# GNU-Radio
## mit GNU-Radio Companion

17. August 2019

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

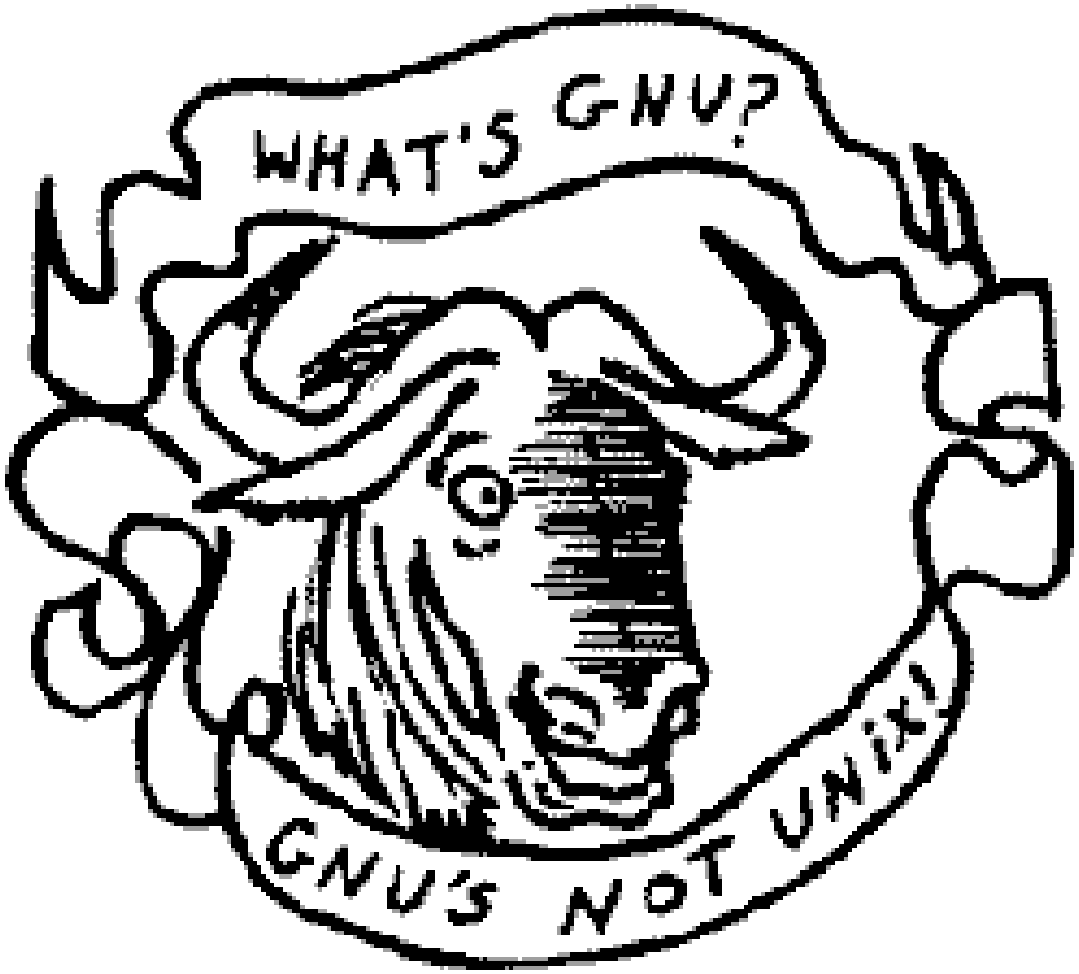
Technik & Architektur
Institut für Elektrotechnik

## 90 Jahre
## USKA Jubiläums-Jahrestreffen der Schweizer Funkamateure (Hamfest)

Martin Klaper, HB9ARK

V 1.4

# Was ist "GNU" ?



# Lernziele

- **Verstehen, was der "GNU Radio Companion" (GRC) ist.**
- **Verstehen, wie der GRC grundsätzlich funktioniert.**
- GNU Radio für Windows installieren können.
- Die Grundfunktionen der GRC Bedienoberfläche bedienen können.
- Vorgefertigte Funktionsblöcke auswählen, einsetzen und konfigurieren können.
- Verbindung von GRC mit Hardware Peripherie erstellen können.
- **Einfache Beispiele selber erstellen können.**
- Komplexere Beispiele modifizieren können.

# Inhalt

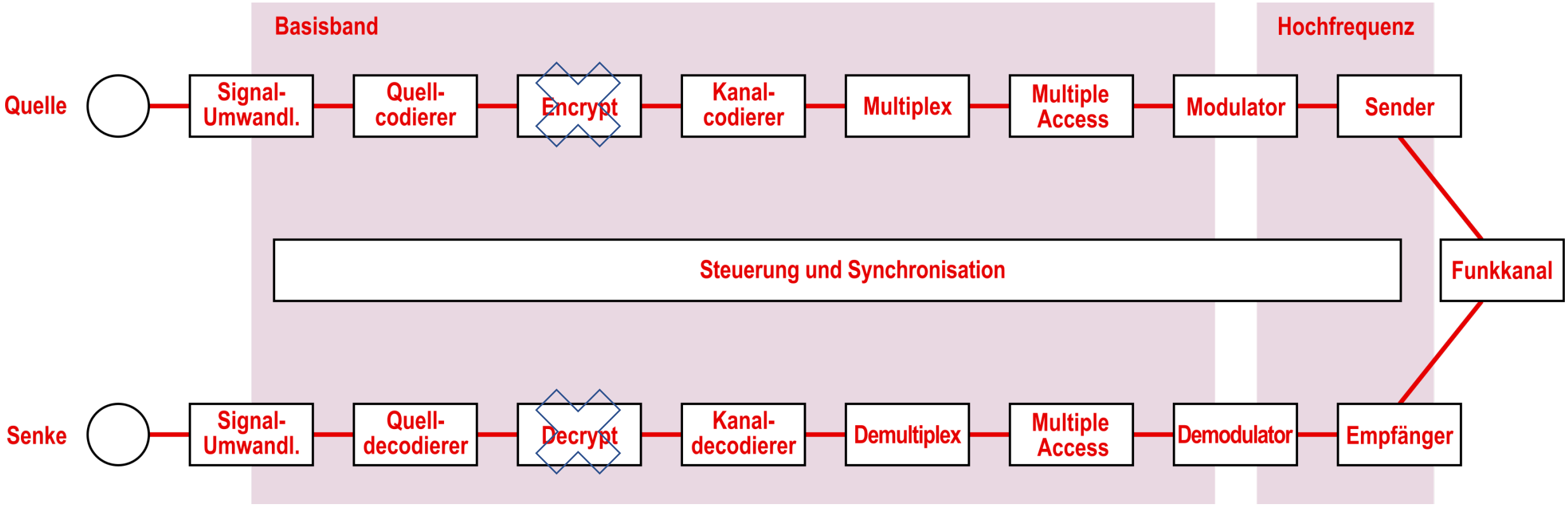
**Theorie**

- Begriffsdefinitionen
- Kennenlernen der Oberfläche
- Grafische Elemente
- Funktionsblöcke
- Quellen
- Senken

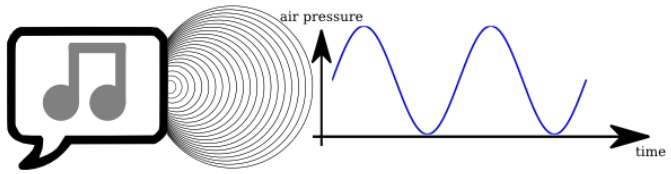
**Praxis: Vorführungen**

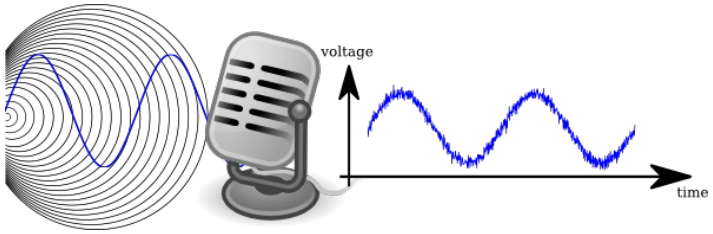
- Kennenlernen der Oberfläche
- Quellen und Senken
- Funktionsblöcke
- Grafische Benutzerelemente
- Signale, Visualisierung, Ausgabe
- Empfänger
- Sender

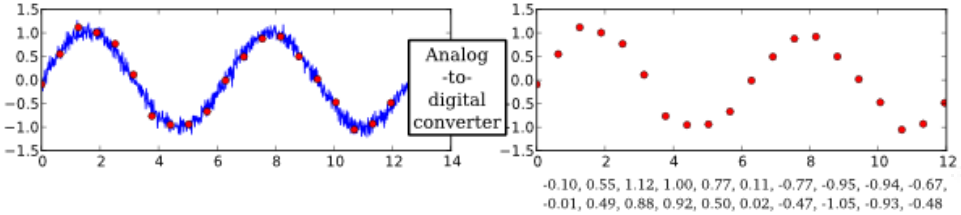
# Struktur eines Kommunikationssystems



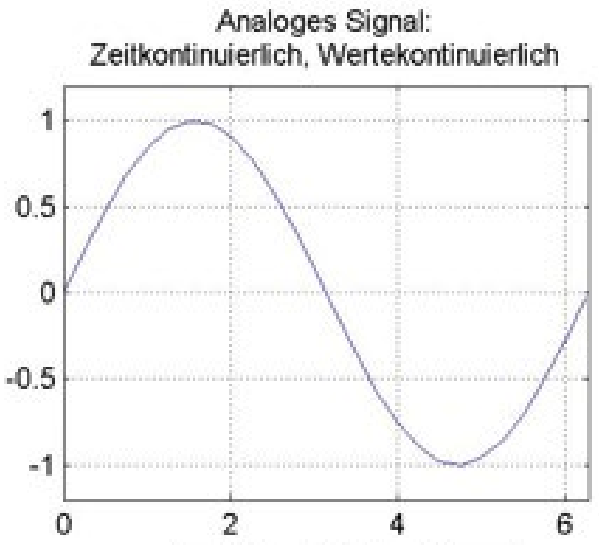
# Abtasten und Digitalisieren

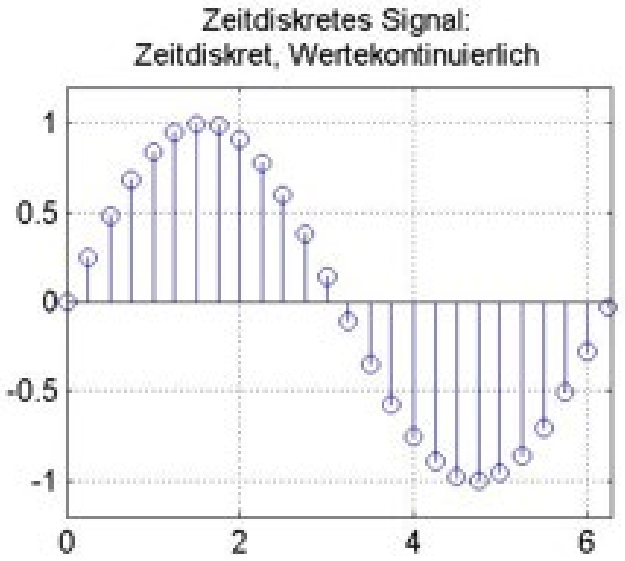


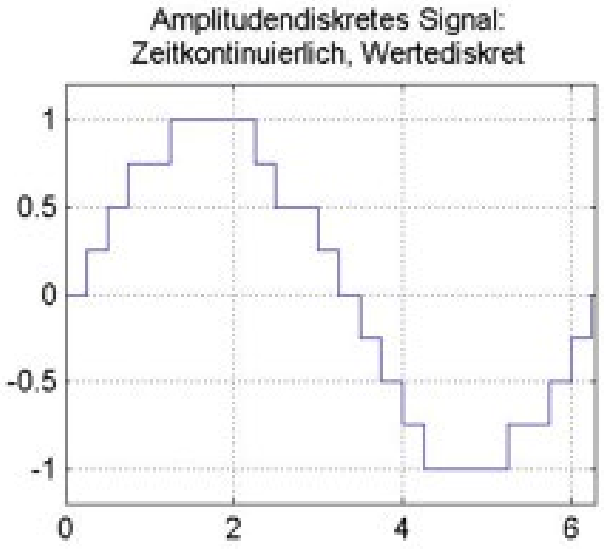



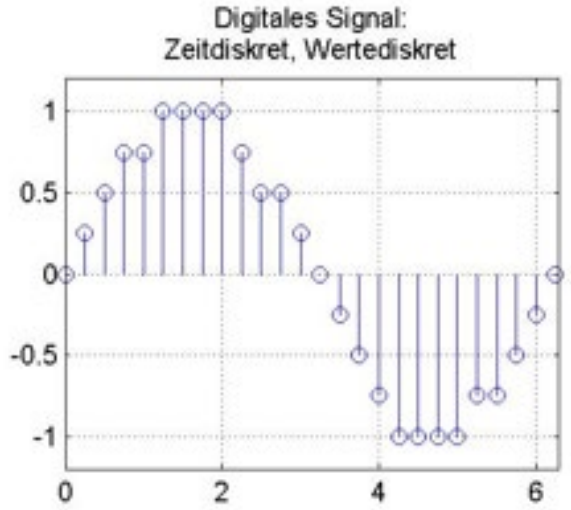




$$f_{Signal\_max} < 2 \cdot f_{Abtastung}$$



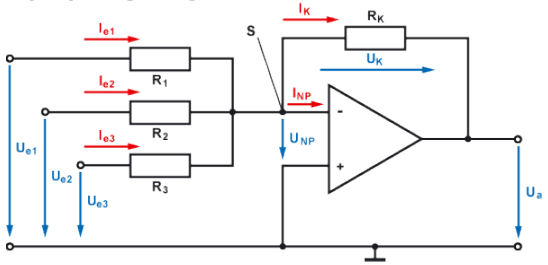
# Warum digital?

- Preisgünstige Hardware (FPGA, µC)
- Vernetzung neuer Dienste erfolgt digital (Email, Mobiltelefonie, MP3,…)
- Dienstqualität und Fehlerschutz (Prüfsummen)
- Kompatibilität und Flexibilität (Codemultiplex, programmierbare Logik)
- Kosten für die Übertragung (Spektrale Effizienz [bit/Hz], Energie Effizienz [Ws/bit], Komplexität)

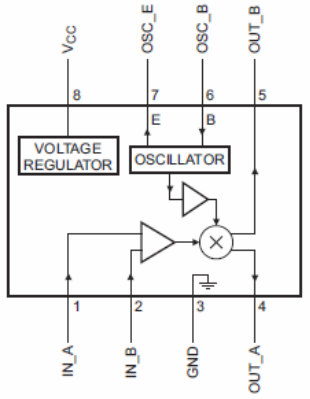
# Vorgänge sind mathematisch beschrieben, dann physikalisch oder rechnerisch realisiert

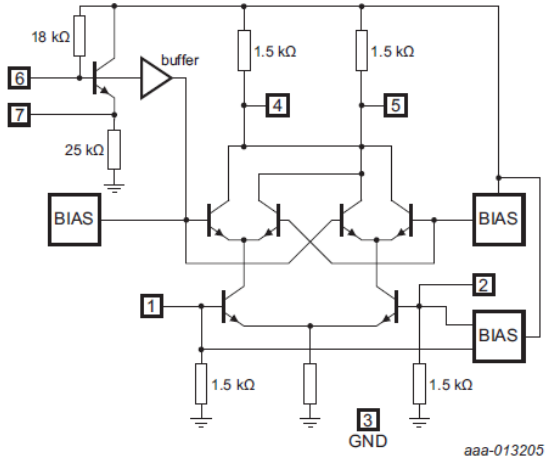
## analog

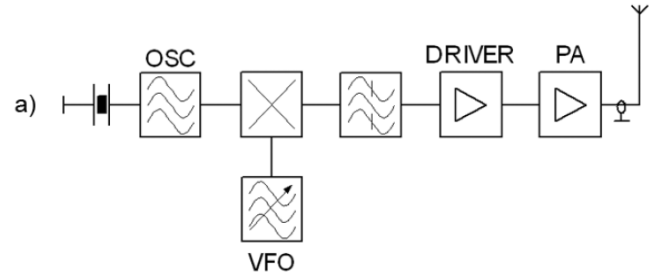
- Addieren



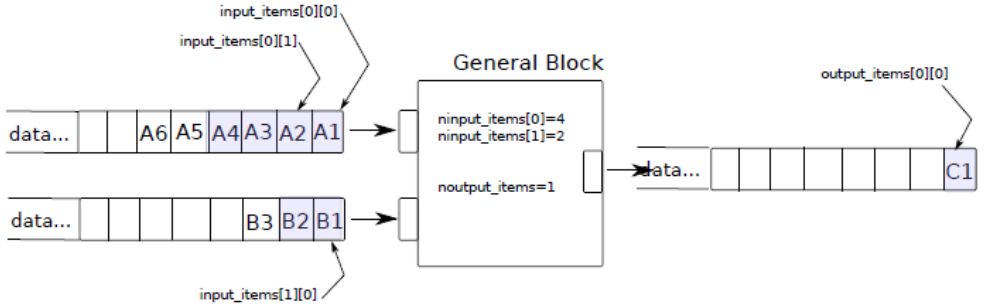

- Multiplizieren (SA612A)




## digital




- Addieren

```
for (size_t i=1;i<input_items.size();i++)
    volk_32f_x2_add_32f(out, out, (const
                float*)input_items[i], noi);
```

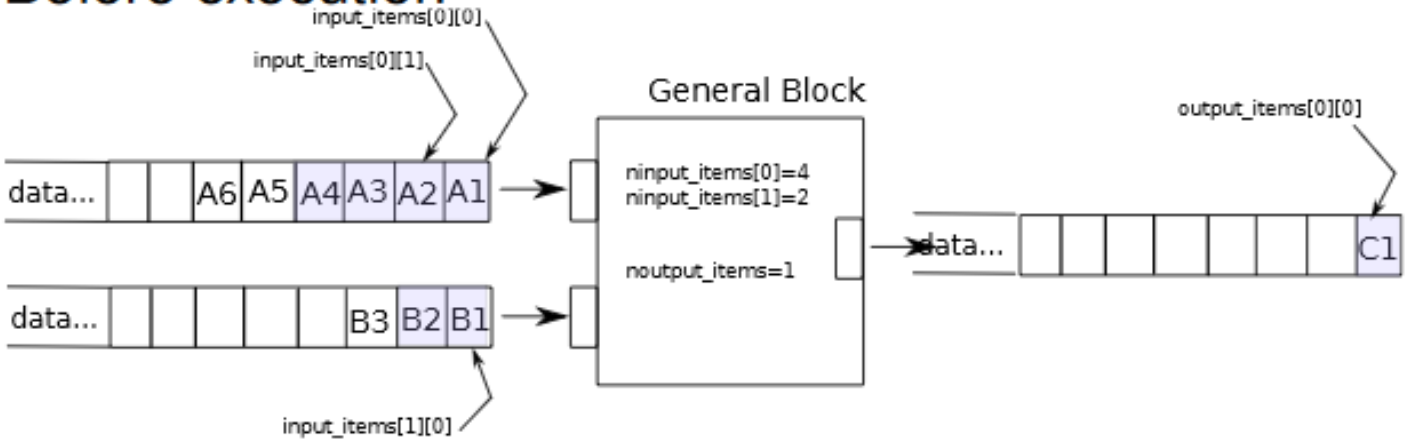
VOLK = Vector-Optimized Library of Kernels

- Multiplizieren

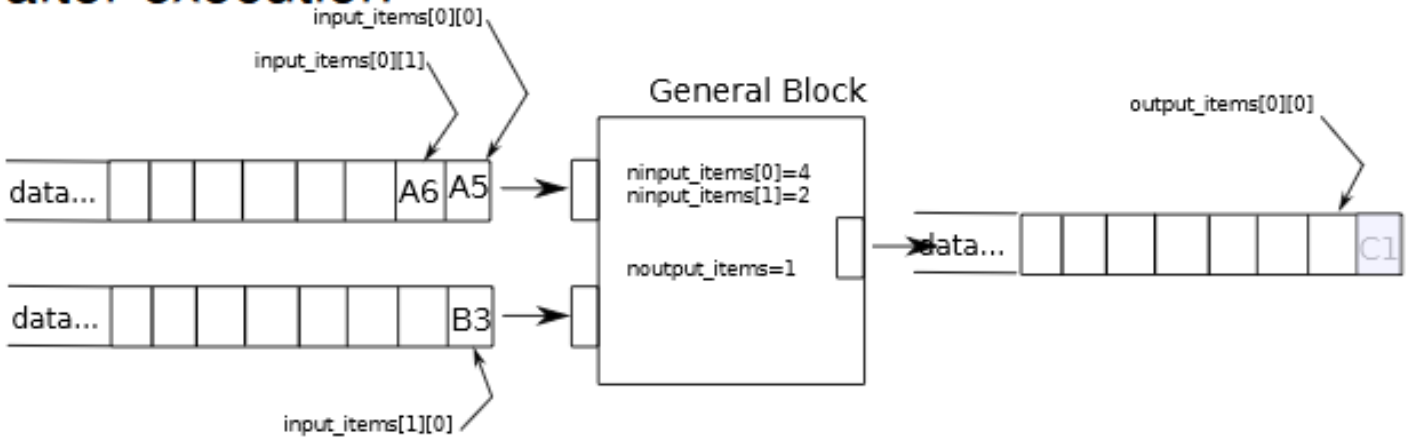
```
for (size_t i=1; i<input_items.size();i++)
    volk_32f_x2_multiply_32f(out, out,
        (float*)input_items[i], noi);
```

# Verarbeitungsschritt in GNU Radio

- Before execution




- after execution



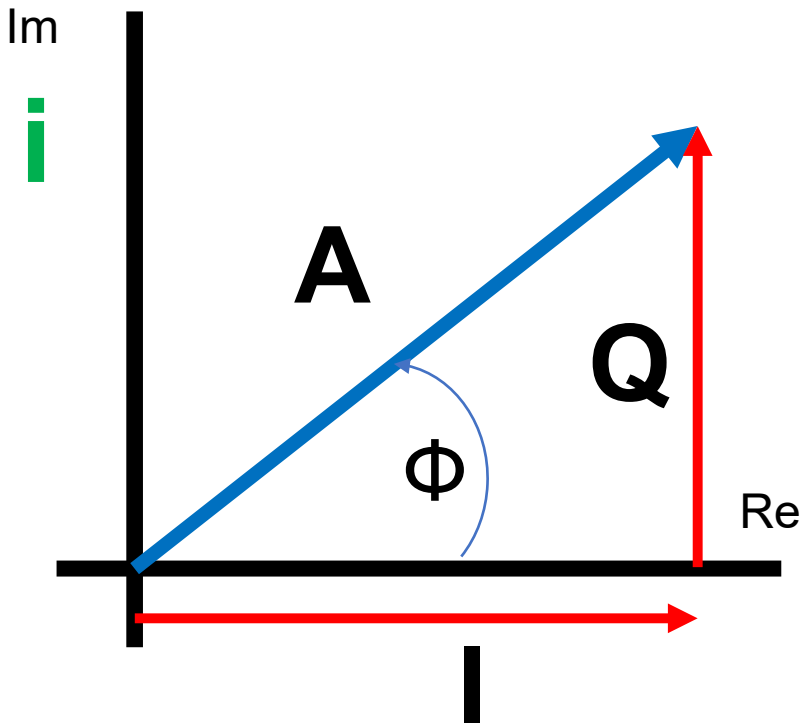
# Moduliertes Trägersignal "cisΦ"

$$s(t) = A(t) \cdot \cos(\omega \cdot t + \Phi(t))$$

Signal — Amplitude — Frequenz — Phase

$$\omega = 2 \cdot \pi \cdot f$$

Frequenz




Nomenklatur:
Q = Quadratur Komponente
I = In-Phase Komponente

$$Q = A \cdot \sin(\Phi)$$

$$I = A \cdot \cos(\Phi)$$

- Zur Modulation eines Sinus-Signales stehen drei Variablen zur Verfügung: Amplitude (AM), Momentanfrequenz (FM) und Phase (PM)
- Durch Umrechnen der polaren- in die kartesischen Koordinaten können die I/Q-Daten berechnet werden.
- I/Q Daten repräsentieren ein Signal **vollständig** (Amplitude und Phase) im Zeitpunkt der Abtastung, sofern $f_{sample} > 2 * f_{max}$
- Mit I/Q Daten können beliebige Bandsegmente aufgezeichnet werden und zeitverschoben analysiert und decodiert werden.
- Limiten sind Abtastfrequenz, Auflösung des A/D Wandlers und Speicherplatz.

# Was ist der GRC?

- **Grafische Benutzeroberfläche zur Entwicklung von Signalverarbeitungsanwendungen**

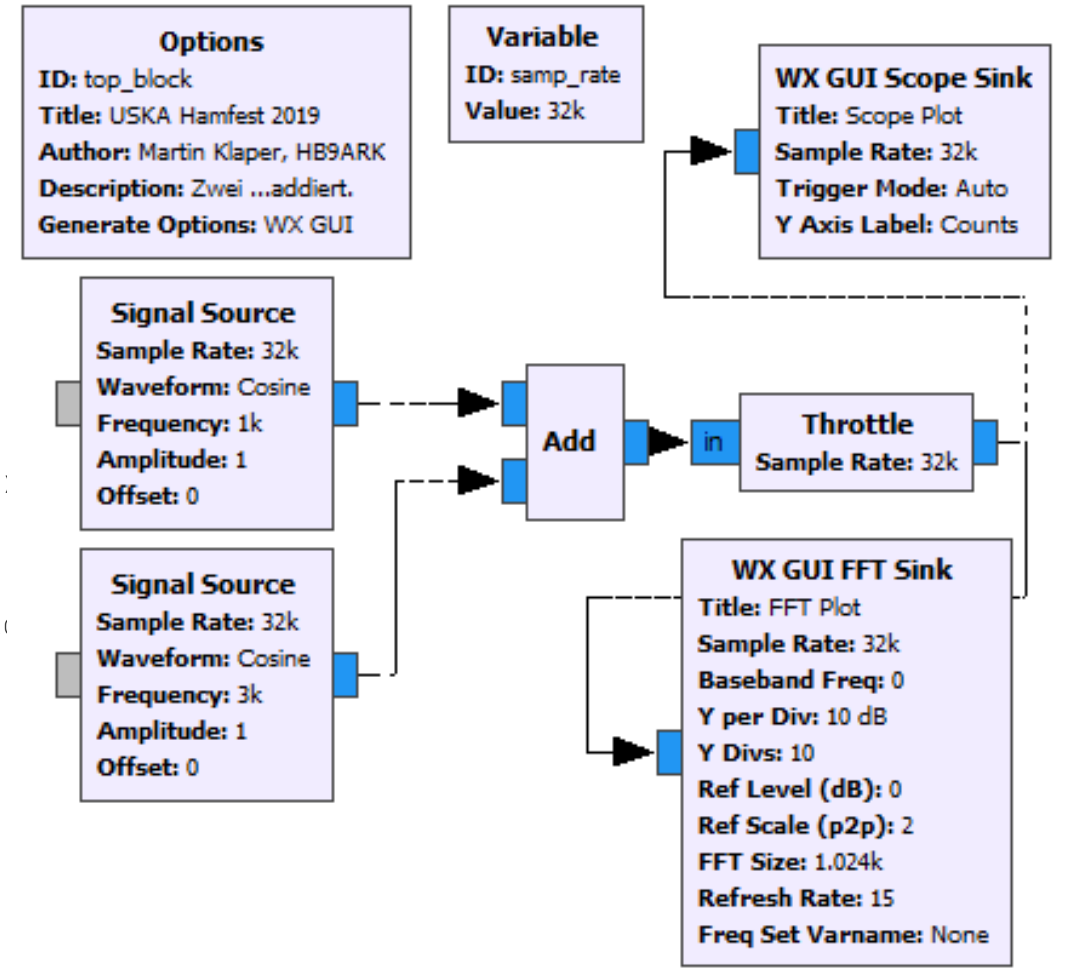
- Einfacher Einstieg für "Nichtprogrammierer".

- Funktionsblöcke werden mit der Maus auf die Arbeitsfläche gezogen und verbunden.

- Auf Knopfdruck wird ein Python-Skript erstellt und gestartet.

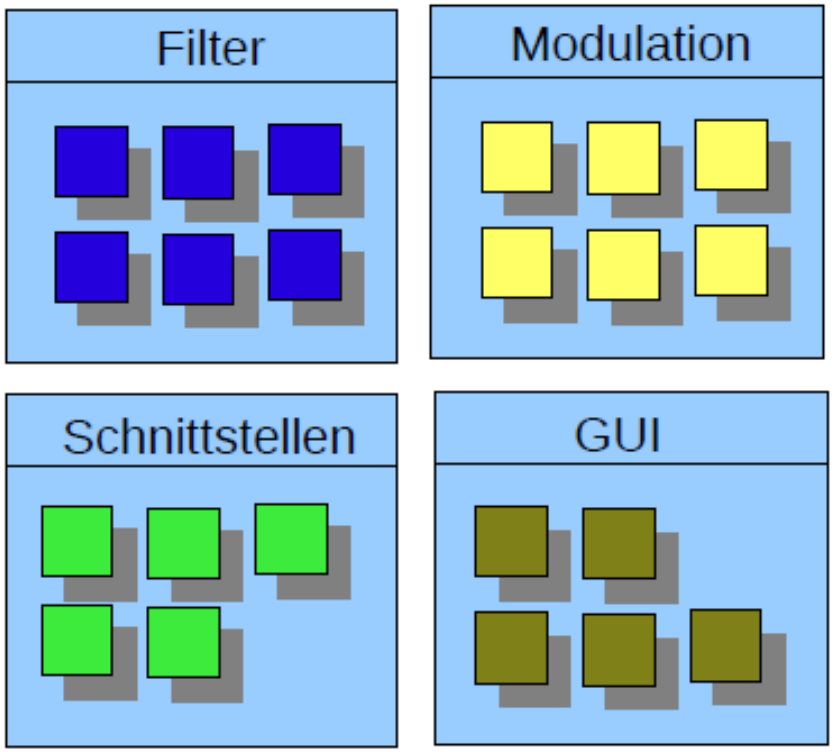
# Python-Skript automatisch generiertes aus Blockschaltbild

```
##################################################
# GNU Radio Python Flow Graph USKA Hamfest 2019
# Title: Top Block Martin Klaper, HB9ARK
# Generated: Fri Jul 12 14:04:27 2019
##################################################
from gnuradio import analog
from gnuradio import blocks
from gnuradio import wxgui
from gnuradio.wxgui import fftsink2
from gnuradio.wxgui import scopesink2

class top_block(grc_wxgui.top_block_gui):
 def __init__(self):

self.wxgui_scopesink2_0 = scopesink2.scope_sink_c(title='Scope Plot',)
        ##################################################
        # Blocks
        ##################################################
        self.Add(self.wxgui_scopesink2_0.win)self.wxgui_fftsink2_0 = fftsink2.fft_sink_c(
        self.Add(self.wxgui_fftsink2_0.win)
        self.blocks_throttle_0 = blocks.throttle(gr.sizeof_gr_complex*1, samp_rate,True)
        self.blocks_add_xx_0 = blocks.add_vcc(1)
        self.analog_sig_source_x_1_0 = analog.sig_source_c(samp_rate, analog.GR_COS_WAVE,
        self.analog_sig_source_x_1 = analog.sig_source_c(samp_rate, analog.GR_COS_WAVE, 1(
        ##################################################
        # Connections
        ##################################################
        self.connect((self.analog_sig_source_x_1, 0), (self.blocks_add_xx_0, 0))
        self.connect((self.analog_sig_source_x_1_0, 0), (self.blocks_add_xx_0, 1))
        self.connect((self.blocks_add_xx_0, 0), (self.blocks_throttle_0, 0))
        self.connect((self.blocks_throttle_0, 0), (self.wxgui_fftsink2_0, 0))
        self.connect((self.blocks_throttle_0, 0), (self.wxgui_scopesink2_0, 0))

def main(top_block_cls=top_block, options=None):
    tb = top_block_cls()
    tb.Start(True)
    tb.Wait()
```



# Baukasten zum Zusammensetzen von Systemen

## Bibliothek mit vielen nützlichen Funktionen




## Ausführungsumgebung zur Laufzeit

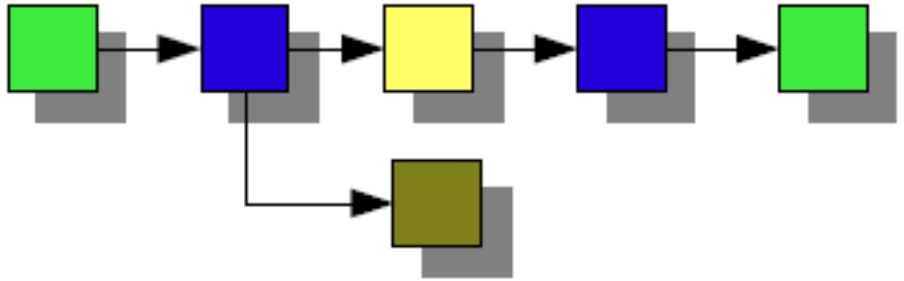


Bild nach Gerrit Buhe, DL9GFA

# Die Programmierung ist auf mehreren Ebenen möglich

- Wer mit den bereits vorhandenen, umfangreichen Funktionsblöcken auskommt, kann ausschliesslich grafisch arbeiten mit "GRC", was wir hier tun.

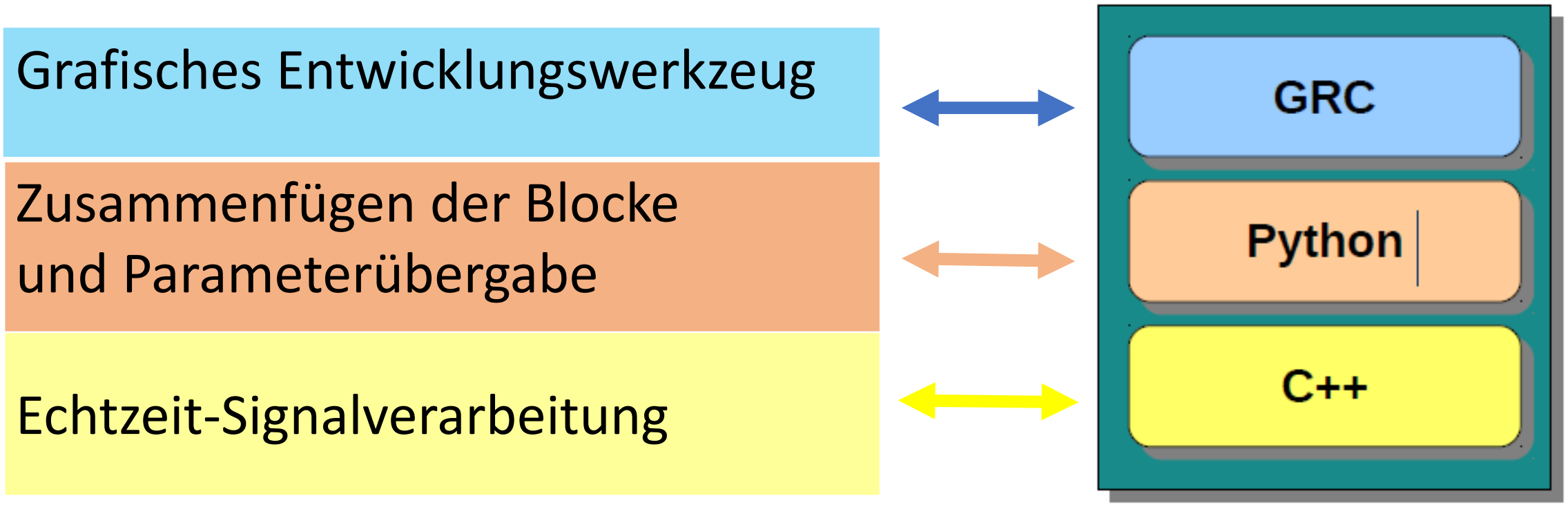


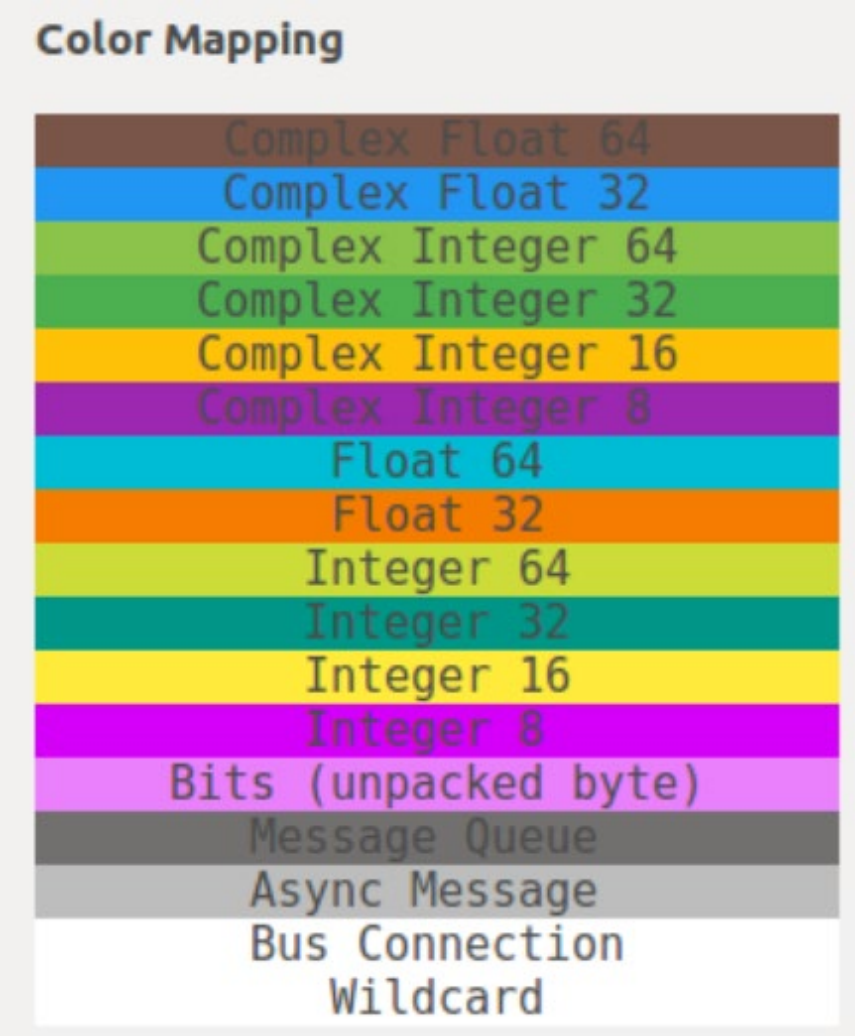

Bild frei nach Gerrit Buhe, DL9GFA

# GNU Radio Companion Demo

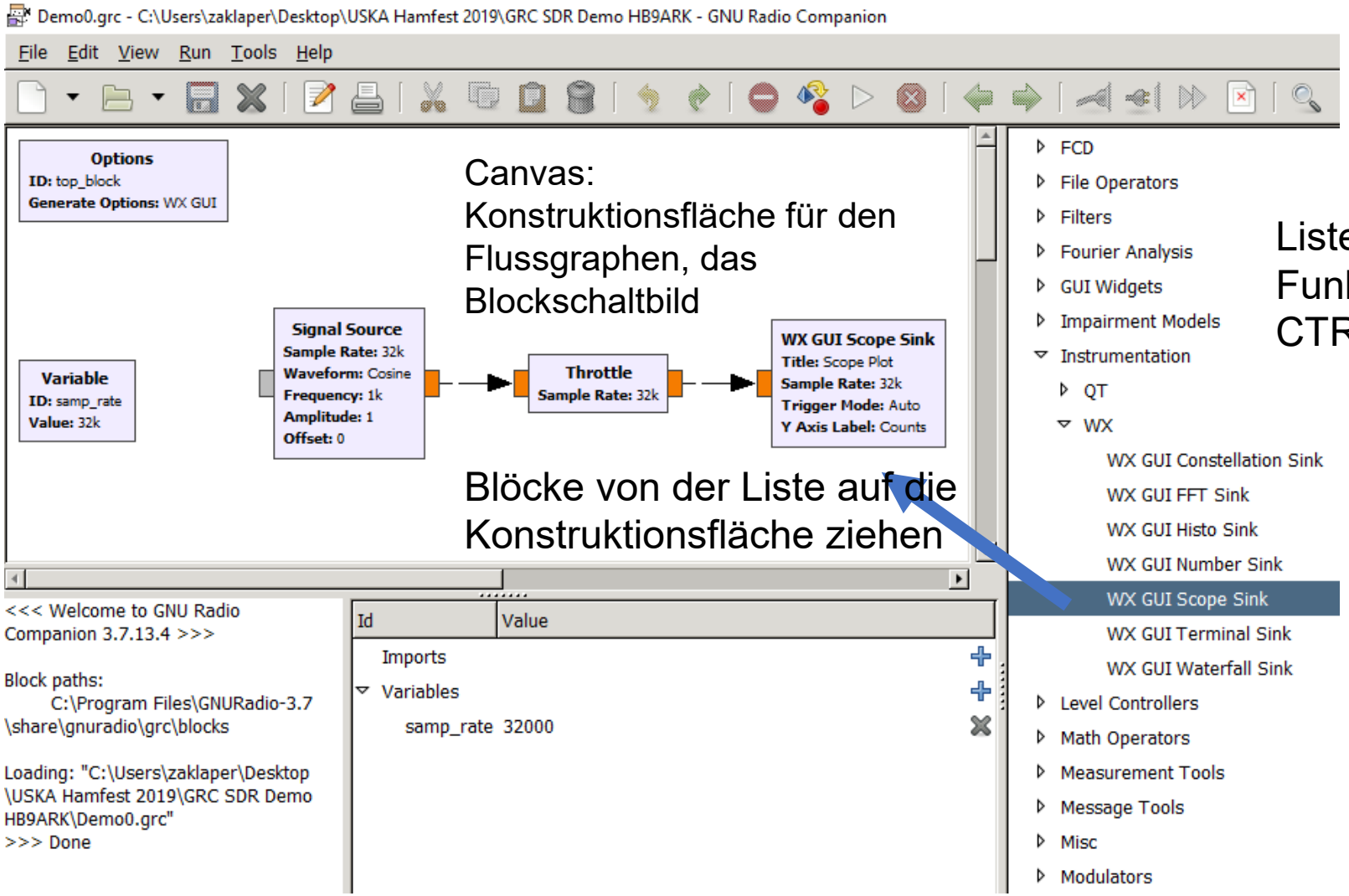
# Demo (Entwicklung im Vortrag)

- Polar- bzw. Kartesische Koordinaten
- Komplexes Signal
- Typen Kompatibilität (gleiche Farben!)

- Addition von zwei Signalen
- Multiplikation von zwei Signalen
- Darstellung in Abhängigkeit der Zeit
- Darstellung als Spektrum, also in Abhängigkeit der Frequenz



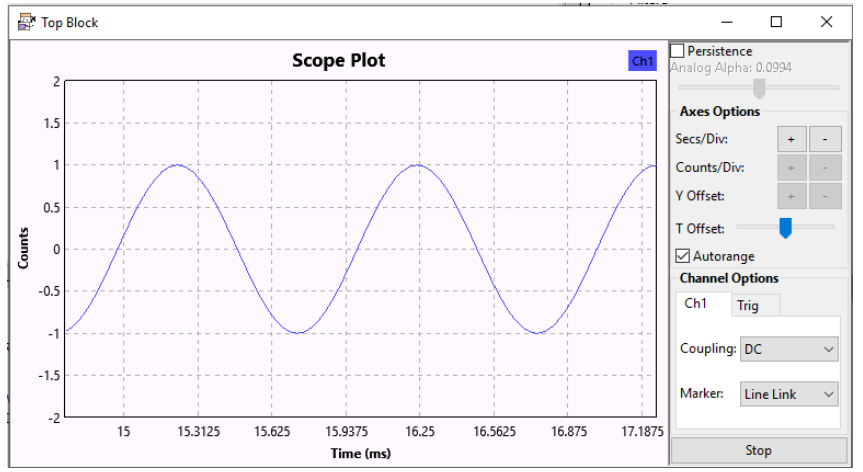
# GNU Radio Companion – Dein Freund & Helfer




Liste aller verfügbaren Funktionsblöcke
CTRL+F zum Suchen

Ports (Lötfahnen) durch Anklicken verbinden. Ports können selektiert und wieder gelöscht warden.

# Sinussignal erzeugen und Zeitverlauf anzeigen (Demo0)




**Options**
**ID:** top_block
**Generate Options:** WX GUI

**Variable**
**ID:** samp_rate
**Value:** 32k

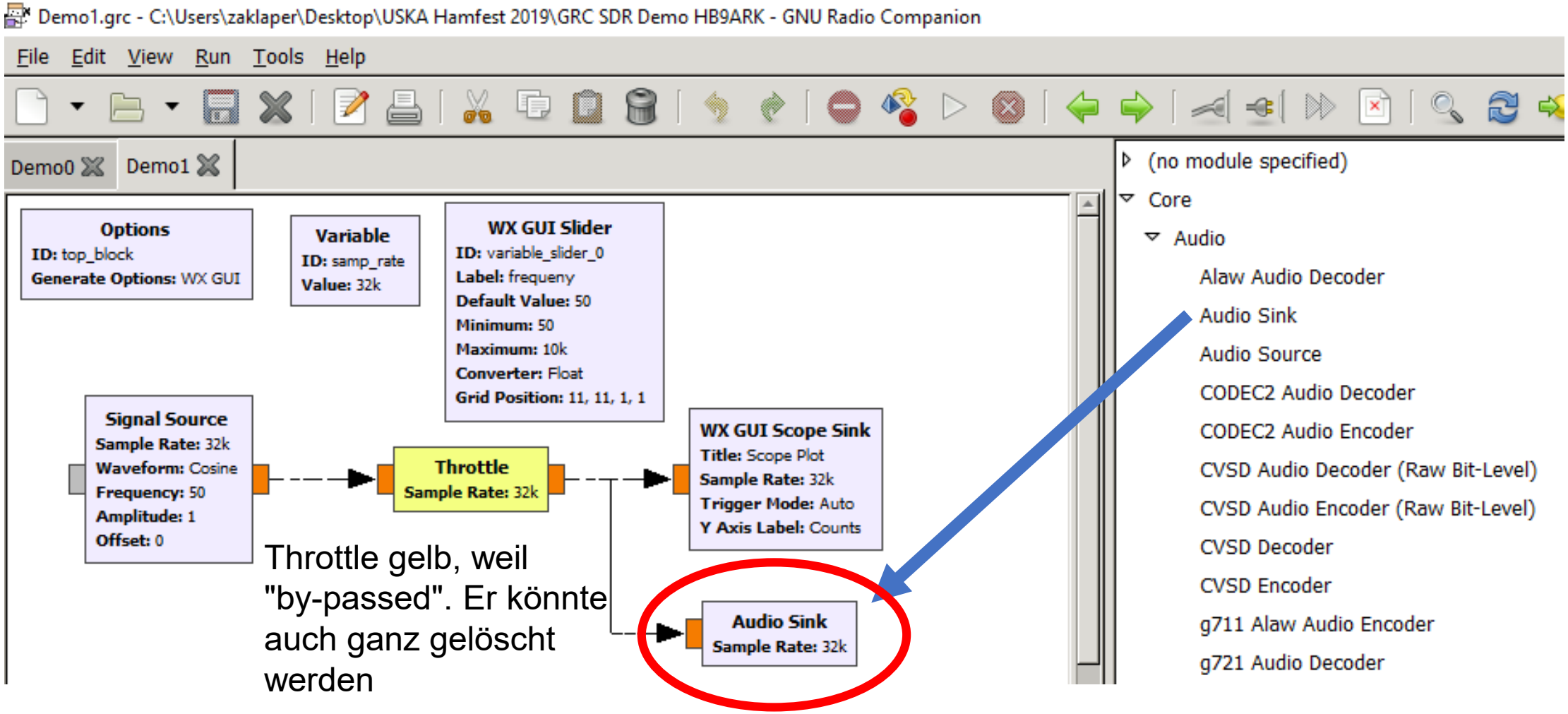
**Signal Source**
**Sample Rate:** 32k
**Waveform:** Cosine
**Frequency:** 1k
**Amplitude:** 1
**Offset:** 0

**Throttle**
**Sample Rate:** 32k

**WX GUI Scope Sink**
**Title:** Scope Plot
**Sample Rate:** 32k
**Trigger Mode:** Auto
**Y Axis Label:** Counts

"Throttle" bremst den Datenstrom und ist nötig,
wenn keine externe Hardware angeschlossen ist.

# … das Sinussignal hörbar machen (Demo1)



# Sinussignal erzeugen und Spektrum anzeigen (Demo2A)

*Demo2A.grc - C:\Users\zaklaper\Desktop\USKA Hamfest 2019\GRC SDR Demo HB9ARK - GNU Radio Compan

File Edit View Run Tools Help

Demo0 Demo2A Demo2B

**Options**
**ID:** top_block
**Title:** USKA Hamfest 2019
**Author:** Martin Klaper, HB9ARK
**Description:** Zwei ...addiert.
**Generate Options:** WX GUI

**Variable**
**ID:** samp_rate
**Value:** 32k

**Signal Source**
**Sample Rate:** 32k
**Waveform:** Cosine
**Frequency:** 1k
**Amplitude:** 1
**Offset:** 0

**Throttle**
**Sample Rate:** 32k

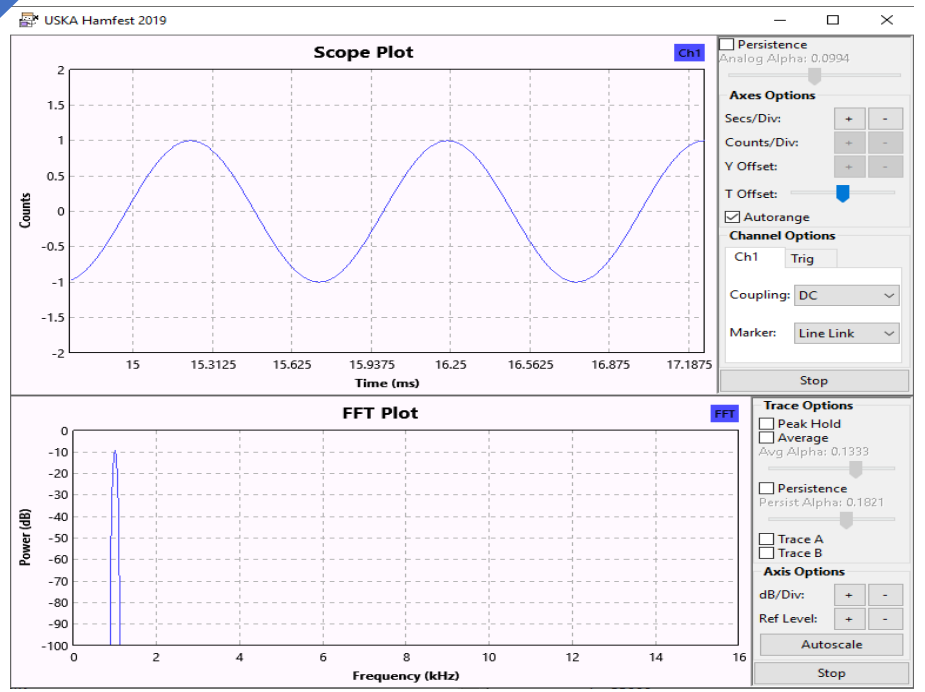
**WX GUI Scope Sink**
**Title:** Scope Plot
**Sample Rate:** 32k
**Trigger Mode:** Auto
**Y Axis Label:** Counts

**Audio Sink**
**Sample Rate:** 32k

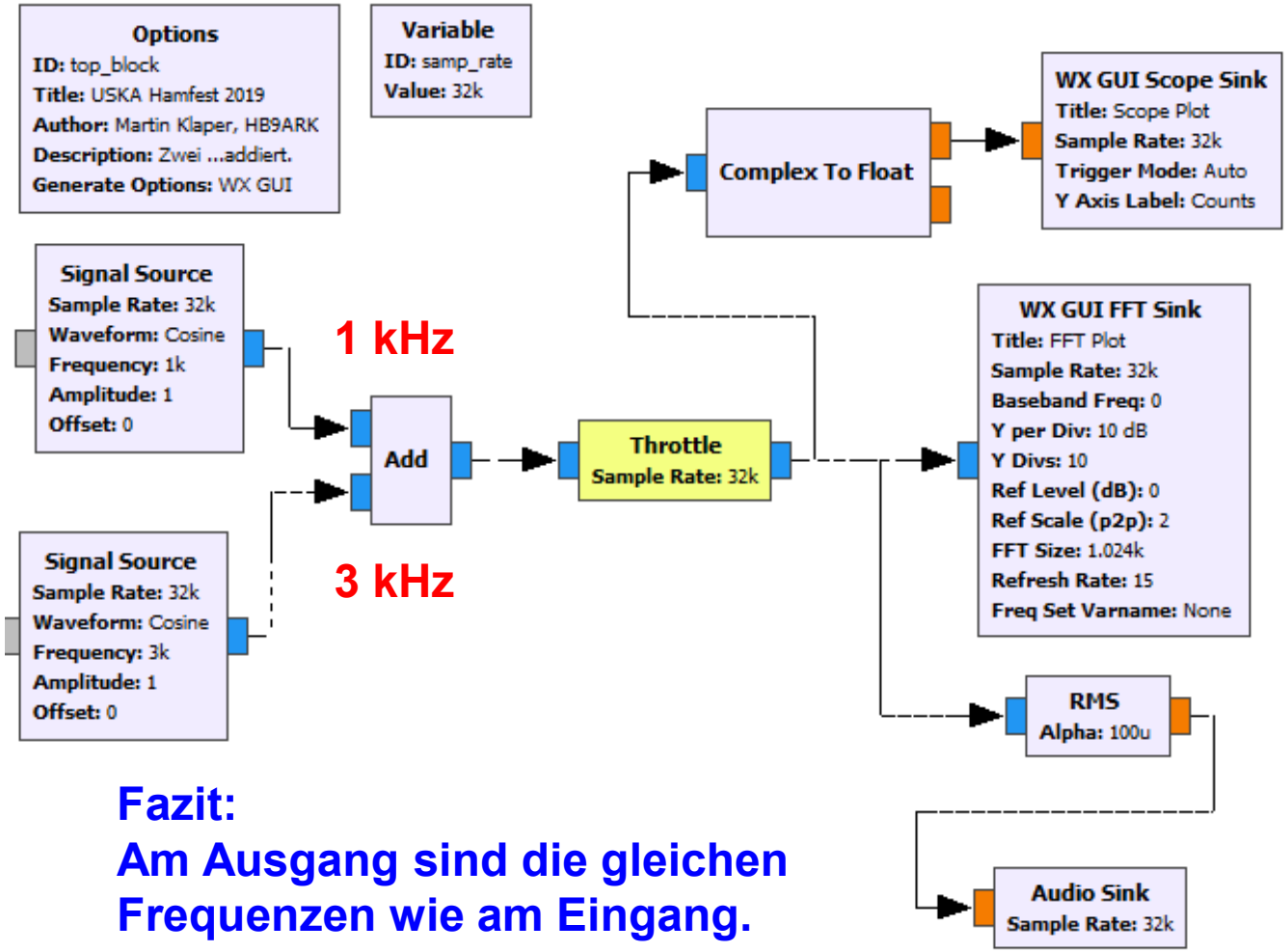
**WX GUI FFT Sink**
**Title:** FFT Plot
**Sample Rate:** 32k
**Baseband Freq:** 0
**Y per Div:** 10 dB
**Y Divs:** 10
**Ref Level (dB):** 0
**Ref Scale (p2p):** 2
**FFT Size:** 1.024k
**Refresh Rate:** 15
**Freq Set Varname:** None

Throttle gelb, weil "by-passed". Er könnte auch ganz gelöscht werden

- Fourier Analysis
- GUI Widgets
- Impairment Models
- Instrumentation
  - QT
  - WX
    - WX GUI Constellation Sink
    - WX GUI FFT Sink
    - WX GUI Histo Sink

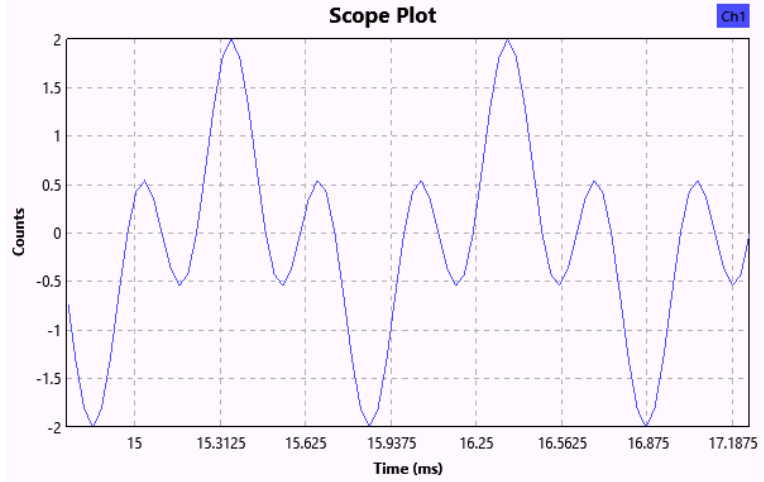


# Addition zweier Signale 1kHz und 3 kHz (Demo 2B)

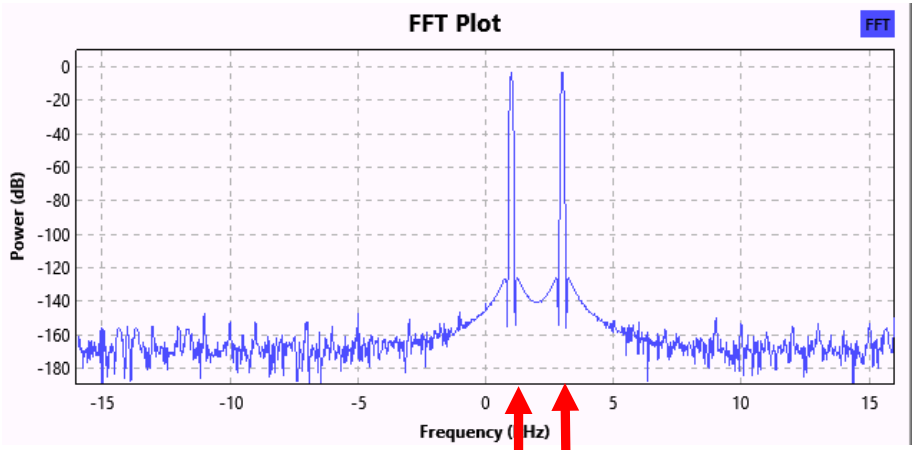



**Fazit:**
**Am Ausgang sind die gleichen Frequenzen wie am Eingang.**
**KEINE neuen Frequenzen**

Zeitbereich

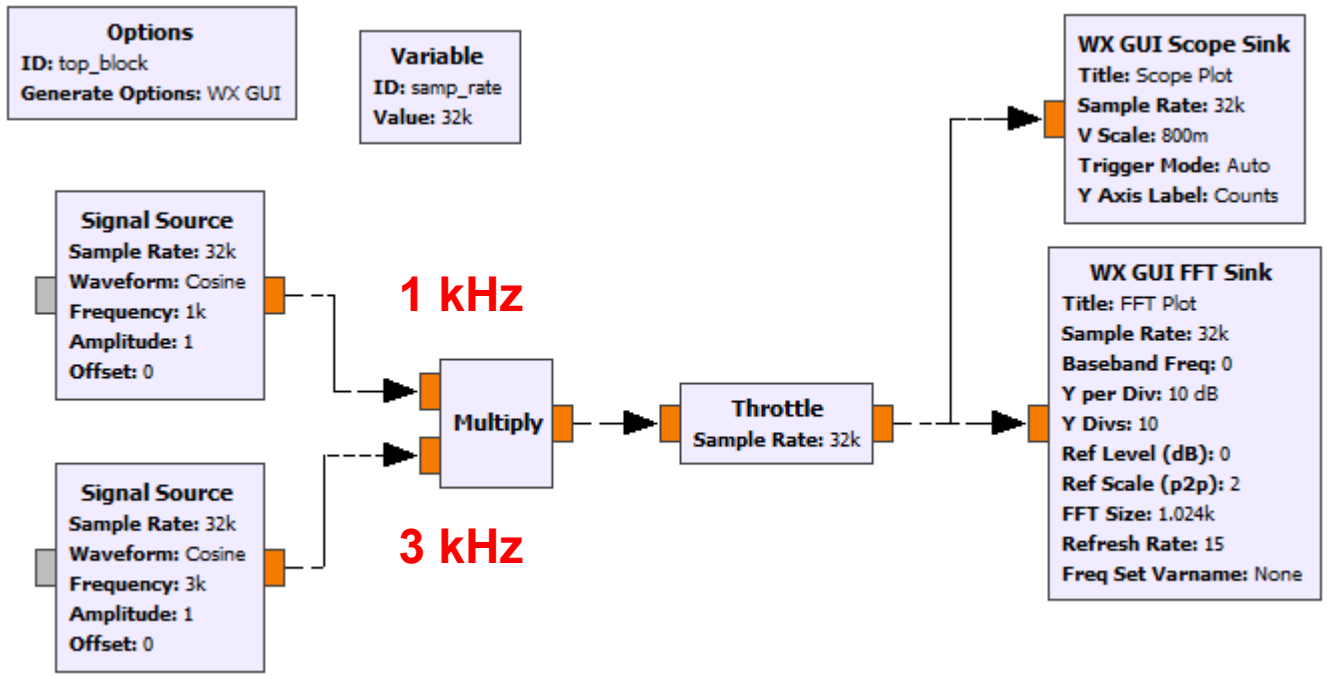



Frequenzbereich




**1 kHz** **3 kHz**

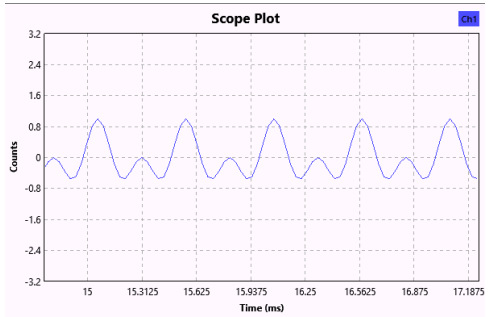
# Multiplikation zweier Signale 1kHz und 3 kHz (Demo 3A+3B)



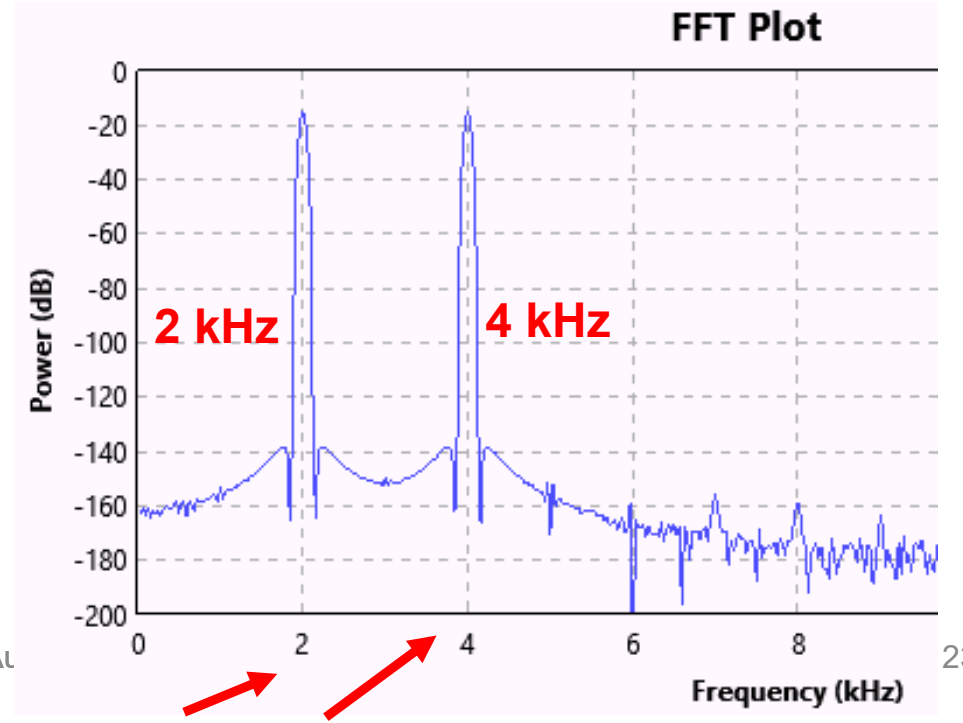

$$\cos\alpha \cdot \cos\beta = 1/2 \cdot \cos(\alpha + \beta) + 1/2 \cdot \cos(\alpha - \beta)$$

**Fazit:**
**Am Ausgang sind neue Frequenzen entstanden!!**
**Summen- und Differenz Frequenz**

Zeitbereich




Frequenzbereich

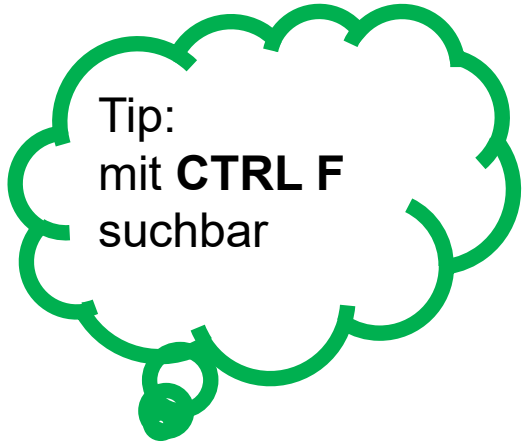


# Riesige Bibliothek an vorgefertigten Baublöcken I

- **Waveform Generators**
  - Constant Source
  - Noise Source
  - **Signal Source** (Sine, Square, Saw Tooth)

- **Modulators**
  - AM Demod
  - Continuous Phase Modulation
  - PSK Mod / Demod
  - DPSK Mod / Demod
  - GMSK Mod / Demod
  - QAM Mod / Demod
  - WBFM Receive
  - NBFM Receive

- **Instrumentation (i.e., GUIs)**
  - Constellation Sink
  - Frequency Sink
  - Histogram Sink
  - Number Sink
  - Time Raster Sink
  - Time Sink
  - Waterfall Sink

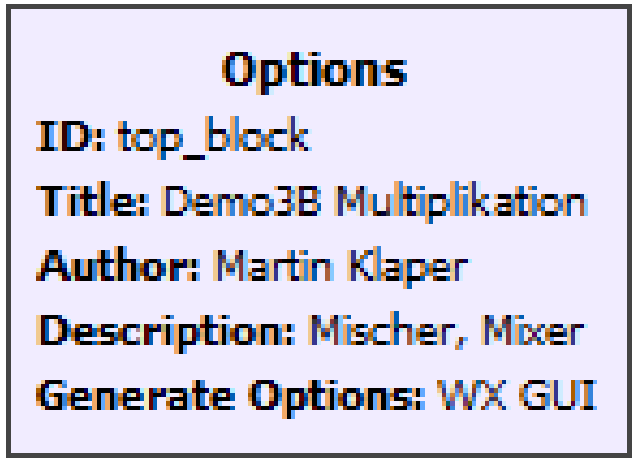


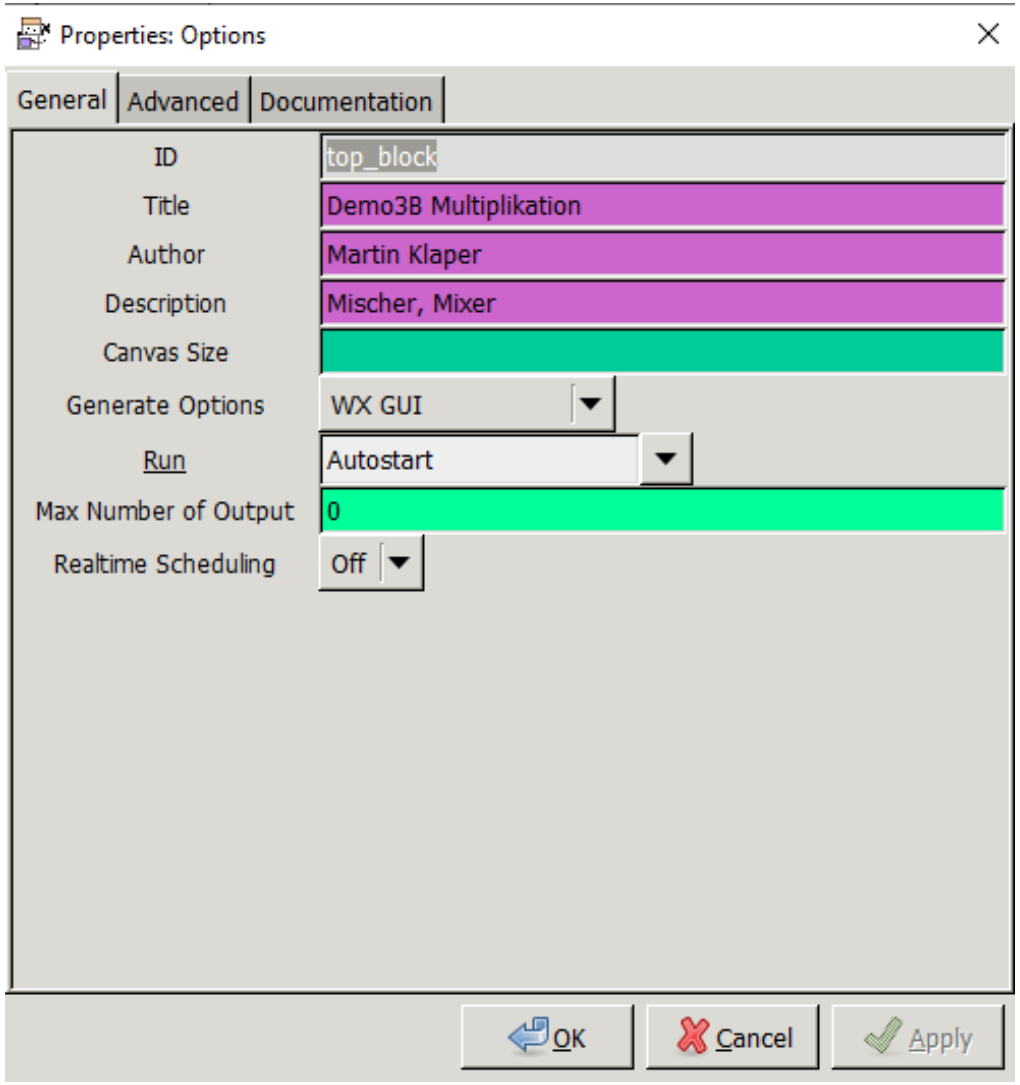

- **Math Operators**
  - Abs
  - **Add**
  - Complex Conjugate
  - Divide
  - Integrate
  - Log10
  - **Multiply**
  - RMS
  - Subtract

# Riesige Bibliothek an vorgefertigten Baublöcken I

- **Channel Models**
  - Channel Model
  - Fading Model
  - Dynamic Channel Model
  - Frequency Selective Fading Model
- **Filters**
  - **Band Pass** / Reject Filter
  - **Low / High Pass Filter**
  - IIR Filter
  - Generic Filterbank
  - Hilbert
  - Decimating FIR Filter
  - Root Raised Cosine Filter
  - FFT Filter
- **Fourier Analysis**
  - **FFT**
  - Log Power FFT
  - Goertzel (Resamplers)
  - **Fractional Resampler**
  - Polyphase Arbitrary Resampler
  - Rational Resampler (Synchronizers)
  - Clock Recovery MM
  - Correlate and Sync
  - **Costas Loop**
  - FLL Band-Edge
  - **PLL Freq Det**
  - PN Correlator
  - Polyphase Clock Sync

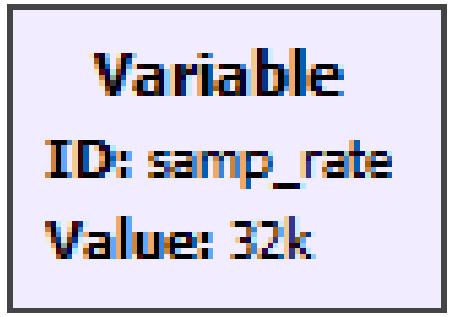
# Der Options Block

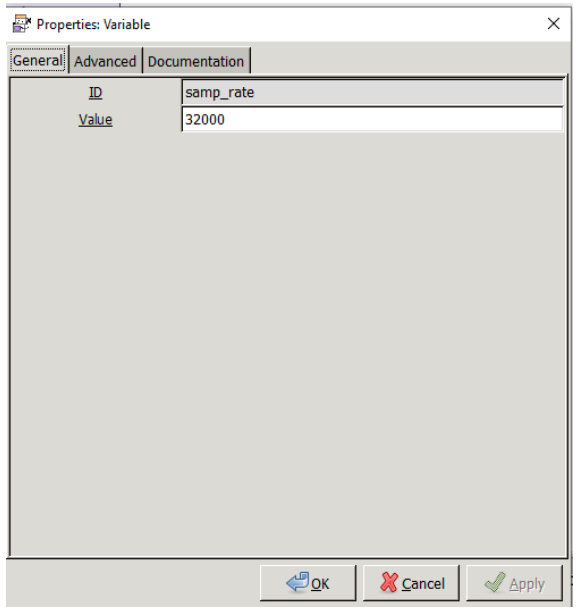


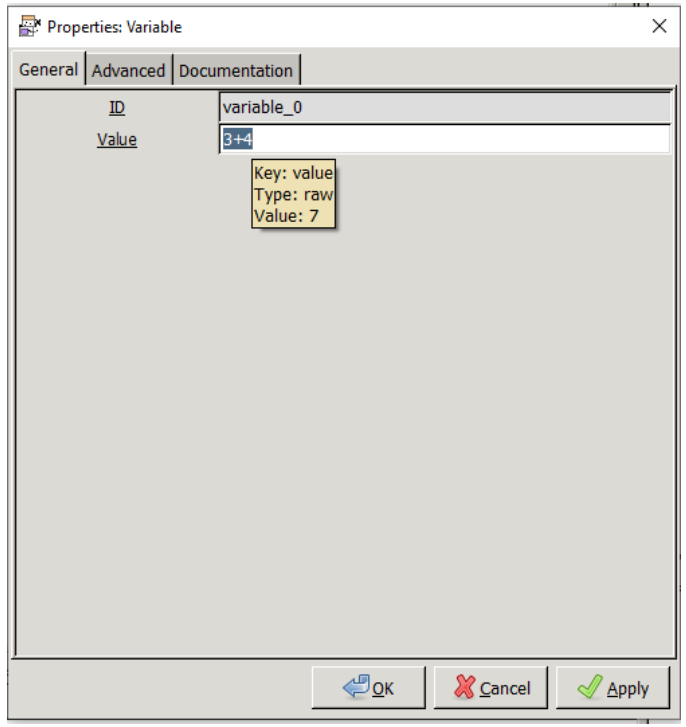




Der **Options** Block dient zum Setzen globaler Parameter

# Variablen (Platzhalter) definieren

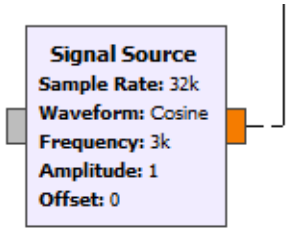


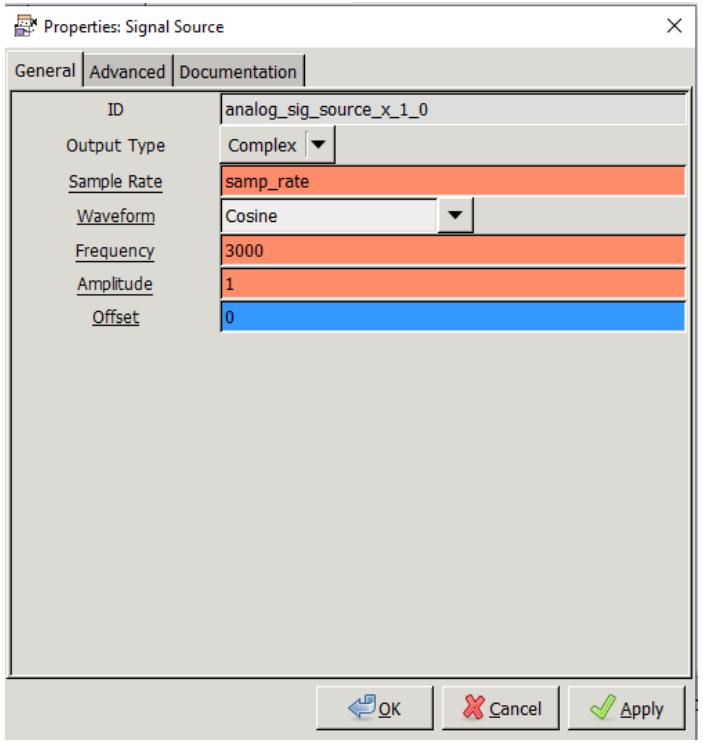




- Variablen erhalten einen frei wählbaren Namen:
  hier samp_rate
- und einen Wert:
  hier 32'000
- Es gilt Python Syntax ('expression')
- Man kann auch "rechnen"
  - 32000 (the default): ein integer
  - 32e6: 32000.0 (floating-point number)
  - int(32e6): 32000 (integer cast of floatingpoint number)

# Ein Funktionsblock im Detail: Signalquelle




Hinweis:
Die für unsere Zwecke interessanten HW Peripherien sind ein wenig versteckt:

| | |
|---|---|
| Industrial: | IO Pluto |
| UHD: | USRP (Matt Ettus) |
| FCD: | Funcube Dongle |
| OSMOCOM: | RTL-SDR |

u. s. w.

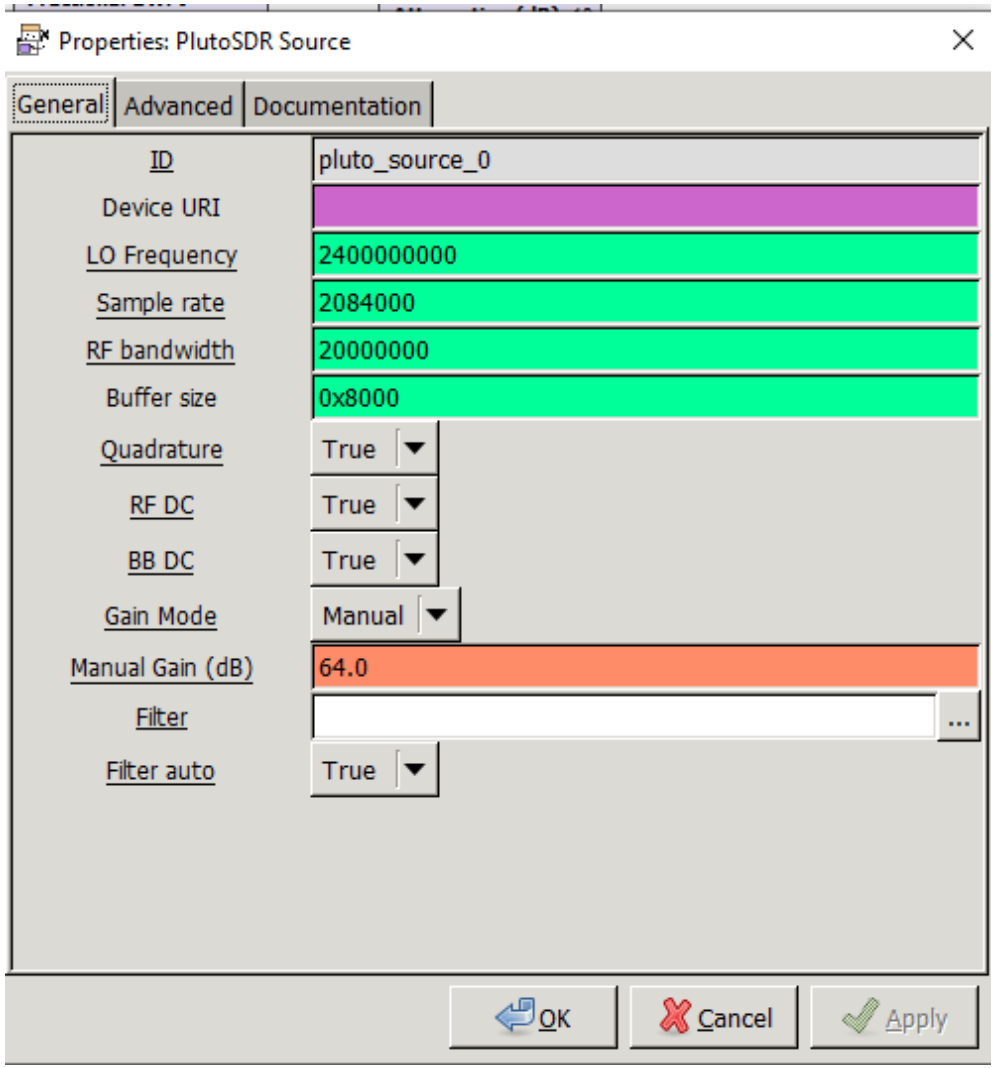
# Alle gängigen Hardware "Frontends" unterstützt

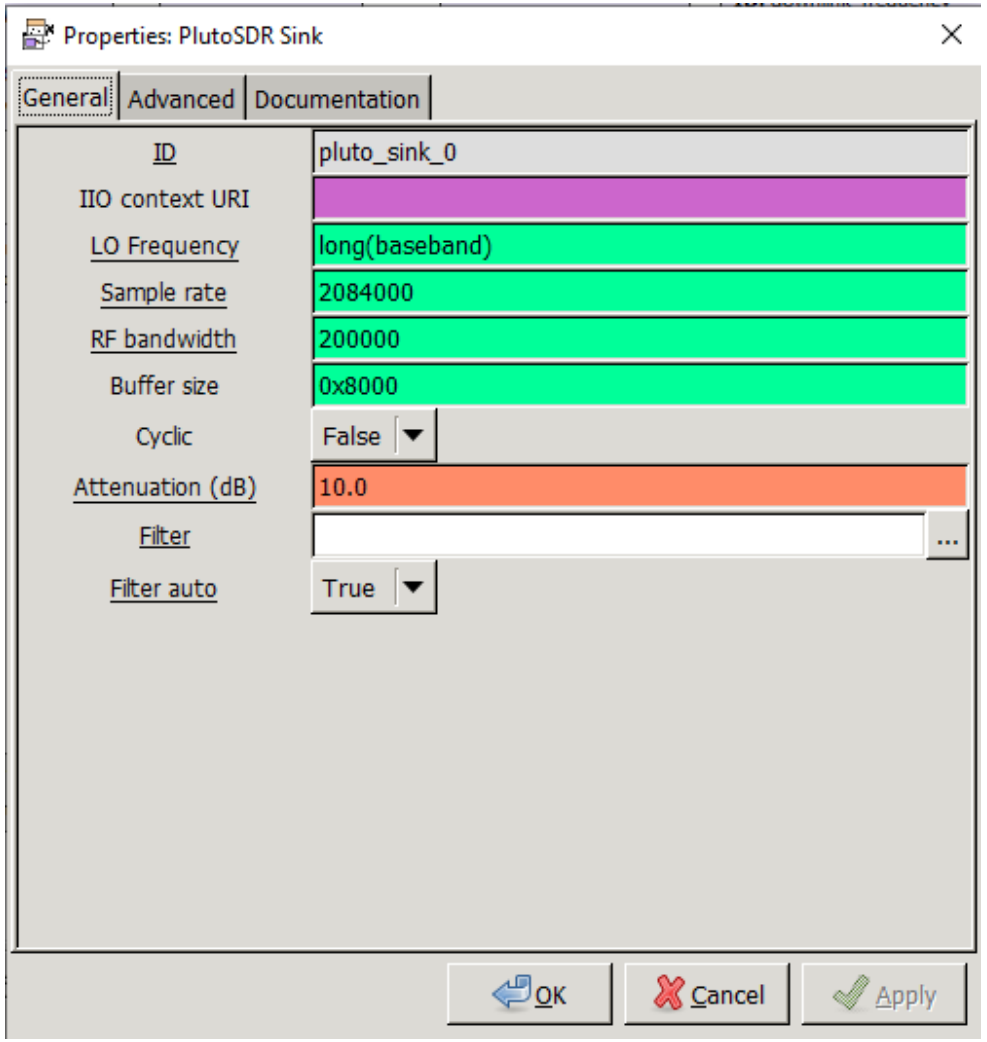
- RTL-SDR Dongles mit OSMOCOM
- Lime-SDR
- Adalm Pluto mit IIO, FMCOMMS
- Red Pitaya
- Funcube Dongle
- Hack-RF
- Perseus
- Blade RF
- SDR-Play
- Softrock
- KIWI-SDR Siehe "European GNU Radio Days 2019
- Universal Software Radio Peripheral (USRP) (Ettus Reserch)
- "Catch-All Clause" für eigene HW



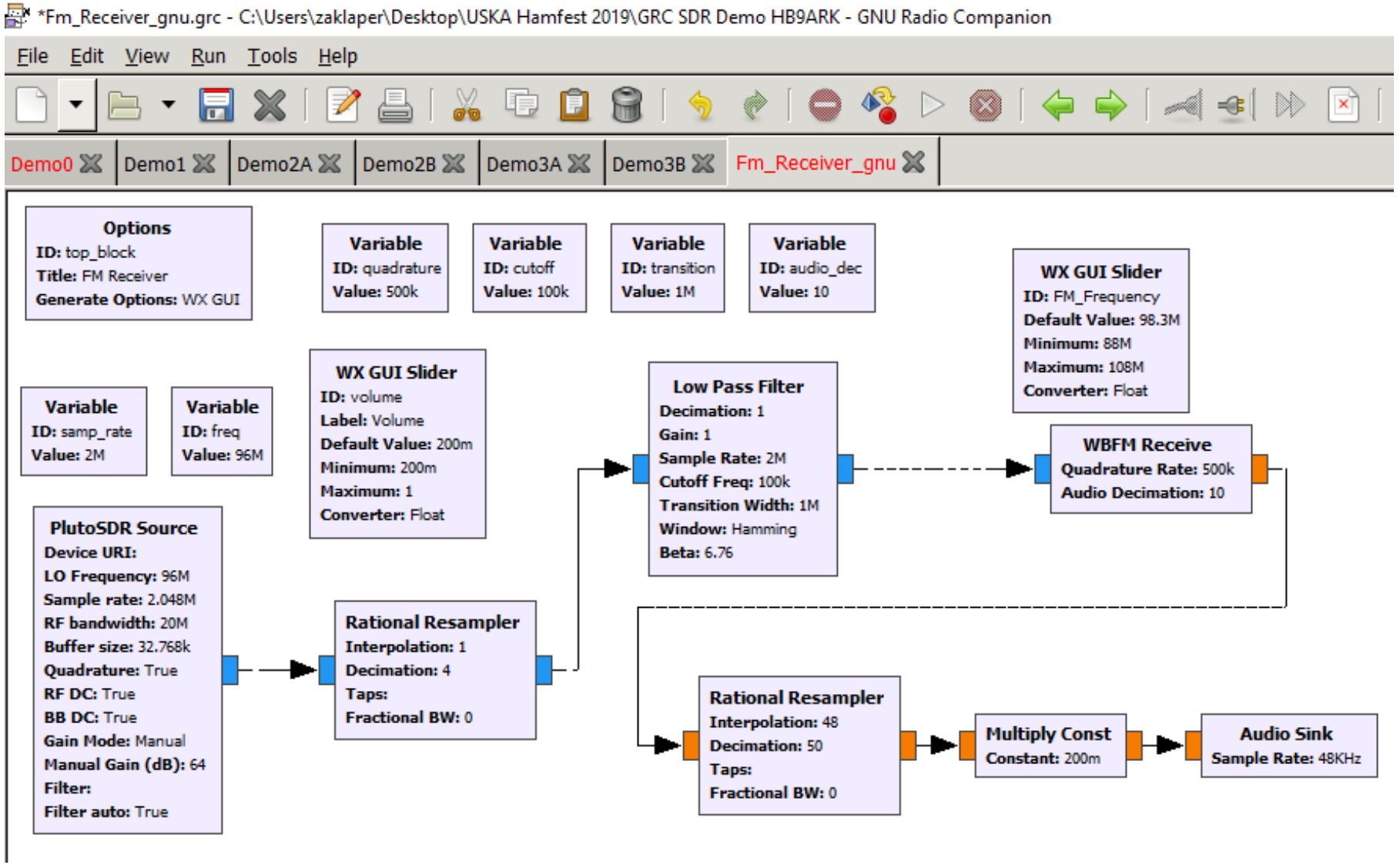



# Ein HW Funktionsblock im Detail: Adalm Pluto

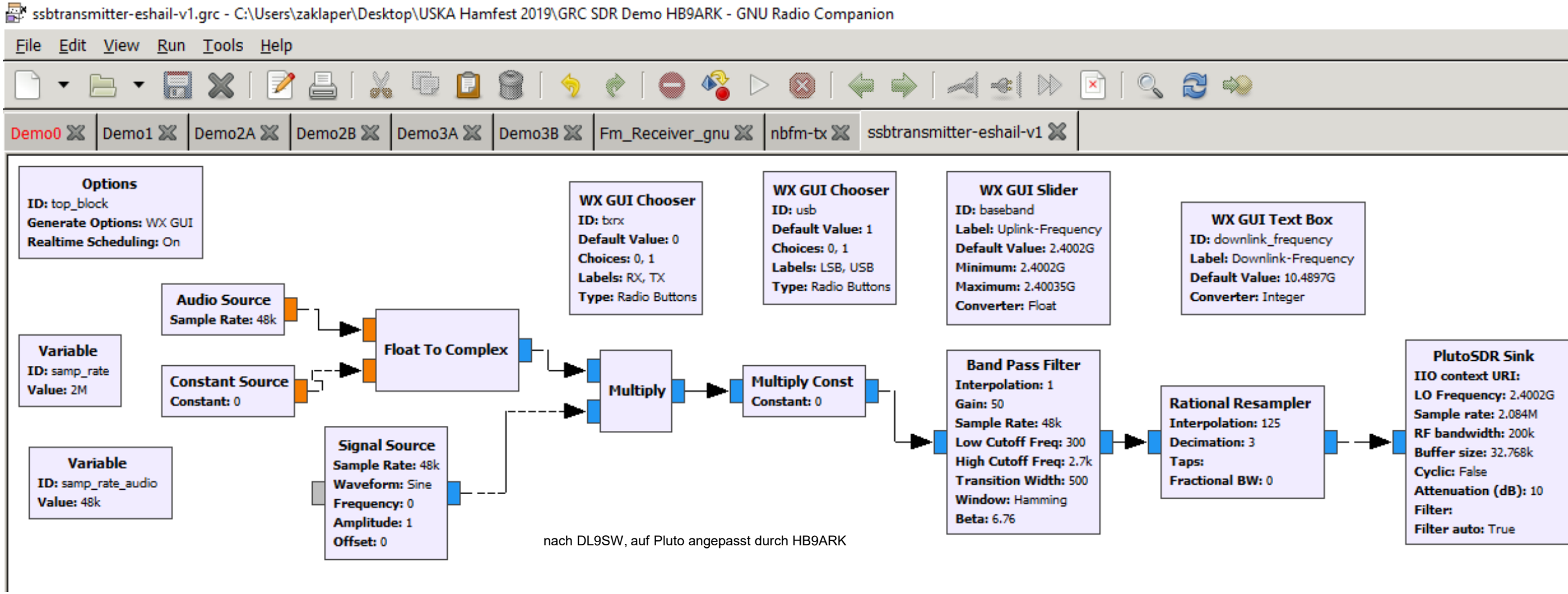


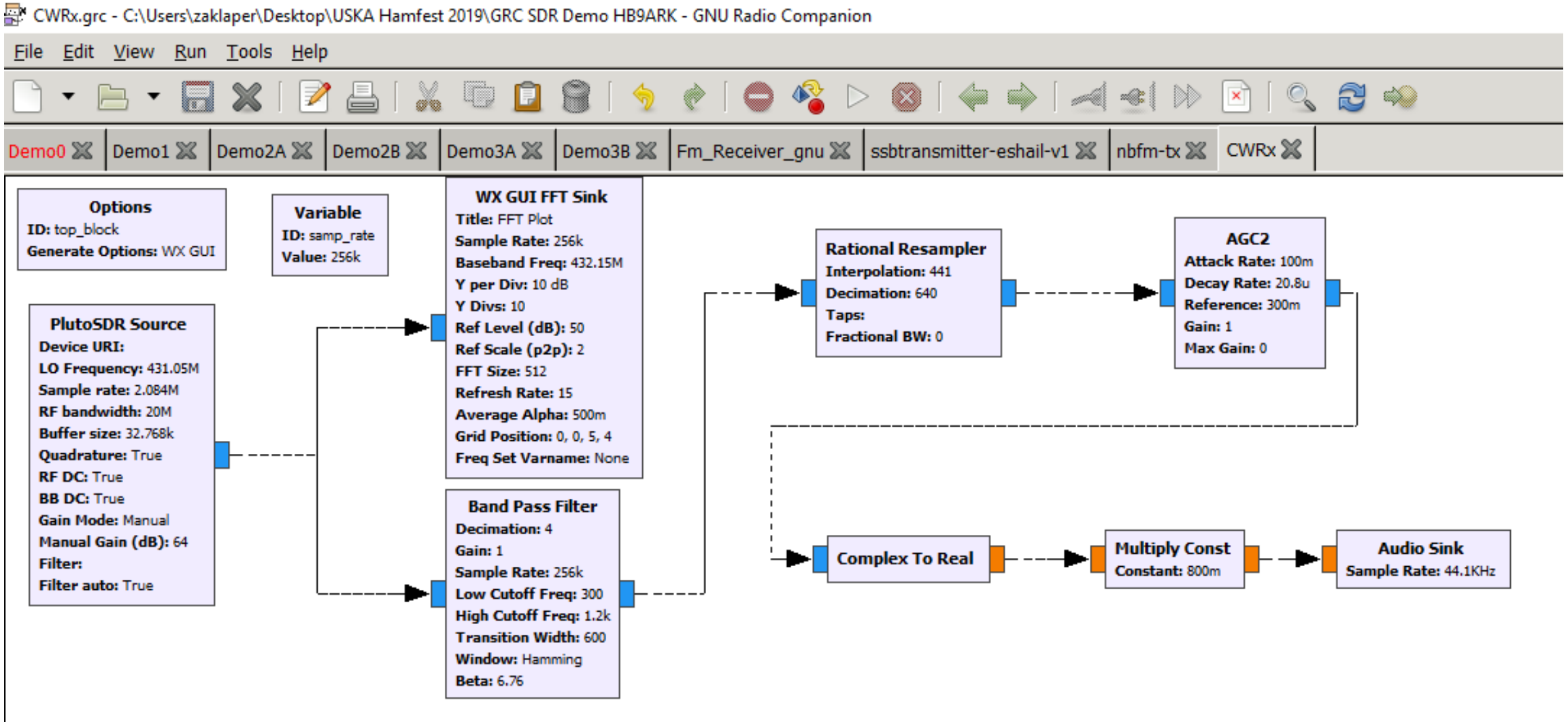
# Demo FM Empfänger GNU (FM Receiver)



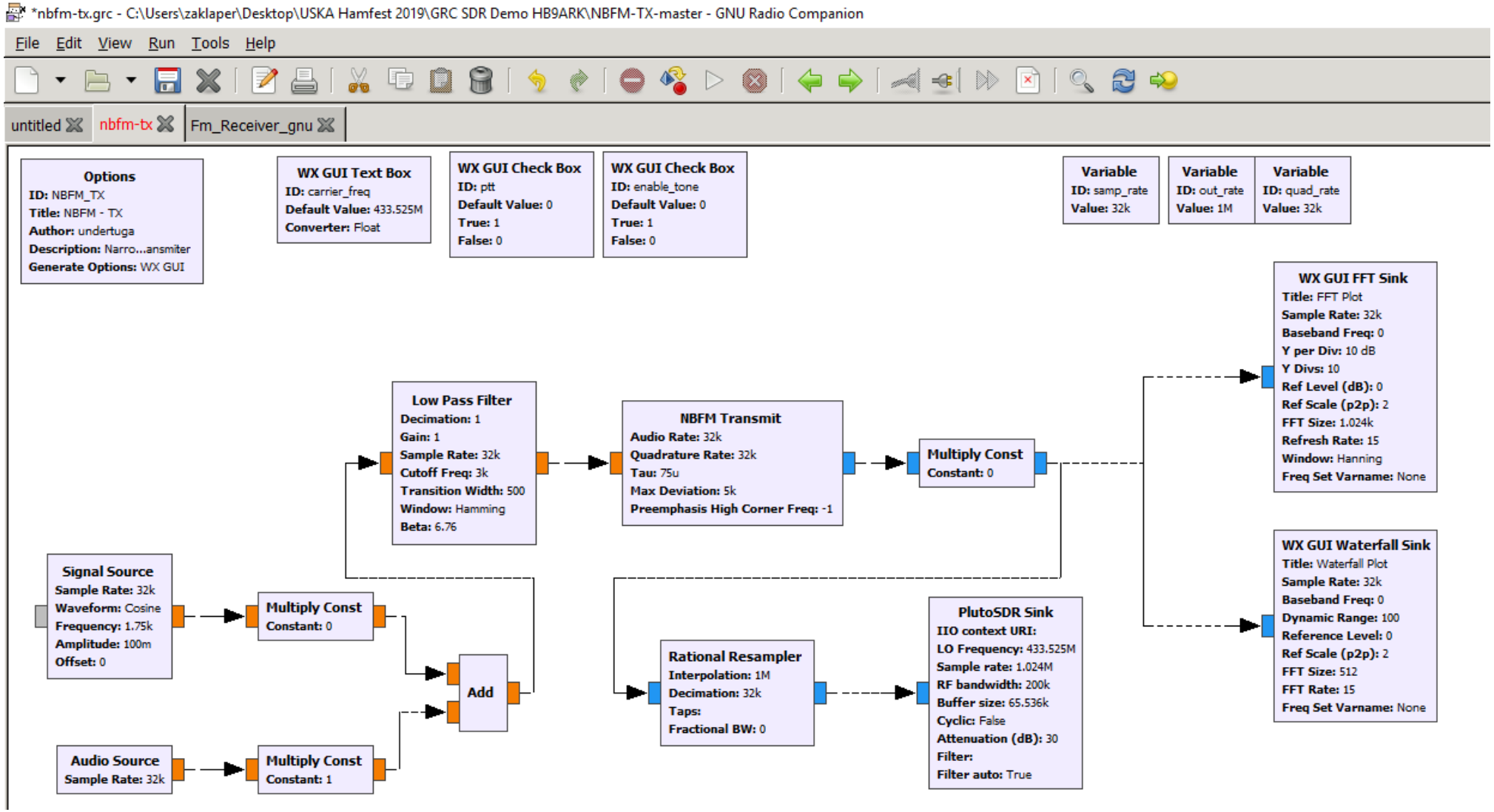
# Demo SSB Sender, z. Bsp. für QO100 (ssbtransmitter)



# Demo CW Empfänger (CWRx)



# Schmalband FM Sender



# ? Kontrollfragen

1. Bei welcher Rechenoperation entstehen neue Frequenzen ?
   *Antwort: Multiplikation, Mischen*
2. Auf welchen drei Ebenen kann in GNU Radio programmiert werden ?
   *Antwort: GRC (GNU Radio Companion), Python, C++*
3. Welche Aufgabe erfüllt ein GNU Radio Block ?
   *Antwort: Implementiert einen Baublock, "Legostein". Wandelt Input Daten in Output Daten um.*
4. Was bedeutet die Abkürzung "**GRC**" ?
   *Antwort: GNU Radio Companion*
5. Wie heisst die Seminarreihe der USKA für anspruchsvolle Funkamateure ?
   *Antwort: USKA Academy*

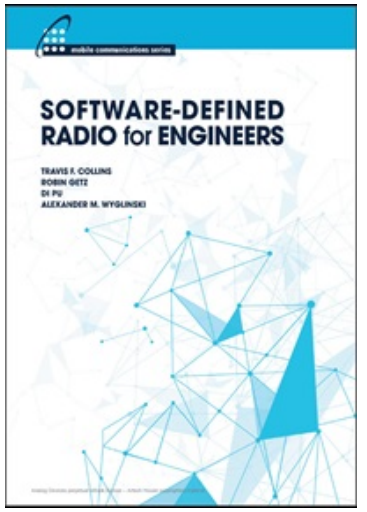
# Installation von GNU Radio mit GNU Radio Companion

- https://github.com/tfcollins/GNURadio_Windows_Build_Scripts/releases/download/1.5.0/gnuradio_3.7.11_iiosupport_win64.msi
- oder
- http://www.gcndevelopment.com/gnuradio/downloads.htm

| *Complete GNURadio 3.7.9.2 64-bit Packages* | 64-Bit Any CPU | 64-bit HASWELL+ (AVX2) CPU | 64-Bit Any CPU (Debug) |
|---|---|---|---|
| Windows Installer | v3.7.13.5/v1.6 (pdbs)<br>v3.7.13.4/v1.5 (pdbs)<br>v3.7.12/v1.4 (pdbs)<br>v3.7.11.1/v1.3 (pdbs)<br>v3.7.11/v1.2<br>v3.7.10.1/v1.1.2<br>v3.7.9.2/v1.1.1 | v3.7.13.5/v1.6<br>v3.7.13.4/v1.5<br>v3.7.12/v1.4<br>v3.7.11.1/v1.3<br>v3.7.11/v1.2<br>v3.7.10.1/v1.1.2<br>v3.7.9.2/v1.1.1 | v3.7.13.5/v1.6 (pdbs)<br>v3.7.13.4/v1.5 (pdbs)<br>v3.7.12/v1.4 (pdbs)<br>v3.7.11.1/v1.3 (pdbs) |

# Quellenangaben

- Ronny Nawrodt, Tutorial Spektralanalyse
- National Instruments, Tutorials
- https://www.ece.uvic.ca/~ece350/lab_manual/ar01s01s02.html
- Introduction to GNU Radio, MAC-TC, Tanguy Risset
- Alexandru Csete, OZ9AEC
- Marcel Joss, HB9TWM, Vorlesung Nachrichtentechnik, Hochfrequenztechnik
- Martin Klaper, HB9ARK, Vorlesungen Mobilkommunikation, Computer & Network Architecture
- GNU Radio Dokumentation und WIKI
- www.hb9hslu.ch
- Literaturangabe (empfehlenswert):




https://www.analog.com/media/en/training-seminars/design-handbooks/Software-Defined-Radio-for-Engineers-2018/SDR4Engineers.pdf

# Wrap Up

- Software Radio ist ein aktuelles und spannendes Thema, insbesondere für den Amateurfunk.
- GNU Radio ist ein vorzügliches und frei verfügbares Werkzeug zur Umsetzung.
- Es kann für viele Spielarten des Amateurfunks eingesetzt und erweitert werden.
- Erfolgreiche Beschäftigung mit Software Radio erfordert Fachwissen auf verschiedenen Gebieten.
- GNU Radio mit GRC verringert die Einstiegshürde deutlich.
- … und noch bequemer geht’s mit einem Seminar der USKA Academy.

# Verzeichnis der Demos

| Datei | Beschreibung | Experiment |
|---|---|---|
| Demo0.grc | Sinus Signal auf dem Oszilloskop anzeigen | |
| Demo1.grc | Sinus Signal auf dem Oszilloskop anzeigen und hörbar machen | Frequenz einstellen, "Throttle" erklären, falsche Sample Rate beim Audio Sink zeigen, Waveform umschalten |
| Demo2A.grc | Sinus Signal auf dem Oszilloskop anzeigen und hörbar machen | Eine einzige Spektral-Linie zeigen, den "Throttle" löschen |
| Demo2B.grc | Zwei Sinus Signale ADDIEREN | Komplexwertig gerechnet, Spektral-Linien bei 1 und 3 kHz, Zeitsignal Amplituden addiert, KEINE neuen Frequenzen |
| Demo3A.grc | Zwei Sinus Signale MULTIPLIZIERT | Spektral-Linien bei 2 und 4 kHz, Zeitsignal Amplituden addiert, NEUE Frequenzen |
| Demo3B.grc | Wie 3A, aber 1 kHz ausgefiltert | Frequenztranslation, Spiegelfrequenz weggefiltert |
| Fm_Receiver_gnu.grc | FM Receiver | DRS auf Rigi 90.9 MHz hören. Aufgabe auf nächstes Mal: Frequenz über Schieber einstellbar machen. Drei Presets mit Radio Buttons wählbar machen |
| ssbtransmitter-eshail-v1.grc | SSB Sender | Mit FT-857 auf 70 cm abhören. |
| CWRx.grc | CW Empfänger | Mit FT-857 senden |