

Dire Wolf Radio Interface Guide

First rough draft - Jan. 2021

There is a surprising amount of confusion about interfacing Dire Wolf with radios. Perhaps it is because the documentation is scattered in different places. Here, I have attempted to gather it all together in one place, hopefully in an easy to use format.

Please share what you have discovered so others can benefit from your experience. The best way would be to open an issue in the “direwolf-doc” project (not the normal “direwolf” project) so nothing slips through the cracks.

I’m especially interested in what you found when using soundcards and USB PTT control built into some modern rigs.

Dire Wolf contains modems used to send Packet Radio & APRS digital data through ordinary voice transceivers.

Just for fun, you might want to try APRS with acoustic coupling. Place a receiver near your laptop computer and use the built-in microphone. Transceiver VOX could be used to transmit when the computer makes a sound. Yes, this actually works, just to prove that it can be done.

A better solution is to use wires for more predictable results and immunity from ambient sounds.



Audio & PTT



There are numerous ways to achieve this. If you have been using your computer to run NBEMS (FLdigi, etc.), PSK-31, WSJT-X (FT8, etc.), RTTY, or SSTV, you probably have the necessary connection already.

This guide contains information on various types of possible interfaces, configuration examples, and troubleshooting.

Contents

1	The Easy Way.....	5
2	Quick Start Guide for USB interfaces with PTT control.....	7
2.1	Linux	7
2.2	Windows.....	7
2.3	Other operating systems	8
3	Setting proper transmit audio level.....	9
4	Computer Audio	10
4.1	Special notes for USB Audio Adapters	11
5	Connection to Radio	13
6	What goes in between?	15
7	PTT	17
7.1	Serial Port	18
7.2	GPIO	21
7.2.1	Configuration - original (sysfs)	22
7.2.2	Configuration - new character device API (libgpod).....	23
7.3	Hamlib - CAT	25
7.3.1	Hamlib PTT Example: Use RTS line of serial port.....	26
7.4	VOX (Voice Operated Transmit).....	28
7.5	USB Audio Adapter GPIO - Linux.....	31
7.6	USB Audio Adapter GPIO - Linux.....	33
7.7	USB Audio Adapter GPIO - Windows	38
8	Troubleshooting	40
8.1	General.....	40
8.2	Not receiving anything	41
8.3	Transmitter turns on but no one receives my signal	41
8.4	Transmitter says on.....	41
8.5	System crashes when transmitting	41
8.6	Permission problem with USB Audio Adapter GPIO	42
8.7	Baofeng transmitter stays on ¼ second after PTT released	42

1 The Easy Way

If you don't have a suitable interface already, the easiest way is to use a USB audio interface specifically designed for this purpose. It has audio in for receive, audio out for transmit, and PTT control to activate the transmitter.

NEED PICTURE

Some examples are:

- **DINAH** <https://hamprojects.info/dinah/>
- **PAUL** <https://hamprojects.info/paul/>
- **DRA-30** <http://www.masterscommunications.com/products/radio-adapter/dra/dra30.html>
- **RA-35** <http://www.masterscommunications.com/products/radio-adapter/ra35.html>
- **DMK URI** http://www.dmkeng.com/URI_Order_Page.htm
- **RB-USB RIM** <http://www.repeater-builder.com/products/usb-rim-lite.html>

DINAH has a 6 pin mini DIN connector, same as the "data" (external modem) connector found on some Kenwood, Icom, and Yaesu transceivers. A suitable cable is included. This is ideal for 9600 bps operation. Be sure to cut the xxxxxxxxxx jumper so the carrier detect signal does not shut off the transmit audio.

I've only used DINAH, in the list above, so I can't vouch for proper operation of the others.

PAUL has the same type of connector, and pinout, found on most traditional TNCs and trackers so you can use existing or pre made radio-specific TNC cables.

Many of the others were originally intended for ALLSTAR voice over IP and have different connector styles or pinouts. Check the pinouts carefully if using one of the others.

If you are using a Raspberry Pi, also consider the DRAWS. It sits right on top of the Raspberry Pi and has:

- Two radio interfaces.
- Voltage regulator so Raspberry Pi can be run from 12 volts.
- GPS for precise timing or building a tracker.
- Battery backed real time clock.



Visit <http://nwdigitalradio.com/draws/> for more information.

2 Quick Start Guide for USB interfaces with PTT control

This will get you up and running ASAP for the most common case of a single USB audio adapter.

A later section will discuss using multiple audio interfaces so multiple radios can be used.

2.1 Linux

Run the **cm108** utility. You should see something like this:

	VID	PID	Product	Sound	ADEVICE	ADEVICE	HID [ptt]
	---	---	-----	-----	-----	-----	-----
**	0d8c	0012	USB Audio Device	/dev/snd/pcmC2D0c	plughw:2,0	plughw:2,0	/dev/hidraw0
**	0d8c	0012	USB Audio Device	/dev/snd/pcmC2D0p	plughw:2,0	plughw:2,0	/dev/hidraw0
**	0d8c	0012	USB Audio Device	/dev/snd/controlC2			/dev/hidraw0

** = Can use Audio Adapter GPIO for PTT.

Devices marked with “**” can use CM108/119 GPIO pins for PTT control.

Edit the sample “direwolf.conf” configuration file.

- Find the ADEVICE line and add the audio device name found above. Example:

ADEVICE plughw:2,0

- Find the PTT example section, for channel 0, and add this or uncomment an existing line:

PTT CM108

Do not add a specific device path. It will be determined automatically.

2.2 Windows

Run the cm108 utility. You should see something like this:

(need example)

Edit the sample “direwolf.conf” configuration file.

- Find the ADEVICE line and add the audio device name found above. Example:

ADEVICE USB

When direwolf starts up, it will produce a list of the audio devices available. Using the associated number is not recommended because the number can change as new devices are added or removed. It is best to use some unique substring of the name listed.

- b. Find the PTT example section, for channel 0, and add this or uncomment an existing line:

```
PTT CM108
```

Do not add a specific device path. It will be determined automatically if there is only one USB audio adapter. The multiple adapter case is explained in a later section.

2.3 Other operating systems

CM108/CM119 GPIO PTT is not available for other operating systems such as Mac OSX or BSD Unix.

More complicated cases, such as multiple USB audio adapters, are explained later.

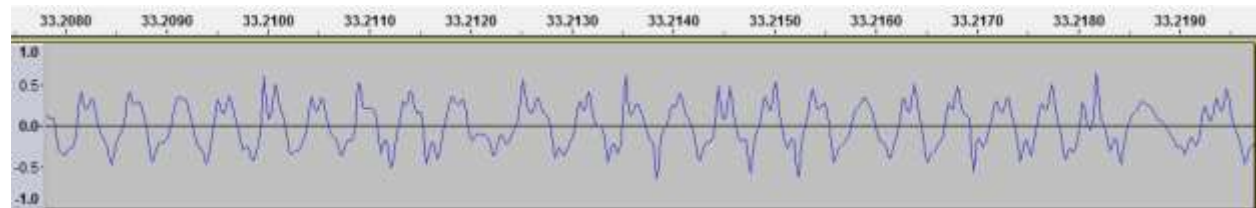
3 Setting proper transmit audio level

The optimal peak FM deviation is somewhere around 3 or 3.5 kHz. Obviously if it is too low, you won't be heard. If too loud, the audio signal will be distorted, making it less likely to be decoded properly.

You might expect it to clip and become more like a square wave but that is not what happens. Instead of being a sine wave, the top and bottom start to get jagged like this:



Here is a more extreme case of over-deviation:



Some interesting examples of really terrible signals can be found in ***A-Closer-Look-at-the-WA8LMF-TNC-Test-CD.pdf***. You don't want to be included in my next collection of really bad signal examples. More importantly you probably want others to understand what you are sending.

Most of us don't have fancy radio test equipment, so how can the FM deviation be set properly?

You should be able to get a reasonably good approximation by ear. Adjust the transmit audio level up and down while listening on a different radio. It will stop getting louder and sound different at the point where it starts to distort. Turn it down a little so you can hear a decrease in volume. If you have an oscilloscope to look at the received audio, that would be more scientific. Notice where it starts to distort and decrease the amplitude to around $\frac{3}{4}$ of that.

After you have configured Dire Wolf for your soundcard and PTT method, you can get a continuous test tone by using "-x" followed by one of these letters on the command line:

- m for the mark tone, typically 1200 Hz.
- s for the space tone, typically 2200 Hz.
- a to alternate between them.

For a multi-radio setup, a radio channel, other than the first, can be specified by putting a number in there too.

4 Computer Audio

Back in the early days of the IBM style personal computer, sound was not a standard feature. It was later added with “sound cards” that were plugged into the motherboard. Now it is a standard feature integrated into the system board. You can use the existing internal sound system or add a dedicated interface for use by the radio. Most often this will be a USB device. The same “sound card” terminology lives on although it is not a “card” anymore.



Starting in the upper left is an old “sound card” that plugged into the motherboard.

Below that is a USB audio adapter that runs in the 5 to 10 dollar range. It has stereo audio output for headphones and mono audio input for a microphone. I added the wires sticking out. This is for PTT control which we will discuss later.

The SignalLink 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 a VOX circuit that turns on the transmitter when transmit audio is present.

Finally, in the lower right we have a Raspberry Pi (RPi) with the UDRC audio board on top. Rather than using USB, it uses the special I²S bus designed for digital audio. The round 6 pin connector matches the “data” connector found on many Kenwood, Icom, and Yaesu transceivers. A newer version of this is called DRAWS. It has two radio connectors, GPS, real time clock, and a voltage regulator so the RPi can be run from 12 volts.

DRAWS, and the earlier UDRC (pictured here) are high performance soundcards that fit on top of the Raspberry Pi. They can use higher audio sampling rates making them ideal for 9600 baud and above.

The 6 pin mini-DIN radio connector matches the “data” connectors on many transceivers, making hook up very easy. For details, see <http://nwdigitalradio.com/draws/> and <https://nw-digital-radio.groups.io/g/udrc/wiki/UDRC%E2%84%A2-and-Direwolf-Packet-Modem#Basic-Configuration>

A simpler audio board (not shown) with an I²C / SPI interface so it doesn't tie up a USB port. This would be especially beneficial with the Pi zero which has only a single USB port that you might want to use for something else. <https://wb7fhc.com/about-the-fe-pi.html>

The Raspberry Pi Foundation even produces one of their own.
<https://www.raspberrypi.org/products/iaudio-codec-zero/>

4.1 Special notes for USB Audio Adapters

These all have two 3.5mm TRS (tip, ring, sleeve) connectors marked headphone and mic. The headphone jack is stereo so you will want to use a stereo (3 conductor) cable. If you use a mono (2 conductor) cable, you would be shorting one of the outputs to ground.

The microphone connector is a little less obvious.

Here is a sample diagram for what you might find inside of one of these USB audio adapters:
http://www.qsl.net/om3cph/sb/CM108_DataSheet_v1.6.pdf See second to last page.

It shows the microphone jack tip capacitively coupled to the CM108 chip with 1 μ F. The ring has a microphone bias voltage.

Here is another that is just the opposite: <http://www.hardwaresecrets.com/datasheets/CM109.pdf>
The tip is a DC bias voltage and the audio input is on the ring.

This <http://www.repeater-builder.com/voip/pdf/cm119-datasheet.pdf> has both on the ring and the tip is not connected.

I guess you just need to try both and see what works. If you are not getting any audio input, or it is extremely weak – only from crosstalk – that might be the problem. The printed circuit board, mentioned a couple pages later, has a jumper so you can easily try either one.

Both the Syba product mentioned above, and a different one from Adafruit, have the tip and ring connected together. They have an open circuit voltage of 4.51 volts which drops down to about half that when connected to ground through a 1.5 k resistor.



The lesson, here, is that you should use a 3.5 mm stereo (TRS) plug, not a mono (TR) plug for the microphone input. If you use a mono cable,

the longer sleeve on the plug might short the microphone input to ground.

Finally, you might want to stick a 1 μF capacitor between the receive audio and the microphone input, due to the DC bias, but I never found it to be necessary.

5 Connection to Radio

The major Japanese Ham Radio manufacturers standardized on the same 6 pin mini DIN connector for use with external modems such as a packet radio TNC. It's usually labeled "data" when it is really audio. I would have called it "external modem" but they didn't ask me for my advice.



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

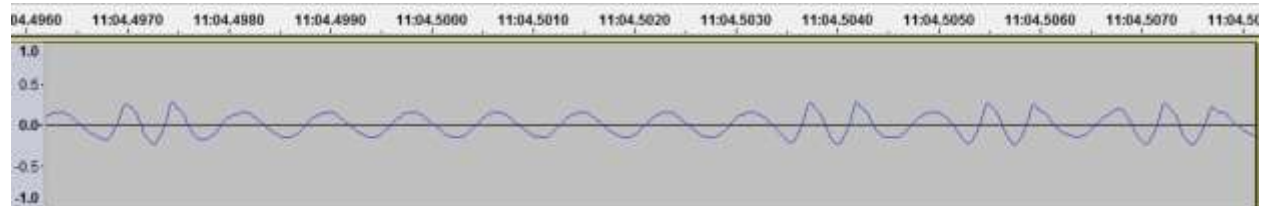
Alinco has the equivalent but it uses a different connector style.

Use this if it is available. The microphone and speaker connections are optimized for voice operation and intentionally introduce distortion which makes it harder for the modem to operate effectively.

VHF/UHF FM transmitters reshape the audio with "pre-emphasis." This boosts the higher frequencies at +6 dB per octave. Suppose your modem was switching between 1200 and 2200 Hz tones of the same amplitude.



The pre-emphasis increases the amplitude of the higher tone by almost a factor of 2.



In the receiver, de-emphasis decreases the amplitude of higher frequencies, again by 6 dB per octave. In theory they should balance out



Frequencies below 300 Hz or so, are eliminated so you don't hear the CTCSS tones (commonly known as PL). This is all fine for voice but makes the modem's job more challenging. Dire Wolf goes to a lot of effort to compensate for this imbalance for superior decoding performance.

More details in ***A-Better-APRS-Packet-Demodulator-Part-1-1200-baud.pdf***

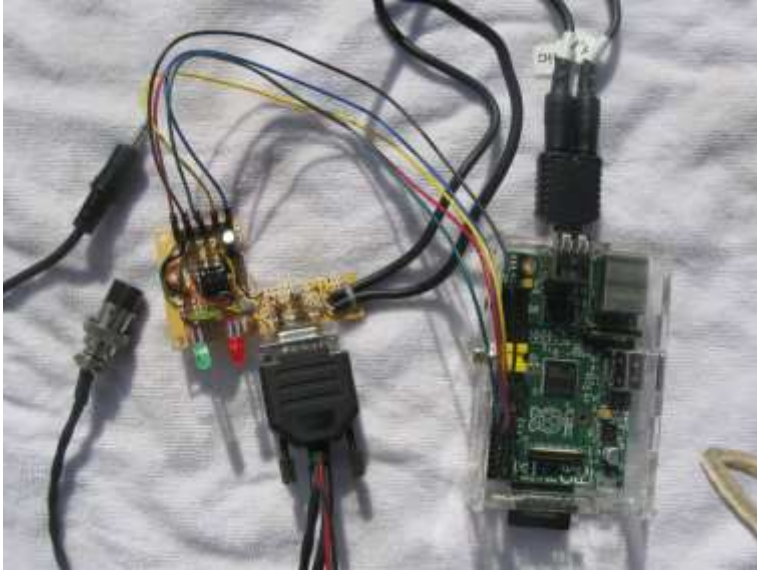
The most common 1200 bps (bits per second) speed is somewhat tolerant of this distortion but for 9600 you absolutely need to use the special connector which bypasses the pre-emphasis and de-emphasis to provide a flat wide audio bandwidth.

Trying to use the microphone and speaker connections, for 9600 bps, will be an exercise in futility.

6 What goes in between?

There is a wide range of options from air to more involved designs with audio isolation transformers and an optoisolator for PTT. Others have covered this topic extensively so I will just provide some examples and links.

This is what I'm using for my digipeater / IGate.



Left: Connections to speaker and Microphone jack of transceiver.

Middle: Interface circuit with a timer to limit transmission time.

It uses the standard 9 pin connector found on most TNCs and trackers so the same radio-specific cables can be used.

The two LEDs are for Data Carrier Detect (DCD) and the Push to Talk (PTT) signals.

The shape of the board was not intentional. It was just a scrap piece of perfboard left over from another project.

Right: Raspberry Pi. A model 1 is more than adequate.

At the top is a USB audio adapter. The current software version can handle 3 audio interfaces at the same time.

Lower left are GPIO connections for PTT and the DCD LED.

I usually take the receive audio and connect it directly to the audio input on the computer.

For transmit, I use a couple resistors, such as 10k and 1k, to

TO BE CONTINUED

7 PTT

Like other many other Ham Radio “soundcard” applications, Dire Wolf provides a wide variety of methods to activate your transmitter. It sounds simple but there are many different options and some require detailed explanation to use them most effectively.

- Serial Port

Back in the previous Century, personal computers had RS-232 serial “com” ports. Using one of the control lines such as RTS (appropriately called Request to Send) was a popular method. Serial ports are now about as scarce as hen’s teeth. You can still use this method by using a USB to serial converter cable. You can also find interfaces which have a USB-to-serial chip used only to generate the PTT signal.

- GPIO or GPIOD

If you have a single board computer, such as a Raspberry Pi, one of the General Purpose Input Output (GPIO) pins can be used.

- GPIO pins inside a USB Audio adapter

The C-Media CM108 and CM119 chips are very popular for USB to Audio adapters. They have GPIO pins that we can use for the PTT signal. This is a very tidy solution because everything goes through a single USB cable.

- Hamlib - CAT (Computer Aided Transceiver)

Most ham transceivers made in the past 30 years or so, have the ability to be controlled from a computer. Originally these used RS-232 serial ports but now USB is the norm. The commands are not standardized so “hamlib” is generally used to deal with all the different possibilities for various brands and models.

- VOX

The transmitter can be activated automatically when transmit audio is present. Use of the VOX, built in to some transceivers, is generally a bad idea because it is designed for voice and keeps the transmitter on much too long after the transmit audio has stopped.

This ***Dire Wolf Radio Interface Guide***, new for release 1.7, gathers scattered information into one place, goes into more detail than the ***User Guide***, and should make finding answers easier.

7.1 Serial Port

Back in the previous Century, personal computers had RS-232 serial or “COM” ports. Using one of the control lines such as RTS (appropriately called Request to Send) was a popular method. Serial ports are now about as scarce as hen’s teeth. One possible method is to use a USB to serial adapter.

If you do anything with RS-232 devices, you should get a cable like this with indicator lights built in. It will save a lot of time and aggravation when troubleshooting. <https://www.gearmo.com/shop/usb2-0-rs-232-serial-adapter-led-indicators/>



Don't connect the RS-232 signal to your transceiver!!! You will need:

- A resistor of about 10k.
- General purpose diode (1N4148, 1N914, etc.) to prevent the base from going very negative.
- General purpose NPN transistor (2N3904, 2N2222, or whatever you have in your junk box).

NEED SCHEMATIC

Some example circuits can be found here:

- <https://www.qsl.net/wm2u/interface.html>
- <http://zs1i.blogspot.com/2010/02/zs1i-soundcard-interface-ii-project.html>

- <https://kb3kai.com/tnc/soft-tnc.pdf>
- [http://www.dunmire.org/projects/DigitalCommCenter/soundmodem/mySoundCardInterface.p
ng](http://www.dunmire.org/projects/DigitalCommCenter/soundmodem/mySoundCardInterface.p
ng)

To use a serial port (either built-in or a USB to RS232 adapter cable) for PTT control, use an option of this form in your configuration file, in the appropriate CHANNEL section:

PTT *device-name* [-]rts-or-dtr [[-]rts-or-dtr]

(The [] indicate something is optional.)

For Windows the device name would be COM1, COM2, etc.

For Linux, the device name would probably be something like

- /dev/ttyS0 or /dev/ttyS1 for a COM port on the PC motherboard, or
- /dev/ttyUSB0 or /dev/ttyUSB1 for a USB-to-RS232 adapter. You would also see this for transceivers, like the IC-7100, that have a USB-to-serial converter built in.

You can also use the Windows format on Linux. COM1 is converted to /dev/ttyS0, COM1 is converted to /dev/ttyS1, and so on. Remember this would apply to a COM port on the motherboard or in a PCI slot. If USB is involved, the names would be different.

Examples:

PTT COM1 RTS

PTT COM1 -DTR

PTT /dev/ttyUSB0 RTS

Normally the higher voltage is used for transmit. Prefix the control line name with “-” to get the opposite polarity. Some interfaces want RTS and DTR to be driven with opposite polarity to minimize chances of transmitting at the wrong time. You can specify two control lines with opposite polarity.

Example:

PTT COM1 -RTS DTR

PTT COM1 RTS -DTR

For the Easy Digi Complete interface, with built-in USB to serial adapter for PTT control, you should drive both control lines with the same polarity. These are both equivalent:

PTT COM1 RTS DTR

PTT COM1 DTR RTS

Alternatively, the RTS and DTR signals from one serial port could control two transmitters. E.g.

CHANNEL 0

PTT COM1 RTS

CHANNEL 1

PTT COM1 DTR

7.2 GPIO

On Linux you can use General Purpose I/O (GPIO) pins if available. This is mostly applicable to a microprocessor board, such as a Raspberry Pi, BeagleBone, Orange Pi, etc., not a general purpose PC.



CAUTION! The general purpose input output (GPIO) pins are connected directly to the CPU chip. There is no buffering or other protection. The interface uses 3.3 volts and will not tolerate 5 volt signals. Static discharge, from careless handling, could destroy your Raspberry Pi.

More information on Raspberry Pi GPIO: <https://www.raspberrypi.org/documentation/usage/gpio/>

There are many GPIO pins. How would you choose an appropriate one? I would try to avoid those with special functions such as UART, SPI, PWM, or I²C. These are my suggestions for the best choices. Note that the physical pin number on the connector, and the GPIO number (used by the software), are not the same.

For the original RPi model 1 (before the “plus” versions):

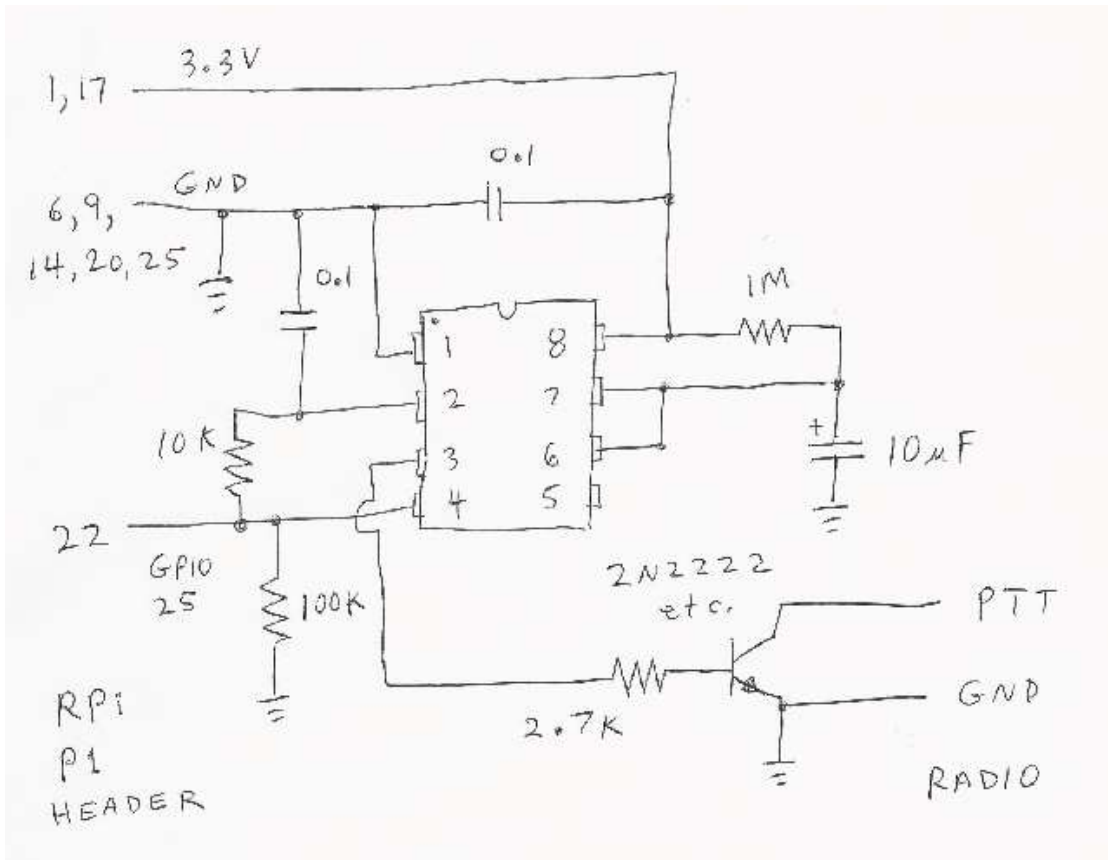
- P1-11 GPIO 17
- P1-15 GPIO 22
- P1-16 GPIO 23
- P1-18 GPIO 24
- P1-22 GPIO 25

The later models (A+, B+, 2, 3, 4) have a larger connector with additional GPIO pins. Any of the pins listed above or these new ones would be suitable.

- P1-29 GPIO 5
- P1-31 GPIO 6
- P1-32 GPIO 12
- P1-33 GPIO 13
- P1-35 GPIO 19
- P1-36 GPIO 16
- P1-37 GPIO 26
- P1-38 GPIO 20
- P1-40 GPIO 21

You could simply use a resistor and transistor to Here is a suggested circuit using a **CMOS** 555 timer (LMC555, TLC555, ICM7555, etc.) to limit transmissions to about 10 seconds in case something goes wrong. **Don't try using the original 555** because it needs a minimum of 4.5 volts and we have only 3.3 here. The time can be increased by making the 10 μ F capacitor larger. It's roughly 1 second for each μ F.

One person reported that the timeout did not work properly and the problem was solved by adding a 0.01 μF capacitor from pin 4 to ground. I suspect this was caused by RF, from the transmitter, getting into the digital circuitry.



You could get by with only a resistor and transistor but a software failure could cause the transmitter to be stuck on, jamming the radio channel, annoying other people, and possibly damaging the transmitter from overheating.

Some transceivers have a built-in transmit timeout setting to achieve the same thing.

7.2.1 Configuration - original (sysfs)

Add a PTT GPIO command to each CHANNEL section. Precede the pin number with “-” to invert the signal.

PTT GPIO [-]pin-number

Example:

PTT GPIO 25

There are more details in the *Raspberry-Pi-APRS.pdf* document.

Behind the scenes, the GPIO pins have device paths such as `/sys/class/gpio/gpio25`. You could set the pin state with a command such as:

```
echo 1 > /sys/class/gpio/gpio25/value
```

You don't need to know that level of detail; I'm just providing a lead in for the next section.

The sysfs-gpio documentation says it is deprecated and will be removed after 2020. Well, that year has come and gone. This interface better not disappear because the new method doesn't work properly on the Raspberry Pi yet when I'm writing this.

7.2.2 Configuration - new character device API (libgpiod)

(Not released yet until it works properly with Raspberry Pi OS.)

Here are some explanations of the difference:

<https://embeddedbits.org/linux-kernel-gpio-user-space-interface/>

<https://www.beyondlogic.org/an-introduction-to-chardev-gpio-and-libgpiod-on-the-raspberry-pi/>

As of January 2021, it still does not work properly on the Raspberry Pi, with the latest updates. The author of the second article is using a version not yet incorporated into the latest Raspberry Pi OS. In April 2021, it does work on the Raspberry Pi. I mention this because you could run into problems with an outdated operating system.

To use this method, simply use "GPIOD" rather than "GPIO."

```
PTT GPIOD [-]pin-number
```

Example:

```
PTT GPIOD 25
```

This is how it is supposed to work.

XXXX

---- config ---- TO BE COINTINUED,,,,

7.3 Hamlib - CAT

Most ham transceivers made in the past 30 years or so, have the ability to be controlled from a computer. Originally these used RS-232 serial ports but now USB is the norm. The commands are not standardized so “hamlib” is generally used so each application developer doesn’t have to figure out all the different possibilities for various brands and models.

Hamlib support is an optional feature. When direwolf starts up, it displays the optional features that were included when it was built. e.g.

```
Includes optional support for:  gpsd hamlib cm108-ptt
```

If you don’t see “hamlib” in there, you will need to rebuild it from source.

If the optional feature is enabled, you can also use this form of the PTT configuration:

```
PTT  RIG  model port [rate]
```

Where,

model identifies the type of radio.

A rig number, not a name, is required here.

For example, if you have a Yaesu FT-847, specify 101.

See <https://github.com/Hamlib/Hamlib/wiki/Supported-Radios> for details.

Get a list of values by running “rigctl --list”.

2 is used to communicate with “rigctl.”

“AUTO” will try to guess what is connected.

port is name of serial port connected to radio.

In the case where model is 2, this would be a host name/address and optional port number. Default port is 4532

rate is an optional serial port rate for CAT control.

Sometimes you might need to override the hamlib default behavior.

Here is an example of where the serial port data rate had to be set explicitly:

<https://groups.io/g/RaspberryPi-4-HamRadio/topic/75478708>

Examples:

- Yeasu FT-817 on /dev/ttyUSB0: PTT RIG 120 /dev/ttyUSB0 9600
- rigctld on localhost: PTT RIG 2 localhost:4532
- Try to guess what is on /dev/ttyS0: PTT RIG AUTO /dev/ttyS0

For more details, see <http://sourceforge.net/p/hamlib/wiki/Hamlib/>

FAQ: <http://sourceforge.net/p/hamlib/wiki/FAQ/>

This would be a good place to go with questions: <http://sourceforge.net/p/hamlib/discussion/>

7.3.1 Hamlib PTT Example: Use RTS line of serial port.

A normal person would not want to use this Rube Goldberg approach because this directly supported by direwolf. This is just a training exercise to show how it works.

First let's try it manually. In one terminal window, start up a daemon with the desired configuration.

```
rigctld -m 1 -p /dev/ttyS0 -P RTS -t 4532
```

“/dev/ttyS0” is the serial port on the mother board.

“-m 1” is for the “dummy” backend, not some particular type of radio.

“-t 4532” is not really necessary because that is the default port.

In another window,

```
echo "\set_ptt 1" | nc localhost 4532
```

```
echo "\set_ptt 0" | nc localhost 4532
echo "\set_ptt 1" | nc localhost 4532
echo "\set_ptt 0" | nc localhost 4532
```

You should observe that the RTS control line changed. If you bought the USB to Serial cable that I suggested, the built-in indicator lights reveal what is going on. Otherwise, hook up a voltmeter. Next...

```
rigctl -m 2 -r localhost:4532
T 1
T 0
T 1
T 0
q
```

“-m 2” means talk to “rigctld.”

“-r localhost:4532” indicates where rigctld is running. You can leave off the “:4532” because that is the default port. You might also see examples with 127.0.0.1 which is equivalent but obscure and confusing to those without any networking background. Actually, it seems you can omit the “-r” option entirely because localhost is the default for rig “model” 2.

Again, we should observe the RTS line of serial port /dev/ttyS0 changing. To use this for PTT, put this in your Dire Wolf configuration file:

```
PTT RIG 2 localhost:4532
```

The “2” is very important. It means communicate with the instance of “rigctld” that we already started up. In this case it is running on the same host but it could be running on a different computer.

7.4 VOX (Voice Operated Transmit)

The transmitter can be activated automatically when transmit audio is present. The **SignalLink USB** uses this technique. Be sure to turn the Delay setting to the minimum position so the transmitter turns off quickly after transmit audio stops.

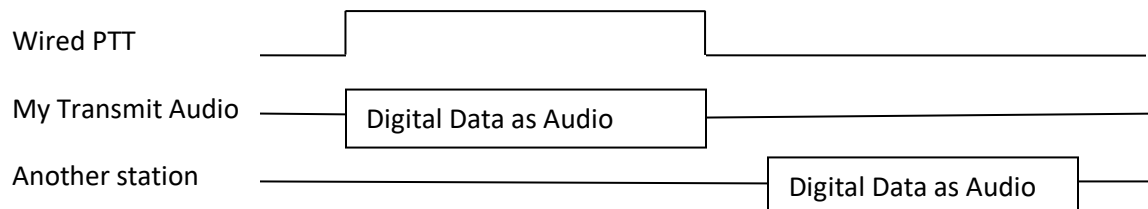
Here are some examples of homebrew circuits:

- <http://wa8lmf.net/ham/tonekeyer.htm#NEW>
- <https://sites.google.com/site/kh6tyinterface/>

It might be tempting to use the VOX built into your transceiver and avoid the extra circuitry for the PTT signal. Is this a good idea?

- For APRS, the short answer is **NO!**
- For connected mode packet, the short answer is **NO!!!**

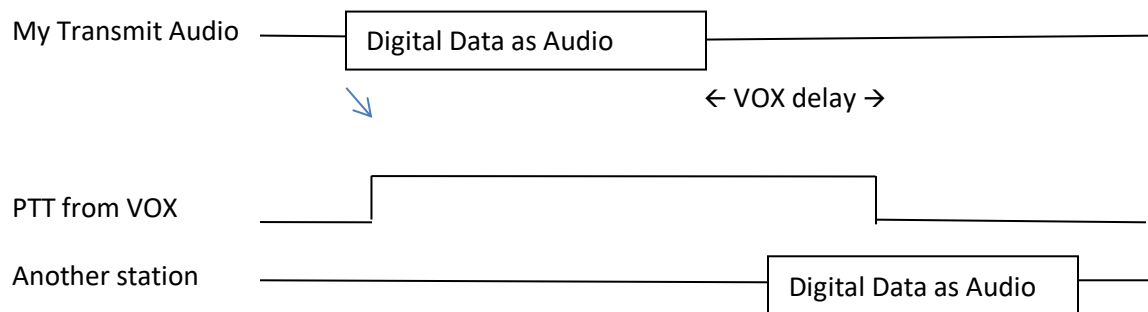
First let's consider the case where we have a wired connection to activate the transmitter. The transmitter is turned on, we send our digital data as an audio signal, and then turn off the transmitter when the audio is finished. Another station detects no other signal, waits a random amount of time (usually less than 1/3 of a second), and starts transmitting.



VOX stands for **Voice** Operated Transmit. It is designed for voice which contains small gaps between words and sentences. It responds quickly when speech begins so it doesn't chop off too much from beginning of the first word. We don't want the transmitter going on and off for every little gap in speech so there is a built in delay before turning the transmitter off again. This is usually referred to as the "VOX delay." On one popular HT, the default is 500 milliseconds and the minimum setting is 250.

Another popular HT doesn't allow the delay time to be configured and the documentation doesn't mention how long it is. It would not be unreasonable to assume they picked something around the default time for another brand.

Keeping the transmitter on a half second after the sound ends is fine for voice. But what about digital data? We saw in the previous section that another station waiting for a clear channel, will usually transmit less than 1/3 second after it no longer hears another digital signal.



Using VOX in this case might be easier but, it is:

- Inconsiderate of the community:

You are interfering with others by sending a quiet carrier, possibly overpowering other stations trying to be heard.

- Bad for APRS:

In this case, you will probably miss the beginning of the next station transmitting because you haven't switched back to receive yet.

- **REALLY BAD** for connected mode packet:

Connected mode uses a rapid back-and-forth exchange to acknowledge that information was received and retry if something gets lost. It is quite likely that the next frame is a response to something you sent. If that response frame is lost, your station will keep trying over again and eventually give up.

My recommendation is to avoid using VOX, built into a transceiver, unless you can be sure the transmitter will turn off very soon (e.g. less than 50 mSec.) after the audio signal is no longer present. If you insist on using VOX, be sure the “VOX delay” setting is at the shortest setting.

The Signalink USB has a built in VOX circuit but it is adjustable into an appropriate range. Turn the delay down to the minimum (fully counterclockwise). According to the documentation, this should turn off the transmitter around 15 or 30 milliseconds after the transmit audio has ended.

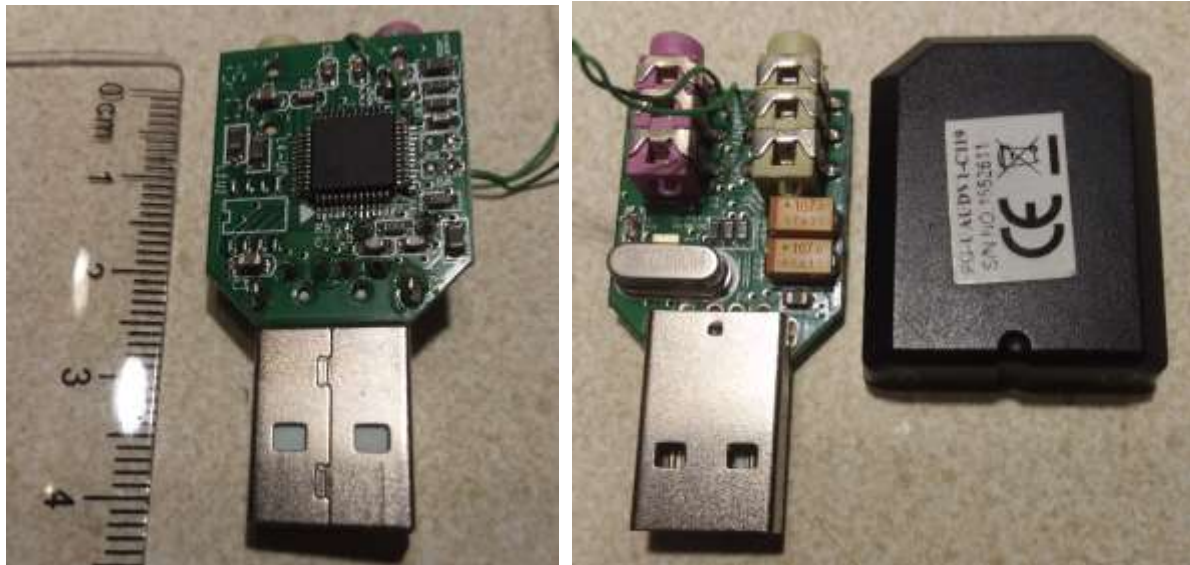
7.5 USB Audio Adapter GPIO - Linux

The C-Media CM108 and CM119 chips are very popular for USB to Audio adapters. They have GPIO pins that we can use for the PTT signal. This is a very tidy solution because everything goes through a single USB cable rather than having separate audio and PTT.

The C-Media CM108, CM119, and similar chips, used in many USB-audio adapters, have varying numbers of general purpose input output (GPIO) pins. These are sometimes connected to push buttons or LEDs.

C-Media chip pin	GPIO number	CM108	CM108AH CM108B	CM109 CM119 CM119A CM119B
43	1	X	X	X
11	2	X		X
13	3	X	X	X
15	4	X	X	X
16	5			X
17	6			X
20	7			X
22	8			X

Here, I pried open a SYBA USB Audio adapter, which happen to contain a CM119, and soldered wires to ground and one of the GPIO pins. This signal would then need to go to a resistor and transistor to pull down the PTT control.



Many products use this technique. Some examples:

- **DINAH** <https://hamprojects.info/dinah/>
- **DMK URI** http://www.dmkeng.com/URI_Order_Page.htm
- **RB-USB RIM** <http://www.repeater-builder.com/products/usb-rim-lite.html>
- **RA-35** <http://www.masterscommunications.com/products/radio-adapter/ra35.html>

There are several similar homebrew projects:

- <http://www.qsl.net/kb9mwr/projects/voip/usbfov-119.pdf>
- <http://rtpdir.weebly.com/uploads/1/6/8/7/1687703/usbfov.pdf>
- <http://www.repeater-builder.com/projects/fob/USB-Fob-Construction.pdf>
- <http://www.egloff.eu/index.php/en/la-technique/interface-audio-usb>

GPIO 3 (pin 13) is used in the homebrew designs because it is easier to tack solder a wire to a pin on the end. All of the commercial products, that I'm aware of, also use the same pin for PTT. It is possible to specify a different pin, in the configuration file, but you probably won't need to.

7.6 USB Audio Adapter GPIO - Linux

The tricky part is that the single physical USB adapter shows up as separate audio and HID (human interaction) devices. Dire Wolf includes a utility application, called **cm108**, to help you make sense of it all.

This is some of what we see for a single USB Audio Adapter:

```
pi@raspberrypi:~ $ cm108
  VID  PID  Product                Sound                ADEVICE              ADEVICE              HID [ptt]
  ---  ---  -----                ---                ---                  ---                  ---
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC1D0c    plughw:1,0          plughw:Device,0     /dev/hidraw3
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC1D0p    plughw:1,0          plughw:Device,0     /dev/hidraw3
** 0d8c 0008 C-Media USB Audio Device /dev/snd/controlC1
413c 2010 Dell USB Keyboard
046d c040 USB-PS/2 Optical Mouse
                                /dev/hidraw1
                                /dev/hidraw2
** = Can use Audio Adapter GPIO for PTT.
```

VID is Vendor ID. 0d8c is C-Media.

PID is Product ID. This is not a good way to distinguish different chips because it is often programmable.

A single USB audio adapter shows up as multiple sound devices:

- pcmC1D0c Card 1, Device 0, “c” for capture (i.e. input)
- pcmC1D0p Card 1, Device 0, “p” for playback (i.e. output)
- controlC1 Card 1, volume control, etc.

The first ADEVICE column has what you would normally use for ADEVICE in the configuration file. The “1” and “0” correspond to the “card” and “device”. “Device” is an unfortunate term because it has too many different meanings. But that is the terminology used by ALSA.

Ignore the second ADEVICE column for now. We will get back to it later.

Finally HID is the corresponding GPIO device in the same chip. There is not a simple relationship between the names. The same USB audio adapter is Sound Card 1 and HID 3.

In this case, your direwolf configuration file would contain:

```
ADEVICE plughw:1,0
CHANNEL 0
PTT cm108
```

When direwolf is started up, important configuration settings are displayed as a help in troubleshooting.

```
Reading config file direwolf.conf
Audio device for both receive and transmit: plughw:1,0 (channel 0)
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, A+, 44100 sample rate / 3.
Using /dev/hidraw3 GPIO 3 for channel 0 PTT control.
```

It automatically figures out which HID corresponds to the soundcard. You could optionally override the automatic mapping but you would probably end up “shooting yourself in the foot” as we will see later. All of the products and homebrew projects, that I have seen, all use GPIO 3 so that is the default if you don’t specify a different GPIO bit number.

Now let’s add two more USB audio adapters.

```
pi@raspberrypi:~$ cm108
VID  PID  Product          Sound          ADEVICE          ADEVICE          HID [ptt]
---  ---  ---
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC1D0c plughw:1,0 plughw:Device,0 /dev/hidraw3
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC1D0p plughw:1,0 plughw:Device,0 /dev/hidraw3
** 0d8c 0008 C-Media USB Audio Device /dev/snd/controlC1 /dev/hidraw3
** 0d8c 000c C-Media USB Headphone Set /dev/snd/pcmC3D0c plughw:3,0 plughw:Set,0 /dev/hidraw5
** 0d8c 000c C-Media USB Headphone Set /dev/snd/pcmC3D0p plughw:3,0 plughw:Set,0 /dev/hidraw5
** 0d8c 000c C-Media USB Headphone Set /dev/snd/controlC3 /dev/hidraw5
1b3f 2008 USB Audio Device /dev/snd/pcmC2D0c plughw:2,0 plughw:Device_1,0 /dev/hidraw4
1b3f 2008 USB Audio Device /dev/snd/pcmC2D0p plughw:2,0 plughw:Device_1,0 /dev/hidraw4
1b3f 2008 USB Audio Device /dev/snd/controlC2 /dev/hidraw4
413c 2010 Dell USB Keyboard /dev/hidraw1
046d c040 USB-PS/2 Optical Mouse /dev/hidraw2

** = Can use Audio Adapter GPIO for PTT.
```

Put this in our configuration file to use 3 radios:

```
ADEVICE plughw:1,0
CHANNEL 0
PTT cm108

ADEVICE1 plughw:3,0
CHANNEL 2
PTT cm108

ADEVICE2 plughw:2,0
CHANNEL 4
PTT cm108
```

Once again, it automatically matches up the soundcard number with the corresponding HID.

```
Reading config file direwolf.conf
Warning: USB audio card 2 (Device_1) is not a device known to work with GPIO PTT.
Audio device for both receive and transmit: plughw:1,0 (channel 0)
Audio device for both receive and transmit: plughw:3,0 (channel 2)
Audio device for both receive and transmit: plughw:2,0 (channel 4)
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, A+, 44100 sample rate / 3.
Channel 2: 1200 baud, AFSK 1200 & 2200 Hz, A+, 44100 sample rate / 3.
Channel 4: 1200 baud, AFSK 1200 & 2200 Hz, A+, 44100 sample rate / 3.
Using /dev/hidraw3 GPIO 3 for channel 0 PTT control.
Using /dev/hidraw5 GPIO 3 for channel 2 PTT control.
Using /dev/hidraw4 GPIO 3 for channel 4 PTT control.
```

Note that the third USB Audio Adapter does not have a C-Media chip so attempts to use the GPIO pins will probably fail. I can’t test it to find out. I pried one of the adapters open and the chip was buried under a blob so there were no pins to probe.

Next, we will unplug this keyboard and mouse then reboot. We don't need them because I'm logging in with ssh over the network.

```
pi@raspberrypi:~ $ cm108
VID  PID  Product                                Sound                                ADEVICE                                ADEVICE                                HID [ptt]
---  ---  -
** 0d8c 0008 C-Media USB Audio Device              /dev/snd/pcmC1D0c                   plughw:1,0                            plughw:Device,0                       /dev/hidraw0
** 0d8c 0008 C-Media USB Audio Device              /dev/snd/pcmC1D0p                   plughw:1,0                            plughw:Device,0                       /dev/hidraw0
** 0d8c 0008 C-Media USB Audio Device              /dev/snd/controlC1                  plughw:2,0                            plughw:Set,0                          /dev/hidraw1
** 0d8c 000c C-Media USB Headphone Set             /dev/snd/pcmC2D0c                   plughw:2,0                            plughw:Set,0                          /dev/hidraw1
** 0d8c 000c C-Media USB Headphone Set             /dev/snd/pcmC2D0p                   plughw:2,0                            plughw:Set,0                          /dev/hidraw1
** 0d8c 000c C-Media USB Headphone Set             /dev/snd/controlC2                  plughw:3,0                            plughw:Device_1,0                     /dev/hidraw2
1b3f 2008 USB Audio Device              /dev/snd/pcmC3D0c                   plughw:3,0                            plughw:Device_1,0                     /dev/hidraw2
1b3f 2008 USB Audio Device              /dev/snd/pcmC3D0p                   plughw:3,0                            plughw:Device_1,0                     /dev/hidraw2
1b3f 2008 USB Audio Device              /dev/snd/controlC3                  plughw:3,0                            plughw:Device_1,0                     /dev/hidraw2

** = Can use Audio Adapter GPIO for PTT.
```

What happened?

The soundcards are listed in the same order, but the numbers have changed.

- The soundcard numbers are now 1, 2, 3 rather than 1, 3, 2.
- The HID numbers are now 0, 1, 2 rather than 3, 5, 4.

That's why you should not specify explicit HID numbers. They change when you add or remove USB devices and reboot. Let direwolf figure out the relationship.

The soundcard numbers have also changed. Suppose you had

- plughw:3,0 connected to the radio for the APRS frequency.
- plughw:2,0 connected to the radio for the Packet BBS frequency.

After rebooting they have been reversed!

Don't despair because there is an easy solution.

I've only been showing you part of the cm108 utility results. Here is the rest of it

```
Notice that each USB Audio adapter is assigned a number and a name. These are not predictable so you could
end up using the wrong adapter after adding or removing other USB devices or after rebooting. You can assign a
name to each USB adapter so you can refer to the same one each time. This can be based on any characteristics
that makes them unique such as product id or serial number. Unfortunately these devices don't have unique serial
numbers so how can we tell them apart? A name can also be assigned based on the physical USB socket.
Create a file like "/etc/udev/rules.d/85-my-usb-audio.rules" with the following contents and then reboot.
```

```
SUBSYSTEM!="sound", GOTO="my_usb_audio_end"
ACTION!="add", GOTO="my_usb_audio_end"
DEVPATH==" /devices/platform/soc/3f980000.usb/usb1/1-1/1-1.3/1-1.3:1.0/sound/card?", ATTR{id}="Fred"
DEVPATH==" /devices/platform/soc/3f980000.usb/usb1/1-1/1-1.4/1-1.4:1.0/sound/card?", ATTR{id}="Wilma"
DEVPATH==" /devices/platform/soc/3f980000.usb/usb1/1-1/1-1.5/1-1.5:1.0/sound/card?", ATTR{id}="Pebbles"
LABEL="my_usb_audio_end"
```

Follow the instructions. Copy/paste the last section into /etc/udev/rules.d/85-my-usb-audio.rules then reboot

We can now turn our attention to the second ADEVICE column.

```

pi@raspberrypi:~ $ cml08
  VID  PID  Product
  ---  ---  ---
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC1D0c plughw:1,0 plughw:Fred,0 /dev/hidraw0
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC1D0p plughw:1,0 plughw:Fred,0 /dev/hidraw0
** 0d8c 0008 C-Media USB Audio Device /dev/snd/controlC1 /dev/hidraw0
** 0d8c 000c C-Media USB Headphone Set /dev/snd/pcmC2D0c plughw:2,0 plughw:Wilma,0 /dev/hidraw1
** 0d8c 000c C-Media USB Headphone Set /dev/snd/pcmC2D0p plughw:2,0 plughw:Wilma,0 /dev/hidraw1
** 0d8c 000c C-Media USB Headphone Set /dev/snd/controlC2 /dev/hidraw1
  1b3f 2008 USB Audio Device /dev/snd/pcmC3D0c plughw:3,0 plughw:Pebbles,0 /dev/hidraw2
  1b3f 2008 USB Audio Device /dev/snd/pcmC3D0p plughw:3,0 plughw:Pebbles,0 /dev/hidraw2
  1b3f 2008 USB Audio Device /dev/snd/controlC3 /dev/hidraw2

** = Can use Audio Adapter GPIO for PTT.

```

I can now put this in my configuration file, using a name rather than a number which can change.

```
ADEVICE plughw:Fred,0
```

The same technique could be used to assign names based on the VID/PID but that would not help with multiple identical adapters.

Names can be assigned based on serial numbers but these adapters don't have serial numbers.

So, we are left with assigning names based on the physical USB socket on the computer.

The keyboard will be plugged in again, and the second USB Audio Adapter (PID 000c) removed. After rebooting again, we see:

```

pi@raspberrypi:~ $ cml08
  VID  PID  Product
  ---  ---  ---
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC1D0c plughw:1,0 plughw:Fred,0 /dev/hidraw0
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC1D0p plughw:1,0 plughw:Fred,0 /dev/hidraw0
** 0d8c 0008 C-Media USB Audio Device /dev/snd/controlC1 /dev/hidraw0
  1b3f 2008 USB Audio Device /dev/snd/pcmC2D0c plughw:2,0 plughw:Pebbles,0 /dev/hidraw3
  1b3f 2008 USB Audio Device /dev/snd/pcmC2D0p plughw:2,0 plughw:Pebbles,0 /dev/hidraw3
  1b3f 2008 USB Audio Device /dev/snd/controlC2 /dev/hidraw3
  413c 2010 Dell USB Keyboard /dev/hidraw2
  046d c040 USB-PS/2 Optical Mouse /dev/hidraw4

** = Can use Audio Adapter GPIO for PTT.

```

Once again, the numbers hav shifted around but the name remains associated with the same USB Audio Adapter. Or more accurately, the USB socket that it is plugged into.

So for our final experiment, swap the two USB Audio Adapters. We don't even need to reboot. We see that the name is associated with the USB socket on the Raspberry Pi, not what is plugged into it.

```

pi@raspberrypi:~ $ cml08
  VID  PID  Product
  ---  ---  ---
  1b3f 2008 USB Audio Device /dev/snd/pcmC1D0c plughw:1,0 plughw:Fred,0 /dev/hidraw0
  1b3f 2008 USB Audio Device /dev/snd/pcmC1D0p plughw:1,0 plughw:Fred,0 /dev/hidraw0
  1b3f 2008 USB Audio Device /dev/snd/controlC1 /dev/hidraw0
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC2D0c plughw:2,0 plughw:Pebbles,0 /dev/hidraw3
** 0d8c 0008 C-Media USB Audio Device /dev/snd/pcmC2D0p plughw:2,0 plughw:Pebbles,0 /dev/hidraw3
** 0d8c 0008 C-Media USB Audio Device /dev/snd/controlC2 /dev/hidraw3
  413c 2010 Dell USB Keyboard /dev/hidraw2
  046d c040 USB-PS/2 Optical Mouse /dev/hidraw4

** = Can use Audio Adapter GPIO for PTT.

```

For more information on assigning persistent names by manufacturer, model, or serial number, see [https://github.com/dh1tw/remoteAudio/wiki/Persistent-USB-Mapping-of-Audio-devices-\(Linux\)](https://github.com/dh1tw/remoteAudio/wiki/Persistent-USB-Mapping-of-Audio-devices-(Linux))

7.7 USB Audio Adapter GPIO - Windows

You might find it helpful to read the preceding Linux section first, to compare how these are different.

All of this magic is performed by the “udev” device manager on Linux. The Windows equivalent is drastically different so an entirely new implementation needs to be created.

At the current time, I know how to get a list of the HID devices and set the GPIO pins but so far have not been able to figure out how to find the relationship between HID and sound devices in the same package.

I would really like to provide the same completeness and convenience as on Linux, but so far I have not been able to figure out how to do it. (Hint, hint, to any Windows programming experts.)

Reluctantly I’m providing a half-baked implementation. It should be fine for those using a single USB Audio Adapter but would be real annoying when trying to use multiple adapters.

First, let’s consider the case of a single USB Audio Adapter.

Your configuration file could contain this:

```
ADEVICE USB
CHANNEL 0
PTT cm108
```

There is only a single USB audio device so it all works well.

Things get more complicated when there are multiple USB Audio Adapters.

```
$ cmd108
VID PID Product HID [ptt]
-----
413c 2010 Dell USB Keyboard \\?\hid\vid_413c&pid_2010&mi_00#9814dafc0b80&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}\kbd
413c 2010 Dell USB Keyboard \\?\hid\vid_413c&pid_2010&mi_01&co101#9814dafc0b80&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
413c 2010 Dell USB Keyboard \\?\hid\vid_413c&pid_2010&mi_01&co102#9814dafc0b80&0001#{4d1e55b2-f16f-11cf-88cb-001111000030}
413c 2010 Dell USB Keyboard \\?\hid\vid_413c&pid_2010&mi_01&co103#9814dafc0b80&0002#{4d1e55b2-f16f-11cf-88cb-001111000030}
0461 4d15 USB Optical Mouse \\?\hid\vid_0461&pid_4d15#8a1d6abc6d0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
** 0d8c 000c C-Media USB Headphone Set \\?\hid\vid_0d8c&pid_000c&mi_03#8&2edc3781&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
** 0d8c 000c C-Media USB Headphone Set \\?\hid\vid_0d8c&pid_000c&mi_03#8&2edc3781&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
1b3f 2008 USB Audio Device \\?\hid\vid_1b3f&pid_2008&mi_03#8&21578911&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
** 0d8c 0008 C-Media USB Audio Device \\?\hid\vid_0d8c&pid_0008&mi_03#8&39d355&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
** = Can use Audio Adapter GPIO for PTT.
```

The good news is that the paths are still there after a reboot.

```
$ cmd108
VID PID Product HID [ptt]
-----
** 0d8c 000c C-Media USB Headphone Set \\?\hid\vid_0d8c&pid_000c&mi_03#8&2edc3781&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
413c 2010 Dell USB Keyboard \\?\hid\vid_413c&pid_2010&mi_00#9814dafc0b80&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}\kbd
** 0d8c 0008 C-Media USB Audio Device \\?\hid\vid_0d8c&pid_0008&mi_03#8&39d355&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
413c 2010 Dell USB Keyboard \\?\hid\vid_413c&pid_2010&mi_01&co101#9814dafc0b80&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
1b3f 2008 USB Audio Device \\?\hid\vid_1b3f&pid_2008&mi_03#8&21578911&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
413c 2010 Dell USB Keyboard \\?\hid\vid_413c&pid_2010&mi_01&co102#9814dafc0b80&0001#{4d1e55b2-f16f-11cf-88cb-001111000030}
413c 2010 Dell USB Keyboard \\?\hid\vid_413c&pid_2010&mi_01&co103#9814dafc0b80&0002#{4d1e55b2-f16f-11cf-88cb-001111000030}
0461 4d15 USB Optical Mouse \\?\hid\vid_0461&pid_4d15#8a1d6abc6d0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
** 0d8c 000c C-Media USB Headphone Set \\?\hid\vid_0d8c&pid_000c&mi_03#8&2edc3781&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
** = Can use Audio Adapter GPIO for PTT.
```

Explicitly specify the desired PTT HID like this:

```
PTT cm108 "\\?\hid#vid_0d8c&pid_000c&mi_03#8&2edc3781&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}"
```

8 Troubleshooting

8.1 General

The first thing you want to do is look at what direwolf says when it starts up.

The optional features included are listed near the beginning. Example:

```
Includes optional support for:  gpsd hamlib cm108-ptt
```

If you want to use hamlib, and it is not listed there, you will need to install hamlib then rebuild from source.

If you want to use a USB audio adapter, with built in PTT control, the “cm108-ptt” optional feature must appear here.

Next we have information about how the modems and PTT are configured. Linux example:

Windows example:

```
Available audio input devices for receive (*=selected):
  0: Microphone (2- C-Media USB Head
Available audio output devices for transmit (*=selected):
  0: Speakers (Realtek High Definiti
  1: DELL U2410-1 (NVIDIA High Defini
  2: Speakers (2- C-Media USB Headph
  3: Realtek Digital Output (Realtek
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, A+, 44100 sample rate.
Note: PTT not configured for channel 0. (Ignore this if using VOX.)
```

In this case, “ADEVICE” was not specified, in the configuration file, so there is no “*” indicating the selected device. The first one, in each group is taken as the default. What you probably want is:

ADEVICE USB

1200 bps is the xxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxx

If not what you were expecting, review the configuration file settings. Check out:

- The wiki <https://github.com/wb2osz/direwolf/wiki>

- Discussion forum <https://groups.io/g/direwolf>

You might find your answer there. If not, ask the group for help. Your question, and any answers will help others in the future. Be sure to include a good description any relevant details.

Do not use the github issues section to ask for help. It is for reporting software defects and making enhancement requests.

8.2 Not receiving anything

Check to see if received audio is getting in at an appropriate level.

Elaborate. Not muted.

8.3 Transmitter turns on but no one receives my signal

Listen with another receiver. Does your signal sound like the others? Adjust the transmit audio level to be similar to others.

8.4 Transmitter says on

This is usually caused by RF, from the transmitter, getting back into the PTT circuit and it latches on.

Move the radio, and especially the antenna, farther away from the computer.

Put ferrite chokes on the cable to stop the RF from getting back to the computer.

Some people like to use audio transformers to isolate the radio from the computer. An optoisolator is used for PTT to break up the ground loop.

8.5 System crashes when transmitting

This is a more severe case of the same thing.

8.6 Permission problem with USB Audio Adapter GPIO

Normally, all of `/dev/hidraw*` are accessible only by root.

```
$ ls -l /dev/hidraw*
crw----- 1 root root 247, 0 Sep 24 09:40 /dev/hidraw0
crw----- 1 root root 247, 1 Sep 24 09:40 /dev/hidraw1
crw----- 1 root root 247, 2 Sep 24 09:40 /dev/hidraw2
crw----- 1 root root 247, 3 Sep 24 09:50 /dev/hidraw3
```

Unnecessarily running applications as root is generally a bad idea because it makes it too easy to accidentally trash your system. We need to relax the restrictions so ordinary users can use these devices.

If all went well with installation, the `/etc/udev/rules.d` directory should contain a file called **99-direwolf-cmedia.rules** containing:

```
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="0d8c", GROUP="audio", MODE="0660"
```

I used the “audio” group, mimicking the permissions on the sound side of the device.

```
$ ls -l /dev/snd/pcm*
crw-rw----+ 1 root audio 116, 16 Sep 24 09:40 /dev/snd/pcmC0D0p
crw-rw----+ 1 root audio 116, 17 Sep 24 09:40 /dev/snd/pcmC0D1p
crw-rw----+ 1 root audio 116, 56 Sep 24 09:50 /dev/snd/pcmC1D0c
crw-rw----+ 1 root audio 116, 48 Sep 29 18:14 /dev/snd/pcmC1D0p
```

Notes:

- The double equal `==` is a test for matching.
- The single equal `=` is an assignment of a value. In my limited experience, Debian type systems can accept either `=` or `:=` but Red Hat type systems recognize only the `=` form.

If you don’t have `/etc/udev/rules.d/99-direwolf-cmedia.rules`, create it as shown above and reboot.

Now we observe that the `/dev/hidraw*`, corresponding to the USB-Audio device now has permissions that allow us to access it.

```
$ ls -l /dev/hidraw*
crw-rw---- 1 root audio 247, 0 Oct 6 19:24 /dev/hidraw0
crw----- 1 root root 247, 1 Oct 6 19:24 /dev/hidraw1
crw----- 1 root root 247, 2 Oct 6 19:24 /dev/hidraw2
crw----- 1 root root 247, 3 Oct 6 19:24 /dev/hidraw3
```



8.7 Baofeng transmitter stays on ¼ second after PTT released

Disable the STE setting. Menu item 35????

