Robotic Radio and CW Robot, a Building Block for Robotic Radio

Rob Frohne, KL7NA
Walla Walla University
100 SW 4th Street
College Place, WA 99324

Abstract
Look Ma! No hands! My radio is logging stations all by itself. Robotic Radio is an exciting new frontier in amateur radio. This paper introduces the idea of Robotic Radio and overviews some of the open source software and hardware blocks useful in this area.

Introduction
Have you noticed that amateur radio isn't the hit with the younger generation it was when you were young? I have. And it's not just young people; almost all the older ones I know who once were active in the amateur radio hobby are now involved in other things instead. Amateur radio was very influential in career decisions for me and a fair number of my science and engineering colleagues. At Walla Walla College when I was an engineering student, electrical engineering was the most popular variety of engineering, and a number of us were active hams. Now at Walla Walla University mechanical engineering is the most popular by a two to one margin, and if the Morse code requirements hadn't changed, there would be almost no hams among our science and engineering students. Why did these changes occur?

When I was young, Heathkit was thriving selling kits that were cheaper because the customer supplied the labor to assemble the electronics. These days, robots make anything selling in large quantities, and often I find I cannot purchase the components for anything close to the price of the completely built device. Computers, which only a few years ago cost thousands of dollars are not even accepted at the many thrift store recyclers, and fast new computers today are only a few hundred dollars. When personal computers were introduced in the late 1970's and 80's almost all the software was written by the users, and for the most part, they had the source code. It wasn't easy to share that software. Books were printed with basic games for users to type in, and then floppies were sent through the mail. It was still worth the effort to put up large antennas to be able to chat with friends in DX locations. Now with cheap cell phones and the Internet, DX communications on the ham bands seems only a passing novelty. Computer software is now more complex, often being written as a team effort. The Internet provides a great distribution system, and it is a daunting challenge to produce software that approaches the quality and feature set of what is already available commercially, comes without source code, but with license agreements that are too long to read, much less truly agree to. This situation has turned us into computer users instead of developers. Fortunately the open source software movement has provided an alternative to that way of operation which is essential for those of us who enjoy experimenting and tweaking more than appliance operating.

Our country and our world both need more scientists and engineers, and amateur radio can still help develop those technical experts if we allow our hobby to change with the times. Robotic Radio is a new aspect of amateur radio that has the potential to keep young and old interested, because it is very easy and inexpensive to get started with, and yet it has challenges similar to those that kept me climbing my antenna tower to tweak things for a better signal when I was a teenager.

Robotic Radio
What is "Robotic Radio?" In short, Robotic Radio is anything in our hobby that was done in the past

by a human operator, but is now done automatically by a "robot." Consider several examples:

- An automatic antenna was one of the first robotic radio accessories to become widely used. It matches the impedance of the transmitter to the antenna automatically, robotically. The operator used to adjust the variable capacitors and inductors manually to accomplish the same task.
- Consider Steppir antennas. In the old days, I used to adjust the length of my antennas manually. Now with Steppir antennas they are robotically adjusted based on the frequency of the transceiver.
- Linradxvixvi, an open source software defined radio project, has the capability of digitally phasing two orthogonally polarized antennas (and their associated SDR receivers) automatically to match the polarization of the incoming signal, and for EME, it also calculates the best corresponding polarization to use on transmit.
- In the old days, I used the "Armstrong" method of antenna rotation. Then I upgraded to a regular antenna rotor. That brought my antenna rotation from manual to remote controlled operation, because I still needed to control the rotor. Robotic antenna rotation is more where the antenna automatically turns to the best position, perhaps based on the location of the station of interest, or maybe on the best signal to noise ratio. Many rotators can be controlled via the serial port of a PC which easily leads to a robotic rotator.
- The ultimate in "Robotic Radio" is where the operator does nothing; the radio is completely autonomous. This can be controversial when transmission is involved, because sometimes robotic operators are not as considerate or capable as human ones. In this paper I advocate for robotic receivers, and leave the controversy for others to address.

Note, that while there is overlap with Cognitive Radio[i], they are not the same. Cognitive Radio describes a radio system which adapts to its environment automatically to improve communications. Robotic Radio describes the situation where the radio does the things the operator used to do autonomously. Because robotics is such an intense interest of today's youth, I believe Robotic Radio has more potential to draw them into amateur radio than Cognitive Radio.

What can we do with "Robotic Radio?" Philip Gladstone figured that out when he set up his PSK Reporter web site.[ii] In conjunction with Fldigi[iii] or Digital Master 780[iv] it collects data from receivers monitoring the PSK HF bands. That data is plotted using Google Maps[v] so anyone can see what the different robotic receivers have heard in the last day. The log of different countries heard is kept for a week, and you can compare how many spots, countries, etc. your robotic receiver has with everyone else's. Philip Gladstone says, "This started out as a project to automatically gather reception records of PSK activity and then make those records available in near real time to interested parties — typically the amateur who initiated the communication. The way that it works is that many amateurs will run a client that will monitor received traffic for call signs (the pattern 'de call sign call sign') and, when seen, will report this fact. This is of interest to the amateur who transmitted and they will be able to see where their signal was received. The pattern chosen is typically part of a standard CQ call. The duplicate check is to make sure that the call sign is not corrupted."

"The way that this would be used is that an amateur would call CQ and could then (within a few minutes) see where his signal was received. This can be useful in determining propagation conditions or in adjusting antenna and/or radio parameters. It will also provide an archive of reception records that can be used for research purposes."

"Many of the monitoring stations like to use this for bragging rights. It is also interesting to see how long it takes to spot 100 different countries. (A well placed station with a decent antenna can do this

within a week of monitoring, but it takes more than a day)."

"The data being gathered also includes more than just PSK spots, though these are in the majority at the moment."[vi] WSPRnet[vii] has a similar, though not quite as sophisticated spot reporting system specifically for the WSPR modes.

Horse Racing

I find the PSK Reporter web site interesting and useful for "horse racing" antennas and modems to determine how tweaks really work in the wild. Imagine a new kind of contest, where robotic receivers are used to spot transmitting amateur stations. There would be no requirement for a amateur radio license, but it could easily lead to one. There would be no need to spend the entire weekend handing out signal reports. The robots would do that while you are enjoying time with your family. You could even combine this type of contest with ordinary type; points could be applied to transmitting amateurs based on how may robots heard them on different bands and vice versa.

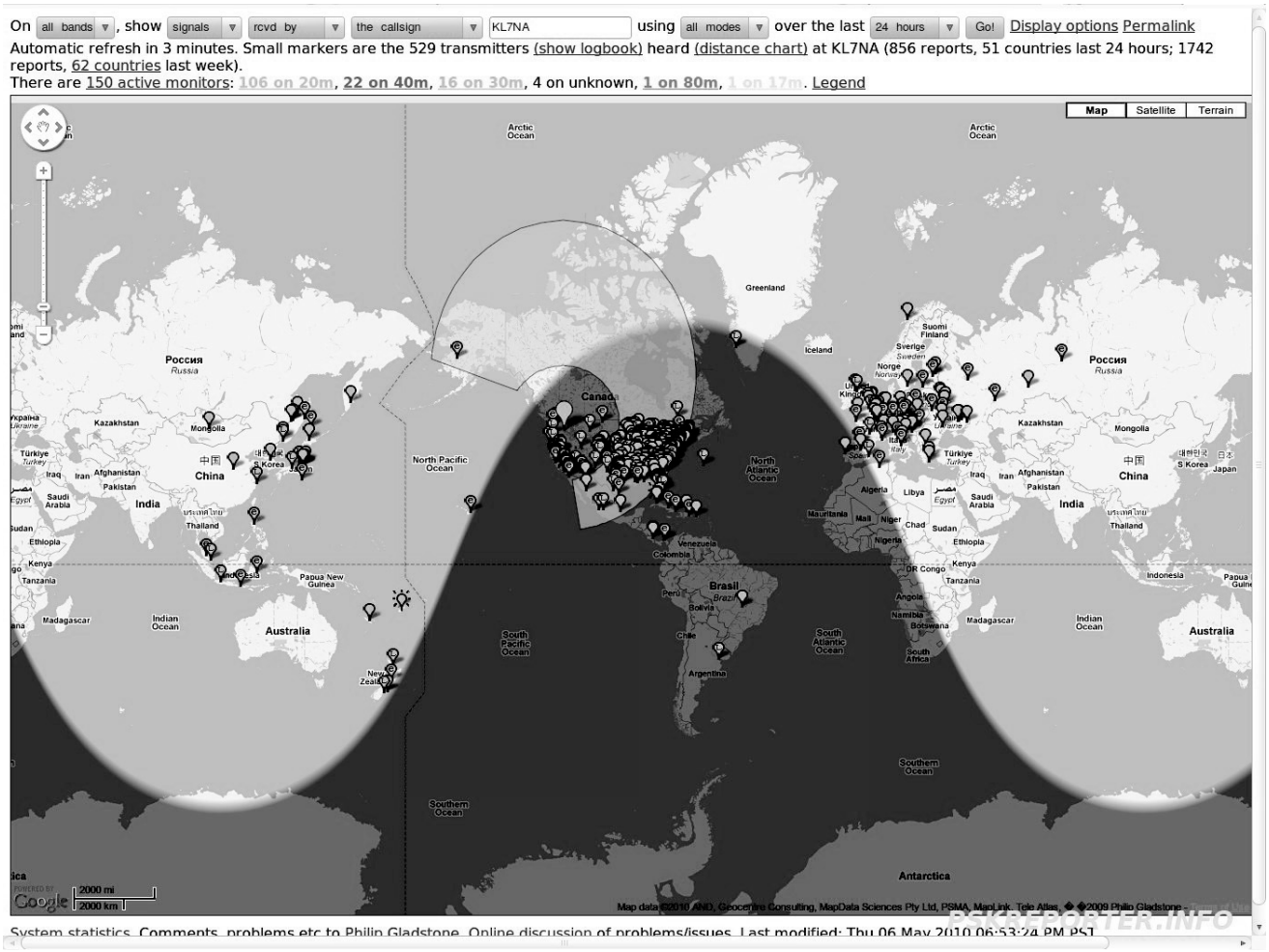

Illustration 1: Spots on July 14, 2010 for KL7NA on 20 Meters

I played around with my station as a robotic receiver, using a four element mono-band Yagi at 100 feet on 20 meters, for a few weeks, rotating the antenna twice a day to the direction where the best DX was likely to be coming from. It was interesting to note that despite the reciprocity theorem of antenna theory, optimizations done for transmit, didn't necessarily help the same way they would in normal amateur operation. My station was easily beat out by stations using simple non-directional antennas having very quiet locations on the East Coast or Europe. My noise level is around S2. I did notice that I could hear more long haul DX than most spotting stations, and that the East Coast was a much better location for spotting a large number of countries than the West Coast, and Europe was even better yet. I think this could become a fun activity for high school teams to compete in, and it wouldn't necessarily increase the QRM on the bands, the way conventional contests do.

Perhaps Robotic Radio contests will someday inspire contestants to go out and locate those arcing insulators and other noisy power line equipment for the power company, and actually improve our electromagnetic environment. It could inspire young people to become involved in software development for robotic radio, and they would learn a skill that is in high demand. I find that the best programmers in my classes are always those who have done it as a hobby.

Robotic Radio is Easy and the Possible Experiments are Many

It is inexpensive and easy to set up a receiving station for Robotic Radio. Almost every school kid has a computer with some kind of sound card, and an Internet connection. For $20.00 they can purchase a Softrock Lite II Receiver Kit for any band from Tony Parks.xiixii Building the Softrock is not difficult

and will teach the builder some useful soldering skills. Any old wire could be used as an antenna. I recommend Ubuntu on the computer with DttSP, SDR-Shell (or another combination of SDR software) and Fldigi, and they are ready to go. If Windows is required, they can install PowerSDR and DigitalMaster780 instead.

There are lots of things that can be optimized, or done to improve, or with a Robotic Radio receiver. The antenna can be improved. The software can be improved. The noise environment can be improved. The number of frequencies covered can be increased. The QTH is important as well. Here are some specific ideas:

- Several receivers could be connected up each to its own antenna. The signals can be combined in software to have an array that listens simultaneously with high directivity in a number of different directions, for example see what K1LT[viii] did on 160 meters or read this beam forming primer by Toby Haynes[ix].
- You could use Linradxvi with its two antenna capability to automatically phase two SDR receivers for the best signal strength. You could make that happen for each individual signal independently.
- You could have the software use different AGC values for each signal. (Presently, I note when a loud station comes on anywhere in the 3 KHz PSK band, all the other signals are attenuated by the AGC on my Kenwood TS-850SAT. I actually think DttSP[x] already does this, so a twenty dollar Softrock might work better than my good old Kenwood.)
- Using Linrad and two orthogonally polarized antennas, you could make a receiver that would record the incoming polarization of each signal received. Have you ever wondered how much you could gain by polarization matching on HF? I have, and my experiments indicate that sometimes it is a lot, and if you need to pick between horizontal and vertical, pick horizontal.
- You could record all the data coming from the CW numbers stations[xi] used by spies, and try and break their codes.
- You can share what your receive with the whole world using WebSDR.
- You could take a bunch of different WebSDR receivers and correlate and combine what they receive, and even use them as a giant networked receiver, the Robotic Receiver of the World!
- You could have your Robotic Receiver flash the screen when it spotted a new one, set your transceiver to the correct frequency and wait for your command to call. You might get your DXCC in a week, without hours of listening. Oh my, maybe that's cheating!

Those who find most fun in designing, engineering and building enjoy Robotic Radio perhaps more than those who simply enjoy operating. I enjoy them both.

Building Blocks for Robotic Radio

Robots are complicated systems made up of many individual subsystems. These subsystems can be thought of as building blocks. If you have played with Legos you will know that understanding what blocks are available and how they connect is necessary if you are going to build anything much. This section is to introduce some of the blocks I have found useful when playing with Robotic Radio using a PC running Ubuntu Linux. It isn't a comprehensive list of everything, just my favorites.

- SDR Hardware
  - Softrock[xii] radios: Tony Parks, KB9YIG, sells quality inexpensive kits. These are my

favorite, because of their simplicity, performance and especially price.
    - High Performance Software Defined Radio[xiii] (HPSDR): TAPR sells circuit boards and built boards that are the hardware building blocks for truly high performance software defined radios.
- Hamlib[xiv] provides a consistent interface to the controls of many different rigs for programmers wanting to incorporate radio control into their programs.
- Jack Audio Connection Kit[xv] is a system for routing audio signals from sources to sinks . It operates like a big virtual patch bay for audio signals in the computer, and is indispensable for me.
    - Jack Control (qjackctl) is a graphical user interface (GUI) to control and connect Jack.
    - Jack Audio Analyzer (jaaa) is a tool to see the input and output of any audio port in Jack. It is your virtual oscilloscope and spectrum analyzer.
- DttSPx (sdr-core on Linux) is the digital processing software used by the Flex Radio products. It accepts commands, serves spectrum and signal strength data through the network using the User Datagram Protocol (UDP); audio usually routes through Jack. This software is well written and very efficient in terms of resources consumed. It was written by Bob McGwier, N4HY, and Frank Brickle, AB2KT. It is written in C. It is my favorite SDR radio processor.
    - SDR-Shell is a GUI for DttSP that uses the QT C++ framework.
    - nc is a command line tool that can control sdr-core over the network and even from the same machine.
    - GHPSDR3 is a GUI that uses DttSP and works with HPSDR as well as other transceivers and receivers. It is made up of a server and client that communicate via the network. The client can be on the same computer as the server.
- Linrad[xvi] is SDR software that the earth-moon-earth (EME) operators prefer. It includes both the DSP algorithms and the GUI.
- Quisk[xvii] is another SDR that includes DSP and GUI. It is written in Python and C.
- Gnuradio[xviii] is a set of signal processing blocks that are tailored to software defined radio. Using Gnu Radio Companion (grc) you can connect these blocks, making a block diagram of your system in a graphical user interface window. These blocks operate in real time, and you can use Jack for input and output. It is written in Python and C++.
- Octave[xix] is a higher level (generally not real time) programming and simulation environment that is really handy for debugging DSP problems because of its wonderful matrix data manipulation capabilities. It aims to be compatible with the more expensive closed source program, MATLAB.
- Fldigi1010iii is a wonderful digital modem program with a very nice GUI. It demodulates around sixty different digital modes, and is written in C++ using the Fast Light Tool Kit.
- WebSDR[xx] is shared radio receiver that makes available on the world wide web the audio, spectrum and signal strength of receivers located around the world. It can be used by a large number of simultaneous users, and was written by Pieter-Tjerk De Boer, PA3FWM. WebSDR can provide receivers for those with Internet access and antenna restrictions. At the time of this writing, it is still not distrubuted under an open source license, because the author hasn't found a suitable open source license for this web based server application. If the client is running Ubuntu 10.04, you can use the pulseaudio module-jack-sink and module-jack-source[xxi] to route audio from your browser (or any other pulseaudio source or sink) to and/or from Jack, and from there on to your robotic radio receiver application.
    - Jack Fifo[xxii] (jack-fifo) feeds programs that normally use the OSS /dev/dsp interface with audio from Jack. It is tailored to the WebSDR server and allows the same audio that is

feeding our WebSDR[xxiii] server to feed sdr-core and fldigi so the same server can send spots to the PSKReporter web site simultaneously.

The building blocks I have listed here are open source, except WebSDR, and all run under Linux (and sometimes on other operating systems).

There are also closed source programs that are useful in Robotic Radio; one of the most interesting is authored by Alex Shovkoplyas, VE3NEA, and is known as CW Skimmer.[xxiv] CW Skimmer is a multi-channel CW decoder and analyzer. When used with a wide-band SDR, it will decode up to 700 CW signals simultaneously. Alex also has a server incarnation with a standard Telnet interface to the loggers and cluster monitors, for spotting purposes. CW Skimmer is a mature product with a lot of features. However, for me and a lot of other experimenters, purchasing a closed source program is like buying a radio without a schematic. It makes it too difficult to experiment with, and experimenting is the principle thing I would want to do with it. Linux is my operating system of choice and CW Skimmer is a Windows only program. So CW Skimmer in particular, doesn't work for me, but the idea does.

CW Skimmer isn't the only program of its type. Back in 1991, one of my engineering students, Jeff McDow, wrote a program that would decode simultaneous CW signals.[xxv] It ran on an AT&T DSP board that plugged into a slot in a personal computer. Agilent Technologies and others sell software defined radios with similar capabilities to the NSA, CIA and other organizations. All of these solutions I could find were also closed source and expensive, or as in Jeff's case, rather ancient.

I decided we need a similar, but open source (GPL[xxvi]) program that runs on Linux, so I assigned the task to my Communications class this last spring. They started it, but unfortunately didn't have anything working by the end of the term. I decided to push it along to at least a point where it works. Software like this is never really finished; the longer you play with it, the more new interesting and useful things you think of including, or perhaps better ways of doing it lure you along. I call this new software CW Robot[xxvii].

CW Robot

CW Robot was made with the idea of using a Softrock, HPSDR or similar receiver and one of the above SDR programs. Let f_s be the sample rate of the system. Using a zero IF, the two channels (I and Q) can be demodulated into two independent baseband signals, one containing the upper sideband from DC to $f_s/2$, translated to the band between DC and $f_s/2$, and the other containing the lower sideband, from $-f_s/2$ to DC, translated and swapped up to $f_s/2$ to DC. This is essentially independent sideband (ISB). Together these two baseband signals represent an RF band of frequencies with bandwidth f_s. All of the CW signals in this band are to be decoded. With an ordinary sound card this could be an entire band of CW signals.

The Fast Fourier Transform (FFT) is used to split each of the two baseband signals above into sub-bands, often known as bins, that are small enough to assume they each contain at most one CW signal. For each sub-band with a signal, that signal is decoded.

Since the audio for CW Robot comes from a program that uses Jack, CW Robot needed to be a Jack client as well. The demodulated output is dumped into text files, one for each sub-band or FFT bin. Eventually there would be a GUI similar to the PSK Viewer window in Fldigi. Spots would be generated and transmitted to the PSKReporter web site. It was further envisioned that it would be used in conjunction with WebSDR to decode signals so that users on the Internet could read the mail on any

CW signal being received. Spots and server code would likely be separate programs, keeping the building blocks so that they could easily be reused for other Robotic Radio purposes.

Eventually the detector would be optimized to use matched filters and a maximum a posteriori (MAP) detector,[xxviii] though for a first cut we would use the decoding part of the single channel CW demodulator program Demorse[xxix], written by Neoklis Kyriazis, 5B4AZ. We decided to use the Fastest Fourier Transform in the West[xxx] (FFTW) to do the FFT, and to characterize the noise, we used the Gnu Scientific Library[xxxi] (GSL) histogram tools.

The "back of the envelope" calculations below suggested this approach might work.

Assume we would like at least 1/10 of a dit time for each FFT. Each of these we will call a fragment. That should be good enough timing resolution to determine where the dits and dahs are. Assume that something around 40 words per minute is a reasonable top speed that must meet this requirement. (Higher speeds will have lower resolution measurements of the dits and dahs.) Suppose we are sampling as f_s = 96 KHz = 1/T, where T is the sample interval in seconds. There are fifty dits per calibrated word[xxxii] in Morse Code, so:

$$\frac{40\,words}{minute} \times \frac{50\,dits}{word} \times \frac{1\,minute}{60\,seconds} = \frac{33.3\,dits}{second}$$

That gives $(1/33.33)\frac{seconds}{dit} = 30\,milliseconds$ and so a tenth of a dit and is 3 milliseconds. That is equal to the time for each FFT which is NT where N is the number of points in the FFT and $T = 1/fs = \frac{1}{96000} seconds$. This gives N of about 288. Let's use 256, which would correspond to 45 wpm because the FFTs can be computed faster for N being a power of two. The bin bandwidth for each FFT sub-band is 1/(NT) which is 375 Hz for the 256 point FFT. This seems somewhat reasonable, at least for a starting point. Use of a matched filter will bring this bandwidth down significantly, and I had some ideas on how to extend this method into a matched filter. The power in the FFT bins would be used to determine a threshold, above which the fragment would be considered to come from a mark (carrier present) and below which it would be considered a space (no carrier present).

Demorse used a similar scheme, but instead of filtering out each individual sub-band with the FFT, it only filtered out one band around 500 Hz using the Goertzel algorithm[xxxiii]. The fragments were put together to build up dits, dahs, character and word spaces, and to determine if the speed is set correctly for each FFT bin. The dits and dahs are encoded into words where bits (1 representing dit) and (0 representing dah) are left shifted into a word initialized to 0x01. This word is then decoded into a character which is printed to the appropriate output file. Our FFT scheme meshed well with Demorse's so we were able to re-use portions of the Demorse program in CW Robot version 0.1.

This method isn't nearly as good as a matched filter with MAP detector, because it assumes that each fragment is correct, but errors do occur. Normally, when a fragment is in a certain state, the next fragment is much more likely to be in that same state than in the other. To incorporate this information, a little hysteresis is introduced to try and inhibit changing briefly from mark (carrier on) to space (carrier off) in the middle of a dit or dah, or changing briefly from space to mark in a space. This helps.

CW Robot is still brand new alpha software under active development, and it does work, but it is no match for CW Skimmer, yet. In order for that to happen, we need to incorporate a matched filter MAP detector, and probably a number of other improvements and tweaks. This is the fun part, tweaking things for improvements. Doing it will increase our skill with software development and signal processing. If you would like to join the effort, we would love to have your help.

In Reflection

Robotic Radio is an exciting new area of amateur radio that has a lot of potential to keep our young people interested in amateur radio, software development, and science and engineering. It is just plain fun to watch your robotic radio station reporting the different countries it has heard. There are so many interesting projects in the area of Robotic Radio. Get involved! Hook up a robotic receiver to spot for us all. Work with the ARRL to develop a contest for robotic receivers. Get a young person involved with Robotic Radio. Demo it in your local schools. Put up your own WebSDR node. Help with hardware development. Read software; write software; write documentation. Help with CW Robot. Experiment! Share your results at the next DCC!

i http://en.wikipedia.org/wiki/Cognitive_radio
ii http://psk.gladstonefamily.net/pskmap.html
iii http://www.w1hkj.com/Fldigi.html
iv http://www.ham-radio-deluxe.com/Programs/DigitalMaster780.aspx
v http://maps.google.com/
vi http://pskreporter.info/
vii http://wsprnet.org
viii http://k1lt.com/
ix http://www.spectrumsignal.com/publications/beamform_primer.pdf
x http://dttsp.org/
xi http://www.spynumbers.com/
xii http://kb9yig.com Note, that when he has new kits available it is like a pile up to order them, so most of the time he is sold out. The Softrock 40 Yahoo Group (http://groups.yahoo.com/group/softrock40/) is where you find out when radios are available and discuss them.
xiii http://openhpsdr.org/
xiv http://hamlib.org
xv http://jackaudio.org/
xvi http://www.sm5bsz.com/linuxdsp/linrad.htm
xvii http://james.ahlstrom.name/quisk/
xviii http://gnuradio.org/
xix http://www.gnu.org/software/octave/
xx http://websdr.org
xxi http://www.pulseaudio.org/wiki/Modules#module-jack-sink
xxii http://people.wallawalla.edu/~Rob.Frohne/jack-fifo/jack-fifo.tar.gz
xxiii http://outside.wallawalla.edu:8901/
xxiv http://www.dxatlas.com/CwSkimmer/
xxv http://people.wallawalla.edu/~Rob.Frohne/ClassHandouts/Communications/Jeff_McDow.pdf
xxvi Gnu Public License http://www.gnu.org/licenses/gpl.html
xxvii http://people.wallawalla.edu/~Rob.Frohne/CW-Robot/cw-robot.tar.gz
xxviii Matched Filter and Maximum a Posteriori Detector for Amateur Radio Digital Modes, Rob Frohne, KL7NA and Mark Priddy, KE7UFF, TAPR/ARRL DCC 2009. http://www.tapr.org/pdf/DCC2009-MatchedFilterandPosterioriDetector-KL7NA-KE7UFF.pdf
xxix http://5b4az.chronos.org.uk/pages/morse.html
xxx http://www.fftw.org/
xxxi http://www.gnu.org/software/gsl/
xxxii http://en.wikipedia.org/wiki/Morse_code
xxxiii http://en.wikipedia.org/wiki/Goertzel_algorithm