
TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

# MASTERARBEIT

## Lokalisierung mit Bluetooth mittels neuronaler Netze

Ausgeführt am Institut für
Distributed and Multimedia Systems
der Universität Wien

unter der Anleitung von
Ao. Univ.-Prof. Dr. Helmut Hlavacs

durch

Stefan Schneider
Bründlfeldweg 60
7000 Eisenstadt

8. Juli 2008

## Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich diese Arbeit ohne die Hilfe Dritter erstellt habe. Alle Stellen, die direkt oder indirekt aus fremden Quellen übernommen wurden, sind als solche gekennzeichnet. Diese Arbeit wurde in gleicher oder ähnlicher Form noch nie einer Prüfungsbehörde vorgelegt.

Wien, im Juli 2008

# Danksagung

Mein Dank gilt meinen Eltern, die mich während meiner Studienzeit in allen Belangen unterstützten.

Außerdem danke ich meinen Freunden Martin Kohl, Michael Pekovits, Wolfgang Roth, Thomas Schnöller und meinem Bruder Michael Schneider für die Bereitstellung der Hardware.

Besonderer Dank gilt meinem Betreuer, Herrn Univ.-Prof. Dr. Helmut Hlavacs, der mir stets mit motivierenden Gedanken zur Seite stand.

# Kurzfassung

In dieser Arbeit wird ein neuer Ansatz für ein Indoor-Positioning-System, basierend auf Bluetooth, vorgestellt. Im Gegensatz zu anderen Arbeiten auf diesem Gebiet wird die Position des zu lokalisierenden Mobilen Gerätes (MD) nicht über Trilateration bestimmt, sondern mit Hilfe eines künstlichen neuronalen Netzes (KNN). Dieser Ansatz wurde gewählt, da Funkwellenausbreitung in Gebäuden durch Hindernisse stark beeinträchtigt wird. Das führt dazu, dass die Position eines MD über Trilateration nur sehr ungenau bestimmt werden kann. Das KNN soll aus vorher genommenen Messwerten Informationen über die Umgebung lernen und so in der Lage sein, ein MD genauer lokalisieren zu können.

Dieser Ansatz wurde in einem Experiment in einer Altbauwohnung untersucht. Dazu wurde ein auf freier Software (Linux, BlueZ und Java) basierendes Messsystem entwickelt. Dieses besteht aus mehreren Base Stations (BS), von denen jede mit einem Standard Bluetooth-Modul (MSI BToes 2.0) ausgestattet wurde, und einem Server, welcher die BSs verwaltet. Mit diesem System wurden Messwerte gesammelt, die für das Training der KNNs verwendet wurden. Es konnte gezeigt werden, dass MDs der Bluetooth-Geräteklasse 2 unter Idealbedingungen mit einem durchschnittlichen Fehler von 5,4cm lokalisiert werden konnten. Die Position von MDs der Bluetooth-Geräteklasse 3 konnte unter Idealbedingungen mit einem durchschnittlichen Fehler von 1,9cm bestimmt werden.

## Abstract

This paper discusses a new approach for an indoor positioning system based on Bluetooth. In contrast to other works from this area the position from the mobile device isn't determined with a trilateration algorithm but through an artificial neural network (ANN). This approach was chosen because the radiowave propagation in buildings is affected through many obstacles. On this account the position from a MD can only be imprecisely estimated with a trilateration algorithm. The ANN should learn about its environment from previously taken measured values and be so able to determine the position of the MD more precisely.

This approach was examined in an experiment. For this purpose a measurement system was developed based on free software (Linux, BlueZ and Java). The measurement system consists of several Base Stations (BS), each of them was equipped with a standard Bluetooth-Modul (MSI BToes 2.0), and a server which administrated the BSs. With this system measured values were taken to train the ANNs. We could show that MDs Class 2 can be located with a mean error of 5,4cm under perfect circumstances. The position from MDs Class 3 can be determined with a mean error of 1,9cm.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| ACL | Asynchronous Connection Less |
| ANN / KNN | Artificial Neural Network / Künstliche neuronale Netze |
| API | Application Programming Interface |
| AM_ADDR | Active Member |
| BD-ADDR | Bluetooth Device Address |
| BS | Base Station |
| BSD | Berkely System Distribution |
| CID | Channel Identifier |
| CLK | Clock |
| CVSD | Continuous Variable Slope Delta Modulation |
| GRPR | Golden Received Power Range |
| GPS | Global Positioning System |
| HCI | Host Controller Interface |
| ISM | Industrial, Scientific and Medical |
| JNI | Java Native Interface |
| KNN / ANN | Künstliche neuronale Netze / Artificial Neural Network |
| LM | Link Manager |
| LMP | Link Manager Protocol |
| LQ | Link Quality |
| MD | Mobile Device |

| | |
|---|---|
| MLFF | Multilayer Feedforward Neural Network |
| MTU | Maximum Transfer Unit |
| PSM | Protocol/Service Multiplexer |
| RD | Reference Device |
| RFCOMM | Radio Frequency Communication |
| RMI | Remote Method Invocation |
| RSSI | Received Signal Strength Indicator |
| SCO | Synchronous Connection Oriented |
| SDK | Software Development Kits |
| TPL | Transmitted Power Level |

# 1 Einleitung

Handys, PDAs und andere mobile informationsverarbeitende Geräte sind zu unseren alltäglichen Begleitern geworden. Die Bestimmung des Aufenthaltsorts des Benutzers eines mobilen Geräts ist eine wichtige Aufgabe, da sie die Grundlage für Location Based Services bildet. Location Based Services bieten dem Benutzer nützliche Dienste an, passend zu seinem Aufenthaltsort.

Das Global Positioning System (GPS) ist momentan das meist genutzte System zur Positionsbestimmung. GPS ist jedoch nur im Freien nutzbar und kann somit nicht für die Positionsbestimmung in Gebäuden (Indoor Positioning) genutzt werden. Der Einsatz von Location Based Services ist jedoch auch innerhalb von Gebäuden interessant. Beispiele dafür wären unter anderem ein Einkaufsassistent in großen Kaufhäusern oder ein elektronischer Museumsführer, der dem Benutzer Informationen zu dem Ausstellungsobjekt anbietet, vor dem er sich befindet. Die Telekommunikations- und Softwareindustrie setzt große Hoffnung in solche Systeme. Es gibt bereits Systeme für die Positionsbestimmung in Gebäuden. Sie alle verwenden jedoch spezielle Hardware, was sich in hohen Anschaffungskosten niederschlägt. Aus diesem Grund konnten sich diese Systeme auch nicht auf breiter Front durchsetzen.

Eine billigere Möglichkeit, Indoor Positioning zu ermöglichen, sind lokale Funknetze wie z.B. Bluetooth. Diese Technologie weist mehrere Eigenschaften auf, die sie dafür interessant macht. Sie ist weit verbreitet, und es müssen keine baulichen Maßnahmen in den Gebäuden durchgeführt werden, um sie verwenden zu können. Außerdem ist sie günstig in der Anschaffung. Viele Leute verfügen bereits über Handys, PDAs und andere mobile Geräte, die mit Bluetooth ausgestattet sind, und sie tragen diese meist bei sich. Alle diese Punkte sprechen dafür, dass sich ein Indoor Positioning System auf Bluetooth-Basis am Markt durchsetzen könnte.

In den letzten fünf Jahren wurden einige Arbeiten zu diesem Thema ausgeführt. Sie unterscheiden sich in der Genauigkeit, mit der die Position bestimmt werden kann und in den Methoden, mit denen die Lokalisierung vorgenommen wurde. Die Grundlage für die Lokalisierung bildet aber immer eine Distanzbestimmung zwischen Base Stations (BS), deren Positionen bekannt sind, und dem Mobilen Device (MD), dessen Position bestimmt werden soll. Diese Distanzbestimmung erfolgt auf Basis der Funkwellenausbreitung. Diese wird aber in Gebäuden durch verschiedene Hindernisse wie Wände, Türen und Möbel beeinträchtigt, was sich auf die Genauigkeit, mit der die Position bestimmt werden kann, negativ auswirkt. In den uns bekannten Ar-

beiten, in denen die Experimente in Gebäuden durchgeführt wurden, konnte die Position mit einer Genauigkeit von ca. 3m bestimmt werden.

Die bisherigen Ansätze werden durch die Umgebung stark beeinflusst. In unserem Ansatz versuchen wir, uns der Umgebung anzupassen, indem wir für die Lokalisierung ein neuronales Netz verwenden. Dazu führen wir an unterschiedlichen Positionen Signalstärkemessungen durch. Das MD ist dabei mit mehreren BSs verbunden. Von jeder Verbindung wird die Signalstärke bestimmt. Die Idee ist, auf diese Weise für jede Position eine Reihe von Werten zu erhalten, die gemeinsam die Position eindeutig identifizieren. Mit den Signalstärkewerten, die an den verschiedenen Punkten genommen wurden, wird dann ein neuronales Netz trainiert. Werden dann aktuell gemessene Signalstärkewerte als Input an das trainierte neuronale Netz angelegt, sollte es in der Lage sein, die momentane Position des MDs zu bestimmen.

Der Nachteil an diesem Ansatz besteht darin, dass ein MD nur in der trainierten Umgebung lokalisiert werden kann. Wir erhoffen uns aber, in der trainierten Umgebung die Position wesentlich genauer bestimmen zu können, als bei den uns bekannten Arbeiten.

Der Ansatz wurde in einem Experiment untersucht. Dabei sollte die Position eines MDs mittels eines künstlichen neuronalen Netzes auf 50cm genau bestimmt werden. Um die dafür notwendigen Signalstärkemessungen durchführen zu können, wurde ein Messsystem entwickelt. Dieses basiert auf freier Software und verwendet Standard Bluetooth-Hardware, um so dem Kriterium einer kostengünstigen Anschaffung gerecht zu werden.

## 1.1 Gliederung

Diese Arbeit ist in zwei Hauptteile gegliedert. Im Teil eins wird auf die zugrunde liegende Technologie und Theorie näher eingegangen. So werden zu Beginn dieses Teils in Kapitel 2 drei Arbeiten, die ebenfalls zum Thema „Lokalisierung mit Bluetooth“ durchgeführt wurden, präsentiert.

In Kapitel 3 wird die Funktionsweise von Bluetooth beschrieben. Weiters beschäftigt sich dieses Kapitel mit verschiedenen Techniken der Lokalisierung und wie diese mit Bluetooth genutzt werden können. Die Methode, ein MD über die Signalstärke zu lokalisieren, scheint die bestmögliche. Es folgt eine Analyse, wie sich diese Technik in der Praxis umsetzen lässt. Die Analyse umfasst mehrere freie Bluetooth-

Softwarestacks sowie die Beschreibung einer geeigneten Klassenbibliothek, die für die Entwicklung der dafür benötigten Messsoftware verwendet werden kann.

Kapitel 4, zum Abschluss des theoretischen Teils, gibt einen Einblick in die Funktionsweise künstlicher neuronaler Netze. Das Training mit und die Auswertung von neuronalen Netzen wird an dieser Stelle ebenfalls besprochen.

Im zweiten Teil stehen die praktischen Tätigkeiten im Vordergund, die im Rahmen dieser Arbeit ausgeführt wurden. Es werden Architektur und Funktionsweise der Software beschrieben, welche für die Durchführung der Bluetooth-Messungen implementiert wurden. In weiterer Folge werden die Experimente zur Auswahl geeigneter Bluetooth-Dongles, sowie zur Erhebung der Messdaten für die Lokalisierung beschrieben. Hier wird auch gezeigt, wie die Umgebung die Funkwellenausbreitung behindert. Am Ende des praktischen Teiles werden die Experimente zur Lokalisierung der MDs mit neuronalen Netzen beschrieben und die Ergebnisse diskutiert.

In der Conclusio werden die Ergebnisse und die daraus gezogenen Erkenntnisse des gesamten Projekts zusammengefasst und Vorschläge für weiterführende Arbeiten gemacht.

Ein Benutzerhandbuch zur Messsoftware und zu anderen Hilfsprogrammen, die während der Arbeit erstellt wurden, befindet sich im Anhang. Weiters sind an dieser Stelle die Matlab-Skripts, die zur Bearbeitung und Analyse der gesammelten Daten verwendet wurden, beschrieben.

## 1.2 Begriffsdefinitionen

*In diesem Punkt werden einige Begriffe klar definiert, die für das Verständnis der Arbeit essentiell sind.*

Position: lokaler Punkt, an dem sich eine Komponente des Funknetzes befindet.

Base Station (BS), Reference Device (RD): beschreiben eine Komponente in einem Funknetz, deren Position bekannt ist und nicht verändert wird. Sie werden in dieser Arbeit als Synonyme verwendet.

Mobile Device (MD): bezeichnet eine mobile Komponente in einem Funknetz, deren Position nicht bekannt ist.

Lokalisierung: beschreibt den Vorgang, in welchem die Position eines Mobile Device anhand von Base Stations/Reference Devices bestimmt wird.

Signalstärke, Signalstärkewert: beschreiben die Qualität eines empfangenen Funksignals. Je höher der Wert desto besser das Funksignal. Bei Bluetooth gibt es keinen eindeutig definierten Signalstärkewert. Es gibt drei Parameter, die für die Berechnung eines Signalstärkewerts in Frage kommen. Das sind: Received Signal Strength Indicator (RSSI), Transmit Power Level (TPL), Link Quality (LQ). Der Signalstärkewert wird anhand von einem oder mehreren dieser Werte berechnet.

# 2 Verwandte Arbeiten

*Die ersten Arbeiten zum Thema „Lokalisierung mit Bluetooth“ wurden im Jahr 2003 veröffentlicht. Alle Artikel, die zu Recherchezwecken für diese Arbeit herangezogen wurden, haben gemeinsam, dass sie die Signalstärke verwenden, um ein Bluetooth-Gerät zu lokalisieren. In der Regel wird auf Basis der gemessenen Signalstärkewerte eine Distanzbestimmung vorgenommen und dann die Position des MD mittels Trilateration berechnet. In diesem Kapitel werden drei ausgewählte Arbeiten vorgestellt, die zum Teil andere Methoden verwenden und einen Überblick über den aktuellen Forschungsstand im Bereich der Lokalisierung mit Bluetooth geben. „Experiments on Local Positioning“ war eine der ersten veröffentlichten Arbeiten in diesem Bereich. „Atlantis: Location Based Services with Bluetooth“ ist ein mehrfach von anderen Arbeiten referenzierter Artikel, und das zugrunde liegende Lokalisierungsnetzwerk hat eine ähnliche Architektur wie jenes dieser Arbeit. Weiters läuft das System ebenfalls unter Linux. Die Autoren der Arbeit „Bluetooth Base Station Minimal Deployment for High Definition Positioning“ versuchen, ebenso wie es in dieser Arbeit geschieht, die Position des zu lokalisierenden Gerätes so genau wie möglich zu bestimmen.*

## 2.1 Experiments on Local Positioning with Bluetooth

Dieser Artikel aus dem Jahr 2003 gehört zu den ersten Arbeiten, die zum Thema Lokalisierung mit Bluetooth veröffentlicht wurden, und er wird auch von mehreren anderen Arbeiten aus diesem Bereich als Referenz geführt. Um ihre Experimente durchführen zu können, haben die Wissenschaftler der Tampere University of Technology in Finnland Antti Kotanen, Marko Hännikäinen, Helena Leppäkoski und Timo D. Hämäläinen ihre eigene Software geschrieben.

Diese Software läuft auf dem MD. Die Positionen der RDs im Raum sind in der Software manuell konfiguriert. Es ist auch möglich, die Startposition des MDs einzugeben. Das MD baut zu jedem RD in Sendeweite eine ACL-Verbindung auf und fungiert als Master im entstehenden Piconetz. Anschließend misst das MD den RSSI-Wert jeder einzelnen Verbindung lokal und über einen Remote-Befehl beim RD. Die gemessenen Werte verwendet es, um die Distanz zwischen sich und dem RD zu berechnen. Dazu bedienen sich die Wissenschaftler einer umgeformten Funktion zur Funkwellenausbreitung. Die so ermittelten Distanzen zu den einzelnen RDs

werden dann dazu verwendet, mit dem erweiterten Kalman-Filter die Position des MDs zu bestimmen. Damit eine Positionsbestimmung möglich ist, muss das MD zumindest mit drei RDs verbunden sein.

Als Bluetooth-Modul wurde das Ericsson Bluetooth Starter Kit verwendet, das über ein USB Kabel an den Host angeschlossen werden kann. Die eingebauten Antennen wurden aber durch externe des Herstellers M/A-COM ersetzt, da diese geringere Schwankungen bei der gemessenen Signalstärke aufweisen. Als Bluetooth-Softwarestack diente der Bluetooth PC Reference Stack (Version R2B), der von Ericsson zur Verfügung gestellt wurde. Dieser wurde unter Windows 2000 installiert. Die während des Projekts erstellte Software wurde mit Microsoft Visual C++ 6.0 entwickelt.

Die Experimente wurden in typischen Büros durchgeführt. Um unnötige Interferenzen und eine Verfälschung der Ergebnisse zu vermeiden, wurden WLAN-Geräte in der Umgebung deaktiviert.

Im Laufe des Experiments wurden 50 Messungen mit der erstellten Software durchgeführt. Bei jeder Messung wurden die X- und Y-Position des MDs verändert. In Abbildung 1 sieht man das Ergebnis des Experiments. Die Positionen der RDs sind in der Abbildung mit x dargestellt. Die tatsächliche Position des MDs während der Messung wird durch die nicht ausgefüllten Kreise angegeben. Diese sind mit einer Linie zur Position des MDs verbunden, die durch die Software errechnet wurde; dargestellt durch die ausgefüllten Kreise. Links sieht man in Abbildung 1 die Ergebnisse der ersten 25 Messungen. Dabei befand sich das MD in einer Höhe von 0,75m. Für die Messungen 26-50, rechts dargestellt, wurde die Position der RDs verändert, und das MD befand sich auf dem Boden.

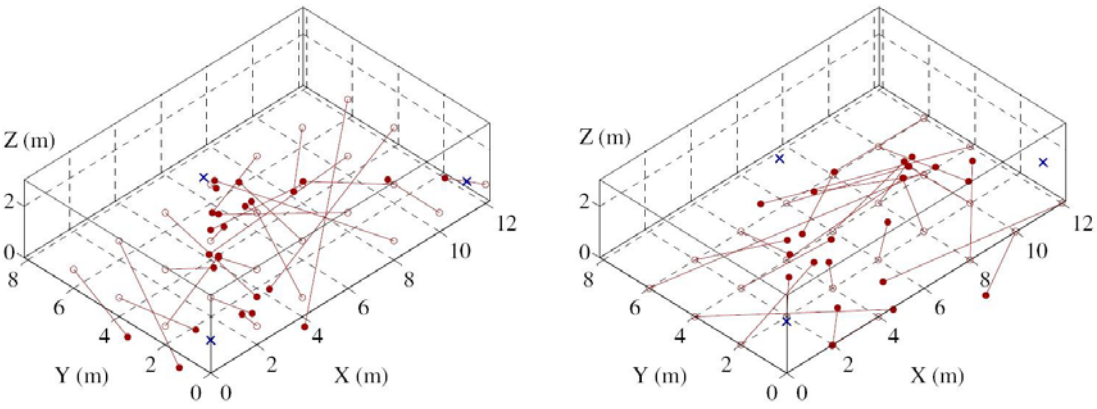


Abbildung 1: Ergebnisse des Experiments [ANT03][6]

Bei jedem der 50 Messpunkte wurde der RSSI Wert 5000mal pro RD durch das MD gemessen und auf der Festplatte gespeichert. Mit diesen Daten wurden die Parameter des erweiterten Kalman-Filters so optimiert, dass der absolute durchschnittliche Fehler zwischen tatsächlicher und berechneter Position minimiert wurde. Der absolute durchschnittliche Fehler für alle Messungen beträgt 3,76m.

Den Grund dafür, dass die Lokalisierung nicht genauer durchgeführt werden kann, sieht man vor allem in der Unzuverlässigkeit des RSSI-Wertes. Es wird empfohlen, auch andere Vorhersagensmodelle zu testen und die Anzahl von RDs zu erhöhen. Es sind jedoch keine wesentlich besseren Resultate zu erwarten, da der RSSI-Wert weiter unzuverlässig ist [ANT03].

## 2.2 Atlantis: Location Based Services with Bluetooth

Jason Yipin Ye erstellte im Rahmen seiner Diplomarbeit an der Brown University ein Bluetooth-Framework, das es erlaubt, Location-Based-Services anzubieten. Dieses System wurde in das Atlantis Framework integriert.

Das Bluetooth-Framework läuft unter Debian Linux und verwendet BlueZ als Bluetooth-Softwarestack. Als Programmiersprache wurde Java verwendet und mittels JBlueZ[1] auf den Stack zugegriffen. Da die JBlueZ API keinen Zugriff auf die Signalstärkeparameter ermöglicht, musste sie mittels JNI erweitert werden. Wie in Punkt 3.8 beschrieben, benötigt das ausführende Programm für einen solchen Zugriff Super-User Rechte. Dieses Problem wurde hier durch die Einrichtung einer speziellen Benutzergruppe geregelt.

Die Basis des Systems bildet eine Reihe von Rechnern, die alle mit demselben Bluetooth-Modul ausgestattet sind, in Abbildung 2 als BASE A, B und C dargestellt. Auf diesen Rechnern, die als Base Stations fungieren, läuft ein Java-Programm, das über Java Remote Method Invocation (RMI) dem zentralen Server Bluetooth-Funktionen anbietet. Der Server kann die BSs dazu veranlassen, nach MDs zu suchen, Verbindung zu einem MD aufzubauen oder zu unterbrechen, sowie den RSSI, TPL und LQ-Wert von Verbindungen zu messen. Das BTGateway verwaltet das Sensornetzwerk und kapselt es so für den Atlantis Server in einem Objekt. Der Atlantis Server muss dann nur das BTGateway auffordern, Signalstärkedaten zu liefern. Das BTGateway

[1] http://jbluez.sourceforge.net/

gibt diese Aufforderung an die BSs weiter, welche dann nach Bluetooth-Geräten suchen und sich zu diesen verbinden. Sobald eine BS mit einem MD verbunden ist, initiiert sie einen Rollentausch. Auf diese Art wird das MD Master und kann so mit bis zu sieben BSs verbunden sein. Die BSs liefern die Signalstärkewerte zurück, welche vom Atlantis Server an den BTLocalizer weitergegeben werden. Dieser sammelt die Daten und normalisiert sie, um so gegen Fehlmessungen, hervorgerufen durch kurzzeitige Interferenzen, resistenter zu sein.

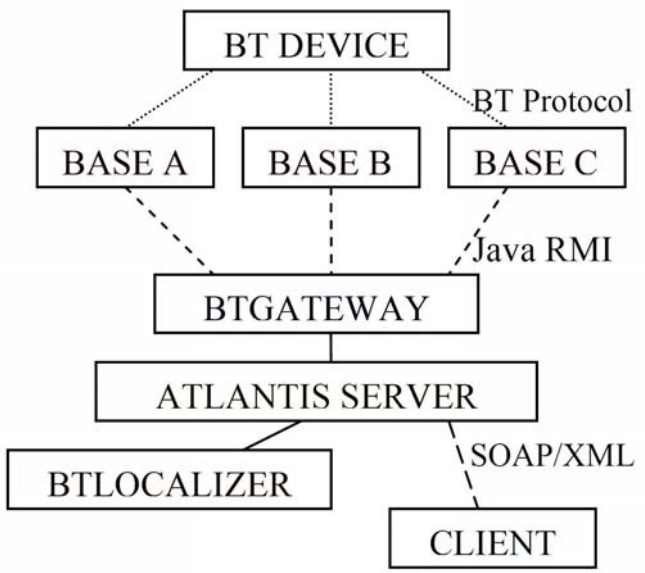


Abbildung 2: Atlantis Localization Framework [JAS05][5]

Der BTLocalizer führt dann eine Distanzschätzung zwischen dem MD und jeder verbundenen BS mit Hilfe einer Heuristik durch, deren genauer Ablauf jedoch wegen eines laufenden Patentverfahrens im Artikel geschwärzt wurde. Annahme des Autors: „Da jedoch unmittelbar vor dem geschwärzten Absatz von drei Parametern, die über die Signalstärke Aufschluss geben und über den HCI Layer ausgelesen werden können, die Rede ist, kann angenommen werden, dass es sich um die RSSI, LQ und TPL Werte handelt, die in der Heuristik verwendet werden. Ein weiterer Umstand, der dafür spricht, ist, dass die entwickelte Base Station nur auf diese drei HCI Parameter den Zugriff ermöglicht“. Sobald Distanzschätzungen von drei oder mehr BS zu dem MD vorhanden sind, kann die Position des MD mittels Trilateration bestimmt werden. Falls die geschätzten Distanzwerte keine Überschneidung ergeben, werden sie solange erhöht, bis diese möglich ist oder die Grenzen des überwachten Gebiets erreicht sind. Die so errechnete Position wird an den Atlantis Server übergeben, der überprüft, ob sich der MD in einem Bereich befindet, in dem ein Service angeboten wird. Falls ja, stellt er dem MD diesen Service zur Verfügung.

Das System wurde dann in einem Experiment getestet. Man wollte es in einer möglichst großen Umgebung zu Alltagsbedingungen testen. Dazu wurden an sechs BSs Cambridge Silicon Dongles der Sendeklasse 1 angebracht (Sendereichweite 100m siehe Punkt 3.2.2.). Die sechs BSs wurden auf einer Etage eines Bürogebäudes verteilt. Es wurden keine störenden Elemente, wie z.B. Möbel, entfernt. Es wurden zwei Durchläufe mit unterschiedlichen MDs unternommen, um festzustellen, wie gut das System mit unterschiedlichen Geräten zurechtkommt. Die Dongles, die als MDs dienten, wurden an einen Laptop mit Windows angeschlossen, auf dem der Standard Microsoft Bluetooth-Stack installiert war.

Die Versuche zeigten, dass die Position des MD umso besser bestimmt werden konnte, je mehr BSs an der Ortung beteiligt waren. BSs, die dem MD am nächsten waren, lieferten bessere Distanzschätzungen als weiter entfernte. In Umgebungen, in denen die BSs weniger durch Mobiliar und andere störende Elemente behindert wurden, konnten ebenfalls bessere Ergebnisse erzielt werden. Der Fehler bei der Positionsbestimmung wird mit ca. 7m angegeben.

Die Ergebnisse führt man auf die Verwendung von Low End Dongles zurück und die schwierige Umgebung, in der getestet wurde. Weiters denkt man an den Einsatz von künstlicher Intelligenz, um auf die Umgebung einer BS, die die Signalausbreitung beeinträchtigen könnte, reagieren zu können [JAS05].

## 2.3 Bluetooth Base Station Minimal Deployment for High Definition Positioning

In dieser Arbeit aus dem Jahr 2005, von Alessandro Genco, Salvatore Sorce und Giuseppe Scelfo, versucht man das MD so genau wie möglich zu lokalisieren und dabei die Anzahl der BSs zu minimieren. Um das zu erreichen, wurde ein genetischer Algorithmus verwendet.

Die Arbeiten wurden in einer mittelalterlichen Ruine in Sizilien durchgeführt (Abbildung 3).

Abbildung 3: Manfredis Ruine in Sizilien [ALE05][3]

Wie man in Abbildung 3 sehen kann, wurde das Gelände in Zellen unterteilt. Die roten Punkte markieren jene Stellen, an denen BSs angebracht werden durften. Blaue Punkte markieren spezielle BSs, die an Durchgängen oder in der Nähe von speziellen Gegenständen angebracht wurden. Aufgrund der Größe des Areals wurden zunächst nur Messungen im oberen Bereich der Ruine, in der Abbildung durch eine grüne Linie getrennt, durchgeführt.

Dabei wurde die LQ von den einzelnen BSs zum MD aufgezeichnet. Da man eine minimale Anzahl von BSs erreichen wollte, wurde anhand der aufgezeichneten Daten geprüft, mit welcher Anordnung von BSs man die eindeutigsten Messwerte und die höchste Genauigkeit erzielen kann. Jede mögliche Anordnung wurde durch ein Chromosom dargestellt, das für jede Möglichkeit, an der eine BS positioniert werden kann, ein Gen enthält. Das in Abbildung 4 gezeigte Chromosom umfasst acht mögliche Positionen für BSs, wovon drei gesetzt sind. Diese werden im Chromosom mit 1 markiert.

| **0** | **0** | **0** | **1** | **1** | **0** | **0** | **1** |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Abbildung 4: Chromosom [ALE05][4]

Für jede durch ein Chromosom dargestellte Anordnung kann über eine Fitness-Funktion ein Qualitätsindex errechnet werden. Dieser ist davon abhängig, wie gut das MD lokalisiert werden kann, und wie viele BSs dafür notwendig sind. Dabei

wird eine maximale Genauigkeit für die Lokalisierung und eine minimale Anzahl von BSs angestrebt.

Zu Beginn des genetischen Algorithmus wurde eine Population von 50 Chromosomen generiert, wobei jede BS die gleiche Wahrscheinlichkeit *p* hatte, Teil eines Chromosoms zu sein. Für jedes Chromosom wird sein Qualitätsindex errechnet. Während jedes Evolutionsschrittes werden 10 Chromosomen gelöscht, abhängig von ihrem Alter. Durch Mischung der überlebenden Genome werden 10 neue Chromosomen angelegt. Es werden 1000 Evolutionsschritte durchgeführt. In jedem Schritt wird nach der besten Lösung gesucht. Wenn diese besser ist als die zuvor gefundene, wird sie gespeichert. So wurde die beste Lösung ermittelt.

Es wurden mehrere Versuche mit dem genetischen Algorithmus durchgeführt, wobei die Wahrscheinlichkeit *p* und die geforderte Genauigkeit für die Lokalisierung verändert wurden. Es wurde mit einer Wahrscheinlichkeit $p$=10 und einer geforderten Genauigkeit von 100cm gestartet. Diese konnte bereits mit 10 BSs erreicht werden. Das beste Ergebnis erreichte man mit einer Wahrscheinlichkeit von $p$=30 und 16 BSs. Hier betrug die Genauigkeit 37,5cm [ALE05].

## 2.4 Kapitelzusammenfassung

Die Basis zur Lokalisierung bildet in den drei vorgestellten Arbeiten die Signalstärke. Es wurden jedoch unterschiedliche Parameter für die Berechnung verwendet, wie man in Tabelle 1 in der Spalte „verwendete Parameter“ sehen kann. Die Autoren der Arbeiten sind sich darüber einig, dass die RSSI-, LQ- und TPL-Parameter bei Bluetooth-Geräten unterschiedlich genau sind, abhängig vom Hersteller, da die Bluetooth-Spezifikation hier einige Freiräume bietet. In der Unzuverlässigkeit dieser Parameter wird auch der Grund für die Ungenauigkeit bei der Lokalisierung gesehen. Es kamen auch alle zu dem Schluss, dass die Position, an denen die BSs aufgestellt werden, einen wesentlichen Einfluss auf die Genauigkeit hat.

Weitere Unterschiede zwischen den vorgestellten Arbeiten waren die angestrebte Genauigkeit, mit der man das MD lokalisieren wollte, und die Umgebung, in der die Experimente durchgeführt wurden. In der aus Italien stammenden Arbeit konnte auf einer mittelalterlichen Ruine mit einem genetischen Algorithmus die Position des MD mit einer beeindruckenden Genauigkeit von 0,375m bestimmt werden. Das Atlantis-Framework erreichte mit Trilateration zwar nur eine Genauigkeit von 7m. Jedoch wurde hier mit sechs BSs die gesamte Etage eines Bürogebäudes überwacht.

Die Testumgebung wurde auch nicht verändert, da man das System unbedingt unter Alltagsbedingungen erproben wollte.

| **Errechnung der Position durch** | **Verwendete Parameter** | **System** | **Lokalisierung über** | **Testumgebung** | **Genauigkeit** |
|---|---|---|---|---|---|
| erweiterten Kalman-Filter | RSSI | Windows 2000/ Ericsson BT Starter Kit | MD/zentral | präpariertes Büro | 3,76m |
| Trilateration | RSSI, LQ, TPL[2] | Debian Linux /Class1 BT-Dongles | BSs/dezentral | Etage in einem Bürogebäude | 7m |
| genetischer Algorithmus | LQ | Standard Komponenten | BSs/dezentral | mittelalterliche Ruine | 0,375m |

Tabelle 1: Übersicht: Verwandte Arbeiten

Die Arbeit „Experiments on Local Positioning with Bluetooth" ist die einzige, in der Handset Based-Ansatz verwendet wird. Dem MD sind die Positionen der RDs bekannt, und es berechnet selbst seine Position anhand des erweiterten Kalman Filters. In den anderen beiden Arbeiten wird ein Network Based-Ansatz verwendet, um das MD zu lokalisieren.

Diese drei Arbeiten zeigen ganz klar, dass die Lokalisierung eines Bluetooth-Gerätes über die Signalstärke möglich ist. Wie dieses Kapitel gezeigt hat und man in Tabelle 1 sehen kann, gibt es mehrere Lösungsansätze, die von verschiedenen Komponenten beeinflusst werden.

[2] Annahme des Autors, da entsprechender Teil in Quelle geschwärzt

# 3 Bluetooth

*In diesem Kapitel wird die Bluetooth-Technologie vorgestellt. Es beginnt mit einer Einführung in Bluetooth. Im Anschluss daran wird die allgemeine Funktionsweise von Bluetooth anhand der einzelnen Layers des Bluetooth-Protokollstacks beschrieben. Es werden weiters die Anwendungsprofile, die ein wichtiger Bestandteil der Bluetooth-Technik sind, sowie die Netzwerktopologie von Bluetooth besprochen. Danach werden mehrere Möglichkeiten zur Lokalisierung im Funknetz vorgestellt und eine mögliche technische Umsetzung mit Bluetooth diskutiert.*

*Die Methode, anhand der Signalstärke die Lokalisierung vorzunehmen, kann mit Bluetooth am besten umgesetzt werden. Da nicht jeder Bluetooth-Softwarestack (BT-SW-Stack) einen Zugriff auf die Signalstärke Parameter erlaubt, werden drei BT-SW-Stacks diesbezüglich untersucht. Mit dem Ergebnis, dass der BlueZ BT-SW-Stack unter Linux sich für eine Realisierung dieser Methode eignet. Die JToot-Library ermöglicht eine komfortable Nutzung von BlueZ und bildet so die Grundlage für die Messsoftware. Sie wird am Ende dieses Kapitels vorgestellt.*

## 3.1 Grundlagen von Bluetooth

Die Bluetooth-Technologie wurde entwickelt, um Kabel zu ersetzen und eine kabellose Kommunikation per Funk über eine kurze Distanz zu ermöglichen. 1994 begann die Firma Ericsson mit der Arbeit an dieser Technologie. Im Jahr 1998 schlossen sich die Firmen Ericsson, IBM, Intel, Nokia und Toshiba zur Bluetooth Special Interests Group (SIG) zusammen. 1999 kamen noch 3Com, Lucent, Microsoft und Motorola zur SIG hinzu. Die führenden Hard- und Softwarehersteller stellten die Einführung als Industriestandard sicher. So wurde noch 1999 die erste Bluetooth Spezifikation 1.0 lizenziert und der darauf basierende Standard IEEE 802.15.1 veröffentlicht. Heute ist Bluetooth einer der meistverwendeten kabellosen Kommunikationsstandards, der in den letzten Jahren stark weiterentwickelt wurde. Momentan sind mehr als 1500 Unternehmen aus der Auto- und Flugzeugindustrie, Unterhaltungs-

elektronik, Computer- und Telekommunikationsbranche Mitglieder der SIG, die bluetoothfähige Produkte entwickeln, herstellen und verkaufen.[3]

Die Technologie wurde nach dem im 10. Jahrhundert lebenden dänischen Wikingerkönig Harald Blåtand (engl. Bluetooth) benannt, der Dänemark und Norwegen vereinigte. Der Name schien passend, da Bluetooth unter anderem Geräte der Telekommunikations- und Computerindustrie verbinden sollte. Weiters setzt sich das offizielle Bluetooth Logo aus den nordischen Runen zusammen, die für das moderne H ᚼ und B ᛒ stehen. Mittlerweile gibt es viele Geräte wie Handys, Laptops, PDAs, USB Dongles, Drucker, Digitalkameras und Videospielkonsolen, die Bluetooth unterstützen. Bei diesen Geräten wird Bluetooth unter anderem für die Synchronisation und den Austausch von Daten, Terminen und Visitenkarten verwendet, sowie zum Verschicken von Nachrichten und zum Spielen.

Es sind meistens kleine Geräte, die die Hauptfunktionen von Bluetooth bestmöglich nutzen: Platzsparsamkeit, minimaler Energieverbrauch und geringe Kosten. Bei Mobiltelefonen ist Bluetooth bereits State of the Art, da sie sich so leicht mit PCs verbinden lassen und der Einsatz von kabellosen Headsets möglich ist.

„Die Zahl verkaufter Geräte mit Bluetooth-Modulen hat sich zwischen 2001 und 2005 von Jahr zu Jahr mehr als verdoppelt: Wurden 2004 beispielsweise noch etwa 150 Millionen Bluetooth-Einheiten verkauft, so waren es im Jahr 2005 schon deutlich mehr als 300 Millionen“ [LÜD07][227].

## 3.2 Bluetooth Protokollstack

Bluetooth besteht aus vielen verschiedenen Protokollen, die auf mehrere Schichten, auch Layer genannt, aufgeteilt sind und gemeinsam die Funktionalität von Bluetooth gewährleisten. Jedes dieser Protokolle erfüllt bestimmte Aufgaben. Dieser Stapel von Protokollen wird Bluetooth-Protokollstack genannt. Die Schichten unterhalb des Host Controller Interface (HCI) (Abbildung 5), oft als Bluetooth-Modul bezeichnet, werden durch Hard- und Firmware realisiert. Die Schichten oberhalb des HCI werden mittels Software auf dem Bluetooth-Host realisiert. Der Bluetooth-Host und das Bluetooth Modul kommunizieren über das HCI, das als Treiber fungiert.

[3] Stand am 22. November 2007 https://www.bluetooth.org/apps/directory/default.aspx?search=All

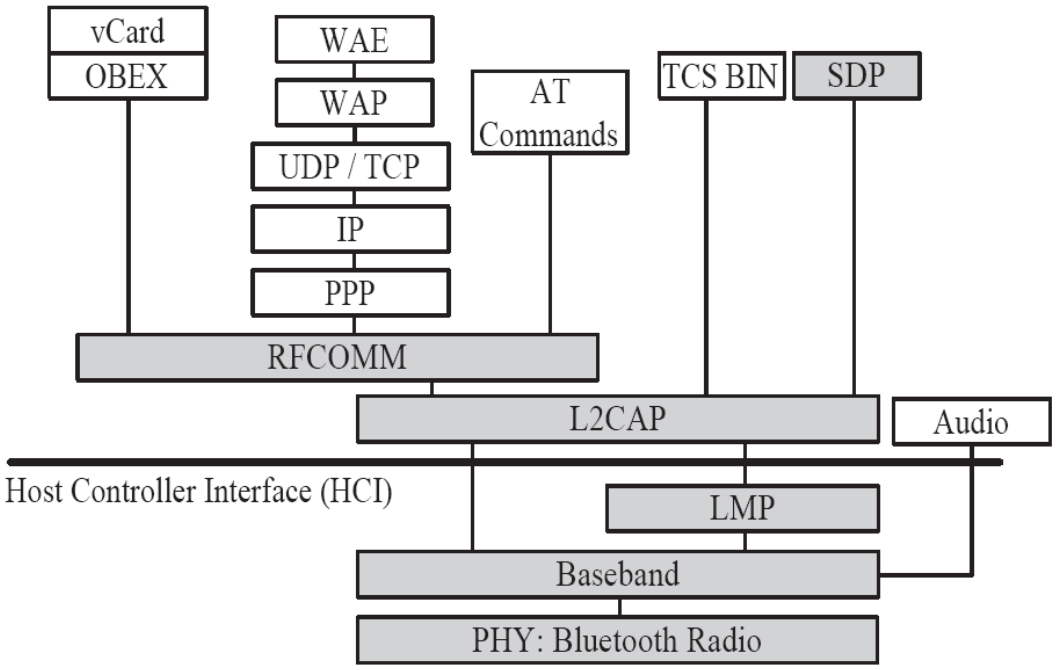


Abbildung 5: Bluetooth Protokollstack [VAR01][122]

In den Kapiteln 3.2.1 bis 3.2.4 wird zunächst die Funktionsweise der unteren/physikalischen Schichten erklärt. Kapitel 3.2.6 bis 3.2.8 beleuchten die Aufgaben der oberen/logischen Schichten des Protokollstacks. Bei der Beschreibung des Protokollstacks beschränkt sich die Arbeit auf die essentiellen Bestandteile, die in Abbildung 5 grau hinterlegt sind.

### 3.2.1 Antennas

Antennen werden in der Bluetooth-Spezifikation nicht als eigener Layer geführt. Sie sind natürlich Kernbestandteil eines jeden Bluetooth-Gerätes und ermöglichen das Senden und Empfangen der Funkwellen, die Bluetooth für die Kommunikation nutzt. Sie bilden somit die Basis der Technologie. Bei der Ausbreitung von Funkwellen können verschiedene Effekte auftreten, die für diese Arbeit von großer Bedeutung sind. Deshalb werden sie hier an dieser Stelle erklärt.

Bluetooth-Geräte verfügen zumeist über Antennen, die ein möglichst kugelförmiges Antennendiagramm aufweisen, um in jede Richtung und jeden Winkel eine Verbindung herstellen zu können. Nur über und unterhalb der Antenne gibt es keinen Empfang.

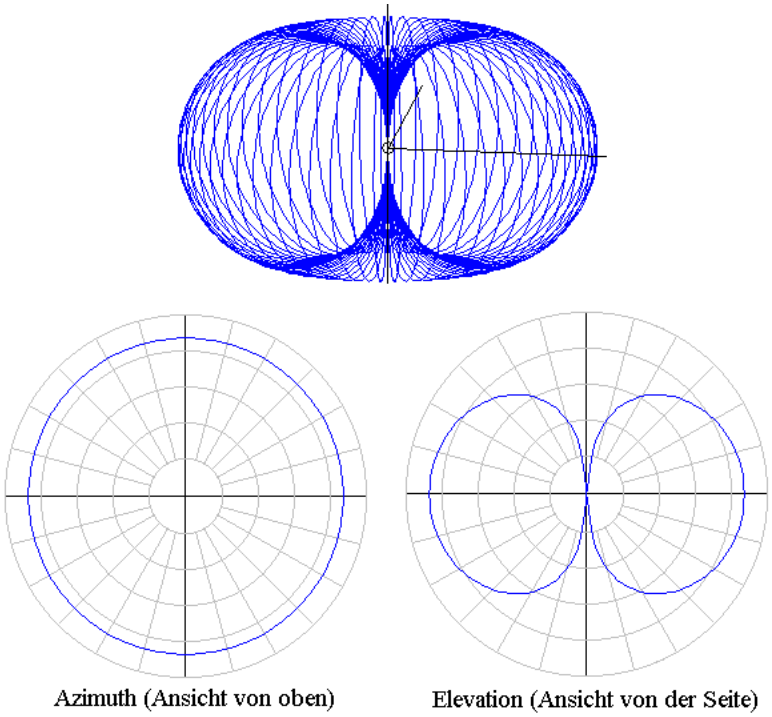


Abbildung 6: Antennendiagramm einer Dipolantenne nach [BRA02][26]

Bei der Positionierung von Antennen muss man die Umgebung beachten, da diese die Leistungsfähigkeit der Antenne erheblich beeinflussen kann [BRA02][25ff].

Die Funkwellen werden auf ihrem Weg vom Sender zum Empfänger, so wie bei jeder Funktechnologie, durch verschiedene Effekte beeinflusst, wie z.B.:

- Freiraumausbreitung: Verteilung der Leistung auf größer werdende Flächen
- Reflexion und Transmission an ebenen, glatten Flächen
- Streuung an Grenzflächen mit Unebenheiten
- Dämpfung durch Materialien (Wände, Personen und Möbel)
- Überlagerung von Wellen verschiedener Ausbreitungspfade.

Weiters können Störquellen, wie z.B. andere Funksysteme, die Funkübertragung beinträchtigen (siehe Kapitel 3.2.2) [LÜD07][38ff].

Wie stark das Signal durch Hindernisse wie Wände, Personen und Möbel gedämpft wird, hängt davon ab, aus welchem Material das Hindernis besteht, von dessen Dicke, sowie vom Einfallswinkel und der Frequenz der Funkwelle. Wenn das Hindernis von der Funkwelle umgangen werden kann, kommt es zu einer deutlich geringeren Dämpfung.

### 3.2.2 Radio

Der Radio-Layer ist der unterste Layer des Protokollstacks und ist für die physikalische Funkverbindung verantwortlich. Funksignale werden auf dieser Ebene moduliert und demoduliert.

Bluetooth-Geräte senden ihre Signale in einem weltweit lizenzfrei verfügbaren ISM-Band[4] im Frequenzbereich von 2,4GHz. In diesem Frequenzbereich arbeiten auch Mikrowellenherde, Schnurlostelefone, der ZigBee und der WLAN Standard, wodurch es leicht zu Interferenzen kommen kann. Um Daten in diesem stark belasteten Frequenzbereich trotzdem sicher übertragen zu können, setzt man bei Bluetooth auf das Frequency Hopping Verfahren und die Adaptive Power Control [BRA02][32].

#### *Frequency Hopping*

Ursprünglich erfunden wurde das Frequency Hopping Verfahren von der in Österreich geborenen Schauspielerin Hedy Lamarr[5] und dem Komponist George Antheil während des Zweiten Weltkrieges. Das Frequency Hopping Verfahren ermöglicht eine sichere und robuste Kommunikation. Diese beiden Eigenschaften sind für Bluetooth sehr wichtig [BRA02][33].

---

[4] Industrial, Scientific and Medical-Band bezeichnet den Frequenzbereich, der von Geräten in Industrie, Wissenschaft und Medizin genutzt werden kann.

[5] http://de.wikipedia.org/wiki/Hedy_Lamarr#Hedy_Lamarr_als_Erfinderin

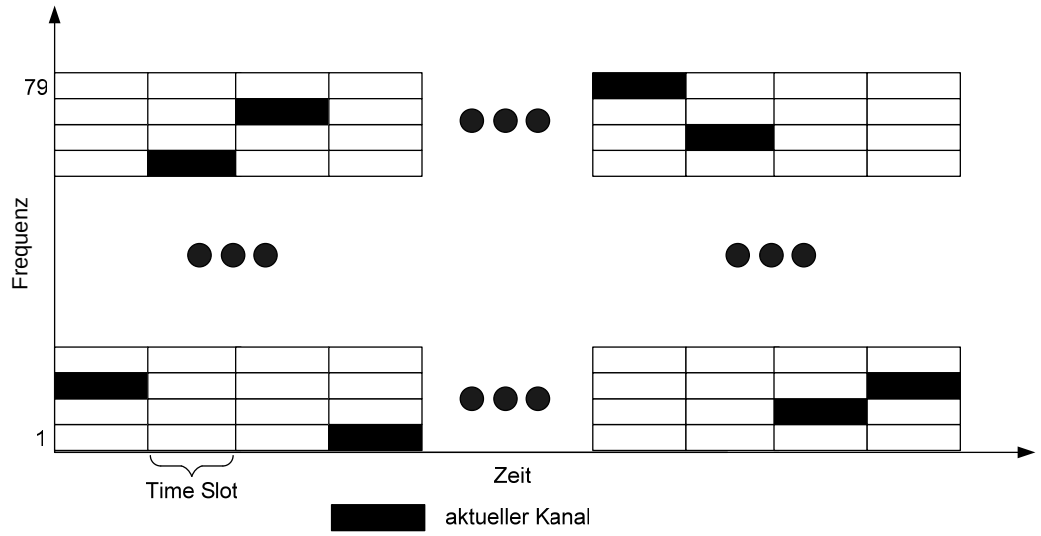


Abbildung 7: Frequency Hopping [LÜD07][234]

Beim Frequency Hopping wird das zur Verfügung stehende Frequenzband in Kanäle aufgeteilt, im Fall von Bluetooth in 79. Auf diese Art wird jedes Paket auf einer anderen Frequenz innerhalb eines Time Slots übertragen (Abbildung 7). Der Master einer Bluetooth-Verbindung errechnet die Reihenfolge, in welcher einzelne Frequenzen angesprungen (Frequency Hopping Sequenz) werden, und teilt sie seinem Gegenüber mit (siehe auch Kapitel 3.4). Dadurch wissen alle Teilnehmer, wann auf welcher Frequenz gesendet bzw. empfangen werden muss. Falls ein Kanal gestört sein sollte, und es daher beim Senden zu einer Kollision kommt, werden die Daten auf der Frequenz, die als nächstes angesprungen wird, noch einmal gesendet. Die Übertragung wird dadurch nur kurz gestört. Der Kanal wird bis zu 1600-mal pro Sekunde gewechselt. Daraus ergibt sich eine Länge von 625 Mikrosekunden für einen Time Slot. Das regelmäßige Wechseln der Sendefrequenz hat den Vorteil, dass es das Abhören der Verbindung erheblich erschwert.

### *Power Classes*

Bluetooth-Geräte werden in drei Sendeklassen eingeteilt. Sie unterscheiden sich hinsichtlich ihres Energieverbrauchs und der davon abhängigen Sendereichweite. Tabelle 2 gibt einen Überblick über die drei Sendeklassen.

| Power Class | Maximum Output Power | Minimum Output Power | Power Control | Range |
|---|---|---|---|---|
| 1 | 100mW (20dBm) | 1mW (0dBm) | Mandatory: +4dBm to 20dBm<br>Optional: -30dBm to 4dBm | 100m |
| 2 | 2.5mW (4dBm) | 0.25mW (-6dBm) | Optional: -30dBm to 4dBm | 20m |
| 3 | 1mW (0dBm) | N/A | Optional: -30dBm to 4dBm | 10m |

Tabelle 2: Power Classes [BTC21][202]

Bei mit Batterie und Akku betriebenen Geräten handelt es sich in der Regel um Bluetooth-Geräte der Sendeklasse drei, da bei Ihnen ein geringer Stromverbrauch äußerst wichtig ist, um eine möglichst lange Betriebsdauer zu gewährleisten. Bei den meisten Bluetooth-Dongles handelt es sich um Geräte der Sendeklasse zwei, die über eine Reichweite von 20m verfügen und damit in den meisten Fällen ausreichend sind.

Wie man in Tabelle 2 erkennen kann, ist der Leistungspegel nicht konstant, sondern schwankt abhängig von der Senderklasse zwischen einem Minimum- und Maximumwert. Während der Kommunikation zweier Bluetooth-Geräte beobachtet die Power Control die Received Signal Strength Identication (RSSI) und sendet, wenn notwendig, eine Anfrage über das Link Manager Protocol (LMP) an den Kommunikationspartner, ob seine Sendeleistung zu reduzieren sei, falls der RSSI Wert höher ist als für eine optimale Funkverbindung notwendig ist. Sollte der RSSI Wert zu niedrig werden, sendet der Empfänger eine Anfrage, ob die Sendeleistung zu erhöhen sei. Diesen Mechanismus nennt man Adaptive Power Control. Dadurch kommt es zu Schwankungen des Leistungspegels [BRA02][34f].

### 3.2.3 Baseband

Der Baseband Layer bildet gemeinsam mit dem Radio Layer den Physical Layer. Sie sind beide hardwaretechnisch auf dem Bluetooth-Chip untergebracht. Auf dieser Ebene werden Daten für die Funkübertragung kodiert und dekodiert. Weiters werden den zu sendenden Daten Address- und Link Control Fields hinzugefügt, bevor sie an den Radio Layer übergeben werden. Das Baseband ist auch für die Fehlererkennung und Korrektur zuständig. Im Fall einer Kollision bei der Datenübertragung veranlasst es, dass die Daten erneut gesendet werden [BRA02][41].

Bei Bluetooth wird bei der Datenübertragung zwischen zwei Verbindungsarten unterschieden:

- Asynchronous Connection Less (ACL)
- Synchronous Connection Oriented (SCO)

Welche Verbindungsart gewählt wird, hängt von der Art der Daten ab, die übertragen werden soll. Eine Verbindung besteht immer zwischen einem Master und einem Slave. Das Gerät, das die Verbindung initialisiert, wird Master und besitzt somit die Verbindung. Der Slave antwortet dem Master (siehe Kapitel 3.4).

### *ACL*

Die asynchrone Verbindung wird für die Übertragung von nicht zeitkritischen Daten genutzt. Ein Master kann gleichzeitig zu mehreren Slaves eine ACL-Verbindung haben, aber nur eine pro Slave. Der Slave kann nur dann Daten senden, wenn er vom Master dazu aufgefordert wurde.

### *SCO*

Die synchronen SCO Verbindungen zwischen Master und Slave sind durch eine garantierte Bandbreite und einen regelmäßigen Datenaustausch gekennzeichnet. Um das zu gewährleisten, werden Time Slots für diese Verbindung reserviert. Der Slave kann in den für ihn reservierten Slots Daten senden, ohne dass eine Extraaufforderung des Masters besteht. Da bei einer SCO Verbindung regelmäßig Daten gesendet werden, verwendet man sie für zeitkritische Audio- und Videodaten. Im Falle einer Datenkollision werden sie, im Gegensatz zur ACL Verbindung, nicht erneut übertragen und gehen verloren. Der zur Übertragung eingesetzte CVSD Code ermöglicht es aber, die Pakete mit verschiedenen Stufen zur Bitfehlerkorrektur zu versehen, die die Qualität bei erfolgreich übertragenen Paketen erhöht. Ein Master kann gleichzeitig drei SCO Verbindungen zwischen einem oder mehreren Slaves aufrechterhalten.

ACL Pakete können nur in diese Time Slots übertragen, die nicht für SCO Pakete reserviert sind. Neben dem Inhalt können sich Pakete auch durch ihre Länge unterscheiden. Um den Overhead, der durch den Protokoll-Header und einen Kanalwechsel entsteht, zu verringern, werden Pakete, die einen, drei und fünf Time Slots lang sind, unterstützt. In Abbildung 8 ist für jede unterstützte Paketlänge ein Beispiel ab-

gebildet. Man erkennt, dass sich ungerade Paketlängen dadurch ergeben, dass der Master immer in ungeraden Time Slots sendet: eins, drei und fünf.

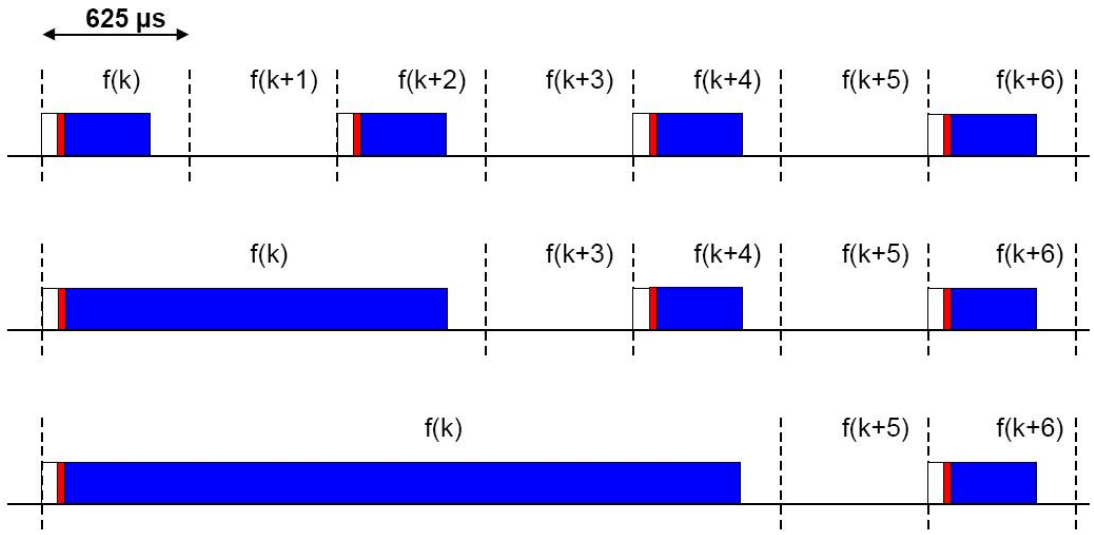


Abbildung 8: Single- und multi-slot Pakete [BTC21][250]

Ein Paket wird immer als Ganzes auf einer Frequenz übertragen. Die Empfänger wissen bei längeren Paketen, dass sie auf der Frequenz bleiben müssen, da sie die Länge des Pakets aus dem Protokoll-Header auslesen können. Nach dem Paket wird die Frequenzy Hopping Sequenz wieder fortgesetzt [BRA02][48f].

### 3.2.4 Link Manager

Der Link Manager ermöglicht, dass sich Bluetooth-Geräte gegenseitig finden können, sowie die Herstellung und Aufrechterhaltung von Verbindungen. Um diese Funktionen umzusetzen, steuert er das Baseband durch verschiedene Zustände, die anhand von Abbildung 9 erklärt werden.

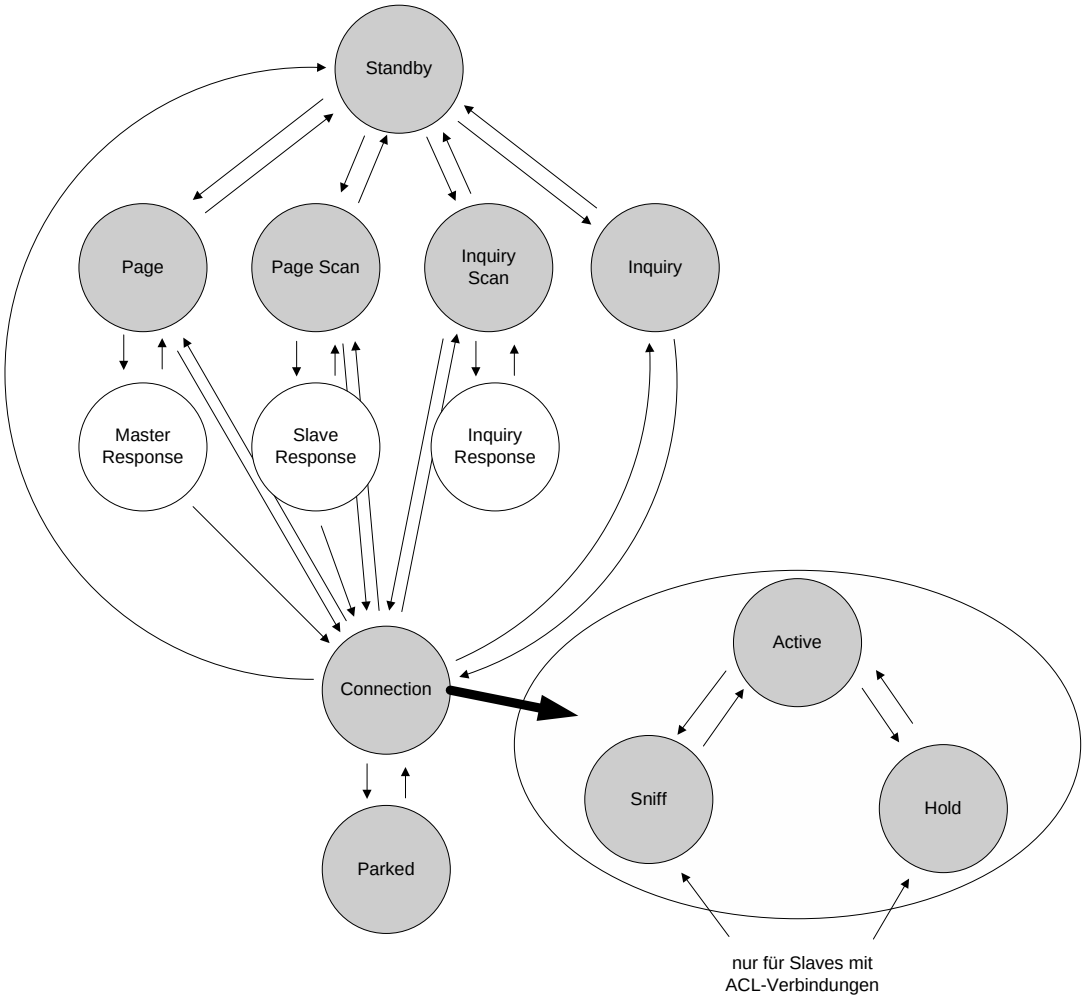


Abbildung 9: Zustände des Link Controllers [LÜD07][241]

- Standby: Das Gerät ist eingeschaltet, aber mit keinem anderen Gerät verbunden.

- Inquiry: Während der Inquiry-Phase versucht das ausführende Gerät alle anderen Bluetooth-Geräte in der Umgebung zu finden, indem es sogenannte

ID-Pakete auf speziell dafür vorgesehenen Kanälen versendet. Dann wird auf die Antworten gewartet, die in Form einer FHS-Nachricht gesendet werden.

- Inquiry Scan und Inquiry Response: Bluetooth-Geräte, die für andere Geräte sichtbar sind, treten in regelmäßigen Abständen in diesen Zustand ein, um auf empfangene ID-Pakete reagieren zu können. Sie wechseln dazu in den Zustand Inquiry Response und senden ein FHS-Paket als Antwort. Dieses Paket enthält ihre eindeutige, 48Bit lange Bluetooth Device Address (BD-ADDR) sowie den Wert ihrer internen Uhr (CLK). Ab der Bluetooth Version 1.2 kann der Inquiry Response auch den RSSI Wert beinhalten.

- Page: Um eine Verbindung aufzubauen, begibt sich das Gerät in den Zustand Page es wird in der hergestellten Verbindung die Rolle des Masters einnehmen. Der Master sendet auf den dafür vorgesehenen Kanälen Page-Nachrichten, die mittels der BD-ADDR an den Slave adressiert sind. Der Master wartet nun auf eine Slave Response-Nachricht. Wenn diese empfangen wird, wechselt er in den Zustand Master Response. Ein Gerät kann auch in den Page-Zustand eintreten, um eine Verbindung aufzubauen, ohne vorher ein Inquiry ausgeführt zu haben, wenn die BD-ADDR des Slave-Gerätes bekannt ist.

- Page Scan: Bluetooth-Geräte treten in regelmäßigen Abständen in den Zustand Page Scan ein, um auf Page-Nachrichten, die an sie adressiert sind, reagieren zu können. Nach Erhalt einer Page-Nachricht wird in den Zustand Slave Response übergegangen.

- Slave Response: Der Slave antwortet dem Master mit einer ID-Nachricht und wartet dann auf eine FHS-Nachricht als Antwort. In der FHS-Nachricht sendet der Master seine BD-ADDR sowie den Wert seiner CLK, was es dem Slave ermöglicht, sich mit dem Master zu synchronisieren. Jetzt kann der Slave in den Zustand Connection übergehen.

- Master Response: Wenn der Master die Slave Response-Nachricht akzeptiert, teilt er dem Slave seine BD-ADDR sowie den Stand seiner CLK in einer FHS-Nachricht mit und kann anschließend in den Zustand Connection übergehen.

- Connection/Connection-Active: Beim Eintritt in die Verbindung schickt der Master eine POLL-Nachricht an den Slave, der diese mit einer NULL-

Nachricht beantwortet. Somit wurde der erfolgreiche Verbindungsaufbau bestätigt.

Der Zustand Connection kann sich - abgesehen vom Normalzustand Active - in drei weiteren Subzuständen befinden:

- Connection-Hold: Der Slave geht für einen festgelegten Zeitraum in diesen Zustand über, um ein Page oder Inquiry durchzuführen oder um Energie zu sparen. Während er in diesem Zustand ist, nimmt er nicht an der ACL-Übertragung teil. Von den Connection-Subzuständen verbraucht der Hold-Zustand am zweit wenigsten Energie.

- Connection-Sniff: Im Sniff-Zustand achtet der Slave nicht permanent darauf, ob er vom Master angesprochen wird. Er vereinbart mit dem Master ein längeres Intervall. Die Zeit zwischen den beiden Zeitpunkten, zu denen der Master ihn ansprechen kann, nutzt der Slave, um Energie zu sparen oder um in ein anderes Piconetz zu wechseln, falls er Teil eines Scatternets ist. Dieser Connection-Subzustand benötigt die meiste Energie.

- Connection-Park: Wenn der Slave in den Park-Zustand übergeht, gibt er seine aktive Rolle im Piconetz auf und wird zu einem parked Slave. Zu bestimmten Zeitpunkten wacht der Slave kurz auf, um die Frequenzy Hopp Sequenz zu synchronisieren. Der Slave kann nur von sich aus entscheiden, wieder aktiv zu werden. Dem Master ist es nicht möglich, den Slave aus dem Park-Zustand zu wecken. In diesem Zustand verbraucht das Gerät die wenigste Energie.

Während des Inquiry- und Page-Vorgangs sind die Bluetooth-Geräte nicht miteinander synchronisiert, was die Kommunikation zwischen den beiden erheblich verlangsamt. Sobald jedoch ein Gerät seinem Kommunikationspartner den Stand seiner CLK mitteilt, kann sie dieser dazu verwenden, um den Kommunikationsprozess zu beschleunigen. [BRA02][67ff] und [LÜD07][242f]. Dies gilt für das gesamte Unterkapitel.

### 3.2.5 Host Controller Interface (HCI)

In der Bluetooth-Spezifikation ist das Host Controller Interface (HCI) definiert, das die Kommunikation zwischen Bluetooth-Host und Bluetooth-Modul regelt und somit das Bindeglied zwischen Hardware und Software ist. Wie man in Abbildung 10 sieht, wird es auf der Softwareseite durch Gerätetreiber und auf der Hardwareseite durch Firmware realisiert. Durch die klare Trennung des Protokollstacks in Hard- und Softwareteil und die Definition des Standardinterfaces kann der Benutzer Bluetooth-Hardware von verschiedenen Herstellern betreiben und dabei sicher sein, dass sie mit seinem Softwarestack funktionieren [BRA02][119ff].

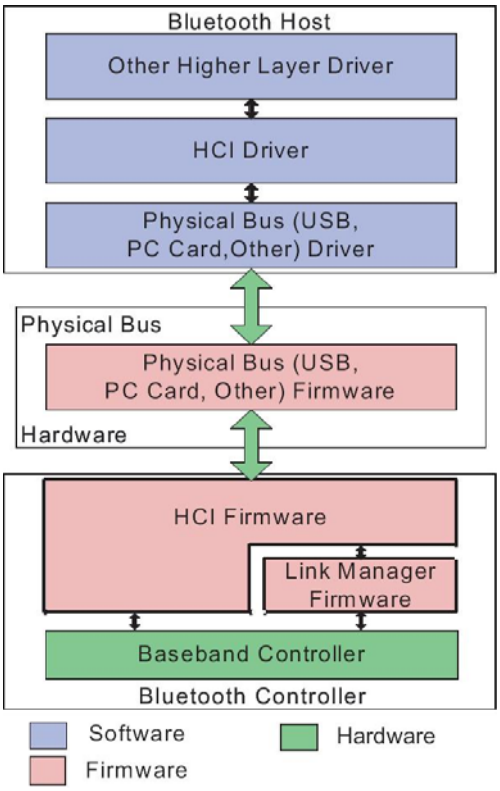


Abbildung 10: Übersicht über die unteren Software Layers [BTC21][527]

Das HCI definiert drei Pakettypen:

- Commands: Befehle die vom Host an das Modul gesendet werden.
- Events: Ergebnisse von Commands und allgemeine Statusinformationen werden so vom Modul an den Host übertragen.
- Data Packets: dienen zum Senden und Empfangen von Daten. Diese können in beide Richtungen fließen.

Programme können über das HCI auf die verschiedenen Funktionen des Bluetooth-Moduls zugreifen. Zu diesen Funktionen gehören unter anderem die gesamte Verbindungskontrolle, Verbindungseinstellungen bzgl. der Energiesparmodi und Role Switch (Rollenwechsel zwischen Master und Slave), Zugriff auf Informationen des lokalen Bluetooth-Moduls sowie verbundener Geräte und die Einstellung verschiedener Baseband Parameters wie z.B. Timeouts [BRA02][148]. Welche Methoden einem Anwendungsprogrammierer zur Verfügung stehen, hängt davon ab, ob sie in der Bluetooth-Stack API realisiert wurden (siehe Kapitel 3.7).

### 3.2.6 Logical Link Control and Adaptation Protocol (L2CAP)

L2CAP regelt den Datentransport für die oberen Schichten. Der gesamte Datenverkehr, abgesehen von Audio- und Videodaten, fließt über eine ACL-Verbindung. Damit mehrere Dienste die ACL-Verbindung gleichzeitig nutzen können, verwendet L2CAP Multiplexing. Um die Datenpakete der einzelnen Dienste nach außen hin unterscheiden zu können, erhält jeder Dienst einen Protocol/Service Multiplexer (PSM). Es kann bis zu 32768 PSMs geben, wobei die PSMs von 1 bis 4096 für bereits existierende und zukünftige Dienste reserviert sind. Innerhalb der ACL-Verbindung werden die unterschiedlichen Dienste mit dem Channel Identifier (CID) unterschieden. Das L2CAP übernimmt auch die Segmentierung und Reassemblierung von Datenpaketen. Das ermöglicht den übergeordneten Layern auch größere Datenmengen, bis zu 65500 Byte, komfortabel zu übertragen.

Für jede Anwendung lässt sich eine individuelle Dienstqualität vereinbaren, die das L2CAP zu garantieren versucht. Die Dienstqualität ist dabei durch Parameter wie die mittlere und maximale Datenrate und durch Grenzwerte für tolerable Verzögerungszeiten festgelegt [LÜD07][247].

Alle Anwendungen müssen L2CAP für den Datentransport verwenden. Die höheren Schichten des Stacks RFCOMM und SDP setzen ebenfalls auf L2CAP auf, was es zum Basisprotokoll der oberen Schichten macht [BRA02][175].

### 3.2.7 RFCOMM (Radio Frequency Communication)

RFCOMM emuliert über L2CAP eine serielle RS 232 Schnittstelle und wird daher auch Kabelersatzprotokoll genannt [VAR01][127]. Dieser Ansatz ermöglicht es, alte Programme, die die serielle Schnittstelle nutzten, einfach an Bluetooth anzupassen. Es wird auch von mehreren Bluetooth-Profilen zur Datenübertragung genutzt [BRA02][195]. RFCOMM ermöglicht es, bis zu 60 solcher serieller Verbindungen zur Verfügung zu stellen [LÜD07][248].

### 3.2.8 Service Discovery Protocol (SDP)

Das Service Discovery Protocol (SDP) ermöglicht es, Dienste auf anderen Bluetooth-Geräten zu finden. Dazu gibt es einen SDP-Server und einen SDP-Client. Die von einem Bluetooth-Gerät angebotenen Dienste werden in einer lokalen Datenbank gespeichert, die vom SDP-Server verwaltet wird. Jeder Diensteintrag in der Datenbank enthält die Informationen, die der Client benötigt, um den Dienst zu nutzen. Der SDP-Client, der einen Dienst nutzen will, verbindet sich zunächst mittels L2CAP zu dem anderen Bluetooth-Gerät. Dem SDP-Server-Dienst ist fix der PSM=0x0001 zugewiesen. Somit kann der Client diesen Dienst sofort nutzen. Der Client kann nun entweder durch die Datenbank des Servers browsen oder gezielte Suchanfragen absetzen, um die gewünschten Services zu finden. Wenn er den gewünschten Dienst gefunden hat, liest er die Informationen, die für dessen Nutzung notwendig sind, aus der Datenbank aus und muss dann eine separate Verbindung zum Dienst aufbauen. Die SDP-Verbindung zwischen Client und Server kann nur für Suchanfragen verwendet werden [BRA02][196ff].

## 3.3 Anwendungsprofile

Um eine einfache Nutzung des Bluetooth-Protokollstacks zu ermöglichen, wurden von der Bluetooth SIG Anwendungsprofile definiert. Sie spezifizieren, wie die einzelnen Schichten des Stacks genutzt werden sollen, um eine Anwendung für den End-User bereitstellen zu können. Sie werden daher auch als vertikaler Schnitt durch den Protokollstack bezeichnet. Da die Hersteller von Bluetooth-Geräten die spezifizierten Anwendungsprofile einhalten, verhalten sich alle Bluetooth-Geräte, die ein bestimmtes Profil anbieten, gleich. Auf diese Weise wird die Interoperabilität zwischen den verschiedenen Geräten sichergestellt. Würden die Hersteller sich nicht daran halten, würde das gesamte Konzept der Adhoc-Netze nicht mehr funktionieren. Deshalb fördert die Bluetooth SIG die Entwicklung neuer Profile, um auch das Funktionieren neuer Entwicklungen zu gewährleisten [BRA02][363f].

Mittlerweile gibt es sehr viele Anwendungsprofile. Wie man in Abbildung 11 sehen kann, sind sie hierarchisch angeordnet, da ein Profil einer Ebene seine Bestandteile an die Profile der folgenden Ebenen vererbt bzw. weitergibt. Das Generic Access Profile ist das Basis-Profil, da es grundlegende Dinge, wie die Darstellung der Bluetooth Device Address des Bluetooth-Gerätenamens und des PINs (auch Bluetooth Passkey genannt) festlegt. Weiters definiert es die Nutzungsregeln für Operationen zum Finden von anderen Geräten und den Verbindungsaufbau. Abbildung 11 zeigt die derzeit standardisierten Profile, die dem End-User zur Verfügung stehen [LÜD07][250ff].

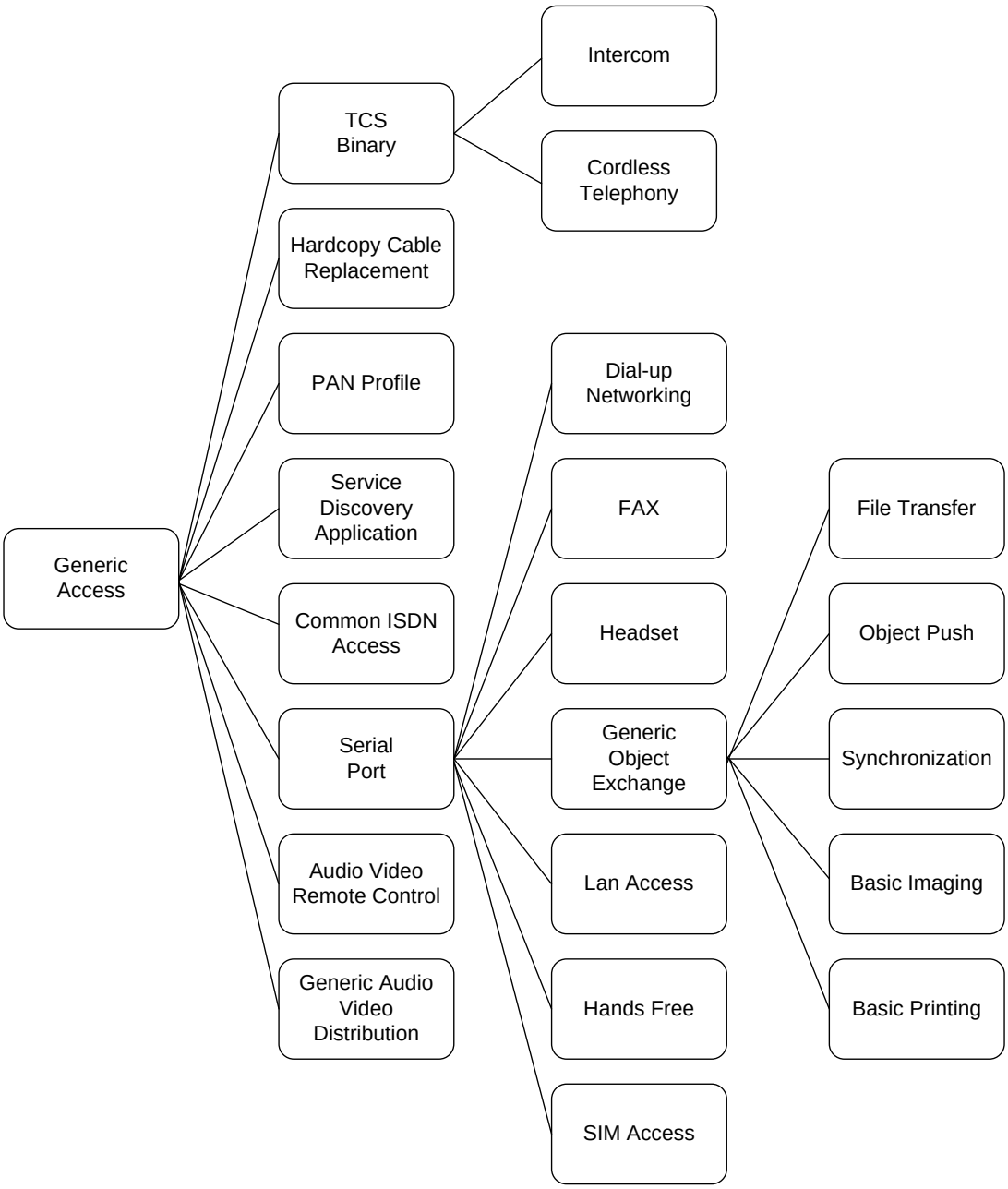


Abbildung 11: Hierarchischer Aufbau der Bluetooth-Profile [LÜD07][252]

## 3.4 Bluetooth Netzwerktopologie

Bluetooth-Netze gehören zu den Adhoc-Netzen, eine fixe Unterscheidung in Clients und Server ist hier nicht möglich. Dieser Ansatz wurde gewählt, da Bluetooth-Geräte meist handliche und mobile Geräte sind, daher oft bewegt werden und meist nur wenige Geräte ein Netz bilden. Eine Einheit eines Bluetooth-Netzwerks kann entweder Master oder Slave sein. Das Gerät, das den Aufbau der Verbindung initiiert, wird standardmäßig auch Master.

Jedes Bluetooth-Gerät besitzt eine eindeutige 48Bit lange Bluetooth Device Address (BD-ADDR). Beim Verbindungsaufbau zu einem Slave übermittelt der Master seine BD-ADDR gemeinsam mit seiner internen Uhr (CLK). Im Baseband-Teil der Bluetooth-Spezifikation ist ein Algorithmus für die Berechnung der pseudo-zufälligen Sequenz für das Frequency Hopping aus der BD-ADDR und der CLK des Masters beschrieben. Der Slave berechnet auf diese Weise die Frequency Hopping Sequenz und ist so mit dem Master synchronisiert. Der Master bestimmt weiters, wann jedes Gerät Daten senden darf. Der Master ermöglicht den Slaves Daten zu senden, indem er Time Slots für Sprach- und Datenpakete für sie reserviert. In Time Slots für Datenpakete darf nur auf eine Anfrage vom Master geantwortet werden. In Time Slots für Sprachpakete müssen regelmäßig Daten gesendet werden, auch wenn keine Anfrage vom Master vorliegt. Der Master bestimmt, wie die gesamte Bandbreite zwischen den Slaves aufgeteilt wird, indem er entscheidet, wie oft er mit jedem Slave kommuniziert. Dieses Verfahren, Time Slots auf mehrere Geräte aufzuteilen, wird Time Division Multiplexing (TDM) genannt [BRA02][8].

Ein Bluetooth-Netzwerk, in dem sich ein Master und ein oder mehrere Slaves befinden, wird als Piconetz (Abbildung 12b) bezeichnet. Slaves werden innerhalb des Piconetzes über eine drei Bit lange ID adressiert. Daraus ergibt sich, dass es in einem Piconetz maximal sieben aktive Slaves geben kann. Die achte Adresse ist für den Master Broadcast reserviert. Es können sich in einem Piconetz auch weitere inaktive Slaves befinden. Diese Geräte befinden sich dann im so genannten Zustand parked. Es kann bis zu 255 parked Slaves in einem Piconetz geben [VAR01][43]. Alle Geräte in einem Piconetz folgen der Frequency Hopping Sequenz des Masters. Die Slaves sind nur mit dem Master verbunden, zwischen den Slaves gibt es keine Verbindung.

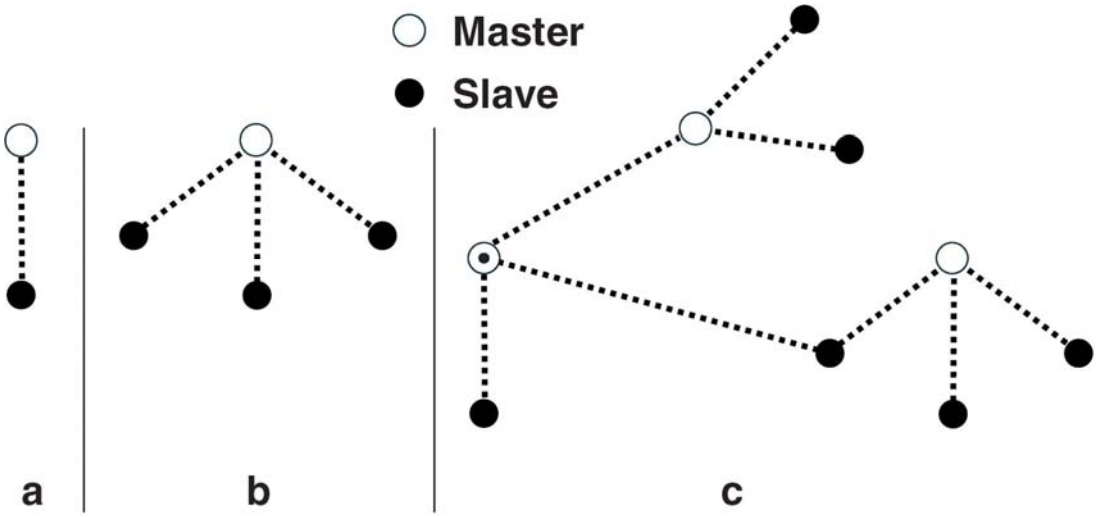


Abbildung 12: Mögliche Topologien vernetzter Bluetooth-Geräte [BTC20][229]

Besteht ein Bluetooth-Netzwerk aus mehr als acht Geräten, spricht man von einem Scatternet. Ein Scatternet ist dann gegeben, wenn sich zwei oder mehr Piconetze überschneiden. Pro Piconetz gibt es einen Master. Ein Gerät kann in einem Piconetz Master und in einem anderen Slave sein. Es ist nicht möglich, dass ein Gerät Master von zwei Piconetzen ist, da der Master die Frequency Hopping Sequenz für ein Piconetz vorgibt.

Das Time Division Multiplexing ermöglicht es Geräten, Teil verschiedener Piconetze zu sein, indem es die Time Slots zwischen den Piconetzen aufteilt. Eine Einheit ist auf diese Weise in der Lage, sequentiell in unterschiedlichen Piconetzen, zu einem gegebenen Zeitpunkt jedoch immer nur in einem Netz aktiv zu sein [VAR01][43] (Abbildung 12c).

## 3.5 Bluetooth Versionen 1.0 - 2.0

Seit der Veröffentlichung des Bluetooth-Standards 1.0 im Jahr 1999, die eine Bruttodatenrate von 1Mbit/s definierte, wurde die Technologie stark weiterentwickelt. „Neue Leistungsmerkmale wie

- ein schnellerer Verbindungsaufbau,
- ein adaptives, an die aktuelle Störsituation angepasstes Frequency Hopping,
- erweiterte flexiblere Datenübertragungsmöglichkeiten,

- eine verbesserte Fehlererkennung und Flusskontrolle

sind Bestandteile der Version 1.1. aus dem Jahr 2001 und der Version 1.2 aus dem Jahr 2003. Die entscheidende Neuerung in der Bluetooth-Version 2.0 aus dem Jahre 2005 ist die Erweiterung der Bruttodatenrate auf bis zu 3Mbit/s durch die Verwendung höherwertiger Modulationsverfahren. Daher trägt diese Version vielfach auch den Zusatz „Bluetooth EDR (Enhanced Data Rate) [LÜD07][227]".

## 3.6 Lokalisierung in Funknetzen

*Im ersten Teil dieses Kapitels wurde die Funktionsweise von Bluetooth erklärt. Im zweiten Teil wird nun erörtert, wie sich die Lokalisierung eines MD mit Bluetooth realisieren lässt.*

*Es existieren mehrere Möglichkeiten, um die Position einer Komponente in einem Funknetz zu bestimmen. Die Positionsbestimmung eines Mobile Devices (MD) erfolgt mit Hilfe mehrerer Base Stations (BS), deren Position bekannt ist. Man kann hier zwischen Network Based Technologies und Handset Based Technologies unterscheiden, je nachdem ob die Position des MD durch das Netzwerk oder durch das MD selbst bestimmt wird. In diesem Teil der Arbeit wird ein kurzer Überblick über die wichtigsten Lokalisierungstechniken gegeben und anschließend diskutiert, welche davon für Bluetooth geeignet sind. Da für den praktischen Teil der Arbeit ein netzwerkbasierter Ansatz gewählt wurde, werden hier auch nur Techniken vorgestellt, die dafür verwendet werden können.*

### 3.6.1 Relevante Lokalisierungstechniken

#### *Radio Signal Strength*

Bei dieser Technik misst man die Signalstärke, mit der das Signal empfangen wird. Es muss nun die Distanz zwischen Sender und Empfänger bestimmt werden. Ist neben der gemessenen Empfangsleistung auch die Sendeleistung sowie die Dämpfung des Kanals bekannt, so kann die Distanz berechnet werden. Eine weitere Möglichkeit, die Distanz zu bestimmen, beruht auf dem Vergleich der aktuellen Signalstärke mit Referenzwerten, die in unterschiedlichen Entfernungen gemessen wurden [LUD04][8].

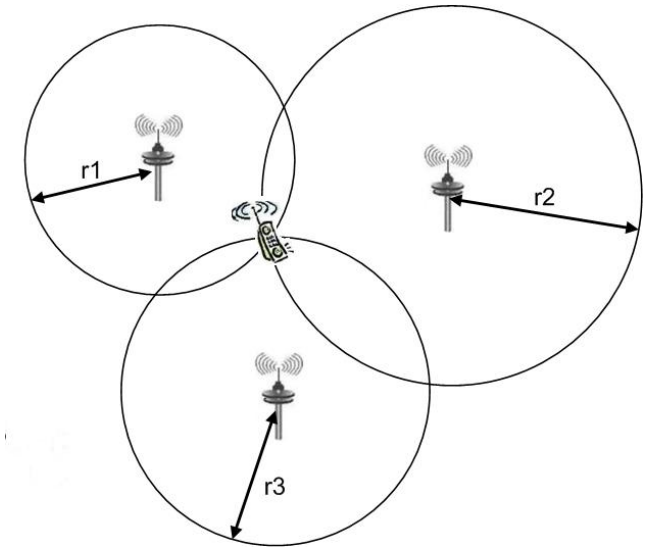


Abbildung 13: Trilateration [ALE05][2]

Wurden die Distanzen zwischen dem MD und mindestens drei BSs ermittelt, kann man mit dem Trilaterationsverfahren die Position des MD berechnen. Dazu werden die ermittelten Distanzen als Radien verwendet. Wie man in Abbildung 13 sieht, ist die Position des MD der Schnittpunkt der drei Kreise.

### *Time of Arrival (TOA)*

Beim TOA Verfahren misst man die Zeit, die ein Funksignal benötigt, um vom Sender zum Empfänger zu gelangen. Anhand der Ausbreitungsgeschwindigkeit des Funksignals kann dann die Entfernung zwischen Sender und Empfänger berechnet werden. Für dieses Verfahren müssen die Uhren von Sender und Empfänger synchronisiert werden.

Es ist auch möglich, die Round Trip Time zu messen, das heißt die Zeit, die das Signal vom Sender zum Empfänger und wieder zurück benötigt. In diesem Fall müssen die Uhren von Sender und Empfänger nicht synchronisiert werden, da die Zeit immer nur beim Sender gemessen wird. Es muss jedoch sichergestellt sein, dass der Empfänger das empfangene Signal unverzüglich beantwortet [LUD04][10].

Hat man die Distanzen zwischen dem MD und mindestens drei BSs ermittelt, kann man die Position mittels Trilaterationsverfahren berechnen.

### *Angle of Arrival (AOA)*

Bei dieser Technik wird der Winkel, in dem das Funksignal des MD bei der BS empfangen wird, über dessen Antenne bestimmt. Wenn der Winkel von zwei BSs zum MD, sowie die Distanz zwischen den beiden BSs bekannt ist, kann die Position des MD über Triangulation berechnet werden [LUD04][9].

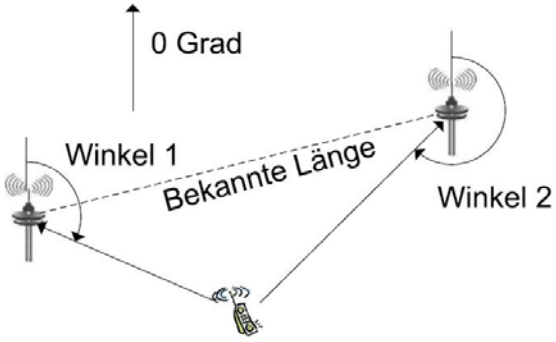


Abbildung 14: Triangulationsalgorithmus [LUD04][9]

### *Cell Identification (Cell-ID)*

Die Position eines mobilen Gerätes wird durch die Funkzelle bestimmt, in der es sich befindet. Das Funknetz besteht aus mehreren Zellen. Eine Zelle wird durch eine BS gebildet, deren Position bekannt ist. Dem mobilen Gerät wird einfach die Position der Basisstation zugeordnet, mit der es gerade verbunden ist. Die Genauigkeit der Positionsbestimmung hängt von der Größe der Funkzellen ab [LUD04][6f].

## 3.6.2 Lokalisierung mit Bluetooth

Da es mit Bluetooth nicht möglich ist, den Winkel, in dem ein Funksignal empfangen wird, zu messen, kann die AOA-Methode nicht zur Positionsbestimmung verwendet werden. Bei Bluetooth kommt es bei der Datenübertragung zu Zeitunterschieden von bis zu 1 µs. Das führt zu einem Fehler von 300m in der Distanzbestimmung mit der TOA-Methode. Das macht sie für Bluetooth ebenfalls unbrauchbar. Das Cell-ID-Verfahren kann verwendet werden. Die Genauigkeit der Positionsbestimmung ist abhängig davon, welcher Geräteklasse die verwendeten Bluetooth-Module entsprechen. Der Radius einer Zelle kann somit 10, 20 oder 100m betragen (siehe Kapitel 3.2.2 Power Classes). Das Radio Signal Strength-Verfahren kann ebenfalls mit Bluetooth realisiert werden, da das HCI-Funktionen bietet, um auf Received Signal Strength Indicator (RSSI), Transmitted Power Level (TPL) und Link

Quality (LQ) Parameter zuzugreifen. Aus diesen Parametern lässt sich ein Signalstärkeparameter errechnen, der einen Schluss auf die Entfernung zwischen MD und BS zulässt.

Das Radio Signal Strength-Verfahren ermöglicht eine genauere Positionsbestimmung als der Cell-ID-Ansatz, mit dem nur festgestellt werden kann, ob sich ein MD in Sendereichweite zu einer BS befindet oder nicht. Aus diesem Grund wird auch die Radio Signal Strength-Methode in dieser Arbeit weiterverfolgt, und die dafür notwendigen Parameter zur Berechnung der Signalstärke werden hier näher erklärt.

### *Received Signal Strength Indicator (RSSI)*

Der RSSI gibt die Stärke eines empfangenen Signals an. Er kann für jede ACL-Verbindung über das HCI mit der Methode *HCI_Read_RSSI* ausgelesen werden. Miteinander kommunizierende Bluetooth-Geräte verwenden den RSSI-Wert, um ihre Sendeleistung aneinander anzupassen (siehe Kapitel 3.2.2 Power Classes).

Der gelesene RSSI-Wert bezieht sich auf die Golden Received Power Range (GRPR). Er ist größer Null, wenn die Stärke des empfangenen Signals über der GRPR liegt, gleich Null, wenn sie innerhalb liegt und kleiner Null, wenn sie sich darunter befindet. Da Bluetooth-Geräte ihre Sendeleistung einander anpassen, kommt es in der Regel zu keinen RSSI-Werten über Null. Ein Wert unter Null weist darauf hin, dass das Gerät, mit dem kommuniziert wird, bereits mit maximaler Leistung sendet, das Signal jedoch noch immer zu schwach ist.

Die GRPR ist durch einen oberen und unteren Schwellenwert begrenzt. Der untere Schwellenwert muss mindestens 6dB über dem vom Bluetooth-Modul messbaren Wert liegen. Maximal darf er bei -56dBm liegen. Der obere Grenzwert für die GRPR liegt 20dB über dem unteren Grenzwert mit einem maximalen Fehler von 6dB (siehe Abbildung 15) [BTC21][594].

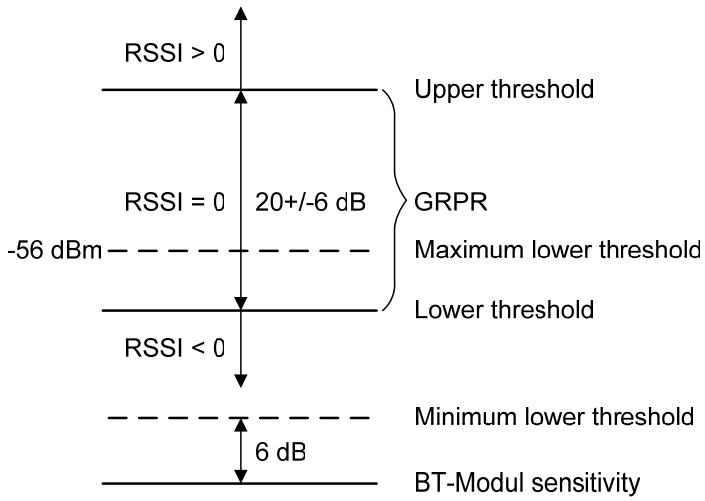


Abbildung 15: Golden Received Power Range [ANT03][3] erweitert

Da der untere Schwellwert für die GRPR vom Hersteller des Bluetooth-Moduls selbst gewählt werden kann und der obere Schwellwert von dem unteren abhängt, müssen beide Werte für jedes Modell eines Bluetooth-Moduls extra bestimmt werden [ANT03][3].

### *Transmit Power Level (TPL)*

Mit dem HCI-Befehl *HCI_Read_Transmit_Power_Level* kann man die aktuelle Sendeleistung für eine ACL-Verbindung feststellen. Man hat die Möglichkeit über einen Parameter festzulegen, ob die maximale oder die aktuelle Sendeleistung gelesen werden soll. Die maximalen Werte liegen laut Spezifikation zwischen -30 und 20dBm [BTC21][542f]. Für die unterschiedlichen Geräteklassen gelten jedoch eigene Grenzwerte (siehe Tabelle 2).

### *Link Quality (LQ)*

Die LQ kann für jede ACL-Verbindung mit dem Befehl *HCI_Read_Link_Quality* über das HCI ausgelesen werden. Der gelesene Wert liegt zwischen 0 und 255. Je höher der Wert desto besser ist die LQ [BTC21][593]. Wie die LQ genau gemessen wird, kann jeder Hersteller von Bluetooth-Modulen selbst entscheiden. Normalerweise bezieht sich der LQ-Wert auf die Bitfehlerrate, die mit zunehmender Entfernung zwischen zwei Bluetooth-Geräten steigt.

## 3.7 Bluetooth-Softwarestacks

Auf den RSSI, TPL und LQ Parameter muss direkt über das HCI zugegriffen werden. Das gestaltet sich in der Praxis recht schwierig. Das HCI fungiert als Schnittstelle zwischen der Bluetooth-Hardware und dem Bluetooth-Softwarestack. Dieser läuft auf dem Bluetooth-Host und implementiert die Layers des Protokollstacks oberhalb des HCI. Dem Anwendungsprogrammierer ist in der Regel nur der Zugriff auf diese Layers gestattet, und er hat somit keinen Zugriff auf die für uns wichtigen Parameter. Aufgrund dieser Tatsache wird in diesem Abschnitt ein Blick auf die wichtigsten Bluetooth-Softwarestacks geworfen.

Am Markt sind zahlreiche kommerzielle Bluetooth-Softwarestacks[6] verfügbar, wie z.B. die von der Firma Broadcom[7], die einen breiten Funktionsumfang bieten, den direkten Zugriff auf das HCI ermöglichen und gut dokumentiert sind. Will man den Broadcom Stack für seine Programmentwicklung nutzen, muss man das Software Development Kit (SDK) der Firma kaufen, dessen Preis sich auf mehrere tausend Euro beläuft. Da kommerzielle Stacks aufgrund der hohen Anschaffungskosten für die im Rahmen dieser Arbeit erstellte Software keine Alternative waren, beschränken wir uns im weiteren Verlauf ausschließlich auf frei zugängliche Bluetooth-Softwarestacks.

Wesentliche Entscheidungskriterien für die Wahl des in dieser Arbeit verwendeten Bluetooth-Softwarestacks bestehen darin, dass der Zugriff auf Parameter, die über die Signalstärke Aufschluss geben, möglich ist und die Unterstützung aktueller Bluetooth-Standards (mind. Bluetooth 2.0) sowie eine ausführliche Dokumentation vorhanden sind.

### 3.7.1 Windows

Windows XP verfügt mit dem Service Pack 2 über einen Bluetooth-Softwarestack, der ein offenes API anbietet und gut dokumentiert ist. Im Microsoft Developer Network finden sich moderierte Foren zum Thema „Bluetooth-Programmierung unter Windows“ [MDN07]. Die zur Verfügung gestellte API ermöglicht aber keinen direkten Zugriff auf das HCI. Somit ist keine Distanzbestimmung möglich und der Bluetooth-Softwarestack von Windows für die zu erstellende Software nicht brauchbar.

---

[6] Überblick über alle Bluetooth-Softwarestacks bei heise.de, Stand 11.12.2007

[7] Früher als Widcomm Stack bekannt, bevor die Firma Widcomm im April 2004 von Broadcom gekauft wurde www.broadcom.com, Stand 11.12.2007

### 3.7.2 FreeBT

Der FreeBT Stack ist neben dem Windows Stack der einzige freie Bluetooth-Softwarestack für das Windows Betriebssystem. Dieser Stack erlaubt dem Programmierer den Zugriff auf das HCI. Bis Ende 2004 schritt die Weiterentwicklung zügig voran, wurde dann aber anscheinend eingestellt. Das grundlegende L2CAP-Protokoll wird vom Stack noch nicht unterstützt. Ein weiterer Punkt, der gegen eine Verwendung dieses Stacks spricht, ist die nicht vorhandene Dokumentation. Es gibt auch keine lebendige Community, die diesen Stack verwendet oder bei Problemen behilflich sein könnte [FRE07].

### 3.7.3 Linux

Der offizielle Bluetooth-Softwarestack von Linux ist BlueZ. Er wurde seit 2001 von Max Krasnyansky entwickelt und unter GNU General Public License veröffentlicht. Seit der Linux Kernel Version 2.4.6 ist er Bestandteil des Betriebssystems. Im Jänner 2004 übernahm Marcel Holtman die Leitung des Projekts. Der Stack wird laufend weiterentwickelt und unterstützt so auch Bluetooth 2.0 + EDR. Das Projekt ist nur spärlich dokumentiert, erfreut sich aber einer großen Benutzer- und Entwickler-Community, die in Mailinglisten nahezu alle Fragen fachkundig beantwortet. BlueZ besteht aus mehreren Modulen, in denen die Protokolle der verschiedenen Schichten implementiert sind. BlueZ ermöglicht dem Programmierer Zugriff zu den Protokollen aller Schichten über BSD-Sockets (Berkely System Distribution). Der Zugriff auf das HCI ist so ebenfalls möglich, jedoch muss das zugreifende Programm Root-Rechte besitzen [BLZ07].

Der Vollständigkeit halber sei an dieser Stelle noch erwähnt, dass es unter Linux noch eine Reihe anderer freier Bluetooth-Stacks wie AXIS OpenBT Stack, IBM BlueDrekar, Qualcomm BlueZ und Nokia Affix Bluetooth Stack gibt. Da BlueZ der offizielle Bluetooth-Stack von Linux ist und alle gestellten Anforderungen erfüllt, wird er auch für die Realisierung dieser Arbeit verwendet.

## 3.8 JTooth Library

Die letzte Komponente, die wir auf der Softwareseite für die Umsetzung der Messsoftware benötigen, ist eine Klassenbibliothek, die den Zugriff auf den BlueZ-Softwarestack ermöglicht. Die Wahl fiel auf die JTooth Library. Sie wurde 2004 von Stefan Mischke entwickelt. Diese Java Klassenbibliothek ermöglicht die Anbindung von Java an BlueZ mittels Java Native Interface (JNI). Die Klassenbibliothek unter-

stützt von den höheren Schichten nur L2CAP und SDP, ermöglicht aber den Zugriff auf das HCI und damit auf die Parameter, die eine Bestimmung der Signalstärke möglich machen. Beim Design der Bibliothek wurde vor allem auf spätere Erweiterbarkeit großes Augenmerk gelegt, um die Implementierung weiterer Protokolle möglichst einfach zu gestalten. Die Bibliothek gestattet auch den Zugriff auf kritische Bluetooth-Funktionen, die sich auf andere Programme, die Bluetooth nutzen, auswirken. Wenn man diese Funktionen der Klassenbibliothek nutzt, muss das ausführende Programm über Root-Rechte verfügen.

### 3.8.1 JNI-Library

Da BlueZ nur Schnittstellen für die Programmiersprache C anbietet, wurde eine JNI-Library erstellt, um diese Schnittstellen über Java ansprechen zu können. So enthält die Klasse *JTooth* auf der Java-Seite nur die Funktionsprototypen, implementiert sind die Funktionen in der Programmiersprache C im Modul *jtooth.c*. Dieses Modul wird gemeinsam mit weiteren Hilfsmodulen zur JNI-Library *libjtooth.so* kompiliert. Diese kann dann in Java geladen und verwendet werden [MIS04][38f].

### 3.8.2 JTooth-Klassenbibliothek

Um die Übersichtlichkeit und Handhabung der JNI-Library zu verbessern, wurden die statischen Methoden der Klasse JTooth durch eine Klassenbibliothek gekapselt. Beim Design orientierte sich Stefan Mischke an den Paketen java.net und java.io, um eine intuitive Benutzbarkeit umzusetzen [MIS04][39f]. Das UML-Diagramm in Abbildung 16 zeigt die Zusammenhänge zwischen den einzelnen Klassen in der JTooth-Klassenbibliothek. An dieser Stelle werden nun die einzelnen Klassen mit ihren Methoden kurz vorgestellt, die bei der Erstellung der Software verwendet wurden.

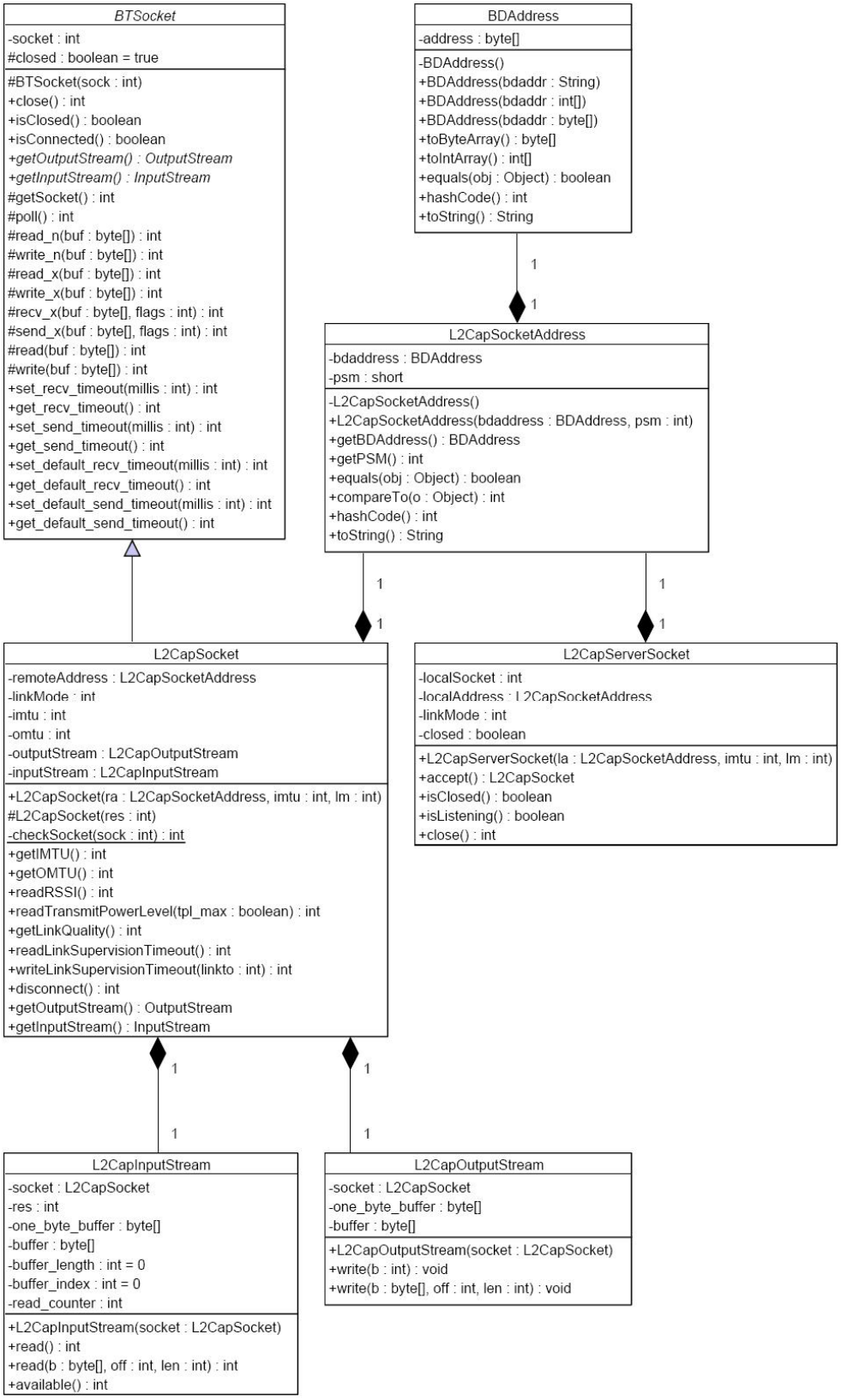


Abbildung 16: JTooth-Klassenbibliothek [MIS04][41]

- BDAddress: Diese Klasse erleichtert die Handhabung von Bluetooth-Adressen. Die BDAddress kann nur über den Konstruktor gesetzt werden, eine spätere Änderung ist bewusst nicht möglich. Die Klasse bietet auch Methoden an, welche die Adresse in unterschiedlichen Formaten zurückliefert.

- L2CapSocketAddress: Für eine eindeutige Dienstidentifizierung muss dem Konstruktor der Klasse neben einer BDAddress noch PSM übergeben werden. Die L2CapSocketAddress kann ebenso wie die BDAddress nur über den Konstruktor initialisiert werden.

- BTSocket: Die abstrakte Klasse BTSocket bietet die Funktionalität zur Benutzung von BSD-Sockets, über die bei BlueZ die einzelnen Protokolle angesprochen werden. BTSocket selbst ist eine abstrakte Klasse. Es ist vorgesehen, dass für jedes Protokoll eine Kindklasse erstellt wird. Die Klasse BTSocket kapselt jene Methoden, die alle Protokolle benötigen.

- L2CapSocket: Diese Klasse ist von BTSocket abgeleitet und ermöglicht eine L2CAP-Verbindung zu anderen Bluetooth-Geräten. Für den Verbindungsaufbau übergibt man dem Konstruktor eine L2CapSocketAddress, die Maximum Transfer Unit (MTU) und den *link_mode*. Die MTU gibt dabei die maximale Größe eines eingehenden Datenpakets an. Über den *link_mode* kann man Verbindungseigenschaften, wie Authentifizierung und Verschlüsselung einstellen. Die Herstellung der Verbindung erfolgt mit dem Aufruf des Konstruktors. Die Verbindung bleibt solange erhalten, bis sie mit *close()* geschlossen wird oder ein Fehler auftritt. Die Klasse L2CapSocket beinhaltet auch die für diese Arbeit essentiellen Methoden *readRSSI(), readTransmitPowerLevel() und getLinkQuality(),* die den Zugriff auf die Signalstärke-Parameter ermöglichen.

- L2CapInputStream und L2CapOutputStream: L2CapInputStream wurde von java.io.InputStream und L2CapOutputStream von java.io.OutputStream abgeleitet. Sie ermöglichen das Senden und Empfangen von Daten über ein L2CapSocket.

- L2CapServerSocket: Diese Klasse nimmt eingehende L2CAP-Verbindungen an. Wie beim L2CapSocket erwartet der Konstruktor eine L2CapSocketAddress, die MTU und den *link_mode.* Mit der Methode *accept()* können dann eingehende Verbindungen auf dem über die L2CapSocketAddress definierten PSM angenommen werden. *Accept()* blo-

ckiert solange, bis eine Verbindung angenommen wird und liefert diese als L2CapSocket zurück [MIS04][39ff].

- JTooth: Manche der in JTooth definierten Methoden passen zu keiner der beschriebenen Klassen der Bibliothek und werden deshalb direkt aus der Klasse JTooth aufgerufen. Dazu gehört auch die wichtige Methode *hci_inquiry(),* die es ermöglicht, Bluetooth-Geräte in der Umgebung zu finden. Über den ersten Parameter kann man angeben, wie lange nach anderen Geräten gesucht werden soll. Die Angabe der Zeitspanne erfolgt dabei in Slots, die 1,28 Sekunden lang sind. Laut Mischke sind sieben Slots ausreichend, um alle Geräte in der Umgebung zu finden. Der zweite Parameter gibt an, wie viele Geräte maximal gefunden werden sollen. Mit dem letzten Parameter kann man festlegen, ob ein Inquiry-Vorgang durchgeführt werden soll oder ob die Anfrage aus dem Cache beantwortet werden kann. Als Ergebnis liefert die Methode ein Array vom Typ BDAddress, das die Bluetooth Device Addresses aller gefundenen Geräte enthält.

- Mit den Methoden *read_page_timeout()* und *wirte_page_timeout()* kann man die Zeitspanne konfigurieren, nach der ein Verbindungsaufbau abgebrochen wird und als fehlgeschlagen gilt. Um diese Funktion ausführen zu können, muss das ausführende Programm über Root-Rechte verfügen, da sich dieser Parameter auch auf andere Programme, die Bluetooth verwenden, auswirkt.

  In der Klasse JTooth sind auch die Methoden gekapselt, welche die Nutzung des SDP-Protokolls ermöglichen. Mit den Methoden *sdp_register_l2cap()* und *sdp_unregister()* kann man Dienste beim lokalen SDP-Server anmelden bzw. abmelden. Die Methode *sdp_search_l2cap()* ermöglicht das Suchen von Diensten auf entfernten Bluetooth-Geräten über den SDP-Client (siehe Kapitel 3.2.8) [MIS04][39].

## 3.9 Kapitelzusammenfassung

Das Kapitel 3 erklärt die Grundlagen der Funktechnologie Bluetooth anhand der einzelnen Schichten des Bluetooth-Protokollstacks. Es werden unter anderem Störquellen wie Wände und Türen besprochen, die eine Schwächung des Funksignals zur Folge haben.

Weiters werden verschiedene Methoden zur Lokalisierung in Funknetzen besprochen, mit dem Ergebnis, dass sich die Radio Signal Strength-Methode am besten für Bluetooth eignet. Um eine Signalstärke berechnen zu können, die Aufschluss über die Distanz zwischen Sender und Empfänger gibt, benötigt man die Parameter Received Signal Strength Indicator (RSSI), Transmit Power Level (TPL) und Link Quality (LQ). Alle drei Werte werden über das Host Controller Interface (HCI) ausgelesen. Der Bluetooth-Softwarestack BlueZ unter Linux ermöglicht einen Zugriff auf diese Parameter. Die JTooth Library ermöglicht die Anbindung von Java an BlueZ und wird in diesem Projekt zur Erstellung einer Messsoftware genutzt.

# 4 Künstliche neuronale Netze

*Dieses Kapitel gibt eine Einführung in künstliche neuronale Netze (KNN). Es werden zunächst kurz die Grundlagen eines KNN erklärt. Danach wird der Netztyp: Multilayer Feedforward Neural Network näher besprochen, der im praktischen Teil der Arbeit zum Einsatz kommt. Der Rest des Kapitels widmet sich dem Training dieses Netztyps sowie den Methoden, das Training optimal zu gestalten und die Performance des KNNs zu verbessern. Der letzte Punkt 4.5 erklärt, wie man überprüfen kann, wie gut ein KNN seine trainierte Aufgabe erfüllt.*

„Die Lernmethoden von KNNs stellen einen robusten Ansatz dar, um Zielfunktionen, die auf realen, diskreten oder Vektorwerten basieren, zu approximieren. Für bestimmte Probleme, wie das Lernen der Interpretation von komplexen realen Sensordaten, gehören KNNs zu den effektivsten Methoden, die derzeit bekannt sind.“ (aus dem Englischen übersetzt) [TOM97][81]

Genau für dieses Problem: „…die Interpretation von komplexen realen Sensordaten“ sollen KNNs in dieser Arbeit verwendet werden. Bevor der Ansatz im Verlauf der Arbeit im Detail besprochen wird, wird hier zunächst eine Einführung in den Bereich der KNNs gegeben.

KNNs stellen vereinfacht Modelle des menschlichen Nervensystems dar. Unser Nervensystem ist ein hoch komplexes, nicht lineares und paralleles Informationsverarbeitungssystem. Es hat die Fähigkeit, seine Neuronen so zu organisieren, dass es bestimmte Aufgaben wesentlich schneller erledigen kann als der schnellste Computer. Dies erreicht das menschliche Gehirn, indem es sich durch einen Lernprozess seiner Umgebung anpasst. Dieses Prinzip wird auch bei KNNs verwendet. Das KNN passt sich ebenfalls einer bestimmten Aufgabe an, indem es sich in einem ähnlichen Lernprozess das Wissen über seine Umgebung aneignet. Es speichert das Wissen dadurch, dass es Verbindungen zwischen zwei Neuronen ein bestimmtes Gewicht zuteilt. Diese Fähigkeit zu lernen und daher in der Lage zu sein, zu generalisieren, ist die Stärke eines KNNs. Es ist so in der Lage, auch vernünftige Outputs für Input-Daten zu liefern, die während der Lernphase nicht trainiert wurden.

## 4.1 Komponenten eines künstlichen neuronalen Netzes

„Ein neuronales Netz ist ein Verbund von Einheiten, in dem diese nach einem bestimmten Muster miteinander verbunden sind und der somit die Kommunikation zwischen den Einheiten ermöglicht. Diese Einheiten werden auch als Neuronen oder Knoten bezeichnet und sind einfache Prozessoren, deren Informationsverarbeitungsfähigkeiten in der Regel beschränkt sind auf eine Regel zum Kombinieren von Eingangssignalen und eine Aktivierungsregel, welche die kombinierten Eingangssignale zum Berechnen eines Ausgangssignals verwendet. Die Ausgangssignale können über Verbindungen, die als Gewichte bezeichnet werden, an andere Einheiten übertragen werden. Die Gewichte erregen oder hemmen das zu kommunizierende Signal. [ROB99][17f]“ Ein solches Neuron eines KNNs ist in Abbildung 17 dargestellt.

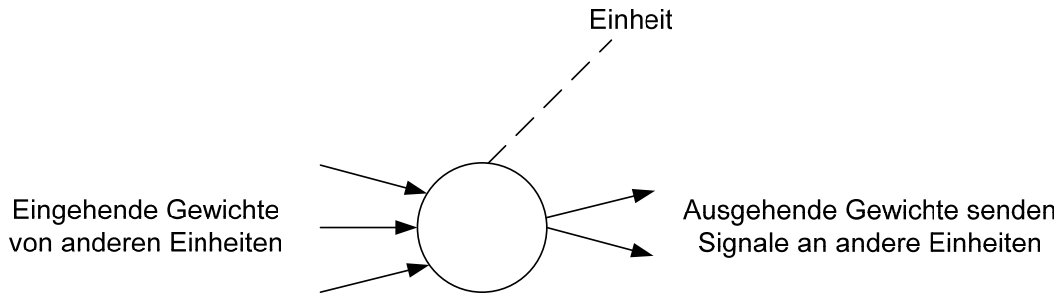


Abbildung 17: eine einfache Netzeinheit [ROB99] [17]

Eine solche einzelne Einheit verfügt über sehr eingeschränkte Datenverarbeitungsfähigkeiten. Werden aber viele von ihnen zu einem Netz verbunden, ist dies in der Lage, eine komplexe Aufgabe durchzuführen. Die Aufgabe, die ein Netz lernt, wird in den Gewichten gespeichert. Die Werte der Gewichte werden während der Trainingsphase bestimmt.

In einem KNN kann man drei Arten von Einheiten bzw. Neuronen unterscheiden:

- Input-Neurons: erhalten Signale/Eingabewerte von der Außenwelt und stellen somit den Input für das KNN dar. Blau dargestellt in Abbildung 18.
- Hidden-Neurons: befinden sich zwischen Input- und Output-Neuronen.
- Output-Neurons: geben Signale an die Außenwelt ab und stellen somit den Output des KNN dar. Rot dargestellt in Abbildung 18.

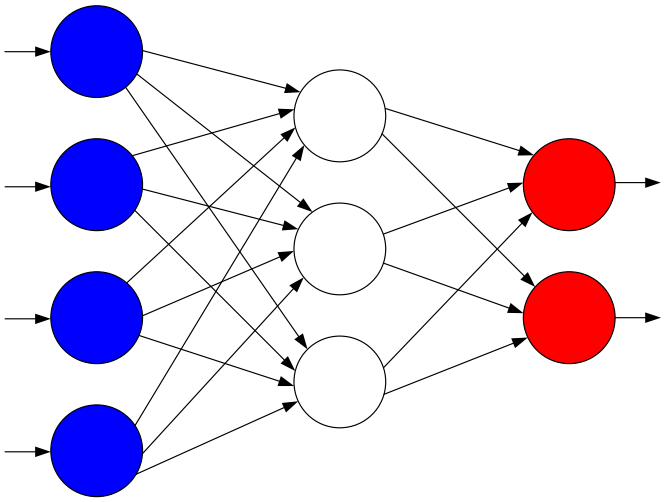

Abbildung 18: schematische Darstellung eines neuronalen Netzes [NNE07]

## 4.2 Multilayer Feedforward Neural Network

Im Laufe der Zeit wurden sehr viele verschiedene Typen von KNNs entwickelt. In dieser Arbeit wurde ein Multilayer Feedforward Neural Network (MLFF) verwendet. Ein solches Netz besteht aus einem Input-Layer, einem oder mehreren mittleren Hidden-Layers und einem Output-Layer. Abbildung 18 zeigt ein MLFF mit einem Hidden-Layer, der sich aus drei Hidden-Neurons zusammensetzt. Die Hidden-Layers werden als versteckt bezeichnet da sie nur interne Eingaben empfangen und interne Ausgaben erzeugen. Sie bleiben also der Außenwelt verborgen. Indem man einen oder mehrere Hidden-Layers dem Netz hinzufügt, ermöglicht man, dass es komplexe Zusammenhänge in den Daten erkennt. Die Verwendung eines Hidden-Layers ist speziell wichtig, wenn das Netz sehr viele Input-Neurons besitzt.

Der Durchlauf in einem MLFF sieht folgendermaßen aus: Die Elemente des Input-Layers werden über die gewichteten Verbindungen dem ersten Hidden-Layer präsentiert. Die Elemente dieses Layers errechnen ihren Output, welcher dann als Input für den nächsten Layer dient. Dies wird solange fortgesetzt, bis der Output-Layer erreicht wird.

Wenn jedes Element in jedem Layer mit jedem Element im nachfolgenden Layer verbunden ist, spricht man von einem fully connected, ansonsten von einem partially connected KNN [DAN96][159ff] (Beginnend von Punkt 4.2). Das KNN in Abbildung 18 ist fully connected.

„MLFF-Netzwerke haben große Beliebtheit erlangt, weil es damit möglich ist, beliebige Abbildungen $g : R^n \rightarrow R^m$ vorzunehmen, wobei gilt $g(x) = z$. Diese Abbildungen sind möglich, wenn genügend viele verborgene Einheiten bereitgestellt werden und wenn das Netzwerk trainiert werden kann, d.h. wenn man eine Gewichtungsmenge findet, die die gewünschte Abbildung vornimmt.“ [DAN96][160]

## 4.3 Training eines MLFF mit dem Backprogagation-Algorithmus

Das Training ist der Prozess, in dem sich das KNN an seine Umgebung anpasst und seine Aufgabe lernt. Dazu müssen zuerst Daten in der Umgebung erhoben werden, die dann dem KNN präsentiert werden können. In dieser Arbeit werden diese Daten durch das in Kapitel 6.2 beschriebene Experiment gewonnen. Sie bestehen aus den Input- und den korrekten Output-Daten.

Für das Training wurde dann der Backprogagation-Algorithmus verwendet. Dieser kann verwendet werden, um ein MLFF-Netzwerk zu trainieren, da er angibt, wie die Gewichtungen in den Hidden-Layers angepasst werden sollen. Der Algorithmus ist ein Optimierungsverfahren, das auf der Gradientensteigung basiert. Er passt die Gewichtungen so an, dass der Gesamtfehler reduziert wird. Der Algorithmus ist in drei Phasen gegliedert:

- Forward-Pass: In der Lernphase werden dem KNN Eingabemuster präsentiert. Für jedes Trainingsmuster wird die Ausgabe berechnet.

- Fehlerbestimmung: Die berechnete Ausgabe wird mit der gewünschten Ausgabe verglichen und der Fehlerwert ermittelt.

- Backward-Pass: Der errechnete Fehlerterm breitet sich rückwärts vom Output- zum Input-Layer durch das Netz aus. Mit Hilfe der Fehlerterme werden die Gewichte im Netz so verändert, dass der Fehler kleiner wird.

Dieser Vorgang wird für jedes Muster der Trainingsmenge solange wiederholt, bis der Fehler das gewünschte Minimum erreicht hat oder eine bestimmte Anzahl von Trainingsiterationen erreicht ist [NNE07] [DAN96][162].

Wie die Berechnung des Fehlers genau vorgenommen wird, soll hier noch etwas genauer erklärt werden. Es wird versucht, die Menge Gewichtungen zu finden, die die

Fehlerfunktion für alle Trainingsmusterpaare minimiert. Beim Backprogagation-Algorithmus erfolgt die Minimierung hinsichtlich des mittleren quadratischen Fehlers des Mean Square Error (MSE). Er wird mit folgender Formel berechnet:

$$E_p = \frac{1}{2}\sum_{i=1}^{n}(t_i - o_i)^2$$

$$E_{tot} = \frac{1}{n}\sum_{p=1}^{n} E_p$$

Es wird zunächst der quadratische Fehler *Ep* für jedes Muster berechnet. *n* ist die Anzahl der Muster, die dem Netz vorgestellt werden. *t* ist die Soll-Ausgabe und *o* die Ist-Ausgabe. Die Soll-Ausgabe wird auch Target, die Ist-Ausgabe Output genannt. Der Faktor ½ dient zur Vereinfachung. Im zweiten Schritt wird dann der durchschnittliche Fehler berechnet. Man könnte natürlich auch den maximalen oder Median-Fehler versuchen zu minimieren, der MSE ist aber der gebräuchlichste [DAN96][177].

Wenn im weiteren Verlauf der Arbeit von einem Fehler in Bezug auf ein KNN gesprochen wird, ist immer der MSE gemeint.

## 4.4 Optimierungsmethoden

Damit ein KNN seine spätere Aufgabe optimal erfüllen kann, gibt es mehrere Methoden, um das zu fördern. Eine Möglichkeit besteht darin, die Daten, die das KNN verarbeiten soll zu, optimieren. Die Generalisierung eines KNNs ist ebenfalls ein wichtiger Punkt. Unter Generalisierung versteht man die Eigenschaft eines KNNs sinnvolle Outputs für nicht trainierte Muster zu liefern. Das Early Stopping ist eine Methode, die das sicherstellen soll. Die hier beschriebenen Methoden wurden für das Training der KNNs, die im Kapitel 7.2 vorgestellt werden, verwendet.

### 4.4.1 Normalisierung

Das Normalisieren der Input- und Output-Daten in einen Bereich von 0 bis 1 ermöglicht der Transferfunktion einen besseren Umgang mit den Daten. Noch besser ist ein Wertebereich von 0,1 bis 0,9 da man sich so im Bereich befindet, der in den Hidden-

Layers verwendet wird. Dies ermöglicht eine bessere Nutzung der Aktivierungsfunktion und man nutzt den Wertebereich optimal aus [TOM97][114f].

Trainiert man ein Netz mit normalisierten Werten, dann benötigt man im Realbetrieb zusätzliche Pre- und Postprocessing-Schritte, in denen die Input- und Output-Daten normalisiert werden.

### 4.4.2 Early Stopping

Beim Early Stopping werden die zur Verfügung stehenden Daten in eine Trainings-, Validations- und Testmenge unterteilt. Das Trainingsset wird wie üblich für den Lernprozess verwendet. Der Fehler für das Validationsset wird ebenfalls beobachtet. Der Fehler für das Trainingsset sinkt kontinuierlich während des Lernvorgangs. Beim Validationsset ist dies zu Beginn auch der Fall. Wird der Fehler für das Validationsset ab einem bestimmten Zeitpunkt wieder kontinuierlich größer, spricht man von Overfitting. Das bedeutet, das KNN beginnt die Trainingsdaten auswendig zu lernen und verliert so seine Fähigkeit, einen vernünftigen Output für nicht trainierte Daten zu generieren. Beim Early Stopping wird der Lernprozess aus diesem Grund dann abgebrochen, wenn der Fehler für das Validationsset wieder zu steigen beginnt, um so ein Overfitting zu vermeiden und die Eigenschaft der Generalisierung des KNNs zu sichern. Das Testset kann dazu verwendet werden, um das Validationsset zu kontrollieren. Sollte der Fehler nur für das Validationsset steigen aber nicht für das Testset, ist das ein Anzeichen für eine schlechte Aufteilung der Daten auf die drei Subsets. Ein gebräuchlicher Ansatz zur Aufteilung der Datenmenge ist 60% der Daten für das Trainingsset und jeweils 20% für das Validations- und Testset zu verwenden [NNT07][175].

## 4.5 Analyse von KNNs

Um die Performance eines trainierten KNN festzustellen, kann man sich z.B. die Größe des Fehlers für das Trainings-, Validations- und Testset ansehen. Es ist jedoch ratsam, sich den Output des KNNs detaillierter anzusehen. Eine Möglichkeit dazu ist die Durchführung einer Regressionsanalyse zwischen dem Output des KNN und den realen Daten (Target). Diese liefert zwei Werte die Aufschluss über die Performance des KNN geben. Zum einen die Steigung (m) der Geraden, die die beste lineare Regression zwischen dem Output des KNN und Target darstellt. Würde das KNN in allen Fällen einen perfekten Output generieren, also der Output entspricht immer den

realen Daten, wäre die Steigung 1. Der zweite Wert ist der Korrelationskoeffizient (r), welcher den linearen Zusammenhang zwischen dem Output und dem Target angibt. Liegt er bei 1 entspricht der Output genau der Realität. Liegt er bei 0 gibt es keinen Zusammenhang [NNT07][189].

In Abbildung 19 sieht man zwei Beispiele für eine Regressionsanalyse. Das KNN links liefert einen Output, der ziemlich exakt dem Target entspricht. Die Steigung der Geraden (rote Linie) liegt nahe an 1 genauso wie der Korrelationskoeffizient. Die blauen Kreise stellen den vom KNN gelieferten Output dar. Die blau strichlierte Linie hat eine Steigung von 1 und dient als Bezugspunkt.

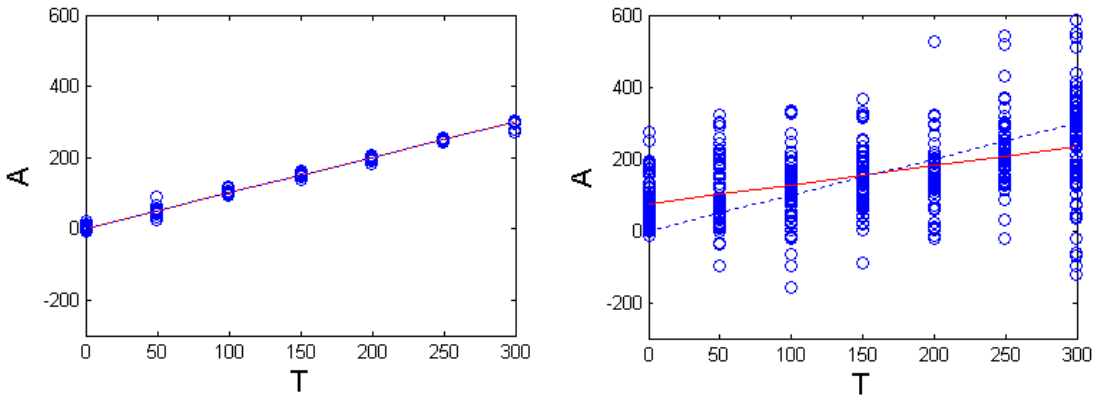


Abbildung 19: Regressionsanalyse

Die Regressionsanalyse für das andere KNN, rechts in Abbildung 19 zu sehen, fällt sehr schlecht aus, wie man an der geringen Steigung sehen kann. Das KNN liefert einen schlechten Output.

## 4.6 Kapitelzusammenfassung

Das Kapitel gibt einen Einblick in den Aufbau und die Funktionsweise von KNNs. Es wurden die Bestandteile eines Netzes sowie der spezielle Netztyp Multilayer Feedforward Neural Network (MLFF) besprochen, der über ein oder mehrere Hidden-Layers verfügt. In weiterer Folge wurde der Backprogagation-Algorithmus besprochen der für das Training eines MLFF verwendet wird. Die Methoden der Normalisierung und des Early Stoppings, die zur Performancesteigerung eines KNNs verwendet werden, wurden ebenfalls beleuchtet. Den Abschluss bildet die Analyse von KNNs, die beschreibt, wie mittels einer Regressionsanalyse die Leistungsfähigkeit eines KNNs beurteilt werden kann.

# 5 Erstellen des Messsystems

*Der zweite Teil der Arbeit schildert die praktischen Tätigkeiten, die im Laufe des Projekts durchgeführt wurden. In einem Versuch wurden drei Bluetooth-Module hinsichtlich der Qualität ihrer Signalstärkeparameter untersucht (Kapitel 6.1). Die notwendigen Messdaten für die Versuche zur Lokalisierung mit neuronalen Netzen wurden in einem weiteren Experiment gewonnen. Dabei konnte auch der Einfluss der Umgebung auf die gemessenen Signalstärkewerte gezeigt werden (Kapitel 6.2). Um diese Versuche durchführen zu können, war die Implementierung eines Messsystems notwendig. Die Architektur und Funktionsweise des Systems werden nun am Beginn des praktischen Teils besprochen. Hinweise für die praktische Handhabung sind im Benutzerhandbuch (Kapitel 9.1) im Anhang der Arbeit zu finden.*

Das Messsystem besteht aus den Programmen BTLocationServer und BaseStation. Die Anwendung BaseStation erfüllt, wie der Name vermuten lässt, die Aufgabe einer Base Station. In dieser Hälfte der Arbeit bezeichnet der Begriff Base Station (BS) nun immer einen Rechner, auf dem das Programm BaseStation ausgeführt wird. Der Begriff Server bezeichnet den Rechner, der die Anwendung BTLocationServer ausführt.

Die Base Stations suchen nach Bluetooth-Geräten, bauen zu ihnen eine Verbindung auf, messen die Signalstärkeparameter und übermitteln sie an den Server. Dieser archiviert die gesammelten Messwerte in Dateien für eine spätere Analyse (siehe Abbildung 20). Für die Implementierung der Programme wurde die Programmiersprache Java verwendet.

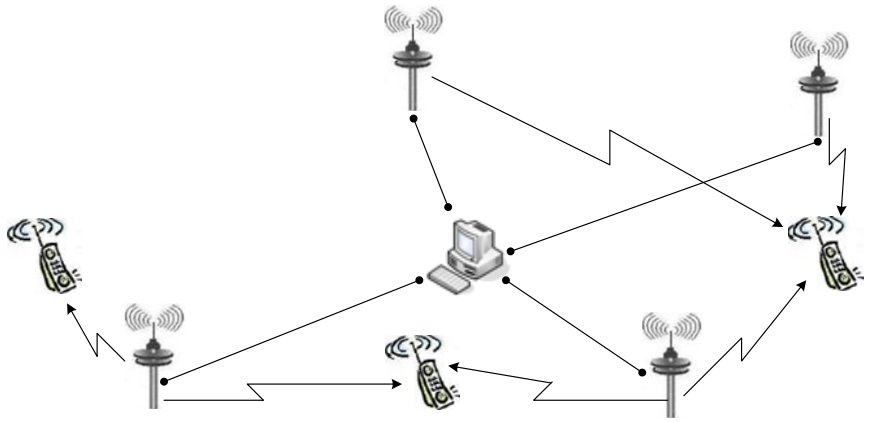

Abbildung 20: Funktionsweise des Messsystems

## 5.1 Design und Funktionsweise der BaseStation

Die Funktionsweise der BS wird mit Hilfe des UML-Diagramms in Abbildung 21 erklärt.

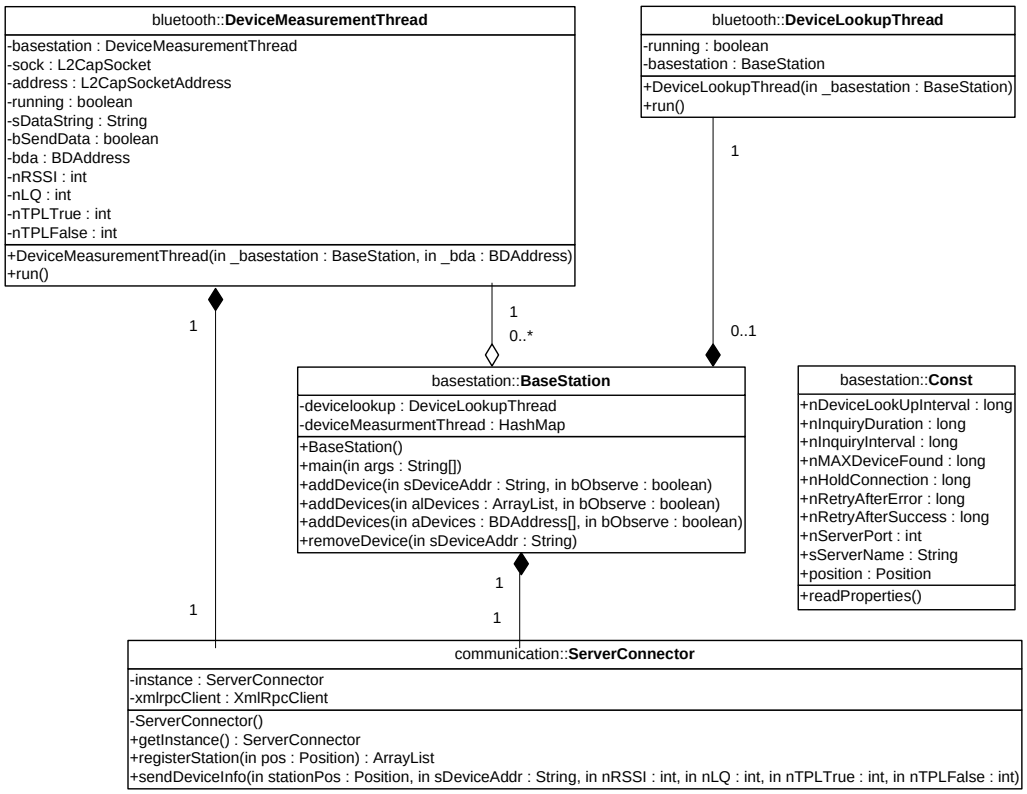


Abbildung 21: BaseStation

Nach dem Start des Programms wird durch die Methode *readProperties* der Klasse *Const* die Konfigurationsdatei der BS gelesen. Diese enthält nur die Informationen, welche für die Anmeldung beim BTLocationServer notwendig sind. Das sind die Position der BS in Form der X-, Y- und Z-Koordinate, sowie der Rechnername und der Port, auf dem der BTLocationServer läuft. Die BS meldet sich dann beim BTLocationServer mit ihren Koordinaten, die sie eindeutig identifizieren, an. Der Server übermittelt der BS in seiner Antwort die restlichen Einstellungen. Diese werden gemeinsam mit den Informationen aus der Konfigurationsdatei in der Klasse *Const* gekapselt. Die Attribute dieser Klasse sind *public* und *static* definiert. Das bedeutet, die anderen Klassen können jederzeit auf diese allgemeinen Einstellungen zugreifen. Der Server liefert der Base Station eine Liste mit den Bluetooth-Adressen der übri-

gen BSs. Die BS registriert sie als Bluetooth-Geräte über die Methode *addDevices*. Der Parameter *bObserve* ist dabei auf *false* gesetzt. Dadurch wird verhindert, dass die Base Stations untereinander Bluetooth-Verbindungen aufbauen.

Der Server hat der Base Station bei der Registrierung auch mitgeteilt, in welchem Modus sie arbeiten soll. Das Programm BaseStation unterstützt drei Modi:

- Development: Es werden drei Dummy Bluetooth-Geräte registriert, und die Base Station übermittelt zufällige Messwerte an den Server. Der Modus wird dazu verwendet, um die Kommunikation zwischen Server und Base Station zu testen. Er wurde während der Entwicklungsphase der Programme häufig genutzt.

- Measurement: In diesem Modus sendet der Server der Base Station die Bluetooth-Adresse eines bestimmten MD. Die Base Station stellt nur zu diesem Gerät eine Verbindung her und überträgt die Signalstärkedaten an den Server. Zum Durchführen der Experimente wurde dieser Modus verwendet.

- Real: Die Base Station sucht in regelmäßigen Abständen nach Bluetooth-Geräten in ihrer Umgebung. Sie stellt zu den gefundenen Geräten eine Verbindung her, misst die Signalstärkeparameter und sendet sie an den Server.

### 5.1.1 Verwendung der JTooth Library

Die JTooth Library wird verwendet, um BlueZ in Java nutzen zu können. Sie wurde im Kapitel 3.8 vorgestellt. Im Programm BaseStation kommt sie in den Klassen *DeviceLookupThread* und *DeviceMeasurementThread* zum Einsatz.

Ein *DeviceLookupThread* wird von der Klasse *BaseStation* gestartet, wenn das Programm im Modus „Real“ läuft. Er sucht in regelmäßigen Abständen nach Bluetooth-Geräten, die sich in der Nähe der BS befinden. Dazu verwendet er die Methode *hci_inquiry* der Klasse *JTooth,* die Teil der JTooth Library ist. Als Returnwert liefert sie die Bluetooth-Adressen der gefundenen Geräte. Der *DeviceLookupThread* meldet die gefundenen Objekte, indem er die Methode *addDevices* der Klasse *BaseStation* aufruft. Sofern die gefundenen Bluetooth-Geräte bei der BS noch nicht registriert wurden, wird für jedes von ihnen ein *DeviceMeasurementThread* gestartet.

Der *DeviceMeasurementThread* wird mit der Bluetooth-Adresse des Gerätes, zu dem er eine Verbindung herstellen soll, initialisiert. Mit der Klasse *L2CapSocket* richtet

er dann eine L2CAP-Verbindung zum angegebenen Gerät ein. Es wird der freie PSM 4097 genutzt. Wenn die Verbindung hergestellt wurde, werden die Signalstärkeparameter über die Methoden *readRSSI, getLinkQuality* und *readTransmitPowerLevel* der Klasse *L2CapSocket* gelesen. Der Transmit Power Level wird zweimal gelesen, einmal die maximale Sendeleistung und dann noch die momentane Sendeleistung. Die vier gelesenen Werte werden dann an den BTLocationServer übertragen.

Wenn die BS im Modus „Measurement“ läuft, wird ein *DeviceMeasurementThread* nach der Registrierung am Server für das von ihm gewünschte Gerät gestartet. Ein *DeviceLookupThread* wird in diesem Modus nicht verwendet.

### 5.1.2 Kommunikation mit dem Server

Zur Kommunikation zwischen den BSs und dem Server wird das XML-RPC-Protokoll verwendet. Die Methoden zur Kommunikation auf der BS-Seite sind in der Klasse *ServerConnector* gekapselt. Zur Anmeldung der BS am Server wird die Methode *registerStation* verwendet. Sie erhält die Konfiguration für die BS als Antwort. Die Methode *sendDeviceInfo* wird vom *DeviceMeasurementThread* genutzt, um die gemessenen Werte an den Server zu schicken.

## 5.2 Design und Funktionsweise des BTLocationServers

Das Programm BTLocationServer fungiert als Server im Messsystem und verwaltet ein Netz von Base Stations. Es verfügt über eine graphische Benutzeroberfläche und es erlaubt dem Nutzer Signalstärkemessungen durchzuführen.

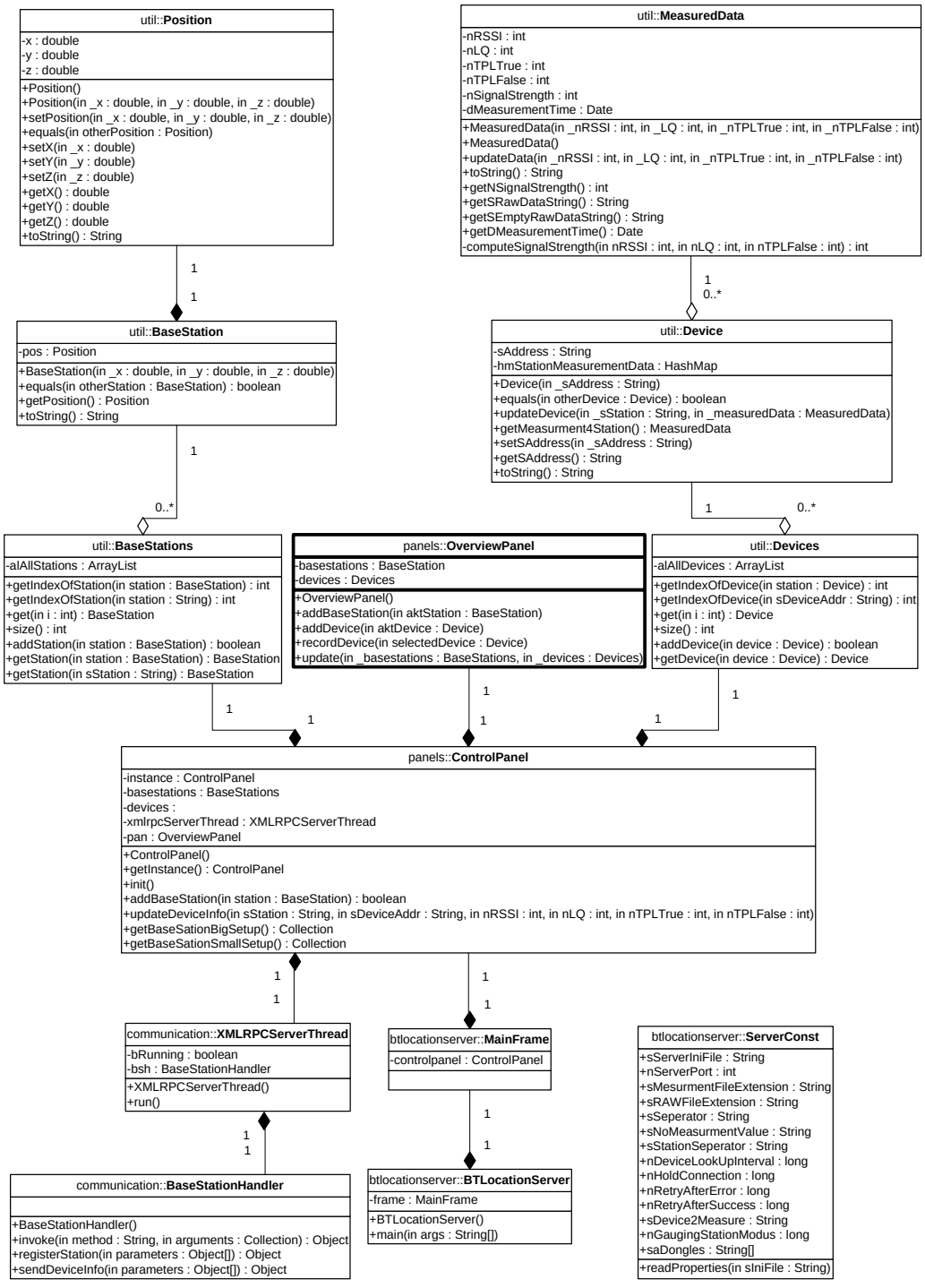


Abbildung 22: BTLocationServer, erste Hälfte

Das Klassendiagramm in Abbildung 22 zeigt den Aufbau des BTLocationServers. Das Diagramm ist aus Platzgründen auf Abbildung 22 und Abbildung 24 aufgeteilt. Die Klasse *OverviewPanel* stellt das Bindeglied zwischen den Abbildungen dar und ist deshalb fett dargestellt.

Beim Start des Programms BTLocationServer wird die Konfigurationsdatei gelesen. Das geschieht in der Klasse *ServerConst* in der Methode *readProperties*. In der Datei sind unter anderem festgelegt:

- der Port, auf dem der BTLocationServer Anfragen der BSs entgegennimmt,
- der Modus, in dem die BSs laufen soll,
- die Bluetooth-Adressen der einzelnen BSs.

Alle Einstellungen werden in der Klasse *ServerConst* gekapselt. Im nächsten Schritt wird das Hauptfenster der Anwendung aufgebaut, das durch die Klasse *MainFrame* repräsentiert wird. Das *MainFrame* fügt das *ControlPanel* der Benutzeroberfläche hinzu. Die Klasse *ControlPanel* ist die zentrale Klasse des gesamten Systems. Hier laufen alle Informationen des Messsystems zusammen. Aus diesem Grund gibt es nur eine Instanz dieser Klasse. Um das sicherzustellen, wurde sie dem Singleton-Pattern entsprechend gestaltet. Beim ersten Zugriff auf die Klasse startet sie einen *XMLRPCServerThread*. Von nun an ist der BTLocationServer für die BSs erreichbar.

Die Anfragen der Base Stations werden durch die Klasse *BaseStationHandler* verarbeitet. Sie enthält die Methoden *registerStation* und *sendDeviceInfo,* die von den Base Stations über XML-RPC aufgerufen werden können. Beim Start einer BS meldet sich diese über die Methode *registerStation* am BTLocationServer an. Die Klasse *BaseStationHandler* meldet dem *ControlPanel* die neue BS über die Methode *addBaseStation*. Als Antwort auf die Registrierung erhält die BS eine Reihe von Einstellungsparametern.

Zur Verwaltung der BSs nutzt das *ControlPanel* die Klasse *BaseStations*. Diese speichert für jede registrierte BS eine Instanz der Klasse *BaseStation* in einer *ArrayList* und ermöglicht einen komfortablen Zugriff auf diese über diverse Methoden.

Sobald eine BS mit einem Bluetooth-Gerät verbunden ist und Signalstärkeparameter gemessen hat, sendet sie diese an den Server. Dazu ruft sie die Methode *sendDeviceInfo* über XML-RPC auf. Der *BaseStationHandler* gibt die Messdaten an das *ControlPanel* weiter, indem er dessen Methode *updateDeviceInfo* aufruft. Dort werden mittels der Klasse *Devices*, welche die gefundenen Bluetooth-Geräte verwaltet, überprüft, ob bereits eine Instanz der Klasse *Device* für das Bluetooth-Gerät angelegt wurde. Falls noch keine existieren sollte, wird eine angelegt. Die empfangenen Messdaten werden in einem neuen Objekt der Klasse *MeasuredData* gespeichert.

Dieses wird bei der entsprechenden Instanz der Klasse *Device* durch Aufruf der Methode *updateDevice* gespeichert. Es wird natürlich auch vermerkt, welche BS die Messdaten genommen hat. Weiters wird der Satz von Messdaten mit einem Zeitstempel versehen. Wenn neue Messdaten von derselben Base-Station eintreffen, überschreiben sie die alten.

### 5.2.1 Berechnung der Signalstärke

In der Klasse *MeasuredData* wird auch die Signalstärke der Verbindung zwischen BS und MD ermittelt. Wie im Theorieteil erklärt wurde, können die Signalstärkeparameter bei den einzelnen Bluetooth-Modulen unterschiedlich festgelegt sein. Darum wurde in einem Versuch ein geeignetes Modul ausgewählt. Der Algorithmus zur Berechnung wurde anhand der Ergebnisse des Experiments entwickelt und ist auf das gewählte Bluetooth-Modul zugeschnitten. Der Versuch wird zwar in Kapitel 6.1 besprochen, der Algorithmus aber bereits hier, da er Teil des Messsystems ist.

Die errechnete Signalstärke liegt zwischen 100 und 900. Dieser Wertebereich wurde gewählt, um einen idealen Input für die neuronalen Netze zu haben. Die Eingabewerte für ein neuronales Netz sollten zwischen 0,1 – 0,9 liegen. Die Signalstärke kann so einfach umgerechnet werden.

Beim Erstellen einer neuen Instanz der Klasse *MeasuredData* wird in der Methode *computeSignalStrength* die Signalstärke berechnet. Dazu erhält sie den gemessenen RSSI-, LQ- und TPL-Wert. Die Variable *TPLFalse* steht für die aktuelle Sendeleistung, nicht für die maximale.

```
private int computeSignalStrength(int rssi, int lq,
int nTPLFalse) {
    int nSignalStrength = 0;
    // 3 <= scaledRSSI <= 30
    int scaledRSSI = 30 + Math.min(rssi, 0);
    // 90 <= scaledRSSI <= 900
    scaledRSSI = scaledRSSI * 30;
    if (nTPLFalse != 4) {
        nSignalStrength = 900;
    } else {
        nSignalStrength = scaledRSSI;
        if (nSignalStrength < 100) {
            nSignalStrength = 100;
        } else if (nSignalStrength > 900) {
            nSignalStrength = 900;
        }
    }
    return nSignalStrength;
}
```

Abbildung 23: Berechnung der Signalstärke

Der RSSI-Wert wird für die Berechnung genutzt, da er sich in den Experimenten als aussagekräftigster Parameter erwies. Für das gewählte Bluetooth-Modul konnten RSSI-Werte von -27 bis 0 gemessen werden. Der gemessene RSSI-Wert wird zuerst zu 30 addiert, um in einen positiven Wertebereich zu kommen. Er hat dann einen Wert zwischen 3 und 30. Im nächsten Schritt wird er mit dem Faktor 30 multipliziert. Für eine gute Verbindung zwischen BS und MD mit einem RSSI von 0 ist somit eine Signalstärke von 900 errechnet. Für einen sehr schlechten RSSI-Wert von -27, der nur einmal gemessen wurde, liegt die Signalstärke bei 90. Die Signalstärke wird mit einem IF-Statement auf ihr Minimum von 100 geprüft. Sollte sie sich darunter befinden, wird sie auf das Minimum gesetzt.

Der gemessene TPL-Wert wird bei der Bestimmung der Signalstärke ebenfalls berücksichtigt. Sollte die eigene Sendeleistung einen Wert geringer als 4 haben, das heißt, sie wurde gesenkt, lässt das auf eine sehr gute Verbindung zwischen BS und MD schließen, und die Signalstärke wird auf ihr Maximum gesetzt.

### 5.2.2 Speichern der Messdaten

Der BTLocationServer ermöglicht es, die gemessenen und berechneten Werte aufzuzeichnen. (Eine genaue Anleitung zur Benutzeroberfläche des BTLocationServer finden sie in Kapitel 9.1.1. Hier wird zunächst das Design des Servers anhand der Klassendiagramme erklärt.)

Im *OverviewPanel* werden dem Benutzer die gefundenen Bluetooth-Geräte angezeigt. Der Benutzer wählt das Gerät, von dem er die Messdaten speichern will. Im erscheinenden *RecordMeasuredDataDialog* hat der Benutzer noch die Möglichkeit, Einstellungen für die Aufzeichnung vorzunehmen. Er kann

- die Koordinaten des Messpunktes eingeben, an dem sich das Bluetooth-Gerät befindet,
- einen Namen für das Bluetooth-Gerät vergeben,
- die Dauer der Messung festlegen,
- die Zeitabstände, in denen Messwerte genommen werden
- und das maximale Alter eines Messdatensatzes

einstellen. Der Benutzer startet dann die Aufzeichnung über eine Schaltfläche. Dabei wird eine Instanz der Klasse *WriterThread* mit den vom Benutzer getroffenen Einstellungen angelegt.

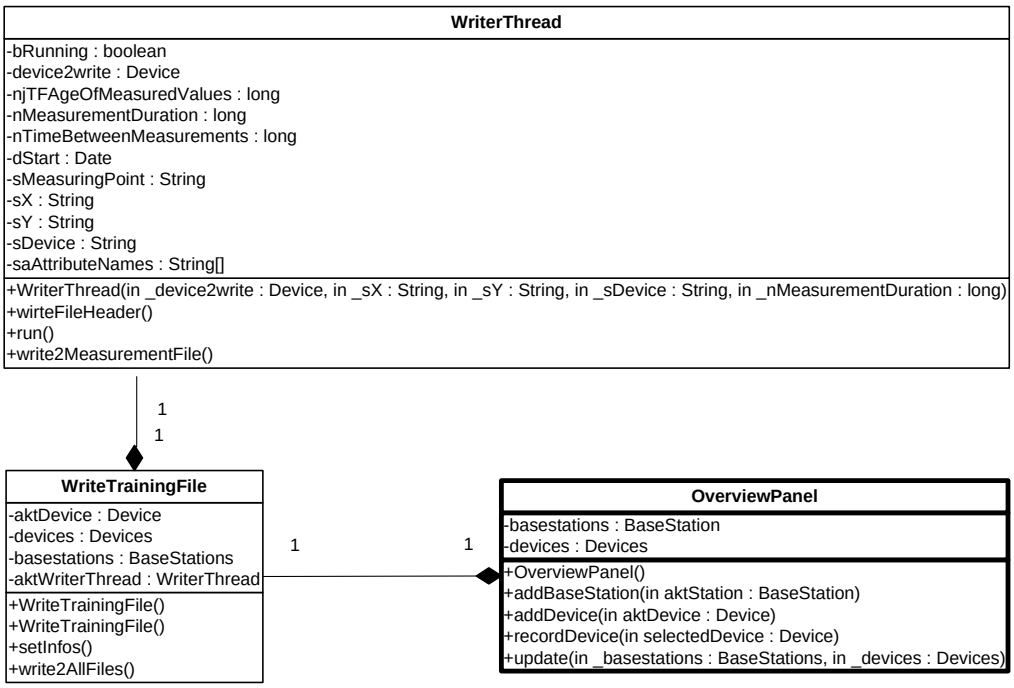


Abbildung 24: BTLocationServer, zweite Hälfte

Der *WriterThread* greift während der gesamten Messdauer im eingestellten Intervall auf die Werte, die von den Base Stations für das Bluetooth-Gerät gemessen wurden, zu. Dazu verwendet er die Methode *getMeasurment4Station* der Klasse *Device*. Natürlich werden die Messwerte im Hintergrund vom System laufend aktualisiert.

Der *WriterThread* legt zwei Dateien an, in denen die Messdaten aufgezeichnet werden. In der ersten wird nur der errechnete Signalstärkewert der BSs aufgezeichnet. In der zweiten werden alle gemessenen Werte gespeichert. Die Namen der Dateien setzen sich aus den eingegebenen Koordinaten und dem Namen für das Bluetooth-Gerät zusammen. An den Namen der zweiten Datei wird noch ein Suffix gehängt, das über die Serverkonfigurationsdatei eingestellt werden kann.

Hier sieht man ein Beispiel für eine Datei, in der nur die errechneten Signalstärkewerte gespeichert werden:

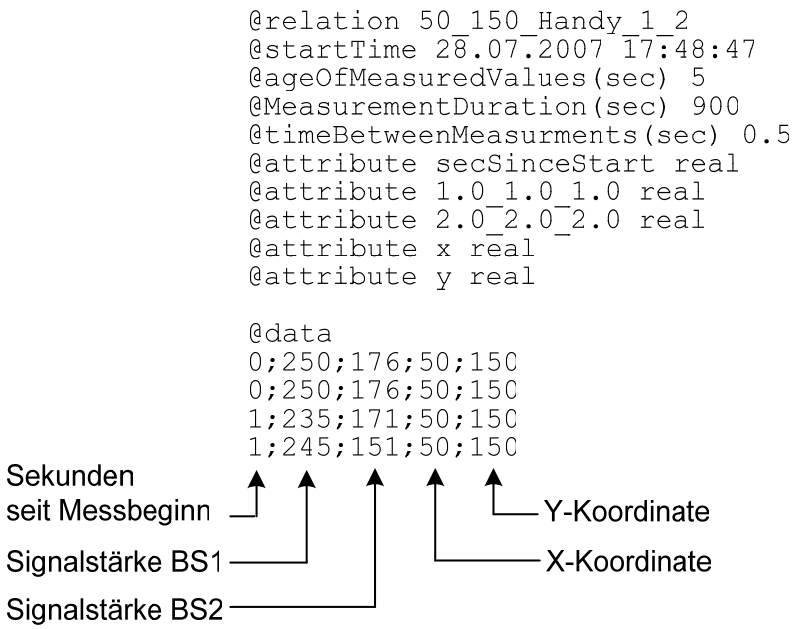


Abbildung 25: Datei mit errechneten Signalstärkewerten

Am Beginn der Datei werden die vom Benutzer getroffenen Einstellungen für die Messung gespeichert. Die mit *@attribute* beginnenden Zeilen beschreiben die Attribute eines Messdatensatzes. Nach dem Eintrag *@data* werden die gemessenen Daten geschrieben. Der erste Eintrag eines Datensatzes gibt an, wie viele Sekunden seit dem Beginn der Messung vergangen sind. Danach folgen die errechneten Signalstärkewerte der BSs, die mit dem Bluetooth-Gerät verbunden sind. Für BSs die nicht mit dem Gerät verbunden sind oder deren gemessene Signalstärkewerte zu alt sind, wird

der Wert, der in der Serverkonfigurationsdatei beim Parameter *NoMeasurmentValue* eingetragen ist, gespeichert. Die letzten beiden Parameter geben die X- und Y-Koordinate an, wo sich das Bluetooth-Gerät befindet.

Hier ein Beispiel für eine Datei, in der alle gemessenen Werte gespeichert werden:

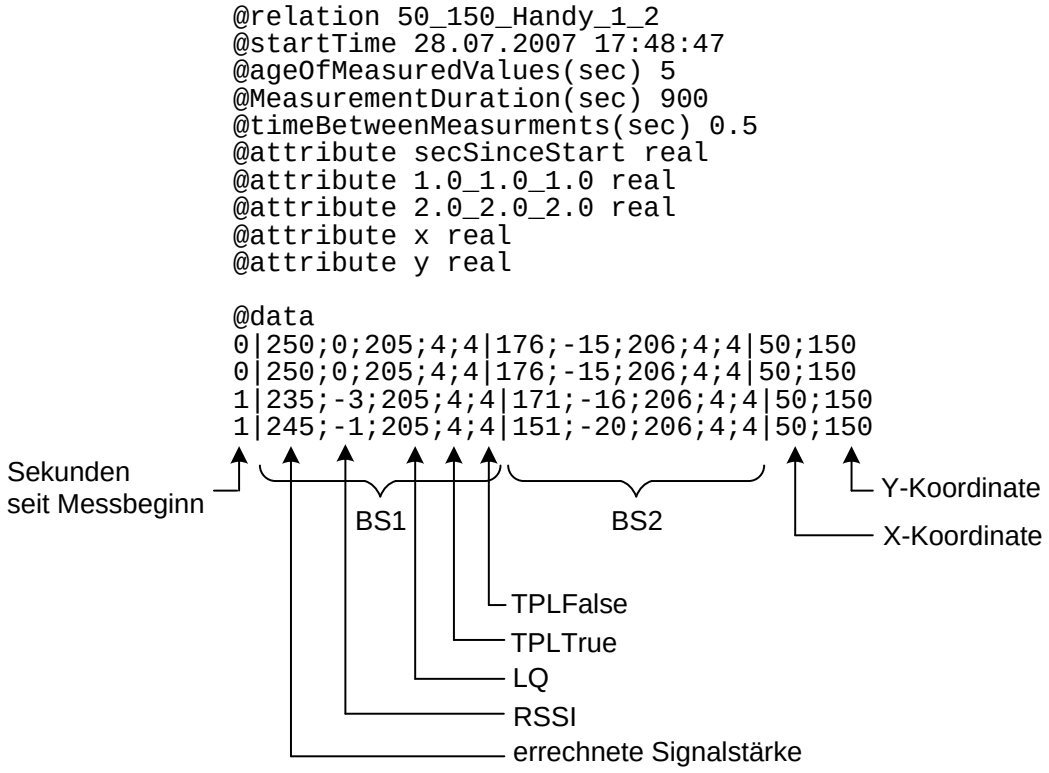


Abbildung 26: Datei mit allen gemessenen Werten

Die geschriebenen Datensätze haben hier folgendes Format: Der erste Eintrag gibt die Sekunden seit dem Beginn der Messung an. Danach folgen die gemessenen Werte der einzelnen Base Stations. Zuerst steht der errechnete Signalstärkewert, danach folgen die Werte RSSI, LQ, TPLTrue und TPLFalse. Die einzelnen Base Stations sind durch das Trennzeichen, das in der Serverkonfigurationsdatei beim Parameter *StationSeperator* angegeben wurde, getrennt. In diesem Beispiel ist das ein „|“. Ist eine registrierte BS nicht mit dem Bluetooth-Gerät verbunden oder sind die gemessenen Werte zu alt, wird für alle Werte dieser Base Station der konfigurierte *NoMeasurmentValue* gespeichert.

# 6 Experimente und Messungen mit Bluetooth

*Mit dem entwickelten Messsystem hat man ein Werkzeug, mit dem man Versuche und Messungen mit Bluetooth-Geräten ausführen kann. Dieses Kapitel widmet sich dem Experiment zur Auswahl eines passenden Bluetooth-Moduls und der Erhebung der Messdaten für die Lokalisierung mit neuronalen Netzen.*

## 6.1 Auswahl eines geeigneten Bluetooth-Moduls

Um die geplanten Versuche zur Lokalisierung durchführen zu können, müssen die Base Stations mit Bluetooth-Modulen ausgestattet werden. Dafür wurden Bluetooth-USB-Adapter, so genannte Bluetooth-Dongles, verwendet. Wie in Kapitel 3.6.2 beschrieben, haben die Hersteller von Bluetooth-Modulen bei der Realisierung der Signalstärkeparameter einiges an Spielraum. Die Signalstärkeparameter sind weiters von der verwendeten Geräteklasse abhängig. Auch in den vorgestellten Arbeiten in Kapitel 2 liest man, dass die Qualität der Signalstärkeparameter von den verwendeten Bluetooth-Modulen abhängig ist. Über die Parameter, welche für die Berechnung der Signalstärke herangezogen werden, ist man sich teilweise uneinig. Werden in 2.1 und 2.2 der RSSI verwendet bzw. miteinbezogen, so wird in 2.3 ausschließlich die LQ verwendet.

Aufgrund dieser Umstände wurden zu Beginn dieser Arbeit Messungen mit Bluetooth-Modulen unterschiedlicher Hersteller und verschiedener Geräteklassen durchgeführt, um festzustellen, welche Geräte und Parameter sich für die Berechnung einer Signalstärke eignen. Die Wahl eines geeigneten Bluetooth-Moduls ist wesentlich, da man nur so qualitativ hochwertige Messwerte erhält und eine genaue Lokalisierung durchführen kann. In dieser Arbeit wird die Position des MDs nicht durch die Distanz zwischen dem MD und den BSs bestimmt, sondern durch die von den BSs gemessenen Werte, die zusammen eine Position eindeutig identifizieren. Es ist daher von Bedeutung, dass das Bluetooth-Modul an den verschiedenen Messpunkten unterschiedliche Werte liefert. Es sollte darüber hinaus empfindlich sein, da eine Lokalisierung mit einer Genauigkeit von 50cm angestrebt wird.

Es wurden drei Bluetooth-Dongles im Hinblick auf die Signalstärkeparameter RSSI, LQ und TPL untersucht. Es handelt sich dabei um handelsübliche Geräte. Sie alle verwenden einen CSR-Chipsatz mit der Bluetooth-Version 2.0 + EDR. In Tabelle 3

ist der Preis jedes Dongles angegeben, um die Kostengünstigkeit des Messsystems zu dokumentieren.

| **Bezeichnung** | **Typ** | **Geräteklasse** | **Preis (inkl. Mwst. 20%)** |
|---|---|---|---|
| D18 | Digitus DN-3018 | 1 | 15,90€ |
| D17 | Digitus DN-3017 | 2 | 12,90€ |
| MSI | MSI BToes 2.0 | 2 | 8,36€ |

Tabelle 3: Bluetooth-Dongles

Als MDs wurden die in Tabelle 4 angegebenen Geräte verwendet. Beide entsprechen der Bluetooth-Version 2.0.

| **Bezeichnung** | **Typ** | **Geräteklasse** |
|---|---|---|
| ednet | Bluetooth-Dongle ednet-87034 | 2 |
| Nokia N70 | Mobiltelefon Nokia N70 | 3 |

Tabelle 4: Mobile Devices

### 6.1.1 Aufbau des Experiments

Eine BS verbindet sich zum MD und misst den RSSI-, LQ- und TPL-Wert der Verbindung einmal pro Sekunde eine Minute lang. Dieser Vorgang wurde dreimal wiederholt, wobei sich das Mobile Device immer an einem anderen Messpunkt befand.

- Messpunkt 1: BS und MD befinden sich im selben Raum. Keine störenden Elemente zwischen den beiden Geräten.

- Messpunkt 2: BS und MD befinden sich in unterschiedlichen Räumen. Es müssen eine geschlossene Tür und ein Kasten durchdrungen werden.

- Messpunkt 3: BS und MD befinden sich in unterschiedlichen Räumen. Es müssen eine Wand und ein Kasten durchdrungen werden.

Dann wurde der gesamte Ablauf mit dem zweiten Mobile Device durchgeführt. Die gemessenen Daten wurden von der BS an den Server übertragen und dort aufgezeichnet. Der ednet-Dongle wurde für die Versuche an einen Laptop angeschlossen. Beide Mobile Devices wurden während der Messungen mit Strom versorgt, um die

Ergebnisse nicht durch einen sinkenden Akkustand zu beeinflussen. Der eben beschriebene Versuch wurde mit allen drei Bluetooth-Dongles ausgeführt.

### 6.1.2 Ergebnis des Experiments

#### *Mobile Device Class 2*

In Abbildung 27 sieht man die Messergebnisse der einzelnen Dongles zum MD der Geräteklasse 2. D18 ist blau, D17 rot und der MSI-Dongle grün dargestellt.

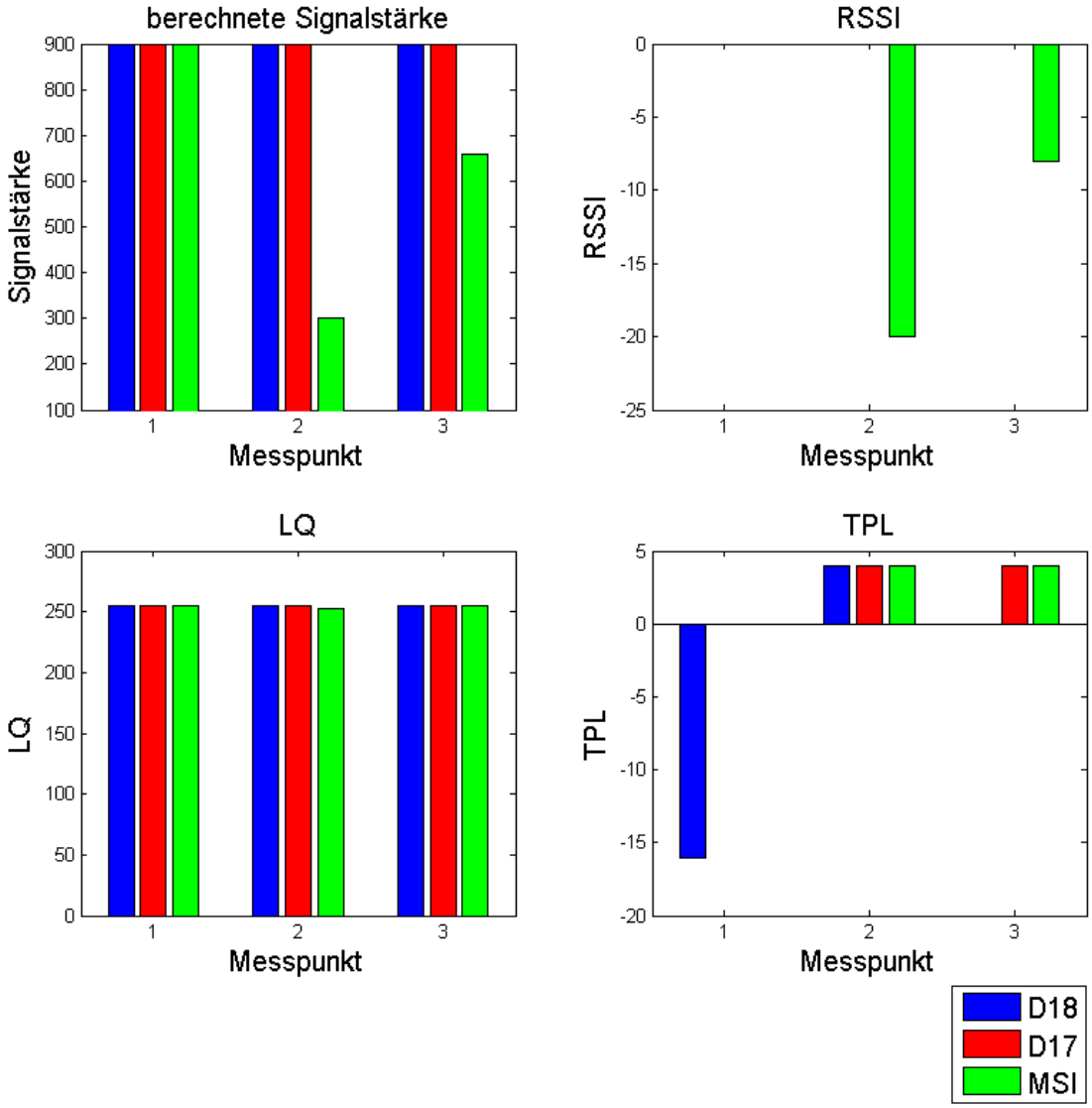


Abbildung 27: Donglevergleich, Messung zu Geräteklasse 2[8]

[8] Die Messdaten, die der Abbildung zugrunde liegen, findet man auf der CD im Ordner *Experimente\DongleAuswahl\Messdaten\Class2*.

Der D18-Dongle empfängt das Funksignal des ednet-Dongles von allen drei Messpunkten ausreichend stark. Der RSSI-Wert ist daher immer 0. Auch die LQ hat den Maximalwert 255 an allen Messpunkten. Nur der TPL-Wert, der die eigene Sendeleistung angibt und sich normalerweise bei 4dBm befindet, muss bei Messpunkt 1 und 3 gesenkt werden. Auch der D17-Dongle erhält an allen Punkten einen RSSI-Wert von 0 und eine LQ von 255. Nur die eigene Sendeleistung muss bei Messpunkt 1 angepasst werden. Unterschiedliche RSSI-Werte an allen drei Punkten kann nur der MSI-Dongle messen. Die LQ liegt bei Messpunkt 2 bei 253, bei den anderen beiden ebenfalls bei 255. Die eigene Sendeleistung muss bei Messpunkt 1 angepasst werden, wo sich die BS und das MD im selben Raum befinden.

### *Mobile Device Class 3*

Zum MD der Geräteklasse 3 und zum Mobiltelefon Nokia N70 konnten die Dongles die in Abbildung 28 abgebildeten Werte messen.

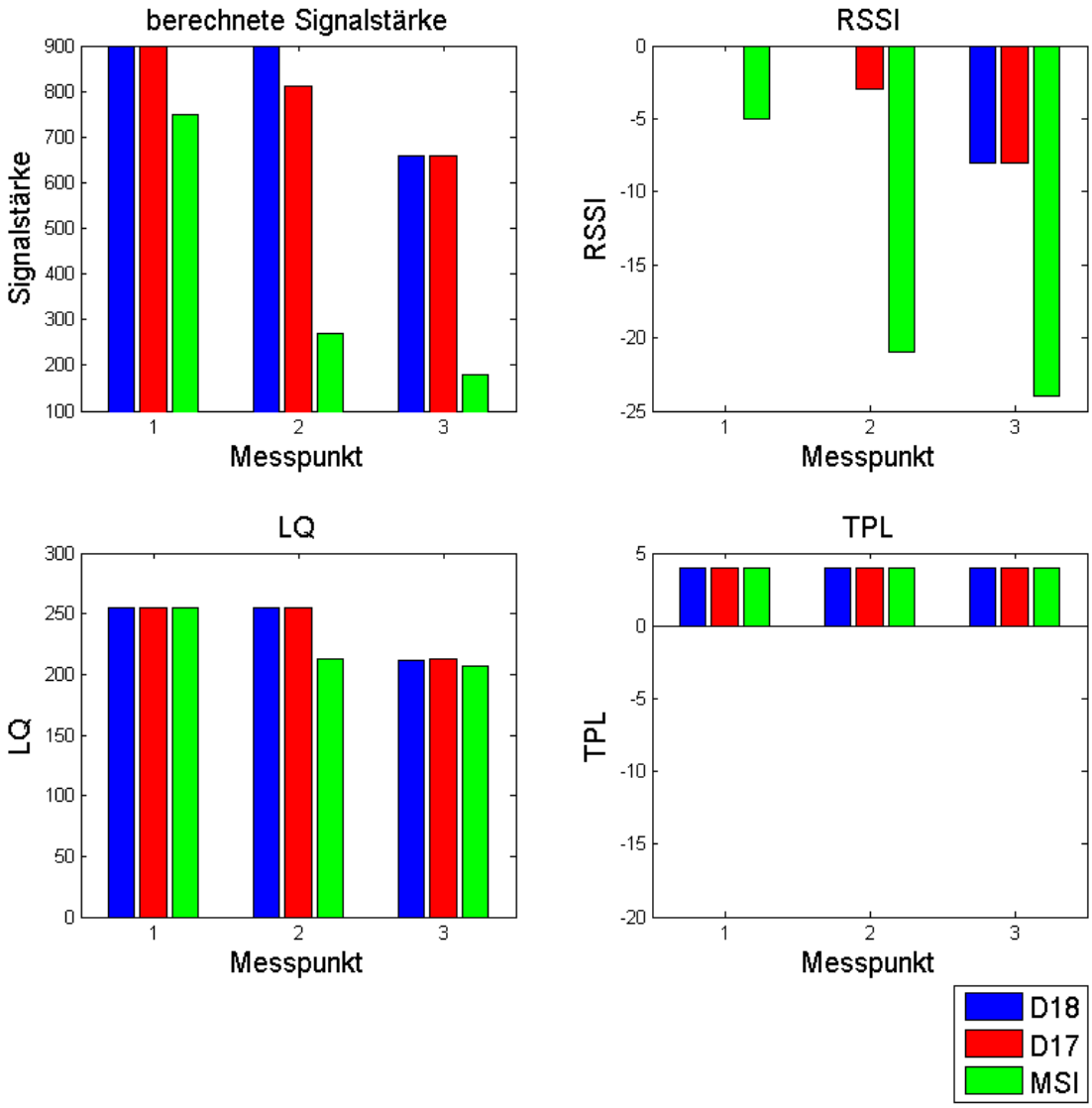


Abbildung 28: Donglevergleich, Messung zu Geräteklasse 3[9]

Der D18-Dongle kann nur an Messpunkt 3 einen abweichenden RSSI- und LQ-Wert messen. Ebenso ist es beim Dongle D17. Weiters misst er noch am Punkt 2 einen RSSI-Wert von -2. Der MSI-Dongle erhält an allen Messpunkten unterschiedliche

[9] Die Messdaten, die der Abbildung zugrunde liegen, findet man auf der CD im Ordner *Experimente\DongleAuswahl\Messdaten\Class3*.

RSSI- und LQ-Werte, wobei sich die LQ-Werte an Messpunkt 2 mit 213 und Messpunkt 3 mit 207 nur minimal unterscheiden. Keiner der drei Dongles musste die eigene Sendeleistung senken, wenn sie mit dem Mobile Device der Geräteklasse 3 verbunden waren.

### *Fazit*

Der D18-Dongle misst unterschiedliche TPL-Werte an den einzelnen Messpunkten zu einem Class 2 Device. Jedoch können mit keinem Signalstärkeparameter die Messpunkte unterschieden werden, wenn er zu einem Class 3 Device verbunden ist. Der Dongle ist daher für die Experimente nicht geeignet.

Beim D17-Dongle ist es umgekehrt. Er misst unterschiedliche RSSI-Werte zu einem Class 3 Device. Es gibt aber keinen Signalstärkeparameter, mit dem man die Messpunkte unterscheiden kann, wenn er zu einem Class 2 Device verbunden ist. Er ist ebenfalls ungeeignet.

Mit dem MSI-Dongle können zu Class 2 und Class 3 Devices unterschiedliche RSSI-Werte an den einzelnen Punkten gemessen werden. Daher kommt dieser Dongle an den Base Stations als Bluetooth-Modul zum Einsatz.

Der in Punkt 5.2.1 vorgestellte Algorithmus zur Berechnung der Signalstärke wurde daher anhand der Messergebnisse des MSI-Dongles entwickelt. Der RSSI-Wert ist in Bezug auf Class 2 und Class 3 Devices aussagekräftig und dient daher als Basis für die Berechnung. Eine Senkung der eigenen Sendeleistung weist eindeutig auf eine kurze Distanz zwischen der Base Station und Mobile Device hin. Deshalb findet der TPL-Wert im Algorithmus ebenfalls Verwendung. Anhand der LQ kann man für Class 2 Devices keinen Rückschluss auf den Messpunkt ziehen. Bei Class 3 Devices würde die berechnete Signalstärke an Aussagekraft verlieren, da die Unterschiede zwischen den Messpunkten geringer würden. Sie wird daher bei der Berechnung nicht berücksichtigt. Die mit diesem Algorithmus berechnete Signalstärke ist in Abbildung 27 und Abbildung 28 ebenfalls dargestellt.

## 6.2 Erhebung der Messdaten zur Lokalisierung

Mit dem MSI-Dongle als Bluetooth-Modul und dem erstellten Messsystem können nun die Daten für die Lokalisierung mit neuronalen Netzen durch ein Experiment erhoben werden.

### 6.2.1 Aufbau des Experiments

Das Experiment wurde in einer typischen Altbauwohnung durchgeführt. Im Wohnzimmer wurde ein quadratisches Messfeld mit einer Seitenlänge von 3m abgesteckt. Im Abstand von 50cm wurden Messpunkte am Boden markiert. In Abbildung 29 wird das Messfeld durch die rot strichlierten Linien dargestellt. Die Messpunkte befinden sich an den Schnittpunkten der Linien. Der erste Messpunkt mit den Koordinaten 0,0 befindet sich in der linken oberen Ecke. Der 49te Messpunkt mit den Koordinaten 300,300 befindet sich in der rechten unteren Ecke.

Der Rechner, der das Programm BTLocationServer ausführte, wurde in der Küche aufgestellt. Er war über ein LAN mit den vier Base Stations verbunden. Jede von ihnen war mit einem MSI-Dongle ausgestattet und führte das Programm BaseStation aus. BS 1 wurde im Wohnzimmer links unten neben dem Messfeld positioniert. Im Vorraum wurde BS 2 aufgestellt. BS 3 befand sich im Wohnzimmer rechts oben neben dem Messfeld und BS 4 wurde im Schlafzimmer positioniert.

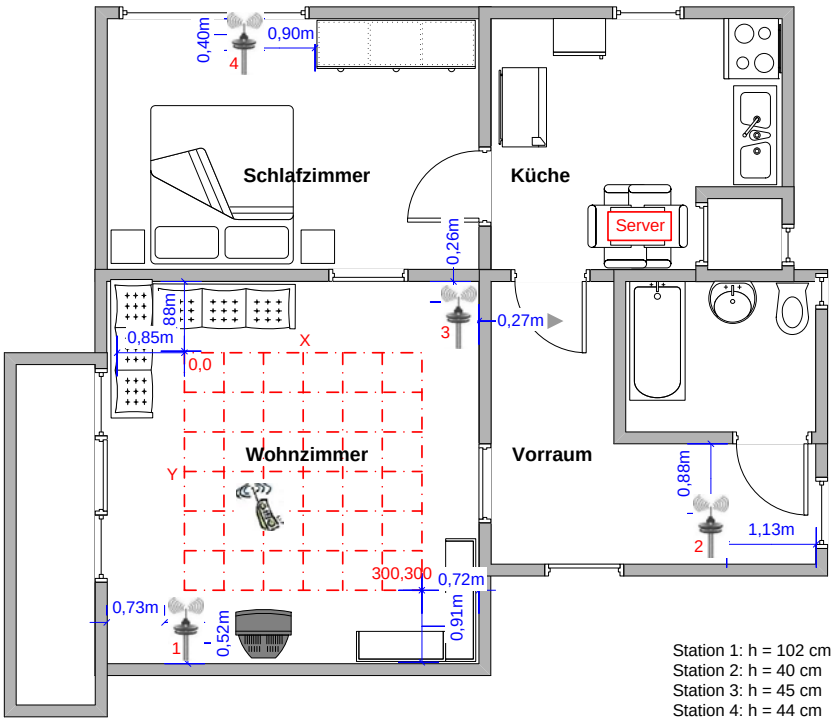


Abbildung 29: Experimentaufbau

Die genauen Positionen des Messfeldes und der BSs kann man Abbildung 29 entnehmen. Die Abmessungen zu einer BS beziehen sich auf den angeschlossenen Bluetooth-Dongle.

### 6.2.2 Durchführung

Mit diesem Aufbau wurden nun Messungen zu den in Tabelle 5 angeführten Mobile Devices durchgeführt.

| **Bezeichnung** | **Typ** | **Geräteklasse** |
|---|---|---|
| D17 | Bluetooth-Dongle Digitus DN-3017 | 2 |
| Nokia N70 | Mobiltelefon Nokia N70 | 3 |

Tabelle 5: Mobile Devices

Das linke Bild in Abbildung 30 wurde während des Experimentdurchlaufs mit dem MD der Geräteklasse 2 gemacht. Das MD der Geräteklasse 3 zeigt das rechte Bild. Wie auf den Fotos zu sehen ist, wurde das MD mittig auf einem Stativ positioniert. Das Stativ wurde mit Hilfe eines Pendels über dem Messpunkt ausgerichtet. Die BSs, die alle im Modus Measurement liefen, stellten dann eine Verbindung zum MD her. Sie ermittelten die Signalstärkeparameter der Verbindung und sendeten sie an den BTLocationServer, der diese aufzeichnete. An jedem der 49 Messpunkte wurden 15 Minuten lang im Abstand von 0,5 Sekunden die Signalstärkeparameter abgetastet. Das MD wurde während der Messungen mit Strom versorgt, um die Werte nicht durch einen sinkenden Akkustand zu beeinflussen. Die Türen zum Wohnzimmer waren während der Messung geschlossen.

Abbildung 30: Fotos vom Experiment

Die vom BTLocationServer angelegten Dateien mit Messwerten wurden mit dem Programm SearchFiles auf Korrektheit überprüft. Wenn es bei einer Aufzeichnung an einem Messpunkt zu einer längeren Unterbrechung der Verbindung zwischen einer BS und dem MD kam, wurde die Messung für diesen Punkt wiederholt.

Da die JTooth Bibliothek keinen Role Switch unterstützt, mussten die BSs die Rolle des Masters einnehmen, da sie die Verbindung initiierten. Aus diesem Grund konnten sich nur zwei BSs gleichzeitig mit dem MD verbinden. Der Verbindungsaufbau einer weiteren BS wurde vom MD zurückgewiesen. So musste an jedem Punkt zweimal gemessen werden. Zuerst waren die BSs 1 und 2 mit dem MD verbunden, dann 3 und 4. Auf diese Weise entstanden pro Messpunkt vier Dateien. Zwei Dateien, die nur den errechneten Signalstärkewert der BSs enthalten und zwei, in denen alle gemessenen Signalstärkeparameter gespeichert sind. Diese Dateien wurden mit dem Programm StationMerge zusammengeführt, sodass pro Messpunkt eine Datei mit den errechneten Signalstärkewerten und eine mit allen gemessenen Signalstärkeparametern existierten.

### 6.2.3 Ergebnisse

*In den Abbildungen zur Präsentation der Ergebnisse werden die Messpunkte durch den Median der Signalstärkewerte, die bei ihnen gemessen wurden, dargestellt.*

#### *Mobile Device Class 2*

In Abbildung 31 sieht man die Signalstärkewerte, die die einzelnen BSs zum MD, dem D17-Dongle, messen konnten. Die BS-Icons neben den Diagrammen geben die Position der jeweiligen BS in Relation zum Messfeld an.

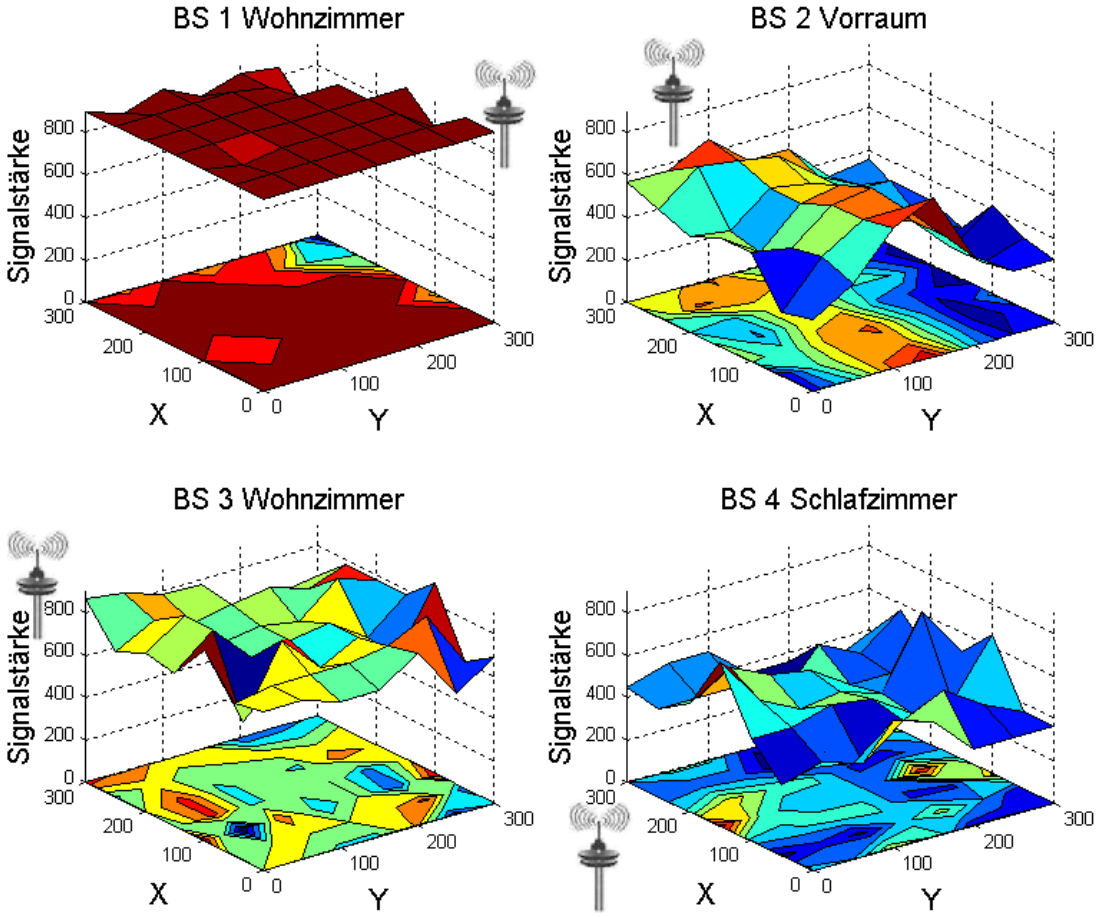


Abbildung 31: Messfeld Geräteklasse 2[10]

BS 1 die unmittelbar neben dem Messfeld positioniert ist, misst sehr hohe Signalstärken. Bei der Projektion des Messfeldes auf die XY-Ebene lässt sich ein kreisförmiger Verlauf erkennen. Dieser Effekt lässt sich durch das Antennendiagramm des Bluetooth-Moduls erklären (siehe Kapitel 3.2.1). Da der MSI-Dongle aufgrund des USB-Steckplatzes an der BS 1 senkrecht angebracht wurde, befindet sich der Messpunkt 300,300 sozusagen „über“ dem Dongle in einem Funkloch. Es wird daher an Messpunkten in diesem Bereich eine deutlich schwächere Signalstärke gemessen.

Der Einfluss, den die Umgebung auf die Messwerte einer BS hat, lässt sich gut am Diagramm für BS 2 erkennen. Die Verbindung zwischen dem MD und BS 2 wird durch eine Tür bzw. eine Mauer beeinträchtigt. An den Messpunkten, die sich auf einer Linie mit der Tür und der BS befinden, ist die Signalstärke deutlich höher. Muss jedoch die Mauer durchdrungen werden, fällt die Signalstärke klar ab. Der in

[10] Die zugrunde liegenden Daten findet man auf der CD im Ordner *Experimente\Lokalisierung\Class2\Messdaten*.

Kapitel 3.2.1 beschriebene Effekt, dass die Dämpfung des Signals unter anderem von der Dicke und dem Material des Hindernisses abhängt, kann hier beobachtet werden.

Mit einer durchschnittlichen Signalstärke von 750 liefert BS 3 durchaus hohe Werte, aber doch geringer als die von BS 1, die bei 894 lagen. Beide BSs befanden sich im Wohnzimmer und wurden durch keine Hindernisse beeinträchtigt. Der einzige Unterschied bestand in der Höhe, in der die BSs positioniert waren. BS 1 befand sich in einer Höhe von 102cm und BS 3 in 45cm Höhe. BS 3 war somit niedriger als die Auflagefläche des Stativs. Diese befand sich in einer Höhe von 85cm und musste sie daher durchdringen. Das könnte der Grund für die niedrigeren Messwerte sein.

Bei BS 4, die im Schlafzimmer aufgestellt war, lässt sich derselbe Effekt wie bei BS 2 beobachten, jedoch nicht ganz so klar. Die Mauer dämpft das Funksignal stärker als die Tür.

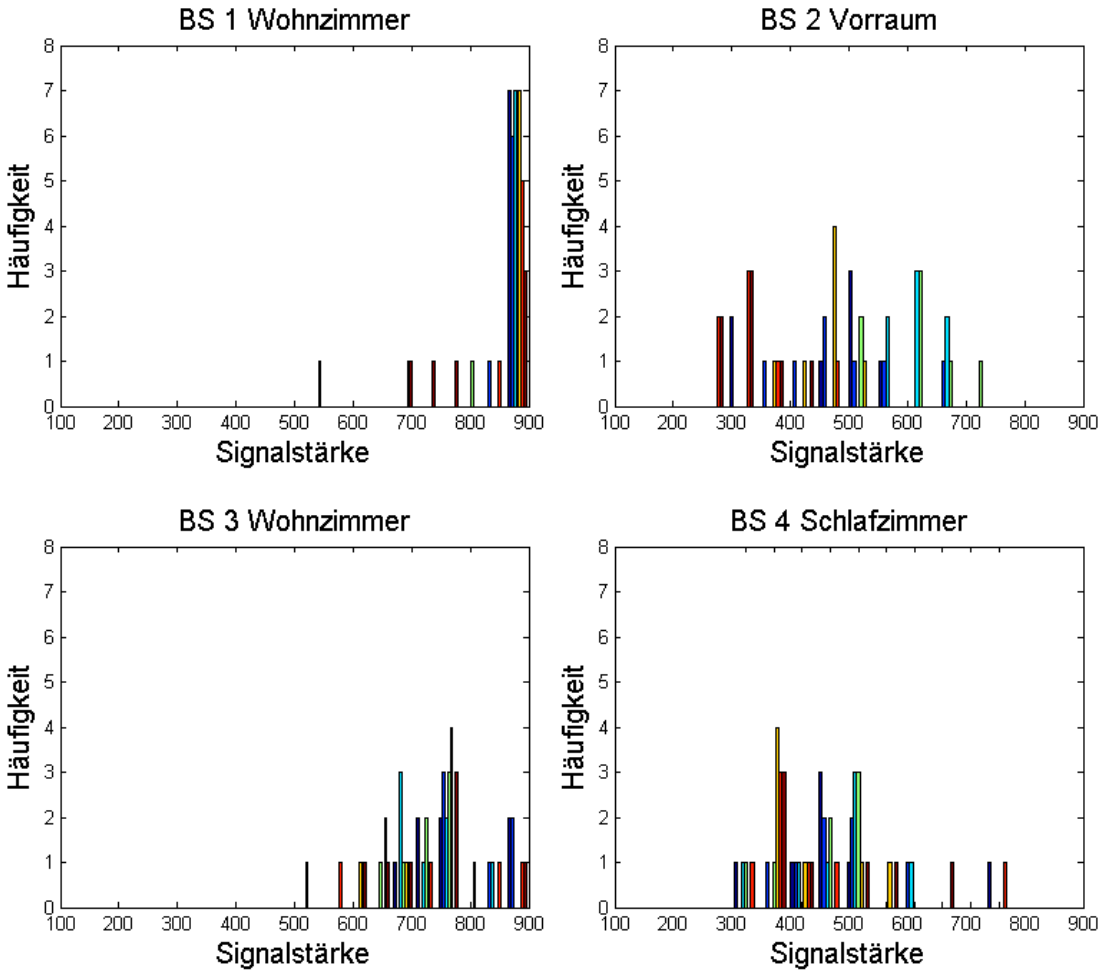


Abbildung 32: Messfeld Geräteklasse 2, Histogramm[11]

In Abbildung 32 sind die Signalstärkewerte, welche die einzelnen BSs zum MD, der Geräteklasse 2, messen konnten, als Histogramm dargestellt. Man erkennt, dass alle BSs, mit Ausnahme von BS 1, für die meisten Messpunkte unterschiedliche Signalstärkewerte erhalten.

[11] Die zugrunde liegenden Daten findet man auf der CD im Ordner *Experimente\Lokalisierung\Class2\Messdaten*.

### *Mobile Device Class 3*

Bei dem Versuchsdurchlauf, in dem das Nokia N70 als MD genutzt wurde, konnten die BSs die in Abbildung 33 dargestellten Signalstärkewerte messen.

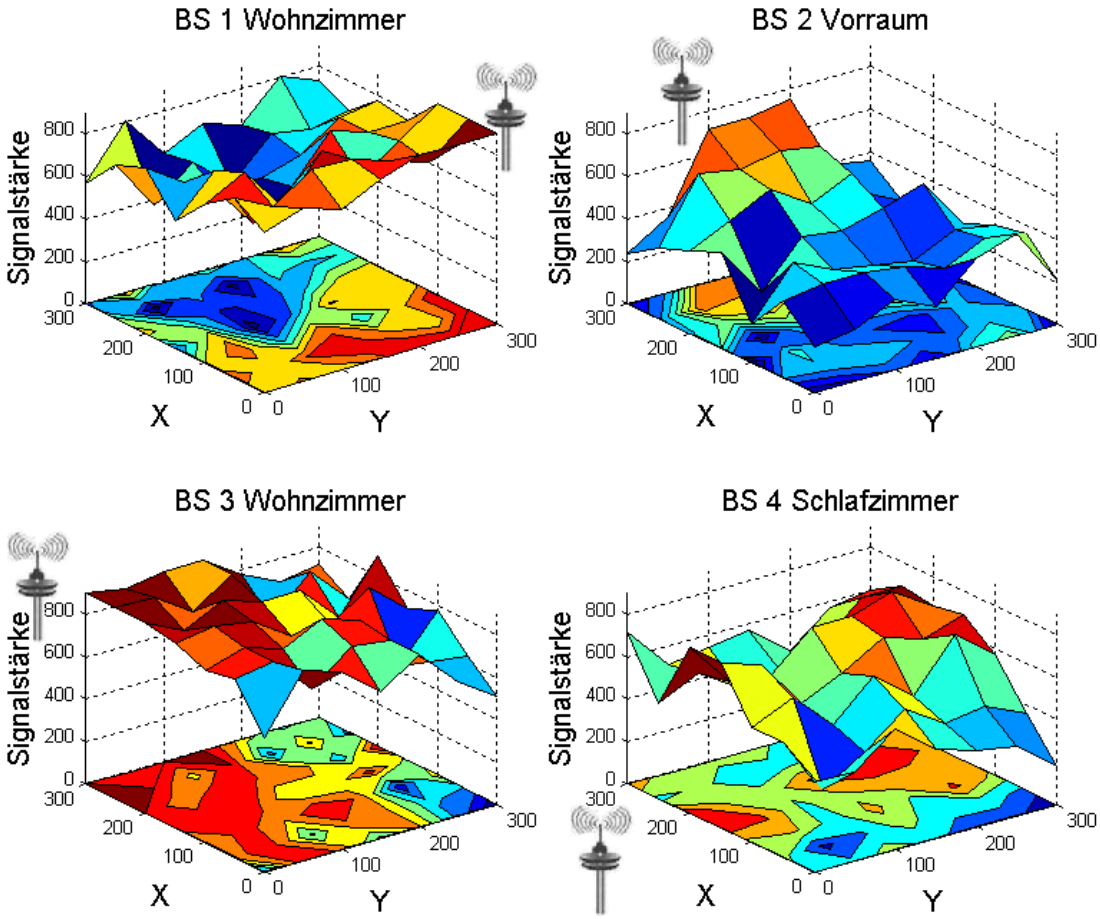


Abbildung 33: Messfeld Geräteklasse 3[12]

Bei den BS 2 und 4 sind die Messergebnisse ähnlich zu denen der Geräteklasse 2. Sie kommen durch die Umgebung der BSs zustande.

Die Ergebnisse der BS 1 und 3 unterscheiden sich von denen der Geräteklasse 2. Der Effekt, hervorgerufen durch das Antennendiagramm, ist bei BS 1 nicht mehr zu erkennen. Man kann bei BS 1 und 3 sehen, dass das Signal besser ist, je näher sich das MD an der BS befindet, und mit zunehmender Distanz schwächer wird. Die Entfer-

[12] Die zugrunde liegenden Daten findet man auf der CD im Ordner *Experimente\Lokalisierung\Class3\Messdaten*.

nung zwischen BS und MD spielt bei Blutooth-Geräten der Geräteklasse 3, aufgrund der geringeren Sendeleistung, eine weit größere Rolle.

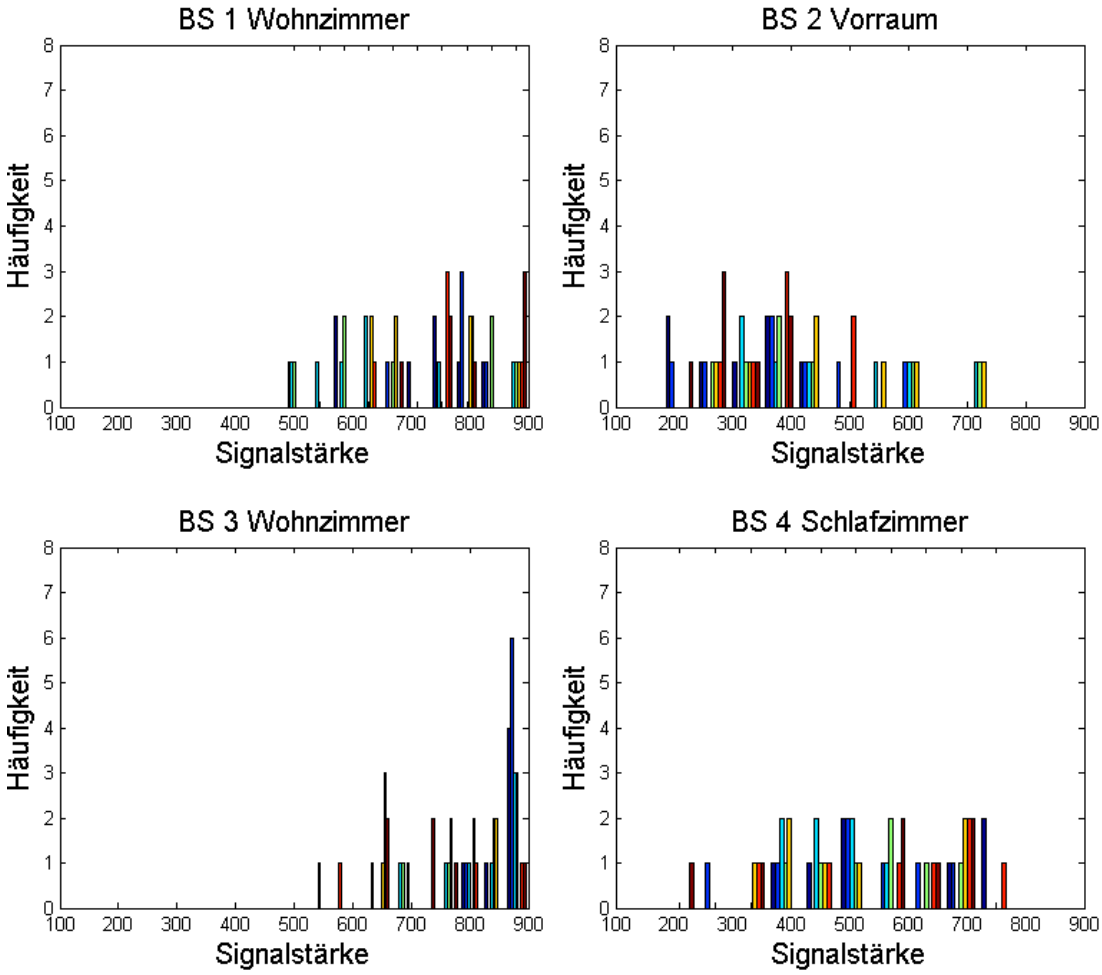


Abbildung 34: Messfeld Geräteklasse 3, Histogramm[13]

In Abbildung 34 sind die Signalstärkewerte, welche die einzelnen BSs zum MD, der Geräteklasse 3, messen konnten, als Histogramm dargestellt. Auch hier erhalten die BSs für die meisten Messpunkte unterschiedliche Signalstärkewerte. Durch die geringere Sendeleistung des MDs kann der Wertebereich der Signalstärke besser ausgenutzt werden. Man sieht dies bei BS 2 und 3, bei denen teilweise sehr niedrige Signalstärkewerte gemessen wurden.

[13] Die zugrunde liegenden Daten findet man auf der CD im Ordner *Experimente\Lokalisierung\Class3\Messdaten*.

# 7 Lokalisierung mit künstlichen neuronalen Netzen

*Dieses Kapitel beschreibt die Experimente, die mit neuronalen Netzen durchgeführt wurden. Es erklärt, wie die Daten, die durch das in Kapitel 6.2 beschriebene Experiment weiterverarbeitet wurden, um dann für das Training und Testen der neuronalen Netze verwendet werden zu können. Weiters wird eine Analyse der trainierten neuronalen Netze präsentiert und die Frage beantwortet, wie gut diese in der Lage sind, MDs zu lokalisieren. Für die Weiterverarbeitung der Daten, das Training, sowie die Analyse der neuronalen Netze wurde das Programm Matlab bzw. die Neural Network Toolbox von Matlab verwendet. Ab diesem Punkt der Arbeit wird nur mehr der berechnete Signalstärkewert verwendet.*

## 7.1 Vorbereitende Arbeiten

Bevor mit dem Training der künstlichen neuronalen Netze begonnen wird, sind noch einige Arbeitsschritte notwendig.

### 7.1.1 Vorbereiten der Datensätze

Im Rahmen des in Kapitel 6.2 beschriebenen Experimentes wurden für beide MDs an 49 Messpunkten die Signalstärkewerte gemessen. Es liegt somit für jedes MD eine Datenmenge in Form der Rohdaten vor. Bei den Rohdaten ist zu beobachten, dass die Signalstärkewerte mit zunehmender Entfernung zwischen BS und MD zur Fluktuation neigen. Als Beispiel dafür sind in Abbildung 35 die Signalstärkewerte, welche die BSs am Messpunkt 100,100 zu dem MD der Geräteklasse 3 messen konnten, abgebildet.

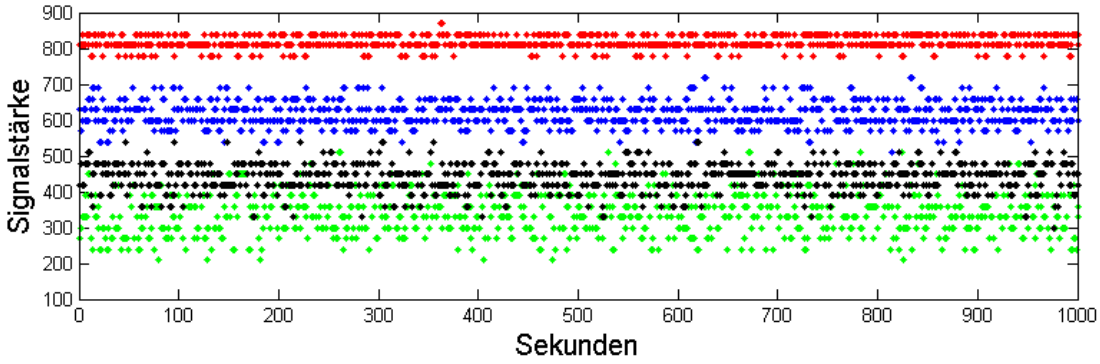


Abbildung 35: Signalstärkewerte am Messpunkt 100,100

Die von BS 1 gemessenen Signalstärkewerte sind relativ konstant, da das MD und die BS nicht weit von einander entfernt sind und auch keine Hindernisse durchdrungen werden müssen. Die von BS 4 gemessenen Werte fluktuieren sehr stark, da die Entfernung deutlich größer ist und auch eine Wand durchdrungen werden muss. Um eine Datenmenge mit eindeutigeren Werten zu erhalten, wurden die gemessenen Signalstärkewerte mit der Formel:

$$y_{i+1} = \alpha * \tilde{y}_i + (1-\alpha) * y_i, y_1 = \tilde{y}_1$$

geglättet. Die Variable α gibt dabei an, wie stark die Messwerte geglättet werden sollen. Je geringer der Wert ist, der für α gewählt wird, umso stärker werden die Werte geglättet. In Abbildung 36 sieht man die Signalstärkewerte, nachdem sie mit einem α von 0,05 geglättet wurden. Die Legende in Abbildung 36 gilt ebenfalls für Abbildung 35.

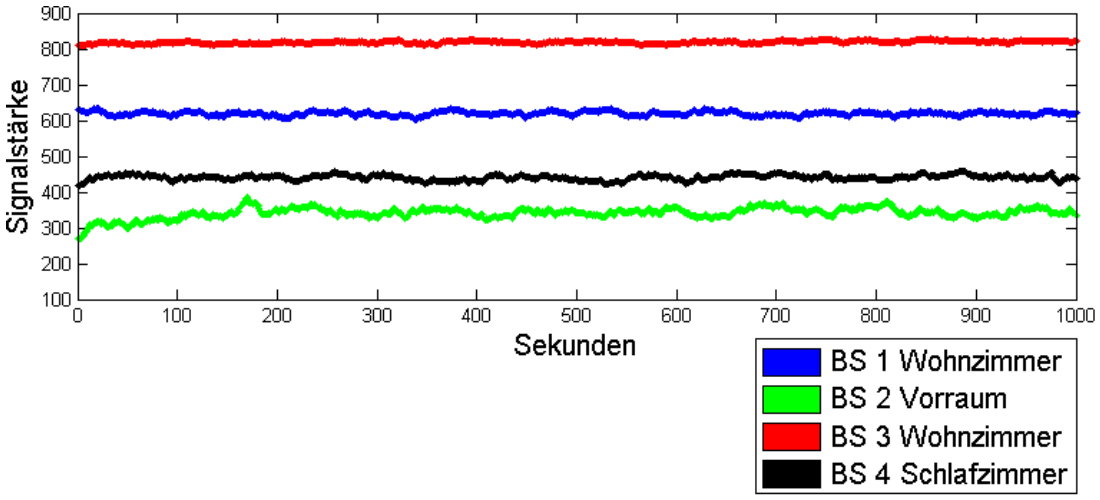


Abbildung 36: geglättete Signalstärkewerte am Messpunkt 100,100

Mit dem Matlab-Skript *glaetten.m* (siehe Kapitel 9.3.1) wurden alle Signalstärkewerte, die an den 49 Messpunkten für die beiden MDs genommen wurden, geglättet. Es liegen nun für jedes MD zwei komplette Datenmengen vor. Einmal in Form der Rohdaten und einmal als geglättete Werte.

### 7.1.2 Auswahl der Datensätze

Im nächsten Schritt wurden aus den vier Datenmengen jeweils eine Trainings-, Test- und Validationsmenge ausgewählt. Dafür wurde das Javaprogramm *LearnFileMaker* (siehe Kapitel 9.2.3) erstellt. Als Inputparameter erhält das Programm den Ordner, der die gewünschte Datenmenge enthält. Eine Datenmenge besteht aus 49 Dateien, für jeden Messpunkt eine, die die Signalstärkewerte beinhalten. Das Programm wählt dann von jedem Messpunkt die gleiche Anzahl an Datensätzen aus. Die Auswahl der Datensätze pro Messpunkt erfolgt zufällig. Dabei wird darauf geachtet, dass keine gleichen Datensätze für einen Messpunkt ausgewählt werden, um so die Generalisierung des KNNs zu fördern. Die so ausgewählten Datensätze werden dann nach einem Prozentschlüssel in eine Trainings-, Test- und Validationsmenge aufgeteilt. Jede dieser Teilmengen enthält für jeden Messpunkt die gleiche Anzahl an Datensätzen, um anschließend repräsentative Aussagen treffen zu können.

Für das Trainieren und Testen der im Anschluss beschriebenen KNNs wurden pro Messpunkt 67 Datensätze ausgewählt. 60% davon wurden als Trainingsmenge, 20% als Testmenge und 20% als Validationsmenge verwendet.

### 7.1.3 Skalieren der In- und Outputdaten

Wie im Theorieteil zu KNNs beschrieben, sollten die In- und Outputdaten in einem Wertebereich von 0,1-0,9 liegen. Der Wertebereich für die Signalstärke wurde deshalb zwischen 100-900 festgelegt und kann nun einfach mit einer Division durch 1000 in den gewünschten Wertebereich umskaliert werden. Die Koordinaten der Messpunkte, die als Outputwerte verwendet werden, liegen im Bereich von 0-300. Sie werden ebenfalls in den Bereich von 0,1-0,9 skaliert. Das Skalieren der In- und Outputdaten wird ebenfalls vom Programm *LearnFileMaker* vorgenommen.

## 7.2 Trainieren der KNNs

Für das Trainieren und Auswerten der KNNs wurde die Neural Network Toolbox (Version 4.0.1) von Matlab (Version 7.0.0.19920(R14)) verwendet. Es wurden zahlreiche Versuche mit neuronalen Netzen durchgeführt. Dabei wurde die Anzahl der Hidden Layer sowie die der Neuronen in diesen Layern variiert. Weiters wurden verschiedene Lernalgorithmen erprobt. Die besten Ergebnisse konnten mit einem fully connected Netz der Größe 4-60-30-2 erzielt werden. Als Lernalgorithmus stellte sich der Bayesian Regularization Backpropagation Algorithmus als geeignet heraus.

Das Training der Netze wurde zuerst mit den Trainings-, Test- und Validationsset durchgeführt. Dabei wird das Training beim ersten Anstieg des SSE für das Validationsset automatisch abgebrochen, um die Generalisierung des Netzes zu gewährleisten. In weiteren Versuchen, ohne Validationsset, wurde festgestellt, dass der SSE für nicht trainierte Daten, sprich für die Testdaten, nach einem kurzen Anstieg zu Beginn im weiteren Trainingsverlauf auf einen weit niedrigeren Wert gebracht werden kann. Deshalb wurden für jedes Netz zwei Trainingsdurchläufe gemacht. Im ersten wurde festgestellt, in welchem Bereich das absolute Minimum des SSE für das Testset liegt. Im zweiten Durchlauf wurde das KNN dann in diesem Bereich händisch gestoppt. So wird das optimale Minimum des SSE für nicht trainierte Daten erreicht und eine maximale Generalisierung gewährleistet. Auf das Validationsset wurde daher ganz verzichtet. Die vier KNNs, die nun präsentiert werden, wurden auf diese Weise trainiert. In den Lerngraphen sieht man daher nur den Verlauf für die Trainings- und Testdaten. Für jede Geräteklasse wurden zwei KNNs trainiert, einmal mit den Rohdaten und einmal mit den geglätteten Daten.

### 7.2.1 KNNs für Mobile Device Class 2

In Abbildung 37 sieht man die Lernkurven der beiden KNNs, die für das MD der Geräteklasse 2, dem D17-Dongle, trainiert wurden. Links die Lernkurve des KNN, das mit den geglätteten Daten trainiert wurde und rechts das Netz, das anhand der Rohdaten lernte. Der Lernverlauf für die Trainingsdaten wird blau, der für die Testdaten rot dargestellt.

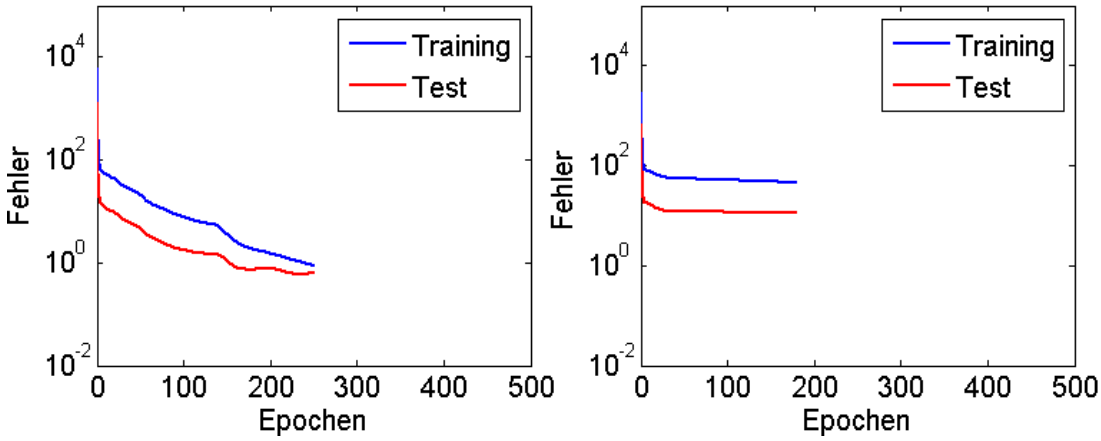


Abbildung 37: Lernkurven KNNs Geräteklasse 2 (links KNN mit geglätteten Daten trainiert, rechts KNN mit Rohdaten trainiert)[14]

Man sieht, dass die geglätteten Daten besser von einem KNN gelernt werden können, da sie weniger widersprüchliche Informationen enthalten. Das KNN, das mit den Rohdaten trainiert wurde, kann bereits nach 181 Lernepochen keine Verbesserungen mehr erzielen. Die genauen Ergebnisse des Lernvorgangs sind in Tabelle 6 festgehalten.

| | **Geglättete Daten** | **Rohdaten** |
|---|---|---|
| **Epochen** | 251 | 181 |
| **SSE-Trainingsset** | 0,863 | 44,433 |
| **SSE-Testset** | 0,593 | 10,973 |

Tabelle 6: KNNs für Mobile Device Class 2

[14] Die Trainings- und Testsets sowie die trainierten KNNs findet man auf der CD im Ordner *Experimente\neuronaleNezte\Class2*

### 7.2.2 KNNs für Mobile Device Class 3

Für das MD der Geräteklasse 3, das Nokia-N70, wurden ebenfalls zwei KNNs trainiert. In Abbildung 38 sind die Lernkurven dargestellt. Das KNN, das die geglätteten Daten lernte, ist links abgebildet, rechts das KNN, welches mit den Rohdaten trainiert wurde. Der Lernverlauf für die Trainingsdaten wird blau, der für die Testdaten rot dargestellt.

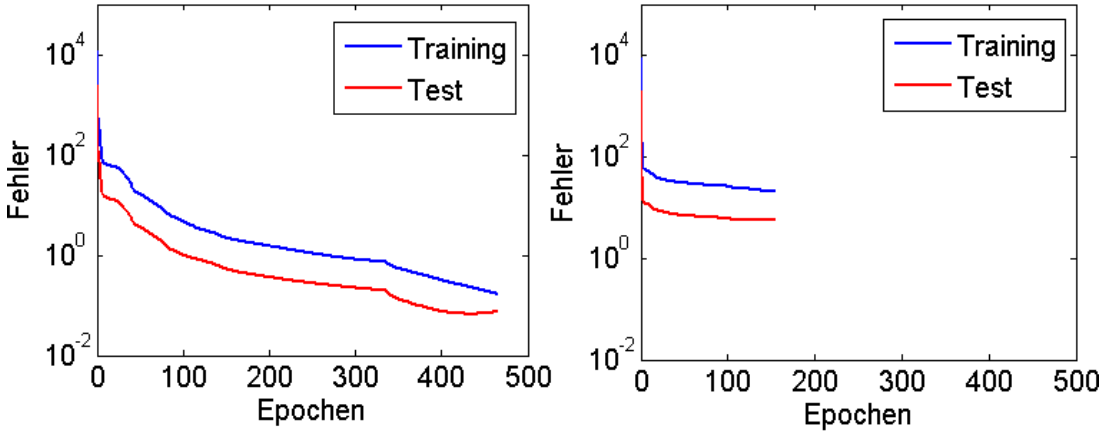


Abbildung 38: Lernkurven KNNs Geräteklasse 3 (links KNN mit geglätteten Daten trainiert, rechts KNN mit Rohdaten trainiert)[15]

Auch beim MD der Klasse 3 sieht man, dass die geglätteten Daten besser gelernt werden können. Es lernt mit 466 Lernepochen am längsten von allen vier KNNs und erzielt auch die besten Ergebnisse für das Trainings- und Testset. In Tabelle 7 sind die genauen Ergebnisse für die KNNs eingetragen.

| | **Geglättete Daten** | **Rohdaten** |
|---|---|---|
| **Epochen** | 466 | 156 |
| **SSE-Trainingsset** | 0,166 | 20,096 |
| **SSE-Testset** | 0,066 | 5,510 |

Tabelle 7: KNNs für Mobile Device Class 3

[15] Die Trainings- und Testsets sowie die trainierten KNNs findet man auf der CD im Ordner *Experimente\neuronaleNetze\Class3*

Bei den KNNs für die Geräteklasse 3 können bei beiden Netzen deutlich bessere Ergebnisse erzielt werden als bei jenen der Geräteklasse 2. Das liegt daran, dass die BSs zu dem MD der Geräteklasse 3 unterschiedlichere Signalstärkewerte messen und es so zu weniger Mehrdeutigkeiten bezüglich der einzelnen Messpunkte kommt.

## 7.3 Experimente zur Lokalisierung mit KNNs

Mit den trainierten KNNs wurden nun abschließende Versuche durchgeführt. Dabei wurde überprüft, ob nun die Position eines MD mit einem KNN auf 50cm genau bestimmt werden kann. Weiters wurde untersucht wie gut sich Datensätze aus der Rohdatenmenge mit dem KNN positionieren lassen, das mit den geglätteten Daten trainiert wurde und umgekehrt. Dieser Versuch wurde wieder für beide Geräteklassen durchgeführt. Dieser Punkt widmet sich nun der Präsentation der Ergebnisse. Die Ausgabedaten wurden für die Darstellung vom Bereich 0,1-0,9 wieder in cm rückskaliert. KNNs, die mit geglätteten Daten trainiert wurden, werden auch als KNN-glatt bezeichnet, jene, die mit Rohdaten lernten, als KNN-roh.

### 7.3.1 Mobile Device Class 2

#### *KNN mit geglätteten Daten*

Zuerst wird das KNN, welches mit den geglätteten Daten für das MD der Geräteklasse 2 trainiert wurde, präsentiert.

#### *Testset mit geglätteten Daten*

Wird die Lokalisierung für das Testset mit geglätteten Daten, mit diesem KNN durchgeführt, sieht man in Abbildung 39, dass dies sehr gut funktioniert.

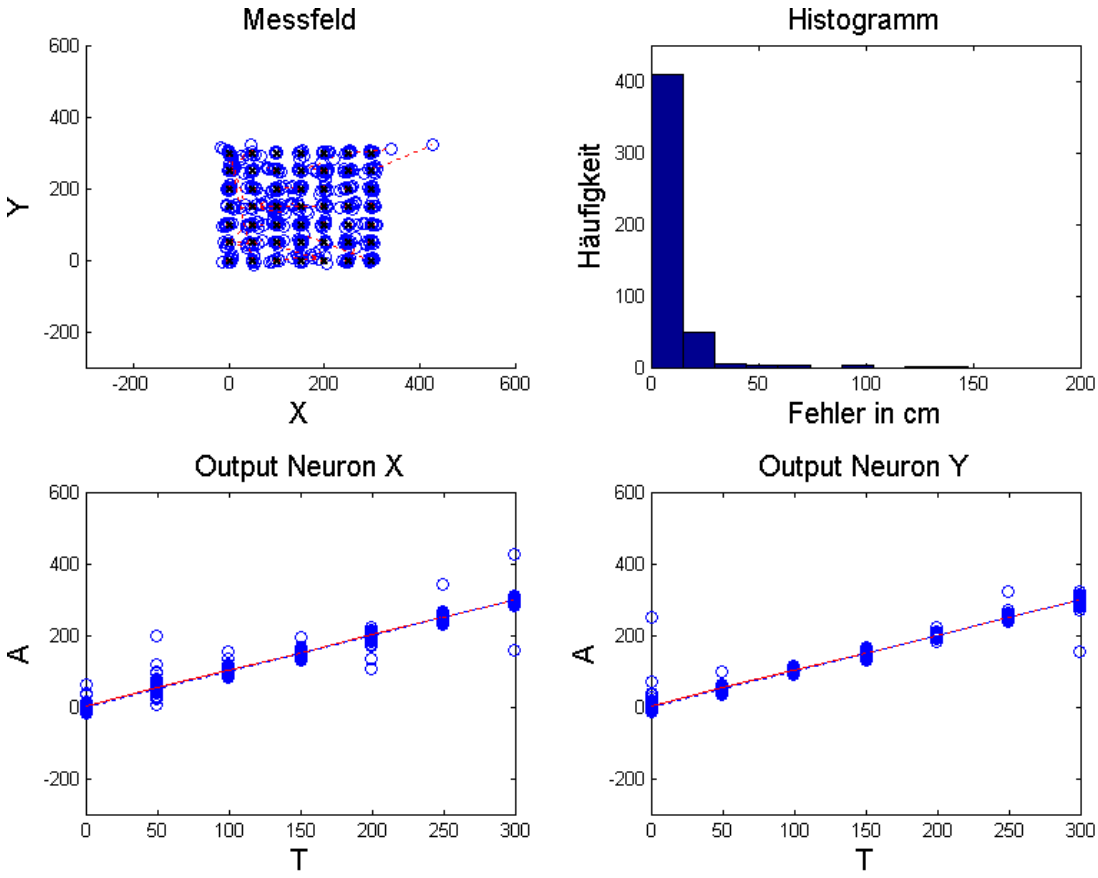


Abbildung 39: Analyse KNN-glatt für Geräteklasse 2 mit geglätteten Testdaten[16]

Das erste Diagramm links oben zeigt das Messfeld mit den 49 Messpunkten. Ein Messpunkt wird durch ein schwarzes X dargestellt. Jeder Testdatensatz, dessen Koordinaten auf dem Messfeld durch das KNN bestimmt wurden, wird durch einen blauen Ring dargestellt. Ein Testdatensatz (blauer Ring) wird mit dem Messpunkt (schwarzes X) an dem er genommen wurde, durch eine strichlierte rote Linie verbunden. Diese ist natürlich nur dann sichtbar, wenn der Testdatensatz durch das KNN falsch lokalisiert wurde. Wie man aber in Abbildung 39 sieht, sind die 49 Messpunkte deutlich erkennbar und nur wenige rote Linien sichtbar. Auch das Histogramm zeigt, dass beinahe alle der 470 Testdatensätze perfekt mit dem KNN lokalisiert werden konnten. Der durchschnittliche Fehler für die Lokalisierung liegt bei 9,6cm, der Median bei 5,4cm. Auch die Regressionsanalysen für die beiden Outputneuronen des KNN liefern gute Ergebnisse. Die Steigungen (m) der approximierten Geraden liegen für das Outputneuron der X-Koordinate bei 0,986, bei der Y-Koordinate bei 0,980. Auch der Korrelationskoeffizient (r) ist bei beiden Koordina-

[16] Das Testset findet man auf der CD im Ordner *Experimente\neuronaleNetze\Class2\Laptop 0-05 67-60-20-20\Trainingsdaten*

ten sehr gut. Für X liegt r bei 0,986 für Y bei 0,988. Die Lokalisierung eines Class 2 MD mit einem KNN funktioniert mit geglätteten Daten sehr gut.

### *Testset mit Rohdaten*

Lokalisiert man mit diesem Netz, das mit geglätteten Daten trainiert wurde, nun die Datensätze aus dem Testset der Rohdaten, führt das zu dem in Abbildung 40 dargestellten Ergebnis.

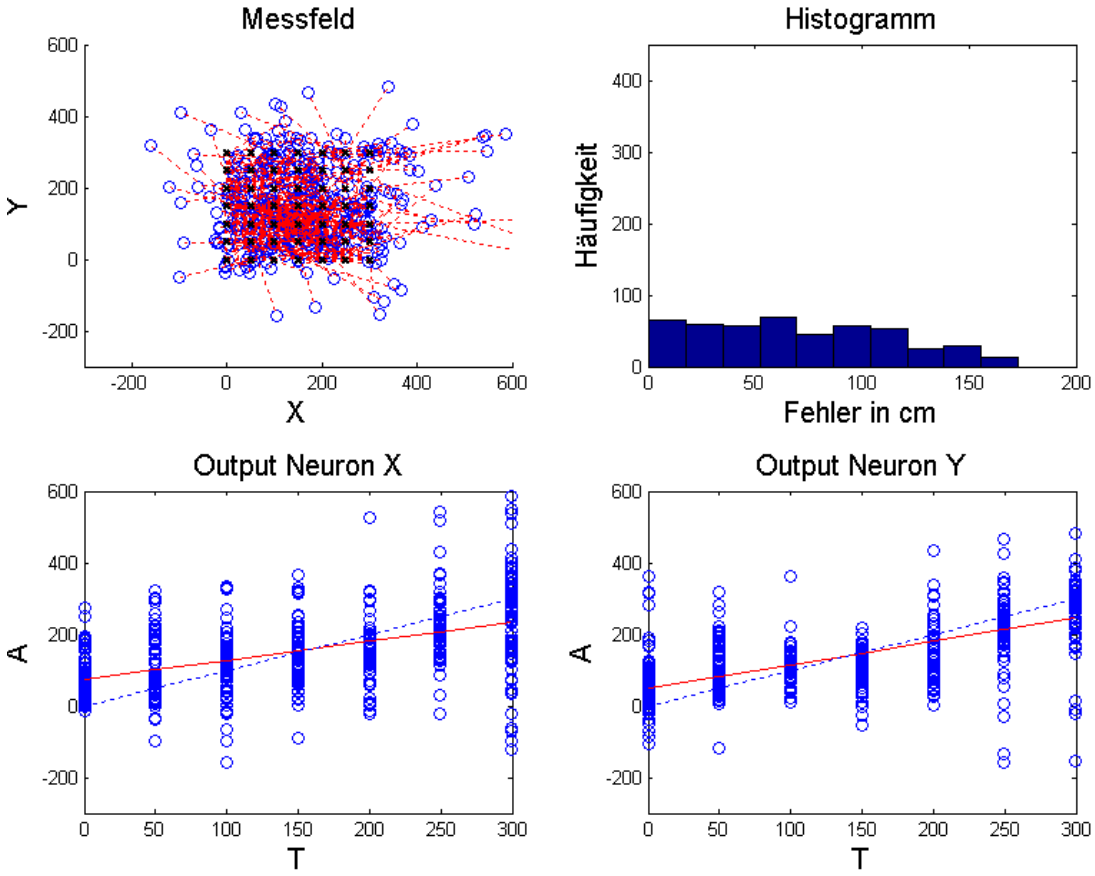


Abbildung 40: Analyse KNN-glatt für Geräteklasse 2 mit rohen Testdaten[17]

Das Messfeld ist nicht mehr erkennbar. Der durchschnittliche Fehler liegt bei 70,9cm der Median bei 65,5cm. Die Regressionsanalysen für die Outputneuronen fallen ebenfalls schlecht aus. Die Steigung der approximierten Geraden liegt für das Outputneuron der X-Koordinate bei 0,538 für die Y-Koordinate bei 0,649. Der Korrelationskoeffizient liegt für die X-Koordinate bei 0,452, für die Y-Koordinate bei 0,614.

[17] Das Testset findet man auf der CD im Ordner *Experimente\neuronaleNetze\Class2\Laptop roh 67-60-20-20\Trainingsdaten*

Die Lokalisierung kann mit dieser Versuchsanordnung am schlechtesten durchgeführt werden.

### *KNN mit Rohdaten*

In diesem Punkt werden die Lokalisierungsversuche, die mit dem KNN, das mit den Rohdaten des MD der Geräteklasse 2 trainiert wurde, präsentiert.

#### *Testset mit geglätteten Daten*

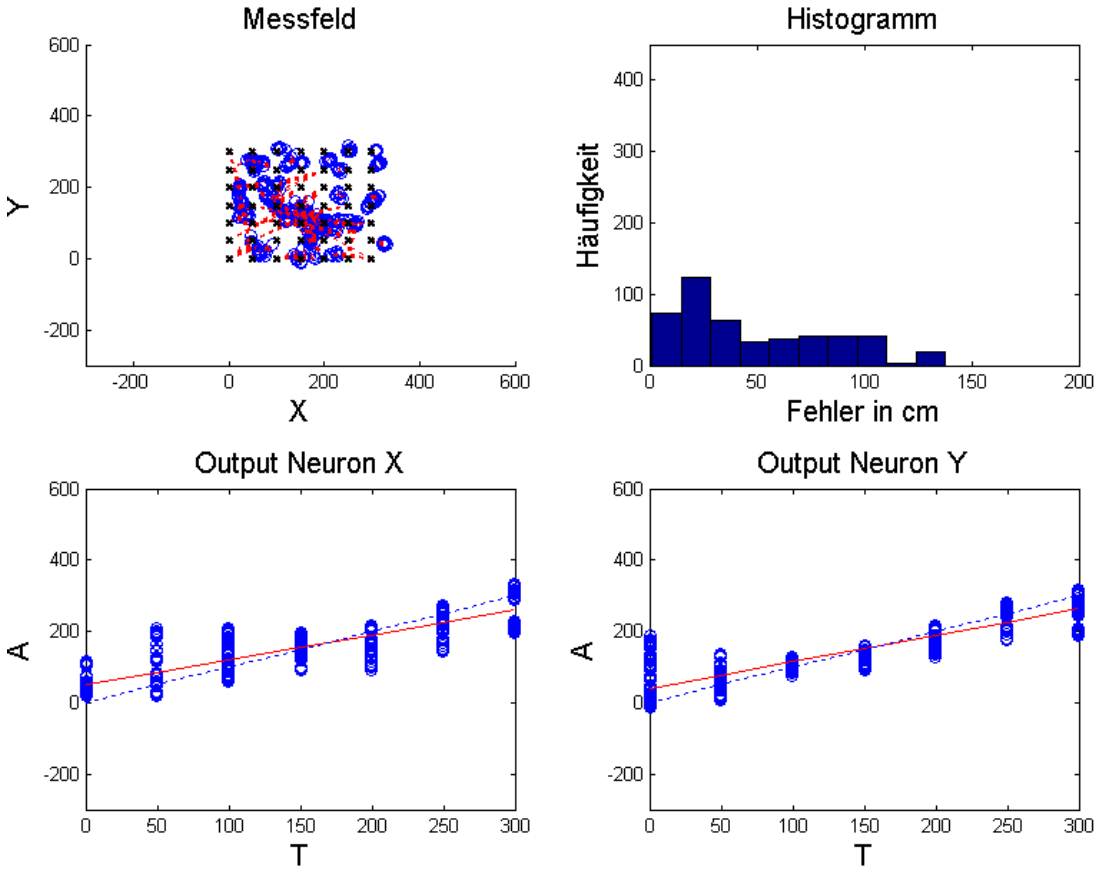


Abbildung 41: Analyse KNN-roh für Geräteklasse 2 mit geglätteten Testdaten[18]

Die Versuchsanordnung KNN-roh mit geglätteten Testdaten funktioniert relativ gut. Für das Outputneuron der X-Koordinate wird eine Steigung von 0,700 erreicht, für die Y-Koordinate 0,741, wie man in Abbildung 41 sehen kann. Das Ergebnis wird

[18] Das Testset findet man auf der CD im Ordner *Experimente\neuronaleNetze\Class2\Laptop 0-05 67-60-20-20\Trainingsdaten*

auch durch die Korrelationskoeffizienten gestützt. Für die X-Koordinate liegt r bei 0,849 für die Y-Koordinate bei 0,88. Der durchschnittliche Fehler und der Median liegen mit Werten von 49cm und 33,8cm noch unter dem Abstand zwischen Messpunkten, der bei 50cm liegt. Man kann ihn somit als gangbaren Ansatz bezeichnen.

### *Testset mit Rohdaten*

Den Abschluss der Versuche mit Class 2 Devices bildet die Anordnung KNN-roh mit dem Testset aus den Rohdaten.

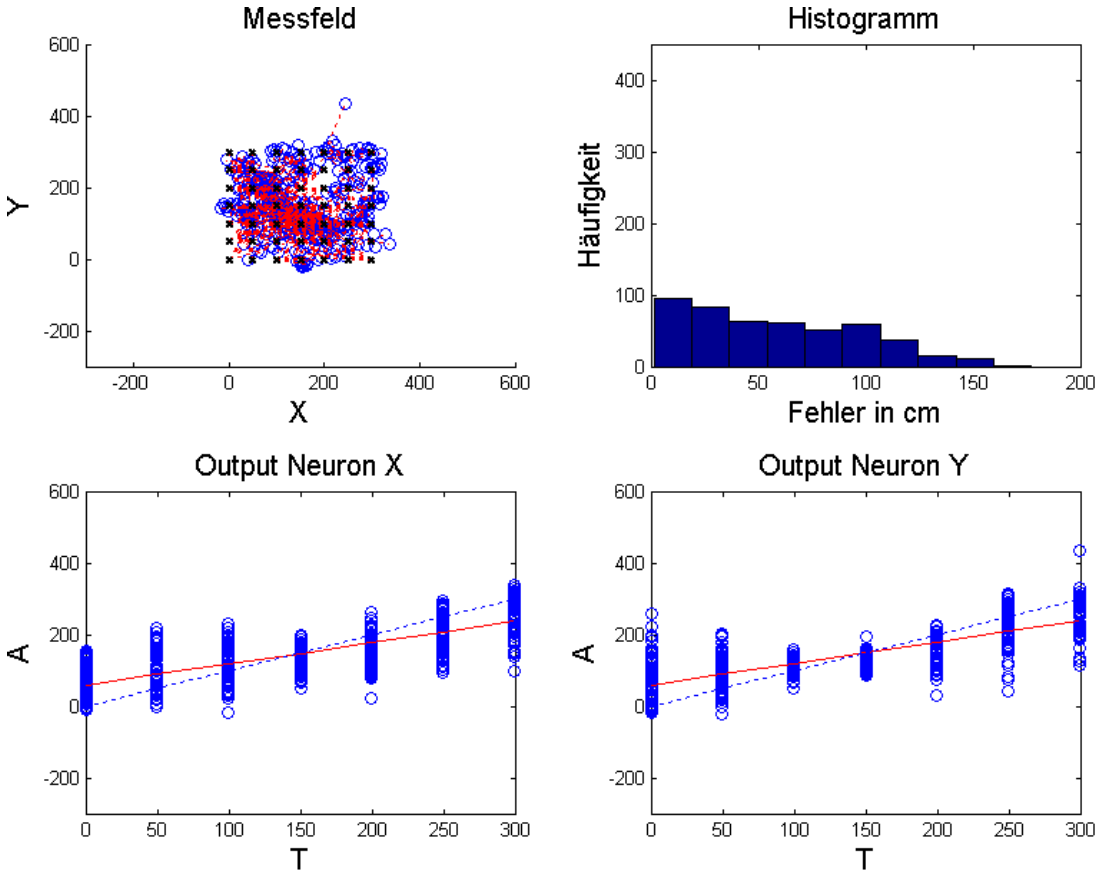


Abbildung 42: Analyse KNN-roh für Geräteklasse 2 mit rohen Testdaten[19]

Am Histogramm in Abbildung 42 erkennt man an der abfallenden Kurve, dass der durchschnittliche Fehler beim Training des KNN reduziert wurde. Trotzdem können mit diesem Versuchsaufbau keine zufriedenstellenden Ergebnisse erreicht werden. Der durchschnittliche Fehler und der Median liegen mit Werten von 58,8cm und

[19] Das Testset findet man auf der CD im Ordner *Experimente\neuronaleNetze\Class2\Laptop roh 67-60-20-20\Trainingsdaten*

53,9cm beide über dem Abstand zwischen den Messpunkten. Die Regressionsanalyse liefert für das X-Koordinaten Outputneuron eine Steigung von 0,593 und für die Y-Koordinate eine Steigung von 0,604. Für die X-Koordinate liegt r bei 0,758 für die Y-Koordinate bei 0,765. Eine Lokalisierung eines MD der Geräteklasse 2 auf 50cm genau, nur auf Basis der Rohdaten, ist somit nicht möglich.

### 7.3.2 Mobile Device Class 3

Derselbe Versuch wurde auch für das MD der Geräteklasse 3, dem Nokia N70, durchgeführt.

#### *KNN mit geglätteten Daten*

Es werden zuerst die Ergebnisse für das KNN, welches mit den geglätteten Daten trainiert wurde, präsentiert.

##### *Testset mit geglätteten Daten*

In der ersten Versuchsanordnung für den Lokalisierungsversuch, mit dem MD der Geräteklasse 3, wurden die geglätteten Testdaten mit dem KNN-glatt verwendet. In Abbildung 43 ist das Ergebnis zu sehen.

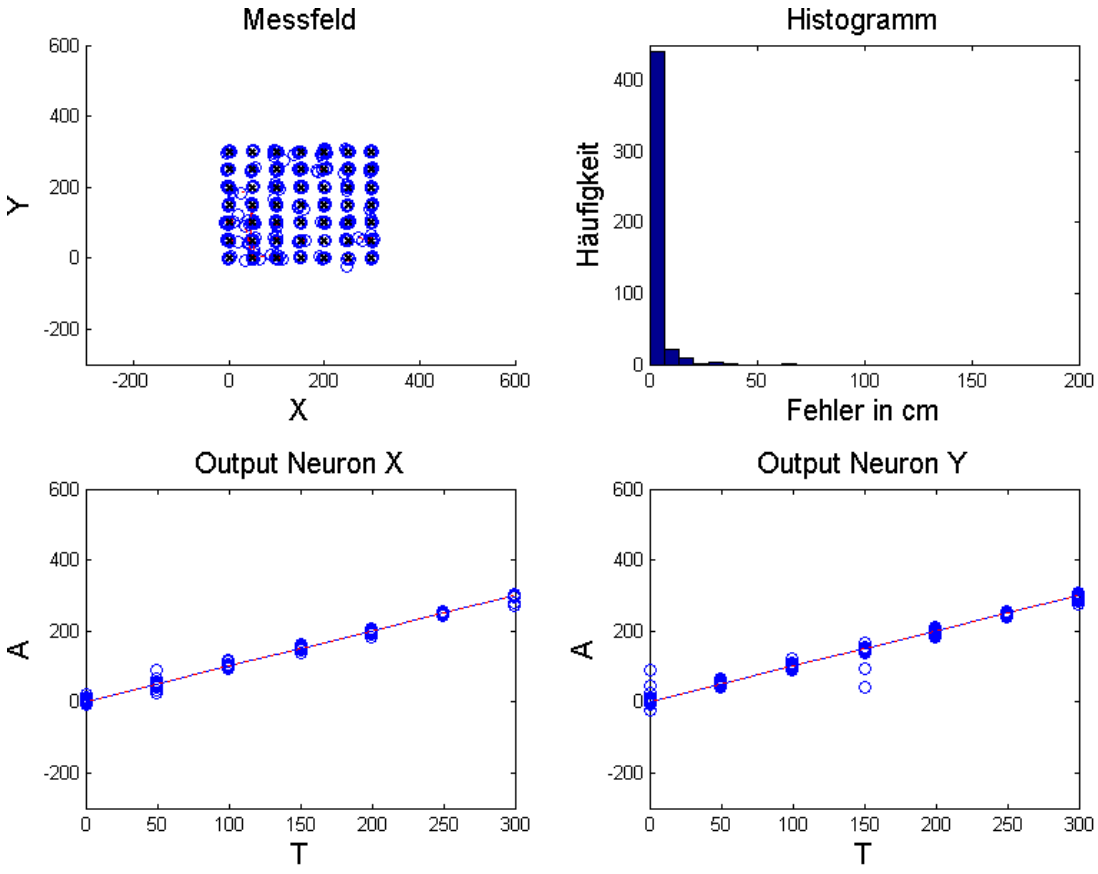


Abbildung 43: Analyse KNN-glatt für Geräteklasse 3 mit geglätteten Testdaten[20]

In diesem Versuchsdurchlauf konnte das beste Resultat erzielt werden. 440 von 470 Testdatensätzen konnten nahezu perfekt lokalisiert werden, wie man im Messfeld-diagramm und Histogramm sehen kann. Auch die Regressionsanalysen für die Outputneuronen liefern mit einer Steigung von 0,997 für das X-Neuron und 0,993 für das Y-Neuron nahezu perfekte Werte. Die Korrelationskoeffizienten sind mit 0,999 für die X-Koordinate und 0,997 für die Y-Koordinate ausgezeichnet. Der durchschnittliche Fehler liegt bei 3,4cm, der Median bei 1,9cm. Es kann daher festgehalten werden, dass die Lokalisierung eines MD der Geräteklasse 3 mit dieser Versuchsanordnung hervorragend durchgeführt werden kann.

### *Testset mit Rohdaten*

Das Ergebnis der Lokalisierung der Testdaten aus dem Rohdatensatz mit dem KNN-glatt ist in Abbildung 44 dargestellt.

[20] Das Testset findet man auf der CD im Ordner *Experimente\neuronaleNetze\Class3\Handy 0-05 67-60-20-20\Trainingsdaten*

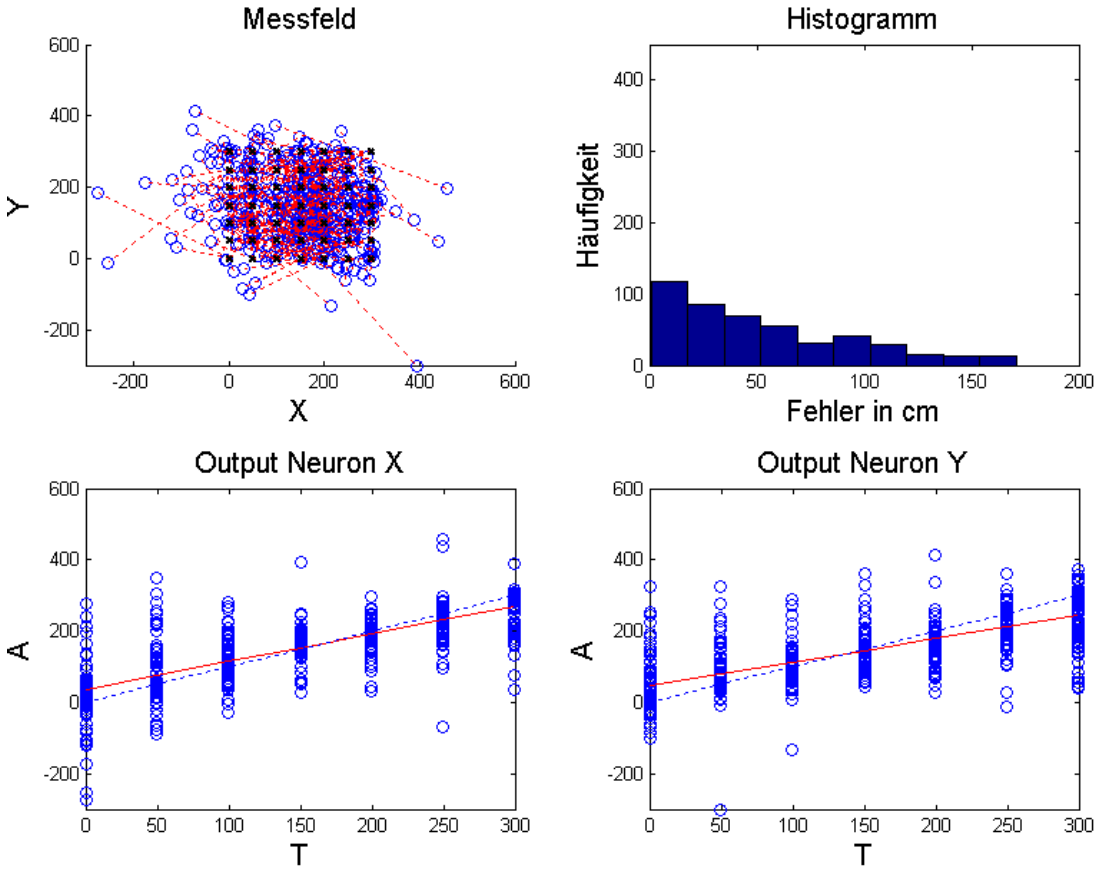


Abbildung 44: Analyse KNN-glatt für Geräteklasse 3 mit rohen Testdaten[21]

Genauso wie bei der Geräteklasse 2 liefert auch hier diese Versuchsanordnung die schlechtesten Ergebnisse. Im Durchschnitt liegt der Fehler bei der Lokalisierung bei 53,8cm, der Median bei 44,9cm. Bei der Regressionsanalyse wird für das Outputneuron der X-Koordinate eine Steigung von 0,777 und für die Y-Koordinate 0,658 erreicht. Weiters liefert die Regressionsanalyse für die X-Koordinate ein r von 0,738 und für die Y-Koordinate ein r von 0,677. Dieses Setup ist daher für eine Lokalisierung nicht geeignet.

### *KNN mit Rohdaten*

Für das KNN, das mit den Rohdaten des Class 3 MD trainiert wurde, wurde dieses Experiment ebenfalls durchgeführt.

[21] Das Testset findet man auf der CD im Ordner *Experimente\neuronaleNetze\Class3\Handy roh 67-60-20-20\Trainingsdaten*

### *Testset mit geglätteten Daten*

Nimmt das KNN-roh die Lokalisierung der geglätteten Testdaten vor, erreicht es dabei das folgende Ergebnis.

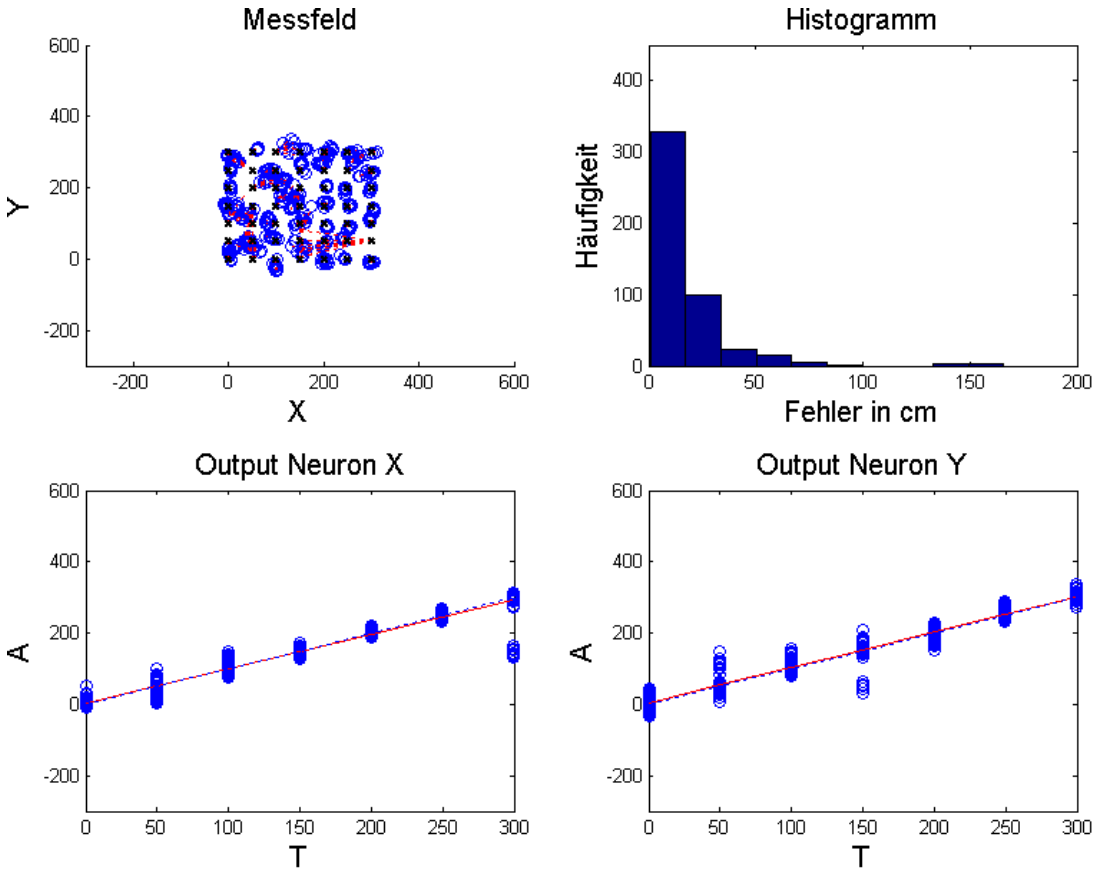


Abbildung 45: Analyse KNN-roh für Geräteklasse 3 mit geglätteten Testdaten[22]

Der durchschnittliche Fehler beim Lokalisierungsvorgang beträgt 18,2cm, der Median 11,9cm. Dem Histogramm in Abbildung 45 kann man entnehmen, dass ca. 330 der 470 Testdatensätze sehr genau lokalisiert werden können. Die Regressionsanalyse stützt ebenfalls diese guten Ergebnisse. So hat die approximierte Gerade für das Outputneuron der X-Koordinate eine Steigung von 0,957, bei der Y-Koordinate wird ein Wert von 0,993 erreicht. Für die X-Koordinate liegt r bei 0,971 und für die Y-Koordinate bei 0,975. Mit dieser Versuchsanordnung konnte das zweitbeste Ergebnis erzielt werden.

[22] Das Testset findet man auf der CD im Ordner *Experimente\neuronaleNetze\Class3\Handy 0-05 67-60-20-20\Trainingsdaten*

### *Testset mit Rohdaten*

In der letzten Versuchsanordnung wurde das Testset mit Rohdaten mit dem KNN-roh lokalisiert.

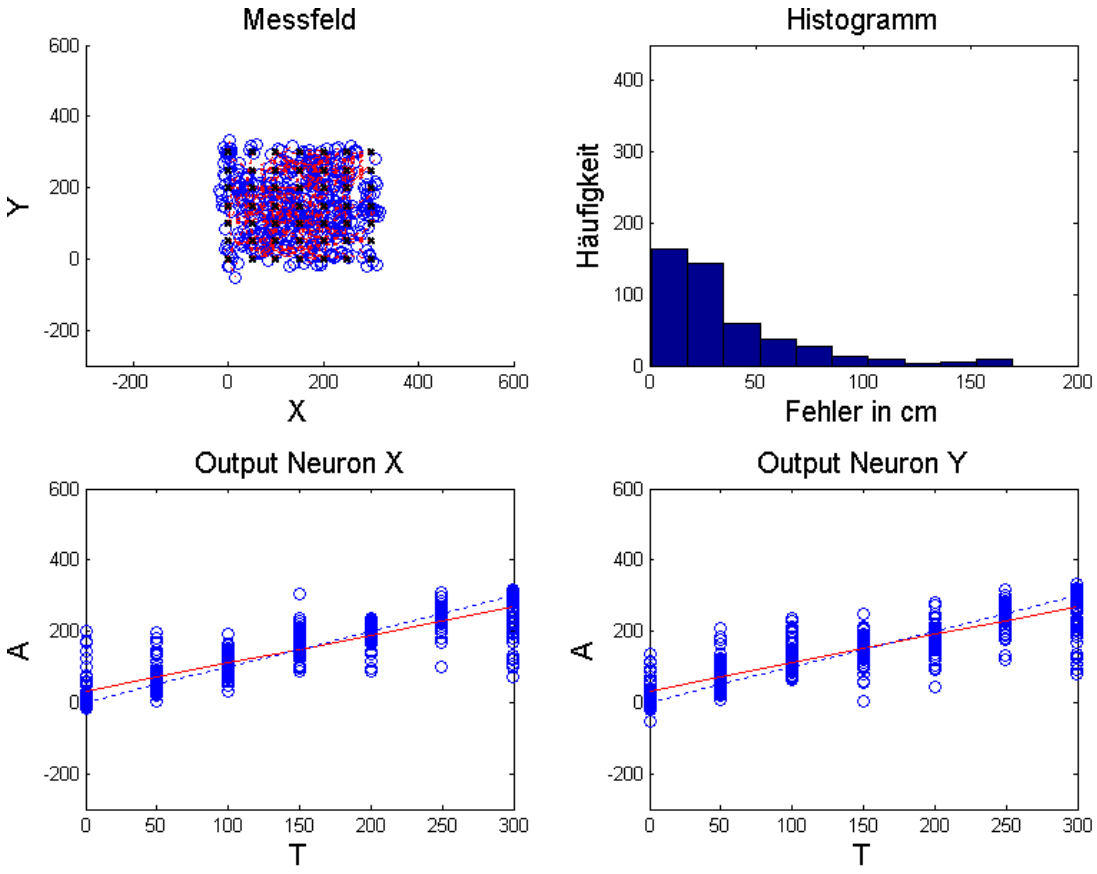


Abbildung 46: Analyse KNN-roh für Geräteklasse 3 mit rohen Testdaten[23]

Im Histogramm in Abbildung 46 sieht man, dass auch hier, genauso wie bei dem KNN-roh der Geräteklasse 2, der durchschnittliche Fehler minimiert wurde. Jedoch liegt der durchschnittliche Fehler bei der Lokalisierung mit Rohdaten durch das KNN-roh nur bei 36,7cm und der Median bei 25,6cm. Diese Werte liegen deutlich unter der Distanz von 50cm zwischen zwei Messpunkten. Eine Lokalisierung ist daher mit diesem Versuchsaufbau möglich. Die Steigung der approximierten Geraden liegt für die X-Koordinate bei 0,793, für die Y-Koordinate bei 0,789. Der Korrelationskoeffizient liegt für die X-Koordinate bei 0,883, für die Y-Koordinate bei 0,87.

[23] Das Testset findet man auf der CD im Ordner *Experimente\neuronaleNetze\Class3\Handy roh 67-60-20-20\Trainingsdaten*

## 7.4 Vergleich Class 2 und Class 3 Devices

In Tabelle 8 und Tabelle 9 sind die Resultate der Experimente zur Lokalisierung mit KNNs für beide Geräteklassen zusammengefasst. Für das MD der Geräteklasse 3 können in allen Versuchsdurchläufen die besseren Ergebnisse erzielt werden.

| **Trainingsdaten** | **glatt** | | **roh** | |
|---|---|---|---|---|
| **Testdaten** | **glatt** | **roh** | **glatt** | **roh** |
| Neuron X r | 0,986 | 0,452 | 0,849 | 0,758 |
| Neuron X m | 0,986 | 0,538 | 0,700 | 0,593 |
| Neuron Y r | 0,988 | 0,614 | 0,880 | 0,765 |
| Neuron Y m | 0,980 | 0,649 | 0,741 | 0,604 |
| Mean | 9,600 | 70,960 | 49,040 | 58,853 |
| Median | 5,444 | 65,577 | 33,860 | 53,930 |

Tabelle 8: Lokalisierung Vergleich Geräteklasse 2

| **Trainingsdaten** | **glatt** | | **roh** | |
|---|---|---|---|---|
| **Testdaten** | **glatt** | **roh** | **glatt** | **roh** |
| Neuron X r | 0,999 | 0,738 | 0,971 | 0,883 |
| Neuron X m | 0,997 | 0,777 | 0,957 | 0,793 |
| Neuron Y r | 0,997 | 0,677 | 0,975 | 0,870 |
| Neuron Y m | 0,993 | 0,658 | 0,993 | 0,789 |
| Mean | 3,417 | 53,850 | 18,290 | 36,760 |
| Median | 1,920 | 44,910 | 11,910 | 25,610 |

Tabelle 9: Lokalisierung Vergleich Geräteklasse 3

Innerhalb der Geräteklasse ergibt sich für beide dieselbe Reihung. Am genauesten kann die Lokalisierung durchgeführt werden, wenn das KNN, welches mit geglätteten Daten trainiert wurde, die Position für geglättete Daten bestimmen soll.

Das KNN, das mit Rohdaten trainiert wurde, kann die Position für geglättete Daten ebenfalls ausreichend genau bestimmen.

Bei Class 3 Devices kann mit einem solchen Netz auch noch mit Rohdaten die Position ausreichend genau bestimmt werden. Für Class 2 Devices ist die Positionierung nicht mehr ausreichend möglich.

In beiden Geräteklassen werden die schlechtesten Ergebnisse erzielt, wenn man versucht, Rohdaten mit dem KNN zu lokalisieren, welches mit geglätteten Daten trainiert wurde.

## 7.5 Lösungsvorschlag

Wie im vorigen Punkt besprochen wurde, eignen sich mit Rohdaten trainierte KNNs am besten für die Lokalisierung von MDs, da sie sowohl für Rohdaten als auch für geglättete Daten den Prozess durchführen können. KNNs, die mit geglätteten Daten trainiert wurden, erzielen zwar bei der Lokalisierung von geglätteten Daten die besten Resultate, sind aber nicht in der Lage, die Lokalisierung mit Rohdaten durchzuführen.

## 7.6 Verbesserungsmöglichkeiten

Um die Ergebnisse bei der Lokalisierung weiter zu verbessern, wäre ein möglicher Ansatz nur jene BSs für den Lokalisierungsprozess zu verwenden, die aussagekräftige Signalstärkewerte liefern. Ein aussagekräftiger Signalstärkewert ist für die Geräteklassen 2 und 3 unterschiedlich definiert.

Wenn bei Class 2 Devices zwischen MD und BS keine Hindernisse liegen, erhält man keine aussagekräftigen Signalstärkewerte da die Signalstärke immer sehr gut ist (850-900) was eine Unterscheidung von Messpunkten unmöglich macht. Bei BS1 war das bei dem Experiment in 6.2.3 der Fall. Das Ergebnis des Experiments hat gezeigt, dass BS1 kaum zu unterscheidende Signalstärkewerte liefert (siehe Abbildung 31 und Abbildung 32 in Kapitel 6.2.3). Für 41 von 49 Messpunkten liegt der gemessene Signalstärkewert zwischen 850 und 900. Berücksichtigt man noch die Schwankung der Signalstärke kann BS1 diese 41 Messpunkte nicht voneinander unterscheiden. Eine Wiederholung des Experiments, in dem ein KNN nur mit den drei restlichen BSs trainiert wird, wäre interessant, da dieses Netz die Lokalisierung von Class 2 Devices wahrscheinlich besser durchführen und die Verfälschung durch BS1 veranschaulichen könnte.

Bei MDs der Geräteklasse 3 ist man mit dem Problem konfrontiert, dass die Messwerte mit zunehmender Entfernung zwischen BS und MD stark fluktuieren, was zu schlechteren Ergebnissen bei der Lokalisierung führt. Ein Beispiel dafür wurde in Abbildung 35 (siehe Kapitel 7.1.1) gezeigt. Man sieht, dass für BS2 am Messpunkt 100, 100 der Signalstärkewert zwischen 200 und 500 fluktuiert. Eine mögliche Lö-

sung des Problems ist das Glätten der Messdaten wie in der Arbeit gezeigt wurde. Um die Lokalisierung aber für Rohdaten effektiver zu gestalten, könnte man BSs für die Messpunkte, an denen ihre Signalstärkefluktuation über 250 liegt, aus dem Lokalisierungsprozess ausschließen. An diesen Messpunkten legt eine solche BS kein Eingangssignal an das KNN an. Das MD wird dann nur von den BSs lokalisiert, die eindeutigere Messwerte liefern.

# 8 Conclusio

*Das letzte Kapitel gibt einen Rückblick auf die Arbeit und fasst die Ergebnisse, die erarbeitet werden konnten, zusammen. Weiters wird ein mögliches Anwendungsszenario für den ausgearbeiteten Ansatz beschrieben. Der Punkt 8.2 „Abschließende Bewertung" beinhaltet ein letztes Fazit und gibt einen Ausblick auf mögliche zukünftige Entwicklungen im Bereich der Lokalisierung mittels Bluetooth.*

In der Einleitung zu dieser Arbeit wurde die Frage gestellt, wie sich die momentane Position eines Benutzers von einem Mobile Device (MD), wie z.B. eines Handys oder PDAs, in einem Gebäude bestimmen lässt. Das Global Positioning System (GPS) kann in Gebäuden nicht verwendet werden. Ein lokales Funksystem wie Bluetooth, mit dem die meisten MDs ausgestattet sind, stellt einen möglichen Ansatz dar.

In den verwandten Arbeiten, die vorgestellt wurden, konnte die Position eines MD in einem Gebäude mit einer Genauigkeit von 3m bestimmt werden. Die Grundlage für die Lokalisierung bildet in diesen vorgestellten Arbeiten eine Distanzbestimmung zwischen dem MD und den Base Stations (BS), deren Position bekannt ist. Die Distanzbestimmung erfolgt auf Basis der Funkwellenausbreitung. Hindernisse wie Wände, Türen und Möbel behindern diese. Aus diesem Grund kann die Position nur mit dieser geringen Genauigkeit von 3m bestimmt werden.

Deshalb wurde in dieser Arbeit ein Ansatz gewählt, der sich der Umgebung anpasst, um so die Exaktheit der Lokalisierung zu erhöhen. Um das zu erreichen, wurden an mehreren Punkten Signalstärkemessungen von den BSs zu einem MD durchgeführt. Mit den daraus gewonnenen Daten wurde ein KNN (künstliches neuronales Netz) trainiert, das dann in der Lage ist, eine Lokalisierung eines MD durchzuführen. Das Ziel ist es, die Position eines MD auf 50cm genau bestimmen zu können.

Um dieses Experiment durchführen zu können, wurde im ersten Schritt ein Messsystem implementiert, mit dem sich die drei Signalstärkeparameter von Bluetooth, der RSSI, LQ und TPL Wert messen lassen. Der Zugriff auf diese Werte gestaltet sich in der Praxis als äußerst schwierig. Um sie lesen zu können, benötigt der Programmierer den Zugriff auf das HCI des Bluetooth-Softwarestacks. Unter Windows ist dies nicht möglich. Der offizielle Bluetooth-Softwarestack von Linux BlueZ bietet diese Möglichkeit, und die kostenlose Softwarebibliothek JTooth ermöglicht die Nutzung von BlueZ in der Programmiersprache Java. Auf dieser Grundlage wurde das Messsystem entwickelt. Das Messsystem besteht aus einem Programm BaseStation (BS)

und dem BTLocationServer, der die Messdaten der einzelnen BaseStations protokolliert.

*Alle im Rahmen dieser Arbeit durchgeführten Experimente wurden für MDs der Geräteklasse 2 und 3 durchgeführt.*

In den verwandten Arbeiten, die vorgestellt wurden, herrscht Uneinigkeit über die Aussagekraft der einzelnen Signalstärkeparameter. Sicher ist, dass die Qualität der Werte bei jedem Bluetooth-Modultyp variieren, da die Bluetooth-Spezifikation dem Hersteller einen gewissen Spielraum einräumt. Dieser Umstand machte es notwendig, mehrere Bluetooth-Module und die einzelnen Signalstärkeparameter in einem Experiment zu untersuchen, um für die weiteren Versuche ein geeignetes Bluetooth-Modul und einen geeigneten Signalstärkewert zu finden. In dem in Kapitel 6.1 durchgeführten Versuch stellte sich der MSI BToes 2.0 Dongle als geeignetes Bluetooth-Modul heraus. Mit einem Anschaffungspreis von 8,36€ wird er auch dem Kriterium eines kostengünstigen Messsystems gerecht. Jeder der BS wurde mit diesem Dongle als Bluetooth-Modul ausgestattet. Als aussagekräftige Signalstärkeparameter erwiesen sich der RSSI und TPL Wert. Der errechnete Signalstärkewert, der in weiterer Folge verwendet wird, wird daher auch aus diesen beiden Parametern errechnet (genaue Formel siehe 5.2.1).

Im darauf folgenden Experiment wurden die Messdaten für die Lokalisierungsversuche erhoben. Dazu wurde ein 3x3m großes Messfeld mit 49 Messpunkten im Abstand von 50cm abgesteckt. Die Signalstärke zum MD wurde von vier BSs an allen 49 Messpunkten für 15 Minuten gemessen. Dabei wurden die Signalstärkewerte in einem Abstand von 0,5s abgetastet und an den BTLocationServer gesendet, wo sie gespeichert wurden. Bei der graphischen Darstellung des Messfeldes aus Sicht der einzelnen BSs konnte der Einfluss der Umgebung auf die Signalstärke deutlich gezeigt werden (Kapitel 6.2.3).

Die gewonnen Messdaten (Rohdaten) wurden im nächsten Schritt mit einer Glättungsfunktion bearbeitet, um eine zweite Datenmenge mit eindeutigeren Werten zu erhalten (geglättete Daten). Es wurde aus den Datenmengen Trainings- und Testmengen ausgewählt. Für beide Geräteklassen wurde ein KNN mit Rohdaten und eines mit geglätteten Daten trainiert. Im abschließenden Versuch wurde nun erprobt, wie gut sich die Testsets mit den KNNs lokalisieren lassen. Es zeigte sich, dass ein KNN, das mit Rohdaten trainiert wurde, Rohdaten und geglättete Daten lokalisieren kann. Ein KNN, das mit geglätteten Daten trainiert wurde, kann zwar nicht trainierte geglättete Daten sehr gut lokalisieren, ist aber nicht in der Lage, Rohdaten zu verar-

beiten. Aus diesem Grund empfiehlt es sich, in der Praxis KNNs mit Rohdaten zu trainieren.

Dieses Ergebnis trifft auf beide Geräteklassen zu, wobei zu erwähnen ist, dass sich MDs der Geräteklasse 3 genauer lokalisieren lassen. Aufgrund der geringeren Sendeleistung können differenziertere Signalstärkewerte gemessen werden, was eine genauere Positionierung zulässt. Die durchschnittliche Fehlerzahl bei MDs der Geräteklasse 3, wenn ein mit Rohdaten trainiertes Netz geglättete Daten lokalisiert, liegt bei 18,29cm. Wenn es Rohdaten lokalisiert, bei 36,67cm. Bei der Geräteklasse 2 liegt die durchschnittliche Fehlerzahl bei 49,04cm bzw. 58,85cm.

## 8.1 Anwendungsszenarien

Anwendungsmöglichkeiten für Location Based Services in Gebäuden gibt es viele. Beispiele dafür wären unter anderem ein elektronischer Museumsführer, ein Einkaufsassistent in großen Kaufhäusern oder eine Friend Finder Applikation. Aber inwieweit können die Ergebnisse dieser Arbeit zur Verbesserung solcher Dienste beitragen?

Es wurde in dieser Arbeit gezeigt, dass die Position eines MD in Gebäuden mit KNN wesentlich genauer bestimmt werden kann als z.B. mit Trilationsverfahren. Der Nachteil bei der Lokalisierung über KNN besteht darin, dass im Vorhinein Messwerte genommen werden müssen, mit denen das KNN trainiert werden kann. Das KNN ist dann in der Lage, in trainierter Umgebung die Position eines MD genau zu bestimmen, aber eben nur in der trainierten Umgebung. Um die Genauigkeit bei der Lokalisierung zu erhöhen, muss also ein hoher Aufwand betrieben werden. Aus diesem Grund wäre in der Praxis ein Hybridmodell ein möglicher Ansatz. In den Bereichen, in denen keine hohe Genauigkeit benötigt wird, kann die Position mittels Trilationsverfahren bestimmt werden, das kein Training oder ähnliche Vorbereitung benötigt. Für Bereiche, in denen die Position genau bestimmt werden muss, werden KNNs trainiert. Bewegt sich der Benutzer mit seinem MD in einem solchen Bereich, wird die Position nicht mehr über Trilation bestimmt, sondern über das für diesen Bereich trainierte KNN.

Mit einem solchen System lassen sich die Vorteile beider Ansätze vereinen: Die Einfachheit des Trilationsverfahrens mit der Genauigkeit des Ansatzes mit einem KNN.

## 8.2 Abschließende Bewertung

In dieser und den vorgestellten verwandten Arbeiten konnte gezeigt werden, dass Lokalisierung mittels Bluetooth möglich ist. Da die Funksignalausbreitung bei Bluetooth stark von der Umgebung beeinflusst wird, können mit Ansätzen, die sich der Umgebung anpassen, bessere Ergebnisse erzielt werden. Das konnte in der Arbeit Bluetooth Base Station Minimal Deployment for High Definition Positioning (siehe Kapitel 2.3) und in der vorliegenden Arbeit gezeigt werden. Es müssen jedoch zuerst Messdaten in der Umgebung genommen werden, aus denen die Lernalgorithmen das Wissen über die Umgebung extrahieren können. Das macht sie in der Verwendung aufwendiger als Systeme, die die Lokalisierung z.B. über Trilation durchführen.

Die Frage, warum Lokalisierungssysteme basierend auf Bluetooth in der Praxis noch keine große Bedeutung spielen, bleibt. Der Hauptgrund ist die begrenzte Akkukapazität, über die MDs verfügen. Wenn ein MD lokalisiert werden soll, muss die Bluetooth-Schnittstelle ständig aktiv sein und auch Daten austauschen. Das kostet dem MD wertvolle Energie. Bei geringem Akkustand wird diese bei Sendeleistung von Bluetooth-Geräten reduziert, was sich wiederum negativ auf die Genauigkeit der Lokalisierung auswirkt.

Ein weiterer großer Energieverbraucher ist das Display eines MD. Um sinnvolle Location Based Services anbieten zu können, ist eine gewisse Größe notwendig, um Informationen übersichtlich darstellen zu können und Userinteraktionen z.B. mittels Touch Screen zu ermöglichen. PDAs bieten diese Funktionen, und mobile Telefone beginnen sich ebenfalls in diese Richtung zu entwickeln, wie man z.B. am iPhone von Apple sehen kann. Auch die Akkukapazitäten konnten im Laufe der letzten Jahre laufend verbessert werden. Diese rasche Weiterentwicklung von MDs lässt darauf hoffen, dass in den nächsten Jahren die Masse der sich im Umlauf befindenden Geräte in der Lage sein wird, LBS über Bluetooth sinnvoll zu nutzen.

Ein weiterer heikler Punkt bei der Realisierung von Lokalisierungssystemen über Bluetooth ist der Bluetooth-Standard selbst. In einem Piconetz kann es nur acht aktive Teilnehmer geben, einen Master und sieben Slaves. Für eine genaue Lokalisierung muss ein MD aber zu mehreren BSs verbunden sein. Diese beiden Umstände machen die Servicierung einer größeren Zahl von Usern äußerst schwierig. Weiters ist zu bedenken, dass die Performance von Bluetooth sinkt, je mehr Bluetooth-Geräte in Sendereichweite sind, da es häufiger zu Datenkollisionen kommt. Um Bluetooth-Lokalisierungssysteme in naher Zukunft umsetzten zu können, wäre es wichtig, dass man es schafft, die Zahl der aktiven Teilnehmer in einem Piconetz zu erhöhen und den Frequenzraum effektiver zu nutzen.

Wünschenswert wäre ebenfalls eine striktere Definition des LQ, RSSI und TPL Parameters, da dies eine exaktere Lokalisierung und eine bessere Vergleichbarkeit von Bluetooth-Modulen ermöglichen würde.

# 9 Anhang

*In diesem Kapitel findet man die Benutzeranleitung zu Software und Matlab Skripten, die im Laufe dieser Arbeit implementiert wurden.*

## 9.1 Benutzerhandbuch Messsystem

Das entwickelte Messsystem besteht aus den Programmen BTLocationServer und BaseStation. Im Benutzerhandbuch sind die Begriffe BTLocationServer und Server gleichbedeutend. Base Station (BS) wird als Synonym für das Programm BaseStation verwendet.

### 9.1.1 BTLocationServer

#### *Installation*

Der BTLocationServer ist an kein bestimmtes Betriebssystem gebunden. Er benötigt eine SWING-fähige Java Umgebung und die Redstone XML-RPC Library für die Kommunikation mit dem Programm BaseStation. Die benötigten Klassen der Redstone XML-RPC Library wurden der Datei BTLocationServer.jar hinzugefügt. Der Benutzer muss daher keine externen Programmbibliotheken bereitstellen.

#### *Start*

Der Server wird über die Kommandozeile mit dem Befehl:

```
java -jar BTLocationServer.jar [Konfigurationsdatei]
```

gestartet. Man hat die Möglichkeit, eine Konfigurationsdatei anzugeben. Wenn keine angegeben wird, versucht der Server standardmäßig die Datei *BTLocationServer.ini* zu laden. Befindet sich diese nicht im selben Verzeichnis wie die Datei *BTLocationServer.jar,* verwendet der Server die Defaulteinstellungen. Die Datei *BTLocationServer.jar* findet man auf der CD im Ordner *Software\BTLocationServer*.

Der BTLocationServer muss vor den Base Stations gestartet werden.

### *Konfiguration*

Die Einstellungen für den Server werden über eine Konfigurationsdatei getroffen. Einige Parameter werden vom Server an die BSs übertragen. Diese Parameter sind in Tabelle 10 kursiv dargestellt.

| **Parametername** | **Beschreibung** | **Defaultwert** |
|---|---|---|
| ServerPort | Port für die Anfragen der Base Stations | 2181 |
| MeasurementFileExtension | Endung für die Datei, welche den errechneten Signalstärkewert enthält | .txt |
| RAWFileExtension | Endung für die Datei, welche errechneten Signalstärkewert und die gemessenen RSSI-, LQ- und TPL-Wert enthält. | _raw.txt |
| Separator | Verwendetes Trennzeichen im MeasurementFile und RAWFile | ; |
| NoMeasurementValue | Ausdruck, der in das MesurmentFile und RAWFile geschrieben wird, wenn keine Messwerte vorliegen | ? |
| StationSeparator | Trennzeichen zwischen den Base Stations im RAWFile | \| |
| *DeviceLookUpInterval* | *Zeitspanne zwischen zwei Messungen der Signalstärkeparameter in Millisekunden* | *1000* |
| *HoldConnection* | *Verbindungsdauer zwischen einer Base Station und einem Mobile Device in Millisekunden* | *60000* |
| *RetryAfterError* | *Wartezeit nach einem erfolglosen Verbindungsaufbau zwischen Base Station und Mobile Device in Millisekunden* | *3000* |
| *RetryAfterSuccess* | *Wartezeit nach einem erfolgreichen Verbindungsaufbau zwischen Base Station und Mobile Device in Millisekunden* | *60000* |
| *Modus* | *Betriebsmodus für die Base Stations Development=0, Real=1 und Measurement=2 (siehe Kapitel 5.1 )* | *1* |
| *Device2Measure* | *Bluetooth-Adresse des Mobile Devices, wenn die Base Stations im Mouds Messung arbeiten* | |
| *Dongles* | *Bluetooth-Adressen der Bluetooth-Module an den Base Stations. Die Bluetooth-Adressen werden durch"\|" voneinander getrennt.* | |

Tabelle 10: Serverkonfigurationsdatei

Den BTLocationServer und ein Beispiel für eine Konfigurationsdatei findet man auf der CD im Ordner *Software\BTLocationServer*.

### Verwendung

Nach dem Start erscheint das Hauptfenster des BTLocationServers am Bildschirm. In den beiden Listen Stations und Devices befinden sich noch keine Einträge.

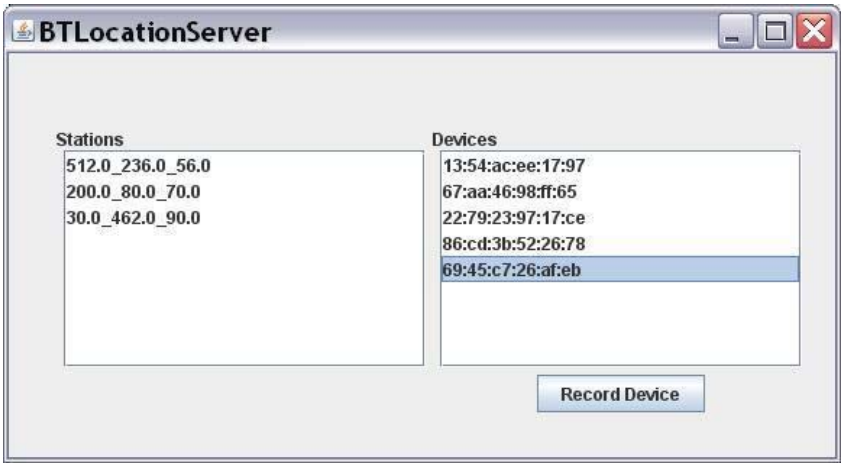


Abbildung 47: BTLocationServer Hauptfenster

Sobald eine BS gestartet wird, meldet sich diese beim BTLocationServer an. Dazu sendet die BS ihre Koordinaten. Sie wird dann automatisch in die Liste *Stations* aufgenommen. Wie man in Abbildung 47 sehen kann, hat die BS, die sich als erste am Server angemeldet hat, die Koordinaten X=512, Y=236 und Z=56. Sobald eine der BSs ein Bluetooth-Gerät in ihrer Nähe findet, versucht sie, eine Verbindung zu diesem aufzubauen. Ist sie damit erfolgreich, misst sie die Signalstärkeparameter der Verbindung und schickt diese gemeinsam mit der Bluetooth-Adresse des gefundenen Geräts an den BTLocationServer. Dieser trägt die Bluetooth-Adresse in die Liste *Devices* ein. Der Benutzer hat mit dieser Liste eine Übersicht über alle Bluetooth-Geräte, die von den BSs gefunden wurden.

Will der User die von den BSs gemessenen Signalstärkeparameter für ein bestimmtes Device aufzeichnen, markiert er durch einen Mausklick die Adresse des gewünschten Gerätes in der Liste *Devices* und betätigt anschließend die Schaltfläche *Record Device*. Es erscheint der Dialog *Record Dialog*.

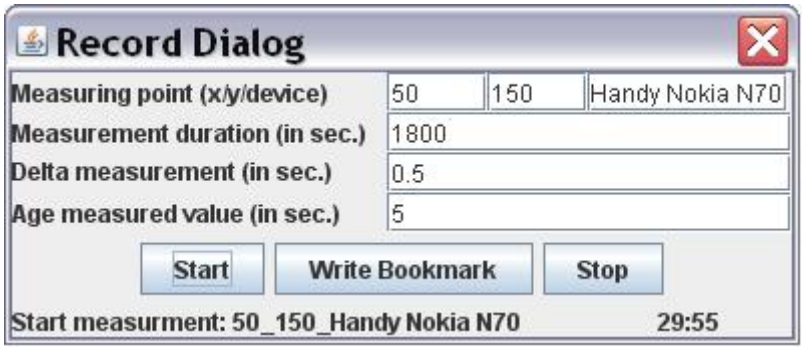


Abbildung 48: Record Dialog

In der ersten Zeile kann der Benutzer die X- und Y-Koordinate, an der sich das Bluetooth-Gerät befindet, eingeben. Er hat auch die Möglichkeit, einen Namen für das Gerät zu vergeben. Im Feld *Measurement duration* gibt der Benutzer die Dauer des gesamten Messungsvorgangs in Sekunden an. Im Feld *Delta measurement* wird die Zeitspanne zwischen zwei Messungen in Sekunden angegeben. Im Feld *Age measured value* gibt man an, wie alt ein Messwert maximal sein darf, dass er noch als gültig betrachtet wird.

Sobald der Benutzer auf den Button *Start* klickt, beginnt der Messvorgang. Es werden zwei Dateien angelegt, in denen die Messwerte gespeichert werden. Der Dateiname setzt sich aus den Koordinaten sowie dem vergebenen Namen für das Bluetooth-Gerät zusammen. Die Endungen der Dateien werden über die Parameter *MeasurementFileExtension* und *RAWFileExtension* in der Serverkonfigurationsdatei angegeben. Mit dem Button *Write Bookmark* wird eine Markierung in beide Dateien geschrieben. Mit dem Button *Stop* kann der Benutzer den Messvorgang vorzeitig abbrechen. Ganz unten rechts im Dialog wird die Restdauer des Messvorgangs angezeigt.

### 9.1.2 BaseStation

#### *Installation*

Das Programm BaseStation verwendet den offiziellen Bluetooth-Softwarestack von Linux: BlueZ. Es kann daher nur unter einer Linux-Distribution betrieben werden. Man kann jede beliebige Distribution verwenden, da BlueZ, laut offizieller Homepage, zu jeder kompatibel ist.

Im Rahmen dieser Arbeit wurde Ubuntu 7.04 verwendet, und es waren die in Abbildung 49 gezeigten Bluetooth-Pakete installiert:

| Paket | Installierte Version |
|---|---|
| bluetooth | 3.9-0ubuntu4 |
| bluez-btsco | 1:0.50-0ubuntu2 |
| bluez-cups | 3.9-0ubuntu4 |
| bluez-gnome | 0.6-1ubuntu3 |
| bluez-hcidump | 1.33-0ubuntu1 |
| bluez-pcmcia-support | 3.9-0ubuntu4 |
| bluez-pin | 0.30-2.1ubuntu3 |
| bluez-utils | 3.9-0ubuntu4 |
| btscanner | 2.1-3 |
| gnome-bluetooth | 0.8.0-0ubuntu4 |
| gnome-themes | 2.18.1-0ubuntu1 |
| gtk2-engines | 1:2.10.1-0ubuntu1 |
| gtk2-engines-ubuntulooks | 0.9.12-4 |
| kdebluetooth | 0.99+1.0beta2-1ub |
| libbluetooth2 | 3.9-0ubuntu1 |
| libbluetooth2-dev | 3.9-0ubuntu1 |
| libbtctl4 | 0.8.2-0ubuntu6 |
| libbtctl4-dev | 0.8.2-0ubuntu6 |
| libgnomebt0 | 0.8.0-0ubuntu4 |
| libgnomebt0-dev | 0.8.0-0ubuntu4 |
| multisync | 0.82-8build1 |
| nautilus-sendto | 0.10-0ubuntu1 |
| qobex | 0.99+1.0beta2-1ub |

Abbildung 49: verwendete Bluetooth-Pakete

Die Bluetooth-Pakete können mit der Synaptic Paketverwaltung installiert werden. Als Nächstes muss man noch bluez-sdp installieren. Dieses Paket findet man auf der CD im Ordner *Software\BaseStation\Libs* in Form der Datei *bluez-sdp_1.4.orig.tar.gz*. Beim Entpacken dieses Archivs wird der Ordner *bluez-sdp-1.4*

angelegt. Im nächsten Schritt öffnet man eine Command Shell und wechselt in den neu angelegten Ordner. Um das Paket erfolgreich installieren zu können, braucht man Superuser-Rechte. Unter Ubuntu gibt man dazu den Befehl:

```
sudo -s
```

ein. Nach Eingabe des Superuser-Passworts kann man bluez-sdp mit der Befehlsfolge:

```
./configure
make install
make
```

installieren. Damit das Programm nach Bluetooth-Geräten suchen und sich mit diesen verbinden kann, muss der ausführende Rechner über ein Bluetooth-Modul verfügen.

### *Konfiguration*

Die Einstellungen für die Base Station werden über eine Konfigurationsdatei getroffen. In der Datei befinden sich nur die Parameter, die für eine Anmeldung am Server notwendig sind.

| **Parametername** | **Beschreibung** | **Defaultwert** |
|---|---|---|
| Position.X | X-Koordinate der Base Station | |
| Position.Y | Y-Koordinate der Base Station | |
| Position.Z | Z-Koordinate der Base Station | |
| ServerName | Name oder IP-Adresse des Rechners auf dem der BTLocationServer läuft. | |
| ServerPort | Port auf dem der BTLocationServer auf Anfragen der Base Stations wartet | 2181 |

Tabelle 11: Base Station-Konfigurationsdatei

Die Datei *BaseStation.ini* ist ein Beispiel für eine solche Datei. Man findet sie auf der CD im Ordner *Software\BaseStation*.

### Start

Wenn das Programm im Modus Real oder Measurement betrieben werden soll, muss es mit Superuser-Rechten ausgeführt werden. Unter Ubuntu gibt man dazu den Befehl:

```
sudo -s
```

ein. Für den Zugriff auf BlueZ wird die JTooth Library verwendet. Zu ihr gehören auch die Dateien *libjtooth.so* und *libbluetooth.so.1*. Beide findet man auf der CD im Ordner *Software\BaseStation\Libs*. Beim Start des Programms gibt man den Ordner, in den man die beiden Dateien kopiert hat, über den Parameter *java.library.path* an. Das Programm wird dann mit folgendem Kommando gestartet:

```
java -Djava.library.path=<Lib Ordner> -jar BaseStation.jar [Konfigu-
                                rationsdatei]
```

Man hat noch die Möglichkeit, eine Konfigurationsdatei anzugeben. Gibt man keine an, versucht das Programm standardmäßig die Datei *BaseStation.ini* zu laden. Befindet sich diese nicht im selben Verzeichnis wie die Datei *BaseStation.jar,* werden die Defaulteinstellungen verwendet. Die Datei *BaseStation.jar* findet man auf der CD im Ordner *Software\BaseStation*.

### Verwendung

Nach dem Start arbeitet das Programm selbständig. Wenn die Base Station vom BTLocationServer in den Modus Real oder Measurement versetzt wurde, verbindet sie sich mit ein oder mehreren Bluetooth-Geräten. Sie misst dann die Signalstärkeparameter der einzelnen Verbindungen und überträgt sie an den BTLocationServer. Arbeitet sie im Modus Development, sendet sie zufällig generierte Werte an den Server. Mit der Tastenkombination Strg+C kann die Anwendung beendet werden.

## 9.2 Weitere Programme

### 9.2.1 SearchFiles

Mit diesem Java Programm können Aufzeichnungen des BTLocationServers auf Vollständigkeit überprüft werden.

#### *Start*

Man startet das Programm von der Kommandozeile mit dem Befehl:

```
java searchfiles.SearchFiles [Ordner]
```

Als Parameter übergibt man den Ordner, in dem sich die Aufzeichnungen befinden. Man findet das Programm auf der CD im Ordner *Software\SearchFiles*.

#### *Verwendung*

Das Programm durchsucht dann die Dateien im Ordner und gibt am Bildschirm aus, in welchen Dateien Unregelmäßigkeiten gefunden wurden. Man kann die Länge der Aufzeichnungen überprüfen und so feststellen, ob die gewünschte Anzahl an Abtastungen der Messwerte durchgeführt wurde. Weiters kann festgestellt werden, ob sich in den Aufzeichnungen ein NoMeasurementValue befindet. Der BLocationServer schreibt das NoMeasurementValue für eine Base Station in die Aufzeichnungsdatei, wenn diese nicht mit dem MD verbunden war und somit keine Messwerte liefern konnte. Ist die Aufzeichnungsdatei lang genug und findet sich in ihr auch kein NoMeasurementValue, wurde sie erfolgreich durchgeführt. Einstellungen bezüglich der Überprüfung können über die in Tabelle 12 beschriebenen globalen Variablen getroffen werden.

| **Variable** | **Beschreibung** |
|---|---|
| bCheckLineAnz | Wenn true, wird die Zeilenanzahl überprüft |
| nAnzLines | Anzahl der Abtastungen |
| bCheck4QuestionMarks | Wenn true, wird nach NoMeasurementValues gesucht |

Tabelle 12: Einstellungsmöglichkeiten SearchFiles

### 9.2.2 StationMerge

Da die JTooth Bibliothek keinen Role Switch unterstützt, können sich nur zwei BS gleichzeitig zu einem MD verbinden. Will man jedoch die Signalstärkeparameter von mehr als zwei BS zu einem MD an einem Messpunkt aufzeichnen, müssen diese Aufnahmen nacheinander durchgeführt werden. So entstehen pro Messpunkt mehrere Dateien. Diese können mit dem Programm StationMerge zusammengeführt werden, sodass man pro Messpunkt eine Datei mit den errechneten Signalstärkewerten und eine mit allen Signalstärkeparametern hat.

#### *Start*

Das Programm kann direkt von der Kommandozeile mit dem Befehl:

```
java stationmerge.StationMerge [Quellordner] [Zielordner]
```

gestartet werden. Der erste Ordner enthält die Dateien, die bei den einzelnen Aufzeichnungen entstanden sind. Der zweite Ordner gibt an, wohin die zusammengeführten Dateien geschrieben werden sollen. Man findet das Programm auf der CD im Ordner *Software\StationMerge*.

#### *Verwendung*

Welche Dateien aus dem Quellordner zusammengeführt werden müssen, erkennt das Programm an den Dateiendungen. Im Rahmen dieser Arbeit wurden, an jedem Messpunkt zwei Aufzeichnungen gemacht. Die Dateinamen beider Aufzeichnungen beginnen mit der Position des Messpunktes. Der Dateiname der ersten Aufzeichnung endet auf _1_2.txt und wird im Programm über die globale Variable sENDLA angegeben. Der Dateiname der zweiten Aufzeichnung endet auf _3_4.txt und wird im Programm über die globale Variable sENDLB angegeben. Der Name der zusammengeführten Datei setzt sich aus der Position des Messpunktes und der Variable sMERGEENDL zusammen. So funktioniert das für die Dateien mit den errechneten Signalstärkewerten. Die Einstellungen für die Dateien mit allen Signalstärkeparametern werden über die Variablen sENDLRAWA, sENDLRAWB und sMERGERAWENDL getroffen.

| Variable | Beschreibung | Defaultwert |
|---|---|---|
| sENDLA | Endung der Datei der ersten Aufzeichnung mit den errechneten Signalstärkewerten | _1_2.txt |
| sENDLB | Endung der Datei der zweiten Aufzeichnung mit den errechneten Signalstärkewerten | _3_4.txt |
| sENDLRAWA | Endung der Datei der ersten Aufzeichnung mit allen Signalstärkeparametern | _1_2_raw.txt |
| sENDLRAWB | Endung der Datei der zweiten Aufzeichnung mit allen Signalstärkeparametern | _3_4_raw.txt |
| sMERGEENDL | Endung der zusammengeführten Datei mit den errechneten Signalstärkewerten | .txt |
| sMERGERAWENDL | Endung der zusammengeführten Datei mit allen Signalstärkeparametern | _raw.txt |

Tabelle 13: Einstellungsmöglichkeiten StationMerge

### 9.2.3 LearnFileMaker

Das Programm LearnFileMaker dient zur Auswahl von Trainings-, Test- und Validationsdaten aus einer Datenmenge.

#### *Start*

Das Programm besteht aus einer Klasse und hat keinerlei äußere Abhängigkeiten. Eine Installation und Konfiguration ist daher nicht notwendig, es kann sofort mit dem Befehl:

```
java learnfilemaker.LearnFileMaker [Ordner mit Datenmenge]
```

gestartet werden. Der Ordner, der über die Kommandozeile angegeben wird, beinhaltet die Datenmenge in Form mehrerer Dateien. Man findet das Programm auf der CD im Ordner *Software\LearnFileMaker*.

#### *Verwendung*

Zur Auswahl der Trainings-, Test- und Validationsdaten können einige Einstellungen über globale Variable direkt im Programmcode vorgenommen werden. Macht man von dieser Möglichkeit Gebrauch, muss das Programm natürlich neu kompiliert werden, bevor die Änderungen wirksam werden. In Tabelle 14 sind die Variablen beschrieben, über die man die Einstellungen treffen kann.

| Variable | Beschreibung |
|---|---|
| bGetSection | Wenn true, wird aus jeder Datei der Datenmenge der gleiche Bereich von Datensätzen ausgewählt. Der Bereich wird über die Variablen nFrom und nAnz angegeben. |
| bGetRandom | Wenn true, werden aus jeder Datei der Datenmenge die Datensätze zufällig ausgewählt. Wie viele Datensätze pro Datei ausgewählt werden, gibt man über nAnz an. |
| nFrom | Datensatz, bei dem der Auswahlbereich beginnt. Ist nur von Bedeutung, wenn bGetSection true ist. |
| nAnz | Anzahl der Datensätze, die ausgewählt werden sollen. |
| bMakeValidationSet | Wenn true, wird eine Validationsmenge gebildet |
| nValRatio | Gibt an, wie viele Prozent der ausgewählten Datensätze als Validationsdaten verwendet werden sollen |
| bMakeTestSet | Wenn true, wird eine Testmenge gebildet |
| nTestRatio | Gibt an, wie viele Prozent der ausgewählten Datensätze für die Testdaten verwendet werden sollen |

Tabelle 14: Einstellungsmöglichkeiten LearnFileMaker

## 9.3 Matlab Skripts

### 9.3.1 Glättung

Die Signalstärkewerte einer Verbindung zwischen BS und MD beginnen mit zunehmender Entfernung zu fluktuieren. Da für das Training mit KNNs eindeutigere Werte zu bevorzugen sind, wurden die gemessenen Daten geglättet, um so eine zweite Datenmenge mit eindeutigeren Werten zu generieren. Die Glättung wurde mit dem Skript *glaetten.m* im Programm Matlab durchgeführt.

#### *Konfiguration*

Das Skript *glaetten.m* und die Dateien mit den errechneten Signalstärkewerten, die zuvor durch das Programm *StationMerge* bearbeitet wurden, sodass pro Messpunkt nur mehr eine Datei vorliegt, werden in ein Arbeitsverzeichnis verschoben. Es befinden sich nur diese Dateien im Arbeitsverzeichnis. Als Arbeitsverzeichnis kann ein beliebiger Ordner gewählt werden. Im Arbeitsverzeichnis muss noch ein Unterverzeichnis mit dem Namen *output* vorhanden sein.

#### *Start*

Das Skript *glaetten.m* muss mit Matlab ausgeführt werden. Nachdem man Matlab gestartet hat, wählt man das Arbeitsverzeichnis als Workspace. Man führt dann das Skript aus, indem man

```
glaetten
```

im Kommandofenster eingibt und mit Eingabe bestätigt. Die Dateien mit den geglätteten Signalstärkewerten findet man im Ordner *output*. Man findet das Skript *glaetten.m* auf der CD im Ordner *Experimente\neuronaleNetze\Scripts*.

#### *Verwendung*

Wie stark die Werte geglättet werden sollen, kann man im Skript über die Variable *alpha* einstellen, indem man einen Wert zwischen 0 und 1 zuweist. 0 entspricht dabei der stärksten Glättung, bei 1 wird keine Glättung vorgenommen.

### 9.3.2 Training

Zum Trainieren der KNN wurde die Neural Network Toolbox von Matlab verwendet. Das Training wurde über das Skript *training.m* gesteuert.

#### *Konfiguration*

Das Skript *training.m* und die Dateien mit den Trainings-, Test- und Validationsdaten, die vom Programm *LearnFileMaker* erzeugt wurden, werden in ein Arbeitsverzeichnis verschoben. Als Arbeitsverzeichnis kann ein beliebiger Ordner gewählt werden.

#### *Start*

Das Skript *training.m* muss mit Matlab ausgeführt werden. Nachdem man Matlab gestartet hat, wählt man das Arbeitsverzeichnis als Workspace. Man führt dann das Skript aus, indem man

```
training
```

im Kommandofenster eingibt und mit Eingabe bestätigt. Man findet das Skript *training.m* auf der CD im Ordner *Experimente\neuronaleNetze\Scripts*.

### *Verwendung*

Nach dem Start erscheint ein Fenster, in dem der Lerngraph angezeigt wird. Das Training wird solange fortgesetzt, bis der gewünschte durchschnittliche Fehler erreicht wird oder das eingestellte Maximum an Lernzyklen überschritten wird. Beide Einstellungen können im kommentierten Skript *training.m* verändert werden. Man hat auch die Möglichkeit, das Training vorzeitig zu stoppen, indem man die Schaltfläche „Stop“ in der linken unteren Ecke im Fenster mit dem Lerngraphen betätigt. Nachdem das Training beendet wurde, speichert das Skript das trainierte Netz im Arbeitsverzeichnis.

# Literaturverzeichnis

[VAR01]
Clemens Vargas R., Gerhard Franken; Bluettoth, 2001

[BRA02]
Bray Jennifer, Charles Sturman, Bluetooth 1.1: connect without cables, 2nd Edition, 2002, ISBN 0-13-066106-6

[LÜD07]
Christian Lüders, Lokale Funknetze Wireless LANs (IEE 802.11), Bluetooth, DECT, 1. Auflage, 2007, ISBN 978-3-8343-3018-5

[BTC20]
Bluetooth Core Specification v2.0 + EDR
http://bluetooth.com/Bluetooth/Learn/Technology/Specifications/
Stand 22.11.2007

[BTC21]
Bluetooth Core Specification v2.1 + EDR
http://bluetooth.com/Bluetooth/Learn/Technology/Specifications/
Stand 22.11.2007

[MDN07]
Microsoft Developer Network Library Bluetooth,
http://msdn2.microsoft.com/en-us/library/aa362932.aspx
Stand 11.12.2007

[FRE07]
FreeBT - Open Source Bluetooth for the Windows Platform,
http://www.freebt.net/
Stand 11.12.2007

[BLZ07]
BlueZ Official Linux Bluetooth protocol stack,
http://www.bluez.org/
Stand 11.12.2007

[MIS04]
Stefan Mischke, Konzeption und Implementierung einer Java-API zur Bluetooth-Kommunikation am Beispiel eines Ortsbestimmungsdienstes, Dezember 2004, Paderborn,
http://gauge.uni-paderborn.de/jtooth/JTooth.pdf
Stand 14.12.2007

[JTH07]
JTooth,
http://gauge.uni-paderborn.de/jtooth/
Stand 14.12.2007

[ANT03]
Antti Kotanen, Marko Hännikäinen, Helena Leppäkoski, Timo D. Hämäläinen, Experiments on Local Positioning with Bluetooth, proc. of International Conference on Information Technology: Computers and Communications (ITCC'03), 28-30 April 2003, Pages 297-303

[JAS05]
Jason Yipin Ye, Atlantis: Location Based Services with Bluetooth, 2005, Department of Computer Science, Brown University

[ALE05]
Alessandro Genco, Salvatore Sorce, Giuseppe Scelfo, Bluetooth Base Station Minimal Deployment for High Definition Positioning, 2005, DINFO - Dipartimento di Ingegneria Informatica, Università di Palermo - Italy

[LUD04]
Ludger Lecke, Positionierung von Mobiltelefonen, 2004, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn

[DAN96]
Dan Patterson, Künstliche neuronale Netze, 2. Auflage, 1996, ISBN 3-8272-9531-9

[TOM97]
Tom M. Mitchell, Machine Learning, International Edition, 1997, ISBN 0-07-115467-1

[ROB99]
Robert Callan, Neuronale Netze im Klartext, 1st Edition, 1999, Translation Copyright 2002, ISBN 3-8273-7071-X

[NNE07]
Neuronale Netze eine Einführung,
http://www.neuronalesnetz.de/downloads/neuronalesnetz_de.pdf
Stand 20.12.2007

[NNT07]
Neural Network Toolbox User's Guide,
http://www.mathworks.com/access/helpdesk/help/pdf_doc/nnet/nnet.pdf
Stand 20.12.2007