GNU-Radio

Ein einfacher Einstieg in die SDR-Technik

Inhaltsverzeichnis

Vorwort

Wenngleich gut aufgebaute klassische Funkempfänger den rein digitalen Varianten noch ein Stückchen voraus sind, sind Softwarelösungen weit flexibler und bieten eine Reihe zusätzlich möglicher Funktionen, die traditionelle Dampfradios nicht aufweisen können. Dieser Vorteil der Digitaltechnik betrifft aber nicht nur den reinen Empfang sondern jede Art der Signalverarbeitung weshalb diese Technik sich schon seit einiger Zeit auch im Amateurfunk ausgebreitet hat.

Für den klassischen Radiobastler ist es allerdings nicht so einfach von der analogen Welt in die des Programmierers zu wechseln. Anstatt eines Lötkolbens braucht es plötzlich Programmierkenntnisse was so Manchen davon abhält, sich näher mit der SDR-Technik zu befassen. GNU Radio ermöglicht den Umstieg von der reinen Analogtechnik auf die digitale SDR-Technik auch ohne Programmierkenntnisse. Wer allerdings tiefer einsteigen möchte, wird nicht ganz ohne Programmierkenntnisse auskommen.

In diesem Vortrag soll das GNU-Radio System vorgestellt werden. Dabei werden die verschiedenen Themen allerdings nur angerissen. Wer tiefer einsteigen möchte, dem seien die Seiten unter https://wiki.gnuradio.org/index.php/Tutorials empfohlen.

Was ist GNU Radio?

GNU Radio ist ein freier Open Source Werkzeugkasten mit dem sich Signalbearbeitungsblöcke mit graphischen Mitteln zu kompletten Geräten wie z.B. SDRs (Software Defined Radios) zusammenstellen lassen. Dafür bietet das System eine große Anzahl (über 500) Signalverarbeitungsblöcke (von der Anbindung an SDR-Sticks bis zur graphischen Ausgabe oder eine Ausgabe über Soundkarten) an. Zusätzlich zu der großen Zahl vorhandener Blöcke kann man auch eigene Blöcke erzeugen, entweder durch Zusammenfassung bereits vorhandener, durch Modifizierung oder man kann selbst komplett neue Blöcke schreiben. GNU-Radio lässt sich auf UNIX- und Mikrosoft Systemen installieren. Die Programmiersprachen sind Python und C++ wobei Python vorwiegend für die weniger zeitkritischen Teile verwendet wird während in den zeitkritischen DSP-Pfaden C++ zur Anwendung kommt.

1. Die Grundprinzipien

Startet man das GNU Radio Programm, so wird eine Zeichenfläche aufgeblendet in der bereits zwei beliebig verschiebbare Blöcke zu sehen sind. Der linke (größere Block) dient dazu, dem Projekt das man starten möchte mit einem Namen zu versehen. Der zweite Block ist ein Block mit dem Variablen definiert und mit Werten besetzt werden können. Nachdem es kein digitales System ohne Abtastrate gibt, wird dieser Variablenblock grundsätzlich vorgegeben.

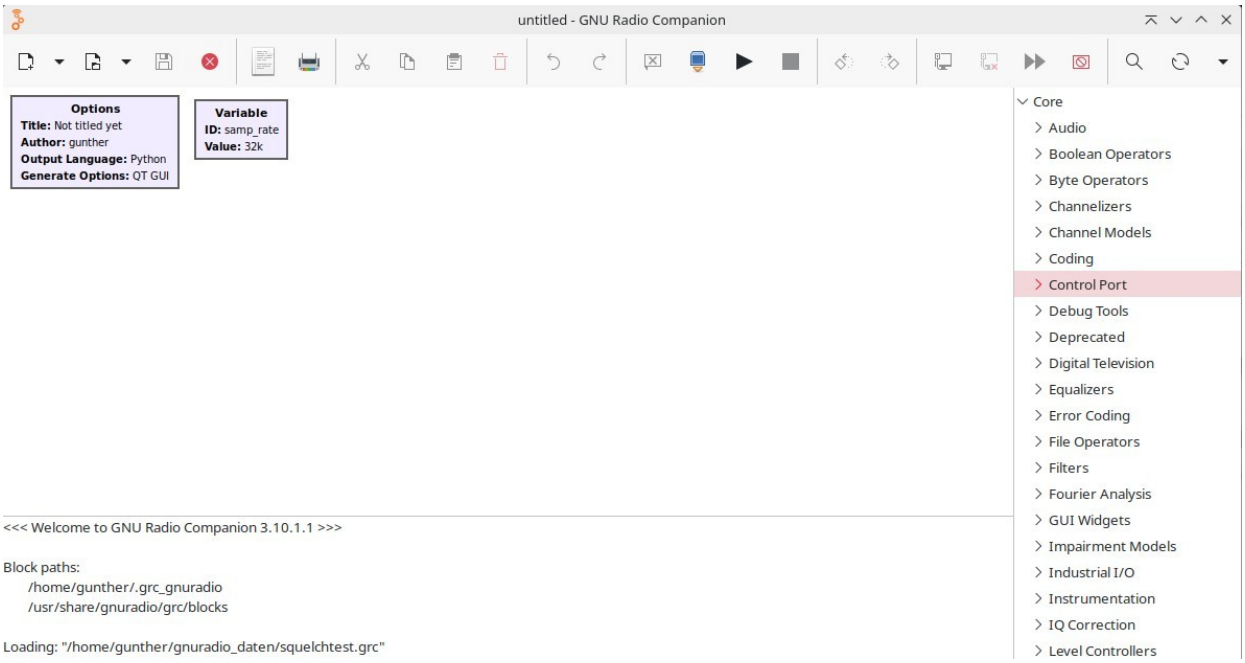

Bild1 : Grundbild direkt nach dem Start

Durch einen Doppelklick auf den jeweiligen Block wird dieser „geöffnet“ und man kann dort neue Werte eintragen oder vorgegebene verändern. Unter der Lasche „Documentation“ gibt es bei einigen Blöcken noch eine Kurzbeschreibung.

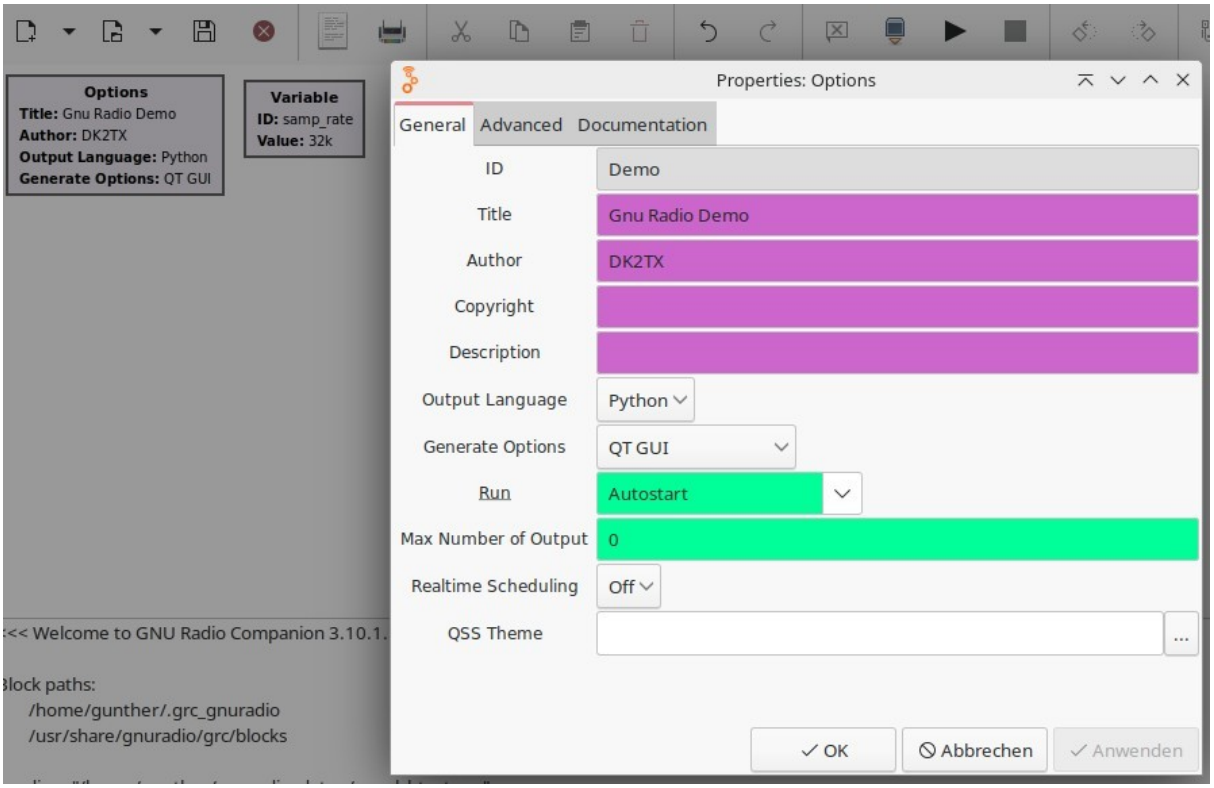

Bild 2: Geöffneter Options-Block

Auf der rechten Seite (siehe Bild 1) ist eine Spalte mit einem Baum von Blöcken zu sehen. Um sein geplantes Projekt aufzubauen zieht man die gewünschten Blöcke aus dieser Spalte auf die Seite links daneben. Um sich bei der großen Anzahl von Blöcken nicht zu verzetteln, ist es meist sinnvoll sich mit Hilfe von Einträgen in dem mit einer Lupe versehenen Feld nur noch eine Auswahl anzeigen zu lassen.

Die Blöcke haben farbig markierte Stecker, links die Eingänge, rechts die Ausgänge, wobei die Farben den Typ des dort anschließbaren Datenstroms bezeichnen. Es gibt 15 Datentypen, deren Farben hier im Bild zu sehen sind

Bild 3: Datentypen und deren Farbkennzeichnung

Die Blöcke lassen sich durch wechselseitiges Anklicken der Stecker verbinden wobei sich natürlich nur Ein- und Ausgänge des gleichen Datentyps miteinander verbinden lassen. Ein Ausgang kann dabei mit beliebig vielen Eingängen verbunden werden während an ein Eingang verständlicherweise nur mit einem Ausgang verbunden werden kann. Bei einer Reihe von Blöcken lässt sich der Typ von Ein- und Ausgängen ändern.

Zusätzlich zu diesen direkten Datenverbindungen gibt es noch Meldungen. Das sind asynchrone Mitteilungen an Bausteine mit denen etwas Vordefiniertes ausgelöst werden kann. Weiterhin gibt es noch als „Tags“ bezeichnete Datensätze, die an Abtastwerte angehängt werden können und die mit den betroffenen Abtastwerten im Datenstrom mitlaufen. Das können z.B. Zeitstempel sein.

2. Das Erste Flussdiagramm

Um die oben beschriebenen Grundprinzipien zu zeigen, soll hier ein Sinusgenerator benutzt werden dessen Ausgangssignal in einer Art Oszilloskop und in einem Spectrum Analyser angezeigt werden

soll. In dem Bild taucht ein weiterer mit „Throttle“ bezeichneter Block auf. Dieser Block ist notwendig um das Projekt „auszubremsen“ wenn keine Hardware vorhanden ist die den wirklichen Takt vorgibt. Der Block gibt den Datenstrom 1:1 weiter, verändert ihn also nicht. Durch Anklicken des Dreiecks in der Kopfzeile lässt sich das Flussdiagramm ausführen.

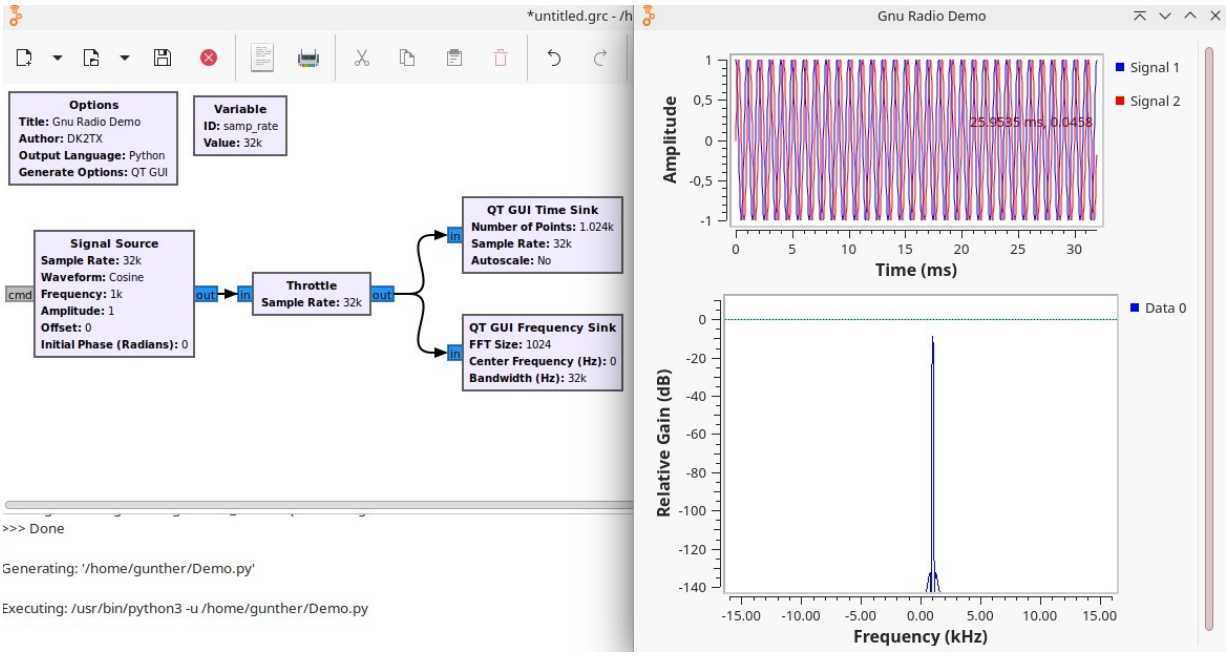

Bild 4: Sinusoszillator mit zwei Ausgabeblöcken

2.1 Die Wirkung von Variablen des „Variable“-Blocks

Im vorherigen Beispiel waren im Signalgenerator die Abtastrate und die Frequenz als Zahlenwerte eingetragen. Statt dessen kann man aber auch Variablenblöcke in das Diagramm setzen, in der Zeile ID den Namen der Variablen eintragen und darunter den Wert. Auf der Zielseite trägt man z.B. anstatt der Frequenz den Variablennamen für die Frequenz ein. Bei Start des Flussdiagramms wird dann der Wert der Variablen übernommen. Wird der Variablenwert während des laufenden Betriebs verändert, hat das vorerst keine Wirkung. Erst bei einem Neustart wird der neue Wert übernommen.

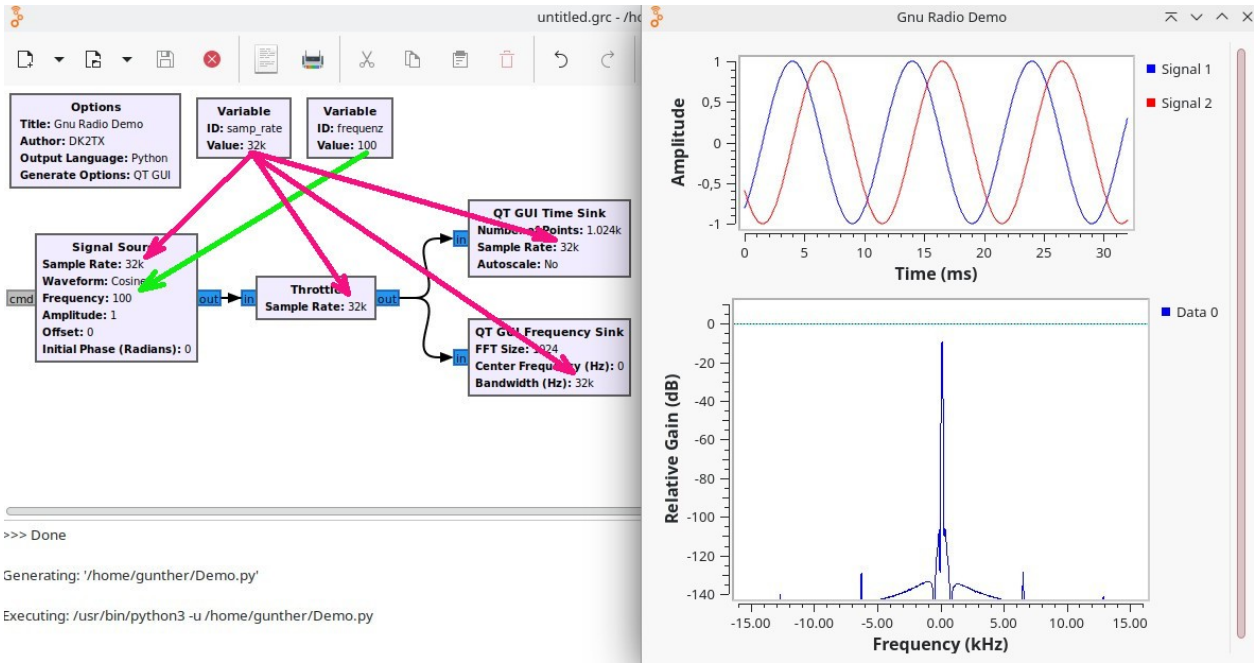

Bild 5: Sinusoszillator mit Variablenblock für die Frequenz

2.2 Während der Laufzeit veränderliche Variable

Oft will man während der Laufzeit Werte verändern wie z.B die Frequenz eines Empfängers. Dafür gibt es einen eigenen Block „QT GUI Range“. Auch dort kann am unter ID einen Variablennamen definieren und in den Zeilen darunter dann den Startwert, den niedrigsten und den höchsten Wert eintragen. Außerdem kann man wählen mit welchen graphischen Mitteln der Wert verändert werden soll (Drehknopf, Schieber, Zahl). Im Beispiel unten wurde ein Schieber und ein Zahlenfenster gewählt

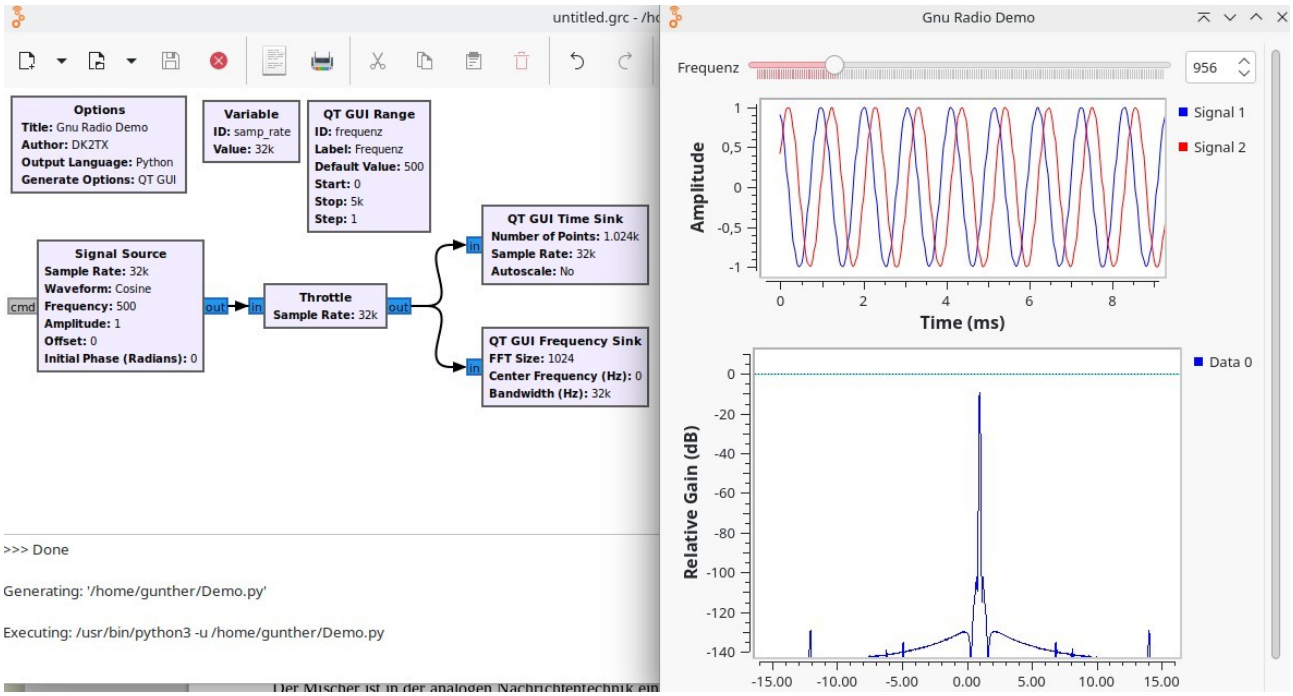

Bild 6: Sinusoszillator mit GUI Range Block für die Frequenzänderung

3. Aufbau eines Mischers

Der Mischer ist in der analogen Nachrichtentechnik eine ganz zentrale Komponente. Ein solcher Mischer soll deshalb als erste Anwendung von GNU Radio gezeigt werden.

3.1 Der klassische Mischer

Beim klassischen Mischer wird das umzusetzende Signal mit dem eines Lokaloszillators multipliziert. Das Ergebnis ist dann eine Mischung aus zwei Signalen gleicher Amplitude mit der Summe und der Differenz der beiden Eingangsfrequenzen. In der analogen Welt muss aus diesen beiden das gewünschte Signal ausgefiltert werden:

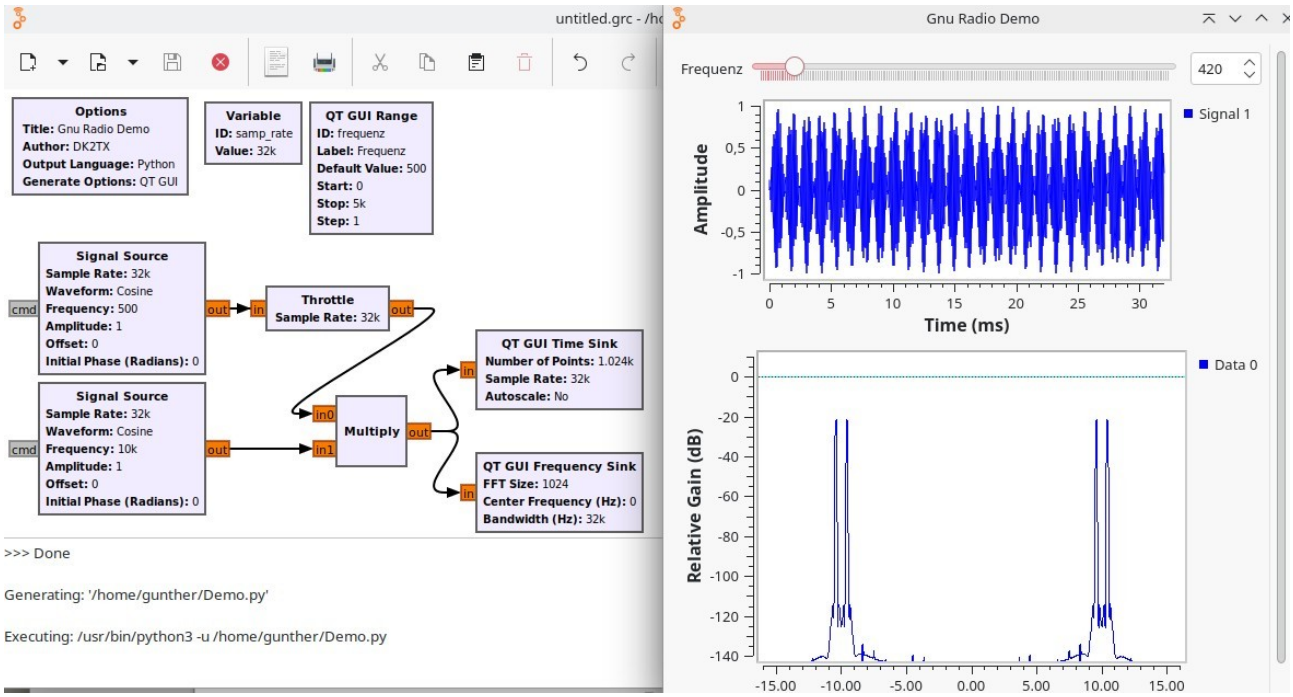

Bild 7: Mischer mit festem Lokaloszillator (hier 10kHz) und einem variablen Oszillator

3.2 Der komplexe Mischer

Bei GNU Radio gibt es die Möglichkeit anstatt mit float-Signalen (wie im oberen Beispiel), mit komplexen Signalen zu arbeiten. Mathematisch ausgedrückt sieht das so aus:

Signal des Oszillators 1: $s_1 = a_1 e^{j\omega_1 t}$;

Signal des Oszillators 2: $s_2 = a_2 e^{\omega_2 t}$;

Nach der Multiplikation beider Signale entsteht dann: $s_1 s_2 = a_1 a_2 e^{j\omega_1 t} e^{j\omega_2 t} = a_1 a_2 e^{j(\omega_1+\omega_2)t}$;

Das ist ein komplexes Sinus-Signal mit der Summe der beiden Oszillatorfrequenzen.

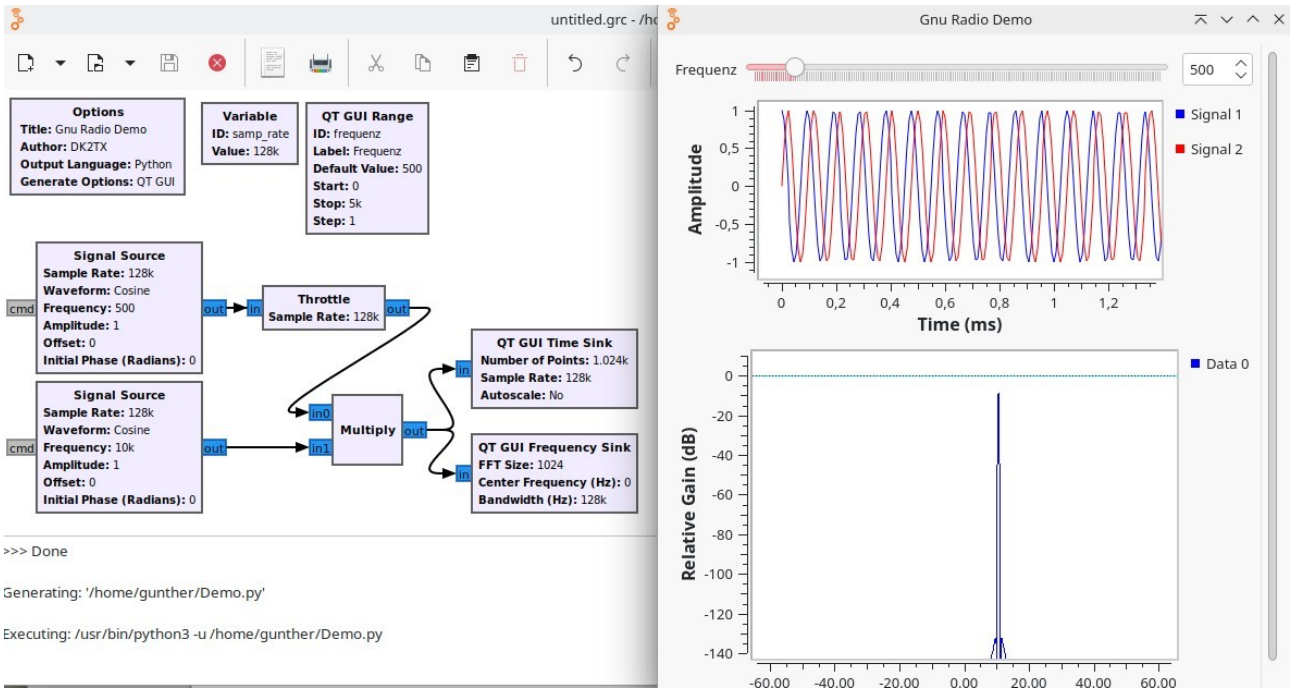

Bild 8: Mischer, wie Bild 7 nur mit komplexen Signalpfaden

Multipliziert man den Imaginärteil z.B. des Oszillators mit -1, konjugiert das Signal also, erhält man als Mischprodukt die Differenz der beiden Oszillatorfrequenzen. Wie man erkennen kann, ist bei diesem Mischer keine nachträgliche Filterung erforderlich.

4. Filter

Filter sind in der Analogtechnik ebenfalls sehr wichtige Komponenten. In der digitalen Welt ist das nicht viel anders. Auch bei GNU Radio gibt es eine große Anzahl verschiedener Filter, wobei diese Filter für meinen persönlichen Geschmack etwas spartanisch beschrieben sind. Etwas ausgeglichen wird der „Mangel“ dadurch, dass eine große Anzahl von Beispielen gibt in denen diese Filter eingesetzt werden und aus denen man die optimalen Einstellungen dann ablesen kann.

4.1 Decimating Filter

Gegenüber den analogen Filtervarianten gibt es in der digitalen Welt einige mehr. So sind die „Decimating Filter“ etwas, was man in der analogen Welt nicht benötigt.

Oft ist es so, dass man auf der Hochfrequenzseite von Anwendungen hohe Abtastraten braucht weil die Abtastfrequenz mindestens doppelt so hoch sein muss als die Bandbreite des zu bearbeitenden Signals. Auf den niederfrequenten Seite würde nur ein Bruchteil der Abtastfrequenz genügen. Mit einem zwischengeschalteten „Decimating Filter“ lässt ich die Abtastfrequenz auf einen Bruchteil der hohen einstellen und der niederfrequente Teil dann mit der kleinen Abtastrate betreiben.

4.2 IIR Filter

IIR-Filter sind solche mit nicht endlicher Impulsantwort. Sie enthalten intern Rückkopplungen, sodass diese Filter, einmal angeregt, theoretisch endlos ausschwingen. Das Verhalten dieser Filter entspricht dem der Analogtechnik.

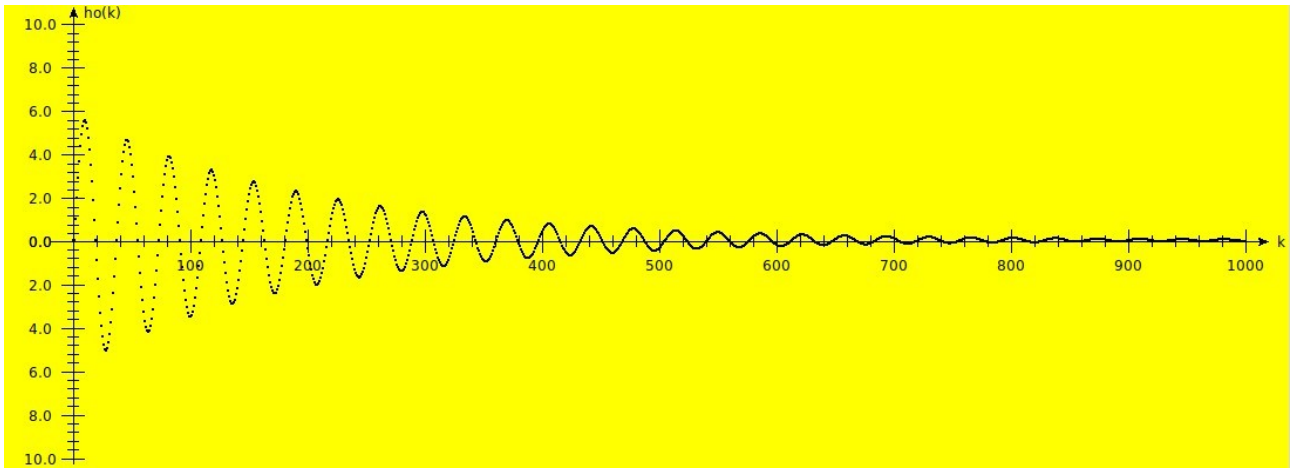

Bild 9: Impulsantwort eines 2-poligen IIR Bandpasses

Will man sich den Frequenzgang eines solchen Filters ansehen, kann man z.B. eine Rauschquelle nutzen und sich das Ergebnis mit dem „Frequency Sink“ Block ansehen.

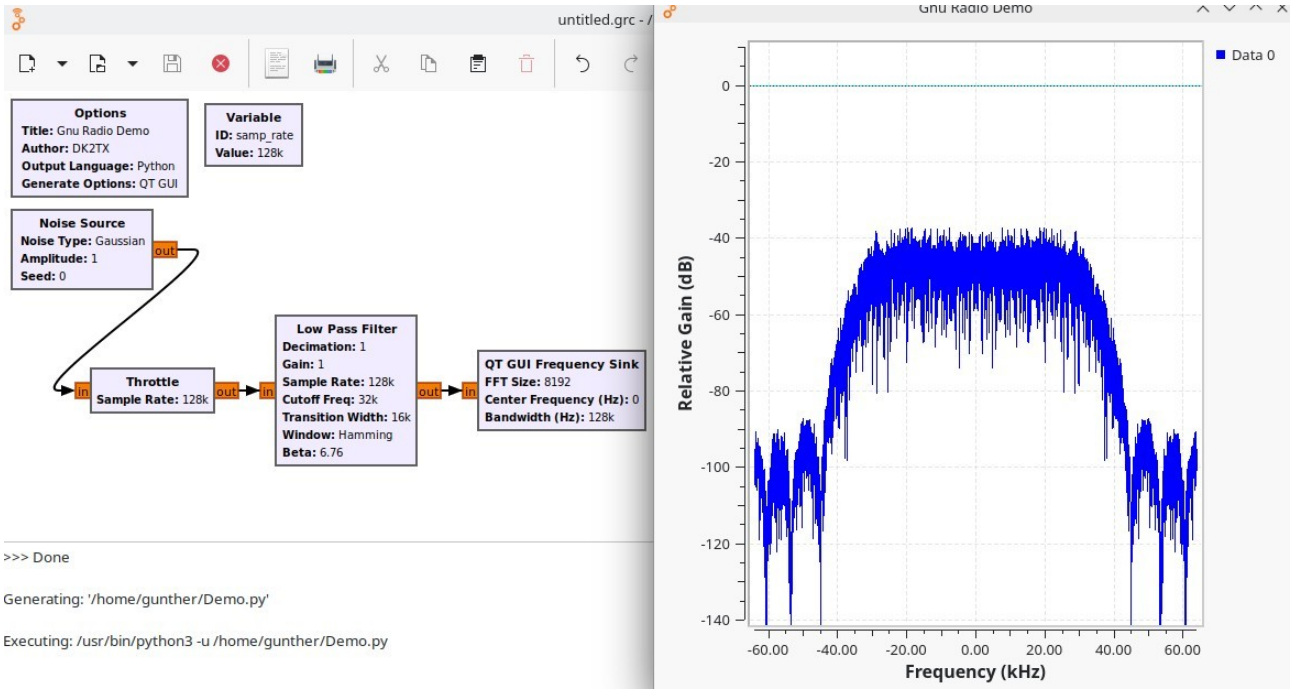

Bild 10: Tiefpass mit einer Rauschquelle ausgemessen

Das Problem von IIR-Filtern ist, dass sie in der Nähe der Resonanzfrequenzen rasche Änderungen der Phasenlage zeigen. Bei akustischen Anwendungen (also normalen Radios) stört das nicht weil

wir diese Art der Phasenverschiebungen nicht hören. Digitale Dekodierungsverfahren können dabei aber schnell außer Tritt geraten.

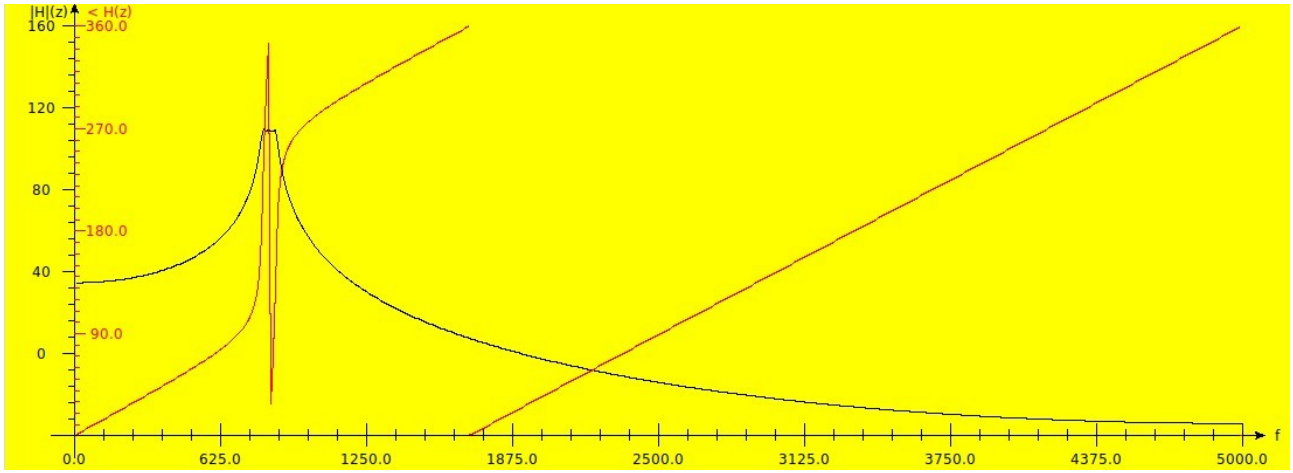

Bild 11: Betrag und Phase eines 6-poligen IIR Bandpasses

4.3 FIR-Filter

FIR-Filter können in der analogen Welt nur mit großem Aufwand realisiert werden, während sie in der digitalen Welt recht einfach zu bauen sind. Trotzdem bleibt ein Nachteil: Während man bei IIR-Filtern mit niedrigen zweistelligen Polzahlen schon ansehnliche Flankensteilheiten erreichen kann, sind bei FIR-Filtern Polzahlen im hohen 3-stelligen Bereich erforderlich. Entsprechend hoch ist dann der Rechenaufwand.

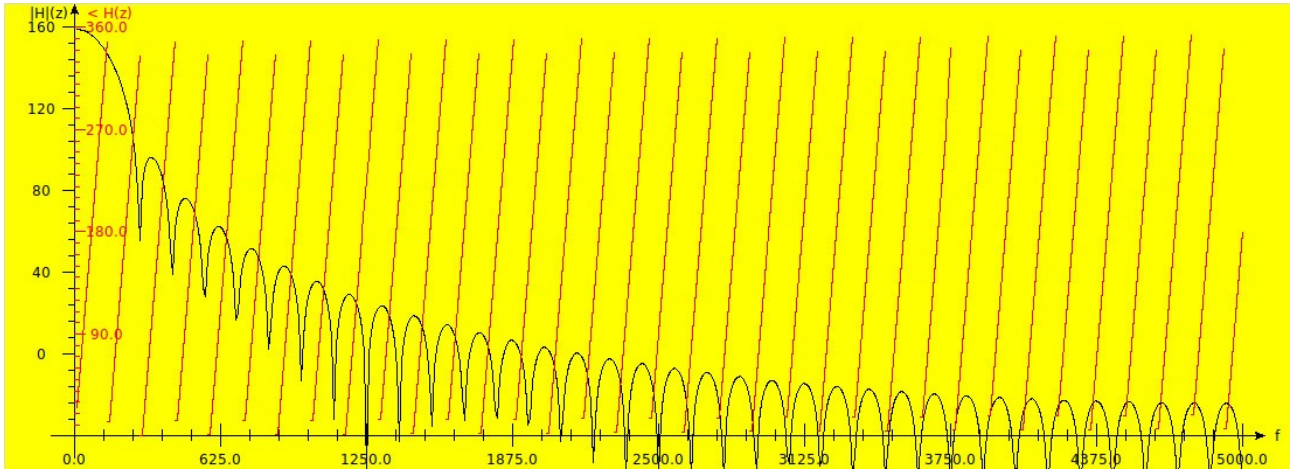

Bild 12: Betrag und Phase eines 138-poligen FIR-Tiefpasses

4.4 Filter mit „Filter Taps“ (Filterkoeffizienten)

Neben Filtern mit vordefinierten Filterkurven gibt es auch solche deren Verhalten erst durch die extra mitzuliefernden Filterkoeffizienten definiert wird. Zum Filter müssen deshalb zusätzlich die Koeffizienten geladen werden.

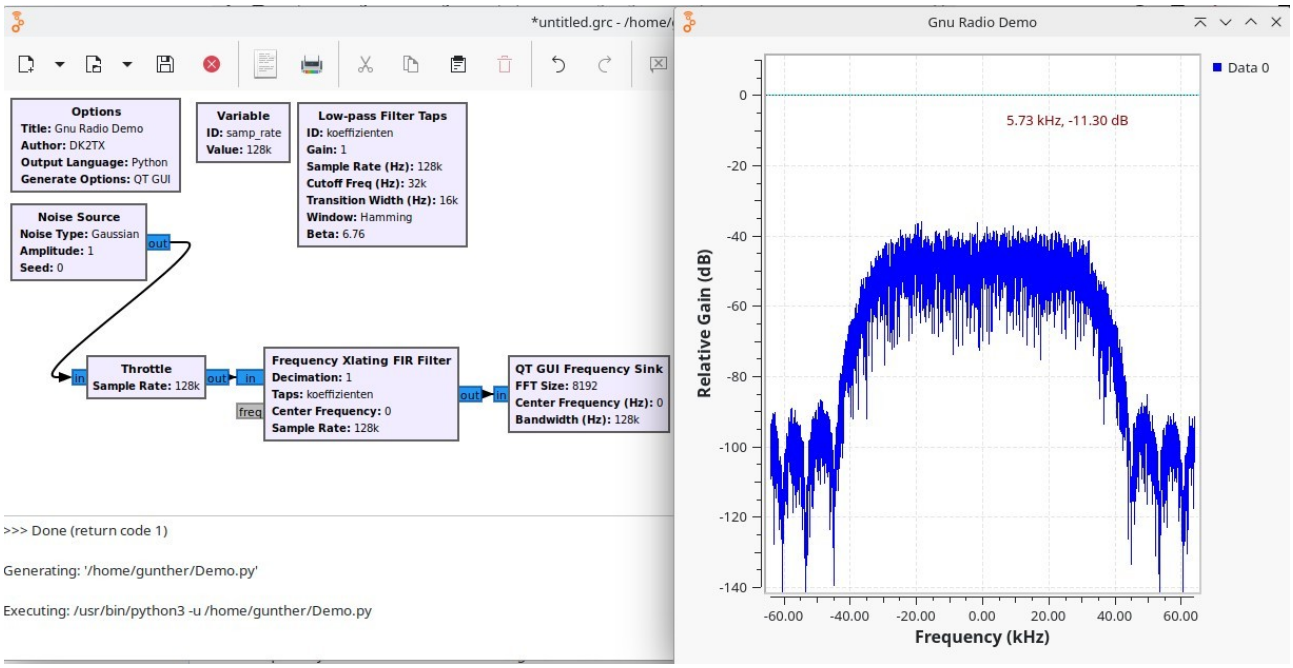

Bild 13: Tiefpass bei dem die Koeffizienten extern vorgegeben werden.

Im Prinzip sehen das Flussdiagramm und das Ergebnis so aus wie in Bild 10, nur dass hier Filter und Koeffizienten getrennt sind. Das gibt einem die Möglichkeit selbst Filterkoeffizienten zu schreiben und das Filter damit zu versorgen. Die Mathematik, die man für die Berechnung der Koeffizienten braucht ist allerdings nicht trivial. Außerdem habe ich noch keine Beschreibung gefunden, wie die Filterstruktur selbst aussieht und in welcher Reihenfolge die Koeffizienten im Filter abgerufen werden.

Neben der Versorgung mit dem Block „Low Pass Filter Tabs“ gibt es auch die Möglichkeit in der Zeile „Taps“ die Koeffizienten mit Hilfe eines Koeffizientengenerators direkt einzutragen wie z.B.

```
firdes.low_pass(1.0, Abtastrate, Abtastrate/(decimation * 2), Transition_Bandbreite)
```

4.5 Frequency Xlating FIR Filter

In einigen Fällen wie z.B. bei der Demodulation von SSB-Signalen möchte man nicht mit dem Basisband hantieren sondern mit einer Zwischenfrequenz. Für diese Fälle gibt es einen Filterblock der alles miteinander erledigt: Heruntertakten der Abtastrate, Umsetzung des Signals auf die

gewünschte Zwischenfrequenz und Filterung auf die gewünschte Bandbreite. Wie so ein Filter eingesetzt wird, ist im Punkt „SSB Demodulation“ zu sehen.

5. Meldungen

Meldungen sind in den Flussdiagrammen asynchrone Nachrichten d.h. die Zeitspanne von der Generierung bis zum Eintreffen beim Empfänger ist nicht definiert. Um Meldungen nutzen zu können wird man in den meisten Fällen selbst Funktionsblöcke schreiben müssen. Auf die Weise, wie sowas geht, soll hier nicht eingegangen werden denn das erfordert Programmierkenntnisse in der Programmiersprache Python. Meldungen kann man mit Hilfe eines „Debug-Blocks“ ausgeben.

Zur Demonstration wird im unten gezeigten Flussdiagramm ein GUI-Range Block eingesetzt, mit dem sich der Wert der Variablen „frequenz“ ändern lässt. Der Baustein „Variable to Message“ generiert bei jeder Änderung der Variablen eine Meldung. Der „Message Debug“-Block gibt diese Meldungen links unten als Zeichenfolge aus.

Im Gegensatz zu allen anderen Verbindungen, werden die Meldungsverbindungen als gestichelte Linie dargestellt.

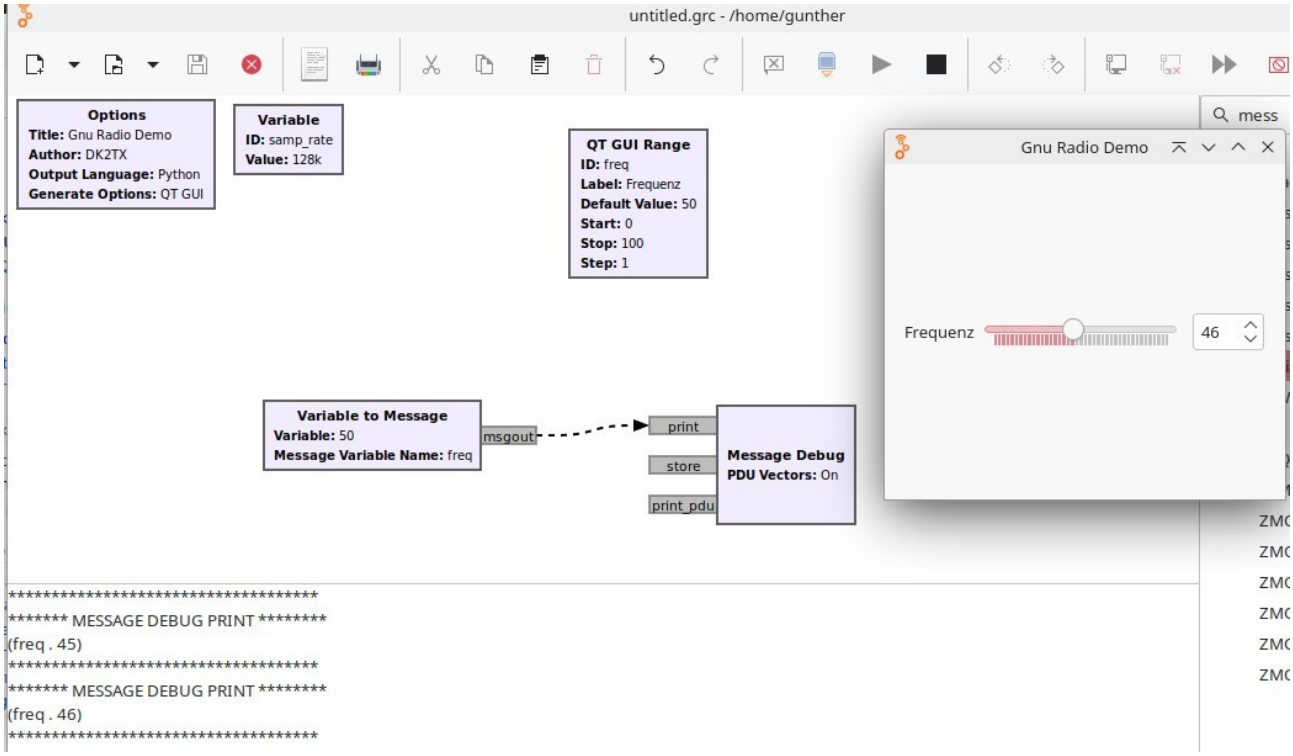

Bild 14: Ein Beispiel für die Generierung von Meldungen.

6. Tags

Im Gegensatz zu Meldungen sind Tags Nachrichten, die mit einzelnen Abtastwerten fest verbunden werden. So kann ein bestimmter Abtastwert der z.B. durch einen Zeitstempel markiert wurde auch

nach einer längeren Bearbeitungskette noch identifiziert werden. Ähnlich wie bei den Meldungen, erfordert auch die Arbeit mit Tags einen größeren Programmieraufwand. Deshalb soll hier auch auf diesen Punkt nicht näher eingegangen werden.

Was allerdings noch erwähnt werden soll ist die Tatsache, dass einige Blöcke von sich aus Tags generieren. Gibt man einen solchen Datenstrom mit dem „GUI Time Sink“-Block aus, werden diese Tags in Textform sichtbar. In ungünstigen Fällen kann es vorkommen, dass die im Time Sink Fenster ausgegebenen Kurven durch sehr viele Tags so überdeckt werden, dass man die Kurven selbst nicht mehr erkennen kann. Das Problem lässt sich lösen, in dem man einen „Tag Gate“ Block einschleift. Der „vernichtet“ die Tags sodass am Ausgang ein Datenstrom ohne Tags entsteht.

7. Datenquellen

In den bisherigen Beispielen wurden als Datenquellen die Blöcke „Signal Source“ und „Noise Source“ benutzt. Für die Kommunikation mit der Aussenwelt existieren weitere Funktionsblöcke.

So gibt es die Möglichkeit Datenströme über TCP, UDP Schnittstellen, die Soundkarte, Dateien und eine ganze Reihe von SDR-Sticks einzulesen.

7.1 Virtual Source

Diese Datenquelle braucht im Flussdiagramm ein Gegenstück – den „Virual Sink“-Block. Der „Virtual Source“ Block gibt genau den Datenstom aus, der in den „Virtual Sink“-Block eingespielt wird. Damit lassen sich Flussdiagramme weit übersichtlicher gestalten als ohne denn die standardmäßigen, automatisch generierten Verbindungen können bei komplexeren Flussdiagrammen auch unter Blöcken durchlaufen sodass es schwierig wird die Verbindungen zu verfolgen.

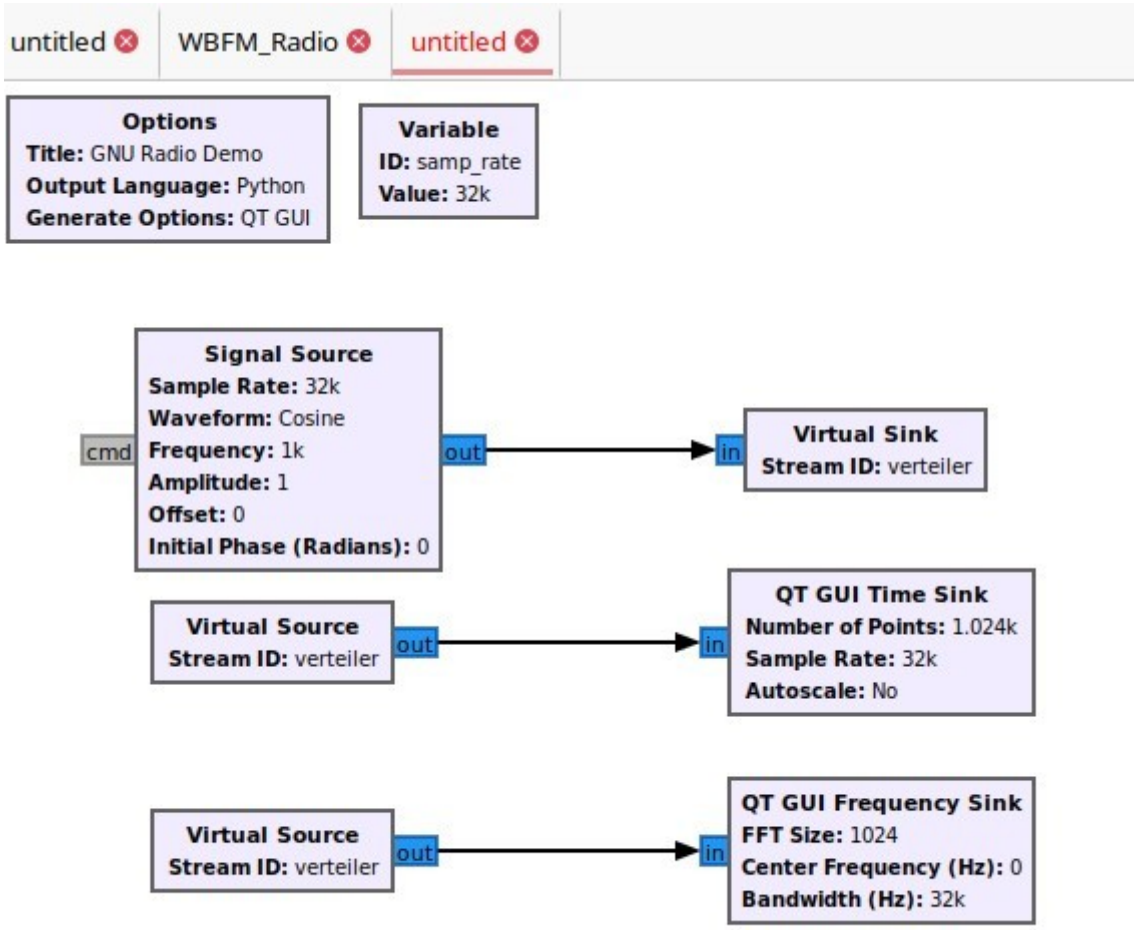

Bild 14: Bild mit Virtual Source Blöcken um die Übersichtlichkeit zu verbessern

7.2 Audio Source

Hier ein einfaches Beispiel in dem der Mikrofoneingang einer Soundkarte eingelesen wird. Der Datenstrom könnte dann z.B. mit einem Equalizer weiterverarbeitet oder über einen GNU Radio FM- oder SSB-Sender ausgegeben werden.

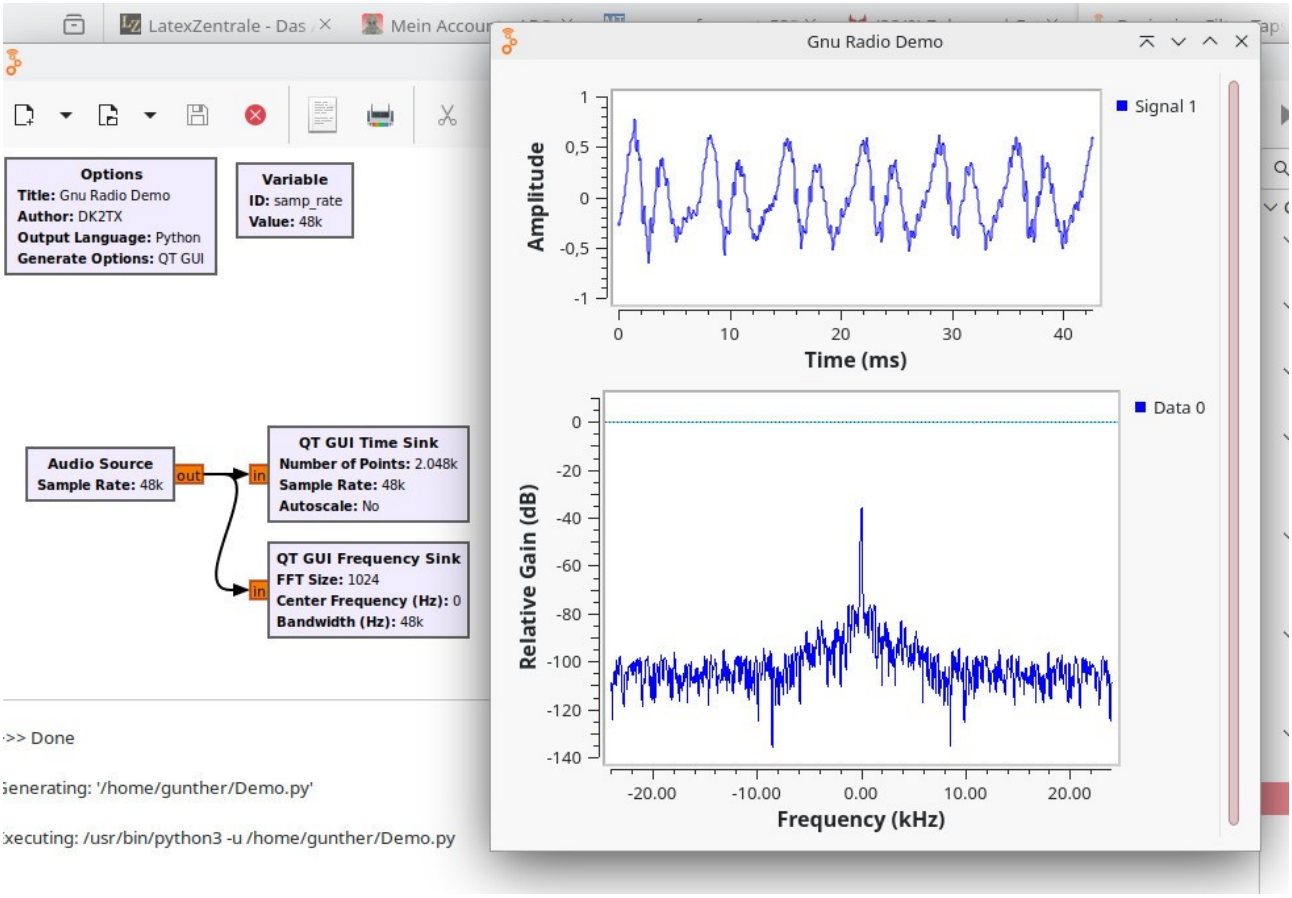

Bild 15: Eingabe eines Signals über die Mikrofoneingang einer Soundkarte

7.3 PlutoSDR Source

Was bei einem Namen wie GNU **Radio** nicht fehlen darf ist natürlich die Möglichkeit Datenströme von SDR-Sticks einzulesen. Einer davon ist das ADALM-Pluto System. Damit ist es dann möglich, Signale von 70 MHz bis über 3000 MHz zu empfangen und weiterzuverarbeiten. Im folgenden Beispiel wird die Ausgabe eines auf das UKW Rundfunkband eingestellten ADALM Pluto auf einen „Frequency Sink“-Block gelenkt. Dort lassen sich ein Aussendungen der Rundfunksender erkennen.

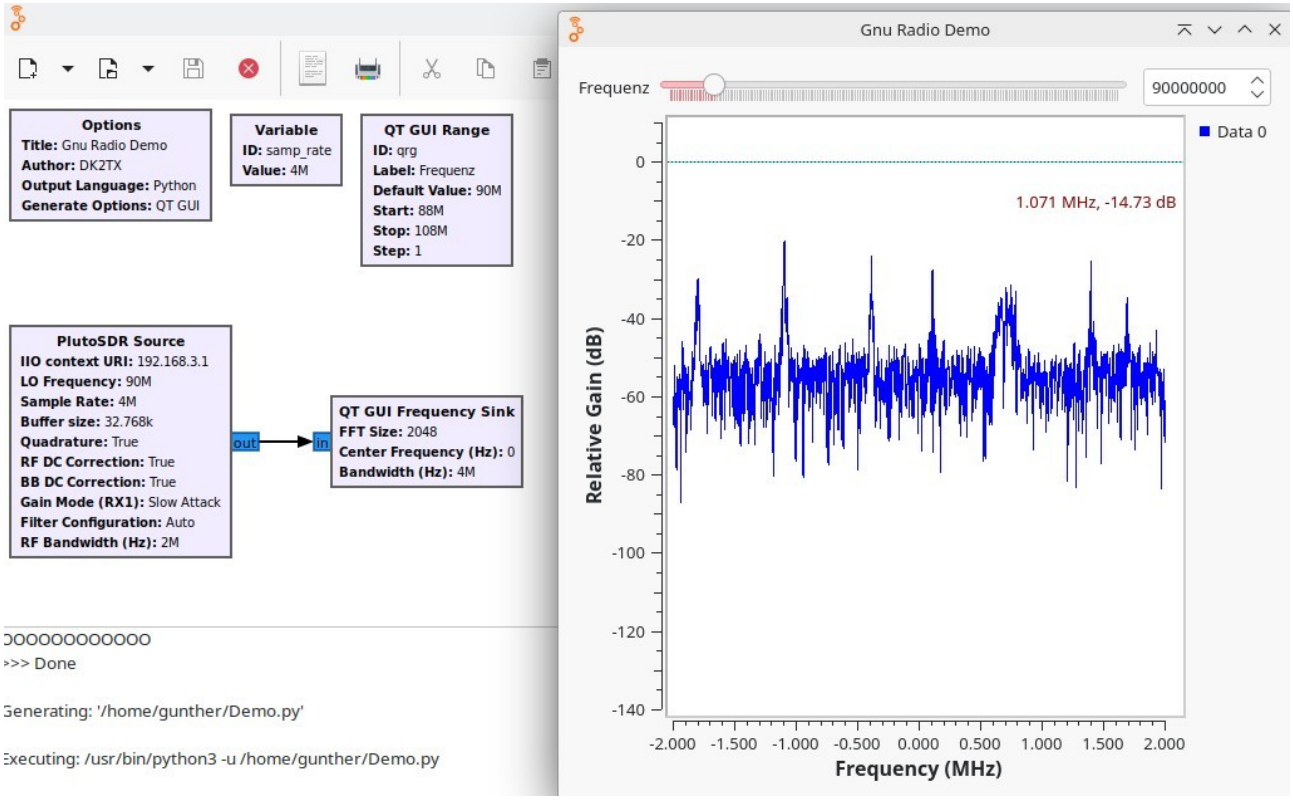

Bild 16: Ausgaben der ADALM Pluto im Rundfunkband um 90 MHz

8. Datensenken

Als Gegenstücke zu den Datenquellen die im vorherigen Kapitel beschrieben wurden gibt es eine große Zahl von Datensenken. Einige davon wurden bereits zur Anzeige von Datenströmen benutzt. Neben diesen Anzeigeblöcken gibt es weitere rein interne wie den „Null Sink“ Block oder den schon erwähnten „Virtual Sink“ Block. Schließlich gibt es noch die, die mit der Außenwelt kommunizieren können. Das kann über UDP, TCP, die Soundkarte oder über eine Datei als Ziel stattfinden. Schließlich gibt es noch das Gegenstück zu den SDR-Sticks. Einige dieser „Sticks“ können. Wie z.B. der ADALM Pluto können auch auf einer vorgegebenen Frequenz senden.

8.1 Virtual Sink

Dieser Block ist das Gegenstück zum „Virtual Source“ – Block und wurde in Punkt 7.1 bereits beschrieben.

8.2 Null Sink

Das Flussdiagramm lässt einen Start nur zu denn alle vorhandenen Signalausgänge mit mindestens einem Eingang verbunden sind. Gelegentlich möchte man einen Block in einer Schaltung nur zum Test nutzen und den dann nicht deaktivieren oder sogar ganz löschen wenn man ihn gerade nicht

braucht. Abhilfe biete das der „Null Sink“-Block der den betroffenen Ausgang korrekt entsorgen kann.

8.3 Audio Sink

Über den „Audio Sink“-Block lassen sich Float-Datenströme über die Soundkarte ausgeben. Bei den Abtastraten kann am aus einer vorgegebenen Liste auswählen. Stelle man die Anzahl der Eingänge im Block auf 2, so kann man mit diesem Block Stereo-Signalfolgen ausgeben. In dem folgenden Beispiel wird das Signal eines Sinusgenerators ober eine Balanceregler auf die beiden Eingänge des „Audio Sink“-Blocks gegeben. In realen Anwendungen könnte das Audiosignal von einem Empfänger stammen.

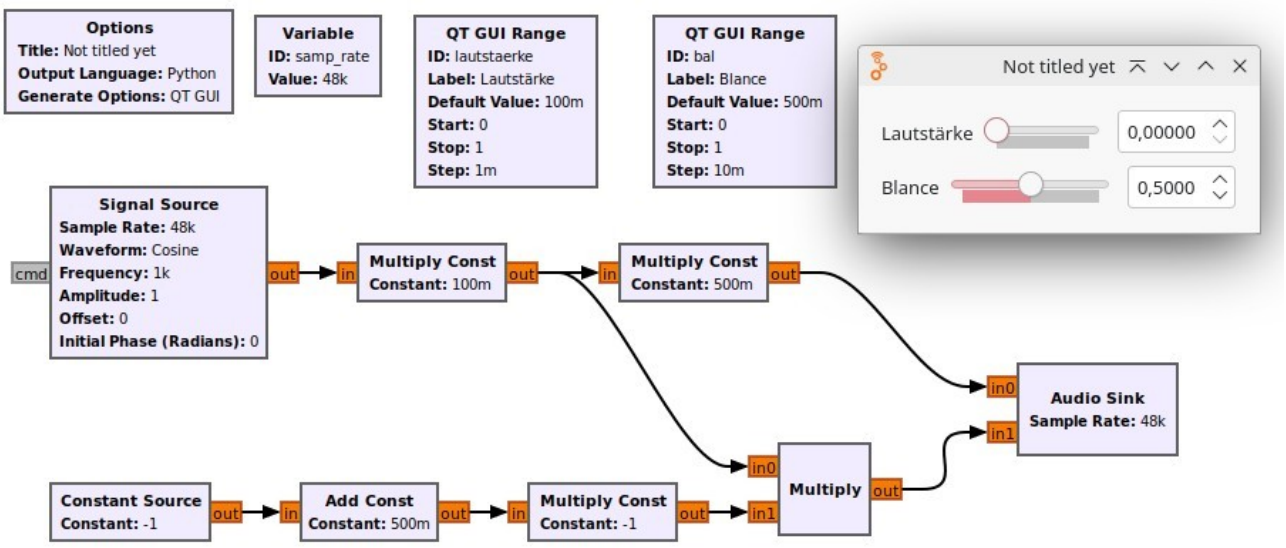

Bild 18: Ausgabe eines Signals über einen Balanceregler auf einen „Audio Sink“-Block

8.4 PlutoSDR Sink

Der ADALM-Pluto lässt sich in beiden Richtungen betreiben, als Empfänger (PlutoSDR Source) und als Sender (PlutoSDR Sink). Für einen Sender der z.B. einen Ton in FM auf 145,550 MHz ausgibt, reichen neben den beiden Standardblöcken drei zusätzliche Blöcke aus. Der Tongenerator ließe sich auch durch einen Audio Source Block ersetzen und damit ein kompletter FM-Sender bauen.

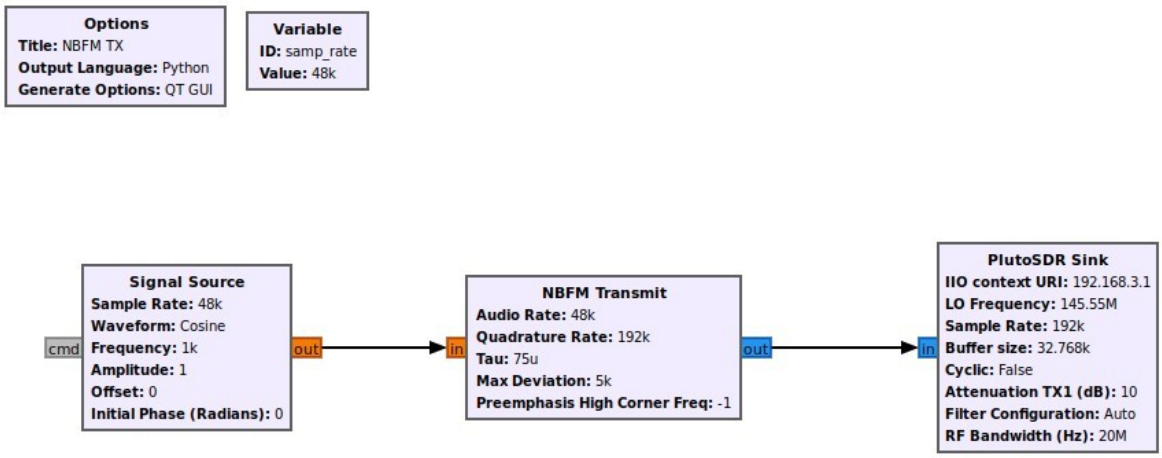

Bild 19: Ausgabe eines 1kHz Tons als FM-Ton auf 145,550 MHz

8.5 GUI Sink

Neben den Ausgaben Richtung Außenwelt gibt es natürlich auch die Möglichkeit, sich die Signale intern anzusehen. Zwei der möglichen Ausgaben wurden in den bereits gezeigten Beispielen schon mehrfach gezeigt, es gibt aber auch noch eine Reihe anderer.

8.5.1 GUI Time Sink

Der „GUI Time Sink“ Block gibt Signalverläufe über der Zeit aus, so wie wir es vom Oszilloskop her kennen. Die Beschriftung der Zeitachse geschieht automatisch und hängt von der Zahl der auszugebenden Punkte sowie von der Abtastrate ab. Maßstab und Beschriftung der Y-Achse lassen sich individuell einstellen.

Neben den Signalen selbst werden, wie im Punkt „Tags“ beschrieben, auch Tags angezeigt. Diese Anzeige lässt sich entweder mit einem Gate Block verhindern und die Tag-Ausgabe in der GUI Time Sink Parametrierung abschalten.

Möchte man einen vergrößerten Ausschnitt der ausgegebenen Kurve sehen, kann man mit der linken Maustaste ein Rechteck um den gewünschten Bereich ziehen. Lässt man die linke Maustaste wieder los, wird der vergrößerte Bereich angezeigt. Mit der rechten Maustaste kommt man wieder auf die Originaldarstellung zurück.

Mit dem Mausrad lässt sich der Maßstab der Y-Achse verändern.

8.5.2 GUI Frequency Sink

Der „GUI Frequency Sink“ Block gibt Signale über der Frequenz aus, so wie wir es vom Spectrum-Analyzer her kennen. Die Beschriftung der Frequenzachse geschieht automatisch und hängt von der Abtastrate ab. Der Frequenzbereich geht (wählbar) von -Abtastrate/2 bis +Abtastrate/2 oder von 0 bis Abtastrate/2. Maßstab und Beschriftung der Y-Achse lassen sich individuell einstellen.

Das Anzeigen von Ausschnitten oder Verändern des Maßstabs geht, wie in Punkt 8.5.1 beschrieben.

9. Blöcke zur Demodulation

Hat man über die Blöcke wie Signalquellen, Filter und Mischer letztendlich das Basisband mit der Nutzinformation isoliert, muss schließlich die Nutzinformation draus extrahiert werden. Das geschieht mit Hilfe der Blöcke zur Demodulation.

9.1. Breitband FM-Demodulation

Wie man aus dem recht einfachen Blockdiagramm erkennen kann, demoduliert er das angebotene Signal nicht nur sondern setzt auch die Abtastfrequenz um. Während die Rate am Pluto-Ausgang bei 192k liegt. Wird sie um einen einstellbaren Faktor (hier 4) reduziert um die Soundkarte, die mit 48k läuft, bedienen zu können. Das hier gezeigte Blockschaltbild ist das eines vollständigen Breitband-FM-Empfängers den man als Mono UKW Empfänger verwenden kann.

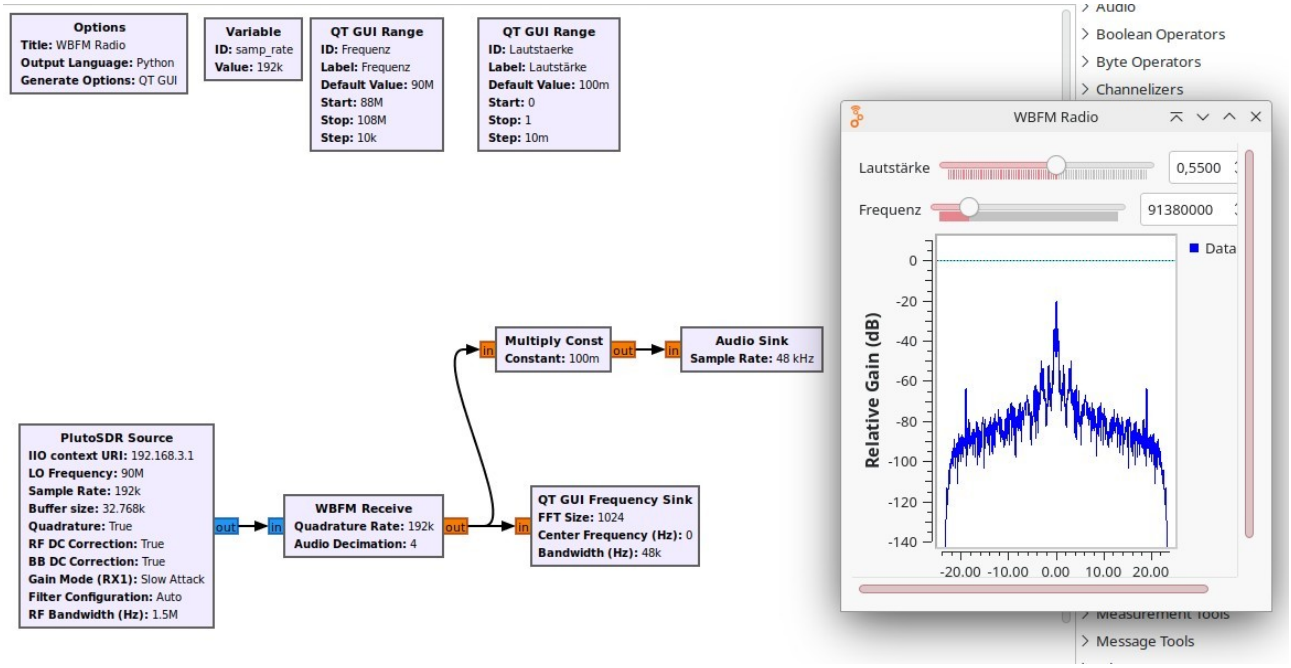

Bild 20: Demodulation eines WBFM Rundfunksignals

9.2 Schmalband FM-Demodulation

Auch für Schmalband-FM gibt es einen eigenen Demodulatorblock. Im Prinzip lässt er sich genauso verwenden wie der für den beitbandigen Fall. In dem Beispiel hier wurde aber noch ein Filter davor gesetzt, dass das zu demodulierende Signal auf 6kHz Bandbreite begrenzt und die Abtastrate des Pluto auf die Hälfte 196k) heruntersetzt. Außerdem wurde noch ein Squelch-Block vor den Demodulator gesetzt. Das hier gezeigte Blockschaltbild zeigt einen vollständigen NBFM Empfänger wie man ihn z.B. für den Empfang von S22 (145,550 MHz) einsetzen kann. Eine

„Besonderheit“ ist in dem Plan noch zu sehen: Es enthält einen deaktivierten Block. Deaktivierte Blöcke können beliebige Variablennamen oder Verbindungen haben. Diese Namen oder Verbindungen stören den Rest des Graphen nicht. In dem Fall wurde ein GUI-Range Block deaktiviert, der sich für das 70cm-Band verwenden lässt. Ein Bandwechsel lässt sich damit einfach durch Deaktivieren des einen und Aktivieren seinen Gegenstücks machen.

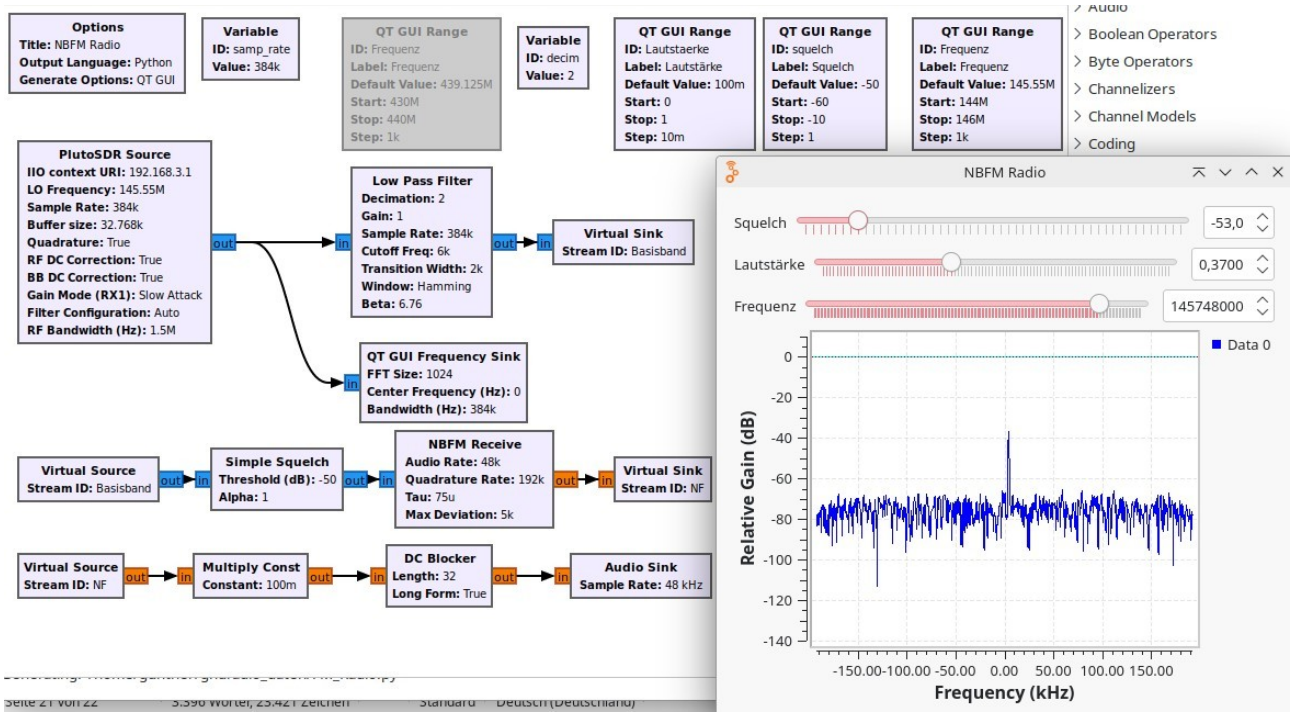

Bild 21: Demodulation eines NBFM-Signals

9.3 AM Demodulation

Auch für die Demodulation gibt es einen eigenen Demodulatorblock. Oberhalb von 70 MHz (was die Untergrenze des Pluto Empfangbandes ist), gibt es nur einen mir bekannten Funkdienst, der AM einsetzt. Um die Funktion des Demodulators zu demonstrieren, muss man deshalb auf eine der Aussendungen in diesem Bereich ausweichen.

Der hier gezeigte Empfänger zeigt allerdings einige kleine Besonderheiten. Das Signal aus dem Pluto wird zuerst in der Bandbreite auf 6kHz reduziert wie im NBFM-Fall auch.
Dann folgt eine AGC-Stufe die den Signalpegel auf einen Bereich hält, der vom Demodulator gut beherrscht werden kann. Der AM-Demodulator besteht letztendlich aus zwei Teilen, dem PLL Carrier Tracker der auf der Empfangsfrequenz einrastet und das Eingangssignal mit dieser eingerasteten Frequenz mischt. Dieses Signal wird dann auf den endgültigen AM-Demodualtor gegeben.

Eine kleine Warnung, an die, die dieses Blockbild nachbauen. Es funktioniert für Rundfunksendungen tadellos. Beim Betrieb wie er in der Fliegerei üblich ist, ist kein Signal in der Luft wenn niemand sendet. Die AGC dreht die Verstärkung in dieser Zeit auf Maximum. Wird jetzt

ein Träger eingeschaltet erzeugt das im System eine sehr hohe Spitze, im Lautsprecher oder Köpfhörer hört man einen unangenehm lauten Knall und es dauert einige Millisekunden, sich die AGC so eingestellt hat, dass ein verzerrungsfreier Empfang möglich ist.

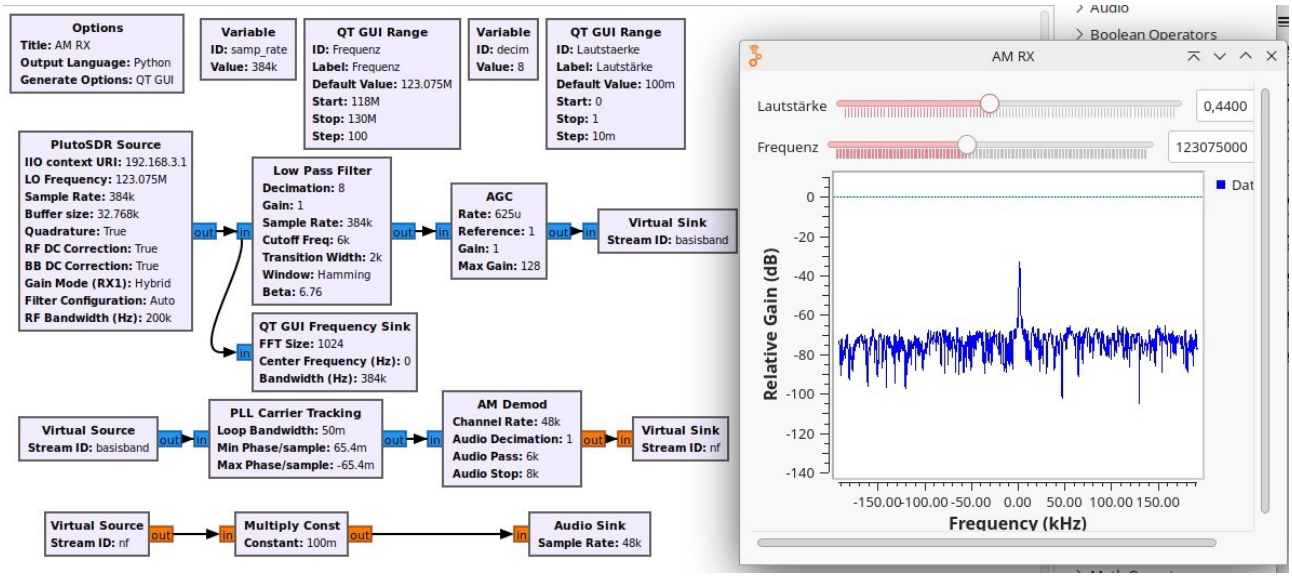

Bild 22: Demodulation eines AM-Signals

9.4 SSB Demodulation

Füe die Demodulation von SSB gibt es keinen eigenen Block. Mit in GNU Radio vorhandenen Blöcken lässt sich aber leicht ein SSB-Demodulator bauen. Im Gegensatz zu den vorherigen Beispielen gibt es aber keinen Einzelblock der für die Demodulation zuständig ist sondern zwei Oszillatoren ein I- und Q-Signale erzeugen mit deren Hilfe das komplexe ZF-Signal nach der Weaver-Methode dekodiert wird. Ersetzt man den Multiplikationsfaktor „-1“ im Multiplizierer links unten durch „1“, könne anstatt von USB-Signalen LSB-Signale dekodiert werden.

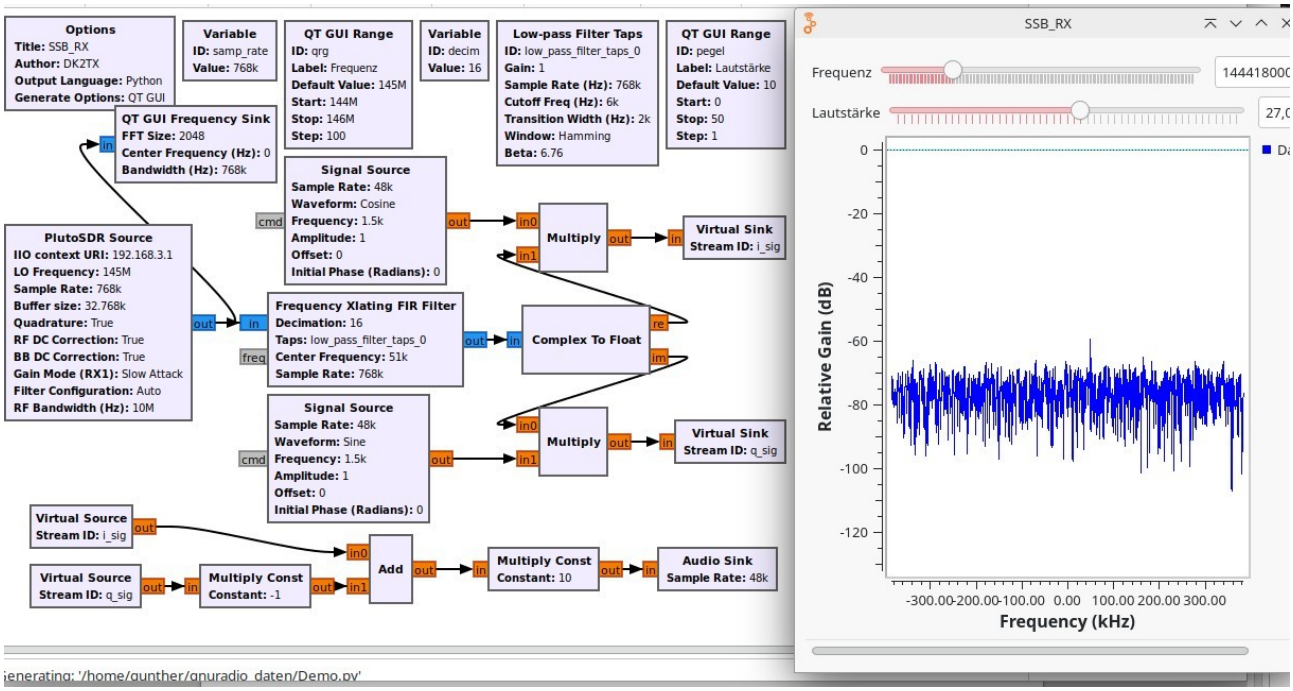

Bild 23: Demodulation von SSB-Signalen nach der Weaver-Methode

10 Quellenverzeichnis

https://wiki.gnuradio.org/index.php/Tutorials