

```
python <path to eigsep_observing>/scripts/capture_spectrum.py <# of 1 sec integrations>
<filename>  --pairs 1
```

1.3.5. *Steps*

For gain calibration, you will want to capture spectra in Terminal 4 from the noise source (switch state `RFNON`) and the ambient load (switch state `RFNOFF`).

For S11 measurements, the following procedure can be implemented in a script or in IPython.

```
from PicoHost import PicoRFSwitch
from cmt_vna import VNA
sw = PicoRFSwitch("/dev/ttyACM2")
vna = VNA(switch_network=sw)
#for feed, load and noise source S11, power_dBm = 0
freqs = vna.setup(fstart=1e6, fstop=250e6, npoints=1000, ifbw=100, power_dBm=0)
vna.add_OSL() #will switch through OSL standards and add to the data attribute

s11s = measure_ant(measure_noise=True, measure_load=True) #dictionary of s11s
vna.data = vna.data | s11s
vna.write_data(<filename>)

#need to do setup again to get different power level for receiver
freqs = vna.setup(fstart=1e6, fstop=250e6, npoints=1000, ifbw=100, power_dBm=-40)
vna.add_OSL()
s11s = measure_rec() #dictionary of s11s
vna.data = vna.data | s11s
vna.write_data(<filename>)
```