

Wireless Spectrum Analysis Suite: User Manual

Subham Saha

July 17, 2019

Contents

1	Dependencies	2
1.1	UHD	2
1.2	PyQt	3
1.3	matplotlib	3
1.4	pyserial	3
2	Program Files	4
3	Getting Started	7
4	Miscellaneous	8
4.1	Read data from binary file	8
4.2	USRP Benchmarking	8

Chapter 1

Dependencies

Here we describe the libraries used to develop the application and how to install the libraries in python. The version of python used in the application is **Python 2.7.15+**. These dependencies have to be installed prior to running the application.

1.1 UHD

UHD is required to interface with the USRP. The easy way to install this is using PyBombs. This installs UHD along with GNU Radio and other dependencies. To install we need python packet manager first. Below shows the steps of installation in the form of shell commands.

```
$ sudo apt-get install python-pip
$ sudo pip install pybombs
$ pybombs recipes add gr-recipes git+https://github.com/gnuradio/gr-recipes.git
$ mkdir prefix/
$ pybombs prefix init -a default prefix/default/ -R gnuradio-default
```

We need to setup the environment variables while using UHD as following

```
$ cd prefix/default
$ source ./setup_env.sh
```

The same terminal has to be used to run the codes. A different terminal will not have the environment variables available.

1.2 PyQt

The version of PyQt library used here is PyQt5. To install the PyQt5, following steps are to be performed:

```
$ sudo apt-get install python-pyqt5
$ sudo apt-get install pyqt5-dev-tools
```

To install QtWebEngine for the map view following steps are performed:

```
$ sudo apt-get install python-pyqt5.qtwebengine
```

1.3 matplotlib

matplotlib is the plotting library used in the application. To install it following is the simplest way:

```
$ python -m pip install -U matplotlib
```

1.4 pyserial

PySerial is the library used to interface with GPS receiver. Following is the step to install it

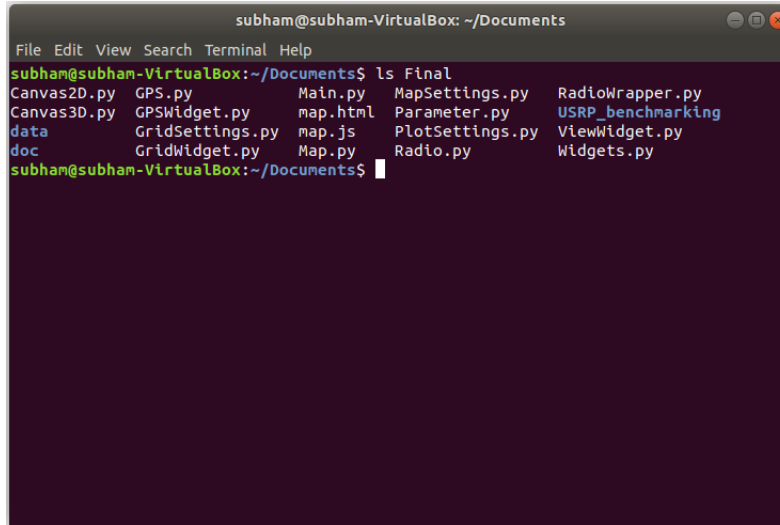
```
$ python -m pip install pyserial
```

Chapter 2

Program Files

After completing installation of all dependencies, the code file can be cloned into host system. The folder has three directories 1) data 2) USRP_benchmarking 3) doc, 14 *.py* files, 1 *.html* file and 1 *.js* file. Figure 2.1 shows the files listed in terminal. The entire program is broken in small python files. These are:

1. *Parameter.py* - All the parameters are kept in the *Parameter.py* file. This has the USRP parameter dictionary which is shared to initialize every view. Also this contains the parameters required for *Widgets.py* file to set the initial conditions.
2. *Widgets.py* - all low level widgets are implemented in this file. These widgets are assembled as settings for each view
3. *Map.py* - implements the Mapview widget. This widget needs two additional files. One is *map.html* and other one is *map.js* to provide required functionality.
4. *Canvas2D.py* - implements the two dimensional plotting canvas required for real-time plot.
5. *Canvas3D.py* - implements three dimensional plotting canvas for grid view.
6. *Radio.py* - implements the interface methods to connect to USRP.
7. *RadioWrapper.py* - wraps the USRP object with QObject class to provide support for signals and slots.

A terminal window titled 'subham@subham-VirtualBox: ~/Documents' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'ls Final' and its output, which lists files in a grid-like format. The files are: Canvas2D.py, GPS.py, Main.py, MapSettings.py, RadioWrapper.py, Canvas3D.py, GPSWidget.py, map.html, Parameter.py, USRP_benchmarking, data, GridSettings.py, map.js, PlotSettings.py, ViewWidget.py, doc, GridWidget.py, Map.py, Radio.py, and Widgets.py. The prompt 'subham@subham-VirtualBox:~/Documents\$' is visible at the bottom.

```
subham@subham-VirtualBox:~/Documents$ ls Final
Canvas2D.py  GPS.py      Main.py     MapSettings.py  RadioWrapper.py
Canvas3D.py  GPSWidget.py map.html    Parameter.py    USRP_benchmarking
data         GridSettings.py map.js      PlotSettings.py ViewWidget.py
doc          GridWidget.py  Map.py     Radio.py        Widgets.py
subham@subham-VirtualBox:~/Documents$
```

Figure 2.1: Files in *Final* folder

8. *GPS.py* - implements the interface methods for the GPS receiver.
9. *PlotSettings.py* - groups the necessary widgets and provide control to the real-time spectrum plot.
10. *GridSettings.py* - groups the necessary widgets and provide control to the grid view.
11. *MapSettings.py* - groups the necessary widgets and provide control to the map view.
12. *ViewWidget.py* - implements the View tab for real-time spectral plot. This is used in the tab widget of main window.
13. *GridWidget.py* - implements the View tab for real-time spectral plot. This is used in the tab widget of main window.
14. *GPSWidget.py* - implements the View tab for real-time spectral plot. This is used in the tab widget of main window.
15. *Main.py* - implements the main window.

Figure 2.2 shows the hierarchy in flow-like structure. The arrow points to the file where the arrow starting files are imported. Each widgets communicate

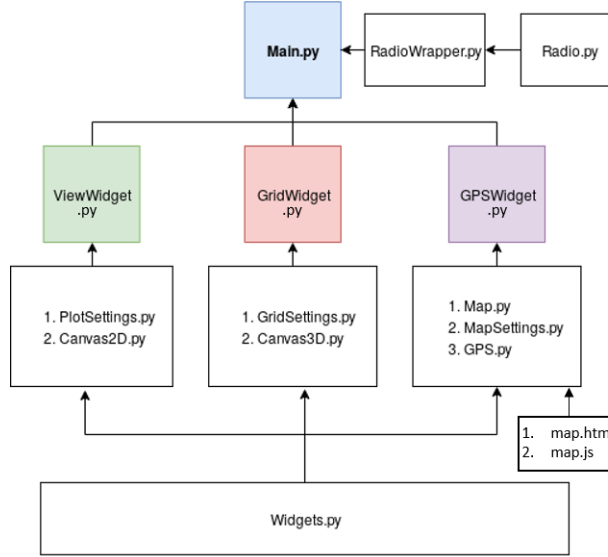


Figure 2.2: File hierarchy

with signals slots. The USRP device object is shared in all top level widgets through the main window. The *Main.py* implements the main window.

The captured data is saved in *data* directory as

- static - saves the captured data from view tab in filename *capture.bin*
- grid - saves the captured data from grid tab in format as *x_ (xposition)y_ (yposition).bin*.
- geo - saves the captured data from GPS tab in format as *lat_ (latitude)lng_ (longitude).bin*.

The `USRP_benchmarking` folder contains the programs to find maximum sampling rate that can be sustained in the host system. These are described in 4.2.

The `doc` folder contains the documentation and user manual.

Chapter 3

Getting Started

After installing the dependencies, to start the widget first UHD environment variables are loaded. Then we navigate to the folder where *Main.py* is present. *Main.py* file is executed to start the application. Below shows the steps to add UHD environment variables:

```
$ cd prefix/default
$ source setup_env.sh
$ cd
```

The last command navigates back to the home folder. The GPS receiver connects through serial port which requires administrator permission. To give the permission to the serial port following command is used:

```
$ sudo chown username /dev/ttyUSB0
```

Now to start the application, in the same terminal navigate to the code folder and run the *Main.py* file as:

```
$ cd Documents/Final
$ python Main.py
```


Chapter 4

Miscellaneous

4.1 Read data from binary file

Reading binary data in python is very easy. To do so, numpy library is used. Following is a code snippet for this purpose:

```
import numpy as np
data = np.fromfile('capture.bin',dtype=np.complex64)
```

4.2 USRP Benchmarking

Two benchmarking codes were written to find out maximum sampling rate sustainable for the host system. The *max_rate_available.cpp* checks the safest sampling rate. To iterate over targetted sampling rates *minimum_rate*, *maximum_rate*, *rate_step* and *duration* parameters can be adjusted. The *max_samples_available.cpp* checks the number of samples before overflow can occur. The *rate* parameter can be changed to change the sample rate of inspection.

These two codes have to be built. To build each code, in *CMakeLists.txt* change the lines as mentioned here

```
add_executable(filename,filename.cpp)
target_link_libraries(filename ...)
```

To build the file, navigate to the code file in terminal. From there use

```
$ cd build
$ cmake ../
$ make
```

To run the executable use

```
$ ./filename
```

Figure 4.1 shows the build of *max_samples_available.cpp* file in terminal window.

```
subham@subham-VirtualBox: ~/Documents/Final-001/USRP_benchmarking/build
File Edit View Search Terminal Help
subhangsubhan-VirtualBox:~/Documents/Final-001/USRP_benchmarking$ mkdir build
subhangsubhan-VirtualBox:~/Documents/Final-001/USRP_benchmarking$ cd build
subhangsubhan-VirtualBox:~/Documents/Final-001/USRP_benchmarking/build$ cmake ../
-- The C compiler identification is GNU 7.4.0
-- The CXX compiler identification is GNU 7.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Boost version: 1.65.1
-- Found the following Boost libraries:
--   program_options
--   system
--   thread
--   chrono
--   date_time
--   atomic
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Found UHD: /usr/local/lib/libuhd.so (Required is at least version "3.8.0")
-- *****
-- * NOTE: When building your own app, you probably need all kinds of different
-- * compiler flags. This is just an example, so it's unlikely these settings
-- * exactly match what you require. Make sure to double-check compiler and
-- * linker flags to make sure your specific requirements are included.
-- *****
-- Linking against shared UHD library.
-- Configuring done
-- Generating done
-- Build files have been written to: /home/subham/Documents/Final-001/USRP_benchmarking/build

subham@subham-VirtualBox: ~/Documents/Final-001/USRP_benchmarking/build
File Edit View Search Terminal Help
-- The CXX compiler identification is GNU 7.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Boost version: 1.65.1
-- Found the following Boost libraries:
--   program_options
--   system
--   thread
--   chrono
--   date_time
--   atomic
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Found UHD: /usr/local/lib/libuhd.so (Required is at least version "3.8.0")
-- *****
-- * NOTE: When building your own app, you probably need all kinds of different
-- * compiler flags. This is just an example, so it's unlikely these settings
-- * exactly match what you require. Make sure to double-check compiler and
-- * linker flags to make sure your specific requirements are included.
-- *****
-- Linking against shared UHD library.
-- Configuring done
-- Generating done
-- Build files have been written to: /home/subham/Documents/Final-001/USRP_benchmarking/build
subhangsubhan-VirtualBox:~/Documents/Final-001/USRP_benchmarking/build$ make
Scanning dependencies of target max_samples_available
[ 50%] Building CXX object cmakeFiles/max_samples_available.dir/max_samples_available.cpp.o
[100%] Linking CXX executable max_samples_available
subham@subham-VirtualBox:~/Documents/Final-001/USRP_benchmarking/build$ ./max_samples_availablemax_samples
```

Figure 4.1: Screenshots of building the files