

# Energieoptimierung funkbasierter Ortungseinheiten

Masterarbeit  
von

**Marius Wodtke**

Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB)  
der Fakultät für Informatik

Erstgutachter:  
Zweitgutachter:  
Betreuernder Mitarbeiter:

Prof. Dr. H. Schmeck  
Prof. Dr. ?. ??????????  
Dipl.-Inform. K. Bao

Bearbeitungszeit: 01. April 2017 – 31. Oktober 2017



---

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Karlsruhe, den 31. Oktober 2017

---



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Bisherige Situation . . . . .	1
1.2	Umgebung für funkbasierte Ortung . . . . .	2
1.3	Zielsetzung der Arbeit . . . . .	2
1.4	Anforderungen an das Bereichsortungssystem . . . . .	3
1.5	Gliederung der Arbeit . . . . .	3
<b>2</b>	<b>Grundlagen</b>	<b>5</b>
2.1	Klassifikation von Ortungssystemen . . . . .	5
2.1.1	Funkprotokoll-Standards . . . . .	5
2.1.2	Topologie . . . . .	5
2.1.2.1	Direkte Fernlokalisierung . . . . .	6
2.1.2.2	Direkte Selbstlokalisierung . . . . .	6
2.1.2.3	Indirekte Fernlokalisierung . . . . .	6
2.1.2.4	Indirekte Selbstlokalisierung . . . . .	6
2.1.2.5	Lokalisierung ohne Basisstationen . . . . .	6
2.1.2.6	Hybride Topologie . . . . .	6
2.1.3	Lokalisierungsprinzip . . . . .	7
2.1.4	Messgrößen . . . . .	8
2.1.4.1	Time of Arrival . . . . .	8
2.1.4.2	Time Difference of Arrival . . . . .	9
2.1.4.3	Roundtrip Time of Flight . . . . .	10
2.1.4.4	Received Signal Strength . . . . .	10
2.2	Grundlagen der IEEE 802.11 Spezifikation . . . . .	10
2.2.1	Scan . . . . .	11
2.2.2	Join . . . . .	11
2.2.3	Reassociation . . . . .	12
2.3	Grundlagen der Bluetooth Spezifikation . . . . .	13
2.3.1	Inquiry Scan . . . . .	13
2.3.2	Advertising . . . . .	14
2.4	Grundlagen von LoRaWAN . . . . .	14
<b>3</b>	<b>Analyse</b>	<b>17</b>
3.1	Verwandte Arbeiten . . . . .	17
3.1.1	Verwandte Arbeiten - Selbstlokalisierung & indirekte Fernlokalisierung mit IEEE 802.11 . . . . .	17
3.1.1.1	WiFi-LLS . . . . .	17
3.1.1.2	AiRISTA Flow RTLS . . . . .	19
3.1.1.3	AeroScout . . . . .	19

3.1.1.4	Selbstlokalisierung mit Szenenanalyse . . . . .	19
3.1.2	Verwandte Arbeiten - Fernlokalisierung & indirekte Selbstlokalisierung mit IEEE 802.11 . . . . .	21
3.1.2.1	RADAR . . . . .	21
3.1.2.2	Verbesserungen an RADAR . . . . .	21
3.1.2.3	Time-of-flight Lokalisierung . . . . .	22
3.1.2.4	Ortung ohne mobile Einheit . . . . .	22
3.1.3	Verwandte Arbeiten - Funkbasierte Lokalisierung mit weiteren Funkprotokollen . . . . .	23
3.1.3.1	Geeignete Messwerte für Bluetooth . . . . .	23
3.1.3.2	Antwortbasierte Inquiry Lokalisierung . . . . .	24
3.1.3.3	RSSI-basierte Inquiry Lokalisierung . . . . .	24
3.1.3.4	RSSI-basierte BLE Lokalisierung . . . . .	25
3.1.3.5	Lokalisierung mit LoRa . . . . .	26
3.2	Auswahl des Funkprotokolls . . . . .	26
3.3	Auswahl der Topologie . . . . .	27
3.4	Auswahl der Messgrößen . . . . .	27
3.5	Auswahl der Hardware . . . . .	27
3.6	Weiteres Vorgehen . . . . .	28
<b>4</b>	<b>Indirekte Fernlokalisierung mit IEEE 802.11</b>	<b>29</b>
4.1	ESP8266 . . . . .	29
4.1.1	ESP8266 Arduino Core . . . . .	31
4.1.2	ESP Open SDK . . . . .	31
4.2	Reichweite von 802.11 . . . . .	32
4.2.1	Methodik . . . . .	32
4.2.2	Ergebnisse . . . . .	33
4.2.3	Bewertung . . . . .	33
4.3	WiFi-LLS-Implementierung . . . . .	34
4.4	Anpassungen für Bereichsortung . . . . .	36
4.5	Untersuchung des Energieverbrauchs . . . . .	38
4.5.1	WiFi-LLS . . . . .	39
4.5.2	Indirekte Bereichsortung . . . . .	41
4.5.3	Kein AP in Reichweite . . . . .	42
<b>5</b>	<b>Direkte Fernlokalisierung mit 802.11</b>	<b>43</b>
5.1	RADAR-Implementierung . . . . .	43
5.2	Anpassungen für Bereichsortung . . . . .	44
5.2.1	Anzahl der verwendeten Kanäle . . . . .	46
5.3	Untersuchung des Energieverbrauchs . . . . .	47
5.3.1	RADAR . . . . .	47
5.3.2	Probe-Request-Lokalisierung . . . . .	48
5.4	Beschleunigungssensor . . . . .	49
5.4.1	LIS3DH . . . . .	49
5.4.2	Abschaltautomatik . . . . .	50
5.4.3	Bewertung . . . . .	51
<b>6</b>	<b>Direkte Fernlokalisierung mit Bluetooth</b>	<b>53</b>
6.1	nRF52832 . . . . .	53

6.1.1	Arduino Bluefruit nRF52 API . . . . .	53
6.2	Reichweite von Bluetooth . . . . .	54
6.2.1	Methodik . . . . .	54
6.2.2	Ergebnisse . . . . .	55
6.2.3	Bewertung . . . . .	55
6.3	BLE-Advertising-Implementierung . . . . .	56
6.4	Untersuchung des Energieverbrauchs . . . . .	56
6.4.1	Theoretische Energieverbrauchsabschätzung . . . . .	56
6.4.2	Tatsächlicher Energieverbrauch von BLE . . . . .	57
<b>7</b>	<b>Direkte Fernlokalisierung mit LoRa</b>	<b>59</b>
7.1	RFM95 . . . . .	59
7.1.1	RadioHead RFM9x für Arduino . . . . .	59
7.2	Reichweite von LoRa . . . . .	60
7.2.1	Methodik . . . . .	60
7.2.2	Ergebnisse . . . . .	61
7.2.3	Bewertung . . . . .	62
7.3	LoRa-Implementierung . . . . .	63
7.4	Untersuchung des Energieverbrauchs . . . . .	63
7.4.1	Theoretische Energieverbrauchsabschätzung . . . . .	63
7.4.2	Tatsächlicher Energieverbrauch von LoRa . . . . .	64
<b>8</b>	<b>Zusammenfassung und Fazit</b>	<b>67</b>
<b>Literaturverzeichnis</b>		<b>69</b>



# 1. Einleitung

Während die Ortung im Außenbereich fest in der Hand von Satellitensystemen wie dem Global Positioning System (GPS) liegen, bietet die Ortung im Innenraum eine Vielzahl verschiedener Technologien. Neben Technologien wie Bluetooth, Radio Frequency Identification (RFID) und Ultra Wide Band (UWB) weckt WLAN wegen seiner großen Verbreitung immer wieder Interesse in Forschung und Industrie.

So hat die Ortung mittels WLAN gerade im medizinischen Bereich durch kommerzielle Lösungen Verbreitung gefunden, Probleme finden sich aber bei Ortungsgenauigkeit gegenüber anderen Techniken und dem vergleichsweise hohen Energieverbrauch des WLAN Protokolls. Während sich viele wissenschaftliche Arbeiten der Ortungsgenauigkeit widmen, ist für den alltäglichen Einsatz die kurze Batterielaufzeit der mobilen Einheiten hinderlich, wenn nicht zum Beispiel Smartphones als mobile Einheiten in Frage kommen.

Auch im Tunnelbau ist eine Ortung von Mitarbeitern und Besuchern von Nöten um in Notfällen bestimmen zu können, ob und wie viele Personen sich im Gefahrenbereich befinden, dies beeinflusst die Arbeit der Rettungskräfte. Das veränderliche Umfeld der Baustelle, auf der große Stahl- und Betonelemente bewegt werden, stellt dabei die genaue Ortung mittels Radiowellen vor große Probleme und es wird nur eine Bereichsortung durchgeführt, bei der jede Tunnelröhre in mehrere hundert Meter große Abschnitte aufgeteilt wird und der Wechsel der Mitarbeiter zwischen den Abschnitten beobachtet wird. Dies stellt zwar nur eine geringe Genauigkeit dar, erlaubt es aber bei Bränden zu erkennen welche Personen sich durch die Abschnitte Richtung Ausgang bewegen und welche in ihrem Abschnitt verharren, sie sind vermutlich bewegungsunfähig oder eingeschlossen.

## 1.1 Bisherige Situation

Die Ortung wird derzeit bei der Ed. Züblin AG mittels Bluetooth durchgeführt, dabei sind die Basisstationen eigenständige Bluetooth-Einheiten die mit dem Ethernet Backbone verbunden sind. Als mobile Einheiten kommen sowohl batteriebetriebene Tags als auch Smartphones zum Einsatz. Das zentrale Sicherheitssystem fragt die gesehenen mobilen Einheiten bei den Basisstationen an und bereitet die Ergebnisse

graphisch auf.

Der Tunnel wird in Bereiche zu circa 500 Meter aufgeteilt, die Tunnelbohrmaschine (TBM) stellt dabei einen Sonderbereich dar, weil sie sich im Gegensatz zu den anderen Bereichen langsam bewegt. Neue Bereiche werden hinter der TBM eingefügt und sind dann stationär.

Die Bluetooth Basisstationen werden in Kästen verstaut die weitere notwendige Technik, wie etwa ein Notfalltelefon, enthalten. Weil diese Kästen nur etwa alle 500 Meter montiert sind, existieren große Lücken in denen die mobilen Einheiten nicht geortet werden. Eine mobile Einheit bleibt im System so lange in einem Bereich bis sie wieder von einer Basisstation erkannt wird.

Wird die mobile Einheit von der Erfassungseinheit vor dem Portal (Tunneleingang) erkannt gilt sie als außerhalb des Tunnels. Abbildung 1.1 zeigt die bisherige Situation mit Bluetooth Basissationen, die Reichweite der der Basistationen ist dabei nicht maßstabsgetreu.

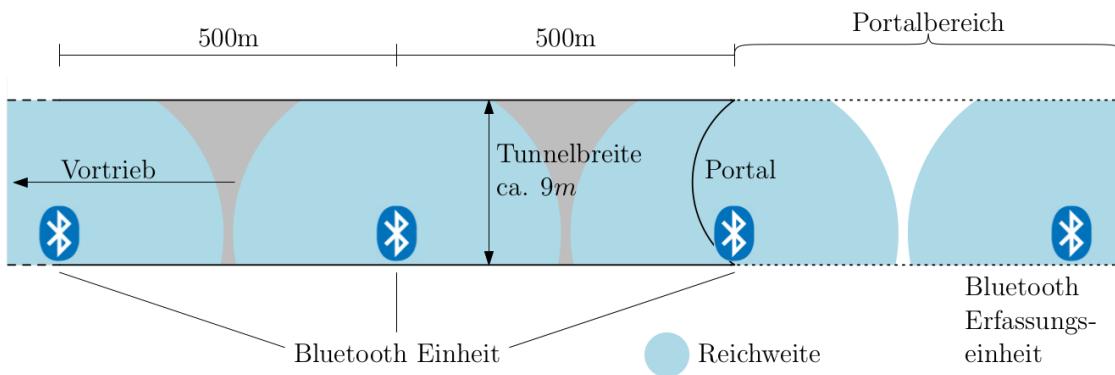


Abbildung 1.1: Bereichsortung mit Bluetooth, aus [Maur16].

## 1.2 Umgebung für funkbasierter Ortung

Als Versuchsumgebung dient die Tunnelbaustelle Rastatt. Dort gelten die Positionen der Kästen für die Technik als unveränderlich. Nur sie bieten Strom, Netzwerkanbindung (LAN) und Schutz vor dem Baustellenumfeld. Für Funkprotokolle, die weniger als 250 Meter Reichweite haben muss daher mit Erfassungslücken gerechnet werden. Auf der TBM und anderen technischen Fahrzeugen sind jedoch mehr Basisstationen möglich.

Es existiert bereits ein WLAN Netzwerk, dessen Access Points als Basisstationen genutzt werden können. Es handelt sich um APs der Firma Lancom, diese stellt auch einen LN-862 für Versuche. Für zukünftige Baustellen soll der Abstand der Versorgungskästen auf 250 Meter sinken, diese Situation ist in Abbildung 1.2 skizziert.

## 1.3 Zielsetzung der Arbeit

Ziel der Arbeit soll der Entwurf und die Implementierung eines Bereichsortungssystems für Personen in Tunnelanlagen sein. Bei einem Bereichsortungssystem handelt es sich um ein Ortungssystem, bei dem die Positionen nicht genau bestimmt werden. Stattdessen wird das Areal, auf dem geortet werden soll, in einzelne Bereiche

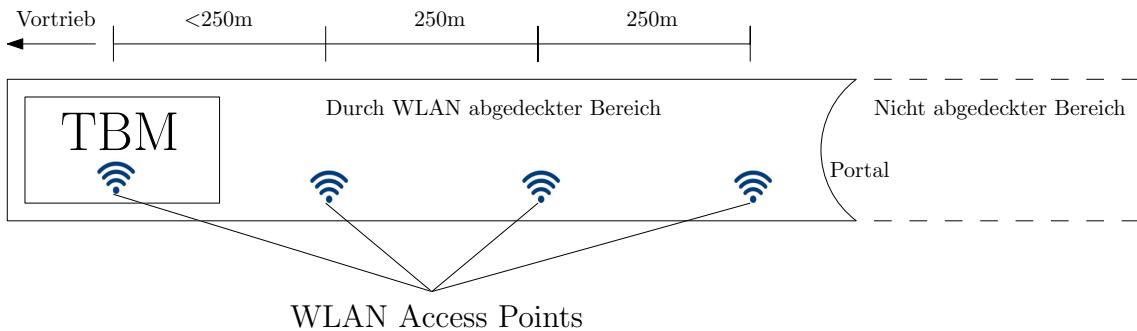


Abbildung 1.2: Zukiünftige Situation der Tunnelbaustellen.

unterteilt und jede mobile Einheit beim Vorgang des Ortens einem dieser Bereiche zugeordnet.

Diese Arbeit grenzt sich von vorherigen Arbeiten dadurch ab, dass die Laufzeit beziehungsweise der Energieverbrauch der mobilen Einheiten im Vordergrund steht. Ziel dieser Arbeit eine mehrmonatige Laufzeit der mobilen Einheiten.

Für diese Arbeit werden mehrere Prototypen für mobile Einheiten entwickelt und auf ihre Charakteristik bezüglich des Energieverbrauchs und der Erkennungssicherheit untersucht. Der Entwurfsraum umfasst dabei die Hardware und Software der mobilen Einheit sowie die Implementierung des Ortungsservers, die für die Funktechnologie benötigte Infrastruktur wird jeweils als gegeben angenommen.

## 1.4 Anforderungen an das Bereichsortungssystem

Da es sich um ein Bereichsortungssystem handeln soll, werden keine direkten Anforderungen an die Genauigkeit der Ortung gestellt. Jedoch soll ein klarer Wechsel zwischen zwei Bereichen, und damit zwei Basisstationen, zuverlässig erkannt werden. Die mobilen Einheiten sollen von Personen um den Hals getragen werden können. Dies bedingt ein geringes Gewicht des Akkus, gleichzeitig soll aber die Laufzeit der mobilen Einheit maximiert werden, sodass ein Akku mit möglichst großer Kapazität gewählt werden sollte. Zuletzt soll unter Rücksichtnahme auf das beschriebene Szenario die Komplexität der benötigten IT-Infrastruktur so gering wie möglich gehalten werden um ein stabiles und kostengünstiges System zu garantieren.

## 1.5 Gliederung der Arbeit

Nachdem zunächst in Kapitel 2 die Grundlagen der funkbasierten Ortung und der verwendeten Funkprotokolle behandelt werden, wird in Kapitel 2 das Problem analysiert und verwandte Arbeiten aufgezeigt. Die Kapitel 4, 5, 6 und 7 beschäftigen sich mit der Implementierung und Untersuchung der Prototypen. In Kapitel 5.4 wird die Integration eines Beschleunigungssensors zur Verlängerung der Laufzeit diskutiert. Das Fazit in Kapitel ?? fasst die Arbeit noch einmal zusammen, vergleicht und bewertet die Ergebnisse.



## 2. Grundlagen

### 2.1 Klassifikation von Ortungssystemen

Die Klassifikation von Ortungssystemen auf Funkbasis kann neben dem verwendeten Protokoll anhand der Topologie, dem Lokalisierungsprinzip und der gemessenen Größen durchgeführt werden [LDBL07]. Außerdem wird zwischen zweidimensionaler und dreidimensionaler Lokalisierung unterschieden.

#### 2.1.1 Funkprotokoll-Standards

Prinzipiell lassen sich verwendete Frequenzen, Modulationsverfahren und Sendeleistung frei wählen, solange die gesetzlichen Vorgaben für die Sender eingehalten werden. Wenn ein eigenes Protokoll verwendet wird, kann die Kodierung der Informationen frei gewählt werden. Ein Beispiel für ein solches Protokoll ist die Ortung mit Ultra-Breitband Signalen (UWB), mit dem Genauigkeiten im Zentimeterbereich erreicht werden können. Bestehende Protokolle zu verwenden hat dagegen den Vorteil, dass üblicherweise *commodity-of-the-shelf* Komponenten verwendet werden können, was die Kosten des Systems senkt. Im Gegenzug können für die Genauigkeit wichtige Parameter nicht mehr frei gewählt werden und der *Overhead* des Protokolls muss übernommen werden. Beispiele für solche Protokolle sind IEEE 802.11 (WLAN), Bluetooth und Radio Frequency Identification (RFID).

#### 2.1.2 Topologie

Die Topologie hat zwei Dimensionen, Fernlokalisierung versus Selbstlokalisierung und direkt versus indirekt und beschreibt wie Basisstationen und mobile Einheiten zusammenwirken. Alle besprochenen Topologien sind in Abbildung 2.1 veranschaulicht, gewellte Pfeile stellen hier zu messende Signale dar, gerade Pfeile sind gerichtete Datenverbindungen.

### 2.1.2.1 Direkte Fernlokalisierung

Bei der direkten Fernlokalisierung werden von mobilen Einheiten gesendete Signale an Basisstationen gemessen und die Ergebnisse an einen zentralen Ortungsserver weitergegeben, dieser führt dann die Lokalisierung durch. Das Ergebnis der Lokalisierung ist nur zentral verfügbar.

### 2.1.2.2 Direkte Selbstlokalisierung

Bei der direkten Selbstlokalisierung senden hingegen die Basisstationen und die mobilen Einheiten messen die eingehenden Signale. Anhand der Messergebnisse bestimmt jede mobile Einheit seine Position, die Ergebnisse der Lokalisierung sind anschließend nur auf der mobilen Einheit verfügbar.

### 2.1.2.3 Indirekte Fernlokalisierung

Die indirekte Fernlokalisierung gleicht der direkten Selbstlokalisierung, allerdings besteht zusätzlich eine Datenverbindung zu einem zentralen Ortungsserver, dem die berechnete Position mitgeteilt wird. Das Ergebnis steht nach kurzer Verzögerung sowohl auf der mobilen Einheit als auch zentral zur Verfügung, optional können dort die Daten von anderen mobilen Einheiten abgerufen werden um sich über deren Positionen zu informieren.

### 2.1.2.4 Indirekte Selbstlokalisierung

Auch die indirekte Selbstlokalisierung erweitert die direkte Fernlokalisierung um eine Datenverbindung zwischen mobiler Einheit und zentralem Ortungsserver. So kann sie sich über die eigene und eventuelle auch andere mobile Einheiten informieren. Dies kann insbesondere bei der Verwendung von Mischtechnologien notwendig sein, wenn die zur Ortung genutzte Technik es nicht erlaubt an den mobilen Einheiten zu empfangen, aber dennoch eine Selbstlokalisierung durchgeführt werden soll.

### 2.1.2.5 Lokalisierung ohne Basisstationen

Die mobilen Einheiten können sich auch gegenseitig lokalisieren, dazu messen sie die Distanz zu anderen mobilen Einheiten. Sie können diese Ergebnisse dann wiederum anderen mobilen Stationen mitteilen um so indirekt mit weiteren Referenzen die Lokalisierung zu präzisieren. Dadurch entsteht eine maßstabslose Karte, die die relative Position der mobilen Einheit zu den anderen mobilen Einheiten erfasst. Diese kann zusätzlich an einen zentralen Ortungsserver versendet werden um sie dort verfügbar zu machen.

### 2.1.2.6 Hybride Topologie

In einer hybriden Topologie orten sich die mobilen Einheiten gegenseitig, werden dabei aber von wenigen Basisstationen unterstützt. Diese Basisstationen geben ihre feste Position bekannt, dadurch ist es möglich der entstehenden Karte Punkte für die Ausrichtung und Skalierung hinzuzufügen. Außerdem können sie mit dem zentralen Ortungsserver kommunizieren um eine indirekte Fernlokalisierung mittels der durch die mobilen Einheiten propagierten Karten eine indirekte Fernlokalisierung durchzuführen.



Abbildung 2.1: Topologien für die Lokalisierung

### 2.1.3 Lokalisierungsprinzip

Das einfachste Lokalisierungsprinzip ist das Umgebungsprinzip, hier wird eine mobile Einheit der Basisstation mit dem stärksten Signal zugeordnet, die Basisstation kann sowohl als Sender als auch als Empfänger auftreten. Die gewonnene Position ist dabei nur symbolisch und ihre Genauigkeit ist abhängig von der Dichte des Netzes. Für eine erfolgreiche Lokalisierung muss nur eine Basisstation in Reichweite der mobilen Einheit sein, was diese Möglichkeit auch in komplexeren Lokalisierungsprinzipien zu einer sinnvollen Ausweichmöglichkeit macht, falls nicht genügend Basisstationen in Reichweite sind.

Die geometrische Bestimmung der Position einer mobilen Einheit mit drei oder mehr Basisstationen nennt man Trilateration. Dabei wird üblicherweise für jede Basisstation die Distanz zur mobilen Einheit bestimmt, anschließend wird ein Kreis mit der gemessenen Distanz um jede Basisstation gebildet und der Schnittpunkt der drei Kreise bestimmt die Position der mobilen Einheit. Da die Distanzen aus den gemessenen Signalparametern geschätzt werden müssen sind sie fehlerbehaftet und es ergibt sich oft kein eindeutiger Schnittpunkt, dann wird zum Beispiel die Mitte der Schnittpunkte gewählt.

Die geometrische Bestimmung ist ebenfalls über die Berechnung der Winkel zu den Basisstationen möglich, dies nennt man Triangulation. Der Winkel kann zum Beispiel über die zeitliche Differenz der ankommenden Signale an den synchronisierten Basisstationen bestimmt werden, dann wird von jeder Basisstation aus eine Halbgerade in diesem Winkel gefällt und der Schnittpunkt bestimmt die Position der mobilen Einheit. Wenn mit gerichteten Antennen gearbeitet wird, kann der Winkel des eingehenden Signals direkt bestimmt werden, dann müssen nur zwei Basisstationen in Reichweite sein um die mobile Einheit zu lokalisieren.

Das aufwendigste Lokalisierungsprinzip ist die Szenenanalyse. Hier werden zunächst in einer *Offline-Phase* Vektoren  $(m_1, m_2, \dots, m_n, p_x, p_y, p_z)^T$  aus Messgrößen und Positionsmarken gesammelt, anhand dieser Fingerabdrücke werden dann in der *Online-Phase* die Positionen neuer Messergebnisse bestimmt. Üblicherweise kommen dabei Verfahren des maschinellen Lernens wie *k-nearest-neighbour*, *neuronale Netze* oder *Support Vektor Machines* (SVM) zum Einsatz um die Muster der Fingerabdrücke aus der *Offline-Phase* zu erkennen und die Positionen  $(p_x, p_y, p_z)^T$  in der *Online-Phase* aus den gemessenen Größen  $(m_1, m_2, \dots, m_n)^T$  zu schätzen. Im zweidimensionalen Fall wird dabei  $p_z = 0$  angenommen.

## 2.1.4 Messgrößen

Beliebte Messgrößen für die Positionsbestimmung sind Ankunftszeit des Signals (*Time of Arrival*, TOA), die Differenz der Ankunftszeiten (*Time Difference of Arrival*, TDOA), die Paketumlaufzeit (*Roundtrip Time of Flight*, RTOF) und die Stärke des empfangenen Signals (*Received Signal Strength*, RSS). Es existieren aber noch mehr messbare Größen, wie etwa die Messung der empfangenen Phase des Signals (*Received Signal Phase*, RSP) und die Messung des Einfallswinkel des Signals mit mehreren gerichteten Antennen (*Angle of Arrival*, AOA).

### 2.1.4.1 Time of Arrival

TOA beruht auf der begrenzten Ausbreitungsgeschwindigkeit des Signals, hochfrequente Signale breiten sich mit Lichtgeschwindigkeit  $c = 299.792.458 \text{ m/s}$ , Ultraschall mit Schallgeschwindigkeit  $c \approx 343 \text{ m/s}$  bei  $20^\circ\text{C}$  aus, damit ergibt sich die Distanz  $d_i = c * (t_{\text{empfangen},i} - t_{\text{gesendet}})$  mit  $i$  sei die empfangende Basisstation. Die Position der mobilen Einheit kann dann mit der Methode der kleinsten Quadrate bestimmt werden, dazu wird die Kostenfunktion  $F(x, y, t_{\text{gesendet}}) = \sum_{i=1}^N \alpha_i^2 f_i^2(x, y, t_{\text{gesendet}})$  mit  $f_i(x, y, t_{\text{gesendet}}) = d_i - \sqrt{(x_i - x)^2 + (y_i - y)^2}$  und  $\alpha_i$  sei die Konfidenz von Basisstation  $i$ , minimiert. Dies erfordert, dass die Basisstationen zeitsynchronisiert sind und  $t_{\text{empfangen},i}$  möglichst ohne vorherige Verarbeitung bestimmt wird um Varianzen zu vermeiden. Zusätzlich ist es von Vorteil, wenn auch die mobile Einheit synchronisiert ist, da dann nur noch mit zwei Variablen optimiert

werden muss, da  $t_{gesendet}$  dann entsprechend der Synchronisation und dem Sendeintervall als gegeben angenommen werden kann. Die Forderung nach Synchronität ist jedoch für  $c = 299.792.458 \text{ m/s}$  sehr stark, da das Signal nur  $3.36 \text{ ns/m}$  benötigt und somit bereits kleine zeitliche Ungenauigkeiten große Fehler verursachen und erst durch zusätzliche Messungen oder Glättungstechniken hohe Genauigkeiten erreicht werden können. Skibniewski et al. konnten hohe Genauigkeit durch die Verwendung von Ultraschall erzielen, da sich durch die geringere Ausbreitungsgeschwindigkeit Imperfektionen in der Synchronisierung weniger stark auswirken [SkJa09]. Bei einer Selbstlokalisierung ist zu beachten, dass  $t_{gesendet}$  den Sendezeitpunkt der Basisstationen darstellt, sie müssen also entweder alle gleichzeitig auf verschiedenen Frequenzen oder in einem vordefinierten Muster nacheinander senden, die Zeitdifferenz des tatsächlichen Sendezeitpunkts zu  $t_{gesendet}$  muss dann vom Empfangszeitpunkt  $t_{empfangen,i}$  abgezogen werden.

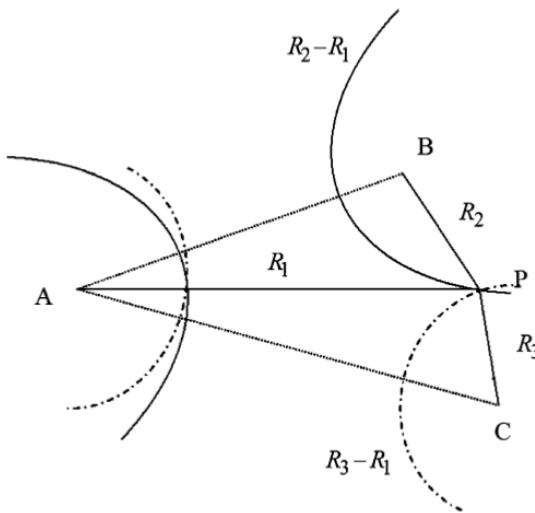


Abbildung 2.2: Positionsbestimmung mit der Differenz der Ankunftszeiten (TDOA), aus [LDBL07].

#### 2.1.4.2 Time Difference of Arrival

Bei TDOA wird statt der direkten Ankunftszeiten die Differenz der Ankunftszeiten gemessen, dies erfordert üblicherweise Zeitsynchronität bei den Basisstationen, nicht jedoch bei der mobilen Einheit, da nur die Ankunftszeiten für die Berechnung relevant sind. Die mobile Einheit liegt dabei auf dem Schnittpunkt zweier Hyperboloiden, die jeweils zwischen den Basisstationen, die TDOA gemessen haben, und einer Referenzstation aufgespannt werden. Die Hyperboloiden werden aufgestellt durch  $R_{i,j} = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2}$  mit  $(x_i, y_i, z_i)$  sei der jeweilige Messstation,  $(x_j, y_j, z_j)$  sei die gemeinsame Referenzstation und  $(x, y, z)$  sei die Position der mobile Einheit. Abbildung 2.2 zeigt die Lösung graphisch, Drane et. al. und Torrieri beschreiben die analytische Lösung der Gleichung mit nichtlinearer Regression [DrMS98] beziehungsweise Taylor-Entwicklung [Torr84]. Li et al. eliminiert zusätzlich die Forderung nach Synchronisation für mobile Einheiten, die sowohl senden als auch empfangen können [LPLaY00]. Dabei wird ein Datenpaket an die mobile Station gesendet und von dieser mit einem Acknowledgement beantwortet, diese Kommunikation wird von anderen Basisstationen

beobachtet und die Differenz der Ankunftszeiten des Datenpakets und des *Acknowledgements* als  $t_{i,1}$  mit  $i \in \text{Basisstationen}$  protokolliert. Zusätzlich ist  $t_{i,0}$  als die Differenz zwischen dem Senden des Datenpakets und dem Empfangen an den anderen Basisstationen durch die bekannte Position aller Basisstationen ebenfalls bekannt. Damit kann die Differenz der Ankunftszeit  $TDOA_{i,j} = (t_{i,0} + t_{i,1}) - (t_{j,0} + t_{j,1})$  mit  $i, j \in \text{Basisstationen}$  ohne vorherige Synchronisation berechnet werden.

#### 2.1.4.3 Roundtrip Time of Flight

Wird RTOF gemessen muss die mobile Einheit sowohl senden als auch empfangen können. Dabei wird die Zeit vom Aussenden eines Paketes bis zur Ankunft der Antwort gemessen, wichtig sind hier zum einen möglichst spät/früh gesetzten Zeitstempel beim Senden/Empfangen des Signals und eine sehr konstante Verarbeitungszeit des Kommunikationspartners. Zunächst muss dabei die Verarbeitungszeit gemessen werden, indem Basisstation und mobile Einheit auf einen Abstand von 0 Metern gebracht werden, anschließend kann die Entfernung  $d = c * (t_{\text{empfangen}} - t_{\text{gesendet}} - t_{\text{Verarbeitung}})$  bestimmt werden. Die Position wird dann analog zu TOA bestimmt. Diese Messgröße kann sowohl für die Fern- als auch für die Selbstlokalisierung verwendet werden, ist aber sehr anfällig gegenüber Schwankungen in der Verarbeitungszeit, die bei Protokollen wie IEEE IEEE 802.11 häufig auftreten.

#### 2.1.4.4 Received Signal Strength

RSS überprüft wie stark das empfangene Signal aus dem allgemeinen Rauschen auf der verwendeten Frequenz hervor sticht. Dafür ist es wichtig, dass immer mit der selben Leistung gesendet wird, dann kann mit einem theoretisch oder empirisch ermittelten Ausbreitungsmodell die Entfernung zwischen mobiler Einheit und Basisstation anhand des Pfadverlustes bestimmt werden. Dann ist  $d = F(S_{\text{gesendet}}, S_{\text{empfangen}})$  mit  $F$  sei das Ausbreitungsmodell. Um gute Ergebnisse zu erzielen sollten jeweils möglichst gleichförmige mobile Einheiten und Basisstationen verwendet werden, denn Lui et al. konnte für WLAN zeigen, dass unterschiedliche Hardware bei Basisstationen und mobiler Einheit erhebliche Varianz in den gemessenen RSS Werten erzeugt [LGLD<sup>11</sup>]. Zusätzliche Probleme entstehen, wenn Spezifikationen wie Bluetooth eine Anpassung der Sendeleistung vorsehen [HoSo07], dann muss eine solche Anpassung eliminiert werden. Bei Bluetooth geschieht das durch die Verwendung von *Inquiry Paketen*, diese dienen zur Entdeckung anderer Bluetooth-fähiger Geräte und werden immer mit voller Sendeleistung gesendet um möglichst alle Geräte in Reichweite zu erreichen. Ein weiteres Problem von RSS ist der starke Einfluss von Hindernissen und anderen Signalen auf dem selben Frequenzband. Gerade bei der Ortung in Innenräumen wirken diese Einflüsse begrenzend auf die Genauigkeit und oft muss eine Kalibrierung durchgeführt werden um die Einflüsse von Wänden und anderen Hindernissen zu eliminieren.

## 2.2 Grundlagen der IEEE 802.11 Spezifikation

Die Spezifikation IEEE 802.11 beschreibt eine Form der drahtlosen Datenübertragung mittels Funkwellen [IEEE12h]. Sie beschreibt die physische Schicht (PHY) und den Mediumszugriff (MAC) für ein Funknetzwerk, dass mit den darüber liegenden

Schichten des OSI-Modells in andere Netzwerke eingebunden werden kann. Wird ein solches Netzwerk in ein *Local Area Network* (LAN) eingebunden spricht man üblicherweise vom *Wireless Local Area Network* (WLAN). IEEE 802.11 wurde 1997 erstmals verabschiedet und wurde häufig erweitert und wird in ihrer ursprünglichen Form praktisch nicht mehr angewendet, da die Datenraten zu gering sind. Die Erweiterungen werden mit Buchstaben benannt (zum Beispiel IEEE 802.11g) und verändern etwa die verwendete Frequenz und Modulationsverfahren.

Zur Vermeidung von Kollisionen kommt *Carrier Sense Multiple Access/Collision Avoidance* (CSMA/CA) zum Einsatz. Bei CSMA/CA wird zunächst das Medium belauscht und gewartet bis keine Signale mehr auf dem Medium sind. Dann muss ein *Inter Frame Spacing* (IFS) abgewartet werden, je nach Priorität ist dieses unterschiedlich lang (*Short IFS < Priority IFS < Data IFS < Extended IFS*), anschließend kann gesendet werden. Tritt trotzdem eine Kollision auf, versucht der Sender es erneut mit einem längeren IFS.

IEEE 802.11 spezifiziert ebenfalls mehrere Operationen, die beispielsweise zur Entdeckung von Ressourcen und der Authentifizierung an einer Ressource dienen, einige, für diese Arbeit wichtige, Operationen werden im Folgenden beschrieben.

### 2.2.1 Scan

*Scan* ist eine Operation zur Entdeckung von Access Points, sie kann von einer Station (Endverbraucher, zum Beispiel Smartphone oder Laptop) passiv oder aktiv ausgeführt werden [IEEE12a]. Bei einem passiven *Scan* empfängt die *Station* und filtert die von Access Points regelmäßig gesendeten *Beacons* heraus, diese gelten dann als entdeckt. Ein *Beacon* ist ein *Management Frame*, der dazu gedacht ist, technische Möglichkeiten des APs zu bewerben, zum Beispiel die möglichen Datenraten und Zeitstempel zur Synchronisierung. Versteckte APs senden keine *Beacons*.

Bei einem aktiven *Scan* sendet die Station einen *Probe Request* aus, dieser kann sowohl an alle APs (*Broadcast*) als auch an einen speziellen AP adressiert sein. Ein *Probe Request* bewirbt, ähnlich wie ein *Beacon*, die technischen Möglichkeiten der Station. Der adressierte AP, beziehungsweise im Falle eines *Broadcasts* alle APs, beantwortet den *Probe Request* mit einer *Probe Response*, in der er mitteilt welche der beworbenen Funktionen er ebenfalls unterstützt. Die Station schließt den Vorgang mit einem *Acknowledgement* ab.

Das 2,4 GHz ISM-Band wird in Europa in 13 je 10 MHz breite Kanäle aufgeteilt, da ein AP immer nur auf einem Kanal aktiv ist, müsste jeder Kanal gescannt werden. Praktisch werden jedoch breitere Kanäle verwendet. IEEE 802.11g verwendet beispielsweise 20 MHz breite Kanäle, sodass effektiv nur die Kanäle 1, 5, 9 und 13 geprüft werden müssen.

Tabelle 2.1 listet alle in der IEEE 802.11 Spezifikation gelisteten *Management Frames*. Ein *Management Frame* wird durch die Typenbits markiert und durch die Bits für den Subtypen weiter unterschieden.

### 2.2.2 Join

Aus den entdeckten APs kann nun einer ausgewählt werden, um seinem Netzwerk (BSS) beizutreten [IEEE12c]. Es kann zwar geschehen, dass mehrere APs eines Netzwerks entdeckt wurden, eine Station kann jedoch zu jedem Zeitpunkt nur mit einem AP assoziiert sein.

Um einem Netzwerk beizutreten muss sich die Station zunächst authentifizieren.

Tabelle 2.1: *Management Frames* nach IEEE 802.11 [IEEE12g]

Type value b3 b2	Type description	Subtype value b7 b6 b5 b4	Subtype description
00	Management	0000	Association request
00	Management	0001	Association response
00	Management	0010	Reassociation request
00	Management	0011	Reassociation response
00	Management	0100	Probe request
00	Management	0101	Probe response
00	Management	0110	Timing Advertisement
00	Management	0111	Reserved
00	Management	1000	Beacon
00	Management	1001	ATIM
00	Management	1010	Disassociation
00	Management	1011	Authentication
00	Management	1100	Deauthentication
00	Management	1101	Action
00	Management	1110	Action No Ack
00	Management	1111	Reserved

Dieser Vorgang wird über einen *Authentication Frame* (siehe Tabelle 2.1) initiiert [IEEE12d]. Das weitere Vorgehen hängt vom Authentifizierungsverfahren ab, zum Beispiel kann der AP einen verschlüsselten *Challenge Text* an die Station senden, der mit einem aus dem Passwort erzeugten Schlüssel entschlüsselt wird und an den AP zurück gesendet werden kann. Ist die Antwort korrekt, bestätigt der AP den Vorgang mit einem *Acknowledgement* und eventuell zusätzlichen Informationen für eine Stromchiffre.

Anschließend kann die Station mit dem AP assoziiert werden [IEEE12e]. Sie erhält nun eine IP und der AP gibt dem Netzwerk bekannt, dass er für die Station zuständig ist. Dies geschieht üblicherweise über das *Address Resolution Protokoll* (ARP) oder das *Internet Group Management Protocol* (IGMP).

Ist eine Station assoziiert kann sie Datenverbindungen mit anderen Teilnehmern im Netzwerk aufbauen, sie könnte beispielweise das HTTP-Protokoll nutzen, um eine Webseite anzufordern.

### 2.2.3 Reassociation

Eine Reassoziation wird durchgeführt, wenn die Station keine gute Verbindung mehr zu ihrem AP hat und ein AP des selben Netzwerks verfügbar ist, der eine bessere Verbindung bietet [IEEE12f]. Um diesen neuen AP zu entdecken muss zunächst ein *Scan* durchgeführt werden.

Anschließend sendet die Station einen *Reassociation Request* an den neuen AP. Im *Reassociation Request* wird der alte AP benannt, sodass der neue AP überprüfen kann, ob die Station tatsächlich mit ihm assoziiert ist, gepufferte Pakete von ihm

entgegennehmen und die Assoziation mit ihm auflösen kann. Der Kommunikationsvorgang zwischen den APs wird auch als *Handoff* bezeichnet. Wird dieser erfolgreich abgeschlossen antwortet der neue AP der Station mit einer *Reassociation Response*. Abschließend wird dem Netzwerk die neue Assoziation mittels ARP mitgeteilt.

## 2.3 Grundlagen der Bluetooth Spezifikation

Bluetooth beschreibt eine Form der Funkkommunikation über kurze Distanzen, die zunächst für Mobiltelefone entwickelt und dann auf weitere Geräteklassen übertragen wurde. Ursprünglich wurde die physische Schicht (PHY) und der Mediumszugriff (MAC) in der Spezifikation 802.15.1 definiert [IEEE02]. Diese wird allerdings nicht mehr vom *Institute of Electrical and Electronics Engineers* (IEEE) gepflegt.

Bluetooth ist bereits in der Version 5.0 erschienen, da jedoch zum Zeitpunkt der Recherche für diese Arbeit kaum kompatible Geräte wird im Folgenden auf Bluetooth 4.0 Bezug genommen [Blue10a]. Bluetooth erlaubt es nach der Entdeckung eines Kommunikationspartners und anschließendem Aufbau einer Verbindung mit ihm im 2,4 GHz ISM-Band. Die Reichweite für die Verbindung hängt neben äußeren Hindernissen und den verwendeten Antennen auch von der jeweiligen Sendeleistung ab. Bluetooth Geräte werden dazu in drei Klassen eingeteilt, siehe Tabelle 2.2.

Tabelle 2.2: Klasseneinteilung für Bluetooth Geräte nach Sendeleistung, aus [Blue10c].

Power Class	Maximum Output Power (Pmax)	Nominal Output Power	Minimum Output Power <sup>1</sup>	Power Control
1	100 mW (20 dBm)	N/A	1 mW (0 dBm)	Pmin<+4 dBm to Pmax Optional: Pmin <sup>2</sup> to Pmax
2	2.5 mW (4 dBm)	1 mW (0 dBm)	0.25 mW (-6 dBm)	Optional: Pmin <sup>2</sup> to Pmax
3	1 mW (0 dBm)	N/A	N/A	Optional: Pmin <sup>2</sup> to Pmax

Mit Bluetooth 4.0 wurde erstmals *Bluetooth Low Energie* (BLE, auch Bluetooth Smart) vorgestellt. BLE wurde für niedrigen Energieverbrauch optimiert, es verzichtet auf den Verbindungsaufbau, um den Zeitabschnitt in dem das Gerät aktiv ist zu reduzieren. Da aber Informationen, die üblicherweise während des Verbindungsaufbaus übertragen werden, nun als Header um das Datenpaket gepackt werden müssen, sinkt die Bruttodatenrate gegenüber Bluetooth deutlich [Riga16]. BLE eignet sich somit vor allem dann, wenn wenige oder selten Informationen ausgetauscht werden müssen. Im Folgenden werden die, für die Lokalisation wichtigen Operationen vorgestellt. Es handelt sich dabei jeweils um die Funktion des Entdeckens von Ressourcen mittels *Inquiry Scan* für Bluetooth beziehungsweise mittels *Advertising* für BLE.

### 2.3.1 Inquiry Scan

Eine Bluetooth Verbindung besitzt immer einen *Master* und einen *Slave* [Blue10f]. Da es sich beim *Inquiry Scan* um eine Operation zum Entdecken von Ressourcen

handelt ist, sind die Rollen noch nicht festgelegt. Die Spezifikation geht aber davon aus, dass der Sender der *Inquiry Message* der *Master* ist. Nachdem der *Master* die *Inquiry Message* versendet hat antwortet der *Slave* nach  $625 \mu\text{s}$  mit einer *Inquiry Response*. Soll eine erweiterte *Inquiry Response* zum Einsatz kommen, wird dies in der *Inquiry Response* gekennzeichnet und die erweiterte *Inquiry Response*  $1250 \mu\text{s}$  später gesendet. Diese Zeiten sind den bei Bluetooth vorgesehenen Frequenzsprüngen angepasst. Abbildung 2.3 zeigt den Vorgang inklusive erweiterter *Inquiry Response*. Die Pakete sind entsprechend ihrer Inhaltsspezifikation benannt, *ID* entspricht der *Inquiry Message*, *FHS* der *Inquiry Response* und *Extended Inquiry Response Packet* der erweiterten *Inquiry Response*.

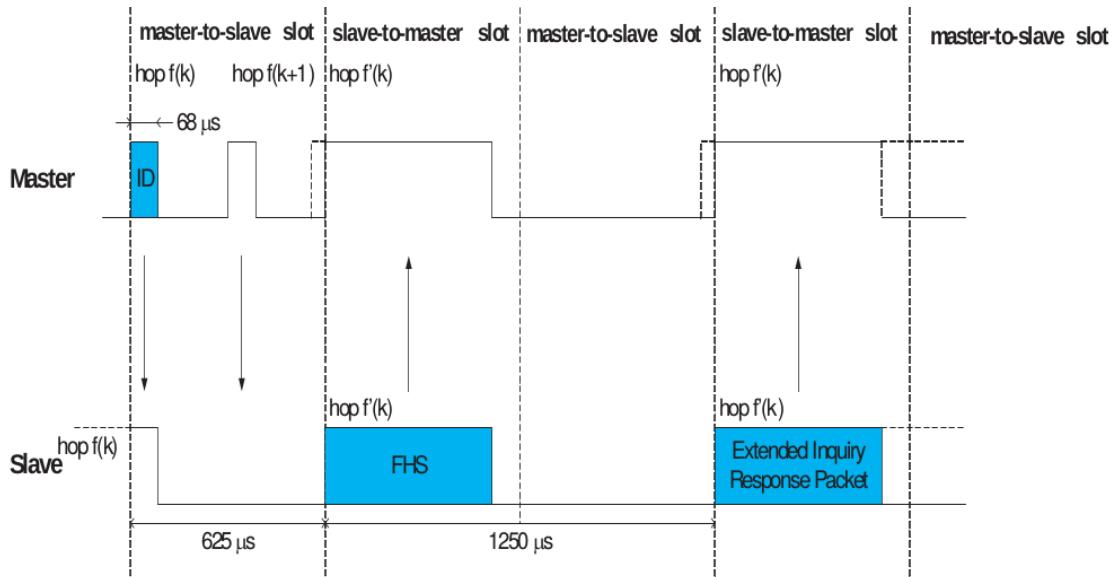


Abbildung 2.3: *Inquiry Scan* Vorgang, aus [Blue10f].

### 2.3.2 Advertising

Für das *Advertising* stellt die Bluetooth Spezifikation drei über das Frequenzband verteilte Kanäle bereit [Blue10b]. Auf diesen können BLE Geräte *Advertising Pakete* in einem unidirektionalen *Broadcast* versenden. *Advertising* Pakete dürfen auch Nutzdaten enthalten, diese können dann von anderen, scannenden BLE Geräten empfangen werden [Blue10d]. Allerdings dürfen die Nutzdaten maximal eine Länge von 31 Byte aufweisen [Blue10g]. Wurde zusätzlich im *Advertising* Paket markiert, dass das Gerät verbindungsfähig ist, kann ein scannendes Gerät nach empfangen des *Advertising* Paketes einen *Scan Request* an das verbindungsfähige Gerät richten, um die Parameter für eine Verbindung auszuhandeln [Blue10e].

## 2.4 Grundlagen von LoRaWAN

Bei *Long Range Wide Area Network* (LoRaWAN) handelt es sich um ein Netzwerkprotokoll, welches die proprietäre Modulationstechnik LoRa verwendet. Es ist für batteriebetriebene Endgeräte optimiert und wird von der LoRa Alliance spezifiziert [SLEK<sup>+</sup>15]. LoRa beherrscht neben unterschiedlichen Frequenzen des ISM Bands, in Europa beispielsweise 433 und 868 MHz, auch eine adaptive Datenrate zwischen

0,3 und 50kbps.

LoRaWAN verwendet keine Kollisionsvermeidung, stattdessen dürfen Endgeräte zu jedem Zeitpunkt senden, solange sie dabei den Kanal zufällig wählen und die maximale Sendezeit beziehungsweise relative Frequenzbelegungsdauer nicht überschreiten. Maximale Sendezeit und relative Frequenzbelegungsdauer werden dabei durch die länderspezifischen Regulierungsbehörden festgelegt.

LoRaWAN unterscheidet drei Geräteklassen. Klasse A Geräte halten nach dem Senden zwei kurze Empfangsfenster offen um eine Antwort des Kommunikationspartners zu empfangen, siehe dazu Abbildung 2.4. Klasse B Geräte öffnen zusätzliche Empfangsfenster zu festgelegten Zeiten, diese werden durch *Beacons* der *Gateways* festgelegt. Klasse C Geräte empfangen immer wenn sie nicht senden. Klasse C Geräte verbrauchen deshalb deutlich mehr Energie als Klasse A beziehungsweise B Geräte, haben dafür aber eine geringere Latenz bei Anfragen durch den Kommunikationspartner.

Soll eine Verbindung nicht direkt zwischen zwei Kommunikationspartnern, sondern über ein *Gateway* zustande kommen, muss diesem *Gateway* zunächst beigetreten werden, danach können bestätigte und unbestätigte Datenpakete über das *Gateway* und ein dahinterliegendes Netzwerk ausgetauscht werden. Tabelle 2.3 zeigt die Nachrichtentypen von LoRaWAN.

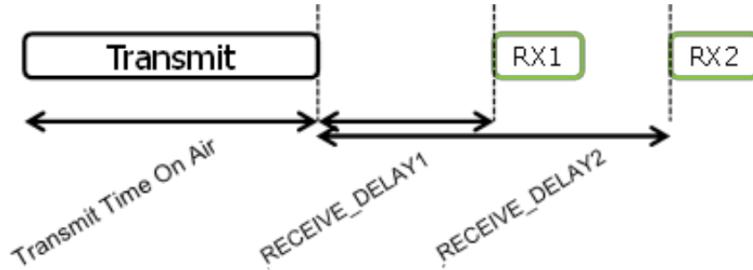


Abbildung 2.4: Senden/Empfangen-Schema von Klasse A Geräten, aus [SLEK<sup>+</sup>15].

Tabelle 2.3: Nachrichtentypen von LoRaWAN, aus [SLEK<sup>+</sup>15].

MType	Description
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	RFU
111	Proprietary



# 3. Analyse

In diesem Kapitel werden zunächst einige verwandte Arbeiten vorgestellt und danach werden auf Basis dieser die Topologie, Protokolle und Messwerte für die Implementierungen dieser Arbeit ausgewählt.

## 3.1 Verwandte Arbeiten

Die verwandten Arbeiten wurden in drei Unterkategorien unterteilt. Die erste Kategorie beschäftigt sich mit Selbstlokalisierung beziehungsweise indirekter Fernlokalisierung mittels IEEE 802.11. Die Arbeiten der zweiten Kategorie basieren ebenfalls auf IEEE 802.11, hier wird jedoch eine Fernlokalisierung oder eine indirekte Selbstlokalisierung durchgeführt. In der letzten Kategorie werden Verfahren beschrieben, die andere Protokolle verwenden.

### 3.1.1 Verwandte Arbeiten - Selbstlokalisierung & indirekte Fernlokalisierung mit IEEE 802.11

Für die Selbstlokalisierung mit IEEE 802.11 werden auf der mobilen Einheit Messwerte für empfangene Pakete bestimmt und daraus die Position der mobilen Einheit berechnet. Diese Information kann anschließend an einen Ortungsserver übertragen werden, dann handelt es sich um eine indirekte Fernlokalisierung.

#### 3.1.1.1 WiFi-LLS

Chen et al. stellen mit dem WiFi-based Local Location System (WiFi-LLS) ein System zur indirekten Fernlokalisierung vor [ChLu07]. Als Messgröße wird die Stärke des empfangenen Signals (*Received Signal Strength*, RSS) genutzt. Diese wird laut IEEE 802.11 Spezifikation als Indikator (RSSI) von der Hardware zurückgegeben.

Für die Ortung wird zunächst der RSSI von Paketen naher Access Points gemessen und zusammen mit der MAC-Adresse der mobilen Einheit in ein Paket gepackt und an den Ortungsserver versendet. Anschließend wird auf dem Ortungsserver ein theoretisches Signalausbreitungsmodell  $P(d) = P(d_0) - 10\log_{10}(\frac{d}{d_0})^n - OAF$  mit der Distanz  $d$ , der Signalstärke  $P(d)$  und der Referenzdistanz  $d_0 = 1\text{m}$  zur Bestimmung

Tabelle 3.1: Überblick über die verwandten Arbeiten

Arbeitstitel	Protokoll	Topologie	Messgröße	Lokalisierungsprinzip
Design and implementation of a wifi-based local locating system	IEEE 802.11	Indirekte Fernlokalisierung	RSSI	Trilateration
On indoor position location with wireless LANs	IEEE 802.11	Direkte Selbstlokalisierung	RSSI	Szenenanalyse
RADAR: An in-building RF-based user location and tracking system	IEEE 802.11	Direkte Fernlokalisierung	RSSI	Szenenanalyse
Time of flight ranging using off-the-self ieee802.11 wifi tags	IEEE 802.11	Direkte Fernlokalisierung	TOF	Trilateration
MonoPHY: Mono-stream-based device-free WLAN localization via physical layer information	IEEE 802.11n	Direkte Fernlokalisierung	CSI	Szenenanalyse
A comprehensive study of bluetooth signal parameters for localization	Bluetooth	-	LQ, RSSI, TPL	-
Indoor localization based on response rate of bluetooth inquiries	Bluetooth	Direkte Fernlokalisierung	Response Rate	Szenenanalyse
Inquiry-based bluetooth indoor positioning via RSSI probability distributions	Bluetooth	Direkte Fernlokalisierung	RSSI	Szenenanalyse
RSSI based Bluetooth low energy indoor positioning	Bluetooth Low Energy	Direkte Fernlokalisierung	RSSI	Trilateration
Low-complexity Outdoor Localization for Long-range, Low-power Radios	LoRa	Direkte Fernlokalisierung	TOA	Trilateration

der Position der mobilen Einheit verwendet.

$P(d_0)$ , der Pfadverlustexponent  $n$  und der Hindernisdämpfungsfaktor  $OAF$  müssen bestimmt werden, jedoch lassen sich  $P(d_0)$  und  $n$  auf einer einzelnen Teststrecke mit unterschiedlichen Abständen von AP und mobiler Einheit bestimmen.  $OAF$  kann sogar für einen Gebäudetyp einmalig bestimmt werden. Dadurch hat das Modell einen konstanten Aufwand. Dies ist für Baustellen interessant, da sich diese Werte einmalig messen und dann sogar über mehrere gleichartige Baustellen übertragen ließen.

In dieser Veröffentlichung steht die Ortungsgenauigkeit im Vordergrund und es werden keine Angaben zum Energieverbrauch gemacht. Als Referenz kann dienen, dass

die mobile Einheit bei WiFi-LLS alle 5 Sekunden einen *Scan* (siehe Abschnitt 2.2.1) durchführt, dann werden die Signalstärken entdeckter APs zusammen mit der eigenen MAC-Adresse in XML codiert und das so erzeugte Paket an den Ortungsserver versendet.

### 3.1.1.2 AiRISTA Flow RTLS

*Ekahau* bietet unter der Marke *AiRISTA Flow RTLS* eine zu WiFi-LLS ähnliche Lösung kommerziell an [AiRI17c]. Ihr *Ekahau B4 Badge Tag* ermittelt regelmäßig den RSSI zu nahegelegenen APs und versendet diese an einen Ortungsserver [LDBL07]. Das Tag bietet darüber hinaus noch einige Zusatzfunktionen, so können über die Datenverbindung auch Nachrichten und Alarmierungen an das Tag gesendet werden und die drei angebrachten Knöpfe können programmiert werden.

Bezüglich des Energieverbrauchs gibt sich das Informationsblatt des *B4 Badge Tag* vage: Das Tag soll abhängig vom Ortungsintervall wochenlang halten, danach muss der 600 mA/h Akku geladen werden [AiRI17b]. Das Informationsblatt zum *Ekahau W4*, welches statt um den Hals am Handgelenk getragen wird, gibt an, dass der verbaute 530 mA/h Akku bei einem Ortungsintervall von 15 Sekunden 500 Stunden (ca. 21 Tage) hält [Ekah17].

*AiRISTA Flow* spricht auf ihrer Website zum Beispiel Krankenhäuser, Schulen und Regierungseinrichtungen an, hier sollen zusätzlich bewegliche Objekte, wie etwa Krankenhausbetten, geortet werden. Die dazu verwendeten Asset Tags werden über einen Beschleunigungssensor aktiviert und können, wenn die Objekte selten bewegt werden, deutlich längere Laufzeiten erreichen [AiRI17a].

Im Umfeld des Tunnelbaus zeigte sich jedoch, dass die Tags keinesfalls wochenlange Laufzeiten aufwiesen, stattdessen entluden sie sich teilweise in unter zwölf Stunden.

### 3.1.1.3 AeroScout

Auch das *AeroScout* System von *Stanley Healthcare* richtet sich an den medizinischen Sektor und soll Objekte und Personen orten [Stan17a], [Stan17b]. Da sich auch dieses System in das bestehende WLAN-Netzwerk einfügt, sollte es ebenfalls auf einer indirekten Fernlokalisierung beruhen und demnach ähnliche Eigenschaften bezüglich des Energieverbrauchs aufweisen.

Das Informationsblatt ihres *T14* Tags für Personen gibt eine Laufzeit von bis zu drei Wochen, abhängig von Konfiguration und Typ des Tags, an [Stan17c]. Eine Angabe zu dem verwendeten Typ, der Konfiguration oder der Kapazität des verbauten Akkus wird nicht gemacht.

### 3.1.1.4 Selbstlokalisierung mit Szenenanalyse

Prasithsangaree et al. stellen ein System zur Selbstlokalisierung vor [PrKC02], es verwendet aber eine *Offline-Phase* zum Sammeln von Fingerabdrücken für Positionen in einem Abstand von 1,5 beziehungsweise 3 Metern. In diesen Fingerabdrücken werden die gemessenen RSSI der von den APs empfangenen Pakete als Merkmale zusammen mit der Position als Label gespeichert. In der anschließenden *Online-Phase* werden die gemessenen RSSI mit den Fingerabdrücken verglichen und die Position als gewichtetes Mittel der Labels bestimmt.

Die *Offline-Phase* ist natürlich im Sinne der Aufgabenstellung nicht sinnvoll, da für

eine Tunnelbreite von im Schnitt zehn Metern 4000 beziehungsweise 2000 Messungen pro Kilometer vorgenommen werden müssten. Generell eignen sich Lösungen mit Szenenanalyse nicht gut für Baustellen, da diese nicht auf die potentiell höhere Genauigkeit angewiesen sind. Üblicherweise müssen dort sehr große Flächen vermessen werden und die Veränderungen durch den Baufortschritt führen dazu, dass regelmäßig neu gemessen werden muss. Außerdem müssen bewegliche Störquellen wie Baummaschinen vorher aus dem Bereich entfernt werden, um unverfälschte Fingerabdrücke zu erhalten. Der Aufwand ein System mit Szenenanalyse auf einer Baustelle zu betreiben ist deshalb sehr hoch und widerspricht der Forderung nach geringer Komplexität.

Die Arbeit zeigt aber die Volatilität der empfangenen Signalstärke auf, dies wurde 2011 von Lui et al. genauer untersucht [LGKD<sup>+</sup>11]. Lui et al. zeigen, dass die gemessene empfangene Signalstärke stark von der beteiligten Hardware abhängt und die Systeme jedes mal neu kalibriert werden müssen wenn sie auf ein neues AP-Modell portiert werden. Auf dem Areal sollte deshalb optimalerweise nur ein AP-Modell verwendet werden.

Abbildung 3.1 zeigt die gemessenen RSSI Werte für die von ihnen getesteten Netzwerk-karten mit unterschiedlichen Distanzen, für einige Karten korreliert die empfangene Signalstärke nur sehr schwach mit der Distanz zwischen Basisstation und mobiler Einheit. Sie zeigen außerdem, dass einige AP-Modelle den RSSI speichern und nur bei größeren Veränderungen aktualisieren und dass die Antenne signifikanten Einfluss auf den protokollierten Wert hat.

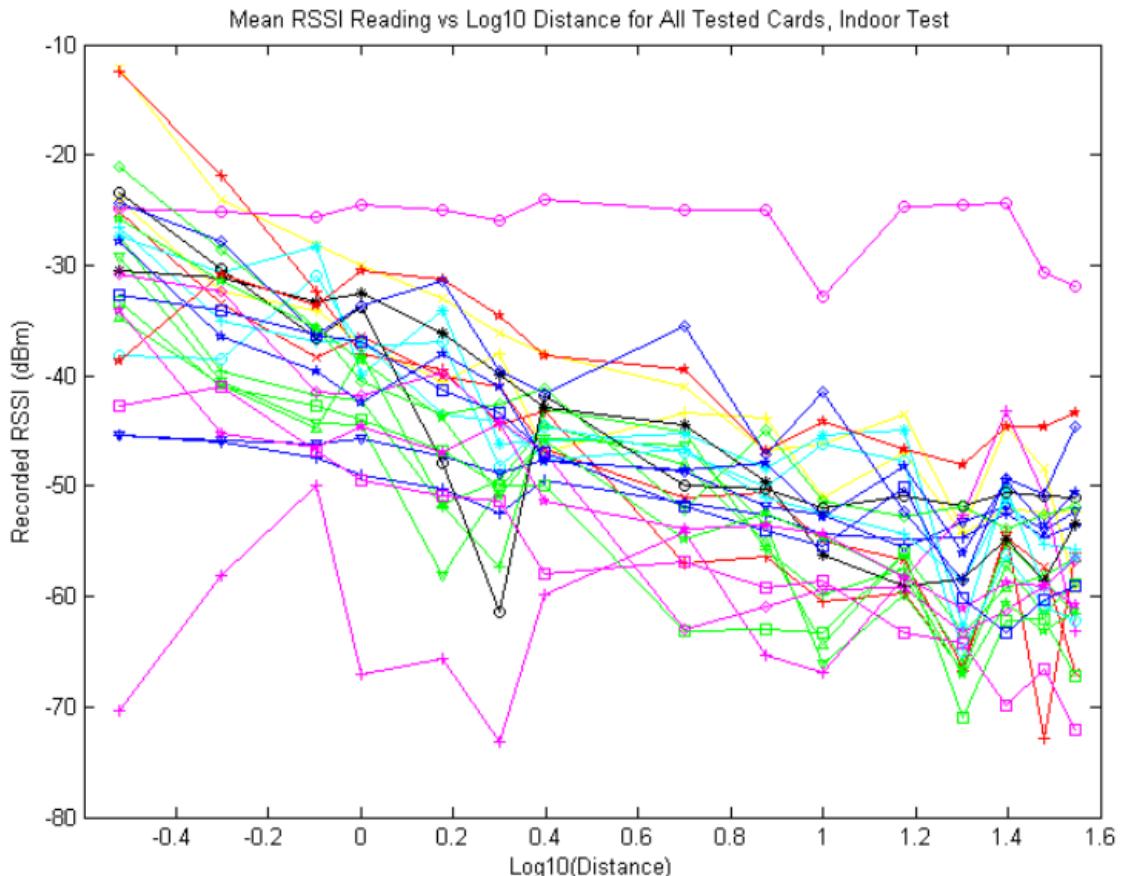


Abbildung 3.1: Gemessener RSSI mit verschiedenen Access Points und Distanzen, aus [LGKD<sup>+</sup>11].

### 3.1.2 Verwandte Arbeiten - Fernlokalisierung & indirekte Selbstlokalisierung mit IEEE 802.11

Für die Fernlokalisierung mit IEEE 802.11 werden auf den Basisstationen Messwerte für empfangene Pakete bestimmt und daraus die Position der mobilen Einheit berechnet. Diese Information kann anschließend an die mobile Einheit übertragen werden, dann handelt es sich um eine indirekte Selbstlokalisierung.

#### 3.1.2.1 RADAR

Das *RADAR* System von Bahl et al. (Microsoft Research) hat als eins der ersten WLAN-basierten Ortungssysteme viel Aufmerksamkeit erfahren [BaPa00]. Als Messgröße wird die Stärke des empfangenen Signals (*Received Signal Strength*, RSS) genutzt, diese wird laut IEEE 802.11 Spezifikation als Index (RSSI) von der Hardware zurückgegeben. Das *RADAR* System ist auf eine *Offline-Phase* angewiesen, in der empirisch ein Signalausbreitungsmodell aufgebaut wird. Es handelt sich also um ein System mit Szenenanalyse.

Die Verwendung einer *Offline-Phase* ist im stark veränderlichen Baustellenumfeld nicht akzeptabel. Zum einen führt der ständige Baufortschritt dazu, dass regelmäßig neu kalibriert werden muss und zum anderen wirken sich auch die großen Baumaschinen auf die Signalausbreitung aus. Damit sich dies nicht im Modell wiederfindet, müssten zunächst alle beweglichen Maschinen aus dem Bereich entfernt werden, um anschließend in der *Online-Phase* ihren Einfluss glätten zu können. Die *Offline-Phase* ist deshalb wirtschaftlich gesehen nicht durchführbar und das empirisch ermittelte Signalausbreitungsmodell müsste durch ein theoretisches ersetzt werden, für eine grobkörnige Bereichsortung sollte dies jedoch ausreichend sein.

Bei *RADAR* sendet die mobile Einheit vier UDP-Pakete pro Sekunde aus, an den Basisstationen wird dann der RSSI gemessen. Die Autoren weisen jedoch darauf hin, dass sich dieser Vorgang leicht umkehren ließe, um von einer Fernlokalisierung auf eine Selbstlokalisierung zu kommen. Bezuglich des Energieverbrauchs äußern sie sich jedoch zu keiner der beiden Varianten.

Die Position wird anschließend bestimmt, indem aus den in der *Offline-Phase* aufgenommenen Werten derjenige mit dem geringsten Abstand zu den gemessenen Werten gewählt wird. Dies wird im *nearest neighbour in signal space* (NNSS) Algorithmus beschrieben. Für die Ortung wird mehrfach gemessen und dann gemittelt, um im Median eine Genauigkeit von unter 3 Metern zu erhalten. Das kurze Sendeintervall von 0,25 Sekunden führt auch bei bewegten Personen zu einer Genauigkeit von 3,5 Metern. Gleichzeitig sorgt das kurze Sendeintervall aber auch für einen hohen Energieverbrauch auf Seiten der mobilen Einheit, eine Reduktion der Sendevorgänge sollte im Kontext der Bereichsortung angestrebt werden, um den Energieverbrauch zu senken und die Batterielaufzeit der mobilen Einheit zu steigern.

#### 3.1.2.2 Verbesserungen an RADAR

Bahl et al. veröffentlichten anschließend noch einige Verbesserungen für das ursprüngliche *RADAR* System [BaPB00]. Diese umfassen unter anderem den Einsatz von Access Points statt PCs als Basisstationen, verbesserte Ortung bewegter Personen und die Erkennung von hinzugekommenen Hindernissen wie etwa Personen. Letzteres geschieht durch die Analyse der Signalstärke von *Beacons* anderer APs, da diese sich nicht bewegen, können Veränderungen in der Signalstärke als Veränderungen auf dem Signalweg gesehen werden. Dies ließe sich auch auf größere Hindernisse

übertragen, hängt aber stark von der strategischen Platzierung und möglichst dichten Verteilung der APs ab.

Auch hier äußern sich die Autoren nicht zum Energieverbrauch, wohl auch deshalb weil sie einen Laptop als mobile Einheit verwenden.

### 3.1.2.3 Time-of-flight Lokalisierung

Aufgrund der Schwächen von RSS-basierten Systemen wurde auch über solche nachgedacht, die stattdessen oder zusätzlich die *Time of Flight* (TOF) messen. Ein Beispiel für ein solches zeigen Wibowo et al. [WiKP09]. Sie fordern optimalerweise Zugriff auf die physische Schicht (PHY) des IEEE 802.11 Protokolls, da auf diesen aber üblicherweise kein Zugriff besteht, messen sie TOF in der darüber liegenden MAC-Schicht.

Eine Basisstation sendet einen *Beacon* aus und protokolliert die Sendezeit, die mobile Einheit empfängt den *Beacon*, protokolliert die Empfangszeit und die Sendezeit der gesendeten Antwort, die Basisstation sichert die Empfangszeit der Antwort. Nun sendet die mobile Einheit die zwei gespeicherten Zeitstempel an die Basisstation, der mit diesen die Verarbeitungszeit auf der mobilen Einheit berechnen kann, um dann die Distanz  $d = c * (\frac{t_{\text{empfangen}} - t_{\text{gesendet}} - t_{\text{Verarbeitung}}}{2})$  zur mobilen Einheit zu bestimmen. Dieses Schema lässt sich leicht von einer Fernlokalisierung in eine Selbstlokalisierung umwandeln, indem man den Initiator des Vorgangs tauscht.

Die Notwendigkeit die Zeitstempel bereits in der PHY- beziehungsweise MAC-Schicht zu setzen erfordert Zugriff auf die Software des als Basisstation verwendeten Access Points. Außerdem müssen die Zeitstempel im Bereich von Nanosekunden gesetzt werden und die Verarbeitungszeit vor/nach dem Setzen muss sehr konstant sein, da eine Abweichung von 100 ns bei  $c = 299.792.458 \text{ m/s}$  bereits einen Fehler von 30 Metern verursacht.

Muthukrishnan et al. beschreiben diese Problematik bei dem Versuch TOF ohne Zugriff auf die Software des APs umzusetzen [MKML06]. Sie kommen zu dem Ergebnis, dass sich die in der Spezifikation eingebauten Zeitstempelfunktionen wie das *Network Time Protocol* (NTP), *Ping* und die Zeitstempel in *Beacons* nicht eignen, da sie zum einen nur eine Auflösung im Millisekundenbereich bieten und zum anderen von der Blockierungskontrolle von IEEE 802.11 (CSMA/CA) abhängen.

### 3.1.2.4 Ortung ohne mobile Einheit

Eine Ortung ohne mobile Einheit erfüllt wegen ihrer Abwesenheit offensichtlich jede Anforderung an die Batterielaufzeit der mobilen Einheit. Mit *MonoPHY* stellen Abdel-Nasser et al. ein System zur Ortung ohne mobile Einheit vor [ANSSY13].

Dazu verwenden sie einen IEEE 802.11n-fähigen Laptop und Access Point und analysieren die *Channel State Information* (CSI) der physischen Schicht (PHY) der zwischen AP und Laptop übertragenen Daten. Um die bestehende Struktur von APs zu nutzen, sollte das System angepasst und die CSI zwischen den APs gemessen werden. Es hat jedoch einige Aspekte, die es ungeeignet für die Aufgabenstellung machen.

Das System unterscheidet nicht zwischen Personen, sondern erkennt nur, dass jemand anwesend ist. Außerdem ist es nur für eine Person in einem  $100 \text{ m}^2$  Apartment gestaltet worden und müsste auf Baustellengröße und die Verfolgung vieler Personen erweitert werden. Aber auch dann ist fraglich, wie gesichert werden kann, dass alle Personen durchgehend erkannt werden können, zum Beispiel wenn sich mehrere

Personen auf oder in einem Transportfahrzeug aufzuhalten. Auch ist es ohne Identifikation schwerer Fehler zu erkennen. Wird fälschlicherweise angezeigt, dass sich noch eine Person im Tunnel befindet, kann oft durch ausrufen des Betreffenden festgestellt werden, dass dieser nicht im Tunnel ist. Hat man dagegen nur die Information, dass sich noch eine Person im Tunnel befindet, hat man keine Möglichkeit schnell herauszufinden ob dies der Wahrheit entspricht.

Weitere Probleme entstehen durch die Baumaschinen und Container. Diese haben einen starken Einfluss auf die CSI und verdecken dadurch möglicherweise nahe Personen und die Fahrzeugführer. Deshalb müssten diese Objekte ebenfalls als Entitäten angezeigt werden. Die Anzeige diverser Kommandostände, Pausenräume und stehen gelassener Baumaschinen verwirrt im Notfall jedoch, da sich in jedem Objekt potentiell eine Person befinden könnte.

Die Veröffentlichung beruht außerdem auf der Verfügbarkeit der CSI, diese sind dort durch die Auswahl einer bestimmten Netzwerkkarte gegeben und sind nicht zwingend in einer bestehenden Struktur von APs verfügbar. Als letzter Kritikpunkt steht die Verwendung einer *Offline-Phase*, die, wie bereits diskutiert, wirtschaftlich nicht umsetzbar ist.

Somit erfüllt die Ortung ohne mobile Einheit zwar die Anforderungen an den Energieverbrauch, jedoch nicht die Forderung nach sicherer Erkennung von Abschnittswechseln, deshalb scheidet diese Technik zumindestens für Baustellen aus.

### 3.1.3 Verwandte Arbeiten - Funkbasierte Lokalisierung mit weiteren Funkprotokollen

Die Arbeiten in dieser Kategorie verwenden nicht IEEE 802.11 für die Lokalisierung. Sie verwenden stattdessen andere Protokolle, die sich durch einen geringeren Energieverbrauch als IEEE 802.11 auszeichnen sollen.

#### 3.1.3.1 Geeignete Messwerte für Bluetooth

Hossain et al. untersuchen die laut Bluetooth Spezifikation zurückgegebenen Messwerte bezüglich ihrer Eignung für die Lokalisierung [HoSo07].

*Link Quality* (LQ) beschreibt, wie gut die Verbindung zwischen zwei Geräten ist. Der Wert wird aus der Bitfehlerrate beim Empfänger berechnet, allerdings ist nicht spezifiziert, wie der Wert zu berechnen ist, er hängt also in hohem Maße vom Hersteller des Empfängers ab.

Der *Received Signal Strength Indicator* (RSSI) misst die Stärke des eingehenden Signals, die Spezifikation sieht jedoch eine *Golden Receive Power Range* (GRPR) vor. Liegt der RSSI über oder unter dieser wird eine Anfrage zum erhöhen oder verringern der Sendeleistung an das andere Gerät verschickt, dies dient während einer aktiven Verbindung dazu den Energieverbrauch zu senken. Problematisch am RSSI ist, dass er relativ zur GRPR bestimmt wird. In der Untersuchung von Hossain et al. führte das dazu, dass der RSSI mit 0 gemessen wurde, wenn er innerhalb der GRPR lag.

*Transmission Power Level* (TPL) ist die Sendeleistung eines Geräts. TPL kann während einer bestehenden Verbindung durch Anfragen des Verbindungspartners verändert werden. Dazu muss diese Energiesparfunktion jedoch unterstützt werden, was bei dem von Hossain et al. verwendeten Gerät nicht der Fall war. Abbildung 3.2 zeigt deshalb für TPL eine waagerechte Linie.

Für *Inquirys* wird die Stärke des eingehenden Signals ohne die Beachtung des GRPR

bestimmt. Außerdem werden *Inquiry*s immer mit voller Sendeleistung gesendet, da sie zur Entdeckung von Ressourcen verwendet werden. Es handelt sich ebenfalls um den RSSI, um jedoch den Unterschied zum RSSI für eine Verbindung deutlich zu machen nennen Hossain et al. diesen Wert *RX Power Level*.

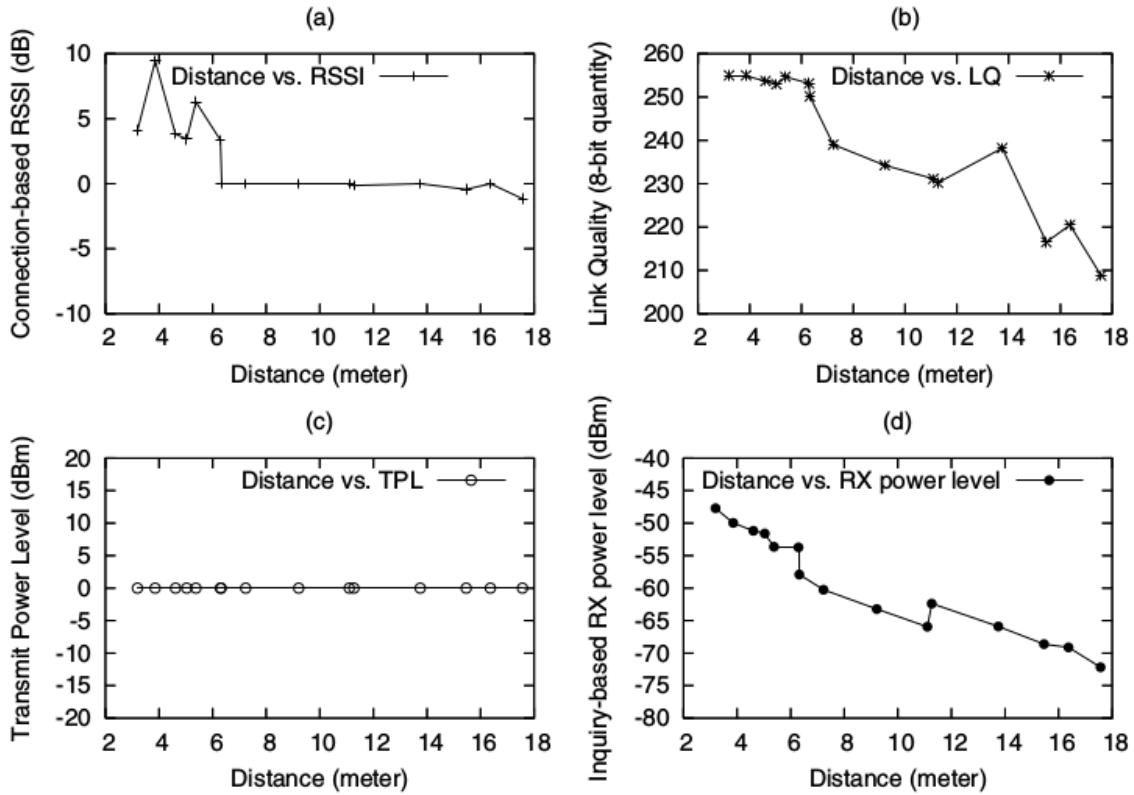


Abbildung 3.2: Korrelation der Messwerte mit der Distanz, aus [HoSo07]

### 3.1.3.2 Antwortbasierte Inquiry Lokalisierung

Bargh et al. stellen ein System zur Fernlokalisierung mittels *Bluetooth Inquiry Scan* vor [BadG08]. Die mobilen Einheiten sind dabei im *discoverable* ("entdeckbar") Modus. Die Basisstationen senden regelmäßig eine *Inquiry Message* aus, die dann von mobilen Einheiten in Reichweite mit einer *Inquiry Response* beantwortet werden. Anschließend werden die Antworten auf einem zentralen Ortungsserver gesammelt und die Positionen der mobilen Einheiten bestimmt. Diese Information wird aus den beantworteten *Inquiry Messages* jeder mobilen Einheit gewonnen. Dazu werden vorher in einer *Offline-Phase* Fingerabdrücke für jeden Raum gesammelt. In diesen wird gespeichert welche *Inquiry Messages* von der mobilen Einheit an einem bestimmten Ort beantwortet wurden. Für die Bereichsortung entfällt diese *Offline-Phase*, da der Bereich über die Beantwortung einer einzelnen *Inquiry Message* implizit bestimmt werden kann. Tabelle 3.2 zeigt die abnehmende Wahrscheinlichkeit für eine Antwort auf die *Inquiry Message* mit steigender Distanz.

### 3.1.3.3 RSSI-basierte Inquiry Lokalisierung

Ling et al. lokalisieren Bluetooth-fähige Geräte über den RSSI der *Inquiry Response* [LRJT<sup>+</sup>10]. Ihre Basisstationen versenden regelmäßig *Inquiry Messages* und messen den RSSI der *Inquiry Responses*. Mobile Einheiten, die geortet werden wollen beziehungsweise sollen, bewerben in der *Inquiry Response* einen speziellen Service. Dies

Tabelle 3.2: Rate der beantworteten *Inquiry Messages* (*Inquiry Response Rate*, IRR) gegen Distanz, aus [BadG08]

location of the object device	distance of devices	average of IRR	variance of IRR
in the room of the Bluetooth dongle	2 m	97 %	1.8
in the hall outside the dongle room	4 m	97 %	1.2
	6 m	99 %	1
	8 m	94 %	1.4
	10 m	86 %	4.6
	12 m	74 %	9.6
	14 m	67 %	4.6
in room across the hall	16 m	NULL	NULL
	18 m	NULL	NULL

erlaubt es diese herauszufiltern und die Privatsphäre anderer Personen zu wahren. Die Autoren verwenden eine anfängliche *Offline-Phase* um Fingerabdrücke für den RSSI an gegebenen zu finden. Allerdings messen sie dazu an den Basisstationen die empfangene Signalstärke von Übertragungen anderer Basisstationen, folglich kann die *Offline-Phase* automatisiert durchgeführt werden. Für die *Online-Phase* werden die gemessenen RSSI Werten über eine Wahrscheinlichkeitsverteilung geglättet, um eine bessere Ortungsgenauigkeit zu erreichen. Für die eigentliche Lokalisierung werden die gemessenen RSSI mit den Wahrscheinlichkeitsverteilungen aus der *Offline-Phase* verglichen und die Wahrscheinlichkeit für die Emission dieser Werte über die Position maximiert.

### 3.1.3.4 RSSI-basierte BLE Lokalisierung

Jianyong et al. stellen ein System zur Lokalisierung auf Basis von *Bluetooth Low Energie* (BLE, auch Bluetooth Smart) vor [JHZZ14].

Sie messen an den Basisstationen den RSSI von *Advertising Paketen*, die zuvor von den mobilen Einheiten versendet wurden. Für die genaue Ortung werden die Ergebnisse mit einem Gauß-Filter geglättet und für jede Basisstation die Parameter für ein Signalausbreitungsmodell bestimmt. Die gemessene Signalstärke  $P = A - 10n \cdot \log(d)$  hängt von der Referenzsignalstärke A im Abstand von einem Meter, dem Dämpfungsfaktor n und der Distanz d ab.

Jianyong et al. bestimmen die Referenzsignalstärke und den Dämpfungsfaktor für jede Basisstation. Da jedoch nur eine Bereichsortung für diese Arbeit gefordert wurde, sollten A und n nur beispielhaft für eine Basisstation bestimmt werden, um den Aufwand beim Aufbau der Infrastruktur zu reduzieren. Abbildung 3.3 zeigt die von Jianyong et al. bestimmten Parameter für das Signalausbreitungsmodell. Abbildung 3.3 zeigt außerdem, dass die verwendeten *CC2540 Development Kit* von Texas Instruments von einer Basisstation nur auf 20 Meter detektiert werden konnte.

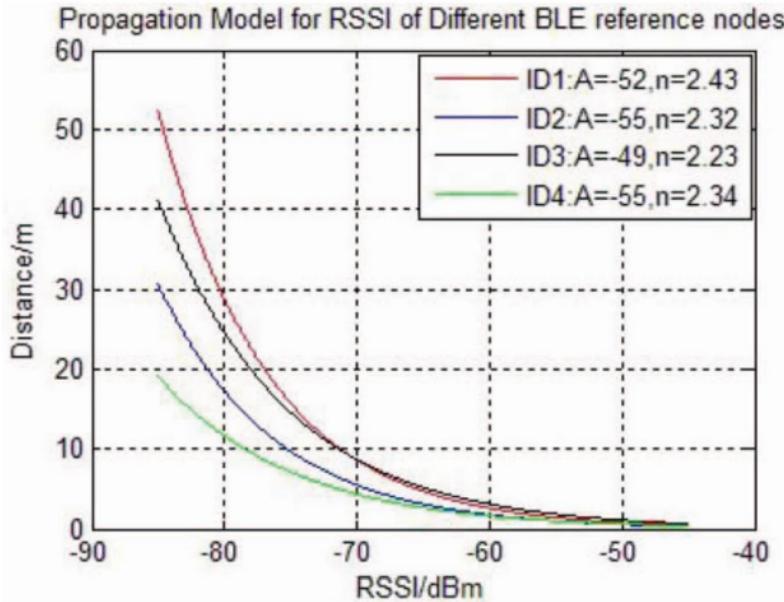


Abbildung 3.3: Signalausbreitungsmodelle aus [JHZZ14]

### 3.1.3.5 Lokalisierung mit LoRa

Kim et al. stellen ein System zur Lokalisierung mit *Long Range* (LoRa) vor [KiKo16]. Sie beziehen sich dabei auf eine Lokalisierung im Außenbereich in einem 30x30km Areal. Sie senden Pakete von der mobilen Einheit und messen die *Time of Arrival* (TOA) an den Basisstationen. Dazu müssen die Basisstationen zeitsynchron arbeiten, dies wird über GPS sichergestellt.

Eine Lokalisierung über den RSSI lehnen sie ab, da dieser im Bereich von 20 bis 30km Entfernung nur um 3-6dBm fällt und die Varianz die Korrelation von RSSI und Distanz über lange Distanzen überdeckt. Sie erreichen, abhängig von der Anzahl der verwendeten Basisstationen eine Genauigkeit zwischen wenigen Hundert und fast eintausend Metern, siehe dazu Abbildung 3.4.

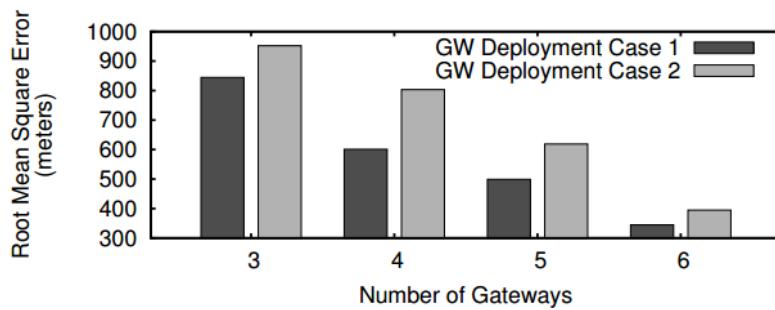


Abbildung 3.4: Quadratischer Fehler der Lokalisierung aus [KiKo16], *Gateways* stellen Basisstationen dar.

## 3.2 Auswahl des Funkprotokolls

Soll das bestehende WLAN-Netzwerk genutzt werden ist die Wahl auf die IEEE 802.11 Spezifikation beschränkt.

Kann man aber die Hardware für die Basisstationen frei wählen, kann auch das Protokoll frei gewählt werden. Es werden deshalb zusätzlich BLE und LoRa betrachtet.

*Bluetooth Low Energy* wird gegenüber normalem Bluetooth aufgrund seiner Charakteristik als Protokoll für geringen Energieverbrauch bevorzugt.

LoRa bietet zwar theoretisch eine sehr hohe Reichweite, diese ist jedoch stark von verwendeten Antennen und den Hindernissen abhängig. Ob LoRa einen signifikanten Vorteil bei der Reichweite hat muss geprüft werden. Sollte dieser Vorteil bestehen, bietet LoRa durch seine bessere Abdeckung eine höhere Sicherheit bei der Erkennung von Abschnittswechseln. Außerdem ist es dadurch potentiell möglich auf triangulierende Positionen zu wechseln ohne mehr Basisstationen aufzubauen zu müssen.

### 3.3 Auswahl der Topologie

Das Wissen um die Position der Nutzer soll nach der Ortung dem Sicherheitssystem der Baustelle zur Verfügung stehen. Das Sicherheitssystem fungiert daher konzeptionell auch als zentraler Ortungsserver. Es muss deshalb eine Fernlokalisierung durchgeführt werden, diese kann aber sowohl direkt als auch indirekt durchgeführt werden.

Weil laut IEEE 802.11 keine für die Lokalisierung geeigneten Messwerte vom AP in das angeschlossene Netzwerk propagiert werden, kommt keine der verwandten Arbeiten zur direkten Fernlokalisierung mit IEEE 802.11 ohne Zugriff auf die Software des AP aus. Es werden deshalb sowohl die direkte, als auch die indirekte Fernlokalisierung mit IEEE 802.11 untersucht, obwohl eine Lösung mit indirekter Fernlokalisierung wegen des nötigen Empfangs von Signalen potentiell mehr Energie verbraucht. Dürfen Veränderungen der Hardware vorgenommen werden, bietet sich ebenfalls eine direkte Fernlokalisierung an, da dort die mobilen Einheiten, abgesehen von der Kollisionsdetektion/-vermeidung, nicht empfangen müssen.

Eine Topologie ohne Basisstationen kommt nicht in Frage, da diese Topologie Einzelpersonen insbesondere in Notsituationen nicht ausreichend gut erfasst.

### 3.4 Auswahl der Messgrößen

Da nur eine Bereichsortung durchgeführt werden soll, sind die Anforderungen an die erzielte Genauigkeit gering. Stattdessen sollte darauf geachtet werden, dass die Anforderung für die Messung gering ist. Messgrößen, die Zeitsynchronität voraussetzen sind daher nicht geeignet.

Als Messgröße wird der *Received Signal Strength Indicator* (RSSI) für alle drei Technologien gewählt, da dieser bei jedem empfangenen Paket von der physischen Schicht gemessen und am empfangenden Gerät leicht ausgelesen werden kann.

Auch bei LoRa wird der RSSI entgegen der Empfehlung von Kim et al. gewählt. Da sich das Szenario im Gegensatz zu dem aus ihrer Veröffentlichung unter Tage abspielt kann keine Synchronisierung über GPS gewährleistet werden. Außerdem müssen die Distanzen zwischen den Basisstationen geringer gewählt werden, da die Ungenauigkeiten aus dieser Veröffentlichung zu hoch für das Szenario sind.

### 3.5 Auswahl der Hardware

Die verwendeten Radios müssen die physische Schicht des jeweils verwendeten Protokolls beherrschen. Es werden deshalb Radios für 802.11, Bluetooth 4.0 oder höher und LoRaWan benötigt.

Die Feather-Serie von Adafruit beinhaltet viele eigenständige Entwicklungs-Boards, darunter finden sich für jede der benötigten Funktechnologien ein Entwicklungs-Board. Das *Adafruit Feather HUZZAH ESP8266* wird für mobile Einheiten mit IEEE 802.11 eingesetzt. Mobile Einheiten mit Bluetooth werden mit dem *Adafruit Feather nRF52 Bluefruit* implementiert, es unterstützt Bluetooth 5.0 inklusive BLE. Unterstützung für LoRa bietet das *Adafruit Feather M0 RFM95 LoRa Radio (900MHz)*, mit diesem werden mobile Einheiten mit LoRa implementiert. Als Akku wurde ein Lithium-Polymer-Akkumulator mit 1400 mAh Kapazität ausgewählt. Er wurde als leicht genug angesehen, um um den Hals getragen zu werden, ein vergleichbarer 2200mAh Akkumulator erschien dafür zu schwer.

## 3.6 Weiteres Vorgehen

Im Folgenden werden Prototypen für die drei ausgewählten Protokolle erstellt, IEEE 802.11 ist dabei in indirekte und direkte Fernlokalisierung geteilt, da die direkte Fernlokalisierung Zugriff auf die Software der Basisstationen erfordert. Zusätzlich wird die Reichweite jeder Funktechnologie im Tunnel überprüft, da diese für das Intervall der Ortungsvorgänge und die Erkennungssicherheit wichtig ist. Abschließend werden in jedem Kapitel die Charakteristika des Stromverbrauchs der jeweiligen mobilen Einheiten überprüft.

## 4. Indirekte Fernlokalisierung mit IEEE 802.11

Bei der indirekten Fernlokalisierung wird die Messgröße auf der mobilen Einheit gemessen. Die Ortungsinformation wird dann implizit durch versenden der gemessenen Werte oder explizit durch versenden der berechneten Position an den Ortungsserver übermittelt. Wird IEEE 802.11 verwendet muss die mobile Einheit mit einem Access Point assoziiert um Informationen an den Ortungsserver übermitteln zu können. Eine Lösung zur indirekten Fernlokalisierung muss deshalb mindestens auf Protokollebene vier des OSI-Modells arbeiten. Für die Übermittelung steht ein WLAN-Netzwerk zur Verfügung.

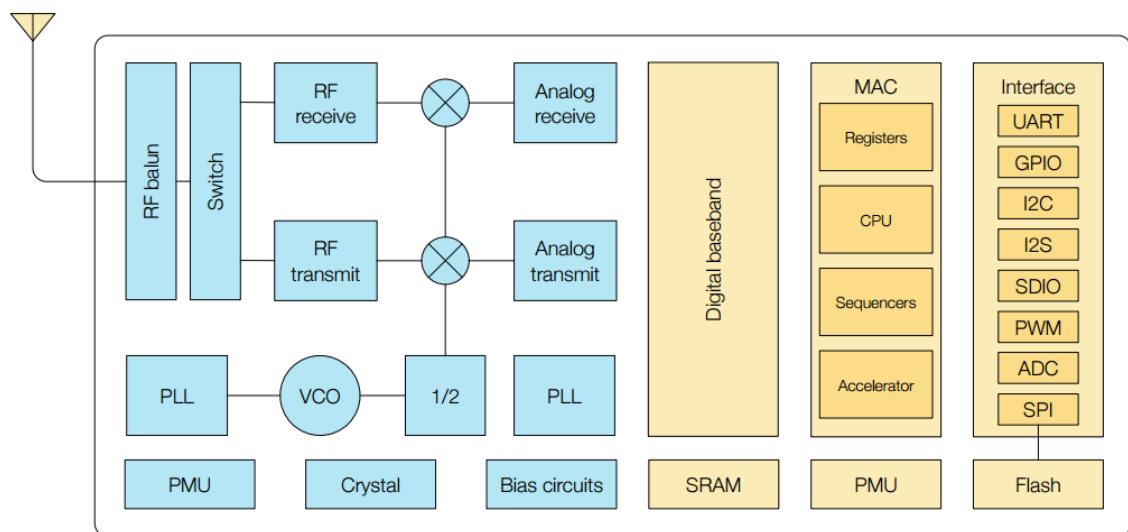


Abbildung 4.1: Blockdiagramm des ESP8266, aus [Espr17]

### 4.1 ESP8266

Der ESP8266 soll als Hardware für das Tag eingesetzt werden, dabei handelt es sich um einen Microcontroller von Espressif. Der ESP8266 besitzt neben einer CPU

eine 802.11b/g/n/e/i-fähige WLAN-Einheit und beherrscht diverse andere, kabelgebundene Kommunikationsstandards wie zum Beispiel GPIO, I2C und SPI, siehe Abbildung 4.1.

Da der ESP8266 selbst weder über Flashspeicher, noch über eine Antenne verfügt wird er auf einem Modul mit diesen Komponenten verbaut. Die in dieser Arbeit betrachteten Module sind das ESP-12S und das ESP-12F. Das neuere ESP-12F sollte eine höhere Reichweite bei der Funkübertragung entfalten, dies wird noch Gegenstand eines Experiments sein. Abbildung 4.2 zeigt die beiden Module nebeneinander, die unterschiedlichen Antennenformen sind deutlich zu erkennen.

Espressif gibt im Datenblatt auch Aufschluss über den Energieverbrauch des ESP8266, siehe dazu Tabelle 4.1. Für die Prototypenentwicklung wird ein ESP-12S Modul auf einem Adafruit Feather Huzzah verwendet, dieses stellt mit dem CP2104 eine serielle Verbindung zum ESP her, reguliert die Spannung für das Modul auf 3,3V und bringt den 2mm Pinabstand des ESP-12S Moduls auf die für Breadboards üblichen 2,5mm. Das Adafruit Feather Huzzah ESP8266 mit ESP-12S ist in Abbildung 4.2 links abgebildet.

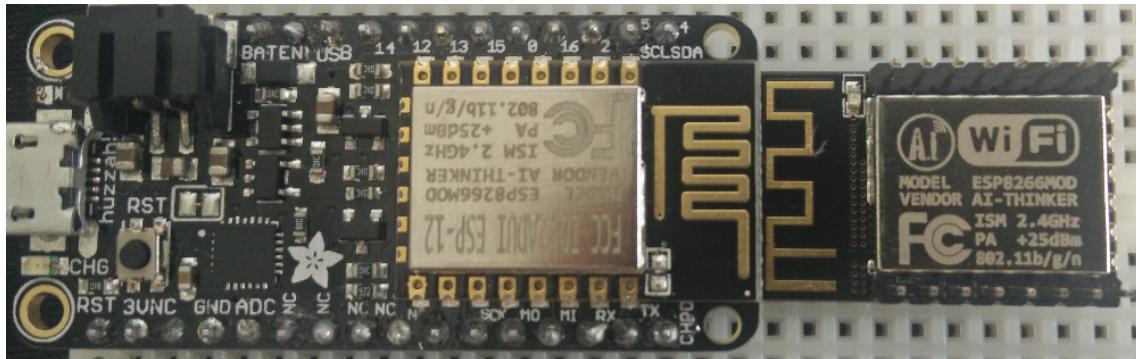


Abbildung 4.2: Vergleich der Antennen, links: ESP12-S verbaut auf einem Adafruit Feather Huzzah, rechts: ESP12-F

Tabelle 4.1: Energieverbrauch des ESP8266 bei verschiedenen Operationen, aus [Espr17]

Parameters	Min	Typical	Max	Unit
Tx802.11b, CCK 11Mbps, P OUT=+17dBm	-	170	-	mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm	-	140	-	mA
Tx 802.11n, MCS7, P OUT =+13dBm	-	120	-	mA
Rx 802.11b, 1024 bytes packet length , -80dBm	-	50	-	mA
Rx 802.11g, 1024 bytes packet length, -70dBm	-	56	-	mA
Rx 802.11n, 1024 bytes packet length, -65dBm	-	56	-	mA
Modem-sleep <sup>①</sup>	-	15	-	mA
Light-sleep <sup>②</sup>	-	0.9	-	mA
Deep-sleep <sup>③</sup>	-	20	-	µA
Power Off	-	0.5	-	µA

### 4.1.1 ESP8266 Arduino Core

Eine einfache Möglichkeit den ESP8266 zu programmieren stellt die bekannte Arduino IDE dar [BCIM17a].

Unter Windows muss zunächst der Treiber für den *CP2104 USB-to-Serial Chip* installiert werden, auf Linux und Mac entfällt dieser Schritt [Frie17]. Nach der Installation der Arduino IDE muss dort in den Einstellungen unter *Additional Boards Manager URLs* die URL [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) hinzugefügt werden. Nach einem Neustart der IDE kann im *Boards Manager* das Paket für den ESP8266 heruntergeladen werden und anschließend das Board *Adafruit HUZZAH ESP8266* ausgewählt werden.

Ist das Board über USB mit dem Computer verbunden, kann nun eigener Code oder eines der Beispiele aus *Examples for Adafruit HUZZAH ESP8266* mit STRG+U auf den ESP geladen werden. Dieser Vorgang wird im Folgenden als *flashen* bezeichnet. Der ESP8266 Arduino Core wird als Open Source Projekt gepflegt und setzt die ESP Open SDK auf einen, für die Arduino IDE üblichen Stil um [BCIM<sup>+</sup>17b]. Dabei wird möglichst die Kompatibilität zu Arduino gewahrt, das führt oft dazu, dass bereitgestellte Funktionen der SDK nicht in Arduino umgesetzt wurden. Sollten die Funktionen dennoch benötigt werden, können die Header-Dateien der SDK direkt importiert werden:

```
//Hier wird ein Header des ESP8266 Arduino Core importiert
#include <ESP8266WiFi.h>

//Hier wird ein Header der ESP Open SDK importiert
extern "C" {#include "user_interface.h"}
```

Der ESP8266 Arduino Core wurde in der Version 2.3.0 verwendet.

### 4.1.2 ESP Open SDK

Statt mit der Arduino Core Umsetzung der ESP Open SDK kann natürlich auch direkt mit ihr programmiert werden [ESP 17].

Dazu muss zunächst das Github Projekt geklont werden: `git clone -recursive https://github.com/pfalcon/esp-open-sdk.git` und anschließend mit `make` kompiliert werden. Wurde die Kompilierung erfolgreich abgeschlossen, sollte die Pfadvariable (PATH) entsprechend der Meldung des make-Tools erweitert werden.

Die in C geschriebenen Programme können nun mit einem modifizierten GCC-Kompiler kompiliert und anschließend mit dem `esptool.py` zunächst in ein Image umgewandelt und dann geflasht werden. Das in *examples* enthaltene *blinky* Beispiel beinhaltet neben dem Beispielcode eine Makefile, in der diese Schritte nachvollzogen werden können, mit `make flash` wird das Beispiel kompiliert und *gefleucht*.

Programme werden in regulärem C unter Zuhilfenahme der in `/sdk/include` enthaltenen Header-Dateien geschrieben, zu beachten ist nur, dass einige Funktionen der `stdlib.h` nicht verwendet werden können. Dies betrifft vor allem direkten Speicherzugriff wie zum Beispiel `memcpy`, hier muss stattdessen `os_memcpy` verwendet werden. Alle betroffenen Funktionen sind in `osapi.h` beschrieben.

Einige frühe Experimente zeigten, dass Programme, die mit der ESP Open SDK geschrieben wurden auf dem ESP8266 schneller starten als solche, die mit dem

ESP8266 Arduino Core geschrieben wurden. Diese Andeutung von Ineffizienzen bei der Übersetzung von Arduino Code wurde zum Anlass genommen, für die nachfolgenden Implementierungen nach der Fertigstellung des Prototypen in Arduino ebenfalls eine Implementierung in C hinzuzufügen, um ein optimales Programm zu erhalten.

Die ESP Open SDK wurde in der Version 2.0.0 verwendet.

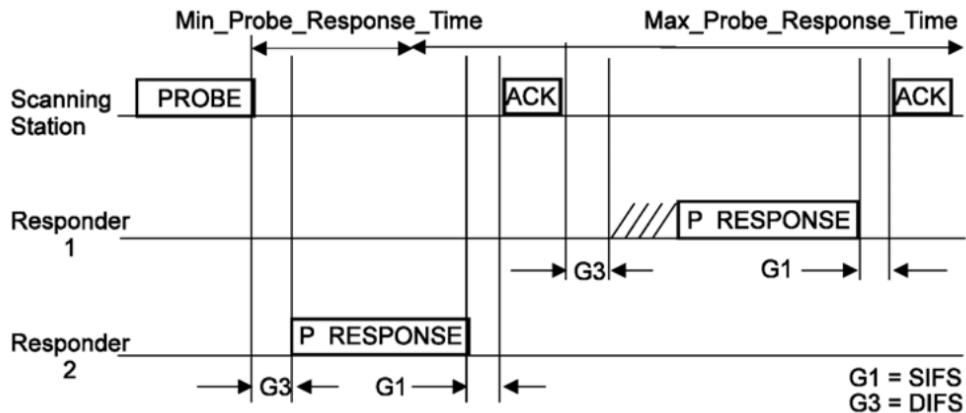


Abbildung 4.3: Ablaufdiagramm eines aktiven Scan, aus [IEEE12b].

## 4.2 Reichweite von 802.11

Um die Intervallzeit für die Lokalisierung festzulegen, muss die Reichweite von 802.11 im Tunnel bestimmt werden. Für die Tests wurde ein *LN-862* Access Point von der Firma Lancom zur Verfügung gestellt. Dieser wurde am hinteren Ende einer Tunnelbohrmaschine in der Tunnelbaustelle Rastatt montiert.

Der Durchmesser des Bahntunnels beträgt 9 Meter, das Ende der Tunnelbohrmaschine befand zum Zeitpunkt der Messungen circa 2 Kilometer weit im Tunnel.

Der AP konnte aufgrund des geringen Platzangebots und den wenigen zur Verfügung stehenden Steckdosen nicht frei platziert werden. Er wurde deshalb unter der ersten stählernen Treppe platziert, diese beeinträchtigt natürlich das Signal. Da es aber üblich ist, IT-Gerätschaften, wie die derzeit verwendeten Bluetooth-Basisstationen, in Metallboxen zu verstauen um sie vor äußeren Einflüssen zu schützen, ist eine gewisse Abschirmung durchaus realitätsnah. Die Platzierung des AP ist auf Abbildung 4.4 eingezeichnet.

Abbildung 4.5 zeigt den *LN-862* hinter der Treppe.

### 4.2.1 Methodik

Die Reichweite wurde in zwei Richtungen geprüft. Zum einen in Richtung des bereits fertig gebohrten Tunnels, hier blockiert nur wenig Stahl das Signal. Lediglich die Treppe, unter der der AP montiert wurde, stellt ein Hindernis dar. Zum anderen wurde die Reichweite in Richtung des Vortriebs geprüft. Dabei stellen eine stählerne Zwischendecke und große Container Hindernisse dar. Die zwei Messtrecken werden in Abbildung fig:rangewlan skizziert.

Außerdem wird die Abschirmung durch ein Gehäuse getestet, dazu wurde eine stabile Plastikbox verwendet.



Abbildung 4.4: Ende der Tunnelbohrmaschine, Pfeil markiert Platzierung des Access Point hinter der Treppe.

Für die Messung wurde der Körper zwischen mobile Einheit und Basisstation gebracht und eine mobile Einheit wurde dann als "außer Reichweite" angesehen, wenn versendete Pakete der mobilen Einheit nicht mehr bei der Basisstation ankamen. In jedem Fall war es möglich durch das Entfernen des körperlichen Hindernisses wieder eine Verbindung herzustellen.

Zur Bestimmung der Distanz wurden die *Tübbinge* verwendet, dies sind Schalungselemente im Tunnel. Im Tunnel Raststatt sind diese fortlaufend nummeriert und genau zwei Meter breit, die Messungen sind deshalb ebenfalls in zwei Meter Schritten angegeben.

### 4.2.2 Ergebnisse

Tabelle 4.2 zeigt die Ergebnisse für die zwei verwendeten ESP8266 Module. Es wurde jeweils mit und ohne Gehäuse gemessen und in jede der beiden beschriebenen Richtungen. Wenige Hindernisse bezeichnet dabei die Richtung des bereits fertig gebohrten Tunnels, viele Hindernisse die Richtung des Vortriebs.

### 4.2.3 Bewertung

Das ESP-12F Modul hatte in jedem der vier Testszenarien eine höhere Reichweite als das ESP-12S, es ist daher im weiteren Verlauf zu bevorzugen.

Für Mitarbeiter im Tunnel kann von einer Maximalgeschwindigkeit von 30 km/h ausgegangen werden, diese wird durch Schienen- oder Lastkraftfahrzeuge erreicht. Um die gemessenen 88 Meter bei 30 km/h zu durchqueren benötigte ein Mitarbeiter circa 10,5 Sekunden, bei einem Sendeintervall von zehn Sekunden finden demnach zwei Sendevorgänge beim durchqueren des Einflussbereichs eines APs statt. Da eine zuverlässige Erkennung von Bereichswechseln gefordert wurde, sollte das Sendeintervall jedoch konservativer gesetzt werden, eine Halbierung auf fünf Sekunden ist daher für eine zuverlässige Erkennung sinnvoll.



Abbildung 4.5: *LN-862* im Tunnel, darauf liegt ein *Raspberry Pi Zero W.*

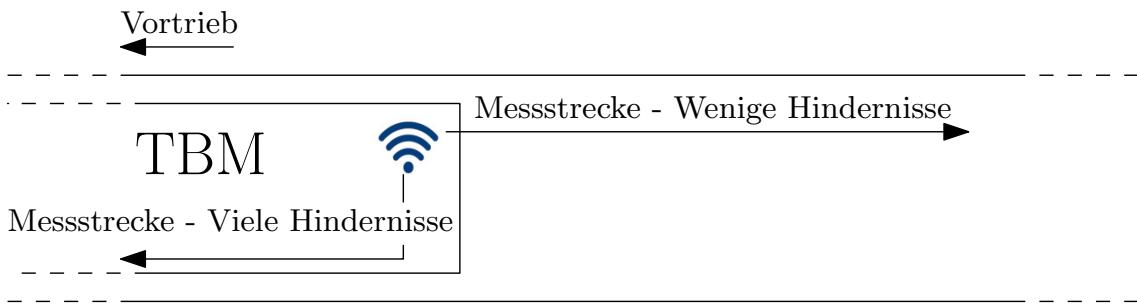


Abbildung 4.6: Messtrecken zur Feststellung der Reichweite von 802.11.

Für die Teststrecke mit vielen Hindernissen wurden geringere Reichweiten gemessen, eine solche Teststrecke findet sich aber nur auf der Tunnelbohrmaschine, welche nur zu Fuß begangen werden kann. Geht man von einer maximalen Bewegungsgeschwindigkeit von 10 km/h für eine laufende Person aus durquerte diese in fünf Sekunden 14 Meter, also deutlich weniger als die gemessenen 32 Meter.

### 4.3 WiFi-LLS-Implementierung

Die mobile Einheit des WiFi-LLS Systems führt alle 5 Sekunden einen *Scan* aus, kodiert die Ergebnisse in XML und versendet sie an den Ortungsserver [ChLu07]. Da der Fokus dieser Arbeit auf dem Energieverbrauch liegt, werden für die Referenzimplementierung einer mobilen WiFi-LLS-Einheit die Kodierung in XML durch eine simple String Kodierung ersetzt und die Ergebnisse werden über UDP an den Ortungsserver übermittelt. Damit wird der Overhead einer TCP-Verbbindung vermieden.

Zunächst muss das Tag dem Netzwerk beitreten (*Join*), einen *Scan* ausführen und ein UDP-Paket versenden. Der Ablauf eines *Scans* wird in Abbildung 4.3 gezeigt, dieser Vorgang muss für jeden nicht überlappenden Kanal durchgeführt werden.

Verwendetes Modul	Aufbau	Strecke	Maximale Einheiten-Sendereichweite
ESP-12S	Offen	Wenige Hindernisse	84m
ESP-12S	In Gehäuse	Wenige Hindernisse	74m
ESP-12S	Offen	Viele Hindernisse	26m
ESP-12S	In Gehäuse	Viele Hindernisse	30m
ESP-12F	Offen	Wenige Hindernisse	88m
ESP-12F	In Gehäuse	Wenige Hindernisse	88m
ESP-12F	Offen	Viele Hindernisse	32m
ESP-12F	In Gehäuse	Viele Hindernisse	32m

Da die *Scan*-Funktion des ESP8266 Arduino Core es nicht erlaubt den RSSI zu einem AP auszulesen muss `user_interface.h` importiert und die *Scan*-Funktion der SDK direkt verwendet werden. Um den Energieverbrauch weiter zu reduzieren, soll der ESP möglichst viel Zeit in Energiesparzuständen verbringen. Der tiefste Schlafzustand, der dennoch eine Aufrechterhaltung der WLAN-Verbindung erlaubt ist der `light_sleep`. Er wird vom ESP8266 automatisch aufgerufen, wenn er keine Aufgaben zu erledigen hat. Er kann aber auch manuell aufgerufen werden, beides wurde getestet.

Einige Parameter können gewählt werden. Die Intervallzeit bestimmt, wie oft der ESP aktiv ist und damit auch wie viel Energie er verbraucht. Die Intervallzeit wird entsprechend der Untersuchung der Reichweite auf fünf Sekunden gesetzt. Des Weiteren kann theoretisch die Zahl der *gescannten* Kanäle gewählt werden. Da sich die Einflussbereiche der APs in einem WLAN-Netzwerk üblicherweise überlappen, ist es aber sinnvoll diese über die vier überlappungsfreien Kanäle zu verteilen. Eine Implementierung, die nur einen Kanal *scanns* tritt deshalb nur außer Konkurrenz an. Im Folgenden findet eine Voruntersuchung des Energieverbrauchs statt. Tabelle 4.3 zeigt den gemessenen Energieverbrauch der Implementierungen in Arduino und C, jeweils mit und ohne manuell aufgerufenen `light_sleep` und eine Implementierung, die nur einen Channel *scanns*. Für die Tests wurde das Adafruit Feather Huzzah ESP8266 verwendet, es wurde mit 5 Volt aus einer USB-Powerbank mit Energie versorgt, der Verbrauch wurde mit einem dazwischen geschalteten TM103 USB-Power-Meter der Marke Muker gemessen. Jeder Versuch wurde mindestens eine Stunde durchgeführt. Wurde in dieser Zeit der Verbrauchswert von 10 mAh nicht überschritten, wurde der Versuch verlängert, da der TM103 den Verbrauch ohne Nachkommastellen anzeigt.

Die Werte wurden stationär in einer Mietwohnung in einem fünfstöckigen Wohnhaus und damit nicht unter realen Bedingungen aufgezeichnet. Unter realen Bedingungen finden durch die Bewegung des Mitarbeiters regelmäßig Reassizierungen statt, im Gegenzug liefert ein *Scan* in einem Tunnel weniger Ergebnisse als in einem Wohnhaus.

Die Tests zeigen, dass die Programmierung mit der ESP Open SDK einen Vorteil beim Energieverbrauch hat, dieser liegt bei ca 10%. Andererseits fällt auf, dass die Reduzierung der *gescannten* Kanäle eine signifikante Senkung des Energieverbrauchs nach sich zieht, die *Scan*-Funktion ist also der Hauptverbraucher.

Die Implementierung in C mit manuellem `light_sleep` werden in Abschnitt 4.5.1 genauer untersucht.

Tabelle 4.3: Energieverbrauch WiFi-LLS-artiger Tags

SDK	manueller light_sleep	Kanäle	Versuchs- dauer in Stunden	Gesamt- verbrauch in mAh	$\varnothing$ Ver- brauch in mA
Arduino Core	Nein	alle	1	22	22
Arduino Core	Ja	alle	1	21	21
ESP Open SDK	Nein	alle	1	19	19
ESP Open SDK	Ja	alle	1	19	19
ESP Open SDK	Ja	einer	2	13	6,5

## 4.4 Anpassungen für Bereichsortung

Bei WiFi-LLS wird der *Scan* durchgeführt, um den RSSI zu nahen Access Points zu erhalten und dann auf dem Ortungsserver die Position der mobilen Einheit mit einer Trilateration zu berechnen. Im Tunnel sind oft nur ein bis zwei APs in Reichweite, außerdem wird eine Bereichsortung als ausreichend angesehen.

Werden die APs geschickt den Bereichen zugeordnet, reicht das Wissen um einen nahen AP, um die mobile Einheit einem Bereich zuzuordnen. Da der mobilen Einheit die MAC-Adresse seines Netzzugangs bekannt sein muss, kann dies als Ortungsinformation verwendet werden. Der Wechsel zwischen zwei Access Points wird exemplarisch in Abbildung 4.7 dargestellt.

Die MAC-Adresse des Access Points wird nun zusammen mit der eigenen MAC-Adresse als Identifikator als String kodiert und per UDP an den Ortungsserver versendet.

Tabelle 4.4 zeigt den gemessenen Verbrauch der Implementierungen in Arduino und C, jeweils mit und ohne manuell aufgerufenen `light_sleep`.

Tabelle 4.4: Energieverbrauch der Bereichsortungstags

SDK	manueller light_sleep	Versuchs- dauer in Stunden	Gesamt- verbrauch in mAh	$\varnothing$ Verbrauch in mA
Arduino Core	Nein	1	14	14
Arduino Core	Ja	3	21	7
ESP Open SDK	Nein	2	12	6
ESP Open SDK	Ja	2	11	5,5

Der Verbrauch liegt wie erwartet unter dem der WiFi-LLS-Implementierung, sogar unter der Implementierung, die nur einen Kanal scannt. Als weitere Optimierung könnte eine mobile Einheit nur dann senden, wenn eine Reassoziation stattgefunden hat. Die mangelnde Transportsicherheit von UDP macht dieses Vorgehen jedoch risikant, wenn das Paket verloren geht wird kein Bereichswechsel erkannt. Um wieder eine begrenzte Transportsicherheit zu erhalten, kann entweder das UDP-Paket mehrfach versendet werden; ohne Reassoziation in einen festen, aber größeren Intervall gesendet werden oder statt eine UDP-Verbindung eine TCP-Verbindung verwendet werden. Somit ergeben sich neue Testszenarien: Ohne zusätzliche Sicherung, UDP-Paket mehrfach (dreifach) versenden, zusätzliches (30 beziehungsweise 60 Sekunden) Sendeintervall, TCP-Verbindung (offen halten oder nach dem Senden schließen).

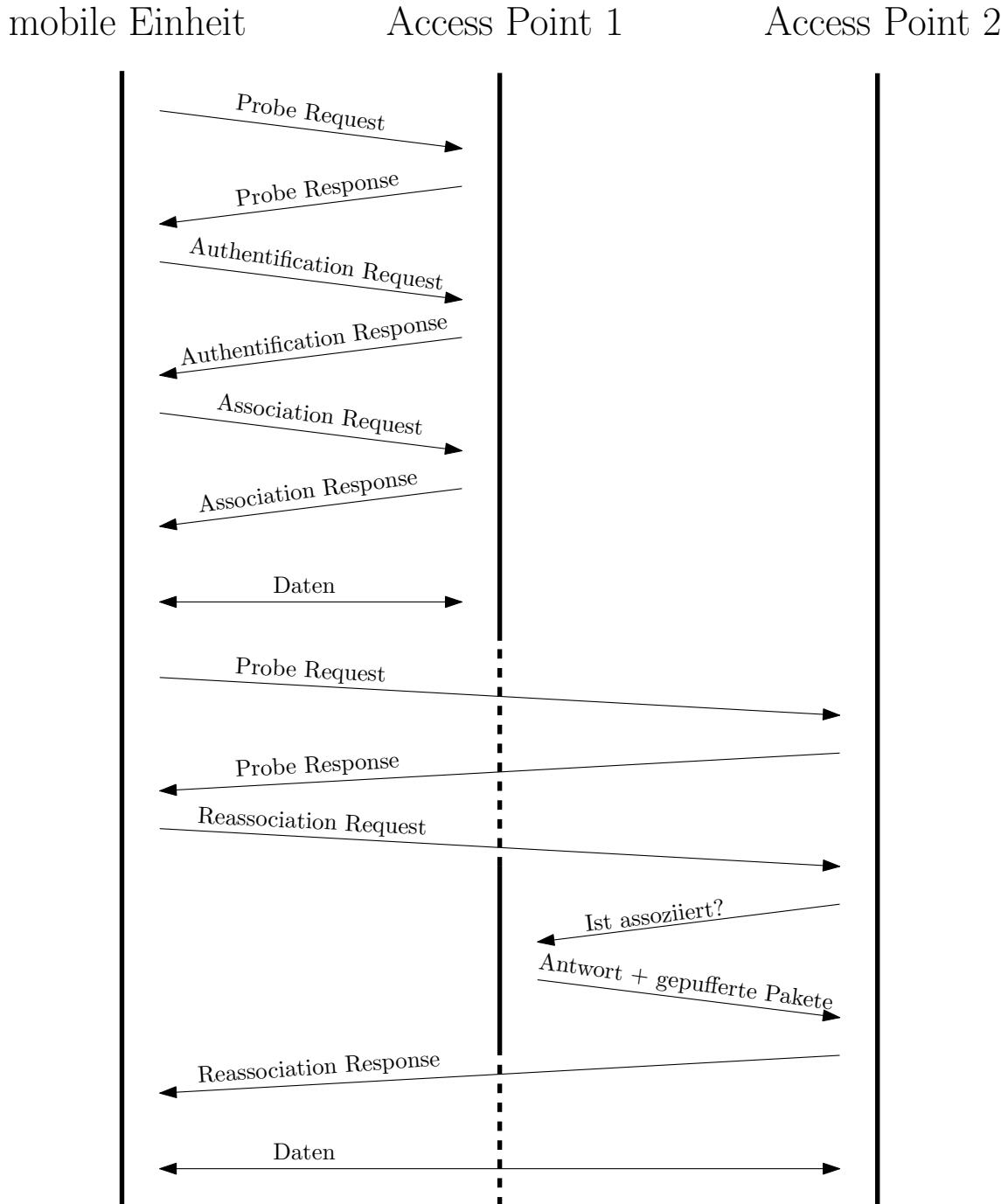


Abbildung 4.7: Vorgang von Assoziation und Reassoziation.

Da der Verbrauch nun stark von der Anzahl der Reassoziationen abhängt sind im gegebenen, stationären Testszenario keine aussagekräftigen Ergebnisse möglich. Dennoch sollen die Tests einen Ausgangswert ermitteln, dieser kann als untere Grenze für den Verbrauch einer Implementierung angesehen werden. Da in den vorherigen Tests die Implementierungen mit der ESP Open SDK verbrauchsärmer waren, wurden alle in Tabelle 4.5 gezeigten Implementierungen mit ihr erstellt, der manuelle `light_sleep` ist immer aktiv.

Der Energieverbrauch sinkt durch das Einsparen von Sendevorgänge deutlich. In Abschnitt 4.5.2 werden die Implementierungen, die eine TCP Verbindung nutzen

Tabelle 4.5: Energieverbrauch der verbesserten Bereichsortungstags

Transportsicherung	Versuchs-dauer in Stunden	Gesamt-verbrauch in mAh	$\varnothing$ Verbrauch in mA
Ohne	5	10	2
Dreifach UDP	10	14	1,4
Zusatzintervall 30s	13	34	2,62
Zusatzintervall 60s	5	8	1,6
TCP (halten)	10	12	1,2
TCP (schließen)	11	12	1,09

genauer untersucht. Diese bieten eine gute Übertragungssicherheit bei geringem Verbrauch.

## 4.5 Untersuchung des Energieverbrauchs

Der Muker TM103 USB-Power-Meter bietet sowohl im Zeit- als auch im Wertebereich nur eine sehr geringe Auflösung. Stattdessen soll der Verbrauch mit einem INA219 Chip genauer untersucht werden. Er kann den Verbrauch mit bis zu 333Hz bestimmen und besitzt dabei eine Messgenauigkeit von 99,5% [Texa15]. Der Stromverbrauch wird über den Spannungsabfall über einen  $0,1\Omega$  Widerstand bestimmt, der verwendete Widerstand besitzt eine Fertigungstoleranz von 1%. Die Messgenauigkeit sinkt deshalb auf  $99,5\% * 99\% = 98,505\%$ . Das über  $I^2C$  auslesbare Register löst den aktuellen Verbrauch in 0,1 Milliamper Schritten auf, Verbräuche darunter können nicht bestimmt werden.

Außerdem wird dieses mal durch den JST-Anschluss für den Akku gemessen, dadurch können eventuelle Ineffizienzen des Lithium-Polymer-Ladeschaltkreises aufgezeichnet werden. Jede Messung wurde über eine Stunde durchgeführt. Abbildung 4.8 zeigt den INA219 integriert auf einer Platine, diese wird für die Messungen verwendet.

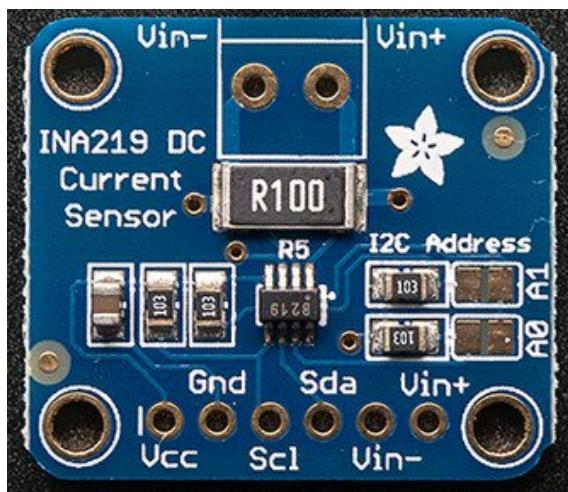


Abbildung 4.8: INA219, die mit R100 beschriftete Komponente über dem INA219 ist der Messwiderstand. Bild von Adafruit Industries.

### 4.5.1 WiFi-LLS

Abbildung 4.9 zeigt den Lastverlauf nach Anschalten der mobilen Einheit für die Implementierung von WiFi-LLS, wenn ein AP zur Verfügung steht.

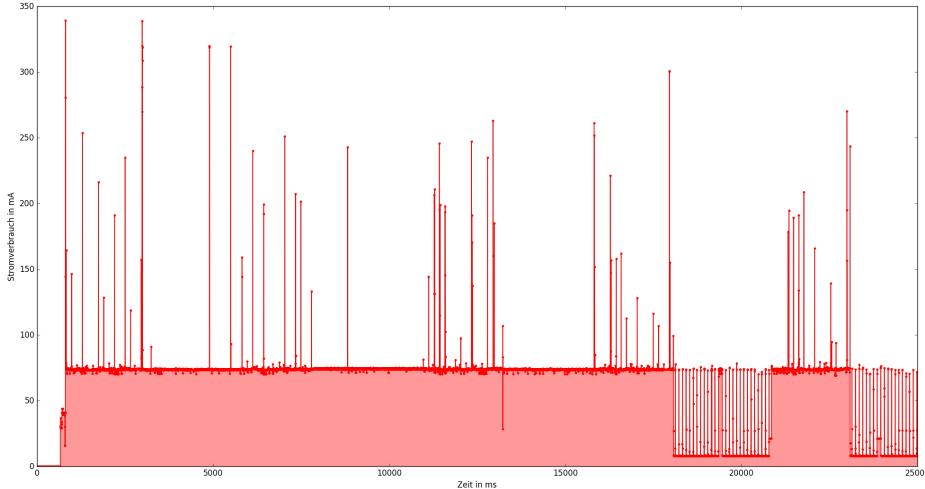


Abbildung 4.9: Lastkurve einer Implementierung von WiFi-LLS.

Diese beginnt bei circa einer Sekunde mit dem *Scan*, nachdem sie diesen beendet hat beginnt sie bei 5 Sekunden mit dem *Join*-Vorgang.

Der ESP8266 empfängt während des *Join* die gesamte Zeit und verbraucht dabei circa 70 Milliamper, nachdem dieser jedoch abgeschlossen ist synchronisiert er sich mit dem AP und lauscht alle 100ms auf den *Beacon* des AP. Ist im *Beacon* eine Aufforderung zum Empfangen für den ESP8266 enthalten empfängt er länger um die Nachricht zu erhalten, ein solches Verhalten ist bei circa 19 Sekunden zu erkennen. Bei 12, 17 und 22 Sekunden werden weitere *Scans* ausgeführt, diese stehen im Zentrum der Implementierung, da sie implizit die Position bestimmen. Die rote Kurve in Abbildung 4.10 zeigt diesen Vorgang genauer, ein *Scan* besteht für jeden gescannten Kanal aus dem Versenden eines *Probe Request* und anschließenden Empfangen der *Probe Responses*. Abschließend wird ein Paket an den Ortungsserver versendet und der Chip wechselt wieder in einen Zustand, in dem er periodisch die *Beacons* des AP empfängt.

Auffällig ist, dass der Chip circa 8,2 Milliamper verbraucht wenn er nicht empfängt, dagegen gibt das Datenblatt des ESP8266 nur einen Verbrauch von 0,9 Milliamper an. Das Experiment wurde deshalb mit einem ESP-12F Modul wiederholt, welches nicht auf einem ESP8266 Feather verbaut war. Dabei zeigte sich, dass dieses in der selben Situation nur 1,2 Milliamper verbraucht. Die Lastkurve des einzelnen ESP-12F Modul ist in Abbildung 4.10 grün dargestellt. Daraus lässt sich schließen, dass die zusätzlichen Komponenten auf dem ESP8266 Feather 7 Milliamper Verbrauch erzeugen, dies ist im Zuge der Laufzeitoptimierung nicht tragbar.

Zusätzlich wurde die Implementierung von WiFi-LLS mit nur einem *gescannten* Kanal geprüft. Abbildung 4.11 zeigt den verkürzten Ortungsvorgang. Da nur ein Kanal *gescannt* wird, wird nur ein *Probe Request* versendet. Nach Empfangen der Antworten wird ein Paket an den Ortungsserver gesendet und der ESP8266 wechselt wieder

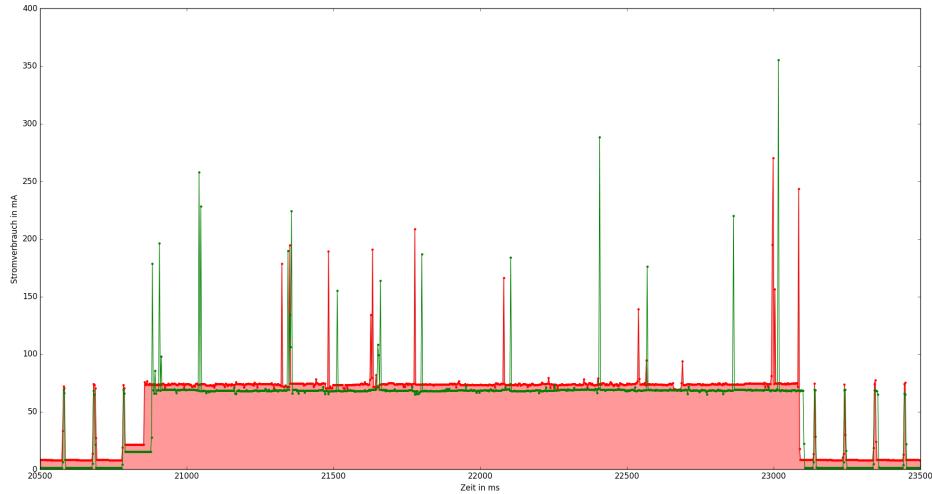


Abbildung 4.10: Lastkurve eines Ortungsvorgangs mit WiFi-LLS.

in den Zustand des periodischen Empfangens.

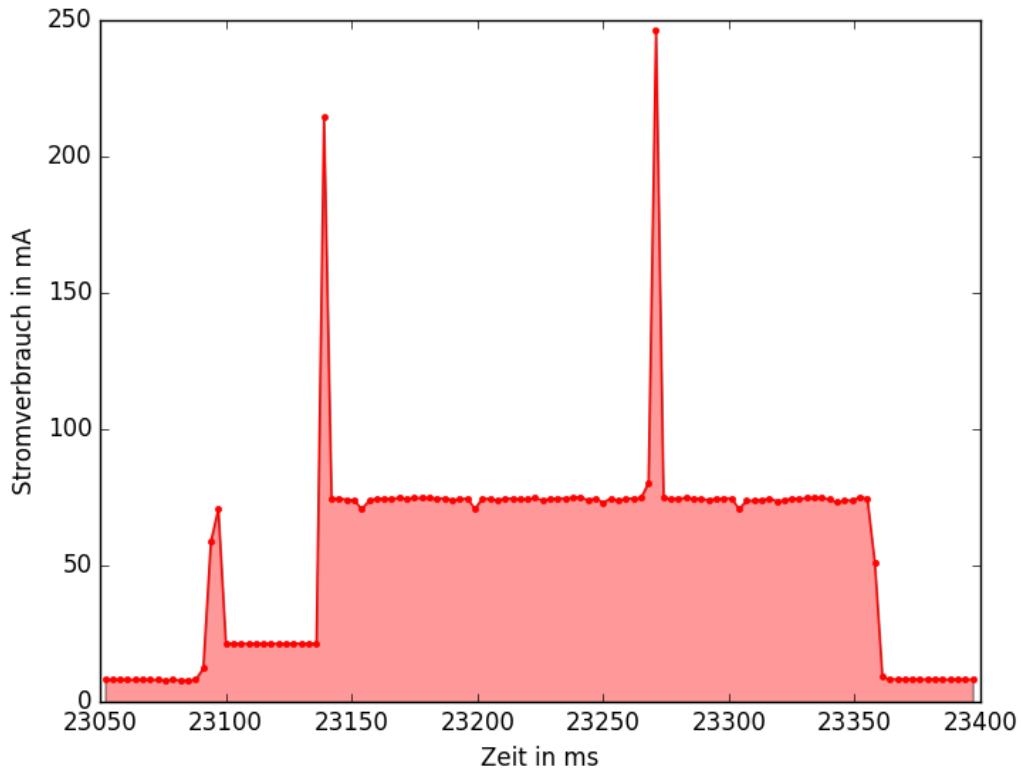


Abbildung 4.11: Lastkurve des Ortungsvorgangs mit einem gescannten Kanal.

Tabelle 4.6 listet den durchschnittlichen Verbrauch der Einheiten über eine Stunde. Die Stromversorgung der Einheiten wurden circa eine Sekunde nach Beginn des Experiments angeschaltet, anschließend treten keine Veränderungen mehr auf.

Tabelle 4.6: Energieverbrauch mobiler Einheiten mit WiFi-LLS Implementierung

Hardware	Programm	$\varnothing$ Verbrauch in mA	Laufzeit in Stunden
ESP8266 Feather	WiFi-LLS alle Kanäle	42,2	33,2
ESP-12F	WiFi-LLS alle Kanäle	36,5	38,3
ESP8266 Feather	WiFi-LLS ein Kanal	18,7	74,9
ESP-12F	WiFi-LLS ein Kanal	11,4	122,8

Um die finale Laufzeit zu bestimmen muss die Kapazität des Akkus bekannt sein. Ein 1400mAh Lithium-Polymer-Akku wurde als leicht genug angesehen, um um den Hals getragen werden zu können. Die Laufzeiten in Tabelle 4.6 wurden für diesen 1400mAh Akku mit  $1400 \text{ mAh} / X \text{ mA} = Y \text{ h}$  bestimmt.

#### 4.5.2 Indirekte Bereichsortung

Abbildung 4.12 zeigt den Lastverlauf nach Anschalten der mobilen Einheit für die Implementierung für Bereichsortung mit aufrecht erhaltener TCP-Verbindung, wenn ein AP zur Verfügung steht. Der Beginn des Lastverlaufs ist dem der WiFi-LLS-Implementierung ähnlich, es werden ebenfalls *Scan* und *Join* durchgeführt. Da jedoch nur für das Event einer vollständigen (Re-)Assoziation gesendet wird bleibt der ESP8266 anschließend im Zustand des periodischen Empfangens von *Beacons*. Diese wird nur von den *Keep-Alive-Paketen* unterbrochen, welche alle 30 Minuten versendet werden.

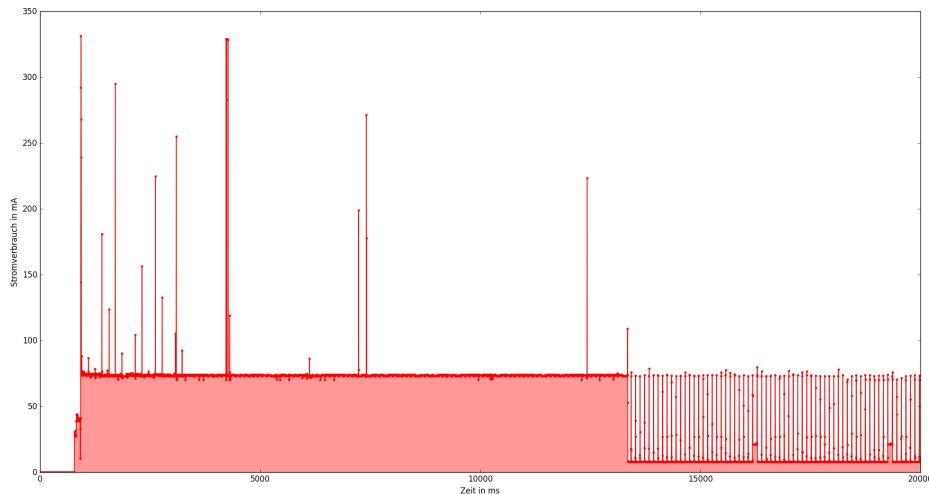


Abbildung 4.12: Lastkurve einer Implementierung indirekter Bereichsortung, welche eine TCP Verbindung offen hält.

Für die Implementierung mit anschließendem Abbau der TCP-Verbindung kommen zusätzliche Pakete direkt nach dem Versenden der Assoziationsinformation an den Ortungsserver hinzu, dafür entfallen die *Keep-Alive-Pakete*. Abbildung 4.13 zeigt dieses Verhalten.

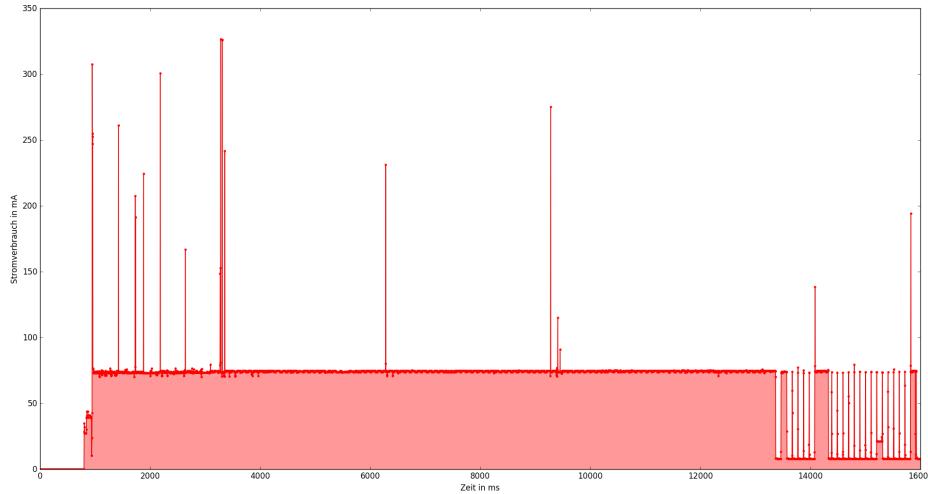


Abbildung 4.13: Lastkurve einer Implementierung indirekter Bereichsortung, welche eine TCP Verbindung nach dem Senden abbaut.

Auch diese Implementierungen wurden sowohl mit einem ESP8266 Feather, als auch mit einem einzelnen ESP-12F Modul getestet, die Ergebnisse sind in Tabelle 4.7 zu finden.

Tabelle 4.7: Energieverbrauch mobiler Einheiten mit Bereichsortung

Hardware	Programm			$\varnothing$ Verbrauch in mA	Laufzeit in Stunden
ESP8266 Feather	Bereichsortung	TCP	Verbindungs halten	15,5	90,3
ESP-12F	Bereichsortung	TCP	Verbindungs halten	8,8	159,1
ESP8266 Feather	Bereichsortung	TCP	Verbindungs schließen	15,4	90,9
ESP-12F	Bereichsortung	TCP	Verbindungs schließen	8,8	159,1

#### 4.5.3 Kein AP in Reichweite

Ergänzen

# 5. Direkte Fernlokalisierung mit 802.11

Bei der direkten Fernlokalisierung wird die Messgröße auf den Basisstationen gemessen. Die Ortungsinformation wird dann implizit durch versenden der gemessenen Werte an den Ortungsserver übermittelt. Dieser berechnet dann aus den gesammelten Werten die Position der mobilen Einheit. Eine Lösung zur direkten Fernlokalisierung muss deshalb nicht mit einem Access Point assoziiert sein. APs versenden die gemessenen Werte jedoch nicht automatisch an den Ortungsserver. Sollen also die APs des WLAN-Netzwerks auch als Basisstationen verwendet werden, muss ihre Software es erlauben den in der Analyse als Messgröße ausgewählten *Received Signal Strength Indicator* (RSSI) bestimmter Pakete abzurufen.

## 5.1 RADAR-Implementierung

Eine mobile Einheit mit *RADAR* versendet alle 0,25 Sekunden ein 6 Byte langes UDP-Paket, der RSSI der Übertragung wird dann auf dem AP gemessen. Das Sendintervall wurde so kurz gewählt, um spontane Schwankungen im RSSI durch mehrfache Messung zu glätten und sich bewegende Personen möglichst genau zu erfassen. Für eine Bereichsortung reicht ein wesentlich längeres Sendeintervall, es wird erneut ein Intervall von fünf Sekunden gewählt, in dem sich ein Mitarbeiter maximal 42 Meter bewegt. Tabelle 5.1 zeigt den mit dem TM103 gemessenen Verbrauch der Implementierung für mobile Einheiten mit RADAR jeweils in Arduino und C, mit unterschiedlich langen Sendeintervallen mit und ohne manuellen `light_sleep`.

Der Energieverbrauch einer Lösung die vier Pakete pro Sekunde sendet ist wie erwartet hoch. Die Implementierungen mit der ESP Open SDK und manuellem `light_sleep` unterscheiden sich ausschließlich im Sendeintervall, dies verändert den Energieverbrauch jedoch stark. Die in Abschnitt 4.4 besprochenen Optimierungen (nur bei AP-Wechsel senden) können auch für die *RADAR*-Implementierung verwendet werden, das *RADAR*-artige mobile Einheit ist dann aber in seiner Implementierung bis auf den Inhalt des UDP-Pakets identisch mit dem der mobilen Einheiten für Bereichsortung. Der Energieverbrauch sollte sich somit kaum unterscheiden und ein

Tabelle 5.1: Energieverbrauch RADAR-artiger mobiler Einheiten

SDK	manueller light_sleep	Senden- intervall in s	Versuchs- dauer in Stunden	Gesamt- verbrauch in mAh	$\varnothing$ Ver- brauch in mA
Arduino Core	Nein	0,25	2	80	40
Arduino Core	Ja	0,25	3	119	39,66
Arduino Core	Nein	5	3	28	9,33
Arduino Core	Ja	5	3	23	7,66
ESP Open SDK	Nein	0,25	1	40	40
ESP Open SDK	Ja	0,25	1	38	38
ESP Open SDK	Nein	5	2	12	6
ESP Open SDK	Ja	5	2	10	5

System mit *RADAR*-artigen mobilen Einheiten benötigt Veränderungen der Software der APs, ein System mit mobilen Einheiten für Bereichsortung ist deshalb vorzuziehen. Die Implementierung in C mit einem Sendeintervall von fünf Sekunden wird in Abschnitt 5.3.1 genauer untersucht.

## 5.2 Anpassungen für Bereichsortung

*RADAR* versendet immer noch UDP-Pakete und arbeitet damit auf Schicht vier (Transport) des OSI-Modells und muss im Netzwerk authentifiziert und mit einem Access Point assoziiert sein. Das ist für eine direkte Fernlokalisierung aber nicht notwendig, der RSSI wird auf Schicht eins (PHY) gemessen. Grundsätzlich kann aufgrund der möglichen Änderungen am AP ein beliebiges Paket mit einer speziellen Kennung versendet und vom AP als Positionsmitteilung der mobilen Einheit erkannt werden.

Ein Sendevorgang, der nur Schicht eins nutzt hat einen geringeren Energieverbrauch, da er nur senden und nie empfangen muss. Ein solcher Sendevorgang könnte aber die Funktion des Netzwerks beeinträchtigen und stellt nicht sicher, dass die eigene Übertragung nicht durch andere Übertragungen gestört wurde. Außerdem ist Schicht eins oft nicht vom Entwickler zugreifbar, es sollte deshalb nicht auf Schicht eins gearbeitet werden.

Stattdessen sollte Schicht zwei (MAC) des OSI-Modells verwendet werden. Da 802.11 für den Mediumszugriff eine Kollisionsvermeidung (CSMA/CA) verwendet wird, muss die mobile Einheit vor dem Senden das Medium belauschen, um zu bestimmen ob es belegt ist. Der Energieverbrauch ist somit pro Sendevorgang höher als bei einer Lösung auf Schicht eins, stellt dafür aber die Verfügbarkeit des Mediums (der Frequenz) für die übrigen Teilnehmer sicher.

Um die Änderungen an der Software des AP gering zu halten wurde der *Probe Request* als zu sendender Frame gewählt. Es handelt sich dabei um einen Management Frame (siehe Tabelle 2.1) der für den, in Abschnitt 2.2.1 beschriebenen, *Scan*-Vorgang verwendet wird. Der *Probe Request* hat dabei den Vorteil, dass er bereits vom AP verarbeitet und mit einer *Probe Response* beantwortet wird.

Es wird also lediglich gefordert, dass der AP den Empfang des *Probe Request* im Zuge der Verarbeitung protokolliert. Im Einzelnen müssen die Empfangszeit, der RSSI und die MAC-Adresse des Absenders protokolliert und für den Ortungsserver abrufbar gemacht werden. Manche kommerzielle APs bieten ein solches Protokoll für *Probe*

*Requests* und *Beacons* im Zuge einer *Rogue Client/AP Detection* an [LANC17]. Abbildung 5.1 zeigt den Vorgang schematisch.

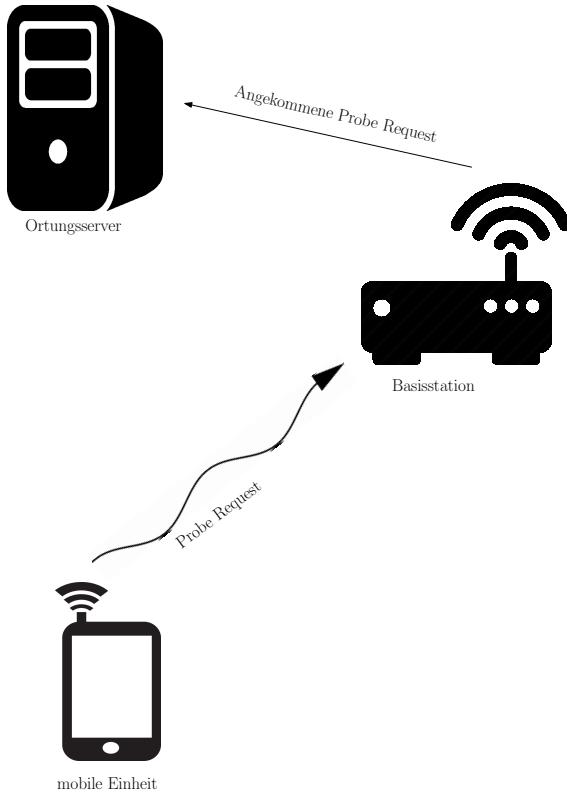


Abbildung 5.1: Schema der Bereichsortung mit Probe Request.

Die ESP Open SDK bietet über die Operationen wie *Scan* und *Join* hinaus mit `wifi_send_pkt_freedom` eine Funktion zum Senden von Paketen auf Schicht zwei an. Der ESP8266 Arduino Core implementiert diese Funktion nicht, stattdessen muss sie mit `extern "C" {#include "user_interface.h"}` importiert werden. `wifi_send_pkt_freedom` setzt den PHY-Header selbst, der MAC-Header und Inhalt des Paketes müssen über einen Puffer übergeben werden.

```
uint8_t packet[26] = {
/*0*/ 0x40, //Version (2bit), Type (2bit), Subtype(4bit)
/*1*/ 0x00, //Flags
/*2*/ 0x00, 0x00, //Duration
/*4*/ 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, //Destination MAC
/*10*/ 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, //Source MAC
/*16*/ 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, //BSSID, all ff=broadcast
/*22*/ 0x00, 0x00, //Sequence Number (12bit), Fragment Number (4bit)
// [End of MAC-Header] [Start of Management tags]
/*24*/ 0x83, //Tag Number (Path Reply 131)
/*25*/ 0x00, //Tag length
};
```

Ein gewöhnlicher *Probe Request* beinhaltet noch zusätzliche Informationen bezüglich seiner technischen Möglichkeiten, wie etwa unterstützte Standards und Datenraten. Da die mobile Einheit aber nicht tatsächlich beitreten will, kann darauf

verzichtet werden.

Da keine Verbindung mehr aufrecht erhalten werden muss, können tiefere Schlafzustände eingenommen werden. Statt des `light_sleep` kann der `deep_sleep` verwendet werden. Dieser schaltet den ESP8266 und seinen Speicher fast vollständig ab, nach Ablauf der angegebenen Schlafzeit wird *Pin 16* mit der Masse verbunden. Damit der ESP wieder aufwacht muss *Pin 16* mit dem *Reset Pin* (RST) verbunden werden, bei einem Reset initialisiert der ESP8266 neu. Bei einer Lösung auf einer höheren Schicht würde dies dazu führen, dass die mobile Einheit versucht dem Netzwerk erneut beizutreten. Hingegen kann bei einer Lösung auf Schicht zwei sofort gesendet und danach wieder geschlafen werden.

Tabelle 5.2 zeigt den Energieverbrauch der Implementierungen jeweils mit manuell herbeigeführtem Schlafzustand, das Sendeintervall liegt bei konstant fünf Sekunden.

Tabelle 5.2: Energieverbrauch mobilen Einheiten mit Probe Request

SDK	manueller Schlafzu- stand	Versuchs- dauer in Stunden	Gesamt- verbrauch in mAh	$\varnothing$ Ver- brauch in mA
Arduino Core	Ohne	2	81	40,5
Arduino Core	<code>light_sleep</code>	2	10	5
Arduino Core	<code>deep_sleep</code>	4	17	4,25
ESP Open SDK	Ohne	5	8,33	8,33
ESP Open SDK	<code>light_sleep</code>	25	75	3
ESP Open SDK	<code>deep_sleep</code>	48	39	0,81

Zu erkennen ist, dass die Verwendung des `deep_sleep` zu einem geringeren Verbrauch führt. Allerdings benötigt die mit dem Arduino Core programmierte mobile Einheit im Vergleich zu der mit der ESP Open SDK programmierten Einheit deutlich länger zum Starten. Sie verbraucht deshalb sogar mehr Energie als die mobilen Einheiten für Bereichsortung aus Abschnitt 4.4, die Implementierung mit der ESP Open SDK verbraucht aber weniger Energie als diese. Hinzu kommt, dass sich der Verbrauch dieser mobilen Einheiten nicht durch die Bewegung des Trägers erhöht, da keine Reassoziationen stattfinden. Um die in Abschnitt 1.4 geforderten Laufzeiten zu erreichen, werden in Abschnitt 5.4.2 weitere Verbesserungen besprochen. Die Implementierung in C mit `deep_sleep` wird in Abschnitt 5.3.2 genauer untersucht.

### 5.2.1 Anzahl der verwendeten Kanäle

Eine Lösung die nicht vor dem Versenden das Spektrum nach den Access Points durchsucht muss entweder auf der Annahme beruhen, dass alle Access Points auf einem Kanal agieren oder in allen in Frage kommenden Kanälen senden.

Je nach Erweiterung der 802.11 Spezifikation ergeben sich unterschiedlich viele solcher Kanäle. 802.11b verwendet eine Kanalbreite von 22MHz, es stehen daher effektiv nur drei Kanäle zur Verfügung: 1, 7 und 13 in Europa beziehungsweise 1, 6 und 11 in Nordamerika. Für 802.11g/n mit 20MHz Kanalbreite sind zwar in Europa theoretisch vier Kanäle verfügbar (1,5,9,13), es werden aber in der Praxis meist dieselben Kanäle wie bei 802.11b verwendet um die Kompatibilität zu 802.11b zu gewährleisten. 802.11n ist auch für eine Kanalbreite von 40 MHz spezifiziert, hier stehen

effektiv nur noch 2 Kanäle zur Verfügung üblicherweise werden Kanal 3 und 11 gewählt.

Die Implementierung in C mit `deep_sleep` wurde sowohl auf einem, als auch auf drei Kanälen getestet. Dabei ergab sich in 24 Stunden kein messbarer Unterschied. Es sollte daher auf eine Festlegung des Kanals verzichtet werden, weil diese die reguläre Funktionsweise des WLAN Netzwerks beeinträchtigen könnte.

## 5.3 Untersuchung des Energieverbrauchs

Erneut soll der Energieverbrauch mit dem INA219 genauer bestimmt werden, der INA219 und die verwendete Methodik werden in Abschnitt 4.5 beschrieben.

### 5.3.1 RADAR

Abbildung 5.2 zeigt den Lastverlauf nach Anschalten der mobilen Einheit für die Implementierung von *RADAR*, wenn ein AP zur Verfügung steht. Der Beginn des Lastverlaufs ist dem der WiFi-LLS Implementierung ähnlich, es werden ebenfalls *Scan* und *Join* durchgeführt.

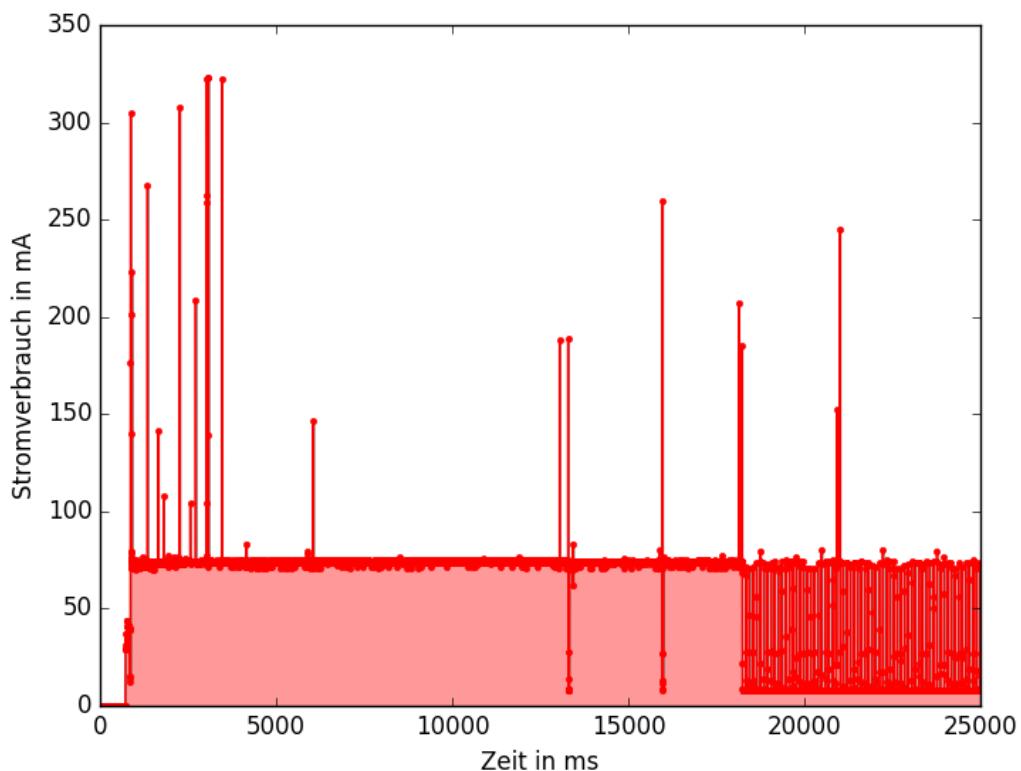


Abbildung 5.2: Lastkurve einer Implementierung von *RADAR*.

Abbildung 5.3 zeigt den Sendevorgang der *RADAR*-Implementierung. Auffällig ist, dass längerfristig empfangen wird, obwohl dies für den Versand des UDP Pakets nicht notwendig ist.

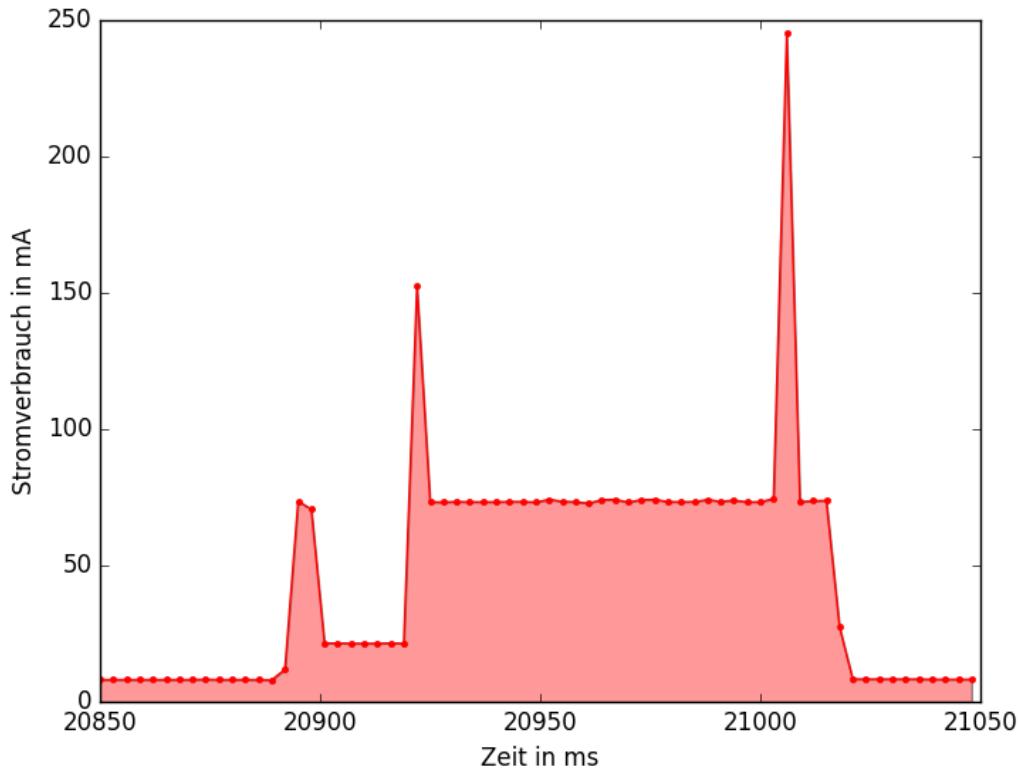


Abbildung 5.3: Lastkurve eines Ortungsvorgangs mit RADAR.

In Tabelle 5.3 ist der durchschnittliche Verbrauch der *RADAR*-Implementierung über eine Stunde gelistet. Es wurde sowohl mit dem ESP8266 Feather, als auch mit dem einzelnen ESP-12F gemessen, die mobile Einheit wurde jeweils erst circa eine Sekunde nach Beginn mit Strom versorgt.

Tabelle 5.3: Energieverbrauch mobiler Einheiten mit *RADAR*-Implementierung

Hardware	Programm	$\varnothing$ Verbrauch in mA	Laufzeit in Stunden
ESP8266 Feather	RADAR	16,7	83,8
ESP-12F	RADAR	10,1	138,6

### 5.3.2 Probe-Request-Lokalisierung

Abbildung 5.4 zeigt den Lastverlauf für einen Sendevorgang der *Probe-Request*-Implementierung, der Verlauf für den ESP8266 Feather ist in Rot und der Verlauf für das ESP-12F Modul ist in Grün dargestellt.

Nachdem der ESP8266 aus dem `deep_sleep` erwacht beginnt eine circa 100ms andauernde Startphase, danach sendet er die drei Probe Requests. Anschließend soll der ESP8266 wieder in den `deep_sleep` versetzt werden, vorher empfängt er jedoch noch 100ms. Die restliche Zeit befindet sich der ESP8266 im Tiefschlaf. Bei dem ESP-12F Modul ist der INA219 nicht in der Lage einen Vebrauch zu messen, er liegt unter 0,1 Milliamper.

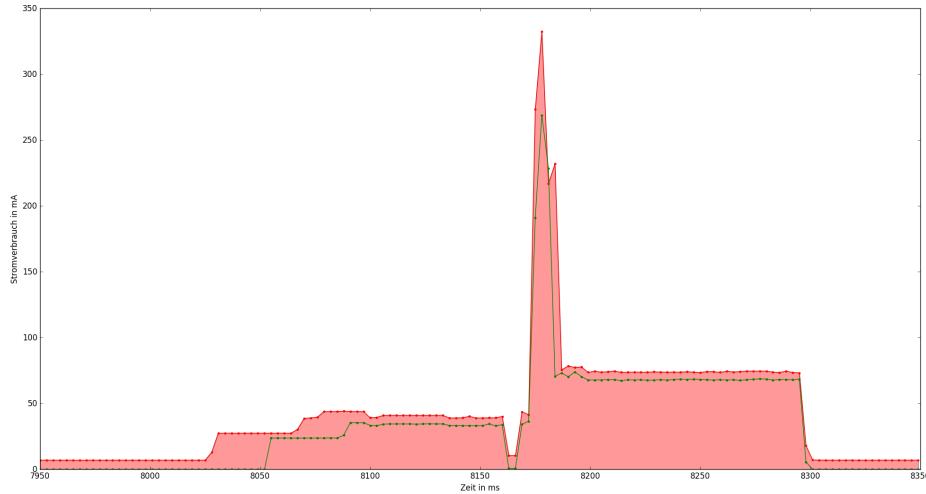


Abbildung 5.4: Lastkurve eines Ortungsvorgangs mit Probe Requests.

Beim ESP8266 Feather misst er jedoch durchgehend einen Verbrauch von über 7 Milliamper, daraus ergeben sich die Unterschiede in den Messungen, die in Tabelle 5.4 dargestellt werden. Es wurde sowohl mit nur einem versendeten Probe Request, als auch mit drei Probe Requests getestet, die Unterschiede im Verbrauch liegen jedoch im Bereich der Messungenauigkeit.

Tabelle 5.4: Energieverbrauch mobiler Einheiten mit Probe Request Ortung

Hardware	Programm	$\varnothing$ Verbrauch in mA	Laufzeit in Stunden
ESP8266 Feather	Probe Request drei Kanäle	9,72	144
ESP-12F	Probe Request drei Kanäle	1,8	777,8
ESP8266 Feather	Probe Request ein Kanal	9,74	143,7
ESP-12F	Probe Request ein Kanal	1,82	769,2

## 5.4 Beschleunigungssensor

Um den Energieverbrauch der *Probe-Request*-basierten Lösung weiter zu senken wird in diesem Abschnitt die Einbindung eines Beschleunigungssensors diskutiert.

Im Tunnel Rastatt, in dem auch die Versuche stattfanden, arbeiten die Bauarbeiter in zwei Schichten zu je zwölf Stunden. Nach zehn Tagen Schicht hat ein Arbeiter fünf Tage frei, jeder Arbeiter arbeitet also genau ein Drittel der Gesamtzeit.

Die mobile Einheit behält jedoch derzeit seinen Senderythmus bei, angesichts des hohen Stromverbrauchs beim Senden ist dies ineffizient. Ein Beschleunigungssensor soll bestimmen, wann die mobile Einheit in Bewegung ist. Dadurch wird sie nicht mehr senden, wenn sie nicht getragen wird.

### 5.4.1 LIS3DH

Der LIS3DH ist ein drei-Achsen-Beschleunigungssensor von ST Microelectronics [ST M15]. Er zeichnet sich durch einen *ultra-low-power* Modus und einen Pin für

externe Unterbrechung (*Interrupt*) aus. Der Beschleunigungssensor bietet ein  $I^2C$  und ein SPI Interface um ihn zu konfigurieren. Der LIS3DH benötigt bei einer Frequenz von einem Herz nur 2 Mikroamper (0,002 mA).

Leider konnte bei einem Herz der *Interrupt* durch Laufbewegungen nicht sicher ausgelöst werden, die Frequenz wurde deshalb auf zehn Herz erhöht. Für eine Frequenz von zehn Herz im *ultra-low-power* Modus listet das Datenblatt einen Verbrauch von 3 Mikroamper. Abbildung 5.5 zeigt eine Platine mit integriertem LIS3DH.

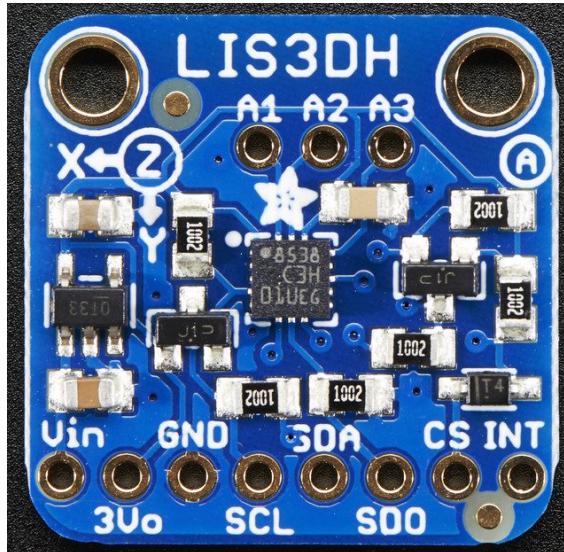


Abbildung 5.5: LIS3DH integriert auf einer Platine für die Entwicklung. Bild von Adafruit Industries.

#### 5.4.2 Abschaltautomatik

Der ESP8266 besitzt einen *Enable Pin* (CH\_PD), ist dieser mit der Versorgungsspannung verbunden werden die internen Spannunggregler aktiviert und der ESP8266 mit Strom versorgt. Die eigentliche Stromversorgung bezieht er dabei jedoch aus dem *Vcc Pin*. Der ESP8266 soll durch den LIS3DH angeschaltet werden, der ESP8266 darf die Stromversorgung bei Bedarf trennen, sie kann danach wieder vom LIS3DH wieder aktiviert werden.

Um dieses Verhalten zu erreichen wird ein *Latch* eingesetzt [Texa03]. Es handelt sich dabei um einen digitalen Schalter mit einem SET-Eingang (An), einem RESET-Eingang (Aus) und einem Ausgang. Der Ausgang wird mit dem *Enable Pin* verbunden, ist er aktiv, wird der ESP8266 aktiv. Der SET-Eingang wird mit dem *Interrupt Pin* des LIS3DH verbunden, er kann damit den Ausgang aktivieren. Der RESET-Eingang wird mit dem ESP8266 verbunden. Es wurde *Pin 16* ausgewählt, da dieser für das Aufwecken aus dem Tiefschlaf zuständig ist. Stattdessen soll er nun die Stromversorgung abschalten und durch die zuvor im `deep_sleep` verbrachte Zeit das Sendeintervall abwarten.

Da das Aufwecken mit *Pin 16* durch das Verbinden des Pins mit der Masse funktioniert, kann damit nicht direkt der RESET-Eingang des Latches betrieben werden. Stattdessen wird das Ergebnis von *Pin 16* mit der Versorgungsspannung über ein XOR-Gatter verschaltet und mit dem RESET-Eingang verbunden [Texa14]. Abbildung 5.6 zeigt das Schema der Verbindung von ESP8266 und LISD3H.

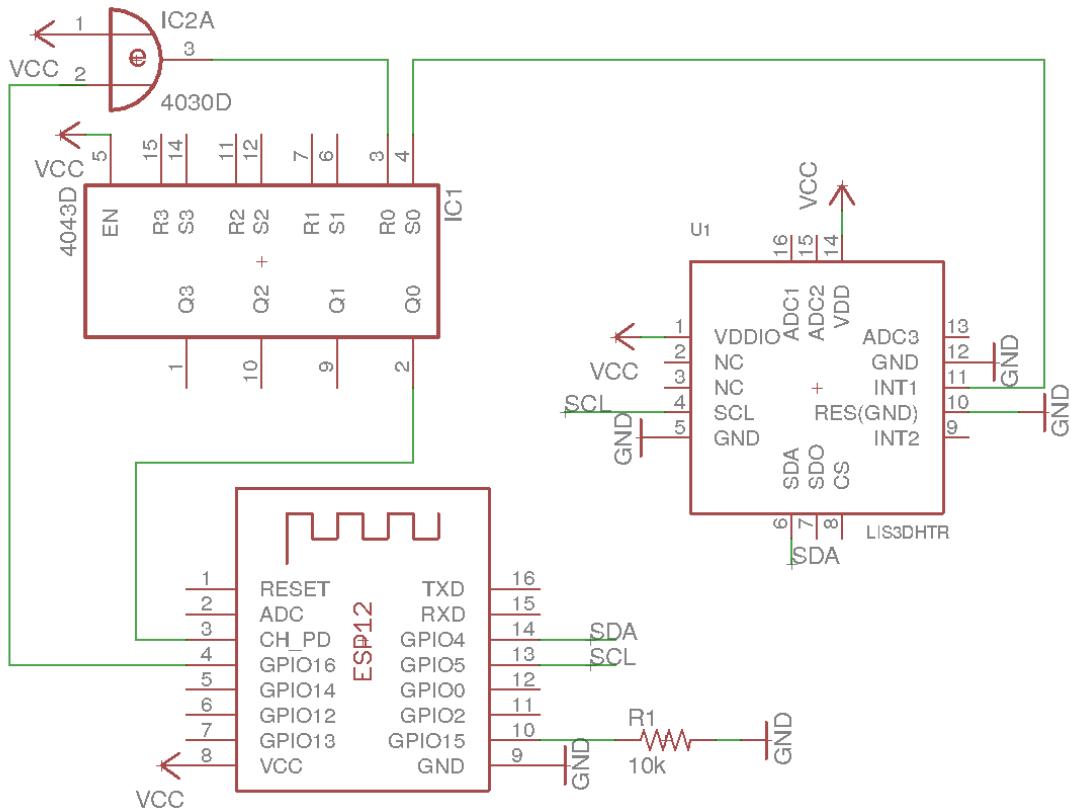


Abbildung 5.6: Schema der Verbindung von ESP8266 und LIS3DH.

### 5.4.3 Bewertung

Der Verbrauch des Beschleunigungssensors ersetzt lediglich den Verbrauch des Mikrocontrollers, die anderen Komponenten bleiben davon unberührt. Zu den 3 Mikroamper Verbrauch addiert sich deshalb der Verbrauch des Spannungswandlers (55 Mikroamper) und des Lithium-Polymer-Ladeschaltkreises (bis zu 100 Mikroamper). Hinzu kommen bis zu 1 Mikroamper für das *Latch* und 10 Mikroamper für das XOR-Gatter.

Die Integration des Beschleunigungssensor kann also den Verbrauch des ESP8266 außerhalb der Arbeitszeiten durch einen Verbrauch von 14 Mikroamper ersetzen, die bis zu 166 Mikroamper Verbrauch der umliegenden Komponenten bleibt jedoch vorhanden. Die Laufzeit für eine nicht bewegte mobile Einheit mit ESP8266 beträgt dann mindestens  $1400mAh / 0,18mA = 7777,77h$ , dies entspricht ca. 324 Tagen.

Bei den vorherigen Prototypen, die eine Assoziation erforderten, muss beachtet werden, dass sie nach dem erneuten Anschalten einen *Scan-* und *Join-*Vorgang ausführen. Damit diese die Einsparung nicht überwiegen, sollte eine Abschaltung erst nach einigen Minuten der Inaktivität erfolgen.



# 6. Direkte Fernlokalisierung mit Bluetooth

Für die direkte Fernlokalisierung mit Bluetooth werden dedizierte Basisstationen eingesetzt. Die Kommunikation zwischen Basisstation und Ortungsserver kann durch ein LAN- oder WLAN-Netzwerk gewährleistet werden. Die Basisstationen bestimmen des *Received Signal Strength Indicator* (RSSI) eingehender Übertragungen der mobilen Enheiten und übermitteln die gemessenen Werte dann an den Ortungsserver. Der Ortungsserver kann mit den gesammelten Werten die Position der mobilen Einheit berechnen.

## 6.1 nRF52832

Der nRF52832 ist eine System-on-Chip Lösung von Nordic Semiconductor. Er vereint eine 32-bit ARM Cortex-M4F CPU, 512kB RAM und einen 2,4GHz Transceiver, der Bluetooth 5.0 inklusive Low Energy und das proprietäre ANT Protokoll unterstützt [Nord17].

Für diese Arbeit wird ein Adafruit Feather nRF52 verwendet, der nRF52832 wird deshalb im Folgenden auf nRF52 abgekürzt. Das Adafruit Feather nRF52 besitzt neben dem nRF52832 Spannungswandler für die 3,3 Volt Umwandlung und einen Schaltkreis für die Verwendung mit Lithium Akkus. Die verbaute CP2104 USB-to-Serial Schnittstelle erlaubt es, den Chip über USB zu programmieren.

Abbildung 6.1 zeigt das Adafruit Feather nRF52. Auch Nordic Semiconductor gibt einige typische Stromverbräuche für ihr System-on-Chip an, diese sind in Tabelle 6.1 aufgeführt.

### 6.1.1 Arduino Bluefruit nRF52 API

Der nRF52 kann ebenfalls mit der Arduino IDE programmiert werden.

Dazu muss dieser zunächst das *Board Support Package* hinzugefügt werden [Town17]. In den Einstellungen wird unter *Additional Boards Manager URLs* die URL [https://www.adafruit.com/package\\_adafruit\\_index.json](https://www.adafruit.com/package_adafruit_index.json) hinzugefügt. Nach einem Neustart der Arduino IDE kann im *Boards Manager* das Paket Adafruit nRF52 installiert

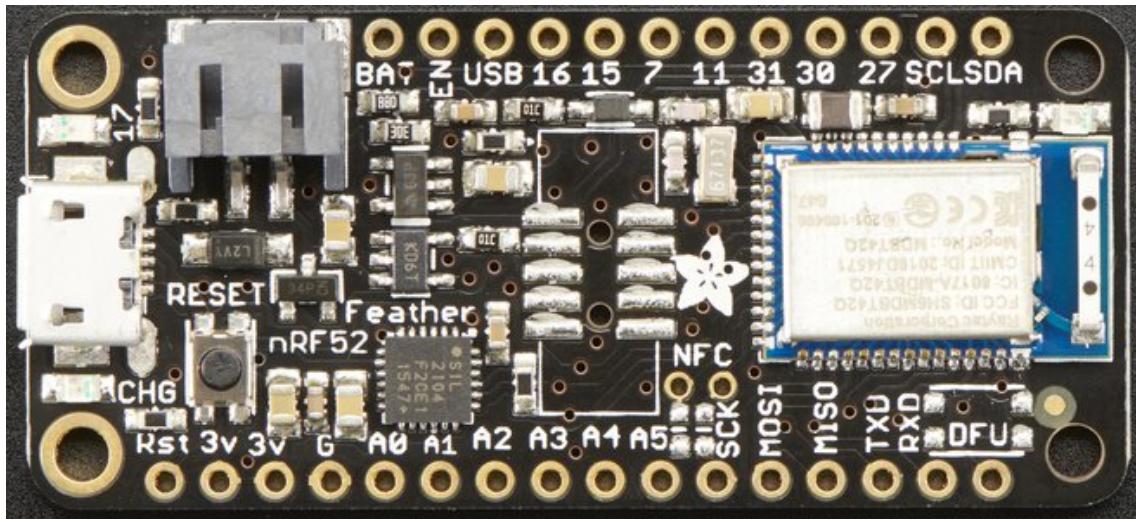


Abbildung 6.1: Adafruit nRF52 Feather. Bild von Adafruit Industries.

werden. Um das Board programmieren zu können wird zusätzlich das *nrfutil* benötigt. Dieses liegt nach der Installation der Boards in  
./arduino15/packages/adafruit/hardware/nrf52/0.6.0/tools/nrfutil-0.5.2 und muss mit sudo pip install -r requirements.txt und sudo python setup.py install installiert werden. Nachdem das Board in der IDE ausgewählt und über USB mit dem Computer verbunden wurde, kann nun eigener Code oder eines der Beispiele aus *Examples for Adafruit Bluefruit nRF52 Feather* mit STRG+U auf den nRF52 geladen werden.

Es wird die Bluefruit nRF52 API Version 0.6.0 verwendet.

## 6.2 Reichweite von Bluetooth

Der Versuch mit Bluetooth wurde an der selben Stelle wie der mit WLAN durchgeführt, siehe dazu Abschnitt 4.2. Es wurde allerdings ein *Raspberry Pi Zero W* als Basisstation verwendet, dieser wurde auf dem *LN-862* platziert, auf Abbildung 4.5 ist sein rotes Gehäuse zu erkennen.

### 6.2.1 Methodik

Die Reichweite wurde erneut in zwei Richtungen geprüft. Zum einen in Richtung der fertig gebohrten Tunnels mit wenigen Hindernissen, zum anderen in Richtung des Vortriebs durch mehrere Stahlhindernisse. Die zwei Messtrecken werden in Abbildung fig:rangefinder skizziert.

Um die Abschirmung durch ein Gehäuse zu simulieren wurde eine stabile Plastikbox verwendet, leider konnte diese nicht vollends geschlossen werden. Für die Messung wurde der Körper zwischen mobile Einheit und Basisstation gebracht und eine mobile Einheit wurde dann als äußer Reichweiteängesehen, wenn versendete Pakete der mobilen Einheit nicht mehr bei der Basisstation ankamen. Durch das Entfernen des körperlichen Hindernisses war es möglich wieder eine Verbindung herzustellen. Die bestimmten Reichweiten werden in zwei Meter Schritten angegeben, da sie mit Hilfe der zwei Meter breiten *Tübbinge* bestimmt wurden.

Tabelle 6.1: Energieverbrauch des nRF52832 in verschiedenen Zuständen, aus [Nord17]

#### Current consumption: Radio

Symbol	Description	Min.	Typ.	Max.	Units
I <sub>RADIO_TX0</sub>	0 dBm TX @ 1 Mb/s Bluetooth Low Energy mode, Clock = HFXO	7.1			mA
I <sub>RADIO_TX1</sub>	-40 dBm TX @ 1 Mb/s Bluetooth Low Energy mode, Clock = HFXO	4.1			mA
I <sub>RADIO_RX0</sub>	Radio RX @ 1 Mb/s Bluetooth Low Energy mode, Clock = HFXO	6.5			mA

#### Current consumption: Radio protocol configurations

Symbol	Description	Min.	Typ.	Max.	Units
I <sub>S0</sub>	CPU running CoreMark from Flash, Radio 0 dBm TX @ 1 Mb/s Bluetooth Low Energy mode, Clock = HFXO, Cache enabled	9.6			mA
I <sub>S1</sub>	CPU running CoreMark from Flash, Radio RX @ 1 Mb/s Bluetooth Low Energy mode, Clock = HFXO, Cache enabled	9.0			mA

#### Current consumption: Ultra-low power

Symbol	Description	Min.	Typ.	Max.	Units
I <sub>ON_RAMOFF_EVENT</sub>	System ON, No RAM retention, Wake on any event	1.2			µA
I <sub>ON_RAMON_EVENT</sub>	System ON, Full RAM retention, Wake on any event	1.5			µA
I <sub>ON_RAMOFF_RTC</sub>	System ON, No RAM retention, Wake on RTC	1.9			µA
I <sub>OFF_RAMOFF_RESET</sub>	System OFF, No RAM retention, Wake on reset	0.3			µA
I <sub>OFF_RAMOFF_GPIO</sub>	System OFF, No RAM retention, Wake on GPIO	1.2			µA
I <sub>OFF_RAMOFF_LPCOMP</sub>	System OFF, No RAM retention, Wake on LPCOMP	1.9			µA
I <sub>OFF_RAMOFF_NFC</sub>	System OFF, No RAM retention, Wake on NFC field	0.7			µA
I <sub>OFF_RAMON_RESET</sub>	System OFF, Full 64 kB RAM retention, Wake on reset	0.7			µA

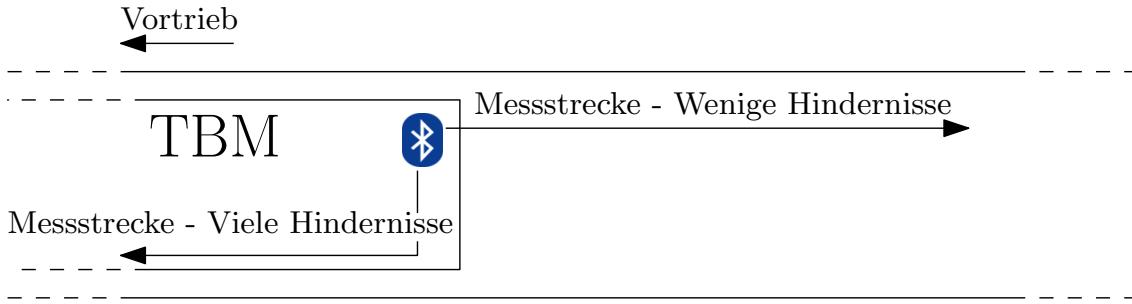


Abbildung 6.2: Messtrecken zur Feststellung der Reichweite von Bluetooth Low Energy.

### 6.2.2 Ergebnisse

Tabelle 6.2 zeigt die Ergebnisse für den nRF52. Für diesen sind keine Ergebnisse mit geschlossenem Gehäuse aufgeführt, denn das Gehäuse ließ sich für diesen Prototyp nicht schließen. Das lose Auflegen des Deckels führte zu keiner Veränderung bei der Reichweite.

### 6.2.3 Bewertung

Um mit 30km/h 32 Meter zu durchqueren benötigt man 3,8 Sekunden. Da jedoch eine hohe Erkennungssicherheit gefordert wurde, sollte das Sendeintervall niedriger gesetzt werden, es wird auf eine Sekunde gesetzt. Damit werden auch bei einer Reichweite von nur 14 Metern auf der Tunnelbohrmaschine ausreichend viele Pakete versendet um den Verlust einzelner zu kompensieren.

Verwendetes Modul	Aufbau	Strecke	Maximale Sendereichweite
nRF52	Offen	Wenige Hindernisse	32m
nRF52	In Gehäuse	Wenige Hindernisse	32m
nRF52	Offen	Viele Hindernisse	14m
nRF52	In Gehäuse	Viele Hindernisse	14m

## 6.3 BLE-Advertising-Implementierung

Die Bluetooth-Low-Energy-Implementierung ist an die Arbeit von Jianyong et al. angelehnt. Es wird immer nach Ablauf des Sendeintervalls ein *Advertising* Paket gesendet.

In der Praxis wird dazu das *Advertising*-Intervall entsprechend gesetzt, dabei handelt es sich um einen in Bluetooth 4.0 spezifizierten Parameter für die Häufigkeit des *Advertisings*. Da die Bluefruit nRF52 API keine Funktion zur Änderung dieses Wertes zur Verfügung stellt muss er direkt geändert werden. Die entsprechende *BLEAdvertising*-Klasse ist in

`~/arduino15/packages/adafruit/hardware/nrf52/0.6.0/libraries/Bluefruit52Lib/src` zu finden.

In *BLEAdvertising.cpp* ist `GAP_ADV_INTERVAL_MS` auf 20 Millisekunden gesetzt, dieser Wert sollte erhöht werden, um den Energieverbrauch zu senken. Beim nRF52 handelt es sich um ein Klasse 2 Bluetooth Gerät mit einer maximalen Sendeleistung bis 4 dBm. Das Sendeintervall wird entsprechend der Untersuchung der Reichweite und maximalen Bewegungsgeschwindigkeit von 30 km/h auf eine Sekunde gesetzt, in dieser Zeit kann sich ein Mitarbeiter maximal 9 Meter bewegen.

Es sollte erneut eine Voruntersuchung des Verbrauchs mit dem Muker TM103 USB-Power-Meter vorgenommen werden. Dieser ist aber nicht in der Lage den Stromverbrauch des nRF52 zu messen.. Die Sendeabschnitte sind zu kurz um einen messbaren Stromverbrauch zu erzeugen.

Deshalb wird zunächst Abbildung 6.1 für eine theoretische Betrachtung des Verbrauchs herangezogen werden.

## 6.4 Untersuchung des Energieverbrauchs

Der Energieverbrauch soll mit dem INA219 genauer bestimmt werden, der INA219 und die verwendete Methodik werden in Abschnitt 4.5 beschrieben.

### 6.4.1 Theoretische Energieverbrauchsabschätzung

Für die Zeit in der nicht gesendet wird, wird der Zustand  $I_{ON\_RAMOFF\_RTC}$  angenommen, da dieser den höchsten Verbrauch aufweist. Für die Sendezeit wird  $I_{RADIO\_TX0}$  angenommen, für ein *Advertising*-Paket, welches zusätzlich den Gerätenamen "TestTag" versendet, werden 24 Bytes (192 Bit) gesendet. Um die Kollisionsvermeidung einzufügen werden vorher 2000 Bit im Zustand  $I_{RADIO\_RX0}$  empfangen, der die restliche Zeit wird in  $I_{ON\_RAMOFF\_RTC}$  verbracht. Es müssen ebenfalls die weiteren Komponenten auf dem Feather bedacht werden. Adafruit gibt für den Spannungswandler einen Verbrauch von 55 Mikroamper und für den Lithium-Polymer-Ladeschaltkreis einen Verbrauch von bis zu 100 Mikroamper an [Frie16].

Der Verbrauch des anderen Komponenten des Feather wird daher konservativ auf 155 Mikroamper geschätzt.

$$y = \left(1s - \frac{\text{Bits\_gesendet}}{1000000b/s} - \frac{\text{Bits\_empfangen}}{1000000b/s}\right) * (I_{ON\_RAMOFF\_RTC} + 155\mu A) + \frac{\text{Bits\_gesendet}}{1000000b/s} * I_{RADIO\_RX0} + \frac{\text{Bits\_empfangen}}{1000000b/s} * I_{RADIO\_RX0}$$

$$y = (1s - 0,000192s - 0,002s) * 0,1569mA + 0,000192 * 7,1mA + 0,002 * 6,5mA$$

$$y \approx 0,156556mA + 0,001363mA + 0,013mA = 0,170919mA$$

## 6.4.2 Tatsächlicher Energieverbrauch von BLE

Abbildung 6.3 zeigt den Lastverlauf für den Start einer mobilen Einheit mit Bluetooth-Low-Energy-Advertising.

Zu Beginn ist eine Startphase zu erkennen, ab 2 Sekunden nach Start des Experiments ist dann das regelmäßige Muster aus Verbrauch im Ruhezustand und kurzen Verbrauchsspitzen beim Senden zu erkennen. Die Kürze des Sendevorgangs bedingt die starke Schwankung bei den Lastspitzen, die Samplingrate von 333Hz reicht hier offenbar nicht aus um dem Sendevorgang vollständig zu erfassen.

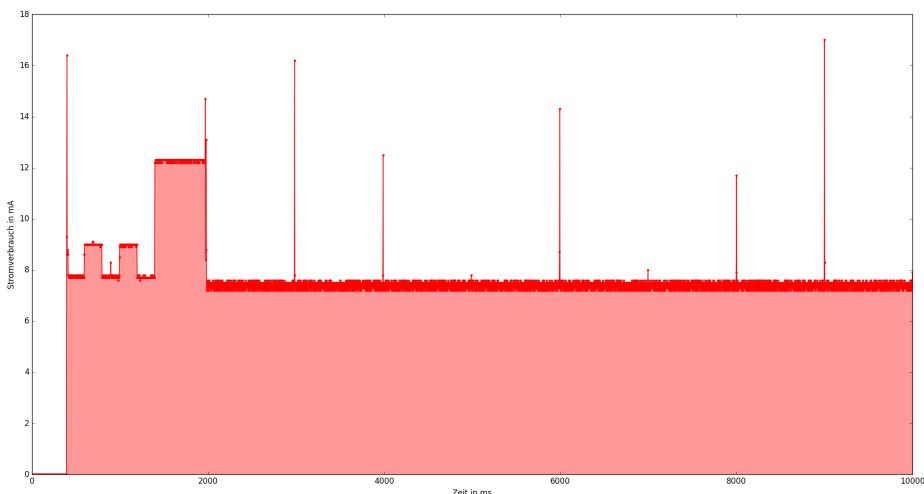


Abbildung 6.3: Lastkurve einer Implementierung von Bluetooth-Low-Energy-Advertising.

Hauptverbrauch liegt jedoch in den 7,2 bis 7,6 Milliamper im Ruhezustand, leider ist in diesem Fall kein einzelnes Modul vorhanden, es kann nur auf dem nRF52 Feather gemessen werden. Da jedoch die selben Komponenten wie beim ESP8266

Feather zum Einsatz kommen, kann angenommen werden, dass auch der Ruheverbrauch vergleichbar ist, dieser liegt zwischen 7 und 7,1 Milliamper. Tabelle 6.3 zeigt deshalb neben dem gemessenen Verbrauch einen projektierten Verbrauch, bei dem ein Ruheverbrauch von 7,05 Milliamper subtrahiert wurde.

Tabelle 6.3: Energieverbrauch mobiler Einheiten mit Bluetooth-Low-Energy-Advertising

Hardware	Programm	$\varnothing$ Verbrauch in mA	Laufzeit in Stunden
nRF52 Feather	Bluetooth Low Energy Advertising	7,37	190
nRF52 (projiziert)	Bluetooth Low Energy Advertising	0,32	4375

# 7. Direkte Fernlokalisierung mit LoRa

Für die direkte Fernlokalisierung mit LoRa werden dedizierte Basisstationen eingesetzt. Die Kommunikation zwischen Basisstation und Ortungsserver kann durch ein LAN- oder WLAN-Netzwerk gewährleistet werden. Die Basisstationen bestimmen den *Received Signal Strength Indicator* (RSSI) eingehender Übertragungen der mobilen Enheiten und übermitteln die gemessenen Werte dann an den Ortungsserver. Der Ortungsserver kann mit den gesammelten Werten die Position der mobilen Einheit berechnen.

## 7.1 RFM95

Bei dem RFM95 handelt es sich um ein *Long Range* (LoRa)-fähiges Radio-Modul für den Frequenzbereich 868/915MHz [Hope06]. 915MHz sind jedoch nur in Amerika lizenzfrei, in Deutschland muss das Radio mit 868MHz betrieben werden.

Für die Entwicklung wird das Modul auf einem Adafruit Feather M0 RFM95 LoRa Radio verwendet. Dieses verwendet zusätzlich einen M0 Mikrocontroller zur Steuerung, einen Lithium-Akku-Ladeschaltkreis und einen Spannungswandler zur einfachen Programmierung des Radios über USB. Zusätzlich ist eine Antenne notwendig, die verwendete Antenne kann dabei einen großen Unterschiede in der Reichweite verursachen, deshalb sollte eine Antenne für 868MHz verwendet werden. Da jedoch eine feste Antenne die mobile Einheit unhandlich machen würde, wird für diese eine Kabelantenne verwendet, diese hat die Länge einer halben Wellenlänge (17,3cm). Abbildung 7.1 zeigt ein Adafruit Feather M0 RFM95 LoRa Radio. Für die Basisstation wird ein Feather M0 RFM95 LoRa Radio mit fester Antenne verwendet.

### 7.1.1 RadioHead RFM9x für Arduino

Der M0 Mikrocontroller ist Arduino kompatibel, wird zusätzlich die RadioHead RFM9x Bibliothek verwendet, kann er über ein SPI-Interface das Radio steuern. Dazu muss zunächst in den Optionen der Arduino IDE unter *Additional Boards Manager URLs* die URL [https://adafruit.github.io/arduino-board-index/package\\_index.html](https://adafruit.github.io/arduino-board-index/package_index.html)

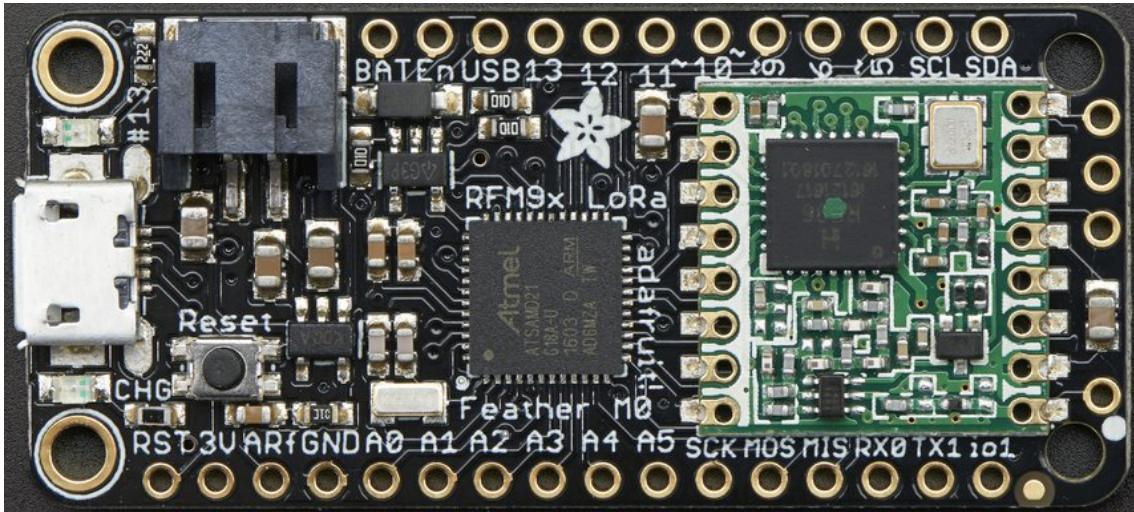


Abbildung 7.1: Adafruit Feather M0 RFM95 LoRa Radio. Bild von Adafruit Industries.

`adafruit_index.json` hinzugefügt werden [Tree16]. Nach einem Neustart der IDE können im *Board Manager* die **Arduino SAMD Boards** installiert werden. Der M0 kann nun programmiert werden, um das RFM95 verwenden zu können muss die RadioHead RFM95 Bibliothek von <https://cdn-learn.adafruit.com/assets/assets/000/035/106/original/RadioHead-1.62.zip> heruntergeladen und nach `/Arduino/libraries/` entpackt werden.

## 7.2 Reichweite von LoRa

Die Versuche mit LoRa wurden an einer anderen Tunnelbaustelle durchgeführt. An der Tunnelbohrstelle Ulm wird mit Sprengungen vorgetrieben. Nach der Sprengung wird der Tunnel mit Spritzbeton ausgekleidet und danach wird in drei Schritten geschalt. Im ersten Schritt wird ein Filz und eine Folie gegen das Eindringen von Wasser eingebracht, danach folgt die Bewehrung und abschließend wird die Bewehrung einbetoniert. Für jeden Schritt wird ein stählerner Schalungswagen verwendet, folglich sind drei stählerne Hindernisse im Tunnel, die Signale absorbieren. Abbildung 7.2 zeigt einen der Schalungswagen, der als Hindernis gedient hat.

### 7.2.1 Methodik

Die Reichweite wurde für zwei Situationen bestimmt. Zum einen durch einen Schalungswagen und dann durch den freien Tunnel, zum anderen durch alle drei Schalungswagen für eine Situation mit maximaler Abschirmung durch die Hindernisse. Die zwei Messtrecken werden in Abbildung fig:rangelora skizziert.

Die mobile Einheit wurde jeweils direkt an der Bewehrung platziert und es wurde auf der selben Seite gelaufen, damit die Abschirmung durch die Schalungswagen maximal ist. Danach wurde die Basisstation immer weiter entfernt, bis keine Pakete der mobilen Einheit mehr empfangen werden konnten. Abbildung 7.4 zeigt die Platzierung der mobilen Einheit an der Bewehrung. Erneut konnte die Plastikbox aufgrund des Versuchsaufbaus nicht geschlossen werden. Die mobile Einheit wurde als "außer Reichweite" angesehen, wenn versendete Pakete der mobilen Einheit



Abbildung 7.2: Schalungswagen für das Betonieren im Tunnel Ulm.

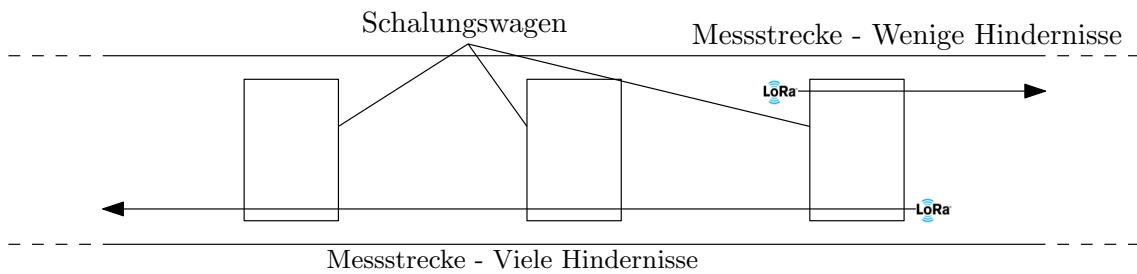


Abbildung 7.3: Messtrecken zur Feststellung der Reichweite von LoRa.

nicht mehr bei der Basisstation ankamen. Durch das Entfernen des körperlichen Hindernisses war es möglich wieder eine Verbindung herzustellen. Es wurde sowohl mit einer Sendeleistung von 5dBm für einen geringen Sendeverbrauch als auch mit 23dBm Sendeleistung für maximale Reichweite gemessen. Die Messungen werden in 12,5 Meter Abständen angegeben, da in diesem Tunnel die Länge eines Schalungselementes 12,5 Meter beträgt. Es werden, wie in Abschnitt 7.1 besprochen, zwei unterschiedliche Typen von Antennen verwendet. Während die Basisstation eine feste Antenne aufweist, verwendet die mobile Einheit eine Kabelantenne entsprechend der halben Wellenlänge.

### 7.2.2 Ergebnisse

Tabelle 7.1 zeigt die Ergebnisse für das RFM95. Da das lose Auflegen des Deckels der Plastikbox zu keiner Veränderung bei der Reichweite führte gibt die Tabelle nur Werte für den offenen Aufbau an. Der Test mit 23dBm Sendeleistung durch alle drei Schalungswagen musste nach 350 Metern abgebrochen werden, weil ein weiterkommen nicht möglich war. Da jedoch für diese Sendeleistung bereits nach circa 250 Metern die Varianz des RSSI den Zusammenhang zwischen Distanz und RSSI überwiegt, kann die tatsächliche Reichweite dieser Sendeleistung nicht ausgenutzt

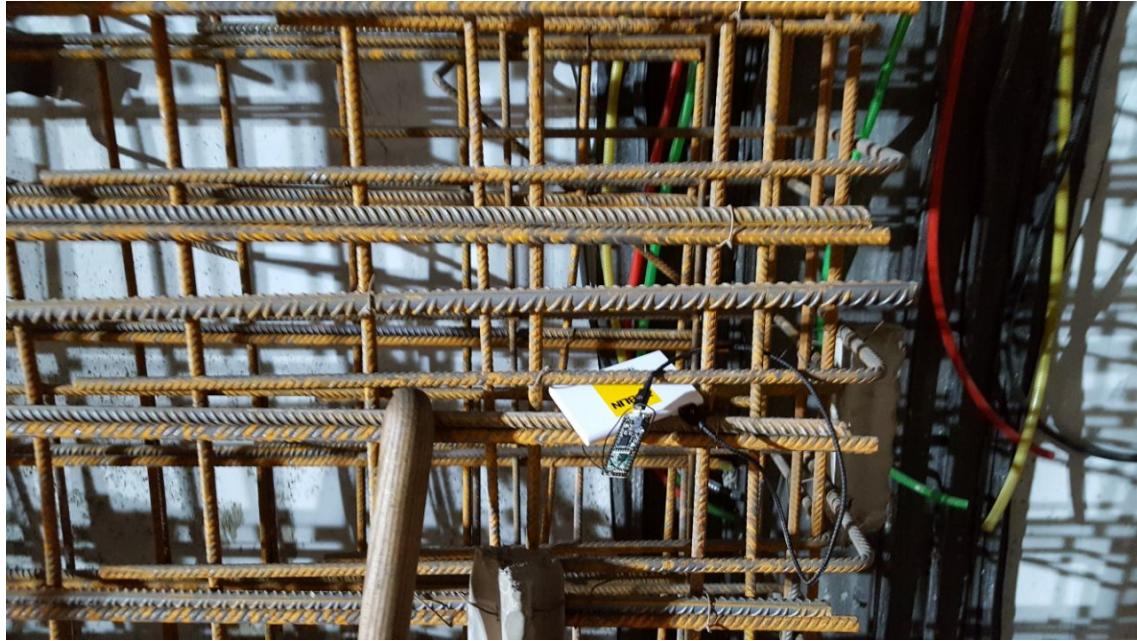


Abbildung 7.4: Platzierung der mobilen Einheit an der Bewehrung des Tunnels.

werden. Für eine sichere Erkennung des 250-Meter-Abschnitts müsste daher alle 500 Meter eine Basisstation aufgestellt werden.

Tabelle 7.1: Sendereichweite LoRa-basierter mobiler Einheiten

Verwendetes Modul	Aufbau	Sendeleistung	Strecke	Maximale Sendereichweite
RFM95	Offen	5dBm	Wenige Hindernisse	250m
RFM95	Offen	5dBm	Viele Hindernisse	100m
RFM95	Offen	23dBm	Wenige Hindernisse	1250m
RFM95	Offen	23dBm	Viele Hindernisse	>350m

### 7.2.3 Bewertung

LoRa entfaltet im Tunnel eine sehr hohe Reichweite und es kann eine lückenlose Überwachung von Personen durchgeführt werden. Voll nutzen lässt sich diese jedoch nicht, da die Varianz des RSSI den Zusammenhang zwischen Distanz und RSSI schon nach circa 250 Metern überwiegt. Hingegen führt eine starke Reduktion der Sendeleistung zu einer mangelnden Penetration von Hindernissen. Daher sollte ein Kompromiss der Sendeleistung geschlossen werden. Eine Sendeleistung von 10dBm sollte eine ausreichende Reichweite im Szenario mit vielen Hindernissen bieten. Alternativ kann die mobile Einheit auch ein Send-Receive Schema umsetzen und von der Basisstation eine dynamisch angepasste Sendeleistung empfangen, die sich nach der Menge der Hindernisse richtet.

Wird die Sendeleistung dynamisch so festgelegt, dass einer Reichweite von 250 Metern erreicht wird benötigt ein Mitarbeiter bei 30 km/h 30 Sekunden um sie zu durchqueren. Eine derart lange Intervallzeit würde jedoch dazu führen, dass eine volle Minute ohne Aktualisierung der Position vergeht, sollte ein Paket verloren gehen. Das Sendeintervall wird deshalb auf zehn Sekunden gesetzt.

## 7.3 LoRa-Implementierung

Die Implementierung für die Lokalisierung mit LoRa ist sehr simpel. Die mobile Einheit versendet regelmäßig ein Paket, welches einen Identifikator für die mobile Einheit enthält. Das Sendeintervall beträgt entsprechend der hohen Reichweite zehn Sekunden.

Die Basisstation empfängt durchgehend und bestimmt den RSSI für eingehende Pakete. Anschließend leitet sie den Identifikator der mobilen Einheit zusammen mit dem RSSI und einer Kennung für die Basisstation an den Ortungsserver weiter.

Auf der mobilen Einheit müssen lediglich die Sendefrequenz, die Sendeleistung und der Identifikator gesetzt werden, dann kann immer nach Ablauf des Sendeintervalls gesendet werden. Die Basisstation muss dabei auf der selben Frequenz aktiv sein und empfangen.

## 7.4 Untersuchung des Energieverbrauchs

Der Energieverbrauch soll mit dem INA219 genauer bestimmt werden, der INA219 und die verwendete Methodik werden in Abschnitt 4.5 beschrieben.

### 7.4.1 Theoretische Energieverbrauchsabschätzung

Für die Berechnung des theoretischen Verbrauchs der mobilen Einheit werden die Datenblätter des M0 Mikrocontrollers und des RFM95 Radios herangezogen. Die dort gelisteten Verbräuche sind in Tabelle 7.2 und Tabelle 7.3 zu finden.

Tabelle 7.2: Stromverbrauch des M0 Mikrocontrollers, aus [NXP 16]

Symbol	Parameter	Conditions	Min	Typ <sup>[1]</sup>	Max	Unit
V <sub>DD</sub>	supply voltage (core and external rail)		1.8	3.3	3.6	V
I <sub>DD</sub>	supply current	Active mode; code while(1{}) executed from flash				
		system clock = 12 MHz V <sub>DD</sub> = 3.3 V	[2][3][4] [5][6]	-	2	-
		system clock = 50 MHz V <sub>DD</sub> = 3.3 V	[2][3][5] [6][7]	-	7	-
		Sleep mode; system clock = 12 MHz V <sub>DD</sub> = 3.3 V	[2][3][4] [5][6]	-	1	-
		Deep-sleep mode; V <sub>DD</sub> = 3.3 V	[2][3][8]	-	2	-
						µA

Da das Adafruit Feather M0 RFM95 Lora Radio sich nur durch die Ersetzung des CP2014 durch den Cortex-M0 zum Adafruit Feather nRF52 unterscheidet werden die 155 Mikroamper für die sonstigen Komponenten übernommen.

Die Sendeleistung des RFM95 Radio ist zwischen 5dBm und 23dBm einstellbar. Das Datenblatt listet jedoch nur Verbräuche zwischen 7dBm und 20dBm Sendeleistung, der Verbrauch wird für diese beiden Sendeleistungen berechnet. Für die Nachricht "TagTest" werden 20 Bytes (160 Bits) versendet, es wird angenommen, dass zuvor 500 Bits für die Kollisionskontrolle belauscht werden.

Tabelle 7.3: Stromverbrauch des RFM95, aus [Hope06]

Symbol	Description	Conditions	Min	Typ	Max	Unit
IDDSL	Supply current in Sleep mode		-	0.2	1	uA
IDDIDLE	Supply current in Idle mode	RC oscillator enabled	-	1.5	-	uA
IDDST	Supply current in Standby mode	Crystal oscillator enabled	-	1.6	1.8	mA
IDDFS	Supply current in Synthesizer mode	FSRx	-	5.8	-	mA
IDDR	Supply current in Receive mode	LnaBoost Off, higher bands LnaBoost On, higher bands Lower bands	- - -	10.8 11.5 12.1	-	mA
IDDT	Supply current in Transmit mode with impedance matching	RFOP = +20 dBm, on PA_BOOST RFOP = +17 dBm, on PA_BOOST RFOP = +13 dBm, on RFO_LF/HF pin RFOP = + 7 dBm, on RFO_LF/HF pin	- - - -	120 87 29 20	-	mA

$$y = \frac{1}{10} * [(10 - \frac{\text{Bits\_gesendet}}{21875b/s} - \frac{\text{Bits\_empfangen}}{21875b/s}) * (M0\_Deep\_sleep\_mode + RMF95\_IDDIDLE + 155\mu A) + \frac{\text{Bits\_gesendet}}{21875b/s} * RFM95\_XdBm + \frac{\text{Bits\_empfangen}}{21875b/s} * RFM95\_IDDR]$$

$$y_{20dBm} = \frac{1}{10} * [(10s - 0,0073s - 0,0229s) * 0,1585mA + 0,0073s * 120mA + 0,0229s * 12,1mA]$$

$$y_{20dBm} \approx \frac{1}{10} * (1,58mA + 0,876mA + 0,2771mA) = 0,2683mA$$

$$y_{7dBm} = \frac{1}{10} * [(10s - 0,0073s - 0,0229s) * 0,1585mA + 0,0073s * 20mA + 0,0229s * 12,1mA]$$

$$y_{7dBm} \approx \frac{1}{10} * (1,58mA + 0,146mA + 0,2771mA) = 0,1953mA$$

#### 7.4.2 Tatsächlicher Energieverbrauch von LoRa

Der Stromverbrauch der Implementierung mit LoRa wurde mit 5dBm und 23dBm Sendeleistung überprüft. Abbildung 7.5 zeigt den Lastverlauf für den Start einer mobilen Einheit mit LoRA bei 23dBm.

Nach einer circa drei sekündigen Startphase wechselt der LoRa Feather in den regulären Betrieb, er sollte dann immer nach zehn Sekunden Ruhezustand senden. Dieser jedoch vom Zeitgeber nicht ganz eingehalten, die mobile Einheit befindet sich zwischen den Sendevorgängen circa elf Sekunden im Ruhezustand, dies sollte bei der Implementierung beachtet werden. Im Ruhezustand liegt der Verbrauch des LoRa Feather bei 2,5 bis 2,7 Milliamper, beim Senden unterscheiden sich die Verbräuche je nach Sendeleistung deutlich.

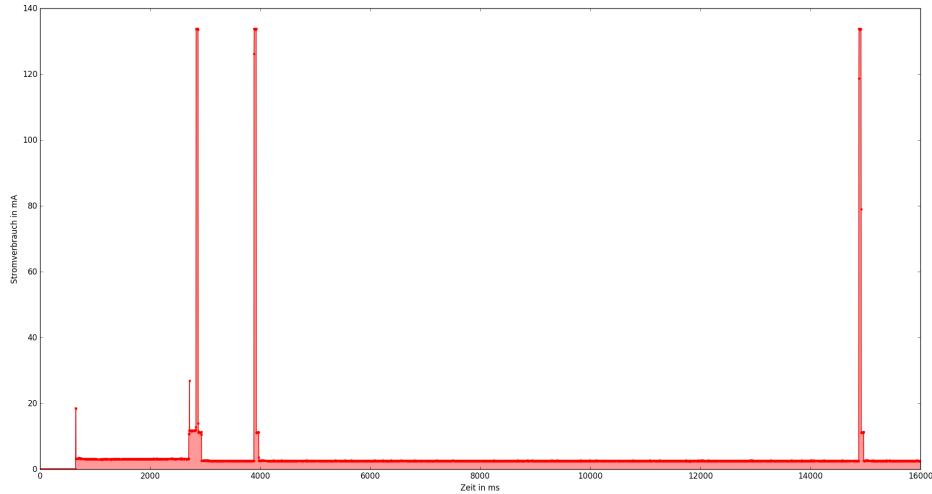


Abbildung 7.5: Lastkurve einer Implementierung mit LoRa.

Die Abbildung 7.6 zeigt die Sendeverbäuche für eine Sendeleistung von 23dBm in Rot und für 5dBm in Grün. Gut zu erkennen ist hier auch, dass ein Sendevorgang bei LoRa deutlich länger als bei Bluetooth Low Energy dauert, obwohl vergleichbar viele Bits übertragen werden.

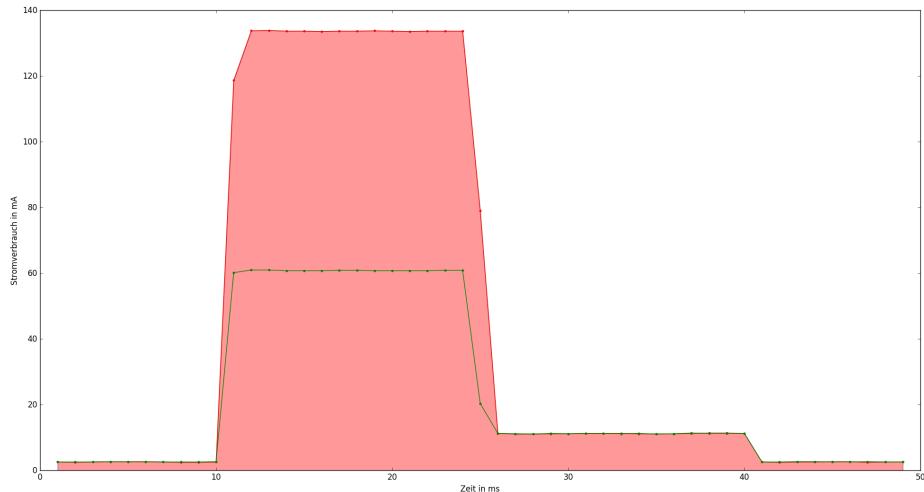


Abbildung 7.6: Lastkurve eines Ortungsvorgangs mit LoRa.

Die Ergebnisse der Messungen sind in Tabelle 7.4 gelistet.

Für den LoRa Feather liegt der Ruheverbrauch deutlich niedriger, da jedoch die selben Schaltungen für Akku-Ladung und Spannungsregelung zum Einsatz kommen scheint der CP2104, welcher zum Programmieren des ESP8266 beziehungsweise nRF52 dient, mindestens 4,5 Milliamper zu verbrauchen.

Tabelle 7.4: Energieverbrauch mobiler Einheiten mit LoRa

Hardware	Programm	Ø Verbrauch in mA	Laufzeit in Stunden
LoRa Feather	LoRa mit 23dBm Sendeleistung	3,16	443
LoRa Feather	LoRa mit 5dBm Sendeleistung	2,88	486,1

## **8. Zusammenfassung und Fazit**

Ergänzen



# Literaturverzeichnis

- [AiRI17a] AiRISTA Flow. Ekahau A4 Asset Tag. [https://www.airistaflow.com/wp-content/uploads/2016/07/AiRISTAFLOW\\_Ekahau\\_RTLS\\_A4\\_DS.pdf](https://www.airistaflow.com/wp-content/uploads/2016/07/AiRISTAFLOW_Ekahau_RTLS_A4_DS.pdf), Mai 2017.
- [AiRI17b] AiRISTA Flow. Ekahau B4 Badge Tag. [https://www.airistaflow.com/wp-content/uploads/2016/09/AiRISTAFLOW\\_Ekahau\\_B4\\_DS.pdf](https://www.airistaflow.com/wp-content/uploads/2016/09/AiRISTAFLOW_Ekahau_B4_DS.pdf), Mai 2017.
- [AiRI17c] AiRISTA Flow. Offizielle Website von AiRISTA Flow. <https://www.airistaflow.com/#>, Mai 2017.
- [ANSSY13] H. Abdel-Nasser, R. Samir, I. Sabek und M. Youssef. MonoPHY: Mono-stream-based device-free WLAN localization via physical layer information. In *Wireless communications and networking conference (WCNC), 2013 IEEE*. IEEE, 2013, S. 4546–4551.
- [BadG08] M. S. Bargh und R. de Groote. Indoor localization based on response rate of bluetooth inquiries. In *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*. ACM, 2008, S. 49–54.
- [BaPa00] P. Bahl und V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Band 2. Ieee, 2000, S. 775–784.
- [BaPB00] P. Bahl, V. N. Padmanabhan und A. Balachandran. Enhancements to the RADAR user location and tracking system. *Microsoft Research* 2(MSR-TR-2000-12), 2000, S. 775–784.
- [BCIM17a] M. Banzi, D. Cuartielles, T. Igoe und D. A. Mellis. Arduino IDE. <https://github.com/arduino/Arduino/>, Juni 2017.
- [BCIM<sup>+</sup>17b] M. Banzi, D. Cuartielles, T. Igoe, D. A. Mellis, C. Klippel, I. Grokhotkov, Espressif Systems, P. Andersson, R. Hempel und C. Rich. ESP8266 Arduino Core. <https://github.com/esp8266/Arduino>, Juni 2017.
- [Blue10a] Bluetooth Special Interest Group. *Specification of the Bluetooth System 4.0*. Bluetooth SIG, Inc. 2010.

- [Blue10b] Bluetooth Special Interest Group. Volume 1 - Part A - 3.3.2.2 Advertisement broadcast channel. In *Specification of the Bluetooth System 4.0*, Band 0. Bluetooth SIG, Inc., 2010, S. 162–163.
- [Blue10c] Bluetooth Special Interest Group. Volume 2 - Part A - 3 TRANSMITTER CHARACTERISTICS. In *Specification of the Bluetooth System 4.0*, Band 0. Bluetooth SIG, Inc., 2010, S. 282.
- [Blue10d] Bluetooth Special Interest Group. Volume 2 - Part A - 4.2.2.2 Advertising Procedure. In *Specification of the Bluetooth System 4.0*, Band 0. Bluetooth SIG, Inc., 2010, S. 187.
- [Blue10e] Bluetooth Special Interest Group. Volume 2 - Part A - 4.2.2.3 Scanning Procedure. In *Specification of the Bluetooth System 4.0*, Band 0. Bluetooth SIG, Inc., 2010, S. 187–188.
- [Blue10f] Bluetooth Special Interest Group. Volume 2 - Part B - 2.5 Inquiry Scan Physical Channel. In *Specification of the Bluetooth System 4.0*, Band 0. Bluetooth SIG, Inc., 2010, S. 327–328.
- [Blue10g] Bluetooth Special Interest Group. Volume 6 - Part B - 2.3 Advertising Channel PDU. In *Specification of the Bluetooth System 4.0*, Band 0. Bluetooth SIG, Inc., 2010, S. 2202–2208.
- [ChLu07] Y. Chen und R. Luo. Design and implementation of a wifi-based local locating system. In *Portable Information Devices, 2007. PORTABLE07. IEEE International Conference on*. IEEE, 2007, S. 1–5.
- [DrMS98] C. Drane, M. Macnaughtan und C. Scott. Positioning GSM telephones. *IEEE Communications Magazine* 36(4), 1998, S. 46–54.
- [Ekah17] Ekahau. Ekahau W4 Wearable Tag. [https://www.airistaflow.com/wp-content/uploads/2016/07/Ekahau\\_RTLS\\_W4\\_DS\\_1\\_15.pdf](https://www.airistaflow.com/wp-content/uploads/2016/07/Ekahau_RTLS_W4_DS_1_15.pdf), Mai 2017.
- [ESP 17] ESP Community. ESP OpenN SDK. <https://github.com/pfalcon/esp-open-sdk>, Juni 2017.
- [Espr17] Espressif Systems IOT Team. ESP8266EX Datasheet. [http://espressif.com/sites/default/files/documentation/0a-esp8266ex-datasheet\\_en.pdf](http://espressif.com/sites/default/files/documentation/0a-esp8266ex-datasheet_en.pdf), Mai 2017.
- [Frie16] L. Fried. Adafruit Feather 32u4 with LoRa Radio Module - Power Management. <https://learn.adafruit.com/adafruit-feather-32u4-radio-with-lora-radio-module/power-management#radio-power-draw>, April 2016.
- [Frie17] L. Fried. Adafruit Feather HUZZAH ESP8266 - Using Arduino IDE. <https://learn.adafruit.com/adafruit-feather-huzzah-esp8266-using-arduino-ide>, Juni 2017.
- [Hope06] Hope Microelectronics. RFM95/96/97/98(W) - Low Power Long Range Transceiver Module. [http://www.hoperf.com/upload/rf/RFM95\\_96\\_97\\_98W.pdf](http://www.hoperf.com/upload/rf/RFM95_96_97_98W.pdf), Januar 2006.

- [HoSo07] A. M. Hossain und W.-S. Soh. A comprehensive study of bluetooth signal parameters for localization. In *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on.* IEEE, 2007, S. 1–5.
- [IEEE02] IEEE Computer Society. *Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs).* IEEE. 2002.
- [IEEE12a] IEEE Computer Society. 10.1.4 Acquiring synchronization, scanning. In *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* IEEE, 2012, S. 977–980.
- [IEEE12b] IEEE Computer Society. 10.1.4.3.3 Active Scanning Procedure. In *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* Band 1. IEEE, 2012, S. 980.
- [IEEE12c] IEEE Computer Society. 10.3 STA authentication and association. In *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* Band 1. IEEE, 2012, S. 1011–1013.
- [IEEE12d] IEEE Computer Society. 6.3.5 Authenticate. In *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* Band 1. IEEE, 2012, S. 117–121.
- [IEEE12e] IEEE Computer Society. 6.3.7 Associate. In *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* Band 1. IEEE, 2012, S. 127–133.
- [IEEE12f] IEEE Computer Society. 6.3.8 Reassociate. In *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* Band 1. IEEE, 2012, S. 133–144.
- [IEEE12g] IEEE Computer Society. 8.2.4.1.3 Type and Subtype fields. In *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* Band 1. IEEE, 2012, S. 382–383.
- [IEEE12h] IEEE Computer Society. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* IEEE. 2012.
- [JHZZ14] Z. Jianyong, L. Haiyong, C. Zili und L. Zhaojun. RSSI based Bluetooth low energy indoor positioning. In *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on.* IEEE, 2014, S. 526–533.
- [KiKo16] S. Kim und J. Ko. Poster: Low-complexity Outdoor Localization for Long-range, Low-power Radios. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion.* ACM, 2016, S. 44–44.
- [LANC17] LANCOM Systems. Rouge-Detection-Funktion. [https://www.lancom-systems.de/docs/LCOS-Refmanual/9.10-Rel/DE/topics/wlanmonitor\\_rogue\\_detection.html](https://www.lancom-systems.de/docs/LCOS-Refmanual/9.10-Rel/DE/topics/wlanmonitor_rogue_detection.html), Juni 2017.

- [LDBL07] H. Liu, H. Darabi, P. Banerjee und J. Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37(6), 2007, S. 1067–1080.
- [LGGLD<sup>+</sup>11] G. Lui, T. Gallagher, B. Li, A. G. Dempster und C. Rizos. Differences in RSSI readings made by different Wi-Fi chipsets: A limitation of WLAN localization. In *Localization and GNSS (ICL-GNSS), 2011 International Conference on*. IEEE, 2011, S. 53–57.
- [LPLaY00] X. Li, K. Pahlavan, M. Latva-aho und M. Ylianttila. Comparison of indoor geolocation methods in DSSS and OFDM wireless LAN systems. In *Vehicular Technology Conference, 2000. IEEE-VTS Fall VTC 2000. 52nd*, Band 6. IEEE, 2000, S. 3015–3020.
- [LRJT<sup>+</sup>10] P. Ling, C. Ruizhi, L. Jingbin, T. Tenhunen und H. Kuusniemi. Inquiry-based bluetooth indoor positioning via RSSI probability distributions. In *the 2nd International Conference on Advances in Satellite and Space Communications (SPACOMM)*, 2010, S. 151–156.
- [Maur16] D. Maurer. *Unterstützung der Sicherheitstechnik im Tunnelbau durch eine Applikation*. Karlsruher Institut für Technologie. 2016.
- [MKML06] K. Muthukrishnan, G. Koprinkov, N. Meratnia und M. Lijding. Using time-of-flight for WLAN localization: feasibility study. 2006.
- [Nord17] Nordic Semiconductor. nRF52832 Product Specification v1.3 - Datenblatt. [http://infocenter.nordicsemi.com/pdf/nRF52832\\_PS\\_v1.3.pdf](http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.3.pdf), Februar 2017.
- [NXP 16] NXP Semiconductors. LPC1102/1104 32-bit ARM Cortex-M0 microcontroller; 32 kB flash and 8 kB SRAM. [http://www.nxp.com/docs/en/data-sheet/LPC1102\\_1104.pdf](http://www.nxp.com/docs/en/data-sheet/LPC1102_1104.pdf), September 2016.
- [PrKC02] P. Prasithsangaree, P. Krishnamurthy und P. Chrysanthis. On indoor position location with wireless LANs. In *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, Band 2. IEEE, 2002, S. 720–724.
- [Riga16] Rigado LLC. Practical BLE Throughput. [https://rigado.zendesk.com/hc/en-us/article\\_attachments/211413708/Practical-BLE-Throughput.pdf](https://rigado.zendesk.com/hc/en-us/article_attachments/211413708/Practical-BLE-Throughput.pdf), Mai 2016.
- [SkJa09] M. J. Skibniewski und W.-S. Jang. Simulation of Accuracy Performance for Wireless Sensor-Based Construction Asset Tracking. *Computer-Aided Civil and Infrastructure Engineering* 24(5), 2009, S. 335–345.
- [SLEK<sup>+</sup>15] N. Sornin, M. Luis, T. Eirich, T. Kramp und O. Hersent. *LoRaWAN Specification*, Band 1.0.1. LoRa Alliance. 2015.
- [ST M15] ST Microelectronics. MEMS digital output motion sensor: ultra-low-power high-performance 3-axis "nanoäccelerometer - Datenblatt. <http://www.st.com/content/ccc/resource/technical/>

- document/datasheet/3c/ae/50/85/d6/b1/46/fe/CD00274221.pdf/files/CD00274221.pdf/jcr:content/translations/en.CD00274221.pdf, Dezember 2015.
- [Stan17a] Stanley Healthcare. AeroScout: Healthcare Asset Tracking & Management. <https://www.stanleyhealthcare.com/solutions/health-systems/supply-chain-asset-management/asset-management>, Mai 2017.
- [Stan17b] Stanley Healthcare. AeroScout: Staff Assist for Improved Security. <https://www.stanleyhealthcare.com/solutions/health-systems/security-protection/staff-security>, Mai 2017.
- [Stan17c] Stanley Healthcare. T14 Patient and Staff Badge. <https://www.stanleyhealthcare.com/sites/stanleyhealthcare.com/files/documents/T14%20Badge%20Data%20Sheet.pdf>, Mai 2017.
- [StTr12] T. Streichert und M. Traub. *Elektrik/Elektronik-Architekturen im Kraftfahrzeug: Modellierung und Bewertung von Echtzeitsystemen*. Springer-Verlag. 2012.
- [Texa03] Texas Instruments. CMOS Quad 3-State R/S Latches - Datenblatt. <http://www.ti.com/lit/ds/symlink/cd4043b.pdf>, Oktober 2003.
- [Texa14] Texas Instruments. SN74AHC1G86 Single 2-Input Exclusive-OR Gate - Datenblatt. <http://www.ti.com/lit/ds/symlink/sn74ahc1g86.pdf>, Dezember 2014.
- [Texa15] Texas Instruments. INA219 Zer $\varnothing$ -Drift, Bidirectional Current/Power Monitor With I<sub>2</sub>C Interface - Datenblatt. <http://www.ti.com/lit/ds/symlink/ina219.pdf>, Dezember 2015.
- [Torr84] D. J. Torrieri. Statistical theory of passive location systems. *IEEE transactions on Aerospace and Electronic Systems* (2), 1984, S. 183–198.
- [Town17] K. Townsend. Bluefruit nRF52 Feather Learning Guide. <https://learn.adafruit.com/bluefruit-nrf52-feather-learning-guide?view=all>, März 2017.
- [Tree16] T. Treece. Adafruit Feather M0 Radio with LoRa Radio Module - Arduino IDE Setup. <https://learn.adafruit.com/adafruit-feather-m0-radio-with-lora-radio-module/setup>, Juni 2016.
- [WiKP09] S. B. Wibowo, M. Klepal und D. Pesch. Time of flight ranging using off-the-self ieee802.11 wifi tags. In *Proceedings of the International Conference on Positioning and Context-Awareness (PoCA '09)*, 2009.

