Scotty Cowling, WA2DFI

PO Box 26843, Tempe, AZ 85285: **scotty@tonks.com**

Hands-On SDR

Sharing Radios on the Network

I would like to thank those of you that took the time to drop me an e-mail and let me know that my introduction to FPGA programming for SDR was useful. In spite of the positive feedback, I am going to shift gears this installment. FPGA aficionados do not despair; I will return to the world of *Verilog* in my next column. As Monty Python would say, "...and now for something completely different".[1]

In this column, I will show you how to listen to various web-based SDRs around the world. I will then walk you through the procedure to set up your own server and share your radio with others, both on your local network and on the Internet. I am going to limit my discussion to receive only. Transmitting is possible, but in my opinion, it is not quite ready for prime time. It is much easier for many "listeners" to listen to the same digitized and down-converted data from the antenna than it is to figure out a way to combine the audio from many "talkers" into one stream of data, and up-convert it for transmission. Well, maybe it is not *that* hard to do, but this feature isn't readily available in any of the software that I could find on line. This leads me to define the difference between software ***controlled*** SDR and software ***shared*** SDR. A software controlled SDR is a radio intended to be used by one operator. A software shared SDR is intended to be used by more than one operator. A shared SDR may or may not be capable of supporting simultaneous users; a controlled SDR does not even need this feature. See the sidebar, Networks, Servers and Clients.

[1]Notes appear on page 42

What Do I Need?

As usual, you probably want to know the answers to a few questions: "What do I need to know?" and "What equipment do I need?"

As a bare minimum, you need nothing more than an Internet connected computer running a browser. Since this is a *Software* Defined Radio column, and you need a computer to run the software, it is not much of a leap of faith to assume that you already meet the minimum criteria! All of the software that I use in this month's column is available for free download. You must be familiar with installing and running *Windows* applications to take advantage of this free software. To get the most out of the final section on the *QtRadio* server, you must be able to install and run Ubuntu *Linux* and its applications.

You do not need any radio hardware if you do not intend to set up a server; you will be using other people's radio hardware instead. To run a **RemoteSDR** server, you will need an RFSPACE radio (such as an SDR-IQ). To run a **ghpsdr3-alex** server, you will need one of the many supported radios.[2] Even if you do not have any radio hardware and just choose to follow along with the text, you can still get in on the remote radio action and have lots of fun!

Listen to Someone Else's Radio

The simplest way to get your feet wet in Internet shared radios is to just point your

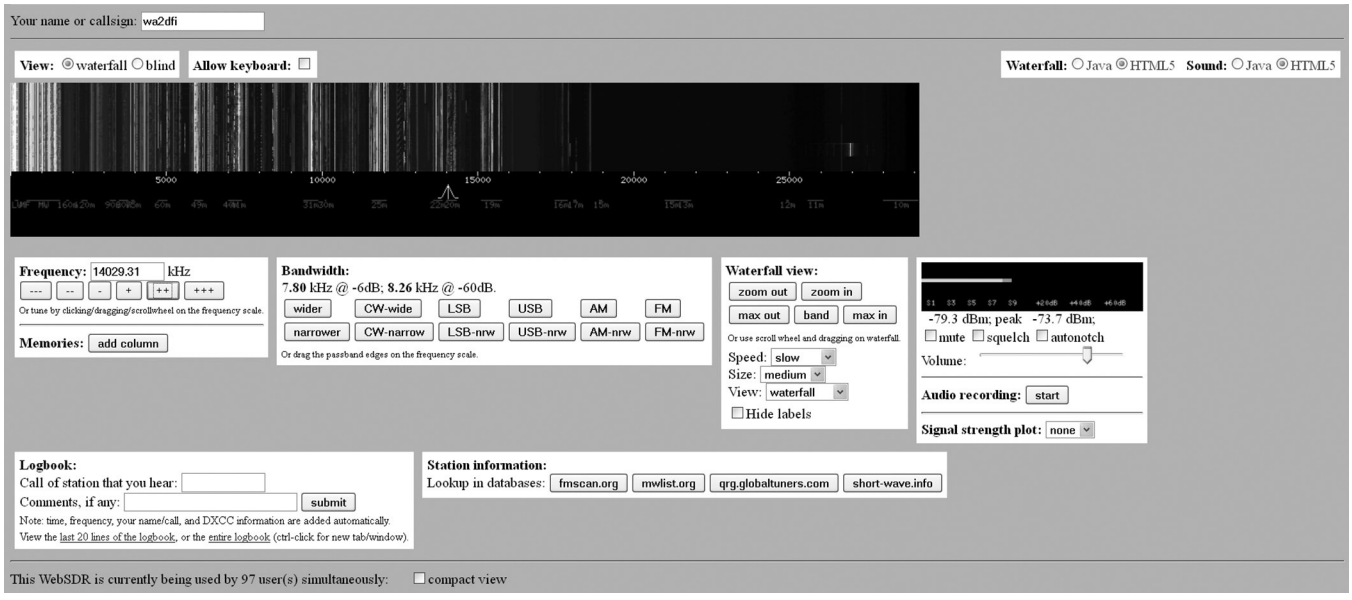

Figure 1 — You can listen to the University of Twente browser-based WebSDR at: websdr.ewi.utwente.nl:8901.

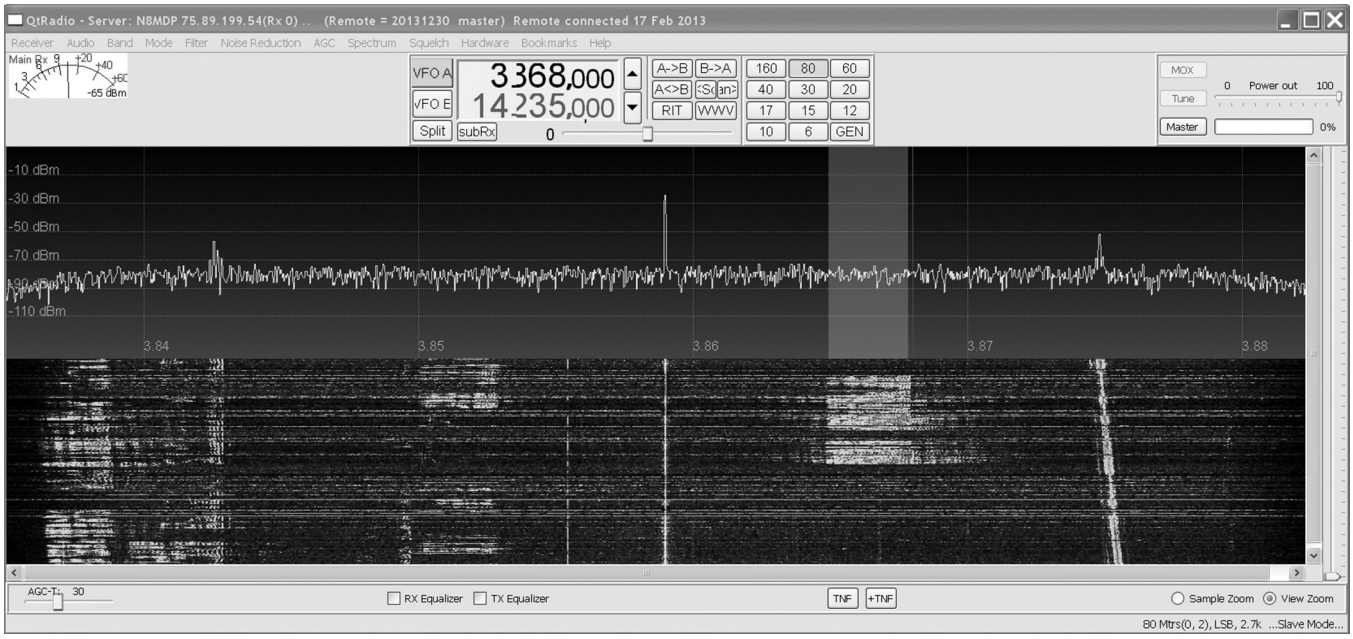

Figure 2 — This is a screenshot of the *Windows QtRadio* Client.

browser here: **websdr.ewi.utwente.nl:8901**. This is the University of Twente WebSDR in The Netherlands. WebSDR is a wide-band, multi client web-based SDR that was originally set up in 2008, making it the oldest on-line SDR. See Figure 1. Their hardware has progressed through several increasingly sophisticated iterations, and their web site contains a wealth of historical and technical information. Best of all, WebSDR does not require you to install any software on your computer in order to listen!

Now let's move on to something that is a bit more involved to set up, namely installing client software on your device. Notice that I said *device* instead of *computer*. The software we will use next is available for many devices, such as Android phones and tablets, computers and tablets running Mac *OSX*, *Linux* and *Windows*. On phones and tablets, the client software is called an *app*, short for *application*. The client for *Windows* is called *QtRadio*, and you can download a pre-built runtime version from the *QtRadio* website.[3] The Android app is called *glSDR* and is available for free from the Google play store.[4] The source files for *Linux* and Mac *OSX* are available, but you will have to compile them yourself. We will get to this later. For *Windows*, simply download the .zip file and extract it to a new directory. Run the **QtRadio.exe** program in the main directory by double-clicking it; there is no installation required. To make it easier to access, you can create a shortcut and place it on your desktop.

Once *QtRadio* is running (Figure 2), click on the **Receiver** menu in the upper left corner and select **Quick Server List**. A new window will open (Figure 3) showing all of

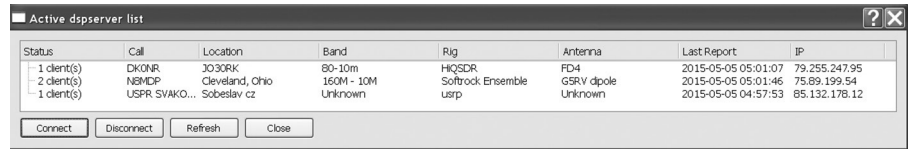

Figure 3 — Here is a screenshot of the *QtRadio* active dspserver list.

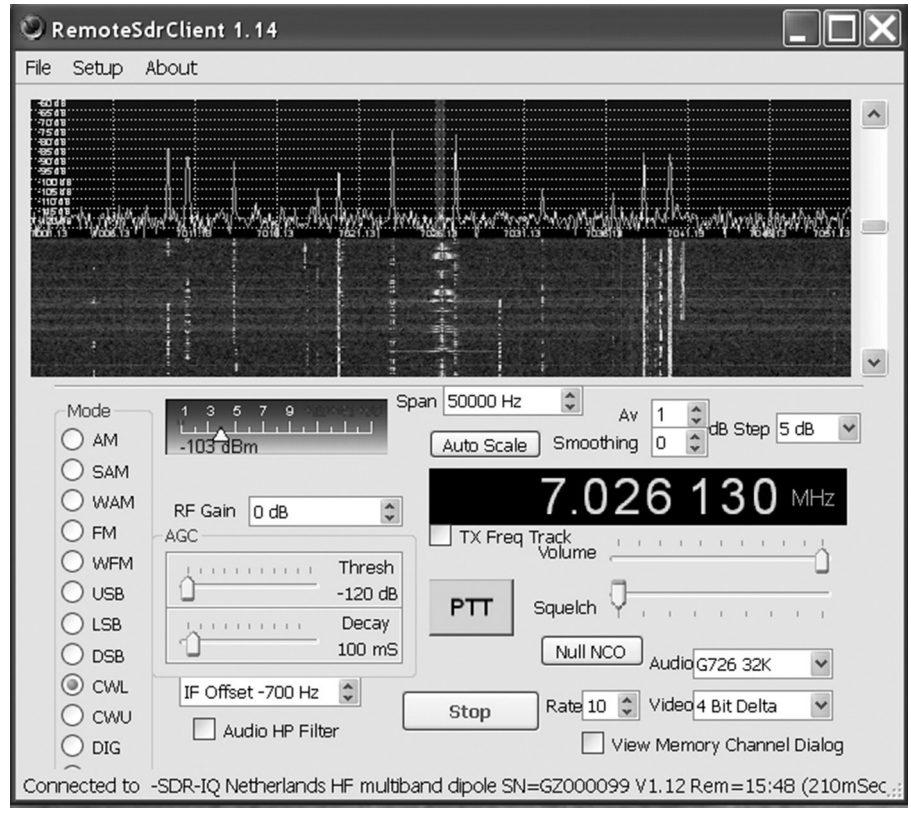

Figure 4 — A screenshot of the *RemoteSDR* Client connected to an SDR-IQ in The Netherlands.

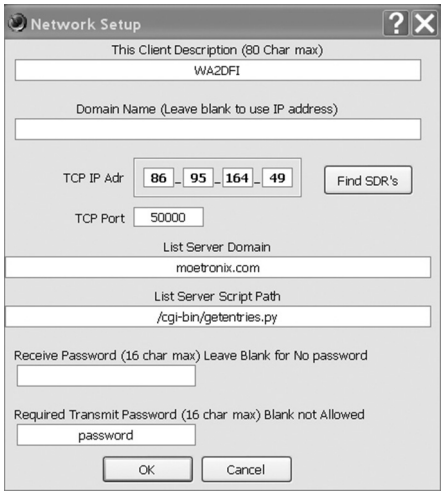

Figure 5 — This is the *RemoteSDR* Client Network Setup screen.

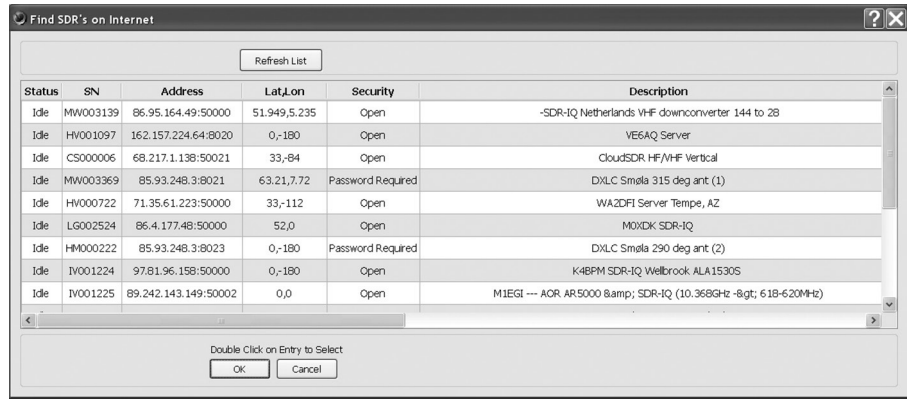

Figure 6 — This is the *RemoteSDR* Client "Find SDR's" screen.

the available dspservers. Double click on one that has 0 clients (highlighted in green) and you are listening on that radio! If you select a radio that is in use, you will not be able to control it, but you will instead hear whatever the master user (the one who is controlling the radio) hears. You only get to control the radio if you were first in line to select it.

RemoteSDR: Easy as Cake

Mangled idioms aside, I would like to segue into the server application, but the *QtRadio* server only runs under *Linux* and is a bit complex to set up. Let's take a look instead at somewhat more hardware limited server/client software that runs on *Windows* but is very easy to install. The limitation is that it requires an RFSPACE radio to act as a server. Even if you do not have an SDR-IQ or SDR-14 receiver from RFSPACE, installing this software will be a useful way to familiarize yourself with simple SDR server/client systems.

The *RemoteSDR* software is available as a free download, and it installs both the server and client programs in one operation.[5] Download the Quick Start Guide (**QuickStartGuide107.pdf**) and the server/client software installer (**RemoteSdrClientWinSetup_114.exe**). Run the installer and select all the default options. This will install both the client and the server on your *Windows* computer. You will see two new icons on your desktop: one for the server and one for the client. Go ahead and launch the client (Figure 4), click on the **Setup** menu and select **Network** (Figure 5). Enter a description for your client and then click the **Find SDR's** button. (Yes, I know that should be plural rather than possessive, but that's the way the programmers typed it!) Figure 6 shows a listing of the SDRs that *RemoteSDR* found. Double click on your selection, click **OK** and then click on the **Start** button. You are now in control of the remote radio!

Now that we can listen to someone else's radio, let's set up the server to let others listen to our radio. Unless you have an RFSPACE radio such as an SDR-IQ, you will not be able to do this, but let's cover it quickly before moving on to a more complex server that supports many different hardware platforms.

When you installed the *RemoteSDR* client application, remember that you also installed the *RemoteSDR* server application. Find its icon on your desktop and open it (Figure 7). Click on **Setup** and select **Server Setup** (Figure 8). All you need to enter here is your SDR description and your location in the **Latitude** and **Longitude** boxes and click **OK**. Figure 9 is a block diagram of the *RemoteSDR* client and server system.

Now open the *RemoteSDR* client and go to **Network Setup** and **Find SDR's**. You should now see your own SDR listed among the others. Try double clicking your own radio, then **OK** and **Start**. Did it work? It probably did not (mine didn't). Why didn't it work? Go back to the client Network Setup screen and look at the TCP address. This is the IP address that was published by the server software. It is the IP address that your ISP provided to your modem and will not be accessible from the local network side that your PC is connected to. You must use a local address, such as 127.0.0.1 (localhost) or the actual IP address of your PC on the local network. Look at the *RemoteSDR* server window for these addresses. Pick one and type it into the client Network Setup screen's **TCP IP Adr** boxes. Then click **OK** and **Start** and you will be listening to your own radio!

In order for others to listen to your radio on the Internet, you have to do one more thing: forward the data from your Internet IP address port 50000 to your local network port 50000. See Figure 10 for an explanation of why this needs to be done. You must set up a route in your home router to make this connection, so that the external clients (on the Internet) can connect to your server on your local network. Note that your radio will appear in the list of Internet SDRs even without port forwarding, but no one outside your local network will be able to connect to your radio and listen until you set up port forwarding. For a more

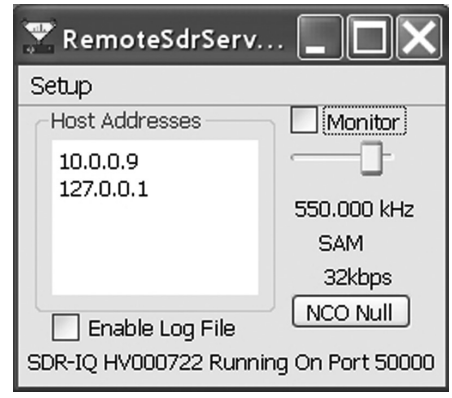

Figure 7 — Here is the *RemoteSDR* Server screen.

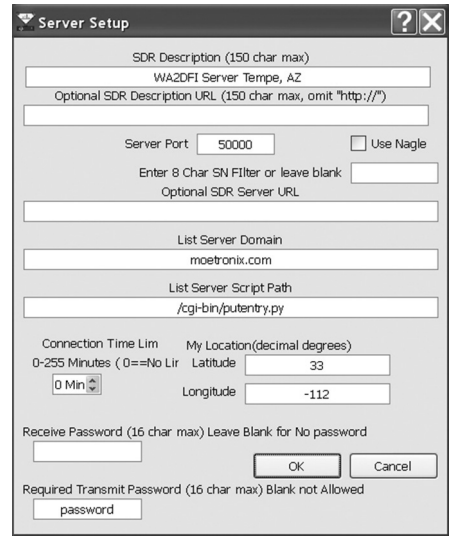

Figure 8 — The *RemoteSDR* Server Setup screen allows you to configure your server.

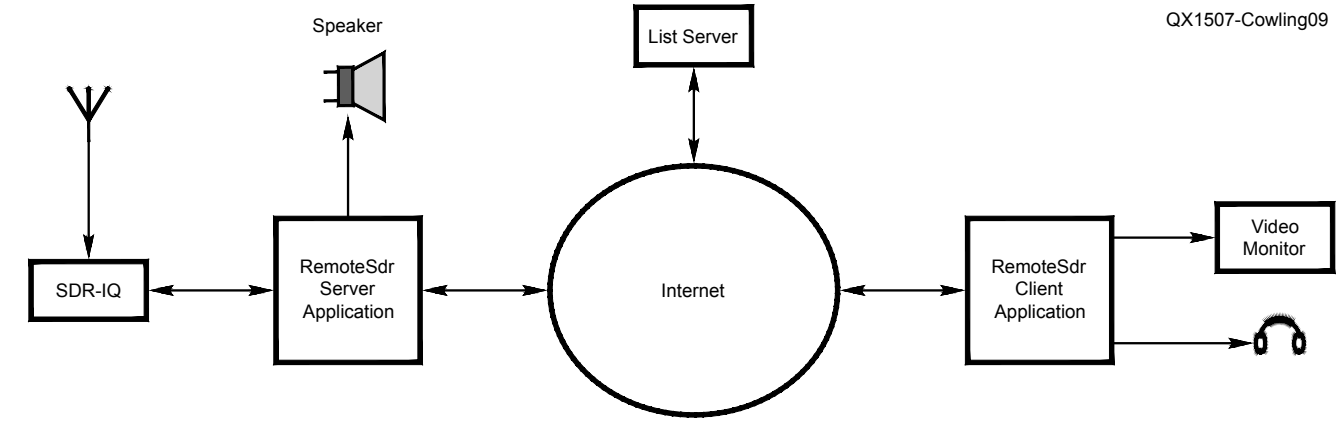

Figure 9 — *RemoteSDR* system block diagram (adapted from the graphic on the RFSPACE website).

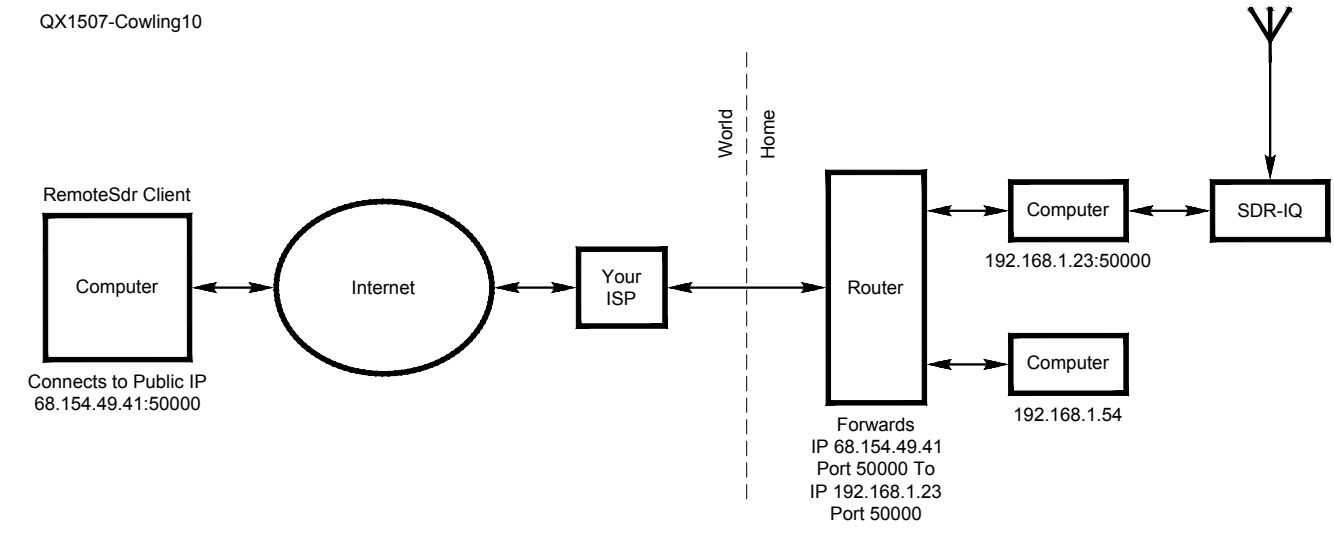

Figure 10 — *RemoteSDR* system port forwarding diagram (adapted from the graphic on the RFSPACE website)

detailed discussion, see section 3 of the *RemoteSDR* Quick Start Guide.

And Now for Something Completely Different (again)…

Actually, our final exercise is a tough one. It is really more of the same as what we have already done, but with different hardware and under a different operating system. Let's set up a ghpsdr3-alex server under Ubuntu 14.04 *Linux*. We have already run the *QtRadio* client software to listen to other servers on the Internet. The obvious next step is to set up a server of our own. The good news is that the ghpsdr3-alex server software supports many different hardware platforms (see Note 2). The not-so-good news is that it runs only under *Linux* and is not trivial to set up. We have never let that stop us before, however, so let's dive in!

Ghpsdr3-alex Server Linux Setup

Thanks to Dan Babcock, N4XWE, we have a 10-step guide to install and run ghpsdr3-alex on a computer running Ubuntu 14.04 LTS 32-bit Linux. This will install all three parts of the software. The **Server** talks to the hardware and the **dspserver** does the heavy lifting and serves up receive data to the ***QtRadio*** receiver client (see Figure 11).

Step 1: This step will make sure that your system is up to date and has the compiler tools installed. Make sure that the universe repository is enabled (it is enabled by default). Open a terminal window and run the following commands.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install make
$ sudo apt-get install gcc
$ sudo apt-get install g++
$ sudo apt-get install autoconf
$ sudo apt-get install automake
$ sudo apt-get install autotools-dev
$ sudo apt-get install libtool
$ sudo apt-get install git
$ sudo apt-get install subversion
```

Step 2: Use the following command 20 times to install the 20 prerequisite packages listed by name in Figure 12. Be sure to substitute the name of each package from that list for *package* in this command.

```
$ sudo apt-get install package
```

Step 3: Install qt5. Visit **qt.io/download-**

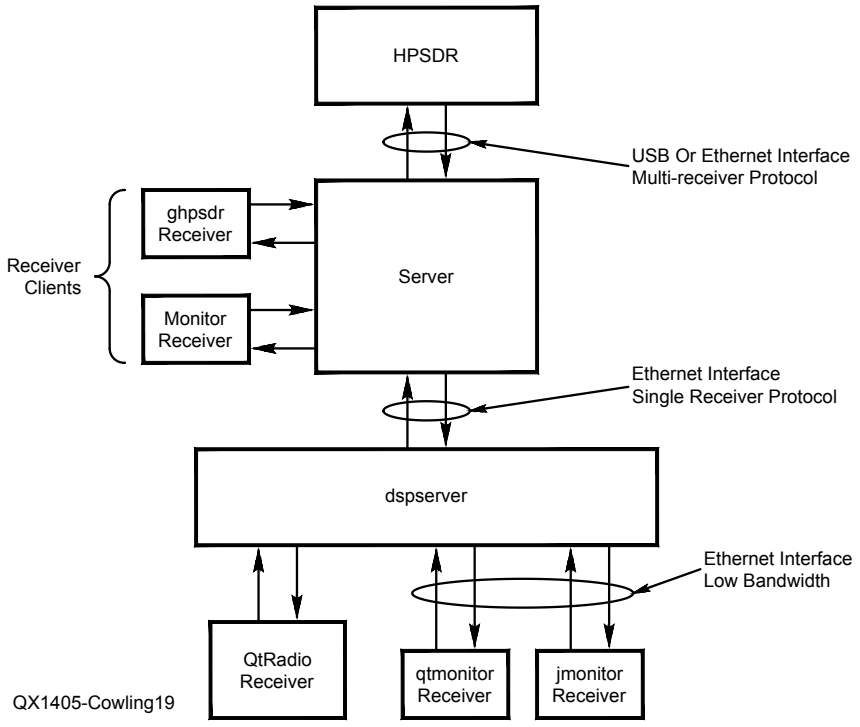

Figure 11 — This is the ghpsdr3-alex system diagram.

open-source and download the current package for your system. Do not use the recommended one; go to the **Offline Installers** page and select the version that matches your system (for example, **Qt 5.4.1 for Linux 32-bit**). Change the file permissions to allow execution, and execute it. Find where the **qmake** file is installed using the output from the locate command. Substitute this path for the example path in the export command below. The path may vary slightly if you are using 64-bit *Linux*.

```
$ chmod +x downloaded_file_name
$ sudo ./downloaded_file_name
$ sudo updatedb
$ locate /bin/qmake
$ export PATH=/opt/Qt5.4.1/5.4/gcc/
bin:$PATH
```

Step 4: Verify that path points to the qt5 version that you installed in Step 3 above. (The qmake version is not important.)

```
$ qmake -v
```

The result should be something like below showing the Qt version that you installed in Step 3. The path may vary slightly if you are using 64-bit *Linux*.

```
QMake version 3.0
Using Qt version 5.4.1 in /opt/
Qt5.4.1/5.4/gcc/bin
```

Step 5: Install the codec2 low bit rate CODEC for audio encoding and decoding.

```
$ cd
$ mkdir src
$ cd src
$ svn co https://svn.code.sf.net/p/
freetel/code/codec2 codec2
$ cd codec2
$ mkdir build
$ cd build
$ cmake ../
$ sudo make
$ sudo make install
$ sudo ldconfig
```

Step 6: Move the codec2 header files to the correct place.

```
$ cd /usr/local/include/codec2
$ sudo cp * ..
```

Step 7: Move the codec2 libraries to the correct place. The directory in the first command is for 32-bit Ubuntu. For 64-bit Ubuntu it will likely be /usr/local/lib/x86_64-linux-gnu instead.

```
$ cd /usr/local/lib/i386-linux-gnu
$ sudo cp * ..
```

Step 8: Install ghpsdr3-alex from the Git repository.

```
cmake
freeglut3-dev
gcc-multilib
libconfig8-dev
libevent-dev
libfftw3-dev
libglu1-mesa-dev
libpulse-dev
libortp-dev
libqt4-opengl-dev
libsamplerate0-dev
libspeexdsp-dev
libssl-dev
libusb-0.1-4
libusb-1.0-0-dev
libusb-dev
libxcb-composite0-dev
portaudio19-dev
qtmobility-dev
xdg-utils
```

Figure 12 — This list shows the prerequisite software packages to be installed in Step 2.

```
$ cd
$ cd src
$ git clone git://github.com/
alexlee188/ghpsdr3-alex
$ cd ghpsdr3-alex
$ git checkout master
```

Step 9: Compile the code.

```
$ autoreconf -i
$ ./configure
$ make -j4 all
$ sudo make install
```

Step 10: Run the server, dspserver and client software. You will have to run each of these in a separate window. To open a new window, type <crtl><alt>T. Before you run **hpsdr-server**, make sure that your openHPSDR, Hermes or SDRstick hardware is connected to the network and powered up. If you are running other SDR hardware, use the appropriate hardware server and command-line options in place of **hpsdr-server**.

```
$ hpsdr-server --metis --samplerate
384000
$ dspserver --hpsdr --lo 0
--nocorrectiq
$ QtRadio
```

When you run the second line in step 10 for the first time, the software will create a new **dspserver.conf** template file for you to fill in with important things like your call sign, location, band, rig and antenna. This information will be listed in the web database that you see when you bring up the **Quick Server List** in *QtRadio*. Edit this with your favorite text editor to reflect your station information. When you run the server for the second time (after you have edited the **dspserver.conf** file) you will be prompted to create a key and self-signed certificate. The three commands to do this are listed in the prompt, but I have reproduced them below to make it a bit easier for you after they have scrolled off your screen. Run these three commands and answer any questions you are asked.

```
$ openssl genrsa –out pkey 2048
$ openssl req –new –key pkey –out
cert.req
$ openssl x509 –req –days 365 –in
cert.req –signkey pkey –out cert
```

To test your setup, click on **Receiver** in the *QtRadio* toolbar and select **Configure** (see Figure 13). In the **Server** tab, enter 127.0.0.1 and click **Add Host** (or just pick 127.0.0.1 from the drop-down list). Click Close, then click on **Receiver** again, but this time select **Connect**. A dialog box may pop up with additional settings, depending on the hardware that you are running. You can make your choices and dismiss the dialog box, or you can just drag it out of the way for now. Meanwhile, you should be up and running your own shared receiver! The IP address 127.0.0.1 is called the localhost address, and connects the *QtRadio* client to the **dspserver** via a direct connection within the computer.

Note that your receiver is now shown in the list of online dspservers when you display them by clicking **Receiver** and selecting **Quick Server List**. You will notice, however, that the IP address is not right, and you cannot connect to it. There is one more thing that we need to do before others can connect to our local receiver, and that is set up port forwarding for ports 8000 and 9000. Since there are many kinds of routers, I cannot give you specifics on how to do this. On the Netgear router that I have, I set up port forwarding so that any incoming packets from the WAN (outside world) addressed to any IP address on ports 8000 or 9000 forward to the local computer on the LAN (local network) that is running dspserver (mine is at address 192.168.1.2). To make things easier, you will probably need to set this computer's IP address to a fixed value rather than use DHCP to assign it. After you set up port forwarding, the IP address that you see in the **Active dspserver** list should be the IP address of your cable modem or router, and you (and everyone else, too) should be able to connect to your receiver from anywhere on the Internet.

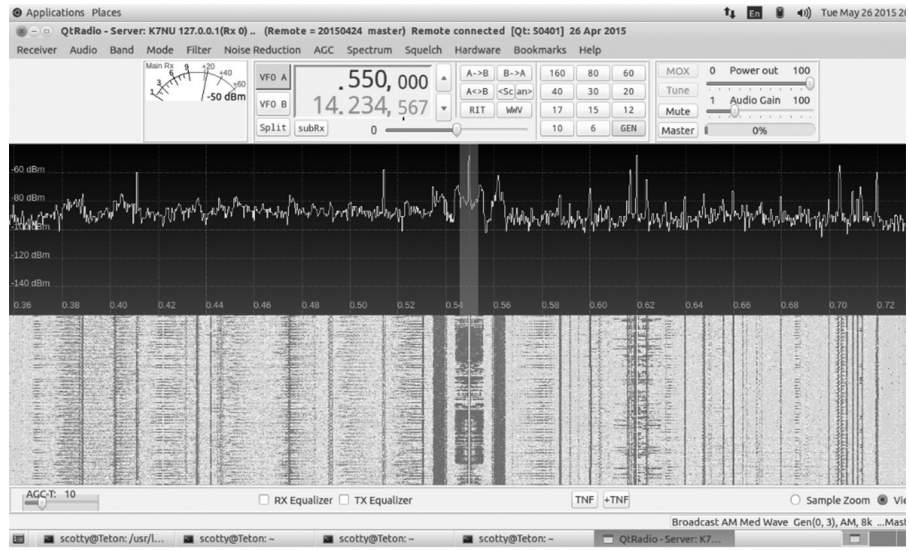

Figure 13 — This screenshot shows the *Linux QtRadio* Client.

Networks and Servers and Clients, Oh My!

While it might be obvious to networking and software gurus, the terms **server** and **client** might not be as familiar to radio experts. In simple terms, a server is a program that serves up data to other programs (called clients) that use the data in some way (display it on your screen, for example). The Internet consists of many (and I mean *billions*) of servers that deliver content to web clients. This content can be HTML web pages, MP3 music streams, MPEG4 video streams or even SDR control and audio streams.

A common example of a client is your web browser. Whether you use Firefox, Chrome, Internet Explorer or something else, this client software connects to a server on the Internet to obtain its content, which it then formats and displays on your monitor (or sends it to your speakers or to another application running on your computer). Clients can also be set-top boxes, game consoles, home theater computers, and the list goes on and on.

In our specific case, the server provides the SDR content to one or more clients. The clients may request specific data (frequency, mode, bandwidth, and so on) from the server, and (if the system is designed properly) the server responds with the appropriate data. It is this server/client architecture that enables many users to share data from one set of receiver hardware.

Back to *Verilog*

After this brief sojourn into the world of Internet shared radios (and now that you are a *Linux* and networking expert), my next column will return to the topic of *Verilog* programming for FPGAs. I will take a look at porting the open-source code for the Altera Cyclone III FPGA on the openHPSDR Hermes board to the Cyclone V FPGA on the BeMicroCVA9 development board from Arrow.[6] The result will be open-source SDR code that anyone can use as a starting point for their own customization, port to yet a different FPGA family or just become more familiar with SDR FPGA design techniques.

Notes

[1]See (**en.wikipedia.org/wiki/And_Now_for_Something_Completely_Different**) for information on the origin of the phrase.

[2]As of this writing, ghpsdr3-alex software supports the following hardware: HPSDR, SDRstick, Softrock, UHFSDR, Perseus, SDR-IQ, HiQSDR, USRP and RTL-SDR DVB-T dongle.

[3]Download the *QtRadio* client application from here: **napan.com/ve9gj/QtRadio_Windows_Master_2014-09-02.zip**

[4]The Android glSDR app is available here: **code.google.com/p/sdr-widget/downloads/detail?name=glSDR32.apk&can=2&q=**

[5]The *RemoteSDR* software and Quick Start Guide are available on this page: **sourceforge.net/projects/remotesdrclient/files**

[6]The BeMicroCVA9 should be available from Arrow Electronics by the time you read this: **parts.arrow.com/item/detail/arrow-development-tools/bemicrocva9**