Inhaltsverzeichnis

Datei:Funkpaketpost.png

- Datei
- Dateiversionen
- Dateiverwendung
- Metadaten

Es ist keine höhere Auflösung vorhanden.

Funkpaketpost.png (529 × 205 Pixel, Dateigröße: 41 KB, MIME-Typ: image/png)

Titelbild zur Linux Funkpaketpost

Dateiversionen

Klicken Sie auf einen Zeitpunkt, um diese Version zu laden.

	Version vom	Vorschaubild	Maße	Benutzer	Kommentar
aktuell	15:37, 11. Mär. 2022		529 × 205 (41 KB)	Detlesa (Diskussion | Beiträge)	

Sie können diese Datei nicht überschreiben.

Dateiverwendung

Die folgenden 2 Seiten verwenden diese Datei:

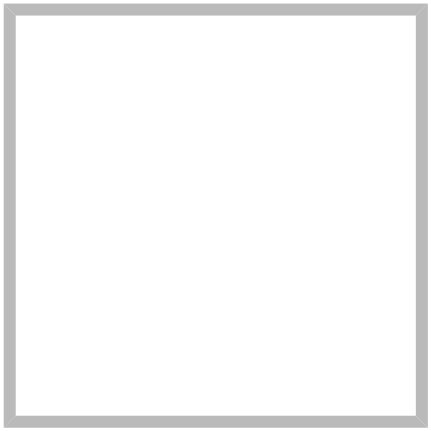
- Linux und Amateur Packet Radio
- Linux und Schmalband Packet Radio mit Terminal

Metadaten

Diese Datei enthält weitere Informationen, die in der Regel von der Digitalkamera oder dem verwendeten Scanner stammen. Durch nachträgliche Bearbeitung der Originaldatei können einige Details verändert worden sein.

Horizontale Auflösung	37,8 dpc
Vertikale Auflösung	37,8 dpc
Speicherzeitpunkt	13:36, 11. Mär. 2022

Roland, OE1RSA



Name	Roland, OE1RSA

Vorlage:User

Roland, OE1RSA

Name	Roland, OE1RSA

DXL - APRSmap Download

Diskussion:HAMNET-70

Linux und Schmalband Packet Radio mit Terminal

Linux und Amateur Packet Radio

Hauptseite

Packet Radio via HAMNET

Linux und Amateur Packet Radio

Inhaltsverzeichnis

Einleitung

Amaterurfunk, Packet Radio und Linux verbindet eine gemeinsame Geschichte die bis in die Anfänge der beiden letzten zurückreicht. So gibt es seit der Version 2.0 des Linux Betriebssystemkerns einen Netzwerktreiber der tief im System verankert ist: den ***AX.25*** Protokoll Treiber, wobei das ***A*** hier tatsächlich für Amateur und nicht etwa Audio steht.

Zu dieser Zeit, in den frühen 90ern des letzten Jahrhunderts gehörte es noch zum guten Ton, dass man ein Amateurfunkrufzeichen hatte wenn man Technik affin war. Und so sollte es nicht allzu sehr verwundern, dass Kernel Entwickler und Funkamateur *Alan Cox* mit Rufzeichen *G4PTS*, wie man zum Beispiel aus der *man-Page* der Software *axcall* erfahren kann, unter Anderem die Quellen für den AX25 Kernel-Treiber beigesteuert hat.

Packet Radio selbst erblickte ein gutes Jahrzehnt vor Linux, rund um das Jahr 1978 (1) das Licht der Welt. Zu dieser Zeit kamen langsam *Personal Computer* in Umlauf die zunächst häufig mit dem Betriebsystem *DOS* ausgestattet waren. *DOS* hatte damals noch die Bedeutung von **D**isk **O**perating **S**ystem und stand noch nicht für die gefürchtete Netzwerkattacke *Denial of Service*. Es war die Zeit der aufkommenden Telefonmodems und der Begriff des Computer Netzwerkes hatte noch nicht Eingang in die Welt der Personal Computer gefunden.

Es war aber auch die Zeit in der die Menschen Software als etwas Eigenständiges zu begreifen begannen. Dadurch, dass Software eine Existenz unabhängig von einem physischen Träger hat weil sie leicht zu kopieren ist fürchteten die Einen um die Früchte ihrer Arbeit gebracht zu werden während Andere in der Kopierbarkeit eben gerade einen großen Vorteil sahen, weil man Software dadurch beliebig oft teilen konnte und trotzdem Jeder die ganze Software behielt.

Es gab aber auch Mischformen: So erkannten findige Leute, dass man sich die leichte Kopierbarkeit zunutze machen konnte um Vertriebskosten einzusparen, wenn auch zu dem Preis, dass Nutzer der Software diese eben genauso leicht weitergeben konnten. Indem man aber nur die maschinlesbaren Bitmuster herausgab konnte man wenigstens andere ProgrammiererInnen daran hindern dass sie die Programme verändern konnten. Die Befürchtung war nämlich, dass auf diese Weise die Arbeit die man in die Entwicklung gesteckt hatte von Anderen als die Eigene ausgegeben werden konnte und man so um die Früchte der eigenen Arbeit gebracht wäre. Weil man die Kopierbarkeit durch die Anwender ohnehin nicht effektiv unterbinden konnte, so ermutigte man die Anwender sogar solche Software weiterzugeben und auf diese Weise Werbungskosten und Vertriebskosten zu sparen. Diese Art Software nannte man "Share Ware" und deren Erzeuger hofften durch das Angebot von Premium Versionen mit essentiellen Zusatzfunktionen schließlich Gewinn zu machen. Die Verfechter der freien Variante waren übrigens nicht durchgehend Altruisten die nur der Welt etwas schenken wollten oder Hobbyisten die nur zu ihrem Vergnügen programmieren, sie hatten ganz einfach ein anderes *Geschäftsmodell* , nämlich eines das auf der Idee basierte, dass man sein Einkommen nicht dadurch erzielt, dass man einfach für die Vervielfältigung von Bitmustern entlohnt wird sondern durch Support- und Beratungsleistung der Anwender.

Warum ist diese Thematik in einer Einleitung zum Thema *Linux und Amateur Packet Radio* so wichtig? Vielen Nutzern der für Packet Radio verfügbaren Software war es nämlich offenbar genug, dass sie die Software, wie man so sagt: *gratis* bekommen konnten. Deshalb finden wir

auch heute wenn wir nach Software für Packet Radio suchen zwar noch eine Menge Programme die man sich aus dem Internet laden kann, aber leider sind für die meisten dieser Programme keine Quellcodes mehr verfügbar. Das führt nun dazu dass niemand sie an die durch die Weiterentwicklung von Hardware und Betriebssystemen veränderten Umstände anpassen kann, außer der Originalautor macht das, sofern er nicht das Interesse verloren hat oder er es nicht mehr kann weil er bereits ein SK ist.

Nun wird man zwar auch im Umfeld von *Linux* den ***Gratis-Software*** Typus finden, er übt sich aber hier in starker Zurückhaltung. In Linux ist der Typus **Freie-Software** deutlich stärker vertreten. Von freier Software spricht man wenn man die Freiheit hat

- die Software auszuführen, wie man möchste, für jeden Zweck,
- die Funktionsweise der Software zu untersuchen und eigenen Bedürfnissen anzupassen,
- die Software weiterzuverbreiten und damit seinen Mitmenschen zu helfen sowie
- die Software zu verbessern und diese Verbesserungen zu veröffentlichen.

Es handelt sich dabei um die von der Free Software Foundation definierten Kriterien an der man freie Software erkennt.

In dieser Artikelserie soll nun versucht werden zu zeigen wie die verschiedenen Aufgaben von *Packet Radio* ausschließlich mit freier Software realisiert werden können. Auch in diesem Zusammenhang wird man auf *veraltete* Software stoßen und auf Probleme für die (noch) keine Komponenten vorhanden sind. Aber anders als bei *Share Ware* kann man freie Software zumindest im Prinzip immer an aktuelle Verhältnisse anpassen und da Packet Radio im Linux Umfeld aus vielen kleinen Teilprogrammen zusammengesetzt wird braucht man für fehlende Funktionen nur jeweils vergleichsweise kleine Zusatzprogramme schreiben. Die Möglichkeit teil-veraltete Software zu aktualisieren wurde habe ich beispielsweise bereits selbst wahrgenommen indem ich das Software **soundmodem** (2) von Thomas Sailer aktualisiert habe.

Das Betriebssystem GNU-Linux

Hier ist, hoffentlich verständlicherweise, nicht der Ort die Grundlagen des Umgangs mit Linux zu erklären. Das Internet ist aber voll mit Informationen. Wer nur mal *schnell* probieren will ob das überhaupt etwas für ihn/sie ist dem sei die *Knoppix* CD (3) ans Herz gelegt.

Es folgen nun ein paar allgemeine Ratschläge ohne in die Tiefe zu gehen.

Da Gnu-Linux nicht nach dem Prinzip: *one size fits all* funktioniert hat man immer wieder einmal die *Qual der Wahl*. Die Software Pakete von Gnu-Linux bekommt man üblicherweise als **Distribution**. Es ist für viele Neueinsteiger aber verblüffend, dass Gnu-Linux nicht *das eine Ding* ist sondern als Sammlung mit verschiedenen Schwerpunktsetzungen verfügbar ist. Trotzdem gilt häufig, dass Informationen die für eine Distribution gelten, mit Vorsicht angewandt, auch in einer anderen Distribution nützlich sein können.

Distributionen sind aber nicht der einzige Weg an Software zu kommen, in letzter Zeit kommen immer mehr Pakete in Umlauf die unabhängig von der Distribution installiert werden können. Wie auch bei anderen Betriebssystemen ist man dabei gut beraten möglichst sorgfältig die Quelle aus der man die Software installiert zu prüfen um böse Überraschungen zu vermeiden.

Dann gibt es natürlich noch den Weg den man bei *freier Software* immer beschreiten können sollte: Die Installation aus den Paketquellen.

Ich setzte in der Folge für meine *Anleitungen* die '*Debian* (4) Distribution voraus. Das ist eine der am weitesten verbreiteten Distributionen die auch die Basis für andere Distribution wie zum Beispiel *Ubuntu* ist. Trotzdem soll diese Einschränkung nicht als Wertung missverstanden werden. Ich fordere Euch, die Amateurfunk Gemeinde, auf diese Artikel um Hinweise zu ergänzen wie ein bestimmtes Problem in einer anderen Distribution zu lösen wäre, sollte es Unterschiede geben. Das hier ist ja schließlich ein *Wiki* in dem wir unser Wissen zusammentragen.

Als **Hardware** kann es für den Anfang ein nicht mehr ganz taufrischer PC durchaus tun. Ihr seid möglicherweise überrascht wenn ihr seht was in dem alten Ding noch drin steckt wenn ihr eine leichtfüßige Distribution wie zum Beispiel Lubuntu (5) installiert. Diejenigen, die einen Raspberry haben können entweder das originale Image installieren oder auch ein in jüngerer Zeit verfügbar gewordenes Debian Image mit dem dann *ganz normale* Debian Pakte aus dem Hauptrepository nachinstalliert werden können.

Wie bereits gesagt ist dies ein zwar interessantes aber viel zu weites Feld weshalb wir uns nun auf unser Hauptthema, '*Linux und Packet Radio* konzentrieren wollen.

Möglichst schnell online gehen

Wer auf dieser Seite gelandet ist will wahrscheinlich möglichst schnell mit Linux ins Packet Radio Netz kommen um ein erstes Erfolgsgefühl zu erleben. Nun, je nach Voraussetzungen mag das schwerer oder leichter sein. Wer zum Beispiel von Windows kommt und von seinem Equipment bereits weis, dass es geeignet ist, sollte auf keine großen Hürden stoßen. Anders mag das sein wenn noch keine Vorerfahrungen vorhanden sind.

In diesem Abschnitt soll deshalb versucht werden das Ziel möglichst schnell anzusteuern, wenn auch auf Kosten des tieferen Verständnisses. Ich habe vor das später nachholen.

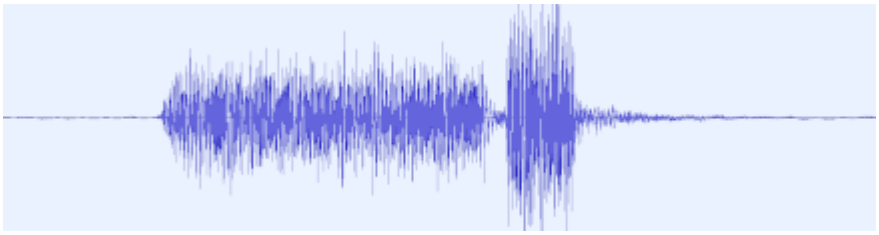
Die erste Frage die wir beantworten müssen ist: Wo ist der nächste *Packet Radio Knoten*'? Für Österreich versucht das UKW Referat eine aktuelle Liste der Digipeater (7) bereitzustellen. Aus dieser Liste ist leider nicht herauszulesen ob es sich beim angegebenen Digipeater um einen **APRS Digipeater** oder um einen **Packet Node** handelt. In diesem Wiki finden sich auf der allgemeinen Packet Radio Seite dazu Hinweise. Es ist jedenfalls eine gute Idee, sobald man eine potentielle Frequenz ermittelt hat seinen Receiver abzuhören und die Empfangslage festzustellen. Digipeater senden Baken in der Betriebsart aus in der sie auch gearbeitet werden können. Die folgenden Hörbeispiele verdeutlichen was zu erwarten ist:

1. https://wiki.oevsv.at/wiki/Datei:afsk12.mp3
   1200 Baud Audio Frequenzmodulation (AFSK)

2. https://wiki.oevsv.at/wiki/Datei:fsk48.mp3
   4800 Baud Frequenzmodulation (FSK) nach FM Demodulator

3. https://wiki.oevsv.at/wiki/Datei:fsk96.mp3
   9600 Baud frequenzmodulation (FSK) nach FM Demodulator

Es lohnt sich darauf hinzuweisen, dass diese Hörbeispiele mit einem Mikrofon vom Lautsprecher des Receivers aufgezeichnet wurden. In der folgenden Abbildung ist deutlich ein kurzer Rauschpegel zu erkennen bevor der *Squelch* den Empfang stummschaltet:

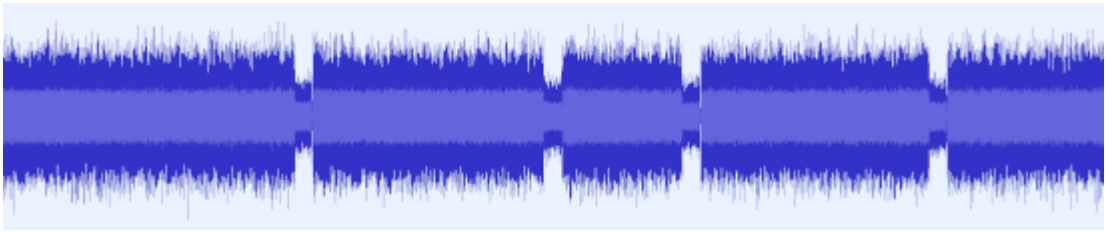


Auf eine weitere Besonderheit sollte man deshalb achten: Hört man den Empfang mit offener Rauschsperre ab so kann man das eigentliche Signal sehr leicht überhören wie das folgende Hörbeispiel zeigt:

- https://wiki.oevsv.at/wiki/Datei:4xfsk96_open_sq.mp3

9600 Baud FSK mit offener Rauschsperre

In der folgenden Abbildung ist zu erkennen was hier passiert:



Da es sich um FM Empfang handelt sehen wir bei offener Rauschsperre einen hohen Rauschpegel. Erst bei Vorhandensein des Trägersignales sehen wir den Signalpegel durch ein Absinken der Amplitude. Wie wir in einem vorhergehenden Beispiel gehört haben empfinden wir das *9600Bd* Signal als Rauschen. Da unser Ohr den Unterschied zwischen im Signalgehalt aber nicht wahrnehmen kann hören wir nur das Absinken der Amplitude. Ein verlässlicher Indikator ist in diesem Fall also nur der Blick auf das S-Meter unseres Transceivers.

Der nächste Schritt besteht nun darin unser Funkgerät fit zu machen. Dabei gibt es verschiedene Wege die man beschreiten kann.

1. Arbeiten mit Hardware TNC und Modem oder
2. arbeiten mit Soundkarte und Software TNC oder
3. arbeiten mit SDR und Transceiver ...

Damit wir dem Titel dieses Abschnittes gerecht werden beschränken wir uns hier zunächst einmal auf ein Minimum, das so lautet:

1. Anschluss unserer Soundkarte an das Funkgerät und
2. Empfang und Dekodierung von APRS Nachrichten mit 1200Bd.

Das ist zugegebenermaßen ein bescheiden klingendes Ziel, wer aber die Schritte sorgfältig durchführt wird damit Basiskenntnisse erlangen, die sich später als nützlich erweisen werden.

Das Verbindungskabel zwischen Transceiver und PC

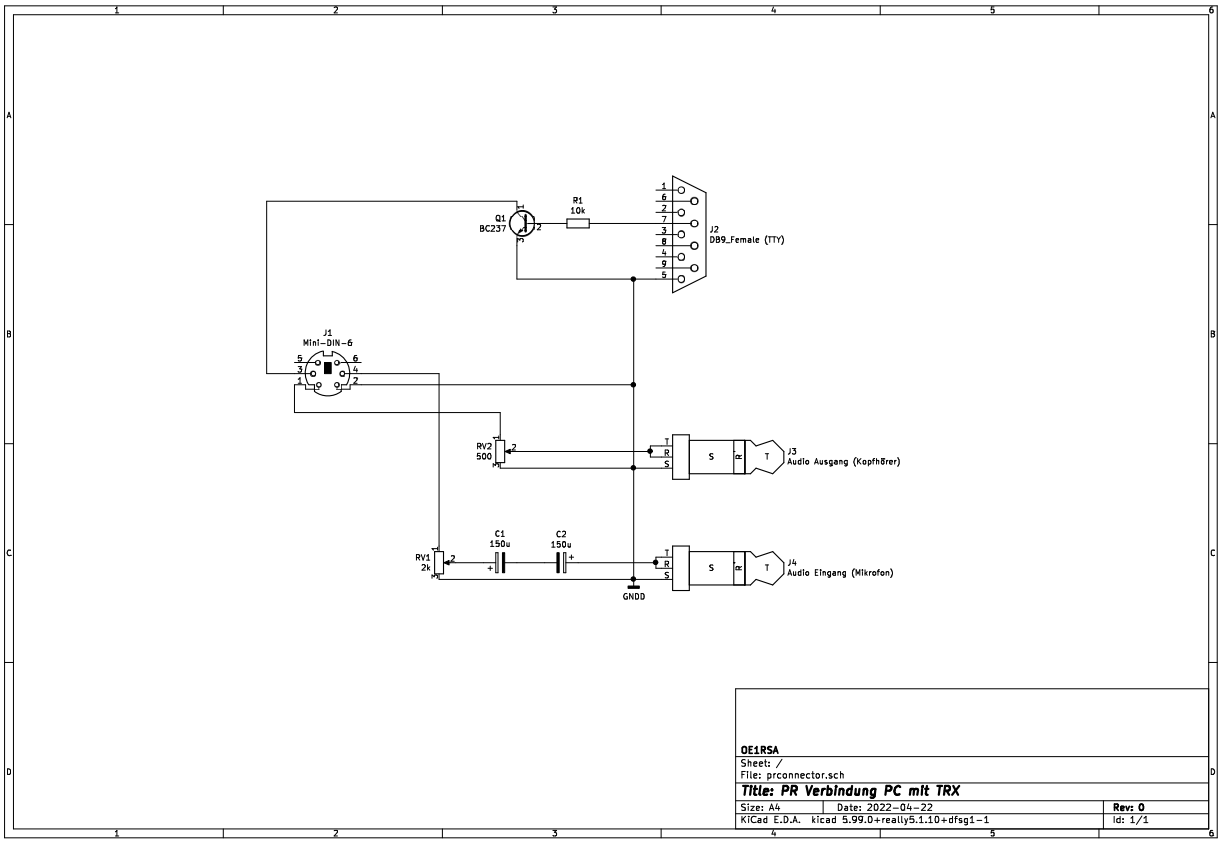
Es beginnt mit dem Anschluss der Soundkarte. Wer bereits Erfahrungen mit dem auf Kurzwelle gerne verwendeten Programm **fldigi** gesammelt hat hat hier sicher einen Vorteil. **ALSA**, die *A* dvanced *L*inux *S*ound *A*rchitecture, das System von Treibern und Hilfsprogrammen ist zwar sehr mächtig, mitunter aber etwas sperrig in der Handhabung. Dazu kommt, dass auf modernen Linuxdistributionen ein Soundmanager wie **pulseaudio** *sitzt*, der üblicherweise meint er habe die alleinige Autorität über die Hardware.

Soferne wir geschafft haben ein digitales - APRS - Signal aufzufangen, z.B. auf der Frequenz 144,800 MHz, stehen wir nun vor der Aufgabe das Signal in unseren Rechner zu bekommen. Obwohl es bei 1200Bd durchaus möglich ist, über das Mikrofon einen Erfolg zu erzielen, so empfehle ich das nicht betriebsmäßig zu machen, weil natürlich die Robustheit bei dieser Methode leidet, und wir außerdem auf diese Weise höhere Geschwindigkeit unter Garantie nicht erreichen werden.

Ein häufig beschrittener Weg ist es, ein für den Transceiver geeignetes Soundkarten Interface zu erwerben. Doch Achtung: Normalerweise sind diese Interfaces nicht für den 9600 Baud Betrieb geeignet. Wer aber mit den klassischen 1200Baud das Auslangen findet und zudem ein bequemes Interface sucht das auch für den Betrieb auf Kurzwelle geeignet ist, ist hier sicher gut beraten. Bevor man sich eine solche Neuanschaffung überlegt lohnt ein genauer Blick ins Handbuch des Transceivers. Modernere Rigs haben oft bereits eine über USB ansprechbare Soundkarte eingebaut, sodass hier nur ein Kabel nötig ist. Ich muss an dieser Stelle leider anmerken, dass, aus mir unverständlichen Gründen, diese Soundkarten häufig intern so verdrahtet sind, dass sie nicht für die *höheren* Geschwindigkeiten wie 9600Bps geeignet sind.

Wenn man die genannten Wege nicht beschreiten kann, oder möchte, so wäre das Nächste ein Blick auf die Rückseite des Transceivers (oder ins Handbuch natürlich). Findet sich dort eine **9-polige Mini DIN** Buchse, oder bietet der Hersteller ein Adapterkabel an, das in einer solchen Buchse endet, so stehen die Chancen meiner Erfahrung nach äußerst gut, dass man darüber nicht nur Betrieb mit 1200Bps sonder auch mit 9600Bps machen kann. Nötig ist dafür nur ein Adapterkabel, das auf der anderen Seite des 9-poligen Steckers zwei Audio Klinkenstecker sowie eventuell einen Stecker für die serielle Schnittstelle aufweist. Dieser serielle Stecker ist aber kein Muss. Er wird nur benötigt wenn wir die PTT über die 9-pol. Mini DIN steuern wollen. Es gibt aber auch Alternativen dazu die über die **hamlib** führen. Dazu später mehr, bzw. der Verweis auf die Dokumentation zu **direwolf**. Bei der Realisierung des Kabels ist **DIY** gefragt und da es durchaus sein kann, dass eure Realiserung von meinem Beispiel abweicht, lade ich euch ein eure Lösung auch im Wiki zu dokumentieren damit es andere leichter haben. Ich beschreibe deshalb nur wie

ich es gemacht habe. Mein Transceiver, ein ICOM IC-9100, hat zwar eine eingebaute USB Audio Karte, die kann man aber leider eben nur für die langsamere 1200bps AFSK Übertragung verwenden. Ich habe deshalb ein Kabel für die *DATA* Buchse zusammengelötet über das man auch die schnellere Betriebsart, mit 9600bps bezeichnet, arbeiten kann. Die Beschaltung meines Kabels sieht so aus:

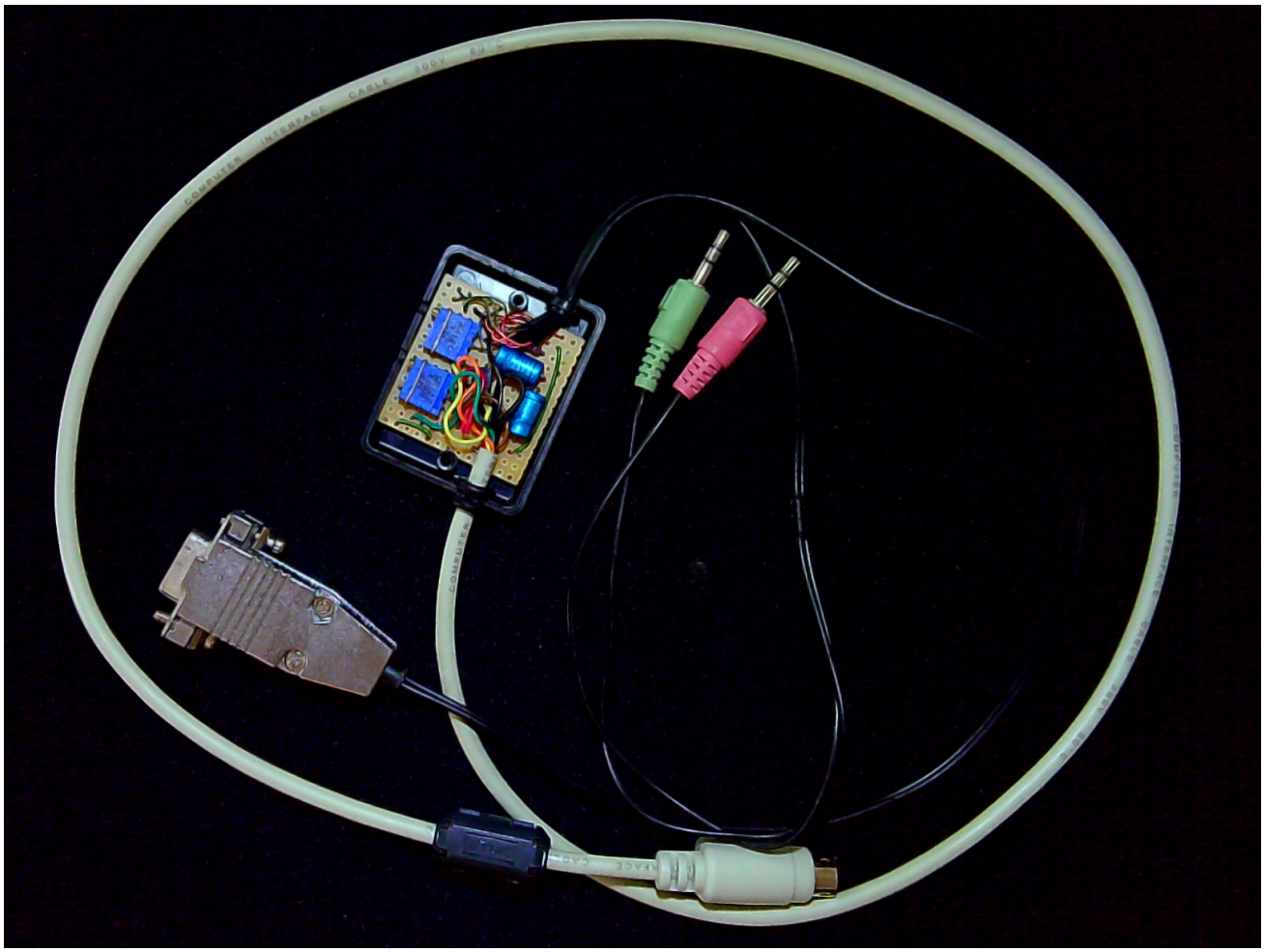


Dazu ein paar Anmerkungen. Wenn jemand meint die meisten der Bauteile sind überflüssig, so hat sie nicht ganz unrecht. Die beiden Potentiometer zur Pegelanpassung sind möglicherweise auf der Soundkarte vorhanden und können dann sogar per Software bedient werden. Auch die beiden Elektrolyt Kondensatoren sind nur notwendig wenn sich herausstellt, dass nicht bereits in der Soundkarte oder im Transceiver so eine Entkopplung durchgeführt wird. Eigentlich ist es ja sogar verblüffend, dass wir an dieser Stelle überhaupt eine Abtrennung der Gleichspannung einbauen, da wir im Prinzip auch sehr niedrige Frequenzen übertragen wollen. Wir dürfen nicht vergessen, dass es sich bei unserer Schnittstelle letztlich um ein System zur Audio Übertragung handelt und da unser Ohr für sehr tiefe Frequenzen unempfindlich ist werden diese Frequenzen eben nicht mehr zuverlässlich übertragen. Ja mehr noch die Gleichspannung wird oft für die Energieversorgung eines externen Mikrofons verwendet oder um Schaltvorgänge auszulösen. Die Kondensatoren sollen uns davor schützen sind aber eben im Einzelfall möglicherweise entbehrlich.

Manche USB-Dongle Soundkarten haben IO-Pins die für die Ansteuerung von LEDs geeignet sind. In diesem Fall ist es möglich diese IO Pins für die Ansteuerung der **PTT** zu verwenden. Ich habe das aus Bequemlichkeit nicht gemacht, sondern habe die **RTS** Leitung einer seriellen Schnittstelle dazu verwendet. Dazu braucht man natürlich entweder einen PC der eine serielle Schnittstelle hat, oder man nimmt einen USB-Dongle der eine serielle Schnittstelle anbietet.

Ich habe das *Ganze* auf einer kleinen Lochrasterplatine aufgebaut wie nachfolgend zu sehen ist:

Ich habe das *Ganze* auf einer kleinen Lochrasterplatine aufgebaut wie nachfolgend zu sehen ist:



Wer sich die Mühe machen möchte kann alles noch viel kompakter machen und eventuell sogar im Gehäuse des USB-Audio Dongles unterbringen. Ich empfehle an dieser Stelle wieder einmal die hervorragende Dokumentation des *Direwolf* Projekts (10).

Der Empfang von APRS Signalen

Im folgenden Abschnitt benötigen wir zunächst einmal nur den Aufnahmepfad unserer Soundkarte. Ich empfehle die Installation der folgenden Hilfsprogramme die im Standard **Debian** repository verfügbar sind:

```
sudo apt install alsa-utils sox
```

Ein nicht essentielles, aber im Problemfall recht hilfreiches, Programm ist *alsacap* von Volker Schatz (9). Da kein Debian Paket verfügbar ist muss man es selbst kompilieren. Man kann diesen Schritt für den Anfang überspringen und später nachholen, sollte man das Programm brauchen. In einem Verzeichnis deiner Wahl führe die folgenden Befehle aus:

```
wget https://www.volkerschatz.com/noise/alsacap.tgz
tar -xzf alsacap.tgz
cd alsacap
make<br />
sudo make install
```

Anmerkung: Im Allgemeinen ist es keine gute Idee einfach Befehle aus dem Internet zu kopieren und auszuführen wenn man keine Idee hat was genau die tun. Besonders heikel sind Befehle in denen **sudo** vorkommt. sudo fragt nach dem root Passwort und führt die folgenden Anweisungen dann mit **root** Rechten aus. (root ist der Benutzer Account der Alles darf.) Wer also hier unsicher ist, dem sei empfohlen sich ein wenig besser mit den Grundlagen des Betriebssystems Linux auseinanderzusetzen. In unserem Fall holt der erste Befehl Daten von der Webseite von Volker Schatz. Ob man dieser Webseite vertraut muss jeder für sich entscheiden. Das ist nicht anders als mit jeder anderen Software auch, die man aus dem Internet lädt. Der nächste packt die Dateien aus und die restlichen drei bauen das Programm aus den Sourcen zusammen und installieren es im System.

Auf einem typischen Laptop System ist es wahrscheinlich nicht die beste Idee die Haupt Soundkarte für unsere Packet Radio Aktivität zu verwenden, da wir in diesem Fall erst die Sound-Infrastruktur der Benutzerumgebung *loswerden* müssen, da wir exklusiven Zugriff auf die Soundkarte benötigen. Eine externe USB Soundkarte ist für unseren Zweck deshalb besser geeignet. Bevor wir die Soundkarte an den USB Port anschließen sehen wir uns an welche Soundkarten bereits im System sichtbar sind:

```
    arecord -l
```

könnte zum Beispiel die folgende Ausgabe liefern:

```
    **** Liste der Hardware-Geräte (CAPTURE) ****
    Karte 0: PCH [HDA Intel PCH], Gerät 0: ALC898 Analog [ALC898 Analog]
      Sub-Geräte: 0/1
      Sub-Gerät #0: subdevice #0
```

Alternativ kann man auch

```
    cat /proc/asound/cards
```

ansehen:

```
    0 [PCH            ]: HDA-Intel - HDA Intel PCH
                         HDA Intel PCH at 0x2ffff20000 irq 167
```

Steckt man nun die Soundkarte in den USB Slot so findet man z.B.:

```
    cat /proc/asound/cards
    0 [PCH            ]: HDA-Intel - HDA Intel PCH
                         HDA Intel PCH at 0x2ffff20000 irq 167
    1 [Device         ]: USB-Audio - USB Audio Device
                         C-Media Electronics Inc. USB Audio Device at usb-
0000:00:14.0-6.1, full speed
```

Beim neu hinzugekommenen Device handelt es sich offenbar um unsere USB Karte. Im nächsten Schritt kontrollieren wir den Eingangspegel und den korrekten Anschluss. Selbstverständlich kann man dazu das mit einer komfortablen grafischen Schnittstelle ausgestattete Programm **audacity** verwenden, es geht aber auch von der Kommandozeile aus mit

```
arecord -D <device> -V stereo -f S16_LE -c 2 -r 48000 /dev/null
```

<device> ist dabei ein Platzhalter für unsere USB Soundkarte. Für den im Beispiel oben gezeigten Fall lautet *<device>*

```
arecord -D plughw:Device,0 -V stereo -f S16_LE -c 2 -r 48000 /dev/null
```

Device ist der Text, der in den eckigen Klammern steht. Statt des Namens hätten wir auch die Zahl 1 nehmen können: plugw:1,0. Der Nachteil dabei ist, dass sich diese Zahl ändern kann wenn wir weitere zusätzliche Hardware hinzufügen.

In etwa die folgende Ausgabe sollte nun auf der Kommandozeile erscheinen:

```
    Aufnahme: WAVE '/dev/null' : Signed 16 bit Little Endian, Rate: 48000 Hz,
stereo
            +######################### 73%|73%######################### +
```

Wenn der Pegel bei 0% ist, so ist definitiv etwas falsch. 99% auf der anderen Seite ist nicht falsch, zeigt aber eine recht hohe Austeuerung an. Der Pegel sollte sich mit dem Programm **alsamixer**, das man in einer separaten Console öffnet einstellen lassen.

Tipp: Wenn man den speziellen Dateinamen /dev/null durch einen echten Dateinamen ersetzt: z. B.: *meineaufnahme.wav* so kann man diese Datei später anhören oder auch *offline* dekodieren. Das ist beim Debugging oder wenn man jemanden um Hilfe bittet mitunter hilfreich.

Wer es nicht ohnehin schon getan hat sollte spätestens jetzt das Programm **direwolf** installieren:

```
sudo apt install direwolf
```

Ich empfehle die sehr gute mit **direwolf** kommende Dokumentation zu lesen. Trotzdem gebe ich hier ein paar kurze Tipps:

Um direwolf als normaler Benutzer in der Konsole zu starten braucht man eine Konfigurationsdatei im Heimverzeichnis. Eine Musterdatei findet man in

```
/usr/share/doc/direwolf/conf/direwolf.conf.gz
```

Diese Datei dekomprimiert man und kopiert sie in das eigene Heimverzeichnis. Mit einem Texteditor muss man zumindest die richtige Soundkarte wählen:

```
ADEVICE plughw:Device,0
```

in unserem Fall. Da wir fürs Erste nur hören ist es nicht wichtig, kann aber nicht schaden unter MYCALL das eigene Rufzeichen einzutragen. Was es mit dem optionalen SSID auf sich hat soll uns fürs erste egal sein. Die Zahl 7 ist hier keine schlechte Wahl. Außerdem setzen wir den Parameter MODEM auf

```
MODEM 1200
```

Der Aufruf des Programms **direwolf** sollte dann in der Konsole die gehörten APRS Aussendungen dekodiert anzeigen:

Herzliche Gratulation der erste Schritt ist geschafft! Es ist klar, dass wir nur an der Obefläche von Packet Radio gekratzt haben. Wir haben außerdem die derzeit populärste Variante, nämlich *APRS* an den Beginn gestellt, auch deshalb, weil vielleicht der eine oder die andere damit schon Kontakt gehabt haben und nun einmal sehen, wie wenig für APRS tatsächlich benötigt wird.

Ich habe im weiteren vor in dieser Artikelserie auf das *echte* Packet Radio genauer einzugehen und zu zeigen, wie das mit Linux *Bordmitteln* unter ausschließlicher Verwendung von freier Software zu bewerkstelligen ist. Dabei soll auch das Thema Packet und HAMNET nicht zu kurz kommen. Geplant ist auch auf **PAT** den freien Winlink Client einzugehen.

Linksammlung

Die (unvollständige) Sammlung der Links verweist typischerweise auf Seiten in englischer Sprache. Leider sind viele der Seiten seit langem nicht mehr gepflegt und möglicherweise auch schon vom Netz verschwunden. In diesem Fall bewährt sich häufig eine Suche im Internetarchiv (6).

- Linux Amateur Radio AX.25 HOWTO [1]
- Direwolf [2]
- linux-ax25.org [3]

- Linux und Amateur Radio [4]
- PAT Winlink Client [5]

Linux und Schmalband Packet Radio mit Terminal

Linux und Schmalband Packet Radio mit Terminal

In diesem Artikel versuche ich zu zeigen, wie man mit Linux “Bordmitteln” eine Verbindung zu einem Packet Radio Knoten aufbauen kann. Wir kommen damit zum Sendebetrieb. Die Beschreibung eines geeigneten Adapterkabels und den Empfang von Signalen habe ich im Artikel *Linux und Amateur Packet Radio* beschrieben. Ich werde in diesem Artikel auf Software für die man den *WINE* Emulator (1) benötigt bewusst verzichten. Nicht, dass mich jemand falsch versteht: *WINE* ist ein großartiges Projekt und es ist absolut nichts falsch daran es zu verwenden, speziell dann wenn es keine andere Lösung gibt. Wir wollen es eben mal ohne *Alkohol* versuchen, hi.

Obwohl auch auf Kurzwelle nicht ganz unmöglich, so wird Schmalband Packet Radio normalerweise im 2m und 70cm Band über Transceiver gemacht, die nur FM Modulation beherrschen. Darauf konzentrieren wir uns auch.

Sehen wir uns an, was wir benötigen, so stoßen wir darauf, dass das ein so genannter *TNC*, ein **T** erminal **N**ode **C**ontroller ist. Die Aufgabe eines TNC’s ist es Daten, die wir in paketierter Form übergeben, unter Kontrolle eines Protokolles, in unserem Fall *AX.25*, an eine Gegenstelle zu übermitteln.

Solche TNC’s waren ursprünglich, in den 80ern des 20. Jahrhunderts, reine Hardwarelösungen und deshalb auch recht unflexibel. Für die Abwicklung von Protokollen sind Computer prädestiniert und seitdem sie ausreichend hohe Geschwindigkeit haben können sie auch die Aufgaben der analogen Signalverarbeitung übernehmen, die in so einem TNC anfallen. Wer schon mit digitalen Übertragungsverfahren im Amateurfunk zu tun hatte weiß, dass oft eine Soundkarte ausreicht. So ist es auch in unserem Fall.

Trotzdem geht es nicht ganz ohne Hardware. Eine Verbindung zwischen Rig, also unserem Transceiver, und dem Computer muss her. Diese Verbindung besteht aus zwei Teilen:

1. Eine analoge Schnittstelle für Audio Übertragung und
2. eine digitale Schnittstelle für die *PTT* Steuerung, also die Sendertastung.

Die Beschreibung eines geeigneten Kabels findet sich, wie bereits gesagt, in dem Artikel *Linux und Amateur Packet Radio*. Wir werden in diesem Abschnitt nun zunächst den Sendefall vorbereiten. Es ist hilfreich wenn euch zusätzlich zum Transceiver ein Kontrollempfänger zur Verfügung steht. Ein Empfänger der FM demodulieren kann ist ausreichend. Wer einen Empfänger hat der auch SSB fähig ist auf 2m oder 70cm, der kann darüberhinaus den exakten Hub seiner Aussendung einstellen. Das Verfahren dazu heisst *Besselnull* und ich werde es eventuell an anderer Stelle im Wiki beschreiben. Wie gesagt, das ist aber nicht unbedingt nötig.

Wir installieren die **hamlib** und **sox**. Dabei handelt es sich einerseits um eine Bibliothek, die es Applikationsprogrammen erlaubt verschiedene Rigs über eine einheitliche Schnittstelle anzusprechen und andererseits um ein vielseitiges Audio Werkzeug.

```
sudo apt install libhamlib-utils libhamlib-doc sox
```

Wir könnten den Transceiver damit über die CAT Schnittstelle steuern und insbesonders die PTT Steuerung vornehmen. In meinem Beispiel mache ich aber davon keinen Gebrauch sondern beschränke mich darauf den RTS Pin der seriellen Schnittstelle zu schalten.

Ein Hinweis scheint angebracht: Falls die Standard Pegellage des RTS Pins bei eurem Transceiver *ungünstig* liegt, so kann es sein, dass der Transceiver sofort in den Sendebetrieb geht sobald das Kabel angeschlossen wird. Schließt das Kabel deshalb erst an sobald die Software eingerichtet ist.

Generell ist es kein Fehler zunächst einmal mit einem Multimeter die Pegel zu überprüfen bevor man das Funkgerät einstöpselt.

Nun kann man das Kommando rictl benutzen um manuell die PTT Funktion zu testen. Bei mir lautet das:

```
rictl -p /dev/ttyUSB1 -P RTS
```

Welches die richtige serielle Schnittstelle ist müsst ihr zuvor herausfinden. Ich empfehle dazu sich einmal anzusehen welche Schnittstellen es im System gibt solange der USB Dongle nicht angeschlossen ist. Das geht mit

```
ls /dev/ttyUSB*
```

Dann schließt man den Dongle an und gibt das Kommand erneut. Die Schnittstelle die dazugekommen ist muss die Gesuchte sein. Es gibt auch noch andere Methoden die dann zum Einsatz kommen sollten wenn man seine Installation dauerhaft machen will. Dazu mehr bei der Einrichtung des *soundmodem*.

Nachdem also das rictl Programm gestartet wurde kann man von dessen Kommandozeile interaktiv PTT auf on oder off setzten. **T 1** sollte das RTS Signal einschalten und **T 0** aus. Wenn man die Funktion mit dem Multimeter validiert hat, so kann man den 6-pol mini DIN Stecker ans Funkgerät anstecken und den Versuch wiederholen. Der Transceiver sollte sich nun auftasten lassen.

Immer noch an der Kunstantenne, ihr habt doch bis jetzt alles an der Dummy-Load gemacht, so hoffe ich, schalten wir nun unseren Kontrollempfänger ein und stellen ihn auf die Frequenz unseres Transceivers ein. Dann geben wir das Kommando

```
AUDIODEV=plughw:Device,0 play -n synth sine 1000 gain 0
```

ein. *Device* ist dabei wieder der Name der Soundkarte, den wir schon bei der Inbetriebnahme der Empfangsseite verwendet haben. Der Wert *1000* steht für 1000Hz und der gain 0 bedeutet maximale Aussteuerung der Soundkarte. Es ist nun zu empfehlen den Trimmer, wo vorhanden, am Adapterkabel auf einen mittleren Wert zu stellen. Auch keine schlechte Idee ist es zunächst mit einem Kopfhörer zu überprüfen ob die 1000Hz am Ausgang der Soundkarte zu hören sind. Wenn alles klar ist so schließen wir das Kabel an den Transceiver an. Sobald wir die PTT mit Hilfe von ricgctl einschalten sollte das Signal nun auch im Kontrollempfänger zu hören sein. Man kann nun vorsichtig den Trimmer größer drehen, bzw. die Lautstärke am Regler der Soundkarte. Man sollte dabei so vorgehen, dass das Signal möglichst groß wird, aber keines Falls zu stark, da dann störende Verzerrungen auftreten. Mein ICOM Receiver macht es mit einfach, da er ganz einfach aufhört zu senden, sobald die Aussteuerung (der Hub) zu groß wird. Ich gehe bis an diese Grenze heran und drehe den Trimmer dann um einen *Tick* kleiner.

Wer will kann an dieser Stelle mit Direwolf weiter machen, oder meinem Weg folgen und das Soundmodem von Thomas Sailer installieren, allerdings in einer von mir überarbeiteten und weitergeführten Version.

…wird fortgesetzt…