SA - Tour de Maison

Jan Forlin, Zarko Dragojevic

21. Februar 2019

Zusammenfassung

Ausgangslage: Fitness-Geräte wie Laufbänder, Stepper oder Rollentrainer (Smart Trainer) werden immer intelligenter. Vor allem in der kalten Jahreszeit spannen Hobby-Sportler ihr Rennvelo in ihren Smart Trainer ein, welcher vom Smartphone oder Tablet aus über Bluetooth gesteuert wird. Die mitgelieferte Smart Trainer Software erlaubt das Nachfahren von Strecken (GPS-Streckendaten), je nachdem auch Videos verfügbar sind. Im Rahmen dieser Arbeit sollte eine Android-Applikation realisiert werden, welche es ermöglicht, zu Hause auf einem Smart Trainer den Rennfahrern der Tour de Suisse virtuell nachzufahren. Dank Integration von Google Street View sollten zu den meisten Strecken auch Bilder zur aktuellen Position geliefert werden.

Ergebnis: Mit 'Tour de Maison' entstand eine Android App, welche über Bluetooth Low Energy mit einem Drivo Elite Smart Trainer kommuniziert. Die App steuert den Smart Trainer basierend auf Streckendaten von Etappen der Tour de Suisse. Die für diese Anwendung aufgearbeiteten Streckendaten stehen auf einem cnlab Server zur Verfügung. Sobald der Hobby-Sportler auf dem Smart Trainer in die Pedale tritt, bestimmt die App anhand von aktueller Steigung, Fahrgeschwindigkeit und Roll-/Luftwiderstand die auf dem Smart Trainer aufzubringende Leistung. Auf dem eingeblendeten Höhenprofil der Strecke, zeigt die App die Positionen des Rennfahrers und des Hobby-Sportlers an. In einem weiteren Fenster sind Google Street View Bilder der aktuellen Position zu sehen. Auf diese Weise kann man eine Tour de Suisse Etappe zuhause nachfahren und erlebt so, welche Leistung die Rennfahrer erbringen. Damit man den Rennfahrern folgen kann, steht ein «Handicap-Faktor» bzw. Gewichtsregler zur Verfügung. Damit kann die aufzubringende Leistung reduziert werden.

Fazit: Die 'Tour de Maison' App ist ein Prototyp für Hobby-Sportler. Es liegen nun vertiefte Kenntnisse zum neuen Fitness Machine Service (FTMS) Protokoll des Bluetooth GATT-Profils vor. Damit sollte auch die Steuerung von anderen Fitness-Geräten und Smart Trainer Produkten, die FTMS unterstützen, möglich sein. Basierend auf der vorliegenden App soll in einem nächsten Schritt eine Webanwendung realisiert werden, über welche mehrere Hobby-Sportler an virtuellen Rennen teilnehmen können. Neben den Hobby-Sportlern sollen und Zuschauer das Renngeschehen online verfolgen können.

1 Aufgabenstellung

1.1 Virtuelle Radrennen

Studiengang:	Informatik
Semester:	HS 2018
Durchführung:	Studienarbeit
Fachrichtung:	Internet-Technologien und -Anwendungen
Institut:	INS
Gruppengrösse:	Zarko Dragojevic und Jan Forlin
Status:	Version 0.3
Betreuer:	Prof. Dr. P. Heinzmann (HSR/cnlab), Patrick Eichler (cnlab)
Industriepartner:	-

1.2 Ausgangslage

Computing und Data Analytics im Sport hat sich zu einer attraktiven Domäne der Informatik entwickelt. Bekannt sind die Auswertungen umfangreicher Datensammlungen zu Fussball-, American Football-, Basketball- oder Eishockey-Spielen. Ebenfalls bekannt sind die Bilder von Radrennfahrern, welche nur noch auf ihre Fahrradcomputer schauen. Weniger bekannt sind Veranstaltungen, bei denen Sportler auf über das Internet vernetzten interaktiven Smart-Trainern mit Elektrobremsen (Rollentrainer, Ergometer) sitzen und virtuell gegeneinander kämpfen [1, 2, 3, 4]. Verschiedene Anbieter stellen Software für virtuelle Rennen und Trainings zu Verfügung. Es gibt auch Anwendungen, bei denen die virtuelle Fahrt durch die Google StreetView Welt führen [5]. An der Tour de Suisse 2017 und 2018 wurde in Zusammenarbeit mit SRF Sport gezeigt, wie ein Testfahrer auf einem Smart-Trainer im Vergleich zu den Radrennprofis im realen Rennen abschneidet. Dazu wurde die Belastung für den Testfahrer auf einem Smart-Trainer entsprechend der Situation der Radrennprofis im Gelände nachgebildet [6]. Diese Anwendung soll nun für die Tour de Suisse 2019 so erweitert werden, dass mehrere Fahrer am Vergleich mitmachen können und dass den Testfahrern Videos und/oder Google Street View Bilder angezeigt werden können. Es ist ein Vergleich der Leistungen von echten Rennfahrern im Gelände mit derjenigen von über das Internet verteilten Rennfahrern auf Smart-Trainern geplant. Die Anwendung für virtuelle Radrennen in der Google StreetView Welt könnte auch im Rahmen von Ausstellungen und Demos der HSR genutzt werden.

1.3 Ziel

Nach dieser Arbeit soll eine robuste Radsimulator-Anwendung mit erweiterten Funktionen vorliegen. Dazu sind verschiedene Möglichkeiten zur Durchführung von virtuellen Radrennen zu studieren. Es soll ein System vorliegen, mit welchem mehrere Testfahrer Etappen der Tour de Suisse auf Smart-Trainern in der «Google Street View Umgebung» nachfahren können. Der Rennverlauf soll über eine Webanwendung visualisiert werden können.

1.4 Aufgaben

- Einarbeitung
  - Grundkenntnisse zur Leistungsmessung bei Radfahrern [7, 8]
  - Studium zu GPS-Tracks (Format, Nutzung, Leistungsberechnung)
  - Studium der Schnittstellen zu verschiedenen Simulatoren

    - Bluetooth (H)
    - ANT+ ist eine nicht-standardisierte, flexible Low-Power-Funktechnik zur Verbindung von verschiedenen Fitness-Sensoren und -Geräten über Entfernungen bis zu 10m. ANT+ unterscheidet Profile für verschiedene Anwendungen wie beispielsweise Heart Rate Service (HRS), Cycling Power Service (CPS), Cycling Speed & Cadence Service (CSCS), Running Speed & Cadence Service (RSCS) oder Fitness Machine Service (FTMS).

- Analyse
  - Studium und Testing von verschiedenen Simulator Anwendung (kommerzielle und Google SteetView Anwendungen)
  - Studium und Testing von verschiedenen virtuellen Rennformaten
  - Realisierung einer Testing Umgebung für Elite Simulatoren und Fahrräder mit SRMLeistungsmessern (Vergleich der Simulationen mit den echten Belastungen)
- Design
  - Festlegung des virtuellen Rennkonzepts (Registrierung, Durchführung, Auswertung)
  - Realisierung von Steuerungsmöglichkeiten von verschiedenen Simulatoren basierend auf GPS und Google Street View Daten zu den zu fahrenden Strecken
- Implementierung
  - Android-Anwendung, welche Fahrten in Google Street View Umgebung basierend auf GPS-Inputs ermöglicht (Steuerung mit Android Tablet, Video/Google StreetView Anzeige auf Bildschirmen und/oder VR-Brille)
  - Möglichkeit der Teilnahme von mehreren Fahrern
  - Anzeige der Position der Fahrer auf einer Webseite mit Karte und Höhenprofil (Zusammen mit Leistungs-, Trittfrequenz- und Pulsangaben)
- Testing
  - Durchführung von virtuellen Rennen mit Testdaten von der Tour de Suisse 2018

2 Management Summary

2.1 Ausgangslage

Immer mehr Geräte werden intelligenter. Die Bedienung der Geräte wird immer mehr über Smartphones abgewickelt. Zugleich senden die Geräte ihre Werte an das Smartphone. Auch im Hometrainer-Bereich hält dieser Trend Einzug. Es gibt bereits Smart Trainer, auf welchen mit Hilfe der mitgelieferten Software, reale Strecken simuliert werden. Die Firma Cnlab hat bereits mit Elite einen Smart Trainer getestet. Leider war die Kommunikation zwischen Smartphone und Smart Trainer nicht zuverlässig genug. Ziel dieser Studienarbeit ist, die Kommunikation zwischen Trainer und Smartphone besser zu verstehen und eine eigene, robuste App zu entwickeln. Dabei soll es möglich sein, vordefinierte Strecken der Tour de Suisse abzufahren.

2.2 Vorgehen

Zwei Studenten sollen eine Android-App entwickeln. Es soll eine Kommunikation zwischen Smart Trainer und Android-Gerät implementiert werden, damit eine reale Strecke auf dem Trainer simuliert werden kann. Die Streckendaten werden von Cnlab geliefert. Der Hobby-Sportler zu Hause sollte ein Bild seiner aktuellen Streckenposition in der App erhalten. Welche Lösungen für die Kommunikation (z.B. Bluetooth oder ANT+) und die Darstellung des Bildes (z.B. Google Street View, Video usw.) umgesetzt werden, sollte im Rahmen der Arbeit ermittelt werden. Es fanden wöchentlich Sitzungen mit den Betreuern statt. Anfangs wurden viele Dokumentation zu den Kommunikationsprotokollen gesammelt und studiert. Mit der Entwicklung wurde bereits früh begonnen. Es entstanden erst einzelne Applikationen, welche die verschiedenen Funktionen testeten. Diese wurden gegen Ende des Projekts zusammengeführt. Während der Dauer der Studienarbeit stand ein Smart Trainer der Firma Elite (Drivo) mit einem Rennvelo zur Verfügung. Elite hat wichtige Dokumente zum Kommunikationsprotokoll 'Bluetooth Low Energy' (auch Bluetooth Smart) beigesteuert.

2.3 Ergebnis

Die App 'Tour de Maison' ist ein gelungener Prototyp, welcher es erlaubt, virtuell eine Strecke auf dem Smartphone und Smart Trainer nachzufahren. Dabei wird ein Bild von Google Street View in der App angezeigt. Sowohl die Daten als auch die Anzeige der Position wird laufend aktualisiert. Ein weiteres Feature ist der Rennmodus. Dieser erlaubt es, gegen die Spitze der echten Tour de Suisse Fahrer zu fahren. Die Streckendaten der Tour de Suisse-Etappen mit den Zeiten der Spitzenfahrer wurden von Cnlab ermittelt und zur Verfügung gestellt.

2.4 Ausblick

Die App ist noch nicht voll ausgebaut. Als Ausbaustufe könnte beispielsweise ein Online Multiplayer-Mode implementiert werden. Dieser soll es ermöglichen, gegen andere Hobby-Rennradfahrer anzutreten. Ein Belohnungssystem würde den Hobby-Rennradfahrer zusätzlich anspornen. Für eine noch immersivere Nutzung könnte die Google Street View Anzeige auf eine 'Virtual Reality'-Anwendung portiert werden.

Inhaltsverzeichnis

3 Einleitung

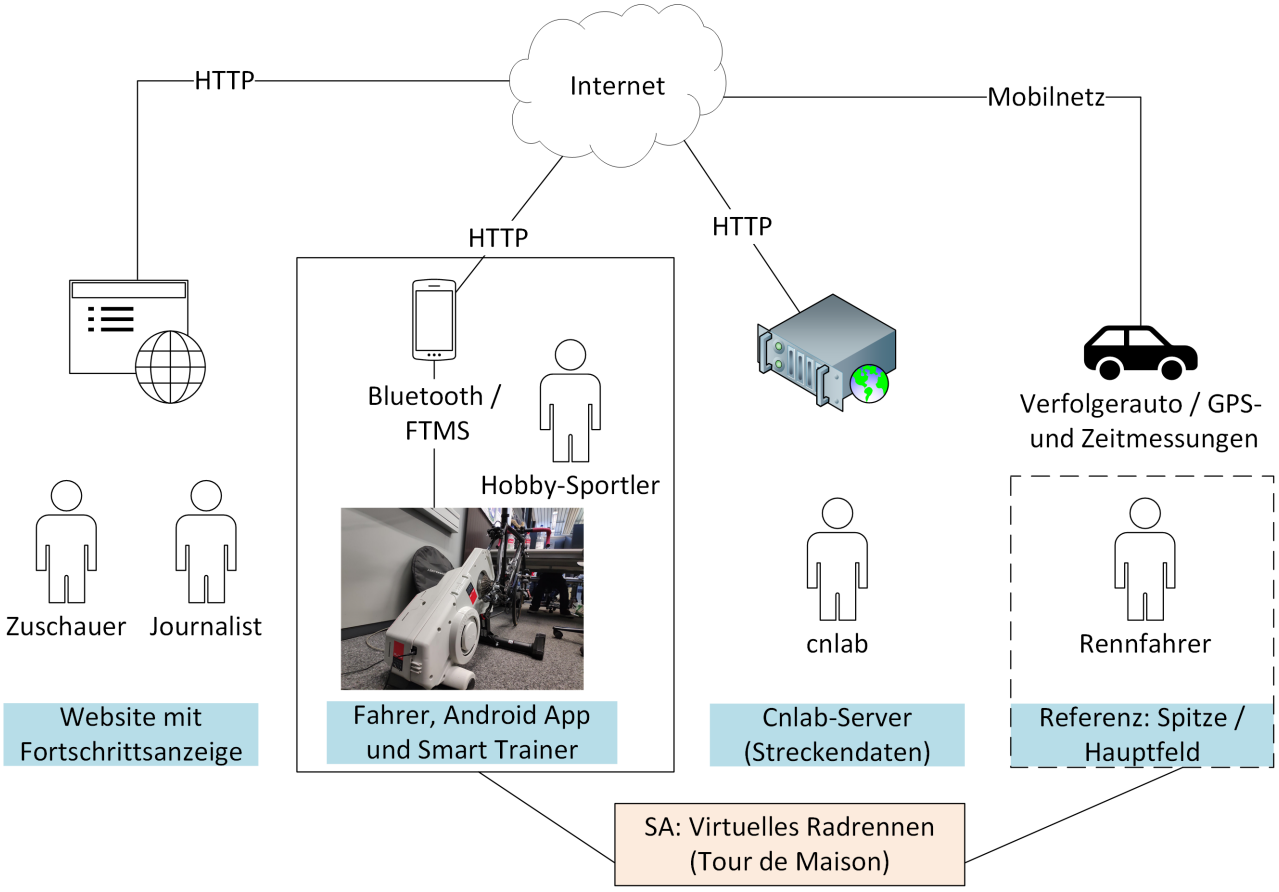

Abbildung 1: BigPicture 'Tour de Maison'

Die Abbildung 1 zeigt eine Übersicht dieser Arbeit. Der Fokus soll dabei auf den Fahrer mit Android App gelegt werden. Die 'Tour de Maison' App verbindet sich mit dem Smart Trainer über Bluetooth Low Energy und ermöglicht es virtuelle Radrennen zu fahren. Es kann ein Streckenausschnitt ausgewählt und abgefahren werden. Obwohl die meisten Smart Trainer mit einer vom Hersteller gelieferten Software kommen, welche das Abfahren von Strecken ermöglichen, wurde auf eine Eigenentwicklung gesetzt. Die Studienarbeit beschreibt, wie die Kommunikation zwischen Smart Trainer und Smartphone über Bluetooth Low Energy funktioniert.

Für den Benutzer der App sind vor allem die Kapitel Übersicht Smart Trainer und 'Tour de Maison' für Anwender interessant. Die Entwickler finden wichtige Informationen in den Kapiteln Übersicht Kommunikation und 'Tour de Maison' für Entwickler.

4 Übersicht Smart Trainer

Ein Smart Trainer bietet im Gegensatz zu normalen Hometrainern die Möglichkeit, den Widerstand dynamisch zu verändern. Dabei erfolgt die Steuerung mit Hilfe einer Smartphone App. Meistens wird diese vom Smart Trainer-Hersteller angeboten. Ein Smart Trainer verfügt über verschiedene Kommunikationsprotokolle, oft Bluetooth oder ANT+. Mit Hilfe von APIs können diese Trainer angesteuert werden. Je nach Hersteller unterstützen die Apps verschiedene Trainingsmodi, beispielsweise das Simulieren einer Strecke.

Bei Smart Trainern sind zwei verschiedene Systeme zu unterscheiden: Wheel-On und Direktantrieb (siehe Abbildung 2). Bei der Wheel-On Variante (rechts) kann ein Rennrad ohne entfernen des Hinterrades auf den Trainer eingespannt werden. Es gibt auch Varianten, bei welchen kein Einspannen möglich ist. Die Übertragung erfolgt zwischen Reifen und Rolle des Trainers. Beim Direktantrieb hingegen, muss das Hinterrad ausgebaut werden und der Rahmen direkt mit dem Smart Trainer verbunden. Die Kraft wird mit Hilfe der Kette zwischen dem Kranz mit Pedaltrieb und der Kassette auf der Nabe des Smart Trainers übertragen.

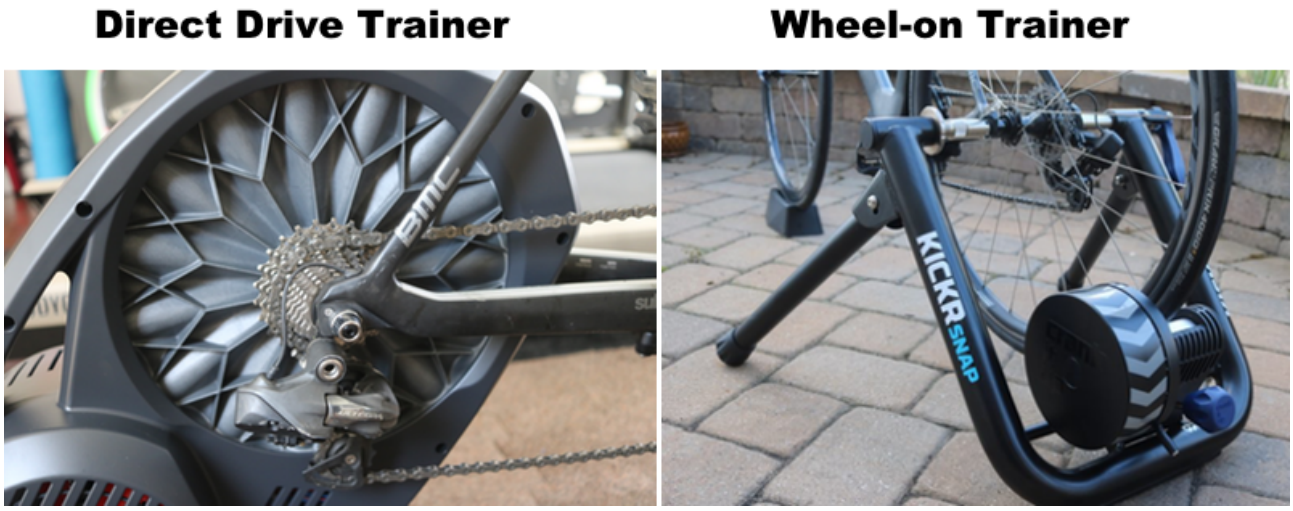

Abbildung 2: Unterschied Direktantrieb und Wheel-on [9]

In der Tabelle 1 sind die Vor- und Nachteile der 2 Systeme aufgeführt. Direktantrieb Systeme sind genauer und benötigen weniger Kalibration. Ausserdem sind sie ruhiger, da der Antrieb nicht über einen Reifen erfolgt. Dadurch tritt weder Schlupf noch Verschleiss des Reifens auf. Das Wheel-On System hat seine Stärken beim günstigeren Preis und dem niedrigeren Gewicht.

	Direktantrieb	Wheel-on
Genauigkeit	+	-
Kalibration	automatisch	manuell
Lautstärke	eher leise	eher laut
Abnutzung Reifen	nicht vorhanden	vorhanden
Kraftübertragung	+	-
Mobilität, Gewicht	-	+
Installationaufwand	-	+
Preis	eher teuer	meist günstiger

Tabelle 1: Vor- und Nachteile Wheel-On und Direktantrieb [9]

4.1 Geräte

Die Tabelle 2 zeigt einen Überblick einiger Smart Trainer. Es werden drei Wheel-On und ein Direktantrieb Trainer verglichen. Die wesentlichen Unterschiede sind im Preis, der Genauigkeit und dem maximalen Widerstand zu finden. Auch in der Kompatibilität zu Software sind diese sich, bis auf eine Ausnahme, ähnlich.

Hersteller	**Tacx**	**Wahoo**	**Elite**	**Elite**
Bezeichnung	Genius Smart T2080	Fitness Kickr Snap	Qubo Digital Smart B+	Drivo
Preis	712.90 CHF	660.00 CHF	413.30 CHF	999.00 CHF
System	Ergotrainer	Ergotrainer	Ergotrainer	Ergotrainer
Typ	Wheel-On	Wheel-On	Wheel-On	Direktantrieb
Bremssystem	Motor (virtuelles Schwungrad)	Elektro-magnetischer Widerstand	Magnet-Widerstandsystem	Elektro-magnetischer Widerstand
Bremswiderstände	dynamisch geregelt (Schwungrad-effekt bis zu 125kg)	k.A.	16 Stufen	Stufenlos
Maximaler Widerstand	1500 Watt	1500 Watt	1600 Watt	2000 Watt
Genauigkeit	n.v.	+/- 3%	n.v.	+/- 1%
Herstellersoftware	Tacx App	wahoo ELEMNT	Elite E-Training App	Elite E-Training App
Bluetooth	✓	✓	✓	✓
ANT+	✓	✓	✓	✓
Zwift [10]	✓	✓	✓	✓
TrainerRoad [11]	✓	✓	✓	✓
Kinomap [12]	✓	✓	✓	✓
Bkool [13]	✓	✓	✓	✓
Rouvy [14]	✓	✓	✓	✓
FulGaz [15]	✗	✓	✓	✓
Strva [16]	✓	✓	(✓)*	(✓)*
thesufferfest [17]	✓	✓	✓	✓

Tabelle 2: Vergleich Smart Trainer [3]

* Import über .tcx Datei

4.2 Vergleich: Bestehende Software

Am Ende der Tabelle 2 sind mehrere Applikationen aufgeführt. Als Teil dieser Arbeit wurden nicht alle ausprobiert und getestet, sondern lediglich 'Zwift' und 'My E-Training'. 'Zwift', weil es auch bei Profi-Sportlern sehr beliebt ist und 'My E-Training', weil sowohl die App als auch das Testgerät - der Drivo - von Elite stammen.

4.2.1 My E-Training

Die App 'My E-Training' stammt von Elite und es gibt sie für Android- und iOS-Devices. Sie funktioniert am zuverlässigsten mit ihren eigenen Smart Trainern. Für die Verwendung muss eine Lizenz

erworben werden, wobei jährliche Kosten anfallen. 'My E-Training' liefert viele Statistiken des Trainings. Neben den typischen Werten wie Geschwindigkeit, Leistung, Kadenz usw. zeigt sie auch noch persönliche Fortschritte auf. Es können unterschiedliche Strecken nachgefahren werden. Für eine Visualisierung der Strecke gibt es Videos. Die Videos können selbst aufgenommen oder von Elite bezogen wurden. Jede Videostrecke muss einzeln erworben werden. Die Preise erstrecken sich von knapp 20 Euro bis über 350 Euro pro Strecke. Abbildung 3 zeigt 2 Screenshots der App.

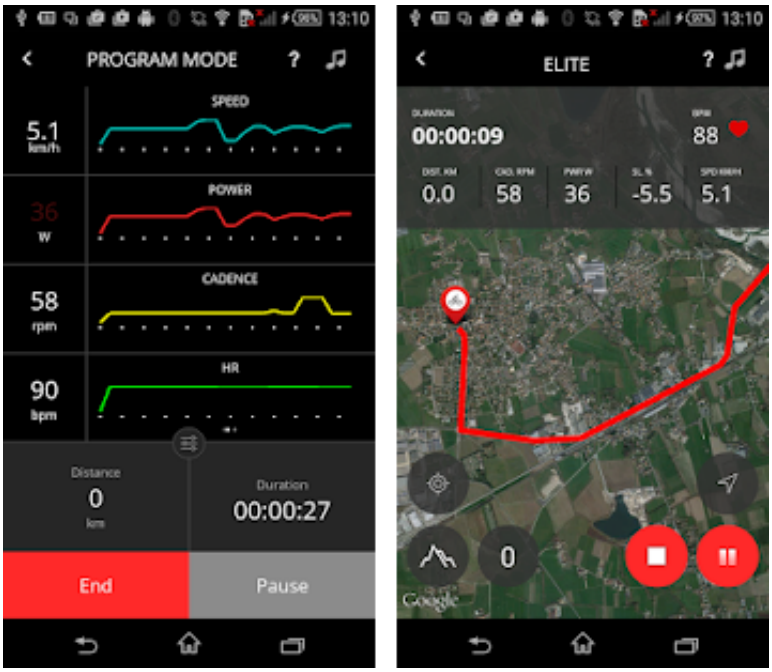

Abbildung 3: Screenshot der 'My E-Training'-App

4.2.2 Zwift

'Zwift' ist ein Spiel, in welchen virtuelle Rennen gegen Onlinespieler angetreten werden können. Die Darstellung erfolgt in 3D (siehe Abbildung 4). Die Strecken sind dabei fiktional/synthetisch. Mit einem Belohnungssystem wird der Spieler angeheitert, immer am Ball zu bleiben. Eine weitere Applikation ist Zwift Run, welche mit smarten Laufbändern funktioniert und virtuelle Laufrennen ermöglicht. Zwift gibt es für iOS-Geräte. Für Android ist bislang lediglich eine Beta-App vorhanden. Sie ist nicht kostenfrei, aber im Gegensatz zu 'My E-Training' gibt es ein Testabo.

Abbildung 4: Screenshot der 'Zwift'-App

4.2.3 Tour de Maison

Die im Rahmen dieser Studienarbeit entwickelte Android-App 'Tour de Maison' ermöglicht es, Tour de Suisse-Etappen zu Hause auf dem Smart Trainer nachzufahren. Sie simuliert dabei die benötigte Leistung aufgrund des Höhenprofils, Geschwindigkeit, Luftwiderstand usw. Zur Visualisierung der aktuellen Fahrerposition kann Google Street View eingeblendet werden. Weil die Messdaten eines realen Rennfahrers in den Streckendaten enthalten sind, kann sich der Hobby-Fahrer mit ihm vergleichen. Die letzten beiden Funktionen unterscheiden 'Tour de Maison' im Wesentlichen von 'Zwift' und 'My E-Training'. In Abbildung 5 befindet sich ein Beispiel-Screenshot von 'Tour de Maison'.

Abbildung 5: Screenshot der 'Tour de Maison'-App

4.3 Vergleich: Karteneinbindung

Weil zu Beginn dieser Arbeit noch nicht klar war, wie die aktuelle Position des Hobby-Fahrers in der Applikation dargestellt werden soll, wurden verschiedene Möglichkeiten in Betracht gezogen. Zur Diskussion standen dabei Lösungsansätze in 2D (Google Maps, Geoportal), 3D (Google Street View, Video der Strecke) oder virtuelle/synthetische Umgebungen (Geopartal-3D, Zwift). Abbildung 6 zeigt eine Beispielaufnahme der virtuellen Welt von Geoportal-3D auf.

Abbildung 6: Geoportal-3D-Screenshot einer 'virtualcity'-MAP in Sankt Gallen

Die einzelnen Lösungsansätze werden nicht näher beschrieben. Für die Entscheidungsfindung wurde im folgenden Kapitel eine Vergleichstabelle (Tabelle 3) erstellt, welche die wichtigsten Entscheidungskriterien beinhaltet.

4.3.1 Vergleichstabelle Karteneinbindung

	2D		3D		Synthetisch	
	Google Maps	Geoportal	Google Street View	Video	Geoportal-3D	Zwift
Kostenlose Lizenz	✓* [18]	✓[19]	✓*	✓	✓[20]	✗
Ressourcen-arm	✓	✓	(✓)	✓	✗	✗
Abdeckung Radwege	✓	✓	(✓)	✗	✗	✗
Web-API	✓	✓	✓	✗	✗	✗
Mobile-API	✓	✗	✓	✗	✗	✓
Gut doku-mentiert	✓	(✓)	✓	✗	✗	(✓)

Tabelle 3: Vergleich möglicher Lösungen zur Darstellung der aktuellen Hobby-Fahrer-Position in der App

*Die Lizenz selbst ist nicht kostenlos. Allerdings stellt Google die ersten 200 Dollar pro Monat, die gemäss ihrem Pricing-Modell an Kosten anfallen, frei zur Verfügung [21]. In dieser Arbeit werden die Grenzwerte bis es effektiv zu Kosten kommt nicht überschritten.

4.3.2 Entscheid Karteneinbindung

Aufgrund der Tabelle 3 konnte eine Entscheidung gefunden werden. Es wird die Google Street View Lösung implementiert. Ausschlaggeben ist die gute Dokumentation von Google APIs und den zahlreichen Tutorials, welche im Netz verfügbar sind. Die Google Maps Lösung hat zwar dieselben Vorteile wie Street View, allerdings ist sie weniger 'spektakulär'. Für den Endbenutzer ist die App mit Google Street View interessanter.

Bei einem zukünftigen Ausbau der App wäre es nützlich, wenn beide Lösungen integriert werden. So könnte beispielsweise eine Google Maps Karte eingeblendet werden, wenn keine Street View Bilder der aktuellen Position vorhanden sind.

5 Übersicht Kommunikation

Für die Kommunikation mit Pulsmessern, Trittfrequenz-, Geschwindigkeits-, Leistungssensoren und Trainergeräten (z.B. Smarte Radtrainer, Laufbänder) haben sich der Bluetooth Low Energy (BLE) Standard und die proprietäre ANT+ Lösung durchgesetzt. Weil das vorhandene Testgerät - der Elite Drivo - Bluetooth Low Energy (BLE) und ANT unterstützt, werden nachfolgend diese zwei Schnittstellen mit ihren Eigenschaften im Detail beschrieben.

5.1 Bluetooth Low Energy (BLE)

2011 wurde mit Bluetooth 4.0 auch Bluetooth Low Energy eingeführt. Die Verbindung wird beim klassischen Bluetooth und BLE gleich auf dem 2.4-GHz-ISM-Band aufgebaut. Dies hat den Vorteil, dass sie sich eine Antenne teilen können. Klassisches Bluetooth muss aber nicht zwangsweise BLE-Unterstützung haben, obgleich es häufig der Fall ist [22].

Wie es der Name bereits vermuten lässt, verfolgt Bluetooth Low Energy das Ziel, möglichst effizient zu arbeiten. Dies wird über verschiedene Wege realisiert. Bevor es in die technischen Details geht, werden einige allgemeine Informationen zu BLE beschrieben.

5.1.1 Allgemeines zu BLE

Nachfolgend werden einige Eigenschaften von Bluetooth Low Energy beschrieben, welche dem Verständnis dienen sollten, auch wenn sie nicht direkt mit der 'Tour de Maison'-App zusammenhängen. Die meisten Informationen stammen von einem Webblog [23][24].

Spezifikationen/Layer von BLE
Ein BLE-Gerät wird in drei Teile unterteilt: **Application**, **Host** und **Controller**. Dabei ist die Application der höchste Layer und für die gesamte Logik zuständig. Der Host beinhaltet mehrere Layer:

- Generic Access Profile (GAP)
- **Generic Attribute Profile (GATT)**
- Logic Link Control and Adaptation Protocol (L2CAP)
- Attribute Protocol (ATT)
- Security Manager (SM)
- Host Control Interface (HCI), Host-Seite

Der Controller beinhaltet folgende Layer (wird von Bluettoht Chip abgehandelt):

- Host Control Interface (HCI), Controller-Seite
- Link Layer (LL)
- Physical Layer (PHY)

In Abbildung 7 wird veranschaulicht, wie die verschiedenen Layer aufgebaut sind.

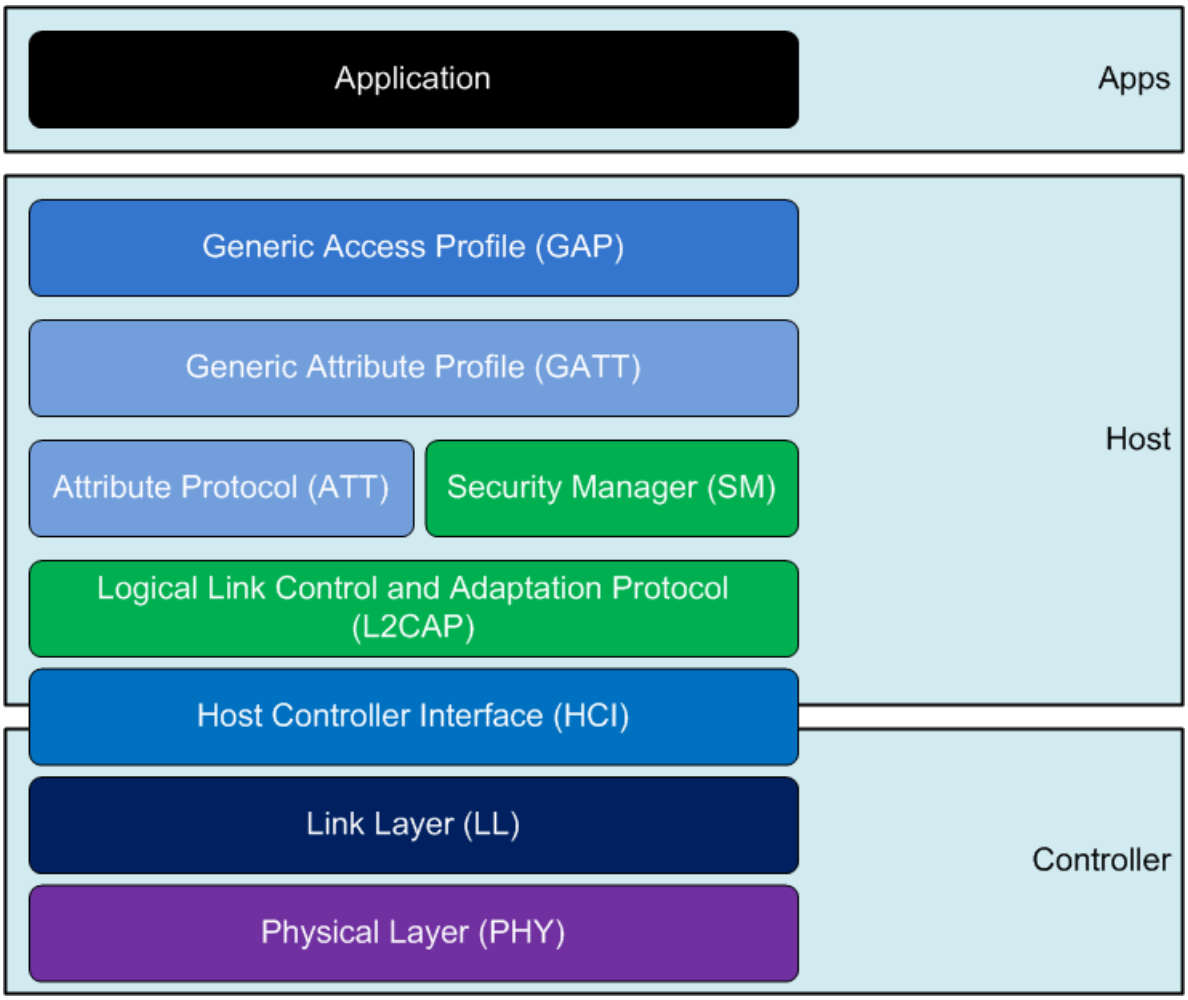

Abbildung 7: Bluetooth Low Energy Layers [25]

Es werden nicht alle Layer näher beschrieben, sondern lediglich jene, welche für die Applikation relevant sind.

Physical Layer
Bluetooth Low Energy arbeitet auf dem 2.4-GHz-ISM-Band. Es unterteilt das Band in 40 Kanäle von 2.4000 GHz bis 2.4835 GHz. Die Kanäle 0, 12 und 39 sind als Signalisierungskanäle reserviert. Die Kanalbreite beträgt 2 MHz. Die Übertragungsrate liegt bei 1 Mbit/s. Die Sendeleistung beträgt 10mW. Unter idealen Bedingungen könnten bis zu 40 Meter überbrückt werden, die typische Einsatzdistanz ist aber nur etwa 10 Meter [26].

Link Layer
Der Link Layer verarbeitet den Transfer von L2CAP-Paketen und stellt für die Integrität der Daten die garantierte Ablieferung der Pakete sicher. Der Aufbau eines BLE-Paket ist in Abbildung 8 dargestellt.

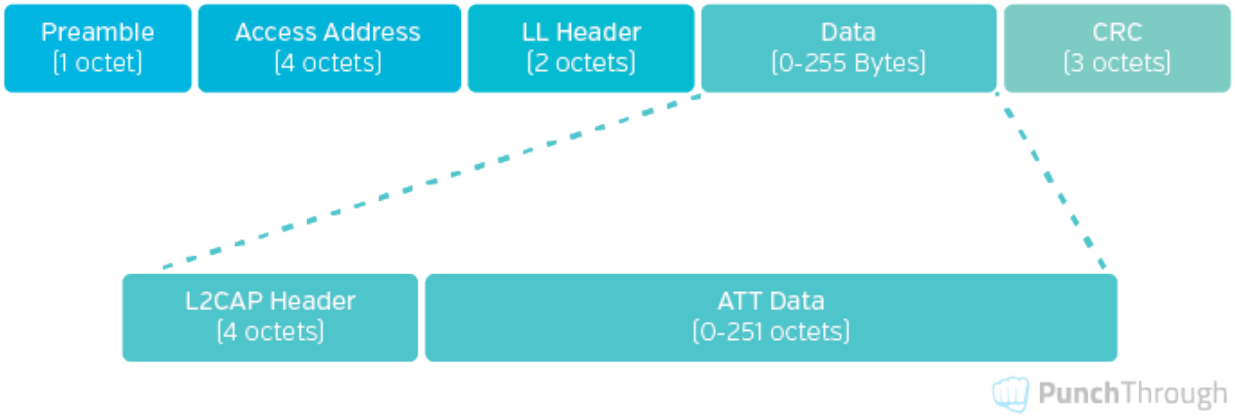

Abbildung 8: BLE Data-Feld: L2CAP Fragmentierung von Paketen [27]

Von grosser Bedeutung ist das Data-Feld. Das Data-Feld ist abhängig von der Bluetooth-Version. Bis und mit Version 4.1 war es maximal 27 Bytes gross. Davon werden 4 Byte für den L2CAP-Header benötigt. Die restlichen 23 Byte sind Daten. Ab Version 4.2 kann es 255 Bytes (ohne Fragmentierung) gross sein. Die Maximum Transmission Unit (MTU) der ATT-Daten kann aber auch beliebig gross gewählt werden. L2CAP kümmert sich dann um die Fragmentierung [27].

L2CAP

Einer der wohl wichtigsten Layer im Bluetooth Stack ist das 'Logical Link Control and Adaptation'-Protokoll (L2CAP). Es sorgt dafür, dass mehrere Layer von den höheren Schichten miteinander arbeiten können. Man kann es auch als Protokoll Multiplexer sehen, welches die Protokolle der höheren Layer entgegennimmt und in Standard-BLE-Paket verschachtelt. Siehe dazu Abbildung 9.

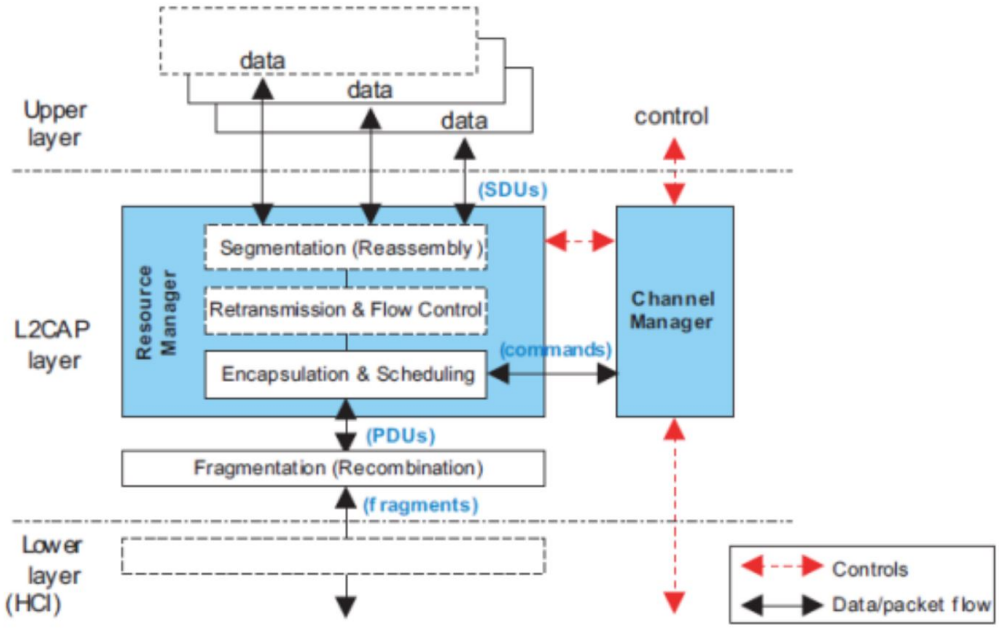

Abbildung 9: L2CAP Architektur Blocks [28]

Des Weiteren dient L2CAP als eine Art Transportschicht, vergleichbar mit TCP im OSI-Modell. Es fragmentiert zu grosse Pakete und setzt sie entsprechend wieder zusammen. Auf diese Weise können

mehrere Protokolle gleichzeitig und nahtlos auf einem einzigen physischen Link existieren. Bei BLE bedient L2CAP hauptsächlich zwei Protokolle: ATT und SMP [24].

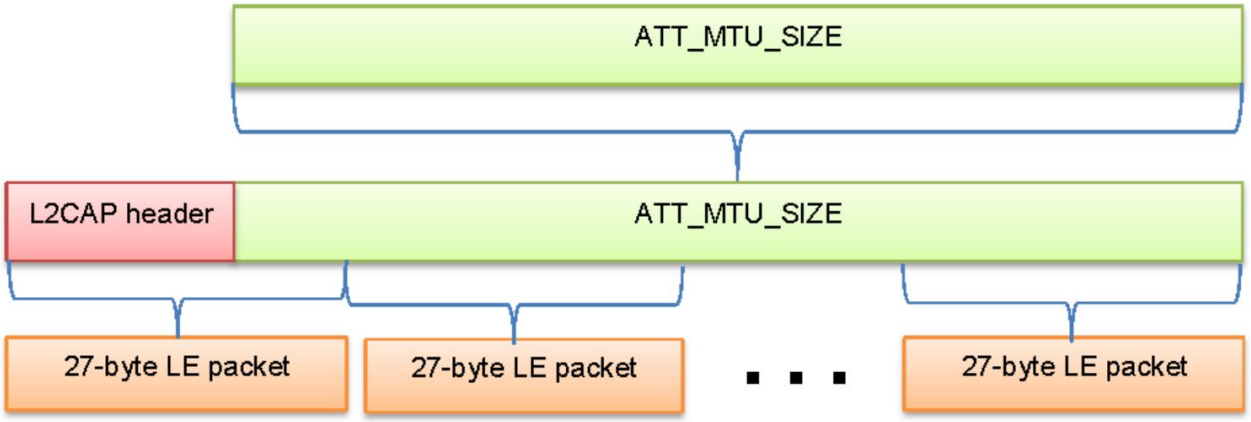

Abbildung 10: L2CAP Fragmentierung von Paketen [28]

Die MTU sollte aber nicht zu gross gewählt werden, denn das führt häufig zu mehr Datenverlust und einem erhöhten Energiebedarf.

Attribute-Protokoll (ATT)
Das Attribute-Protokoll (ATT) ist ein einfaches Client/Server-Protokoll. Eine wichtige Eigenschaft, welche berücksichtigt werden muss, ist, dass es **stateless** [29] ist. Dies bedeutet, dass sich der Client/Server keine vorherigen Zustände oder deren Reihenfolge merken. Das kann zur Folge haben, das Informationen verloren gehen.
Wie weiter bereits erwähnt, wird das ATT-Protokoll von L2CAP entgegengenommen und in entsprechende Pakete verschachtelt. L2CAP ist zustandsorientiert.
ATT-Operationen können optional Acknowledgements (ACKs) enthalten. Somit lassen sich sehr robuste Applikationen bauen. Weil aber der Sender auf die ACKs warten muss, wird die Durchsatzrate beeinträchtigt. Operationen, welche ohne ACKs ausgeführt werden, gehen dennoch nicht verloren, weil sich der Link Layer um das erneute Senden von verlorenen Paketen kümmert. Jedoch weiss man hierbei nicht, welche Pakete erfolgreich übertragen wurden [30].

In dieser Arbeit fungiert die App als Client und der Smart Trainer als Server. Die Daten sind in Form von Attributen organisiert und werden mittels 'Universally unique identifier (UUID)' zugeordnet. Es ist Aufgabe des Clients, die gelieferten Daten des Servers richtig zu interpretieren. Wenn der Client Daten sendet, müssen sie in einer konsistenten Form an den Server geliefert werden. Der Server kann jederzeit die Daten ablehnen, wenn sie in ein falsches Format haben [24]. Wie ein Attribut im Detail aussieht, wird vom Generic Attribut (GATT) beschrieben. Dazu folgt ein eigenes Kapitel.

5.1.2 Bluetooth GATT

GATT ist die Abkürzung für **G**eneric **Att**ributes. Es ist eine hierarchische Datenstruktur, von Meldungen zwischen Bluetooth-Geräten. Es baut auf dem ATT-Protokoll auf. GATT beschreibt im Detail, wie ein Attribut aussieht und wie die Daten bei einer BLE-Verbindung übertragen werden. Es befasst sich hauptsächlich mit Datentransfer-Abläufen und den Formaten der Attribute.
Das GATT-Profil beinhaltet Services, welche wiederum zusammenhängende Characteristics enthalten. Eine gute Illustration des GATT-Profil befindet sich in der Abbildung 11. Services und Characteristics werden in den Kapiteln 5.1.3 und 5.1.4 beschrieben. Das GATT-Profil wird beim klassischen Bluetooth (Basic Rate) nicht implementiert, dieses verwendet lediglich angepasste Profile.

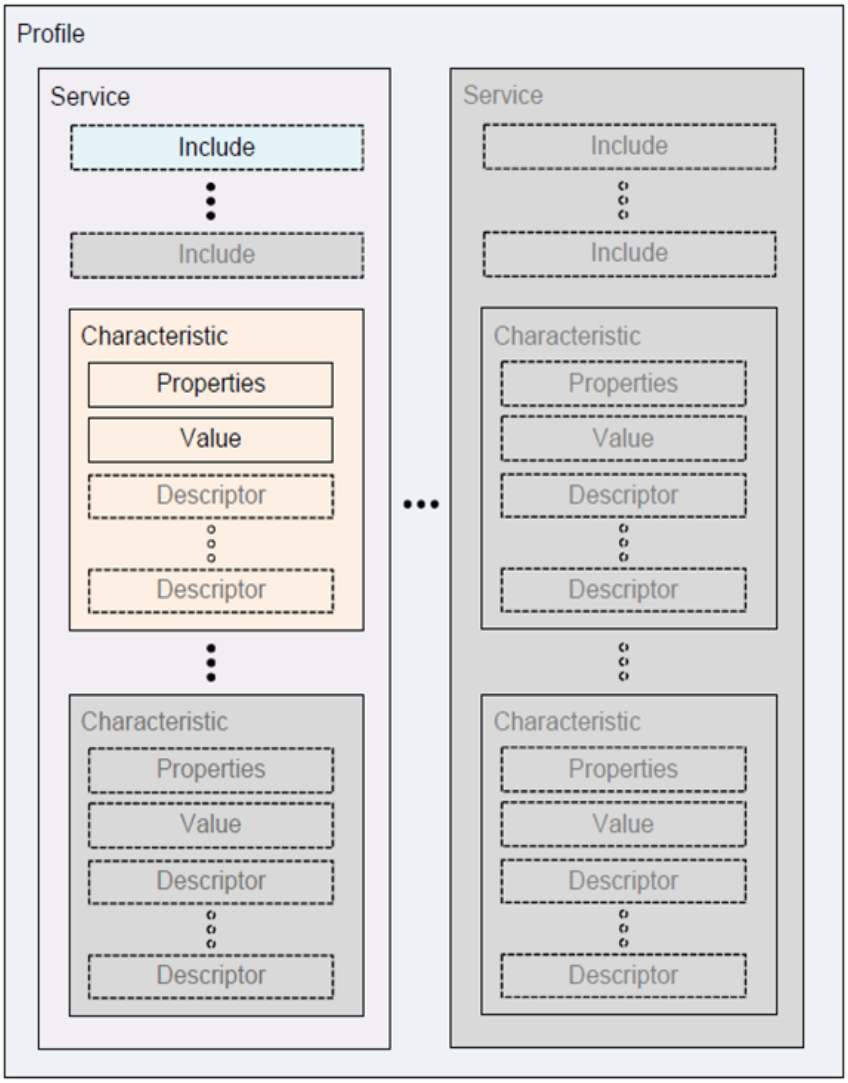

Abbildung 11: Hierarchie vom GATT-Profil [31]

Das GATT-Profil beinhaltet die beiden Rollen Client und Server. Weil der Elite Drivo als Server fungiert, wird die App entsprechend der Client.

GATT-Client
Der GATT-Client sendet Anfragen an einen Server und erhält Antworten. Zu Beginn weiss der GATT-Client nichts, über die Attribute des Servers. Deshalb muss zuerst ein Service-Discovery erfolgen. Sobald der Client weiss, was vom Server alles implementiert und unterstützt ist, können Lese- und

Schreib-Operationen ausgeführt werden.

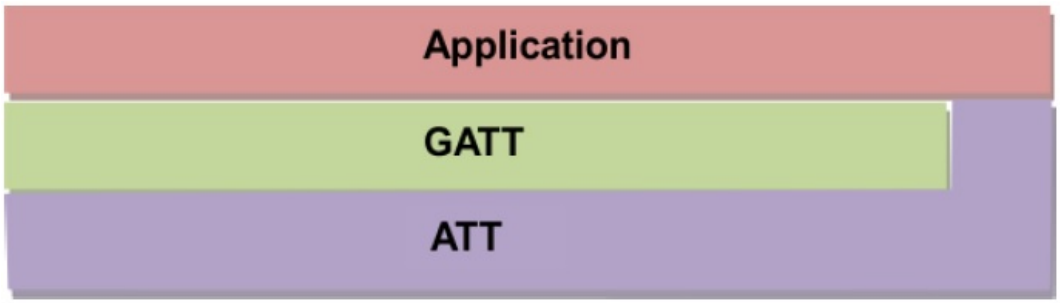

Abbildung 12: Abstraktion des GATT-Client [32]

Der GATT-Client wird, wie in der Abbildung 12 dargestellt, abstrahiert. Diese Abstraktion ist Teil des BLE-Standards und wird nicht näher erläutert.

GATT-Server
Der GATT-Server empfängt Anfragen von einem Client und sendet Antworten zurück. Es gibt auch Notifications, welche er von sich aus an den Client sendet, falls er so konfiguriert wurde. Er ist zuständig, die vorhanden Daten dem Client zur Verfügung zu stellen. Jedes BLE-Gerät muss zumindest einen einfachen GATT-Server implementieren, welcher auf Client-Requests antwortet - auch wenn es lediglich eine Fehlermeldung ist, welche zurückgeschickt wird [24].

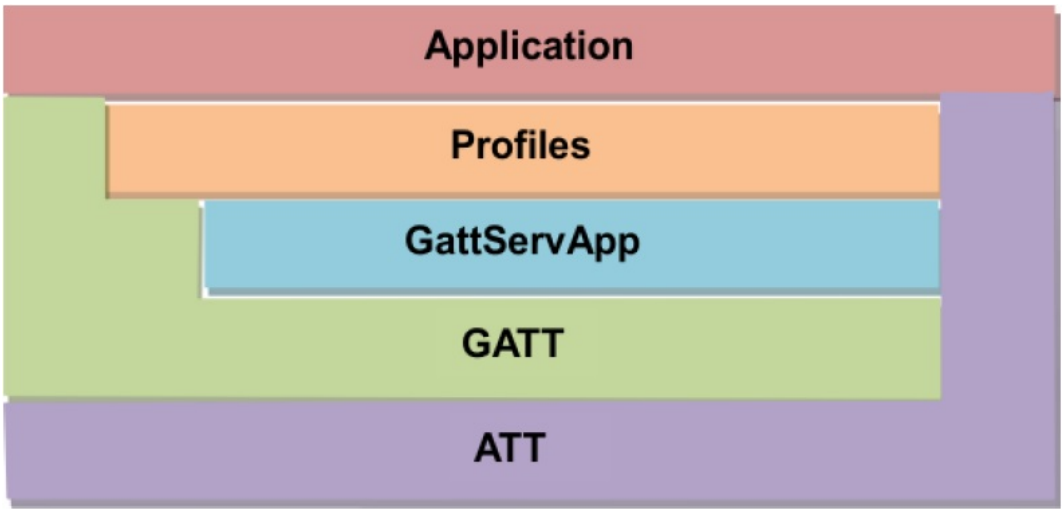

Abbildung 13: Abstraktion des GATT-Server [32]

Wie der GATT-Client wird auch der GATT-Server abstrahiert (Abbildung 13). Diese Abstraktion ist Teil des BLE-Standards und wird nicht näher erläutert.

5.1.3 GATT Services

GATT Services sind eine Sammlung/Gruppierung von konzeptuell zusammenhängenden Characteristics, welche beschreiben, wie sich ein Teil eines BLE-Geräts verhält [33]. Neben dem Namen des Services, ist vor allem auch die dem Service zugewiesene Nummer (UUID) von grosser Bedeutung. Damit ist ein Service eindeutig identifizierbar. Ein BLE-Gerät kann mehrere Services unterstützen. Welche dies beim Testgerät sind, wird im Kapitel 5.1.5 beschrieben.

5.1.4 GATT Characteristics

GATT Characteristics sind definierte Attribut-Typen, welche einen einzelnen logischen Wert enthalten [34]. Sie sind zentral für den Datenaustausch bei BLE-Geräten, weil hier definiert ist, welche Werte übertragen, gelesen oder geschrieben werden. Characteristics sind immer einem Service zugewiesen. Auch sie besitzen eine eindeutige UUID, über welche sie angesprochen werden.

5.1.5 GATT UUIDs

Die UUID ist eine 128-Bit lange eindeutige ID. Sie wird verwendet um die Services und Characteristics des GATT-Profils zuzuordnen. Es gibt bereits sehr viele Spezifikationen für BLE-Geräte wie z.B. Herzfrequenz-Sensoren, Fitness Maschinen oder Benutzerinformationen. Des Weiteren kann ein Unternehmen eine eigene, proprietäre UUID erhalten und diese für eigene Services verwenden. Die Standard Bluetooth-Spezifikationen der Characteristics und Services haben folgendes Format:

- 0000**XXXX**-0000-1000-8000-00805f9b34fb

Jene Services und Characteristics welche in den Standard kommen, erhalten eine Nummer zugewiesen, welche für den Platzhalter 'XXXX' eingefügt werden kann. Z.B. hat der Fitness Machine Service die Nummer 0x1826. Welche Standard-Services und Characteristics vom Elite Drivo unterstützt werden, kann der Tabelle 4 entnommen werden.

GATT Service	**UUID**	**GATT Characteristic**	**UUID**
Generic Access	0x1800	Device Name	0x2A00
		Appearance	0x2A01
		Peripheral Preffered Connection Parameters	0x2A04
Generic Attribute	0x1801	Service Changed	0x2A05
Cycling Speed and Cadence	0x1816	CSC Measurement	0x2A5B
		CSC Feature	0x2A5C
		Sensor Location	0x2A5D
Cycling Power Service	0x1818	Sensor Location	0x2A5D
		Cycling Power Measurement	0x2A63
		Cycling Power Feature	0x2A65
		Cycling Power Control Point	0x2A66
Fitness Machine Service	0x1826	Fitness Machine Feature	0x2ACC
		Indoor Bike Data	0x2AD2
		Training Status	0x2AD3
		Supported Resistance Level Range	0x2AD6
		Supported Power Range	0x2AD8
		Fitness Machine Control Point	0x2AD9
		Fitness Machine Status	0x2ADA
Device Information Service	0x180A	Serial Number String	0x2A25
		Firmware Revision String	0x2A26
		Hardware Revision String	0x2A27
		Software Revision String	0x2A28
		Manufacturer Name String	0x2A29

Tabelle 4: Vom Elite Drivo unterstützte Standard GATT-Characteristics

Das Testgerät (Elite Drivo) hat ausserdem noch einen Elite-eigenen Service. Dieser heisst 'Elite Real Trainer' und hat folgendes Format:

- 347b**0001**-7635-408b-8918-8ff3949ce592

Alle Characteristics innerhalb des Elite-Services haben das gleiche Format. Lediglich die vier markierten Stellen sind anders. Welche eigenen Characteristics vom Elite Drivo unterstützt werden, kann der Tabelle 5 entnommen werden.

GATT Service	UUID	GATT Characteristic	UUID
Elite Real Trainer	0x0001	Trainer Brake	0x0010
		Out of Range Flag	0x0011
		Trainer Capabilities	0x0017
		System Weight	0x0018
		Unbekannt	0x0012
		Unbekannt	0x0013
		Unbekannt	0x0014
		Unbekannt	0x0015
		Unbekannt	0x0016

Tabelle 5: Proprietäre Elite-GATT-Characteristics

5.1.6 Attribute

Attribute sind die kleinste Dateneinheit bei GATT und ATT. Sie sind adressierbar und können verschiedene Daten und Informationen enthalten. Sowohl GATT als auch ATT können ausschliesslich mit Attributen arbeiten, deshalb müssen alle Daten von Client und Server in Form von Attributen organisiert sein. Der Attribut-Wert ist dann der eigentliche Dateninhalt (Byte), welcher geschrieben oder gelesen wird.

5.1.7 Properties

Wie in Abbildung 11 zu sehen ist, besitzt jede GATT-Characteristic ein Property-Flag. Es gibt diverse Typen von Properties [35]. Am häufigsten anzutreffen sind *Read*, *Write*, *Notify* und *Indicate*. Sie beschreiben, wie mit einer Characteristic interagiert werden kann. Wie der Name schon verrät, können bei 'read' Lese- bzw. bei 'write' Schreiboperationen ausgeführt werden. Bei 'notify' sendet der GATT-Server die Daten an den Client, wenn sich etwas ändert. Der Client braucht diese nicht extra anzufordern, jedoch muss er die Notification vorher auf dem Server einschalten. Ähnlich wie 'notify', muss auch 'indicate' eingeschaltet werden. Auf Indications lassen sich anschliessend Schreiboperationen ausführen, welche vom GATT-Server beantwortet werden (ACKs). Dabei kann er die Befehle akzeptieren oder verweigern und den GATT-Client so informieren.

5.2 Fitness Machine Service

Weil bei 'Tour de Maison' der 'Fitness Machine Service' (FTMS) von zentraler Bedeutung ist, ist dieser hier noch separat beschrieben. Alle Informationen stammen aus der FTMS-Dokumentation [36].

FTMS ist ein Standard-Service und stammt von Bluetooth SIG. Wie es der Name sagt, kommt er vor allem bei Fitness-Geräten zum Einsatz. Dies können Smart Trainer, Laufbänder, Rudergerät usw. sein. Welche Characteristics das Testgerät 'Elite Drivo' unterstützt, kann in Tabelle 4 nachgelesen werden. Weil in der 'Tour de Maison'-App mit den beiden Characteristics 'Indoor Bike Data' (IBD) und 'Fitness Machine Control Point' (FTMSCP) interagiert wird, sind diese in den nachstehenden Kapiteln beschrieben.

5.2.1 'Indoor Bike Data'-Characteristic

Die 'Indoor Bike Data'-Characteristic enthält die meisten trainings-relevanten Daten. So lassen sich beispielsweise die Geschwindigkeit, Kadenz oder zurückgelegte Distanz auslesen. Nicht jedes Trainings-

Gerät unterstützt die gleichen Funktionen. Die beiden ersten Byte (Flags) der IBD-Characteristic beschreiben die vorhanden Features. Die darauffolgenden Bytes sind die Werte der jeweiligen Features. Ein Beispiel-Datensatz der IBD-Characteristic befindet sich in der Tabelle 6.

Byte	0 1	2 3	4 5	6 7 8	9 10	11 12	13 14
Wert	74 08	00 00	00 00	5D 07 00	45 00	00 00	E4 62
Flag	Features	Speed	Cadence	Total Distance	Resistance Level	Power	Elapsed Time

Tabelle 6: Beispielhaftes Byte-Array der 'Indoor Bike Data'-Characteristic

Welche weiteren Flags es gibt und wie der Aufbau des Feature-Flags ist, kann direkt dem 'Indoor Bike Data'-Kapitel der FTMS-Dokumentation entnommen werden ([36, p. 43]). In FTMS-Dokumentation sind auch die Einheiten und Auflösungen der einzelnen Flags angegeben.

5.2.2 'Fitness Machine Control Point'-Characteristic

Die 'Fitness Machine Control Point'-Characteristic wird für das Einstellen des Widerstands auf dem Smart Trainer benötigt. Sie ist vom Property-Typ 'indicate'. Es gibt verschiedene Flags, auf welche zugegriffen werden kann. Der Smart Trainer gibt auf jeden Zugriff eine Antwort. Vor der Verwendung der Characteristic müssen die Indications eingeschaltet werden. Dies erfolgt auf dem GATT-Client. In der App 'Tour de Maison' wird auf fünf verschiedene Flags zugegriffen, welche in der Tabelle 7 beschrieben sind.

Flag	Definition	Parameter	Beschreibung
0x00	RequestControl	Keine	Kontrolle des Smart Trainer erlangen
0x01	Reset Trainer	Keine	Trainer zurücksetzen (IBD-Werte nullen)
0x04	Target Resistance	UInt8 (0-200)	Trainer-Widerstand in %
0x07	Start/Resume Training	Keine	Training starten
0x08	Pause/Stop Training	UInt8	Training stoppen (01)/pausieren (02)

Tabelle 7: Flags und Parameter der 'Fitness Machine Control Point'-Characteristic

Neben den in Tabelle 7 aufgeführten Flags, gibt es noch zahlreiche weitere. Sie sind im FTMS-Dokument beschrieben ([36, p. 50] - Tabelle 4.15). Der Smart Trainer antwortet auf jeden Befehl mit einer 80 gefolgt vom Flag und dem Resultat. Die Resultat-Werte liegen im Bereich von 01-05. Welche Bedeutungen sie haben, kann ebenfalls dem FTMS-Dokument ([36, p. 64] - Tabelle 4.24) entnommen werden.

5.2.3 Datenrate und Leistung von BLE

Die Datenübertragungsrate liegt bei BLE bei 1 Mbit/s. Die Sendeleistung beträgt meistens 10mW, womit die Reichweite unter idealen Bedingungen bei ca. 40 Metern liegt.

Ein erheblicher Unterschied zum klassischen Bluetooth liegt in der Nettodatenraten. Diese ist mit 0.27 Mbit/s weniger als die Hälfte der Basic Rate [37]. Ausserdem kann ein Verbindungsaufbau bereits nach 3ms statt 100ms erfolgen und eine Datenübertragung bereits nach 6ms abgeschlossen sein. So wird der geringe Energiebedarf bei BLE sichergestellt [38].

5.3 ANT+

ANT+ baut auf dem 'ultra-low power ANT'-Protokoll auf. Bei ANT handelt es sich um ein proprietäres Protokoll von dynastream, einer Tochterfirma von Garmin. Es ist für jedermann nutzbar. Eine Dokumentation ist öffentlich verfügbar [39, 40]. ANT ist BLE ähnlich, jedoch liegt der Schwerpunkt bei ANT auf Sensordaten. Deshalb ist es nicht für grössere Datenmengen bestimmt.

Es unterstützt ausserdem diverse Topologien: Peer-to-peer, star, connected star, tree und fixed mesh Topologien sind möglich. Das Protokoll arbeitet auf dem 2.4 GHz ISM-Band. Es nutzt die RF Frequenzen zwischen 2400MHz bis 2524MHz, wobei die Frequenz 2457MHz für ANT+ Geräte reserviert ist.

ANT+ setzt auf standardisierte Profile um die Kommunikation zwischen Sensor und Gerät herstellerunabhängig zu gewährleisten. Es gibt eine grosse Auswahl an verschiedenen Profilen, welche hauptsächlich für sportliche Aktivitäten beschrieben werden. Zusätzlich werden auch Smart-Home, Altenpflege oder Spitalanwendungen angeboten.

5.3.1 ANT Protokoll

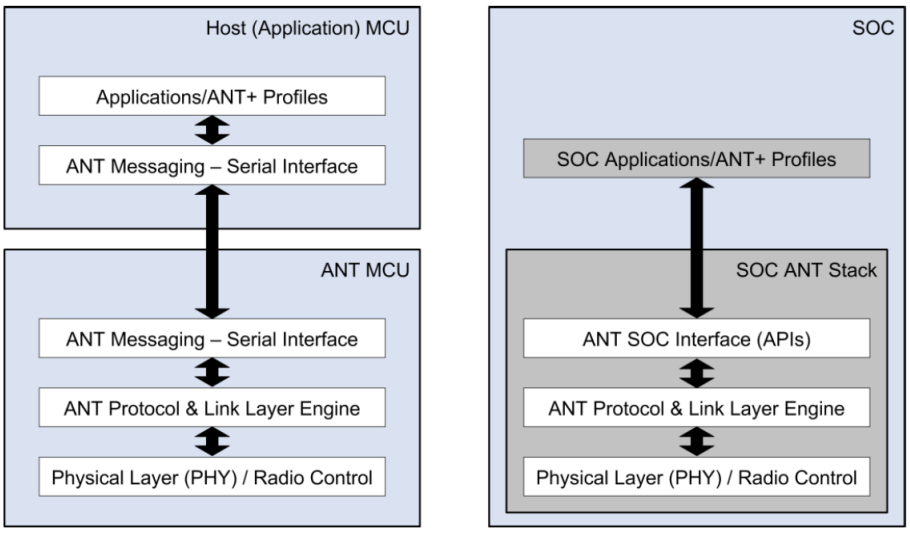

Abbildung 14: ANT Layers [40, p. 8]

ANT übernimmt die Bearbeitung des Physical, Network und Transport Layer. Zudem sind Verschlüsselungsmechanismen auf tiefster Ebene verfügbar (Siehe Abbildung 14).
Das Interface zwischen dem ANT-Gerät und dem Host ist simpel gehalten. Ein Mikrocontroller (Micro Controller Unit, MCU) kann auf Serieller Ebene mit dem ANT MCU über ANT Messages kommunizieren (siehe Abbildung 15). Somit können einfache Mikrocontroller an ANT-Netzwerke angeschlossen werden. Eine weitere Möglichkeit bietet ein System on a chip (SOC). Dabei sind ANT Modul und Mikrocontroller in einem Chip vereint.

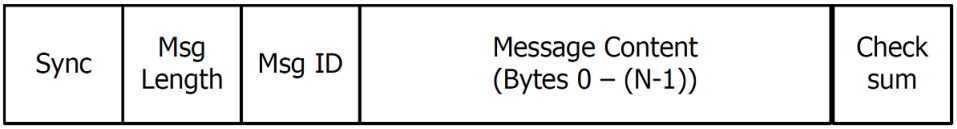

Abbildung 15: ANT Messanging Packet [40, p. 34]

Abbildung 15 zeigt eine ANT Message auf serieller Ebene. Alle Felder bis auf den Message Content sind 1 Byte lang. Die Definitionen der verschiedenen Message IDs sind in der Dokumentation der ANT Spezifikation (Kapitel 9) zu entnehmen [40, p. 49]. Die Checksumme wird durch ein XOR aller zuvor enthalten Bytes inklusive SYNC gebildet.
Eine Erweiterung bietet das Extended Message Format, welches zusätzliche Informationen zur Verfügung stellt. Es gibt zwei verschiedene Arten: flagged und legacy. Welche Version zur Anwendung kommt, hängt dabei vom verwendeten ANT-Gerät ab.

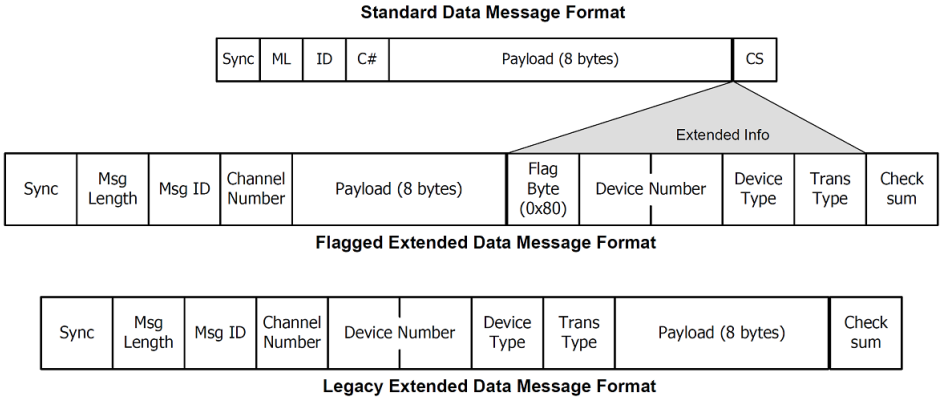

Abbildung 16: ANT Messanging Packet [40, p. 35]

In Abbildung 16 sind die Unterschiede der verschiedenen Formate zu sehen. In den erweiterten Daten können beispielsweise Werte für RSSI, Timestamp etc. versendet werden. Weitere Informationen befinden sich in Kapitel 7 der ANT Spezifikation [40, p. 34].

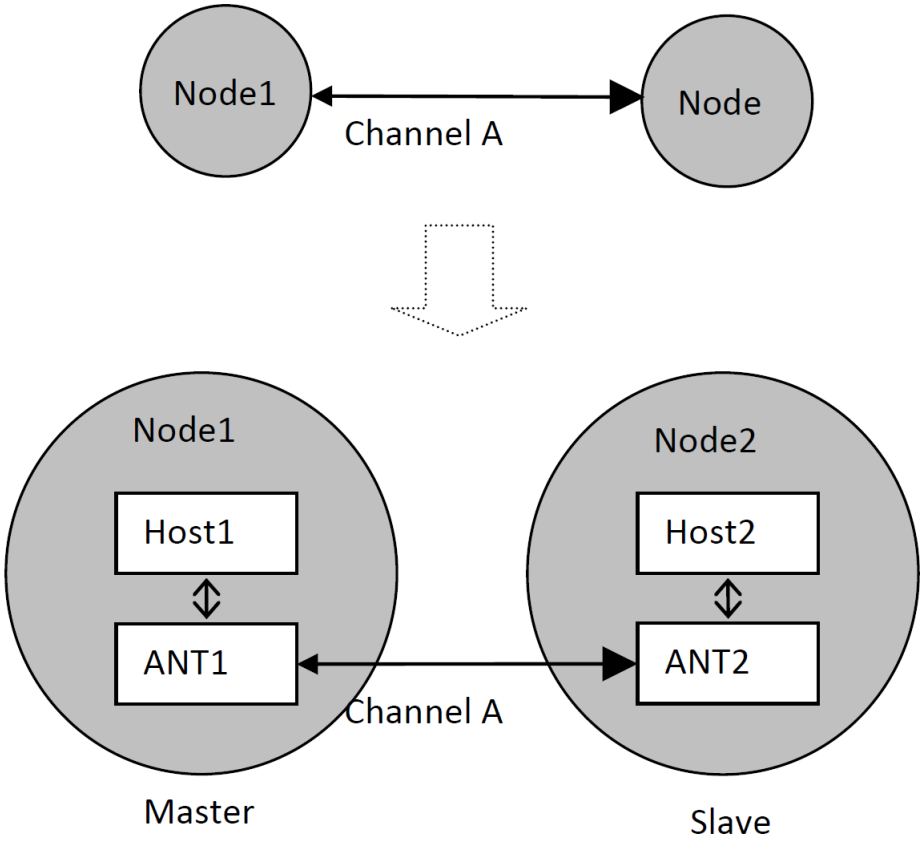

Abbildung 17: ANT Messanging Packet [40, p. 13]

Die Kommunikation zwischen ANT Nodes findet über Channel statt. Eine Verbindung hat mindestens einen Master sowie einen Slave (siehe Abbildung 17). Die wichtigen Bestandteile eines Channels sind: Channel Type, RF Frequency, Channel ID, Channel Period und Network.
Es gibt verschiedene Channel Typen, welche in der ANT-Dokumentation in Kapitel 5.2 beschrieben sind [40, p. 13]. Für ANT+ Geräte werden meist die Channel «Bidirectional Slave» oder «Bidirectional Master» verwendet. ANT bietet 125 einzigartige RF Frequenzen. Die Berechnungsformel ist RF_Frequency_val = (Vorgesehehne Frequenz – 2400 MHz) / 1 MHz. ANT+ nutzt dabei die Frequenzen 2450Mhz und 2457Mhz. Diese sollten laut Dokumentation von nicht ANT-Geräten vermieden werden.

Die Channel ID ist essentiell für den Verbindungsaufbau und setzt sich aus Transmission Type, Device Type und Device Number zusammen. Es handelt sich um ein vier Byte grosse Feld mit den vorherig genannten Feldern. Mit dem 'transmission type' werden die verfügbaren Übertragungsmöglichkeiten beschrieben. In den jeweiligen ANT+ Profilen werden die möglichen Typen definiert.

Mithilfe der Channel Period wird die Anzahl Messages definiert. Es handelt sich um ein 16 Bit Feld, welches mit folgender Formel berechnet wird: Channel_Period_val = 32768 / MessageRate(Hz) Die Standard Rate ist 4 Hz, der dazugehörige Wert würde also 8192 lauten. Die Messagerate ist direkt proportional zum Stromverbrauch abhängig. Eine kleine Channel Period erlaubt schnellere Datentransfers und schnellere Gerätesuche. Mit dem Network wird der Netzwerktyp definiert. Es gibt public, managed und private networks. ANT+ ist ein managed network. Die verschiedenen ANT+ Profile beschreiben dabei die Verwendung des Netzwerks.

5.3.2 ANT+ Profile

Wie bereits beschrieben handelt es sich bei ANT+ um ein Managed network. Die verschiedenen Profile beschreiben, welche Daten wie versendet werden. Es gibt für jedes Profil eine Dokumentation. Kompatible Geräte sind einfach anhand der verwendeten Symbole zu erkennen (siehe abbildung 18). Für einen Smart Trainer wichtige ANT+ Profile sind in der Tabelle 8 ersichtlich:

Profil	**Name**	**Beschreibung**	**Benötigte Kanäle**
PWR	Leistungssensor	Zeigt die abgegebene Leistung in Watt	1
CTF	Kurbel Drehmoment Frequenz	Zeigt das Drehmoment	1
SPD	Geschwindigkeitssensor	Zeigt die Geschwindigkeit	1
CAD	Kadenzsensor	Zeigt die Kadenz der Tritte	1
SPD CAD	Geschwindigkeits- und Kadenzsensor	Je ein SPD und CAD Sensor	2
S&C	Geschwindigkeits- und Kadenzsensor	SPD und CAD kombiniert in 1 Sensor.	1
FE-C	Fitness Equipment Controls (FE-C)	Erlaubt das Steuern von Smarten Fitnessgeräten	min. 2

Tabelle 8: Übersicht ANT+ Profile

Abbildung 18: ANT+ Profil SPD, CAD, S&C [41, p. 1]

Für jegliche ANT+ Profile ist eine Dokumentation auf thisisant.com verfügbar. Sie enthalten weitergehende Informationen zum Gebrauch und setzen der Parameter [42]

5.3.3 Datenrate und Leistung

Wie bereits erwähnt, eignet sich ANT nicht für grössere Daten. Trotzdem bietet ANT eine Dateiübertragung an. Der Standard heisst ANT-FS und bietet eine maximale Übertragungsgeschwindigkeit von ungefähr 10kbps [43, p. 41]. ANT bietet verschiedene Übertragungsarten. Im normalen Modus sind bis zu 12.8kbit/s möglich. Im Burst Modus 20kbit/s und im Advanced Burst 60kbits [40, p. 23].

Die Reichweite wird in den offiziellen Dokumenten von thisisant.com nicht genannt. Es besteht einzig ein Forenbeitrag, welcher die maximale Reichweite bei Sichtverbindung mit 30m und bei normalen Bedingungen mit 10m angibt [44].

ANT ist ein auf möglichst niedrigen Stromverbrauch ausgelegtes Protokoll. Je nach Anwendung ist es möglich, einen Sensor mit einer Knopfzelle zu betreiben. Auf der Website thisisant.com ist ein Energieverbrauchsrechner vorhanden. Mit Hilfe dieses Tools kann der theoretische Stromverbrauch ermittelt werden [45].

5.4 ANT vs. BLE

In der Tabelle 9 werden die Technologien ANT und BLE verglichen. Auf den Ersten Blick ist ersichtlich, dass diese Technologien sehr ähnlich sind. Beide Senden auf demselben Frequenzband und nutzen die gleiche Modulation. ANT zeigt Stärken im geringen Energieverbrauch und den möglichen Topologien, während BLE mit der Reichweite und Geschwindigkeit trumpft.

Technologie	**ANT**	**ANT+**	**Bluetooth**	**Bluetooth LE**
Standardisation	Proprietär	Proprietär	Bluetooth SIG	Bluetooth SIG
Topologien				
Point-to-point	✓	✓	✓	✓
Star	✓	✓		✓
Tree	✓	✓		
Broadcast	✓	✓		
Scanning Mode	✓	✓		
Shared Cluster	✓	✓		
Mesh	✓	✓		✓
Scatternet			✓	✓
Band	2.4 GHz	2.4 GHz	2.4 GHz	2.4 GHz
Reichweite	30m bei 0dBm		1-100m	10-600m
max. Datenrate	Broadcast: 12,8 kbit/sBurst: 20 kbit/s Advanced Burst: 60kbit/s	Broadcast: 12,8 kbit/sBurst: 20 kbit/s Advanced Burst: 60kbit/s	1-3 Mbit/s	125 kbit/s,250 kbit/s,500 kbit/s,1 Mbit/s,2 Mbit/s
max. Teilnehmer	65533 pro Kanal (8 Kanäle)		1 master7 active slaves,200+ inactive	1 master,7 slaves,mesh: 32767
Sicherheit	AES 128 / 64	128 Bit AES-CTR	56-128 Bit key	AES-128
Modulation	GFSK	GFSK	GFSK	GFSK
Fehlerkorrektur	XOR	XOR	CRC	CRC
Energie verbrauch	-	2 mW [46]	-	10 mW

Tabelle 9: Vergleichstabelle ANT und Bluetooth

5.5 Kommunikationstechnologie für 'Tour de Maison'

'Tour de Maison' ist eine Android-Applikation. Weil Android-Geräte nur sehr selten eine ANT+ Unterstützung besitzen, wird die gesamte Applikation auf BLE entwickelt. Es gäbe zwar die Möglichkeiten einen BLE zu ANT-Konverter zwischen Smartphone und Smart Trainer zu schalten. Allerdings wandelt dieser lediglich die BLE-Pakete in entsprechende ANT-Pakete um. Daher hilft dies nicht weiter. Die Applikation muss dennoch über das BLE-Protokoll funktionieren, weshalb kein ANT verwendet wird.

6 'Tour de Maison' für Anwender

Dieses Kapitel richtet sich an den Endnutzer der App 'Tour de Maison'. Es wird erklärt, wie die App zu bedienen ist und welche Besonderheiten zu beachten sind.

6.1 Voraussetzungen

Die App wurde für Android entwickelt. Es wird mindestens die Android-Version 4.3 vorausgesetzt. Die App wurde nur mit dem Elite Drivo getestet. Andere Smart Trainer wurden zwar nicht getestet, sollten sie aber die Bluetooth-Standards implementiert haben, könnte die App trotzdem funktionieren.

6.2 Installation

Im Google Play Store ist die App nicht zu finden. Dazu mussdie .apk Datei auf das Smartphone übertragen werden und danach über einen Datei-Explorer installiert werden. Nach der Installation müssen die Berechtigungen für den Standortzugriff genehmigt werden.

6.3 Bedienungsanleitung

Vor dem Starten der App muss sowohl Bluetooth, als auch der Ortungsdienst aktiv sein. Andernfalls können keine BLE-Geräte gefunden werden. Nach dem Starten muss man eine Verbindung zum Smart Trainer aufbauen. In Abbildung 19 ist der Startscreen von 'Tour de Maison' zu sehen.

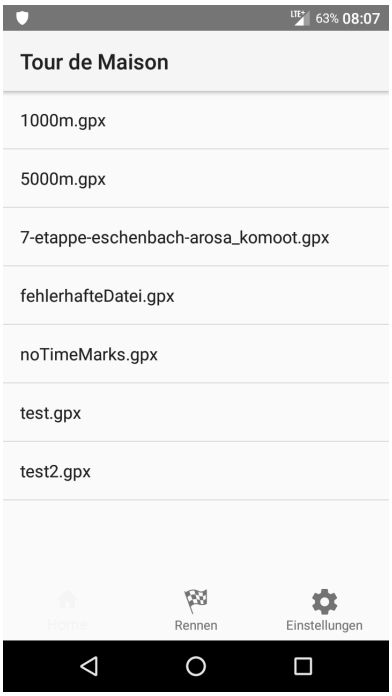

Abbildung 19: 'Tour de Maison' Startscreen - Etappenübersicht

In den Einstellungen ist der Assistent zum Verbinden mit dem Smart Trainer zu finden. Nachdem Klick auf 'Connection' wird der Assistent zum Verbinden aufgerufen (siehe Abbildung 20). Aus der Liste kann der gewünschte Smart Trainer ausgewählt und verbunden werden.

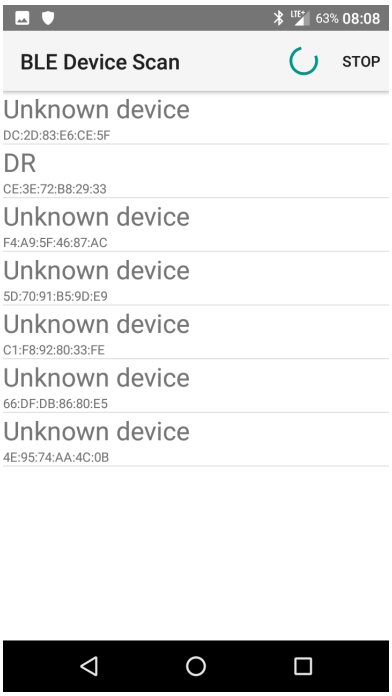

Abbildung 20: 'Tour de Maison' Verbindungsassistent

Nach dem Auswählen des Smart Trainers wird automatisch zur Auswahl der Etappe weitergeleitet (siehe Abbildung 21). Der grüne Balken (AppBar) signalisiert dabei eine vorhandene Verbindung. Sollte die Verbindung unterbrochen werden, wird dies durch einen roten Balken signalisiert. In diesem Fall muss der Smart Trainer neu verbunden werden.

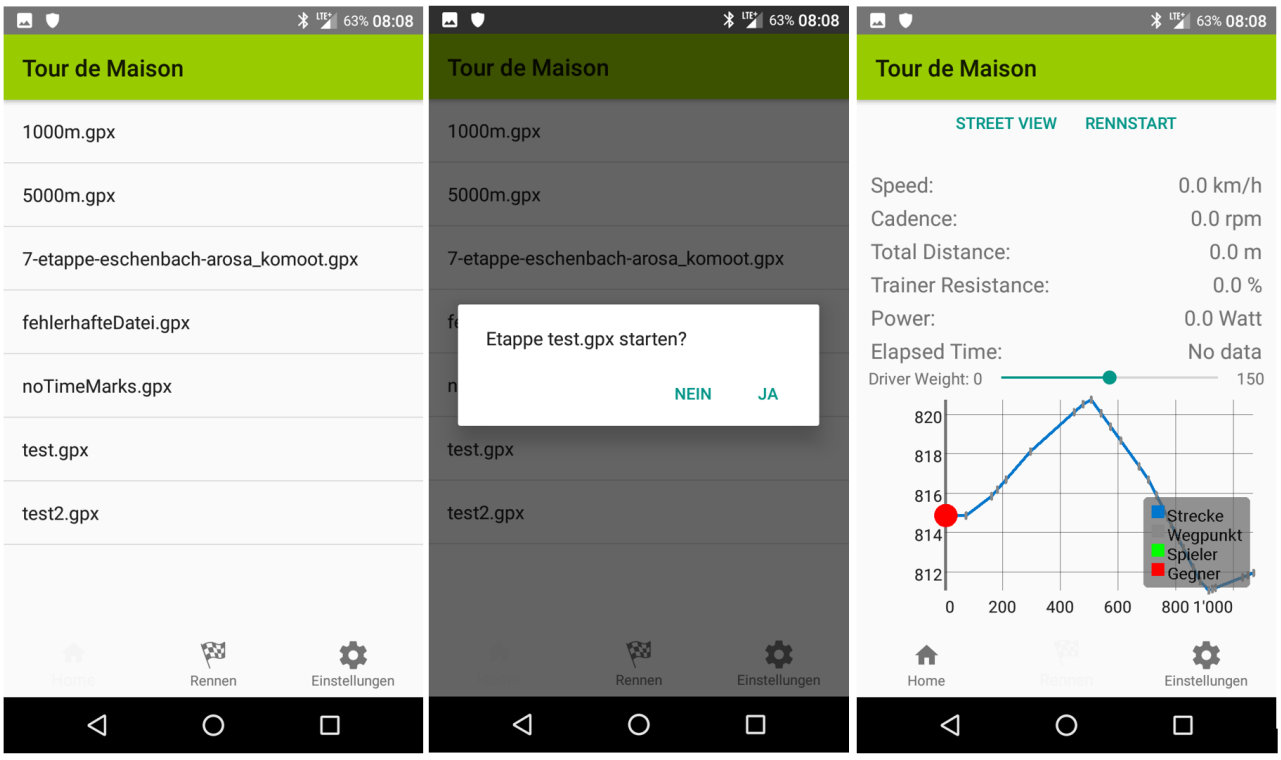

Abbildung 21: 'Tour de Maison' Ablauf Etappenauswahl

Wenn eine Etappe ausgewählt wurde, kann mit dem Button 'Rennstart' das Rennen gestartet werden. Mit Stop kann das Rennen gestoppt werden, der Smart Trainer der virtuelle Gegner werden dadurch pausiert.

6.4 Werte

Geschwindigkeit, Kadenz, Distanz, Widerstand und die aktuelle Leistung werden auf dem Bildschirm angezeigt. Die Graphen zeigen einerseits das Höhenprofil mit aktueller Position und die Änderungsrate der Steigung in Prozent (siehe Abbildung 22) an. Im Landscape-Modus ist zusätzlich das aktuelle Bild von Google Street View zu sehen.

Abbildung 22: 'Tour de Maison' Rennen

6.5 Besonderheiten BLE

Um die Bluetooth Low Energy Technologie (BLE) in Android zu nutzen, muss der Ortungsdienst aktiviert werden. Erst dann werden verfügbare BLE-Geräte aufgelistet [47]. 'Tour de Maison' ortet jedoch nicht den Benutzer. Es wird einzig die BLE Schnittstellen angesprochen.

7 'Tour de Maison' für Entwickler

7.1 Software-Architektur

In Abbildung 23 ist die Software-Architektur der 'Tour de Maison'-Applikation dargestellt. Im Zentrum stehen die Activities welche die anzuzeigenden Fragments mit deren Adaptern laden. Ein Fragment enthält z.B. die Liste mit allen verfügbaren Etappen. In der Schicht 'Resources' befinden sich die XML-Layouts der Activities und Fragments. In den Values sind die Farben und 'Application Themes' definiert.
Eine wichtige Regel in der Software Entwicklung ist es, keine Abhängigkeiten aus unteren Schichten in die Oberen zu haben. Wie im Architekturdiagramm ersichtlich ist, greifen daher die UI-Packages auf die unteren Schichten zu. Dazu gehört das View Model, welches alle Daten während der App-Laufzeit persistiert. Des Weiteren kommuniziert der Bluetooth Low Energy Service mit dem Smart Trainer. Die Daten, welche er vom Smart Trainer empfängt, werden mittels Broadcast an die Activity übergeben. Hilfsklassen, welche z.B. die Leistung berechnen oder einen String in ein Byte-Objekt umwandeln, wurden in ein eigenes Utils-Package ausgelagert.
Bei grösseren Applikation ist häufig noch eine mittlere Schicht 'Business-Logik' zwischen UI und Library anzutreffen. Dadurch ist die Software noch besser abstrahiert, wodurch es weniger Abhängigkeiten gibt. Bei 'Tour de Maison' ist der grösste Teil der Logik in der Klasse 'MainActivity.java' abgehandelt. Bei einem Ausbau der App sollte die Logik aus der Activity in eine eigene Schicht extrahiert werden.

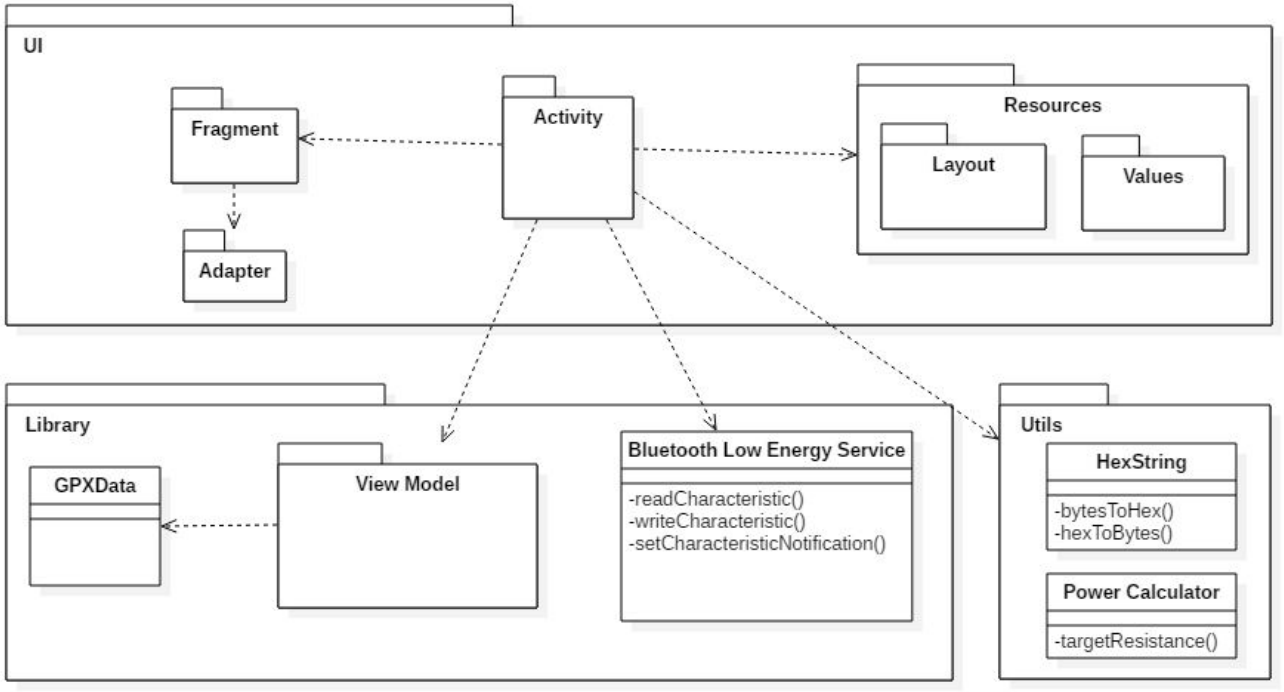

Abbildung 23: Software-Architektur der 'Tour de Maison'-App

7.2 Zustandsdiagramme

Die Reihenfolge der Schreibbefehle auf dem Smart Trainer muss geregelt sein, ansonsten kann es leicht zu App-Abstürzen kommen oder zur Befehls-Verweigerung seitens Smart Trainer. Um die Vorgänge besser zu verstehen, wurden zwei Zustandsdiagramme erstellt. Eines visualisiert den Verbindungsaufbau mit dem Aktivieren der GATT-Notifications. Das andere den Vorgang zum Beschreiben der 'Fitness Machine Control Point'-Characteristic.

7.2.1 Verbindung, Notifications und Indications

In der 'Tour de Maison'-App werden vor allem die beiden GATT-Characteristics 'Indoor Bike Data' (IBD) und 'Fitness Machine Control Point' (FTMSCP) angesteuert. 'Indoor Bike Data' ist vom Property-Typ 'Notification'. Dies bedeutet, dass der GATT-Server die Daten an den Client sendet, auch wenn dieser sie gerade nicht angefordert hat. Grundsätzlich sendet der Server Daten, wenn sie sich verändern. Weil die IBD-Characteristic ein 'Elapsed Time'-Byte besitzt, welches die abgelaufene Zeit (in Sekunden) seit dem Start des Trainings bzw. seit dem Einschalten des Smart Trainer misst, schickt dieser sekündlich alle Daten der IBD-Characteristic. Damit der Smart Trainer die Daten zu senden beginnt, müssen vorher die Notifications aktiviert werden.

'Fitness Machine Control Point' hat den Property-Typ 'Indication'. Sie sind von der Funktionsweise sehr ähnlich wie Notifications, allerdings erfolgt auf ein Paket eine Antwort (ACKs). Wenn also ein FTMSCP-Befehl an den Smart Trainer geschickt wird, sendet dieser eine Antwort zurück. Die Befehl wird dabei entweder akzeptiert oder verweigert. Es gibt verschiedene Typen von Antworten (siehe Kap. 5.2.2) Wie bei den Notifications auch, müssen die Indications zuerst eingeschaltet werden.

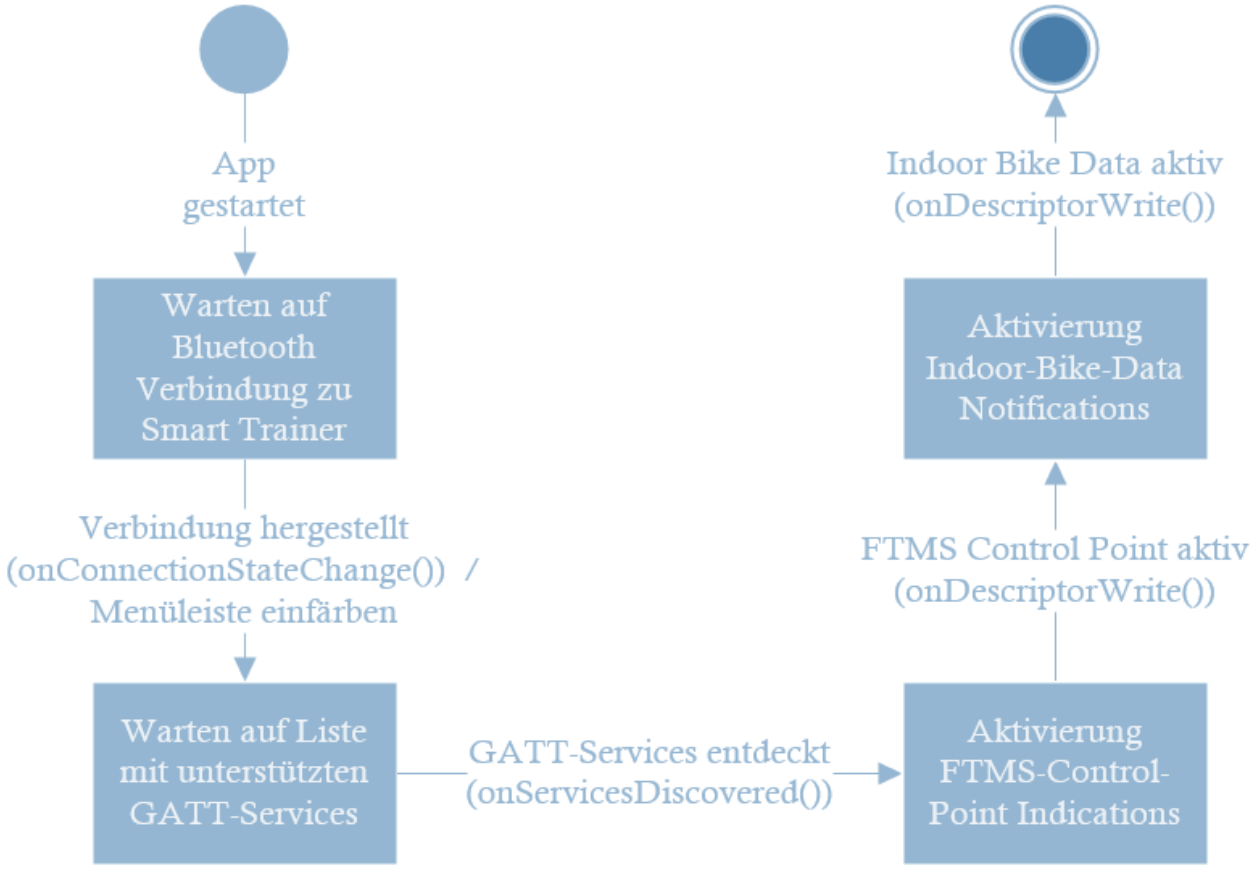

Abbildung 24: Zustandsdiagramm 'Verbindung, Notifications und Indications'

Wie in Abbildung 24 aufgezeigt, durchläuft das Programm verschiedene Zustände, bevor die IBD-Notification und FTMSCP-Indication eingeschaltet werden. Zunächst wird überhaupt eine Bluetooth-Verbindung benötigt. Anschliessend müssen die unterstützten GATT-Services des Smart Trainers abgefragt werden. Erst wenn diese bekannt sind, darf auf die einzelnen Characteristics zugegriffen werden, weil es ansonsten zu einer 'NullPointer'-Exception kommen kann. Danach können die FTMSCP-

Indication und IBD-Notification aktiviert werden. Die Applikation darf immer nur einen Befehl auf dem Smart Trainer ausführen, bevor der nächste folgt. Die Applikation muss zwingend warten, bis der Smart Trainer wieder bereit ist, Befehle zu empfangen und bearbeiten. Wenn der Smart Trainer zu viele Befehle gleichzeitig erhält, akzeptiert er keinen davon - auch nicht den ersten - und die Befehle gehen verloren. Das Programm wird mittels GATT-Callbacks (z.B. onServicesDiscovered(), onCharacteristicWrite() usw.) informiert, welche Aktion der Smart Trainer gerade fertig ausgeführt hat. Danach kann ein neuer Schreibbefehl gesendet werden. Die GATT-Callbacks sind in der Klasse 'BluetoothLeService.java' implementiert. Beim Erhalt eines Callbacks wird geprüft, welcher Befehl ausgeführt wurde und dann die MainActivity darüber informiert. Erst jetzt löst diese wieder neue Befehle aus.

7.2.2 'Fitness Machine Control Point'-Interaktion

Nachdem die FTMSCP-Indication auf dem Smart Trainer aktiviert wurden, kann die FTMSCP-Characteristic beschrieben werden. Es werden verschiedene Fälle unterschieden (siehe 5.2.2). Die 'Fitness Machine Control Point'-Characteristic wird primär für das Setzen des Widerstands auf dem Smart Trainer verwendet und um diesen zu starten, zu pausieren oder zu stoppen. Damit er den Zugriff nicht verweigert, muss als Erstes die Kontrolle angefordert werden (0x00). Der Smart Trainer wird zunächst zurückgesetzt (0x01), damit die Zeit, Distanz usw. wieder auf null ist, bevor das Rennen beginnt. Nach einem Reset muss die Kontrolle (0x00) erneut angefordert werden, dann kann ein Training gestartet (0x07) und letztendlich der Widerstand (0x04 XX) eingestellt werden.

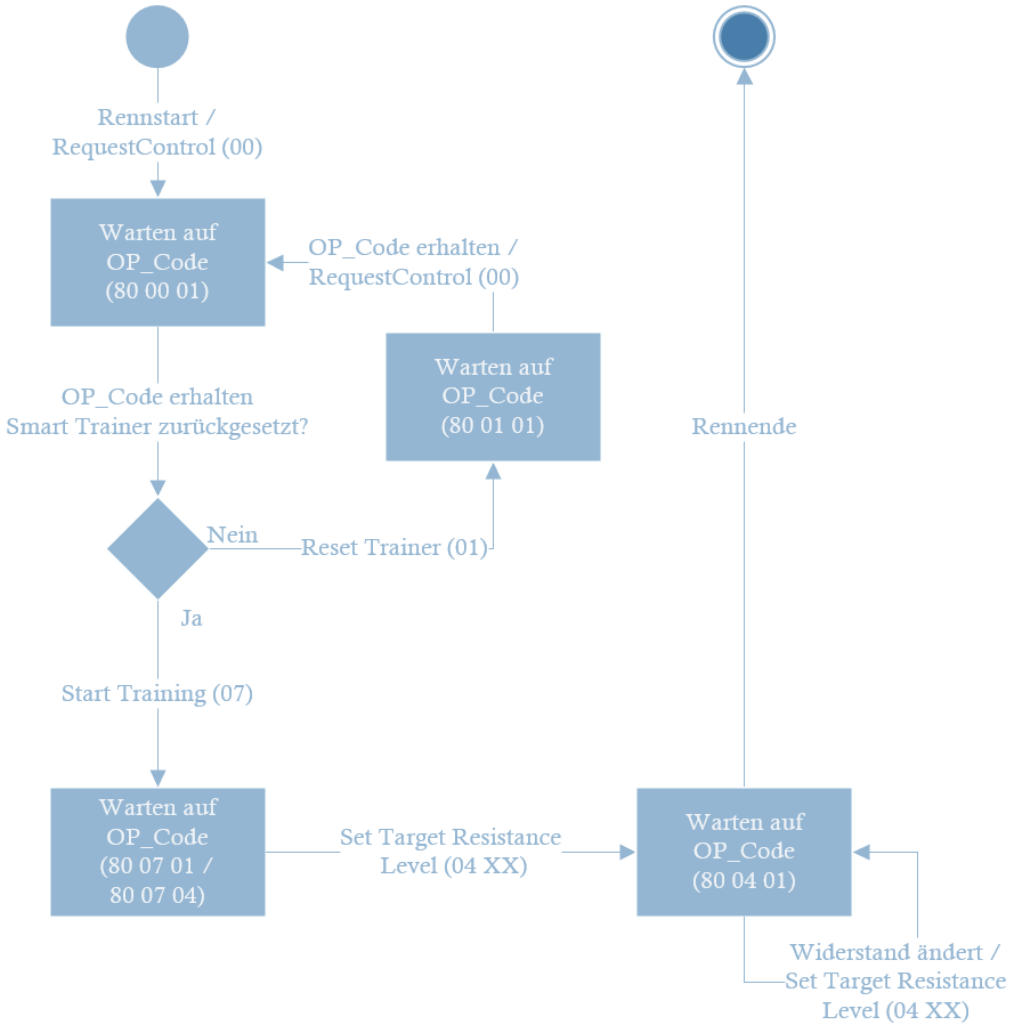

Abbildung 25: Zustandsdiagramm 'Fitness Machine Control Point'-Interaktion

7.3 Klassen

Nachfolgend werden alle Java-Klassen, welche für die Android-Applikation erstellt wurden, erläutert. Dies vereinfacht es, die ganze Applikation sowie die Überlegungen dahinter zu verstehen. Die drei Klassen 'BluetoothLeService.java', 'DeviceScanActivity.java' und 'SampleGattAttributes.java' stammen grösstenteils aus einer Google Sample App [48]. Deren Code wurde in der Analysephase verwendet, um sich bei den ersten Gehversuchen mit BLE an die Materie anzutasten. Der Bluetooth-Verbindungsaufbau der Sample App wurde übernommen und nicht neu geschrieben.

BluetoothLeService.java
In der Klasse 'BluetoothLeService.java' ist der Verbindungsaufbau zwischen dem Android-Gerät und dem Smart Trainer implementiert. Daher erfolgt die Kommunikation mit dem Smart Trainer über diese Klasse. Hier werden die Daten des Smart Trainer entgegengenommen und an die *MainActivity* weitergeleitet. Die Interpretation der empfangenen Daten wird in dieser Service-Klasse auf ein Minimum beschränkt, weil die Logik in den Activities und Fragments abzuhandeln ist. Es wird nicht gänzlich darauf verzichtet, weil es auch einige Vorteile haben kann. Die Abfrage des Verbindungsstatus (Connected/Disconnected) ist in dieser Klasse sinnvoll. Bei Änderungen des Verbindungsstatus wird entsprechend die MainActivity informiert. Ausserdem wird geprüft, von welcher GATT-Characteristic die Daten empfangen wurden. So lassen sie sich später in der MainActivity leichter verarbeiten.

Neben dem Empfang der Bluetooth-Daten, werden auch die auf dem Smart Trainer zu schreibenden Werte von der BluetoothLeService-Klasse aus verschickt. Es gibt zwei Arten von Schreibinteraktionen: Einerseits die gewünschten Bytes, welche in einer Characteristic einzustellen sind. Andererseits die Aktivierung der GATT-Notifications oder -Indications. Erst nachdem diese eingeschaltet wurden beginnt der Smart Trainer die Daten zu senden.

DeviceScanActivity.java
Wie die BluetoothLeService-Klasse stammt auch die DeviceScanActivity aus der Google Sample Applikation. In der 'DeviceScanActivity.java'-Klasse wird nach allen verfügbaren Bluetooth-Geräte gescannt. Die gefundenen Geräte werden in einer Liste dargestellt. Bei einem Klick auf ein Element aus der Liste versucht die Activity eine Verbindung zum Bluetooth-Gerät aufzubauen. Anschliessend wird die MainActivity aufgerufen und dort der aktuelle Verbindungsstatus mittels einfärben der Menüleiste (Android: ActionBar) dem Benutzer mitgeteilt. Wenn seit dem Start der App noch keine Verbindung aufgebaut wurde, bleibt diese weiss. Nachdem ein Bluetooth-Gerät erfolgreich verbunden wurde, wird die Menüleiste grün eingefärbt bzw. rot, sollte die Verbindung verloren gehen. Diese drei Status sind in Abbildung 26 illustriert.

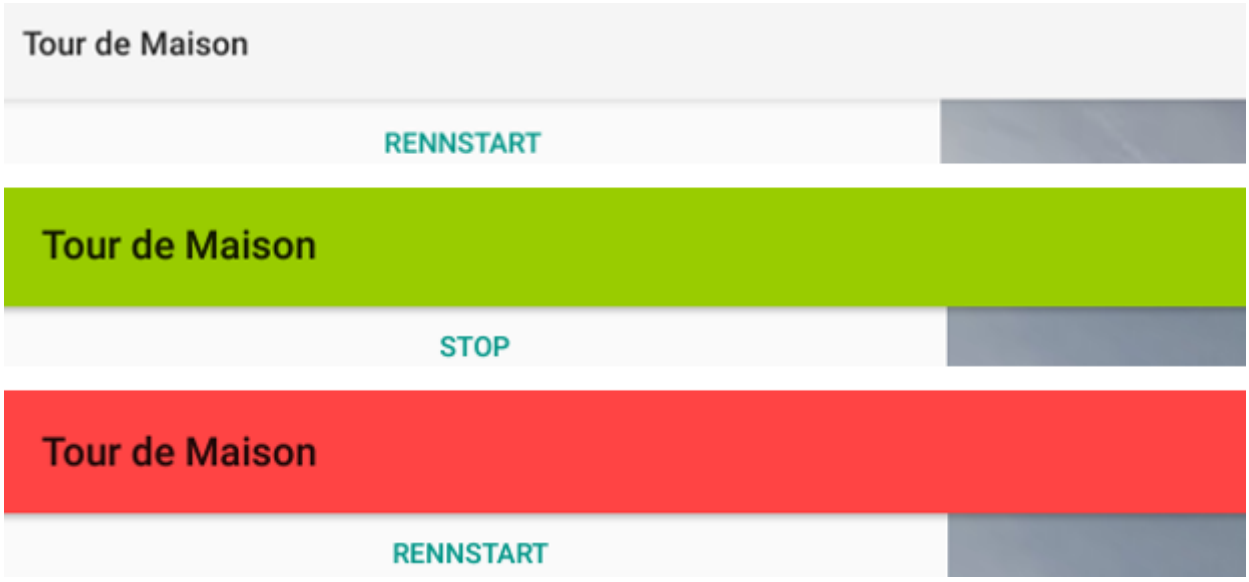

Abbildung 26: Verbindungsstatus - Weiss: Noch keine Verbindung aufgebaut, Grün: Verbunden, Rot: Getrennt

Beim Aufruf der MainActivity können dieser mittels Intent noch nützliche Parameter übergeben werden. So werden der Gerätename und die MAC-Adresse des Bluetooth-Geräts übergeben. Der Name wurde bislang nicht verwendet, aber er könnte dem Benutzer angezeigt werden, damit dieser weiss, womit er sich verbunden hat. Zur Wiederverwendung wird die MAC-Adresse in der MainActivity in einer Variable gespeichert. So versucht das Smartphone automatisch mit dem Bluetooth-Gerät eine neue Verbindung aufzubauen, falls diese zwischenzeitlich verloren ging.

GPXData.java
GPXData.java verwaltet die Positionsdaten auf den Streckenabschnitten. Sowohl die Wegpunkte als auch die aktuelle Position der Spieler werden verarbeitet.
Bevor die Aufgabe der Klasse erklärt wird, müssen erst die Begriffe Wegpunkt, Route und Track erläutert werden. Streckendaten mit GPS Koordinaten finden sich in vielen Anwendungen. Beispielsweise eine Routenplanung für Autofahrt, Radwegroute etc. Diese können in verschiedenen Formaten vorliegen. Meist findet man 'GPS Exchange Format (GPX)' und 'Keyhole Markup Language (KML)' vor.

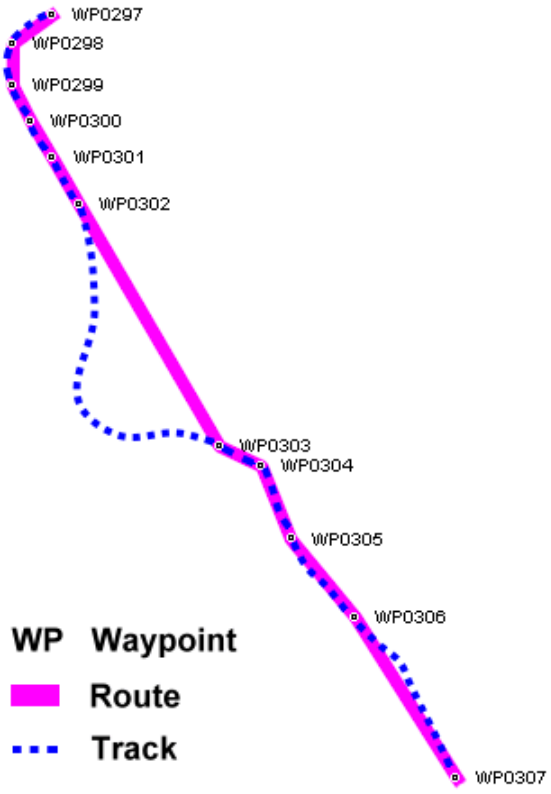

Abbildung 27: Unterschied Wegpunkt, Track und Route [49]

In Abbildung 27 ist der Unterschied zwischen Route und Track ersichtlich. Einzelne Wegpunkte werden mit Hilfe einer Linie zu einer Route verbunden. Diese können jedoch von dem eigentlichen Weg (Track) abweichen. Dabei entsteht ein gewisser Fehler. Die Strecke ist meistens kürzer als in der Realität. Mithilfe von Komoot lassen sich bestehende Streckendaten importieren und die jeweiligen Wegpunkte an bekannte Wege anpassen [50].

Im Konstruktor der Klasse wird der Pfad für die ausgewählte Etappe mitgegeben. Dabei handelt es sich um eine '.gpx'-Datei. Es werden nur '.gpx'-Daten akzeptiert, '.kml' wird nicht unterstützt. Die '.gpx'-Datei enthält die einzelnen Wegpunkte, welche wiederum über die Koordinaten, Höhe und einer Zeitmarke verfügen. Nach dem Parsen werden mehrere Arrays erstellt. Um die Logik möglichst einfach zu halten, entstpricht die Grösse der Arrays gleich der Anzahl Wegpunkte. Somit ist bei jedem Punkt der Höhenunterschied (dpsDistEle), die Steigung (dpsDistSlope) in Prozent und der Zeitabstand (timedifference) für den virtuellen Fahrer bekannt. In der Tabelle 10 ist der Aufbau der Arrays zu sehen. P steht für Wegpunkt. Die Funtkion totalLength addiert die einzelnen Abstände der Wegpunkte zusammen. Beim Aufruf gibt sie den Wert zur aktuellen Position zurück. Die Steigung wird mit der Formel "m = (y2 -y1) / (x2 -x1)"berechnet. Wobei für y die Höhe eingesetzt wird und für (x2 - x1) der Abstand zwischen den Punkten. Auffällig ist der Wert bei timedifference beim Index 0, die 4000 werden benötigt um den Countdown im RaceFragment zu ermöglichen. Andernfalls würde der virtuelle Fahrer beim Rennstart direkt losfahren.

Index	0	1	2	n
dpsDistEle	x = 0m y = Höhe P0	x = totalLength y = Höhe P1	x = totalLength y = Höhe P2	x = totalLength y = Höhe Pn
dpsDistSlope	x = 0m y = Steigung P0	x = totalLength y = Steigung(P1,P2)	x = totalLength y = Steigung(P2,P3)	x = totalLength y = Steigung(Pn,Pn+1)
timedifference in millis	4000	Differenz(P1, P0)	Differenz(P2, P1)	Differenz(Pn, Pn-1)

Tabelle 10: Array von GPXData.java

HexString.java
Die Hilfsklasse 'HexString.java' erlaubt es, Byte-Daten in einen hexadezimalen String umzuwandeln oder umgekehrt. Dies ist erforderlich, um die Daten in der App richtig anzuzeigen, bzw. um sie als Byte dem Smart Trainer zu übergeben. Die Konvertierung von Strings zu Bytes ist bei hardware-naher Programmierung (z.B. Mikrocontroller) sehr gängig. Deshalb wurde diese Hilfsklasse nicht selbst geschrieben, sondern stammt aus einem Projekt des Unternehmens 'Polidea' [51]. Sie unterliegt der Apache-2.0 Lizenz und darf dieser entsprechend verwendet werden.

MainActivity.java
'MainActivity.java' ist die grösste und wichtigste Klasse in der gesamten 'Tour de Maison'-Applikation. Sie wird beim Start der App aufgerufen. Sie ist immer aktiv, ausser während des Scanvorgangs der verfügbaren BLE-Geräte. Sie ist zuständig für das Laden und Ersetzen des aktuellen Fragments. Zudem enthält sie die Instanzen und Methoden für das Lesen und Schreiben auf dem Smart Trainer. Die empfangenen Byte-Daten werden zu dezimalen Zahlen konvertiert und im raceViewModel abgespeichert.

Ein sogenannter Broadcast-Receiver empfängt den Intent/die Daten der 'BluetoothLeService.java'-Klasse. So wird der Reihe nach geprüft, ob die Verbindung aufgebaut ist, ob die auf dem Smart Trainer verfügbaren GATT-Services geladen wurden und ob der Smart Trainer bereit ist um wieder neue Werte zu schreiben. Dies ist notwendig, weil ansonsten Schreibbefehle verloren gehen könnten. Wenn der Smart Trainer zu viele Befehle gleichzeitig erhält, ignoriert er sie. Deshalb dürfen neue Werte erst wieder geschrieben werden, wenn der Smart Trainer bereit ist.

Die MainActivity ist zudem für die Interaktion mit der 'Fitness Machine Control Point'-Characteristic zuständig. Dabei wird die benötigte Fahrerleistung von Watt in einen prozentualen Anteil der maximalen 'Target Resistance' umgerechnet. Dann wird dieser als Byte dem Smart Trainer übergeben und gesetzt. Neben dem eingestellten Widerstandswert werden alle weiteren Daten der 'Indoor Bike Data'-Characteristic gelesen und im RaceFragment angezeigt. Dies ist in Abbildung 28 ersichtlich. Damit lässt sich überprüfen, ob der Trainer die Einstellung richtig übernommen hat.

MyStageRecyclerViewAdapter.java
Diese Hilfsklasse wird für das befüllen der Liste im Fragment Stage benötigt.

powerCalculator.java
Die Berechnung der Leistung, welche der Fahrer auf dem Smart Trainer zu erbringen hat, erfolgt in der Klasse 'powerCalculator.java'. Dabei werden verschiedene Faktoren berücksichtigt, wie z.B. die aktuelle Fahrergeschwindigkeit, das Systemgewicht (Fahrer inkl. Fahrrad) und die Steigung. Zudem gibt es bereits vordefinierte Methoden zum Setzen der Windgeschwindigkeit, Fahrermasse oder des Drafting-Faktors. Zum jetzigen Stand werden noch nicht alle diese Methoden verwendet. Bei einem Ausbau der Applikation kämen sie zum Einsatz. Die verwendeten Formeln zur Berechnung befinden sich in Form von Kommentaren direkt im Code.

Abbildung 28: Darstellung der ausgelesenen 'Indoor Bike Data'-Werte und der 'Target Resistance'

RaceFragment.java

Im RaceFragment ist die gesamte Rennlogik implementiert. Alle angezeigten Daten stammen dabei vom raceViewModel. Es handelt sich um Observables, d.h. sie werden bei Änderungen laufend angepasst. Die Klasse setzt beim Starten eines Rennens den Trainer zurück. Zu gleich wird der Timer für den virtuellen Fahrer aktiviert. Während dem Rennen sind im RaceFragment die Graphen welche das Höhenprofil und die Steigung in Prozent vorhanden. Mit einem grünen Punkt wird der Smart Trainer Fahrer angezeigt, der Rote Punkt signalisiert die Position des Gegners. Die Punkte werden laufend aktualisiert und neu dargestellt.

Als weiteres Feature kann über den Button Streetview oder durch drehen des Geräts in den Landscape Modus das StreetViewFragment angezeigt werden. Beim Rennende wird der Gewinner über eine Mitteilung mittels Toast informiert.

raceViewModel.java

Eine der essentielleren Klassen ist 'raceViewModel.java'. Es handelt sich um eine Klasse, welche von Androids *ViewModel* [52] erbt. Sie wird für die Persistenz der Daten während der Lebenszeit der Applikation benötigt. Aufgrund der Funktionsweise von Android Applikationen und deren App-Lifecycle, ist es nicht immer bekannt, wann Android eine Activity bzw. ein Fragment beendet oder neu lädt. Dies kann jederzeit durch eine User-Interaktion geschehen. Beispielsweise wird eine Activity zerstört und neu erstellt, wenn die Orientierung des Smartphones (Portrait/Landscape) geändert wird. Objekte, Arrays, Variablen usw. überleben das Zerstören einer Activity nicht. Um den entstehenden Datenverlust zu vermeiden, persistiert man die Daten. Wenn sie lediglich während der App-Laufzeit verfügbar sein sollen, reicht ein ViewModel aus. Für eine längerfristige Aufbewahrung kann man eine Datenbank verwenden.

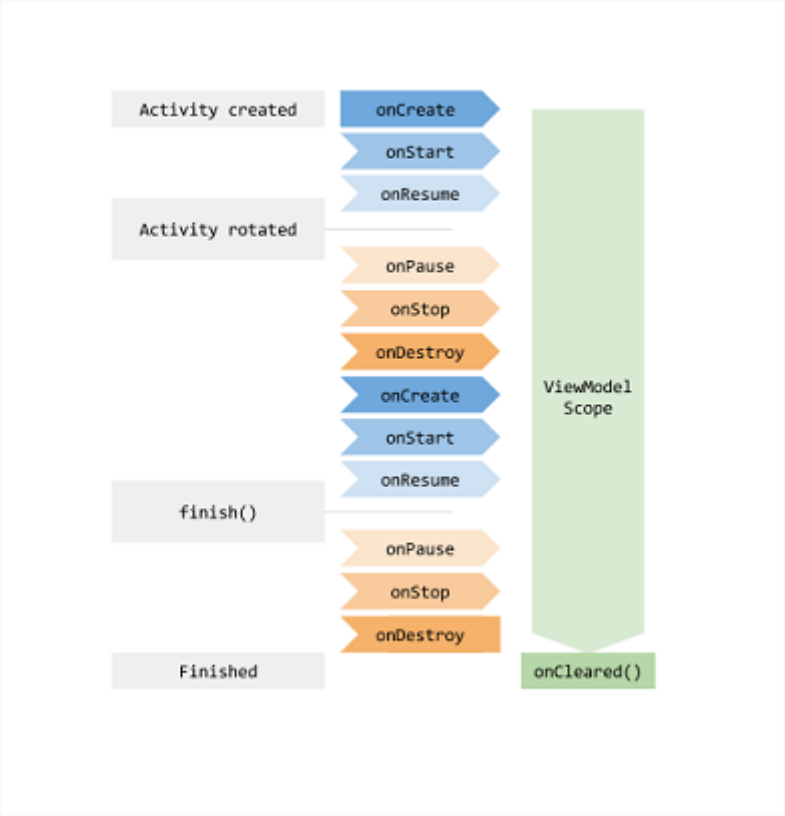

Abbildung 29: Lifecycle von Android Activities und ViewModels [52]

Bei der 'Tour de Maison'-App sind die Daten nur während der App-Laufzeit vorhanden. Beim Schliessen der App werden sie gelöscht. Daher reicht ein ViewModel aus. Die in 'raceViewModel.java' abgelegten Daten sollen immer aktuell sein. Des Weiteren dient die Klasse als 'Single source of truth (SSOT)'. Dadurch ist sichergestellt, dass die Daten genau einmal vorhanden sind, was Inkonsistenzen verhindert. Die Daten werden in MutableLiveData-Objekten abgelegt. Diese sind 'observable' was bedeutet, dass eine Activity oder Fragment informiert wird, sobald sich die Daten ändern. Darauf kann mit Methoden-Aufrufen reagiert werden um beispielsweise die UI-Elemente zu aktualisieren.

In Abbildung 29 ist der Lifecycle einer Android Activity sowie eines ViewModels dargestellt. Insbesondere ist es wichtig zu verstehen, welche Methoden einer Activity nacheinander aufgerufen werden und wovon der Aufruf dieser ausgelöst werden kann. Der Aufruf kann z.B. durch die Rotation des Smartphone ausgelöst werden oder auch wenn zwischen zwei Apps gewechselt wird. Sobald eine App in den Hintergrund gerät, wird die Activity zerstört. Ein ViewModel hingegen wird erst 'bereinigt', wenn die App ganz beendet wurde und nicht mehr im Arbeitsspeicher vorhanden ist.

SampleGattAttributes.java
Bei 'SampleGattAttributes.java' handelt es sich um eine Hilfsklasse. Hier können die UUIDs der GATT-Services und -Characteristics mit den entsprechenden Namen abgelegt werden. So lässt sich der Zugriff auf die gewünschten UUIDs vereinfachen, weil ihnen ein kurzer Name statt einer langen Nummer gegeben werden kann.

SettingsFragment.java
Die Klasse 'SettingsFragment.java' hat noch wenig Inhalt. Es wurde ein Knopf aufgebaut, um die De-

viceScanActivity aufzurufen, wo dann eine Verbindung zu einem BLE-Gerät gemacht werden kann. Bei einem künftigen Ausbau sollten in diesem Fragment z.B. Fahrer- und Fahrrad-Gewicht einstellbar sein. Für das Anpassen der Schwierigkeitsstufe könnten auch zurzeit fixierte Werte wie Rollwiderstand, Gravitation o.Ä. variabel konfiguriert werden.

StageFragment.java
Mit dieser Klasse wird das Auswählen der Etappe ermöglicht. Sie benötigt unbedingt die Hilfsklasse MyStageRecyclerViewAdapter um die Liste mit den vorhanden Dateien anzuzeigen. Während dem Aufruf von onAttach() werden die einzelnen Dateinamen aus dem assets Ordner geladen und dargestellt.

StreetViewFragment.java
Die Implementierung von Google Street View wird in einem eigenen Fragment abgehandelt. Dies als Fragment umzusetzen hat den Vorteil, dass sich die Darstellung auf den verschiedenen Bildschirmen von Smartphones und Tablets einfacher gestaltet. Die MainActivity wertet die Orientierung des Smartphone aus. Falls sich dieses im Landscape-Modus befindet, wird in der rechten Bildschirmhälfte das aktuelle Street View Bild angezeigt. In der Klasse 'StreetViewFragment.java' wird geprüft, ob eine Etappe geladen ist. Wenn nicht, wird ein entsprechender Text angezeigt. Des Weiteren wird in diesem Fragment die aktuelle Position (Latitude/Longitude) - und somit das passende Street View Bild - gesetzt. Die nächste Position wird erst abgerufen, wenn der Fahrer den GPS-Messpunkt der Etappe überschritten hat.

Damit das Sichtfeld des Street View Bildes in etwa der Fahrtrichtung/dem Strassenverlauf entspricht, wird neben der richtigen Position auch die Kamera-Ausrichtung (Englisch: Bearing) angepasst. Diese ist nicht immer ganz präzise. Sie wird aufgrund zweier nebeneinanderliegenden GPS-Messpunkte berechnet. In Fällen, wo die Messpunkte weit auseinander liegen, kann es vorkommen, dass die Ausrichtung falsch ist. Sie wird als Winkel in der 'Bearing'-Funktion der Street View Kamera mitgegeben. Der eingestellte Winkel ist relativ zum Norden (0°) und geht im Gegenuhrzeigersinn (Osten: 90°).

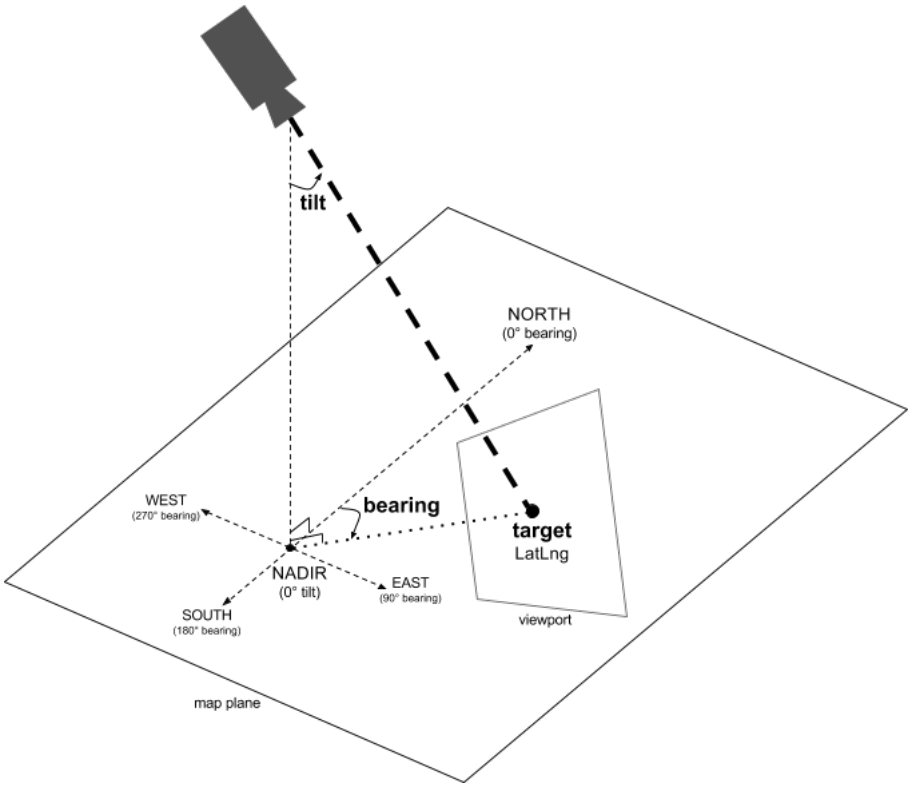

Abbildung 30: Visualisierung der Street View Kamera [53]

In Abbildung 30 ist Kamera bzw. das Sichtfeld von Street View visualisiert. Neben der Ausrichtung (bearing) lässt sich auch die Neigung einstellen. Somit könnte die Kamera auch bei steilen Passagen korrekt eingestellt werden. Weil aber bei den Beispieletappen die Steigungen nur sehr klein waren, wurde diese Funktion nicht eingesetzt. Der einzustellende Neigungswinkel lässt sich aufgrund des Höhenunterschieds von zwei GPS-Messpunkten berechnen. Der Winkel für die Ausrichtung wird in diesem Fragment lediglich gesetzt. Dessen Berechnung erfolgt in der Klasse 'GPXData.java'

Hinweis: Damit auf die Google Street View-API zugegriffen werden kann, wird ein Google Cloud-API-Key für die entsprechende API benötigte. Der in 'Tour de Maison' verwendete Key wurde von 'cnlab' zur Verfügung gestellt und ist im Android-Manifest der App hinterlegt.

Android-Manifest

Im Android-Manifest müssen verschiedene Punkte wie z.B. das App-Theme konfiguriert werden. Ausserdem sind hier die benötigten App-Berechtigungen gesetzt. Dies betrifft vor allem den Standort-Dienst, ohne diesen können keine BLE-Geräte gefunden werden. Daher muss zuerst die Berechtigungen erlaubt und anschliessend der Ortundgsdienst eingeschaltet werden. Vorher werden keine BLE-Geräte in der DeviceScanActivity angezeigt.

Neben den Berechtigungen ist hier auch der Key für den Zugriff auf die Google Street View-API hinterlegt.

7.4 Libraries

In der Tabelle sind die verwendeten Libraries aufgeführt.

Name	Beschreibung	Link
Street View	StreetView Anzeige	https://developers.google.com/maps/documentation/android-sdk/streetview
GPX Parser	Parser für .gpx Dateien	https://github.com/ticofab/android-gpx-parser
GraphView	Graphen Funktionalität	http://www.android-graphview.org/

Tabelle 11: Verwendete Libraries

8 Schlussfolgerungen

8.1 Fazit

Mit 'Tour de Maison' ist eine sehr interessante App entstanden, welche verschiedene Aspekte vereint hat. So wurde im Detail die Kommunikation mit BLE-Gerät studiert. Wie diese aufgebaut sind und wie mit ihnen interagiert werden kann, ist nun bekannt. Durch die Applikation wurden nicht nur sportliche Aspekte wie das Radfahren behandelt, sondern man hat sich auch intensiv mit der Android-Entwicklung auseinandergesetzt. Zudem wurde viel Neues über die Verwendung von Koordinaten und die Berechnung der Distanzen zwischen zwei Punkten gelernt. Die Integration von Google Street View macht die App für den Anwender interessant.

Einige Sachen kamen wie sie erwartet wurden, andere haben überrascht. So wurde am Anfang geplant, dass man mit Redmine und in einem Scrum-Vorgehen arbeiten wird. Die Planung mit Redmine wurde aber bereits nach wenigen Wochen nicht mehr konsequent geführt, weil die Wochensitzungen ausreichten. Welche Tasks bis zur nächsten Woche fertiggestellt werden sollten, funktioniert in einem so kleinen Team (zwei Studenten und zwei Betreuer) besser ohne ein grosses Planungs- und Versionierungstool. Des Weiteren hat sich Scrum als komplett obsolet entpuppt, weil ohnehin die meiste Zeit parallel gearbeitet wurde. Wenn es sich um zwei Entwickler handelt, welche auch noch die Arbeitsplätze direkt nebeneinander haben, weiss jeder ziemlich genau über die Arbeiten des anderen Bescheid. Durch die Aufgabenplanung in den Wochensitzungen und die parallel Arbeit erübrigt sich Scrum.

8.2 Ausblick

'Tour de Maison' kann als Prototyp betrachtet werden. Die App läuft robust und die wichtigsten Punkte wurden im Wesentlichen umgesetzt. Dennoch gibt es noch viel Ausbaumöglichkeiten. So könnte z.B. Google Street View noch für 'Virtual Reality'-Anwendungen angepasst werden. Dadurch gäbe es ein deutlich immersiveres Erlebnis.

Ein weiterer Ausbaupunkt wäre die Online-Competition, in der Hobby-Fahrer von zu Hause aus virtuell gegeneinander antreten könnten und dies in Echtzeit. Wenn hier das Verfolgerauto der Rennfahrer auch mitmachen würde, könnte man theoretisch die ganze Tour de Suisse mit den Profis mitfahren und sich mit ihnen vergleichen.

9 Abbildungsverzeichnis

Abbildungsverzeichnis

10 Literaturverzeichnis

Literatur

[1] wahoofitness.com, "5 indoor cycling apps to challenge you and your friends – wahoo fitness blog," 2016. [Online]. Available: https://blog.wahoofitness.com/5-indoor-cycling-apps-to-challenge-you-and-your-friends/

[2] L. Keller, "Bike-app zwift: So fährt man velo virtuell im wohnzimmer," 2018. [Online]. Available: https://www.blick.ch/digital/esports/indoor-trainings-app-zwift-so-trainiert-man-virtuell-auf-dem-velo-id8438638.html

[3] Alexander Altwasser, "Rollentrainer test | vergleichstabelle ergo- und virtual reality trainer," 2018. [Online]. Available: http://www.rollentrainer-test.de/vergleich_tabelle_ergotrainer_virtual_reality.html

[4] VirtualTraining s.r.o., "Rouvy | homepage - howitworks," 2018. [Online]. Available: https://rouvy.com/de/how-it-works

[5] J. Vincent, "This man is cycling around the uk in virtual reality using google street view," 2016. [Online]. Available: https://www.theverge.com/2016/8/9/12411262/cycle-vr-google-stree-view-uk

[6] SRF, "Tour de suisse: Srf bringt «veloclub» zurück ins fernsehen," 2018. [Online]. Available: http://www.srf.ch/medien/news/tour-de-suisse-srf-bringt-veloclub-zurueck-ins-fernsehen/

[7] Andreas Zorn, "Geschwindigkeit und leistung auf fahrrädern berechnen," 2008. [Online]. Available: http://www.kreuzotter.de/deutsch/speed.htm

[8] PBVertriebs-Gmbh, "ebikes.at: Leistungsrechner," 2011. [Online]. Available: http://www.elektro-fahrrad.at/service/beratung/leistungsrechner/

[9] Tariq Ali, "Direct drive vs wheel-on: What's the difference and which trainer should you buy | smart bike trainers," 2017. [Online]. Available: https://www.smartbiketrainers.com/direct-drive-vs-wheel-whats-difference-trainer-buy-3065

[10] zwift.com, "Zwift - supported trainers," 2018. [Online]. Available: https://zwift.com/hardware

[11] trainerroad.com, "Supported smart trainers with erg mode," 2018. [Online]. Available: https://www.trainerroad.com/equipment-checker

[12] kinomap.com, "Compatibility check," 2018. [Online]. Available: https://www.kinomap.com/en/compatibility

[13] bkool.com, "Bkool device compatibility," 2018. [Online]. Available: https://www.bkool.com/simulator/simulator2hEN

[14] rouvy.com, "Trainers & sensors – rouvy help desk," 2018. [Online]. Available: https://support.rouvy.com/hc/en-us/sections/115001264209

[15] fulgaz.com, "Apple tv and trainer compatibility | fulgaz," 2018. [Online]. Available: https://fulgaz.com/apple-tv-and-trainer-compatibility/

[16] strava.com, "How to get your activities to strava," 2018. [Online]. Available: https://support.strava.com/hc/en-us/articles/223297187-How-to-get-your-Activities-to-Strava#devices

[17] thesufferfest.com, "Smart trainer compatibility," 2018. [Online]. Available: https://support.thesufferfest.com/hc/en-us/articles/209334306-Smart-Trainer-Compatibility

[18] Google, "Developer guide | street view api | google developers," 2018. [Online]. Available: https://developers.google.com/maps/documentation/streetview/intro

[19] Geoportal, "Api faq — geoadmin api 3.0 documentation," 2018. [Online]. Available: http://api3.geo.admin.ch/api/faq/index.html

[20] ——, "Alle gebäude der schweiz gibt es nun digital und in 3d," 27.09.2018. [Online]. Available: https://www.swisstopo.admin.ch/de/home.detail.news.html/swisstopo-internet/news2018/news_release/20180927.html

[21] Google, "Preisgestaltung und -modelle – google maps platform | google maps platform | google cloud," 2018. [Online]. Available: https://cloud.google.com/maps-platform/pricing/?hl=de

[22] B. Ray, "Bluetooth vs. bluetooth low energy: What's the difference?" 2018. [Online]. Available: https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy

[23] Fares Othmane Mokthari, "Bluetooth low energy: introduction (first part), mindorks, medium," 2018. [Online]. Available: https://medium.com/mindorks/bluetooth-low-energy-3656ac323c4e

[24] ——, "Bluetooth low energy: speak up about the architecture - (second part)," 2018. [Online]. Available: https://medium.com/mindorks/bluetooth-low-energy-on-raspberry-second-part-516b5e8ad7c2

[25] Dev.Ti, "Bluetooth low energy scanning and advertising," 2018. [Online]. Available: http://dev.ti.com/tirex/content/simplelink_academy_cc2640r2sdk_1_12_01_16/modules/ble_scan_adv_basic/ble_scan_adv_basic.html

[26] ITWissen.info, "Ble (bluetooth low energy) :: Bluetooth-low-energy :: Itwissen.info," 23.08.2017. [Online]. Available: https://www.itwissen.info/BLE-Bluetooth-low-energy-Bluetooth-Low-Energy.html

[27] PunchThrough, "Maximizing ble throughput part 2: Use larger att mtu - punch through," 07.11.2016. [Online]. Available: https://punchthrough.com/blog/posts/maximizing-ble-throughput-part-2-use-larger-att-mtu

[28] Dev.Ti, "Logical link control and adaptation layer protocol (l2cap) — ble5-stack user's guide 1.00.00 documentation," 2018. [Online]. Available: http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_35_00_33/docs/ble5stack/ble_user_guide/html/ble-stack/l2cap.html

[29] Techtarget, "Was ist stateless (zustandslos)? - definition von whatis.com," 2018. [Online]. Available: https://whatis.techtarget.com/de/definition/Stateless-zustandslos

[30] Silabs.com, "Designing for bluetooth low energy applications," 2018. [Online]. Available: https://www.silabs.com/documents/referenced/white-papers/designing-for-bluetooth-low-energy-applications.pdf

[31] bluetooth.org, "Gatt overview | bluetooth technology website," 2018. [Online]. Available: https://www.bluetooth.com/specifications/gatt/generic-attributes-overview

[32] Dev.Ti, "Generic attribute profile (gatt) - stack," 2018. [Online]. Available: http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_35_00_33/docs/ble5stack/ble_user_guide/html/ble-stack/gatt.html

[33] bluetooth.org, "Gatt services | bluetooth technology website," 2018. [Online]. Available: https://www.bluetooth.com/specifications/gatt/services

[34] ——, "Gatt characteristics | bluetooth technology website," 2018. [Online]. Available: https://www.bluetooth.com/specifications/gatt/characteristics

[35] Oreilly.com, "Getting started with bluetooth low energy," 2018. [Online]. Available: https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html

[36] bluetooth.org, 14.02.2017. [Online]. Available: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=423422&_ga=2.268612660.433587979.1543218544-163624481.1543218544

[37] Lairdtech, "Ble overview | laird connectivity," 2018. [Online]. Available: https://connectivity.lairdtech.com/resources/blog/ble-overview-0

[38] Radio Electronics, "What is bluetooth smart | low energy ble | radio-electronics.com," 2016. [Online]. Available: https://www.radio-electronics.com/info/wireless/bluetooth/what-is-bluetooth-smart-low-energy-ble.php

[39] Garmin, "Ant+ documentation," 2018.

[40] H. Chin, "Ant message protocol and usage."

[41] K. Kent, "Ant+ device profiles: Bike speed, bike cadence, combined bike speed & cadence," 2016.

[42] thisisant.com, "Downloads - this is ant," 2018. [Online]. Available: https://www.thisisant.com/developer/resources/downloads/#documents_tab

[43] T. Zhao, "Wireless personal area network," 2012.

[44] kebinw, "The range of ant," 2012. [Online]. Available: https://www.thisisant.com/forum/view/viewthread/2782/

[45] thisisant.com, "Power estimator - this is ant," 2018. [Online]. Available: https://www.thisisant.com/developer/components/developer

[46] ITWissen.info, "Ant+ :: Ant plus :: Itwissen.info," 2018. [Online]. Available: https://www.itwissen.info/ANT-plus-ANT-plus.html

[47] stackoverflow.com, "Location needs to be enabled for bluetooth low energy scanning on android 6.0 - stack overflow," 2017. [Online]. Available: https://stackoverflow.com/questions/33045581/location-needs-to-be-enabled-for-bluetooth-low-energy-scanning-on-android-6-0#44291991

[48] Google, "googlesamples/android-bluetoothlegatt," 2018. [Online]. Available: https://github.com/googlesamples/android-BluetoothLeGatt

[49] Berklas, "Datei:wayroutrackp.png," 2008. [Online]. Available: https://de.wikipedia.org/wiki/Datei:Wayroutrackp.png

[50] Komoot.de, "Gpx-dateien exportieren und importieren – komoot feedback & hilfe," 2018. [Online]. Available: http://support.komoot.de/knowledgebase/articles/281653-gpx-dateien-exportieren-und-importieren

[51] Polidea, "Polidea/rxandroidble," 2018. [Online]. Available: https://github.com/Polidea/RxAndroidBle/blob/master/sample/src/main/java/com/polidea/rxandroidble2/sample/util/HexString.java

[52] Google, "Viewmodel overview | android developers," 2018. [Online]. Available: https://developer.android.com/topic/libraries/architecture/viewmodel

[53] ——, "Camera and view | maps sdk for android | google developers," 2018. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/views

11 Abkürzungsverzeichnis

Begriff	Erklärung
ACK	Acknowledge - Bestätigung einer Anfrage
ANT	Proprietärer Funkstandard der Firma Dynastream
API	Application Programming Interface, Englisch für Programmierschnittstelle
ATT	Attribute protocol - Bluetooth
BLE	Bluetooth Low Energy
CPS	Cycling Power Service
CSCS	Cycling Speed Cadence Service
FE-C	Fitness Equipment Control - ANT+ Profil
FTMS	Fitness Machine Service - BLE
FTMSCP	Fitness Machine Control Point - BLE
GAP	Generic Access Profile - BLE
GATT	Generic Attribute Profile - BLE
GPX	GPS Exchange Format
HCI	Host Control Interface - BLE
HRS	Heart Rate Service - ANT
IBD	Indoor Bike Data - BLE
KML	Keyhole Markup Language
L2CAP	Logic Link Control and Adaption Protocol - BLE
LL	Link Layer
MTU	Maximum Transmission Unit
PHY	Physical Layer
RSCS	Running Speed & Cadence Service
SM	Security Manager - BLE
SOC	System On a Chip
SSOT	Single Source Of Truth
UUID	Universally Unique Identifier
VR	Virtual Reality

12 Anhang

12.1 Persönliche Berichte

12.1.1 Reflexion Zarko Dragojevic

Obwohl ich das Modul 'Mobile und GUI Engineering', wo unter anderem Android beigebracht wird, bereits erfolgreich abgeschlossen habe, hatte ich noch wenig Kenntnisse in diesem Bereich. Zu Beginn der Studienarbeit hiess es, man müsse eine Android-App entwickeln, welche das und jenes kann. Auf den ersten Blick schien es keine schwierige Aufgabe zu sein. Wie viel sich aber hinter der ganzen Android-Entwicklung verbirgt, wurde nach und nach klarer. Dennoch gelang es die anfallenden Schwierigkeiten zu meistern. Hätte ich jedoch vor der Arbeit gewusst, was es alles an Know-How bedarf, hätte ich nie geglaubt, dass wir beide in der Lage sind, dies umzusetzen. Daher ist es schön zu sehen, wie viel man umsetzen kann, wenn man sich mit dem Thema befasst.

Was mich zusätzlich erstaunt hat, war die schwer zu findenden Dokumentation und Beispiele zu Bluetooth Low Energy. Es ist eine sehr gängige Technologie, daher hatte ich erwartet, dass man mehr Dokumentationen zu dem Thema findet. Dies erschwerte die ganze Einarbeitung in das Bluetooth-Thema. So ging gerade zu Beginn der Arbeit viel Zeit verloren. Würden man es daher nochmals neu machen, könnte man den Bluetooth-Teil sicherlich besser umsetzen.

12.1.2 Reflexion Jan Forlin

Vor der Studienarbeit habe ich nicht wirklich viel von Smart Trainer gehört. Die Kombination eines solchen Trainer mit der Anwendung um zu Hause einen Abschnitt der Etappe abzufahren hat mich interessiert. Bis auf das Engineering Projekt hatte ich noch nie solch ein Projekt umgesetzt. Ich war nie im Bereich der Software-Entwicklung tätig. Neu war die Front-End Entwicklung für Android. Um die App überhaupt zu entwickeln, habe ich gleichzeitig während der Arbeit das Modul 'Mobile GUI Engineering"besucht.

In der Analysephase haben wir uns erst mit den Technologien Bluetooth Smart und ANT+ auseinandergesetzt. Leider haben wir nur Dokumente gesammelt und gelesen aber keine Debug App erstellt. Die Arbeit war jedoch sehr wichtig, um überhaupt mit dem Trainer kommunizieren zu können. Auch die Einarbeitung in LaTex hat uns einige rote Köpfe beschert. Nach der Beseitigung vieler Probleme und Formatierung hat sich LaTex als sehr gute Software gezeigt. Während der Entwicklung der App mussten wir uns mit Kommunikationsproblemen zwischen Smart Trainer und Smartphone auseinandersetzen. Das Debugging gestaltete sich schwierig, da der Smart Trainer keine Fehlermeldung zurückgibt. Trotzdem ist uns zum Schluss eine funktionierende App entstanden auf die wir stolz sein können. Ich habe viel über die Entwicklung von Android Applikationen gelernt. In zukünftigen Projekten kann ich auf dieses Wissen zurückgreifen und Probleme schneller lösen.

Ich möchte mich bei meinem Partner Zarko für die Zusammenarbeit bedanken. Wir waren zwar nicht immer gleicher Meinung und haben meist länger an Entscheidungen diskutiert. Gegen Ende der Arbeit ist uns jedoch dies leichter gefallen. Weiters möchte ich mich bei Prof. Dr. Heinzmann und P. Eichler bedanken. Sie haben uns wöchentlich in einer Sitzung ihr Feedback gegeben und die nächsten Schritte besprochen.

12.2 Erklärungen

In diesem Kapitel sind die Eigenständigkeitserklärung für eprints.ch, Urheber- und Nutzungsrechte sowie Einverständniserklärung.

12.3 Projektplanung

Als Projektplanung dienten vor allem die Wochensitzungen mit den Betreuern und die dort entstandenen Sitzungsprotokolle. Ein grober Zeitplan wurde auch erstellt und grössenteils eingehalten. Diese Dokumente sind Teil des Archivs, welches abgegeben wurde.