HF Power Monitor Manual

Lucas McDonald

Table of Contents

Introduction . . . 1
Software Installation . . . 1
  Dependencies . . . 1
  Configuration (GUI only) . . . 2
GUI Operating Instructions . . . 2
  Main Window . . . 2
  Program Settings . . . 4
  Graph Window . . . 5
  Graph Settings . . . 6
Command Line Operating Instructions . . . 7
  Main Window . . . 7
Error Filtering . . . 9
Hardware Description . . . 9
  Parts . . . 9
  PWR-8GHS-RC Power Meter . . . 10
  HMC252 RF Switch . . . 10
  Component Block Diagram . . . 10
Hardware Construction . . . 11
  Chassis . . . 11
  Layout . . . 13
Client Software Code Description . . . 14
  GUI . . . 14
  Command Line . . . 14
  Shared Files . . . 15
  Processing Files . . . 15
Device Networking Description . . . 16
  Monitor/Record . . . 16
  Copy . . . 17
  Recorded Files . . . 18
Server Software Description . . . 18
  Starting the Server . . . 18
  Stopping the Server . . . 19
  Server Code . . . 19
Troubleshooting . . . 19
  Raspberry Pi Connection . . . 19
  Power Meter Connection . . . 19
  Monitoring / Recording Issues . . . 19

Introduction

The HF Power Monitoring system was developed during the 2017 Arecibo REU program. The system allows the user to hook up forward and reflected power inputs from each of the six HF transmitters. The software provides the user with GUI and Command Line interfaces to monitor the forward and reflected power of each transmitter.

Software Installation

Dependencies

The software was developed and tested using Python 3.6.1. Any Python version below 3 is not supported.

First, ensure the package manager is updated. Under Debian,

```
sudo apt-get update
sudo apt-get upgrade
```

The Command Line interface requires:

```
pickle
```

The GUI application requires:

```
tkinter
pickle
matplotlib
```

`tkinter` can be installed (under Debian) with `sudo apt-get install python-tk`.

`pickle` and `matplotlib` can be installed with `sudo pip3 install pickle matplotlib`.

Matplotlib may state it is unable to install `freetype` and `png`. If so, under Debian, type:

```
sudo apt-get install libpng-dev
sudo apt-get install libfreetype6-dev
```

Each package can be installed with pip3:

```
sudo pip3 install tkinter pickle matplotlib
```

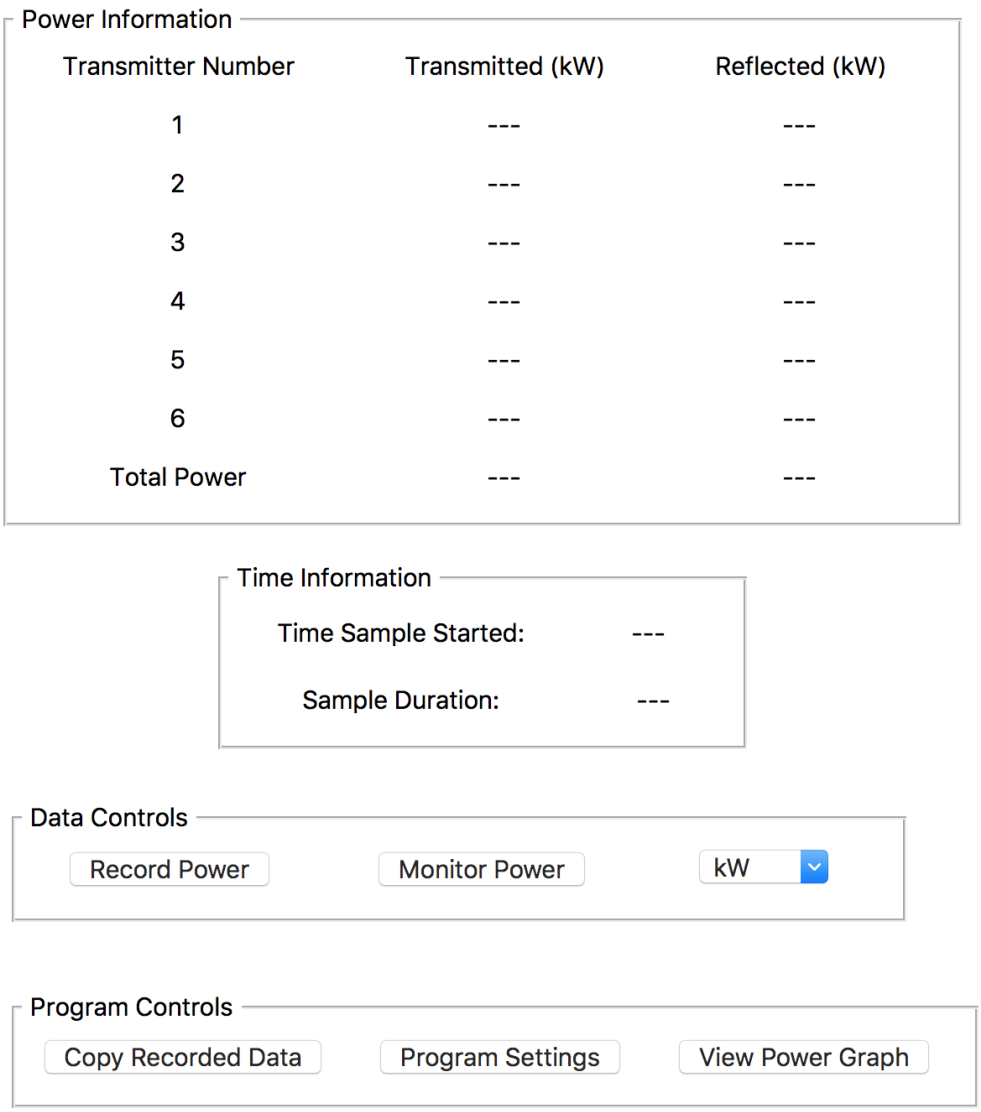
Configuration (GUI only)

Open the file "HF_Power_Monitor.sh" in a text editor. On the line with "cd", enter the absolute filepath of the "hfmon" folder after "cd". The "HF_Power_Monitor.sh" file can now be placed anywhere to make launching the GUI easy (e.g. on the Desktop for easy access).

GUI Operating Instructions

1. Ensure the rack-mount system is plugged in and all Ethernet ports are connected.
2. Turn on the rack-mount system and wait about 30 seconds for the internal devices to connect to the network.
3. Launch the "HF_Power_Monitor.sh" script. The main GUI window will appear.

Main Window



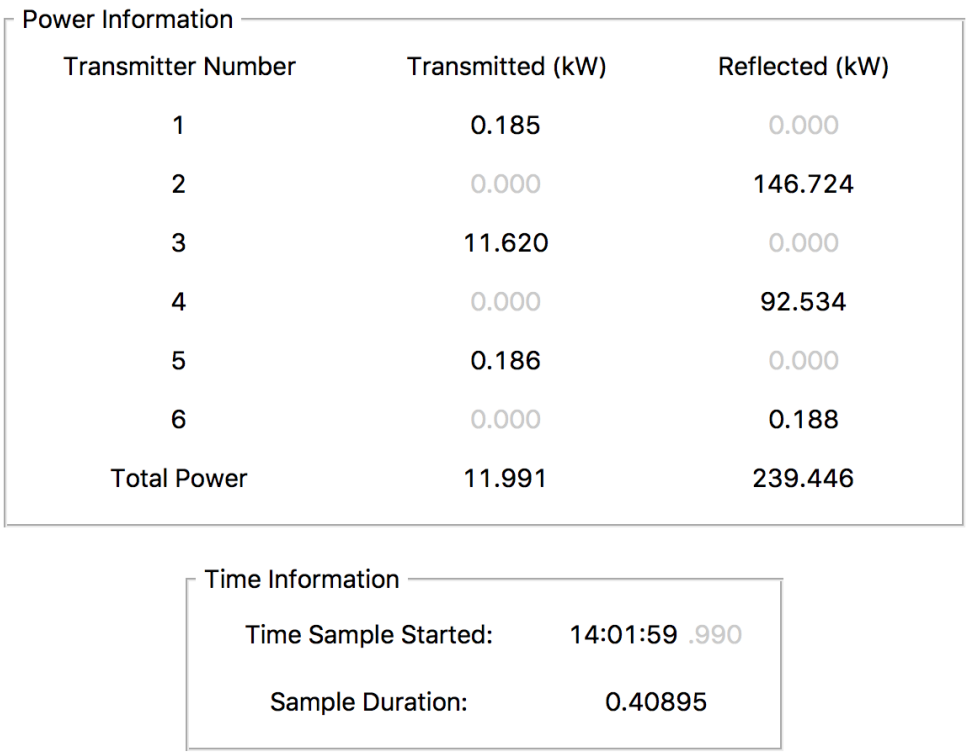
- Power Information: Displays the transmitted and reflected power readings from each of the six transmitters, as well as the total power for both transmitted and reflected readings.

- Time Information: Displays a timestamp for when the reading started as well as an indicator for how long the program took to sample all six transmitters.
- Data Controls: Controls for the data collection and power monitoring.
  - Record Power: Display power readings in the GUI and write the power data to a file on the Raspberry Pi.
  - Monitor Power: Display power readings in the GUI, but do not write the readings to a file.
  - Units dropdown: Select the units for display in the GUI and the graph window. Does not change the recorded units in the Raspberry Pi's data file; this is always in Watts.
- Program Controls: General controls for the program.
  - Copy Recorded Data: Copies the recorded data from the Raspberry Pi to a designated filepath on the client computer. The `~/hfmon/data/` folder will be copied to this location. The client must enter their sudo password and the Raspberry Pi's password ("raspberry" by default) in the shell window to complete this action.
  - Program Settings: Settings for the data collection program. Includes:
    - IPs of each power meter and the Raspberry Pi's IP
    - Whether the program should attempt to read from the transmitted or reflected power meter
    - Sampling period
  - View Power Graph: Opens a separate window that displays a graph of the power over time.

If this is your first time launching the program, you will get an error saying that "Default settings were loaded." Open Program Settings to ensure the settings (most importantly, the device IPs) are correct.

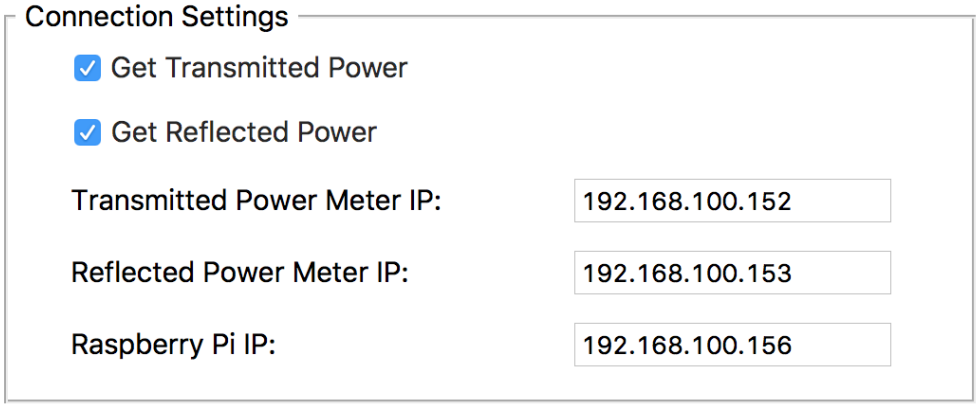
To start viewing power readings, click either the record or monitor buttons. While monitoring, you can set the displayed units, or open up the graph window. You can also do these while not monitoring.

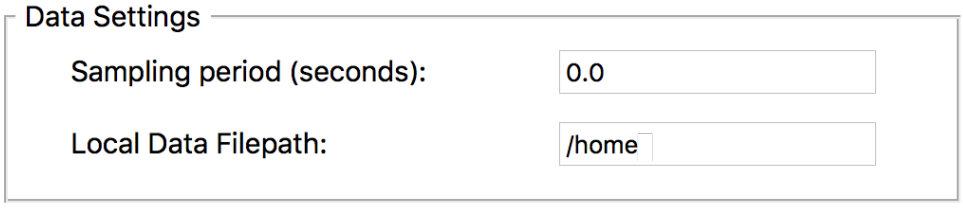
While monitoring, the display appears as:



If the values are gray, there is no power being read (0 W/kW or -99 dBm).

Program Settings





A sampling period lower than 0.5 seconds can lead to an inaccurate sample rate. Enter 0 to sample as quickly as possible.

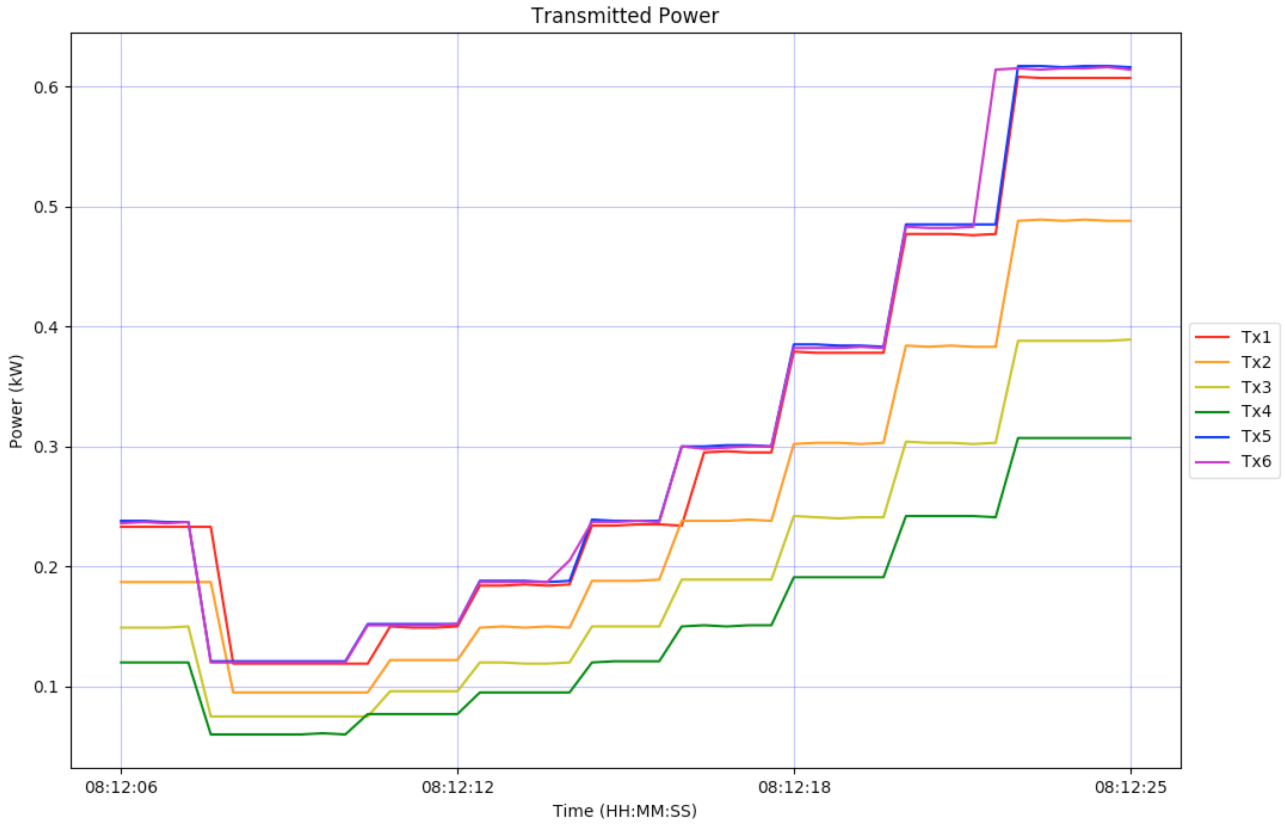
- Connection Settings

- Get Transmitted/Reflected Power: If this is enabled, the program will attempt to get readings from the corresponding power meter.
- Transmitted/Reflected/Raspberry Pi IP: Fields that hold the IPs of the power meters and the Raspberry Pi. The default IPs for Arecibo's main network should be:
  - Transmitted Power Meter IP: 192.168.100.153
  - Reflected Power Meter IP: 192.168.100.152
  - Raspberry Pi IP: 192.168.100.156

- Data Settings
  - Sampling Period: Specified amount of time between samples. Each sample will almost never take more than 500 seconds. Entering a value less than 0.5 seconds is allowed, but accurate sample rates cannot be guaranteed. To sample as quickly as possible, enter 0.
  - Local Data Filepath: When the Copy Recorded Data button is pressed, the "data" folder on the Raspberry Pi will be copied to this location.

These settings are shared between the GUI and Command Line interfaces. That is, editing a setting in the GUI will also edit the setting in Command Line.

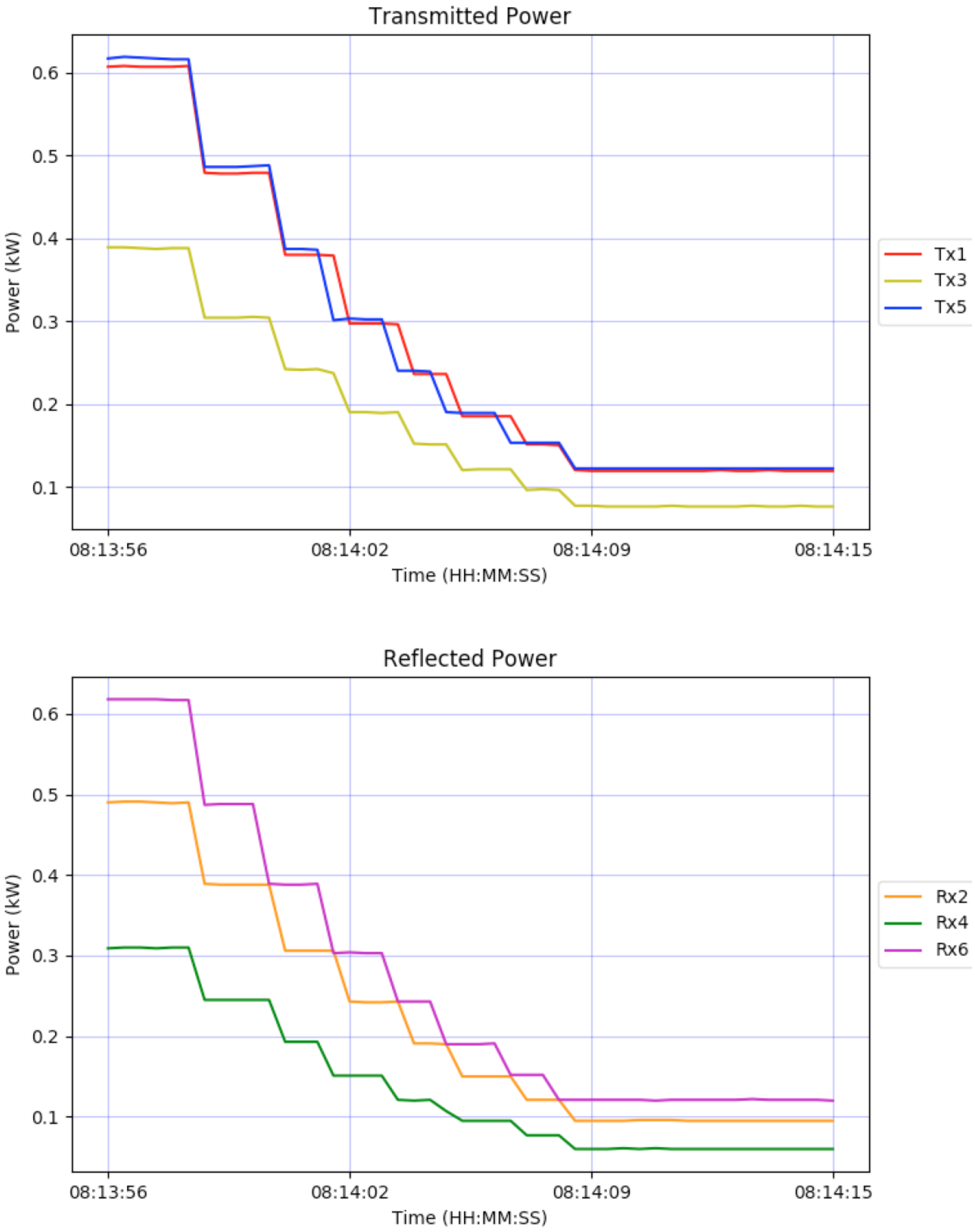
Graph Window



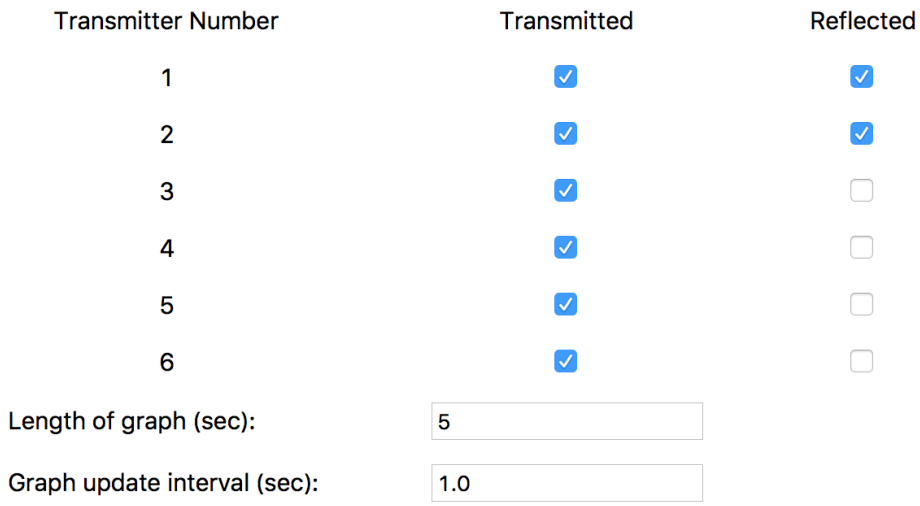
Along the x-axis, timestamps are shown. The length of the x-axis can be configured in Graph Settings. Along the y-axis is the power. The power units can be changed from the main window's units dropdown menu.

If any reflected plots are enabled in the Graph Settings menu, a smaller graph of the transmitted

power will be displayed along with the reflected power as shown below:



Graph Settings



The checkboxes indicate whether the plot of the specified transmitter and direction should be plotted.

In the above picture, each transmitted plot will be displayed on the transmitted power graph, while only reflected plots 1 and 2 will be displayed on the reflected power graph.

The "Length of graph" field indicates how long the x-axis should display values in seconds.

The "Graph update interval" field indicates how often the graph should refresh its view. A low value will update the graph view more quickly, but will be significantly more CPU intensive. A higher value will take longer to update the graph view, but will be less CPU intensive.

Command Line Operating Instructions

1. Ensure the rack-mount system is plugged in and all Ethernet ports are connected.
2. Turn on the rack-mount system and wait about 30 seconds for the internal devices to connect to the network.
3. Resize the terminal window to at least 160 characters wide.
4. Launch the command line interface by typing `python3 hf_terminal.py` from the `~/hfmon/python` folder.

The first few lines of the terminal will indicate the status of the connection to the Raspberry Pi. If successful, they will read:

```
Successfully connected to RPi at 192.168.100.156 on port 12345
```

Main Window

The following will appear:

```
**********************
** HF Power Monitor **

Commands:

    record [units]     Monitor and write data from each transmitter.
                            If units are ommitted, kW is used by default.
    monitor [units]    Monitor the power of each transmitter,
                            but don't write data.
    switch [t/r] [n]   Set the transmitted or reflected
                            switch to the nth input.
    settings           View and modify program settings.
    copy               Copy all data files from the Raspberry Pi
                            to this computer.
    quit               Quit the program.


Input a command:
```

- record [units]: Start displaying power readings in the temrminal window and writing those power readings to file on the Raspberry Pi. Allowed units are W, dBm, and kW. If no units are input, kW are used by default. Press Ctrl+C to stop recording.
- monitor [units]: Start displaying power readings in the terminal window, but don't write those readings to file. Allowed units are W, dBm, and kW. If no units are input, kW are used by default. Press Ctrl+C to stop monitoring.
- switch [t/r] [n]: Sets the HMC252 switch for the transmitted or reflected input to the nth input. For instance, typing `switch r 3` sets the switch for the reflected input to input 3.
- settings: Displays program settings:

  ```
  1) Get Transmitted Power: True
  2) Get Reflected Power: True
  3) Transmitted Power Meter IP: 192.168.100.153
  4) Reflected Power Meter IP: 192.168.100.152
  5) Raspberry Pi IP: 192.168.100.156
  6) Local filepath to store data: /Users/Lucas/Desktop
  7) Sampling Period: 1.0

  To modify a setting, type "settings change [num] [desired value]".
  ```

  These settings are shared between the GUI and Command Line interfaces. That is, editing a setting in the GUI will also edit the setting in Command Line. To edit a setting, type `settings change [num] [desired value]`. For instance, if I were to change the local filepath to my desktop, I would type `settings change 6 /Users/Lucas/Desktop`. The filepath must be absolute.

- copy: Copies the data file on the Raspberry Pi to the location specified in the local filepath. Requires the user to enter the administrator password and the Raspberry Pi's password

("raspberry" by default).

- quit: Ends the program.

Error Filtering

The file error_filter.py filters out common errors. It will filter out these errors:

- Reading a high power (~1.3 kW) when a low power (0 W) is input.

  (Currently disabled by being commented out. Uncomment this section in the file if you wish to use it.)

- Power meter timeout

and create a new file `[filename]_filtered.data`. It is not required to call this script, but it is provided if the user wishes to correct these errors from their datasets.

For instance, to filter data from July 16, 2017 in the file 20170716.data, type `python error_filter.py 20170716` to produce the file `20170716_filtered.data` in the `~/hfmon/data/filtered` folder. Or, to filter all files in the data folder, type `python error_filter.py`. This can be run with Python 2 or 3.

Hardware Description

Parts

Item	Description	Quantity	Part Number
1	HMC252 RF switch	2	HMC252
2	PWR-8GHS-RC power meter	2	PWR-8GHS-RC
3	Raspberry Pi Model B+	1	
4	Acopian 5V 2A power supply	1	5EB200
5	Terminal block	1	
6	AC Power input with fuse	1	
7	Lansing B style enclosure (14" depth, 2U)	1	B2F14-001A
8	SMA Cables	14	
9	LED / LED holder	1	
10	Power switch	1	
11	Mini-Circuits VAT-10+ 10dB attenuator	12	VAT-10+

12	L-Com CAT3 RJ45 coupler	3	ECF504-SC3
13	USB extender cables	2	
14	Micro USB cable	1	
15	Jump wires	10	
16	SMA Female-Female Mounts	12	
17	Spade connectors	4	
18	Metal mounting bracket	2	
19	Electrical Wire	As needed	

PWR-8GHS-RC Power Meter

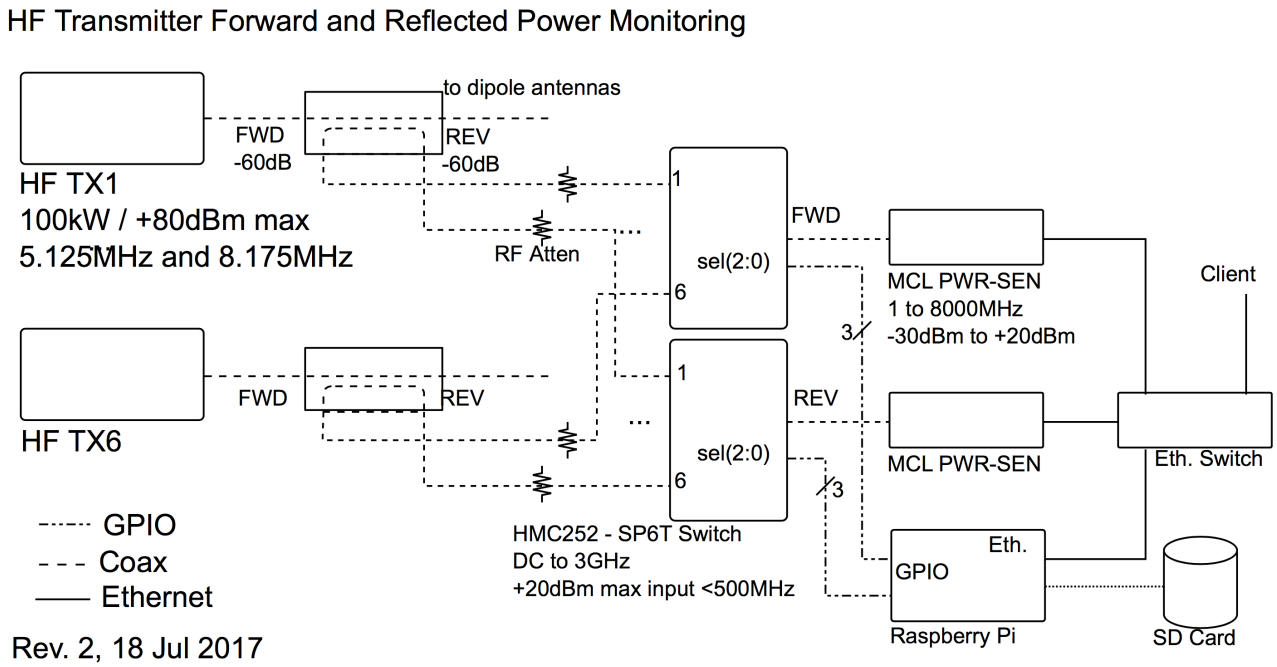
Characteristics

- Sampling Period: The power meters have a sampling period of 30 milliseconds for one sample. If requests made and received rapidly (as they are in this project), the sampling period sharply increases. Sampling at each power meter six times (once for each of the six transmitters) takes around 300 to 400 milliseconds.
- Measurement Uncertainty: The meters have a typical measurement uncertainty of .15 dBm, but a maximum uncertainty of around .4 dBm.
- Bandwidth: The power meters can receive input from 950 kHz to 8 GHz. Any signal lying within this range will contribute to the power meters' readings, while any signal lying outside this range will have absolutely no effect on the readings. Centering the frequency of the power meters does not attenuate signals outside a specified region or change the bandwidth of the power meters.
- Communication Interface: Communication with the power meters is described in the Monitor/Record section. Communication is done via an HTTP request that returns an HTML file containing the power.
- Maximum input: 20 dBm. HF transmits at 100 kW max, or 70 dBm max. The directional couplers have a loss of about 50 dB. Additional attenuators of 10 dB are located at each input, m

HMC252 RF Switch

- The switch can take around 100 nanoseconds to switch inputs. In the software, a small delay of 1 ms was added between switching the inputs and reading the input from the power meter. This drastically improves sampling accuracy while adding only 6 ms to the total sampling length, a number around 400 milliseconds.

Component Block Diagram



For each transmitter:

1. The transmitted signal is passed through the directional coupler, where the signal is coupled to the "forward power" output of the directional coupler at a loss of 60 dB.
2. The transmitted signal reaches the antenna, but part of the signal is reflected. The reflected signal is passed through the directional coupler and coupled to the "reflected power" output at a loss of 60 dB.
3. Both forward and reflected outputs are attenuated by 10 dB to not damage the HMC252 switch, which has a maximum input of 20 dBm.

Each forward input goes to one HMC252 switch, while each reflected input goes to the other switch.

When the Raspberry Pi is recording,

1. The Raspberry Pi outputs 3 control bits to the HMC252, which interprets the bits as a decoder. The HMC252 is a SP6T switch, so the control bits determine which of the six inputs is connected to the output of the switch.
2. The Raspberry Pi reads the power meter, which is connected to the output of the switch, getting a value for the power of the connected input. The Raspberry Pi stores this value.

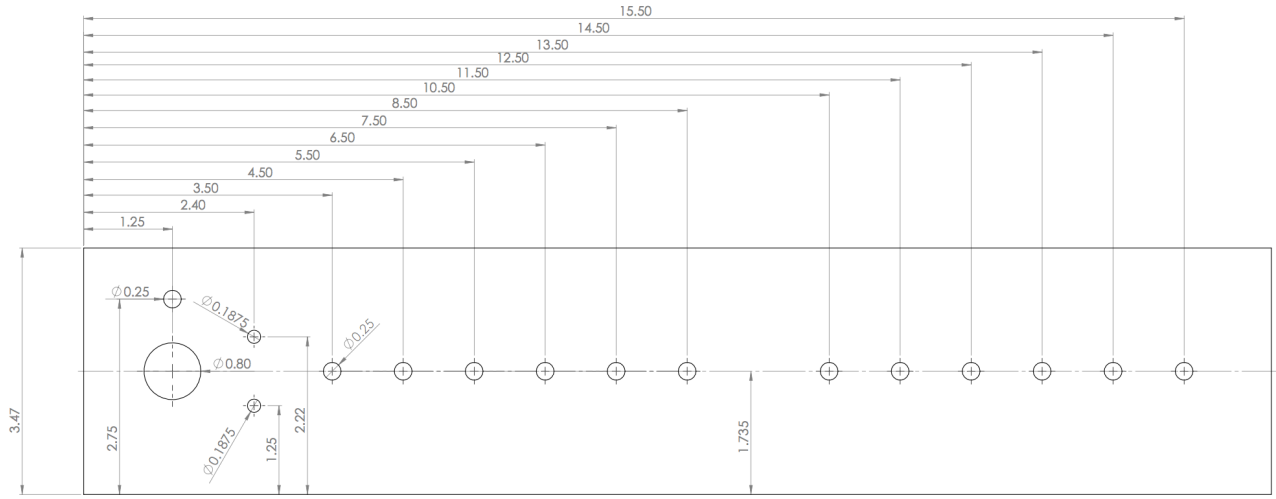
The Raspberry Pi sends six sets of control bits in total, one for each transmitter. Once each power is read, the power data is stored to the SD card and transmitted back to the client.

Hardware Construction

Chassis

A Lansing 2U chassis holds the hardware components of the project. The cuts for the chassis were

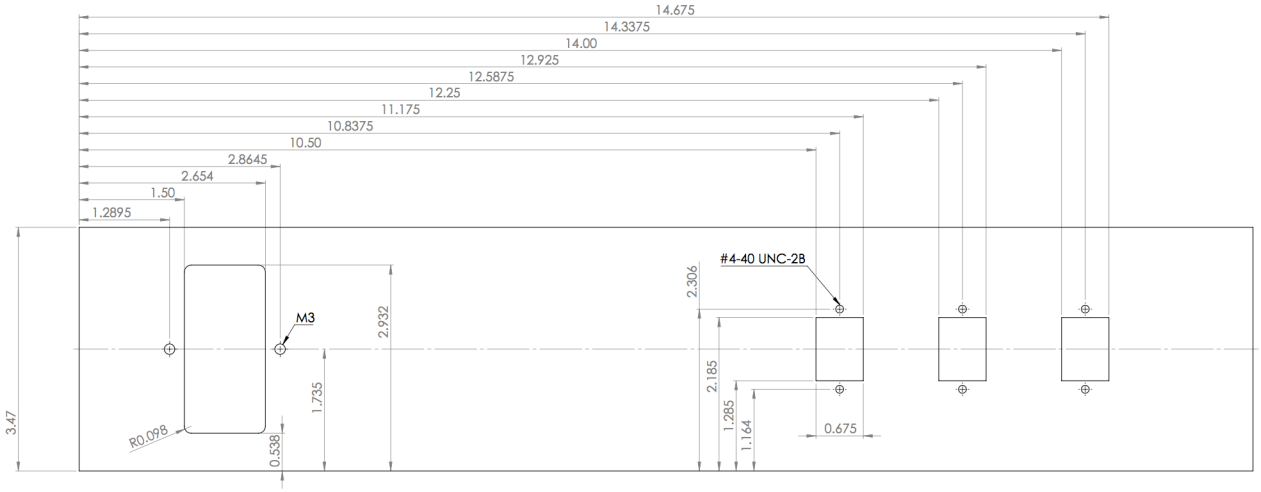
drawn in SolidWorks. The front panel was cut as:



From left to right, the cuts are for:

- The power switch (bottom) and power indicator LED (top)
- A 5V DC input (top) and a Ground input (bottom)
- Transmitted power inputs 1-6
- Reflected power inputs 1-6

The back of the chassis was cut as:

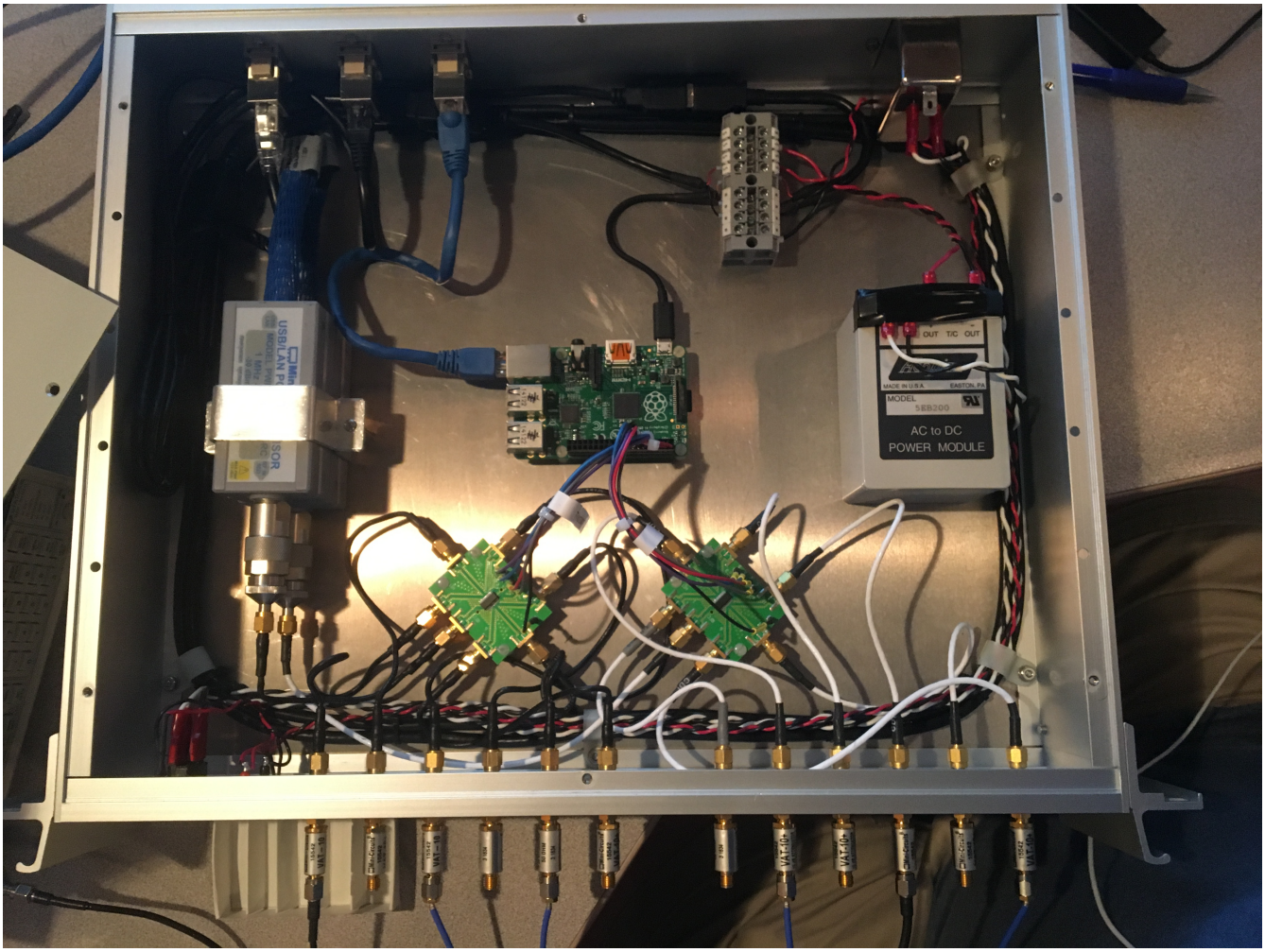


From left to right, the cuts are for:

- 120VAC input with fuse
- Raspberry Pi Ethernet Connection
- Transmitted Power Meter Ethernet Connection
- Reflected Power Meter Ethernet Connection

Layout

Inside of box:



Front of box:

Back of box:

Client Software Code Description

The software was developed and tested using Python 3.6.1. Any Python version below 3 is unsupported.

The code is contained in `~/hfmon/python`. All files are commented and can be viewed or edited.

There are two interfaces for the power monitoring system: a GUI and a command line application. Each interface has files specific to that interface, but both interfaces share a number of files. All files in the application will be described here.

GUI

The program for the GUI is split into multiple Python files. A short description of each follows:

- hf_gui_main.py: The file launched by the shell script. Creates a window displaying information about the power readouts from each transmitter and direction, as well as timestamps for each sample. Also contains buttons for controlling the GUI.
- hf_gui_settings.py: Window created by hf_gui_main.py when the "Program Settings" button is pressed. Contains settings for the program that the user has access to modify. These include:
  - IPs of each power meter and the Raspberry Pi's IP
  - Whether the program should attempt to read from the transmitted or reflected power meter
  - Sampling period
- hf_gui_graph.py: Window created by hf_gui_main.py when the "View Graph" button is pressed. Displays a matplotlib plot of the power over a specified amount of time.
- hf_gui_graph_settings.py: Window created by hf_gui_graph.py when the "Graph Settings" button is pressed. Displays options for displaying forward or reflected data from each transmitter, as well as a field to input the length of time displayed on the plot.

Command Line

- hf_terminal.py should be launched to use the command line interface. The command line interface is more lightweight than the GUI. It provides the same monitoring and recording features, as well as the "copy" command.

Shared Files

- take_data.py: File that contains various functions for interfacing with the power meters. Functions include sending as the HTTP request / receiving the HTML response, monitoring all power meters, and writing the power data to a file.
- set_switches.py: Contains one method, set_switch(num, pinA, pinB, pinC) for setting the input pins to switch to the transmitter specified by "num".
- gui_settings.pckl: A file containing the settings for the GUI and Command Line interfaces. hf_gui_settings.py provides direct access to modifying this from the GUI application, while "settings change [num] [value]" provides access to modifying this from the Command Line application. This file should not be modified outside of these applications.
- graph_settings.pckl: A file containing the settings for the graph window for the GUI application. Can be directly modified from the hf_gui_graph_settings.py window. This file should not be modified outside of these applications.

Processing Files

- error_filter.py: This script can be called with `error_filter.py [filename]`. It will create a new file (without modifying the original data) titled "[filename]_filtered.data in the ~/hfmon/data/filtered folder. It is not required to call this script, but it is provided if the user thinks it would make data analysis easier.
    - Reading a high power (~1.3 kW) when a low power (0 W) is input. Detected by checking if the previous value and the next value are both 0, but the current value is between 1.27 kW and 1.32 kW (most frequently occurring error values). If this is the case, the current value is set to 0.

  (Currently disabled by being commented out. Uncomment this section in the file if you wish to use it.)

    - Power meter timeout: Detected with ' ' (6 spaces). Filtered by replacing the value ' ' with the previous value.

An image of the filtering process is shown below. The left-most changed value is due to a spike in the power reading, while the right-most changed value is due to a power meter timeout.

Pre-filter:

```
04:43:53.551,000241,000000,000153,000000,000245,000000,000000,000195,000000,000122,000000,000246,0.43382
04:43:54.536,000241,000000,000154,000000,000244,000000,000000,000194,000000,000122,000000,000247,0.41426
04:43:55.521,000242,000000,000153,000000,000245,000000,000000,000194,000000,000122,000000,000247,0.45267
04:43:56.506,000242,000000,000153,000000,000245,      ,000000,000193,000000,000122,000000,000245,1.43332
04:43:57.943,000242,000000,000154,000000,000246,000000,000000,000192,000000,000122,000000,000247,0.58189
04:43:58.928,000242,000000,000154,000000,000245,000000,000000,000194,000000,000122,000000,000247,0.41564
04:43:59.913,000239,001314,000154,000000,000247,000000,000000,000193,000000,000122,000000,000247,0.47385
04:44:00.899,000243,000000,000153,000000,000245,000000,000000,000194,000000,000121,000000,000248,0.45479
04:44:01.884,000243,000000,000154,000000,000244,000000,000000,000195,000000,000123,000000,000247,0.45203
04:44:02.869,000240,000000,000152,000000,000245,000000,000000,000192,000000,000121,000000,000247,0.41450
04:44:03.855,000241,000000,000154,000000,000244,000000,000000,000194,000000,000122,000000,000247,0.42889
```

Post-filter:

```
04:43:53.551,000241,000000,000153,000000,000245,000000,000000,000195,000000,000122,000000,000246,0.43382
04:43:54.536,000241,000000,000154,000000,000244,000000,000000,000194,000000,000122,000000,000247,0.41426
04:43:55.521,000242,000000,000153,000000,000245,000000,000000,000194,000000,000122,000000,000247,0.45267
04:43:56.506,000242,000000,000153,000000,000245,000000,000000,000193,000000,000122,000000,000245,1.43332
04:43:57.943,000242,000000,000154,000000,000246,000000,000000,000192,000000,000122,000000,000247,0.58189
04:43:58.928,000242,000000,000154,000000,000245,000000,000000,000194,000000,000122,000000,000247,0.41564
04:43:59.913,000239,000000,000154,000000,000247,000000,000000,000193,000000,000122,000000,000247,0.47385
04:44:00.899,000243,000000,000153,000000,000245,000000,000000,000194,000000,000121,000000,000248,0.45479
04:44:01.884,000243,000000,000154,000000,000244,000000,000000,000195,000000,000123,000000,000247,0.45203
04:44:02.869,000240,000000,000152,000000,000245,000000,000000,000192,000000,000121,000000,000247,0.41450
04:44:03.855,000241,000000,000154,000000,000244,000000,000000,000194,000000,000122,000000,000247,0.42889
```

Device Networking Description

This section describes how the commands in each interface operate within each device and between devices for the program to function properly.

Monitor/Record

The following describes what happens at each device when the client calls the "monitor" or the "record" command.

1. The user clicks a button on the GUI or inputs a command to the terminal that requires sending a monitor command to the Raspberry Pi.
2. The monitor command is put into an array along with arguments the Raspberry Pi needs to execute the command, such as the IPs of the power meters. The full contents of the array are:

   ```
   [command, power meter timeout, sampling period, pins used by the decoders, IP of
   the forward power meter, IP of the reflected power meter, boolean to get the
   transmitted power, boolean to get the reflected power]
   ```

   The array is encoded into a bitstring file using the "pickle" module. The client connects to the socket at the Raspberry Pi's IP on port 12345. The binary file is sent to the Raspberry Pi via the socket.

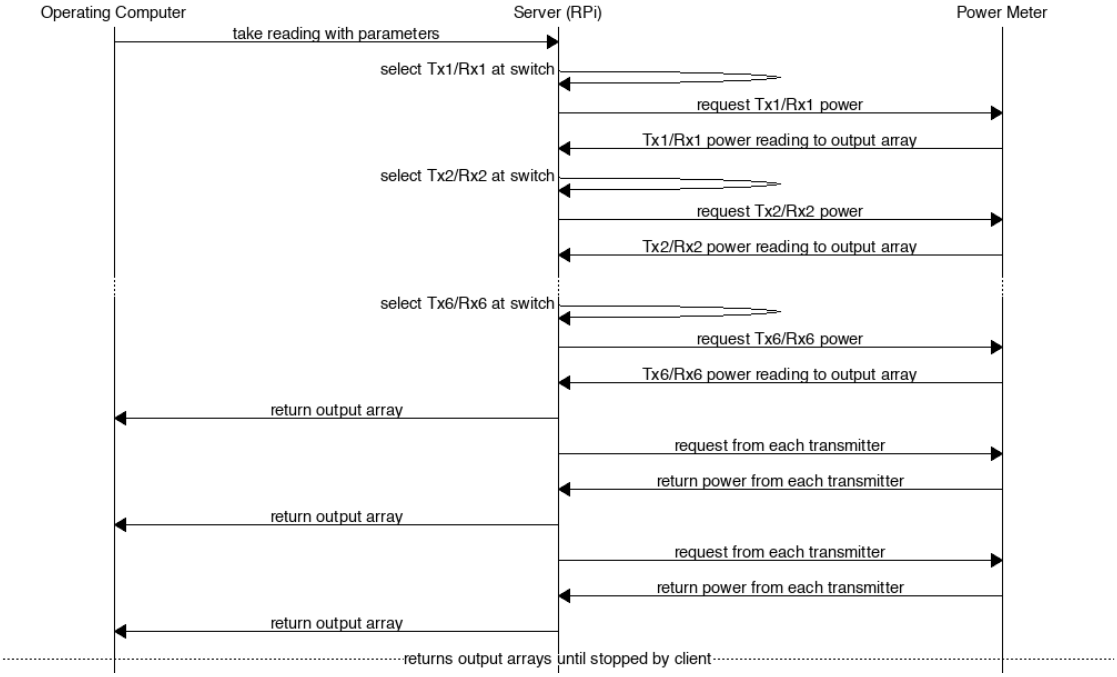
3. The Raspberry Pi listens on the socket and receives the text file. It "unpickles" the binary text file, extracting the array from it. The Raspberry Pi reads the command and begins to take power readings.
4. The Raspberry Pi begins to record forward and reflected power simultaneously in separate threads. The thread begins by setting the HMC252 to switch to the first input. It uses the Raspberry Pi's GPIO pins to set the corresponding switch using decoder bits.
5. The thread then sends a Telnet request to the corresponding power meter, which responds to the Telnet session with the power in dBm encoded as a bitstring. The thread converts the bitstring to a float. It also adds a correction factor to account for the losses from the directional couplers (around 60 dB). This is done for each of the six inputs. The thread then returns an array back to the main thread.
6. Once both threads have returned their arrays, they are appended to an output array. The contents of the output array are:

```
[time sampling started, Tx1, Tx2, ..., Tx6, Rx1, Rx2, ..., Rx6, sample duration]
(a total of 14 elements)
```

This array is encoded to binary using pickle and sent back to the client.

7. The client unpickles the file and updates the UI elements accordingly to display the contents of the output array.
8. Steps 4-7 are repeated until the client stops listening (stops monitoring or recording). This means that the client does not continue to send an input array, but the Raspberry Pi continues to send an output array.
9. When the client stops listening / monitoring, the Raspberry Pi closes the connection but continues to operate as a server to handle future monitoring requests.

A message sequence chart of this process is below:



Copy

The copy commands are done using an scp call to copy files from the Raspberry Pi to the local device. More exactly,

```
sudo scp -r pi@[rpi_ip]:/home/pi/hfmon/data [local_filepath]
```

Where "rpi_ip" is the Raspberry Pi's IP and "local_filepath" is the desired location to copy the data folder. Both are defined either in the Terminal or GUI application. For instance, if the desired location is "/Users/Lucas/Desktop", and I'm using the default Raspberry Pi IP, the program would call:

```
sudo scp -r pi@192.168.100.156:/home/pi/hfmon/data /Users/Lucas/Desktop/
```

and the data folder would appear on the Desktop with all of the data from the device. If using the GUI, the user may need to enter the root password and the Raspberry Pi password ("raspberry") in the terminal window behind the GUI to perform this action.

Recorded Files

Files are recorded with the filename YYYYMMDD.data. If today is July 16, 2017, the filename would be `20170716.data`. Files are recorded in `~/hfmon/data/`.

A sample of the beginning of a file:

```
#HF transmitted power data for 2017-07-16
#HH:MM:SS.SSS, Tx1 , Rx1  , Tx2  , Rx2  , Tx3  , Rx3  , Tx4  , Rx4  , Tx5  , Rx5  , Tx6  , Rx6  , time to take sample

00:00:00.726,000242,000000,000154,000000,000246,000000,000000,000194,000000,000122,000000,000245,0.43886
00:00:01.711,000241,000000,000153,000000,000244,000000,000000,000194,000000,000122,000000,000249,0.43345
00:00:02.697,000241,000000,000153,000000,000246,000000,000000,000192,000000,000122,000000,000247,0.41688
00:00:03.682,000240,000000,000154,000000,000245,000000,000000,000193,000000,000121,000000,000246,0.45174
00:00:04.667,000241,000000,000154,000000,000244,000000,000000,000194,000000,000121,000000,000246,0.43662
00:00:05.652,000242,000000,000153,000000,000245,000000,000000,000193,000000,000123,000000,000247,0.44159
00:00:06.638,000243,000000,000154,000000,000245,000000,000000,000194,000000,000121,000000,000247,0.43719
00:00:07.623,000241,000000,000154,000000,000245,000000,000000,000193,000000,000122,000000,000245,0.44114
00:00:08.608,000243,000000,000154,000000,000245,000000,000000,000194,000000,000122,000000,000245,0.43314
00:00:09.593,000243,000000,000154,000000,000245,000000,000000,000193,000000,000123,000000,000246,0.43079
00:00:10.578,000245,000000,000155,000000,000245,000000,000000,000194,000000,000121,000000,000247,0.43562
00:00:11.564,000241,000000,000154,000000,000243,000000,000000,000196,000000,000122,000000,000247,0.43019
00:00:12.549,000242,000000,000152,000000,000246,000000,000000,000193,000000,000123,000000,000245,0.43545
00:00:13.534,000241,000000,000153,000000,000245,000000,000000,000196,000000,000123,000000,000246,0.44111
00:00:14.519,000242,000000,000154,000000,000245,000000,000000,000194,000000,000124,000000,000246,0.46593
00:00:15.504,000241,000000,000154,000000,000244,000000,000000,000193,000000,000122,000000,000246,0.42054
```

Files are recorded as ASCII characters in Watts and are zero-extended to fill six values to make visual inspection of the data easier. The file header indicates the date and what each column represents.

An ASCII representation of each line:

```
HH:MM:SS.SSS, Tx1  , Tx2  , Tx3  , Tx4  , Tx5  , Tx6  , Rx1  , Rx2  , Rx3  , Rx4  ,
Rx5  , Rx6  , sample\n
```

Each line writes 105 bytes to the file. If sampled as quickly as possible (around 180 milliseconds), an hour of sampling will write 2.1 MB and a day of sampling will write 50.4 MB.

Server Software Description

Starting the Server

On startup, the Raspberry Pi calls `sudo python3 /home/pi/hfmon/python/hf_server_socket.py`. This starts the server software as soon as the Raspberry Pi starts up so the user does not have to manually restart the server. This startup call is configured in /etc/profile.

Stopping the Server

To disable the automatic startup call for debugging purposes, first type `sudo nano /etc/profile`, then comment out the line that calls the script (at the bottom of the profile file) with a #.

Server Code

The server code is in the file `hf_server_socket.py` and is also commented in the file.

The server is based on Python's socketserver class. It handles requests to the Pi, setting the switches, reading from the power meters, and returning the values. Only one client can connect to the server at a time to avoid conflicts in setting the switches.

If a client disconnects from the server, the server handles closing the session with the client and continues to listen for another client.

Troubleshooting

Raspberry Pi Connection

If there is an error or modifications need to be made to the server code on the Raspberry Pi, the first thing to do is to stop the software from running on startup if you wish to view errors in a terminal window.

- Connection Timeout: The Raspberry Pi is not connected to the network. Ensure that it is powered on and connected to the network.
- Connection Refused: The Raspberry Pi is powered on and connected to the network, but the server software is not running. Ensure the server software (hf_server_socket.py) is running on the Raspberry Pi without errors.

Power Meter Connection

- Unable to connect to power meters via Telnet, but devices are online via ping: Cycle the power on the chassis.

Monitoring / Recording Issues

- When hooked up to the HF coupler outputs and "Disable RF" is checked, power values fluctuate from 0 to ~1.3 kW: No current solution.
- Sampling period increases to ~6 seconds and power fields are blank: The corresponding power meter has disconnected from the network. Possible solutions:
  - Investigate and fix the connection issue. Ensure the power meter is connected to the network and is properly powered. Also ensure the IP is correct in settings.
  - Stop monitoring, open Program Settings, and disable the affected power meter. The

Raspberry Pi will not attempt to read values from this power meter.

- Power reading consistently takes longer than normal: The power meters need to be set up in "Fastest" sampling mode. The server attempts to perform this setting on startup, but the call has a small chance of failure. Cycle the power on the chassis to fix this.
- Power recording is empty: This value in the recordings indicates the corresponding power meter timed out and the power could not be read. This occurs somewhat often (~25 per 24 hours of recording). There are six spaces in this error indicator. Run the script `error_filter.py [filename]` to filter out these errors.