Building a Better Demodulator for APRS / AX.25 Packet Radio

Part 2, 9600 Baud

John Langner, WB2OSZ

ROUGH DRAFT – May 17, 2015

⚠

These are not professionals on a closed course.

You CAN try this at home.

In the first part of this series we discussed 1200 baud audio frequency shift keying (AFSK) where we switch between two audio frequencies to represent the logic 0 and 1 states. The mismatch between FM transmitter pre-emphasis and the receiver de-emphasis will cause the amplitudes of the two tones to be different. This makes it more difficult to demodulate them accurately.

9600 baud operation is an entirely different animal. We don't use AFSK. Instead, the RF carrier is shifted between 2 frequencies representing the two logic levels. When the signal comes out of an FM receiver, the audio should be one of two voltage levels. The demodulator only needs to decide which of the two levels is present. In the ideal world, it would look something like this.
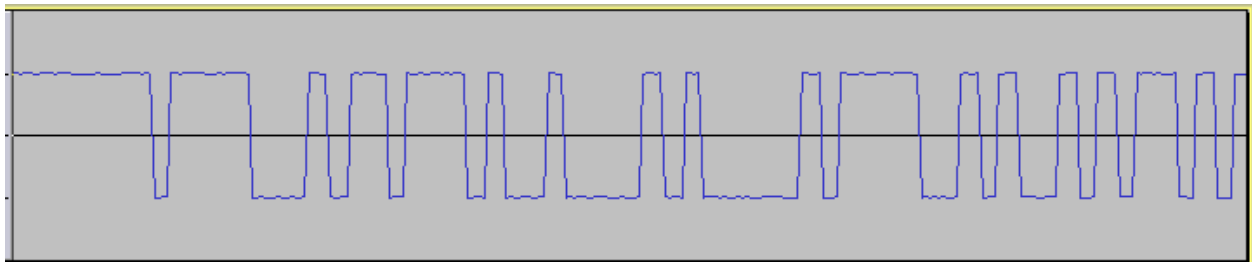

This is easy to decode. If the signal is positive, call it a logic 1. If negative, call it a logic 0.

It's not so pretty in the real world after sending a digital signal over the radio. This is a genuine sample of what we get from a receiver when the signal is fairly strong.
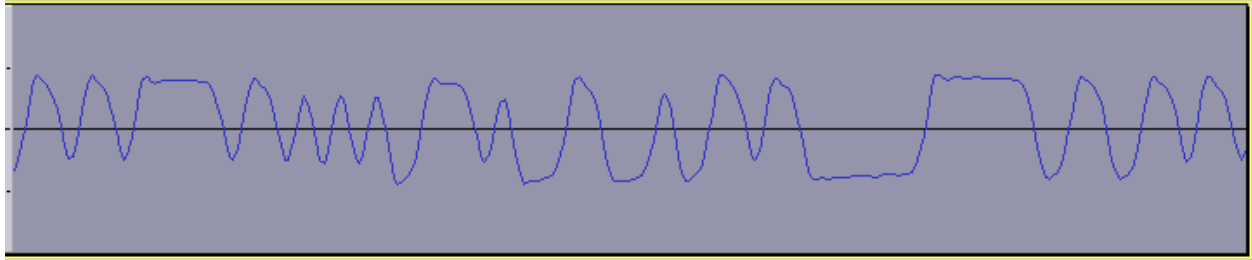
If high frequencies are attenuated, the nice square corners turn into overshoot and ringing. We don't reach the full amplitude when the signal is changing at the maximum rate (e.g. 4800 Hz square wave for 9600 baud). If the low frequencies are attenuated, the horizontal lines droop towards the middle shifting the zero reference level. There are system non-linearities especially if we are not centered in the FM discriminator bandpass. We can never escape the omnipresent noise.

When the signal gets weaker, this is what we get from the receiver.
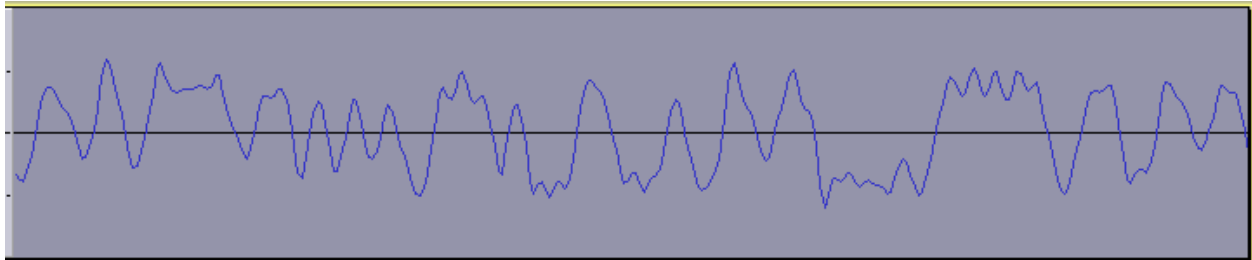
The early 9600 baud modem implementations by K9NG, G3RUH, et al were all done in hardware.

- K9NG   - Need to find link… ARRL Proceedings of the Computer Networking Conferences 1-4 ?
- G3RUH - http://www.amsat.org/amsat/articles/g3ruh/109.html
- KD2BD -http://www.amsat.org/amsat/articles/kd2bd/9k6modem/

They all have a lowpass filter to attenuate high frequency noise while allowing the desired signal through. There are a couple different ways to remove any DC bias. The simple way is to use a DC blocking capacitor on the input. Another is to detect high and low peaks of the signal then set the slicing level around the midpoint.

I took a slightly different approach in software which has a roughly equivalent end result. An automatic gain control normalizes the received signal to the same amplitude regardless of the input amplitude and any DC bias.   The result is compared with zero to obtain the final logic 1 or 0.
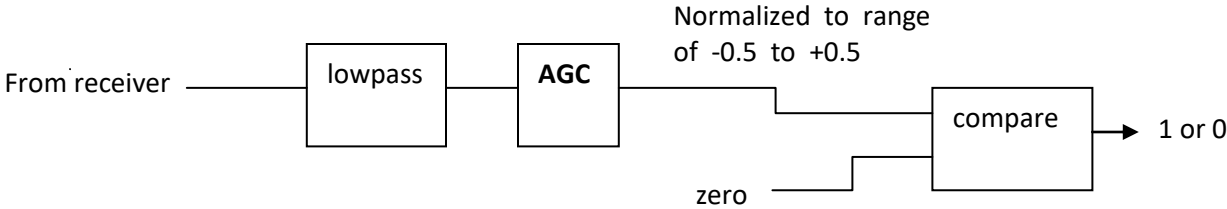
Lacking any indigenous 9600 baud packet activity, I had to create some of my own by walking around the neighborhood with a transmitter, in and out of regions with poor radio coverage to make it more interesting.

"PR9" of the receiver "data" connector was connected to the microphone input on a PC. (See http://wa8lmf.net/miscinfo/MiniDIN6-Packet.pdf for a good description.) The activity was recorded with a command like this:

```
arecord -f S16_LE -r 44100 walkabout9600b.wav
```

It's a poor substitute for the TNC Test CD, but at least it allows repeatable tests from genuine signals.

```
atest -B 9600 walkabout9600b.wav

…
…
DECODED[58] WB2OSZ audio level = 83(77)
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>

DECODED[59] WB2OSZ audio level = 84(78)
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>

59 packets decoded in 35 seconds.
```

The recording is 30 minutes and 44 seconds.   We can process it more than 50 times faster than real time.   It is possible to squeeze out a few more frames by using the single bit fix up option which now works properly for 9600 baud in version 1.2.

```
atest -B 9600 -F 1 walkabout9600b.wav
…
…
63 packets decoded in 54 seconds.
```

The data is put through a scrambler before transmission so you would expect the transmitted signal to look fairly random and symmetrical with the same number of 0 and 1 bits over a longer period. You might expect the optimal slicing position to be right in the middle.

Not knowing what to expect, I did a little experiment. Instead of a single comparator, with a slicing level of zero, I ran a bunch in parallel with other small negative and positive slicing points. Each had its own HDLC decoder and duplicates were removed.

In the screen display, we have the vertical bars and underscores after the audio levels. Each one of these character positions corresponds to a decoder with a slightly different threshold point to decide if we have a logic 1 or 0.

| means a frame was received with a correct CRC.
_   means no frame was received.

We use the "-P +" option to mean multiple slicers. The test starts off pretty boring. With a very strong signal, we all of them decode the signal properly.

```
atest -B 9600 -P + walkabout9600b.wav

DECODED[1] WB2OSZ audio level = 83(77)      ||||||||||
[0] WB2OSZ>TRSW1T:'c0Tl"C[/>"5'}=<0x0d>

DECODED[2] WB2OSZ audio level = 83(77)      ||||||||||
[0] WB2OSZ>TRSW1S:'c0Xl"C[/>"5'}=<0x0d>
```

…

It gets more interesting when the transmitter is on the edges poor coverage areas and the received signal becomes weak.

```
DECODED[30] WB2OSZ audio level = 85(82)      __|||||__
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>

DECODED[31] WB2OSZ audio level = 84(81)      _____||_
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>

DECODED[32] WB2OSZ audio level = 85(79)      ____|____
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>

DECODED[33] WB2OSZ audio level = 84(79)      ____|____
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>

DECODED[34] WB2OSZ audio level = 85(79)      _____|__
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>

DECODED[35] WB2OSZ audio level = 85(78)      ___|||||||
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>
```

…

Notice how smaller numbers of the decoders got a correct result. There is not one that is always better than the others. As the transmitter heads back home, the signal is stronger and decoding is very reliable again.

```
DECODED[65] WB2OSZ audio level = 83(77)      ||||||||||
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>

DECODED[66] WB2OSZ audio level = 84(78)      ||||||||||
[0] WB2OSZ>TRSW0P:'c00l"C[/>"5)}=<0x0d>


66 packets decoded in 55 seconds.
```

When the duplicates are removed, we end up with 66 rather than the previous 59 when always setting the slicing point at exactly zero.

When the number of frames received by each decoder are added up, this is what we find.

Slice At	Frames
-.08	43
-.06	49
-.04	53
-.02	57
0	59
.02	58
.04	61
.06	57

.08   52
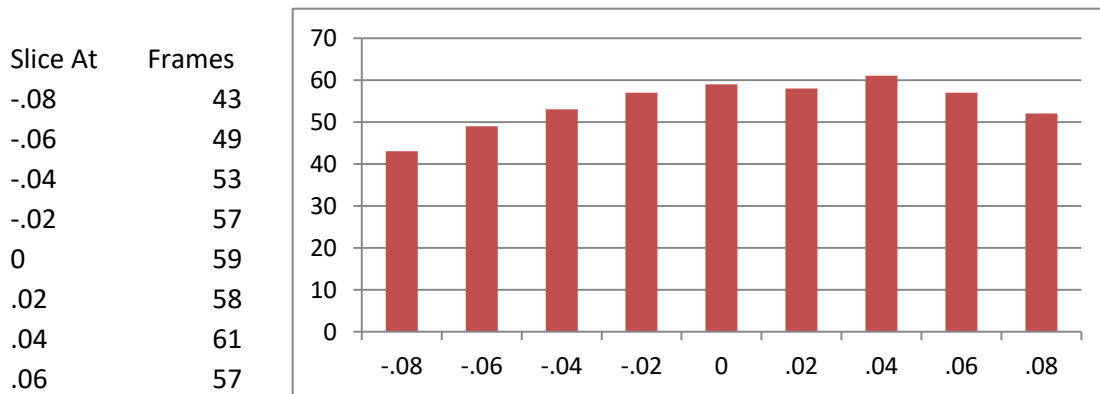
This should not imply that slicing at +0.04 is somehow the optimal point. I think it just happened due to the small sample size.

Final results are:

Number of data slicers	Single bit fix-up	Command Line Options	Time, seconds	Frames Decoded
One	no		35	59
Nine, remove duplicate frames	no	`-P +`	55	66

If we attempt to fix up single bit errors, we get a few more:

Number of data slicers	Single bit fix-up	Command Line Options	Time, seconds	Frames Decoded
One	yes	`-F 1`	54	63
Nine, remove duplicate frames	yes	`-P + -F 1`	256	68

256 seconds might look like a lot but recall that the recording is 1844 seconds so this is still 7 times faster than real-time.

Now let's try a real-time side-by-side comparison.

9600 Baud TNC comparison

Here we compare 9600 baud decoder performance. For this experiment we need:

- Kenwood TM-D710A.
- Software Defined Radio USB dongle. (such as http://www.amazon.com/Receiver-RTL2832U-Compatible-Packages-Guaranteed/dp/B009U7WZCA/ref=pd_cp_e_0)
- Serial communication cable for D710A (http://www.amazon.com/gp/product/B000068OER is a lower cost alternative to the official Kenwood PG-5G) – connect to COM port on control panel.
- Radio data cable with 6 pin mini-DIN connector – same type of connector used for PS/2 keyboard and mouse. The data communications cable from the Kenwood PG-5H package does not appear to be suitable. It uses the PR1 pin. We need the PR9 pin.
- Tee adapter to connect single antenna to two receivers.

Wire up everything as shown below.   In this case, we use an indoor antenna so we can get weak signals with the transmitter only a few blocks away.
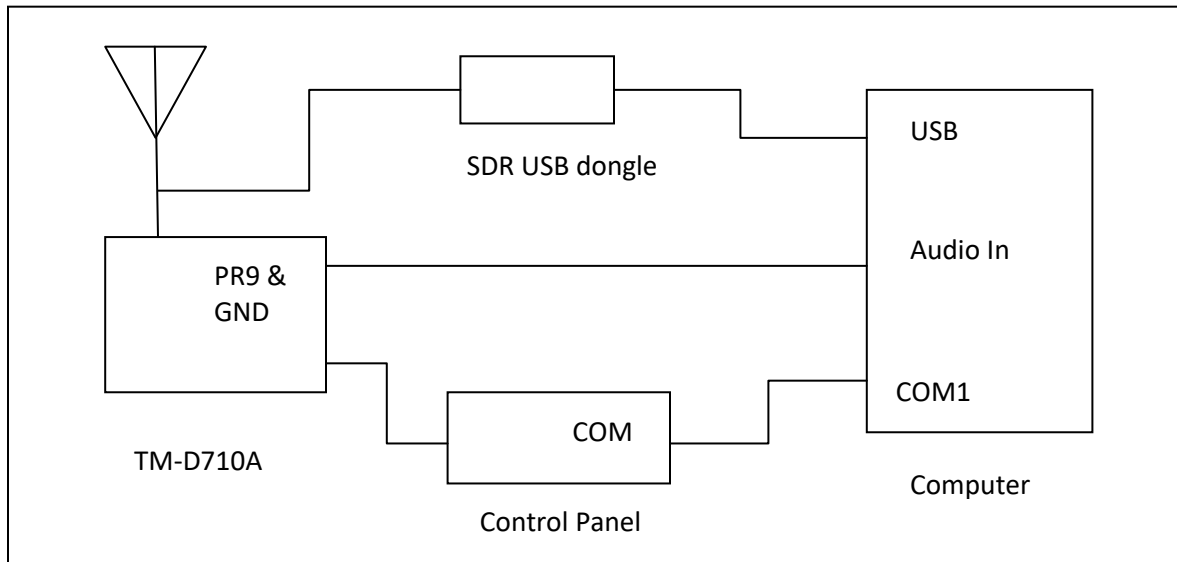
Connect the PR9 pin from the DATA connector on the transceiver to the audio input on the computer. This has wider bandwidth than the PR1 signal or the speaker output.

> Don't waste your time attempting 9600 baud operation with the Microphone and Speaker connections of the radio. They do not have the necessary bandwidth.
>
> Most modern mobile rigs have a special "data" connector which is intended for use with external modems.

After many days of listening, no indigenous 9600 baud activity was heard so I had to generate my own by walking around the neighborhood with a Kenwood TH-D72A transmitting beacons at the maximum rate (every 0.2 minute) at EL power.

Prepare D710A

Tune to 144.99 MHz. Use the TNC button on the control panel to select "PACKET" (not APRS) mode. Enable the COM port with menu 604.

Using some sort of serial port terminal emulator application, such as minicom, connect to /dev/ttyS0. Disable any sort of digipeater settings or beaconing so it is not distracted by trying to transmit. We also don't want to fry the SDR USB dongle! If the control panel shows PACKET12, change the speed by typing this command:

```
HBAUD 9600
```

Enable monitoring:

```
MONITOR ON
```

Exit from the terminal application.

Prepare Dire Wolf, first instance

Be sure to use Dire Wolf version 1.2 or later. In this test we are using Linux and the system board "soundcard" which is the default audio device. Create a configuration file, direwolf.conf-96 with the following:

```
CHANNEL 0
MODEM 9600
FIX_BITS 0
```

9600 baud defaults to N9GH/G3RUH style encoding. The default of "FIX_BITS 1" is turned off so we get a fair apples-to-apples comparison.

Start up one instance of Dire Wolf like this.

```
direwolf -c direwolf.conf-96
```

Prepare Dire Wolf, second instance

This one will be using an SDR dongle rather than a sound card. Prepare another configuration file, direwolf.conf-sdr, with the following:

```
        CHANNEL 0
        FIX_BITS 0
        AGWPORT 8002
        KISSPORT 8003
```

It is not necessary to set the modem characteristics because this will be done on the command line.

Start up the SDR and Dire Wolf in a single command line like this:

```
rtl_fm -f 144.982M -o 4 -s 48000 | direwolf -c direwolf.conf-sdr -r 48000 -B 9600 -
```

Note how we use the command line to specify the audio input device (- at the end) and data rate (-B 9600). Both applications must use 1 audio channel and the same sample rate (48000).

The astute reader might notice the strange frequency **144.982** when we are using 144.99. It seems the SDR frequency is off a little. This was necessary to get similar + and – swings around the center frequency. Your results might be different.

```
        WB2OSZ audio level = 142(+130/-131)   [NONE]   |||||||||
```

When **144.99** was used, we see an imbalance like this:

```
        WB2OSZ audio level = 144(+98/-163)   [NONE]   ||||_____
```

When tuned too far the other way, at **144.977**, we see:

```
        WB2OSZ audio level = 140(+149/-109)   [NONE]   ____|||||
```

IMPORTANT POINT:

The position of the vertical bars and underscores have much different meanings for 300, 1200, and 9600 baud operation.

- 300 Baud HF SSB, multiple tone pairs.   (config file: MODEM 300 5@40)

    Here we have 5 completely separate demodulators with different tone pairs. Each of the decoded signal positions corresponds to a different tone pair. Mistuning of an HF SSB signal will shift the audio frequencies up or down. The position in the middle indicates proper tuning. Positions toward the left mean the audio tones are too low.

- 1200 Baud AFSK, multiple gains for "space" tone.   (config file: MODEM 1200 E+)

    The character positions correspond to different gain ratios between the mark and space filters. Toward the right means more gain for the space tone.

- 9600 Baud K9NG/G3RUH, multiple slicing levels.   (config file: MODEM 9600 +)

In this case, it becomes a tuning indicator. The automatic gain control (AGC) should remove any DC bias. I think this effect is due to discriminator non linearity on the edges but need to study this in more detail.

Compare them.

Start up a recording so we can go back and try more experiments with this data at a later time.

```
arecord -f S16_LE -r 44100 walkabout9600c.wav
```

Run the "aclients" application with command line arguments like this

```
aclients  /dev/ttyS0=D710  8000=DireWolf-soundcard  8002=DireWolf-SDR
```

```
john@hamshack:~$ aclients  /dev/ttyS0=D710  8000=DireWolf-soundcard  8002=DireWolf-SDR

Client 0 now connected to D710 on /dev/ttyS0
Client 1 now connected to DireWolf-soundcard on localhost (127.0.0.1), port 8000
Client 2 now connected to DireWolf-SDR on localhost (127.0.0.1), port 8002
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=

Totals after 2 minutes, D710 9, DireWolf-soundcard 9, DireWolf-SDR 9
```

It starts off with all receiving the same thing while the transmitter is near.

```
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
                                        WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=

Totals after 4 minutes, D710 16, DireWolf-soundcard 17, DireWolf-SDR 14
```

As the signal gets weaker, the left and right columns start missing some that are in the middle column.

```
                                        WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=         WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
                                        WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
                                        [ b  Ps>    E  P QR  A . o H

Totals after 6 minutes, D710 16, DireWolf-soundcard 20, DireWolf-SDR 15

                                        lrm + .     c5 $
                                        WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
```

What happened there? It looks like garbage. When receiving 9600 baud we occasionally see random noise that just happens to have a valid CRC. Remember that there are only 65536 possible values for the CRC and sometimes random noise will just happen to match. Below is what it looked like in the first receiving window. At the end of this test we will subtract these from the totals.

```
audio level = 104(+142/-151)     ___|_____
[0.3]
<0x84>[<0x14>b<0x14><0xb9>Ps<0xbd><0x98><0xaa><0xb1>E<0xe6><0xc5>P<0xa0>QR<0x93><0xf
b>A<0xd7>.<0xfe>o<0xe3>H<0xb5>

 audio level = 105(+142/-158)     _____|
[0.8]
lrm<0xb3>+<0xab>.<0x82><0x07><0xd3><0xf2><0x9e>c5<0x80>$<0x93><0xe3><0x11><0x86>
```

Totals after 8 minutes, D710 16, DireWolf-soundcard 22, DireWolf-SDR 15

Totals after 10 minutes, D710 16, DireWolf-soundcard 22, DireWolf-SDR 15

Totals after 12 minutes, D710 16, DireWolf-soundcard 22, DireWolf-SDR 15

                        WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=

The transmitter was out of range for a few minutes. When it starts to come back in range, the middle is the first to start receiving it properly.

Totals after 14 minutes, D710 16, DireWolf-soundcard 23, DireWolf-SDR 15

WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=

Totals after 16 minutes, D710 18, DireWolf-soundcard 25, DireWolf-SDR 15

WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=

Totals after 18 minutes, D710 21, DireWolf-soundcard 28, DireWolf-SDR 15

WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
                                             YD p bP   && NU 8y ^k   2 I
QmW

Once again, random noise just happened to have a valid CRC. We will subtract this from the final totals. This is what it looked like in the receiving window:

```
audio level = 180(+204/-196)     _____|
[0.8]
YD<0x97><0x87>p<0xd3>bP<0xb2><0xa0><0x96><0xda>&&<0xb0><0xdc>NU<0x1e>8y<0xbc>^k<0x89>
<0x0c><0xd6><0xe2>2<0x16>I<0x01><0xda><0x82>QmW<0xf0><0xd8>gE<0xf4><0xae><0xfc>.<0xbf
><0xd9><0xb3><0xe5><0xe8><0x9e><0xe6>0<0xe1><0xc0><0x0b><0x15>"<0xe5>A
```

Totals after 20 minutes, D710 22, DireWolf-soundcard 29, DireWolf-SDR 16

WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
                                        WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=   WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=

Totals after 22 minutes, D710 28, DireWolf-soundcard 36, DireWolf-SDR 18

```
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=

Totals after 24 minutes, D710 37, DireWolf-soundcard 45, DireWolf-SDR 27

WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
WB2OSZ>TRSW1S <UI R>:'c0ol#'[/>"4j}=    WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=          WB2OSZ>TRSW1S:'c0ol#'[/>"4j}=
```

As the transmitter returns home, all three can hear it properly.

Total scores, subtracting frames that look like garbage:

37 + 6 = 43   45 -2 + 6 = 49   27 – 1 + 6 = 32

This is not the most realistic test scenario – only one transmitter is involved - but it does provide useful data for comparison.

Results

If we give the highest number a score of 100% and scale the others proportionally, the scores are:

- 100%   Dire Wolf, audio input on system board.
- 88%   Kenwood TM-D710A
- 65%   Dire Wolf, software defined radio (SDR) adapter.

The exact percentages should not be taken too seriously because they can be manipulated by the amount of time spent in the zone where Dire Wolf could receive the signal but the TNC inside the TM-D710A could not.

The software defined radio did pretty well when you consider that it costs only about $22.

There is potential for possible improvement by:

- Finding better **rtl_fm** configuration options.
- Using higher quality hardware such as the FUNcube Pro dongle.
- Using other SDR software such as gqrx.

This experiment applies only to 9600 baud operation. Results with 1200 baud could be much different.

There is one remaining mystery. Why doesn't the data change? When walking around, the GPS location should change and the beacon content should change. What is going on here? I don't know. After getting an initial GPS fix, the HT was inside for quite a while where it would lose the GPS signal. Did the radio decide to shut off the GPS and use the last known location even after going outside and walking around?