Internet Remote LF Receiver

A. Maitland Bottoms, AA4HS

Abstract— **Some AMRAD members are controlling and listening to a radio receiver via their Internet connections. The receiver in question covers the LF and HF bands and uses an active antenna described in the September 2001 QST magazine article. This article outlines the digital and network aspects of the project. The specifications and source code of the entire system are available, and the reader is both permitted and expected to tailor them to suit his own tasks.**

Keywords— **AMRAD, radio, RX-320, Ogg Vorbis, LF, remote**

I. Introduction

RADIO puts you in touch with far-away places. There is the content of radio transmissions to listen to, but also - especially with the AM and sideband modes - a background environment of natural and man-made phenomena. Us hobbyists are fans not just of what is being transmitted, but also of how it happens. We come up with questions to ask each other: "What does that sound like to you?", "Is that noise something near me? Should I move the antenna?", "How well is your receiver working?".

I live in an urban area. When I listen to my radio, especially in the LF band, the dominant feature is man-made noise. When I travel, I find reception is often better away from here. So now I have a radio 100 miles from my house, and I control it and listen to it over the Internet. I can easily share it with my friends, and do so without lugging equipment and driving for hours.

A. Hardware requirements

A radio. A computer. The computer has to connect to the radio: serial port and sound card. The computer should connect to a network: a dialup modem probably won't make you happy, but DSL and cable modem and ISDN connections will. Listening on your local LAN will work very well. Computers made since 1998 will likely work fine. 1997 and before computers will be starved for CPU power, and may have lower quality sound card interfaces. (That's for PCs. An SGI or Sparc system will likely work well since 1994.) For the radio I have used an Icom PCR-1000, a Ten-Tec RX-320, and the FM tuner part of my Hauppauge video card.

B. Software requirements

All software used in this project is free. Many people new to free software find themselves confused because the word "free" in the term "free software" is not used the way they expect. To them free means "at no cost". An English dictionary lists almost twenty different meanings for "free". Only one of them is "at no cost". The rest refer to liberty and lack of constraint. When we speak of Free Software, we mean freedom, not price.[1] All specifications and source code used in this project are available, and you are at your liberty to customize it for your own application.

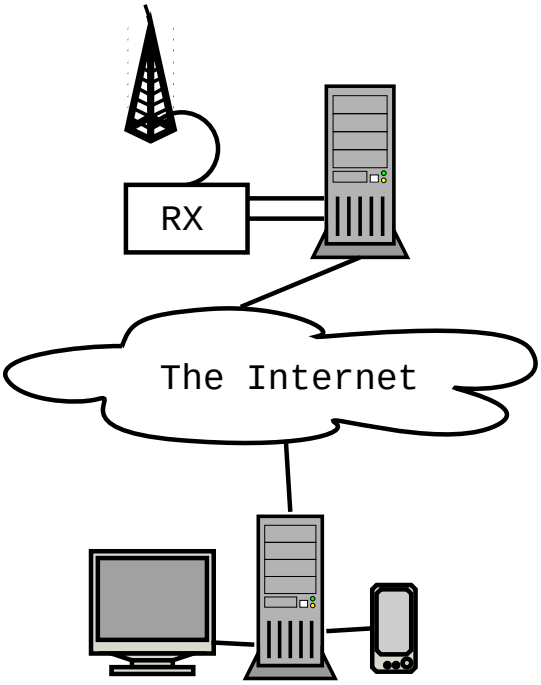

Fig. 1. Internet Radio Concept

C. System assemby

Figure 1 shows the system concept. The remote receiver has a coax cable connection to an antenna (in the reference implementation, the antenna system is the K0BRA active antenna[2]). The receiver is controlled over an RS-232 serial port connection, and received signal audio goes to a computer sound card input. The receiver control computer also handles its own Internet connection, and provides control and digital audio interfaces. It need not have a local display or audio monitor capability. A person equipped with a Internet connected computer with audio capabilities can tune and listen to the remote receiver.

II. Control

Several receivers are available which have computer control. The Ten-Tec RX-320 and the Icom IC-PCR1000 use RS-232 signaling, but the protocol is different. While various control interfaces are available, including Graphical User Interfaces (GUIs), this project focusses on command line tools.

Rather than hack are the GUIs to make them operate remotely, the approach used here is to use command line tools as elements of scripts which are run in response to Web Browser inputs.

Fig. 2. Web Browser Interface

The web form allows setting of frequency, mode, and bandwidth. The setting of audio level is handled by the administrator ahead of time. Also, the web page shows current status of the radio, and provides links to signal strength logs, streaming audio, and other related information.

A. Apache, CGI and Python

The receiver controller runs the Apache web server[3], set up to allow running of "CGI scripts". The web page HTML is dynamically assembled by a script written in the Python[4] language. The script also handles interpreting the form entries submitted from the browser, and calling the command line radio control applications. Configuring Apache is a large task, but provides for access control and logging.

One detail to remember is that the script is run as the user set by the Apache configuration, which generally does not have permission to access the serial ports.

Python was chosen as a straightforward and relatively safe way to handle the web interface. There are built-in libraries that take care of the connection to the Apache server and running command line applications.

B. Radio control

Command line applications for radio control are likely available on the Internet, if not provided by the radio vendors themselves. For the Ten-Tec RX-320 receiver, install the Debian package **rx320**[5]. For the Icom PCR-1000, install the Debian package **icomlib-bin**[6], [7].

III. Audio Transport Methods

When it comes to handling digital audio data, there is more than one way to do it. Some things to consider while deciding are the balance between latency, bandwidth, and processing load. Also, methods scale differently: using a network connection per client will fill available bandwidth faster than the multicast protocols.

A. Network Audio

Some workstations sound software, such as **esd**, the Enlightenment Sound Daemon, have the ability to pass data over TCP network connections. Most often these tools pass uncompressed data, and have both high bandwidth and high latency.

B. Multicast

Great for your local LAN and corporate network. Any number of clients can monitor the data stream, and bandwidth usage is minimized. The **vat** and **rat** tools are both clients and servers for audio multicast data.

C. Speak-Freely

One of the lowest latency tools is **speak-freely**[10], which uses the UDP protocol and can server multiple users on a reflector server and also operate in multicast mode.

D. H.323

H.323 and other related real-time multi-client conferencing protocols might be useful for a remote radio receiver. One free implementation of these protocols is **gnomemeeting**[11].

E. Streaming Audio

"Ogg Vorbis is a completely open, patent-free, professional audio encoding and streaming technology with all the benefits of Open Source."[8] And it is quite suitable for listening to a remote receiver. Combined with an **Icecast2** server, several people can listen to the audio content at the same time, even though they may be far from the receiver and each other. The encoded data from a band-limited RX-320 receiver can easily be heard over a 33.6 kbps dilaup Internet connection. Clients for listening to Ogg Vorbis signals are available on many kinds of computing platforms. (The **xmms** and **winamp** applications handle it.)

Setting up an Ogg Vorbis audio streaming server is not the easiest of tasks. But the BBC managed to put together some systems to serve BBC radio programs over the Internet, so you can do it too.

IV. Logging

Your computer will no doubt be spending more time listening to the radio than you will. What is nice is the ability to check on the activity "while you were out...".

A. Signal Strength

The radio control command line utilities allow the S-meter to be queried. It is straightforward to use this feature as an input to the **Multi-Router Traffic Grapher**[9]. While this tool is ordinarily used to monitor network usage in a packet counting mode, it does also have a meter reading mode.

Figure 3 shows the output of such logging, and the daily variation of propagation shows clearly.

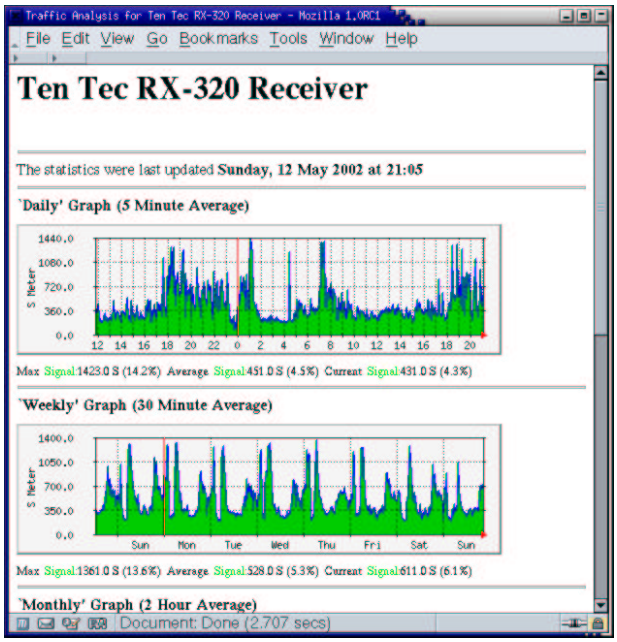

Fig. 3. Signal Strength Logs

B. Audio Archives

Although hard drives with very high capacity are now obtainable at lower cost than ever, there is no compelling need to copy raw samples to the hard disk.

The ogglog.py file shows one approach to audio logging. It is called with a URL for an Ogg Vorbis stream, and a length of time to log. This script can be called as a **cron job** and thus keep a rolling 1 week buffer of recorded audio.

Once again, Python, along with libraries for HTTP protocol, ogg and vorbis data handling, made implementation fast. The first attempt does have some bugs - a few audio samples are lost as one hour's log stops and the next hour's log starts. Hopefully a future version will handle logging without any dropouts.

Note that the logging may, but does not have to, run on the computer directly connected to the radio. One could place an embedded computer system in the radio that provides web-based control and streaming audio output, the use another computer (with lots of hard disk space) to do the actual logging.

V. Signal Processing

Once the audio is in digital form, it is natural to handle it with computer programs rather than analog electric circuits. A few noteworthy programs deserve mention here: **sox**[12], **multimon**[13] and **gnuradio**[14]. Most (if not all) of the PSK31 applications combine the signal processing with a graphical user interface, so using them from a remote system may be awkward.

The **sox** application allows for file format conversion, along with some basic signal processing which includes downsampling, upsampling, high pass and low pass filters.

For decoding digital data, **multimon** application understands the AFSK typically used in amateur FM packet radio, as well as DTMF tone decoding and other useful modes.

Most ambitious is the GNU Radio project. If one has the hardware to sample a band, **gnuradio** will demodulate one or more signals into baseband. Already broadcast FM demodulators are working, as well as a good start on HDTV signal generation and reception.

References

[1] The Debian Project *What Does Free Mean?* http://www.debian.org/intro/free

[2] Frank Gentges, K0BRA *An Active Antenna for LF* QST, ARRL, September 2001

[3] The Apache Project *Apache web server* http://www.apache.org/

[4] Guido Van Rossum, et al. *The Python Programming Language* http://www.python.org/

[5] A. Maitland Bottoms *The rx320 Debian package* http://packages.debian.org/rx320

[6] icomlib authors, et al. *icomlib* upstream URL

[7] A. Maitland Bottoms *The icomlib-bin Debian package* http://packages.debian.org/icomlib-bin

[8] Ogg Vorbis *Icecast streaming audio server* http://www.vorbis.com/

[9] Tobias Oetiker, Dave Rand and others *MRTG: Multi Router Traffic Grapher* http://www.mrtg.org/

[10] John Walker and Brian C. Wiles *Speak Freely* http://

[11] Damien Sandras et al *Gnome Meeting* http://www.gnomemeeting.org/

[12] Lance Norskog And Sundry Contributors *sox: sound exchange* http://packages.debian.org/sox

[13] Tom Sailer HB9JNX/AE4WA *Multimon* http://packages.debian.org/multimon

[14] Eric Blossom, MIT SpectrumWare group, et al *GNU Radio* http://www.gnu.org/software/gnuradio

[15] Frank Gentges, K0BRA *An Amateur LF handbook* ARRL press, 2003

A. Maitland Bottoms has been an AMRAD member since the mid-1980s. Computing experience includes the Xerox-820 single board Z-80 computer and the Motorola 56001 DSP. More recently he has been working as a Debian developer.