Lokalisierung von IoT Devices mithilfe von Funktechnologien

Pascal Westenweller

Bachelor-Thesis

Studiengang Technische Informatik

Fakultät für Informationstechnik

Technische Hochschule Mannheim

30.04.2025

Durchgeführt bei der Firma Exxeta AG, Karlsruhe

Betreuer: Prof. Dr. Eckhart Körner, Technische Hochschule Mannheim

Zweitkorrektor: Dr. Daniel Schell, Exxeta AG

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Hochschule Mannheim öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Mannheim, 30.04.2025

Pascal Westenweller

Abstract

Diese Arbeit beschäftigt sich mit der Lokalisierung eines IoT Gerätes mithilfe von Funktechnologien. Ziel der Arbeit ist es ein System zu erstellen, welches in der Lage ist ein Gerät im Innenraum möglichst präzise zu lokalisieren. Verwendet wird hierbei eine drahtlose Technologie, welche Daten bereitstellt, ein Gerät, welches die Daten auslesen kann und die dann mithilfe einer Windows Applikation verarbeitet werden, um die Position des Gerätes zu bestimmen. Technologischer Hintergrund, sowie Modell und Implementierung einer konkreten Lösung sind Bestandteil dieser Arbeit.

Inhalt

1 Einleitung

1.1 Motivation

Positionserkennung in Innenräumen ist noch immer ein Problem [1]. Über die Jahre wurden viele Technologien entwickelt, um Positionierung mit sehr hoher Genauigkeit zu ermöglichen. Die Technologien reichen von Funktechnologien bis hin zu optischen Systemen. Die verschiedenen Systeme bieten unterschiedliche Vor- und Nachteile darunter die Genauigkeit. Funktechnologie basierte Systeme haben Genauigkeiten im Meterbereich bis hin zum Zentimeterbereich. Die meisten Funktechnologie basierten Systeme nutzen den Fakt, dass die Signalstärke in eine Distanz umgerechnet werden kann. Dies wird aber problematisch, wenn das Signal nicht auf direktem Weg den Empfänger erreichen kann. Viele Systeme haben daher Probleme mit komplexen Innenräumen, in welchen Wände oder Gegenstände Signale reflektieren, absorbieren oder interferieren. Der Vorteil WLAN basierter Systeme ist, dass diese meist kaum weitere Hardware benötigen, da viele Unternehmen, öffentliche Plätze oder private Haushalte bereits WLAN-Router installiert haben. Daher werden nur Empfänger benötigt. Heutzutage gibt es bereits WLAN fähige Mikrokontroller wie den ESP32 für sehr geringe Kosten. Auch Smartphones haben typischerweise WLAN-Chips integriert und könnten daher auch ohne weitere Kosten in diesem System genutzt werden. Diese Arbeit beschreibt eine neue Methode WLAN-Signale zu nutzen, um die Positionserkennung eines Gerätes im Innenraum zu ermöglichen.

1.2 Umfeld der Arbeit

Die Arbeit wurde mit der *Exxeta AG* in Karlsruhe erarbeitet. Exxeta ist ein international tätiges IT-Beratungs- und Softwareunternehmen mit Hauptsitz in Karlsruhe. Exxeta wurde im Jahr 2001 gegründet und beschäftigt zum heutigen Standpunkt über 1200 Mitarbeiter:innen. Das Unternehmen hat Standorte europaweit und arbeitet eng mit Großunternehmen und Mittelstandskunden in den Bereichen Business Consulting, IT-Architektur und individueller Softwareentwicklung zusammen. Deren Branchenbereiche reichen von Finanzwirtschaft über Automotiv, und Energie [2]. Die Exxeta stellte einen mehrere Betreuer als Ansprechpartner bereit. Von der technischen Hochschule

Mannheim hat sich freundlicherweise Prof. Dr. Eckhart Körner bereit erklärt diese Arbeit zu unterstützen.

1.3 Ziele der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung und Evaluierung eines WLAN-basierten Positionierungssystems zur präzisen Positionsbestimmung eines Empfängers in Innenräumen. Im Weiteren werden unterschiedliche Unterziele genannt, die in dieser Arbeit erarbeitet werden. Diese Ziele sollen einen möglichst breiten Überblick über die Eigenschaften des Systems und dessen Vor- und Nachteile zeigen.

1.3.1 Genauigkeit

Die Genauigkeit soll beschrieben, wie weit die vom System berechnete Position von der Tatsächlichen Position des Mikrokontrollers entfernt ist. Als Maß wird die euklidische Distanz genutzt, welche sich über die Formel

$$d = \sqrt{(q_1 - p_1)^2 + \cdots + (q_n - p_n)^2}$$

für die Punkte p und q berechnen lässt. Das Ziel des Systems ist diesen Abstand zu minimieren. Im deutschen Sprachgebrauch beschreibt eine „hohe Genauigkeit“ eine kleine Distanz. Daher wird in dieser Arbeit die Genauigkeit als „besser“ betitelt, falls die Distanz kleiner wird. Diese wird als „schlechter“ beschrieben, sollte die Distanz größer werden.

1.3.2 Stabilität

Stabilität soll bedeuten, dass Messungen unter gleichen internen Bedingungen zu identischen oder nahezu identischen Ergebnissen führen. Das System soll die Genauigkeit an einer Position ohne eine Veränderung der Daten nahezu konstant halten, egal zu welcher Zeit man diese Messung durchführt.

1.3.3 Störfaktoren

Störfaktoren bezeichnen äußere Einflüsse auf das System, welche die Genauigkeit oder die Stabilität verringern lassen können. Auf WLAN basierte Positionierungssysteme sind häufig anfällig für Störfaktoren wie z.B. Reflexionen, Interferenzen oder Absorp-

tion [3]. Daher soll das System auf mögliche Störfaktoren untersucht werden. Es werden anschließend Methoden gezeigt, um den Einfluss dieser Faktoren zu minimieren oder gar zu beseitigen.

1.3.4 Performance

Die Performance soll Auskunft darüber geben, wie schnell die Position des Mikrokontrollers berechnet werden kann. Es gibt viele Variablen im System, die die Dauer dieser Berechnung beeinflussen können. In Kapitel 5 werden Verbesserungen zur Grundbasis des Systems gezeigt und auch deren Einfluss auf die Performance. Ebenfalls wird genannt, wie lange die Dauer ist, bis das System betriebsfähig ist. Das System erfordert eine Ausmessung des Raumes, welche je nach Raum und Variablen eine unterschiedlich lange Zeit beansprucht. Es werden Vorschläge gezeigt, die einen „tradeoff“ zwischen Genauigkeit und Dauer des Aufbaus bieten.

1.3.5 Kosten

Das System soll in der Lage sein viele Objekte im Raum bestimmen zu können. Viele System, wie z.B. optische Systeme benötigen daher speziell angefertigte Hardware, welche in der Lage ist, die Objekte zu lokalisieren. Das hier vorgeschlagene System basiert auf Hardware, welche bereits in den meisten Haushalten und Büroeinrichtungen gefunden werden, kann [4] [5]. Daher besteht die Möglichkeit die Kosten gering zu halten und soll daher auch ein Ziel dieser Arbeit werden.

1.4 Struktur der Arbeit

Die Arbeit ist in folgende Kapitel strukturiert:

Kapitel 1 nennt die Motivation und Ziele der Arbeit

Kapitel 2 beschreibt Probleme aktueller Verfahren und einen Ansatz für Verbesserungen

Kapitel 3 führt Grundlagen und Begrifflichkeiten der Arbeit ein

Kapitel 4 beschäftigt sich mit der Umsetzung des Systems

Kapitel 5 zeigt Ergebnisse des Systems und der Verbesserungen

Kapitel 6 gibt eine Zusammenfassung der Arbeit

2 Lösungsansätze für die Lokalisierung von IoT Devices

Es gibt bereits Systeme, welche die Innenraum Lokalisierung ermöglichen. Dieses Kapitel erläutert die Vor- und Nachteile dieser Systeme. Des Weiteren wird beschrieben, wie das System in der hier beschriebenen Arbeit in Hinsicht auf die genannten Ziele Verbesserungen bringen soll.

2.1 Probleme aktueller Verfahren

Um die Position von Geräten im Raum zu bestimmen, wurden bereits viele Verfahren entwickelt. Diese reichen von Systemen, welche optischen Sensoren benutzen bis hin zu Systemen, welche Signaldaten verarbeiten. Da diese Systeme voneinander sehr unterschiedlich funktionieren können werden nun ein paar Systeme vorgestellt und auf deren Vor- und Nachteile eingegangen [6].

Optische Systeme: Diese Systeme nutzen Licht, um Objekte im Raum zu identifizieren und zu finden. Der Vorteil dieser Systeme ist eine hohe Genauigkeit und schnelle Abtastraten. Die Nachteile dieser Systeme sind häufig hohe Kosten und die Anforderung der direkten Sichtbarkeit des Zielobjekts.

Funktechnologien: Bei diesen Systemen werden Funksignale ausgelesen und verarbeitet. Hier gibt es sehr viele Ansätze, die meisten basieren auf der Triangulation. Hier werden drei Signalquellen verwendet und der Empfänger kann mithilfe von Signalstärken, AoA (Angle of Arrival) oder ToA (Time of Arrivel) messen, wie weit jede Signalquelle entfernt ist. Der Vorteil besteht darin, dass Funktechnologien wie WLAN verwendet werden können, welche meist geringe oder gar keine Anschaffungskosten haben, da diese Technologie bereits weit verbreitet ist. Die Nachteile dieser Systeme sind meist eine schlechtere Genauigkeit, meist im Meterbereich und Hindernisse können die Signale beeinflussen.

Ultraschall: Schallwellen außerhalb des menschlichen Gehörsspektrum können genutzt werden, um Distanzen zu messen. Auch hier wird die Triangulation verwendet, um die Position des Gerätes zu bestimmen. Bei diesen Systemen ist der Vorteil eine hohe Genauigkeit, um die 3 cm. Die Nachteile sind Hindernisse und erhöhte Kosten aufgrund der benötigten Hardware.

2.2 Verbesserungsvorschläge in Hinblick auf die Ziele

Das hier vorgestellte System baut auf der WLAN-Technologie auf. Dieser Ansatz bietet den Vorteil, dass es in bereits bestehende WLAN-Systeme integriert werden kann. In Kapitel 4.2 wird eine Hardware beschrieben, welche in der Lage ist, WLAN-Signale zu empfangen und zu verarbeiten und zusätzlich günstig zu erwerben ist. Statt Triangulation wird stattdessen für beliebig viele Punkte im Raum die Signalstärken aufgenommen. Anhand der historischen Daten kann dann die gemessene Signalstärke „gesucht“ werden und Schätzung gegeben werden, an welcher sich das Gerät befinden könnte. Damit wird versucht die Probleme der Triangulation zu umgehen, da die Messpunkte die exakten Signalstärken im Raum repräsentieren. Somit soll es möglich sein, Objekte hinter Gegenständen und Hindernissen erfolgreich zu lokalisieren oder gar hinter Wänden und in anderen Räumen.

3 Grundlagen

Die Arbeit baut auf der Grundlage vieler Technologien auf. Dieses Kapitel soll die nötigsten Grundlagen vermitteln, um die Arbeit besser verstehen zu können.

3.1 Elektromagnetische Wellen

Elektromagnetische Wellen beschreiben die Änderung von elektromagnetischen Feldern in wellenförmiger Form über Raum und Zeit [7]. Elektromagnetische Wellen werden über die Eigenschaften Frequenz f, Wellenlänge λ und Ausbreitungsgeschwindigkeit c beschreiben. Elektromagnetische Wellen benötigen kein Medium, um sich auszubreiten. In einem Vakuum breitet sich diese mit der Lichtgeschwindigkeit aus. In einem anderen Medium verändert sich dessen Ausbreitungsgeschwindigkeit. Der Zusammenhang der Eigenschaften der Wellen lässt sich über die Formel $\lambda = \frac{c}{f}$ beschreiben. Die Energie einer elektromagnetischen Welle wird meist über die Einheit Bel beschreiben. Diese Einheit beschreibt den dekadischen Logarithmus zweier gleichartiger Energiegrößen [8]. Die Einheit wurde nach Alexander Graham Bell benannt. Ein Bel wird definiert über: $B = log_{10}\left(\frac{P_1}{P_2}\right)$. Gebräuchlicher ist das Dezibel kurz dB, welches ein Zehntel eines Bels beträgt: $1\,dB = \frac{1}{10}B$.

3.2 WLAN

Der Begriff WLAN ist eine Abkürzung für den Begriff Wireless Local Area Network. Es ist eine Technologie zum drahtlosen Übertragen von Daten in einem lokalen Netzwerk. In diesem Dokument wird der Begriff WLAN für den *IEEE*-Standard 802.11 genutzt. Die WLAN-Technologie arbeitet in verschiedenen Frequenzbereichen [9]. Die häufigsten zwei Frequenzen sind 2.4 Ghz und 5 Ghz und fallen daher in den Bereich der Mikrowellen. Diese Spektren werden wiederrum in Bänder zerlegt. Die Hardware, welche in dieser Arbeit verwendet wird, kann nur Frequenzen auf dem 2.4 Ghz Band empfangen und senden. In Europa wird das 2.4 Ghz Spektrum in 13 Bänder zerlegt:

Tabelle 1: 2.4-Ghz WLAN Frequenzbänder

Kanal	Frequenzbereich
1	2399,5 - 2424,5 MHz

2	2404,5 - 2429,5 MHz
3	2409,5 - 2434,5 MHz
4	2414,5 - 2439,5 MHz
5	2419,5 - 2444,5 MHz
6	2424,5 - 2449,5 MHz
7	2429,5 - 2454,5 MHz
8	2434,5 - 2459,5 MHz
9	2439,5 - 2464,5 MHz
10	2444,5 - 2469,5 MHz
11	2449,5 - 2474,5 MHz
12	2454,5 - 2479,5 MHz
13	2459,5 - 2484,5 MHz

Man erkennt, dass sich gewisse Frequenzen überschneiden können. Daher nutzen viele WLAN-Router nur einen Teil dieser Frequenzbereiche. WLAN-Geräte sind in der Lage die Signalstärke zu messen. Diese wird dann in der Einheit dBm gemessen [10]. Diesen Wert wird Received Signal Strength Indicator (RSSI) genannt [11]. Die Einheit ist definiert als das Dezibel zur Bezugsgröße 1 Milliwatt. Sprich:

$$dBm = 10 \cdot log_{10}\left(\frac{P_{empfangen}}{1mW}\right)$$

In einer Anwendung lässt sich dieser Wert meist als Byte auslesen mit einem Wert zwischen ca. -110 und 20. Um Informationen zwischen Sender und Empfänger austauschen zu können gibt es ein standardisiertes *IEEE* 802.11-Protokoll. Ein Router richtet hierbei ein Netzwerk ein und macht es für andere Geräte zugänglich. Dieses Netzwerk erhält einen „Namen“, einen Service Set Identifier (SSID) [12]. Zur Kommunikation werden Pakete versendet zwischen Sender und Empfänger. Für diese Arbeit relevant sind die Management Pakete und „Beacon“ Frames [13]. Diese werden üblicherweise in festen Abständen von WLAN-Routern versendet, meist in Intervallen von 100 ms. Ein Gerät kann diese empfangen und Informationen auslesen unter anderem die SSID und RSSI. Ein Gerät kann aber auch sogenannte „Probe-Requests“ [14] senden mit einer Netzwerk SSID. Falls ein Router mit passender SSID diese Nachricht empfängt, sendet dieser eine entsprechende „Probe-Response“. Diese Methode zum Auslesen der

Signalstärke auf Wunsch des Gerätes wird in der Arbeit in Kapitel 4 genauer beschrieben.

3.3 Omnidirektionale Antennen

Um ein WLAN-Signal im Raum zu empfangen, werden Antennen verwendet. Diese können unter anderem direktional oder omnidirektional sein. Direktionale Antennen sind dafür ausgelegt Signale aus einer bestimmten Richtung zu empfangen [15]. Omnidirektionale Antennen hingegen sind dafür konzipiert Signale aus allen Richtungen zu empfangen. Dies kommt meist mit dem Nachteil, dass Signale aus ungewünschten Richtungen aufgenommen werden können. Da WLAN in den nicht lizenzierten Frequenzbändern betrieben wird, kann es sein, dass Signale anderer Geräte da empfangene Signal stören können. Der Vorteil omnidirektionaler Antennen besteht darin, dass keine Ausrichtung zum Sender nötig ist. Dies ist besonders hilfreich, wenn man keine feste Positionierung des Empfängers gewährleisten kann oder die Positionen der Sender unbekannt sind.

3.4 TCP/UDP

Sowohl TCP als auch UDP sind beides Transportprotokolle der Internet-Protokollfamilie (IP) [16] [17]. TCP (Transmission Control Protocol) ist ein verbindungsorientiertes Protokoll. Die Aufgaben des Protokolls sind [18]:

- **Segmentierung der Datenströme**
  Teilt große Datenmengen in Segmente auf und setzt diese bei Empfang wieder korrekt zusammen
- **Verbindungsmanagement**
  Stellt eine Verbindung zwischen zwei Endpunkten her und baut diese auch wieder ab
- **Fehlerbehandlung**
  Überprüft, ob Datenpakete erfolgreich ihr Ziel erreicht haben und sendet diese bei Verlust erneut
- **Flusssteuerung**
  Lastet die Übertragungsstrecke dynamisch aus
- **Anwendungsunterstützung**
  Adressiert Anwendungen mithilfe von Port-Nummern

UDP (User Datagram Protocol) hingegen ist ein verbindungsloses Protokoll. Diese übernimmt Aufgaben wie Verbindungsmanagement, Fehlerbehandlung und Flusssteuerung und Segmentierung nicht. Der Vorteil dieses Protokolls ist die einfachere Umsetzung im Gegensatz zu TCP. Anwendungen können mithilfe von Socket APIs auf die Funktionalitäten der Übertragungsprotokolle zugreifen. Die Umsetzung und der Nutzen der Protokolle werden in Kapitel 4 genauer beschrieben.

3.5 Delauney Triangulierung

Die Delauney Triangulierung ist in der Lage aus einer Punktewolke ein Netz aus Dreiecken zu generieren [19]. Das Verfahren ist nach dem russischen Mathematiker Boris Nikolajewitsch Delone benannt. Dieses Verfahren ermöglicht die Triangulation einer Punktewolke mit der Eigenschaft, dass jeder Innenwinkel eines Dreiecks möglichst maximiert wird. Dies ist besonders nützlich, wenn man die Eckdaten der Dreiecke als Interpolationsdaten verwenden möchte, da dies den Interpolationsfehler minimiert [20]. Die Berechnung erfolgt über das „Circumcircle“ Kriterium. Hierbei enthält der Umkreis der Eckpunkte a,b,c jedes Dreiecks keine weiteren Punkte der Punktewolke. Die Umsetzung des Kriteriums wird in Kapitel 4.6.4 beschrieben.

3.6 Baryzentrische Koordinaten

Um die Interpolation innerhalb eines Dreiecks zu beschreiben, werden baryzentrische Koordinaten verwendet [21]. Die Koordinaten wurden von A. F. Möbius 1827 entdeckt. Diese dienen als „Gewichtungsfaktoren“ der einzelnen Eckpunkte. Resultat ist ein Punkt, welcher als geometrisches Zentrum agiert, z.B. $P = (t_1, t_2, t_3)$. Der Punkt P ist nun das geometrische Zentrum der Gewichtungsfaktoren t_1, t_2, t_3. Die Gewichtungsfaktoren können als Distanz betrachtet werden, wie weit der Punkt vom jeweiligen Eckpunkt entfernt ist. Ein Wert von 1 bedeutet der Punkt liegt genau auf dem Eckpunkt und 0 bedeutet dieser ist maximal entfernt. In dieser Arbeit werden homogene baryzentrische Koordinaten betrachtet. Diese haben die Eigenschaft, dass die Summe der Gewichtungsfaktoren immer 1 ergibt, also gilt: $t_1 + t_2 + t_3 = 1$. Möchte man nun einen Punkt innerhalb des Dreiecks beschreiben, kann jeder Gewichtungsfaktor nur zwischen 0 und 1 liegen. Um nun in einem Dreieck den interpolierten Wert der drei Eckpunkt zu bestimmen, wird lediglich jeder Wert am Eckpunkt mit dem entsprechenden Gewichtungsfaktor multipliziert und die Ergebnisse summiert. Z.B. ein Dreieck

mit den Gewichtungen t_1, t_2, t_3 und den Werten 2, 4, 5 würde am Punkt P eine Interpolation von $P(t_1, t_2, t_3) = t_1 \cdot 2 + t_2 \cdot 4 + t_3 \cdot 5$ haben.

4 Umsetzung

Das folgende Kapitel beschreibt die praktische Umsetzung als ein System. Das System besteht aus einem Mikrokontroller und einem Server. Sowohl der Mikrokontroller als auch der Server führen Programm aus, welche in diesem Kapitel beschreiben werden. Es wird darauf eingegangen, wie diese ablaufen und welche Funktionalitäten diese bieten.

4.1 Übersicht

Das System besteht aus zwei Kernkomponenten, einem Mikrokontroller und einem Server. Der Server erstellt einen WLAN-Hotspot und lauscht auf einem TCP-Socket für eingehende Verbindungen. Der Mikrokontroller verbindet sich zum konfigurierten WLAN-Hotspot, solange dieser nicht schon verbunden ist. Anschließend versucht dieser eine TCP-Verbindung zum Default Gateway des WLAN-Hotspots herzustellen. Sobald diese Schritte erfolgreich ausgeführt worden sind, besteht nun eine TCP-Verbindung zwischen dem Server und dem Mikrokontroller. Beide hören nun auf eingehende Nachrichten und können ebenfalls welche versenden.

4.2 Mikrokontroller Hardware

Die Hardware muss in der Lage sein WLAN-Signale empfangen, senden und verarbeiten zu können. Um in der Lage zu sein, verschiedenste Arten von Gegenständen zu orten, sollte diese Hardware ebenfalls geringe Anschaffungskosten haben. Die Wahl fiel daher auf ein SoC (System on a Chip) namens ESP32 [22] von dem chinesischen Hersteller *Espressif*. Dieses Board hat einen WLAN-Chip integriert und kann programmiert werden. Daher kann man ein Programm auf diesem laufen lassen, um die erforderlichen WLAN-Daten zu empfangen und an einen Server zu schicken. Ebenfalls gibt es einige Modelle, welche man bereits für sehr geringe Kosten kaufen kann. Eines der Modelle kann bereits für den Preis von 2,95€ erwerben [23]. Zusätzlich wird ein Windows fähiges Gerät als Server benötigt. Der Server muss in der Lage sein einen WLAN-Hotspot einzurichten, zu welchem der ESP32 sich verbinden kann.

4.3 Server Hardware

Der Server muss in der Lage sein einen WLAN-Hotspot zu erstellen, per TCP zu kommunizieren und unter dem *Windows* Betriebssystem laufen. In diesem Projekt wurde ein Laptop verwendet, welcher einen WLAN-Chip integriert hat. Der Server lässt ein Programm laufen, welches für das Windows-Betriebssystem entwickelt worden ist. Der WLAN-Hotspot muss ein 2,4 Ghz Netzwerk erstellen können, da der ESP32 ausschließlich für dieses ausgelegt ist.

4.4 Bibliotheken

Die Software nutzt einige Bibliotheken und Frameworks für die Umsetzung. Sowohl die Software des Servers als auch die des ESP32 wurde in der Sprache C++ entwickelt. Beide verwenden die Standardbibliotheken der Sprache. Der Server verwendet zusätzlich die Win32 API von *Microsoft*, um TCP-Verbindungen aufzubauen, eine grafische Oberfläche zu erhalten und mit dem System zu kommunizieren. Die grafische Oberfläche und damit das UI wurde in OpenGL geschrieben und kann daher leicht auch auf andere Plattformen umgeschrieben werden. Die Software des ESP32 nutzt die ESP-IDF API, welche die meisten Funktionalitäten der Hardware zu Verfügung stellt. Diese API ist komplexer und aufwendiger zu nutzen, wie Arduino Bibliotheken z.B. Wifi.h, bieten damit aber mehr Kontrolle und Möglichkeiten, welche in diesem Projekt zum Vorteil genutzt worden sind.

4.5 ESP32-Software

Auf dem ESP32 läuft dauerhaft ein Programm. Dieses Programm ist dafür zuständig Befehle vom Server zu empfangen und Scans durchzuführen. Das Programm startet damit die nötigen Bibliotheken zu initialisieren und eine Verbindung zum Server Hotspot herzustellen, bevor eine Verbindung per TCP angefragt wird. Danach befindet sich das Programm in einer endlosen Schleife. Die Schleife testet immer wieder das Netzwerk und ob Befehle über dieses übertragen worden sind. Sowie Fehlerkorrekturen, welche in Kapitel 4.5.3 genauer beschrieben werden.

4.5.1 ESP32 WLAN Funktionen

Der ESP32 benötigt viele Netzwerk Funktionalitäten. Daher wurde direkt mit dem von Espressif gestellten ESP-IDF Framework gearbeitet. Der ESP32 muss in der Lage sein:

1. Eine Verbindung zu einem WLAN-Netzwerk herzustellen
2. Probe Requests auf dem gewünschten Frequenzband zu senden und empfangen
3. Eine TCP-Verbindung zu einem Server herzustellen
4. Über eine TCP-Verbindung Daten zu empfange und zu senden

Die esp_wifi.h Bibliothek bietet bereits für Punkt 1 und 2 die nötigen Funktionen. Es wird der WLAN-Treiber initialisiert und anschließen können per Funktionen Verbindung hergestellt und getrennt werden. Zusätzlich kann eine Callback Funktion bei der Initialisierung übergeben werden, um Statusinformationen über die Verbindung zu erhalten. Ein globales Objekt wird dann genutzt, um den Status der Verbindung zu halten. Der Status kann anschließend im Hauptcode abgefragt werden, um basierend auf dem Status anderen Code auszuführen. Funktionen, um Probe Request zu senden und zu empfangen gibt es nicht. Stattdessen bietet die Bibliothek die Möglichkeit einen „Raw" ieee80211 Frame zu senden. Nach dem Senden wird gewartet, bis ein globales Objekt names „scanData" beschrieben wird. Dieses Objekt dient dazu, um nach einem Beacon Frame auf eine Beacon Response zu warten. Um Beacon Frames auch empfangen zu können, wird der ESP32 in den so genannten „promiscuous mode" versetzt. In diesem Modus ruft der Mikrokontroller eine Callback Funktion auf bei jedem empfangenen Packet. Diese Funktion kann einen sehr großen Performance Verlust haben, da diese bei jedem empfangenen Packet aufgerufen wird. Daher wird die Möglichkeit geboten nur nach bestimmten Packet Typen zu filtern. Beacon Frames gehören zu den „Management Frames", daher wird der Filter so konfiguriert, dass nur diese in die Callback Funktion übergeben werden. Anschließend wird in der Funktion der Packet Typ überprüft und es können die Daten den Frames entnommen werden, wenn es sich um einen Beacon Frame handelt. Es wird überprüft, ob die Daten von dem korrekten Netzwerk kommt mithilfe der SSID, welche bei der Request im scanData Objekt gespeichert worden sind. Falls dies der Fall ist, wird ein boolean Wert im Objekt auf True gesetzt und der RSSI-Wert kann ausgelesen werden. Die Logik, um die Signalstärke eines bestimmten Netzwerkes auszulesen, wurde in einer Funktion gekapselt. Diese Funktion nimmt als Parameter ein Objekt der Netzwerkdaten, eine Anzahl an Versuchen und einen Timeout Wert. Die Funktion testet erst, ob schon bekannt ist, auf welchem Channel das Netzwerk sich befindet, falls dies der Fall ist, wird nur dieser Channel gescannt, falls nicht wird über alle 13 Channels gescannt. Der Scan schickt eine Probe Request auf dem ausgewählten Channel und wartet anschließen den Timeout Wert des Funktionsparameters ab. Falls das Empfangen zu lange dauert, wird dieser Vorgang so oft wiederholt, wie der Parameter für die Anzahl der Versuche angibt. War der Scan erfolgreich, wird im Netzwerk Objekt gespeichert, auf welchem Channel das Netzwerk

sich befindet, um weitere Scans schneller durchzuführen. Falls alle Scans fehlschlagen, wird der Channel im Netzwerk Objekt als invalide markiert und die Funktion gibt keine gemessene Signalstärke zurück.

4.5.2 ESP32 TCP Funktionen

Der ESP32 muss in der Lage sein eine TCP-Verbindung zu einem Server herzustellen und über diese Daten zu empfangen und zu senden. Verwendet wird hierzu die lwIP socket Bibliothek [24], da diese vom ESP32 unterstützt wird. Diese verhält sich ähnlich zu Sockets, wie man diese unter *Linux* oder *Windows* findet. Da der ESP32 immer die Verbindung zum Sever herstellt, wird im ESP32 Code kein listening socket erstellt. Wenn der ESP32 keine Verbindung zu einem Server hat, muss das Programm nicht aktiv laufen. Daher ist der socket nicht blockierend bei einem Verbindungsversuch. Sobald eine Verbindung zu einem Server hergestellt wurde, wird der socket in den nicht blockierenden Zustand versetzt. Dies ermöglicht es Nachrichten auszulesen, ohne den Programmfluss zu stoppen.

4.5.3 ESP32 Programmablauf

Das Programm auf dem ESP32 besteht aus eines Initialisierungsphase und einer endlosen Schleife. Die Initialisierungsphase initialisiert die nötigen Netzwerkfunktionen und erstellt einige globale Variablen sämtliche Zustände des Programms zu halten. Zusätzlich wird eine Verbindungsversuch zum WLAN-Netzwerk durchgeführt. Das Netzwerk und die Zugangsdaten zum Netzwerk werden im Code gespeichert und können daher nicht zur Laufzeit geändert werden. Falls die Verbindung erfolgreich war, wird zusätzlich einmalig versucht eine TCP-Verbindung hergestellt zu werden. Anschließend springt das Programm in eine endlose Schleife. In dieser wird so lange versuch eine Verbindung zum WLAN-Netzwerk hergestellt zu werden, bis diese erfolgreich war. Anschließend wird versucht eine TCP-Verbindung hergestellt zu werden, falls es noch kein gibt. Als Ziel der Verbindung wird das Default Gateway des Netzwerks gewählt. Am Ende der Schleife wird ein überprüft, ob ein Zeitstempel schon länger wie 6 Sekunden abgelaufen ist. Dies wird dazu verwendet, um die WLAN-Verbindung zu trennen. Während der Laufzeit des Gerätes kam es öfters vor, dass die Callback Funktion, welche dem ESP32 mitteilen soll, dass das Gerät die Verbindung verloren hat, nicht aufgerufen wird. Daher wird nach einer kurzen Zeit die Verbindung beendet und beim nächsten Schleifendurchlauf wird diese dann wieder hergestellt. Zum Schluss

wird das Netzwerk überprüft, ob Nachrichten eingegangen sind. Der Server sendet periodisch immer wieder „Ping“ Nachrichten, um den sicherzustellen, dass die TCP-Verbindung zwischen ESP32 und dem Server noch funktioniert. Der ESP antwortet mit einer Ping Nachricht und setzt den Zeitstempel neu, da die Verbindung noch existiert und nicht neu verbunden werden muss.

4.5.4 ESP32 Scanablauf

Der ESP32 ist in der Lage zwei Arten von Messungen durchzuführen. Eine bei welcher jedes gespeicherte Netzwerk genau einmal die Signalstärke ausgelesen wird und eine, bei welcher die Signalstärke für jedes Netzwerk mehrmals ausgelesen wird und dann der Durchschnittwert gebildet wird. Das erste Messverfahren wird in diesem Dokument als „Scan“ bezeichnet und das zweite Messverfahren als „Avg-Scan“. Die Funktionsweiße des Scans wurde bereits in Kapitel 4.5.1 beschreiben. Der Scan muss die Verbindung zum aktuellen WLAN beenden. Daher werden die Ergebnisse des Scans global gespeichert, da nicht sichergestellt werden kann, dass die Verbindung am Ende des Scans zum WLAN-Netzwerk und die TCP-Verbindung wieder hergestellt werden kann. Es werden genau drei Versuche durchgeführt eine Verbindung zum WLAN-Netzwerk wieder herzustellen. Anschließend versucht eine TCP-Verbindung herzustellen und falls dies erfolgreich war, werden die Scandaten übertragen. Falls die Verbindung oder die Übertragung scheitern, werden die Daten weiterhin global gehalten und in der hauptschleife wird dann bei jedem Durchlauf immer wieder versucht diese Daten zu übertragen. Damit wird sichergestellt, dass selbst bei Verbindungproblemen nach dem Scan die Scandaten noch immer am Server ankommen.

4.6 Server-Software

Die Server Software wurde vollständig in C++ entwickelt und nur für das Windows Betriebssystem. Es besteht aus einer grafischen Oberfläche, welche mithilfe von OpenGL gerendert wird. Alle Netzwerkfunktionen, Nutzereingaben, etc. wurden mit der von Microsoft gestellte Win32 API entwickelt. Das Programm ermöglicht dem Nutzer automatisch eine Verbindung zum ESP32 herzustellen, Signalstärke Daten beliebiger WLAN-Netzwerke aufzuzeichnen und die Position des ESP32 im Raum zu finden. Die grafische Oberfläche kann genutzt werden, um ein Raum Layout zu laden und Befehle an den ESP32 zu senden. Fehler und Statusinformationen werden als „Popup-Nach-

richten“ in der Applikation angezeigt, um diese benutzerfreundlich zu halten. Die Umsetzung der grafischen Oberfläche wird in dieser Arbeit nicht beschrieben, da diese keine Relevanz zum Forschungsthema hat.

4.6.1 Server Netzwerkfunktionen

Auf dem Server gibt es Funktionen zum Verbinden, Empfangen und Senden von Nachrichten, aber auch eine Funktion zum „Hören“ auf eingehende Verbindungen. Der ESP32 initiiert immer die Verbindung. Daher muss der Server eine Funktion unterstützen, um eingehende Verbindungen anzunehmen. Die Verbindung und die eingehenden Verbindungen werden in einem Objekt gespeichert. Das Objekt hält einen socket welcher die Verbindung zum ESP32 hält und einen socket, welcher im listening Modus ist, um auf neue Verbindungen zu warten. Zusätzlich gibt es Funktionen, um Nachrichten zu Senden und zu Empfangen. Diese sind nicht blockierend, um die Applikation nicht „hängen“ zu lassen.

4.6.2 Server Programmablauf

Das Programm auf dem Server läuft mit zwei Threads. Der eine Thread ist für die Anzeige der Oberfläche zuständig und reagiert auf Benutzereingaben. Der andere Thread behandelt sämtliche Netzwerkfunktionen. Die Netzwerkfunktionen werden mit Mutexen blockiert. Beide Threads können auf die Funktionen zugreifen, ohne dass Race Conditions entstehen. Die grafische Oberfläche beinhaltet Knöpfe und Texteingaben. Ereignisse bei diesen werden über Callback-Funktionen gehandhabt. Diese modifizieren den Zustand des Programms, welcher in globalen Variablen gespeichert wird. Das Programm startet mit der Initialisierung eines Standardzustands, welcher über die grafische Oberfläche modifiziert werden kann. Netzwerk- und Fensterfunktionen werden initialisiert und das Programm zeigt die Benutzeroberfläche an. Der Netzwerk Thread testet durchgängig auf eingehende Verbindungen. Sobald eine Verbindung besteht, werden die Nachrichten empfangen und verarbeitet. Der Hauptthread rendert alle nötigen UI-Elemente und den aktuellen Modes im welchen sich das Programm befindet. Welche Modi es gibt und wie diese dem Nutzer angezeigt werden, erläutert das Kapitel 4.6.3.

4.6.3 Überblick der Benutzeroberfläche

Die Benutzeroberfläche ist ein drei Modi aufgeteilt. Der erste Modus „Heatmap-Modus“ ist dafür zuständig die Scandaten aufzuzeichnen, anzuzeigen und die Heatmaps zu erstellen. Eine Heatmap beschreibt eine Verteilung der Signalstärken über eine Fläche. In der Applikation wird eine Heatmap als ein farbiges Bild im Raum dargestellt. Die zweidimensionale Auflösung einer Heatmap ist als Konstante im Code gespeichert und kann nicht zur Laufzeit geändert werden. Jede Farbe im Bild repräsentiert die Signalstärke des Netzwerks an diesem Punkt im Raum. An der rechten Seite des Fensters befindet sich ein Menü, um Funktionen während der Laufzeit auszuführen und den Status des Programms zu ändern. Der zweite Modus „Search-Modus“ dient dazu den ESP32 im Raum zu lokalisieren.

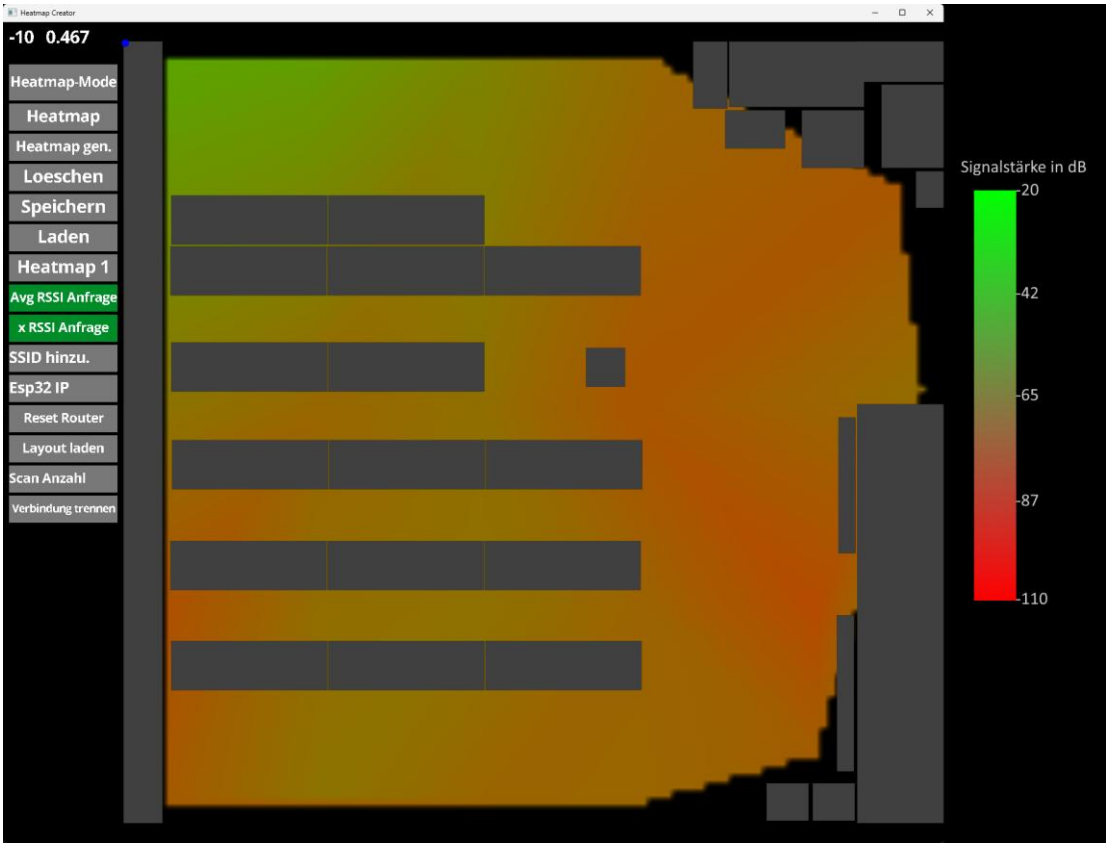

Abbildung 1: Beispiel einer Heatmap

Der dritte Modus kann genutzt werden, um die Scandaten als Graphen zu visualisieren und als JSON-Daten zu exportieren. Der dritte Modus „Display-Modus“ wurde viel bei den Tests und Forschung genutzt, wird aber nicht genauer in diesem Dokument beschrieben, da dieser nicht relevant für die Lokalisierung des ESP32 ist. Im Heatmap-Modus kann zwischen der Ansicht der aufgenommenen Scandaten und der berechneten

Heatmaps gewechselt werden. Die Heatmaps werden erst nach drücken des generieren Knopfes angezeigt. Der Nutzer kann zusätzlich so viele Punkte wie die Anzahl der konfigurierten Heatmaps in das Layoutout platzieren. Diese werden dann als Router Positionen angezeigt. Die Scandaten werden als farbige Punkte im Layout visualisiert. Eine grüne Farbe bedeutet eine gute Signalstärke und eine rötliche Farbe eine Schlechtere. Die Scandaten können mit einem linken Klick auf diese auch wieder gelöscht werden. Im Search-Modus wird weiterhin das Layout des Raumes angezeigt, nun aber eine orangefarbene Distanz Heatmap. Diese Distanz Heatmap wird als Summe aller Differenz jeder Heatmaps mit der gemessen Signalstärke berechnet. Je sichtbarer und heller der Wert in der Distanz Heatmap, desto eher liegt die berechnete Position des ESP32 an dieser Position.

Abbildung 2: Beispiel einer Distanz Heatmap

Ebenfalls kann der Nutzer die tatsächliche Position des ESP32 mit einem Rechtsklick angeben und die Software berechnet automatisch die Distanz. Das Layout wird bei den Berechnungen nicht berücksichtigt.

4.6.4 Berechnung der Heatmaps

Im Heatmap-Modus können Mespunkte im Raum aufgenommen werden. Zwischen den Punkten gibt es jedoch keine Informationen über die Signalstärke. Die Signalstärke im Raum fällt aber mit der Distanz ab. Daher wird die Annahme getroffen, dass auch die Signalstärke zwischen zwei oder mehr Messpunkten, welche sich an unterschiedlichen Positionen im Raum befinden, sich ändern kann. Jede Heatmap für jeden definierten Router hat eine feste zweidimensionale Größe. Dieser ist aktuell bei allen Tests auf einen Wert von 100x100 gesetzt worden. Es können daher pro Router für den gesamten Raum 10000 Signalstärkewert gespeichert werden. Die Messpunkte können frei im Raum platziert werden. Daher musste ein Algorithmus genutzt werden, welcher in der Lage ist aus der Punktewolke eine Heatmap zu generieren. Der hier gewählte Ansatz besteht daraus erst die Punktewolke in ein Netz von Dreiecken umzuwandeln per Delauney Triangulierung. Anschließend wird für jeden Punkt im Dreieck der auf die Heatmap abbildet werden kann die lineare Interpolation der drei Eckpunkte berechnet mithilfe von baryzentrischen Koordinaten. Die Delauney Triangulierung erfolgt mittels eines naiven Algorithmus. Dieser iteriert über alle möglichen Dreiecke, die aus der Punktewolke gebildet werden können. Anschließend wird das Umkreis Theorem geprüft. Sollte das Kriterium erfolgreich sein, wird das Dreieck gespeichert. Dieser Algorithmus läuft so lange, bis über alle Punkte iteriert worden ist. Da immer über drei Punkte iteriert werden muss und anschließend nochmal jeder Punkt aufgrund des Theorems getestet wird, hat der Algorithmus eine Laufzeitkomplexität von $O(n^4)$ wobei n die Anzahl der Punkte in der Punktewolke ist. Anschließen wird über jedes Dreieck iteriert und mithilfe der baryzentrischen Koordinaten werden die Interpolationsdaten berechnet. Der Algorithmus berechnet zuerst eine AABB (Axis Aligned Bounding Box) [25] um das Dreieck und testet anschließend jeden Punkt aus der AABB, ob die baryzentrischen Gewichtungen stimmen. Der Code hierfür sieht wie folgt aus:

```
void drawInterpolatedTriangle(Image& image, WORD x1, WORD y1, WORD x2, WORD
y2, WORD x3, WORD y3, BYTE val1, BYTE val2, BYTE val3)noexcept{
    WORD minX = std::min(x1, std::min(x2, x3));
    WORD maxX = std::max(x1, std::max(x2, x3));
    WORD minY = std::min(y1, std::min(y2, y3));
    WORD maxY = std::max(y1, std::max(y2, y3));

    float totalArea = 1.f/((x2-x1)*(y3-y2)-(y2-y1)*(x3-x2));
```

```
        for(WORD y=minY; y <= maxY; ++y){
            for(WORD x=minX; x <= maxX; ++x){
                float m1 = ((x2-x)*(y3-y)-(x3-x)*(y2-y))*totalArea;
                float m2 = ((x3-x)*(y1-y)-(x1-x)*(y3-y))*totalArea;
                float m3 = 1-m1-m2;
                if(m1 >= 0 && m2 >= 0 && m3 >= 0){
                    BYTE value = val1*m1 + val2*m2 + val3*m3;
                    image.data[y*image.width+x] = RGBA(value, 255-value, 0);
                }
            }
        }
    }
```

Das Kriterium der Delauney Triangulation versagt jedoch, falls es Quadrate gibt. Hier beinhalten die zwei möglichen Dreiecke immer vier Punkt. Um dies zu umgehen werden alle Punkte aus der Punktewolke um einen zufälligen Wert zwischen -0.05 und 0.05 sowohl in der X-Richtung als auch in der Y-Richtung verschoben. Dies verhindert in den meisten Fällen exakte Quadrate.

4.6.5 Berechnung der Position des ESP32

Die Software kann automatisch die geschätzte Position des ESP32 berechnen. Zuerst müssen die Heatmaps der einzelnen Router berechnet werden. Danach wird die Software in den Search-Modus geändert. Sobald nun Scandaten vom ESP32 an den Server übertragen werden, wird die Position des ESP32 im Layout als Punkt angezeigt. Falls die tatsächliche Position des ESP32 gesetzt worden ist, wird die euklidische Distanz in Metern berechnet und angezeigt. Zuerst wird eine Distanz-Heatmap für jede Heatmap der Router erstellt. Es wird anschließen der Absolutwert der Differenz der empfangenen Signalstärke mit jedem Wert in der Heatmap berechnet:

```
void getDistanceMap(Image* heatmapsInterpolated, BYTE idx, DistanceMap& map){
    for(DWORD j=0; j < DATAPOINTRESOLUTIONX*DATAPOINTRESOLUTIONY; ++j){
        DWORD& storedValue = heatmapsInterpolated[idx].data[j];
        int diffRed = R(storedValue)-R(searchValue[idx]);
        map.distances[idx][j] += abs(diffRed);
    }
}
```

Die resultierenden Distanz-Heatmaps beinhalten den Abstand zwischen dem gemessenen Wert zum gespeicherten Wert in der Heatmap. Zuletzt wird eine Summe über alle Distanz-Heatmaps gebildet. Die resultierende Distanz-Heatmap beschreibt zu jedem Punkt die Abweichung zwischen dem gemessenen Wert und dem gespeicherten Wert

als Kombination aller Heatmaps. Die finale Position ist das Minimum der Distanz-Heatmap. Sollte es mehrere minimale Werte in der Distanz-Heatmap geben, wird der geometrische Mittelpunkt bestimmt und als finale Position genutzt.

4.7 Kommunikation zwischen ESP32 und Server

Die Kommunikation zwischen ESP32 und Server findet ausschließlich per TCP statt. Der ursprüngliche Prototyp hat auf UDP aufgebaut, welches einfacher zu implementieren war und schneller gewesen ist. In einem privaten Umfeld war UDP sehr zuverlässig, mit einem Packet Verlust von unter ~8%. Bei den Tests an der Hochschule stieg der Packet Verlust auf über ~80% an. Eine Möglichkeit dieses Problem zu beheben wäre ein Request-Response System. Hier könnte der Sender auf die Antwort einer Nachricht warten und falls diese Antwort nicht in einer festgelegten Zeitspanne eingeht, wird die Nachricht erneut gesendet. Grundlegend ist dies, was TCP macht. Daher wurde die Applikation mit TCP neu geschrieben und der Packet Verlust tritt nun nicht mehr auf. TCP bietet zwar Mechanismen, um sicher zu stellen, dass Nachrichten ankommen, sollte aber dennoch ein Problem auftauchen und die Nachrichten erreichen nie ihr Ziel, muss dies dem Nutzer mitgeteilt werden. Da die Sende- und Empfangsfunktionen nicht den Programmfluss stoppen sollen, kann nicht gewartet werden, bis das Betriebssystem meldet, dass die Nachrichten nie deren Ziel erreicht haben. Daher wurde ein einfaches Protokoll auf der Anwendungsschicht geschrieben, um den Nutzer zu informieren, ob die Kommunikation Probleme aufwies. Jede Nachricht wird nach einem fest definierten Protokoll übertragen. Das erste Byte der Nachricht legt den Typ der Nachricht fest. Der Typ kann ein einfaches „Acknowledgement“ sein oder auch die Übertragung eines neuen Netzwerks, welches der ESP32 scannen soll. Damit kann im Programm z.b. über ein switch-case der Typ der Nachricht explizit analysiert werden und entsprechender Code ausgeführt werden. Als Beispiel kann der Nutzer in der Applikation dem ESP32 ein neues Netzwerk zusenden, welches im Scan ausgemessen werden soll. Sobald der ESP32 die Nachricht empfängt, kann anhand des Typen festgestellt werden, dass die Nachricht noch zusätzlich den Namen des Netzwerks beinhaltet. Der ESP32 kann darauf ein Acknowledgement an den Server schicken, um den Nutzer zu informieren, dass das neue Netzwerk im nächsten Scan mitberücksichtigt wird. Der folgende Codeausschnitt soll die möglichen Nachrichtenformate, die während der Arbeit genutzt worden sind, zeigen:

```
/*  Nachrichtenformate:
        1 Byte (0x00) SEND_POSITION_X
```

```
        1 Byte (0x01) SEND_POSITION_Y
        1 Byte (0x02) SEND_SIGNALSTRENGTH   | 1 Byte Anzahl | 1 Byte RSSI
Router 1  | 1 Byte RSSI Router 2  | 1 Byte RSSI Router 3,...
        1 Byte (0x03) ADD_ROUTER            | 1 Byte Länge  | n Bytes SSID
        1 Byte (0x04) RESET_ROUTERS
        1 Byte (0x05) SETSENDIP             | 4 Bytes IP            | 2 Bytes
PORT
        1 BYTE (0x06) REQ
        1 Byte (0x07) ACK
        1 BYTE (0x08) REQUEST_AVG
        1 BYTE (0x09) REQUEST_SCANS         | 2 Bytes Count
        1 BYTE (0x0A) SCAN_INFO             | 2 Bytes Anzahl Erfolgreicher
Scans            | 2 Bytes Fehlerhafte Scans | 2 Bytes Durchschnittliche Zeit
pro Scan
        1 BYTE (0x0B) REQ_STATUS
        1 BYTE (0x0C) SEND_STATUS           | 4 Bytes IP            | 2 Bytes
Port          | 1 Byte SSID Anzahl        | n Bytes SSIDs
        1 BYTE (0x0D) ALIVE_REQ
        1 BYTE (0x0E) ALIVE_ACK
    */
```

5 Ergebnisse und Verbesserungen

In diesem Kapitel werden die Ergebnisse verschiedener Tests des Systems aus Kapitel 4 beschrieben. Die Ergebnisse werden anhand der Ziele in Kapitel 1.3 gemessen. Es werden dann mögliche Verbesserungen am System ausprobiert und die Ergebnisse gezeigt. Falls eine Verbesserung Vorteile zeigt oder die Ziele explizit verbessert, wird die Verbesserung in den nachfolgenden Test beibehalten. Dies reduziert die Anzahl der Tests, die durchgeführt werden müssen, da die Anzahl der möglichen Kombinationen aller Verbesserungen reduziert wird.

5.1 Messumfeld

Sämtliche Messungen für die zunächst folgenden Test wurden in einem Vorlesungsraum der technischen Hochschule Mannheim durchgeführt. Der Raum ist nahezu rechteckig und beinhaltet Gegenstände wie Tische, Stühle, Schränke, etc. Dieser Raum repräsentiert typische Aufbauten von Office Räumen oder Büros. Am Ende der Test wird noch ein ganzes Stockwerk getestet. Dies bietet die Möglichkeit das System in einzelnen Räumen zu analysieren sowohl aber auch über größere Begebenheiten. Das folgende Layout soll zeigen, wie der Raum aufgebaut ist:

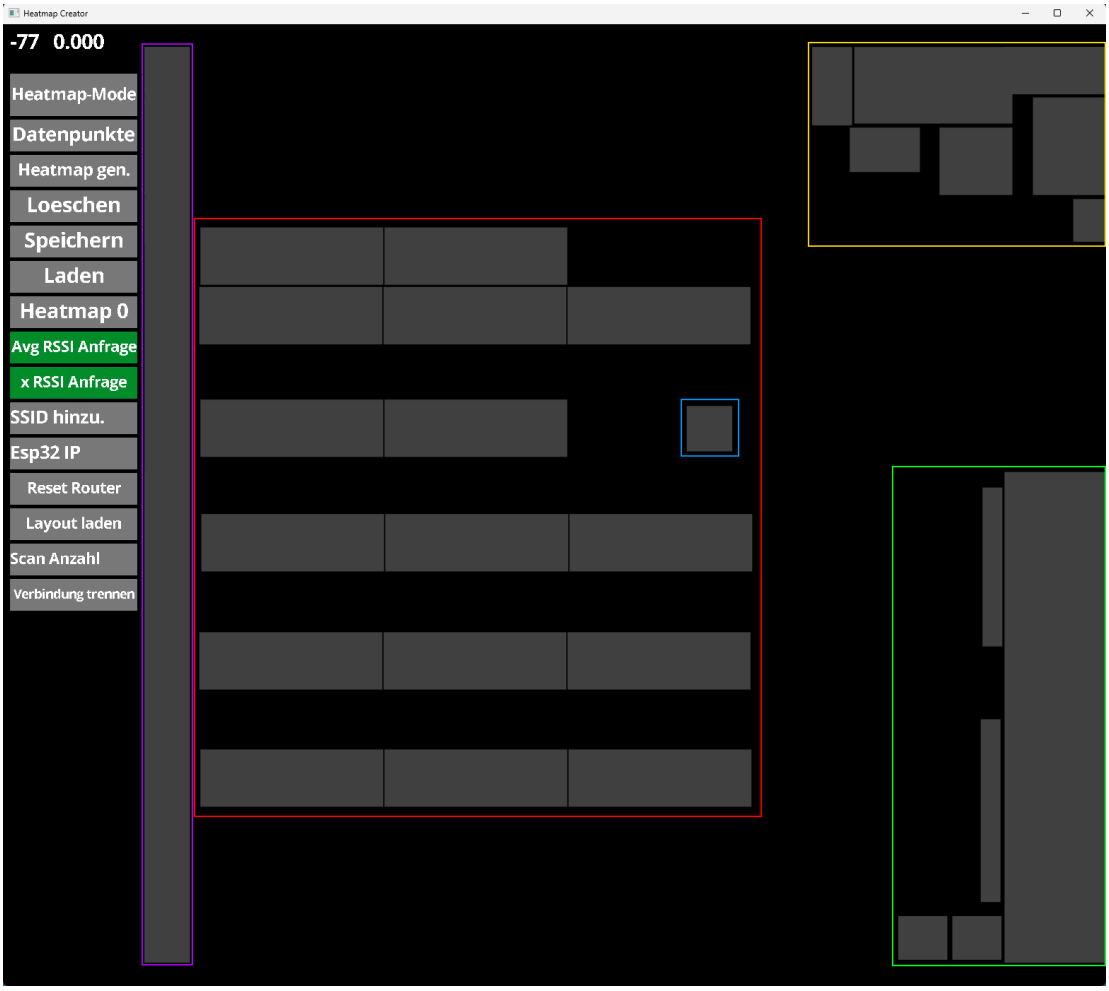

Abbildung 3: Raumlayout des Vorlesungsraums

Das Layout zeigt das Innenleben des Raums, Wände um den Raum herum sind nicht im Layout eingezeichnet. Die folgende Tabelle beschreibt, was im Layout zu sehen ist:

Tabelle 2: Beschreibung der Gegenstände im Layout des Vorlesungsraums

Markierter Bereich	**Bedeutung**
Rotes Rechteck	Tische und Stühle
Blaues Rechteck	Eine solide Säule, welche sich vom Boden bis unter die Decke streckt
Lila Rechteck	Eine kleine Erhöhung des Bodens, auf welcher des ESP32 nicht platziert wurde
Gelbes Rechteck	Schränke, Plakatwände, ein Waschbecken und ein Mülleimer

Grünes Rechteck	Schränke und Plakatwände

Der ESP32 wurde nicht auf diesen Hindernissen platziert. Der ESP32 wurde immer flach auf dem Boden platziert und wenn möglich mit einem minimalen Abstand zu den Gegenständen. Die Tische und Gegenstände im Raum sind unten herum offen.

5.2 Basismessung

Die Basismessung beschreibt den minimalen Aufbau des Systems. Diese soll zeigen, wie das System unter minimalem Aufwand performt. Die Basismessung besteht aus drei Routern, fünf Messpunkten und einem einzigen Scan pro Messpunkt. Es wurden drei Router genutzt, da man mit diesen die Position triangulieren kann. Unter optimalen Bedingungen würde die Signalstärke um jeden Router herum kreisförmig abfallen. Daher würden zwei Router demnach zwei Schnittpunkte der Ringe aufweisen. Ein dritter Router sorgt dann dafür, dass es nur noch genau einen Schnittpunkt gibt. Die fünf Messpunkte wurden gewählt, da die Annahme ist, dass die Signalstärke zwischen diesen ungehindert abfällt. Es würde, daher reichen diese in den vier Ecken des Raumes zu platzieren. Ein fünfter Punkt wurde in der Mitte des Raumes platziert, um die Signalstärke zusätzlich in der Mitte des Raumes aufzunehmen. Dieser dient dazu, dass die Messpunkte nicht zu weit auseinanderliegen und die Interpolation zwischen diesen korrekt verläuft. An jedem Messpunkt wird davon ausgegeben, dass die Signalstäre sich dort über die Zeit nicht ändert oder schwankt. Daher wird jeder Messpunkt genau einmal aufgenommen und gespeichert. Die finale Messung, um die Position zu bestimmen, wird ebenfalls mit genau einer Messung der Signalstärken durchgeführt.

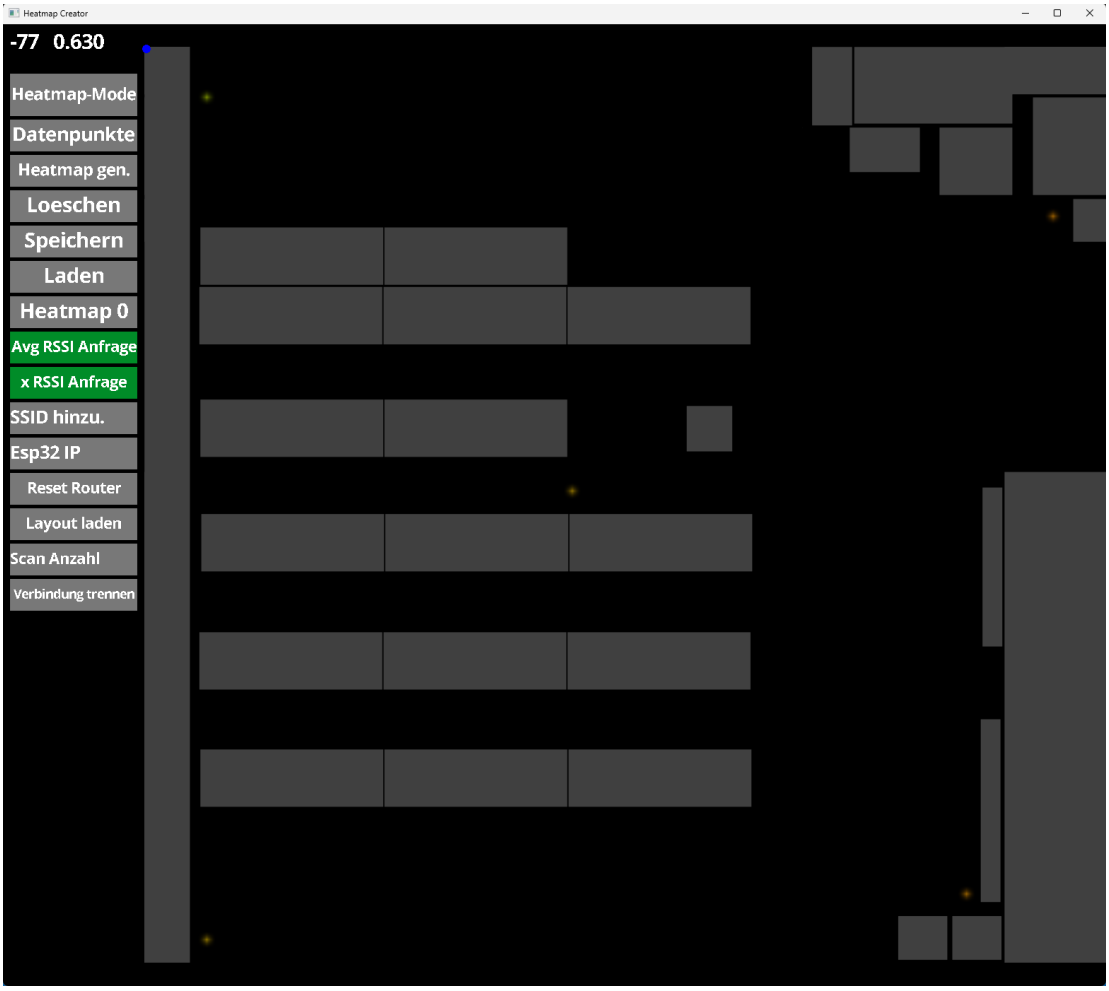

Abbildung 4: Verteilung der Positionen Im Raum als farbliche Punkte

5.2.1 Ergebnis der Basismessung

Um das System zu testen, wurden zufällige Positionen im Raum gewählt und die Positionen berechnet. Anschließend wurden diese dann mit der tatsächlichen Position des ESP32 verglichen. Die Genauigkeit des Systems mit diesem Aufbau ist sehr variabel. Die folgenden Distanzen wurden vom System berechnet:

0.343m, 5.648m, 4.271m, 4.367m, 1.750m, 1.156m, 6.208m, 3.459m, 3.918m, 1.480m, 0.824m

Die Genauigkeit beträgt im Durschnitt eine Distanz von 3.039 Meter. Jedoch gibt es eine hohe Standardabweichung von 1.928 Meter. Der Aufbau hat ca. 5 Minuten gedau-

ert. Eine Position kann ca. alle 80 Millisekunden gefunden werden. Die genauste Messung gab es bei einer Position nahe der Mitte des Raumes. Dort befand sich auch ein Messpunkt.

5.3 Erweiterung der Basismessung mit zusätzlichen Messpunkten

Dies ist die erste Verbesserung zur Basismessung. Die Messung an einem Messpunkt ist die wahre Signalstärke an dieser Position im Raum. Signalstärken fallen nicht radial um einen Router ab, da in Räumen Wände, Gegenstände, etc. die Signalstärke beeinflussen können. Daher sollte für jeden Punkt im Raum eine Messung durchgeführt werden. Jedoch ist es unrealistisch eine sehr hohe Anzahl an Messpunkten aufzunehmen, da dies mehr Zeit in Anspruch nimmt.

5.3.1 Erweiterung auf 20 Messpunkte

Die Basismessung wurde nun mit 20 Messpunkte durchgeführt. Diese wurden möglichst gleichmäßig im Raum verteilt, um den Raum möglichst breitflächig mit Messpunkten abzudecken. Die folgende Abbildung zeigt die Verteilung der Messpunkte:

Abbildung 5: Verteilung von 20 Messpunkten im Raum

Die berechneten Distanzen betrugen die folgenden Werte:

1.015m, 1.085m, 1.898m, 2.537m, 2.331m, 1.921m, 3.412m

Die Genauigkeit im Gegensatz zur Basismessung hat sich auf einen Durchschnittswert von 2.028 Meter verbessert. Die Standardabweichung lag bei dieser Messung bei 0.776 Meter. Es dauerte insgesamt ca. 12 Minuten alle Messpunkte aufzunehmen. Die Position kann weiterhin ca. alle 80 Millisekunden berechnet werden.

5.3.2 Erweiterung auf 59 Messpunkte

Die Messpunktanzahl der Basismessung wurde nun mit 59 Messpunkten durchgeführt. Die folgende Abbildung zeigt die Verteilung der Messpunkte im Raum:

Abbildung 6: Verteilung von 59 Messpunkten im Raum

Folgende Distanzen wurden gemessen:

3.431m, 0.436m, 1.396m, 1.852m, 3.677m, 0.747m, 1.777m, 2.319m, 1.783m, 0.981m

Die Genauigkeit hat sich gegenüber der Messung mit 20 Messpunkten auf einen Durchschnittswert von 1.840 Meter verbessert. Im Gegensatz dazu, hat sich die Standardabweichung der Messung auf 0.981 Meter erhöht. Hier dauerte die Messung aller Messpunkte insgesamt ca. 30 Minuten. Auch hier kann eine Position ca. alle 80 Millisekunden vom System bestimmt werden.

5.3.3 Auswertung der Erweiterung der Messpunkte

Mehr Messpunkte haben die Genauigkeit präziser gemacht von einem durchschnittlichen Wert von 3.039 Meter der Basismessung zu einem Durchschnittswert von 1.840 Meter mit 59 Messpunkten. Es zeigt sich also, dass mehr Messpunkte die Genauigkeit erhöhen. Die Standardabweichung hat sich aber nicht im selben Ausmaß verringert. Daher lässt sich daraus schließen, dass sich eine hohe Varianz im System befindet. Daher werden im nächsten Abschnitt die Signalstärken selbst an den Routern untersucht, um weitere Verbesserungen finden zu können.

5.4 Untersuchung der Signalstärken an festen Punkten im Raum

Mit den Ergebnissen des Kapitel 5.3 wurde festgestellt, dass das System eine hohe Varianz hat. Daher werden die Signalstärken an festen Punkten im Raum untersucht, um herauszufinden, ob diese der Grund zur Varianz sind und im anschließenden Test wird dann versucht eine Lösung zu finden. Bei diesem Test wird der ESP32 an feste Punkte im Raum gesetzt und es werden mehrere Scans durchgeführt. Mit den Ergebnissen soll festgestellt werden, ob die Signalstärke im Raum an einer festen Position variieren kann. Das Experiment wird mit dem gleichen Aufbau durchgeführt, wie die Messdaten bisher durchgeführt worden sind. Die Messungen an einer Position werden dann gespeichert und analysiert.

5.4.1 Beobachtungen der Signalstärken im Raum

Die folgende Grafik zeigt die Anzahl der gemessenen Signalstärken für drei unterschiedliche Router. Die Distanzen der Router sind unterschiedlich weit weg gewählt worden, um ein möglichst großes Spektrum der Signalstärken zu zeigen. Pro Messpunkt wurden mehrere hundert Scans durchgeführt und die gemessene Signalstärke pro Scan wurde gespeichert. Die folge Abbildung zeigt die gemessenen Werte:

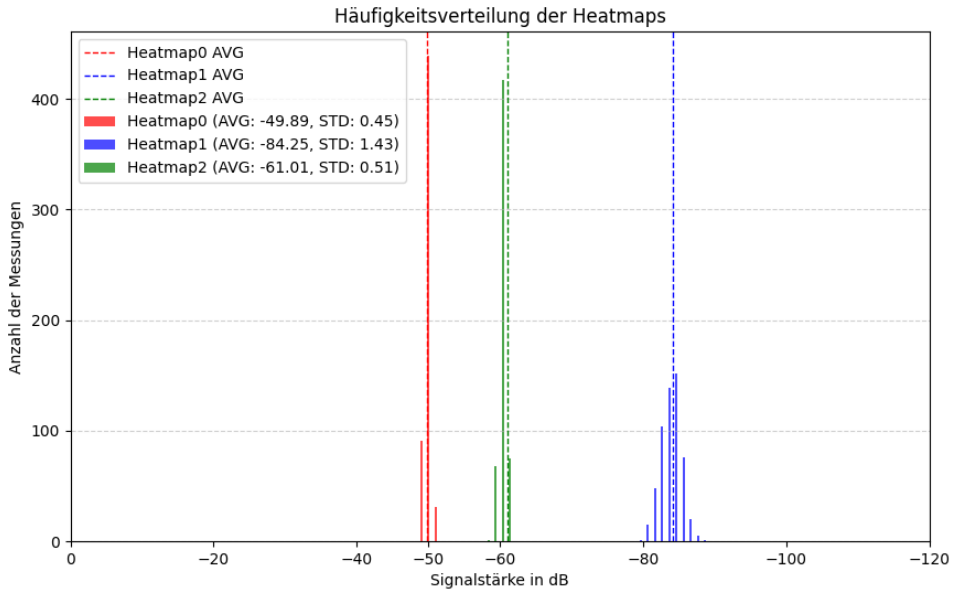

Abbildung 7: Streuung der Signalstärken an festen Punkten im Raum

Anhand der Daten wird feststellen, dass die Signalstärken nicht konstante Werte halten. Ebenfalls streuen die gemessenen Signalstärken basierend auf der Signalstärke unterschiedlich. Signalstärken in den niedrigeren Bereichen zeigen eine höhere Standardabweichung. Signalstärken im höheren Bereich streuen dagegen weniger. Es kann aber ebenfalls festgestellt werden, dass sich eine nahezu symmetrische Verteilung um ein Maximum bildet.

5.4.2 Schlussfolgerung der Signalstärken an festen Punkten im Raum

Die Messdaten zeigen, dass sich eine nahezu symmetrische Verteilung um ein Maximum bildet. In den folgenden Messungen werden daher nun mehrere Aufzeichnungen der Signalstärken pro Messpunkt durchgeführt und ein Durchschnittswert über diese gebildet. Damit soll der gemessene Signalstärkewert mehr konsistent pro Messpunkt werden.

5.5 Aufzeichnung mehrerer Signalstärken pro Messpunkt mit Durchschnittsbildung

Bei dieser Verbesserung werden pro Messpunkt mehrere Signalstärken aufgezeichnet und der Durchschnitt dieser gebildet. Das Ziel ist die Streuung der Signalstärken an einem festen Punkt zu reduzieren. Da aus Kapitel 5.3 bereits bekannt ist, dass eine

größere Anzahl von Messpunkten zu besseren Ergebnissen führt, werden nun 20 und 59 Messpunkte getestet. Der Durchschnitt wird über insgesamt 512 Messungen gebildet.

5.5.1 Durchschnittsbildung mit 20 Messpunkten

Die Verteilung der Messpunkte im Raum ist identisch zu der Verteilung aus Kapitel 5.3.1. Folgende Distanzen wurden gemessen:

0.251m, 1.534m, 1.247m, 0.470m, 3.165m, 0.603m, 1.867m, 0.318m, 0.722m, 0.959m, 0.879m

Die Genauigkeit hat sich bei diesem Test auf einen Wert von 1.092m im Gegensatz zum Test ohne Durchschnittsbildung verbessert. Die Standardabweichung zeigt einen Wert von 0.812m. Der Nachteil ist jedoch eine verringerte Performance. Die Zeit zum Ausmessen aller Messpunkte betrugt insgesamt ca. 24 Minuten. Eine Position kann nun nur noch alle 19-28 Sekunden gefunden werden. Die Dauer der Positionsbestimmung kann sehr schwanken. Dies kommt daher, da insgesamt 512 Scans erfolgreich durchgeführt werden müssen. Es kann öfters passieren, dass ein Scan mehrere Wiederholungen braucht, bis der ESP32 die Daten empfängt.

5.5.2 Durchschnittsbildung mit 59 Messpunkten

Die Messpunkte wurden identisch zu der Messung aus Kapitel 5.3.2 verteilt. Folgende Distanzen wurden gemessen:

1.557m, 3.746m, 2.899m, 4.389m, 2.823m, 1.880m, 2.694m, 0.444m, 1.136m, 2.677m, 2.256m

Die Genauigkeit verschlechterte sich bei diesem Test auf einen Wert von 2.409m im Gegensatz zur Messung ohne Durchschnittsbildung. Die Standardabweichung betrug 1.078m. Die Dauer zu Aufbau betrug hier eine Zeit von insgesamt 46 Minuten. Die Dauer zur Positionsbestimmung bleibt identisch zu der mit 20 Messpunkten.

5.5.3 Auswertung der Durchschnittwertberechnung per Messpunkt

Der Test mit 20 Messpunkten zeigt eine Verbesserung der Genauigkeit gegenüber aller bisherigen Tests. Die Standardabweichung hat sich jedoch nicht signifikant verändert. Der Test mit 59 Messpunkten erweis sogar eine schlechtere Genauigkeit gegenüber

dem Test ohne Durchschnittswertbildung. Da die Standardabweichung sich nicht verbessert hat, scheint die Varianz des Systems nicht nur in der zeitlichen Änderung der Signalstärken an festen Punkten zu liegen. Die Verbessrung der Genauigkeit bei 20 Messpunkten und die Ergebnisse der Messungen der Signalstärken an festen Punkten im Raum lassen dennoch darauf schließen, dass eine Durchschnittswertbildung sinnvoll ist. Daher wird diese für die weiteren Test beibehalten.

5.6 Auswirkung der Rotation des ESP32 auf die gemessene Signalstärke

Der verwendete ESP32 verwendet eine einfache direktionale Antenne. Der folgende Test soll untersuchen, ob die Signalstärke sich ändert, wenn der ESP32 rotiert wird. Der Test nutzt einen Router und einen ESP32 in direkter Sicht zum Router. Der ESP32 wird in 45° Schritten rotiert und die Signalstärke aufgenommen. Die Positionen des ESP32 und des Routers werden nicht verändert.

5.6.1 Signalstärkeänderungen bei Rotation des ESP32 ohne Antenne

Während der vorherigen Tests, wurde der ESP32 immer an feste Positionen im Raum gelegt, es wurde aber nicht eine konsistente Rotation sichergestellt. Die folgenden Diagramme sollen zeigen, wie die aufgenommen Signalstärke sich an einem festen Punkt im Raum ändert, wenn der ESP32 rotiert wird.

Tabelle 3: Signalstärken bei Rotation des ESP32 mit direktionaler Antenne

Rotation des ESP32 in Grad	**Durchschnittliche Signalstärke in dB**
0°	-52.13 dB
45°	-51.83 dB
90°	-51.43 dB
135°	-55.10 dB
180°	-54.71 dB
225°	-53.74 dB
270°	-57.39 dB
315°	-55.78 dB

Die Tabelle zeigt eine deutliche Änderung der Signalstärke bei einer Rotation des ESP32. Es wird eine maximale Signalstärke von -68.71 dB als Durschnitt gemessen und eine minimale Signalstärke von -79.92 dB. Dies ist eine Spanne von 11.21 dB. Daraus lässt sich schließen, dass die Rotation des ESP32 eine wichtige Rolle in der Aufnahme der Messdaten spielt. Da nicht immer eine bestimmte Rotation garantiert werden kann, z.B. wenn der ESP32 in Gegenständen verbaut ist, wird stattdessen eine Alternative gesucht. Omnidirektionale Antennen bieten eine Aufnahme von Signalen unabhängig von der Rotation. Der Test wird nun mit einer omnidirektionalen Antenne wiederholt, um zu untersuchen, ob diese die Schwankung in der gemessenen Signalstärke verringern kann.

5.6.2 Auswirkung der Rotation des ESP32 auf die gemessene Signalstärke mit einer omnidirektionalen Antenne

In diesem Test wurde die direktionale Antenne des ESP32 mit einer externen omnidirektionalen Antenne ersetzt. Bei der Messung der Signalstärke unter verschiedenen Rotationen mit der direktionale Antenne des ESP32 wurde festgestellt, dass die Signalstärke variiert. Mit diesem Test soll untersucht werden, ob die omnidirektionale Antenne die Varianz der Signalstärke verringert. Der ESP32 wurde um 45° rotiert und die Signalstärken aufgenommen. Der ESP32 wurde an feste Position im Raum platziert, von welcher dieser sich nicht bewegt hat.

Tabelle 4: Signalstärken bei Rotation des ESP32 mit omnidirektionaler Antenne

Rotation des ESP32 in Grad	**Durchschnittliche Signalstärke in dB**
0°	-52.13 dB
45°	-51.83 dB
90°	-51.43 dB
135°	-55.10 dB
180°	-54.71 dB
225°	-53.74 dB
270°	-57.39 dB
315°	-55.78 dB

Anhand der Tabelle können direkt zwei Eigenschaften festgestellt werden. Erstens hat sich die Signalstärke im Gegensatz zum Test ohne externe Antenne verbessert. Die Signalstärke hat sich von einem Durchschnittswert von -73.18 dB auf -54.01 dB erhöht. Zweitens hat sich die Spanne zwischen minimaler und maximaler Signalstärke verringert. Das Minimum beträgt -57.39 dB und das Maximum -51.43 dB. Die Differenz zwischen diesen beträgt einen Wert von 5,96 dB. Daraus lässt sich schließen, dass die omnidirektionale Antenne die Varianz, die die Rotation verursacht, zwar verringert, aber nicht vollständig behebt. Dennoch wurde die Varianz verringert und zusätzlich die allgemeine Signalstärke verbessert. Daher kann das System auch in größeren Räumen verwendet werden, was vorher nicht möglich war, da der ESP32 keine Scandaten mehr aufzeichnen konnte, da die Signale den Router nicht mehr erreicht haben.

5.7 Systemtest mit omnidirektionaler Antenne

Dieser Test beinhaltet alle bereits untersuchten Verbesserungen und zusätzlich die omnidirektionale Antenne. Es wird zuerst der Vorlesungsraum der Hochschule genutzt, wie auch in den vorherigen Tests. Der Test wurde mit 20 Messpunkten durchgeführt, da die Durchschnittsmessung mit dieser Anzahl die besten Ergebnisse in Hinsicht zu den Zielen ergeben hat. Es wurden die folgenden Distanzen aufgezeichnet:

0.945m, 1.634m, 1.084m, 3.145m, 1.479m, 2.642m, 1.151m, 4.027m

Der Test ergibt eine Genauigkeit von 2.013m und einer Standardabweichung von 1.055m. Die Dauer für den Aufbau betrug ca. 24 Minuten und eine Position kann ca. alle 19-28 Sekunden berechnet werden. Die folgende Abbildung zeigt die Heatmap des einen Routers des Tests:

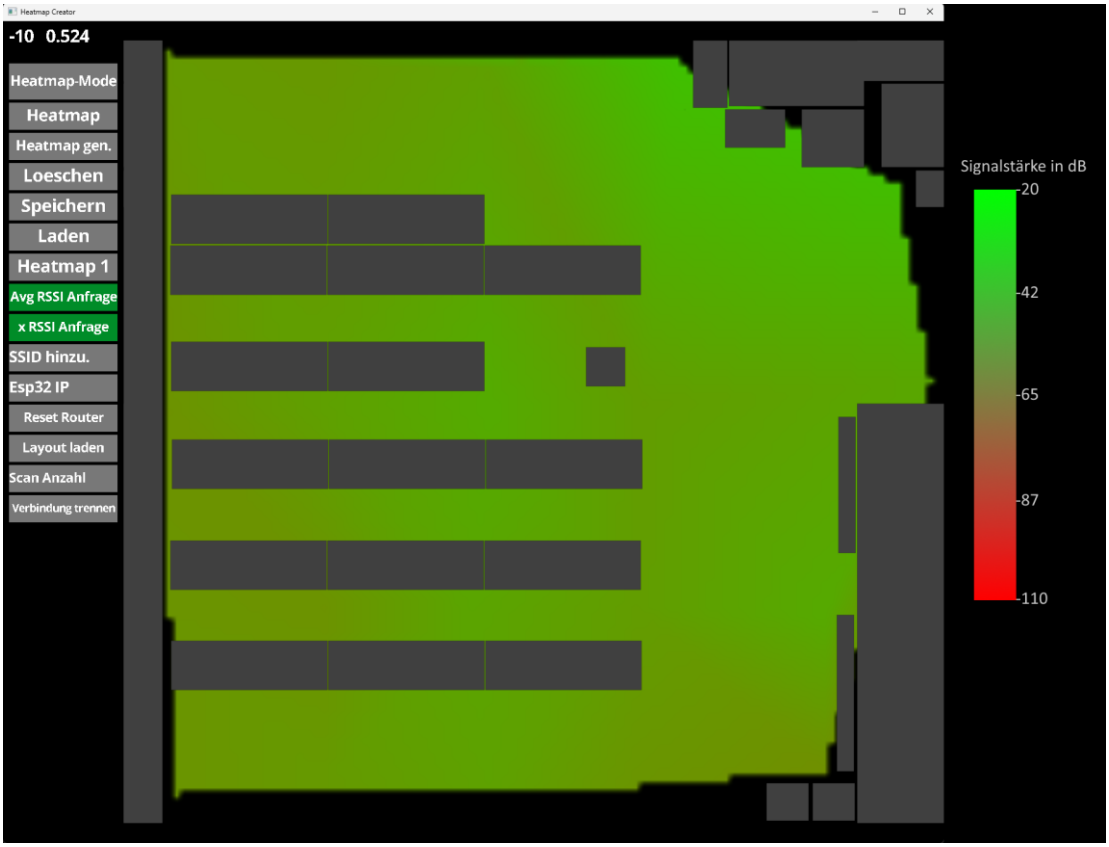

Abbildung 8: Heatmap des Vorlesungsraums mit Antenne

In der Abbildung ist zu erkennen, dass die Signalstärke im Raum großflächig gleichmäßig ist. Die vorherigen Tests haben bereits gezeigt, dass die Signalstärke an einem Punkt im Raum schwanken kann. Die Position kann also nicht effektiv vom System gefunden werden, da das Spektrum der Signalstärken, die im Raum gemessen wurden, sehr klein ist. Diese Eigenschaft wurde in den vorherigen Tests auch nicht berücksichtigt. Das Spektrum der Signalstärken kann in einem Raum nicht vergrößert werden. Eine Möglichkeit bestände aber darin einen größeren Raum zu nehmen oder ein gesamtes Stockwerk. Der nächste Test soll zeigen, ob dies Vorteile bringt.

5.7.1 Systemtest in einem gesamten Stockwerk

Dieser Test ersetzt den bisher genutzten Vorlesungsraum mit einem gesamten Stockwerk. Damit kann getestet werden, wie das System sich in einem neuen Umfeld verhält. Das gewählte Stockwerk ist deutlich größer als der Verlesungsraum und bietet zusätzlich Hindernisse, die es im Vorlesungsraum nicht gab. Der Vorlesungsraum war ein größtenteils rechteckiger Raum mit einigen Gegenständen wie Tischen, Stühlen, Plakatwänden, etc. Der neue Test in einem Stockwerk beinhaltet mehrere Räume und

dichte Wände, die die Möglichkeit bieten Signale zu absorbieren oder zu blocken. Im Test mit der omnidirektionalen Antenne konnte festgestellt werden, dass die Signalstärke im Raum ein kleines Spektrum aufwies. Die Erwartung in diesem Test ist, da die Distanz zwischen Router und ESP32 erhöht werden kann und zusätzlich Wände und Gegenstände das Signal blockieren können, dass das Spektrum der Signalstärke sich vergrößert und so das System eine bessere Genauigkeit aufweist. Die folgende Abbildung zeigt das Layout des Stockwerks:

Abbildung 9: Stockwerk Layout und Räume

Die roten Rechtecke maskieren einzelne Räume. Zwischen den Räumen befinden sich Wände. Die Räume sind alle offen miteinander verbunden. Für den Test wurden 3 Router verwendet. Diese wurden möglichst weit voneinander entfernt platziert. Die folgende Abbildung zeigt die Heatmaps der einzelnen Router:

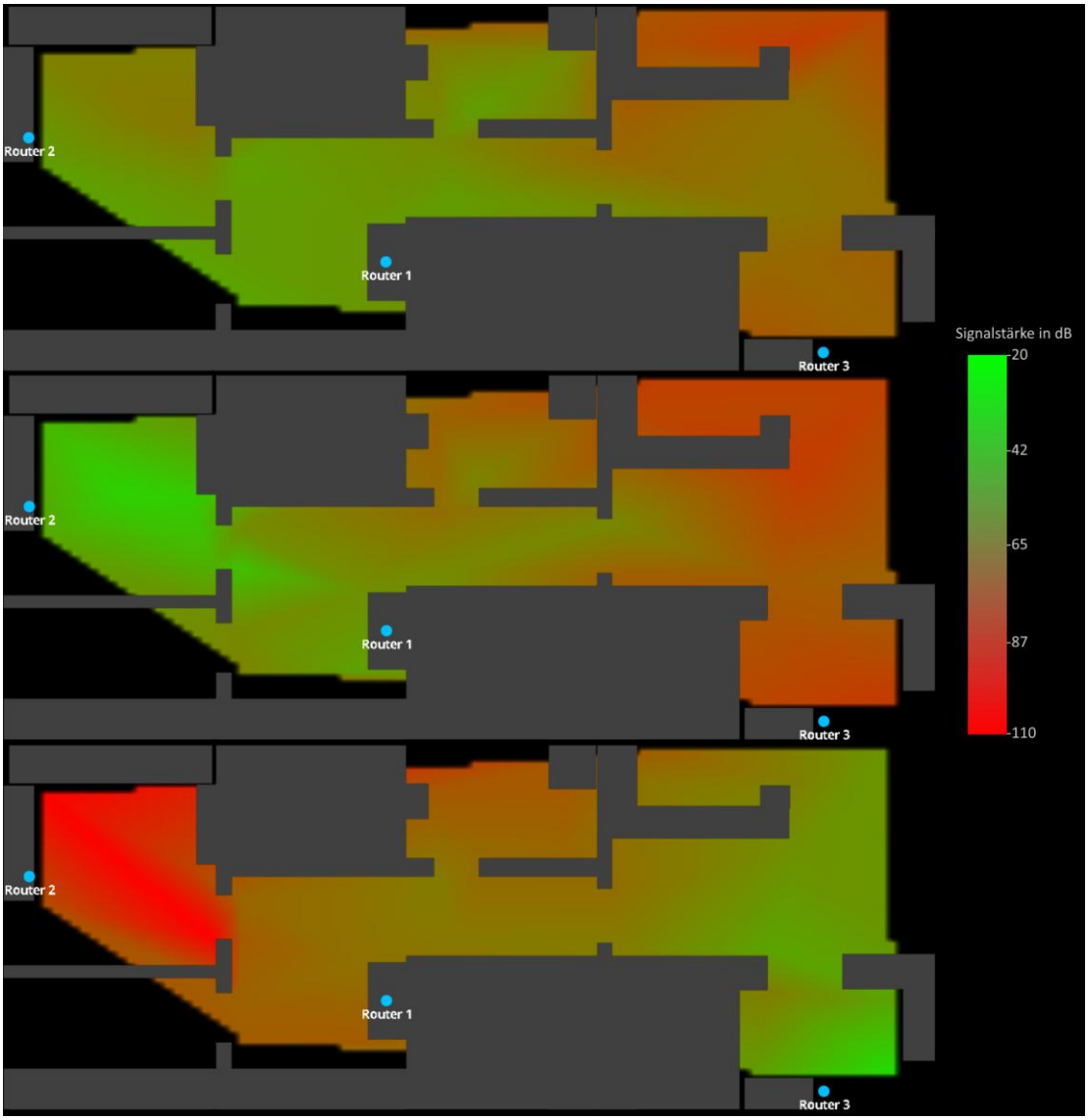

Abbildung 10: Heatmaps der Router in einem Stockwerk

Das Spektrum der Signalstärken der einzelnen Router verläuft sehr breitflächig durch das gesamte Stockwerk. Der Raum im Stockwerk links unten konnte nicht ausgemessen werden, daher enden die Heatmaps dort. Bei diesem Test wurden die folgenden Distanzen gemessen:

1.160m, 0.696m, 0.357m, 0.299m, 0.338m, 2.224m, 1.537m, 0.676m, 1.711m, 1.286m

Die Messung ergab eine durchschnittliche Genauigkeit von 1.028m mit einer Standardabweichung von 0.627m. Dieser Test zeigt eine Verbesserung gegenüber dem Test im Vorlesungsraum. In diesem Test wurde die höchste Genauigkeit gemessen sowie die niedrigste Standardabweichung. Zwei Faktoren haben sich gegenüber dem Test im

Vorlesungsraum geändert. Erstens zeigen die Heatmaps ein höheres Spektrum der Signalstärken. Im Test ohne die Antenne im Vorlesungsraum, war das Spektrum der Signalstärken ebenfalls breiter, jedoch musste aufgrund der niedrigen Reichweite des ESP32 der Laptop welcher als Server agiert durch den Raum verschoben werden. Daher hat sich auch die Position der Messperson im Raum geändert. Beim Test im Stockwerk konnte aufgrund der Antenne eine höhere Verbindungsreichweite des ESP32 erzielt werden. Daher musste die Position des Laptops nicht verändert werden.

6 Zusammenfassung und Ausblick

Es wird nun eine konkrete Antwort auf die Forschungsfrage gegeben. Die Forschungsfrage lautete:

Das Ziel dieser Arbeit ist die Entwicklung und Evaluierung eines WLAN-basierten Positionierungssystems zur präzisen Positionsbestimmung eines Empfängers in Innenräumen.

Die Tests haben gezeigt, dass es durchaus möglich ist einen Empfänger im Innenraum zu lokalisieren mithilfe von WLAN-Technologien. Der Aufbau der Experimente erfolgte über alltagsübliche Hardware, wie Smartphones und Laptops. Daher kann das System in bereits installierte WLAN-Systeme integrieren lassen. Des Weiteren haben die Tests gezeigt, dass eine durchschnittliche Genauigkeit von ca. 1 Meter möglich ist. Dennoch gibt es öfters Ausreißer mit Distanzen von ca. 3 Metern. Das System hat viele konfigurierbare Parameter, wie z.B. die Anzahl der Messpunkte oder die Anzahl der Messungen pro Messpunkt. Man kann mithilfe der Parameter einen „tradeoff“ zwischen Performance und Genauigkeit erreichen. Dennoch hat das System auch gezeigt, dass es sehr unterschiedliche Ergebnisse liefern kann, falls die Innenraumbedingungen sich ändern. Dynamische Objekte, wie Menschen, Türen, etc. beeinflussen die Performance des Systems deutlich. In dieser Arbeit wurden keine Möglichkeiten untersucht, wie man diese Faktoren minimieren kann. Es wäre vielleicht möglich bereits lokalisierte Gegenstände als Messpunkte zu nutzen, welche dynamisch die Heatmaps der Router aktualisieren. Ebenfalls wurde eine andere Methode zur Bestimmung der Position bereits implementiert, aber nicht weiter untersucht. Bei dieser Methode wird anstatt der minimalen Differenz der Heatmaps mehrer „Cluster“ gebildet. Es wird zuerst ein Schwellwert bestimmt und alle Punkte in der Distanz Heatmap, welche unter dem Schwellwert liegen werden rekursiv zusammen in Bereiche gruppiert, falls diese nebeneinander liegen. Diese werden dann als Cluster bezeichnet. Die Position ist dann der geometrische Mittelpunkt des größten Clusters. Diese Methode hat Randprobleme an den Heatmaps verbessert und hohe Distanzfehler verringert. Zusätzlich gibt es im WLAN-Standard eine Möglichkeit Zeitstempel zu erfassen. Diese benötigen aber Hardware, welche das Feature unterstützten und die Hardware, die für die Arbeit genutzt worden ist, hat dieses Feature nicht unterstützt. Es wäre also möglich, anstatt Signalstärken zu messen, welche durch viele Faktoren beeinflusst werden können, stattdessen Zeiten aufzunehmen und als Daten in den Heatmaps zu nutzen.

Abbildungsverzeichnis

Abkürzungsverzeichnis

SoC System on a Chip

WLAN Wireless Local Area Network

IEEE Institute of Electrical and Electronics Engineers

UDP User Datagram Protocol

TCP Transmission Control Protocol

Tabellenverzeichnis

Literaturverzeichnis

[1] A. Basiri, E. S. Lohan, T. Moore, A. Winstanley, P. Peltola, C. Hill, A. Pouria und P. Silva, „Indoor location based services challenges, requirements and usability of current solutions,“ Computer Science Review, Bd. 24, 01 04 2017.

[2] „Exxeta,“ [Online]. Available: https://exxeta.com/. [Zugriff am 24 4 2025].

[3] „Mapsted,“ [Online]. Available: https://mapsted.com/indoor-location-technologies/wifi-positioning. [Zugriff am 10 4 2025].

[4] „bitkom,“ [Online]. Available: https://www.bitkom.org/Presse/Presseinformation/WLAN-zu-Hause#:~:text=Das%20sind%20die%20Ergebnisse%20einer,Internethaushalte%20verbinden%20ihre%20Ger%C3%A4te%20drahtlos.. [Zugriff am 10 4 2025].

[5] „spiceworks,“ [Online]. Available: https://community.spiceworks.com/t/data-snapshot-wi-fi-security-in-the-workplace-and-beyond/970572. [Zugriff am 10 4 2025].

[6] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales und J. Fangmeyer Jr., „Evolution of Indoor Positioning Technologies: A Survey,“ Bd. 2017, 1 1 2017.

[7] „Max-PLANCK-INSTITUT,“ [Online]. Available: https://www.mpifr-bonn.mpg.de/563183/allgemeines#:~:text=Unter%20elektromagnetischen%20Wellen%20versteht%20man,erfolgt.. [Zugriff am 24 4 2025].

[8] „chemie.de,“ [Online]. Available: https://www.chemie.de/lexikon/Bel_%28Einheit%29.html. [Zugriff am 24 4 2024].

[9] „Elektronik Kompendium,“ [Online]. Available: https://www.elektronik-kompendium.de/sites/net/1712061.htm. [Zugriff am 24 4 2025].

[10] U. Siart, „Das Dezibel – Definition und Anwendung,“ 13 6 2014.

[11] S. Burrell, „wray castle,“ 9 2024. [Online]. Available: https://wraycastle.com/de/blogs/knowledge-base/rssi-signal-strength. [Zugriff am 24 4 2025].

[12] „kaspersky,“ [Online]. Available: https://www.kaspersky.de/resource-center/definitions/what-is-an-ssid. [Zugriff am 24 4 2025].

[13] „Elektrik Kompendium,“ [Online]. Available: https://www.elektronik-kompendium.de/sites/net/2010231.htm. [Zugriff am 24 4 2025].

[14] nayarasi, „mrn-cciew,“ [Online]. Available: https://mrncciew.com/2014/10/27/cwap-802-11-probe-requestresponse/. [Zugriff am 24 4 2025].

[15] „cisco,“ [Online]. Available: https://www.cisco.com/c/de_de/support/docs/wireless-mobility/wireless-lan-wlan/82068-omni-vs-direct.html. [Zugriff am 2025 4 24].

[16] K. R. a. S. W. R. Fall, Tcp/ip illustrated, Addison-Wesley Professional, 2012.

[17] „Elektronik Kompendium,“ [Online]. Available: https://www.elektronik-kompendium.de/sites/net/0812281.htm. [Zugriff am 24 4 2025].

[18] „Elektronik Kompendium,“ [Online]. Available: https://www.elektronik-kompendium.de/sites/net/0812271.htm. [Zugriff am 24 4 2025].

[19] O. R. Musin, „Properties of the Delaunay triangulation,“ in *Proceedings of the thirteenth annual symposium on Computational geometry*, 1997, pp. 424-426.

[20] L. Chen und J.-c. Xu, „OPTIMAL DELAUNEY TRIANGULATIONS,“ *Journal of Computational MAthematics,* Bd. 22, Nr. 2, pp. 299-308, 31 1 2004.

[21] E. W. Weisstein, „Barycentric coordinates,“ *https://mathworld. wolfram. com/,* 2003.

[22] „espressif.com,“ espressif, [Online]. Available: https://www.espressif.com/en/products/socs/esp32. [Zugriff am 24 4 2025].

[23] „reichelt,“ [Online]. Available: https://www.reichelt.de/de/de/shop/produkt/wifi-smd-modul_esp32-s2_4_mb_spi_3_3_v_18_x_31_x_3_3_mm-300166. [Zugriff am 12 4 2025].

[24] „espressif docs,“ [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/lwip.html. [Zugriff am 12 4 2025].

[25] H. A. B. a. O. Sulaiman, M. A. a. Aziz, M. a. Bade und Abdullah, „Implementation of axis-aligned bounding box for opengl 3D virtual environment,“ *ARPN Journal of Engineering and Applied Sciences,* Bd. 10, Nr. 2, pp. 701-708, 2015.