Realisierung eines Passiv-Radarsystems basierend auf Software-Defined-Radio-Technologie

Bachelor-Thesis

zur Erlangung des akademischen Grades B.Sc.

Fabian Dittié

[redacted]

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Design, Medien und Information
Department Medientechnik

Erstprüfer: Prof. Dr.-Ing. Jan Mietzner

Zweitprüfer: Dipl.-Ing. Reinhard Breuer

Hamburg, 21. 06. 2021

Inhaltsverzeichnis

Abstract

Radar technologies raised attention and gained a new scope of purpose over the past years. While bistatic radar is developed even further in the military sector to detect stealth targets, radar finds a wide variety of additional usage in navigation and transportation as well. Today we can build inexpensive receivers with the size of an USB flash drive reaching from 20 MHz to 1800 MHz. Therefore, it is possible to receive local broadcast signals with appropriate aerials to save and process the signals in the software to implement a passiv radar system in theory. This thesis developed a passiv radar based on Software-Defined-Radio (SDR) receivers from the company Nooelec with implementation in the MATLAB software environment to detect airplanes. Furthermore, this thesis is an introduction to SDR and passiv radar technology for future students and constitutes a foundation to develop a SDR based passiv radar.

Zusammenfassung

Radartechnologien gewannen mit den Jahren an Aufmerksamkeit und Aufgabenfeldern. Während im Militärsektor das bistatische Radar zur Ortung von getarnten Objekten weiterentwickelt wird, findet Radar auch unzählige Anwendungen in der Navigation und im Transportwesen. Wir sind heute in der Lage preiswerte Empfänger in der Größe eines USB-Sticks herzustellen, die Frequenzen von 20 MHz bis 1800 MHz verarbeiten können. Dadurch ist es theoretisch möglich mit geeigneten Antennen lokale Sendesignale zu empfangen, die durch Software gespeichert und verarbeitet werden, um ein Passiv-Radarsystem zu realisieren. Diese Thesis dokumentiert die Realisierung eines Passiv-Radars, mit Software-Defined-Radio (SDR) Empfängern der Firma Nooelec, unter der Einbindung in die MATLAB-Softwareumgebung, um Flugzeuge lokalisieren zu können. Darüber hinaus soll sie eine Einführung in die SDR-und Passiv-Radartechnik für zukünftige Studierende sein und eine Grundlage zur Weiterentwicklung eines auf SDR basierenden Passiv-Radars darstellen.

1 Einführung

Die Radartechnik wird in drei Typen klassifiziert: monostatisches, bistatisches und multistatisches Radar [1, S.1-13]. Wenn der Sender und der Empfänger an gleicher Stelle positioniert sind, oder in einem System im Multiplexingverfahren über eine gemeinsame Antenne senden und empfangen, wird ein monostatisches Radar realisiert [2, S.22]. Das monostatische Radar hat den Vorteil der Zeitsynchronisation zwischen Sender und Empfänger, da beide in einem kohärenten System eingebunden werden [3].
Beim bistatischen Radar hingegen sind Sender und Empfänger lokal voneinander getrennt [2, S.22]. Durch den Entfernungsabstand von Sender und Empfänger zum Ziel entsteht eine geometrische Beziehung zwischen dem Sendesignal und den reflektierten Echosignalen gegenüber dem Empfänger, wodurch ein größerer und komplexerer Überwachungsraum entsteht [4, S.1-4]. Übergeordnet werden Radarsysteme mit mehreren voneinander getrennt liegenden Antennen, die als Sender oder Empfänger eingesetzt werden, als multistatisches Radar bezeichnet. Dabei gibt es sowohl Systeme mit mehreren Sendern und einem Empfänger, als auch Systeme die umgekehrt mit nur einem Sender und mehreren Empfängern aufgebaut sind [5].

Die Radarklassifizierungen lassen sich darüber hinaus in ihre Funktionen als aktives oder passives Radar einteilen. Das aktive Radar besitzt eine eigene Sendeantenne, über die das generierte Sendesignal in Richtung des Überwachungsraumes aktiv gesendet wird. Entgegen dazu besitzt das Passiv-Radar keinen aktiven Sender, sondern bedient sich der bereits vorhanden Sendesignale aus externen Quellen. Das monostatische Radar ist dementsprechend immer ein Aktiv-Radar, während bistatische, bzw. multistatische Radarsysteme sowohl aktiv, als auch passiv sein können [4, S.1-4] [6, S.1-10].

2 Passiv-Radartheorie

2.1 Bistatisches Passiv-Radar

In dieser Thesis wird ein bistatisches Passiv-Radar (BPR) realisiert. Wie einführend erklärt, handelt es sich dabei um ein Radarsystem, bei dem Sender und Empfänger von einander getrennt sind und kein aktiver Sender im System implementiert ist.
Der IEEE Standard 686-1997 definiert das bistatische Radar wie folgt:
„A radar using antennas for transmission and reception at sufficiently different locations that the angles or ranges from those locations to the target are significantly different“ [7]. Von diesem Standard ausgehend muss die Entfernung zwischen Sender und Empfänger nur „*ausreichend*“ und „*signifikant*“ erkennbar sein und ist nicht genauer definiert. Wenn von dem System selber kein Sendesignal zur Erzeugung von Echosignalen generiert wird, müssen bereits vorhandene Signale verwendet werden. In Hamburg und Umgebung, können in Abhängigkeit von der Lokalität des Überwachungsraumes und des technischen Vermögens des Empfängers, kommerzielle Sendesignale im Bereich von u.a. Ultrakurzwellen-Rundfunk (UKW), Digital Audio Broadcasting (DAB) und Digital Video Broadcasting (DVB-T) für das Passiv-Radar genutzt werden. Dabei werden die Reflexionen des bekannten Sendesignals vom Flugobjekt in Bezug auf ihren Dopplerversatz ausgewertet, um die Bewegungen im Überwachungsraum zu erkennen. Die Distanzbeziehungen zwischen dem Sender, Zielobjekt und dem Empfänger spielen bei dem bistatischen Passiv-Radar eine übergeordnete Rolle.

2.2 Passiv-Radar Geometrie

Die Haupteigenschaft eines Passiv-Radars liegt in seiner Radargeometrie. In Abb. 2.1 sehen Sie die Geometrie des monostatischen und bistatischen Radars in einem einfachen Modell gegenüber gestellt. Beim monostatischen Aktiv-Radar wird die Distanzbeziehungen in der Strecke R_T vom Sender zum Zielobjekt und in der rückläufigen Strecke R_R dargestellt. Beim bistatischen Radar lässt sich zusätzlich der Direktweg zwischen dem Sender Tx und dem Empfänger Rx anhand der Radarbasis L erkennen. Der bistatische Winkel β beschreibt den Winkel, zwischen dem auf das Zielobjekt eintreffende Sendesignal und dem zurückgeworfenen Echosignal. Aufgrund der fehlenden Radarbasis und dem fehlenden bistatischen Winkels wird das monostatische Radar in der Radartheorie ohne Radargeometrie charakterisiert [4, S.1-3; S.9].

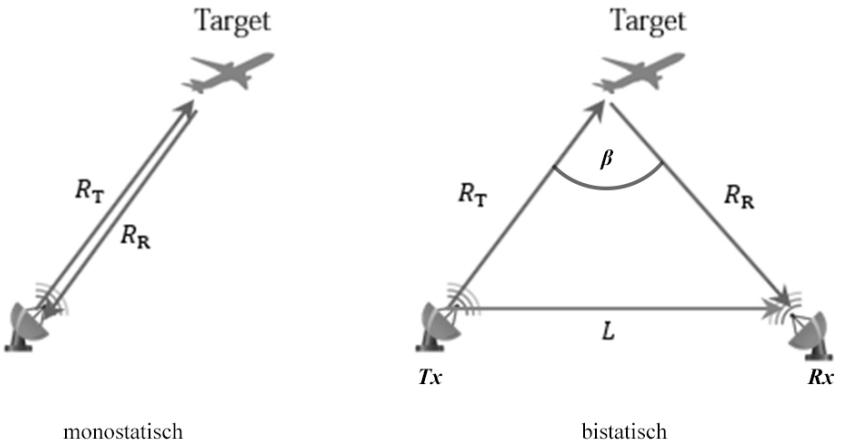


Abbildung 2.1: Geometrischer Aufbau eines monostatischen und bistatischen Radars [8].

In Bezugnahme des kartesischen Koordinatensystems besitzt das Zielobjekt, Tx und Rx jeweils eine x-, y-, und z-Koordinate. Somit können die Distanzen R_T und R_R geometrisch wie folgt ermittelt werden [4, S.10]:

$$R_T(t) = \sqrt{(x_{Ziel}(t) - x_{Tx})^2 + (y_{Ziel}(t) - y_{Tx})^2 + (z_{Ziel}(t) - z_{Tx})^2}. \tag{2.1}$$

$$R_R(t) = \sqrt{(x_{Ziel}(t) - x_{Rx})^2 + (y_{Ziel}(t) - y_{Rx})^2 + (z_{Ziel}(t) - z_{Rx})^2}. \tag{2.2}$$

Beim BPR stellt Tx eine örtliche Sendeanlage dar, deren Signale für das Passiv-Radarsystem geeignet sind. Rx beinhaltet eine Referenzantenne und mindestens eine Beobachtungsantenne. Die Referenzantenne empfängt das Direktsignal der Sendeanlage, damit das Sendesignal dem System bekannt ist. Dieses Signal wird als Referenzsignal bezeichnet. Die Beobachtungsantenne empfängt die Reflexionen des Sendesignals vom Zielobjekt. Dieses Signal wird als Echosignal bezeichnet.
Die wichtigste Größe des BPR ist die bistatische Entfernung R, welche die Differenz zwischen der Entfernung von Sender-Zielobjekt-Empfänger und der Größe der Basis L ist [4, S.9-10].

$$R(t) = R_T(t) + R_R(t) - L \tag{2.3}$$

Die bistatische Entfernung lässt sich auch als zeitlicher Versatz zwischen dem Echosignal und dem Referenzsignal ausdrücken[4, S.11]:

$$R = c \cdot \tau. \tag{2.4}$$

Lichtgeschwindigkeit $c = 3 \cdot 10^8$ $[m/s]$

Die Bedeutung der bistatischen Entfernung enthält die Dimensionierung des Bereichs der Auflösung in der Entfernung von R $[m]$. Eine konkrete Aussage über die Auflösung der Entfernung wird über die bistatische Entfernungsauflösung ΔR getroffen, die sich auf den Zeitversatz zwischen Referenzsignal und Echosignal τ aus Gleichung 2.4 stützt. Die Länge der zeitlichen Verzögerung $\Delta\tau$ ist umgekehrt proportional zur Bandbreite B des Signals und kann mit $\Delta\tau = \frac{1}{B}$ beschrieben werden [4, S.13]. Aus diesem Zusammenhang ergibt sich für die bistatische Entfernungsauflösung:

$$\Delta R = c \cdot \Delta\tau = \frac{c}{B}. \tag{2.5}$$

ΔR bedeutet, dass ein Minimum von R $[m]$ an Bewegung vom Zielobjekt zwischen zwei Messwerten gemessen werden kann. Beim Entwickeln eines Passiv-Radarsystems wird angestrebt, ΔR möglichst klein zu halten. Sollte die bistatische Entfernungsauflösung mehrere hundert Meter oder Kilometer betragen, wäre die Auflösung für das in dieser Thesis angestrebte System zu gering.

Bei der Messung von bewegten Zielobjekten ist eine weitere, wichtige Größe die bistatische Geschwindigkeit V. Sie ist definiert als zeitliche Ableitung der bistatischen Entfernung R [4, S.12].

$$V(t) = \frac{dR(t)}{dt} \tag{2.6}$$

Anders ausgedrückt, kann V als Produkt der Wellenlänge λ und dem Dopplerversatz f_d errechnet werden [4, S.12].

$$V = -\lambda \cdot f_d. \tag{2.7}$$

Wellenlänge $\lambda = \frac{c}{f_c}\ [m]$
Trägerfrequenz $f_c\ [Hz]$
Dopplerfrequenz$f_d = f_c \cdot 2 \cdot \frac{v}{c}\ [Hz]$
Geschwindigkeit des Zielobjektes $v\ [m/s]$

Die bistatische Geschwindigkeit V hängt sowohl von der Entfernung, als auch vom Geschwindigkeitsvektor des Zielobjektes ab und gibt Auskunft darüber, wie sich das Zielobjekt entlang der Azimutachse und auf Entfernung im Bezug auf die Empfängerposition R_x bewegt.

Bei der bistatischen Geschwindigkeitsauflösung ΔV wird die Dopplerfrequenzauflösung Δf_d als umgekehrt proportional zur Länge des Integrationsfensters T definiert und bildet somit $\Delta f_d = \frac{1}{T}$. In Bezug auf V und die Gleichung 2.7 ergibt sich für die bistatische Geschwindigkeitsauflösung [4, S.13]:

$$\Delta V = \lambda \cdot \Delta f_d = \frac{\lambda}{T}. \tag{2.8}$$

An der bistatischen Entfernungs- und Geschwindigkeitsauflösung ist zu erkennen, dass die Funktionalität eines Passiv-Radars bedingt von externen Faktoren abhängt.

Sowohl die Signalbandbreite B aus Gleichung 2.4, als auch die Wellenlänge λ aus Gleichung 2.7 sind Attribute, die im externen Sendesignal stecken, auf das kein Einfluss genommen werden kann. Lediglich die Integrationszeit T kann verbessert werden, was direkten Einfluss auf den Signal-Rausch-Abstand (engl.: signal-to-noise ratio; SNR) hat. Auf den SNR wird in Abschnitt 2.4 genauer eingegangen.

2.3 Bistatische Entfernungsgleichung

Eine Größe mit Aussagekraft über die Reichweite des Radars ist die bistatische Entfernungsgleichung. Sie bestimmt die maximal mögliche Entfernung in der Echosignale des Zielobjektes vom Radarsystem erkannt werden können. Das in dieser Thesis realisierte System benötigt nur wenige Kilometer und muss nicht den Grenzwert der bistatischen Entfernungsgleichung ausreizen. Für alle anderen Systeme, die Zielobjekte in signifikanter Entfernung messen oder dezentral von lokalen Sendesignalen stationiert sind, ist die bistatische Entfernungsgleichung relevant.

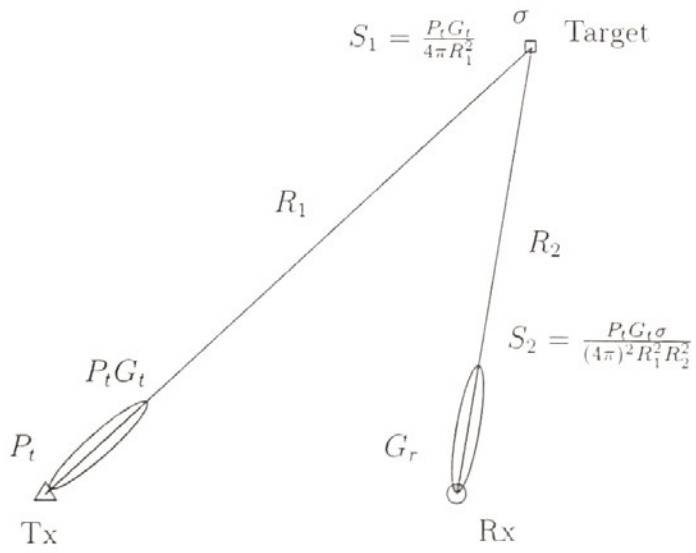


Abbildung 2.2: Bistatische Umgebung der Entfernungsgleichung [4, S.19].

Die Abb. 2.2 ist eine detaillierte Darstellung der Abb. 2.1, in der Eigenschaften der Antennen und des Zielobjektes berücksichtigt werden. Die Sendeanlage Tx sendet ihr Signal mit der Sendeleistung P_t. Die Richtwirkung der Sendeantenne wird durch G_t

beschrieben. Die Richtwirkung der Sendeantenne verstärkt die Leistungsdichte S_1 des Signals in Richtung der maximalen Abstrahlcharakteristik. Wenn sich das Zielobjekt in einer Entfernung von R_1 zum Sender befindet, kann die Leistungsdichte errechnet werden [4, S.19].

$$S_1 = \frac{P_t G_t}{4\pi R_1^2} \tag{2.9}$$

Sollte die Sendeantenne isotrop bzw. omnidirektional, in alle Richtungen mit der gleichen Sendeleistung P_t abstrahlen, wird die Richtwirkung $G_t = 1$ gesetzt. So wird in Gleichung 2.9 die Sendeleistung P_t durch die Kugeloberfläche $4\pi r^2$ geteilt, wenn von R_1 als Radius ausgegangen wird.
Wenn das Sendesignal mit der Leistungsdichte S_1 am Zielobjekt eintrifft, definiert der Radarquerschnitt (engl.: radar cross section; RCS) σ, wie viel Leistung von dem Zielobjekt reflektiert wird. Der Radarquerschnitt ist abhängig von der effektiven Reflexionsfläche des Zielobjektes. Im idealen Fall wird die reflektierte Leistungsdichte von einem kugelförmigen isotropen Reflektor S_s betrachtet. Dazu wird die einfallende Leistungsdichte S_i in Bezug genommen. RCS wird als Verhältnis zwischen der reflektierten Leistungsdichte des Isotropreflektors S_s und der einfallenden Leistungsdichte S_i definiert [4, S.20].

$$\sigma = \lim_{R \to \infty} 4\pi r^2 \frac{S_s}{S_i} \tag{2.10}$$

Die Kugeloberfläche $4\pi r^2$ liegt der theoretischen Annahme des isotropen Kugelreflektors zugrunde. Für den Radarquerschnitt ist die Fläche auf der Kugel relevant, auf die das Sendesignal eintrifft und reflektiert wird. In der praktischen Anwendung im bistatischen Passiv Radar muss berücksichtigt werden, dass sich S_i und S_s in unterschiedliche Richtungen ausbreiten. Der Radarquerschnitt σ wird in $[m^2]$ angegeben. In dem Bezugsbeispiel aus Abb. 2.2 wird die Leistungsdichte S_1 um σ und der Strecke des reflektierten Signals R_2 erweitert, um die reflektierte Leistungsdichte S_2 bilden zu können.

$$S_2 = \frac{P_t G_t \sigma}{(4\pi)^2 R_1^2 R_2^2}. \tag{2.11}$$

Die empfangene Leistung P_r ist abhängig von der reflektierten Leistungsdichte S_2 und der Öffnungsweite A_{ef} der Empfängerantenne.

$$P_r = \frac{P_t G_t \sigma A_{ef}}{(4\pi)^2 R_1^2 R_2^2}. \tag{2.12}$$

Die Öffnungsweite A_{ef} ist abhängig von der Richtwirkung G_r der Empfängerantenne und der Wellenlänge λ.

$$A_{ef} = \frac{G_r \lambda^2}{4\pi} \tag{2.13}$$

Die empfangene Leistung P_r kann nach dem Einsetzen von 2.13 in 2.12 folgendermaßen berechnet werden:

$$P_r = \frac{P_t G_t G_r \sigma \lambda^2}{(4\pi)^3 R_1^2 R_2^2}. \tag{2.14}$$

Nur wenn die empfangene Leistung größer ist als die thermische Rauschleistung des Empfängers $P_r > P_n$ ist es möglich das Signal zu Verarbeiten [4, S.21].

$$P_n = k_B T_0 B_r \tag{2.15}$$

Boltzmann-Konstante $k_B = 1,380649 \cdot 1o^{-23} \ \left[\frac{J}{K}\right]$

Rauschtemperatur $T_0 = T_{ref}(10^{\frac{N_f}{10}} - 1) \ [K]$

Empfängerbandbreite $B_r \ [Hz]$

Die Boltzmann-Konstante ist das Verhältnis zwischen der absoluten Temperatur und der kinetischen Energie einzelner Moleküle. Mit steigender Temperatur erhöht sich die kinetische Energie.
Die Rauschtemperatur ist von dem Rauschfaktor N_f und der Referenz-Temperaturkonstante $T_0 = 290 \ K$ abhängig.

Das thermische Rauschen ist vor allem in höheren Frequenzbändern (ab 1 GHz) entscheidend. In niedrigeren Frequenzbändern, wie UKW und DVB-T, ist es wahrscheinlicher, dass das Rauschen von externen, elektrischen Störfeldern, das thermische Rauschen verdeckt.
Durch die empfangene Leistung P_r kann abschließend die Entfernungsgleichung R_m aufgestellt werden.

$$R_m = \left(\frac{P_t G_t^2 G_r^2 \sigma \lambda^2}{(4\pi)^2 P_r} \right)^{\frac{1}{4}} \tag{2.16}$$

2.4 Rausch-Signal-Abstand

Der SNR ist das Verhältnis zwischen der Leistung des Echosignals und der Rauschleistung. Zur Optimierung des Radarsystems wird der SNR so groß wie möglich gewählt. Der SNR ist neben den Signaleigenschaften vom Empfangssignal und Rauschsignal, von der zeitlichen Dauer der Betrachtung, der Integrationszeit, abhängig. In der digitalen Signalverarbeitung der Sende- und Echosignale bezieht sich dies auf die Anzahl der betrachteten Samples N, die das kohärente Verarbeitungsintervall definieren (engl.: coherent processing interval; CPI) [4, S.21-22].

$$N = B \cdot T \tag{2.17}$$

Bandbreite $B = \frac{1}{f_s}\ [Hz]$

Abtastfrequenz $f_s\ [Hz]$

Integrationszeit $T\ [s]$

Aus der Gleichung 2.17 kann ein Bezug zur bistatischen Entfernungsauflösung aus Gleichung 2.5 hergestellt werden. Mit Erhöhung der Bandbreite B erhöht sich das

CPI, daraus ableitend erhöht sich der SNR und die bistatische Entfernungsauflösung ΔR verringert sich, was bedeutet, dass das Radarsystem in der Lage ist, kleinere Distanzunterschiede vom Zielobjekt zwischen zwei Messpunkten zu erkennen. Daraus resultiert, dass die Bandbreite eine essenzielle Größe in der Passiv-Radartechnik darstellt, die direkten Einfluss auf die Genauigkeit und Qualität des Radars nimmt. In Abb. 2.3 wird anhand eines frequenzmodulierten Radiosignals (FM-Signal) gezeigt, dass sich mit steigender Integrationszeit T der SNR erhöht.

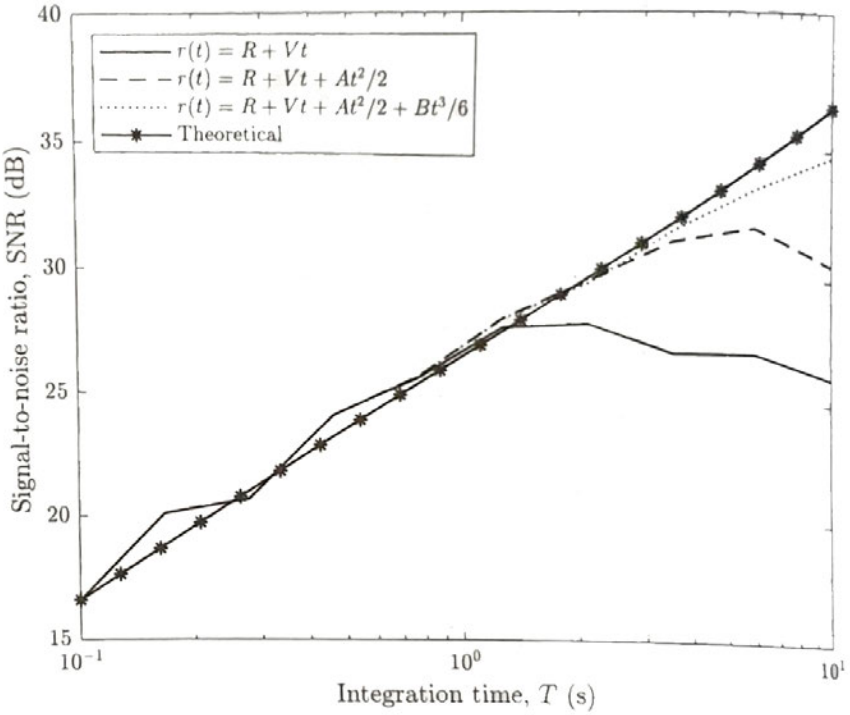


Abbildung 2.3: SNR eines reellen FM-Echosignals in Abhängigkeit der Intergrationszeit bei verschiedenen Bewegungen $r(t)$ [4, S.160].

Die Abbildung zeigt allerdings auch, dass der Zusammenhang zwischen Integrationszeit und SNR nicht idealerweise linear steigend verläuft. Er ist abhängig von der Bewegung und Position des Zielobjektes. Parameter, wie die Richtwirkung der Sendeantenne und der Radarquerschnitt, sind von dem Bewegungsverhalten des Zielobjektes abhängig und müssen berücksichtigt werden. Deshalb lässt sich der SNR ebenfalls in Bezugnahme dieser Parameter ausdrücken.

$$SNR = \frac{P_t G_t G_r \sigma \lambda^2 BT}{(4\pi)^3 R_1^2 R_2^2 P_n L} \tag{2.18}$$

Dabei werden sämtliche Signalverluste u.a. durch die Sendestrecke, der Signalverarbeitung oder die Antennenrichtwirkung im Term L berücksichtigt.

2.5 Direktsignal-Interferenz

Passiv-Radare haben die Eigenschaft, dass sich das Direktsignal von der Sendeanlage nicht nur im Referenzsignal, sondern auch im Echosignal wiederfindet. Häufig überdeckt dabei das Direktsignal die relevanten Reflexionen im Echosignal. Diese Störung ist als Direktsignal-Interferenz (engl.: direct path interference; DPI) bekannt. Die Leistung des Direktsignals im Echosignal wird durch P_{DPI} beschrieben [4, S.31].

$$P_{DPI} = \frac{P_t G_t G_r(\phi_{tx}) \lambda^2}{(4\pi R_b)^2} \tag{2.19}$$

Die Richtwirkung G_r der Empfängerantenne in Gleichung 2.19 steht in Abhängigkeit zu dem Winkel ϕ_{tx} von der Empfängerantenne zur Sendeantenne. Die Entfernung R_b ist die in Abschnitt 2.2 eingeführte Radarbasis.

Unter Berücksichtigung der Positionsbeziehungen von Sender, Empfänger und Zielobjekt und den Bezugswinkeln zur Sendeanlage ϕ_{tx} und Zielobjekt ϕ_{target} lassen sich die Referenz- und Echosignale mathematisch beschreiben [4, S.34]. In Abb. 2.4 werden hierzu die Richtwirkungen bzw. die Strahlungscharakteristik des Referenzsignals $A_{ref}(\phi)$ und des Echosignals $A_{surv}(\phi)$ mit ihren Bezugswinkeln dargestellt.

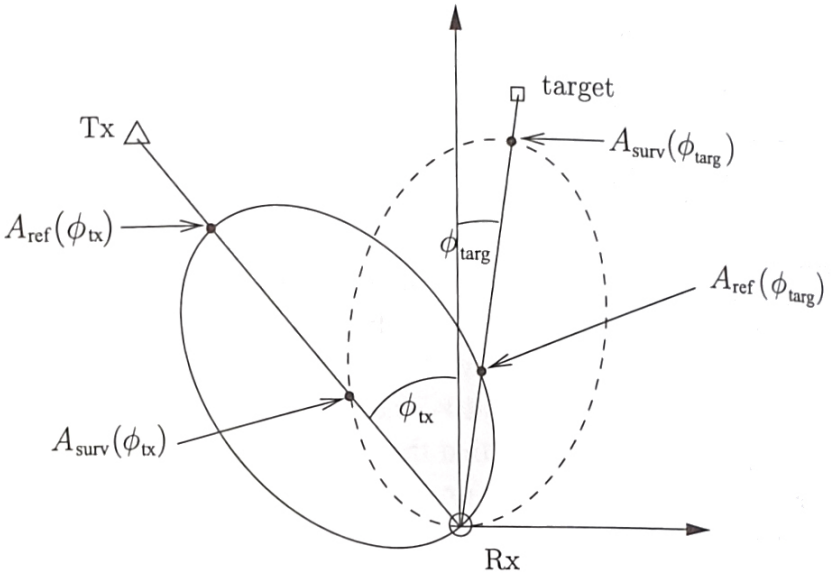


Abbildung 2.4: Bistatische Umgebung mit Bezugswinkeln [4, S.34].

Im Referenzsignal $x_r(t)$ steckt das Direktsignal $x_{tx}(t)$ verrechnet mit der Richtwirkung der Referenzantenne in Richtung des Senders und das Echosignal $x_{targ}(t)$ mit der Richtwirkung in Richtung vom Zielobjekt [4, S.34].

$$x_r(t) = A_{ref}(\phi_{tx})x_{tx}(t) + A_{ref}(\phi_{targ})x_{targ}(t) \tag{2.20}$$

Für das Echosignal $x_e(t)$ gilt die Gleichung 2.20 ebenfalls. Allerdings müssen die Richtwirkungen der Beobachtungsantenne verwendet werden.

$$x_e(t) = A_{surv}(\phi_{tx})x_{tx}(t) + A_{surv}(\phi_{targ})x_{targ}(t) \tag{2.21}$$

Richtwirkung Referenzantenne A_{ref} $[V]$ oder $[dB]$

Richtwirkung Beobachtungsantenne A_{surv} $[V]$ oder $[dB]$

Die Direktsignal-Interferenzen müssen aus dem Echosignal entfernt werden, um ein neues Echosignal $x'_e(t)$ zu erzeugen. Dabei wird von dem Echosignal $x_e(t)$ das Produkt vom Referenzsignal $x_r(t)$ und dem Verhältnis der Richtwirkungen zum Sender abgezogen [4, S.35].

$$x'_e(t) = x_e(t) - \frac{A_{surv}(\phi_{tx})}{A_{ref}(\phi_{tx})} x_r(t) \tag{2.22}$$

Ausgeschrieben ist $x'_e(t)$:

$$x'_e(t) = A_{surv}(\phi_{targ}) x_{targ}(t) - A_{ref}(\phi_{targ}) \frac{A_{surv}(\phi_{tx})}{A_{ref}(\phi_{tx})} x_{targ}(t) = \tag{2.23}$$

$$\left(A_{surv}(\phi_{targ}) - A_{ref}(\phi_{targ}) \frac{A_{surv}(\phi_{tx})}{A_{ref}(\phi_{tx})} \right) \cdot x_{targ}(t)$$

Wie in Gleichung 2.23 erkennbar, wird das ursprüngliche Echosignal $x_{targ}(t)$ durch einen neuen Faktor verändert, wenn zur DPI-Entfernung das Referenzsignal wie in 2.22 abgezogen wird.

2.6 Sendesignaleigenschaften

Passiv-Radarsysteme hängen stark vom Sendesignal ab. Die Wahl des Frequenzbands und des Signals kann von mehreren Faktoren abhängig sein. Das primäre Auswahlkriterium ist die lokale Verfügbarkeit der Sendesignale. In einigen Szenarien wird die Auswahl der Sendesignale durch den Mangel an verfügbaren, kommerziellen Signalen mit ausreichender Sendeleistung vorab bestimmt. Auch das Modulationsverfahren des Signals oder die Hardwareeigenschaften vom Sender sowie vom Empfänger können ausschlaggebend für die Wahl des Sendesignals sein. Final ist die Entscheidung abhängig von dem Einsatzgebiet des Signals. Ob das Radar u.a. zur Entfernungsmessung, Geschwindigkeitsmessung, Zielortung oder Objektanalyse eingesetzt wird, welcher Überwachungsraum abgedeckt werden muss und in welcher Richtung dieser liegt, ist hierbei relevant.

In dieser Thesis werden Radiosignale als Sendesignale gewählt. Sie eignen sich für das System aufgrund ihrer Präsenz im Raum Hamburg. Radiosignale befinden sich im UKW-Bereich, im Frequenzband bei 87,5-108 MHz und werden frequenzmoduliert, weshalb Sie auch FM-Signale genannt werden. Die Bandbreite eines FM-Signals beträgt 150 kHz. Die einzelnen FM-Signale liegen dabei 200 kHz auseinander, um Signalüberlappungen zu vermeiden. Wie das Spektrum des FM-Signals aufgeteilt ist, lässt sich in Abbildung 2.5 erkennen.

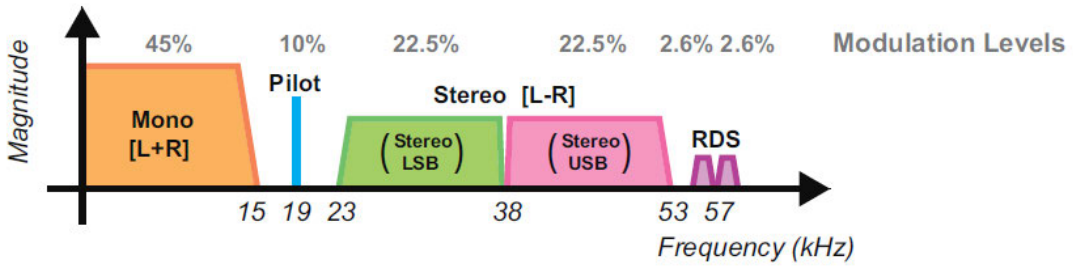


Abbildung 2.5: Einseitiges Spektrum FM-Signals im Basisband [10, S.363].

Das Mono-Signal setzt sich aus dem linken und rechten Audiokanal zusammen und ist auf 15 kHz begrenzt. Das Pilot-Signal bei 19 kHz dient als Indikator für den Demodulator am Empfänger. Wenn das Pilot-Signal vorhanden ist, weiß der Empfänger, dass im FM-Signal ein Stereo-Signal existiert, welches demoduliert werden muss. Das Stereo-Signal liegt auf der Unterträgerfrequenz von 38 kHz. Bei 57 kHz liegt das RDS-Signal (engl.: Radio Data System) und beinhaltet die Metadaten wie u.a. Interpret, Musiktitel, Albumtitel und Radiostation [10, S.363].
Der größte Nachteil bei FM-Signalen sind die Differenzen in der Bandbreite, die vom Signalinhalt abhängig sind.

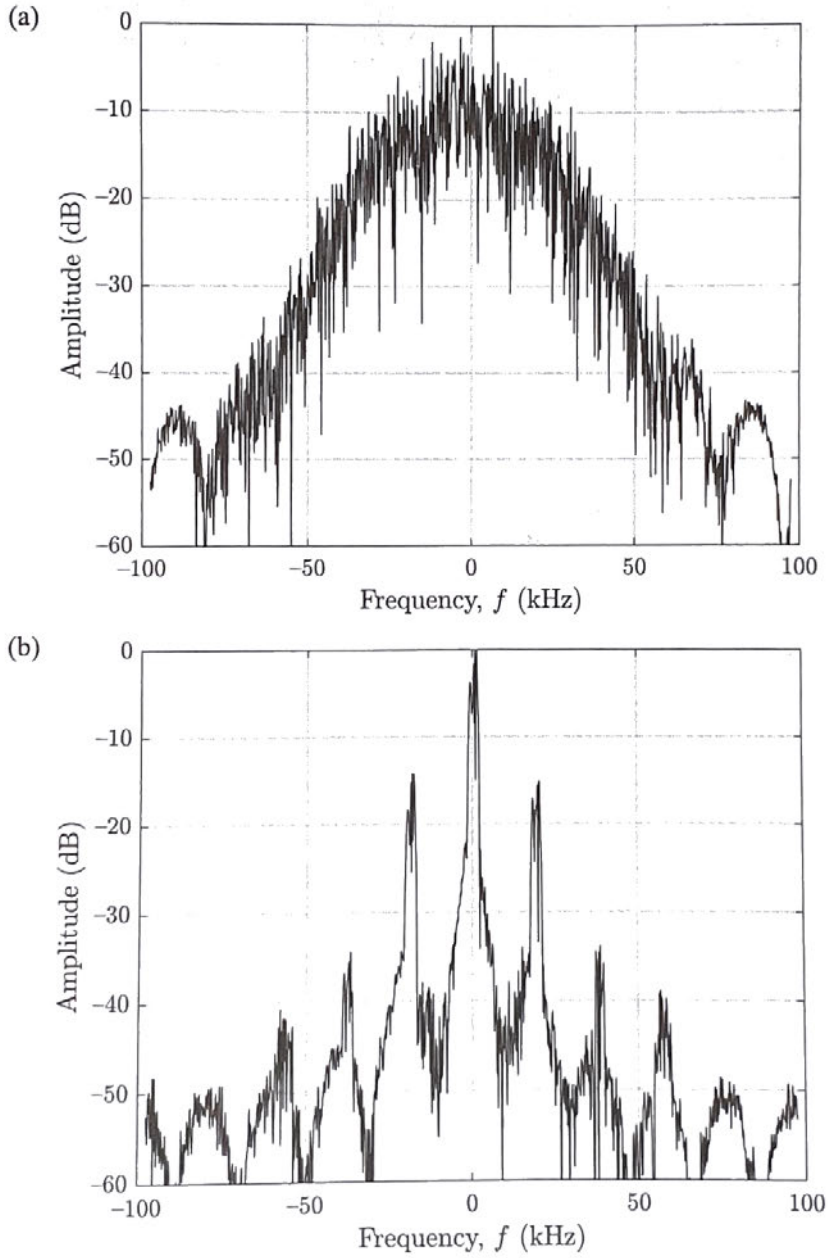


Abbildung 2.6: FM-Spektrum von (a) Pop-Musik und (b) Sprache [4, S.55].

Die Unterschiede werden in Abb. 2.6 gezeigt. Während das Signal mit Pop-Musik ein kontinuierliches Spektrum liefert, besteht das Sprachsignal aus Spitzen, an denen ein Spektrum erfasst wird und Täler zwischen den Worten und Silben, in denen keine Frequenzen mit signifikanten Signalpegel gemessen werden konnten.

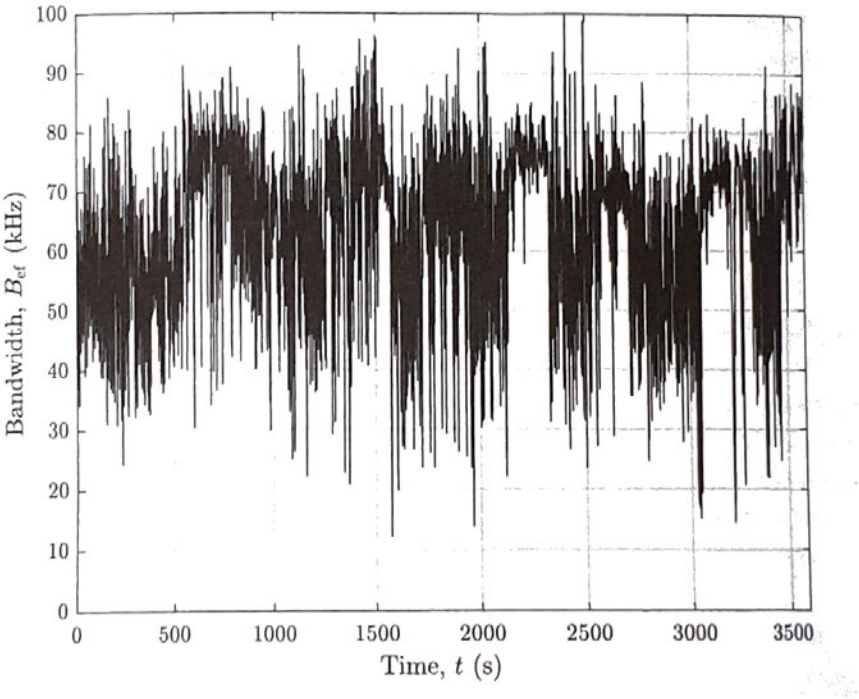


Abbildung 2.7: Bandbreite eines FM-Signals gemessen über eine Dauer von 1 Stunde [4, S.56].

Dies hat zur Folge, dass die Bandbreite, wie in Abb. 2.7 zu sehen, mit der Zeit stark variiert. In Gleichung 2.17 und 2.5 wurde festgehalten, dass die Bandbreite ausschlaggebend für den SNR, der die Fähigkeit des Radarsystems Objekte zu erkennen beschreibt und für die bistatische Entfernungsausflösung ist, die definiert, in welchen Entfernungsabständen eine Bewegung des Zielobjektes entdeckt wird. Im Falle eines Signalinhalts mit einem diskontinuierlichen Spektrum bzw. einer geringen Bandbreite, könnte der SNR zu weit sinken, sodass Echosignale nicht mehr zu erkennen sind. Die bistatische Entfernungsauflösung könnte dabei so weit sinken, bis die Bewegungsabstände zwischen zwei Messpunkten zu groß und somit nicht mehr für die Radaranwendung zu gebrauchen sind.

2.7 Ambiguitätsfunktion

Die Ambiguitätsfunktion (engl.: ambiguity function; AF) beschreibt die Verwandtschaften zwischen Signalen und ihren in Zeit und Frequenz versetzten Versionen [4, S.37]. Mathematisch wird die Ambiguitätsfunktion durch eine Korrelation beschrieben.

$$\psi(\tau, f_d) = \int_{-\infty}^{\infty} x(t)x^*(t-\tau)e^{j2\pi f_d t}dt \tag{2.24}$$

Um die Korrelation verständlicher auszudrücken und um sie im Kapitel 8.1 leichter programmieren zu können, sollte man zunächst die mathematische Faltung betrachten [9, S.15-16].

$$s(t) * g(t) = conv\{s, g\}(t) = \int_{-\infty}^{\infty} s(t)g(t-\tau)dt \tag{2.25}$$

Nach dem Faltungstheorem ist eine Faltung im Zeitbereich eine Multiplikation im Frequenzbereich [9, S.133-134]. Es gilt: $s(t) * g(t) = S(f) \cdot G(f)$.
Zeitsignale werden mit der Fouriertransformation in den Frequenzbereich überführt [9, S.66-67].

$$S(f) = FT\{s\}(f) = \int_{-\infty}^{\infty} s(t) \cdot e^{-j2\pi ft}dt \tag{2.26}$$

Mit dem Faltungstheorem und der Fouriertransformation aus 2.26 kann die Faltung der Beispielsignale $s(t)$ und $g(t)$ als Multiplikation vereinfacht ausgedrückt werden.

$$FT\{conv(s, g)\} = FT\{s\} \cdot FT\{g\} \tag{2.27}$$

Die Korrelation ist nach Definition eine in Zeit umgekehrte Faltung [9, S.206-207].

$$\rho(s, s) = corr\{s, g\}(t) = \int_{-\infty}^{\infty} s(t)g(t+\tau)dt \tag{2.28}$$

Damit ein Vorzeichenwechsel und damit eine Umkehrung in der Zeit passieren kann, muss das komplex konjugierte Signal von $g(t+\tau)$ gebildet werden.
Es gilt: $g^*(t+\tau) = g(t-\tau)$.
Daraus kann die vereinfachte Korrelation gebildet werden.

$$FT\{corr(s,g)\} = FT\{s\} \cdot FT^*\{g\} \tag{2.29}$$

Zur Rückführung in den Zeitbereich muss abschließend die inverse Fouriertransformation angewendet werden [9, S.67].

$$S(t) = IFT\{S\}(t) = \int_{-\infty}^{\infty} S(f) \cdot e^{j2\pi ft} df \tag{2.30}$$

Damit kann die Berechnung der Ambiguitätsfunktion vereinfacht dargestellt werden.

$$IFT\ [FT\{corr(s,g)\}] = IFT\ [FT\{s\} \cdot FT^*\{g\}] \tag{2.31}$$

Wenn die Ambiguitätsfunktion, anstatt einem Signal und seiner Zeit verschobenen Version, zwei verschiedene Signale beurteilt, wird von der Kreuzambiguitätsfunktion (engl.: cross ambiguity function; CAF) gesprochen. Die CAF verwendet die in der Herleitung verwendete Kreuzkorrelation, bei der zwei unterschiedliche Signale verglichen werden. Die CAF wird auch für das Radarsystem dieser Thesis relevant sein und die Verwandschaft zwischen dem Referenzsignal und dem Echosignal prüfen.

Die Abbildung 2.8 zeigt die Ambiguitätsfunktion eines Rauschsignals. Rauschsignale haben die Eigenschaft der zufälligen Signalwerte. Durch die Anwendung der Autokorrelation zwischen dem Rauschsignal und seiner zeitlich verschobenen Version wird nur ein Korrelationswert am Maximum ausgegeben, wenn die maximale, zeitliche Verzögerung der beiden Signale verglichen wurde. Dies ist ein Idealfall, der in einer deutlich erkennbaren Signalspitze in beiden Dimensionen resultiert. Der Zeitversatz wird in Form der bistatischen Entfernung auf der y-Achse, der Frequenzversatz in Form der bistatischen Geschwindigkeit, auf der x-Achse angegeben.

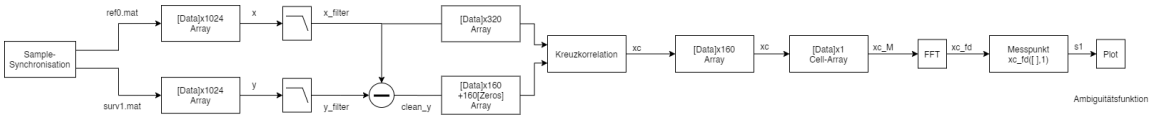


Abbildung 2.8: Ambiguitätsfunktion eines Rauschsignals in der (a) dreidimensionalen und (b) zweidimensionalen Ebene [4, S.42].

2.8 Zusammenfassung

In dem Kapitel der Passiv-Radartheorie werden die Eigenschaften eines bistatischen Passiv-Radars zusammengefasst. Dabei werden die wichtigsten Parameter zur Beschreibung der Leistungsfähigkeit und Funktionalität eines Passiv-Radars mittels der Radargeometrie eingeführt. Die Signaleigenschaften werden beschrieben und in den Bezug der Radaranwendung gesetzt. Abschließend wird die Grundidee der Signalverarbeitung für Passiv-Radare mit der Kreuzambiguitätsfunktion vorgestellt.

3 Hardware

3.1 Antennentechnik

Zur Messung des Sendesignals benötigt das bistatische passiv-Radar eine Referenzantenne für das Direktsignal und eine Beobachtungsantenne für die Echosignale aus dem Überwachungsbereich. Als Referenzantenne wird eine Teleskopantenne verwendet, die dem SDR-Empfänger in 3.4 beilag.

Abbildung 3.1: Die als Referenzantenne verwendete Teleskopantenne [11].

Die Teleskopantenne besitzt eine omnidirektionale Richtwirkung. Das bedeutet, dass sie in alle Richtungen mit gleicher Verstärkung Signale empfängt.
Damit bereits durch die Messung und die Richtwirkung der Antennen eine Trennung der Referenz- und Echosignale ermöglicht wird, sollte die Beobachtungsantenne eine direktionale Richtwirkung besitzen. Wenn beide Antennen omnidirektional gerichtet sind, ist in der Signalverarbeitung die Trennung der Echosignale von den Direkt-

signalen nicht möglich. Malanowski beschreibt die ideale Messumgebung mit zwei direktional gerichteten Antennen [4, S.10-36].
Als Beobachtungsantenne wird die UKW-Richtantenne WB205 der Firma Wittenberg verwendet.

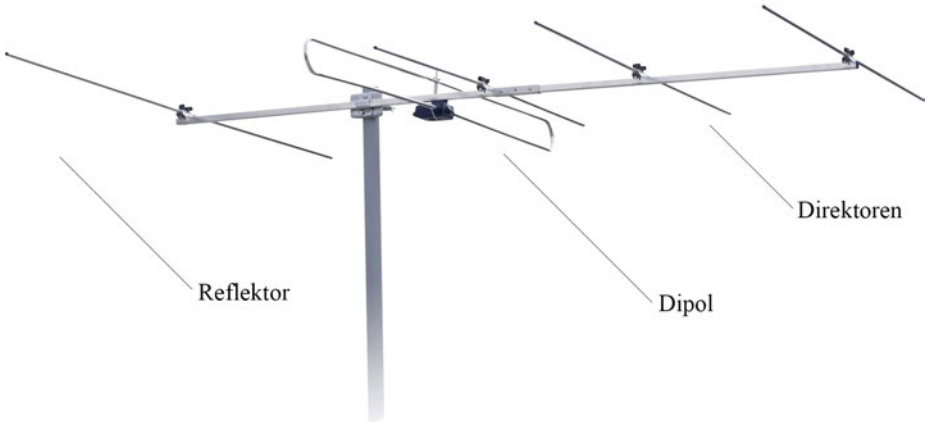


Abbildung 3.2: WB 205 von Wittenberg Antennen und Zubehör UG [12].

Wittenberg WB 205 5-Element [12]
Frequenzbereich: 87,5-108 MHz
Impedanz: 75 Ω
Antennenelemente: 5
Antennengewinn: 6,5-7,5 dBi
Länge: 1760 mm
Anschluss: F-Stecker

In Abb. 3.2 ist erkennbar, dass es sich bei der WB 205 um eine 5-Element Yagi-Uda-Antenne handelt. Sie besitzt in der Mitte den Dipol, vorne das Wellenleitersystem bestehend aus den drei Direktoren und hinter dem Dipol befindet sich der Reflektor. Bei der Yagiantenne wird in Richtung des Wellenleitersystems die Richtwirkung verstärkt, während der Reflektor die Signale in entgegengesetzter Richtung unterdrückt. Damit die Richtcharakteristik der WB 205 in der Messung der Echosignale berücksichtigt werden kann, muss ein Polardiagramm angefertigt werden.

3.2 Messung der Beobachtungsantenne

Die Hochschule für Angewandte Wissenschaften in Hamburg verfügt über ein Labor für Kommunikationstechnik, in dem die Messung der WB 205 durchgeführt wurde. Die Messkammer wurde als Faradayscher Käfig gebaut. In ihr befindet sich die R&S HL562E Ultralog Messantenne, die einen Frequenzbereich von 30 MHz bis 3 GHz hat. Ab einer Trägerfrequenz von $f_c > 200\ MHz$ besitzt die Antenne eine starke Richtwirkung. Für $f_c < 200\ MHz$ gilt die im Datenblatt aufgeführte Richtcharakteristik für 30 MHz.

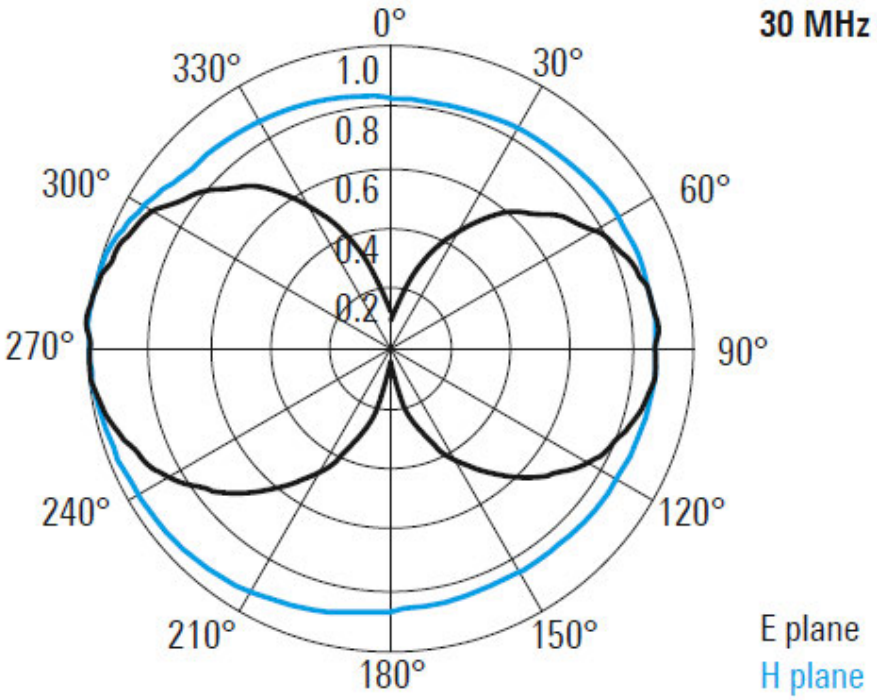


Abbildung 3.3: Polardiagramm der R&S HL562E Ultralog Antenne in der horizontalen (H plane) und vertikalen (E plane) Ebene bei 30 MHz .

Anhand der horizontalen Richtwirkung in Abb. 3.3 kann die Richtcharakteristik als omnidirektional und dementsprechend die Antenne als Rundstrahler angesehen werden.

Abbildung 3.4: Labor für Kommunikationstechnik HAW Hamburg am Berliner Tor.

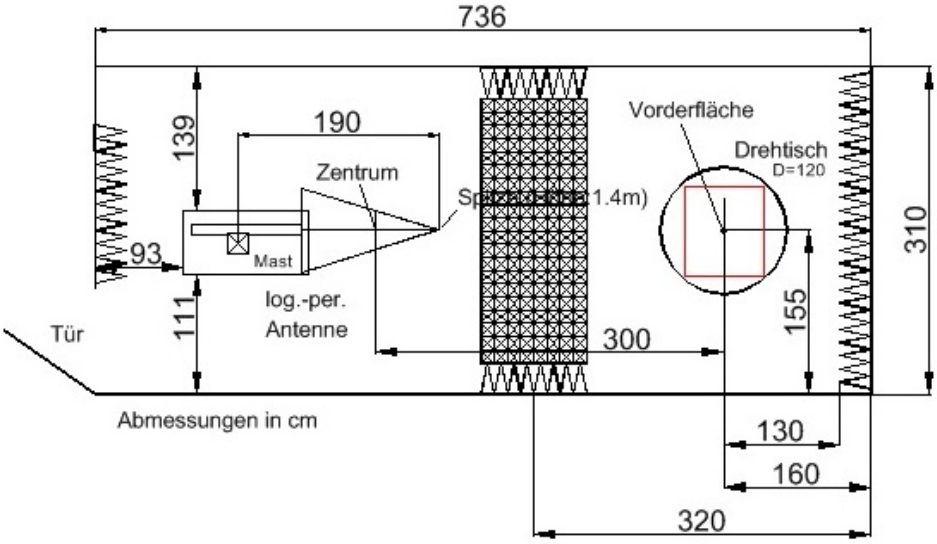


Abbildung 3.5: BricsCAD Konstruktionszeichnung des Messlabors.

Am anderen Ende der Messkammer befindet sich ein drehbarer Tisch, der mit einem Motor angesteuert wird. Für die Messung wird die WB 205 Antenne auf dem Tisch gegenüber der Messantenne platziert. Die R&S HL562E Ultralog Antenne wird als Sender und die WB 205 als Empfänger konfiguriert. Zwischen den Antennen liegt eine Sendestrecke von 3 m. In der Messkammer werden sowohl Absorber an den Wandoberflächen, als auch Holzgegenstände verwendet, deren effektive Absorptionsfläche einen Absorptionsgrad von $\alpha_0 = 0$ haben, um ungewollte Reflexionen zu verhindern.

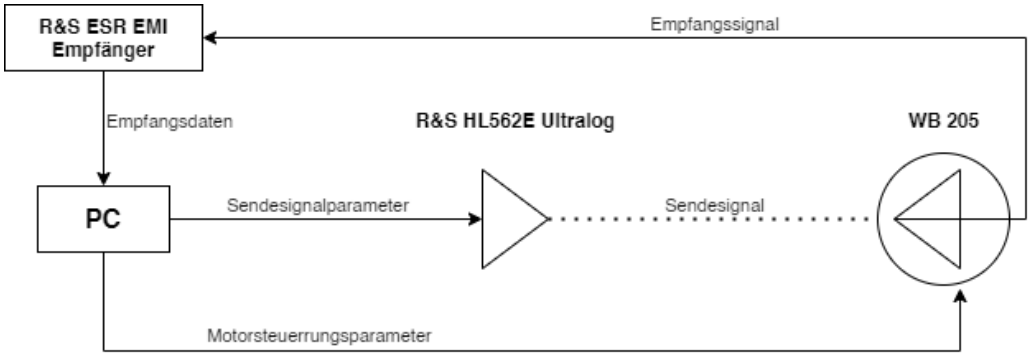


Abbildung 3.6: Versuchsaufbau zur Messung der Richtwirkung der WB 205 Antenne.

Bei der Messung der Beobachtungsantenne werden von der Messsoftware ein Signal mit den Parametern des Sendesignals an die Sendeantenne und ein Signal mit der gewünschten Motorposition an die Motorsteuerung im Tisch gesendet. Die WB 205 wird darauf in die gewünschte horizontale Position ausgerichtet und empfängt das Sendesignal in Bezug auf den eingestellten Empfangswinkel zum Sender. Das Empfangssignal wird dann zum R&S ESR EMI Receiver, ein für die Messtechnik entworfenes Signalempfängersystem, welches das Empfangssignal abbildet, auswertet und die Empfangsdaten an die Messsoftware weiter gibt [14]. Während der Messung wurde ein breitbandiges FM-Signal von 87,5 MHz bis 108 MHz als Sendesignal kontinuierlich an die WB 205 Antenne gesendet. Die WB 205 wurde dabei mit einer Auflösung von 5° auf der Azimutachse gedreht, sodass 72 Messwerte für die Betrachtung der gesamten 360° erhoben wurden. Bei jedem Messpunkt wird ein Leistungspegel und der lineare Verlauf über die Bandbreite für das Empfangssignal ermittelt.

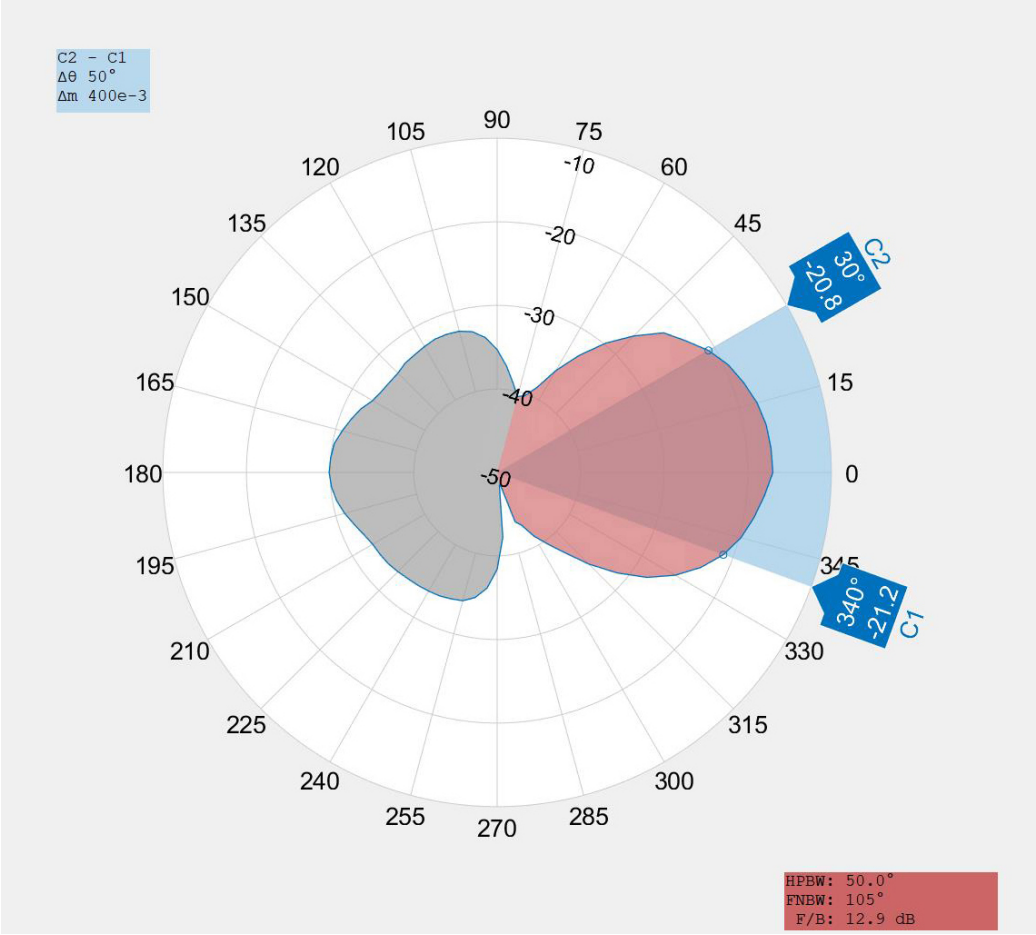


Abbildung 3.7: Polardiagramm der WB 205 Antenne auf der Azimutachse für $f_c = 103,6\ MHz$.

Das Polardiagramm wurde in MATLAB umgesetzt [15] und ist in Anlage A.1 einzusehen.
Die HPBW (engl.: Half Power Beamwidth) beschreibt die Breite der Hauptkeule (rot) bei der halben Leistung -3 dB und wird mit der Fläche zwischen C1 und C2 in Abb. 3.7 gekennzeichnet. Die FNBW (engl.: First Null Beamwidth) gibt die Breite von der 0°-Hauptachse bis zur ersten erkennbaren Nebenkeule an. Die F/B (engl.: Front to Back Ratio) gibt die Pegeldifferenz zwischen dem Maximalwert der Hauptkeule (rot) auf der 0°-Achse und dem Maximalwert der Nebenkeule (grau) auf der 180°-Achse an. Das Polardiagramm zeigt, dass die Nebenkeule im Bezug zur Hauptkeule sehr breit und unsymmetrisch ist. Bei einer höheren Auflösung von einem Messwert pro 1° anstatt 5° würde die Charakteristik deutlicher erkennbar und definierter sein. Jedoch ist festzuhalten, dass die WB 205 eine 50° breite Hauptkeule mit einer Nebenkeulen-

dämpfung von nur 13 dB besitzt. Daraus resultiert, dass die Antenne nicht mit der Nebenkeule in die Richtung der Sendeanlage positioniert werden sollte. Stattdessen bietet sich eine Ausrichtung von 75°, oder 275° zum Sender an, um die Signalverstärkung für das Direktsignal zu minimieren. Desweitern muss berücksichtigt werden, dass die Messung sich auf eine Trägerfrequenz von $f_c = 103,6\ MHz$ bezieht.

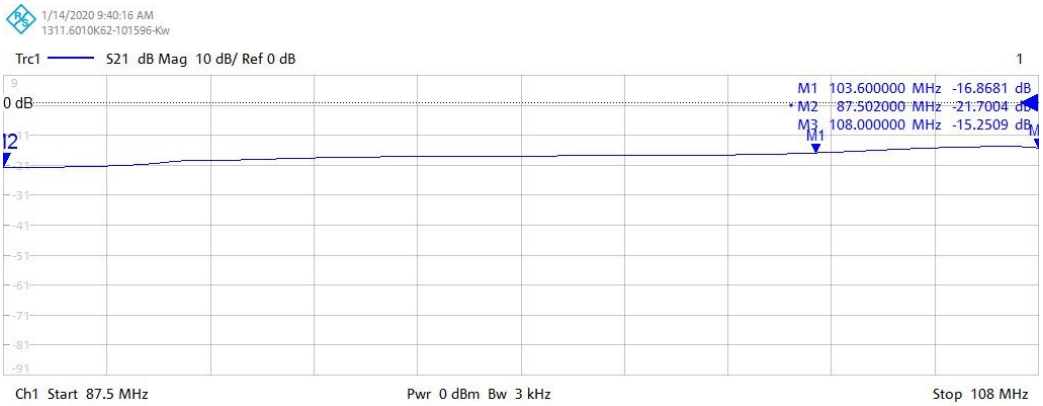


Abbildung 3.8: Linearität des Antennengewinns auf der 0°-Hauptachse.

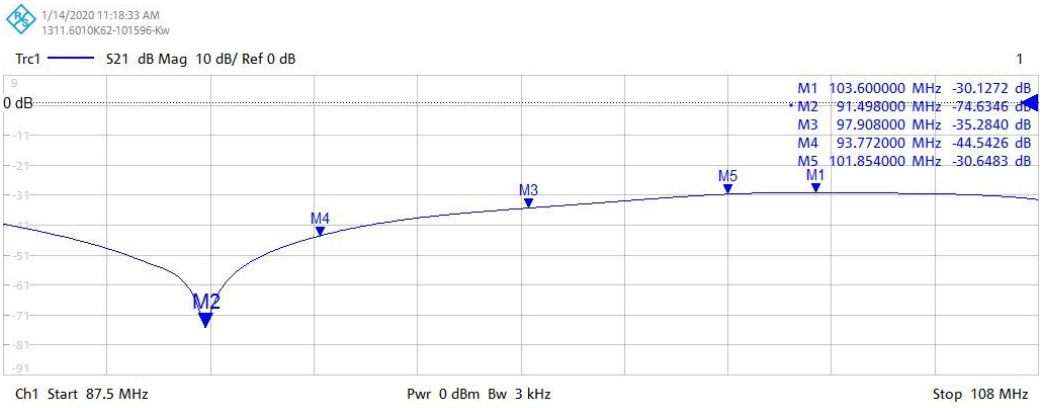


Abbildung 3.9: Linearität des Antennengewinns auf der 180°-Nebenachse.

Während der Verlauf des Antennengewinns in der 0°-Richtung in Abb. 3.8 nahezu linear ist, zeigt Abb. 3.9 einen nicht linearen Verlauf über alle Frequenzen auf der 180°-Achse. Aus der Differenz von M1 ($f_c = 103,6\ MHz$) zu M2 ($f_c = 91,5\ MHz$) ist erkennbar, dass die Richtcharakteristik für Signale, die von hinten kommen, stark frequenzabhängig ist. Sollte die Abtastfrequenz für die Feldmessung von 103,6 MHz abweichen, ist das Polardiagramm (3.7) nicht mehr akkurat, sondern nur noch schematisch als Referenz zu verwenden.

3.3 Software-Defined-Radio

Software-Defined-Radio beschreibt eine Gruppe von Empfängertechnik, die sich auf die notwendigen Hardwareanwendungen zur größtmöglichen flexiblen Anwendung spezialisiert hat. Sie verzichtet dabei auf alle möglichen Konfigurationen und Signalverarbeitungsschritte, die innerhalb einer Software realisiert werden können. Ein SDR besteht immer aus einem Sendeempfänger und einem Signalmodul (z.B. Mikroprozessor), mit einem Decoder und Controller für die gewählte Schnittstelle. Das SDR ist die primäre Vorgabe zur Realisierung des Passiv-Radars und wird somit Hauptbestandteil der Empfangstechnik in dieser Thesis. Seitdem SDR-Technologie preiswert, in einer Spanne von 20-300 €, zu erwerben ist und mit USB-Schnittstellen ausgestattet wurde, ist die Idee eines Passiv-Radars auf der Grundlage der SDR-Empfänger zunehmend relevanter geworden. Bereits im Jahr 2013 realisierte J. Vierinen das erste nachgewiesene, auf SDR basierende Passiv-Radar [16]. Auch M.J. Ryan schrieb in seiner Dissertation von 2016 [18], sowie Jean-Michel Friedt und sein Team in ihrer Arbeit von 2018 über die Realisierung von Passiv-Radarsystemen mit SDR-Einbindung. Heute gibt es eine Vielzahl von SDR-Systemen die sich auf dem aktuellen Markt etabliert haben. Die aufgeführten Systeme sollen die Marktvarianz aufzeigen und stehen nicht repräsentativ für ihre gesamte Produktgruppe.

3.3.1 HackRF (One)

Das HackRF One ist eine Open-Source-Hardware der Firma Great Scott Gadgets [19]. HackRf basiert auf dem RFFC5072 Breitband-Frequenzwandler mit integriertem Mischer [20] und dem MAX2837 Breitband-Sendeempfänger [21]. In Abbildung 3.10 ist das HackRF abgebildet und in Tabelle 3.1 sind die wichtigsten technischen Daten aufgelistet.

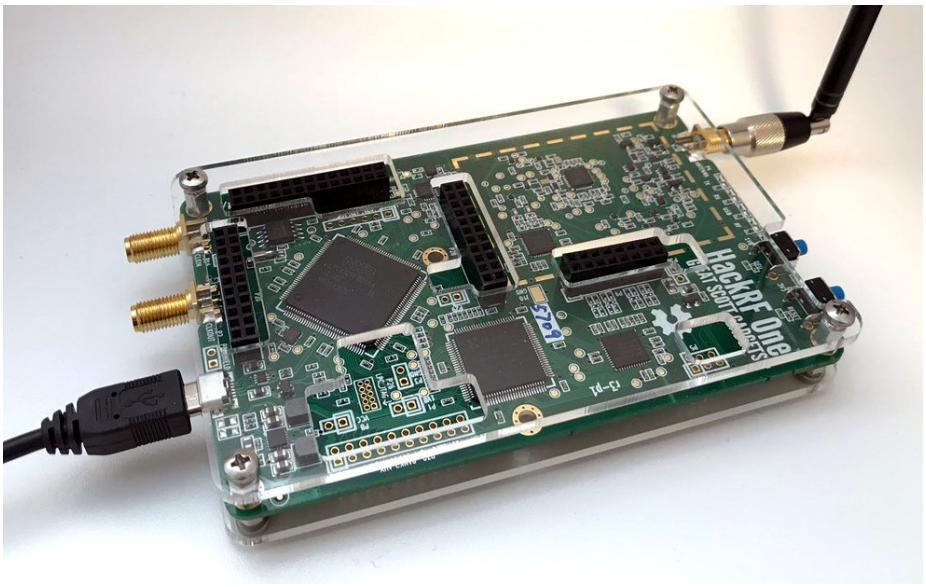


Abbildung 3.10: Great Scott Gadgets HackRF One [22].

Technische Daten	
Stromanschluss	USB 2.0 Bus
Richtungsabhängigkeit	half duplex
Frequenzbereich	1 MHz - 6 GHz
Auflösung	8 bit
Pc-Schnittstelle	USB-C 2.0
Antennen-Schnittstelle	SMA (50 Ω)
Takt-Synchronisation	SMA
Sendeempfänger	MAX2837
Frequenzwandler	RFFC5072
programmierbare Knöpfe	2
Software Support	MATLAB, GNU, SDR♯, Cubic

Tabelle 3.1: Technische Daten des HackRF One [19].

3.3.2 bladeRF (2.0 micro)

Das bladeRf 2.0 micro der Firma Nuand gehört in seiner xA4 und xA9 Ausführung zu den neusten Ableger der bladeRF-Reihe. Das bladeRF basiert auf dem ALTERA CYCLONE V FPGA Sendeempfänger [24] und dem ARM926EJ-S Mikroprozessor[25] [23].

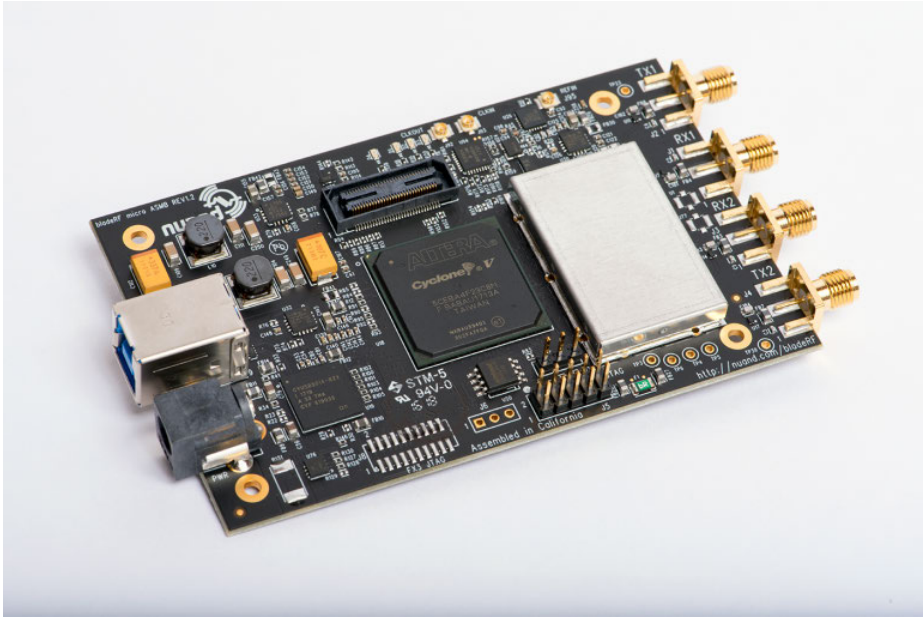


Abbildung 3.11: Nuand bladeRF 2.0 micro xA4 [23].

Technische Daten	
Stromanschluss	USB 3.0 Bus und 5V DC extern
Richtungsabhängigkeit	half duplex
Frequenzbereich	47 MHz - 6 GHz
Auflösung	12 bit
Pc-Schnittstelle	USB 3.0
Antennen-Schnittstelle	2 SMA (50 Ω)
Tack-Synchronisation	intern
Sendeempfänger	ALTERA CYCLONE V FPGA
Prozessor	ARM926EJ-S
Delay Offset Korrektur	intern
Software Support	MATLAB, GNU, SDR♯

Tabelle 3.2: Technische Daten des bladeRF 2.0 micro xA4 [23].

3.3.3 USRP (B200)

National Instrument vertreibt unter der Ettus Research Marke eine breite Auswahl an hochpreisigen SDR-Systemen. Das B200 gehört zu den einfacheren USRP (engl.: Universal Software Radio Peripheral) Systemen [27] und basiert auf einem Spartan-6 FPGA (engl.: Field Programmable Gate Array) Prozessor [28] und einem AD9364 Sendeempfänger [26].

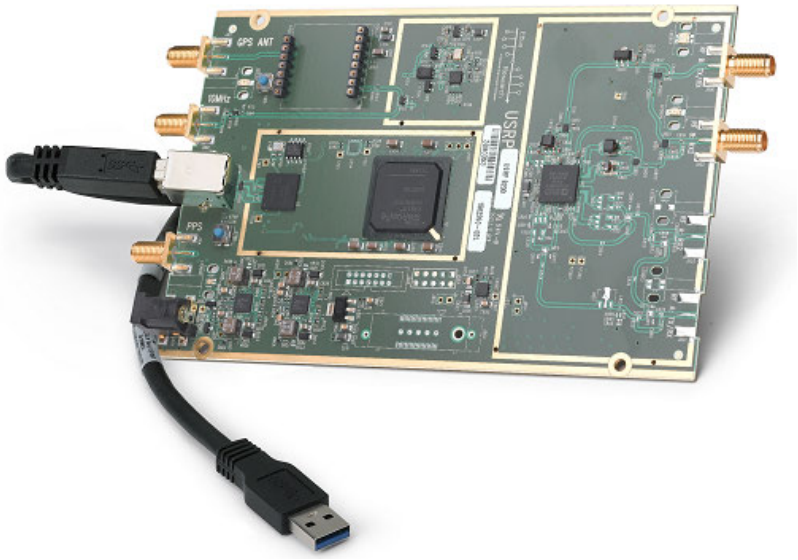


Abbildung 3.12: Ettus Research USRP B200 [27].

Technische Daten	
Stromanschluss	USB 3.0 Bus
Richtungsabhängigkeit	full duplex
Frequenzbereich	70 MHz - 6 GHz
Auflösung	12 bit
Pc-Schnittstelle	USB 3.0
Antennen-Schnittstelle	2 SMA Tx, 2 SMA Rx(50 Ω)
Tack-Synchronisation	intern, extern SMA
Sendeempfänger	AD9364
Prozessor	Spartan-6 FPGA
Software Support	MATLAB, GNU, SDR♯

Tabelle 3.3: Technische Daten des USRP B200 [27].

3.3.4 RTL-SDR

RTL-SDR vereint mehrere Systeme von verschiedenen Anbietern, die den RTL2832U-Chip von Realtek verwenden. RTL-SDR ist das meist verbreitete SDR-System und verwendet eine Reihe an unterschiedlichen Sendeempfängern.

RTL-SDR Sendeempfänger	
Tuner	**Frequenzbereich**
Elonics E4000	54 - 2200 MHz
Rafael Micro R820T	24 - 1766 MHz
Rafael Micro R820T2	24 - 1766 MHz
Fitipower FC0013	22 - 1100 MHz
Fitipower FC0012	22 - 948 MHz
FCI FC2580	146 - 308 MHz und 438 - 924 MHz

Tabelle 3.4: Auflistung der Tuner, die mit dem RTL2832U für SDR verwendet werden [29].

Ursprünglich wurde das RTL-SDR als DVB-T-Empfänger konzipiert, bis 2010 herausgefunden wurde, dass auf die Rohdaten im RTL2832U zugegriffen werden kann. Durch die Entwicklung eines Treibers für den RTL2832U ist es möglich die DVB-T-Funktion des Empfängers zu deaktivieren und 8 Bit Datensamples über den USB-Port zu übertragen [29].
Für die Thesis werden zwei Nooelec NESDR Mini 2 RTL-SDR Sticks verwendet [11], die im Rahmen dieser Arbeit zur Realisierung des Passiv-Radars vorgegeben wurden.

3.4 NESDR Mini 2 RTL-SDR

Die wichtigsten Elemente der Baugruppe vom Nooelec NESDR Mini 2 RTL-SDR werden in Abb. 3.13 abgebildet.

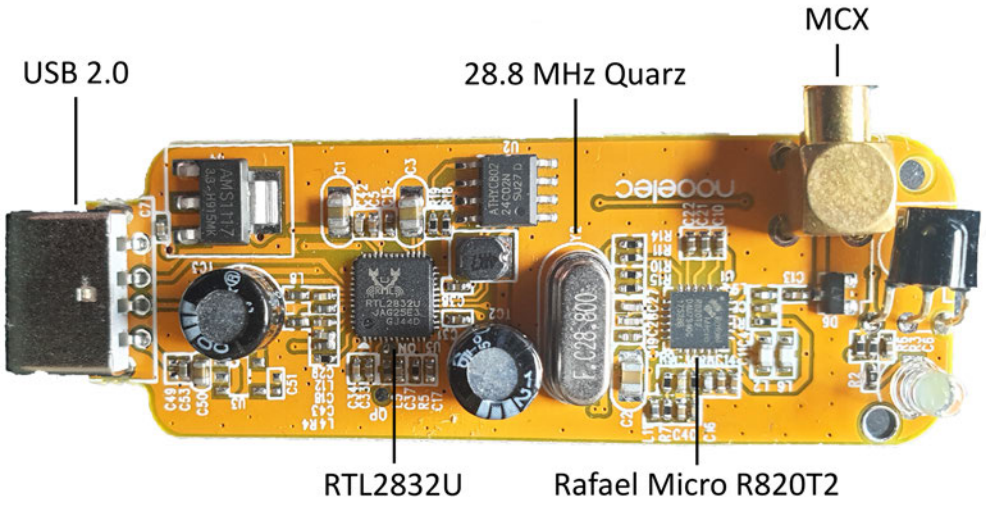


Abbildung 3.13: Nooelec NESDR Mini 2 RTL-SDR.

3.4.1 Rafael Micro R820T2

Im NESDR Mini 2 ist ein Rafael Micro R820T2 als Sendeempfänger verbaut [30]. Nach dem Datenblatt agiert der Empfänger R820T2 in einem Frequenzbereich von 42 - 1002 MHz [30]. In der Praxis wird vom NESDR Mini 2 ein Frequenzbereich von 22 - 1766 MHz erreicht. Zu den in Abb. 3.14 gezeigten Schnittstellen, gibt das Datenblatt an, dass der Empfänger einen maximalen Leistungspegel von 10 dBm verarbeiten und an Pin 24 (RF-IN) führen kann. Der Kanal RF-IN besitzt ein Eigenrauschen von 3,5 dB. Als minimale Betriebsspannung erwartet der R820T2 3 V am Pin 23 (VCC). Maximal kann er eine Spannung von 3,6 V ohne Schäden bedienen, empfohlen wird von Rafael Microelectronics beim langfristigen Gebrauch eine Betriebsspannung von 3,3 V, die im System vom NESDR Mini 2 umgesetzt wurde.

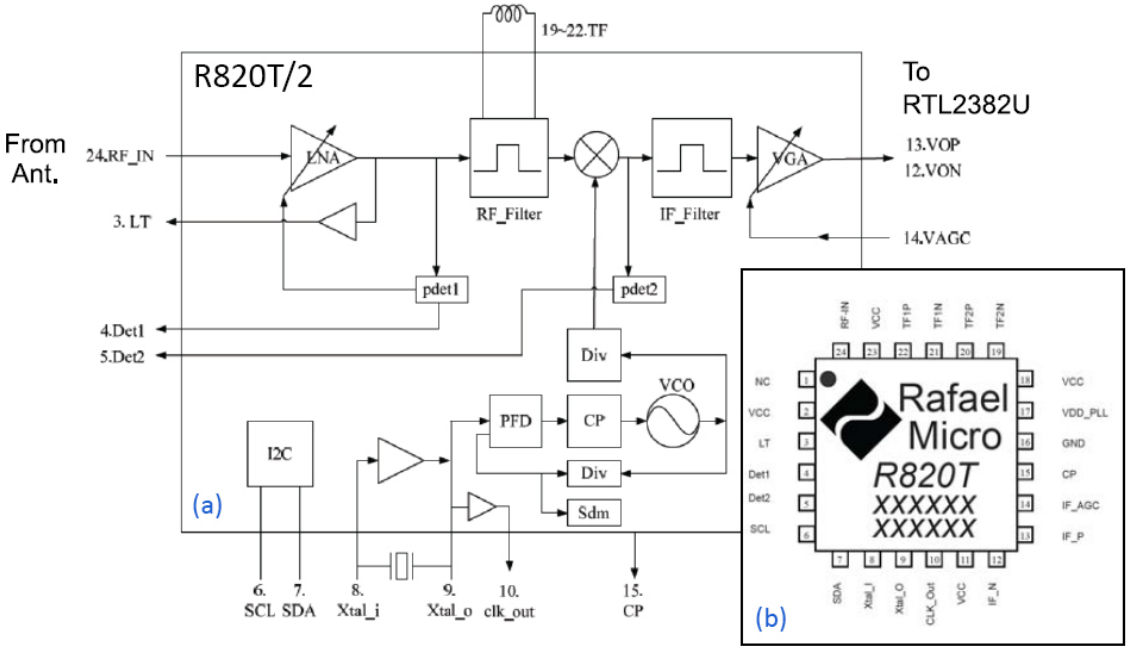


Abbildung 3.14: Blockschaltbild (a) und Pinbelegung (b) des Rafael Micro R820T2 [30].

Anhand der Abb. 3.14 lässt sich der Sendeempfänger in einem vereinfachten Blockschaltbild darstellen.

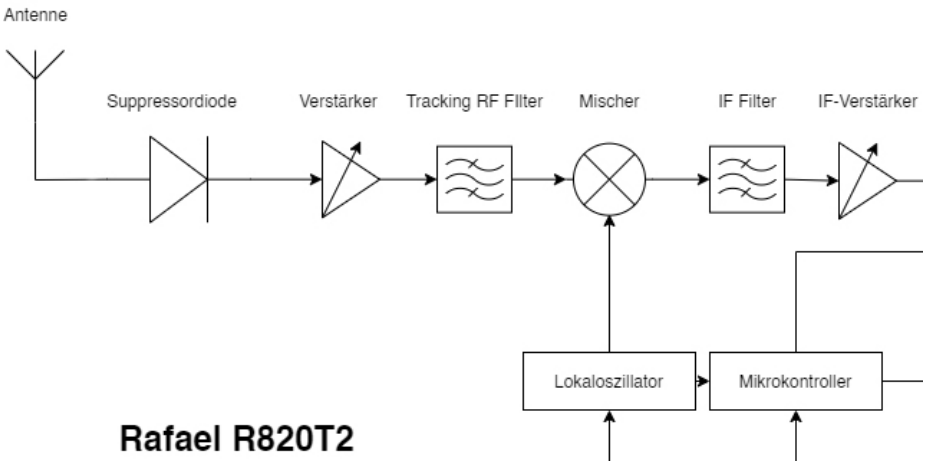


Abbildung 3.15: Blockschaltbild vom Rafael Micro R820T2.

Das Sendesignal wird von der Antenne über die Mikro-Koaxial Schnittstelle (engl.: micro coaxial connector; MCX) übergeben und trifft im R820T2 zuerst auf eine Suppressordiode, welche die Schaltung vor Spannungsimpulsen und Überschwingern schützt. Danach wird es verstärkt und auf seine Radiofrequenz (engl.: Radio frequency: RF) mit einem Glättungs-Tiefpass-Filter beschränkt. Anschließend wird das Signal mit der Takt-Frequenz des Lokaloszillators gemischt und auf den Zwischenfrequenzbereich (engl.: Intermediate frequency; IF), bzw. dem Zero-IF (IF = 0 MHz) für die Analog-Digital-Wandlung umgesetzt und verstärkt. Der Mikrocontroller bekommt vom Steuerprotokoll die eingestellten Parameter, wie u.a. die Trägerfrequenz, Abtastrate und Signalverstärkung. Abhängig von den übermittelten Parametern werden die Einstellungen der elektronischen Bauteile vom Mikrocontroller geändert. Vereinfacht zusammengefasst, besteht die Aufgabe des R820T2 darin, die Empfangsparameter einzustellen, um das korrekte Sendesignal zu empfangen und es für die Analog-Digital-Wandlung im RTL2832U vorzubereiten.

3.4.2 Realtek RTL2832U

Der Realtek RTL2832U ist ein DVB-T Demodulator, der ebenfalls FM, DAB und DAB+ unterstützt. Er kann mit Empfängern bei Zwischenfrequenzen von 36,123 MHz, 4,57 MHz, oder 0 MHz mit 28,8 MHz Taktfrequenz eingesetzt werden [31]. Aus diesem Grund bereitet der R820T2 die zero-IF mit der Taktfrequenz von 28,8 MHz vom Quarz vor. In Abb. 3.16 wird das Blockschaltbild aus Abb. 3.15 um den RTL2832U erweitert, um den gesamten NESDR Mini 2 RTL-SDR darzustellen.

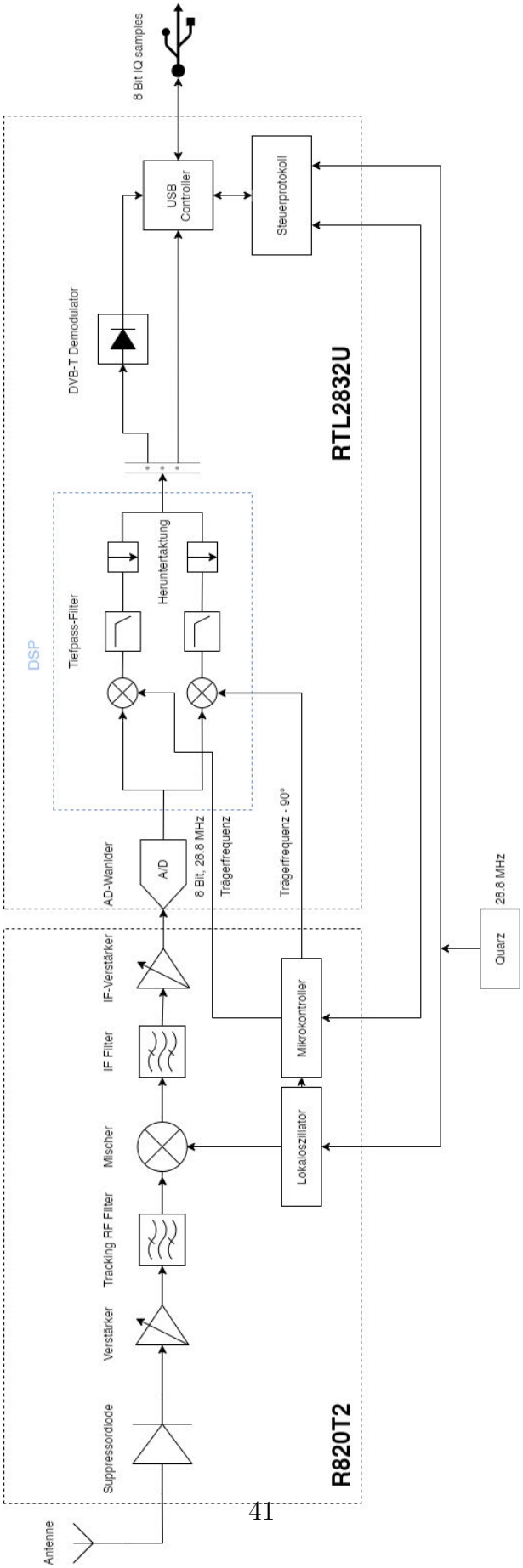


Abbildung 3.16: Blockschaltbild vom Nooelec NESDR Mini 2 RTL-SDR.

Das vom R820T2 überführte zero-IF-Signal wird im 8 Bit Analog-Digital-Wandler bei der Taktrate von 28,8 MHz gewandelt und anschließend im digitalen Signalprozessor in IQ-Signale (engl.: In-Phase Quadrature signal) aufgeteilt. Die IQ Signale werden in Kapitel 5.3 erklärt. Abhängig davon, ob auf dem Steuerprotokoll des RTL2832U ein DVB-T, oder SDR Patch installiert wurde, werden die IQ-Signale demoduliert (DVB-T Patch), oder als modulierte IQ-Rohdaten (SDR Patch) an den USB-Controller geschickt, der sie im Multiplexingverfahren über die USB 2.0 Schnittstelle an den Computer weiterleitet.
Zur Aufnahme von Signalen wird in umgekehrter Richtung zuerst der USB-Controller angesteuert, der alle relevanten Parameter an das Steuerprotokoll im RTL2832U gibt. Das Steuerprotokoll setzt den Operationsmodus (DVB-T, oder SDR) im RTL2832U fest, gibt einen Initialwert zum Start an den Lokaloszillator und die Empfangsparameter wie Trägerfrequenz, Bandbreite bzw. Abtastfrequenz und Verstärkungen an den Mikrocontroller im R820T2. Dieser stellt daraufhin den Sendeempfänger ein.

3.5 Zusammenfassung

Dieses Kapitel führt die verwendete Antennentechnik ein und charakterisiert diese. Hierzu wird die Messung der Beobachtungsantenne detailliert beschrieben. Das SDR wird in diesem Kapitel eingeführt und marktführende Modelle werden in einer Kurzübersicht vorgestellt. Das in dieser Thesis verwendete NESDR Mini 2 RTL-SDR der Firma Nooelec, wird im Detail in seiner elektronischen Baugruppe und Funktionsweise anhand von Signalflussdiagrammen erklärt.

4 Software

Damit die Softwareeinbindung von RTL-SDR funktionieren kann, wird der USB-Treiber libusb1.0.dll und der Operationstreiber librtlsdr.dll benötigt [32].

4.1 Software-Treiber

Bei den Treibern libusb1.0.dll und librtlsdr.dll handelt es sich mehr um erweiternde Bibliotheken als um eigenständige Treiber. Die libusb Bibliothek sorgt nicht nur für die Kommunikation zwischen der Software und dem SDR, sondern richtet auch die USB-Schnittstelle unter dem Betriebssystem für den RTL-SDR ein. Die librtlsdr Bibliothek überschreibt den Operationsmodus im Steuerprotokoll vom RTL2832U aus Abb. 3.16 und deaktiviert den DVB-T Demodulator, sodass die IQ-Rohdaten auch aus anderen Frequenzbändern zum PC übertragen werden können.

4.1.1 Installation unter Linux

Linux bietet die einfachste und flexibelste Einbindung der Bibliotheken, in dem diese aus dem GitHub der Treiber-Entwickler geladen, in CMake aufgebaut und anschließend kompiliert werden. Der Vorgang wird über die Linux-Kommandozeile ausgeführt, deren Kommandos für die Installation in Anlage B hinterlegt sind. CMake ist eine Open-Source-Software, die geeignet zum Aufbau von Code-Paketen und deren Kompilierung ist [34]. Abschließend wird die Installation des DVB-T Standard-Treibers von Linux blockiert, damit das System nicht automatisiert den Standard-Treiber beim Erkennen von DVB-T-Hardware installiert und einrichtet.

4.1.2 Installation unter Windows

Auf dem Windows-Betriebssystem ist der Aufbau und die Kompilierung der Bibliotheken, die im GitHub für Windows veröffentlicht wurden, nicht ohne Probleme mög-

lich. Die Bibliotheken sind zum Zeitpunkt der Einbindung (Stand: 01.04.2020) um mehrere Jahre veraltet und entsprechen nicht mehr dem strukturellen Aufbau von Windows 10, wodurch Fehler im USB-Register und im Windows-Dateipfad auftreten. Stattdessen wird unter Windows das Programm Zadig verwendet [35], welches zur Installation von USB-Treibern dient, um die libusb Bibliothek einzurichten.
Die librtlsdr Bibliothek wird in der verwendeten Software bei der Konfiguration des SDR oder durch die Ausführung von externen Plugin-Anwendungen des Programms auf dem RTL-SDR installiert.

Nach dem Starten von Zadig muss, wie in Abb. 4.1 gezeigt, innerhalb der Navigationsleiste *Options* ⇒ *List All Devices* + *Ignore Hubs or Composite Parents* ausgewählt werden, damit alle angeschlossenen USB-Geräte erkannt und aufgelistet werden.

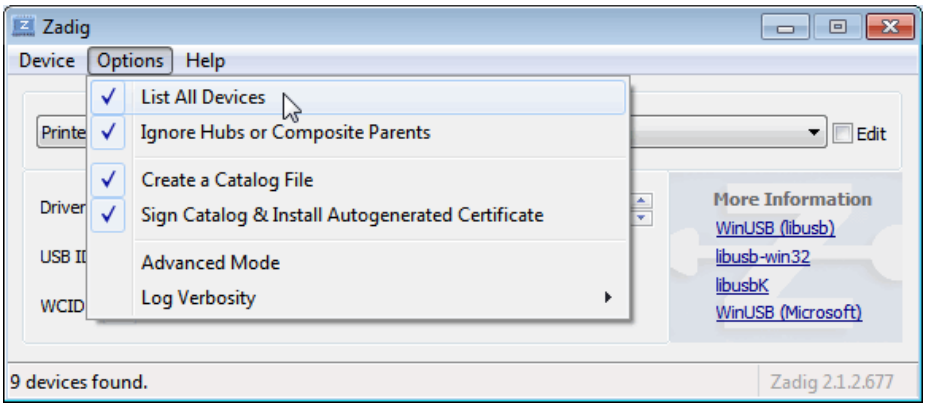


Abbildung 4.1: Installationsschritt 1 von libusb1.0 in Zadig.

Abhängig vom RTL-SDR wird das SDR in Zadig als RTL2838UHIDIR (Abb. 4.2), oder Bulk-In, Interface (Abb. 4.3) bezeichnet.

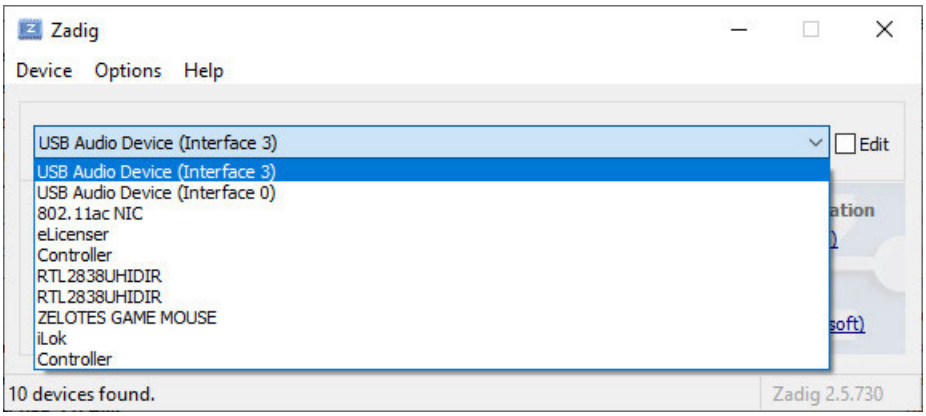


Abbildung 4.2: Installationsschritt 2 von libusb1.0 in Zadig.

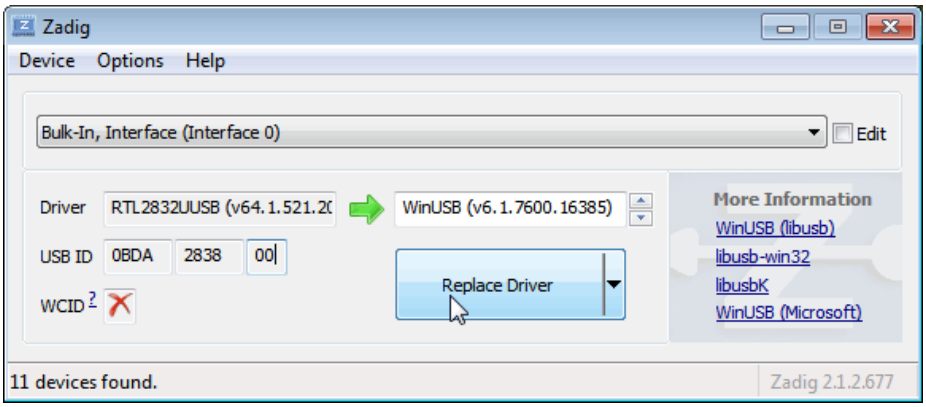


Abbildung 4.3: Installationsschritt 3 von libusb1.0 in Zadig.

Nach Auswahl des RTL-SDR muss, wie in Abb. 4.3 gezeigt, der RTL2832 Treiber durch den WinUSB-Treiber ersetzt werden. WinUSB ist die Bezeichnung von libusb1.0 innerhalb Zadig.
Sollte der RTL-SDR nicht über die USB-Schnittstelle funktionieren, müssen die im Kompositum übergeordneten Objekte des RTL2832U ebenfalls aktualisiert werden. Dafür werden die Objekte in Abb. 4.1 unter *Options* ⇒ *Ignore Hubs or Composite Parents* deaktiviert und anschließend mit dem WinUSB-Treiber aktualisiert.

4.2 SDR Software

Die Signalverarbeitung in der Thesis wird in der Windows-Umgebung umgesetzt. Auf Windows gibt es eine Reihe von Software für die Signalverarbeitung, die RTL-SDR mit der librtlsdr Bibliothek unterstützen. Die bekanntesten werden im Folgenden aufgelistet.

4.2.1 Airspy SDR#

Betriebssystem: Windows, Linux, macOS
Sprache: C++

SDR# wurde aufgrund seiner Stabilität unter Windows und einer vollständigen, grafischen Benutzeroberfläche (engl.: graphical user interface; GUI) zur meist verbreiteten SDR-Anwendung [36]. Es ist auf seine Hauptfunktion als Spektrumanalysator begrenzt, lässt sich allerdings durch viele verschiedene Bibliotheken erweitern. Außerhalb der Bibliotheken lässt sich SDR# nicht eigenständig ändern oder anpassen. In seiner Grundversion unterstützt SDR# folgende SDR:

- Airspy
- HackRF
- USRP
- RTL-SDR

4.2.2 CubicSDR

Betriebssystem: Windows, Linux, macOS
Sprache: C++

CubicSDR ist ein Spektrumanalysator und besitzt die Fähigkeit Frequenz- und Amplitudenmodulierte Signale zu demodulieren [37]. Im Vergleich zu SDR# sieht es von der Benutzerfläche ähnlich aus, besitzt allerdings einen kleineren Anwendungsbereich, weil es sich nicht durch Bibliotheken oder externe Plugins erweitern lässt.

CubicSDR unterstützt folgende SDR:

- SDRPlay
- HackRF
- BladeRF
- Airspy
- NetSDR+
- RTL-SDR
- Red Pitaya

4.2.3 GNU Radio

Betriebssystem: Windows, Linux, macOS
Sprache: C++, Python

GNU Radio ist ein Entwicklungs-Toolkit, welches Signalverarbeitungsblöcke für die Implementierung von SDR liefert [38]. Es kann beliebig angepasst und durch neue Blöcke in C++, oder Python erweitert werden. GNU Radio besitzt eine Block basierte Architektur, in der die Signalverarbeitungsblöcke im GUI in der Form eines Blockschaltdiagramms miteinander verknüpft werden. Auch wenn GNU Radio für andere Betriebssysteme geschrieben wurde, ist es für Linux und Unix Systeme optimiert und externe Erweiterungen lassen sich unter Linux einfacher einbinden als unter Windows. GNU Radio unterstützt folgende SDR:

- ADALM-PLUTO
- USRP
- XTRX
- UmTRX
- Funcube Pro+ Dongle
- KerberosSDR
- HackRF
- Lime SDR

- Microtelecom Perseus
- New Horizons NH7020
- Myriad-RF
- BladeRF
- RTL-SDR

4.2.4 MATLAB

Betriebssystem: Windows, Linux, macOS
Sprache: C, C++, Java

MATLAB dient als industrieller Standard für numerische Datenanalyse und Simulation in sämtlichen mathematischen Teilbereichen und lässt sich durch Toolboxen und Add-Ons beliebig erweitern [39]. Mathworks hat 2014 ein Add-On für seine Signal Processing Toolbox veröffentlicht, wodurch die Kommunikation mit RTL-SDR möglich wurde. Als finale Vorgabe dieser Thesis wird MATLAB R2020a für die Signalverarbeitung verwendet. MATLAB unterstützt folgende SDR:

- RTL-SDR
- USRP
- Zynq
- BladeRF
- ADALM-PLUTO

4.3 RTL-SDR Einbindung in MATLAB

Für die RTL-SDR Unterstützung in Matlab sind vorab Bedingungen zu erfüllen. Zuerst muss die libusb1.0 Bibliothek über Zadig installiert und für die SDR-Empfänger konfiguriert werden. Anschließend müssen weitere MATLAB Toolboxen installiert werden.

- Communications Toolbox
- DSP System Toolbox
- Signal Processing Toolbox

Sind die gesetzten Bedingungen erfüllt, kann das *Communications Toolbox Support Package for RTL-SDR Radio* installiert werden. Damit MATLAB eine Kommunikation zum RTL2832U des SDR aufbauen kann, ist dieses Paket für die Communications Toolbox essenziell. Nach der Installation kann in der Navigation über *Add-Ons* ⇒ *Manage Add-Ons* ⇒ *Communications Toolbox Support Package for RTL-SDR Radio* ⇒ *Setup* das RTL-SDR für die Toolbox konfiguriert werden.

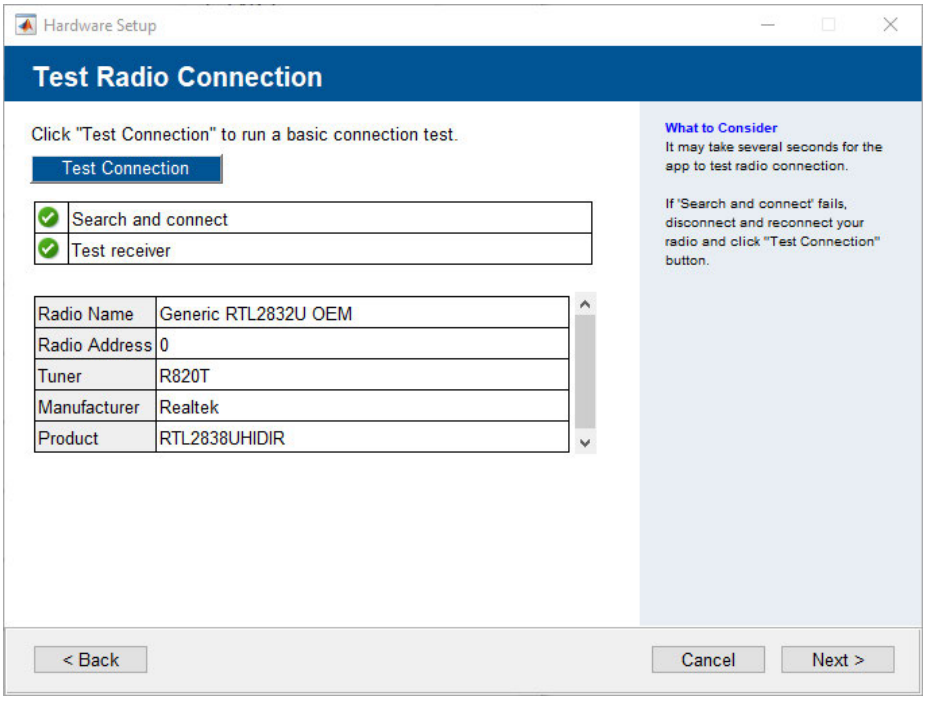


Abbildung 4.4: Communications Toolbox Support Package for RTL-SDR Radio Konfigurationsfenster.

In Abb. 4.4 wird über den Verbindungstest geprüft, ob MATLAB eine Kommunikation zum SDR aufbauen kann. Dann wird das Steuerprotokoll des SDR um die librtlsdr Bibliothek erweitert und im SDR-Operationsmodus angesteuert. Nach dem erfolgreichen Test ist das SDR in MATLAB einsetzbar. Wenn mehrere SDR gleichzeitig in

MATLAB eingebunden werden, teilt MATLAB den SDR-Empfängern USB-Adressen zu. Die Logik der Adressierung [40] ist im Anhang C hinterlegt.
Das *Communications Toolbox Support Package for RTL-SDR Radio* bietet ebenfalls Erweiterungen und Einbindungen für Simulink. Dies ist eine Anwendung, die ähnlich wie GNU Radio eine Block basierte Architektur besitzt und die Signalverarbeitung in der Form von Blockschaltbildern darstellt. Die erweiterten Simulink-Blöcke werden im MATLAB-Verzeichnis bei den Standard-Blöcken installiert. Simulink benötigt keine weiteren Einstellungen, um mit RTL-SDR arbeiten zu können.
Keine notwendige Bedingung, dafür allerdings eine praktische Erweiterung, stellen die Simulink-Blöcke aus „Software Defined Radio using MATLAB & Simulink and the RTL-SDR“ dar [41].

Abbildung 4.5: Simulink Signalverarbeitungsblöcke der „Software Defined Radio using MATLAB & Simulink and the RTL-SDR“-Bibliothek.

Um die Simulink-Blöcke einzubinden, muss über die Navigation ein Suchpfad zu dem Verzeichnis festgelegt werden, in dem sich die Blöcke befinden.

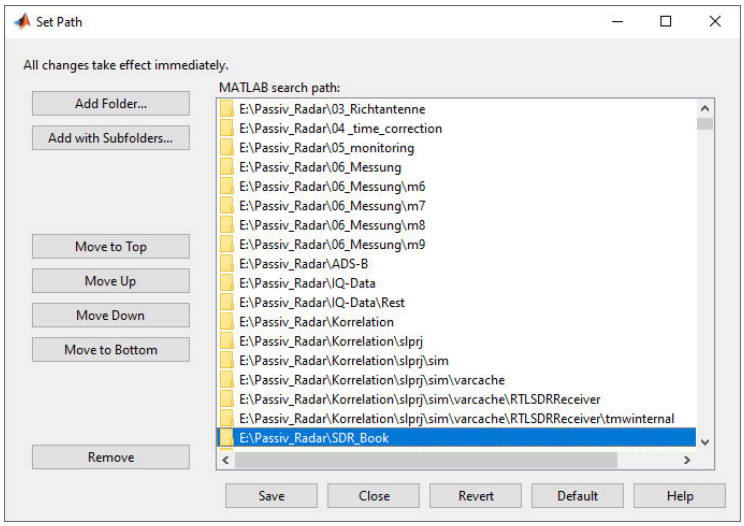


Abbildung 4.6: MATLAB Benutzeroberfläche zur MATLAB-Suchpfad-Einstellung.

4.3.1 Zusammenfassung

In diesem Kapitel werden die Bedingungen zur Softwareeinbindung von RTL-SDR unter Windows definiert und deren Konfiguration erörtert. Die marktführenden Programme zur Signalverarbeitung von SDR werden in einem Kurzprofil vorgestellt und charakterisiert. Abschließend wird das in dieser Thesis verwendete Programm für die Signalverarbeitung, MATLAB, eingeführt und seine Installation und Konfiguration für RTL-SDR-Systeme detailliert erklärt.

5 Erfassung reeller Sendesignale

5.1 MATLAB Kommunikation

Wenn das *Communications Toolbox Support Package for RTL-SDR Radio* installiert und konfiguriert ist, kann MATLAB mit dem Systemobjekt *SDRRTLReceiver* vom SDR aufgenommene Signale verarbeiten [42]. Die Syntax von *SDRRTLReceiver* ist im Anhang A.2 hinterlegt. In Simulink können die Signale mit dem *SDR-RTL Receiver*-Bock verarbeitet werden.

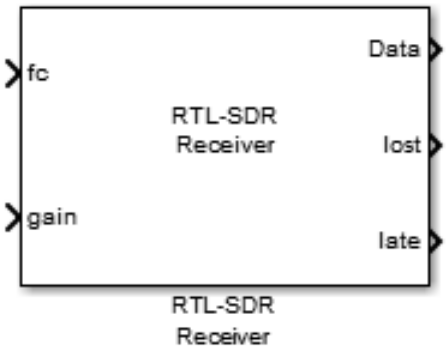


Abbildung 5.1: *SDR-RTL-Receiver*-Bock in Simulink.

Der Simulink-Block ist Teil der *Communications Toolbox* und ist als virtueller Block in einem schreibgeschützten Verzeichnis definiert. Virtuelle Blöcke lassen sich innerhalb von Simulink nicht durch Funktionsblöcke erweitern. Durch sein schreibgeschütztes Verzeichnis, lässt sich der *SDR-RTL Receiver*-Bock nicht in einen non-virtuellen Block umwandeln.

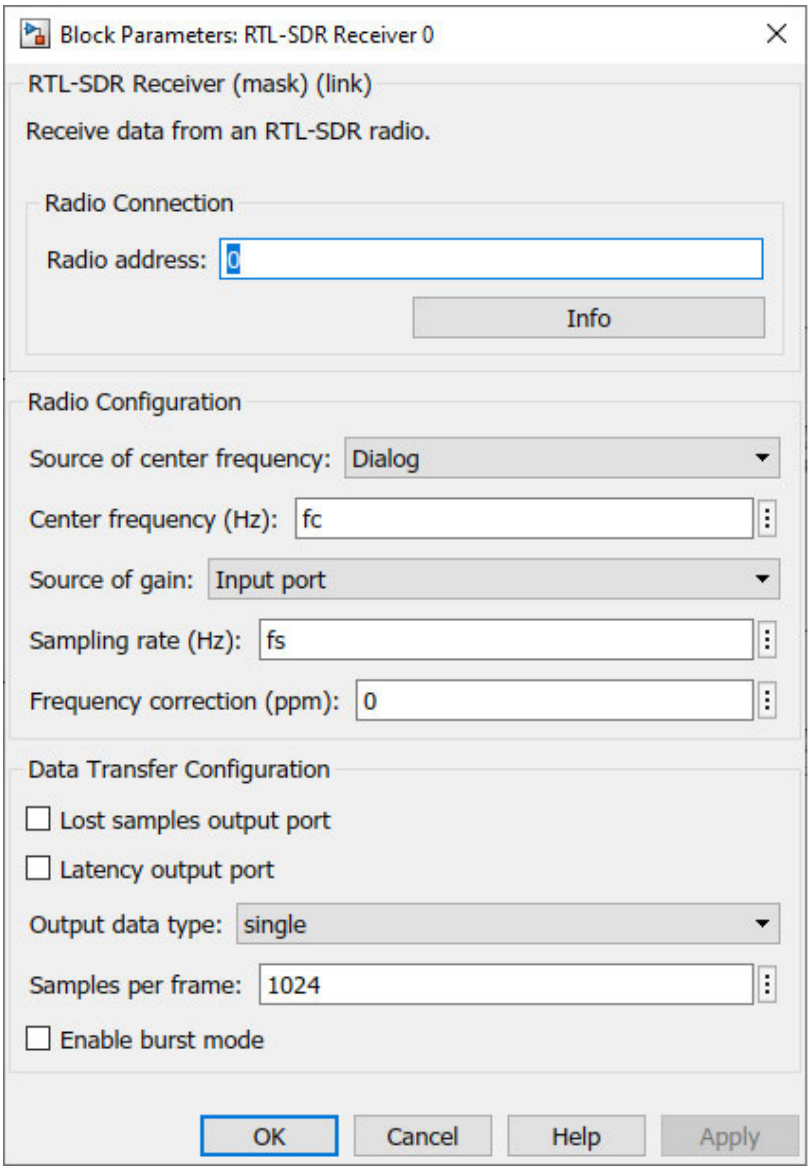


Abbildung 5.2: Einstellungsoberfläche des *SDR-RTL-Receiver* in Simulink.

In Abb. 5.2 können die SDR-Systemparameter direkt im Simulink-Block definiert werden und müssen nicht über die Kommandozeile eingespeist werden.

5.2 Aufzeichnung der SDR-Signale

In MATLAB und Simulink gibt es mehrere Möglichkeiten die SDR-Signale aufzuzeichnen.

5.2.1 SDRRTLReceiver-Objekt

Wie eingehend erwähnt, lassen sich mit dem *SDRRTLReceiver*-Objekt Signale vom SDR aufnehmen. Das Systemobjekt kann nicht unbegrenzt aufzeichnen und braucht eine Unterbrechung. Diese kann durch eine vorgegebene Simulationszeit oder durch einen Stop-Befehl realisiert werden. MATLAB speichert während der Simulation die Daten zwischen und überführt sie nach der Simulation in ein Daten-Array im *Workspace*.

5.2.2 Listener Callback

Listener Callbacks warten auf ein Event eines Objektes oder Klasse und zeichnen deren Parameterdaten auf [43]. Während diese Anwendung beim ADALM-PLUTO SDR funktioniert, ist das RTL-SDR-Objekt als virtuelles Systemobjekt klassifiziert und unterstützt keine *Listener*.

5.2.3 Simulink RTL-SDR-Block

In Simulink lassen sich die Daten des SDR über den Data Ausgang in einen *to File*- oder *to Workspace*-Block als Daten-Array oder Zeitreihe speichern [44]. Alternativ können unter *Model Configuration Parameters* ⇒ *Data Import/Export* verschiedene Speicheroptionen für die gesamte Simulink-Simulation eingestellt werden.

Trotz ihrer unterschiedlichen Bedienung haben alle Möglichkeiten gemeinsam, dass die Signaldaten nicht in Echtzeit während der Simulation, sondern erst nach Abschluss der Simulation aus dem Zwischenspeicher in den Zugriffsspeicher überführt werden. Dadurch kann zur weiteren Signalverarbeitung nicht auf die Daten während der laufenden Aufnahme zugegriffen werden. Zur Veranschaulichung werden in dieser Thesis die SDR-Daten durch Simulink aufgezeichnet und gespeichert.

5.3 In-Phase-&-Quadrature-Verfahren

In der Funktechnik werden hochfrequente Signale, sog. Bandpasssignale, für die Übertragung verwendet. Für die Signalverarbeitung muss das Bandpasssignal in das niederfrequentere Basisband überführt werden, ohne die Phaseninformation zu verlieren. Der in Abb. 3.16 gekennzeichnete DSP wird als Quadraturmischer im RTL2832U eingesetzt. Nach der Umformung in das Basisband und der Heruntertaktung, werden die IQ-Daten zum PC übertragen und können von der Software ausgelesen werden. Um die IQ-Daten für die Signalverarbeitung verständlich zu machen, wird die Zusammensetzung des Basisbandsignals anhand des Quadraturmischers in Abb. 5.3 hergeleitet [9, S.182-189].

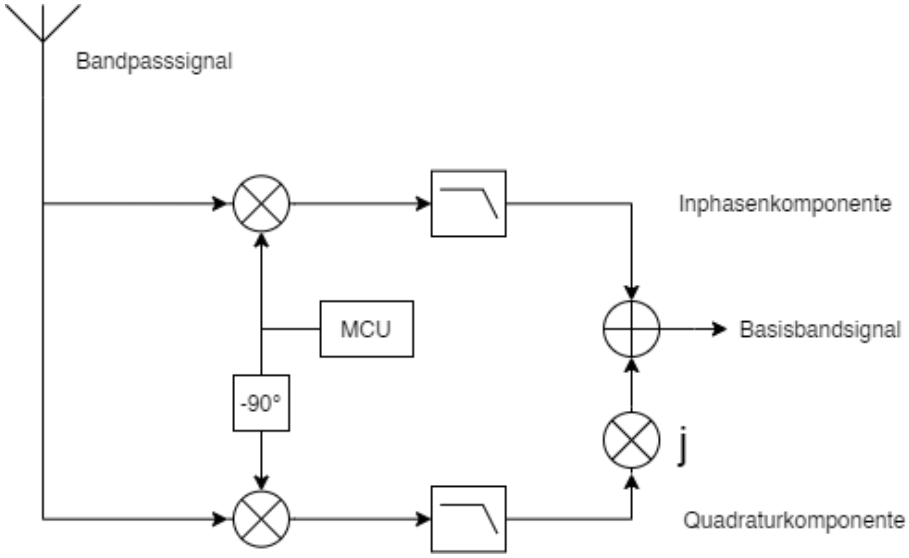


Abbildung 5.3: Signalflussdiagramm eines Quadraturmischers.

Das Bandpasssignal s_{BP} ist als Kosinus-Signal in Abhängigkeit der Trägerfrequenz f_c und des Phasenversatzes ϕ mit der Amplitude a definiert [9, S.182-183].

$$s_{BP}(t) = a(t) \cdot cos(2\pi f_c t + \phi(t)) \tag{5.1}$$

In Abb. 5.3 wird dargestellt, dass sich das Basisbandsignal s_{BB} aus der Summe der Inphasenkomponente s_I und der um j verschobenen Quadraturkomponente s_Q zusammensetzt.

$$s_{BB}(t) = s_I(t) + js_Q(t) \tag{5.2}$$

Für die Inphasenkomponente wird das Bandpasssignal mit dem Signal des Mikrocontrollers (MCU) multipliziert und anschließend tiefpassgefiltert. Der MCU überträgt die Schwingung des Lokaloszillators $2cos(2\pi ft)$ mit der Trägerfrequenz f_c, die vom Steuerprotokoll an den Mikrocontroller übermittelt wird.

$$s_I(t) = s_{BP}(t) \cdot 2cos(2\pi f_c t) * TP \tag{5.3}$$

Gleichung 5.1 in 5.3 eingesetzt:

$$s_I(t) = [a(t) \cdot cos(2\pi f_c t + \phi(t)) \cdot 2cos(2\pi f_c t)] * TP. \tag{5.4}$$

Der Tiefpass ist in seiner idealen Übertragungsfunktion im Zeitbereich als Sinc-Funktion mit der Grenzfrequenz f_g charakterisiert [9, S.165].

$$h(t) = 2f_g \cdot \frac{sin(2\pi f_g t)}{2\pi f_g t} \tag{5.5}$$

Mit $si(x) = \frac{sin(x)}{x}$ kann die Übertragungsfunktion vereinfacht werden.

$$h(t) = 2f_g si(2\pi f_g t) \tag{5.6}$$

Damit ergibt sich für die Inphasenkomponente:

$$s_I(t) = [a(t) \cdot cos(2\pi f_c t + \phi(t)) \cdot 2cos(2\pi f_c t)] * [2f_g si(2\pi f_g t)]\,. \tag{5.7}$$

Durch die trigonometrische Regel $cos(x) \cdot cos(y) = \frac{1}{2}\left[cos(x-y) + cos(x+y)\right]$ kann Gleichung 5.7 umgeformt werden.

$$s_I(t) = a(t) \cdot [cos(\phi(t)) + cos(4\pi f_c t + \phi(t))] * [2 f_g si(2\pi f_g t)] \tag{5.8}$$

Durch die Charakteristik des Tiefpasses fällt der hochfrequente Signalanteil weg.

$$s_I(t) = a(t) \cdot \left[cos(\phi(t)) + \cancelto{0}{cos(4\pi f_c t + \phi(t))}\right] * \left[\cancelto{0}{2 f_g si(2\pi f_g t)}\right] \tag{5.9}$$

Somit ist die Inphasenkomponente definiert als:

$$s_I(t) = a(t) \cdot cos(\phi(t)). \tag{5.10}$$

Die Quadraturkomponente ist in Abb. 5.3 die tiefpassgefilterte, zeitliche Multiplikation von dem Bandpasssignal und dem um $-\frac{\pi}{2} = 90°$ versetzten Signal vom MCU.

$$s_Q(t) = s_{BP}(t) \cdot 2cos\left(2\pi f_c t + \frac{\pi}{2}\right) * TP \tag{5.11}$$

$$= \left[a(t) \cdot cos(2\pi f_c t + \phi(t)) \cdot 2cos(2\pi f_c t + \frac{\pi}{2})\right] * [2 f_g si(2\pi f_g t)] \tag{5.12}$$

Durch die trigonometrische Regel $cos\left(x + \frac{\pi}{2}\right) = -sin(x)$ kann Gleichung 5.12 umgeformt werden.

$$s_Q(t) = [a(t) \cdot cos(2\pi f_c t + \phi(t)) \cdot 2 \cdot (-sin(2\pi f_c t)] * [2 f_g si(2\pi f_g t)] \tag{5.13}$$

Die Gleichung 5.13 kann mit der trigonometrischen Regel $sin(x) \cdot cos(y) = \frac{1}{2}\left[sin(x-y) + sin(x+y)\right]$ wiederum erneut umgeformt werden.

$$s_Q(t) = -a(t) \cdot [sin(-\phi(t)) + sin(4\pi f_c t + \phi(t))] * [2 f_g si(2\pi f_g t)] \tag{5.14}$$

Durch die mathematische Faltung des Signals mit dem Tiefpass, werden die hochfrequenten Anteile im Signal gefiltert.

$$s_Q(t) = -a(t) \cdot \left[sin(-\phi(t)) + \cancelto{0}{sin(4\pi f_c t + \phi(t))} \right] * \left[\cancelto{0}{2 f_g si(2\pi f_g t)} \right] \tag{5.15}$$

$$= -a(t) \cdot sin(-\phi(t)) \tag{5.16}$$

Danach kann mit der trigonometrischen Regel $-sin(-x) = sin(x)$ Gleichung 5.15 zur finalen Quadraturkomponenten umgeformt werden.

$$s_Q(t) = a(t) \cdot sin(\phi(t)) \tag{5.17}$$

Das Basisbandsignal kann durch das Einsetzen der Iphasen- und Quadraturkomponente in Gleichung 7.1 beschrieben werden.

$$s_{BB}(t) = a(t) \cdot cos(\phi(t)) + j \cdot a(t) \cdot sin(\phi(t)) \tag{5.18}$$

$$s_{BB}(t) = a(t) \cdot [cos(\phi(t)) + j \cdot sin(\phi(t))]$$

Die aufgenommenen FM-Signale vom SDR werden in MATLAB als modulierte, komplexe Basisbandsignale erfasst.

5.4 Demodulation

Die Demodulation des Mono-FM-Signals ist für das Passiv-Radar nicht relevant. Allerdings ist das demodulierte Signal eine wichtige Kontrollinstanz der Signalqualität des modulierten Signals. Die Demodulation kann innerhalb von Simulink mit den Simulink-Blöcken aus *Software Defined Radio using MATLAB & Simulink and the RTL-SDR* realisiert werden [41, S.375-383].

In Abb. 5.7 wird zuerst ein Dezimierungsfilter mit einem Dezimierungsfaktor von 10 angewendet. Dadurch wird die Abtastrate von 2,4 MHz auf 240 kHz verringert.

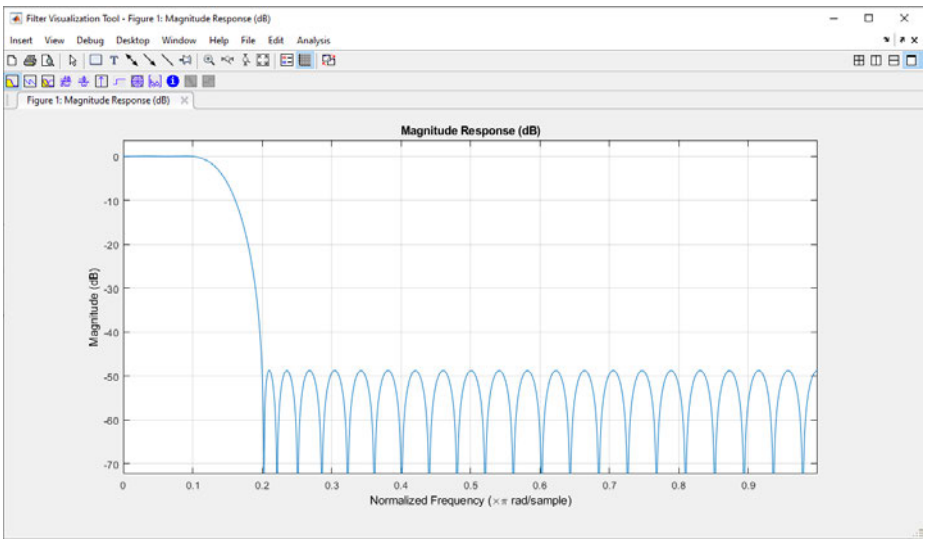


Abbildung 5.4: Filterkurve des ersten Dezimierungsfilters.

Das dezimierte Signal wird aufgeteilt und parallel um einen Sample zeitverzögert und komplex konjugiert und anschließend multipliziert. Danach wird das komplexe Signal in Betrag und Phasenwinkel aufgeteilt. Die Phasenwinkelinformation wird in einen weiteren Tiefpass-Dezimierungsfilter mit einem Dezimierungsfaktor von 5 geschickt.

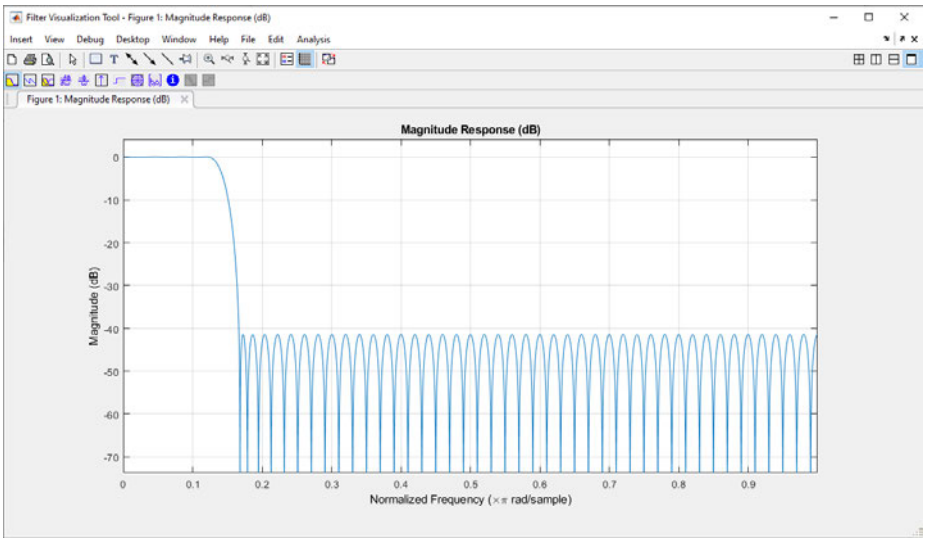


Abbildung 5.5: Filterkurve des zweiten Dezimierungsfilters.

Nach der Anwendung des zweiten Dezimierungsfilters werden Frequenzen ab 15 kHz raus gefiltert bei einer Abtastrate von 48 kHz. In Abb. 2.5 wurde gezeigt, dass im FM-Spektrum des Nutzsignals, das Mono-Signal bis 15 kHz definiert ist. Der letzte Schritt ist die Entzerrung der linearen Vorverzerrung. Diese ist kontinental fest definiert. Entzerrungsfilter greifen in Europa bei 2122,1 Hz ein [41, S.383].

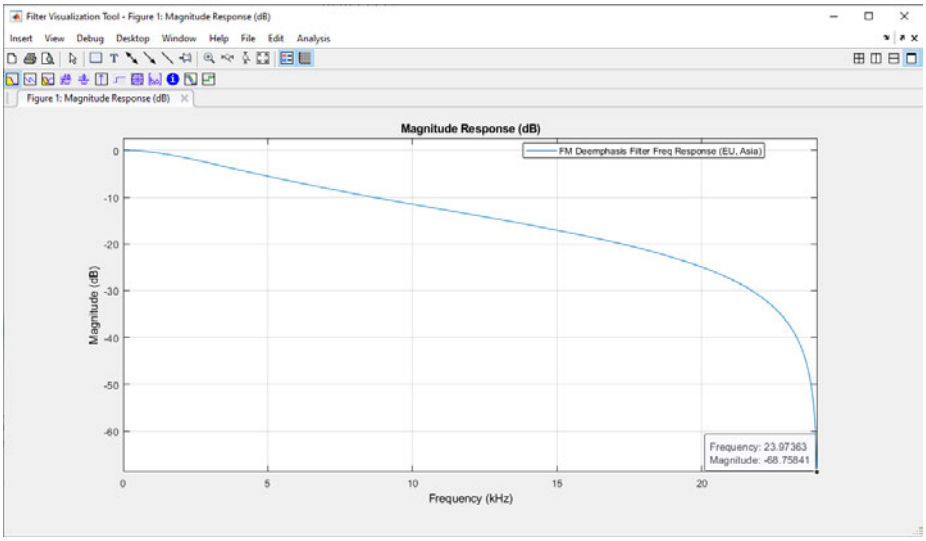


Abbildung 5.6: Filterkurve des Entzerrungsfilters.

Das demodulierte Signal kann während der Simulation im Zeitbereich (Scope-Block), im Frequenzbereich (FFT Analyzer-Block) oder über eine akustische Ausgabe (Audio Speaker-Block) wiedergegeben werden. So kann ermittelt werden, ob das gesamte Empfängersystem funktioniert und die erfassten Daten brauchbar sind.
Zur Visualisierung werden in Abb. 5.8 und Abb. 5.9 die Spektren eines FM-Signals vor und nach der Demodulation gezeigt. Abbildung 5.10 zeigt den zeitlichen Verlauf des modulierten Signals in Real- und Imaginärteil nach dem ersten Dezimierungsfilter und den Verlauf des modulierten Signals, nachdem die Phaseninformation vom Betrag getrennt wurde.

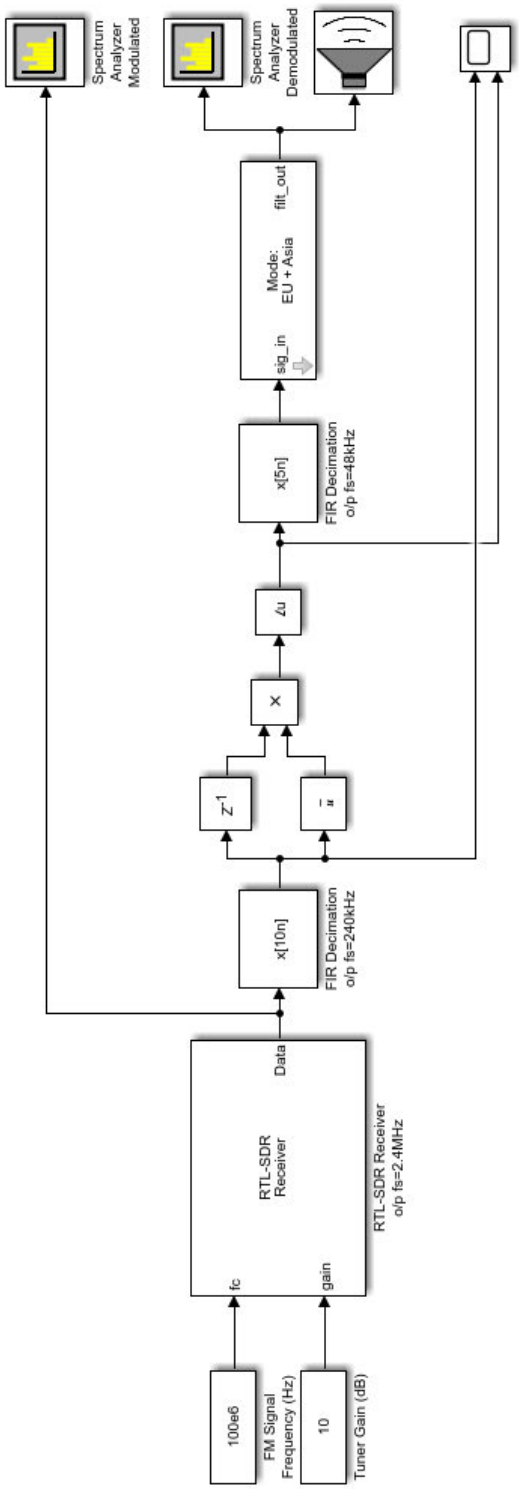


Abbildung 5.7: Simulink-Schaltbild zur Mono-FM-Demodulation aus *Software Defined Radio using MATLAB & Simulink and the RTL-SDR* [41, S.381].

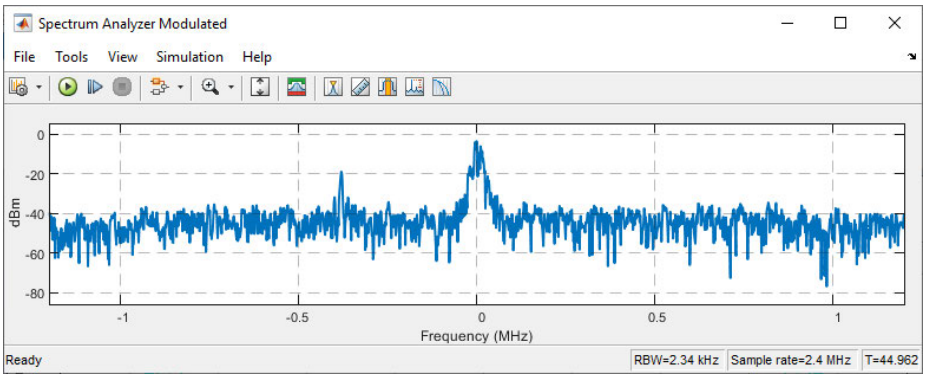


Abbildung 5.8: Frequenzspektrum eines modulierten FM-Signals.

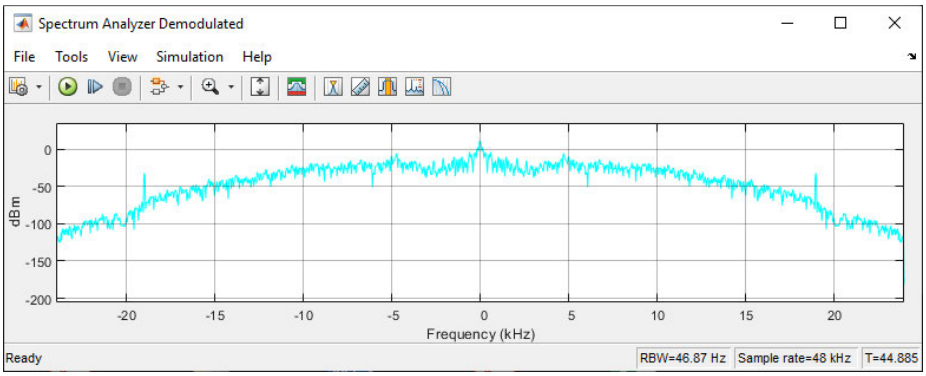


Abbildung 5.9: Frequenzspektrum eines demodulierten FM-Signals.

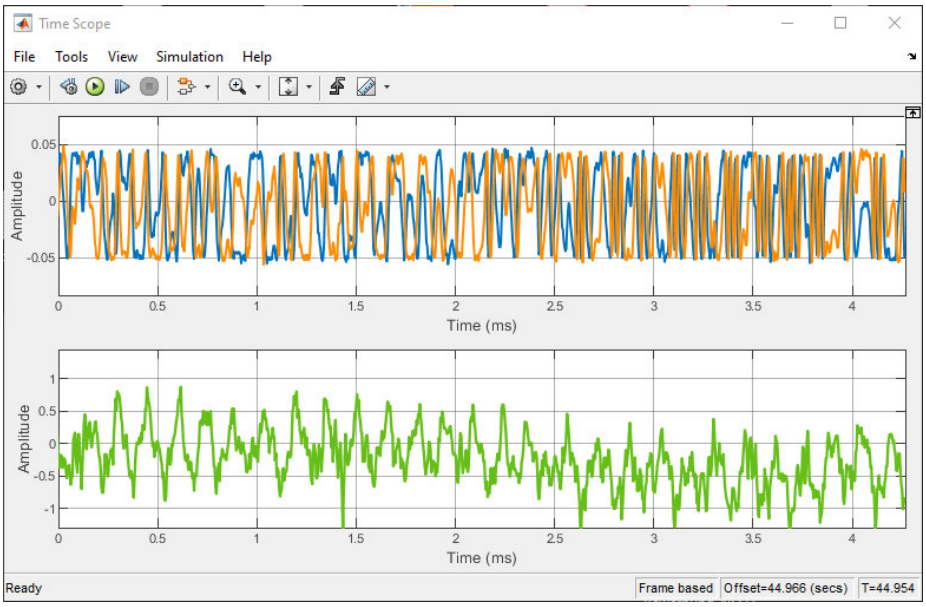


Abbildung 5.10: Zeitlicher Verlauf eines modulierten FM-Signals im Realanteil (blau) und Imaginäranteil (orange) und dem demodulierten Signal (grün).

5.5 Zusammenfassung

In diesem Kapitel wird beschrieben wie MATLAB, nach erfolgreicher Konfiguration des SDR, Signale erfassen kann. Dazu werden die Einbindungen des RTL-SDR Paketes in MATLAB und Simulink erklärt und verschiedene Methoden zur Aufnahme der Signale vorgestellt. Eine Echtzeit-Anwendung wird als Ergebnis der methodischen Möglichkeiten für die Thesis ausgeschlossen. Die aufgenommenen Signale werden als IQ-Daten definiert und hergeleitet. Abschließend wird eine Methode zur Demodulation des Mono-FM-Signals vorgestellt, um neben der spektralen und zeitlichen Visualisierung, eine weitere Methode der Signalüberwachung einzuführen.

6 SDR Synchronisation

6.1 Kohärentes SDR-System

Die größte Herausforderung dieser Thesis besteht in der Synchronisation beider SDR-Empfänger. Wie in Kapitel 2 beschrieben, werden für die Umsetzung eines bistatischen Passiv-Radars mindestens zwei Antennen benötigt. Diese werden mit zwei NESDR Mini 2 RTL-SDR eingebunden, die in Kapitel 3.4 eingeführt wurden. Grundvoraussetzung für ein Passiv-Radar ist ein kohärentes Empfängersystem. Das bedeutet, dass die beiden SDR nicht unabhängig voneinander im System Signale verarbeiten dürfen, sondern gleich getaktet, synchron arbeiten. Damit dies realisiert wird, werden die SDR in Zeit und Frequenz synchronisiert. Darüber hinaus muss das Empfängersystem auf die Sendeanlage des verwendeten Sendesignals konfiguriert werden. Ziel der Synchronisation ist es, ein synchron erfasstes Referenz- und Echosignal zu erhalten, damit die Ambiguitätsfunktion eine aussagekräftige Beziehung der Ähnlichkeit zwischen den beiden Signalen errechnen kann.

6.2 Trägersynchronisation

Sender und Empfänger sind im bistatischen Passiv-Radarsystem voneinander entkoppelt. Beide verwenden voneinander unabhängige Lokaloszillatoren. Die Schwingungen der Oszillatoren besitzen nicht die gleiche Frequenz und werden zu unterschiedlichen Zeitpunkten als Referenzfrequenz dem Nutzsignal zugeführt. Dies, in Verbindung mit Störungen und externen Einflüssen auf den Sendekanal, hat einen Frequenzversatz in der Trägerfrequenz im Empfänger von Δf in Abb. 6.1 zur Folge. Die Auswirkungen werden in Abb. 6.2 betrachtet.

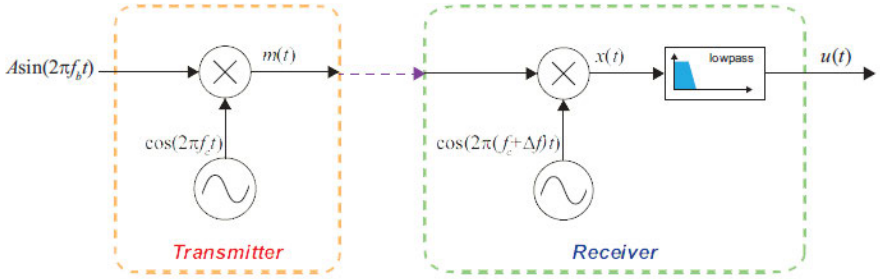


Abbildung 6.1: Demodulationssystem mit versetzter Trägerfrequenz [10, S.242].

In Abb 6.2 wird der Versatz deutlich sichtbar. Das SDR betrachtet das Signal nicht an der Trägerfrequenz, sondern mehrere kHz versetzt.

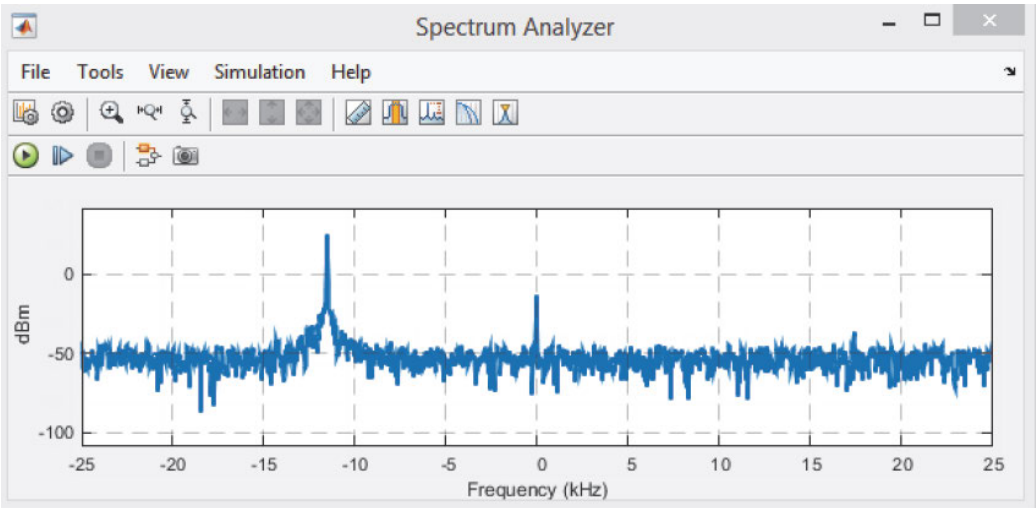


Abbildung 6.2: Frequenzversatz von -12,5 kHz bei 25 PPM [10, S.579].

Im Normalfall sollte die Frequenzdifferenz nur einem geringen Maß, PPM (engl.: Parts Per Million), entsprechen [10, S.578].

$$\delta_{ppm} = \frac{\Delta f}{f_c \cdot (-10)^{-6}} \tag{6.1}$$

$$\Delta f = \frac{f_c \cdot \delta_{ppm}}{10^6} \tag{6.2}$$

Im Datenblatt des R820T2 [30, S.23] wird die minimale Frequenzgenauigkeit mit ±50 PPM angegeben. In der Praxis lag das Minimum bei ±53 PPM. Dadurch ist ein maximaler Frequenzversatz von 5724 Hz beim SDR im UKW-Bereich möglich. Solange die Sendeanlage oder das SDR nicht neu gestartet wird und statisch bleibt, ist der Versatz der Trägerfrequenz konstant. So kann der Versatz mit der Simulink-Schaltung aus Abb. 6.3 ermittelt und anschließend im SDR-Block kompensiert werden.

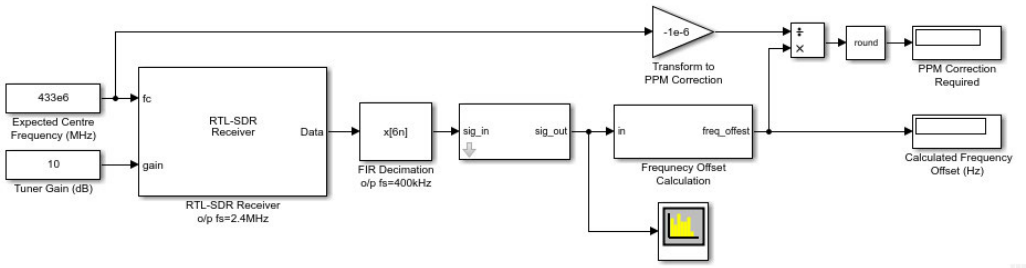


Abbildung 6.3: Simulink-Schaltung zur Messung des Frequenzversatzes [10, S.579].

Die Schaltung dezimiert und entfernt den Gleichspannungsanteil des Sendesignals und vergleicht den Maximalwert der Fourier-Transformierten mit der Trägerfrequenz und gibt den Frequenzversatz in Hz und PPM aus.

6.3 Frequenzsynchronisation

Beide SDR werden über die USB-Schnittstelle mit Spannung versorgt. Sobald das SDR in Betrieb genommen wird, ist der Lokaloszillator mit 28,8 MHz getaktet. Durch das USB-Multiplexing werden die SDR nicht gleichzeitig, sondern seriell gestartet. Dadurch werden die Lokaloszillatoren ebenfalls zu unterschiedlichen Zeitpunkten mit der Taktfrequenz angeregt, was zu Frequenz-und Phasenunterschieden in den erfassten IQ-Daten führt. Um die SDR-Empfänger in ihrer Frequenz zu synchronisieren, müssen die Lokaloszillatoren synchronisiert werden. Es ist möglich dies durch einen externen Taktgeber oder durch eine Daisy Chain der internen Quarzkristalle zu realisieren. Die Daisy Chain beschreibt ein serielles Bussystem, bei dem das Signal von einem Master-Port durch mehrere serielle Slave-Ports durchgeschleift wird. In der Thesis wird die Frequenzsynchronisation der beiden SDR mit einer Daisy Chain realisiert.

In Abb. 6.4 ist der Quarz an xtal_i (Pin 8) und xtal_o (Pin 9) angeschlossen.

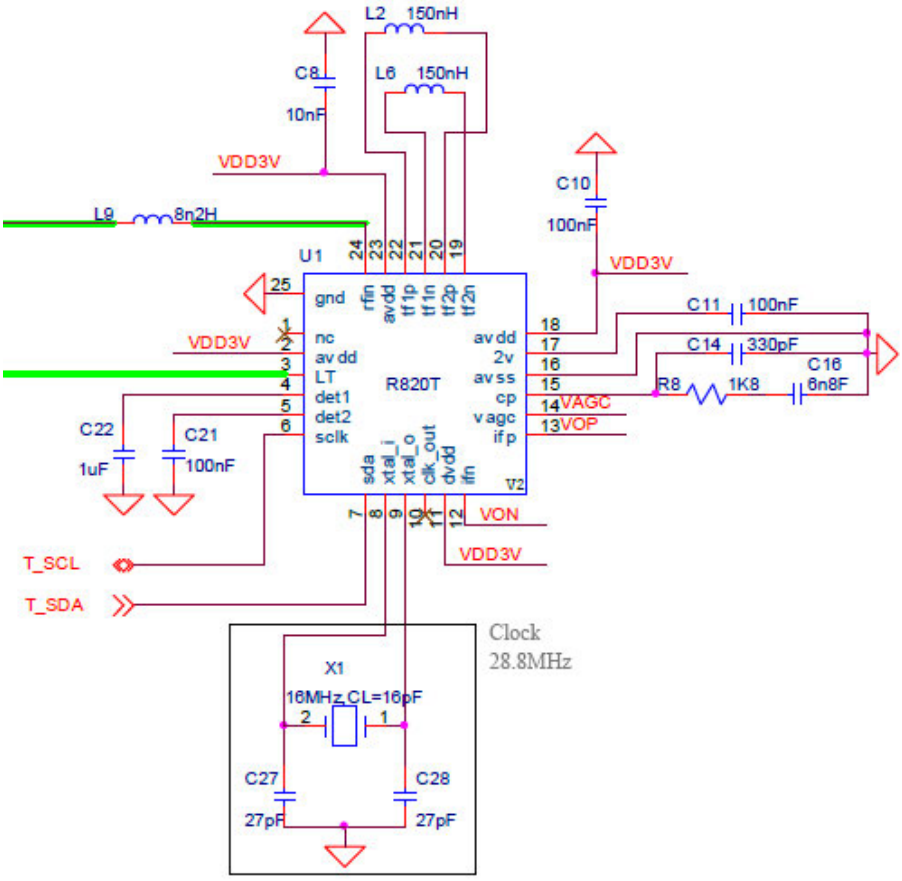


Abbildung 6.4: Ausschnitt aus dem Schaltplan vom R820T/2 [30, S.18].

Zur Durchführung der Daisy Chain muss der Quarzkristall vom Slave-SDR entfernt werden. Daraufhin wird eine Leitung vom xtal_o des Master-SDR zum xtal_o des Slave-SDR gelegt. Damit die beiden SDR-Empfänger nicht über das USB-Hub geerdet sind, wird eine Verbindung zwischen den beiden MCX-Anschlüssen hinzugefügt. Das Ergebnis ist in Abb. 6.5 zu sehen.

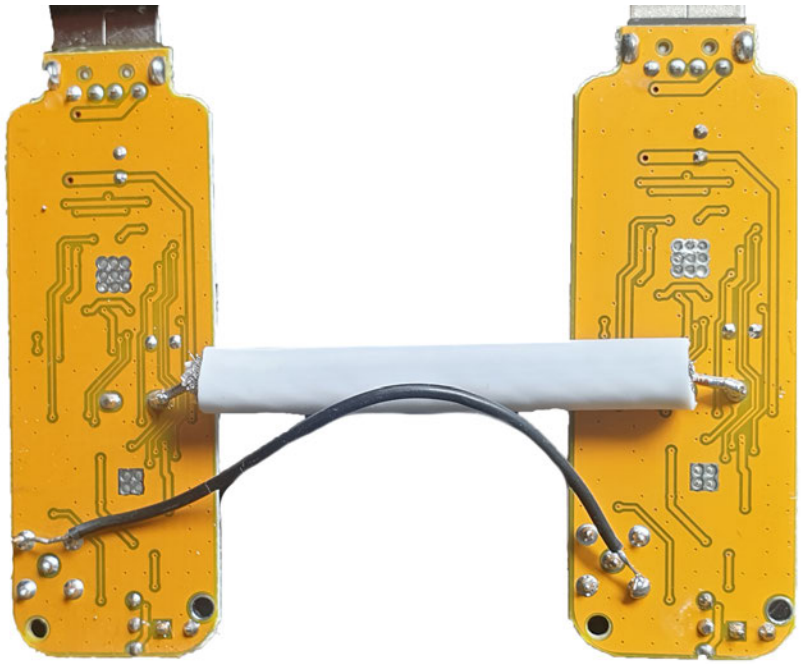


Abbildung 6.5: Daisy Chain zwischen den Taktausgängen von zwei SDR-Empfängern.

Bei erfolgreicher Implementierung der Daisy Chain muss im xtal_o des Slave-SDR die Taktfrequenz von 28,8 MHz anliegen. Dies ist in Abb. 6.6 zu sehen. Die größte Schwäche der Daisy Chain liegt in ihrer Fehleranfälligkeit. Wenn ein Signalfehler im Master-Port oder in einem der Slave-Ports vorliegt, wird dieser durch den gesamten Signalweg gezogen und im schlimmsten Fall sogar verstärkt. Es können in Summe drei SDR in einer Daisy Chain verkettet werden, bis der Takt nicht mehr fehlerfrei übertragen wird. Deshalb ist ab vier eingebundenen SDR-Empfängern ein externer Taktgeber notwendig. Die Frequenzsynchronisation der IQ-Signale kann in Abb. 6.7 betrachtet werden. Für den mathematischen Beweis der Frequenzsynchronisation, kann die Korrelation der Fourier-Transformierten beider Signale verwendet werden. Wenn die Korrelation keinen Versatz aufzeigt, sind die Signale synchron.

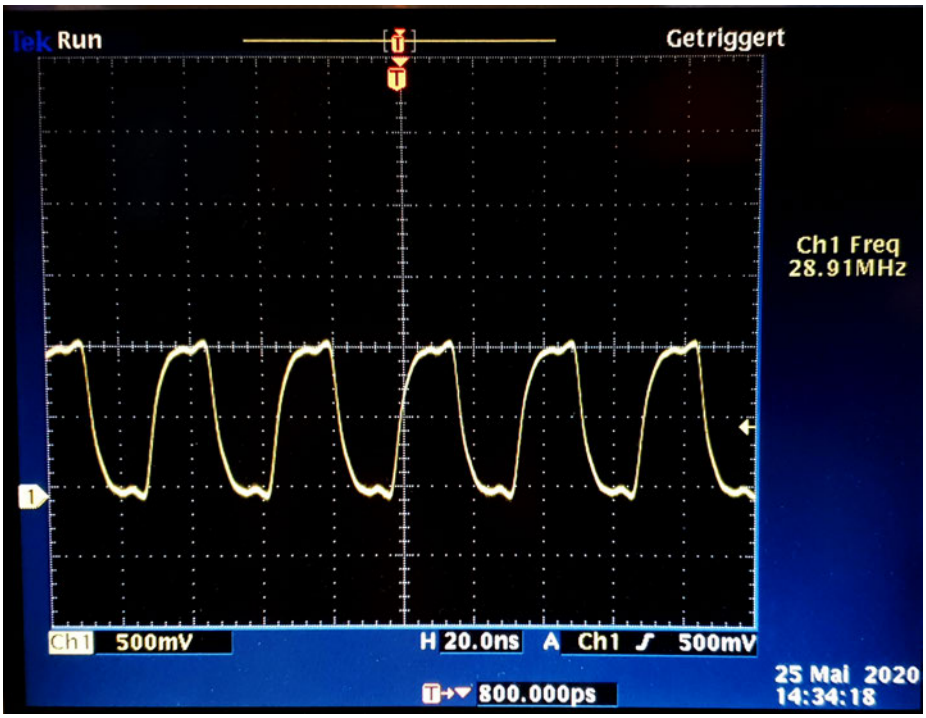


Abbildung 6.6: Die gemessene Taktfrequenz am Slave-SDR (mit Tektronik TDS 3012).

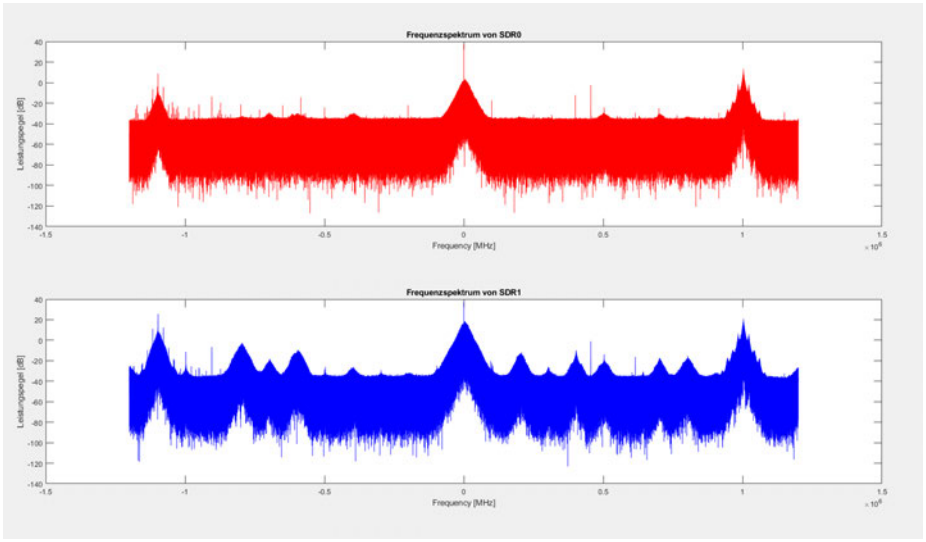


Abbildung 6.7: Frequenzsynchronisation von SDR0 und SDR1 in MATLAB visualisiert.

6.4 Frequenz-Zittern

J.-M Friedt beschrieb, dass die Lokaloszillatoren mit einer Phasenregelschleife (engl.: phase locked loop; PLL) multipliziert werden, um die Trägerfrequenz zu erreichen [17].

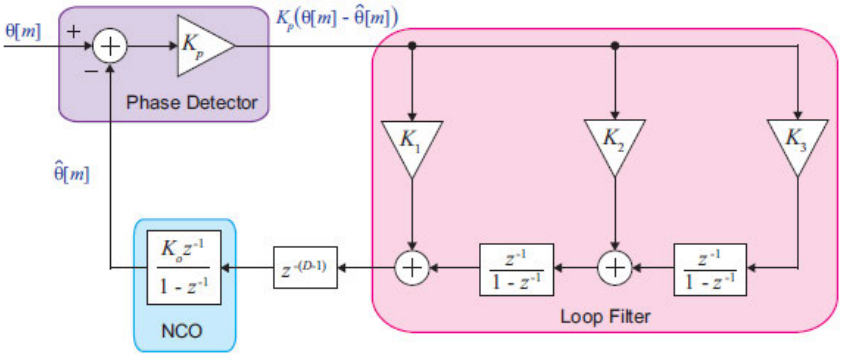


Abbildung 6.8: Signalflussdiagramm eines PLL [10, S. 258].

Ein PLL besteht in seiner einfachsten Form aus einem Phasenkomparator, einem Schleifenfilter und einem numerisch gesteuerten Oszillator (engl.: numerically controlled oscillator; NCO). Der Phasenkomparator vergleicht die Phasenlage zwischen dem Eingangssignal des PLL und dem Ausgangssignal des NCO und schickt das Fehlersignal in den Schleifenfilter. Die Übertragungsfunktion des Filters und der Wirkungsgrad des Oszillators sind abhängig von dem Wirkungsbereich des PLL [10, S. 257-260]. Mit Frequenz-Zittern (engl.: dithering) erreicht der PLL feine Frequenzübergänge, die in der Quantisierung der AD-Wandlung entscheidend sind. Problematisch ist dabei, dass die PLL in beiden SDR-Empfängern nicht gleich getaktet sind und damit ein Phasenversatz zwischen beiden IQ-Signalen bei der AD-Wandlung entsteht [17, S. 2].

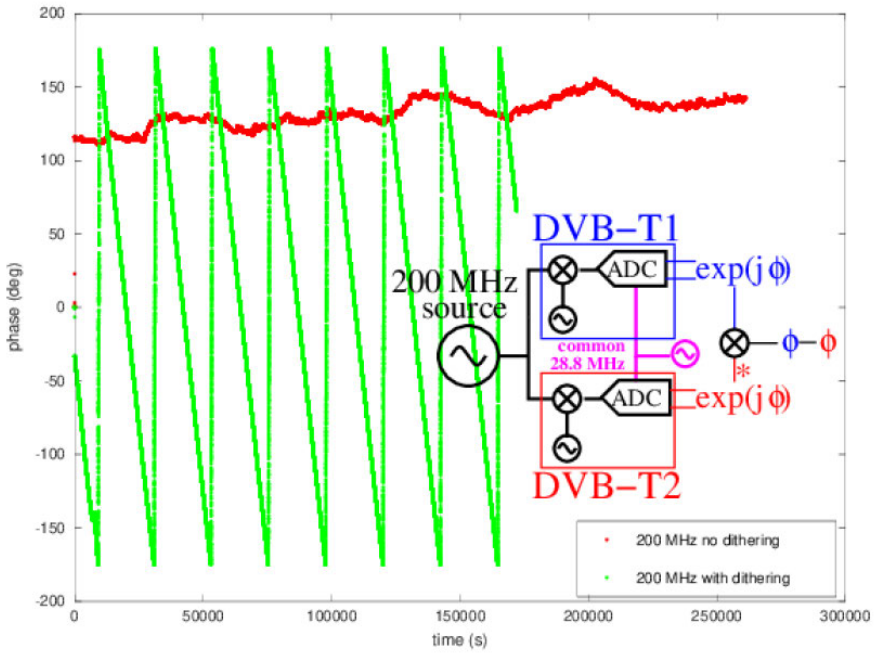


Abbildung 6.9: Vergleich des Phasenverlaufs zwischen eingeschalteten Dithering (grün) und abgeschalteten Dithering (rot) [17, S. 2].

Abbildung 6.9 zeigt, dass sich mit aktiviertem Dithering die Phase in Abhängigkeit von der Zeit ändert. Mit deaktiviertem Dithering verhält sich die Phase linearer. Die verbleibende Veränderung der Phase liegt an den temperaturabhängigen Bauteilen und nimmt mit der Betriebsdauer und Temperatur zu [17, S.2]. Das Dithering lässt sich innerhalb der librtlsdr Bibliothek ändern. Die librtlsdr Bibliothek wird in MATLAB im *Communications Toolbox Support Package for RTL-SDR Radio* eingebunden. Innerhalb dieser Toolbox ist der Zugriff auf die librtlsdr Bibliothek nicht möglich, weshalb die modifizierte Bibliothek mit deaktiviertem Dithering nicht in MATLAB eingesetzt werden kann. Unter Linux kann das original librtlsdr-Verzeichnis [33] einfach gebaut, kompiliert und in GNU Radio verwendet werden. Die Modifizierung vom librtlsdr-Verzeichnis ist im GitHub von *keenerd* dokumentiert [45].

6.5 Sample-Synchronisation

Nach der Frequenzsynchronisation müssen die SDR-Empfänger zeitlich synchronisiert werden.

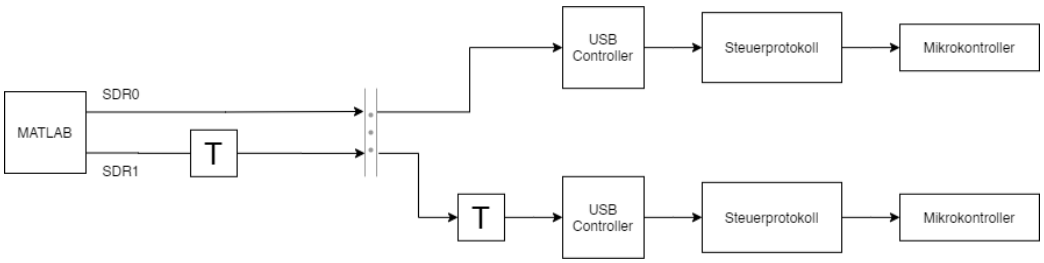


Abbildung 6.10: Signalweg von MATLAB zu SDR0 und SDR1.

In Abb. 6.10 wird davon ausgegangen, dass SDR0 die Adressierung 0 und SDR1 die Adressierung 1 von MATLAB erhalten hat. Sobald die Simulation zur Erfassung der Signale von MATLAB initiiert wird, steuert MATLAB mittels der libusb über den USB-Bus, den USB-Controller des SDR0 an. Nachdem der SDR0 angesteuert wurde, steuert MATLAB die zweite USB-Adresse an. Abhängig von der Rechenzeit und des Signalweges von SDR0 wird das Signal zum SDR1 auf dem Weg zum USB-Bus das erste Mal verzögert. Die zweite Verzögerung entsteht durch das USB-Multiplexing. Das bedeutet, dass SDR0 und SDR1 einen Sample-Versatz in Abhängigkeit der Rechenzeit und USB-Adressierung haben.

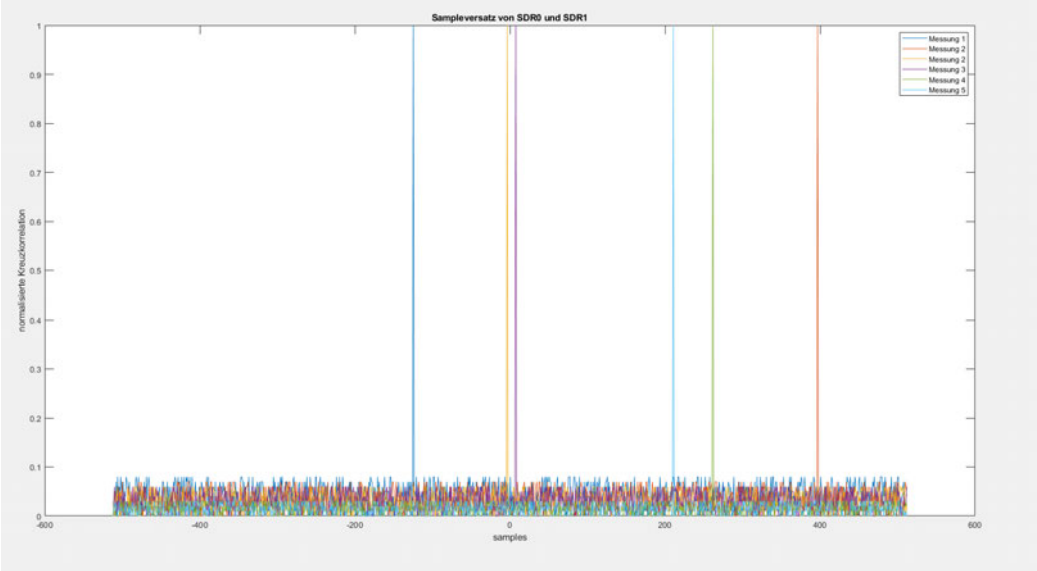


Abbildung 6.11: Sample-Versatz von SDR0 und SDR1 in sechs Messungen.

Um das Verhalten der zeitlichen Verzögerung zwischen den beiden SDR-Empfängern zu analysieren, wird an beide SDR eine gemeinsame Antenne angeschlossen. Anschließend werden die Daten vom SDR0 und SDR1 durch eine Korrelation miteinander verglichen. Das Sample, in dem die maximalen Korrelation auftritt, gibt die Länge der Verzögerung an. Die Daten wurden mit einer Abtastfrequenz von $f_s = 2,4MHz$, einer Trägerfrequenz von $f_c = 103,6MHz$ und einer Framelänge von 1024 Samples aufgenommen. In Abb. 6.11 wird gezeigt, dass der zeitliche Versatz nicht konstant ist, sondern bei jeder Messung neu gesetzt wird. Während der Laufzeit der Simulation bleibt die zeitliche Verzögerung allerdings konstant. Aus dieser Erkenntnis kann eine mögliche Synchronisation gewonnen werden.

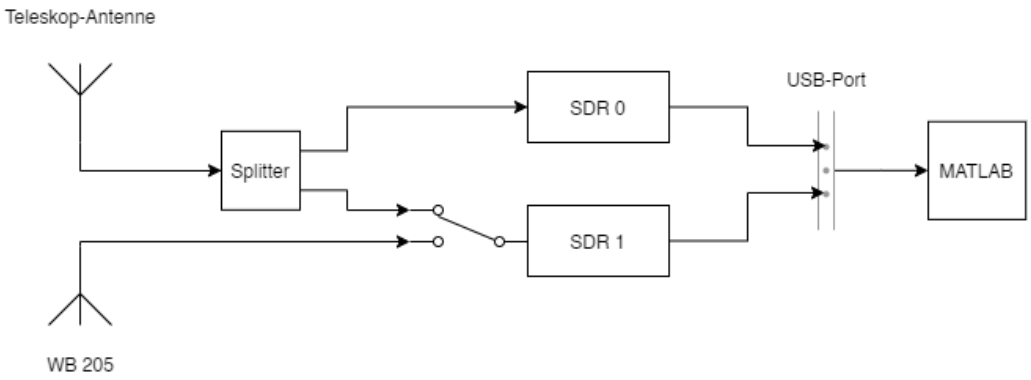


Abbildung 6.12: Blockschaltbild vom Synchronisationsaufbau.

Zu Beginn der Messung bekommen beide SDR-Empfänger das Antennensignal der omnidirektionalen Teleskop-Antenne. SDR1 wird danach umgesteckt auf die Beobachtungsantenne WB 205. In MATLAB kann, wie in Abb. 6.13 zusehen, ein deutliches Synchronisationssignal (Sync) erkannt werden. Beim Umstecken der Antenne wird nur das Eigenrauschen des SDR übertragen (Switch). Der restliche Anteil im Signal ist das Empfangssignal mit dem relevanten Signalinhalt. Die Signale werden mit einem Buffer anhand des Synchronisationssignals synchronisiert und auf das Empfangssignal begrenzt zugeschnitten. Abhängig davon welches Signal vor- oder nacheilt wird das Signal mit einer Anzahl von Nullen, dem Versatz entsprechend, ergänzt. Das Buffer-System ist in Anhang A.5 hinterlegt. Der genaue Programmverlauf wird in Kapitel 8.1 betrachtet. In Abb. 6.14 und Abb. 6.15 wird der zeitliche Versatz und der Effekt der Synchronisierung dargestellt.

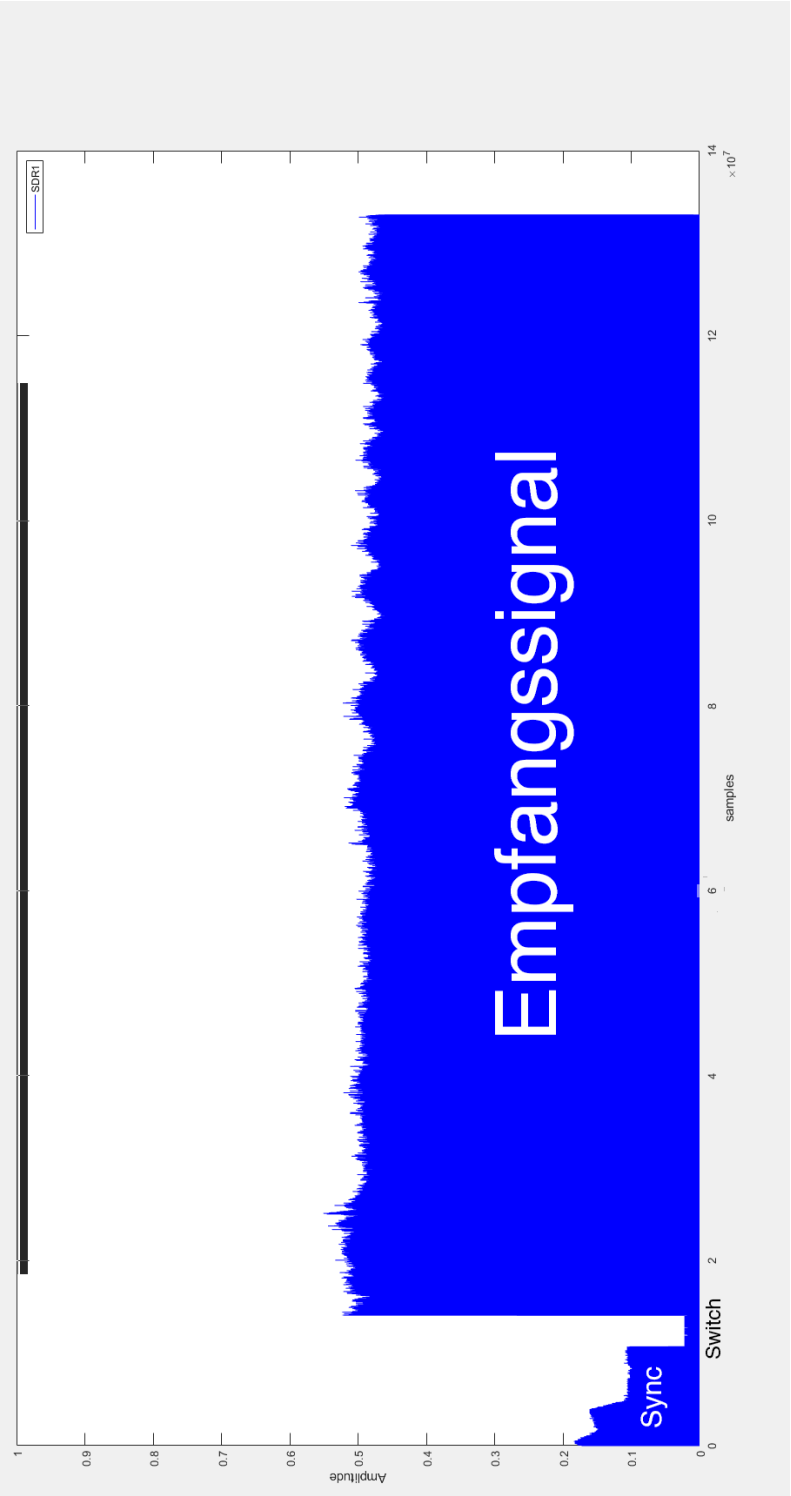


Abbildung 6.13: Aufteilung des Beobachtungssignals.

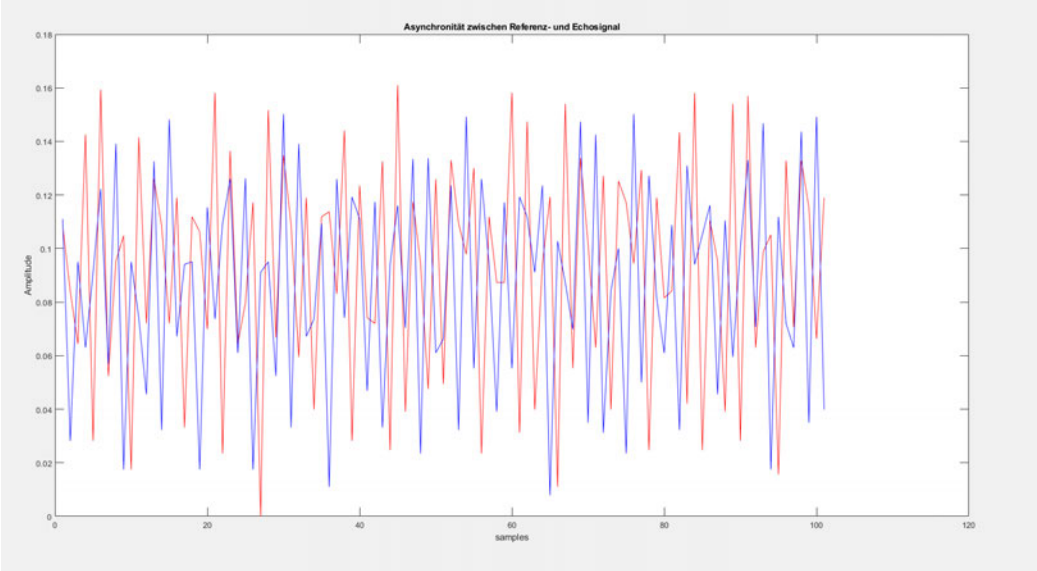


Abbildung 6.14: Sample-Versatz von SDR0 (rot) und SDR1 (blau).

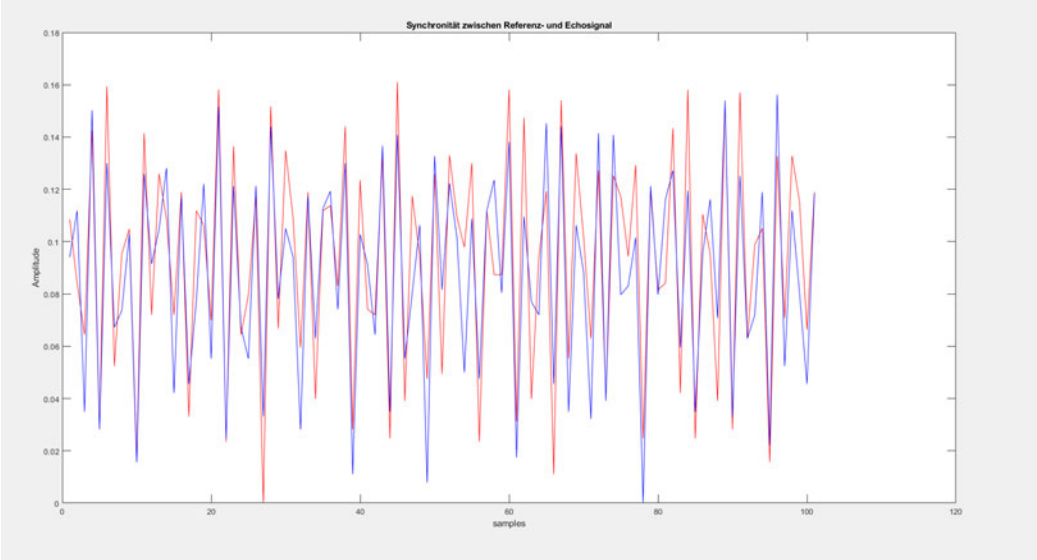


Abbildung 6.15: Korrigierter Sample-Versatz von SDR0 (rot) und SDR1 (blau).

6.6 Zusammenfassung

Dieses Kapitel behandelt die notwendige Synchronisation der SDR-Empfänger um ein kohärentes System zu erstellen. Es wird dabei auf die Probleme und Lösungen der Asynchronität im Zeit und Frequenzbereich eingegangen und die Synchronität zwischen Sender und Empfänger hergestellt.

7 Feldmessung

7.1 Messposition

Die wichtigste Entscheidung der Feldmessung liegt in der Positionierung des Empfängersystems. Es muss dabei beachtet werden, dass die in Kapitel 2.2 behandelte Radar-Geometrie erfüllt werden kann. Die Positionierung ist vom Überwachungsraum und der Position der Sendeanlage abhängig.

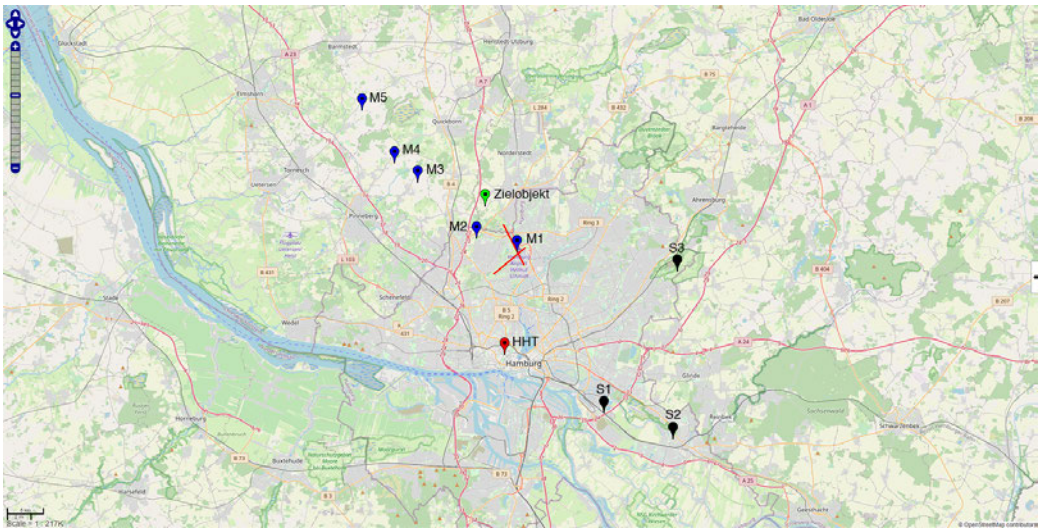


Abbildung 7.1: Verteilung von Sendeanlagen und Messpunkten im Raum Hamburg um den Hamburger Flughafen herum angeordnet.

In Hamburg existieren vier kommerzielle Sendeanlagen. Der Heinrich-Hertz-Turm (rot, HHT), Sender Billwerder-Moorfleet (schwarz, S1), Fernmeldeturm Hamburg-Lohbrügge (schwarz, S2) und der Sender Hamburg-Rahlstedt (schwarz, S3). Aufgrund der Zugänglichkeit wurde der Messpunkt M1 am Flughafen Hamburg gewählt. Die Sendeanlage S1 und S2 können nicht verwendet werden, da sie mit der Messposition M1 fast auf einer Linie zum Überwachungsraum und dem potenziellen Flugobjekt (grün) liegen und keine ausreichende Radar-Geometrie hergestellt werden kann. Sen-

deanlage S3 wird aufgrund des Sendeinhaltes ausgeschlossen. Nach dem Ausschlussverfahren wird der Heinrich-Hertz-Turm als Sendeanlage verwendet. Weitere mögliche Messpunkte (M2-M5) sind in Abb. 7.1 eingezeichnet und verlaufen längs des Überwachungsraumes um Quickborn herum. Die geografischen Entfernungen von ihrer Längengrad- und Breitengradlage, werden mit der Haversine Formel errechnet.

$$d = 2r \cdot arcsin\left[\sqrt{sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + cos(\phi_1)cos(\phi_2)sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right] \tag{7.1}$$

Der MATLAB-Code für die Haversine Formel und die interaktive Karte ist im Anhang A.4 und A.3 hinterlegt.

7.2 Messaufbau

Vor der Messung wurden die bereits benannten Konfigurationen des Empfängersystems durchgeführt. Die Teleskop-Referenzantenne wurde frei im Sichtfeld zur Sendeanlage installiert. Die Beobachtungsantenne WB 205 wurde in Abhängigkeit ihrer Richtwirkung aus Abb. 3.7 installiert. Dabei wurde darauf geachtet, die Antenne im Winkel von 80° zur Sendeanlage aufzustellen, um möglichst wenig Direktsignalanteil im Echosignal zu erhalten und gleichermaßen die Hauptkeule auf den Überwachungsbereich zu richten.

Abbildung 7.2: Messaufbau zur Signalerfassung am Flughafen Hamburg.

Der Trägerfrequenz-Versatz aus Kapitel 6.2 wurde gemessen und korrigiert. Die Signale wurden abschließend mit dem im Kapitel 5.4 vorgestellten Verfahren analysiert. Dabei wurde darauf geachtet, dass die Trägerfrequenz im Spektrum korrekt zu erkennen ist und das Signal im Zeitbereich nicht übersteuert oder ein zu geringer Signalpegel übertragen wird. Die Signale wurden dementsprechend gepegelt. Zudem wurde in das demodulierte Signal rein gehört, um zu überprüfen, ob der Signalinhalt korrekt wiedergegeben wird oder nur aus Rauschen besteht. Die Aufnahmeparameter wurden wie folgt gesetzt:

SDR0-Adresse = 0

SDR1-Adresse = 1

$f_c = 97,1\ MHz$

$f_s = 2,4\ MHz$

Vorverstärkung SDR0 = 10 dB

Vorverstärkung SDR1 = 10 dB

Frequenzkorrektur SDR0 = 6 PPM

Frequenzkorrektur SDR1 = 8 PPM.

Die Abtastfrequenz wurde auf 2,4 MHz gesetzt, weil der Sendeempfänger R820T2 eine maximale Bandbreite von 2,4 MHz aufnehmen kann. Die Trägerfrequenz von 97,1 MHz entspricht der Frequenz vom Radiosender Energy Hamburg. Ein Radiosender, der vom Heinrich-Hertz-Turm isotrop ausgestrahlt wird und Pop-Musik als Hauptprogramm führt.

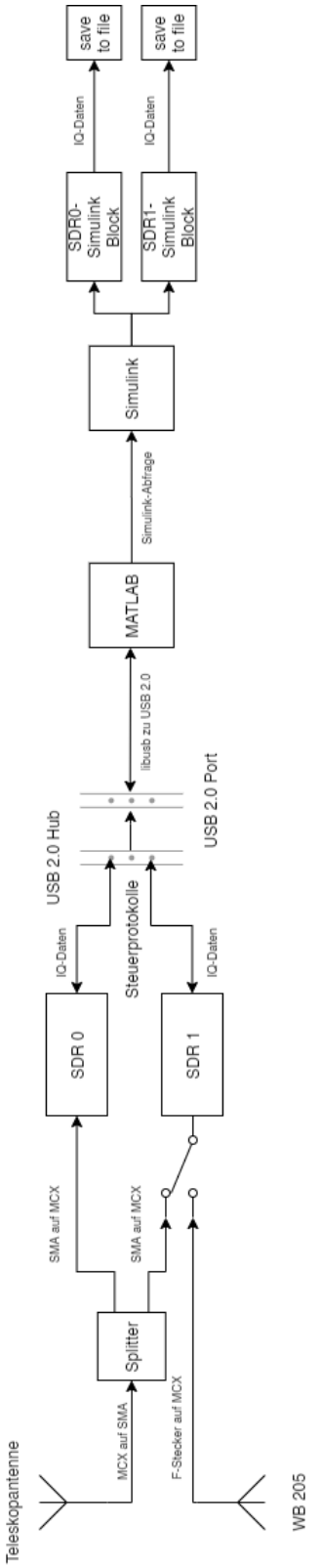


Abbildung 7.3: Signalweg des Messaufbaus.

Beim Messvorgang wurde darauf geachtet, ob es Bewegungen im ADS-B (engl.: Automatic Dependent Surveillance-Broadcast) gibt. ADS-B ist der Funk der Flugsicherung, indem Flugobjekte ihre Position mitteilen. Sobald ein Flugobjekt im ADS-B sichtbar war, wurde die Simulink-Simulation mit dem geteilten Antennensignal der Referenzantenne für das Synchronisationssignal gestartet.

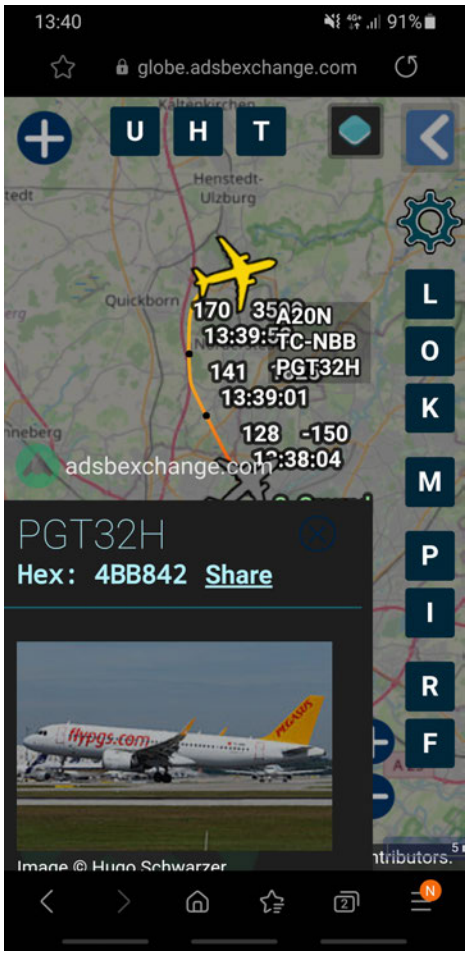


Abbildung 7.4: Screenshot aus dem ADS-B von ADS-B Exchange [46].

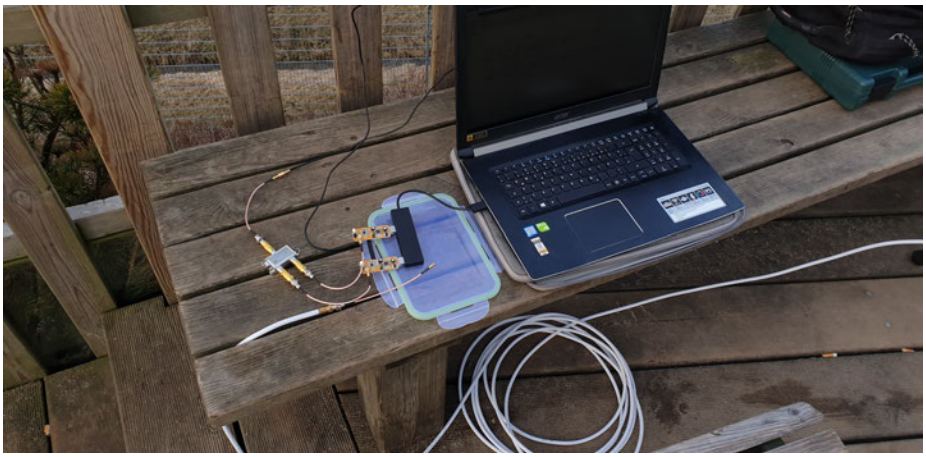


Abbildung 7.5: Signalsplitter zum Aufteilen des Referenzantennensignals aus SDR0 und SDR1.

Danach wird die Beobachtungsantenne an den SDR1 angeschlossen und die Aufnahme des relevanten Echosignal beginnt. Die IQ-Signale werden über die libusb an MATLAB gesendet und dort in die Simulink-Umgebung überführt, bis sie in einem Speicher-Block in eine Datei als Zeitreihe gespeichert werden.

7.3 Zusammenfassung

In diesem Kapitel wurde die Auswahl der Positionierung für die Messung der Echosignale und die Wahl der Sendeanlage behandelt. Im Weiteren ging es um die detaillierte Beschreibung des Messvorgangs anhand des Blockschaltplans des Messaufbaus.

8 Signalverarbeitung

Die Signalverarbeitung des Passiv-Radars in MATLAB wird in zwei Hauptaspekte geteilt. Die zeitliche Synchronisation und die Ambiguitätsfunktion, mit Einbindung des CLEAN-Algorithmus, sollen die zielführenden Verarbeitungsschritte sein.

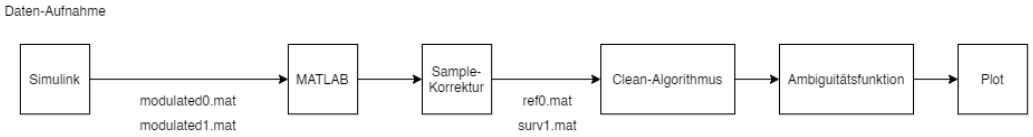


Abbildung 8.1: Blockschaltbild der angewandten Signalverarbeitung in MATLAB.

Die Signalverarbeitung knüpft direkt an die Aufnahme der Referenz- und Echosignale an. Als erster Schritt werden die Signale zeitlich in der Sample-Korrektur synchronisiert. Anschließend wird der CLEAN-Algorithmus angewendet um zuletzt die Ambiguitätsfunktion zu bilden. Im Idealfall können in der Ambiguitätsfunktion Zielobjekte gesehen werden.

8.1 Sample-Synchronisation

Die Sample-Korrektur behebt die zeitliche Asynchronität zwischen dem SDR0 und SDR1. In Abb. 8.3 ist der Signalweg innerhalb der Zeitsynchronisation zusammengefasst. Die modulierten und asynchronen Signale von SDR0 und SDR1 werden nach der Aufnahme in MATLAB geladen. Beide Zeitreihen ([Data]x1024) werden in Arrays und Vektoren (1x[Data]) umgeformt. Das Echosignal wird visualisiert und die Grenzen vom Synchronisationssignals und Empfangssignal werden grafisch erfasst. Der Bereich des Synchronisationssignal wird aus beiden Vektoren für die Kreuzkorrelation verwendet. In der Kreuzkorrelation wird die Signalverzögerung in Samples bestimmt. Dabei definiert das Maxima der Korrelation den Punkt, an dem die beiden Signale maximal korrelieren.

Die maximale Korrelation ergibt die zeitliche Verzögerung [47].

$$\Delta t = \tau_{peak} = arg\ max(\rho_{sdr0,sdr1}(\tau)) \tag{8.1}$$

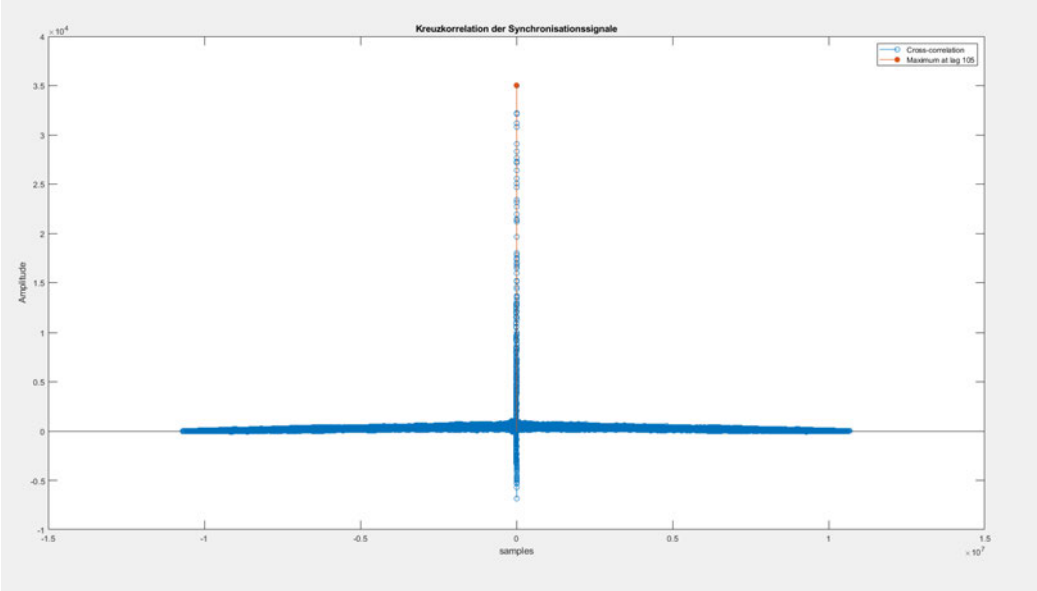


Abbildung 8.2: Kreuzkorrelation und Maxima von desync0 und desync1.

Aus der Kreuzkorrelation in Abb. 8.2 kann die Verzögerung deutlich abgelesen werden. Eine positive Verzögerung bedeutet, dass SDR0 dem SDR1 voraus eilt und SDR1 nach rechts verschoben werden muss. Eine negative Verzögerung bedeutet dementsprechend eine Verschiebung nach links. Nachdem die Verzögerung in der Kreuzkorrelation bestimmt wurde, wird sie in einem Buffer korrigiert. Dieser wird in einer If-Schleife realisiert und verschiebt SDR1 abhängig vom Vorzeichen. Hierzu werden Nullen der Anzahl der Verzögerung hinzugefügt. Nach der Synchronisation durch den Buffer werden die Signale auf ihr Empfangssignal begrenzt und als ref0.mat (Referenzsignal) und surv1.mat (Echosignal) gespeichert. Der Programmcode für die Synchronisation ist in Ablage A.6 hinterlegt.

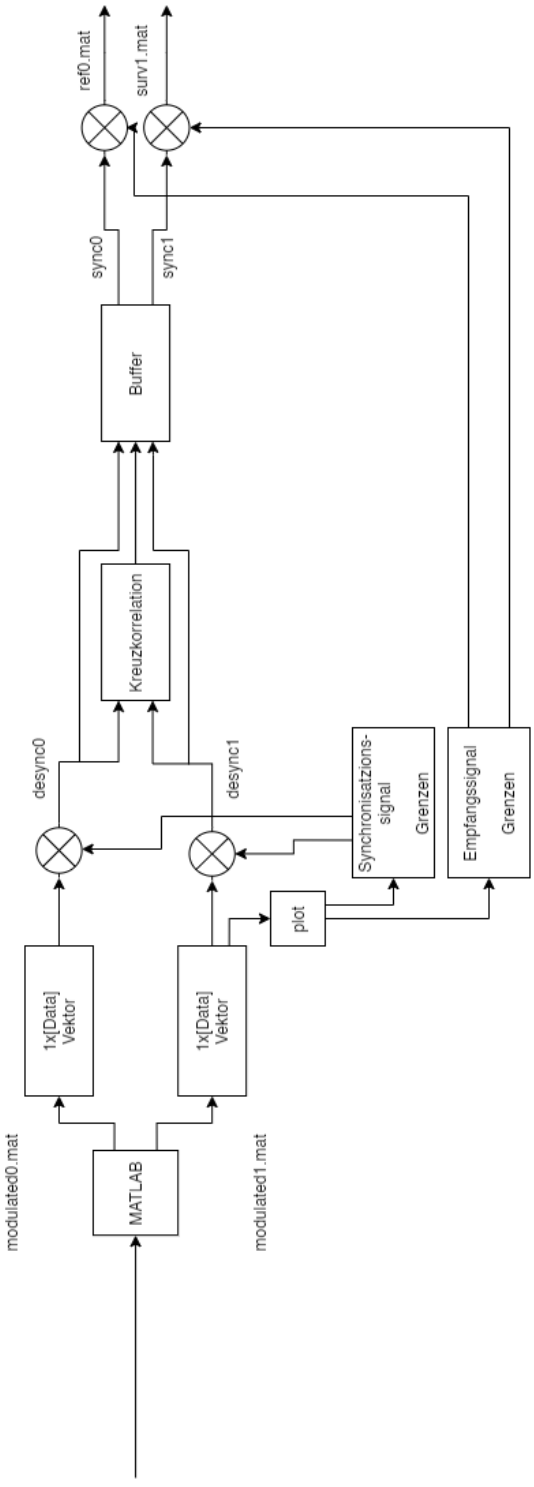


Abbildung 8.3: Blockschaltbild der angewandten Signalverarbeitung in MATLAB.

8.2 CLEAN-Algorithmus

In Malanowskis Buch über die Signalverarbeitung vom bistatischen Passiv-Radar werden unterschiedlichste iterative Filter und Block-Filter vorgestellt [4, S. 171-200]. In dieser Thesis wird der CLEAN-Algorithmus aufgrund seines einfachen Aufbaus verwendet. In der Passiv-Radartechnik müssen Algorithmen verwendet werden, um das Direktsignal aus dem Beobachtungssignal zu filtern. Der Direktpfad besitzt im Großteil der Fälle eine größere Signalleistung als die Reflexionen vom Zielobjekt, was zu einer Überdeckung der Echosignale führen kann.
Der CLEAN-Algorithmus betrachtet die Maximalwerte der Ambiguitätsfunktion [4, S.194].

$$\{R_0, V_0\} = arg\ max|\psi(R, V)| \tag{8.2}$$

Sollten DPI im Beobachtungssignal enthalten sein, die mögliche Echosignale verdecken, werden diese bei Entfernung $\{0\}$, Doppler $\{0\}$ sichtbar. Die bistatische Entfernung und Geschwindigkeit werden dazu verwendet, ein starkes Echosignal zu generieren, indem das Referenzsignal in Abhängigkeit von R_0 und V_0 versetzt wird.

$$x_m(t) = x_r \cdot (t - \frac{R_0}{c}) \cdot e^{-j2\pi \frac{V_0}{\lambda} t} \tag{8.3}$$

Somit ist das starke Echosignal wie folgt definiert:

$$x'_e(t) = x_e(t) - x_m(t). \tag{8.4}$$

Unter der Berücksichtigung, dass R_0 und V_0 betrachtet werden, kann $x_m(t) = x_r(t)$ definiert werden. Der CLEAN-Algorithmus gibt daraus vor, dass das Referenzsignal vom Echosignal abgezogen wird, um ein neues Echosignal ohne DPI zu generieren.

8.3 Zielobjekt-Erfassung

Der Kern zur Erfassung von Zielobjekten liegt in einer klar definierten und gesäuberten Ambiguitätsfunktion. Abbildung 8.5 zeigt die Entwicklung der Ambiguitätsfunktion zur Erfassung von Zielobjekten in MATLAB. Die synchronisierten Signale werden in Arrays ([Data]x1024) umgewandelt und anschließend mit einem Tiefpass gefiltert. Der Tiefpass verhindert, dass benachbarte Trägerfrequenzen von anderen Sendeanlagen das Signalbild stören könnten.

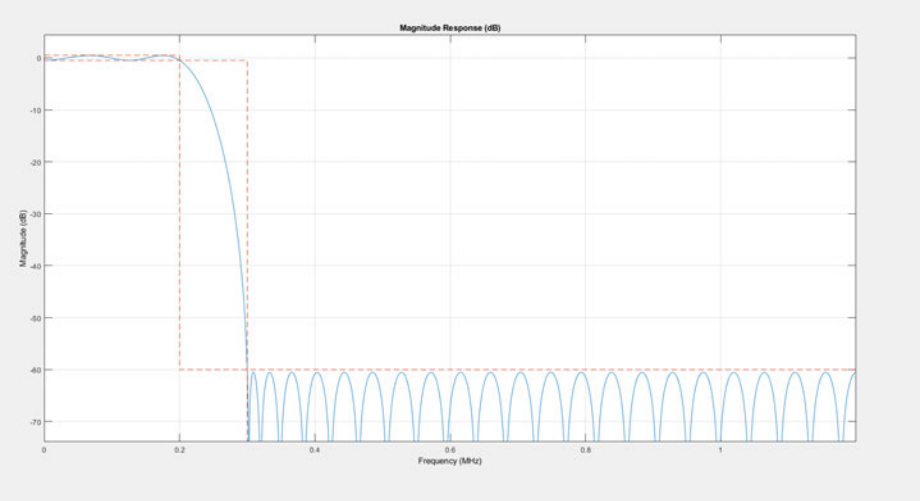


Abbildung 8.4: Filterkurve des verwendete Tiefpasses.

Anschließend wird zur Anwendung des CLEAN-Algorithmus das Referenzsignal vom Echosignal abgezogen. Danach werden die Arrays für die Kreuzkorrelation dimensioniert und anschließen korreliert. Das Korrelationssignal wird dann in ihrer Darstellung der Entfernung auf 20 Km (160 Samples) begrenzt. Für die Dopplerinformation wird das Korrelationssignal mit der schnellen Fourier-Transformation in den Frequenzbereich überführt. Zum Schluss wird die Dopplerinformation in Abhängigkeit der Entfernung in der Ambiguitätsfunktion abgebildet. Der vollständige MATLAB-Code ist in Anhang A.7 hinterlegt. Ein Blockschaltbild des gesamten Systems ist in Anhang D zu finden.

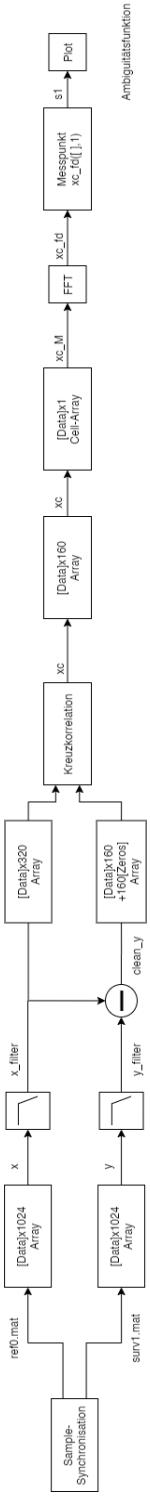


Abbildung 8.5: Blockschaltbild zur Entwicklung der Ambiguitätsfunktion.

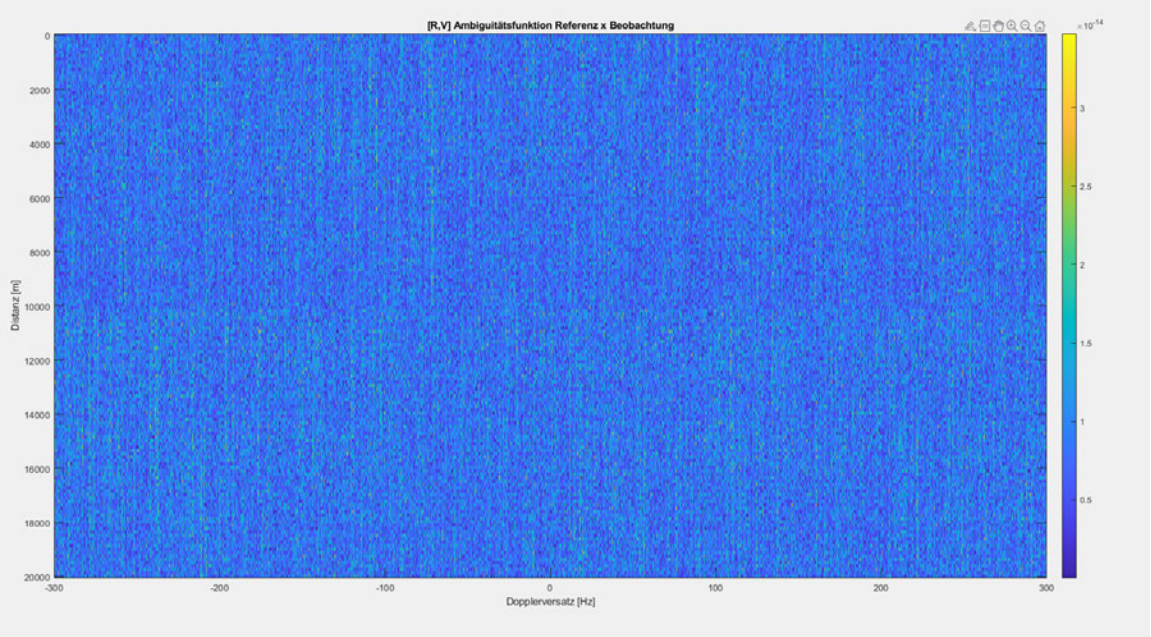


Abbildung 8.6: Ambiguitätsfunktion der Kreuzkorrelation ohne CLEAN-Algorithmus.

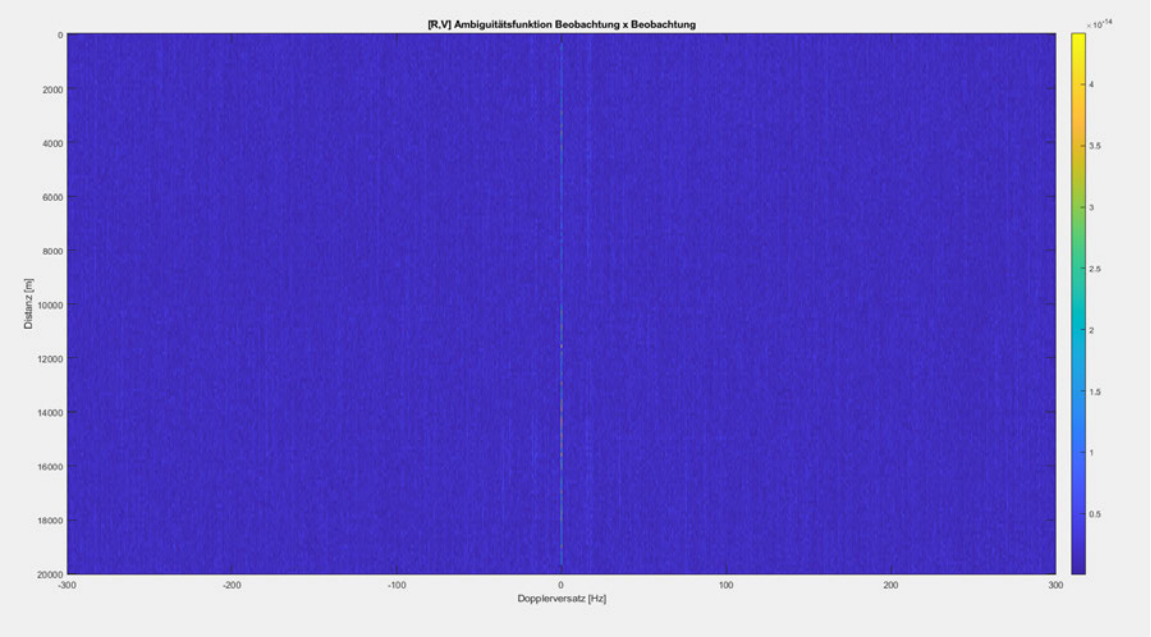


Abbildung 8.7: Ambiguitätsfunktion der Autokorrelation des Echosignals ohne CLEAN-Algorithmus.

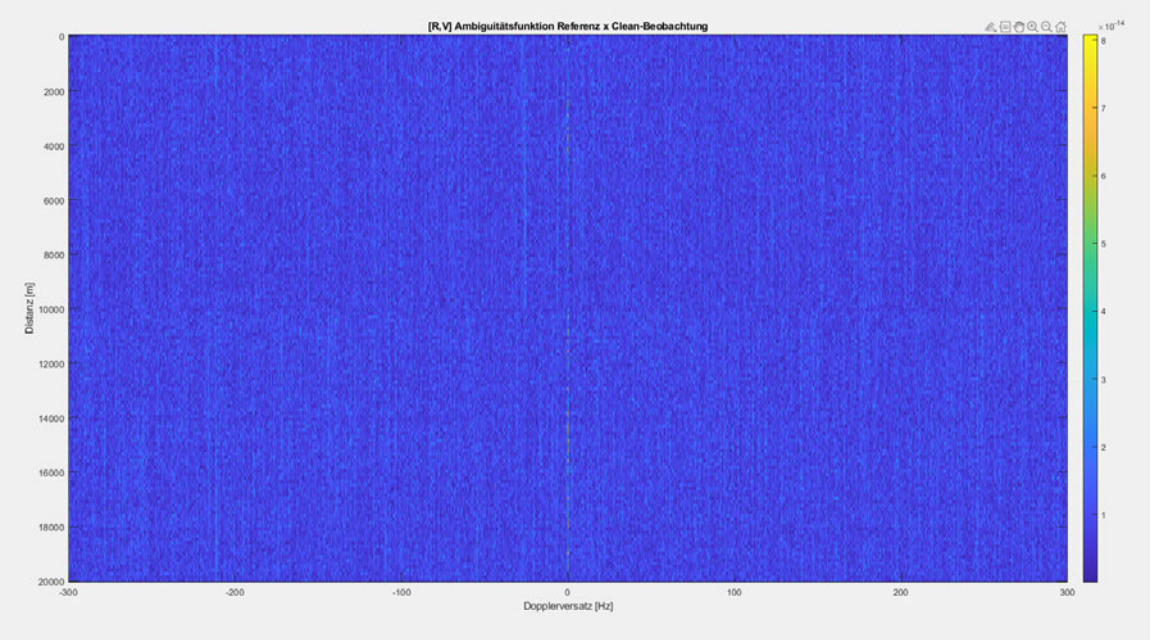


Abbildung 8.8: Ambiguitätsfunktion der Kreuzkorrelation mit CLEAN-Algorithmus.

Während in der Ambiguitätsfunktion der Autokorrelation aus Abb. 8.7 die DPI deutlich sichtbar ist, lässt Abb. 8.6 keine DPI erkennen. Durch den CLEAN-Algorithmus in Abb. 8.8 sind die DPI besser zu erkennen, als im direkten Vergleich zu der Kreuzkorrelation ohne den CLEAN-Algorithmus. Leider sind in keiner Darstellung Zielobjekte erkennbar. Die möglichen Gründe des Fehlers werden in Kapitel 10 thematisiert.

8.4 Zusammenfassung

Dieses Kapitel erklärt die Programmierung des Radarsystems in MATLAB und geht auf die Signalverarbeitung in der Zeitsynchronisation sowie Zielerfassung ein. Die Bedeutung und Einbindung des CLEAN-Algorithmus für die Verarbeitung der Signale wurde detailliert erklärt. Die Negativergebnisse bei der Zielerfassung wurden visuell präsentiert.

9 Ergebnisse

Die Realisierung eines auf SDR basierendem Passiv-Radar ist möglich, wie bereits andere Arbeiten in dem Bereich gezeigt haben, jedoch ist sie nicht leicht zu verwirklichen. In dieser Thesis wurden die wichtigsten Eigenschaften vom bistatischen Passiv-Radar konzentriert zusammengetragen und angewendet. Die Implementierung der NESDR Mini 2 RTL SDR in MATLAB unter Windows wurde erfolgreich durchgeführt. Darüber hinaus konnten die ersten Sendesignale durch die Synchronisierung der SDR-Empfänger aufgezeichnet und demoduliert abgespielt werden. Final wurde die Signalverarbeitung nach dem Vorbild von Malanowski für die Erfassung von bewegten Zielobjekten durchgeführt. Bedauerlicherweise ohne positives Endergebnis. Die abschließenden Bilder der Ambiguitätsfunktionen verschiedener Signalverarbeitungen konnten keine Zielobjekte abbilden. Eine erfolgreiche Umsetzung würde bedeuten, dass die Realisierung eines Passiv-Radars zu einem Bruchteil der Kosten eines üblichen Passiv-Radarsystems erfolgen könnte. Die Erfassung von Zielobjekten könnte weiterentwickelt werden zu einer Spurbildung und präziseren Ortung, bis hin zur Prognose des weiteren Flugverlaufes. Durch die Implementierung anderer SDR-Systeme und unterstützenden Software können zukünftig auch andere Herangehensweisen berücksichtigt werden. Meine Thesis möchte ich mit einem persönlichen Fazit abschließen.

10 Fazit

In meinem Fazit werde ich das Negativergebnis erörtern und Weiterentwicklungen dieses Projektes aufzeigen.

Im direkten Vergleich mit Friedts Ergebnissen [17, S. 4] ist deutlich erkennbar, dass meiner Ambiguitätsfunktion die Zielobjekte fehlen. Im Rahmen dieser Thesis konnten

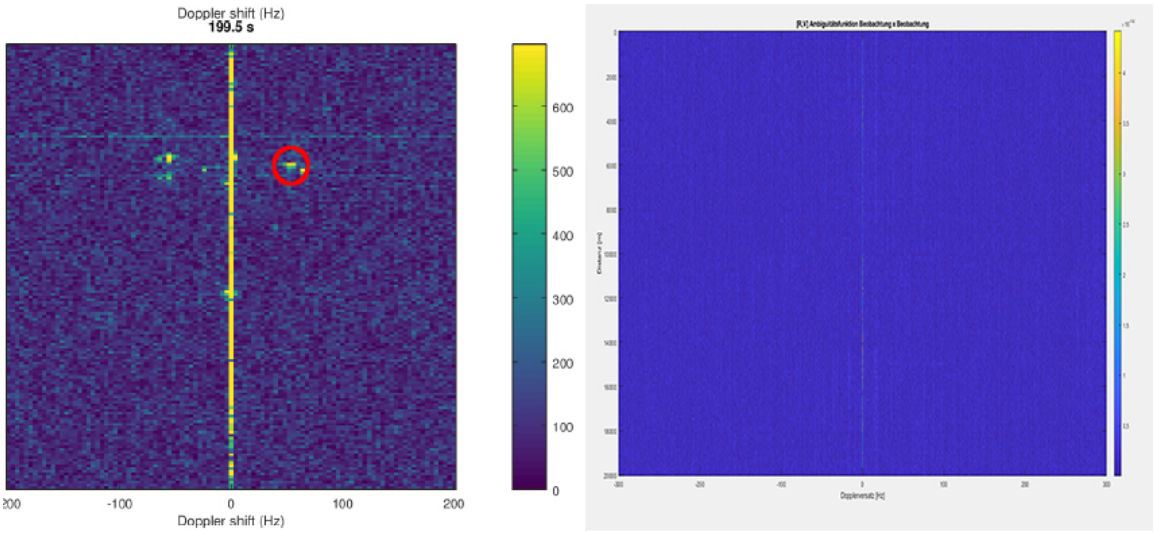


Abbildung 10.1: Ambiguitätsfunktion Friedt (links) und Dittié (rechts).

Fehlerquellen nicht mehr ausgiebig untersucht werden. Deswegen möchte ich mögliche Fehler und Verbesserungen benennen, um einen zukünftigen Erfolg dieses Projektes zu unterstützen. Der erste Ansatzpunkt liegt im Algorithmus. Eine fehlerhafte Anwendung des CLEAN-Algorithmus in Kapitel 8.2 kann eine nicht ausreichende Beseitigung des DPI verursachen. Des Weiteren können iterative Algorithmen bessere Ergebnisse erzielen. Ein weiterer Kritikpunkt wäre die Synchronisation. Zwar ist die Zeitsynchronisation erfolgreich umgesetzt worden, dennoch konnte das Frequenz-Zittern unter Windows nicht im Treiber deaktiviert werden. Eine Phasenverschiebung zwischen SDR0 und SDR1 hat einen Frequenzversatz zur Folge, der ausreichend sein

kann, um den Kontext im Signal nicht erkennbar zu machen. Die letzte Möglichkeit bezieht sich auf die bistatische Entfernungsgleichung und den Einfluss der Richtwirkung der Empfangsantennen. Malanowiski bezieht sich in seinem Buch regelmäßig auf den Versuchsaufbau von zwei stark gerichteten Antennen [4]. Die von mir verwendete Referenzantenne besitzt eine omnidirektionale Richtwirkung. Die WB 205 Beobachtungsantenne weist keine steile Hauptkeule in Kapitel 3.2 vor. Durch eine starke Richtwirkung auf Sender und Ziel können Signalüberlappungen schon innerhalb der Messung vorgebeugt werden. Final ist festzuhalten, dass der Direktsignalanteil in meinem Echosignal zu stark war, um Echosignale sichtbar zu machen und das bei einer Fortsetzung dieses Projektes, an der Beseitigung des DPI angesetzt werden sollte.

A MATLAB-Code

A.1 Polardiagramm

```
%% Antennendiagramm der WB 205 %%

%Rohdaten der Messung
load('E:\Passiv_Radar\03_Richtantenne\
    radiation_pattern_numbers.mat');

%Winkelinformationen der Messdaten
load('E:\Passiv_Radar\03_Richtantenne\ang.mat');

%Singalstärke der Messdaten
load('E:\Passiv_Radar\03_Richtantenne\mag.mat');

%polarpattern(angle,magnitude)
p = polarpattern(ang,mag);
p.AntennaMetrics = 1;
```

A.2 SDRRTLReceiver Syntax

```
%% SDRRTLReceiver Syntax %%

%Überprüfung von angeschlossenen SDR und den SDR-Parametern
sdrinfo

%Erstellung eines SDR-Systemobjekts
rx = comm.SDRRTLReceiver(RadioAddress,'Name',Value);

%Beispiel: SDR0 mit fc= 103.6 MHz, fs= 2.4 MHz @1024 double
    Samples
rxsdr = comm.SDRRTLReceiver('0','CenterFrequency',103.6e6,...
          'SampleRate',2400000,...
          'SamplesPerFrame',1024,...
          'OutputDataType','double');

%Empfang von Signaldaten vom Systemobjekt
rxdata = sdrrx();

%Beispiel innerhalb einer for-Schleife über 1000 Samples
for n=1:1000
    rxdata = rxsdr();
end
release(rxsdr)
```

A.3 Mapping

```
%% Erstellung der geobase offline Karte %%

clear all

%Einführung der des Figure-Objektes

figure('Name','HamburgMap','NumberTitle','off',...
        'WindowState','maximized');

%Einführung der geographischen Achsen
gx=geoaxes;

%geografische Verteilung
geoscatter(53.641849,10.005593,'filled');

%Grenzen der geographischen Achsen
geolimits(gx,[53.5851 53.7448],[9.7871 10.2979]);

%% webmap im Open Street Map baseLayer öffnen %%

webmap('Open Street Map');

%map centering und scaling
%wmcenter(centerLatitude,centerLongitude,zoomLevel)

%Hamburg Airport HAM lat: 53.63308, lon: 9.99481
wmcenter(53.63308,9.99481,11);

%Marker der Sendestationen setzen

%Heinrich-Hertz-Turm
%89.1 MHz Deutschlandfunk Kultur
```

```
33 %91.7 MHz 917xfm
34 %93.0 MHz Freies Sender Kombinat
35 %93.4 MHz delta radio
36 %95.0 MHz Hamburg Zwei
37 %96.0 MHz TIDE/ Hamburger Lokalradio
38 %97.1 MHz Energy Hamburg
39 %98.1 MHz Klassik Radio
40 %100 MHz R.SH Hamburg
41 %104 MHz Radio Hamburg Cityfenster
42
43 wmmarker(53.563142, 9.976109,...
44 'FeatureName','Heinrich-Hertz-Turm',...
45 'OverlayName','Heinrich-Hertz-Turm');
46
47 %Sender Billwerder-Moorfleet
48 %87.6 MHz NDR2
49 %88,7 MHz Deutschlandfunk
50 %89.5 MHz NDR1 Welle Nord
51 %90.3 MHz NDR 90.3
52 %92.3 MHz NDR Info
53 %94.2 MHz N-Joy
54 %99.2 MHz NDR Kultur
55 %103.6 MHz Radio Hamburg
56
57 wmmarker(53.519189, 10.102883,...
58 'FeatureName','Sender Billwerder-Moorfleet',...
59 'OverlayName','Sender Billwerder-Moorfleet','Color','k');
60
61 %Fernmeldeturm Hamburg-Lohbrügge
62 %88.1 Hamburg Zwei
63 %93.7 Radio BOB!
64 %100.9 Energy Hamburg
65 %102.0 R.SH Hamburg
66 %107.7 delta Radio
```

```
67
68 wmmarker(53.499722, 10.190556,....
69 'FeatureName','Fernmeldeturm Hamburg-Lohbrügge',...
70 'OverlayName','Fernmeldeturm Hamburg-Lohbrügge','Color','k');
71
72 %Sender Hamburg-Rahlstedt
73 %106.8 Alsterradio
74
75 wmmarker(53.625933, 10.196228,...
76 'FeatureName','Sender Hamburg-Rahlstedt',...
77 'OverlayName','Sender Hamburg-Rahlstedt','Color','k');
78
79 %Landebahn 05/23 lat: 53.63736, lon: 10.00228 ? lat:
      53.61789, lon: 9.96288
80 %Landebahn 15/33 lat: 53.65491, lon: 9.97482 ? lat: 53.62476,
       lon: 10.00082
81
82 % Markierung der Landebahnen
83 %05/23 (West/Ost)
84
85 Landebahn.lat1=[53.63736 53.61789];
86 Landebahn.lon1=[10.00228 9.96288];
87 wmline(Landebahn.lat1,Landebahn.lon1,'Color','red','LineWidth
      ',3);
88
89 %15/33 (Nord/Süd)
90
91 Landebahn.lat2=[53.65491 53.62476];
92 Landebahn.lon2=[9.97482 10.00082];
93 wmline(Landebahn.lat2,Landebahn.lon2,'Color','red','LineWidth
      ',3);
94
95 %Messstandort1 1.21km Entfernung(unter Einflugsschneise)
96 wmmarker(53.64047,9.99223,'FeatureName',...
```

```
97  '1.21km Entfernung(unter Einflugsschneise)',...
98  'OverlayName','1.21km Entfernung(unter Einflugsschneise)','
      Color','b');
99
100 %Messstandort2 4.11km Entfernung
101 wmmarker(53.650925,9.940260,...
102 'FeatureName','4.11km Entfernung',...
103 'OverlayName','4.11km Entfernung','Color','b');
104
105 %Messstandort3 10.83km Entfernung
106 wmmarker(53.693179,9.865474,...
107 'FeatureName','10.83km Entfernung',...
108 'OverlayName','10.83km Entfernung','Color','b');
109
110 %Messstandort4 13,34km Entfernung
111 wmmarker(53.707492,9.835981,...
112 'FeatureName','13,34km Entfernung',...
113 'OverlayName','13,34km Entfernung','Color','b');
114
115 %Messstandort5 18,27km Entfernung
116 wmmarker(53.746965,9.794742,...
117 'FeatureName','18,27km Entfernung',...
118 'OverlayName','18,27km Entfernung','Color','b');
119
120 %Zielobjekt
121 wmmarker(53.67516,9.95147,...
122 'FeatureName','potenzielles Flugobjekt',...
123 'OverlayName','potenzielles Flugobjekt','Color','g');
```

A.4 Haversine Formel

```
%% Haversine Formel %%

function [hav]=havdistance(latlon_ori,latlon_dest)
latlon_ori=evalin('base','latlon_ori');
latlon_dest=evalin('base','latlon_dest');

%Umrechnung der Koordinaten in Bogenmaß
lat_ori=latlon_ori(1)*pi/180;
lon_ori=latlon_ori(2)*pi/180;
lat_dest=latlon_dest(1)*pi/180;
lon_dest=latlon_dest(2)*pi/180;

deltalat=lat_dest-lat_ori;
deltalon=lon_dest-lon_ori;
radius=6371;

%Haversine Formel
hav=2*radius*asin(sqrt(sin(deltalat/2)^2+cos(lat_ori)*cos(
    lat_dest)*sin(deltalon/2)^2));
end
```

A.5 Buffer Synchronisation

```
%% Buffer Synchronisation %%
clear all
close all
d = 21;
x = [rand(1,100)];

%positiv delay y shifts left
%y = [x(:,abs(d)+1:end), zeros(size(x,1),d)];
%negativ delay, y shifts right
y = [zeros(size(x,1),abs(d)), x(:, 1:end-abs(d))];

figure
plot(x,'r')
hold on
plot(y,'b')
title('Sampleversatz zwischen x und y')
xlabel('samples');
ylabel('Amplitude');
hold off

[xc,lags]=xcorr(x,y);
[~,I] = max(xc);

delay=lags(I);
%delay<0 -> blau später - muss nach links verschoben werden
%delay>0 -> blau früher - muss nach rechts verschoben werden

figure('NumberTitle', 'off', 'Name', 'Delay');
stem(lags,xc,'-o')
hold on
stem(lags(I),xc(I),'filled')
title('Korrelation zwischen x und y')
```

```
33 xlabel('samples');
34 ylabel('Amplitude');
35 hold off
36 legend(["Cross-correlation",sprintf('Maximum at lag %d',lags(
       I))])
37
38 if delay>0 %rechts
39     y2=[zeros(size(y,1),delay), y(:,1:end-delay)];
40     x2=[zeros(size(x,1),delay), x(:,delay+1:end)];
41 elseif delay<0 %links
42     y2=[y(:,abs(delay)+1:end),zeros(size(y,1),abs(delay))];
43     x2=[x(:,1:end-abs(delay)),zeros(size(x,1),abs(delay))];
44 else %0 = no shift
45     y2=y;
46     x2=x;
47 end
48
49 figure
50 plot(x2,'r')
51 hold on
52 plot(y2,'b')
53 title('Synchronisation zwischen x und y')
54 xlabel('samples');
55 ylabel('Amplitude');
56 hold off
```

A.6 Zeitliche Synchronisation

```
%% Zeitliche Synchronisation %%
clear all
close all
clc
%%
%laden der modulierten IQ-Signale
load('modulated0.mat');
load('modulated1.mat');

%%
%Umformung von Zeitreihe in Array und in einen 1x[] Vektor
s0=reshape(modulated0.Data.',1,[]);
s1=reshape(modulated1.Data.',1,[]);

%%
%Definierung der Messparameter
%Trägerfrequenz
Fc=97.1e6;

%Abtastfrequenz
Fs=2.4e6;

%Framelänge in Samples
Frame=length(modulated0.Data(1,:));

%(length(s0)/Frame)*(Frame/Fs) -> Frame kürzt sich weg
%Gesamtsignallänge in s
runtime_full=length(s0)/Fs;

%Zeitachse
t_full=0:1/Fs:runtime_full-1/Fs;

```

```
33 %Länge des Vektors
34 npts_full = length(t_full);
35
36 %%
37 %Schaubild von Referenz- und Echosignal in Abhängigkeit der
       Samples
38 figure(1)
39 subplot(2,1,1);
40 plot(abs(s0),'r');
41 title('Referenzsignal')
42 xlabel('samples');
43 ylabel('Amplitude');
44 legend('SDR0');
45 subplot(2,1,2)
46 plot(abs(s1),'b');
47 title('Echosignal')
48 xlabel('samples');
49 ylabel('Amplitude');
50 legend('SDR1');
51
52 %%
53 %Schaubild von Referenz- und Echosignal in Abhängigkeit der
       Zeit
54 figure(2)
55 subplot(2,1,1);
56 plot(t_full,(abs(s0)),'r');
57 title('Referenzsignal')
58 xlabel('Time (s)');
59 ylabel('Amplitude');
60 legend('SDR0');
61 subplot(2,1,2)
62 plot(t_full,abs(s1),'b');
63 title('Echosignal')
64 xlabel('Time (s)');
```

```
65 ylabel('Amplitude');
66 legend('SDR1');
67
68 %%
69 %Start des Synchronisationssignal
70 syncstart=1;
71 %Ende des Synchronisationssignal - muss anhand der figure(1)
       benannt werden
72 syncstop=10690000;
73
74 %Start des Empfangssignals - muss anhand der figure(1)
       benannt werden
75 signalstart=14160896;
76
77 %asynchronen Synchronisationssignals
78 desync0=s0(1,syncstart:syncstop);
79 desync1=s1(1,syncstart:syncstop);
80 %%
81 %Schaubild von beiden Sychnronisationssignalen getrennt
82 figure('NumberTitle', 'off', 'Name', 'Vollständiges
       getrenntes Sync-Signal');
83 subplot(2,1,1);
84 plot(abs(desync0),'r');
85 title('Sync vom Referenzsignal')
86 xlabel('samples');
87 ylabel('Amplitude');
88 legend('desync0');
89 subplot(2,1,2)
90 plot(abs(desync1),'b');
91 title('Sync vom Echosignal')
92 xlabel('samples');
93 ylabel('Amplitude');
94 legend('desync1');
95 %%
```

```
96 %Schaubild von beiden Sychnronisationssignalen überlappend
97 figure('NumberTitle', 'off', 'Name', 'Vollständiges
       überlappendes Sync-Signal');
98 plot(abs(desync0(1,3000:3100)),'r');
99 hold on
100 plot(abs(desync1(1,3000:3100)),'b');
101 title('Asynchronität zwischen Referenz- und Echosignal')
102 xlabel('samples');
103 ylabel('Amplitude');
104 hold off
105
106 %%
107 %Referenzwert vom finddelay-Befehl
108 d = finddelay(desync0,desync1);
109
110 %%
111 %Sampleversatz zwischen den Synchronisationssignalen
112 %d = finddelay(x,y)
113 %xcorr==c=conv(x,fliplr(y));
114 %Kreuzkorrelation und Versatzwerte der
       Synchronisationssignalen
115 [xc_sync,lags]=xcorr(desync0,desync1);
116 %Maxima der Kreuzkorrelation = Sampleversatz
117 [~,I_sync] = max(xc_sync);
118 %Verzögerrung aus der Korrelation
119 delay_signal1=lags(I_sync);
120
121 %%
122 %
123 figure('NumberTitle', 'off', 'Name', 'crosscorrelation delay
       application');
124 stem(lags,xc_sync,'-o')
125 hold on
126 stem(lags(I_sync),xc_sync(I_sync),'filled')
```

```
127 title('Kreuzkorrelation der Synchronisationssignale')
128 xlabel('samples');
129 ylabel('Amplitude');
130 hold off
131 legend(["Cross-correlation",sprintf('Maximum at lag %d',lags(
      I_sync))])
132
133 %%
134 %If-Schleife für den Buffer zur Korrektur der Sample
135 %[Xa,Ya] = alignsignals(X,Y)
136 %Eilt SDR0 dem SDR1 voraus muss das SDR1 Signal nach rechts
      verschoben werden
137 %[000000...000,Signal]
138 if delay_signal1>0 %rechts
139
140     signal0=s0;
141     signal1=[zeros(size(s1,1),delay_signal1), s1(:,1:end-
        delay_signal1)];
142
143
144 elseif delay_signal1<0 %links
145 %Eilt SDR1 dem SDR0 voraus muss das SDR1 Signal nach links
      verschoben werden
146 %[Signal,00000...00000]
147
148     signal0=s0;
149     signal1=[s1(:,1:end-abs(delay_signal1)),zeros(size(s1,1),
        abs(delay_signal1))];
150
151 else %0 = no shift
152
153     signal0=s0;
154     signal1=s1;
155
```

```
156 end
157
158 %%
159 %synchronisierten Synchronisationssignale
160 sync0=signal0(1,syncstart:syncstop);
161 sync1=signal1(1,syncstart:syncstop);
162
163 %%
164 %Kreuzkorrelation und Versatzwerte
165 [xc_cor,lags]=xcorr(sync0,sync1);
166 %Maxima der Kreuzkorrelation = Sampleversatz
167 [~,I_cor] = max(xc_cor);
168 %Verzögerrung aus der Korrelation
169 delaycor_signal=lags(I_cor);
170
171 %%
172 figure('NumberTitle', 'off', 'Name', 'comparison signal0 and
      corrected signal1');
173 plot(abs(signal0(1,3000:3100)),'r')
174 hold on
175 plot(abs(signal1(1,3000:3100)),'b')
176 title('Synchronität zwischen Referenz- und Echosignal')
177 xlabel('samples');
178 ylabel('Amplitude');
179 hold off
180 %%
181 %Signale auf die Empfangssignallänge zurecht schneiden
182 %Referenzsignal
183 ref0=signal0(1,signalstart:end);
184 %Echosignal
185 surv1=signal1(1,signalstart:end);
186
187 %%
188 %Kreuzkorrelation der synchronisierten Signale
```

```
189 figure('NumberTitle', 'off', 'Name', 'crosscorrelation delay
      compensation');
190 stem(lags,xc_cor,'-o')
191 hold on
192 stem(lags(I_cor),xc_cor(I_cor),'filled')
193 title('Kreuzkorrelation der zeitlich korrigierten Signale')
194 xlabel('samples');
195 ylabel('Amplitude');
196 hold off
197 legend(["Cross-correlation",sprintf('Maximum at lag %d',lags(
      I_cor))])
198
199 %%
200 %Speichern der synchronisierten Signale in v7.3 Struktur
201 %schnellere Ladezeiten
202 save('ref0','ref0','-v7.3');
203 %%
204 save('surv1','surv1','-v7.3');
205
206 %%
207 %Frequenzanalyse
208 %FFT der Synchronisierten Signale
209 FFT0=fftshift(fft(sync0,[],2));
210 FFT1=fftshift(fft(sync1,[],2));
211
212 %%
213 %Achsen Definition
214 n = length(sync0);
215 f = (-n/2:n/2-1)*(Fs/n);
216 %f = (0:n-1)*(Fs/n);
217
218 %%
219 %Signalleistung
220 power0 = abs(FFT0).^2/n;
```

```
power1 = abs(FFT1).^2/n;

%%
%Zusatz: Frequenzsynchronisation
subplot(211)
plot(f,10*log10(power0),'r')
xlabel('Frequency')
ylabel('Power')
hold on
subplot(212)
plot(f,10*log10(power1),'b')
xlabel('Frequency')
ylabel('Power')
hold off

%%
[xc_freq,lags]=xcorr(FFT0,FFT1);
[~,I_freq] = max(xc_freq);
delayfreq_signal=lags(I_freq);
```

A.7 Ambiguitätsfunktion

```
%% Ambiguitätsfunktion mit angewendetem CLEAN-Algorithmus %%
close all
clear all
clc

%%
disp('Konstanten bestimmen')

%Lichtgeschwindigkeit
c=299792458;      %[m/s]

%Abtastfrequenz
fs=2.4e6;         %[1/s]

%Bandbreite
B=fs;             %[1/s]

%Trägerfrequenz
fc=97.1e6;        %[1/s]

%Framelänge
framesize=1024; %samples

%Dauer eines Samples
sampletime=1/fs;              %[s]

%Dauer eines Frames
frametime=framesize/fs;       %[s]
Delta_R=c*sampletime;%[m]

%erwartet Geschwindigkeit des Flugobjektes
ap_min=75;  %[m/s]=270km/h
```

```
33 ap_max=100; %[m/s]=360km/h
34
35 %Definition der Geschwindigkeit
36 v_ap=ap_max;%[m/s]
37
38 %zurückgelegte Distanz in einem sample
39 apdistance_min_sample=ap_min*sampletime; %[m]
40 apdistance_max_sample=ap_max*sampletime; %[m]
41
42 %zurückgelegte Distanz in einem Frame
43 apdistance_min_frame=ap_min*frametime; %[m]
44 apdistance_max_frame=ap_max*frametime; %[m]
45
46 lambda=c/fc;
47
48 sampletime=1/B;
49
50 sampledistance=c*sampletime; %[m]
51
52 %Doppler
53 fd=fc*2*v_ap/c;
54
55 %Dopplerzeit
56 fd_time=1/fd;
57
58 %number of signal / SNR Verbesserung
59 batchlength=fix(fd_time/sampletime);
60 %batchlength=B*T;
61
62 %Periode
63 T=Delta_R/v_ap;
64
65 %Dopplerauflösung
66 fd_res=1/T;
```

```
67
68 %bistatische Entfernung
69 R=sampletime*c;
70
71 %bistatische Entfernungsauflösung
72 delta_R=c/B;
73
74 %bistatische Geschwindigkeitsauflösung
75 delta_V=lambda/T;
76
77 %bistatische Geschwindigkeit
78 V=-lambda*fd;
79
80 %Fensterbreite der FFT
81 NFFT=fix(T/(sampletime*1024));
82
83 %Maximale beobachtete Entfernung
84 max_range=20000; %[m]
85
86 %eff. Signal-Samples bei einer max_range von 20000m
87 n=fix(max_range/sampledistance);
88
89 disp('IQ-Daten laden')
90
91 %%
92 %synchronisierte IQ-Daten der Beobachtungsantenne
93 load('surv1.mat');
94
95 %synchronisierte IQ-Daten der Referenzantenne
96 load('ref0.mat');
97
98 disp('IQ-Daten umformen')
99
100 %%
```

```
101 %formt die synchronisierten Vektoren in Arrays der Größe []
      x1024
102 x=reshape(ref0,1024,[])';
103 %%
104 y=reshape(surv1,1024,[])';
105 %%
106 %Tiefpassfilter um die Signale zu reinigen
107 %lowpass
108 %Frequency units MHz
109 %Input sample rate 2.4
110 %Passband und Stopband mus abhängig von der gewünschten
      Filterwirkung
111 %gesetzt werden
112 filterBuilder
113
114 %%
115 %Filterkoeffizienten setzen
116 filter_coeff=Hlp.Numerator;
117 %%
118 %preallocation der Arrays
119 x_filter=zeros(size(x,1),size(x,2)+size(filter_coeff,2)-1);
120 y_filter=zeros(size(y,1),size(y,2)+size(filter_coeff,2)-1);
121
122 %%
123 %Faltung von Signal und filter
124 for i=1:size(x,1)
125     x_filter(i,:)=conv(x(i,:),filter_coeff);
126     y_filter(i,:)=conv(y(i,:),filter_coeff);
127 end
128 %%
129 %Anwendung des Clean-Algorithmus
130 clean_y=y_filter-x_filter;
131
132 %%
```

```
133 %x und y gefiltert neu definieren
134 x=x_filter;
135 y=y_filter;
136
137 %Dimensionierung der Signale für die Kreuzkorrelation
138 x1=x(:,1:n*2);
139 y1=[y(:,1:n),zeros(size(y,1),n)];
140 x2=[x(:,1:n),zeros(size(x,1),n)];
141 y2=y(:,1:n*2);
142 clean_y1=clean_y(:,1:n*2);
143 clean_y2=[clean_y(:,1:n),zeros(size(clean_y,1),n)];
144
145 %preallocation der Arrays
146 xc=zeros(size(x1,1),size(x1,2)+size(y1,2)-1);
147 ac=zeros(size(x1,1),size(x1,2)+size(y1,2)-1);
148 clean_xc=zeros(size(clean_y1,1),size(x1,2)+size(y1,2)-1);
149
150 disp('Auto-und Kreuzkorrelation')
151 %%
152 %Korrelation
153 %xc = Kreuzkorrelation(Referenz,Beobachtung)
154 %ac = Autokorrelation (Beobachtung,Beobachtung)
155 %clean_xc = Kreuzkorrelation (Referenz, Clean Beobachtung
156 for i=1:size(x1,1)
157     xc(i,:)=xcorr(x1(i,:),y1(i,:));
158     ac(i,:)=xcorr(y2(i,:),y1(i,:));
159     clean_xc(i,:)=xcorr(x1(i,:),clean_y2(i,:));
160 end
161
162 %%
163 %Signale dimensionieren
164 xc1=xc(:,1:n);
165 ac1=ac(:,1:n);
166 clean_xc1=clean_xc(:,1:n);
```

```
167
168 %%
169 %Arrays in Entfernung und Doppler für die Ambiguitätsfunktion
170
171 disp('Erstellung Cell-Arrays')
172 for i=1:fix(size(xc1,1)/NFFT)
173     xc_M{i,1}=xc1(i*NFFT-(NFFT-i+i-1):i*NFFT,1:size(xc1,2));
174     xc_fd{i,1}=fftshift(fft(xc_M{i,1},[],1));
175
176     ac_M{i,1}=ac1(i*NFFT-(NFFT-i+i-1):i*NFFT,1:size(ac1,2));
177     ac_fd{i,1}=fftshift(fft(ac_M{i,1},[],1));
178
179     clean_xc_M{i,1}=clean_xc1(i*NFFT-(NFFT-i+i-1):i*NFFT,1:
        size(clean_xc1,2));
180     clean_xc_fd{i,1}=fftshift(fft(clean_xc_M{i,1},[],1));
181 end
182
183 %%
184 %Dimensionierung der Achsen
185 disp('Erstellung Grafiken')
186 xa=linspace(-300,300,n*2);
187 ya=linspace(n*sampledistance,0,n*2);
188
189 %lambda=c/Fc;
190 %dv=lambda/2*1/NFFT_time;
191
192 %%
193 %Analyse eines Messpunktes
194 Messpunkt=30;
195 %%
196 %Messpunkt Kreuzkorrelation Omni x Direkt
197 s1=xc_fd{Messpunkt,1}.';
198 figure(1)
199 imagesc(xa,ya,(abs(s1)));
```

```
200 xlabel('Dopplerversatz [Hz]');
201 ylabel('Distanz [m]');
202 colorbar
203 title('[R,V] Ambiguitätsfunktion Referenz x Beobachtung')
204
205 %%
206 %Messpunkt Autokorrelation Direkt x Direkt
207 s2=ac_fd{Messpunkt,1}.';
208 figure(2)
209 imagesc(xa,ya,(abs(s2)));
210 xlabel('Dopplerversatz [Hz]');
211 ylabel('Distanz [m]');
212 colorbar
213 title('[R,V] Ambiguitätsfunktion Beobachtung x Beobachtung')
214
215 %%
216 %Messpunkt Kreuzkorrelation Omni x Clean-Direkt
217 s3=clean_xc_fd{Messpunkt,1}.';
218 figure(3)
219 imagesc(xa,ya,(abs(s3)));
220 xlabel('Dopplerversatz [Hz]');
221 ylabel('Distanz [m]');
222 colorbar
223 title('[R,V] Ambiguitätsfunktion Referenz x Clean-Beobachtung
      ')
```

B Linux Treiberinstallation

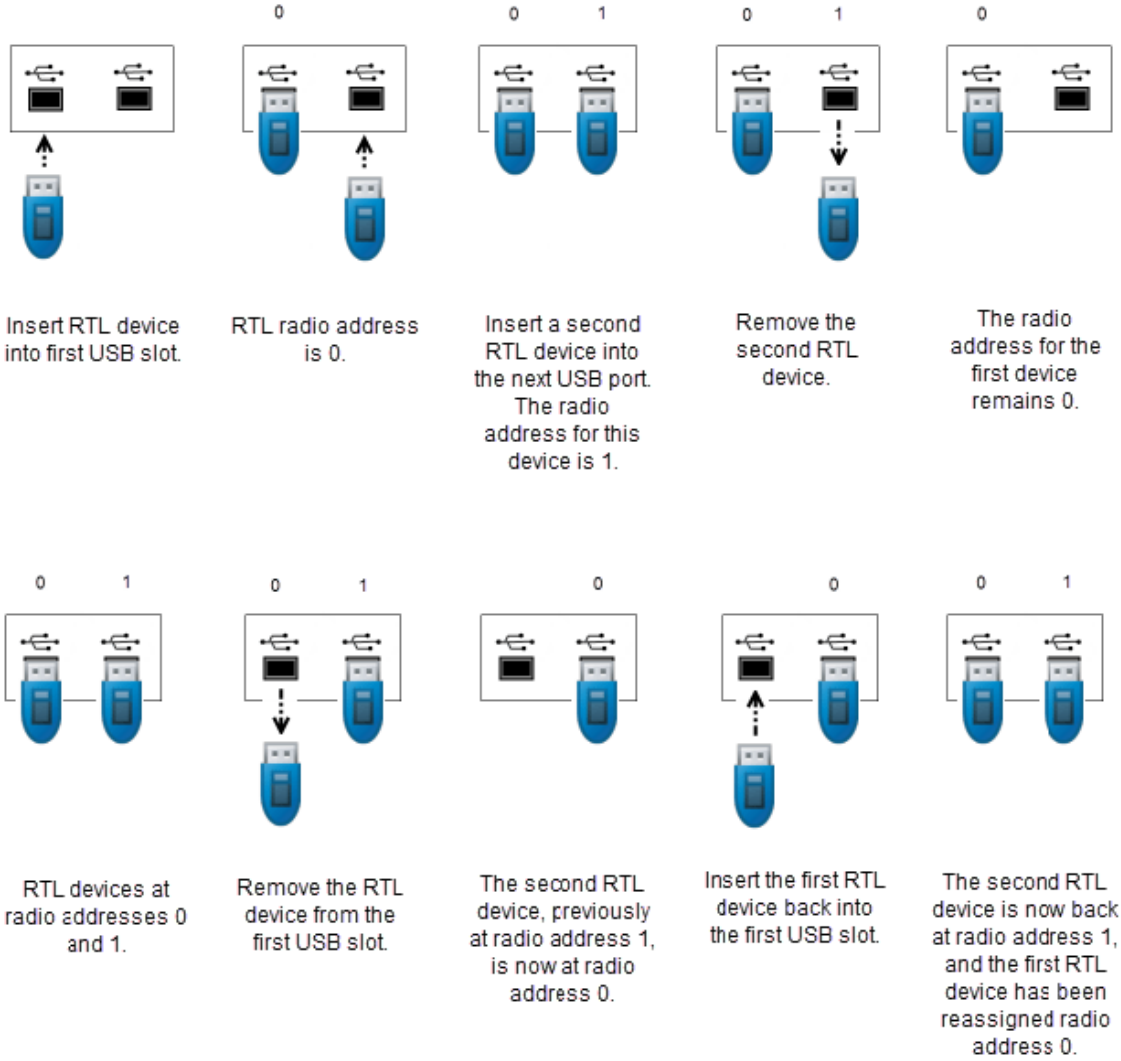
```
//Installation der rtlsdr und linusb1.0 Bibliothek unter
   Linux//

sudo apt-get update
sudo apt-get install rtl-sdr
\newline
sudo apt-get install libusb-1.0-0-dev git cmake
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr/
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo cp ../rtl-sdr.rules /etc/udev/rules.d/
sudo ldconfig

//Osmocom git Status: Online, Stand: 15.10.2020//

//Blacklist vom DVB-T Standard//
echo 'blacklist dvb_usb_rtl28xxu' | sudo tee ? append /etc/
   modprobe.d/blacklist-dvb_usb_rtl28xxu.conf
```

C MATLAB USB-Adressierung für SDR



D Gesamtsystem

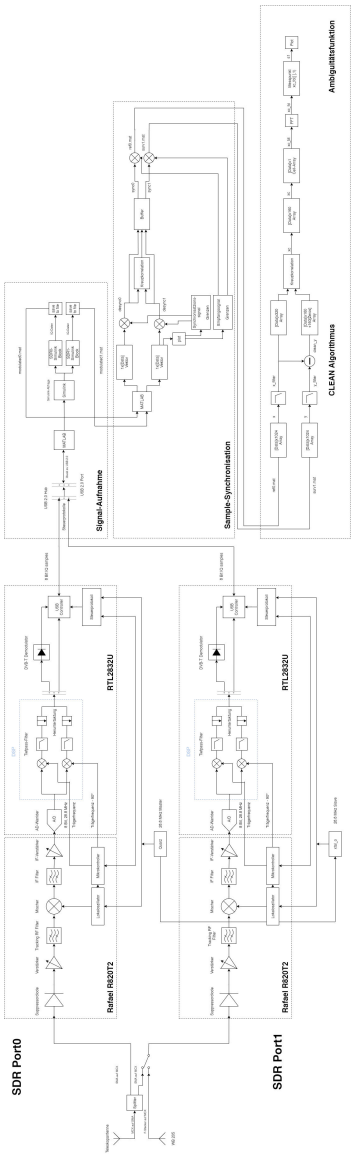


Abbildung D.1: Blockschaltbild des gesamten umgesetzten Systems.

Abkürzungsverzeichnis

SDR	Software-Defined-Radio
BPR	Bistatisches Passiv-Radar
UKW	Ultrakurzwelle
DAB	Digital Audio Broadcasting
SNR	Signal-Rauschabstand
RCS	Radarquerschnitt
CPI	Köhärentes Verarbeitungsintervall
DPI	Direct Path Interference
AF	Ambiguitätsfunktion
HPBW	Half Power Beam Width
USRP	Universal Software Radio Peripheral
FPGA	Field Programmable Gate Array
RF	Radio-Frequenz
DSP	Digitaler Signalprozessor
IQ	In-Phase Quadrature
PPM	Parts Per Million
PLL	Phasenregelschleife
NCO	Numerically Controlled Oscillator

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] I. Balajti, „One radar-one, equation concept for radars of ATC“, *2017 18th International Radar Symposium (IRS)*, Prague: 2017.

[2] J. Göbel, *Radartechnik Grundlagen und Anwendungen*. Berlin: VDE VERLAG GMBH, 2011.

[3] G. Curry, *Radar Essentials: A Concise Handbook for Radar Design and Performance Analysis*. Raleigh, NC: Scitech Pub. Inc., 2012.

[4] M. Malanowski, *Signal Processing for Passive Bistatic Radar*. London: Artech House, 2019.

[5] M. Skolnik, *Radar Handbook*. New York: McGraw-Hill, 2008.

[6] F. Heunis, *Passive Coherent Location Radar using Software-Defined Radio techniques*, University of Cape Town, Department of Electrical Engineering, 2010.

[7] The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard Radar Definitions*, IEEE Std 686-1997, New York, 1997.

[8] The Mathworks, Inc., *„Introduction to Statistical Radar Models for Object Tracking“*, [Online], https://www.mathworks.com/help/fusion/ug/introduction-to-radar-for-object-tracking.html,
letzter Zugriff: 31. 05. 2021.

[9] J.-R. Ohm und H.D. Lüke, *Signalübertragung, Grundlagen der digitalen und analogen Nachrichtenübertragungssysteme*. Berlin: Springer-Verlag, 2010.

[10] R.W. Stewart und K.W. Barlee und D.S.W. Atkinson und L.H. Crockett, *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. Glasgow: Strathclyde Academic Media, 2015.

[11] Nooelec, Inc., *„Nooelec NESDR Mini 2+ 0.5PPM TCXO USB RTL-SDR Receiver (RTL2832 + R820T2) w/ Antenna“*, [Online], https://www.nooelec.com/store/sdr/sdr-receivers/nesdr-mini-2-plus.html,
letzter Zugriff: 03. 06. 2021.

[12] Wittenberg Antennen und Zubehör UG, letzter Zugriff: 04. 05. 2020.

[13] Rohde & Schwarz GmbH & Co. KG, *„VHF/UHF Antennas R&S HL562E Ultralog“*, [Online], https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_common_library/dl_brochures_and_datasheets/pdf_1/HL562E_2020.pdf,
letzter Zugriff: 02. 02. 2020.

[14] Rohde & Schwarz GmbH & Co. KG, *„R&S ESR EMI Test Receiver - Product Brochute“*, [Online], https://www.rohde-schwarz.com/de/broschuere-datenblatt/esr/?filter%5B_facet.CommonLibraryType%5D%5B0%5D=Product+brochure,
letzter Zugriff: 02. 02. 2020.

[15] The Mathworks, Inc., *„polarpattern class“*, [Online], https://www.mathworks.com/help/antenna/ref/polarpattern-class.html,
letzter Zugriff: 03. 06. 2021.

[16] J. Vierinen, *„BUILDING YOUR OWN SDR-BASED PASSIVE RADAR ON A SHOESTRING“*, Supplyframe, Inc., 2015, [Online], https://hackaday.com/2015/06/05/building-your-own-sdr-based-passive-radar-on-a-shoestring/
letzter Zugriff: 03. 06. 2021.

[17] W. Feng und G. Cherniak und J.-M. Friedt und M.Sato, *„Software defined radio implementation of passive RADAR using low-cost DVB-T receivers“*, Center of NorthEast Asia Studies, Tohoku University Sendai und FEMTO-ST Besançon, 2018, [Online], http://jmfriedt.free.fr/URSI.pdf,
letzter Zugriff: 25. 05. 2021.

[18] M.J. Ryan, *„Low Cost Passive Radar Through Software Defined Radio“*, University of Southern Queensland, Faculty of Health, Engineering & Sciences, 2016.

[19] Great Scott Gadgets, *„HackRF wiki“*, [Online], https://github.com/mossmann/hackrf/wiki,
letzter Zugriff: 06. 06. 2021.

[20] Qorvo, Inc, *„RFFC5072 85 - 4200 MHz Wideband Synthesizer / VCO with Integrated 6 GHz RF Mixer“*, 2014, [Online], https://www.qorvo.com/products/p/RFFC5072,
letzter Zugriff: 06. 06. 2021.

[21] Maxim Integrated, *„MAX2837 2.3GHz to 2.7GHz Wireless Broadband RF Transceiver“*, 2015, [Online], https://www.maximintegrated.com/en/products/comms/wireless-rf/MAX2837.html,
letzter Zugriff: 06. 06. 2021.

[22] TAPR, *„HackRF One 1 MHz to 6 GHz SDR“*, 2020, [Online], https://tapr.org/product/hackrf-one-1-mhz-to-6-ghz-sdr/,
letzter Zugriff: 06. 06. 2021.

[23] Nuand, *„bladeRF 2.0 micro“*, [Online], https://https://www.nuand.com/bladerf-2-0-micro/,
letzter Zugriff: 06. 06. 2021.

[24] Altera Corporation, *„Cyclone V Device Datasheet“*, 2013, [Online], https://docs.rs-online.com/6c2b/0900766b8125afe4.pdf,
letzter Zugriff: 06. 06. 2021.

[25] ARM Limited, *„ARM926EJ-S Technical Reference Manual“*, 2003, [Online], https://www.microchip.com/downloads/en/DeviceDoc/ARM_926EJS_TRM.pdf,
letzter Zugriff: 06. 06. 2021.

[26] Analog Devices, Inc., *„RF Agile Transceiver Data Sheet AD9364“*, 2014, [Online], https://www.analog.com/media/en/technical-documentation/data-sheets/AD9364.pdf,
letzter Zugriff: 06. 06. 2021.

[27] Ettus Research, *„USRP B200 (Board Only)“*, National Instruments, Inc., [Online], https://www.ettus.com/all-products/ub200-kit/,
letzter Zugriff: 06. 06. 2021.

[28] Xilinx, Inc., *„Spartan-6 FPGA Data Sheet: DC and Switching Characteristics“*, 2015, [Online], https://www.xilinx.com/support/documentation/data_sheets/ds162.pdf,
letzter Zugriff: 06. 06. 2021.

[29] H. Welte, *„rtl-sdr“*, 2019,
[Online], https://osmocom.org/projects/rtl-sdr/wiki,
letzter Zugriff: 06. 06. 2021.

[30] Rafael Microelectronics, Inc., *„R820T High Performance Low Power Advanced Digital TV Silicon Tuner Datasheet“*, 2011, [Online], https://datasheetspdf.com/pdf-file/792285/RafaelMicroelectronics/R820T/1,
letzter Zugriff: 06. 06. 2021.

[31] Realtek Semiconductor Corp., *„RTL2832U DVB-T COFDM DEMODULATOR + USB 2.0“*, [Online], https://www.realtek.com/en/products/communications-network-ics/item/rtl2832u,
letzter Zugriff: 06. 06. 2021.

[32] S Markgraf, *„rtl-sdr-release“*, H. Welte (Osmocom), 2014,
[Online], http://osmocom.org/attachments/download/2242/RelWithDebInfo.zip,
letzter Zugriff: 06. 06. 2021.

[33] E. Wild, *„omocom/rtl-sdr“*, H. Welte (Osmocom), 2020,
[Online], https://github.com/osmocom/rtl-sdr,
letzter Zugriff: 07. 06. 2021.

[34] CMake, *„CMake“*, Kitware, Inc,
[Online], https://cmake.org/,
letzter Zugriff: 07. 06. 2021.

[35] P. Batard, *„Zadig“*,
[Online], https://zadig.akeo.ie/,
letzter Zugriff: 07. 06. 2021.

[36] Airspy, *„SDR Software Download“*, Airspy.us,
[Online], https://airspy.com/download/,
letzter Zugriff: 07. 06. 2021.

[37] C.J. Cliffe, *„CubicSDR Downloads“*, CubicSDR,
[Online], https://cubicsdr.com/?cat=4,
letzter Zugriff: 07. 06. 2021.

[38] D. Kozel, *„Documentation“*, GNU Radio project,
[Online], https://www.gnuradio.org/docs/,
letzter Zugriff: 07. 06. 2021.

[39] The Mathworks, Inc, *„Products and Services“*,
[Online], https://www.mathworks.com/products.html?s_tid=gn_ps,
letzter Zugriff: 07. 06. 2021.

[40] The Mathworks, Inc, *„Configure Multiple RTL-SDR Radios“*,
[Online], https://www.mathworks.com/help/supportpkg/rtlsdrradio/ug/configure-multiple-radios.html,
letzter Zugriff: 07. 06. 2021.

[41] The Mathworks, Inc, *„Software-Defined Radio Using MATLAB, Simulink, and the RTL-SDR“*,
[Online], https://www.mathworks.com/campaigns/offers/download-rtl-sdr-ebook.html,
letzter Zugriff: 07. 06. 2021.

[42] The Mathworks, Inc, *„comm.SDRRTLReceiver“*,
[Online], https://www.mathworks.com/help/supportpkg/rtlsdrradio/ug/comm.sdrrtlreceiver-system-object.html,
letzter Zugriff: 08. 06. 2021.

[43] The Mathworks, Inc, *„Listener Callback Syntax“*,
[Online], https://www.mathworks.com/help/matlab/matlab_oop/listener-callback-functions.html,
letzter Zugriff: 08. 06. 2021

[44] The Mathworks, Inc, *„RTL-SDR Receiver“*,
[Online], https://www.mathworks.com/help/supportpkg/rtlsdrradio/ug/rtlsdrreceiver.html,
letzter Zugriff: 08. 06. 2021

[45] keenerd, *„keenerd/rtl-sdr“*,
[Online], https://github.com/keenerd/rtl-sdr/commit/3f2632dcaf5ea2c2f28af99a94c64b804a063bfd,
letzter Zugriff: 11. 06. 2021

[46] ADS-B Exchange, *„ADS-B Exchange“*,
[Online], https://globe.adsbexchange.com/,
letzter Zugriff: 13. 06. 2021

[47] The Mathworks, Inc, *„finddelay“*,
[Online], https://de.mathworks.com/help/signal/ref/finddelay.html,
letzter Zugriff: 14. 06. 2021

Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangaben eindeutig kenntlich gemacht. Ebenfalls erkläre ich mich einverstanden, dass meine Thesis im akademischen Rahmen der Hochschule für Angewandte Wissenschaften Hamburg veröffentlicht werden darf.

Ort, Datum Fabian Dittié