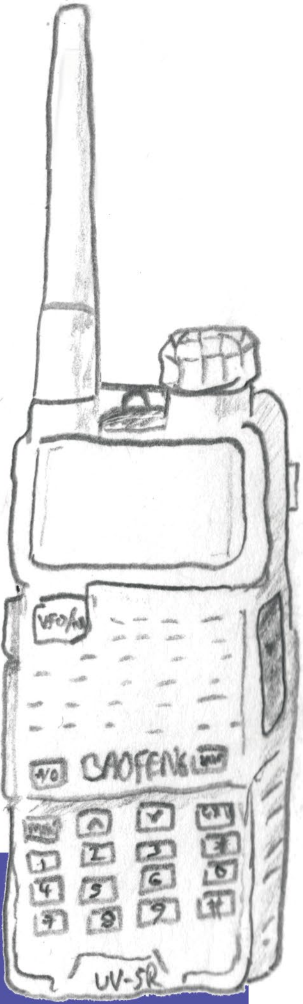
a thing or two about

2nd Edition
Updated and Revised



Messing Around With

Packet Radio

O’REALLY?

of the W

introduction

This booklet is made to give the reader a basic introduction into packet radio. It emphasizes quick, cheap & dirty over specialized knowledge or equipment. It comes from a very practical non-engineering, non-amateur radio perspective. As such, this publication is aimed towards readers that want to get started quickly and feel comfortable to find out things through trial and error, tough with a bit of guidance.

The reader will find that not a lot of attention is given to the rules and regulations governing the spectrum. This is because these rules differ per country, it is left to the reader to look them up. However, take heed and assume that whatever you're going to do in terms of transmission, will be illegal without a radio license.

Packet radio is a bit of an ambiguous term but generally refers to digital forms of radio communication. So rather than using voice, communication happens by encoding 1's and 0's in sound. "Packet" refers to 'bursts' of data that are being sent out now and then. Simultaneously however, it also refers to the fact that one can connect to 'packet switched networks' such as the internet. This booklet will cover both digital and packet switched interpretations.

Packet radio is a technology of the 70's and can be considered a return to the digital roots of radio. Before the transmission of analog voice became common on the airwaves, radio communication happened mostly through digital forms such as Morse code. As a way of building computer networks, packet radio had its heyday in the 90's when dial up internet was prohibitively expensive or cumbersome to many.

In recent years there has been a resurgence of interest due to greater availability of radios, computer equipment, software and documentation. Additionally it is an interesting technology because it shows that 'there is more than one way to do it' when it comes to building computer networks, underlining the plurality of possible internets. Although slow, packet radio systems can establish links over very large distances whilst relying on autonomous infrastructures. That makes the technique an interesting proposition for building autonomous and resilient networks.

TOC

TABLE OF CONTENTS

Hardware

Software

Addendum

This material is not copyrighted. Feel free to reproduce it, distribute it or amend it.

the TRINITY of packet Radio

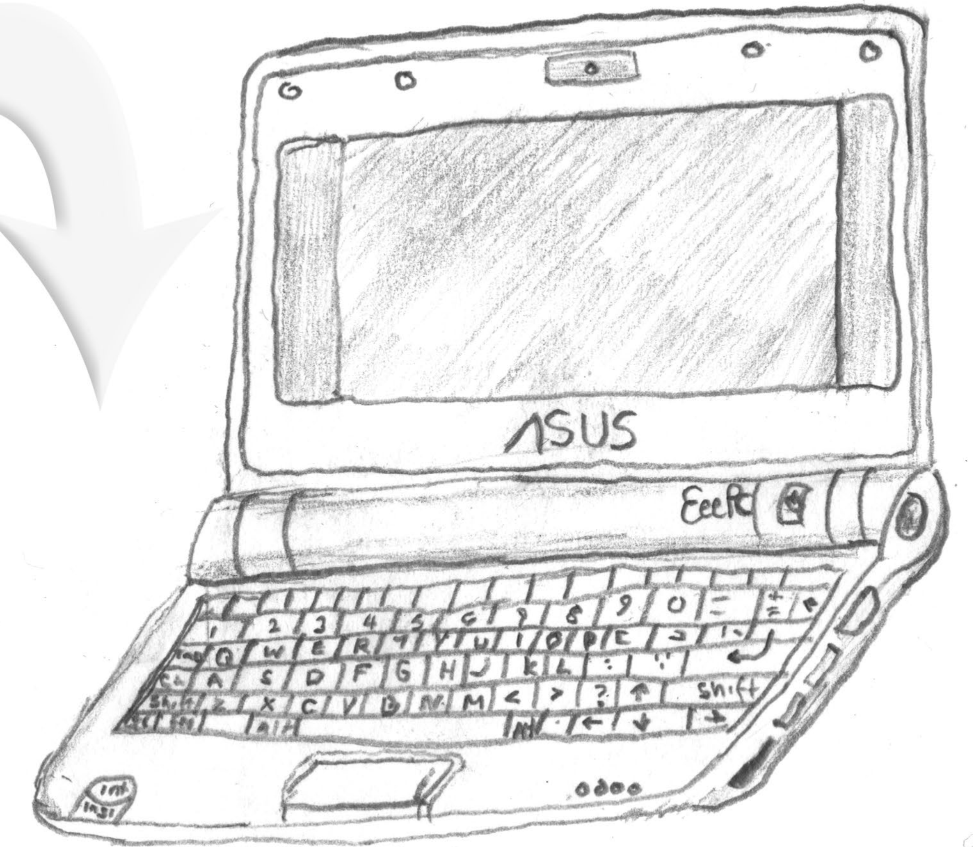
In principle packet radio requires three components for each node in the network : a radio transceiver, a modem and a computer

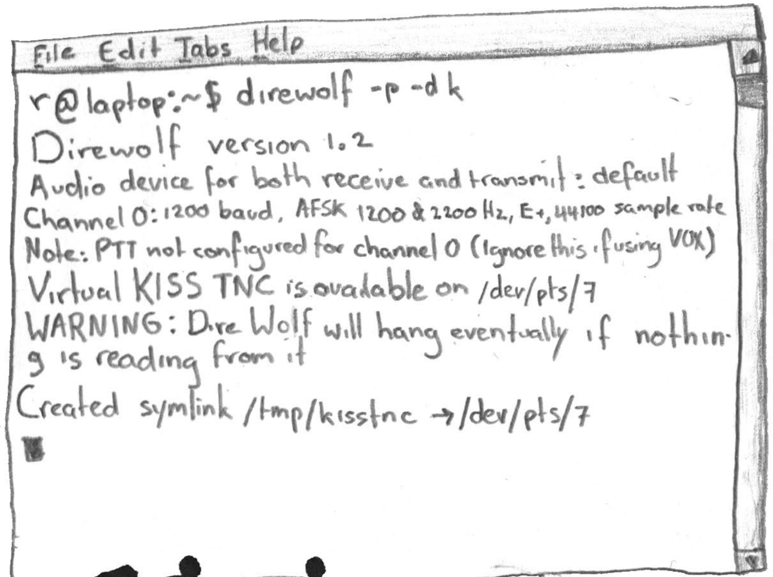
The setup used in this book is the following:

the transceiver is a Baofeng UV5-R Radio,

the modem is a software modem called 'DIREWOLF',

the computer is a laptop running Gnu/Linux

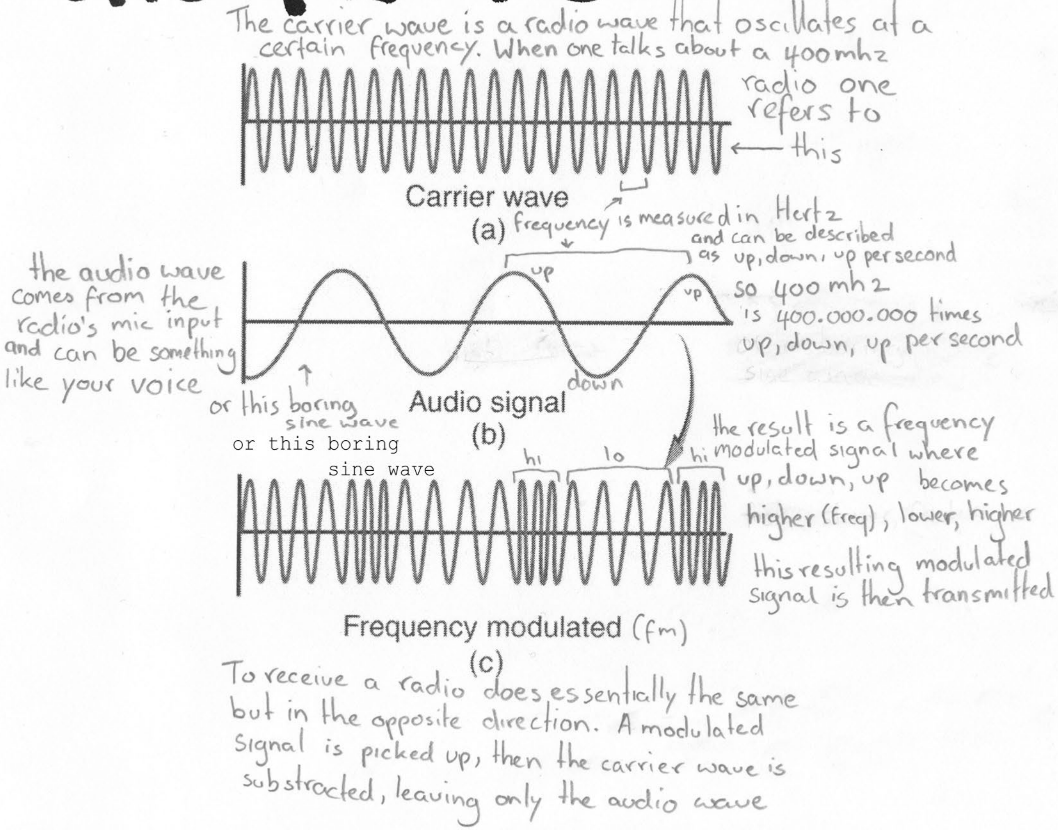




```
r@laptop:~$ direwolf -p -d k
Direwolf version 1.2
Audio device for both receive and transmit : default
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, E+, 44100 sample rate
Note: PTT not configured for channel 0 (Ignore this if using VOX)
Virtual KISS TNC is avadable on /dev/pts/7
WARNING: Dire Wolf will hang eventually if nothin-
g is reading from it
Created symlink /tmp/kisstnc → /dev/pts/7
█
```

Traditionally the setup requires an extra device, the TNC (Terminal Node Controller) which functions both as a modem and as a way to trigger the radio's PTT(Push-to-talk). However, because we use a software modem and the VOX (Voice Operated Switch) setting on the transceiver, a TNC is not necessary. To speed up transmissions one can choose to build an additional device to trigger the PTT, this is covered on page.

the TRANSCEIVER


The carrier wave is a radio wave that oscillates at a certain frequency. When one talks about a 400mhz radio one refers to
← this
Carrier wave
(a)
frequency is measured in Hertz and can be described as up, down, up per second
so 400 mhz is 400.000.000 times up, down, up per second
up
up
down
the audio wave comes from the radio's mic input and can be something like your voice
or this boring sine wave
Audio signal
(b)
hi
lo
hi
the result is a frequency modulated signal where up, down, up becomes higher (freq), lower, higher
this resulting modulated signal is then transmitted
Frequency modulated (fm)
(c)
To receive a radio does essentially the same but in the opposite direction. A modulated signal is picked up, then the carrier wave is substracted, leaving only the audio wave

To transmit, a radio mixes together a carrier wave and an audio wave.
The carrier wave is a radio wave that oscillates at a certain frequency..

frequency is measured in Hertz and can be described as up, down, up per second so 400mhz is 400.000.000 times up, down, up per second.

the audio wave comes from the radio’s microphone input and can be something like your voice.

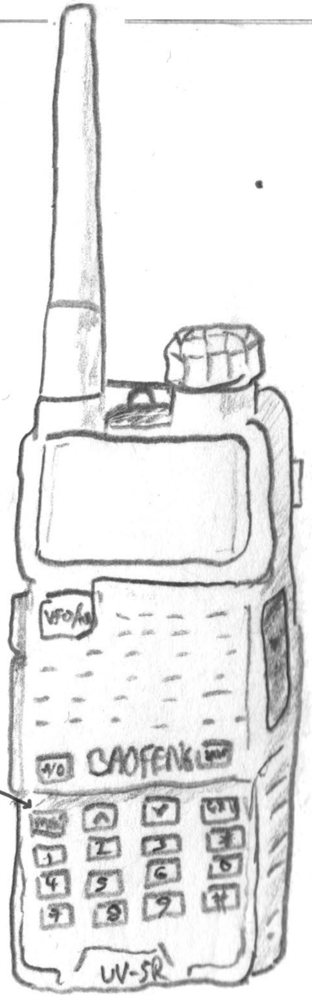
the result is a frequency modulated signal where up, down up, becomes higher(freq), lower, higher this resulting modulated signal is then transmitted.

to receive a radio does essentially the same but in the opposite direction. A modulated signal is picked up, then the carrier wave is subtracted leaving only the audio wave.

ABOUT THE UV5-R

The transceiver used here is the Baofeng UV5-R dualband radio which operates between 130.00-174.00 MHz and 400.00-480.00MHz

‘The (Chinese) Radio Documentation Project’ offers translated and annotated manuals for a variety of radios, including the UV-5R.
See:
http://radiodoc.github.io/

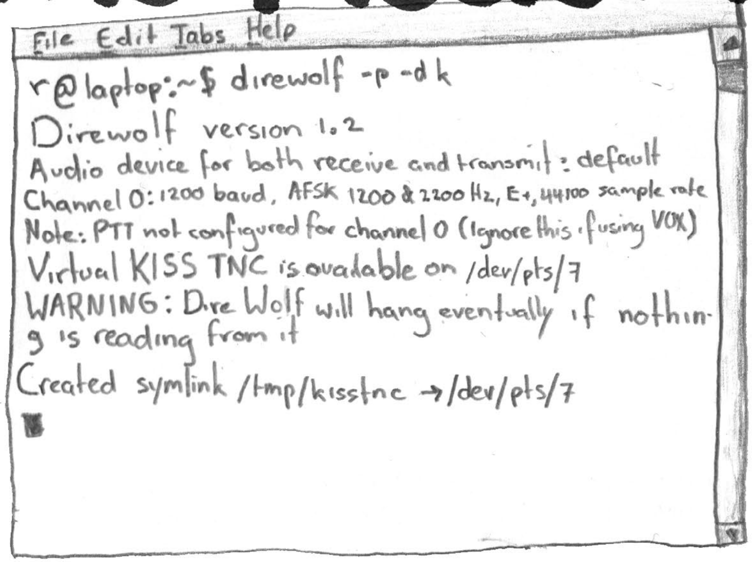


ACTIVATING VOX

To automatically transmit without having to press a button you must activate VOX, Voice OPerated Switch.

To turn on the VOX press MENU then 4, then MENU and use the UP and DOWN arrows to set it to a value of 2. To save press MENU again and then EXIT to leave the menu.

THE modem

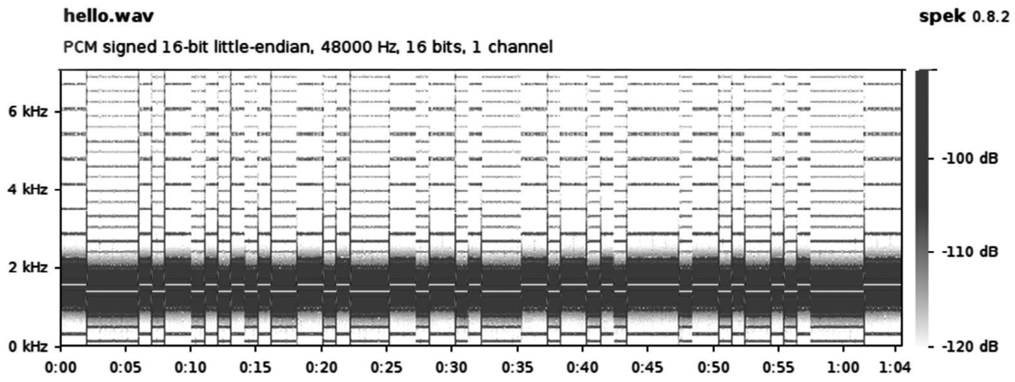


```
r@laptop:~$ direwolf -p -d k
Direwolf version 1.2
Audio device for both receive and transmit: default
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, E+, 44100 sample rate
Note: PTT not configured for channel 0 (Ignore this if using VOX)
Virtual KISS TNC is available on /dev/pts/7
WARNING: Dire Wolf will hang eventually if nothing is reading from it
Created symlink /tmp/kisstnc →/dev/pts/7
█
```

Modem stands for Modulator/Demodulator. In our case it is a piece of software that controls the computer's sound-card. Since we are working with digital data the modem is required to convert the digital information (1s and 0s) into audible, analog sound waves (which the radio can transmit). The modem does this by using so called AFSK (Audo Frequency Shift Keying) which is a fancy way of saying that it changes the tone (frequency) of the sound based on whether the bit is a 1 or 0.

The standard for packet radio is 1200 bits per second ('baudrate' in modem jargon) where the '1' becomes a 1200Hz tone and the '0' becomes a 2200Hz tone.

One of the simple modems we can use is an application called 'MINIMODEM' which can be used to encode or decode any kind of data into sounds.



^a spectrograph of 8 bit ascii HELLO modulated into sound.

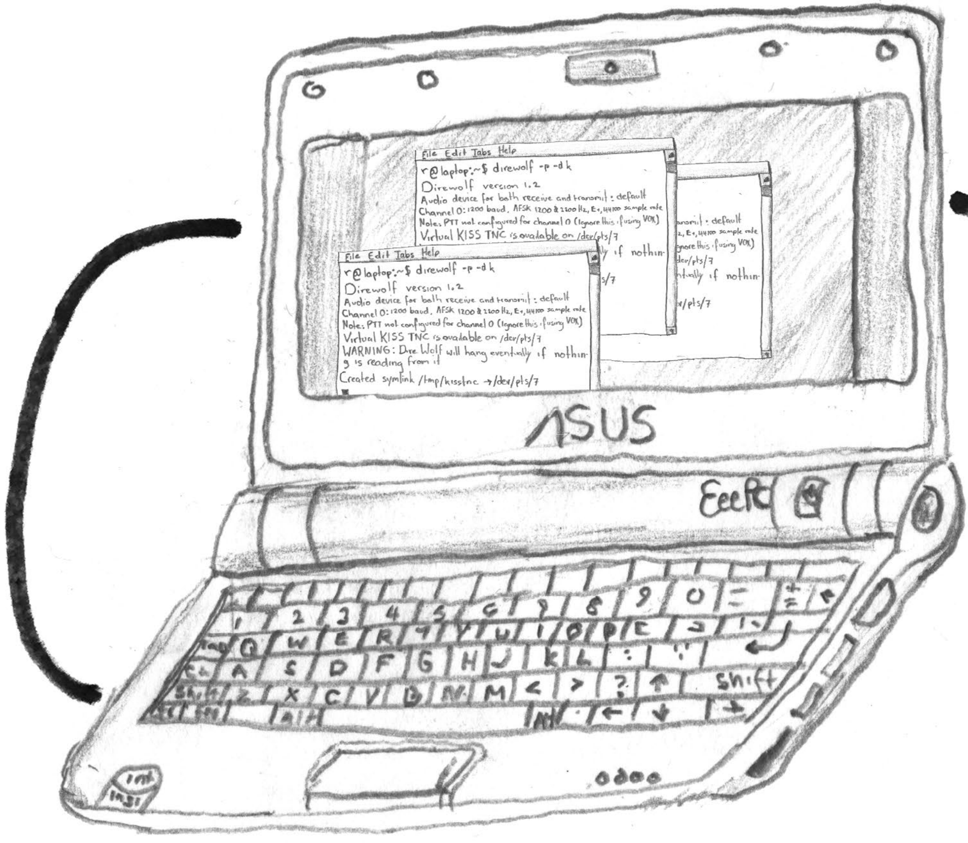
The binary of that looks like this:01101000 01100101 01101100 01101100 01101111

The spectrograph was produced with the following command:

$ echo 'hello' | minimodem -t 1 --ascii -f hello.wav

Afterwards it is rendered with 'SPEK'

the computer

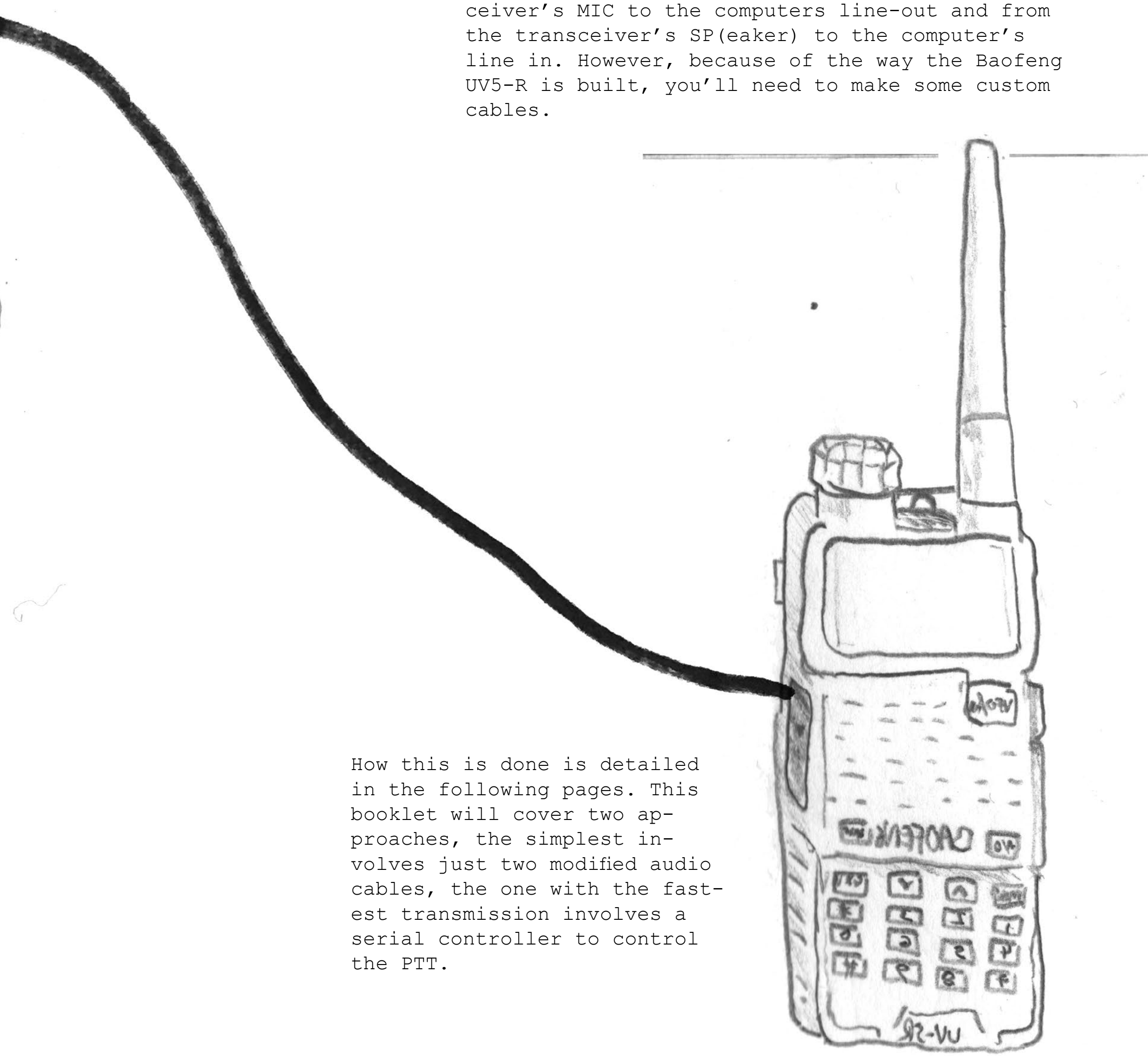


```
r@laptop:~$ direwolf -p -d k
Direwolf version 1.2
Audio device for both receive and transmit : default
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, E+, 44100 sample rate
Note: PTT not configured for channel 0 (Ignore this if using VOX)
Virtual KISS TNC is available on /dev/pts/7
WARNING: Dire Wolf will hang eventually if nothin-
g is reading from it
Created symlink /tmp/kisstnc -> /dev/pts/7
```

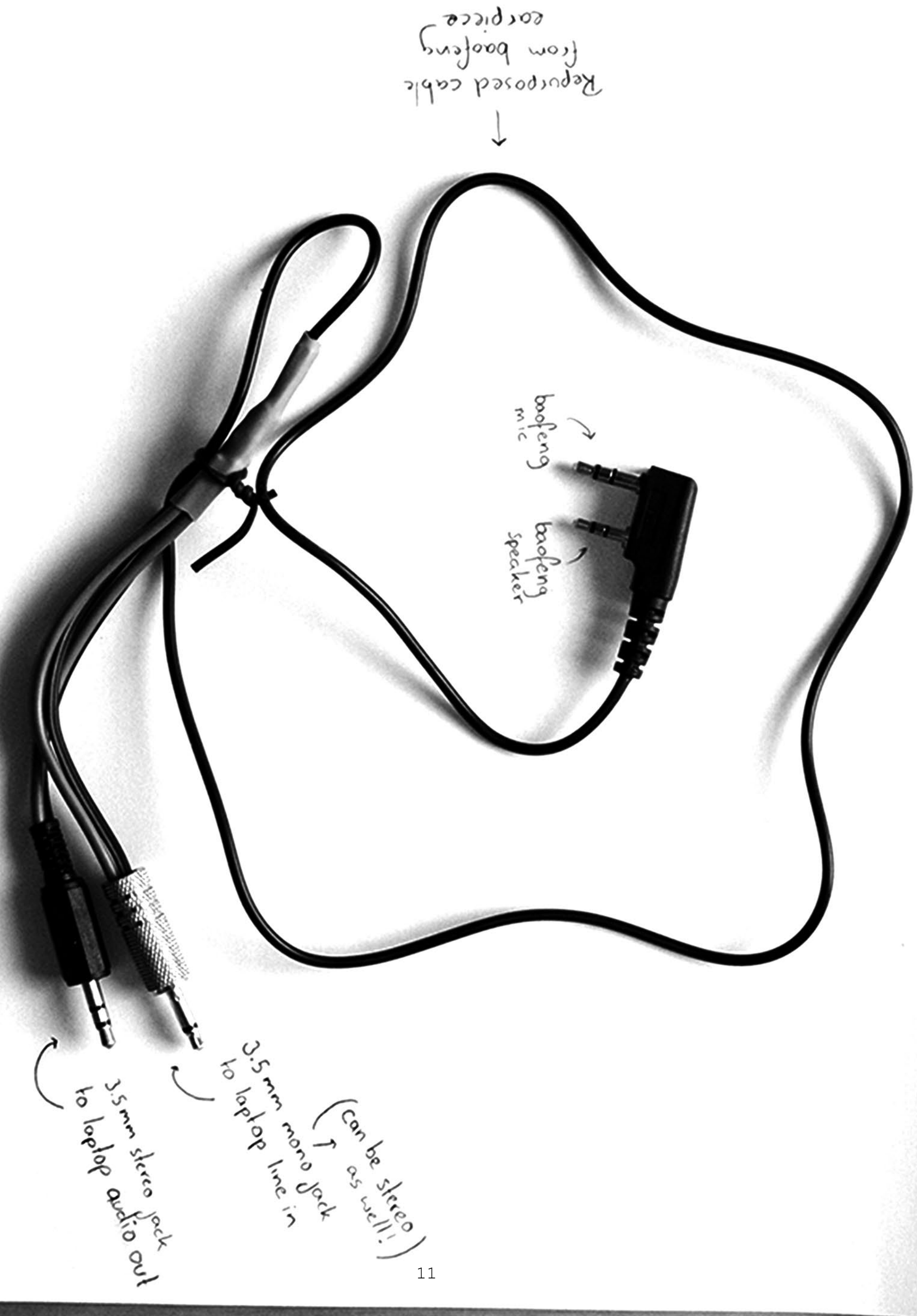
The centerpiece of the trinity of packet radio. The computer interfaces the transceiver to the software modem, bypassing the need for a hardware modem. This computer may have any shape or size, as long as it has a sound card and either line-in and line-out jacks or a single combined in-and-output jack (TRRS). This booklet furthermore assumes that the computer is running a Gnu/ Linux distribution. That is because many of the applications mentioned have been written for Gnu/Linux based systems and are thus well documented and supported. Once one gets the hang of it Gnu/Linux based systems are easier to play around with (you're actually allowed to!) and more flexible for using in non-standard ways, which is exactly what will be happening.

hooking up

Connecting the transceiver to the computer is as easy as connecting an audio cable from the transceiver's MIC to the computers line-out and from the transceiver's SP(eaker) to the computer's line in. However, because of the way the Baofeng UV5-R is built, you'll need to make some custom cables.

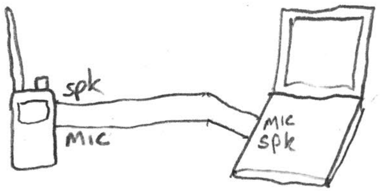


How this is done is detailed in the following pages. This booklet will cover two approaches, the simplest involves just two modified audio cables, the one with the fastest transmission involves a serial controller to control the PTT.

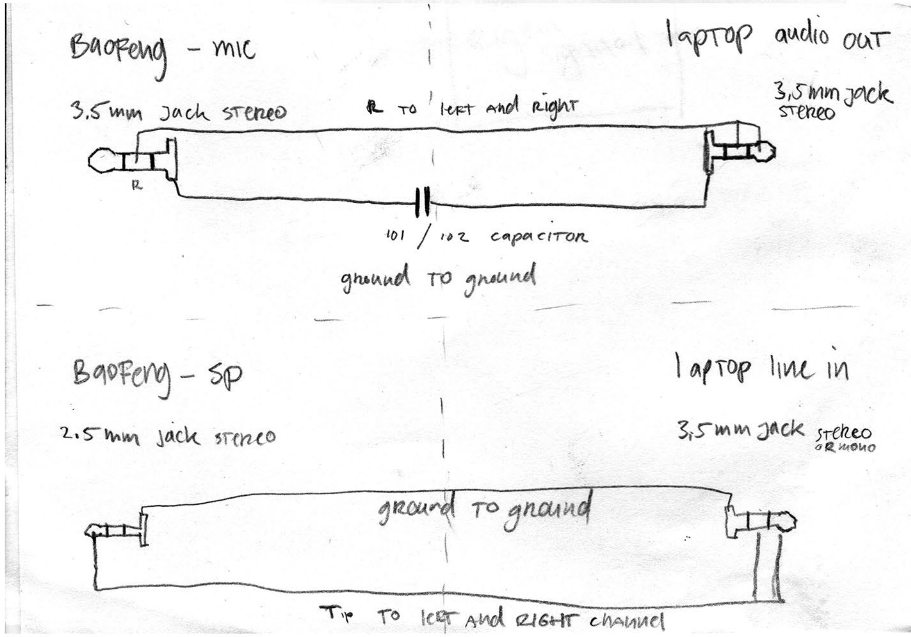


minijack to Baofeng

You can re-purpose the cable of the ear-piece that comes with the Baofeng to make the required 'Kenwood-type'(2.5mm / 3.5 mm) connection to the radio.



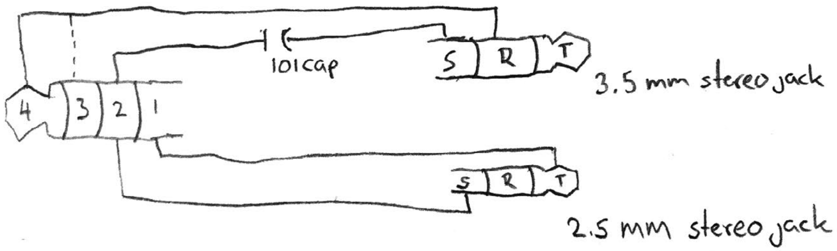
Schematics to use with computers that have a separate line-in and line-out ports.



TRRS to Baofeng

Tip Ring Ring Sleeve

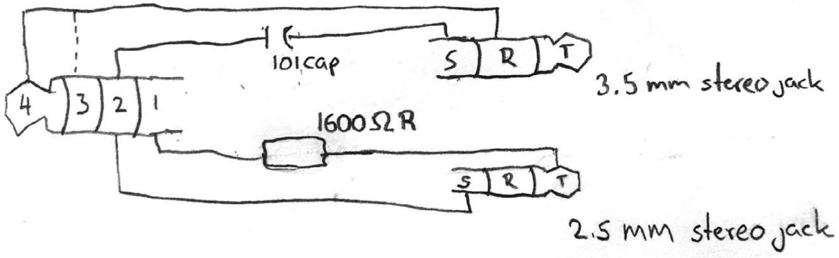
A schematic for connecting the Baofeng to another device using a TRRS connector

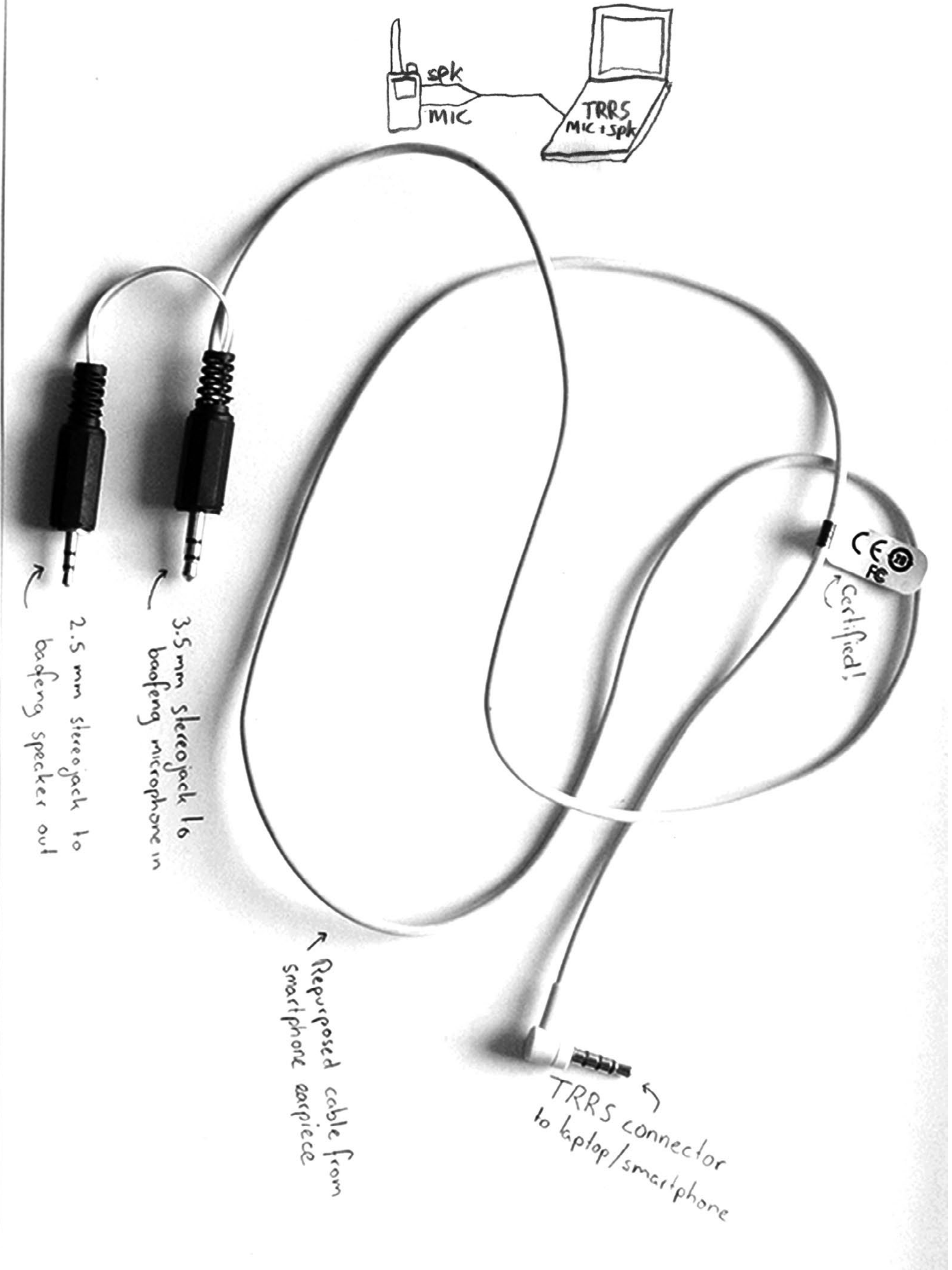


4 left audio
3 right audio
2 ground
1 microphone

Apple

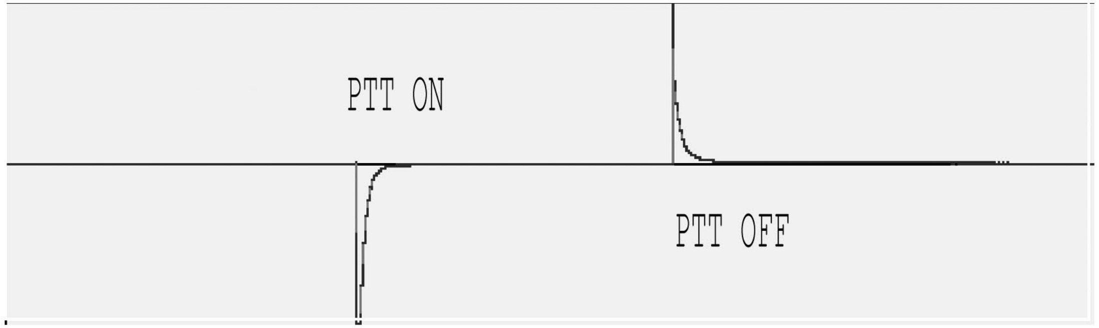
Apple devices require an impedance of 1600 ohm for the microphone to work so you'll need to add a resistor




spk
MIC
TRRS
MIC+spk
2.5 mm stereojack to baofeng speaker out
3.5mm stereojack to baofeng microphone in
Certified!
Repurposed cable from smartphone earpiece
TRRS connector to laptop/smartphone

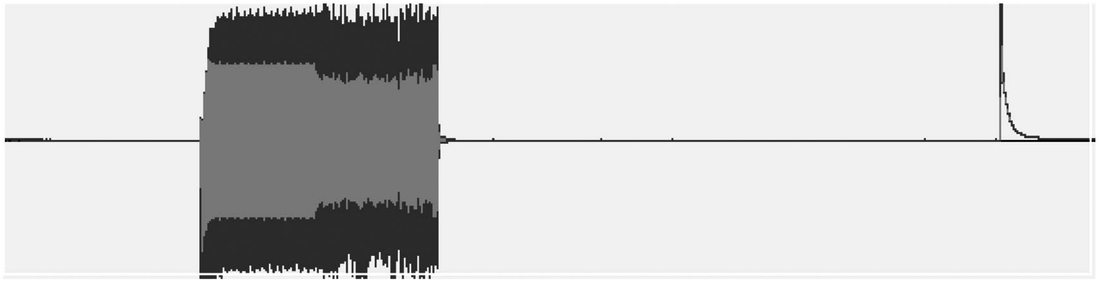
Fixing the VOX

The capacitor between the grounds is required to prevent the radio from always transmitting when connecting the radio to the laptop. That is because the PTT (push-to-talk) is triggered internally by connecting the grounds of the speaker and the mic. The capacitor ensures that the PTT is only activated when there is an audio signal on the cable.



A recording of pressing and then releasing the vox. The spikes you see are the capacitor working. Recorded with 'AUDACITY'

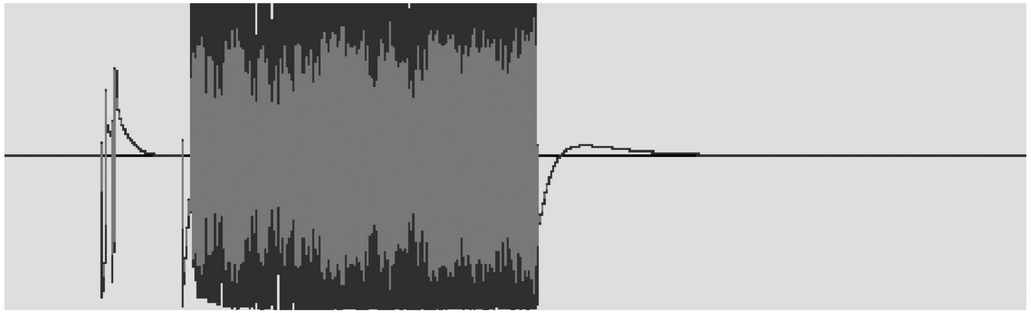
The whole sequence of activating the vox and discharging the capacitor takes a few milliseconds during which some transmitted information can get lost. Thats why many packet radio softwares have a function to send a tone to activate the transmitter before sending out the actual data (called TX delay).



Tx delay| Data |---------Tx tail +Vox delay -------|

The above image shows what the audio of a radio packet looks like. The first bit is the TX delay, then the actual packet, followed by a 'TX tail', a small extra tone to keep the channel open between packets. The long silence you see is the result of the VOX, which keeps the transmitter open for a few seconds after hearing nothing. That is why using a hardware TNC to activate/ deactivate the PTT results in lower latency.

The image below shows the audio overdriven or 'clipping', resulting in noise and thus lost data.



Configuring audio levels of the sound-card is very important since audio is what carries the signals. Make sure that both the input and output audio levels are ok so that the audio is neither too loud (clipping = loss of data) or too soft (might not activate VOX)

Steps to configure the audio:
Turn the volume knob on the radio transceiver to minimum In 'PAVUCONTROL' or 'ALSAMIXER' set the gain for the internal microphone to about 18 in alsamixer (or on 'base' in pavucontrol).

Record the incoming audio (for example using 'AUDACITY') from the radio and adjust the volume knob on the radio so that the audio doesn't clip but is clearly audible.

Check your output levels by transmitting audio and using another radio to hear if the signal that is transmitted is not is not overdriven. Usually setting the output volume to around 50% is about right.

Adjusting the volume knob

Hardware PTT

VOX is nice, but in cases where you need to send packets back and forth often (like during a TCP handshake) it is hard to get the transmit and receive timing windows right. One way to prevent collisions of transmitted and received signals is to only activate your PTT when you actually send data. This method would also work for walkie-talkies without VOX ;)

Most software modems have an option to trigger PTT via a serial port (/dev/tty or /dev/ttyUSB) so here is a basic recipe using common components to make a PTT ‘switch’ for the Baofeng UV5-R.

Using two transistors, it works by to closing the ground loop between the Baofengs SPK and MIC whenever there is a signal on the serial interface’s RTS pin. A similar approach can be taken for other radios.

<u>PARTS</u>
-FTDI232 USB to Serial break out board (+USB cable)
-TRRS/Minijack-to-BAO cable as previously build
-PNP Transistor (BC557)
-NPN Transistor (BC547)

<u>ASSEMBLY</u>
Hook up your transistors to the FTDI232

Connect the collector of the BC547 NPN transistor to the ground of the speaker jack of your TRRS/Minijack-to-BAO cable

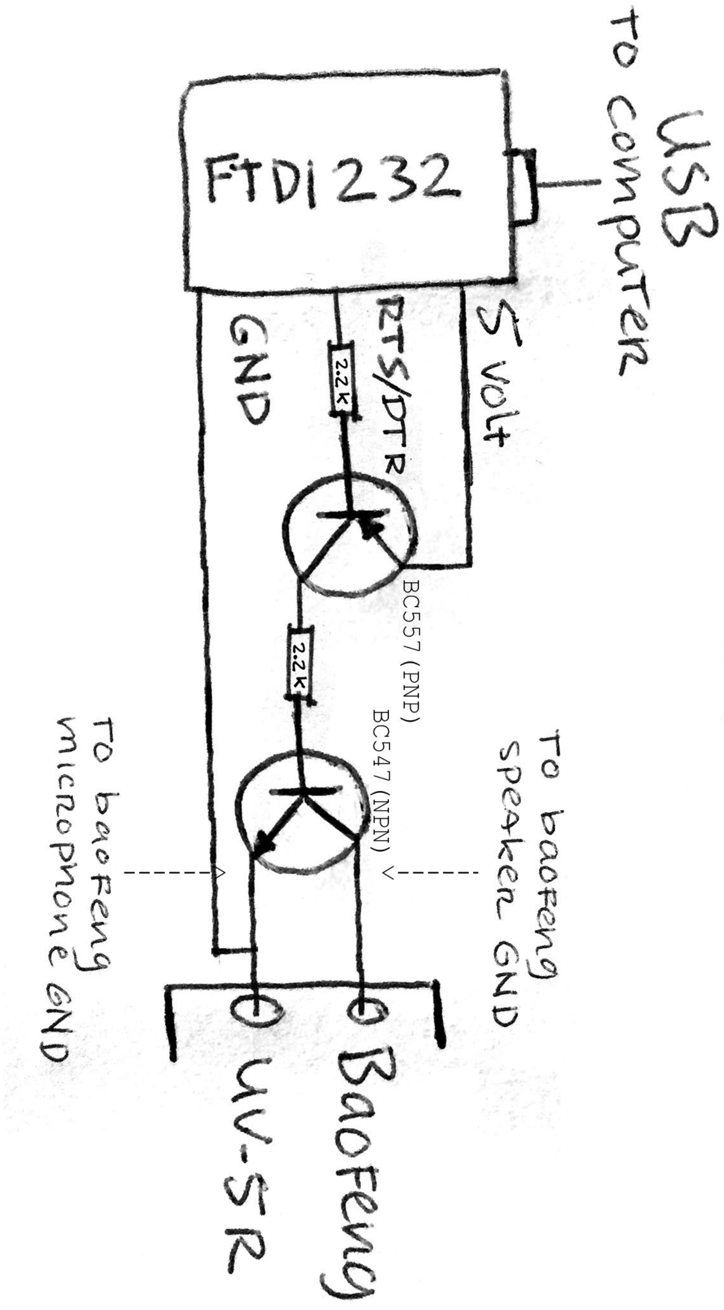
Connect the emitter of the BC547 NPN transistor to the ground of the microphone jack of your TRRS/Minijack-to-BAO cable

Connect the emitter of the BC547 NPN transistor to the ground of the FTDI232

<u>CONFIGURE</u>
Edit the direwolf configuration file to enable the PTT switch (default location ~/direwolf.conf):
add to ‘PTT Settings’

```
PTT /dev/ttyUSB0 RTS
```

Where ttyUSB0 is the name of the USB-to-Serial adaptor (use **$ dmesg** to find out the name) and RTS is the name of the port on your adaptor (Some only have a ‘DTR’ port, in that case use DTR instead of RTS).



Packet switching

ISO's OSI Model

International Organization for Standardization's Open Systems Interconnection model

Protocol Layers

1	physical	cable	wifi radio	walkie-talkie
2	datalink	Ethernet	IEEE 802.11	AX25
3	network	ip	ip	ip
4	transport	tcp	tcp	tcp
5	session			
6	presentation			
7	application			

irrelevant for the scope of this book

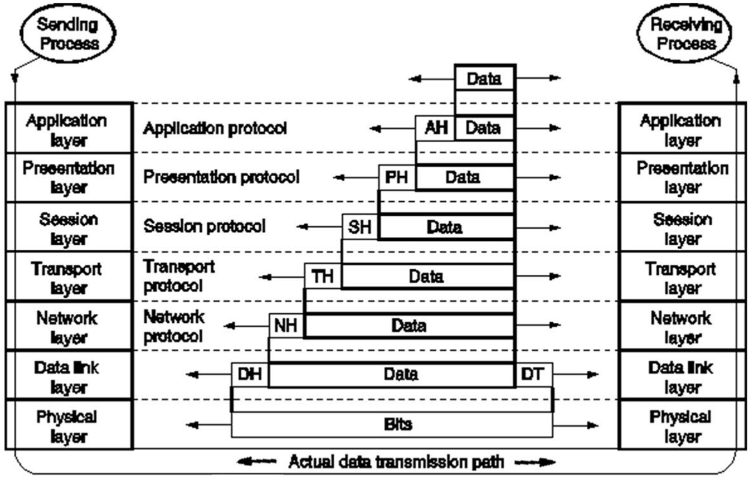
THE ISO OSI MODEL

A theoretical model that describes how information moves from an application running on one networked computer to an application running on another networked computer.

It describes everything from the protocols handling the physical layer (controlling voltages of interfaces) to protocols that work on the application layer (how user requests get handled).

The model is neat because it allows each protocol to be swapped out by another of the same layer, without having too much influence on the rest of the stack. In other words when you surf to a website, your browser doesn't (need to) care if that connection is made via ethernet, wifi, or in our case walkie-talkie.

Similarly once you've set up your walkie-talkie as a network interface you can run all networking applications over it (ping, telnet, ssh etc).



more info: https://fmfi-uk.hq.sk/Informatika/Distribuovane%20Systemy/knihy/ICN/ch1s4.htm

AX.25

Some good documentation of the protocol:
http://www.tldp.org/HOWTO/text/AX25-HOWTO
ftp://ftp.tapr.org/projects/n8urbook/n8ur-dr1.pdf

The protocol that makes the software modems work is 'AX.25'. It sits in the Layer 2 of the OSI model, the layer that controls hardware and defines hardware addresses (such as MAC-addresses). The interesting thing is that the AX.25 drivers been part of the Linux kernel for a long time, so Linux machines support it 'out of the box'. What AX.25 allows us to do is to turn our sound card into, another network interface, with an IP-address so that fits neatly into the Linux networking stack. AX.25 originated around 1980 in amateur radio circles (AmateurX25, X25 being a common networking protocol at the time). Therefore this software has a lot of amateur radio typicalities. One of which is the usage of an amateur radio 'callsign' as a hardware address (MAC-address).

To use AX.25 install some utilities first:

```
$ sudo apt-get install ax25-apps ax25-utils ax25-tools
libax25
```

Then configure the file that defines the interface in:

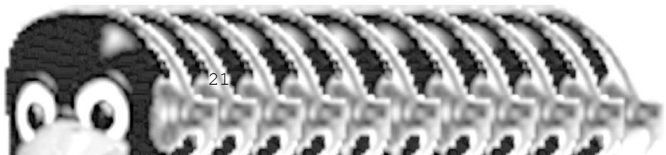
```
$ sudo nano /etc/ax25/axports
```

In the uncommented, empty line in /etc/ax25/axports put:

```
    # name callsign speed paclen window description

    ax0      TESTIN-1    38400  255    7    description
```

The most important things are the interface name (ax0) and the callsign. The interface name works like any other network interface name (en0, en1 on mac, eth0, wlan0 linux)
Callsign is what replaces the 'hardware' address. AX.25 specs require your callsign to be six uppercase characters, with an additional dash and (any) decimal number. For example: TESTIN-1. Remember, hardware addresses are supposed to be unique, so pick AND agree upon a UNIQUE callsign!



soundmodem

Another software modem, modulating data into sound. 'SOUNDMODEM' dates back to 2002, so it is pretty old but should work!
Install: **$ sudo apt-get install soundmodem**
Configure: **$ sudo soundmodemconfig**

Create a new configuration, and give it a name:
File->New->Configuration
Tab IO:
Mode: "alsa"
Audio Driver: default (type it, it's not selectable)
Half Duplex: tick it
PTT Driver: Ignore, since we're using the transceiver's VOX as a PTT driver
Tab Channel Access:
TxDelay: "300" (this is the TX delay discussed earlier, you have to test your setup to see the best value, all values in ms)
Slot Time: "100"
P-Persistence: "40"
Full Duplex: (Not selected)
TxTail: "10"
File->New->Channel
Double-Click on "Channel 0"
Tab Modulator:
Mode: "afsk"
Bits/s: "1200"
Frequency 0: "1200"
Frequency 1: "2200"
Differential Encoding: (selected)
Do the same for Tab Demodulator
Tab Packet IO:
Mode: "KISS"
File: "/dev/soundmodem0"
Unlink File: (selected)
To save the config:
File -> Save
To run it: **$ sudo soundmodem**
then: **$ sudo kissattach /dev/soundmodem0 ax0 10.0.0.2**

(where ax0 is the interface name you put in /etc/ax25/axports and the IP-address is the range you plan on using)

DIREWOLF

‘DIREWOLF’ is an actively developed software modem. To get the latest version (1.3 J at the time of writing) it is best built from source.

INSTALL

Install direwolf via your package manager:

```
        $ sudo apt-get install direwolf
```

Install the latest development version from the git: https://github.com/wb2osz/direwolf

```
 $ sudo apt-get install build-essential libasound2-dev
 $ cd ~
 $ git clone https://www.github.com/wb2osz/direwolf
 $ cd direwolf
 $ git checkout dev
 $ make
 $ sudo make install
 $ make install-conf
```

To use direwolf you first need to run the modem and then make a networkdevice with an IP-address for it (see AX.25 page *).

RUN

Start direwolf with no colors, print debug info to terminal (packets, soundlevels)and using the specified config:

```
        $ direwolf -t 0 -d k -c ~/direwolf.conf -p
```

In a new virtual terminal, attach a network device with an IP-address to your soundcard:

```
        $ sudo /usr/sbin/kissattach `ls -l /tmp/kisstnc
| gawk ‘{print $11}’` ax0 10.0.0.1
```

(where ax0 is the name of your interface set in /etc/ax25/axports.

If you get an error, try this to attach KISS to your ax0 interface:

```
$ sudo /usr/sbin/kissattach /dev/ptmx ax0 10.0.0.1
```

It will output something mentioning a pts number (like /dev/pts/5). Use that number in the next command:

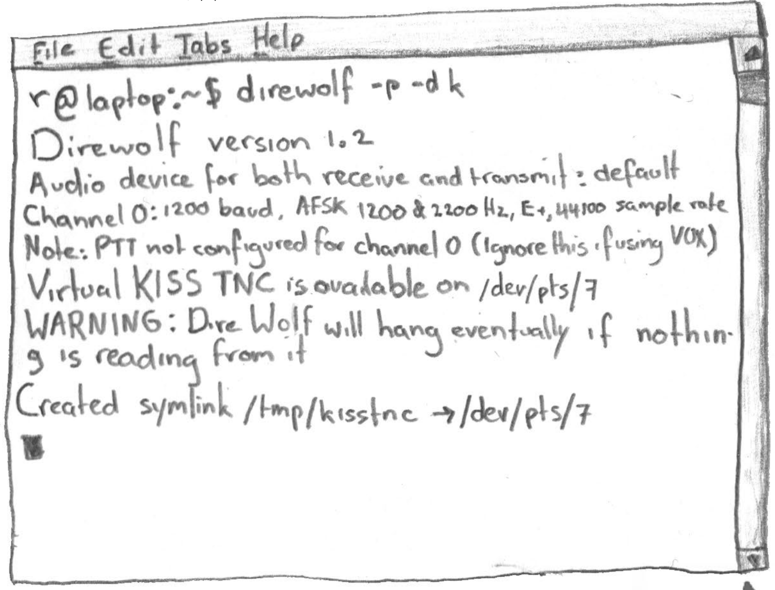
```
        $ sudo mkiss /tmp/kisstnc /dev/pts/5
```

If all is well you should see no message after that last command. If not, link mkiss directly to the pseudo terminal direwolf creates for example:

```
        $ sudo mkiss /dev/pts/4 /dev/pts/5
```

However, if you now try to make a network connection over that IP-range (try: **$ ping 10.0.0.2**) you should hear a distinct screech coming from your speakers. That is the sound of success!

Now, 1200 Baud is not that fast, if you don’t get any ping backs, send ping’s with a lower interval, like 25 seconds: **$ ping 10.0.0.2 -i 25**(and check your sound levels ;))



```
r@laptop:~$ direwolf -p -d k
Direwolf version 1.2
Audio device for both receive and transmit: default
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, E+, 44100 sample rate
Note: PTT not configured for channel 0 (Ignore this if using VOX)
Virtual KISS TNC is avalable on /dev/pts/7
WARNING: Dire Wolf will hang eventually if nothin-
g is reading from it
Created symlink /tmp/kisstnc → /dev/pts/7
```

Official Direwolf logo

Getting started

Now that you've set up both the software and the hardware, some things to try and some things to help debug.

Before you start transmitting:

Make sure all traffic to other radios is routed over your AX25 interface:

```
$ sudo route add -net 10.0.0.0/24 ax0
```

where the IP (10.0.0.*) and ax0 is whatever you've configured before).

Add other nodes you know to your ARP table so the machines don't need to start a lengthy ARP negotiation:

```
$ sudo arp -s 10.0.0.1 -H ax25 TESTIN-1
```

where the IP (10.0.0.1) and TESTIN-1 is the IP-address of the other host and his/her callsign defined in /etc/axports.

To check if your AX.25 network interface is created issue:

```
$ ifconfig
```

To stop 'DIREWOLF', you also need to stop KISSATTACH and MKISS to free up the (virtual) devices they create using the following command :

```
$ sudo killall direwolf; sudo killall kissattach; sudo killall mkiss
```

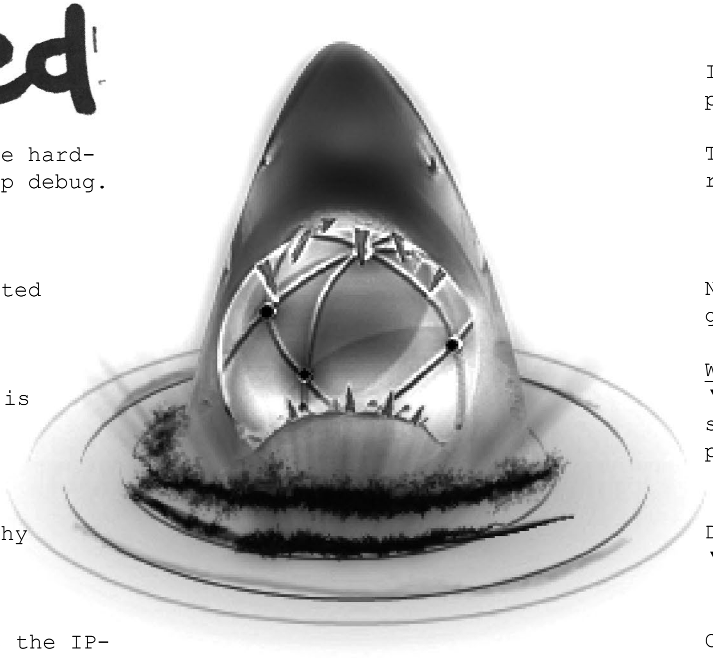
Send texts or files over UDP using 'NETCAT':

on the transmission side:

```
$ echo "something to send" | nc -u 10.0.0.1 5555
```

the IP-address is dependent on the other's axports settings and the port is arbitrary as long as it is the same on the receive side:

```
$ nc -l -u -p 5555
```



If there's a webserver running on one of the nodes just point your browser towards it.

To run a webserver (watch out this will serve the current directory):

```
$ python -m SimpleHTTPServer 80
```

Next up, try to 'WGET' some files, start an SSH session, go wild!

WIRESHARK

'WIRESHARK' can listen to your network interfaces and show you data packets and their content including your packet radio interface. To install:

```
$ sudo apt-get install wireshark
```

Depending on your privileges you MIGHT need to run 'WIRESHARK' as root:

```
$ sudo wireshark
```

Choose the interface you want to listen to, in our case 'ax0' to view ax25 packets.

More info on the ax25 protocol:
http://www.tldp.org/HOWTO/text/AX25-HOWTO

Tweak the AX.25 protocol parameters to improve or optimize performance. The following settings can be adjusted in all software modems. Especially tweaking SLOTTIME and TXDELAY seem to make a difference. In direwolf the following needs to be added to direwolf.conf:

```
#TX Properties:
# all values except PERSIST are x10 ms
DWAIT 0
SLOTTIME 70
PERSIST 63
TXDELAY 150
TXTAIL 10
```

The settings described here are optimized for using VOX. When using a PTT-driver a shorter SLOTTIME and TXDELAY are fine.

Internet Gateway

A GATEWAY TO THE REAL INTERNET

Local networks are fun, but browsing through the REAL WEB with 1200baud is even more fun! You need two computers and two baofeng radios. One of the two computers will be set up as a gateway, with one network interface connected to the internet and one AX.25 network interface.

The other computer, the client, will only have an AX.25 connection and will set the gateway computer as its internet gateway.

GATEWAY
First connect your gateway machine to the internet using ethernet (or wifi connection...).

Do the following to setup a minimal gateway to the internet, assuming eth0 is your WAN (wide area network) connecting the gateway to the internet and ax0 is your packet radio link, the LAN (local area network).

Enable IP Forwarding:

```
       $ echo 1 > /proc/sys/net/ipv4/ip_forward
```

Flush any previous nat, mangle etc from iptables:

```
       $ iptables -F
       $ iptables -t nat -F
       $ iptables -t mangle -F
```

Enable NAT:

```
       $ iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
       $ iptables -A FORWARD -i ax0 -j ACCEPT
```

CLIENT
On the client, to use the gateway:
Set the default ip-route to the gateway you just configured where the IP-address is the ax25 IP of server:

```
       $ sudo route add default gw 10.0.0.2
```

Set an external nameserver (e.g. 8.8.8.8) or just take the one that was assigned to WAN interface of the gateway.(add 8.8.8.8 directly in the /etc/network/interfaces after ‘dns-nameservers’.

Enjoy!

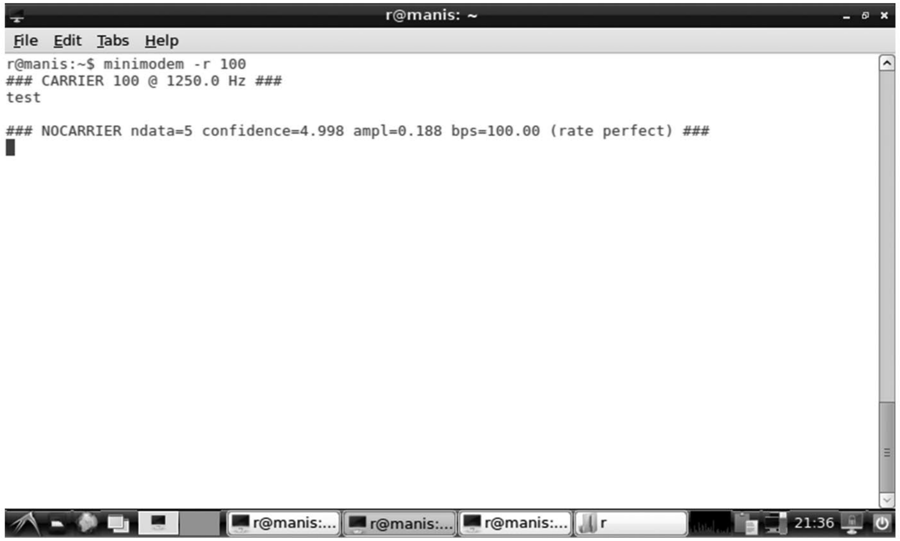
Silicon Graphics ‘WebFORCE’ Internet Gateway Splash Page, 1998-06-15, Source: http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?coll=0650&db=bks&srch=&fname=/SGI_Admin/Gateway_IG/sgi_html/ch03.html, retrieved March 3, 2016.

minimodem

The following sections will cover some alternatives to AX.25. One the most simple and robust software modems we can use is 'MINIMODEM'. Minimodem converts input from the command line to sounds (which via your audio cables are then output via the radio). It can both receive and transmit (encode/decode). Opening two terminals with minimodem while having a radio connected to your soundcard is the simplest way to do packet radio. Just make sure to double check your audio input/output levels. Install minimodem:

$ sudo apt-get install minimodem

Rx



```
r@manis:~$ minimodem -r 100
### CARRIER 100 @ 1250.0 Hz ###
test

### NOCARRIER ndata=5 confidence=4.998 ampl=0.188 bps=100.00 (rate perfect) ###
```

RUN

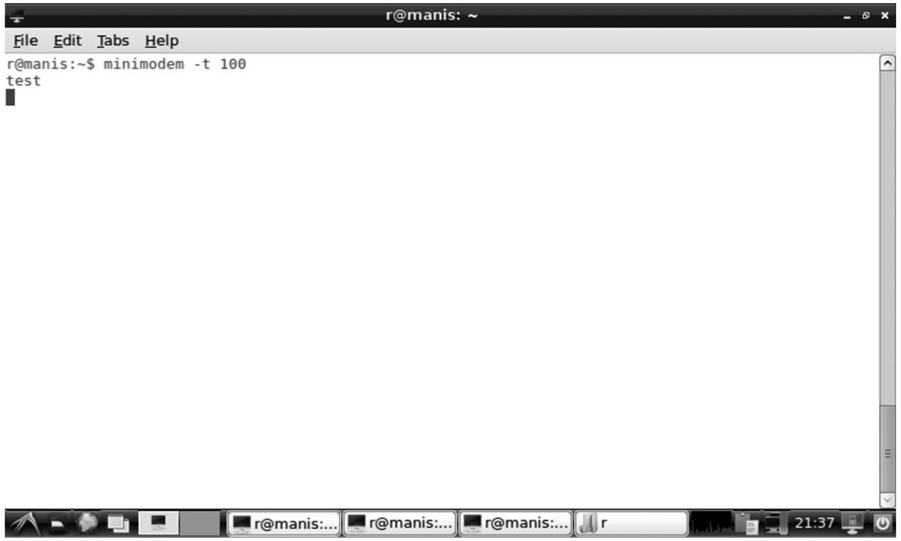
In one terminal start 'MINIMODEM' in interactive Receive (Rx) mode:

$ minimodem -r 100

In the other terminal window start 'MINIMODEM' in interactive Transmit (Tx) mode to send data, in this case the text 'test':

$ minimodem -t 100

Tx



```
r@manis:~$ minimodem -t 100
test
```

The receiving end will now see the text 'test' appear one letter at a time.

minimodem advanced

Besides sending data to yourself, you can also use minimodem ‘Over-The-Air’ to send files or chat with others and what not.

WIRELESS: SPEAKERS AND MICROPHONES
Try chatting to your neighbor using your laptops built in speakers and microphone!

WIRELESS: RADIO
Next step is using two radios and two computers. With the audio levels set correctly and the radios tuned in on the SAME frequency you can send data over radio! Do not use the emergency channels though, this WILL land you in jail.
The Baofeng radio’s need a trigger (Tx Delay) to open up the Push To Talk function and transmit. If you have a hard time transmitting, send a single character followed by your ‘real’ message. Or you can write a small script to take care of this ;)

PIPING
You can also send files by using minimodem in pipeline mode. To send a copy of today’s New York Times at a speed of 300 baud type:

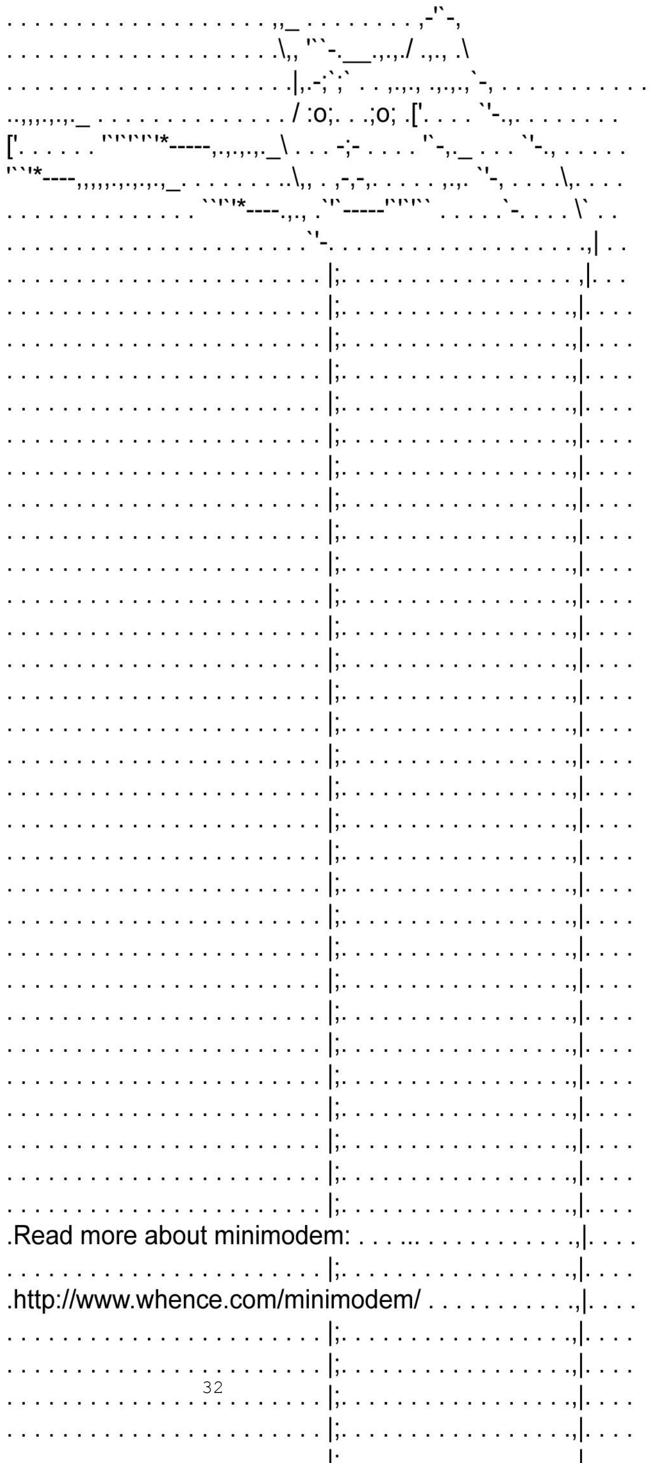
```
   $ curl http://mobile.nytimes.com |  minimodem -t 300
```

BINARIES
Send cat pictures at a speed of 300 baud (yes that will take forever):

```
   $ cat catphoto.jpg | minimodem -t 300
```

ASCII ART
If waiting for images is too boring, you can ‘cat’ your favorite ascii cat into minimodem:

```
   $ cat asciiart.txt | minimodem -t 300
```



.Read more about minimodem:

.http://www.whence.com/minimodem/

SSTV

SLOW SCAN TELEVISION

Just like minimodem, 'SSTV' is technically not packet radio. We will include it in this booklet because it is a good way of transmitting images over radio. Whereas transmitting a JPEG image via minimodem might easily result in a corrupted image (or even un-renderable image if the header is corrupted) SSTV's specification is pretty tolerant to noise and will always produce some result.

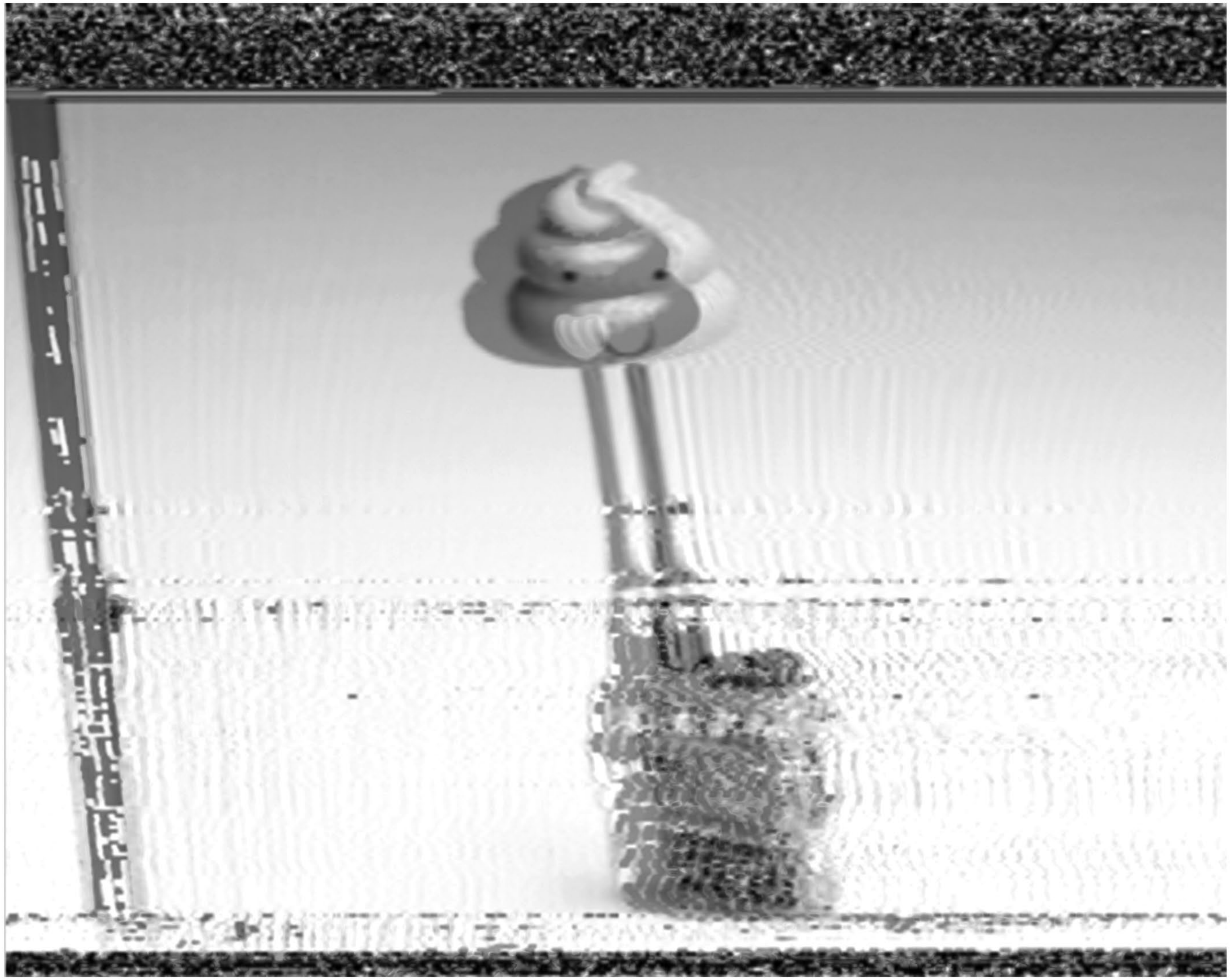
INSTALL

```
        $ sudo apt-get install qsstv
        $ sudo qsstv
```

Mac OSX:
KD6CJI MULTISCAN 3B SSTV
http://www.qsl.net/kd6cji/

SSTV YOUTUBE SCENE
Youtube censorship evasion using SSTV. Search for "sstv porn".
Still hot in 2016!

More info (and a really nice WIKI):
http://www.sigidwiki.com/wiki/Slow-Scan_Television_%28SSTV%29



Sources

THE (CHINESE) RADIO DOCUMENTATION PROJECT
http://radiodoc.github.io/
Repository of high quality and in-depth User Manuals for Chinese Handheld Two-Way radios. Currently only has information on the Baofeng UV-5 R. Unsure if actively maintained.

AUDACITY - http://www.audacityteam.org/
Free, open source, cross-platform software for recording and editing sounds.

```
        $ sudo apt-get install audacity
```

SOUNDMODEM
Mirror to the old homepage: http://soundmodem.vk4msl.yi.org/
Good guide on how to use and configure:http://www.george-smart.co.uk/wiki/AX25_Soundmodem
Multiplatform soundcard modem written by Thomas Sailer, HB9JNX/AE4WA. The software allows a standard PC sound-card to be used as a packet radio “modem” with all processing done on the main computer CPU.

```
        $ sudo apt-get install soundmodem
```

DIREWOLF
Git repository: https://github.com/wb2osz/direwolf
Community and mailing list: https://groups.yahoo.com/neo/groups/direwolf_packet/info
Dire Wolf is a software “soundcard” modem/TNC and APRS encoder/decoder that is under active development. It can be used for a wide variety of packet radio applicaions and has a lively community. Best built from source. Be sure to get the dependencies first:

```
        $ sudo apt-get install libasound2-dev build-essential
```

AX.25
A Good reference guide: http://www.tldp.org/HOWTO/text/AX25-HOWTO
More general reference on packet radio: ftp://ftp.tapr.org/projects/n8ur-book/n8ur-dr1.pdf
Official homepage: http://www.ax25.net/
Protocol specification: https://www.tapr.org/pub_ax25.html
Wiki documenting various applications to use AX.25:
        http://www.linux-ax25.org/wiki/Main_Page
The family of protocols used for packet radio.It makes it possible to encapsulate TCP/IP frames into AX.25 frames so you can do internet over walkie talkie. It is still part of the GNU/Linux kernel as of today.

```
        $ sudo apt-get install libax25 ax25-apps ax25-tools
```

WIRESHARK
Homepage: https://www.wireshark.org/
The Swiss Amry knife of packet inspection. See whats happening behind the screens of your network interfaces! Free, open source and cross-platform. Go Deep.

```
        $ sudo apt-get install wireshark
```

KISS/MKISS (Keep It Simple Stupid / Multi KISS)
Protocol specification: http://www.ax25.net/kiss.aspx
Series of protocols designed to communicate with a (virtual TNC). Operates on layer 1 of the OSI stack and communicates between network interface and the soundcard. The ‘kissattach’ application is used to do this. Installed via ax25utils/libax25

HOMEBREW
homepage: http://brew.sh/
git repository: https://github.com/Homebrew/homebrew
OSX’s missing package manager. Features a great deal of essential GNU/Linux tools ported to OSX. Notable ports include ‘WGET’ that is missing from the more recent OSX versions... install:

```
$ /usr/bin/ruby -e “$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)”
```

MINIMODEM - http://www.whence.com/minimodem/
General-purpose software audio FSK modem for GNU/Linux systems. Can be run ‘non-realime’ on OSX via ‘HOMEBREW’.Can be used to decode default packet radio using $ minimodem -r 1200 --ascii. Minimodem does not communicate with the kernel, however and output is only printed to the terminal.

```
 $ sudo apt-get install minimodem
```

MINIMODEM OSX

```
        $ brew update
        $ brew install minimodem
```

Homebrew will throw errors because minimodem depends on Pulseaudio that doesn’t work on OSX. You can ignore this but are limited to encoding/decoding/modulating/demodulating with audio files, transmit “hello”:

```
        $ echo “hello” | minimodem -t 30 -f test.wav | afplay test.wav
```

receive (record audio):

```
        $ sox -d --channels 1 test.wav
```

then decode/demodulate:

```
        $ minimodem -r 30 -f test.wav
```

work in progress one-liner for receiving:

```
        $ sox -d --channels 1 test.wav -q | minimodem -r 30 -f test.wav
```

WGET
homepage: https://www.gnu.org/software/wget/
GNU Wget is a free software package for retrieving files using HTTP, HTTPS and FTP, the most widely-used Internet protocols. It is a non-interactive commandline tool, so it may easily be called from scripts, cron jobs, terminals without X-Windows support, etc.

```
$ sudo apt-get install wget
```

SSTV
Protocol description: http://www.barberdsp.com/files/Dayton%20Paper.pdf
Protocol description + handbook: http://www.sstv-handbook.com/download/sstv-handbook.pdf
Slow Scan Television is a method for picture transmission used by amateur radio operators to transmit and receive images. Really resilliant to interference and still used by the International Space Station to send images back to earth today.

```
        $ sudo apt-get install qsstv
```

        OSX > http://www.qsl.net/kd6cji/

SIGNAL IDENTIFICATION WIKI
http://www.sigidwiki.com
Wiki to help identify radio signals in the wild through audio and visuals.

HARDWARE PTT SWITCHES
Tom Karpiniec’s Serialport switch build:
https://karp.id.au/post/baofeng_interface_ptt/
Baofeng UV5-R wiring details: http://www.miklor.com/COM/UV_Technical.php
PTT Driver using a relay:
http://www.w7ay.net/site/Applications/cocoaModem/MiscInfo/Contents/PTT.html
More traditional drivers:
http://echolink.ru/downloads/ptt_to_computer.pdf
For pre-made PTT switches look for ‘Easy Digi’ on ebay:
http://www.aracc.org/easydigi!.pdf

NETCAT
http://nc110.sourceforge.net/
The TCP/IP Swiss army knife... Netcat is a simple Unix utility which reads and writes data across network connections, using TCP or UDP protocol.

“Finally an introduction into cutting-edge 80’s technology”

“Excellent overview […] but is this legal?”

“Outdated as soon as its printed, just like a real book about software!”