Karlsruhe Institute of Technology

Energieoptimierung WLAN-basierter Ortungseinheiten

Masterarbeit
von

Marius Wodtke

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

Erstgutachter:	Prof. Dr. H. Schmeck
Zweitgutachter:	Prof. Dr. ?. ?????????
Betreuender Mitarbeiter:	Dipl.-Inform. K. Bao

Bearbeitungszeit: 01. April 2017 – 31. Oktober 2017

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig 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

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.
Die geringe Genauigkeit hat zudem Vorteile bezüglich des Datenschutzes, da sie verhindert, dass die Arbeiter mit Bewegungsprofilen analysiert werden können und zum Beispiel geprüft wird wer sich wie lange im Pausenraum aufhält.
Die Ortung wird derzeit bei einem Referenzunternehmen mittels Bluetooth durchgeführt, dabei sind die Knoten 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 Knoten an und bereitet die Ergebnisse graphisch auf.

1.1 Zielsetzung der Arbeit

Ziel der Arbeit soll der Entwurf und die Implementierung eines Bereichsortungssystems für Personen in Tunnelanlagen unter Verwendung eines bestehenden Netzwerks von WLAN Access Points (APs) 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 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. Statt der nur wenige Tage umfassenden Laufzeiten anderer WLAN basierter Tags ist das Ziel dieser Arbeit eine Laufzeit von mehreren Monaten.
Der Entwurfsraum umfasst in einem ersten Schritt keine Änderungen an den Access Points, ihre Anzahl, Position, Software und Hardware ist gegeben.
Anschließend wird diese Beschränkung gelockert und die Software der APs kann verändert werden, bei diesen Veränderungen sollen aber die grundlegenden Mechanismen der 802.11 Spezifikation erhalten bleiben. So soll es nicht assozierten Clients nicht möglich sein direkt mit Servern im Netzwerk zu kommunizieren, da dies die Sicherheit des gesamten Netzwerks gefährden könnte.
In einer zweiten Lockerung der Beschränkungen soll auch die Hardware veränderbar sein, dies widerspricht zwar der Annahme der bestehenden Struktur von WLAN APs, da diese potenziell ausgetauscht werden müssten, erweitert den Handlungsspielraum jedoch enorm und erlaubt es die vorherigen Implementierungen mit einer energetisch effizienten und potenziell nicht WLAN-basierten Protokollen zu vergleichen.
Für jeden dieser Schritte müssen die angrenzenden Komponenten (Ortungsserver und mobile Einheiten) implementiert und hinsichtlich des Energieverbrauchs und der Erkennungssicherheit evaluiert werden. Abschließend sollen die drei Systeme miteinander verglichen werden.

1.2 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 Access Points, zuverlässig erkannt werden. Bei den von den Zielpersonen getragenen Positionssendern, den sogenannten Tags, soll es sich in den ersten beiden Schritten um Geräte handeln, die auf WLAN Basis arbeiten und somit mit den Access Points kompatibel sind. Im dritten Schritt ist dies nicht erforderlich und die Kompatibilität wird durch technische Änderungen am AP wiederhergestellt. Die Tags sollen eine Laufzeit von bis zu drei Jahren erreichen, sie müssen jedoch mindestens eine Laufzeit von sechs Monaten aufweisen, um als verwendbar angesehen zu werden. Dabei sind Größe und Gewicht der Lösung zu beachten, zwar kann die Laufzeit eines Tags jederzeit durch die Vergrößerung des Energiespeichers herbeigeführt werden, die Tags müssen jedoch mühelos von den Zielpersonen an einem Band um den Hals getragen werden können. Zuletzt soll unter Rücksichtnahme auf das beschriebene Szenario die Komplexität der IT-Infrastruktur so gering wie möglich gehalten werden um ein stabiles und kostengünstiges System zu garantieren.

1.3 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.

1.3.1 Protokoll

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 802.11 (WLAN), Bluetooth und Radio Frequency Identification (RFID).

1.3.2 Topologie

Die Topologie hat zwei Dimensionen, Fernlokalisierung versus Selbstlokalisierung und direkt versus indirekt und beschreibt wie Knoten und mobile Einheiten zusammenwirken.
Bei der direkten Fernlokalisierung werden von mobilen Einheiten gesendete Signale an Knotenpunkten gemessen und die Ergebnisse an einen zentralen Ortungsserver weitergegeben, dieser führt dann die Lokalisierung durch. Das Ergebnis der Lokalisierung ist nur zentral verfügbar.
Bei der direkten Selbstlokalisierung senden hingegen die Knoten 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.
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.
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.

1.3.3 Lokalisierungsprinzip

Das einfachste Lokalisierungsprinzip ist das Umgebungsprinzip, hier wird eine mobile Einheit dem Knoten mit dem stärksten Signal zugeordnet, der Knoten 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 ein Knoten in Reichweite der mobilen Einheit sein, was diese Möglichkeit auch in komplexeren Lokalisierungsprinzipien zu einer sinnvollen Ausweichmöglichkeit macht, falls nicht genügend Knoten in Reichweite sind.
Die geometrische Bestimmung der Position einer mobilen Einheit mit drei oder mehr Knoten nennt man Trilateration. Dabei wird üblicherweise für jeden Knoten die Distanz zur mobilen Einheit bestimmt, anschließend wird ein Kreis mit der gemessenen Distanz um jeden Knoten 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 Knoten möglich, dies nennt man Triangulation. Der Winkel kann zum Beispiel über die zeitlich Differenz der ankommenden Signale an den synchronisierten Knoten bestimmt werden, dann wird von jedem Knoten 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 Knoten in Reichweite sein um die mobile Einheit zu lokalisieren.
Das aufwendigste Lokalisierungsprinzip ist die Szenenanalyse. Hier werden zunächst in einer offline-Phase Vektoren $(m_1, m_2, ..., m_n, p_x, p_y, p_z)^T$ 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 $(m_1, m_2, ..., m_n)^T$ zu schätzen. Im zweidimensionalen Fall wird dabei $p_z = 0$ angenommen.

1.3.4 Messgrößen

Beliebte Messgrößen für die Positionsbestimmung sind Ankunftszeit des Signals (time of arrival, TOA), die Differenz der Ankuftszeiten (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 Einfallsswinkel des Signals mit mehreren gerichteten Antennen (angle of arrival, AOA).

1.3.4.1 Time of arrival

TOA beruht auf der begrenzten Ausbreitungsgeschwindigkeit des Signals, hochfrequente Signale breiten sich mit Lichtgeschwindigkeit $c = 299.792.458\ m/s$, Ultraschall mit Schallgeschwindigkeit $c \approx 343\ m/s$ bei 20 $°C$ aus, damit ergibt sich die Distanz $d_i = c * (t_{empfangen,i} - t_{gesendet})$ mit i sei der empfangende Knoten. Die Position der mobilen Einheit kann dann mit der Methode der kleinsten Quadrate bestimmt

werden, dazu wird die Kostenfunktion $F(x, y, t_{gesendet}) = \sum_{i=1}^{N} \alpha_i^2 f_i^2(x, y, t_{gesendet})$ mit $f_i(x, y, t_{gesendet}) = d_i - \sqrt{(x_i - x)^2 + (y_i - y)^2}$ und α_i sei die Konfidenz von Knoten i, minimiert. Dies erfordert, dass die Knoten zeitsynchronisiert sind und $t_{empfangen,i}$ 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 $c = 299.792.458\ m/s$ sehr stark, da das Signal nur $0.00000000336\ s/m$ 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 Knoten 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.

1.3.4.2 Time difference of arrival

Bei TDOA wird statt der direkten Ankunftszeiten die Differenz der Ankunftszeiten gemessen, dies erfordert üblicherweise Zeitsynchronität bei den Knoten, nicht jedoch bei der mobilen Einheit, da nur die Ankuftszeiten für die Berechnung relevant sind. Die mobile Einheit liegt dabei auf dem Schnittpunkt zweier Hyperboloiden, die jeweils zwischen den Knoten, die TDOA gemessen haben, und einem Referenzknoten aufgespannt werden. Die Hyperboloiden werden aufgestellt durch
$R_{i,j} = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}$ mit (x_i, y_i, z_i) sei der jeweilige Messknoten, (x_j, y_j, z_j) sei der gemeinsame Referenzknoten und (x, y, z) sei die Position der mobile Einheit. Abb. 1.1 zeigt die Lösung

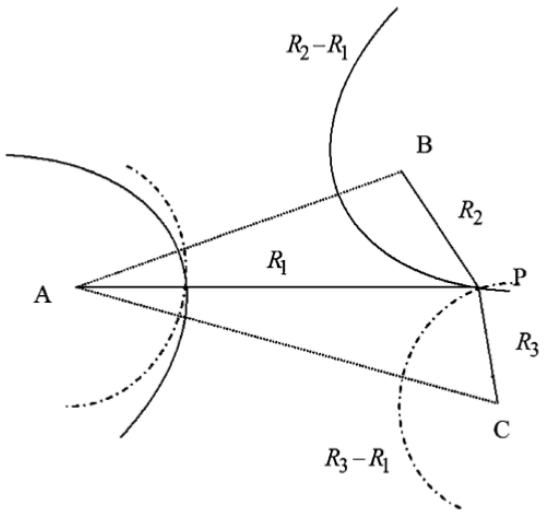

Abbildung 1.1: Positionsbestimmung mit der Differenz der Ankunftszeiten (TDOA), aus [LDBL07].

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

1.3.4.3 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 Knoten und mobile Einheit auf einen Abstand von 0 Metern gebracht werden, anschließend kann die Entfernung $d = c * (t_{empfangen} - t_{gesendet} - t_{Verarbeitung})$ 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 802.11 häufig auftreten.

1.3.4.4 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 Knoten anhand des Pfadverlustes bestimmt werden. Dann ist $d = F(S_{gesendet}, S_{empfangen})$ mit F sei das Ausbreitungsmodell. Um gute Ergebnisse zu erzielen sollten jeweils möglichst gleichförmige mobile Einheiten und Knoten verwendet werden, denn Lui et al. konnte für WLAN zeigen, dass unterschiedliche Hardware bei Knoten und mobiler Einheit erhebliche Varianz in den gemessenen RSS Werten erzeugt [LGLD+11]. 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.

1.4 Gliederung der Arbeit

[Kommt wenn ich entschieden habe wann in welcher Reihenfolge die Kapitel liegen]

2. Phase 1 - Keine Veränderungen an den APs

In einem ersten Schritt sollten laut Aufgabenstellung keine Veränderungen an den Access Points vorgenommen werden können. Weil die Ortung deshalb auf WLAN basieren muss sind die Knoten im Folgenden immer APs. Die Unveränderlichkeit bedeutet insbesondere auch, dass keine Messwerte verwendet werden können, die direkt am AP gemessen werden und nicht als Teil der 802.11 Spezifikation in das dahinterliegende Netzwerk weitergeleitet werden. Da time of arrival Synchronisation und präzise Timer bei den APs vorraussetzt und time difference of arrival nur von den APs gemessen werden kann, eignen sich diese Messgrößen nicht für diese Aufgabenstellung. Received signal strength und roundtrip time of flight können auf der mobilen Einheit gemessen werden, nicht jedoch an den APs, da die in der PHY- beziehungsweise MAC-Schicht gemessenen Werte nicht an weitere Empfänger im Netzwerk propagiert werden. Somit scheidet die direkte Fernlokalisierung mangels Messgrößen aus, es wird stattdessen eine indirekte Fernlokalisierung durchgeführt bei der die mobile Einheit eine Selbstlokalisierung durchführt und das Ergebnis dem Ortungsserver über eine Datenverbindung mitteilt.

2.1 Vorherige Arbeiten

Zunächst werden vorherige Arbeiten behandelt, es werden sowohl wissenschaftliche als auch kommerzielle Lösungen betrachtet.

2.1.1 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 802.11 Spezifikation als Index (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 $P(d) = P(d_0) - 10log_{10}(\frac{d}{d_0})^n - OAF$ mit der Distanz d, der Signalstärke $P(d)$ und der Referenzdistanz $d_0 = 1\text{m}$ zur Bestimmung 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.

2.1.2 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].

2.1.3 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.

2.1.4 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 Baumaschinen 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 [LGLD+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.
Abb. 2.1 zeigt die gemessenen RSSI Werte für die von ihnen getesten Netzwerkkarten mit unterschiedlichen Distanzen, für einige Karten korreliert die empfangene Signalstärke nur sehr schwach mit der Distanz zwischen Knoten 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.

2.2 Grundlagen der 802.11 Spezifikation

802.11 beschreibt eine Form der drahtlosen Datenübertragung mittels Funkwellen [IEEE12g]. Die Spezifikation beschreibt die physische Schicht (PHY-Layer) und den Mediumszugriff (MAC-Layer) 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). Die 1997 erstmals verabschiedete Spezifikation 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 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 (SIFS<PIFS<DIFS<EIFS), anschließend kann gesendet werden.

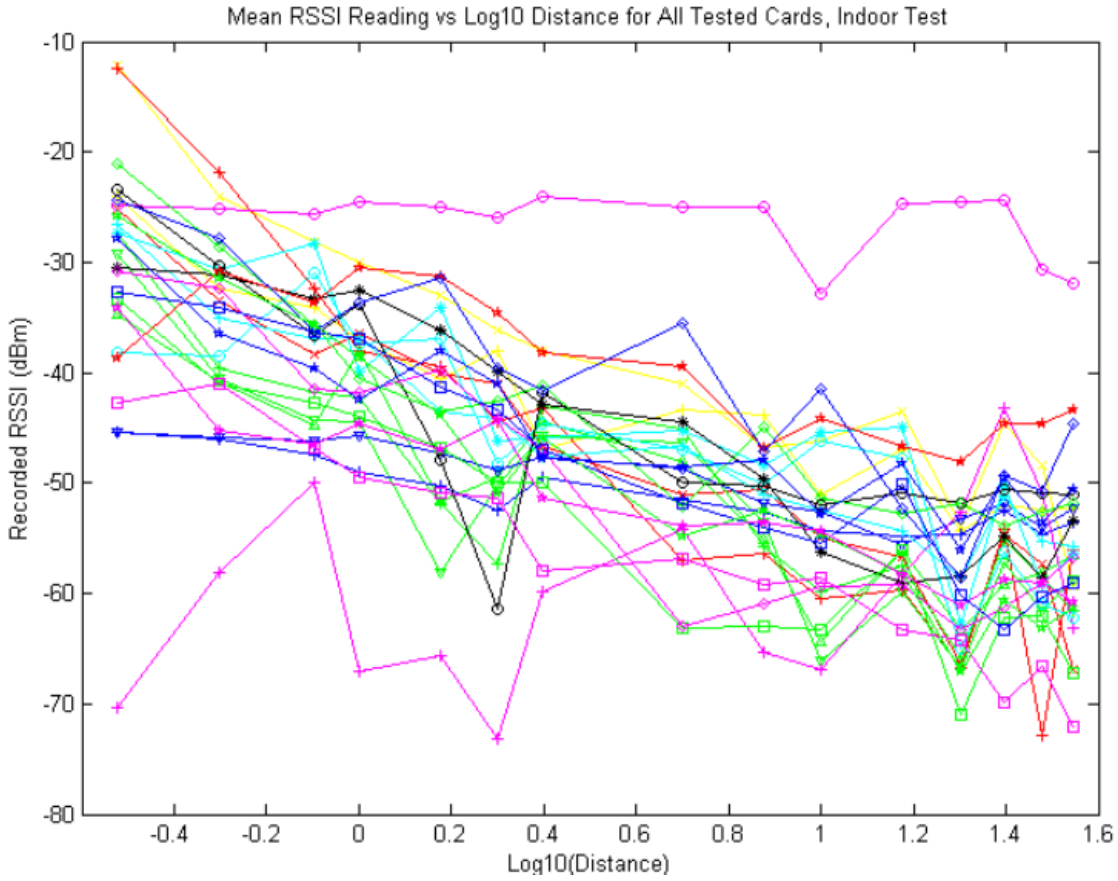

Abbildung 2.1: Gemessener RSSI mit verschiedenen Access Points und Distanzen, aus [LGLD+11].

Tritt trotzdem eine Kollision auf, versucht der Sender es erneut mit einem längeren IFS.
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 addressierte 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 Acknowlegement 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. 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 802.11 Spezifikation gelisteten Management Frames. Ein Management Frame wird durch die Typenbits markiert und durch die Bits für den Subtypen weiter unterschieden.

Tabelle 2.1: Management Frames nach 802.11 [IEEE12f]

Type	Subtype	Beschreibung
00	0000	Association Request
00	0001	Association Response
00	0010	Reassociation Request
00	0011	Reassociation Response
00	0100	Probe Request
00	0101	Probe Response
00	0110	Timing Advertisement
00	0111	Reserved
00	1000	Beacon
00	1001	ATIM
00	1010	Disassociation
00	1011	Authentification
00	1100	Deauthentification
00	1101	Action
00	1110	Action No Ack
00	1111	Reserved

2.2.2 Join

Aus den entdeckten APs kann nun einer ausgewählt werden, um seinem Netzwerk (BSS) beizutreten [IEEE12b]. Es kann zwar geschehen, dass mehrere APs eines Netzwerks entdeckt wurden, eine Station kann jedoch zu jedem Zeitpunkt nur mit einem AP assoziert sein.
Um einem Netzwerk beizutreten muss sich die Station zunächst authentifizieren. Dieser Vorgang wird über einen Authentication Frame (siehe Tabelle 2.1) initiiert [IEEE12c]. 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 [IEEE12d]. 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). [evtl IGMP?]
Ist eine Station assoziert 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 Reassociation 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 [IEEE12e]. 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 assoziert 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 Reassiciation Response. Abschließend wird dem Netzwerk die neue Assoziation mittels ARP mitgeteilt.

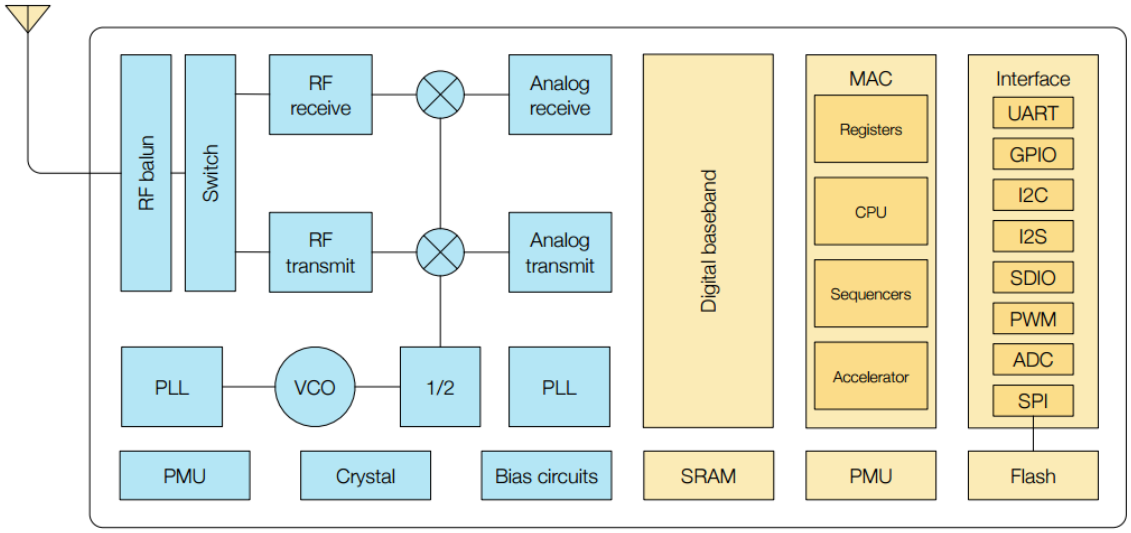

Abbildung 2.2: Blockdiagramm des ESP8266, aus [Espr17]

2.3 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 diverse andere, kabelgebundene Kommunikationsstandards wie zum Beispiel GPIO, I2C und SPI, siehe Abb. 2.2.
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 ESP12-S und das ESP12-F. Das neuere ESP12-F sollte eine höhere Reichweite bei der Funkübertragung entfalten, dies wird noch Gegenstand eines Experiments sein. Abb. 2.3 zeigt die beiden Module nebeneinander, die unterschiedlichen Antennenformen sind deutlich zu erkennen.
Espressif gibt im Datenblatt auch Aufschluss über den Energieverbrauch des ESPs, siehe dazu Abb. 2.4. Für die Prototypenentwicklung wird ein ESP12-S Modul auf einem Adafruit Feather Huzzah verwendet, dieses stellt mit dem CP2104 eine serielle Schnittstelle zum ESP her, reguliert die Spannung für das Modul auf 3,3V und bringt den 2mm Pinabstand des ESP12-S Moduls auf die für Breadboards üblichen 2,5mm. Das Adafruit Feather Huzzah mit ESP12-S ist in Abb. 2.3 links abgebildet.

Abbildung 2.3: Vergleich der Antennen, links: ESP12-S verbaut auf einem Adafruit Feather Huzzah, rechts: ESP12-F

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-sleep①	-	15	-	mA
Light-sleep②	-	0.9	-	mA
Deep-sleep③	-	20	-	μA
Power Off	-	0.5	-	μA

Abbildung 2.4: Energieverbrauch des ESP8266 bei verschiedenen Operationen, aus [Espr17]

2.3.1 ESP8266 Arduino Core

Eine einfache Möglichkeit den ESP8266 zu programmieren stellt die bekannte Arduino IDE dar [Mass17a].
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 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 [Mass17b]. 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.

2.3.2 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 geflasht.
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 ESP 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.

2.4 WiFi-LSS 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 eines WiFi-LLS-Tags 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-Verbindung vermieden.
Zunächst muss sich das Tag dem Netzwerk beitreten (Join) und anschließend einen Scan ausführen und ein UDP-Paket versenden. 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 ESP automatisch aufgerufen, wenn er keine Aufgaben zu erledigen hat. Er kann aber auch manuell aufgerufen werden, beides wurde getestet.
Einige Parameter können gewählt werden: Insbesondere die Intervallzeit bestimmt, wie oft der ESP aktiv ist und damit auch wie viel Energie er verbraucht. Für Mitarbeiter im Tunnel kann von einer Maximalgeschwindigkeit von 30 km/h ausgegangen werden, diese wird durch Schienen- oder Lastkraftfahrzeuge erreicht. Die maximale Reichweite des ESP12-S beziehungsweise ESP12-F Moduls ist nicht bekannt und muss noch bestimmt werden, sie wird zwischen 50 und 100 Metern angenommen. Es wurde daher ein Intervall von 5 Sekunden gewählt, in dieser Zeit bewegt sich ein Mitarbeiter bei 30 km/h ca. 42 m.
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 4 nichtüberlappenden Kanäle zu verteilen. Eine Implementierung die nur einen Kanal scannt tritt deshalb nur außer Konkurenz an.
Tabelle 2.2 zeigt den gemessen Energieverbrauch der Implementierungen in Arduino und C, jeweils mit und ohne manuell aufgerufenen `light_sleep` und eine Implementierung, die nur einen Channel scannt. Für die Tests wurde das Adafruit Feather Huzzah verwendet, es wurde mit 5 V aus einer USB-Powerbank mit Energie versorgt, der Verbrauch wurde mit einem dazwischen geschalteten USB-Power-Meter gemessen.
Die Werte wurden stationär in einer Mietwohnung in einem 5-stöckigen Wohnhaus und damit nicht unter realen Bedingungen aufgezeichnet. Unter realen Bedingungen finden durch die Bewegung des Mitarbeiters regelmäßig Reassizationen statt, im Gegenzug liefert ein Scan in einem Tunnel weniger Ergebnisse als in einem Wohnhaus. Ein Test unter realen Bedingungen ist im Abschnitt **??** zu finden, er beinhaltet jedoch nur noch die Implementierung in C mit manuellem `light_sleep` und einem Scan auf allen 4 Kanälen.

Tabelle 2.2: Energieverbrauch WiFi-LLS-artiger Tags

SDK	manueller `light_sleep`	Anzahl Kanäle	Spannung in V	∅ Verbrauch in mA	∅ Verbrauch in mW
Arduino Core	Nein	4	5	22	110
Arduino Core	Ja	4	5	21	105
ESP Open SDK	Nein	4	5	19	95
ESP Open SDK	Ja	4	5	19	95
ESP Open SDK	Ja	1	5	6,5	32,5

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 Channel eine signifikante Senkung des Energieverbrauchs nach sich zieht, die Scan-Funktion ist also der Hauptverbraucher.
Um die finale Laufzeit zu bestimmen muss die Kapazität des Akkus bekannt sein. Ein Lithium-Polymer Akku hat eine Nennspannung von $3,7\ V$, seine Kapazität wird in

mAh angegeben. Ein beispielhafter 1400 mAh Lithium-Polymer Akku liefert folglich 1400 $mAh * 3,7\ V = 5180\ mWh$. Die Laufzeit beträgt dann 5180 $mWh/95\ mW \approx 54,53\ h$, also etwas mehr als 2,25 Tage.

2.5 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 das Tag einem Bereich zuzuordnen. Da dem Tag, welches im Sinne der 802.11 Spezifikation als Station arbeitet, die IP-Adresse seines Netztzugangs (Gateway) bekannt sein muss, diese Information kann dann als Ortungsinformation verwendet werden. Die IP-Adresse wird nun zusammen mit der eigenen MAC-Adresse als Identifikator als String kodiert und per UDP an den Ortungsserver versendet.
Tabelle 2.3 zeigt den gemessen Verbrauch der Implementierungen in Arduino und C, jeweils mit und ohne manuell aufgerufenen `light_sleep`.

Tabelle 2.3: Energieverbrauch der Bereichsortungstags

SDK	manueller `light_sleep`	Spannung in V	∅ Verbrauch in mA	∅ Verbrauch in mW
Arduino Core	Nein	5	14	70
Arduino Core	Ja	5	7	35
ESP Open SDK	Nein	5	6	30
ESP Open SDK	Ja	5	5,5	27,5

Der Verbrauch liegt wie erwartet unter dem der WiFi-LLS Implementierung, sogar unter der Implementierung, die nur einen Kanal scannt. Wird der selbe 1400 mAh beziehnungsweise 5180 mWh Lithium-Polymer Akku angenommen liegt die Laufzeit diesmal bei 5180 $mWh/27,5\ mW \approx 188,36\ h$, also bei fast 8 Tagen.
Als weitere Optimierung könnte ein Tag nur dann senden, wenn eine Reassoziation stattgefunden hat. Die mangelnde Transportsicherheit von UDP macht dieses Vorgehen jedoch riskant, wenn das Paket verloren geht wird kein Bereichswechsel erkannt. Um wieder eine begrenzte Transportsicherheit zu erhalten kann entweder das UDP-Paket mehrfach versandt werden; auch 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).
Da der Verbrauch nun stark von der Anzahl der Reassoziationen abhängt sind im gegeben, stationären Testszenario keine aussagekräftigen Ergebnisse möglich. Dennoch sollen die Tests eine 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 2.4 gezeigten Implementierungen mit ihr erstellt, der manuelle `light_sleep` ist immer aktiv.

Tabelle 2.4: Energieverbrauch der verbesserten Bereichsortungstags

Transportsicherung	Spannung in V	∅ Verbrauch in mA	∅ Verbrauch in mW
Ohne	5	2	10
Dreifach UDP	5	1,4	7
Zusatzintervall 30s	5	2,62	13,1
Zusatzintervall 60s	5	1,6	8
TCP (halten)	5	1,2	6
TCP (schließen)	5	1,07	5,35

Der Energieverbrauch sinkt durch das Einsparen von Sendevorgänge deutlich. Für das Tag, das eine TCP-Verbindung öffnet, dem Ortungsserver seine Assoziation mitteilt und die Verbindung dann wieder schließt, ergibt sich für den angenommenen 5180 mWh Lithium-Polymer Akku eine Laufzeit von $5180\ mWh/5,35\ mW \approx 968,22\ h$. Dies entspricht gut 40 Tagen mit einer Akkuladung, diese werden jedoch in einem Szenario ohne Reassoziation erreicht. Die tatsächliche Laufzeit muss unter realen Umständen gemessen werden, siehe dazu Abschnitt **??**. Da die zusätzlichen Reassoziationen jedoch auch die vorherigen Implementierungen betreffen kann man schließen, dass Tags für die Bereichsortung durch die Ausnutzung von Assoziation und Reassoziation einen erheblich reduzierten Energieverbrauch gegenüber herkömmlichen Tags für die Trilateration aufweisen.

3. Phase 2 - Mit Softwareänderungen an den APs

Alle bisher betrachteten Lösungen arbeiteten mindestens auf Protokollebene 4 des OSI-Modells. Die Lösungen benutzten TCP oder UDP um ihre Position dem Ortungsserver mitzuteilen und mussten somit dem Netzwerk beitreten, da es sonst nicht möglich gewesen wäre mit dem Ortungsserver zu kommunizieren.
Da nun die Software der Access Points veränderbar ist, ergeben sich neue Freiheiten bezüglich der Kommunikation. Da die Zielsetzung in Abschnitt 1.1 jedoch die Einschränkung macht, dass ohne Authorisierung nicht mit dem Ortungsserver kommuniziert werden darf können die in Kapitel 2 vorgestellten Verfahren der indirekten Fernlokalisierung nicht von diesen Freiheiten Gebrauch machen.
Stattdessen werden in diesem Kapitel Verfahren der direkten Fernlokalisierung betrachtet. Die Veränderbarkeit der Software des AP wird dazu genutzt auf diesem Messgrößen zu ermitteln und dann über eine Datenverbindung an den Ortungsserver übermittelt. Da für die APs keine Beschränkungen bezüglich des Energieverbrauchs vorliegen, kann die Verbindung zu Ortungsserver regulär auf Schicht 4 oder 5 aufgebaut werden. Für die Kommunikation zwischen Tag und AP sollte jedoch auf niedrigere Protokollebenen ausgewichen werden, um den Energieverbrauch des Tags zu senken.

3.1 Vorherige Arbeiten

Zunächst werden vorherige Arbeiten behandelt, sie wurden so ausgewählt, dass möglichst viele Messgrößen abgedeckt sind.

3.1.1 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 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 4 UDP-Pakete pro Sekunde aus, an den Knoten 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. Bezüglich 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 gemessen 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 einem 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 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 Knoten, 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.

3.1.3 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 802.11 Protokolls, da auf diesen aber üblicherweise

kein Zugriff besteht messen sie TOF in der darüber liegenden MAC-Schicht.
Ein Knoten sendet einen Beacon aus und protokolliert die Sendezeit, die mobile Einheit empfängt den Beacon Frame, protokolliert die Empfangszeit und die Sendezeit der gesendeten Antwort, der Knoten sichert die Empfangszeit der Antwort. Nun sendet die mobile Einheit die zwei gespeicherten Zeitstempel an den Knoten, der mit diesen die Verarbeitungszeit auf der mobilen Einheit berechnen kann, um dann die Distanz $d = c * (\frac{t_{empfangen} - t_{gesendet} - t_{Verarbeitung}}{2})$ 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 Knoten 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 $100ns$ bei $c = 299.792.458m/s$ bereits einen Fehler von $30m$ 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 802.11 (CSMA/CA) abhängen.

3.1.4 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 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 $100m^2$ gestaltet worden und müsste auf Baustellengröße und die Verfolgung mehrerer dutzend 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 aufhalten. Auch ist es ohne Identifikation schwerer Fehler zu erkennen. Wird fälschlicherweise angezeigt, dass sich noch eine Peron 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 eine 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.

3.2 RADAR Implementierung

Ein RADAR Tag versendet alle 0,25s ein 6 Byte langes UDP-Paket, der RSSI der Übertragung wird dann auf dem AP gemessen. Das Sendeintervall 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 5 Sekunden gewählt, in dem sich ein Mitarbeiter maximal 42 m bewegt. Tabelle 3.1 zeigt den gemessenen Verbrauch der Implementierung für RADAR Tags jeweils in Arduino und C, mit unterschiedlich langen Sendeintervallen mit und ohne manuellen `light_sleep`.

Tabelle 3.1: Energieverbrauch RADAR-artiger Tags

SDK	manueller `light_sleep`	Sende-intervall in s	Spannung in V	∅ Verbrauch in mA	∅ Verbrauch in mW
Arduino Core	Nein	0,25	5	40	200
Arduino Core	Ja	0,25	5	40	200
Arduino Core	Nein	5	5	8	40
Arduino Core	Ja	5	5	8	40
ESP Open SDK	Nein	0,25	5	40	200
ESP Open SDK	Ja	0,25	5	38	190
ESP Open SDK	Nein	5	5	6	30
ESP Open SDK	Ja	5	5	5	25

Der Energieverbrauch einer Lösung, die 4 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 die projezierte Laufzeit jedoch stark. Für den 1400 mAh Akku ergibt sich eine Laufzeit von 5180 $mWh/190\ mW \approx 27,26\ h$ für ein Sendeintervall von 0,25s, hingegegen ergibt sich für ein Sendeintervall von 5s eine Laufzeit von 5180 $mWh/25\ mW = 207,2\ h$. Das Tag mit dem längeren Sendeintervall hält somit 7,6 mal so lang wie das andere mit einer Akkuladung durch. Dennoch ist der Verbrauch nur marginal geringer als der der Bereichsortungstags ($27,5\ mW$).
Die in Abschnitt 2.5 besprochenen Optimierungen (nur bei AP-Wechsel senden) können auch für die RADAR Implementierung verwendet werden, das RADAR-artige Tag ist dann aber in seiner Implementierung bis auf den Inhalt des UDP-Pakets identisch mit dem Bereichsortungstag. Der Energieverbrauch sollte sich somit kaum unterscheiden und ein System mit RADAR-artigen Tags benötigt Veränderungen

der Software der APs, ein System mit Bereichortungstags ist deshalb vorzuziehen.

3.3 Time-of-Flight Implementierung

[Kommt noch Text, wird nicht implementiert, weil schon theoretisch teurer als RADAR]

3.4 Anpassungen für Bereichsortung

RADAR versendet immer noch UDP Pakete und arbeitet damit auf Schicht 4 (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 1 (PHY) gemessen. Grundsätzlich kann aufgrund der möglichen Änderungen am AP ein beliebiges Paket mit einer speziellen Kennung versendet und vom AP als Positionsmitteilung des Tags erkannt werden.
Ein Sendevorgang, der nur Schicht 1 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 beeinträchtigt wurde. Es sollte deshalb nicht auf Schicht 1 gearbeitet werden.
Stattdessen sollte Schicht 2 (MAC) des OSI-Modells verwendet werden. Da 802.11 für den Mediumszugriff eine Kollisionsvermeidung (CSMA/CA) verwendet muss das Tag 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 1, stellt dafür aber die Verfügbarkeit des Mediums (der Frequenz) für die übrigen Teilnehmer sicher.
Um die Änderungen an der Software des AP jedoch gering zu halten wurde der Probe Request als zu sendenden 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 sein solches Protokoll für Probe Requests und Beacons im Zuge eine *Rouge Client/AP Detection* an [LANC17].
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 2 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 muss 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 Möglichenkeiten, wie etwa unterstützte Standards und Datenraten, da das Tag 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 ESP und seinen Speicher fast vollständig ab, nach Ablauf der angegebenen Schlafzeit wird Pin 16 mit der negativen Seite der Stromquelle verbunden. Damit der ESP wieder aufwacht muss Pin 16 mit dem Reset Pin (RST) verbunden werden, bei einem Reset initialisiert der ESP neu. Bei einer Lösung auf einer höheren Schicht würde dies dazu führen, dass das Tag versucht dem Netztwerk erneut beizutreten, hingegen kann bei einer Lösung auf Schicht 2 sofort gesendet und danach wieder geschlafen werden.
Tabelle 3.2 zeigt den Energieverbrauch der Implementierungen jeweils mit mauell herbeigeührtem Schlafzustand, das Sendeintervall liegt bei konstant 5 Sekunden.

Tabelle 3.2: Energieverbrauch Probe Request Tags

SDK	manueller Schlafzustand	Spannung in V	∅ Verbrauch in mA	∅ Verbrauch in mW
Arduino Core	Ohne	5	40,5	202,5
Arduino Core	light_sleep	5	5	25
Arduino Core	deep_sleep	5	4,25	21,25
ESP Open SDK	Ohne	5	8,33	41,66
ESP Open SDK	light_sleep	5	3	15
ESP Open SDK	deep_sleep	5	0,8	4

Zu erkennen ist, dass die Verwendung des `deep_sleep` zu einem geringeren Verbrauch führt. Allerdings benötigt das mit dem Arduino Core programmierte Tag deutlich im Vergleich zu dem mit der ESP Open SDK programmierten Tag deutlich länger zum Starten. Es verbraucht deshalb sogar mehr Energie als die Bereichsortungstags aus Abschnitt 2.5, die Implementierung mit der ESP Open SDK verbraucht aber weniger Energie als diese. Für den 1400 mAh Akku ergibt sich eine Laufzeit von 5180 $mWh/4\ mW \approx 1295\ h$ für die Implementierung in C mit `deep_sleep`. Dies entspricht einer Laufzeit von fast 54 Tagen. Hinzu kommt, dass sich der Verbrauch dieser Tags nicht durch die Bewegung des Trägers erhöht, da keine Reassoziationen stattfinden. Um die in Abschnitt 1.2 geforderten 6 Monate Laufzeit zu erreichen werden in Abschnitt **??** weitere Verbesserungen besprochen.

Literaturverzeichnis

[AiRI17a] AiRISTA Flow. Ekahau A4 Asset Tag. https://www.airistaflow.com/wp-content/uploads/2016/07/AiRISTAFlow_Ekahau_RTLS_A4_DS.pdf, 2017.

[AiRI17b] AiRISTA Flow. Ekahau B4 Badge Tag. https://www.airistaflow.com/wp-content/uploads/2016/09/AiRISTAFlow_-Ekahau_B4_DS.pdf, 2017.

[AiRI17c] AiRISTA Flow. Offizielle Website von AiRISTA Flow. https://www.airistaflow.com/#, 2017.

[ANSSY13] H. Abdel-Nasser, R. Samir, I. Sabek und M. Youssef. MonoPHY: Monostream-based device-free WLAN localization via physical layer information. In *Wireless communications and networking conference (WCNC), 2013 IEEE*. IEEE, 2013, S. 4546–4551.

[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.

[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, 2017.

[ESP 17] ESP Community. ESP Opnen SDK. https://github.com/pfalcon/esp-open-sdk, 2017.

[Espr17] Espressif Systems IOT Team. ESP8266EX Datasheet. http://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf, 2017.

[Freu15] J. Freudiger. How talkative is your mobile device?: an experimental study of Wi-Fi probe requests. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2015, S. 8.

[Frie17] L. Fried. Adafruit Feather HUZZAH ESP8266 - Using Arduino IDE. https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/using-arduino-ide, 2017.

[Goog17] Google. Offizielle Website des Android-Projekts. https://www.android.com/intl/de_de/phones/, 2017.

[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.

[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. 10.3 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.

[IEEE12c] 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.

[IEEE12d] 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.

[IEEE12e] 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.

[IEEE12f] 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.

[IEEE12g] IEEE Computer Society. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE. 2012.

[LANC17] LANCOM Systems. Rouge-Detection-Funktion. https://www.lancom-systems.de/docs/LCOS-Refmanual/9.10-Rel/DE/topics/wlanmonitor_rogue_detection.html, 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.

[LGLD+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.

[Mass17a] T. I. D. A. M. Massimo Banzi, David Cuartielles. Arduino IDE. https://github.com/arduino/Arduino/, 2017.

[Mass17b] T. I. D. A. M. C. K. I. G. E. S. P. A. R. H. C. R. Massimo Banzi, David Cuartielles. ESP8266 Arduino Core. https://github.com/esp8266/Arduino, 2017.

[MiSA03] A. Mishra, M. Shin und W. Arbaugh. An empirical analysis of the IEEE 802.11 MAC layer handoff process. *ACM SIGCOMM Computer Communication Review* 33(2), 2003, S. 93–102.

[MKML06] K. Muthukrishnan, G. Koprinkov, N. Meratnia und M. Lijding. Using time-of-flight for WLAN localization: feasibility study. 2006.

[NLLP04] L. M. Ni, Y. Liu, Y. C. Lau und A. P. Patil. LANDMARC: indoor location sensing using active RFID. *Wireless networks* 10(6), 2004, S. 701–710.

[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.

[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.

[Stan17a] Stanley Healthcare. AeroScout: Healthcare Asset Tracking & Management. https://www.stanleyhealthcare.com/solutions/health-systems/supply-chain-asset-management/asset-management, 2017.

[Stan17b] Stanley Healthcare. AeroScout: Staff Assist for Improved Security. https://www.stanleyhealthcare.com/solutions/health-systems/security-protection/staff-security, 2017.

[Stan17c] Stanley Healthcare. T14 Patient and Staff Badge. https://www.stanleyhealthcare.com/sites/stanleyhealthcare.com/files/documents/T14%20Badge%20Data%20Sheet.pdf, 2017.

[Torr84] D. J. Torrieri. Statistical theory of passive location systems. *IEEE transactions on Aerospace and Electronic Systems* (2), 1984, S. 183–198.

[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.