Entwurf eines robusten drahtlosen Kommunikationssystems für die industrielle Automatisierung unter harten Echtzeitbedingungen auf Basis von Ultrawideband-Impulsfunk

Von der Fakultät für
Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

Doktors der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

vorgelegt von
Johannes Hund, M.Sc.
geboren am 10.11.1981 in Fürstenfeldbruck

Gutachter: Prof. Dr.-Ing. Rolf Kraemer
Gutachter: Prof. Dr. Theo Vierhaus
Gutachter: Prof. Dr. Adam Wolisz

Tag der mündlichen Prüfung: 29.05.2012

Danksagung

Mein besonderer Dank gilt meinem Doktorvater, Prof. Rolf Kraemer. Er leitete und beriet mich in vielen Diskussionen stets ehrlich, respektvoll und auf Augenhöhe. Er scheute weder zeitlichen noch finanziellen Aufwand um mir bei meiner Arbeit eine optimale Unterstützung zu gewähren, die ich stets als besonderes Privileg wahrnahm.

Weiterhin danke ich Dr. Christian Schwingenschlögl, der es mir nicht nur ermöglichte, diese innovative Forschungsarbeit in der Industrie zu betreiben, sondern mir auch stets den Rücken freihielt, sodass ich mich voll auf die Forschung konzentrieren konnte.

Ein herzliches Dankeschön auch an die Forschungsteams von IHP Microelectronics und Siemens Corporate Technology. Besonders hervorheben will ich, seitens Siemens, den Einsatz von Dr. Andreas Heinrich, der mir in den ersten Jahren der Promotion in zahllosen Diskussionen als Sparring-Parter mit wertvollen Tipps beiseite stand, sowie Dr. Rainer Sauerwein für die „Grundsteinlegung“ dieses gemeinsamen Forschungsvorhabens. Weiterhin danke ich Michael Bahr und Dr. Alejandro Ramirez, die mir in späteren Phasen der Promotion wertvolle Diskussionspartner waren. Seitens IHP Microelectronics geht mein Dank besonders an Dr. Olonbayar und Dan Kreiser für die harmonische Zusammenarbeit auch über die große (organisatorische und geographische) Entfernung hinweg.

Besten Dank auch an Stephan Huckenholz und Dmitryo Krush, den beiden Studenten, deren Diplomarbeit bzw. Praktikum ich betreuen durfte. Ihre tatkräftige Unterstützung bei der Implementierung des Prototyps war ein Quell von vielen neuen Ideen und Lösungen.

Ich will mich an dieser Stelle auch bei meinen treuen Gefährten bedanken, die mir schon immer beistanden. Vielen Dank auch an die unzähligen helfenden Hände, die es mir ermöglichten, all dies trotz meines Handicaps zu erreichen.

Niemals genug werde ich jedoch meiner Freundin Daniela Högner und meiner Familie danken können. Seid euch sicher, dass ich es immer zu schätzen wusste und weiß, wie viel ihr für mich auf euch nehmt und wie viel Verständnis, Unterstützung und Geduld ihr mir entgegenbringt.

Abstract

Goal of this thesis was to investigate the suitability of Ultra-Wideband Impulse Radio (IR-UWB) for wireless communication in the sensor-actuator layer of industrial factory automation by designing and evaluating a wireless communication system based on this technology. Another requirement was that the system should be using standardized protocols or products wherever possible.

The application scenario results in hard real-time constraints in the order of few milliseconds. Those are to be fulfilled in rough environments with high noise figures and many metallic objects, that are causing a lot of multipath effects.

Thus, the primary subjects were to reduce latency and improve robustness for the existing IR-UWB based standard IEEE 802.15.4a. A very appealing feature of that standard is that it allows the use of very cheap, low complexity non-coherent devices.

It was discovered, that it is possible to improve the robustness of transmissions while still using these low-cost, low-complexity devices with only few additions to the existing standard. Several contributions to the state of the art of low-cost, low-complexity receiver design for IR-UWB and changes in modulation and coding were made to enhance the robustness of the existing standard.

To use the resulting improved PHY layer efficiently, an exisiting specialized MAC layer for factory automation, as described in the draft standard IEEE 802.15.4e, was used. This MAC layer enabled further application specific cross-layer improvements that exploit the known communication pattern to provide a lower latency and higher resilience.

The resulting communication system is configurable to adopt to different automation applications and provides at least equal and partly better performance than existing wire-bound or wireless solutions for the sensor-actuator level of industrial factory automation, like e.g. AS-Interface, Bluetooth I/O or WISA. For the representative case of 32 slave nodes with one byte of payload data each, the system provides a cyce time of 1.88 ms. It therefore allows to fulfill a hard realtime boundary of 15 ms with an error probability below 10^{-9}.

During the work, a highly configurable simulator for IR-UWB based on industry-grade channel models has been developed to verify the results. The communication system was implemented in hardware based on FPGA. Since the RF-frontend was not ready for test, the verification of MAC and baseband functionality was acomplished using a baseband back-to-back test method.

UWB, Impulse Radio, Automation, Wireless

Zusammenfassung

Ziel dieser Dissertation war es, die Eignung von Ultra-Breitband-Pulsfunk (IR-UWB) für die drahtlose Kommunikation in der Sensor/Aktor-Ebene der Fertigungsautomatisierung zu untersuchen. Dazu wurde ein drahtloses Kommunikationssystem auf Basis standardisierter Protokolle entworfen und untersucht.

Diese Anwendung erfordert die Erfüllung harter Echtzeitfähigkeit im Bereich weniger Millisekunden in industriellen Umgebungen. Ein solches Umfeld stellt aufgrund eines hohen Rauschpegels und vieler metallischer Oberflächen, die Multipfad-Effekte verursachen, sehr hohe Ansprüche an das Latenzverhalten und die Robustheit.

Deshalb waren die Hauptziele die Reduzierung von Latenz und gleichzeitige Erhöhung der Robustheit für den existierenden, auf IR-UWB basierenden Standard IEEE 802.15.4a. Dieser Standard ist unter anderem deshalb vielversprechend, da er sich mit sehr preisgünstigen, nicht-kohärenten Empfängern von geringer Komplexität umsetzen lässt, die trotzdem relativ robust gegenüber Multipfad-Effekten sind.

Es wurde gezeigt, dass sich auch mit diesen günstigen Geräten durch Optimierung des Standards eine hohe Übertragungssicherheit bei geringer Latenz realisieren lässt.

Es wurden Modifikationen zur Optimierung der Robustheit und Latenz des bisher üblichen Designs von nicht-kohärenten IR-UWB-Empfängern vorgestellt. Durch Ergänzungen zu der im Standard beschriebenen Modulation und Kodierung konnte die Verlässlichkeit zusätzlich gesteigert werden.

Um diese optimierte PHY-Schicht effizient einsetzen zu können, wurde eine spezialisierte MAC-Schicht für die Automatisierung, die in einem Entwurf für den kommenden Standard IEEE 802.15.4e beschrieben wird, eingesetzt. Da bei dieser MAC-Schicht die Kommunikationsmuster im Voraus bekannt sind, war weitere schichtübergreifende, applikations-spezifische Optimierung möglich, die eine weitere Reduzierung der Latenz sowie eine Erhöhung der Robustheit erbrachte.

Im Zuge der Arbeit wurde ein flexibel konfigurierbarer Simulator für IR-UWB auf Basis von industriell akzeptierten Kanalmodellen erstellt. Dieser Simulator wurde auch zur Evaluation und Verifikation der Forschungsergebnisse benutzt.

Das entworfene Gesamtsystem ist über mehrere Parameter konfigurierbar und dadurch an weitere Anwendungen in der Automatisierung anpassbar.

Eine dieser Konfigurationen wurde durch Simulation evaluiert. Sie zeigt mindestens die gleiche und zum Teil bessere Performance als bisherige drahtlose oder drahtgebundene Lösungen für die Sensor/Aktor-Ebene der Fertigungsautomatisierung, wie z.B. AS-Interface, Bluetooth I/O oder WISA. Für den repräsentativen Fall von 32 Teilnehmern mit jeweils einem Byte Prozessdaten erreicht sie eine Zykluszeit von 1,88 ms. Damit kann eine eine mittlere Reaktionszeit von 985 μs erreicht werden und eine harte Echtzeitschranke von 15 ms mit einer Fehlerwahrscheinlichkeit unter 10^{-9} eingehalten werden.

Das Kommunikationssystem wurde auch in Hardware auf FPGA-Basis implementiert. Da das benutzte analoge Front-End, ein früher Prototyp eines IEEE 802.15.4a-kompatiblen Front-Ends, noch keine repräsentativen Messungen zuließ, wurde die Funktion durch eine Basisbandverbindung über Kabel verifiziert.

UWB, Pulsfunk, Automatisierung, drahtlos

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation 2

1.1.1 Automatisierungstechnik 2

1.1.1.1 Prozessautomatisierung 3

1.1.1.2 Fertigungsautomatisierung 4

1.1.1.3 Gebäudeautomatisierung 5

1.1.2 Drahtlose Automatisierung 5

1.1.3 Warum UWB? 7

1.2 Problemstellung 9

1.3 Ergebnisse 10

1.4 Aufbau der Dissertation 11

2 Stand von Wissenschaft und Technik **13**

2.1 Kriterien 13

2.1.1 Mittlere Reaktionszeit 13

2.1.2 Telegrammfehlerrate / -latenz 14

2.1.3 Restfehlerrate 14

2.1.4 Datenrate 15

2.1.5 Koexistenzfähigkeit 15

2.2 Stand der Technik 16

2.2.1 Kommunikation in der Fertigungsautomatisierung 16

2.2.1.1 IO-Link 17

2.2.1.2 AS-Interface 19

2.2.2 Drahtlose Automatisierung in der Fertigung 22

2.2.2.1 WISA 22

2.2.2.2 Wireless HART 24

2.2.2.3 Industrial WLAN 27
2.2.3 Ultra-Breitband-Kommunikation 28
2.2.3.1 Herkunft und Geschichte von UWB 28
2.2.3.2 Das UWB-Prinzip 29
2.2.3.3 Regulierungsvorschriften bei UWB 31
2.2.3.4 OFDM-basierte UWB-Kommunikation 36
2.2.3.5 Impulsfunk 38
2.2.3.6 Modulation von Impulse Radio 41
2.2.3.7 Time Hopping 42
2.2.3.8 Pulse Repetition Rate 43
2.2.3.9 Die UWB-PHY-Schicht des Standards IEEE 802.15.4a 44
2.2.3.10 Schätzung der Reichweite von IR-UWB 51
2.3 Stand der Wissenschaft 58
2.3.1 Industrielle drahtlose Automatisierung 58
2.3.2 Ultra-Wideband-Forschung 61
2.3.2.1 Lokalisierung 62
2.3.2.2 Body Area Networks 63
2.3.2.3 Time Reversal 63
2.4 Zusammenfassung 64

3 Definition eines parametrierbaren Modells zur Simulation einer UWB-IR-Übertragung gemäß IEEE 802.15.4a 65
3.1 Aufbau eines Transceivers für IEEE 802.15.4a 66
3.1.1 Sender 66
3.1.1.1 Kodierung 66
3.1.1.2 Präambel-Erzeugung 66
3.1.1.3 Modulation 66
3.1.1.4 Analoges Front-End des Senders 67
3.1.2 Nichtkohärenter Energie-Detektions-Empfänger 67
3.1.2.1 Analoges Front-End des Empfängers 67
3.1.2.2 Präambel-Detektion 67
3.1.2.3 Daten-Demodulation 68
3.1.2.4 Dekodierung 68
3.2 Simulationsmodell der Übertragung 68

3.2.1 Sender 69
3.2.1.1 Enkodierung 69
3.2.1.2 Präambel-Erzeugung 70
3.2.1.3 Modulation 71
3.2.1.4 Pulsformung 73
3.2.2 Hochfrequenz-Ebene 73
3.2.2.1 Hochmischen 74
3.2.2.2 Kanal 75
3.2.2.3 Heruntermischen 81
3.2.3 Empfänger 81
3.2.3.1 Verstärker 81
3.2.3.2 Integrator 83
3.2.3.3 ADC 83
3.2.3.4 Präambel-Detektion 84
3.2.3.5 Daten-Demodulation 85
3.2.3.6 Dekodierung 85

4 Defintion und Analyse eines robusten UWB-Systems für die drahtlose Automatisierung 89

4.1 Definition der Anforderungen und der Zielapplikation 90
4.1.1 Knotentypen 90
4.1.1.1 Controller 90
4.1.1.2 Sensor 91
4.1.1.3 Aktor 91
4.1.2 Kommunikationsverhalten 92
4.2 Modellierung einer MAC-Schicht für harte Echtzeitanforderungen 92
4.2.1 Zeit- und Synchronisationsanforderungen 93
4.2.1.1 Polling 94
4.2.1.2 Multi-Polling 95
4.2.1.3 TDMA mit Zeitschlitzadressierung 97
4.2.1.4 Rollenbasiertes TDMA 97
4.2.1.5 Vergleich 98
4.2.2 Definition der MAC-Schicht auf Basis von Draft IEEE 802.15.4e 98
4.2.2.1 Superframe 99

4.2.2.2 Frame . . . 100
4.2.2.3 Beacon . . . 101
4.2.2.4 Knoten . . . 104
4.2.2.5 Controller . . . 104
4.2.2.6 Adressvergabe . . . 105
4.3 Optimierung einer robusten PHY-Schicht auf UWB-Basis . . . 105
4.3.1 Modellierung einer PHY-Schicht für IR-UWB in industriellen Anwendungen . . . 106
4.3.1.1 Kohärentes und nichtkohärentes Empfangen . . . 106
4.3.1.2 Einzelpulse und Pulsfolgen . . . 107
4.3.1.3 Modulation . . . 107
4.3.1.4 Synchronisation . . . 107
4.3.2 Adaption der UWB-PHY-Schicht von IEEE 802.15.4a . . . 108
4.3.2.1 Hopping-Positionen . . . 108
4.3.2.2 Burst-Länge . . . 109
4.3.2.3 Präambel-Wiederholungen . . . 110
4.3.2.4 L-Spreading . . . 110
4.3.2.5 Start of Frame Delimiter . . . 111
4.3.2.6 Kodierung . . . 112
4.3.2.7 Zusammenfassung . . . 114
4.3.3 Maßnahmen zur Erhöhung der Robustheit der PHY-Schicht . . . 114
4.3.3.1 Mehrpulsige Präambel-Modulation . . . 114
4.3.3.2 Überlappende Addition der Samples . . . 119
4.3.3.3 Soft-Bit-Informationen . . . 124
4.3.3.4 Vorzeichenbehaftete Energiedetektion . . . 127
4.3.3.5 Phasenrichtige Addition der Samples in der Präambel-Erkennung . . . 131
4.3.3.6 Gesamtverbesserung gegenüber der Referenz . . . 137
4.4 Cross-Layer-Verbesserungen . . . 138
4.4.1 Zwei-Stufen-Synchronisation . . . 138
4.4.2 Soft-Bit-Kombination . . . 141
4.5 Zusammenführung in ein verbessertes Zielsystem . . . 147
4.5.1 Konfigurierbarkeit der Präambel . . . 148

4.5.2 Konfigurierbarkeit des Datenteils . . . 149
4.5.3 Konfigurierbarkeit des Daten-Frame . . . 149
4.5.4 Konfigurierbarkeit des Beacon . . . 150
4.5.5 Konfigurierbarkeit des Superframe . . . 150
4.6 Beispielkonfiguration: TAWASI . . . 151
4.6.1 Latenz . . . 152
4.6.2 Restfehlerrate . . . 153
4.6.3 Code-Effizienz . . . 154
4.6.4 Datenrate . . . 155
4.6.5 Mittlere Reaktionszeit . . . 156
4.6.6 Telegrammfehlerrate/-latenz . . . 156
4.6.7 Energiebudget . . . 160
4.6.8 Vergleich . . . 162

5 Implementierung eines Hardware-Prototypen 163
5.1 Applikationsschicht . . . 164
5.2 MAC-Schicht . . . 164
5.2.1 Scheduler . . . 165
5.2.2 MAC-Logikeinheit . . . 166
5.2.2.1 Konfigurations-Zustandsautomat . . . 166
5.2.2.2 Datenfluss-Zustandsautomat . . . 166
5.3 Basisband-Prozessor . . . 166
5.3.1 Sender . . . 167
5.3.2 Empfänger . . . 167
5.4 HF-Front-End . . . 168
5.5 Maßnahmen zur Verifikation der Implementierung . . . 169
5.5.1 Hardware-Simulation . . . 170
5.5.2 „Back-to-Back“-Verifikation im FPGA . . . 170
5.5.3 Gesamtsystem . . . 171
5.6 Zusammenfassung . . . 172

6 Zusammenfassung 173
6.1 Zusammenfassung . . . 173
6.2 Ergebnisse . . . 173
6.3 Diskussion . . . 175

6.4 Zukünftige Arbeiten . . . 175

A Details der Implementierung des Hardware-Prototypen 177

A.1 Software . . . 178
A.2 Pseudo-Prozessor . . . 178
A.2.1 Befehle . . . 178
A.3 MAC-Schicht . . . 180
A.3.1 CPU-Interface . . . 180
A.3.2 Scheduler . . . 182
A.3.3 MAC-Logikeinheit . . . 182
A.3.3.1 Konfigurations-Zustandsautomat . . . 183
A.3.3.2 Datenfluss-Zustandsautomat . . . 184
A.3.4 Sende- und Empfangseinheit . . . 184
A.4 Basisband-Prozessor . . . 184
A.4.1 Sender . . . 184
A.4.1.1 PHY-Header . . . 185
A.4.1.2 Datenkodierung . . . 186
A.4.1.3 Präambel-Generator . . . 186
A.4.1.4 Scrambler . . . 187
A.4.1.5 Symbolformer . . . 187
A.4.1.6 Frontcore . . . 188
A.4.2 Empfänger . . . 189
A.4.2.1 Präambel-Detektion und -Synchronisation . . . 189
A.4.2.2 Demodulation . . . 190
A.4.2.3 Auslesen des PHY-Headers . . . 190
A.4.2.4 Dekodierung . . . 190
A.5 HF-Front-End . . . 191
A.5.1 Sender . . . 191
A.5.2 Empfänger . . . 191
A.6 Ausblick . . . 191

B Verzeichnisse 193

Glossar . . . 193
Abkürzungsverzeichnis . . . 195
Liste der Symbole . . . 200

Eigene Publikationen 203
Patente 205
Literaturverzeichnis 207

Abbildungsverzeichnis

1.1 Beispiel einer Multipfadausbreitung eines Signals 8
1.2 Einfluss von Multipfad-Effekten auf das Signal 9

2.1 Ebenen der industriellen Automatisierung[7] 17
2.2 Nachrichtenaustausch im AS-Interface-Protokoll [7] 20
2.3 Verteilung von Zeitschlitzen und Frequenzen bei WISA [43] . . . 23
2.4 Beispiele für Graphen in wireless HART [58] 26
2.5 Spektrale Masken für UWB-Kommunikation der *Handheld*- und *Indoor-Device*-Klasse nach der FCC-Regulierung 34
2.6 Spektralmaske der ETSI für UWB-Kommunikation 35
2.7 Maximale Spitzenwerte nach Bundesnetzagentur 37
2.8 Einteilung in Bandgruppen nach ECMA 368 [81] 37
2.9 Gaußsche Monozyklen in der Zeitdomäne [10] 39
2.10 Gaußsche Monozyklen in der Frequenzdomäne [10] 40
2.11 Spektrale Darstellung einer UWB-Übertragung ohne Zufallsmechanismen [10] 42
2.12 Spektrale Darstellung einer UWB-Übertragung mit Zufallsmechanismen [10] 43
2.13 Multiple Access mit Time Hopping 43
2.14 Kanäle in IEEE 802.15.4a 46
2.15 FEC-Kodierungsschema nach IEEE 802.15.4a [3] 47
2.16 Strutur der Prämbel nach IEEE 802.15.4a [23] 48
2.17 IEEE 802.15.4a Symbol-Frame 49
2.18 Empfangener Signalpegel gegen Entfernung für IEEE 802.15.4a . 54
2.19 Störfrequenzen verschiedener industrieller Prozesse [70] 56
2.20 Verlauf der SNR über die Entfernung für IEEE 802.15.4a 57

3.1 Schematische Darstellung des Senders 69
3.2 Referenzpulsform des Standards IEEE 802.15.4a 73
3.3 Schematische Darstellung des Hochfrequenz-Teils 74
3.4 Darstellung des simulierten HF-Signals eines einzelnen Pulses . . 75
3.5 Kanalmodell 1: Wohnhaus LOS 77
3.6 Kanalmodell 2: Wohnhaus NLOS 78
3.7 Kanalmodell 7: Industrie LOS 79
3.8 Kanalmodell 8: Industrie NLOS 80
3.9 Schematische Darstellung des Empfängers 82

4.1 Zustände der MAC-Schicht 92
4.2 Angenommener Paketaufbau für den theoretischen Vergleich der Zykluszeiten verschiedener MAC-Ansätze 95
4.3 Anufbau des Beacons im Multipolling-Ansatz 96
4.4 Vergleich der Zykluszeiten der verschiedenen Kommunikationsschemata 99
4.5 Superframe im Regelbetrieb 100
4.6 Zustandsdiagramm eines Knoten 104
4.7 Anteil der erfolgreich übermittelten und der verlorenen Pakete, aufgeteilt nach dem Grund des Paketverlustes 115
4.8 Gegenüberstellung der Signalstärke der Präambel und des Datenteils bei einer Präambel mit Einzelpulsen 116
4.9 Gegenüberstellung der Signalstärke der Präambel und des Datenteils bei einer Präambel mit Bursts aus 4 Pulsen 117
4.10 Verlängertes L-Spreading bei Präambel-Bursts 118
4.11 Veränderung der Rate nicht erkannter Präambeln durch mehrpulsige Präambeln 119
4.12 Bit-Ermittlung im Datenteil 120
4.13 Vergleich der Brutto-Bit-Fehlerrate bei fehlerhafter Synchronisation mit und ohne überlappendem Sampling 124
4.14 Vergleich zwischen harter Entscheidung mit RS-Code und Soft-Bit-basiertem Faltungs-Code 126
4.15 Autokorrelation von fünf Präambel-Symbolen mit binärem Code ohne Vorzeichendetektion 128

4.16 Autokorrelation von fünf Präambel-Symbolen mit ternärem Code mit Vorzeichendetektion 128
4.17 Schematische Darstellung eines Empfängers mit Vorzeichendetektion 129
4.18 Rate von nicht erkannten Präambeln mit und ohne vorzeichenbehafteter Präambel-Erkennung 130
4.19 Rate von Präambel-Fehlern mit und ohne vorzeichenbehafteter Präambel-Erkennung bei vier Pulsen pro Präambel-Sub-Symbol 131
4.20 Vergleich der Rate verlorener Präambeln von Präambel-Erkennung mit angepasstem Code und Präambel-Erkennung mit aufsummierten Samples 132
4.21 Addition der Samples zu Präambel-Sub-Symbolen in zwei verschiedenen Phasen 134
4.22 Vergleich der Paketfehlerrate eines standardkonformen Systems mit der optimierten PHY-Schicht 137
4.23 Abweichung der Synchronisation in Samples in Anhängigkeit der Clock Drift 141
4.24 Erste empfangene Übertragung 142
4.25 Zweite empfangene Übertragung 143
4.26 Addition der ersten und zweiten Übertragung 143
4.27 Dritte empfangene Übertragung 143
4.28 Addition aller drei Übertragungen 143
4.29 Benötigte Übertragungen mit harter Bit-Entscheidung und mit Soft-Bit-Kombination 145
4.30 Vergleich von CI-Fehlern bei Präambel- und Datenmodulation . 146
4.31 Paketfehlerraten mit verschiedenen FEC-Systemen 147
4.32 Konfigurationsmöglichkeiten in der Präambel 148
4.33 Struktur des Datenframes 149
4.34 Struktur des Beacon 150
4.35 Restfehlerrate von TAWASI-Datenpaketen 154
4.36 Fehlerrate des ersten Zyklus für einen Knoten 157
4.37 Häufigkeitsverteilung der benötigten Zyklen 158
4.38 Extrapolierte Häufigkeitsverteilung der benötigten Zyklen 158

4.39 Extrapolation der Telegrammfehlerrate in Abhängigkeit von Zyklen und SNR 159

5.1 Schematische Darstellung des Prototypen 164
5.2 Zustandsdiagramm des Schedulers 165
5.3 Struktur des Senders 167
5.4 Zustandsdiagramm des Basisbandempfängers 168
5.5 Zustandsdiagramm der Dekodierung im Basisbandempfänger . . 169

A.1 Zustandsdiagramm der MAC-Logikeinheit 183
A.2 Struktur des Basisbandsenders 185

Tabellenverzeichnis

2.1 Anfrage des AS-Interface-Masters 21
2.2 Antwort des AS-Interface-Slaves 21
2.3 Low-Duty-Cycle-Einschränkungen der ETSI 36
2.4 TFCs nach ECMA 368 38
2.5 PHY-Header nach IEEE 802.15.4a 47

4.1 MAC-Header eines Frames nach Draft IEEE 802.15.4e 100
4.2 Subtypen von Frames nach Draft IEEE 802.15.4e 101
4.3 Nutzlast des Beacon nach Draft IEEE 802.15.4e 102
4.4 Flags im Beacon nach Draft IEEE 802.15.4e 102
4.5 Gegenüberstellung von TAWASI und dem Stand der Technik . . 162

A.1 Adressierung des Konfigurationsblocks 181

Kapitel 1

Einleitung

Drahtlose Kommunikation ist ein Forschungsgebiet, das immer weiter in Gebiete und Anwendungsbereiche vordringt, die bisher nur mit drahtgebundener Kommunikation möglich waren. Zahlreiche Anwendungen der klassischen drahtgebundenen Kommunikation werden durch drahtlose Anwendungen ersetzt oder ergänzt. Dafür steht eine breite Palette an Technologien zur Verfügung, die sich durch verschiedene Eigenschaften für die jeweilige Anwendung anbieten.

In sehr anspruchsvollen Bereichen existieren durch hohe Anforderungen an das drahtlose System bisher keine oder nicht zufriedenstellende Lösungen. Hier sind es die individuellen Eigenschaften von Technologien, die möglicherweise ursprünglich für eine andere Problemstellung geschaffen wurden, die den entscheidenden Unterschied machen, der eine drahtlose Adaption dieser Anwendung erlaubt.

In dieser Dissertation habe ich deshalb untersucht, ob sich Ultra-Breitband-Technologie *IR-UWB*[1], die zum Teil sehr unterschiedliche Eigenschaften im Vergleich zu herkömmlicher, wellenbasierter Kommunikation aufweist, für den Einsatz in der Sensor/Aktor-Ebene der Fertigungsautomatisierung eignet. Dieses Anwendungsgebiet stellt hohe Anforderungen an die Kommunikation, die bisher verfügbare Funktechnologien nur unzureichend erfüllen können. Mit Hilfe der *IR-UWB*-Technologie könnte diese Lücke geschlossen werden.

[1] Impulse Radio UWB - *Ultra-Breitband-Kommunikation auf Basis von Pulsfunk*

1.1 Motivation

Kürzere Produktlebenszyklen, viele Produktvarianten und schnelle technologische Entwicklung erfordern viel Flexibilität, nicht zuletzt auch in der Fertigung. Diese muss darüber hinaus kosteneffizient und zuverlässig sein, was nur durch einen hohen Grad an Automatisierung erreicht werden kann.

1.1.1 Automatisierungstechnik

Automatisierungstechnik dient dazu, Systeme mit Steuerungen zu versehen, die ihnen eine weitestgehend autonome Funktion, ohne weiteren Eingriff des Menschen, ermöglicht. Es existiert eine große Palette von autonomen Automatisierungslösungen in verschiedensten Bereichen der Technik.

Nicht alle Automatisierungssysteme müssen vollkommen autonom ablaufen, es können auch nur Teilprozesse automatisiert werden. Unter Mechanisierung versteht man beispielsweise, wenn Vorgänge automatisch ablaufen, statt von Hand (manuell) erledigt zu werden, jedoch der Befehl zur Auslösung der jeweiligen automatischen Vorgänge manuell geschieht. Innerhalb dieser angestoßenen Teilprozesse besteht allerdings wiederum ein autonomes Automatisierungssystem.

Für jedes Automatisierungssystem lassen sich aber allgemein zwei Aufgaben definieren: Überwachen und Steuern. Lunze schreibt darüber in [50, S. 2]:

> » Überwachen heißt, die wichtigsten Prozessgrößen zu messen und die Messwerte zu sammeln, auszuwerten und für das Bedienpersonal in zweckmäßiger Weise darzustellen. Die Auswertung umfasst auch die Verknüpfung der Informationen, die von unterschiedlichen Sensoren kommen. Mit Hilfe eines Modells können nicht messbare Größen berechnet und Fehler aufgedeckt werden.
>
> Steuern heißt, Prozesse zielgerichtet zu beeinflussen. Die von außen vorgebbaren Größen werden so gewählt, dass die in der Anlage ablaufenden Prozesse in einer gewünschten Folge aktiviert oder wichtige Prozessgrößen trotz des Einwirkens von Störungen auf vor-

geschriebenen Werten gehalten werden. Der Begriff „Steuerung“ in seiner allgemeinen Bedeutung umfasst kontinuierlich wirkende Regelungen und diskret arbeitende Steuerungen genauso wie Steuereingriffe des Bedienpersonals. «

Für beide Aufgaben ist die Kommunikation von entscheidender Bedeutung. Daten müssen, im Rahmen der Überwachung, von Sensoren zur Steuereinheit übertragen werden. Ebenso wie im Rahmen der Steuerung Daten von der Steuereinheit zu den aktiven Komponenten, den sogenannten Aktoren übertragen werden. Mittels dieses Informationsaustausches versuchen Reglereinheiten, die gemessenen Ist-Werte an einen vorgegebenen Soll-Wert anzugleichen.

Um diesen kontinuierlichen Regelprozess mit diskret arbeitenden Aktoren oder auch allgemein mit Digitaltechnik durchführen zu können, wird ein diskreter Zyklus abgearbeitet. In diesem Zyklus werden die Überwachungsdaten bzw. Steuerdaten jedes Teilnehmers verarbeitet.

Die Kommunikation erfolgt deshalb ebenfalls in diskreten Zyklen, die dem Regelzyklus angepasst sein müssen.

Je nach Aufgabe des Automatisierungssystems variieren diese Zyklen in Dauer und Anzahl der Teilnehmer. Die wichtigsten Anwendungsszenarien von Automatisierung in der Industrie, nämlich die Prozess-, Fertigungs- und Gebäudeautomatisierung, werden im Folgenden erläutert. Im Wesentlichen unterscheiden sich diese durch unterschiedliche Anforderungen an die Zyklus- und Latenzzeit sowie Datenrate.

Es gibt eine Vielzahl weiterer Anwendungen der Automatisierung in der Industrie, auf die hier nicht weiter eingegangen wird, wie etwa die Automatisierung von Energie-, Gas- und Datennetzen (aktuell als „*smart grid*“ bekannt) sowie die Automatisierungstechnik in Fahrzeugen und Verkehrsnetzen.

1.1.1.1 Prozessautomatisierung

Typische Anwendungen der Prozessautomatisierung sind z.B. Reaktionsprozesse in der chemischen Industrie. Vergleiche dazu [50, S. 4]:

»Die Realisierung verfahrenstechnischer Prozesse beinhaltet vielfältige Automatisierungsaufgaben, so dass für dieses Gebiet ein eigener Begriff geprägt wurde: Prozessautomatisierung. Die Palette der zu lösenden Aufgaben beginnt bei der Informationsgewinnung, -übertragung und -aufbereitung für das Bedienpersonal in der Warte einer großen verfahrenstechnischen Anlage und führt über die Realisierung von Durchfluss-, Temperatur- oder Füllstandsregelungen bis zu Rezeptsteuerungen, mit denen Batchprozesse realisiert werden. Darüber hinaus kann die Automatisierungseinrichtung auch die Produktionsplanung übernehmen. «

Die Prozessautomatisierung beschreibt also eine Vielzahl von Aufgaben, vor allem im Bereich der chemischen Industrie. Diese haben meist Zykluszeiten im Minuten- bis Sekundenbereich, erfordern aber oft auch erheblich geringere Reaktionszeiten im zweistelligen Millisekundenbereich, besonders für Ausnahmesituationen.

1.1.1.2 Fertigungsautomatisierung

In dieser Arbeit betrachte ich vor allem die industrielle Automatisierung, auch als Fertigungsautomatisierung oder im englischen meist als *factory automation* bezeichnet. Vergleiche [50, S. 8]:

»Die Automatisierung ist ein unverzichtbares Element moderner Fertigungssysteme. Sie überwacht und steuert die Bewegung von Werkzeugmaschinen und Robotern. [...] Neben einer Qualitätserhöhung und der Verkürzung der Fertigungszeiten führt die Automatisierung von Fertigungsprozessen auch zu einer Erhöhung der Flexibilität. Unterschiedliche Produkte lassen sich auf denselben Werkzeugmaschinen fertigen, weil lediglich die Steuerungsalgorithmen ausgetauscht werden müssen. «

Demnach bedeutet die Automatisierung von Fertigungsprozessen eine Erhöhung von Qualität und Flexibilität, erfordert dafür aber die Verarbeitung vieler Mess- und Stellgrößen unter Echtzeitbedingungen.

Typische Anwendungen sind vor allem Fabrikanlagen, wie z.B. Fertigungsstraßen, Spritzgusswerke, Druckmaschinen oder Postsortieranlagen.

In der Fertigungsautomatisierung sind normalerweise Zykluszeiten im einstelligen Millisekundenbereich üblich sowie Reaktionszeiten im Sub-Millisekundenbereich.

Diese Reaktionszeiten unterliegen sogenannten harten Echtzeitbedingungen. Das bedeutet, dass diese Reaktionszeit auf keinen Fall überschritten werden darf, da sonst der zu steuernde Prozess ernsthaft gestört wird. Im Fall der Fertigungsautomatisierung führt dies meist zu einem Stoppen der Produktion.

Dies unterscheidet sich von den sogenannten weichen Echtzeitbedingungen, die zwar eine zeitliche Begrenzung vorgeben, bei der eine Überschreitung dieser Begrenzung aber keine prozessschädigende oder sogar gefährliche Auswirkungen hat. Weiche Echtzeitbedingungen herrschen z.B. bei Nutzerinterfaces. Hier empfindet man eine Überschreitung der Reaktionszeit als störend, sie hat jedoch keine negative Wirkung auf den zugrunde liegenden Prozess.

1.1.1.3 Gebäudeautomatisierung

Die Gebäudeautomatisierung umfasst vielfältige Automatisierungsaufgaben.

Typische Anwendungen der Gebäudeautomatisierung sind Gebäudeüberwachung und -steuerung, wie z.B. Rauchmelder, Heizungs-, Klimaanlagen- und Lichtsteuerungen, Zugangskontrollen, Aufzugssteuerungen und Alarmanlagen.

Für die Gebäudeautomatisierung sind bei vielen Anwendungen Zykluszeiten im Bereich von Minuten, Stunden und teilweise auch Tagen möglich. Der Energieverbrauch muss aber sehr gering sein, da einerseits viele Sensoren beteiligt sind, andererseits diese zum Teil ohne externe Stromversorgung betrieben werden, sodass eine Batterie jahrelang halten muss.

1.1.2 Drahtlose Automatisierung

Je intelligenter und autonomer die Automatisierungssysteme werden, desto mehr Informationen benötigen sie, und desto mehr Aktionsmöglichkeiten müs-

sen sie haben. Die rasante Entwicklung der Sensortechnologie ermöglicht es, immer mehr und zunehmend genauere Informationen über den automatisierten Ablauf zu sammeln.

Das bedeutet, dass mehr und mehr Geräte an die Steuerung angeschlossen werden müssen, was wiederum einen erheblichen Aufwand bei der Verkabelung erfordert. Wenn zusätzlich auch noch die Flexibilität gesteigert wird, kommen drahtgebundene Steuerungen schnell an ihre Grenzen. Als Beispiel seien Werkzeugmaschinen oder andere Fertigungsanlagen, deren Produktion häufig wechselt genannt.

Oft ist eine Verkabelung auch technisch sehr schwer lösbar, wie z.B. aufgrund beengter Verhältnisse innerhalb einer Maschine. Materialermüdung bei Bewegung und Vibration verschleißen Kabel sehr schnell. Bei beweglichen oder rotierenden Teilen wie Roboterarmen sind wartungsintensive Schleifringe eine Schwachstelle. Dies sind mehrere Beispiele von Problemen, die durch drahtlose Kommunikation eleganter und effizienter gelöst werden können.

Automatisierung ist ein weites Feld, dessen einzelne Anwendungen zwar gewisse Gemeinsamkeiten haben, aber auch gravierende Unterschiede, die sich wiederum in den Anforderungen an ein drahtloses System und dem daraus resultierenden Protokolldesign widerspiegeln.

Als Gemeinsamkeit lässt sich bei Automatisierungslösungen feststellen, dass die Kommunikation fast ausschließlich autonom zwischen Maschinen stattfindet, auch M2M-Kommunikation genannt. An manchen Stellen gibt es menschliche Nutzer, z.B. bei der Konfiguration, der Überwachung des Betriebes oder dem Auslesen von Zählerständen. In diesen Fällen werden Daten auf einem Knoten des drahtlosen Netzes speziell für den Benutzer aufbereitet. Die Kommunikation zwischen Maschinen findet über vergleichsweise kurze binäre Datenstrukturen statt. Dies bedingt andere Strukturen und Kommunikationsmuster als die im Endkundenbereich üblichen Drahtlosnetzwerke, die auf Web- oder Multimedia-Anwendungen optimiert sind.

Die Kommunikation in Automatisierungsnetzen ist im Regelfall sehr vorhersehbar und daher gut planbar. Meist gibt es spezielle Ereignisse, deren Auf-

treten eine plötzliche Änderung der Kommunikationsmuster erfordern, wie ein Feueralarm oder ein kritische Zustände eines chemischen Prozesses.

Die Automatisierung wird horizontal in mehrere Ebenen eingeteilt, von denen die „oberen“ Ebenen Funktionen der „unteren“ Ebenen nutzen, um ihre Aufgaben zu erfüllen. Dabei steigt mit jeder tieferen Ebene die Zahl der Komponenten in den Systemen an. Für eine Verkabelung bedeutet dies ein exponentielles Wachstum der Anzahl der Geräte mit zunehmender Tiefe der Ebene.

Je nach Anwendung und Ebene der Automatisierung werden verschiedene Anforderungen bzgl. Reaktionszeit, Energieverbrauch, Fehlerrate, Verlässlichkeit, Sicherheit und Datenintegrität gestellt. Durch diese Vielzahl möglicher Anforderungen an das Funksystem wird eine breite Palette von technologischen Lösungen benötigt, um die jeweils Optimale für ein Problem anwenden zu können.

1.1.3 Warum UWB?

Ultra-Breitband-Kommunikation *UWB*[2] ist eine relativ junge (bzw. erst kürzlich wiederentdeckte) Technologie der drahtlosen Kommunikation. Sie wird in Abschnitt 2.2.3 auf Seite 28 genauer vorgestellt. In diesem Abschnitt soll verdeutlicht werden, warum eine bestimmte Ausprägung der *UWB*-Technologie, genannt *IR-UWB*, besonders interessant für die Anwendung in der Fertigungsautomatisierung ist.

Herkömmliche drahtlose Kommunikationssysteme senden eine Trägerwelle, auf die die Daten aufmoduliert werden, wohingegen *IR-UWB* die Energie in diskreten Pulsen von sehr kurzer zeitlicher Dauer versendet. Dies ermöglicht einen sehr einfachen Aufbau des Empfängers, der nur die Präsenz oder Abwesenheit eines Pulses zu einem bestimmten Zeitpunkt ermitteln muss. Dadurch wird aber trotzdem eine hohe Robustheit gegenüber vielen Problemen, die sonst in der drahtlosen Kommunikation auftreten, erreicht.

Als ein Beispiel seien hier die Reflexionen an metallischen Oberflächen, die sogenannten Multipfad-Effekte, gezeigt:

[2] Ultra-Wideband - *Ultra-Breitband*

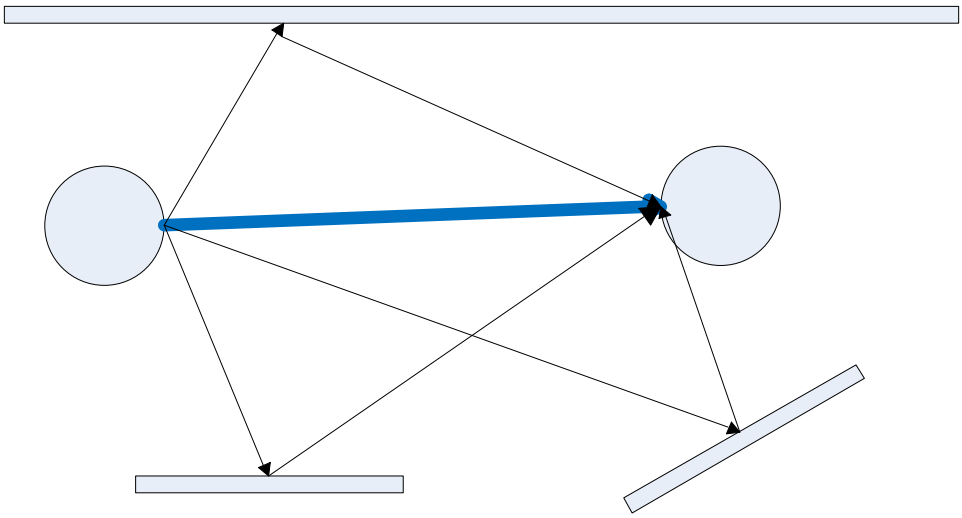


Abbildung 1.1: Beispiel einer Multipfadausbreitung eines Signals

Bei einer Funkübertragung wird das Signal nicht nur entlang des direkten Sichtpfades zum Empfänger übertragen, wie in Abbildung 1.1 dick eingezeichnet, sondern auch durch Reflexion über weitere Pfade. Aufgrund der verschiedenen Länge dieser Pfade kollidieren mehrere, unterschiedlich zeitverzögerte, Kopien des Signals beim Empfänger.

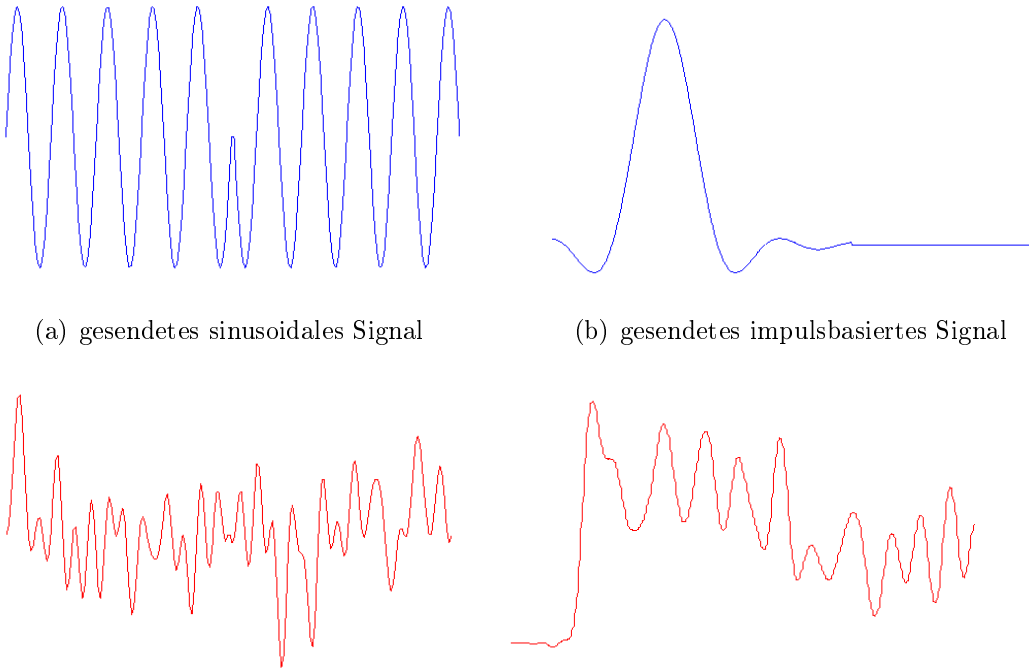
Diese Kopien überlagern sich, was zu einer Störung des Signals beim Empfänger führt. Bei einer herkömmlichen Modulation auf eine Trägerwelle erfordert es mehrerer Korrekturen durch einen Equalizer, um das ursprüngliche Signal wiederherzustellen. Bei *IR-UWB* kann ein einfacher Energie-Detektions-Empfänger die gesendete Information empfangen, indem er das Energieniveau im Kanal misst.

Die Pulsdauer T_C ist gegenüber dem *Delay Spread* sehr klein. Deswegen kommt es innerhalb des betrachteten Zeitfensters nur zu Überlagerungen durch Reflexionen von Objekten in unmittelbarer Umgebung des Empfängers. Dadurch können alle später eintreffenden Reflexionen längerer Pfade ignoriert werden. Bei einer T_C von 2 ns sind das 60 cm Wegunterschied zwischen den Pfaden.

In Abbildung 1.2 sind die gesendeten Signale eines trägerbasierten Kommu-

nikationssystems mit *BPSK*[3]-Modulation 1.2(a) und eines *IR-UWB*-Systems 1.2(b).

Es ist erkennbar, dass das das empfangene Signal des sinosoidalen Systems 1.2(c) eine aufwendige Rekonstruktion benötigt, um die modulierten Informationen wiederherzustellen. Beim empfangenen *IR-UWB*-Signal 1.2(d) kann durch Vergleich der Energieniveaus die Information empfangen werden.



(a) gesendetes sinusoidales Signal

(b) gesendetes impulsbasiertes Signal

(c) empfangenes sinusoidales Signal

(d) empfangenes impulsbasiertes Signal

Abbildung 1.2: Einfluss von Multipfad-Effekten auf das Signal

1.2 Problemstellung

Zur Untersuchung der Eignung von *IR-UWB* für die industrielle Fertigungsautomatisierung der Sensor/Aktor-Ebene soll ein robustes drahtloses *IR-UWB*-Funksystem erstellt und untersucht werden. Dabei soll das System geeignet sein, ein bestehendes, drahtgebundenes Protokoll, wie z.B. das *AS-Interface*-Protokoll, ersetzen zu können. Dazu muss es für 32 Teilnehmer eine mittlere

[3] *Binary Phase Shift Keying*

Reaktionszeit von unter 5 ms aufweisen und harte Echtzeitbedingungen mit einer Zeitschranke von 20 ms mit einer Telegrammfehlerrate unter 10^{-9} erfüllen können.

Dies setzt eine Balance zwischen der Robustheit der Übertragung einerseits und Latenzen im Mikrosekundenbereich andererseits voraus.

Wo immer möglich, sollen Standardprotokolle verwendet oder zumindest als Basis im Sinne einer Erweiterung genommen werden. Dadurch wird eine Standardisierbarkeit des Systems in einer zukünftigen Version des Standards ermöglicht.

1.3 Ergebnisse

Ich habe ein flexibles Funksystem auf *IR-UWB*-Basis entwickelt und untersucht. Als das am besten geeignete *MAC*[4]-Protokoll stellte sich eine *MAC*-Schicht heraus, die konform zu dem Standardentwurf der Arbeitsgruppe IEEE 802.15.4e ist. Als *PHY*[5]-Schicht wurde auf Basis von Untersuchungen ein neuer Modus für den Standard IEEE 802.15.4a entwickelt, der für die Automatisierung optimiert ist. Dieser Modus enthält Optimierungen an der Paketstruktur zur Reduktion von Latenz und Erhöhung der Robustheit. Weiterhin wurden schichtübergreifende Optimierungen untersucht und angewendet.

Außerdem wurden Methoden zur Verbesserung des Designs nichtkohärenter Energie-Detektions-Empfänger für IEEE 802.15.4a entwickelt und verifiziert, die den Stand der Technik übertreffen.

Dieses Funksystem erweitert den Stand der Technik der die drahtlose Fertigungsautomatisierung in der Sensor/Aktor-Ebene, indem es das erste standardbasierte System für diese Anwendung ist und das erste auf *IR-UWB*-Basis.

Es wurde eine konkrete Anwendung des Systems untersucht, die als drahtlose Adaption des *AS-Interface*-Protokolls dient, für das es bisher nur drahtgebundene Lösungen gab. Sie erreicht dabei Zykluszeiten von 1,8 ms bei 32

[4] Media Access Control - *Medienzugriffssteuerung*
[5] *Physikalische Schicht*

Teilnehmern. Das System erreicht damit eine mittlere Reaktionszeit unterhalb einer Millisekunde und kann harte Echtzeitbedingungen von 15 ms mit einer Fehlerrate von 10^{-9} erfüllen. Darüber hinaus ist ist das System fähig zur Koexistenz mit anderen Systemen in ISM-Bändern und erfüllt mindestens die Datenintegritätsklasse I2.

Diese Leistungsmerkmale übertreffen in wichtigen Parametern bisherige drahtlose und drahtgebundene Systeme.

Zur Evaluation der Forschungsergebnisse habe ich eine konfigurierbare Simulationssoftware für die *PHY*-Schicht auf Basis anerkannter Kanalmodelle geschrieben. Außerdem wurde eine Implementierung der *MAC*-Schicht sowie des digitalen Teils der *PHY*-Schicht auf *FPGA*[6]-Basis vorgenommen, um eine Verifikation der Forschungsergebnisse in Hardware zu ermöglichen.

Die *FPGA*-Implementierung benutzt als analoges *Front-End* den weltweit ersten Prototypen eines IEEE 802.15.4a-konformen *IR-UWB*-Transceivers des IHP (Leibniz Institut für innovative Mikroelektronik). Leider war zum Zeitpunkt des Verfassens dieser Dissertation der Transceiver noch in einem sehr frühen Entwicklungsstadium, weshalb Messungen in einer realistischen Umgebung und unter reellen Störbedingungen nicht erfolgen konnten. Es wurde jedoch die Kompatibilität der Hardware sowie die Funktionalität des Konzeptes nachgewiesen.

1.4 Aufbau der Dissertation

Diese Dissertation ist gliedert sich wie folgt aufgebaut: In Kapitel 2 wird der bisherige Stand von Wissenschaft und Technik in den für diese Arbeit relevanten Themen aufgezeigt. In Kapitel 3 wird das Modell einer Paketübertragung gemäß IEEE 802.15.4a und die damit entwickelte Simulationssoftware vorgestellt. Kapitel 4 erläutert das entworfene System, mitsamt den verwendeten Protokolle und den dazu nötigen Modifikationen. Kapitel 5 beschreibt die *FPGA*-Implementierung des Systems. In Kapitel 6 wird die Arbeit zusam-

[6]Field Programmable Gate Array - *Programmierbare Logik*

mengefasst und diskutiert sowie der Ausblick auf weitere Forschungsinhalte gegeben.

Kapitel 2

Stand von Wissenschaft und Technik

2.1 Kriterien

Um die Leistungsfähigkeit von Kommunikationssystemen der industriellen Automatisierung bewerten zu können, sind numerische Kriterien notwendig. Anhand dieser Kriterien lassen sich jeweils bestimmte Aspekte der Systeme quantifizieren.

2.1.1 Mittlere Reaktionszeit

Die mittlere Reaktionszeit gibt die durchschnittliche Dauer vom Auftreten eines Ereignisses bis zu dessen Meldung an die Steuerung an. Da dieser Wert den systematischen Ansatz bewertet, wird von erfolgreichen Übertragungen ausgegangen.

Der Wert wird berechnet durch die Addition der Hälfte der Dauer eines Zyklus T_{Zyklus} und der Dauer eines Telegramms $T_{\text{Telegramm}}$.

$$T_{\text{Reaktion}} = \frac{T_{Zyklus}}{2} + T_{\text{Telegramm}} \tag{2.1}$$

2.1.2 Telegrammfehlerrate / -latenz

Jede Telegrammübertragung kann mit einer bestimmten Wahrscheinlichkeit fehlschlagen. Diese Wahrscheinlichkeit ist abhängig von den Paketfehlerraten aller zur Übertragung des Telegramms benötigten Datenpakete.

Die Fehlerrate einzelner Telegramme wirkt sich vor allem durch eine Erhöhung der Latenz aus. Deshalb kann als Vergleichskriterium die Wahrscheinlichkeit der erfolgreichen Übertragung in Abhängigkeit von der benötigten Zeit genommen werden. Diese benötigte Zeit ist ein Vielfaches der Zykluszeit.

Als obere Schranke im Sinne harter Echtzeitkriterien wird die Latenz entsprechend einer Telegramm-Fehlerrate von 10^{-9} angesetzt.

2.1.3 Restfehlerrate

Die Restfehlerrate ist ein Maß für die Absicherung der Datenintegrität. Sie gibt die Anzahl der unerkannten Paketfehler, das sind Pakete mit Bit-Fehlern, welche fälschlicherweise als korrekt erkannt werden, in Abhängigkeit von der Bit-Fehlerrate an. Dazu wird betrachtet, wie sich Bit-Fehler beim Paketempfang auswirken.

Für die Nutzlast *FEC*[1]-kodierter Pakete mit einem *FEC*-Code der Hamming-Distanz k kann die Restfehlerrate

$$\begin{aligned} R(p) &= \sum_{k=d}^{n} \binom{n}{k} p^k (1-p)^{n-k} \\ &= 1 - \sum_{k=0}^{d-1} \binom{n}{k} p^k (1-p)^{n-k} \end{aligned} \tag{2.2}$$

in Abhängigkeit der Bit-Fehlerrate p und der Anzahl der Bits im Paket n berechnet werden [28].

[1] Forward Error Correction - *Fehlerkorrekturcode*

2.1.4 Datenrate

Ein Kommunikationssystem in der Automatisierung ist normalerweise für eine definierte Anzahl an Bits pro Teilnehmer und pro Zyklus ausgelegt, wobei die Datenrate selbst eine untergeordnete Rolle spielt.

Es können aber auch Übertragungen mit anderen Datenlängen oder gar heterogene Netze mit Komponenten verschiedener Informationslänge vorkommen. Dabei müssen bei manchen Komponenten die Übertragung auf mehrere Zyklen aufgeteilt werden. Als Beispiel seien hier analoge 16-Bit-Komponenten oder Konfigurationsdaten in genannt. Für diese Fälle ist eine Betrachtung der Datenrate des Netzwerkes wichtig.

Es lassen sich Brutto- und Nettodatenraten für drahtlose Systeme bestimmen. Dabei ist die Bruttodatenrate

$$v_{Brutto} = \frac{N_{Bit}}{T_{Sym}} \tag{2.3}$$

von der Anzahl der Bits pro Symbol N_{Bit} abhängig und invers zur Dauer eines Symbols T_{Sym}.

Die Nettodatenrate unterscheidet sich zwischen einem zentralen Gateway und einem peripheren Knoten. Sie berechnet sich durch

$$v_{Netto} = \frac{N_{Nutzdaten} \cdot N_{Pkt}}{T_{Zyklus}} \tag{2.4}$$

wobei $N_{Nutzdaten}$ die Anzahl der Bits der Nutzlast pro Paket, N_{Pkt} die Anzahl der Pakete pro Zyklus und T_{Zyklus} die Dauer eines Zyklus ist.

2.1.5 Koexistenzfähigkeit

Unterschiedliche Funksysteme, die jedoch dieselben Frequenzen benutzen, stören sich gegenseitig. Die Koexistenzfähigkeit lässt sich also anhand Reduktion der Leistung eines anderen Funksystems im selben Frequenzbereich ausdrücken. Das Ausmaß dieser Störung lässt sich durch Messungen ermitteln.

In der Fertigungsautomatisierung benötigen vor allem Systeme in den *ISM*[2]-Bändern Koexistenztechnologien, da diese mit anderen Funksystemen, wie z.B. die Kommunikation der übergeordneten Automatisierungs-Ebene störungsfrei zusammenarbeiten müssen.

2.2 Stand der Technik

Zuerst werden vorhandene Systeme und Lösungen zur drahtgebundenen und drahtlosen Kommunikation in der Sensor/Aktor-Ebene im Hinblick auf die erwähnten Kriterien betrachtet. Anhand dieses Vergleiches lässt sich das in dieser Dissertation erstellte System einordnen. In Abschnitt 2.2.3 wird dann der Stand der verfügbaren *UWB*-Kommunikationstechnologie vorgestellt, auf dem die vorgestellte Lösung aufbaut.

2.2.1 Kommunikation in der Fertigungsautomatisierung

In Netzen der industriellen Automatisierung treffen viele verschiedene Anforderungen bzgl. Anzahl der Knoten, Reaktionszeiten und benötigten Datenraten aufeinander. Um diese optimal abzudecken, werden verschiedene miteinander verbundene Systeme eingesetzt, die unterschiedlichen Ebenen der Automatisierung zuzuordnen sind. Diese Ebenen haben jeweils eigene Aufgaben und auf sie zugeschnittene Merkmale.

Die Feldbusebene ist dabei die oberste Steuerungsebene. Sie ist auf hohen Durchsatz für (vergleichsweise) wenige, zeitlich unkritische Nachrichten mit vielen Daten optimiert. Darunter folgt die Geräteebene als Verbindungsglied zwischen den einzelnen Geräten und dem Feldbus. Zuunterst liegt die Sensor-/Aktor-Ebene. Sie ist auf kurze Reaktionszeit bei vielen Teilnehmern optimiert, mit häufigen Paketen von kurzer Dauer für Sensordaten oder Steuernachrichten.

[2]Industial, Scientific, Medical - *International freigegebene Funkbänder für industrielle, wissenschaftliche oder medizinische Anwendungen*

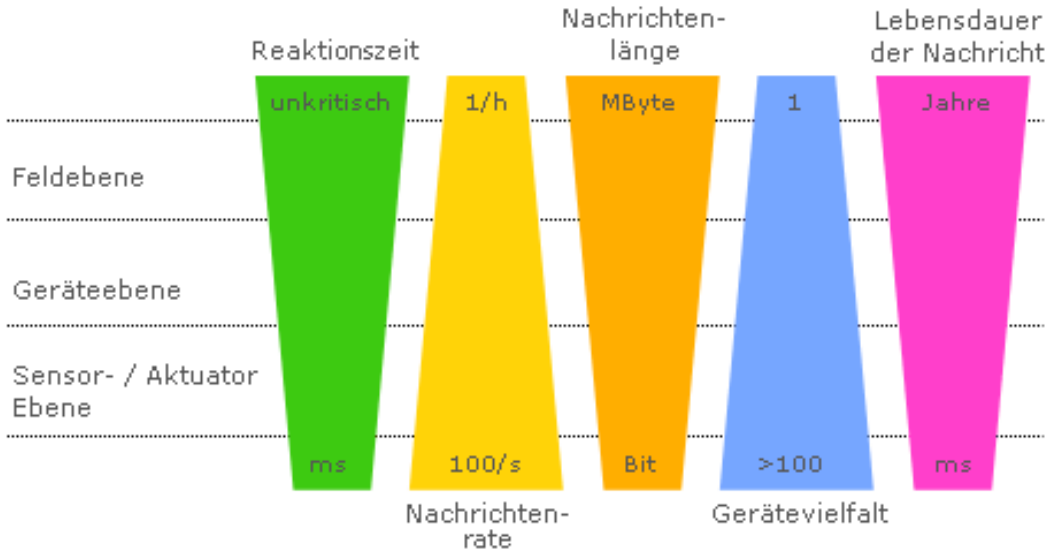


Abbildung 2.1: Ebenen der industriellen Automatisierung[7]

Die Steuerung kann entweder zentral oder verteilt erfolgen. Bei verteilten Steuerungen werden Entscheidungen autonom innerhalb der einzelnen Ebenen getroffen. In beiden Fällen wird zumindest eine Aggregation der Daten bzw. Steuerbefehle an den Übergängen verschiedener Ebenen stattfinden.

2.2.1.1 IO-Link

Das System „IO-Link“[3] bietet eine kabelgebundene Punkt-zu-Punkt-Verbindung für die Fertigungsautomatisierung an. Damit können einzelne Geräte der Sensor/Aktor-Ebene in die Feldbusebene abgebildet werden. Alternativ kann auch ein *Master* mit mehreren IO-Link-Ports als *Gateway* eingesetzt werden. IO-Link ist dabei vor allem ein digitales Übertragungssystem für die ansonsten analog übertragenen Prozessdaten. Dadurch präsentiert sich ein IO-Link-*Master* als Feldgerät, abstrahiert die Sensor/Aktor-Ebene, und bietet der Steuerung Zugriff auf die angeschlossenen Sensoren und Aktoren.

IO-Link überbrückt bis zu 20 m Entfernung zwischen einem *Master* und einem *Device*. Dabei wird ein industrieübliches, ungeschirmtes 3-Leiter-Kabel

[3]IO steht hierbei für *Input-Output*, Eingabe-Ausgabe

verwendet. IO-Link arbeitet mit drei verschiedenen Brutto-Baudraten: 4.800 Baud, 38.400 Baud und 230.400 Baud. Beim ersten Verbindungsaufbau (genannt „aufwachen“) versucht der *Master* jeweils dreimal auf jeder Baudrate das *Device* zu erreichen, dabei geht er von schneller Baudrate zu langsamer vor. Jedes *Device* kann nur auf einer bestimmten Baudrate kommunizieren. Sobald das *Device* die Anfrage des *Master*s erfolgreich empfangen kann, antwortet es, und der *Master* wechselt in einen Modus zyklischer Datenübertragung.

Prozessdaten werden zyklisch in Daten-Frames übertragen. Die Prozessdaten können eine vom *Device* festgelegte Länge haben von 0 bis 32 Byte (je Input und Output) haben. Es gibt außerdem azyklische Daten, wie Parameter oder Ereignisse.

Das System benutzt einen *Polling*-Ansatz, bei dem stets der *Master* die Kommunikation eröffnet. Siehe dazu [61]:

> »Das IO-Link-*Device* sendet immer nur nach einer Aufforderung des IO-Link-*Master*s. Prozessdaten werden nach dem IDLE-Telegramm des *Master*s gesendet. *Device*-Parameterdaten und Ereignisse werden explizit vom *Master* angefragt. «

Die Zykluszeit der Prozessdaten beträgt bei 38,4 kBaud und einer Telegrammlänge von 2 Byte typischerweise 2 ms. Dabei gibt jedes *Device* eine minimale Zykluszeit vor, der *Master* legt daraufhin die endgültige Zykluszeit fest und gibt sie bekannt.

IO-Link definiert über zwei verschiedene Arten von Datenfeldern in den diversen Telegrammtypen zwei verschiedene Kommunikationskanäle für zyklische und azyklische Daten. Dadurch sind spezielle Zeitfenster für den asynchronen Datenaustausch reserviert, in dem aber jeweils immer nur eine der zwei Datenformen (Ereignisse oder Parameter) ausgetauscht werden können.

Die verschiedenen Telegrammtypen enthalten immer Felder für Prozessdaten, sodass die Dienstgüte der Prozessdaten hinsichtlich Determinismus und Geschwindigkeit erhalten bleibt. Die Daten können dabei mit einem einzelnen oder auch verteilt über mehrere Telegramme übertragen werden.

Zur Berechnung der Kenngrößen muss die Struktur der Daten genauer betrachtet werden. Es wird ein typischer Nachrichtenaustausch von 2ms Dauer bei einer Übertragungsrate von 38,4 kBaud angenommen. Dabei werden immer 2 Byte Nutzlast übertragen, die durch verschiedene Telegrammtypen in Up- und Downlink aufgeteilt werden können. Die Nettodatenrate dieser Übertragung von 2 Byte in 2 ms liegt bei 8 kBit/s.

Da nur zwei Teilnehmer kommunizieren, ist die mittlere Reaktionszeit gleich der Zykluszeit, also 2ms. IO-Link erlaubt bis zu zwei Wiederholungen eines fehlgeschlagenen Telegramms, wodurch sich eine maximale Telegrammlatenz von 6 ms ergibt.

Die Daten werden in *UART*[4]*-Frames* übertragen, diese sind mit einem Paritäts-Bit gesichert. Diese Absicherung ergibt eine Hamming-Distanz von 2 für jedes Byte. Dadurch lässt sich mit Formel (2.2) eine Restfehlerrate pro Byte von 10^{-7} für die angenommene Bit-Fehlerrate von 10^{-4} berechnen.

2.2.1.2 AS-Interface

Das Protokoll *AS-Interface* ist ein kabelgebundenes Protokoll zur Kommunikation zwischen einer Steuerung und den zu steuernden Aktoren sowie den Sensoren, welche die zur Steuerung benötigten Daten bereitstellen.

Die *PHY*-Schicht von *AS-Interface* benutzt dabei ein Signal im Frequenzbereich von 50 kHz bis 300 kHz. Dieses Signal wird über eine Entfernung von maximal 100 m übertragen.

Die Daten werden auf die Energieversorgung moduliert, durch eine gleichstromfreie Basisbandmodulation namens *APM*[5]. Dabei wird ein Manchester-Code übertragen, indem er als Strom versendet und als abgeleitete Spannung empfangen wird. Für mehr Details sei auf [8] verwiesen. Ein Bit wird dabei in 6 μs übertragen.

[4] Universal Asynchronous Receiver/Transmitter - *universeller asynchroner Sender/Empfänger*

[5] *Alternierende Puls-Modulation - Basisband-Modulationsverfahren, wird im AS-Interface-Protokoll eingesetzt.*

Die Netztopologie kann beliebig sein, alle Geräte benutzen ein einziges *Shared Medium*. Es können ein *Master* und bis zu 32 *Slaves* angeschlossen sein. In der Spezifikation v2.1 mittels „erweiterten Adressiermodus“ sind auch 62 *Slaves* möglich, wodurch sich Adressierung und Anzahl der Datenbits ändern.

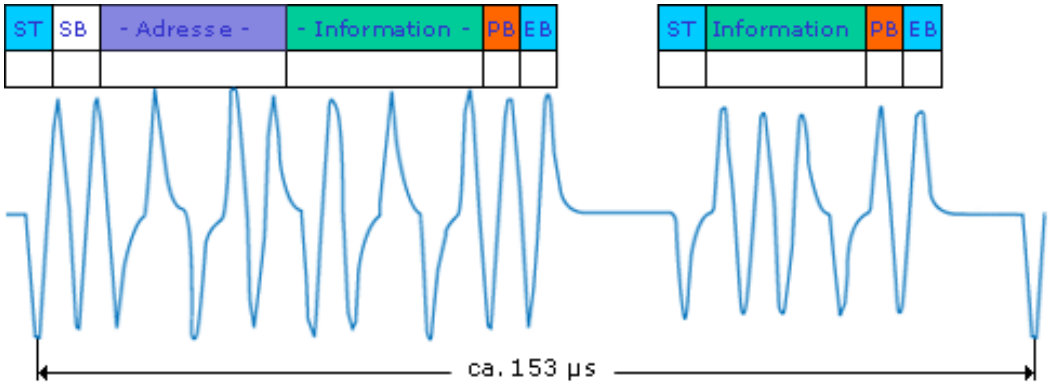


Abbildung 2.2: Nachrichtenaustausch im AS-Interface-Protokoll [7]

Die Kommunikation erfolgt über zyklische *Polling*-Anfragen des *Masters* und sofortigen Antworten der *Slaves*. Abbildung 2.2 zeigt ein Beispiel für einen solchen Nachrichtenaustausch. Im typischen Fall dauert ein solcher Nachrichtenaustausch $\sim 153\,\mu$s. Die Brutto-Datenübertragungsrate ist damit 167 kBit/s.

Dabei sendet der *Master* eine 14 Bit, entsprechend $84\,\mu$s, lange Anfrage an einen *Slave*. Auf diese Antwort muss innerhalb einer festgelegten Zeit, typischerweise $16\,\mu$s, maximal so lang wie die *Slave*-Antwort, geantwortet werden muss. Die Antwort des *Slaves* ist 7 Bit, entsprechend $42\,\mu$s, lang. Nach Eingang der Antwort erfolgt eine Pause von 9-12 μs vor der nächsten Anfrage. Sollte bis zum Ende der Wartezeit keine Antwort eingegangen sein, dann startet der *Master* direkt die nächste Anfrage.

Tabelle 2.1 auf der nächsten Seite zeigt den Aufbau der Anfrage des *Masters*. Die *Slave*-Antwort ist in Tabelle 2.2 auf der nächsten Seite gezeigt. Die Start- und Stoppbits dienen der Synchronisation und sind mit 0 und 1 belegt. Das Paritäts-Bit wird auf gerade Parität gesetzt.

Der erweiterte Adressierungsmodus benutzt ein Bit der Nutzlast zur Adressierung. Dies geschieht transparent für Geräte, die keine erweiterte Adressierung

Tabelle 2.1: Anfrage des AS-Interface-Masters

Feld	Start	Steuerung	Adresse	Nutzlast	Parität	Stopp
Regulär	1 Bit	1 Bit	5 Bit	4 Bit	1 Bit	1 Bit
erweitert	=	=	6 Bit	3 Bit	=	=

unterstützen, da das Bit als „select Bit“ zwischen zwei *Slaves* mit der gleichen Adresse unterscheidet. Deshalb muss nur einer der beiden *Slaves* dieser Adresse die erweiterte Adressierung unterstützen. Die Antwort der *Slaves* bleibt in beiden Adressierungsmodi gleich. Dadurch ergibt sich für den erweiterten Adressiermodus eine eine Nutzlast von 3 Bit *Downlink* und 4 Bit *Uplink* pro Teilnehmer und Zyklus.

Tabelle 2.2: Antwort des AS-Interface-Slaves

Feld	Startbit	Nutzlast	Parität	Stoppbit
Länge	1 Bit	4 Bit	1 Bit	1 Bit

Ein kompletter Zyklus im *AS-Interface*-Protokoll besteht aus einem Nachrichtenaustausch von Prozessdaten mit jedem *Slave*, einem Nachrichtenaustausch zum Management und einem weiterem zum Auffinden neuer *Slaves* sowie optional einer Reserve für azyklische Daten.

Die Länge eines solchen Zyklus im normalen Modus kann man also mit 33 Nachrichten annehmen. Daraus resultiert eine Zykluszeit von 5,08 ms, woraus man eine mittlere Reaktionszeit von 2,7 ms errechnen kann. Wird der erweiterte Adressierungsmodus verwendet, dann liegt bei einem vollbesetzten Netz die Zykluszeit bei 9,97 ms und die mittlere Reaktionszeit bei 5,14 μs.

Es können auch mehrere Nachrichten hintereinander kombiniert werden, um z.B. 16-Bit-Komponenten anzubinden. Die Kombination der Pakete zum sog. kombinierten Informationsaustausch wird von der Applikationsschicht vorgenommen, wofür das *AS-Interface*-Protokoll mehrere Profile definiert. Für *Slaves* mit kombiniertem Informationsaustausch erhöht sich Zyklus- und Resaktionszeit, da pro Zyklus immer nur ein Informationsaustausch pro *Slave* stattfindet. Für ein Feldgerät mit einem Byte Nutzdaten verlängert sich die Zykluszeit so auf 40 ms.

Durch die 4 Bit Nutzlast pro Zyklus von 5 ms im normalen Adressierungsmodus beträgt die Netto-Datenübertragungsrate bidirektional 800 Bit/s. Da die Zykluszeit im erweiterten Adressierungsmodus auf ca. 10 ms ansteigt, verringert sich die Netto-Datenrate im *Uplink* auf 400 Bit/s, im *Downlink* sogar auf 300 Bit/s.

Durch die Absicherung mit dem Paritäts-Bit ergibt sich eine Hamming-Distanz von 2. Bei einer angenommenen Bit-Fehlerrate von 10^{-4} resultiert dies in einer Restfehlerrate von 10^{-8} für die Slave-Antwort und 10^{-7} für die Master-Anfrage[6].

2.2.2 Drahtlose Automatisierung in der Fertigung

Im Gegensatz zur Gebäudeautomatisierung werden drahtlose Lösungen in der Fertigungsautomatisierung nur selten eingesetzt. Als Hauptgründe hierfür werden meist die Übertragungsqualität im Hinblick auf Multipfad-Effekte und die Notwendigkeit der Koexistenz genannt [15]. Für diese beiden Punkte ist *IR-UWB* eine vielversprechende Lösung. Dieser Abschnitt gibt einen Überblick über bestehende Lösungen.

2.2.2.1 WISA

Das System *WISA*[7] der Firma ABB besteht unter anderem aus einem drahtlosen Kommunikationssystem für die Sensor/Aktor-Ebene, genannt *WISA*-COM. Die Kommunikation erfolgt im 2,4 GHz-Band und basiert auf der *PHY*-Schicht von *Bluetooth*. Jeder Sensor verfügt über einen zur *PHY*-Schicht von IEEE 802.15.1 standardkonformen Bluetooth-Transceiver Die Basisstation hingegen besitzt ein proprietäres Full-Duplex-*Front-End* [76]. Die maximale Entfernung zur Basisstation liegt bei 5 m.

Zur Übertragung wird jedem Sensor ein eigener Zeitschlitz und eine Frequenz zugeordnet. *WISA*-COM arbeitet mit einer Zykluszeit von 2 ms. Insgesamt

[6] Diese Berechnung betrachtet nur die Nutzlast, Steuer- und Addressfelder. Fehler in den Start- und Stoppbits führen automatisch zur Verwerfung des Paketes

[7] *Wireless Interface for Sensors and Actors*

werden fünf Bluetooth-Kanäle gleichzeitig benutzt. Pro Basisstation können damit bis zu 120 Sensoren betrieben werden.

Wie Abbildung 2.3 zeigt, dient ein Frequenzkanal zum Downlink und weitere vier zum Uplink. Der Zyklus von 2 ms ist in 16 Zeitschlitze von jeweils 128 μs unterteilt. In den Uplink-Kanälen werden den Sensoren entweder lange Zeitschlitze von 128 μs für eine Nutzlast von 32 Bit oder kurze Zeitschlitze von 64 μs für eine Nutzlast von 1 Bit zugeteilt. Der Kanal auf dem Downlink benutzt immer Zeitschlitze von 128 μs, die 32 Bit können jedoch auf mehrere Sensoren aufgeteilt werden [43].

Die Nettodatenrate im *Uplink* liegt dadurch für Sensoren, die kurze Zeitschlitze nutzen, bei 500 Bit/s. Für Geräte, die lange Zeitschlitze nutzen liegt die Nettodatenrate bei 16 kBit/s. Die Nettodatenrate im *Downlink* beträgt 2,5 kBit/s für jeden Teilnehmer.

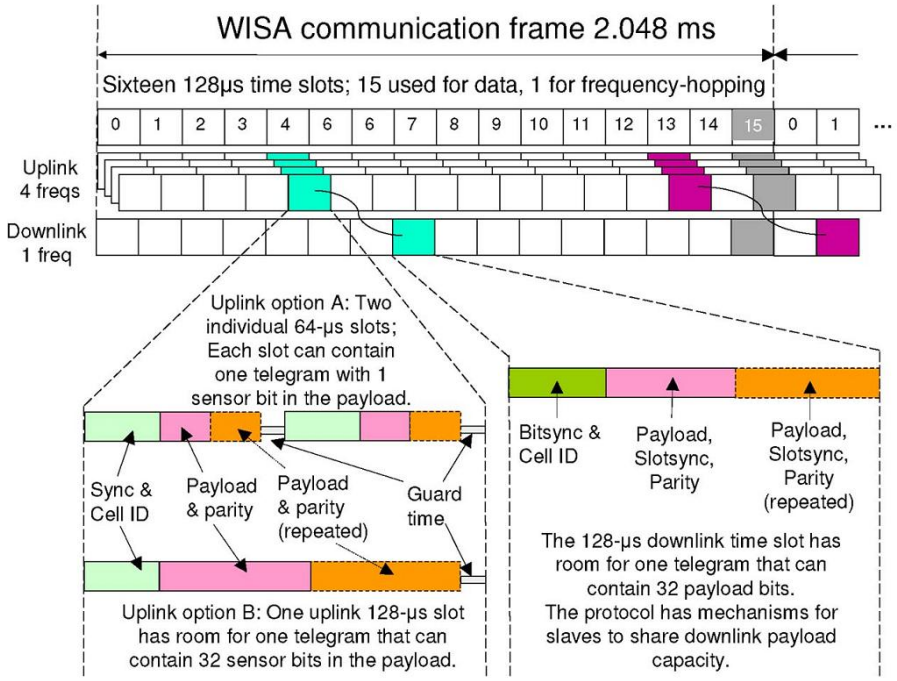


Abbildung 2.3: Verteilung von Zeitschlitzen und Frequenzen bei WISA [43]

Die Basisstation sendet dabei ständig ein Synchronisationssignal auf dem dedizierten Downlink-Frequenzkanal. Alle Sensoren verbleiben in einem Ruhezu-

stand, bis eine Änderung der Sensordaten stattfindet. Erst dann synchronisiert sich der Sensor und sendet solange in seinem Zeitschlitz, bis der Empfang der Nachricht bestätigt wird. Dieser Prozess kann zwischen 5 und 20 ms dauern. [70]

Das System benutzt ein optimiertes Frequenzsprungverfahren, um schmalbandige Interferenz- und Fading-Effekte zu vermeiden. Andere Funksysteme in den 2,4 GHz *ISM*-Bändern beeinträchtigen das System allerdings trotzdem, wie in [70] untersucht wurde. So konnte eine gleichzeitige WLAN-Übertragung die Latenzen des Systems bis auf die fünffache Länge bringen. Die effektive Datenrate der WLAN-Übertragung sank dabei von 3.129 kByte/s auf 860 kByte/s.

ABB *WISA* hat als Design-Ziel eine Telegrammfehlerrate von 10^{-9} bei einer maximalen Latenz[8] von 34 ms ab Zustandsänderung eines Sensors.

Die Daten werden durch ein Paritäts-Bit abgesichert und als Redundanz wiederholt. Die Restfehlerrate, mit einer angenommenen Bit-Fehlerrate von 10^{-4} für jede der Wiederholungen, liegt damit bei 10^{-8} für kurze Zeitschlitze und bei 10^{-6} für lange Zeitschlitze.

Die Redundanz der doppelten Übertragung verringert die Restfehlerrate jedoch nur dann, wenn ein Empfänger bei zwei scheinbar erfolgreichen Wiederholungen einer Übertragung einen Unterschied feststellt und beide verwirft. Ansonsten verringert die zusätzliche Datenübertragung nur die Paketfehlerrate, wobei nur Datenfehler korrigiert werden können, jedoch keine Präambel-Fehler.

2.2.2.2 Wireless HART

Wireless HART ist eine drahtlose Adaption des HART Protokolls der „HART Communication Foundation“. Das HART Protokoll ist ein drahtgebundenes Protokoll, das bis zu zwei *Master Devices* mit bis zu 15 *Slave Devices* über ein Kabelpaar verbinden kann. Das Protokoll beinhaltet auch höhere Schichten, von der Netzwerkschicht bis hin zur Applikationsschicht. [34]

[8] Im Sinne einer harten Echtzeitanforderung

Wireless HART stellt eine drahtlose Adaption unterhalb der Applikationsschicht (genannt „User Layer“) dar, die kompatibel mit herkömmlichen HART-Geräten ist [35]. Die *PHY*-Schicht und das Paketformat der *MAC*-Schicht sind kompatibel zum Standard IEEE 802.15.4. Daher benutzt die *PHY*-Schicht *DSSS*[9]-Modulation und kommuniziert in den 2.4 GHz *ISM*-Bändern. Es wird jedoch zusätzlich *Channel Hopping* über 15 der 16 spezifizierten Frequenzkanäle verwendet.

Das Kernstück von wireless HART bildet das Protokoll *TSMP*[10] von Dust Networks. Dieses Protokoll erlaubt den Aufbau selbstorganisierender und selbstheilender Mesh-Netzwerke mit redundanten Datenpfaden, die bis zu 100.000 Knoten umfassen können.

TSMP ist ein *TDMA*[11]-basiertes Verfahren, alle Knoten müssen auf 1 ms genau synchronisiert sein. Die Zeit wird in Zeitschlitze unterteilt, in die jede Transaktion passen muss ($\sim$ 10 ms) [58]. Alle Knoten müssen sich auf einen absoluten Zeitschlitz-Index *ASN*[12] einigen, der die Anzahl der Zeitschlitze seit dem Beginn des Netzwerkes angibt. Zur ständigen Synchronisation wird jedes Paket so nah wie möglich am Beginn des Zeitschlitzes gesendet. Jeder Empfänger erkennt den zeitlichen Versatz zum Beginn des Zeitschlitzes in seinem Zeitrahmen und übermittelt diese Information in einem *ACK*[13]-Paket an den Sender. Der Sender kann sich mit dieser Information synchronisieren.

Zusätzlich zur zeitlichen Einteilung werden auch mehrere Frequenzkanäle genutzt. Die Kombination von Zeitschlitzen und Frequenzen ergibt eine Matrix aus sog. Zellen. Übertragungen in verschiedenen Zellen sind voneinander unabhängig.

Diesen Zellen werden sog. *Links* zugeordnet, die eine gerichtete Verbindung zwischen mindestens zwei Knoten darstellen. Ein *Link* kann dazu dediziert

[9] Direct Sequence Spread Spectrum - *Modulationsform, bei der jedes Bit durch einen bestimmten Code bzw. dessen Invertierung übertragen wird.*

[10] Time Synchronized Mesh Protocol - *Synchronisations- und Mesh-Netz"-werk"-proto"-koll*

[11] Time Division Multiple Access - *Medienzugriffssteuerung auf Basis von Zeitschlitzen*

[12] Absolute Slot Number - *Netzwerkweit bekannte Nummer eines Zeitschlitzes im TSMP-Protokoll*

[13] Acknowledgement - *Bestätigungsnachricht über ein empfangenes Paket*

einem Sender und einem Empfänger zugeordnet werden, kann aber auch zwischen mehreren Sendern per *Slotted ALOHA* verteilt werden oder für einen Broadcast genutzt werden. Je nach Art des *Links* wachen die entsprechenden Knoten für die Dauer dieses *Links* auf. Anhand einer pseudo-zufälligen Folge wird der genutzte Frequenzkanal durch

$$\text{Kanal} = \text{LUT}((ASN + \text{offset}) \,\text{modulo}\, N_{\text{Kanäle}}) \tag{2.5}$$

bestimmt. Dabei ist *ASN* der oben erwähnte Index des Zeitslots, $N_{\text{Kanäle}}$ die Anzahl der zur Verfügung stehenden Kanäle, offset der y-Wert des *Links* in der Zeitslot-Kanal-Matrix (also der „nominelle“ Kanal), und LUT eine *Look Up Table* mit einer statischen, pseudo-zufälligen Zuordnung.

Diese *Links* sind einem *Superframe* zugeordnet. Der *Superframe* besteht aus mehreren Zeitschlitzen und wird zyklisch wiederholt. Dabei kann jeder *Superframe* eine beliebige, konstante Länge (in Anzahl von Zeitschlitzen) haben. Ein *Superframe* bildet einen gerichteten Graphen auf die *TDMA*-Struktur ab. Jede Kante dieses Graphen wird auf einen *Link* abgebildet.

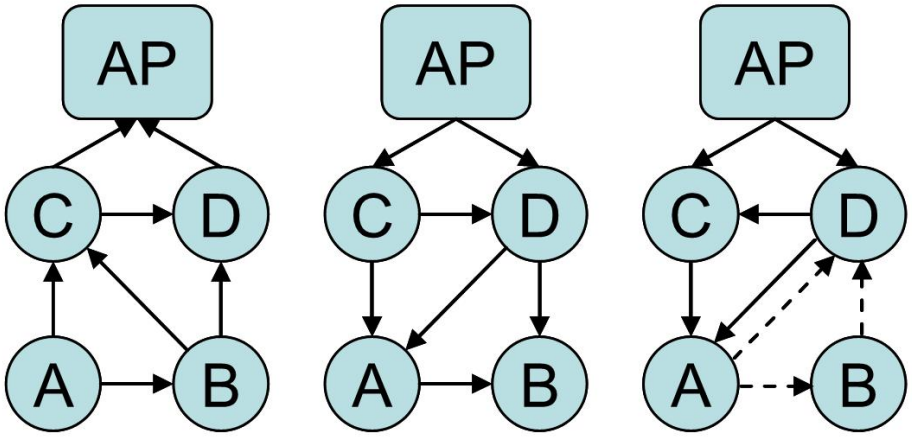


Abbildung 2.4: Beispiele für Graphen in wireless HART [58]

Abbildung 2.4 (aus [58]) zeigt drei Beispiele für Graphen in wireless HART in einem kleinen Netz aus vier Knoten und einem zentralen Zugangspunkt, genannt AP.

Der linke Graph ist ein sog. *Upstream*-Graph, er sammelt Daten von allen Knoten zum AP. Der mittlere Graph ist entgegengerichtet, ein sog. *Downstream*-Graph, er dient als Broadcast vom AP zu allen Knoten. Das rechte Bild zeigt einen Multicast-Graphen vom AP zu den drei Knoten C, D und A und ein *Peer-To-Peer-Mesh* von A nach D.

Jeder Knoten kann dadurch Teil mehrerer Graphen sein. Wenn die Quelle eines Graphen ein Paket versendet, versieht es die Applikationsschicht, abhängig vom Ziel und der geforderten Dienstgüte, mit einer Graph ID. Diese leitet das Paket durch die Multi-Hop-Route. Da bei manchen Knoten der Graph mehrere nächste Hops vorsieht, wird das Paket automatisch über mehrere redundante Routen versendet.

Aufgrund der durch die *PHY*-Schicht bedingten langen Zeitschlitze eignet sich wireless HART nicht für die Fertigungsautomatisierung. Sein Einsatzgebiet ist vor allem Prozessautomatisierung.

2.2.2.3 Industrial WLAN

Für wireless LAN (IEEE 802.11) gibt es mehrere proprietäre Erweiterungen, die von den Herstellern aber nicht veröffentlicht wurden. Dabei handelt es sich um Mechanismen, welche die auf *CSMA*[14] basierende *MAC*-Schicht um die Fähigkeit für deterministischen, reservierten Kanalzugriff erweitern.

Diese Erweiterungen bieten deterministische Latenzzeiten im Bereich von 10 ms, während WLAN-typische Reichweiten von 30-50 m erreichbar sind.

Eine IWLAN-Implementierung der Siemens AG mit 16 *Clients* und einer vorgegebenen Zykluszeit von 50 ms erreicht eine mittlere Reaktionszeit von 140 ms [55]. Dabei wurde eine „iPCF“ genannte Abwandlung der PCF-*MAC*-Schicht des Standards IEEE 802.11 benutzt um deterministische Zugriffszeiten zu ermöglichen. Um iPCF einsetzen zu können, muss ein homogenes Netz aus iPCF-fähigen Geräten vorliegen.

[14] Carrier Sensing Multiple Access - *Verteiltes Zugriffsverfahren, bei dem nur bei freiem Träger gesendet wird.*

Eine Alternative, die vor allem in heterogenen WLAN-Netzen einsetzbar ist, trägt den Namen „iQoS“. Diese Technologie funktioniert auch mit standardkonformen WLAN-Geräten. Per Konfiguration des *Access Points* kann bestimmten Teilnehmern eine feste Bandbreite zugeteilt werden. Dazu werden in regelmäßigen Zyklen (typischerweise 50 ms) Zeitschlitze für diese Teilnehmer reserviert. Die restliche Zeit steht über *CSMA* allen Teilnehmern zur Verfügung. Dabei sollten nicht mehr als vier Teilnehmern eine reservierte Bandbreite zugeteilt werden [55].

Durch die geringe Anzahl der Teilnehmer und die im Vergleich hohen Zykluszeiten eignet sich industrial WLAN nicht für die Sensor/Aktor-Ebene. Typischerweise wird es in der Geräteebene und in Feldbussen eingesetzt.

2.2.3 Ultra-Breitband-Kommunikation

UWB-Kommunikation ist ein vielversprechender Ansatz für die digitale Drahtloskommunikation und bildet die Grundlage für das in dieser Dissertation vorgestellte Funksystem. In diesem Kapitel wird ein Überblick über das Thema *UWB* gegeben. Es wird dabei ausschließlich auf die Nutzung zur Kommunikation eingegangen, *UWB* ist aber auch eine aufstrebende Technologie für Radarsysteme und bildgebende Verfahren.

2.2.3.1 Herkunft und Geschichte von UWB

Der Ende der 1980er Jahre vom US-amerikanischen *Department of Defence* geprägte Begriff *„Ultra Wideband“* beschreibt eine Funktechnologie, die zuvor unter verschiedenen anderen Namen bekannt war [9]: *Impulse Radio*, *Carrier-Free Radio*, *Baseband Communications*, *Time Domain*, *Nonsinusoidal Communications*, *Orthogonal Functions* und *Large-Relative-Bandwidth Radio/Radar Signals*.

Die ersten (modernen) Vorstöße, die nach heutiger Ansicht *UWB*-Technologien sind, wurden Ende der 1960er Jahre vor allem von Dr. Gerald Ross von der Sperry Rand Corporation gemacht [26]. Über die letzten 50 Jahre wurde die *UWB*-Technologie weiterentwickelt, so wurden zwischen 1960 und 1999 über

2000 technische Aufsätze in Fachzeitschriften abgedruckt und über 100 US-Patente zu *UWB* erteilt [65]. Da es aber noch keine offizielle Regulierung (s. Kapitel 2.2.3.3 auf Seite 31) für die verwendeten Frequenzbereiche gab, wurden diese ersten Entwicklungen ohne Ausblick auf kommerzielle Produkte unternommen [64].

Im Februar 2002 veröffentlichte die *FCC*[15] die weltweit ersten Regulierungsvorgaben für *UWB*-Systeme [22] ,siehe auch Abschnitt 2.2.3.3, und ermöglichte damit die kommerzielle Nutzung von *UWB*. Die strengen Vorgaben definierten zwei Anwendungsfelder für *UWB*-Kommunikation, für die jeweils eine *Task Group* in der IEEE gegründet wurde [6]: Die IEEE TG 802.15.3a, die *UWB* für hohe Bit-Raten über kurze Reichweiten und IEEE TG 802.15.4a für niedrige Bit-Raten über mittlere Reichweiten.

Von vielen vorgeschlagenen *PHY*-Schichten in der IEEE TG 802.15.3a standen zwei Kandidaten zur endgültigen Auswahl: *DS-UWB*[16] und *MB-OFDM*[17], die jeweils von einem Industrie-Konsortium[18] unterstützt wurden. Die TG 820.15.3a löste sich als Folge eines andauernden „Standardisierungskrieges“ ohne Ergebnis auf. Die WiMedia Befürworter erreichten allerdings bei der ECMA eine Standardisierung ihres Vorschlages unter ECMA368. Der Vorschlag der WiMedia Alliance wird in Abschnitt 2.2.3.4 auf Seite 36 genauer vorgestellt.

Die IEEE TG 802.15.4a verabschiedete 2007 den Standard IEEE 802.15.4a [3], der in Abschnitt 2.2.3.9 auf Seite 44 genauer vorgestellt wird.

2.2.3.2 Das UWB-Prinzip

UWB ist ein Ansatz zur effizienten Nutzung sehr großer Bandbreiten. Der Kerngedanke der *UWB*-Technik ist, das bereits an viele schmalbandige Dienste vergebene Spektrum nochmals zu verwenden und somit die Gesamtausnutzung zu steigern. Dabei wird eine minimale Sendeleistungsdichte (auch *PSD*[19] genannt) verwendet, sodass es zu keiner störenden Beeinflussung der primären

[15] Federal Communications Commission - *US-amerikanische Regulierungsbehörde*
[16] Direct-Sequence UWB - *Zeitlich gespreiztes Übertragungsverfahren*
[17] Multi-Band Orthogonal Frequency Multiplexing - *UWB-Frequenzspreizverfahren*
[18] *UWB* Forum bei *DS-UWB* und WiMedia Alliance bei MB-OFDM
[19] Power Spectral Density - *Energiedichte in Frequenzdomäne*

Dienste kommt. Der *UWB*-Dienst erhöht die bereits vorhandene Rauschleistungsdichte nur minimal [68]. Dieses Prinzip der nochmaligen Nutzung des Bandes wird auch *„Underlay-Technologie“* genannt. Anstelle ein schmales Frequenzband exklusiv zu nutzen, wird ein sehr breites Frequenzband, in dem auch andere Funksysteme aktiv sein können, genutzt. Dazu wird die maximale spektrale Energiedichte sehr gering gehalten. Die Energie wird also über ein breites Frequenzband verteilt.

Der Empfänger kann die empfangene Energie über dieses breite Frequenzband integrieren und damit die gesendeten Daten empfangen. Durch den Signalpegel unterhalb der Rauschschwelle sieht für schmalbandigen Funkverkehr eine *UWB*-Übertragung lediglich wie weißes Rauschen aus. Dadurch ist die Kommunikation durch andere Funksysteme nur schwer detektier- und störbar.

Weil die Übertragung über eine sehr große Bandbreite stattfindet, ist sie sehr robust gegenüber bestimmten schmalbandigen bzw. frequenzselektiven Kanaleffekten. Zu diesen gehören unter anderem:

- Selective Fading :
  Da von *Frequency Selective Fading* nur bestimmte Frequenzen betroffen sind und *UWB* ein sehr breites Spektrum für die Übertragung nutzt, ist *UWB*-Kommunikation relativ unempfindlich gegenüber selektivem Fading auf einzelnen Frequenzen.

- Störsender (*Jamming*) :
  Durch die große genutzte Bandbreite ist ein gezieltes Stören einer Übertragung erheblich aufwändiger als bei schmalbandigen Übertragungen, da ein viel breiteres Spektrum mit einem Störsignal belegt werden muss.

- Abhörsicherheit :
  Da eine *UWB*-Übertragung für andere Funkteilnehmer wie Rauschen aussieht, ist es schwierig, eine solche Übertragung ohne Kenntnis der Netzparameter abzuhören.

Die Ausnutzung einer hohen Bandbreite erhöht erheblich die maximale Kanalkapazität nach der Shannon-Formel (Formel (2.6)). Diese beschreibt die maximale Kapazität eines Kanals, die linear von der Bandbreite B und loga-

rithmisch von dem Signal-Rausch-Abstand $\frac{S}{N}$ (engl. SNR[20]) abhängt. Dadurch steigert eine Erweiterung der Bandbreite die Kanalkapazität erheblich mehr als eine Erhöhung der Sendeenergie.

$$C = B \log_2(1 + \frac{S}{N}) \tag{2.6}$$

Diese hohe Kanalkapazität kann, je nach Anwendung, für eine hohe Anzahl möglicher Nutzer, eine hohe Datenrate pro Nutzer oder eine größere Reichweite (durch *FEC*-Kodierung) benutzt werden.

Allerdings wirkt diesen Vorteilen die geringe verwendbare Sendeleistung entgegen, wodurch die Kanalkapazität von *UWB*-Systemen wieder eingeschränkt wird. Dies ist vor allem an der Reichweite bemerkbar, weil die *SNR* mit der Entfernung logarithmisch fällt.

2.2.3.3 Regulierungsvorschriften bei UWB

Die Regulierung von *UWB* stellte die Regulierungsgremien vor erhebliche Probleme [26]. Zum ersten Mal sollten Bänder zur parallelen Nutzung durch verschiedene Funksysteme freigegeben werden, wobei die „eigentlichen“ Nutzer (auch Primärsystem genannt) nicht durch die *UWB*-Kommunikation gestört werden dürfen.

US-amerikanische Regulierung der FCC

Am 14. Februar 2002 erließ die *FCC* einen „First Report and Order“ [22], der die freie Nutzung von *UWB*-Funk erlaubt und setzte damit den Startschuss für die kommerzielle Nutzung.

In dieser Regulierung der *FCC* gilt ein Signal dann als *UWB*-Signal, wenn es der *UWB*-Definition der *DARPA*[21] entspricht, dies ist gegeben, wenn die *Fractional Bandwidth* B_f größer als 20% ist. Die *Fractional Bandwidth* wird nach Formel (2.7) berechnet, wobei f_H und f_L die oberen und unteren 10-

[20] Signal to Noise Ratio - *Signal-zu-Rauschen-Verhältnis*

[21] Defence Advanced Research Projects Agency - *Einrichtung der US-ame"-ri"-ka"-nischen Rüstungsforschung*

dB-Grenzen des Signals um die Frequenz mit der höchsten Abstrahlung (f_M) sind[22].

$$B_f = 2 \cdot \frac{f_H - f_L}{f_H + f_L} \tag{2.7}$$

Zusätzlich zu dieser Definition gelten die *UWB*-Vorschriften auch für ein Signal, bei dem die spektrale Bandbreite des Signals 500 MHz oder mehr beträgt [22].

Die Regulierung legt durchschnittliche und absolute Höchstwerte fest. Als absoluter Höchstwert, der innerhalb von 50 MHz um f_M erlaubt ist, sind 0 dBm festgelegt. Der durchschnittliche Höchstwert ist in dBm pro MHz (*RMS*[23] über 1 ms) angegeben und hängt von f_M und der Applikation ab.

Folgende *UWB*-Systeme sind beschrieben (vgl. [14]):

- Ground Penetrating Radars:
  Systeme zur Untersuchung von Böden. Diese werden lizenziert für öffentliche Sicherheit, Wissenschaft und für kommerziellen Bergbau und Bauvorhaben. Sie arbeiten mit *UWB*-Bandbreiten unterhalb von 10,6 GHz.

- Through D-Wall Imaging Systems:
  Systeme zum Durchleuchten von Wänden. Diese sind lizenzpflichtig und werden nur für staatliche oder lokale Organe der öffentlichen Sicherheit lizenziert. Dabei unterscheidet man Systemen, die unterhalb von 960 MHz operieren und jenen, die zwischen 1,9 und 10,6 GHz arbeiten.

- Surveillance Systems:
  Überwachungssysteme mit einer *UWB*-Bandbreite zwischen 1,9 und 10,6 GHz. Die Lizenzen werden nur an Organe der öffentlichen Sicherheit sowie Öl- und Energieversorger vergeben.

- Medical Imaging Systems:
  Medizinische Bildgebungsgeräte arbeiten im Spektrum von 3,1 bis 10,6

[22] Dies bedeutet, dass die empfangene Signalstärke bei f_H und $f_L$um 10 dB niedriger ist als die höchste ausgestrahlte Signalstärke.

[23] Root Mean Square - *Effektivwert*

GHz. Sie dürfen ausschließlich von medizinischem Fachpersonal bzw. unter deren Aufsicht betrieben werden.

- Vehicular Radar Systems:
  Diese Kfz-Radarsysteme dürfen nur betrieben werden, solange der Motor des Fahrzeuges läuft. Sie werden in Frequenzen zwischen 22 und 29 GHz mit einer Mittelfrequenz $f_C > 24,075$ GHz betrieben. Bei einem Abstrahlwinkel, der höher als 30° ist, werden die Abstrahlgrenzwerte um 25 dB (35 dB ab 2014) gesenkt.

- Indoor UWB Systems:
  Diese Geräteklasse ist nur für den Betrieb im Innenraum zugelassen. Es darf nicht möglich sein, diese Geräte im Freien zu benutzen, das Signal darf auch nicht nach Außen gelangen. Wie dies sicherzustellen ist, wurde in der Regulierung nicht festgelegt. Es gibt verschiedene Ansätze zur Ermittlung ob sich das Gerät im Freien befindet. Einfachere Verfahren weisen den Nutzer auf die rechtliche Situation hin. Es gibt aber auch Lösungen, die auf GPS-Empfang als Kennzeichen für den Aufenthalt im Freien basieren. Die Geräte dürfen außerdem nur zu einem assoziierten Transceiver übertragen. Zu den Applikationen gehören Kabelersatzsysteme, wie z.B.Wireless USB. Auf die erlaubte Sendeleistung dieser Klasse wird im Anschluss eingegangen.

- Hand Held UWB Devices:
  Diese Geräteklasse muss in sich abgeschlossen sein (inkl. der Antenne), d.h. ohne Infrastruktur auskommen. Auch sie dürfen nur senden falls ein Empfänger assoziiert ist. Auf die erlaubte Sendeleistung dieser Klasse wird im Anschluss eingegangen. Zielapplikationen sind portable Geräte, die auch im Freien betrieben werden dürfen.

Für *UWB*-Kommunikation sind hauptsächlich die letzten beiden Geräteklassen interessant.

Die Spektralmasken für diese Geräteklassen zeigen einen maximalen durchschnittlichen Höchstwert von -41,3 dBm/MHz (*RMS* pro ms) für die Frequenzen von 3,1 bis 10,6 GHz. In Abbildung 2.5 gezeigt markiert die grüne Linie die Spektralmaske für die Klasse der *Handheld UWB Devices* und die rote Linie

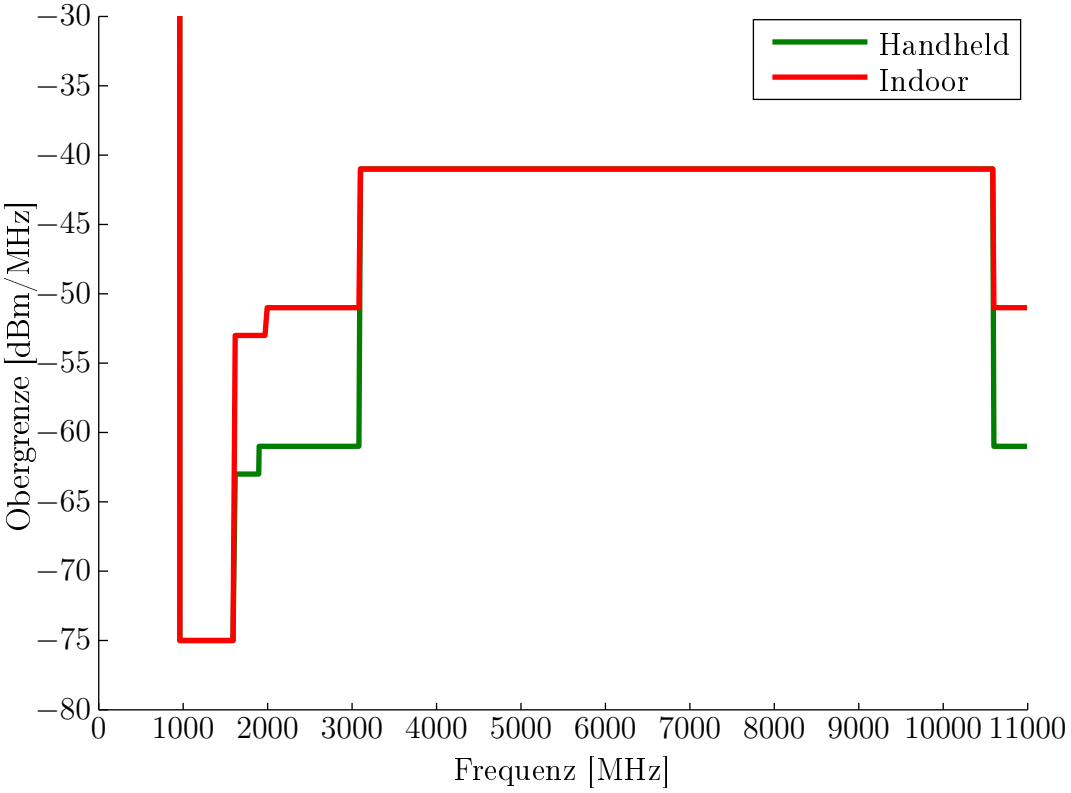


Abbildung 2.5: Spektrale Masken für UWB-Kommunikation der *Handheld*- und *Indoor-Device*-Klasse nach der FCC-Regulierung

die Klasse der *Indoor UWB Devices*. Es ist erkennbar, das für die *Handheld Devices* strengere Richtwerte abseits des Hauptfrequenzbereiches gelten. Der tiefe Maximalwert zwischen 1 und 2 GHz verhindert eine Überdeckung des GPS-Signals durch *UWB*-Geräte.

Europäische Regulierung des ETSI

Das *ETSI*[24] hat im Februar 2002 mit der EN 302 065 [21] einen harmonisierten europäischen Standard für *UWB*-Kommunikation veröffentlicht. Dieser definiert die operative Bandbreite (*Operating Bandwidth*) einer Übertragung als das Frequenzband, unterhalb dessen niedrigster (f_L) und oberhalb dessen höchster (f_H) Randfrequenz die Übertragungsleistung jeweils nur bis zu 0,5% der totalen Übertragungsleistung dieser Übertragung beträgt, vgl. Formel

[24]European Telecommunications Standards Institute - *Europäisches Regulierungsorgan*

(2.8). Zur Vereinfachung der Messungen dürfen die -23 dB Punkte als Ober- und Untergrenze verwendet werden.

$$\sum_{-\infty}^{f_L} P \leq 0,005 \cdot \sum_{-\infty}^{\infty} P \vee \sum_{f_H}^{\infty} P \leq 0,005 \cdot \sum_{-\infty}^{\infty} P \tag{2.8}$$

Diese operative Bandbreite muss größer als 500 MHz sein, damit die Übertragung unter die regulatorischen Vorschriften von [21] fällt.

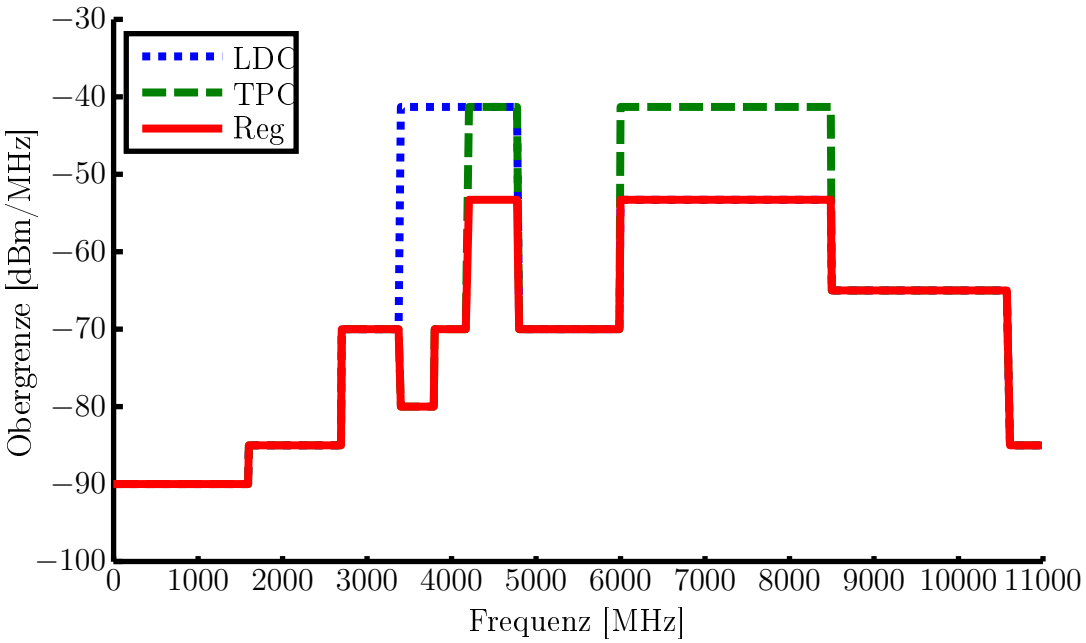


Abbildung 2.6: Spektralmaske der ETSI für UWB-Kommunikation

Die maximale durchschnittliche *PSD* ist für Teile des Spektrums in Abhängigkeit der verwendeten Koexistenz-Technologien angegeben, wie Abbildung 2.6 zeigt. *LDC*[25] und *TPC*[26] stehen dabei für die jeweilige Koexistenz-Technologie und „Reg“ für eine Übertragung ohne Koexistenz-Technologie.

So ermöglicht die Nutzung eines *LDC* die Nutzung der Frequenzen zwischen 3,2 und 4,8 GHz mit bis zu -41,3 dBm/MHz, während sich dieses Frequenzband ohne *LDC* nur mit viel geringerem *PSD* nutzen lässt.

[25] Low Duty Cycle - *geringer Aktivierungszyklus*
[26] Transmit Power Control - *geregelte Sendeleistung*

Der *LDC* ist definiert wie in Tabelle 2.3 angegeben.

Tabelle 2.3: Low-Duty-Cycle-Einschränkungen der ETSI

LDC Parameter	**Wert**
Maximale Sendezeit	$\leq 5\,\text{ms}$
Minimale durchschnittliche Sendepause (Durchschnitt über eine Sekunde)	$\geq 38\,\text{ms}$
minimale akkumulierte Sendepause ($\sum$Tx off)	$\geq 950\,\text{ms}$ / s
maximale akkumulierte Sendezeit ($\sum$Tx on)	18 s / h

Eine *TPC* mit einer Regelung von 12 dB ist vorgeschrieben, um die höhere Spektralenergiedichte von -41,3 dBm/MHz in dem Band von 6 bis 8,5 GHz nutzen zu können. Ansonsten ist ein Höchstwert von -53,3 dBm/MHz einzuhalten. Für Geräte zur Benutzung in Straßen- oder Schienenfahrzeugen ist generell eine *TPC* vorgeschrieben.

Für impulsbasierte Kommunikation (vgl. Kapitel 2.2.3.5 auf Seite 38) schreibt die *ETSI* außerdem eine *PRR*[27] von mindestens 1 MHz vor.

In Deutschland wird die *ETSI*-Norm durch die „Allgemeinzuteilung von Frequenzen für die Nutzung durch Anwendungen geringer Leistung der Ultra-Wideband (UWB) Technologie“ [17] umgesetzt, wobei zusätzlich eine Spektralmaske für den maximalen Spitzenwert der Leistung definiert wird, wie in Abb. 2.7 auf der nächsten Seite gezeigt.

2.2.3.4 OFDM-basierte UWB-Kommunikation

Eine Möglichkeit, die hohe Bandbreite auszunutzen, besteht darin, Frequenzspreizungen und Mehrkanalverfahren zu verwenden, wie Frequenzsprungverfahren oder *OFDM*[28]. Als Beispiel für ein solches Verfahren wird *MB-OFDM* vorgestellt.

High-Rate-*UWB* auf Basis von *MB-OFDM* wurde von der WiMedia Alliance entwickelt und in ECMA 368 [40] standardisiert. Diese Technologie teilt das vorhandene Spektrum von 3 bis 10 GHz in 15 Bänder zu jeweils 528 MHz auf.

[27] Pulse Repetition Rate - *Pulswiederholungsrate*

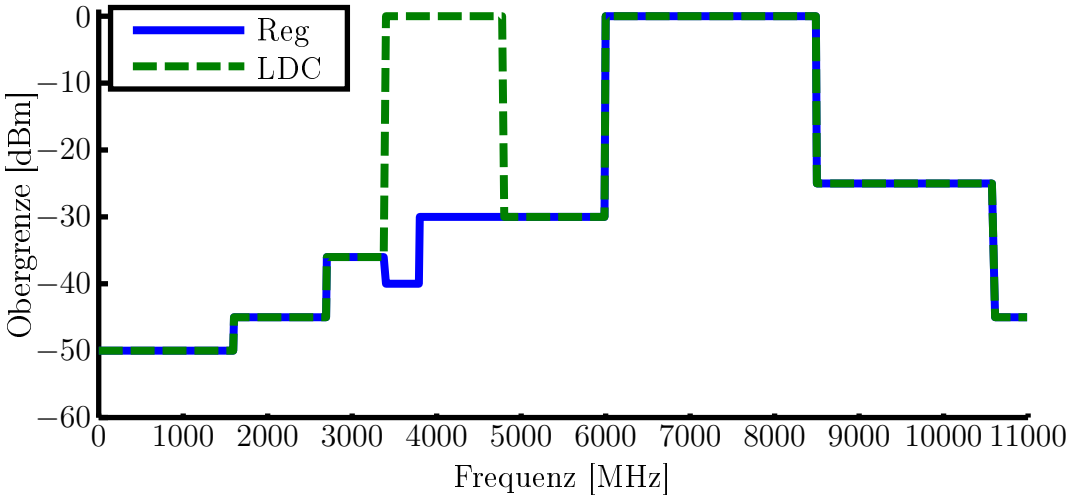


Abbildung 2.7: Maximale Spitzenwerte nach Bundesnetzagentur

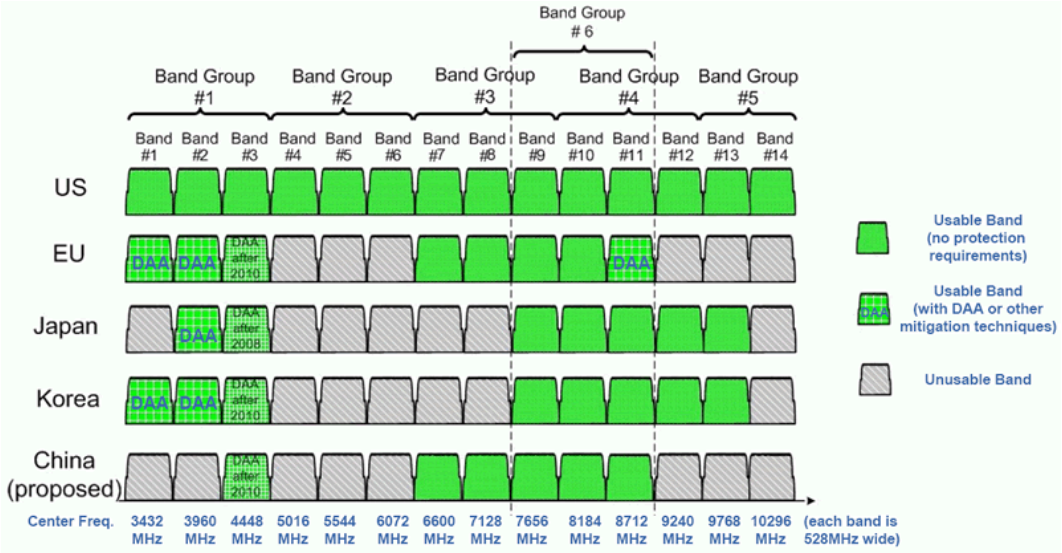


Abbildung 2.8: Einteilung in Bandgruppen nach ECMA 368 [81]

Diese Bänder werden zu vier Bandgruppen mit jeweils drei Bändern und einer Bandgruppe mit nur zwei Bändern zusammengefasst. Zusätzlich wurde eine sechste Bandgruppe definiert, die drei Bänder umfasst, um weltweite Regulationsbestimmungen abzudecken (vgl. Abb. 2.8).

Jedes WiMedia-Netz (sog. *Piconet*) benutzt eine Bandgruppe und springt innerhalb dieser Gruppe nach jedem Symbol auf ein anderes Band. Dazu folgt

jedes *Piconet* einem von zehn möglichen *TFC*[29]. Diese Codes bestehen aus sechsstelligen Folgen, die zyklisch durchlaufen werden. Die Position im Code legt das nächste Band für das Frequenzsprungverfahren fest.

Tabelle 2.4: TFCs nach ECMA 368

Code	**Band**					
1	1	2	3	1	2	3
2	1	3	2	1	3	2
3	1	1	2	2	3	3
4	1	1	3	3	2	2
5	1	1	1	1	1	1
6	2	2	2	2	2	2
7	3	3	3	3	3	3
8	1	2	1	2	1	2
9	1	3	1	3	1	3
10	2	3	2	3	2	3

Tabelle 2.4 zeigt die TFCs für Bandgruppen mit drei Bändern.

2.2.3.5 Impulsfunk

Das Impulsfunkprinzip geht auf die ursprüngliche Funktechnik von Heinrich Hertz zurück, bei der er 1888 mittels eines Induktors Funken benutzte, um eine sehr breitbandige, elektromagnetische Strahlung herzustellen [68]. Daher entstammt auch das Wort „funken“.

Beim Impulsfunk wird ein sehr kurzer elektrischer Impuls (im Bereich von Nanosekunden) auf eine Antenne angelegt und erzeugt dabei im Funkkanal ein Impulsecho.

Je nach zeitlicher Länge und der verwendeten Impulsform dieses geschalteten Impulses spreizt sich die Energie auf eine große Bandbreite auf. Die Abbildungen 2.9 auf der nächsten Seite und 2.10 auf Seite 40 zeigen (in Zeit- und

[29] Time Frequency Code - *Zeit-Frequenz-Sprungfolge*

Frequenzdomäne), wie ein kurzer Impuls in Form der 1. Ableitung der Gauß-Funktion (engl. *Gaussian Monocycle)* die Energie auf ein breites Spektrum verteilt.

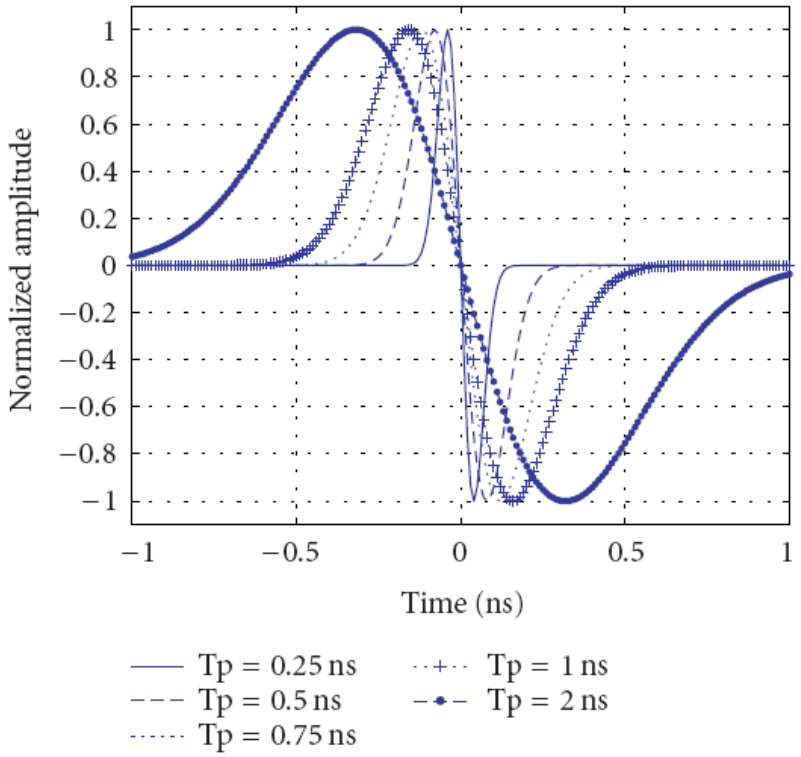


Abbildung 2.9: Gaußsche Monozyklen in der Zeitdomäne [10]

Das Impulsecho kann der Empfänger aufnehmen und so einen gesendeten Puls erkennen. Dazu kann man die erwartete Form der Impulsantwort berechnen und mit der empfangenen Impulsantwort vergleichen, wodurch sich die aufmodulierten Informationen gewinnen lassen. Damit lassen sich sowohl phasen- als auch zeitbasierte Modulationen zurückgewinnen. Dies geschieht rein in der Zeitdomäne.

Alternativ dazu kann der Empfänger das Energieniveau im Kanal messen. Anhand des Vergleichs von zwei Zeiträumen kann festgestellt werden, in welchem von beiden sich ein Puls befunden hat. Damit lassen sich allerdings nur die Informationen einer zeitbasierten Modulation zurückzugewinnen. Dies ist allerdings auch durch nichtkohärenten Empfang möglich.

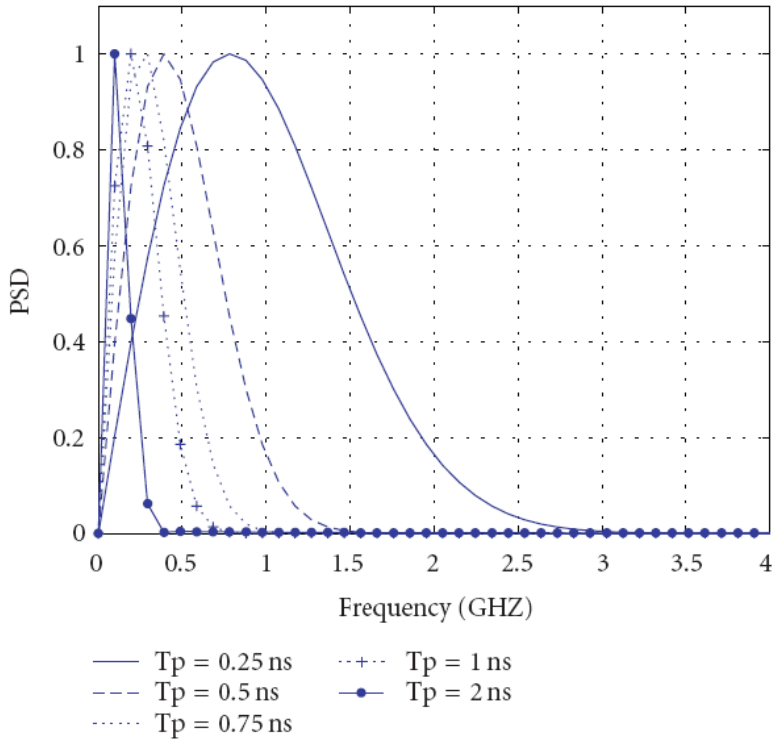


Abbildung 2.10: Gaußsche Monozyklen in der Frequenzdomäne [10]

IR-UWB bietet dadurch folgende Vorteile [10]:

- Robustheit gegen Multipfad-Effekte:
  Durch die Auflösung in der Zeitdomäne ist das erste Auftreffen eines Impulses leichter zu erkennen und von den Multipfad-Echos zu unterscheiden. Dadurch ergibt sich eine große Robustheit gegen Multipfad-Ausbreitung.

- Genaue Ankunftszeit:
  Durch eine genaue Bestimmung der Ankunftszeit eines Impulses lässt sich die *TOF*[30] sehr genau bestimmen, wodurch sich die Entfernung zwischen Sender und Empfänger bestimmen lässt.

- Geringe Komplexität:
  Da die Impulse direkt auf das Basisband gesendet werden und aus zeitlicher Auflösung der Kanalenergie ermittelt werden können, ist die ver-

[30] Time of flight - *Flugzeit*

wendete Hardware erheblich einfacher im Vergleich zu herkömmlichen (sinosoidalen) Funksystemen, bei denen das Signal aufwendig heruntergemischt und Frequenz- oder Phasenverschiebungen korrigiert werden müssen.

2.2.3.6 Modulation von Impulse Radio

Um Informationen zu übertragen, wird eine Folge von Pulsen verwendet, sog. *Pulse Trains* oder auch *Burst*s. Die Informationen werden mittels verschiedener Basisband-Modulationen aufmoduliert.

PPM[31] moduliert Informationen dadurch auf, dass jeder Puls von seiner nominellen Position zeitlich nach vorne oder hinten abweicht. Dadurch trägt jeder Puls, der früher eintrifft, die Information „0“ und jeder, der später eintrifft, die Information „1“. Dies erfordert eine sehr genaue Synchronisation. Eine Variante davon, die *BPM*[32], moduliert eine Information über die Position einer ganzen Folge von Pulsen. Dieser *Burst* kann entweder in der ersten oder zweiten Hälfte eines Symbol-Frames stehen, was ebenfalls wieder „0“ oder „1“ bedeutet [3, 54].

BPSK moduliert binäre Informationen durch Umkehren der Phase des Pulses. Bei bestimmten Pulsformen ist der Empfang diese Modulation nur mit kohärentem Empfang möglich. Entscheidend ist dabei der Betrag des Integrals eines Pulses, da dabei erkennbar wird, in wie weit sich Komponenten eines Pulses in der Integration auslöschen.

PAM[33] moduliert Informationen durch die Höhe der Amplitude und ist dadurch systembedingt sehr anfällig auf Rauschen.

OOK[34] lässt einzelne Pulse aus, um zwischen „0“ und „1“ zu unterscheiden. Das bringt den Vorteil, dass weniger gesendete Energie verwendet wird, wodurch sich das *link budget* besser ausnutzen lässt. Der Nachteil ist, dass die Synchronisation, besonders bei längeren „0“-Ketten schwierig ist.

[31] *Pulse Position Modulation*
[32] *Burst Position Modulation*
[33] *Pulse Amplitude Modulation*
[34] *On-Off Keying*

2.2.3.7 Time Hopping

Bei einer gleichmäßigen Wiederholung der Impulse bilden sich periodische Energiespitzen, sog. *Comb Lines* (s. [10, S.67]). Diese Linien sind bedingt durch die gleichmäßige Periodizität des Signals. Wie Abbildung 2.11 anhand des Spektralbildes einer simplen *Pulse Train* zeigt.

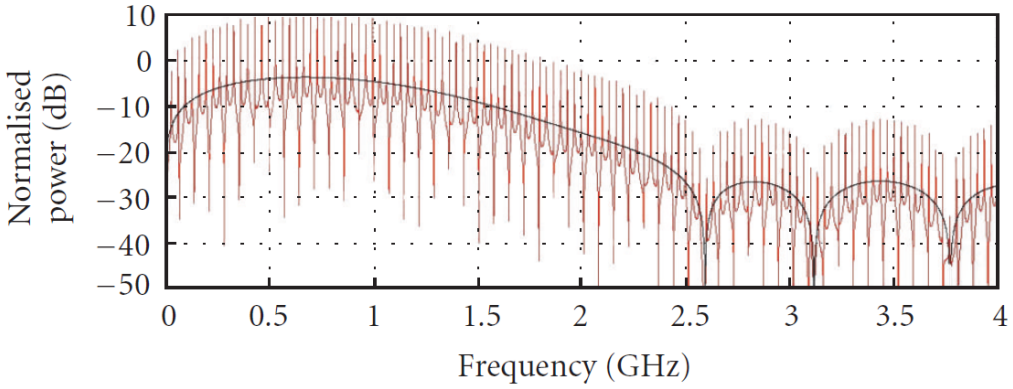


Abbildung 2.11: Spektrale Darstellung einer UWB-Übertragung ohne Zufallsmechanismen [10]

Deshalb werden bei einem *IR-UWB*-Signal durch Techniken wie *DS-UWB* oder *TH*[35] künstlich Unregelmäßigkeiten erzeugt. Dadurch wird die Spektrallinie geglättet und das Signal ähnelt mehr einem Hintergundrauschen, wie Abbildung 2.12 auf der nächsten Seite zeigt [10].

Bei *TH-UWB* werden die Positionen für einzelne Pulse bzw. *Burst*s durch eine sog. *THS*[36] festgelegt. Diese Sequenz hat eine geringe Autokorrelation und dient einerseits dazu, wie oben beschrieben, Leistungsspitzen zu vermeiden und andererseits die Kollisionswahrscheinlichkeit der Übertragungen mehrerer Sender zu verringern.

Abbildung 2.13 auf der nächsten Seite zeigt beispielhaft den Zugriff zweier Sender (rot und blau) mit verschiedenen *THS* oder zweier Sender mit derselben *THS*, aber verschiedenen Startzeitpunkten. Durch die Zufallsfolge sind Kollisionen (wie in Zeitabschnitt 4) sehr unwahrscheinlich.

[35] Time Hopping - *Zeitsprungverfahren*
[36] *Time-Hopping-Sequenz*

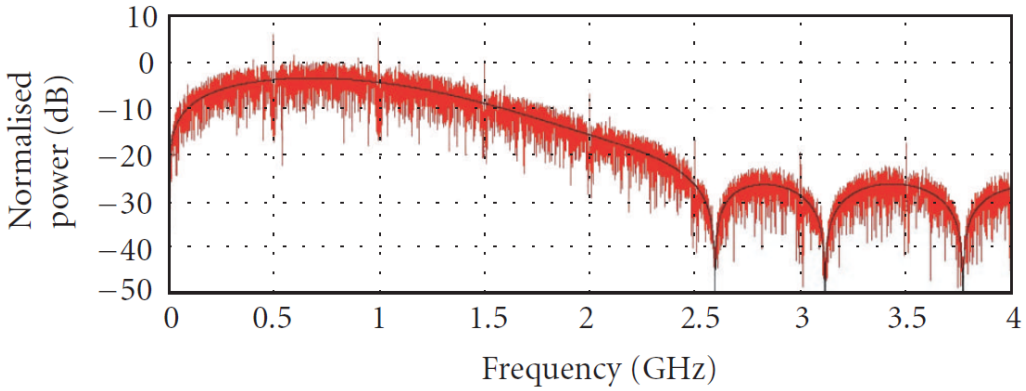


Abbildung 2.12: Spektrale Darstellung einer UWB-Übertragung mit Zufallsmechanismen [10]

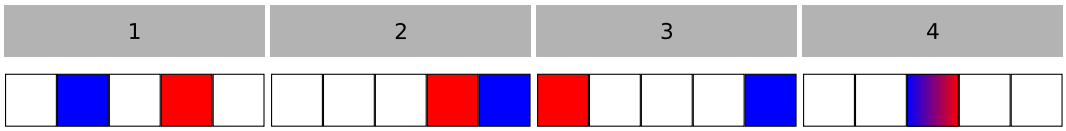


Abbildung 2.13: Multiple Access mit Time Hopping

Die Generierung der *THS* erfolgt bei *TH-UWB* (z.B. bei [3, 39]) über sog. *Pseudonoise Codes*[37]. Diese erzeugen eine Sequenz, die sehr nah an weißem Rauschen liegt. Alternativ dazu werden auch pseudo-chaotische Folgen untersucht (s. [73]).

2.2.3.8 Pulse Repetition Rate

Den zeitlichen Abstand zwischen zwei aufeinanderfolgenden Pulsen nennt man *PRP*[38]. Dabei wird der nominelle Abstand zweier Pulse zugrunde gelegt, also ohne *PPM* oder *TH*. Das Inverse davon (s. Formel (2.9)) definiert die Rate,

[37] Pseudo-zufällige Folgen
[38] Pulse Repetition Period - *Periode zwischen zwei Pulsen*

in der Pulse gesendet werden, genannt PRR.

$$PRR = \frac{1}{PRP} \tag{2.9}$$

Da die Vorschriften der Regulierung die PSD auf 1 ms mitteln, ist die maximale Energie pro Puls P_P direkt von der PRR abhängig. Formel (2.12) beschreibt diese Abhängigkeit im Verhältnis zu einer maximal möglichen PRR (PRR_{max}), die durch das Inverse der Pulslänge definiert ist.

$$PRR_{max} = \frac{1}{PRP_{min}} = \frac{1}{T_p} \tag{2.10}$$

$$PSD_{max} = -41,3\,\text{dBm/MHz} = 74.1\,\text{nW/MHz} \tag{2.11}$$

$$P_P = \max(74.1\frac{\text{nW}}{\text{MHz}} \cdot \text{W} \cdot \frac{PRR_{max}}{PRR}, 1\,\text{mW}) \tag{2.12}$$

Da die Regulierung die PSD pro MHz vorschreibt, ist die Energie pro Puls ebenfalls von dem Spektrum (W), auf das die Energie verteilt wird, abhängig.

Die Energie pro Puls bedingt die Reichweite, wodurch die Reichweite mit steigender Pulsrate abnimmt. Da die maximal erreichbare Datenrate von der Pulsrate abhängig ist, bedeutet dies, dass hohe Datenraten nur auf kurzer Reichweite erreicht werden können, während Kommunikation mit niedrigeren Datenraten auch über größere Entfernungen möglich sind.

2.2.3.9 Die UWB-PHY-Schicht des Standards IEEE 802.15.4a

Der Standard IEEE 802.15.4a [3] ist, wie in Abschnitt 2.2.3.1 auf Seite 28 erwähnt, ausgerichtet auf *WPAN*[39]. Er ist dabei eine Erweiterung des Standards IEEE 802.15.4 [2], in der zwei neue *PHY*-Schichten definiert werden.

Die Zielsetzung des Standards war vor allem Low-Power-/Low-Complexity-Geräte herstellen zu können. Diese sollten lange Zeit (bis hin zu Jahren) mit

[39] Wireless Personal Area Networks - *Drahtlose Netze mit Reichweiten im Bereich mehrerer Meter*

einer Batterieladung auskommen können, z.B. für Sensornetze. Außerdem ist die Verbindung von Kommunikation und präzisem Ranging eines der Designziele des Standards [54].

Eine dieser *PHY*-Schichten basiert auf *IR-UWB* und wird hier genauer betrachtet, wobei der Begriff des Standards synonym mit dieser *PHY*-Schicht verwendet wird.

Details der PHY-Schicht

Die *PHY*-Schicht bildet einen Kompromiss zwischen hoher Datenrate und robuster Übertragung. Besonderes Augenmerk wurde auf geringe Anforderungen an die Transceiver gelegt. Deswegen ist nur ein sehr kleiner Teil der *PHY*-Schicht verpflichtend vorgeschrieben und große Teile optional. So können auch sehr einfache Geräte standardkonform implementiert werden. Trotzdem sind auch leistungsfähigere Geräte möglich, die bessere Features bieten.

Dazu bietet die *PHY*-Schicht verschiedene Modi an, von denen die Geräte einen Passenden aushandeln müssen. Spektral nutzt der Standard 12 Kanäle von jeweils 499,2 MHz Bandbreite, die wiederum in drei Bänder aufgeteilt sind. Diese drei Bänder sind:

1. Das Sub-Gigaherz-Band von 249,6-749,6 MHz mit einem einzigen Kanal
2. Das *Low*-Band von 3,1 GHz bis 4,8 GHz mit drei Kanälen
3. Das *High*-Band von 6,0 GHz bis 10,6 GHz mit 8 Kanälen

Zusätzlich zu diesen Kanälen sind vier breitere festgelegt:

- Kanal 4, der mit 1331,2 MHz das *Low*-Band abdeckt
- Kanal 7 mit 1081,6 MHz Bandbreite um die Mittenfrequenz f_C von 6489,6 MHz
- Kanal 11 mit 1331,2 MHz Bandbreite bei 7987,2 MHz
- Kanal 15 mit 1354,97 MHz Bandbreite bei 9484,8 MHz

Die größere Kanalbreite erlaubt eine höhere Energie pro Puls. Während bei einer maximalen *PSD* von -41,3 dBm/MHz in den 499,2 MHz breiten Kanälen maximal eine Sendeleistung von -14,3 dBm möglich ist, kann im Kanal 15

aufgrund der hohen Bandbreite eine Leistung von -9,9 dBm verwendet werden. Abbildung 2.14 skizziert die Kanäle, wobei die Y-Koordinate die maximal verwendbare Leistung darstellt.

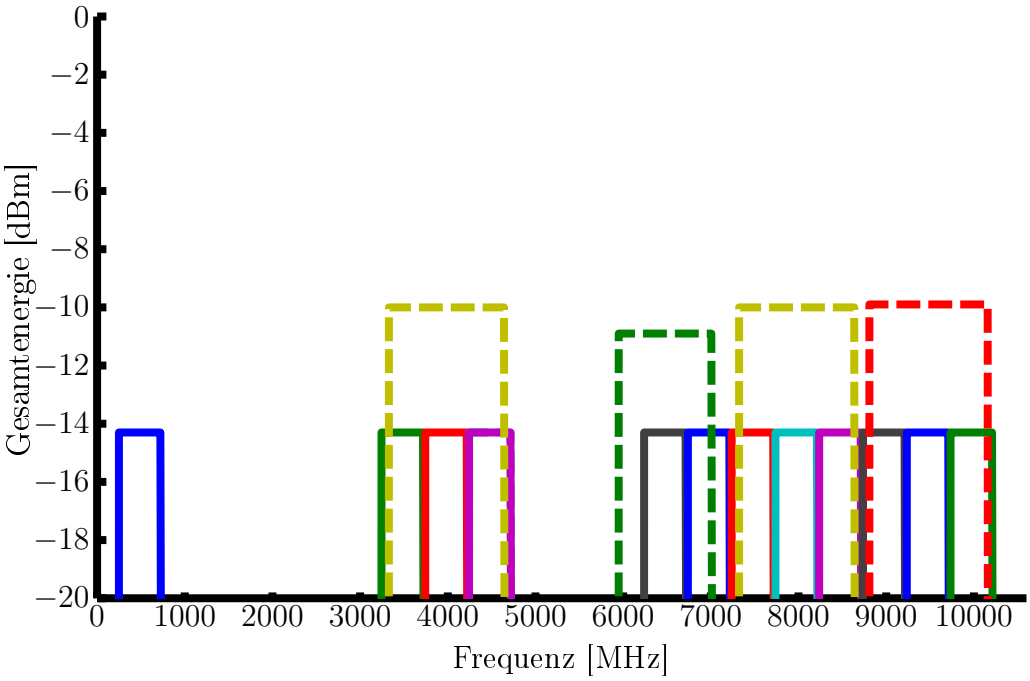


Abbildung 2.14: Kanäle in IEEE 802.15.4a

Kodierung

In bestimmten Modi werden die Nutzdaten vor dem Senden mit einem systematischen (K+8,K) *Reed-Solomon-Code* über GF(2^6) kodiert.

An den Anfang der kodierten Daten wird ein *PHY*-Header angefügt. Der *PHY*-Header ist aufgebaut wie in 2.5 auf der nächsten Seite angegeben. In dem Feld „Datenrate“ wird für die mandatorische Basisrate der Wert b01 gesetzt, für höherratige Modi des Standards sind weitere Werte definiert. Die Felder „Länge der Präambel“ und „Ranging“ betreffen dabei die Lokalisierung und werden in dieser Arbeit nicht benötigt.

Anschließend werden *PHY*-Header und Nutzdaten mit einem systematischen Faltungs-Code der Rate $R = \frac{1}{2}$ kodiert.

Die beiden Codes werden konkateniert, wobei der *Reed-Solomon-Code* als innerer Code verwendet wird, wie Abbildung 2.15 auf der nächsten Seite zeigt.

Tabelle 2.5: PHY-Header nach IEEE 802.15.4a

Feld	**Größe**	**Beschreibung**
Datenrate	2 Bit	Datenrate (Modus) des Paketes
Länge	7 Bit	Länge der Nutzlast in Byte
Ranging	1 Bit	Gibt an, ob auf das Paket in einer festgelegten Zeit mit einer Ranging-Antwort reagiert werden muss.
erweiterter Header	1 Bit	Reserviert für zukünftige Erweiterungen
Länge der Präambel	2 Bit	gibt die Länge der Präambel an (16, 64, 1024 oder 4096 Wiederholungen)
SECDED-Code	6 Bit	Prüfsumme zur Absicherung des *PHY*-Header

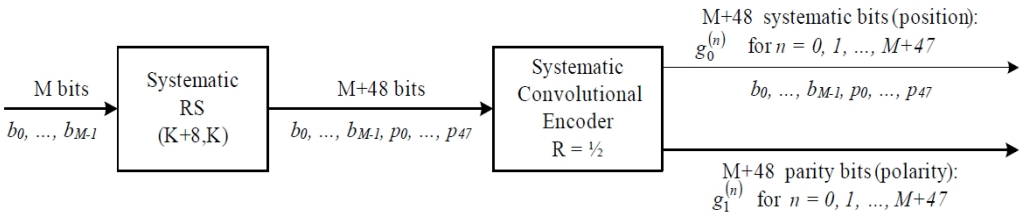


Abbildung 2.15: FEC-Kodierungsschema nach IEEE 802.15.4a [3]

Die daraus resultierenden Symbole werden durch eine Doppelmodulation von *BPM* und *BPSK* moduliert. Dabei werden die Daten des Faltungs-Codes aufgeteilt, die eigentlichen Datenbits werden durch die *BPM*-Modulation übertragen, die Redundanz-Bits des systematischen Faltungs-Codes über die *BPSK*-Modulation. Auf diese Weise kann ein einfacher, nichtkohärenter Empfänger auf Basis von Energie-Detektion die Daten durch die *BPM*-Modulation empfangen, während ein komplexerer, kohärenter Empfänger durch die *BPSK*-modulierten Redundanz-Bits einen zusätzlichen Prozessgewinn erhält.

Präambel-Modulation

Die Präambel besteht aus 16, 64, 1024 oder 4096 Wiederholungen des Prä-

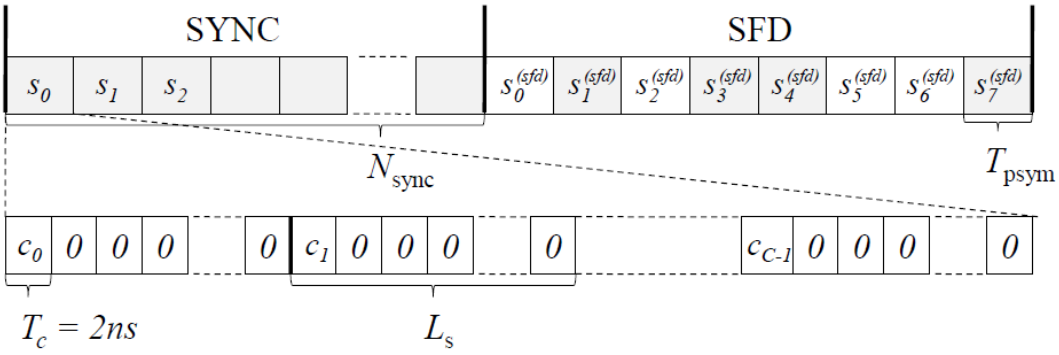


Abbildung 2.16: Strutur der Prämbel nach IEEE 802.15.4a [23]

ambel-Symbols und aus einem *SFD*[40] aus acht kodierten Präambel-Symbolen. Das Präambel-Symbol besteht aus einem ternären Code mit hoher Autokorrelation, dem sog. Präambel-Code. Es können zwei verschiedene Präambel-Code-Längen verwendet werden. Verpflichtend muss eine 31 Symbole lange Präambel unterstützt werden, optional auch eine 127 Symbole lange Präambel, die nur bei hohen Datenraten mit einer *Mean PRR* von 62,4 MHz eingesetzt wird. Der Standard definiert acht verschiedene Präambel-Codes mit 31 Symbolen und 13 Präambel-Codes mit 127 Symbolen. Alle Knoten eines Netzes benutzen den selben Präambel-Code, weshalb die längeren Präambel-Codes mehr Netze im selben Frequenzkanal erlauben.

Der Code wird gespreizt, indem zwischen jedem Code-Symbol eine definierte Anzahl von Nullsymbolen eingefügt wird. Dieser Vorgang wird als L-Spreading bezeichnet und die Anzahl der eingefügten Nullsymbole als L.

Der *SFD* besteht aus dem Code [0 +1 0 -1 +1 0 0 +1]. Jedes dieser Code-Symbole wird mit dem Präambel-Symbol multipliziert.

Modulation

Die Daten werden im Standard durch zwei parallele Modulationen übertragen: Eine verpflichtende *BPM*-Modulation mit variabler *Burst*-Länge sowie eine optionale *BPSK*-Modulation für kohärente Empfänger. Jedes Symbol wird

[40] Start of Frame Delimiter - *Synchronisationspunkt zwischen Präambel und Datenteil*

durch einen *Burst* übertragen, die *Burst*-Länge hängt von der vereinbarten Bit-Rate der Übertragung ab (Parameter N_{cpb}).

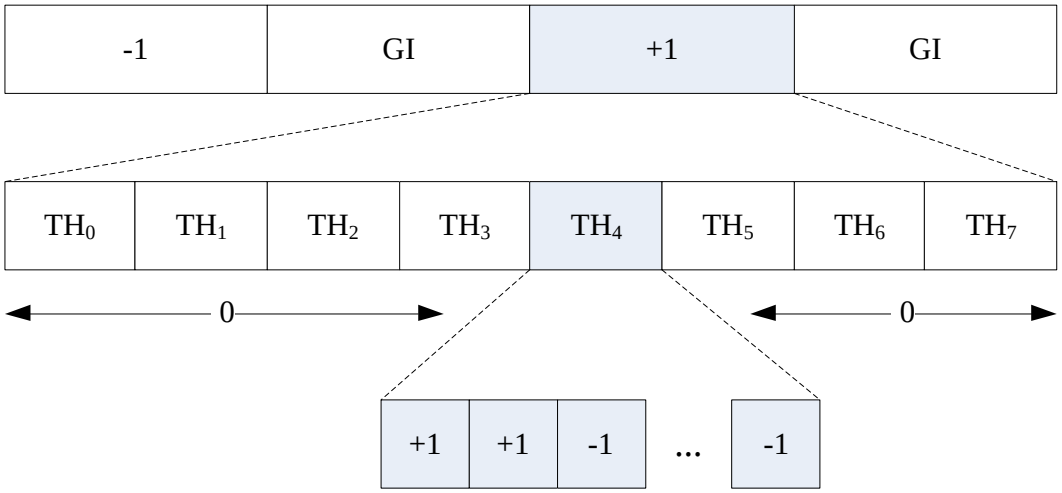


Abbildung 2.17: IEEE 802.15.4a Symbol-Frame

Der *Burst* wird entweder in der ersten oder zweiten Hälfte des Symbol-Frames übertragen. Jede dieser Hälften ist wiederum zweigeteilt, wobei dort jeweils im ersten Viertel übertragen wird und das zweite als *GI*[41] freigehalten wird. Diese Viertel, in denen die Übertragungen stattfinden, sind wieder in mehrere Hopping-Positionen (Parameter N_{Hops}) unterteilt, um *Time Hopping* zu ermöglichen (vgl. Abschnitt 2.2.3.7 auf Seite 42). Ein beispielhafter Symbol-Frame für $N_{Hops} = 8$, mit einem *THS*-Wert von 4 und einem Daten-Bit mit dem Bit-Wert 1 ist in Abbildung 2.17 gezeigt.

Die optionale *BPSK*-Modulation verändert die Phasen der einzelnen Pulse innerhalb eines *Burst*. Diese Phasen werden durch ein sog. *Burst Scrambling* vorgegeben. Dabei werden die Phasen durch ein *LFSR*[42] vorgegeben (dasselbe, das auch die *THS* vorgibt). Abhängig vom Redundanz-Bit des systematischen Faltungs-Codes wird daraufhin die Phase jedes Pulses im *Burst* entweder gekippt oder beibehalten.

Auf diese Weise kann jedes Symbol ein Daten-Bit durch die *BPM*-Modulation befördern, während die *BPSK*-Modulation ein weiteres Bit zur Absicherung

[41] Guard Interval - *Sicherheitsabstand*
[42] Linear Feedback Shift Register - *lineares Schieberegister mit Rückkoppelung*

durch einen systematischen Faltungs-Code trägt. Ein kohärenter Empfänger, der beide Modulationen demodulieren kann, erhöht also nicht die Datenrate, sondern die Robustheit der Übertragung.

Timing

Im Standard wird zwischen einer *Peak PRR* und einer *Mean PRR* unterschieden. Die *Peak PRR* beträgt immer 499,2 MHz und ist die größte Frequenz, mit der ein Sender Pulse aussenden können muss. Diese wird nur innerhalb eines *Bursts* erreicht. Die mittlere *PRR* bezeichnet dagegen die Anzahl der Pulse innerhalb eines Symbol-Frames (= Pulse innerhalb eines *Burst*s) geteilt durch die Dauer dieses Symbol-Frames.

$$PRR_m = \frac{N_{CPB}}{T_{dsym}} \tag{2.13}$$

Die Dauer des Symbol-Frames wird durch Parameter der *PHY*-Schicht bestimmt, konkret durch die Anzahl der Hopping-Positionen N_{Hops} und die Anzahl der Pulse pro *Burst* N_{CPB}.

$$T_{dsym} = 4 \cdot N_{Hops} \cdot N_{CPB} \cdot T_{chip} \tag{2.14}$$

Während der Präambel werden alle Pulse auf der *Mean PRR* gleichverteilt gesendet. Erst im Datenteil des Paketes werden die Pulse zu *Bursts* gruppiert, die auf der *Peak PRR* gesendet werden, wobei die *Mean PRR* zwischen Präambel und Datenteil gleichbleibt.

Die *PHY*-Modi des Standards sehen drei mögliche *Mean PRRs* vor: 15,6 MHz und 3,90 MHz verpflichtend und optional 62,4 MHz. Um die verschiedenen *PHY*-Modi mit verschiedenen Datenraten (zw. 0,11 Mbit/s und 27,24 Mbit/s) zu erhalten, werden für jede *Mean PRR* die Chips per Burst (N_{CPB}) variiert, während die Anzahl der Hopping-Positionen (N_{Hops}) gleichbleibt. Die Datenrate errechnet sich aus der Umkehrung der Symboldauer.

$$Bitrate = \frac{1\,\text{bit}}{T_{dsym}} \tag{2.15}$$

2.2.3.10 Schätzung der Reichweite von IR-UWB

Zum Zeitpunkt der Erstellung der Dissertation gab es noch keinen verfügbaren Transceiver auf Basis von UWB-Impulsfunk. Da die Arbeit aber auf *IR-UWB* basiert, ist es wichtig, eine Abschätzung für die möglichen Reichweiten in Abhängigkeit der *PHY*-Parameter zu ermitteln.

Dabei wurden allgemeingültige Formeln für *IR-UWB* aufgestellt und auf den Spezialfall IEEE 802.15.4a angewandt. In [29] wurden, zeitgleich zu meiner Arbeit, detailliertere Berechnungen zu den standardkonformen Modi von IEEE 802.15.4a vorgestellt. Die hier vorgestellten Berechnungen dienen nur als allgemeinere, gröbere Abschätzung der möglichen Reichweite bei bekannter mittlerer *PRR*.

Um diese Abschätzung vornehmen zu können, wird die Reichweite eines einzelnen Pulses ermittelt. Dazu muss berechnet werden, welche Leistung ein Sender pro Puls abstrahlen darf, ohne die Regulierungsvorschriften zu verletzen.

Aus dem Rahmen der in Unterabschnitt 2.2.3.3 auf Seite 31 vorgestellten Regulierungsvorschriften ergibt sich ein spektrales Leistungsbudget pro Millisekunde und MHz,

$$\overline{PSD_{\mathrm{ms}}} = 10^{\frac{-41{,}3\,\mathrm{dBm}}{10\,\mathrm{dBm}}} \cdot 10^{-3} \frac{W}{\mathrm{MHz}} = 7{,}41 \cdot 10^{-8} \frac{W}{\mathrm{MHz}} \tag{2.16}$$

das von der Obergrenze für die mittlere Leistung pro ms und MHz Bandbreite der Regulierung abhängig ist. Für diese Grenze wurden $-41{,}3\,\mathrm{dBm}$ genommen, da dieser Wert sowohl der US-amerikanischen als auch der europäischen Regulierung entspricht. Nimmt man als Vereinfachung eine „perfekte“ Pulsform an, welche die Energie jedes Pulses gleichmäßig über die gesamte Bandbreite des Signals BW_s (in MHz) verteilt, dann kann das Leistungsbudget für eine Millisekunde durch

$$\overline{P_{\mathrm{ms}}} = 10^{\frac{-41{,}3\,\mathrm{dBm}}{10\,\mathrm{dBm}}} \cdot 10^{-3} \cdot \frac{BW_s}{\mathrm{MHz}} \tag{2.17}$$

berechnet werden. Um aus diesem Budget die Sendeleistung einzelner Pulse errechnen zu können, muss das Budget auf die Anzahl der Pulse aufgeteilt werden. Dafür ist relevant, wie viele Pulse pro Millisekunde

$$\overline{N_{max}} = \frac{1\,\mathrm{ms}}{T_C} \tag{2.18}$$

möglich sind, was von der Länge eines Pulses T_C abhängig ist. So kann die Leistung, die pro Puls eingesetzt werden kann, durch

$$P_{Pulse} = min(\overline{P_{\mathrm{ms}}}\frac{N_{max}}{N_{Pulses}}, 1\,\mathrm{mW}) \tag{2.19}$$

berechnet werden. Dabei ist P_{Pulse} durch die Regulierung auf maximal 1 mW beschränkt. Die Anzahl der versendeten Pulse N_{Pulses} pro Millisekunde hängt unter anderem von dem Kommunikationsschema des Systems ab, da jeder einzelne Teilnehmer das Leistungsbudget voll ausnutzen kann. Geht man vereinfachend von einem einzelnen Sender aus, der durchgehend sendet, so kann N_{Pulses} mit Hilfe der Pulswiederholungsrate PRR durch

$$N_{Pulses} = PRR \cdot 1\,\mathrm{ms} \tag{2.20}$$

berechnet werden.
Für ein IEEE 802.15.4a-System, mit $T_C = 2\,\mathrm{ns}$ langen Pulsen (unter der Annahme der oben erwähnten „perfekten“ Pulsform) mit $BW_s = 500\,\mathrm{MHz}$ Bandbreite und einer mittleren Pulswiederholungsrate $PRR = 15,6\,\mathrm{MHz}$, kann eine maximale Leistung pro Puls von

$$\begin{aligned}
P_{Pulse} &= 10^{\frac{-41,3\,\mathrm{dBm}}{10\,\mathrm{dBm}}} \cdot 10^{-3}\frac{W}{\mathrm{MHz}} \cdot 500\,\mathrm{MHz} \cdot \frac{N_{max}}{N_{Pulses}} \\
&= 74,110^{-9}\frac{\mathrm{W}}{\mathrm{MHz}} \cdot 500\,\mathrm{MHz} \cdot \frac{\frac{1ms}{T_C}}{PRR \cdot 1\,\mathrm{ms}} \\
&= 74,110^{-9}\frac{\mathrm{W}}{\mathrm{MHz}} \cdot 500\,\mathrm{MHz} \cdot \frac{1}{15,6 \cdot 10^3\,\mathrm{Hz} \cdot 2 \cdot 10^{-9}\,\mathrm{s}} \\
&= 74,110^{-9}\frac{\mathrm{W}}{\mathrm{MHz}} \cdot 500\,\mathrm{MHz} \cdot 32.05 \\
&= 1,187\,\mathrm{mW} \Longrightarrow 1\,\mathrm{mW}
\end{aligned} \tag{2.21}$$

errechnet werden. Da die Regulierung eine Obergrenze von 1 mW angibt, kann nicht mehr als 1 mW pro Puls verwendet werden. Es ist erkennbar, dass bei einer gegebenen Pulsform lediglich die PRR entscheidet, welche Leistung pro Puls eingesetzt werden kann, da diese Leistung sich indirekt proportional zur PRR verhält.

Dadurch ergibt sich eine Obergrenze PRR_{max}

$$\begin{aligned} P_{Pulse} &= 1\,\mathrm{mW} \\ 1\,\mathrm{mW} &= P_{\mathrm{ms}} \cdot \frac{1}{PRR \cdot T_C} \\ PRR &= 10^3\,\frac{1}{\mathrm{W}} \cdot P_{\mathrm{ms}} \cdot \frac{1}{T_C} \\ PRR_{max} &= 10^3 \cdot 10^{\frac{-41{,}3\,\mathrm{dBm}}{10\,\mathrm{dBm}}} \cdot 10^{-3}\frac{1}{\mathrm{MHz}} \cdot 500\,\mathrm{MHz} \cdot \frac{1}{2 \cdot 10^{-9}\ \mathrm{s}} \\ &= 18,53\,\mathrm{MHz} \end{aligned} \tag{2.22}$$

der PRR für die Nutzung der maximalen Abstrahlleistung von 1 mW. Jede geringere PRR kann die maximale Abstrahlleistung nutzen, ab dieser Rate muss jedoch dann die Leistung reduziert werden.

Um mit der abgestrahlten Leistung die maximale Reichweite ermitteln zu können, muss auch der Pfadverlust (auch Freiraumdämpfung genannt) berechnet werden. Die gängigste Formel hierfür ist die Pfadverlustformel nach de Friis.

$$P_r = P_t + G_t + G_r + 20log_{10}(\frac{\lambda}{4\pi R}) \tag{2.23}$$

Dabei ist P_r die empfangene Leistung in dB, P_t die gesendete Leistung in dBm, R ist die Entfernung und λ die Wellenlänge, wobei $R >> \lambda$ gelten muss. G_t und G_r sind die Antennengewinne bzw. -verluste in dB.

Da die Wellenlänge für Pulsfunk nicht definiert ist, kann die Formel nicht direkt angewendet werden. Es gibt verschiedene Ansätze einer Erweiterung der Formel für *IR-UWB*, z.B. [62, 24, 59]. Für eine grobe Abschätzung kann aber die Mittenfrequenz des Signals f_C verwendet werden. Diese Näherung trifft besonders in Fällen zu, in denen $BW_s << f_C$ gilt, also bei denen die Bandbreite des Signals erheblich kleiner ist als die Mittenfrequenz [25].

Wird eine $PRR < 18,53\,\mathrm{MHz}$, die eine maximal erlaubte Abstrahlleistung von $P_t = 1\,\mathrm{mW} = 0\,\mathrm{dBm}$ erlaubt, zugrunde gelegt, dann kann die empfangene Leis-

tung, unter der Annahme vollkommen isotroper Antennen sowohl am Sender als auch am Empfänger ($\Longrightarrow G_t = G_t = 1$), als

$$\begin{aligned} P_r &= 0\,\text{dBm} + 20log_{10}(\frac{c}{4\pi \cdot d \cdot f_C}) \\ &= 20log_{10}(\frac{c}{4\pi}) - 20log_{10}(f_C) - 20log_{10}(d)\,\text{dBm} \end{aligned} \tag{2.24}$$

berechnet werden.

Wird dies für die Mittenfrequenz des *Mandatory High Band*, also des verpflichtenden Hochfrequenzkanals[43] von IEEE 802.15.4a, $f_C = 7.9\,\text{GHz}$ berechnet, so erhält man ein Verhältnis von Empfangsenergie und Entfernung wie in Abbildung 2.18 dargestellt.

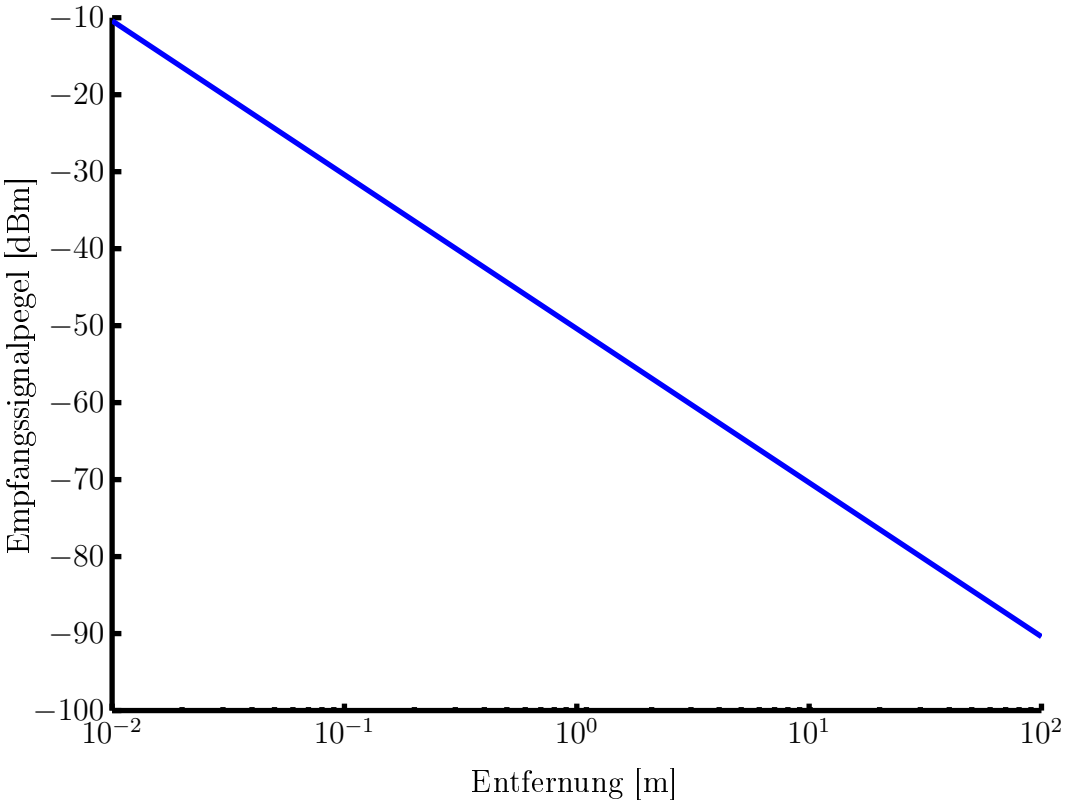


Abbildung 2.18: Empfangener Signalpegel gegen Entfernung für IEEE 802.15.4a

[43] Niedrigere Frequenzen sind in der europäischen Regulierung nur mit erheblich geringerer Energie nutzbar.

Dieser Signalpegel entspricht dem eines einzelnen Pulses. Wenn nicht jeder einzelne Puls beim Empfänger aufgelöst werden muss, lassen sich durch Methoden wie Integration mehrerer Pulse wesentlich höhere Signalpegel erzielen.

Als weiterer Wert für die Abschätzung der Reichweite muss das Rauschniveau beim Empfänger bekannt sein. Dieses Rauschen kann mehrere Ursachen haben:

1. Interferenzen bzw. Störungen durch industrielle Prozesse
2. andere Funksysteme auf derselben Frequenz
3. Effekte durch Multipfad-Ausbreitung
4. thermisches Grundrauschen

Diese vier Rauschquellen werden in den folgenden Abschnitten genauer untersucht.

Industrielle Prozesse

In [70] wurden die Ergebnisse von Messungen in verschiedenen Fertigungsprozessen vorgestellt. Dabei wurde beobachtet, welcher Frequenzbereich durch welche industriellen Prozesse gestört wird.

Abbildung 2.19 auf der nächsten Seite zeigt die direkten Störfrequenzen (orange) und deren harmonische Oberwellen (weiß) als Ergebnisse dieser Messungen. Dabei ist erkennbar, dass typische industrielle Fertigungsprozesse keine harmonischen Frequenzen über 1,5 GHz erzeugen. Geräte zur Mikrowellentrocknung strahlen eventuell noch höhere Frequenzen im GHz-Bereich aus, jedoch selten bis in die Bereiche, die den Hochfrequenzkanal von IEEE 802.15.4a betreffen.

Die Störung durch Ausstrahlung des Fertigungsprozesses selbst oder harmonischer Frequenzen davon können also vernachlässigt werden.

Koexistenz mit anderen Funksystemen

Das genutzte Frequenzband von 7,65 GHz bis 8,15 GHz ist außer für UWB-Systeme nur für die militärische Nutzung und Wetter- und Kommunikationssatelliten reserviert [18]. In einer Fertigungshalle ist dadurch keine Interferenz durch andere Funksysteme zu erwarten.

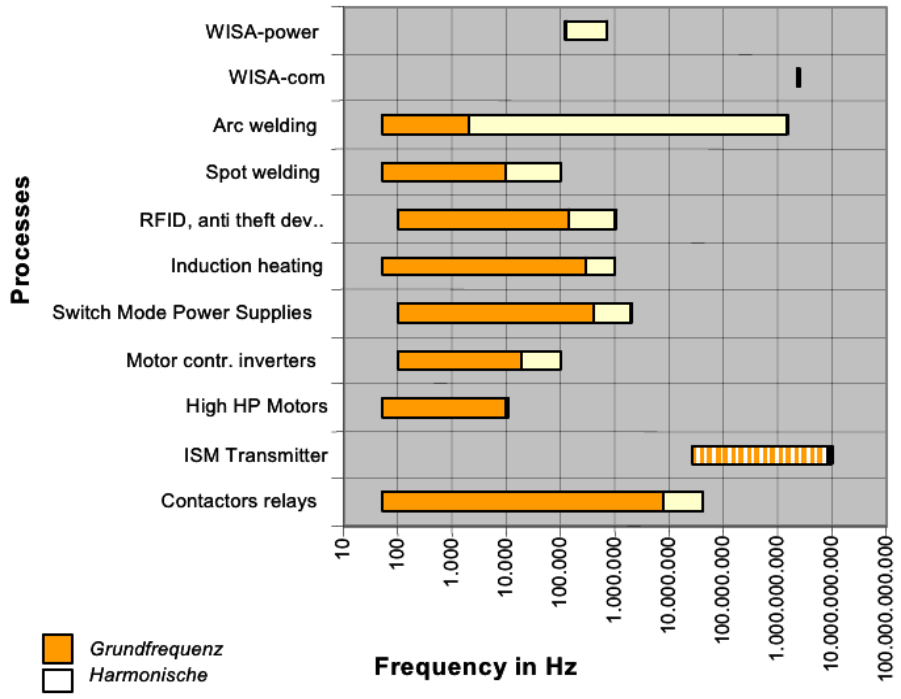


Abbildung 2.19: Störfrequenzen verschiedener industrieller Prozesse [70]

Multipfadeffekte

Abhängig von der vorherrschenden Multipfad-Situation verteilt sich jeder Puls über einen längeren Zeitraum, genannt Delay-Spread. Dieser wirkt sich einerseits in der Verringerung des Energieniveaus innerhalb von T_C, andererseits in einem höheren Energieniveau in leeren Zeiträumen aus. Diese Effekte führen zu einer nichtlinearen Verschlechterung des *SNR*, wirken sich jedoch bei *IR-UWB* geringer aus als für andere Funksysteme, da der Empfänger ein längeres Zeitintervall als T_C verwenden kann, um Pulse mitsamt der Multipfad-Echos zu empfangen.

Für eine Abschätzung der Obergrenze der Reichweite werden diese Effekte deshalb nicht berücksichtigt.

Thermisches Rauschen

Dieses Grundrauschen wird durch Nyquist-Johnson-Rauschen angenähert.

Die logarithmische Darstellung des Rauschens

$$P_{noise} = 10log_{10}(k_B T \cdot 1000) + 10log_{10}(BW_s) \tag{2.25}$$

ist von der Bandbreite des Signals BW_s abhängig, außerdem von der Temperatur in Kelvin T und der Bolzmann-Konstante $k_B = 1,38 \cdot 10^{-23}\,\mathrm{J/K}$. Wenn als Temperatur $30°C = 303K$ angenommen wird, so ergibt sich ein Rauschniveau durch Nyquist-Johnson-Rauschen von

$$P_{noise} = 10log_{10}(k_B \cdot 303 \cdot 1000) + 10log_{10}(BW_s) = -87\,\mathrm{dBm} \tag{2.26}$$

Mit diesem Rauschpegel und dem Signalniveau lässt sich die *SNR* durch

$$SNR(d) = P_r(d) - P_{noise} = P_r(d) + 87\,\mathrm{dB} \tag{2.27}$$

errechnen. Der Verlauf dieser *SNR* wird in Abbildung 2.20 gezeigt.

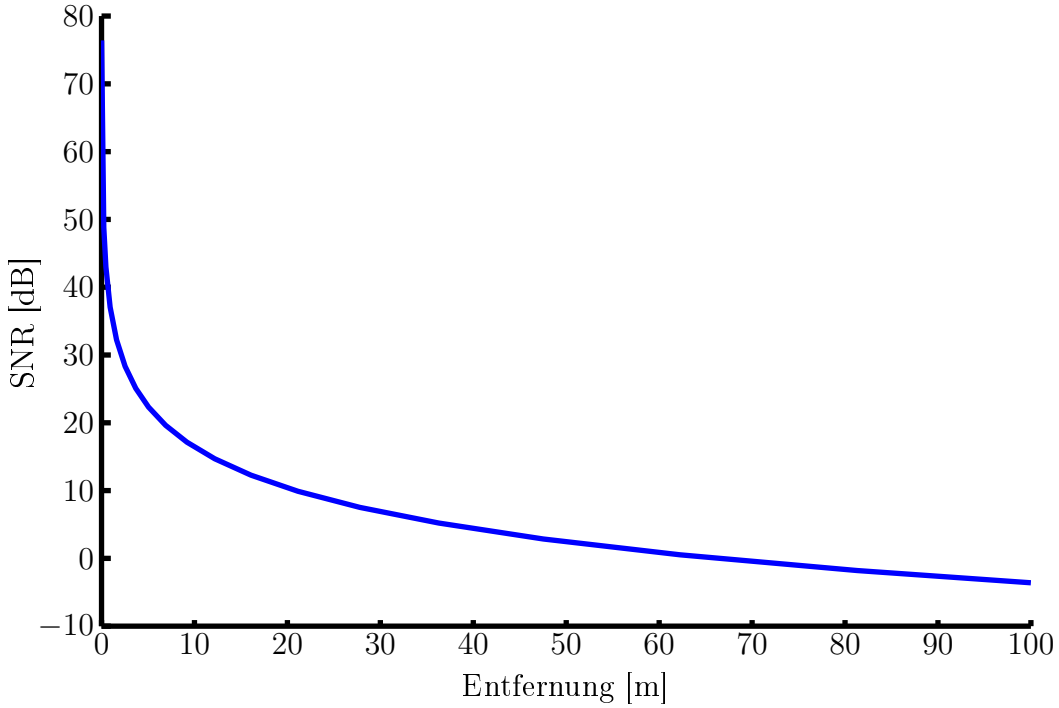


Abbildung 2.20: Verlauf der SNR über die Entfernung für IEEE 802.15.4a

Abhängig vom Empfänger gibt es eine Untergrenze für die *SNR*, genannt SNR_{min}, sodass Pakete gerade noch im Rahmen der geforderten Dienstgüte empfangen werden können, wodurch sich die maximale Reichweite d_{max} durch

$$\begin{aligned}
\mathrm{SNR}_{min} = SNR(d_{max}) &= P_r(d_{max}) - P_{noise} \\
P_r(d_{max}) &= \mathrm{SNR}_{min} + P_{noise} \\
&= 20log_{10}(\frac{c}{4\pi}) - 20log_{10}(f_C) - 20log_{10}(d_{max}) \\
20log_{10}(d_{max}) &= 20log_{10}(\frac{c}{4\pi}) - \mathrm{SNR}_{min} - P_{noise} - 20log_{10}(f_C) \\
log_{10}(d_{max}) &= log_{10}(\frac{c}{4\pi}) - \frac{1}{20}\mathrm{SNR}_{min} - \frac{1}{20}P_{noise} - log_{10}(f_C) \\
d_{max} &= \frac{c}{4\pi \cdot f_C} \cdot 10^{-\frac{1}{20}(\mathrm{SNR}_{min}+P_{noise})} \qquad (2.28)
\end{aligned}$$

berechnen lässt.

Damit lässt sich ermitteln, dass ein Empfänger, der mindestens ein *SNR* von 10 dB benötigt, eine Reichweite von ca. 27 m erreicht. Es ist auch erkennbar, dass bereits ein Prozessgewinn von wenigen dB die Reichweite erheblich erhöht.

Während die relativ geringe Sendereichweite des High-Bands von IEEE 802.15.4a zuerst wie ein Nachteil erscheint, im Sinne einer geringeren möglichen Größe einer Netzwerkzelle, so erlaubt eine geringe Ausbreitungsentfernung wiederum eine bessere räumliche Wiederverwendung desselben Kanals, damit mehr Zellen und dadurch eine höhere Anzahl von Knoten, was für die Anwendung in der Fertigungsautomatisierung von Vorteil ist.

2.3 Stand der Wissenschaft

Als Ergänzung zum Stand der Technik wird eine Auswahl an aktiven Forschungsthemen der Automatisierung und des *UWB*-Funks vorgestellt, die einen Bezug zu dieser Arbeit haben.

2.3.1 Industrielle drahtlose Automatisierung

Die drahtlose Automatisierung in der Fertigungsautomatisierung hat im letzten Jahrzehnt immer tiefere Ebenen erschlossen. Dabei werden in der wissen-

schaftlichen Literatur meist nur Bluetooth, IEEE 802.15.4 und WLAN als potentielle *PHY*-Schichten betrachtet. Hancke und Allen stellen in [33] einen Überblick über die wichtigsten Eigenschaften von *UWB* für die drahtlose Automatisierung zusammen, in dem *UWB* als vielversprechende Technologie für drahtlose Automatisierung auf mehreren Ebenen bezeichnet wird. Allerdings wird kein konkreter Vorschlag der Benutzung gemacht. Diese Dissertation ergänzt diese Betrachtungen, indem die konkrete Anwendung der *IR-UWB*-Technologie für die tiefen Ebenen der Automatisierung untersucht wird.

Zeitgleich mit meiner Dissertation untersuchten Buda und Wullert die Leistung von *MB-OFDM*-basiertem *UWB*-Funk im industriellen Einsatz. In [16] stellen sie dazu eine Messkampagne mit einer Single-Link-Lösung in einem Hochregallager vor. Im Gegensatz zu meiner Arbeit wird nur ein einzelnes Sender-Empfängerpaar betrachtet und kein Impulsfunk, sondern das in Abschnitt 2.2.3.4 beschriebene Protokoll ECMA-368 verwendet.

Der wichtigste und aktivste Forschungsbereich für drahtlose Kommunikation in der industriellen Automatisierung sind Fehlerkontrollschemata zur Reduktion von Bit-Fehlern und dem Verlust von ganzen Paketen durch fehlende Synchronisation. In dieser Arbeit werden diese Mechanismen erweitert, um die benötigte Zuverlässigkeit zu erhalten.

Die vorhandenen Transceiver ermöglichen oft nur eine Adaption der Protokolle ab der Applikationsschicht [20], wobei die Kanalbedingungen eigentlich Änderungen auf der *MAC*- und *PHY*-Ebene verlangen [15]. Deshalb wird in dieser Arbeit das Design der tieferen Netzwerkschichten eines Transceivers betrachtet.

Durch Adaptionen in der *MAC*-Schicht von IEEE 802.11 wird in [78] ein drahtloses Protokoll für die Feldbusebene vorgestellt. Dabei wird das *Polling*-basierte PROFIBUS-Protokoll durch ein angepasstes *Round-Robin*-Verfahren abgebildet. Ein ähnlicher Ansatz wurde von mir verfolgt, da drahtgebundene Polling-Protokolle durch ein Zeitschlitzverfahren ersetzt wurden.

Klassische Methoden für Optimierung auf den *MAC*- und *PHY*-Ebenen sind z.B. zusätzliche Redundanz über *FEC*-Codes, wie sie auch in dieser Arbeit eingesetzt werden. Diese werden aber auf einzelne Kanäle angewendet und bieten

in schmalbandigen Kanälen wenigen bis gar keinen Schutz gegen frequenzselektive *Deep-Fading*-Probleme [80].

Um diese Probleme zu umgehen, ist eine bekannte Möglichkeit das Forschungsfeld der *Diversity*-Techniken. Diese ermöglichen, allgemein gesagt, die gleichzeitige Übertragung derselben Information über mehrere Kanäle. Dadurch kann die Kanalkapazität gesteigert werden. Je unabhängiger die Fehlerquellen einzelner Kanäle voneinander sind, desto mehr steigt die Wahrscheinlichkeit eines fehlerfreien Kanals an. Daraus resultiert die minimale theoretische Fehlerrate $P_r = P_e^n$ für n komplett unabhängige Kanäle mit der Fehlerrate P_e [15]. Diese Reduktion der Fehlerrate wird als *Diversity Gain* bezeichnet. Anstelle der *Diversity* wird in dieser Arbeit untersucht, ob die erhöhte Kanalkapazität von Ultrawideband-Systemen die Probleme auch beheben können.

Das Forschungsgebiet des *Network Coding* [4] verspricht eine bessere Ausnutzung der Kanalkapazität, durch paketübergreifende algebraische Operationen [44]. Diese zusätzliche Kanalkapazität kann einerseits (wie z.B. in [41, 42]) zur Erhöhung des Datendurchsatzes genutzt werden und andererseits zur Erhöhung der Robustheit der Übertragung. Ein großer Vorteil des Network-Coding-Ansatzes ist, dass sich die Paket-übergreifenden Maßnahmen auch auf Pakete auswirken, die aufgrund von Fehlern in der Präambel oder dem Header verloren gehen, wobei üblicherweise keine Kodierung hilft.

Zu diesem Thema veröffentlichte ich ein adaptives Network-Coding-Schema zur Erhöhung der Robustheit von drahtlosen Übertragungen auf dem WPNC 2010 [HHA+10]. Allerdings sind diese Schemata nur bedingt in der Automatisierung anwendbar, da diese das Empfangen einer Serie von Paketen begünstigen (wie z.B. bei Dateiübertragungen). In der Automatisierung ist jedoch üblicherweise nur das letzte Paket bzw. der aktuellste Wert von Bedeutung. Ich stelle in dieser Arbeit allerdings eine an die Belange der Automatisierung angepasste Variante vor.

Weitere relevante Forschungsbereiche für die drahtlose Automatisierung sind [79]:

- Netzplanung, Installation und Betrieb:
  Hierfür müssen Planungsmethoden und Konfigurationsmöglichkeiten be-

reitgestellt werden, um das Netz bezüglich der Zykluszeit, Teilnehmerzahl, Frequenznutzung und geforderter Dienstgüte planen und anpassen zu können, wie z.B. in [19] vorgestellt. Idealerweise kann ein Netz sich autonom (re-)konfigurieren, was ein Forschungsthema aus dem Gebiet der Ad-hoc-Netze ist [77]. Auch die in dieser Arbeit verwendete *MAC*-Schicht bietet die Möglichkeit dynamischer Rekonfiguration.

- Das Design von deterministischen *MAC*-Protokollen:
  Protokolle, die harte Echtzeitkriterien erfüllen müssen, nutzen üblicherweise eine Priorisierung auf Paketbasis, wie z.B. in CAN-Bussen oder dem HIPERLAN-I-Standard. Die Siemens AG patentierte eine meiner Erfindungen in diesem Forschungsgebiet, eine Anwendung des Network Coding, das eine prioritätsabhängige Absicherung durch ineinander kodierte Paketwiederholungen erlaubt [P1]. Anstelle von Priorisierung kann auch ein festes Zeitschlitzverfahren zu deterministischem Verhalten führen. In dieser Arbeit wurde ein deterministisches *MAC*-Protokoll verwendet, das ein Zeitschlitzverfahren statt einer Priorisierung vornimmt, eine prioritätsabhängigkeit wäre jedoch eine zukünftige Erweiterung.

- Skalierbarkeit:
  In einer Fabrikumgebung kann sowohl die Zahl verschiedener Netzwerke als auch die Zahl einzelner Knoten pro Netzwerk sehr hoch werden [79], besonders wenn vorsorglich redundante Knoten eingebaut sind. Deshalb sind Protokolle nötig, die eine große Anzahl von Knoten und/oder Subnetzen verwalten können, ohne an Performance zu verlieren. Im hier vorgestellten System profitiert die Skalierbarkeit vor allem von der kurzen Reichweite der *UWB*-Technologie, die eine räumliche Wiederverwendung derselben Frequenz in mehreren unabhängigen Zellen ermöglicht.

2.3.2 Ultra-Wideband-Forschung

Kernthemen der *UWB*-Forschung sind zum Zeitpunkt des Verfassens der Arbeit *UWB*-Antennen und *Ultra-Low-Power*-Kommunikation für Sensornetze, die aber keine Relevanz für dieser Arbeit haben.

Gebiete der *UWB*-Forschung mit Bezug zu den Themen dieser Arbeit sind:

2.3.2.1 Lokalisierung

Ein Gebiet in der *UWB*-Forschung sind Radar- und Lokalisierungslösungen. Da sich bei den Pulsen von *IR-UWB*, eine entsprechend hohe Sample-Rate vorausgesetzt, alle Multipfad-Fragmente auflösen lassen und sich deshalb die Ankunftszeit und dadurch indirekt die Flugzeit eines Signals sehr genau bestimmen lassen, ist *UWB* sehr gut geeignet für Radar-, Tracking- und Lokalisierungslösungen.

In der Forschungsgemeinde wird zwischen der aktiven und passiven Lokalisierung unterschieden [84], wobei beim aktiven Ansatz das zu lokalisierende Objekt selbst Signale aussendet, während es beim passiven Ansatz diese lediglich reflektiert.

Zum passiven Ansatz gehören Impulsantwort-basierte Systeme wie medizinische Radarlösungen [69], zum Aufspüren von Menschen in komplexen Umgebungen [83] sowie Radar zur Durchdringung von Wänden [82] und zur Untersuchung der Bodenbeschaffenheit [32]. Auch Anwendungen zur Unfallvermeidung im Straßenverkehr und militärische Anwendungen im Bereich unbemannter Flugzeuge gehören zum Forschungsgebiet des passiven *UWB*-Radars [27].

Der aktive Ansatz ist als Alternative zu *RFID*[44]-Systemen zu verstehen. Im Standard IEEE 802.15.4a [3] ist eine optionale Zwei-Wege-Lokalisierung enthalten. Dabei muss auf ein als „Ranging-Paket“ markiertes Paket innerhalb einer definierten Zeit mit einer speziell aufgebauten Antwort reagiert werden, die einen Zeitstempel des Empfangs enthält.

Da hierbei zwei Pakete für eine Zeitmessung nötig sind und mindestens drei Zeitmessungen (innerhalb kurzer Zeit) für eine Positionsbestimmung, gibt es erhebliches Potential für *MAC*-Schicht- und Cross-Layer-Verbesserungen. Ich untersuchte in [HKSH09] mehrere vorgeschlagene Verbesserungen der *MAC*-Schicht bzgl. Verbesserungen für Positionsbestimmungen und Reduktion der Latenzzeit.

[44] Radio Frequency ID - *aktive Tags zur drahtlosen Identifikation*

Durch die erforderliche genaue Synchronisierung und die kurzen Latenzzeiten ließe sich auch das hier vorgestellte Automatisierungssystem für die aktive Lokalisierung verwenden.

2.3.2.2 Body Area Networks

Die Erforschung von körpernaher drahtloser Kommunikation, bzw. drahtloser Kommunikation innerhalb des Körpers, auf Basis von *IR-UWB* bildet den Kern des kommenden Standards IEEE 802.15.6 für drahtlose Netzwerke in und am (menschlichen) Körper, auch *BAN*[45] genannt. Diese Netzwerke sollen Anwendungen in der Medizin sowie im Sport- und Entertainment-Bereich ermöglichen. Es existieren Forschungsprojekte für *UWB*-Transceiver für *BAN* [30], Antennen [31] und Kanalmodelle [74] sowie für spezialisierte Routing-Protokolle [13].

Im Bereich der *BAN* gibt es auch zeitkritische Anwendungen, zum Teil mit ähnlichen Anforderungen wie in der Automatisierung. Dadurch ist ein Einsatz des hier vorgestellten Systems auch in diesem Bereich denkbar.

2.3.2.3 Time Reversal

Ein großer Teil dieser Arbeit befasst sich mit der Verbesserung der Zuverlässigkeit von *IR-UWB*-Übertragungen.

Ein Forschungsgebiet, welches als vielversprechend für die Verlässlichkeit von *IR-UWB* gilt, ist das *Time-Reversal*-Prinzip. Es ermöglicht eine stärkere Robustheit gegen Multipfad-Effekte und zusätzlich eine höhere Informationssicherheit [71]. Allerdings wird eine sehr hohe Sample-Rate benötigt, um alle Multipfad-Fragmente auflösen zu können.

Eine theoretische Ausprägung des Prinzips geht als vereinfachende Annahme von einem symmetrischen Kanal zwischen zwei Knoten aus, d.h. dass die Kanalimpulsantwort in beiden Richtungen ungefähr gleich aussieht. Dadurch könnte die empfangene Impulsantwort zeitlich umgekehrt als Filter des Senders benutzt werden. So würden mehrere kleine Impulse mit verschiedenen

[45] Body Area Network - *(drahtloses) Netzwerk im und am Körper*

Verzögerungen ausgesendet. Da diese Verzögerungen ungefähr zu dem relativen Unterschied der Pfadlängen passen, gäbe es einen Zeitpunkt, an dem sich alle Pulse im Empfänger treffen [49].

Mehrere Variationen zur praktischen Umsetzung dieses theoretischen Schemas werden diskutiert, wie etwa bei der Benutzung von Antennen-Arrays zum Empfang der Impulsantwort [63] oder ein Digitalisieren und Versenden der Impulsantwort, wodurch kein symmetrischer Kanal notwendig ist und zusätzlich die Anwendung in (im Regelbetrieb) unidirektionalen Kanälen ermöglicht wird [5], so wie sie auch in dem hier vorgestellten System vorliegen.

2.4 Zusammenfassung

Zusammenfassend lässt sich sagen, dass auf dem Gebiet der Automatisierung Forschungsbestrebungen laufen, um fehlertolerante, effizientere Steuerungen mit geringeren Anforderungen an die Kommunikation zu erstellen, während auf dem Gebiet der drahtlosen industriellen Netzwerke vor allem die Forschung zur Reduktion von Fehlerrate und Latenz im Vordergrund steht.

UWB ist eine Familie neuer Technologien. Vor allem die *IR-UWB*-Technik bietet sich für viele Anwendungen an, die mit bisherigen Technologien nur schwer realisierbar sind. Eine davon ist die drahtlose Automatisierung in schwierigen Umgebungen, wie z.B. in der Fertigungsautomatisierung. Dabei ist für die technische Umsetzung oft die benötigte hohe Sample-Rate ein Problem. Nichtkohärente Empfänger mit niedrigeren Sample-Raten versprechen dagegen durch eine geringe Komplexität schnellere Realisierbarkeit, erfordern aber Anpassungen zur Erhöhung der Robustheit.

Kapitel 3

Definition eines parametrierbaren Modells zur Simulation einer UWB-IR-Übertragung gemäß IEEE 802.15.4a

Ich habe einen Simulator implementiert, der einen standardkonformen Sender, einen nichtkohärenten Empfänger sowie den dazwischenliegenden Funkkanal realisiert. Diese Simulation diente zur Evaluierung und Verifizierung der theoretischen Arbeiten. Dazu wurde der Simulator um die neu entwickelten Verfahren erweitert.

Damit war es möglich, die *PHY*-Schicht in verschiedenen Konfigurationen auf Schwachstellen zu untersuchen, sowie Lösungsansätze gegen diese Schwachstellen zu analysieren. Dafür muss der Simulator weitreichende Einstellmöglichkeiten zur Konfiguration der Komponenten bieten.

Dieses Kapitel ist wie folgt aufgebaut: Zuerst wird ein Überblick über den Übertragungsvorgang gegeben, dann wird ein Modell dieses Vorgangs in vektormathematischen Formeln dargestellt. Dabei werden auch die Parameter, mit denen die Simulation konfiguriert werden kann, vorgestellt. Für eine komplette Liste der nutzbaren Parameter und Details der Implementierung sei auf Anhang **??** verwiesen.

3.1 Aufbau eines Transceivers für IEEE 802.15.4a

In diesem Abschnitt wird das Design eines Senders und eines nichtkohärenten Empfängers vorgestellt. Das Design dieses Empfängers wurde auch als Referenz genommen, um die Optimierungen in der theoretischen Arbeit gegen die „herkömmliche Herangehensweise“ vergleichen und als Verbesserung verifizieren zu können.

3.1.1 Sender

3.1.1.1 Kodierung

Der Sender erhält einen Vektor von Bytes (die Nutzlast). Diese Bytes werden in 6-Bit-Symbole gewandelt und optional mit dem im Standard beschriebenen *Reed-Solomon-Code* versehen. Den daraus resultierenden Symbolen wird ein 13 Bit langer *PHY*-Header vorangestellt, der durch einen 6 Bit langen *SECDED*-Code gesichert ist.

Die kodierten Daten werden bitweise durch einen systematischen Faltungskodierer geleitet. Die ausgegebenen Daten-Bits und Prüfsummen werden getrennt gespeichert.

3.1.1.2 Präambel-Erzeugung

Dem Datenpaket wird eine Präambel vorangestellt. Diese besteht aus mehreren Wiederholungen des Präambel-Symbols, das aus dem ternären Präambel-Code erzeugt wird, der dem Netz zugeordnet ist. Die Präambel wird abgeschlossen durch den *SFD*, der aus einem achtstelligen Code, gespreizt mit dem Präambel-Symbol, besteht.

3.1.1.3 Modulation

Es folgen die modulierten Daten. Für die Modulation wird die Pseudo-Zufallsfolge der *THS* ermittelt, deren Initialwert aus dem Präambel-Code errechnet wird. Die Daten-Bits werden mit dieser *THS BPM*-moduliert, die Prüf-

summen des äußeren Faltungs-Codes hingegen *BPSK*-moduliert, wie in Abschnitt 2.2.3.9 auf Seite 44 beschrieben.

3.1.1.4 Analoges Front-End des Senders

Aus dem digitalen Basisbandsignal wird im analogen Sender-*Front-End* durch Pulsformung und Hochmischen auf die Mittenfrequenz das Transportsignal erzeugt.

3.1.2 Nichtkohärenter Energie-Detektions-Empfänger

3.1.2.1 Analoges Front-End des Empfängers

Im analogen Empfänger-*Front-End* wird das Signal verstärkt und heruntergemischt. Zur Verstärkung wird eine *AGC*[1]-Steuerung eingesetzt.

Beim nichtkohärenten Empfang wird das Energieniveau über ein *Integrationsintervall* integriert und aufgrund dieses Energieniveaus ermittelt, ob Pulse in diesem Zeitraum vorlagen. Wenn innerhalb dieses Intervalls Pulse mit verschiedenem Vorzeichen eintreffen, löschen sich diese aus. Deshalb wird das Signal vor dem Integrator gleichgerichtet oder mit einer Gilbertzelle quadriert. Dadurch ist das Signal am Eingang des Integrators immer positiv.

Das Ausgangssignal des Integrators wird durch einen *ADC*[2] in einen Digitalwert gewandelt.

3.1.2.2 Präambel-Detektion

Um die Präambel zu erkennen, wird eine Korrelation zwischen dem Präambel-Code und dem empfangenen Signal vorgenommen. Empfangene Präambel-Symbole sind durch Maxima des resultierenden Korrelationswertes erkennbar.

[1] Automatic Gain Control - *automatische Regelung des Empfangsverstärkers*
[2] Analog to Digital Converter - *Digitalisierer*

Wenn in diesen Präambel-Symbolen der *SFD*-Code gefunden wird[3], ist das Ende der Präambel erreicht.

3.1.2.3 Daten-Demodulation

Zu Beginn der Daten-Demodulation wird die Zufallsfolge der *THS* zurückgesetzt. Dann wird für jedes Daten-Symbol der entsprechende Wert der *THS* ermittelt. Das Energieniveau der beiden der *THS* entsprechenden Hopping-Positionen wird verglichen. Daraus resultiert dann der Bit-Wert für das jeweilige Symbol.

3.1.2.4 Dekodierung

Die ersten 19 ermittelten Bit-Werte bilden den *PHY*-Header. Dieser kann durch den *SECDED*-Code verifiziert und gegebenenfalls repariert werden. Der Rest der Daten wird zu 6-Bit-Symbolen angeordnet und durch einen Reed-Solomon-Decoder dekodiert. Daraus resultieren die Nutzdaten, die wieder zu Bytes angeordnet werden.

3.2 Simulationsmodell der Übertragung

Der oben skizzierte Übertragungsvorgang wurde in einem numerischen Simulationsverfahren dargestellt. Dieses Simulationsverfahren kann über Parameter konfiguriert werden, dadurch war es möglich den Einfluss verschiedener Parameter isoliert zu betrachten und Optimierungen vorzunehmen.

In den Formeln, die in der mathematischen Darstellung aufgeführt werden, wird zur besseren Lesbarkeit jedem Vektor $\overrightarrow{V}$ eine Akzessor-Funktion $\overrightarrow{V}(n) = V_n$ zugeordnet. Weiterhin bezeichnet $|\overrightarrow{V}|$ die Anzahl der Elemente von $\overrightarrow{V}$, während $|\overrightarrow{V}(n)|$ bzw. $|V_n|$ den Absolutbetrag des n-ten Elementes bezeichnet. Der Operator $\sqcup$ bezeichnet die Konkatenations-Operation $\{a, b\} \sqcup \{c, d\} = \{a, b, c, d\}$.

[3]Die Präambel besteht aus mehreren Wiederholungen des Präambel-Symbols und dem *SFD*, der aus acht kodierten Präambel-Symbolen besteht.

3.2.1 Sender

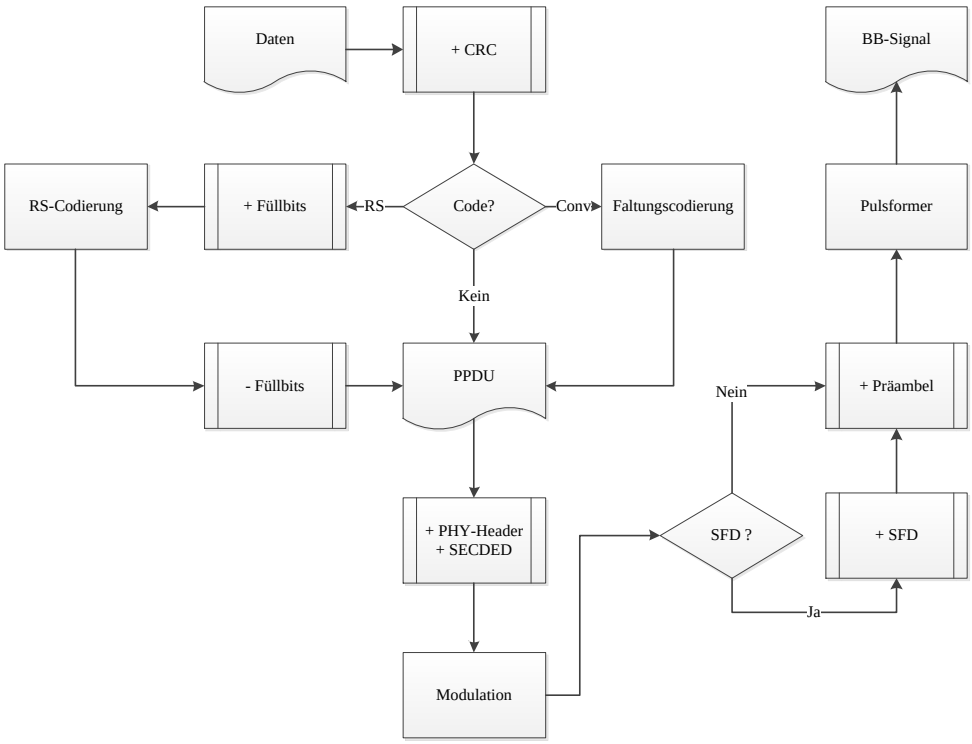


Abbildung 3.1: Schematische Darstellung des Senders

Abbildung 3.1 zeigt eine schematische Darstellung des Senderteils des Simulators.

3.2.1.1 Enkodierung

Als Nutzlast wird ein Vektor von Bits $\overrightarrow{NL}$ festgelegt. Über diesen Vektor wird eine 16 Bit lange CRC[4]-Prüfsumme berechnet und an den Vektor angehängt.

$$\overrightarrow{CRC_{Tx}} = \text{crc16}(\overrightarrow{NL}) \tag{3.1}$$

$$\overrightarrow{NL} = \overrightarrow{NL} \sqcup \overrightarrow{CRC_{Tx}} \tag{3.2}$$

[4]Cyclic Redundancy Check - *Prüfsumme zur Verfikation von Daten*

Die Daten können optional mit dem im Standard beschriebenen *Reed-Solomon-Code* oder einem Faltungs-Code kodiert werden.

Dem Anfang des Datenvektors wird ein *PHY*-Header vorangestellt. Dieser 13 Bit lange *PHY*-Header wird über einen 6 Bit langen *SECDED*-Code $\overrightarrow{C}$ geschützt, der sich durch

$$\begin{aligned}
\overrightarrow{C}_0 &= Bit_1 \otimes Bit_2 \otimes Bit_4 \otimes Bit_5 \otimes Bit_7 \otimes Bit_9 \otimes Bit_{11} \otimes Bit_{12} && (3.3)\\
\overrightarrow{C}_1 &= Bit_1 \otimes Bit_3 \otimes Bit_4 \otimes Bit_6 \otimes Bit_7 \otimes Bit_{10} \otimes Bit_{11} \otimes Bit_{13} && (3.4)\\
\overrightarrow{C}_2 &= Bit_2 \otimes Bit_3 \otimes Bit_4 \otimes Bit_8 \otimes Bit_9 \otimes Bit_{10} \otimes Bit_{11} && (3.5)\\
\overrightarrow{C}_3 &= Bit_5 \otimes Bit_6 \otimes Bit_7 \otimes Bit_8 \otimes Bit_9 \otimes Bit_{10} \otimes Bit_{11} && (3.6)\\
\overrightarrow{C}_4 &= Bit_{12} \otimes Bit_{13} && (3.7)\\
\overrightarrow{C}_5 &= Bit_1 \otimes Bit_2 \otimes Bit_3 \otimes Bit_4 \otimes \overrightarrow{C}_3 \otimes \overrightarrow{C}_4 && (3.8)
\end{aligned}$$

berechnet, wobei $\otimes$ die XOR-Operation[5] bezeichnet. Die restlichen Felder des *PHY*-Headers werden nicht belegt, da diese nicht benötigt werden.

Bei der Enkodierung kann über den Parameter dlen die Länge der Nutzlast und über den Parameter fec_code die Art des *FEC*-Codes angegeben werden.

3.2.1.2 Präambel-Erzeugung

Die Präambel wird aus einem der acht im Standard verfügbaren ternären Präambel-Codes erzeugt, die Auswahl des Präambel-Codes $\overrightarrow{C_P}$ erfolgt durch den Parameter pcode. Zwischen den einzelnen Symbolen des Präambel-Codes wird durch das Kroneker-Produkt $\odot$ jeweils die als L-Spreading bezeichnete Pause eingefügt, deren Länge mit dem Parameter L festgelegt wird. Aus dieser Operation entsteht das Präambel-Symbol $\overrightarrow{Psym}$.

$$\begin{aligned}
\overrightarrow{Psym} &= \overrightarrow{C_P} \odot L \\
&= \bigsqcup_{n=1}^{31} \overrightarrow{C_P}(n) \sqcup \{0, ..., 0\}_L
\end{aligned} \tag{3.9}$$

[5] Bitweises exklusives ODER

Dieses Präambel-Symbol wird mehrmals wiederholt, wobei die Anzahl der Wiederholungen N_{Prep} durch den Parameter Nprep angegeben wird.

$$\overrightarrow{Prea} = \bigsqcup_{n=1}^{N_{Prep}} \overrightarrow{Psym} \tag{3.10}$$

Im Anschluss an die Wiederholungen des Präambel-Symbols kann optional der *SFD* angehängt werden, angegeben durch den Parameter sendSFD. Der *SFD* besteht aus einem achtstelligen Code $\overrightarrow{C_{SFD}}$, der durch elementweise Vektormultiplikation mit dem Präambel-Symbol $\overrightarrow{Psym}$ gespreizt wird.

$$SFD_i = \overrightarrow{C_{SFD}}(i) \cdot \overrightarrow{Psym} \tag{3.11}$$

$$\overrightarrow{Prea} = \overrightarrow{Prea} \sqcup \overrightarrow{SFD} \tag{3.12}$$

3.2.1.3 Modulation

Aus dem Präambel-Code wird ein binärer Initialwert *Seed* von 16 Bit Länge für eine Pseudo-Zufallsfolge erzeugt, indem die Nullsymbole entfernt und alle Symbole des Wertes -1 durch 0 ersetzt werden.

$$\begin{aligned} Seed_i &= \frac{Psym_i + 1}{2} \; \forall \; Psym_i \in \overrightarrow{Psym} \neq 0 \\ &= \begin{cases} 1 & \text{wenn } Psym_i = 1 \\ 0 & \text{wenn } Psym_i = -1 \\ \emptyset & \text{wenn } Psym_i = 0 \end{cases} \end{aligned} \tag{3.13}$$

Mit diesem *Seed* wird die Zufallsfolge $\overrightarrow{PN}$ für das *Time Hopping* durch ein 16-Bit-*LFSR* ermittelt.

$$PN_1 = \text{LFSR}(\overrightarrow{Seed}) \tag{3.14}$$

$$PN_{i+1} = \text{LFSR}(PN_i) \tag{3.15}$$

Die 16-Bit-Werte von $\overrightarrow{PN}$ werden daraufhin auf Werte zwischen 0 und der Anzahl der Hopping-Positionen N_{Hops} gebracht, die mit dem Parameter Nhops angegeben werden. $\overrightarrow{PN}_{i,n}$ bezeichnet hierbei das n-te Bit von $\overrightarrow{PN}_i$.

$$\overrightarrow{THS}_i = \sum_{n=0}^{log_2(N_{Hops})} \overrightarrow{PN}_{i,n} \cdot 2^n \tag{3.16}$$

Mit dieser *THS* werden die kodierten Daten $\overrightarrow{Data}$ zu *BPM/BPSK*-Symbolen moduliert. Dazu wird, wie in Abschnitt 2.2.3.9 auf Seite 44 beschrieben, in jedem Datensymbol *Dsym* an der relevanten Hopping-Position *HP* ein *Burst* eingefügt, bezeichnet durch $\overrightarrow{Burst}$. Daraus resultiert ein ternärer Vektor von Datensymbolen $\overrightarrow{Dsym}$, dessen Elemente die zeitliche Position und Phase einzelner Pulse angeben. Dabei ist über den Parameter Ncpb die Anzahl der Pulse pro *Burst* N_{CPB} konfigurierbar.

$$\overrightarrow{Burst} = (\bigsqcup^{N_{CPB}} \{1\} \otimes PN_i) \cdot 2 - 1 \tag{3.17}$$

$$\overrightarrow{HP}(i) = \bigsqcup_{j=1}^{N_{Hops} \cdot 4} \bigsqcup^{N_{CPB}} \{0\} \tag{3.18}$$

$$\overrightarrow{HP}(i)(THS_i + 2 \cdot N_{Hops} \cdot Data_i) = \overrightarrow{Burst} \tag{3.19}$$

$$Dsym_i = \overrightarrow{HP}(i) \tag{3.20}$$

$$\Rightarrow HP_{i,j} = \begin{cases} \overrightarrow{Burst} & \text{wenn } j = THS_i + 2 \cdot N_{Hops} \cdot Data_i \\ \bigsqcup^{N_{CPB}} \{0\} & \text{wenn } j \neq THS_i + 2 \cdot N_{Hops} \cdot Data_i \end{cases}$$

Der Ausdruck $(\bigsqcup^{N_{CPB}} \{1\} \otimes PN_i) \cdot 2 - 1$ bezeichnet dabei das sog. *Burst Scrambling*, wie in in Abschnitt 2.2.3.9 auf Seite 44 beschrieben.

$$\overrightarrow{Frame} = \overrightarrow{Prea} \sqcup \overrightarrow{Dsym} \tag{3.21}$$

Die Präambel wird vor dem Datenteil angehängt, und für dieses komplette Paket $\overrightarrow{Frame}$ wird die Pulsformung durchgeführt.

3.2.1.4 Pulsformung

Der Vektor $\overrightarrow{Frame}$ drückt die zeitliche Lage und die Phase einzelner Pulse im 499,2-MHz-Basisbandsignal aus. Deshalb wird dieser Vektor durch einen Filter geschickt, der die Elemente in Pulse wandelt. Diese Pulse werden in einer durch den Parameter FsBB angegebenen Sample-Frequenz dargestellt. Als Standardwert wird die Nyquist-Frequenz 1 GHz verwendet.

$$\overrightarrow{BB_{TX}} = \text{rcosflt}(\overrightarrow{Frame}) \tag{3.22}$$

Als Pulsform wurde die Referenz-Pulsform aus dem Standard [3] genommen, ein *Root-Raised-Cosine*-Puls mit einem *Rolloff-Faktor* von 0,6 und einem *Delay* von 3. Diese Pulsform ist in Abbildung 3.2 dargestellt, mit einer Auflösung von 100 GHz.

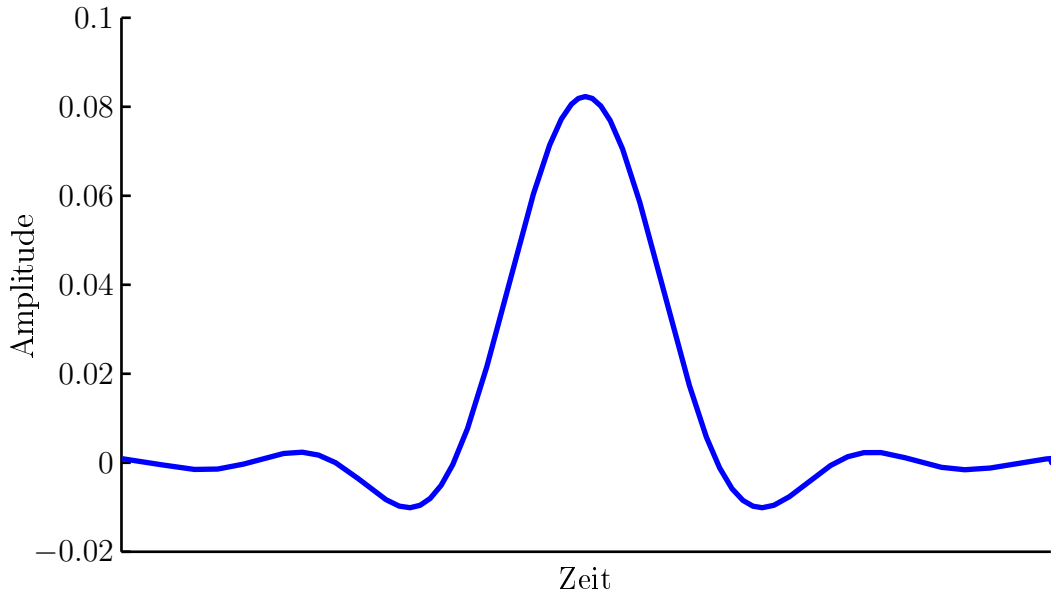


Abbildung 3.2: Referenzpulsform des Standards IEEE 802.15.4a

3.2.2 Hochfrequenz-Ebene

Dieser Teil des Simulators wandelt das Basisbandsignal in das Hochfrequenzsignal (HF-Signal). Das HF-Signal wird durch Kanaleffekte verändert und für

den Empfänger wieder heruntergemischt. Abbildung 3.3 zeigt schematisch den Aufbau des Hochfrequenz-Teils.

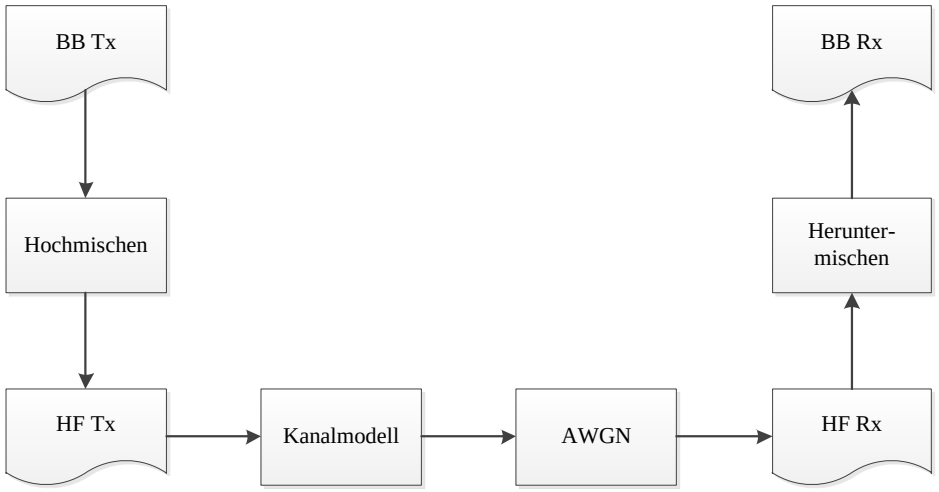


Abbildung 3.3: Schematische Darstellung des Hochfrequenz-Teils

3.2.2.1 Hochmischen

Zum Hochmischen des Basisbandsignals wird eine Amplituden-Modulation auf einer über den Parameter Fc einstellbaren Trägerfrequenz f_C vorgenommen. Der Default-Wert ist 7,9872 GHz, entsprechend dem mandatorischen *High-Band*-Kanal von IEEE 802.15.4a.

$$\overrightarrow{HF_{TX}} = \overrightarrow{BB_{TX}} \cdot cos(f_C \cdot t) \tag{3.23}$$

Dieses HF-Signal wird mit der durch den Parameter FsHF angegebenen Sample-Frequenz dargestellt, als Standardwert wird die Nyquist-Frequenz von 15,974 GHz verwendet. In Abbildung 3.4 auf der nächsten Seite ist das HF-Signal eines einzelnen Pulses gezeigt. Für diese Darstellung wurde das Signal zur besseren Erkennbarkeit allerdings mit einer Sampling-Rate von 150 GHz dargestellt.

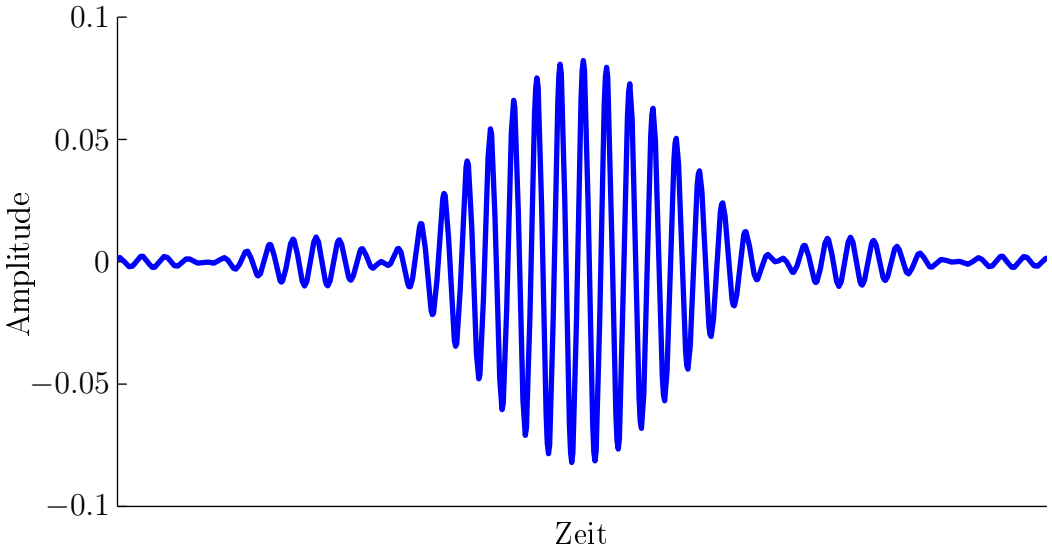


Abbildung 3.4: Darstellung des simulierten HF-Signals eines einzelnen Pulses

3.2.2.2 Kanal

Die Kanaleffekte werden durch drei Komponenten dargestellt: Als erste Kanaleinwirkung wird eine pseudo-zufällige Flugzeit vorangestellt, wobei die maximale Entfernung d_{max} über den Parameter tof_rng einstellbar ist.

$$TOF = \frac{d_{max} \cdot \sigma}{c} \tag{3.24}$$

$$\overrightarrow{HF_{TX}} = (\bigsqcup_{n=0}^{TOF} \{0\}) \sqcup \overrightarrow{HF_{TX}} \tag{3.25}$$

Bei diesen Formeln steht σ für eine gleichverteilte Zufallsvariable $0 \leq \sigma \leq 1$ und c für die Lichtgeschwindigkeit.

Eine weitere Kanalauswirkung wird durch eine Kanalimpulsantwort $\overrightarrow{CIR}$ dargestellt. Diese wird durch die Faltungsoperation $*$ auf das Signal angewendet.

$$\overrightarrow{HF_{Kanal}} = \overrightarrow{HF_{TX}} * \overrightarrow{CIR} \tag{3.26}$$

Die Kanalimpulsantwort bildet die Auswirkungen von Multipfad-Effekten nach. Sie wird mittels eines in der Industrie anerkannten Kanalmodells erzeugt, wel-

ches im Folgenden beschrieben wird. Der Parameter cm_num gibt an, welches der neun verschiedenen Kanalmodelle benutzt wird.

Multipfad-Kanalmodelle

Für die Kanalmodellierung wurde eine bestehende, auf den Kanalmodellen von Molisch et al. [53, 52] basierende, *MATLAB*-Implementierung von neun verschiedenen *IR-UWB*-Kanälen verwendet.

Die Molisch-Modelle sind erweiterte Saleh-Valenzuela-Modelle. Diese basieren auf der beobachteten Clusterung von Multipfad-Echos bei Übertragungen mit hoher Bandbreite. Die Erzeugung der Multipfad-Echos wird durch zwei Poisson-Prozesse dargestellt, davon steuert ein Prozess die Ankunftszeit der Cluster und der andere die Ankunftszeit von Echos innerhalb des Clusters. Jeder der Prozesse hat zwei Parameter: die *Arrival Rate* und den *Decay Factor* [66].

Molisch et al. erstellten mit diesen Parametern Kanalmodelle für neun Szenarien, jeweils basierend auf Messdaten. Für die *Decay Factors* wurden dabei aufgrund der besseren Übereinstimmung des frequenzselektiven Verhaltens Nakagami- oder lognormal-Verteilungen verwendet, anstatt der Rayleigh-Verteilungen des ursprünglichen Saleh-Valenzuela-Modells [51].

Im Folgenden werden exemplarisch einige relevante Kanalmodelle vorgestellt. Anschaulich wird dafür jeweils eine Impulsantwort (blau) eines *Burst*s aus 16 Pulsen (rot-gestrichelt) im Basisband gezeigt.

- Residential *LOS*[6]

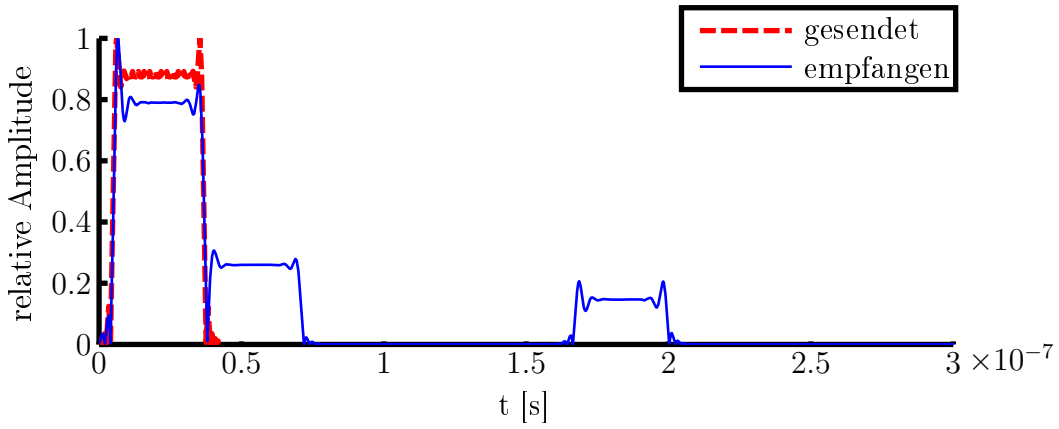


Abbildung 3.5: Kanalmodell 1: Wohnhaus LOS

Dieses Kanalmodell (Nummer 1) entstand aus Messungen mit direkter Sichtverbindung in einem Wohnhaus. Abbildung 3.5 zeigt die Kanalimpulsantwort eines *Bursts* von 16 Pulsen im Kanalmodell 1. Es ist erkennbar, dass neben dem ersten Pfad weitere Reflexionen, sog. „Multipfad-Echos“, eintreffen. Die Reflexionen in diesem Kanalmodell treffen in wenigen, kompakten Clustern ein, die allerdings bis zu 200 ns auseinander liegen können. Dadurch verursacht das Modell eher geringe Störungen, die aber trotzdem zu vereinzelten Bit-Fehlern führen können. Deshalb eignet sich das Modell für die Darstellung von guten, aber nicht idealen Bedingungen. In meiner Arbeit benötigte ich diese Kanäle für die Untersuchung der Auswirkungen von Umgebungen mit geringen Multipfad-Effekten.

[6]Line Of Sight - *Sichtverbindung*

- Residential *NLOS*[7]

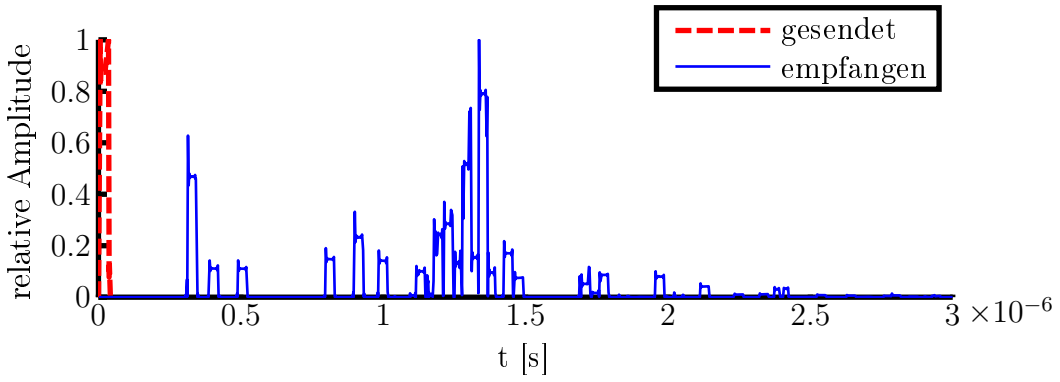


Abbildung 3.6: Kanalmodell 2: Wohnhaus NLOS

Dieses Kanalmodell (Nummer 2) entstand ebenfalls aus Messungen in einem Wohnhaus, allerdings bestand dabei keine Sichtverbindung. Da es keinen direkten Pfad gibt, treffen nur vielzählige, verschieden starke Multipfad-Echos ein, wie Abbildung 3.6 zeigt. Das Eintreffen mehrerer Cluster mit verschieden starken Echos erschwert vor allem die Präambel-Erkennung. Ich benutzte dieses Modell deshalb als schlechtesten Fall für die Präambel-Detektion.

[7]Non Line Of Sight - *ohne Sichtverbindung*

- Industrie *LOS*

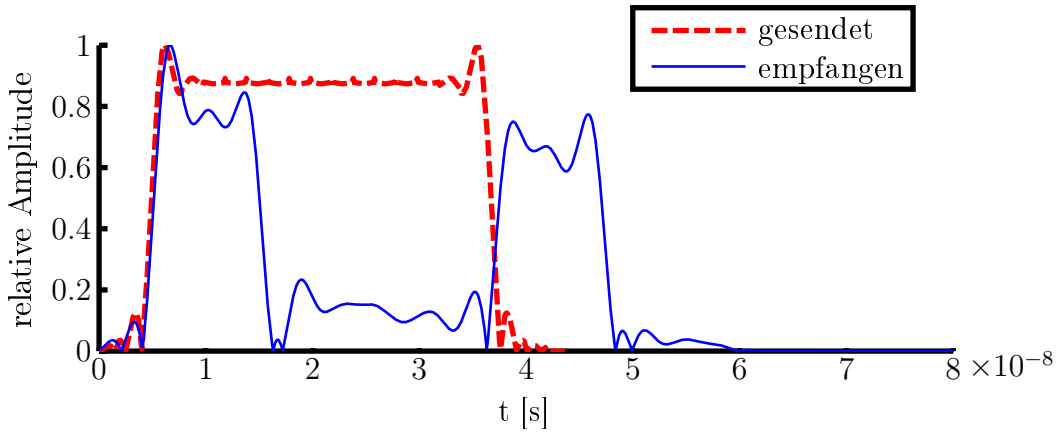


Abbildung 3.7: Kanalmodell 7: Industrie LOS

Dieses Kanalmodell (Nummer 7) entstand aus Messungen mit direkter Sichtverbindung in einer Fabrikhalle. Der direkte Pfad und die Multipfad-Echos liegen zeitlich recht nah zusammen, wie 3.7 zeigt. Da das Ziel der Arbeit ein System für die drahtlose Kommunikation im industriellen Umfeld ist, wurde für die meisten Untersuchungen dieses Kanalmodell verwendet. In einer dicht zugestellten Umgebung mit vielen metallischen Objekten sind durch den *LOS*-Kanal nur die Multipfad-Fragmente von nahe gelegenen Objekten bemerkbar. Multi-Pfade von weiter entfernten Objekten erreichen den Empfänger meist mit zu wenig Energie im Vergleich zum *LOS*-Pfad, um den Empfang zu stören.

- Industrie *NLOS*

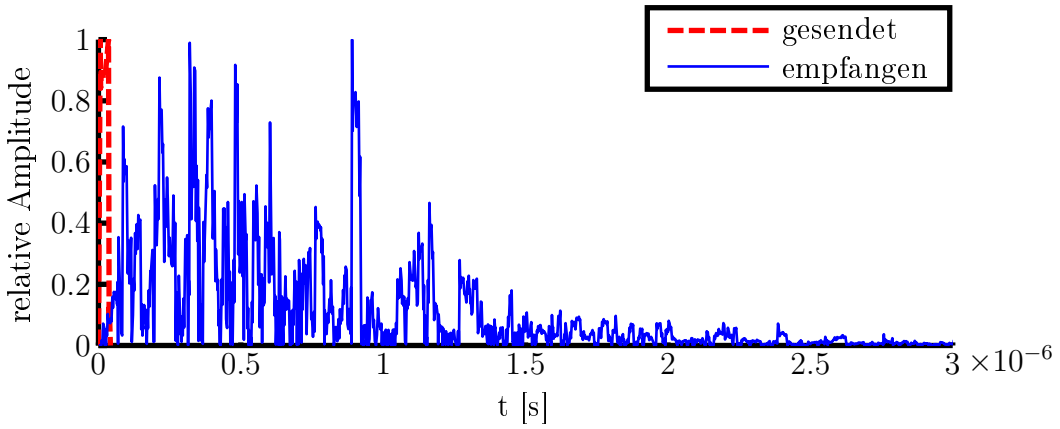


Abbildung 3.8: Kanalmodell 8: Industrie NLOS

Dieses Kanalmodell (Nummer 8) dient zur Simulation einer Übertragung ohne direkte Sichtverbindung in einer Fabrikhalle. Der Empfänger kann viele, zeitlich weit auseinanderliegende Reflexionen empfangen, wie Abbildung 3.8 zeigt. Dieses Kanalmodell stellt sehr schwierige Kanalbedingungen dar. Durch den langen Zeitraum, in dem die Multipfad-Echos auftreten und die hohe Anzahl einzelner Echos verteilt sich die Energie unregelmäßig über einen großen Zeitraum, sodass die ursprüngliche zeitliche Position des *Burst* nur schwer zu ermitteln ist.

- Weitere Kanalmodelle
  Es existieren noch weitere Kanalmodelle, die für diese Arbeit jedoch nicht relevant sind, z.B. Kanalmodelle für Büroumgebungen, Außenbereiche und Schneedecken.

Als letzte Kanalauswirkung wird dem Signal optional additives weißes Rauschen ($AWGN$[8]) beigefügt, mit einer über den Parameter noiseSNR einstellbaren SNR, die sich auf den $Peak$-Wert eines einzelnen Pulses bezieht.

$$Power = \max(\text{rcosflt}(1) \cdot cos(f_C \cdot t)) \tag{3.27}$$

$$\overrightarrow{HF_{Rx}} = \overrightarrow{HF_{Kanal}} + AWGN(Power, SNR) \tag{3.28}$$

3.2.2.3 Heruntermischen

Beim Heruntermischen des HF-Signals kann über den Parameter Fdev eine Abweichung auf die Trägerfrequenz addiert werden. Dann wird durch eine Amplituden-Demodulation das Basisbandsignal wiederhergestellt. Dieses Basisbandsignal wird mit derselben Sampling-Frequenz des Basisbandes dargestellt wie bei der Pulsformung in Abschnitt 3.2.1.4 auf Seite 73.

$$\overrightarrow{BB_{Rx}} = \overrightarrow{HF_{Rx}} \cdot cos((f_C + F_{dev}) \cdot t) \tag{3.29}$$

3.2.3 Empfänger

Der Empfänger besteht aus drei großen Blöcken: Der analogen Vorverarbeitung mit Integrator, Quadrierer, Verstärker und ADC, dem Basisband-Empfänger aus Präambel-Detektor und Daten-Demodulator und letztendlich der Dekodierung des PHY-Headers einerseits und der Nutzdaten andererseits. Abbildung 3.9 auf der nächsten Seite zeigt schematisch den Aufbau des Empfängers.

3.2.3.1 Verstärker

Es wird ein statischer Wert *amp* zur Verstärkung benutzt, so dass der mittlere Signalpegel = 1 ist. Dadurch ist der Signalpegel der Pulse ≥ 1 und das Rauschen ≤ 1.

[8] Additive White Gaussian Noise - *addiertes Gauß-Rauschen*

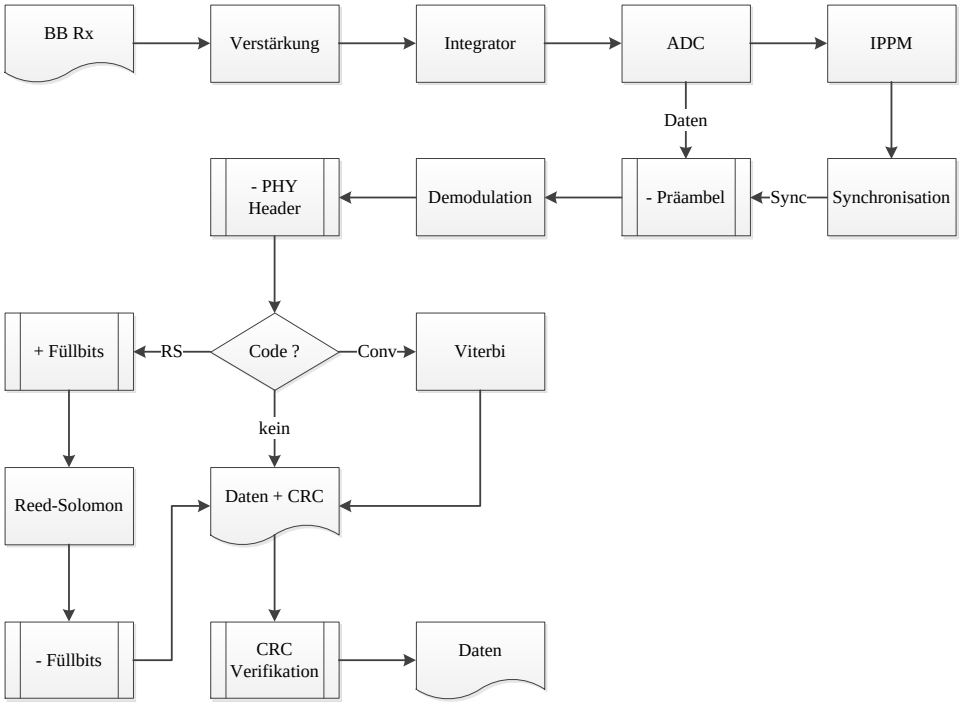


Abbildung 3.9: Schematische Darstellung des Empfängers

$$amp = \frac{\sum \overrightarrow{BB_{Rx}}}{|\overrightarrow{BB_{Rx}}|} \tag{3.30}$$

$$\overrightarrow{BB_{Rx}} = \overrightarrow{BB_{Rx}} \cdot amp \tag{3.31}$$

Nach dem Verstärken kann das Signal optional „gleichgerichtet“, also auf positive Werte gebracht werden, um eine Auslöschung entgegengesetzter Pulse zu vermeiden. Dazu kann das Signal entweder im Betrag genommen werden, entsprechend einem I/Q-Mischer,

$$\overrightarrow{BB_{Rx}} = \overrightarrow{BB_{Rx}} \times \frac{\overrightarrow{BB_{Rx}}}{\overrightarrow{BB_{Rx}}} \tag{3.32}$$

oder quadriert werden, entsprechend einer Gilbert-Zelle,.

$$\overrightarrow{BB_{Rx}} = \overrightarrow{BB_{Rx}} \times \overrightarrow{BB_{Rx}} \tag{3.33}$$

Dabei heben sich durch die quadratische, nicht-lineare Verstärkung Pulse deutlicher vom Rauschen ab, als bei der Betragsoperation.

Mit dem Parameter rectify kann angegeben werden, welche Methode zur Gleichrichtung angewendet wird, oder ob das Signal vorzeichenbehaftet bleibt.

3.2.3.2 Integrator

Beim Integrator kann durch den Parameter intlen die Länge des Integrationsintervalls T_s festgelegt werden. Über diese Zeit wird das Integral des Zeitdomänen-Signals ermittelt.

$$\overrightarrow{BB_{Int}}(n) = \int_{n \cdot T_s}^{(n+1) \cdot T_s} \overrightarrow{BB_{Rx}}(t) dt \tag{3.34}$$

3.2.3.3 ADC

Im *ADC* wird der Signalvektor auf diskrete Werte reduziert. Die Anzahl dieser Werte entspricht der Anzahl ENOB an effektiven Bits im Ausgabewert des *ADC*. Diese wird über den Parameter ADCbits angegeben.

$$\overrightarrow{BB_{ADC}} = \frac{\overrightarrow{BB_{Int}}}{\max(\overrightarrow{BB_{Int}})} \cdot 2^{ENOB} \tag{3.35}$$

Die Ausgabe des *ADC* ergibt einen Vektor von digitalisierten Samples.

3.2.3.4 Präambel-Detektion

Der erste Schritt der Präambel-Detektion erfolgt durch eine Kreuzkorrelation des Signals gegen den bekannten Präambel-Code.

$$Corrval_n = \sum_{i=1}^{31} \overrightarrow{C_P}(i) \cdot \overrightarrow{BB_{ADC}}(i+n) \tag{3.36}$$

Die erste Spitze des Betrages des Korrelationswertes wird durch das erste Maximum, das über einem Schwellenwert *thresh* liegt[9], erkannt und deutet auf ein gefundenes Präambel-Symbol hin. Ab jedem vermuteten erstem Präambel-Symbol PF_n wird ein Vektor der k folgenden Präambel-Symbole $\overrightarrow{PF}(n)$ selektiert.

$$\overrightarrow{PF}(n)(k) = \overrightarrow{Corrval}(n + k \cdot 31) \; \forall \; n \; / \; |Corrval_n| > thresh \tag{3.37}$$

Die letzten acht Elemente des Vektors $\overrightarrow{PF}$ werden mit dem *SFD*-Code $\overrightarrow{C_{SFD}}$ verglichen. Der *SFD* ist gefunden, wenn es eine Vermutung $\overrightarrow{PF}(n)$ gibt, bei der an jeder Stelle $\overrightarrow{PF}(n)(k)$, an welcher der Betrag des *SFD*-Codes $|C_{SFD}(k)| = 1$ ist, der Wert von $\overrightarrow{PF}(k) \geq thresh$ ist.

$$S = n + N_{Prep} \cdot |\overrightarrow{Psym}| \tag{3.38}$$

wenn

$$\overrightarrow{PF}(n)(S - 7 + k) \geq thresh \; \forall \; k \; / \; |C_{SFD}(k)| = 1 \tag{3.39}$$

Sobald der *SFD*-Code in PF gefunden ist, ist das Ende der Präambel S erreicht. Wurde kein *SFD* im Vektor gefunden, ist die Synchronisation fehlgeschlagen.

Bei erfolgreicher Synchronisation wird der Vektor $\overrightarrow{BB_{ADC}}$ ab dem Element S zur Daten-Demodulation verwendet.

[9] Ich habe Verfahren zur Bestimmung und Anpassung von *thresh* implementiert, diese haben allerdings keine Relevanz für diese Arbeit.

3.2.3.5 Daten-Demodulation

Um den Datenteil zu demodulieren, werden jeweils die Samples, die einer Hopping-Position entsprechen, aufsummiert.

$$HP_n = \sum_{i=1}^{N_{cps}} \overrightarrow{BB_{ADC}}(n \cdot T_s + i) \tag{3.40}$$

Dazu ist die Länge einer Hopping-Position in Samples N_{cps} nötig, die sich aus der Länge des *Integrationsintervalls* T_s, der Länge des Pulszeitraumes T_C sowie der Anzahl der Pulse pro *Burst* N_{CPB} berechnet.

$$N_{cps} = \frac{N_{cpb} \cdot T_C}{T_s} \tag{3.41}$$

Außerdem wird die ermittelte *THS* benötigt. Mit den Summen $\overrightarrow{HP}$ werden für jedes Symbol jeweils die beiden der *THS* entsprechenden Hopping-Positionen für den Bit-Wert „0“ Z_i und „1“ O_i verglichen.

$$\begin{aligned}
\overrightarrow{Z}(i) &= \overrightarrow{HP}(4 \cdot N_{Hops} \cdot i + \overrightarrow{THS}_i) && (3.42)\\
\overrightarrow{O}(i) &= \overrightarrow{HP}(4 \cdot N_{Hops} \cdot i + \overrightarrow{THS}_i + 2 \cdot N_{Hops}) && (3.43)\\
\overrightarrow{\Delta E}(i) &= \overrightarrow{O}(i) - \overrightarrow{Z}(i) && (3.44)
\end{aligned}$$

Aus dem Vorzeichen von ΔE resultieren die demodulierten Bit-Werte.

$$RBit(n) = \frac{\Delta E(n)}{|\Delta E(n)|} > 0 \tag{3.45}$$

Diese Bit-Folge wird der Funktion zur Dekodierung des Symbols übergeben.

3.2.3.6 Dekodierung

Die ersten 19 Bits des Paketes entsprechen dem *PHY*-Header. Die letzten 6 Bits davon sind ein *SECDED*-Code.

$$\overrightarrow{C_R} = \{RBit_{13}, ..., RBit_{19}\} \tag{3.46}$$

Dieser wird zur Verifizierung und gegebenenfalls Reparatur des *PHY*-Headers benutzt, indem das sogenannte Syndrom durch

$$\overrightarrow{Syndrom} = \overrightarrow{C_R} \otimes \overrightarrow{C_L} \tag{3.47}$$

ermittelt wird. Dabei wird über die ersten empfangenen 13 Bits mit den Formeln (3.3) bis (3.8) auf Seite 70 die lokale Prüfsumme $\overrightarrow{C_L}$ berechnet und per XOR-Operation ($\otimes$) mit der empfangenen Prüfsumme $\overrightarrow{C_R}$ verglichen.

Dieses Syndrom ermöglicht es, einen Fehler zu lokalisieren oder zwei Fehler festzustellen. Dazu wird das sechste Bit des Syndroms $\overrightarrow{Syndrom_5}$ betrachtet. Wenn dieses Bit den Wert '1' hat, ist ein Fehler aufgetreten. Über die restlichen Bits des Syndroms lässt sich dann die Position des Fehlers bestimmen und der Fehler korrigieren. Hat $\overrightarrow{Syndrom_5}$ den Wert '0' haben, so sind entweder kein Fehler oder eine gerade Anzahl an Fehlern aufgetreten. Dies kann anhand der restlichen Bits des Syndroms ermittelt werden, denn wenn mindestens eines davon nicht den Wert '0' hat, sind mindestens zwei Fehler aufgetreten.

Wenn keine Fehler aufgetreten sind, oder der Fehler repariert werden konnte, beginnt die Datenauswertung des *PHY*-Headers. Mit Hilfe dieser Informationen ist es dann möglich, das Paket zu dekodieren. Entsprechend der in Parameter fec_code angegebenen Kodierung wird dazu entweder ein *Reed-Solomon-Decoder* oder ein *Viterbi-Decoder* eingesetzt.

Für den *Reed-Solomon-Code* muss der Bit-Vektor allerdings erst in 6-Bit-Symbole aufgeteilt werden.

$$RSym_n = \{RBit(n \cdot 6), ..., RBit((n+1) \cdot 6 - 1)\} \tag{3.48}$$

Außerdem muss die Anzahl der Code-Blöcke

$$N_{Block} = \frac{|\overrightarrow{RSym}|}{63} - 1 \tag{3.49}$$

ermittelt werden und gegebenenfalls muss der letzte Block auf die Blockgröße des Codes von 55 Symbolen aufgefüllt werden. Diese Füll-Bits müssen zwi-

schen den Nutzdaten und den acht Prüfsymbolen, die das Paket abschließen, eingefügt werden.

$$N_{Fill} = |\overrightarrow{RSym}| - N_{Block} \cdot 63 - 7 \tag{3.50}$$

$$\overrightarrow{RSym} = \bigsqcup_{n=1}^{N_{Block}\cdot 63+55-N_{Fill}} \overrightarrow{RSym}(n) \sqcup \bigsqcup^{N_{Fill}} \{0\} \sqcup \bigsqcup_{m=|\overrightarrow{RSym}|-8}^{|\overrightarrow{RSym}|} \overrightarrow{RSym}(m) \tag{3.51}$$

Nach der Dekodierung des *FEC*-Codes wird die empfangene Nutzlast des Paketes $\overrightarrow{RNL}$ durch Vergleich der empfangenen 16-Bit-*CRC*-Prüfsumme $\overrightarrow{CRC_{Tx}}$ mit der lokal berechneten 16-Bit-*CRC*-Prüfsumme

$$\overrightarrow{CRC_{Rx}} = \mathrm{crc16}(\overrightarrow{RNL}) \tag{3.52}$$

verifiziert. Es wird kein Fehler festgestellt, wenn der bitweise Vergleich durch die XOR-Operation $\otimes$

$$\overrightarrow{CRC_{Tx}} \otimes \overrightarrow{CRC_{Rx}} = 0 \tag{3.53}$$

keine Unterschiede zeigt.

Andernfalls wird zur Untersuchung der Anzahl und Lage von Bit-Fehlern einer Übertragung wird ein Vektor $\overrightarrow{Err}$ der Unterschiede zwischen $\overrightarrow{NL}$ und $\overrightarrow{RNL}$ verwendet.

$$\overrightarrow{Err} = \overrightarrow{RNL} \otimes \overrightarrow{NL} \tag{3.54}$$

Mit diesen Analysewerkzeugen war ich in der Lage, Fehlerquellen ausfindig zu machen und den Einfluss von veränderten Konfigurationen auf diese Fehlerquellen zu untersuchen. Daraus entwickelte ich Optimierungen, um *IR-UWB* besser für die industrielle Automatisierung einsetzen zu können. Zur Verifikation der Optimierungen erweiterte ich das Modell und konnte damit die Veränderung quantifizieren.

Kapitel 4

Defintion und Analyse eines robusten UWB-Systems für die drahtlose Automatisierung

In diesem Kapitel stelle ich ein robustes, echtzeitfähiges, drahtloses Kommunikationssystem auf *UWB*-Basis vor. Dabei verfolgte ich einen Top-Down-Ansatz. Zielsetzung war ein System, das mit bis zu 32 Teilnehmern harte Echtzeitanforderungen von 20 ms bei einer mittleren Reaktionszeit unter 5 ms erfüllen kann und dabei eine Fehlerrate von weniger als 10^{-9} erreicht.

Ausgehend vom *AS-Interface*-Netzwerkprotokoll der Sensor/Aktor-Ebene, das in Abschnitt 2.2.1.2 auf Seite 19 genauer vorgestellt wurde, erstellte ich ein drahtloses System auf Basis von *UWB*, um dieses Protokoll zu ersetzen. Dafür habe ich zuerst Anforderungen an eine *MAC*-Schicht definiert. Sie muss eine deterministische Zykluszeit unter 5 ms und einen Zusatzaufwand pro Paket bieten, der die oben genannten zeitlichen Anforderungen erfüllen kann. Im Standard-Entwurf IEEE 802.15.4e fand ich eine den Anforderungen genügende *MAC*-Schicht. Darauf folgte die Optimierung der Robustheit der *PHY*-Schicht, um industriellen Anforderungen genügen zu können und verifizierte diese Optimierungen durch Simulationen. Letztlich konnte ich durch schichtübergreifenden Informationsaustausch (*Cross-Layer*-Technik) Latenz und Robustheit weiter verbessern, ebenfalls durch Simulationen verifiziert.

4.1 Definition der Anforderungen und der Zielapplikation

Als zugrundeliegende Applikation wird ein typisches Automatisierungssystem mit zentralem Controller und mehreren Sensor/Aktor-Achsen angenommen. In dieser Netzstruktur ist ein Controller sternförmig mit Sensoren und Aktoren verbunden. Jedem Aktor ist mindestens ein Sensor zugeordnet, aufgrund dessen Daten er gesteuert wird. Eine solche Paarung wird als Achse bezeichnet. Diese Art des Netzaufbaus ist typisch für die Automatisierung und wird unter anderem in Fertigungszellen verwendet.

Die Anzahl der Achsen in einer Fertigungszelle hängt von der Art der Fertigung ab: Gentry-Fräsmaschinen haben üblicherweise fünf oder mehr Achsen [36, S. 221], [47] zeigt Industrie-Roboterarme mit je sechs Achsen und [12] stellt eine automatisierte Fertigungszelle einer Blechbiegemaschine mit 12 Achsen vor.

Bei der Wahl des *MAC*-Protokolls wurde deshalb die Zykluszeit in Abhängigkeit der Anzahl der Achsen betrachtet. Jede Achse bedingt mindestens zwei Knoten, einen Aktor und mindestens einen Sensor. In der weiteren Ausführung stellen die Symbole N_{Knoten} die Anzahl der Knoten und T_{Zyklus} die Zykluszeit der Applikation dar.

4.1.1 Knotentypen

Im o.g. Szenario kann man zwischen drei fundamentalen Knotentypen unterscheiden, wobei in der Realität ein einzelner Knoten auch mehrere logische Rollen annehmen kann.

4.1.1.1 Controller

Der *Controller* ist die zentrale Steuerungseinheit der Zelle. Er hat üblicherweise eine externe Stromversorgung und ist über ein weiteres Netz (sog. *„Backhaul Network“*) mit einem Steuerungsrechner verbunden. Dabei ist es auch möglich, dass ein Teil der Steuerungsintelligenz direkt auf dem Controller ausgeführt

wird. Eine weitere Dezentralisierung der Steuerung mit Steuerungslogik bzw. Datenvorverarbeitung auf einem Sensor bzw. Aktor wird nicht angenommen.

4.1.1.2 Sensor

Ein Sensor-Knoten stellt eine Einheit aus dem eigentlichen Sensor, der einen physischen Wert misst und, um diesen Wert zu übertragen, einem Adapter für das Drahtlosnetz dar. Ein Sensor hat im Regelbetrieb die Aufgabe, die Steuerung mit Informationen über den Zustand der zu steuernden Maschine zu versorgen.

In den meisten Funksystemen verbraucht das Empfangen bzw. das Warten auf Empfang mehr Energie als das Senden. Der erhöhte Energieverbrauch der Empfangsbereitschaft eines Knotens fällt nicht nur beim direkten Empfang an, sondern auch in den Zeiträumen, in denen der Empfänger auf ein Paket wartet, das sogenannte „*Idle Listening*“. Deshalb sollte ein Sensor möglichst selten Pakete empfangen müssen. Dafür werden lange Ruhepausen zwischen zyklischen Empfangsperioden eingesetzt. Das Verhältnis der Empfangsperiode zur Ruhepause wird als „*Duty Cycle*“ bezeichnet.

Ein geringer *Duty Cycle* ist besonders in Situationen wichtig, bei denen Sensoren keine ständige Energieversorgung haben. So kann ein Sensor beispielsweise auch batteriebetrieben arbeiten, seine Energie durch *Energy Harvesting* ernten oder durch einen Kondensator versorgt werden, der nur in bestimmten Prozessabschnitten geladen wird (zum Beispiel Sensoren auf einem Roboterarm).

Im Regelbetrieb wird ein Sensor nur Synchronisationsnachrichten empfangen und seine Messdaten versenden. Über diese Synchronisationsnachricht kann dann vom Controller auch eine längere Nachricht zur Rekonfiguration angekündigt werden. Dies erlaubt jedem einzelnen Sensor lange Ruhezeiten, was vorteilhaft für Energiebudget und Energieverbrauch ist.

4.1.1.3 Aktor

Ein *Aktor* bindet einen Motor, ein Ventil, eine Bremse oder ähnliche aktive Komponenten der zu steuernden Maschine in das Drahtlosnetz ein. Da diese

Geräte generell eine ständige Stromversorgung benötigen, ist ein *Aktor*-Knoten üblicherweise mit an diese Stromversorgung angebunden.

Die Aufgabe des Aktors ist es, Steuernachrichten vom Controller an das angebundene Gerät weiterzugeben. Dazu muss ein *Aktor* üblicherweise nur empfangen. Das *MAC*-Protokoll soll jedoch auch die Möglichkeit bieten, dass ein Aktor senden kann, z.B. um Ausnahmezustände oder Defekte melden zu können.

4.1.2 Kommunikationsverhalten

In dieser Arbeit nehme ich keine direkte Kommunikation innerhalb einer Achse, also keine direkte Kommunikation zwischen Sensor und Aktor an. Jeder Sensor oder Aktor kommuniziert ausschließlich mit dem Controller, was eine sternförmige Topologie bedingt. Der Controller stellt der übergeordneten Steuerung die Sensordaten zur Verfügung und erhält bzw. generiert Steuerdaten für die Aktoren.

4.2 Modellierung einer MAC-Schicht für harte Echtzeitanforderungen

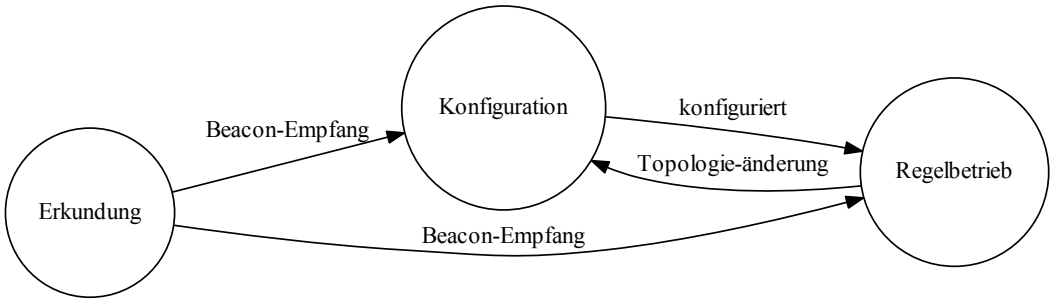


Abbildung 4.1: Zustände der MAC-Schicht

Es gibt mehrere Zustände im Betrieb eines Netzes. Kurz nach dem Einschalten muss das Netz konfiguriert werden und bei Änderung der Topologie eine

Rekonfiguration stattfinden. Sensoren und Aktoren starten in einem Erkundungszustand, in dem sie auf den Empfang eines *Beacon* warten. Zuerst wird allerdings der Normalbetrieb betrachtet. In diesem Normalbetrieb muss der Controller innerhalb einer Zykluszeit einmal mit jedem Knoten kommunizieren, um Sensordaten bzw. Steuerbefehle weiterzuleiten.

4.2.1 Zeit- und Synchronisationsanforderungen

Innerhalb einer Fertigungszelle aus Controller und mehreren Sensoren und Aktoren muss der Medienzugriff kontrolliert werden, also eine Regelung gefunden werden, wann welcher Teilnehmer sendet und welche Teilnehmer empfangen. Hierbei wird zwischen verteilten und zentralen Verfahren unterschieden. Im Gegensatz zum verteilten Verfahren, bei dem jeder Netzwerkknoten gleichwertig ist, weist bei einem zentralen Verfahren eine Kontrollinstanz jedem Netzteilnehmer Übertragungsrechte zu.

In dieser Arbeit wird ein zentraler Ansatz verfolgt, da die logische Struktur des Netzes bereits einen *Controller* enthält.

Für das drahtlose Netzwerk stellt der *Controller* den koordinierenden Knotenpunkt dar. Alle dem Netz angehörenden Knoten müssen sich in direkter Funkreichweite des Controllers befinden[1], da der Medienzugriff vom Controller gesteuert wird.

Zur Regelung, wann welcher Knoten senden bzw. empfangen muss, gibt es diverse Verfahren, die sich durch die Aufteilung des Funkmediums und die Koordination dieser Aufteilung unterscheiden. Das Medium kann durch verschiedene Frequenzkanäle *FDMA*[2], verschiedene Codes *CDMA*[3] oder durch die Verteilung von Zeitschlitzen *TDMA* aufgeteilt werden.

[1] Die Möglichkeit, die Reichweite des Steuerungsnetzes durch Multi-Hop zu erhöhen, wurde nicht in Betracht gezogen, da dies die Latenz und Fehlerrate erhöhen und Maßnahmen zur Routenfindung und Nachbarschaftserkundung nach sich ziehen würde, die das Netz verlangsamen würden.

[2] Frequency Division Multiple Access - *Zugriffsverfahren auf Basis verschiedener Frequenzkanäle*

[3] Code Division Multiple Access - *Zugriffsverfahren auf Basis individueller Codes*

Aufbauend auf dem *AS-Interface*-Protokoll soll eine optimierte *MAC*-Schicht für ein drahtloses System auf Basis von *UWB* gefunden werden. Dazu wird das Zeitverhalten der *MAC*-Schicht in Abhängigkeit von der Anzahl der Knoten betrachtet. Den Anforderungen des Automatisierungssystems entsprechend wird eine feste Zykluszeit T_{Zyklus} gesetzt. Um diese einzuhalten, muss die *MAC*-Schicht einen kompletten Kommunikationszyklus zwischen dem Controller und allen zugeordneten Knoten innerhalb von dieser Zykluszeit T_{Zyklus} durchführen können. T_{Zyklus} hängt stark von möglichen Modi der *PHY*-Schicht sowie der Anzahl der Knoten und Länge der Nutzdaten ab, was später betrachtet wird (siehe Abschnitt 4.5 auf Seite 147).

4.2.1.1 Polling

Sowohl das *AS-Interface*-Protokoll als auch das IO-Link-Protokoll verwenden ein *Polling*-Verfahren, bei dem der Controller in einer festgelegten Reihenfolge jeweils ein Paket an die einzelnen Sensoren und Aktoren schickt, das diese sofort beantworten. Dabei erfolgt die Synchronisation über ein Start- und ein Stoppbit (siehe auch Abschnitt 2.2.1.2 auf Seite 19). Da das *AS-Interface*-System drahtgebunden überträgt, ist ein *AGC*-Setting nicht nötig [7].

Die Zeit, die ein drahtloses System mit einem solchem *MAC*-Ansatz auf Basis von zyklischem *Polling* für einen Zyklus benötigt, lässt sich deshalb generell durch

$$T_{Zyklus}(Polling) = N_{Knoten} \cdot (T_{wait} + T_{Anfrage} + T_{wait} + T_{Antwort}) \qquad (4.1)$$

berechnen. Dabei steht T_{wait} für einen zeitlichen Sicherheitsabstand, bedingt durch die Reaktionszeit eines Knotens. $T_{Anfrage}$ und $T_{Antwort}$ stehen für die Übertragungsdauer der jeweiligen Pakete.

In einem drahtlosen System, besonders in anspruchsvollen Umgebungen, benötigt jedes Paket eine Präambel zur Synchronisation und zur *AGC*-Einstellung, außerdem einen *PHY*-Header und Sicherungsmechanismen wie einen *CRC*-Code. Für den Vergleich der Zykluszeiten verschiedener *MAC*-Ansätze wird ein typischer Paketaufbau, wie er in Abbildung 4.2 auf der nächsten Seite ge-

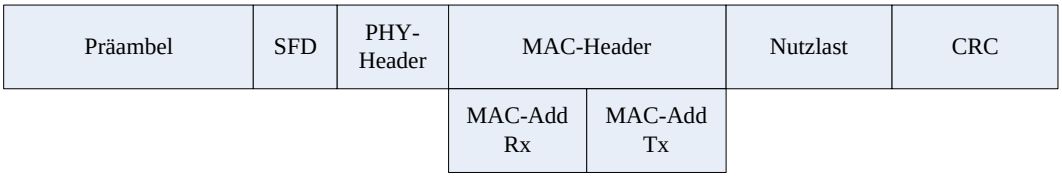


Abbildung 4.2: Angenommener Paketaufbau für den theoretischen Vergleich der Zykluszeiten verschiedener MAC-Ansätze

zeigt ist, angenommen. Deshalb können $T_{Anfrage}$ und $T_{Antwort}$ ersetzt werden durch eine von der Nutzlast abhängige Paketlänge

$$T_{Pkt} = T_{Prea} + T_{PHR} + (2 \cdot N_{MAdd} + N_{Nutzlast} + N_{CRC}) \cdot T_{sym} \tag{4.2}$$

wobei T_{Prea} die Dauer der Präambel, T_{PHR} die Länge des *PHY*-Headers, N_{MAdd} die Länge einer *MAC*-Adresse (in Bits) und N_{CRC} die Anzahl der Bits für den *CRC*-Code symbolisieren. Zur Vereinfachung wird ein minimaler *MAC*-Header, bestehend nur aus Quell- und Zieladresse angenommen und dass mit einem Symbol immer nur ein Bit übertragen wird.

Zur Vereinfachung der Berechnung wird weiterhin angenommen, dass die *Polling*-Anfrage und die darauffolgende Antwort gleich lang sind (gleich große $N_{Nutzlast}$). Dadurch kann Formel 4.2 in Formel 4.1 auf der vorherigen Seite eingesetzt werden und zu

$$\begin{aligned} T_{Zyklus}(Polling) &= 2 \cdot N_{Knoten} \cdot (T_{wait} + T_{Pkt}) \\ &= 2 \cdot N_{Knoten} \cdot [[T_{wait} + T_{Prea} + T_{PHR} \\ &+ (2 \cdot N_{MAdd} + N_{Nutzlast} + N_{CRC}) \cdot T_{sym}] \end{aligned} \tag{4.3}$$

umgeformt werden.

4.2.1.2 Multi-Polling

Um eine geringere T_{Zyklus} zu erhalten, sendet der Controller ein *Beacon* aus, in welchem er eine Liste der zu pollenden Knoten und deren Nutzdaten verschickt.

Diese senden ihre Antworten nacheinander in der Reihenfolge, die in der Liste angegeben ist. Auf diese Weise werden pro Zyklus statt $2 \cdot N_{Knoten}$ nur noch $N_{Knoten} + 1$ Pakete verschickt, wobei die Paketlänge von Beacon und Antwort verschieden ist.

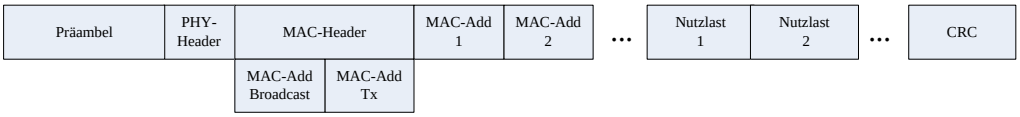


Abbildung 4.3: Anufbau des Beacons im Multipolling-Ansatz

In diesem Ansatz enthält der *Beacon*, wie in Abbildung 4.3 gezeigt, die Nutzlast der Anfrage für jeden adressierten Knoten. Diese Nutzlast wird für jeden Knoten als gleich lang angenommen, ebenso lang wie wie die Nutzlast der Antworten. Weiterhin ist eine Liste von *MAC*-Adressen enthalten, die die Reihenfolge der erwarteten Antworten vorschreibt. Damit lässt sich die Länge eines *Beacon*s durch

$$\begin{aligned} T_{Beacon}^{MultiPoll} &= T_{wait} + T_{Prea} + T_{PHR} \\ &\quad + (2 \cdot N_{MAdd} + N_{Knoten} \cdot N_{Nutzlast} \\ &\quad + N_{Knoten} \cdot N_{MAdd} + N_{CRC}) \cdot T_{sym} \end{aligned} \tag{4.4}$$

ausdrücken, wodurch die Zykluszeit durch

$$\begin{aligned} T_{Zyklus}(MultiPoll) &= T_{Beacon}^{MultiPoll} + N_{Knoten} \cdot T_{Pkt} \\ &= (N_{Knoten} + 1) \cdot [T_{wait} + T_{Prea} + T_{PHR} \\ &\quad + (2 \cdot N_{MAdd} + N_{Nutzlast} + N_{CRC}) \cdot T_{sym}] \\ &\quad + N_{Knoten} \cdot (N_{MAdd} + N_{Nutzlast}) \cdot T_{sym} \end{aligned} \tag{4.5}$$

ausgedrückt werden kann.

In diesem Ansatz müssen weniger Wartezeiten eingehalten und weniger Präambeln übertragen werden, dadurch ist die Zykluszeit geringer als im Polling-Ansatz.

4.2.1.3 TDMA mit Zeitschlitzadressierung

Ein weiterer Ansatz reduziert die Anzahl der übertragenen MAC Ãdressen. Dazu wird jedem Knoten im Netz zur Konfigurationszeit ein fester Zeitschlitz im Zyklus zugewiesen (z.B. über eine MAC-Adresse), also wie bei einem klassischen $TDMA$-Verfahren. Dadurch ergibt sich die Adressierung der Übertragung anhand ihrer Position im Zyklus. Der Controller eröffnet den Zyklus, hier auch *Superframe* genannt, mit einem *Beacon*, in welchem er die Konfiguration des *Superframe*s mitteilt, also die Anzahl der Zeitschlitze, Länge der Nutzlast etc. Zur vereinfachten Berechnung wird die Länge dieser Konfigurationsinformation der Länge einer Nutzlast gleichgesetzt.

Der *Beacon* enthält außerdem eine Nutzlast (in der gleichen Länge wie die Nutzlast der Antworten) für jeden Knoten im Netz. Jedem *Beacon* folgen mehrere Zeitschlitze, in welchen die Knoten in einer fest konfigurierten Reihenfolge ihre Antworten übertragen.

Die Zeit, die der *Beacon* benötigt, reduziert sich damit auf

$$T_{Beacon}^{TDMA} = T_{wait} + T_{Prea} + T_{PHR} + (N_{Knoten} \cdot N_{Nutzlast} + N_{CRC}) \cdot T_{sym} \quad (4.6)$$

wodurch sich die Zykluszeit als

$$\begin{aligned} T_{Zyklus}(TDMA) &= T_{Beacon}^{TDMA} + N_{Knoten} \cdot T_{Pkt} \\ &= (N_{Knoten} + 1) \cdot [T_{wait} + T_{Prea} + T_{PHR} \\ &+ (N_{Nutzlast} + N_{CRC}) \cdot T_{sym}] \\ &+ (N_{Knoten} \cdot N_{Nutzlast}) \cdot T_{sym} \end{aligned} \quad (4.7)$$

ausdrücken lässt.

4.2.1.4 Rollenbasiertes TDMA

Zusätzlich zu der speziellen Rolle des Controllers können den Knoten Rollen als Sensor bzw. Aktor zugeteilt werden, wie in Abschnitt 4.1.1 auf Seite 90 beschrieben.

Dadurch trägt der *Beacon* selbst keine Nutzlast mehr, sondern eröffnet lediglich einen *Superframe* und gibt dessen Konfiguration bekannt. In den folgenden

Zeitschlitzen wird unidirektional die Nutzlast zu jedem Aktor bzw. von jedem Sensor übertragen.

Die Zeit, die ein *Beacon* in Anspruch nimmt, verkürzt sich damit auf

$$T_{Beacon}^{Role} = T_{wait} + T_{Prea} + T_{PHR} + (N_{Nutzlast} + N_{CRC}) \cdot T_{sym} \tag{4.8}$$

wodurch sich eine reduzierte Zykluszeit von

$$\begin{aligned} T_{Zyklus}(Role) &= T_{Beacon}^{Role} + N_{Knoten} \cdot T_{Pkt} \\ &= (N_{Knoten} + 1) \cdot [T_{wait} + T_{Prea} + T_{PHR} \\ &\quad + (N_{Nutzlast} + N_{CRC}) \cdot T_{sym}] \end{aligned} \tag{4.9}$$

ergibt.

4.2.1.5 Vergleich

Für die vier in diesem Abschnitt vorgestellten Arten von *MAC*-Protokollen bzw. Kommunikationsschemata wurden Zykluszeiten mit einer steigenden Anzahl von Knoten berechnet. Als Obergrenze wurden 62 Knoten genommen, da dies die maximale Anzahl Knoten in *AS-Interface* ist. Wie in Abbildung 4.4 auf der nächsten Seite zu sehen ist, werden die kürzesten Zykluszeiten mit dem rollenbasierten Ansatz erzielt. Wie weiterhin erkennbar ist, wird durch die Einführung eines Beacon beim Group-Polling die Zykluszeit verkürzt. Dies liegt daran, das der Zeitbedarf des *Beacon* geringer ist als der eingesparte Zusatzaufwand der Polling-Anfragen.

4.2.2 Definition der MAC-Schicht auf Basis von Draft IEEE 802.15.4e

Für das in dieser Dissertation vorgestellte System wurde eine *MAC*-Schicht aus dem Entwurf des Standards IEEE 802.15.4e [1] ausgewählt. Diese enthält einen Modus für sternförmige Netzwerke mit kurzer Latenz, genannt *LL-Star*. Diese *MAC*-Schicht nutzt die in Abschnitt 4.2.1.4 auf der vorherigen Seite vorgestellte rollenbasierte *TDMA*-Struktur. Sie bildet deshalb das Kommunikationsschema mit dem geringstmöglichen Zusatzaufwand auf drahtlose Kommunikation ab.

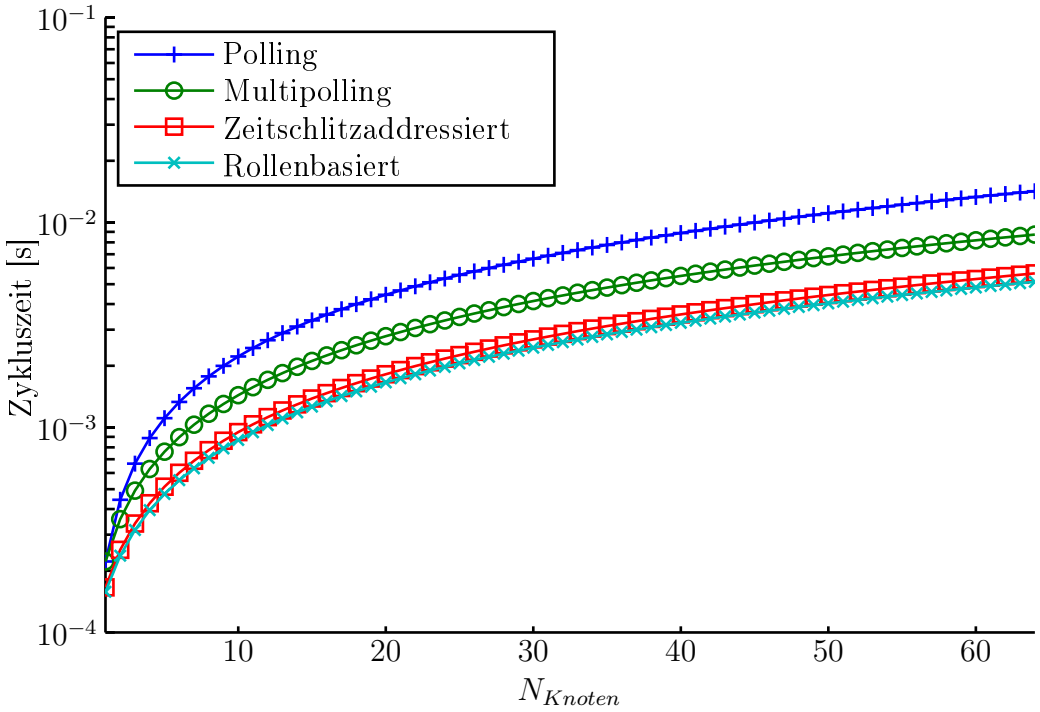


Abbildung 4.4: Vergleich der Zykluszeiten der verschiedenen Kommunikationsschemata

Da dieser Entwurf nicht öffentlich zugänglich ist, wird an dieser Stelle ein kurzer Überblick über die *MAC*Schicht gegeben. Die in diesem Abschnitt vorgestellten Paketstrukturen sind jedoch nicht Teil meiner Arbeit, sondern wurden von der zuständigen *Task Group* 802.15.4e erstellt.

4.2.2.1 Superframe

Der *Superframe* besteht aus mehreren gleichgroßen Zeitschlitzen, genannt *BTS*[4]. Jeweils aus einem oder mehreren dieser *BTS* werden Zeitschlitze für *Beacon*, Management-Frames und Knoten (also Sensoren und Aktoren) gebildet. Die Länge des *BTS* errechnet sich aus der Länge eines Frames mit einer definierten Anzahl Bytes $N_{BTSBytes}$ und einem Guard-Intervall aus 12 Präambel-Symbolen[5].

[4]Base Time Slot - *Basiszeitschlitz*

[5]Dieselbe Länge des Guard-Intervalls, wie im Standard IEEE 802.15.4 gefordert [2]

Ein *Superframe* im Regelbetrieb besteht dabei aus einem *Beacon*-Zeitschlitz und einem Zeitschlitz für jeden Knoten im Netzwerk, wie in Abbildung 4.5 dargestellt. Dabei wird ein *Superframe* durch den *Beacon* eröffnet, dann folgen alle Sensoren und daraufhin die Aktoren. Jeder Zeitschlitz kann sich in seiner Länge unterscheiden, wobei die Länge immer ein ganzzahliges Vielfaches des *BTS* sein muss.

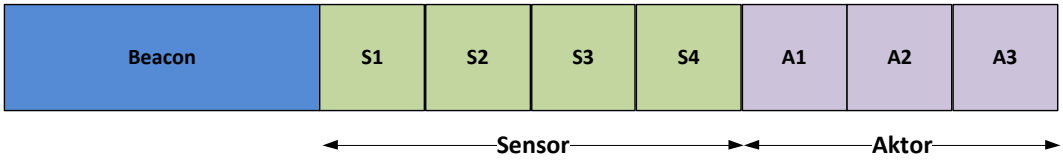


Abbildung 4.5: Superframe im Regelbetrieb

Jedem Knoten ist ein Zeitschlitz zugeordnet, der seiner *MAC*-Adresse entspricht.

4.2.2.2 Frame

Jeder Frame hat einen 8 Bit langen Header, der die in Tabelle 4.1 angegebenen Informationen enthält.

Tabelle 4.1: MAC-Header eines Frames nach Draft IEEE 802.15.4e

Feld	**Länge**	**Beschreibung**
Typ des Frames	3 Bit	Kompatibilität zu IEEE 802.15.4, Wert b100
Security	1 Bit	Kompatibilität zu IEEE 802.15.4, Wert b0
Frame-Version	1 Bit	Kompatibilität zu IEEE 802.15.4, Wert b0
ACK-Anforderung	1 Bit	Kompatibilität zu IEEE 802.15.4, Wert b0
Subtyp des Frames	2 Bit	Spezifiziert die Art des Frames, s. Tab. 4.2

Der Wert „100“ für den Frametyp sagt aus, dass es sich um einen Frame des IEEE-802.15.4e-Protokolls handelt, der in Tabelle 4.2 auf der nächsten Seite angegebene Subtyp spezifiziert letztlich die Art des Frames innerhalb des IEEE-802.15.4e-Protokolls.

Tabelle 4.2: Subtypen von Frames nach Draft IEEE 802.15.4e

Wert des Feldes	**Subtyp**
b00	Beacon
b01	Kommando
b10	ACK
b11	Daten

Auf den Header folgt die Nutzlast und darauf eine 16 Bit lange ITU-*CRC*-Prüfsumme über *MAC*-Header und Nutzlast. Insgesamt addiert diese *MAC*-Schicht also einen Zusatzaufwand von 3 Byte zur Nutzlast, was im Vergleich zu den üblichen proprietären Automatisierungssystemen zwar viel, für eine standardisierte *MAC*-Schicht aber wenig ist.

4.2.2.3 Beacon

Der *Beacon* wird vom *Controller* versendet und erfüllt mehrere Aufgaben:

- Er eröffnet den Superframe
- Er synchronisiert die Knoten im Netz
- Er übermittelt die Länge des Superframes in *BTS*
- Er übermittelt die Länge eines *BTS*
- Er enthält ein Bit-Feld, das den Empfang von Paketen im letzten Superframe anzeigt (*Group ACK*)[6]

Diese notwendigen Informationen liefert der *Beacon*-Frametyp von Draft IEEE 802.15.4e. Der *Beacon* ist ein regulärer IEEE 802.15.4e Frame mit dem Subtyp „00“. Die Nutzlast des *Beacon* enthält die in Tabelle 4.3 auf der nächsten Seite angegebenen Felder.

Tabelle 4.4 auf der nächsten Seite zeigt den Aufbau der Flags.

[6]IEEE 802.15.4e erlaubt auch, das *Group ACK* zu einem späteren Zeitpunkt zu senden, nämlich als eigenen Frame zwischen den Zeitschlitzen der Sensoren und Aktoren, um optional auch gleich Zeitschlitze für Wiederholungen zu reservieren. Darauf wurde verzichtet, da dies wieder den Zusatzaufwand eines Frames bedeutet hätte.

Tabelle 4.3: Nutzlast des Beacon nach Draft IEEE 802.15.4e

Feld	**Länge**	**Beschreibung**
Flags	1 Byte	siehe Tabelle 4.4
Gateway ID	1 Byte	eindeutige ID des Controllers
Sequenznummer der Konfiguration	1 Byte	wird genutzt, um zwischen verschiedenen Konfigurationen zu wechseln.
Größe des *BTS*	1 Byte	Länge eines *BTS*, angegeben als Anzahl von Bytes
Anzahl der *BTS* im Superframe	0-1 Byte	Länge des Superframes in *BTS*
Group ACK	variabel	Bit-Feld, das den Empfang der Sensornachrichten quittiert

Tabelle 4.4: Flags im Beacon nach Draft IEEE 802.15.4e

Feld	**Länge**	**Beschreibung**
Modus	3 Bit	Übertragungsmodus des Netzes
Übertragungsrichtung der Aktoren	1 Bit	Kann die Übertragungsrichtung der Aktoren umkehren
Reserviert	1 Bit	Reserviert für zukünftige Benutzung
Management-Zeitschlitze	3 Bit	Anzahl an *BTS*, die für Management-Zeitschlitze reserviert werden

Das erste Feld der Flags gibt den Übertragungsmodus für das Netz an. Dieser unterscheidet zwischen dem Regelbetrieb („Online Mode“), Erkundungsmodus und Konfigurationsmodus, wie in Abbildung 4.1 auf Seite 92 dargestellt. Erkundungs- und Konfigurationsmodus werden hier nicht weiter betrachtet, sie dienen zum Auffinden neuer Knoten und zum Konfigurieren des Netzes. Für diese Arbeit ist nur der Regelbetrieb relevant.

Die Übertragungsrichtung der Aktoren entscheidet, ob ein Aktor in seinem Zeitschlitz im folgenden *Superframe* ein Datenpaket an den Controller sendet (Bit-Wert ’0’) oder eines von ihm empfängt (Bit-Wert ’1’). Die normale Übertragungsrichtung kann also umgekehrt werden, um z.B. Fehlermeldungen zur Steuerung zu senden. Der Standard sieht nicht vor, dass Sensoren bei einem

Wert '0' dieses Bits in ihrem Zeitschlitz empfangen, die normale Übertragungsrichtung soll beibehalten werden.

Durch die Angabe eines Wertes im letzten Feld der Flags können sogenannte Management-Zeitschlitze reserviert werden. Diese Zeitschlitze folgen direkt nach dem *Beacon*, es werden dabei immer ein Uplink- und ein Downlink-Zeitschlitz reserviert. Sie dienen zur Übertragung von Management-Information, wie z.B. Änderungen von *MAC*-Adressen oder *PHY*-Konfigurationen. Da in diesen Zeitschlitzen Sender bzw. Empfänger nicht implizit erschlossen werden können, müssen hier adressierte Pakettypen eingesetzt werden.

Die Gateway ID im entsprechenden Feld des Beacon identifiziert verschiedene Controller und damit verschiedene Netze. Dies ist relevant, wenn mehrere Netze dieselbe Frequenz und denselben Code benutzen müssen.

Die Sequenznummer der Konfiguration zeigt an, welche Konfiguration gerade benutzt wird. Diese Konfigurationen werden z.B. über Management-Zeitschlitze an die Knoten übermittelt, wobei ihnen eine Sequenznummer zugeordnet wird. Für den aktuellen *Superframe* gilt stets die Konfiguration, deren Sequenznummer im Beacon gesendet wird.

Die Größe eines *BTS* wird als Anzahl von Bytes angegeben. Die tatsächliche (zeitliche) Länge entspricht der Länge eines Daten-Frames mit dieser Anzahl an Bytes als Nutzlast, entsprechend der aktuellen *PHY*-Konfiguration.

Das *Group ACK* ist ein Bit-Feld, dessen Länge der Anzahl der Sensoren im Netz entspricht (auf Byte angeordnet, weshalb eventuell Füll-Bits angefügt werden müssen). Die Reihenfolge der Bits entspricht der Reihenfolge der Zeitschlitze der Sensoren. Für jeden Sensor, von dem im letzten Superframe ein Paket empfangen wurde, wird das entsprechende Bit auf den Wert „1“ gesetzt. Dadurch erhält jeder Sensor Informationen darüber, ob sein zuletzt gesendetes Paket erfolgreich empfangen wurde.

Diese *MAC*-Schicht bietet damit alle notwendigen Merkmale, um meine Anforderungen für das beabsichtigte System zu erfüllen. Ich konnte deshalb die *MAC*-Schicht unverändert übernehmen.

4.2.2.4 Knoten

Jeder Knoten benötigt initial nur die *PHY*-Konfiguration, die auch der Controller benutzt, seine eigene *MAC*-Adresse und die Information, ob er als Sensor oder Aktor fungieren soll.

Für die *PHY*-Konfiguration wird eine Standardkonfiguration vereinbart, auf der eine neue Konfiguration zuerst propagiert wird. Typischerweise benötigen Anwendungen nur eine einzige *PHY*-Konfiguration, die vor Inbetriebnahme des Netzes vereinbart wird. Die dynamische Festlegung einer *PHY*-Konfiguration muss auf der Applikationsschicht erfolgen und ist in dieser Arbeit nicht vorgesehen.

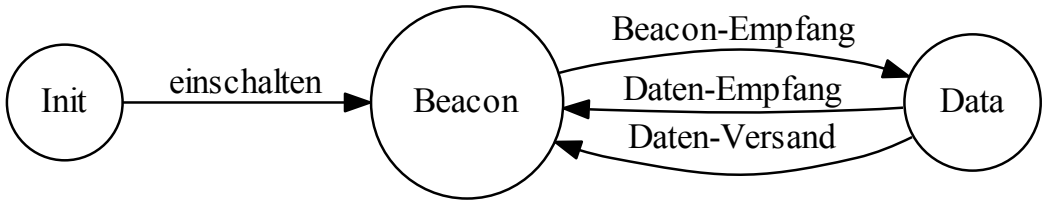


Abbildung 4.6: Zustandsdiagramm eines Knoten

Wie in Abbildung 4.6 gezeigt, wartet jeder Knoten nach dem Einschalten auf einen *Beacon*. Mit den Informationen des *Beacon* kann der Knoten die Länge eines *BTS* berechnen, wann sein Zeitschlitz beginnt und wann der nächste *Beacon* gesendet wird. Ab dann wird der Knoten in seinem Zeitschlitz ein Paket empfangen bzw. senden und dann den nächsten *Beacon* empfangen.

4.2.2.5 Controller

Ein Controller benötigt mehr Vorabinformationen als die Knoten.

- die Anzahl der Knoten im Netzwerk
- die Anzahl der Sensoren
- die Länge des *BTS*
- die eigene Gateway ID (bei mehreren Netzen pro Kanal)

Sobald er diese Informationen erhalten hat, wird der Controller *Beacon*s aussenden. Jeder *Beacon* eröffnet einen *Superframe*. In den Zeitschlitzen der Aktoren sendet der Controller ein Datenpaket mit den aktuellen Steuerbefehlen für den Aktor und in den Zeitschlitzen der Sensoren geht er in den Empfangsmodus.

4.2.2.6 Adressvergabe

Die *MAC*-Adresse weist den Platz des Zeitschlitzes im Superframe zu. Sie wird vor Inbetriebnahme statisch vergeben, z.B. per DIP-Schalter-Feld oder mit einem Programmiergerät, wie es auch bei *AS-Interface* gemacht wird.

Es muss im System ebenso viele *MAC*-Adressen wie Knoten geben. Die *MAC*-Adressen aller Aktoren müssen höher sein als die Anzahl der Sensoren. Auf diese Weise wird erreicht, dass die Zeitschlitze der Sensoren am Anfang des Superframes liegen, vor den Zeitschlitzen der Aktoren, wie es auch in IEEE 802.15.4e festgelegt ist. Dies bringt unter anderem den Vorteil, dass die Sensordaten der Steuerung vorliegen, bevor die Steuerbefehle ausgesendet werden.

Eine automatische Verteilung logischer *MAC*-Adressen zur Zuordnung der Zeitschlitze, die den festgelegten Adressen zugeteilt werden, wäre auch möglich. Dies lag aber außerhalb des Umfangs dieser Dissertation und verbleibt deshalb ein Thema für zukünftige Arbeiten.

4.3 Optimierung einer robusten PHY-Schicht auf UWB-Basis

Als erster Schritt, wurde die *MAC*-Schicht festgelegt, als nächster Schritt wurde eine *PHY*-Schicht modelliert und angepasst.

Um mit der *PHY*-Schicht die Nutzbarkeit der viel versprechenden Eigenschaften von *IR-UWB* für die drahtlose Kommunikation in der industriellen Automatisierungstechnik untersuchen zu können, wählte ich *IR-UWB* als Basis.

Eine wichtige Anforderung der Industrie ist, dass ein System einem Standard entspricht. Standardisierte Protokolle werden stets proprietären Protokollen vorgezogen, da keine Abhängigkeit von einem einzelnen Hersteller entsteht. Deshalb wurde das Protokoll nicht von Grund auf neu entwickelt, sondern aus bestehenden Standards weiterentwickelt. Dies soll auch die Möglichkeit eröffnen, dieses Protokoll wieder in eine zukünftige Erweiterung des Standards zurückzuführen.

4.3.1 Modellierung einer PHY-Schicht für IR-UWB in industriellen Anwendungen

Die *PHY*-Schicht nutzt *IR-UWB*-Signale, wie in Abschnitt 2.2.3.5 auf Seite 38 beschrieben. Dabei können verschiedene Pulsformen verwendet werden. Zentralfrequenz und spektrale Breite des Signals leiten sich aus der Pulsform ab. Die Eigenschaften der Pulsform sind für die Verarbeitung im analogen Teil des Transceivers sehr wichtig, aber von der Verarbeitung im Basisband, also im digitalen Teil des Transceivers, weitgehend unabhängig. Hier werden Aspekte der digitalen Stufe betrachtet, wobei für die Verarbeitung im Basisband die Pulsform als gegeben angesehen wird. Nur die Pulsdauer T_C, und für kohärente Empfänger ebenfalls die Phase des Pulses, sind für die Informationsübertragung relevant.

4.3.1.1 Kohärentes und nichtkohärentes Empfangen

Ein kohärenter Empfänger versucht, das im Kanal vorliegende Signal gegen einen lokal gespeicherten Referenzpuls zu vergleichen, um einen Puls zu detektieren. Dazu ist eine hohe Sample-Rate und pulsgenaue Synchronisation nötig. Die Phasenlage des Pulses lässt sich beim kohärenten Empfang gut ermitteln.

Für den nichtkohärenten Empfang wird Energie-Detektion genutzt. Dabei wird für bestimmte Zeitabschnitte das Energieniveau im physikalischen Kanal beobachtet, um einen Puls bzw. eine Kette von Pulsen (genannt *Burst*) zu detektieren. Dieser Zeitabschnitt muss länger als ein einzelner Puls sein und kann auch länger als mehrere Pulse sein, um eine Pulskette zu erfassen. Die Signal-

verarbeitung ist einfacher als beim kohärenten Empfang, da kein Vergleich auf eine lokale Referenz des Pulses durchgeführt werden muss.

4.3.1.2 Einzelpulse und Pulsfolgen

Werden die Informationen in Form von Pulsketten, sog. *Burst*s, übertragen, dann erleichtert das größere *SNR* einem Energie-Detektor die Unterscheidung, ob in dem Bezugsintervall ein *Burst* übertragen wurde oder nicht. Weiterhin kann das beobachtete *Integrationsintervall* länger sein, da nur der gesamte *Burst* und nicht jeder Puls einzeln empfangen werden muss. Da in Multipfad-Kanälen Komponenten einzelner Pulse später eintreffen können, ermöglicht ein längeres *Integrationsintervall* mehr Energie eines Pulses einzusammeln. Zusätzlich können Multipfad-Komponenten bei Übertragung eines *Burst*s mit späteren Pulsen kollidieren und das Energieniveau im Kanal anheben, wodurch die *SNR* steigt.

4.3.1.3 Modulation

Die Modulation der Daten erfolgt im Basisband in der Zeitdomäne. Es stehen verschiedene Basisband-Modulationstechniken zur Verfügung, wie in Abschnitt 2.2.3.5 auf Seite 38 beschrieben. Da Pulsfolgen eine besonders robuste Übertragung auch bei längeren *Integrationsintervall*en ermöglichen, wie in Abschnitt 4.3.1.2 gezeigt, wurde die *Burst*-Positions-Modulation (*BPM*) als Modulationsform gewählt. Je nach Beschaffenheit des Empfängers ist diese Modulation auch mit *BPSK* kombinierbar.

4.3.1.4 Synchronisation

Für die Erkennung des aktuellen Zeitschlitzes einerseits und der Position eines *Burst* innerhalb des Symbol-Zeitraumes andererseits, ist jeweils eine Genauigkeit der Synchronisation auf den *MAC*-Zeitschlitz bzw. die *Burst*-Länge notwendig. Diese ist abhängig von zwei Faktoren: dem zeitlichen Abstand zum Zeitpunkt der letzten Synchronisation zwischen Sender und Empfänger und dem Abweichen der Schwingungsfrequenz der einzelnen Uhren (genannt *Clock*

Drift). Um die Synchronität auf *PHY*-Paket-Ebene zu erreichen, wird jedem Paket eine zyklische Sequenz vorangestellt, die *Präambel*. Die minimal benötigte Länge der Präambel hängt von unterschiedlichen Faktoren ab. Die wichtigsten sind der Grad der bestehenden Synchronisation und das *AGC*-Setting.

4.3.2 Adaption der UWB-PHY-Schicht von IEEE 802.15.4a

In dieser Arbeit wurde die *UWB-PHY*-Schicht des Standards IEEE 802.15.4a, die in Abschnitt 2.2.3.9 auf Seite 44 vorgestellt wurde, als Referenz verwendet. Mit dieser *PHY*-Schicht lassen sich alle in Abschnitt 4.3.1 auf Seite 106 beschriebenen Aspekte (wie z.B. nichtkohärenter Empfang) realisieren.

Diese *PHY*-Schicht ist für Sensornetze und damit vor allem für niedrigen Stromverbrauch optimiert. Um die *PHY*-Schicht für die Zielapplikation anzupassen, wurden die Parameter der bestehenden Modi analysiert und ein zusätzlicher Modus entworfen, der über den Standard hinausgehende Optimierungen enthält. Diese Optimierungen werden in Abschnitt 4.3.3 vorgestellt.

Im folgenden werden Parameter vorgestellt, die die *PHY*-Schicht beeinflussen. Die Modi des Standards definieren Werte für diese Parameter. Ich habe untersucht, welchen Einfluss die jeweiligen Parameter auf Zeitverhalten und Robustheit haben, um Werte für die Zielapplikation zu finden, die den Anforderungen des Zielsystems genügen.

4.3.2.1 Hopping-Positionen

Die Anzahl der Hopping-Positionen N_{Hops} gibt vor, wie viele mögliche Positionen ein Symbol für das Time Hopping bietet. Damit wird auch implizit der Wertebereich der Pseudonoise-Folge festgelegt. Je höher der Wert von N_{Hops} ist, desto glatter ist die Spektralline und desto geringer die Wahrscheinlichkeit von *Burst*-Kollisionen bei gleichzeitiger Benutzung des Mediums durch mehrere Übertragungen.

Da sich durch höhere N_{Hops} der Symbol-Takt verringert (da die Länge des Symbol-Frames von N_{Hops} abhängig ist), hängt auch die Pulswiederholungsrate *PRR* und damit die verwendbare Energie pro Puls von N_{Hops} ab. Da immer

nur ein *Burst* pro Symbol übertragen wird, nimmt bei höherem Symbol-Takt die Energie pro Zeiteinheit ab, was wiederum die spektrale Leistungsdichte senkt. Weiterhin hängt die mittlere *PRR* des Datenteils

$$mPRR_D = \frac{N_{Hops}}{T_{Sym}} = \frac{N_{Hops}}{N_{CPB} \cdot N_{Hops} \cdot 4 \cdot T_C} = \frac{1}{4 \cdot N_{Hops} \cdot T_C} \tag{4.10}$$

nur von T_C und N_{Hops} ab. Da T_C für das Basisband gegeben ist, kann die mittlere *PRR* der Daten nur durch N_{Hops} gesteuert werden.

In der Zielapplikation sind keine gleichzeitigen Übertragungen vorgesehen. Deshalb ist eine geringere Anzahl möglicher Hopping-Positionen als im Standard vorgegeben ausreichend, was wiederum einen kürzeren Symbol-Frame und damit eine schnellere Übertragung erlaubt. Allerdings muss hierbei für eine gegebene Pulsform und Amplitude das Energiebudget pro Puls betrachtet werden, da dies abhängig von der mittleren *PRR* ist und die Reichweite des Systems bedingt.

4.3.2.2 Burst-Länge

Die *Burst*-Länge wird angegeben durch die Anzahl von Pulsen bzw. Chips und wird deshalb als „Chips pro *Burst* (N_{CPB})“ bezeichnet. Diese *Burst*-Länge beeinflusst direkt die Länge des Symbol-Frames, verhält sich aber neutral zur Energie pro Symbol.

Bei mehreren Pulsen pro *Burst* überlagern sich beim Empfänger die Multipfad-Fragmente mit den nachfolgenden Pulsen und aufgrund eines längeren möglichen *Integrationsintervall*s kann der Empfänger auch mehr Pulse pro Sample aufsammeln. Deshalb führen längere *Burst*s zu höheren Energieansammlungen beim Empfänger, was wiederum die Empfangsleistung erhöht.

Eine geringere Anzahl von Pulsen pro *Burst* erschwert das Empfangen, da das Energieniveau geringer wird. Außerdem muss das *Integrationsintervall* des Empfängers immer größer als die Länge eines *Burst* sein. Deshalb bedingt eine geringere *Burst*-Länge eine höhere Sample-Rate.

Der Wert der Zielapplikation wurde bei 16 Chips per *Burst* belassen, um kompatibel zur Basisrate von IEEE 802.15.4a zu bleiben.

4.3.2.3 Präambel-Wiederholungen

Wie in Abschnitt 2.2.3.9 auf Seite 44 beschrieben, besteht die Präambel eines Paketes nach IEEE 802.15.4a aus einem Synchronisationsteil und einem *SFD*. Der Synchronisationsteil besteht dabei aus mehreren Wiederholungen eines Präambel-Symbols. Die Anzahl der Wiederholungen (N_{Prep}) ist im Standard in vier Abstufungen zwischen 16 und 1024 Wiederholungen definiert.

Mehrere Wiederholungen des Präambel-Symbols erleichtern hierbei die Erkennung, falls das digitale Basisband des Empfängers eine Autokorrelation zur Präambel-Detektion verwendet. Bei Empfängern, die stattdessen eine Kreuzkorrelation verwenden, ermöglichen mehrere Präambel-Wiederholungen eine genauere Einstellung des *AGC* und erhöhen damit die Empfangswahrscheinlichkeit.

Für die Zielapplikation, in der viele kurze Pakete in sehr kurzer Zeit übertragen werden müssen, erweisen sich viele Präambel-Wiederholungen allerdings als ungünstig, da dadurch die Übertragungsdauer eines jeden Paketes erheblich steigt.

4.3.2.4 L-Spreading

Das *L-Spreading* ist die Länge eines Präambel-Sub-Symbols, also eines Code-Symbols des 31 Zeichen langen Codes eines Präambel-Symbols. Ein großes *L-Spreading* erhöht den Abstand der einzelnen Code-Symbole, wodurch die Inter-Symbol-Interferenz innerhalb eines Präambel-Symbols verringert wird. Da eine Erhöhung des *L-Spreading*s aber jede Wiederholung des Präambel-Symbols verlängert, hat das *L-Spreading* eine große Auswirkung auf die Länge der Präambel und damit auf die Länge jedes Pakets.

In den Modi des Standards wird das *L-Spreading* so gesetzt, dass sich die *PRR* der Präambel

$$mPRR_P = \frac{16}{31 \cdot L \cdot T_C} \tag{4.11}$$

und die durchschnittliche *PRR* des Datenteils aus Formel (4.10) entsprechen. In Formel (4.11) kommen auf die Dauer eines Präambel-Symbols 16 Pulse, da 16 der 31 Code-Symbole jedes Präambel-Codes $\neq 0$ sind. Wenn sich die mittlere

PRR der Präambel und des Datenteils entsprechen, dann können Pulse in Datenteil und Präambel auch immer mit derselben Energie versendet werden. Dies ist besonders wichtig für kohärente Empfänger oder Energie-Detektions-Empfänger mit einer hohen Sample-Rate, die die Auflösung einzelner Pulse erlaubt.

Aus einer Gleichsetzung von Formel (4.10) und Formel (4.11) lässt sich L für alle standardkonformen Modi durch

$$L = 2 \cdot N_{Hops} - 1 \tag{4.12}$$

berechnen.

Unabhängig von den standardkonformen Modi lässt sich über L die *PRR* der Präambel steuern und damit die nutzbare Energie pro Puls. Weiterhin kann L entsprechend dem *Delay Spread* gesetzt werden, um die Inter-Symbol-Interferenz in der Präambel zu verringern bzw. zu vermeiden.

4.3.2.5 Start of Frame Delimiter

Der *SFD* ist eine ternäre Folge mit hoher Autokorrelation, die mit dem Präambel-Symbol gespreizt wird. Diese Folge dient als Ende der Präambel und stellt damit den Synchronisationspunkt für die Bit-Synchronisation dar.

Da der *SFD* die Übertragung von acht zusätzlichen Präambel-Symbolen bedingt, wurde untersucht, inwiefern eine Paketübertragung ohne *SFD* möglich ist.

Zur Verringerung der Paketlänge benutzte ich eine verkürzte Präambel ohne *SFD*. Um ein Paket ohne *SFD* übertragen zu können, muss ein anderer eindeutiger Synchronisationspunkt vorhanden sein. Dies ist gegeben, wenn z.B. nur ein einzelnes Präambel-Symbol (ohne Wiederholungen) übertragen wird. Ein Empfänger hat dadurch das Maximum der Korrelation des Präambel-Symbols als Synchronisationspunkt. Besonders bei fehlender oder fehlerhafter Information über einen ungefähren Startpunkt des Paketes bedingt diese Vorgehensweise allerdings einen Schwellwert, der einen Minimalwert für den Korrela-

tionswert darstellt. Dies verhindert eine falsche Synchronisation auf ein dem Präambel-Code ähnliches Muster im Rauschen.

4.3.2.6 Kodierung

Für Energie-Detektion, so wie sie in der Zielapplikation verwendet werden soll, bietet der Standard die Verwendung eines optionalen *Reed-Solomon-Code* an, der in Paragraph 2.2.3.9 auf Seite 46 genauer beschrieben ist. Dieser systematische Blockcode arbeitet auf Symbolen mit einer Länge von 6 Bit. Der Code hängt an jeweils 55 Symbole einen Block von acht Symbolen zur Fehlerkorrektur an. Dies ermöglicht eine Korrektur von bis zu vier fehlerhaften Symbolen. Bei kürzeren Paketen wird bei der Kodierung und Dekodierung die Nutzlast mit Füll-Bits bis zur entsprechenden Größe aufgefüllt. Bei kleinen Nutzlasten sichern die meisten Symbole zur Fehlerkorrektur also nur die Füll-Bits ab.

Da in der Zielapplikation nur sehr kurze Pakete mit einer Nutzlast von wenigen Bytes versendet werden, ist dieser Code nicht sehr gut geeignet. Die Größe eines Paketes mit diesem Code lässt sich als

$$N_{RS} = N_{Nutzdaten} + 48 \cdot ceil(\frac{N_{Nutzdaten}}{330}) \tag{4.13}$$

darstellen, wobei *ceil* das aufrunden bezeichnet.

Als Alternative bietet sich ein Faltungs-Code oder auch Turbo-Code an, speziell im Hinblick auf die in Abschnitt 4.3.3.3 auf Seite 124 vorgestellte Soft-Bit-Empfangstechnologie. Turbo-Codes sind zwar sehr effizient, benötigen aber relativ komplexe Encoder und Decoder. Deshalb schlage ich als alternativen Code die Benutzung eines industrieüblichen Faltungs-Codes der Rate $\frac{1}{2}$ vor, für den es kostengünstige Viterbi-Decoder als „*off the Shelf*“ Hardware gibt. Wird dieser Faltungs-Code statt der Standardkodierung verwendet, so erhält man eine Paketlänge von

$$N_{Conv} = N_{Nutzdaten} \cdot \frac{1}{R} \tag{4.14}$$

wobei R die Rate des Faltungs-Kodierers ist. Um zu ermitteln, bis zu welcher Nutzlastgröße (in Bit) $N_{Nutzdaten}$ ein Paket mit einem Faltungs-Code der Rate

$R = \frac{1}{2}$ kleiner ist als ein Reed-Solomon-kodiertes Paket mit einer gleich langen Nutzlast, kann man die Paketlängen gleichsetzen:

$$\begin{aligned} N_{RS} &= N_{Conv} \\ N_{Nutzdaten} + 48 &= N_{Nutzdaten} \cdot 2 \\ N_{Nutzdaten} &= 48 \end{aligned} \tag{4.15}$$

Zum Vergleich der Absicherung muss die Korrekturfähigkeit ermittelt werden. Nimmt man als Faltungs-Code den in IEEE 802.11 verwendeten (133,171)-Faltungs-Code mit der Gedächtnislänge 7 an, so erhält man nach [60, S.122] eine freie Distanz d von 10 Bit, was einer Korrekturfähigkeit von

$$\frac{d-1}{2} = 4,5 \approx 5 \tag{4.16}$$

Bit entspricht. Der (63,55)-*Reed-Solomon-Code* besitzt eine Korrekturfähigkeit von vier Symbolen auf ein Codewort von 55 Symbolen. Allerdings ist ein Codewort für kurze Paketlängen nur mit Füll-Bits belegt.

Die jeweilige Absicherungsfähigkeit wurde simulativ verglichen, die Ergebnisse sind in Abschnitt 4.3.3.3 auf Seite 124 ausführlicher dargestellt.

Aus den Simulationen ergab sich, dass für Pakete mit $N_{Nutzdaten} \leq 48$ Bit ein Faltungs-Code vergleichbare Absicherung bei kürzeren Paketlängen bietet. Da in der Industrieautomatisierung Paketgrößen von wenigen Bits bis hin zu zwei Byte üblich sind [57], ist der Faltungs-Code dafür offensichtlich besser geeignet.

Zusätzlich profitiert der Faltungs-Code auch noch von weiteren Optimierungen, in Abschnitt 4.3.3.3 auf Seite 124 und 4.4.2 auf Seite 141 beschrieben, die mit dem *Reed-Solomon-Code* nicht möglich sind.

Allerdings ist die Dekodierung des Faltungs-Codes durch einen Viterbi-Decoder aufwändiger als die Dekodierung des *Reed-Solomon-Code*. Dafür ergeben sich bei der Benutzung des *Reed-Solomon-Code* weitere Herausforderungen, da die Daten in 6-Bit-Symbolen angeliefert werden müssen, was bei der Hardware-Implementierung zu erhöhtem Speicherbedarf und zusätzlichen Konvertierungsschritten führt.

4.3.2.7 Zusammenfassung

Abschließend lässt sich sagen, dass sich durch Parameterwerte,von den im Standard beschriebenen Modi abweichen, die Latenzzeit verkürzen lässt.

4.3.3 Maßnahmen zur Erhöhung der Robustheit der PHY-Schicht

Die Anwendung in der industriellen Automatisierung bedingt zusätzlich zu kurzen Latenzen auch eine hohe Robustheit. In diesem Abschnitt werden deshalb Optimierungen an der *PHY*-Schicht vorgenommen, die die Robustheit der Übertragungen erhöhen.

4.3.3.1 Mehrpulsige Präambel-Modulation

Bei genauer Simulation der *UWB-PHY*-Schicht von IEEE 802.15.4a stellte sich heraus, dass die Modulation der Präambel erheblich anfälliger gegenüber Rauschen und Multipfad-Effekten ist als die Datenmodulation.

Dies führt zu einem hohen Anteil von Paketverlusten durch misslungene oder fehlerhafte Erkennung der Präambel. Abbildung 4.7 auf der nächsten Seite zeigt eine Aufschlüsselung von 1000 simulierten Paketübertragungen über einen sehr stark eingeschränkten Kanal mit einer *SNR* von 1 dB und dem industriellen *NLOS*-Kanalmodell (Kanalmodell Nummer 8). Es ist erkennbar, dass der Großteil der Paketverluste durch fehlerhafte Synchronisation entsteht, in ein paar Fällen ist die Synchronisation sogar gänzlich fehlgeschlagen. Weit weniger Pakete gingen durch Demodulations-Fehler im Datenteil verloren, obwohl kein *FEC*-Code verwendet wurde. Um die Paketverluste aufgrund von Präambel-Fehlern zu reduzieren, habe ich Ursachen für diese Fehler untersucht.

Das Problem lässt sich anhand der *ADC*-Werte eines Paketes verdeutlichen.

Wie Abbildung 4.8 auf Seite 116 zeigt, wird in der Präambel erheblich weniger Energie als im Datenteil gesendet und auch empfangen. Dies liegt daran, dass die Präambel aus Einzelpulsen mit großem Abstand moduliert wird (hier

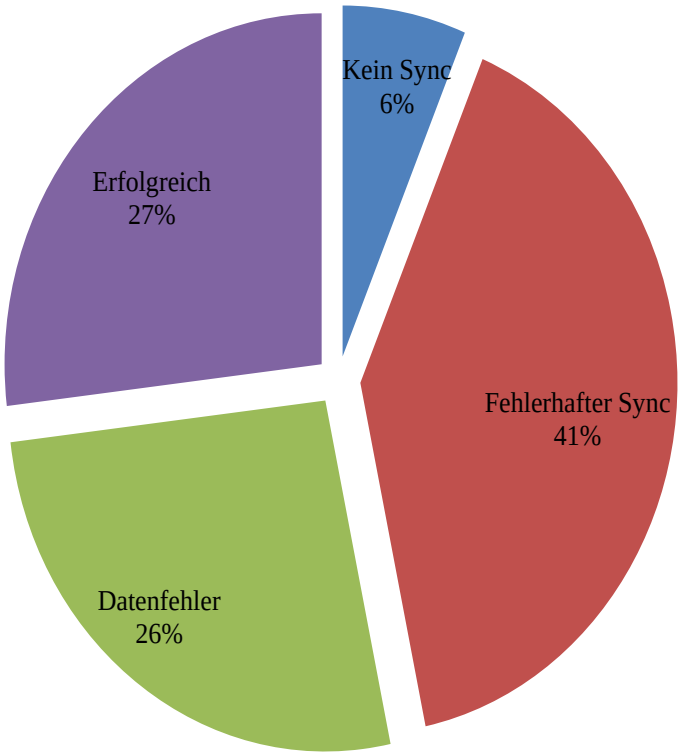


Abbildung 4.7: Anteil der erfolgreich übermittelten und der verlorenen Pakete, aufgeteilt nach dem Grund des Paketverlustes

Präambel-Sub-Symbol genannt), während der Datenteil mit *Burst*s aus mehreren Pulsen übertragen wird. Da *niedrigratige Energie-Detektion* als Empfänger-Architektur verwendet wird, wobei ein Sample länger als ein Puls ist, ist die Energie, die in einem Sample der Präambel aufgenommen werden kann, geringer als die in einem Sample im Datenteil. Dies liegt daran, dass dort nur ein einzelner Puls in demselben Zeitraum aufgenommen wird, in dem im Datenteil mehrere Pulse aufgenommen werden.

Dies ermöglicht eine höhere Energie-Ausbeute, wenn *niedrigratige Energie-Detektion* verwendet wird. Die Anzahl der Pulse pro Präambel-Sub-Symbol wird mit dem Parameter N_{Pburst} bezeichnet.

Abbildung 4.9 auf Seite 117 zeigt zum Vergleich die ADC-Werte eines Paketes mit $N_{Pburst} = 4$. Es ist erkennbar, dass die ADC-Werte der mehrpulsigen Präambel in Abbildung 4.9 erheblich höher sind als bei der Präambel mit

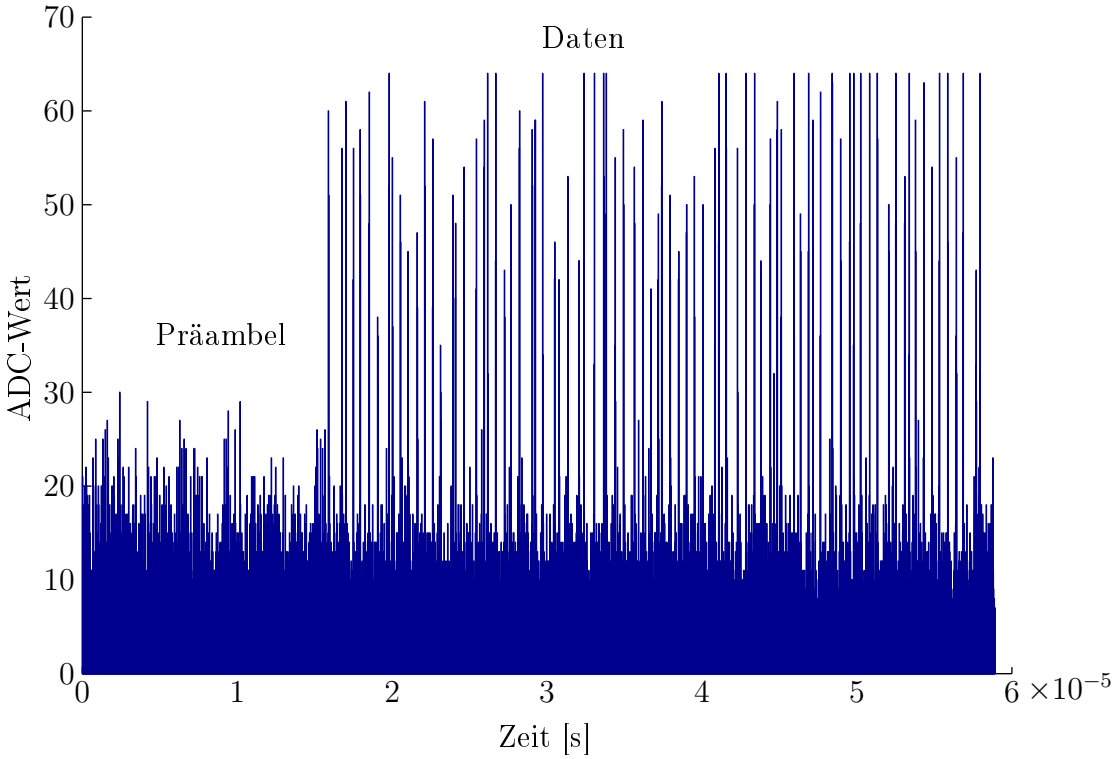


Abbildung 4.8: Gegenüberstellung der Signalstärke der Präambel und des Datenteils bei einer Präambel mit Einzelpulsen

Einzelpulsen in Abbildung 4.8. Da der Rauschpegel in beiden Fällen gleich bleibt[7], wird auf diese Weise die *SNR* in der Prämbel erhöht, was den Empfang robuster macht bzw. die Reichweite erhöht.

Ein Punkt, der dabei betrachtet werden muss, ist das *L-Spreading*, also die Länge eines Präambel-Sub-Symbols und damit die Pause zwischen den Einzelpulsen bzw. *Burst*s der Präambel. Dieses hat Auswirkungen auf die mittlere *PRR*, und damit auf das verwendbare Energiebudget. Es gibt zwei Möglichkeiten, wie das *L-Spreading* für mehrpulsige Präambeln angepasst werden kann.

Eine Möglichkeit dieser Anpassung ist, das *L-Spreading* im gleichen Ausmaß wie N_{Pburst} zu verlängern. Wie Abbildung 4.10 auf Seite 118 beispielhaft zeigt, würde bei einem regulären *L-Spreading* von 15 in Skizze (1) bei $N_{Pburst} = 4$ das *L-Spreading* auf $4 \cdot 15$ in Skizze (2) verlängert werden.

[7] Die erkennbare Änderung in den *ADC*-Werten des Datenteils wird durch die angepasste Verstärkung bedingt

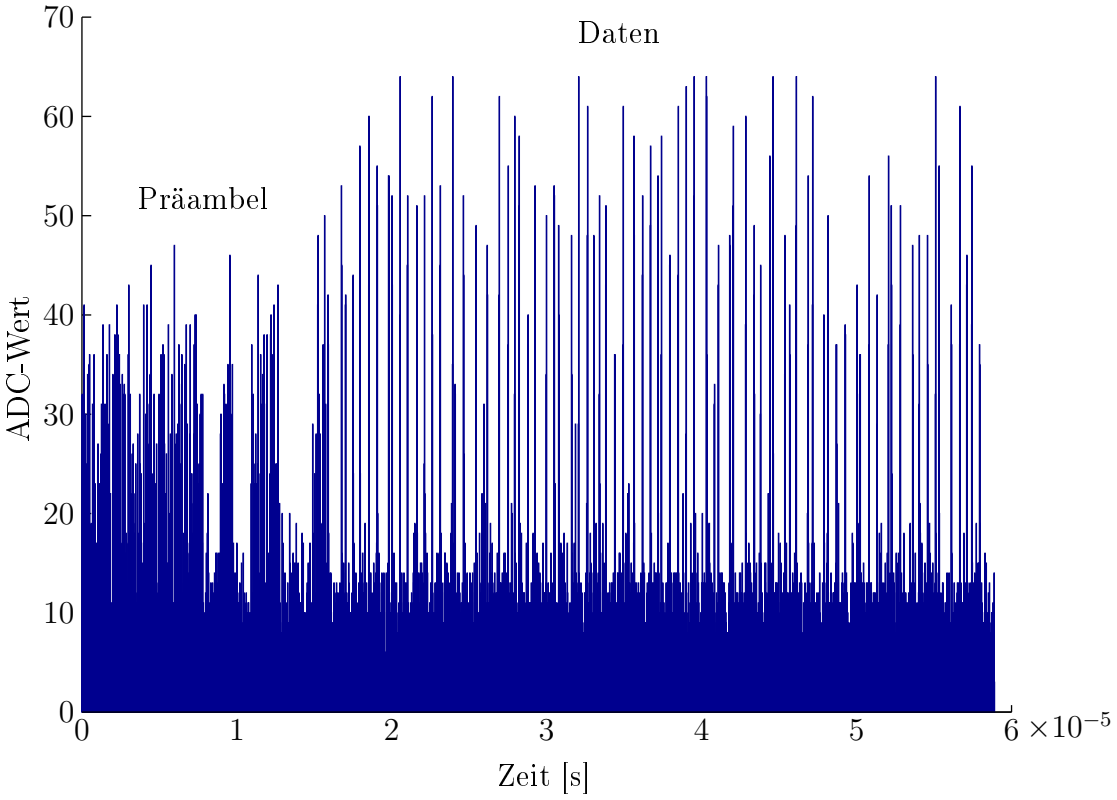


Abbildung 4.9: Gegenüberstellung der Signalstärke der Präambel und des Datenteils bei einer Präambel mit Bursts aus 4 Pulsen

Dies ermöglicht eine Beibehaltung derselben mittleren *PRR*, verlängert aber die Präambel um den Faktor N_{Pburst}. Da die Präambel jedoch aufgrund der höheren *SNR* besser erkennbar ist, reichen auch weniger Wiederholungen des Präambel-Symbols aus, um ein Paket empfangen zu können. Dies gilt insbesondere, da auch das verlängerte *L-Spreading* die Robustheit gegen Multipfad-Effekte erhöht. Deshalb kann die Anzahl der gesendeten Präambel-Symbole reduziert werden.

Alternativ hierzu kann die Länge eines Präambel-Sub-Symbols gleich bleiben, wozu das *L-Spreading* aber um $N_{Pburst}-1$ reduziert werden muss. Zum Beispiel würde aus einem regulären *L-Spreading* von 15 bei $N_{Pburst} = 4$ ein *L-Spreading* von $15-(4-1) = 12$ resultieren. Somit bleibt die Länge der Präambel gleich, aber es erhöht sich die Anzahl der Pulse innerhalb der Präambel und damit die mittlere *PRR*.

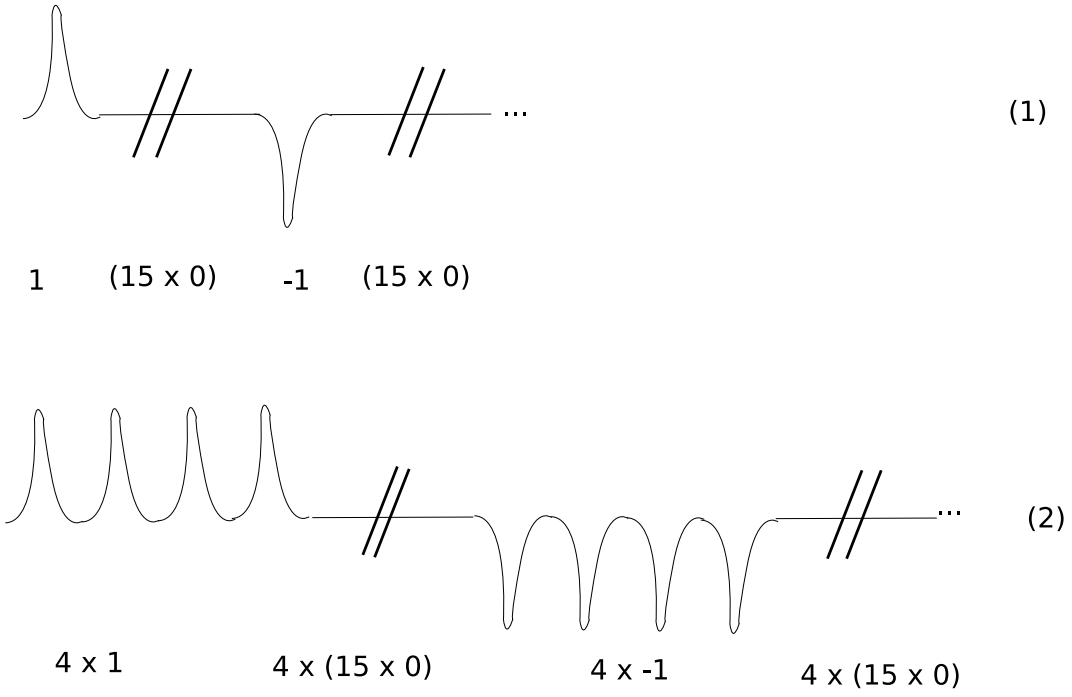


Abbildung 4.10: Verlängertes L-Spreading bei Präambel-Bursts

Der optimale Wert für N_{Pburst} und welche der beiden Methoden zur Bestimmung des *L-Spreading* besser geeignet ist, hängt stark von den zeitlichen Anforderungen für ein Paket in der Zielapplikation ab. Zusätzlich haben die vorherrschenden Kanalbedingungen einen starken Einfluss auf die optimalen Werte. So ist z.B. bei einem sehr langem *Delay Spread* ein höheres *L-Spreading* zu bevorzugen. Außerdem kann für Applikationen mit vorhersehbarem (deterministischem) Funkverkehr a priori ein Energiebudget berechnet werden, um die optimalen Werte zur Konfiguration des Netzes zu finden.

Durch Simulation wurde der Einfluss von mehreren Pulsen in der Präambel auf die durch fehlerhafte Präambel-Erkennung erzeugte Paketfehlerrate untersucht. Dazu wurde der Simulator um die Einstellmöglichkeit der Pulse pro Präambel-Sub-Symbol erweitert. Der Empfänger blieb dabei unverändert, da die Samples für ein Präambel-Sub-Symbol aufaddiert werden.

Wie Abbildung 4.11 auf der nächsten Seite für $N_{Pburst} \in 1, 2, 4$ zeigt, verringert sich die Paketfehlerrate durch nicht erkannte Präambeln mit steigendem N_{Pburst}. Die rot eingezeichnete Linie markiert die Mindestanforderung von

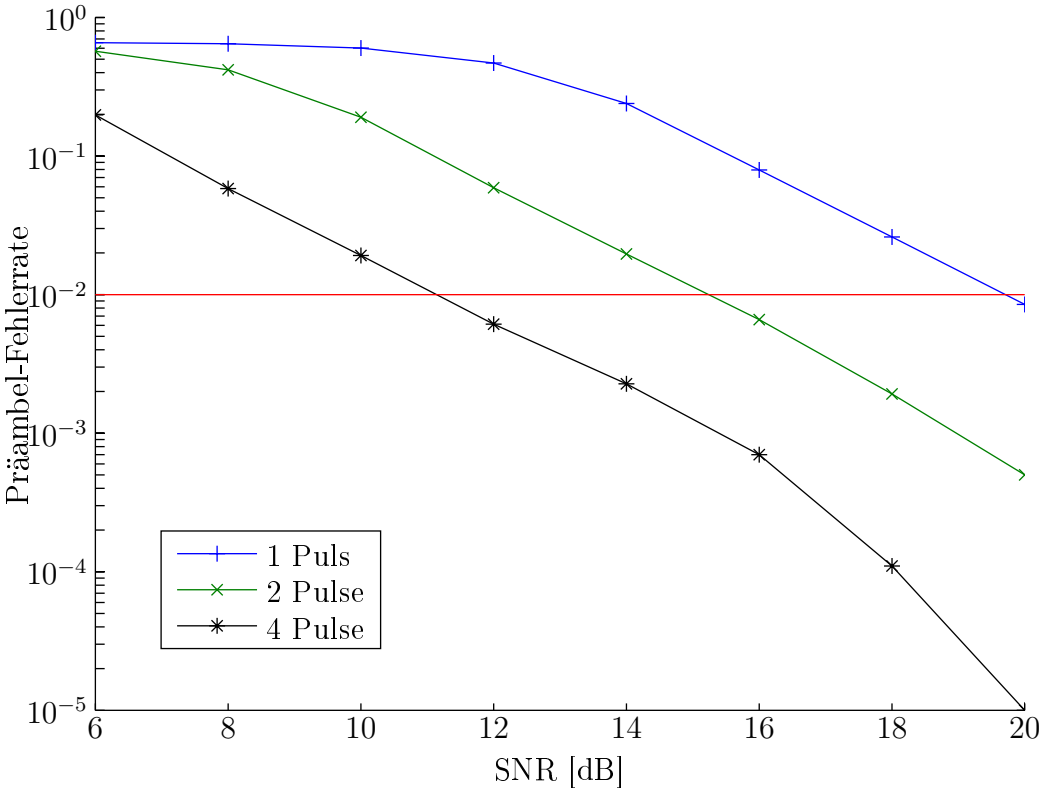


Abbildung 4.11: Veränderung der Rate nicht erkannter Präambeln durch mehrpulsige Präambeln

10^{-2}. Diese wird mit $N_{Pburst} = 4$ bei einer 8 dB tieferen SNR erreicht als bei der Referenz mit $N_{Pburst} = 1$. Diese Ergebnisse habe ich auch in einem Konferenzbeitrag auf der Konferenz ICUWB 2010 [HOSK10] veröffentlicht und die Technologie wurde zum Patent [P3] angemeldet.

4.3.3.2 Überlappende Addition der Samples

In einem regulären Empfänger werden zur Demodulation alle Samples, die zu einer Hopping-Position gehören, aufaddiert. Das bedeutet, wenn das *Integrationsintervall* des Empfängers die halb so lang wie eine Hopping-Position ist, werden jeweils zwei Samples aufaddiert. Gleichzeitig wird die entsprechende Hopping-Position für das Symbol ermittelt.

Nach der Addition werden, die aufaddierten Samples, die der relevanten Hopping-Position in der ersten (Z_i) und zweiten (O_i) Hälfte des Datensymbols entspre-

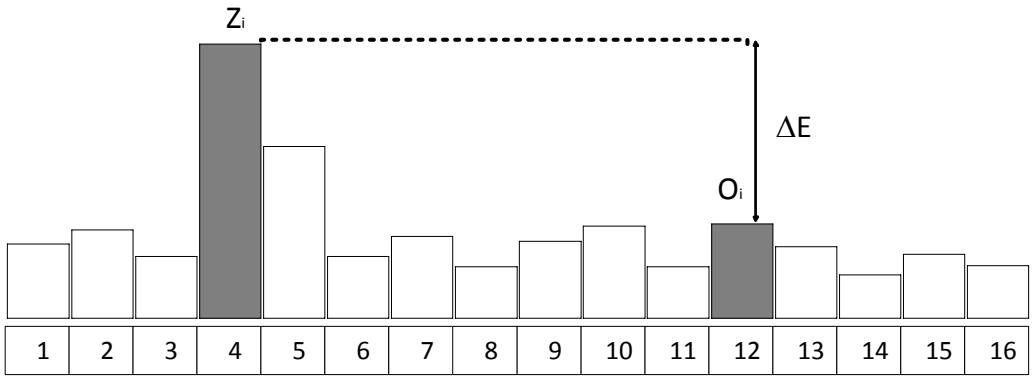


Abbildung 4.12: Bit-Ermittlung im Datenteil

chen, verglichen. Abbildung 4.12 zeigt Beispiel mit vier möglichen Hopping-Positionen, von denen für das aktuelle Symbol die Nummer 4 relevant ist.

Formell berechnen sich die Werte durch:

$$Z_i(t) = \sum_{n=1}^{\frac{T_{sym}}{T_s}} (t - h_i \cdot T_s \cdot N_{spb} - n \cdot T_s) \tag{4.17}$$

$$O_i(t) = \sum_{n=1}^{\frac{T_{sym}}{T_s}} (t - \frac{T_{sym}}{2} - h_i \cdot T_s \cdot N_{spb} - n \cdot T_s) \tag{4.18}$$

wobei T_s die Länge eines Samples ist, T_{sym} die Länge des Symbols, N_{spb} die Anzahl der Samples pro Burst und h_i die Hopping-Position des i-ten Symbols. Das Vorzeichen der Differenz

$$\Delta E_i = O_i(t) - Z_i(t) \; \forall \, t = i \cdot T_s \tag{4.19}$$

entscheidet über den ermittelten Bit-Wert, ein negatives ΔE ergibt den Wert '0', ein positives ΔE ergibt den Wert '1'. Der Betrag von ΔE drückt die Sicherheit der Entscheidung aus, Abschnitt 4.3.3.3 auf Seite 124 beschreibt eine Soft-Bit-Technik, die diese Information verwertet.

Bei ungenauer bzw. fehlerhafter Synchronisation, *Clock Drift* oder Veränderungen in der Kanalimpulsantwort während eines Pakets kann es vorkommen, dass der Hauptteil der Energie nicht in den Samples, die direkt der Hopping-Position

zugeordnet sind, liegt, sondern in einem daran angrenzenden Sample. Dadurch wird in den aufaddierten Samples der betreffenden Hopping-Positionen weniger Energie aufgesammelt, was die *SNR* verringert und Bit-Fehler begünstigt.

Als Gegenmaßnahme können zu jeder Hopping-Position die angrenzenden Samples addiert werden. Da durch die größere Anzahl an addierten Samples auch mehr Rauschen aufgesammelt wird, können die angrenzenden Samples jeweils mit Gewichtsfaktoren belegt werden, die den Einfluss auf die Hopping-Position mindern.

An obigem Beispiel angewandt, werden die angrenzenden Samples je vor und nach den Hopping-Positionen 4 und 12 jeweils mit einem Faktor multipliziert und zu den entsprechenden Hopping-Positionen hinzugezählt.

Es wird also ein Vektor $\bar{w}$ festgelegt, der die Gewichtung der angrenzenden Samples festlegt. Dieser Vektor richtet sich nach der zu erwartenden Verschiebung des ursprünglich gesendeten *Burst*s. Für jede Hopping-Position werden dann die direkten Samples aufsummiert und zusätzlich die angrenzenden Samples addiert, gewichtet mit dem anfangs festgelegten Vektor. Die so errechneten Werte für jede Hopping-Position dienen anschließend zur Bestimmung des Bit-Wertes, wie oben beschrieben.

In Implementierungen sollten für diese Gewichte ganzzahlige Vielfache von Zwei verwendet werden, da sich damit die Multiplikationen sehr effizient als Bit-Shift-Operationen realisieren lassen.

Das folgende Beispiel verdeutlicht das Verfahren anhand von Abbildung 4.12 auf der vorherigen Seite:

Es wird der in Abbildung 4.12 dargestellte Symbol-Frame zugrunde gelegt. Eine Hopping-Position besteht hier aus zwei Samples, die wiederum eine Länge von acht Pulsen haben.

Nach der bisher üblichen Methode werden die 32 Samples des Symbol-Frames jeweils in Zweiergruppen zu den möglichen Hopping-Positionen aufsummiert

$$\overrightarrow{HP} = \{Sample_{2n-1} + Sample_{2n} \, \forall \, n = 1, 2, 3, ..., 16\} \tag{4.20}$$

Mit dem hier vorgestellten Verfahren werden zusätzlich die jeweils angrenzenden Samples, multipliziert mit den jeweiligen Gewichten, addiert.

$$\overrightarrow{HP} = \{Sample_{2n-1} + Sample_{2n} + \bar{w}_1 \cdot Sample_{2n-2} + \bar{w}_2 \cdot Sample_{2n+1} \\ \forall\, n = 1, 2, 3, ..., 16\} \tag{4.21}$$

Im Beispiel wird die vierte der vier möglichen Hopping-Positionen verwendet. Der Bit-Wert wird über das Vorzeichen von ΔE ermittelt, in dem die Werte der Hopping-Position 12 und 4 voneinander subtrahiert werden.

$$\Delta E = \overrightarrow{HP}_{12} - \overrightarrow{HP}_4 \tag{4.22}$$

Um den Effekt anhand von konkreten Zahlen zeigen zu können, werden folgende Annahmen getroffen:

- Zur Vereinfachung wird angenommen, dass der Rauschpegel für die Dauer eines Symbol-Frames gleich bleibt. Diese Annahme lässt sich mit der kurzen Dauer eines Symbol-Frames rechtfertigen, die im verpflichtenden Standardmodus ca. 1 μs dauert.

- Die Werte der Samples werden wie folgt festgelegt:
  Für jeden Puls innerhalb des Samples wird der Wert um 1,0 erhöht, für jede leere Pulsposition um 0,1. Ein Sample von acht Pulsen hätte demnach den Wert 8, ein Sample ohne Pulse den Wert 0,8.

- Zur Verdeutlichung des Unterschiedes wird für diesen Symbol-Frame eine Verschiebung der Synchronisation um 4 Pulspositionen nach hinten angenommen. D.h. die Inhalte der Samples sind zeitlich um 4 Pulse verschoben.

  Die Samples 7 und 9 haben demnach den Wert 4,4, das Sample 8 den Wert 8. Alle anderen Samples haben den Wert 0,8.

- Der Vektor zur Gewichtung der angrenzenden Samples wird mit $\overrightarrow{w} = \{1, 1\}$ festgelegt.

Dadurch ergibt sich für die Bit-Entscheidung ohne die vorgestellte Verbesserung

$$\begin{aligned}\Delta E &= \overrightarrow{HP}_{12} - \overrightarrow{HP}_4 \\ &= Sample_{23} + Sample_{24} - Sample_7 - Sample_8 \\ &= 0,8 + 0,8 - 8 - 4,4 = -10,8\end{aligned} \tag{4.23}$$

Und für die Bit-Entscheidung mit der vorgestellten Verbesserung

$$\begin{aligned}\Delta E &= \overrightarrow{HP}_{12} - \overrightarrow{HP}_4 \\ &= Sample_{22} + Sample_{23} + Sample_{24} + Sample_{25} \\ &- Sample_6 - Sample_7 - Sample_8 - Sample_9 \\ &= 0,8 + 0,8 + 0,8 + 0,8 - 0,8 - 4,4 - 8 - 4,4 \\ &= 3,2 - 17,6 = -14,4\end{aligned} \tag{4.24}$$

Der Absolutwert von ΔE ist ein Indikator für die Verlässlichkeit der Bit-Entscheidung. Schon in diesem Beispiel mit relativ idealen Bedingungen ist der Betrag für die überlappende Addition höher. Dies bedeutet eine geringere Wahrscheinlichkeit der Fehlentscheidung bei schlechteren Kanalbedingungen (niedrigeres *SNR*, Multipfad-Effekte, nichtlineare Störungen).

Dieses Verfahren wurde simulativ verifiziert. Dazu wurde als zusätzliche Einstellmöglichkeit anstelle der statischen Addition der Samples ein Gewichtungs-Vektor overlap verwendet, der einerseits die Angabe von Samples pro Hopping-Position erlaubt und andererseits auch definiert, wie viele Samples vor und nach der eigentlichen Hopping-Position mit welcher Gewichtung addiert werden.

Abbildung 4.13 zeigt, dass die Bit-Fehlerrate bei sinkender *SNR* mit überlappender Addition langsamer steigt als mit dem herkömmlichen Verfahren. Dazu wurden die Brutto-Bit-Fehler, mit und ohne überlappender Addition, in 300 Bit langen Paketen ohne *FEC*-Code gemessen. Die Synchronisation wurde mit einem künstlichen Jitter von ±3 Samples auf den Synchronisationspunkt gestört. Es wurde das industrielle *LOS*-Kanalmodell verwendet, als Gewichtungsfaktor der überlappenden Addition wurde jeweils $\frac{1}{4}$ gesetzt. Die rote Linie

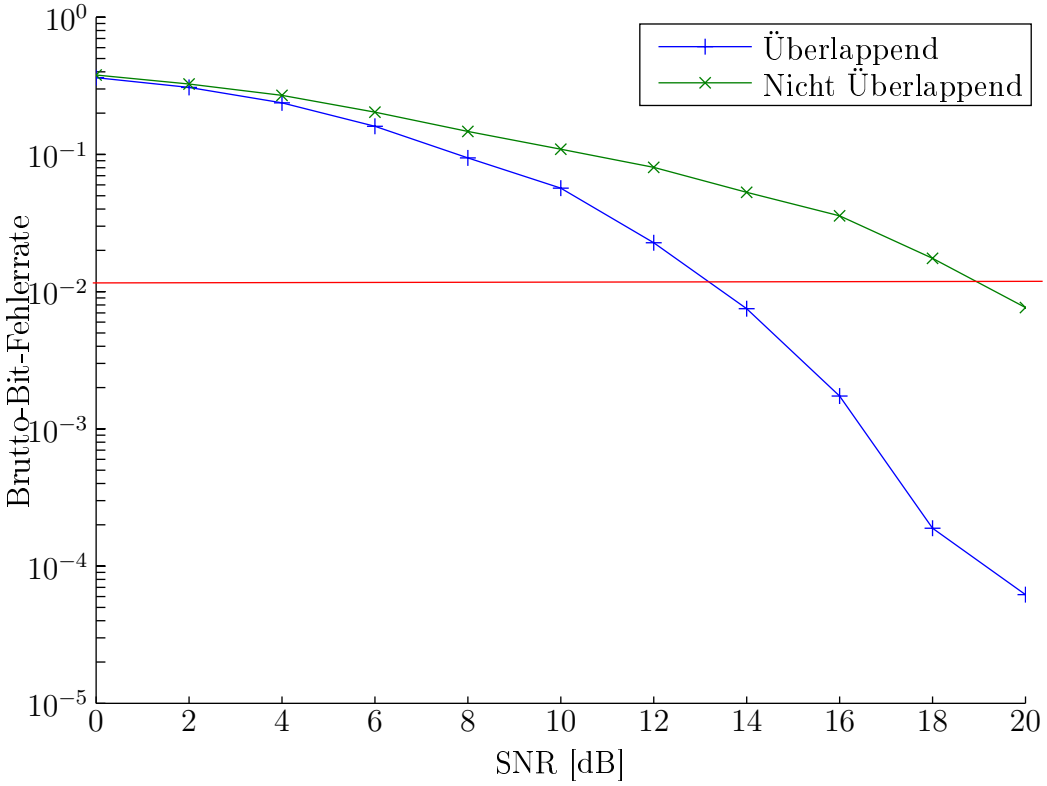


Abbildung 4.13: Vergleich der Brutto-Bit-Fehlerrate bei fehlerhafter Synchronisation mit und ohne überlappendem Sampling

markiert die Brutto-Bit-Fehlerrate von 10^{-2}, welche als Minimalanforderung zu verstehen ist.

Diese Ergebnisse veröffentlichte ich auch erfolgreich auf der internationalen Konferenz für *UWB* (ICUWB) 2010 [HOSK10]. Besonders effektiv erweist sich das vorgestellte Verfahren zusammen mit dem Soft-Bit-Verfahren, welches im nächsten Abschnitt vorgestellt wird.

4.3.3.3 Soft-Bit-Informationen

Wie im vorherigen Abschnitt angedeutet, gibt das Vorzeichen der Differenz (ΔE) zwischen den Energieniveaus der zutreffenden Hopping-Positionen des Symbols die Entscheidung für einen Bit-Wert vor. Weiterhin kann man den

Absolutwert als Verlässlichkeit der Bit-Entscheidung interpretieren: je geringer der Wert, desto unsicherer die Entscheidung für einen Bit-Wert.

Normalerweise wird diese Information nicht verwertet. Ich zeige hier jedoch ein Verfahren, diese Information verwendet, um *Soft-Bit*-Werte zu erzeugen. *Soft-Bit*-Arithmetik ist eine Anwendung der „Fuzzy Logic“ in der Signalverarbeitung, die ursprünglich in Viterbi-Decodern eingesetzt wurde. Dabei wird jedes Bit einerseits durch den seinen Wert, andererseits durch die Verlässlichkeit dieses Wertes charakterisiert. Für genauere Erklärungen sei auf [37, 11] verwiesen.

In dieser Anwendung wird jeder Bit-Wert durch eine reelle Zahl zwischen -1,0 und +1,0 dargestellt. Eine absolut sichere „0“ wird durch -1 dargestellt, eine absolut sichere „1“ durch 1. Alle Werte dazwischen bilden Abstufungen der Sicherheit der Bit-Entscheidung. Um einen konkreten Bit-Wert wie bei einer harten Entscheidung zu erhalten, wird betrachtet, ob der Wert größer als 0 ist.

Um Soft-Bit-Informationen im Empfänger zu erzeugen, muss der Wert aller einzelnen ΔE normalisiert werden. Dazu muss entweder das Maximum von ΔE über ein Paket benutzt werden oder ein Maximalwert aus der Anzahl der aufaddierten Samples und dem Maximalwert des verwendeten *ADC* berechnet werden.

Der Vorteil von Soft-Bits liegt vor allem in deren Verarbeitung in einem *FEC*-Decoder. Diese profitieren von den Informationen über die Verlässlichkeit der Bit-Werte bei der Entscheidung über ein Codewort. Besonders Turbo-Codes erreichen mit Soft-Bit-Daten erheblich bessere Ergebnisse [67], aber auch Viterbi-Decoder nutzen die zusätzlichen Informationen. Soft-Bit-fähige Turbo- und Viterbi-Decoder sind als „Off-the-Shelf“-Komponenten verfügbar. Für *Reed-Solomon-Code*s ist Soft-Bit-unterstützte Dekodierung noch ein aktives Forschungsthema [72, 45]. Deshalb müssen beim Vergleich von *Reed-Solomon-Code*s gegen Soft-Bit-unterstütze *FEC*-Codes direkte (harte) Bit-Entscheidungen für den *Reed-Solomon-Code* verwendet werden.

Wie schon in Abschnitt 4.3.2.6 auf Seite 112 beschrieben, eignet sich für die kurzen Pakete in der Fertigungsautomatisierung ein Faltungs-Code besser als

der im Standard beschriebene *Reed-Solomon-Code*, besonders wenn Soft-Bit-Informationen für die Dekodierung ausgewertet werden können.

In einem simulativen Vergleich wurde die Leistung der Codes betrachtet. Dazu wurde im Simulator die Demodulation um Soft-Bit-Generierung erweitert. Für direkte („harte“) Bit-Entscheidungen wird nur das Vorzeichen verwendet, für die Soft-Bit-Variante wurde ein Viterbi-Decoder mit Soft-Bit-Eingang instanziert. Dadurch wurde die Paketfehlerrate von Paketen von einem Byte Länge verglichen, einmal mit harter Bit-Entscheidung und *Reed-Solomon-Code* und einmal mit Faltungs-Code und Soft-Bit-Erkennung. Um den Effekt zu isolierern wurde eine perfekte Synchronisation angenommen.

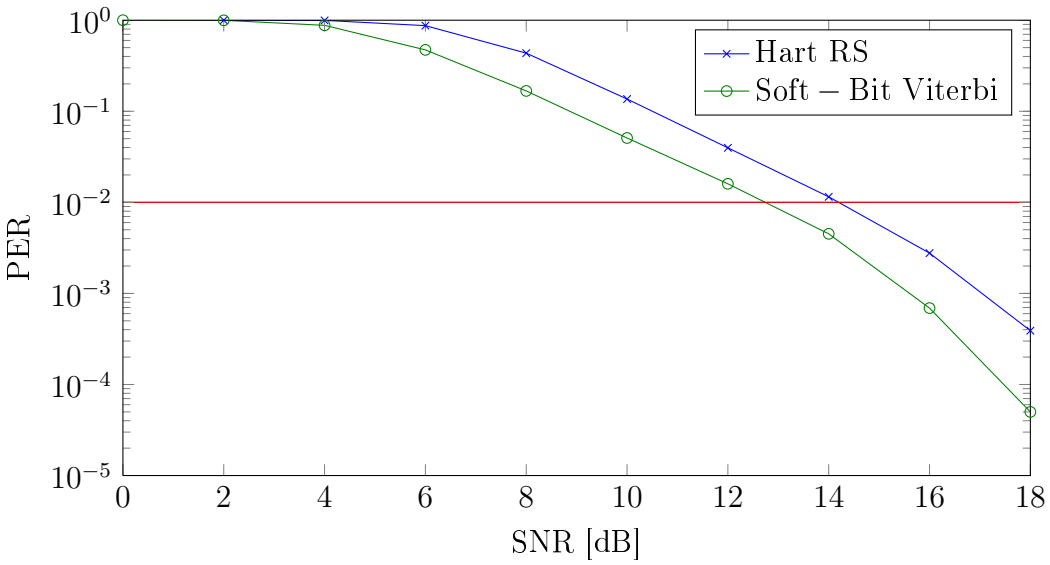


Abbildung 4.14: Vergleich zwischen harter Entscheidung mit RS-Code und Soft-Bit-basiertem Faltungs-Code

Abbildung 4.14 zeigt den Verlauf der Paketfehlerrate für die zwei oben vorgestellten Methoden. Es ist erkennbar, dass die Fehlerrate für den Soft-Bit-basierten Faltungs-Code geringer ist. Der Abstand zwischen den beiden Fehlerraten beträgt ca. 2 dB, wie die rote Linie für die maximal akzeptable Fehlerrate von 10^{-2} zeigt.

Diese Ergebnisse wurden in einem Paper auf der Konferenz ICUWB 2011 veröffentlicht [HKS11].

4.3.3.4 Vorzeichenbehaftete Energiedetektion

Bei einem Energie-Detektions-Empfänger mit relativ langem *Integrationsintervall* wird üblicherweise ein Mischer verwendet, der den Absolutwert des Eingangssignals bildet. Meist wird auch eine Komponente verwendet, die den Eingangswert quadriert, wie z.B. eine Gilbert-Zelle.

Dies bewirkt, dass auch Pulse mit negativer Phase positive Werte liefern. Dadurch löschen sich Pulse mit entgegengesetzten Vorzeichen in der Integration nicht aus. Besonders beim Empfang des Datenteils ist dies wichtig, weil sich hier durch das *Burst Scrambling* die Pulse eines *Burst* im Vorzeichen unterscheiden, wodurch der *Burst* ohne diese Komponente in der Integration größtenteils ausgelöscht werden würde. Durch das Quadrieren werden höhere Werte außerdem überproportional verstärkt, wodurch sich Samples mit Pulsen auch noch stärker vom Rauschen unterscheiden.

Ein negativer Effekt ist allerdings, dass in der Präambel die Vorzeicheninformationen des ternären Präambel-Codes verloren gehen. Dies hat zur Folge, dass der Präambel-Empfangsteil auf den binären Anteil des Codes (also den Betrag) ausgerichtet werden muss, wodurch sich schlechtere Autokorrelationseigenschaften ergeben.

Abbildung 4.16 auf der nächsten Seite zeigt die Autokorrelationsfunktion von fünf Präambel-Symbolen mit ternärem Code. Im Vergleich dazu zeigt Abbildung 4.15 auf der nächsten Seite die Autokorrelationsfunktion von fünf Präambel-Symbolen mit binärem Code. Es ist erkennbar, dass in Abbildung 4.16 die Spitzen der Autokorrelationsfunktion bei ternärem Code erheblich weiter über dem restlichen Plot liegen. Das bedeutet, dass bei binärem Code der Schwellwert der Präambel-Erkennung weiter vom Rauschniveau entfernt ist, wodurch ein größeres Sicherheitsintervall (in den Abbildungen als „SI“ bezeichnet) besteht. Deswegen ist mit dem ternären Präambel-Code eine bessere Präambel-Erkennung möglich.

Als Optimierung des bisherigen Designs von Energie-Detektions-Empfängern untersuchte ich die Benutzung einer Komponente zur Ermittlung des Vorzeichens während der Präambel-Erkennung. Dies ist bei Pulsformen mit positivem

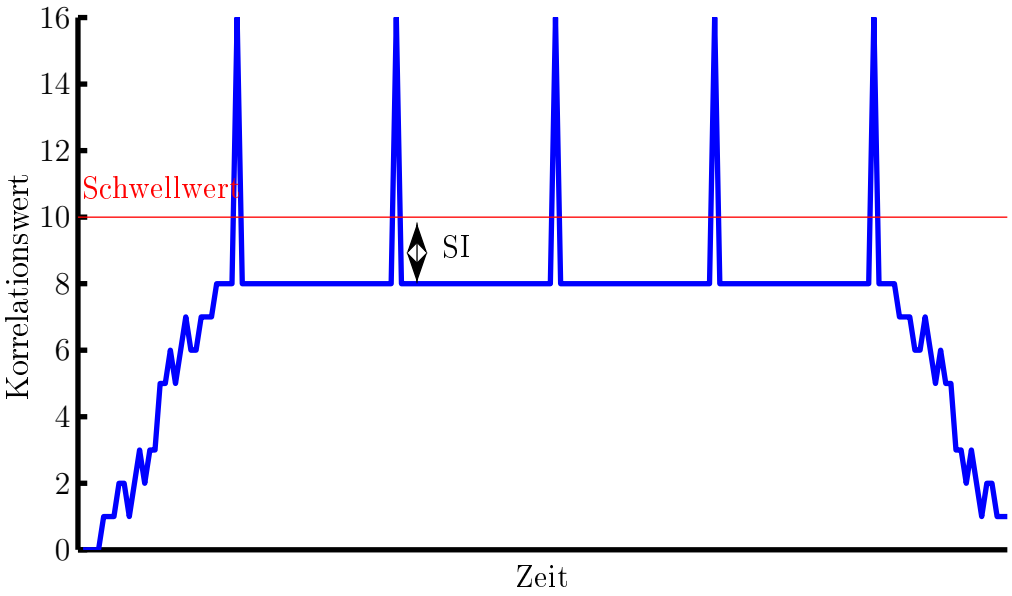


Abbildung 4.15: Autokorrelation von fünf Präambel-Symbolen mit binärem Code ohne Vorzeichendetektion

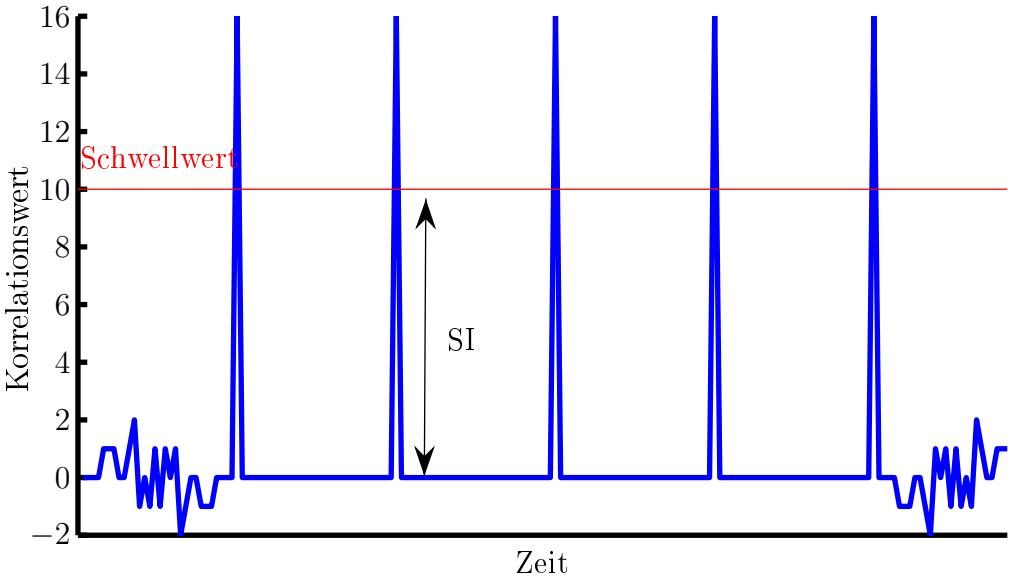


Abbildung 4.16: Autokorrelation von fünf Präambel-Symbolen mit ternärem Code mit Vorzeichendetektion

Betrag des Intervalls[8] mit einem einfachen *ADC* oder einem Komparator, der bereits vor der Quadrierung und Integration das Vorzeichen ermittelt, möglich.

Dieses Vorzeichen, gemittelt über das *Integrationsintervall*, wird der Präambel-Erkennung zusätzlich zum digitalisierten Wert des Integrators zur Verfügung gestellt. Dafür muss die Rate des Vorzeichens an das *Integrationsintervall* angeglichen werden, indem entweder ein paralleler Integrator eingesetzt wird oder die Digitalwerte zusammengefasst werden. Diese Methode profitiert deshalb auch von mehrpulsigen Präambeln, wie sie in Abschnitt 4.3.3.1 vorgestellt wurden.

Für den Empfang der Daten ist das Vorzeichen nicht nötig, weshalb diese Komponente nach erfolgreichem Präambel-Empfang abgeschaltet werden kann. Abbildung 4.17 skizziert einen Ausschnitt aus dem Empfänger.

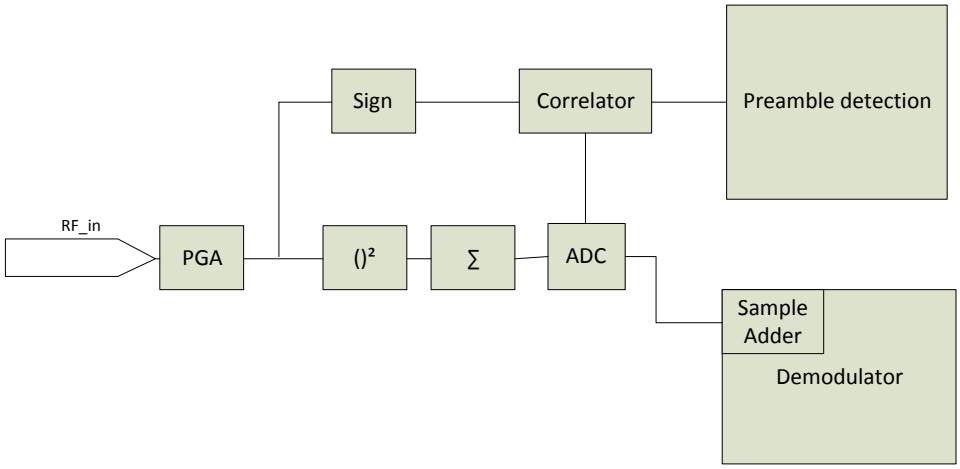


Abbildung 4.17: Schematische Darstellung eines Empfängers mit Vorzeichendetektion

Mit dieser Erweiterung kann ein Empfänger die Vorzeicheninformationen in der Präambel ermitteln und für die Korrelation den ternären Code benutzen. Zur numerischen Analyse wurde der Simulator um eine optionale, vorzeichenbehaftete Präambel-Erkennung erweitert, in der das Vorzeichen ermittelt wurde

[8] Beispielsweise *„Root-Raised-Cosine“* oder *„Gaussian Doublet“-Pulse*

und nach dem Quadrieren und Integrieren wieder auf den Wert multipliziert wird.

Zur besseren Vergleichbarkeit wurde den folgenden Abbildungen die herkömmliche Korrelation von binärem und ternärem Code verwendet. Wenn keine vorzeichenbehaftete Präambel-Erkennung möglich ist, lässt sich aber auch eine optimierte Korrelation verwenden, die beim binären Code die „0“-Symbole durch „-1“ ersetzt [48].

Wie in Abbildung 4.18 gezeigt, liegt die Rate der nicht erkannten Präambeln mit vorzeichenbehafteter Präambel-Erkennung niedriger als ohne Vorzeichenerkennung. Die rote Linie kennzeichnet die 10%-Marke, die als absolute Untergrenze für einen sinnvollen Betrieb zu verstehen ist.

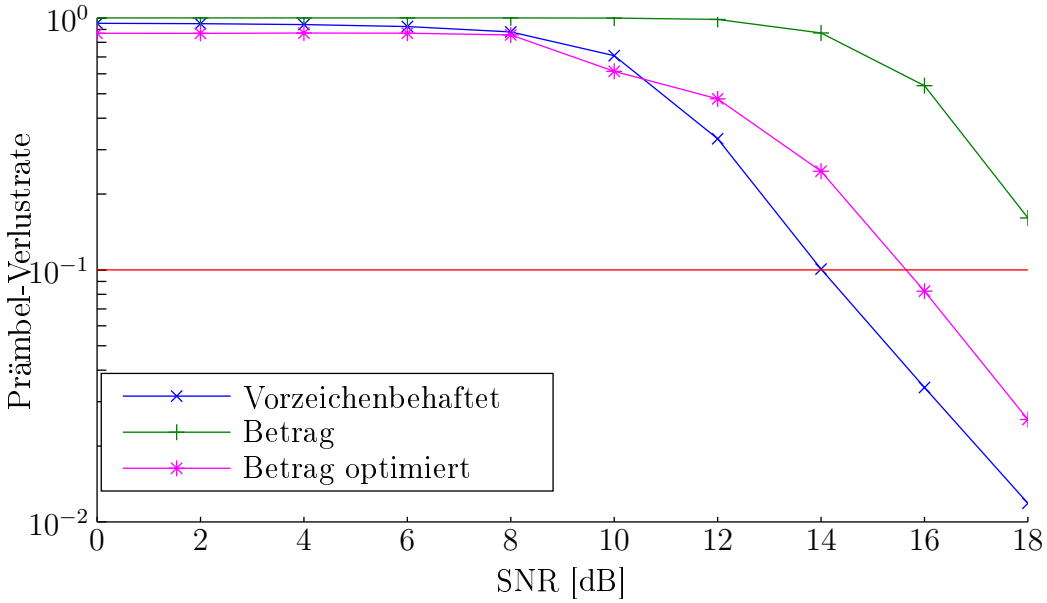


Abbildung 4.18: Rate von nicht erkannten Präambeln mit und ohne vorzeichenbehafteter Präambel-Erkennung

Abbildung 4.18 wurde mit einzelpulsigen Präambeln erstellt. Ein Vergleich mit vier Pulsen pro Präambel-Sub-Symbol, wie in Abschnitt 4.3.3.1 vorgestellt, ist in Abbildung 4.19 auf der nächsten Seite dargestellt. Dabei wurde nicht nur eine komplett verlorene Präambel, sondern auch fehlerhafte Synchronisation mit einer Abweichung von mehr als einem Sample als Fehler gewertet. Es zeigt sich, dass die vorzeichenbehaftete Präambel-Erkennung nicht nur weniger verlorene Präambeln verursacht, sondern auch eine genauere Synchronisation er-

möglicht. Als horizontaler Vergleichswert wurde eine rote Linie für realistische Mindestanforderung bei 1% eingezeichnet. Der Unterschied beträgt rund 10 dB.

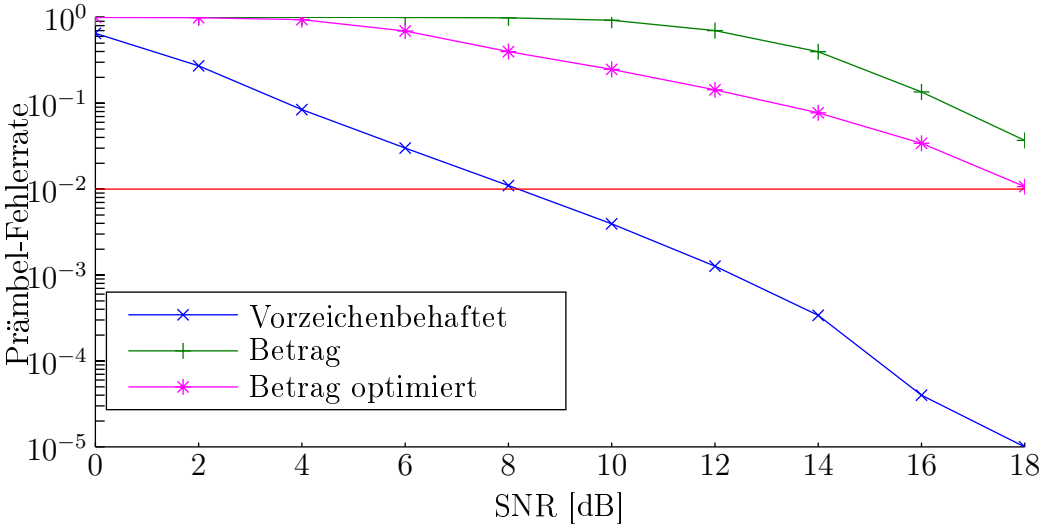


Abbildung 4.19: Rate von Präambel-Fehlern mit und ohne vorzeichenbehafteter Präambel-Erkennung bei vier Pulsen pro Präambel-Sub-Symbol

Diese Ergebnisse habe ich auf dem Kongress ICUWB 2010 präsentiert [HOSK10].

4.3.3.5 Phasenrichtige Addition der Samples in der Präambel-Erkennung

Je nach verwendetem *Integrationsintervall* und *L-Spreading* ergibt sich eine bestimmte Anzahl von Samples pro Präambel-Sub-Symbol. Wenn mehrere Samples pro Präambel-Sub-Symbol für die Präambel-Erkennung verarbeitet werden müssen, muss die Anzahl der Samples an die Anzahl der Code-Symbole angepasst werden.

Um im digitalen Basisband eines nichtkohärenten Empfängers eine bekannte Signalfolge, hier konkret der Präambel-Code, im Eingangssignal zu detektieren, gibt es verschiedene Methoden, von denen hier zwei betrachtet werden. Entweder wird das Eingangssignal mit dem gesuchten Code korreliert (Kreuzkorrelation) oder, bei sich wiederholenden Signalen, mit der vorangegangenen Wiederholung, also dem zwischengespeicherten Eingangssignal (Autokorrelation).

Beide Methoden bieten Vor- und Nachteile. Im Folgenden benutzte ich die Kreuzkorrelation, da sich diese sowohl für die Detektion eines einzelnen Präambel-Symbols als auch einer kompletten Präambel mit *SFD* eignet. Die Detektion des *SFD* kann nur über Kreuzkorrelation erfolgen, die Detektion von Präambel-Symbolen auch über Autokorrelation.

Für eine Kreuzkorrelation des Eingangssignals mit dem gesuchten Präambel-Code muss sich die Anzahl der Code-Symbole entsprechen. Die Präambel-Detektion kann nun den Präambel-Code erweitern, sodass er für jedes einzelne Sample ein Code-Symbol enthält. Je kürzer die Samples sind, desto mehr Probleme bereitet dann allerdings ein längerer *Delay Spread*, da sich die eigentlich gesuchten Signale über mehrere Samples ausbreiten, von denen die Späteren als nicht belegt erwartet werden. Dies verschlechtert die Empfangsleistung. Ebenso kann starkes Rauschen die Erkennungsleistung eines angepassten Präambel-Codes stark abschwächen.

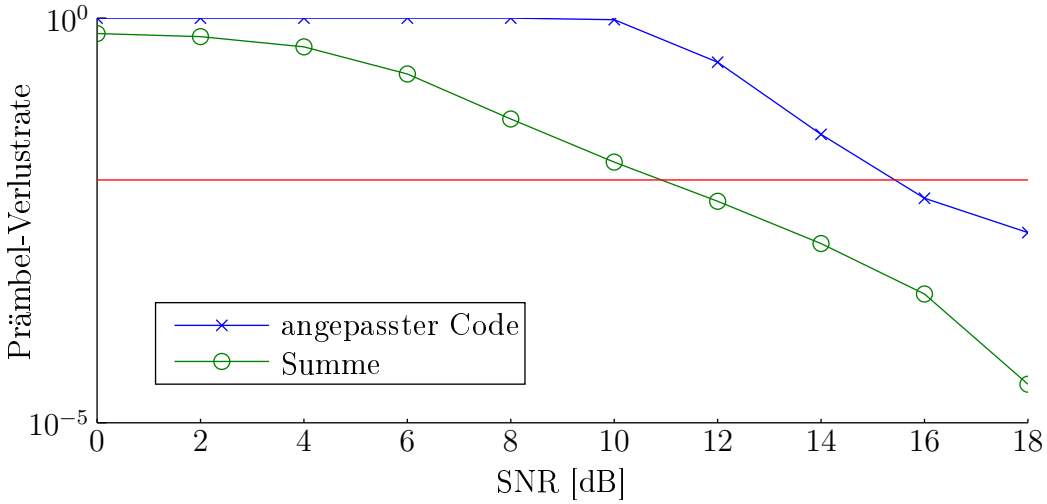


Abbildung 4.20: Vergleich der Rate verlorener Präambeln von Präambel-Erkennung mit angepasstem Code und Präambel-Erkennung mit aufsummierten Samples

Alternativ hierzu können zusätzliche Samples zu den jeweiligen Präambel-Sub-Symbolen aufaddiert werden, sodass die Korrelation direkt mit dem Präambel-Code stattfinden kann. Dies senkt auch die Frequenz, mit der der Präambel-Detektor arbeiten muss, was wiederum den Energieverbrauch verringert. Simulationen, deren Ergebnisse in Abbildung 4.20 auf der vorherigen Seite dar-

gestellt werden, zeigen außerdem, dass die Erkennungsleistung dieses Ansatzes erheblich besser ist.

Hier stellt sich allerdings die Frage, welche Samples zusammenaddiert werden müssen. Bei einer „blinden“ Addition der Samples verringert sich die Genauigkeit der Synchronisation auf die Länge eines Präambel-Sub-Symbols. Um dies zu umgehen, gibt es mehrere Möglichkeiten. Zur Verdeutlichung werden die Herangehensweisen zuerst an einen Beispiel erklärt und dann verallgemeinert.

Bei diesem Beispiel wird eine standardmäßige Präambel versendet, also mit $N_{Pburst} = 1$ und einem *L-Spreading* von 15 Pulsen mit jeweils 2 ns Dauer. Ein Präambel-Sub-Symbol hat also eine Länge von 32 ns. Diese Präambel wird mit einem *Integrationsintervall* von 16 ns abgetastet.

Für die erste Empfangsstufe (das Erkennen einzelner Präambel-Symbole) müssen jeweils zwei Samples zusammenaddiert werden, um die Präambel-Sub-Symbole zu erhalten. Da es immer zwei Samples sind, die aufaddiert werden, gibt es zwei verschiedene Möglichkeiten, den Strom an Samples zu addieren, von hier an als „Phase“ bezeichnet.

Abbildung 4.21 auf der nächsten Seite zeigt, wie die empfangenen Samples den gesendeten Präambel-Sub-Symbolen zugeordnet werden. Phase 1 und 2 geben die zwei Möglichkeiten an, wie die Samples im Empfänger aufaddiert werden können, wobei Phase 2 die korrekte Phasenlage ist.

Um die Richtige der beiden Phasen zu ermitteln, können mehrere Präambel-Detektoren parallel auf jeweils einer der Phasen arbeiten, wie in [56] beschrieben. Dabei wird das beste Ergebnis mehrerer parallel arbeitender Präambel-Detektoren als Synchronisationspunkt ausgewählt.

Für kürzere Samples oder höheres *L-Spreading* funktioniert diese Methode aber nur zusätzlich mit weiteren parallelen Präambel-Detektoren, was Komplexität und Kosten des Empfängers erhöht. Schon für das beschriebene Beispiel werden zwei parallele Präambel-Detektoren, jeweils inklusive *SFD*-Erkennung, benötigt, diese Anzahl steigt exponentiell mit dem Verhältnis zwischen Samples und *L-Spreading*.

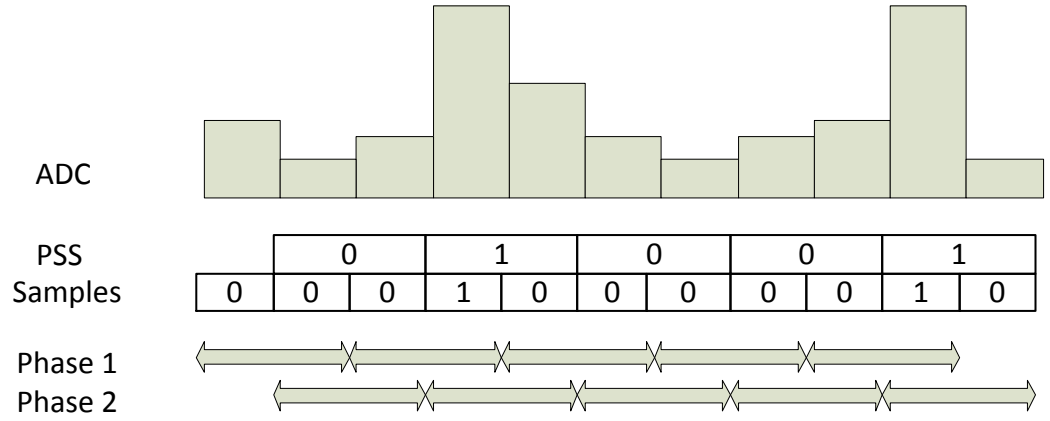


Abbildung 4.21: Addition der Samples zu Präambel-Sub-Symbolen in zwei verschiedenen Phasen

Ich entwickelte deshalb eine Komponente, die während der Addition der Samples, noch vor der Korrelation, die Phasenlage ermittelt und gegebenenfalls korrigiert.

Die Funktionsweise dieser Komponente beruht darauf, dass sich bei dieser Art der Präambel-Modulation das Sample, das Pulse enthält, stets am Anfang des Präambel-Sub-Symbols befindet. Dadurch zeichnet sich die richtige Phase für die Addition dadurch aus, dass das erste Sample eines Präambel-Sub-Symbols mit einem Wert von 1 bzw. -1 einen größeren Wert enthält als die restlichen Samples dieses Präambel-Sub-Symbols.

Die richtige Phasenlage ist also gegeben, wenn der Anfang eines Präambel-Sub-Symbols immer am lokalen Maximum der Samples des Präambel-Sub-Symbols liegt. Dies lässt sich bei der Addition der Samples durch ein iteratives Verfahren erreichen, welches ich hier vorstelle. Ich bezeichne das Verfahren als *IPPM*[9].

Die Komponente durchläuft den Algorithmus 1 auf der nächsten Seite bei jedem Sample. Dabei enthält MAX den lokalen Maximalwert, und die Ausgabe wird an den Korrelator des Präambel-Detektors weitergegeben.

IPPM versetzt den Anfangszeitpunkt des Präambel-Sub-Symbols und damit die Phasenlage der Addition immer dann, wenn eine Konstellation auftritt, in

[9]Iterative Poly Phase Matching - *iterative Mehrphasenkorrektur*

Algorithmus 1: Iterative Mehrphasenanpassung

Input : *Sample* - neuer Sample-Wert des ADC

Output : *Ausgabe* - Summierter Wert des Sub-Symbols für Präambel-Detektor

```
foreach Sample do
    Zaehler ← Zaehler + 1
    Summe ← Summe + Sample
    if Zaehler == NSubSym then
        /* Nächstes Präambel-Symbol, Phase bleibt erhalten   */

        MAX ← Summe/NSubSym                    // Maximalwert wird auf
        Durchschnittswert gesetzt

        Zaehler ← 0
        Summe ← 0
        Ausgabe ← Summe
    end
    if MAX < Sample then
        /* Der Samplewert ist höher als das bisherige Maximum,
           Phasenwechsel                                          */

        MAX ← Sample
        Zaehler ← 0
        Summe ← 0
    end
end
```

der ein späteres Sample innerhalb eines Präambel-Sub-Symbols größer ist als das erste Sample.

Es sollte vermieden werden, dass bei Präambel-Sub-Symbolen mit dem Wert „0“ die Phasenlage verrutscht, wenn in späteren Samples ein höherer Rauschpegel vorherrscht als im ersten Sample. Deshalb wird als initialer Schwellwert der Maximums-Suche für jedes Präambel-Sub-Symbol der Durchschnittswert

über das letzte Präambel-Sub-Symbol statt dem Wert „0“ verwendet.

Der Maximalwert wird dadurch zu Beginn jedes Präambel-Symbols knapp oberhalb des Rauschniveaus liegen. Dies dient auch der Absicherung, dass kein zu hoher Maximalwert, wie er beispielsweise bei der Kollision von Pulsen entstehen könnte, die Anpassung der Phase verhindert. Dies wird verhindert, indem nach jedem „leeren“ Präambel-Sub-Symbol der initiale Wert abgesenkt wird.

Es gäbe mehrere Wege, dieses Verfahren weiter zu verbessern. Zum Beispiel den Wert des Rauschniveaus über mehrere Präambel-Sub-Symbole zu mitteln, um den Maximalwert zu erhalten oder den Maximalwert um einen geringen Wert pro Präambel-Sub-Symbol zu senken. Die Komponente könnte dadurch auch benutzt werden, um die *AGC*-Steuerung zu unterstützen. Allerdings würde dies die Komponente komplexer gestalten, was sich negativ auf den Energieverbrauch auswirken würde, besonders da diese Komponente auf der Taktrate des *ADC*s laufen müsste.

Die Komponente wurde im Rahmen einer von mir betreuten Diplomarbeit [38] in *VHDL*[10] implementiert und zum Patent [P4] angemeldet.

[10] Very Highspeed integrated circuits Description Language - *Hardware-Beschreibungssprache*

4.3.3.6 Gesamtverbesserung gegenüber der Referenz

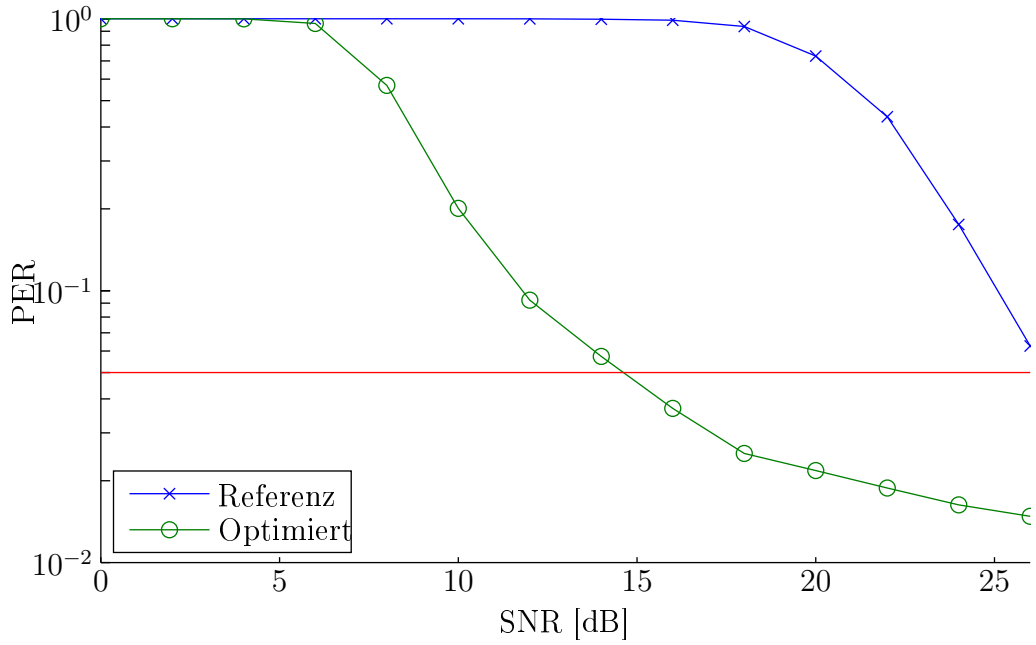


Abbildung 4.22: Vergleich der Paketfehlerrate eines standardkonformen Systems mit der optimierten PHY-Schicht

In Abbildung 4.22 zeigt einen Vergleich zwischen einem standardkonformen System und einem System mit den oben beschriebenen Optimierungen. Es ist erkennbar, das das optimierte System den gerade noch akzeptablen Grenzwert von 5% bereits bei einer 12 dB tieferen *SNR* als das Referenzsystem erreicht.

Als Referenz wurde ein standardkonformes System mit der Basisrate von IEEE 802.15.4a verwendet. Das optimierte System benutzt eine verkürzte Präambel aus einem einzelnen Präambel-Symbol mit $N_{Pburst} = 4$, der Datenteil benutzt weniger Hopping-Positionen als im Standard $N_{Hops} = 4$ und einen Faltungs-Code. Der Empfänger verwendet vorzeichenbehaftete Präambel-Erkennung mit *IPPM* und im Datenteil überlappende Addition mit Soft-Bit-Demodulation.

Neben der erkennbaren Steigerung der Robustheit ist das Paket im optimierten Modus mit 42 μs auch erheblich kürzer als ein standardkonformes Paket mit 122 μs.

4.4 Cross-Layer-Verbesserungen

Zusätzlich zu den Verbesserungen zur Senkung der Latenz in der *MAC*-Schicht und zur Erhöhung der Robustheit in der *PHY*-Schicht werden in diesem Abschnitt schichtübergreifende Konzepte vorgestellt, die zum Erreichen der Zielkriterien beitragen.

4.4.1 Zwei-Stufen-Synchronisation

Die optimale Länge der Präambel wird durch folgende Faktoren beeinflusst:

- welcher Sender an welchen Empfänger sendet
- die Zeit seit der letzten Übertragung von diesem Sender an diesen Empfänger
- die Qualität der Oszillatoren auf Sender- und Empfängerseite
- die zeitliche Veränderlichkeit des Kanals

In der *MAC*-Schicht ist vorgesehen, dass sich die Struktur des Superframes nach dem *Beacon* richtet. Deshalb muss jeder Teilnehmer auch den *Beacon* empfangen.

Um die Zeit zur Übertragung längerer Präambeln zu sparen, wird untersucht, ob der *Beacon* für eine „grobe“ Synchronisation benutzt werden kann, sodass für die folgenden Datenpakete die Verwendung einer kürzeren *Präambel* ausreichend ist.

Eine mögliche *Superframe*-Struktur hierfür wäre, dass der *Beacon* eine reguläre Präambel aus mehreren Wiederholungen des Präambel-Symbols und einen *SFD* enthält, die Datenpakete des Superframes sollen jedoch nur eine einzige Wiederholung des Präambel-Symbols ohne *SFD* enthalten.

Folgende Voraussetzungen müssen dazu erfüllt sein: Der Empfänger muss einen speziellen Präambel-Detektor enthalten, der auf eine Präambel ohne *SFD* reagieren kann. Außerdem muss dem Präambel-Detektor bekannt sein, ob das nächste Paket eine Präambel mit oder ohne *SFD* enthält, also ob das nächste Paket, welches der Empfänger empfangen muss, ein Daten-Frame oder ein

Beacon ist. Diese Information liegt in der *MAC*-Schicht vor. Es ist also ein Informationsaustausch über Schichtgrenzen hinweg erforderlich.

Dies erfordert allerdings auch, dass diese kurze *Präambel* ausreicht, um die restlichen Aufgaben der *Präambel* neben der Synchronisation zu erfüllen, die in Abschnitt 4.3.2.3 auf Seite 110 vorgestellt werden. Dies kann anhand der *AGC*-Einstellung verdeutlicht werden.

Die Kommunikation findet immer zwischen einem Knoten und dem *Controller* statt. Von einem Knoten aus gesehen, sind die Kommunikationsteilnehmer im Regelbetrieb stets die gleichen wie bei der vorangegangenen Übertragung, welche vor weniger als einer Zykluszeit T_{Zyklus} stattfand. So kann ein *Aktor* in den meisten Fällen (abhängig von der Veränderlichkeit des Kanals) denselben Wert für das *AGC*-Setting, den er im Beacon ermittelt hat, beibehalten, da der Sender derselbe ist. Der *Controller* hingegen muss für jeden Sensor ein individuelles Setting speichern, das er für die Übertragung laden und anhand der kurzen *Präambel* des Datenpakets dann verbessern kann.

Um zu berechnen, wie weit die interne Clock eines Empfängers nach einer erfolgreichen Synchronisation über den Beacon von seinem Controller abdriften kann, muss die Qualität des Oszillators bekannt sein. Diese wird in ppm (*parts per million* - Millionstel) gemessen. Der Standard schreibt eine Mindestgüte von 20 ppm vor [3]. Das bedeutet, dass der interne Oszillator eines standardkonformen Empfängers nach einer Million Perioden um die Dauer von höchstens 20 Perioden abweichen darf.

Da die Oszillatoren beider Transceiver, also der des Controllers und auch der jedes einzelnen Knotens, driften, wird für eine Berechnung des maximalen Fehlers angenommen, dass zwei Oszillatoren genau entgegengesetzt driften. Weiterhin wird der letzte Empfänger des Zyklus betrachtet und zur Vereinfachung eine komplette Zykluszeit als Abstand zwischen dem Synchronisationspunkt des Beacons und dem des letzten Datenpaketes angenommen. Eigentlich müsste mindestens die Zeit eines Zeitschlitzes abgezogen werden. Diese Abschätzung ist demnach als Obergrenze einzustufen.

Die Drift wird allgemein durch

$$T_{Drift} = (20\,\mathrm{ppm} \cdot 2) \cdot T_{Cycle} \tag{4.25}$$

berechnet.

Für eine typische Zykluszeit von 2 ms wird die Drift durch

$$\begin{aligned} T_{Drift} &= (20\,\text{ppm} \cdot 2) \cdot 2 \cdot 10^{-3}\,\text{s} \\ &= 40 \cdot 10^{-6} \cdot 2 \cdot 10^{-3}\,\text{s} = 80\,\text{ns} \end{aligned} \tag{4.26}$$

berechnet.

Selbst im schlimmsten Fall ist also nur eine Drift zu erwarten, die erheblich kürzer ist als ein einzelnes Präambel-Symbol. Mehr noch, die Auswirkung der *Clock Drift* ist gegenüber der Flugzeit eines Paketes nahezu vernachlässigbar. Die Vermutung war nun, dass, eine erfolgreiche Synchronisation auf den Beacon vorausgesetzt, für die Synchronisation der Datenpakete ein einzelnes Präambel-Symbol ausreichend ist.

Um dies zu verifizieren wurde eine Simulation durchgeführt, bei der die Auswirkungen einer künstlichen Drift auf den Beginn des Empfangs untersucht wurden. Dazu wurde der Startpunkt des Signals mit der oben erwähntem Drift verändert. Der Empfänger versucht sich ab diesem Startpunkt mit einem einzigen Präambel-Symbol zu synchronisieren. Dabei wurde die Abweichung des resultierenden Synchronisationspunktes und dem tatsächlich Anfang des Paketes in Samples gemessen.

Wie in Abbildung 4.23 auf der nächsten Seite für den schlechten *SNR*-Wert von 2 dB zu sehen ist, treten mit längerem Abstand zum erwarteten Synchronisationszeitpunkt mehr Fehler bei der Synchronisation auf. Dies liegt am Algorithmus der Synchronisation, der auf die erste Spitze der Korrelationswerte nach Überschreiten eines Schwellwertes synchronisiert. Ist dieser Schwellwert niedriger als die durch das Rauschen erzeugten Korrelationswerte (in der oben gezeigten Abbildung lag er bei „0“), dann können zufällige Muster im Rauschen fälschlicherweise als Präambel-Symbol identifiziert werden. Ist aber durch den Beacon ein solcher Schwellwert vorgegeben, dann treten diese fehlerhaften Synchronisationen mit Wahrscheinlichkeiten von $10^{-}9$ nicht mehr im relevanten Rahmen auf.

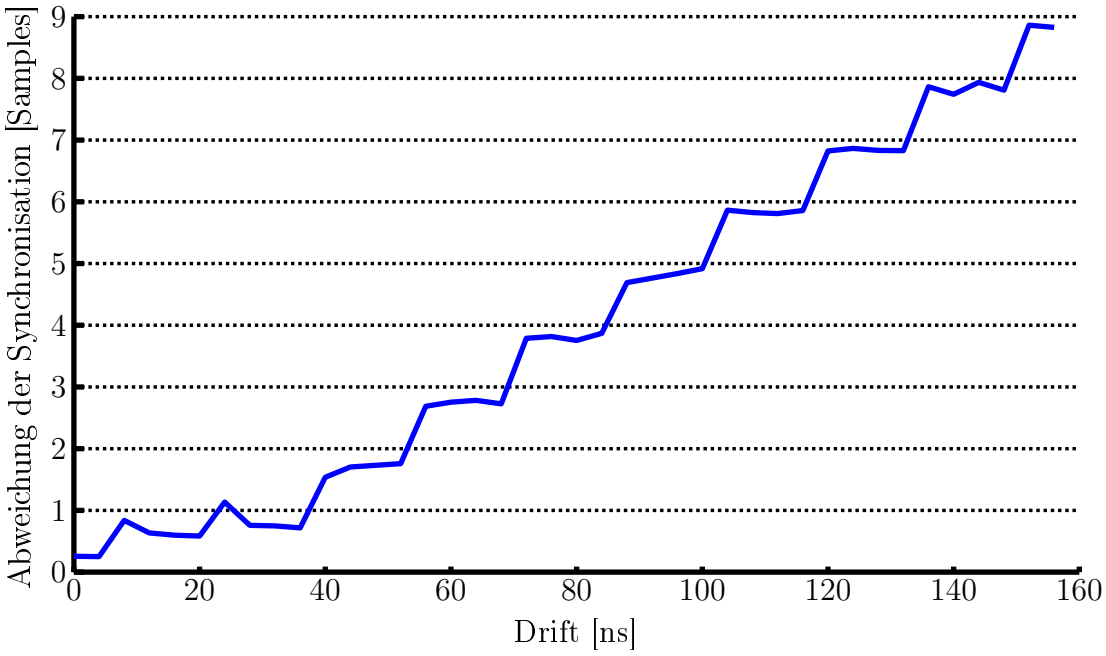


Abbildung 4.23: Abweichung der Synchronisation in Samples in Anhängigkeit der Clock Drift

4.4.2 Soft-Bit-Kombination

In der industriellen Automatisierung wird eine Steuerung durch regelmäßiges Senden von kurzen Paketen mit Prozessinformationen realisiert. Dies sind Sensordaten und Steuerbefehle für Aktoren. Bei vielen Applikationen bleiben diese Informationen über mehr als eine Zykluszeit des Funknetzes konstant, besonders wenn die Zykluszeit kürzer ist als das Aktualisierungsintervall der Sensoren oder der Zyklus der übergeordneten Steuerung.

Wenn die Prozessinformationen eines Knotens im Netzwerk länger als eine Zyklus-Dauer konstant bleiben, bedeutet dies, dass mehrere Datenpakete nacheinander mit derselben Nutzlast versendet werden. Bei einem Paketformat, das bei gleicher Nutzlast auch das gleiche Paket sendet, wird dadurch mehrmals hintereinander das gleiche Paket verschickt.

Um diese Redundanz auszunutzen, entwickelte ich ein System auf Basis des Soft-Bit-Empfangs von Abschnitt 4.3.3.3 auf Seite 124.

Dabei behält ein Empfänger die empfangenen Soft-Bit-Informationen in einem Zwischenspeicher, auch wenn die Daten nicht erfolgreich empfangen werden konnten. Sollte ein weiteres Paket empfangen werden, muss festgestellt werden, ob es eine erneute Übertragung des vorangegangenen Paketes ist. Wenn es das gleiche Paket ist und der Empfang wieder misslingt, kann der Empfänger die empfangenen Soft-Bits dieses Paketes zu denen im Zwischenspeicher addieren und versuchen, die daraus resultierenden Daten zu empfangen.

Da Soft-Bits mit einer höheren Sicherheit der Entscheidung einen höheren Betrag als „unsichere“ Soft-Bits haben, wird bei der Addition ein Soft-Bit mit einer „sichereren“ Entscheidung ein entgegengesetztes, „unsichereres“ Soft-Bit überdecken. Hingegen werden sich zwei gleichgerichtete relativ „unsichere“ Soft-Bits addieren und die Sicherheit erhöhen.

Das folgende Beispiel soll die Technik exemplarisch verdeutlichen. Dabei wird ein Paket von 10 Bit (zur Vereinfachung wird 1-0-1-0 etc. gesendet) dreimal hintereinander gesendet. Die Abbildungen 4.24 bis 4.28 verdeutlichen die empfangenen Soft-Bits. Die Höhe der Balken bedeutet die Höhe der Sicherheit mit einem Wertebereich zwischen -1 und 1, wobei die Richtung den Bit-Wert symbolisiert. Die Farbe weist darauf hin, ob dieser Bit-Wert korrekt oder inkorrekt ist.

Abbildung 4.24 zeigt den Empfang der ersten Übertragung. Dabei wurden das sechste und neunte Bit falsch erkannt, das Paket wurde also fehlerhaft empfangen.

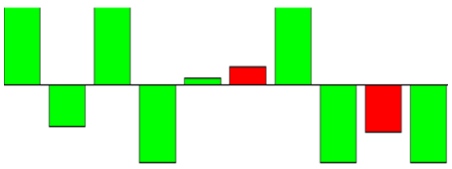


Abbildung 4.24: Erste empfangene Übertragung

Diese Soft-Bit-Informationen werden zwischengespeichert. Beim Empfang der zweiten Übertragung (s. Abbildung 4.25 auf der nächsten Seite) wurden das zweite und fünfte Bit falsch erkannt.

Abbildung 4.25: Zweite empfangene Übertragung

Da dies eine erneute Übertragung desselben Paketes ist, können die Soft-Bits der ersten und zweiten Übertragung addiert werden, was zu dem Ergebnis in Abbildung 4.26 führt.

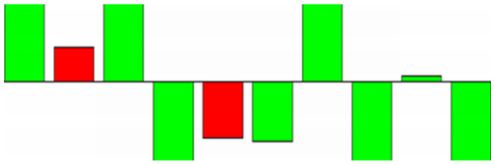


Abbildung 4.26: Addition der ersten und zweiten Übertragung

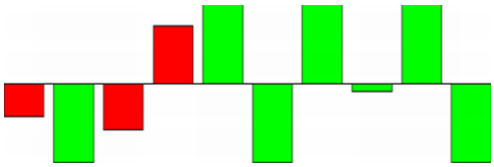


Abbildung 4.27: Dritte empfangene Übertragung

Selbst wenn beim Empfang der dritten Übertragung in Abbildung 4.27 immer noch Fehler auftreten, so kann durch Addition aller drei Übertragungen (siehe Abbildung 4.28) das ursprüngliche Paket wiederhergestellt werden. So kann durch Kombination von drei fehlerhaften Übertragungen desselben Paketes trotzdem eine erfolgreiche Übertragung erreicht werden.

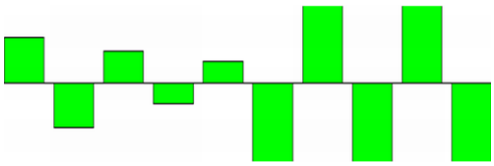


Abbildung 4.28: Addition aller drei Übertragungen

Der Simulator wurde erweitert, um Paket-übergreifende Simulationen zu ermöglichen und einen Soft-Bit-Cache verwalten zu können.

In einer Simulation habe ich die Anzahl der benötigten Übertragungen bis zu einer erfolgreichen Paketübertragung von einem System mit Soft-Bit-Kombination gegen eine Referenz mit harter Bit-Entscheidung verglichen. In beiden Fällen wurde kein *FEC*-Code verwendet. Abbildung 4.29 auf der nächsten Seite zeigt, dass mit der Soft-Bit-Kombination erheblich weniger Wiederholungen für eine erfolgreiche Übertragung benötigt werden. Nach 100 erfolglosen Übertragungen wurde die Simulation abgebrochen, da Werte in diesem Bereich keine Relavanz mehr haben. Akzeptable Werte sind bis zu 4 Übertragungen, wie mit der roten Linie markiert wurde.

Diese Technik ähnelt dem sog. „*Chase Combining*“, das als Hybrid-*ARQ*[11]-Mechanismus in *HSPA*[12]-Netzen verwendet wird [75]. Allerdings muss dort der Empfangserfolg eines jeden Pakets mit einer *ACK*- oder *NACK*[13]-Nachricht bestätigt werden. Im Falle eines Paketverlustes, signalisiert durch das Ausbleiben einer *ACK*-Nachricht oder Versand einer *NACK*-Nachricht, wird eine erneute Übertragung veranlasst.

Anders als beim *Chase Combining* finden in dem hier vorgestellten Verfahren Wiederholungen implizit statt. Trotzdem muss aber der Empfänger die Information darüber haben, ob es sich bei einem empfangenen Paket um ein neues Paket oder die Wiederholung eines vorangegangenen Paketes ist.

Diese Information kann durch zwei Wege erlangt werden. Wenn die Anzahl von Zyklen des Funknetzes pro Zyklus der Applikation bekannt ist, kann die *MAC*-Schicht die Anzahl der Wiederholungen verfolgen und diese Information der *PHY*-Schicht zur Verfügung stellen.

Für Fälle, in denen diese Information nicht bekannt ist, schlage ich vor, eine Markierung zwischen Präambel und Datenteil einzufügen. Mit dieser Markierung teilt der Sender mit, ob das versendete Paket eine Wiederholung des Vorangegangenen oder ein neues Paket ist. Der Empfänger kann anhand dieser Information entscheiden, ob eine Soft-Bit-Kombination vorgenommen werden

[11] Automatic Repeat Request - *Automatische Übertragungswiederholung bei Paketverlust*

[12] High Speed Packet Access - *Schneller Datenübertragungsmodus in UMTS-Mobilfunknetzen*

[13] *negatives* Acknowledgement-*Paket, weist den Sender auf einen Paketverlust hin*

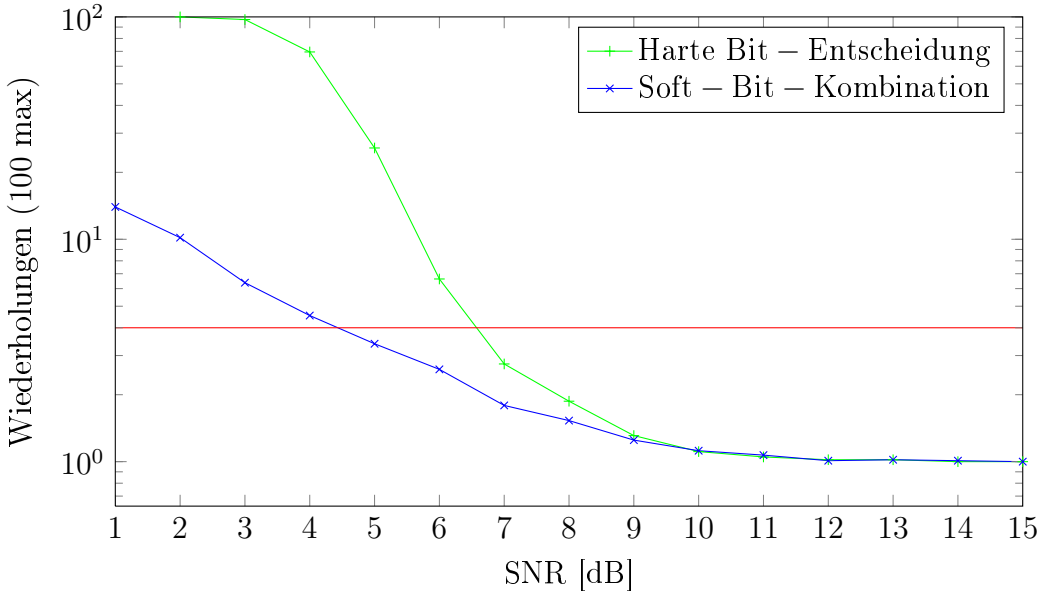


Abbildung 4.29: Benötigte Übertragungen mit harter Bit-Entscheidung und mit Soft-Bit-Kombination

kann oder der bisherige Zwischenspeicher mit den neu empfangenen Soft-Bits zu überschreiben ist, falls das Paket fehlerhaft empfangen wurde.

Diese Markierung, genannt CI[14], ist eine Boolesche Information. Da diese Technik aber besonders die Robustheit für sehr schwierige Funkkanäle erhöhen soll, wird diese Information abgesichert, indem sie über mehrere Symbole hinweg gespreizt wird. Dazu wird ein Code mit sehr guten Autokorrelations-Eigenschaften benötigt.

Dies kann z.B. ein Barker-Code sein oder aber einer der Präambel-Codes. Da sich die Daten-Modulation gegenüber der Präambel-Modulation als robuster erweist, die Präambel-Modulation aber mehr Code-Symbole in kürzerer Zeit überträgt, muss man hier die Effektivität vergleichen. Deshalb habe ich die Simulation für die Benutzung eines Barker-Codes mit vier Symbolen (+1 +1 -1 +1) in Daten-Modulation und die Benutzung eines der optionalen Präambel-

[14]change indicator - *Veränderungsanzeiger: Präfix vor einem Frame, der anzeigt, ob es seit der letzten Übertragung zu einer Änderung der Daten kam.*

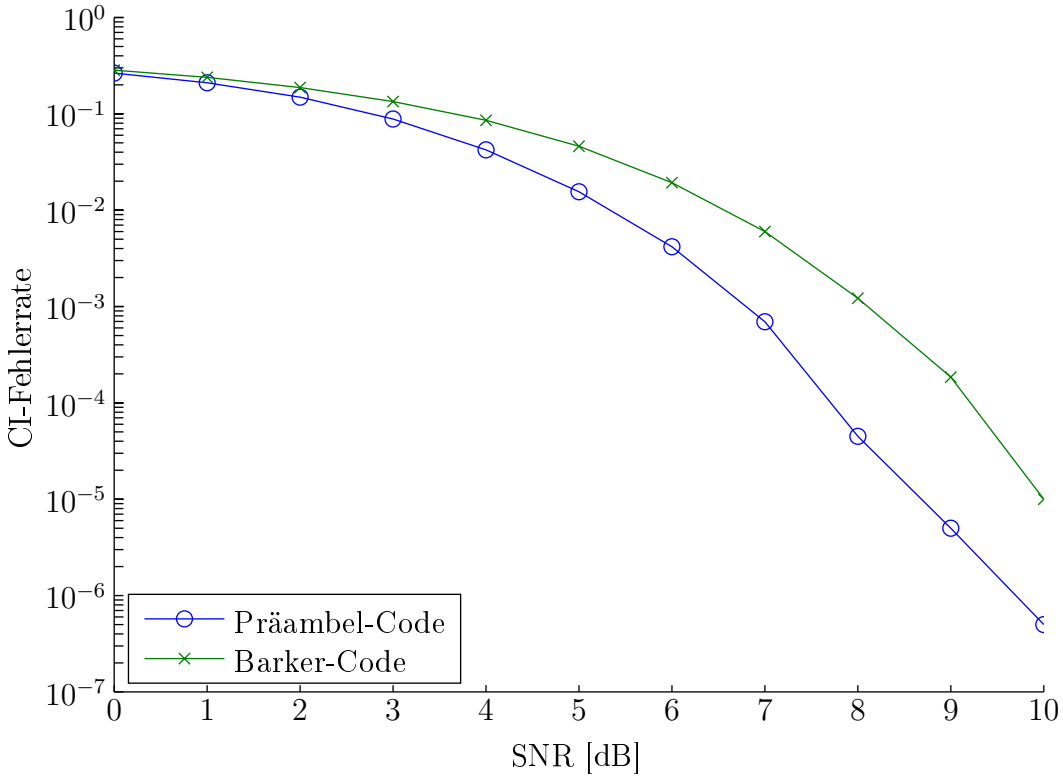


Abbildung 4.30: Vergleich von CI-Fehlern bei Präambel- und Datenmodulation

Codes des Standards IEEE 802.15.4a (Code Nummer 8) als *CI* angepasst und beide Möglichkeiten simuliert.

Der Code in Daten-Modulation zeigte bessere Ergebnisse in Kanälen mit schweren Multipfad-Effekten, wohingegen der Präambel-Code robuster gegen Rauschen ist. Abbildung 4.30 zeigt den Verlauf von falsch erkannten *CI* beider Modulationen bei sinkender *SNR*. Dabei wurde das industrielle *LOS*-Kanalmodell verwendet.

Der Präambel-Code wurde gewählt, da er zeitlich kürzer ist und außerdem eine Wiederverwendung von Komponenten erlaubt. Ein Nachteil hierbei ist die Kreuzkorrelation zwischen dem jeweils verwendeten Präambel-Code in Präambel und *CI*, die höher ist als die Kreuzkorrelation gegen den Barker-Code in Daten-Modulation. Die Suche nach einem optimalen Code für den *CI* ist jedoch ein Thema für zukünftige Arbeiten.

Auch der Viterbi-Decoder profitiert erheblich von der Soft-Bit-Kombination.

In einer weiteren Simulation mit vier Übertragungen der Pakete zeigt sich in Abbildung 4.31, dass ein System mit Soft-Bit-Kombination und einem Viterbi-Code eine deutlich geringere Paketfehlerrate aufweist als eines mit harter Bit-Entscheidung mit *Reed-Solomon-Code*. Ein System mit Soft-Bit-Kombination ohne *FEC*-Code erreicht ebenfalls niedrigere Paketfehlerraten als das System mit harter Bit-Entscheidung.

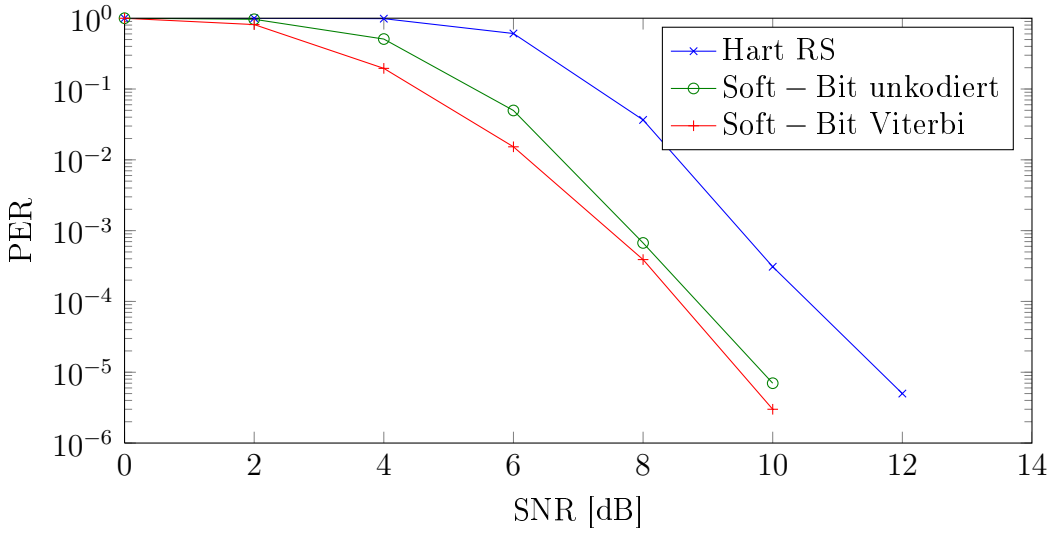


Abbildung 4.31: Paketfehlerraten mit verschiedenen FEC-Systemen

Diese und weitere Simulationsergebnisse habe ich in [HKS11] veröffentlicht. Das Verfahren ist zum Patent angemeldet [P5].

4.5 Zusammenführung in ein verbessertes Zielsystem

Die oben beschriebenen Verbesserungen ermöglichten es, ein flexibles System für die drahtlose Automatisierung zu entwerfen.

Dabei besteht das Automatisierungssystem aus einem Netz aus sternförmigen Zellen, die jeweils einem Controller zugeordnet sind. Jede Zelle bildet ein eigenständiges Netzwerk der Sensor/Aktor-Ebene, dessen Controller als Gate-

way zu einer übergeordneten Steuerung fungiert. Die Zellen operieren auf unterschiedlichen Frequenzkanälen und/oder Präambel-Codes und können sich dadurch auch überdecken. Je nach Größe der Zellen können Frequenzkanäle und Präambel-Codes auch öfters verwendet werden.

In diesem Abschnitt werden die oben beschriebenen Lösungen zu einem System zusammengeführt und die Berechnung der erreichbaren Zykluszeiten des Funksystems vorgestellt. Dieses System bietet verschiedene Konfigurationsmöglichkeiten. In Abschnitt 4.6 auf Seite 151 wird eine Konfiguration als drahtloser Ersatz für *AS-Interface* vorgestellt.

4.5.1 Konfigurierbarkeit der Präambel

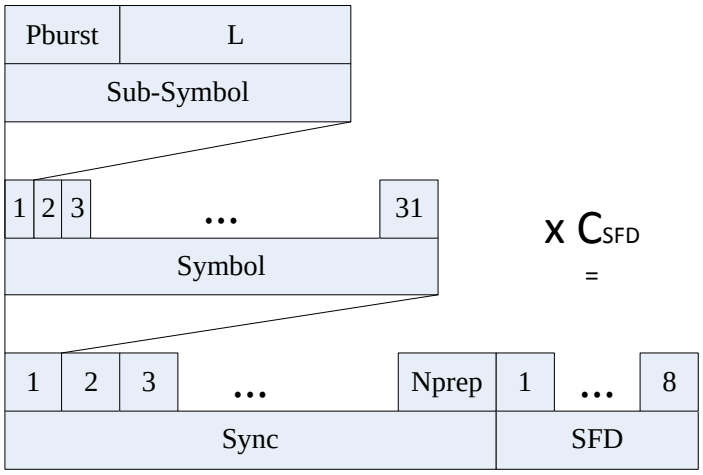


Abbildung 4.32: Konfigurationsmöglichkeiten in der Präambel

Die Präambel kann entweder aus einem einzelnen Präambel-Symbol oder aus N_{Prep} Wiederholungen des Präambel-Symbols und einem *SFD* bestehen. Der *SFD* besteht immer aus acht Präambel-Symbolen. Die Länge eines Präambel-Symbols (T_{Psym}) setzt sich zusammen aus der Anzahl der Pulse und einer Pause zwischen den Pulsen, hier als L bezeichnet. Diese beiden Werte zusammen ergeben den Abstand zwischen den *Burst*s der Präambel-Sub-Symbole und damit die Länge eines Präambel-Sub-Symbols.

$$T_{Prea} = N_{Prep} \cdot T_{Psym} + SFD \cdot T_{Psym} \tag{4.27}$$

$$T_{Psym} = 31 \cdot (L + N_{Pburst}) \cdot T_C \tag{4.28}$$

4.5.2 Konfigurierbarkeit des Datenteils

Den Daten wird ein *PHY*-Header von 13 Bit Länge, abgesichert von einem 6 Bit langen *SECDED*-Code, vorangestellt. Die Daten selbst können mit dem standardkonformen *Reed-Solomon-Code* oder einem Faltungs-Code kodiert werden. Die Länge eines Symbols ist abhängig von der Anzahl der Hopping-Positionen und der Pulse pro Burst.

$$N_{Bits}(Nutzlast, FEC) = 18 + Nutzlast + FEC \tag{4.29}$$

$$T_{Sym} = 4 \cdot N_{Hops} \cdot N_{CPB} \cdot T_C \tag{4.30}$$

4.5.3 Konfigurierbarkeit des Daten-Frame

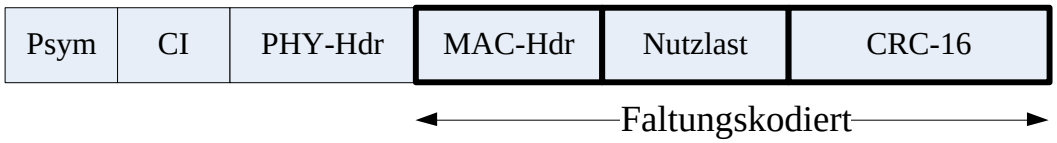


Abbildung 4.33: Struktur des Datenframes

Der Daten-Frame wird durch eine verkürzte Präambel mit einem einzelnen Präambel-Symbol ohne *SFD* eingeleitet. Er enthält zwischen Präambel und *PHY*-Header einen *CI*, bestehend aus einem Präambel-Symbol von einem optionalen, ungenutzten Präambel-Code. Die Nutzlast eines Daten-Frame besteht aus einem 8 Bit langen *MAC*-Header, der Nutzlast der Applikation und einer 16 Bit langen *CRC*-Prüfsumme. Sie wird optional mit einem Faltungs-Code der Rate $R = \frac{1}{2}$ abgesichert.

$$NL_{Frame} = \frac{1}{R} \cdot (8 + NL_{App} + 16) \tag{4.31}$$

$$\begin{aligned} T_{Frame} &= CI + T_{Prea} + (NL_{Frame} + 18) \cdot T_{Sym} \\ &= 2 \cdot 31 \cdot (L + N_{Pburst}) \cdot T_C \\ &+ (NL_{Frame} + 18) \cdot (4 \cdot N_{Hops} \cdot N_{CPB} \cdot T_C) \end{aligned} \tag{4.32}$$

4.5.4 Konfigurierbarkeit des Beacon

Abbildung 4.34: Struktur des Beacon

Die Struktur des Beacon ist in Abschnitt 4.2.2.3 auf Seite 101 beschrieben. Sie ergibt eine fünf Byte lange Datenstruktur, gefolgt vom *Group ACK*, das ebenso viele Bits N_{Sensor} enthält, wie Sensorknoten im Netz enthalten sind. Diese Struktur muss auf das nächste Byte *aligned* werden, weshalb die Länge auf das nächste Vielfache von Acht erhöht wird. Diese Nutzlast wird mit einem der vorgestellten *FEC*-Codes abgesichert. Dem Beacon wird eine gewöhnliche Präambel mit *SFD* vorangestellt.

$$NL_{Beacon} = 40 + N_{Sensor} + FEC \tag{4.33}$$

$$\begin{aligned} T_{Beacon} &= (N_{Prep} + 8) \cdot [31 \cdot (L + N_{Pburst}) \cdot T_C] \\ &+ (NL_{Beacon} + 18) \cdot (4 \cdot N_{Hops} \cdot N_{CPB} \cdot T_C) \end{aligned} \tag{4.34}$$

4.5.5 Konfigurierbarkeit des Superframe

Der Superframe besteht, wie in Abbildung 4.5 auf Seite 100 dargestellt, aus einem Beacon-Zeitschlitz und je einem Zeitschlitz für jeden Knoten. Die Zeitschlitze der Knoten haben die Länge eines Daten-Frame zzgl. eines Sicherheitsabstandes in der Länge von 12 Präambel-Symbolen.

$$
\begin{aligned}
T_{Superframe} &= T_{Beacon} + N_{Knoten} \cdot (12 \cdot T_{Psym} + T_{Frame}) \qquad (4.35)\\
&= [N_{Knoten} \cdot (12+2) + N_{Prep} + 8] \cdot [31 \cdot (L + N_{Pburst}) \cdot T_C]\\
&+ [N_{Knoten} \cdot (2 \cdot 8 + 16 + NL_{App}) + 2 \cdot (7 + N_Sensor)]\\
&\cdot (4 \cdot N_{Hops} \cdot N_{CPB} \cdot T_C)
\end{aligned}
$$

4.6 Beispielkonfiguration: TAWASI

Die Formeln aus Abschnitt 4.5 sollen an dem konkreten Beispiel eines drahtlosen Ersatzes für das *AS-Interface*-Protokoll verdeutlicht werden. Als Name für diese Lösung wurde ein in Linux-Kreisen übliches Retronym gewählt: *TAWASI*[15]. Über dieses Funksystem habe ich einen Konferenzbeitrag [HBSK11] auf dem Jahreskolleg der Automatisierung (KommA 2011) veröffentlicht.

Dazu werden für N_{Knoten} die 32 möglichen Knoten des *AS-Interface*-Protokolls (bei normaler Adressierung) und für die Nutzlast der Applikation 1 Byte festgelegt (*AS-Interface* ermöglicht 4 Bit Nutzlast in jeder Richtung).

Die T_C wird mit 2 ns als gegeben betrachtet. Für die *PHY*-Parameter des Datenteils werden $N_{CPB} = 16$ und $N_{Hops} = 4$ festgelegt. Der Wert von N_{CPB} entspricht dem Standard und erlaubt dadurch die Benutzung eines standardkonformen analogen Front-End. Der geringere Wert von N_{Hops} im Vergleich zum Standard halbiert die Symbolzeit zum Preis von geringerem gleichzeitigem Zugriff.

Für den Beacon wurde ein industrieüblicher (171,133)-Faltungs-Code der Rate $\frac{1}{2}$ mit einer Tiefe von 7 verwendet. Dieser erzeugt zwar einen längeren Beacon, aber dadurch müssen bei den Aktoren nicht zwei verschiedene Decoder verwendet werden. Wenn dieser zusätzliche Implementierungsaufwand[16] der zwei verschiedenen *FEC*-Decodern tragbar ist, dann ist ein *Reed-Solomon-Code* beim Beacon vorzuziehen, da der Beacon mit 144 Bit eine Länge besitzt, die den

[15] TAWASI Ain't Wireless AS-I - *Drahtloser Ersatz für das AS-Interface-Protokoll*

[16] Da der tatsächliche Aufwand stark von der verwendeten Chiptechnologie abhängig ist, können hier keine genauen Angaben gemacht werden.

Reed-Solomon-Code rechtfertigt. Im Weiteren wird die Variante angenommen, die nur einen Viterbi-Decoder auf den Knoten voraussetzt. Dadurch ändern sich die *PHY*-Parameter mit Ausnahme der Präambel-Wiederholungen zwischen *Beacon* und Datenpaketen nicht und die *PHY*-Schicht des Empfängers muss nicht zwischen *Beacon* und Datenpaket rekonfiguriert werden.

Die Parameter für den Präambel-Teil wurden mit $L = 12$, $N_{Pburst} = 4$ und für den Beacon zusätzlich mit $N_{Prep} = 16$ festgelegt. Dabei wurde die Balance zwischen Robustheit und Latenz eher in Richtung Latenz gewählt, da anzunehmen ist, dass durch die Maßnahmen von Abschnitt 4.3.3 auf Seite 114 eine ausreichende Verlässlichkeit gegeben ist.

4.6.1 Latenz

Werden die Konfigurations-Parameter von *TAWASI* in die Formeln 4.28 bis 4.35 eingesetzt, so erhält man diese Ergebnisse:

$$\begin{aligned}
T_{Sym} = &\quad 4 \cdot N_{Hops} \cdot N_{CPB} \cdot T_C = 256 \cdot T_C &&= 512\,\text{ns} \\
T_{Psym} = &\quad 31 \cdot L \cdot T_C = 496 \cdot T_C &&= 992\,\text{ns} \\
NL_{Beacon} = &\quad 40 + N_{Knoten} + FEC &&= 144\,\text{Bit} \\
T_{Beacon} = &\quad N_{Prep} \cdot T_{Psym} + (18 + NL_{Beacon}) \cdot T_{Sym} &&\approx 105,9\,\mu\text{s} \\
NL_{Frame} = &\quad 2 \cdot (8 + NL_{App} + 16) &&= 64\,\text{Bit} \\
T_{Frame} = &\quad T_{Psym} + T_{Psym} + (18 + NL_{Frame}) \cdot T_{Sym} &&\approx 43,01\,\mu\text{s} \\
T_{Superframe} = &\quad T_{Beacon} + N_{Knoten} \cdot (12 \cdot T_{Psym} + T_{Frame}) &&\approx 1,88\,\text{ms}
\end{aligned} \tag{4.36}$$

Das System kann also eine Zykluszeit von 1,88 ms erreichen. Dies ist mehr als doppelt so schnell wie das *AS-Interface*-System mit einer Zykluszeit von 5 ms. Wenn man annimmt, das die Stuerung auf *AS-Interface* ausgelegt ist, wird das Funksystem für jeden Zyklus der Steuerung mindestens zwei (genauer $\frac{5}{1,88} = 2,66$) Zyklen durchlaufen. Dies erzeugt zusätzliche Redundanz, die besonders in Verbindung mit der Soft-Bit-Kombination die Robustheit erhöht.

4.6.2 Restfehlerrate

Die Restfehlerrate wird berechnet als Anzahl der empfangenen Pakete, die fehlerhafte Bits enthalten, aber nicht als fehlerhafte Daten erkannt werden. Dieser Wert wird in Abhängigkeit der Bit-Fehlerrate berechnet.

Zuerst wird betrachtet, wie sich Bit-Fehler im *PHY*-Header auswirken. Der *PHY*-Header ist über einen *SECDED*-Code geschützt, dessen Hamming-Distanz 2 beträgt. Es könnte also eine gerade Anzahl (≥ 4) an Bit-Fehlern unbemerkt bleiben. Da jedoch nur das Längenfeld des *PHY*-Header genutzt wird, hat ein Bit-Fehler dort maximal eine falsche Längenangabe des Paketes zur Folge. Ein Paket mit diesem Defekt wird jedoch in der *MAC*-Schicht immer verworfen werden, da hier zufällige Werte im *CRC*-Code stehen und zusätzlich die Länge des Paketes nicht mit der erwarteten Länge übereinstimmt.

Deshalb wird nur der Nutzdatenteil des *MAC*-Pakets betrachtet. Der Faltungs-Code, der statt einem *Reed-Solomon-Code* eingesetzt wird, senkt lediglich die Bit-Fehlerrate, aber ermöglicht keine Fehlererkennung auf *PHY*-Ebene. Für die Fehlererkennung der *MAC*-Ebene wird zur Kompatibilität mit dem Draft IEEE 802.15.4e eine 16-Bit-*CRC*-„Prüfsumme“ mit dem ITU-T-Generator-Polynom genutzt. Dieser Code besitzt eine Hamming-Distanz[17] von 4 [46].

Zu beachten ist, dass in diesem Fall die Bit-Fehlerrate nach dem Viterbi-Decoder ausschlaggebend ist. Durch die Soft-Bit-Technik, speziell die paketübergreifende Kombinationsmethode, ist es nicht möglich, den Einfluss auf die Bit-Fehlerrate im Kanal direkt zu berechnen.

Berechnet man mit Formel (2.2) auf Seite auf Seite 14 diese Restfehlerrate für verschiedene Bit-Fehlerraten, so erhält man den Plot in Abbildung 4.35 auf der nächsten Seite. Diese Gerade dient auch zur Bestimmung der Datenintegritätsklasse.

Die Werte in 4.35 auf der nächsten Seite bedeuten, wie wahrscheinlich ein unbemerkter Fehler ist. Legt man z.B. eine Netto-Bit-Fehlerrate (nach dem

[17] Die eigentliche Hamming-Distanz ist von der Länge der Nutzlast abhängig (längere Pakete bedeuten eine geringere Hamming-Distanz). Der hier vorgestellte Wert ist eine in der Literatur übliche Verallgemeinerung, die mit sehr hoher Wahrscheinlichkeit für kurze Paketlängen zutrifft.

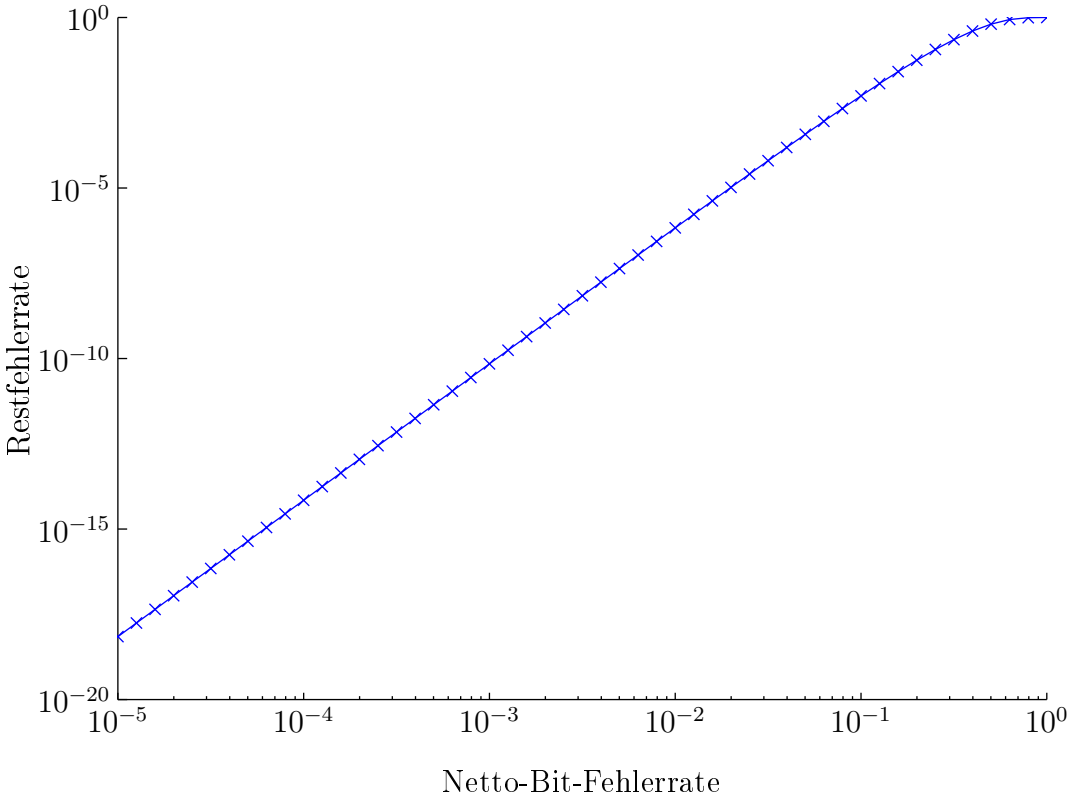


Abbildung 4.35: Restfehlerrate von TAWASI-Datenpaketen

Viterbi-Decoder) von $\leq 10^{-4}$ zugrunde, dann beträgt die Restfehlerrate $\leq 10^{-14}$.

Das System erfüllt damit für die Anforderungen der Datenintegritätsklasse I2 der DIN-Norm 19244, bei Bit-Fehlerraten $\leq 10^{-3}$ auch die Datenintegritätsklasse I3.

4.6.3 Code-Effizienz

Die Code-Effizienz kann durch

$$E = \frac{k(1-p)^n}{n} \tag{4.37}$$

bestimmt werden, wobei k die Anzahl der Informations-Bits (= Nutzdaten) ist, n die Zahl der Bits pro Frame gesamt und p die untersuchte Bit-Fehlerrate.

Für das *MAC*-Paket (wenn man den *MAC*-Header nicht zu den Nutzdaten zählt) ergibt sich damit eine Effizienz von nur 25%. Für das *PHY*-Paket sogar eine Code-Effizienz von lediglich 10%. Diese geringe Effizenz ist mit der Kompatibilität zu Standards zu rechtfertigen, für proprietäre Systeme wäre eine effizientere Integritätsprüfung oder ein komprimiertes Paketformat angemessener.

4.6.4 Datenrate

Da ein Symbol ein Bit überträgt und die Länge $T_{Sym} = 512\,\text{ns}$ hat, beträgt die Brutto-Datenrate

$$\frac{1Bit}{T_{Sym}} \approx 1.9\,\text{MBit/s} \tag{4.38}$$

.

Die Berechnung einer Nettodatenrate erfordert eine genauere Definition des Begriffs. Durch Betrachtung eines einzelnen Knotens (Sensor oder Aktor) lässt sich dessen Nutzdatenrate als ein Byte pro Zyklus angeben, was einer Datenrate von

$$\frac{8Bit}{T_{Superframe}} \approx 4.3\,\text{kBit/s} \tag{4.39}$$

entspricht. Betrachtet man den Controller, so werden in einem Superframe bis zu 32 Pakete mit jeweils einem Byte Nutzlast übertragen, was einer Nettodatenrate von

$$\frac{32 \cdot 8Bit}{T_{Superframe}} \approx 136.2\,\text{kBit/s} \tag{4.40}$$

entspricht.

Dieser Datendurchsatz ist im Vergleich zur Bruttodatenrate sehr gering, was hauptsächlich an den vielen benötigten Präambeln, der großen Zahl verschiedener Nutzer innerhalb des kurzen Zeitfensters und der starken Kodierung der Daten liegt. Allerdings wird in einem Automatisierungssystem auch keine hohe Datenrate benötigt. Das Hauptaugenmerk des Systems liegt auf kurzen Latenzzeiten, hoher Robustheit und dichter Netzstruktur. Diese priorisierten Optimierungsziele wirken einem hohen Datendurchsatz entgegen.

4.6.5 Mittlere Reaktionszeit

Die mittlere Reaktionszeit berechnet sich durch die Hälfte der Zykluszeit plus die Übertragungszeit eines Paketes. Für das *TAWASI*-System ergibt sich dadurch:

$$T_{Reaktion} = \frac{T_{Superframe}}{2} + T_{Frame} \approx 985\,\mu\text{s} \tag{4.41}$$

Dies ist ein sehr guter Wert für die drahtlose Automatisierung und erfüllt die geforderte Reaktionszeit von 5 ms.

4.6.6 Telegrammfehlerrate/-latenz

Es lassen sich generell verschiedene Fehlerraten berechnen bzw. simulieren. Da immer nur der aktuellste Wert von Bedeutung ist, wirkt sich jeder Fehler vor allem durch die Erhöhung der Latenzzeit aus, da hier erst die nächste erfolgreiche Übertragung einen neuen Wert liefert.

Die Wahrscheinlichkeit eines fehlgeschlagenen Zyklus kann aus Sicht eines Knotens betrachtet werden. Sie wird erreicht, wenn entweder die Übertragung des *Beacon* oder die Übertragung des Datenpaketes fehlschlägt[18]. Diese ist abhängig von der *SNR*, welche wiederum von der Entfernung der Knoten zum Controller sowie dem elektromagnetischen Rauschniveau in der Umgebung abhängig ist.

Dazu wurden Simulationen durchgeführt, in denen ein Zyklus aus Sicht eines Knotens simuliert wird. Dazu wird zuerst das Senden und Empfangen eines Beacon simuliert. Danach das Senden und Empfangen eines Datenpaketes, wobei ein zufälliger Jitter auf den erwarteten Sendezeitpunkt addiert wird, um die Clock Drift zu simulieren. Sind beide Paketübertragungen erfolgreich, so wird der Zyklus als erfolgreich gewertet. Schlägt eine der beiden fehl, wird der Zyklus als nicht erfolgreich gewertet und der nächste Zyklus eingeleitet.

[18] Dies ist eine strenge Betrachtung, da auch Implementierungen denkbar sind, die bei einer fehlgeschlagenen Beacon-Übertragung den bisherigen Modus beibehalten. Dadurch kann meist das Datenpaket auch bei fehlgeschlagenem Beacon-Empfang übertragen werden.

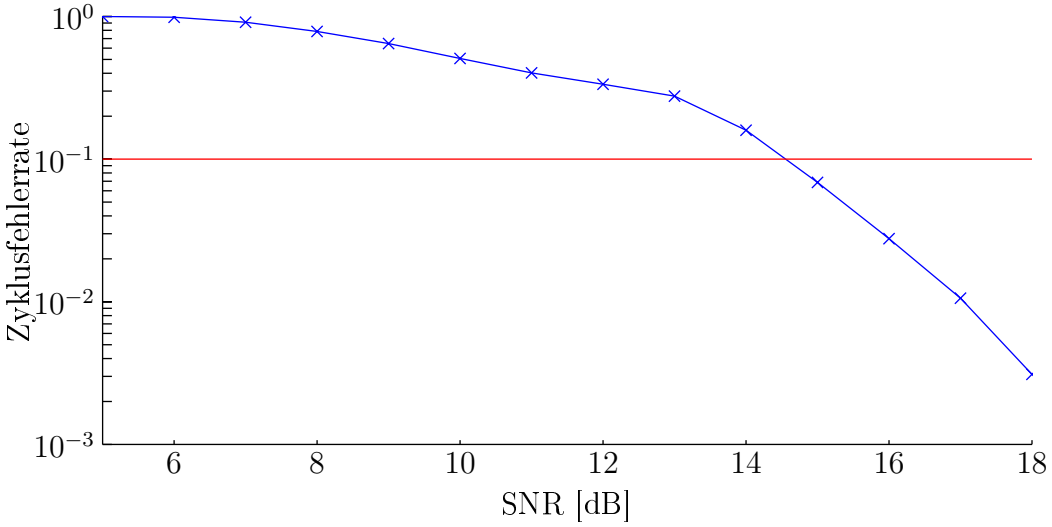


Abbildung 4.36: Fehlerrate des ersten Zyklus für einen Knoten

Abbildung 4.36 zeigt den Verlauf dieser Zyklus-Fehlerrate über die *SNR*. Diese Fehlerrate bezeichnet die Wahrscheinlichkeit für einen Knoten, dass ein Kommunikationszyklus fehlschlägt. Da der Zyklus der Kommunikation kürzer ist als der Zyklus der Applikation, ist die Fehlerrate über mehrere Zyklen hinweg aussagekräftiger.

Wenn diese Wahrscheinlichkeit über mehrere Zyklen hinweg betrachtet wird, dann kann die Wahrscheinlichkeit der zu erwartenden Latenz als Vielfache eines Zyklus angeben werden. Es wird für eine bestimmte Anzahl Durchgänge jeweils die Zahl der benötigten Zyklen bestimmt. Dabei wird weiterhin angenommen, dass bei jedem zweiten Zyklus neue Daten anliegen, also jedes Paket genau zwei Mal gesendet wird, einmal mit positivem *CI* und einmal mit negativem.

Abbildung 4.37 auf der nächsten Seite zeigt eine Simulation von 10^6 Durchgängen bei einer *SNR* von 15 dB. Wie zu sehen ist, bildet die Häufigkeitsverteilung im semilogarithmischen Plot nahezu eine Gerade[19]. Das bedeutet, die Wahrscheinlichkeit der Zyklenzahl ist logarithmisch degressiv.

Da die Simulation des weiteren Verlaufs exponentiell Zeit in Anspruch nimmt wird die Gerade extrapoliert. Um diese Gerade zu bestimmen, wird zuerst die Erfolgswahrscheinlichkeit des ersten Zyklus p benötigt, die im Beispiel für 15

[19]Die Daten lassen durch die paketübergreifende Optimierung der Robustheit eine stärkere Krümmung erkennen, weshalb die Gerade als obere Grenze zu verstehen ist.

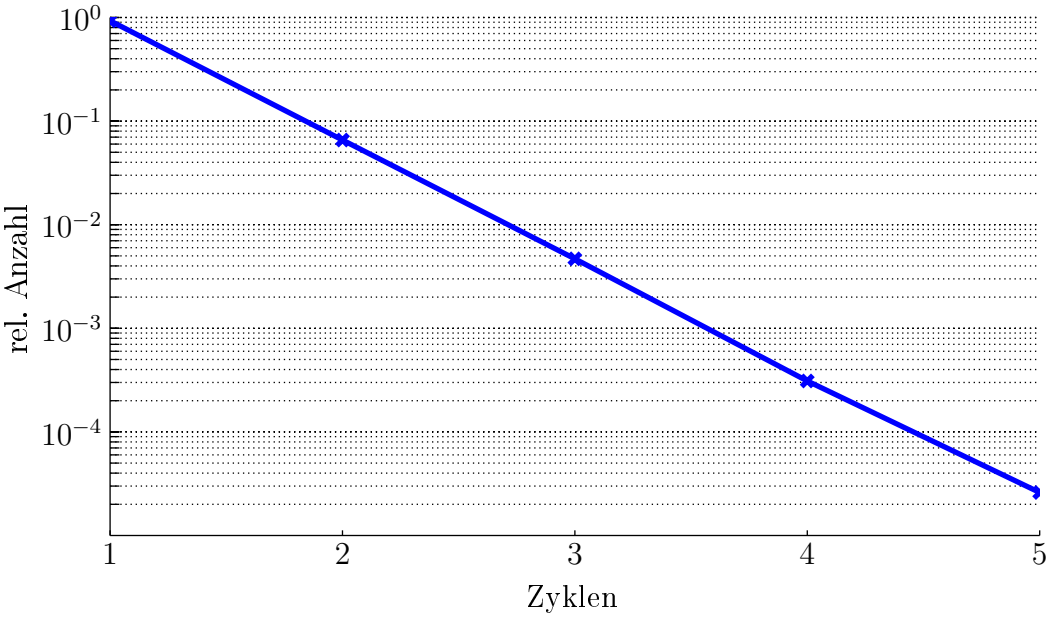


Abbildung 4.37: Häufigkeitsverteilung der benötigten Zyklen

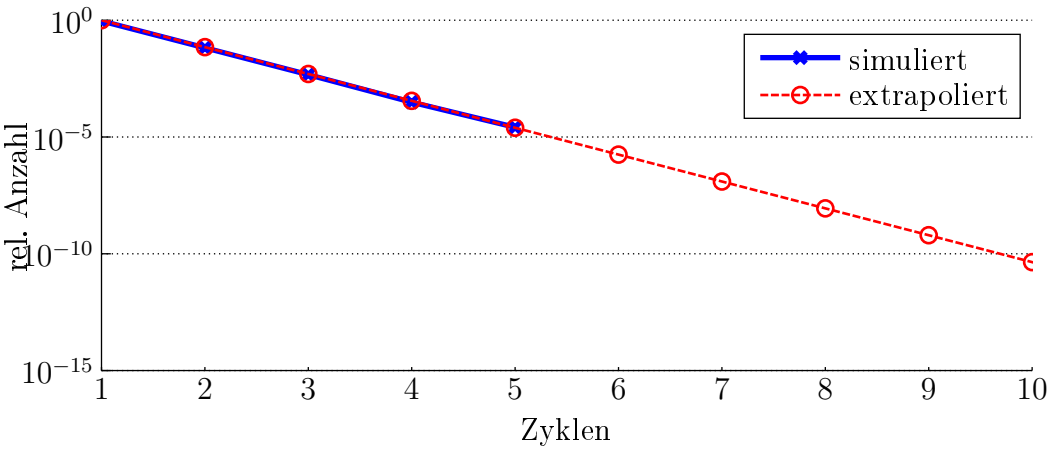


Abbildung 4.38: Extrapolierte Häufigkeitsverteilung der benötigten Zyklen

dB bei 0.929 liegt, wie aus Abbildung 4.36 auf der vorherigen Seite ablesbar ist. Die Fehlerwahrscheinlichkeit des ersten Zyklus beträgt demnach $1-p$. Der weitere Verlauf lässt sich durch

$$y = (1-p)^x \tag{4.42}$$

annähern. Diese Funktion wurde in Abbildung 4.38 zusätzlich zu den Messdaten eingetragen, wobei erkennbar ist, dass sie eine gute Näherung für die

Obergrenze darstellt.

Aus Abbildung 4.38 auf der vorherigen Seite lässt sich ablesen, dass eine Telegrammfehlerrate von 10^{-9} in 8 Zyklen erreicht werden kann, da die relative Anzahl beim neunten Zyklus bereits unter 10^{-9} liegt. Diese 8 Zyklen entsprechen einer Zeit von 15,04 $\approx$ 15 ms. Das bedeutet, dass die Latenz des Systems bei höchstens 15 ms, mit einer Wahrscheinlichkeit von über 93% jedoch unter 2 ms liegt. Dies resultiert aus der Erfolgswahrscheinlichkeit des ersten Zyklus von 0.929 und der Zyklusdauer von 1,88 ms.

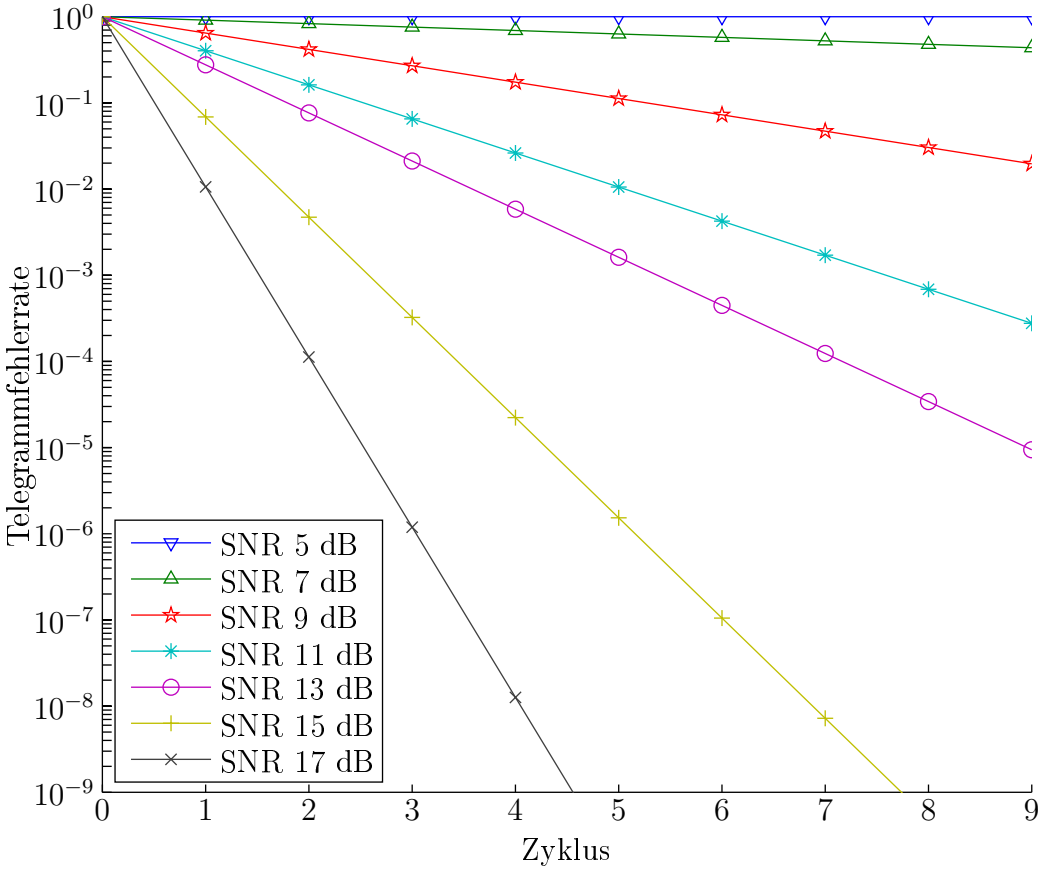


Abbildung 4.39: Extrapolation der Telegrammfehlerrate in Abhängigkeit von Zyklen und SNR

Für weitere *SNR* kann Extrapolation aus Formel (4.42) auf die Fehlerwahrscheinlichkeit des ersten Zyklus von Abbbildung 4.36 auf Seite 157 vorgenommen werden. Abbildung 4.39 zeigt die Ergebnisse für den *SNR*-Bereich von 5-17 dB.

4.6.7 Energiebudget

Mit den Formeln aus Abschnitt 2.2.3.10 auf Seite 51 lässt sich eine Obergrenze für die Reichweite des *TAWASI*-Systems abschätzen. Da durch die sternförmige Topologie alle Knoten innerhalb der Funkreichweite des Controllers liegen müssen, ist die maximale Ausdehnung des Netzes durch die Nutzreichweite des Controllers bedingt.

Dabei ist allerdings die Anzahl der Sensoren und Aktoren bzw. das Verhältnis der beiden Zahlen zueinander von entscheidender Bedeutung. Geht man von einem reinen Sensornetz aus, dann überträgt der Controller nur den *Beacon*. Besteht das Netz jedoch nur aus Aktoren, dann sendet der Controller durchgehend in jedem Zeitschlitz.

Durch die Veränderungen in der *PHY*-Schicht ist die mittlere *PRR* für die Präambel und den Datenteil unterschiedlich, wodurch die Berechnung erschwert wird.

Daher wird für die Betrachtung eine mittlere *PRR* über die Anzahl der zu sendenden Pulse pro Zyklus N_{Cyc}

$$\overline{PRR} = \frac{N_{Cyc}}{T_{Zyklus}} \tag{4.43}$$

berechnet.

Die Berechnung der Anzahl der vom *Controller* zu sendenden Pulse benötigt die Zahl der Pulse im Beacon

$$N_{Beacon} = (N_{Prep} + 6) \cdot 16 \cdot N_{Pburst} + (NL_{Beacon} + 18) \cdot N_{cpb} \tag{4.44}$$

wobei für jedes der N_{Prep} Präambel-Symbole jeweils 16 Präambel-Sub-Symbole mit jeweils N_{Pburst} Pulsen gesendet werden und zusätzlich noch 6 Präambel-Symbole im $SFD \neq 0$ sind. Jedes Datensymbol enthält N_{CPB} Pulse.

Weiterhin lässt sich die Zahl der Pulse pro Daten-Frame durch

$$N_{Pkt} = 16 \cdot N_{Pburst} + (NL_{Frame} + 18) \cdot N_{cpb} \tag{4.45}$$

berechnen, wobei in der Präambel 16 Präambel-Sub-Symbole mit jeweils N_{Pburst} Pulsen übertragen werden und im Datenteil N_{CPB} Pulse pro Symbol.

Die mittlere Pulswiederholungsrate des Controllers wird demnach abhängig von der Anzahl der Aktoren $0 \leq N_{Aktor} \leq 32$ durch

$$\begin{aligned} N_{Cyc}^{Ctrl}(N_{Aktor}) &= \frac{N_{Beacon} + N_{Aktor} \cdot N_{Pkt}}{T_{Zyklus}} \\ &= (N_{Prep} + 6 + N_{Aktor}) \cdot N_{Pburst} \\ &+ ((NL_{Beacon} + 18) + (NL_{Frame} + 18) \cdot N_{Aktor}) \cdot N_{CPB} \end{aligned} \tag{4.46}$$

berechnet. Dadurch lassen sich je nach Anzahl der Aktoren im Netz ein bester Fall $PRR(0)$ ohne Aktoren und ein schlechtester Fall $PRR(32)$, bei dem sich ausschließlich Aktoren im Netz befinden, berechnen:

$$PRR(0) = \frac{N_{Cyc}^{Ctrl}(0)}{T_{Zyklus}} \approx 1,5\,\text{MHz} \tag{4.47}$$

$$PRR(32) = \frac{N_{Cyc}^{Ctrl}(32)}{T_{Zyklus}} \approx 23,9\,\text{MHz} \tag{4.48}$$

Bis zu einer mittleren Pulswiederholungsrate von 18,53 MHz ($\approx PRR(25)$) kann die maximale Leistung von 0 dBm benutzt werden. Dadurch lässt sich mit der Formel (2.27) für die oben angenommene SNR von 15 dB eine Reichweite von

$$\begin{aligned} d_{max} &= \frac{c}{4\pi \cdot f_C} \cdot 10^{-\frac{1}{20}(0+15-87)} \\ d &\approx 15\,\text{m} \end{aligned} \tag{4.49}$$

für Netze mit 25 oder weniger Aktoren berechnen. Dies ist in etwa vergleichbar mit der Reichweite, für die andere drahtlose Automatisierungssysteme, wie z.B. WISA, ausgelegt sind.

Die mittlere Pulswiederholungsrate der Sensoren ist mit

$$\begin{aligned} PRR_{Sensor} &= \frac{N_{Cyc}^{Sensor}}{T_{Zyklus}} \\ &= \frac{N_{Pkt}}{T_{Zyklus}} \\ &= \frac{16 \cdot N_{Pburst} + (NL_{Frame} + 18) \cdot N_{cpb}}{T_{Zyklus}} \\ &= 731\,\text{kHz} \end{aligned} \tag{4.50}$$

niedrig genug, dass in jedem Fall die maximale Leistung von 0 dBm pro Puls verwendet werden kann. Dies erhöht zwar die Empfangswahrscheinlichkeit für Datenpakete von Sensoren, beeinflusst aber nicht die Nutzreichweite des Netzes, da jeder Sensor die *Beacon*s des Controllers empfangen muss.

4.6.8 Vergleich

Eine Gegenüberstellung von *TAWASI* mit den vergleichbaren Systemen aus Kapitel 2 ist in Tabelle 4.5 gezeigt.

Tabelle 4.5: Gegenüberstellung von TAWASI und dem Stand der Technik

	Einheit	AS-I	AS-I (erw.)	IO-Link	WISA	TAWASI
Teilnehmer		32	64	2	120	32
Daten Uplink	Bit pro Knoten	4	4	8	1	8
Daten Downlink	Bit pro Knoten	4	3	8	ca. 4	8
Datenrate Uplink	Bit/s	800	400	8.000	500	4.300
Datenrate Downlink	Bit/s	800	300	8.000	2.500	4.300
Zykluszeit	ms	5	10	2	2	2
mittlere Reaktion	μs	2.700	5.100	2.048	1.088	985
Restfehlerrate[20]		10^{-8}	10^{-8}	10^{-7}	10^{-8}	10^{-14}
Echtzeit-Latenz	ms	n/a	n/a	6	34	15
max. Entfernung	m	100	100	20	5	ca. 15

Kapitel 5

Implementierung eines Hardware-Prototypen

Die vorgestellten Technologien, insbesondere in der Ausprägung *TAWASI*, wurden in Hardware implementiert. Dabei wurde die Implementierung aber möglichst allgemein gehalten, um ohne weitere Neuentwicklungen auch andere Szenarien abdecken zu können. So ist unter anderem auch standardkonformer Betrieb möglich.

Da zum Zeitpunkt der Erstellung der Arbeit keine fertige Implementierung von IEEE 802.15.4a zur Verfügung stand, auf der aufgebaut werden könnte, wurde der Prototyp, mit Ausnahme des analogen *Front-End*, auf einem *FPGA* neu implementiert. Dieses Kapitel gibt einen groben Überblick über die Implementierung, für weitere Details sei auf Anhang A verwiesen.

Abbildung 5.1 auf der nächsten Seite zeigt den schematischen Aufbau meines Prototypen. Dieser ist auf einem Virtex 5 *FPGA* implementiert, der einerseits über eine serielle Schnittstelle mit einem PC verbunden ist, anderseits über einen speziellen Stecker (*exp-Slot*) mit dem analogen *Front-End*.

Auf dem *FPGA* wurde ein minimaler Prozessor implementiert, der die serielle Kommunikation steuert und auf einen *RAM*[1]-Speicher zugreifen kann. Dieses *RAM* wird auch vom *MAC*-Controller genutzt, der wiederum je einen Basisband-Controller zum Senden und Empfangen steuert. Diese sind mit

[1] Random Access Memory - *Speicher mit wahlfreiem Zugriff*

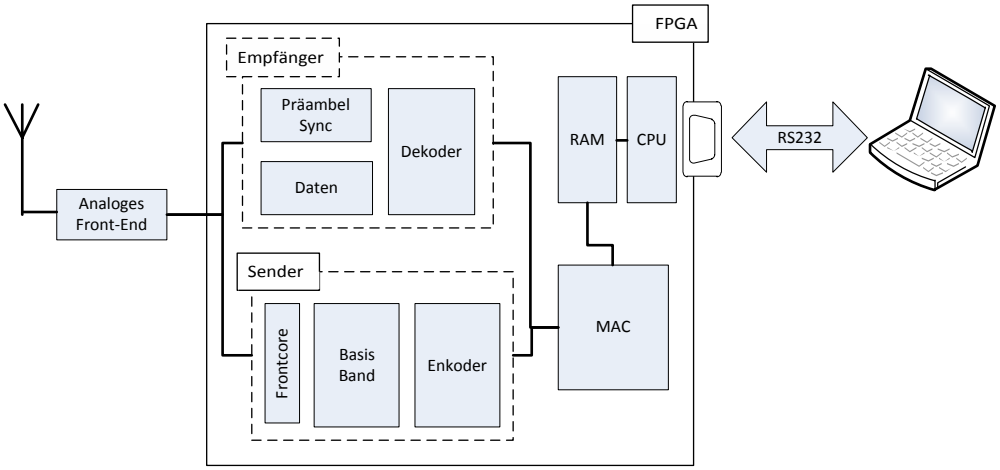


Abbildung 5.1: Schematische Darstellung des Prototypen

dem analogen *Front-End* verbunden, das letztlich die Verbindung zur Antenne darstellt.

5.1 Applikationsschicht

Die Applikationsschicht wird durch eine Software dargestellt, die über RS232 Konfigurationsdaten und Steuerbefehle auf das Board überträgt. Dort werden die Befehle von einer Schaltung (CPU) namens „Pseudo-Prozessor“ ausgewertet. Diese Schaltung greift auf das oben erwähnte gemeinsame *RAM* zu, das zur Speicherung von Konfigurations- und Prozessdaten und zur Kommunikation mit der *MAC*-Schicht dient.

5.2 MAC-Schicht

Die *VHDL*-Implementierung der *MAC*-Schicht besteht hauptsächlich aus drei Komponenten: das oben erwähnte *Dual Port RAM* als Interface zum Prozessor sowie dem sogenannten *Scheduler* und der *MAC-Logikeinheit*. Es ist eine kombinierte *MAC*-Schicht, die anhand der Konfigurationsdaten das entsprechende

Verhalten eines Sensors, Aktors oder Controllers annimmt. In einer eventuellen späteren Serienfertigung wird auch nur eine Art von Chip für mehrere Produkte eingesetzt werden, ebenso wie z.B. auch bei WLAN die *Access Points* und Endgeräte den gleichen Chip verwenden.

5.2.1 Scheduler

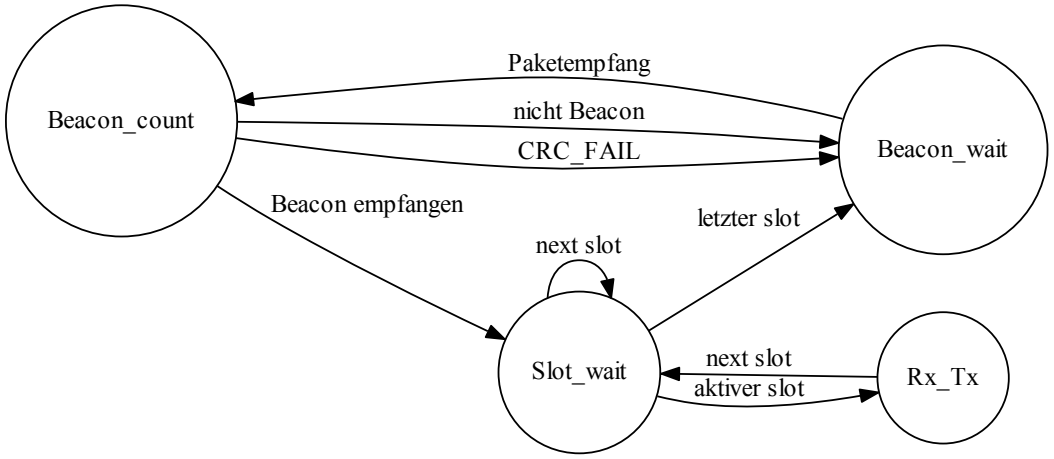


Abbildung 5.2: Zustandsdiagramm des Schedulers

Die Aufgabe des Schedulers ist es, die aktuelle zeitliche Position im *Superframe* festzustellen und dadurch festzulegen, wann der Knoten senden und empfangen muss. Dazu besteht er aus zwei interagierenden Zustandsautomaten, von denen einer den *Beacon*-Zeitschlitz und der andere die Position innerhalb des *Superframe* behandelt.

Abbildung 5.2 zeigt ein Zustandsdiagramm des Schedulers, anhand dessen die Synchronisation der Knoten erklärt werden kann. Nach dem Einschalten befindet sich der Scheduler im Zustand „Beacon_wait“, ein Knoten wartet in diesem Zustand auf den Empfang eines *Beacon*. Wenn ein Knoten in diesem Zustand eine Präambel empfängt, wechselt er in den Zustand „Beacon_count“. In diesem Zustand wird die Zeit ab dem Synchronisationspunkt des empfangenen Paketes gemessen. Falls das empfangene Paket fehlerfrei und ein *Beacon* ist, wird die Länge des *Beacon* berechnet. Mit dieser Länge und dem Synchro-

nisationspunkt kann der momentane zeitliche Versatz zum Ende des *Beacon*-Zeitschlitzes bestimmt werden. Dadurch synchronisiert sich ein Knoten mit dem Controller.

Über die bekannte Länge eines Zeitschlitzes wird die aktuelle Position im *Superframe* festgestellt, wobei sich der Scheduler im Zustand „Slot_wait“ befindet, solange er in diesem Zeitslot nicht senden oder empfangen muss. Im Zustand „Rx_Tx“ werden Datenpakete versendet bzw. empfangen.

5.2.2 MAC-Logikeinheit

Die *MAC*-Logikeinheit steuert den Fluss von Konfigurations- und Prozessdaten. Dazu enthält sie zwei unabhängige Zustandsautomaten.

5.2.2.1 Konfigurations-Zustandsautomat

Der Konfigurations-Zustandsautomat liest die *PHY*-Konfiguration aus dem Speicher. Bei einem Controller wird die *MAC*-Konfiguration ebenfalls aus dem Speicher gelesen, ein Knoten erhält die *MAC*-Konfiguration aus dem Beacon.

5.2.2.2 Datenfluss-Zustandsautomat

Dieser Automat bereitet entweder ein Datenpaket für den Versand vor oder schreibt empfangene Daten in den Speicher. Um die Zeitanforderungen einhalten zu können, wird beim Versenden immer das Paket für den kommenden Zeitschlitz vorbereitet.

5.3 Basisband-Prozessor

Die Basisband-Komponenten für Sender und Empfänger sind voneinander unabhängig.

Abbildung 5.3: Struktur des Senders

5.3.1 Sender

Beim Sender sind Enkodierung und Versand voneinander unabhängig durchführbar. Dadurch kann während dem Versenden eines Paketes bereits das nächste vorbereitet werden. Dies ist nötig, um mehrere Pakete mit verkürzten Präambeln schnell hintereinander senden zu können. Außerdem könnte auf diese Weise eine zukünftige Version des Senders,die mehrere *Front-End*s ansteuert, mit nur einer Komponente zur Kodierung auskommen.

Die Ausgangssignale meines Basisband-Prozessors steuern eine Vorentwicklung von IHP Microelectronics an, eine Komponente namens *Frontcore*. Sie setzt die *Burst*-basierten bzw. Präambel-Sub-Symbol-basierten Signale der Basisband-Komponente auf eine Binärdarstellung des Basisband-Signals auf Pulsebene um.

Auf meinen Vorschlag hin wurde im *Frontcore* zusätzliche Konfigurierbarkeit implementiert, um auch das Aussenden mehrpulsiger Präambeln, sowie eine kleinere Zahl als 16 Pulse pro *Burst*, zu erlauben.

5.3.2 Empfänger

Der Empfänger besteht aus einer Komponente zur Präambel-Detektion und -Synchronisation und einer zur Demodulation und Dekodierung. Dies ermöglicht den Empfang einer Präambel, während die Daten des letzten empfangenen Paketes noch dekodiert werden.

Die Zustandsübergänge der Präambel-Empfangskomponente sind in Abbildung 5.4 auf der nächsten Seite dargestellt. Der Präambel-Empfang benötigt a priori die Information, ob ein einzelnes Präambel-Symbol oder eine reguläre Präambel mit *SFD* empfangen werden muss. Die demodulierten Daten

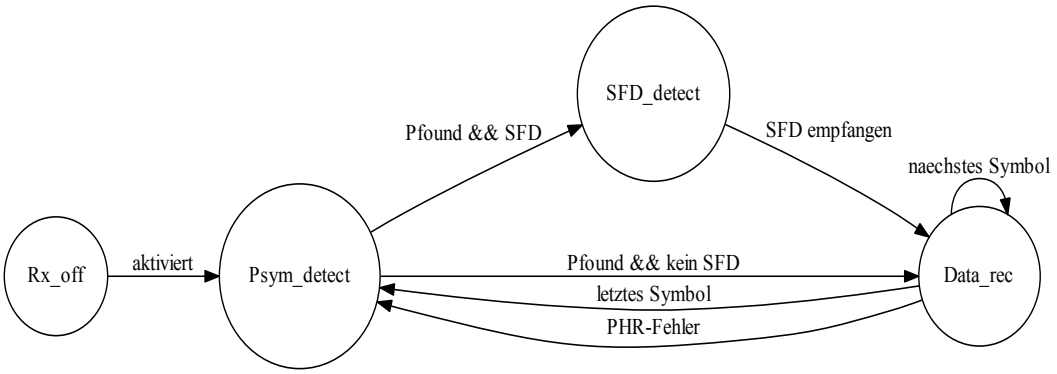


Abbildung 5.4: Zustandsdiagramm des Basisbandempfängers

werden gepuffert und eine weitere Komponente führt die Demodulation und gegebenenfalls Dekodierung der Daten durch.

In Abbildung 5.5 auf der nächsten Seite sind die Zustandsübergänge der Komponente zur Dekodierung sind dargestellt. Sobald das letzte Datensymbol demoduliert wurde, geht die Empfangskomponente wieder in den Zustand zum Präambel-Empfang über. Dieser Aufbau ermöglicht eine schnelle Rückkehr zum Präambel-Empfang bei paralleler Dekodierung der empfangenen Daten. Das ermöglicht zum Beispiel dem letzten Knoten im *Superframe* die Präambel des *Beacon* zu empfangen, während die Daten des Daten-Frame noch dekodiert werden.

5.4 HF-Front-End

Das HF-*Front-End* wurde von IHP Microelectronics entwickelt. Das *Front-End* verwendete das *Mandatory High Band* des Standards IEEE 802.15.4a, mit einer Mittenfrequenz von 7.9 GHz. Zum Zeitpunkt des Verfassens der Arbeit war der Chip der erste verfügbare Transceiver nach IEEE 802.15.4a. Allerdings befand er sich im frühen Prototyp-Stadium. Anstelle von Antennen wurden in dieser Arbeit die *Front-End*s über eine Kabelbrücke zwischen Sender und Empfänger verbunden.

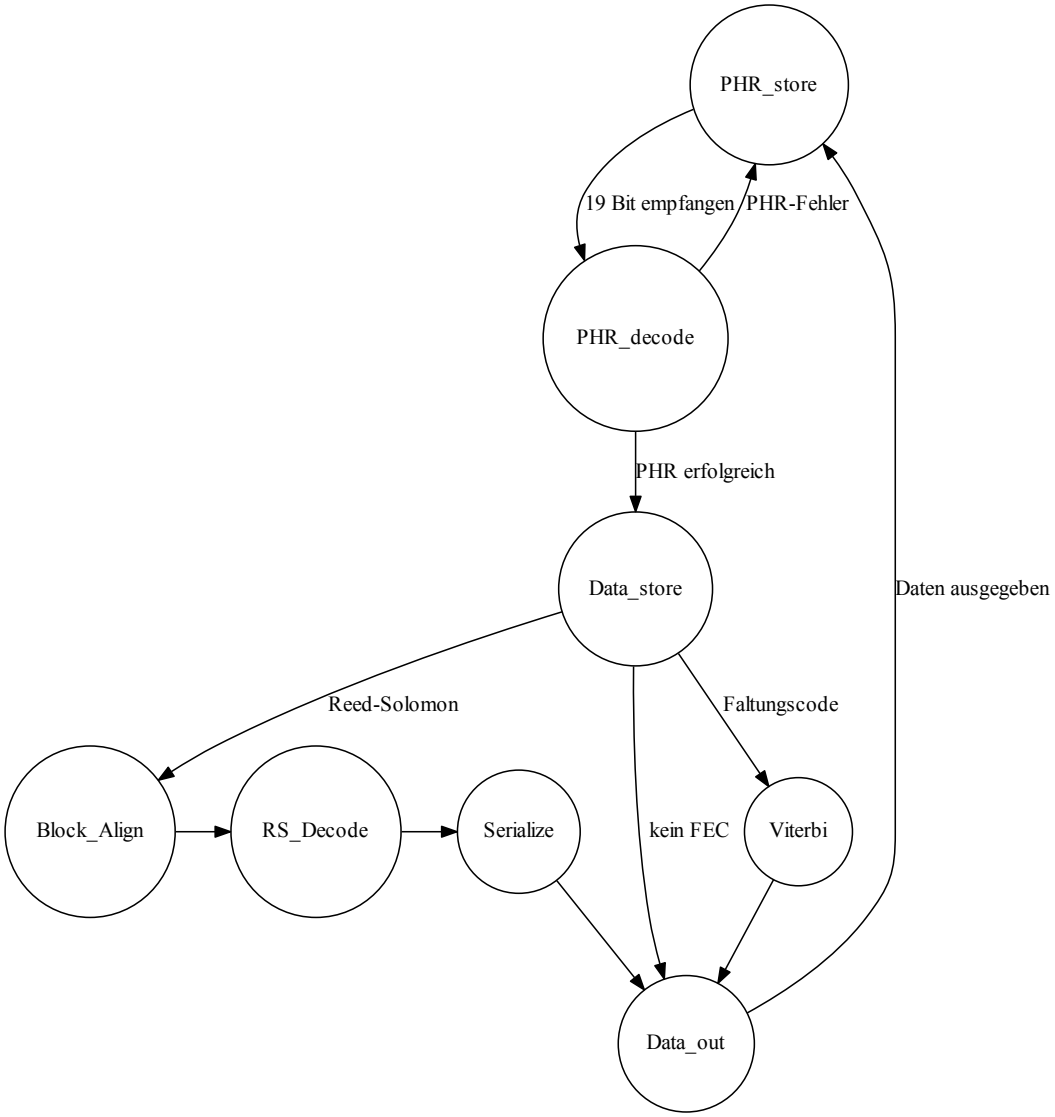


Abbildung 5.5: Zustandsdiagramm der Dekodierung im Basisbandempfänger

5.5 Maßnahmen zur Verifikation der Implementierung

Für die Funktionsweise der oben beschriebenen Implementierung wurden qualitative Verifikationen vorgenommen. Dazu wurden folgende Maßnahmen verwendet:

5.5.1 Hardware-Simulation

Da der Basisband-Controller in *Mathworks SimuLink* implementiert wurde, konnten erste Tests bereits während der Implementierung direkt mit dem Simulator aus Kapitel 3 erfolgen. Zur Verifikation der Funktionsweise von Basisband- und *MAC*-Controller wurden die Hardware-Simulations-Tools *Mentor Graphics ModelSim* und *Xilinx ISim* verwendet.

Dazu schrieb ich eine Software, die es ermöglicht, *MATLAB*-Daten in einer *VHDL*-Testbench zu laden und umgekehrt. Damit konnten Pakete der simulierten Hardware durch das Kanalmodell in MATLAB verändert und in der Hardwaresimulation empfangen werden[2]. Durch diese Methode konnte die Funktionsweise der Hardware-Implementierung von Sender und Empfänger bzgl. Paketformate und Modulation erfolgreich gegen die Kanalmodelle verifiziert werden.

Die Hardware-Simulation ermöglichte durch ihre Flexibilität schnelle Variationen im Aufbau der Schaltung. Sie erlaubte durch ihre geringe Geschwindigkeit allerdings nur einzelne Tests, Messkampagnen zur Bestimmung von Fehlerraten und eine Überprüfung des Zeitverhaltens beim Senden und Empfangen mehrerer Pakete waren damit nicht durchführbar.

5.5.2 „Back-to-Back"-Verifikation im FPGA

Um Tests in Realzeit zu ermöglichen, musste die Schaltung auf *FPGA*-Hardware laufen. Zur Isolation möglicher Fehlerquellen in der parallelen Entwicklung von Digitalteil und analogem *Front-End* habe ich eine Verbindung der digitalen Basisband-Komponenten untereinander eingesetzt, sogenannte *„Back-to-Back"*-Tests.

Für die Hardware-basierten Tests von *MAC*-Schicht und Basisband-Prozessor wurde deshalb eine Komponente entwickelt, um den Integrator und *ADC* des Empfängers zu emulieren, genannt „HF-Emulator". Diese Komponente setzte

[2] Auch das Empfangen und Versenden von Paketen zwischen der Hardware-Implementierung und der in Kapitel 3 auf Seite 65 beschriebenen Simulations-Software ist somit möglich.

das 499,2-MHz-Basisbandsignal in 6-Bit-*ADC*-Werte auf 62,4 MHz um. Dadurch war es möglich, zwei Basisband-Prozessoren auf verschiedenen *FPGA*s zu verbinden. Durch Debug-Signale konnte an einem digitalen Oszilloskop das Zeitverhalten, die Paketstruktur sowie die Synchronisation überprüft und verifiziert werden.

5.5.3 Gesamtsystem

Um das Gesamtsystem zu testen wird der Emulator konnte durch das analoge *Front-End* ersetzt. Da nur ein Exemplar des *Front-End* mit funktionierender Empfangsseite verfügbar war, war nur eine unidirektionale Kommunikation möglich. Die Kommunikation erfolgte dabei auch über eine geschirmte SMA-Kabelbrücke statt der Antennen nur mit einer teilweise sehr hohen, nicht reproduzierbaren, Bit-Fehlerrate.

Der Prototyp mit dem *Front-End*, das empfangen konnte, wurde als Knoten konfiguriert und der andere als Controller. Die Antennenstecker beider HF-*Front-End*s wurden per Kabel verbunden. Da das Basisband-Signal und Debug-Signale beider *FPGA* zusätzlich an ein digitales Oszilloskop angeschlossen wurden, konnte so verifiziert werden, dass *Beacon*s empfangen und verarbeitet wurden.

Dazu wurden die Basisbandsignale[3] beider Knoten auf einem digitalen Oszilloskop dargestellt und beobachtet, ob bei Änderung der *MAC*-Adresse das Datenpaket im richtigen Zeitschlitz im Superframe gesendet wird, wodurch sich der zeitliche Abstand zum Beacon ändern muss.

Quantitative Messungen zur Bestimmung der Fehlerraten waren wegen der eingeschränkten Testbedingungen nicht möglich.

[3]Für eine direkte Darstellung des HF-Signals stand keine ausreichende Laborausstattung zur Verfügung

5.6 Zusammenfassung

Eine Ausprägung der vorgestellten Technologien wurde in Hardware implementiert. Die Funktion von Basisband-Prozessor und *MAC*-Schicht konnte experimentell bestätigt werden. Mit den derzeit verfügbaren Testbedingungen konnten jedoch noch keine quantitativen Messungen mit dem Gesamtsystem durchgeführt werden, welche die Theorien und Simulationen zur Verbesserung gegenüber dem Stand der Technik bestätigen.

Diese Bestätigung verbleibt somit für zukünftige Arbeiten, sobald entsprechende Hardware verfügbar ist. Bis dahin stützen sich die Theorien auf die Simulationen.

Kapitel 6

Zusammenfassung

6.1 Zusammenfassung

In dieser Dissertation wurde die Tauglichkeit von *IR-UWB* für die drahtlose Kommunikation in der Sensor/Aktor-Ebene der Fertigungsautomatisierung evaluiert. Dazu wurden bestehende Protokolle der drahtlosen Automatisierung und der *IR-UWB*-Kommunikation untersucht. Dabei wurde ein neues Funksystem, basierend auf einer optimierten *PHY*-Schicht auf Basis des Standards IEEE 802.15.4a und einer *MAC*-Schicht auf Basis eines Entwurfs für den Standard IEEE 802.15.4e entworfen, simuliert und implementiert.

6.2 Ergebnisse

Es wurden Methoden zur Ermittlung der maximal zulässigen Sendeleistung und damit zur Abschätzung der Reichweite von *IR-UWB*-Kommunikation erstellt. Dabei wurde festgestellt, dass die Reichweite stark von der Pulswiederholrate abhängig ist, welche die Datenrate bedingt. Es wurde erkannt, dass zur Nutzung der maximalen Leistung pro Millisekunde eine Obergrenze für die mittlere Pulswiederholrate von 18 MHz existiert.

Außerdem wurde eine simulative Evaluation verschiedener Modi des Standards IEEE 802.15.4a vorgenommen. Anhand dieser Simulationen konnte der Ein-

fluss der Parameter, die in den Modi von IEEE 802.15.4a festgelegt sind, auf Fehlertypen und deren Häufigkeit untersucht werden. Dabei wurde die Präambel als größte Schwachstelle der Übertragungen identifiziert.

Es wurden mehrere Verbesserungen an der Robustheit des standardisierten *PHY*-Protokolls und am Design eines niederratigen, nichtkohärenten Energie-Detektions-Empfängers vorgestellt. Diese Verbesserungen optimieren die Erkennung der Präambel für nichtkohärente Empfänger und reduzieren Fehler in der Daten-Demodulation bei inkorrekter Synchronisation. Durch diese Verbesserungen konnte ein Gewinn von 10 dB erzielt werden.

Es wurde eine Methode gefunden, mit einem Energie-Detektions-Empfänger Phaseninformationen von gleichphasigen Pulsketten zu ermitteln. Dadurch wurde, aufgrund eines ternären Codes, eine robustere Präambel-Erkennung als bisher möglich. Außerdem wurde eine Methode zur Generierung von Soft-Bit-Informationen bei *BPM*-modulierten Daten entwickelt, was auch zur effizienteren Dekodierung von *FEC*-Codes verhilft, wodurch kürzere Pakete ermöglicht werden.

Es wurde eine Cross-Layer-Technik für Automatisierungsnetze vorgestellt, die repetitive Übertragungen ausnutzen kann, um durch Soft-Bit-Kombination Paketverluste zu vermindern. Weiterhin wurde gezeigt, dass bei Zykluszeiten im Bereich von < 10 ms eine zweistufige Synchronisation auf Superframe- und Paketebene erfolgen kann, wodurch verkürzte Präambeln benutzt werden können.

Diese Verbesserungen ermöglichten ein optimiertes *IR-UWB*-System für die drahtlose Fertigungsautomatisierung. Dieses auf Standardprotokollen basierende System erreicht bei einer maximalen Telegrammfehlerrate von 10^{-9} bei 15 ms eine kürzere Echtzeitschranke für 32 Teilnehmer als es der bisherige Stand der Technik mit proprietären Protokollen ermöglicht. Darüber hinaus ermöglicht es eine höhere Datenrate pro Teilnehmer als bisherige Systeme von $\approx 4.3\,\mathrm{Kbit/s}$ und eine kürzere mittlere Reaktionszeit in Up- und Downlink von jeweils 985 μs.

Das vorgestellten Optimierungen und das Gesamtsystem wurden per Simulation verifiziert. Ein Prototyp von *MAC*-Schicht und Basisband-Prozessor dieses

Systems wurde in Hardware implementiert, dessen Funktionsweise qualitativ verifiziert wurde. Für vier Erfindungen dieser Dissertation wurden Patente angemeldet und es wurden fünf Konferenzbeiträge veröffentlicht.

6.3 Diskussion

Die vorgestellten Metriken des Systems konnten nur mittels numerischer Simulation bestimmt werden. Das nach meinem Wissen einzige existierende analoge *Front-End* befand sich noch in einem frühen Prototyp-Stadium, in dem keine realistische Messung in Hardware möglich war.

Die Aussagekraft der Simulationen hängt von der Übereinstimmung des Modells mit der Wirklichkeit ab. Dabei sind vor allem die Kanalmodelle entscheidend. Die verwendeten Kanalmodelle wurden aufgrund hunderter reeller Messungen erstellt, gelten als die offiziellen Kanalmodelle der IEEE und wurden als solche in der Wissenschaftsgemeinde diskutiert und akzeptiert. Den Kanalmodellen kann darum eine hohe Glaubwürdigkeit beigemessen werden.

Trotzdem könnten gemessene Werte von den simulierten abweichen, allerdings ist nicht zu erwarten, dass dies fundamentale Auswirkungen auf das System hätte.

6.4 Zukünftige Arbeiten

Diese Arbeit eröffnet ein weites Feld für zukünftige Forschung.

Nachfolgende Arbeiten könnten weitere Konfigurationen neben *TAWASI* beinhalten sowie höhere Funktionen der *MAC*-Schicht, besonders eine autonome Adressierung, behandeln. Auch die Operationen wie Beitreten und Verlassen eines Netzwerks erfordern weitere Überlegungen, um die Leistung des Systems durch diese Operationen nicht zu verringern. Weiterhin könnte eine Adaptionsschicht zwischen *AS-Interface* und *TAWASI* vermitteln, im besten Falle transparent.

Der Sicherheits- bzw. Vertrauensaspekt wurde in dieser Arbeit nicht betrachtet. Trotz der geringen Reichweite der *PHY*-Schicht sind Angriffe denkbar, die dann direkt die industrielle Infrastruktur betreffen könnten. Deshalb sind weitere Überlegungen zur Sicherheit und Vertraulichkeit der Daten auf dem Funknetz notwendig.

Die Anwendung des Systems in anderen Gebieten der Automatisierung eröffnet viele Möglichkeiten zukünftiger Forschungsarbeit.

Anhang A

Details der Implementierung des Hardware-Prototypen

Ergänzend zum Kapitel 5 stellt dieses Kapitel weitere Details des Prototypen vor.

Die Implementierung des Prototypen besteht aus zwei physikalischen Komponenten:

- Einem frühen Prototypen eines impulsbasierten *UWB*-Transceivers von IHP Microelectronics. Dieses *Front-End* wird in Abschnitt A.5 auf Seite 191 genauer vorgestellt.

- Entwicklungs-Boards mit VIRTEX-5 *FPGA* vom Typ XC5VSX95T von Xillinx. Auf diesen wurde der Basisband-Prozessor, die *MAC*-Schicht und der sog. Pseudo-Prozessor implementiert.

Dieses Board ist über eine serielle Schnittstelle mit einem PC verbunden. Von diesem PC werden über eine Software Konfigurationsdaten und Prozessdaten auf den Prototypen übertragen. Es werden auch Debug-Informationen vom Prototypen an den PC gesendet.

A.1 Software

Über ein *GUI*[1] kann eine Konfiguration für die *PHY*-Schicht erstellt und Prozessdaten in den Speicher des *FPGA* geschrieben und ausgelesen werden. Die Software ist in C# implementiert und mit jedem seriellen Port nutzbar. Zusätzlich wurde eine Kommandozeilen-Software in Python implementiert. Ergänzend zu den beiden genannten Möglichkeiten der Ansteuerung können die Funktionen des so gennannten Pseudo-Prozessors auch mit einer Terminalsoftware angesprochen werden.

A.2 Pseudo-Prozessor

Die Applikationsschicht wird als Pseudo-Prozessor bezeichnet. Sie steht im Prototypen stellvertretend für einen Prozessor bzw. Mikrocontroller, auf dem die Applikationsschicht in Software implementiert ist. Der Pseudo-Prozessor erhält Befehle über die eine serielle Schnittstelle des *FPGA* und arbeitet, entsprechend den Befehlen, auf einem *dual-Port-RAM*. Dieses wird gemeinsam mit der *MAC*-Schicht genutzt und enthält die Konfiguration der *PHY*-Schicht sowie die Prozessdaten.

A.2.1 Befehle

Der Befehlssatz des Prozessors umfasst vier Befehle, die über die serielle Schnittstelle empfangen werden und jeweils ein bestimmtes Verhalten auslösen. Jeder Befehl besteht aus einem Byte und hat mindestens zwei Parameter von jeweils einem Byte Größe. Diese Parameter müssen zwar immer übertragen werden, sind aber nicht immer belegt.

1. Schreiben
   Der Schreibbefehl wird durch ein Byte mit dem Wert 0x41 (ASCII 'A') eingeleitet. Es folgt die Speicheradresse, an die geschrieben werden soll, und darauf folgt die Länge des zu schreibenden Datenwortes in Bytes.

[1] Graphical User Interface - *Graphische Benutzeroberfläche*

Anschließend folgen die Bytes, die geschrieben werden sollen. Der Pseudo-Prozessor schreibt die Bytes an die angegebene Speicheradresse und gibt keinen Rückgabewert aus.

2. Lesen
   Der Lesebefehl, Wert 0x42 (ASCII 'B'), ist genauso aufgebaut wie der Schreibbefehl. Er liest die angegebene Anzahl Bytes ab der angegebenen Speicheradresse und gibt sie über die serielle Schnittstelle aus.

3. Schreiben und Zeitmessen
   Dieser Befehl, Wert 0x43 (ASCII 'C'), ähnelt dem Schreibbefehl, allerdings wird zusätzlich ein Counter gestartet. Er wird gestoppt, sobald der Knoten ein positives *ACK* für den Versand der Daten erhalten hat. Das ist der Fall, wenn im *Group ACK* im *Beacon* das Bit für diesen Knoten auf '1' gesetzt ist. Diese Funktion erlaubt die Messung einer Latenz vom Zeitpunkt neu eingetroffener Daten am Sensor bis zum Empfang einer Bestätigung über den Erhalt dieser Daten vom *Controller*.

4. Debug-Modus
   Dieser Befehl, Wert 0x44 (ASCII 'D'), hat nur einen Parameter, der zweite wird ignoriert (es müssen aber aus Kompatibilitätsgründen trotzdem 3 Bytes gesendet werden). Dieser Parameter stellt den Debug-Modus an oder aus. Im Debug-Modus werden bestimmte Informationen bei Zustandsänderungen der tieferen Ebenen Byte-weise an den PC gesendet.

Diese Funktionen würde in einem endgültigen System eine Software auf einem Mikrocontroller übernehmen. Der Prototyp benutzt keinen Mikrocontroller, da dies weiteren lizenzrechtlichen, finanziellen sowie höheren Implementierungsaufwand bedeutet hätte. Für eine Evaluation der *MAC*- und *PHY*- Schichten ist die verwendete Implementierung ausreichend. Der Pseudo-Prozessor lässt sich durch die Anbindung über ein *shared RAM* ohne Aufwand durch beliebige Mikrocontroller ersetzen. Außerdem lassen sich mit dem oben erwähnten Pseudo-Prozessor schneller Debug-Informationen aus dem Chip übertragen als bei Verwendung eines Mikrocontrollers.

A.3 MAC-Schicht

Das Verhalten der *MAC*-Schicht ist unterschiedlich bei Knoten (also Sensoren und Aktoren) und dem Controller. Ich habe entschieden, nur eine Logik zu erstellen, die anhand der Konfigurationsdaten das entsprechende Verhalten annimmt. Im Folgenden werden die drei wichtigsten Komponenten der *MAC*-Schicht vorgestellt: das Interface zum oben erwähnten Prozessor sowie der sogenannte *Scheduler* und die *MAC-Logikeinheit*.

A.3.1 CPU-Interface

Das Interface zum Prozessor besteht aus einem *Dual-Port-RAM*, also einem gemeinsamen Speicher mit je einem Interface zum Prozessor und zur *MAC*-Schicht. Dabei wurde das CPU-seitige Interface mit einer Datenwortbreite von 32 Bit ausgelegt, da 32 Bit die derzeit üblichste Architektur für echtzeitfähige Mikrocontroller darstellt. Das *MAC*-seitige Interface benutzt einen 8 Bit breiten Datenbus.

Der Speicher wird sowohl für die Konfiguration als auch für Prozessdaten genutzt. Die Adressierung des Speichers ist wie folgt festgelegt: Er ist in 128 Blöcke zu je 128 Byte[2] für Prozessdaten von bis zu 127 Knoten[3] unterteilt. Die *MAC*-Adresse gibt den 128-Byte-Block an, der für die Daten verwendet wird. Die ersten 128 Byte werden für die Konfigurationsdaten genutzt.

Das *RAM* besitzt also eine Größe von $128 \cdot 128\,\text{Byte} = 16\,\text{KByte}$. Für Anwendungen wie *TAWASI* könnte der Speicher auch erheblich kleiner gewählt werden. Für den *FPGA*-basierten Prototypen wurde allerdings das Ziel der Flexibilität über die Optimierung gestellt. So ist jede mögliche Konfiguration realisierbar und die Größe des Speichers stellt auf dem verwendeten *FPGA* kein Problem dar. Die Konfigurationsdaten bestehen aus drei Teilen, nämlich der *MAC*-Adresse des Knotens, den *PHY*-Parametern sowie, im Falle des Controllers, Informationen über den Aufbau des Netzes. Diese werden in 16 bzw. 8 Byte gespeichert, wie in Tabelle A.1 angegeben.

[2] Obergrenze der möglichen Bytes pro Paket in IEEE 802.15.4a
[3] Obergrenze der Knoten pro Netz, gegeben durch Draft IEEE 802.15.4e

Tabelle A.1: Adressierung des Konfigurationsblocks

Adresse	**Zuordnung**	**Beschreibung**
0x00	alle	*MAC*-Adresse
0x01	Konfiguration	N_{Hops}
0x02	Konfiguration	Präambel-Wiederholungen beim Datenpaket (*Lower Byte*)
0x03	Konfiguration	Präambel-Wiederholungen beim Datenpaket (*Upper Byte*)
0x04	Konfiguration	Länge der Bit-Maske des Scramblers
0x05	Konfiguration	N_{CPB}
0x06	Konfiguration	Anzahl Samples pro Hopping-Position
0x07	Konfiguration	L-Spreading (7 Bit) & Sensor-Flag
0x08	Konfiguration	N_{Pburst} (6 Bit) & *SFD*-Flag & Kodierungs-Flag
0x09	Controller	*MAC*-Flags, wie im *Beacon*
0x0A	Controller	Gateway ID
0x0B	Controller	Gateway-Konfigurations-ID
0x0C	Controller	Länge des *BTS*, wie im *Beacon*
0x0D	Controller	Anzahl der *BTS*, wie im *Beacon*
0x0E	Controller	Anzahl der *BTS* für Sensoren
0x0F	Controller	Präambel-Wiederholungen beim *Beacon* (*Lower Byte*)
0x10	Controller	Präambel-Wiederholungen beim *Beacon* (*Upper Byte*)

Dabei entscheidet die *MAC*-Adresse zusammen mit dem Sensor-Flag über das Verhalten der *MAC*-Schicht. Die *MAC*-Adresse 0x00 ist für den *Controller* reserviert. Jede andere *MAC*-Adresse nimmt das Verhalten eines Knotens an. Das Sensor-Flag entscheidet dabei, ob der Knoten als Sensor oder Aktor reagiert. Das CPU-Interface besitzt ein Eingangssignal, das die Konfiguration aus dem Speicher ausliest und daraufhin die *PHY*- und *MAC*-Schaltungen zurücksetzt.

A.3.2 Scheduler

Der Scheduler stellt die aktuelle zeitliche Position im Superframe fest, aktiviert den Empfänger und löst den Sendevorgang aus. Dazu besteht er aus zwei interagierenden Zustandsautomaten, wie in Abbildung 5.2 auf Seite 165 gezeigt. Der Scheduler benötigt mehrere Eingangssignale, wie z.B.:

- die Zeit vom SFD des *Beacon* bis zum Ende des *Beacon* in Taktzyklen
- die Länge eines *BTS* in Taktzyklen
- den Typ des lokalen Transceiver (Controller oder Knoten)
- die Rolle (Sensor oder Aktor) des Knotens

Dabei wartet ein Knoten auf den Empfang eines *Beacon* und zählt die (bekannte) Zeit vom Empfang eines Pakets (genauer des *SFD*) bis zum Ende des *Beacon* ab. Dadurch synchronisiert sich der Knoten auf den Superframe. Nach dem *Beacon*-Frame beginnt der zweite Zustandsautomat die *BTS* anhand ihrer Länge abzuzählen und setzt dabei die Ausgangssignale entsprechend des aktuellen *BTS* und der Eingangssignale. Diese Ausgangssignale sind:

- Die Nummer des aktuellen Zeitschlitzes
- Die Empfangsbereitschaft
- Ein Impuls zum Auslösen des Sendens (nach dem Guard-Intervall)
- Ein Impuls bei neuem Zeitschlitz

Anhand der Anzahl der Zeitschlitze pro Superframe geht der Scheduler nach jedem Superframe selbstständig wieder in den Beacon-Modus über. Bei einem Controller signalisiert der Scheduler, dass ein Beacon auszusenden ist, anstatt auf den Beacon zu warten.

A.3.3 MAC-Logikeinheit

Die *MAC*-Logikeinheit ist die Kernkomponente der Implementierung. Sie enthält zwei abhängige Zustandsautomaten, einen für die Konfiguration und einen zur Datenverarbeitung. Weiterhin besitzt sie zwei Komponenten, welche

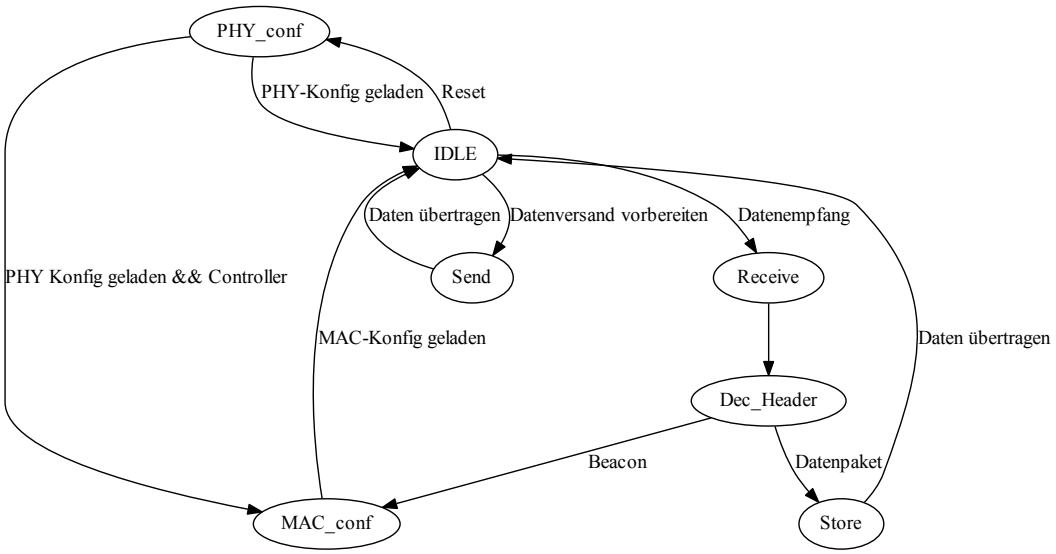


Abbildung A.1: Zustandsdiagramm der MAC-Logikeinheit

die Länge des *Beacon* und die Länge eines *BTS* anhand der Konfigurationsdaten berechnen und dem Scheduler zur Verfügung stellen. Diese Berechnung ist komplett in Hardware implementiert, da sie innerhalb des Schutzintervalls eines Zeitschlitzes erfolgen muss, welches in der *TAWASI*-Konfiguration ca. 11 μs lang ist.

A.3.3.1 Konfigurations-Zustandsautomat

Nach einem Reset liest dieser Zustandsautomat die Konfiguration aus dem Speicher. Er verbleibt dann in einem Reaktionszustand (genannt IDLE). In diesem Zustand reagiert er auf die Signale des Schedulers und steuert dadurch den Zustand des Datenfluss-Zustandsautomaten entsprechend der aktuellen Position im Superframe. Außerdem wird bei einem Controller das Aussenden des *Beacon* angestoßen.

Wird ein Paket empfangen, dann wird der *MAC*-Header ausgewertet. Beim Empfang der Daten eines *Beacon* werden diese verarbeitet, indem die Konfiguration ergänzt wird und das *Group ACK* ausgewertet wird. Bei Datenpaketen wird der Datenfluss-Zustandsautomat auf Empfang gestellt.

A.3.3.2 Datenfluss-Zustandsautomat

Dieser Automat bereitet entweder den Datenversand vor oder schreibt empfangene Daten in den Speicher. Anhand des aktuellen Zeitschlitzes, der über den Scheduler ermittelt wird, wird durch diesen Automaten gegebenfalls die Vorbereitung eines Datenpakets für den nächsten Zeitschlitz ausgelöst. Bei einem Datenempfang werden die Prozessdaten in den Speicher geschrieben.

A.3.4 Sende- und Empfangseinheit

Für das Senden und Empfangen von Paketen stellt die *MAC*-Schicht Komponenten zur Verfügung, welche die *CRC*-16-Prüfsumme parallel berechnen und, im Falle des Empfängers, die Verifikation vornehmen. Diese Komponenten führen auch eine Konvertierung zwischen Bytes und 6-Bit-Symbolen durch.

A.4 Basisband-Prozessor

Die ursprüngliche Implementierung des Basisband-Prozessors entstand im Rahmen einer Diplomarbeit unter meiner Betreuung [38]. Die Implementierung wurde mit dem Design-Tool „Synplicity DSP“ vorgenommen, das einen graphischen Entwurf der Schaltung in der Simulationsumgebung „Mathworks Simulink“ erlaubte. Das Tool generierte daraus Architektur-unabhängigen *VHDL*-Code[4]. Diese Implementierung wurde an mehreren Stellen zur besseren Konfigurierbarkeit geändert und ein Teil der Implementierung wurde aus Performance-Gründen in nativem *VHDL* reimplementiert.

A.4.1 Sender

Der Sender besteht aus mehreren Komponenten, deren Funktionsweise stark von den eingestellten *PHY*-Parametern abhängt. Er kann ein Paket senden

[4]Das ist *VHDL*-Code, der für *FPGA*s verschiedener Hersteller und sogar für ASICs verwendet werden kann.

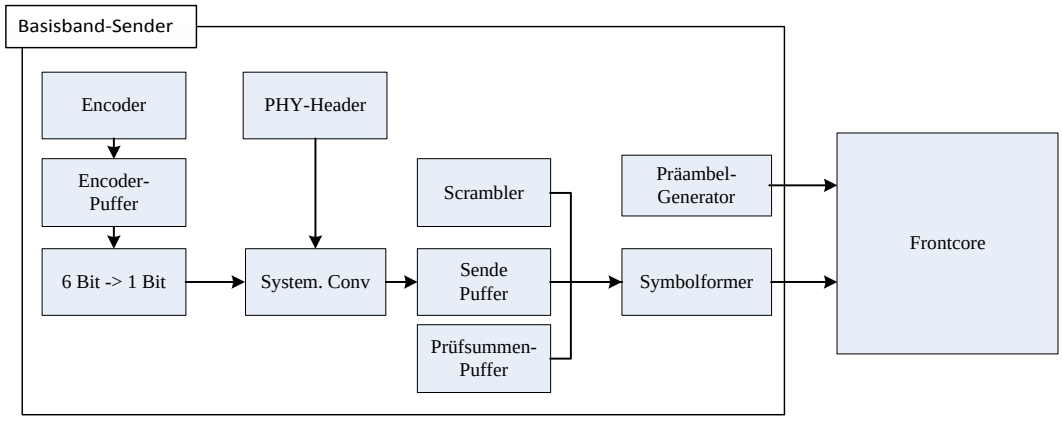


Abbildung A.2: Struktur des Basisbandsenders

und gleichzeitig ein neues Paket vorbereiten. Dabei steuert ein zentraler Zustandsautomat den Vorverarbeitungs- und Sendevorgang. Dazu wird erst ein Paket vorbereitet, indem die Nutzdaten kodiert werden. Die kodierten Nutzdaten werden dann zusammen mit dem *PHY*-Header in einem Sendepuffer vorgehalten. Sobald der Sendezeitpunkt erreicht ist (was dem Sender durch ein Eingangssignal mitgeteilt wird), wird eine Präambel gemäß der Konfiguration gesendet und schließlich das Paket aus dem Sendepuffer gesendet. Gleichzeitig kann schon ein neues Paket kodiert werden.

A.4.1.1 PHY-Header

Die Informationen für den *PHY*-Header werden als Eingangssignale an den Sender angelegt. Nur die Länge der Nutzlast inklusive Kodierung und der *SECDED*-Code muss berechnet werden. Diese Daten werden in den Sendepuffer geschrieben, sobald das vorherige Paket komplett gesendet wurde. Sobald der *PHY*-Header im Sendepuffer vorliegt, wird ein Signal gegeben, das die Verarbeitung der Daten freigibt.

A.4.1.2 Datenkodierung

Die Daten werden dem Sender mit einem 6 Bit breiten Bus zugeführt[5]. Zusätzlich muss die Anzahl der Bits in der Nutzlast bekannt sein. Abhängig von dem Kodierungs-Flag der *PHY*-Konfiguration können die Daten mit dem systematischen *Reed-Solomon-Code* versehen werden. Dazu werden diese 6-Bit-Symbole in Gruppen zu je 55 Symbolen in den Reed-Solomon-Encoder geleitet. Sollte die letzte Gruppe kleiner als 55 Symbole (also 330 Bit) sein, so wird sie mit Füll-Bits auf diese Größe gebracht. Die Ausgabe des Reed-Solomon-Encoder wird in einem FIFO-Puffer zwischengespeichert. Dabei werden die eventuell angefügten Füllbits wieder entfernt.

Sobald der Sendepuffer bereit ist für das neue Paket, also das vorangegangene Paket komplett gesendet und der *PHY*-Header im Sendepuffer abgelegt wurde, geht die Verarbeitung weiter. Der Inhalt dieses Puffers wird über einen Serialisierer von 6-Bit-Symbolen in Einzel-Bits gewandelt und im Sendepuffer gespeichert.

Im Sendepuffer laufen diese einzelnen Bits (sowohl bei den Daten als auch beim *PHY*-Header durch einen systematischen Faltungs-Code, der zusätzlich zu jedem Daten-Bit eine Prüfsumme ausgibt. Diese Prüfsummen landen in einem zweiten, synchronen Sendepuffer.

A.4.1.3 Präambel-Generator

Sobald der Sender das Signal für den Kanalzugriff erhält, wird eine Präambel gemäß der Konfiguration ausgesendet. In der ursprünglichen Version des Senders wurde die Präambel auf Puls-Ebene erzeugt, wozu ein großer Block der Schaltung mit einem Takt von 499,2 MHz laufen musste.

Dies ist mit dem benutzten *FPGA* nicht realisierbar, deshalb wurde ein Redesign vorgenommen. Danach wurde die Präambel in Blöcken zu 16 T_C erzeugt, wodurch sich die Frequenz auf 31,20 MHz senkte. Die Serialisierung auf ein 499,2-MHz-Signal findet dann erst im so genannten *Frontcore* statt. Allerdings sind dadurch nicht mehr alle Konfigurationen möglich.

[5]Da ein (63,55)-Code verwendet wurde, werden 6-Bit-Symbole benötigt.

Dazu wird der betreffende Präambel-Code aus einem ROM geladen und gegebenenfalls an das *L-Spreading* angepasst. Dieses Präambel-Symbol wird abhängig von den Präambel-Wiederholungen in der Konfiguration wiederholt. Wenn das *SFD*-Flag gesetzt ist, wird auch noch der *SFD* generiert. Genau einen Taktzyklus vor Abschluss der Präambel wird durch ein Signal der Datenversand ausgelöst.

A.4.1.4 Scrambler

Für das Versenden der Daten muss die *THS* bekannt sein. Dazu wird eine Pseudo-Zufallsfolge benötigt, die durch den sogenannten *Scrambler* erstellt wird. Dieser *Scrambler* ist ein *LFSR*, das mit einem *Seed* initialisiert wird, der aus dem Präambel-Symbol errechnet werden kann.

Der Zustand des *LFSR* definiert für jedes Symbol zwei Vorgaben: die Hopping-Position, die für dieses Symbol verwendet wird, und einen Bit-Vektor der Länge N_{CPB} für das *Burst Scrambling*.

Da die Anzahl der Hopping-Positionen variabel ist, muss die Ermittlung der aktuellen Hopping-Position an diesen Parameter angepasst werden. Dies geschieht über ein binäres UND der Bits aus der Zufallsfolge mit einer Bit-Maske. Die Länge dieser Bit-Maske lässt sich durch $log_2(N_{Hops})$ berechnen. Da sich diese Berechnung nicht effizient in Hardware realisieren lässt, wird die Länge der Bit-Maske als Parameter übergeben.

A.4.1.5 Symbolformer

Mit der Hopping-Position für das aktuelle Symbol und den Bits aus den Sendepuffern können jetzt die Symbole gesendet werden. Ebenso wie bei der Präambel wurde auch hier in der ursprünglichen Version der Implementierung das Basisbandsignal direkt auf einer Frequenz von 499,2 MHz erzeugt. Dabei wurde jede Hopping-Position abgezählt und an der betreffenden Hopping-Position schließlich ein *Burst* eingefügt, wobei auch das *Burst Scrambling* mit der Zufallsfolge vorgenommen wurde.

Durch das oben erwähnte Redesign werden nun auch im Datenteil immer Blöcke von $16 \cdot T_C$ verarbeitet. Dabei wird nur die Position des *Burst*s berücksichtigt. Das endgültige Basisband-Signal wird dann im sog. *Frontcore* erzeugt.

A.4.1.6 Frontcore

Der *Frontcore* ist eine von IHP Microelectronics entwickelte Komponente. Ich konnte sie im Basisband-Prozessor auf dem *FPGA* einsetzen, um die benötigte Taktrate zu verringern. In späteren Versionen des HF-*Front-End* soll der *Frontcore* integriert werden, wodurch der *FPGA* entlastet werden kann. Diese Komponente setzt die Signale des Basisband-Prozessors in das digitale Basisbandsignal um. Dieses Signal hat eine Taktrate von 499,2 MHz, die mit einem *FPGA* nur schwer erreichbar ist. Sie besteht aus einem Übersetzer und einem Serialisierer.

Der Übersetzer hat drei Eingänge mit einem Takt von 31,20 MHz und einer Datenbreite von jeweils einem Bit. Die drei Eingänge des Übersetzers geben an ob es sich um Präambel- oder Datenteil handelt, die Phasenlage des Symbols und ob Pulse gesendet werden sollen oder nicht. Abhängig von diesen Eingängen liegen am Ausgang zwei 16 Bit lange Vektoren an. Diese Vektoren werden vom Serialisierer Bit-weise in einem Takt von 499,2 MHz ($= \frac{1}{T_C}$) an den HF-Teil ausgegeben. Dabei gibt das erste Bit an, ob ein Puls gesendet werden soll und das zweite die Phasenlage.

Der Übersetzer muss daher für den standardkonformen Betrieb fünf verschiedene, vorkonfigurierte *Burst*-Formen ausgeben können.[6]

Auf meinen Vorschlag hin wurde zusätzliche Konfigurierbarkeit implementiert, um auch das Aussenden mehrpulsiger Präambeln sowie eine geringere Anzahl als 16 Pulse pro *Burst* zu erlauben. Über einen zusätzlichen Eingang kann dazu ein Konfigurationsmodus gestartet werden, um die Ausgabevektoren des Übersetzers für die jeweiligen Eingangsworte festzulegen. Der Übersetzer besitzt dabei auch einen *Scrambler* und nimmt das *Burst Scrambling* autonom

[6] Positives und negatives Präambel-Sub-Symbol, *Burst* in Phase sowie gegenphasig und kein Signal

vor. Dafür muss im Konfigurationsmodus der *Seed* des *Scramblers* angegeben werden.

A.4.2 Empfänger

Der Empfänger ist komplexer als der Sender. Er besteht aus einer Komponente zur Präambel-Detektion und -Synchronisation und einer weiteren zur Demodulation und Dekodierung der Daten.

A.4.2.1 Präambel-Detektion und -Synchronisation

Die Ausgabewerte des *ADC* werden durch das *IPPM*-Verfahren zu Präambel-Sub-Symbolen aufaddiert. Diese addierten Werte werden zur Kreuzkorrelation in einen 31 Symbole langen Korrelator geleitet. Dieser Korrelator ist ein Schieberegister, das bei jedem Taktzyklus den Vektor der Eingabewerte mit einem Vektor von Sollwerten multipliziert. Das Ergebnis dieser Multiplikation ergibt einen skalaren Wert für die Übereinstimmung des Eingangssignals mit dem gesuchten Signal, also den Korrelationswert. Da das *Front-End* keine Vorzeichen-behafteten Werte liefert, werden angepasste Sollwerte verwendet.

Die Korrelationswerte werden mit einem Schwellwert verglichen. Dieser Schwellwert wird einmal initialisiert und dann nach jeder Übertragung persistent gespeichert. Ist der Korrelationswert höher als der Schwellwert und gleichzeitig niedriger als sein Vorgänger, dann wird er als gefundenes Präambel-Symbol gewertet. Der Schwellwert wird entsprechend angepasst und ein Signal leitet den Fund weiter.

Wird kein *SFD* erwartet, dann wird direkt in den Daten-Empfangsmodus umgeschaltet.

Für den Empfang eines *SFD* wird das Signal in ein Schieberegister geschrieben, das mit der Länge eines Präambel-Symbols getaktet ist. Ist der Inhalt des Schieberegisters mit dem *SFD*-Code identisch, dann wurde der Paketanfang gefunden und der Empfänger in den Daten-Empfangsmodus umgeschaltet.

A.4.2.2 Demodulation

Die Daten-Demodulation benutzt die direkten Eingangssignale (ohne *IPPM*). Während des Präambel-Empfanges werden Samples gespeichert. Sobald der Zeitpunkt des Paketbeginns feststeht, werden ab diesem Sample jeweils die Samples, die einer Hopping-Position entsprechen, aufaddiert. Dabei wird die vorgestellte „überlappende Addition“ angewendet. Die resultierenden Werte werden in einem Ringpuffer von der Länge eines Symbols gespeichert. Aus diesem Speicher werden die beiden Werte ausgelesen, deren Adresse mit der Hopping-Position für dieses Symbol übereinstimmt. Aus der Differenz der beiden Werte ergibt sich ein Soft-Bit, das dann zur Dekodierung weitergegeben wird.

A.4.2.3 Auslesen des PHY-Headers

Um den *PHY*-Header zu empfangen, werden die ersten 18 Soft-Bits durch einen Komparator in Bit-Werte gewandelt. Diese Bit-Werte werden in ein Schieberegister geschoben, dessen Bit-Werte parallel ausgelesen und mittels *SECDED*-Code verifiziert und gegebenenfalls repariert werden. Wird ein Fehler im *PHY*-Header festgestellt, so wird der Empfänger zurückgesetzt. Wenn kein Fehler festgestellt wurde, kann die Länge des Pakets ausgelesen werden, danach geht der Empfänger zurück in den Präambel-Empfangsmodus.

A.4.2.4 Dekodierung

Die restlichen Soft-Bits werden abhängig von der verwendeten Kodierung verarbeitet. So können sie entweder einem Soft-Bit-fähigen Viterbi-Decoder[7] zugeführt oder per Komparator in Bit-Werte gewandelt werden. In beiden Fällen bleiben „harte“ Bit-Werte zurück, die in 6-Bit-Symbole deserialisiert werden. Im Falle von Reed-Solomon-kodierten Daten werden diese Symbole für die Benutzung des Reed-Solomon-Decoders zu Blöcken von 55+8 Symbolen geordnet und, falls nötig, der letzte Block mit Füll-Bits aufgefüllt. Die daraus

[7] Es wurde der Viterbi-Decoder aus der Bibliothek von Synplicity DSP verwendet

resultierenden 6-Bit-Symbole, entweder aus dem Reed-Solomon-Decoder oder dem Komparator, werden dann der *MAC*-Schicht bereitgestellt.

A.5 HF-Front-End

Als HF-*Front-End* wurde ein von IHP Microelectronics entwickelter Prototyp eines IEEE 802.15.4a-kompatiblen Energie-Detektions-Empfängers verwendet.

A.5.1 Sender

Über den Frontcore liegt am Eingang des Senders das Basisband-Signal als 2 Bit breites Signal mit einer Taktrate von 499,2 MHz an. Dieses Signal entscheidet darüber, ob und mit welcher Phase ein Puls gesendet wird. Dieses Signal wird durch eine Passband-Modulation auf eine Trägerwelle mit 7,987 GHz aufmoduliert. Die Abstrahlleistung war zum Zeitpunkt des Verfassens der Arbeit nicht regelbar.

A.5.2 Empfänger

Die Empfangsseite des HF-Chips mischt über einen I/Q-Mischer das Signal herunter und gibt ein „gleichgerichtetes“, verstärktes Basisband-Signal aus, welches nur positive Pulse enthält. Dieses Signal wird mit einem speziell dafür angefertigten Integrator mit einem *Integrationsintervall* von 16 ns gesammelt, und mit einem 6-Bit-*ADC* digitalisiert.

A.6 Ausblick

Die Implementierung in generischem *VHDL* und die Realisierung auf einem handelsüblichen *FPGA* ermöglichen einen *Proof-of-Concept* in Verbindung mit einer Weiterentwicklung des *Front-End*. Sie bieten weiterhin eine gute Ausgangsbasis für die Umsetzung eines ASIC für ein endgültiges System.

Anhang B

Verzeichnisse

Glossar

Aktor
: Netzwerkknoten, der Steuernachrichten vom Controller empfängt, und an eine aktive Komponente weiterleitet.

AS-Interface
: drahtgebundenes Protokoll zur Automatisierung auf der Sensor/Aktor-Ebene.

Beacon
: Zyklisch verschicktes Paket zur Synchronisation, das einen Superframe eröffnet.

Bluetooth
: Digitales Funksystem nach Standard IEEE 802.15.1.

Burst
: Eine Reihe von Pulsen.

Burst Scrambling
: Maßnahme zur Glättung des Spektrums. Dabei wird jedem Puls eines Bursts eine zufällige Phasenlage zugeordnet.

Controller

Zentraler Knoten und Koordinator des Netzwerks.

Front-End

analoger Hochfrequenzteil eines Senders oder Empfängers.

Group ACK

Bit-Feld zur Empfangsbestätigung für mehrere Pakete.

Integrationsintervall

Zeitraum, über den integriert wird. Bestimmt die Taktrate des ADC.

L-Spreading

Länge eines Prämbel-Sub-Symbols bzw. Pause zwischen Pulsen in der Präambel.

MATLAB

Mathworks MATrix LABoratory, Programmiersprache und -umgebung für numerische Analysen.

niedrigratige Energie-Detektion

Empfangsart für Pulsfunk, bei der die Energiedichte im Kanal über einen Zeitraum gemessen wird, der länger ist als ein einzelner Puls.

Polling

Zyklische Abfrage von peripheren Knoten durch einen Zentralen.

Präambel

Zyklische Sequenz, die einer Paketübertragung vorangestellt ist, unter anderem um zu synchronisieren und die *gain control* einzustellen.

Reed-Solomon-Code

Blockcodeschema, das insbesondere vor zusammenhängenden Fehlern schützt. Die Kodierung erfolgt in einem Gaulois-Feld, so werden beispielsweise 300 Bits als 55 6-Bit-Symbole kodiert und mit acht 6-Bit-Symbolen geschützt.

Soft-Bit

Soft-Bits sind eine Anwendung der *Fuzzy Logic* in der Signalverarbeitung, jedes Bit wird durch einen reellen Wert dargestellt, der sowohl den Bit-Wert als auch die Verlässlichkeit dieses Wertes charakterisiert.

Superframe

Zyklus, in dem der Controller Beacons aussendet. Bildet die Zykluszeit der Applikation ab.

Abkürzungsverzeichnis

ACK

Acknowledgement - Bestätigungsnachricht über ein empfangenes Paket.

ADC

Analog to Digital Converter - Digitalisierer.

AGC

Automatic Gain Control - automatische Regelung des Empfangsverstärkers.

APM

Alternierende Puls-Modulation - Basisband-Modulationsverfahren, wird im AS-Interface-Protokoll eingesetzt..

ARQ

Automatic Repeat Request - Automatische Übertragungswiederholung bei Paketverlust.

ASN

Absolute Slot Number - Netzwerkweit bekannte Nummer eines Zeitschlitzes im TSMP-Protokoll.

AWGN

Additive White Gaussian Noise - addiertes Gauß-Rauschen.

BAN

Body Area Network - (drahtloses) Netzwerk im und am Körper.

BPM

Burst Position Modulation.

BPSK

Binary Phase Shift Keying.

BTS

Base Time Slot - Basiszeitschlitz.

CDMA

Code Division Multiple Access - Zugriffsverfahren auf Basis individueller Codes.

CI

change indicator - Veränderungsanzeiger: Präfix vor einem Frame, der anzeigt, ob es seit der letzten Übertragung zu einer Änderung der Daten kam..

CRC

Cyclic Redundancy Check - Prüfsumme zur Verfikation von Daten.

CSMA

Carrier Sensing Multiple Access - Verteiltes Zugriffsverfahren, bei dem nur bei freiem Träger gesendet wird..

DARPA

Defence Advanced Research Projects Agency - Einrichtung der US-amerikanischen Rüstungsforschung.

DS-UWB

Direct-Sequence UWB - Zeitlich gespreiztes Übertragungsverfahren.

DSSS

Direct Sequence Spread Spectrum - Modulationsform, bei der jedes Bit durch einen bestimmten Code bzw. dessen Invertierung übertragen wird..

ETSI

European Telecommunications Standards Institute - Europäisches Regulierungsorgan.

FCC

Federal Communications Commission - US-amerikanische Regulierungsbehörde.

FDMA

Frequency Division Multiple Access - Zugriffsverfahren auf Basis verschiedener Frequenzkanäle.

FEC

Forward Error Correction - Fehlerkorrekturcode.

FPGA

Field Programmable Gate Array - Programmierbare Logik.

GI

Guard Interval - Sicherheitsabstand.

GUI

Graphical User Interface - Graphische Benutzeroberfläche.

HSPA

High Speed Packet Access - Schneller Datenübertragungsmodus in UMTS-Mobilfunknetzen.

IPPM

Iterative Poly Phase Matching - iterative Mehrphasenkorrektur.

IR-UWB

Impulse Radio UWB - Ultra-Breitband-Kommunikation auf Basis von Pulsfunk.

ISM

Industial, Scientific, Medical - International freigegebene Funkbänder für industrielle, wissenschaftliche oder medizinische Anwendungen.

LDC

Low Duty Cycle - geringer Aktivierungszyklus.

LFSR

Linear Feedback Shift Register - lineares Schieberegister mit Rückkoppelung.

LOS

Line Of Sight - Sichtverbindung.

MAC

Media Access Control - Medienzugriffssteuerung.

MB-OFDM

Multi-Band Orthogonal Frequency Multiplexing - UWB-Frequenzspreizverfahren.

NACK

negatives *Acknowledgement*-Paket, weist den Sender auf einen Paketverlust hin.

NLOS

Non Line Of Sight - ohne Sichtverbindung.

OFDM

Orthogonal Frequency Multiplexing - Frequenzspreizverfahren.

OOK

On-Off Keying.

PAM

Pulse Amplitude Modulation.

PHY

Physikalische Schicht.

PPM

Pulse Position Modulation.

PRP

Pulse Repetition Period - Periode zwischen zwei Pulsen.

PRR

Pulse Repetition Rate - Pulswiederholungsrate.

PSD

Power Spectral Density - Energiedichte in Frequenzdomäne.

RAM

Random Access Memory - Speicher mit wahlfreiem Zugriff.

RFID

Radio Frequency ID - aktive Tags zur drahtlosen Identifikation.

RMS

Root Mean Square - Effektivwert.

SECDED

Single Error Correction, Double Error Detection - spezielle Familie von Hamming-Codes zur Fehlererkennung und -korrektur. Sie ermöglichen die Erkennung von zwei Fehlern oder Korrektur von einem einzelnen Fehler..

SFD

Start of Frame Delimiter - Synchronisationspunkt zwischen Präambel und Datenteil.

SNR

Signal to Noise Ratio - Signal-zu-Rauschen-Verhältnis.

TAWASI

TAWASI Ain't Wireless AS-I - Drahtloser Ersatz für das AS-Interface-Protokoll.

TDMA

Time Division Multiple Access - Medienzugriffssteuerung auf Basis von Zeitschlitzen.

TFC

Time Frequency Code - Zeit-Frequenz-Sprungfolge.

TH

Time Hopping - Zeitsprungverfahren.

THS

Time-Hopping-Sequenz.

TOF

Time of flight - Flugzeit.

TPC

Transmit Power Control - geregelte Sendeleistung.

TSMP

Time Synchronized Mesh Protocol - Synchronisations- und Mesh-Netzwerkprotokoll.

UART

Universal Asynchronous Receiver/Transmitter - universeller asynchroner Sender/Empfänger.

UWB

Ultra-Wideband - Ultra-Breitband.

VHDL

Very Highspeed integrated circuits Description Language - Hardware-Beschreibungssprache.

WISA

Wireless Interface for Sensors and Actors.

WPAN

Wireless Personal Area Networks - Drahtlose Netze mit Reichweiten im Bereich mehrerer Meter.

Liste der Symbole

$N_{BTSBytes}$

Anzahl der Bytes im Paket, mit dem der BTS berechnet wird.

N_{CPB}

Burstlänge in Chips bzw. Pulsen.

N_{Hops}

Anzahl der möglichen Hopping-Positionen.

N_{Knoten}

Anzahl der Sensor/Aktor-Knoten im Netz (ausgenommen Controller).

$N_{Nutzdaten}$

Anzahl der Bits der Nutzdaten.

N_{Pburst}

Anzahl der Pulse pro Präambel-Sub-Symbol.

N_{Prep}

Anzahl der Wiederholungen des Präambel-Symbols im Synchronisationsteil.

T_C

Zeitintervall von der Länge eines Pulses.

T_{Prea}

Dauer der Präambel.

T_{Psym}

Länge eines Präambel-Symbols.

T_{Sym}

Symboldauer.

$T_{\mathbf{Zyklus}}$

Zykluszeit, innerhalb der ein Controller einen Nachrichtenaustausch mit jedem Knoten vornimmt.

f_C

Mittenfrequenz.

Eigene Publikationen

[HBSK11] Johannes Hund, Michael Bahr, Christian Schwingenschlögl, and Rolf Kraemer. Drahtlose Adaption von AS-Interface durch Optimierung von IEEE 802.15.4a. In *Procedeedings of 2. Jahreskolloquium Kommunikation in der Automation (KommA 2011)*, Magdeburg, Germany, 2011.

[HHA+10] J. Hund, A. Heinrich, A.Ziller, C. Schwingenschlögl, and R. Kraemer. A packet-level adaptive forward error correction scheme for wireless networks. In *Positioning, Navigation and Communication, 2010. WPNC 2010. 7th Workshop on*, Dresden,Germany, March 2010.

[HKS11] Johannes Hund, Rolf Kraemer, and Christian Schwingenschlögl. Extension of IEEE 802.15.4a to improve Resilience for Wireless Automation using Soft-bit Combination. In *2011 IEEE International Conference on Ultra-Wideband (ICUWB 2011)*, Bologna, Italy, September 2011.

[HKSH09] J. Hund, R. Kraemer, C. Schwingenschlögl, and A. Heinrich. Overview of MAC layer enhancements for IEEE 802.15.4a. In *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on*, pages 43–49, Hannover,Germany, March 2009.

[HOSK10] J. Hund, S. Olonbayar, C. Schwingenschlögl, and R. Kraemer. Evaluation and Optimization of Robustness in the IEEE 802.15.4a

Standard. In *Ultra-Wideband, 2010. ICUWB 2010. IEEE International Conference on*, Nanjing, China, March 2010.

Patente

[P1] M.Khan, J.Hund, A. Ziller, C.Schwingenschlögl, „Packet-type based Resilience using Network Coding“, Internationales Patent WO2009/030560, Erstanmeldung 03.09.2007, veröffentlicht 12.03.2009

[P2] J.Hund, A.Heinrich, C.Schwingenschlögl, „Verfahren zur pulsbasierten Ultra-Breitband-Kommunikation zwischen zumindest einem Sende-Knoten und zumindest einem Empfangs-Knoten“, Internationales Patent WO2010/043480, Erstanmeldung 15.10.2008, veröffentlicht 22.04.2010

[P3] J.Hund, S.Huckenholz, A.Heinrich, C.Schwingenschlögl, „Verfahren und Vorrichtung zum drahtlosen Übertragen von ...“, Internationale Patentanmeldung PCT/EP2011/053020, Erstanmeldung 01.03.2010

[P4] J.Hund, S.Huckenholz, A.Heinrich, D.Krush, „A Preamble Detector“, Europäische Patentanmeldung 10006698.4, Erstanmeldung 28.06.2010

[P5] J.Hund, A.Heinrich, C.Schwingenschlögl, „Verfahren zum Verarbeiten von über einen Kommunikationskanal übertragene Pakete“, Deutsche Patentanmeldung 102011002823.4, Erstanmeldung 18.01.2011

Literaturverzeichnis

[1] *IEEE Draft Standard for local and metropolitan area networks, Part 15.4: low rate WPANs, Amendment 5: Amendment to the MAC sublayler.*

[2] *IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs).* IEEE Std 802.15.4-2006, 2006. `http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf`.

[3] *IEEE Standard 802.15.4a-2007: Standard for local and metropolitan area networks, Part 15.4: low rate WPANs, Amendment 1: Add Alternate PHYs*, 2007. `http://standards.ieee.org/getieee802/download/802.15.4a-2007.pdf`.

[4] Ahlswede, R., N. Cai, S.Y.R. Li und RW Yeung: *Network information flow.* IEEE Transactions on Information Theory, 46(4):1204–1216, 2000.

[5] Akogun, A.E., R.C. Qiu und N. Guo: *Demonstrating time reversal in ultra-wideband communications using time domain measurements.* In: *51st International Instrumentation Symposium*, Seiten 8–12, 2005.

[6] Arslan, Huseyin, Zhi Ning Chen und Maria-Gabriella Di Benedetto: *Ultra Wideband Wireless Communication.* Wiley-Interscience, Oktober 2006, ISBN 0471715212.

[7] AS-Interface Association: *AS-Interface online academy.* online, Abruf am 25.08.2011. http://as-interface.net/academy/content/sys/start/start.en.html.

[8] AS-Interface Association: *Die Funktionsweise des APM-Verfahrens.* online, Abruf am 25.08.2011, July 2011. http://as-interface.net/academy/content/tech/kap3/kap3_13.de.html.

[9] Barrett, T.W.: *History of ultrawideband (uwb) radar & communications: Pioneers and innovators.* In: *Proceedings of Progress in Electromagnetics Symposium 2000 (PIERS2000)*, Cambridge, MA, July 2000.

[10] Benedetto, Maria Gabriella Di, Thomas Kaiser, Andreas F. Molisch, Ian Oppermann, Christian Politano und Domenico Porcino: *UWB Communication Systems - a comprehensive Overview.* EURASIP Book Series on Signal Processing and Communications. Hindawi Publishing Corporation, 2006.

[11] Benthin, M. und KD Kammeyer: *Extraction of soft bit information with the use of m-ary modulation.* 1995.

[12] Blechnet: *Mit einer automatisierten Fertigungszelle alles im Griff.* online, Abruf am 10.9.2011. http://www.blechnet.com/themen/umformen/articles/176152/index3.html.

[13] Braem, B., B. Latré, I. Moerman, C. Blondia und P. Demeester: *The wireless autonomous spanning tree protocol for multihop wireless body area networks.* In: *Mobile and Ubiquitous Systems-Workshops, 2006. 3rd Annual International Conference on*, Seiten 1–8. IEEE.

[14] Breed, Gary: *A summary of fcc rules for ultra wideband communications.* online, Abruf am 11.02.2010. http://www.highfrequencyelectronics.com/Archives/Jan05/HFE0105_Tutorial.pdf.

[15] Buda, Aurel, Volker Schuermann und Joerg F. Wollert: *Wireless technologies in factory automation.* InTech, Factory Automation, 2010. http://www.intechopen.com/articles/show/title/wireless-technologies-in-factory-automation.

[16] Buda, Aurel und Jörg F. Wollert: *Leistungsbewertung von Ultra Wideband (UWB) Funktechnologien nach ECMA 368 für den Einsatz in industriellen Kommunikationssystemen* . In: *Procedeedings of 2. Jahreskolloquium Kommunikation in der Automation (KommA 2011)*, Magdeburg, Germany, 2011.

[17] Bundesnetzagentur: *Allgemeinzuteilung von Frequenzen für die Nutzung durch Anwendungen geringer Leistung der Ultra- Wideband (UWB) Technologie.* online, Abruf am 18.04.2009. `http://www.bundesnetzagentur.de/media/archive/13155.pdf`.

[18] Bundesnetzagentur: *Frequenznutzungsplan.* online, Abruf am 11.07.2011. `http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/BNetzA/Sachgebiete/Telekommunikation/Regulierung/Frequenzordnung/Frequenznutzungsplan/Frequenznutzungsplan2008_Id17448pdf.pdf?__blob=publicationFile`.

[19] Chan, T. M., K. F. Man, K. S. Tang und S. Kwong: *A jumping-genes paradigm for optimizing factory wlan network.* Industrial Informatics, IEEE Transactions on, 3(1):33 –43, feb. 2007, ISSN 1551-3203.

[20] De Pellegrini, F., D. Miorandi, S. Vitturi und A. Zanella: *On the use of wireless networks at low level of factory automation systems.* Industrial Informatics, IEEE Transactions on, 2(2):129–143, 2006.

[21] ETSI: *EN 302 065 electromagnetic compatibility and radio spectrum matters (ERM); ultra wideband (UWB) technologies for communication purposes; harmonized EN covering the essential requirements of article 3.2 of the r&tte directive.* online, Abruf am 03.05.2008. `http://pda.etsi.org/pda/home.asp?wki_id=slrdeL5XsC56B5ABMYqW`.

[22] Federal Communications Commission: *First Report and Order: Revision of Part 15 of the Commission's Rules Regarding Ultra-Wideband Transmission Systems.* online, Abruf am 06.05.2008. `http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-02-48A1.pdf`.

[23] Flury, Manuel: *Interference Robustness and Security of Impulse-Radio*

Ultra-Wide Band Networks. Docteur ès sciences, ecole polytechnique federale de lausanne, 2010.

[24] Foerster, Jeff: *Channel modeling sub-committee report final.* online, Abruf am 16.10.2008. http://www.ieee802.org/15/pub/2003/Mar03/02490r1P802-15_SG3a-Channel-Modeling-Subcommittee-Report-Final.zip.

[25] Foerster, Jeff und Qinghua Li: *UWB Channel Modeling Contribution from Intel.* online, Abruf am .

[26] Fontana, Dr. Robert J.: *A brief history of uwb communications.* online, Abruf am 10.02.2009. http://www.ferret.com.au/n/A-brief-history-of-UWB-communications-n681131.

[27] Fontana, R.J.: *Recent applications of ultra wideband radar and communications systems.* Ultra-Wideband, Short-Pulse Electromagnetics 5, Seiten 225–234, 2002.

[28] Geib, Bernhard: *Praktikum Verlässliche Systeme - Versuch 10.* online, Abruf am 06.07.2011. www.cs.hs-rm.de/~rnlab/LVaktuell/RelSys/Praktikum/VersuchVS10.pdf.

[29] Gigl, T., F. Troesch, J. Preishuber-Pfluegl und K. Witrisal: *Maximal operating distance estimation for ranging in ieee 802.15. 4a ultra wideband.* In: *Positioning Navigation and Communication (WPNC), 2010 7th Workshop on*, Seiten 10–18. IEEE, 2010.

[30] Gyselinckx, B., C. Van Hoof, J. Ryckaert, R.F. Yazicioglu, P. Fiorini und V. Leonov: *Human++: autonomous wireless sensors for body area networks.* In: *Custom Integrated Circuits Conference, 2005. Proceedings of the IEEE 2005*, Seiten 13–19. IEEE, 2005.

[31] Hall, P.S. und Y. Hao: *Antennas and propagation for body centric communications.* In: *Antennas and Propagation, 2006. EuCAP 2006. First European Conference on*, Seiten 1–7. IEEE, 2006.

[32] Han, J. und C. Nguyen: *Development of a tunable multiband uwb radar sensor and its applications to subsurface sensing.* Sensors Journal, IEEE, 7(1):51–58, 2007.

[33] Hancke, G.P. und B. Allen: *Ultrawideband as an industrial wireless solution.* Pervasive Computing, IEEE, 5(4):78 –85, oct.-dec. 2006, ISSN 1536-1268.

[34] HART Communication Foundation: *HART] specifications.* online, Abruf am 28.06.2011. `http://www.hartcomm.org/protocol/about/aboutprotocol_specs.html`.

[35] HART Communication Foundation: *Wireless HART - technical data sheet.* online, Abruf am 28.06.2011. `www.hartcomm.org/protocol/training/resources/wiHART_resources/wirelesshart_datasheet.pdf`.

[36] Hirsch, A.: *Werkzeugmaschinen: Grundlagen : Lehr- und "Ubungsbuch : mit 413 Abbildungen und 27 Tabellen.* Viewegs Fachb "ucher der Technik. Vieweg, 2000, ISBN 9783528049508. `http://books.google.de/books?id=EDGIvAoysckC`.

[37] Hochwald, B.M. und S. Ten Brink: *Achieving near-capacity on a multiple-antenna channel.* Communications, IEEE Transactions on, 51(3):389–399, 2003.

[38] Huckenholz, Stephan: *Prototype simulation of transmitter and receiver for uwb communication ieee 802.15.4a.* Diplomarbeit, TU Muenchen, Jun 2010.

[39] Iacobucci, M.S. und M.G. Di Benedetto: *Computer method for pseudo-random codes generation.* National Italian Patent RM2001A000592, dec 2001.

[40] International, ECMA: *ECMA 368 - high rate ultra wideband phy and mac standard.* online, Abruf am 05.08.2008. `http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-368.pdf`.

[41] Katti, S., D. Katabi, W. Hu, H. Rahul und M. Medard: *The importance of being opportunistic: Practical network coding for wireless environments.* In: *Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[42] Katti, S., H. Rahul, W. Hu, D. Katabi, M. Médard und J. Crowcroft: *XORs in the air: practical wireless network coding.* IEEE/ACM Transactions on Networking (TON), 16(3):497–510, 2008.

[43] Kjellsson, J., A.E. Vallestad, R. Steigmann und D. Dzung: *Integration of a wireless i/o interface for profibus and profinet for factory automation.* Industrial Electronics, IEEE Transactions on, 56(10):4279–4287, 2009.

[44] Koetter, R. und M. Medard: *An algebraic approach to network coding.* IEEE/ACM transactions on networking, 11(5):782–795, 2003.

[45] Koetter, R. und A. Vardy: *Algebraic soft-decision decoding of Reed-Solomon codes.* IEEE Transactions on Information Theory, 49(11):2809–2825, 2003, ISSN 0018-9448.

[46] Koopman, P. und T. Chakravarty: *Cyclic redundancy code (crc) polynomial selection for embedded networks.* In: *Dependable Systems and Networks, 2004 International Conference on*, Seiten 145–154. IEEE, 2004.

[47] KUKA Roboter: *Typen"ubersicht.* online, Abruf am 10.9.2011. `http://www.kuka-robotics.com/res/sps/a737ee03-5832-4c95-9d91-84e0de80c664_Typen%C3%BCbersicht_en.pdf`.

[48] Lei, Z., F. Chin und Y.S. Kwok: *Uwb ranging with energy detectors using ternary preamble sequences.* In: *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, Band 2, Seiten 872–877. IEEE, 2006.

[49] Liu, X.F., B.Z. Wang, S.Q. Xiao und J.H. Deng: *Performance of impulse radio uwb communications based on time reversal technique.* Progress In Electromagnetics Research, 79:401–413, 2008.

[50] Lunze, Jan: *Automatisierungstechnik.* Oldenbourg Wissensch.Vlg, 2007, ISBN 9783486580617.

[51] Molisch und Foerster: *Channel Models for Ultrawideband Personal Area Networks.* online, Abruf am 15.10.2008. `http://www.merl.com/papers/docs/TR2004-071.pdf`.

[52] Molisch, A. F., K. Balakrishnan, C. C. Chong, S. Emami, A. Fort, J. Karedal, J. Kunisch, H. Schantz, U. Schuster und K. Siwiak: *IEEE 802.15. 4a channel model-final report.* online, Abruf am 03.11.2008. `https://mentor.ieee.org/802.15/file/04/15-04-0662-00-004a-channel-model-final-report-r1.pdf`.

[53] Molisch, Andreas F., Kannan Balakrishnan, Dajana Cassioli, Chia Chin Chong, Shahriar Emami, Andrew Fort, Johan Karedal, Juergen Kunisch, Hans Schantz, Ulrich Schuster und Kai Siwiak: *IEEE 802.15.4a channel model - final report.* online, Abruf am 17.0.2008, April 2005. `http://grouper.ieee.org/groups/802/15/pub/04/15-04-0662-02-004a-channel-model-final-report-r1.pdf`.

[54] Nardis, L. De und M. G. Di Benedetto: *Overview of the IEEE 802.15.4/4a standards for low data rate wireless personal data networks.* In: *Positioning, Navigation and Communication, 2007. WPNC '07. 4th Workshop on*, Seiten 285–289, Hannover,, März 2007.

[55] Neteon: *SIMATIC NET White Paper: Industrial Wireless LAN – Industrial Features and Current Standards.* Online, Abruf am 5.9.2011. `http://www.neteon.net/PDFFiles/Industrial_wireless_land_features_and_standards.pdf`.

[56] Olonbayar, S., D. Kreiser und R. Kraemer: *Performance and design of IR-UWB transceiver baseband for wireless sensors.* In: *Ultra-Wideband, 2009. ICUWB 2009. IEEE International Conference on*, Seiten 809–813, Vancouver, BC, September 2009.

[57] Pellegrini, F. De, D. Miorandi, S. Vitturi und A. Zanella: *On the use of wireless networks at low level of factory automation systems.* IEEE Transactions on Industrial Informatics, 2(2):129–143, Mai 2006, ISSN 1551-3203.

[58] Pister, K.S.J. und L. Doherty: *Tsmp: Time synchronized mesh protocol.* In: *Proceedings of the IASTED International Symposium*, Band 635, Seite 391.

[59] Pozar, D.M.: *Closed-form approximations for link loss in a uwb radio sys-*

tem using small antennas. Antennas and Propagation, IEEE Transactions on, 51(9):2346–2352, 2003.

[60] Prof. Dr.-Ing. B. Dorsch: *Codierung zur Fehlerkorrektur.* online, Abruf am 11.09.2011. `www.nt.tu-darmstadt.de/nt/fileadmin/kt/Lehre/WS0506/ecc/skript.pdf`.

[61] Profibus Nutzerorganisation (PNO) e.V.: *IO-Link im Durchblick: Spezifikation in Kürze.* online, Abruf am 13.07.2011. `http://www.io-link.com/share/Downloads/IO-Link_im_Durchblick.pdf`.

[62] Promwong, Sathaporn und Jun-Ichi Takada: *Free space link budget estimation scheme for ultra wideband impulse radio with imperfect antennas.* IEICE Electronics Express, 1(7):188–192, 2004.

[63] Qiu, R.C., C. Zhou, N. Guo und J.Q. Zhang: *Time reversal with miso for ultrawideband communications: Experimental results.* Antennas and Wireless Propagation Letters, IEEE, 5(1):269–273, 2006.

[64] Roberto Aiello, Anuj Batra: *Ultra Wideband Systems - Technologies and Applications.* Elsevier, 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA, 2006.

[65] Ross, Dr. Gerald F.: *Early Motivations and History of Ultra Wideband Technology.* online, Abruf am 10.02.2009. `www.multispectral.com/history.html`.

[66] Saleh, A.A.M. und R. Valenzuela: *A statistical model for indoor multipath propagation.* Selected Areas in Communications, IEEE Journal on, 5(2):128 –137, february 1987, ISSN 0733-8716.

[67] Savitha, HM und M. Kulkarni: *Performance comparison of hard and soft-decision turbo coded ofdm systems.* In: *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, Seiten 551–555. IEEE.

[68] Sörgel, Werner: *Charakterisierung von Antennen für die Ultra-Wideband-Technik.* Dissertation, Fakultät für Elektrotechnik und Informationstechnik der Universität Fridericiana Karlsruhe (TH), Mai 2007. `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000007210`.

[69] Staderini, E.M.: *Uwb radars in medicine.* Aerospace and Electronic Systems Magazine, IEEE, 17(1):13–18, 2002.

[70] Steigmann, Richard und Jan Endresen: *White Paper - Introduction to WISA.* online, Abruf am 27.06.2011. http://www05.abb.com/global/scot/scot209.nsf/veritydisplay/c81a48ddc34c5125c12571f100410e74/$file/White_Paper%20Introduction%20to%20WISA%20V2.pdf.

[71] Strohmer, T., M. Emami, J. Hansen, G. Papanicolaou und A.J. Paulraj: *Application of time-reversal with mmse equalizer to uwb communications.* In: *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, Band 5, Seiten 3123–3127. IEEE, 2004.

[72] Su-quan, D., Y. Zhi-xing, PAN Chang-yong und W. Jun: *Symbol-level soft-decision decoding of Reed-Solomon codes.*

[73] Sushchik, M. Jr., N. Rulkov, L. Larson, L. Tsimrin, H. Abarbanel, K. Yao und A. Volkovskii: *Chaotic pulse position modulation: a robust method of communicating with chaos.* IEEE Commun. Lett., 4(4):128–130, April 2000, ISSN 1089-7798.

[74] Takizawa, K., T. Aoyagi, J. Takada, N. Katayama, K. Yekeh, Y. Takehiko und K.R. Kohno: *Channel models for wireless body area networks.* In: *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, Seiten 1549–1552. IEEE, 2008.

[75] TSG-RAN Working Group 1: *Performance comparison of bit level and symbol level Chase combining.* online, Abruf am 11.9.2011. http://ftp.3gpp.org/tsg_ran/WG1_RL1/TSGR1_20/Docs/PDFs/R1-01-0472.pdf.

[76] Vikström, Peter: *WISA Wireless Interface for Sensors and Actuators in Industrial Applications.* online, Abruf am 18.06.2009. http://users.tkk.fi/virranko/sensor_networks/vikstrom2.pdf.

[77] Weniger, K.: *Pacman: Passive autoconfiguration for mobile ad hoc networks.* Selected Areas in Communications, IEEE Journal on, 23(3):507–519, 2005.

[78] Willig, A.: *Polling-based mac protocols for improving real-time performance in a wireless profibus.* Industrial Electronics, IEEE Transactions on, 50(4):806–817, 2003.

[79] Willig, A.: *Recent and emerging topics in wireless industrial communications: A selection.* Industrial Informatics, IEEE Transactions on, 4(2):102 –124, may 2008, ISSN 1551-3203.

[80] Willig, A., M. Kubisch, C. Hoene und A. Wolisz: *Measurements of a wireless link in an industrial environment using an ieee 802.11-compliant physical layer.* Industrial Electronics, IEEE Transactions on, 49(6):1265 – 1282, dec 2002, ISSN 0278-0046.

[81] WiMedia Alliance: *Resources.* Online, Abruf am 03.02.2009. `http://www.wimedia.org/en/resources`.

[82] Yang, Y. und A.E. Fathy: *See-through-wall imaging using ultra wideband short-pulse radar system.* In: *Antennas and Propagation Society International Symposium, 2005 IEEE*, Band 3, Seiten 334–337. IEEE, 2005.

[83] Yarovoy, AG, LP Ligthart, J. Matuzas und B. Levitas: *Uwb radar for human being detection.* Aerospace and Electronic Systems Magazine, IEEE, 21(3):10–14, 2006.

[84] Zetik, R., J. Sachs und R. Thoma: *Uwb localization-active and passive approach [ultra wideband radar].* In: *Instrumentation and Measurement Technology Conference, 2004. IMTC 04. Proceedings of the 21st IEEE*, Band 2, Seiten 1005–1009. IEEE, 2004.