



BEUTH HOCHSCHULE FÜR TECHNIK BERLIN  
University of Applied Sciences

# Bachelorarbeit

Medieninformatik

---

---

Ampelphasen-Informationssystem für FahrradfahrerInnen  
auf Grundlage persistenter geo- und zeitbasierter Daten

---

---

Berlin, den 6. März 2015

*Autorin:*

Jacoba BRANDNER

*Matrikelnummer:*

786635

*Betreuerin:*

Frau Prof. Dr. Gudrun GÖRLITZ

*Gutachterin:*

Frau Prof. Dr. Petra SAUER

# **INHALT**

<b>1 Einführung</b>	<b>5</b>
1.1 Motivation . . . . .	5
1.2 Zielstellung . . . . .	5
1.3 Aufbau der Arbeit . . . . .	6
<b>2 Bestehende Konzepte</b>	<b>7</b>
2.1 Ampelinformationssysteme . . . . .	7
2.1.1 Grüne Welle auf Radwegen . . . . .	7
2.1.2 Ampelinformationssysteme im Auto . . . . .	8
2.1.3 Ampelinformationssysteme als mobile Applikation . . . . .	11
2.2 Fahrraderweiterungen . . . . .	13
2.2.1 Displaylose Fahrradnavigation . . . . .	13
2.2.2 Intelligente Fahrradlenker . . . . .	13
2.2.3 Das Samsung Smart Bike . . . . .	14
2.2.4 Der COBI Fahrradcomputer . . . . .	15
<b>3 Lösungsansätze</b>	<b>17</b>
<b>4 Grundlagen</b>	<b>19</b>
4.1 Technische Grundlagen . . . . .	19
4.1.1 Android . . . . .	19
4.1.2 Mobile Sensorik . . . . .	21
4.1.3 Standortbezogene Dienste . . . . .	22
4.2 Berechnung der Geschwindigkeitsempfehlung . . . . .	26
<b>5 Szenarien im Ampelbereich</b>	<b>27</b>
<b>6 Anforderungsdefinition</b>	<b>29</b>
6.1 Funktionalität . . . . .	29
6.1.1 Berechnungen . . . . .	29
6.1.2 Datengrundlagen . . . . .	30
6.2 Die graphische Oberfläche . . . . .	32
6.2.1 Empfehlungen . . . . .	32

<b>7 Konzeption</b>	<b>33</b>
7.1 Anwendungsaufbau . . . . .	33
7.2 Datengrundlage . . . . .	33
7.2.1 Ampeldaten . . . . .	34
7.2.2 Ampelobjekt im JSON-Format . . . . .	35
7.3 Anwendungsfälle . . . . .	36
7.4 Klassenarchitektur . . . . .	37
7.5 Die graphische Oberfläche . . . . .	38
7.6 Testfälle . . . . .	39
7.7 Entwicklungsumgebung . . . . .	40
<b>8 Der Prototyp</b>	<b>41</b>
8.1 Die Manifest- und build.gradle-Datei . . . . .	41
8.2 MainActivity-Klasse . . . . .	43
8.3 Umsetzung Szenarien . . . . .	43
8.3.1 Einlesen der Ampeldaten . . . . .	43
8.3.2 Ermittlung der nächsten Ampel . . . . .	44
8.3.3 Berechnung der Geschwindigkeitsempfehlung . . . . .	46
8.4 Installationsanleitung . . . . .	46
<b>9 Evaluation</b>	<b>47</b>
9.1 Systemtest und Ergebnisse . . . . .	47
9.1.1 Ermittlung der nächsten Ampel . . . . .	48
9.1.2 Berechnung und Anzeige der Geschwindigkeitsempfehlung . . . . .	48
<b>10 Ergebnis und Ausblick</b>	<b>49</b>
10.1 Ampelhinweissystem . . . . .	49
10.1.1 Datengrundlage . . . . .	49
10.2 Ausblick . . . . .	49
<b>Akronyme</b>	<b>50</b>
<b>Glossar</b>	<b>52</b>
<b>Abbildungsverzeichnis</b>	<b>53</b>
<b>Literaturverzeichnis</b>	<b>54</b>
<b>Anhang</b>	<b>59</b>

---

## **Zusammenfassung**

Im Berliner Verkehrswesen ist ein deutlicher Trend zu bemerken. Das Fahrrad wird zum ökologischen und gesundheitlichen, aktiven Lebensstil und wird dem hohen Verkehrsaufkommen der Automobile, insbesondere in der Stadtregion, entgegenwirken. "Fahrradfahren boomt in Berlin stärker als bislang bekannt" (J.Anker, Berliner Morgenpost, am 6.06.2014)

Neue Fahrradwege und Vergrößerung des Fahrradstraßennetzes sind regionale Baumaßnahmen, die dabei aktuell diesen Fahrradtrend unterstützen. Grund der neuen Fahrradeuphorie ist nicht zuletzt die erfolgreiche Etablierung der E-Bikes. E-Bikes erfreuen sich großer Beliebtheit und ermöglichen auch längere Touren ohne große Anstrengung.

## **Abstract**

Im Berliner Verkehrswesen ist ein deutlicher Trend zu bemerken. Das Fahrrad wird zum ökologischen und gesundheitlichen, aktiven Lebensstil und wird dem hohen Verkehrsaufkommen der Automobile, insbesondere in der Stadtregion, entgegenwirken. "Fahrradfahren boomt in Berlin stärker als bislang bekannt" (J.Anker, Berliner Morgenpost, am 6.06.2014)

Neue Fahrradwege und Vergrößerung des Fahrradstraßennetzes sind regionale Baumaßnahmen, die dabei aktuell diesen Fahrradtrend unterstützen.

Grund der neuen Fahrradeuphorie ist nicht zuletzt die erfolgreiche Etablierung der E-Bikes. E-Bikes erfreuen sich großer Beliebtheit und ermöglichen auch längere Touren ohne große Anstrengung.

# 1 EINFÜHRUNG

## 1.1 MOTIVATION

Im Berliner Verkehrswesen ist ein deutlicher Trend zu bemerken. Das Fahrrad wird zum ökologischen und gesundheitlichen, aktiven Lebensstil und wird dem hohen Verkehrsaufkommen der Automobile, insbesondere in der Stadtregion, entgegenwirken. „Fahrradfahren boomt in Berlin stärker als bislang bekannt“ [J.A14]

Neue Fahrradwege und Vergrößerung des Fahrradstraßennetzes sind regionale Baumaßnahmen, die dabei aktuell diesen Fahrradtrend bekräftigen. [J.A14]

Grund der neuen Fahrradeuphorie ist nicht zuletzt die erfolgreiche Etablierung der E-Bikes<sup>1</sup>. E-Bikes erfreuen sich großer Beliebtheit und ermöglichen auch längere Touren ohne große Anstrengung. (Vgl. [Dam14] S.70ff)

Die Digitalisierung der Autoinnenräume mit Navigation und Bordelektronik sowie die Verbindungen zu Smartphones stellen aktuell keine Besonderheit mehr dar. Wird das Fahrrad nun als „vollwertiges“ Mitglied im Straßenverkehr angesehen, kann zusätzliche Elektronik wie Navigation die FahrradfahrerInnen unterstützen. Sicherheit und eine rechtzeitige Ankunft am Ziel sind die Hauptaspekte der VerkehrsteilnehmerInnen. Das Halten an der Ampel kann dabei schnell zu Verzögerungen führen. Doch wer die Restzeit im Voraus kennt, kann sich darauf einstellen und so die verlorenen Zeitabschnitte reduzieren.

## 1.2 ZIELSTELLUNG

Der Fahrtfluss der RadfahrerInnen soll nicht unnötig unterbrochen werden. Rote Ampeln zwingen zum Anhalten – das Anfahren kostet Kraft und ist deshalb unbeliebt. So kommt es, dass viele RadfahrerInnen die Straße bei rot überqueren und die Verkehrssicherheit aller gefährden, wo doch das Radfahren an sich zur Gesundheit beiträgt und gut für die Umwelt ist. Angesichts des Nutzenpotentials eines Ampelinformationssystems lässt sich die Zielstellung klar und deutlich formulieren. Durch reibungsloses Passieren der Ampeln wird der Verkehr sicherer und das Radfahren attraktiver.

Um die Ampeldaten zu erfassen, gibt es verschiedene Möglichkeiten. Eine 100 prozentige Deckung erreicht man nicht einmal durch manuelle Ablesung jeder Ampel, denn viele Lichtsignalanlagen werden verkehrsabhängig gesteuert. Fußgängerampeln beeinflussen erst nach Knopfdruck den Verkehr,

---

<sup>1</sup> Elektrofahrrad. Ein Fahrrad mit elektrischem Hilfsmotor

Funkempfänger oder Infrarotdetektoren in Straßenbahnen oder Bussen wird durch Induktionsschleifen in der Fahrbahn der Verkehr erfasst und angepasst. Weiter sind Busse und Straßenbahnen in Berlin in der Lage, aus gewisser Entfernung über Funk Grün anzufordern, was ebenfalls in den Verkehr eingreift. ( Vgl. [ber09] S.4f)

Absolut entscheidend ist die Richtigkeit der Datengrundlage, also die Position und Signalschaltpläne der Ampeln, denn darauf baut die Funktionalität der Anwendung auf. Für die Auswertung dieser Daten werden die potentiellen Wartezeiten an der nächsten Ampel vorzeitig errechnet und den FahrerInnen mitgeteilt. Resultierend können die NutzerInnen die Geschwindigkeit anpassen und die verbleibende Wegstrecke zur Ampel nutzen, um bei Grün ohne anzuhalten die Kreuzung zu überqueren. Für die Datenerhebung werden zugleich die mobilen Systeme der RadfahrerInnen genutzt. So kann zunächst der Prototyp des Ampelhinweissystem entwickelt werden.

Das Ziel der Arbeit ist ein Konzept und dessen prototypische Anwendung eines Ampelhinweissystem, welches einem auf Basis der zu erstellenden Ampeldatenbank Informationen über die Ampelschaltung zukommen lässt und ihn so interaktiv durch das Verkehrsnetz führt.

## 1.3 AUFBAU DER ARBEIT

Zunächst wird analysiert welche Studien, Projekte oder Anwendungen es zu diesem Thema bereits gibt. *Dann wird erläutert, ob sich eine mobile Anwendung oder eher eine Arduinoinstallation anbietet.* Basierend darauf werden im vierten Kapitel die technischen Grundlagen erklärt. Hier werden also Definitionen und Entwicklungswerkzeuge beschrieben und ein Überblick über mögliche Einsatzgebiete gegeben. Für das Verständnis der Umsetzung ist die Klärung der theoretischen Berechnungsgrundlagen erforderlich.

### Kapitel 5: Szenarien

Im Anforderungsanalysekapitel werden die Anforderungen an Funktionalität und Design für die Anwendung ermittelt und strukturiert.(6)

Das 7+8 Kapitel bildet mit der Konzipierung und Implementierung des Prototyps den Kern dieser Arbeit. *Architektur, Design, (Theorie — funktioniert nicht wie in Konzeption beschrieben, weil... ) es wird auf.. einn gegangen.... mobile Anwendung. Applikation (App). die NutzerInnen auf Ampeln hinweist und die Dauer der Phase. Zusammenfassend wird das Konzept noch einmal kritisch betrachtet und evaluiert.*

Kapitel 9 beschreibt die Umsetzung des exemplarischen Prototyps. Dieser wird dann in Architektur, Funktionalität und Design erläutert und schließlich in mehreren Testreihen evaluiert. Anhand einiger Systemtests wird eine Optimierung der Anwendung herausgearbeitet und umgesetzt.

Den Abschluss dieser Arbeit bildet eine Zusammenfassung der Ereignisse dieser Arbeit und einen Ausblick auf zukünftige Entwicklung hinsichtlich des Themas.

## 2 BESTEHENDE KONZEPTE

Die Verkehrsstrategie des Senats sieht vor, dass die Fahrradnutzung bis zum Jahr 2025 20 Prozent des gesamten Verkehrs ausmachen soll. "Wir brauchen eine intelligente Konstruktion, die alle Verkehrsarten verbindet", sagte Berlins derzeitiger Bürgermeister Michael Müller (SPD). [J.A14]

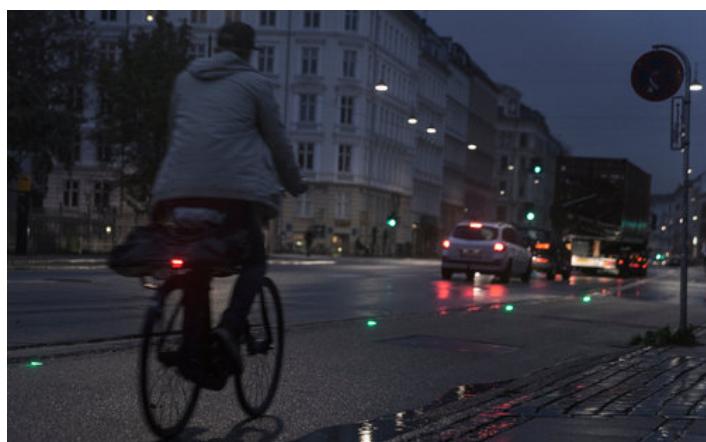
Sowohl in Radwegen integriert, als auch für den Einsatz in Kraftfahrzeugen gibt es bereits Projekte zu Ampelassistenten in Bordcomputern, Navigationssystemen, oder aber auch als mobile Applikation. Solche Anwendungen erkennen rote Ampeln und ermitteln die optimale Fahrgeschwindigkeit für die Grüne Welle. Auch Erweiterungen für Fahrräder werden immer vielfältiger — vom einfachen Navigationssystem bis hin zu intelligenten Aufsätzen, die an das Smartphone gekoppelt sind.

### 2.1 AMPEL INFORMATIONSSYSTEME

Unter dem Prinzip "Grüne Welle" wird die Abstimmung der Ampelschaltzustände, sodass ein Fahrzeug in einer bestimmten Geschwindigkeit mehrere Ampeln passieren kann ohne anzuhalten, verstanden. Im folgenden Abschnitt werden die existierenden Lösungen und Ansätze für Ampelinformati-onssysteme dargestellt.

#### 2.1.1 GRÜNE WELLE AUF RADWEGEN

In Kopenhagen unterstützen grüne Licht-emittierende Dioden (LEDs) auf Radwegen die RadfahreInnen indem sie wenn diese mit einem Tempo von 20 km/h fahren, sie begleiten und so signalisieren, dass sie sich auf der Grünen Welle befinden.



---

Abbildung 2.1 Kopenhagen: LEDs signalisieren die Grüne Welle bei 20 km/h Quelle: [Car14]

Abbildung 2.1 zeigt einen solchen Radweg mit einem Fahrradfahrer, der sich im grünen Bereich befindet. Zusätzlich erkennen Sensoren im Radweg Fahrradgruppen und veranlassen dann die Ampel zu einer längeren Grünphase. In einem anderen Stadtteil sind Leuchttafeln, die die verbleibende Zeit der Ampelphase anzeigen, am Radwegrand installiert [Sch14].

Kopenhagen als Vorbild nehmend hat Berlin mit vier Ampeln in Schöneberg eine Grüne Welle für RadfahrerInnen umgesetzt und plant bereits die zweite [Neu14]. Auch hier möchte man die Benutzung des Rades attraktiver machen und den Fahrradverkehr beschleunigen.

## 2.1.2 AMPELINFORMATIONSSYSTEME IM AUTO

In den letzten Jahrzehnten gab es immer wieder Intentionen, eine Anzeige von Geschwindigkeitsempfehlungen im Fahrzeug umzusetzen. Der folgende Abschnitt stellt existierende Lösungen und Ansätze für Ampelinformationssysteme im Auto vor.

### PROJEKT WOLFSBURGER WELLE

Die VW-Forschung initiierte in den 80er Jahren mit dem Projekt "Wolfsburger Welle" die ersten Untersuchungen zur "Grünen Welle" Informationen im Fahrzeug; mit der Idee, beim Annähern an eine Ampel die optimale Geschwindigkeit im Fahrzeug zu geben. [Zim84] "Dazu sendet die Ampelanlage ihren aktuellen Phasenzustand und eine Prognose für den nächsten Zustandwechsel an alle Fahrzeuge, die sich annähern. Der Fahrzeugcomputer setzt dann die aktuelle Fahrzeuggeschwindigkeit mit dem Abstand zur Ampel und der aktuellen Ampelphase in Bezug. Daraus wird errechnet, ob das Fahrzeug im Moment mit der grünen Welle 'mitschwimmt' oder ob die Geschwindigkeit außerhalb des optimalen Bereichs liegt" ([Tho09]).

### PROJEKT TRAVOLUTION

Im Sommer 2008 wurde das Projekt TRAVOLUTION (TRAffic & eVOLUTION), von den Projektpartnern<sup>1</sup> abgeschlossen. Es besteht aus den Teilprojekten VERKEHRSADAPTIVE NETZSTEUERUNG MIT GENETISCHEN ALGORITHMEN und DER INFORMIERTE FAHRER. Im Netzsteuerungsprojekt wurden 46 Lichtsignalanlagen in Ingolstadt mit der Netzsteuerungssoftware BALANCE ausgestattet, wodurch sie intelligent auf den Verkehr reagieren und die Schaltung an den Verkehr anpassen. Ziel des zweiten Teilprojektes war es, die Autofahrer über die Ampelphasen zu informieren. Die Car-to-Infrastructure (C2I) auf Basis von Wireless Local Area Network (WLAN) umsetzend, senden mit Kommunikationsmodulen ausgestattete Ampeln die Grünphasen an den Bordcomputer der Autos, welcher dann die Geschwindigkeit für ein reibungsloses Passieren errechnet [BBK<sup>+</sup>09] und wie in Abbildung 2.2 zu sehen ist, anzeigt. Im Rahmen des Projektes wurden zwei Anwendungsfälle umgesetzt. Die Restrotanzeige – die die Dauer der verbleibenden Rotphase angibt, und die "Dynamische Grüne Welle" – die Anzeige der

---

<sup>1</sup> [tra]

Progressionsgeschwindigkeit<sup>2</sup>. Die Vorhersage der Schaltbilder ist aufgrund der verkehrsabhängigen Logik bei nicht festzeitgesteuerten Lichtsignalanlage (LSA) nur mit einer gewissen Wahrscheinlichkeit möglich. So werden sekündlich die Grünwahrscheinlichkeiten vom LSA-Kommunikationsmodul aktualisiert und an das Fahrzeugmodul gesendet, welches diese in Relation zu eigener Position, Geschwindigkeit und Richtung setzt und die entsprechende Empfehlung ausgibt.




---

Abbildung 2.2 Der Bordcomputer zeigt die optimale Geschwindigkeit an. Quelle: [tra]

Fundierend auf TRAVOLUTION sind Folgeprojekte wie zum Beispiel das ebenfalls von Audi ins Leben gerufene "Ampelinfo online" entstanden. Über Mobilfunk ist in der Car-to-X (C2X)-Anwendung das Auto mit dem zentralen Verkehrsrechner, welcher die Ampelanlagen steuert, vernetzt und visualisiert die entsprechenden Informationen im Bordcomputer. [Amp14]

## PROJEKT KOLIBRI

In Bayern wurde im April 2011 das Pilot-Projekt KOLIBRI<sup>3</sup> mit den Teststrecken der B13 bei München mit sieben und der St2145 in der Nähe von Regensburg mit acht ampelgeregelten Kreuzungen gestartet. Gemeinsam untersuchten die Projektpartner<sup>4</sup> die Funktionen und Auswirkungen eines Ampelassistenten außerhalb von Ortschaften ( [kol] ). "Per Mobilfunk übertragen die Ampeln ihre Daten an die Zentrale der TRANSVER GmbH. Dort wertet sie ein Computer aus und sendet die Ergebnisse an die Fahrzeuge. Ein Anzeigefeld im Bordcomputer oder eine Applikation auf dem Smartphone zeigt an, ob sich das Fahrzeug in der Grünen Welle bewegt.“ ( [Bat13] )

Die FahrerInnen wurden sowohl fahrzeugintegriert<sup>5</sup> als auch via Smartphone, wie Abbildung 2.3 zeigt, über die Schaltung der nächsten Ampel informiert und erhielten Empfehlungen über die aktuelle Progressionsgeschwindigkeit. Nach zwei Jahren erfolgreicher Arbeit war das Pilotprojekt abgeschlossen.

---

<sup>2</sup> Geschwindigkeit die gehalten werden muss, um die Grüne Welle zu erreichen

<sup>3</sup> Kooperative Lichtsignaloptimierung – Bayrisches Pilotprojekt

<sup>4</sup> <http://www.kolibri-projekt.de/Sites/kolibri3.html>

<sup>5</sup> On-Board-Computer



---

Abbildung 2.3 Anzeige der Geschwindigkeitsempfehlung. Quelle: [kol]

## PROJEKT TESTFELD TELEMATIK

Ende des Jahres 2013 wurde in Wien das Projekt TESTFELD TELEMATIK – Feldversuch zur Stärkung österreichischen Know-Hows im Bereich umweltverträglicher Mobilität erfolgreich abgeschlossen. Per C2X-Kommunikation bringt das Projekt Kooperative Dienste wie Ampelinformationen direkt ins Auto. Über Navigationssysteme, integrierte Systeme, Nachrüst-Plattformen oder mobile Endgeräte erreicht die FahrerInnen die Information der optimalen Geschwindigkeit sowie die Dauer der jeweiligen Ampelphase [Jan14]. Abbildung 2.4 zeigt die mobile Anwendung auf einem Tablet mit der Anzeige von einer Grünen Welle bei 50 km/h. Um an die Informationen zu kommen wurden unter anderem Kameras und Sensoren, beispielsweise als Induktionsschleife in die Fahrbahn eingelassen.



---

Abbildung 2.4 Mobile Anwendung des Projekts Testfeld-Telematik Quelle: [Jan14]

Andere Autohersteller wie BMW, Volvo und Volkswagen kooperieren als Forschungsprojekt “Car 2 Car Communication Consortium“ mit TESTFELD TELEMATIK, ebenfalls mit dem Ziel die Sicherheit an Kreuzungen zu verbessern. Im Auto installierte Sensoren kommunizieren mit Kameras und Scanner in der Ampel. Allerdings funktioniert das System nur mit dem ambitionierten Ziel, wenn alle Auto-

hersteller zusammenarbeiten und sich auf den gleichen Standard einigen. [Elf13]

## TOYOTA

Auch Toyota hat ein System entwickelt, welches eine spezielle Infrastruktur an Kreuzungen, die Installation von Infrarot-Sendern, die mit dem Toyota-Navigationssystem kommunizieren erfordert. An roter Ampel werden die Fahrer über die verbleibende Wartezeit informiert. Die ausgestatteten Navigationssysteme wurden bis jetzt jedoch ausschließlich in Japan getestet. [Toy11]

### 2.1.3 AMPELINFORMATIONSSYSTEME ALS MOBILE APPLIKATION

Smartphones sind bereits mit einem Global Positioning System (GPS)-Empfänger ausgestattet und haben Internetzugang, was unter anderem die Entwicklung von mobilen Ampelassistenten ermöglicht. Die nachfolgenden Applikationen existieren bereits oder befinden sich in der Testphase.

## MOBILE APPLIKATION ENLIGHTEN

Connected Signals ist ein Startup, 2014 gegründet, aus Eugene in Oregon (USA), das Echtzeit-Verkehrssignalinformationen sammelt und so Modelle zur Vorhersage von Ampelsignalen entwickelt. Die von Connected Signals erstellte Smartphone Applikation ENLIGHTEN bietet Treiber mit Vorhersagen, wie lange man an einer roten Ampel stehen wird. [Con14a]

Connected Signals verbindet sich mit dem Verkehrmanagementsystem der jeweiligen Stadt, um Informationen über Signal- und Sensorzustände zu erhalten. Diese Informationen werden mit der Karte als auch den Geschwindigkeitsbegrenzungen kombiniert und in ein herstellerunabhängiges Format umgewandelt. Dann wendet Connected Signals die proprietäre Technologie an, um den Status der LSA abzubilden und eine entsprechende Vorhersage über zukünftiges Verhalten zu treffen. [Con14b] ENLIGHTEN verwendet Informationen über den aktuellen Ampelstatus, Tages- und Stoßzeit und bereits gespeicherte Daten, um eine Vorhersage und dessen Wahrscheinlichkeit zu treffen. Nur bei der Wahrscheinlichkeit die hoch genug ist, folglich einer genauen Vorhersage entspricht, visualisiert ENLIGHTEN die Restrotdauer. Für eine ungenaue Vorhersage können zum Beispiel Variationen in den Zeitplänen (Stoßzeiten) verantwortlich sein.

Connected Signals nutzt GPS, kombiniert mit Auskünften der Fahrzeugsystemen wie den Blinker, Geschwindigkeitsmesser und den Bremspedal-status zur genauen Lokalisierung des Autos. Um präzise Ampelinformationen zu liefern werden Werkzeuge von der automatischen Erkennung von Lücken in digitalen Stadtplänen, bis hin zur Bestimmung der Ampelphasenpläne und Zeitplänen basierend auf dem Verkehr an Kreuzungen eingesetzt. [Con14c]

ENLIGHTEN steht in einer kleinen, aber wachsenden Zahl von Städten in den Vereinigten Staaten zur Verfügung.

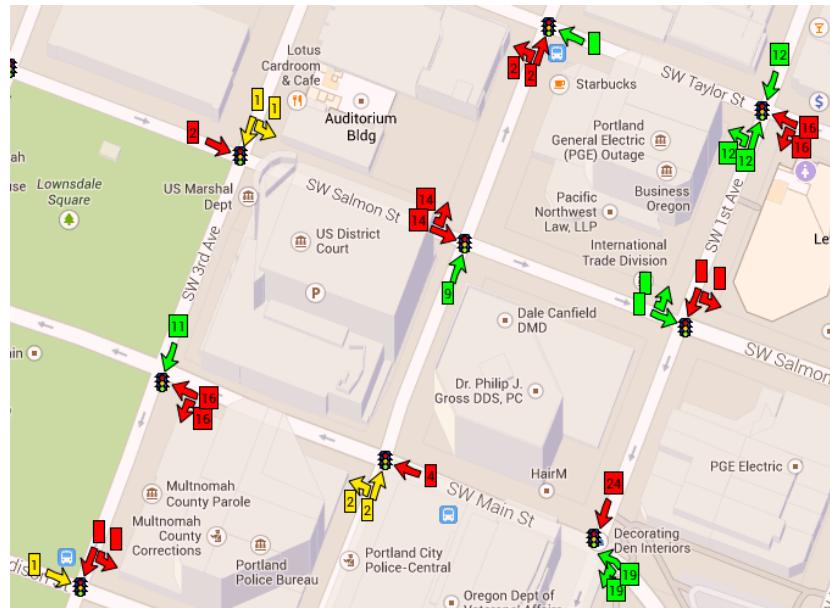


Abbildung 2.5 Live Ampelsignalvorhersage in Portland Quelle: [Con14b]

## MOBILE APPLIKATION SIGNAL GURU

Signal Guru wurde von den Wissenschaftlern des MIT<sup>6</sup> und der Universität von Princeton entwickelt. Unter den vorgestellten Projekten hebt sich Signal Guru insofern ab, als dass die Informationen nicht direkt von einer Vermittlung (Server oder LSA) kommt, sondern von der Anwendung selbst ermittelt wird. Die App errechnet über die Smartphones vieler Nutzer – welche miteinander kommunizieren – die Wahrscheinlichkeit, wann eine Ampel grün wird und errechnen ein daraus ein Zeitmuster zur Vorhersage. Wie in Abbildung 2.6 ist zu sehen ist, muss die eingebaute Kamera durch die Windschutzscheibe die Ampel registrieren. Bei Testläufen im Straßenverkehr vielen die Ergebnisse bei statisch geschalteten Ampeln deutlich besser aus als bei angepassten Ampelschaltungen [KPM11]



Abbildung 2.6 Signal Guru muss in der Lage sein die Ampel zu 'sehen'. Quelle: [KPM11]

Dieser Ansatz ist für Fahrräder jedoch nicht umsetzbar, da das Smartphone in der Halterung am

---

<sup>6</sup> Massachusetts Institute of Technology

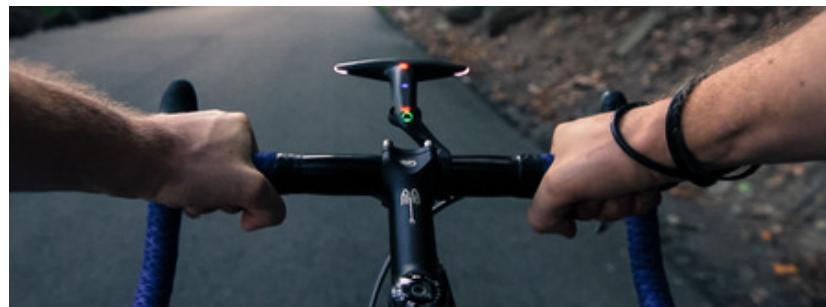
Lenker die LSA nicht erfassen kann.

## 2.2 FAHRRADERWEITERUNGEN

Um das Fahrrad intelligenter und attraktiver zu machen gibt es verschiedene Erweiterungen mit zahlreichen Funktionen. Die hier aufgeführten Fahrraderweiterungen führen zum gewünschten Ziel und ergänzen die Navigation um zusätzliche Eigenschaften, die unter anderem den Weg dorthin *erleichtern*.

### 2.2.1 DISPLAYLOSE FAHRRADNAVIGATION

Das HAMMERHEAD ist ein Gerät in der Form eines “Hammers“ und wird an den Fahrradlenker angebracht. Mit verschiedenfarbigen LEDs bestückt zeigt es den Weg, warnt vor Hindernissen und ersetzt die vorderen Scheinwerfer.



---

Abbildung 2.7 Hammerhead – LEDs zeigen den Weg. Quelle: [ham]

Via Bluetooth ist HAMMERHEAD an das Smartphone gekoppelt, auf dem die zugehörige Navigationsanwendung läuft mit der man Routen eingeben, teilen und speichern kann [ham].

Ein sehr ähnliches Prinzip verfolgt das CYCLENAV von der Firma Schwinn. Unterschiede findet man hier im Design und einem integrierten Lautsprecher, der Abbiegehinweise ausgibt und auf Wunsch wiederholt [cyc14].

### 2.2.2 INTELLIGENTE FAHRRADLENKER

Mehr Technologie, aber auch umfangreichere Funktionen bietet der vom amerikanischen Startup HELIOS-BIKES entwickelte HELIOS-Lenker. Neben dem Frontlicht hat der Lenker wie in Abbildung 2.8 zu sehen, an den Enden LEDs die einen zum gewünschten Standort leiten. Sie passen ihre Farbe der Geschwindigkeit an und haben auf Wunsch auch eine Blinkfunktion. Verbindet man den Lenker mit einem Smartphone, lässt sich die Farbe der LEDs individualisieren. Die Verbindung zum Handy hat weitere Vorteile: Dank des eingebauten GPS-Trackers und eingesteckter SIM-Karte lässt sich das

Fahrrad per SMS über den derzeitigen Standort abfragen [hel14], was im Falle eines Diebstahl sehr hilfreich sein kann.



---

Abbildung 2.8 Helios-Lenker Quelle: [hel14]

VANHAWKS VALOUR heißt das Rad, das ab April 2015 lieferbar ist. Wie im HELIOS-Lenker steckt auch hier ein über das Smartphone steuerbares Navigationssystem, das die Abbiegehinweise per LED signalisiert, im Lenker. Auf den gefahrenen Routen markt sich das Rad durch einen Erschütterungssensor erfasste Hindernisse wie Unebenheiten in der Fahrbahn und ermittelt beim nächsten Mal darauf rücksicht nehmend eine andere Route. Es ist darüber hinaus in der Lage, mit anderen VANHAWKS VA-LOUR-Rädern zu kommunizieren und dessen Routenbegebenheiten ebenfalls zu berücksichtigen. Mittels Radarsensoren registriert das Fahrrad Autos im toten Winkel und benachrichtigt die FahrerInnen durch ein vibrierenden Lenker [van].

### 2.2.3 Das Samsung Smart Bike

Auf der Mailänder Designwoche hat Samsung ein Smartbike vorgestellt, das mit verschiedenen intelligenten Komponenten wie Bluetooth, einer Kamera und Laserprojektoren ausgestattet ist. Der Rahmen ist aus Aluminium und leicht geschwungen, was Vibrationen abfangen soll. Wie Abbildung 2.9 zeigt, zeichnen vier Laserprojektoren den eigenen, begleitenden Fahrradweg auf die Straße und sollen so die Sicherheit erhöhen, indem sie den Sicherheitsabstand markieren und aus dem toten Winkel sichtbar sind. Natürlich ist auch dieses Fahrrad mit dem Smartphone verbunden, das sich dank eines Magneten einfach am Lenker anbringen lässt. Darüber kann man die Laserprojektoren ein- und ausschalten, dafür einen Timer bestimmen und über die eingebaute Kamera unter dem Sattel den Verkehr hinter sich im Auge behalten.

Das Smartphone fungiert außerdem als Navigationsgerät und durch den eingebauten GPS-Empfänger lassen eigene und Routen von anderen Nutzern speichern und intelligent verarbeiten [sma14]. Wenn also viele Menschen mit einem Samsung Smartbike unterwegs sind, erkennt das Rad die Route als angenehm und navigiert dort entlang.



Abbildung 2.9 Samsung Smart Bike Quelle: [sma14]

## 2.2.4 DER COBI FAHRRADCOMPUTER

Ein Projekt aus Frankfurt am Main entwickelt das System COBI (Connected Biking), das alle standardisierten Fahrradsysteme wie Lampen, Navigation, Tachometer etc. vereinen soll. COBI ist ein Modul mit integrierter Frontleuchte in das man das Smartphone, welches dann mit der installierten COBI-App als Fahrradcomputer dient, legt. Durch eine wasser- und stoßfeste Hülle ist es vor Umwelteinflüssen geschützt. Zu dem Lenkersystem gibt es auch Rückstrahler die beim Bremsen intensiver leuchten und eine Blinkfunktion haben.



(a) Frontlicht und Smartphonehalterung

(b) Bremslicht und Blinker

Abbildung 2.10 COBI – Das smarte Fahrradsystem. Quelle: [cob14b]

Möchte man das Smartphone trotzdem nicht am Lenker haben, bleibt die Verbindung zum Modul über Funk bestehen. Steuern lässt sich das System dann über einen Controller, den man am Lenker angebracht, mit dem Daumen bedienen kann. Ist es jedoch in der Halterung, wird das Smartphone über den E-Bike-Akku oder einen zusätzlich integrierten Akku aufgeladen. Wie bei den anderen genannten Systemen ist in der COBI-App eine Navigationsanwendung, wie auch die tracking&share

Funktion inklusive. Darüber hinaus verfügt es über einen Diebstahlschutz, Fitnesstracker sowie die Möglichkeit einer Anbindung an Spotify<sup>7</sup>.

Das Projekt ist bereits voll finanziert und der Versand der vorbestellten Systeme beginnt vorrausichtlich im Frühjahr 2015 [cob14a].

---

<sup>7</sup> Digitaler Musikstreaming Dienst

## 3 LÖSUNGSANSÄTZE

Zur Umsetzung der beschriebenen Ampelinformationsanwendung kommen zwei Möglichkeiten in die engere Wahl. In diesem Kapitel werden die Realisierung durch eine Smartphone-App und die einer Arduino-Anwendung gegenübergestellt. Arduino ist eine Open-Source-Elektronikplatform, basierend auf einfach bedienbarer Hard- und Software die für interaktive Projekte vorgesehen ist. Mit eigener Entwicklungsumgebung und Programmiersprache lässt sich der Arduino steuern. Über viele Sensoren kann ein Arduino die Umgebung erfassen und beeinflusst die Umwelt mit LEDs, Motoren und anderen Akteuren. [arda] Zu beachten sind die Komponenten wie Sensorik, Internetverbindung, Stromversorgung und Darstellung der Informationen.

### GPS

Als Grundlage aller modernen Navigations- und Ortungssysteme im Bereich der Navigation ist GPS für die Fahrradpositionsbestimmung obligatorisch. In einem Smartphone ist ein GPS-Empfänger inklusive, für eine Arduino-Anwendung ein entsprechendes Modul vonnöten (Vgl. [Som10] S. 227).

### Datenspeicherung

Ein weiterer wichtiger Aspekt ist die Datenspeicherung. Mindestens die Position der Ampeln und die Phasen der Schaltpläne können sowohl in einer Datenbankbank als auch einer Datei gespeichert und dort von der Anwendung angefragt und ausgewertet werden. Eine Client-Server Architektur aufbauen, um via Internet auf eine Datenbank zuzugreifen ist in beiden Fällen möglich. Alternativ dazu liefert Android die native Datenbank SQLite und die Möglichkeit, Dateien im internen Speicher abzulegen und auszulesen mit. Auch ein Arduinokann Dank der SD Bibliothek auf eine SD-Karte zugreifen und enthaltene Dateien lesen oder schreiben. Für die Kommunikation zwischen Mikrocontroller und SD-Karte ist die SPI-Schnittstelle vonnöten, welche nur über Pins auf großen Arduino-Boards – und nicht auf den Kleinen, die ohne Weiteres in den Fahrradlenker passen – verfügbar ist. [ardb]

### Stromversorgung

Die Stromversorgung ist im Smartphone durch den integrierten Akku gegeben. Die Laufzeit ist vom Typ abhängig, genügt jedoch für die alltägliche Radstrecke. Für die mobile Stromversorgung des Arduino-Boards wird eine 9-Volt Batterie benötigt, die zusätzlichen Platz beansprucht und wassergeschützt und gut erreichbar angebracht werden muss. Es gibt weiterhin die Möglichkeit den Strom aus dem Nabendynamo zu gewinnen. Außerdem sollte dann der Arduino in der

Lage sein Strom zu speichern, sodass die Anwendung beim Halt an der Ampel nicht ausschaltet.

### Darstellung

Ein Darstellungskonzept muss bei beiden Möglichkeiten erstellt werden. Auf dem Smartphone ist besonders auf Erkennbarkeit bei schlechten Witterungsbedingungen, das ggf. spiegelnde Display berücksichtigend zu achten. Dafür sind aufgrund des vorhandenen Displays in gewisser Größe wesentlich mehr Informationen darstellbar. Als Arduino-Anwendung sind lediglich ein paar helle LEDs am bzw. im Lenker erforderlich. Diese müssen doch eindeutig und intuitiv lesbar sein, um den vollen Informationsumfang zu gewähren. Da die darzustellenden nicht von hoher Komplexität sind, bietet sich hier der Arduino zur Umsetzung der Anwendung an.

## ERGEBNIS

Die Entscheidung fällt auf die Smartphone-Anwendung. Diese überzeugt durch das alleinige Vorhandensein der oben genannten benötigten Komponenten. Auch die vielen Erweiterungen die es bereits für das Fahrrad in Form einer mobilen Anwendung gibt, zeigen dass Lösungen für oben genannte Probleme wie zum Beispiel die Nutzung bei schlechten Witterungsbedingungen existieren.

## 4 GRUNDLAGEN

Dieses Kapitel befasst sich mit sowohl den mathematischen als auch den technischen Grundlagen der zu behandelnden Thematik, welche für das weitere Verständnis der Arbeit beitragen.

### 4.1 TECHNISCHE GRUNDLAGEN

Im Zuge dieser Arbeit wird eine Smartphone-Anwendung erstellt, deren Grundlage für die Implementierung die Software-Plattform Android und der im Smartphone integrierte GPS-Sensor ist. Die eigene Position und Geschwindigkeit wird mittels GPS ermittelt, um in Verbindung mit der festen Position der nächsten LSA die optimale Geschwindigkeit für das Erreichen der “Grünen Welle“ zu errechnen. Im folgenden Abschnitt werden Funktionsweise und Besonderheiten der verwendeten Technologien beschrieben.

#### 4.1.1 ANDROID

Android ist ein von Google entwickeltes Linux-basiertes Betriebssystem für Mobilgeräte. Android-Anwendungen werden mit der Programmiersprache Java und der Auszeichnungssprache Extensible Markup Language (XML) entwickelt. Mit dem Android Software Development Kit (SDK)<sup>1</sup> werden die Werkzeuge und Application Programming Interfaces (APIs) zur Verfügung gestellt, die erforderlich sind, um Mobilanwendungen auf der Android-Plattform erzeugen zu können.

Zu den wichtigsten SDK-Werkzeugen gehören der AndroidSDK Manager, der AVD-Manager, der Emulator und der Dalvik Debug Monitor Server (DDMS). Der SDK Manager verwaltet die SDK-Pakete, sowie die installierten Pakete und System-Images. Der AVD-Manager bietet eine grafische Oberfläche in der Android Virtuell Devices verwaltet, und im Emulator ausgeführt werden können. Mithilfe des DDMS können Android Anwendungen auf Fehler untersucht werden. [andi]

Mit dem Android Native Development Kit (NDK)<sup>2</sup> existiert auch eine Möglichkeit, mit dem Teile einer Anwendung in hardwarenahen Programmiersprachen wie C oder C++ implementiert werden können. Programmcode der in solchen Sprachen geschrieben ist, eignet sich zum Beispiel bei CPU-intensiven Operationen wie Signalverarbeitungen oder Physik-Simulationen besonders gut. Hier ist allerdings sicherzustellen, ob die erforderlichen Bibliotheken in dem SDK auch verfügbar sind. [andh]

---

<sup>1</sup> Das Android SDK steht unter <https://developer.android.com/sdk/index.html> zum Download bereit

<sup>2</sup> Das Android NDK steht unter <https://developer.android.com/tools/sdk/ndk/index.html> zum Download bereit

Einen Überblick über die komplexe Android-Systemarchitektur, welche nachfolgend (nach [Ele15] S. 2ff und [Ele14] S.63ff) kurz beschrieben wird, zeigt die folgende Abbildung.

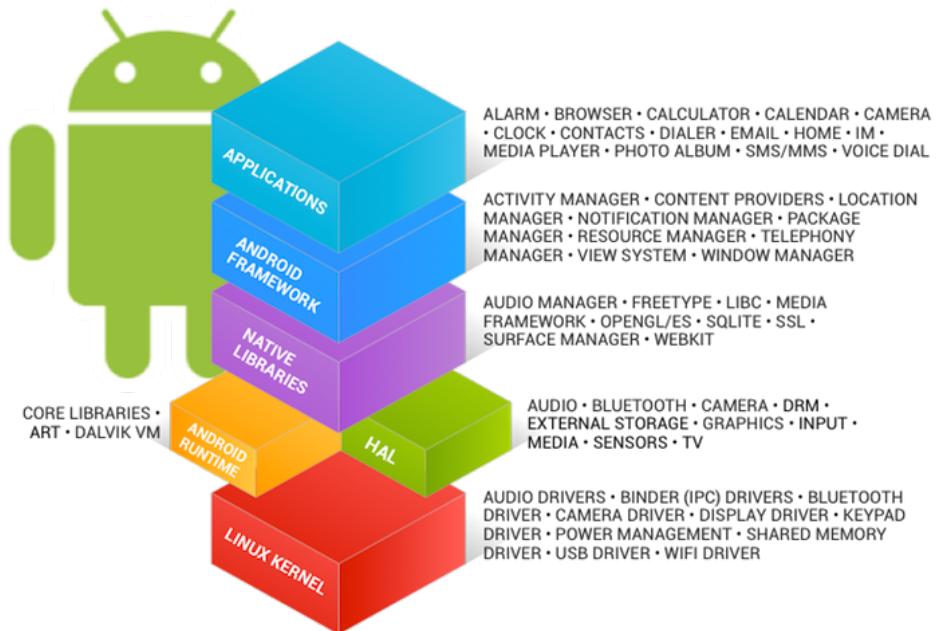


Abbildung 4.1 Die Android-Systemarchitektur Quelle: [figa]

**Linux Kernel:** Android basiert auf dem Linux-Kernel. Wie in jedem Unix-System, stellt der Kernel-Treiber für Hardware, Netzwerk, Dateisystemzugriff und Prozessmanagement bereit.

**Android Runtime:** Die Android Laufzeitumgebung nutzt die aus dem Apache-Harmony-Projekt nachimplementierten Java-Schnittstellen, die Dalvik Virtual Maschine (VM) und die mit der Version 5.0 eingeführten Android Runtime (ART). [andb]

Der Großteil von Android ist in Java implementiert und wird von einer Java Java Virtual Maschine (JVM) ausgeführt. Androids aktuelle Umsetzung der JVM heißt Dalvik. Dalvik wurde speziell auf mobile Geräte zugeschnitten und kann den Java-Bytecode nicht direkt ausführen. So wird der kompilierte Java-Bytecode vom erneut in Dalvik-Bytecode kompiliert und von der Dalvik-VM, oder ab der Version 5.0 von der ART ausgeführt.

Der Hauptunterschied der beiden VMs besteht darin, dass ART die binäre Maschinensprache bei der App-Installation, und nicht wie die Dalvik-VM, bei der Ausführung erstellt.

**Hardware Abstraction Layer (HAL):** Der HAL bietet die Möglichkeit eigene Gerätespezifikationen und deren benötigte Treiber, verbunden über eine Software-Schnittstelle zwischen dem Android-Stack und der Hardware, zu implementieren. [andf]

**Bibliotheken:** Systemeigene Bibliotheken sind C/C++ Bibliotheken und sind vorinstalliert. Dazu gehören alle Bibliotheken im grünen Bereich von Abbildung 4.1:

- **SURFACE MANAGER** Der für die Displayverwaltung verantwortliche Oberflächen-Manager
- **OPENGL/ES** Eine 2D und 3D -Grafikbibliothek
- **SGL** Eine 2D-Grafikbibliothek
- **MEDIA-FRAMEWORK** eine Medien-Bibliothek zur Wiedergabe von Audio- und Video-Daten
- **FREETYPE** eine Bibliothek zur Darstellung von Computerschriften als Rastergrafik
- **SSL** Das Secure-Socket-Layer für die Internet-Sicherheit
- **SQLITE** Ist eine ausgereifte Datenbank die den internen Gerätespeicher nutzt
- **WEBKIT** WebKit ist die Standard-Browser-Engine und erlaubt das Rendern und Anzeigen von HTML Seiten
- **LIBC** Eine C-Bibliothek mit Basisfunktion

**Application Framework:** Androids Application-Framework ist eine Umgebung die unterschiedliche Dienste zur Verfügung stellt. Sie bietet EntwicklerInnen Zugriff auf die im Kern verwendeten APIs sowie auf die Java-Bibliotheken die für Android erstellt wurden.

**Applications:** Auf der obersten Ebene in Abbildung 4.1 befinden sich die Anwendungen die den täglichen Telefon-Bedarf wie Adressbuch, Media-Player, E-Mail, Internet-Browser etc. decken. Zusätzlich unterstützt Android verschiedene Anwendungen von Drittanbietern.

### 4.1.2 MOBILE SENSORIK

Ein Sensor<sup>3</sup> ist ein Bauelement, das physikalische Eigenschaften wie Helligkeit, Temperatur oder Beschleunigung sowohl quantitativ als auch qualitativ erfassen und in ein analoges Signal umwandeln kann. [sen]

Die meisten Android-Mobilgeräte verfügen über integrierte Sensoren, die die Bewegung, Ausrichtung und verschiedene Umgebungsbedingungen messen. Diese Sensoren sind nützlich wenn man dreidimensionale Gerätbewegungen, Positionierungen oder Änderungen in der Umgebung des Gerätes überwachen möchte. So können zum Beispiel Spieleanwendungen den Beschleunigungssensor nutzen, um komplexe BenutzerInnengesten und Bewegungen wie Neigung, Erschütterung, Drehung oder Schwenkung erfassen.

Die Android-Plattform unterstützt Bewegungssensoren zum Messen von Beschleunigungen und Drehungen in drei Achsen, Umgebungssensoren zur Ermittlung verschiedener Umweltparameter wie Luftdruck und -feuchtigkeit, oder Beleuchtung und Temperatur, und Positionssensoren zum Messen der physikalischen Position des Gerätes. Android bietet mit dem Android Sensor Framework eine Sammlung von Klassen und Schnittstellen an mithilfe dessen man auf diese Sensoren zugreifen und deren Daten erfassen kann. [andd]

---

<sup>3</sup> aus dem Lateinischen, deutsch: “*fühlen*“

### 4.1.3 STANDORTBEZOGENE DIENSTE

Standortbezogene Dienste (auch Location Based Service (LBS)) sind Informationsdienste, die die geographische Position des Endgeräts nutzen, um für die NutzerInnen einen Mehrwert bereitzustellen. Sie ermöglichen den NutzerInnen den eigenen Standort zu bestimmen, zu erfahren was sich in der Nähe befindet und das (z.B. in sozialen Netzwerken) zu bewerten. Für die Bereitstellung von LBS sind die vier Schlüsselkomponenten Mobilgerät, Content-Provider, Kommunikationsnetzwerke und Lokalisierungstechniken erforderlich. (Vgl. [FA11] S. 4f)

Zur Bestimmung von Standortdaten gibt es mehrere Techniken. Einige davon werden im Folgenden erklärt.

#### STANDORTBESTIMMUNG VIA NETZWERK

Netzwerkgestützte Standortbestimmung basiert auf Informationen des WLANs oder der Funkzellen des Mobilfunknetzes.

Funkzellentriangulierung verwendet die bekannte Geschwindigkeit von Funksignalen, um den Abstand zum Empfangsgerät zu berechnen. Das Mobilgerät muss mit einem Funkturm in Verbindung stehen. Bewegt sich das Gerät, verbindet es sich mit einem anderen Turm und die Signalstärke des sich nähernden Turms wird stärker. Unter Kenntnis der einzigartigen ID und der Position des Funkturms mit dem das Gerät verbunden ist und ggf. derer, mit denen das Gerät zuletzt verbunden war, lässt sich der eigene Standort ermitteln. (Vgl. [MS12] S. 8f)

Für eine gute Lokalisierung braucht es mindestens drei, vorzugsweise vier Mobilfunkmasten oder Antennen. In Städten, wenn sich mehr Signale der Mobilfunkmasten überlappen kann eine Genauigkeit von bis zu 200 Metern erreicht werden – bei einer geringen Funkturmdichte sinkt diese auf bis zu mehreren Kilometern, wie in Abbildung 4.2c illustriert.

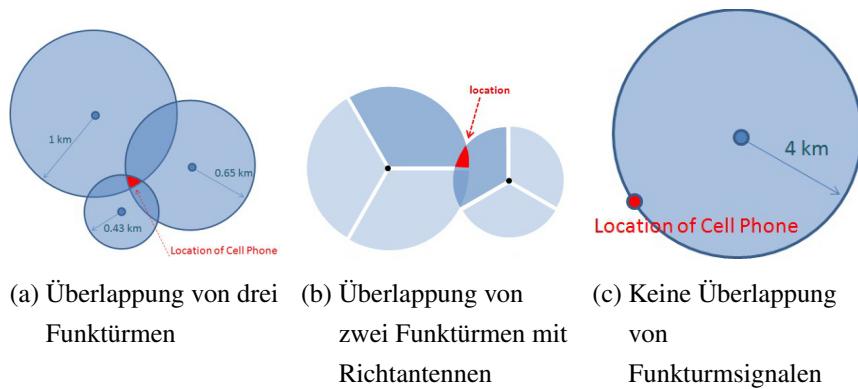


Abbildung 4.2 Funkzellentriangulierung Quelle: [figb]

So eine Überlappung von drei Funktürmen ist in Abbildung 4.2a veranschaulicht. Diese Genauigkeit kann, wie in Abbildung 4.2b demonstriert, erhöht werden wenn Richtantennen auf dem Funkturm installiert sind. So kann zusätzlich die Richtung des Mobiltelefonsignals ermittelt werden. (Vgl. [FA11])

S. 23)

WLAN-basierte Standorterkennung funktioniert durch Funksignale, von WLAN-Access Points<sup>4</sup> ausgesendet, um den genauen Standort eines jeden Wi-Fi-fähigen Gerätes zu ermitteln. Bei Aktivierung scannt die Software die Umgebung nach WLAN-Access Points und berechnet die Position des Gerätes indem sie die empfangenen Signale mit der Referenzdatenbank vergleicht. Wie bei der funkzellen-basierten Standorterkennung erhöht sich auch hier die Genauigkeit (auf 20 bis 40 Meter in Europa) mit wachsender Signaldichte.

Effektiv werden die gleichen Prinzipien der Funkzellentriangulation übernommen. Nur werden WiFi-SSIDs<sup>5</sup> statt Funzellenidentifikationsnummern zum Feststellen der Sendequellen verwendet. (Vgl. [FA11] S. 32ff)

Netzwerkgestützte Standortbestimmung schont zwar im Gegensatz zu GPS den Akku und ist innerhalb von Gebäuden nutzbar, liefert jedoch wesentlich ungenauere Ergebnisse.

### STANDORTBESTIMMUNG VIA GPS

Die satellitengestützte Positionsbestimmung GPS gewährleistet die Bestimmung des exakten Standpunktes und ist so wesentlicher Bestandteil ortsgebundener Anwendungen wie zum Beispiel der in Kapitel 2 beschriebenen.

GPS wurde ursprünglich vom US-Militär entwickelt und dann Mitte der 90er Jahre der Öffentlichkeit zur Verfügung gestellt. Noch heute bleibt es mit einer Genauigkeit von bis zu drei Metern die genaueste Lokalisierungstechnologie. (Vgl. [FA11] S. 24f)

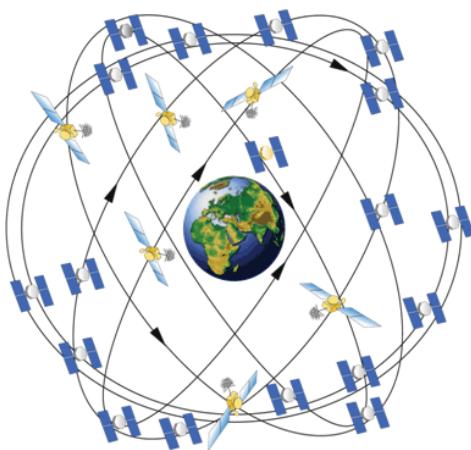


Abbildung 4.3 GPS Satelliten Konstellation Quelle: [figc]

Das Globale Positionsbestimmungssystem umfasst 27 Satelliten die ständig die Erde umkreisen und dabei kontinuierlich ihre eigene, aktuelle Position und Almanach-Daten aussenden. Almanach-Daten enthalten Daten über jeden Satelliten in der Konstellation. Abbildung 4.3 zeigt eine Darstellung der

<sup>4</sup> Drahtloser Zugangspunkt der als Schnittstelle für kabellose Kommunikationsgeräte dient

<sup>5</sup> Basic Service Set Identification (BSSID), entspricht der MAC-Adresse des Wireless Access Points oder wird als Zufallszahl erzeugt

GPS-Satellitenkonstellation. Jeder Satellit folgt einer definierten Bahn, sodass mindestens vier Satelliten von jedem Standpunkt auf der Erde aus „sichtbar“ sind. (Vgl. [MS12] S. 4f)

Mittels Triangulation errechnet daraus ein GPS-Empfänger die eigene Position. Daten von einem einzigen Satelliten lässt die Position des GPS-Empfängers auf deinen großen Bereich der Erde einschränken. Das Hinzufügen eines zweiten engt die Position auf den Bereich, in dem sich die zwei Bereiche überlappen ein. Mit den Daten eines dritten Satelliten bekommt man bereits eine relativ genaue Positionierung. Mit jedem weiteren Satelliten wird die Präzision erhöht und durch zusätzliche Positionsinformation wie Höhe erweitert. GPS-Empfänger benutzen regelmäßig vier bis sieben, oder gar mehr Satelliten. (Vgl. [FA11] S. 25f)

Trotzdem hat GPS, insbesondere für mobile Plattformen einige Nachteile. Es verbraucht viel Energie, was die Akkulaufzeit des Mobiltelefons beeinträchtigt. Bevor der Standort berechnet werden kann müssen mehrere Satelliten ermittelt werden, was nach dem Kaltstart einige Zeit in Anspruch nehmen kann. (Vgl. [MS12] S. 5)

Inzwischen wird auf etlichen Mobiltelefonen zusätzlich das russische Global Navigation Satellite System (GNSS) names Global Navigation Satellite System (Global Navigation Satellite System (GLONASS)<sup>6</sup>) eingesetzt<sup>7</sup>. Die europäische Variante in der Testphase mit ähnlichem Aufbau heißt Galileo, das chinesische, ausschließlich inländisch genutzte, GNSS heißt BeiDou<sup>8</sup>. (Vgl. [FA11] S. 24)

### ASSISTED GLOBAL POSITIONING SYSTEM (A-GPS)

Viele moderne Smartphones sind heutzutage mit der GPS-Variante A-GPS ausgestattet. Hierbei werden Zusatzinformationen über die nächstgelegenen Satelliten via Mobilfunk bezogen, sodass die Bestimmung der Erstposition sehr viel schneller ablaufen kann. Mobilfunktürme sind oft mit einem GPS-Empfänger ausgestattet, welche kontinuierlich die Satellitenpositionen beziehen. Diese Daten werden, sobald angefordert, an das Mobiltelefon gesendet. (Vgl. [FA11] S. 26f)

A-GPS benötigt daher nur die Sicht auf einen Satelliten was auch genauere Ergebnisse in der Ortsbestimmung erzielt. Diese ist durch die Assistenzinformationen auch empfindlicher und somit in Städten, in denen hohe Gebäude die Sicht auf die Satelliten verdecken können, präziser.

### STANDORTBESTIMMUNG UNTER ANDROID

Android unterstützt mit dem `android.location` Paket den Zugriff auf die Ortungsdienste. Als zentrale Komponente des Location Frameworks stellt der `LocationManager` APIs zur Lokalisierung des Geräts bereit. Mit dem `LocationManager` ist die Anwendung in der Lage alle Location Provider<sup>9</sup> des letzten bekannten Standortes abzufragen, sich für regelmäßige Updates zur Position des Gerätes

---

<sup>6</sup> GLONASS, russisch Globalnaja nawigazionnaja sputnikowaja sistema, dt. Globales Satellitennavigationssystem

<sup>7</sup> [http://en.wikipedia.org/wiki/List\\_of\\_smartphones\\_supporting\\_GLONASS\\_navigation](http://en.wikipedia.org/wiki/List_of_smartphones_supporting_GLONASS_navigation)

<sup>8</sup> BeiDou, dt. Großer Bär

<sup>9</sup> Location Provider, dt. Standortanbieter. Ein Standortanbieter bietet regelmäßige Berichte über die geographische Lage des Gerätes

anzumelden und sich wieder abzumelden wenn sich das Gerät außerhalb gegebener Parameter befindet. [andg]

Die geographische Positionsangabe besteht aus Längengrad (Latitude) und Breitengrad (Longitude). Beide werden unter anderem vom `LocationManager` über das `Location`-Objekt als Gleitkommawert geliefert. Daneben auch Informationen wie die Höhe in Metern über der Mehreshöhe, Peilung, Zeitstempel und die Geschwindigkeit.

Der Abstand zwischen zwei Punkten wird mit `Location.distanceBetween()` abgerufen. Diese Methode nimmt die Längen- und Breitenkoordinaten der beiden Punkte und ein Array mit float-Werten für die Ergebnisse. Dieses Array muss eine Größe von mindestens eins haben und gibt die ungefähre Entfernung in Metern im Index Null zurück. Abstandsberechnungen werden mit dem Referenzsystem WGS84<sup>10</sup>-Ellipsoid definiert. ( Vgl. [MS12] S.43)

Der Fused Location Provider verwaltet die zugrunde liegenden Ortungstechnologie. Er ermöglicht zum Beispiel den Zugriff auf letzte Position und minimiert den Energieverbrauch der Anwendung indem er auf Grundlage aller eingehenden Standortanfragen und verfügbaren Sensoren die effizientesten auswählt. [ande]

---

<sup>10</sup> World Geodetic System

## 4.2 BERECHNUNG DER GESCHWINDIGKEITSEMPFEHLUNG

Präsentiert das System während der Anwendung eine Geschwindigkeitsempfehlung, ist diese abhängig von der Fahrgeschwindigkeit und vom Abstand zur Ampel. Für einen Streckenabschnitt zwischen zwei Punkten (Position der Ampel und Position der Rades) wird die Zeitspanne  $\Delta t$  benötigt. Diese errechnet man mit  $t_2 - t_1$ , woraus sich dann die Durchschnittsgeschwindigkeit im untersuchten Streckenabschnitt ergibt.

Angenommen die Progressionsgeschwindigkeit  $v$  wird zum Zeitpunkt  $t_1$  ermittelt, die LSA schaltet zum Zeitpunkt  $t_2$  auf Rot und Abstand zur Ampel beträgt  $s$ , dann gilt:

$$v = \frac{s}{t_2 - t_1}$$

Der Abstand zur Ampel wird also durch die verbleibende Zeit dividiert. Die aus den erfassten Daten erstellten Ampelsignalpläne und Lage der angesteuerten Ampel sind die Basis dieser Berechnung. Die aktuelle Position des Fahrrads wird vom GPS-Sensor des Smartphones ermittelt und daraus der Abstand zur Ampel errechnet. Die Abbildung 4.4 soll die Berechnungsgrundlagen veranschaulichen:

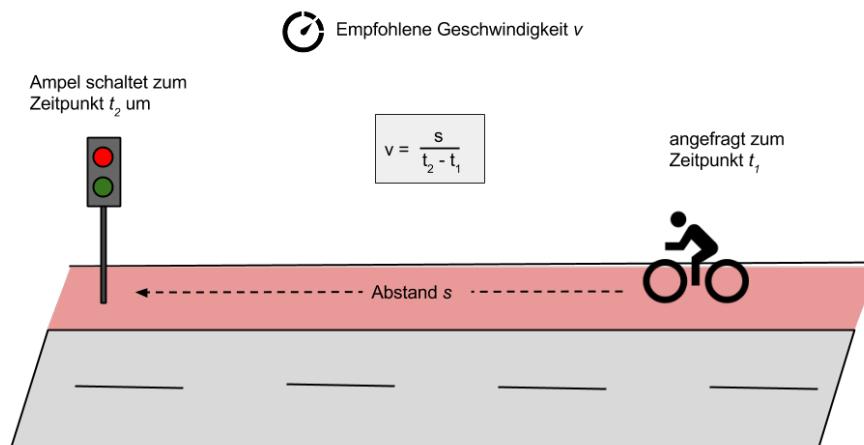


Abbildung 4.4 Veranschaulichung der Berechnung

Da das System für Fahrräder entwickelt wird, ist auch die Beschleunigung von Bedeutung. FahrradfahrerInnen können nicht unbegrenzt beschleunigen, um die gewünschte Geschwindigkeit zu erreichen. Sie ist abhängig von der Geschwindigkeit und der Zeitspanne in der diese zu erreichen ist. Die Formel für Beschleunigung lautet:

$$a = \frac{v}{(t_2 - t_1)^2}$$

Um die entsprechende LSA während der Grünphase zu passieren, muss letztendlich die empfohlene Geschwindigkeit  $v$  eingehalten werden, wobei die maximale Beschleunigung nicht überschritten wird.

## 5 Szenarien im Ampelbereich

Alle in Kapitel 2 angeführten Studien zu Ampelinformationssystemen und Konzepte zu Fahrraderweiterungen haben die Gemeinsamkeit des selbstkontrollierten Fahrverhaltens der FahrerInnen. Ausgesprochen werden lediglich Empfehlungen, die möglichst intuitiv und schnell vermittelt werden. Grundlegend sollte die Anwendung in der Lage sein, die passende Empfehlung oder Handlungsauforderung anzuzeigen, die sich aus folgenden Szenarien ergeben.

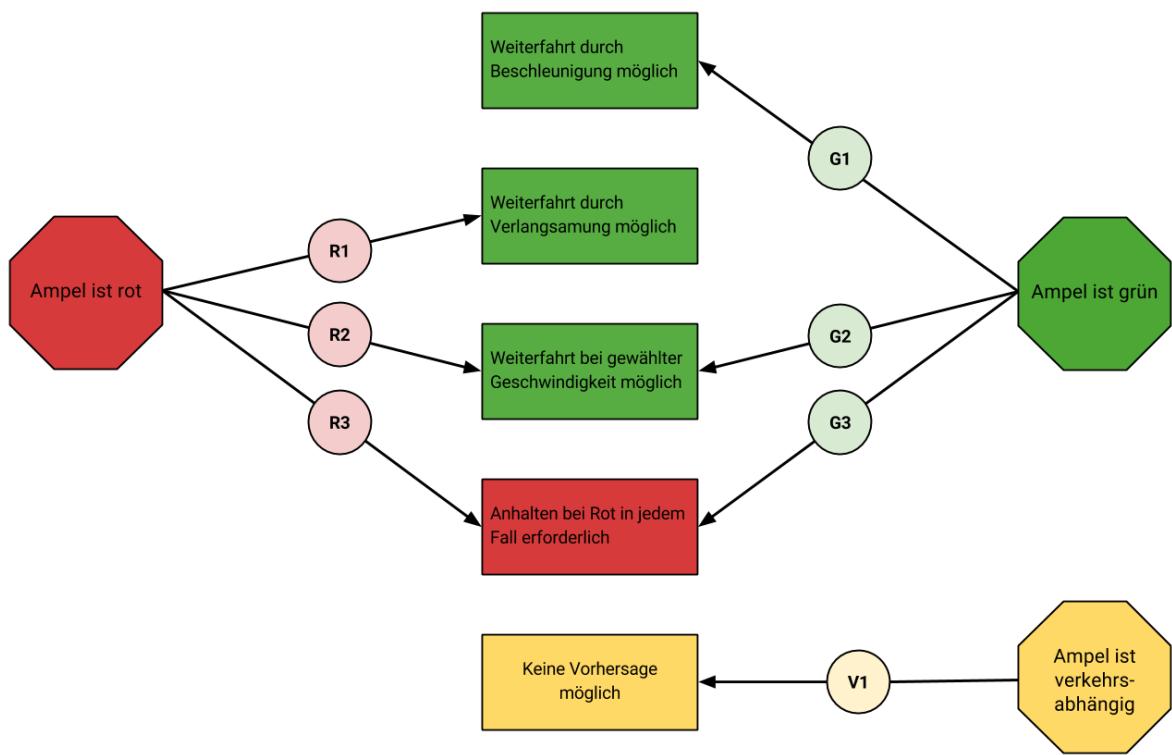


Abbildung 5.1 Szenarien im Ampelbereich

### Szenario R1:

Die Fahrradfahrerin oder der Fahradfahrer nähert sich einer roten Ampel. Die Anwendung zeigt den Countdown der Restrotzeit an und empfiehlt langsamer zu fahren, um die Ampel ohne anzuhalten passieren zu können.

### Szenario R2:

Die Fahrradfahrerin oder der Fahradfahrer nähert sich einer roten Ampel. Die Anwendung zeigt an, es besteht kein Aktionsbedarf und eine Weiterfahrt bei gleichbleibender Geschwindigkeit ist

gewährleistet.

**Szenario R3:**

Die Fahrradfahrerin oder der Fahrradfahrer nähert sich einer roten Ampel. Die Anwendung meldet, das Anhalten bei roter Ampel ist in jedem Fall erforderlich.

**Szenario G1:**

Die Fahrradfahrerin oder der Fahrradfahrer nähert sich einer grünen Ampel. Die Anwendung zeigt den Countdown der Restgrünzeit an und empfiehlt schneller zu fahren, um die Ampel ohne anzuhalten passieren zu können.

**Szenario G2:**

Die Fahrradfahrerin oder der Fahrradfahrer nähert sich einer grünen Ampel. Die Anwendung zeigt an, es besteht kein Aktionsbedarf und eine Weiterfahrt bei gleichbleibender Geschwindigkeit ist gewährleistet.

**Szenario G3:**

Die Fahrradfahrerin oder der Fahrradfahrer nähert sich einer grünen Ampel. Die Anwendung meldet, das Anhalten bei roter Ampel ist in jedem Fall erforderlich, da eine unrealistisch hohe Geschwindigkeit zum Erreichen der grünen Ampel erforderlich wäre.

**Szenario V1:**

Die Fahrradfahrerin oder der Fahrradfahrer nähert sich einer verkehrsabhängigen Ampel. Das bedeutet, die Schaltung ist unregelmäßig und die Wahrscheinlichkeit des Zutreffens der Vorhersage zu gering, eine solche treffen zu können. Die Anwendung zeigt also an, dass es ihr nicht möglich ist eine Vorhersage zu treffen.

Da die Szenarien *R2* und *G2* genau wie die Szenarien *R3* und *G3* zusammengefasst werden können, ergeben sich aus den sieben Szenarien im Ampelbereich die fünf nun aufgezählten Systemzustände.

- Zustand a: Anhalten bei Rot in jedem Fall erforderlich
- Zustand b: Weiterfahrt mit gleichbleibender Geschwindigkeit möglich. Kein Aktionsbedarf
- Zustand c: Weiterfahrt durch Verlangsamung möglich
- Zustand d: Weiterfahrt durch Beschleunigung möglich
- Zustand e: Keine Vorhersage möglich

Im weiteren Verlauf dieser Arbeit wird unter anderem beschrieben wie diese Systemzustände in die Konzeption der zu entwickelnden Anwendung eingebunden werden.

## 6 ANFORDERUNGSDEFINITION

Dieses Kapitel soll helfen eine Vorstellung für die Anforderungen an die Ampelhinweis-App zu bekommen. Im heutigen Technologiezeitalter spielt die Benutzbarkeit und Funktionalität des Produkts eine große Rolle. Es geht nicht nur darum, Interesse zu wecken, sondern auch die NutzerInnen für das Produkt zu begeistern.

Aus den oben genannten Szenarien werden im Folgenden die Anforderungen hergeleitet, die die zu entwickelnde Smartphone-Anwendung erfüllen soll. Für die Umsetzung einer solchen Applikation ist die Positions- und Geschwindigkeitsbestimmung obligatorisch. Im Zusammenhang mit den Ampeldaten, bestehend aus Lage- und Schaltpunkt sollen die notwendigen Berechnungen durchgeführt und deren Ergebnisse als Empfehlung ausgesprochen.

### 6.1 FUNKTIONALITÄT

Die Bestimmung der Position und Geschwindigkeit des Fahrrads ist die zentrale Voraussetzung für eine zeitgerechte Empfehlung. Sie sollte ebenso schnell wie präzise erfolgen, sodass keine Verzögerung der berechneten Ergebnisanzeige eintritt und die Informationen eine hohe Genauigkeit betragen. Von ebenso hoher Relevanz sind die Ampeldaten bestehend aus der genauen Position der LSA und deren Schaltpläne. Da nicht alle LSA automatisch geschaltet sind, sondern einige verkehrsabhängig gesteuert werden – wie zum Beispiel FußgängerInnenampeln, die erst auf Druck aktiviert werden – ist es nicht möglich für diese eine genaue Vorhersage zu treffen. Wird die nächstgelegene Ampel als solche identifiziert, muss die Anwendung sofort signalisieren, dass keine Empfehlung zu erwarten ist. Ebenso unbeachtet bleiben zunächst die Linksabbieger-Ampeln die auf die Hauptampel folgen. Da zuerst die Hauptampel passiert werden muss und die Linksabbieger-Ampel versetzt zu dieser geschaltet ist, ist beim Abbiegewunsch das Anhalten unvermeidbar.

Die meisten Fahrradampeln sind entweder mit gar keiner oder einer sehr kurzen Gelbphase ausgeschaltet. Deshalb wird diese im zunächst nicht beachtet, bzw. aus Sicherheitsgründen als Rotphase behandelt.

#### 6.1.1 BERECHNUNGEN

Abschnitt 4.2 legt die Berechnungsgrundlagen hierfür dar. Die ausgesprochene Empfehlung basiert auf der berechneten Empfehlungsgeschwindigkeit, welche abhängig vom Abstand zu Ampel und der aktuellen Geschwindigkeit ist. Der Abstand zur Ampel wird durch die verbleibende Zeit, bis die Ampel umschaltet, dividiert um die Progressionsgeschwindigkeit zu erhalten.

Die Berechnung bedarf sinnvoller Geschwindigkeitsparameter. Wenn man zu langsam fährt, ist es schwer den Lenker gerade zu halten oder man fällt irgendwann vom Rad. Demnach ist eine Untergrenze von 5 km/h festzulegen. Die Obergrenze richtet sich in erster Linie nach der zulässigen Höchstgeschwindigkeit. Denn auch wenn Fahrräder nicht von den allgemeinen Geschwindigkeitsbegrenzungen der StVO nach §3 Abs. 3 der StVO betroffen sind, gelten per §3 Abs. 1 der StVO die per Schild angeordneten Geschwindigkeiten. Eine Höchstgeschwindigkeit von 30 km/h soll als Ausgangspunkt für die zu implementierende Anwendung dienen. Gegebenenfalls kann diese später individuell angepasst werden.

Ebenfalls zu beachten ist die aufzubringende Beschleunigung. Laut [Bes] liegt die mögliche Beschleunigung bei FahrradfahrerInnen zwischen ein und zwei Metern pro Sekunde zu Quadrat. Wird die berechnete Geschwindigkeit mit der aktuellen und den Begrenzungsparametern ins Verhältnis gesetzt, ist die Empfehlung abzuleiten.

Die Prognose findet statt, sobald die Ampel circa 300 Meter voraus liegt. Bei einer größeren Entfernung von zum Beispiel einem Kilometer kann die Gefahr der Ablenkung bestehen. Sollte die Anwendung auf der Strecke durchgängig vorschlagen schneller zu fahren, könnte die fahrende Person dies als Ansporn sehen und so nicht mehr auf den Verkehr achten.

Bei dieser doch recht kurzen Entfernung genügt für den zu entwickelnden Prototyp der Abstand als Berechnungsbestandteil in Luftlinie. Bei Optimierungsbedarf bietet es sich hierfür an, Straßendaten einzubinden, um den genauen Abstand ermitteln zu können. Hierbei werden dann beeinflussende Elemente wie Hügel oder Kurven berücksichtigt.

Außerdem muss ermittelt werden, welche Ampel die nächste ist um dann die Entfernung zu dieser zu berechnen. Da keine Straßendaten vorliegen werden zunächst alle Ampeln im genannten Radius von 300 Metern geortet und deren Distanz errechnet. Um die Richtung zu bestimmen, wird die Distanz zu jeder LSA und nach der Zurücklegung von einigen Metern erneut ermittelt und mit dem vorherigem Wert verglichen. Die Ampel bei der sich die Distanz verringert ist die gesuchte.

### 6.1.2 DATENGRUNDLAGEN

Die Positionsdaten, Information der Verkehrsabhängigkeit und Schaltpläne, bestehend aus geltender Tage, Start- und Endzeit des Schaltplans, sowie der Start- und Endzeit der Grünphase befinden sich in einer JavaScript Object Notation (JSON)-Datei, welche im internen Speicher abgelegt ist, wodurch sich Abfragen zu externen Servern, die eine Verbindung mit einem Netzwerk voraussetzen erübrigen. Diese sollte allein aus Performancegründen keinen Internetzugriff voraussetzen, da ein direkter Zugriff auf den internen Speicher nicht den Umweg über einen Server gehen muss. Ist das Datenvolumen aufgebraucht, ist die Netzwerkverbindung zu langsam, die benötigten Daten zeitnah anzufragen und zu erhalten.

Die Ampeldaten müssen manuell erfasst und in ein sinnvolles Format gebracht werden. Es wurde entschieden, die Ampeldaten in einer Datei händisch aufzubereiten. Da sowohl die Ampelpositionsdaten als auch die Ampelschaltpläne statisch sind, werden sie nur gelesen und nicht verändert, wofür bei der begrenzten Anzahl von Ampeln auf der Teststrecke keine Datenbank vonnöten ist. Weiter braucht eine Datei weder Treiber noch Bibliothek, was den Overhead<sup>1</sup> gering hält. Bei einer ggf. späteren Einbindung des gesamten Stadtbildes oder der Erweiterung über zusätzlicher Städte ist über ein optimiertes Datenmodell nachzudenken.

Trotz des erhöhten Stromverbrauchs bei der Verwendung von GPS ist es von Vorteil auf die netzwerkgestützte Standortbestimmung zu verzichten, da die Genauigkeit dieser Verfahren nicht so präzise ist. In der zu entwickelnden Anwendung kommt es auf jeden Meter an. Ob die Genauigkeit des integrierten GPS-Sensors genügt, muss ergo im Rahmen dieser Arbeit getestet werden.

Eine Möglichkeit dem hohen Akkuverbrauch von GPS entgegenzuwirken wäre die Verwendung eines externen GPS-Geräts, das über Bluetooth mit dem Smartphone verbunden ist und so dessen Akkulaufzeit verlängert. Außerdem ist die Qualität des GPS-Empfängers höher als die des im Smartphone verbauten und kann somit eine präziesere Ortung erlangen. (Vgl. [FA11] S. 28) Der Nachteil hierbei ist die Notwendigkeit eines zweiten Gerätes das vermutlich an bestimmte Smartphones nicht angeschlossen werden kann.

---

<sup>1</sup> Zusatzinformationen

## 6.2 DIE GRAPHISCHE OBERFLÄCHE

Ebenso wie die Bestimmung der Position und Geschwindigkeit spielt auch die graphische Darstellung eine große Rolle. Gerade bei dem Gebrauch im Verkehr ist die Eindeutigkeit und schnelle Erfassbarkeit der zu übermittelnden Informationen bedeutend. Um nicht von dem Verkehr abzulenken muss die Information, sowohl in Bedeutung als auch in Darstellung auf einen kurzen Blick erkennbar sein. Demnach sollte die Oberfläche möglichst einfach gehalten werden.

Da die Anwendung während der Fahrt nicht bedienbar sein muss – denn auch das würde vom Verkehr ablenken – müssen keine Bedienelemente eingeblendet werden und es besteht mehr Raum für die Informationsanzeige. Es ist davon auszugehen, dass die Anwendung ausschließlich im Freien Gebrauch findet. Dies und die dort herrschenden wechselnden Helligkeiten, zum Beispiel bei der Fahrt durch Schatten, setzen eine Verwendung von hohen Kontrasten voraus.

Weiter muss das automatische Abschalten des Gerätebildschirms deaktiviert werden, sodass sich dieser auch ohne Interaktion nicht ausschaltet und die Anwendung den FahrerInnen jederzeit unterstützende Fahrweisungen geben kann.

### 6.2.1 EMPFEHLUNGEN

Im Allgemeinen sollte die Anwendung in der Lage sein die sich aus Kapitel 5 resultierenden vier Systemzustände zu visualisieren und gegebenenfalls eine entsprechende Empfehlung auszusprechen. Es muss ausdrücklich darauf hingewiesen werden, dass es sich bei den Empfehlungen um eben solche handelt und man diese nur befolgen darf, wenn die Umstände und die Verkehrsregeln es erlauben. Ebenso sollte darauf hingewiesen werden, nicht zu lange auf das ausführende Gerät zu schauen und die Aufmerksamkeit hauptsächlich dem zu widmen.

Zur Verdeutlichung und einfacher Erfassbarkeit der angezeigten Empfehlungen bietet es sich an, die entsprechenden Farben zu nehmen. Also ist für Zustand *b*, der das Erreichen der Grünen Welle ohne Veränderung der eigenen Geschwindigkeit steht Grün und für den Zustand *a*, der ausdrückt dass keine Weiterfahrt ohne an roter Ampel Anhalten zu müssen möglich ist, Rot zu verwenden. Die Zustände *c* und *d* drücken aus, ein Erreichen der Grünen Welle ist möglich, jedoch nur bei Veränderung der eigenen Fahrgeschwindigkeit. Zur inhaltlichen Differenzierung ist eine andere Farbe zu wählen, sodass bereits aus dem Augenwinkel eine Handlungsempfehlung erkennbar ist. Dahingehend soll ersichtlich sein, ob das Tempo leicht oder stark erhöht bzw. verlangsamt werden muss. Wird die Empfehlung der leichten Geschwindigkeitverlangsamung ausgesprochen, muss auch das ersichtlich sein.

Unterstützend zur Handlungsaufforderung kann ein Countdown angezeigt werden, der die Dauer der jeweiligen Ampelphase herunterzählt.

## 7 KONZEPTION

Die erarbeiteten Anforderungen an ein Ampelinformationssystem für FahrradfahrerInnen werden in diesem Kapitel für die Konzeption angewendet. Beginnend mit dem Aufbau der Anwendung werden in den folgenden Abschnitten das Design, die von der Anwendung genutzten Daten, Anwendungsfälle, die Architektur und schließlich die Komponenten der Entwicklungsumgebung, welche eingesetzt werden aufgeführt.

### 7.1 ANWENDUNGS AUFBAU

Die Anwendungsarchitektur verwendet die Vorgaben für Android-Applikationen. So wird die Hauptkomponente mit Hilfe sogenannter Activities realisiert. Als Basisklasse definiert eine Activity das User Interface (UI) einer mobilen Anwendung, stellt also die Ansicht mit der BenutzerInnen interagieren können bereit.

Eine Activity wird von der Bibliotheksklasse `android.app.Activity` abgeleitet, als Klasse implementiert und muss in der Manifestdatei der App aufgeführt werden. [anda]

In der zu entwickelnden Fahrradapplikation wird keine Navigation innerhalb der Anwendung voneinander sein, weshalb nur eine Activity implementiert wird.

### 7.2 DATENGRUNDLAGE

Neben den Daten der eigenen Position ist die Datengrundlage der Ampeln von höchster Priorität. Für die Umsetzung des Prototyps wurde im Rahmen dieser Arbeit als Ausführungsstelle die Stadt Plau am See in Mecklenburg-Vorpommern festgelegt.

Die Vorteile der Anwendung im ländlichen Bereich liegen in der Verkehrsarmut und die daraus resultierende geringe Menge an Ampeln. Dieser Umstand ermöglicht die Erfassung sämtlicher Ampeln der Stadt, was ein überschaubares Testbild gibt. Weiter gibt es keine Straßenbahnen oder Busse, welche den Verkehr durch Grünanforderung beeinflussen. Dadurch lässt sich die Anzahl der Ampeln, bei denen durch Verkehrsabhängigkeit eine Vorhersage möglich ist, minimieren.

## 7.2.1 AMPELDATEN

Die Ampeldaten werden manuell erfasst, strukturiert und im JSON-Format als Datei auf dem Mobilgerät abgelegt. Um einen Überblick zu gewähren, werden die wichtigsten Komponenten skizziert.

Jede Kreuzung hat unterschiedliche Schaltpläne, die zu festgelegten Tageszeitabschnitten greifen. Die Tabelle 7.1 zeigt, wie ein solcher Tagesplan einer Kreuzung aussehen kann, wobei an anderen Tagen die Uhrzeiten sowie die Reihenfolge, Quantität und Dauer der Zeitschaltprogramme variieren können.

Für jedes Schaltzeitprogramm existiert ein Signalschaltzeitplan der Informationen über die Ampelbezeichnung der Kreuzung und deren Rot-, Grün, Gelb, Rotgelb und Dunkelphasen. In letzterer ist die LSA ausgeschaltet, was zum Beispiel nachts häufiger der Fall ist.

Die Abbildung 7.1 zeigt auf, wie so ein Schaltzeitplan für eine Kreuzung mit vier Ampeln aussehen kann. Der Buchstabe R in der Bezeichnung steht hierbei für “Radfahrrampel“.

BEZEICHNUNG DER KREUZUNG	
Uhrzeit	Schaltprogramm
05:00	Früh
11:00	Tag
17:00	Spät
23:00	Nacht

Tabelle 7.1 Zeitschaltplan

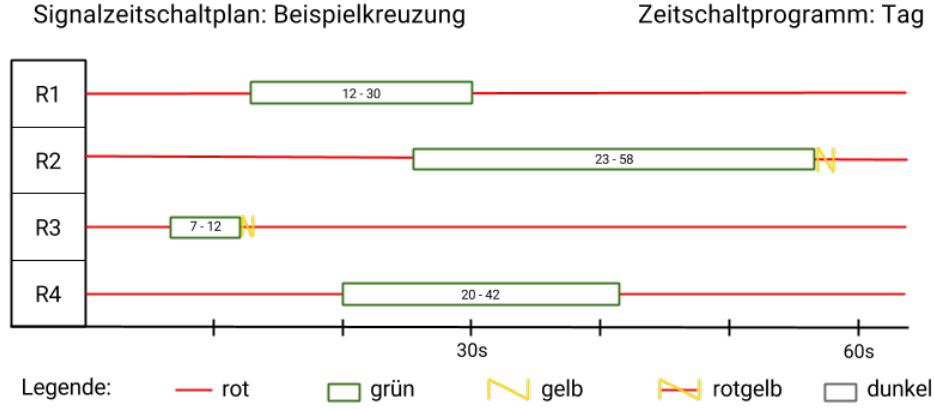


Abbildung 7.1 Beispielhafter Signalschaltplan einer Kreuzung

In dem oben gezeigtem Beispiel steht demnach die Ampel R1 zwischen 11 und 17 Uhr in jeder Minute für die Sekunden 12 - 30 Grün und den Rest der Zeit Rot.

Auch die Position jeder Ampeln wird händisch erfasst und in das WGS84-Format konvertiert, um dann zusammen mit den Phasendaten in das gewählte JSON-Format gebracht zu werden.

Als Referenzzeit ist die Zeit zwischen 14 und 20 Uhr von Freitag bis Montag gewählt. Diese kann selbstverständlich bei Bedarf ausgedehnt bis vervollständigt werden.

## 7.2.2 AMPELOBJEKT IM JSON-FORMAT

JSON ist ein leichtgewichtiges, auf JavaScript basierendes, Datenaustauschformat, das für Menschen gut les- und schreibbar und für Maschinen einfach analysier- und generierbar ist. Ein JSON-Objekt besteht aus einer von zwei Grundstrukturen. Eine Auflistung von Schlüssel/Wert Paaren oder eine geordnete Liste von Werten, als Array realisiert. Der folgende Codeabschnitt zeigt beispielhaft ein JSON-Ampelobjekt:

---

```
1 {
2     "name" : "SteinstrA1",
3     "lat": 52.552918,
4     "lon": 13.423992,
5     "dependsOnTraffic": false
6     "timetable" : [
7         {
8             "days": ["Mo", "Di", "Mi"] ,
9             "timeFrom": 14,
10            "timeTo": 20,
11            "greenFrom": 22,
12            "greenTo": 27
13        },
14        {
15            "days": ["Do", "Fr"] ,
16            "timeFrom": 13,
17            "timeTo": 21,
18            "greenFrom": 2,
19            "greenTo": 27
20        }
21    ]
22 }
```

Listing 7.1 JSON-Ampelobjekt

---

Das hier aufgeführte Ampelobjekt hat den Namen `SteinstrA1`, welcher auf die Kreuzung auf der sich die Ampel befindet schließen lässt. Es folgen die zwei Werte `lat` und `lon`, die die genaue Position der LSA beschreiben und der boolsche Wert `dependsOnTraffic`, der auf `false` gesetzt ist, sobald die Ampel verkehrsunabhängig ist. Im anderen Fall ist sie verkehrsabhängig und fällt somit aus den Berechnungen heraus, benötigt also auch keine weiteren Daten. Das Array `timetable` beinhaltet Schaltplanobjekte mit einem Array aus Tagen und der Start-und Endzeiten an denen diese gelten. Die Werte `greenFrom` und `greenTo` bezeichnen den Start und das Ende jeder Grünphase des Schaltplans. Da als Referenzzeit die Zeit zwischen 10 und 16 Uhr gewählt wurde, werden die Schaltpläne, welche diese nicht betreffen nicht aufgenommen.

## 7.3 ANWENDUNGSFÄLLE

Aus den in Kapitel 6 beschriebenen Anforderungen und den in Kapitel 5 erarbeiteten Szenarien ergeben sich die folgenden fünf Use-Cases, die von der Anwendung erfüllt werden sollen.

ID	Anwendungsfall	Beschreibung
<b>UC2</b>	Ich fahre langsamer, um die grüne Ampel zu passieren	Der Countdown der aktuellen Ampelphasendauer wird sekündlich aktualisiert und zusätzlich zur Aufforderung langsamer zu fahren angezeigt
<b>UC1</b>	Ich fahre schneller, um die grüne Ampel zu passieren	Es wird eine Beschleunigungsaufforderung ausgesprochen. Unterstützend wird der Countdown der aktuellen Ampelphasendauer sekündlich aktualisiert und zusätzlich angezeigt
<b>UC3</b>	Ich halte meine Geschwindigkeit, um die grüne Ampel zu passieren	Die aktuelle Geschwindigkeit ist genauso hoch wie die berechnete Empfehlungsgeschwindigkeit. Es wird angezeigt, dass kein Handlungsbedarf besteht, also das Tempo gehalten werden kann.
<b>UC4</b>	Ich muss in jedem Fall bei roter Ampel anhalten	Die berechnete Empfehlungsgeschwindigkeit übersteigt die festgelegte Höchstgeschwindigkeit oder die zu erreichende Beschleunigung ist in dem Maße nicht zu erreichen, also ist die Distanz zur Ampel zu lang. Eine Anzeige mit dem Signal “keine Weiterfahrt ist möglich“ erscheint.
<b>UC5</b>	Es ist nicht möglich eine Vorhersage zu treffen	Eine verkehrsabhängige Ampel mit individueller Steuerung nähert sich. Es ist dem System nicht möglich eine wahrscheinliche Vorhersage zu treffen und so die entsprechende Empfehlung auszusprechen. Es wird <i>ein gelbes Fragezeichen, symbolisch dafür</i> angezeigt.

Tabelle 7.2 Anwendungsfälle

Das in Abbildung 7.2 gezeigte Use-Case-Diagramm veranschaulicht die in der obigen Tabelle 7.2 aufgeführten Anwendungsfälle.

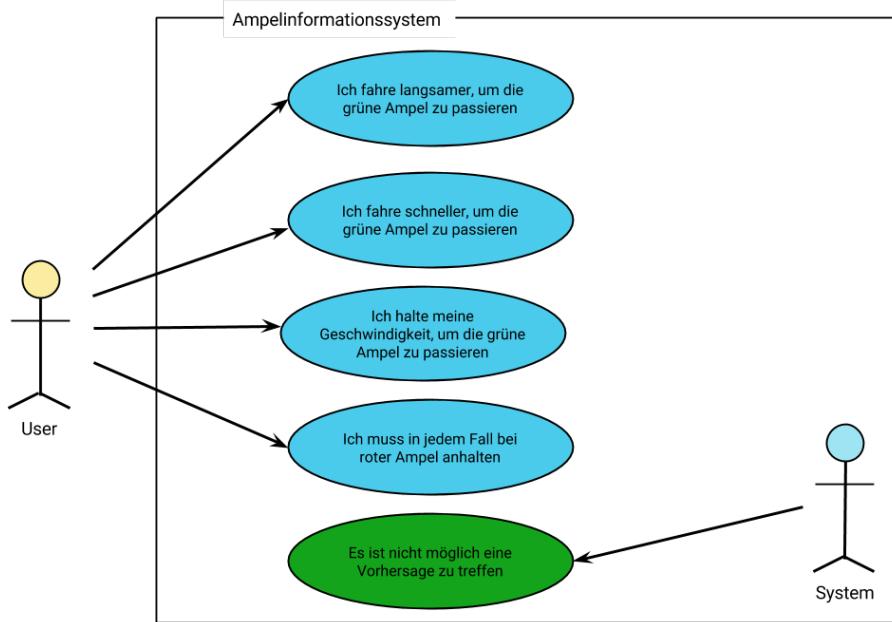


Abbildung 7.2 Use-Case-Diagramm

## 7.4 KLASSENARCHITEKTUR

Die zu erstellende Anwendung erhält den Namen AIS, was für Ampel-Informationssystem steht und der vorläufige Name der Anwendung ist und besteht aus den nachfolgend beschriebenen Klassen.

Die **MainActivity** ist die Hauptklasse der Android-Applikation. Sie definiert die grafische Oberfläche der Anwendung und stellt die notwendigen Methoden bereit.

Der **JSONParser** liest die erstellte JSON-Datei ein und wandelt deren Inhalt in ein Array um. Dieses Beinhaltet Ampelobjekte mit deren Namen, Position und der Information darüber, ob sie verkehrsabhängig sind oder nicht. Sind sie es nicht, wird außerdem zu jeder Ampel ein Array, bestehend aus den Sigalschaltplanobjekten gespeichert. Diese enthalten benötigten alle Informationen über den Gel tungszeitraum und der Grünphasen.

**GPSTracker** implementiert die Schnittstelle **LocationListener** und kann so kontinuierlich die Position des Endgeräts verfolgen. Mit der eigenen Position und der aller Ampeln aus dem vom **JSONParser** erstellten Array ermittelt er außerdem die nächste Ampel. Ist diese identifiziert, wird über den **OnSetListener** die Klasse **SpeedHandler** aktiviert, welche anhand der Daten aus dem Signalschaltplanobjekt den Countdown der Rotanzeige und die optimale Geschwindigkeit berechnet und die Anzeige entsprechend aktualisiert.

Das nachfolgende Klassendiagramm gibt eine Übersicht über die oben beschriebenen Klassen.

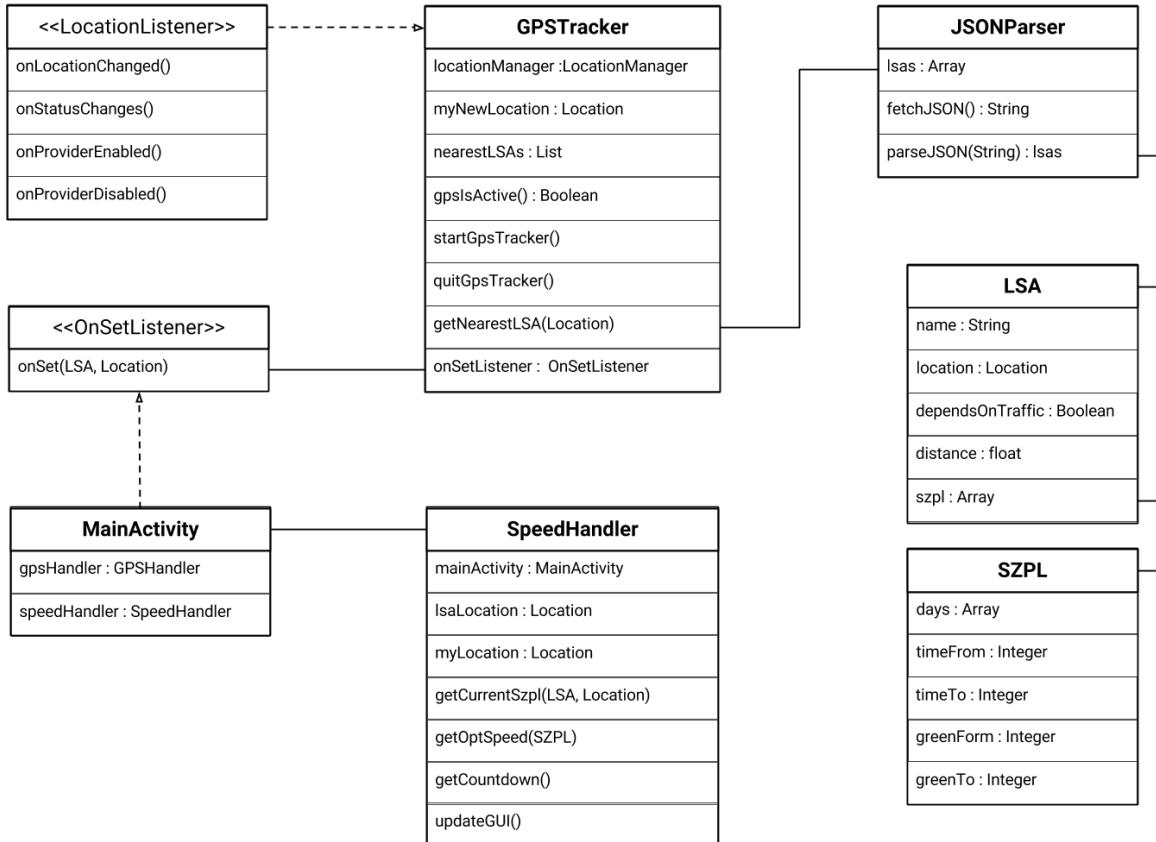


Abbildung 7.3 Klassendiagramm

## 7.5 DIE GRAPHISCHE OBERFLÄCHE

Aus den Anforderungen an die graphische Oberfläche in Kapitel 6 entsteht das Design. Durch die überschaubare Anzahl an Funktionalitäten kann der Aufbau einfach gehalten werden. Aus den Empfehlungsanzeigen, welche den Systemzuständen entsprechen sind vier Ansichten abzuleiten. Die folgende Abbildung zeigt den Entwurf der Benutzeroberfläche. Abbildung 7.4a zeigt die grafische Umsetzung des Zustands *a*, in dem die Ampelschaltung keine Weiterfahrt ermöglicht. Es wird ein großer roter Kreis mit einem schwarzen Kreuz verwendet. Rot ist eine Signalfarbe und steht auch bei Ampeln für “Halt” oder “Stop”. Auch das Kreuz wird häufig als *verweigerndes* Symbol eingesetzt und ist somit intuitiv als solches erkennbar.

Abbildung 7.4b setzt die Visualisierung des Zustands *b* um, bei dem man mit der aktuellen Geschwindigkeit in der Grünen Welle fährt, um. Hier wird ein großer grüner Kreis verwendet, in der Mitte steht “ok”. Durch die nicht alleinige Benutzung der Ampelfarben Rot und Grün ist die Ansicht auch für Menschen mit einer Rot-Grün-Sehschwäche eindeutig interpretierbar.



Abbildung 7.4 Entwurf des Designs anhand der Systemzustände

Die Abbildungen 7.4c und 7.4d für die Zustände *c* und *d* sind in der Anordnung der Elemente identisch. Mittig im Bild ist eine eingekreiste rote Zahl die den Countdown der Restrotanzeige darstellt. Sie wird also mit jeder Sekunde aktualisiert. Ober- und unterhalb des Countdowns befinden sich jeweils zwei hellblaue Pfeile. Je nach Differenz der aktuellen zur berechneten Geschwindigkeit sind ein oder zwei Pfeile ausgefüllt. Wird also empfohlen viel schneller zu fahren, sind die oberen Pfeile aktiv, bei der Aufforderung etwas das Tempo zu drosseln der untere. Es sind immer alle Pfeile zu sehen. Durch die Anzeige “ein von zwei“ ist die Bedeutung der Geschwindigkeitsstufen klarer. Als Farbe für die Pfeile wurde Hellblau gewählt. Sie unterscheidet sich sowohl farblich als auch in der Helligkeitsstufe zu den anderen Farben und steht im hohen Kontrast zu dem Hintergrund. Auf schwarzem Hintergrund wirken die Farben intensiver und sind so auch aus dem Augenwinkel leichter zu erkennen und unterscheiden. Die in Abbildung 7.4e gezeigte Darstellung erscheint, wenn keine Vorhersage möglich ist weil die Ampel verkehrsabhängig und die Vorhersage dadurch zu ungenau wird. Als Farbe wurde die dritte Ampelfarbe Gelb gewählt, was weder Rot für Anhalten, noch grün für Weiterfahren ist und so signalisiert, die Anwendung kann keine Empfehlung aussprechen. Das mittig platzierte Fragezeichen unterstützt die Aussage.

Die aktuelle Geschwindigkeit wird bewusst nicht angezeigt. Auch nicht die Progressionsgeschwindigkeit. Die Differenz ist beim Fahrrad nicht so hoch wie bspw. im Auto. Bei einer Varianz von wenigen km/h genügt die Anzeigevariante “schneller“ und “noch schneller“.

## 7.6 TESTFÄLLE

Folgende Testfälle werden während der Entwicklung stetig durchgeführt. Das erfolgreiche Bestehen dieser Tests ist eine notwendige Qualitätseigenschaft der zu entwickelnden Applikation.

**Sicherheitshinweis:**

Die Anwendung muss nach jeder Installation auf den Vorrang der Verkehrssicherheit hinweisen und auf die Bestätigung des Nutzers oder der Nutzerin warten.

**Anwendung starten:**

Die Anwendung muss sich zu jedem Zeitpunkt starten lassen.

**Anwendung beenden:**

Die Anwendung muss sich zu jedem Zeitpunkt beenden lassen.

**Einlesen der Datei:**

Die am nächsten gelegene Ampel und deren Rot- oder Grünphase kann nur ermittelt werden, wenn die Daten richtig eingelesen werden. Auch kann der Countdown der Restrotanzeige nur mit den Daten des Schaltplans angezeigt werden.

**Ermittlung der Position:**

Die Empfehlung der Geschwindigkeit kann nur dann korrekt angezeigt werden, wenn die Position des Geräts ermittelt wurde. Weiter wird der eigene Standort neben dem der Ampel für die Berechnung der am nächsten gelegenen LSA benötigt.

## 7.7 ENTWICKLUNGSUMGEBUNG

Für die Erstellung der Smartphone-Applikation wurde folgende Soft- und Hardware verwendet:

**SOFTWARE**

- Android Studio<sup>1</sup> Version 1.1. Enthält Android Studio IDE, Android SDK-Tools, Android 5.0 Plattform, Android 5.0 Emulator System Image mit Google APIs
- git, Version 1.7.10.4

**HARDWARE**

- Lenovo Thinkpad W520 (Intel® Core™ i7-2630QM, 2,00GHz, 4GB RAM) als ersten Entwicklungsrechner (Betriebssystem: Debian<sup>2</sup> 7.8, 64-bit-Version)
- Lenovo Thinkpad X200 (Intel® Core™ i7-2630QM, 2,00GHz, 8GB RAM) als zweiten Entwicklungsrechner (Betriebssystem: Debian 7.8, 64-bit-Version)
- Android Testgeräte: Samsung Galaxy Note 2, Samsung Galaxy Nexus, Samsung Nexus S, LG Nexus 4, LG Nexus 5, Motorola RAZR Maxx, HTC Desire HD

---

<sup>1</sup> Download unter <http://developer.android.com/sdk/index.html>

<sup>2</sup> Download unter <https://www.debian.org/index.de.html>

# 8 DER PROTOTYP

In diesem Kapitel wird die nach dem in Kapitel 7 präsentierten Lösungswegs die detaillierte Beschreibung der technischen Realisierung der Anwendung vorgestellt.

Nach der Erklärung der Konfigurationsdateien wird auf die Umsetzung der Szenarien eingegangen. Im Zuge dessen werden die implementierten Algorithmen vorgestellt, wobei sich der erste mit dem Auffinden der nächsten relevanten Ampel befasst und der zweite die empfohlene Geschwindigkeit berechnet.

## 8.1 DIE MANIFEST- UND BUILD.GRADLE-DATEI

Das Android-Manifest dient der Festlegung wichtiger Eigenschaften der Anwendung und gehört zu jedem Android-Projekt. Die XML-Datei (`AndroidManifest.xml`) ist im Hauptverzeichnis des Projekts zu finden und im Listing 8.1 abgebildet. In der zweiten Zeile wird hier der Paketname des Programms festgelegt. Im `application`-Tag werden Variablen gesetzt, die das in dargestellte Icon und den Namen der Anwendung definieren. Darüber hinaus wird hier die Activity der Applikation definiert. Zuerst wird der Name der Activity gesetzt. Die Variable `screenOrientation` legt das Format der Anzeige fest und verhindert ein automatisches Drehen des Bildschirms.

---

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2   package="com.example.cobi.ais">
3     <application
4       android:allowBackup="true"
5       android:icon="@drawable/ic_launcher"
6       android:label="@string/app_name"
7       android:theme="@style/AppTheme" >
8       <activity
9         android:name=".MainActivity"
10        android:screenOrientation="portrait">
11          <intent-filter>
12            <action android:name="android.intent.action.MAIN" />
13            <category android:name="android.intent.category.LAUNCHER" />
14          </intent-filter>
15        </activity>
16      </application>
17    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
18  </manifest>
```

Listing 8.1 AndroidManifest.xml

Im **intent-filter**-Tag dass diese Activity beim Start der App ausgeführt wird. Hätte die Anwendung über mehrere Activities implementiert, würden die anderen ebenfalls hier aufgeführt werden. Unterhalb des **application**-Tags, in Zeile 17, werden nun die Berechtigung des GPS-Zugriffs der Applikation, um Standortdaten, also die jeweiligen geographischen Kordinaten des Endgeräts zu beziehen gesetzt.

Android Studio-Projekte enthalten eine Top-Level-Build-Datei und eine Build-Datei für jedes Modul. Die Build-Dateien heißen **build.gradle** und sind einfache Textdateien, die die Groovy<sup>1</sup>-Syntax verwendend. Hier lässt sich der Build mit den Elementen, welche vom Android Plugin Gradle<sup>2</sup> unterstützt werden, konfigurieren. [andc]

Listing 8.2 zeigt einen Ausschnitt aus der Build-Datei für die Anwendung.

Die Eigenschaft **compileSdkVersion** beschreibt mit welcher Android SDK Version kompiliert werden soll und **buildToolsVersion** gibt an, welche Version des Build-Tools verwendet wird.

---

```
1 compileSdkVersion 21
2 buildToolsVersion "21.1.2"
3 defaultConfig {
4     applicationId "com.example.cobi.ais"
5     minSdkVersion 9
6     targetSdkVersion 21
7     versionCode 1
8     versionName "1.0"
9 }
```

Listing 8.2 Auszug aus der **build.gradle**-Datei

---

Es folgt das **defaultConfig** Element, das die Kerneinstellungen und fügt diese dynamisch aus dem Build-System in die Manifestdatei ein. Hier wird unter anderem festgelegt, für welche Android Versionen die Anwendung geschrieben wurde (**targetSdkVersion**) und das minimale API-Level<sup>3</sup> der Anwendung (**minSdkVersion**).

Letzteres ist auf “10“ gesetzt und zeigt, dass die Anwendung auf Mobilgeräten auf denen mindestens die Android-Version 2.3.3 installiert sein muss. Die Android-Version wurde auf 2.3.3 festgelegt, da das die Mindestanforderung für das editieren der SharedPreferences, welche für das einmalige Anzeigen des Warndialogs benötigt werden, ist. Damit werden trotzdem noch 99,6% der Geräte, die auf den Google-Play-Store zugreifen können, abgedeckt. [andj]

---

<sup>1</sup> Programmier- und Skriptsprache

<sup>2</sup> Gradle ist ein auf Java basierndes Build-Management-Automatisierungs-Tool.

<sup>3</sup> Eine Übersicht über die API-Level und der dazugehörigen Android-Version findet sich unter <http://developer.android.com/guide/topics/manifest/uses-sdk-element.html> – Zugriff: 03.03.2015

## 8.2 MAINACTIVITY-KLASSE

Die Klasse `MainActivity` ist die Activity, die beim Start der Anwendung ausgeführt wird. Hier werden die Anzeigeelemente initialisiert und die zentralen Funktionalitäten delegiert.

Die Methode `onCreate` ist die erste Methode im Activity-Lebenszyklus<sup>4</sup> und wird direkt nach dem Start der Activity ausgeführt. Hier werden die Anzeigeelemente gesetzt und nach jeder Installation ein Dialog erstellt, der auf die oberste Priorität der Straßenverkehrsordnung hinweist.

Es wird eine Instanz des `JSONParser`-Objekts erstellt, welche die JSON-Datei aus dem Ressourcenordner liest und in ein Objektarray umwandelt. Neben der `SpeedHandler`-Objektinstanz, welche für die Geschwindigkeitsempfehlung und dessen Anzeige zuständig ist, wird die des `GPSTracker` erstellt. Auf diesen ist der `OnSetListener` registriert, welcher ein Ereignis wirft sobald er die nächstgelegende Ampel gesetzt hat. Sobald dieser Fall eingetreten ist, wird der `SpeedHandler` beauftragt, den dazugehörigen Signalschaltplan zu holen, um dann anhand dieser Daten die Berechnungen durchzuführen und deren Ergebnisse anzuzeigen.

## 8.3 UMSETZUNG Szenarien

Wie in Kapitel 5 beschrieben, lassen sich die sieben möglichen Szenarien auf insgesamt fünf komprimieren, da die Szenarien R2 und G2, genauso wie die Szenarien R3 und G3 dasselbe Ergebnis haben. Die zusammengefassten fünf Szenarien eignen sich für eine prototypische Umsetzung, welche im Folgenden beschrieben wird.

### 8.3.1 EINLESEN DER AMPELDATEN

Neben der Geräteposition bilden die Ampelposition und deren Signalschaltpläne die Grundlage der Anwendung. Hierbei ist das korrekte Einlesen und Auswerten der Daten obligatorisch. Die Aufgabe des `JSONParsers` ist es also, die manuell erstellte JSON-Datei, welche die Ampeldaten beinhaltet, richtig zu konvertieren. Die Klasse `JSONParser` liest also die Datei ein und wandelt dann die enthaltenen JSON-Ampelobjekte in Java-Ampelobjekte um.

Hierfür wird das JSON-Array durchlaufen und für jedes enthaltende Objekt ein Ampelobjekt erzeugt. Aus den Werten `lat` und `lon`, stehend für `latitude` und `longitude`, wird ein `Location`-Objekt erzeugt, das als Attribut gesetzt wird. Der boolsche Wert `dependsOnTraffic` ist auf `true` gesetzt, sobald die LSA verkehrsabhängig ist. Ist dies der Fall, ist eine Vorhersage der Ampel aufgrund beeinflusster Parameter nicht möglich und es wird ein Objekt mit den beiden genannten Attributen und dem bezeichnenden Namensattribut erzeugt. Andernfalls werden die zu der Ampel hinzugehörigen Schaltpläne durchlaufen und für jeden ein selbiges Objekt erzeugt, welche in einem Array gespeichert dem Ampelobjekt als weiteres Attribut übergeben wird. Es ist zu beachten, dass jeder Signalschaltplan über ein Array mit Tagen verfügt. Denn ein Schaltplan kann an mehreren Tagen gültig sein. Die Tage

<sup>4</sup> <http://developer.android.com/training/basics/activity-lifecycle/starting.html> – Zugriff: 02.03.2015

sind als Strings gespeichert und werden für die weitere Verwendung in Integer umgewandelt, wobei die Woche am Sonntag beginnt. Der Sonntag wird also als 1 gespeichert, der Montag als 2 und so weiter.

### 8.3.2 ERMITTlung DER NÄCHSTEN AMPEL

Zur Ermittlung der nächsten Ampel ist zunächst die eigene Position vonnöten. Dazu stellt der in dem `android.location`-Paket enthaltene `LocationManager` die entsprechenden Schnittstellen bereit. Für den Empfang der GPS-Daten implementiert die Klasse `GpsTracker` das `LocationListener` Interface und registriert sich beim `LocationManager` für GPS-Updates. Letzteres ist im Listing 8.3 abgebildet. Die Methode erwartet neben dem `ProviderType` und dem zu benachrichtigenden Listener (`this`) zwei weitere Parameter. Der erste zeigt an nach wie vielen Millisekunden ein Update gesendet werden soll, der zweite dass ein Update nur gesendet werden soll, wenn sich die Position um die angegebene Distanz verändert hat.

---

```
1 // Registrieren fuer GPS Updates (alle 3 sekunden, nach 5 Metern)
2 locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
3                                         3000, Constants.MY_DISTANCE, this);
```

Listing 8.3 Registrierung für GPS-Updates in `GpsTracker.java`

---

Zusammen mit dem `LocationListener` wird unter anderem die Methode `onLocationChanged`, welche aufgerufen wird sobald neue Positionsdaten vorhanden sind implementiert. Das hier gespeicherte `Location`-Objekt liefert neben dem Längen- und Breitengrad auch die Genauigkeit des GPS-Signals in Metern und Geschwindigkeit in Metern pro Sekunde. Sofern die Genauigkeit nicht zu gering ist, wird mit ihm wird die Methode `getNearestLSA` aufgerufen, welche die nächste Ampel bestimmt.

Die Methode `getNearestLSA` ermittelt zunächst alle Ampeln im festgelegten Radius und speichert diese zusammen mit der Distanz zum Gerät in einer temporären Liste. Dann werden diese wieder durchlaufen und die Distanzen verglichen. Die Ampel, zu der sich die Entfernung zum Gerät verringert und die kleinste Distanz zum Gerät hat, ist die Gesuchte. Ist die nächste Ampel ermittelt, kann der `OnSetListener` der `MainActivity` diese zusammen mit der Geräteposition übergeben.

Vergrößert sich die Distanz zu der LSA wieder befindet sie sich außerhalb des durchsuchten Radius bedeutet das, dass der Fahrer oder die Fahrerin sich von der Ampel oder der Kreuzung entfernt. Also werden diese dann zusammen mit den anderen gespeicherten Ampeln gelöscht.

```

1 // wurden schon Ampeln gespeichert?
2 if (nearestLSAs == null ) {
3     nearestLSAs = new ArrayList<LSA>();
4
5     for (LSA lsa : lsas) {
6         // fuer saemtliche Ampeln die Distanz errechnen
7         distance = myLocation.distanceTo(lsa.getLsaLocation());
8
9         // ist die Distanz im Radius?
10        if (distance <= Constants.MIN_LSA_DISTANCE) {
11            // Ampel mit Distanz speichern
12            LSA nlsa = new LSA(distance,
13                lsa.getName(),
14                lsa.getLsaLocation(),
15                lsa.isDependsOnTraffic(),
16                lsa.getSzpls());
17            nearestLSAs.add(nlsa);
18        }
19    }
20
21 // Ampeln in der Umgebung gefunden, noch keine Ampel festgelegt
22 } else if(nearestLSAs != null && nearestLSA == null) {
23     for(LSA lsa : nearestLSAs){
24         // fuer jede Ampel in der Umgebung Distanz errechnen
25         distance = myLocation.distanceTo(lsa.getLsaLocation());
26
27         // verkleinert sich Distanz im Gegensatz zur Vorherigen?
28         // Ist die Distanz von allen nahem LSAs am geringsten?
29         if (distance < lsa.getDistance()
30             && distance <= Constants.MIN_LSA_DISTANCE
31             && minDistance > distance){
32             minDistance = distance;
33             // naechste Ampel ermittelt!
34             nearestLSA = lsa;
35             if(onSetListener != null){
36                 // LSA gefunden? per Listener MainActivity benachrichtigen
37                 onSetListener.onLSASet(nearestLSA, myLocation);
38             }
39         }
40     }
41 }
42
43 // Entfernung ist hoeher als gegebene Distanz?
44 // und Entfernung ist groesser als vorher
45 if(nearestLSA != null) {
46     if(myLocation.distanceTo(nearestLSA.getLsaLocation()) > Constants.MIN_LSA_DISTANCE
47         || myLocation.distanceTo(nearestLSA.getLsaLocation()) > (distance+0.1)) {
48
49         // Ampeln und temporaere Liste loeschen
50         nearestLSAs = null;
51         nearestLSA = null;
52     }
53 }
```

Listing 8.4 getNearestLSA() GpsTracker.java

### 8.3.3 BERECHNUNG DER GESCHWINDIGKEITSEMPFEHLUNG

An dieser Stelle wird die Berechnung der Geschwindigkeitsempfehlung erklärt. Sobald alle benötigten Daten bestehend aus aktueller, eigener Position, Entfernung zur nächsten Ampel und deren Signalschaltplan wird die benötigte Geschwindigkeit errechnet, die Ampel bei Grün zu erreichen. Diese Geschwindigkeit wird mit der Formel:

$$v = \frac{s}{t_2 - t_1}$$

berechnet, wobei  $t_1$  die aktuelle Sekunde,  $t_2$  der Zeitpunkt zu dem die Ampel auf Rot schaltet und  $s$  die Entfernung zur Ampel ist.

Für FahrradfahrerInnen ist die Beschleunigung ebenfalls von nicht geringer Bedeutung, weil sie begrenzt ist. Die Formel für die Beschleunigung lautet:

$$a = \frac{v}{(t_2 - t_1)^2}$$

Die Variable  $v$  ist hier die errechnete Empfehlungsgeschwindigkeit.

Anhand der errechneten Geschwindigkeit, Beschleunigung und Geschwindigkeit des Geräts wird das Graphical User Interface (GUI) aktualisiert. Hier wird berücksichtigt, ob die errechnete Geschwindigkeit, sowie die Beschleunigung innerhalb festgelegter Parameter liegen. Da bei einer Handlungsaufforderung zusätzlich zur Empfehlung der Countdown der Ampel dargestellt wird, geschieht dies sekündlich.

Jede App hat ihren eigenen UI-Thread und nur Objekte, die auf dem UI-Thread ausgeführt werden haben Zugang zu anderen Objekten auf diesem Thread. Um Daten aus einem Hintergrundthread auf den UI-Thread zu verschieben wird der Handler verwendet. Also wird in der umsetzenden SpeedHandler-Klasse eine Handler- und eine Runnable-Instanz erzeugt. Ein Handler ist eine Schnittstelle für die Kommunikation zwischen Threads. Jede Handler-Instanz wird mit einem Thread und dessen Warteschlange verbunden. Ein Runnable ist ein Objekt, das etwas ausführen kann. Das Runnable wird nun mittels `post()` im Handler abgelegt, wodurch Runnable aus dem Handler im UI-Thread ausgelesen und gestartet wird.

## 8.4 INSTALLATIONSANLEITUNG

Um die Anwendung zu installieren muss deren .apk-Datei auf das Gerät geladen und von dort gestartet werden. Die .apk-Datei ist das als Zwischencode ausführbare Kompilat, welches dann auf dem Gerät zu Plattformcode kompiliert wird. Zum Auffinden und ausführen der Datei wird ein Dateimanager, wie zum Beispiel der kostenlose ASTRO File Manager<sup>5</sup> benötigt.

---

<sup>5</sup> Der ASTRO File Manager steht im Google Play Store unter <https://play.google.com/store/apps/details?id=com.metago.astro&hl=de> bereit.

## 9 EVALUATION

Um die Funktionalität des Prototyps zu erproben, wurden folgende Testgeräte gewählt. Es handelt sich hierbei um Geräte mit unterschiedlichen Bildschirmgrößen und Android-Versionen.

Testgerät	Android-Version	Bildschirm
LG Nexus 5	5.0	1920 x 1080 Pixel
LG Nexus 4	4.4.4	1280 x 768 Pixel
Samsung Galaxy Note 2	4.4.2	1280 x 720 Pixel
Samsung Galaxy Nexus	4.3	1280 x 720 Pixel
Samsung Nexus S	4.1.2	800 x 480 Pixel
Motorola RAZR Maxx	4.0.4	540 x 960 Pixel
HTC Desire HD	2.3.5	480 x 800 Pixel

---

Tabelle 9.1 Verwendete Testgeräte

Die Tests wurden durchgeführt, indem mit dem Fahrrad durch die Stadt Plau am See gefahren wurde. Neben dem Smartphone mit laufender Anwendung wurde ein Tachometer für die Geschwindigkeitskontrolle am Lenker angebracht.

Für das Durchlaufen der folgenden Testreihen wurde vorübergehend eine Displayanzeige implementiert, welche die Ergebnisse der Berechnungen visualisiert.

### 9.1 SYSTEMTEST UND ERGEBNISSE

Allumfassend lässt sich sagen, dass das Design sich auf jeder Displaygröße gut angepasst hat. Die Auflösung sämtlicher Anzeigeelemente haben die Größenverhältnisse sehr genau übernommen und waren gleichermaßen erkenn- und sichtbar.

Bei dem GPS-Sensor gab es jedoch bezüglich des Findens von Satelliten und deren Signalübergabe zwischen ihnen und dem internen GPS-Empfänger Unterschiede. Hier zeigten die neueren Smartphones den Vorteil, dass die Dauer des Kaltstarts, insbesondere bei bewölktem Himmel, deutlich kürzer war.

### 9.1.1 ERMITTlung DER NÄCHSTEN AMPEL

In der ersten Testphase wurde vorwiegend die Richtigkeit der Ermittlung der nächsten Ampel überprüft.

Insgesamt wurden ... von ... LSAs korrekt aufgefunden. Der Algorithmus schlug also bei ... Ampel fehl, was wahrscheinlich durch ... verursacht wurde. Bei den restlichen Ampeln traf dies nicht zu und sie konnten problemlos zugeordnet werden. *Bei der Erkennung der gerade passierten Ampel traten Probleme auf. Hier fehlte bei einigen Geräten das nötige Positionsupdate, doch eine gewisse Ungenauigkeit des GPS-Sensors war zu erwarten.*

Die Erfolgsquote liegt bei ... % und ist damit als erfolgreich zu bezeichnen.

### 9.1.2 BERECHNUNG UND ANZEIGE DER GESCHWINDIGKEITSEMPFEHLUNG

In der zweiten Testphase lag das Hauptaugenmerk auf der ausgesprochenen Handlungsaufforderung und deren korrekter Berechnung. Hierfür wurden sämtliche Ampeln abgefahren und Stichprobenartig davor gehalten, um den Countdown und die Aktualität der Anzeige zu überprüfen. Um genauere Ergebnisse zu erlangen, wurde zusätzlich die Handlungsaufforderung bewusst missachtet.

Die Anzeige, eine Vorhersage ist aufgrund einer verkehrsabhängigen Ampel nicht möglich, wurde sofern die Ermittlung der nächsten Ampel korrekt war, .

Bei den neueren Geräten wurde jedes Mal die richtige Empfehlung ausgesprochen, bei den Älteren gab es kleinere Abweichungen. Dies ist auf die fehlerhafte Geschwindigkeitsausgabe des GPS-Empfängers zurückzuführen, was anhand des Tachometers abzulesen war.

Bei der Countdownanzeige gab es teilweise geringfügige Abweichungen (1-2 Sekunden). Mit sehr hoher Wahrscheinlichkeit lag das an der fehlerhaften Erfassung der Phasendaten. Nach der Beseitigung dieser Fehler durch die Anpassung der Daten war die Anzeige in jedem Fall richtig.

Abschließend ist zu sagen, dass die Anwendung stabil läuft. Probleme sind ... GPS.... Genauigkeit...

Die protokollierten Testergebnisse sind im Anhang zu finden.

# 10 ERGEBNIS UND AUSBLICK

## 10.1 AMPELHINWEISSYSTEM

Alle LSA aufnehmen = sehr zeitaufwändig, da Positionsdaten als Lageplan im Portable Document Format (PDF)-Format vorliegen. Keine genauen Latitude / longitude Werte vorhanden. An Ampeln in OSM kann man sich schlecht orientieren, da dort nur KraftfahrzeugAmpeln aufgeführt sind. Für Fahrräder ist zB die Abbiegeampel auf der gegenüberliegenden Straßenseite.

Kompliziert den Überblick über die LSA-Daten zu gewinnen. Pro Tag unterschiedliche Zeitabschnitte, manchmal mit bis zu zusätzlichen zwei Alternativschaltplänen. Dazu kommen Einschaltpläne plus Phasenübergänge. Weitere Daten wie Feindlichkeitsmatrizen usw.

Liegt am Land – in Großstadt wahrscheinlich andere Genauigkeitsergebnisse wegen der vielen und hohen Gebäude. Optimierung ggf. umsetzen

Personenbezogene Daten aufnehmen? Höchstgeschwindigkeit, maximale Beschleunigung

### 10.1.1 DATENGRUNDLAGE

Wie liegen die Daten vor? Echtzeitdaten benutzen! Schnittstelle zur Verfügung stellen - gerade für die verkehrsabhängigen Ampeln. Dann muss der Teil der Anwendung, welcher die Daten einliest, anders aufgebaut werden.

## 10.2 AUSBLICK

Alternative: Straßendaten, routennavigation

vorteile, ampel ist sicher, nachteile: App wird viel größer (mehrere MB), interaktion einbauen, navigation einbauen etc.

*Mit Straßendaten = andere Formel notwendig. Winkel / Hügel mit einbeziehen + Erdkugel etc.*

Unterschiede Klein-Großstadt? → Datenbasis ist anders

Nachts auf dem Land – Ampeln ausgeschaltet. In Berlin nur 30 Prozent der Ampeln.

## ABKÜRZUNGEN

A-GPS	Assisted Global Positioning System.
API	Application Programming Interface.
App	Applikation.
C2I	Car-to-Infrastructure oder Vehicle-to-Infrastructure.
C2X	Car-to-X oder Vehicle-to-X.
DDMS	Dalvik Debug Monitor Server.
DSRC	Dedicated Short Range Communication.
GLONASS	Global Navigation Satellite System.
GNSS	Global Navigation Satellite System.
GPS	Global Positioning System.
GUI	Graphical User Interface.
HAL	Hardware Abstraction Layer.
JSON	JavaScript Object Notation.
JVM	Java Virtual Machine.
LBS	Location Based Service.
LED	Licht-emittierende Diode.
LSA	Lichtsignalanlage.
NDK	Native Development Kit.
PDF	Portable Document Format.
SDK	Software Development Kit.

UI      User Interface.

VM      Virtual Maschine.

WLAN    Wireless Local Area Network.

XML     Extensible Markup Language.

# GLOSSAR

## **Activity**

Eine Activity ist eine Android-Anwendungskomponente, welche die Bildschirm, beziehungsweise eine Ansicht zur Interaktion bereitstellt.

## **Arduino**

Die Arduino-Plattform ist eine quelloffene, aus Soft- und Hardware bestehende Physical-Computing-Plattform. Die Hardware besteht aus einem einfachen I/O-Board mit einem Mikrocontroller und analogen und digitalen Ein- und Ausgängen. Die Entwicklungsumgebung basiert auf Processing, die auch technisch weniger Versierten den Zugang zur Programmierung und zu Mikrocontrollern erleichtern soll.

## **C2I**

direkter, drahtloser Datenaustausch zwischen Fahrzeugen jeglicher Art und infrastrukturellen Einrichtungen wie Funkbaken und Lichtsignalanlagen auf Basis von WLAN, Bluetooth oder Dedicated Short Range Communication (DSRC).

## **C2X**

direkter Informationsaustausch zwischen Fahrzeugen jeglicher Art, Verkehrsleittechnik wie z.B. Lichtsignalanlagen und Verkehrsleitzentralen.

## **DSRC**

funkgestützte sicherheitsrelevante und private Dienste, die in der Automotive-Technik von mobilen Stationen ausgeführt werden können.

## **Smartphone**

Mobiltelefon, das sich von einem klassischen Mobiltelefon durch einen größeren berührungs-empfindlichen Bildschirm (Touchscreen) und diverse Sensoren, wie dem GPS unterscheidet. So ist eine Interaktion mit der Umgebung und den AnwenderInnen möglich.

# ABBILDUNGSVERZEICHNIS

2.1	Grüne Welle durch LEDs . . . . .	7
2.2	Projekt Travolution . . . . .	9
2.3	Projekt Kolibri . . . . .	10
2.4	Projekt Testfeld-Telematik Ampelinformation . . . . .	10
2.5	Connected Signals live Vorhersage . . . . .	12
2.6	Signal Guru . . . . .	12
2.7	Hammerhead . . . . .	13
2.8	Helios-Lenker . . . . .	14
2.9	Samsung Smart Bike . . . . .	15
2.10	COBI . . . . .	15
4.1	Android-Systemarchitektur . . . . .	20
4.2	Funkzellentriangulierung . . . . .	22
4.3	GPS Satelliten Konstellation . . . . .	23
4.4	Berechnung Progressionsgeschwindigkeit . . . . .	26
5.1	Szenarien . . . . .	27
7.1	Signalplan . . . . .	34
7.2	Use-Case-Diagramm . . . . .	37
7.3	Klassendiagramm . . . . .	38
7.4	Systemzustände im Ampelbereich . . . . .	39

# QUELLCODEVERZEICHNIS

7.1	JSON-Ampelobjekt . . . . .	35
8.1	AndroidManifest.xml . . . . .	41
8.2	Auszug aus der build.gradle-Datei . . . . .	42
8.3	Registrierung für GPS-Updates in GpsTracker.java . . . . .	44
8.4	getNearestLSA() GpsTracker.java . . . . .	45

# LITERATURVERZEICHNIS

- [Amp14] *Audi connect – Intelligent durch den Stadtverkehr mit Ampelinfo online.* [http://www.audi.de/de/brand/de/vorsprung\\_durch\\_technik/content/2014/06/ampelinfo-online.html](http://www.audi.de/de/brand/de/vorsprung_durch_technik/content/2014/06/ampelinfo-online.html), Juni 2014
- [anda] *Activities.* <http://developer.android.com/guide/components/activities.html>, . – Zugriff: 27.01.2015
- [andb] *Android Lollipop.* <http://developer.android.com/about/versions/lollipop.html>, . – Zugriff: 16.01.2015
- [andc] *Configuring Gradle Builds.* <https://developer.android.com/tools/building/configuring-gradle.html>, . – Zugriff: 03.03.2015
- [andd] *Sensors Overview.* [http://developer.android.com/guide/topics/sensors/sensors\\_overview.html](http://developer.android.com/guide/topics/sensors/sensors_overview.html), . – Zugriff: 17.01.2015
- [ande] *Location APIs.* <http://developer.android.com/google/play-services/location.html>, . – Zugriff: 16.02.2015
- [andf] *Android Interfaces.* <https://source.android.com/devices/>, . – Zugriff: 16.02.2015
- [andg] *Location and Maps.* <http://developer.android.com/guide/topics/location/index.html>, . – Zugriff: 17.01.2015
- [andh] *Android NDK.* <https://developer.android.com/tools/sdk/ndk/index.html>, . – Zugriff: 15.01.2015
- [andi] *Android Tools Help.* <https://developer.android.com/tools/help/index.html>, . – Zugriff: 15.01.2015
- [andj] *Dashboards.* <http://developer.android.com/about/dashboards/index.html>, . – Zugriff: 03.03.2015
- [arda] *What is arduino?* <http://www.arduino.cc/>, . – Zugriff: 16.02.2015
- [ardb] *SD Library.* <http://arduino.cc/en/pmwiki.php?n=Reference/SD>, . – Zugriff: 16.02.2015

- [Bat13] BATTENBERG, Dr. A.: Pilotprojekt zur Verbesserung des Verkehrsflusses präsentiert Ergebnisse: Grüne Welle auf der Landstraße spart Zeit und Sprit. (2013), Juni. – Pressemitteilung
- [BBK<sup>+</sup>09] BRAUN, Robert ; BUSCH, Fritz ; KEMPER, Carsten ; HILDEBRANDT, Robert ; WEICHENMEIER, Florian ; MENIG, Cornelius ; PAULUS, Ingrid ; PRESSLEIN-LEHLE, Renate: TRAVOLUTION – Netzweite Optimierung der Lichtsignalsteuerung und LSA-Fahrzeug-Kommunikation. In: *Straßenverkehrstechnik* 10 (2009), S. 365–374
- [ber09] *Grüne Wellen Freie Fahrt für ganz Berlin?* dmp digital- & offsetdruck GmbH, Juli 2009
- [Bes] *Beschleunigung.* <http://elweb.info/dokuwiki/doku.php?id=beschleunigung>, . – Zugriff: 28.02.2015
- [Car14] CARDWELL, Diane: Copenhagen Lighting the Way to Greener, More Efficient Cities. In: *The New York Times* (2014), Dezember
- [cob14a] COBI. *World's Smartest Connected Biking System.* <https://www.kickstarter.com/projects/cobi/cobi-worlds-smallest-connected-biking-system>, 2014. – Zugriff: 28.12.2014
- [cob14b] COBI. <http://www.cobi.bike/press.html>, 2014. – Zugriff: 28.12.2014
- [Con14a] Connected Signals. <http://connectedsignals.com>, 2014. – Zugriff: 26.01.2015
- [Con14b] Connected Signals – Real-Time Signal Status and Predictions. <http://connectedsignals.com/signals.php>, 2014. – Zugriff: 26.01.2015
- [Con14c] Connected Signals' Technology. [http://connectedsignals.com/enlighten\\_tech.php](http://connectedsignals.com/enlighten_tech.php), 2014. – Zugriff: 26.01.2015
- [cyc14] CycleNav Smart Bike Navigator. <http://www.schwinnbikes.com/usa/news/cyclenav-smart-bike-navigator/>, März 2014. – Zugriff: 28.12.2014
- [Dam14] DAMBECK, Holger: Wir fahren schon mal vor. In: *Technology Review - Das Magazin für Innovation* (2014), August, S. 70ff.
- [Ele14] ELENKOV, Nikolay: *Android Apps for absolute Beginners*. San Francisco : No Starch Press, Inc., 2014. – ISBN 978–1–59327–581–5
- [Ele15] ELENKOV, Nikolay: *Android Security Internals, An In-Depth Guide to Android's Security Architecture*. San Francisco : No Starch Press, Inc., 2015. – ISBN 978–1–59327–581–5
- [Elf13] ELFLEIN, Nicole: Der perfekte Beifahrer. In: *Pictures of the Future* (2013), Frühjahr, S. 104–106
- [FA11] FERRARO, Richard ; AKTIHANOGLU, Murat: *Location-Aware Applications*. Shelter Island : Manning Publications Co., 2011. – ISBN 978–1–935182–33–7

- [figa] *The Android Source Code.* <https://source.android.com/source/index.html>, . – Zugriff: 17.02.2015
- [figb] *Pinpointing Cell Phone Users using Emergency9.* <http://www.emergency9.co.za/general-info/emergency9-mobile-panic-cell-phone-triangulation>, . – Zugriff: 26.01.2015
- [figc] *Constellation Arrangement.* <http://www.gps.gov/multimedia/images/constellation.jpg>, . – Zugriff: 26.01.2015
- [ham] *Hammerhead.* <http://hammerhead.io/>, . – Zugriff: 28.12.2014
- [hel14] *Helios.* <http://www.ridehelios.com/>, 2014. – Zugriff: 28.12.2014
- [J.A14] J.ANKER: Fahrradfahren boomt in Berlin stärker als bislang bekannt. In: *Berliner Morgenpost* (2014), Juni
- [Jan14] JANDRISITS, Dipl.-Ing.(FH) M.: Testfeld Telematik – Publizierbarer Endbericht. 2014. – Forschungsbericht
- [kol] *Kooperative Lichtsignaloptimierung.* <http://www.kolibri-projekt.de/>, . – Zugriff: 28.12.2014
- [KPM11] KOUKOUMIDIS, Emmanouil ; PEH, Li-Shiuan ; MARTONOSI, Margaret: SignalGuru: Leveraging Mobile Phones for Collaborative Traffic Signal Schedule Advisory / MIT and Princeton University. 2011. – Forschungsbericht
- [MS12] MILETTE, Greg ; STROUD, Adam: *Professional Android™ Sensor Programming*. Indianapolis, Indiana : John Wiley & Sons, Inc, 2012. – ISBN 978–1–118–18348–9
- [Neu14] NEUMANN, Peter: Berlin bekommt zweite grüne Welle für Radler. In: *Berliner Zeitung* (2014), September
- [Sch14] SCHOENEBECK, Gudrun von: *Kopenhagen verwöhnt Radfahrer mit Grünen Wellen.* <http://www.ingenieur.de/Themen/Fahrrad/Kopenhagen-verwoehnt-Radfahrer-Gruenen-Wellen>, August 2014. – Zugriff: 28.12.2014
- [sen] *Sensor.* <http://www.itwissen.info/definition/lexikon/Sensor-sensor.html>, . – Zugriff: 16.02.2015
- [sma14] *Samsung Smart Bike.* <http://www.maestrosacademy.it/progetto-sbike>, 2014. – Zugriff: 28.12.2014
- [Som10] SOMMER, Ulli: *Arduino – Mikrocontroller-Programmierung mit Arduino/Freeduino*. 85586 Poing : Franzis Verlag GmbH, 2010. – ISBN 978–3–446–43823–1

- [StV] *Straßenverkehrs-Ordnung (StVO)*. – Ausfertigungsdatum: 06.03.2013, Neufassung in Kraft getreten am 01.04.2013
- [Tho09] THOMA, Stephan: *Mensch-Maschine-Interaktion*, Technische Universität München, Diss., Oktober 2009
- [Toy11] TOYOTA: *TMC Develops Navigation System Compatible with Vehicle-Infrastructure Co-operative Safety System*. <http://www2.toyota.co.jp/en/news/11/06/0629.html>, 2011. – Zugriff: 08.12.2014
- [tra] *Verkehrsoptimierung mit Genetischen Algorithmen – Projektpartner*. <http://www.travolution-ingolstadt.de/index.php?id=73>, . – Zugriff: 30.12.2014
- [van] Vanhawks. <https://www.vanhawks.com/>, . – Zugriff: 28.12.2014
- [Zim84] ZIMDAHL, Walter: Guidelines and some developments for a new modular driver information system / Volkswagenwerk AG. 1984. – Forschungsbericht. – 178 – 182 S.

# **ANHANG**

## **EIDESSTATTLICHE ERKLÄRUNG**

## **TESTPROTOKOLL**