Frequenzspektrum analysieren

# Vorbereitung:

Der PlutoSDR wurde ordnungsgemäß mit dem Computer verbunden.

Die erforderliche GNU Radio Software wurde gestartet und konfiguriert.

# Erstellung des Testaufbaus:

Wir haben einen Testaufbau erstellt, der es ermöglicht, das Frequenzspektrum mithilfe des PlutoSDR zu empfangen.

Hierfür wurden einige Blöcke in GNU Radio verwendet, um die Signale zu verarbeiten und darzustellen. Diese Blöcke waren Pluto SDR Source, Variable, QT GUI Range und Frequency Sink

# Konfiguration der Blöcke:

Während des Tests stellten wir fest, dass einige Blöcke in der Standardkonfiguration nicht optimal funktionierten.

Wir mussten daher bestimmte Parameter und Einstellungen anpassen, um die gewünschten Ergebnisse zu erzielen.

PlutoSDR Source :

* IIO context URI auf pluto.local
* LO Frequency: int(freq)
* Sample rate: samp\_rate

Variable:

* Value auf 2.4e6 ändern

QT GUI Range:

* Id auf freq
* Type float
* Default Value auf 100e6
* Start bei 80e6
* Stop bei 120e6
* Step: 1e6
* Widget: Counter + Slider

QT GUI Frequency Sink:

* Center Frequency: freq
* Bandwidth: samp\_rate
* Config: Control Panel auf Yes

# Funktionsweiße der Bauteile

PlutoSDR Source: Dieser Block nimmt Signale von einem PlutoSDR-Gerät auf und leitet sie an andere Teile des Programms weiter.

QT GUI Range: Ein grafischer Regler, mit dem Benutzer Werte einstellen können, die im Programm verwendet werden.

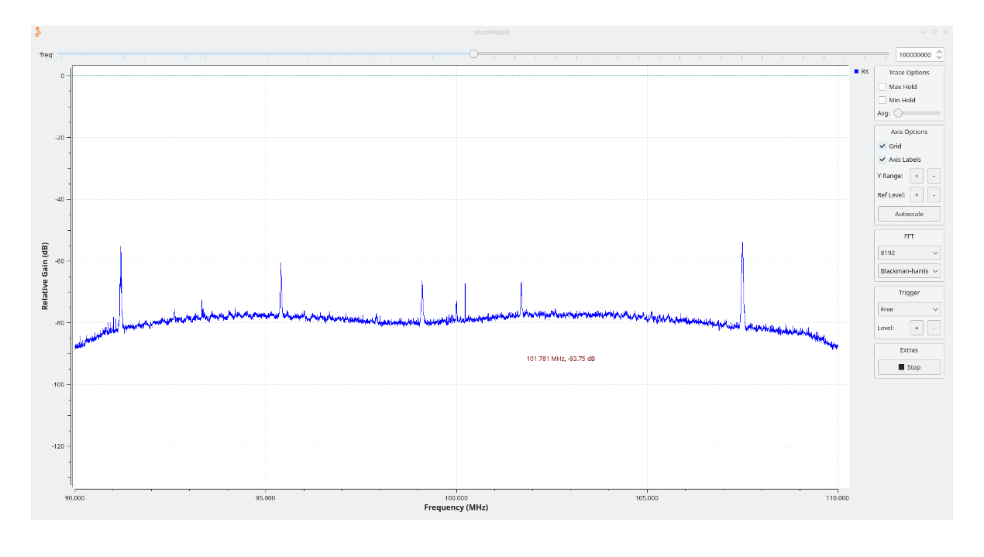
Variable: Ein Ort, um Werte zu speichern, die im Programm verwendet werden können.

QT GUI Frequency Sink: Zeigt Frequenzdaten auf dem Bildschirm an.

# Recherche zur Ermittlung der Frequenzen:

Um die empfangenen Signale genauer zu analysieren, haben wir recherchiert, um herauszufinden, welche Radiofrequenzen bestimmten Sendern zuzuordnen sind.

|  |  |  |
| --- | --- | --- |
| Frequenz/MHz | Sender | Senderstandort |
| 91,2 | Ö1 | Schöckl |
| 95,4 | Radio Steiermark | Schöckl |
| 99,1 | Antenne Steiermark | Schöckl |
| 100,0 | Radio Grün Weiß | Schlossberg |
| 100,2 | Hitradio Ö3 | Sankt Katharein (Laming)/ Maxl |
| 101,7 | FM4 | Schöckl |
| 107,5 | Kronehit Steiermark | Plabutsch Fürstenstand |



# Ergebnisse:

Durch die Anpassung der Blockkonfiguration konnten wir das Frequenzspektrum erfolgreich empfangen und analysieren. Die Recherche zur Ermittlung der Senderfrequenzen ermöglichte es uns, die empfangenen Signale besser zu verstehen und zu interpretieren.

# Fazit:

Der Test des PlutoSDR mit GNU-Radio war erfolgreich. Wir konnten das Frequenzspektrum effektiv analysieren und haben wichtige Erkenntnisse über die Konfiguration der Blöcke sowie die Zuordnung der Senderfrequenzen gewonnen. Diese Ergebnisse legen nahe, dass das System für weitere Experimente und Anwendungen geeignet ist.

Radio Frequenzen abhören

# Einstellung des Radiosenders:

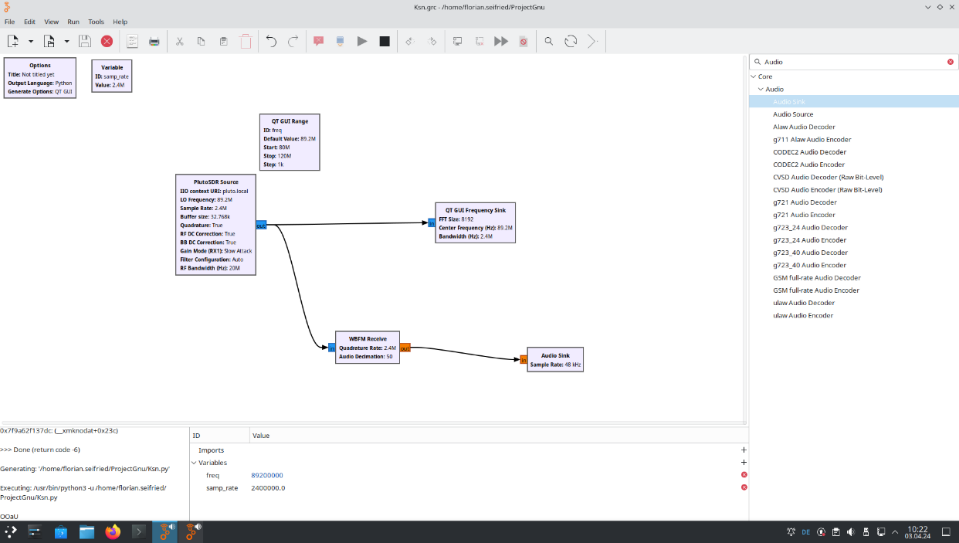
Wir haben versucht, einen Radiosender einzustellen, indem wir die entsprechenden Blöcke in GNU Radio konfiguriert haben.

Dazu wurden die Blöcke PlutoSDR Source, QT GUI Range, Variable, QT GUI Frequency Sink, WBFM Receive und Audio Sink verwendet.

Bei diesen stellten wir um:

Variable:

* ID: samp\_rate
* Value: 2.4e6



PlutoSDR Source:

* LO Frequency: int(freq)
* IIO context URI: pluto.local
* Sample rate: samp\_rate

QT GUI Range:

* ID: freq
* Default Value: 89.2e6
* Start: 80e6
* Stop: 20e6
* Step: 1k

QT GUI Frequency Sink:

* FFT Size: 8192
* Center Frequency: 89.2e6

WBFM Receive:

* Quadrature Rate: (int)samp\_rate
* Audio Decimation = Quadrature Rate / Sample Rate von Audio Sink = 50

Audio Sink:

* Sample Rate: 48k

# Funktionsweiße:

PlutoSDR Source: Dieser Block nimmt Signale von einem PlutoSDR-Gerät auf und leitet sie an andere Teile des Programms weiter.

QT GUI Range: Ein grafischer Regler, mit dem Benutzer Werte einstellen können, die im Programm verwendet werden.

Variable: Ein Ort, um Werte zu speichern, die im Programm verwendet werden können.

QT GUI Frequency Sink: Zeigt Frequenzdaten auf dem Bildschirm an.

WBFM Receive: Ein Block, der FM-Signale empfängt und in hörbare Töne umwandelt.

Audio Sink: Spielt Audiodaten über Lautsprecher oder Kopfhörer ab.

# Ausgabe über die Lautsprecher:

Nach erfolgreicher Einstellung des Radiosenders haben wir die Ausgabe über die Lautsprecher konfiguriert.

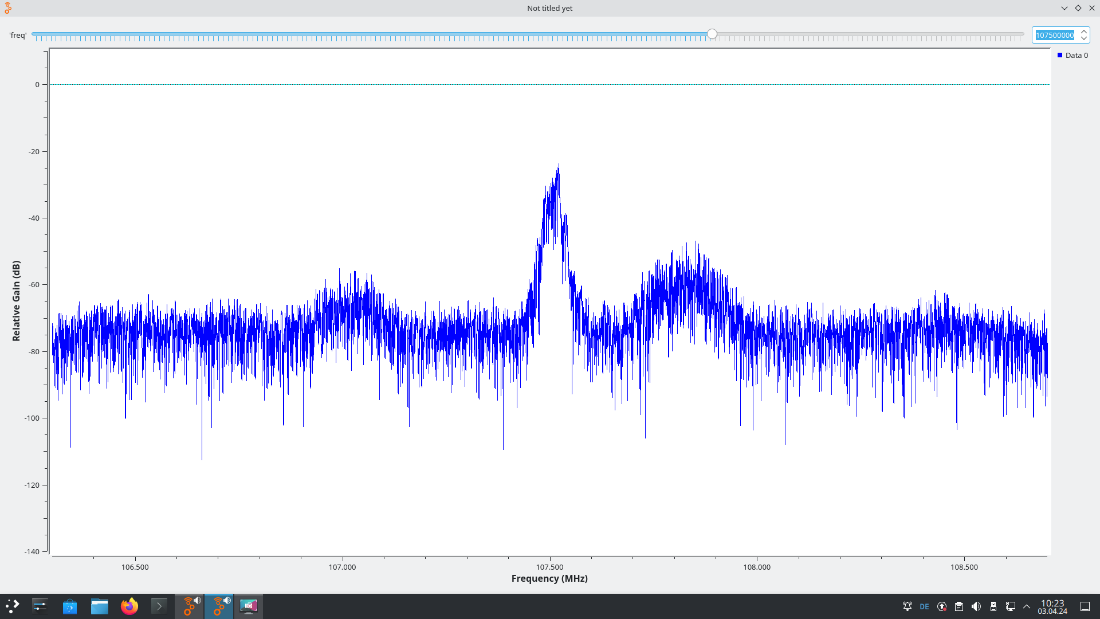
Wir haben sicherheitsrelevante Parameter wie die Lautstärke und die Audiokanäle überprüft, um eine klare Wiedergabe sicherzustellen, da anfangs die Ausgabe nicht funktioniert hat. Der Fehler dabei war, dass der Lautsprecher nicht richtig konfiguriert war

# Test der Wiedergabe:

Nach Abschluss der Konfiguration haben wir die Wiedergabe über die Lautsprecher getestet.

Durch das Abstimmen auf verschiedene Frequenzen konnten wir schließlich einen Sender empfangen und über die Lautsprecher hören. Dieser Sender war bei der Frequenz 107,5 MHz.

# Ergebnisse:

Wir konnten erfolgreich einen Radiosender mit dem PlutoSDR empfangen und über die Lautsprecher ausgeben.

Beispielsweise konnten wir den Sender "Kronehit" klar und deutlich hören.

# Fazit:

Der Test der Einstellung und Ausgabe eines Radiosenders mit dem PlutoSDR und GNU Radio war erfolgreich. Wir konnten zeigen, dass das System in der Lage ist, Radiosignale zu empfangen und über die Lautsprecher auszugeben. Dies ermöglicht interessante Anwendungsfälle und Experimente im Bereich der drahtlosen Kommunikation.

Dopplereffekt

# Konfiguration der Blöcke:

Wir haben spezifische Blöcke in GNU Radio verwendet, um das Frequenzspektrum zu analysieren und die Veränderungen der Frequenz zu verfolgen.

Wir verwendeten die Blöcke Options, Variable, 4x QT GUI Range, PlutoSDR Source, Signal Source, PlutoSDR Sink, 4x QT GUI Time Sink, 3x QT GUI Frequency Sink, Complex to real, Audio Sink, Multiply Const, Band Reject Filter und Low Pass Filter

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

Automatisch generierte BeschreibungDiese Blöcke wurden entsprechend konfiguriert, um das auditive Feedback zu generieren, das den Dopplereffekt darstellen sollte.

Hierbei stellen wir um:

Variable:

* ID: samp\_rate
* Value: 2,4e6

**Sender Seite:**

QT GUI Range:

* ID: tone\_freq
* Default Value: 500
* Start: -2e3
* Stop: 2e3
* Step: 1

QT GUI Range:

* ID: tx\_attenuation
* Default Value: 10
* Start: 0
* Stop: 100
* Step: 1

PlutoSDR Sink:

* LO Frequency: 3500000000
* Sample Rate: int(samp\_rate)
* RF\_Bandwidth: 20000000
* Buffer Size: 32768
* Cyclic: False
* Attenuation TX1 (Db): tx\_attenuation
* Filter Auto: True

QT GUI Time Sink:

* Y Axis Label: Amplitude
* Number of Points: 1024
* Sample Rate: samp\_rate
* Y min: -1
* Y max: 1

Signal Source:

* Sample Rate: 2,4e6
* Waveform: Cosine
* Frequency: 500
* Amplitude: 1

**Empfänger Seite:**

PlutoSDR Source:

* LO Frequency: 3500000000
* Sample Rate: int(samp\_rate)
* RF Bandwith: 20000000
* Buffer Size: 32768
* Quadrature: True
* RF DC Correction: True
* BB DC Correction: True
* Manual Gain (RX1)(dB): rx\_gain

In Config stellen wir um fürs QT GUI Time Sink:

* Line 1 Label: TX re
* Line 2 Label: RX im

QT GUI Time Sink:

* Y Axis Label: Amplitude
* Number of Points: 1024\*8
* Sample Rate: samp\_rate
* Y min: -1
* Y max: 1

In Config stellen wir um fürs QT GUI Time Sink:

* Line 1 Label: RX re
* Line 2 Label: RX im

QT GUI Frequency Sink:

* FFT Size: 1024\*8
* Bandwidth: samp\_rate
* Y min: -140
* Y max: 10

In Config stellen wir um fürs QT GUI Frequency Sink:

* Line 1 Label: RX

Low Pass Filter:

* Decimation: 50
* Sample Rate: samp\_rate
* Cutoff Freq: tone\_freq\*2
* Transition Width: 250

QT GUI Time Sink:

* Sample rate: samp\_rate/50

In Config:

* Line 1 Label: LPF re
* Line 2 Label: LPF im

QT GUI Frequency Sink:

* FFT Size: 1024\*8
* Bandwidth: samp\_rate/50

In Config:

* Line 1 Label: LPF

Band Reject Filter:

* Sample Rate: samp\_rate/50
* Low Cutoff Freq: 495
* High Coutoff Freq: 505
* Transition Width: 5

Multiple Const:

* Constant: audio\_gain

QT GUI Range:

* ID: audio\_gain
* Default Value: 1
* Start: 0
* Stop: 100
* Step: 1

QT GUI Time Sink:

* Sample Rate: samp\_rate/50

In Config:

* Line 1 Label: band reject re
* Line 2 Label: band reject im

QT GUI Frequency Sink:

* FFT Size: 1024\*8
* Bandwidth: samp\_rate/50

In Config:

* Line 1 Label: band reject

Complex to Real

Audio Sink:

* Sample Rate: 48kHz

# Funktionsweiße der Blöcke:

PlutoSDR Source: Dieser Block nimmt Signale von einem PlutoSDR-Gerät auf und leitet sie an andere Teile des Programms weiter.

QT GUI Range: Ein grafischer Regler, mit dem Benutzer Werte einstellen können, die im Programm verwendet werden.

Variable: Ein Ort, um Werte zu speichern, die im Programm verwendet werden können.

QT GUI Frequency Sink: Zeigt Frequenzdaten auf dem Bildschirm an.

Audio Sink: Spielt Audiodaten über Lautsprecher oder Kopfhörer ab.

Signal Source: Dieser Block erzeugt ein Signal innerhalb des GNU Radio-Programms. Es kann verwendet werden, um Testsignale zu erzeugen oder als Quelle für Echtzeitsignale in Simulationen oder Experimenten.

PlutoSDR Sink: Dieser Block sendet Daten von GNU Radio an ein PlutoSDR-Gerät. Es ermöglicht das Senden von Signalen oder Daten an das PlutoSDR-Gerät für die Übertragung über Funk oder andere Anwendungen.

QT GUI Time Sink: Dieser Block ist eine grafische Darstellung von Zeitbereichssignalen. Es zeigt die zeitabhängige Amplitude von Signalen auf dem Bildschirm an, was besonders nützlich ist, um das zeitliche Verhalten von Signalen zu analysieren.

Complex to Real: Dieser Block wandelt komplexe Signale in reelle Signale um. Komplexe Signale haben zwei Teile: Realteil und Imaginärteil. Dieser Block extrahiert den Realteil aus dem komplexen Signal und gibt ihn als reelles Signal aus.

Multiply Const: Dieser Block multipliziert ein Eingangssignal mit einer konstanten Zahl. Es ist nützlich für verschiedene Anwendungen, wie z.B. das Anpassen der Amplitude eines Signals oder das Einführen von Verstärkung oder Dämpfung.

Band Reject Filter: Dieser Block filtert ein Signal, indem er eine bestimmte Bandbreite von Frequenzen unterdrückt und alle anderen durchlässt. Es ist besonders effektiv, um Störungen oder unerwünschte Frequenzen aus einem Signal zu entfernen.

Low Pass Filter: Dieser Block lässt nur die niedrigen Frequenzen eines Signals passieren und unterdrückt höhere Frequenzen. Er wird häufig verwendet, um Rauschen zu reduzieren oder Signale zu glätten, indem hochfrequente Komponenten entfernt werden.

# Durchführung des Tests:

Nach der Konfiguration haben wir verschiedene Objekte in Bewegung gebracht, um den Dopplereffekt zu simulieren.

Wir erwarteten, dass die Veränderungen in der Frequenz des empfangenen Signals durch das auditive Feedback hörbar würden.

# Ergebnisse:

Leider konnten wir trotz sorgfältiger Konfiguration der Blöcke und Durchführung des Tests kein zufriedenstellendes auditives Feedback erzeugen.

Es gab ein Problem, das trotz mehrerer Versuche nicht behoben werden konnte.

Die genaue Ursache für das Versagen des Tests konnte nicht identifiziert werden, da wir verschiedene Konfigurationen ausprobiert haben, ohne Erfolg zu erzielen.

Ein Bild, das Text, Screenshot, Reihe, Zahl enthält.

Automatisch generierte Beschreibung

# Fazit:

Der Versuch, den Dopplereffekt durch auditives Feedback zu visualisieren, war leider nicht erfolgreich. Trotz der Verwendung geeigneter Blöcke und sorgfältiger Konfiguration konnten wir das gewünschte Ergebnis nicht erzielen. Es bleibt unklar, welche spezifischen Probleme dazu geführt haben, dass das auditive Feedback nicht wie erwartet funktioniert hat.