HAW HAMBURG

BACHELORTHESIS
Markus Blechschmidt

Sicherheitsfallstudie Drahtloser Eingabegeräte

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Markus Blechschmidt

Sicherheitsfallstudie Drahtloser Eingabegeräte

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Technische Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Klaus-Peter Kossakowski
Zweitgutachter: Prof. Dr. Franz Korf

Eingereicht am: 03. Februar 2021

Markus Blechschmidt

Thema der Arbeit

Sicherheitsfallstudie Drahtloser Eingabegeräte

Stichworte

Sicherheit, drahtlos, Eingabe, Tastatur

Kurzzusammenfassung

Die Bachelorarbeit untersucht die Sicherheitslücken moderner USB-Funktastaturen (nicht Bluetooth) mit besonderem Fokus auf die verbreiteten Angriffsmethoden Sniffing (z.B. zum Abfangen von Passwörtern) und Injection (z.B. zum Ausführen bösartiger Befehle).

Markus Blechschmidt

Title of Thesis

Security Case Study of Wireless Input Devices

Keywords

security, wireless, input, keyboard

Abstract

The bachelor's thesis examines the security vulnerabilities of modern USB-powered, non-Bluetooth wireless keyboards with focus on common attack methods such as sniffing (e.g. for intercepting passwords) and injection (e.g. for executing malicious commands).

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ADC Analog-to-Digital Converter.

BFSK Binary Frequency Shift Keying.

BLE Bluetooth Low Energy.

BNetzA Bundesnetzagentur.

BSI Bundesamt für Sicherheit in der Informationstechnik.

CTR Counter Mode.

DAC Digital-to-Analog Converter.

DSP Digital Signal Processing.

ISM Industrial, Scientific and Medical.

ITU Internationale Fernmeldeunion.

LNA Low Noise Amplifier.

LO Lokaler Oszillator.

MMCR Mueller and Müller Clock Recovery.

NRZ Non Return to Zero Kodierung.

PA Power Amplifier.

SDR Software Defined Radio.

WPCR Whole Packet Clock Recovery.

Symbolverzeichnis

τ Der Umfang eines Kreises geteilt durch die Länge seines Radius [14].

1 Einleitung

Die meisten IT-Sicherheitsforschungen und -maßnahmen sind softwarebasiert. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) hat einen Großteil der heutigen Erkenntnisse in einer "Cyber-Fibel" zusammengefasst und herausgegeben [2]. Darin wird Wissen über die Einrichtung einer sicheren Internetverbindung, vertrauenswürdige Software, Gefahren durch Schadsoftware, die Wahl sicherer Anmeldedaten und die Privatsphäre bei Kommunikation vermittelt.

Für die meisten dieser Probleme wurden bereits technische Lösungen gefunden. Programme und Dateien können signiert und verifiziert werden, Passwort-Manager können Anmeldedaten verwalten, und Ende-zu-Ende-Verschlüsselung ermöglicht private Kommunikation. Darüber hinaus macht Zwei-Faktor-Authentisierung den Identitätsnachweis sicherer und Hashing sowie Salting verhindern die nicht autorisierte Veröffentlichung von Passwörtern.

Infolgedessen hat sich der Fokus von Angreifer_innen verschoben. "Die am häufigsten ausgenutzte Schwachstelle ist der Mensch." [25] Anstatt zu versuchen, Fehler in einem Computersystem auszunutzen, versuchen Cyber-Kriminelle, Benutzer dazu zu bringen, Malware unbeabsichtigt zu installieren. Ein Beispiel dafür ist Ransomware. Diese wird meist über Dateimakros verbreitet. Der Angriff beruht darauf, dass die Person, die die infizierte Datei erhält, die Warnung unbewusst automatisch quittiert. Mittels Spear-Phishing kann die Datei legitimer erscheinen [25].

Eine weitere Schwäche im menschlichen Verhalten ist das inhärente Vertrauen in Hardware. Die Vertrauenswürdigkeit von Dateien kann mithilfe von Signaturen festgestellt werden. Da Hardware keine Prüfsummenverfahren durchlaufen kann, besteht die einzige Möglichkeit für vertrauenswürdige Hardware darin, sie selbst zu erstellen. Infolgedessen ist das Vertrauen in Zweit- oder Drittanbieterhardware für die meisten Menschen alternativlos.

Beispielsweise kann eine Sicherheitslücke im Update-Mechanismus von USB-Stick-Controllern ausgenutzt werden, um bösartige Firmware auf dem Controller zu installieren [28]. Dies ermöglicht die Manipulation des Speicherzugriffs und die Emulation von Ein- oder Ausgabegeräten, um z. B. Virenscans zu umgehen, Dateien zu verstecken, Tastenanschläge zu protokollieren oder zu injizieren. Das BadUSB-Paper [6] beschreibt USB-Flash-Laufwerke als ein "sehr beliebtes Übertragungsmedium", das "die Leute gerne für völlig harmlos halten."

1.1 Struktur

Die vorliegende Arbeit ist in fünf Kapitel und Anhang unterteilt. Das 1. Kapitel stellt einen allgemeinen Bezug her, leitet das Thema ein, erklärt das Ziel und grenzt es ab. Im 2. Kapitel werden technische Grundlagen vermittelt, die zum Verständnis dieser Arbeit beitragen sollen. Hierbei werden jene Aspekte beleuchtet, die bei der Identifikation von Herausforderungen und Implementierungen von Lösungen hilfreich sind. Kapitel 3 beschreibt diese Herausforderungen und dafür entwickelte Implementierungen. Es wird dargelegt, welche Probleme sich für das Erreichen des Zieles aufwerfen und wie diese adressiert werden. Kapitel 4 dokumentiert die Versuche und präsentiert die Ergebnisse. Der Versuchsaufbau, die verwendeten Werkzeuge und Tastaturen werden vorgestellt. Daraufhin folgt eine Beschreibung der Versuchsabläufe. Nach jedem Versuch werden die erlangten Daten ausgewertet. Das 5. Kapitel präsentiert die gewonnenen Erkenntnisse. Die gefundenen Schwachstellen werden diskutiert und die untersuchten Geräte miteinander verglichen. Die Arbeit endet mit Ansätzen für weitere Forschungen. In Anhang A werden die gesammelten Daten der untersuchten Tastaturen dokumentiert. Anhang B enthält Installationsanleitungen.

1.2 Ziel

Ziel dieser Bachelorarbeit ist es, anhand einer gezielten technischen Untersuchung einiger Funktastaturen als Beispiel für drahtlose Eingabegeräte aufzuzeigen, welche Sicherheitslücken im Bereich der Verbraucherelektronik auftreten. Die Arbeit befasst sich mit Funktastaturen, da hier die drahtlose Eingabe auf dem Vormarsch ist. Über diese werden sensible Daten wie beispielsweise Passwörter eingegeben, welche entsprechend geschützt werden müssen. Wenn Eingaben nicht hinreichend gut authentifiziert werden, könnten

Angreifer_innen Remote Code Execution auf dem angeschlossenen Rechner erlangen. Anstelle vollständiger Exploits wird in dieser Arbeit nur untersucht, ob Angriffe grundsätzlich Schwachstellen oder fehlende Sicherheitsmaßnahmen ausnutzen können.

1.3 Zielgruppe

Diese Arbeit soll Grundlagenwissen vermitteln, das für die weitere Sicherheitsforschung mit Software Defined Radios nützlich ist. Leser_innen dieser Arbeit sollten Grundkenntnisse in paketorientierter Kommunikation, zu Signaltheorie und Digital Signal Processing (DSP) und ein Interesse an IT-Sicherheit haben.

1.4 Abgrenzung

Diese Arbeit konzentriert sich auf proprietäre Funkprotokolle und schließt damit Protokolle wie Bluetooth (IEEE 802.15.1) oder Zigbee (IEEE 802.15.4) von der Betrachtung aus. Des Weiteren ist es nicht das Ziel dieser Arbeit, einen vollständigen Exploit pro zu testendem Gerät bereitzustellen, sondern einen Konzeptnachweis zu liefern, dass eine Entwicklung solcher Exploits möglich ist.

2 Technische Grundlagen

Zum Erreichen des Ziels dieser Arbeit ist es notwendig, die Signale, die zwischen Tastatur und Dongle gesendet werden, zu empfangen, zu demodulieren, zu dekodieren und zu interpretieren. In diesem Kapitel werden die in dieser Arbeit behandelten technischen Themen sowie das zum Verständnis notwendige Hintergrundwissen erläutert. Für grundlegendes Verständnis von digitaler Signalverarbeitung sei hier auf die SDR-Tutorials von Michael Ossmann [31] und Dr. Marc Lichtman [17] hingewiesen.

2.1 Drahtlose Kommunikation

In dieser Arbeit werden Geräte betrachtet, die Radiowellen nutzen, um zu kommunizieren. Radiowellen werden von der Internationale Fernmeldeunion als "elektromagnetische Wellen definiert, deren Frequenzen vereinbarungsgemäß unterhalb 3000 GHz liegen, und die sich ohne künstliche Führung im freien Raum ausbreiten." [3]

Der freie Raum wird hierbei als geteiltes Übertragungsmedium genutzt. Damit dieses Medium kollisionsfrei von mehreren Beteiligten genutzt werden kann, ist eine Medienzugriffskontrolle erforderlich.

2.1.1 Frequenzbänder

Die einfachste Methode der Zugriffssteuerung ist der Frequenzvielfachzugriff.

Das elektromagnetische Spektrum wird aufgeteilt in verschiedene Frequenzbänder. Diese werden durch eine untere und eine obere Frequenzgrenze definiert. Das kann analog zur Aufteilung von Internetprotokoll-Adressen in Netzbereiche verstanden werden. Diese Aufteilung erfolgt meist durch Behörden von Nationalstaaten. In Deutschland ist dafür laut § 53 und § 54 des Telekommunikationsgesetzes die Bundesnetzagentur (BNetzA) zuständig [9].

Jedem Frequenzband wird dabei auch eine Frequenznutzung zugeordnet. Die Zuweisung ist in dem von der BNetzA veröffentlichten Frequenzplan einsehbar [1]. Ein Beispiel hierfür ist der Frequenzteilplan 203, welcher den Frequenzbereich von 87,5 MHz bis 108 MHz für die Frequenznutzung zur Ton-Rundfunk-Übertragung – also UKW Radio – vorsieht.

ISM-Bänder

Um gewisse Frequenzbänder nutzen zu dürfen, bedarf es meist des Erwerbs einer Lizenz. So durften z.B. Mobilfunkanbieter 2019 an der Versteigerung von deutschen 5G Frequenzen teilnehmen [10]. Ein anderes Beispiel sind Amateurfunker, welche eine Prüfung ablegen müssen, bevor sie sendend am Funkverkehr teilnehmen dürfen.

Von besonderem Interesse im Bereich der Heimanwendungen sind daher die ISM-Bänder, da diese ohne Erwerb einer Lizenz genutzt werden dürfen [1, p. 714].

Beispiele für ISM-Bänder sind das 27 MHz Band (26,957 MHz – 27,283 MHz), welches im privaten Bereich Verwendung bei ferngesteuerten Spielzeugen findet, das 433 MHz Band (433,05 MHz – 434,79 MHz), welches für Handsprechfunkgeräte und funksteuerbare Steckdosen verwendet wird, und das 2,4 GHz Band (2,4 GHz – 2,5 GHz), welches bekannt ist für WLAN und Bluetooth.

Jedoch ist auch beim Senden auf diesen Bändern die Einhaltung gewisser Regularien notwendig, beispielsweise bezüglich der maximal zulässigen äquivalenten Strahlungsleistung, Bandbreite und relativen Frequenzbelegungsdauer (also das Verhältnis zwischen Sendedauer und Funkstille eines Senders). So darf z.B. nicht durchgehend gesendet werden. Daher nutzen die meisten Systeme, die über ein ISM-Band kommunizieren, paketorientierte Kommunikation.

2.1.2 Kanäle

Frequenzbänder werden weiter in Kanäle unterteilt. Dies kann ähnlich zur Aufteilung von Netzbereichen in Adressen gesehen werden.

Für einige Frequenzbänder sind die Kanalbandbreite und das Kanalraster von der BNetzA vorgegeben. UKW Radio hat eine Kanalbandbreite von 300 kHz und ein Kanalraster von 100 kHz. Das bedeutet, dass die Trägerfrequenz eines Senders immer ein ganzzahliges

Vielfaches von 0,1 MHz ist (z.B. 100,6 MHz) und deren untere und obere Grenzfrequenz bei Trägerfrequenz +/- 0,15 MHz liegt.

An diesem Beispiel sieht man, dass ein Kanal prinzipiell ein Frequenzband im Frequenzband ist. Darüber hinaus können Kanäle sich überlappen, was zu Kollisionen führen kann. Da UKW Sender lokal - also ortsgebunden und von begrenzter Reichweite - sind, werden die Lizenzen für Kanäle lokal so vergeben, dass es nicht zu Überlappungen kommt.

Bei ISM-Bändern gibt es keine Vorgabe von Kanalbandbreite oder Kanalraster, weswegen gilt, "Funkdienste, die innerhalb dieser Frequenzbereiche wahrgenommen werden, müssen Störungen, die durch diese Anwendungen verursacht werden, hinnehmen" und "ISM-Anwendungen in diesen Frequenzbereichen dürfen bei Funkdiensten, die in diesen Frequenzbereichen betrieben werden, keine Störungen verursachen." [1] Es gilt das Prinzip der Störungsvermeidung, als auch das der Störungstoleranz.

Demzufolge müssen Geräte konkurrierende Zugriffssteuerungen auf ISM-Bändern implementieren. Hierfür wird häufig das Frequenzsprungverfahren verwendet.

Frequenzsprungverfahren

Eine einfache Methode, um Zugriffskollisionen mit anderen sendenden Geräten zu vermeiden, ist, in der Auswahl des Kanals flexibel zu sein. Bekannt hierfür ist vor allem BLE. Hierbei gibt es eine Liste von 40 Kanälen. Drei dieser Kanäle werden genutzt, um Verbindungen auszuhandeln. Die Kommunikation erfolgt dann über die 37 weiteren Kanäle. Dabei wird über die Kanäle nach einem pseudozufälligen Muster iteriert. Wenn erkannt wird, dass auf einem Kanal zu viel Interferenz vorherrscht, wird dieser aus der Liste entfernt.

Hierbei ist der Kanalwechsel ein regelmäßig vorkommendes Ereignis im Sendeverhalten. Dies hat einige Vorteile. Die Kommunikation ist widerstandsfähig gegen plötzlich auftauchende Störquellen (z.B. eine schlecht geschirmte Mikrowelle), da der betroffene Kanal durch häufiges Kanalwechseln schnell verlassen wird. Aufgrund der vom Sender und Empfänger geteilten Kanalliste und Sprungmuster ist der nächste Kanal eindeutig und der betroffene Kanal wird gemieden. Weiterhin hat es sicherheitstechnische Vorteile, da eine laufende Verbindung ohne Wissen über Kanalliste und Sprungmuster schwerer zu verfolgen ist.

Ein Angreifer könnte jedoch versuchen, Störfunk zu verursachen, der nur einen Kanal nicht betrifft. Dies würde die Verbindung auf diesen Kanal zwingen. Es wurden bereits Werkzeuge entwickelt, die Interferenz nutzen, um bereits bestehende BLE Verbindungen zu sniffen [36].

Kanalwechsel können prinzipiell auch ad-hoc erfolgen.

2.1.3 Modulation

Daten werden aus Symbolen zusammengesetzt. Diese Symbole müssen in physischer Form existieren. So repräsentieren z.B. magnetische Felder auf Festplatten oder elektrische Pegel auf Leitungen Symbole von Daten. Modulation beschreibt den Prozess des Kodierens dieser Symbole als physikalischen Wert.

Vor dem Modulieren können die Symbole der Daten auch noch mit einem Leitungscode (z.B. Manchester-Code) kodiert werden. Im Laufe dieser Arbeit wird nur auf Non Return to Zero Kodierung (NRZ) eingegangen. Hierbei wird das Symbol 0 mit einem Wert und 1 mit einem anderen Wert kodiert. Es gibt keinen Wert für einen Ruhezustand.

Im elektromagnetischen Spektrum werden Symbole in den Eigenschaften von Radiowellen kodiert [15]. Wenn digitale Daten auf Radiowellen moduliert werden, wird dies Keying genannt, da die Menge der Symbole, aus denen digitale Daten bestehen, begrenzt ist.

Amplitudenmodulation

Amplitudenmodulation von NRZ wird meist durch On-Off-Keying (OOK) realisiert. Hierbei wird 1 als Senden und 0 als Pause kodiert.

Signale des Senders werden mit zunehmendem Abstand vom Empfänger schwächer. Daher muss variable Verstärkung auf der Empfängerseite eingesetzt werden. Dies erschwert die Unterscheidung zwischen Signal und Rauschen [13].

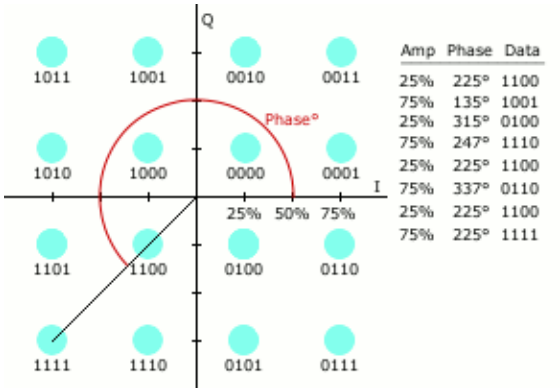

Abbildung 2.1: 16-QAM mit Liste von beispielhaften Konstellationskoordinaten und darin kodierten Daten von [26]

Phasenmodulation

Phasenmodulation wird in digitalem Datenverkehr vorwiegend in Kombination mit Amplitudenmodulation verwendet, um hohe Datenraten zu erreichen. Dies wird als QAM bezeichnet.

Phasenmodulation erfordert einen präzisen Frequenzgenerator und Synchronisation zu Beginn des Sendens. Es findet daher vor allem Anwendung in Technologien wie WLAN.

Frequenzmodulation

Frequenzmodulation von NRZ wird als Binary Frequency Shift Keying (BFSK) bezeichnet. Bei BFSK werden 0 und 1 als zwei verschiedene Frequenzen kodiert, auf denen gesendet wird. Dabei liegt die Symbolfrequenz für 0 meist unter und für 1 über der Trägerfrequenz des Kanals.

Frequenzmodulation ist einfacher zu implementieren als Phasenmodulation und robuster als Amplitudenmodulation. Es wird daher beispielsweise von BLE eingesetzt. Wie in Abschnitt 4.4.1 gezeigt wird, verwenden alle untersuchten Geräte BFSK.

2.2 Software Defined Radio

Klassische Radiosysteme verwenden dedizierte Hardware, um Funkwellen zu senden oder zu empfangen, und um Signale zu modulieren oder zu demodulieren. Das macht sie zwar sehr effizient und kostengünstig, schränkt aber auch ihre Nutzbarkeit ein. Beispiele für solche Systeme sind FM-Stereoanlagen, Pager und drahtlose Netzwerkkarten.

Bisherige Arbeiten im Bereich der Sicherheit von Eingabegeräten nutzen Werkzeuge, die nur gewisse Modulationsverfahren und Protokolle beherrschen [37, 27].

SDRs sind für die Sicherheitsforschung von besonderem Interesse, da sich nahezu jedes Funkprotokoll mit ihnen nachvollziehen und implementieren lässt [11, 5]. Dies macht die Verwendung von funkanwendungsspezifischen Entwicklungstools obsolet. Einheitliche Hardware APIs wie `gr-osmosdr` machen Applikationen portabel. Wie alle andere Hardware sind auch SDRs in den vergangenen Jahren günstiger und leistungsfähiger geworden.

2.2.1 Funktionsweise

Während SDRs zum Senden und Empfangen die gleichen Grundbausteine wie klassische Radios benötigen (Verstärker, Mischer, Filter), führen sie alle anderen Operationen in der digitalen Domäne (d. h. in Software) aus.

Grundlegend kann man sich SDRs wie Soundkarten mit Audioeingang und/oder -ausgang vorstellen, aber mit einer ausreichend schnellen Abtastrate, um im Radiofrequenzspektrum zu arbeiten. Weiterhin wird ein Tuner genutzt, um in höheren Frequenzbereichen arbeiten zu können.

Die in dieser Arbeit verwendeten SDRs benutzen eine Direct Conversion Architektur. Eine Übersicht über diese Architektur bietet das Blockdiagramm in Abbildung 2.2.

Am Anfang des Signalpfades beim Empfang steht die Antenne, welche Radiofrequenzwellen empfängt.

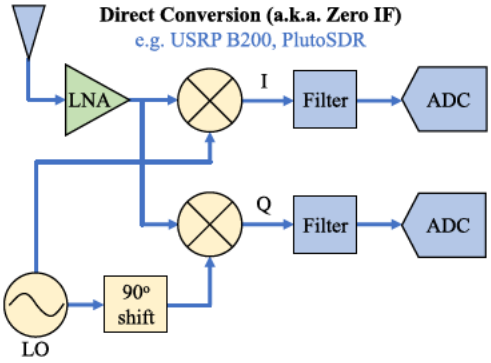

Abbildung 2.2: Blockdiagramm der Empfängerarchitektur eines SDR von [17]

Frequenzeinstellung

Nach Verstärkung durch einen LNA wird das Signal mit zwei Mixern runtergemischt. Hierbei wird ein Lokaler Oszillator (LO) zur Frequenzeinstellung genutzt. Der LO erzeugt zwei Sinuswellen mit der gewählten Frequenz. Eine der Wellen hat eine $\tau/4$ Phasenverschiebung zu der anderen. Das ist notwendig für komplexe Messwerte [16]. Beide Wellen werden mit dem Radiofrequenzsignal in jeweils einen Mixer gegeben. Die Signale, die von den Mixern ausgehen, werden dann mit Tiefpassfiltern von Artefakten befreit (Antialiasing).

Abtastung

Die Ausgabe des Mixers mit dem direkten LO Signal wird als In-Phase (I) bezeichnet, die des Mixers mit dem $\tau/4$ phasenverschobenen Signal als Quadrature (Q). Beide Signale werden von ADCs in der Zeitdomäne abgetastet und in der Spannungsdomäne quantisiert. Das Resultat sind zeit- und wertdiskrete komplexe Messwerte. Diese werden dann über einen digitalen Bus (z.B. USB) an einen Computer weitergeleitet.

Komplexe Zahlen werden in der digitalen Signalverarbeitung bevorzugt, da viele Modulations- und Demodulationsoperationen einfacher herzuleiten und zu implementieren sind. So ist z.B. das Demodulieren von BFSK möglich, indem der Winkel zwischen zwei aufeinander folgenden Messwerten ermittelt wird.

Senden

Der sendende Signalpfad von SDRs ist eine Spiegelung des empfangenden Signalpfades. Hierbei werden anstelle von ADCs DACs verwendet, um von der digitalen in die analoge Domäne umzuwandeln. Die Filter werden genutzt, um Oberwellen zu entfernen (Antialiasing). Anstelle eines LNA tritt ein Power Amplifier (PA), der das Signal vor dem Senden verstärkt.

Kennwerte

Die wichtigsten Eigenschaften von SDRs sind ihre maximale Abtastrate und der Frequenzbereich, in den sie sich einstimmen können.

Die Abtastrate bestimmt die maximale Bandbreite, die zu einem Zeitpunkt beobachtet werden kann. Da komplexe Messwerte verwendet werden, ist die Größe des Spektrums, das zu einem Zeitpunkt betrachtet werden kann, gleich zur Abtastrate. Beispielsweise bedeutet das, wenn der LO auf 2,45 GHz gestellt ist, und mit 20MSamp/s abgetastet wird, liegt die niedrigste empfangene Frequenz bei 2,44GHz und die höchste bei 2,46GHz.

Der Frequenzbereich, der beobachtet werden kann, wird durch die minimale und maximale einstellbare Frequenz des LO vorgegeben. So ist es z.B. ohne weiteres nicht möglich, mit einem RTL-SDR im 2,4 GHz ISM-Band zu empfangen, da die maximale LO Frequenz bei 1,75GHz liegt.

3 Umsetzung

Drahtlose Eingabegeräte werden meist ohne Veröffentlichung eines detaillierten Datenblattes auf den Markt gebracht. Dadurch sind die Parameter der verwendeten Funkprotokolle nicht bekannt.

Bei dieser Arbeit müssen Signale unbekannter Funkprotokolle empfangen, demoduliert und dekodiert werden. In diesem Kapitel werden die Schwierigkeiten, die mit diesen Schritten einhergehen, aufgezeigt. Danach werden die Implementierungen vorgestellt, die zur Lösung dieser Probleme entwickelt wurden.

3.1 Herausforderungen

Wie eingangs erwähnt, lassen sich die Problemfelder Empfang, Modulation und Kodierung erkennen. Das Problemfeld der Kodierung kann weiter in das der Alignment- und Symbolratenerkennung unterteilt werden.

3.1.1 Empfang

Wie in Abschnitt 2.1.2 erläutert, können Systeme, die auf ISM-Bändern funken, ein Sendeverhalten haben, das ohne weiteres Wissen über das System keine Vorhersage darüber zulässt, welcher Kanal für die nächste Übertragung genutzt wird.

3.1.2 Modulation

Wenn die verwendete Modulation unklar ist, erweisen sich SDRs als vorteilhaft, da durch DSP jedes erdenkliche (De-)Modulationsverfahren implementiert werden kann. In folgenden Kapiteln dieser Arbeit wird festgestellt, dass die hier betrachteten Tastaturen nur BFSK verwenden. Daher wird auf dieses Problem in Folge nicht weiter eingegangen.

3.1.3 Symbolrate

Demodulierte Signale ergeben einen Strom von Symbolen, die als Daten interpretiert werden müssen. Dabei ist wichtig, den Symbolstrom korrekt zu zerteilen. In der Signalverarbeitung wird der Begriff Symboltakt-Synchronisation verwendet. Erfolgt dies inkorrekt, werden manche Symbole gar nicht, andere wiederum mehrfach erfasst. Dies ist vergleichbar mit dem Kommunizieren über eine RS-232 Schnittstelle bei der falschen Symbolrate.

3.1.4 Alignment

Das Umwandeln von Signalen zu Symbolströmen ist alleine nicht ausreichend, um die im Signal kodierten Daten zu erhalten. Die Symbole müssen wieder in Datenstrukturen gepackt werden z.B. Bits zu Bytes. Hierzu müssen diese Strukturen innerhalb des Symbolstroms erkannt werden. Passiert dies fehlerhaft, ist sich das analog zum Verpassen des Startbits bei einer RS-232-Kommunikation vorzustellen.

3.2 Implementierungen

In diesem Abschnitt werden verschiedene Lösungsansätze für die erkannten Herausforderungen präsentiert und bewertet.

3.2.1 Empfangen auf allen Kanälen

Die Versuche dieser Arbeit fanden nicht in einem nach außen abgeschirmten Labor statt. Daher wurde von dem Einsatz von Störfunk, um die Geräte auf einen Kanal zu zwingen, abgesehen.

An sich ist es möglich, das Frequenzsprungverhalten eines Senders nachzuvollziehen und in Software in Verbindung mit dem SDR zu implementieren.

Einerseits ist dies sehr aufwendig, da das Frequenzsprungverhalten wie z.B. im Fall von BLE während der Verbindung geändert werden kann. Um an diese Information zu gelangen ist somit auch das Reverse Engineering der Symbolrate, des Alignments und der darauf aufbauenden Protokollbestandteile notwendig. Dies kommt einem Henne-Ei-Problem

gleich. Andererseits benötigen SDRs zum Ändern ihrer LO Frequenz prinzipiell mehr Zeit als ein anwendungsspezifisches Radio. Kanalwechsel könnten so schnell erfolgen, dass sie sich mit einem SDR nicht nachvollziehen lassen.

Eine bessere Methode ist das Empfangen mehrerer Kanäle auf einmal. Mit der stetigen Weiterentwicklung von SDRs stieg auch die Abtastrate, was hierbei nützlich ist. So hat z.B. eines der verwendeten SDRs eine maximal mögliche Abtastrate von 160MSamp/s. Damit kann das gesamte 2,4 GHz ISM-Band auf einmal betrachten werden.

Um das zu tun, muss die Frequenz des LO auf die Mitte des Bereiches, der betrachtet werden soll, gestellt werden. Mithilfe von generischen SDR Frontends und Analysewerkzeugen kann die Mittenfrequenz und Bandbreite der vom System genutzen Kanäle dann bestimmt werden. In anderen DSP Programmen wie GNU Radio Companion können die Signale dann isoliert werden.

Mischen und Filtern

Nehmen wir als Beispiel, dass der LO des SDRs auf eine Frequenz von 2,45 GHz gestellt wurde und wir einen Kanal auf 2,424 GHz mit einer Bandbreite von 500 kHz betrachten wollen.

Durch den LO und die Mixer im SDR wurde die Mittenfrequenz des Kanals von 2,424 GHz auf -26 MHz gemischt. Hierbei sei angemerkt, dass negative Frequenzen möglich sind, da wir mit komplexen Signalen arbeiten. Zunächst muss das Signal auf 0 Hz Mittenfrequenz gebracht werden. Dies erreichen wir durch die Multiplikation mit einem 26 MHz Signal.

Das 500 kHz breite Signal des Kanals kann danach mit einem 250 kHz Tiefpassfilter isoliert werden [32], bevor es weiterverarbeitet wird.

Das Multiplizieren und Filtern kann für beliebig viele Kanäle, die von Interesse sind, parallelisiert werden. Das ist sehr vorteilhaft. Die Grenze ist hierbei vor allem die Rechenkapazität des Computers. Ein anderer Vorzug dieses Verfahrens ist, dass die Mittenfrequenzen der Kanäle willkürlich sein können.

Nachteilig hieran ist, dass ein Flowgraph (vgl. Abbildung 3.1), so wie er in GNU Radio Companion nutzt, recht schnell unübersichtlich und schwer anzupassen wird.

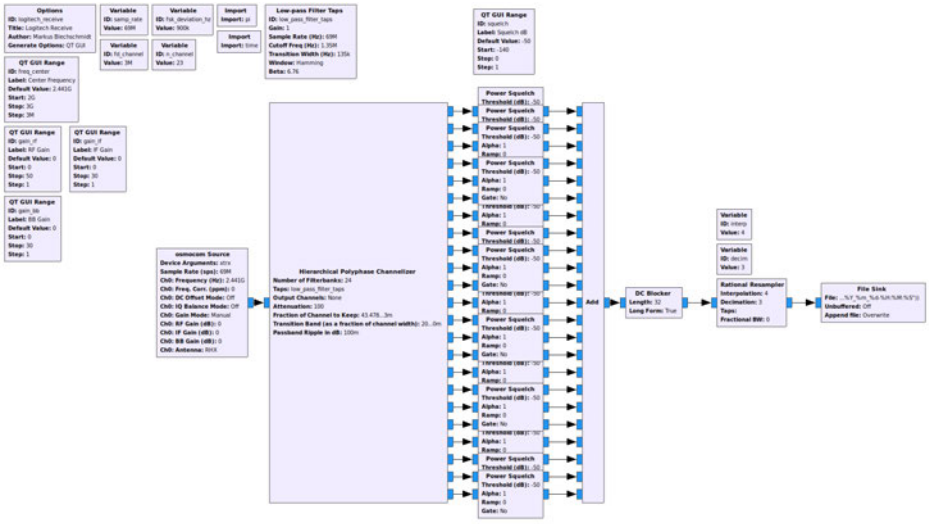

Abbildung 3.1: Flowgraph zum gleichzeitigen Empfangen von 23 Kanälen mit `Hierarchical Polyphase Channelizer`

Channelizer

Wie in Folge dieser Arbeit gezeigt wird, nutzen drahtlose Eingabegeräte oft ein Kanalraster. Das ermöglicht die Nutzung von Channelizern.

Hierzu empfiehlt GNU Radio die Verwendung ihres `Hierarchical Polyphase Channelizer` [19]. Auf die innere Funktionsweise von Channelizern wird an dieser Stelle nicht eingegangen, da es den Rahmen dieser Arbeit übersteigt.

Channelizer zerteilen den betrachteten Frequenzbereich in Kanäle. Die Abtastrate der ausgehenden Signale entspricht dabei der Abtastrate des einkommenden Signals geteilt durch die Anzahl der Kanäle. Die ausgehenden Signale haben dabei meist eine Abtastrate, die höher ist als die Bandbreite der Übertragung, die von Interesse ist, und sollten daher auch durch einen Tiefpassfilter gehen. Dieser Tiefpassfilter kann dem Channelizer als Parameter bei der Erstellung übergeben werden [35].

Es sei beispielsweise angenommen, dass drei Kanäle auf 2,416 GHz, 2,424 GHz und 2,432 GHz mit einer Bandbreite von 300 kHz von Interesse sind. Es lässt sich ein Kanalraster von 8 MHz feststellen. Der mittlere Kanal liegt bei 2,424 GHz.

Um nun alle drei Kanäle zu empfangen, wird der LO des SDR auf 2,424 GHz gestellt und die ADC auf eine Abtastrate von 24 MHz. Zur Auftrennung der Kanäle kann ein Channelizer mit 3 Kanälen und einem 150 kHz Tiefpassfilter instanziiert werden.

Die Signale können dann durch Addition zu einem Signal vereint und weiterverarbeitet werden. Dies geht jedoch meist mit einer Verschlechterung des Signal-Rausch-Verhältnises einher, da das Rauschen mehrerer Kanäle kombiniert wird. Mit Squelch Blöcken auf allen Kanälen kann dies etwas reduziert werden. In Abbildung 3.1 ist ein Flowgraph zu sehen, der diese Idee implementiert.

Das Design hat prinzipiell funktioniert. Jedoch wurde das Signal-Rausch-Verhältnis dabei zu sehr beeinträchtigt.

3.2.2 Automatische Frameerkennung

Da auf ISM-Bändern nicht durchgehend gesendet werden darf, erfolgt die Kommunikation in Paketen. Um ein Paket erfolgreich zu dekodieren, müssen Anfang und Ende der Übertragung des Pakets erkannt werden. Das Signal zwischen Anfang und Ende des Pakets wird als Frame bezeichnet.

Der Dynamic Burst Detector ist ein hierarchischer Block, der zur dynamischen Frameerkennung entwickelt wurde. Die Implementierung ist in Abbildung 3.2 zu sehen. *Dynamisch* bedeutet, dass die Schwellwerte an das Hintergrundrauschen angepasst werden.

Normale Squelch Blöcke von GNU Radio unterscheiden zwischen Signal und Rauschen anhand eines konstanten Wertes. Beim Experimentieren mit verschiedene Antennen und SDR Parametern kann dies schnell zu False Positives und Negatives führen.

Im Dynamic Burst Detector wird mit dem `Complex to Mag^2` Block die Energie im Signal berechnet und danach durch Multiplikation mit einer Konstanten skaliert. Das so erhaltene Signal wird geteilt und geht in zwei `Single Pole IIR Filter` Blöcke. Diese agieren wie RC-Tiefpassfilter in einer analogen Schaltung [22]. Der obere Filter glättet das Signal und entfernt so hochfrequentes Rauschen. Der untere Filter agiert wie ein langes Moving Average, mit welchem die Energie des Grundrauschens ermittelt wird. Die Parameter der Filter wurden experimentell bestimmt und sind von der Abtastrate des Signals abhängig.

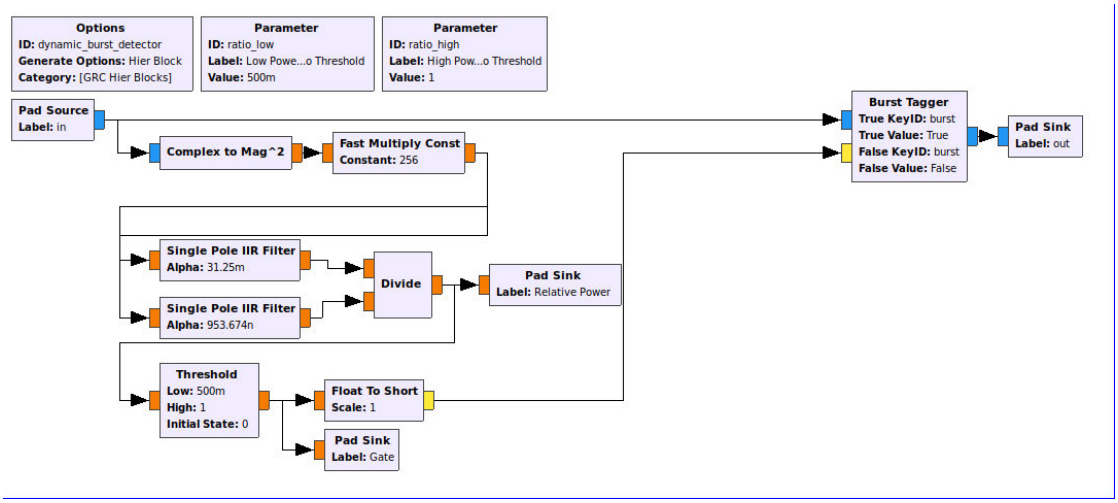

Abbildung 3.2: GNU Radio Flowgraph des Dynamic Burst Detector Blocks

Danach wird das Verhältnis zwischen geglättetem Energiesignal und der Energie des Grundrauschens errechnet. Dieses Verhältnis wird durch eine `Pad Sink` nach außen als Signal verfügbar gemacht. So kann das Signal-Rausch-Verhältnis visualisiert und zur Anpassung der SDR Parameter genutzt werden.

Das Signal-Rausch-Verhältnis passiert danach einen `Threshold` Block. Der obere und der untere Schwellwert des `Threshold` Blocks können über Parameter von außen gesetzt werden (siehe Abbildung 3.4). Dieser implementiert eine Hysterese. Übersteigt das Signal-Rausch-Verhältnis den oberen Schwellwert, gibt der Block `1`en aus, untersteigt es den unteren, `0`en.

Nach einer Typumwandlung wird dieses Signal an den Trigger-Eingang von einem `Burst Tagger` Block angeschlossen. Bei jeder Änderung des Signals auf dem Trigger-Eingang wird der dazugehöre Messwert vom anderen Eingang des Blocks mit einem Tag versehen. Ändert sich das Triggersignal auf 1, so wird ein “burst:True” anghängt. Dieses markiert den Start eines Frames. Bei der Triggeränderung zu 0 wird zur Markierung des Endes “burst:False” angehängt.

So werden anhand des Signal-Rausch-Verhältnis Frames erkannt und markiert.

Die durch Hysterese gewonnene Stabilität und die automatische Anpassung ans Grundrauschen erwiesen sich als vorteilhaft in den weiteren Implementierungen. Der Dynamic Burst Detector wird im Flowgraph in Abbildung 3.4 verwendet.

3.2.3 Automatische Symbolratenerkennung

GNU Radio kommt mit einigen Standardblöcken, die der Taktrückgewinnung dienen. Ein Beispiel hierfür ist die Mueller and Müller Clock Recovery (MMCR). Hierbei wird bei jedem Messwert mithilfe einer Fehlermetrik eine interne Uhr an die Symbolrate aus dem Signal angepasst [21]. Dies ist ein Algorithmus von 1976 und für die Symbolratenerkennung bei durchgehenden Übertragungen geeignet.

Da diese Arbeit das Empfangen kurzer Übertragungen fordert, ist MMCR kein gangbarer Weg.

Zu den Vorteilen von Software Radios zählt, dass lange Aufzeichnungen von Signalen im Speicher des Computers gehalten werden können. So können Methoden der Symbolratenerkennung implementiert werden, die eine lange Abfolge von Messwerten betrachten, statt jeden einzeln.

Auf dieser Idee basiert Michael Ossmans Whole Packet Clock Recovery (WPCR) [33]. NRZ kodierte Daten wechseln den Signalpegel synchron mit der Symbolrate. Bei WPCR werden die Flanken eines gegebenen NRZ Signals in eine Serie von Dirac-Pulsen umgewandelt. Diese Pulskette ist an den Stellen unterbrochen, an denen im NRZ Signal kein Pegelwechsel vorliegt. Dies wird in Abbildung 3.3 veranschaulicht.

Der obere Plot präsentiert ein Signal von zufälligen Bits, die mit 6,4kSymb/s NRZ kodiert sind, als Funktion der Zeit. Der Plot darunter zeigt die resultierende Pulskette als Funktion der Zeit. In diesem Versuch existiert bei der Darstellung ein Zeitversatz, weswegen die Pulse nicht an den Flanken ausgerichtet sind. Der Zusammenhang zwischen Flanken und Pulsen ist dennoch erkennbar. Unten ist ein FFT Plot des Pulssignals zu sehen. Hierbei wird die relative Energie im Spektrum als Funktion der Frequenz dargestellt. Es sind drei Peaks erkennbar. Einer ist bei 0 Hz (ganz links), da im Signal ein Gleichspannungsversatz besteht. Der zweite Peak ist bei 6,4 kHz. Das ist die Symbolrate. Der dritte Peak befindet sich bei der doppelten Symbolrate (12,8 kHz).

Es ist erkennbar: Wenn die Pulskette durch Fourier Transformation von der Zeit- in die Frequenzdomäne umgewandelt wird, so entspricht der Peak mit der geringsten Frequenz, die nicht 0 Hz ist, der Symbolrate.

Das Listing 3.1 wurde dem Kommandozeilenprogramm von Michael Ossmann entnommen [34]. `a` enthält den demodulierten Frame, `d` die Pulskette. `scipy.fft` liefert ein Spektrum äquivalent zum unteren Plot von Abbildung 3.3. Dieses wird in `f` gepeichert.

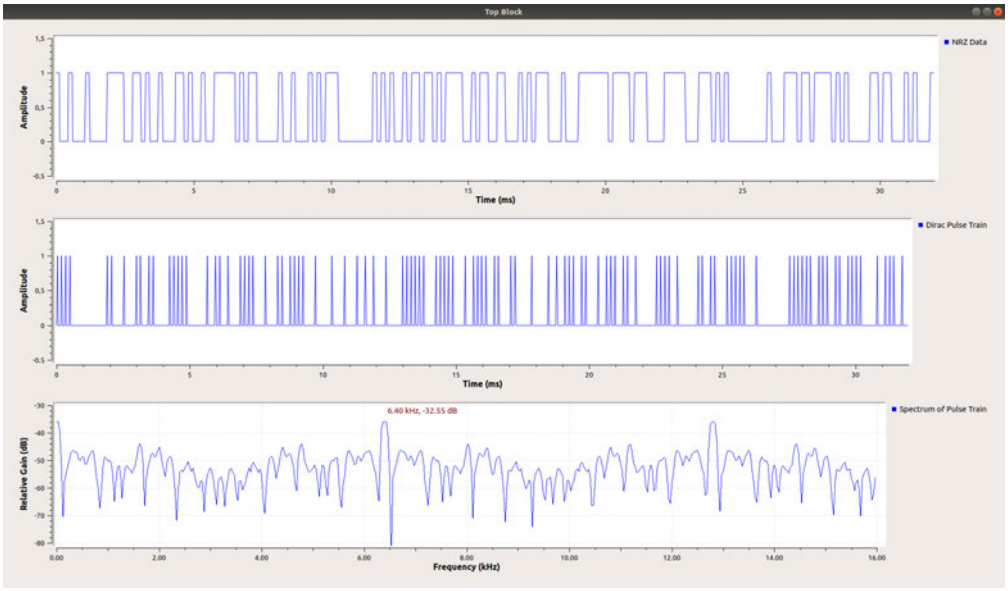

Abbildung 3.3: Ermittlung der Symbolrate eines NRZ Signals durch Fast Fourier Transformation

`find_clock_frequency` implementiert einen Peak-Detection-Algorithmus. Wenn in `f` ein Peak `p` gefunden wurde, wird mit diesen Variablen die Symbolrate und -phase ermittelt. Mit ermittelter Rate und Phase wird so über das Signal iteriert, dass in der Mitte eines jeden Symbols abgetastet wird.

Basierend auf diesem Code wurde ein Block für GNU Radio entwickelt. Eine Verwendung von diesem ist im Abbildung 3.4 zu sehen.

Am Anfang des Graphen steht ein `File Source` Block, welcher rohe Messwerte aus einer Datei in Dauerschleife ausgibt. Diese wurde vorher mit einem SDR aufgenommen. Zusammen mit dem `Throttle` Block verhält sich das wie ein SDR, das immer wieder das Signal empfängt, das in der Datei aufgezeichnet ist. Dies dient einerseits der einfacheren Analyse einer Aufzeichnung, aber auch dem Debuggen von Flowgraphs.

Die beiden folgenden Filter und der Squelch Block isolieren das Signal. Der in Abschnitt 3.2.2 behandelte `Dynamic Burst Detector` Block markiert Frames im Signal. Mit dem `Quadrature Demod` Block wird aus dem BFSK ein NRZ-Signal. Der darauf folgende Filter glättet das Signal.

Danach kommt der `Whole Packet Clock Recovery` Block. Dieser ist in Python geschrieben und schließt die `wpcr` Fuktion von Michael Ossmann an GNU Radio an.

Listing 3.1: Abschnitt aus der Funktion `wpcr` von Michael Ossmann

```
f = scipy.fft(d, len(a))
p = find_clock_frequency(abs(f))
if p == 0:
   return []
cycles_per_sample = (p*1.0)/len(f)
clock_phase = 0.5 + numpy.angle(f[p])/(tau)
if clock_phase <= 0.5:
   clock_phase += 1
symbols = []
for i in range(len(a)):
   if clock_phase >= 1:
      clock_phase -= 1
      symbols.append(a[i])
   clock_phase += cycles_per_sample
```

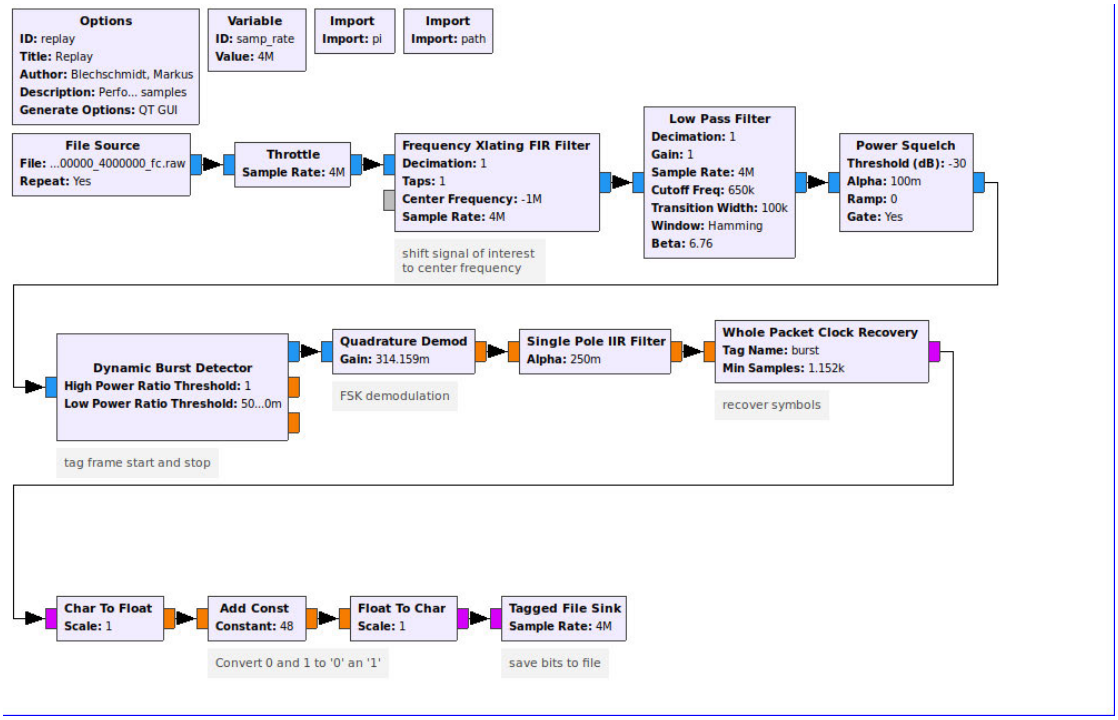

Abbildung 3.4: GNU Radio Flowgraph mit Block basierend auf WPCR

Hierzu werden, nach Eingang eines mit "burst:True" getaggtem Messwertes, alle folgenden Messwerte in einen Buffer geschrieben. Sobald ein Messwert mit dem Tag "burst:False" eingeht und der Buffer mit genug Samples gefüllt ist, wird mit `wpcr` versucht, Symbole aus dem Buffer zu extrahieren. Ist dies erfolgreich, kommen die Symbole aus dem Ausgang des Blocks. Schlussendlich wird der Buffer geleert und der Zyklus beginnt erneut.

Die folgenden Blöcke dienen der Typenumwandlung und Speicherung. Der `Tagged File Sink` Block legt anhand der Tags für jeden erkannten Frame eine Datei an, in der die Symbole des Frames gespeichert werden. So kann beispielsweise mit Systemtools wie `grep` nach Gemeinsamkeiten zwischen den Symbolketten gesucht werden.

Während dieser Flowgraph als Proof of Concept funktioniert und Symbole korrekt extrahiert, haben sich bei der Benutzung Fehler gezeigt, deren Ursache bisher nicht eindeutig zurückgeführt werden konnte. Eine Vermutung ist, dass das GNU Radio Framework nicht gut mit Blocks umgehen kann, die sporadisch (in Abhängigkeit vom Inhalt des Signals) Samples generieren, wodurch es zu Skipping in den Buffern kommt. Das Problem konnte bisher auch noch nicht im Dialog mit GNU Radio Entwickler_innen gelöst werden. Aufgrund dessen kam diese Implementierung bei den Versuchen nicht zum Einsatz.

3.2.4 Automatische Alignmenterkennung

Funkkommunikation ist anfällig gegenüber Interferenzen und anderen Störungen. Um die Integrität der empfangenen Daten prüfen zu können, werden diese meist mit einer Prüfsumme versehen.

Wenn mehrere aufgezeichnete Pakete den gleichen Inhalt haben, kann gefolgert werden, dass dieser Inhalt fehlerfrei empfangen wurde. Somit muss auch die Prüfsumme darin intakt sein.

Mithilfe einiger verschiedener Pakete, die als intakt verifiziert sind, ist es möglich, das Alignment und die Prüfsummenparameter zu ermitteln.

CRC Summen

Cyclic Redundancy Check (CRC) ist ein Prüfsummenverfahren, welches aufgrund seiner einfachen Implementierung bevorzugt in digitaler Kommunikation genutzt wird. CRC-Summen haben definierende Parameter, welche bei proprietären Protokollen nicht be-

kannt sind. Zu diesen zählen unter anderem die Länge, das verwendete Polynom und der Initialisierungsvektor.

RevEng ist ein Kommandozeilenprogramm, welches genutzt werden kann, um diese CRC Parameter aus einer Menge von Beispieldaten mit Prüfsumme zu ermitteln. Hierbei muss eine Annahme über die Länge der Prüfsumme getroffen werden. Die Beispieldaten müssen als Hexadezimalstrings übergeben werden und die Prüfsumme darin am Ende Byte-aligned sein.

Wie in den folgenden Kapiteln vorgestellt wird, beginnen Pakete vom gleichen Gerät oft mit gleichen Symbolen (Bits). Diese können genutzt werden, um die Pakete untereinander auszurichten.

Um das Byte-Alignment festzustellen, kann RevEng mit jedem möglichen Alignment aufgerufen werden. Eine gefundenes Prüsummenverfahren indiziert ein korrektes Alignment. Hierbei wird die Annahme getroffen, dass die Prüfsumme in den letzten Symbolen eines Paketes ist und das Paketende korrekt erkannt wurde.

Der Code in Listing 3.2 implementiert dieses Verfahren. `bitstrings` enthält eine Liste von bereits untereinander ausgerichteten Bits.

Listing 3.2: Codeabschnitt zur automatischen Alignmenterkennung

```
# probiere alle moeglichen CRC Laengen
for crc_len in (16, 8):
   # iteriere ueber alle moeglichen Alignments
   for start_idx in range(crc_len):
      num_bytes = (len(bitstrings[0]) - start_idx) // 8
      format_string = "{{:0{:d}x}}".format(num_bytes)
      # erstelle Sub-Bitstrings
      sub_bitstrings = [
         bitstring[start_idx:start_idx+num_bytes*8]
         for bitstring in bitstrings
      ]
      # packe bits in Hexadezimalstring
      hex_strings = [
         format_string.format(int(sub_bitstring, 2))
         for sub_bitstring in sub_bitstrings
      ]
```

```
    # fuehre RevEng aus
    reveng = run(["./reveng-2.1.0/reveng", "-w", str(crc_len)
        ↪ , "-s"]
        + hex_strings, stdout=PIPE, stderr=DEVNULL)
    if 0 == reveng.returncode:
        print("skip: {:d}".format(start_idx))
        print("num bytes: {:d}".format(num_bytes))
        print("(num bits: {:d})".format(num_bytes*8))
        print("CRC: {}".format(reveng.stdout.decode()))
```

Die Ausgabe des Programms (siehe Listing 3.3) enthält das Alignment (wie viele Bits zu Beginn übersprungen werden müssen), die Anzahl der Bytes im Paket und die Parameter der verwendeten CRC-Summe. Wie sich zeigt, werden zwei mögliche Alignments identifiziert. Dies liegt daran, dass die Pakete in diesem Beispiel sich erst ab dem 76. Bit unterscheiden.

Die Parameter der CRC-Summe können zur Integritätsprüfung anderer empfangener Pakete genutzt werden.

Listing 3.3: Beispielausgabe des Codes 3.2

```
skip: 7
num bytes: 27
(num bits: 216)
CRC: width=16 poly=0x1021 init=0x1b62 refin=false refout=false
    ↪ xorout=0x0000 check=0x6498 residue=0x0000 name=(none)

skip: 15
num bytes: 26
(num bits: 208)
CRC: width=16 poly=0x1021 init=0xcb0a refin=false refout=false
    ↪ xorout=0x0000 check=0x446f residue=0x0000 name=(none)
```

4 Versuche

In diesem Kapitel werden die verwendete Werkzeuge und untersuchten Geräte vorgestellt. Danach werden die durchgeführten Versuche beschrieben und erlangten Daten ausgewertet.

4.1 Versuchsaufbau

Wie in Abbildung 4.1 zu sehen ist, wird nur ein Computer (a) genutzt, um die Versuche durchzuführen. An diesen sind die Tastatur (b) über das Dongle (c), sowie das SDR (d) angeschlossen. Für das Dongle wurde hier ein USB-Hub verwendet, um diesen besser sichtbar zu machen.

Das in 4.2 abgebildete SDR (a) ist ein XTRX Pro in einem USB 3.0 Adapter. Dies ist ein 2x2 MIMO SDR, was bedeutet, dass es über zwei Empfangs- und zwei Sendeanschlüsse verfügt. Zwar wird in keinem der Versuche gleichzeitiges Senden und Empfangen verwendet, dennoch ermöglicht dies das Umschalten zwischen zwei Antennenarten. An einem Empfangsport des SDRs ist eine Vivaldi-Antenne (b) angeschlossen, am anderen eine 2,4 GHz Dipolantenne (c). Das XTRX Pro nutzt die aktive GPS-Antenne (d) für einen GPS-synchronisierten Oszillator. Dadurch wird Clock-Drift vermindert, was zur Qualität des Signals beiträgt [8].

In manchen Versuchen wurde anstelle des XTRX ein rad1o benutzt. Dieses ist in Abbildung 4.3 zu sehen. Das rad1o verfügt über nur einen Antennenanschluss, über den entweder gesendet oder empfangen werden kann. Zur Verbindung mit dem Computer nutzt es USB 2.0.

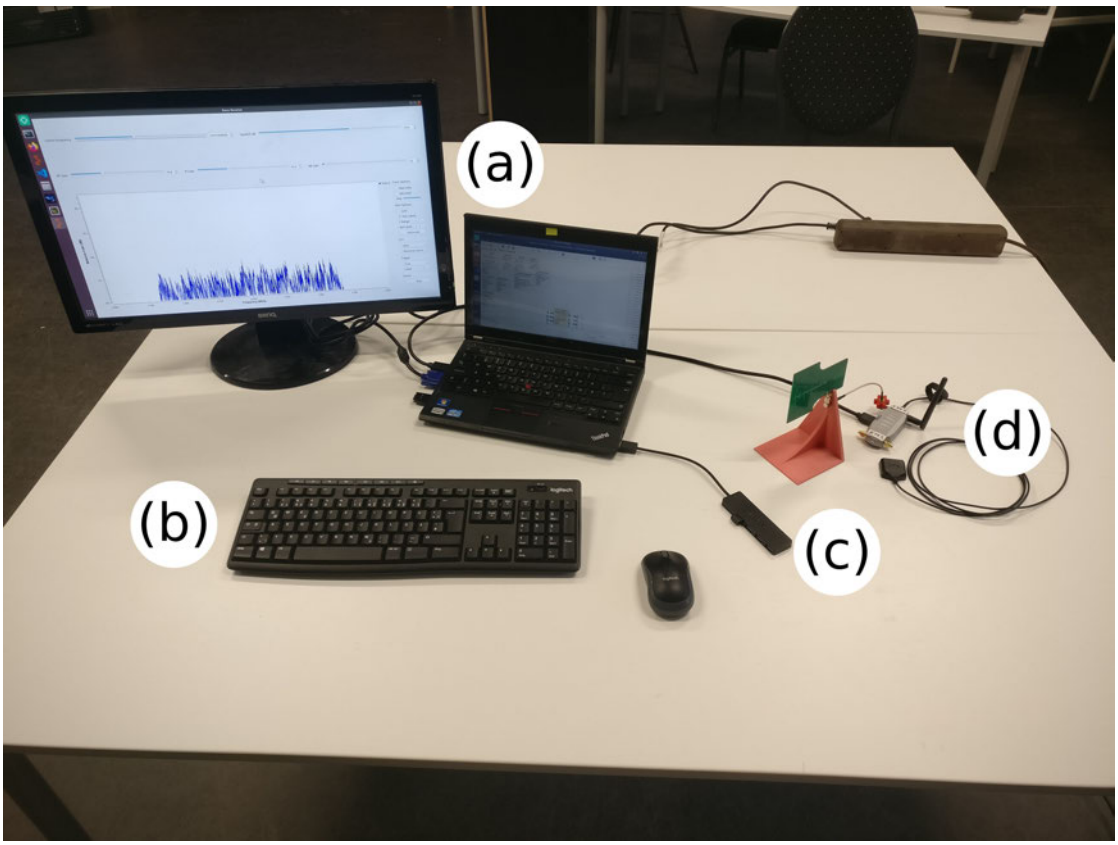

Abbildung 4.1: Annotierter Versuchsaufbau

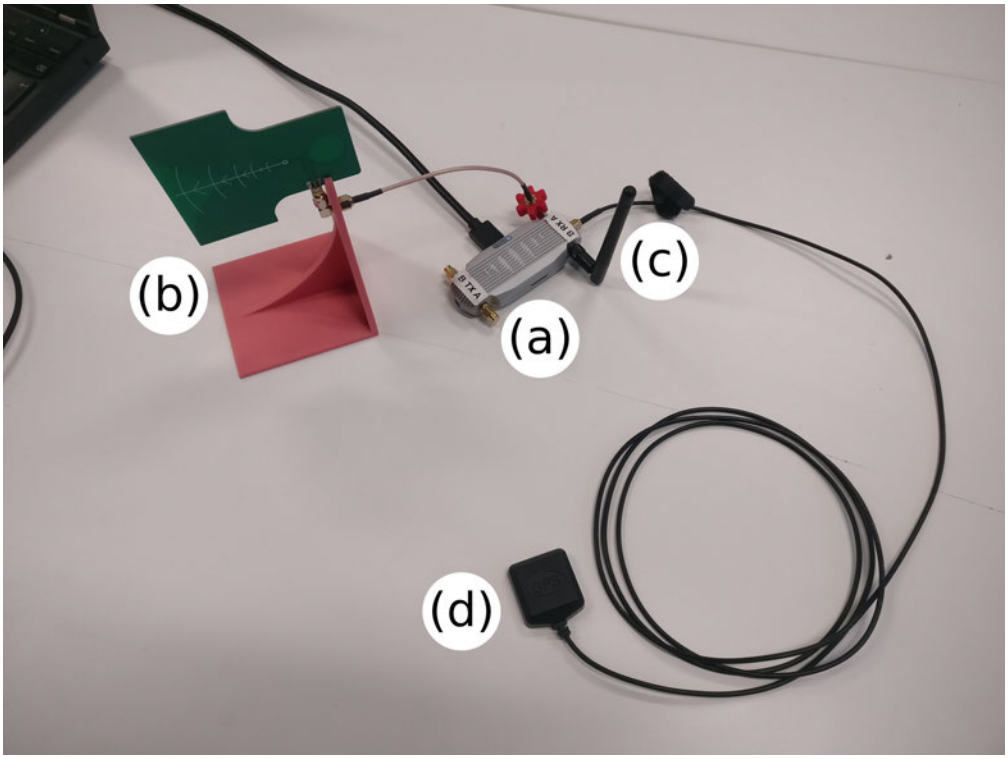

Abbildung 4.2: Annotiertes XTRX mit Antennen

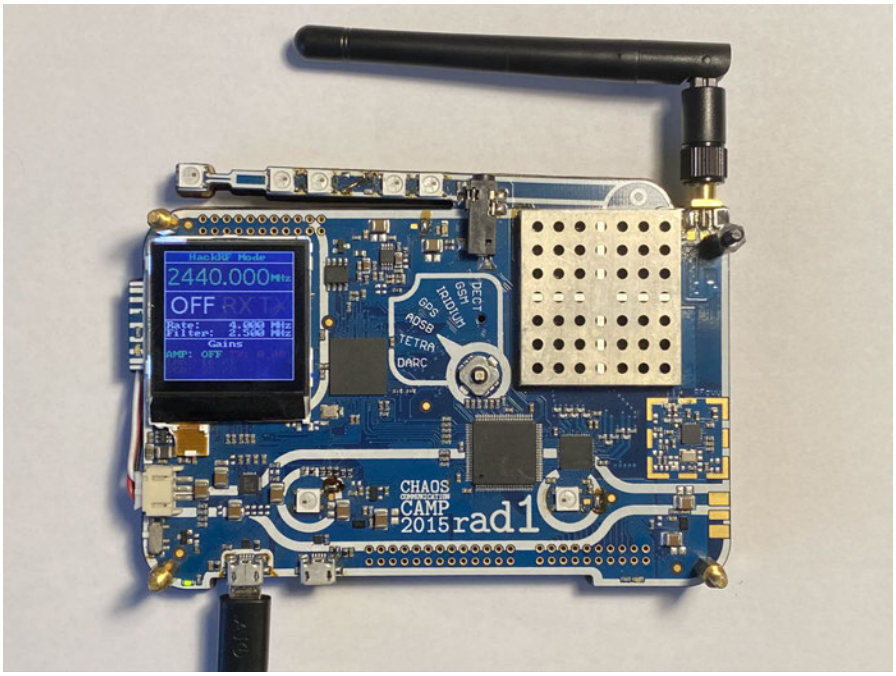

Abbildung 4.3: rad1o

4.1.1 Werkzeuge

Für die Umsetzung dieser Arbeit wurden vorwiegend quelloffene Hardware und Software genutzt.

Hardware

In dieser Arbeit wurden zwei verschiedene SDRs verwendet.

SDR	LO Frequenz in MHz	Abtastrate in MSamp/s
rad1o	1 bis 4000	20
XTRX Pro	0,1 bis 3800	160

Tabelle 4.1: Liste der verwendeten SDRs mit Informationen von [24]

Das rad1o verhält sich wie ein HackRF One [20].

Antennen Zum Empfangen wurden 2,4 GHz Dipolantennen und eine Vivaldi-Antenne verwendet. Vivaldi-Antennen haben den Vorteil, dass mit ihnen ein breites Frequenzband empfangen werden kann. Darüber hinaus sind sie gerichtet, wodurch sich auf die Signalquellen zielen lässt. Beide Eigenschaften erleichtern die Suche nach Signalen, die von Interesse sind.

Zum Senden wurden nur 2,4 GHz Dipolantennen verwendet, da Vivaldi-Antennen nicht zwingend über eine kontinuierliche Impedanz über die gesamte Bandbreite verfügen. Abrupte Impedanzenänderungen im Signalpfad führen zu Reflektionen, was beim Senden den PA beschädigen kann.

Computer Der in dieser Arbeit zur digitialen Signalverarbeitung genutzte Computer ist ein Lenovo ThinkPad X230 Laptop mit Intel Core i5-3320M CPU und 8GB RAM.

Software

Die in dieser Arbeit verwendeten Softwarekomponenten sind in Tabelle 4.2 aufgelistet.

Name	Version	Beschreibung
GNU Radio	3.7.11-10	Entwicklungs-Toolkit, das Signalverarbeitungsblöcke zur Implementierung von Software-Radios bereitstellt
gr-osmosdr	0.1.8.0.1.4-14build1	GNU Radio Backend, Treiber
Linux	4.15.0-132-lowlatency	Kernel
RevEng	2.1.0	CRC-Rechner und Algorithmus-Finder
SDRangel	4.2.3	SDR Frontend
Ubuntu	18.04.5 LTS	Betriebssystem
Universal Radio Hacker	2.8.9	Suite zur Untersuchung von Funkprotokollen
xtrx	siehe Anhang B.1	Treiber

Tabelle 4.2: Liste der verwendeten Software

Aspekt	Tastatur
niedrigster Preis	Docooler
beste Bewertung	Sonkir K-20
meiste Stückzahl verkauft	Logitech K270
beworbene Verschlüsselung	AmazonBasics

Tabelle 4.3: Liste der untersuchten Geräte

Um das XTRX mit GNU Radio und SDRangel verwenden zu können, wurden gr-osmosdr und SDRangel aus den Forks der XTRX Developer gebaut. Die Anleitung hierzu befindet sich in Anhang B.

4.1.2 Geräteauswahl

Für die Auswahl der Geräte wurden Aspekte in Betracht gezogen, die Einfluss auf die Verbreitung der Geräte haben. Als solche werden hier Preis, Kundenrezensionen, verkaufte Stückzahl und beworbene Verschlüsselung gewertet. Diese Aspekte beeinflussen die Kaufentscheidung vieler Kunden und werden von vielen Onlineshops als Filter- und Sortieroptionen angeboten. Die Tastaturen wurden am 16.08.2020 von amazon.de bezogen. Die Auswahl ist in Tabelle 4.3 zu sehen.

4.2 Voruntersuchung

Bei den Voruntersuchungen sollen Erkenntnisse gewonnenen werden, die beim Entwurf der weiteren Versuche relevant sein können.

4.2.1 Benutzung

Die grundlegende Funktionsweise ist bei allen Geräten gleich. Zu jeder Tastatur gehört ein USB-Dongle (und ggf. eine Maus). Die Reihenfolge von Einschalten der Tastatur oder Einstecken des Dongles ist irrelevant. Software zum Verwalten der Verbindung wird nicht angeboten. Somit kann gefolgert werden, dass die Kopplung zwischen Tastatur und Dongle während der Herstellung passieren muss.

4.2.2 USB

Um weitere Informationen zu den USB-Donglen zu erhalten, wurden die Ausgaben der Systemtools dmesg, lsusub und usb-devices betrachtet. Die Ausgaben sind in Anhang A dokumentiert. Darin ist zu sehen, dass für alle Dongle der `hid-generic` Treiber vom System geladen wird. Somit erscheinen diese äquivalent für das OS wie kabelgebundene Tastaturen. Vermutlich sind HID Keyboard Codes [39] in den gesendeten Daten enthalten.

4.2.3 Empfang

In diesem Abschnitt wird beschrieben, wie das Sendeverhalten zu jedem Gerät ermittelt wurde und welche Auswirkung dies auf die weiteren Versuche hat.

Spektrum

Um herauszufinden, welche Frequenzen von den Geräten genutzt werden, wurde das Spektrum mit SDRangel betrachtet (vgl. Abbildung 4.4). Das XTRX ist hierbei auf Grund der höheren Abtastrate von Vorteil. Der Modus des Wasserfalldiagramms in SDRangel wird auf Max-Hold gestellt. Hierdurch bleiben die kurzen Signale länger sichtbar.

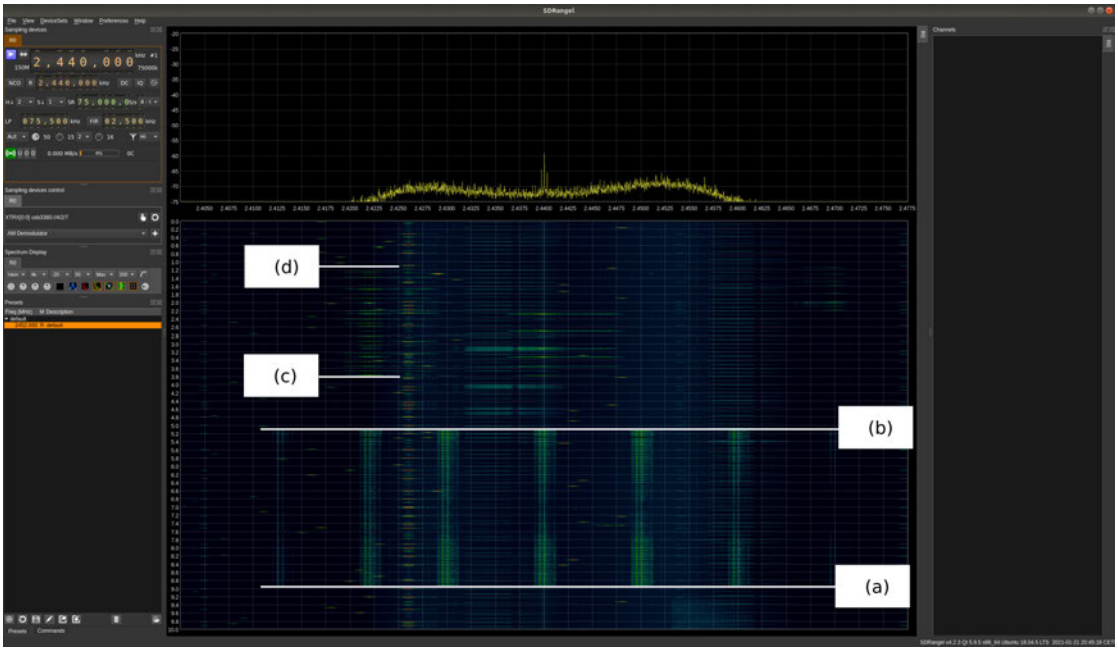

Abbildung 4.4: Annotiertes Wasserfalldiagramm in SDRangel mit Signalen von Sonkir K-20 Tastatur und Dongle

Zum Ermitteln der Kanalfrequenzen werden die Tastaturen ohne angeschlossene Dongle eingeschaltet. In Abbildung 4.4 ist zu sehen, wie zu Zeitpunkt (a) die Sonkir K-20 Tastatur eingeschaltet wird. Danach ist das Senden von Signalen auf mehreren Kanälen zu sehen. Zu Zeitpunkt (b) wurde der Dongle in den Computer gesteckt. Daraufhin hören die Signale auf. Vermutlich sind diese ähnlich zu BLE Advertisements und dienen dem Verbindungsaufbau zum Dongle.

Zwischen (c) und (d) wurde die Shifttaste wiederholt gedrückt. Hier sieht man, dass die Kommunikation auf nur einem Kanal stattfindet. Wenn einn paar Sekunden lang keine Taste gedrückt wird, werden in regelmäßigem Zeitabstand kurze Signale gesendet - vermutlich eine Art Heartbeat.

Docooler, Sonkir K-20 und Logitech MK270 zeigen alle das soeben beschriebene Verhalten. Die Frequenzen der Kanäle sind in Anhang A dokumentiert. Hieran ist zu erkennen, dass Docooler ein 8 MHz und Logitech ein 3 MHz Kanalraster nutzt. Fünf der acht Kanäle von Sonkir fallen in ein 10 MHz Kanalraster.

Das Verhalten der AmazonBasics Tastatur ist in Abbildung 4.5 zu sehen.

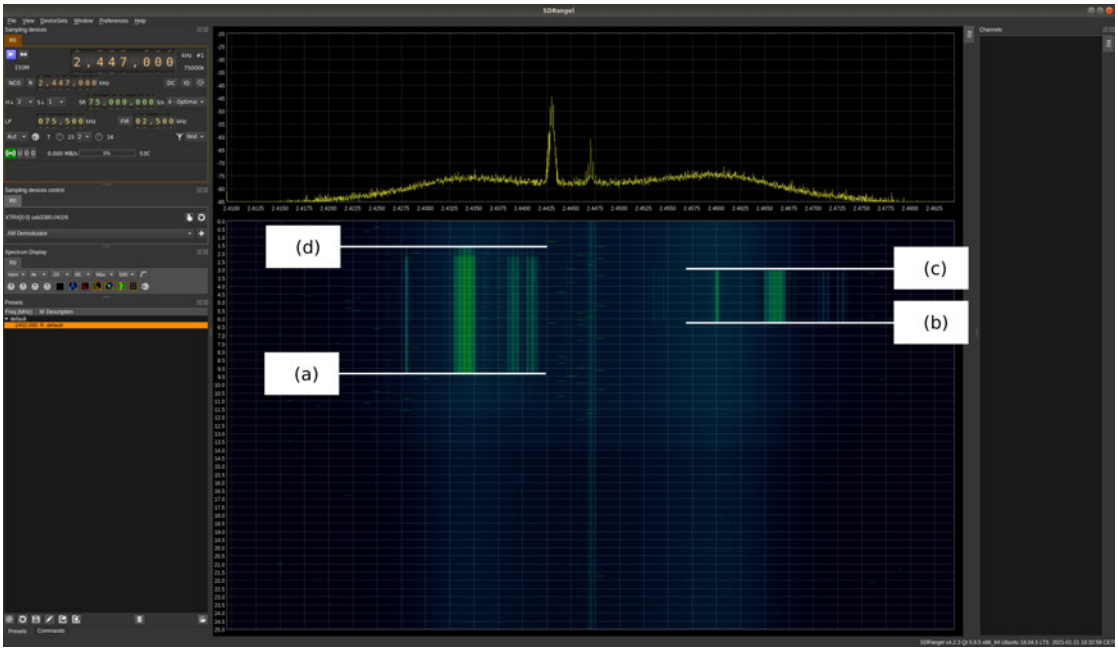

Abbildung 4.5: Annotiertes Wasserfalldiagramm mit Signalen von AmazonBasics Tastatur und Dongle

Bei (a) wurde das Dongle in die Tastatur gesteckt. Zwischen (b) und (c) wurde die Shifttaste wiederholt gedrückt. Zur Zeit (d) wurde das Dongle wieder abgeschlossen.

Darüber hinaus konnte beobachtet werden, dass die Tastatur ohne das Signal des Dongles selbst keine Signale sendet.

Da die untersuchten Geräte keinen Frequenzwechsel während einer Verbindung vollführen, wurde von Methoden zum Empfangen auf allen Kanälen abgesehen. Versuche haben gezeigt, dass die in 3.2.1 erwähnte Minderung der Signalqualität nicht den Vorteil überwiegt, den Kanal nicht suchen zu müssen. Oftmals reichte es aus, auf einem Kanal zu empfangen und die Tastatus aus- und wieder einzuschalten, bis der beobachtete Kanal genutzt wird.

Infolgedessen wurde Universal Radio Hacker (URH) für alle weiteren Untersuchungen verwendet. URH unterstützt zum Zeitpunkt der Versuche leider nicht das XTRX. Daher wird von hier an mit dem rad1o fortgefahren.

URH verfügt über einen *Spectrum Analyzer*, welcher einer funktionsärmeren Version von SDRangel gleicht.

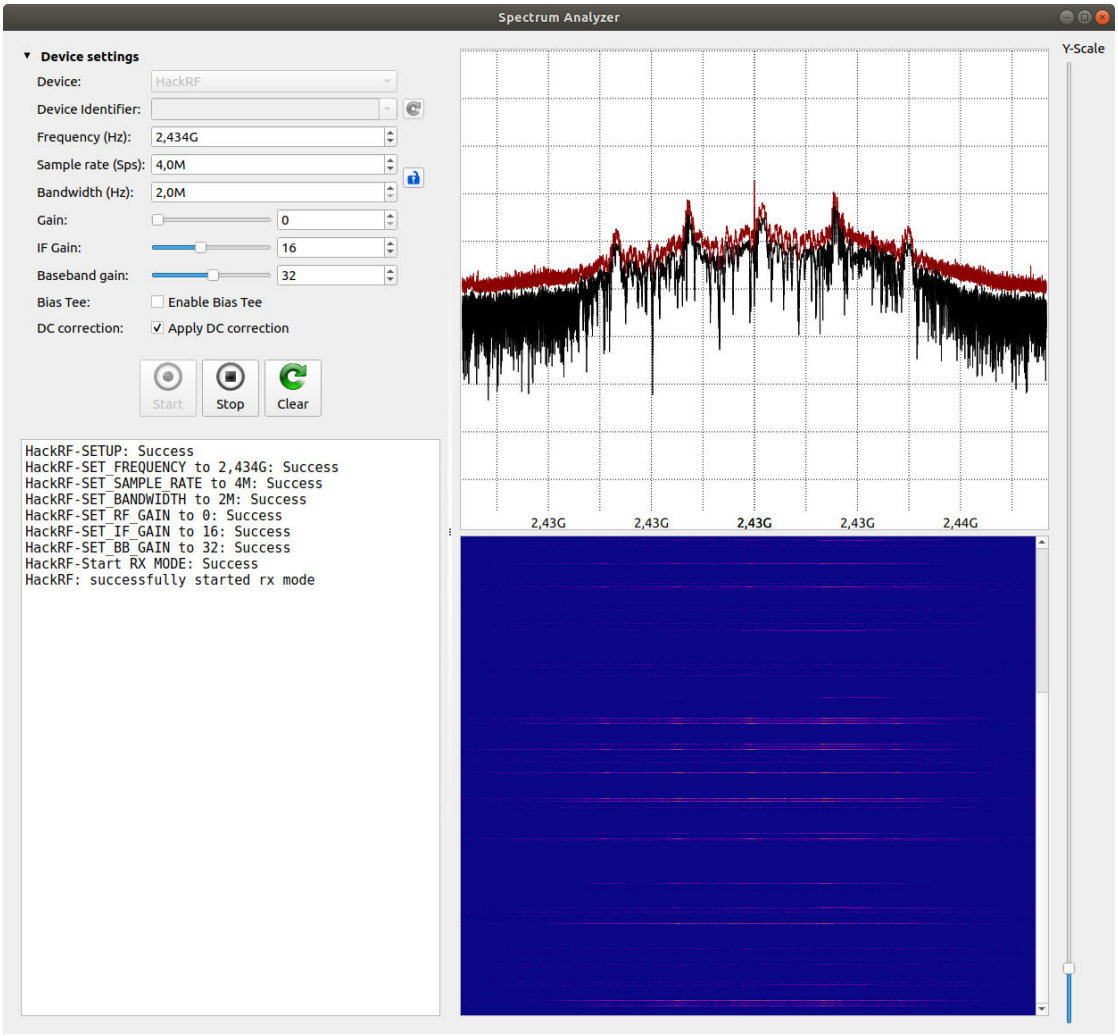

Abbildung 4.6: URH *Spectrum Analyzer* Fenster mit Signalen vom AmazonBasics Dongle

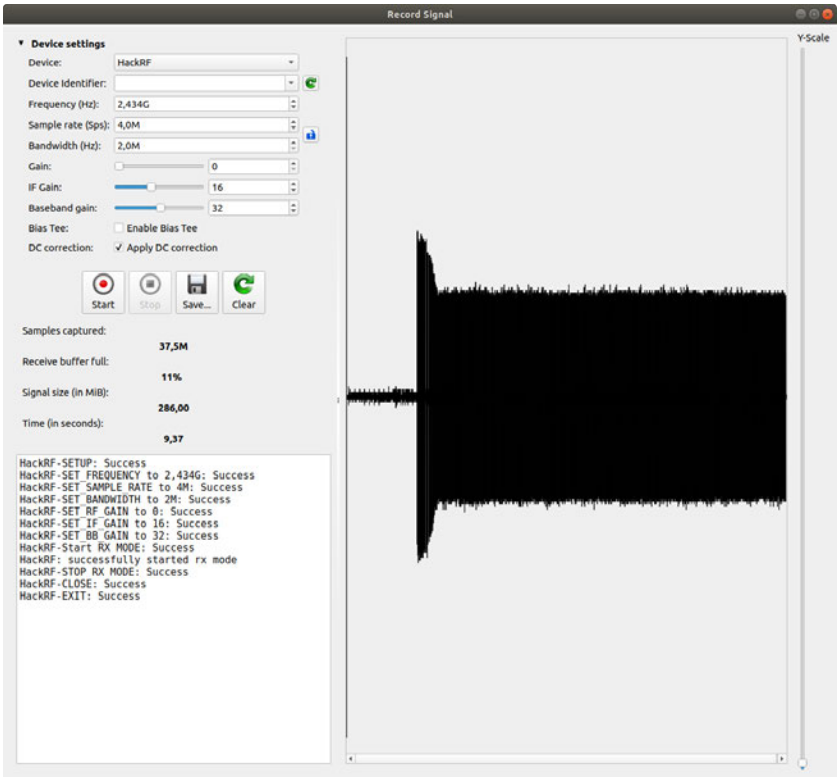

Abbildung 4.7: URH *Record Signal* Fenster mit Aufzeichnung zum Moment des Einsteckens des AmazonBasics Dongles

Wie in Abbildung 4.6 zu sehen ist, können hier die LO-Frequenz, Abtastrate sowie Filter-, Verstärker- und Operationsparameter des SDRs eingestellt werden. Das Spektrogramm und Wasserfalldiagramm auf der rechten Seite dienen der Inspektion des Signals.

In 4.2.3 wurde festgestellt, dass im Fall der Docooler, Sonkir K-20 und Logitech MK270 Tastaturen diese nur aus- und angeschaltet werden müssen, bis das Signal auf dem beobachteten Kanal erscheint. Im Fall der AmazonBasics gibt es nur zwei Frequenzen, die von Interesse sind.

Wenn das Signal im *Spectrum Analyzer* gut sichtbar ist, kann das Fenster geschlossen und die *Record Signal* Funktion von URH geöffnet werden. Die Einstellungen des *Spectrum Analyzer* werden dabei übernommen.

Das Signal wird beim Aufzeichnen in Echtzeit geplottet (siehe 4.7). Hierdurch kann gesehen werden, wie verschiedene Interaktionen (z.B. das Drücken einer Taste) sich auf das Signal auswirken. Nach dem Speichern einer Aufzeichnung kann der Buffer geleert

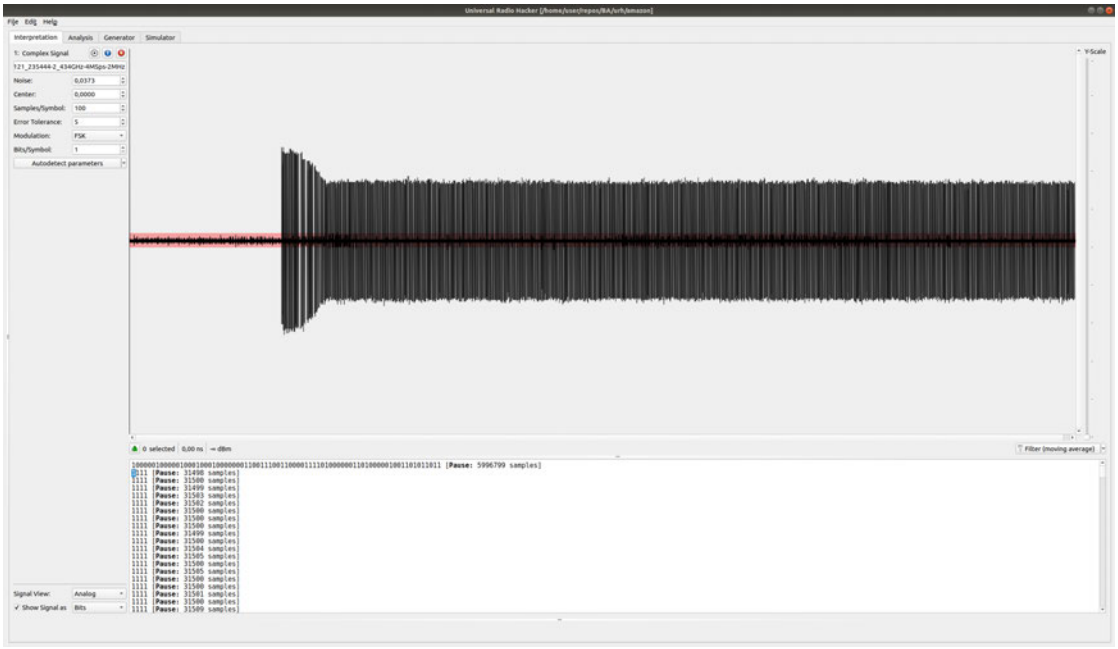

Abbildung 4.8: URH *Interpretation* Tab mit dem in Abbildung 4.7 aufgezeichneten Signal

werden und eine neue Aufzeichnung beginnen. So können Versuchsparameter, wie die gedrückte Taste, schnell variiert und im Dateinamen vermerkt werden. Das ist hilfreich bei der Korrelationsanalyse.

Wenn das Record Signal Fenster geschlossen wird, öffnen sich die gespeicherten Aufzeichnungen im Interpretation Tab (siehe 4.8).

4.3 Injection

Die Möglichkeit der Injection von Tastendrücken soll durch das Demonstrieren eines Replayangriffs getestet werden. Ist dies möglich, könnte von jeder beliebigen Taste das Signal aufgezeichnet, isoliert und in gewünschter Reihenfolge gesendet werden. Somit wären beliebige Eingaben möglich.

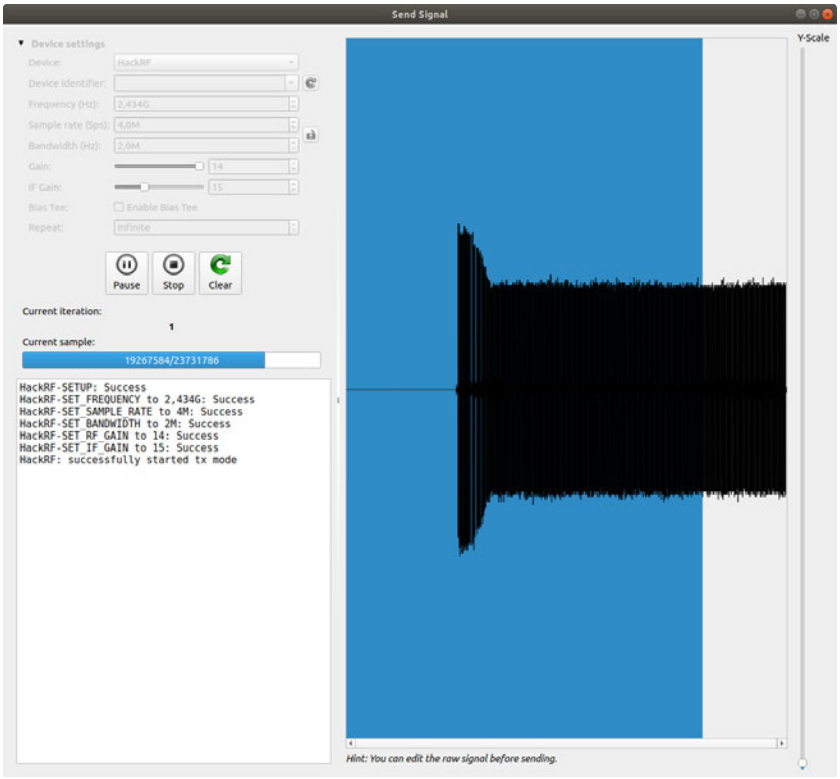

Abbildung 4.9: URH *Send Signal* Fenster mit dem in Abbildung 4.7 aufgezeichneten Signal

4.3.1 Replay

URH ermöglicht die Retransmission der aufgezeichneten Signale. Wenn bei einem geöffneten Signal auf den Button mit dem Play Icon gedrückt wird, öffnet sich das *Send Signal* Fenster mit dieser Aufzeichnung (siehe Abbildung 4.9).

Auch hierbei wurden die Einstellungen des Spectrum Analyzer übernommen. Es sei angemerkt, dass der RF Gain auf +14dB gestellt ist. Dadurch wird beim rad1o der PA aktiviert, was beim Senden von aufgezeichneten Signalen notwendig ist. Für einen Versuch, der länger als die Aufzeichnung laufen soll, kann das Signal in einer Endlosschleife gesendet werden.

4.3.2 Ergebnisse

Ein Replayangriff war bei allen Geräten grundlegend möglich. Oftmals erschienen einige der aufgezeichneten Tastendrücke nicht.

Dies lässt sich eventuell darauf zurückführen, dass die aufgezeichneten Signale Sendungen von Tastatur **und** Dongle beinhalten. Ein anderes Problem könnten Sequenznummern sein, welche in unerwarteter Reihenfolge vom Dongle empfangen werden. Dadurch kann es zu Fehlern im Protokoll kommen.

Mit der AmazonBasics Tastatur war es möglich, das Wort "secret" komplett aufzuzeichnen und zu retransmittieren.

4.4 Sniffing

Um zu zeigen, dass eine Sniffing-Attacke möglich ist, muss eine Korrelation zwischen den gedrückten Tasten und gesendeten Daten hergestellt werden. Die aufgezeichneten Signale müssen hierzu dekodiert werden.

4.4.1 Dekodierung

Im Interpretation Tab von URH können Parameter für das Dekodieren eingestellt werden. Hierzu gehört z.B. der Noise Schwellwert, welcher im Plot durch das rote Band dargestellt wird (siehe Abbildung 4.8). Alle Messwerte, deren Amplitude geringer als das Band ist, werden von der Demodulation ausgeschlossen.

Demodulation

In URH kann zwischen den Modulationsarten ASK, FSK und PSK gewählt werden. Um zu sehen, welche richtig ist, muss dabei der Signal View von Analog auf Demodulated gestellt sein.

Alle untersuchten Tastaturen nutzen BFSK.

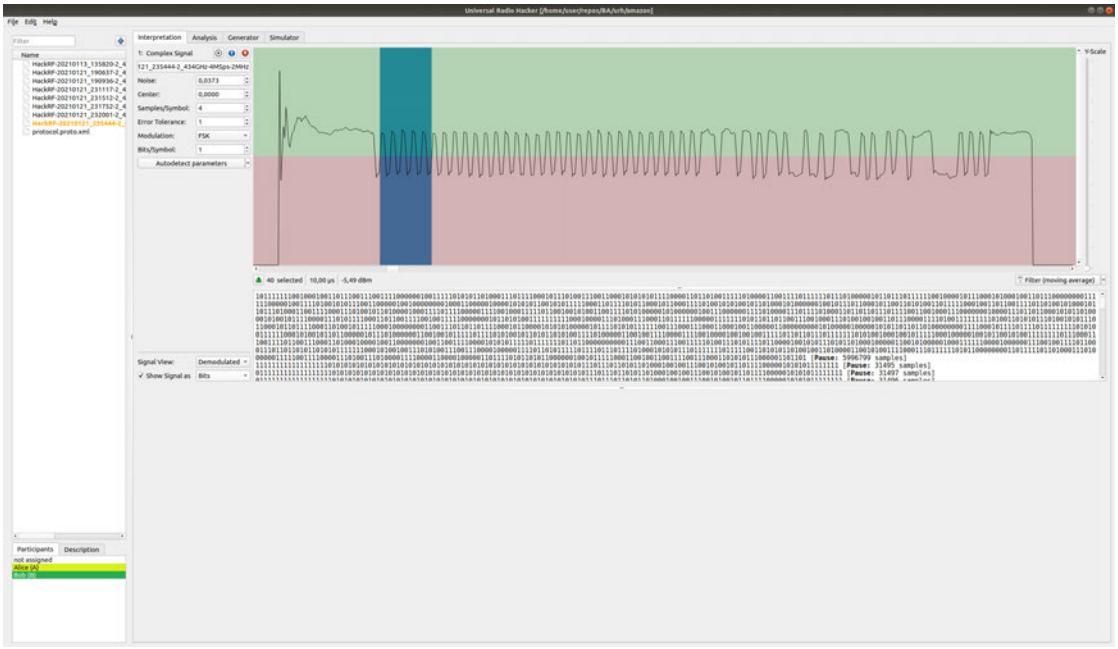

Abbildung 4.10: URH *Interpretation* Tab mit *Demodulated* Signal View des in Abbildung 4.7 aufgezeichneten Signals

Symbolrate

Im Demodulated Signal View kann mithilfe der Cursor gemessen werden, aus wie vielen Messwerten ein Symbol besteht (siehe Abbildung 4.10). Wird dies mit der Abtastrate des Signals multipliziert, erhält man die Symbolrate. Die Symbolraten aller Tastaturen sind in Anhang A dokumentiert.

Paketaufbau

Wenn im Interpretation Tab die korrekten Parameter gefunden wurden, um das Signal zu demodulieren, kann in den Analysis Tab gewechselt werden. In diesem werden die demodulierten Symbole aller Pakete in Tabellenform dargestellt.

Bei der Ausrichtung der Pakete untereinander ist von Hilfe, dass digitale Radiosysteme Pakete mit einer Präambel versehen. Dies ist eine Folge von abwechselnden 1en und 0en, die der Synchronisation beim Empfang dienen. Nach der Präambel folgt meist die immer gleiche Symbolkette (also Bitmuster), die der Identifikation der Pakete dient. Die

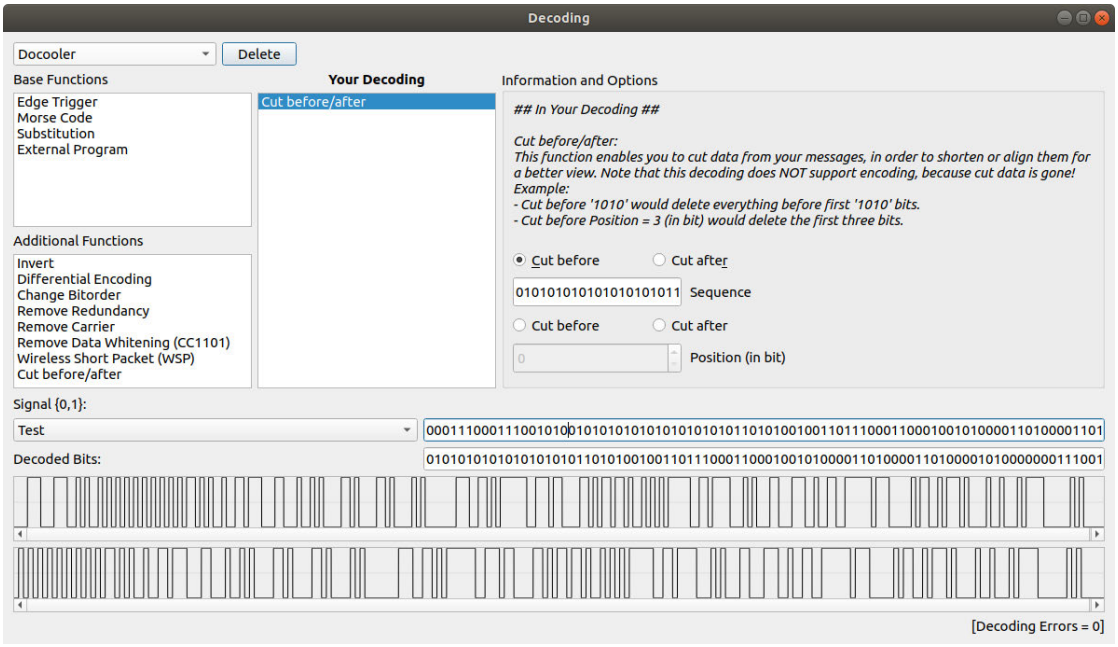

Abbildung 4.11: URH *Decoding* Fenster mit Sequence String, der alle Symbole vor Präambel und Bitmuster entfernt.

Länge der Präambel und das darauffolgende Bitmuster können händisch durch Vergleiche ermittelt werden.

Docooler und Sonkir K-20

Im Falle von Docooler ist die Länge der Präambel mindestens 16 bit. Das darauf folgende Bitmuster ist in 4.1 zu sehen.

Listing 4.1: Identifizierendes Bitmuster der Docooler Tastatur

```
1101010010011
```

Mit der Präambel und dem Muster kann dann ein *Decoding* erstellt werden (siehe Abbildung 4.11). Dies weist URH an, alle demodulierten Symbole an der gleichen Stelle zu trennen. Somit erhalten alle Pakete untereinander die gleiche Ausrichtung.

Im *Analysis* Tab (siehe Abbildung 4.13) können dann mithilfe der *Mark diffs in protocol* Funktion Pakete mit gleichem und unterschiedlichem Inhalt gefunden werden. Diese wurden mit dem in 3.2.4 beschriebenem CRC Werkzeug analysiert. Auf diese Weise konnten das Alignment und die CRC Parameter für die Docooler und Sonkir K-20 Tastaturen

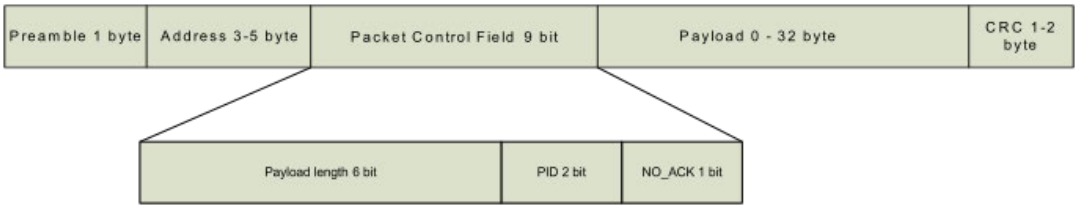

Abbildung 4.12: Ein Enhanced ShockBurst^{TM}-Paket mit Nutzdaten (0-32 Byte) und vergrößertem Paketkontrollfeld aus [29]

ermittelt werden. Diese sind in Anhang A dokumentiert. Mit dem Alignment wurde der Suchstring im *Decoding* (siehe Abbildung 4.11) so angepasst, dass die CRC Summe an Bytegrenzen ausgerichtet ist.

Logitech K270

Das CRC Werkzeug konnte bei der Logitech K270 nur Lösungen finden, wenn die Pakete die gleiche Länge haben. Die Lösungen unterschiedlicher Paketlängen sind nicht gleich.

Da die Logitech K270 eine FFC-ID hat, werden interne Fotos online dokumentiert [12]. Auf diesen ist zu sehen, dass ein $NRF31504E$ verwendet wird. Zu dieser Bauteilnummer wurde kein Datenblatt gefunden. Jedoch sind in der Dokumenation von NRF verwendbare Protokolle gelistet [30].

Darunter ist Enhanced ShockBurst^{TM}. Datenblätter von anderen Produkten, die auch dieses Protokoll implementieren, sind verfügbar. Diese geben Auskunft über den Paketaufbau (siehe Abbildung 4.12) und über die CRC Parameter [29, p. 33].

Diese Parameter wurden in URH übernommen und ergaben erfolgreiche Dekodierungen.

AmazonBasics

Zu den Paketen der Tastatur von AmazonBasics lieferte das CRC Werkzeug leider keine Ergebnisse. Die Fotos der FCC vom Inneren der Tastatur zeigen, dass Chip-on-Board verwendet wurde, was keine Identifikation zulässt [4]. Aus diesen Gründen wurde bei den weiteren Versuchen in dieser Arbeit von der AmazonBasics Tastatur abgesehen.

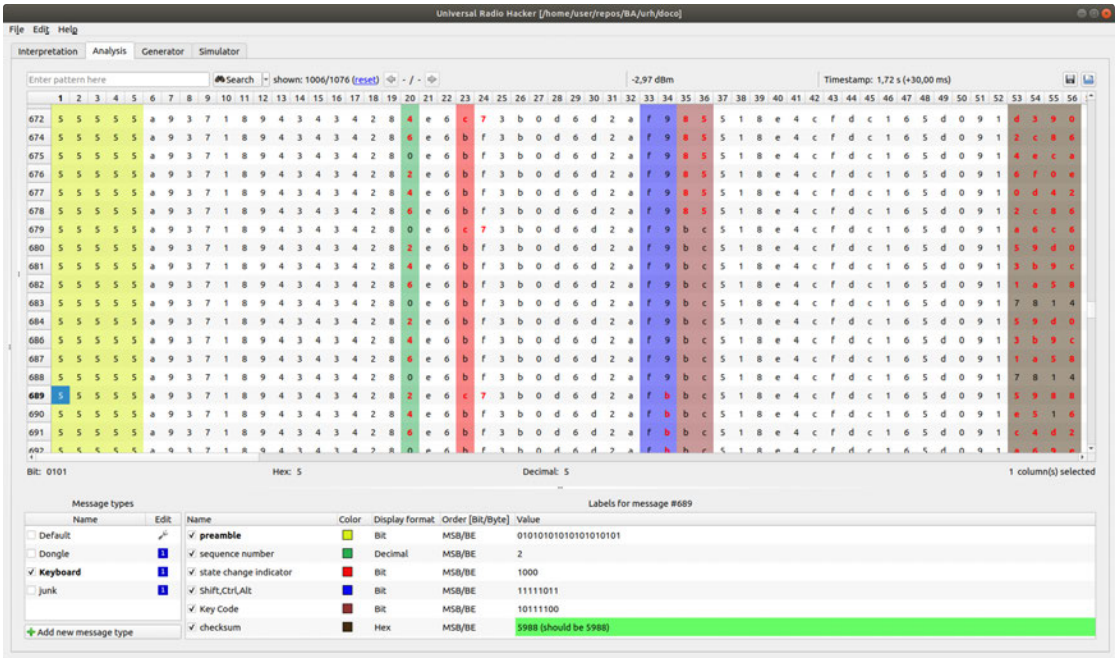

Abbildung 4.13: URH *Analysis* Tab mit Paketen von der Docooler Tastatur

Korrelation

Um Informationen aus den dekodierten Paketen zu gewinnen, wurde versucht, die Paketinhalte mit Tastendrücken zu korrelieren.

Ein Zusammenhang zum Absender (Tastatur oder Dongle) wurde zuerst untersucht. Dies war bei allen Tastaturen über die Paketlänge möglich. Wenn die Tastaturen ruhen, sind nur kurze Pakete in den Aufzeichnungen sichtbar. Dies ist vermutlich eine Art Heartbeat.

Wenn Tasten gedrückt werden, sind weit längere Pakete in den Aufzeichnungen zu sehen. Bei ESB geben 6 bits des Pakets die Länge der Nutzdaten an. Auch bei Docooler und Sonkir K-20 korrelieren die Länge und gewisse Bits im vorderen Teil der Pakete. Ob es sich hierbei um Längen- oder Adressbits handelt, ist dabei irrelevant, da sie nur der Identifikation des Absenders dienen.

Basierend auf diesen Merkmalen wurden Nachrichtentypen und -filter angelegt. Mit diesen wurde der Fokus auf die von den Tastaturen gesendeten Pakete gerichtet.

In Abbildung 4.13 sind Pakete der Docooler Tastatur zu sehen. Die Pakete sind von oben nach unten in chronologischer Reihenfolge geordnet. Sie sind so dekodiert, dass die Prüfsumme an Bytegrenzen ausgerichtet ist. Es werden nur Pakete gezeigt, die von der Tastatur gesendet wurden. Ein häufig auftretendes Paket wurde als Referenz gewählt. Dieses ist wohl ein Heartbeat von der Tastatur. Unterschiede zu der Referenz werden mit roten Zeichen dargestellt. So sind Wertwechsel einfach erkennbar.

Bitfelder

Die eingefärbten Hintergründe in Abbildung 4.13 markieren Bitfelder. URH hat eine Funktion zum automatischen Erkennen von Bitfeldern. Diese erwies sich jedoch als unzuverlässig. Daher erfolgt bei den meisten Tastaturen die Erkennung von Bitfeldern manuell anhand der Wertwechsel und ihrer Korrelation zu Tastendrücken. Die Bitfelder zu der Logitech K270 wurden wie in Abbildung 4.12 dargestellt angelegt.

Die zugewiesene Bedeutung der Bitfelder erscheint in einer Tabelle unter den Paketen. Hier werden auch die Werte der Bitfelder zu einem ausgewählten Paket dargestellt und die CRC Summe überprüft. Dies hilft bei der Erkennung fehlerhafter Aufzeichnungen.

4.4.2 Ergebnisse

Sowohl bei der Docooler als auch bei der Sonkir K-20 war ein eindeutiger Zusammenhang zwischen der gedrückten Taste und dem Wert eines Bitfeldes in dem gesendeten Paket ersichtlich.

Docooler und Sonkir K-20

Die Sonkir K-20 Tastatur überträgt den USB HID Keyboard Code [39] der gedrückten Taste in Klartext. Bei der Docooler werden der Keyboard Code und der Wert `0xbc` mit eine XOR verrechnet. In beiden Fällen gibt es ein Bitfeld, welches den Zustand von Tasten wie Ctrl, Alt, und Shift beinhaltet.

Logitech K270

Bei dem Logitech K270 wurde das gleiche Verschlüsselungsverfahren bei "KeyKeriki v2.0" festgestellt [37]. Die Pakete enthalten 64 Bit verschlüsselte Daten, einen 32 Bit Counter und eine weitere Checksumme. Der Counter wird mit jedem Paket inkrementiert. Auch bei wiederholtem Drücken der gleichen Taste unterscheiden sich die verschlüsselten Daten komplett.

5 Fazit und Ausblick

Wie in Tabelle 5.1 zu sehen ist, konnte bei allen untersuchten Tastaturen mittels einfacher Replayangriffe die Anfälligkeit gegenüber Injection nachgewiesen werden. Hierbei wird die Stärke von SDRs genutzt, Signale ohne weiteres Wissen über Modulation oder Kodierung generieren zu könnnen. Bemerkenswert ist, dass ein Replayangriff bei allen Tastaturen möglich ist, obwohl in allen Protokollen Sequenznummern und in manchen sogar Verschlüsselung verwendet werden. Diese Beobachtung lässt darauf schließen, dass beim Empfang keine Überprüfung stattfindet, ob eine Sequenznummer mehrfach verwendet wird.

Docooler und Sonkir SK-20 nutzen keine Verschlüsselung, bzw. einen konstanten Schlüssel, der einfach zu ermitteln ist. Hierbei liegt auch kein Schutz gegen Sniffing vor.

Tastatur	**Anfälligkeit für Injection**	**Anfälligkeit für Sniffing**
Docooler	nachgewiesen	nachgewiesen
Sonkir SK-20	nachgewiesen	nachgewiesen
Logitech K270	nachgewiesen	**nicht** nachgewiesen
AmazonBasics	nachgewiesen	**nicht** nachgewiesen

Tabelle 5.1: Übersicht der nachgewiesenen Anfälligkeiten aller Tastaturen gegenüber Injection und Sniffing.

Generell kann von dem Einsatz von Funktastaturen in sicherheitskritischen Situationen nur abgeraten werden. Jedoch sollte auch bei kabelgebundenen Geräten im Zusammenhang mit SDRs auf die Gefahren von TEMPEST hingewiesen werden [41, 7]. Hierbei können mithilfe von SDRs aus den Abstrahlungen schlecht geschirmter Datenleitungen die übertragenen Daten rekonstruiert werden.

Die Forscher von KeyKeriki v2.0 vermuten, dass Logitech Tastaturen AES im Counter Mode (CTR)[40] verwenden. Hierbei wird angenommen, dass Dongle und Tastatur sich einen privaten Schlüssel teilen. Dabei wird mit einer fortlaufenden Sequenznummer und dem Schlüssel ein Strom pseudozufälliger Werte erzeugt. Ver- und Entschlüsselung

bestehen aus der XOR-Operation mit diesen Werten. Verschlüsselte Daten und Sequenznummer werden in einem Paket übertragen.

In weiteren Forschungen könnten mithilfe der weiteren Funktion von URH einzelne Protokolle umfassender analysiert werden.

So könnte beispielsweise mittels des URH Fuzzer die Möglichkeit eines Bit-Flipping-Angriffs untersucht werden. Wird AES-CTR o.ä. verwendet, müsste ein Bit-Flip in den verschlüsselten Daten zu einem Bit-Flip im Klartext führen. Die Injection beliebiger Tastendrücke wäre somit möglich.

URH hat auch die Fähigkeit, Verbindungen zu simulieren. Das könnte Möglichkeiten der Manipulation des Schlüsselaustausches oder der Sequenznummer aufzeigen. Wenn das Setzen der Sequenznummer durch einen Angriff möglich ist, kann die Verwendung der immer wieder gleichen pseudozufälligen Werte erzwungen werden. Dies würde Sniffing- und Injection-Angriffe erleichtern.

Eine Integration des CRC-Werkzeugs in URH wäre auch denkbar, um die Analyse von Symbolen zu verbessern. Eventuell könnte so auch die Trefferrate der automatischen Protokollerkennung verbessert werden. URH könnte auch von einem pcap-Plugin profitieren, wodurch die Verwendung von etablierten Tools wie Wireshark ermöglicht wäre.

Weitere Angriffsvektoren könnten durch Analyse von Gerätefirmwares gefunden werden. Hierzu muss diese erst aus den Geräten ausgelesen werden. Oftmals wird dies durch einen Ausleseschutz verhindert, welcher erst überwunden werden muss [38].

Es könnte auch versucht werden, Geräte zum Senden zu nutzen, die nicht dafür gedacht sind. So haben beispielsweise VGA-Grafikkarten oftmals eine Abtastrate, die im Radiofrequenzbereich liegt. Es wurde bereits gezeigt, dass VGA-Hardware als SDR-Sender nutzbar ist [18, 23]. Durch fehlende Aliasingfilter könnten hierbei auch ohne LO höhrere Frequenzen erreicht werden. Interessant wäre beispielsweise ein Konzeptnachweis für die Überbrückung einer Air Gap. In diesem Szenario könnte die Maschine hinter der Air Gap einen Tastaturempfängerdongle angeschlossen haben und in der nähe einer infizierten Maschine mit schlecht geschirmten Kabel stehen. Dieser Angriff ist von Interesse, da keine physische Nähe der angreifenden Person zur Zielmaschine mehr notwendig wäre.

Literaturverzeichnis

[1] BNetzA: *Entwurf zum FREQUENZPLAN*. 5 2019. – URL https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Frequenzen/20190705_Frequenzplan_EntwurfStandMai.pdf. – Zugriffsdatum: 2021-01-24

[2] BSI: *Startseite - Cyberfibel*. 2021. – URL https://www.cyberfibel.de/. – Zugriffsdatum: 2021-01-24

[3] ITU: *Radio RegulationsArticles*. 2012. – URL http://search.itu.int/history/HistoryDigitalCollectionDocLibrary/1.41.48.en.101.pdf. – Zugriffsdatum: 2021-01-24

[4] Administrator: *KB33 AmazonBasics Wireless Keyboard Teardown Internal Photos*. 2 2018. – URL https://fccid.io/PRDKB33/Internal-Photos/Internal-Photos-3738319. – Zugriffsdatum: 2021-01-24

[5] Atkin, Justin: *My Other Camera is in Space | GOES-15,16,17 and Himawari 8 HRIT*. 9 2018. – URL https://www.youtube.com/watch?v=jGWFg7EDnyY. – Zugriffsdatum: 2021-01-24

[6] Bertin IT ; CNIM Group.: *BadUSB, an unpatchable flaw?* 11 2014. – URL https://www.infosecurityeurope.com/__novadocuments/81050. – Zugriffsdatum: 2021-01-24

[7] Borchers, Detlef: *Hacking at Random: CCC demonstriert TEMPEST-Messung bei Wahlcomputern*. 8 2009. – URL https://www.heise.de/security/meldung/Hacking-at-Random-CCC-demonstriert-TEMPEST-Messung-bei-Wahlcomputern-751445.html. – Zugriffsdatum: 2021-01-24

[8] Brannon, Brad: *Sampled Systems and the Effects of Clock Phase Noise and Jitter*. 2004. – URL https://www.analog.com/media/en/technical-documentation/application-notes/AN-756.pdf. – Zugriffsdatum: 2021-01-24

[9] Bund: *Telekommunikationsgesetz Teil 5 § 54 Frequenznutzung*. 2021. – URL https://dejure.org/gesetze/TKG/54.html. – Zugriffsdatum: 2021-01-24

[10] Bundesverwaltungsgericht: *Entscheidungen über die Vergabe von Frequenzen für 5 G im Wege der Versteigerung sind rechtmäßig*. 6 2020. – URL https://www.bverwg.de/de/pm/2020/38. – Zugriffsdatum: 2021-01-24

[11] Böck, Hanno: *Patientendaten von Rettungsdiensten ungeschützt im Internet*. 7 2018. – URL https://www.golem.de/news/behoerdenfunk-patientendaten-von-rettungsdiensten-ungeschuetzt-im-internet-1807-135622.html. – Zugriffsdatum: 2021-01-24

[12] Chung, Lori: *YR0042 2.4GHz Cordless Keyboard Teardown Internal Photos EUT Photo - for FCC 15C Logitech Far East*. 11 2013. – URL https://fccid.io/JNZYR0042/Internal-Photos/Internal-Photos-2121382. – Zugriffsdatum: 2021-01-24

[13] CurlyMo ; pilino1234 ; wo_rasp: *pilight ATTiny pre-filter*. 2019. – URL https://manual.pilight.org/electronics/wiring.html#attiny-pre-filter. – Zugriffsdatum: 2021-01-24

[14] Hartl, Michael: *The Tau Manifesto*. 2020. – URL https://tauday.com/tau-manifesto. – Zugriffsdatum: 2021-01-24

[15] Lichtman, Dr. M.: *Digital Modulation*. 2020. – URL https://pysdr.org/content/digital_modulation.html. – Zugriffsdatum: 2021-01-24

[16] Lichtman, Dr. M.: *IQ Sampling*. 2020. – URL https://pysdr.org/content/sampling.html. – Zugriffsdatum: 2021-01-24

[17] Lichtman, Dr. M.: *PySDR: A Guide to SDR and DSP using Python*. 2020. – URL https://pysdr.org/index.html. – Zugriffsdatum: 2021-01-24

[18] multiple: *siro*. 11 2013. – URL https://wiki.das-labor.org/w/VGAtoBaseband. – Zugriffsdatum: 2021-01-24

[19] MULTIPLE: *Polyphase Channelizer*. 9 2019. – URL https://wiki.gnuradio.org/index.php/Polyphase_Channelizer. – Zugriffsdatum: 2021-01-24

[20] MULTIPLE: *rad1o Badge Wiki*. 1 2019. – URL https://rad1o.badge.events.ccc.de/. – Zugriffsdatum: 2021-01-24

[21] MULTIPLE: *Clock Recovery MM*. 12 2020. – URL https://wiki.gnuradio.org/index.php/Clock_Recovery_MM. – Zugriffsdatum: 2021-01-24

[22] MULTIPLE: *Single Pole IIR Filter*. 3 2020. – URL https://wiki.gnuradio.org/index.php/Single_Pole_IIR_Filter. – Zugriffsdatum: 2021-01-24

[23] MULTIPLE: *VGAtoIQBaseband*. 6 2020. – URL https://osmocom.org/projects/osmo-fl2k/wiki/Osmo-fl2k. – Zugriffsdatum: 2021-01-24

[24] MULTIPLE: *Summary Of Features For Some SDRs*. 1 2021. – URL https://wiki.gnuradio.org/index.php/Hardware#Summary_Of_Features_For_Some_SDRs. – Zugriffsdatum: 2021-01-24

[25] NEUMANN, Linus: *Hirne Hacken - Menschliche Faktoren der IT-Sicherheit*. 12 2019. – URL https://media.ccc.de/v/36c3-11175-hirne_hacken#t=760. – Zugriffsdatum: 2021-01-24

[26] NEUMAYR, Benedikt: *File:16-QAM Demonstration.gif*. 4 2017. – URL https://commons.wikimedia.org/wiki/File:16-QAM_Demonstration.gif. – Zugriffsdatum: 2021-01-24

[27] NEWLIN, Marc: *MouseJack - Injecting Keystrokes into Wireless Mice*. 2 2016. – URL https://github.com/BastilleResearch/mousejack/blob/master/doc/pdf/MouseJack-whitepaper-v1.1.pdf. – Zugriffsdatum: 2021-01-24

[28] NOHL, Karsten ; KRISSLER, Sascha ; JAKOBLELL: *BadUSB - On accessories that turn evil*. 8 2014. – URL https://srlabs.de/wp-content/uploads/2014/11/SRLabs-BadUSB-Pacsec-v2.pdf. – Zugriffsdatum: 2021-01-24

[29] NORDIC SEMICONDUCTOR: *nRF24LU1 Product Specification v1.1*. 2008. – URL https://www.mouser.com/datasheet/2/297/Product_Specification_nRF24LU1_v1_1-4717.pdf. – Zugriffsdatum: 2021-01-24

[30] Nordic Semiconductor: *nRF Connect SDK - Protocols*. 1 2021. – URL https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/protocols.html. – Zugriffsdatum: 2021-01-24

[31] Ossmann, Michael: *Software Defined Radio with HackRF*. 2014. – URL https://greatscottgadgets.com/sdr/. – Zugriffsdatum: 2021-01-24

[32] Ossmann, Michael: *Software Defined Radio with HackRF, Lesson 10 - Filters*. 2014. – URL https://greatscottgadgets.com/sdr/10/. – Zugriffsdatum: 2021-01-24

[33] Ossmann, Michael: *GRCon16 - Whole Packet Clock Recovery*. 10 2016. – URL https://www.youtube.com/watch?v=rQkBDMeODHc. – Zugriffsdatum: 2021-01-24

[34] Ossmann, Michael: *clock-recovery - clock recovery experiments*. 07 2017. – URL https://github.com/mossmann/clock-recovery. – Zugriffsdatum: 2021-01-24

[35] Rondeau, Tom: *PFB Channelizers and Synthesizers*. 1 2014. – URL http://www.trondeau.com/examples/2014/1/23/pfb-channelizers-and-synthesizers.html. – Zugriffsdatum: 2021-01-24

[36] Ryan, Mike: *Project Ubertooth User Commands*. 7 2018. – URL https://github.com/greatscottgadgets/ubertooth/blob/master/host/doc/ubertooth-btle.md. – Zugriffsdatum: 2021-01-24

[37] Schroeder, Thorsten ; Moser, Max: *KeyKeriki v2.0 – 2.4GHz*. 11 2010. – URL http://www.remote-exploit.org/articles/keykeriki_v2_0__8211_2_4ghz/. – Zugriffsdatum: 2021-01-24

[38] Scott, Micah E.: *Glitchy Descriptor Firmware Grab - scanlime:015*. 10 2016. – URL https://www.youtube.com/watch?v=TeCQatNcF20. – Zugriffsdatum: 2021-01-24

[39] USB Implementers' Forum: *Universal Serial Bus (USB) HID Usage Tables*. 10 2004. – URL https://www.usb.org/sites/default/files/documents/hut1_12v2.pdf. – Zugriffsdatum: 2021-01-24

[40] Wikipedia: *Counter Mode*. – URL https://de.wikipedia.org/wiki/Counter_Mode. – Zugriffsdatum: 2021-01-24

[41] Williams, Elliot: *TEMPEST: A Tin Foil Hat For Your Electronics And Their Secrets*. 10 2015. – URL https://hackaday.com/2015/10/19/tempest-a-tin-foil-hat-for-your-electronics-and-their-secrets/. – Zugriffsdatum: 2021-01-24

A Tastaturdaten

A.1 Docooler

- M/N: MG-A3
- VID:PID: 4037:2800
- FCC ID: n/a
- S/N: n/a
- Input
  - an/aus Schalter
  - QWERTY
  - 67 Tasten
  - Touch Pad
- Output
  - Power LED
  - Funk Status LED
- 8 Kanäle (GHz)
  - 2,406
  - 2,414
  - 2,422
  - 2,438

  - 2,446
  - 2,454
  - 2,462
  - 2,470
- Symbolrate: 500 kBaud
- CRC
  - width 16
  - poly 0x1021
  - init 0x753f
  - refin false
  - refout false
  - xorout 0x0000
  - check 0x72ca
  - residue 0x0000

Listing A.1: Ausgabe von `dmesg` beim Anschließen des Docooler Dongles

```
[ 300.153218] usb 1-1.2: new full-speed USB device number 3
   ↪ using ehci-pci
[ 300.234897] usb 1-1.2: New USB device found, idVendor=4037,
   ↪ idProduct=2800
[ 300.234899] usb 1-1.2: New USB device strings: Mfr=0, Product
   ↪ =1, SerialNumber=0
[ 300.234900] usb 1-1.2: Product: 2.4G Composite Devic
[ 300.237826] input: 2.4G Composite Devic as /devices/pci0000
   ↪ :00/0000:00:1a.0/usb1
   ↪ /1-1/1-1.2/1-1.2:1.0/0003:4037:2800.0003/input/input17
[ 300.289573] hid-generic 0003:4037:2800.0003: input,hidraw2:
   ↪ USB HID v1.10 Keyboard [2.4G Composite Devic] on usb
   ↪ -0000:00:1a.0-1.2/input0
```

```
[ 300.292384] input: 2.4G Composite Devic as /devices/pci0000
   ↪ :00/0000:00:1a.0/usb1
   ↪ /1-1/1-1.2/1-1.2:1.1/0003:4037:2800.0004/input/input18
[ 300.344584] hid-generic 0003:4037:2800.0004: input,hidraw3:
   ↪ USB HID v1.10 Mouse [2.4G Composite Devic] on usb
   ↪ -0000:00:1a.0-1.2/input1
```

Listing A.2: Ausgabe von `lsusb` zum Docooler Dongle

```
Bus 001 Device 005: ID 4037:2800
```

Listing A.3: Ausgabe von `usb-devices` zum Docooler Dongle

```
T: Bus=01 Lev=02 Prnt=02 Port=01 Cnt=01 Dev#= 5 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=4037 ProdID=2800 Rev=02.00
S: Product=2.4G Composite Devic
C: #Ifs= 2 Cfg#= 1 Atr=a0 MxPwr=100mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=03(HID ) Sub=01 Prot=01 Driver=
   ↪ usbhid
I: If#= 1 Alt= 0 #EPs= 1 Cls=03(HID ) Sub=01 Prot=02 Driver=
   ↪ usbhid
```

A.2 Sonkir K-20

- M/N: SK-11-GM
- VID:PID: 248a:8367
- FCC ID: n/a
- S/N: n/a
- Input
  - an/aus Schalter
  - QWERTZ

  - 107 Tasten
  - extra Maus
- Output
  - Caps Lock LED
  - Num Lock LED
- 8 Kanäle (GHz)
  - 2,405
  - 2,413
  - 2,422
  - 2,430
  - 2,440
  - 2,450
  - 2,460
  - 2,470
- : Symbolrate: 2 MBaud

Listing A.4: Ausgabe von `dmesg` beim Anschließen des Sonkir K-20 Dongles

```
[ 735.854525] usb 1-1.2: new full-speed USB device number 5
   ↪ using ehci-pci
[ 735.940234] usb 1-1.2: New USB device found, idVendor=248a,
   ↪ idProduct=8367
[ 735.940239] usb 1-1.2: New USB device strings: Mfr=1, Product
   ↪ =2, SerialNumber=0
[ 735.940241] usb 1-1.2: Product: Wireless Receiver
[ 735.940243] usb 1-1.2: Manufacturer: Telink
[ 735.945941] input: Telink Wireless Receiver as /devices/
   ↪ pci0000:00/0000:00:1a.0/usb1/1-1/1-1.2/1-1.2:1.0/0003:248A
   ↪ :8367.000B/input/input23
```

```
[ 735.997936] hid-generic 0003:248A:8367.000B: input,hidraw2:
   ↪ USB HID v1.11 Mouse [Telink Wireless Receiver] on usb
   ↪ -0000:00:1a.0-1.2/input0
[ 736.000983] input: Telink Wireless Receiver as /devices/
   ↪ pci0000:00/0000:00:1a.0/usb1/1-1/1-1.2/1-1.2:1.1/0003:248A
   ↪ :8367.000C/input/input24
[ 736.053285] hid-generic 0003:248A:8367.000C: input,hidraw3:
   ↪ USB HID v1.11 Keyboard [Telink Wireless Receiver] on usb
   ↪ -0000:00:1a.0-1.2/input1
```

Listing A.5: Ausgabe von `lsusb` zum Sonkir K-20 Dongle

```
Bus 001 Device 007: ID 248a:8367
```

Listing A.6: Ausgabe von `usb-devices` zum Sonkir K-20 Dongle

```
T: Bus=01 Lev=02 Prnt=02 Port=01 Cnt=01 Dev#= 7 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=248a ProdID=8367 Rev=01.00
S: Manufacturer=Telink
S: Product=Wireless Receiver
C: #Ifs= 2 Cfg#= 1 Atr=a0 MxPwr=50mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=03(HID ) Sub=01 Prot=02 Driver=
   ↪ usbhid
I: If#= 1 Alt= 0 #EPs= 1 Cls=03(HID ) Sub=01 Prot=01 Driver=
   ↪ usbhid
```

A.3 Logitech K270

- M/N: Y-R0042
- VID:PID: 046d:c534
- FCC ID: JNZYR0042
- S/N: 1950SY03M0K9
- Input

  - an/aus Schalter
  - QWERTZ
  - 112 Tasten
  - extra Maus
- Output
  - Caps Lock LED
- 12 Kanäle (GHz)
  - 2,405
  - 2,408
  - 2,411
  - 2,414
  - 2,432
  - 2,435
  - 2,441
  - 2,444
  - 2,462
  - 2,465
  - 2,471
  - 2,474
- : Symbolrate: 2 MBaud

Listing A.7: Ausgabe von `dmesg` beim Anschließen des Logitech K270 Dongles

```
[ 983.662126] usb 1-1.2: new full-speed USB device number 6
   ↪ using ehci-pci
[ 983.745705] usb 1-1.2: New USB device found, idVendor=046d,
   ↪ idProduct=c534
[ 983.745710] usb 1-1.2: New USB device strings: Mfr=1, Product
   ↪ =2, SerialNumber=0
[ 983.745713] usb 1-1.2: Product: USB Receiver
[ 983.745715] usb 1-1.2: Manufacturer: Logitech
[ 983.748538] input: Logitech USB Receiver as /devices/pci0000
   ↪ :00/0000:00:1a.0/usb1/1-1/1-1.2/1-1.2:1.0/0003:046D:C534
   ↪ .0009/input/input23
[ 983.800441] hid-generic 0003:046D:C534.0009: input,hidraw2:
   ↪ USB HID v1.11 Keyboard [Logitech USB Receiver] on usb
   ↪ -0000:00:1a.0-1.2/input0
[ 983.803668] input: Logitech USB Receiver as /devices/pci0000
   ↪ :00/0000:00:1a.0/usb1/1-1/1-1.2/1-1.2:1.1/0003:046D:C534
   ↪ .000A/input/input24
[ 983.855286] hid-generic 0003:046D:C534.000A: input,hiddev1,
   ↪ hidraw3: USB HID v1.11 Mouse [Logitech USB Receiver] on
   ↪ usb-0000:00:1a.0-1.2/input1
```

Listing A.8: Ausgabe von `lsusb` zum Logitech K270 Dongle

```
Bus 001 Device 006: ID 046d:c534 Logitech, Inc. Unifying
   ↪ Receiver
```

Listing A.9: Ausgabe von `usb-devices` zum Logitech K270 Dongle

```
T: Bus=01 Lev=02 Prnt=02 Port=01 Cnt=01 Dev#= 6 Spd=12 MxCh= 0
D: Ver= 2.00 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=046d ProdID=c534 Rev=29.01
S: Manufacturer=Logitech
S: Product=USB Receiver
C: #Ifs= 2 Cfg#= 1 Atr=a0 MxPwr=98mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=03(HID ) Sub=01 Prot=01 Driver=
   ↪ usbhid
```

```
I: If#= 1 Alt= 0 #EPs= 1 Cls=03(HID ) Sub=01 Prot=02 Driver=
   ↪ usbhid
```

A.4 AmazonBasics

- beworben mit 128-Bit-AES-Verschlüsselung
- M/N: KS1-DE
- VID:PID: 3938:1059
- FCC ID: PRDKB33
- S/N: L19A00701
- Input
  - QWERTZ
  - 105 Tasten
- Output
  - Caps Lock LED
  - Num Lock LED
- Sendefrequenzen
  - Dongle: 2,434 GHz
  - Tastatur: 2,466 GHz
- : Symbolrate: 1 MBaud

Listing A.10: Ausgabe von `dmesg` beim Anschließen des AmazonBasics Dongles

```
[ 1269.016221] usb 1-1.2: new full-speed USB device number 9
   ↪ using ehci-pci
[ 1269.097007] usb 1-1.2: New USB device found, idVendor=3938,
   ↪ idProduct=1059
```

```
[ 1269.097009] usb 1-1.2: New USB device strings: Mfr=1,
   ↪ Product=2, SerialNumber=0
[ 1269.097010] usb 1-1.2: Product: 2.4G RF Keyboard & Mouse
[ 1269.097011] usb 1-1.2: Manufacturer: MOSART Semi.
[ 1269.098853] input: MOSART Semi. 2.4G RF Keyboard & Mouse as
   ↪ /devices/pci0000:00/0000:00:1a.0/usb1
   ↪ /1-1/1-1.2/1-1.2:1.0/0003:3938:1059.000F/input/input29
[ 1269.150530] hid-generic 0003:3938:1059.000F: input,hidraw2:
   ↪ USB HID v1.10 Keyboard [MOSART Semi. 2.4G RF Keyboard &
   ↪ Mouse] on usb-0000:00:1a.0-1.2/input0
[ 1269.152694] input: MOSART Semi. 2.4G RF Keyboard & Mouse as
   ↪ /devices/pci0000:00/0000:00:1a.0/usb1
   ↪ /1-1/1-1.2/1-1.2:1.1/0003:3938:1059.0010/input/input30
[ 1269.204541] hid-generic 0003:3938:1059.0010: input,hiddev1,
   ↪ hidraw3: USB HID v1.10 Mouse [MOSART Semi. 2.4G RF
   ↪ Keyboard & Mouse] on usb-0000:00:1a.0-1.2/input1
```

Listing A.11: Ausgabe von `lsusb` zum AmazonBasics Dongle

```
Bus 001 Device 009: ID 3938:1059
```

Listing A.12: Ausgabe von `usb-devices` zum AmazonBasics Dongle

```
T: Bus=01 Lev=02 Prnt=02 Port=01 Cnt=01 Dev#= 9 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=3938 ProdID=1059 Rev=01.03
S: Manufacturer=MOSART Semi.
S: Product=2.4G RF Keyboard & Mouse
C: #Ifs= 2 Cfg#= 1 Atr=a0 MxPwr=100mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=03(HID ) Sub=01 Prot=01 Driver=
   ↪ usbhid
I: If#= 1 Alt= 0 #EPs= 1 Cls=03(HID ) Sub=01 Prot=02 Driver=
   ↪ usbhid
```

B Installation

Listing B.1: Installation von Entwicklertools aus Paketquellen

```
sudo apt-get update
sudo apt-get install build-essential git libusb-1.0-0-dev cmake
   ↪ dkms python-cheetah python
```

B.1 xtrx

XTRX hat ein Repository namens *images* auf GitHub. Dieses beinhaltet die Quellen für die notwendigen Libraries als Subrepositories. In Listing B.2 werden die Quellen bezogen, ein Patch angewandt, kompiliert und installiert. Der Patch war auf Grund einer Versionskollision mit Python notwendig.

Listing B.2: Installation von gr-osmosdr mit XTRX support

```
git clone https://github.com/xtrx-sdr/images.git xtrx
cd xtrx
git checkout 8c20ce1
git submodule init
git submodule update
mkdir sources/build
cd sources/liblms7002m
# apply patch
git apply << EOF
diff --git a/CMakeLists.txt b/CMakeLists.txt
index 8822807..0d587ec 100644
--- a/CMakeLists.txt
+++ b/CMakeLists.txt
```

```
@@ -41,7 +41,7 @@ set(CROSS_COMPILE_LIB_PATH "/usr/\${CC_ARCH}/
   ↪ lib")
 set(LIBLMS7002M_LIBRARY_DIR lib\${LIB_SUFFIX})
 set(LIBLMS7002M_INCLUDE_DIR include)

-find_package(PythonInterp 3.4 REQUIRED)
+find_package(PythonInterp 2.7 REQUIRED)
 message(STATUS "Python found at \${PYTHON_EXECUTABLE}")
 execute_process(COMMAND \${PYTHON_EXECUTABLE} -c "from Cheetah
    ↪ .Template import Template" RESULT_VARIABLE retcode)
 if(NOT "\${retcode}" STREQUAL "0")
EOF
cd ../build
cmake ..
make -j$(nproc)
sudo make install
```

B.2 gr-osmosdr

Bei der Installation von gr-osmosdr musste wegen einer API-Änderung ebenfalls ein Patch gefunden und angewandt werden.

Listing B.3: Installation von gr-osmosdr mit XTRX support

```
git clone https://github.com/xtrx-sdr/gr-osmosdr
cd gr-osmosdr
git checkout 778cf41
# apply patch
git apply << EOF
diff --git a/lib/xtrx/xtrx_obj.cc b/lib/xtrx/xtrx_obj.cc
index 5c73259..1d58de5 100644
--- a/lib/xtrx/xtrx_obj.cc
+++ b/lib/xtrx/xtrx_obj.cc
@@ -68,7 +68,7 @@ xtrx_obj::xtrx_obj(const std::string &path,
   ↪ unsigned loglevel, bool lmsreset)
```

```
  unsigned xtrxflag = (loglevel & XTRX_O_LOGLVL_MASK) | ((
     ↪ lmsreset) ? XTRX_O_RESET : 0);
  std::cerr << "xtrx_obj::xtrx_obj␣=␣" << xtrxflag << std::endl
     ↪ ;

- int res = xtrx_open_string(path.c_str(), &_obj);
+ int res = xtrx_open_list(path.c_str(), NULL, &_obj);
  if (res < 0) {
    std::stringstream message;
    message << "Couldn't␣open␣" ":␣Error:␣" << -res;
EOF
mkdir build
cd build
cmake ..
make -j$(nproc)
sudo make install
```

B.3 SDRangel

Listing B.4: Abschnitt aus der Funktion wpcr

```
git clone https://github.com/xtrx-sdr/sdrangel.git
cd sdrangel
git checkout a45c37ab
mkdir build
cd build
cmake ..
make -j$(nproc)
sudo make install
```

B.4 RevEng

Listing B.5: Abschnitt aus der Funktion wpcr

```
wget https://sourceforge.net/projects/reveng/files/2.1.0/reveng
   ↪ -2.1.0.tar.xz/download
tar xf download
# apply patch
sed -i 's/#define BMP_SUB 16/#define BMP_SUB 32/' reveng-2.1.0/
   ↪ config.h
sed -i 's/#define BMP_BIT 32/#define BMP_BIT 64/' reveng-2.1.0/
   ↪ config.h
make -C reveng-2.1.0
```

Glossar

Dongle Ein Dongle ist ein kleines Stück Computer-Hardware, das in einen Anschluss an einem anderen Gerät eingesteckt wird, um diesem zusätzliche Funktionen zu bieten.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original