2 *Installing the Software*

2.1 Installation

Remember the hour or two of slogging through software installation I promised (or warned) you about in the introduction? Well, it's here. Unless you happen to be a hardcore Linux guru, this will likely be your least favorite part of the book. However, if you follow the instructions carefully, you should be through the installation wilderness before you know it.

First, you need to be aware of the basic software tools out there for SDRs. We'll be working primarily in gnuradio, so installing this is a must.

Before I proceed, a word about operating systems. Two words, really.

"Use Linux."

It is definitely possible to install gnuradio on OS X and Windows. I've done it and made them work. It was not fun nor was it easy, and with every operating system update, for every new version of gnuradio released, something may change just enough to complicate things anew. For this reason I strongly advise you to follow the Linux path. The gnuradio tool has been developed natively for Linux and has always been easiest to get up and running on that platform. This will likely always be the case.

One final note: you might be tempted to create a virtual machine (VM) and install your software there. You may be able to get this to work for your software, but the hardware will likely be a different story. VMs typically have their biggest issues when dealing with hardware connectivity and performance. For this reason, I have not included any information on VM installations in this book, and will assume that you will dedicate a partition to Linux.

2.2 Binaries vs Source Code

I will use the Ubuntu Linux distribution in this book. Honestly, it is just easier for me to support one Linux distribution rather than write installation instructions for multiple Linux distributions. By the time you read this, the instructions may have changed, so see *www.fieldxp.com* for updates.

The instructions here are for Ubuntu 14.04.4. Your mileage may vary with other Ubuntu versions. One of the reasons I picked version 14.04 is because it is a Long Term Support (LTS) Ubuntu version. These instructions should also work with Ubuntu 16.04 (also an LTS version).

Some software comes pre-installed with a Linux distribution. These program are sometimes called "binaries." Additional software can be installed via the Ubuntu Software Center or Synaptic Package Manager. Many people prefer to install software using **`apt-get`** on the command line. The **`apt-get`** command is not covered here.

Yet there may be times when software must be installed from "source." This means obtaining the source code for a software application, and compiling the application from the source code.

"Source code" is the lines of computer code a software developer writes in a particular computer language such as C, C++, Java, etc.

"Compiling" is the act of taking the source code and creating a program (also known as a binary).

2.3 Why Compile from Source Code?

There are different reasons why one may want to compile a program from source instead of using the binary available from the Ubuntu Software Center or Synaptic Package Manager:

Availability: The program may not be available via the Ubuntu Software Center or Synaptic Package Manager. Some companies and software developers only make their program available via source code.

Latest version: The latest version of the software may not be the one accessible by the Ubuntu Software Center or Synaptic Package Manager. If you want the latest version then you have to compile it manually.

Features: Some software features must be enabled when the software is compiled. A binary may have certain features disabled. If the binary that someone is using lacks a feature (and the feature exists), then the software will have to be compiled with the feature enabled.

Optimization: Compiling a program allows the user to obtain better performance by targeting the machine (e.g. Intel or AMD processor) that will be running the program.

We will be using a program called PyBOMBS to install the SDR tools from source. This will enable us to run a later version of these tools than what may be available otherwise.

2.4 Steps for Installing from Source Code

While the exact steps may vary, in general one must:

1: Download (or obtain) the source code

2: Unpack the source code

3: Configure the source code

4: Build (or compile) the program binary

5: Install the program binary

6: (Optional) Uninstall the program

Fortunately, using the PyBOMBS program, much of the above will be automated. The following sections 2.5 through 2.7 are provided here as a support in case you need to install from source some software in the future. If you don't have an urgent need to install some software from source, you might want to skip over this material and go directly to section 2.8.

2.5 Obtain and Uncompress the Source Code

However the software is obtained (from a web site, CD, email attachment, etc.), it will most likely come as a compressed tarball.

A tarball is a file containing one or more files. If there are directories present, then the hierarchy of directories and files is preserved. An archive is not a compressed file, but rather an orderly collection of files.

- The ".tar" extension denotes a tarball.

The tarball is then compressed in order to reduce the size of the file. The ".gz" or ".bz" or ".bz2" extension denotes a compressed file.

- Files with the ".gz" extension are compressed with the GZip algorithm
- Files with the ".bz" extension are compressed with the BZip or BZip2 algorithm.
- Files with the ".bz2" extension are compressed with the BZip2 algorithm.

Example: After obtaining the file example.tar.gz, one must first uncompress the file.

```
gunzip example.tar.gz
```

This will expand the compressed tarball and a file named "example.tar" will be created.

2.6 Extract (un-tar) and Configure the Source Code

To unpack the tarball:

```
tar -xvf example.tar
```

- The “x” option extracts the contents of a tar file
- The “v” option verbosely lists the files inside the tar file
- The “f” option is used when specifying a file name

There will now be a directory named “example” in the current directory. Change into this directory by:

```
cd example
```

The source code is present in the “example” directory. It must now be configured before compiling. Multiple things will happen with configuring. The configure program will check the computer for items such as memory, CPU type, what tools for compiling are present, etc.

To start the configuration program, type:

```
./configure
```

- The “./” tells Linux to run the program even if it is not in the path.
- The path specifies a set of directories which the operating system will search for when the user issues a command.

2.7 Compiling and Installing the Software

The information gathered from the configure program will then be used to generate something called a makefile.

The makefile will be used by the Make program to create the program (binary) from the source code.

To create the program, type:

```
make
```

The make program will consult the makefile that specifies the order that the different source code files are to be compiled.

Alternatively, one may use the command:

```
make clean
```

- Running **`make clean`** can get rid of files that are not needed after compiling (and save disk space).

The time it takes to run the **`make`** command will depend on a number of factors such as the size and complexity of the program being compiled, the CPU and speed of the computer, etc.

- In the Linux world, the make program usually uses the “gcc” compiler to compile the software.

To now install the compiled program, type:

```
make install
```

To uninstall the program, make sure the following command is executed from the directory where the **`make`** command was run:

```
make uninstall
```

2.8 Installing gnuradio on Linux

So you listened to my warnings and decided to go with Linux. Great! This means, however, that you'll need a computer with Linux installed on it, specifically Ubuntu version 14.04.04 or Ubuntu version 16.04. If you don't have one of these operating systems installed and are unsure of how to do so, please find a guide online to walk you through the process. It's also a good idea to apply any Ubuntu system updates and to reboot before following the instructions in this chapter.

As you set up your Linux partitions, ensure that your computer has at least 50 GB of space available for the operating system, gnuradio, and the data files with which we'll be working. 100 GB or more would be better.

Once you have an Ubuntu installed, the general flow we're going to follow is this:

- First, we will install an application called git, which is a very common tool used to fetch files from a software repository (often online).
- Second, we will install some dependencies (software upon which other software depends).
- Next, we will download something called pip. This is a program that helps us install other programs that are written in the Python programming language.
- After that, we will use the pip tool to download and install an application called pyBOMBS. This is a special software manager that is used to install gnuradio as well as other SDR related software utilities (for instance, software that enables HackRF functionality from inside gnuradio).
- Lastly, we will handle the environment variables required by gnuradio. These are just settings that Linux needs to run gnuradio properly.

Don't worry if a few of the steps I just mentioned are a bit confusing. I'll walk you through each of the steps below, giving you the specific commands to type.

First, a word of warning. Some gnuradio install guides will direct you to run the following simple command WHICH YOU DO NOT WANT TO TYPE. But this is a command which you may see:

```
sudo apt-get install gnuradio
```

This will indeed install gnuradio, but it will not be a very recent version. The steps we'll go through below will get the latest version of gnuradio available, and this will make a difference. Several of the exercises in this book series will not work with the older version of gnuradio.

I have also provided a rough time estimate for how long each command will take. This estimate assumes a mid-range 2015 workstation and a 10 Mbit/second Internet connection (your mileage may vary).

Let us assume that we're starting in the home directory. To make sure, go ahead and type the following in a terminal window:

```
cd
```

(Time required: instantaneous)

Time to make sure Ubuntu Linux is up-to-date:

```
sudo apt-get update
```

(Time required: could take a few minutes)

First we'll install git, using apt-get. You'll need to enter your password before the command will complete.

```
sudo apt-get install git
```

(Time required: less than a minute)

Now it is time to install some dependencies, software that the SDR tools will depend on in order to install and/or run:

```
sudo apt-get install libyaml-dev
sudo apt-get install libssl-dev
sudo apt-get install python-dev
```

(Time required: less than a minute)

Next, we'll create a directory in which to place gnuradio and other software utilities. If you don't want to call it "sdr" feel free to give it another name or location, just be careful to use that other name throughout the guide in place of "sdr" name.

```
mkdir sdr
cd sdr
```

(Time required: instantaneous)

Now we'll use install a tool that will be used to manage Python packages:

```
sudo apt-get install python-pip
```

(Time required: less than 1 minute)

The next step is to update the python-pip software we just installed:

```
sudo easy_install pip
```

(Time required: less than 1 minute)

Now we use the pip tool to install PyBOMBS:

```
sudo pip install PyBOMBS
```

(Time required: less than 1 minute)

The following commands will add "recipes" for PyBOMBS. Different pieces of software will be installed by PyBOMBS via various recipes.

```
pybombs recipes add gr-recipes git+https://github.com/gnuradio/gr-recipes.git
pybombs recipes add gr-etcetera git+https://github.com/gnuradio/gr-etcetera.git
```

(Time required: about 1 minute)

Next we will designate the previously created "sdr" directory as the target for software installed by PyBOMBS:

```
pybombs prefix init ~/sdr -a myprefix
```

(Time required: instantaneous)

Finally, we will use PyBOMBS to install gnuradio and gr-osmosdr (this provides gnuradio with an interface to the HackRF hardware):

```
pybombs install gnuradio gr-osmosdr
```

(Time required: hard to say, maybe 40 to 70 minutes)

With Linux, we can use the source command on a file to execute a list of commands in that file (rather than typing each command one at a time). The install process we just completed will generate a number of environment variables that need to be set for gnuradio to work. Fortunately, all of these variables have been automatically added to a single file that you can simply source with the following command:

```
source ~/sdr/setup_env.sh
```

(Time required: instantaneous)

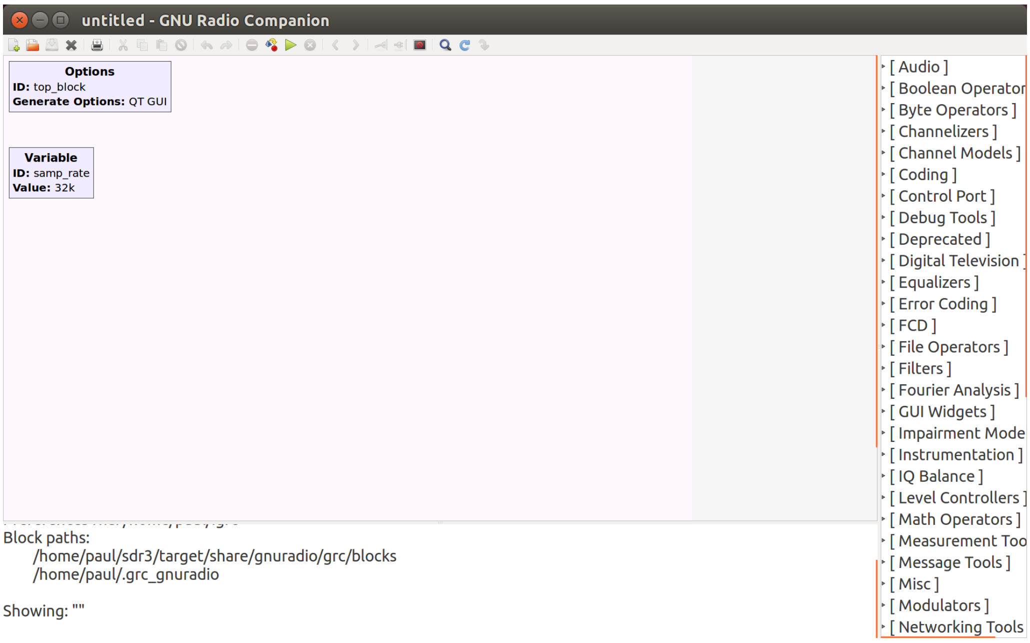
Running this source command will be enough to get gnuradio running for now, but it will need to be run again every time you reboot your machine. To run this command automatically at boot time, edit the file called ".bashrc" in your home directory and add the source command shown above to the end of the file.

At this point, we should have a working installation of gnuradio. Go ahead and try to run its graphical interface by typing:

```
gnuradio-companion
```

(Time required: instantaneous)

If successful, you should see a window similar to this appear:



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

```
Block paths:
    /home/paul/sdr3/target/share/gnuradio/grc/blocks
    /home/paul/.grc_gnuradio

Showing: ""
```

Both gnuradio and PyBOMBS undergo changes over time. This software work is undertaken by talented programmers in the open source community. Despite their best efforts, there may be hiccups during the installation process. The installation instructions given here may not work in the future. See the *www.fieldxp.com* website for more up-to-date instructions.

Almost done, now let's close the gnuradio window and go back to our terminal window. We need to install some additional graphical support software in gnuradio. This will enable more advanced displays so we can better see what's happening on the screen. The following will use pip to install OpenGL support for Python.

```
sudo pip install PyOpenGL PyOpenGL_accelerate
```

We may need to type in a few "Y" characters to make this go.

Finally, we will enable our new graphical functionality in gnuradio by creating a file using gedit (or your favorite text editor) in the following location:

```
gedit ~/.gnuradio/config.conf
```

This file needs to contain the following contents, so we'll type (or paste) it in.

```
[wxgui]
style=gl
fft_rate=30
waterfall_rate=30
scope_rate=30
number_rate=5
const_rate=5
const_size=2048
```

Now we should have everything we need to get started, including gnuradio and all of the hardware drivers and utilities needed by the HackRF unit (the SDR hardware we mentioned in the Preface). Keep in mind we will not need SDR hardware such as the HackRF for this volume.

The following is for those who do have a HackRF SDR unit. We've done a sanity check already on gnuradio, but we can also do a simple test of the HackRF utilities and drivers (think of a driver as the software that the operating system uses to manage hardware). First, we need to plug in the HackRF device to a USB port on our computer. Then we run the following command:

```
hackrf_info
```

(Time required: instantaneous)

If things are working, you should see a message telling you that a HackRF board was found, along with information about the board. That message will look something like this:

```
Found HackRF board.
Board ID Number: 2 (HackRF One)
Firmware Version: <a series of interesting characters>
Part ID Number: <more interesting characters>
Serial Number: <even more interesting characters>
```

If the hackrf_info command produces an error message, then type:

```
cd ~/sdr/src/hackrf/host/libhackrf
sudo cp 53-hackrf.rules /etc/udev/rules.d/.
```

Now unplug the HackRF unit, plug it in again, and retry the hackrf_info command.

2.9 Validating your gnuradio Installation

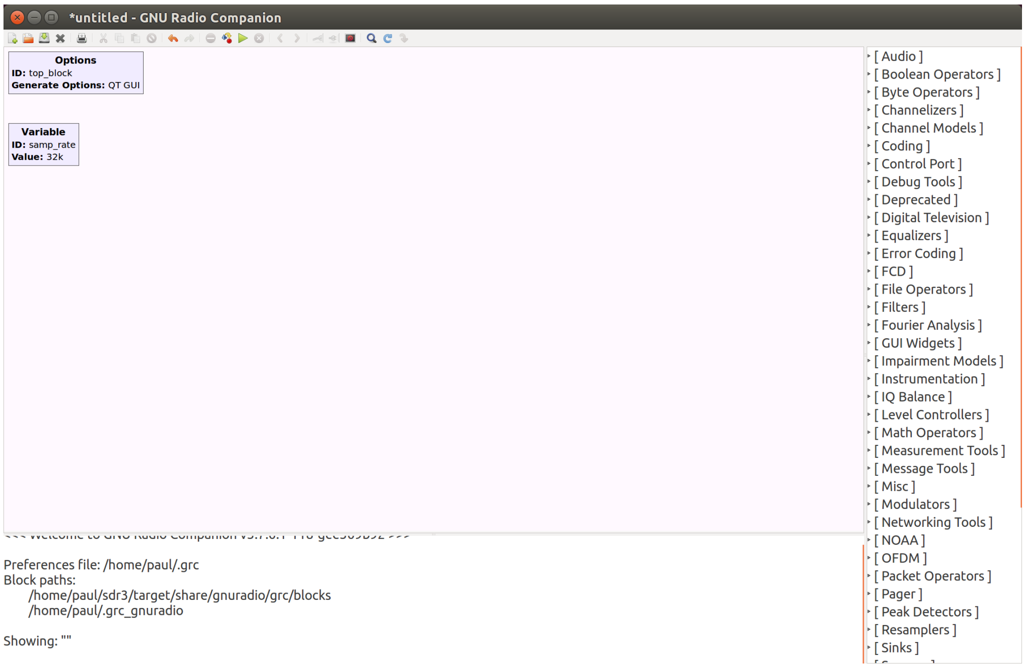
Regardless of which operating system you used to install gnuradio, you should create the following project just to make sure you have things working OK. This is not a thorough or exhaustive test, just a sanity check to make sure the big pieces are in place and functioning correctly.

I'll guide you through a couple of steps that may not make a lot of sense at first, but don't worry about that yet. It will become progressively more clear as I peel back the layers of the SDR onion. Please note that screenshots in the book may not match up exactly with your gnuradio-companion display (due to, for instance, different versions of gnuradio).

1) open gnuradio-companion

- For Linux, you will simply type **`gnuradio-companion`** in a terminal window

- When you're done, you should see something like this:



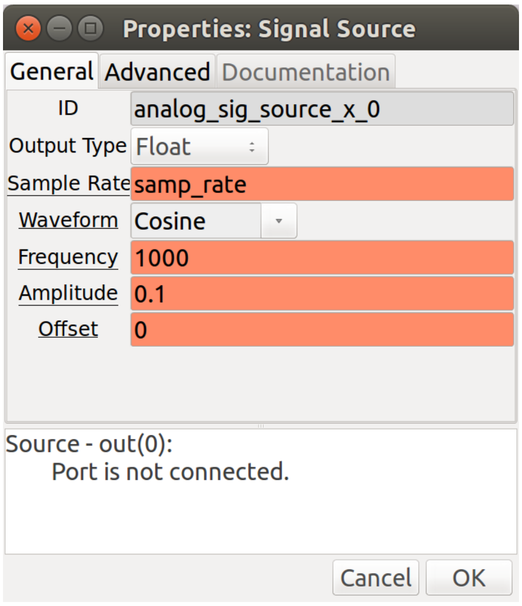
```
<<< Welcome to GNU Radio Companion v3.7.6.1-110-gce309b92 >>>

Preferences file: /home/paul/.grc
Block paths:
	/home/paul/sdr3/target/share/gnuradio/grc/blocks
	/home/paul/.grc_gnuradio

Showing: ""
```

2) Type Control-F to bring up a search bar on the top right side of the screen

3) Type **`signal`** into the window (you don't have to hit the return key). You'll see an option under Waveform Generators called **`Signal Source`**. Double click it and it will appear in the diagram.

4) Erase the existing text in the search window and type **`audio`** in its place. You'll see an option called **`Audio Sink`**. Double click this and it too will appear in the diagram.

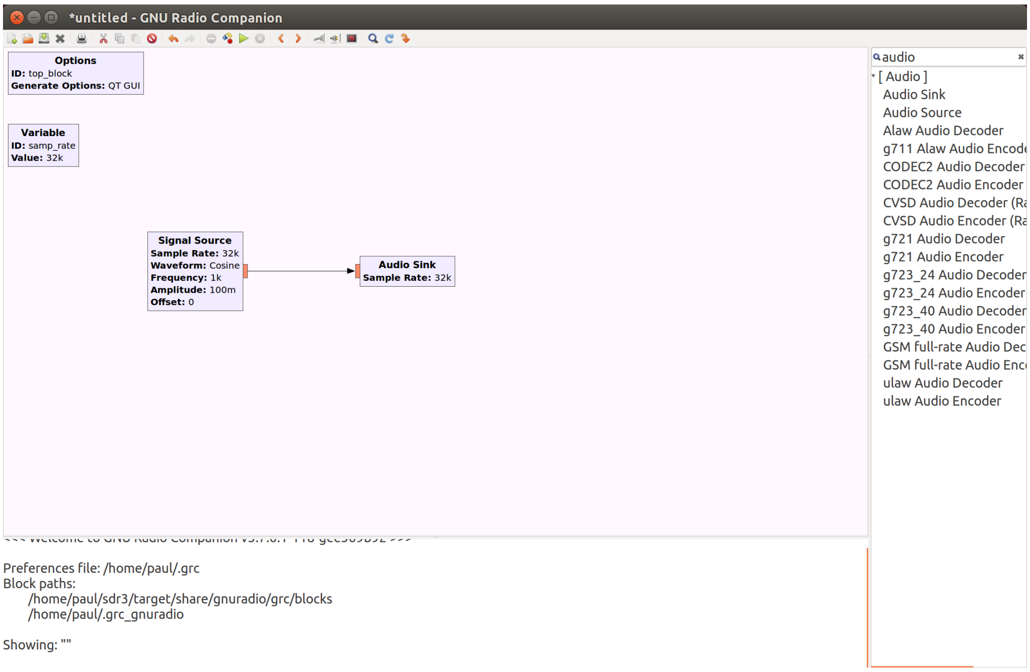
5) Double click the Signal Source block and another window will come up. The contents won't make a lot of sense, but that's OK. We only need to change two things. Enter **`0.1`** for the Amplitude and change the Output Type from Complex to **`Float`**. Then click OK.



ID	analog_sig_source_x_0
Output Type	Float
Sample Rate	samp_rate
Waveform	Cosine
Frequency	1000
Amplitude	0.1
Offset	0

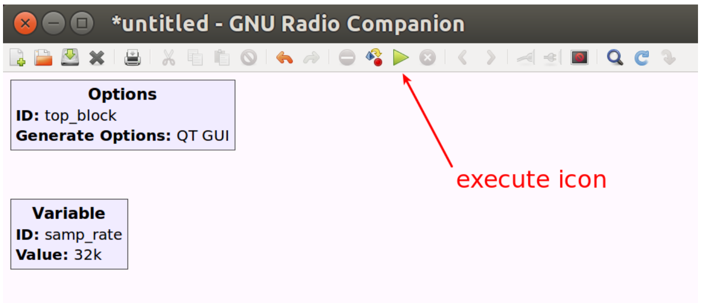
Source - out(0):
Port is not connected.

6) Now click the orange tab on the signal source block and then click again on the orange tab on the audio sink block. You should see a connection appear between the two blocks.



```
Preferences file: /home/paul/.grc
Block paths:
     /home/paul/sdr3/target/share/gnuradio/grc/blocks
     /home/paul/.grc_gnuradio

Showing: ""
```

7) Click the Execute icon on the toolbar. Gnuradio will ask you to save the project at this point. Name the project verify.grc and click the Save button..



8) After saving, gnuradio will take a second to generate and run your project. After a few seconds, you should hear a moderately-pitched tone. If so, then your gnuradio sanity check has passed! Let's move on to bigger and better things.