GNUradio Workbook

1. Einleitung

GNUradio ist eine mächtiges, aber auch komplexes Programm für digitale Signalverarbeitung am PC. Es ermöglicht das Design und die Erprobung digitaler (De–)Modulatoren und Codecs. Dazu bietet es verschiedenste vordefinierte Code–Bibliotheken an, die als abstrakte Blöcke in das eigene Programm gesetzt und miteinander verknüpft werden können.

So können sehr einfach eigene Programme um digitale Signalverarbeitungsbausteine erweitert werden. Da aber oftmals bei der Nutzung von GNUradio nicht das Entwickeln von Programmen, sondern die Entwicklung der Codecs und Modulatoren selbst im Vordergrund stehen, enthält die Bibliothek auch Funktionen, um unter anderem Audio-Dateien einzulesen, Daten in verschiedensten Formaten abzuspeichern, Datenströme von Software–Defined–Radios abzuhören und mittels QT Daten visuell aufbereiten zu können. All dies wird in einer grafischen Entwicklungsumgebung — dem GNUradio Companion — zusammengeführt, um das experimentieren möglichst einfach und intuitiv zu gestalten. Also — Los Geht's!

2. Installation

In folgendem wird erklärt, wie GNUradio zu installieren ist. Folge einem der Links zu deinem entsprechenden Betriebssystem:

- Windows
- MacOS
- Linux

2.1. Windows

Die Installation unter Windows erfolgt über das Paket **radioconda**. Darin sind alle notwendigen Abhängigkeiten, sowie weitere hilfreiche Tools kombiniert.

Aufgabe A — Installation von radioconda

a) *Lade den radioconda-Installer von folgender URL herunter: https://github.com/radioconda/radioconda-installer/releases/download/2025.03.14/radioconda-2025.03.14-Windows-x86_64.exe*

b) *Führe die heruntergeladene Executable-Datei mit Administrator-rechten aus.*

c) *Folge den Anweisungen des Installers — Fertig.*

2.2. MacOS

Auf MacOS bietet das Installationspaket **radioconda** eine einfache Installationsmöglichkeit für GNUradio mit allen notwendigen Abhängigkeiten.

Aufgabe B — Installation von radioconda

a) *Lade den radioconda-Installer von folgender URL herunter:*
   - ***Intel Mac:*** *https://github.com/radioconda/radioconda-installer/releases/download/2025.03.14/radioconda-2025.03.14-MacOSX-x86_64.pkg*,
   - ***M-Chips:*** *https://github.com/radioconda/radioconda-installer/releases/download/2025.03.14/radioconda-2025.03.14-MacOSX-arm64.pkg*

b) *Führe die heruntergeladene PKG–Datei aus.*

c) *Folge den Anweisungen des Installers — Fertig.*

2.3. Linux

Es gibt für so gut wie jede Distribution ein GNUradio Paket. Auf **repology** findet man für die unterschiedlichen Systeme die paketierten Versionen. Diese können dann einfach mit dem Paket-Installer des Systems installiert werden (bspw. apt unter Debian/Ubuntu/Mint, dnf unter Fedora/RHEL/CentOS, nix-shell oder ähnliches).

3. Erste Schritte mit GNUradio

Nun, da wir GNUradio installiert haben, ist es soweit, das Interface zum ersten Mal zu öffnen.

Sollte euch zum starten von dem GNUradio companion keine Option im Startmenü bereitstehen, öffnet eine Kommandozeile und startet das Programm mittels dem Befehl gnuradio-companion.

3.1. Eine Übersicht

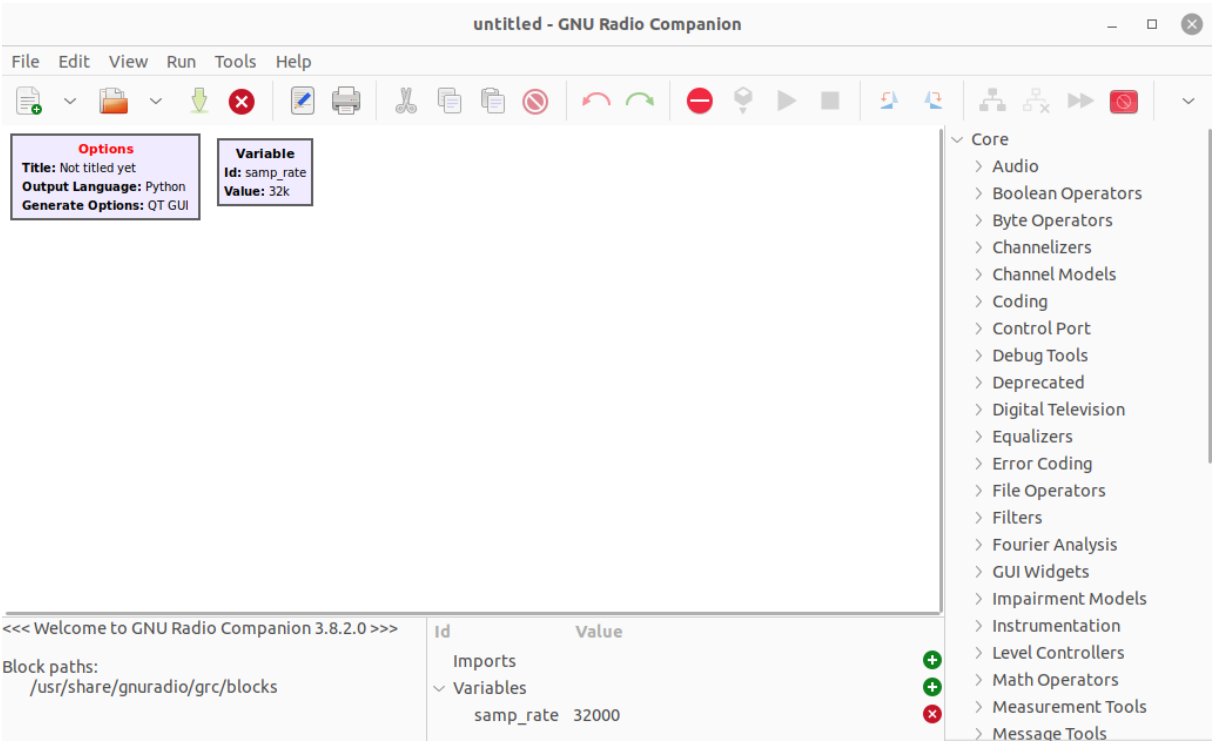

Abbildung 1: GNUradio nach erster Öffnung

Der Arbeitsbereich von GNUradio ist in 4 Bereiche unterteilt.

Im Zentrum steht der Flowgraph. Dieser ist im Moment mit zwei Blöcken — Options und Variable — ausgestattet. Im Flowgraph findet der Hauptteil der Arbeit statt. Hier werden neue Blöcke angelegt, untereinander verknüpft und Einstellungen getroffen.

Rechts vom Flowgraph findet sich die Blockbibliothek. Hier werden alle zur Verfügung stehende Signalverarbeitung-Bausteine gelistet.

Unten Links findet man den Konsolen-Output. Hier werden Fehler oder Konsolen-ausgaben der Signalverarbeitungsprogramme angezeigt. Es ist oftmals sehr hilfreich im Falle von Fehlern hier den Backtrace anzuschauen, um nachzuvollziehen, wo genau der Fehler aufgetreten ist.

Zwischen der Konsole und der Blockbibliothek ist die Variablenübersicht. Sie sammelt die im Flowgraph definierten Variablen und zeigt sie zusammen mit ihrem Wert in einer übersichtlichen Liste an.

Geht ruhig einmal alle Komponenten durch und schaut, dass ihr euch einen guten Überblick verschafft.

3.2. Quellen und Senken

Jede funktionierende Signalverarbeitungskette braucht einen Anfang und ein Ende. In GNUradio nennen wir diese Blöcke **Quellen** (Sources) und **Senken** (Sinks).

- **Quellen** generieren oder importieren Daten. Das kann ein rein mathematischer Signalgenerator sein (z.B. eine Sinus-Welle), eine Datei auf deiner Festplatte (.wav, .raw) oder echte Hardware wie ein angeschlossenes Software-Defined-Radio (SDR). Quellen haben naturgemäß **nur Ausgänge**.
- **Senken** nehmen Daten auf, um sie zu verbrauchen. Sie stellen das Ende einer Kette dar. Beispiele sind Lautsprecher (Audio Sink), Dateien zum Abspeichern von Aufnahmen oder grafische Diagramme zur Visualisierung (QT GUI Sinks). Senken haben **nur Eingänge**.

3.2.1. Die Farbkodierung der Datentypen

Ein absolut fundamentales Konzept im GNUradio Companion ist die Farbkodierung. Wenn du dir die Ein- und Ausgänge (Ports) der Blöcke ansiehst, fällt sofort auf, dass diese farbig markiert sind. Die Farbe gibt den Datentyp an. Ein Block kann nur mit einem anderen Block verbunden werden, wenn die Farben der Ports übereinstimmen!

- **Blau (Complex Float 32):** Komplexe Fließkommazahlen. Dies besteht immer aus zwei Werten: einem In-Phase (I) und einem Quadrature (Q) Anteil. Das ist der Standard für fast alle Hochfrequenz- und SDR-Anwendungen.
- **Orange (Float 32):** Normale (reelle) Fließkommazahlen. Wird oft für Audio, Basisband-Signale oder fertig demodulierte Signale verwendet.
- **Grün (Integer 32):** Ganze Zahlen (32-bit).
- **Gelb (Short 16):** Kurze ganze Zahlen (16-bit).
- **Magenta (Byte 8):** Einzelne Bytes. Dies wird oft genutzt, wenn wir am Ende einer Kette echte digitale Nutzdaten (Bitströme, Pakete, Textnachrichten) extrahieren.

Brauchst du einen anderen Typen, weil z.B. dein Filter orange ist, dein SDR aber blau ausgibt, musst du einen Konvertierungs-Block (z. B. *„Complex to Float“*) dazwischenschalten.

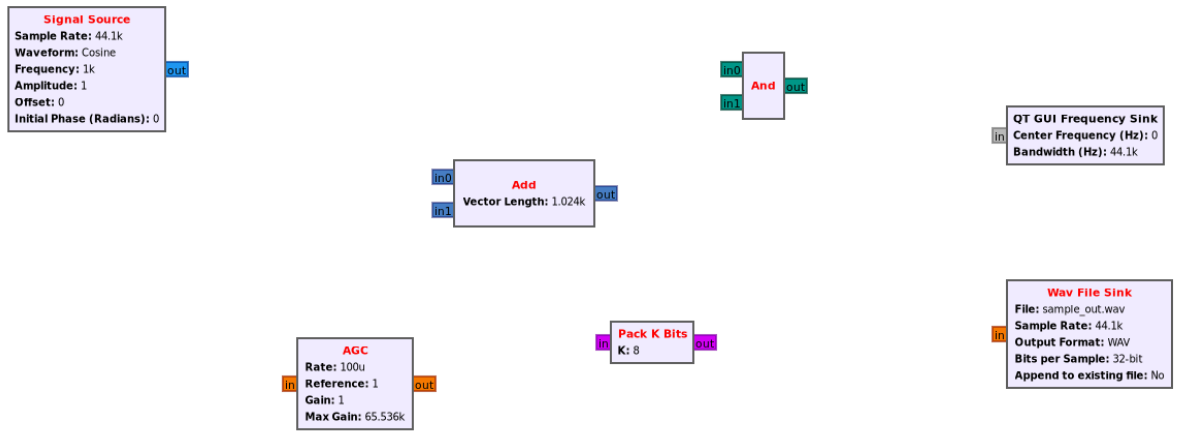

Abbildung 2: Die Farben der Ports geben den Datentyp vor. Verbindungen sind nur zwischen gleichen Farben möglich.

3.2.2. Streams, Bitstreams und Vektoren

Bevor wir unseren ersten Flowgraph bauen, müssen wir noch verstehen, **wie** die Daten durch die Blöcke fließen. GNUradio unterscheidet hier grob zwischen Streams und Vektoren.

Streams (Datenströme) Standardmäßig arbeiten fast alle Blöcke in GNUradio Stream-basiert. Stell dir das wie ein Fließband vor: Die Daten (z.B. einzelne Fließkommazahlen) tröpfeln kontinuierlich, eine nach der anderen, durch die Verbindungen. Die meisten Filter, Verstärker und Modulatoren arbeiten so. Ein „Bitstream“ ist dabei nichts anderes als ein Stream, bei dem jeder Tropfen auf dem Fließband nur eine 0 oder 1 (oft verpackt als Byte) ist.

Vektoren (Arrays/Blöcke) Manchmal reicht es nicht, einen Wert nach dem anderen zu betrachten. Das bekannteste Beispiel ist die FFT (Fast Fourier Transform), mit der wir unser Spektrum visualisieren. Um Frequenzen zu berechnen, muss der Block einen ganzen „Batzen“ an Daten gleichzeitig betrachten (z.B. 1024 Samples auf einmal). In GNUradio nennt man das einen Vektor. Das Fließband transportiert dann nicht mehr einzelne Pakete, sondern ganze Paletten. In den Block-Eigenschaften wird dies oft durch den Parameter `vlen` (Vector Length) definiert. Man muss sehr aufpassen: Ein Float-Stream (`vlen=1`) lässt sich nicht direkt mit einem Float-Vektor-Eingang (`vlen=1024`) verbinden, obwohl beide Ports orange sind! Hierfür gibt es spezielle Blöcke wie *„Stream to Vector“*.

> **Hint**
>
> Solltet ihr euch einmal vertan haben und müsst Verbindungen entfernen, tut dies über die Entf-Taste (Engl. Delete). Auf MacOS ist die Äquivalente Kombination Command+Entf.

Aufgabe 3 — Audio einlesen und visualisieren

In dieser Aufgabe wollen wir eine bereitgestellte WAV-Datei einlesen, per Lautsprecher abspielen und uns parallel das Frequenzspektrum (die Zusammensetzung der Töne) ansehen.

a) *Stelle die Variable* „sample_rate“ *im gezeigten Block per Doppelklick auf 44100 Hz, dem Wert, mit welchem die Audiodatei aufgenommen wurde.*

b) *Suche in der Blockbibliothek (rechte Leiste, nutze die Lupe) nach dem Block* ***Wav File Source*** *und ziehe ihn per Drag & Drop in den Arbeitsbereich.*

c) *Suche nach den Blöcken* ***Audio Sink*** *und* ***QT GUI Frequency Sink*** *und füge auch diese hinzu.*

d) *Öffne die Eigenschaften der* ***Wav File Source*** *mit einem Doppelklick und wähle unter* `File` *die von uns bereitgestellte Audio-Datei aus.*

e) *Verbinde den Ausgang der Quelle mit den Eingängen beider Senken. Klicke dazu erst auf den Ausgangsport, dann auf den Eingangsport.*
Achtung: *Bei der* ***QT GUI Frequency Sink*** *musst du in den Eigenschaften den* `Type` *auf* `Float` *umstellen, da dieser oft standardmäßig auf* `Complex` *(Blau) steht und somit nicht zu unserer orangen Audio-Quelle passt.*

f) *Klicke oben in der Menüleiste auf den Play-Button (oder drücke* `F6`*), um den Flowgraph auszuführen. Du solltest nun die Audio-Datei hören und das Spektrum im Fenster sehen können.*

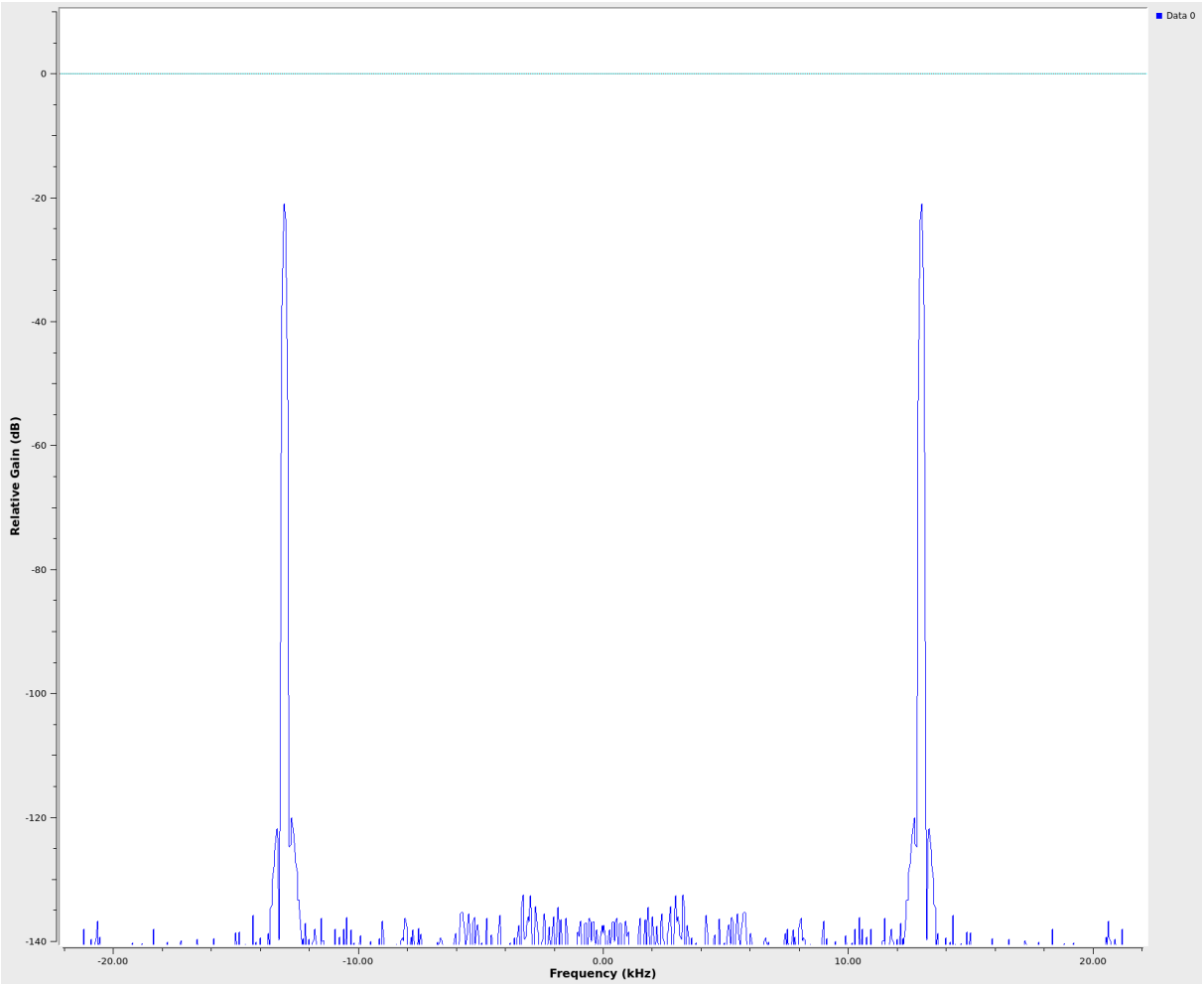

Abbildung 3: Das Live-Spektrum unserer Audio-Datei. Die X-Achse zeigt die Frequenz, die Y-Achse die Signalstärke.

Hint

Der Takt der Schaltung: Audio-Senken (`Audio Sink`) oder angeschlossene SDR-Hardware geben durch ihre physische Abtastrate (Sample Rate) den Takt und damit die Ausführungsgeschwindigkeit des Flowgraphs vor. Hat man keine solche Hardware-gebundene Quelle oder Senke im Flowgraph (z.B. wenn man nur Daten aus einer Datei liest und wieder in eine Datei schreibt), würde der PC die Daten so schnell berechnen, wie die CPU es hergibt. Das lastet den Rechner zu 100% aus und lässt ihn teils einfrieren. In solchen reinen „Software“-Flowgraphs muss zwingend ein **Throttle**-Block eingefügt werden, der künstlich bremst. Da wir hier eine `Audio Sink` nutzen, regelt die Soundkarte den Takt – ein Throttle ist hier also nicht nötig!

3.3. Ein erster Filter

Wahrscheinlich ist dir beim Abspielen der WAV-Datei sofort aufgefallen, dass im Hintergrund ein fieser, sehr hochfrequenter Fiep-Ton zu hören ist. Eine der Kernkompetenzen von GNUradio ist das präzise Filtern von Signalen – genau das werden wir nun tun.

Dazu werfen wir einen genaueren Blick auf unser Frequenzspektrum (das laufende Fenster der `QT GUI Frequency Sink`). Die eigentliche Audio-Information (z. B. Sprache oder Musik) spielt sich meist in den tieferen Frequenzen (bis ca. 3-4 kHz) ab. Der Störton ist im Spektrum deutlich als schmaler, spitzer Ausschlag („Peak“) bei einer wesentlich höheren Frequenz zu erkennen.

Um die nützlichen, tiefen Frequenzen passieren zu lassen und die störenden, hohen Frequenzen abzuschneiden, benötigen wir einen **Low Pass Filter** (Tiefpassfilter).

Aufgabe 4 — Dem Fiep-Ton an den Kragen

Wir bauen nun einen Filter in den Signalfluss ein, um die Störung zu eliminieren.

a) *Füge einen **Low Pass Filter** aus der Bibliothek zu deinem Flowgraph hinzu.*

b) *Lösche die bisherigen direkten Verbindungen zwischen deiner Quelle und den Senken (Klick auf die Linie, dann `Entf`). Schalte den **Low Pass Filter** genau dazwischen, sodass das Signal der Quelle in den Filter fließt, und das gefilterte Signal aus dem Filter heraus in die beiden Senken verteilt wird.*

c) *Stelle in den Eigenschaften des Filters den Datentyp (`FIR Type`) auf `Float->Float (Real Taps)`, damit der Block orange Ein- und Ausgänge bekommt und in unsere Schaltung passt.*

d) *Passe nun die wichtigsten Filter-Parameter an:*
   - *Setze die `Cutoff Freq` (Grenzfrequenz) auf einen Wert, der knapp unterhalb des Störtons im Spektrum, aber über der des Audio-Spektrums liegt. Alles oberhalb dieser Frequenz wird blockiert.*
   - *Setze die `Transition Width` (Übergangsbreite des Filters) auf z. B. `500` Hz. Kein Filter der Welt schneidet perfekt bei exakt 5000 Hz ab. Die Transition Width gibt an, wie „steil“ die Flanke abfällt.*
   - *Achte darauf, dass die `Sample Rate` des Filters der Abtastrate deiner Audio-Quelle entspricht (Standard ist meist die Variable `samp_rate`).*

e) *Führe den Flowgraph erneut aus. Wenn die Grenzfrequenz richtig gewählt ist, sollte der Fiep-Ton nun kaum hörbar sein!*

> ***Hint***
>
> *Es kann bei MacOS nutzern zu Programmabstürzen kommen. Bau in diesem Fall eine throttle nach der WAV File Source ein, um die Datenrate zu limitieren.*

4. Analoge Modulation und Codecs

Nachdem wir im letzten Kapitel gelernt haben, wie wir einfache Signale in GNUradio einlesen, filtern und ausgeben können, machen wir nun den Schritt zu den Kernfunktionen der modernen Nachrichtentechnik. In diesem Kapitel werden wir ein rohes Funksignal analysieren, in seine Bestandteile zerlegen und anschließend das Audio extrahieren.

4.1. Rekapitulation: Modulation und Codecs

Wenn wir Daten – sei es Sprache, Musik oder ein digitales Katzenbild – über ein Medium wie die Luft (Funk) oder ein Kabel übertragen wollen, können wir diese Daten meist nicht in ihrer Rohform senden.

Codecs (Coder/Decoder) Ein Codec ist ein Algorithmus, der einen Datenstrom in ein anderes Format umwandelt. In der digitalen Signalverarbeitung (DSP) nehmen wir ein solches Signal, das zumeist im sogenannten **Basisband** (den niedrigen Frequenzen, z.B. 0 Hz bis 20 kHz für Audio) liegt. Ein Audio-Codec wie WAV oder MP3 digitalisiert die analogen Schallwellen in Nullen und Einsen, ein Video-Codec komprimiert Pixel.

Modulation Die Modulation ist der Prozess, bei dem wir unser Basisband-Signal (unsere eigentliche Information) auf eine wesentlich höhere Frequenz – die sogenannte **Trägerfrequenz** (Carrier) – „aufsatteln“. Warum machen wir das? Eine Antenne, die ein 1 kHz Audio-Signal direkt abstrahlen soll, müsste physikalisch mehrere Dutzend Kilometer lang sein! Durch die Modulation verschieben wir das Signal auf hohe Frequenzen (z.B. 100 MHz beim UKW-Radio), wo Antennen kompakt und praktikabel sind.

In der analogen Welt hat man dafür riesige Schränke voller Spulen, Kondensatoren und Mischer gebraucht. In GNUradio machen wir exakt dasselbe – nur eben rein mathematisch im Prozessor!

4.2. I/Q-Signale

Wenn du dich in GNUradio bewegst, wirst du fast ausschließlich mit den blauen Ports arbeiten: `Complex Float 32`. Doch was bedeutet das?

Das I/Q-Format (In-Phase und Quadrature) ist das absolute Fundament von Software-Defined Radio. Wenn ein analoges Radiosignal von einer Antenne empfangen wird, ist es zunächst nur eine einzige, rasant schwankende Spannung (ein reelles Signal, in GNUradio **orange**). Wenn wir dieses Signal digitalisieren, stoßen wir in der Mathematik auf ein Problem: Reelle Signale haben immer ein symmetrisches Spektrum. Eine Welle bei +10 kHz existiert mathematisch spiegelbildlich auch bei –10 kHz (Spiegelfrequenz). Das macht das digitale Filtern enorm fehleranfällig und ineffizient.

Die brillante Lösung liegt in den komplexen Zahlen. Ein I/Q-Signal besteht für jeden Abtastwert (Sample) nicht nur aus einem, sondern aus **zwei** Zahlenwerten:

- **I (In-Phase):** Der reale Teil.
- **Q (Quadrature):** Der imaginäre Teil (um genau 90 Grad phasenverschoben).

Stell dir einen Zeiger (Vektor) auf einem Ziffernblatt vor, der sich gegen den Uhrzeigersinn dreht.

- Die **Länge** des Zeigers ist die **Amplitude** (Signalstärke).
- Der **Winkel** des Zeigers ist die **Phase**.
- Die **Geschwindigkeit**, mit der er sich dreht, ist die **Frequenz**.

Da wir nun Phase und Amplitude unabhängig voneinander mathematisch kontrollieren können, existieren für den Computer keine ungewollten Spiegelfrequenzen mehr. Ein Signal bei +10 kHz ist nun mathematisch eindeutig von –10 kHz unterscheidbar. GNUradio verarbeitet an jedem blauen Port keine einzelnen Zahlen, sondern immer fortlaufende Koordinatenpaare (I, Q), die diesen rotierenden Vektor beschreiben.

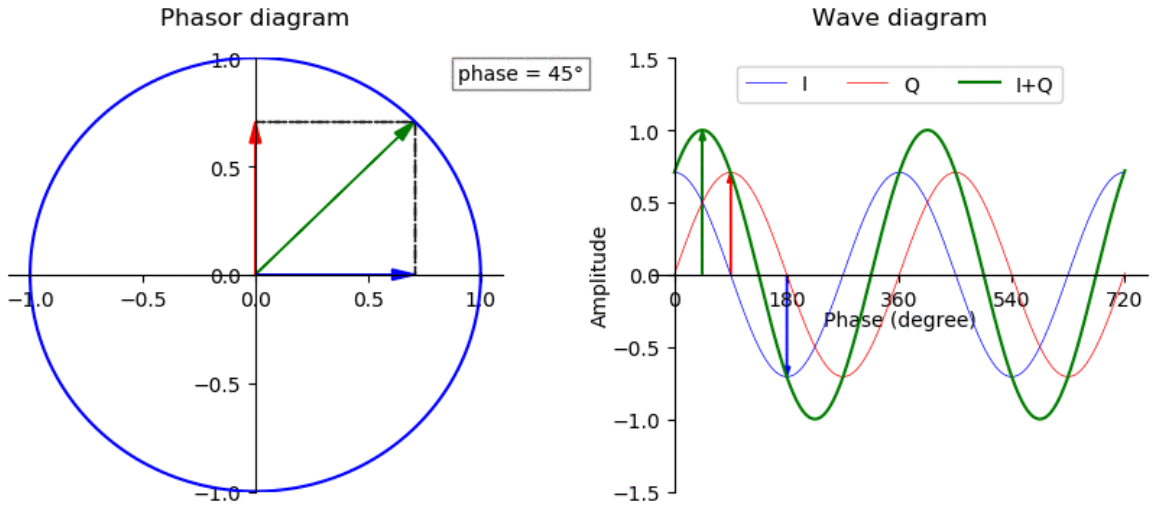

Abbildung 4: Ein I/Q-Sample als komplexer Zeiger. Modulationsarten (AM, FM, PM) verändern letztlich nur die Länge oder den Winkel dieses Vektors. Von Vigneshdm1990 - Eigenes Werk, CC BY-SA 4.0

4.3. Das Frequenzfluss-Diagramm (Wasserfall)

Um einen besseren überblick über die Aktivitäten zu bekommen, welche sich auf den verschiedenen Funkfrequenzen abspielen, nutzen wir ein wichtiges Werkzeug der Signalverarbeitung: Die Wasserfall-Anzeige (Waterfall Sink).

Stell dir das Wasserfalldiagramm wie die Notenrolle eines automatischen Klaviers vor.

- Die **X-Achse** (horizontal) zeigt die Frequenz (Tonhöhe).
- Die **Y-Achse** (vertikal) zeigt die Zeit, die kontinuierlich nach unten wegläuft.
- Die **Farbe** (oder Helligkeit) an einem bestimmten Punkt zeigt die Signalstärke (Amplitude) an. Dunkelblau bedeutet Rauschen („Stille“), leuchtendes Gelb oder Rot bedeutet, dass auf dieser Frequenz gerade ein starkes Signal gesendet wird.

Praktisch gesehen kann man sich die Wasserfall-Anzeige vorstellen als die Aufzeichnung jedes Frequenzdiagrams aus Aufgabe 3 über die Zeit.

4.4. I/Q Dateien einlesen und sichten

Wir haben dir eine aufgezeichnete Datei (`signal_record.iq`) zur Verfügung gestellt. Diese Datei enthält rohe I/Q-Daten, die mit einem echten SDR-Empfänger aufgenommen wurden. Irgendwo in diesem Datensalat versteckt sich ein moduliertes Audio-Signal. Suchen wir es!

Aufgabe 5 — Das Unbekannte Signal sichten

Wir bauen nun einen Flowgraph, um das I/Q-File wie auf dem Bildschirm eines Spektrumanalysators sichtbar zu machen.

a) *Ziehe eine **File Source** in deinen Arbeitsbereich. Wähle das `.iq`-File aus. Setze die `Sample Rate` auf den Wert, mit dem die Datei aufgezeichnet wurde (Hier 40 kHz).*

b) *Sorge dafür, dass dein PC nicht einfriert! Da wir Daten von einer Festplatte lesen, würde GNUradio sie in Millisekunden verarbeiten und die CPU überlasten. Setze zwingend einen **Throttle** Block direkt hinter die File Source (Sample Rate gleichsetzen).*

c) *Füge eine **QT GUI Waterfall Sink** hinzu und verbinde sie mit dem Throttle. Achte darauf, dass alle Ports **Blau** (Complex) sind.*

d) *Starte den Flowgraph. Suche im Wasserfalldiagramm nach einer durchgehenden, hell leuchtenden Spur abseits der Mitte. Diese Spur ist unser gesuchtes Funksignal!*

4.5. Das Signal zentrieren

Das Signal in deiner Datei liegt nicht bei 0 Hz (der Mitte des Spektrums), sondern ist um einen bestimmten Offset verschoben. Um es demodulieren zu können, müssen wir es genau ins Zentrum (ins Basisband) schieben und alle störenden Nebensignale wegschneiden.

GNUradio bietet dafür einen hochoptimierten, extrem mächtigen All-in-One-Block: den **Frequency Xlating FIR Filter**. Dieser Block macht drei Dinge gleichzeitig: Er verschiebt das Signal, er filtert es und er reduziert auf Wunsch die Sample-Rate (Decimation), um die CPU zu entlasten.

Damit wir den Offset nicht mühsam raten müssen, bauen wir uns nun eine grafische Benutzeroberfläche (GUI) mit einem Schieberegler und einem Textfeld!

Aufgabe 6 — Den Xlating FIR Filter interaktiv einrichten

a) *Füge den Block **QT GUI Range** hinzu. Dieser erstellt einen Schieberegler. Nenne die ID `freq_offset`. Setze den Default Value auf `0`, das Minimum auf `-500000` und das Maximum auf `500000` (entspricht ± 500 kHz). Dem Label kannst du einen beliebigen Namen wie bspw. „Frequenz-Offset“.*

b) *Trenne die Verbindung zur Waterfall Sink. Füge den **Frequency Xlating FIR Filter** direkt hinter den Throttle ein.*

c) *Stelle den Filter ein: Trage bei `Center Frequency` deine Variable `freq_offset` ein. Unter `Taps` definieren wir on-the-fly den Filter: `firdes.low_pass(1, samp_rate, 5000, 1000)`. Setze `Decimation` auf `1`, da unser input mit 40 kHz bereits eine gute sample rate besitzt, um sie direkt weiterzuverarbeiten.*

d) *Hänge deine Waterfall Sink nun **hinter** den Filter. (Tipp: Vielleicht ist es hilfreich, eine zweite Waterfall-Sink wie in Aufgabe 5 direkt an den I/Q-Stream aus der Datei zu setzen, um einfacher kreuzreferenzieren zu können).*

e) *Starte den Flowgraph. Schiebe den Regler langsam hin und her, bis die leuchtende Signalspur exakt bei `0 Hz` (in der Mitte) liegt. Lies den Wert deines Schiebereglers ab und trage ihn in hier ein:*

_____

Glückwunsch, du hast das Signal gefangen!

4.6. AM-Demodulation: Komplexe Daten reell machen

Unser Signal ist nun perfekt bei 0 Hz zentriert. Es handelt sich um ein Amplituden-Moduliertes (AM) Signal. Die Audio-Information – also die Sprache – steckt hier ausschließlich in der **Schwankung der Signalstärke**.

Wenn wir an unseren I/Q-Zeiger zurückdenken: Wir benötigen nur noch die **Länge** dieses rotierenden Vektors. Der Winkel (die Phase) ist uns völlig egal. Wir müssen also die komplexen (blauen) Daten in reelle (orange) Daten umwandeln. GNUradio nutzt dafür den Block **Complex to Mag** (Magnitude / Betrag). Dieser misst zu jedem Zeitpunkt schlichtweg die Länge des Zeigers und wirft diesen Wert als normales Audiosignal aus.

Aufgabe 7 — Audio hörbar machen

Wir extrahieren nun das Audio und leiten es an die Lautsprecher weiter.

a) *Füge hinter deinem Xlating Filter den Block **Complex to Mag** ein. Achte auf den Farbwechsel: Blau geht rein, Orange kommt raus! Wir sind zurück in der reellen Audiowelt.*

b) *Ein AM-Signal schwankt oft um einen sehr hohen Gleichanteil (DC-Offset). Würden wir das in den Lautsprecher schicken, würde die Membran extrem auslenken und blockieren. Füge einen **DC Blocker** (Länge z.B. 1024) hinzu, um diese Gleichspannung zu entfernen.*

c) *Die PC-Soundkarte erwartet exakt 44.100 Hz. Unser Signal hat aber aktuell `samp_rate` (also z.B. 40.000 Hz). Füge einen **Rational Resampler** hinzu (Interpolation: `44100`, Decimation: z z.B. `40000`).*

d) *Schalte einen **Multiply Const** Block (Typ Float) dazwischen, um die Lautstärke zu regeln (Constant = z.B. `20`), mit einem Wert, der für euch sinn macht.*

e) *Füge eine **Audio Sink** (44,1 kHz) am Ende hinzu. Starte den Flowgraph. Du solltest nun die Musik hören!*

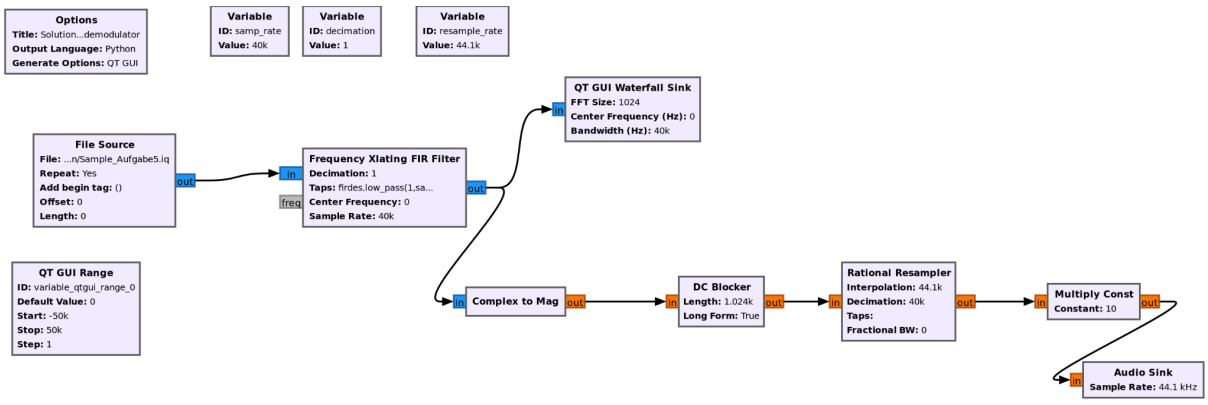

Abbildung 5: Der fertige AM-Demodulator unter Verwendung der hochintegrierten GNUradio-Blöcke.

4.7. Exkurs: Wie funktioniert Amplitudenmodulation und Mischen eigentlich?

Bevor wir den magischen `Xlating FIR Filter` in seine Einzelteile zerlegen, müssen wir verstehen, was mathematisch bei der Modulation eigentlich passiert.

Die Antwort ist verblüffend simpel: **Es ist nur eine Multiplikation!** Wenn du ein niederfrequentes Signal (deine Stimme) nimmst und es mit einem hochfrequenten Signal (einem reinen Sinus-Träger) multiplizierst, passiert die Magie der Amplitudenmodulation. Das hochfrequente Trägersignal wird genau im Takt deiner Stimme lauter und leiser.

Genau dasselbe Prinzip gilt rückwärts für den Frequenz-Offset, den wir gerade mit dem Schieberegler korrigiert haben. Um ein Funksignal im Frequenzspektrum zu verschieben, müssen wir es in der Hardware (oder in Software) einfach mit einem sogenannten **Lokalen Oszillator (LO)** multiplizieren. Ein LO ist nichts anderes als ein reiner Sinus-Ton-Generator. Die Multiplikation zweier Frequenzen führt physikalisch dazu, dass sich die Frequenzen summieren und subtrahieren. Liegt unser Signal bei +150 kHz, multiplizieren wir es mit einem Oszillator, der auf –150 kHz läuft. Das Resultat: Das Signal rutscht exakt auf 0 Hz!

Um dieses Prinzip der Multiplikation visuell greifbar zu machen, betrachten wir das folgende interaktive Schaubild:

4.8. Demodulation „Von Hand“: Den Mischer selbst bauen

Nachdem wir nun wissen, dass die Frequenzverschiebung (das Zentrieren des Signals) in Wahrheit nur eine Multiplikation mit einem Lokalen Oszillator ist, werfen wir den „bequemen“ Xlating FIR Filter aus unserem Programm und bauen die Signalverarbeitungskette von Grund auf selbst nach!

Das erfordert drei mathematische Grundbausteine:

Einen reinen Sinus-Generator (Local Oscillator).

Einen Multiplizierer (Mischer).

Einen Tiefpass-Filter (Low Pass), der das Rauschen links und rechts unseres zentrierten Signals abschneidet.

Aufgabe 8 — Den Mischer von Hand bauen

Wir ersetzen den All-in-One-Block durch elementare Mathematik-Blöcke.

a) *Lösche den **Frequency Xlating FIR Filter** aus deinem Flowgraph.*

b) *Füge eine **Signal Source** hinzu. Das ist unser Oszillator! Wähle als Typ `Complex`. Trage bei der Frequenz `- freq_offset` ein. Wir multiplizieren mit der **negativen** Frequenz, um das Signal zurück auf die 0-Linie zu ziehen.*

c) *Füge einen **Multiply** Block (Typ Complex) hinzu. Verbinde den Ausgang deines Throttles mit dem ersten Eingang des Multiply-Blocks, und den Ausgang der Signal Source mit dem zweiten Eingang. Du mischst gerade Hochfrequenz!*

d) *Jetzt ist das Signal zwar bei 0 Hz, aber es gibt noch sehr viel Rauschen links und rechts davon. Füge hinter dem Multiply-Block einen **Low Pass Filter** ein. Stelle ihn ein: `FIR Type: Complex->Complex`, `Cutoff Freq: 5000` Hz, `Transition Width: 1000` Hz.*

e) *Verbinde den Low Pass Filter wieder mit deinem `Complex to Mag` Converter und der Waterfall Sink. Starte den Graphen. Das Ergebnis sollte identisch zu vorher klingen – aber diesmal hast du den Mischer von Hand gebaut!*

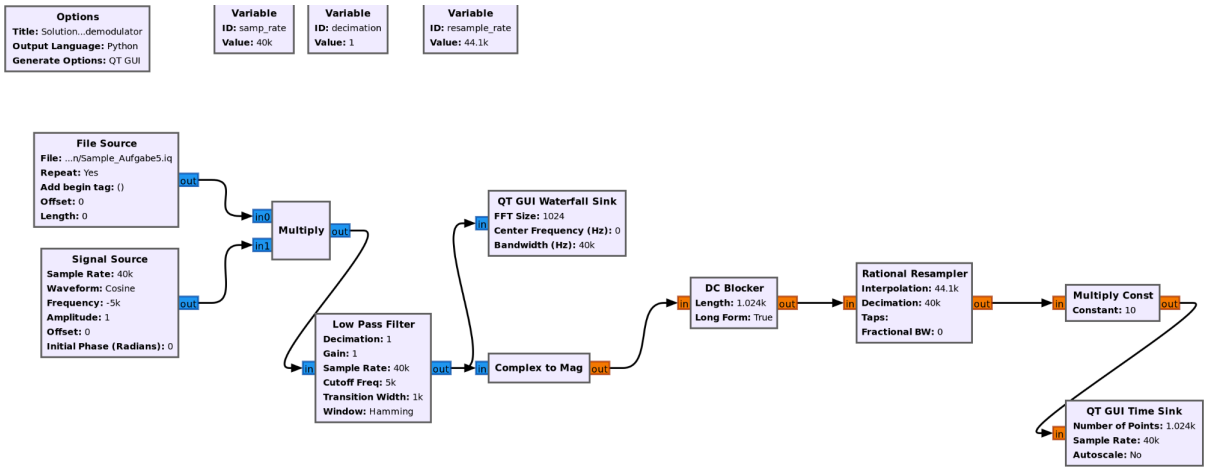

Abbildung 6: Die fundamentalste Schaltung der Funktechnik: Ein Signal wird mathematisch mit einem Oszillator multipliziert, um es auf eine andere Frequenz zu verschieben (Mischen).

Hint

Hier wird jetzt klar, warum eine darstellung mittels I/Q-Samples für die Demodulation wichtig ist. Würden wir mit rein reelen Samples arbeiten wäre eine Multiplikation mit dem negativen `freq_offset` äquivalent zu einer Multiplikation mit der positiven variante. Dementsprechend hätten wir nicht nur den gewünschten Effekt, unser Signal auf 0 Hz herunter zu mischen, sondern hätten gleichzeitig das Signal bei $2 * \text{freq\_offset}$ hochgemischt und müssten dies nun erst rausfiltern - wie dies bei real gebauten Filtern oft der Fall ist.

4.9. Bonusaufgabe: Eigene Signale FM-modulieren und austauschen

Bisher haben wir nur AM-Signale empfangen und decodiert. Aber was wäre der Amateurfunk ohne das Senden? In dieser Abschlussaufgabe wendet ihr das Blatt: Ihr nehmt euch eigene Audiodateien, verwandelt sie in astreine Frequenz-Modulierte (FM) I/Q-Funksignale, tauscht die Dateien im Raum aus und decodiert die Signale eurer Nachbarn!

Bei der Frequenzmodulation (wie beim UKW-Radio) schwankt nicht die Stärke des Signals, sondern wir wackeln an der Frequenz des Trägers.

Aufgabe 9 — Werde zum eigenen FM-Radiosender

Wir bauen zwei separate Flowgraphs: Einen zum Senden (Speichern in I/Q) und einen zum Empfangen.

Der Sender (TX):

a) *Erstelle einen leeren Flowgraph. Nutze eine **Wav File Source**, um eine Musik- oder Sprachdatei (WAV) einzulesen. (Kein Throttle nötig!).*

b) *Für ein qualitativ gutes FM-Signal brauchen wir im I/Q-Bereich eine höhere Bandbreite. Füge einen **Rational Resampler** ein und sample das Audiosignal auf z.B. 500 kHz hoch (Interpolation: `500000`, Decimation: Audio-Samplerate).*

c) *Füge den genialen Block **WBFM Transmit** hinzu. Vorne geht oranges Audio rein, hinten kommt ein blaues, moduliertes FM-I/Q-Signal heraus! Setze die Audio Rate auf `500000`.*

d) *Speichere das blaue Signal mit einer **File Sink** ab (z.B. als `mein_radio.iq`). Lass den Graphen kurz laufen (Achtung, I/Q-Dateien werden schnell sehr groß!).*

Der Empfänger (RX):

e) *Tausche deine `.iq`-Datei mit der eines anderen Teilnehmenden aus.*

f) *Erstelle einen neuen Flowgraph. Nutze **File Source** (Lade das fremde I/Q-File) → **Throttle** (auf 500 kHz) → **WBFM Receive** → **Rational Resampler** (zurück auf 48 kHz Soundkarten-Takt) → **Audio Sink**.*

g) *Spiele den Graphen ab. Wenn du alles richtig gemacht hast, hörst du nun kristallklar das FM-modulierte Audio deiner Kommilitonen!*