



# Dire Wolf Software TNC

John Langner, WB2OSZ  
PART of Westford, September 18, 2018



## Topics

- History: What is a TNC?
- Main Theme: Replace old hardware TNC with only software.
- Building a better demodulator.

If you do a web search for an introduction to APRS or Packet Radio, you will find a lot of information.

Most of it is outdated.

Many of the “how to get started” guides tell you to spend a couple hundred \$ on a device from the 1980’s. The cost barrier scares most people away from trying something new. Others might spend their money foolishly because they don’t know about the other alternatives.

Tonight I will be talking a little bit about the history of the Packet Radio TNC and why it’s no longer needed in most cases. You can do the same thing better and cheaper by using free open source software.

Ham radio transceivers are designed for voice use not modems to send data. The radios distort the signals, making it harder for the modems to perform well. I will discuss how the signals are distorted and one technique to compensate for this distortion.



Back before personal computers, this is how we sent text over ham radio.

Ask audience: How many of you have used one of these?

(??? and WN2BUB.)



# Radio + Teletype = RTTY



Simple Modem

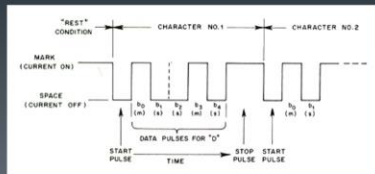
Audio & PTT



Modem was called a "terminal unit".

170 Hz AFSK

AFSK → SSB → FSK



Teletype machines were first developed in the 1930's. As soon as they became available on the surplus market, hams began to use them on the air.

It wasn't difficult to get started. You just needed to add a simple modem which was called a "terminal unit."

It was not a efficient way to send text. First you would turn on the transmitter and it would sit there transmitting a 100% duty cycle carrier. The frequency would shift by 170 Hz in a certain pattern for each key pressed. There was no error detection so you would often see garbled messages.

It used a 5 bit code, commonly known as "baudot." This allows only 32 different combinations, not enough for all letters and digits. Some of the codes were for control functions like carriage return, line feed, or bell. 26 of the codes were for letters. A "shift" control code was used to get digits or special characters instead of letters. If noise clobbered the shift or unshift code, you would see digits and special characters instead of letters or vice versa.



## Packet Radio - 1978

- Radical new concept.
- Montreal Amateur Radio Club.
- Vancouver Area Digital Communications Group.
- Each transmission was a short burst (“packet” or “frame”) containing:

Flag	Adresse	Kontrollinfo	Daten	CRC	Flag
01111110	112/224 Bits	8/16 Bits	n * 8 Bits	16 Bits	01111110

U(nprotocol) oder S(upervisory) Datenpaket

In 1978 Canadian hams began experimenting with a much different method of sending data over the air.

Rather than keeping the transmitter on and sending one character at a time, it was sent in a short package (or “packet”).

Each transmission contained: (explain these ...)

Source address. (e.g. ham callsign)

Destination address.

Optional repeater addresses.

Control / protocol bytes.

Information part.

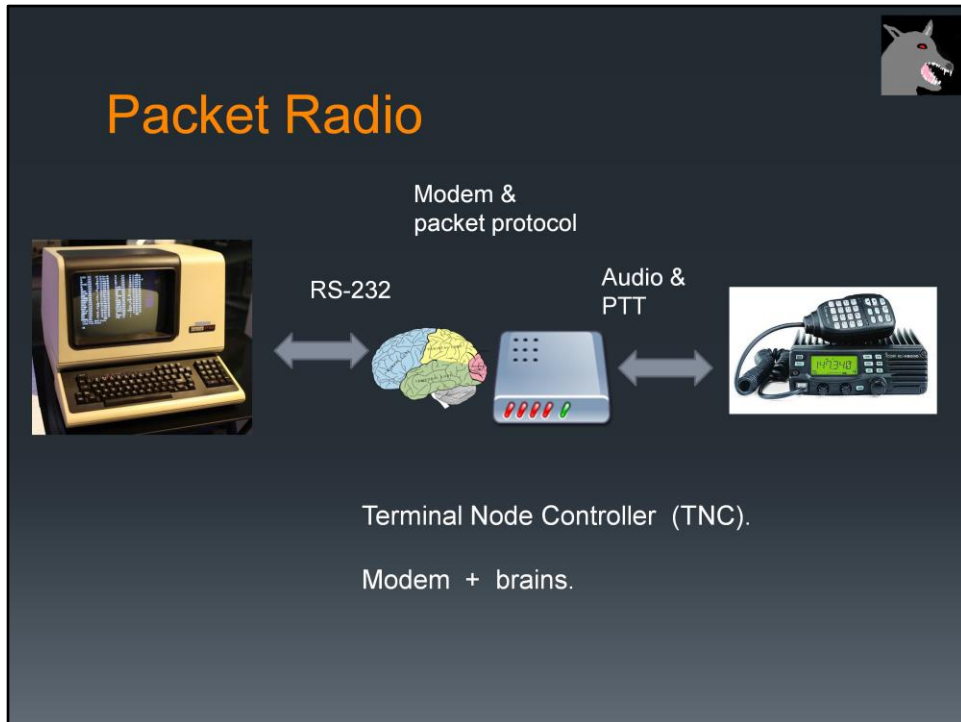
Error checking (FCS, CRC).

[[ Why is part of this in some other language? (Looks like German.)

This is what I found in the Power Point clip art collection and never went back to fix it.

]]

# Packet Radio



With RTTY we had only a simple modem to convert between digital data and audio tones. When a key was pressed the corresponding code was sent immediately.

We now have some brains in between the terminal and modem to implement the protocol used over the air.

This device is called a Terminal Node Controller (TNC).

The operator could take a while to compose a message but the complete message gets sent in a short burst.



## Advantages of Packet Radio

- Short bursts.
- Addresses.
- Shared channel.
- Error detection.
- ACK & retry – for “connected” mode.
- Repeaters.
- “Transparent” for “binary” data.

This new approach offers many advantages.

- Rather than tying up a radio channel with one person typing a character at a time, a message could be sent in a short burst.
- Each packet contains information about where it came from and where it should be going.
- This allows many people to share one frequency.
- Error detection provided confidence that the data was not corrupted.
- The TNC would send acknowledgements when data was received correctly and perform retries when it doesn't get thru the first time.
- Data was not limited to printable characters. You can send files such as JPEG images.

## TAPR TNC-1 kit - 1983



American ham radio operators had a disadvantage. The FCC did not allow the ASCII code to be used over ham radio until 1980. Special permission was required to perform the same type of experimentation as the Canadians.

The Vancouver Amateur Digital Communications Group made their design available as a bare printed circuit board. It was necessary to gather up all the other parts which discouraged most.

In 1983, Tucson Amateur Packet Radio (pronounced tapper) introduced their TNC-1 kit which made it a lot easier.

- All parts including modem and power supply.
- Documentation about 2 inches thick.
- \$350 Did not include case.

This was later available as the Heathkit HD-4040.

They threw a lot of hardware at the problem. About 27 integrated circuits.

The TNC-2 came along a couple years later. It was smaller and cheaper. MFJ and



others produced products based on this design. For a while , everyone was churning out new TNC products.



## Dumb Terminal to Human Interface

```
K1OJH>ALL,WORLI*:NEPRA meeting tonight at 7:00
```

```
cmd> c k7ve
```

```
*** CONNECTED to K7VE
```

```
I will bring the cable that you need tonight.
```

```
Great! See you there.
```

```
(ctrl-C)
```

```
cmd> d
```

```
*** DISCONNECTED from K7VE
```

Not good for computer to computer.

The command line interface was designed for a person sitting at a dumb terminal talking to another person.

Soon people got bored with talking person to person and started building applications such as bulletin board systems. This interface not well suited for communication with a computer application.

Characters going TO the TNC:

- Commands.
- Data for another connected station.

Characters coming FROM the TNC:

- Monitoring of activity.
- Response to commands.
- Text from other connected station.
- Status messages – vendor specific.
- Echoing of what was typed.

# K.I.S.S. Interface - 1986



RS-232

Very simple  
(KISS) TNC



Adresse	Kontrollinfo	Daten
112/224 Bits	8/16 Bits	n * 8 Bits

Flag	Adresse	Kontrollinfo	Daten	CRC	Flag
01111110	112/224 Bits	8/16 Bits	n * 8 Bits	16 Bits	01111110

Smaller brain needed for TNC.

- Transmit: TNC adds CRC and HDLC flags.
- Receive: TNC checks for correct CRC and removes it.

This led to the “KISS” interface, in 1986, which was better suited for talking to a computer application. This was meant only as a temporary stop gap measure until something better could be devised. We are still using it more than 30 years later.

Most of the protocol handling is moved to the computer. The TNC simply adds wraps the packet contents in HDLC.

Full featured TNCs have a KISS mode which allows an application to bypass the usual user interface and get closer to what actually goes over the radio.

The protocol is too simple and doesn’t provide any information about the TNC status or what is happening on the radio channel. For example an application might send a dozen packets to be transmitted. There is no way for the application to know whether they have been transmitted, are still waiting on a busy channel, or whether some of them were discarded because the TNC ran out of memory. Different people started filling in the gaps with different incompatible schemes.



## APRS – Data Types

*“APRS is not a vehicle tracking system. ...”*

- Positions (usually transmitting station.)
- Objects (usually on behalf of other entity.)
- Weather Reports.
- Telemetry.
- “Messages” to an individual or bulletins to groups.
- Queries and Responses.
- APRS of Things. (like Internet of Things)

This is not a general Introduction to APRS but I would like to emphasize it is far more tracking cars or a house saying, “I’m still where I was 2 minutes ago. I’m still where I was 2 minutes ago. I’m still where I was 2 minutes ago.”

APRS has a lot of potential and we are using only a small part of it.

Position and Object reports are similar there is an important distinction.

A “Position” report is information about the sender. It refers to the station in the AX.25 source address.

An “Object” report is information about something else. The Object report format has a separate field for the Object name which would be different than that sender of the packet.

Many hams have home weather stations and share information over APRS. This allows information to be aggregated without use of the Internet.

Telemetry ...

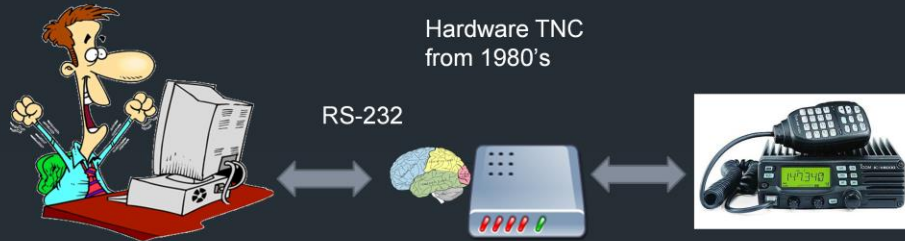
Etc.....

APRS of Things

<https://microhams.blob.core.windows.net/content/2018/03/MHDC2018-K7UDR.pdf>



## Packet / APRS – late 20<sup>th</sup> Century



15 or 20 years ago this was the accepted way of doing things.

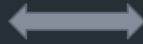
You would run some sort of application on your computer and have a special box between your computer and radio.



## TNC replaced by software



Audio &  
PTT



Cheaper.

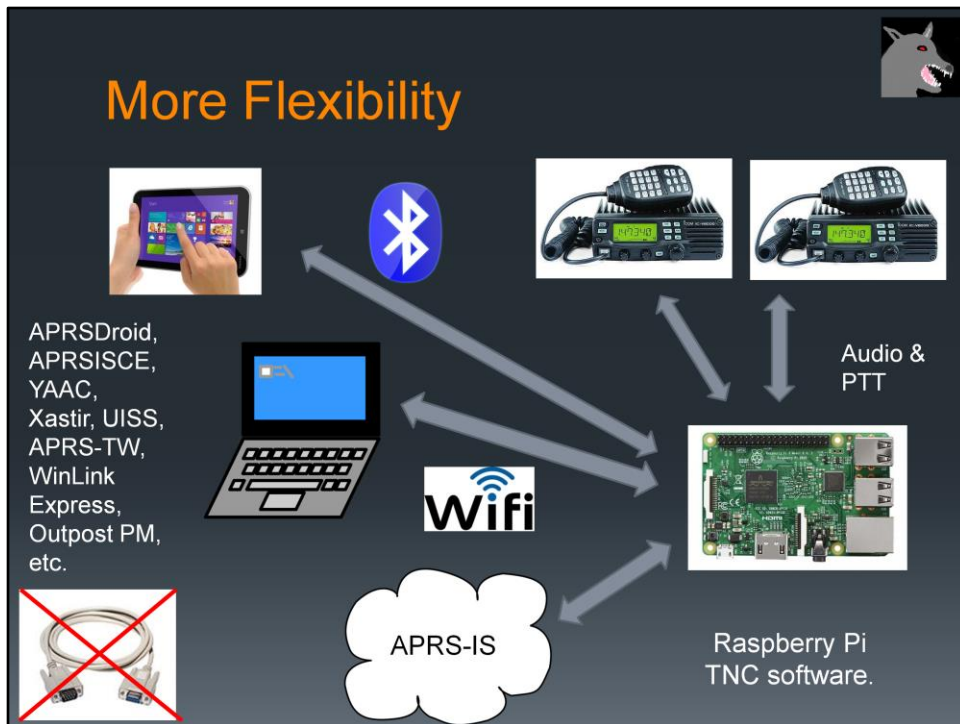
Better Results.

If you are already using a computer, there is no need for a separate TNC.

The TNC functionality can be handled by software running on the PC.

Cheaper, better results. Easy upgrades – software update instead of buying new EPROMs.

There are a lot of “Introduction to APRS” articles – mostly 10 to 15 years old - that say you need to spend \$\$\$ for a device in the middle. This hasn’t been true for a long time.



The software TNC approach is very flexible. You are no longer constrained by that serial port cable and a single radio interface.

You could have a software TNC running on a Raspberry Pi in your “shack” near your radios. It can run digital repeaters on multiple frequencies at the same time. It can be a router between different radio networks with user defined filters to limit what is allowed through.

At the same time, it can be an Internet Gateway, connecting your local radio network to others around the world over the Internet.

Multiple applications can be using it in place of a traditional TNC at the same time.

On a nice summer day, you could be relaxing next to the swimming pool with your laptop or tablet instead of being in your basement. The application you are running connects to the TNC software using WiFi or BlueTooth.





## What is Dire Wolf ?

Open source software replacement for the traditional TNC.

- Windows.
- Linux - x86, x86\_64 PC, Raspberry Pi.
- Mac OSX.
  
- GPS Tracker.
- Digipeater.
- Internet Gateway (IGate).
- APRStt gateway.
  
- Virtual TNC for applications such as APRSIS32, YAAC, Xastir, SARTrack, APRS-TW, UISS, Linux AX25, WinLink Express (RMS Express), Outpost PM, Lincac, and many others.

Adlib ...

Runs on these operating systems...

Provides these functions without additional applications...

Dire Wolf can be used with any application designed for use with an old fashioned TNC.

APRSIS32, YAAC, and Xastir are popular APRS applications which display icons on a map.

SARTrack looks similar but was specifically designed for Search and Rescue.

(Note: It's time to stop mentioning UI-View32. The author became a silent key in 2004 and left instructions to destroy the source code. It is frozen at that point and I've seen mention that it has problems with anything newer than Windows XP.)

APRS-TW is specifically for watching telemetry. You can do things such as setting alarms when values cross thresholds.

UISS is for contacting the International Space Station.

Linux AX25 is a collection of applications which allow you to run TCP/IP over ham radio.

WinLink Express (formerly RMS Express) and Outpost PM are messaging applications used by emergency communications groups.

Linpac is short for Linux Packet.



## Where did the name come from?

Decoded  
Information from  
Radio  
Emissions for

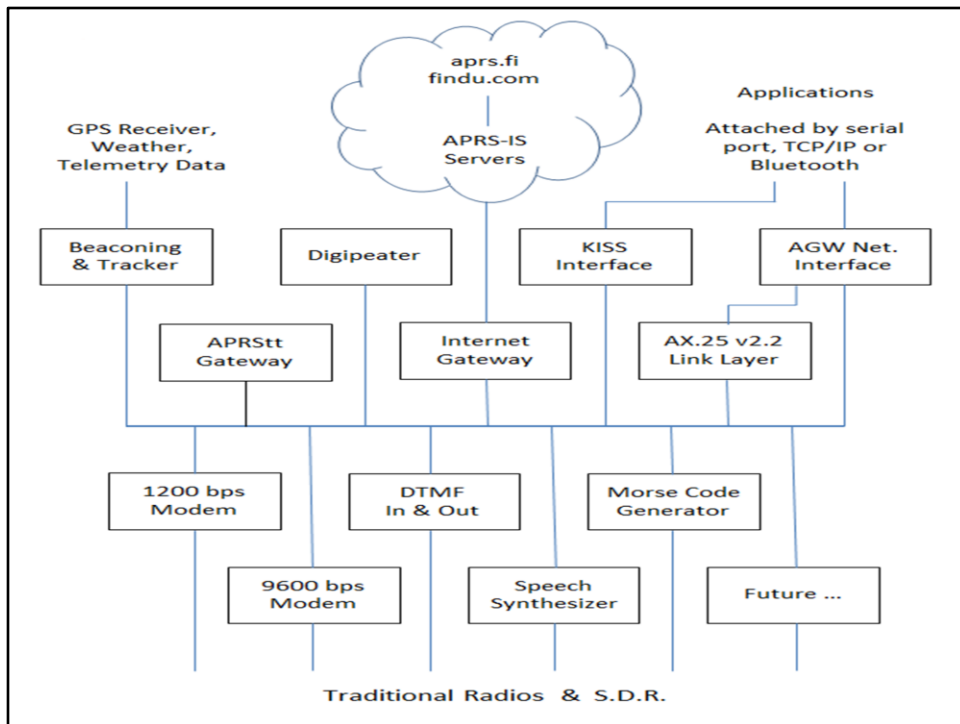
Windows  
Or  
Linux  
Fans

What should it be called?

Something like "soft TNC" would too generic and boring.

Can we come up with some sort of acronym that incorporates words that describe it?

I started juggling words around and came up with this.



Here is a block diagram of the Dire Wolf functionality.

Don't try to read all the little boxes. We will be visiting each of them separately.

The key point here is the radio is at the bottom layer.

Above that are modems and other ways of sending information over the radio.

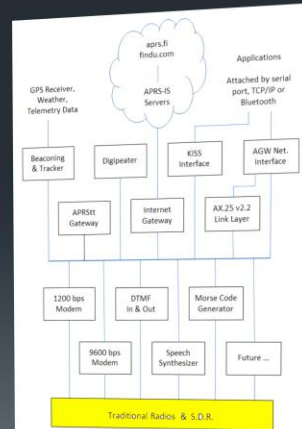
Above that are different ways to produce or consume data going over the radio.

Finally along the top, we have other external interfaces such as GPS receivers, weather stations, the Internet, and other local applications.

## Traditional Radio Interface



Receiver audio → computer.  
Computer audio → transmitter.



Starting at the bottom of the block diagram, this is what connects to your radio.

The interface to the radio is just audio in, audio out, and some sort of PTT control.

All of these devices look much different but they all do the same thing.

The early computers did not have audio built into the motherboard so you needed a separate “sound card” as shown in the upper left.

Back in the old days, it was common to use one of the control lines, from the serial port, to turn on the transmitter.

Below that is a USB audio adapter that runs in the 5 to 10 dollar range. It has audio output for headphones and audio input for a microphone. What do we do for the push-to-talk signal. If you look closely you will see a couple wires sticking out. The most common chips have extra general purpose digital I/O pins available. These are sometimes used for pushbuttons or LED indicators. We can use one of the pins for our push-to-talk control.

This is a very tidy solution because it can all be done with a single USB port. There are at least 3 commercial products available that use this approach and many homebrew plans.

The Signalink USB is an overgrown version of the USB audio adapter. It also has audio isolation transformers and a relay to break up the ground between the radio and computer. It also has a VOX circuit that turns on the transmitter when transmit audio is present.

Finally, in the lower right we have a Raspberry Pi with the UDRC audio board on top. In this case we use Raspberry Pi General Purpose I/O pins for the PTT control.

The round 6 pin connector matches the "data" connector found on many transceivers.

-- stop here --

<b>DMK URI</b>	<a href="http://www.dmkeng.com/URI_Order_Page.htm">http://www.dmkeng.com/URI_Order_Page.htm</a>
<b>RB-USB RIM</b>	<a href="http://www.repeater-builder.com/products/usb-rim-lite.html">http://www.repeater-builder.com/products/usb-rim-lite.html</a>
<b>RA-35</b>	<a href="http://www.masterscommunications.com/products/radio-adapter/ra35.html">http://www.masterscommunications.com/products/radio-adapter/ra35.html</a>

There are several similar homebrew projects:

<http://www.qsl.net/kb9mwr/projects/voip/usbfob-119.pdf>  
<http://rtpdir.weebly.com/uploads/1/6/8/7/1687703/usbfob.pdf>  
<http://www.repeater-builder.com/projects/fob/USB-Fob-Construction.pdf>  
<https://irongarment.wordpress.com/2011/03/29/cm108-compatible-chips-with-gpio>

Don't read out loud:

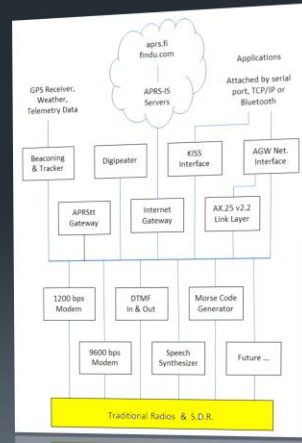
Why the block diagram on the right? My thinking was that it shows where we have zoomed in on the larger picture. Is this helpful or distracting?



## Software Defined Radio (SDR) Interface

- Pipe into stdin

```
rtl_fm -f 144.39M | direwolf -
```
- Listen for audio on a UDP port.  
(e.g. `gqrX v2.3` and later)
- Virtual audio cable. `SDR#`.



For about \$20 you can get a little gadget that receives from 500 kHz up to 1.7 GHz. Just add a computer, and the right software, and you have a very versatile receiver.

Dire Wolf can take audio from various SDR applications in different ways.

A simple FM receiver, with no graphical user interface, called `rtl_fm`, produces audio thru stdout. In this case we can pipe it in thru stdin represented by the dash at the end of the line.

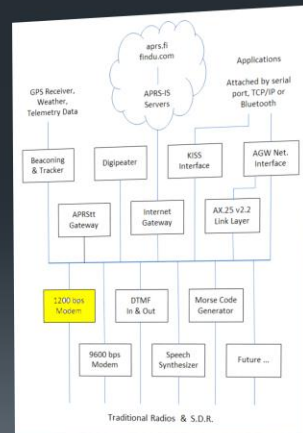
`gqrX` has a provision to send digital audio by UDP. Dire Wolf can be configured to listen for audio over a UDP port.

The last time I looked at `SDR#`, it had no special provision for sending received audio into another application for processing. In this case, we can use a virtual audio cable to route audio from one application into another.

## 1200 bps modem



AFSK 1200 / 2200 Hz.





It is interesting to note that the the AX.25 protocol standard says absolutely nothing about modems.

According to one of the early Canadian packet radio pioneers, they happened to get a big pile of old Bell 202 modems for 75 cents each. It was a quick and dirty TEMPORARY way to get something up and running.

Everyone kept saying it would soon be replaced by something better but here we are, 40 years later, and it is still the most commonly used.




9600 bps

K9NG G3RUH – around 1988

5 kHz of audio bandwidth.

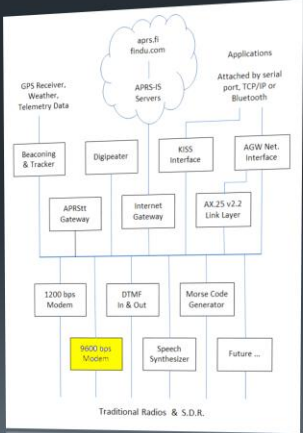
Will not work with microphone and speaker connections.



Squelch/COR  
"DATA Out 9600"  
(FM Discriminator)  
GND/COM

"Data Out 1200"  
(Receive Audio)  
PTT  
"Data In"  
(Transmit Audio)

Stephen H. Smith WA8LMF@aol.com 10 March 2011



GPS Receiver, Weather, Telemetry Data  
Beaconing & Tracker  
APRS Gateway  
1200 bps Modem  
9600 bps Modem  
Digitaler  
Internet Gateway  
DTMF In & Out  
KISS Interface  
Morse Code Generator  
AGW Net. Interface  
Attached by serial port, TCP/IP or Bluetooth  
AX.25 v2.2 Link Layer  
Applications  
Traditional Radios & S.D.R.

Around 1988 (30 years ago, as I'm writing this) pioneers such as K9NG and G3RUH came up with hardware solutions to send data at 9600 bits per second.

I wouldn't call them "modems" because they just shoved the digital signal into the transmitter resulting in Frequency Shift Keying of the RF signal.

This will not work with the microphone and speaker connectors because the audio passband is too narrow and not flat.

It is necessary to bypass the audio filtering to get wider flatter audio passband. The older literature referred to making a connection directly to the discriminator. In more recent times, there is often a connector available for external modems.

The big 3 ham radio manufacturers (Kenwood, Icom, Yaesu) standardized on the same 6 pin mini DIN connector. Alinco has the same functionality but a different connector.

The 9600 pin comes right from the discriminator without de-emphasis. This has enough bandwidth for 9600 baud data.

The "data in" pin bypasses the pre-emphasis and goes right into the modulator.

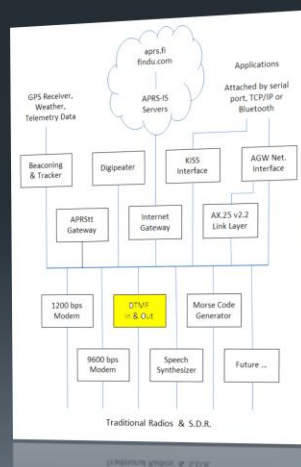
## DTMF decode / encode

Received tone sequences are converted to packet like:

```
DTMF>APDW15:t12345#
```

Transmit tones by putting DTMF in the destination address.

```
WB2OSZ>DTMF:123 456
```



Dire Wolf also has the ability to encode and decode DTMF, commonly known as “Touch Tones.”

In the receive direction, tone sequences terminated by the # key are converted to the same internal format as normal packets.

The information part of each APRS packet begins with a character called the data type indicator. In this case we set it to lower case “t” to indicate DTMF tone sequence.

In the transmit direction, a destination field of “DTMF” means that touch tones should be sent instead of the normal AX.25 frame.

Otherwise these are handled like any other packets. Upper level applications can process them and create them.

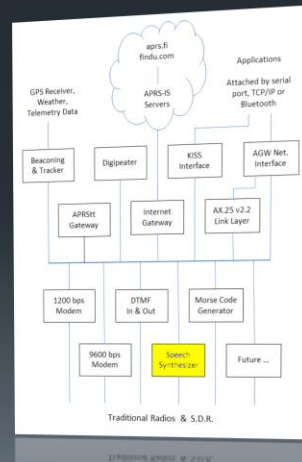
## Speech Synthesizer

Any packet with SPEECH is sent to a user-supplied script which can invoke a text-to-speech synthesizer.

```
WB2OSZ>SPEECH:Hello, World!
```

Configuration file example:

```
CBEACON dest=SPEECH  
info="Club meeting tonight  
at 7 pm."
```



Dozens of free speech synthesizers are available. Just feed in text and out comes spoken words.

When the destination address contains "SPEECH" the information is not sent as a normal packet.

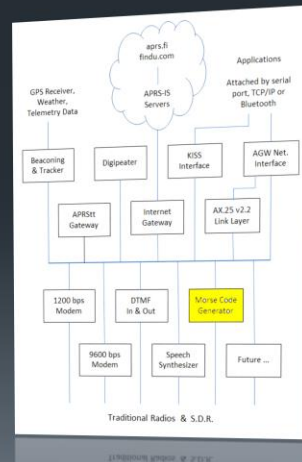
Instead, a user-defined script is invoked. This can pass the text along to the speech synthesizer of choice.

## Send Morse Code



Any packet with MORSE as the destination is sent as Morse Code.

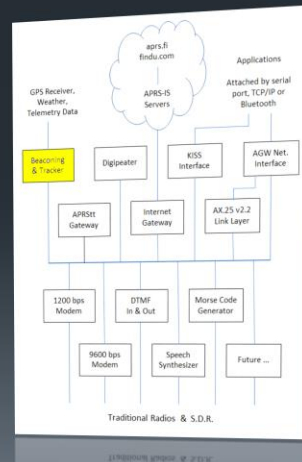
```
WB2OSZ>MORSE:CQ CQ
```



Similarly, if the destination address is "MORSE" the information part is sent as Morse Code.

## Beacons

- Periodic position or object packets.
- GPS location - tracker.
- "Custom" - invoke user script.
- Weather. wxnow.txt



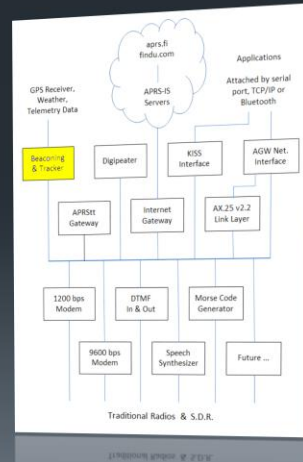
Beacons are automatic periodic transmissions usually containing a location and an icon for something like a house or a car.

When the location comes from a GPS receiver, we have a tracker.

Beacons can have variable content. In this case, a user defined script is run to generate the information part. This could be weather information or anything else you want to put there.

## Telemetry Tool Kit

- Building blocks for your own customized solutions.
- User-defined script to generate content.
- Raspberry Pi A/D converter example.



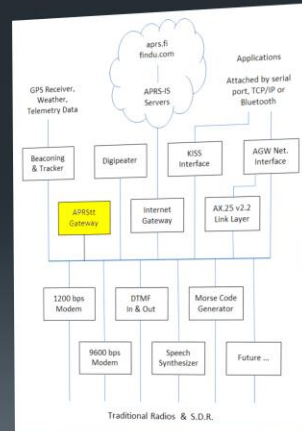
A collection of scripts and examples that can be used as building blocks for constructing your own customized solutions for your particular needs.

Beacon invokes user-defined script to generate packet content.

Includes an example of connecting an A/D converter to a Raspberry Pi and transmitting the measured voltage as APRS Telemetry.

## APRStt Gateway

Converts Touch Tone sequences into APRS objects.

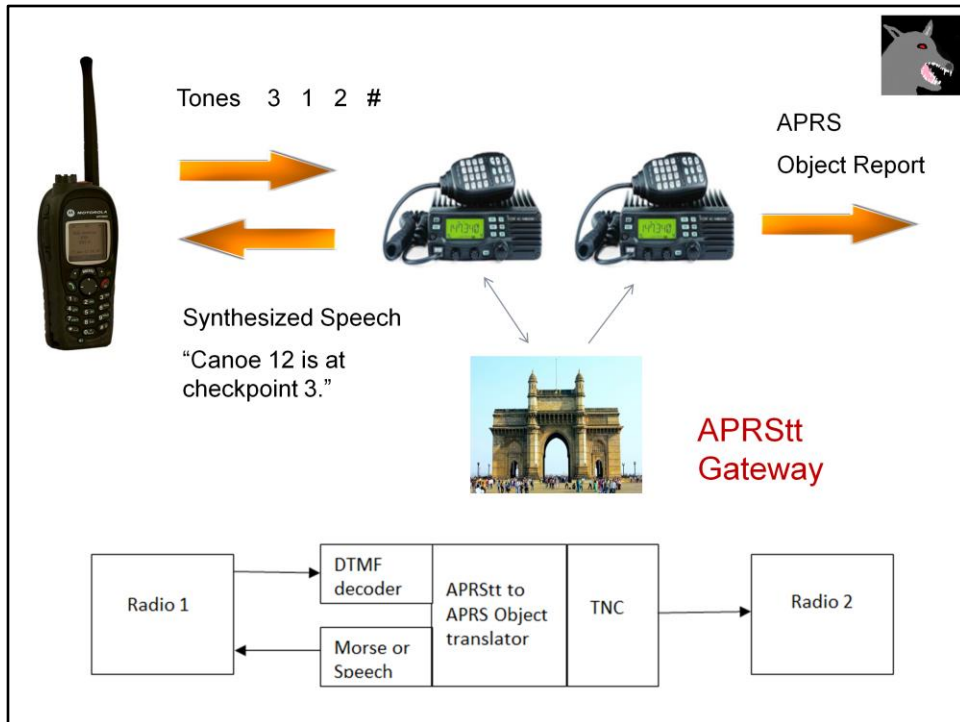


Very few hams have portable equipment for APRS but nearly everyone has a handheld radio that can send DTMF tones.

APRStt allows a user, equipped with only DTMF (commonly known as Touch Tone) generation capability, to enter information into the global APRS data network.

It's been around since the turn of the century but never caught on.

Earlier implementations required specialized hardware and were not easily adaptable to new situations.



Suppose the organizers of a canoe race wanted to keep track of the locations of the participants but didn't have a budget to put a GPS APRS tracker on each canoe.

Instead they have observers at various points along the way.

If the observer at checkpoint 3 wanted to report that canoe 12 was passing by, she would turn on her transmitter and send the tones 3 1 2 #.

The APRStt gateway would interpret the first digit as a location.

The other two digits would be an object number. This object might be a race participant, staff member, or piece of equipment.

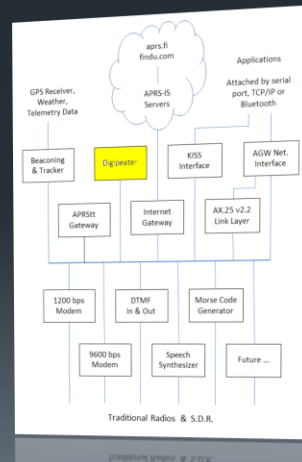
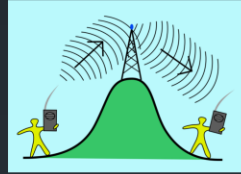
The gateway would confirm receipt of the report by responding with synthesized speech.

It would also send an APRS Object Report, typically on a different radio channel.



## Digital Repeater “digipeater”

- Extends range.
- Store and Forward.
- Multiple channels.
- Filtering for special situations.  
(addresses, distance, data type, symbol)



A digital repeater serves the same purpose as a voice repeater. It allows communication between stations that can't hear each other directly.

A voice repeater receives on one frequency and simultaneously retransmits on a different frequency.

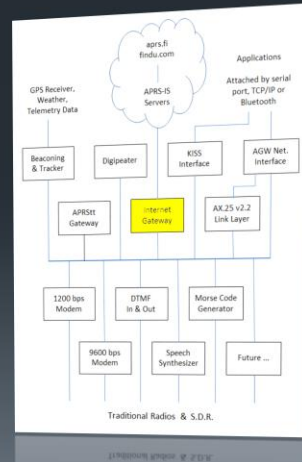
A digital repeater generally operates on a single frequency. A packet is stored in memory and then retransmitted when the channel is clear. There is a provision for the maximum number of repeater hops. This is decremented each time so a packet does not go bouncing back and forth forever.

## Internet Gateway "IGate"



Connect disjoint radio networks.

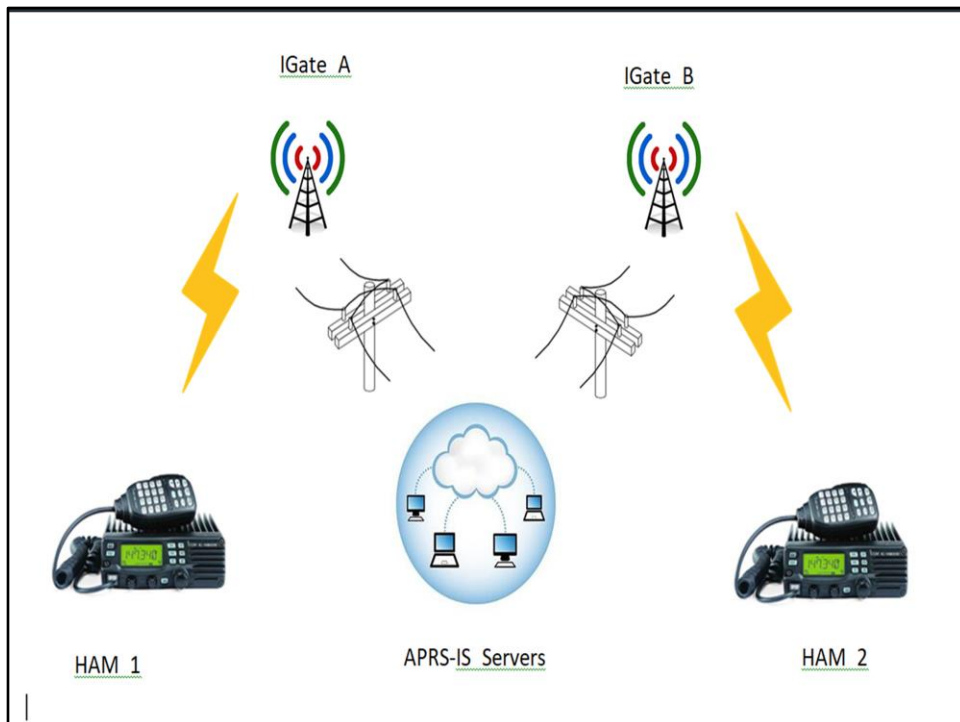
View APRS activity at <http://aprs.fi> or <http://findu.com> or with other applications.



IGate stations allow communication between disjoint radio networks by allowing some content to flow between them over the Internet.

IGate stations also allow someone without a radio, or outside of your local area, to view APRS activity at <http://aprs.fi> or <http://findu.com> or with other applications.

Information can also be relayed from the Internet Servers (IS), through your station, on to the radio channel.



Properly configured IGate stations offer two way communication.

RF to Internet AND Internet to RF.

Here is an example of how they can be used to send an APRS message to someone on the other side of the world.

Ham 1 sends an APRS message addressed to Ham 2.

On the local radio network, this might be repeated by a couple digital repeaters before being picked up by an Internet Gateway station.

The gateway station A passes the message along to the global network of servers.

The servers check their database for recent activity of Ham 2 and which IGate stations reported that information.

The servers forward the message to those IGate stations. The message could then be repeated by digital repeaters along the way.

When Ham 2 receives the message, an automatic acknowledgement goes back to the

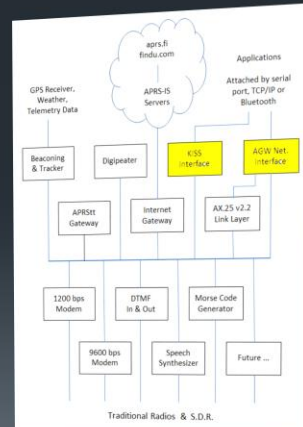
sender in a similar way.

The messaging application, used by Ham 1, reports that the message was received. If there is no response for a while, multiple retries are made until it gives up and reports failure.

Some people set up receive only IGate stations which go from RF to the Internet but not the other way. This breaks the messaging system because there might not be another nearby return path for a reply.

## Application Interfaces

- KISS and AGW network interfaces.
  - AGW interface allows access to more brains in the TNC such as connected mode.
- Applications attached by:
  - RS-232 Serial Port.
  - TCP/IP ( Ethernet, WIFI )
  - Bluetooth.
- Many simultaneous applications.



Besides operating standalone, Dire Wolf can also be used as a virtual TNC by many applications that were written for use with the old TNCs.

We are no longer restricted to an RS-232 signal cable. We can now use TCP/IP, WiFi, and Ethernet. Many applications can access the virtual TNC at the same time.

KISS, the “temporary” stop gap measure from 30 years ago is still going strong.

There is also a newer “AGW” protocol (from AGWPE) which allows applications to access more intelligence in the TNC.

For example, the AX.25 link layer for connected mode operation. No sense in each application having to re-invent this.



## Building a Better Demodulator

The most technically challenging part of this was building a the 1200 baud demodulator.

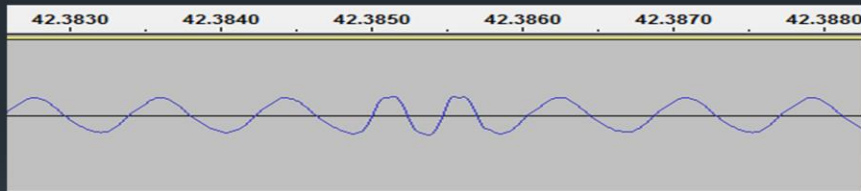
I have no academic or professional background in digital signal processing so I had no clue on how to go about doing this.

It involved a lot of reading and experimenting and relentless tweaking until it worked better.



## Audio Frequency Shift Keying

- 1200 bits per second, 1200 & 2200 Hz.



- Which of the two tones is present?

The AX.25 protocol standard says absolutely nothing about the modem to be used.

As mentioned earlier, the early experimenters used the Bell 202 standard because they happened to acquire a pile of modems dirt cheap. They used those to get started until something better could be developed.

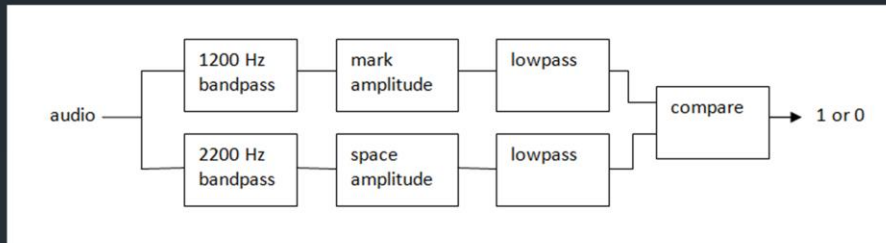
Here we are, 40 years later still using the same thing.

Data is sent at 1200 bits per second using two different audio frequencies: 1200 & 2200 Hz, often called “mark” and “space,” to represent the logic 1 and 0 values. This is called audio frequency shift keying (AFSK).

A demodulator needs to listen for those two tones and decide which one is present.



## How to Demodulate AFSK



This is one way we can implement a demodulator for AFSK. We start off with a bandpass filter for each of the audio frequencies. In the old days, you might use an inductor and capacitor. Later we progressed to active filters with op amps. Today it can be done in software.

What we are interested in is the amplitude of the tone, not the sine wave. In the old days, this would be a full wave rectifier. The software equivalent is taking the absolute value so the negative peaks become positive.

A low pass filter allows the lower speed data through while removing the higher frequency ripple and noise.

Finally, we have a comparator to decide which tone is stronger and the result is logic 1 or 0.

This goes to an HDLC decoder that looks for patterns in the bit stream and spits out groups called "frames."





## After much reading & experimenting

- OK with perfect signals.
- Not so good with real-world signals.

I was thrilled when I was able to decode ideal signals.

It soon turned into disappointment when I tried it on real world signals and it did not work very well compared to a TNC from the previous Century.

Why? How were the real world signals different than the ideal signals?



## WA8LMF TNC Test CD

- Los Angeles, afternoon rush hour, frequency completely saturated.
- Track 1: 25 minutes of flat audio from the discriminator.
- Track 2: De-emphasis to mimic typical receiver. (explained later)
- The de facto standard for measuring demodulator performance.

WA8LMF has provided the ham community with a wonderful resource for evaluating the performance of TNCs with real world signals.

He drove around Los Angeles, during rush hour gathering a wide variety of APRS signals. It was wonderful, under deviation, over deviation, people stomping on each other, even some really strange looking signals that make you wonder where they came from.

(Note: Look for something like “A closer look at the TNC Test CD” in the documentation directory.)

This is an important point. Track 1 has the wide flat audio taken from the discriminator before any de-emphasis.

Track 2 has low pass filtering to produce a signal more like what would typically come out of the speaker.

This has become the de facto standard for measuring TNC decode performance.



## TNC Test CD Results

Do not take too seriously. Collected by different people, at different times, under different conditions. Most don't specify Track 1 / Track 2.

▪ Linux PC multimon	*	130	
▪ Linux PC soundmodem	*	412	
▪ AGWPE	*	500	
▪ AEA PK90		728	
▪ TNC-X		818	
▪ MFJ-1274		883	
▪ AX25 Java Soundcard Modem	*	964	
▪ KPC-3 (non-plus)		967, 986	
▪ SCS Tracker DSP TNC 1.5s		988 / 943	(track 1/2)
▪ Dire Wolf	*	1011 / 1004	(track 1/2)
▪ UZ7HO Soundmodem 0.83b	*	1021	

\* = software on PC

Many people have reported number of frames decoded from the TNC Test CD. I have gathered many of the reports together here.

Remember that these numbers were gathered by different people, different times, under different conditions, so don't take them too seriously.

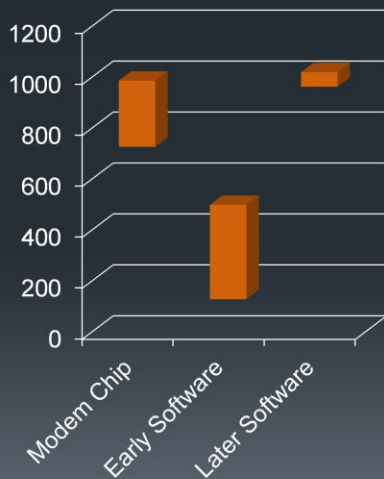
[ Original sources are cited in one of the Dire Wolf documents. ]

The software modems are in yellow. They include the worst and the best.

If we make a little graph, an interesting pattern becomes clearer.



## TNC Decoding Comparison



- Poor results from early software modems.
- Bad reputation.
- More recent software better than hardware modems.
- The tarnished reputation endures.

The early TNCs with modem chips come in at the 700 to high 900 range.

The early “soundcard modem” software did not work very well, giving this approach a bad reputation.

You will find outdated articles claiming that you need to buy special hardware for acceptable results. Was true at one time but not anymore.

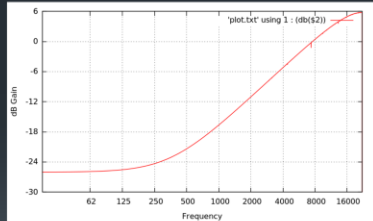
In more recent times, the top two are free software running on general purpose computers.



# VHF FM voice transceivers.

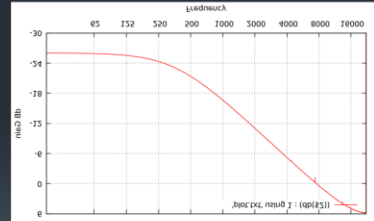
## Transmitter pre-emphasis

+ 6 dB per octave.



## Receiver de-emphasis

- 6 dB per octave



This is a big part of our problem.

When using modems, we would like the audio going into the transmitter to come out of the receiver with a minimum of distortion.

As we saw earlier, some transceivers have special sockets to bypass the audio filtering when using external modems.

VHF FM transceivers are designed for voice and do bad things with signals from modems.

On transmit, we have pre-emphasis. This means the higher audio frequencies are increased in amplitude.

On receive, the opposite happens. De-emphasis uses a low pass filter to decrease the amplitude of higher frequencies.

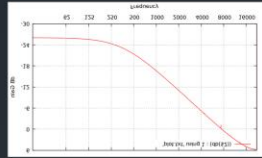
In an ideal world, they would cancel out but we often end up with an audio passband which is not flat.

It might OK for voice but bad for digital data. The distortion makes it more difficult for modems to demodulate signals properly.

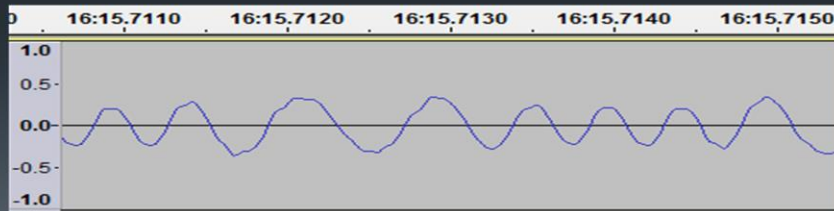


## Transmit / Receive mismatch

Flat on transmit.



Higher frequency weaker.



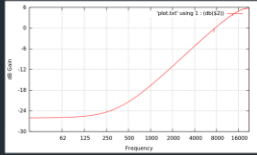
The situation is worse when we have a mismatch between the transmitter and receiver.

If you have no pre-emphasis on the transmitter and a low pass filter on the receiver, you end up with a situation like this.

This is an actual signal captured on the air. The higher audio tone is considerably weaker than the lower frequency.

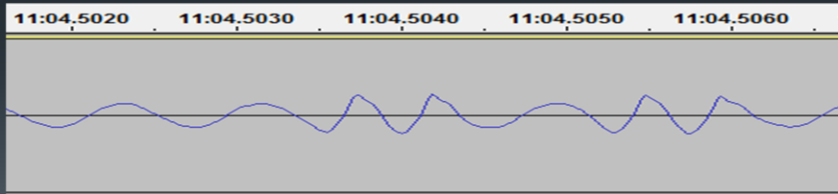


## Transmit / Receive mismatch



Flat on receive.

Higher frequency  
is stronger .



This is the opposite situation.

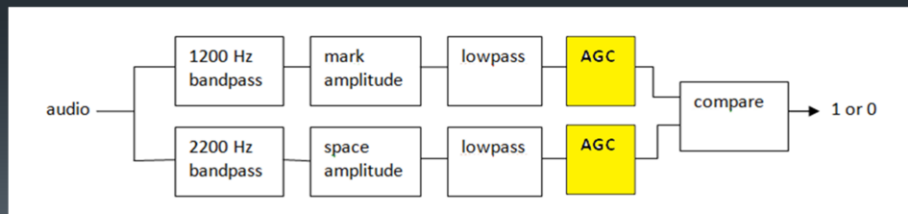
The transmitter has pre-emphasis and boosts the higher audio frequency.

On the receiving end there is no de-emphasis. As a result, the higher audio tone is stronger.



## Transmit / Receive mismatch

- Less reliable.
- Automatic gain control?
- Better but... AGC time ... noise



With either mismatch case, the demodulator has a difficult time rapidly deciding which tone is stronger and it is less reliable.

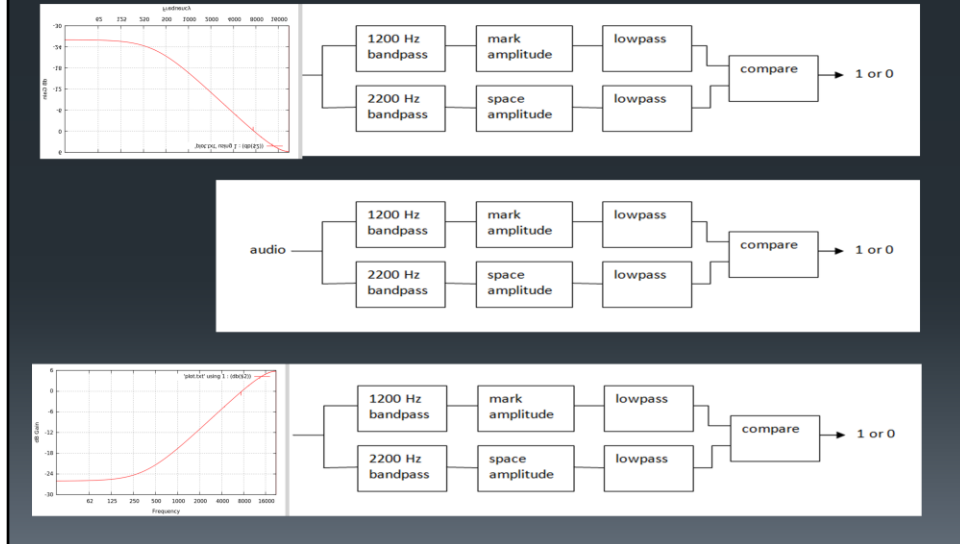
My first attempt at improvement was to add automatic gain control so both would get normalized to the same range.

Results were better but AGC takes time to adjust and can get thrown off by bursts of noise.





## Multiple demodulators.



A brute force approach would be to use multiple demodulators in parallel.

Different filters in front of them would compensate for the various cases of the two tones being out of balance.

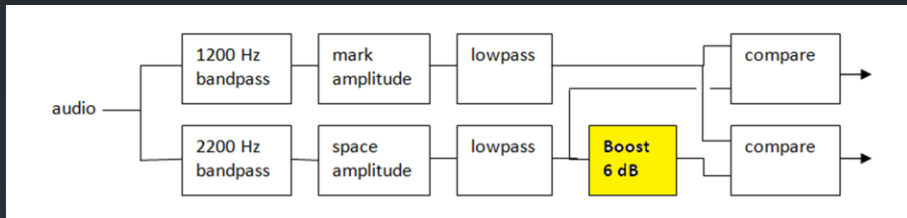
Duplicates, arriving at nearly the same time, would be removed.

The demodulator is compute intensive using hundreds of multiplies and adds for every audio sample. Running three, like this, would triple the CPU power demand.



## Boost Gain for Higher Tone.

- Multiple fixed gains.

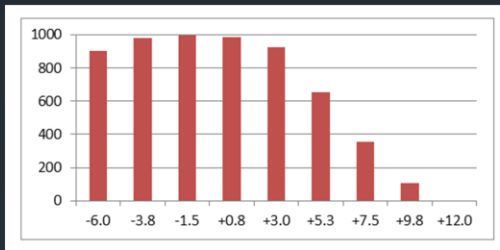


Less compute power needed.

I used a variation on the theme where differing amounts of gain are applied to one of the amplitudes after the computationally intensive part.

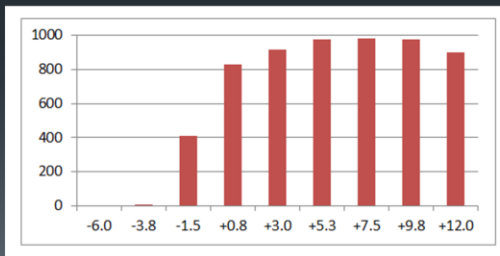
In this case, one of the outputs worked better for Track 1 and the other worked better for Track 2.

## What is best gain?



Track 1 – flat audio.

Gain < 1 compensates for transmitters with pre-emphasis.



Track 2 – de-emphasis.

Gain around 2 (+6 dB) compensates for de-emphasis.

What would be the best gains to boost the higher frequency?

I did a little experiment, using various gains, and counting the number of packets decoded.

For track 1, we find the optimal gain is about **minus** 1.5 dB.

For track 2, about 7.5 is the best.



## What is best gain?

- No one best value so we run 9 decoders in parallel & remove dupes.
- The | or \_ character indicates success/failure for each of the 9.
- Track 1 does better with lower gains. e.g.

```
... Digipeater W6SCE-10 audio level = 50(15/21)      | | | | | _  
[0] K6SYV-10>ANP391,W6SCE-  
10*:!3444.00NS12000.40W#PHG7730/Wn,SCAn/FIGUEROA  
Mt./A=003248<0x0d>
```

- Track 2 does better with higher gains. e.g.

```
... Digipeater W6SCE-10 audio level = 19(2/1)      _ | | | | | | |  
[0] K6SYV-10>ANP391,W6SCE-  
10*:!3444.00NS12000.40W#PHG7730/Wn,SCAn/FIGUEROA  
Mt./A=003248<0x0d>
```

On the right side we have an indication of which decoders were successful. The vertical bar means an error free frame was received. An underscore means failure.

In the first example, a frame from Track 1 does better with the lower gains on the higher frequency.

In the second example, a frame, from Track 2 does better with higher gains.

The best performing demodulators all use some variation on this theme where multiple parallel decoders are used to compensate for the mismatched amplitudes of the two tones.



## One Bad Apple Don't Spoil the Whole Bunch



- There is an old proverb, "*One bad apple spoils the barrel,*" which applies to AX.25 frames used for APRS and traditional packet radio.
- One bad bit → FCS wrong → discard frame.

start flag	addresses, control, information	FCS	end flag
01111110		16 bits	01111110

↑ One corrupted bit causes FCS not to match

There is an old proverb, "*One bad apple spoils the barrel,*" which applies to AX.25 frames used for APRS and traditional packet radio.

If one bit gets corrupted, the frame check sequence is wrong and the frame is discarded.

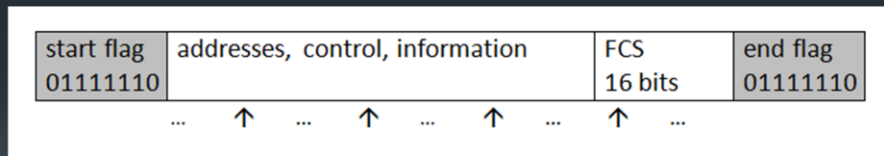
In the traditional connected mode packet, the TNC will retry several times. With APRS, it's just gone.



## One Bad Apple Don't Spoil the Whole Bunch



- The Osmond Brothers offered the advice,  
*"Give it one more try before you give up..."*
- That can also apply to AX.25 frames.



- Invert each of the bits – one at a time! – and recalculate the FCS.

For single bit errors, we can try to invert each of the bits – one at a time! – and recalculate the FCS. My experimentation found this recovered a lot of packets that would normally be discarded.

# Oops!



- False positives on the FCS check

& end up with bogus data.

There is one little problem with flipping various bits trying to find a valid FCS. If we invert one of the bits that was not corrupted we sometimes still get a good CRC. Callsigns contain punctuation characters. The information part has unprintable characters.

The reason is that –

The 16 bit FCS has 65,536 different possible values. Even if totally random data goes into the checking process, you will end up with a valid FCS one out of every 65,536 times.

When you try hundreds of bit flipping combinations and process lots of packets, a fair number will just happen to get past the FCS check and produce bad data.



## Sanity check heuristic

A good AX.25 frame will have:

- An address part that is a multiple of 7 bytes.
- Between 2 and 10 addresses.
- Only upper case letters, digits, and space in the addresses.
- For APRS, certain values in the frame control and protocol octets.
- For APRS, the information part has only printable ASCII characters or:
  - 0x0a           line feed
  - 0x0d           carriage return
  - 0x1c           used by MIC-E
  - etc. other non-printable characters used with APRS.

After applying this extra step of sanity checking, no bad data was visually observed for the single bit fixing case. In very large sample sizes, there were a few cases of bad data getting thru when flipping more than one adjacent bit. Obvious errors are fairly common when flipping two non-adjacent bits.





## A new high score.

Not traditional forward error "correction."

Try flipping each bit, one at a time, until we have a valid FCS (CRC) and the sanity check passes.

	No errors	One bad bit
Track 1:	1011	1028
Track 2:	1004	1023

## Summary



- Ignore the advice from around the turn of the Century
- You don't need to spend \$200 on a box from the 1980's to get started with APRS / Packet Radio.
- Free Software:
  - Better Performance
  - More Features
  - Lower Cost

Ignore the advice from all those Packet Radio / APRS tutorials written 15 or 20 years ago.

They will tell you need to spend a substantial amount of money on special hardware.

If software TNCs are mentioned at all, they are dismissed as having poor performance.

Nowadays the newer software TNCs have left the 1980's style hardware behind in the dust. They have better performance, more features, and lower cost.



## Questions?

For more information:

<https://github.com/wb2os/direwolf>

[https://groups.yahoo.com/neo/groups/direwolf\\_packet/info](https://groups.yahoo.com/neo/groups/direwolf_packet/info)

wb2osz @ arrl . net

- APRS is a registered trademark of APRS Software and Bob Bruninga, WB4APR. <http://www.aprs.net/>
- Linux is a registered trademark owned by Linus Torvalds.