

**Masterarbeit**

**Simulation und Implementierung  
energieeffizienter Funkkommunikation  
für intelligente Logistiklager**

**cand. M.Sc. Jens Drenhaus**

---

Betreuer : Prof. Dr.-Ing. Christian Wietfeld

Robert Falkenberg, M.Sc.

Eingereicht am : 30. Juli 2017

Jens Drenhaus:  
*Simulation und Implementierung energieeffizienter  
Funkkommunikation für intelligente Logistiklager*

BETREUER:  
Prof. Dr.-Ing. Christian Wietfeld  
Robert Falkenberg, M.Sc.

Dortmund

Juli 2017

---

## ABSTRACT

---

ToDo

---

## ZUSAMMENFASSUNG

---

ToDo

Diese Arbeit beschäftigt sich mit der Entwicklung und Bewertung eines geeigneten Kanalzugriffverfahrens. Der Lösungsansatz ist die Analyse des [CSMA-CA](#) Verfahrens aus dem IEEE 802.15.4 Protokoll. Mit Hilfe eines Simulationsmodells werden Variationen der Standardparameter untersucht. Dabei zeigt die Modifikation des dort verwendeten Backoff-Algorithmus positive Auswirkungen sowohl auf die Zuverlässigkeit der Übertragungen als auch auf den Energieverbrauch. Durch eine prototypische Implementierung dieses Ansatzes werden die Ergebnisse der Simulation im Feldversuch validiert.



---

## INHALTSVERZEICHNIS

---

<b>1</b>	<b>EINLEITUNG</b>	<b>1</b>
1.1	Motivation . . . . .	2
1.2	Lösungsansatz . . . . .	2
1.3	Struktur der Arbeit . . . . .	2
<b>2</b>	<b>GRUNDLAGEN</b>	<b>5</b>
2.1	Funkkommunikation . . . . .	5
2.1.1	Freiraumausbreitung . . . . .	6
2.1.2	Rauschen . . . . .	8
2.1.3	Modulation . . . . .	9
2.1.4	Kanalausnutzung . . . . .	11
2.2	Kommunikationsprotokolle . . . . .	13
2.2.1	Der IEEE 802.15.4 Standard . . . . .	15
2.3	Energieverbrauch . . . . .	17
2.3.1	Energiemodell . . . . .	18
2.3.2	Energiemessung . . . . .	18
2.4	Simulationen . . . . .	20
2.4.1	Ereignisorientierte Simulation . . . . .	20
2.4.2	OMNeT++ . . . . .	21
2.5	Hardware . . . . .	22
2.5.1	MSP430 . . . . .	23
2.5.2	CC1200 Funkchip . . . . .	26
2.6	Entwicklungssoftware . . . . .	28
<b>3</b>	<b>VERWANDTE ARBEITEN</b>	<b>31</b>
3.1	Übersicht . . . . .	31
3.2	Drei Projekte im Detail . . . . .	34
<b>4</b>	<b>AUSWAHL DES KANALZUGRIFFSVERFAHRENS</b>	<b>39</b>
4.1	ALOHA . . . . .	39
4.2	Listen Before Talk . . . . .	40
4.3	Carrier Sense Multiple Access / Collision Avoidance . . . . .	40
4.4	Schlussfolgerung für die weitere Untersuchung . . . . .	42
<b>5</b>	<b>SYSTEMBESCHREIBUNG UND ANALYSEMETHODE</b>	<b>45</b>
5.1	Ein geeignetes Szenario . . . . .	45
5.1.1	Anordnung . . . . .	46
5.1.2	Energieversorgung . . . . .	47
5.1.3	Datenaufkommen . . . . .	47
5.1.4	Kommunikationsfluss . . . . .	48
5.1.5	Kanalmodell . . . . .	50
5.2	Analyse- und Messmethode . . . . .	50
5.2.1	Messaufbau . . . . .	50
<b>6</b>	<b>SIMULATION</b>	<b>53</b>
6.1	IEEE 802.15.4 Modell . . . . .	53

6.2	Erweiterungen des Modells . . . . .	53
6.3	Simulationsergebnisse . . . . .	53
7	AUSWERTUNG	57
8	PROTOTYPISCHE IMPLEMENTIERUNG	59
8.1	Funktioinstest . . . . .	59
8.2	Messungen . . . . .	59
9	ZUSAMMENFASSUNG	65
A	CSMA/CA BLOCKDIAGRAMM	67
B	CC1200 REGISTERKONFIGURATION	69
C	CSMA/CA IMPLEMENTIERUNG	75
D	ZUFALLSZAHLENERZEUGUNG	79
	LITERATUR	81

---

## ABKÜRZUNGEN

---

6LoWPAN	IPv6 für <a href="#">WPAN</a> mit geringer Leitungsaufnahme, engl. <i>IPv6 over Low Power WPAN</i> ↗ <a href="#">Glossar</a> .
ADC	Analog-Digital Wandler, engl. <i>Analog to Digital Converter</i> .
AP	Zugangspunkt, engl. <i>Access Point</i> .
AWGN	Additives weißes gaußsches Rauschen, engl. <i>Additive White Gaussian Noise</i> .
CAP	Konkurrenzzugriff Zeitraum, engl. <i>Contention Access Period</i> .
CCA	Kanalbewertung, engl. <i>Clear Cannel Assessment</i> .
CDMA	Codevielfachzugriff, engl. <i>Code Division Multiple Access</i> .
CPS	Cyber-physisches System, engl. <i>Cyber-Physical System</i> .
CPU	Zentrale Recheneinheit, engl. <i>Central Processing Unit</i> .
CS	Engl. <i>Chip Select</i> .
CSMA	Vielfachzugriff mit Trägerprüfung, engl. <i>Carrier Sense Multiple Access</i> .
CSMA-CA	Vielfachzugriff mit Trägerprüfung und Kollisionserkennung, engl. <i>Carrier-Sense Multiple Access / Collision Avoidance</i> .
CTS	Auf Deutsch „klar zum Senden“, engl. <i>clear to send</i> .
DES	Ereignisorientierte Simulation, engl. <i>Discrete Event Simulation</i> .
EPC	Elektronischer Produkt Code, engl. <i>Electronic Product Code</i> .
ETSI	Engl. <i>European Telecommunications Standards Institute</i> .
FDMA	Frequenzvielfachzugriff, engl. <i>Frequency Division Multiple Access</i> .
FEL	Liste der zukünftigen Ereignisse bei <a href="#">DESs</a> , engl. <i>Future Event List</i> .
FRAM	Ferroelektrischer Speicher mit wahlfreiem Zugriff, engl. <i>Ferroelectric to Random Access Memory</i> .
FSK	Frequenzumtastung, engl. <i>Frequency Shift Keying</i> .
GPIO	Allgemeiner Ein- und Ausgabepin, engl. <i>General Purpose Input Output</i> .
GTIN	Globale Handelsgut Identifikationsnummer, engl. <i>Global Trade Item Number</i> .

IEEE	Engl. <i>Institute of Electrical and Electronics Engineers</i> .
ISM	Engl. <i>Industrial Scientific Medical</i> <a href="#">Glossar</a> .
ISO	Engl. <i>International Organization for Standardization</i> .
ISR	Engl. <i>Interrupt Service Routine</i> .
LBT	Auf Deutsch etwa „erst horchen, dann sprechen“, engl. <i>Listen Before Talk</i> .
LED	Leuchtdiode, engl. <i>Light Emitting Diode</i> .
LLC	Engl. <i>Logical Link Control</i> .
MAC	Medienzugriffssteuerung, engl. <i>Media Access Control</i> .
MCPS-SAP	Engl. <i>MAC Common Part Sublayer Service Access Point</i> .
MISO	Engl. <i>Master In Slave Out</i> <a href="#">Glossar</a> .
MLME-SAP	Engl. <i>MAC Sublayer Management Entity Service Access Point</i> .
MOSI	Engl. <i>Master Out Slave In</i> <a href="#">Glossar</a> .
OMNeT++	Engl. <i>Objective Modular Network Testbed in C++</i> <a href="#">Glossar</a> .
OSI	Offene Systemvernetzung, engl. <i>Open Systems Interconnection</i> .
PAN	Netzwerk für das unmittelbare Umfeld, engl. <i>Personal Area Network</i> .
PD-SAP	Engl. <i>PHY Data Service Access Point</i> .
PLME-SAP	Engl. <i>Physical Layer Management Entity Service Access Point</i> .
PSDR	Produktspezifische Zustellrate, engl. <i>Product Specific Delivery Ratio</i> .
RFID	Identifizierung durch elektromagnetische Wellen, engl. <i>Radio-Frequency Identification</i> .
RISC	Rechner mit reduziertem Instruktionssatz, engl. <i>Reduced to Instruction Set Computer</i> .
RTS	Auf Deutsch „Anfrage zum Senden“, engl. <i>request to send</i> .
SAP	Dienstzugangspunkt, engl. <i>Service Access Point</i> .
SCL	Engl. <i>Serial Clock</i> <a href="#">Glossar</a> .
SDMA	Raumfachzugriff, engl. <i>Space Division Multiple Access</i> .
SNR	Signal-Rausch-Verhältnis, engl. <i>Signal to Noise Ratio</i> .
SPI	Engl. <i>Serial Peripheral Interface</i> <a href="#">Glossar</a> .
SRD	Kurzstreckenfunkgerät, engl. <i>Short Range Device</i> .
TDMA	Zeitvielfachzugriff, engl. <i>Time Division Multiple Access</i> .
WLAN	Drahtloses lokales Netzwerk, engl. <i>Wireless Local Area Network</i> .

WPAN	Drahtloses Netzwerk für das unmittelbare Umfeld, engl. <i>Wireless Personal Area Network.</i>
WSN	Drahtloses Sensornetzwerk, engl. <i>Wireless Sensor Network.</i>

---

## GLOSSAR

---

6LoWPAN	Implementierung des Internet Protokolls für <a href="#">WPANs</a> mit geringer Leitungsaufnahme ( <a href="#">6LoWPAN</a> ) und damit Voraussetzung für den Zugang zum Internet.
ALOHA	Kanalzugriffsverfahren, das 1970 von Norman Abramson an der Universität von Hawaii entwickelt wurde [1].
Bluetooth	Funktechnologie zum Datenaustausch zwischen Geräten über kurze Distanzen im 2,4GHz Band.
C	Imperative Programmiersprache, die vorwiegend zur maschinennahen Programmierung verwendet wird.
C++	Erweiterung der Programmiersprache <a href="#">C</a> . Erlaubt objektorientiertes Programmieren.
CC1200	Funkchip für den Sub-GHz Bereich von Texas Instruments. Siehe auch <a href="#">Unterabschnitt 2.5.2</a> .
Client	Computer oder Gerät, das Dienste von einem <a href="#">Server</a> in einem Netzwerk anfordert.
CS Leitung	Signalleitung zur Auswahl des <i>Slaves</i> beim <a href="#">SPI</a> , engl. <a href="#">Chip Select (CS)</a> . Siehe <a href="#">Unterabschnitt 2.5.1</a> .
Energy Harvesting	Auf Deutsch „Energie ernten“. Bezeichnet die Gewinnung elektrischer Energie aus der Umgebung durch Licht, Bewegung oder Temperaturunterschiede.
Hidden Station	Auf Deutsch „verborgene Station“. Beschreibt die Unfähigkeit, die Funkaktivität weiter entfernter Sender zu detektieren. Siehe auch <a href="#">Unterunterabschnitt 2.1.4.1</a> .
IEEE 802.11g	Standardisierter Protokollstapel für den Informationsaustausch in <a href="#">WLANs</a> .
IEEE 802.15.4	Standardisierter Protokollstapel für drahtlose Netzwerke mit geringer Datenrate.
Industrie 4.0	Beschreibt den zunehmenden Einfluss von Kommunikations- und Informationstechnik in klassischen Industrieabläufen. Grundlage dafür bilden intelligente, vernetzte Systeme.
ISM-Band	Frequenzbereiche für die Nutzung in Industrie, Wissenschaft und Medizin, engl. <a href="#">Industrial Scientific Medical (ISM)</a> .
MISO Leitung	Datenleitung beim <a href="#">SPI</a> für die Datenübertragung vom <i>Slave</i> zum <i>Master</i> , engl. <a href="#">Master In Slave Out (MISO)</a> . Siehe <a href="#">Unterabschnitt 2.5.1</a> .

MOSI Leitung	Datenleitung beim <b>SPI</b> für die Datenübertragung vom <i>Master</i> zum <i>Slave</i> , engl. <b><i>Master Out Slave In (MOSI)</i></b> . Siehe <a href="#">Unterabschnitt 2.5.1</a> .
Node	Auf Deutsch „Knoten“. Umverteilungspunkt oder Endpunkt bei Datenübertragungen.
OMNeT++	<b>C++</b> Framework für ereignisorientierte Simulationen, speziell für Netzwerkprotokolle ( <b><i>Objective Modular Network Testbed in C++</i></b> ).
Routing	Ermitteln eines geeigneten Weges einer Datenübertragung in einem Netzwerk.
SCL Leitung	Taktleitung zur synchronen Datenübertragung beim <b>SPI</b> , engl. <b><i>Serial Clock (SCL)</i></b> . Siehe <a href="#">Unterabschnitt 2.5.1</a> .
Server	Computer innerhalb eines Netzwerks, der anderen Computern oder Geräten, den <b>Clients</b> , Dienste zur Verfügung stellt.
SPI	Synchrones serielles Bussystem zum Datenaustausch zwischen einem <i>Master</i> -Gerät und einem oder mehreren <i>Slave</i> -Geräten. Siehe <a href="#">Unterabschnitt 2.5.1</a> .
Transceiver	Funkgerät mit integrierter Sende- und Empfangseinheit, engl. <b><i>Transmitter Receiver</i></b> .
Watchdog	Auf Deutsch „Wachhund“. Komponente eines Mikrocontrollers zum kontrollierten Neustart. Siehe <a href="#">Unterabschnitt 2.5.1</a> .
ZigBee	Standardisierter Protokollstapel für Netzwerke mit geringem Datenaufkommen. ZigBee basiert auf dem <b>IEEE 802.15.4</b> Standard und erweitert diesen um eine Netzwerk- und Anwendungsschicht.



# 1

---

## EINLEITUNG

---

Bei der Entwicklung in Richtung **Industrie 4.0** werden Industrieanlagen und andere physische Systeme zur effizienten Auslastung und Energienutzung über einen zunehmenden Kommunikationsfluss miteinander vernetzt. Verkabelte Systeme sind in den meisten Anwendungsfällen zu unflexibel oder gar nicht realisierbar, sodass auf kabelllose Kommunikation ausgewichen werden muss. Moderne eingebettete Systeme stellen hierbei die nötige Performanz zur Verfügung. So entstehen **Cyber-physische Systeme**, engl. **Cyber-Physical Systems (CPSs)**, die eine Dezentralisierung komplexer Systeme ermöglichen und zu steigender Flexibilität führen. In großen Logistiklagern bieten vernetzte Frachtcontainer ein hohes Maß an Selbstorganisation, wie die Überwachung des Frachtgutes oder die automatische Inventarisierung [2]. Es entsteht ein **drahtloses Sensornetzwerk**, engl. **Wireless Sensor Network (WSN)**, in welchem die Frachtcontainer als Netzwerknoten, den so genannten **Nodes**, fungieren.

In Logistiklagern ist nur der Betrieb von besonders energieeffizienten oder gar energieautarken Lösungen denkbar, da Batteriewechsel oder Akku-Ladevorgänge die Dynamik und Zuverlässigkeit des gesamten Lagers negativ beeinträchtigen. Aus diesem Grund müssen derartige Systeme besonders sparsam mit der verfügbaren Energie haushalten. Eine Schlüsselrolle stellt in diesem Zusammenhang die Funkkommunikation dar, die im Verhältnis zur übrigen Intelligenz der Frachtcontainer besonders energieintensiv ist [2][3][4].

Folglich müssen Übertragungen zuverlässig übermittelt werden. Diesem Ziel wirken in einem Logistiklager jedoch zahlreiche Faktoren entgegen: Z.B. erhöht die sehr hohe Teilnehmerzahl die Interferenz auf den einzelnen Übertragungsstrecken, was zu massiven Übertragungskollisionen führt [3][5].

Aktuelle Forschungen beschäftigen sich sowohl mit den speziellen Anforderungen intelligenter, vernetzter Logistiklager [3], als auch mit Skalierungseffekten von Sensornetzen allgemein [5]. Die bei großskali gen Anwendungen inhärente Konkurrenz der Teilnehmer wird hierbei wiederholt als wesentliches Problem benannt.

### 1.1 MOTIVATION

Auf Grund der genannten hohen Anzahl an konkurrierenden **Nodes** innerhalb eines Netzwerkes ist der Kanalzugriff ein kritischer Punkt im Bezug auf eine verlässliche und effiziente Funkkommunikation. Eine Netzwerküberlastung kann aus einem hohen Datenaufkommen einzelner **Nodes** oder aus einer großen Zahl konkurrierender **Nodes** innerhalb eines **WSN** resultieren. Kommen beide Fälle zusammen, wird das Netzwerk noch stärker beansprucht. Schlagen Übertragungen fehl, müssen sie ggf. wiederholt werden, was die Netzlast weiter erhöht und sich negativ auf den Energieverbrauch der **Nodes** auswirkt.

Diese Arbeit beschäftigt sich daher mit der Entwicklung und Bewertung eines geeigneten Kanalzugriffverfahrens für derartige Logistikzenarien. Dabei sollen die folgenden Fragestellungen beantwortet werden:

1. Wo stoßen gängige Kanalzugriffverfahren an ihre Grenzen?
2. Wie können sie hinsichtlich ihrer Leistungsstärke modifiziert werden?
3. Welche Einschränkungen ergeben sich daraus?
4. Welchen Einfluss nimmt das Kanalzugriffverfahren auf den Energieverbrauch?

### 1.2 LÖSUNGSANSATZ

Der Lösungsansatz ist die Analyse des *Carrier Sense Multiple Access / Collision Avoidance* (**CSMA-CA**) Verfahrens aus dem **IEEE 802.15.4** Protokoll. Mit Hilfe eines Simulationsmodells wird der Algorithmus zum Kanalzugriff zunächst mit Standardparametern unter den Rahmenbedingungen eines Logistiklagers untersucht. Im Anschluss werden Auswirkungen der Variation **CSMA-CA**-Parameter simuliert. Zur Beurteilung der Energieeffizienz fließen Messdaten realer **Transceiver** in die Simulation ein. Durch eine prototypische Implementierung dieses Ansatzes werden die Simulationsergebnisse im Feldversuch hinsichtlich der Zuverlässigkeit und des Energieverbrauchs validiert.

### 1.3 STRUKTUR DER ARBEIT

TODO: aktualisieren

Die Arbeit gliedert sich wie folgt. Kapitel 3 zeigt die Ergebnisse bisheriger Untersuchungen auf und führt hin zur Motivation, welche in ?? ausgeführt wird. Grundlegend für die eigentliche Untersuchung beschreibt ?? die Struktur des analysierten Szenarios. In ?? werden zwei standardisierte Kanalzugriffsverfahren unter Berücksichtigung

des Energieverbrauchs diskutiert. Es folgt eine umfassende Beschreibung des verwendeten Simulationsmodells

TODO: weitere Beschreibung der Kapitel vervollständigen.



# 2

---

## GRUNDLAGEN

---

In diesem Kapitel werden die erforderlichen Grundlagen beschrieben, auf denen die Auslegung der Simulation, der Bewertung der Ergebnisse und die prototypische Implementierung aufbauen. Dabei gliedert sich das Kapitel in 5 Teile: [Abschnitt 2.1](#) und [Abschnitt 2.2](#) behandeln Grundzüge der Funkkommunikation sowie den damit verbundenen Verarbeitungsvorschriften für die Systemsoftware, den Protokollen. Es folgt [Abschnitt 2.3](#) mit relevanten Faktoren und Diagnosemöglichkeiten hinsichtlich des Energieverbrauchs bei Funkkommunikation. [Abschnitt 2.4](#) gibt eine Einführung in ereignisorientierte Simulationen. In [Abschnitt 2.5](#) werden die verwendeten Hardwareplattformen und ihre Besonderheiten vorgestellt.

### 2.1 FUNKKOMMUNIKATION

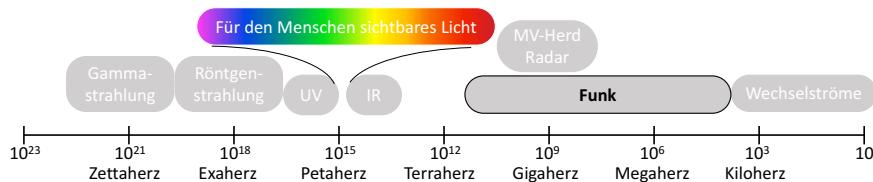


Abbildung 2.1: Einordnung von Funkwellen im elektromagnetischen Spektrum

Funkkommunikation ist die Übertragung von Informationen durch elektromagnetische Wellen. Dabei stehen Funkwellen verschiedener Frequenzen als Übertragungsmedium zu Verfügung. Abschnitte des Funkspektrums werden Bänder genannt. Ein Frequenzband kann je nach Anwendung in weitere Abschnitte, den sogenannten Kanälen, unterteilt werden. Der Radio- und Fernsehrundfunk, der Mobilfunk oder Satellitenübertragungen sind typische Beispiele. Aber auch [WLAN](#), [Bluetooth](#), schnurlose Telefone, Rauchmelder oder [WSNs](#) nutzen Funkwellen zum Informationsaustausch. Eine Einordnung der Funkwellen im gesamten elektromagnetischen Spektrum ist in [Abbildung 2.1](#) zu sehen. Den verschiedenen Anwendungen sind dabei bestimmte Frequenzabschnitte zugeordnet. Diese Zuordnung ist teilweise mit kostenpflichtigen Lizenzen versehen, die in Deutschland die Bundes-

netzagentur verwaltet. So werden z. B. einzelne Frequenzbereiche an Mobilfunkanbieter versteigert. Neben den lizenzierten Bändern, stehen auch Bereiche speziell für den Einsatz in Industrie, Wissenschaft und Medizin zur Verfügung. Diese Frequenzen werden im **ISM** Band zusammengefasst. Die in Deutschland zur Verfügung stehenden Bereiche befinden sich u.a. im 434MHz Band, im 2,4GHz Band und im 5,7GHz Band. Der 868MHz ist vornehmlich für Alarmfunkanlagen vorgesehen. Der Bereich 868,6MHz – 868,7MHz steht jedoch auch für allgemeinen Datenaustausch zur Verfügung [6].

### 2.1.1 Freiraumausbreitung

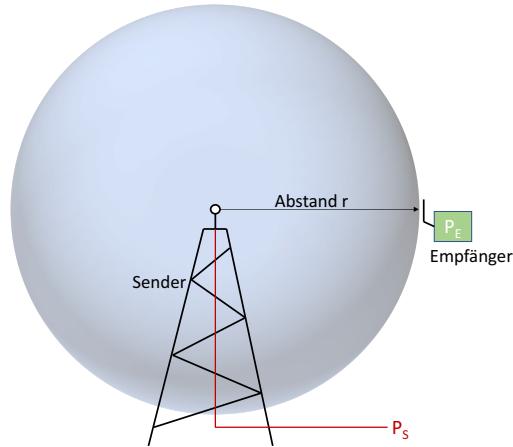


Abbildung 2.2: Idealisierte Funkwellenausbreitung mit isotropen Antennen

Zur Beschreibung der Wellenausbreitung seien zunächst isotrope Strahler betrachtet, also Antennen, von denen sich die Funkwellen mit der Lichtgeschwindigkeit  $c$  gleichmäßig in alle Raumrichtungen ausbreiten, wie in [Abbildung 2.3](#) skizziert. Wird ein zu übertragenes Signal mit der Leistung  $P_S$  in eine solche Sendeantenne eingespeist, so verteilt sich diese Leistung im Abstand  $r$  auf die Fläche einer Kugel mit Radius  $r$ . Die Leistungsdichte an einem beliebigen Punkt im Umfeld des Senders beträgt:

$$S = \frac{P_S}{A_{\text{Kugel}}} = \frac{P_S}{4\pi r^2} \quad (2.1)$$

Um der elektromagnetischen Welle Leistung zu entnehmen, muss die Empfangsantenne im mathematischen Sinn eine Fläche besitzen, die von der Leistungsdichte durchsetzt wird. Diese wird Antennenwirkfläche  $A_W$  genannt.

$$P_E = S \cdot A_W \quad (2.2)$$

ABSTAND	P <sub>E</sub>
10m	0,57mW
20m	0,14mW
30m	0,06mW

Tabelle 2.1: Empfangsleistung bei Freiraumausbreitung und isotropen Rundstrahler mit der Sendeleistung 1Watt bei 100MHz

Die Antennenwirkfläche ist nicht die geometrische Fläche der Antenne, sondern ein frequenzabhängiger Wert. Bei einer isotropen Antenne ist die Antennenwirkfläche definiert zu:

$$A_W = \frac{c^2}{4\pi f^2} \quad (2.3)$$

Nach Einsetzen von [Gleichung 2.3](#) und [Gleichung 2.1](#) in [Gleichung 2.2](#) ergibt sich die Empfangsleistung:

$$P_E = \frac{P_S}{4\pi r^2} \cdot \frac{c^2}{4\pi f^2} = P_S \cdot \left( \frac{c}{4\pi r f} \right)^2 \quad (2.4)$$

In [Tabelle 2.1](#) sind exemplarisch die Empfangsleistungen einer Übertragung bei 100Mhz und eine Sendeleistung von 1Watt eingetragen. Dabei wird ersichtlich, dass am Empfänger nur noch ein Bruchteil der ursprünglichen Signalleistung zur Verfügung steht. Die Verringerung der Sendeleistung wird Dämpfung genannt und als logarithmisches Maß in Dezibel angegeben. Sie ist das Verhältnis von P<sub>S</sub> zu P<sub>E</sub>. Nach [Gleichung 2.4](#) ergibt sich die *Freiraumdämpfung* F zu:

$$F = 10 \lg \left( \frac{P_S}{P_E} \right) = 10 \lg \left( \frac{4\pi r f}{c} \right)^2 \quad (2.5)$$

In [Abbildung 2.3](#) ist die Freiraumdämpfung für drei verschiedene Frequenzen des **ISM**-Bands über die Entfernung zwischen Sender und Empfänger aufgetragen. Es ist gut zu erkennen, dass hohe Frequenzen stärker gedämpft werden als niedrige.

Bei realen Antennen entspricht die abgestrahlte Leistung auf Grund von z. B. ohmschen Verlusten innerhalb der Antenne, nicht der eingespeisten Leistung. Dieser Verlustfaktor, also das Verhältnis von abgestrahlter zu eingespeister Leistung wird durch den Antennenwirkungsgrad  $\eta$  ausgedrückt.

In der Regel strahlen Antennen nicht gleichmäßig in alle Raumrichtungen, wie in [Abbildung 2.3](#). Antennen können speziell so konzipiert werden, dass sie die abgestrahlte Leistung in bestimmten Richtungen bündeln. Es kann z. B. in einigen Anwendungen sinnvoll sein, wenn die Sendeantenne weniger stark nach oben und nach unten strahlt, dafür umso mehr in die Breite. Das ist bei Dipolantennen der

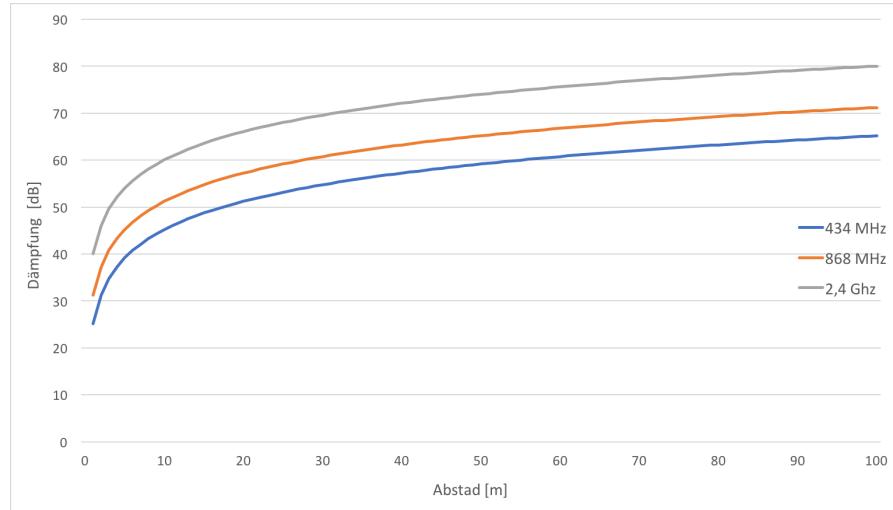


Abbildung 2.3: Freiraumdämpfung bei verschiedene Frequenzen

Fall. Ein weiteres Beispiel ist der Richtfunk. Hierbei wird die Strahlungsleistung in eine Raumrichtung maximiert. Parabolantennen weisen dieses Merkmal auf. Zur Berechnung wird diese Eigenschaft im Richtfaktor  $D$  einer Antenne ausgedrückt. Es ist das Verhältnis der Leistungsdichte  $S_{\max}$  in der Hauptstrahlrichtung der Antenne zu der Strahlungsleistung  $S$  einer gleichförmig strahlenden Antenne an gleicher Stelle.

Die Größen  $\eta$  und  $D$  charakterisieren eine Antenne und werden im Antennengewinn  $g$  zusammengefasst.

$$g = \eta \cdot D \quad (2.6)$$

Unter Berücksichtigung des Antennengewinns von Sende- und Empfangsantenne kann [Gleichung 2.4](#) zur Antennengleichung für Freiraumsbreitung erweitert werden:

$$P_E = P_S \cdot g_S \cdot g_E \cdot \left( \frac{c}{4\pi r_f} \right)^2 \quad (2.7)$$

### 2.1.2 Rauschen

Ein Signal wird während der Übertragung durch äußere Störungen beeinträchtigt, die z. B. durch andere Funksender oder die kosmische Hintergrundstrahlung hervorgerufen werden. Die Überlagerung aller störenden äußeren Einflüsse wird als Rauschen bezeichnet. Rauschen kann daher als Überlagerung vieler elektromagnetischer Wellen unterschiedlicher Frequenzen und mit unterschiedlichen Signalamplituden aufgefasst werden. Um diese Effekte auf dem Funkkanal bei Übertragung zu berücksichtigen, ist ein Kanalmodell notwendig. Unter der Annahme, dass die einzelnen Schwingungen völlig unabhängig von einander sind, liegt ein [additives weißes gaußsches Rauschen](#),

engl. **Additive White Gaussian Noise (AWGN)** vor und man spricht von einem **AWGN**-Kanal.

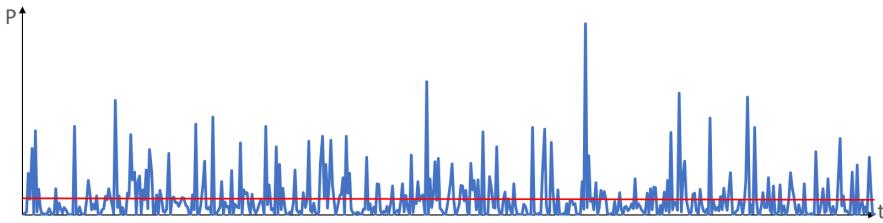


Abbildung 2.4: Zeitlicher Verlauf der momentanen Rauschleistung eines **AWGN**-Kanals

Bei diesem Kanalmodell sind die Signalamplituden des Rauschsignals normalverteilt, d.h. der Großteil der vorkommenden Amplituden weicht nur wenig von dem durchschnittlichen Wert ab, wohingegen extreme Amplituden nur selten vorkommen. Der Verlauf der Rauschleistung eines solchen Signals ist in [Abbildung 2.4](#) dargestellt. Weiterhin ist die Rauschleistungsdichte  $N_0$  bei einem **AWGN**-Kanal über alle Frequenzen konstant. Die Rauschleistung  $N$  eines Kanals mit der Bandbreite  $B$  ergibt sich dann zu:

$$N = N_0 \cdot B \quad (2.8)$$

Wie in [Unterabschnitt 2.1.1](#) beschrieben, fällt die Signalleistung bis zum Ort des Empfängers stark ab. Damit ein Nutzsignal mit der Signalleistung  $P$  vom Empfänger detektiert werden kann, muss  $P$  größer als die vorherrschende Rauschleistung  $N$  sein. Der Abstand von  $N$  zu  $P$  wird als *Störabstand* oder als **Signal-Rausch-Verhältnis, engl. Signal to Noise Ratio (SNR)** bezeichnet und logarithmisch in Dezibel angegeben.

$$\text{SNR} = 10 \lg \left( \frac{P}{N} \right) = 10 \lg \left( \frac{P}{N_0 \cdot B} \right) \quad (2.9)$$

### 2.1.3 Modulation

Bei einer digitalen Datenübertragung werden die zu übermittelnden Informationseinheiten als Symbole bezeichnet. Die Geschwindigkeit, mit der Symbole übertragen werden, ist die Symbolrate und hat die Einheit *Baud*. In einem Symbol können mehrere Bits zusammengefasst sein. Die Anzahl der Bits pro Symbol bestimmt den Modulationsindex  $M$ . In [Abbildung 2.5](#) ist eine Bitfolge mit Modulationsindex  $M = 2$  und  $M = 4$  gegenübergestellt. Es ist zu erkennen, dass es bei einem Bit pro Symbol zwei unterschiedliche Symbole gibt, bei zwei Bits pro Symbol dagegen vier unterschiedliche Symbole. Es gilt der Zusammenhang:

$$M = \text{Anzahl}_{\text{Bits/Symbol}}^2 \quad (2.10)$$

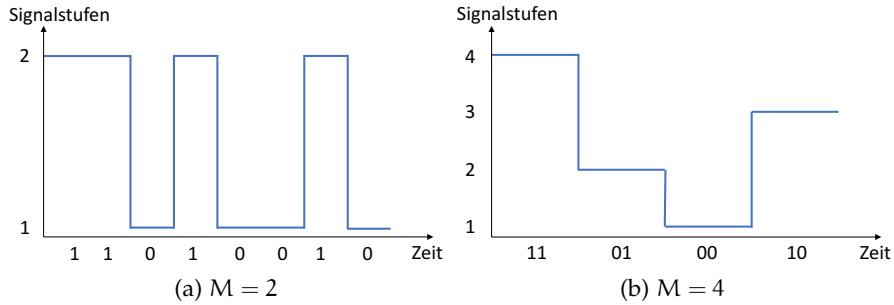


Abbildung 2.5: Gegenüberstellung einer Bitfolge bei verschiedenen Signalstufen. a) Ein Symbol entspricht einem Bit b) In einem Symbol werden zwei Bits zusammengefasst, es gibt vier Signalstufen

Die Anzahl der übertragenen Bits pro Sekunde steigt daher bei größerem Modulationsindex. Für die Datenrate R gilt:

$$R = \log_2(M) \quad (2.11)$$

Die digitalen Daten müssen für die Übertragung auf ein analoges Signal, den sogenannten Träger, modelliert werden. Dies kann z. B. durch die Manipulation der Amplitude, der Frequenz oder der Phasenlage einer Sinusschwingung erzielt werden. Dieser Vorgang wird auch als Umtastung bezeichnet. Besonders robust gegen Überlagerungen anderer Signale ist die **Frequenzumtastung, engl. Frequency Shift Keying (FSK)**. Hierbei wird die Frequenz der Trägerschwingung in Abhängigkeit des zu übertragenden Symbols leicht verändert, wie in [Abbildung 2.6](#) angedeutet. In dem abgebildeten Beispiel, wird nur ein Bit pro Symbol übertragen, d.h der Modulationsindex M ist gleich zwei. Man spricht daher in dem Fall von einer **2-FSK**.

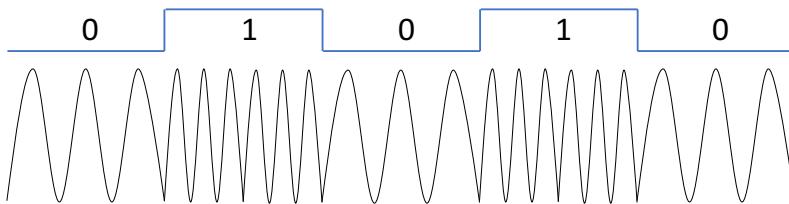


Abbildung 2.6: Ein digitales Signal wird mittels **FSK** mit  $M = 2$  auf ein Trägersignal modelliert. Dabei wird die Frequenz des Trägers variiert.

Abrupte Übergänge im zeitlichen Signalverlauf führen zu einem breiten Frequenzspektrum. Daher kann bei einer Datenübertragung nicht beliebig schnell zwischen zwei Symbolen umgeschaltet werden, die verfügbare Bandbreite B des Funkkanals begrenzt die Datenrate. Die theoretisch maximal erreichbare Datenrate ist nach Nyquist:

$$R_{\max} = 2 \cdot B \cdot \log_2(M) \quad (2.12)$$

Nach [Gleichung 2.11](#) lässt sich die Datenrate durch Erhöhen des Modulationsindexes steigern. Störungen auf dem Funkkanal erhöhen jedoch dabei die Wahrscheinlichkeit, dass die verschiedenen Symbole am Empfänger nicht mehr korrekt voneinander unterscheiden werden können. Das Shannon-Hartley-Gesetz gibt die maximale Datenrate für eine Übertragung auf einem verrauschten Kanal mit der Rauschleistung  $N$  an, bei der die Wahrscheinlichkeit einer fehlerfreien Übertragung noch größer Null ist.

$$R_{\max} = B \cdot \log_2 \left( 1 + \frac{P_{\text{Signal}}}{N} \right) \quad (2.13)$$

Ein Datenstrom wird üblicherweise nicht direkt in die Sendefrequenz umgetastet, sondern zunächst auf eine Zwischenfrequenz. Diese wird im Anschluss auf die tatsächliche Sendefrequenz, z. B. 868MHz, gemischt.

[Unterabschnitt 2.1.1](#) hat gezeigt, dass Übertragungen bei hohen Frequenzen wegen der stärkeren Dämpfung Nachteile in der Ausbreitung gegenüber niedrigeren Frequenzen aufweisen. In diesem Abschnitt wird deutlich, dass hohe Frequenzen dafür bezüglich hoher Datenraten geeigneter sind. In den höheren Frequenzbereichen steht den einzelnen Kanälen mehr Bandbreite zur Verfügung, was nach [Gleichung 2.13](#) eine größere Datenrate erlaubt.

#### 2.1.4 Kanalausnutzung

Um einen Funkkanal für mehrere Benutzer zugänglich zu machen und dabei dennoch eine funktionierende Kommunikation zu gewährleisten, ist ein Zugriffsverfahren notwendig. Nachfolgend sind dazu verschiedene Techniken aufgeführt.

**FDMA** Beim [Frequenzvielfachzugriff, engl. Frequency Division Multiple Access \(FDMA\)](#) wird jedem Nutzer ein eigenes Teilband innerhalb des vorgesehenen Frequenzbereiches zugeordnet, sodass im Endeffekt die Übertragungen der einzelnen Benutzer in parallelen Kanälen stattfinden. Es ist darauf zu achten, dass sich die einzelnen Kanäle nicht gegenseitig stören. Dies kann z. B. durch das Hinzufügen von Schutzkanälen realisiert werden, in denen keine Übertragung stattfindet. Dieses Verfahren schränkt die Datenrate der Teilnehmer wegen der geringeren Bandbreite ein (vgl. [Unterabschnitt 2.1.3](#)).

**TDMA** Beim [Zeitvielfachzugriff, engl. Time Division Multiple Access \(TDMA\)](#) wird nicht das Funkspektrum als begrenzte Ressource aufgeteilt, sondern die Zeit. Jedem Teilnehmer wird ein eigener Zeitschlitz zugeteilt, in dem dieser senden darf. Dieses Verfahren erfordert eine genaue Synchronisierung aller Teilnehmer im Netzwerk.

**CDMA** Bei dem Verfahren mit **Codevielfachzugriff**, engl. *Code Division Multiple Access* (**CDMA**) wird ein schmalbandiges Signal auf einen sehr viel breiteren Frequenzbereich gespreizt. Diese Spreizung erfolgt für jeden Nutzer nach einem individuellen Muster. Die breitbandigen Signale sind unempfindlich gegen schmal- und breitbandige Störungen. Somit können sie trotz Überlagerung am Empfänger wieder von einander getrennt werden. Dazu verwendet der Empfänger jeweils das gleiche Spreizmuster, wie der Sender.

**SDMA** Beim **Raumvielfachzugriff**, engl. *Space Division Multiple Access* (**SDMA**) werden den Nutzern verschiedene räumliche Sektoren zugeteilt. Dies kann mittels der Richtcharakteristik von geeigneten Antennen erreicht werden (vgl. [Unterabschnitt 2.1.1](#)).

**CSMA** **Vielfachzugriff mit Trägerprüfung**, engl. *Carrier Sense Multiple Access* (**CSMA**) ist ein dezentrales, asymmetrisches Verfahren. Das bedeutet, dass keine zentrale Koordination oder Synchronisation der Nutzer nötig ist. Die Nutzer beanspruchen nach einem bestimmten Algorithmus das alleinige Zugriffsrecht auf den Kanal. Zur Koordination beobachten die Nutzer dabei den Kanal. Eine detaillierte Beschreibung solcher Verfahren erfolgt in [Kapitel 4](#).

#### 2.1.4.1 *Hidden Station Problem*

Ein Effekt, der bei der zuletzt genannten Technik zu beachten ist, ist das **Hidden Station**. Dieser tritt beim Abhören des Funkkanals auf, wenn es Nutzer im Netzwerk gibt, die außerhalb der Reichweite anderer Nutzer liegen. [Abbildung 2.7](#) zeigt drei Nutzer mit den jeweili-

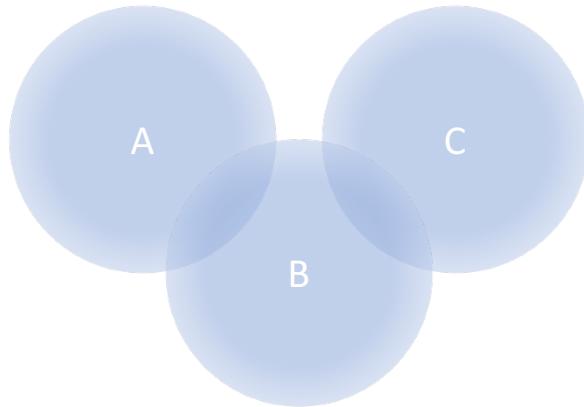


Abbildung 2.7: Veranschaulichung des **Hidden Station** bei drei Nutzern. Abgebildet sind drei Funksender A, B und C und ihre Sendereichweiten

gen Funkreichweiten. Möchte in diesem Szenario Nutzer A an Nutzer

B senden, während Nutzer B gerade inaktiv ist, startet A die Übertragung. Wenn nun gleichzeitig auch Nutzer C an Nutzer B senden möchte, ist dieser nicht in der Lage, die Aktivität von A zu detektieren, da Nutzer C außerhalb der Reichweite von Nutzer A liegt. In der Folge kollidieren die Übertragungen von Nutzer A und Nutzer C.

Ein Ansatz, der diesem Problem entgegenwirkt, ist das Versenden von **RTS**- und **CTS**-Nachrichten. In dem beschriebenen Beispiel würde Nutzer A zunächst eine **RTS**-Nachricht versenden, woraufhin Nutzer B mit einer **CTS**-Nachricht antwortet. Somit findet bei Nutzer B auch eine Aktivität statt, die Nutzer C detektieren kann.

## 2.2 KOMMUNIKATIONSPROTOKOLLE

In den vorangegangen Abschnitten wurden bereits einige Verfahren beschrieben, die für die erfolgreiche Übermittlung von Informationen in einem Funknetzwerk notwendig sind. Dazu kommen ggf. noch weitere Aspekte, wie die Adressierung der Netzwerkeinnehmer oder das Verlegen und Zusammenfügen von größeren Datenmengen in kleinere Pakete, die für eine Übertragung geeignet sind. Diese Aufgaben werden zum Teil von verschiedenen Protokollen übernommen. Um einen Austausch und eine unabhängige Entwicklung dieser Protokolle zu ermöglichen, werden Protokolle nach den Teilbereichen einer Kommunikation eingeordnet. Dazu existiert das **OSI**-Referenzmodell der *International Organization for Standardization*. Das Modell besteht aus aufeinander aufbauenden Schichten, den sogenannten *Layern*. Die sieben Layer des Modells sind in [Abbildung 2.8](#) zu sehen.

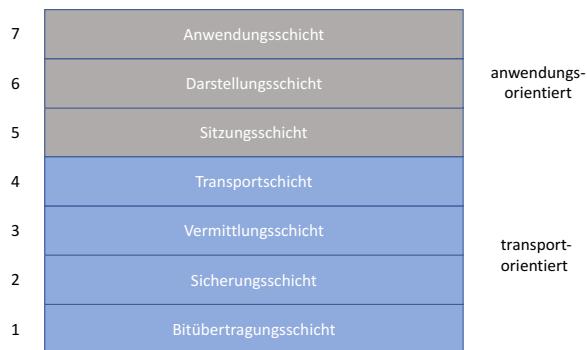


Abbildung 2.8: **OSI**-Referenzmodell

**LAYER 1:** Umwandlung der Bits in ein für den jeweiligen Übertragungskanal geeignetes Signal.

**LAYER 2:** Gewährleistung einer fehlerfreien Übertragung und Regelung des Kanalzugriffs.

**LAYER 3:** Ermittlung eines geeigneten Weges über mehrere Netzwerk-knoten ([Routing](#)).

**LAYER 4:** Zuordnung von Daten zu verschiedenen Anwendungen, welche dieselbe Verbindung nutzen, wie z. B. E-Mail Client und Webbrower.

**LAYER 5-7:** Diese Schichten werden üblicherweise von dem Anwendungsprogramm zusammenhängend abgedeckt. Sie sind nicht mehr direkt auf die Datenübertragung bezogen, sondern beschreiben die Funktionalität der eigentlichen Anwendung.

Es ist zu ergänzen, dass die Sicherungsschicht im Unterschied zum ursprünglichen [OSI](#)-Modell in einigen Bereichen in zwei Unterschichten aufgeteilt wird. Dies ist der Fall, wenn Protokolle notwendig sind, die explizit den Kanalzugriff regeln. Die untere Teilschicht dient dann der [Medienzugriffssteuerung](#), engl. [Media Access Control \(MAC\)](#), während die obere Schicht eine Schnittstelle zur darüber liegenden Vermittlungsschicht beschreibt. Diese wird als [Logical Link Control \(LLC\)](#)-Schicht bezeichnet.

Bei Übertragungen wird der Protokollstapel beim Sender von oben nach unten durchlaufen. Dabei fügen die Protokolle der einzelnen Schichten jeweils Informationen zur eigentlichen Nachricht hinzu, bis hin zur Übertragung des Signals über ein physikalisches Medium, z. B. ein Kabel oder Funkwellen. Am Empfänger durchläuft das Signal den Protokollstapel dann in umgekehrter Reihenfolge von unten nach oben. Dabei entfernen die Protokolle die jeweils zuvor hinzugefügten Informationen und nutzen diese zur Erfüllung ihrer Aufgaben, bis die ursprüngliche Nachricht in der Anwendung des Empfängers angekommen ist. Dazu definieren die einzelnen Schichten Schnittstellen — sogenannte [Service Access Points \(SAPs\)](#). Über diese Schnittstellen kommunizieren die Schichten mit Hilfe der vier folgenden Dienst-Primitiven:

- **REQ:** *Request*, auf Deutsch „Anforderung“
- **IND:** *Indication*, auf Deutsch „Meldung“
- **RESP:** *Response*, auf Deutsch „Antwort“
- **CONF:** *Confirmation*, auf Deutsch „Bestätigung“

[Abbildung 2.9](#) veranschaulicht die Art und Weise, wie diese Primitiven eingesetzt werden. Mit **REQ** fordert Schicht N einen Dienst der darunter liegenden Schicht N – 1 an. Nach der Bearbeitung in Schicht N – 1 antwortet diese mit **CONF**. Andersherum wird eine **IND** immer von der tieferen Schicht N – 1 ausgelöst, um die höherliegende Schicht N über ein Ereignis zu informieren. In diesem Fall antwortet N mit **RESP**. Durchläuft beispielsweise eine Nachricht beim Empfänger den

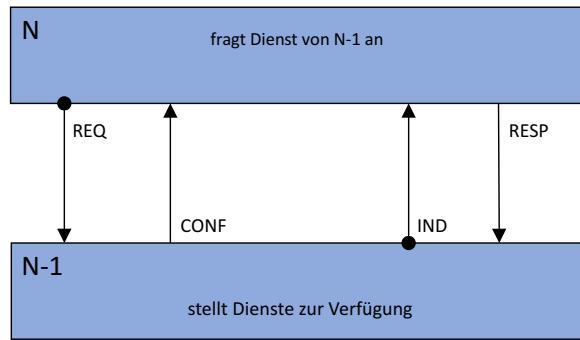


Abbildung 2.9: Mit dem Konzept der Dienst-Primitiven tauschen Protokollschichten Informationen aus

Protokollstapel, wie oben beschrieben, informieren die einzelnen Protokollschichten nach der Verarbeitung der Nachricht die nächst höhere Schicht durch eine **IND** Primitive. Neben den eigentlichen Nutzdaten ist es auch möglich, dass protokolleigene Kommandos über das Medium übertragen werden. Dies ist der Fall, wenn eine Protokollinstanz des einen Gerätes Informationen mit der entsprechenden Protokollinstanz eines anderen Gerätes austauschen muss. Ein Beispiel dafür ist die Anforderung und Zuweisung einer Geräteadresse. Diese Kommandos werden wie gewöhnliche Nutzdaten vom Sender auch an die darunterliegenden Schichten weitergeleitet und über das Medium übertragen. Empfängerseitig durchlaufen sie den Stapel allerdings nur bis zum entsprechenden Protokoll. Auf diese Art werden die vielseitigen Herausforderungen einer erfolgreichen Kommunikation aufgeteilt.

### 2.2.1 Der IEEE 802.15.4 Standard

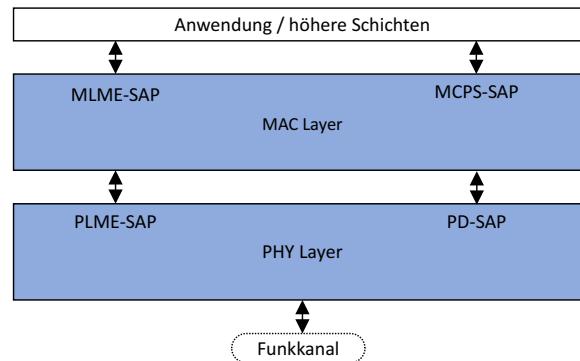


Abbildung 2.10: 802.15.4 Protokollstapel. Der Standard definiert die physikalische Schicht sowie die Kanalzugriffsschicht. Eingezeichnet sind auch die Schnittstellen (**SAPs**) zum Datenaustausch zwischen den Schichten.

Der [IEEE 802.15.4](#) Standard ist ein Protokollstapel für drahtlose Netzwerke mit geringer Datenrate. [WSNs](#) sind ein klassisches Beispiel solcher Netze. [IEEE 802.15.4](#) definiert die physikalische Schicht — den *PHY Layer* — sowie den unteren Teil der Sicherungsschicht — den *MAC Layer*. Die beiden Schichten übernehmen dabei die folgenden Aufgaben:

**MAC LAYER** Hier geschieht die notwendige Organisation der Nutzdaten. Es wird eine Rahmenstruktur definiert, in der die Nutzdaten zusammen mit Steuerinformationen untergebracht werden. Z.B. werden die Nachrichten mit einer Sequenznummer versehen, um ggf. Empfangsbestätigungen zuordnen zu können. Weiterhin ermöglicht das Hinzufügen einer Prüfsumme eine Bitfehlererkennung. Der *MAC Layer* bietet weiterhin die Möglichkeit, periodische Funksignale — so genannte *Beacons* — zu erzeugen, um Netzwerkteilnehmern regelmäßig organisatorische Informationen im Netzwerk bereitzustellen und eine Synchronisation der Teilnehmer zu ermöglichen. Der Standard sieht vor, dass es einen koordinierenden Funknoten im Netzwerk gibt — den [PAN-Coordinator](#). Dieser kann mithilfe der *Beacons* dann z. B. festlegen, dass eine Kommunikation nur innerhalb eines definierten Zeitfensters stattfinden darf und sich daran eine Funkpause anschließt, in der alle Teilnehmer in einen energiesparenden passiven Modus wechseln können. Das Fenster, in dem die Kommunikation stattfindet, wird als [Contention Access Period \(CAP\)](#) bezeichnet. Dazu wird ein [CSMA-CA](#) Algorithmus definiert (vgl. [Unterabschnitt 2.1.4](#) [7]).

**PHY LAYER** Die Aufgabe der physikalischen Schicht im [IEEE 802.15.4](#) Standard ist die Ansteuerung des [Transceivers](#). Dazu gehört die Wahl eines geeigneten Frequenzkanals und die Übertragung der einzelnen Bits unter Berücksichtigung der vom [Transceiver](#) bereitgestellten Modulationsverfahren (vgl. [Unterabschnitt 2.1.3](#)). Daneben stellt der *PHY Layer* Dienste zur Bewertung des Funkkanal zur Verfügung ([CCA](#)). Dieser Dienst wird vom darüber liegenden *MAC Layer* während des [CSMA-CA](#) Algorithmus angefordert. Eine genaue Beschreibung dieses Verfahrens erfolgt in [Abschnitt 4.3](#).

Die Struktur des [IEEE 802.15.4](#) Protokollstapels ist in [Abbildung 2.10](#) dargestellt. Dort sind auch die [SAPs](#) eingezeichnet. Die Nutzdaten werden dabei über den [MAC Common Part Sublayer SAP \(MCPS-SAP\)](#) und den [PHY Data SAP \(PD-SAP\)](#) weitergeleitet. Steuerinformationen und Dienstanfragen werden dagegen über den [MAC Sublayer Management Entity SAP \(MLME-SAP\)](#) und den [Physical Layer Management Entity SAP \(PLME-SAP\)](#) ausgetauscht.

MODUS	STROMAUFNAHME	LEISTUNGS AUFNAHME
Power Down	0,0005mA	0,0015mW
IDLE	1,5mA	4,5mW
TX mit 14dBm	46,0mA	138mW
TX mit 10dBm	36,0mA	108mW
RX	23,5mA	70mW

Tabelle 2.2: Nennwerte der Leistungsaufnahme des CC1200 Funkchips von Texas Instruments bei einer Betriebsspannung von 3,0V und einer Sendefrequenz von 869,5MHz [8]

### 2.3 ENERGIEVERBRAUCH

Einleitend sei hier eine Anmerkung zum Begriff ‚Energieverbrauch‘ angeführt: Energie wird im eigentlichen Sinne nicht verbraucht, sondern umgewandelt - z. B. von mechanischer Energie in elektrische Energie oder von elektrischer Energie in Strahlungsenergie. Dabei wird ein Teil der Energie immer in Wärme umgewandelt, die in der Regel nicht weiter genutzt werden kann. Mit ‚Energieverbrauch‘ ist im Folgenden stets der Bedarf an elektrischer Energie  $E_{el}$  gemeint, der für die Funktion des Systems, speziell des **Transceivers**, notwendig ist.

Energie ergibt sich aus der Leistungsaufnahme über die Zeit:

$$E = P \cdot t \quad (2.14)$$

Die elektrische Leistung  $P_{el}$  ist definiert als Produkt von Spannung und Strom:

$$P_{el} = U \cdot I \quad (2.15)$$

Die SI-Einheit der Leistung ist *Watt*, Sendeleistungen werden oft auch als logarithmisches Maß, bezogen auf ein *Milliwatt*, in *dBm* angegeben. Die SI-Einheit der Energie ist *Joule* (J), die elektrische Energie wird üblicherweise auch in *Wattsekunden* (Ws) bzw. *Milliwattsekunden* (mWs) angegeben.

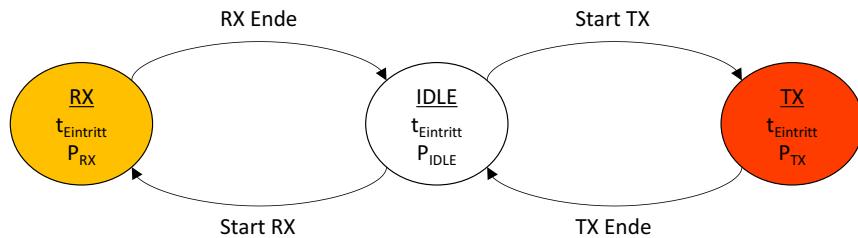
$$1\text{mWs} = 0,001\text{Ws} = 0,001\text{J} \quad (2.16)$$

Bei Funksystemen ist der **Transceiver** die relevante Komponente für den Energieverbrauch. Dieser ist für das Senden und Empfangen der Nachrichten zuständig. Wie in [Unterabschnitt 2.1.1](#) beschrieben, ist für eine erfolgreiche Funkübertragung eine ausreichend hohe Sendeleistung notwendig, aber auch die Verstärkung und Aufbereitung des schwachen Empfangssignals ist mit einer hohen Leistungsaufnahme verbunden. In [Tabelle 2.2](#) sind exemplarisch die Angaben zur Leis-

tungsaufnahme des CC1200 Funkchips von *Texas Instruments* aufgelistet. Die Leistungsaufnahme im Sendemodus (TX) und Empfangsmodus (RX) ist um mehrere Zehnerpotenzen höher als in den inaktiven bzw. datenverarbeitenden Modi (Power Down, IDLE).

### 2.3.1 Energiemodell

Um den Energieverbrauch des **Transceivers** während der Funkkommunikation zu erfassen, kann dieser mit einem Zustandsautomaten modelliert werden, wie in [Abbildung 2.11](#) dargestellt. Zu den einzel-



[Abbildung 2.11](#): Zustandsmodell eines **Transceivers** mit drei Betriebsmodi zur Erfassung des Energieverbrauchs

nen Modi ist jeweils die entsprechende Leistungsaufnahme im Modell hinterlegt. Der **Transceiver** startet im IDLE-Zustand, dabei wird  $t_{\text{Eintritt}}$  auf Null gesetzt. Findet z. B. nach fünf Sekunden der erste Sendevorgang statt, wird zum ersten Mal der Energieverbrauch bestimmt:  $E = P_{\text{IDLE}} \cdot 5\text{s}$  (vgl. [Gleichung 2.14](#)) und das Modell wechselt in den TX-Zustand. Der Zeitpunkt des Zustandswechsels wird in  $t_{\text{Eintritt}}$  gespeichert. Dauert der Sendevorgang beispielsweise zwei Sekunden, wird beim dann folgenden Zustandsübergang zu IDLE der Energieverbrauch aktualisiert:  $E = E + P_{\text{TX}} \cdot 2\text{s}$ . Bei allen weiteren Zustandsübergängen wird analog verfahren. Durch Hinzufügen weiterer Zustände, für z. B. einen *Power Down*- oder Einschwingmodus mit den jeweiligen Leistungsaufnahmen, kann das Modell beliebig verfeinert werden.

### 2.3.2 Energiemessung

Um die tatsächliche Leistungsaufnahme des verwendeten **Transceiver** zu bestimmen, wird der in diesem Abschnitt beschriebene Messaufbau verwendet. Die Leistungsaufnahme kann bei bekannter Betriebsspannung  $U_B$  nach [Gleichung 2.15](#) durch die Messung des Stroms ermittelt werden. Dazu wird hier das *PowerScale* System verwendet, das eine Strommessung im  $\mu\text{A}$  Bereich erlaubt. Dabei wird eine Messeinheit über eine Schnittstelle mit dem zugehörigem Computerprogramm verbunden, wie in [Abbildung 2.12a](#) zu erkennen ist. Der verwendete **Transceiver** ist der CC1200 von *Texas Instruments* (s. [Unterab-](#)

schnitt 2.5.2). Dieser wird ebenfalls über eine Schnittstelle mit einem Diagnoseprogramm auf dem Computer verbunden, um den Chip in den zu vermessenden Modus zu versetzen. Zum Vermessen der Leistungsaufnahme im TX-Modus kann hier die Einstellung der verschiedenen Sendeleistungen vorgenommen werden. Zur Messung der Leistungsaufnahme im RX-Modus ist in Ergänzung zu den Komponenten in Abbildung 2.12a ein weiterer CC1200 notwendig, um Funknachrichten zu erzeugen.

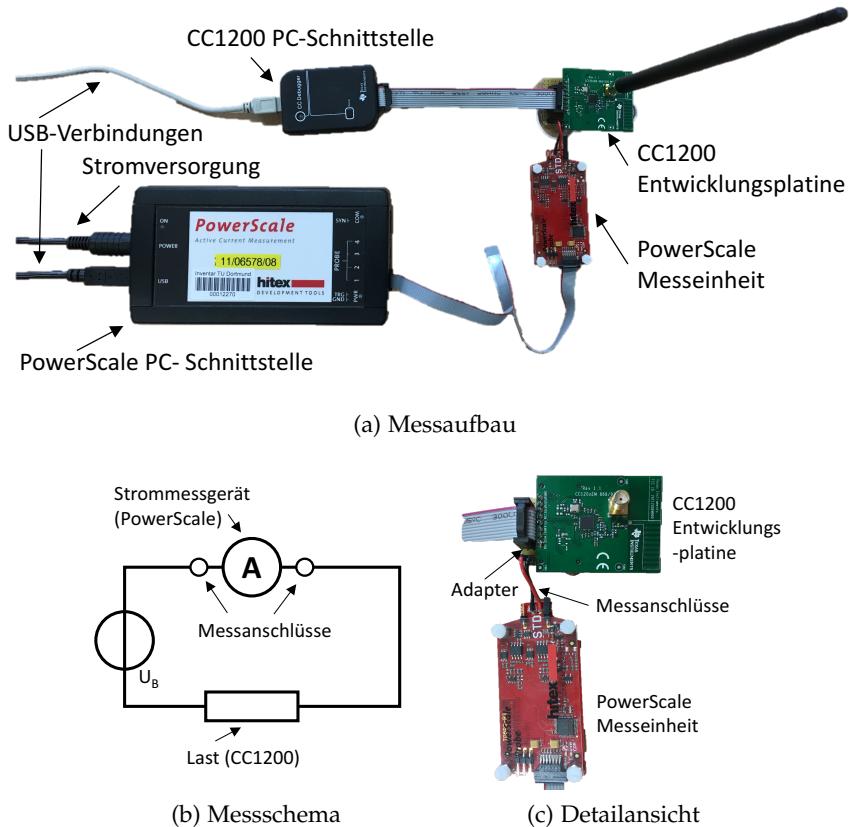


Abbildung 2.12: Aufbau zur Energiemessung des CC1200 Transceivers

Zur Messung des Stroms in einem elektrischen System muss das Strommessgerät in den Stromkreislauf integriert werden. Ein Schaltbild des Messaufbaus ist in Abbildung 2.12b dargestellt. Um diese Integration des PowerScale-Systems zu ermöglichen, ist ein Adapter notwendig, der die Stromzufuhr des glscC1200 unterbricht und die Anschlusspunkte für das Messsystem bereitstellt. Abbildung 2.12c zeigt den Anschluss im Detail. Durch die Verbindungen zur Messeinheit wird die Stromzufuhr zum Funkchip wieder geschlossen.

Das Messprogramm auf dem Computer speichert den Verlauf der momentanen Leistungsaufnahme. Im Anschluss kann der für den jeweiligen Betriebsmodus relevante Zeitbereich ausgewählt werden und das Programm berechnet den Energieverbrauch durch Aufsum-

mierung der gemessenen Leistungswerte für den markierten Zeitraum.

## 2.4 SIMULATIONEN

Simulationen können dann eingesetzt werden, wenn ein System zu komplex für eine konkrete Berechnung ist oder ein Versuchsaufbau unverhältnismäßig aufwendig wäre. Nachdem ein System in einem Simulationsmodell implementiert ist, liegt der Vorteil darin, schnell einen Überblick über die Auswirkungen verschiedener Parametervariationen zu erhalten. Nachteilig ist, dass ein Simulationsmodell in der Regel nicht alle Aspekte des realen Systems berücksichtigt. Je detaillerter das Modell sein muss, desto größer ist der Aufwand der Implementierung.

Im Bezug auf Netzwerke liegt der Vorteil von Simulationen darin, dass an vielen Stellen des Netzwerkes, z. B. an Empfängern in einem Sensornetzwerk, derselbe Prozess stattfindet. Dieser Prozess muss nur einmal formal beschrieben sein und kann in der Simulation dann immer wieder zu entsprechender Zeit durchlaufen werden. Ebenso ist eine Übersicht aller in der Simulation enthaltenen Größen zu jedem Zeitpunkt möglich. Ein weiterer Vorteil besteht in der exakten Reproduzierbarkeit der Untersuchung.

### 2.4.1 Ereignisorientierte Simulation

Bei **ereignisorientierten Simulationen**, engl. *Discrete Event Simulations (DESs)* werden die zu bearbeitenden Ereignisse (engl. *Events*) in eine Liste, die so genannte **Future Event List (FEL)**, eingeordnet und gemäß ihrer zeitlichen Abfolge bearbeitet. Ein einfaches Beispiel für eine **DES** ist der Kassenvorgang an der Supermarktkasse, wie in [Abbildung 2.13](#) angedeutet. Eine entsprechende Simulation könnte ein Modell der einkaufenden Personen, ein Modell der Kasse und ein Modell der Warteschlange beinhalten. Wenn das Personenmodell die Menge an eingekauften Artikeln enthält, das Modell der Kasse die Bearbeitungszeit in Abhängigkeit der Artikelanzahl und das Modell der Schlange die Zahl der wartenden Personen verwaltet, läuft die Simulation wie folgt ab: Das Simulationsprogramm kann nach einer geeigneten Zufallsverteilung „Anstellen-Events“ erzeugen. Ist die Warteschlange leer, wird von dem Modell der Warteschlange direkt ein „Kassieren-Event“ erzeugt. Dadurch wird das Modell der Kasse aufgerufen, das die Bearbeitungszeit berechnet und ein „Verlassen-Event“ an entsprechender Stelle in die **FEL** einfügt. Der Vorteil besteht darin, dass die Simulation nur die einzelnen *Events* berechnen muss, die Zeit dazwischen wird nicht simuliert. Mit diesem Beispiel kann dann z. B. die durchschnittliche Länge der Warteschlange in Abhängigkeit von der Menge der eingekauften Artikel berechnet werden.

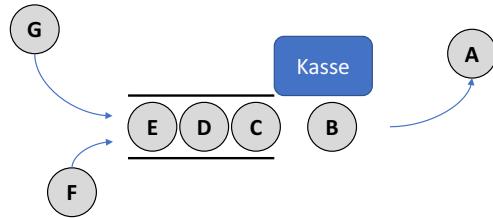


Abbildung 2.13: angedeuteter Kassievorgang als Beispiel für eine ereignisorientierte Simulation

Betrachtet man das genannte Beispiel aus einer abstrakten Sicht, so handelt es sich um ein System, bei dem eine Instanz, hier die Kasse, nach einer festgelegten Art eingehende Aufträge abarbeiten muss. Dieses Muster liegt auch den Vorgängen in Kommunikationsnetzen zugrunde, daher eignet sich eine [ereignisorientierte Simulation, engl. \*Discrete Event Simulation\*](#) besonders gut zur Untersuchung von Netzwerken.

#### 2.4.2 OMNeT++

[OMNeT++](#) ist ein Simulationsframework für [DESS](#), bei dem die Modellierung der einzelnen Komponenten in der Programmiersprache [C++](#) erfolgt. Wesentliche Aufgabe des Frameworks ist die Verwaltung der [FEL](#) und der Aufruf des Programmcodes der einzelnen Komponenten. [OMNeT++](#) bietet zusätzlich eine eigene Programmiersprache — die [NED](#)-Sprache — an. Mit dieser können Gruppierung, Anordnung und Verbindungen der einzelnen Komponenten beschrieben werden. Die wichtigsten Bestandteile von [OMNeT++](#) sind:

**MODULE** Die einzelnen Komponenten der Simulation werden Module genannt. Mit der [C++](#)-Klasse `cModule` stellt das Framework eine Basisklasse zur Implementierung bereit. Während die Implementierung in [C++](#) erfolgt, wird die gegenseitige Beziehung zwischen verschiedenen Modulen in [NED](#) beschrieben. Module können ineinander verschachtelt werden und besitzen Verbindungen zu anderen Modulen.

**GATES** Um *Events* an andere Module zu senden, besitzen Module *Gates*. Diese sind die eindeutigen Endpunkte einer Verbindung zwischen zwei Modulen. Schickt ein Modul ein *Event* an ein *Gate*, wird es genau dem Modul zugestellt, welches mittels der [NED](#)-Sprache damit verknüpft ist. Module können mehrere *Gates* besitzen. Außerdem können *Events* direkt an ein *Gate* eines anderen Moduls geschickt werden, ohne eine zuvor festgelegte Verbindung. Dies ist z. B. für große Netzwerke mit variabler Anzahl an Teilnehmern sinnvoll.

**MESSAGES** Events werden in **OMNeT++ Messages**, auf Deutsch „Nachrichten“, genannt. Im Zusammenhang mit Netzwerksimulationen ist anzumerken, dass dabei nicht zwangsläufig Nachrichten im Sinne des Netzwerkes gemeint sind, sondern dass es sich hierbei um jegliche Art von *Event* in der Simulation handeln kann, z. B. ein *Timer-Event*.

## 2.5 HARDWARE

In diesem Kapitel werden die Hardwarekomponenten vorgestellt, die zur prototypischen Implementierung des Systems verwendet werden. Die für die Funkkommunikation zentrale Komponente ist der CC1200 Funkchip von Texas Instruments. Für die Implementierung des Kanalzugriffprotokolls und zur Ansteuerung des Funkchips kommt der MSP430FR9596 Mikrocontroller von Texas Instruments zum Einsatz. Für beide Chips steht eine Entwicklungsplatine zur Verfügung. Ent-

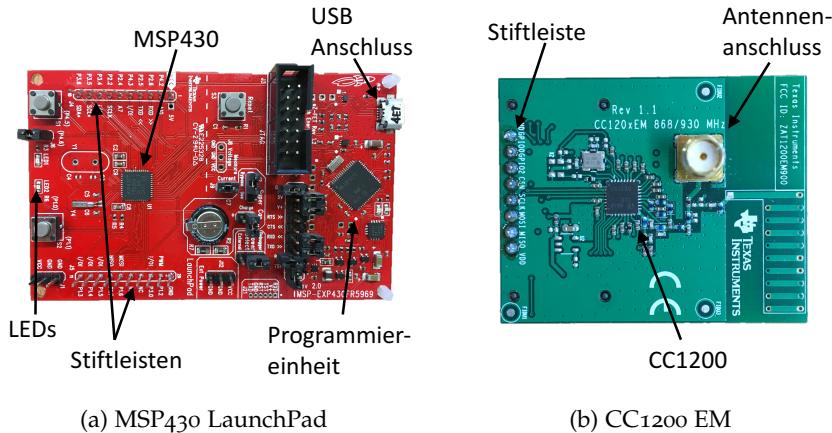


Abbildung 2.14: Entwicklungsplatten für Mikrokontroller und **Transceiver** zur prototypischen Anwendungsimplementierung

wicklungsplatten vereinfachen die Entwicklung von Anwendungen, indem sie die nötige Beschaltung der Chips mit zusätzlichen Bauelementen realisieren und wichtige Schnittstellen zur Verfügung stellen. Abbildung 2.14a zeigt das MSP430 LaunchPad. Dieses verfügt neben zwei LEDs und zwei Tastern über Elektronik zur Programmierung des Mikrocontrollers per USB Verbindung. Daneben sind einige der GPIO-Pins zur einfachen Verdrahtung an Stiftleisten herausgeführt. In Abbildung 2.14b ist die CC120x EM Platine zu sehen. Darauf ist der CC1200 Funkchip mit den entsprechenden externen Kondensatoren und Widerständen versehen, die für exakte Frequenzerzeugung notwendig sind. Weiterhin ist ein coaxialer Antennenanschluss vorhanden. An der Stiftleiste auf der linken Seite befinden sich die Signalleitungen zur Ansteuerung des Funkchips.

Modell	MSP430FR9596
Hersteller	Texas Instruments
Betriebsspannung	1,8V - 3,6V
Architektur	16 Bit RISC
Taktrate	32kHz - 24MHz
Speicher	64kB FRAM

Tabelle 2.3: Eigenschaften des MSP430 Mikrocontrollers [9]

Die folgenden Abschnitte beschreiben die beiden Hardwarekomponenten im Detail und geben wichtige Hinweise für die Implementierung.

#### 2.5.1 MSP430

Mikrocontroller vereinen einen Prozessor (**CPU**), Speicher und weitere Peripherie, wie Timer, acfadc und serielle Kommunikation in einem Chip. Die einzelnen Pins des Mikrocontrollers können zur Ansteuerung von Aktoren oder zum Auslesen von Sensoren genutzt werden. Die Pins werden als *General Purpose Input/Output* (**GPIO**)-Pins bezeichnet. Die Mikrocontroller der MSP430 Reihe sind darüber hinaus speziell für Anwendungen mit begrenzten Energieressourcen konzipiert. Tabelle 2.3 zeigt die Hauptmerkmale des verwendeten Modells.

Die Programmierung erfolgt üblicherweise in der Programmiersprache **C**. Für die Konfiguration und Ansteuerung der einzelnen Peripheriekomponenten sind spezielle Register im Speicher des Mikrocontrollers vorgesehen. Obwohl sie sich für das Programm wie normale Speicheradressen verhalten und auch so beschrieben oder gelesen werden können, steuern die Bits dieser Register das Verhalten der integrierten Peripherie. So bestimmt beispielsweise der Wert in einem **GPIO**-Register die Spannungspegel der Pins am entsprechenden Port. Andersherum kann aus Registern, die dem **ADC** zugeordnet sind, das Ergebnis der Analog-Digital Wandlung entnommen werden. Da auf Mikrocontrollern in vielen Fällen kein Betriebssystem läuft, das einzelne Programmstarts verwaltet, wird ein Programm für Mikrocontroller als Endlosschleife programmiert, wie in Quelltext 2.1 angedeutet.

Da die Entwicklung des Programms nicht auf dem Mikrocontroller selbst stattfindet, wird eine Software benötigt, die das **C**-Programm für die Verwendung auf dem Chip kompiliert. Im Anschluss wird der Quellcode mit einem *Programmer* auf den Chip übertragen. Auf dem *LaunchPad* ist dieser, wie eingangs erwähnt, bereits vorhanden.

---

```
// Code, der zu Beginn einmal durchlaufen wird
// zB. Konfigurationen der Peripherie

while(1) {
    // hier erfolgt die Implementierung der Anwendung
}

// Programmende wird nie erreicht
```

---

Quelltext 2.1: Strukturbispiel eines Programms für Mikrocontroller

Für die Realisierung von Anwendungen mit geringer Leistungsaufnahme kann der MSP430 in sieben unterschiedliche Energiesparmodi versetzt werden, bei denen stufenweise einzelne Komponenten und Taktquellen abgeschaltet werden. Bei der Rückkehr aus den Modi 3.5 und 4.5 bleiben alle Pins des Controllers durch das Kontrollbit `LOCKLPM5` im `PM5CTL0` Register gesperrt, um zunächst alle Konfigurationen vorzunehmen und die Pins anschließend freizugeben. Dieser Zustand tritt auch nach jedem Systemstart ein. Um die Pins des MSP430 nutzen zu können, muss dieses Bit zu Beginn des Programms auf Null gesetzt werden.

Eine weitere Besonderheit beim MSP430 ist, dass der [Watchdog-Timer](#) standardmäßig aktiviert ist. Ein [Watchdog-Timer](#) ist ein Zähler, der unabhängig von der [CPU](#) hardwareseitig in einigen Mikrocontrollern integriert ist und bei Erreichen eines eingestellten Wertes einen Neustart des Systems auslöst, so er nicht regelmäßig vor seinem Ablauf durch die Software zurückgesetzt wird. Diese Technik dient dem automatischen Neustart des Controllers, für den Fall eines unerwarteten Fehlverhaltens. Es ist beim MSP430 daher darauf zu achten, den [Watchdog-Timer](#) entsprechend regelmäßig zurückzusetzen oder ganz zu deaktivieren.

Drei für die Programmierung des Funkchips und die Implementation des Kanalzugriffverfahrens wichtige Komponenten sind die Interrupt-Behandlung, die Verwendung von *Timern* und die [SPI](#) Kommunikation.

**INTERRUPTS** Um auf Ereignisse Ereignisse, wie das eintreffen eines Bytes über die serielle Schnittstelle, die Änderung des Spannungspiegels an einem Pin oder den Ablauf eines *Timers* reagieren zu können, kann der normale Programmablauf durch einen Interrupt unterbrochen werden. Tritt ein Interrupt ein, wird zunächst das zugehörige Interrupt /emphFlag zur Signalisierung gesetzt. Ist der jeweilige Interrupt softwareseitig aktiviert, beginnt der Mikrocontroller mit der Abarbeitung einer separaten Programmsequenz - der [Interrupt Service Routine \(ISR\)](#). Die Startadresse dieses Programmabschnitts ist in der Interruptvektor Tabelle gespeichert. Der MSP430 verfügt über 26 ver-

schiedene Interruptvektoren, die zum Großteil den Peripheriekomponenten zugeordnet sind. Einigen Komponenten sind auch zwei Vektoren zugeordnet. So können individuelle Behandlungsroutine (**ISRs**) geschrieben werden, ohne erst die Quelle des Interrupts ermitteln zu müssen. Dies führt zu einer schnelleren Behandlung des Ereignisses. Ist die **Interrupt Service Routine** abgearbeitet, wird mit dem normalen Programmablauf fortgefahrene.

Bei der Verwendung von Interrupts ist darauf zu achten, dass es nicht zu unkontrollierten Manipulationen von gemeinsam genutzten Daten kommt. Durchläuft das Hauptprogramm z. B. ein Array, in dem die Daten der letzten seriellen Übertragung gespeichert sind, muss sicher gestellt werden, dass diese Daten bis zum Ende des Durchlaufs nicht von einer **ISR** überschrieben werden.

**TIMER** Der Grundbaustein eines *Timers* ist ein Zählregister, dessen Wert in definierten Abständen inkrementiert wird. Diese Abstände hängen mit dem Systemtakt zusammen. Die Taktrate des Zählers lässt sich über Verteiler bezogen auf die Systemtaktrate einstellen. Durch diese Einstellung wird die minimal zu messende Zeit des *Timers* festgelegt. Die Größe des Zählregisters bestimmt die Auflösung des *Timers*, also die Anzahl von Schritten bis zum Erreichen des Höchstwertes und damit zum Ablauf *Timers*. Die Funktionsweise ist in [Abbildung 2.15](#) für eine Auflösung von vier Bit, also acht Zählwerten, verdeutlicht. Die Auflösung beim MSP430 beträgt 16 Bit. Da die Einstellung der Taktrate  $f_{\text{Timer}}$  nur in bestimmten Schritten

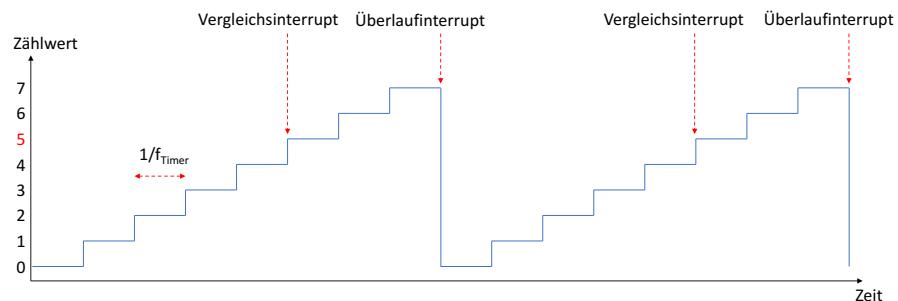


Abbildung 2.15: Funktionsweise eines 4-Bit *Timers*. Der rotmarkierte Wert kennzeichnet den eingestellten Vergleichswert.

möglich ist, bietet der MSP430 zusätzlich die Möglichkeit, einen Interrupt auszulösen, wenn ein definierter Vergleichswert erreicht wird, in [Abbildung 2.15](#) in rot markiert. Dieser Wert kann über ein Register eingestellt werden und erlaubt so eine sehr feine Einstellung des Interruptzeitpunkts.

**SPI** Das **Serial Peripheral Interface** (SPI) des Mikrocontroller bietet die Möglichkeit eine synchrone serielle Buskommunikation mit Peripheriegeräten, wie z. B. dem CC1200 Funkchip, herzustellen. Die

Kommunikation wird dabei stets von einem Gerät initialisiert, dem *Master*. Die angeschlossenen Geräte werden als *Slaves* bezeichnet. Dazu werden drei gemeinsame Signalleitungen benötigt und zusätzlich eine Leitung pro *Slave*:

1. **MOSI Leitung**
2. **MISO Leitung**
3. **SCL Leitung**
4. **CS Leitung**

Zu Beginn der Kommunikation muss der *Master* die beteiligten Kommunikationspartner aktivieren, dies geschieht über die **CS Leitung**, engl. *Chip Select*. Anschließend erzeugt der *Master* auf der **SCL Leitung** regelmäßige Signalflanken, engl. *Serial Clock*. Mit dieser Frequenz wird dann der Pegel der **MOSI Leitung** entsprechend der zu übertragenen Bits auf Null oder Eins geändert, sodass der *Slave* diese bei jeder Taktflanke abtasten kann, engl. *Master Out Slave In*. Gleichzeitig kann der *Slave* über die **MISO Leitung** Daten an den *Master* in analoger Weise senden, engl. *Master In Slave Out*. Der MSP430 realisiert diese Bussystem in einer Hardwareeinheit zur seriellen Kommunikation. Nach entsprechender Konfiguration, kann ein zu sendendes Byte in dafür vorgesehenes Register geschrieben werden. Der Controller startet dann automatisch den Kommunikationsvorgang. Die übertragenen Daten des *Slaves* werden ebenfalls in einem Register abgelegt und können ausgelesen werden. Nach vollständiger Übertragung kann ein Interrupt ausgelöst werden. Ein Mitschnitt der **SPI** Kommunikation zwischen MSP430 und dem CC1200 Funkchip ist in [Abbildung 2.17](#) zu sehen.

### 2.5.2 CC1200 Funkchip

Der CC1200 Funkchip von Texas Instruments ist ein Transceiver für den Sub-GHz Bereich mit umfassenden Konfigurationsmöglichkeiten zur Sendefrequenz, Modulation (vgl. [Unterabschnitt 2.1.3](#)), zur Paketverarbeitung und Kanalprüfung. Die wesentlichen Merkmale sind in [Tabelle 2.4](#) aufgelistet. Einen besonders sparsamen Betrieb erreicht der CC1200 durch den sogenannten *Sniff*-Modus. [Abbildung 2.16](#) veranschaulicht die Vorteile gegenüber dem normalen Empfangsmodus. Der **Transceiver** fügt vor den Nutzdaten eine Präambel und ein Synchronisationsbyte ein, um den Beginn der Nachricht zu markieren und die Empfangselektronik kalibrieren zu können. Der CC1200 ist in der Lage, den Beginn einer Nachricht nach vier Präambel-Bits zu detektieren. Des Weiteren benötigt der Chip nur ca. 0,6ms, um vom Ruhemodus in den Empfangsmodus zu wechseln [8]. Diese Eigenschaften erlauben es im *Sniff*-Modus in regelmäßigen Abständen

Modell	CC1200
Hersteller	Texas Instruments
Betriebsspannung	3,3V
primäre Frequenzen	169MHz, 433MHz, 868MHz, 915MHz
maximale Sendeleistung	16dBm
Empfindlichkeit	-109dBm bei 50kBits/s
maximale Datenrate	1,25MBit/s
Sende-/Empfangsspeicher	je 128 Byte

Tabelle 2.4: Eigenschaften des CC1200 Funkchips [9]

kurz in den Empfangsmodus (RX) zu schalten, während der Chip zum größten Teil im energiesparenden Ruhemodus verbleibt. Dazu muss die Präambellänge deutlich erhöht werden, um kein Paket zu verpassen, wie in Abbildung 2.16b zu erkennen ist. Sie muss so groß sein, dass im dargestellten ungünstigsten Fall — das Paket beginnt genau nachdem der Chip gerade in den Ruhemodus gewechsel ist — noch mindestens vier Präambel-Bytes beim nächsten Wechsel zu RX detektiert werden können. Es ist noch anzumerken, dass dieses Verfahren nur für den Empfang von Paketen vorgesehen ist. Zum Zweck der Kanalüberprüfung ist der kontinuierliche RX-Modus nötig.

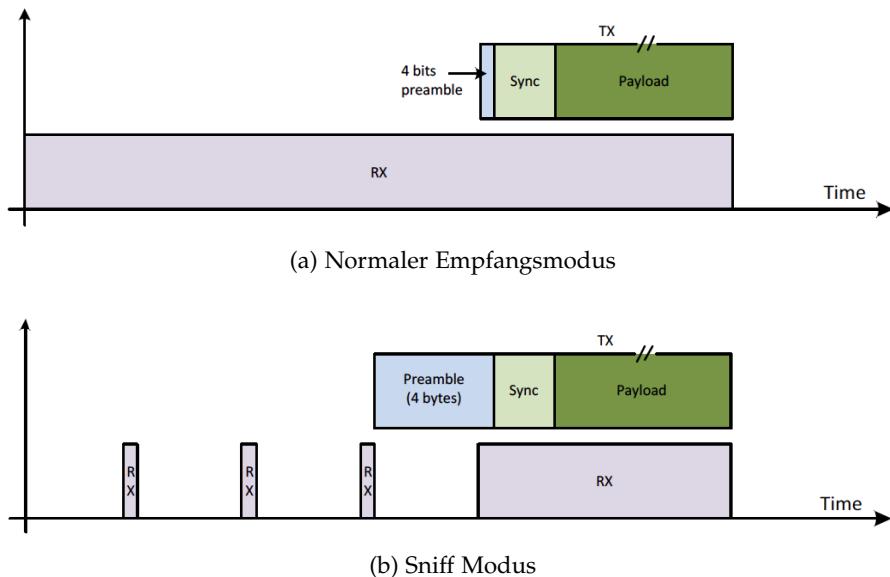


Abbildung 2.16: Auszug aus dem CC1200 User's Guide zur Veranschaulichung des energiesparenden Sniff-Modus [10]

Die Ansteuerung des CC1200 Funkchips erfolgt über SPI. Der Informationsaustausch zwischen Mikrocontroller und CC1200 lässt sich dazu in drei Rubriken einteilen:

1. Schreiben bzw. Lesen der insgesamt 209 Konfigurationsregister zur Einstellung der Sendefrequenz, Modulationsart, Paketlänge, etc..
2. Senden von Kommandos, den so genannten *Strobes*, um den Funkchip in den gewünschten Modus zu versetzen.
3. Austausch von Sende- und Empfangsdaten.

Die Art der Informationsübertragung wird durch das erste Byte bestimmt, welches der Mikrocontroller an den Funkchip sendet. Letzterer erwartet dann weitere Informationen in Form von Bytes in einem festgelegten Muster. Wird durch das erste Byte signalisiert, dass der Chip seinerseits Daten über den Bus senden soll, gilt dies analog. Für den Fall, dass keine Daten vom Chip angefordert werden, sendet dieser bei jeder Übertragung ein Statusbyte an den Mikrocontroller, aus dem der aktuelle Modus und etwaige Fehlermeldungen entnommen werden können. Diese individuellen Übertragungsmuster erfordern, dass die [CS Leitung](#) während des ersten Bytes und allen evtl. folgenden Bytes kontinuierlich durch den Mikrocontroller auf Null-Pegel gehalten wird. Dies ist bei Konfiguration des MSP430 zur automatischen SPI Steuerung (vgl. [Unterabschnitt 2.5.1](#)) nicht gegeben, da die [CS Leitung](#) dabei nach jedem Byte wieder deaktiviert wird. Um die Übertragung nicht abzubrechen, muss der Pin für die [CS Leitung](#) separat durch die Software angesteuert werden. Beispielhaft ist in [Abbildung 2.17](#) der Schreibvorgang in ein Konfigurationsregister des CC1200 zu sehen, das sich im erweiterten Adressraum befindet. In diesem Fall signalisiert das erste Byte lediglich, dass es sich um einen Zugriff auf den erweiterten Adressraum handelt. Der Funkchip erwartet dann die Adresse im zweiten übertragenen Byte und schließlich den Wert, der an diese Adresse geschrieben werden soll.

## 2.6 ENTWICKLUNGSSOFTWARE

Neben der bereits in [Unterabschnitt 2.4.2](#) beschriebenen Simulationsumgebung [OMNeT++](#) und dem in [Unterabschnitt 2.3.2](#) vorgestellten *PowerScale* Systems sind noch weitere Entwicklungs- und Diagnosewerkzeuge für diese Arbeit notwendig. Deren Aufgaben im Rahmen dieser Arbeit werden hier aufgeführt.

**CODE COMPOSER STUDIO** Eine Entwicklungsumgebung zur Programmierung des MSP430 Mikrocontrollers. Neben der Verwaltung der einzelnen Quellcodedateien und der in [Unterabschnitt 2.5.1](#) genannten Übersetzung des C-Codes in Maschinenbefehle, bietet diese Software die Möglichkeit zum *Debugging*, auf Deutsch „Fehlersuche“. Dabei kann der auf den Mikrocontroller geladene Programmcode schrittweise ausgeführt werden. Außerdem können die Registerwerte und Variablen ausgelesen werden. Die

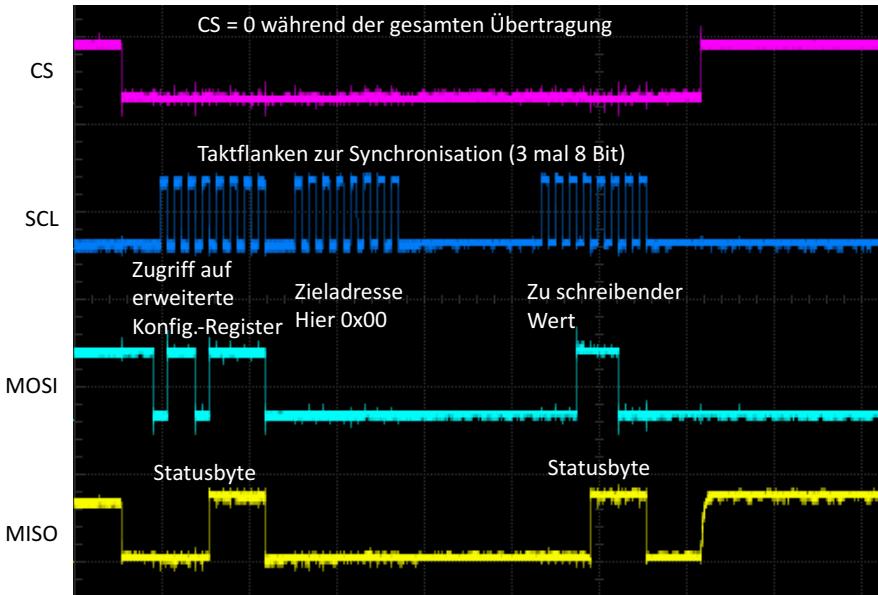


Abbildung 2.17: Mitschnitt der SPI-Kommunikation zwischen MS430 und CC1200. Zu sehen ist der Schreibvorgang eines Konfigurationsregisters des CC1200 im erweiterten Adressraum.

Möglichkeit des *Debuggings* muss von der Hardware unterstützt werden. Das *LaunchPad* bietet diese Möglichkeit.

**RF STUDIO / PACKET SNIFFER** Zwei Softwarewerkzeuge die über eine Hardwareschnittstelle — dem *CC12xx Debugger* — die Register des CC1200 Funkchips direkt schreiben und auslesen können. Zum einen kann der Funkchip so auch ohne Mikrocontroller konfiguriert und in Betrieb genommen werden, zum anderen ist es möglich, empfangene Daten auf dem Computer anzeigen zu lassen und einige Verarbeitungsinformationen des CC1200 auszulesen. Dazu gehören die Empfangsqualität, das Ergebnis der Bitfehlerprüfung sowie der Zeitstempel von empfangenen Nachrichten.

**HTERM** Ein Terminalprogramm zur byteweisen seriellen Datenübertragung. Dieses wird benutzt, um Informationen mit dem Mikrocontroller auszutauschen. Dieser Informationsaustausch bezieht sich nicht auf die Programmierung des Controllers, sondern auf den Austausch von Daten aus dem laufenden Programm heraus. Dabei besteht die Möglichkeit, die Daten sowohl in Textform, als auch in binärer oder hexadezimaler Darstellung einzugeben und anzeigen zu lassen.



# 3

---

## VERWANDTE ARBEITEN

---

Die Untersuchung von **WSN** und den damit verbundenen Fragestellungen ist Forschungsgegenstand auf vielen verschiedenen Gebieten. Dieses Kapitel gibt einen Überblick über verschiedene Anwendungsfelder sowie zu Untersuchungen spezieller Aspekte. Anschließend werden drei Projekte mit besonderer Relevanz für die vorliegende Arbeit genauer beschrieben.

### 3.1 ÜBERSICHT

Obwohl Simulationen zur Erforschung von **WSNs** wegen der großen Anzahl unterschiedlicher Einflüsse nur bedingt zu umfassenden Ergebnissen führen können, sind sie dennoch wichtiger, begleitender Bestandteil vieler Untersuchungen.

**IEEE 802.15.4 MODELL** Ein Modell für die Simulationsumgebung **OMNeT++** aus dem Jahr 2007 [11]. Der Fokus dieser Implementierung liegt auf der Modellierung des **CSMA-CA** Algorithmus

**STANDALONE IEEE 802.15.4 MODELL** Eine nahezu vollständige Modellierung des **IEEE 802.15.4** Protokolls für **OMNeT++** [12]. Neben den eigentlichen Protokollschichten, ist die Einbindung einer Anwendungsschicht bereits vorgesehen. Die Implementierung berücksichtigt einige Neuerungen des Protokolls im Vergleich zum zuvor genannten Modell.

**BATTERIEMODELL** Zur genaueren Abbildung der Batterielaufzeit energiesensitiver Geräte innerhalb eines Simulationsszenarios wird in [13] ein Batteriemodell vorgestellt, welches die komplexen elektro-chemischen Vorgänge berücksichtigt.

Um den Einschränkungen und Vereinfachungen bei simulativen Untersuchungen von **WSN** entgegen zu wirken, gibt es eine Vielzahl von Versuchsplattformen, so genannten *Testbeds*. Dort können die zu untersuchenden Systeme unter Real- bzw. Laborbedingungen erprobt werden.

**MOTELAB** Dieser Versuchsaufbau wurde 2005 an der Universität von Harvard errichtet und ist eine der ersten und am längsten betriebenen Untersuchungsplattformen für Sensornetze [14]. Diese

besteht aus 190 drahtlos verknüpften **Nodes**. Dabei kommt die „MicaZ“-Hardwareplattform zum Einsatz. Diese unterstützt u.a. Licht-, Temperatur-, Luftdruck-, und Beschleunigungssensoren und nutzt das **ZigBee** Protokoll im 2,4GHz Band zur Kommunikation. Zusätzlich sind die **Nodes** kabelgebunden mit einem zentralen Server verbunden. Dieser ermöglicht es, die **Nodes** zu programmieren und Daten mitzuschreiben. Über eine Webschnittstelle können Benutzer auf das Testbed zugreifen und individuelle Anwendungen darauf testen [15].

**KANSEI** Diese Plattform setzt sich aus drei Testsystemen zusammen. Hauptbestandteil bildet ein statisches Netz mit 210 **Nodes** in einem Labor. Dazu kommt ein Satz an portablen Sensorknoten, der je nach Anwendungsfall zur Messgrößenerfassung an bestimmten Orten eingesetzt werden kann. Des Weiteren gibt es **Nodes**, die an mobilen Robotern angebracht sind [16]. Die **Nodes** sind neben Licht-, Temperatur- und Infrarotsensoren auch mit einem Magnetometer und einem Mikrofon ausgestattet. Die Kommunikation findet im 433MHz Band statt.

**WISEBED** Insgesamt 750 **Nodes** sind Bestandteil dieses Testbeds. Dabei sind sie in mehrere Gruppen aufgeteilt, die sich an neun verschiedenen Orten in Europa befinden. Die Sensorhardware der einzelnen Gruppen ist unterschiedlich. Alle Sensorknoten können Umgebungsmesswerte wie Licht und Temperatur erfassen, einige besitzen darüber hinaus noch Beschleunigungssensoren. Mit der Ausnahme von zwei Hardwareplattformen, die im 869MHz Band operieren, nutzen die Sensoren das 2,4GHz Band. Die Teilnetze sind über das Internet miteinander verbunden, sodass sich ein hierarchisches **WSN** ergibt [17]. Die Auswirkungen mehrerer Netzwerkebenen lassen sich so untersuchen.

**WSNs** werden in verschiedenen Anwendungsfällen eingesetzt. Genauso vielfältig sind die Anforderungen an die Netze in den einzelnen Szenarien. Forschungen an Sensornetzen in ihrer vorgesehenen Umgebung sind daher die logische Folge.

**VINELAB** In [18] wird ein Sensornetzwerk zur intelligenten Gebäudesteuerung untersucht. Dazu sind 48 **Nodes** innerhalb eines Gebäudes verteilt. Die Messung von Temperatur, Licht und Luftfeuchtigkeit an den einzelnen Stellen erlaubt die Erstellung einer Gradientenkarte.

**CITY SENSE** Hier sind 100 **Nodes** vorgesehen, die in der Stadt Cambridge an Laternen und Häusern verteilt werden, um Wetterbedingungen und Luftverschmutzung zu überwachen. Dieses Netzwerk nutzt dabei den **WLAN** Standard **IEEE 802.11g** zur Kommunikation [19].

**OULU SMART** Dieses Projekt erprobt die Integration von Computersystemen in innerstädtischen Gebieten in den Bereichen Kommunikation, Information und Mensch zu Computer Interaktion. Das System in der nordfinnischen Stadt Oulu besteht aus drei Komponenten. Das *panOULU WLAN* bildet aus mehreren **WLANs** eine das gesamte Stadtgebiet überspannende Struktur zum Internetzugang. Zudem stellt *panOULU BT* eine Gruppe von **Bluetooth** Netzwerken zur Verfügung. Durch die Kenntnis über den Ort der Zugangspunkte und die begrenzte Reichweite von einigen zehn Metern können hiermit ortsbezogene Dienste angeboten werden. Die dritte Komponente stellt das *panOULU WS* dar — ein **WSN**, das die gesammelten Messdaten an großen Bildschirmen öffentlich zugänglich macht. Dabei kommt das **IEEE 802.15.4** Protokoll sowie **6LoWPAN** zum Einsatz. Die Messdaten können über mehrere **Nodes** hinweg zum **AP** übertragen werden [20].

**GREENORBS** Im Rahmen von *GreenOrbs* werden mit 330 Sensorknoten Informationen zur Forstbeobachtung gesammelt. Dabei werden die Messdaten aller **Nodes** an einer zentralen Datensenke gesammelt. Die Studie [5] zu dieser Struktur wird in [Abschnitt 3.2](#) näher beschrieben.

Neben den grundlegenden Untersuchungen in Laborprüfständen und der realen Umgebung, werden auch technologiespezifische Themen untersucht.

**COLLECTION TREE PROTOKOLL** In [21] werden umfassende Untersuchungen zum **Routing** vorgenommen. Ausgehend von der Annahme, dass Daten an verschiedenen Stellen, in verschiedenen **WSNs** gesammelt werden und über weitere übergeordnete Netze weitergeleitet werden, wird hier ein Protokoll zu diesem Zweck untersucht. Dazu werden Messungen auf 13 verschiedenen Testbeds vorgenommen.

**TOPOLOGIE KONTROLLE** Um in Sensornetzen mit dichter Anordnung der **Nodes** — gerade auch in industriellen Anwendungen — die nötige Sendeleistung der einzelnen Teilnehmer zu senken, wird in [22] eine Technik zur optimalen Auslegung der Netzwerktopologie vorgestellt.

Die Energieeffizienz ist, wie eingangs beschrieben, ein wesentliches Kriterium bei der Entwicklung zukünftiger drahtloser Sensornetze. Daher gibt es auch Untersuchungen, die diesen Aspekt in besonderer Weise aufgreifen.

**FLOCKLAB** In diesem Testbed mit 21 **Nodes** sind diese jeweils auf eine spezielle Überwachungseinheit aufgesteckt. Ziel ist die exakte Überwachung der Software und des Energieverbrauchs. Eine

zeitlich genaue Überwachung einzelner Pins der Sensorhardware ist möglich [23].

**SENSELAB** Auch in diesem Testbed ist die Überwachung des Energieverbrauchs an jedem **Node** möglich. Das Netzwerk umfasst 1024 Messpunkte, die auf vier Stellen in Frankreich verteilt sind und über das Internet miteinander in Verbindung stehen. Die Sensorhardware ist so ausgelegt, dass die Internetverbindung sowohl kabelgebunden, als auch drahtlos per **WLAN** erfolgen kann. [24].

**XMAC** Hierbei handelt es sich um eine Erweiterung des **IEEE 802.15.4** Standards hinsichtlich der Verwendung auf Geräten mit extrem reduzierter Leistungsaufnahme [4]. Durch eine effizientere Präambeldetektion wird die Zeit, die der Empfänger aktiv auf dem Kanal lauschen muss, deutlich reduziert.

Abschließend sind im Folgenden drei Arbeiten aufgeführt, die sich speziell mit energieeffizienten, vernetzten Logistiklagern beschäftigen:

**INBIN** In [2] wird ein energieautarkes Ladehilfsmittel vorgestellt. Der Schwerpunkt liegt dabei auf der Erzeugung der nötigen Energie durch **Energy Harvesting**

**INBIN TESTBED** [3] befasst sich mit der Untersuchung zur Leistungsfähigkeit eines vernetzten Logistiklagers. Eine genaue Beschreibung folgt in [Abschnitt 3.2](#)

**PHYNETLAB** Das in [25] vorgestellte Testbed ist darauf ausgerichtet, industrielle Anforderungen in Ergänzung zu typischen **WSN**-Tests zu unterstützen. Exemplarisch wird die Leistungsfähigkeit eines gängigen Kanalzugriffsverfahrens evaluiert. Eine genauere Beschreibung folgt ebenfalls in [Abschnitt 3.2](#)

### 3.2 DREI PROJEKTE IM DETAIL

Drei Arbeiten, die relevante Aspekte für die Untersuchungen in dieser Arbeit enthalten, werden an dieser Stelle noch einmal ausführlicher beschrieben. Die erste beschäftigt sich mit Skalierungseffekten von **WSNs** im Allgemeinen, während die weiteren den Anwendungsfall in Logistiklagern berücksichtigen.

**GREENORBS** Die in [5] beschriebene Studie untersucht die Skalierungseffekte eines Langzeit-Sensornetzwerks unter Realbedingungen am Anwendungsbeispiel des *GreenOrbs* Projektes, wie zuvor erwähnt. Es handelt sich um ein nicht-hierarchisches Netzwerk, bei dem die Informationen per Multi-Hop von den einzelnen **Nodes** zur Datensonne geleitet werden. Die Untersuchungen erstrecken sich daher von

Routing-Anforderungen bis hin zur Analyse einzelner Verbindungen im Hinblick auf Signalstärke und Paketverlustrate. Die generierten Daten pro **Nodes** variieren von drei Paketen in einer Stunde bis zu drei Paketen in 200 Sekunden. Der Kanalzugriff erfolgt dabei nach einem Standard **CSMA-CA** Verfahren<sup>1</sup>. Eine Betrachtung des Energieverbrauches erfolgt nicht. Die Studie listet drei verschiedene Ursachen für verlorene Pakete auf:

- Übertragungsabbrüche
- Empfangsspeicher-Überlauf
- Sendespeicher-Überlauf

Erstere sei mit 61,08 Prozent die wesentliche Ursache [5]. Bezogen darauf stellt sich die hohe Anzahl an Kollisionen als wesentliches Problem heraus. In dem Zusammenhang wird auch auf das „Hidden-Station-Problem“ hingewiesen [5]. Zusammenfassend liege das Problem in der Parallelität der Netzwerkoperationen und müsse beim Entwurf skalierbarer Netzwerkprotokolle weiter untersucht werden [5].

**INBIN TESTBED** Die Leistungsstärke von **WSNs** im Kontext eines Logistiklagers ist Untersuchungsgegenstand in [3]. Die Bedeutung einer energieeffizienten und zuverlässigen Funkkommunikation wird hierbei herausgestellt. Dazu formulieren die Autoren eine Kommunikationsinfrastruktur mit Stern-Topologie als passend für die Anwendung in einem Logistiklager. Alle **Nodes** kommunizieren direkt mit einem zentralen Server, eine Kommunikation zwischen den **Nodes** erfolgt nicht. Weiterhin werden zwei verschiedene Arten von Datenverkehr benannt.

- Hintergrund-Datenverkehr
- Datenverkehr bei Transportanfragen

Während ersterer in regelmäßigen Abständen von den Frachtcontainern selbst initiiert wird, beschreibt letzterer einen *Request-Response* Prozess ausgehend vom zentralen Server. Neben einem kleinskaligen Versuchsaufbau stellt das Paper Simulationsergebnisse der beschriebenen Szenarios mit 10500 **Nodes** vor. Als Kommunikationsprotokoll kommt ein **ALOHA**-ähnliches [1] Protokoll zum Einsatz — mit dem Ergebnis, dass für hohes Datenaufkommen ein alternatives Kanalzugriffverfahren nötig sei [3].

---

<sup>1</sup> In [5] wird *TinyOS* als Betriebssystem benannt. Dieses verwendet laut Dokumentation das genannte Verfahren innerhalb des Software Stacks [26].

**PHYNETLAB** Dieses Testbed ist für die Erprobung von modernen vernetzten Logistikprozessen in einem Warenlager konzipiert. Die energieeffiziente Kommunikation der Frachtcontainer ist hierbei eine wesentliche Komponente. Neben der strukturellen Beschreibung dieses speziellen Szenarios eines Sensornetzes erfolgt eine detailliertere Beschreibung der Hard- und Softwarekomponenten. Das System besteht aus drei Schichten, so genannten *Layern*.

1. *Application-Layer*: Zur Datenerhebung, Firmwareverwaltung und Verbindung zu anderen Diensten.
2. *Edge-Layer*: *Access Points (APs)* bilden die Schnittstelle vom Sensornetzwerk zum Internet.
3. *Device-Layer*: Hier sind die intelligenten Frachtcontainer angesiedelt.

Während die Dienste des *Application-Layers* auf Internet-*Server* ausgelagert werden können, bilden die *APs* des *Edge-Layers* und die Frachtcontainer als Teil des *Device-Layers* die Komponenten des Sensornetzwerks innerhalb des Logistiklagers: Ein *AP* ist zum einen über eine mobile Internetverbindung mit Geräten des *Application-Layers* verbunden, zum anderen kommuniziert er über zwei separate Netzwerke mit den Frachtcontainern. Eines davon ist ein *ZigBee* Netzwerk, welches als Hintergrundnetzwerk fungiert. Hierüber können statistische Daten gesammelt oder Einstellungen an den Frachtcontainern für bestimmte Experimente vorgenommen werden. Darüber hinaus lässt sich über dieses Hintergrundnetzwerk die Firmware der eigentlichen Sensorhardware des Frachtcontainers aktualisieren [25].

Beispielhaft wird die Leistungsfähigkeit des *Listen Before Talk* Algorithmus als etabliertes Kanalzugriffverfahren mit dem Testbed untersucht. Dazu sendet ein *AP* eine Anfrage nach einem bestimmten Produkt und die entsprechenden Frachtcontainer antworten. So kommt es zu besonders vielen nahezu gleichzeitigen Zugriffen auf den Funkkanal. Dabei kommen ab einer Anzahl von 17 gleichzeitig antwortenden Frachtcontainern bereits weniger als die Hälfte der Antworten am *AP* an. Diese Rate sei für diesen Anwendungsfall noch hinreichend, katastrophal sei dagegen der drastisch steigende Energiebedarf der einzelnen Frachtcontainer [25]. Bei zwei beteiligten Sendern steigt der durchschnittliche Energiebedarf um mehr als das Doppelte im Vergleich zu einem Sender. Sind 38 Sender beteiligt, steigt der Energiebedarf um den Faktor 18. Der effiziente Kanalzugriff stelle daher noch Optimierungspotential dar. Das Wissen über die Leistungsfähigkeit des Netzwerks sei auch grundlegend für die Festsetzung der Anzahl an *APs* im gesamten Lager

Die vorliegende Arbeit greift die dargelegten Probleme in den genannten Arbeiten auf: Sie untersucht den Kanalzugriff in der Funk-

kommunikation für den Anwendungsfall in einem Logistiklager unter Berücksichtigung der Energieeffizienz.



# 4

## AUSWAHL DES KANALZUGRIFFSVERFAHRENS

Nach dem zunächst kurz das ALOHA Protokoll als klassisches Kanalzugriffverfahren beschrieben wird, stellt dieses Kapitel zwei etablierte Verfahren im Detail vor.

### 4.1 ALOHA

Als traditionelles Beispiel des Kanalzugriffes sei hier das *ALOHA*-Protokoll aufgeführt. Kern des Verfahrens ist ein zufälliger Kanalzugriff der Netzwerkteilnehmer. Dabei wird davon ausgegangen, dass die Zwischenankunftszeit der gesendeten Pakete negativ exponentiell verteilt ist. Wird eine Kollision erkannt, wiederholt sich der Sendevorgang. In [Abbildung 4.1](#) ist dieser Fall bei dem ersten bzw. zweiten Sendevorschuss der Geräte A und B zu sehen. Die Geräte schließen auf eine Kollision, da nach einer bestimmten Zeit eine Bestätigung des Empfängers ausbleibt. Die Übertragung wird dann jeweils nach einer zufälligen Zeit wiederholt. Der maximale Durchsatz liegt dabei bei ca. 18 Prozent [1].

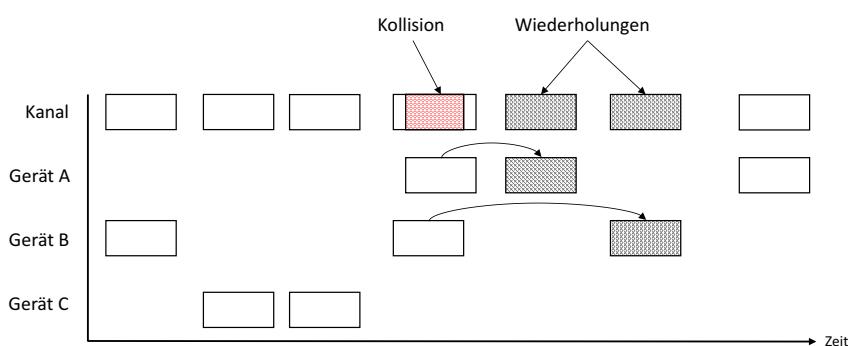


Abbildung 4.1: Beispiel für den Kanalzugriff nach dem ALOHA Protokoll

Die wiederholten Sendevorgänge wirken sich ungünstig auf den Energieverbrauch der **Nodes** aus. Mit steigender Anzahl der Teilnehmer, wie es in Logistanwendungen der Fall ist, stößt dieses System an seine Grenzen [3].

#### 4.2 LISTEN BEFORE TALK

Das **ETSI** befasst sich in [27] mit der elektromagnetischen Verträglichkeit und der Nutzung des Funkspektrums durch **SRDs**. Um die Ausnutzung des Funkkanals als begrenzte physikalische Ressource zu maximieren, beschreibt dieser Standard einen **LBT** Kanalzugriff.

Ist ein Gerät im Begriff, eine Übertragung zu starten, soll es zunächst den Funkkanal nach Aktivitäten anderer Geräte abhören und die eigentliche Übertragung noch zurückhalten. Die Abhörzeit  $t_L$  setzt sich aus einer festen und einer zufälligen Komponente zusammen:

$$t_L = t_F + t_{PS} \quad (4.1)$$

$t_F$  beträgt 5ms, während  $t_{PS}$  einen zufälligen Wert zwischen 0ms und 5ms annimmt. Wird innerhalb von  $t_F$  keine Aktivität erkannt, beginnt der Sendevorgang direkt im Anschluss und  $t_{PS}$  wird zu 0 gesetzt. Wohingegen im Fall eines belegten Kanals die Abhörzeit unterbrochen wird bis der Kanal wieder frei ist. Im Anschluss wird der Kanal für die gesamte Dauer von  $t_L$  abgehört, die durch den Anteil von  $t_{PS}$  variieren kann. Abbildung 4.2 verdeutlicht diese Verfahren exemplarisch für drei Geräte. Gerät C detektiert keine Aktivität auf dem Kanal innerhalb der 5ms Abhörzeit und startet daher unverzüglich mit der Übertragung. Gerät B und Gerät A, die zu diesem Zeitpunkt jeweils eine Übertragung ausstehen haben, können diese erst nach zwei bzw. drei Abhörzyklen beginnen.

In Abbildung 4.2 wird auch deutlich, wann sich die Geräte im Sende-, bzw. Empfangsmodus befinden. Um den Kanal abzuhören, muss sich ein Gerät im Empfangsmodus befinden. Der **LBT** Algorithmus sieht vor, dass im Fall eines belegten Kanals solange gewartet wird, bis dieser wieder frei ist. Auch in dieser Zeit muss ein Gerät im Empfangsstatus verbleiben, um den Zeitpunkt festzustellen, wann der Kanal wieder frei ist und der Backoff-Mechanismus erneut gestartet werden kann. Im Vergleich zu Gerät C ergibt sich für Gerät A eine erheblich längere Zeitspanne im Empfangsmodus.

#### 4.3 CARRIER SENSE MULTIPLE ACCESS / COLLISION AVOIDANCE

Das **IEEE** standardisiert in [7] die Bitübertragung (*Physical Layer*) und den Kanalzugriff (**MAC-Layer**) als Teil der Verbindungssicherung<sup>1</sup> für Sensornetzwerke mit niedriger Datenrate. Das Protokoll sieht vor, dass vor Beginn einer Übertragung ein **Carrier Sense Multiple Access / Collision Avoidance** Algorithmus durchlaufen wird. Ähnlich dem **LBT** sieht dieser vor, eine Übertragung zunächst nach einem bestimmten Muster zurückzuhalten. Dieser Vorgang wird *Backoff* genannt. In der Grundversion muss ein Gerät zwei Variablen zur Durchführung des Algorithmus verwalten:

<sup>1</sup> im Bezug auf das **Open Systems Interconnection** Referenzmodell der ISO

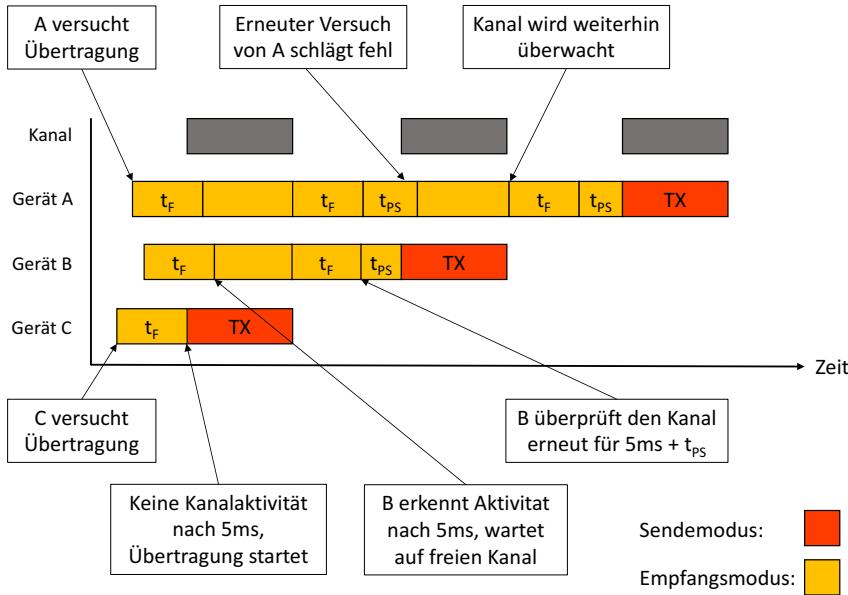


Abbildung 4.2: Beispiel für den LBT Algorithmus bei drei konkurrierenden Kanalzugriffen. Die orange markierten Zeiten repräsentieren Empfangsstatus, die roten Sendestatus.

1. NB: Anzahl der Durchläufe des Algorithmus für den aktuellen Sendeversuch
2. BE: Backoff-Exponent, der den Backoff-Zeitraum eingrenzt

Alle Zeitangaben, die in dem Standard beschrieben werden, beziehen sich auf die Übertragungsdauer eines Symbols, die *Symbol Period*. Dem **CSMA-CA** Verfahren liegt der Einheitszeitraum *Unit Backoff Period* zugrunde, der aus 20 *Symbol Periods* besteht [7]. Mit den in dieser Arbeit gewählten Einstellungen<sup>2</sup> ergibt sich:

$$\text{Unit Backoff Period} = 1\text{ms}$$

Die Backoff-Zeiten sind ganzzahlige Vielfache dieses Zeitraums und werden zufällig bestimmt. Die Menge der möglichen Werte ist:

$$\{0, (2^{BE}) - 1\} \text{ Unit Backoff Periods}$$

Die Variable BE begrenzt somit die Auswahl der möglichen Backoff-Zeiten. Ein Blockdiagramm des Algorithmus ist in [Anhang A](#) zu sehen. In [Abbildung 4.3](#) ist das Verfahren exemplarisch mit drei konkurrierenden Sendeversuchen dargestellt. Jedes Gerät hält seine Übertragung zunächst für eine zufällige Zeit, wie zuvor beschrieben, zurück. Im Anschluss wird die Aktivität auf dem Funkkanal überprüft ([Clear Channel Assessment \(CCA\)](#)). Die Prüfung des Kanals soll nach [7] 0,4ms

<sup>2</sup> Symbolrate =  $20 \frac{\text{kSymbols}}{\text{s}}$ , (Bitrate =  $20 \frac{\text{kBit}}{\text{s}}$ , BPSK Modulation)

dauern. In Abbildung 4.3 stellt Gerät B nach der Kanalprüfung keine Aktivität fest und beginnt mit dem Sendevorgang. Für dieses Gerät hat der Algorithmus an dieser Stelle erfolgreich terminiert. Die Kanalprüfung der Geräte A und C fällt negativ aus, sodass der Algorithmus die nächste Iteration durchläuft. Dabei werden nun die Variablen NB und BE jeweils um eins erhöht, wie Anhang A zu entnehmen ist. Dadurch steigt die Anzahl der möglichen Backoff-Zeiten mit jedem Durchlauf. Dies bedeutet nicht, dass die konkreten Backoff-Zeiten stetig größer werden, es stehen nur mehrere verschiedene Zeiträume zur Verfügung. Anzumerken ist, dass sich die Geräte während des Backoffs nicht im Empfangsstatus befinden müssen.

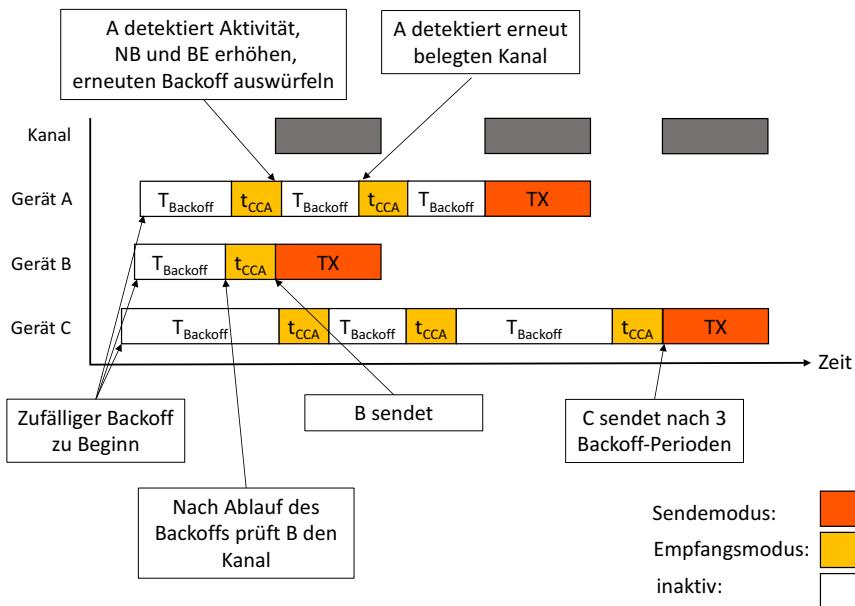


Abbildung 4.3: Beispiel für den CSMA-CA Algorithmus bei drei konkurrierenden Kanalzuriffen. Die orange markierten Zeiten repräsentieren Empfangsmodus, die roten Sendemodus. Während der weiß markierten Zeiten ist kein aktiver Modus notwendig.

#### 4.4 SCHLUSSFOLGERUNG FÜR DIE WEITERE UNTERSUCHUNG

Ein unkoordinierter Kanalzugriff nach dem ALOHA Ansatz führt bei einer großen Teilnehmeranzahl zu vielen Kollisionen. Übertragungen müssen dann wiederholt werden, was sich negativ auf den Energieverbrauch auswirkt. Dieser Ansatz ist daher für den Einsatz in einem großskaligen WSN mit stark eingeschränkter Energieverfügbarkeit der Nodes nicht geeignet [3].

Die eine genaue Untersuchung des LBT und des CSMA-CA Verfahrens sind dagegen in Erwägung zu ziehen [25][5]. Im Vergleich

von Abbildung 4.2 und Abbildung 4.3 wird jedoch deutlich, dass der CSMA-CA Algorithmus hinsichtlich der Energieeffizienz deutliche Vorteile aufweist. Der dramatische Anstieg des mittleren Energieverbrauchs pro Node bei Verwendung von LBT wird in [25] demonstriert. Daher wird im Folgenden das CSMA-CA Zugriffsverfahren nach dem IEEE 802.15.4 Standard näher untersucht.



# 5

---

## SYSTEMBESCHREIBUNG UND ANALYSEMETHODE

---

Dieses Kapitel gliedert sich in zwei Teile. Im ersten Teil werden zum Zweck der Vergleichbarkeit die Rahmenbedingungen der in [Kapitel 6](#) beschriebenen Simulation festgelegt, der zweite Teil beschäftigt sich mit der Analysemethode und definiert Messgrößen für die spätere Bewertung der Ergebnisse.

### 5.1 EIN GEEIGNETES SZENARIO

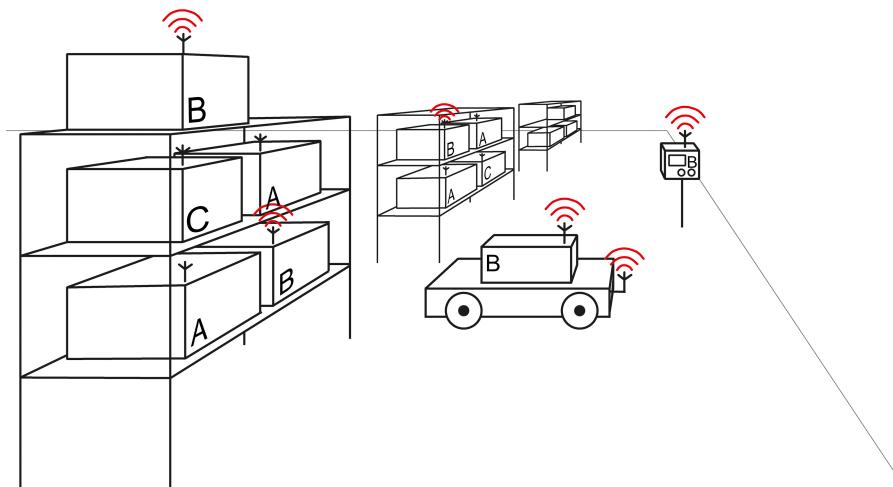


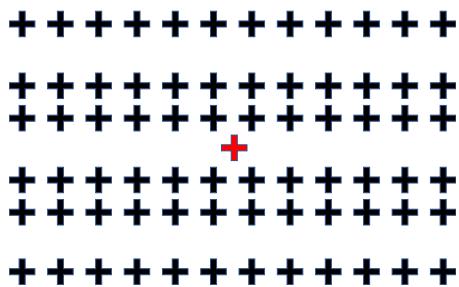
Abbildung 5.1: Beispiel für ein Logistikzenario mit vernetzten Frachtcontainern. Die Beschriftungen A, B, C stehen für drei verschiedene Produkte

Hauptbestandteil eines intelligenten Logistiklagers sind die Frachtcontainer mit den verschiedenen Gütern. Daneben gibt es Systeme zur Beförderung und Einlagerung der Frachtcontainer, dafür können auch mobile Roboter zu Einsatz kommen, wie in [Abbildung 5.1](#) angedeutet. Dienstprogramme, wie eine automatische Inventarisierung oder Kommissionierung, also das Zusammenstellen von bestimmten Produkten für einen Auftrag, gehören ebenfalls zu einem selbst organisierten Warenlager. Sowohl die Systeme zur Beförderung als auch die Computer, auf denen die Dienstprogramme ausgeführt werden, müssen in der Lage sein, mit den Frachtcontainern zu kommunizie-

ren. Sie werden damit Teil des Funknetzwerkes. Üblicherweise sind in einen Kommunikationsprozess die Container involviert, die das gleiche Produkt oder eine bestimmte Kategorie an Frachtgütern beinhalten. In [Abbildung 5.1](#) hat z. B. gerade ein Kommunikationsprozess bezogen auf das Produkt B begonnen. Charakteristisch ist, dass dabei eine Anfrage von einer Stelle ausgeht und für eine Vielzahl an Empfängern — den Frachtcontainern — bestimmt ist, die diese Anfrage beantworten. Von welcher Quelle genau die Anfrage stammt, ist für die Funktion des Kanalzugriffverfahren unerheblich, da sich immer o.g. Muster ergibt. Eine mobile Transporteinheit, die den kürzesten Weg zu einem Produkt finden möchte, oder eine Verfügbarkeitsanfrage über das Internet sind nur zwei Beispiele.

### 5.1.1 Anordnung

Als Grundlage für die spätere Simulation wird ein Anordnung angenommen, die der typischen Platzierung von Frachtcontainern in Logistiklagern entspricht. Dabei werden die Container in mehreren Zeilen von Hochregalen eingelagert, sodass sich die **Nodes** des zu untersuchenden [WSN](#) in einer dreidimensionalen Matrix befinden.



[Abbildung 5.2:](#) Die einzelnen Frachtcontainer werden in einem Logistiklager in Hochregalen zeilenweise eingelagert. Die Grafik zeigt die Aufsicht der Containerpositionen. An der rot markierten Position befindet sich der [AP](#)

Weiterhin wird in dieser Arbeit angenommen, dass es einen einzelnen Funkknoten gibt dessen Reichweite das gesamte Lager abdeckt und im weiteren Verlauf dieser Arbeit als [AP](#) bezeichnet wird. Eine Anordnung mit mehreren [APs](#) zur Reduzierung der Sendeleistungen ist sinnvoll [25], jedoch hier nicht Gegenstand der Untersuchung. Die Beobachtungen hinsichtlich der Netzlast in dem hier gewählten Ansatz lassen sich unter Berücksichtigung von Überlappungen auf ein Szenario mit mehreren [APs](#) übertragen. [Abbildung 5.2](#) zeigt diese Anordnung aus der Aufsicht.

### 5.1.2 Energieversorgung

Für die spätere Bewertung der Energiebilanz wird angenommen, dass der AP über eine unkritische Energieversorgung verfügt — z. B. direkt an das Stromnetz angeschlossen ist. Dagegen steht an den einzelnen Nodes nur eine sehr begrenzte Menge an Energie zu Verfügung, mit der effizient zu haushalten ist [2].

### 5.1.3 Datenaufkommen

Um die Paketgröße der ausgetauschten Nachrichten abzuschätzen, werden die folgenden Punkte berücksichtigt:

**IDENTIFIZIERUNG DES FRACHTCONTAINERS** Um einen Frachtcontainer zu identifizieren, ist eine eindeutige Adresse notwendig. Hier bietet sich die MAC Adresse an, die im IEEE 802.15.4 Standard verwendet wird [7]. Diese ist acht Byte lang [7, S. 151].

**IDENTIFIKATION DES FRACHTGUTES** Wie in [2] beschrieben, werden Güter in automatisierten Logistikprozessen mit der RFID-Technologie identifiziert. Hierbei wird der *Electronic Product Code (EPC)* des Produkts ausgelesen. Zur Identifizierung eines einzelnen Produkts dient die *globale Handelsgut Identifikationsnummer, engl. Global Trade Item Number (GTIN)*, die typischerweise in einem Strichcode codiert ist. Da die RFID-Technologie aber auch das berührungslose Auslesen von größeren Gebinden erlaubt, enthält der EPC zusätzlich Informationen über die logistische Einheit, wie z. B. Verpackungseinheit oder Palette. Der EPC besteht in der gebräuchlichsten Form aus 12 Byte [28].

**NACHRICHTEN TYP** Die unterschiedlichen Nachrichten-Arten können in einem Byte codiert werden. Das erlaubt theoretisch bis zu 256 verschiedene Nachrichtenarten oder das Verwenden von einzelnen Bits innerhalb dieses Bytes für Steuerinformationen, z. B. Sequenznummern.

**PARAMETER** Dieser Teil der Nachricht bietet Platz für zusätzliche Informationen je nach Nachrichtenart und wird pauschal mit zwei Byte veranschlagt.

Die genannten Punkte bilden die wesentlichen Bestandteile für den Informationsaustausch innerhalb eines Logistklagers. Tabelle 5.1 gibt einen Überblick über die resultierende Paketlänge. Es wird hierbei davon ausgegangen, dass eine Nachricht in der Regel alle Felder, jedoch mindestens Nachrichten-Typ und ein weiteres Feld enthält. Daher wird in Abbildung 5.3 ein Nachrichtenrahmen mit fester Länge definiert.

FELD	LÄNGE IN BYTE
Node ID ( <a href="#">MAC</a> -Adresse)	8
Produkt ID ( <a href="#">EPC</a> )	12
Nachrichten-Typ	1
Parameter	2
Summe	23

Tabelle 5.1: Bestandteile einer exemplarischen Nachricht für die Kommunikation in Logistiklagern

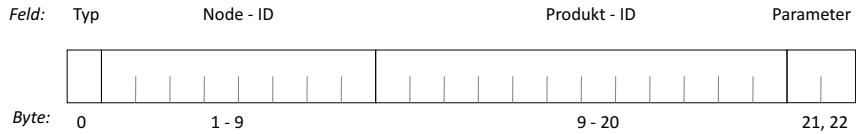


Abbildung 5.3: Nachrichtenrahmen mit fester Länge für den Informationsaustausch im Logistiklager

#### 5.1.4 Kommunikationsfluss

Die zwei Kernpunkte des Informationsaustausches für das zu untersuchende Szenario sind:

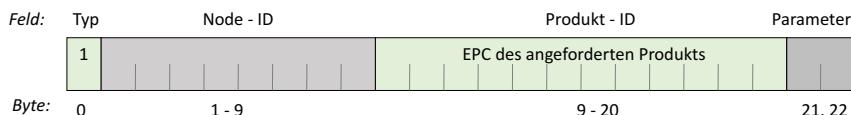
1. Anfragen nach einem bestimmten Produkt
2. Meldung der Frachtcontainer über deren Inhalt

Darüber hinaus sind weitere Nachrichten zur Organisation nötig, wie etwa:

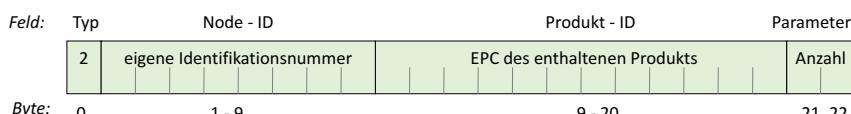
- A. Anmeldung neuer Frachtcontainer
- B. Zuweisung eines Produktes zu einem Frachtcontainer
- C. Meldung der Frachtcontainer über geringen Warenbestand
- D. Fehlermeldungen

Für die Untersuchung eines leistungsfähigen Kanalzugriffprotokolls spielen diese Nachrichten jedoch eine untergeordnete Rolle. Die Erprobung des Kanalzugriffprotokolls bezieht sich auf einen *Anfrage-Antwort* Zyklus, der vom [AP](#) initiiert wird. Dazu werden in [Abbildung 5.4](#) zwei Nachrichtentypen nach dem Schema aus [Unterabschnitt 5.1.3](#) definiert. Der erste Typ ist *PAGE* und dient der Anfrage nach einem Produkt, der zweite Typ heißt *QUANTITY* und ist für die Antworten der Frachtcontainer vorgesehen. Die *PAGE*-Nachricht wird als Broadcast in das Netz geschickt und löst damit nahezu gleichzeitige *QUANTITY*-Nachrichten der [Nodes](#) aus. Im Gegensatz zu [3]

wird von regelmäßigen Status-Nachrichten von Seiten der Frachtcontainer abgesehen. Der Ablauf eines Kommunikationszyklus ist in [Abbildung 5.6](#) als Sequenzdiagramm dargestellt.



(a) Typ 1 Nachricht: PAGE



(b) Typ 2 Nachricht: QUANTITY

Abbildung 5.4: Nachrichtentypen zur Abfrage von bestimmten Produkten. a) PAGE Nachricht mit angefordertem Produkt. b) QUANTITY Nachricht mit Anzahl der im Frachtcontainer verfügbaren Menge.

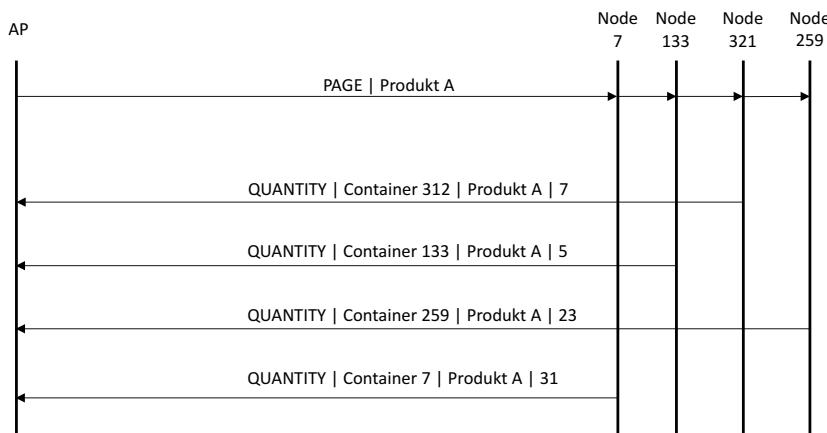


Abbildung 5.5: Sequenzdiagramm der Nachrichten zwischen AP und Nodes bei einer Produktanfrage

Wie in [Kapitel 3](#) aufgezeigt, sinkt die Leistungsfähigkeit der Systeme in [5] und [3] mit steigender Anzahl an parallelen Vorgängen innerhalb des Netzwerkes. Die Wahl des oben beschriebenen Kommunikationsflusses provoziert diese parallelen Vorgänge durch die hohe Zahl nahezu gleichzeitig versandter QUANTITY-Nachrichten und ist daher zur Untersuchung der Kanalzugriffverfahrens geeignet.

Anzumerken ist, dass eine Anfrage über den AP immer ein bestimmtes Produkt betrifft. Somit antwortet nur eine Teilmenge der sich insgesamt im Lager befindlichen Frachtcontainer. Die Ergebnisse dieser Arbeit können daher auch zu einer optimalen Verteilung einer Ware auf Frachtcontainer genutzt werden.

### 5.1.5 Kanalmodell

Im hier untersuchten Szenario wird als Kanalmodell die Freiraumausbreitung angenommen. Eine Analyse mit anderen Kanalmodellen ist zwar nicht Gegenstand dieser Arbeit, die Simulation erlaubt jedoch den Austausch des Kanalmodells für künftige Untersuchungen.

## 5.2 ANALYSE- UND MESSMETHODE

Ein zentrales Kriterium zur Beurteilung der Leistungsfähigkeit eines Kanalzugriffverfahrens in dem geschilderten Szenario ist die Anzahl an *QUANTITY*-Nachrichten, die nach einer *PAGE*-Nachricht beim **AP** ankommen. Weiterhin ist für die Bewertung der Energieeffizienz des Protokolls der Energieverbrauch an den einzelnen **Nodes** relevant. Weiterhin ist die Zeitspanne relevant, nach welcher der **AP** alle Nachrichten empfangen hat. Im Folgenden werden dazu Messgrößen für die Simulation definiert.

**PRODUCT SPECIFIC DELIVERY RATIO** Die *produkspezifische Zu-stellrate* engl. *Product Specific Delivery Ratio* (*PSDR*) bezieht die am **AP** erfolgreich empfangenen Antworten auf die Anzahl der Frachtcontainer mit gleichem Produkt:

$$r_{PS} = \frac{N_{\text{Antworten}}}{N_{\text{Container mit gleichem Produkt}}} \quad (5.1)$$

**MITTLERER ENERGIEVERBRAUCH** .. TODO .. Bespr. mit Robert, passive knoten ausgelassen, da Energieverbrauch unabhängig von Kanalzugriff .. sniffen sowieso nur minimalistisch auf dem kanal  
Kanalauslastung

### 5.2.1 Messaufbau

- a. *PowerScale* System zur Messung des Energieverbrauchs am **Node**
- b. *Code Composer Studio* Entwicklungsumgebung zur Ausführungssteuerung Prototypenprogramms
- c. *Packet Sniffer* Analysesoftware zum Auslesen des CC1200 Funkchips
- d. *Real Time Spectrum Analyzer* Messgerät zur kontinuierlichen Anzeige der momentanen Signalstärke im eingestellten Frequenzbereich
- e. *hterm* Terminalprogramm zum seriellen Datenaustausch mit dem **AP**

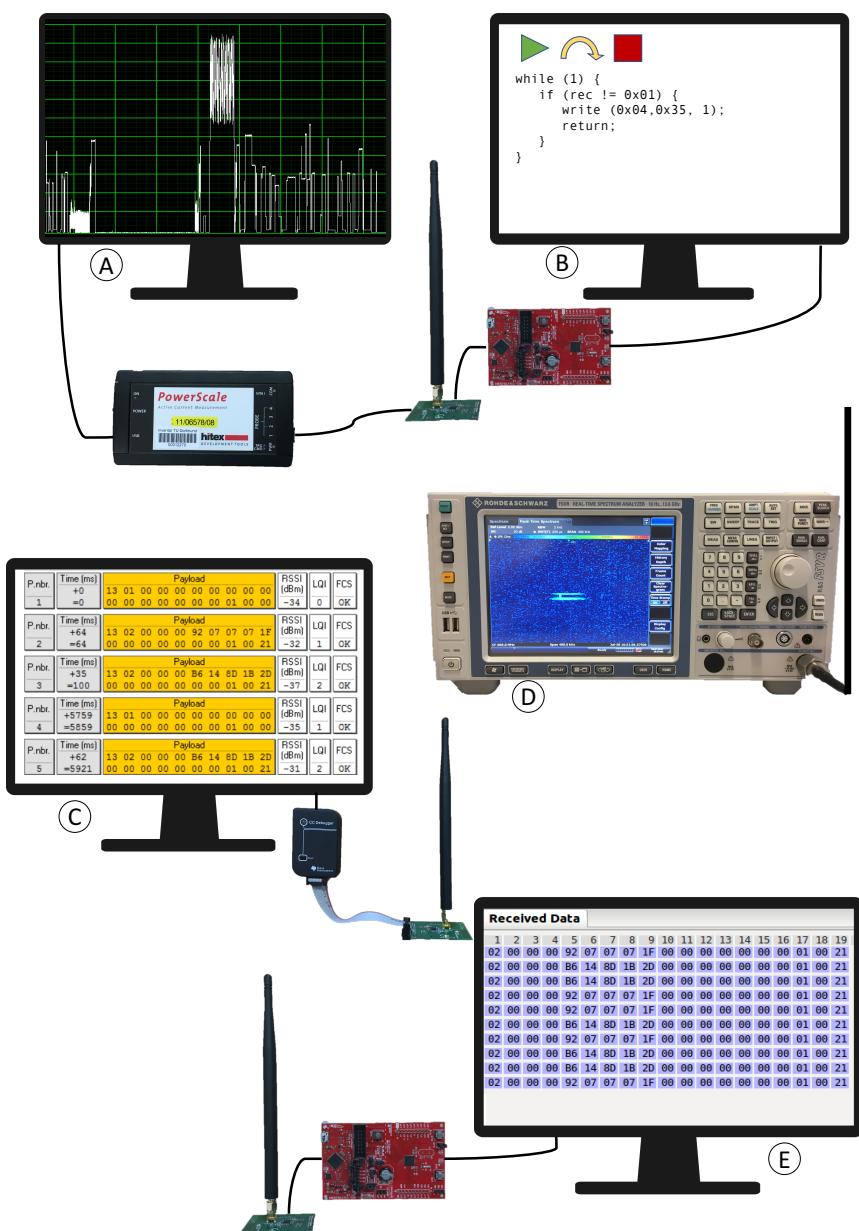


Abbildung 5.6: ToDo



# 6

---

## SIMULATION

---

### 6.1 IEEE 802.15.4 MODELL

Beschreibung, Aufbau

Besonders wichtig : Modellierung des Funkkanals

Beschreibung des Fehlers in der Datensammlung für Kollisionen

Bewertung des Modells für die Verwendung in dieser Arbeit

### 6.2 ERWEITERUNGEN DES MODELLS

Austausch/Neuerstellung der Anwendungsschicht

Ergänzung um automatische, zufällige Platzierung der Container

Besonders wichtig: Anpassung der Transceivermodi -> Energiemodell

-> Einbindung realer Messwerte

### 6.3 SIMULATIONSERGEBNISSE

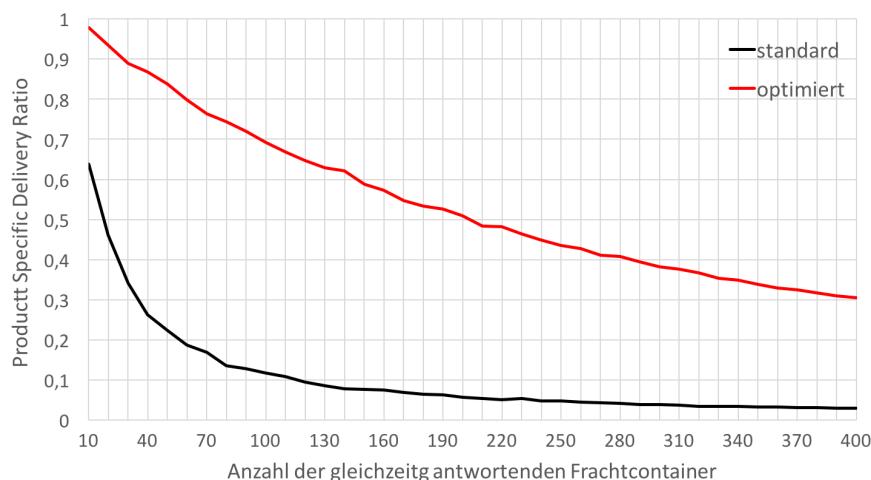


Abbildung 6.1: ToDo

Beschreibung -> näherungsweise neg.exp.  
 Konvergenz interessant evtl. noch ein Durchlauf bis 1000  
 Begründung des höheren Durchsatzes: Erleuterung der Auswahl von BE  
 evtl. Statistik mit einbeziehen

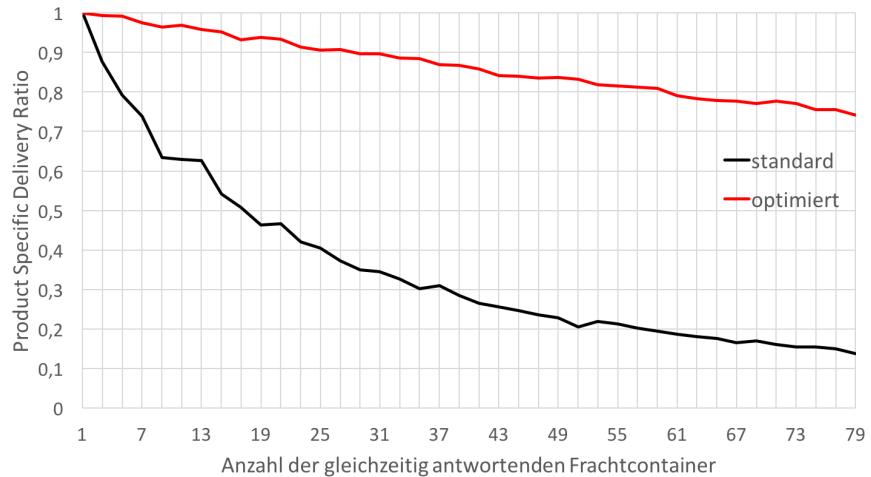


Abbildung 6.2: ToDo

besonders effizient im Bereich bis ca. 80 Sendern  
 bei standard bereist bei wenigen Sendern massive Kollisionen

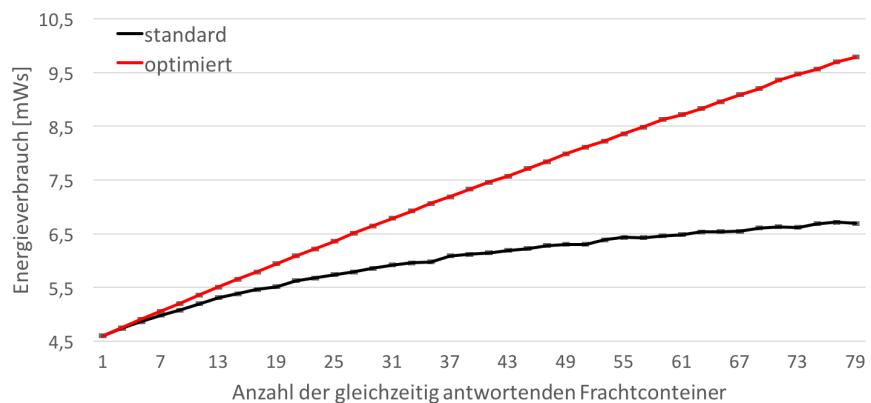


Abbildung 6.3: ToDo

Hinweis auf Energieverbrauch pro Sender während gesamter Laufzeit  
 Bezug von Energieverbrauch auf erfolgreich zugestellte Nachrichten  
 -> dazu evtl. noch ein Diagramm  
 Energieverbrauch pro Sendeversuch

Auswirkung der Algorithmus auf Antwortzeiten  
 Bezug und Bewertung dieses Effekts für das gegebene Szenario

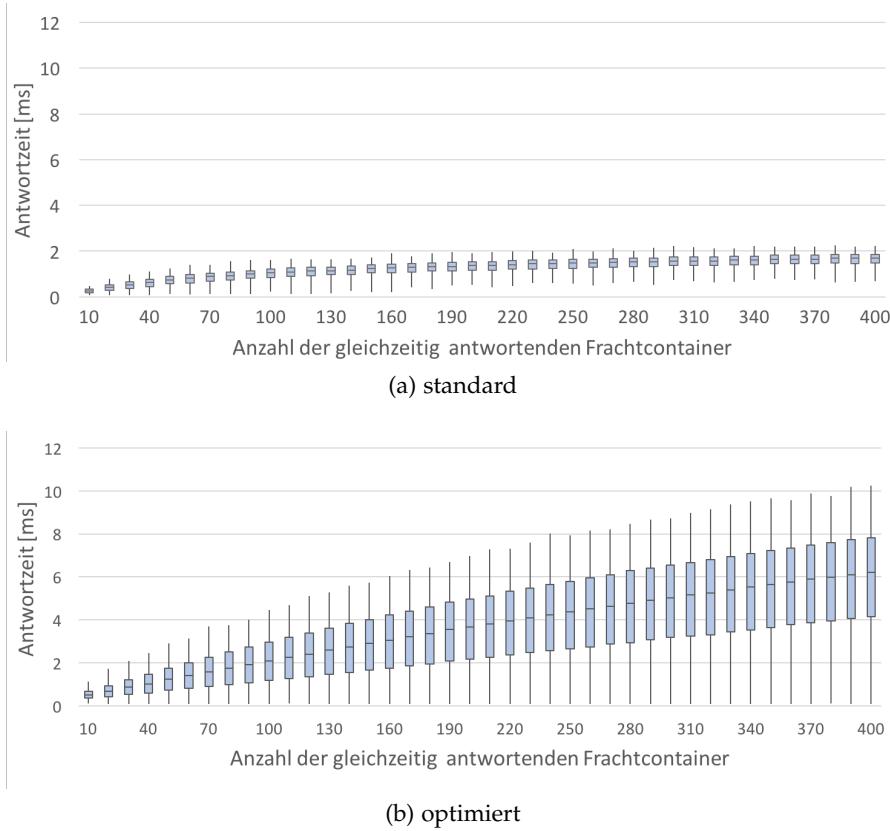


Abbildung 6.4: Boxplots der Antwortverzögerungen durch den [CSMA-CA](#) Algorithmus

Angabe von theoretischer oberer Schrake !?

Hinweis auf geringe Anzahl der Antworten insgesamt bei standard, wegen massiver Kollisionen Hinweis zur Verteilung der einzelnen Antworten beim optimierten Verfahren

-> immer auch Antworten direkt nach Anfrage (Ausreißer nahe 0) -> effektive Kanalnutzung  
dazu evtl noch Histogramm



# 7

---

## AUSWERTUNG

---



# 8

---

## PROTOTYPISCHE IMPLEMENTIERUNG

---

### Beschreibung der Architektur

#### 8.1 FUNKTIONSTEST

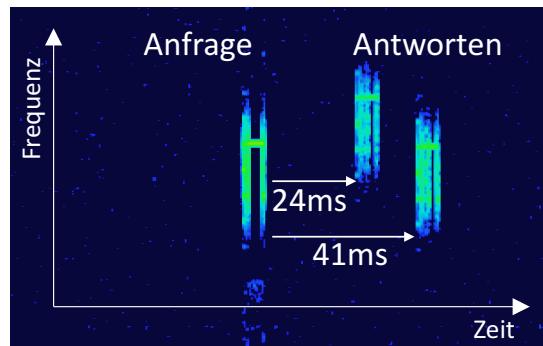
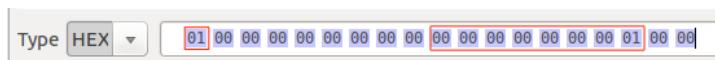


Abbildung 8.1: ToDo



(a) Terminal: Daten zur Übermittlung durch den AP

Received Data																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
02	00	00	00	B6	14	8D	1B	2D	00	00	00	00	00	00	00	01	00	21	00
00	00	00	00	92	07	07	07	1F	00	00	00	00	00	00	00	01	00	21	00

(b) Terminal: Empfangene Daten am AP

Abbildung 8.2: Ausschnitt aus dem Terminalprogramm *hterm*. a) Sendezelle mit Bytes einer PAGE-Nachricht b) Empfangsbereich mit zwei QUANTITY-Nachrichten

#### 8.2 MESSUNGEN

P.nbr.	Time [ms]	Payload		RSSI (dBm)	LQI	FCS
1	+0 =0	13 01 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00	00 00 00 00 00 00 00 00 00 00 01 00 21	-36	0	OK
2	+31 =31	13 02 00 00 00 B6 14 8D 1B 2D	00 00 00 00 00 00 00 00 00 01 00 21	-36	2	OK
3	+16 =48	13 02 00 00 00 92 07 07 07 1F	00 00 00 00 00 00 00 00 00 01 00 21	-33	2	OK

Abbildung 8.3: Ausgabe der mitgeschnittenen Kommunikation in der *Packet Sniffer* Software. Die Zusatzinformationen zu Signalstärke und Bitfehlern sind rot markiert.

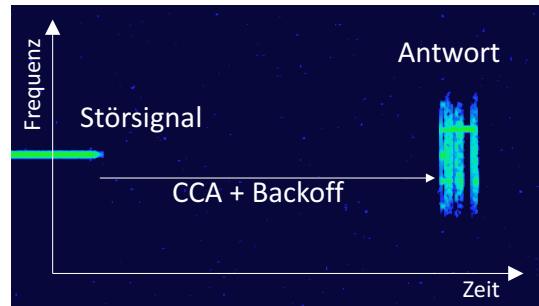
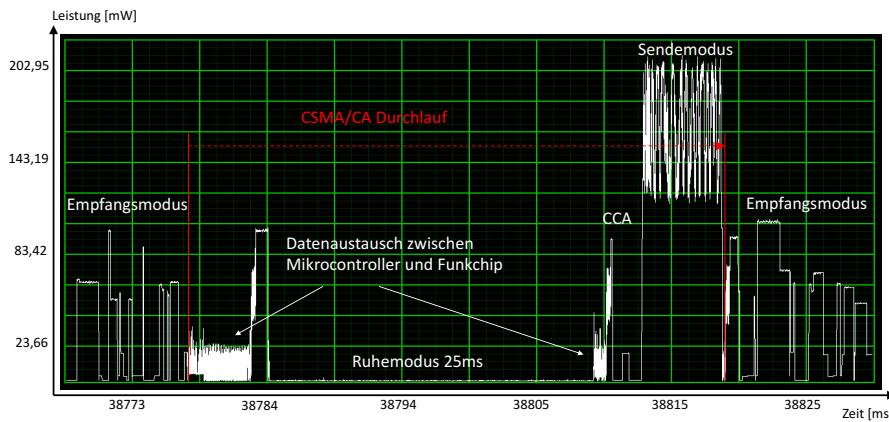
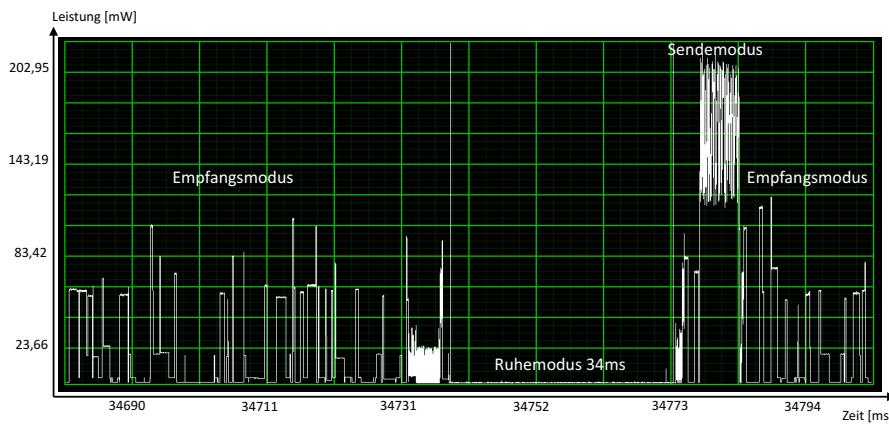


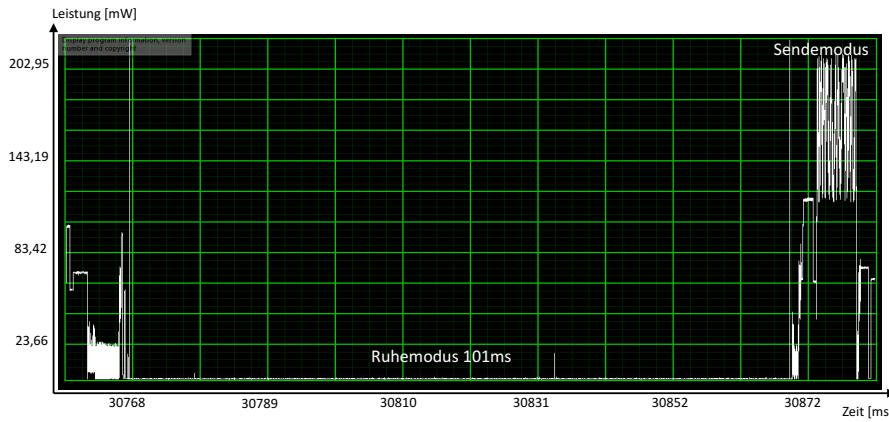
Abbildung 8.4: Zeitlicher Verlauf der Kanalaktivität mit schmalbandigem Störsignal. Nachdem der Kanal frei ist, erscheint das Paket.



(a) Detaillierte Darstellung der Leistungsaufnahme über die Zeit bei 25ms Backoff



(b) Leistungsaufnahme über die Zeit bei 34ms Backoff



(c) Leistungsaufnahme über die Zeit bei 101ms Backoff

Abbildung 8.5: Drei Messungen des *PowerScale* Systems zum Energieverbrauch des cc1200 Funkchips während des CSMA-CA Algorithmus

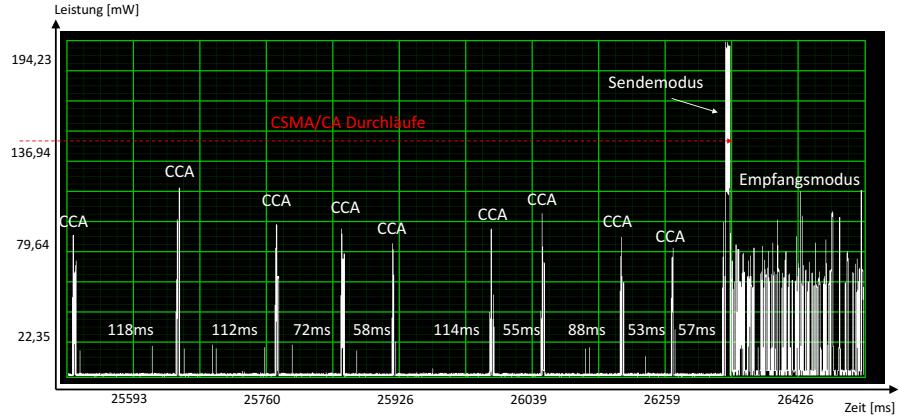


Abbildung 8.6: Analyse des Energieverbrauchs während des **CCA**. Da der Kanal durch ein Störsignal belegt ist, werden während des **CSMA-CA** Algorithmus wiederholt **CCAs** durchgeführt.

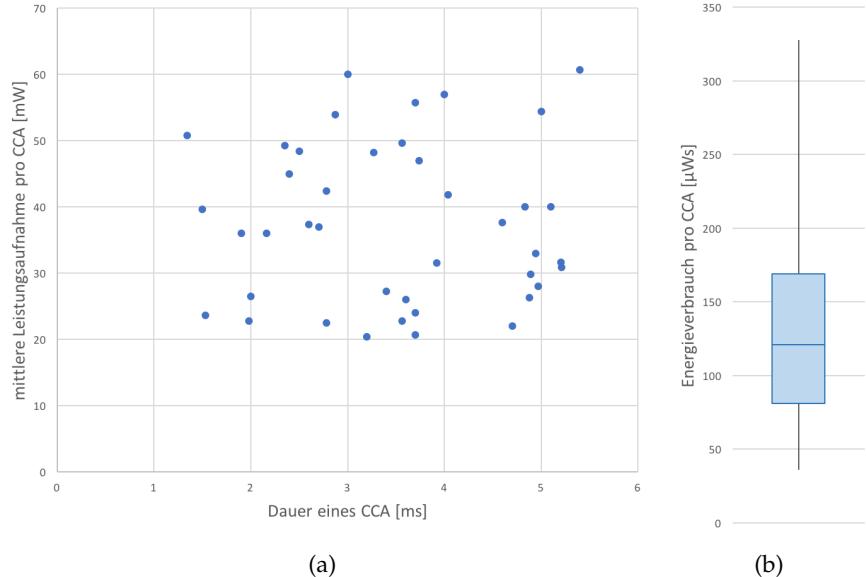


Abbildung 8.7: Analyse des Energieverbrauchs: a) Punktdiagramm von 40 **CCA** Messungen. Ein Messpaar besteht aus der Dauer und der mittleren Leistungsaufnahme b) Boxplot zum resultierenden Energieverbrauch von 40 **CCAs**.

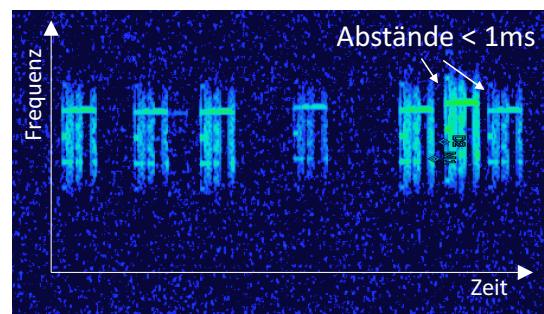


Abbildung 8.8: Am *Spectrum Analyzer* sind sieben Pakete zu sehen. Die Zwischenankunftszeit der letzten beiden Pakete liegt unter einer Millisekunde.



# 9

---

## ZUSAMMENFASSUNG

---



# A

---

## CSMA / CA BLOCKDIAGRAMM

---

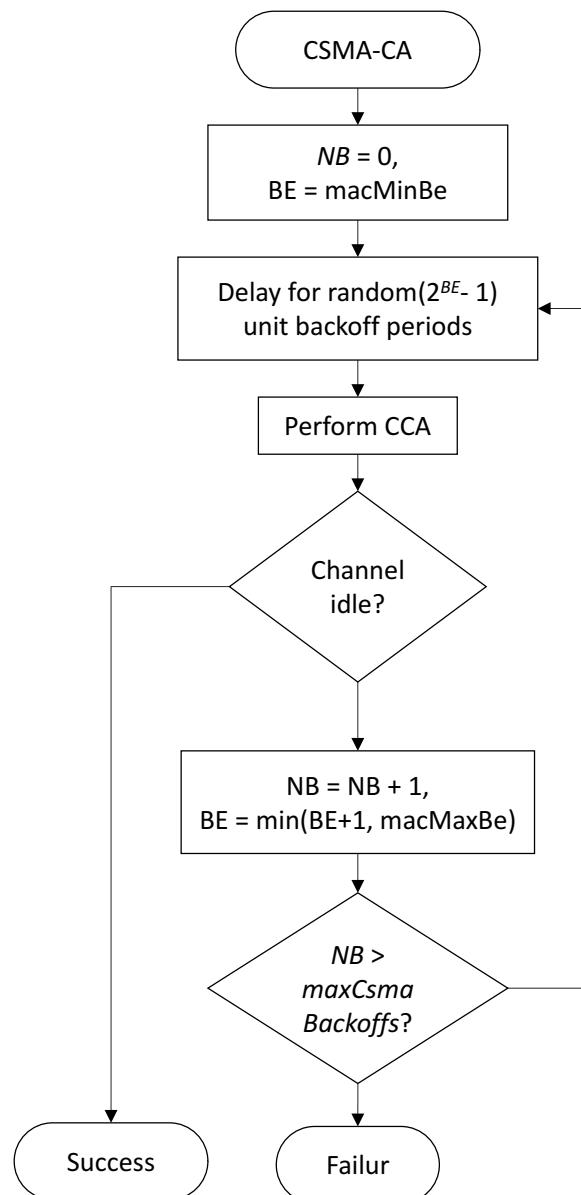


Abbildung A.1: Blockdiagramm des **CSMA-CA** Algorithmus nach dem Standard [7].



# B

---

## CC1200 REGISTERKONFIGURATION

---

```
/*-----  
| File: fr_reg_config.h  
|  
| Save register configuration for CC1200 in rf_settings_t array.  
| Defines register contents for CC1200 Register Space and  
| Extended Register Space  
|  
| Created with TI Smart RF Studio 7  
|  
| Note: Uses #defines from rf.h  
-----*/  
  
#ifndef RF_REG_CONFIG_H  
#define RF_REG_CONFIG_H  
  
#include "rf.h"  
  
static const rf_setting_t preferredSettings[]=  
{  
    {RF_IOCFG3,          0x06},  
    {RF_IOCFG2,          0x0F}, /* TXONCCA_DONE */  
    {RF_IOCFG1,          0x30},  
    {RF_IOCFG0,          0x06}, /* PKT_SYNC_RXTX */  
    {RF_SYNC3,           0x93},  
    {RF_SYNC2,           0x0B},  
    {RF_SYNC1,           0x51},  
    {RF_SYNC0,           0xDE},  
    {RF_SYNC_CFG1,       0xA9},  
    {RF_SYNC_CFG0,       0x03},  
    {RF_DEVIATION_M,     0x06},  
    {RF_MODCFG_DEV_E,   0x0B},  
    {RF_DCFILT_CFG,      0x4C},  
    {RF_PREAMBLE_CFG1,  0x14},  
    {RF_PREAMBLE_CFG0,  0x8A},  
    {RF_IQIC,             0xC8},  
    {RF_CHAN_BW,          0x10},  
    {RF_MDMCFG1,          0x42},  
    {RF_MDMCFG0,          0x05},  
    {RF_SYMBOL_RATE2,     0x8F},  
    {RF_SYMBOL_RATE1,     0x75},
```

```

{RF_SYMBOL_RATE0,           0x10},
{RF_AGC_REF,               0x27},
{RF_AGC_CS_THR,            0x11}, /* -82dBm, add 99 due tue RSSI
    offset -> 17 -> 0x11*/
{RF_AGC_GAIN_ADJUST,       0x00}, /* no RSSI offset stored, RSSI
    value will be 99dBm over reality, add -99 manually*/
{RF_AGC_CFG3,              0xB1},
{RF_AGC_CFG2,              0x20},
{RF_AGC_CFG1,              0x11},
{RF_AGC_CFG0,              0x9C}, /* 0x94 -> 0x9C RSSI_VALID_CNT
    = 9*/
{RF_FIFO_CFG,               0x00},
{RF_DEV_ADDR,               0x00},
{RF_SETTLING_CFG,           0x0B},
{RF_FS_CFG,                 0x12},
{RF_WOR_CFG1,               0x08},
{RF_WOR_CFG0,               0x21},
{RF_WOR_EVENT0_MSB,         0x00},
{RF_WOR_EVENT0_LSB,         0x00},
{RF_RXDCM_TIME,             0x00},
{RF_PKT_CFG2,               0x04}, /* CCA_MODE = 001 (RSSI below
    threshold)*/
{RF_PKT_CFG1,               0x03},
{RF_PKT_CFG0,               0x20},
{RF_RFEND_CFG1,              0x0F},
{RF_RFEND_CFG0,              0x00},
{RF_PA_CFG1,                 0x7F},
{RF_PA_CFG0,                 0x56},
{RF_ASK_CFG,                  0x0F},
{RF_PKT_LEN,                   0xFF},
{RF_IF_MIX_CFG,                0x1C},
{RF_FREQOFF_CFG,               0x20},
{RF_TOC_CFG,                  0x03},
{RF_MARC_SPARE,                0x00},
{RF_ECG_CFG,                  0x00},
{RF_MDMCFG2,                  0x02},
{RF_EXT_CTRL,                  0x01},
{RF_RCCAL_FINE,                 0x00},
{RF_RCCAL_COARSE,                0x00},
{RF_RCCAL_OFFSET,                0x00},
{RF_FREQOFF1,                  0x00},
{RF_FREQOFF0,                  0x00},
{RF_FREQQ2,                     0x56},
{RF_FREQQ1,                     0xCC},
{RF_FREQQ0,                     0xCC},
{RF_IF_ADC2,                     0x02},
{RF_IF_ADC1,                     0xEE},
{RF_IF_ADC0,                     0x10},
{RF_FS_DIG1,                     0x07},
{RF_FS_DIG0,                     0xAF},
{RF_FS_CAL3,                     0x00},
{RF_FS_CAL2,                     0x20},

```

{RF_FS_CAL1,	0x40},
{RF_FS_CAL0,	0x0E},
{RF_FS_CHP,	0x28},
{RF_FS_DIVTWO,	0x03},
{RF_FS_DSM1,	0x00},
{RF_FS_DSM0,	0x33},
{RF_FS_DVC1,	0xFF},
{RF_FS_DVC0,	0x17},
{RF_FS_LBI,	0x00},
{RF_FS_PFD,	0x00},
{RF_FS_PRE,	0x6E},
{RF_FS_REG_DIV_CML,	0x1C},
{RF_FS_SPARE,	0xAC},
{RF_FS_VC04,	0x14},
{RF_FS_VC03,	0x00},
{RF_FS_VC02,	0x00},
{RF_FS_VC01,	0x00},
{RF_FS_VC00,	0xB5},
{RF_GBIAS6,	0x00},
{RF_GBIAS5,	0x02},
{RF_GBIAS4,	0x00},
{RF_GBIAS3,	0x00},
{RF_GBIAS2,	0x10},
{RF_GBIAS1,	0x00},
{RF_GBIAS0,	0x00},
{RF_IFAMP,	0x09},
{RF_LNA,	0x01},
{RF_RXMIX,	0x01},
{RF_XOSC5,	0x0E},
{RF_XOSC4,	0xA0},
{RF_XOSC3,	0x03},
{RF_XOSC2,	0x04},
{RF_XOSC1,	0x03},
{RF_XOSC0,	0x00},
{RF_ANALOG_SPARE,	0x00},
{RF_PA_CFG3,	0x00},
{RF_WOR_TIME1,	0x00},
{RF_WOR_TIME0,	0x00},
{RF_WOR_CAPTURE1,	0x00},
{RF_WOR_CAPTURE0,	0x00},
{RF_BIST,	0x00},
{RF_DCFILTOFFSET_I1,	0x00},
{RF_DCFILTOFFSET_I0,	0x00},
{RF_DCFILTOFFSET_Q1,	0x00},
{RF_DCFILTOFFSET_Q0,	0x00},
{RF_IQIE_I1,	0x00},
{RF_IQIE_I0,	0x00},
{RF_IQIE_Q1,	0x00},
{RF_IQIE_Q0,	0x00},
{RF_RSSI1,	0x80},
{RF_RSSI0,	0x00},
{RF_MARCSTATE,	0x41},

```

{RF_LQI_VAL,           0x00},
{RF_PQT_SYNC_ERR,     0xFF},
{RF_DEM_STATUS,        0x00},
{RF_FREQOFF_EST1,     0x00},
{RF_FREQOFF_EST0,     0x00},
{RF_AGC_GAIN3,         0x00},
{RF_AGC_GAIN2,         0xD1},
{RF_AGC_GAIN1,         0x00},
{RF_AGC_GAIN0,         0x3F},
{RF_CFM_RX_DATA_OUT,   0x00},
{RF_CFM_TX_DATA_IN,   0x00},
{RF_ASK_SOFT_RX_DATA, 0x30},
{RF_RNDGEN,            0xFF}, // enable Rnd Num Gen NO
    RETENTION when going to sleep !!!
{RF_MAGN2,             0x00},
{RF_MAGN1,             0x00},
{RF_MAGN0,             0x00},
{RF_ANG1,              0x00},
{RF_ANG0,              0x00},
{RF_CHFILT_I2,          0x02},
{RF_CHFILT_I1,          0x00},
{RF_CHFILT_I0,          0x00},
{RF_CHFILT_Q2,          0x00},
{RF_CHFILT_Q1,          0x00},
{RF_CHFILT_Q0,          0x00},
{RF_GPIO_STATUS,         0x00},
{RF_FSCAL_CTRL,         0x01},
{RF_PHASE_ADJUST,       0x00},
{RF_PARTNUMBER,          0x20},
{RF_PARTVERSION,         0x11},
{RF_SERIAL_STATUS,       0x00},
{RF_MODEM_STATUS1,       0x10},
{RF_MODEM_STATUS0,       0x00},
{RF_MARC_STATUS1,        0x00},
{RF_MARC_STATUS0,        0x00},
{RF_PA_IFAMP_TEST,       0x00},
{RF_FSRF_TEST,           0x00},
{RF_PRE_TEST,            0x00},
{RF_PRE_OVR,             0x00},
{RF_ADC_TEST,             0x00},
{RF_DVC_TEST,             0x0B},
{RF_ATEST,               0x40},
{RF_ATEST_LVDS,          0x00},
{RF_ATEST_MODE,          0x00},
{RF_XOSC_TEST1,           0x3C},
{RF_XOSC_TEST0,           0x00},
{RF_AES,                 0x00},
{RF_MDM_TEST,             0x00},
{RF_RXFIRST,              0x00},
{RF_TXFIRST,              0x00},
{RF_RXLAST,               0x00},
{RF_TXLAST,               0x00},

```

```

{RF_NUM_TXBYTES,           0x00},
{RF_NUM_RXBYTES,           0x00},
{RF_FIFO_NUM_TXBYTES,      0x0F},
{RF_FIFO_NUM_RXBYTES,      0x00},
{RF_RXFIFO_PRE_BUF,       0x00},
{RF_AES_KEY15,             0x00},
{RF_AES_KEY14,             0x00},
{RF_AES_KEY13,             0x00},
{RF_AES_KEY12,             0x00},
{RF_AES_KEY11,             0x00},
{RF_AES_KEY10,             0x00},
{RF_AES_KEY9,              0x00},
{RF_AES_KEY8,              0x00},
{RF_AES_KEY7,              0x00},
{RF_AES_KEY6,              0x00},
{RF_AES_KEY5,              0x00},
{RF_AES_KEY4,              0x00},
{RF_AES_KEY3,              0x00},
{RF_AES_KEY2,              0x00},
{RF_AES_KEY1,              0x00},
{RF_AES_KEY0,              0x00},
{RF_AES_BUFFER15,          0x00},
{RF_AES_BUFFER14,          0x00},
{RF_AES_BUFFER13,          0x00},
{RF_AES_BUFFER12,          0x00},
{RF_AES_BUFFER11,          0x00},
{RF_AES_BUFFER10,          0x00},
{RF_AES_BUFFER9,           0x00},
{RF_AES_BUFFER8,           0x00},
{RF_AES_BUFFER7,           0x00},
{RF_AES_BUFFER6,           0x00},
{RF_AES_BUFFER5,           0x00},
{RF_AES_BUFFER4,           0x00},
{RF_AES_BUFFER3,           0x00},
{RF_AES_BUFFER2,           0x00},
{RF_AES_BUFFER1,           0x00},
{RF_AES_BUFFER0,           0x00},
};

#endif

```

Quelltext B.1: Struktur mit Adresse-Wert Paaren für die Registerkonfiguration des CC1200 Funkchips



# C

---

## CSMA / CA IMPLEMENTIERUNG

---

```
//////////  
///! PUBLIC rf_send()  
///!  
//////////  
void rf_send_fix(com_frame_t* frame) {  
  
    uint8  status = 0;  
    uint8  cca_state;  
    uint16 tx_on_cca_failed;  
    uint8  writeByte;  
    uint8  rnd;  
    uint16 backoff;  
  
    // Configure GPIO Interrupt  
    // no ISR no INT enable just set the right edge select  
    // for checking the ISR flag on P3.4 later for end of  
    // transmission.  
    // Line connected to P3.4 looks like this:  
    //  
    //      start sending           send complete  
    //  
    //      -----|-----|-----|-----  
    //  
    P3DIR &= ~BIT4;                      // Set P3.4 to input  
    direction  
    P3REN |= BIT4;                       // Set P3.4 pullup/down  
    Resistor  
    P3OUT &= ~BIT4;                      // Select P3.4 pull-down  
    P3IE  &= ~BIT4;                      // Disable Interrupt on P3.4  
    P3IES |= BIT4;                       // falling edge  
    P3IFG &= ~BIT4;                      // clear P3.4 interrupt flag  
  
    // copy data frame into txBuffer  
    // add length byte  
    txBuffer[0] = RF_PAYLOADLEN;  
    uint16 i;  
    for(i=0; i<RF_PAYLOADLEN; i++){  
        txBuffer[i+1] = frame->array[(RF_PAYLOADLEN)-1-i];  
    };  
}
```

```

    }

    // enter CSMA
    while(csma_state == BUSY){

        // choose random backoff
        status = spi_cmd_strobe(RF_SRX); // RX state for
            further randomized number
        status = read_reg(RF_RNDGEN, &rnd, 1);
        backoff = (uint16) (rnd & 0b0000000001111111); // use
            all 7 Bits -> 0 to 127
        status = read_reg(RF_RNDGEN, &rnd, 1);
        backoff += (uint16) (rnd & 0b0000000001111111); // use
            all 7 Bits again -> 0 to 255

        // put CC1200 into SLEEP during backoff
        status = spi_cmd_strobe(RF_SIDLE);
        status = spi_cmd_strobe(RF_SPWD); // until CS goes
            LOW again

        // backoff (blocking)
        TA3CCR0 = 0; // stop timer
        TA3CTL |= TACLR; // clear count value
        TA3CCTL0 &= ~CCIFG; // clear TACCR3
            interrupt flag
        TA3CCR0 = 125; // start timer SMCLK
            /8/125 = 1kHz => 1ms
        while(backoff != 0){ // wait for backoff
            counter
            while(!(TA3CCTL0 & CCIFG)); // wait for interrupt
                flag -> (ISR disabled)
            TA3CCTL0 &= ~CCIFG; // clear TACCR3
                interrupt flag
            backoff--;
        }

        //Wake Up
        status = spi_cmd_strobe(RF_SIDLE); // wake up CC1200
        status = spi_cmd_strobe(RF_SNOP); // debugging
        writeByte = 0xFF;
        status = write_reg(RF_RNDGEN, &writeByte, 1); //
            reactivate RNDGEN, no retention reg!

        // ensure RX mode and CARRIER_SENSE_VALID
        writeByte = 0x10; // 16->
            CARRIER_SENSE_VALID
        status = write_reg(RF_I0CFG2, &writeByte, 1);
        P3IFG &= ~BIT5; // clear P3.5
            interrupt flag
    }
}

```

```

status = spi_cmd_strobe(RF_SRX); // ensure RX to
    perform CCA
status = spi_cmd_strobe(RF_SNOP); // debugging
while(!(P3IFG & BIT5)); // wait for CS to be
    valid -> interrupt an P3.5 (ISR disabled)
P3IFG &= ~BIT5; // clear P3.5
    interrupt flag

// Write packet to TX FIFO
status = write_tx_fifo(txBuffer, sizeof(txBuffer));

// try to send
writeByte = 0x0F; // 15->TXONCCA_DONE
status = write_reg(RF_I0CFG2, &writeByte, 1);
P3IFG &= ~BIT5; // clear P3.5
    interrupt flag
status = spi_cmd_strobe(RF_STX); // try to send
while(!(P3IFG & BIT5)); // wait for CCA
    decision -> interrupt an P3.5 (ISR disabled)
P3IFG &= ~BIT5; // clear P3.5
    interrupt flag

// debugging
status = read_reg(RF_RSSI1, &rssi, 1);

//check CCA state
status = read_reg(RF_MARC_STATUS0, &cca_state, 1);
tx_on_cca_failed = (cca_state & 0b00000100);
if(!tx_on_cca_failed){ // NOT TXONCCA_FAILED
    csma_state = SUCCESS;
}
}
csma_state = BUSY;

// wait for interruptflag that packet has been sent.
// assuming the CC1200-GPIO connected to P3.4 is
// set to GPIOx_CFG = 0x06 -> CC1200 PKT_SYNC_RXTX
    interrupt
while(!(P3IFG & BIT4));
status = spi_cmd_strobe(RF_SNOP);

//flush TX FIFO
status = spi_cmd_strobe(RF_SIDLE);
status = spi_cmd_strobe(RF_SFTX);

status = spi_cmd_strobe(RF_SRX);
P3IFG &= ~BIT4; // clear P3.4 interrupt
    flag
P3IE |= BIT4; // Enable Interrupt on P3
    .4
P3IFG &= ~BIT4; // clear P3.4 interrupt
    flag

```

```

    }

//#####
// interrupt service routines:
//#####

//////////



//! PORT3.4 ISR indicating 'TX complete' as well as 'new data
// arrived'
//!
//////////


#pragma vector=PORT3_VECTOR
__interrupt void Port_3(void)
{
    P3IE &= ~BIT4;                                // Disable
                                                // Interrupt on P3.4

    uint8 status;
    status = read_rx_fifo(rxBuffer, sizeof(rxBuffer));

    if (rxBuffer[21] & 0b10000000) {                // check CRC
        g_callback((rxBuffer), SRC_RF);           // pointer
                                                // of secound byte, skip length byte
    }
    // flush RX-FIFO
    status = spi_cmd_strobe(RF_SIDLE);
    status = spi_cmd_strobe(RF_SFRX);

    status = spi_cmd_strobe(RF_SRX);

    P3IE |= BIT4;                                // Enable
                                                // Interrupt on P3.4
    P3IFG &= ~BIT4;                            // clear P3.4
                                                // interrupt flag
}

```

---

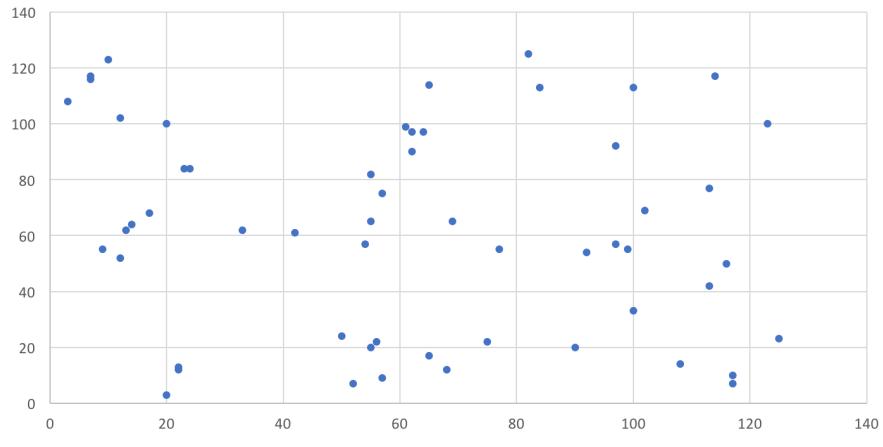
Quelltext C.1: Auszug aus dem Quellcode des Prototypen:  
Implementierung der Sendefunktion.

# D

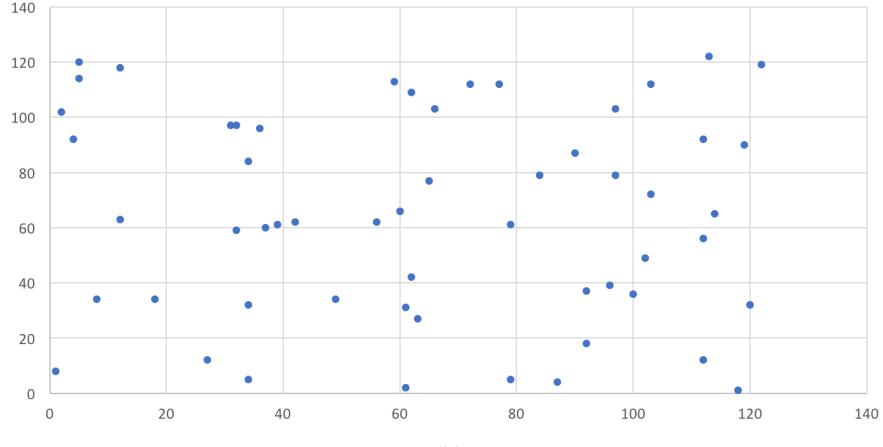
---

## ZUFALLSZAHLENERZEUGUNG

---



(a)



(b)

Abbildung D.1: Analyse der vom CC1200 Funkchip erzeugten Zufallszahlen. a) Vom Prozessor erzeugte Pseudozufallszahlen b) Pseudozufallszahlen kombiniert mit Empfängerrauschen.



---

## LITERATUR

---

- [1] N. Abramson. "The ALOHA system: another alternative for computer communication". In: *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*. 1970.
- [2] J. Emmerich, M. Roidl, T. Bich und M. ten Hompel. "Entwicklung von energieautarken, intelligenten Ladehilfen am Beispiel des inBin". In: *Logistics Journal* (2012).
- [3] M. Roidl, J. Emmerich, A. Riesner, M. Masoudinejad, D. Kaulbars, C. Ide, C. Wietfeld und M. T. Hompel. "Performance Availability Evaluation of Smart Devices in Materials Handling Systems". In: *2014 IEEE/CIC International Conference on Communications in China - Workshops (CIC/ICCC)*. 2014.
- [4] P. Nguyen, M. Schappacher, A. Sikora und V. F. Groza. "Extensions of the IEEE802.15.4 Protocol for Ultra-Low Energy Real-Time Communication". In: *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*. 2016.
- [5] Y. Liu, Y. He, M. Li, J. Wang, K. Liu und X. Li. "Does Wireless Sensor Network Scale? A Measurement Study on GreenOrbs". In: *IEEE Transactions on Parallel and Distributed Systems* (2013).
- [6] Bundesnetzagentur. *Funkanwendungen auf den ISM-Bändern*. 2015.  
URL: <http://emf3.bundesnetzagentur.de/pdf/ISM-BNetzA.pdf>.
- [7] IEEE Computer Society. *IEEE Standard for Low-Rate Wireless Networks*. 2015.
- [8] Texas Instruments. *CC1200Low-Power, High-Performance RF Transceiver*. SWRS123D. Rev. D. 2014.
- [9] Texas Instruments. *MSP430FR59xx Mixed-Signal Microcontrollers*. 2017.
- [10] Texas Instruments. *CC120X Low-Power High Performance Sub-1 GHz RF Transceivers Userâ€™s Guide*. SWRU346B. Rev. B. 2013.
- [11] F. Chen und F. Dressler. "A Simulation Model of IEEE 802.15.4 in OMNet++". In: *6. GI/ITG KuVS Fachgespräche Drahtlose Sensornetze, Poster Session*. 2007.
- [12] M. Kirsche und M. Schnurbusch. "A New IEEE 802.15.4 Simulation Model for OMNET++ / INET". In: *Proc. of 1st OMNeT++ Community Summit*. 2014.
- [13] L. M. Feeney. "Towards a better battery model for INET". In: *Proceedings of the OMNet++ Community Summit 2016*. 2016.

- [14] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton und T. Razafrinalambo. "A survey on facilities for experimental internet of things research". In: *IEEE Communications Magazine* 49.11 (2011), S. 58–67.
- [15] G. Werner-Allen, P. Swieskowski und M. Welsh. "Motelab: A Wireless Sensor Network Testbed". In: *IPSN*. 2005.
- [16] A. Arora, E. Ertin, R. Ramnath, M. Nesterenko und W. Leal. "Kansei: a high-fidelity sensing testbed". In: *IEEE Internet Computing* 10.2 (2006), S. 35–47.
- [17] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas und D. Pfisterer. "WISEBED: An Open Large-Scale Wireless Sensor Network Testbed". In: *Sensor Applications, Experimentation, and Logistics*. 2010.
- [18] Univ. of VA. *VineLab Wireless Testbed*. 2009. URL: <http://www.cs.virginia.edu/~whitehouse/research/testbed/>.
- [19] J. Bers, A. Gosain, I. Rose und M. Welsh. *CitySense: The Design and Performance of an Urban Wireless Sensor Network Testbed*. 2010.
- [20] T. Ojala. "Open Urban Testbed for Ubiquitous Computing". In: *2010 International Conference on Communications and Mobile Computing*. Bd. 1. 2010, S. 442–447.
- [21] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, Maria Kazandjieva, David Moss und Philip Levis. "CTP: An Efficient, Robust, and Reliable Collection Tree Protocol for Wireless Sensor Networks". In: *ACM Transactions on Sensor Networks (TOSN)*. 2013.
- [22] T. M. Chiwewe und G. P. Hancke. "A Distributed Topology Control Technique for Low Interference and Energy Efficiency in Wireless Sensor Networks". In: *IEEE Transactions on Industrial Informatics* 8.1 (2012), S. 11–19.
- [23] J. Beutel, R. Lim, A. Meier, L. Thiele, C. Walser, M. Woehrle und M. Yuecel. "The FlockLab testbed architecture". In: *Proceedings of the 7th International Conference on Embedded Networked Sensor Systems*. 2009.
- [24] C. Burin Des Rosiers, G. Chelius, E. Fleury, A. Fraboulet, A. Gallais und other. "SensLAB Very Large Scale Open Wireless Sensor Network Testbed". In: *Proc. 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*. 2011.
- [25] R. Falkenberg, M. Masoudinejad, M. Buschhoff, A. K. R. Venkatapathy, D. Friesel, M. ten Hompel, O. Spinczyk und C. Wietfeld. "PhyNetLab: An IoT-Based Warehouse Testbed". In: *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. Prague, Czech Republic, 2017. URL: <https://arxiv.org/abs/1706.09219>.

- [26] Stanford Information Networks Group. *TinyOS Documentation Wiki - CC2420 Layer Descriptions*. 2009. URL: [http://tinyos.stanford.edu/tinyos-wiki/index.php/CC2420\\_Layer\\_Descriptions](http://tinyos.stanford.edu/tinyos-wiki/index.php/CC2420_Layer_Descriptions).
- [27] ETSI European Standard. *ETSI EN 300 220-1: Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW; Part 1: Technical characteristics and test methods*. V2.4.1. 2012.
- [28] Global Standards One GS1. *EPC Tag Data Standard*. 2017. URL: [https://www.gs1.org/sites/default/files/docs/epc/GS1\\_EPC\\_TDS\\_i1\\_10.pdf](https://www.gs1.org/sites/default/files/docs/epc/GS1_EPC_TDS_i1_10.pdf).



# Eidesstattliche Versicherung

---

Drenhaus, Jens 151278  
Name, Vorname Matr.-Nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Masterarbeit mit dem Titel

*Simulation und Implementierung energieeffizienter Funkkommunikation für intelligente Logistiklager*

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

---

Dortmund, 30. Juli 2017  
Ort, Datum Unterschrift

**Belehrung:**

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00€ geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG - )  
Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

---

Dortmund, 30. Juli 2017  
Ort, Datum Unterschrift