HAW HAMBURG

Bachelorarbeit

Mathis Eike Heeren

Implementierung und Bewertung eines Softmodems für die physikalische Schicht des HART-Standards und den Einsatz auf Mikrocontrollern

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Mathis Eike Heeren

Implementierung und Bewertung eines Softmodems für die physikalische Schicht des HART-Standards und den Einsatz auf Mikrocontrollern

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Elektro- und Informationstechnik*
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Paweł Buczek
Zweitgutachter: Dr.-Ing. Stefan Boßung

Eingereicht am: 30. August 2022

Mathis Eike Heeren

Thema der Arbeit

Implementierung und Bewertung eines Softmodems für die physikalische Schicht des HART-Standards und den Einsatz auf Mikrocontrollern

Stichworte

Softmodem, HART, Bell-202, Mikrocontroller, STM32, FSK, BFSK, CPFSK

Kurzzusammenfassung

In dieser Arbeit wird untersucht, ob die Funktionalität eines integrierten Modem-Schaltkreises durch eine Softwarenachbildung auf einem Mikrocontroller übernommen werden kann. Anhand einer Demonstrationsversion wird eine Einschätzung über das Potenzial dieses Konzepts gegeben.

Mathis Eike Heeren

Title of Thesis

Implementation and evaluation of a soft modem for the physical layer of the HART standard and the use on microcontrollers

Keywords

Softmodem, HART, Bell-202, Microcontroller, STM32, FSK, BFSK, CPFSK

Abstract

This thesis investigates if the functionality of an integrated modem circuit can be taken over by a software emulation on a microcontroller. Based on a demonstration version, an estimation of the potential of this concept is given.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ADC Analog-To-Digital-Converter.

APRS Automatic-Packet-Reporting-System.

BER Bit-Error-Rate.

BFSK Binary-Frequency-Shift-Keying.

CD Carrier-Detect.

CLI Command-Line-Interface.

CPFSK Continuous-Phase-Frequency-Shift-Keying.

DAC Digital-To-Analog-Converter.

DMA Direct-Memory-Access.

EMA Exponential-Moving-Average.

FIFO First-In-First-Out.

FSK Frequency-Shift-Keying.

GCC GNU-Compiler-Collection.

GPIO General-Purpose-Input-Output.

HAL Hardware-Abstraction-Layer.

HART Highway-Addressable-Remote-Transducer.

IC Integrated-Circuit.

IDE Integrated-Development-Environment.

IIR Infinite-Impulse-Response.

IRQ Interrupt-Request.

ISO International-Organization-for-Standardization.

ISR Interrupt-Service-Routine.

LSB Least-Significant-Bit.

Modem Modulator-Demodulator.

MSPS Million-Samples-Per-Second.

OPAMP Operational-Amplifier.

OSI Open-System-Interconnection.

RTS Request-To-Send.

RxD Receive-Data.

SAR Successive-Approximation.

SNR Signal-To-Noise-Ratio.

Softmodem Software-Modulator-Demodulator.

STM STMicroelectronics.

TxD Transmit-Data.

UART Universal-Asynchronous-Receiver-Transmitter.

Listings

1 Einführung

Für das Übermitteln von Informationen zwischen zwei Geräten in einem industriellen Umfeld hat sich die 4...20 mA Stromschleife oder auch 4...20 mA Stromschnittstelle durchgesetzt [15]. Zu übermittelnde Informationen werden durch Variation der Stromstärke des in der Stromschleife fließenden Gleichstroms übertragen. Zum Beispiel können die Messwerte eines Sensors durch einen Gleichstrom zwischen 4 und 20 mA repräsentiert werden. Informationen können über eine 4...20 mA Stromschnittstelle nur in eine Richtung übermittelt werden. Der Grund für diese Einschränkung ist, dass nur eines der beiden kommunizierenden Geräte den Gleichstrom beeinflussen und so Informationen senden kann. Das andere Gerät beobachtet dauerhaft den Wert des Gleichstroms und empfängt so die Informationen des sendenden Geräts, siehe Abbildung 1.1.

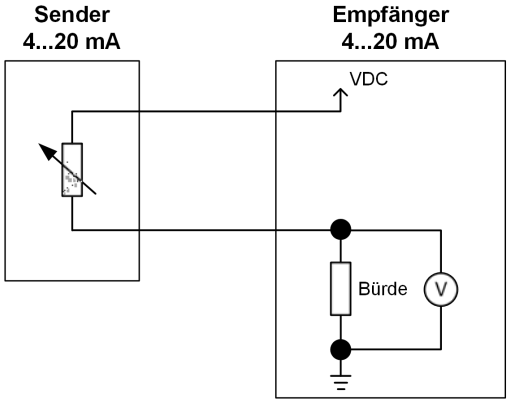

Abbildung 1.1: Funktionsschema einer 4...20 mA Stromschnittstelle

Ein Fehler, wie zum Beispiel ein Kabelbruch in der Leitung, wird in einer 4...20 mA Stromschleife erkannt, da der Strom im Störfall unter den Mindestwert von 4 mA fällt. Um die Übertragung zusätzlicher Informationen sowie die Kommunikation in beide

Richtungen über eine 4...20 mA Stromschnittstelle zu ermöglichen, wird der Highway-Addressable-Remote-Transducer (HART)-Standard verwendet. Die Übertragung des HART-Signals erfolgt mit einer anderen Methode als die Übertragung des Gleichstromsignals der Stromschnittstelle, siehe Kapitel 3. Das HART-Signal zur Übertragung zusätzlicher Daten wird parallel zu dem 4...20 mA Gleichstromsignal der Stromschnittstelle gesendet, ohne diese bedeutend zu stören, siehe Abbildung 1.2.

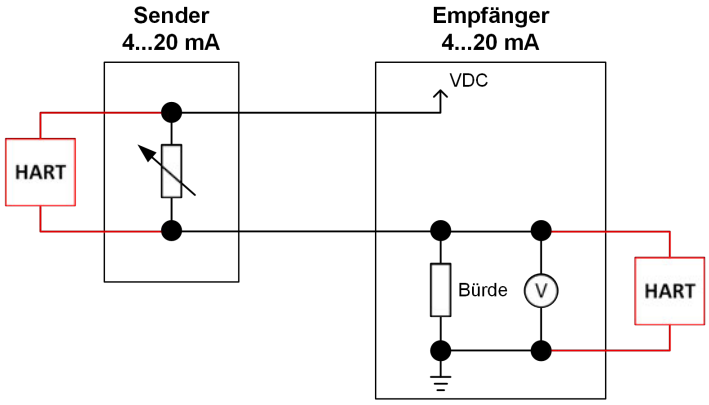

Abbildung 1.2: 4...20 mA Stromschnittstelle mit HART-Erweiterung

Der HART-Standard ermöglicht es, in Geräten viele zusätzliche Funktionen zu implementieren. Es kann zum Beispiel die Unterscheidung zwischen verschiedenen Fehlern eines Geräts oder die Möglichkeit, Geräte über weite Entfernungen zu konfigurieren, realisiert werden.

1.1 Motivation

Um das HART-Signal parallel zu dem Gleichstromsignal der Stromschnittstelle zu übertragen, wird ein Modulator-Demodulator (Modem) benötigt. HART-Modems werden typischerweise in Form von Integrated-Circuits (ICs) wie zum Beispiel dem Low-Power-HART-Modem-IC AD5700/AD5700-1 [3] vertrieben. Die zertifizierten HART-Modem-ICs erhöhen die Kosten für die Herstellung von HART-fähigen Geräten. Der AD5700-1 ist zum Beispiel am 14.08.2022 ab einer Abnahme von 1500 Stück für 6 € bis 8 € pro Stück erhältlich gewesen. Des Weiteren wird für sie zusätzlicher Bauraum auf der Leiterplatte benötigt.

Um den benötigten Hardwareaufwand zu senken, soll die Funktionalität eines HART-Modem-ICs von einem Software-Modulator-Demodulator (Softmodem) übernommen werden. Das Softmodem ist eine Softwarenachbildung des HART-Modem-ICs und wird in dieser Arbeit auf einem Mikrocontroller betrieben. Der Mikrocontroller wurde als Zielplattform für das Softmodem gewählt, da er für die Funktion der meisten elektronischen Geräte benötigt wird und somit bereits in den Geräten vorhanden ist.
Das Ziel dieser Arbeit ist es, zu untersuchen, ob die Funktionalität eines HART-Modem-ICs durch ein HART-Softmodem auf einem Mikrocontroller übernommen werden kann. Diese Arbeit soll darüber hinaus als Grundlage für weitere Untersuchungen dienen und eine Einschätzung über das Potenzial des Konzepts ermöglichen.

2 Zielsetzung

Ziel dieser Arbeit ist zu zeigen, dass ein Softmodem, das auf einem Mikrocontroller betrieben wird, in der Lage ist, die Funktionalität eines HART-Modems zu übernehmen. Anhand einer zu erstellenden Demonstrationsversion des Softmodems wird dessen Eignung als Ersatz des HART-Modem-ICs untersucht.

2.1 Informationsbeschaffung

Zunächst wird untersucht, welche essenziellen Funktionen das HART-Modem-IC [3] für die Durchführung einer HART-Kommunikation bereitstellt. Das Modem-IC wird mit einer Universal-Asynchronous-Receiver-Transmitter (UART)-Schnittstelle über die Signale Request-To-Send (RTS), Carrier-Detect (CD), Transmit-Data (TxD) und Receive-Data (RxD) gesteuert, siehe Abbildung 2.1.

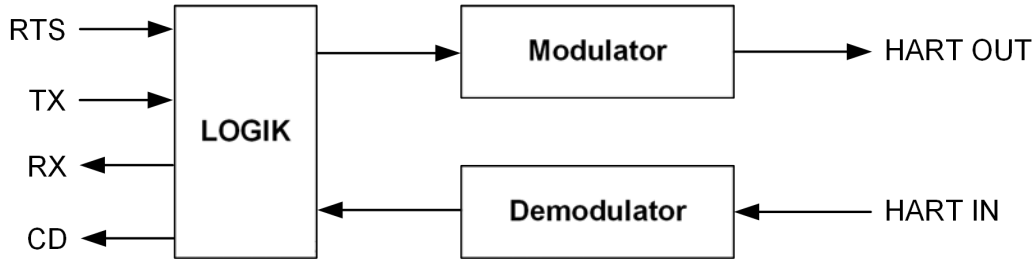

Abbildung 2.1: Blockschaltbild eines HART-Modem-ICs

1. Um eine Kommunikation zu initiieren, muss das RTS-Signal gesetzt werden. Das Modem-IC erzeugt daraufhin ein Trägersignal an dem HART-Ausgang.
2. Über das TxD-Signal können, solange das RTS-Signal gesetzt ist, Bytes im UART-Format (Zeichen) übergeben werden. Das Modem-IC erzeugt ein entsprechend der übergebenen Zeichen moduliertes Signal an seinem HART-Ausgang.

3. Das CD-Signal ist aktiv, wenn ein Trägersignal an dem HART-Eingang anliegt. Mit dem CD-Signal wird vor dem Senden einer Nachricht sichergestellt, dass keine andere Kommunikation aktiv ist.
4. Das Modem-IC demoduliert das empfangene HART-Eingangs-Signal. An dem RxD-Ausgang werden die demodulierten UART-Zeichen ausgegeben.

2.2 Anforderungen

Aus der vorangegangenen Informationsbeschaffung leiten sich die Anforderungen für die Entwicklung des Softmodems ab. Die Anforderungen sind in funktionale Anforderungen (FA) Unterabschnitt 2.2.1 und nicht-funktionale Anforderungen (NFA) Unterabschnitt 2.2.2 unterteilt. Des Weiteren werden die Anforderungen in 3 Stufen priorisiert.
Die genutzte Hardware ist fest vorgegeben und darf nur um wenige passive Bauteile erweitert werden, da das übergeordnete Ziel die Minimierung der Bauteile und Kosten ist.
Auf Grundlage dieser Rahmenbedingungen wird die Entwicklung des Softmodems durchgeführt.
Für die Definition einiger Anforderungen ist ein Vorgriff auf Inhalte des Grundlagenkapitels notwendig.

2.2.1 Funktionale Anforderungen

Muss-Anforderungen:

FA.1.1 In der Simulation wird eine Symbolfolge ohne Phasensprung gesendet. Die Modulations-Methode kann so vor der Implementierung validiert werden.

FA.1.2 In der Simulation wird eine Symbolfolge empfangen. So wird die ausgewählte Demodulations-Methode vor der Implementierung überprüft.

Soll-Anforderungen:

FA.2.1 Das implementierte Softmodem sendet eine Symbolfolge ohne Phasensprung. Durch das Senden ohne Phasensprung werden mögliche Oberwellen vermieden.

FA.2.2 Eine von dem Softmodem gesendete Symbolfolge wird von dem USB-HART-Modem [8] empfangen. So kann überprüft werden, ob das implementierte Softmodem eine Symbolfolge an ein zertifiziertes HART-Modem senden kann.

FA.2.3 Das Softmodem kann empfangene Symbole unterscheiden. Dies ist für das korrekte Empfangen einer Nachricht notwendig.

Wunsch-Anforderungen:

FA.3.1 Daten werden von dem Softmodem entsprechend der Spezifikation der digitalen Datenübertragung aus der physikalischen Schicht [7] gesendet. So ist das Senden ganzer Nachrichten mit dem Softmodem möglich.

FA.3.2 Entsprechend der Spezifikation der digitalen Datenübertragung [7] gesendete Daten werden von dem Softmodem empfangen. Das Empfangen ganzer Nachrichten mit dem Softmodem wird so ermöglicht.

FA.3.3 Eine von einem USB-HART-Modem [8] gesendete Symbolfolge wird von dem Softmodem empfangen. Es wird so gezeigt, dass das Empfangen einer Nachricht eines zertifizierten HART-Modems durch das implementierte Softmodem möglich ist.

2.2.2 Nicht-funktionale Anforderungen

Muss-Anforderungen:

NFA.1.1 Für die Umsetzung wird wenig zusätzliche Hardware-Beschaltung benötigt, da das Ziel die Minimierung der Bauteile und Kosten ist.

NFA.1.2 Die Simulationen werden in der Programmiersprache Python umgesetzt, da Python eine frei zugängliche Programmiersprache ist.

NFA.1.3 Der Programmcode des Softmodems wird in der Programmiersprache C realisiert, da der verwendete Mikrocontroller mit der Programmiersprache C kompatibel ist.

NFA.1.4 Das Softmodem ist für die Verwendung auf der vorgegebenen Hardware geeignet. Dies ist notwendig, da das Softmodem auf dieser Hardware betrieben werden soll.

Soll-Anforderungen:

NFA.2.1 Die Generierung des zu sendenden Signals erfolgt innerhalb der spezifizierten Toleranzen der physikalischen Schicht. Das Softmodem muss die Signale innerhalb der vorgegebenen Toleranzen erzeugen, um eine Zulassung erhalten zu können.

Wunsch-Anforderungen:

NFA.3.1 Das Senden und Empfangen von Symbolfolgen erfolgt über eine 4...20 mA Stromschnittstelle. Die Funktionsfähigkeit des Softmodems in einer realitätsnahen Anwendung wird so bewiesen.

NFA.3.2 Die Lesbarkeit des erstellten Programmcodes ist höher zu priorisieren als Optimierungen des Rechenaufwands. Optimierungen werden erst vorgenommen, wenn sie notwendig sind. »We should forget about small efficiencies, say about 97 % of the time: premature optimization is the root of all evil.« [12]

NFA.3.3 Bei der Implementierung wird die Ausführungszeit höher priorisiert als der Speicherbedarf. Der Grund ist, dass ausreichend Speicherplatz zur Verfügung steht und die Ausführungszeit die knappere Ressource ist.

NFA.3.4 Das Softmodem erfüllt alle Anforderungen der Spezifikation der physikalischen Schicht. Um eine Zulassung als HART-Modem erhalten zu können, muss das Softmodem alle Anforderungen der FSK-Physical-Layer-Specification [7] einhalten.

2.3 Verwendete Hardware

Das Softmodem wird auf einer STM32 Nucleo-144 Entwicklungs-Platine [23] NUCLEO-L552ZE-Q entwickelt. Auf der Nucleo-Platine ist ein Arm Cortex M33 Mikrocontroller des Typen STM32L552ZET6QU verbaut. Um den Energiebedarf des Mikrocontrollers zu senken, wird er auf einer maximalen Taktfrequenz von 8 MHz betrieben.
Die Nucleo-Platine Abbildung 2.2 (gelb) ist mit einer EVAL-AD5421SDZ Entwicklungs-Platine [1] für einen 4 bis 20 mA Digital-To-Analog-Converter (DAC) verbunden. Die 4...20 mA Entwicklungs-Platine (grün) repräsentiert die 4...20 mA Stromschnittstelle des Geräts, in dem das Softmodem verbaut ist. Um die 4...20 mA Entwicklungs-Platine mit einem Computer ansteuern zu können, wird eine zusätzliche Steuerungsplatine EVAL-SDP-CB1Z [2] benötigt (hellblau).
Die 4...20 mA Entwicklungs-Platine ermöglicht es, einen HART-Modem-IC wie in [3, S. 19] anzuschließen. Die Anschlüsse für den HART-Modem-IC werden in diesem Aufbau genutzt, um die Nucleo-Platine an die 4...20 mA Entwicklungs-Platine anzuschließen. So wird der Anschluss des Softmodems an die 4...20 mA Stromschnittstelle eines Geräts realitätsnah nachgebildet und ermöglicht, die Anforderung NFA.3.1 zu testen.

Als Gegenstelle, mit der das Softmodem durch die 4...20 mA Stromschnittstelle kommunizieren soll, wird ein zertifiziertes USB-HART-Modem [8] verwendet. Das USB-HART-Modem (blau) ist über eine zusätzlich auf der 4...20 mA Entwicklungs-Platine angebrachte Bürde (Widerstand) mit der 4...20 mA Stromschnittstelle verbunden (rot).
Der funktionale Zusammenhang der Hardwarekomponenten ist als Blockschaltbild in Abbildung 2.3 veranschaulicht.

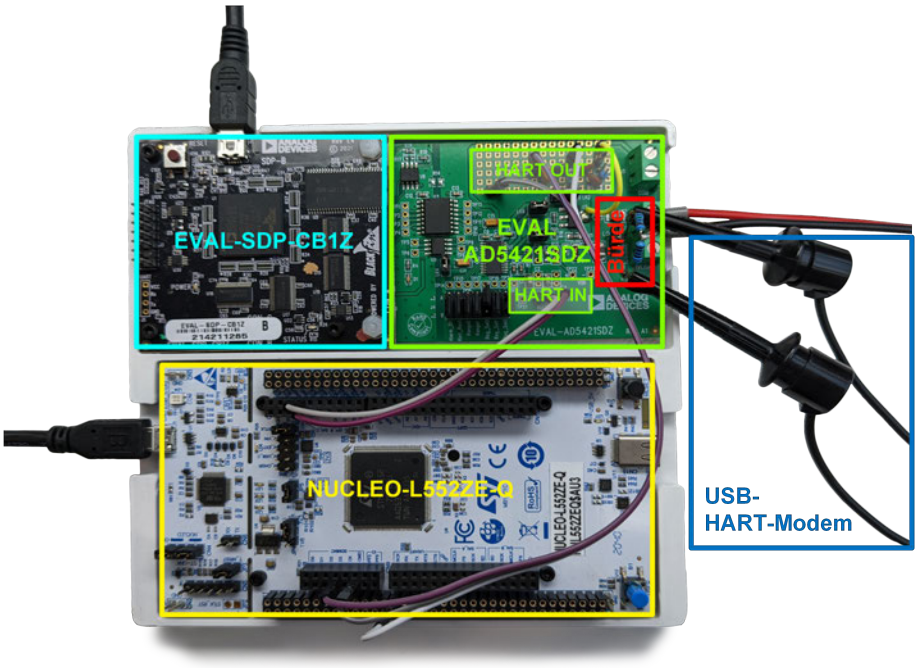

Abbildung 2.2: Hardwareaufbau des Testsystems

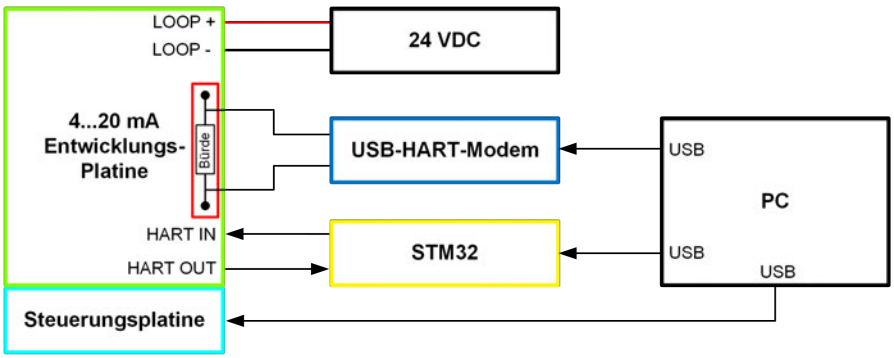

Abbildung 2.3: Blockschaltbild des Testsystems

3 Grundlagen

Um den simultanen Austausch von Informationen zwischen elektronischen Geräten auf einem Übertragungskanal nachvollziehen zu können, ist Kenntnis über das Modell der allgemeinen Nachrichtenübertragung Abschnitt 3.1, sowie über die Modulation Abschnitt 3.2 und Demodulation Abschnitt 3.3 notwendig. Auf die Frequenzumtastung wird in Abschnitt 3.4 genauer eingegangen, da diese Modulationstechnik im Softmodem zur Anwendung kommt. Abschließend werden in Abschnitt 3.5 Definitionen aus dem HART-Standard, welche für die Realisierung des Softmodems notwendig sind, beschrieben. Die in diesem Kapitel beschriebenen Grundlagen und Konzepte für den Entwurf des Softmodems werden in Kapitel 4 angewandt.

3.1 Die allgemeine Nachrichtenübertragung

Bei einer Nachrichtenübertragung werden Nachrichten in Form von Signalen ausgetauscht [25, S. 11]. Ein Signal ist die physikalische Repräsentation der Informationen, die in einer Nachricht enthalten sind [17, S. 903]. In dieser Arbeit werden nur elektronische Signale betrachtet und im Folgenden synonym verwendet.

Das shannonsche Kommunikationsmodell Abbildung 3.1 beschreibt eine allgemeine Nachrichtenübertragung. Die Informationsquelle erzeugt eine Nachricht, die an den Sender weitergegeben wird. Das von dem Sender erzeugte Signal wird über einen Kanal zu dem Empfänger übertragen. Während der Übertragung des Signals durch den Kanal kann das Signal von einer Störquelle beeinflusst werden. Der Empfänger übersetzt das empfangene Signal in eine Nachricht für die Informationssenke. [25, S. 11]

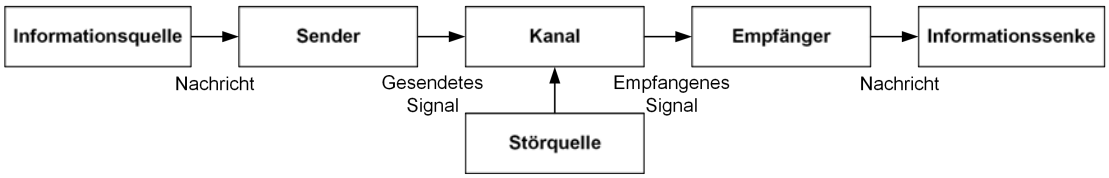

Abbildung 3.1: Nachrichtenübertragung nach dem shannonschen Kommunikationsmodell

In diesem Kapitel wird vereinfachend angenommen, dass die Signalübertragung durch einen idealen Kanal siehe Gleichung 3.1 erfolgt.

$$H(f) = 1 \tag{3.1}$$

Die Übertragungsfunktion eines idealen Kanals ist frequenzunabhängig und führt weder zu einer Dämpfung noch zu einer Verzögerung des Signals. Das von dem Sender erzeugte Signal wird somit unverändert bis zu dem Empfänger übertragen.

3.2 Modulation

Für die in Abschnitt 3.1 beschriebene Umwandlung einer Nachricht in ein Signal durch den Sender kommt in dem Softmodem eine Modulationstechnik zum Einsatz.

Durch die Modulation wird das Signal einer Nachricht aus dem Basisband in einen anderen Frequenzbereich verschoben. Die Verschiebung von Signalen in verschiedene Frequenzbereiche ermöglicht es, mehrere Signale simultan über einen Kanal zu übertragen [17, S. 978].
Da das Basisband des Kanals, in welchem das Softmodem Signale überträgt, bereits durch das Gleichstromsignal der 4...20 mA Stromschnittstelle belegt ist, siehe Kapitel 1, ist eine Modulation des Signals in ein anderes Frequenzband notwendig.

3.3 Demodulation

Der Empfänger des Softmodems muss aus dem von einem Sender modulierten Signal die ursprüngliche Nachricht zurückgewinnen. Dies geschieht unter anderem durch einen Demodulator, der aus dem modulierten Signal das Basisbandsignal zurückgewinnt [17, S. 978].
Bei der Demodulation wird grundsätzlich zwischen kohärenter und inkohärenter Demodulation unterschieden. Eine Demodulation erfolgt kohärent, wenn sie synchron zu dem Trägersignal, das heißt mit gleicher Frequenz und Phase, erfolgt [25, S. 247]. Die kohärente Demodulation ist aufwendiger als die inkohärente Demodulation, da eine zusätzliche Synchronisation auf das Eingangssignal notwendig ist. Der Vorteil der kohärenten Demodulation ist, dass sie durch die zusätzlichen Informationen über das Trägersignal weniger störempfindlich ist [25, S. 247].

3.4 Frequenzumtastung

Durch die Frequenzmodulation wird die Frequenz eines Trägersignals in Abhängigkeit des zu modulierenden Signals verändert.

Die Frequenzumtastung, oder auch Frequency-Shift-Keying (FSK), ist ein Sonderfall der Frequenzmodulation. Sie wird für die Übertragung digitaler Daten genutzt. [17, S. 992]

Bei der Frequenzumtastung wird jedem zu übertragenden Symbol, eine bestimmte Frequenz zugeordnet. Die Frequenz des Trägersignals wird dem zu übertragenden Symbol entsprechend angepasst.
Die binäre Frequenzumtastung, also eine FSK-Modulation mit 2 diskreten Frequenzen, wird unter anderem als 2-FSK [11, S. 204], Binary-Frequency-Shift-Keying (BFSK) [25, S. 271] oder weiterhin als FSK [17, S. 993] bezeichnet.
Das Umschalten zwischen den verschieden diskreten Frequenzen kann mit und ohne Phasensprung durchgeführt werden. Das Umschalten ohne Phasensprung Abbildung 3.2 wird auch als Continuous-Phase-Frequency-Shift-Keying (CPFSK) bezeichnet und führt zu einer Verringerung der benötigten Bandbreite im Vergleich zu dem Umschalten mit Phasensprung. Des Weiteren ermöglicht CPFSK ein schnelleres Einschwingen des Empfängers auf das modulierte Signal [17, S. 993].

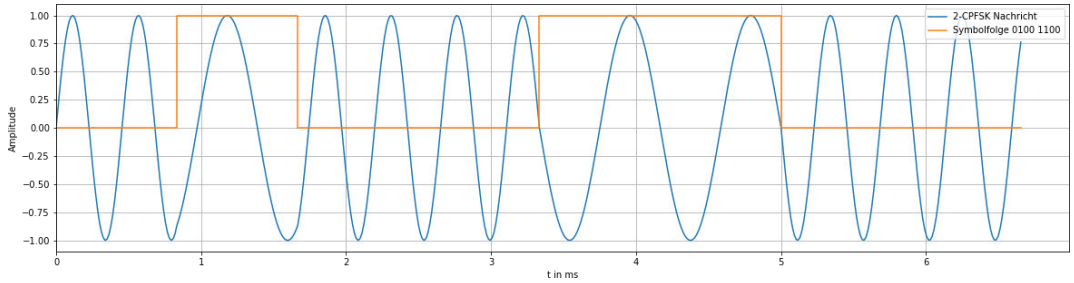

Abbildung 3.2: Veranschaulichung der binären Frequenzumtastung ohne Phasensprung

Ein FSK-Demodulator hat den Vorteil, dass er zwischen bekannten diskreten Frequenzen unterscheidet und infolgedessen weniger empfindlich gegenüber Störungen ist als ein Demodulator für analoge Frequenzmodulationen [25, S. 270].

3.5 Highway-Addressable-Remote-Transducer-Standard

3.5.1 Physikalische Schicht

Der HART-Standard beschreibt ein Protokoll in Schichten des Open-System-Interconnection (OSI)-Referenzmodells nach der Norm International-Organization-for-Standardization (ISO) 7498 [9].
In der Datensicherungsschicht (Data Link Layer) [6] wird das HART-Protokoll beschrieben. Die Implementierung des HART-Protokolls erfolgt nicht in dieser Arbeit.
Die Spezifikation der physikalischen Schicht (Physical Layer) [7] definiert die Eigenschaften der Signalübertragung und erzeugten Signalformen, siehe Abbildung 3.3. [7, S. 9]

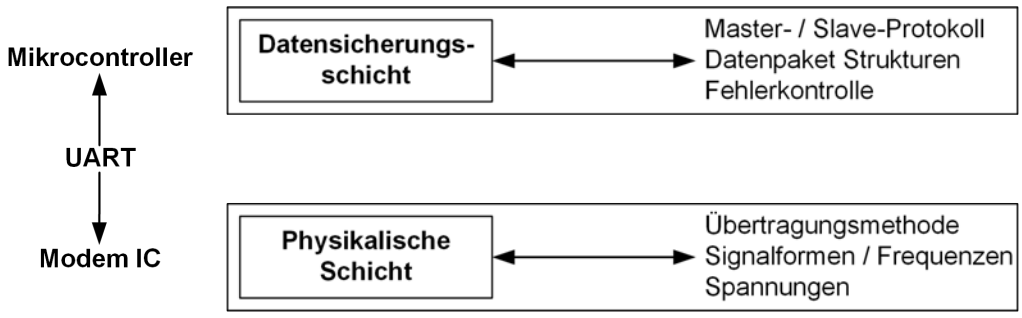

Abbildung 3.3: Der HART-Standard im Kommunikationsmodell

Das Softmodem operiert auf der physikalischen Schicht und stellt der Datensicherungsschicht Schnittstellen in Form von Funktionen zur Verfügung. Durch die bereitgestellten Schnittstellen werden die Daten übergeben.

3.5.2 Spezifikation der digitalen Datenübertragung

Basierend auf dem Bell 202 Standard, ist die digitale Datenübertragung des HART-Standards definiert. Die Frequenz des Trägers wird so moduliert, dass sie mit einer kontinuierlichen Phase zwischen zwei Frequenzen wechselt (Phasen kontinuierliches BFSK oder auch 2-CPFSK), siehe Abschnitt 3.4.
Aus der CPFSK-Modulation mit zwei unterschiedlichen Frequenzen folgt, dass ein Symbol einem Bit entspricht. Die Begriffe Symbol und Bit werden in der Spezifikation der physikalischen Schicht synonym verwendet.
Die beiden Modulationsfrequenzen sind als Mark-Frequenz mit $1,2\,\mathrm{kHz}$ (logisch 1) und Space-Frequenz mit $2,2\,\mathrm{kHz}$ (logisch 0) definiert. Die Symbolrate der Übertragung liegt bei $1,2\,\mathrm{kBaud}$. Abweichungen in den Frequenzen dürfen unabhängig voneinander eine Toleranz von $\pm 1\,\%$ nicht überschreiten. [7, S. 31-33]

3.5.3 Signalfolge für die digitale Übertragung von Daten

Der HART-Standard ermöglicht eine bidirektionale Halbduplex-Kommunikation zwischen einem Feldgerät (Slave) und der Zentrale (Master). Die digitale Datenübertragung erfolgt hierbei durch eine asynchrone Kommunikation zwischen den kommunizierenden Geräten. Es kommt ein Frage-Antwort-Protokoll zur Anwendung. Der Master initiiert die Kommunikation durch das Senden eines Trägersignals (Carrier). Solange das Trägersignal aktiv ist, kann der Master Daten an den Slave senden. Um das Ende der Übertragung durch den Master festzustellen, nutzt der Slave eine Trägererkennung (Carrier-Detect). Sobald der Carrier des Masters nicht mehr aktiv ist, antwortet der Slave dem Master, indem er ein Trägersignal gefolgt von den Daten der Antwort sendet. Nachdem die Antwort vollständig übermittelt wurde, stoppt der Slave das Senden des Carriers und beendet so die Kommunikation mit dem Master.

Zu übertragende Daten werden in Datenrahmen übertragen. Ein Datenrahmen besteht aus einer Präambel und den zu übertragenden Daten. Die zu übertragenden Daten werden auch als Nachricht bezeichnet. [7, S. 17]

Die Präambel und Nachricht werden durch eine Reihe von Bytes übertragen. Zwischen aufeinander folgenden Bytes darf keine Lücke auftreten. Für die Codierung der Bytes wird das UART-Zeichen-Format genutzt, siehe Abbildung 3.4. Ein Zeichen besteht aus einer Abfolge von 11 Symbolen und enthält die Daten eines Bytes. Jedes Zeichen setzt sich aus einem Startsymbol (logisch 0), 8 Datensymbolen (Least-Significant-Bit (LSB)-First), einem Paritätssymbol (mit einer ungeraden Parität) und einem Stopsymbol (logisch 1) zusammen. [6, S. 42-43]

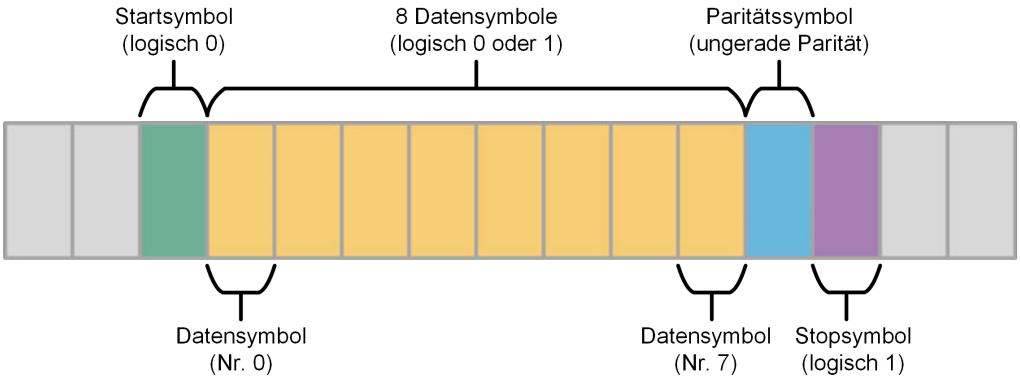

Abbildung 3.4: Das UART-Zeichen-Format

Die Präambel besteht aus einer Folge von 5 bis maximal 20 Zeichen, deren Datensymbole den Wert 0xFF repräsentieren [7, S. 18]. Durch die Präambel kann sich der Empfänger auf das zu empfangende Signal einstellen. Da der Empfänger für die Synchronisation Zeit benötigt, wird davon ausgegangen, dass einige Zeichen der Präambel nicht korrekt empfangen werden. Die empfangenen Daten sind auch bei dem Verlust einiger Zeichen der Präambel gültig. [6, S. 42]

4 Entwurf des Modulations- und Demodulationsverfahrens

Dieses Kapitel beschreibt, wie die im Softmodem verwendeten Methoden ausgewählt werden. Nach eingehender Recherche möglicher Konzepte für die Umsetzung des Softmodems stehen mehrere Methoden für das Senden und Empfangen von Symbolen zur Auswahl. Die möglichen Methoden werden in einer Simulation miteinander verglichen, um das für die Umsetzung des Softmodems am besten geeignete Verfahren zu bestimmen.
Abschnitt 4.1 beschreibt die Methoden zur Übertragung von Symbolen. Die Konzepte für den Empfang von Symbolen sind in Abschnitt 4.2 erläutert. Beide Abschnitte werden mit dem Vergleich der vorgestellten Methoden sowie der Analyse der Simulationsergebnisse und der darauf basierenden Auswahl des am besten geeigneten Verfahrens abgeschlossen. Durch die Simulation der Methoden vor der Umsetzung in einer hardwarenahen Programmiersprache wird der Zeitaufwand für die Entwicklung der möglichen Konzepte gesenkt.

4.1 Modulationsverfahren

Das Ziel ist es, eine Folge mit zwei verschiedenen Symbolen ohne Phasensprung modulieren zu können. Hierfür müssen in Abhängigkeit des zu modulierenden Symbols die zugehörigen Amplitudenwerte bestimmt werden. Für die Berechnung der Amplitudenwerte ist die Generierung eines zeit- und wertediskreten Sinussignals mit zwei unterschiedlichen Frequenzen notwendig. Die generierten Amplitudenwerte werden später durch einen DAC in ein Spannungssignal umgewandelt.

Ein kontinuierliches Sinussignal $x(t)$ mit der Amplitude A und der Frequenz f wird durch Gleichung 4.1 beschrieben.

$$x(t) = A \cdot \sin(\Phi(t)) = A \cdot \sin(2 \cdot \pi \cdot f \cdot t) \tag{4.1}$$

$$\Phi(t) = 2 \cdot \pi \cdot f \cdot t \tag{4.2}$$

Gleichung 4.1 zeigt, dass eine zeitliche Änderung im Sinussignal $x(t)$ ausschließlich von der Winkelfunktion $\Phi(t)$ Gleichung 4.2 hervorgerufen wird. Die Diskretisierung von $x(t)$ erfolgt in Gleichung 4.3 durch die Abtastung mit der Abtastfrequenz F_S.

$$x\left(t = n \cdot \frac{1}{F_\mathrm{S}} = n \cdot T_\mathrm{S}\right) = x[n] = A \cdot \sin(\Phi(n \cdot T_\mathrm{S})) = A \cdot \sin(2 \cdot \pi \cdot f \cdot n \cdot T_\mathrm{S}) \tag{4.3}$$

$$\Phi(n \cdot T_\mathrm{S}) = \Phi[n] = 2 \cdot \pi \cdot f \cdot n \cdot T_\mathrm{S} \tag{4.4}$$

$$\Phi[n] = n \cdot \Delta\Phi_f = \Phi[n-1] + \Delta\Phi_f \text{ mit } \Delta\Phi_f = 2 \cdot \pi \cdot f \cdot T_\mathrm{S} \tag{4.5}$$

Die Winkelfunktion $\Phi[n]$ aus Gleichung 4.4 bildet das Argument n auf den Wertebereich $W = [0,\ \infty)$ ab. Die Periodizität in $x[n]$ Gleichung 4.3 stammt aus der idealen Sinusfunktion. Daraus folgt, dass für die Berechnung jedes einzelnen Amplitudenwerts der Aufruf der Sinusfunktion aus einer Bibliothek notwendig ist.
Um zwischen zwei Symbolen ohne Phasensprung wechseln zu können, wird der Momentanwert der Phase $\Phi[n]$ gespeichert. Gleichung 4.5 zeigt, dass $\Phi[n]$ von einem konstanten Faktor $\Delta\Phi_f$ und dem aktuellen Abtastpunkt n abhängig ist. Aufgrund dessen ist es möglich, $\Phi[n]$ durch eine Addition des vorherigen Werts $\Phi[n-1]$ und $\Delta\Phi_f$ zu berechnen. Der Faktor $\Delta\Phi_f$ ist von der Frequenz des zu erzeugenden Symbols abhängig, es kann also für das Mark-Symbol $\Delta\Phi_{f_\mathrm{Mark}}$ und für das Space-Symbol $\Delta\Phi_{f_\mathrm{Space}}$ bestimmt werden. Wird der gespeicherte Wert $\Phi[n-1]$ in der Frequenz F_S um $\Delta\Phi_{f_\mathrm{Mark,Space}}$ inkrementiert, hat dies die Erzeugung der Amplitudenwerte des entsprechenden Symbols zur Folge. Durch den Wechsel des Werts $\Delta\Phi_{f_\mathrm{Mark,Space}}$, um den $\Phi[n-1]$ inkrementiert wird, kann zwischen den beiden Symbolfrequenzen ohne Phasensprung umgeschaltet werden. [5, S. 14-16]

Unter Berücksichtigung des Abtasttheorems $F_\mathrm{S} > 2 \cdot f$ mit $f = 2,2\,\mathrm{kHz}$ und einer Symbolrate von $1,2\,\mathrm{kBaud}$, sind pro erzeugtem Symbol mindestens 3,67 Amplitudenwerte zu berechnen, siehe Gleichung 4.6.

$$\frac{F_\mathrm{S}}{f_\mathrm{Baud}} > \frac{2 \cdot 2,2\,\mathrm{kHz}}{1,2\,\mathrm{kHz}} = 3,67 \tag{4.6}$$

Für die Umsetzung des Softmodems wird in dieser Arbeit die Abtastfrequenz $F_S = 38,4\,\text{kHz}$ gewählt. Es ergeben sich somit 32 Punkte pro Symbol. Die Abtastfrequenz von $38,4\,\text{kHz}$ wurde gewählt, um das Signal-To-Noise-Ratio (SNR) des erzeugten Signals zu verbessern. Eine Optimierung der Abtastfrequenz unter Berücksichtigung des geforderten SNR der FSK-Physical-Layer-Specification ist nicht Teil dieser Arbeit.
Der in diesem Abschnitt beschriebene allgemeine Ansatz soll für den Einsatz in eingebetteten Anwendungen angepasst werden. Ziel ist es, den Aufruf der Sinusfunktion (aus einer Bibliothek) durch eine Lookup-Tabelle zu ersetzen, um möglichst gleiche und kurze Ausführungszeiten für die Bestimmung der Sinuswerte zu ermöglichen.
Im Folgenden werden zwei Konzepte beschrieben und anhand ihrer Eigenschaften und der gewonnenen Simulationsergebnisse miteinander verglichen.

4.1.1 Sinusgenerator mit beliebiger Frequenz

Eine Periode einer Sinusschwingung wird an N Punkten abgetastet. Die N abgetasteten Werte werden in einer Lookup-Tabelle gespeichert.
Durch Inkrementieren des Index $i_{\Phi n} \in [0,\ N-1]$ mit dem Inkrement Δi_Φ, siehe Gleichung 4.7, wird die Lookup-Tabelle durchlaufen und so die entsprechenden Amplitudenwerte des Verlaufs einer Sinusschwingung ausgegeben.

$$i_{\Phi n} = i_{\Phi n-1} + \Delta i_\Phi \tag{4.7}$$

Für einen periodischen Signalverlauf muss der Index $i_{\Phi n}$, nachdem er die letzte Position der Lookup-Tabelle erreicht hat, an die erste Position zurückgesetzt werden.
Das Überlaufverhalten des Integer-Datentyps wird genutzt, um eine Modulo-Operation für das Zurücksetzen des Index am Ende jedes Durchlaufs der Lookup-Tabelle einzusparen. Dazu wird das Indexintervall der erstellten Lookup-Tabelle entsprechend Gleichung 4.8 abgebildet.

$$i_{\Phi n} \in [0,\ N-1] \mapsto I_{\Phi n} \in [0, 1) \tag{4.8}$$

$I_{\Phi n}$ entspricht dem Momentanwert der Phase einer Sinusschwingung im Intervall $[0,\ 1)$ und wird durch eine Festkommazahl im Q-Format [4, S. 86] in der Form Q0.m repräsentiert. Die Erzeugung der Symbolfrequenzen sowie der Wechsel zwischen den Frequenzen ohne Phasensprung wird nach der im vorigen Abschnitt beschriebenen Methode realisiert.

Durch die Abbildung der Phase einer Sinusschwingung auf das Intervall $[0, 1)$ statt $[0, 2\pi)$ vereinfacht sich die Berechnung von $\Delta\Phi_f$, da der Faktor 2π entfällt, siehe Gleichung 4.9. Die Umrechnung von $\Delta\Phi_f$ zu einer Q0.m-Festkommazahl ist in Gleichung 4.10 beschrieben.

$$\Delta\Phi_f = f \cdot T_\mathrm{S} = \frac{f}{F_\mathrm{S}} \tag{4.9}$$

$$\Delta I_{\Phi_f} = \Delta\Phi_f \cdot (2^m - 1) \tag{4.10}$$

Das Verhältnis von F_S zu f ist die Anzahl der Abtastpunkte innerhalb einer Sinusschwingung mit der Frequenz f. Daraus folgt, dass wenn der momentane Phasenwert im Intervall [0, 1) mit $\Delta\Phi_f$ in der Frequenz F_S inkrementiert wird, $\Delta\Phi_f$ der Kehrwert der Abtastpunkte pro Schwingung sein muss, um einen Überlauf des Index mit der Frequenz f zu erzeugen, siehe Gleichung 4.11.

$$f = \Delta\Phi_f \cdot F_\mathrm{S} \tag{4.11}$$

Jeder Überlauf des Index entspricht einem Durchlauf der Lookup-Tabelle. Es wird ein Sinussignal mit der Frequenz f generiert.
Die aktuelle Phase ist, wie im vorherigen Abschnitt beschrieben, im Index $I_{\Phi n}$ gespeichert. Durch das Anpassen von ΔI_{Φ_f} können aus einer Lookup-Tabelle Sinusschwingungen mit verschiedenen Frequenzen erzeugt werden. Der Wechsel des Inkrements ΔI_{Φ_f} zwischen $\Delta I_{\Phi_{f_\mathrm{Mark}}}$ und $\Delta I_{\Phi_{f_\mathrm{Space}}}$ ermöglicht, dass die Frequenz der generierten Schwingung ohne Phasensprung verändert wird.

Die Anzahl der Punkte N in der Lookup-Tabelle bestimmt das Raster der darstellbaren Phasenwerte. Der Index $I_{\Phi n}$ in Festkommadarstellung muss auf den nächstliegenden Wert für $I_{\Phi n}$ gerundet werden, der in der Lookup-Tabelle vorhanden ist. Wenn die Anzahl der Punkte N in der Lookup-Tabelle als 2er-Potenz gewählt wird, kann die Abbildung von $I_{\Phi n}$ auf den nächstliegenden Wert in der Tabelle durch eine binäre Schiebeoperation durchgeführt werden.
Die in diesem Unterabschnitt beschriebenen Konzepte sind aus dem Programmcode der FSK-Sendemethode des Github-Repository [24] abgeleitet worden. In dem Github-Repository wird die Entwicklung eines Automatic-Packet-Reporting-System (APRS) beschrieben. Für die Übertragung der Nachrichten in dem Projekt wird der Bell 202 Standard genutzt. Da die physikalische Schicht des HART-Standards auch auf Bell 202 basiert, ist die Analyse des Programmcodes als Referenz relevant.

4.1.2 Sinusgenerator mit 2 verschiedenen Frequenzen

Das Dokument [5, S. 15-16] der Firma Texas Instruments beschreibt eine weitere Methode, um einen Sinusgenerator mit 2 verschiedenen Frequenzen zu realisieren.
Wie bei der Methode im vorherigen Abschnitt wird eine Lookup-Tabelle mit N Punkten erzeugt, die durch Steigerung des Index $i_{\Phi n}$ durchlaufen werden, siehe Gleichung 4.7. Im Gegensatz zu der Methode aus dem vorherigen Abschnitt wird die Anzahl der Punkte in der Lookup-Tabelle N so gewählt, dass der Wert des Inkrements $\Delta i_{\Phi_{f_{\mathrm{Mark,Space}}}}$ für die Erzeugung von f_{Mark} und f_{Space} eine ganze Zahl ist. Gleichung 4.12 und Gleichung 4.13 müssen erfüllt sein.

$$\Delta i_{\Phi_{f_{\mathrm{Mark}}}} = \frac{f_{\mathrm{Mark}}}{F_{\mathrm{S}}} \cdot N \; mit \; \Delta i_{\Phi_{f_{\mathrm{Mark}}}} \in \mathbb{N} \tag{4.12}$$

$$\Delta i_{\Phi_{f_{\mathrm{Space}}}} = \frac{f_{\mathrm{Space}}}{F_{\mathrm{S}}} \cdot N \; mit \; \Delta i_{\Phi_{f_{\mathrm{Space}}}} \in \mathbb{N} \tag{4.13}$$

Für einen periodischen Signalverlauf muss der Index $i_{\Phi n}$, nachdem er die letzte Position der Lookup-Tabelle erreicht hat, an den Anfang der Lookup-Tabelle zurückgesetzt werden. Hierfür ist eine Modulo-Operation notwendig. Die beschriebene Methode ermöglicht es, zwei durch die Wahl der Parameter festgelegte Sinusschwingungen aus einer Lookup-Tabelle zu erzeugen und die Frequenz der generierten Schwingung ohne Phasensprung zu ändern.

4.1.3 Vergleich der Sendemethoden

Der Vergleich ist in drei Abschnitte unterteilt. Im ersten Abschnitt werden die systematischen Eigenschaften der Methoden analysiert. Der zweite Teil untersucht das Verhalten der Methoden durch eine Simulation und deren Auswertung. Die Auswahl der Sendemethode erfolgt im abschließenden Fazit des Vergleichs.

Eigenschaften

- Die in Unterabschnitt 4.1.1 beschriebene Einsparung der Modulo-Operation für das Zurücksetzen des Index am Ende jedes Durchlaufs der Lookup-Tabelle, kann für das in Unterabschnitt 4.1.2 beschriebene Verfahren nicht allgemein genutzt werden. Der Grund ist, dass der Parameter N bei dem Sinusgenerator mit 2 verschiedenen Frequenzen nicht frei wählbar ist und somit nicht immer als 2er-Potenz gewählt werden kann.
- Der Sinusgenerator mit beliebiger Frequenz kann durch Anpassung des Parameters ΔI_{Φ_f} Signale mit geänderten Frequenzen generieren. Hierfür ist bei dem Sinusgenerator mit 2 verschiedenen Frequenzen eine erneute Auslegung der in Unterabschnitt 4.1.2 beschriebenen Parameter und eine erneute Generierung der Lookup-Tabelle notwendig.
- Der Sinusgenerator mit 2 verschiedenen Frequenzen nutzt eine Lookup-Tabelle mit einer an den Fall angepassten Anzahl von Amplitudenwerten. Für den Sinusgenerator mit beliebiger Frequenz wird eine Lookup-Tabelle mit 2^n Amplitudenwerten genutzt. Dies führt zu einem größeren Speicherbedarf, verglichen mit der angepassten Lookup-Tabelle.
- Das Signal des Sinusgenerators mit beliebiger Frequenz hat höhere Quantisierungsfehler der Amplitudenwerte als das Signal des Sinusgenerators mit 2 verschiedenen Frequenzen. Ursache ist, dass zusätzlich zu der Quantisierung durch den DAC, noch ein Fehler durch die Rundung des Index auf den nächstliegenden Wert in der Lookup-Tabelle hinzukommt. Der Fehler durch die Rundung des Index kann durch eine Vergrößerung der Lookup-Tabelle verkleinert werden.

Simulation

Es werden die Frequenzen $f_{\text{Mark}} = 1,2\,\text{kHz}$ und $f_{\text{Space}} = 2,2\,\text{kHz}$ mit den Methoden aus Unterabschnitt 4.1.1 und Unterabschnitt 4.1.2 über 2 Symbolzeiten (1,67 ms) generiert. Mit der Abtastfrequenz $F_{\text{S}} = 38,4\,\text{kHz}$ ergeben sich 32 Punkte pro Symbol.

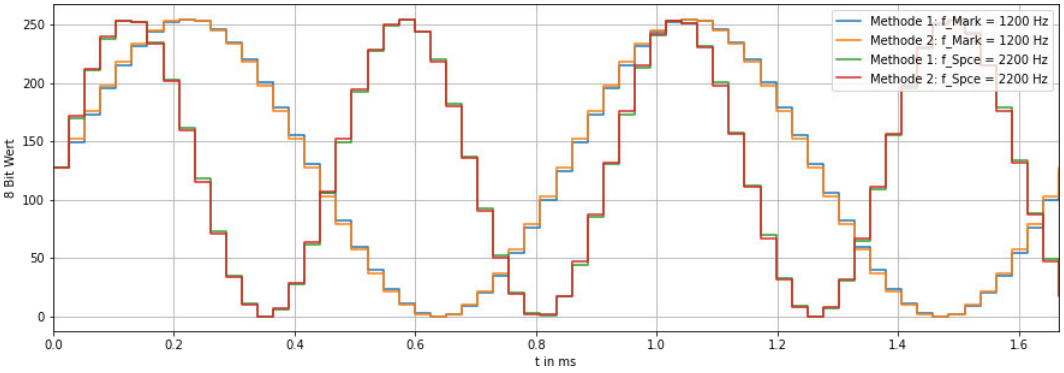

Abbildung 4.1: Vergleich der generierten Signale des Sinusgenerators mit beliebiger Frequenz (Methode 1, $N = 256$) und des Sinusgenerators mit 2 verschiedenen Frequenzen (Methode 2, $N = 192$)

Abbildung 4.1 zeigt, dass die in Unterabschnitt 4.1.1 und Unterabschnitt 4.1.2 beschriebenen Verfahren Sinussignale erzeugen, die sich nur in der Amplitude unterscheiden. Die gezeigten Abweichungen in der Amplitude der Signale schränken die Einsetzbarkeit für das Softmodem nicht ein. Für eine CPFSK-modulierte Datenübertragung ist die Frequenz der Signale ausschlaggebend.

Fazit

Beide vorgestellten Sendemethoden sind grundsätzlich geeignet, um das modulierte Signal zu erzeugen. Die Analyse der Eigenschaften hat gezeigt, dass der Sinusgenerator mit beliebiger Frequenz eine Rechenoperation weniger benötigt als der Sinusgenerator mit 2 verschiedenen Frequenzen. Des Weiteren können Anpassungen in den erzeugten Frequenzen leichter vorgenommen werden als in der Variante mit 2 festgelegten Frequenzen. Entsprechend der Anforderung NFA.3.3 ist eine geringere Anzahl der für die Ausführung benötigten Taktzyklen höher zu gewichten als eine Einsparung im Speicherbedarf. Dadurch und aufgrund des in diesem Abschnitt beschriebenen Vergleichs wird der Sinusgenerator mit beliebiger Frequenz aus Unterabschnitt 4.1.1 für die Umsetzung in dem Softmodem gewählt.

4.2 Demodulationsverfahren

Das Softmodem benötigt eine Methode, um das empfangene Signal zu demodulieren. Für die Auswahl des Verfahrens für die Demodulation des 2-CPFSK-Signals Abbildung 4.2 werden vier Demodulationsverfahren beschrieben und simuliert. Die Auswahl des Demodulationsverfahrens für den Einsatz in dem Softmodem erfolgt in Unterabschnitt 4.2.5 durch einen Vergleich der vorgestellten Verfahren. Alle vier beschriebenen Demodulationsverfahren nutzen in den Simulationen einen Infinite-Impulse-Response (IIR)-Tiefpassfilter 2ter Ordnung. Der Entwurf des Tiefpassfilters für das in Unterabschnitt 4.2.5 ausgewählte Demodulationsverfahren ist in Abschnitt 4.3 näher beschrieben.
Die Simulationen der vier Demodulationsverfahren weisen abweichende Mindestanforderungen an die Abtastfrequenzen F_S auf. Aufgrund dessen werden sie im Folgenden mit unterschiedlichen Abtastfrequenzen simuliert. Die Abtastfrequenz F_S wird jeweils so gewählt, dass das Funktionsprinzip der simulierten Methode gezeigt werden kann. Um jedes Symbol mit gleich vielen Punkten abzutasten, wird die Abtastfrequenz als Vielfaches der Symbolrate von $1,2\,\mathrm{kBaud}$ gewählt. Die Abtastfrequenz F_S für die Implementierung im Softmodem wird nach der Auswahl des genutzten Demodulationsverfahrens festgelegt.

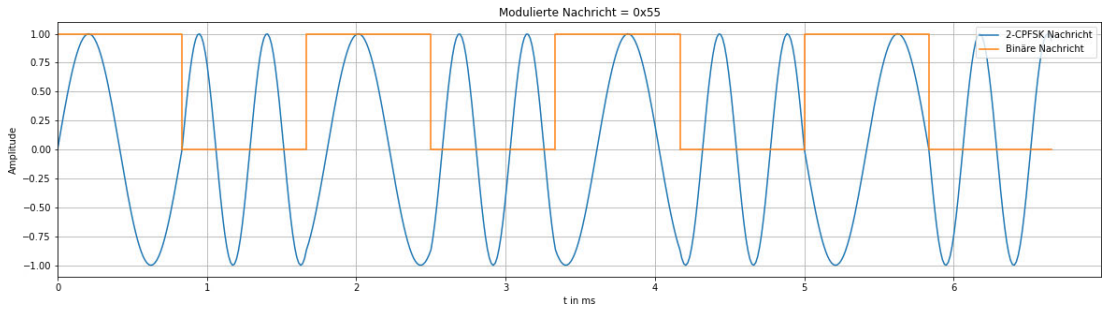

Abbildung 4.2: Testsignal für die Veranschaulichung der Demodulationsverfahren, modulierte Nachricht 0x55, kein UART-Format

4.2.1 Nullstellenzähler

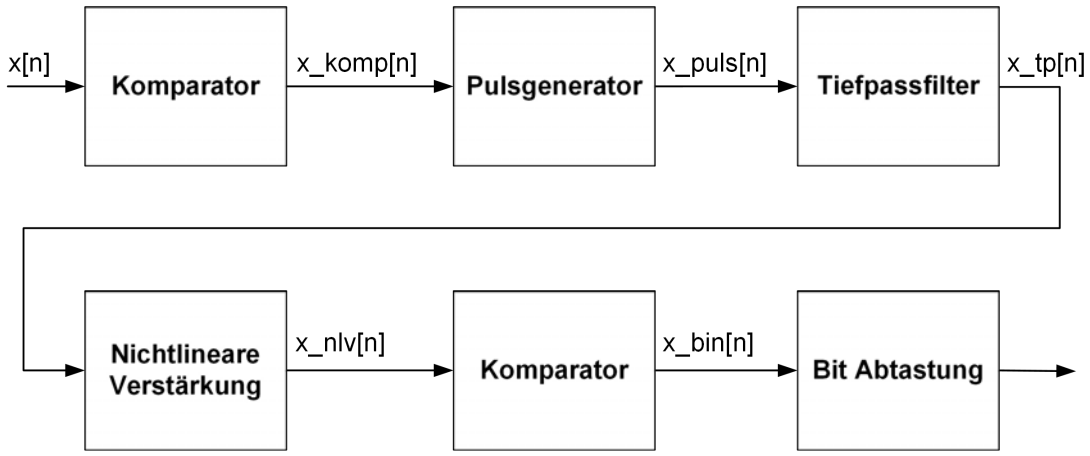

Abbildung 4.3: Blockschaltbild des Nullstellenzählers

Abbildung 4.3 zeigt den Aufbau des Nullstellenzählers. Das Eingangssignal $x[n]$ in Abbildung 4.4 wird mit $9,6\,\mathrm{kHz}$ abgetastet. Ein Komparator wandelt das Eingangssignal in ein Rechtecksignal $x_{\mathrm{komp}}[n]$ um. Bei jedem Nulldurchgang des Eingangssignals tritt ein Flankenwechsel auf. Der Pulsgenerator erkennt die Flankenwechsel in dem Rechtecksignal und erzeugt für jeden Flankenwechsel einen Impuls $x_{\mathrm{puls}}[n]$. Die Impulse werden durch einen Tiefpassfilter gefiltert $x_{\mathrm{tp}}[n]$ und das Signal somit vollständig demoduliert. [14, S. 47-48]
Die Symbolfolge ist in $x_{\mathrm{tp}}[n]$ invertiert, da das Space-Symbol (logisch 0) durch eine höhere Frequenz repräsentiert wird als das Mark-Symbol (logisch 1) und infolgedessen mehr Impulse erzeugt werden. In $x_{\mathrm{tp}}[n]$ zeigt sich, dass die getestete Symbolfolge nur zu kleinen Unterschieden in dem demodulierten Signal führt.
Der Nullstellenzähler funktioniert am besten, wenn das zu demodulierende Signal möglichst viele Nulldurchgänge pro Symbol aufweist. Die Symbolrate und Übertragungsfrequenz des in Unterabschnitt 3.5.2 definierten Signals sind für die untere Übertragungsfrequenz f_{Mark} identisch. Die Folge ist, dass es wenige Nullstellen pro Symbol gibt und somit die Robustheit des Demodulationsverfahrens im Vergleich zu anderen Demodulationsverfahren bei einer gleichen Abtastfrequenz gering ist. [18, S. 2]
Durch eine nicht lineare Verstärkung des demodulierten Signals $x_{\mathrm{nlv}}[n]$ kann das Ergebnis der Demodulation weiter verbessert werden.

Die Verbesserung ist möglich, da die nicht lineare Verstärkung größere Werte im Vergleich zu kleineren Werten überproportional stark verstärkt und so zu einer stärkeren Separation der Ergebnisse führt. Aus dem Ausgangssignal des Demodulators $x_{\mathrm{nlv}}[n]$ wird die Symbolfolge korrekt zurückgewonnen. Die Methode für die Rückgewinnung der Symbolfolge ist für alle betrachteten Demodulationsverfahren gleich und wird in Abschnitt 4.4 vorgestellt.

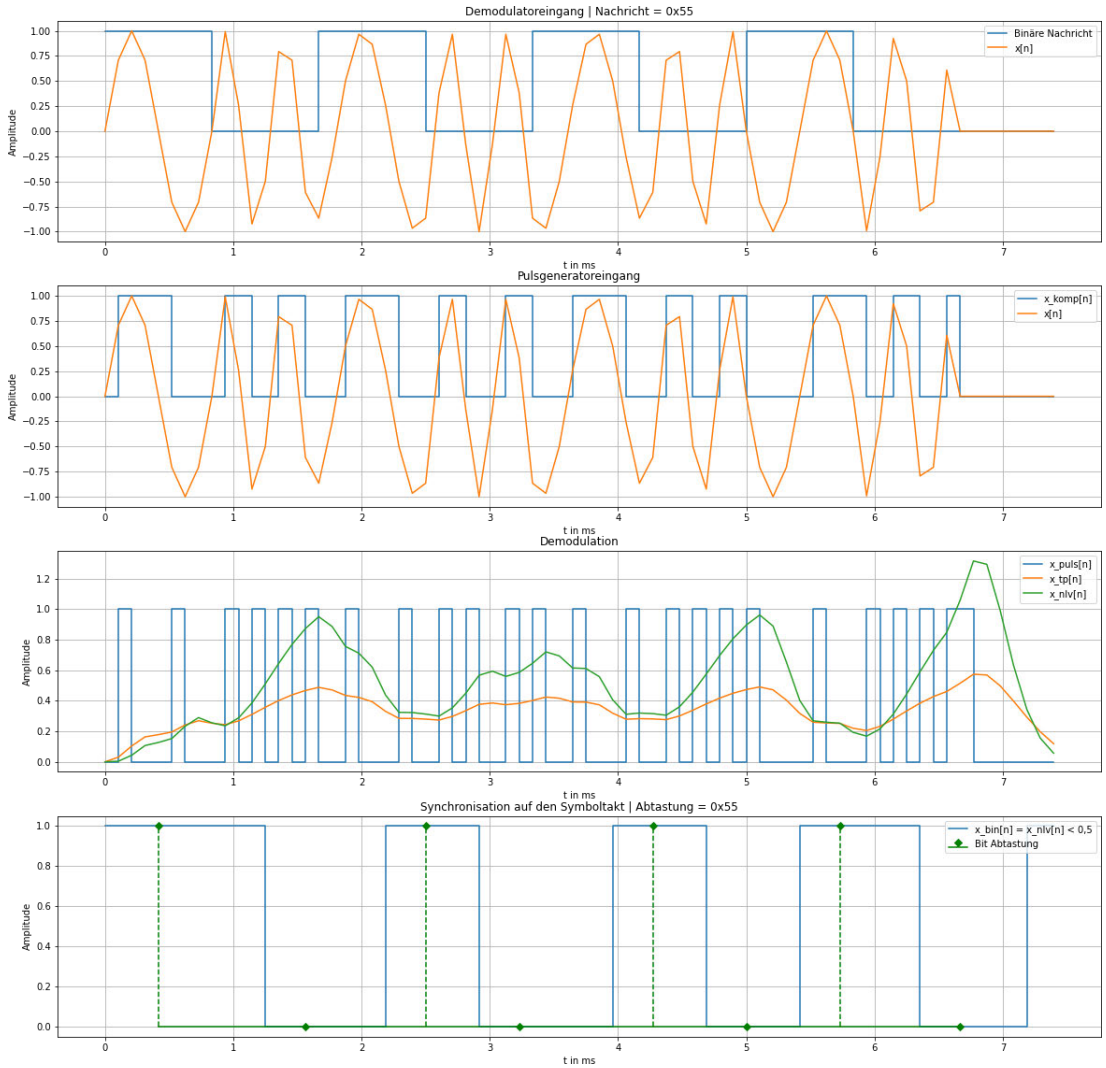

Abbildung 4.4: Simulation der Signale in dem Nullstellenzähler mit der Abtastfrequenz $F_{\mathrm{S}} = 9,6\,\mathrm{kHz}$

4.2.2 Delay-Line-Demodulator mit Multiplikator

Der Delay-Line-Demodulator Abbildung 4.5 multipliziert das Eingangssignal mit einer verzögerten Version des Eingangssignals. Der Ausgang des Multiplikators wird mit einem Tiefpassfilter gefiltert.

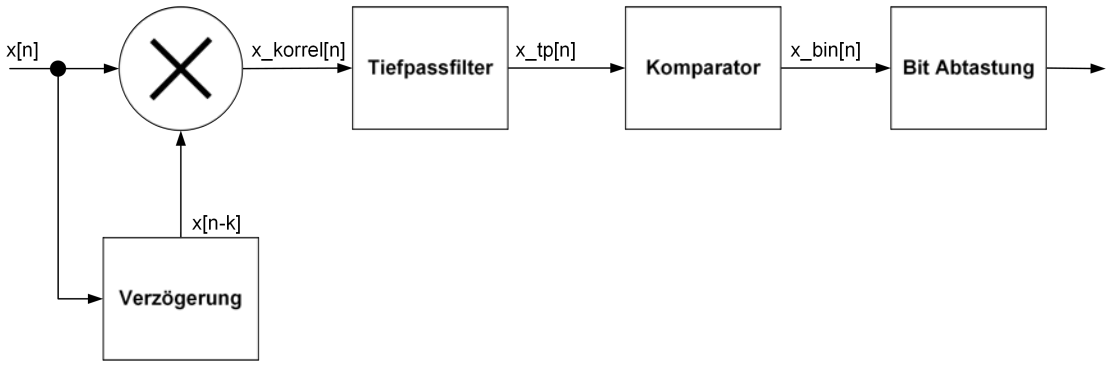

Abbildung 4.5: Blockschaltbild Delay-Line-Demodulator mit Multiplikator

Die Multiplikation des Eingangssignals mit seiner verzögerten Version zeigt die Korrelation zwischen den beiden Signalen. Sind beide Signale positiv oder beide negativ, ist der Ausgang des Multiplikators positiv. Ist eines der beiden Signale negativ und das andere positiv, ist der Ausgang negativ.
Durch die analytische Berechnung des Multiplikatorausgangs Gleichung 4.14 wird gezeigt, dass das Ausgangssignal aus einer Überlagerung von einem konstanten Term und einer Schwingung mit der doppelten Signalfrequenz besteht, Gleichung 4.15.

$$x_{\text{korrel}}[n] = x[n] \cdot x[n-k] = a \cdot \cos(2\pi \cdot f \cdot n \cdot T_{\text{S}} - \varphi) \cdot a \cdot \cos(2\pi \cdot f \cdot [n-k] \cdot T_{\text{S}} - \varphi) \tag{4.14}$$

$$x_{\text{korrel}}[n] = \frac{a^2}{2} \cdot \cos(2\pi \cdot f \cdot k \cdot T_{\text{S}}) + \frac{a^2}{2} \cdot \cos(2\pi \cdot f \cdot (k - 2 \cdot n) \cdot T_{\text{S}} + 2 \cdot \varphi) \tag{4.15}$$

Der erste Teil des Ausgangssignals Gleichung 4.15 ist nicht von n und somit auch nicht von der Zeit abhängig. Es ergibt sich ein konstanter Wert in Abhängigkeit der Frequenz f und der Verzögerung k. Der zweite Teil mit der doppelten Signalfrequenz wird durch einen Tiefpassfilter herausgefiltert.

Unter der Annahme eines idealen Tiefpassfilters folgt Gleichung 4.16.

$$x_{\mathrm{tp}}[n] = \frac{a^2}{2} \cdot \cos(2\pi \cdot f \cdot k \cdot T_{\mathrm{S}}) \tag{4.16}$$

Gleichung 4.16 ist abhängig von der Frequenz f. Es ergeben sich für f_{Mark} und f_{Space} unterschiedliche Werte am Ausgang des Tiefpassfilters. Durch geeignete Wahl der Verzögerung k kann die Differenz zwischen den Ausgangswerten des Tiefpassfilters maximiert werden, siehe Gleichung 4.17. Der Vorfaktor des Cosinus-Terms hat keinen Einfluss auf die Wahl der optimalen Verzögerung und wird bei der Bestimmung des Parameters k vernachlässigt.

$$|d| = |\cos(2\pi \cdot f_{\mathrm{Mark}} \cdot k \cdot T_{\mathrm{S}}) - \cos(2\pi \cdot f_{\mathrm{Space}} \cdot k \cdot T_{\mathrm{S}})| \tag{4.17}$$

Die numerische Berechnung von Gleichung 4.17 in Abbildung 4.6 zeigt, dass die Differenz bei einer Verzögerung von $k = 4$ maximal wird.

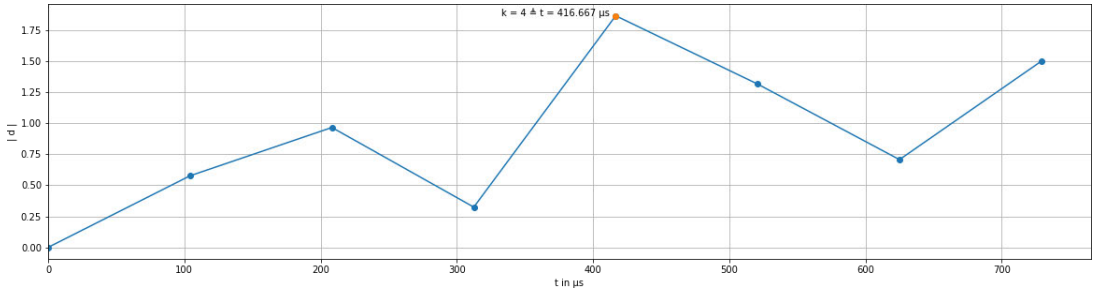

Abbildung 4.6: Bestimmung der optimalen Verzögerung mit 8 Abtastpunkten pro Symbol, Abtastfrequenz $F_{\mathrm{S}} = 9,6\,\mathrm{kHz}$

Durch die Simulation des Delay-Line-Demodulators wird dessen Funktionsprinzip visualisiert. In Abbildung 4.7 ist das mit $9,6\,\mathrm{kHz}$ abgetastete Eingangssignal $x[n]$ und das um $k = 4$ Abtastzeitpunkte verzögerte Eingangssignal $x[n-k]$ abgebildet. An dem Ausgang des Multiplikators $x_{\mathrm{korrel}}[n]$ ist die Überlagerung des konstanten Terms in Form der invertierten Symbolabfolge der binären Nachricht und einer Schwingung mit der doppelten Signalfrequenz des Eingangssignals zu erkennen. Das Signal $x_{\mathrm{tp}}[n]$ zeigt die Wirkung eines realen Tiefpassfilters auf den Ausgang des Multiplikators. Die Signalkomponente mit der doppelten Signalfrequenz ist fast vollständig herausgefiltert worden.

Das demodulierte Signal $x_{\mathrm{tp}}[n]$ hat die Form der invertierten binären Nachricht. Die in diesem Unterabschnitt beschriebenen theoretischen Konzepte sind aus [18, S. 2-3] entnommen und auf den Entwurf des Softmodems angepasst worden.

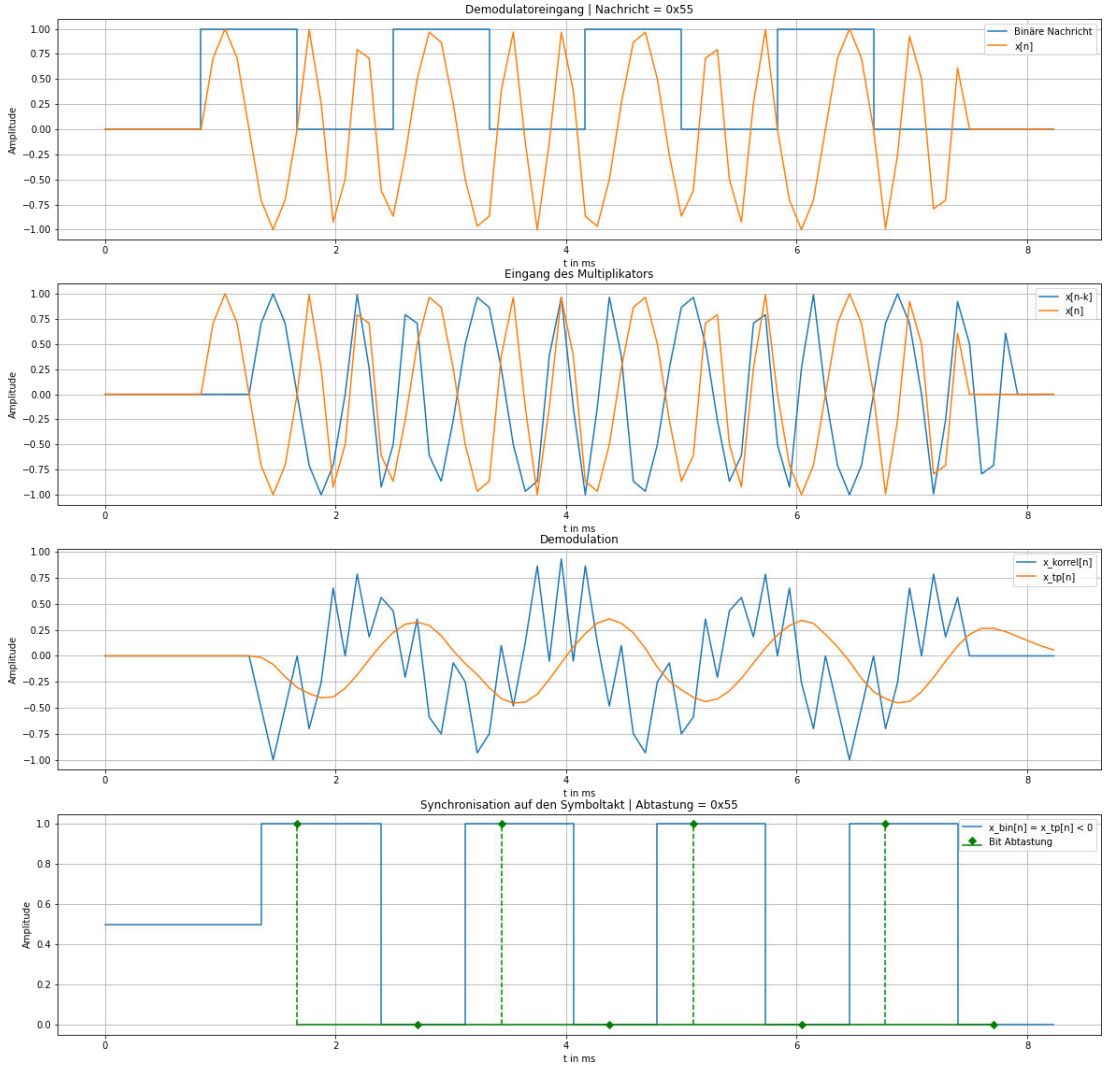

Abbildung 4.7: Simulation der Signale in dem Delay-Line-Demodulator mit Multiplikator mit der Abtastfrequenz $F_{\mathrm{S}} = 9,6\,\mathrm{kHz}$ und $k = 4$

4.2.3 Delay-Line-Demodulator mit Exklusiv-Oder-Gatter

Der Delay-Line-Demodulator mit Exklusiv-Oder-Gatter Abbildung 4.8 basiert auf dem im vorherigen Abschnitt beschriebenen Funktionsprinzip. Die Korrelation zwischen dem Eingangssignal und der verzögerten Version des Eingangssignals wird durch ein Exklusiv-Oder-Gatter bestimmt.

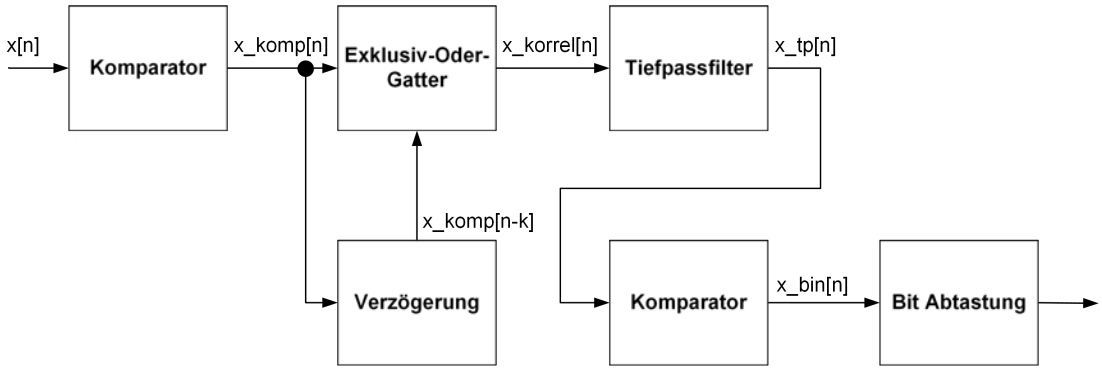

Abbildung 4.8: Blockschaltbild Delay-Line-Demodulator mit Exklusiv-Oder-Gatter

Die Simulation des Delay-Line-Demodulators mit Exklusiv-Oder-Gatter benötigt eine höhere Abtastfrequenz als die Simulation der anderen Verfahren. Für die Veranschaulichung dieses Demodulators wird im Vergleich zu den anderen Verfahren die doppelte Abtastfrequenz gewählt. Aufgrund der veränderten Abtastfrequenz wird die Verzögerung auf $k = 9$ bestimmt, siehe Gleichung 4.17. Das mit $F_S = 19,2\,\text{kHz}$ abgetastete Eingangssignal $x[n]$ in Abbildung 4.9 wird mit seiner um $k = 9$ Abtastzeitpunkte verzögerten Version durch ein digitales Exklusiv-Oder-Gatter korreliert $x_{\text{korrel}}[n]$. Hierfür wird das Eingangssignal von einem Komparator in die binären Signale $x_{\text{komp}}[n]$ und $x_{\text{komp}}[n-k]$ mit den möglichen Werten 0 und 1 umgewandelt. Das Signal am Ausgang des Exklusiv-Oder-Gatters wird durch einen realen Tiefpassfilter integriert $x_{\text{tp}}[n]$. Es zeigt sich die Symbolabfolge der binären Nachricht. Die in diesem Unterabschnitt beschriebenen theoretischen Konzepte sind aus [18, S. 3-8] entnommen.

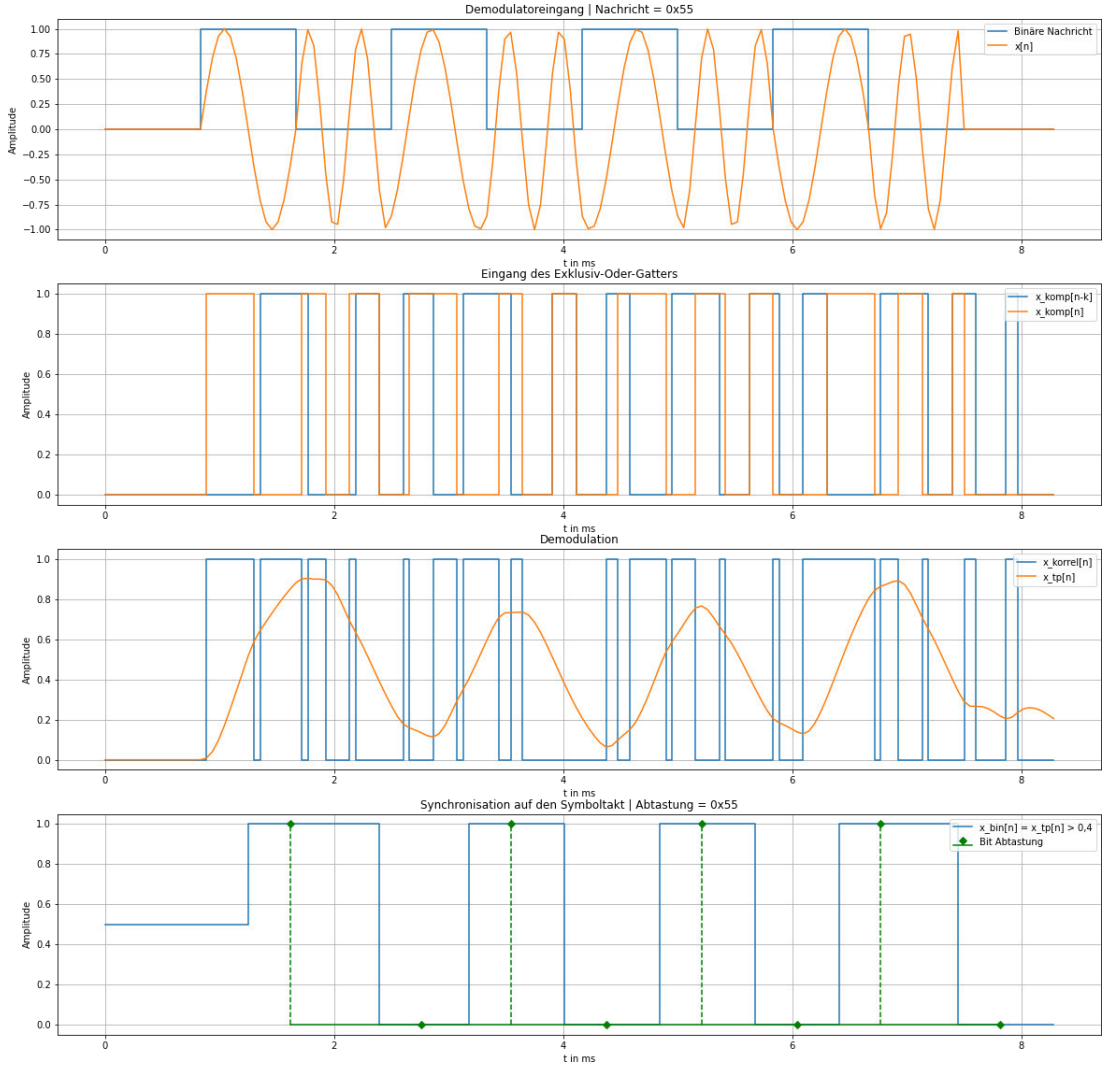

Abbildung 4.9: Simulation der Signale in dem Delay-Line-Demodulator mit Exklusiv-Oder-Gatter mit der Abtastfrequenz $F_S = 19,2\,\text{kHz}$ und $k = 9$

4.2.4 Quadraturdemodulator

Das empfangene Signal Gleichung 4.18 kann durch Anwendung des Additionstheorems aus Gleichung 4.19 in eine Überlagerung einer Sinus- und einer Cosinuskomponente aufgeteilt werden, Gleichung 4.20.

$$x[n] = a \cdot \cos(2\pi \cdot f \cdot n \cdot T_\mathrm{S} - \varphi) \tag{4.18}$$

$$cos(\alpha - \beta) = cos(\alpha) \cdot cos(\beta) + sin(\alpha) \cdot sin(\beta) \tag{4.19}$$

$$x[n] = a \cdot \cos(\varphi) \cdot \cos(2\pi \cdot f \cdot n \cdot T_\mathrm{S}) + a \cdot \sin(\varphi) \cdot \sin(2\pi \cdot f \cdot n \cdot T_\mathrm{S}) \tag{4.20}$$

Der Sinusanteil wird auch als Inphasenkomponente I und der Cosinusanteil als Quadraturkomponente Q bezeichnet [25, S. 255]. Gleichung 4.21 zeigt das empfange Signal aufgeteilt in seine I- und Q-Komponente.
Die Betragsbildung aus der I- und der Q-Komponente Gleichung 4.22 liefert den Betrag des empfangenen Signals. Die Phase des empfangenen Signals ist für die Demodulation der übertragenen Symbole nicht relevant und wird hier nicht weiter betrachtet.

$$x[n] = I \cdot \cos(2\pi \cdot f \cdot n \cdot T_\mathrm{S}) + Q \cdot \sin(2\pi \cdot f \cdot n \cdot T_\mathrm{S}) \tag{4.21}$$

$$a = \sqrt{I^2 + Q^2} \tag{4.22}$$

Der Quadraturdemodulator Abbildung 4.10 nutzt 4 Multiplikatoren, um durch Korrelation zu bestimmen, welche I- und Q-Komponenten das empfangene Signal bei den Symbolfrequenzen f_Mark und f_Space enthält. Durch einen Tiefpassfilter wird aus dem Ergebnis der Multiplikation die Komponente mit der doppelten Signalfrequenz herausgefiltert. Anschließend wird der Betrag der Komponenten $a_\mathrm{Mark}[n]$ und $a_\mathrm{Space}[n]$ gebildet, siehe Abbildung 4.11. Nach der Betragsbildung wird verglichen, ob der Betrag der Signalkomponenten bei der Frequenz f_Mark oder f_Space größer ist und so entschieden, welches Symbol empfangen wurde, siehe $x_\mathrm{bin}[n]$. Für die Demodulation ist entscheidend, welcher der Beträge größer ist. Da die Berechnung der Wurzel diese Entscheidung nicht beeinflusst, kann sie eingespart werden. Der in diesem Unterabschnitt beschriebene Quadraturdemodulator basiert auf den aus [10, S. 64-65] entnommenen Konzepten.

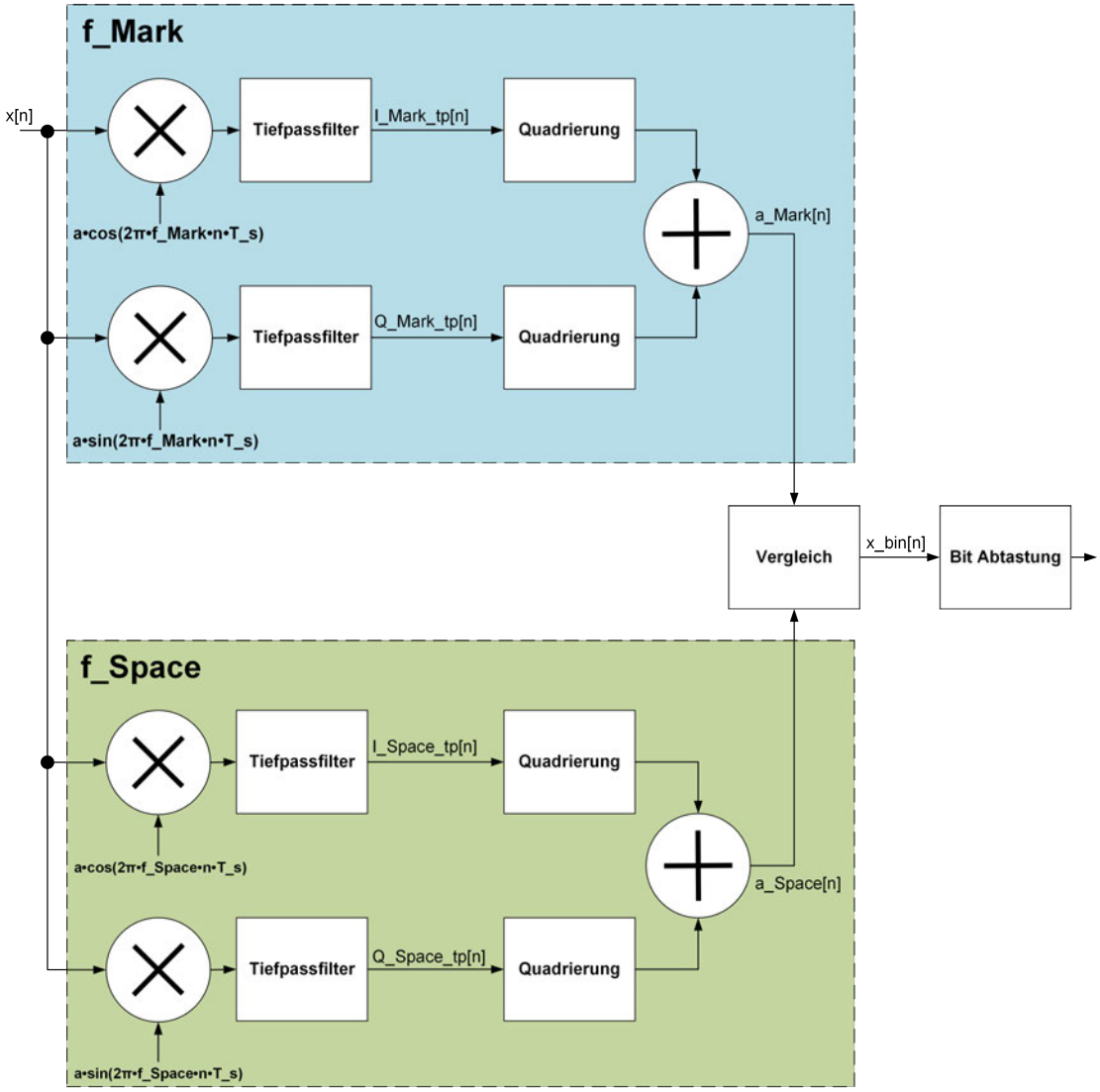

Abbildung 4.10: Blockschaltbild Quadraturdemodulator

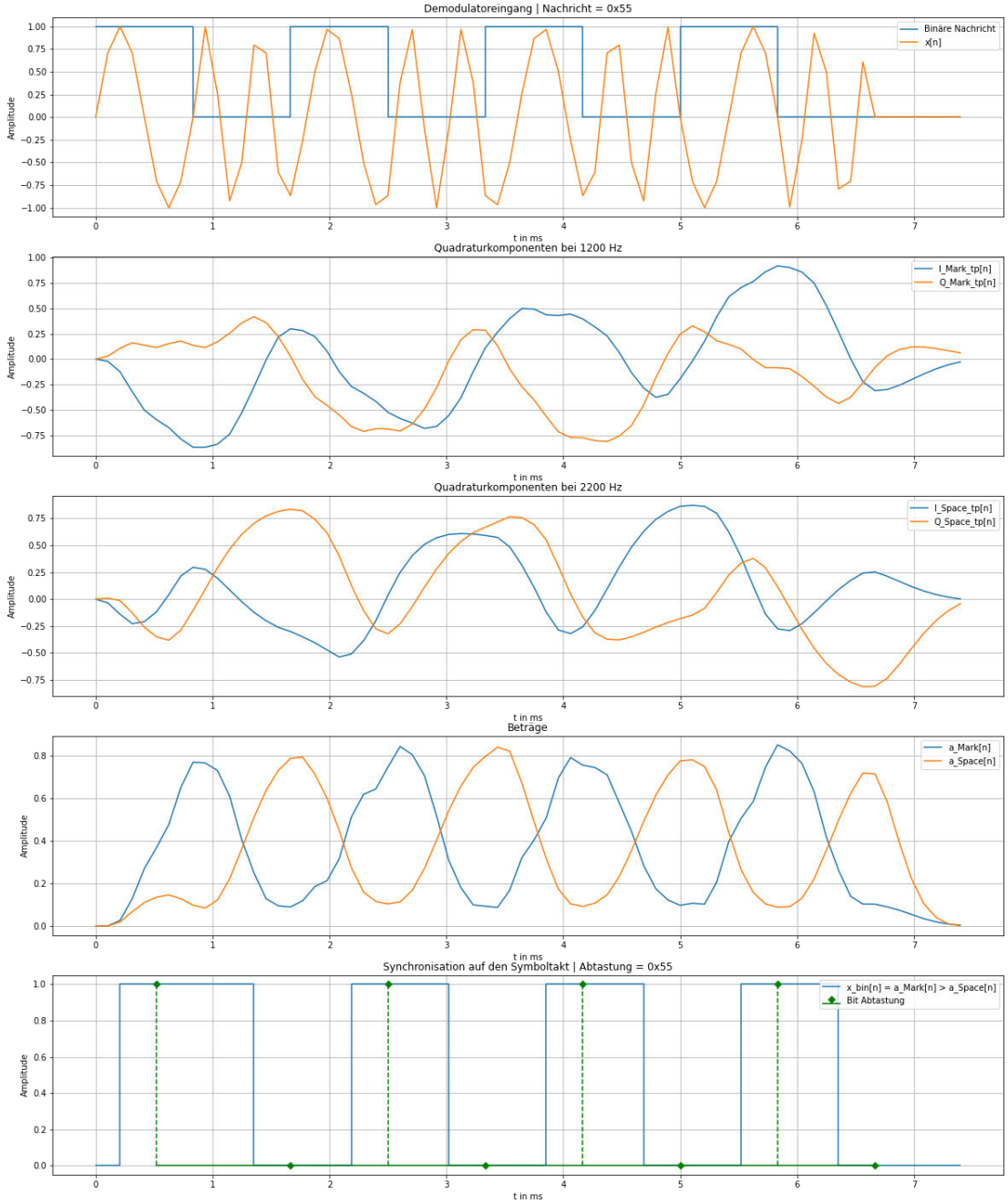

Abbildung 4.11: Simulation der Signale in dem Quadraturdemodulator mit der Abtastfrequenz $F_S = 9,6\,\mathrm{kHz}$

4.2.5 Auswahl des Demodulationsverfahrens

Für die Erfüllung der Anforderungen aus Kapitel 2 wird das am besten geeignete Demodulationsverfahren bestimmt. Die in Abschnitt 4.2 beschriebenen Demodulationsverfahren werden im Folgenden verglichen. Der Vergleich betrachtet den Rechenaufwand und die Robustheit der Verfahren.
Der Rechenaufwand wird durch die Anzahl der benötigten Rechenoperationen bestimmt. Zuweisungen und Vergleiche fließen in dieser Betrachtung nicht mit in den Rechenaufwand ein. Bei dieser Simulation wird für jedes Demodulationsverfahren der gleiche Tiefpassfilter genutzt. Da es sich um einen Vergleich zwischen den Verfahren handelt, wird der genutzte Tiefpassfilter als eine Tiefpassfilteroperation gewertet. So wird ein Vergleich auch für unterschiedliche Tiefpassfilterimplementierungen ermöglicht.
Die Robustheit der Verfahren gegenüber abweichenden Frequenzen wird durch die Variation von jeweils einer Frequenz untersucht. Tabelle 4.1 enthält die Ergebnisse der Simulationen.

Eigenschaften	Nullstellenzähler bei $F_S = 9,6\,\text{kHz}$	Delay-Line-Demodulator mit Multiplikator bei $F_S = 9,6\,\text{kHz}$	Delay-Line-Demodulator mit Exklusiv-Oder-Gatter bei $F_S = 19,2\,\text{kHz}$	Quadraturdemodulator bei $F_S = 9,6\,\text{kHz}$
Anzahl der Operationen	3 x Multiplikationen 1 x TP-Filter	1 x Multiplikation 1 x TP-Filter	1 x Exklusiv-Oder-Gatter 1 x TP-Filter	8 x Multiplikationen 2 x Additionen 4 x TP-Filter
Min. F_S	9,6 kHz	4,8 kHz	13,2 kHz	4,8 kHz
Min. f_{Mark}	1124 Hz	797 Hz	801 Hz	219 Hz
Max. f_{Mark}	1242 Hz	1350 Hz	1329 Hz	2024 Hz
Min. Toleranz f_{Mark}	±42 Hz	±150 Hz	±129 Hz	±824 Hz
Min. f_{Space}	2125 Hz	2093 Hz	2000 Hz	1554 Hz
Max. f_{Space}	2268 Hz	3360 Hz	3054 Hz	4633 Hz
Min. Toleranz f_{Space}	±68 Hz	±107 Hz	±200 Hz	±646 Hz
Min. f_{Baud}	1130 Hz	1130 Hz	1164 Hz	1130 Hz
Max. f_{Baud}	1280 Hz	1280 Hz	1238 Hz	1280 Hz
Min. Toleranz f_{Baud}	±70 Hz	±70 Hz	±36 Hz	±70 Hz

Tabelle 4.1: Übersicht der Demodulationsverfahren und ihrer Eigenschaften in der Simulation.

- Die 4 Demodulationsverfahren lassen sich aufgrund Ihrer Anforderungen an die Hardware in 2 Gruppen unterteilen. Der Delay-Line-Demodulator mit Multiplikator und der Quadraturdemodulator benötigen einen Analog-To-Digital-Converter (ADC). Der Delay-Line-Demodulator mit Exklusiv-Oder-Gatter und der Nullstellenzähler benötigen einen Komparator. Die verwendete Hardware aus Abschnitt 2.3 verfügt über die notwendigen Komponenten und ist für jedes der 4 Demodulationsverfahren geeignet.

- Der Nullstellenzähler weist in diesem Vergleich die geringste Toleranz gegenüber abweichenden Frequenzen auf. Die in Unterabschnitt 4.2.1 beschriebene Abnahme der Robustheit aufgrund weniger Nullstellen pro Symbol wird durch die Simulation bestätigt.

- Verfahren, die einen Komparator nutzen, weisen in dieser Simulation eine geringere Robustheit auf als jene, die einen ADC nutzen. Die Ursache hierfür ist vermutlich, dass die Verfahren mit einem Komparator weniger Informationen über das Signal nutzen können als die Verfahren mit einem ADC.
  Der Delay-Line-Demodulator mit Exklusiv-Oder-Gatter ist eine Variante des Delay-Line-Demodulators, die besonders für den Einsatz in Systemen geeignet ist, die für die Ausführung eine Multiplikation mehr Taktzyklen benötigen als für die Ausführung einer Exklusiv-Oder-Operation [18, S. 3-8].

- In den Simulationen ist der Quadraturdemodulator am robustesten. Der benötigte Rechenaufwand ist für den Quadraturdemodulator allerdings mit Abstand am höchsten. Der Delay-Line-Demodulator mit Multiplikator hat in diesem Vergleich das beste Verhältnis zwischen dem notwendigen Rechenaufwand und der System-Robustheit. Die in Unterabschnitt 3.5.2 beschriebenen Frequenztoleranzen werden von beiden Verfahren um mehr als das Vierfache erfüllt.

Der Delay-Line-Demodulator mit Multiplikator wird als Demodulationsverfahren für das Softmodem gewählt. Weitergehende Simulationen des Delay-Line-Demodulator mit Multiplikator haben gezeigt, dass er resistent gegenüber Signalen ist, die nicht mit einer Amplitude von 0 anfangen oder eine Phasenverschiebung aufweisen. Auch nicht sinusförmige Signale können demoduliert werden, da das Demodulationsverfahren auf der Korrelation zwischen positiven und negativen Signalamplituden beruht.
Bei der Festlegung der Abtastfrequenz muss beachtet werden, dass der Delay-Line-Demodulator in einer realen Umsetzung im Gegensatz zu der Simulation externen Störungen ausgesetzt ist. Um dies zu kompensieren, wird die Abtastfrequenz größer gewählt als

die in der Simulation bestimmte minimal notwendige Abtastfrequenz. Für die Umsetzung des Delay-Line-Demodulators wird in dieser Arbeit die Abtastfrequenz $F_\mathrm{S} = 9,6\,\mathrm{kHz}$ gewählt. Es ergeben sich somit 8 Abtastpunkte pro Symbol. Eine weitere Optimierung der Abtastfrequenz wird im Rahmen dieser Arbeit nicht vorgenommen.

4.3 Tiefpassfilterentwurf

Um Rechenaufwand einzusparen, wird eine Alternative für den IIR-Filter aus den in Abschnitt 4.2 beschriebenen Simulationen gesucht. Weitere Untersuchungen haben gezeigt, dass ein Exponential-Moving-Average (EMA)-Filter als Tiefpassfilter für den Delay-Line-Demodulator mit Multiplikator genutzt werden kann. Durch die Verwendung des EMA-Filters werden, im Vergleich zu dem IIR-Filter 2. Ordnung aus den Simulationen, 3 Multiplikationen und 3 Additionen pro Abtastpunkt eingespart. Der Amplitudengang des EMA-Filters ist weniger steil als der des IIR-Filters 2. Ordnung, jedoch ausreichend für die Anwendung in dem Demodulator.
Gleichung 4.23 zeigt die Differenzengleichung des EMA-Filters. Die Grenzfrequenz des Filters kann mit Gleichung 4.24 bestimmt werden. [16]

$$y[n] = \alpha \cdot x[n] + (1-\alpha) \cdot y[n-1] \tag{4.23}$$

$$f_{\mathrm{g\,3dB}} = arccos\left(\frac{\alpha^2 + 2\cdot\alpha - 2}{2\cdot\alpha - 2}\right) * \frac{F_\mathrm{S}}{2\pi} \tag{4.24}$$

Mit $\alpha = 0,4$ und $F_\mathrm{S} = 9,6\,\mathrm{kHz}$ folgt aus Gleichung 4.24 eine Grenzfrequenz $f_{\mathrm{g\,3dB}} = 798\,\mathrm{Hz}$, Abbildung 4.12 (grün). Die größte gewünschte Frequenz in dem demodulierten Signal ist 600 Hz, Abbildung 4.12 (rot). Die Frequenz tritt auf, wenn das modulierte Signal zwischen den Symbolen 0 und 1 alterniert, da dies bei einer Symbolrate von $1,2\,k$Baud wie ein Rechtecksignal mit $f = 600\,\mathrm{Hz}$ wirkt.

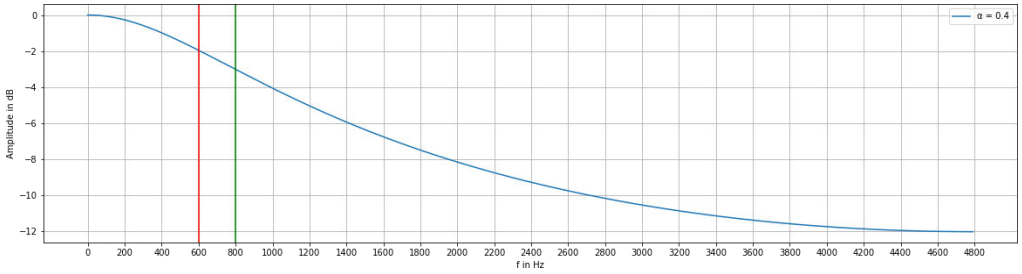

Abbildung 4.12: Amplitudengang des Exponential-Moving-Average-Filters

4.4 Synchronisation auf den Symboltakt

Der Abtastzeitpunkt der empfangenen Symbole muss mit dem Symboltakt synchronisiert werden. In dem demodulierten Signal werden die Symbole so synchronisiert, dass sie nach der halben Symboldauer abgetastet werden. Durch die Abtastung nach der halben Symboldauer wird die Erkennung der Symbole optimiert.
Die Synchronisation auf den Symboltakt wird durch einen Software-Timer realisiert [24]. Das demodulierte Signal wird hierzu von einem Komparator in ein binäres Signal umgewandelt, siehe $x_{\text{bin}}[n]$ in Abbildung 4.9.
Der Software-Timer zählt von der Anzahl der Abtastzeitpunkte pro Symbol abwärts. Die Symbole werden abgetastet, wenn der Zähler bei der Hälfte der Abtastzeitpunkte pro Symbol angekommen ist, siehe Abbildung 4.13 (c). Durch die gerade Anzahl der Abtastzeitpunkte pro Symbol wird entweder an dem Abtastzeitpunkt vor (grün) oder nach (blau) der halben Symboldauer abgetastet. Die Simulationen der Demodulationsverfahren in diesem Kapitel haben gezeigt, dass dies zu keiner fehlerhaften Erkennung der Symbole führt. Bei jedem Flankenwechsel des binären Signals zum Beispiel (b) und (e) wird der Zähler zurückgesetzt und so synchronisiert. Die Synchronisation funktioniert somit nach dem ersten Flankenwechsel des binären Signals (b). Vor diesem Ereignis ist die Abtastung in der Mitte des Symbols nicht sichergestellt (a). Erfolgt zwischen aufeinanderfolgenden Symbolen kein Flankenwechsel, erreicht der Software-Timer das Ende seines Zählintervalls und setzt sich wieder auf seinen Ausgangswert zurück (d).

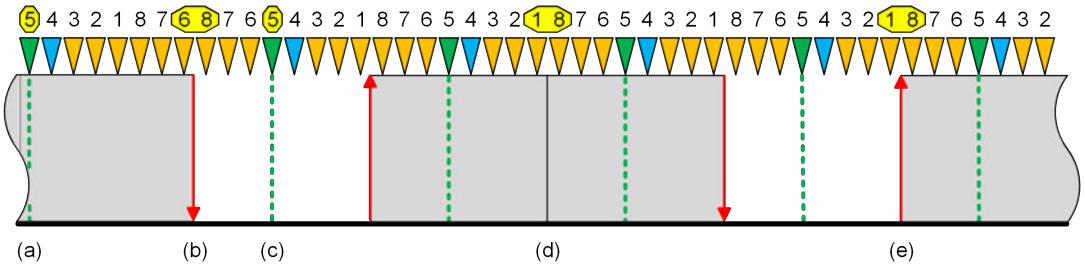

Abbildung 4.13: Synchronisation auf den Symboltakt mit 8 Abtastzeitpunkten pro Symbol

Aufgrund der in Unterabschnitt 3.5.3 beschriebenen Codierung der Daten als UART-Zeichen ist der maximale Abstand zwischen zwei Flanken des binären Signals durch das Start-Symbol auf 10 Symbolzeiten beschränkt, siehe Abbildung 4.14. Die erneute Synchronisation auf den Symboltakt erfolgt entsprechend spätestens nach 10 Symbolzeiten. Der Software-Timer wird so gegen einen Verlust der Synchronität aufgrund langer konstanter Symbolfolgen geschützt.

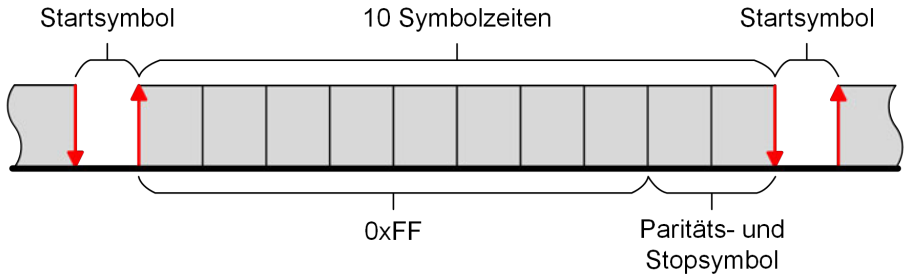

Abbildung 4.14: Maximaler Abstand zwischen zwei Flanken des binären Signals eines UART-Zeichens mit ungerader Parität

5 Umsetzung des Softmodems

Für die Erstellung des Programmcodes wird eine Integrated-Development-Environment (IDE) verwendet. Der Hersteller der STMicroelectronics (STM)32 Nucleo-144-Entwicklungs-Platine aus Abschnitt 2.3 bietet eine eigens für die Arbeit mit STM32-Mikrocontrollern optimierte IDE an. Die STM32CubeIDE [19] basiert auf Eclipse und ermöglicht unter anderem Peripheriekonfiguration und Codegenerierung.
Die im Folgenden beschriebenen Umsetzungen sind mit dem Ziel erstellt worden, möglichst viele Aufgaben in die Hardware-Peripheriegeräte des Mikrocontrollers auszulagern. Durch dieses Design wird gleichzeitig der benötigte Rechenaufwand und Jitter durch unterschiedliche Laufzeiten der Software gesenkt. Für die Interaktion mit den Hardware-Peripheriegeräten wird der herstellereigene Hardware-Abstraction-Layer (HAL) [22] genutzt.
Die STM32 Nucleo-144-Entwicklungs-Platine wird mit einer Taktfrequenz von 8 MHz betrieben. Als Taktquelle dient der interne Multi-Speed-RC-Oszillator des Mikrocontrollers [21, S.335]. Die Verwendung des internen RC-Oszillators führt verglichen mit einem externen Quarzoszillator zu Einsparungen im Energie- und Bauteilaufwand. Der RC-Oszillator weist über einen Temperaturbereich von 0 bis 85° C eine Abweichung von -3,5 % bis 3 % auf [20, S. 220]. Das Softmodem muss mit einer Genauigkeit von ±1 % arbeiten, siehe Unterabschnitt 3.5.2. Für den Testbetrieb des Softmodems bei Zimmertemperatur ist der RC-Oszillator geeignet, da er bei Zimmertemperatur ausreichend geringe Abweichungen aufweist. Vor der Implementierung des Softmodems in einer anderen Umgebung sollte aufgrund des starken Temperaturdrifts des RC-Oszillators evaluiert werden, ob eine Steigerung der Genauigkeit durch beispielsweise Verwendung eines externen Quarzoszillators zum Kalibrieren des RC-Oszillators notwendig ist.

5.1 Struktureller Aufbau des Softmodems

In Anlehnung an das in Abschnitt 3.1 beschriebene shannonsche Kommunikationsmodell sind die Funktionalitäten zur Nachrichtenübertragung in separate funktionale Module unterteilt, siehe Abbildung 5.1.
Das Sendemodul stellt eine Funktion zur Verfügung, die übergebene Daten in ein HART-konformes Signal umwandelt. Des Weiteren meldet sich das Sendemodul, wenn die übergebenen Daten vollständig gesendet wurden.
Eine von dem Empfangsmodul bereitgestellte Funktion wandelt das empfangene Signal in Bytes um. Das Empfangsmodul ist aktiv, bis eine Nachricht vollständig empfangen wurde, und informiert über den Erhalt einer neuen Nachricht.
Sende- und Empfangsmodul werden von dem Softmodem verwaltet. So ist sichergestellt, dass nicht beide gleichzeitig operieren. Dies ist notwendig, da bei einem gleichzeitigen Betrieb eine von dem Sendemodul gesendete Nachricht in dem Empfangsmodul nicht von einer externen Nachricht unterschieden werden kann. Es würde zu einer fehlerhaften Meldung über den Empfang einer neuen Nachricht kommen.

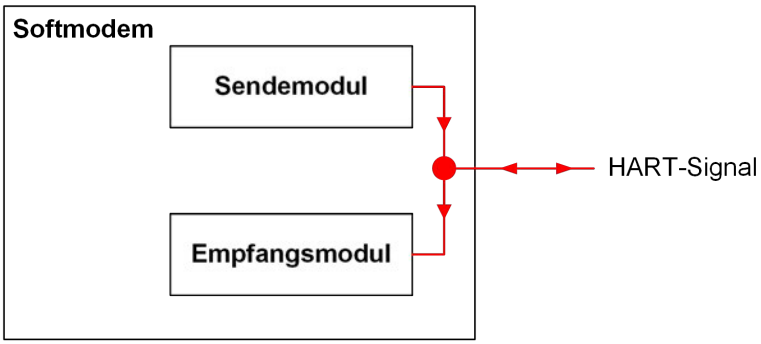

Abbildung 5.1: Blockschaltbild des strukturellen Aufbaus des Softmodems

5.2 Struktur der erstellten Software

Durch die STM32CubeIDE wird eine Projektstruktur erzeugt, siehe Abbildung 5.2. In dem Core-Ordner sind die Anwendungsdateien des Benutzers abgelegt. Der Drivers-Ordner stellt alle benötigten Treiber für den Mikrocontroller bereit. Das Projekt wurde um einen Softmodem-Ordner für die Dateien der Softmodemumsetzung erweitert. Das Softmodem ist, wie in Abschnitt 5.1 beschrieben, in drei Funktionsmodule in Form von Treibern unterteilt. Jedes dieser Funktionsmodule nutzt eine Header-Datei, um seine Schnittstellen zu definieren. Der Programmcode in der Quelltext-Datei des jeweiligen Funktionsmoduls wird durch seine Schnittstellen gekapselt. Die Separation des Quelltextes steigert die Wiederverwendbarkeit der einzelnen Treiber.

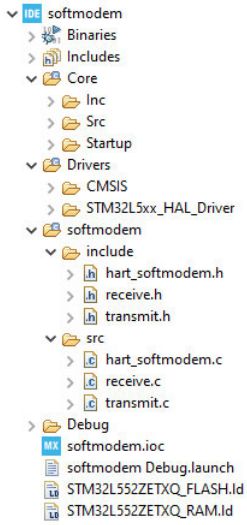

Abbildung 5.2: Projektstruktur des Softmodems

Alle für eine Instanz eines Funktionsmoduls notwendigen Variablen sind in einer statisch allozierten Datenstruktur gespeichert. So wird es unter anderem erleichtert, die Treiber nachträglich so zu erweitern, dass mehrere Softmodem-Instanzen parallel betrieben werden können. Mittels eines Pointers auf diese Datenstruktur werden die internen Zustände der Funktionsmodule verborgen und dem Benutzer gleichzeitig ermöglicht, mit der gewünschten Instanz des Moduls zu interagieren. Es handelt sich hierbei somit um einen opaken Datentyp.

5.3 Der Sendevorgang

Dieser Abschnitt beschreibt, wie während des Sendevorgangs zu übertragende Daten in ein physisches Signal nach Abschnitt 3.5 umgewandelt werden. Um eine Signalfolge zur digitalen Übertragung eines Datenrahmens zu erzeugen, werden drei Teilfunktionen benötigt:

1. Die zu sendenden Daten werden für die Übertragung in die Form eines HART-Datenrahmens gebracht.
2. Für die Übermittlung des Datenrahmens wird dieser durch das Sendemodul in eine Symbolabfolge umgewandelt.
3. Ein Symbol wird entsprechend des zu übermittelnden Symbolwerts moduliert.

Der Sendevorgang nutzt für die Umsetzung dieser drei Teilfunktionen Hardware-Peripheriegeräte des Mikrocontrollers und das Sendemodul. In dem Sendemodul befindet sich die Software, um den Ablauf des Sendevorgangs zu steuern.

5.3.1 Realisierung der Modulation

Das modulierte Signal soll an einem General-Purpose-Input-Output (GPIO) des Mikrocontrollers ausgegeben werden. Um an einem GPIO ein analoges Signal ausgeben zu können, muss dieser intern mit dem Ausgang eines DAC verbunden sein. Der DAC wandelt mit einer Frequenz von $F_\mathrm{S} = 38,4\,\mathrm{kHz}$ digitale Signalwerte in analoge Signalwerte um und gibt diese an seinem Ausgang aus. Mit dem DAC des Mikrocontrollers ist es möglich, bis zu ein Million-Samples-Per-Second (MSPS) umzuwandeln. Der DAC ist somit für die Anwendung mit $F_\mathrm{S} = 38,4\,\mathrm{kHz}$ geeignet [21, S. 798]. Die Umwandlung eines Signalwerts in dem DAC wird von einem Timer getriggert. Dieser Timer ist so eingestellt, dass er periodisch von einem festgelegten Wert abwärts zählt. Erreicht der Timer den Wert 0, wird der DAC getriggert und der Timer auf seinen Ausgangswert zurückgesetzt. Der Timer fungiert somit als Frequenzteiler und erzeugt einen periodischen Trigger für den DAC. Der Wert, auf den der Timer eingestellt werden muss, um den Trigger mit einer Frequenz von $F_\mathrm{S} = 38,4\,\mathrm{kHz}$ zu erzeugen, wird in Gleichung 5.1 bestimmt.

$$\left\lfloor \frac{f_\mathrm{CLK}}{F_\mathrm{S}} \right\rfloor - 1 = \left\lfloor \frac{8\,\mathrm{MHz}}{38,4\,\mathrm{kHz}} \right\rfloor - 1 = 208 - 1 \tag{5.1}$$

Zu beachten ist, dass der Wert um eins verringert werden muss, da der Timer auch die Null mitzählt. Durch die Rundung bei der Wertbestimmung in Gleichung 5.1 wird die Ausgangsfrequenz mit einem systematischen Fehler von 0,16 % erzeugt. Nach den Definitionen aus Unterabschnitt 3.5.2 ist eine Abweichung der Symbolfrequenz von 1 % erlaubt und die Abweichung von 0,16 % somit zulässig. Es ist jedoch notwendig, diese Abweichung bei der Auswahl des Oszillators für eine spätere Anwendung entsprechend zu berücksichtigen.

Der DAC erhält per Direct-Memory-Access (DMA) den jeweils nächsten Amplitudenwert, der umgewandelt werden soll. Nach jeder Umwandlung eines Werts durch den DAC wird das DMA-Modul getriggert und der Wert für die nächste Umwandlung bereitgestellt. Das DMA-Modul ist so eingestellt, dass es die Adressen eines definierten Speicherbereichs durchläuft und dabei jeweils den an der aktuellen Speicherstelle abgelegten Wert an den DAC transferiert. Erreicht das DMA-Modul das Ende des zugewiesenen Speicherbereichs startet es wieder an der Ausgangsadresse.
Bei einer Umwandlungsrate von $F_S = 38,4\,\text{kHz}$ des DAC und einer Symbolfrequenz von $1,2\,\text{kBaud}$ ergeben sich 32 Punkte pro Symbol. Das Sendemodul nutzt für die Bestimmung des Amplitudenwerts von jedem dieser 32 Punkte pro Symbol eine Implementierung des Sinusgenerators aus Unterabschnitt 4.1.1 in Festkommaarithmetik. Die von dem Sendemodul erzeugten Werte werden in einem Output-Buffer gespeichert. Ein Output-Double-Buffer ermöglicht es, dass das Sendemodul einen Teil des Output-Buffers mit den zuletzt erzeugten Amplitudenwerten befüllen und die Hardware gleichzeitig auf den anderen Teil des Output-Buffers zugreifen kann, ohne dass es zu einer Korruption der Daten in dem Output-Buffer kommt. In dem Output-Double-Buffer werden die Amplitudenwerte für zwei Symbole gespeichert. Während das Symbol aus dem aktiven Teil des Output-Buffers ausgegeben wird, kann der inaktive Teil mit den Werten für das nächste Symbol befüllt werden. Sobald das DMA-Modul das Ende eines Teils des Output-Double-Buffers erreicht, erzeugt es einen Interrupt-Request (IRQ), der von dem Sendemodul genutzt wird, um den Ablauf des Sendevorgangs zu steuern. In Abbildung 5.3 sind die in diesem Abschnitt beschriebenen Zusammenhänge grafisch veranschaulicht.

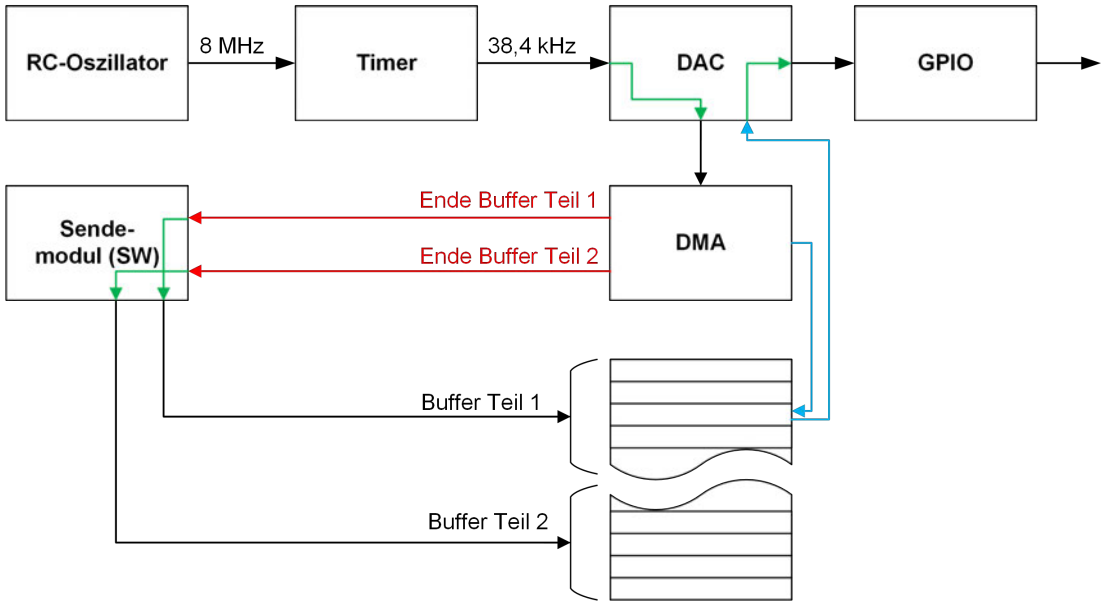

Abbildung 5.3: Funktionsschema der Modulation

Der DAC des Mikrocontrollers kann mit einer Auflösung von 8 oder 12 Bit betrieben werden. Aus der zeitlichen Diskretisierung des Sinussignals mit 32 Punkten pro Symbol resultieren Abstände zwischen den diskretisierten Amplitudenwerten des Signals. Bei einer Full-Scale-Spannung von $3,3\,\mathrm{V}$ sind die Schritte zwischen den Amplitudenwerten so groß, dass die Auflösung des LSB von $12,94\,\mathrm{mV}$ ausreichend genau ist.
Durch die Verwendung des DAC mit einer Auflösung von 8 Bit können die Werte der Lookup-Tabelle als 8 Bit-Variablen gespeichert werden. Im Vergleich zu dem 12 Bit-Betrieb des DAC, bei dem die Werte in 16 Bit-Variablen gespeichert werden, wird so die Hälfte des Speicherbedarfs der Lookup-Tabelle eingespart.

5.3.2 Ablauf des Sendevorgangs

Ziel dieses Abschnitts ist es, die Steuerung des Sendevorgangs durch das Sendemodul zu verdeutlichen. Das Sendemodul wird in diesem Abschnitt auch als Treiber bezeichnet. Anhand einiger Beispielfunktionsaufrufe wird die Anwendung des Sendemoduls beschrieben. Vor der ersten Ausführung eines Sendevorgangs muss das Sendemodul einmalig initialisiert werden. Die Initialisierung erfolgt durch den Aufruf der Initialisierungsfunktion in Listing 5.1. Der Rückgabewert der Initialisierungsfunktion ist der Handle für die Treiber-Instanz.

```
1 tx_handle = transmit_init();
```

Listing 5.1: Beispiel Initialisierung des Sendemoduls

Durch den Aufruf der Initialisierungsfunktion werden die Hardware-Peripheriegeräte und die Variablen in der Datenstruktur, auf die der Handle zeigt, initialisiert.

Der Sendevorgang wird durch einen Funktionsaufruf des Benutzers gestartet, siehe Listing 5.2. Die Sendefunktion bekommt den Handle auf die Treiber-Instanz, einen Pointer auf das Daten-Array, welches gesendet werden soll und die Anzahl der Elemente in dem Daten-Array übergeben. So ist es möglich, unterschiedlich große Datenmengen zu versenden, ohne dass der Treiber dauerhaft den Speicherplatz für die größtmögliche Menge zu versendender Daten vorhalten muss. Der letzte Parameter des Aufrufs ist der Pointer auf eine Callback-Funktion, die aufgerufen wird, nachdem die Daten vollständig versendet wurden. Der Rückgabeparameter der Funktion erlaubt es dem Benutzer zu überprüfen, ob die Anfrage für das Versenden der Daten akzeptiert oder abgelehnt wurde.

```
1 tx_status_value = transmit_data(tx_handle, data_buffer, data_buffer_size,
      transmit_callback_function);
```

Listing 5.2: Beispiel Funktionsaufruf für das Senden von Daten

Die Callback-Funktion ermöglicht es, dass der Benutzer den Sendevorgang startet und der Treiber im Non-Blocking-Mode die Daten versendet. Der Betrieb des Treibers im Non-Blocking-Mode ermöglicht die Nutzbarkeit des Softmodems parallel zu anderen Anwendungen auf dem Mikrocontroller. Sobald der Treiber die Übermittlung abgeschlossen hat, wird der Benutzer durch den Aufruf der übergebenen Callback-Funktion benachrichtigt.

Innerhalb des Treibers wird durch den Aufruf der Sendefunktion eine State-Machine gestartet, die in Abhängigkeit der Parametrisierung des Treibers und der übergebenen Daten den Double-Buffer mit den Amplitudenwerten für die ersten beiden Symbole befüllt. Anschließend startet die State-Machine die notwendigen Hardware-Peripheriegeräte, um die Daten nach dem im vorherigen Abschnitt beschriebenen Konzept zu versenden. Jeder IRQ des DMA-Moduls führt zu einem Zustandswechsel in der State-Machine. Mit jedem Zustandswechsel wird das nächste Symbol bestimmt und in dem inaktiven Teil des Double-Buffers abgelegt. Wenn alle Symbole auf diese Weise übermittelt wurden, schaltet die State-Machine die Hardware-Peripheriegeräte wieder ab und gibt den Treiber für die Übermittlung einer weiteren Nachricht frei.
Durch das Abschalten der Hardware-Peripheriegeräte wird Energie gespart und weitere IRQs vermieden.

5.4 Der Empfangsvorgang

Die in dem empfangenen Signal enthaltenen Daten sollen zurückgewonnen werden. Hierzu sind folgende Schritte notwendig:

1. Das empfangene 2-CPFSK-modulierte Signal muss demoduliert werden.
2. Um die Symbole in dem demodulierten 2-CPFSK-Signal korrekt zu erkennen, ist im Empfangsmodul eine Synchronisation auf den Symboltakt notwendig.
3. Aus der empfangenen Symbolabfolge werden Zeichen rekonstruiert.
4. Die rekonstruierten Zeichen werden zu einem Datenpaket zusammengesetzt.

Das Empfangsmodul enthält den Algorithmus für die Demodulation des empfangenen Signals sowie die Ablaufsteuerung für den Empfang einer Nachricht.

5.4.1 Realisierung der Demodulation

Digitalisierung des Signals

Das empfangene Eingangssignal wird durch einen ADC für die weitere Verarbeitung innerhalb des Mikrocontrollers digitalisiert. Der ADC tastet das Signal mit einer Abtastfrequenz von $F_\mathrm{S} = 9,6\,\mathrm{kHz}$ ab und wandelt die analogen Signalwerte in Zahlen mit einer Auflösung von 8 Bit um. Die Auflösung des ADC wird auf 8 Bit statt auf die maximal möglichen 12 Bit eingestellt, da dies zu einer Halbierung des Speicherbedarfs für alle Signalwerte des Demodulationsverfahrens führt. Für die Demodulation mit dem Delay-Line-Demodulator ist die Abtastung des Signalverlaufs mit einer Auflösung von 8 Bit ausreichend genau, siehe Unterabschnitt 4.2.5.
In dem Mikrocontroller ist ein Successive-Approximation (SAR)-ADC verbaut. Die für eine Umwandlung benötigte Zeit t_conv ist abhängig von der Zeit für die SAR t_SAR und der eingestellten Abtastdauer t_smpl, siehe Gleichung 5.2.

$$f_\mathrm{conv} = \frac{1}{t_{smpl} + t_{SAR}} \tag{5.2}$$

$$f_\mathrm{conv\,max} = \frac{1}{t_\mathrm{smpl\,min} + t_\mathrm{SAR\,8Bit}} = \frac{1}{312,5\,\mathrm{ns} + 1,06\,\mu\mathrm{s}} = 728,6\,\mathrm{kHz} \tag{5.3}$$

$$f_\mathrm{conv\,min} = \frac{1}{t_\mathrm{smpl\,max} + t_\mathrm{SAR\,8Bit}} = \frac{1}{80,06\,\mu\mathrm{s} + 1,06\,\mu\mathrm{s}} = 12,33\,\mathrm{kHz} \tag{5.4}$$

Bei einer Taktfrequenz von 8 MHz sind so je nach gewählter Abtastdauer Umwandlungsraten von bis zu $12,33\,\mathrm{kHz}$ bei Wahl der maximal möglichen Abtastdauer Gleichung 5.4 oder bis zu $728,6\,\mathrm{kHz}$ bei Wahl der minimal möglichen Abtastdauer Gleichung 5.3 erreichbar. Der ADC liegt mit $F_\mathrm{S} = 9,6\,\mathrm{kHz}$ unterhalb beider Umwandlungsraten und kann somit auf die maximal mögliche Abtastdauer eingestellt werden. [21, S. 700, 703, 719]
Um das empfangene Signal mit dem ADC umwandeln zu können, wird es mit einem GPIO verbunden. Der GPIO ist innerhalb des Mikrocontrollers an den Eingang eines Operational-Amplifier (OPAMP) angeschlossen. Der OPAMP dient als Impedanzwandler und ist wiederum mit dem Eingang des ADC verbunden.
Die Impedanzwandlung ist notwendig, da die maximal zulässige Anschlussimpedanz des ADC von der verwendeten Hardware aus Abschnitt 2.3 überschritten wird und deswegen gesenkt werden muss. Der ADC ist bis zu einer maximalen externen Impedanz von $50\,\mathrm{k\Omega}$ zugelassen [20, S. 239]. Die auf der EVAL-AD5421SDZ Entwicklungs-Platine angebrachte Filterschaltung für das empfangene HART-Signal weist eine Impedanz von mehr als

150 kΩ auf. Der Effekt der Impedanzwandlung auf das Eingangssignal des ADC wird durch die Messung in Abbildung 5.4 verdeutlicht. In dem linken Bild ist zu sehen, dass der ADC das Eingangssignal beeinflusst. Es sind pro Symbol 8 Artefakte zu erkennen. Die Artefakte ähneln Ladekurven und könnten durch die Sample-And-Hold-Schaltung des SAR-ADC hervorgerufen werden. Diese Vermutung wird bekräftigt, da bei einer Abtastfrequenz von $F_S = 9,6\,\text{kHz}$ jedes Symbol an 8 Stellen abgetastet wird. Das rechte Bild zeigt, dass dieses Problem durch die Impedanzwandlung behoben wird. Aufgrund dessen wird dieser Effekt in dieser Arbeit nicht weiter untersucht.

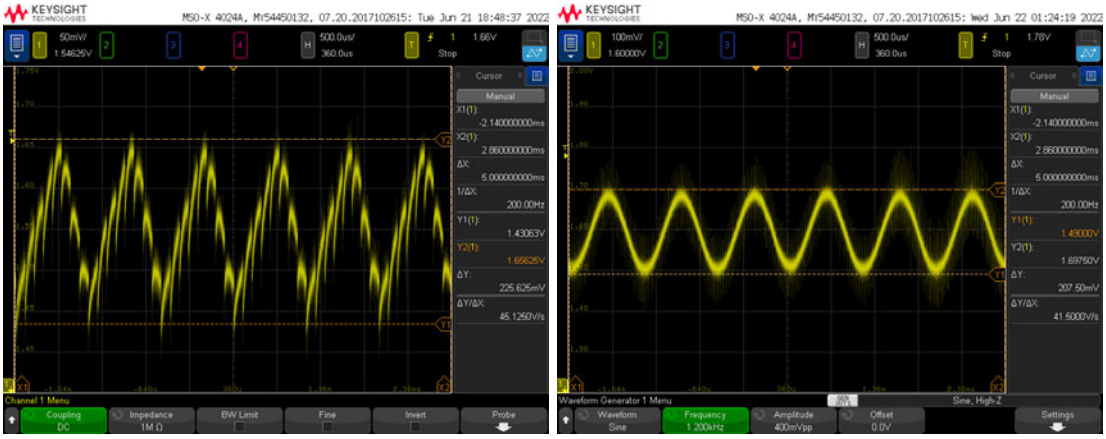

Abbildung 5.4: Eingangssignal des ADC, links ohne und rechts mit Impedanzwandler. $f = 1,2\,\text{kHz}$, ADC im 8 Bit-Modus mit $F_S = 9,6\,\text{kHz}$

Der ADC wird von einem Timer getriggert, der analog zu dem Timer aus Unterabschnitt 5.3.1 eingestellt wird. Aufgrund der abweichenden Abtastfrequenz des ADC von $F_S = 9,6\,\text{kHz}$ wird der Timer auf einen Wert von $833 - 1$ eingestellt.
Nachdem eine Umwandlung in dem ADC abgeschlossen wurde, wird der ermittelte Signalwert durch das DMA-Modul in den inaktiven Teil eines Input-Double-Buffers gespeichert. Die Einstellung des DMA-Moduls erfolgt ebenfalls analog zu Unterabschnitt 5.3.1 mit dem Unterschied, dass hier Daten im Input-Buffer abgelegt werden. Bei einer Abtastfrequenz von $F_S = 9,6\,\text{kHz}$ und einer Symbolfrequenz von $1,2\,\text{kBaud}$ ergeben sich 8 Abtastzeitpunkte pro Symbol. Der Input-Double-Buffer ist so ausgelegt, dass in jedem Teil des Input-Buffers die Werte eines Symbols abgelegt werden. Durch die erzeugten IRQs des DMA-Moduls kann das Empfangsmodul sicher auf den inaktiven Teil des Input-Double-Buffers zugreifen, um die aufgenommenen Daten aus dem Speicher zu lesen. Die

in diesem Abschnitt beschriebenen Verbindungen zwischen den eingesetzten Hardware-Peripheriegeräten werden in Abbildung 5.5 gezeigt.

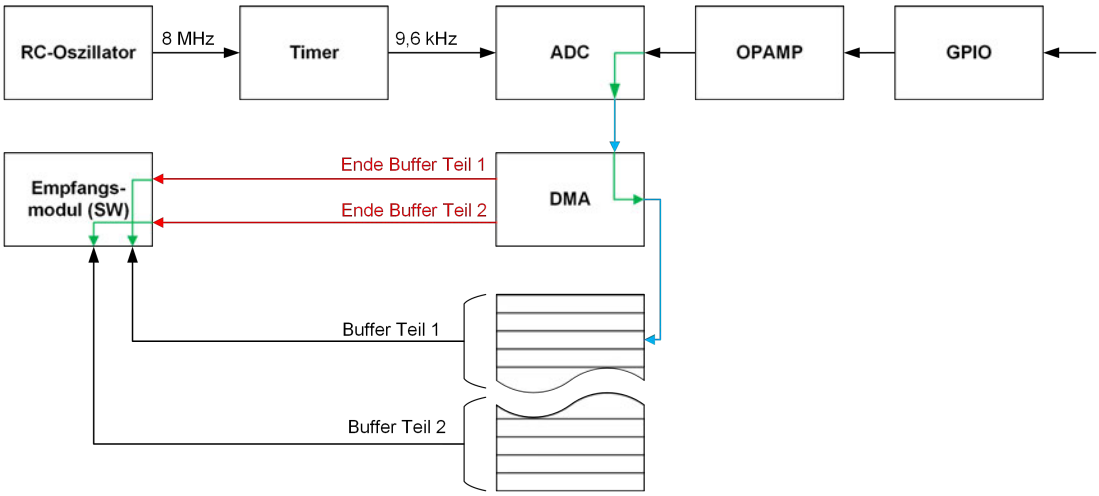

Abbildung 5.5: Funktionsschema der Demodulation

Datenverarbeitung

Dieser Abschnitt beschränkt sich auf die Besonderheiten der Implementierung. Die digitalisierten Signalwerte werden durch Anwendung des Delay-Line-Demodulators mit Multiplikator aus Unterabschnitt 4.2.2 demoduliert. Der Programmcode für die Demodulation ist in Festkommaarithmetik implementiert worden, um den zusätzlichen Rechenaufwand für die Verwendung von Gleitkommazahlen einzusparen.
Die optimale Verzögerung des Eingangssignals bei einer Abtastfrequenz von $F_S = 9,6\,\mathrm{kHz}$ wurde bereits im Grundlagenteil des Delay-Line-Demodulators mit Multiplikator mit $k = 4$ bestimmt. Diese Verzögerung wird in dem Empfangsmodul durch einen First-In-First-Out (FIFO)-Buffer realisiert. Da für die Berechnung der Demodulation immer auf den um vier Abtastzeitpunkte verzögerten Wert des Signals zugegriffen werden muss, kann ein vereinfachter Ringpuffer mit nur einem Zeiger als FIFO-Buffer genutzt werden. Bei dieser Variante des Ringpuffers wird ein Array mit vier Elementen verwendet. An einer der vier Speicherstellen kann der verzögerte Wert ausgelesen und anschließend der neue Wert gespeichert werden. Die Position in dem Array wird durch eine Variable, die zirkular von null bis drei zählt, bestimmt. So wird nach vier Schritten wieder auf den vor

vier Abtastzeitpunkten abgelegten Wert zugegriffen. Zur Veranschaulichung des Prinzips ist in Listing 5.3 der vereinfachte Ringpuffer als Pseudocode ausgeführt.

```
1 x_delayed = ring_puffer[index];
2 ring_puffer[index] = x_new;
3 index = (index+1) % 4;
```

Listing 5.3: Beispiel Implementierung des vereinfachten Ringpuffers

Nach der Demodulation hat das demodulierte Signal am Ausgang des Tiefpassfilters, wie in Unterabschnitt 4.2.2 beschrieben, die invertierte Form der Symbolabfolge. Durch einen Vergleich des Amplitudenwerts des demodulierten Signals mit einem konstanten Referenzwert wird entschieden, ob das Signal einer eins oder null zugeordnet wird. Der Vergleich mit dieser Entscheidungsgrenze wird so gewählt, dass die Invertierung der Symbolabfolge aufgehoben wird. Das so erzeugte binäre Signal kann anschließend, wie in Abschnitt 4.4 beschrieben, auf den Symboltakt synchronisiert abgetastet werden. Durch die synchronisierte Abtastung werden die empfangenen Symbole bestimmt.

5.4.2 Ablauf des Empfangsvorgangs

Der Treiber des Empfangsmoduls ist analog zu dem bereits beschriebenen Treiber des Sendemoduls aufgebaut. Nach der Initialisierung kann die Empfangsfunktion aufgerufen werden. Die Parameter der Empfangsfunktion entsprechen denen der Sendefunktion mit dem Unterschied, dass die Callback-Funktion dem Benutzer nicht nur einen Statuswert, sondern auch die Anzahl der empfangenen Bytes zurückgibt. Der Benutzer kann mit diesen Informationen gezielt auf den befüllten Teil des zuvor übergebenen Buffers zugreifen.

Der Aufruf der Empfangsfunktion startet eine State-Machine, die für die Steuerung des Empfangsvorgangs zuständig ist und den übergebenen Buffer mit den empfangenen Bytes befüllt. Durch diese State-Machine werden die notwendigen Hardware-Peripheriegeräte aktiviert.
Bei jedem IRQ des DMA-Moduls wird die Interrupt-Service-Routine (ISR) aufgerufen, die den abgetasteten Abschnitt des Signals demoduliert. Wenn von dieser Funktion ein Symbol erkannt wurde, wird eine zweite State-Machine aufgerufen, welche für die Bestimmung der übermittelten Zeichen in der Symbolabfolge zuständig ist.

Sobald ein Zeichen vollständig empfangen wurde, wird die State-Machine für den Empfangsvorgang getriggert und das Byte aus dem empfangenen Zeichen in dem Buffer gespeichert.
Die Hardware-Peripheriegeräte bleiben nach dem Aufruf der Empfangsfunktion aktiv bis eine Nachricht empfangen wurde. Der Anfang und das Ende einer Nachricht wird von der State-Machine für die Bestimmung der übermittelten Zeichen erkannt und entsprechend gemeldet. Der Empfangsvorgang endet nach der Meldung über das Ende der Nachricht mit dem Abschalten der Hardware-Peripheriegeräte und dem Aufruf der Callback-Funktion.

6 Messungen und Tests

Um den Erfüllungsgrad der Anforderungen aus Kapitel 2 zu bestimmen, werden in diesem Kapitel Tests durchgeführt. Diese Tests sollen durch Messung der zu überprüfenden Systemeigenschaften zeigen, ob die beschriebenen Anforderungen erfüllt wurden oder nicht. Für die Durchführung der Tests ist das Testsystem aus Abschnitt 2.3 notwendig. Sofern im Einzelfall nicht anders definiert gelten für die Tests die nachfolgenden Bedingungen.

Voraussetzungen:

1. Das Testsystem ist an 24 V Gleichspannung angeschlossen.
2. Auf der Stromschnittstelle wird ein Gleichstrom $\geq 4\,\mathrm{mA}$ gesendet.
3. Das USB-HART-Modem ist von einem Test-Computer aus steuerbar und parallel zu der 250 Ω-Bürde angeschlossen.
4. Ein Oszilloskop ist mit einem Tastkopf an dem als „LOOP-“ bezeichneten Pin der 4...20 mA Entwicklungs-Platine [1] angeschlossen und mit der Masse des Testsystems verbunden.

Testsequenz:

1. Das Oszilloskop ist auf Single-Shot eingestellt.
2. Mit dem USB-HART-Modem wird ein Frame mit der Nachricht 0xFFFFFFFFFF0000000000 gesendet, siehe Abbildung 6.1. Die Testnachricht ist so aufgebaut, dass am Anfang der Nachricht Mark-Symbole und am Ende der Nachricht Space-Symbole aufeinander folgen. Dies ermöglicht die Messung beider erzeugten Symbolfrequenzen, ohne eine Änderung der Testnachricht vorzunehmen.

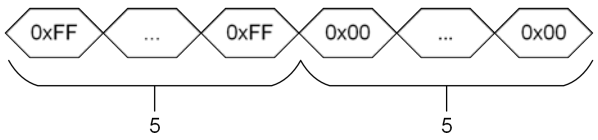

Abbildung 6.1: Testnachricht für die Messung beider Symbolfrequenzen

6.1 Taktfrequenz-Messungen

Die Taktfrequenz-Messungen dienen dazu, die Abweichungen in den von den Timern erzeugten Taktfrequenzen zu bestimmen, um deren Einfluss auf die Sende- und Empfangsfunktionen abschätzen zu können.
Es gelten folgende Änderungen der Bedingungen für die Messung der Taktfrequenzen.

Voraussetzungen:

4. Ein Output-Compare-Channel des zu messenden Timers wird an einem GPIO im Modus Toggle-On-Match ausgegeben.
5. Der Tastkopf des Oszilloskops wird an diesen GPIO angeschlossen und mit der Masse des Testsystems verbunden.

Testsequenz:

3. Mit dem Oszilloskop wird die Frequenz des Rechtecksignals bestimmt.

6.1.1 Taktfrequenz der Signalerzeugung

Testspezifikation

Zweck:

Bestimmung der Abweichung in der generierten Taktfrequenz des Timers für die Signalerzeugung. Die Ergebnisse können als Basis für einen Vergleich der Abweichungen in der Frequenz vor und nach der Signalerzeugung dienen.

Erwartetes Ergebnis:

Das Signal hat eine Frequenz von $f = 19,2\,\text{kHz} + 0,16\,\%$, die durch die Rundung bei der Einstellung des Timers entstehen.

Ergebnis

Das gemessene Signal hat eine Frequenz von $f = 19,291\,\text{kHz}$. Die Abweichung der Frequenz beträgt $+0,47\,\%$ und liegt somit $+0,31\,\%$ über dem erwarteten Ergebnis. Die Ursache der Abweichung ist vermutlich der verwendete RC-Oszillator. Um zu überprüfen, ob die Sendefunktionen des Softmodems durch diese Abweichungen eingeschränkt sind, werden im Folgenden weitere Tests durchgeführt.

6.1.2 Taktfrequenz der Signalabtastung

Testspezifikation

Zweck:

Bestimmen der Abweichung in der generierten Abtastrate, die der Timer erzeugt, um den Einfluss auf die Empfangsfunktionen einschätzen zu können.

Erwartetes Ergebnis:

Das Signal hat eine Frequenz von $f = 4,8\,\text{kHz} + 0,04\,\%$, die durch die Rundung bei der Einstellung des Timers entstehen.

Ergebnis

Das gemessene Signal hat eine Frequenz von $f = 4,8151\,\text{kHz}$. Die Abweichung von der geforderten Frequenz beträgt $+0,31\,\%$. Die theoretische Abweichung des Timers von $+0,04\,\%$ ist im Vergleich zu der gemessenen Abweichung gering. Die Vermutung aus dem vorherigen Test über eine Abweichung des RC-Oszillators von $+0,31\,\%$ wird durch diesen Test unterstützt.
Die Simulation in Unterabschnitt 4.2.5 hat ergeben, dass für den Delay-Line-Demodulator mit Multiplikator eine maximale Abweichung von $\pm 5,8\,\%$ in der Abtastfrequenz zulässig ist. Nach der ersten Einschätzung ist die erzeugte Abtastfrequenz genau genug.

6.2 Signalmessungen

Die Signalmessungen dienen der Überprüfung der von dem Softmodem erzeugten Signale.

6.2.1 Übertragene Wellenform des Mark-Symbols

Testspezifikation

Zweck:

Überprüfung der erzeugten Signalfrequenz auf die Einhaltung der spezifizierten Frequenz. Dieser Test ist notwendig, um NFA.2.1 zu überprüfen.

Testsequenz:

3. Mit dem Oszilloskop eine Periodenlänge des Signals während der Übertragung eines Mark-Symbols bestimmen.

Erwartetes Ergebnis:

Das Signal hat eine Frequenz von $f = 1,2\,\text{kHz} \pm 1\,\%$.

Ergebnis

Das gemessene Signal hat eine Frequenz von $f = 1208,02\,\text{Hz}$. Die Abweichung von der geforderten Frequenz beträgt $+0,67\,\%$. Das gemessene Signal erfüllt somit die Forderung der Spezifikation.

6.2.2 Übertragene Wellenform des Space-Symbols

Testspezifikation

Zweck:

Überprüfung der erzeugten Signalfrequenz auf die Einhaltung der spezifizierten Frequenz. Dieser Test ist notwendig um NFA.2.1 zu überprüfen.

Testsequenz:

3. Mit dem Oszilloskop wird eine Periodenlänge des Signals während der Übertragung eines Space-Symbols bestimmt.

Erwartetes Ergebnis:

Das Signal hat eine Frequenz von $f = 2,2\,\text{kHz} \pm 1\,\%$.

Ergebnis

Das gemessene Signal hat eine Frequenz von $f = 2196,8\,\text{Hz}$. Die Abweichung von der geforderten Frequenz beträgt somit $-0,15\,\%$. Das gemessene Signal erfüllt demnach die Forderung der Spezifikation.

6.2.3 Phasensprung in der übertragenen Wellenform

Testspezifikation

Zweck:

Überprüfung des Wechsels zwischen der erzeugten Signalfrequenz des Mark- und Space-Symbols ohne Phasensprung. Dieser Test dient der Überprüfung von FA.2.1 und NFA.2.1.

Testsequenz:

3. Mit dem Oszilloskop wird das Signal während des Übergangs von einem Mark-Symbol zu einem Space-Symbol und umgekehrt aufgenommen.

Erwartetes Ergebnis:

Die Frequenz des Signals ändert sich fließend, das heißt ohne einen Sprung in dem Verlauf der Amplitude.

Messungen

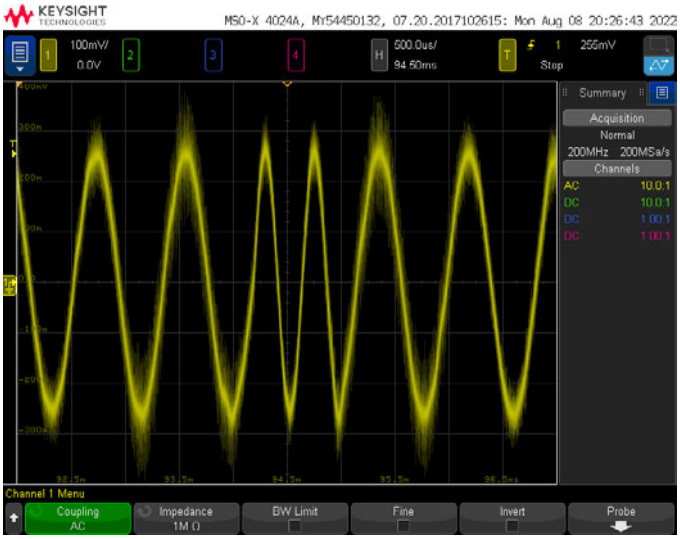

Abbildung 6.2: Messung des Phasensprungs in der erzeugten Signalfrequenz

Ergebnis

Das gemessene Signal in Abbildung 6.2 wechselt seine Frequenz von $1,2\,\mathrm{kHz}$ auf $2,2\,\mathrm{kHz}$ und wieder zurück ohne sichtbaren Phasensprung. Das gemessene Signal erfüllt somit die Forderung der Spezifikation.

6.3 Leistungstest

Mit den Leistungstests soll bestimmt werden, wie sich das System während des Betriebs verhält.

6.3.1 Beanspruchung der verfügbaren Rechenzeit

Testspezifikation

Zweck:

Bestimmen, wie viel der verfügbaren Rechenzeit des Mikrocontrollers von dem Softmodem während des Sende- und Empfangsvorgangs beansprucht wird. So soll eine Aussage über die Nutzbarkeit des Softmodems parallel zu anderen laufenden Anwendungen auf dem Mikrocontroller ermöglicht werden.

Voraussetzungen:

4. Der Programmcode des Softmodems wird so angepasst, dass ein GPIO-Pin am Anfang jeder ISR des DMA-Moduls eingeschaltet und am Ende der ISR wieder ausgeschaltet wird.
5. Der Programmcode des Softmodems wird mit dem GNU-Compiler-Collection (GCC)-Compiler und der Optimierungsstufe -O0 (Keine Optimierung des erzeugten Codes) kompiliert.
6. Der Tastkopf des Oszilloskops wird an diesen GPIO angeschlossen und mit der Masse des Testsystems verbunden.

Testsequenz:

3. Mit dem Oszilloskop wird jeweils eine Periodenlänge des Signals während des Sende- und Empfangsvorgangs aufgenommen und das Verhältnis der Impulsdauer zur Periodendauer bestimmt. Dieses Verhältnis entspricht der Beanspruchung der verfügbaren Rechenzeit des Mikrocontrollers in Prozent.

Erwartetes Ergebnis:

Weniger als 50 % der verfügbaren Rechenzeit werden beansprucht.

Ergebnis

Die gemessenen Beanspruchungen sind 20 % während des Empfangsvorgangs und 18, 75 % während des Sendevorgangs. Das erwartete Ergebnis von ≤ 50 % der verfügbaren Rechenzeit ist eine Annahme, da nicht bekannt ist, neben welchen anderen Anwendungen das Softmodem betrieben wird. Das festgestellte Ergebnis einer Beanspruchung von maximal 20 % der verfügbaren Rechenzeit während der Operation des Softmodems zeigt, dass das Softmodem auch neben anderen Anwendungen betrieben werden kann, wenn diese insgesamt weniger als 80 % der verfügbaren Rechenzeit benötigen.

6.3.2 Langzeit-Funktionstest

Testspezifikation

Zweck:

Überprüfen, mit wie vielen Fehlern das wiederholte Senden und Empfangen einer Testnachricht zwischen dem Softmodem und einem zertifizierten USB-HART-Modem erfolgt. So kann eine Einschätzung über die Robustheit des Sende- und Empfangsvorgangs getroffen werden. Des Weiteren werden die Anforderungen an den Empfang FA.2.3, FA.3.2 und FA.3.3 sowie das Senden FA.2.2 und FA.3.1 getestet. Auch die Anforderungen an die Kommunikation über die Hardware des Testsystems NFA.1.4 und NFA.3.1 können mit diesem Test überprüft werden. Durch diesen Test wird besonders die kritische Synchronisation auf den Anfang des empfangenen Signals untersucht.

Voraussetzungen:

4. Die Python Simulationsumgebung aus dem Anhang ist auf dem Test-Computer installiert.

Testsequenz:

1. In dem Simulationsordner wird eine Kommandozeile geöffnet.
2. Der Langzeittest mit 5000 Wiederholungen wird gestartet durch:

   ```
   poetry run python simulation\test_softmodem.py 5000
   ```

   Durch diesen Test wird automatisiert die durch ein Command-Line-Interface (CLI) übergebene Anzahl von Testnachrichten über das USB-HART-Modem gesendet und die Antwort des Softmodems auf Übereinstimmung untersucht. Die Testergebnisse werden von dem Testprogramm in der Datei `test_softmodem_log.txt`

dokumentiert. Die Testnachricht besteht aus der Folge $5 \cdot$ 0xFF, $150 \cdot$ 0x55, $2 \cdot$ 0xFE und $1 \cdot$ 0x00 und somit insgesamt 158 Bytes, siehe Abbildung 6.3.

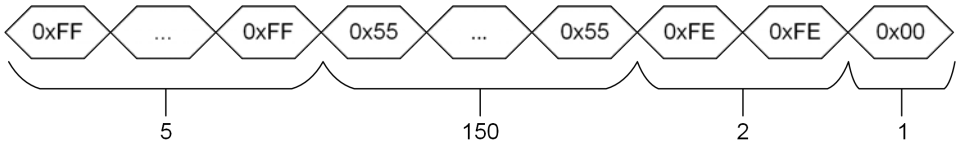

Abbildung 6.3: Aufbau der Testnachricht des Langzeit-Funktionstests

Erwartetes Ergebnis:

Die Spezifikation der physikalischen Schicht fordert eine Bit-Error-Rate (BER) von 0,01 % [7, S. 34].

Ergebnis

In 04 h 01 min 42,310 s wurden 5.000 Testnachrichten und somit 785.000 Bytes gesendet. 25 Testnachrichten und 3.758 Bytes wurden als fehlerhaft gewertet. Die Byte-Fehlerrate liegt bei 0,48 %.

Die Softmodem-Demonstratorversion ist so eingestellt, dass sobald ein fehlerhaftes Byte eingeht, der Empfang des gesamten Frames abgebrochen wird. Infolgedessen werden durch einen Fehler während des Empfangs eines Bytes alle anschließenden Bytes ebenfalls als fehlerhaft gewertet. Der Test ist darüber hinaus in der aktuellen Version nicht in der Lage, die Anzahl der Fehler in einem Byte zu bestimmen.

Die Testergebnisse sind lediglich als grobe Größenordnung zu werten. Die aktuelle Version des Tests benötigt Funktionen der pySerial Bibliothek [13], die von den Erstellern als unzuverlässig markiert wurden, um das USB-HART-Modem anzusteuern. In dem zeitlichen Rahmen dieser Arbeit ist es nicht möglich, den Test so weiterzuentwickeln, dass er belastbare Ergebnisse erzeugt. Eine erste Auswertung und Klassifizierung des Fehlerbilds der als fehlerhaft markierten Nachrichten deutet jedoch auf eine Schwäche der anfänglichen Synchronisation des Softmodems auf das empfangene Signal hin.

7 Diskussion der Ergebnisse und Ausblick

7.1 Bewertung der Umsetzung

Die Tests aus dem vorherigen Kapitel zeigen, dass die Umsetzung des Sendemoduls in der Lage ist, ein 2-CPFSK-Signal zu modulieren. Alle in Kapitel 2 definierten Muss- und Soll-Anforderungen an das Senden eines Signals sind erfüllt worden. Über die geforderten Anforderungen hinaus sind auch alle funktionalen Wunsch-Anforderung erfüllt worden. Im Rahmen dieser Arbeit war es nicht möglich zu überprüfen, ob das Sendemodul alle Anforderungen der FSK-Physical-Layer-Specification erfüllt. Aufgrund dessen wird die nicht-funktionale Wunsch-Anforderung NFA.3.4 als nicht erfüllt bewertet.
Die FSK-Physical-Layer-Specification schreibt eine Zeit für das Abklingen des erzeugten Signals nach dem Senden einer Nachricht vor. Für eine spätere Messung der Abklingzeit auf der verwendeten Hardware sei erwähnt, dass die Generierung des modulierten Signals an dem letzten erzeugten Amplitudenwert endet und es somit zu längeren und unterschiedlichen Abklingzeiten kommen kann, siehe Abbildung 7.1 (gelb). Abhilfe könnte eine Erweiterung schaffen, die das Signal am Ende einer Nachricht bis zu dem nächsten Nulldurchgang weiter moduliert und so zu einer Verkürzung der Abklingzeit führt (rot).

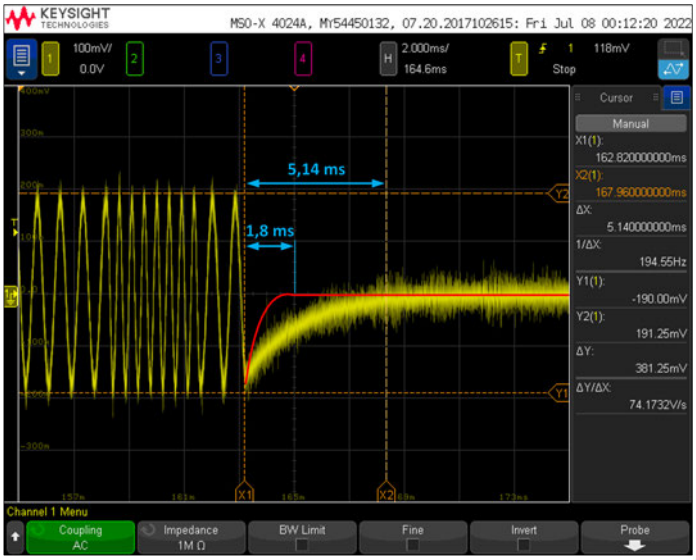

Abbildung 7.1: Beispiel Abklingzeiten eines Signals mit (rot) und ohne (gelb) Modulation bis zum Nulldurchgang am Ende einer Nachricht

Das Sendemodul ist so gestaltet worden, dass es selbstständig vor und nach einer zu sendenden Nachricht Carrier-Symbole hinzufügt. Die hinzugefügten Carrier-Symbole ermöglichen der Gegenstelle, sicher ein- und auszuschwingen und ersetzen die RTS-Funktion eines HART-Modem-ICs.
Es ist somit möglich, die Sendefunktionen eines HART-Modem-ICs mit dem Sendemodul zu ersetzen und Nachrichten an ein zertifiziertes USB-HART-Modem zu senden. Die Umsetzung des Sendemoduls ist insgesamt sehr gut gelungen.

Das Empfangsmodul des Softmodems benötigt in der aktuellen Version ein RTS-Signal in der Länge von zwei Symbolzeiten am Anfang und von einer Symbolzeit am Ende der Nachricht, um diese vollständig zu empfangen. Wenn das RTS-Signal für eine wie in der Physical-Layer-Specification beschriebene Zeit vor und nach der Nachricht gesendet wird, werden die Nachrichten somit vollständig empfangen. Das Testen des Empfangsmoduls wird erschwert, da für das USB-HART-Modem keine Software verfügbar ist, die automatisch die RTS-Funktion in geeigneter Weise steuert. Durch die Einschränkung in der Ansteuerung des USB-HART-Modems kommt es bei den Tests zu ungewollten Artefakten. Diese werden durch Anpassungen in der Fehlererkennung des Empfangsmoduls und die zusätzliche Übertragung eines Platzhalter-Bytes am Ende jeder Nachricht kompensiert.

Bei der Auswertung des Erfüllungsgrads der Zielsetzung muss beachtet werden, dass bei der Erstellung der Anforderungen noch nicht absehbar war, ob das Softmodem im Rahmen dieser Arbeit so weit entwickelt werden kann, dass die Umsetzung des Empfangsmoduls erfolgreich eine Nachricht empfängt.
Der Langzeit-Funktionstest in Unterabschnitt 6.3.2 zeigt, dass im Schnitt 1 aus 200 Nachrichtenübertragungen fehlschlägt und das Empfangsmodul eine Byte-Fehlerrate von $0,48\,\%$ aufweist. Für ein Testsystem, das in vergleichsweise kurzer Zeit entwickelt wurde, ist dies ein großartiger Erfolg. Die Anforderungen an das Empfangsmodul aus der Zielsetzung dieser Arbeit werden bis auf die Wunsch-Anforderung NFA.3.4 alle erfüllt.
Die Ergebnisse des Langzeit-Funktionstests können, wie in Unterabschnitt 6.3.2 beschrieben, nur als ungefähre Größenordnung dienen. Da der Langzeit-Funktionstest nicht die Anzahl der Fehler in einem Byte bestimmen kann, wird im Folgenden die gemessene Byte-Fehlerrate mit der geforderten BER der Spezifikation unter der Annahme von einem fehlerhaften Bit pro Byte verglichen. Die Spezifikation der physikalischen Schicht fordert eine BER von $0,01\,\%$ [7, S. 34]. Die Byte-Fehlerrate von $0,48\,\%$ ist im Vergleich zu der BER von $0,01\,\%$ ein schlechtes Ergebnis, da sie den geforderten Maximalwert um das 48-fache übersteigt. Es ist, wie in den Tests beschrieben, davon auszugehen, dass die tatsächliche BER des Empfangsmoduls geringer ist. Des Weiteren bietet sich noch viel Potenzial, um das bisherige Ergebnis zu verbessern.
Die Fehlererkennung des Empfangsmoduls ist sehr strikt und bricht bei der Erkennung eines Fehlers in einem Bit den gesamten Empfangsvorgang ab. Dies ist notwendig, da das Empfangsmodul aus der empfangenen Symbolfolge erkennen muss, wann das Ende einer Nachricht erreicht ist. Die Logik für die Erkennung des Endes einer Nachricht muss viele Sonderfälle beachten und funktioniert deswegen nur korrekt, wenn das vorangegangene Zeichen in der Nachricht valide war. Durch eine zusätzliche Hardwareschaltung für die Erkennung des Carriers könnte auf diese Erkennung in der Software verzichtet werden. Dieses würde die Software robuster gegen Fehler machen und zu einer besseren Les- und Wartbarkeit des Programmcodes beitragen.
Das Fehlerbild des Langzeit-Funktionstests deutet darauf hin, dass eine Verbesserung der Synchronisation auf den Anfang der Nachricht großes Potenzial bietet, um die Fehlerrate zu verbessern. Das Empfangsmodul wurde ohne detaillierte Kenntnis der zu empfangenden Nachricht entwickelt, um Abhängigkeiten zu minimieren und um es auch in anderen Anwendungen einsetzen zu können. Aufgrund dessen nutzt es nicht aktiv das Wissen über den Inhalt der Präambel, um sich auf den Inhalt des demodulierten Signals zu synchronisieren. Passiert bei der Synchronisation ein Fehler und es wird ein falsches Symbol des Signals als Start-Symbol interpretiert, kann die gesamte Nachricht nicht

korrekt empfangen werden. Durch das Wissen über den Inhalt der Präambel könnte das erste empfangene Zeichen mit dem erwarteten Zeichen der Präambel verglichen werden. Wenn diese nicht übereinstimmen, können weitere Synchronisationsversuche unternommen werden, bis die Synchronisation erfolgreich ist.
Des Weiteren kann durch eine Optimierung der Entscheidungsgrenze, die das demodulierte Signal einem der beiden Symbolwerte zuordnet, die Wahrscheinlichkeit für die korrekte Erkennung der Symbole weiter gesteigert werden.
Überdies bietet die verwendete Hardware weiteres Potenzial für eine Verbesserung der BER. Das auf der verwendeten Hardware empfangene Signal des USB-HART-Modems weist ein nicht zu vernachlässigendes Rauschen auf, siehe Abbildung 7.2. Die Signalamplitude des empfangenen Signals ist um etwa das Vierfache größer als die Amplitude des empfangenen Rauschens.

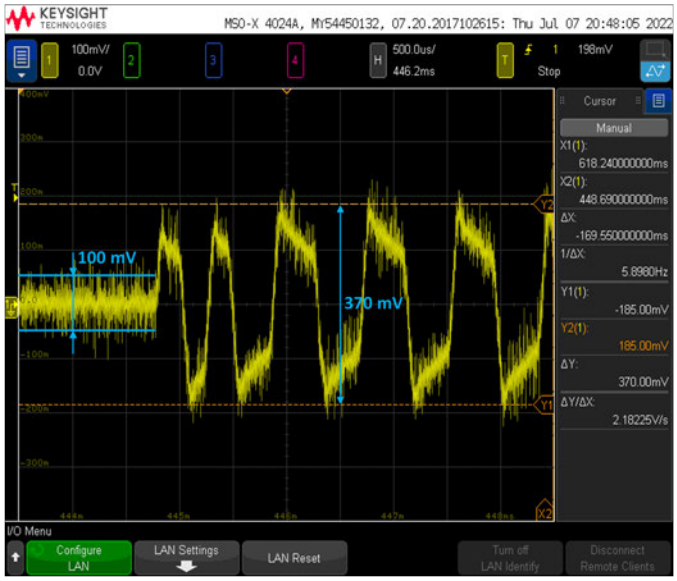

Abbildung 7.2: Auf der verwendeten Hardware empfangenes Signal des USB-HART-Modems

Der gemessene Abstand zwischen den Amplituden des Signals und des Rauschens lässt ein geringes SNR erwarten. Im Rahmen der weiteren Entwicklung des Softmodems sollte der Hardwareaufbau des Testsystems angepasst werden, um das SNR und damit auch die BER zu verbessern.
Es ist also realistisch anzunehmen, dass die BER des Empfangsmoduls noch signifikant verbessert werden kann.

Abschließend kann gesagt werden, dass die aus den Simulationen gewählte Demodulationsmethode des Empfangsmoduls sehr gut funktioniert. Auch die trapezförmigen Signale des USB-HART-Modems Abbildung 7.2 werden von ihr korrekt demoduliert. Die Empfangsfunktionen eines HART-Modem-ICs können mit dem Empfangsmodul in der aktuellen Version zwar noch nicht zuverlässig ersetzt werden, Nachrichten eines zertifizierten USB-HART-Modems werden jedoch schon jetzt meistens richtig empfangen.

7.2 Fazit

Das Ziel dieser Arbeit war es zu untersuchen, ob der Funktionsumfang eines HART-Modem-ICs durch ein Softmodem auf einem Mikrocontroller übernommen werden kann, und eine Einschätzung über das Potenzial dieses Konzepts zu geben.

Durch die Simulation verschiedener Modulations- und Demodulationsverfahren wurde es ermöglicht, in einem direkten Vergleich das am besten geeignete Verfahren zu bestimmen. Die anschließende Implementierung der ausgewählten Verfahren in einer ersten Umsetzung des Softmodems hat gezeigt, welche Verbesserungen in der folgenden Entwicklung noch notwendig sein werden. Vor allem wurde jedoch deutlich, dass das Potenzial für das im Rahmen dieser Arbeit untersuchte Konzept sehr groß ist.
Der verwendete Mikrocontroller bietet durch eine Vielzahl bereitgestellter Hardware-Peripheriegeräte die Möglichkeit, den Anschluss an eine 4...20 mA Stromschnittstelle ohne zusätzliche aktive Bauteile umzusetzen. Darüber hinaus wurde in Unterabschnitt 6.3.1 gezeigt, dass das Softmodem auf dem verwendeten Mikrocontroller maximal 20 % der verfügbaren Rechenzeit für den Sende- und Empfangsvorgang beansprucht. Das Softmodem kann somit auch neben anderen Softwareanwendungen auf dem Mikrocontroller betrieben werden und ist daher produktiv nutzbar. Die erstellte Umsetzung wird aktuell mit einer Taktfrequenz von nur 8 MHz betrieben, was den Einsatz des Softmodems auch für Low-Power-Anwendungen denkbar macht. Ein Mikrocontroller hat sich als eine sehr geeignete Plattform für den Einsatz des Softmodems erwiesen.

7.3 Ausblick

Aufgrund der zeitlichen Beschränkung dieser Arbeit konnten nur die Funktionen des Softmodems umgesetzt werden, die für die Beantwortung der Forschungsfrage notwendig waren. Für die Verwendung des Softmodems in einer realen HART-Kommunikation ist es jedoch notwendig, dass das Softmodem um eine Funktion erweitert wird, die vor dem Senden einer Nachricht überprüft, ob die Nachricht gesendet werden kann, ohne eine aktive Kommunikation zu stören. In der Bewertung der Umsetzung wurde bereits vorgeschlagen, eine zusätzliche Hardwareschaltung für die Erkennung des Carriers zu nutzen, um die Fehler des Empfangsmoduls zu reduzieren. Diese Hardwareschaltung könnte auch genutzt werden, um zu überprüfen, ob eine Nachricht auf die Leitung gesendet werden darf oder nicht. Des Weiteren wäre es durch diese Schaltung möglich, den ADC zwischen dem Empfang einzelner Nachrichten zu deaktivieren und so den Energiebedarf zu senken.
Die Verbesserungen, die einen direkten Einfluss auf die Qualität der Kommunikation haben, wurden bereits in der Bewertung der Umsetzung vorgestellt. Deswegen wird an dieser Stelle auf mögliche Verbesserungen in der Gestaltung der Treiber eingegangen.
Die Parameter der Module, wie zum Beispiel die Anzahl der Abtastpunkte pro Symbol, werden durch Definitionen in den Header-Dateien der Module festgelegt. Eine Änderung dieser Parameter hat aktuell keinen Effekt auf die Einstellung der Hardware-Peripheriegeräte. Eine Anpassung der Hardware-Peripheriegeräte-Einrichtung, sodass diese von den Definitionen in den Header-Dateien der Module abhängig ist, könnte hier eine potenzielle Fehlerquelle ausräumen.
Während des Sende- und Empfangsvorgangs wird, wie in der Umsetzung beschrieben, jeweils ein Symbol in einem Teil des Double-Buffers abgelegt. Es kommt somit nach jedem Symbol zu einem IRQ des DMA-Moduls. Durch eine Vergrößerung des Double-Buffers könnten mehrere Symbole in jeden Teil des Double-Buffers gelegt und so die Anzahl der IRQs gesenkt werden. Es gilt hier zwischen dem Speicherbedarf und der Anzahl der IRQs zu priorisieren.
Die Treiber sind so gestaltet, dass sie im Non-Blocking-Mode operieren. Nach einer IRQ werden in der ISR alle Berechnungen durchgeführt, die bis zur nächsten IRQ notwendigen sind. Die Zeit, in der die ISR aktiv ist, sollte jedoch so kurz wie möglich gehalten werden. Eine zusätzliche Funktion, die in der Hauptschleife des Programms aufgerufen wird und dort die Berechnungen tätigt, die nicht unbedingt in der ISR durchgeführt werden müssen, könnte die Ausführungszeit der ISR verkürzen.

Literaturverzeichnis

[1] Analog Devices, Inc.: *Evaluation Board User Guide UG-250, Evaluation Board for 16-Bit, Serial Input, Loop-Powered 4 mA to 20 mA DAC.* 2011. – URL https://www.analog.com/media/en/technical-documentation/user-guides/UG-250.pdf. – Zugriffsdatum: 26.07.2022. – Rev. A

[2] Analog Devices, Inc.: *SDP User Guide UG-277, SDP-B Controller Board.* 2011. – URL https://www.analog.com/media/en/technical-documentation/user-guides/UG-277.pdf. – Zugriffsdatum: 26.07.2022. – Rev. B

[3] Analog Devices, Inc.: *Low Power HART Modem AD5700/AD5700-1.* 2016. – URL https://www.analog.com/media/en/technical-documentation/data-sheets/AD5700_5700-1.pdf. – Zugriffsdatum: 26.07.2022. – Rev. G

[4] ARM Limited: *RealView Development Suite: AXD and armsd Debuggers Guide, ARM DUI 0066G.* Version 3.0. ARM Limited, 2006

[5] Baudoin, G ; Virolleau, F ; Venard, O ; Jardin, P: *Teaching DSP through the Practical Case Study of an FSK Modem.* September 1996. – URL https://www.ti.com/lit/an/spra347/spra347.pdf. – Zugriffsdatum: 16.07.2022

[6] FieldComn Group: Token-Passing Data Link Layer Specification HCF_SPEC-081. Austin, USA : FieldComn Group, Mai 2012. – Standard. – URL https://store.fieldcommgroup.org/collections/hart/products/hrt-spec. Rev. 9.0

[7] FieldComn Group: FSK Physical Layer Specification HCF_SPEC-054 FCG TS20054. Austin, USA : FieldComn Group, Mai 2016. – Standard. – URL https://store.fieldcommgroup.org/collections/hart/products/hrt-spec. Rev. 9.1

[8] InfoTech, Vector: *HART USB Interface Adapter for Notebook / PC - HART-USB-002*. 2020. – URL https://www.vectorinfotech.com/FileStore/Full/_Product99142_ViT_Hart_Modem_HART-USB-002.pdf. – Zugriffsdatum: 25.07.2022

[9] ISO: *Information technology - Open Systems Interconnection - Basic reference model: The basic model, ISO/IEC 7498-1:1994-11*. November 1994

[10] Jorge Torres Gómez, Fidel E. Hernández Montero, Joachim Habermann: *DEMODULATORS FOR BFSK SIGNALS BASED ON MATCHED FILTERS: A SURVEY*. 2016. – URL https://revistatelematica.cujae.edu.cu/index.php/tele/article/download/229/212/670. – Zugriffsdatum: 23.07.2022

[11] Karrenberg, Ulrich: *Signale - Prozesse - Systeme: eine multimediale und interaktive Einführung in die Signalverarbeitung*. 7., neu bearbeitete und erweiterte Auflage. Berlin [Heidelberg] : Springer Vieweg, 2017. – ISBN 9783662526590 9783662526583

[12] Knuth, D.E.: *Structured Programming with Goto Statements*. 6. Stanford University. Computer Science Department, 1974

[13] Liechti, Chris: *pyserial: Python Serial Port Extension*. – URL https://github.com/pyserial/pyserial. – Zugriffsdatum: 11.08.2022

[14] Mishra, Kanchan: *CPFSK Demodulation Techniques*. 2007. – URL https://kanchan-iitg.tripod.com/sitebuildercontent/sitebuilderfiles/btpthesis.pdf. – Zugriffsdatum: 21.07.2022

[15] NAMUR: *Signalpegel für die Ausfallinformation von digitalen Messumformern mit analogem Ausgangssignal und dessen sichere Erkennung, NAMUR NE 43:2021-07-26*. Juli 2021

[16] P, Pieter: *Exponential Moving Average*. 2020. – URL https://tttapa.github.io/Pages/Mathematics/Systems-and-Control-Theory/Digital-filters/Exponential%20Moving%20Average/Exponential-Moving-Average.html. – Zugriffsdatum: 25.07.2022

[17] Plassmann, Wilfried: *Handbuch Elektrotechnik: Grundlagen und Anwendungen für Elektrotechniker*. 6., neu bearb. Aufl. Wiesbaden : Springer Vieweg, 2013 (Handbuch Elektrotechnik). – ISBN 9783834820716

[18] Seguine, Dennis: *Simplified FSK Detection, Document No. 001-15224*. April 2017. – URL https://www.infineon.com/dgdl/Infineon-AN2336_PSoC_1_Simplified_FSK_Detection-ApplicationNotes-v07_00-EN.pdf?fileId=8ac78c8c7cdc391c017d07237cdd46c0. – Zugriffsdatum: 21.07.2022. – Rev. *F

[19] STMicroelectronics: *STM32CubeIDE - Integrated Development Environment for STM32 - STMicroelectronics*. – URL https://www.st.com/en/development-tools/stm32cubeide.html. – Zugriffsdatum: 19.08.2022

[20] STMicroelectronics: *Datasheet - STM32L552xx*. Oktober 2020. – URL https://www.st.com/resource/en/datasheet/stm32l552cc.pdf. – Zugriffsdatum: 03.08.2022. – Rev. 6

[21] STMicroelectronics: *Reference manual STM32L552xx and STM32L562xx advanced Arm®-based 32-bit MCUs*. Dezember 2020. – URL https://www.st.com/resource/en/reference_manual/dm00346336-stm32l552xx-and-stm32l562xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf. – Zugriffsdatum: 03.08.2022. – Rev. 7

[22] STMicroelectronics: *Description of STM32L5 HAL and low-layer drivers, UM2659*. August 2021. – URL https://www.st.com/resource/en/user_manual/um2659-description-of-stm32l5-hal-and-lowlayer-drivers-stmicroelectronics.pdf. – Zugriffsdatum: 19.08.2022. – Rev. 2

[23] STMicroelectronics: *User manual STM32L5 Nucleo-144 board (MB1361) UM2581*. Mai 2022. – URL https://www.st.com/resource/en/user_manual/um2581-stm32l5-nucleo144-board-mb1361-stmicroelectronics.pdf. – Zugriffsdatum: 26.07.2022. – Rev. 5

[24] Tran, Matthew: *STM32 APRS*. Januar 2021. – URL https://github.com/dragonlock2/STM32_APRS. – Zugriffsdatum: 16.07.2022

[25] Werner, Martin: *Nachrichtentechnik: eine Einführung für alle Studiengänge*. 8., vollständig überarbeitete und erweiterte Auflage. Wiesbaden [Heidelberg] : Springer Vieweg, 2017 (Lehrbuch). – ISBN 9783834825810 9783834825803

A Anhang

Der Anhang zur Arbeit befindet sich auf CD und kann beim Erstgutachter eingesehen werden.
In dem Anhang enthalten sind:

1. Die PDF-Version der Arbeit.
2. Die erstellten Simulationen.
3. Die erstellte Softmodem-Demonstrationsversion.
4. Die aufgenommenen Messdaten der Messungen und Tests.

Erklärung zur selbstständigen Bearbeitung

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