Georg-August-Universität Göttingen
Zentrum für Angewandte Informatik

ISSN 1612-6793
ZAI-BSC-2015-22

Bachlelorarbeit

im Studiengang Angewandte Informatik

Entwicklung einer Android-Applikation zur Indoor-Lokalisierung über WiFi-Netzwerke als Prototyp für die SUB Göttingen

---

Lukas Hoffmann

Am
Geographischen Institut
Abteilung Kartographie, GIS & Fernerkundung

Bachelor- und Masterarbeiten
des Zentrums für Angewandte Informatik
an der Georg-August-Universität Göttingen

3. Dezember 2015

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

1 Einleitung

Es existieren verschiedene Möglichkeiten, die Position eines mobilen Gerätes außerhalb von Gebäuden zu orten. Die bekannteste Methode ist das „Global Positioning System“ (GPS). Der Empfang von GPS ist infolge auftretender Abschirmung innerhalb von Gebäuden meist reduziert bis gar nicht vorhanden. Was tut aber ein Tourist, der spät am Flughafen eintrifft und schnell das richtige Terminal finden muss? In dieser Situation wäre es von Vorteil, wenn mit Hilfe eines mobilen Geräts eine Route zum Terminal berechnet werden könnte, die dann dem Nutzer in Echtzeit präsentiert wird. Es existieren also Einsatzbereiche, in denen Ortung und Navigation innerhalb von Gebäuden gefragt ist. Eine mögliche Lösung zur Positionsbestimmung stellt die Lokalisation über Drahtlosnetzwerke dar. Da Wireless-LAN-Access-Points in großen Gebäuden zahlreich vorhanden sind, kann eine Ortung der Position über die unterschiedlichen Signalstärken stattfinden.
Ziel der vorliegenden Bachelorarbeit ist es, einen Prototyp in Form einer Applikation für das Betriebssystem Android zu entwickeln, der es ermöglichen soll, mithilfe von Wireless-LAN und einem Kartensystem eine Datenbank aufzubauen. Über diese werden dann im nächsten Schritt Berechnungen zur Bestimmung der aktuellen Position in der niedersächsischen Staats- und Universitätsbibliothek Göttingen ausgeführt.

2 Technische Grundlagen

In diesem Kapitel werden die theoretischen Prinzipien und Begrifflichkeiten erläutert, die als Grundlage für die folgenden Kapitel dienen sollen.

2.1 Lokalisieren über Drahtlosnetzwerke

Da Wireless-LAN-Technologie heutzutage in fast jedem Gebäude zum Einsatz kommt, kann diese weiträumige Ausbreitung von Drahtlosnetzwerken genutzt werden, um mobile Geräte zu lokalisieren. Die sogenannten Wireless-LAN-Positionssysteme können somit, ohne weiteres Installieren von Software oder Manipulation der Hardware, die Position von nahezu jedem mobilen Gerät orten, das Wifi-kompatibel ist. Lokalisierung über Drahtlosnetzwerke erlaubt ebenfalls den Gebrauch von *location-based-services*. Diese Services nutzen die Positionsdaten, um Benutzern Informationen zu liefern, die auf den bestimmten Standort zugeschnitten sind. Nützliche Einsatzbereiche sind beispielsweise die Navigation in einem Einkaufszentrum oder das Wiederfinden eines verlorenen Kindes innerhalb von großen

Gebäuden. Verloren gegangene Gegenstände können auch mit dieser Technologie wiedergefunden werden (vgl. HENNIGES 2012).

Wireless-LAN-Positionssysteme kommen immer mehr zum Einsatz. Google veröffentlichte 2013 die „*Google Maps Floor Plan Maker*"-Applikation für das Betriebssystem Android. Benutzer dieser Applikation werden angewiesen, Punkte auf bestimmten Karten abzulaufen, während Drahtlosnetzwerke in der Umgebung gescannt werden. Die Ergebnisse werden im Anschluss an Google gesendet. Die Applikation arbeitet Hand in Hand mit *Google Maps Floor Plans*, wo Benutzer Bilder von Grundrissen hochladen können. Mithilfe dieser Technologie ist eine Positionierung innerhalb von tausenden Gebäuden möglich, die über den ganzen Globus verteilt sind. Als Beispiele können der Frankfurter Flughafen, das MGM Resorts Casino oder der Tokyo Tower aufgelistet werden (vgl. GOOGLE 2015).

Ein weiteres Projekt ist das OpenWLANMap, welches ebenfalls frei nutzbare WLAN-basierte Positionsdaten sammelt. Jeder Nutzer kann ähnlich wie beim „*Google Maps Floor Plan Maker*" mit einer Applikation Daten sammeln und einsenden. Das Projekt soll eine Alternative zu kommerziellen Angeboten darstellen und die Nutzer können sich entscheiden, ob ihre Daten mit der Öffentlichkeit geteilt werden sollen (vgl. OPENWLANMAP 2015).

2.2 Fingerprint Lokalisierung

Eine Lokalisierung in GPS-schwachen Bereichen wird durch Techniken erreicht, die die Position eines mobilen Gerätes basierend auf der Stärke von Funksignalen zu bekannten *Wifi-Access-Points*[1] (drahtlosen Zugangspunkten) schätzen. Auch wenn diese Techniken es möglich machen, eine ungefähre Positionsschätzung zu geben, so treten Probleme auf, wenn die Etage oder der Raum identifiziert werden soll, in der sich das Gerät befindet. Dies hat den Grund, dass Funksignale innerhalb von Gebäuden von Möbeln und Wänden absorbiert, gestreut und reflektiert werden. Eine Lösung für dieses Problem ist die sogenannte *Fingerprint*-Technik.

Sie wird in zwei Phasen unterteilt. In der ersten, der sogenannten offline-Phase, wird eine Datenbank aufgebaut, die zu jedem ausgewählten Punkt in einer Karte mit zugehörigem Koordinatensystem die *received signal strength*[2] (empfangenen Signalstärken) zu allen erreichbaren APs speichert. Der Datensatz aus der Position und den von ihr aus erreichbaren APs mit den dazugehörigen RSS wird *Fingerprint* genannt. Die RSS ist die Messung eines mobilen Geräts, die die Stärke von jedem eingehenden Paket in Dezibel angibt (vgl. FARID et.

[1] Im Folgenden abgekürzt als AP
[2] Im Folgenden abgekürzt als RSS

al. 2014). Da diese zu- bzw. abnimmt, je nachdem wie nah oder fern sich der Empfänger vom jeweiligen AP aufhält, steht die RSS in starker Korrelation zur Distanz. Die Aufnahme der Datenbank, im Folgenden als *radio map* bezeichnet, erfolgt inkrementell, indem der Benutzer seine aktuelle Position markiert und zusammen mit den aktuell gemessenen Signalstärken abspeichert. Dies wird dann für verschiedene Positionen wiederholt, bis das gewünschte Areal abgedeckt ist.

Wenn die offline-Phase erfolgreich abgeschlossen ist, folgt die online-Phase. Um die Position eines mobilen Geräts zu bestimmen, werden die aktuellen RSS zu den APs mit denen in der *radio map* verglichen. Hierbei wird ein Algorithmus verwendet, der die Position auswählt, deren Signalstärken am ehesten den abgespeicherten entsprechen.

Die Lagerung der Daten und die Berechnung der Position kann client- oder serverbasiert realisiert werden. Bei einem serverbasiertem System wird in der offline-Phase die, mit dem mobilen Gerät aufgenommene, *radio map* an den Server gesendet. In der online-Phase folgen dann ausschließlich die aktuellen Messwerte der erreichten APs und nach der Ausführung des Algorithmus wird die berechnete Position zurück an den Client geschickt (vgl. Abb.1). Bei einem client-basiertem System (vgl. Abb.2) werden die aufgenommenen Positionen mit ihren Signalstärken im internen Speicher des Gerätes gelagert und ebenfalls wieder abgerufen, wenn die aktuelle Position bestimmt werden soll. Online- und offline-Phase finden demzufolge beide auf dem mobilen Gerät statt (vgl. HANSEN et. al. 2014).

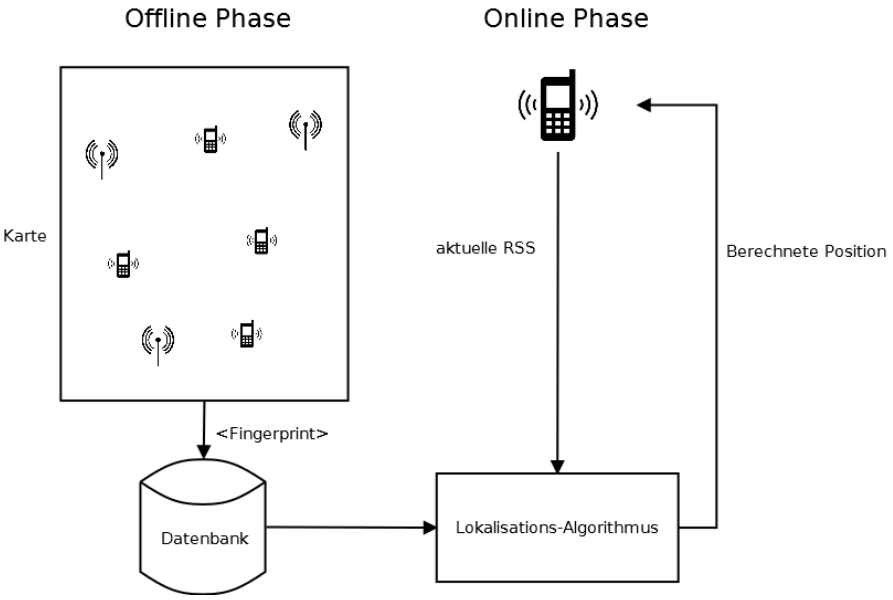

Abbildung 1: Schema einer serverbasierten Umsetzung der Fingerprint-Technik.

Quelle: Eigene Darstellung, nach HE, SUINING / GARY CHAN (2015), S.2.

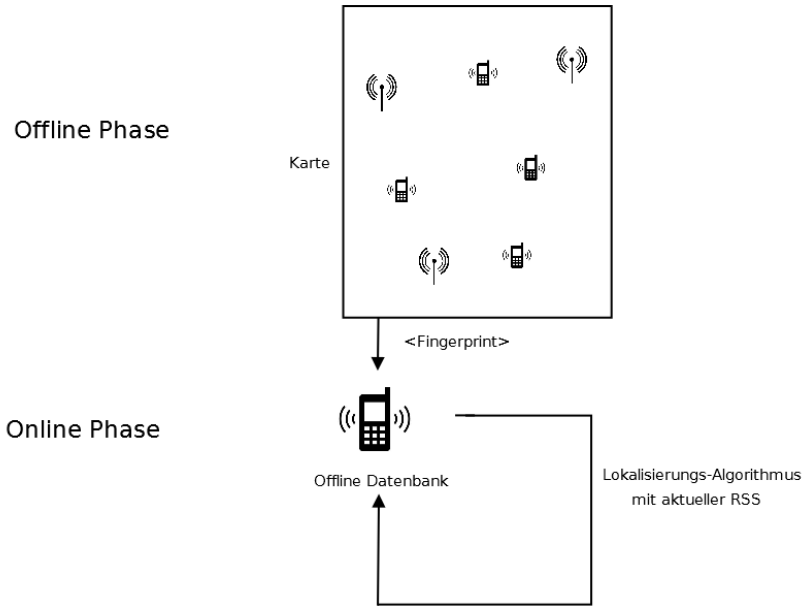

Abbildung 2: Schema einer gerätebasierten Umsetzung der Fingerprint-Technik

Quelle: Eigene Darstellung, nach HE, SUINING / GARY CHAN (2015), S.2.

2.3 Webservices und REST

Damit ein Zugriff auf die Informationen eines Webservers möglich ist, muss ein Webservice auf ihm integriert sein. Webservices bieten anderen Programmen eine Webschnittstelle an, mithilfe derer der Client Anfragen an den Service stellen kann.

Representational State Transfer (REST) ist eine einfache Technologie, über die ein Webservice aufgebaut werden kann. Der Grundgedanke hinter REST ist, statt komplexe Mechanismen zur Verbindung zwischen Computern einzusetzen, das einfache *Hypertext Transfer Protocol* für die Kommunikation zu nutzen. Die HTTP-Methoden dienen als Interface, um einen Zugang zu den auf dem Server liegenden Ressourcen zu ermöglichen. Jede Ressource wird durch eine einzigartige *Uniform resource locator*[3] repräsentiert. Da das World Wide Web selbst auf HTTP basiert, kann es auch als eine REST-basierte Architektur angesehen werden (vgl. ELKSTEIN 2015).

[3] Im Folgenden abgekürzt als URL

2.4 ArcGIS

ArcGIS ist ein von der Firma ESRI entwickeltes geografisches Informationssystem, das zur Erstellung und Nutzung von Karten verwendet wird. Weiterhin werden viele Funktionen zur Kompilierung von geografischen Daten, zur Analyse von kartenbezogenen Informationen und zum Datenbankenmanagement zur Verfügung gestellt. Neben einfachen Desktop-Anwendungsmöglichkeiten ist es mit ArcGIS möglich, Karten und geografische Informationen für eine Organisation, eine spezielle Gemeinschaft oder offen im Internet verfügbar zu machen (vgl. ESRI 2015$_a$).

2.4.1 ArcGIS Server und Web-Layer

Teil der genannten Infrastruktur ist das serverbasierte Geoinformationssystem ArcGIS Server, welches drei Teile umfasst. Als erstes werden Geoinformations- und Karten-Services bereitgestellt, als zweites standardbasierte Web-Service-Schnittstellen für den Zugriff auf die Benutzung von GIS-Services und als letztes eine leistungsstarke Geodatabase-Verwaltung in verschiedenen Datenbank-Managementsystemen wie zum Beispiel Oracle oder PostgreSQL.
Geoinformations-Services sind spezielle Webservices, die den Zugriff auf geografische Informationen ermöglichen. Werden GIS-Services von einem ArcGIS-Server bereitgestellt, dann repräsentiert ein Service immer genau eine Ressource, wie beispielsweise eine Karte oder eine Datenbankverbindung. Services mit den dazugehörigen Ressourcen werden auf der Serverseite gelagert, um sie so für Anwendungen der Client-Seite verfügbar zu machen. Ist die Geoinformationsarbeit erledigt, sendet der Server das Ergebnis in einem üblichen Format, wie zum Beispiel in Form eines Textes oder Bildes, zurück an den Client (vgl. ESRI 2015$_b$).
Wenn ein GIS-Service eine Karte als Ressource repräsentiert, wird von einem Karten-Service gesprochen. Er ermöglicht den Nutzern Zugriff auf die Inhalte einer Karte. Mithilfe von ArcGIS Online können Karten und Daten als gehostete *Web-Layer* veröffentlicht werden, auf die dann unter anderem auch mobile Applikationen zugreifen können (vgl. ESRI 2013).
Web-Layer sind für die Visualisierung, Bearbeitung und Abfrage von Karten konzipiert und unterteilen sich in zwei Typen: den Feature-Layern und den Raster-Layern.
Feature-Layer lagern geografische Daten als Formen oder Vektoren, die auch als *Features* bezeichnet werden. Jedes *Feature* beinhaltet geometrische Informationen und spezielle Attribute, die in die Karte gezeichnet werden können. Daher eignen sie sich am besten für die Visualisierung von Daten, die über ihren Grundkarten liegen. Raster-Layer speichern die Daten in Form von Bildern und unterstützen eine schnelle Visualisierung von Karten mit

vorab gezeichneter Kartenbilder bzw. Kacheln. Für beide Layer-Typen werden Karten-Services bereitgestellt (vgl. ESRI 2015c).

ArcGIS Server nutzt für alle Services Standardwebschnittstellen und unterstützt auch das in Kapitel 2.2 beschriebene REST. So kann auf jede Ressource, die auf dem ArcGIS-Server liegt, über eine URL in der Form `http://<Hostname>/arcgis/rest/<Service-Name>` zugegriffen werden. Von dieser Basis-URL kann zu jedem Service, jeder Ressource und zu jeder, zu den Services gehörigen, Operationen weiter navigiert werden (vgl. ESRI 2014).

2.4.2 Anfragen an die Web-Layer

Feature-Layer unterstützen Suchanfragen an ihre *Features* mittels eines *QueryTasks*. Dieser wird mit der URL des gewünschten Layers aufgebaut, um ihm anschließend Parameter für die Suchanfrage zu übergeben. Danach kann die Suchanfrage ausgeführt werden und wenn sie erfolgreich war, werden die Ergebnisse in einem Datensatz bereitgestellt. Einige Eigenschaften der Suchparameter basieren auf einfachen SQL-Klauseln. Welche Attribute der Ergebnisse weiter verwendet werden sollen, kann mit *outfields* eingeschränkt werden und an welche Bedingungen die Suchanfrage gebunden ist, wird mit dem Setzen der Eigenschaft *where* erzielt. Weiterhin ist noch die Eigenschaft *geometry* vorhanden, mit der geometrische Eingrenzungen auf der Karte möglich sind (vgl. ESRI 2015d).

3 Konzept der entwickelten Applikation

Es werden im Folgenden die Anforderungen an die Applikation vorgestellt, um dann die Wahl des SDKs und den Aufbau der App zu erläutern und zu begründen.

3.1 Spezifikation der Anforderungen

Die, im Rahmen der vorliegenden Bachelorarbeit entwickelte, Applikation soll eine Lokalisierung innerhalb von Gebäuden des gesamten Geländes der Georg-August Universität Göttingen möglich machen. Im Besonderen in der Niedersächsischen Staats- und Universitätsbibliothek Göttingen, da dort die App getestet werden soll. Um dies zu erreichen und weil Vorhandenes genutzt werden soll, wird die Fingerprint-Technik verwendet und mit dem Kartensystem des ArcGIS-Servers der Universität verknüpft. Dieser bietet Karten und Feature-Layer zu allen Gebäuden und Räumen der Universität an (vgl. Kap. 2.4.1). Für die App ist ebenfalls ein Zugriff auf die Drahtlosfunktionen des mobilen Gerätes erforderlich, weil die Signalstärken zu allen erreichbaren APs als Daten verwendet werden müssen. Da

nicht alle APs verwendet werden sollen, sondern nur die der Universität Göttingen, muss ein Filter integriert sein. Diese Entscheidung war nötig, da Drahtlosnetzwerke nur temporär verfügbar seien können, wie zum Beispiel ein mobiler WLAN-Hotspot eines Smartphones. Wenn diese Netzwerke bei der Berechnung der Position wegfallen, kann das Ergebnis verfälscht werden. Als eine weitere Anforderung kann die Navigation aufgeführt werden. Es sollten bestimmte Gebäude auswählbar sein, in denen dann die aktuelle Position markiert werden kann, damit diese mit den aktuellen RSS zu allen APs gespeichert werden kann. Die so aufgebaute *radio map* sollte um Punkte erweiterbar sein und es sollte eine Lösch-Funktion existierten, falls eine *radio map* aktualisiert werden muss oder falls in der Aufnahme ein Fehler gemacht wurde. Das Abspeichern von Mittelwerten von Signalstärken zu bestimmten Positionen soll unterstützt werden, um in einer späteren Analyse testen zu können, ob dies eine Verbesserung in der Genauigkeit ergibt. Die interne Datenstruktur muss so aufgebaut werden, dass ein Algorithmus diese verwenden kann, um mit den aktuellen Signalstärken die aktuelle Position zu bestimmen. Weiterhin sollten die aufgenommenen Daten exportiert werden können, um diese analysieren zu können.

3.2 Wahl der SDK und Entwicklungsumgebung

Da die ArcGIS-Serverdaten möglichst einfach in die Applikation integriert werden sollen, fiel die Wahl des SDKs auf das *ArcGIS Runtime SDK for Android*. Außerdem bietet das mobile Betriebssystem Android den für die App notwendigen Zugriff auf die Drahtlosdaten des mobilen Gerätes an. Eine Umsetzung auf dem iPhone ist aus diesem Grund nicht möglich, da Apple bisher keine öffentliche API für das Empfangen von Drahtlosdaten bereitgestellt hat (vgl. HANSEN et. al. 2015).
Android-Applikationen werden in der Programmiersprache Java geschrieben, weshalb das *Android Software Development Kit* verwendet wurde. Die benutzte Version 10.2.6 des *ArcGIS Runtime SDK for Android* ist für die Entwicklungsumgebung Android Studio ausgelegt. Deshalb fand die Entwicklung der Android-Applikation in Android Studio 1.2.2 statt. Über den SDK-Manager wurde zusätzlich die Android-Plattform 5.1.1 (API-Level 22) installiert. Das Projekt zur Grundlage der Android-App bekam den Namen „GoeRadioMap“.

Das Testen und Entwickeln von Android-Applikationen ist grundsätzlich ohne die Verwendung eines Android-Gerätes möglich. Es besteht die Möglichkeit über den, in der Android-SDK enthaltenen, *Android Virtual Device Manager* ein Gerät zu emulieren, auf dem dann die App kompiliert und ausgeführt werden kann. „GoeRadioMap“ wurde jedoch auf

dem Gerät „ASUS MeMO Pad HD 7“ mit der Modellnummer ME173X entwickelt und getestet.

3.3 Aufbau und Funktionen

An dieser Stelle wird zuerst ein Überblick über den Aufbau der Applikation gegeben, um dann im weiteren Verlauf dieses Kapitels auf die einzelnen Teilstücke im Detail einzugehen. „GoeRadioMap“ besteht im Wesentlichen aus zwei großen Komponenten. Die erste ist die ArcGIS-Karte, die zur Navigation zu einem Gebäude oder Raum mit anschließender Positionsmarkierung dient. Die zweite Komponente ist die Aufnahmestruktur der Signalstärken zu allen erreichbaren APs. Mit beiden Komponenten werden dann *Fingerprints* erstellt und in einer passenden Datenstruktur auf dem mobilen Gerät gespeichert. Die, für die Applikation grundlegende, Architektur ist schlussfolgernd eine gerätebasierte Umsetzung der Fingerprint-Technik (vgl. Kap. 2.1). Das Abspeichern und weitere Funktionalitäten zur Bedienung werden über das Menü ausgeführt, welches sich in der oberen rechten Ecke des Bildschirms befindet. Zur Hilfe in der Navigation ist ein Fenster angelegt worden, dass Informationen zu dem aktuell ausgewählten Gebäude bzw. Raum bereitstellt. Damit der Benutzer der Applikation einen Überblick über die momentan vom mobilen Gerät erreichbaren APs bekommt, werden diese in einem weiteren Fenster mit ihren Signalstärken aufgelistet. Die Liste wird dann im Sekundentakt aktualisiert. Das komplette Layout ist für den Portrait-Modus optimiert und daher wurde das Wechseln in den *Landscape*-Modus deaktiviert.

3.3.1 Organisation der ArcGIS-Server-Daten

In „GoeRadioMap“ wird auf vier Ressourcen des ArcGIS-Servers mithilfe der zugrundeliegenden REST-API zugegriffen. Die erste Ressource ist die *basemap*, die die Grundkarte von der Stadt Göttingen mit allen Gebäuden der Universität beinhaltet. Sie besteht aus gekachelten Kartenschnitten und wird über die URL `http://134.76.20.130/arcgis/rest/services/Basemap/Basemap_9/MapServer` angesprochen. Die anderen Ressourcen sind drei Feature-Layer. Der erste enthält die Gebäudeinformationen, der zweite die Rauminformationen und der dritte zeigt Wände, Möbel, Säulen etc. an. Sie können über die folgenden URLs erreicht werden:

Feature-Layer	URL
Gebäude	`http://134.76.20.130/arcgis/rest/services/Lageplan/Raeume_Buildings_FS/FeatureServer/2`
Räume	`http://134.76.20.130/arcgis/rest/services/Lageplan/Raeume_Buildings_FS/FeatureServer/1`
Möbel, Säulen, Wände etc.	`http://134.76.20.130/arcgis/rest/services/Lageplan/Raeume_Buildings_FS/FeatureServer/0`

Tabelle 1: URLs der Feature-Layer des ArcGIS-Servers

Die relevanten Attribute der *Features* für die Android-Applikation werden in den folgenden Tabellen mit einer Beschreibung und ihrem Datentyp dargestellt. Über den eindeutigen Namen der Attribute können, wie in Kapitel 2.3.2 beschrieben, Anfragen an die Layer gestellt werden.

Attribute	**Beschreibung**	**Datentyp**
piz	Eindeutige ID für ein Gebäude	String
ug	Anzahl der Untergeschosse	Double
og	Anzahl der Obergeschosse	Double
geb_bez	Gebäudebeschreibung	String

Tabelle 2: Relevante Attribute des Feature-Layers für Gebäude

Attribute	**Beschreibung**	**Datentyp**
gebid	Eindeutige ID für das Gebäude eines Raumes	String
raumnr	Raumnummer	String
raumname	Raumbezeichnung	String
raumart	Art der Raumnutzung	String

Tabelle 3: Relevante Attribute des Feature-Layers für Räume

Attribute	**Beschreibung**	**Datentyp**
gebid	Eindeutige ID für das Gebäude der Grundskizze	String
etage	Etage der Grundskizze	String

Tabelle 4: Relevante Attribute des Feature-Layers für die Grundskizzen

Anzumerken ist, dass das Attribut *piz* des Layers für Gebäude dem Feld *gebid* aus für die Räume entspricht.

3.3.2 Benutzung und Navigation

Die Auswahl eines Gebäudes auf der Karte erfolgt über ein Antippen auf dem Touchscreen. Anschließend werden alle Räume der ausgesuchten Etage präsentiert. Die standardmäßige Selektion des Erdgeschosses kann über ein Auswahlmenü im oberen rechten Eck der Karte verändert werden. Jedes weitere Antippen innerhalb des Gebäudes bewirkt, dass an genau dieser Stelle ein roter Punkt erscheint und ein davor gesetzter verschwindet. Zusätzlich wird der Raum, in der sich die markierte Position befindet, hellblau eingefärbt und die Raumnummer, die Raumart und der Gebäudename in dem Informationsfenster angezeigt. Der Benutzer der App kann so die exakte Position angeben, an der er sich befindet und diese so lange in der Karte korrigieren, bis die gewünschte Lage ausgewählt ist.

Abbildung 3: Screenshot von GoeRadioMap mit Raumauswahl

Danach kann der Menüpunkt „Aktuellen Fingerprint speichern“ ausgewählt werden, um die Position mit den aktuell gemessenen Signalstärken zu speichern. Eine Benachrichtigung erscheint, um dem Benutzer zu signalisieren, dass die Aufnahme erfolgreich war. Es existiert kein Limit in der Datenerhebung, d. h., es können so viele Punkte aufgenommen werden, wie es der Benutzer für angemessen erachtet und diese pro Etage und Gebäude hinzugefügt werden. Wird die Position nicht geändert und ein weiterer Fingerprint gespeichert, wird dessen Signalstärke mit den vorher gelagerten gemittelt. Somit wird dem Benutzer auf einfachem Wege möglich gemacht, sich zwischen den beiden Aufnahmemöglichkeiten zu

entscheiden. Zur Übersichtlichkeit werden die Positionen als blaue Punkte angezeigt, die bereits mit ihren Messwerten gespeichert wurden. Der zweite Unterpunkt im Menü hat die Funktion, die Aufnahme der aktuell ausgewählten Etage zu löschen. Auch hier wird dem Benutzer durch eine Nachricht zu verstehen gegeben, dass das Löschen erfolgreich war und die Selektion des Gebäudes wird aufgehoben, sodass ein neues ausgewählt werden muss. Ein Wechseln des Gebäudes ohne vorheriges Löschen ist mit dem dritten Unterpunkt des Menüs möglich. Nachdem dieser gedrückt wurde, verschwindet die alte Auswahl und es kann anschließend eine andere universitäre Einrichtung angetippt und somit selektiert werden. Wenn die Aufnahme einer Etage abgeschlossen ist, kann zu testzwecken ebenfalls über das Menü der best-match-Algorithmus angewendet werden. Dieser berechnet auf Grundlage der vorher aufgenommenen Daten und den aktuellen Signalstärken als Eingabewert die gegenwärtige Position des mobilen Gerätes und färbt diese grün ein. Als letzte Funktionalität bietet „GoeRadioMap“ den Export aufgenommener Daten zur momentan ausgewählten Etage an. In dem Ordner /storage/emulated/0 wird eine csv-Datei[4] mit einem geeignetem Namen angelegt, die alle gespeicherten Daten enthält.

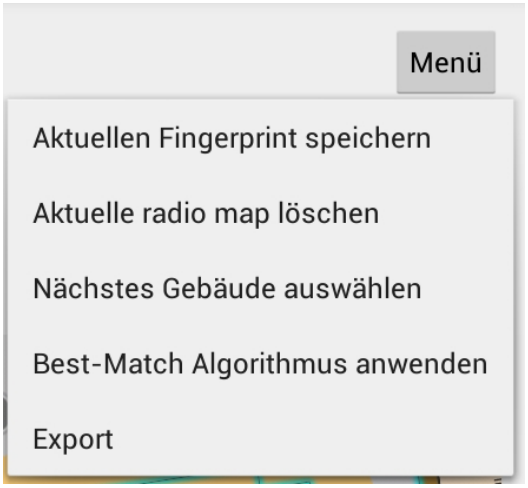

Abbildung 4: Menü von GoeRadioMap

3.3.3 Logischer Aufbau der internen Datenstruktur

Die interne Datenstruktur wird aus den vier Bestandteilen Position, *Access-Points*, Messungen der Signalstärke und der *radio map* zusammengesetzt. Die Position umfasst die Koordinaten x und y der ArcGIS-Basemap, welche ein zweidimensionales Netz aufbauen. Zusätzlich wird die Etage gespeichert, aus der sich ein dreidimensionales Koordinatensystem ergibt. Die APs

[4] csv-Dateien sind „Komma“ separierte Dateien, die für den Gebrauch in Excel optimiert sind.

werden durch die MAC-Adresse[5] eindeutig beschrieben und besitzen einen Namen. Messungen der Signalstärken sind die RSS zu einem bestimmten AP und die *radio map* vereinigt all diese Strukturen in einem logischen Kontext. Der Aufbau dieser Struktur kann in Tabellenform angegeben werden. Im folgenden Beispiel sind an sechs Positionen Messungen zu fünf verschiedenen APs aufgenommen worden (vgl. Abb.5). Die daraus resultierende Tabelle (vgl. Tab.5) weist verschiedene Charakteristika auf.

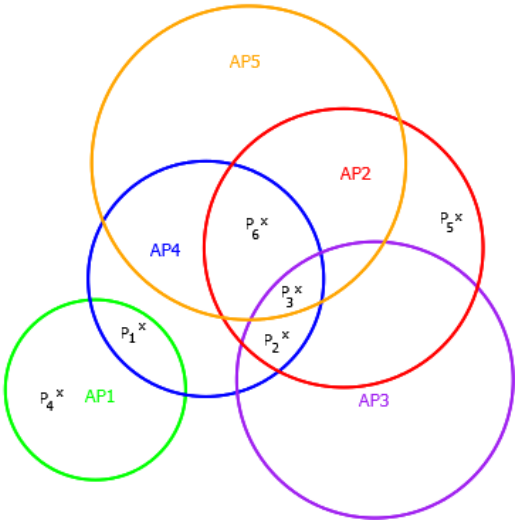

Abbildung 5: Schema zur Aufnahme von sechs Positionen zu fünf APs

Access Points

Position	AP_1	AP_2	AP_3	AP_4	AP_5
P_1(x,y,etage)	M_{11}(RSS)			M_{41}(RSS)	
P_2(x,y,etage)		M_{22}(RSS)	M_{32}(RSS)	M_{42}(RSS)	
P_3(x,y,etage)		M_{23}(RSS)	M_{33}(RSS)	M_{43}(RSS)	M_{53}(RSS)
P_4(x,y,etage)	M_{14}(RSS)				
P_5(x,y,etage)		M_{25}(RSS)			
P_6(x,y,etage)		M_{26}(RSS)		M_{46}(RSS)	M_{56}(RSS)

Tabelle 5: Logischer Aufbau in Form einer Tabelle der internen Datenstruktur

Von einer Position können mehrere APs erreicht werden, zu denen jeweils ein Signalstärkenmesswert existiert. Es kann vorkommen, dass von einer Position einer oder mehrere APs nicht erreicht werden. In diesem Fall bleibt der Eintrag leer. Wenn eine Aufnahme gemacht wird und die Position noch nicht vorhanden ist, werden alle Messungen

[5] „Die MAC-Adresse, auch als Ethernet-Adresse bezeichnet, ist eine eindeutige, unverwechselbare Adresse für alle Workstations, Server oder Rechner, die über Ethernet-Adapterkarten an ein Ethernet angeschlossen sind. Es handelt sich um eine 48 Bit lange Hardwareadresse, die zur eindeutigen Identifikation eines Knotens im Netzwerk dient“ (ITWISSEN 2015).

von der besagten Position zu den entsprechenden APs gespeichert. Falls die exakte Position als Eintrag schon vorhanden ist, wird aus den neuen Messwerten und den bereits gespeicherten der Mittelwert gebildet und falls ein neuer AP erreicht wird, wird dieser mit den dazugehörigen RSS hinzugefügt. Um beliebig viele Messungen zu einer Position zu mitteln, wird die Anzahl der bisher berechneten Mittelwerte in einer Variablen gespeichert, die in der folgenden Formel den Buchstaben „n“ zugewiesen bekommt.

$$RSS_{new} = \frac{RSS_{old} \times n + RSS_{aktuell}}{n+1}$$

Abbildung 6: Berechnung des Mittelwerts der gespeicherten Signalstärken

Die neue Signalstärke ergibt sich dann aus n-mal der zuletzt gespeicherten Stärke addiert mit der aktuell gemessenen. Das Ergebnis wird durch n plus eins dividiert, da inklusive der aktuellen Messung genau n plus eins Messungen durchgeführt wurden.

3.3.4 Speicherverwaltung der Applikation

Die Speicherverwaltung der aufgenommenen *Fingerprints* ist filebasiert, d.h., für jede Etage eines Gebäudes wird eine neue Datei angelegt, sobald der erste *Fingerprint* in ihr aufgenommen wird. Diese Datei erhält einen eindeutigen Namen und wird im internen Speicher des mobilen Gerätes gelagert. Das Hinzufügen eines *Fingerprints* ist immer mit einem vorherigen Laden der entsprechenden Datei verbunden, um so sicherzustellen, dass das Speichern an der richtigen Stelle geschieht. Dieser Aufbau hat zur Folge, dass der Benutzer für eine spätere Ortung die Etage, auf der er sich befindet, angeben muss. Das Löschen einer Datei kann über das Menü ausgeführt werden. Alle angelegten Dateien werden ebenfalls gelöscht, wenn die Applikation deinstalliert wird. Zu jeder Zeit der Aufnahme in einer bestimmten Etage können die Daten, wie bereits erwähnt, exportiert werden. Der Name der so gespeicherten csv-Datei setzt sich aus der Gebäude-ID, der Etage und dem aktuellen Datum und der Zeit zusammen. Ist die Niedersächsische Staats- und Universitätsbibliothek Göttingen im Erdgeschoss ausgewählt und wird die Datei am 24.09.2015 um 12:28:00 Uhr exportiert, dann hat die Datei den Namen 5383_EG_24092015at122800.csv. Diese Konstruktion ermöglicht eine genaue Zuordnung und erleichtert die Auswertung. In Abb. 7 ist der Inhalt einer solchen Datei einzusehen. Die Aufteilung ist der Tabelle 5 der internen Datenstruktur nachempfunden.

	A	B	C	D	E	F	G	H	I
1		Position		GoeMobile	GoeMobile	eduroam	eduroam	GoeMobile	GoeMobile
2	x	y	etage	28:8a:1c:22:12:c4	00:26:3e:64:c6:48	00:0b:0e:4	00:26:3e:6	00:0b:0e:c4	00:0b:0e:5a
3	11.060.617.521.182.600	67.172.985.574.408.200	2.OG	24.0	59.0	37.0	57.0		
4	11.060.798.259.654.100	6.717.283.800.077.490	2.OG		70.0	35.0	70.0	28.0	28.0
5	1.106.073.981.051.590	6.717.310.439.364.710	2.OG		77.0	37.0	74.0	39.0	30.0
6	11.060.536.321.085.200	671.731.347.147.925	2.OG		68.0	35.0	70.0		26.0
7	1.106.025.056.988.610	6.717.323.650.719.940	2.OG		70.0	30.0	68.0	28.0	28.0
8									

Abbildung 7: Aufbau der exportierbaren csv-Datei in Excel

3.3.5 Positionsbestimmung mittels des best-match-Algorithmus

Damit eine Position mit der Fingerprint-Technik gefunden werden kann, wird ein Algorithmus verwendet, der den Unterschied zwischen den in der *radio map* gelagerten Messungen und den aktuellen RSS des mobilen Geräts betrachtet. Der best-match-Algorithmus gibt die Position zurück, die am ehesten einer Messung in der Datenbank entspricht. Dazu wird der euklidische Abstand ohne Wurzelziehen verwendet, da die beste Position die mit dem kleinsten Abstand ist. Seien M_1, M_2 und M_3 die gelagerten Messwerte zu drei APs und M_A, M_B und M_C die eingehenden der gegenwärtigen Messung, dann lautet die benutzte Formel:

$$Distance_{Euklid} = (M_A - M_1)^2 + (M_B - M_2)^2 + (M_C - M_3)^2$$

Abbildung 8: Formel des euklidischen Abstands ohne Wurzelziehen

Die Berechnungen auf den Messungen beziehen sich auf die enthaltenen Signalstärken. Die Umsetzung in der Applikation sieht vor, dass immer die aktuell drei stärksten Messwerte mit den dazugehörigen APs dem Algorithmus als Eingabewerte übergeben werden (vgl. HOSSAIN et. al. 2012). Dieser läuft dann über die gelagerten Positionen und berechnet für jede Spalte, mithilfe den dazugehörigen Messwerten, den euklidischen Abstand. Wenn über alle Positionen iteriert wurde, wird die Position mit dem geringsten Ergebnis zurückgegeben. Dabei können zwei verschiedene Fälle auftreten, die im Folgenden genauer betrachtet werden. Der erfolgreiche Fall, also eine Rückgabe der Position, tritt ein, wenn drei APs der aktuellen Position mit ihren Messungen übergeben werden können und es mindestens einen passenden Fingerprint in der *radio map* gibt, der diese drei APs mit ihren Signalstärken beinhaltet.
Der Algorithmus ist nicht erfolgreich, wenn weniger als drei APs von der aktuellen Position des mobilen Gerätes erreicht werden oder wenn in der *radio map* keine einzige Position enthalten ist, die alle drei eingehenden APs mit Messungen referenziert. So ein Fall würde

genau dann eintreten, wenn die Messung aus Tabelle 5 die aktuelle wäre, da keine Position mit Messwerten von AP_1, AP_2 und AP_3 eingelagert ist.

4 Implementierung von GoeRadioMap

Dieses Kapitel beschreibt die genaue Umsetzung des Konzeptes von „GoeRadioMap“ in Java-Code.

4.1 Die Benutzeroberfläche

„GoeRadiomap“ besteht aus einer *Activity* (vgl. DEVELOPER ANDROID 2015_a), die das Verwalten aller Aufgaben übernimmt und sämtliche implementierte Abläufe steuert. Sie ist das zentrale Kontrollelement der Android-Applikation, trägt den Namen *MainActivity* und wird im Manifest auch als solche deklariert, sodass sie beim Start der Applikation als erstes ausgeführt wird. Das, zur *Activity* gehörende, Layout wird in der Datei *activity_main.xml* definiert. Es besteht aus den folgenden sechs Elementen, die mit dem *RelativeLayout* (vgl. DEVELOPER ANDROID 2015_b) relativ zueinander angeordnet werden: einem *MapView* (vgl. ARCGIS ANDROID 2015_a), das die angezeigte ArcGIS-Kartenstruktur präsentiert; einem *TextView* (vgl. DEVELOPER ANDROID 2015_c), das dem Benutzer Informationen bereitstellt, einem *Spinner* (vgl. DEVELOPER ANDROID 2015_d), der die Etagenauswahl ermöglicht, einem *ListView* (vgl. DEVELOPER ANDROID 2015_e), welches die aktuellen Signalstärken zu allen erreichbaren APs mit Namen und RSS-Level anzeigt (vgl. Abb.3), und einem Button, der das Menü öffnet (vgl. Abb.4).

Das *MapView* bietet eine bereits standardmäßig integrierte Navigation an. Mithilfe von Berührungen auf dem Touch-Screen ist es möglich, in der Karte zu zoomen und den Ausschnitt in alle Himmelsrichtungen zu verschieben. Ein Verkleinern wird durch Auseinanderziehen und ein Vergrößern durch Zusammenziehen zweier Finger bewerkstelligt. Ein einfaches Wischen ermöglicht die Verschiebung des Ausschnittes.

4.2 Verwaltung von Karte und Layern

Dem *MapView* werden in der *MainActivity* in der `onCreate()`-Methode mehrere Layer über ihre entsprechenden URLs (vgl. Kap. 3.3.1) hinzugefügt. Zum einen die *basemap*, die als *ArcGISTiledMapServiceLayer* deklariert wird und den Namen `baseMapLayer` bekommt, und zum anderen die drei Feature-Layer für Gebäude, Räume und Grundskizzen, die als *ArcGISFeatureLayer* deklariert und als `buildingsFLayer`, `roomsFLayer` und `linesFLayer` betitelt werden. Die Auswahl eines bestimmten Gebäudes bzw. Raumes erfolgt über den implementierten *OnSingleTapListener*. Wird in dem *MapView* der Touch-Screen angetippt, so wird die Methode *onSingleTap*() aufgerufen. Um das Antippen zu verwalten und so Gebäude und Räume anzuzeigen, werden Instanzen der Klasse *CustomAsyncTask* verwendet, die von der Android-Klasse *AsyncTask* (vgl. DEVELOPER ANDROID 2015$_f$) abgeleitet ist. Mithilfe dieser Klasse ist es möglich, einen Dienst zu starten, der asynchron zu dem der *MainActivity* läuft. Für den *AsyncTask* kann eine Aufgabe definiert werden, die dann im Hintergrund durchgeführt wird. Für diesen Zweck wird die Funktion *doInBackground*() überschrieben. In der Benutzeroberfläche kann abgefragt werden, in welchem Status sich der *AsyncTask* befindet. Lautet der Status „finished“, so können mittels Überschreiben der Funktion *onPostExecute*() die Ergebnisse der Aufgabe verarbeitet und dem Dienst der *MainActivity* verfügbar gemachen werden. In *GoeRadioMap* sind mehrere *AsyncTasks* implementiert, denen als Aufgabe das Auflösen einer *Query* zukommt und die dazu Parameter für diese Suchanfrage übergeben bekommen. Diese werden anschließend im Hintergrund ausgeführt und das Ergebnis als Objekt der ArcGIS-API Klasse *FeatureResult* (vgl. ARCGIS ANDROID 2015$_b$) zurückgegeben. Eine Konstruktion mit asynchronen Tasks ist notwendig, da das Betriebssystem Android auf eine Beendigung der Aufgaben warten würde, wären diese in nur einem Dienst implementiert. Wenn eine der Aufgaben komplex ausfällt, würde das Betriebssystem die Applikation nach einer bestimmten Zeit schließen, da es noch keine Antwort bekommen hat.

4.2.1 Der buildingTask

In der *MainActivity* werden zwei *CustomAsyncTasks* verwendet: der *BuildingTask* und der *HighlightSingleRoomTask*. Die App soll beim ersten Antippen des *MapViews* alle Räume des an diesen Koordinaten liegenden Gebäudes im Erdgeschoss präsentieren. Beim zweiten Antippen soll ein bestimmter Raum markiert und an genau dieser Stelle ein roter Punkt angezeigt werden. Hierzu sind in der Methode *onSingleTap*() zwei Zustände implementiert.

Im ersten wird der *buildingTask* mit den Anfrageparametern `query.setOutFields(new String[]{"*"})` und `query.setGeometry(pointTapped)`, also keinerlei Einschränkungen in der SQL-Abfrage, aber dafür in der Geometrie, von außen aufgerufen. Der Task startet nun mit den übergebenen Parametern einen *QueryTask* (vgl. ARCGIS ANDROID 2015c) an den `buildingsFLayer` und wenn dieser erfolgreich war, werden die Ergebnisse in einem Objekt der Klasse *FeatureResult* gespeichert. Das Ergebnis sollte an dieser Stelle nur das Feature (hier das Gebäude) an der angetippten Stelle sein, da die Geometrie mit in den Parametern enthalten war und nur ein Gebäude an diesem Punkt existieren kann.

```
switch(state){
    case 0:
        menuButton.setEnabled(false);
        filename = "";
        QueryParameters query = new QueryParameters();
        query.setOutFields(new String[]{"*"});
        query.setSpatialRelationship(SpatialRelationship.INTERSECTS);
        query.setGeometry(pointTapped);
        task = new BuildingTask(MainActivity.this, mapView, x, y);
        task.execute(query);

        state=1;
        break;
```

Quellcode 1: Ausschnitt des ersten Zustands der Methode onSingleTap() in MainActivity

Anschließend wird in der *onPostExecute*()-Methode das Gebäude selektiert, wenn es gefunden wurde und der *SwitchFloorTask*, der ebenfalls eine Ableitung von *CustomAsyncTask* ist, aufgerufen.

4.2.2 Der SwitchFloorTask mit DisplayRooms und DisplayLines

Der *SwitchFloorTask* hat die Aufgabe, den Etagenwechsel über den *Spinner* zu verarbeiten. Immer wenn eine neue Etage selektiert wird, werden die einzelnen Räume und Grundskizzen aktualisiert. Da er mit den gleichen Suchparametern und der gleichen URL wie der *buildingTask* arbeitet, ist das Ergebnis ebenfalls genau das angetippte Gebäude. Sobald der Task erfolgreich terminiert ist, werden mithilfe der Attribute „og“ und „ug“ die Einträge für den *Spinner* befüllt. In der Methode `onItemSelected()` ist implementiert, was geschieht, wenn ein Eintrag ausgewählt wird. Da an dieser Stelle die Räume und die Grundskizzen der passenden Etage gezeichnet werden sollen, werden zunächst weitere Suchparameter für die ebenso asynchronen Tasks *DisplayRooms* und *DisplayLines* definiert. Beide Tasks können mit den richtig gesetzten Sucheigenschaften *where* und *outfields* die Räume bzw. die

Grundskizzen zeichnen. Außerdem wird die Variable „Etage“ in der *MainActivity* auf die ausgewählte gesetzt und wurde aus diesem Grund statisch deklariert. In *DisplayRooms* werden die Räume je nach Art in einer anderen Farbe koloriert. Handelt es sich um Flure oder Treppen, wird in einem hellen Gelb eingefärbt, bei Seminarräumen, Hörsälen, Laboren oder PC-Pools in Grün und bei dem Rest in Orange.

```
    MainActivity.roomsFLayer.clear();
    QueryParameters displayRooms = new QueryParameters();
    displayRooms.setOutFields(new String[]{"raumart",
            "gebid"});
    displayRooms.setWhere("gebid ='" + piz
            + "' AND etage ='" + selectedFloor + "'");

   new DisplayRooms(context, piz).execute(displayRooms);

    MainActivity.linesFLayer.clear();
    QueryParameters displayLines = new QueryParameters();
    displayLines.setOutFields(new String[]{"etage",
            "gebid"});
    displayLines.setWhere("gebid ='" + piz
            + "' AND etage ='" + selectedFloor + "'");

    new DisplayRoomLines(context, piz).execute(displayRooms);

 }
```

Quellcode 2: Ausschnitt der Methode onItemselected() in SwitchFloorTask

4.2.3 Der HighlightSingleRoomTask

Im zweiten Zustand in der angesprochenen *onSingleTap()*-Methode wird der *HighlightSingleRoomTask* ausgeführt. Quellcode 3 zeigt, dass, solange ein AsyncTask noch nicht beendet ist, keine weitere Reaktion auf das Antippen auf dem *MapView* folgt. Damit ist gewährleistet, dass der *HighlightSinglerRoomTask* erst ausgeführt wird, wenn der *buildingTask* beendet ist. Diese Architektur ermöglicht eine sichere Übergabe der Attribute des ausgesuchten Gebäudes von dem einen Task zu dem anderen.

```
if(task != null){
    AsyncTask.Status s = task.getStatus();
    if(s!=AsyncTask.Status.FINISHED) return;
}
```

Quellcode 3: Abfrage des Zustandes eines Tasks in der Methode onSingleTap() in MainActivity

Die Suchanfrage des *HighlightSingleRoomTasks* wird dann an den `roomsFLayer` gerichtet und dort der Raum hellblau eingefärbt, der sich an der Stelle des Antippens befindet. Außerdem wird diese Stelle mit einem roten Kreissymbol markiert. Damit im gleichen Gebäude die Auswahl der Räume und Stellen geändert werden können, wird immer wieder in

Zustand zwei gesprungen, wenn das Gebäude erneut auf dem Bildschirm berührt wird. Dabei werden die genauen Identifikationsnummern der Graphikobjekte für das Einfärben und dem Kreissymbol gespeichert, damit die alten gelöscht werden können, bevor sie mit neuen Koordinaten wieder angelegt werden.

```
case 1:

  menuButton.setEnabled(true);
  loadRadiomap();
      ...
  String gebid = task.getBuildingID();
  QueryParameters roomQuery = new QueryParameters();
  roomQuery.setWhere("gebid ='" + gebid
    + "' AND etage ='" + etage + "'");
  roomQuery.setOutFields(new String[]{"raumnr",
    "raumname", "raumart", "type", "rgid", "gebid"});
  roomQuery.setSpatialRelationship(SpatialRelationship.INTERSECTS);
  roomQuery.setGeometry(pointTapped);
  HighlightSingleRoom task =
    new HighlightSingleRoom(MainActivity.this, pointTapped);
  task.execute(roomQuery);
  lastRedDotMapPoint = pointTapped;
  break;
```

Quellcode 4: Ausschnitt des zweiten Zustands der Methode onSingleTap() in MainActivity

4.3 Drahtlosnetzwerk-Funktionen

In *GoeRadioMap* wird die Android-Klasse *WifiManager* (vgl. DEVELOPER ANDROID 2015$_g$) benutzt, um Scans auszuführen, die Informationen über alle erreichten APs sammeln. Dazu wird ein *BroadcastReceiver* (vgl. DEVELOPER ANDROID 2015$_h$) in der *MainActivity* registriert, der, sobald der Scan abgeschlossen ist, die Resultate verfügbar macht. Die Scans werden in einem Objekt der Klasse *ScanRunnable* gestartet, das den global angelegten *WifiManager* übergeben bekommt und die Schnittstelle *Runnable* (vgl. DEVELOPER ANDROID 2015$_i$) implementiert. Damit ist es möglich, das Absuchen in einem, parallel zum Dienst der *MainActivity* ausgeführten Thread im Sekundentakt durchzuführen. Da die Funktionen des *WifiManagers* eine eingeschaltete Wireless-LAN-Verbindung voraussetzen, wird diese vor dem Starten des Threads überprüft und gegebenenfalls manuell aktiviert. Außerdem benötigt die Applikation die Rechte für den Zugriff auf Drahtlosinformationen und für das Ändern des Zustandes der WLAN-Verbindung, die in der AndroidManifest.xml angegeben werden.

Aus den Resultaten werden die Namen bestimmter APs mit ihren RSS in jeweils ein Element des angelegten *ListViews* geschrieben, um sie dem Benutzer zu präsentieren. Alle APs, die nicht den Namen „eduroam“ oder „GoeMobile“ besitzen und somit nicht zur Universität Göttingen gehören, werden weder angezeigt noch weiter verwendet. Die Signalstärken

werden dabei mit der Funktion *calculateSignalLevel()* in Zahlen von Null bis 100 umgerechnet.

```
registerReceiver(new BroadcastReceiver() {
  @Override
  public void onReceive(Context c, Intent intent) {
    results = wifiManager.getScanResults();
    size = results.size();
    arraylist.clear();
      for(int i=0;i<size;i++) {
        if(results.get(i).SSID.equals("eduroam")     ||
        results.get(i).SSID.equals("GoeMobile")){
          HashMap<String, String> item = new HashMap<String,  String>();
          int rss_level = WifiManager.calculateSignalLevel(results.get(i).level, 100);
          item.put(ITEM_KEY, "Acsess Point: " + results.get(i).SSID + " | Signalstaerke:"
          + rss_level);
          arraylist.add(item);
        }
      }
        adapter.notifyDataSetChanged();

  }
}, new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
```

Quellcode 5: Ausschnitt des Füllen des ListViews in der Methode onCreate() in MainAcitivity

4.4 Aufbau der Datenstruktur und Funktionen der Klasse „Radiomap“

4.4.1 Die Klassen „AccessPoint“, „Measurement“ und „Position“

Die Klasse „AccessPoint“ fasst die zwei Zeichenketten des Namens und der MAC-Adresse eines APs zusammen. Außerdem ist die Funktion *equals()* implementiert, die über den Vergleich von MAC-Adressen entscheidet, ob zwei Objekte dieselben sind.

Die Attribute der Klasse „Measurement“ sind die *double*-Variablen „RSS“ und „RSSLevel“ und die Integer-Variable „counter“. In „RSS“ werden empfangene Signalstärken in Dezibel gespeichert und in „RSSLevel“ die mit der bereits angesprochenen Funktion *calculateSignalLevel()* berechneten. Ein Zähler ist notwendig, da bei der Aufnahme Mittelwerte der Signalstärken gebildet werden können und gesichert werden muss, wie viele Berechnungen bereits durchgeführt wurden. Da, aus demselben Grund, Gleitkommazahlen entstehen können, wurde der Dateityp *double* verwendet. Die Funktion *compareTo()* gestattet den Größenvergleich zweier Messungen über ihr berechnetes RSS-Level.

Die Klasse „Position“ umfasst die X- und Y-Koordinaten in der ArcGIS-Kartenstruktur und die Etage. Die Koordinaten haben den Dateityp *double* und die Etage wird als Zeichenkette gelagert. Auch hier ist die Vergleichsfunktion *equals()* implementiert, um entscheiden zu können, ob zwei Objekte dieselbe Position darstellen.

4.4.2 Aufbau und Funktionen der Klasse „Radiomap“

In der Klasse „Radiomap“ werden die, in Kapitel 4.4.1 dargelegten, Klassen zu einem Datentyp zusammengefasst, um somit die Aufnahmen intern zu speichern. Da der Aufbau der logischen Tabelle (vgl. Kap. 3.3.3) als Grundkonzept beibehalten werden soll, wird als Datentyp eine *HashMap* (vgl. DEVELOPER ANDROID 2015_j) verwendet. Dieser wird als Schlüssel die Position und als Wert eine zweite *HashMap* zugewiesen, die wiederum als Schlüssel Objekte der Klasse „AccessPoint“ und als Wert Objekte der Klasse „Measurement“ referenziert.

```
HashMap<Position, HashMap<AccessPoint,Measurement>> map;
```

Quellcode 6: Anlegen der HashMap in Radiomap

Die, im überschriebenen *default*-Konstruktor angelegte, *HashMap* soll mit *Fingerprints* gefüllt werden. Dazu wurde die Methode *setMeasurement()* geschrieben, die zu einer gegebenen Position und einem *Access-Point* die dazugehörige Messung einfügt. Existiert noch kein Eintrag zu der übergebenen Position, dann wird dieser mit einer *AccessPoint-Measurement-HashMap* angelegt, die dann mit den übergebenen Werten gefüllt wird. Eine Mittelwert-Berechnung der Messungen wird durchgeführt, wenn sowohl die Position als auch die Messung mit dem AP schon vorhanden ist. In diesem Fall wird die, unter dieser Position abgespeicherte, Messung aktualisiert. Die Variablen RSS, RSS-Level und counter werden benutzt, um ein neues Measurement-Objekt zu erzeugen, das die gemittelten Werte und den aktualisierten Zähler enthält. Hierzu wird die Formel aus Abb. 8 verwendet (vgl. Kap. 3.3.3). Sobald die Berechnung der neuen Werte abgeschlossen ist, wird das alte Objekt durch das neue ersetzt.

```
public void setMeasurement (Position p, AccessPoint ap, Measurement m) throws Exception{

  HashMap<AccessPoint, Measurement> apm;
  if (!map.keySet().contains(p))
  {
    map.put(p, new HashMap<AccessPoint, Measurement>());
  }
  if (map.keySet().contains(p))
  {
    if(map.get(p).keySet().contains(ap))
    {
      apm = map.get(p);
      Measurement old_m= apm.get(ap);
      int old_cnt = old_m.getCounter();

      double old_RSS = old_m.getRSS();
      double old_RSSLevel = old_m.getRSSLevel();
      double new_RSS = (old_RSS * old_cnt + m.getRSS())/((double) old_cnt +1);
      double new_RSSLevel = (old_RSSLevel * old_cnt + m.getRSSLevel())/
       ((double) old_cnt +1);

      Measurement new_m =new Measurement(new_RSS, new_RSSLevel,old_cnt +1);
      apm.put(ap, new_m );
    }
    else
    {
      apm = map.get(p);
      apm.put(ap, m);
    }
 }
```

Quellcode 7: Ausschnitt der Funktion setMeasurement() in Radiomap

Die Funktion *getMeasurement()* hat die Aufgabe, zu einer Position und einem AP die Messung zurückzuliefern. Auf Grund der Konstruktion über zwei *HashMaps*, bei denen die Position und der AP Schlüssel sind, reicht es aus, die Schlüsselmengen auf Inhalt zu prüfen und falls die Messung vorhanden ist, diese zurückzugeben.

```
public Measurement GetMeasurement(Position p, AccessPoint ap) throws Exception
{
  if(map.keySet().contains(p))
  {
    if(map.get(p).keySet().contains(ap))
    {
      return (map.get(p)).get(ap);
    }
  }
    throw new Exception("Kein Measurement an dieser Stelle vorhanden");
}
```

Quellcode 8: Die Funktion getMeasurement() in Radiomap

Eine weitere Funktion der Klasse „Radiomap“ stellt die Umsetzung des best-match-‚Algorithmus dar (vgl. Kap. 3.3.5). Sie trägt den Namen *bestMatch()* und ihr werden die aktuellen Messungen des mobilen Geräts mit ihren APs in Form von zwei Vektoren übergeben, die mindestens eine Länge von drei Elementen aufweisen müssen. Danach werden über alle, in dem Radiomap-Objekt gelagerten, Positionen iteriert und in jedem Schritt die gelagerten Fingerprints mit den aktuellen verglichen. Es wird versucht, drei Measurement-

Objekte zu finden, die an einer Position die gleichen APs aufweisen, die den einkommenden aktuellen entsprechen. Wenn alle drei gefunden werden, wird der euklidische Abstand ausgerechnet und der Variable „distance“ zugewiesen. Anschließend wird über den Vergleich mit der Variablen „bestDistance“ festgestellt, ob die Messwerte der Position den geringsten Abstand zu den aktuellen haben und falls dies der Fall ist, wird „bestDistance“ neu gesetzt. Wenn unter diesen Umständen in einem Iterationsschritt keine drei passenden APs in der Datenbank gefunden werden, wird die implementierte Ausnahme der Funktion *getMeasurement()* gefangen aber nicht weiter geworfen, da es vorkommen kann, dass von einer Position nicht alle APs erreicht werden können. Wird, die mit dem maximalen double-Wert deklarierte Variable, „bestDistance“ niemals neu gesetzt, bedeutet das, dass kein *Fingerprint* in der Datenbank vorhanden ist, der zu allen drei APs Messungen gelagert hat. Ist dies nicht der Fall, wird die Position zurückgegeben, deren Messung den niedrigsten euklidischen Abstand besitzt und somit am nächsten zur gegenwärtigen Position des mobilen Geräts liegt.

```
Set<Position> Positions = map.keySet();
...
for (Position p: Positions){
  try{
    Measurement m0 = GetMeasurement(p, ap.get(0));
    Measurement m1 = GetMeasurement(p, ap.get(1));
    Measurement m2 = GetMeasurement(p, ap.get(2));
    //Euklidischer Abstand ohne Wurzel
    double distance=Math.pow((double)(m.get(0).getRSS()-m0.getRSS()),2.0) +
                Math.pow((double)(m.get(1).getRSS() - m1.getRSS()), 2.0) +
                Math.pow((double)(m.get(2).getRSS() - m2.getRSS()), 2.0);
    if(distance<bestDistance)    {
      bestPosition = p;
      bestDistance = distance;
    }
  }
  catch(Exception ex){
 //Ignoriere hier extra
  }
}
if (bestDistance == Double.MAX_VALUE)
  throw new Exception("Niemals alle 3 AP/Measurements in der Radiomap
    vorhanden");
else
  return bestPosition;
```

Quellcode 9: Ausschnitt der Funktion bestMatch() in Radiomap

Die letzte wesentliche Funktion ist die *export()*-Funktion. Sie verwirklicht es, die aufgebaute Datenstruktur in eine Datei zu speichern (vgl. Kap. 3.3.4). Mit der Java-Klasse *Filewriter* (vgl. DEVELOPER ANDROID 2015k) wird der, in Abbildung 7 zu sehende, Aufbau umgesetzt. Dieser schreibt Zeichenketten in eine komma-seperated-list-Datei, wobei jedes Feld der

entstehenden Tabelle durch ein Semikolon getrennt wird. In der ersten Zeile werden die Überschrift „Position“ und die Namen der gelagerten APs geschrieben.

```
writer.write(";Position;;");
for(int i=0; i<aps.size(); i++){
    writer.write(aps.get(i).getName() + ";");
}
writer.write(System.getProperty("line.separator"));
```

Quellcode 10: Schreiben der ersten Zeile der Funktion export() in Radiomap

Darauf folgen in der zweiten Zeile, die mit dem Ausdruck `System.getProperty("line.separator")` eingeleitet wird, die Überschriften der X- und Y-Koordinaten und der Etage sowie die MAC-Adressen der gelagerten APs.

```
writer.write("x;y;etage;");
for(int i=0; i<aps.size();i++){
    writer.write(aps.get(i).getMAC() + ";");
}
```

Quellcode 11: Schreiben der zweiten Zeile der Funktion export() in Radiomap

Anschließend werden ab der dritten Zeile die Felder mit den entsprechenden Attributen der Positionen und der Signalstärken gefüllt. Hier wurde sich für den berechneten Level der RSS als Eintrag entschieden. Wenn keine Messung zu einem AP existiert, wird das Feld freigelassen.

```
for(int j=0;j<positions.size();j++){
    writer.write(positions.get(j).getX() + ";" );
    writer.write(positions.get(j).getY() + ";" );
    writer.write(positions.get(j).getEtage() + ";");
    for(int k=0;k<aps.size();k++){
        try{
            Measurement m = GetMeasurement(positions.get(j),aps.get(k));
            writer.write(m.getRSSLevel() + ";");
        }
        catch(Exception ex){
            writer.write(";");
        }
    }
    writer.write(System.getProperty("line.separator"));
```

Quellcode 12: Schreiben ab der dritten Zeile der Funktion export() in Radiomap

Der Pfad, der angibt, wo die Datei auf dem mobilen Gerät gelagert werden soll, und der Name der Datei werden dem *Filewriter* in Form einer Zeichenkette übergeben. Die Informationen über die aktuelle Zeit und das aktuelle Datum, die Teil des Namens sind, werden durch ein Anlegen eines Objektes der Java-Klasse *Date* (vgl. DEVELOPER ANDROID 2015l) erhalten.

4.5 Das Menü

In diesem Kapitel wird auf die Umsetzung der einzelnen Unterpunkte des Menüs eingegangen. Dazu wird zunächst die dazu gebrauchte Implementierung des internen Speicherns und des Ladens der Klasse Radiomap aufgezeigt.

4.5.1 Laden und Speichern

Für das Speichern und Laden werden die Java-Klassen *FileInputStream* (vgl. DEVELOPER ANDROID 2015m) bzw. *FileOutputStream* (vgl. DEVELOPER ANDROID 2015n) verwendet. Über die Schnittstelle *Serializeable* (vgl. DEVELOPER ANDROID 2015o), die alle Elemente der Datenstruktur Radiomap und die Klasse Radiomap selbst implementieren, ist es möglich, die *radio map* intern zu speichern und zu laden. Der dafür eindeutige Dateiname, der aus der Gebäude-Identifikationsnummer und der Etage besteht, wird im Dienst „SwitchFloorTask“ gesetzt. So ist auch hier, wie bei der exportierbaren Datei, eine genaue Zuordnung möglich.
Alle Funktionalitäten, die das Menü bietet, können erst im zweiten Zustand des *OnSingleTapListeners()* verwendet werden und sind, bis dieser erreicht wird, deaktiviert. Dies hat den Grund, dass alle Unterpunkte einen Zugriff auf ein vorher geladenes Radiomap-Objekt benötigen, was auf Grund des erforderten Dateinamens erst nach Beendigung des *SwitchFloorTasks* sichergestellt ist.

4.5.2 Fingerprint speichern, Löschen einer Datei und Wechseln des Gebäudes

Das Speichern eines Fingerprints geht mit einem vorherigen Laden der entsprechenden Datei und einem abschließenden Schreiben in diese einher. Dadurch ist die *radio map* zu einer Etage eines Gebäudes stetig erweiterbar. Die Position und die Messungen werden über die beschriebene Funktion *setMeasurement()* in das aktuelle Radiomap-Objekt geschrieben. Dafür werden die Koordinaten des zuletzt gesetzten roten Punkts und die aktuelle Etage für die Position und die Messwerte des parallel laufenden Threads genutzt (vgl. Kap. 4.3). Die Universitäts-APs werden durch eine vorherige Abfrage gefiltert.
Dank des Aufbaus über zwei Zustände reicht es für das Löschen einer Datei aus, das aktuell geladene Radiomap-Objekt zu leeren und abschließend abzuspeichern. Nach dem Löschen werden alle Layer geleert und der Zustand ändert sich in den ersten, sodass ein neues Gebäude angewählt werden kann. Dieselbe Umsetzung findet sich im dritten Menüpunkt zum Wechseln des Gebäudes wieder, nur dass hier das Löschen entfällt.

4.5.3 Anwendung des best-match-Algorithmus

Der Menüpunkt „best-Match-Algorithmus anwenden“ gebraucht die Funktion *bestMatch()* aus der „Radiomap“-Klasse, die die aktuellen APs mit ihren Messwerten als Eingabe benötigt und die ersten drei übergebenen verwendet (vgl. Kap. 4.4.2). Die Zuordnung von Messung und AP wird garantiert, indem der *Fingerprint* in einem ausschließlich für diesen Zweck entworfenen Datentyp „WifiMeasurement“ eingebettet wird. Da sich entschieden wurde, die drei APs mit den stärksten Signalstärken für die Berechnung zu benutzen, werden die *Fingerprints* vor der Ausführung der Funktion absteigend nach ihren berechneten RSS-Leveln sortiert. Um dies zu erreichen, wird die Java-Klasse *TreeSet* (vgl. DEVELOPER ANDROID 2015_p) ausgenutzt, die Elemente beim Hinzufügen ordnet. Die Entscheidung, wo das Element eingefügt werden soll, geschieht über die in „WifiMeasurement“ und „Measurement“ implementierte *Comparable*-Schnittstelle (vgl. DEVELOPER ANDROID 2015_q). Ein Objekt der Klasse „WifiMeasurement“ ist größer bzw. kleiner, wenn das RSS-Level der dazugehörigen Messung größer bzw. kleiner ist. Insgesamt werden in das *TreeSet* nur die gefilterten *Fingerprints* eingegliedert.

```
Vector<AccessPoint> ap = new Vector<AccessPoint>();
Vector<Measurement> m = new Vector<Measurement>();

if(filterResults.size()>=3){
  TreeSet<WifiMeasurement> actualMeasurements = new
    TreeSet<WifiMeasurement>();
  for (int i = 0; i < filterResults.size(); i++){
    int rss_level = WifiManager.calculateSignalLevel
      (results.get(i).level, 100);
    actualMeasurements.add(new WifiMeasurement
     (new AccessPoint(filterResults.get(i).BSSID,
      filterResults.get(i).SSID),new Measurement
     (filterResults.get(i).level, rss_level)));
  }

  for(WifiMeasurement wm : actualMeasurements){
        ap.add(wm.getAp());
        m.add(wm.getM());
  }
  try{

        Position p = radiomap.BestMatch(ap, m);
                        ...
  }
```

Quellcode 13: Ausschnitt zur Bestimmung der Position in der Methode onClick() in MainActivity

5 Analyse der Daten der SUB-Göttingen

Die, in dieser Bachelorarbeit entwickelte, Applikation „GoeRadioMap“ wurde in der Niedersächsischen Staats- und Universitätsbibliothek Göttingen getestet. In zwei Testphasen wurden dazu *Fingerprints* in verschiedenen Etagen aufgenommen, die dann im Nachfolgenden für eine Lokalisationsberechnung mit dem implementierten Algorithmus verwendet wurden.

5.1 Die erste Testphase

In der ersten Testphase soll analysiert werden, ob die Genauigkeiten von Aufnahmen der Signalstärken mit und ohne Mittelwertberechnungen voneinander abweichen und wenn dies zutrifft sollen Schlussfolgerungen gezogen werden.

5.1.1 Ablauf der Datenerhebung

Es wurden um 13 Uhr im Erdgeschoss 27 Fingerprints der *radio map* hinzugefügt. Die Reihenfolge der, zu den *Fingerprints* gehörenden, Positionen sind in Abbildung 10 zu sehen.

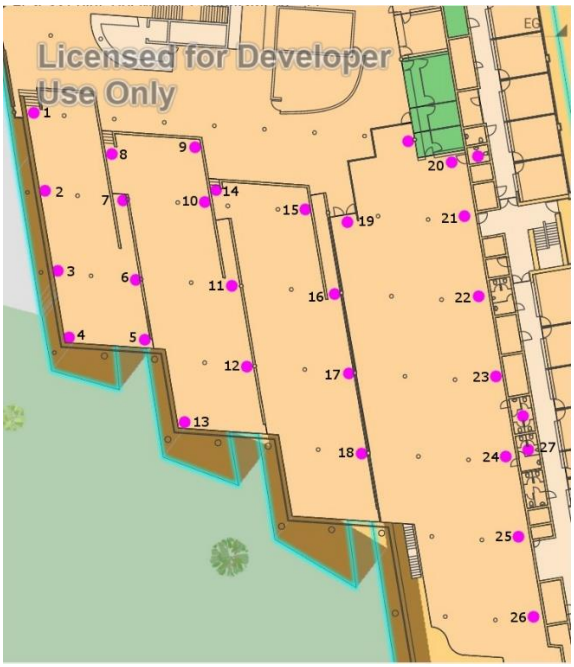

Abbildung 9: Ausgewählte Positionen im Erdgeschoss der SUB-Göttingen

Im direkten Anschluss an die Aufnahme wurde an jeder einzelnen Position eine Lokalisationsberechnung ausgeführt und Abweichungen wurden protokolliert. Die so entstehenden Ergebnisse wurden dann mit einer zweiten *radio map* verglichen, die die

gleichen Aufnahmebedingungen wie die vorher beschriebene hat, mit der Ausnahme, dass an jeder Position drei *Fingerprints* erstellt wurden. Untersucht wird, ob die, in der zweiten *radio map* entstehenden, Mittelwerte der Messungen eine höhere Genauigkeit in der Positionsbestimmung liefern. Da die erste *radio map* des Erdgeschosses der SUB-Göttingen gelöscht werden muss, bevor die Aufnahme der zweiten beginnen kann, muss sichergestellt werden, dass die gleichen Positionen in GoeRadioMap ausgewählt werden. Die Lösung war die Integration eines Feature-Layers, der alle ausgewählten Positionen als Punkt-Geometrie beinhaltet und der eigens für diesen Zweck angelegt wurde. Dies ermöglicht die Markierung jeder Position, die für den Test ausgewählt wurde. Nach Beendigung des Hinzufügens jedes *Fingerprints* wurde die entsprechende csv-Datei exportiert. Das Erstellen beider *radio maps* und die jeweilige Lokalisationsberechnung jeder Position hat ungefähr 70 Minuten in Anspruch genommen.

5.1.2 Auswertung der Daten

Die sich im Anhang befindende Tabelle 5 zeigt die Ergebnisse der Lokalisationsberechnung des best-match-Algorithmus an jeder der 27 Positionen. Zu jeder Position wurde der Abstand zur errechneten Position angegeben, um so eine Abschätzung in der Genauigkeit treffen zu können. In der *radio map* ohne Mittelwertsberechnungen der Signalstärken wurden ca. 37 Prozent der Positionen richtig zugeordnet und die durchschnittliche Abweichung der Position betrug gerundet 11,48 Meter. Die *radio map* mit Mittelwertsberechnungen der Signalstärken hatte eine 44-prozentige richtige Zuordnung und die durchschnittliche Abweichung der Position lag bei gerundet 7,56 Meter.
Eine korrekte Zuordnung der aktuellen Messwerte der APs zu den in der *radio map* gelagerten Fingerprints geschieht genau dann, wenn die Signalstärken der drei stärksten erreichten APs auf allen Positionen stark voneinander abweichen. Falls in einer Etage mehr als drei APs vorhanden sind, ist eine richtige Zuordnung auch dann gegeben, wenn an den Positionen der Fingerprints unterschiedliche APs die höchsten Signalstärken aufweisen. Die Schwankungen der RSS-Level sind somit der größte Einflussfaktor für die Genauigkeit des Fingerprint-Systems. Im Erdgeschoss der Niedersächsischen Staats- und Universitätsbibliothek sind insgesamt 14 APs auf den 27 Positionen erreicht worden. Das Diagramm in Abbildung 11 zeigt die Stärke des berechneten RSS-Levels auf allen ausgewählten Positionen im Erdgeschoss ohne Mittelwerte und Abbildung 12 das Diagramm mit berechneten Mittelwerten der Signalstärken.

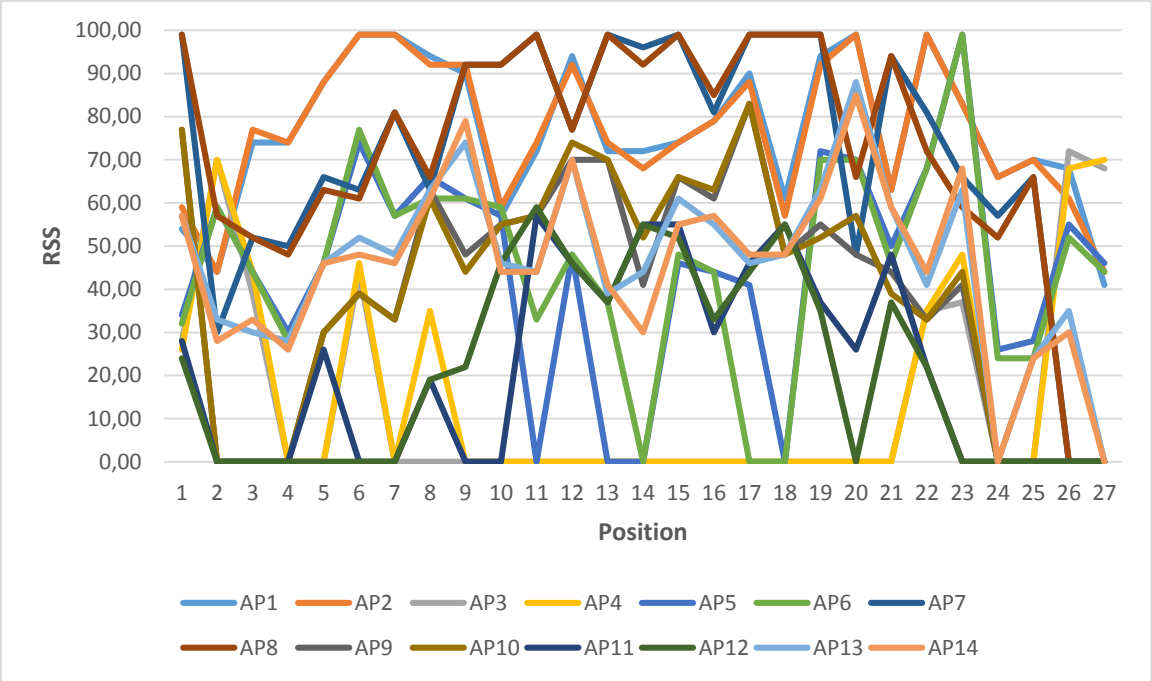

Abbildung 10: RSS-Level der Positionen ohne Mittelwerte der Signalstärken

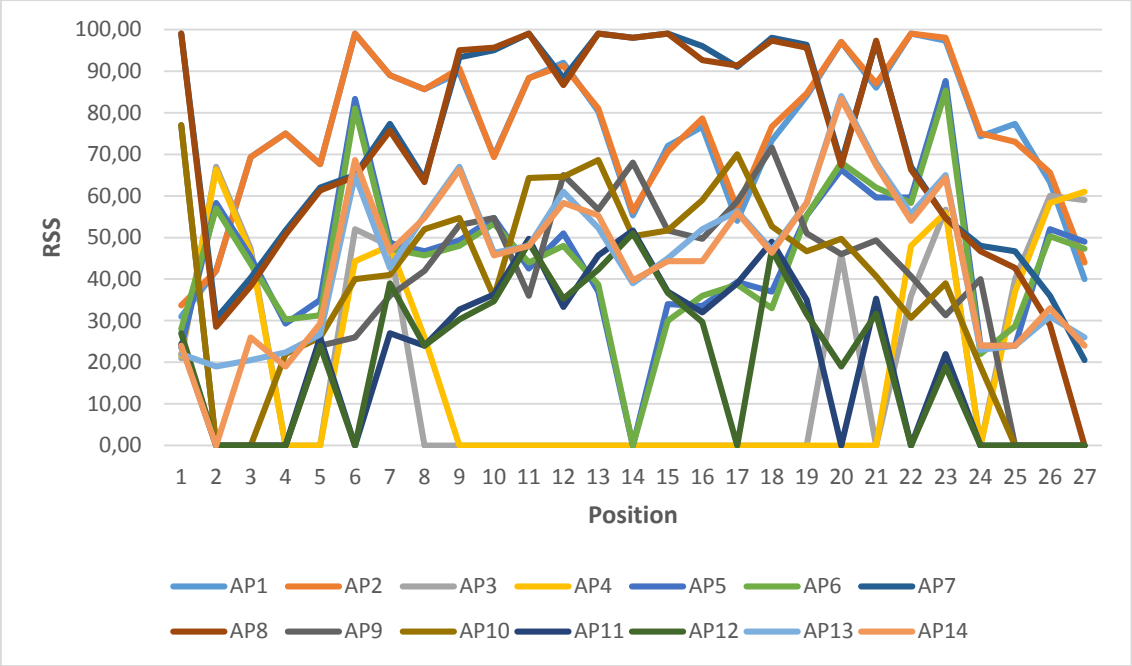

Abbildung 11: RSS-Level der Positionen mit Mittelwerten der Signalstärken

Die Positionen eins, zwei und drei können bei beiden *radio maps* richtig identifiziert werden, da die drei stärksten APs, die von ihnen aus erreicht werden, in Kombination und mit ihren Signalstärken jeweils einzigartig sind. Während die Position eins in der *radio map* mit Mittelwertsberechnungen der Signalstärken mithilfe der APs acht, zehn und zwölf berechnet wird, so wird der Fingerprint der Position zwei ausgewählt, wenn die Signalstärken der APs drei, vier und fünf denen in der *radio map* gelagerten am ehesten entsprechen. Die Position

kann im Umkehrschluss falsch berechnet werden, wenn zwei Positionen mit den gleichen drei stärksten APs bestimmt werden und die im Fingerprint gelagerten Messwerte nicht weit voneinander abweichen. So wird ebenfalls in der *radio map* mit Mittelwertberechnungen der Signalstärken die Position sieben anstatt fünf berechnet, weil sich beide die APs eins, zwei und sieben als die stärksten teilen und ihre Messwerte maximal 20 RSS-Level auseinander-liegen. In dem Diagramm in Abbildung 13 sind die Differenzen der Signalstärken beider *radio maps* zu sehen. Da Differenzen bis zu 36 RSS-Level entstehen können, sind die Mittelwertberechnungen und die damit verbundene Verminderung der Schwankungen der RSS-Level eine Erklärung für die Steigerung der Genauigkeit in der Positionsbestimmung gegenüber der *radio map* ohne Mittelwertberechnung. Als weiterer Punkt für die Erklärung der Verbesserung der Genauigkeit kann die Elimination von Nullwerten aufgeführt werden. Durch die Aufnahme von drei Fingerprints pro Position ist die Chance höher alle erreichbaren APs mit zu berücksichtigen, da es vorkommen kann, dass nicht in jedem Scan jeder AP abgetastet wird.

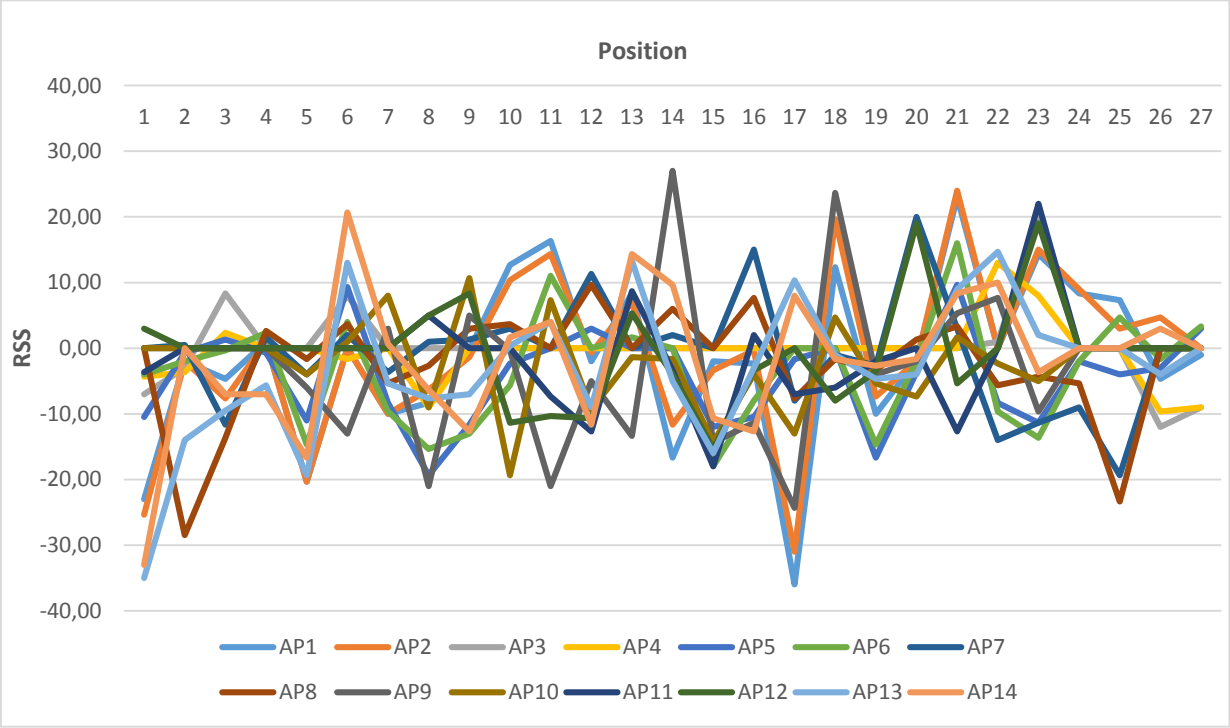

Abbildung 12: Differenzen der RSS-Level der radio maps mit und ohne Mittelwertberechnung

5.2 Die zweite Testphase

In den Diagrammen in Abbildungen 11 und 12 ist zu erkennen, dass immer zwei APs nebeneinander liegen müssen, da fortwährend zwei Linien mit kleinen Abweichungen

übereinander liegen. Nach genauer Analyse stellte sich heraus, dass das Paar jeweils aus einem AP mit dem Namen „eduroam“ und einem mit dem Namen „GoeMobile“ besteht, welche die zwei offiziellen Universitätsnetze sind. Da es für eine Triangulation der Position ungünstig ist, wenn zwei APs dieselben Informationen bieten, wurde die Applikation zu Testzwecken angepasst und eine zweite Testphase durchgeführt.

5.2.1 Ablauf der Datenerhebung

In der zweiten Testphase wurden, analog zur Aufnahme mit Mittelwerten der ersten Testphase, um 13 Uhr im Erdgeschoss jeweils drei Fingerprints der *radio map* an denselben Positionen hinzugefügt und anschließend an jeder Position eine Lokalisierungsberechnung durchgeführt. Der einzige Unterschied war hierbei, dass ein zusätzlicher Filter eingebaut wurde, der nur die „eduroam“-Netze und somit nur einen der beiden nebeneinander liegenden APs für die Positionsberechnung berücksichtigt. Untersucht wird, ob mit dieser Herangehensweise eine Verbesserung der Genauigkeit erzielt werden kann.

5.2.2 Auswertung der Daten

In Tabelle 7 sind Ergebnisse der Lokalisationsberechnung mithilfe des eingebauten Filters zu sehen. Es wurden 17 von 27 Positionen richtig zugeordnet, was einen Prozentsatz von ca. 63 Prozent entspricht. Die durchschnittliche Abweichung betrug 4,07 Meter, welche eine deutliche Verbesserung gegenüber der *radio map* mit Mittelwertsberechnung der Signalstärken der ersten Testphase darstellt. Durch das Streichen der nebeneinander liegenden APs können Positionen, die vorher keinen einzigartigen Fingerprint hatten, richtig zugeordnet werden. Die Position fünf, die im Vorherigen (vgl. Kap. 5.1.2) fälschlicherweise als die Position sieben identifiziert wurde, wird nun korrekt berechnet. Dies hat den Grund, dass sich die Positionen zwar immer noch die APs eins und sieben teilen, aber durch das Wegfallen der APs gleicher Signalstärken der dritte AP unterschiedlich ist (vgl. Abb. 14).

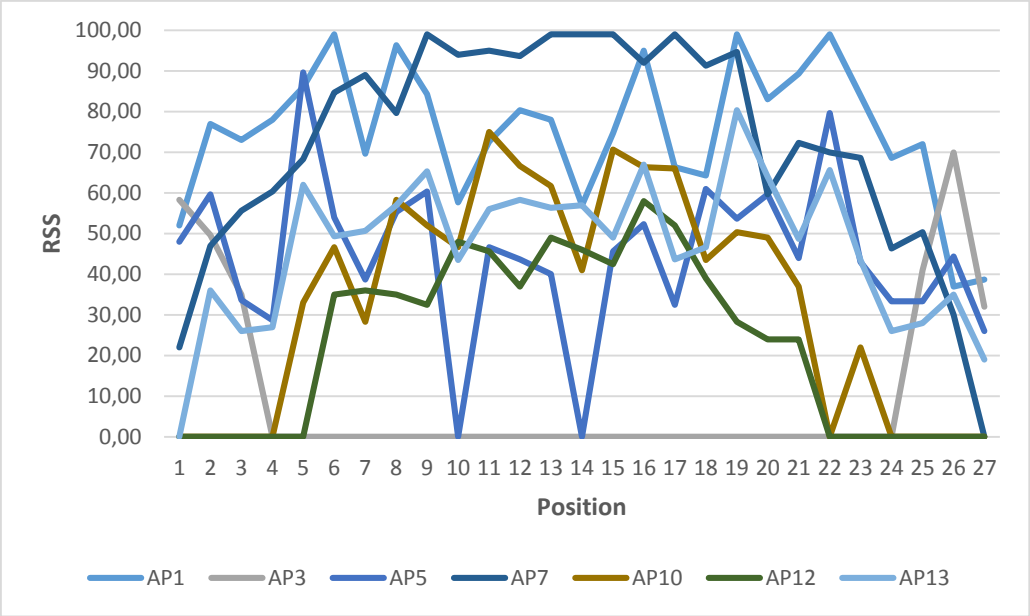

Abbildung 13: RSS-Level der Positionen mit Mittelwertberechnung und Filter

6 Zusammenfassung und Ausblick

Die im Rahmen dieser Bachelorarbeit entwickelte Android-Applikation „GoeRadioMap“ enthält alle Funktionen, die nötig sind, um in jedem Gebäude des Universitätsgeländes der Stadt Göttingen *radio maps* zu erstellen und auf dem mobilen Gerät zu lagern. Sie baut auf bereits vorhandenen Strukturen auf, indem sie das ArcGIS-Kartensystem benutzt. Mithilfe von diesem ist eine benutzerfreundliche und sinnvoll implementierte Navigation integriert. „GoeRadioMap“ kann die, vom Android-Betriebssystem mitgegebenen, Drahtlosfunktionen geschickt nutzen, um die durch Wifi-Scans erhaltenen Daten in geeignete Datenstrukturen zu lagern. Mit den Datenstrukturen ist eine einfache Weiterverarbeitung möglich. Der bereits vorhandene Lokalisierungsalgorithmus kann auf Grundlage dessen Positionsbestimmungen ausführen.

Weitere Arbeiten können diese Bachelorarbeit und die Implementation von „GoeRadioMap“ als Grundlage verwenden, um die als Prototyp gedachte Applikation weiterzuentwickeln. Die Analyse hat gezeigt, dass eine Positionsbestimmung mit der Fingerprint-Technik optimiert werden kann und bereits mit einem einfachen Algorithmus und Filtern ein akzeptables Ergebnis erzielt wird. Es könnte eine server-basierte Umsetzung der Fingerprint-Technik zum Einsatz kommen, bei der die auf dem mobilen Gerät gelagerten *radio maps* an einen Server gesendet werden, auf dem dann die Lokalisationsbestimmung durchgeführt wird. Die Position könnte dann als Ergebnis wieder an die Android-Applikation zurückgesendet werden. Eine Integration des Codes von „GoeRadioMap“ in die bereits entwickelte „GrasApp“ wäre

ratsam, da Studenten ausschließlich auf die *radio maps* zugreifen müssen, die auf dem Server oder auf dem eigenen Gerät gelagert sind. „GrasApp“ ist eine Android-basierte Applikation, die ebenfalls im Rahmen einer Bachelorarbeit entwickelt wurde und das ArcGIS-gestützte Gebäude- und Raumauskunftssystem der Universität Göttingen für Android verfügbar macht. Da „GrasApp“ demnach das gleiche Betriebs- und Kartensystem wie die in dieser Bachelorarbeit entwickelte Applikation benutzt, ist die Kompatibilität gewährleistet (vgl. UNIVERSITÄT GÖTTINGEN 2015).

Literaturverzeichnis

ARCGIS ANDROID, (2015_a): *MapView*. Unter: https://developers.arcgis.com/android/api-reference/reference/com/esri/android/map/MapView.html [Stand 30.11.2015].

ARCGIS ANDROID, (2015_b): *FeatureResult*. Unter: https://developers.arcgis.com/android/api-reference/reference/com/esri/core/map/FeatureResult.html [Stand 30.11.2015].

ARCGIS ANDROID, (2015_c): *QueryTask*. Unter: https://developers.arcgis.com/android/api-reference/reference/com/esri/core/tasks/ags/query/QueryTask.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_a): *Activity*. Unter: http://developer.android.com/reference/android/app/Activity.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_b): *Relative Layout*. Unter: http://developer.android.com/guide/topics/ui/layout/relative.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_c): *TextView*. Unter: http://developer.android.com/reference/android/widget/TextView.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_d): *Spinner*. Unter: http://developer.android.com/guide/topics/ui/controls/spinner.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_e): *ListView*. Unter: http://developer.android.com/guide/topics/ui/layout/listview.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_f): *AsyncTask*. Unter: http://developer.android.com/reference/android/os/AsyncTask.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_g): *WifiManager*. Unter: http://developer.android.com/reference/android/net/wifi/WifiManager.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_h): *BroadcastReceiver*. Unter: http://developer.android.com/ reference/android/content/BroadcastReceiver.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_i): *Runnable*. Unter: http://developer.android.com/reference/java/lang/Runnable.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_j): *Hashmap*. Unter: http://developer.android.com/reference/java/util/HashMap.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_k): *Filewriter*. Unter: http://developer.android.com/reference/java/io/FileWriter.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_l): *Date*. Unter: http://developer.android.com/reference/java/util/Date.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_m): *FileInputStream*. Unter: http://developer.android.com/reference/java/io/FileInputStream.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_n): *FileOutputStream*. Unter: http://developer.android.com/reference/java/io/FileOutputStream.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_o): *Serializable*. Unter: http://developer.android.com/reference/java/io/Serializable.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_p): *TreeSet*. Unter: http://developer.android.com/reference/java/util/TreeSet.html [Stand 30.11.2015].

DEVELOPER ANDROID, (2015_q): *Comparable*. Unter: http://developer.android.com/reference/java/lang/Comparable.html [Stand 30.11.2015].

ELKSTEIN, DR. M., (2015): *A fast-training course for REST - Representational State Transfer, a new approach to systems architecture and a lightweight alternative to web services*. Unter: http://rest.elkstein.org/ [Stand 30.11.2015].

ESRI, (2013): *ArcGIS Resources – Was ist ArcGIS for Server?* Unter: http://resources.arcgis.com/de/help/main/10.1/index.html#/na/01540000037p000000/ [Stand 30.11.2015].

ESRI, (2014): *ArcGIS Resources - Überblick über die Geoverarbeitungs-REST-Services*. Unter: http://resources.arcgis.com/de/help/main/10.2/index.html#//005700000065000000 [Stand 30.11.2015].

ESRI, (2015_a): *ArcGIS Resources – Was ist ArcGIS?*. http://resources.arcgis.com/de/help/getting-started/articles/026n00000014000000.htm [Stand 30.11.2015].

ESRI, (2015_b): *Einführung in ArcGIS for Server*. http://resources.arcgis.com/de/help/getting-started/articles/026n00000007000000.htm [Stand 30.11.2015].

ESRI, (2015_c): *Hilfe zu ArcGIS Online - Gehostete Web-Layer*. Unter: https://doc.arcgis.com/de/arcgis-online/share-maps/hosted-web-layers.htm [Stand 30.11.2015].

ESRI, (2015_d): *ArcGIS for Developers – Query Task.* Unter: https://developers.arcgis.com/ios/objective-c/guide/query-task.htm [Stand 30.11.2015].

FARID, ZAHID / NORDIN, ROSDIADEE / DAUD, WAN MUHAMAD ANAS WAN / ZAKIAH HASAN, SITI, (2014): *Leveraging existing WLAN infrastructure for wireless indoor positioning based on fingerprinting and clustering technique*. Malaysia.

GOOGLE, (2015): *About Indoor Maps*. Unter: https://support.google.com/gmm/answer/1685872?hl=en&ref_topic=3517937 [Stand 30.11.2015].

HANSEN, RENE / THOMSEN, BENT / THOMSEN, LONE LETH / ADAMSEN, FILIP STUBKJÆR, (2014): *SmartCampusAAU - An Open Platform Enabling Indoor Positioning and Navigation*. Dänemark.

HE, SUINING / GARY CHAN, S.-H., (2015): *Wifi Fingerprint-based Indoor Positioning Recent Advances and Comparisons*.

HENNIGES, ROBIN, (2012): *Current approches of Wifi Positioning*. Berlin.

HOSSAIN, SAZZAD / H. S. ARIFFIN, SHARIFAH / FISAL, NORSHEILA / CHOONG KHONG, NENG / HASSAN, N. S. ABU / LATIFF, L.A., (2012): *Accuracy Enhancement of Fingerprint Indoor Positioning System*. Malaysia.

ITWISSEN, (2015): *MAC-Adresse*. Unter: http://www.itwissen.info/definition/lexikon/MAC-Adresse-MAC-address.html [Stand 30.11.2015].

OPENWLANMAP, (2015): *Was ist OpenWLANMap?* Unter: http://openwlanmap.org/.
UNIVERSITÄT GÖTTINGEN, (2015): *GrasApp for Android*. Unter: https://www.uni-goettingen.de/de/476872.html [Stand 30.11.2015].

Anhang

Positionsnr.	*Berechnete Position ohne Mittelwertberechnung*	*Abweichung [m]*	*Berechnete Position mit Mittelwertberechnung*	*Abweichung [m]*
1	1	0	1	0
2	2	0	2	0
3	3	0	3	0
4	13	19	4	0
5	15	27	7	19
6	7	10	1	26
7	4	20	8	7
8	1	12	1	12
9	10	8	9	0
10	10	0	14	2
11	12	11	17	19
12	19	23	5	14
13	18	24	12	12
14	19	17	14	0
15	15	0	5	27
16	16	0	17	11
17	17	0	16	11
18	4	42	18	0
19	19	0	15	6
20	21	8	20	0
21	22	11	19	16
22	22	0	22	0
23	14	45	24	11
24	25	11	24	0
25	26	11	24	11
26	25	11	26	0
27	27	0	27	0

Tabelle 6: Ergebnisse der berechneten Positionen der ersten Testphase

Positionsnr.	**Geänderte Aufnahme**	***Abweichung [m]***
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0
7	7	0
8	9	12
9	9	0
10	10	0
11	5	14
12	10	22
13	13	0
14	15	11
15	15	0
16	17	10
17	17	0
18	18	0
19	19	0
20	21	8
21	22	11
22	22	0
23	24	11
24	24	0
25	24	11
26	26	0
27	27	0

Tabelle 7: Ergebnisse der berechneten Positionen der zweiten Testphase

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich diese hier vorgelegte Abschlussarbeit des Bachelor-Studiengangs Angewandte Informatik zur Erlangung des Akademischen Grades „Bachelor of Science“ (B.Sc.) der Georg-August-Universität Göttingen, mit dem Titel „Entwicklung einer Android-Applikation zur Indoor-Lokalisierung über WiFi-Netzwerke als Prototyp für die SUB Göttingen“ selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen oder anderen Quellen, insbesondere dem Internet, entnommen sind, sind als solche eindeutig und wiederauffindbar kenntlich gemacht. Alle diese Quellen sind vollständig und abschließend in einem Literaturverzeichnis angegeben. Die vorliegende Arbeit ist in gleicher oder ähnlicher Form noch nicht veröffentlicht.

Göttingen, 03.12.2015

Ort, Datum

Unterschrift