



Logistisches Informationssystem für Taxis unter Wahrung der Privatsphäre

Dokumentation

Art der Arbeit: Bachelor-Thesis

Studiengang: Informatik
Autoren: Stephan Menzi, Hansjürg Jaggi
Betreuer: Dr. Reto König
Auftraggeber: EasyCab GmbH
Experten: Dr. Federico Flueckiger
Datum: 21.01.2016

Management Summary

Moderne Unternehmen im Personentransport möchten den Zustand der Fahrzeugflotte, wie die Positionen und Auslastung, die zurückgelegte Strecke oder den Fahrzeugzustand in Echtzeit erkennen, um in der Zentrale umgehend reagieren zu können ohne den Fahrer direkt kontaktieren zu müssen. Die Fahrer ihrerseits möchten aber ein gewisses Mass an Privatsphäre beweisbar gewährleistet haben.

Die Firma EasyCab in Bern, welche Transporte für Menschen mit Behinderungen ausführt, wünscht eine solche Hard- und Software-Lösung zum Tracking der Fahrzeugflotte. Dafür wurde die Berner Fachhochschule angefragt, im Rahmen eines Projekts einen entsprechenden Vorschlag zu unterbreiten. Da EasyCab explizit keine Smartphone App gewünscht hat, wurde der Prototyp eines Tracking-Geräts auf der Basis des Einplatinencomputers Raspberry Pi entwickelt. Für die Verbindung zum Internet wird ein Smartphone über ein USB-Kabel mit dem Gerät verbunden. Die Daten werden von einem Serverdienst in einer Datenbank abgespeichert und in einer Webapplikation dargestellt.

Dieser Raspberry Pi steckt in einem selbst entworfenen und mit einem 3D-Drucker hergestellten Gehäuse. Ein GPS-Empfänger, ein NFC-Leser sowie eine selbst entwickelte Platine mit fünf LED und einem Button sind an den Raspberry Pi angeschlossen. Das Gerät erkennt das Fahrzeug über einen darin angebrachten NFC-Chip. Wenn der Fahrer das Tracking aktivieren will, schliesst er das Gerät an den USB-Port des Fahrzeugs an, verbindet das Smartphone mit dem Raspberry Pi und authentifiziert sich mit einem NFC-Chip.

Die LED am Gerät zeigen an, ob eine Internetverbindung besteht, GPS-Daten empfangen werden und ob Fahrer und Fahrzeug authentifiziert sind. All diese Funktionen erledigt ein in Python programmierter Dienst. Mit dem Button lässt sich das Tracking ausschalten. Wenn der Fahrer ganz sicher gehen will dass er durch das Gerät nicht lokalisiert wird, kann er es einfach ausschalten.

Dem Server werden die MAC-Adresse des Smartphones und UUIDs der beiden NFC-Chips übermittelt und dieser generiert daraus eine Session-ID. Diese wiederum wird mit den Konfigurationsdaten an das Tracking-Gerät zurück gesendet. Das Tracking-Gerät sendet von da an Positionsdaten an den Server, wo sie in eine Datenbank geschrieben werden. Durch die Session-ID können die Daten mit dem Fahrer, dem Fahrzeug und dem Smartphone verknüpft werden.

Die Daten werden mit AES verschlüsselt, über das leichtgewichtige Protokoll MQTT an eine Server übertragen und in einer Datenbank gespeichert. Der Disponent sieht in einer Webapplikation die aktuellen Positionen der Fahrzeuge auf einer Karte. Für jedes Fahrzeug werden der Name des Fahrers, die Telefonnummer des Smartphones sowie die zuletzt gemeldete Position angezeigt. Aktive und inaktive Fahrzeuge können farblich unterschieden und auch gefiltert werden. Der Disponent kann Routen zwischen einem Start- und Zielort oder direkt vom Fahrzeug zu einem Zielort suchen.

Über eine Administrationsoberfläche werden die Stammdaten und Applikationskonfiguration verwaltet. Gesammelte Trackingdaten können als CSV-Datei exportiert werden, damit diese für Auswertungen genutzt werden können. Dies hilft der Personalabteilung, um festzustellen ob die gesetzlich vorgeschriebenen Ruhezeiten eingehalten werden.

Die ganze Lösung wurde mit offenen Technologien entwickelt und kann somit um weitere Funktionen und Sensoren erweitert werden.

Kein Teil des Projektes war der Betrieb und die Sicherheit des Servers, die Beschaffung der Smartphones oder die Einführung der Trackinglösung bei EasyCab.

Inhaltsverzeichnis

1	Einleitung	2
1.1	Abgrenzung	2
1.2	Marktübersicht	2
1.2.1	Traccar	3
1.2.2	OpenGTS	3
1.2.3	GEOLINK	3
1.2.4	TomTom Fleet Management	3
1.2.5	Fazit	3
2	Vorgehen	4
2.1	Vorarbeiten	4
2.2	Projekt	4
2.3	Arbeitstechnik	5
2.4	Materialbeschaffung	5
3	Zeitplan	6
3.1	Taskliste	6
3.2	Milestones	6
4	Anforderungen	8
4.1	Aktoren	8
4.1.1	Fahrer	8
4.1.2	Disponent	8
4.1.3	Maintainer	8
4.1.4	Revisor	8
4.2	Daten	8
4.3	Datenübertragung	9
4.4	Hardware	9
4.5	Webbasierte Kartenapplikation	9
4.6	Webbasierte Administrations-Oberfläche	9
4.7	Schutz der Privatsphäre	10
4.8	Offenheit von Technologie und Architektur	10
4.9	Optionen	10
4.9.1	Stressbutton	10
4.9.2	Fahrzeugdaten	10
4.9.3	Gehäuse	10
4.9.4	Konfiguration per USB Memory-Stick	10
5	Aktoren und Stories	11
5.1	Aktor: Fahrer	11
5.1.1	Story: Fahrer meldet sich mit NFC-Chip an und wird getrackt	11
5.1.2	Story: Fahrer hat keinen NFC-Chip dabei	12
5.1.3	Story: Tracking-Gerät ausschalten	13
5.1.4	Story: Gerät zurücksetzen	13
5.2	Aktor: Maintainer	14
5.2.1	Story: Einbau eines Tracking-Geräts in ein Fahrzeug	14
5.2.2	Story: Ein neues Tracking-Gerät vorbereiten	14
5.2.3	Story: Ein neues Fahrzeug hinzufügen	14
5.2.4	Story: Einen neuen Fahrer hinzufügen	15
5.2.5	Story: Ein neues Smartphone hinzufügen	15

5.2.6	Story: Pairing von Tracking-Gerät und Smartphone	16
5.2.7	Story: Konfiguration auf Tracking-Gerät anpassen	16
5.3	Aktor: Disponent	17
5.3.1	Story: Nur aktive Fahrzeuge anzeigen	17
5.3.2	Story: Nur inaktive Fahrzeuge anzeigen	17
5.3.3	Story: Zurückgelegte Strecke eines Fahrers anzeigen	18
5.3.4	Story: Ort auf Karte suchen	18
5.3.5	Story: Route berechnen	19
5.3.6	Story: Route von Fahrzeug als Startpunkt berechnen	19
5.4	Aktor: Revisor / Qualitätssicherung	20
5.4.1	Story: Daten auswerten	20
6	Technologien / Komponenten	21
6.1	Raspberry Pi	21
6.2	Sensoren	22
6.2.1	NFC Leser	22
6.2.2	GPS	22
6.3	Betriebssystem	22
6.4	Apache Webserver	22
6.5	Bluetooth	23
6.6	USB-Tethering	23
6.7	Datenbank	23
6.8	Python	24
6.9	Django	24
6.10	JavaScript / Google Maps API / jQuery	24
6.11	MQTT	25
6.12	Elektronische Bauteile	26
6.12.1	Elektronische Grundlagen	26
6.12.2	LED	27
6.12.3	Widerstände	27
6.13	CAN-Bus	28
7	Umsetzung	29
7.1	Tracking-Gerät	29
7.1.1	1. Prototyp	29
7.2	Tethering	30
7.3	LED und Button	30
7.3.1	Erster Entwurf	30
7.3.2	Zweiter Entwurf	31
7.3.3	Versuchsaufbau	32
7.3.4	Prototyp	33
7.4	Gehäuse	33
7.5	Server-Konfiguration	36
7.6	Web-Applikation	37
7.6.1	1. Prototyp	37
7.6.2	Erweiterung mit Karten-Such-Funktionalitäten	37
7.7	Software	38
7.7.1	Client / Tracking-Gerät	38
7.7.2	Server	39
7.7.3	Daten-Verschlüsselung	42
8	Tests	43
8.1	Tracking-Gerät	43
8.1.1	Maintainer installiert neues Tracking-Gerät in Fahrzeug (Story 5.2.1)	43
8.1.2	Fahrer meldet sich mit NFC-Chip an und wird getrackt (Story 5.1.1)	44
8.1.3	Fahrer hat keinen NFC-Chip dabei (Story 5.1.2)	45
8.1.4	Tracking-Gerät aus- und anschliessend einschalten (Story 5.1.3)	46
8.1.5	Gerät zurücksetzen (Story 5.1.4)	47

8.1.6	Pairing von Tracking-Gerät und Smartphone (Story 5.2.6)	48
8.1.7	Konfiguration des Tracking-Geräts anpassen (Story 5.2.7)	50
8.2	Google Maps Kartenapplikation	51
8.2.1	Nur aktive, inaktive oder alle Fahrzeuge anzeigen (Stories 5.3.1 und 5.3.2)	51
8.2.2	Zurückgelegte Strecke eines Fahrzeugs anzeigen (Story 5.3.3)	52
8.2.3	Ort auf Karte suchen (Story 5.3.4)	53
8.2.4	Route berechnen (Story 5.3.5)	54
8.2.5	Route von Fahrzeug als Startpunkt berechnen (Story 5.3.6)	55
8.3	Administrations-Oberfläche	56
8.3.1	Ein neues Fahrzeug hinzufügen, bearbeiten und löschen (Story 5.2.3)	56
8.3.2	Einen neuen Fahrer hinzufügen, bearbeiten und löschen (Story 5.2.4)	57
8.3.3	Ein neues Smartphone hinzufügen, bearbeiten und löschen (Story 5.2.5)	58
8.3.4	Positions-Daten exportieren (Story 5.4.1)	59
9	Handbuch	60
9.1	Tracking Gerät	60
9.1.1	Tracking beginnen	60
9.1.2	Tracking beenden	60
9.2	Karten-Applikation	61
9.2.1	Fahrzeugdetails	61
9.2.2	Zurückgelegte Route eines Fahrzeugs	62
9.2.3	Fahrzeugübersicht und Filter	62
9.2.4	Standort suchen	63
9.2.5	Route zwischen zwei Standorten anzeigen	63
9.2.6	Route von Fahrzeug zu einem Standort anzeigen	64
9.3	Administrations-Oberfläche	65
9.3.1	Applikations-Konfigurationen bearbeiten	65
9.3.2	Fahrer, Smartphone oder Fahrzeug hinzufügen	66
9.3.3	Fahrer, Smartphone oder Fahrzeug bearbeiten	66
9.3.4	Fahrer, Smartphone oder Fahrzeug entfernen	67
9.3.5	Positions-Daten exportieren	67
9.4	Installation	68
9.4.1	Tracking Gerät	68
9.4.2	Server	73
9.5	Konfiguration des Tracking-Geräts anpassen	74
10	Installation	75
10.1	Raspberry PI	75
10.1.1	Betriebssystem	75
10.1.2	Zugriff per SSH	75
10.1.3	Anpassen der primären Partition	75
10.1.4	Benötigte Software installieren	76
10.2	Server	77
10.2.1	Zeitzone anpassen	77
10.2.2	Python / PIP	77
10.2.3	Apache & MySQL	77
10.2.4	Mosquitto	78
10.2.5	Python Module	78
11	Fazit	80
11.1	Ergebnis	80
11.2	Zukunft	80
11.3	Persönliches Fazit	81
11.4	Dank	81
12	Arbeits-Journal	82
	Glossar	87

Quellenverzeichnis	88
Abbildungsverzeichnis	89
Selbständigkeitserklärung	90

1 Einleitung

Moderne Unternehmen im Personentransport möchten den Zustand der Fahrzeugflotte, wie die Positionen und Auslastung, die zurückgelegte Strecke oder den Fahrzeugzustand in Echtzeit erkennen, um in der Zentrale umgehend reagieren zu können ohne den Fahrer direkt kontaktieren zu müssen. Die Fahrer ihrerseits möchten aber ein gewisses Mass an Privatsphäre beweisbar gewährleistet haben.

Die Firma EasyCab in Bern organisiert Transporte für Menschen mit Behinderungen. Die Firma wünscht eine solche Hard- und Software-Lösung zum Tracking der Fahrzeugflotte. Dafür wurde die Berner Fachhochschule angefragt, im Rahmen eines Projekts einen entsprechenden Vorschlag zu unterbreiten.

In jedem Fahrzeug der Firma EasyCab wird ein Tracking-Gerät installiert, welches die momentane Position des Fahrzeugs an einen zentralen Datenspeicher sendet. Das Gerät soll neben der genaueren Lokalisierung der Fahrzeuge und Fahrer auch dazu verwendet werden können, die gesetzlich vorgeschriebenen Pausen der Fahrer zu kontrollieren.

Durch eine offene Architektur der Schnittstellen wird sichergestellt, dass nachträglich Erweiterungen wie zusätzliche Sensoren integriert werden und dass auch verschiedene Zielsysteme die Daten des Trackers auswerten können.

Damit die Privatsphäre der Fahrer sichergestellt werden kann, ist zu beachten, dass nur Daten ihrer Arbeits-Fahrten getrackt werden. Das Tracking muss also vom Fahrer deaktiviert werden können.

Das Projekt wird im Rahmen der Module *BTI7302 Projekt 2* und *BTI7321 Bachelor Thesis* durchgeführt.

1.1 Abgrenzung

Mit der Applikation lässt sich ermitteln, wann welches Fahrzeug von welchem Fahrer verwendet wird und welche Strecken zurückgelegt werden. Die Analyse dieser Daten - zum Beispiel zur Überprüfung der gesetzlich vorgeschriebenen Pausen - ist nicht Ziel dieses Projektes.

Des weiteren nicht Teil des Projektes sind:

- Die Einrichtung und Verwaltung des Servers inklusive Zugangsberechtigung.
- Die Verwaltung der Datenbank inklusive Datensicherung und Zugangsberechtigung.
- Die Beschaffung und Bereitstellung der Smartphones.
- Die Wartung der Geräte.
- Einrichtung und Gewährleistung einer VPN Verbindung.
- Die Problematik der Netz- und GPS-Qualität.
- Schutz vor Ortung mit anderen Applikationen und Systemen, wie beispielsweise die Ortung über Mobilfunksignale.

Im Rahmen des Projekts werden 5 Prototypen ausgeliefert. Weitere Geräte muss EasyCab selber beschaffen und gemäss Anleitung zusammenbauen.

1.2 Marktübersicht

Es gibt diverse Tracking-Lösungen auf OpenSource-Basis und von kommerziellen Anbietern, aber es konnte keine Lösung gefunden werden, die alle Anforderungen erfüllte.

1.2.1 Traccar

Traccar ist eine OpenSource Tracking-Lösung für Fahrzeuge, bestehend aus einer Server- und Client-Komponente. Die Server-Komponente wurde vorwiegend in Java umgesetzt, für die Client-Komponente wurden iOS- und Android-Apps entwickelt.

Um die Lösung für dieses Projekt zu erweitern, müssten die Client-Applikationen umgeschrieben werden, damit diese keine Smartphone-App verwenden.

1.2.2 OpenGTS

OpenGTS ist ebenfalls eine OpenSource-Lösung um Fahrzeuge zu tracken und bietet zudem die Unterstützung einer Vielzahl von verschiedenen Tracking-Geräten. Die Applikation ist ebenfalls vorwiegend in Java umgesetzt.

Es müsste ebenfalls die Client-Komponente angepasst und umgeschrieben werden, damit diese Lösung für das Projekt verwendet werden kann.

1.2.3 GEOLINK

Bei GEOLINK handelt es sich um einen kommerziellen Dienst, mit welchem Fahrzeuge getrackt werden können. Der Dienst kann über ein Abonnement monatlich bezahlt werden. Je nach Anzahl der Fahrzeuge variiert der zu bezahlende Betrag.

Mit dem GEOLINK Tracker wird zusätzlich ein Tracking-Gerät angeboten, welches mit offener Hardware (auf Basis eines Arduino-Rechners) gebaut wird.

1.2.4 TomTom Fleet Management

TomTom Fleet Management ist ein rein kommerzielles Angebot. Hiermit können TomTom-Navigationssysteme verwendet werden, um die Fahrzeuge zu tracken.

1.2.5 Fazit

Auf Lösungen von kommerziellen Anbietern wurde aus folgenden Gründen verzichtet:

- Die verwendeten Daten müssen im Besitz des Kunden bleiben und nicht auf externen Servern abgelegt werden.
- Der Kunde soll unabhängig von zusätzlichen Anbietern, ausser seiner eigenen IT-Abteilung, seinem Telekom-Anbieter und den Hardware-Lieferanten, sein.
- Anpassungen und Erweiterung sind von den Angeboten des Lieferanten abhängig.

Auf die Verwendung einer bestehenden OpenSource-Lösungen als Grundlage des Projekts wurde verzichtet, da eine Neuentwicklung vom Aufwand her vergleichbar ist und eine grössere Flexibilität für Erweiterungen bietet. Des weiteren wird eine Lösung mit einer einheitlichen Technologie - in unserem Fall Python - entwickelt.

2 Vorgehen

2.1 Vorarbeiten

Im Rahmen des Moduls BTI7302 Projekt 2 im Frühlingssemester 2015 wurden folgende Vorarbeiten ausgeführt:

- Planung des Projekt.
- Anforderungen.
- Evaluation der Hardware.
 - Computer
 - Sensoren
 - Kommunikation
- Evaluation des Betriebssystems und der Software.
- Übertragung von Daten mit MQTT.
- Aufbau eines Prototypen.

2.2 Projekt

Mit den Erkenntnissen aus den Vorarbeiten wird das Projekt angepasst. Einige der im Rahmen der ersten Planungen evaluierten Funktionen wurden als optional zurückgestuft.

In dieser zweiten Phase, im Rahmen der Bachelor-Thesis, wird der Prototyp weiter ausgearbeitet und die Website mit der Administrations-Oberfläche implementiert. Die wichtigsten Projektarbeiten sind:

- Ausarbeitung der Use Cases / Storyboards.
- Button zum Deaktivieren / manuellen Aktivieren des Geräts.
- LED zum Anzeigen der verschiedenen Status des Gerätes.
- Administrations-Oberfläche zum Verwalten der Daten.
- Verschlüsselung der Datenübertragung.
- Handbuch zur Installation und Handhabung des Tracking-Geräts und der Web-Komponenten erstellen.
- Entwerfen und Drucken eines Gehäuses
- Testen.

2.3 Arbeitstechnik

Das Projekt-Team trifft sich jede Woche am Freitag zur Besprechung der Aktivitäten der letzten Woche, zur gemeinsamen Planung des weiteren Vorgehens und um sich den offenen Tasks zuzuwenden. Weitere Treffen finden nach Bedarf statt.

Tasks werden in einer "Google Drive" Tabellenkalkulation festgehalten.

	A	B	C	D	E
1	ID	Task	Priority	Responsible	Status
2	1	Aufbau erster Prototyp Tracking-Gerät	medium	Hanjo&Stephan	done
3	2	Buttons und Switches	medium	Hanjo&Stephan	in progress
4	2.1	Schaltplan entwerfen	high	Stephan	done
5	2.2	LED und Button Prototyp	medium	Stephan	to do
6	2.3	Hardware für Buttons und LED	high	Stephan	to do
7	2.4	Python Module für Buttons und LED	medium	Stephan	to do
8	3	Kommunikation zwischen Geräten	medium	Hanjo&Stephan	in progress
9	3.1	Tethering zwischen Smartphone und Tracking-Gerät	medium	Hanjo	in progress
10	3.2	Sicherheit und Verschlüsselung	medium	Hanjo&Stephan	to do
11	4	Echtzeit-Darstellung der Daten auf Karte	medium	Hanjo	done
12	5	Administrationsoberfläche	medium	Hanjo	in progress
13	5.1	Installation Django mit Basis-Administrations-Oberfläche	medium	Hanjo	done
14	5.2	Applikations-Einstellungen per Django verwalten	medium	Hanjo	in progress
15	5.3	Konfiguration der Administrations-Oberfläche und Layout	medium	Hanjo	to do
16	6	Meetings	medium	Hanjo&Stephan	in progress
17	6.1	Treffen mit Experten	high	Stephan	to do
18	6.2	Treffen mit EasyCab	high	Hanjo&Stephan	to do
19	7	Gehäuse	low	Hanjo	to do
20	8	Tests	low	Hanjo&Stephan	to do
21	9	Präsentation	low	Hanjo&Stephan	to do
22	10	Dokumentation	medium	Hanjo&Stephan	in progress
23	10.1	Abschnitt MQTT zusammenfassen	high	Stephan	done
24	10.2	Storyboards / Use-Cases erarbeiten	high	Hanjo&Stephan	done
25	10.3	Marktübersicht erstellen	high	Hanjo&Stephan	done
26	10.4	Benutzerhandbuch	low	Hanjo&Stephan	to do

Abbildung 2.1: Task-Liste

Jede zweite Woche trifft sich das Team ausserdem mit dem Betreuer des Projekts, um den momentanen Stand zu aktualisieren und allfällige Fragen zu klären.

Der Quelltext wird in einem Git-Repository verwaltet.

Die Dokumentation wird auf ShareL^AT_EX geschrieben. ShareL^AT_EX ist ein Online L^AT_EX Editor zur kooperativen Dokumentbearbeitung. Die kostenlose Version ermöglicht das gleichzeitige Bearbeiten eines Dokuments von zwei Personen.

Das Testing der Applikation und des Geräts findet laufend während der Entwicklung statt. Dem Kunden wird Ende Oktober 2015 eine Serie mit Testgeräten ausgeliefert.

Diese Testgeräte werden nur die Grundfunktionalitäten beinhalten, ermöglicht aber dem Kunden, Feedback zu geben und dabei zu helfen, ein Gerät zu schaffen, welches auch seinen Wünschen entspricht.

2.4 Materialbeschaffung

Zum Aufbau der Applikation werden grundsätzlich Komponenten aus dem Inventar der Berner Fachhochschule verwendet. Zusätzliche Komponenten werden vom Projekt-Team bestellt und nachträglich über den Betreuer zurückerstattet.

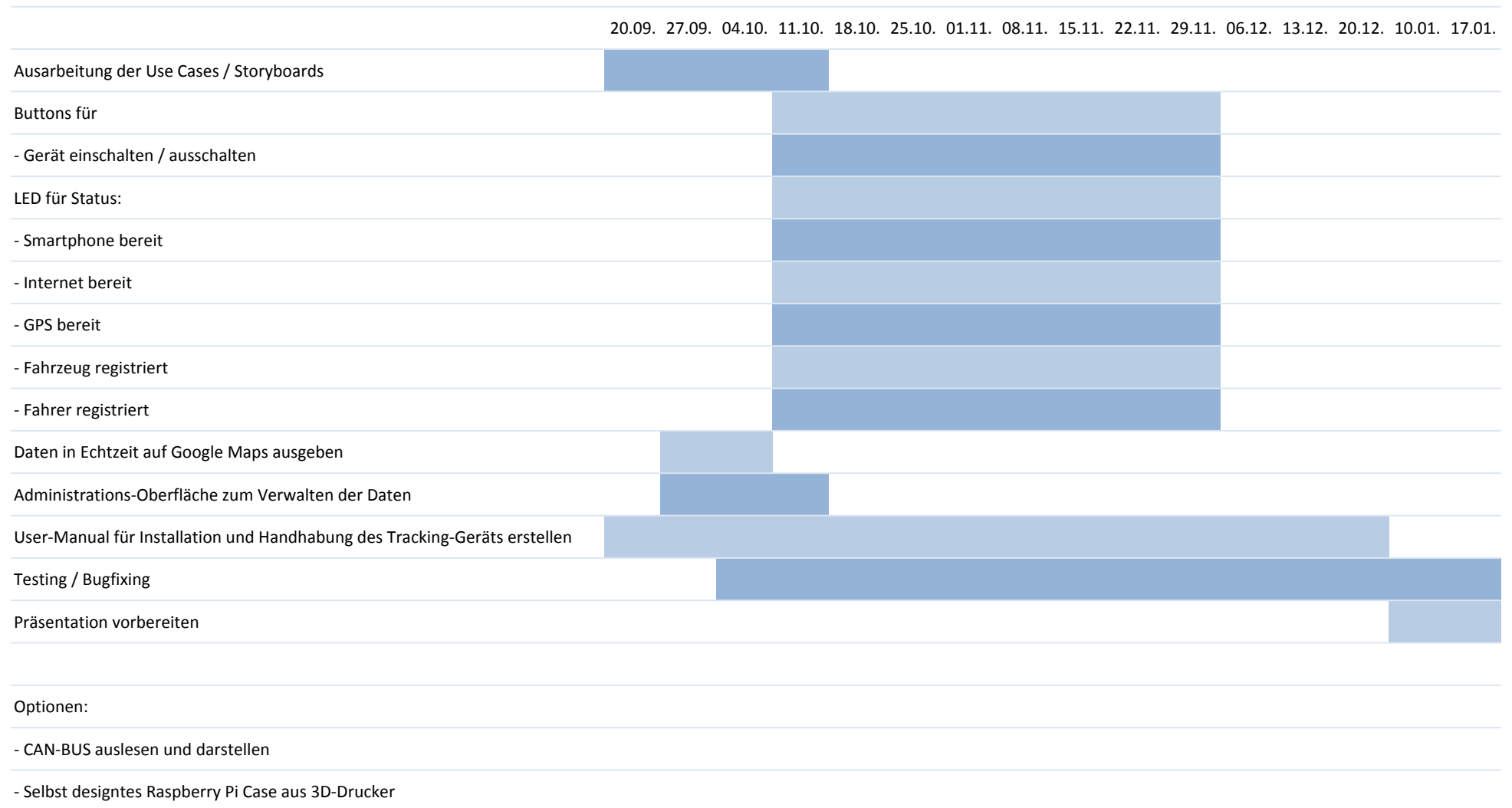
3 Zeitplan

3.1 Taskliste

- Ausarbeitung der Use Cases / Storyboards
- Button für
 - Gerät deaktivieren / manuell aktivieren
- LED für Status:
 - Smartphone verbunden
 - Internet bereit
 - GPS bereit
 - Fahrzeug registriert
 - Fahrer registriert
- Daten in Echtzeit auf Google Maps ausgeben.
- Administrations-Oberfläche zum Verwalten der Daten.
- Verschlüsselung der Datenübertragung.
- User-Manual für Installation und Handhabung des Tracking-Geräts erstellen.
- Optionen:
 - CAN-Bus auslesen und darstellen
 - Selbst designtes Raspberry Pi Case aus 3D-Drucker
 - Stress-Button für Fahrer.

3.2 Milestones

Milestones der Vorarbeiten	
30.04.2015	Finden der optimalen Hard- und Software
10.06.2015	Erster lauffähiger Prototyp
10.06.2015	Daten werden in Datenbank gespeichert
Milestones der Bachelor Thesis	
11.10.2015	Storyboards, Anforderungen und Einleitung bereit.
31.10.2015	Testgeräte (ohne LED, Buttons und CAN-Bus Funktionalitäten) an Kunden übergeben.
29.11.2015	Hardware bereit mit LEDs und Buttons und ev. CAN-Bus.
23.12.2015	Produkt fertiggestellt bis auf Dokumentation, Präsentation und Bugfixing
21.01.2016	Abgabe Bachelorarbeit, Präsentation vorbereitet.



4 Anforderungen

4.1 Akteure

Die Anforderungen an die Applikation unterscheiden sich je nach Akteur, welcher sie verwenden will. Es werden folgende Akteure berücksichtigt:

4.1.1 Fahrer

Der Fahrer steuert das Fahrzeug und bestimmt, wann das Tracking-Gerät aktiviert wird und Daten senden soll.

Er interagiert mit der Applikation über das Tracking-Gerät welches im Fahrzeug installiert wurde, ein Firmen-Smartphone sowie einen ihm zugewiesenen NFC-Chip.

4.1.2 Disponent

Der Disponent will eine Übersicht über die Fahrzeugflotte und die Positionen der einzelnen Fahrzeuge.

Er greift auf die Applikation über die webbasierte Kartenapplikation zu.

4.1.3 Maintainer

Der Maintainer (oder auch Administrator) verwaltet die Fahrzeuge, Fahrer, verwendeten Smartphones und die Applikations-Konfiguration in der Datenbank, installiert NFC-Tags in den Fahrzeugen und verteilt NFC-Tags an die Fahrer.

Er greift auf die Applikation über die webbasierte Administrations-Oberfläche zu.

4.1.4 Revisor

Der Revisor kann die gesammelten Daten verwenden, um fehlende Pausenzeiten oder sonstige Unstimmigkeiten aufzudecken.

Er greift auf die Applikation über die webbasierte Administrations-Oberfläche zu.

4.2 Daten

Bei den Daten wird zwischen Konfigurations-, Authentifizierungs-, Lokalisierungs- und Stammdaten unterschieden, wobei die Konfigurationsdaten verschiedene Applikations-Konfigurationen enthalten, die Authentifizierungsdaten die Daten der Fahrer, Taxis und Smartphones beinhalten und die Lokalisierungsdaten den eigentlichen Koordinaten entsprechen. Als Stammdaten gelten die registrierten Fahrer, Fahrzeuge und Smartphones, welche über eine webbasierte Administrations-Oberfläche gepflegt werden können.

Sämtliche Daten werden in einer Datenbank abgelegt und mit Django verwaltet.

Bevor die Lokalisierungsdaten übermittelt werden, sendet das Tracking-Gerät die momentanen Authentifizierungsdaten, damit daraus auf dem Server eine Sitzung generiert und persistent in der Datenbank abgelegt werden kann.

Das Gerät erhält daraufhin die eindeutige ID der Sitzung sowie die Konfigurationsdaten.

Sobald die Sitzung aufgebaut worden ist, werden die Positionsdaten in periodischen Abständen gesendet. Der Aktualisierungs-Intervall ist in den Konfigurationsdaten hinterlegt.

4.3 Datenübertragung

Die Übertragung der Daten soll eingeschränkte Bandbreiten, eine hohe Latenz oder Netzausfälle berücksichtigen. Die Daten werden in einem universellen Datenformat übermittelt, damit sie später ohne Unterstützung des Projekt-Teams verwendet und erweitert werden können.

4.4 Hardware

EasyCab hat explizit keine Smartphone App gewünscht, damit der Fahrer seine Privatsphäre verifizieren kann und um sicher zu stellen, dass ein Smartphone unabhängig vom Fahrer genutzt werden kann. Zu Beginn des Projekts war zudem nicht bekannt, dass es sich bei den Smartphones um Firmen-Geräte handelt.

Deshalb wird eine Lösung mit einem Einplatinencomputer wie Raspberry Pi entwickelt. Der Einplatinencomputer soll austauschbar sein.

Das Tracking-Gerät stellt über das Smartphone des Fahrers per Tethering eine Verbindung zum Internet her. Die Stromversorgung wird über die Fahrzeug-Batterie gewährleistet.

Das Smartphone ist ein Firmengerät und ist weder einem bestimmten Fahrer noch Fahrzeug zugeordnet. Es muss eine Möglichkeit bestehen, den Fahrer und das Fahrzeug zu erkennen sowie das Tracking-Gerät in einem anderen Fahrzeug zu platzieren. Im Verlauf des Projekts stellte sich heraus, dass die Firma ausschliesslich iPhones verwenden wird.

Der Fahrer kann sich am Tracking-Gerät per NFC-Chip authentifizieren. Sollte der Fahrer keinen NFC-Chip dabei haben, übernimmt ein Disponent die Zuweisung in der Taxi-Zentrale über die Web-Oberfläche.

Die Fahrzeuge werden mit einem statischen NFC-Chip ausgestattet, damit Tracking-Geräte einfach ausgetauscht werden können und keine Abhängigkeit zum Fahrzeug haben.

Das Gerät soll dem Benutzer zeigen, ob es mit dem Smartphone verbunden ist, Verbindung zum Internet hat, GPS-Signale empfängt und ob die NFC-Chips für Fahrzeug und Fahrer erkannt worden sind. Diese Anzeige soll mit einfachen Elektronikbauteilen herstellbar sein.

4.5 Webbasierte Kartenapplikation

Die in der Applikation gesammelten Daten werden über eine Webapplikation auf einer Karte abgebildet. Die Applikation bietet die Möglichkeit, sowohl aktive Fahrer mit der momentanen Position als auch inaktive Fahrer mit der zuletzt gemeldeten Position anzuzeigen. Ausserdem kann für jeden Fahrer der zurückgelegte Pfad innerhalb einer Zeitspanne verfolgt werden. Funktionalitäten zur Suche und Routen-Planung über Google Maps sind ebenfalls verfügbar.

4.6 Webbasierte Administrations-Oberfläche

Die Administrations-Oberfläche erlaubt es, die Stammdaten (Fahrer, Smartphones, Fahrzeuge) in der Datenbank zu verwalten.

Die Messdaten können exportiert werden. Beim Export wird eine Filteroption angeboten, damit die Daten ausgewählter Fahrer oder Fahrzeuge für einen bestimmten Zeitraum exportiert werden können.

Zusätzlich kann über die Applikation gesteuert werden, wie lange die Daten in der Datenbank gespeichert werden sollen und in welchem Abstand die Taxi-Koordinaten gesendet werden.

4.7 Schutz der Privatsphäre

Die Daten dürfen nur für die Disponenten und die Geschäftsleitung sichtbar sein. Die Übertragung muss verschlüsselt erfolgen. Auch der Datenschutz der betroffenen Fahrer muss gewährleistet sein. Diese müssen jederzeit die Ortung durch unser System deaktivieren können. Die Ortung mit anderen Systemen als unsere Applikationen und Geräte liegt nicht in unserer Verantwortung und ist nicht Teil dieser Arbeit.

4.8 Offenheit von Technologie und Architektur

Die verwendeten Technologien und Architekturen sollen offen und nachvollziehbar sein. Der Maintainer soll die Möglichkeit haben, weitere Geräte zu bauen, und die Applikation nachträglich mit neuen Funktionen und Sensoren zu erweitern.

Dafür wird im Benutzerhandbuch eine Anleitung zum Bau und zur Konfiguration von Tracking-Geräten enthalten sein.

4.9 Optionen

Folgende Optionen wurden als Wünsche von EasyCab eingebracht. Diese müssen aber in diesem Projekt nicht zwingend umgesetzt werden.

4.9.1 Stressbutton

Ein sogenannter Stress-Buttons, über welchen der Fahrer Probleme als Sprachnachricht aufzeichnen kann. Über eine Smartphone- oder Web-Applikation können die aufgezeichneten Nachrichten administriert und je nachdem an eine zentrale Stelle gesendet werden, welche die Angaben auswertet.

Verschiedene Ansätze und Probleme für diese Funktion wurden im Verlauf der Vorarbeiten evaluiert. Die Funktion wird aber bis auf weiteres zurückgestellt.

4.9.2 Fahrzeugdaten

Die Firma EasyCab hat den Wunsch geäußert, dass weitere Angaben des Fahrzeuges ausgelesen werden können:

- Kilometerstand um Unterhalt und Revisionen planen zu können.
- Tankfüllung und Treibstoffverbrauch.

Auch dies wurde in den Vorarbeiten angeschaut und wäre über einen CAN-Bus Anschluss technisch möglich. Aber auch diese Funktion wird bis auf weiteres zurückgestellt.

4.9.3 Gehäuse

Für die einfachere Handhabung des Tracking-Geräts wird als optionale Anforderung eingeplant, ein Plastik-Gehäuse mit Hilfe eines 3D-Druckers zu erstellen.

4.9.4 Konfiguration per USB Memory-Stick

Damit die Einstellungen des Tracking-Geräts ohne Zugriff auf die Kommando-Zeile angepasst werden können, ist die zusätzliche Anforderung aufgetaucht, dass im Tracking-Gerät ein USB Memory-Stick angeschlossen werden kann, auf welchem sich eine Konfigurationsdatei befindet. Diese soll automatisch auf dem Tracking-Gerät übernommen werden.

5 Aktoren und Stories

Die Anwendungsfälle der Applikation unterscheiden sich nach Akteur.

5.1 Akteur: Fahrer

Der Fahrer interagiert mit der Applikation über das Tracking-Gerät, das Smartphone und seinem zugewiesenen NFC-Chip.

5.1.1 Story: Fahrer meldet sich mit NFC-Chip an und wird getrackt

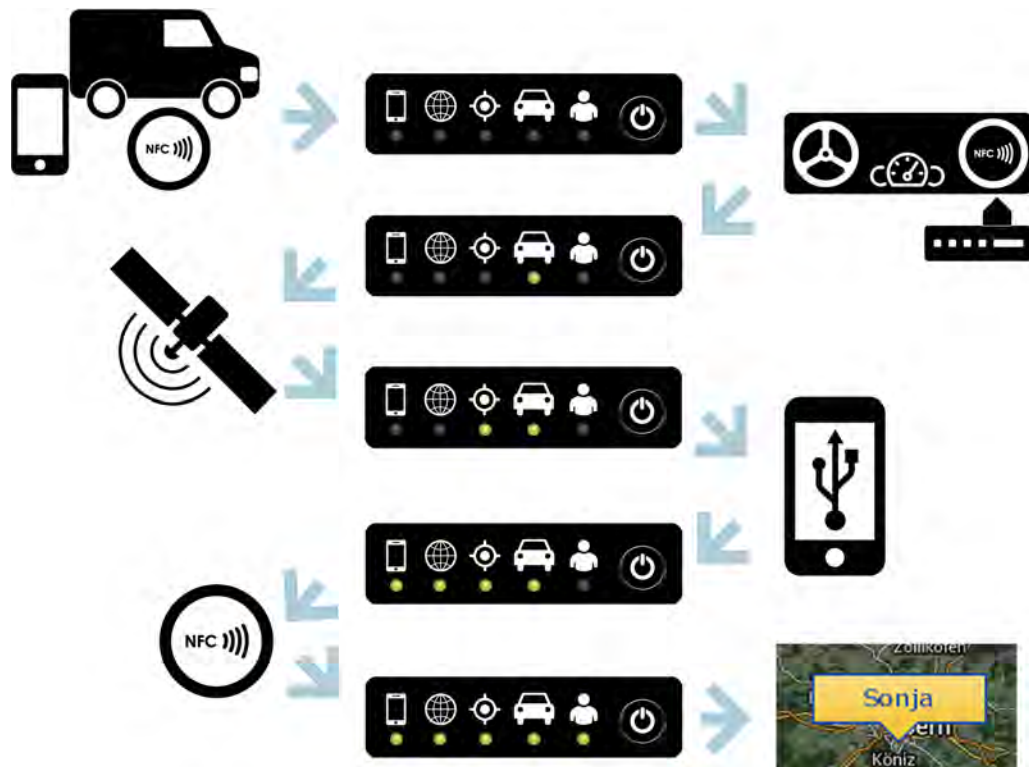


Abbildung 5.1: Story: Fahrer meldet sich mit NFC-Chip an und wird getrackt

1. Fahrer steigt in das Fahrzeug ein. Er hat das Smartphone und den ihm zugewiesenen NFC-Chip dabei.
2. Das Tracking-Gerät erkennt den im Auto installierten NFC-Chip.
3. Die LED "Fahrzeug" leuchtet.
4. Das Tracking-Gerät erhält GPS-Daten und die LED "GPS" leuchtet.
5. Das Smartphone wird per USB-Kabel mit dem Tracking-Gerät verbunden und die LED "Smartphone" leuchtet.

6. Das Tracking-Gerät erhält eine Verbindung zum Internet und die LED "Internet" leuchtet. Falls nicht, muss auf dem Smartphone sichergestellt werden, dass eine Verbindung zum Internet besteht und das USB-Tethering auf dem Smartphone aktiviert ist.
7. Der Fahrer meldet sich mit seinem NFC-Chip an.
8. Die LED "Fahrer" leuchtet.
9. Auf dem Bildschirm des Disponenten wird das Fahrzeug als aktiv angezeigt. Die Position sowie der Name des Fahrzeuges und des Fahrers werden ausgegeben.

5.1.2 Story: Fahrer hat keinen NFC-Chip dabei

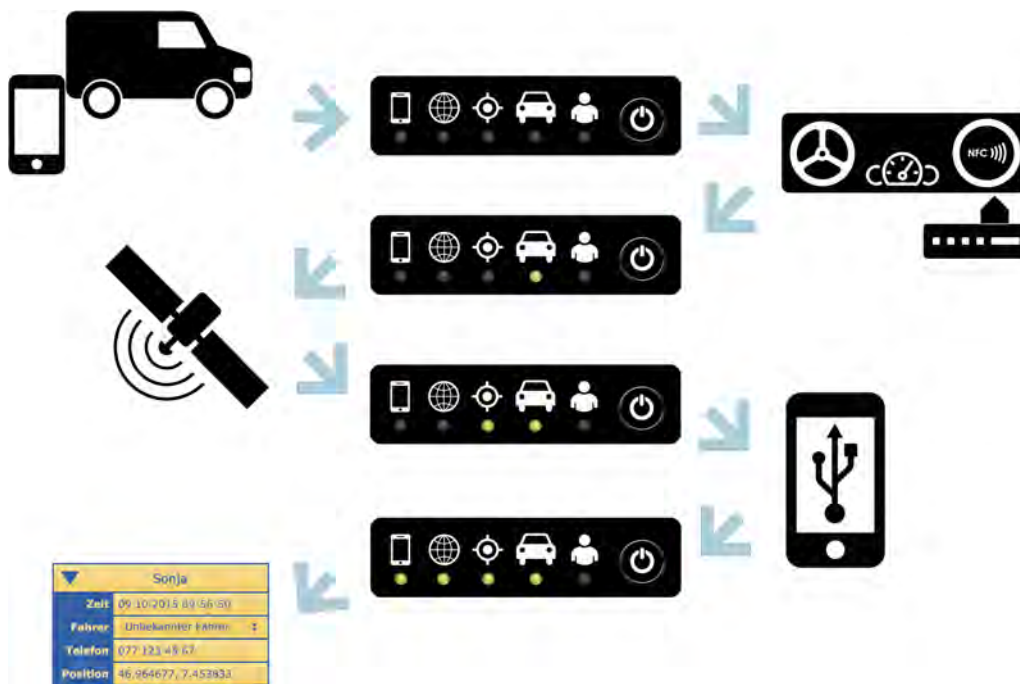


Abbildung 5.2: Story: Fahrer hat keinen NFC-Chip dabei

1. Fahrer steigt in das Fahrzeug ein. Er hat das Smartphone dabei, aber keinen NFC-Chip.
2. Das Tracking-Gerät erkennt den im Auto installierten NFC-Chip.
3. Die LED "Fahrzeug" leuchtet.
4. Das Tracking-Gerät erhält GPS-Daten und die LED "GPS" leuchtet.
5. Das Smartphone wird per USB-Kabel mit dem Tracking-Gerät verbunden und die LED "Smartphone" leuchtet.
6. Das Tracking-Gerät erhält eine Verbindung zum Internet und die LED "Internet" leuchtet. Falls nicht, muss auf dem Smartphone sichergestellt werden, dass eine Verbindung zum Internet besteht und das USB-Tethering auf dem Smartphone aktiviert ist.
7. Auf dem Bildschirm des Disponenten wird das Fahrzeug als aktiv angezeigt. Die Position und der Name des Fahrzeuges sowie die Telefon-Nummer des verwendeten Smartphones wird angezeigt. So kann der Disponent den Fahrer kontaktieren, identifizieren und somit den Fahrer dem Fahrzeug zuweisen.

5.1.3 Story: Tracking-Gerät ausschalten



Abbildung 5.3: Story: Fahrer will nicht getrackt werden

Sobald die Verbindung zwischen Smartphone und Tracking-Gerät getrennt wird - sei dies durch Trennen des Smartphones vom Gerät oder indem der Fahrer sich mit dem Smartphone vom Tracking-Gerät entfernt - wird das Tracking deaktiviert. Zusätzlich kann das Tracking-Gerät auch per Knopfdruck oder durch Trennen der Stromversorgung ausgeschaltet werden.

Nachdem das Tracking-Gerät wieder eingeschaltet wird oder eine Verbindung zum Strom-, beziehungsweise Daten-netz erhält, versucht es, sich mit der letzten bekannten Konfiguration zu reaktivieren.

1. Fahrer meldet sich vom Tracking-Gerät ab, indem er entweder den Button auf dem Gerät drückt oder die Verbindung zum Smartphone trennt.
2. Die LED auf dem Tracking-Gerät leuchten nicht mehr.
3. Auf dem Bildschirm des Disponenten wird das Fahrzeug als inaktiv angezeigt. Die letzte Position wird angezeigt.

5.1.4 Story: Gerät zurücksetzen

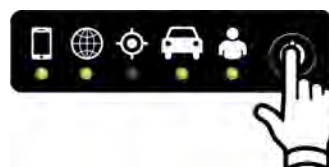


Abbildung 5.4: Story: Gerät zurücksetzen

1. Das Ein- und Ausschalten des Geräts durch zweimaliges Drücken des Buttons kann vor allem sinnvoll sein, wenn es Probleme mit der Erkennung der GPS-Position gibt.

5.2 Akteur: Maintainer

Der Maintainer ist als Administrator der Applikation zu verstehen und sorgt dafür, dass die Geräte korrekt aufgesetzt und die Fahrzeuge entsprechend ausgerüstet sind. Ausserdem erfasst der Maintainer zusätzliche Fahrer und Fahrzeuge in der Datenbank und weist die entsprechenden NFC-Chips zu.

5.2.1 Story: Einbau eines Tracking-Geräts in ein Fahrzeug

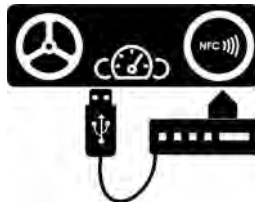


Abbildung 5.5: Story: Einbau eines Tracking-Geräts in ein Fahrzeug

1. Das Tracking-Gerät wird im Auto am Armaturenbrett über dem NFC-Chip des Fahrzeuges installiert.
2. Das Gerät wird an den USB-Port des Fahrzeuges angeschlossen.

5.2.2 Story: Ein neues Tracking-Gerät vorbereiten



Abbildung 5.6: Story: Ein neues Tracking-Gerät vorbereiten

1. Image-Datei auf Micro-SD Karte schreiben
2. Hardware-Komponenten verbinden

5.2.3 Story: Ein neues Fahrzeug hinzufügen



Abbildung 5.7: Story: Ein neues Fahrzeug hinzufügen

1. Es wird ein neuer NFC-Chip (Sticker) benötigt.
2. In Administrations-Oberfläche ein neues Fahrzeug hinzufügen. Hier muss der Name des Fahrzeugs sowie die UUID des NFC-Chips eingegeben werden.

3. Den NFC-Chip im Fahrzeug befestigen.

5.2.4 Story: Einen neuen Fahrer hinzufügen

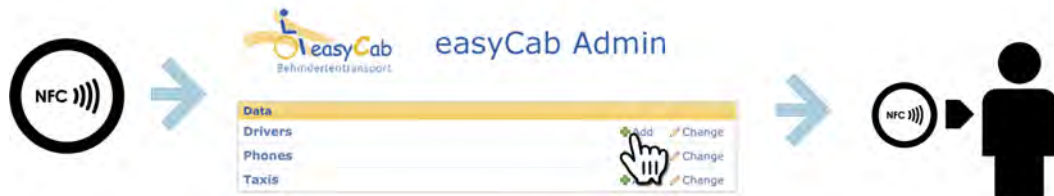


Abbildung 5.8: Story: Einen neuen Fahrer hinzufügen

1. Es wird ein neuer NFC-Chip benötigt.
2. In Administrations-Oberfläche einen neuen Fahrer hinzufügen. Hier können die persönlichen Daten des Fahrers sowie die UUID des NFC-Chips eingegeben werden.
3. Der NFC-Chip kann dem Fahrer übergeben werden.

5.2.5 Story: Ein neues Smartphone hinzufügen

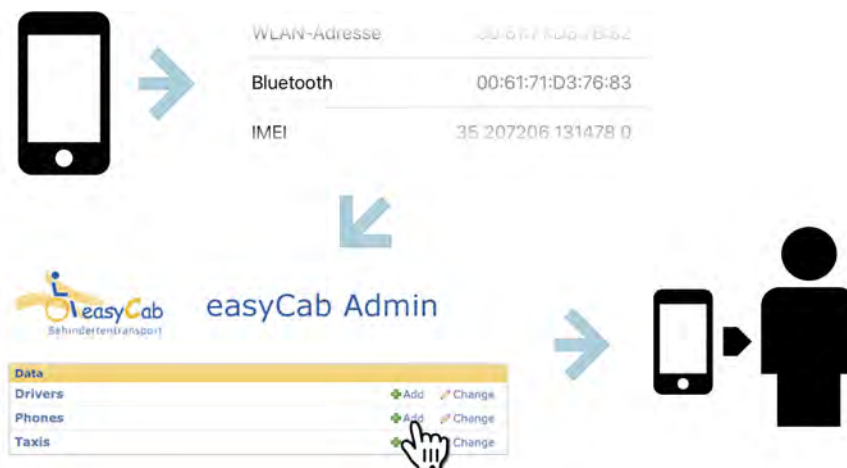


Abbildung 5.9: Story: Ein neues Smartphone hinzufügen

1. Es wird ein neues Smartphone benötigt.
2. In den Einstellungen des Smartphones muss die MAC-Adresse ausgelesen werden.
3. Über die Administrations-Oberfläche kann ein neues Smartphone hinzugefügt werden. Die MAC-Adresse ist zwingend notwendig, damit das Gerät verwendet werden kann. Um später die Telefonnummer für den Disponenten anzuzeigen, kann die Telefonnummer des Geräts hinterlegt werden. Optional kann dem Smartphone auch ein bezeichnender Name zugewiesen werden.
4. Das Smartphone kann nun verwendet werden.

5.2.6 Story: Pairing von Tracking-Gerät und Smartphone



Abbildung 5.10: Story: Pairing von Tracking-Gerät und Smartphone

1. Das Smartphone per USB-Kabel mit dem Tracking-Gerät verbinden. Das Tracking-Gerät muss am Strom angeschlossen sein. USB-Tethering auf dem Smartphone ist aktiviert.
2. Das Smartphone ist mit dem Gerät verbunden und die LED für Internet leuchtet.

5.2.7 Story: Konfiguration auf Tracking-Gerät anpassen

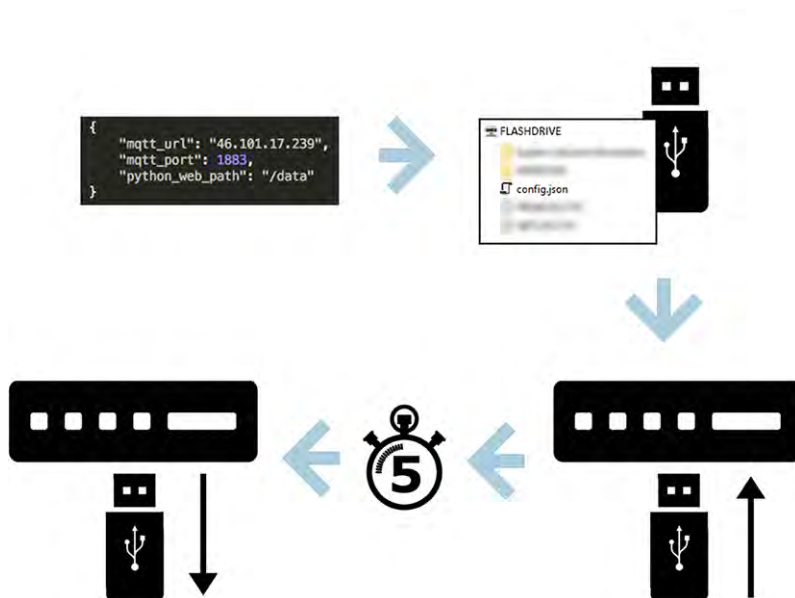


Abbildung 5.11: Story: Konfiguration auf Tracking-Gerät anpassen

1. Die Konfiguration wird gemäss dem Benutzerhandbuch erstellt und unter dem Dateinamen `config.json` auf einem USB Memory-Stick im Hauptverzeichnis abgelegt.
2. Der Memory-Stick wird an einen freien USB-Port des Tracking-Geräts angeschlossen.
3. Nach 5 Sekunden kann der Memory-Stick entfernt werden und die Konfiguration ist entsprechend angepasst.

5.3 Akteur: Disponent

Der Disponent arbeitet in der Taxi-Zentrale und kann die Taxis über eine Weboberfläche auf einer Karte anzeigen und verfolgen.

5.3.1 Story: Nur aktive Fahrzeuge anzeigen

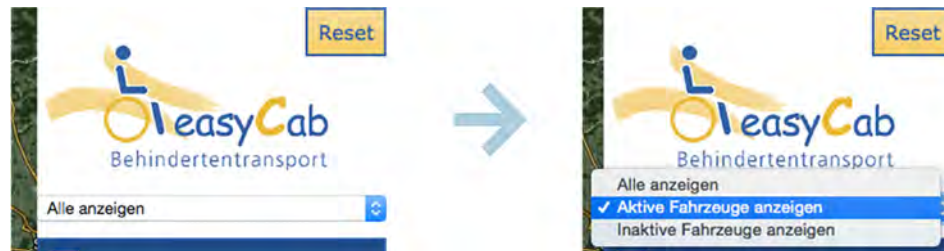


Abbildung 5.12: Story: Nur aktive Fahrzeuge anzeigen

1. In der Webapp "Aktive Fahrzeuge anzeigen" auswählen.
2. Die inaktiven Fahrzeuge werden ausgeblendet

5.3.2 Story: Nur inaktive Fahrzeuge anzeigen

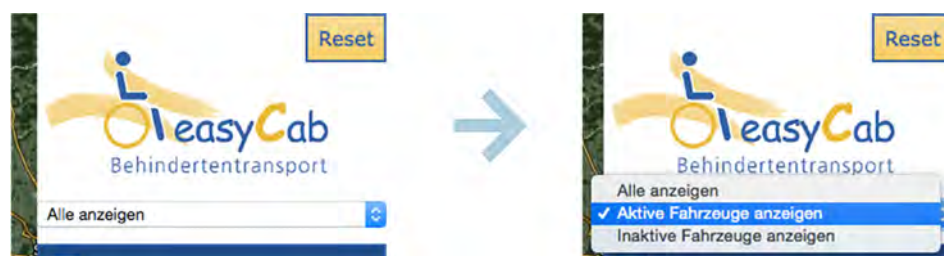


Abbildung 5.13: Story: Nur inaktive Fahrzeuge anzeigen

1. In der Webapp "Inaktive Fahrzeuge anzeigen" auswählen.
2. Die aktiven Fahrzeuge werden ausgeblendet.

5.3.3 Story: Zurückgelegte Strecke eines Fahrers anzeigen

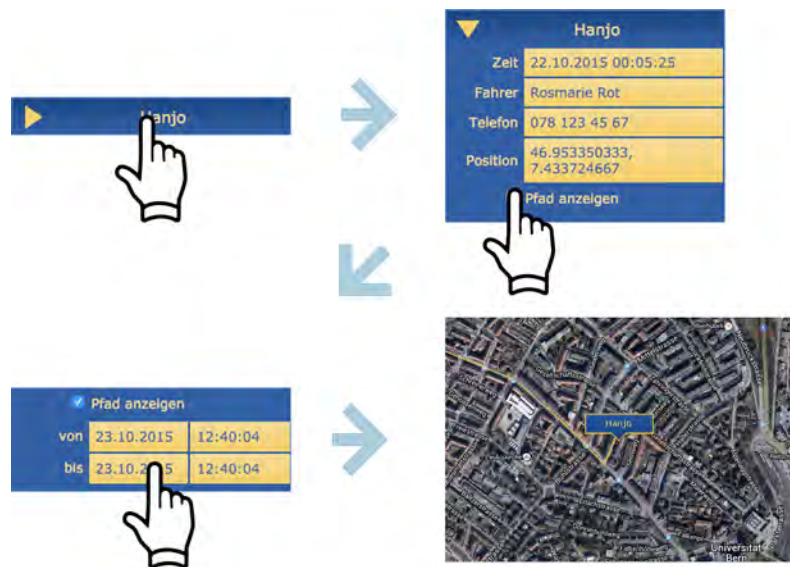


Abbildung 5.14: Story: Zurückgelegte Strecke eines Fahrzeugs anzeigen

1. In der Webapp das Fahrzeug auswählen, von welchem die Strecke nachverfolgt werden soll.
2. Die Checkbox "Pfad anzeigen" aktivieren.
3. Die Start- und Endzeit des Pfads kann über die jeweiligen Formularfelder ausgewählt werden.
4. Der Pfad erscheint auf der Karte.

5.3.4 Story: Ort auf Karte suchen



Abbildung 5.15: Story: Ort auf der Karte suchen

1. In der Webapp den gesuchten Ort unter Start-Ort oder Ziel-Ort eingeben.
2. Die Position wird auf der Karte je nachdem als "Start" oder "Ziel" angezeigt.

5.3.5 Story: Route berechnen

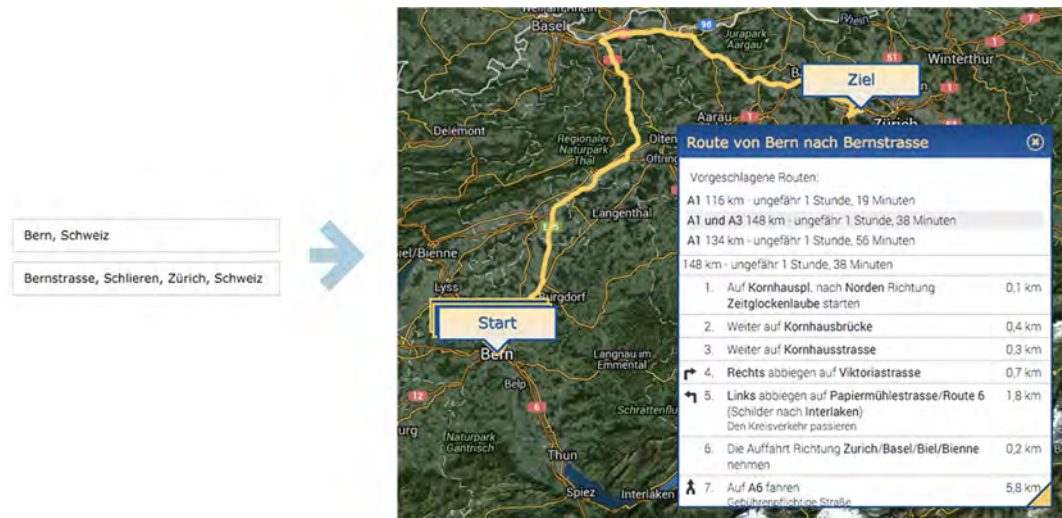


Abbildung 5.16: Story: Route berechnen

1. In der Webapp den Startort und den Zielort eingeben.
2. Die Strecke wird auf der Karte angezeigt. Die Streckenbeschreibung kann frei verschoben, vergrößert oder verkleinert werden und es können alternative Strecken ausgewählt werden.

5.3.6 Story: Route von Fahrzeug als Startpunkt berechnen



Abbildung 5.17: Story: Route von Fahrzeug als Startpunkt berechnen

1. In der Webapp den Ziel-Ort eingeben.
2. Das gewünschte Taxi aus dem Menu auswählen und den Link "Route zu Ziel-Ort anzeigen" anklicken.

- Die Strecke wird auf der Karte angezeigt. Die Streckenbeschreibung kann frei verschoben, vergrößert oder verkleinert werden und es können alternative Strecken ausgewählt werden.

5.4 Akteur: Revisor / Qualitätssicherung

Der Revisor verwendet die Applikation, um aufgrund der Daten festzustellen, ob gesetzliche Richtlinien wie Pausenzeiten eingehalten werden.

5.4.1 Story: Daten auswerten

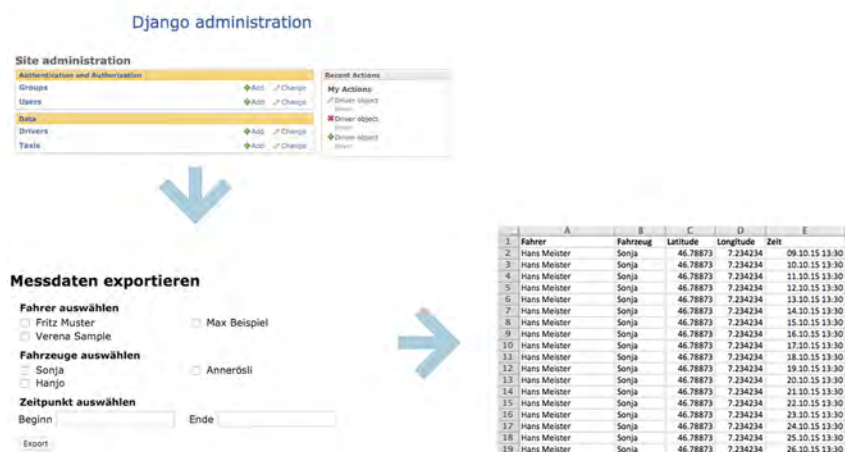


Abbildung 5.18: Story: Daten auswerten

- Anmeldung in Web-Administrationsoberfläche.
- Die gespeicherten Positionen können entweder komplett oder nach Fahrer, Taxi und/oder Datum gefiltert als CSV-Datei heruntergeladen werden.
- Die Daten können in einer Tabellenkalkulations-Software wie Excel entsprechend gefiltert und analysiert werden.

6 Technologien / Komponenten

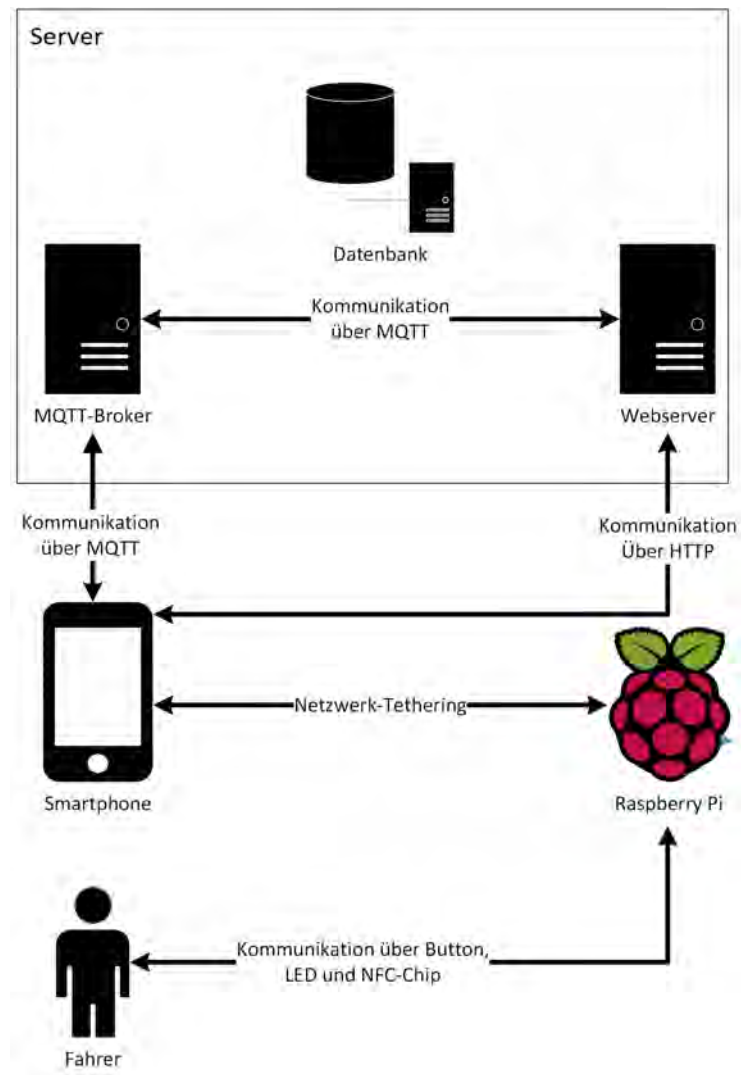


Abbildung 6.1: Netzwerk Überblick

6.1 Raspberry Pi

Wir evaluierten mehrere Einplatinencomputer und tendierten zuerst auf den von Tinkerforge neu eingeführten REDbrick, stellten aber fest, dass dieser standardmässig eine eigene, proprietäre Linux-Distribution verwendet und ausserdem für die zuerst angedachten Audio-Funktionalitäten zur Kommunikation zwischen Fahrer und Disponent keine geeigneten Sensoren von Tinkerforge existieren.

Des weiteren wurde auch der BeagleBone Black angeschaut. Dieser Einplatinencomputer ist dem Raspberry Pi sehr ähnlich, verfügt aber über nur einen USB-Anschluss und ist zudem leicht teurer als der Raspberry Pi. Die bessere Performanz der BeagleBone Hardware wird für dieses Projekt nicht benötigt.

Der Entschied fiel schlussendlich auf den Raspberry Pi aus folgenden Gesichtspunkten:

- Offene Architektur.
- 4 USB-Anschlüsse.
- Freie Wahl der Linux-Distribution.
- Riesige Bibliothek an bestehenden Projekten.
- Bereits bestätigte Möglichkeit, den CAN-Bus eines Fahrzeugs auszulesen.
- Unschlagbar günstiger Preis

6.2 Sensoren

Sowohl für die GPS-Daten als auch den Kilometerstand und die Aktivierung per NFC werden zusätzliche Sensoren verwendet. Hier standen ebenfalls mehrere Produkte zur Auswahl.

6.2.1 NFC Leser

Zum Auslesen von NFC-Tags wird ein Tinkerforge MasterBrick sowie ein Tinkerforge NFC-Leser verwendet.

6.2.2 GPS

Die GPS-Koordinaten werden über einen USB GPS-Empfänger ausgelesen. Im Projekt wird das Modell "u-blox 6" der Firma Navilock verwendet. Die Daten können unter Linux mit GPSTools ausgelesen werden.

6.3 Betriebssystem

Als Basis für den Raspberry PI wird die Distribution Minibian[8] verwendet, da diese mit einer Grösse von 200 MByte sehr schlank ist und ausser einer für Raspberry Pi optimierten Debian Grundinstallation keine weiteren, überflüssigen Komponenten enthält.

Bei den ersten Versuchen verwendeten wir thethingbox.io. Dies ist besonders für den Einstieg in das Internet of Things mit dem Raspberry Pi geeignet. Aber es ist darum auch grösser als Minibian und enthält viele nicht benötigte Komponenten sowie eine graphische Benutzeroberfläche zum einfachen Zusammenklicken von Modulen, welche aber für dieses Projekt nicht benötigt wird.

6.4 Apache Webserver

Der Apache Webserver wird verwendet, um die Daten auf der Karte sowie die Administrations-Oberfläche anzuzeigen. Zusätzlich ist das WSGI-Modul für Apache erforderlich, um die Daten aus Django auf dem Web auszugeben.

6.5 Bluetooth

Als Kommunikationstechnologie für das Tracking-Gerät wird Bluetooth verwendet, da mit dieser Technologie sowohl der CAN-Bus Controller angesprochen als auch das Netzwerk-Tethering mit dem Smartphone hergestellt werden kann.

Zusätzlich verbraucht Bluetooth weniger Energie als WLAN-Technologie. Dies ist zwar im Kontext dieses Projekts nicht von zentraler Bedeutung, hilft aber, dass im Fahrzeug die Batterie weniger belastet wird.

Im Verlauf des Projektes stellte sich allerdings Tethering per USB als die geeignetere Technologie heraus. Und da das Auslesen weiterer Daten per CAN-Bus zurückgestellt wurde, wird Bluetooth in der finalen Version des Prototypen nicht mehr verwendet.

6.6 USB-Tethering

Da beim Tethering per Bluetooth bei jedem Verbindungsaufbau eine Interaktion des Benutzers mit dem Smartphone nötig ist, damit die Verbindung hergestellt werden kann und dies für EasyCab nicht optimal ist, wurde der Ansatz, das Gerät per Bluetooth mit dem Smartphone zu koppeln, wieder verworfen und durch ein USB-Tethering des Smartphones ersetzt.

Dies bedingt zwar, dass der Fahrer das Smartphone per Kabel mit dem Tracking-Gerät verbinden muss, damit eine Verbindung zum Internet hergestellt werden kann, hat aber den Vorteil, dass keine weitere Interaktion auf dem Smartphone oder dem Tracking-Gerät notwendig ist und dass auch klar definiert werden kann, wann und mit welchem Smartphone das Tracking-Gerät verbunden wird.

6.7 Datenbank

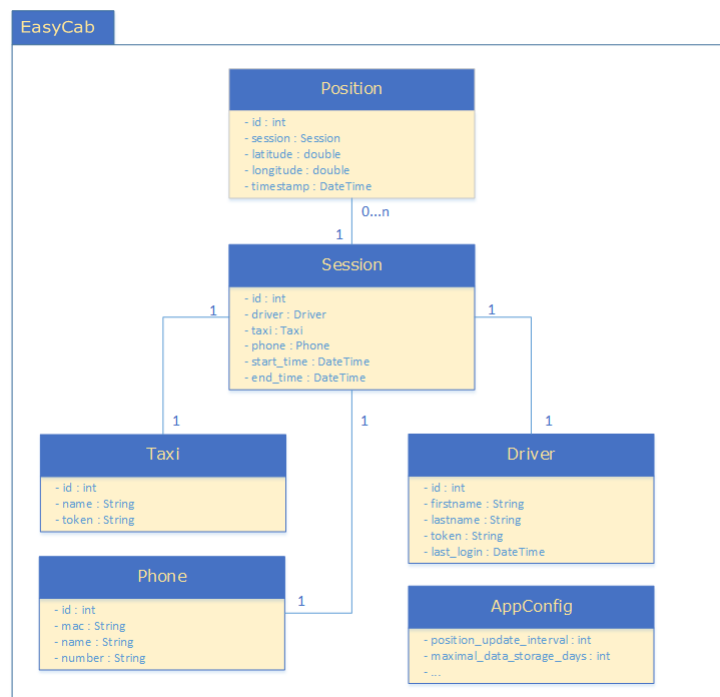


Abbildung 6.2: Domänen-Modell der Datenbank

Da EasyCab bereits intern mit MySQL arbeitet, wird für die Speicherung der Daten ebenfalls auf MySQL gesetzt. Allerdings wird durch den MQTT-Broker eine Schnittstelle geschaffen, über welche die Daten auch problemlos

in eine andere Datenbank (MS SQL, Oracle, NoSQL, ...) gespeichert werden können und durch den Einsatz von Django kann auch durch eine einfache Konfigurations-Anpassung eine andere Datenbank-Technologie verwendet werden.

Die Tabelle "Phone" wird eingefügt damit die Telefonnummer des Smartphones ermittelt werden kann, welches mit dem Tracking-Gerät verbunden ist und der Disponent den Fahrer so direkt kontaktieren und identifizieren kann, falls sich dieser ohne NFC-Chip angemeldet hat.

Die Tabelle "Session" wird für eine Sitzung von jeweils einem Fahrer und einem Smartphone in einem Taxi verwendet. Wenn dem Tracking-Gerät weitere Sensoren hinzugefügt werden, kann eine zusätzliche Tabelle erstellt werden, welche auf die Tabelle 'Session' referenziert.

In der Tabelle "AppConfig" werden Applikationsrelevante Einstellungen wie der gewünschte Intervall der Messungen oder der Zeitraum, wie lange die Daten aufbewahrt werden sollen, hinterlegt. Die Tabelle ist unabhängig von den übrigen Tabellen und wurde in erster Linie daher erstellt, damit Anpassungen an der Applikation ohne Eingriff auf das Tracking-Gerät vorgenommen werden können.

Da für das Projekt Django verwendet wird, sind Anpassungen an diesen Tabellen nicht direkt, sondern über die in Django definierten Models vorzunehmen.

6.8 Python

Python ist eine universelle, interpretierte höhere Programmiersprache. Die Vorteile von Python sind eine hohe Lesbarkeit und die Möglichkeit, mit dem Betriebssystem zu interagieren. Die Entwickler des Raspberry Pi haben von Anfang an auf Python als Scriptsprache gesetzt. Das "Pi" im Namen steht für "Python Interpreter". EasyCab möchte allfällige Weiterentwicklungen an der Applikation mit Python vornehmen, daher wird für das Projekt vorwiegend auf diese Sprache gesetzt.

6.9 Django

Django ist ein in Python geschriebenes, OpenSource Web Application Framework, das einem Model-View-Presenter-Schema folgt. Da für die Applikation bereits Python verwendet wird, bietet es sich an, die Administrations-Oberfläche sowie die Snippets, welche per JavaScript geladen werden mit Django umzusetzen. So kann auch eine hohe Kompatibilität mit unterschiedlichen Datenbank-Systemen gewährleistet werden.

In diesem Projekt wird Django vor allem verwendet, um Datenbankabfragen auszuführen und die Daten in JSON- oder HTML-Komponenten aufzubereiten. Durch das Verwenden von Django lässt sich die Administrations-Oberfläche mit sehr geringem Aufwand umsetzen.

Durch das Verwenden der Django QuerySet API kann zudem auf proprietäre SQL-Abfragen verzichtet werden. Dies erhöht nicht nur die Portabilität für andere Umgebungen, sondern erledigt zusätzlich eine Prüfung gegen SQL-Injections.

6.10 JavaScript / Google Maps API / jQuery

JavaScript wird für die Weboberflächen verwendet, um die Positionsdaten per Google Maps API anzuzeigen. In einem ersten Schritt wurden die Daten per Polling gelesen. Da Mosquitto aber seit Version 1.4 erlaubt, über einen Websocket angesprochen zu werden, kann die Karte per JavaScript in Echtzeit aktualisiert werden.

jQuery wurde gewählt, um Modifikationen am HTML Document Object Model mit weniger Quelltext und gleichzeitig hoher Browserkompatibilität vornehmen zu können.

Da die Darstellung der Karte zwingend JavaScript benötigt, kann die Applikation nicht verwendet werden, wenn im Browser JavaScript deaktiviert wird.

6.11 MQTT

MQTT ist ein Protokoll zur Übertragung von Telemetrie-Daten für das Internet of Things (IoT). Es ist ein offenes, leichtgewichtiges Protokoll für Datenübertragung bei eingeschränkten Bandbreiten oder hoher Latenz und Netzausfällen.

Anstelle einer Punkt-zu-Punkt Verbindung gibt es einen zentralen Server, der Broker genannt wird. Dieser verwaltet die übermittelten Nachrichten als Topics. Diese werden in einer hierarchischen Struktur ähnlich einem Dateisystem dargestellt. Ein / dient dabei als Trennzeichen. Es genügt, ein Nachricht auf einem Topic zu veröffentlichen, es muss nicht speziell konfiguriert werden.

Bsp. `fahrzeuge/FAHRZEUG_NAME/sensoren/SENSOR_NAME`

MQTT arbeitet nach dem Publisher/Subscriber Prinzip. Als Publisher können Nachrichten auf Topics veröffentlicht werden. Subscriber können sie sich für ein Topic registrieren, um die entsprechenden Nachrichten zu erhalten. Diese erhalten sie sobald diese veröffentlicht werden. Ein Subscriber kann sich auf ein einzelnes Topic einschreiben, um nur Nachrichten für dieses Topic zu erhalten oder mit Hilfe der beiden Wildcards `+` oder `#` alle Nachrichten von Topic-Gruppen zu erhalten.

Ein Client kann gleichzeitig Publisher als auch Subscriber sein. Somit kann über MQTT Geräte auch gesteuert und konfiguriert werden.

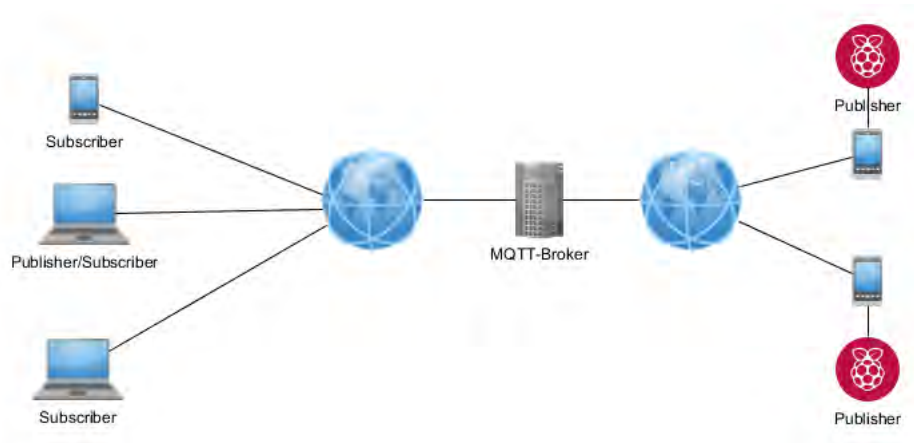


Abbildung 6.3: MQTT

Ein Broker bewahrt eine Nachrichten auf, bis sie allen aktuellen Subscribern geschickt wurde. Eine Nachricht kann aber auch als Retained Message veröffentlicht werden. Diese werden auch an Subscriber gesendet, die sich nach der Veröffentlichung eingeschrieben haben.

Wenn ein Client sich mit einem Broker verbindet, kann er diesem mitteilen, dass er einen "will" hat. Wenn der Client unvorhergesehen vom Broker getrennt wird, kann der Broker diesen "last will" in Form einer Nachricht ausführen. Dadurch kann zum Beispiel Steuerungen mitgeteilt werden, dass ein Sensor ausgefallen ist. MQTT definiert drei Quality of Service (QoS) Stufen. Der QoS definiert, wie stark der Broker/Client versuchen wird, dass die Nachricht empfangen wird. Höhere Stufen des QoS sind zuverlässiger, beinhalten aber höhere Latenz und verlangen höhere Bandbreiten.

- 0: Der Broker/Client wird die Nachricht einmal verschicken, ohne Bestätigung.
- 1: Der Broker/Client wird die Nachricht mindestens einmal verschicken, mit einer Bestätigung.
- 2: Der Broker/Client wird die Nachricht genau einmal verschicken und dabei einen Four Step Handshake benutzen.

Um MQTT leicht zu halten, sind ausser einer Username/Passwort Unterstützung in der API nur wenige eigene Sicherheits-Mechanismen eingebaut. Die Daten und die Broker müssen also mit einem unabhängigen Mechanismen und Verschlüsselungs-Protokollen geschützt werden.

[12]

6.12 Elektronische Bauteile

Damit der Fahrer über den Zustand des Geräts informiert ist und er den Tracking-Dienst beenden und neu starten kann, werden LED und ein Button an das Gerät angeschlossen. Diese werden über die GPIO Schnittstelle des Raspberry Pi angesteuert.

6.12.1 Elektronische Grundlagen

Um die benötigten Bauteile, in erster Linie die Widerstände, zu bestimmen, sind einige elementare Regeln und Gesetze der Elektronik notwendig:[13]

- **Ohmsches Gesetz:**

Der Elektrische Strom I durch eine Leitung und die darüber angesetzte Spannung U sind proportional. Der Quotient ist der Elektrische Widerstand R .

$$\frac{U}{I} = R$$

- **Kirchhoffsche Knotenregel:**

In einem Knoten eines Schaltkreise ist die Summe der Eingangströme gleich der Summe der Ausgangsströme. Im der Abbildung wäre das $I_1 = I_2 + I_3 + I_4$

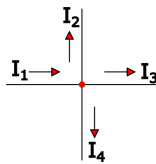


Abbildung 6.4: Knotenregel

- **Kirchhoffsche Maschenregel:** Alle Teilspannungen in einer Masche addieren sich zu Null. In der Abbildung wäre das $+U_1 - U_2 - U_3 = 0$

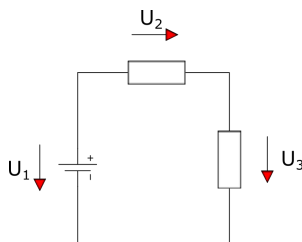


Abbildung 6.5: Maschenregel

- **Berechnen des Vorwiderstand für eine LED:**

Der benötigte Vorwiderstand R_V ist der Quotient der Spannung über dem Widerstand U_V und dem Strom I .

$$R_V = \frac{U_V}{I}$$

Die Spannung über dem Vorwiderstand ist nach der Maschenregel die Gesamtspannung U_{Ges} minus die Spannung über dem LED U_L

$$U_V = U_{Ges} - U_L$$

- **Elektrische Leistung:**

Die Elektrische Leistung P bezeichnet die Energie pro Zeit. Für Gleichstrom ist die Leistung das Produkt aus Spannung U und Stromstärke I .

$$P = U \cdot I \text{ und da } I = \frac{U}{R} \text{ ist}$$

$$P = \frac{U^2}{R}$$

6.12.2 LED

Die verwendeten LED sind für eine Durchlassspannung von 2.1V und einen maximalen Nenn-Durchlassstrom von 20 mA ausgelegt. Bei einem zu hohen Strom würde die LED innert kürzester Zeit zerstört. Bei zu niedrigem Strom würde sie nicht leuchten. Darum muss ein Vorwiderstand den Strom regulieren.

6.12.3 Widerstände

Die Raspberry Pi Foundation gibt keine genauen elektronischen Spezifikationen betreffend der Input/Output Pins heraus. Die verwendeten Spezifikationen stammen von der Webseite von Mosaic Industries. [7]

Der Raspberry Pi sollte über die 3.3 V Pins und die GPIO-Pins zusammen eine maximale Strommenge vom etwa 50 mA liefern. Bei höheren Strommengen würde das Raspberry Pi zuwenig Strom für den Betrieb haben oder es könnte beschädigt werden.

Es werden 5 LED und der Button parallel geschaltet. Alle 5 LED und der Button dürfen nach der Knotenregel zusammen nicht mehr als 50 mA Strom beziehen.

Der Button braucht nur eine minimale Menge Strom. Deshalb wird ein hoher Widerstand von 10 kΩ verwendet. Die LED sollten jeweils weniger als 10 mA ziehen. Wir nehmen pro LED eine maximale Strommenge von 8 mA an.

Berechnung der Spannung U_V über den Vorwiderstand:

$$U_V = 3.3V - 2.1V$$

$$U_V = 1.2V$$

Und über den Widerstand R_V

$$R_V = \frac{1.2V}{8mA}$$

$$R_V = 150\Omega$$

Das ist der minimale Widerstand. Der nächsthöhere als Bauteil erhältliche Widerstand ist 220Ω. Damit ist auch der Umstand, dass der tatsächliche Widerstandswert um bis zu 5% abweichen kann, berücksichtigt. Tests mit höheren Widerständen als 220 Ω lieferten keine befriedigende Lichtleistung.

Jeder Widerstand hat auch eine Nennleistung, die die maximal Elektrische Leistung die der Widerstand verträgt. Die Leistung des Widerstand ist:

$$P = \frac{1.1^2}{220\Omega}$$

$$P = 5.5mW$$

Darum können die verbreiteten Widerstände mit 250 mW Nennleistung verwendet werden.

6.13 CAN-Bus

Controller **A**rea **N**etwork ist ein serielles Bus-System das als Feldbus Aktoren und Sensoren mit einem Steuerungsgerät verbindet. Es wurde von der Robert Bosch GmbH ab 1983 als Steuergerät für die Automobil-Industrie entwickelt. Mittlerweile wurde es zur Standard-Schnittstelle für Fahrzeugdaten und wird auch in medizinischen oder industriellen Anlagen eingesetzt.

7 Umsetzung

Dieses Kapitel zeigt die Schritte bei der Umsetzung der Applikation und des Tracking-Geräts.

7.1 Tracking-Gerät

7.1.1 1. Prototyp

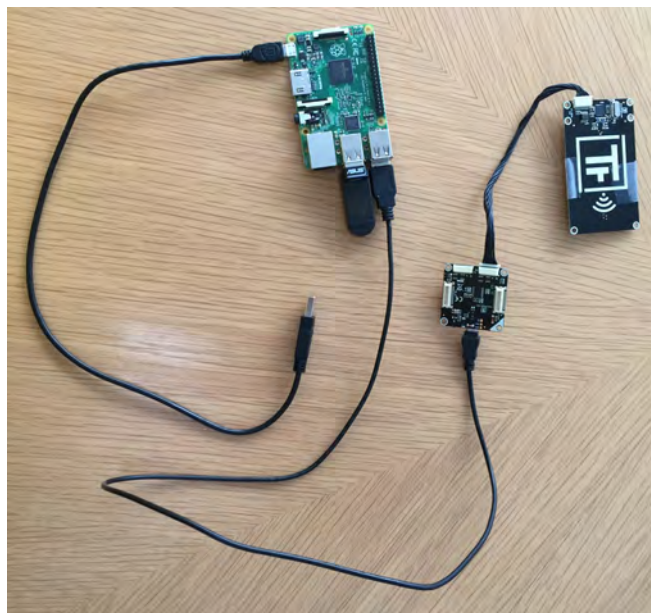


Abbildung 7.1: Erster Prototyp des Tracking-Geräts

Der erste Prototyp des Tracking-Geräts, welcher an EasyCab zu Testzwecken ausgeliefert wird, beinhaltet noch keine LED oder Buttons. Die GPS-Koordinaten werden bereits an den MQTT-Broker gesendet sobald folgende Voraussetzungen erfüllt werden:

- Tracking-Gerät ist per USB ans Stromnetz angeschlossen
- Tracking-Gerät ist per Bluetooth mit dem Smartphone verbunden, Bluetooth-Tethering am Smartphone ist aktiviert.
- Tracking-Gerät wurde per NFC-Chip zur Identifikation des Fahrers aktiviert.
- Tracking-Gerät empfängt GPS-Daten

Die Funktionalität, dass das Gerät das Fahrzeug aufgrund eines NFC-Tags erkennt wurde noch nicht implementiert. In dieser ersten Version des Tracking-Geräts entspricht ein Gerät jeweils einem Fahrzeug.

Die Daten werden in diesem Prototyp noch unverschlüsselt gesendet.

Die Koppelung des Smartphones mit dem Tracking-Gerät funktioniert, indem das Tracking-Gerät alle 30 Sekunden an alle sichtbaren Bluetooth-Geräte eine Koppelungs-Anfrage sendet, welche auf dem entsprechenden Smartphone bestätigt werden muss.

7.2 Tethering

Für das Tethering der Smartphones wurde zu Beginn Bluetooth als Technologie ausgewählt. Bluetooth verwendet im Vergleich zu WLAN weniger Strom und während beim Tethering per WLAN ein Passwort verwendet werden muss, welches für dieses Projekt bei jedem Gerät identisch sein müsste, lässt sich das Tethering per Bluetooth auch ohne Passwort-Authentifikation umsetzen. Das Tethering per USB wurde zu Beginn ignoriert, da das Smartphone per Kabel mit dem Tracking-Gerät verbunden werden muss.

Bei der Umsetzung der Idee stellte sich aber heraus, dass Bluetooth einige Nachteile mit sich bringt. So muss beim Pairing des Smartphones mit dem Tracking-Gerät jedes Mal auf dem Smartphone bestätigt werden, dass die Verbindung hergestellt werden soll, was für den produktiven Einsatz relativ umständlich ist. Ausserdem wird jedes Gerät mit aktiviertem Bluetooth in der Umgebung für die Verbindung angefragt und könnte sich somit auch ein externes Smartphone für die Verbindung nutzen lassen. Auch das Trennen der Verbindung ist relativ umständlich, da in den Einstellungen Bluetooth deaktiviert werden muss um sicher zu stellen, dass keine Verbindung aufgebaut wird.

So wurde erneut evaluiert, welche Technologie für das Tethering verwendet werden soll und haben wir uns dennoch für USB entschieden. Zwar muss das Smartphone so per Kabel mit dem Tracking-Gerät verbunden werden, aber der Punkt des Stromverbrauchs verliert an Relevanz, es kann physisch bestimmt werden, welches Smartphone verbunden werden soll und die Verbindung lässt sich auch physisch und sehr einfach trennen.

In der Praxis stellte sich heraus, dass das Tethering per USB auf dem iPhone wie erhofft funktionierte und das Gerät eine Verbindung zum Internet aufbaut, sobald ein iPhone verbunden wird. Bei Android-Smartphones muss vor jeder Verbindung das Tethering in den Einstellungen erneut aktiviert werden und bei Windows Smartphones wurde USB-Tethering seit Windows 8 grundsätzlich deaktiviert.

Im Verlauf des Projekts stellte sich heraus, dass EasyCab ausschliesslich iPhones verwenden wird und somit bot sich das USB-Tethering als ideale Lösung an. Falls zu einem späteren Zeitpunkt andere Smartphones verwendet werden sollten, muss die Technologie für das Smartphone-Tethering allerdings neu evaluiert werden.

7.3 LED und Button

Die ersten Entwürfe wurden mit Fritzing erstellt. Fritzing ist eine ursprünglich von der Fachhochschule Potsdam entwickelte Software um Pläne für elektronische Schaltkreise zu entwerfen. Fritzing ist besonders für Prototyping mit Plattformen wie Raspberry Pi oder Arduino geeignet. [6]

7.3.1 Erster Entwurf

Im ersten Plan waren zweifarbige LED, die zwischen rot und grün wechseln, vorgesehen. Die Auswahl der GPIO erfolgte hier noch rein aus Gründen der Übersichtlichkeit.

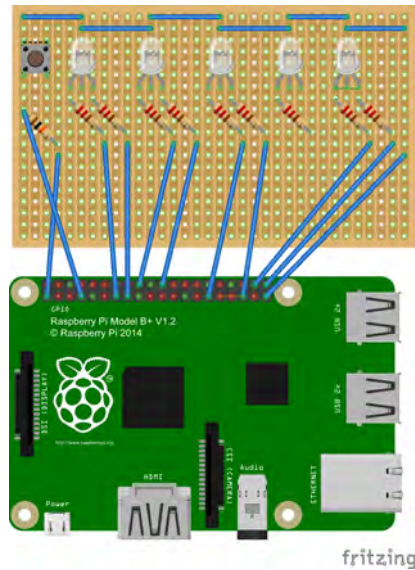


Abbildung 7.2: LED und Button: Erster Entwurf

Dieser erste Entwurf wurde aus folgenden Gründen verworfen:

1. Licht an/aus ist verständlicher und auffälliger als Farbwechsel.
2. Die Platine soll so klein wie möglich sein.
3. Die Anzahl der Bauteile soll auf ein Minimum beschränkt werden. Bei dieser Lösung müssten für jede LED zwei GPIO inkl. Verkabelung verwendet werden. Ausserdem müssten pro LED 2 Vorwiderstände eingebaut werden.

7.3.2 Zweiter Entwurf

Die zweifarbigen LED wurden durch einfarbige, grüne ersetzt. Auch hier wurde die GPIO so ausgewählt, dass die Zeichnung übersichtlich ist.

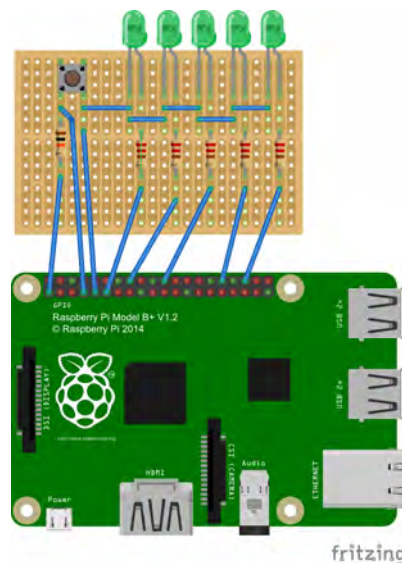


Abbildung 7.3: LED und Button: Zweiter Entwurf

7.3.3 Versuchsaufbau

Auf der Grundlage des zweiten Entwurfs wurde der erste Test auf einer Steckerplatine aufgebaut. Mit diesem Versuchsaufbau wurde auch die erste Version des Python Scripts zur Steuerung der LED und Buttons implementiert und getestet.

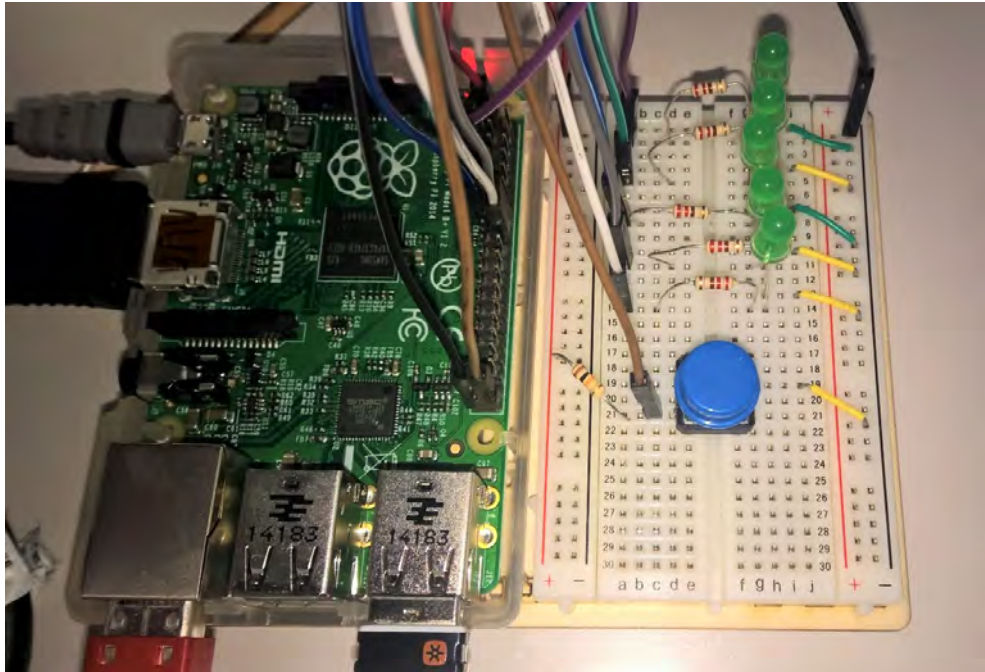


Abbildung 7.4: LED und Button: Versuchsaufbau

Zur Steuerung der LED und des Buttons wurden die Pins

- 11 (GPIO 17) Fahrer
- 13 (GPIO 27) Fahrzeug
- 15 (GPIO 22) GPS
- 16 (GPIO 23) Internet
- 18 (GPIO 24) Bereitschaft
- 22 (GPIO 25) Button

gewählt. Diese liegen nebeneinander und es ist einfacher die richtigen, weil nachfolgenden Pins, zu finden.

Die 3.3V Spannungsquelle für den Button und die Masse (engl. Ground) sind frei wählbar. In diesem Aufbau wurde für die Quelle der Pin 1 und für die Masse der Pin 39, jeweils am Ende der Pin-Leiste, gewählt. Im endgültigen Gerät werden dann die Pins 17 und 20, die direkt bei den GPIO für Button und LED liegen, verwendet.

Für den Button wird ein Pullup-Widerstand verwendet. Wenn der Button nicht gedrückt ist, liegt am GPIO eine hohe Spannung an. Der GPIO erhält also das Signal "HIGH". Wird der Button gedrückt fließt der Strom zur Masse und das GPIO erhält das Signal "LOW". Das sind die zwei Signale, die ein GPIO auslesen kann.



Abbildung 7.5: GPIO Layout Raspberry Pi B+ und 2

7.3.4 Prototyp

Nachdem die erste Versuchsaufbau erfolgreich war, wurde ein erster Prototyp auf eine Platine aufgelötet.

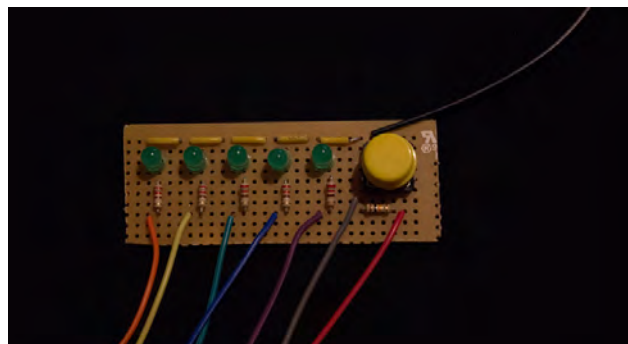


Abbildung 7.6: Prototyp der LED/Button Platine

7.4 Gehäuse

Für das Tracking-Gerät soll mit einem 3D-Drucker ein Gehäuse gedruckt werden. Hierzu wurde mit der freien 3D-Modellierungs-Software Blender [2] ein Gehäuse modelliert. Hierbei wurden folgende Punkte berücksichtigt:

- Die einzelnen Komponenten sollen ohne Schrauben im Gehäuse stabil platziert werden können
- Die USB- und Micro-USB-Buchsen des Raspberry Pi müssen erreichbar bleiben
- LED und Button müssen erreichbar, respektive sichtbar sein

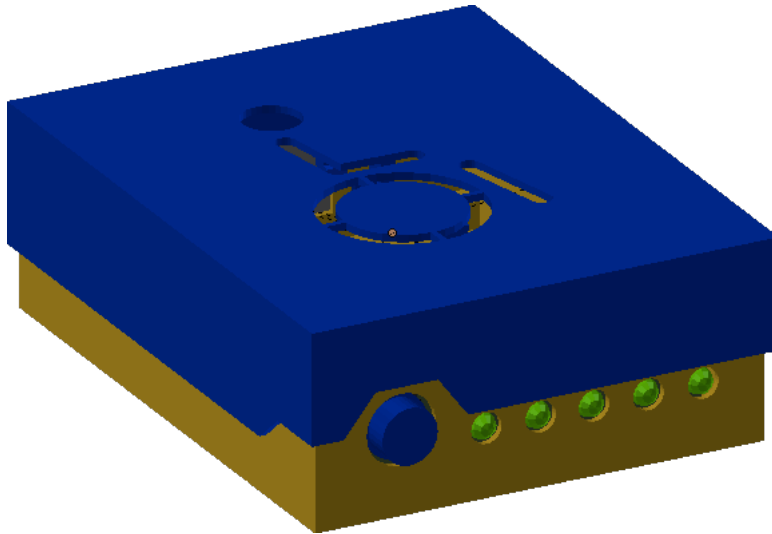


Abbildung 7.7: 3D-Ansicht des ersten Gehäuse-Prototypen

Um die Proportionen des Gehäuses möglichst genau definieren zu können, wurden die Komponenten ebenfalls in Blender als 3D-Modelle schemenhaft nachgebaut und im Modell des Gehäuses platziert.

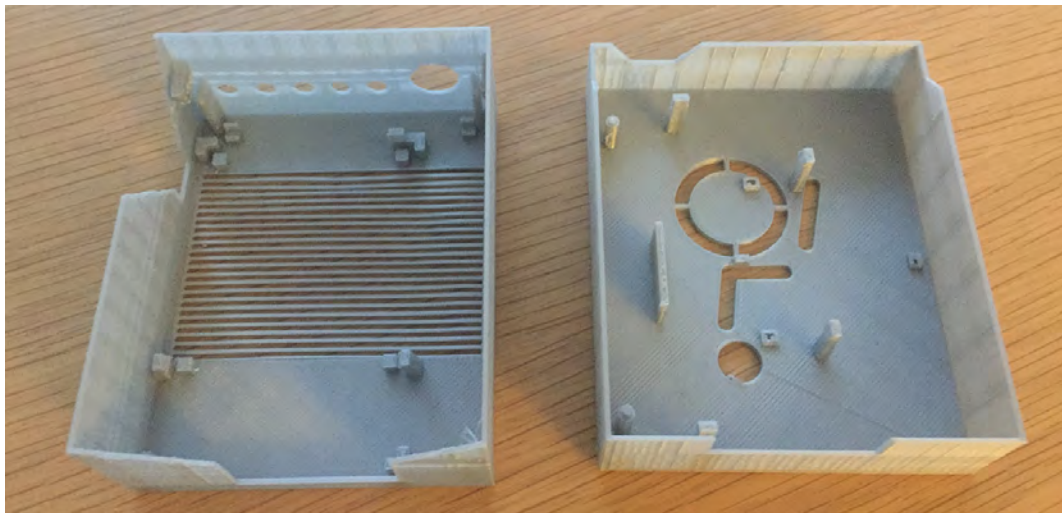


Abbildung 7.8: 3D-Ausdruck des ersten Prototypen

Nach dem Ausdruck des ersten Prototypen stellte sich heraus, dass die Wände zu dünn waren und das Gehäuse daher nicht die erwünschte Stabilität hatte. Somit wurde ein zweiter Prototyp mit breiteren Wänden erstellt. Ebenfalls wurde die Höhe des Geräts vergrößert, damit die Kabel der Komponenten besser untergebracht werden können.

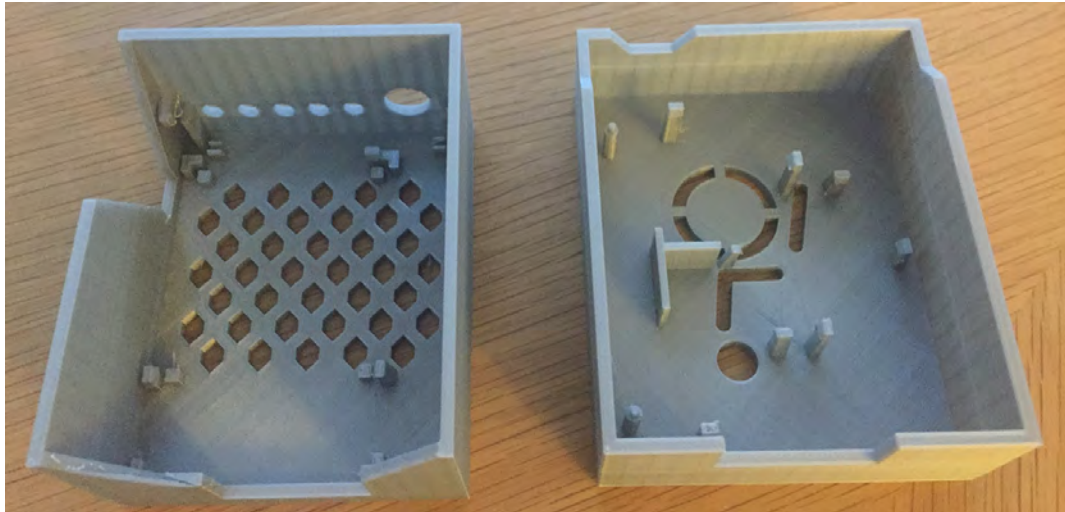


Abbildung 7.9: 3D-Ausdruck des zweiten Prototypen

Nachdem der zweite Prototyp gedruckt wurde, stellte sich heraus, dass die LED und der Button nicht mehr in den unteren Teil des Gehäuses eingefügt werden konnten, da das Gehäuse und die Clippings nicht mehr dehnbar genug waren. So wurde das Gehäuse erneut überarbeitet und die Clippings dünner modelliert, sowie die gesamte Struktur der LED/Button Platinenhalterung überarbeitet.

Beim Ausdruck dieser dritten Version wurde erstmals der Online-Dienst www.teil3.ch zum Drucken verwendet. Dieser Dienst bietet an, eine STL-Datei hochzuladen, welche dann ausgedruckt und per Post gesendet wird. Ein grosser Vorteil dieses Dienstes ist, dass für den Druck verschieden Kunststoffe und unterschiedliche Farben gewählt werden können.

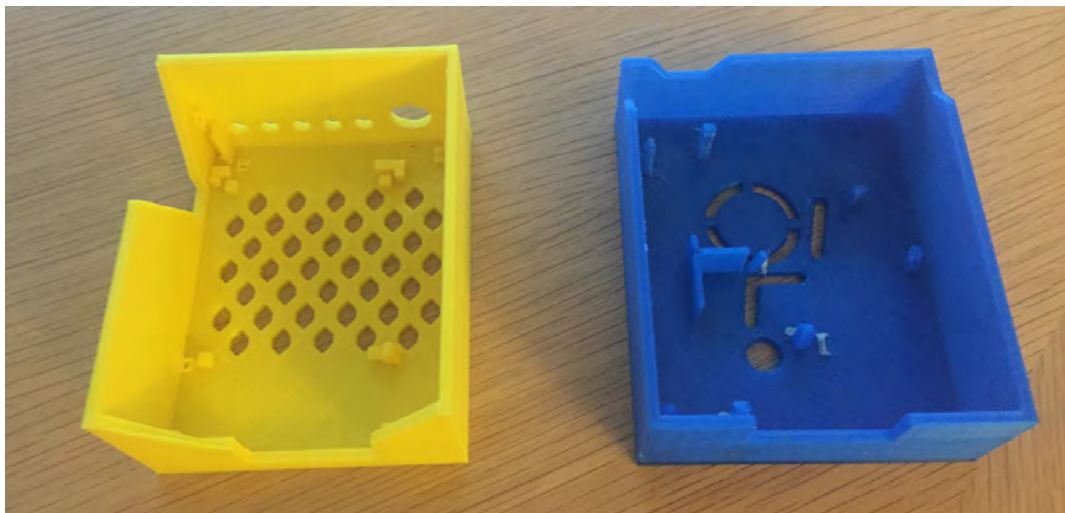


Abbildung 7.10: 3D-Ausdruck des dritten Prototypen

Es stellte sich aber heraus, dass die LED/Button Platinenhalterung erneut überarbeitet werden musste, da der Button zu nahe an der Wand des Gehäuses war und so nicht mehr gedrückt werden konnte, respektive immer gedrückt war. So wurde die Öffnung für den Button angepasst, indem eine zusätzliche Ausbuchtung für den inneren, breiteren Ring des Buttons eingefügt wurde. Zusätzlich wurden die Clipping stabilisiert, da der von www.teil3.ch verwendete Kunststoff brüchiger schien und beim Einbau einige Clippings abgebrochen waren.

Nachdem das Gehäuse in den endgültigen Farben vorlag, konnten auch die Aufkleber für das Gehäuse entworfen werden, damit später zu erkennen ist, welches LED zu welchem Status gehört. Die ursprüngliche Idee war es, die Icons in gelber Schrift auf den blauen Deckel des Gehäuses aufzukleben. Da jedoch der Ausdruck auf eine

transparente Folie kein deckendes Gelb ergab und somit die Icons nicht leserlich genug erschienen, entschieden wir uns für die Variante mit blauen Icons auf dem gelbern unteren Teil des Gehäuses.



Abbildung 7.11: Gehäuse mit aufgeklebten Icons in gelb und blau

7.5 Server-Konfiguration

Für die Entwicklungsphase wird eine cloud-basierte Server-Instanz der Firma “Digital Ocean” verwendet. Als Betriebssystem wurde Ubuntu in der LTS Version 14.04 gewählt.

Der Test-Server ist für die erste Konfiguration entsprechend vorbereitet, dass Mosquitto mit Unterstützung für Websockets installiert wurde. Ausserdem ist auf dem Server MySQL mit den entsprechenden Zusatzpaketen für die Kommunikation mit Python installiert und eine Installation für Django sorgt dafür, dass die Daten per Python aus der Datenbank gelesen und administriert werden können.

7.6 Web-Applikation

7.6.1 1. Prototyp

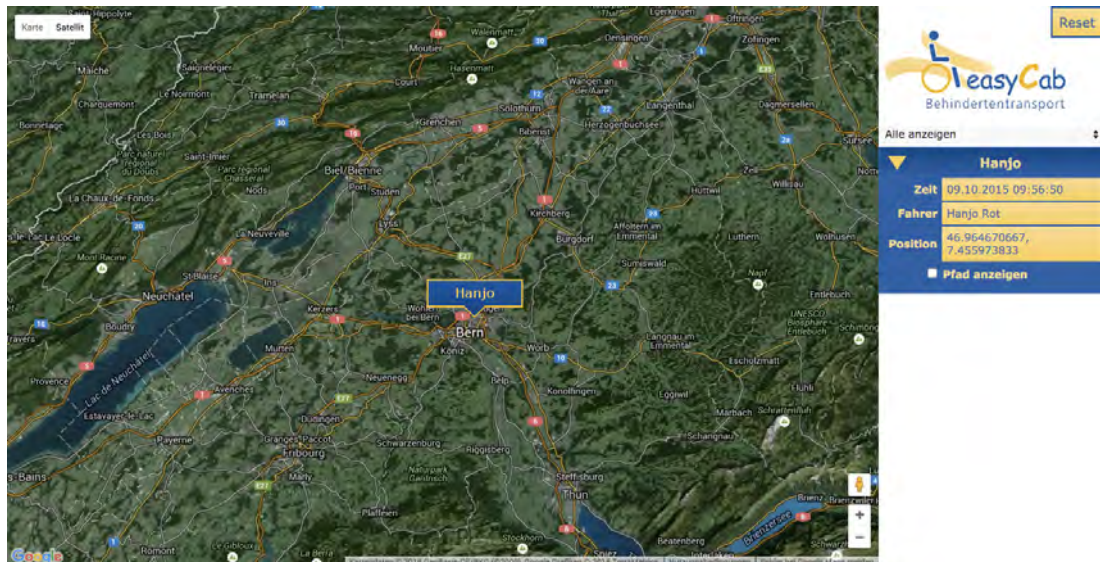


Abbildung 7.12: Erster Prototyp der Karten-Applikation

Django administration



Abbildung 7.13: Erster Prototyp der Administrations-Oberfläche

Im ersten Prototyp der Web-Applikation werden die Koordinaten der Fahrzeuge bereits in Echtzeit auf der Karte angezeigt. Die Daten für Fahrer, Fahrzeuge und Smartphones können in der Administrations-Oberfläche bearbeitet und eingesehen werden. Noch nicht implementiert wurden folgende Funktionalitäten:

- Export der Daten inklusive Filter nach Fahrer, Fahrzeugen und Datum.
- Eingabemaske zum Anpassen von Applikationseinstellungen.

Das Design der Web-Applikation ist noch nicht endgültig und ist als Vorschlag zu verstehen, welcher durch den Kunden noch ergänzt oder angepasst werden kann.

7.6.2 Erweiterung mit Karten-Such-Funktionalitäten

Aufgrund des Inputs von EasyCab stellte sich heraus, dass ein Bedürfnis besteht, zusätzlich nach einem Start- und Ziel-Ort zu suchen. Der Disponent nutzt diesen Anwendungsfall sehr oft und würde es begrüßen, dies ebenfalls in

der Applikation vornehmen zu können. Daher wurde die Applikation mit zwei Suchfeldern und einem Anzeige-Feld für die entsprechenden Routen-Berechnungen erweitert.

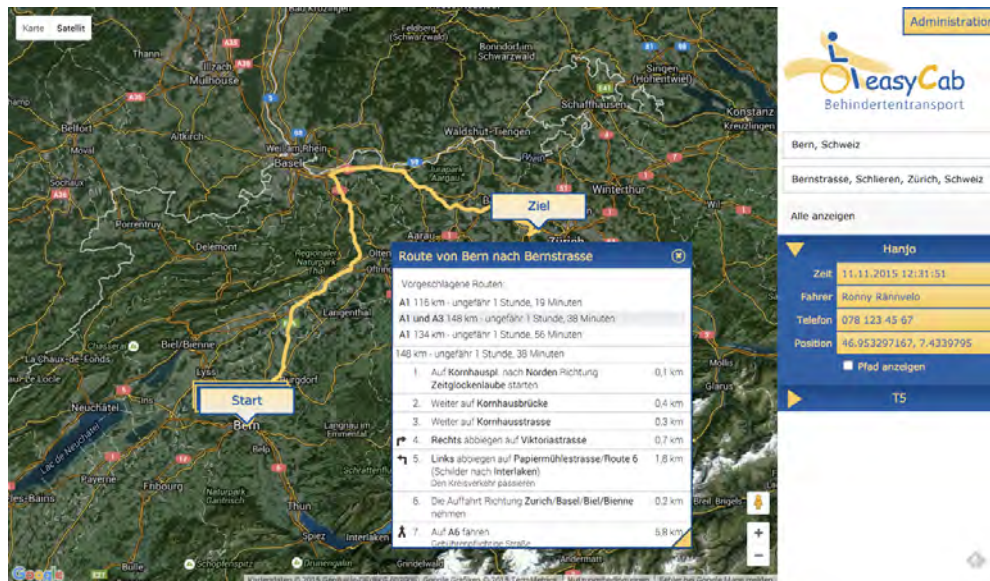


Abbildung 7.14: Karte mit Such- und Streckenberechnungs-Funktionalität

7.7 Software

Grundsätzlich werden Shell-Scripts sowohl beim Tracking-Gerät wie auch auf dem Server im Verzeichnis `/root/` abgelegt, Python-Scripts befinden sich unter `/usr/local/python/`.

7.7.1 Client / Tracking-Gerät

Die Logik der easyCab Applikation soll der Linux-Daemon `easycabd` übernehmen. Der Daemon wird gestartet, sobald das Tracking-Gerät hochgefahren ist.

Einige Shell-Scripts dienen zusätzlich als Unterstützung und werden vom Daemon verwendet.

Der OpenSource GPS-Dienst `gpsd` wird verwendet, um die Daten des GPS-Empfängers periodisch auszulesen.

EasyCab Linux-Daemon

Das Script `/etc/init.d/easycabd` startet das Python-Script `/usr/local/python/easycab-daemon.py` als Linux-Daemon. Dies ist der eigentliche Kern der Applikation und stellt sicher dass

- der NFC-Leser korrekt funktioniert
- der GPS-Sensor korrekt funktioniert
- sobald der NFC-Leser einen Chip für Fahrer und Fahrzeug gelesen hat, die GPS-Daten per MQTT an den Broker gesendet werden.

Falls keine GPS-Daten empfangen werden können, wird das Script `/root/restart-gpsd.sh` aufgerufen, welches den `gpsd` Daemon neu startet. Dieses Script wurde erstellt, da der Aufruf von `service gpsd restart` den Dienst nicht sauber beendet und somit auch nicht korrekt neu gestartet werden kann.

Der `easycabd` Daemon kann über `service easycabd start|restart|stop` kontrolliert werden.

FlashConfig Linux-Daemon

Das Script `/etc/init.d/flashconfigd` startet das Python-Script `/usr/local/python/flashconfig-daemon.py` als Linux-Daemon. Dieses Script überprüft über die `Monitor` Klasse des Python-Pakets `pyUdev` auf Veränderungen der Anschlüsse des Raspberry Pi. Sobald ein Gerät entfernt oder hinzugefügt wurde, kann eine Methode aufgerufen werden.

In dieser Methode wird überprüft, ob auf einem angeschlossenen USB-Gerät eine Datei mit dem Dateinamen `config.json` gefunden wurde und der Inhalt der Konfigurations-Datei unter `/usr/local/python/config.json` wird entsprechend der gefundenen Datei angepasst. Die Datei wird nicht komplett ersetzt, sondern es wird nur nach Werten gesucht, welche nicht der bestehenden Konfiguration entsprechen und diese werden ersetzt. Somit müssen nicht alle Werte der Konfigurationsdatei angegeben werden.

RPi.GPIO

Um die LED und den Button anzusprechen, wird das Python Modul `RPi.GPIO` verwendet. Dieses ermöglicht den Zugriff auf die GPIO Schnittstelle. Die aktuelle Version 0.5.11 des Moduls unterstützt keine I2C, SPI oder seriellen Funktionalitäten. [11]

LED

Das Script `/usr/local/python/ledhandler.py` initialisiert und steuert die LED. In einer ersten Version steuerte das Script auch den Button und wurde als Thread vom `easycabd` Daemon gestartet. Weil der Thread mit zu hoher Verzögerung auf das Drücken des Buttons reagierte, wurden diese Funktionen in einen eigenen Daemon ausgelagert.

Button Daemon

Das Script `/usr/local/python/button-daemon.py` behandelt die Button-Ereignisse und wird beim Start des Tracking-Geräts als unabhängiger Daemon gestartet.

Beim Drücken des Buttons wird eine leere Datei `/block` ins Dateisystem des Tracking-Geräts geschrieben. Im `easycabd` Daemon wird in der Arbeits-Schleife in jedem Durchgang geprüft, ob diese Datei vorhanden ist. Falls ja, wird das Tracking deaktiviert und die LED ausgeschaltet.

Nach erneutem Drücken des Buttons wird der `easycabd` Daemon neu gestartet. Beim Initialisieren des Daemons wird die Datei `/block` entfernt falls vorhanden.

7.7.2 Server

Auf dem Server wird Mosquitto mit der aktivierten Option für Websockets installiert. Django wird verwendet, um die per MQTT empfangenen Dateien persistent in der Datenbank abzulegen und in Form von JSON-Objekten und HTML-Snippets erreichbar zu machen.

EasyCab Linux-Daemon

Die Scripts `/etc/init/easycab.sh` und `/etc/init/easycab.conf` starten das Python-Script `/usr/local/server/python/daemon.sh` als Linux-Daemon, welcher die Daten per MQTT ausliest und über Django in die Datenbank schreibt.

Der Daemon kann über `service easycabd start|restart|stop` kontrolliert werden.

Django

Im Verzeichnis `/usr/local/server/python/django/` befinden sich die Quelldateien für die Django-Webapplikation. Innerhalb dieses Verzeichnisses wurden wiederum Django-Apps in den Unterverzeichnissen `easycab`, `data` und `markers` abgelegt, wobei die App `easycab` ebenfalls Konfigurationen an die Apps `data` und `markers` beinhaltet / vererbt. Django-Apps sind jeweils so aufgebaut, dass die entsprechenden Datenmodelle in der Datei `models.py` definiert werden. In der Datei `urls.py` angegeben werden kann, über welche URLs die Applikation aufgerufen werden kann. In dieser Datei werden wiederum View-Klassen referenziert, welche in der Datei `views.py` definiert werden und bestimmen, was unter den definierten URLs angezeigt werden soll.

Die globalen Django Projekteinstellungen befinden sich in der Datei `/usr/local/server/python/django/easycab/settings.py`. Es handelt sich hierbei um folgende Angaben:

- Datenbank-Zugangsdaten und Art der Datenbank
- Verwendete Zusatz-Module oder Frameworks
- Lokalisierungsdaten (Sprachcode, Zeitzone)
- Konfiguration der statischen Verzeichnisse
- Netzwerk-Konfigurationen

Das Python-Script `/usr/local/server/python/django/manage.py` steuert die Django-Applikation und kann mit entsprechenden Parametern aufgerufen werden, also `python manage.py [parameter]`. Durch Aufrufen des Scripts ohne Parameter erhält man eine Auswahl aller möglichen Parameter.

Eine kurze Übersicht über die wichtigsten Parameter:

<code>runserver</code>	Startet den Django Server
<code>makemigrations</code>	Bereitet Anpassungen an der Datenbank-Struktur vor. Muss aufgerufen werden, wenn an den Daten-Modellen Anpassungen vorgenommen wurden.
<code>migrate</code>	Führt die mit <code>makemigrations</code> vorbereiteten Anpassungen an der Datenbank aus.
<code>collectstatic</code>	Kopiert die Dateien im Verzeichnis <code>/usr/local/server/python/django/data/static</code> in das Verzeichnis <code>/var/www/static</code> , welches vom Apache Webserver freigegeben wurde. Dieses Parameter wird verwendet, damit Anpassungen an Assets (Bildern, CSS- oder JavaScript-Dateien) auf der Website sichtbar werden.
<code>clean_markers</code>	Leert den Inhalt des Verzeichnisses <code>/usr/local/server/python/django/cache</code> , welches die gerenderten Marker-Bilder für die Karten-Anzeige beinhaltet. Dieses Parameter muss aufgerufen werden, wenn die Darstellung der Marker angepasst wird, damit diese anschliessend neu generiert werden können.

Google Maps Marker Icons

Für die dynamische Darstellung von Marker-Icons mit dem jeweiligen Taxi-Namen auf der Karte wurde das Modul `django-markers` verwendet. Dieses Modul ermöglicht es, auf einem Template-Bild einen dynamischen Text zu rendern, welcher als URL-Parameter mitgegeben werden kann. Die Font-Datei kann in den statischen Dateien hinterlegt und ebenfalls angegeben werden. Die entsprechenden Dateien des Moduls sind im Verzeichnis `/usr/local/server/python/django/markers` abgelegt.

Da Python allerdings kein Antialiasing zum Rendern von Texten kennt und somit der Text auf den erstellten Bildern sehr grob ausgegeben wurde, musste das Erzeugen des Bildes in der Datei `/usr/local/server/python/django/markers/models.py` erweitert werden. Der Antialiasing-Effekt konnte erreicht werden, indem das Bild zuerst in dreifacher Auflösung generiert und anschliessend mit aktiviertem Antialiasing gerendert wird.

Das so entstandene Bild weist sanftere Kanten auf, es stellte sich aber heraus, dass dadurch der Font zu fein gerendert wurde. Um dies zu beheben, wird der Font zweifach, um ein Pixel versetzt auf das Template-Hintergrund-Bild gerendert. Das so entstandene Bild kommt dem Standard-Font-Rendering des Browsers sehr nahe und ergibt ein befriedigendes Ergebnis.

Die erstellten Marker-Bilder werden im Verzeichnis `/usr/local/server/python/django/cache` gespeichert, damit diese nicht bei jedem Aufruf neu generiert werden müssen.

JavaScript

Bei der JavaScript-Implementation wurde jQuery eingesetzt, um die Manipulationen des HTML Markups und die Kommunikation per asynchronen Web-Requests zu vereinfachen und den Quelltext übersichtlich halten zu können. Ausserdem werden folgende jQuery-Module eingesetzt:

GeoComplete	Ein Modul zur Abfrage von Orten auf einer JavaScript Google Maps Integration. Wird für die Such-Funktionalitäten verwendet.
DatePair / TimePicker	Diese Module erleichtern die Datums- und Zeiteingabe auf Formularfeldern, indem Datums- oder Zeitdaten einfach per Mausklick ausgewählt werden können. Das Modul wird sowohl bei der Karten-Implementation für die Pfad-Suche der Fahrzeuge als auch in der Administrations-Oberfläche für den Positionsdaten Export verwendet.
Accordion	Dieses Modul ist Bestandteil des jQuery UI Frameworks und ermöglicht die Darstellung einer Liste von Elementen als sogenanntes Akkordion. Das heisst, dass bei der Wahl eines Eintrags dieser angezeigt und die anderen Einträge ausgeblendet werden. Diese Funktionalität wird für die Darstellung des Menüs für die Karten-Applikation verwendet.

Um Redundanzen zu vermeiden, wurden Projektübergreifende Daten und Funktionen ein `EasyCabUtil` Objekt definiert. Dieses Objekt wird sowohl für die Karten-Applikation als auch für den Positionsdaten-Export verwendet. Die Applikationskonfiguration aus der Datenbank wird ebenfalls beim Laden des Script-Objekts in diesem Objekt hinterlegt. Damit die Kartenapplikation nicht geladen wird, bevor die Konfigurationsdaten aus der Datenbank gelesen wurden, wird die `EasyCab` Klasse erst initialisiert, sobald das `EasyCabUtil` Objekt komplett geladen wurde.

Kartenapplikation

Die Kartenapplikation besteht aus einem Menü-Bereich, in welchem die einzelnen Fahrzeuge ausgegeben werden. Beim Klick auf ein Fahrzeug erscheinen die Details der letzten Messung sowie die Möglichkeit, den zurückgelegten Pfad des Fahrzeugs für eine frei definierbare Zeitspanne auf der Karte anzuzeigen.

Der Menü-Bereich wird beim Laden der Seite leer ausgegeben. Beim Laden der Seite werden mit einer asynchronen JavaScript-Anfrage auf den Server die aktuellen Menü-Einträge als HTML-Snippet geladen und eingefügt. Ausserdem werden die Marker für die Fahrzeuge auf der Karte angezeigt. Bei einer Positionsänderung eines Fahrzeugs wird der Kartenausschnitt ausserdem entsprechend angepasst, dass alle Fahrzeuge auf der Karte angezeigt werden.

Initial beim Laden der Karte wird ein JavaScript-Objekt mit den Stammdaten der Applikation (Fahrer, Fahrzeuge und Smartphones) durch eine asynchrone JavaScript-Anfrage vom Server gelesen. Diese Daten werden verwendet damit bei den Positionsmitteilungen per MQTT nur eine UUID der entsprechenden Komponenten mitgeteilt werden müssen.

Eine erweiterte Anforderung von `EasyCab` ist, dass Orte auf der Karte gesucht und optional auch Routen-Berechnungen vorgenommen werden können. Hierzu wurden zwei zusätzliche Eingabefelder für die Eingabe eines Start- und Zielortes im Menübereich eingefügt. Nachdem der Benutzer einen Ort eingegeben hat, wird dieser auf der Karte mit der Bezeichnung "Start" oder "Ziel" angezeigt. Falls sowohl ein Start- als auch ein Zielort angegeben wurden, wird eine Route zwischen den beiden Orten ausgegeben. Hierzu werden die Funktionalitäten der Google Maps API verwendet. Die Routen werden in einem über die Karte gelegten Container ausgegeben, welcher durch den Einsatz von jQuery

verschoben oder in der Grösse angepasst werden kann. Es ist auch möglich, die Route vom aktuell ausgewählten Fahrzeug zum Zielort zu berechnen, falls dieser definiert wurde.

Positionsdaten Export

Der Export der Positionsdaten basiert in erster Linie auf einer View in Django und wird mit JavaScript entsprechend erweitert, dass nur Kombinationen von Fahrer, Fahrzeug und Zeitspanne möglich sind, welche Ergebnisse liefern. Hierzu werden nach Anpassungen am Filterformular die entsprechenden Formularelemente als asynchrone JavaScript-Anfragen in Form von HTML-Snippets vom Server geladen.

7.7.3 Daten-Verschlüsselung

Für die Verschlüsselung der gesendeten Positionsdaten wird das Python-Paket pycrypto benötigt. Dies ermöglicht die einfache Verschlüsselung von Strings. Der Schlüssel zum Verschlüsseln der Daten kann in den Applikationskonfigurationen abgelegt werden und wird initial vom Client gelesen und für die Kommunikation der Positionsdaten verwendet. [9]

Um die Daten über JavaScript zu entschlüsseln, wird die CryptoJS Library verwendet. Dieses Projekt unterstützt eine Vielzahl verschiedener Verschlüsselungs-Algorithmen. [3]

Als Algorithmus wurde der Advanced Encryption Standard (AES) gewählt. [1]

8 Tests

In diesem Kapitel werden die Resultate der jeweiligen Tests dokumentiert. Die meisten Tests beziehen sich auf die Stories in Kapitel 5

8.1 Tracking-Gerät

8.1.1 Maintainer installiert neues Tracking-Gerät in Fahrzeug (Story 5.2.1)

Ausgangslage

Der Maintainer hat ein Tracking-Gerät, welches im Fahrzeug installiert werden soll. Der NFC-Chip zur Identifikation des Fahrzeugs wurde bereits installiert.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Maintainer schliesst das Tracking-Gerät am USB-Port des Fahrzeugs an und verbindet Smartphone.	LED "Smartphone" und "Internet" leuchten	✓	
2	Tracking-Gerät wird oberhalb des NFC-Chips platziert	LED "Fahrzeug" leuchtet	✓	
3	Tracking-Gerät empfängt GPS-Signal	LED "GPS" leuchtet	✓	

Fazit

Das Tracking-Gerät wurde installiert und die LED für "Fahrzeug" und "GPS" leuchten.

8.1.2 Fahrer meldet sich mit NFC-Chip an und wird getrackt (Story 5.1.1)

Ausgangslage

Der Fahrer betritt das Fahrzeug mit seinem zugewiesenen NFC-Chip. Das Tracking-Gerät wurde vom Maintainer bereits installiert und am Strom-Netz angeschlossen und erhält ein GPS-Signal. Somit leuchten die LED für "Fahrzeug" und "GPS" bereits.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Fahrer schliesst Smartphone an USB-Port des Tracking-Geräts an.	LED "Smartphone" leuchtet	✓	
2	Tethering-Verbindung mit Smartphone wurde hergestellt	LED "Internet" leuchtet	✓	
3	Fahrer meldet sich an.	LED "Fahrer" leuchtet	✓	
4	LED "Fahrzeug", "Smartphone", "GPS" und "Internet" leuchten	Fahrzeug wird auf Kartenapplikation angezeigt	✓	

Fazit

Alle LED leuchten und Fahrzeug wird auf der Kartenapplikation angezeigt.

8.1.3 Fahrer hat keinen NFC-Chip dabei (Story 5.1.2)

Ausgangslage

Der Fahrer betritt das Fahrzeug ohne NFC-Chip. Das Tracking-Gerät wurde vom Maintainer bereits installiert und am Strom-Netz angeschlossen und erhält ein GPS-Signal. Somit leuchten die LED für "Fahrzeug" und "GPS" bereits.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Fahrer schliesst Smartphone an USB-Port des Tracking-Geräts an.	LED "Smartphone" leuchtet	✓	
2	Tethering-Verbindung mit Smartphone wurde hergestellt	LED "Internet" leuchtet	✓	
3	LED "Fahrzeug", "Smartphone", "GPS" und "Internet" leuchten	Fahrzeug wird auf Kartenapplikation angezeigt	✓	
4	LED "Fahrer" leuchtet nicht	Auf der Karte wird beim Fahrzeug ein Auswahlfeld angezeigt, über welches der Disponent den Fahrer zuweisen kann	✓	
5	Fahrer wird vom Disponenten manuell zugewiesen	Auswahl wird in der Datenbank persistent gespeichert	✓	

Fazit

Das Tracking des Fahrzeugs funktioniert auch ohne dass das Tracking-Gerät per NFC-Tag des Fahrers aktiviert wird. Der Disponent kann über die Karte den Fahrer auswählen und dieser wird in der Datenbank entsprechend der aktuellen Sitzung des Fahrzeugs hinterlegt.

8.1.4 Tracking-Gerät aus- und anschliessend einschalten (Story 5.1.3)

Ausgangslage

Die LED des Tracking-Geräts für "Fahrzeug", "Smartphone", "GPS" und "Internet" leuchten und die momentane Position wird auf der Karte angezeigt.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Der Button des Tracking-Geräts wird einmalig gedrückt	Die LED leuchten nicht, das Gerät sendet keine Daten	✓	
2	Der Button des Tracking-Geräts wird erneut einmalig gedrückt	Das Gerät wird wieder aktiviert, die LED leuchten wieder und Daten werden gesendet	✓	

Fazit

Durch einmaliges Drücken des Buttons wird das Gerät deaktiviert. Allerdings muss mit einer kurzen Verzögerung gerechnet werden, da zweimaliges Drücken das Gerät zurücksetzt und der entsprechende Interval abgewartet werden muss, bevor das Gerät ausgeschaltet wird.

8.1.1.5 Gerät zurücksetzen (Story 5.1.1.4)

Ausgangslage

Das Tracking-Gerät ist an den Strom angeschlossen.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Der Button des Tracking-Geräts wird zweimal kurz nacheinander gedrückt	Das Tracking-Gerät wird zurückgesetzt. Alle LED leuchten nicht	✓	

Fazit

Der Test verlief erfolgreich und das Gerät wurde zurückgesetzt.

8.1.6 Pairing von Tracking-Gerät und Smartphone (Story 5.2.6)

Ausgangslage

Das Tracking-Gerät ist am Stromnetz angeschlossen. Dieser Test wird sowohl mit einem iPhone als auch mit einem Android- und einem Windows-Smartphone durchgeführt.

Test mit iPhone

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Fahrer schliesst Smartphone an USB-Port des Tracking-Geräts an.	LED "Smartphone" leuchtet	✓	
2	Tethering-Verbindung wurde hergestellt	LED "Internet" leuchtet	✓	

Test mit Android-Smartphone (Nexus 5, Android 6.0)

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Fahrer schliesst Smartphone an USB-Port des Tracking-Geräts an.	LED "Smartphone" leuchtet	✓	
2	Tethering-Verbindung wurde hergestellt	LED "Internet" leuchtet	✗	Da bei Android nicht persistent hinterlegt werden kann, dass Tethering per USB aktiviert ist, muss dies bei jedem Verbinden mit dem Smartphone in den Einstellungen erneut festgelegt werden.

Test mit Windows Smartphone

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Fahrer schliesst Smartphone an USB-Port des Tracking-Geräts an.	LED "Smartphone" leuchtet	✓	
2	Tethering-Verbindung wurde hergestellt	LED "Internet" leuchtet	✗	Die Möglichkeit, die Internet-Verbindung per USB freizugeben wurde bei Windows Phone mit der Version 8 entfernt.

Fazit

Das Smartphone kann mit dem Tracking-Gerät verbunden werden und erhält so eine Verbindung ins Internet.

Da EasyCab ausschliesslich iPhones einsetzt, war der Test grundsätzlich erfolgreich. Falls zu einem späteren Zeitpunkt auf Android, Windows oder ein anderes Smartphone-Betriebssystem gewechselt werden sollte, muss erneut evaluiert werden, mit welcher Technologie die Verbindung zum Smartphone hergestellt werden soll.

8.1.7 Konfiguration des Tracking-Geräts anpassen (Story 5.2.7)

Ausgangslage

Das Tracking-Gerät ist an den Strom angeschlossen.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Die Konfigurationsdatei wird erstellt, auf das Hauptverzeichnis eines Memory-Sticks geschrieben und am Tracking-Gerät angeschlossen. Nach 5 Sekunden wird der Memory-Stick wieder entfernt.	Die Konfiguration des Tracking-Geräts wurde entsprechend angepasst.	✓	

Fazit

Der Test verlief erfolgreich und die Konfiguration des Tracking-Geräts kann über einen Memory-Stick aktualisiert werden.

8.2 Google Maps Kartenapplikation

8.2.1 Nur aktive, inaktive oder alle Fahrzeuge anzeigen (Stories 5.3.1 und 5.3.2)

Ausgangslage

Die Kartenapplikation ist im Web-Browser geöffnet, es werden sowohl aktive Fahrzeuge als auch inaktive Fahrzeuge angezeigt (Standardverhalten)

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Über das Anzeigefilterfeld wird die Option "Aktive Fahrzeuge anzeigen" ausgewählt	Auf der Karte und im Menü-Bereich werden ausschließlich aktive Fahrzeuge eingeblendet	✓	
2	Über das Anzeigefilterfeld wird die Option "Inaktive Fahrzeuge anzeigen" ausgewählt	Auf der Karte und im Menübereich werden ausschließlich inaktive Fahrzeuge eingeblendet	✓	
3	Über das Anzeigefilterfeld wird die Option "Alle anzeigen" ausgewählt	Auf der Karte und im Menübereich werden alle, also sowohl aktive als auch inaktive Fahrzeuge eingeblendet	✓	

Fazit

Die Anzeige der Fahrzeuge kann über die Filterfunktion angepasst werden.

8.2.2 Zurückgelegte Strecke eines Fahrzeugs anzeigen (Story 5.3.3)

Ausgangslage

Die Kartenapplikation ist im Web-Browser geöffnet.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Das Fahrzeug wird im Menübereich ausgewählt	Die letzten Messdaten des Fahrzeugs sowie das Auswahlfeld "Pfad anzeigen" wird eingeblendet	✓	
2	Das Auswahlfeld "Pfad anzeigen" wird aktiviert	Ein zusätzliches Formular zur Auswahl des Start- und End-Zeitpunkts der Strecke wird eingeblendet	✓	
3	Start- und End-Zeitpunkt für die Routenberechnung wird gesetzt	Der Pfad der zurückgelegten Route zwischen Start- und End-Zeitpunkt wird auf der Karte eingeblendet	✓	
3	Das Auswahlfeld "Pfad anzeigen" wird deaktiviert	Der Pfad wird von der Karte entfernt	✓	

Fazit

Der Test verlief erfolgreich und die Strecke eines Fahrzeugs für einen bestimmten Zeitpunkt kann auf der Karte ausgegeben werden.

8.2.3 Ort auf Karte suchen (Story 5.3.4)

Ausgangslage

Die Kartenapplikation ist im Web-Browser geöffnet.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Über das Eingabefeld "Start-Ort eingeben" oder "Ziel-Ort eingeben" wird eine Adresse eingegeben	Per Auto-Complete Funktion werden entsprechend der Eingabe Vorschläge unterbreitet, welche aufgrund der Eingabe in Frage kommen	✓	
2a	Während der Eingabe wird die Eingabe-Taste gedrückt	Der erste Eintrag der Auswahlliste wird ins Eingabefeld übertragen	✓	
2b	Ein Eintrag der Auswahlliste wird angeklickt	Der ausgewählte Eintrag der Auswahlliste wird ins Eingabefeld übertragen	✓	
3	Das Eingabefeld wurde entsprechend Schritt 2a oder 2b mit einem Ort gefüllt	Der Ort wird auf der Karte als "Start" oder "Ziel" markiert	✓	

Fazit

Der Test verlief erfolgreich. Über die Such-Funktion kann ein Ort per Auto-Complete Funktion auf der Karte angezeigt werden.

8.2.4 Route berechnen (Story 5.3.5)

Ausgangslage

Die Kartenapplikation ist im Web-Browser geöffnet.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Im Eingabefeld "Start-Ort eingeben" wird "Biel/Bienne, Schweiz", im Eingabefeld "Ziel-Ort eingeben" wird "Zürich, Schweiz" eingegeben	Eine Route zwischen Start- und Zielort wird auf der Karte eingeblendet und die Route wird aufgrund von Google Map Informationen in einem Container auf der Karte angezeigt. Der Container mit der Karte kann frei über der Karte bewegt oder skaliert werden	✓	
3	Es werden im Container mehrere Routen angezeigt. Die zweite Route wird ausgewählt	Sowohl im Container als auch auf der Karte wird die Route entsprechend aktualisiert	✓	

Fazit

Der Test verlief erfolgreich. Eine Route zwischen Start- und Zielort kann errechnet werden. Falls vorhanden, können alternative Routen ausgewählt werden.

8.2.5 Route von Fahrzeug als Startpunkt berechnen (Story 5.3.6)

Ausgangslage

Die Kartenapplikation ist im Web-Browser geöffnet.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Das Fahrzeug wird im Menü oder auf der Karte ausgewählt	Die Karte zentriert sich auf das entsprechende Fahrzeug. Im Menüübersicht werden die letzten Messwerte angezeigt	✓	
2	Der gewünschte Zielort wird entweder im Feld "Start-Ort eingeben" oder "Ziel-Ort eingeben" eingefügt	Eine Route zwischen dem Fahrzeug und dem eingegebenen Start- oder Zielort wird auf der Karte eingeblendet und die Route wird aufgrund von Google Map Informationen in einem Container auf der Karte angezeigt. Der Container mit der Karte kann frei über der Karte bewegt oder skaliert werden	✓	
3	Es werden im Container mehrere Routen angezeigt. Die zweite Route wird ausgewählt	Sowohl im Container als auch auf der Karte wird die Route entsprechend aktualisiert	✓	

Fazit

Der Test verlief erfolgreich. Eine Route zwischen einem Fahrzeug und einem bestimmten Ort kann errechnet werden. Falls vorhanden, können alternative Routen ausgewählt werden.

8.3 Administrations-Oberfläche

8.3.1 Ein neues Fahrzeug hinzufügen, bearbeiten und löschen (Story 5.2.3)

Ausgangslage

Administrations-Oberfläche ist im Web-Browser geöffnet.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Link "Hinzufügen" beim Eintrag "Fahrzeug" wird geklickt	Maske zum Hinzufügen eines Fahrzeugs erscheint	✓	
2	Formular wird mit Test-Daten ausgefüllt	Fahrzeug wird gespeichert und die Übersicht der Fahrzeuge wird angezeigt.	✓	Validierung funktioniert: Alle Felder müssen ausgefüllt werden. Ausserdem werden Sonderzeichen korrekt abgespeichert.
3	Der erstellte Eintrag wird in der Liste ausgewählt indem der Name angeklickt wird	Bearbeitungs-Maske des Fahrzeugs öffnet sich	✓	
4	Die erfassten Daten werden bearbeitet und erneut gespeichert	Daten werden aktualisiert	✓	
5	Der erstellte Eintrag wird in der Liste über das Auswahl-Feld ausgewählt und entfernt	Das Fahrzeug verschwindet aus der Liste und wird aus der Datenbank gelöscht	✓	

Fazit

Der Test verlief erfolgreich. Fahrzeuge können erstellt, bearbeitet und gelöscht werden.

8.3.2 Einen neuen Fahrer hinzufügen, bearbeiten und löschen (Story 5.2.4)

Ausgangslage

Administrations-Oberfläche ist im Web-Browser geöffnet.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Link "Hinzufügen" beim Eintrag "Fahrer" wird geklickt	Maske zum Hinzufügen eines Fahrers erscheint	✓	
2	Formular wird mit Testdaten ausgefüllt	Fahrer wird gespeichert und die Übersicht der Fahrer wird angezeigt.	✓	Validierung funktioniert: Alle Felder müssen ausgefüllt werden. Ausserdem werden Sonderzeichen korrekt abgespeichert.
3	Der erstellte Eintrag wird in der Liste ausgewählt indem der Name angeklickt wird	Bearbeitungsmaske des Fahrers öffnet sich	✓	
4	Die erfassten Daten werden bearbeitet und erneut gespeichert	Daten werden aktualisiert	✓	
5	Der erstellte Eintrag wird in der Liste über das Auswahlfeld ausgewählt und entfernt	Der Fahrer verschwindet aus der Liste und wird aus der Datenbank gelöscht	✓	

Fazit

Der Test verlief erfolgreich. Fahrer können erstellt, bearbeitet und gelöscht werden.

8.3.3 Ein neues Smartphone hinzufügen, bearbeiten und löschen (Story 5.2.5)

Ausgangslage

Administrations-Oberfläche ist im Web-Browser geöffnet.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Link "Hinzufügen" beim Eintrag "Smartphones" wird geklickt	Maske zum Hinzufügen eines Smartphones erscheint	✓	
2	Formular wird mit Test-Daten ausgefüllt	Fahrzeug wird gespeichert und die Übersicht der Smartphones wird angezeigt.	✓	Validierung funktioniert: Alle Felder müssen ausgefüllt werden. Ausserdem werden Sonderzeichen korrekt abgespeichert.
3	Der erstellte Eintrag wird in der Liste ausgewählt indem der Name angeklickt wird	Bearbeitungs-Maske des Smartphones öffnet sich	✓	
4	Die erfassten Daten werden bearbeitet und erneut gespeichert	Daten werden aktualisiert	✓	
5	Der erstellte Eintrag wird in der Liste über das Auswahlfeld ausgewählt und entfernt	Das Smartphone verschwindet aus der Liste und wird aus der Datenbank gelöscht	✓	

Fazit

Der Test verlief erfolgreich. Smartphones können erstellt, bearbeitet und gelöscht werden.

8.3.4 Positions-Daten exportieren (Story 5.4.1)

Ausgangslage

Administrations-Oberfläche ist im Web-Browser geöffnet.

Test

Schritt	Aktion	Erwartung	Resultat	Kommentar
1	Button "Positions-Daten exportieren" wird geklickt	Formular zum Filtern der zu exportierenden Daten öffnet sich	✓	
2	Ein Fahrzeug wird ausgewählt	Liste der auswählbaren Fahrer wird reduziert auf jene Fahrer, welche mit dem ausgewählten Taxi Messwerte produzierten.	✓	
3	Ein weiteres Fahrzeug wird ausgewählt	Liste der auswählbaren Fahrer wird erweitert mit jenen Fahrern, welche mit dem ausgewählten Taxi Messwerte produzierten.	✓	
4	Ein Zeitraum wird definiert und per Klick auf "Alle Fahrer auswählen" werden die entsprechenden Fahrer zum Export hinzugefügt.	Alle angezeigten Fahrer sind ausgewählt.	✓	
5	Durch Klick auf "Export" wird der Export generiert.	Die Daten werden als CSV-Datei zum Download angeboten.	✓	

Fazit

Der Test verlief erfolgreich. Je nach Menge der Daten kann es aber mehrere Minuten dauern, bis die Daten aufbereitet sind und heruntergeladen werden können. Ausserdem kommt es bei der exportierten CSV-Datei zu Problemen mit Sonderzeichen, wenn die Datei in Microsoft Excel geöffnet wird. Da die Datei aber mit korrektem UTF8-Encoding gespeichert wird und in einem regulären Text-Editor korrekt geöffnet werden kann, liegt dies nicht an der exportierten Datei.

9 Handbuch

9.1 Tracking Gerät

9.1.1 Tracking beginnen

1. Das Tracking-Gerät über den NFC-Chip des Fahrzeuges stellen.
2. Das Tracking-Gerät per Micro-USB-Kabel an den USB-Anschluss des Fahrzeugs anschliessen.
3. Die LED "Fahrzeug" leuchtet.
4. Das Smartphone mit dem USB-Ladekabel an das Tracking-Gerät anschliessen.
5. Das Tracking-Gerät erhält eine Verbindung zum Internet und die LED "Internet" und "Smartphone" leuchten. Falls nur die LED "Smartphone" leuchtet, muss auf dem Smartphone sichergestellt werden, dass eine Verbindung zum Internet besteht.
6. Den NFC-Chip über die Oberseite des Gerätes halten.
7. Die LED "Fahrer" leuchtet.
8. Wenn die LED "GPS" leuchtet, erhält das Gerät GPS-Daten und die aktuellen Positionsdaten werden übermittelt.

9.1.2 Tracking beenden

Das Tracking des Fahrzeugs kann beendet werden, indem entweder der Button auf dem Tracking-Gerät einmalig gedrückt wird oder indem die USB-Verbindung zum Smartphone getrennt wird und somit keine Internet-Verbindung mehr aufgebaut werden kann.

9.2 Karten-Applikation

Die Web-Oberfläche wird in die Kartenansicht auf der linken und den Menübereich rechts unterteilt. Auf der Kartenansicht werden die zuletzt mitgeteilten Positionen der Fahrzeuge oder Routen abgebildet, während der Menübereich zur Interaktion mit der Applikation dient.

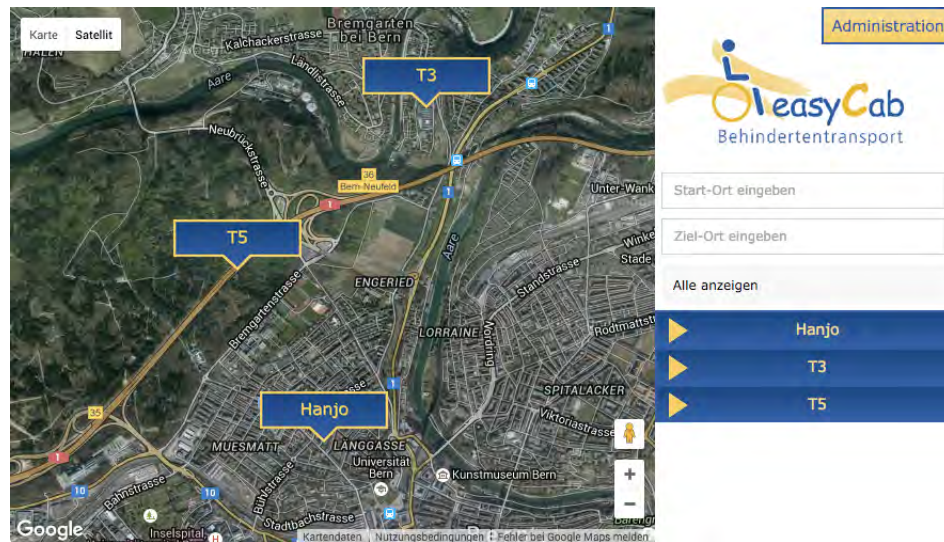


Abbildung 9.1: Initiale Ansicht der Kartenapplikation

9.2.1 Fahrzeugdetails

Um Details eines Fahrzeugs anzuzeigen, kann entweder das Marker-Symbol des Fahrzeugs auf der Karte oder der Eintrag des Fahrzeugs im Menübereich angeklickt werden. Die Karte zentriert sich auf das Fahrzeug und die Details des Fahrzeugs werden im Menü-Bereich eingeblendet.

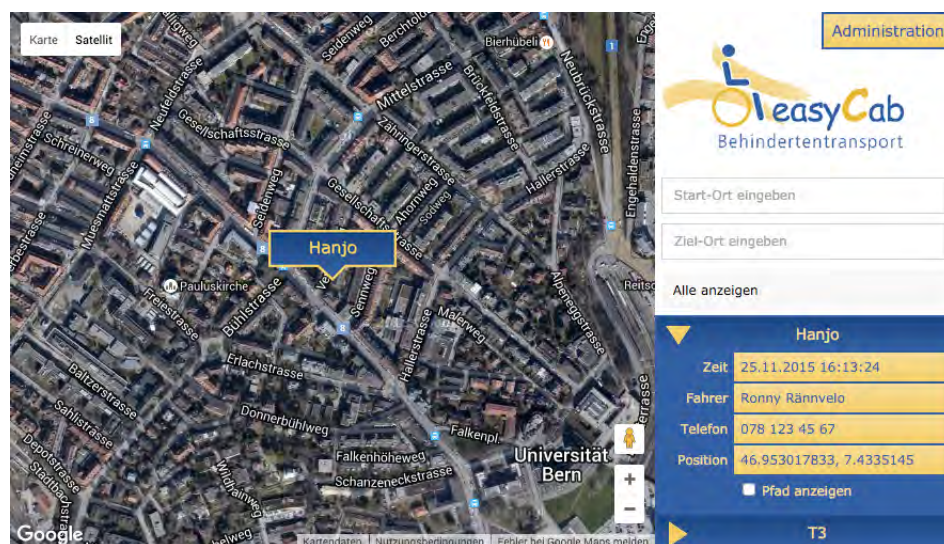


Abbildung 9.2: Detailansicht eines Fahrzeugs

9.2.2 Zurückgelegte Route eines Fahrzeugs

Nachdem die Fahrzeugdetails angezeigt werden, kann das Kontrollkästchen "Pfad anzeigen" aktiviert werden. Es erscheinen weitere Eingabefelder über welche der gewünschte Zeitraum eingeschränkt werden kann.

Nachdem ein Zeitraum für die Route angegeben wurde, wird die entsprechende Route des Fahrzeugs auf der Karte angezeigt.

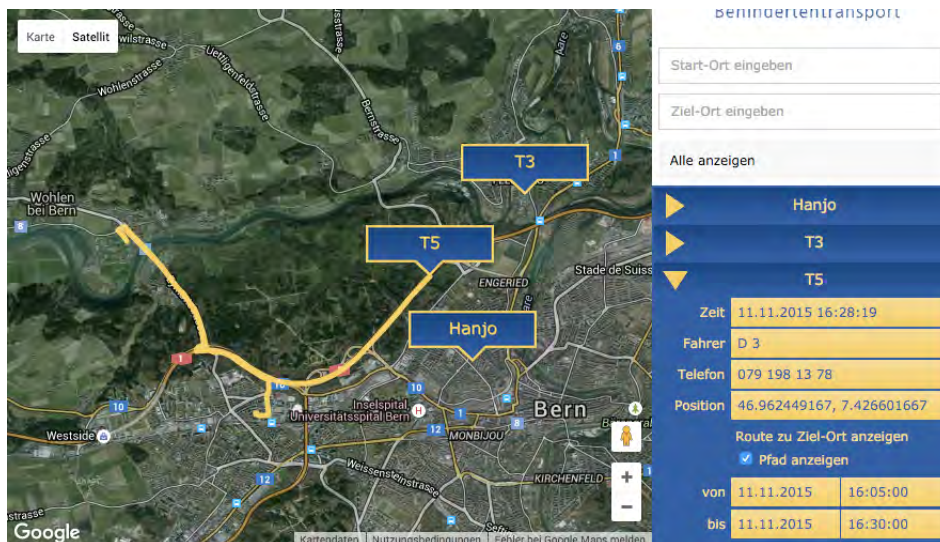


Abbildung 9.3: Zurückgelegte Route eines Fahrzeugs

9.2.3 Fahrzeugübersicht und Filter

Sowohl auf der Karte wie im Menübereich werden alle Fahrzeuge aufgelistet. Durch das Filter-Auswahlmenü kann die Ansicht eingeschränkt werden, dass entweder nur aktive oder nur inaktive Fahrzeuge angezeigt werden.

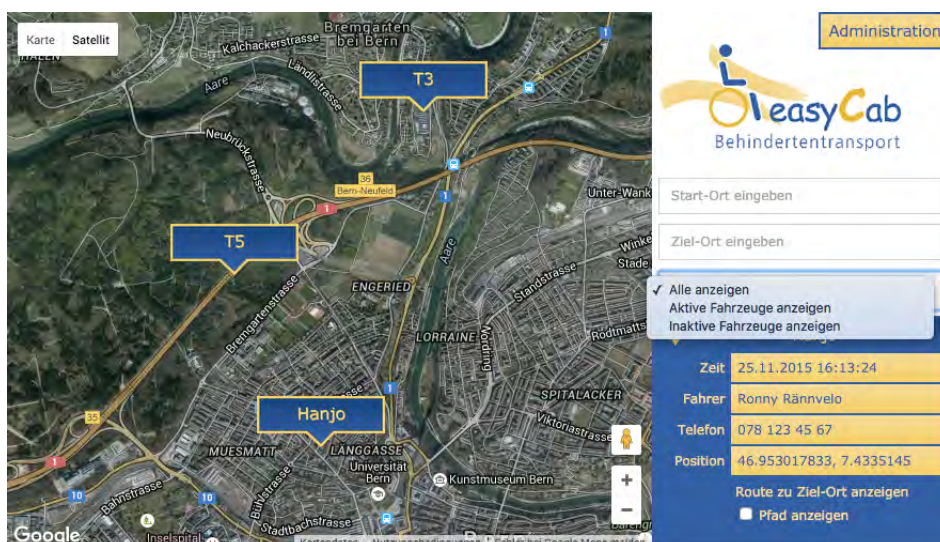


Abbildung 9.4: Karten Filterfunktion

9.2.4 Standort suchen

Über die beiden Eingabefelder “Start-Ort eingeben” oder “Ziel-Ort eingeben” kann eine entsprechende Adresse auf der Karte gesucht werden. Bei der Eingabe werden automatisch zur momentanen Eingabe passende Adressen aufgelistet, welche durch Klick auf den entsprechenden Eintrag ausgewählt werden und im Eingabefeld eingetragen werden können. Wird die Eingabetaste während der Eingabe gedrückt, wird der erste Eintrag der Liste ins Eingabefeld eingetragen.



Abbildung 9.5: Auto-Complete Funktion

Nachdem die Adresse eingetragen wurde, wird diese auf der Karte entweder als “Start” oder “Ziel” angezeigt.



Abbildung 9.6: Anzeige eines gesuchten Standorts

9.2.5 Route zwischen zwei Standorten anzeigen

Eine Route zwischen zwei Standorten wird automatisch berechnet, sobald sowohl ein Start- und Zielort eingegeben wurde.

Ausserdem erscheint die Routen-Beschreibung in einem Overlay-Element, welches frei auf dem Browser-Fenster bewegt und skaliert werden kann. Falls mehrere Routen-Vorschläge vorhanden sind, können diese im Overlay-Element ausgewählt werden.

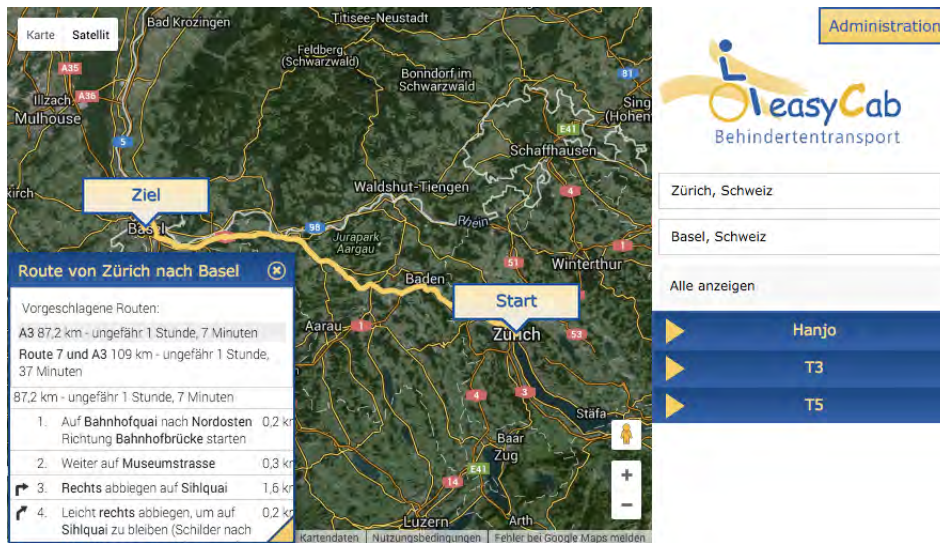


Abbildung 9.7: Route zwischen zwei Standorten

9.2.6 Route von Fahrzeug zu einem Standort anzeigen

Um die Route von einem Fahrzeug zu einem bestimmten Standort anzuzeigen, muss zuerst die Ziel-Adresse im Feld "Ziel-Ort eingeben" eingegeben werden. Anschliessend erscheint in den Fahrzeugdetails ein Link "Route zu Ziel-Ort anzeigen" eingeblendet, über welchen eine Route zwischen dem Fahrzeug und dem Zielort auf der Karte angezeigt wird.

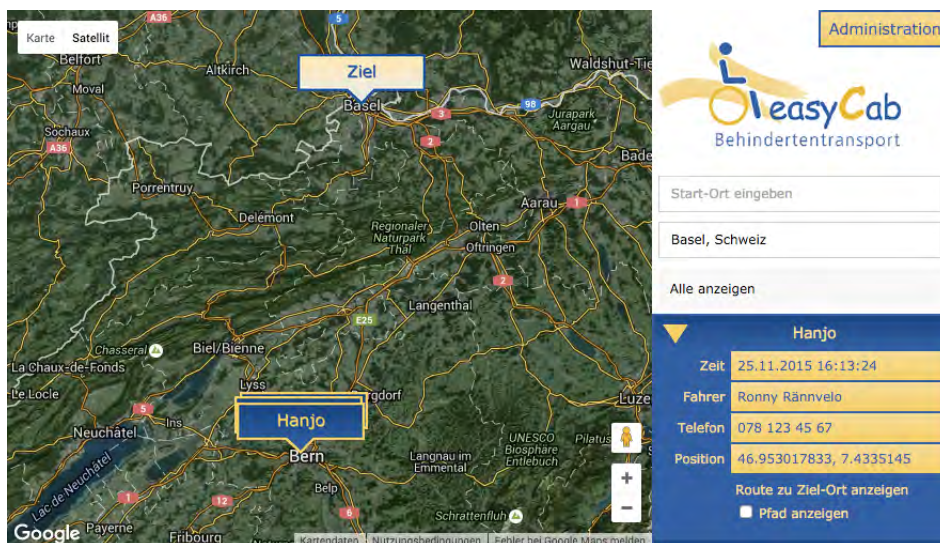


Abbildung 9.8: Ansicht nachdem ein Ziel-Standort gesucht wurde

Ausserdem erscheint die Routenbeschreibung in einem Overlay-Element, welches frei auf dem Browser-Fenster bewegt und skaliert werden kann. Falls mehrere Routenvorschläge vorhanden sind, können diese im Overlay-Element ausgewählt werden.



Abbildung 9.9: Route zwischen einem Fahrzeug und Standort

9.3 Administrations-Oberfläche

Über die Administrations-Oberfläche können gewisse Applikationseinstellungen und die Stammdaten (Fahrer, Smartphones und Fahrzeuge) bearbeitet werden.

Die Administrations-Oberfläche kann über den Button “Administration“ auf der Kartenapplikation erreicht werden.

Um auf die Administrations-Oberfläche zugreifen zu können, muss zuerst über die Kommando-Zeile auf dem Server mit dem Befehl `django-admin createsuperuser` ein Superuser-Konto erstellt worden sein.

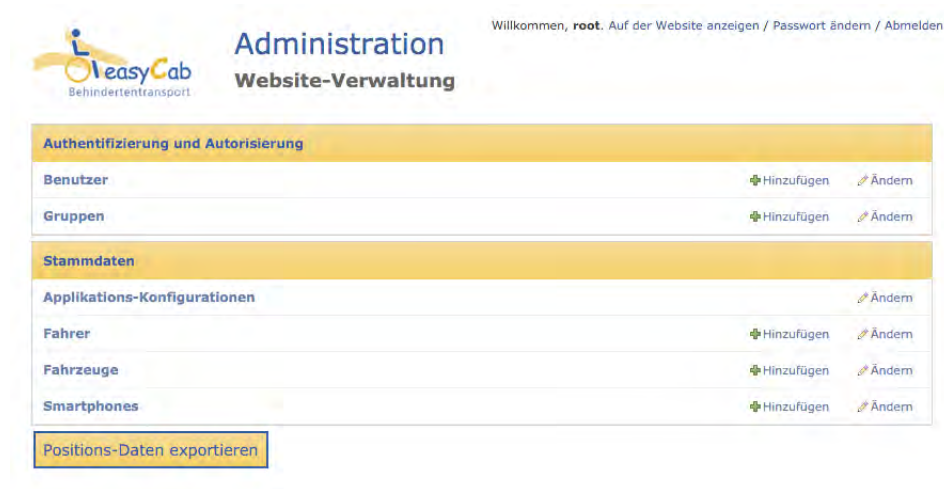


Abbildung 9.10: Initiale Ansicht der Administrations-Oberfläche

9.3.1 Applikations-Konfigurationen bearbeiten

Durch Klicken des “Applikations-Konfigurationen“ Links können folgende Konfigurationen angepasst werden:

- Aktualisierungs-Intervall in Sekunden
- Tage, nach welchen Positionsdaten gelöscht werden

- Schlüsselwort für die Datenübertragung
- Session-Timeout in Sekunden

The screenshot shows the 'Administration' page for 'Applikations-Konfiguration ändern'. The breadcrumb trail is 'Start > Stammdaten > Applikations-Konfigurationen > AppConfig object'. The page contains two input fields: 'Aktualisierungs-Intervall in Sekunden' with the value '5' and 'Tage, nach welchen Positions-Daten gelöscht werden:' with the value '30'. At the bottom right, there are two buttons: 'Sichern und weiter bearbeiten' and 'Sichern'.

Abbildung 9.11: Ansicht "Applikations-Konfigurationen bearbeiten"

9.3.2 Fahrer, Smartphone oder Fahrzeug hinzufügen

Ein neuer Fahrer, ein Smartphone oder Fahrzeug kann über den jeweiligen "Hinzufügen" Link erfasst und hinzugefügt werden.

The screenshot shows the 'Administration' page for 'Fahrer hinzufügen'. The breadcrumb trail is 'Start > Stammdaten > Fahrer > Hinzufügen Fahrer'. The form contains three input fields: 'Firstname:', 'Lastname:', and 'Token:'. At the bottom right, there are three buttons: 'Sichern und neu hinzufügen', 'Sichern und weiter bearbeiten', and 'Sichern'.

Abbildung 9.12: Hinzufügen eines neuen Stammdaten-Elements

9.3.3 Fahrer, Smartphone oder Fahrzeug bearbeiten

Zum Bearbeiten eines Fahrers, Smartphones oder Fahrzeugs kann über den "Ändern" Link die Liste aller erfassten Einträge geöffnet werden. Anschliessend kann der gewünschte Eintrag ausgewählt und angepasst werden.

The screenshot shows the 'Administration' page for 'Fahrer ändern'. The breadcrumb trail is 'Start > Stammdaten > Fahrer > Vicco von Bülow'. The form contains three input fields: 'Firstname:' with the value 'Vicco', 'Lastname:' with the value 'von Bülow', and 'Token:' with the value '04:D8:2A:92:9E:33:80'. At the bottom left, there is a red 'Löschen' button. At the bottom right, there are three buttons: 'Sichern und neu hinzufügen', 'Sichern und weiter bearbeiten', and 'Sichern'.

Abbildung 9.13: Bearbeiten eines Stammdaten-Elements

9.3.4 Fahrer, Smartphone oder Fahrzeug entfernen

Zum Entfernen eines oder mehrerer Fahrer, Smartphones oder Fahrzeuge kann über den “Ändern” Link die Liste aller erfassten Einträge geöffnet werden. Anschliessend können die zu löschenden Einträge und die Aktion “löschen” ausgewählt werden. Durch Klick auf den “Ausführen” Button werden die Einträge gelöscht.



Abbildung 9.14: Entfernen von Stammdaten-Elementen

9.3.5 Positions-Daten exportieren

Durch Klicken auf den Button “Positions-Daten exportieren” öffnet sich ein Overlay-Fenster mit einem Formular, über welches die Kriterien für den Export definiert werden können. So kann der Export nach Fahrern, Fahrzeugen und Datum gefiltert werden. Wenn keine Optionen ausgewählt werden, werden keine Daten exportiert. Je nach Auswahl werden die Auswahlmöglichkeiten entsprechend angepasst, dass nur Optionen verfügbar sind, für welche Positionsdaten in der Datenbank hinterlegt sind.



Abbildung 9.15: Overlay Formular “Positions-Daten exportieren”






9.4 Installation



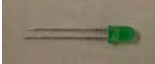





9.4.1 Tracking Gerät

Werkzeug und Verbrauchsmaterial

- USB-Adapter für SD-Karte
- Lötstation
- Löthalterung
- Abisolierzange
- kleiner Seitenschneider
- Lötzinn

Benötigte Bauteile

Anzahl	Bauteil	
1	Raspberry Pi B+ oder 2 	
1	Micro-SD Karte 	8 GB
1	GPS Dongle 	
1	Tinkerforge Masterbrick 	
1	Tinkerforge NFC Brick 	
1	Tinkerforge Kabel 	Länge 15 cm
1	USB-Kabel A zu Mini-B 	

Anzahl	Bauteil	
1	USB-Kabel A zu Micro-A 	
1	Experimentierplatine 	7.2 × 2.8 cm 10 × 26 Löcher Rastermass 2.54 mm Löchgrösse 1 mm
5	LED Grün 	Durchmesser 5 mm Durchlassspannung 2.2 V
1	Push-Button 	
5	Kohleschicht-Widerstand 220 Ω 	Nennleistung 250 mW
1	Kohleschicht-Widerstand 10 k Ω 	Nennleistung 250 mW
5	Draht-Jumperkabel 	Länge 7.5 mm
8	Jumperkabel 	Female Anschluss 15 cm

Button und LED

Für die Bau einer Platine mit Button und LED sind grundlegende Kenntnisse des Lötens von Elektronikbauteilen erforderlich. Diese Anleitung wird nicht auf die benötigten Techniken eingehen.

1. Bauteile vorbereiten:
Beine der Widerstände biegen

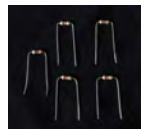


Abbildung 9.16: gebogene Widerstände

Kabel auf die richtige Länge zuschneiden und das Ende auf ca. 5mm abisolieren.
Abisolierte Kabelenden verlöten.

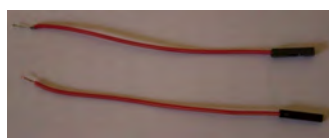


Abbildung 9.17: Kabel abisoliert und verlötet

Falls die Lötstellen der Platine durch Leiterbahnen verbunden sind, diese so unterbrechen, dass keine Verbindung von Quelle zu Masse möglich ist.

- LED an den grün markierten Stellen einsetzen. **K**athode und **A**node beachten. Die Anode ist im Normalfall das längere Beine. Zur Sicherheit die Spezifikation der LED lesen.

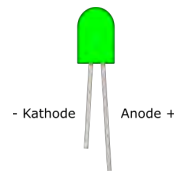


Abbildung 9.18: LED

Die LED anlöten und die überstehenden Beine abschneiden.

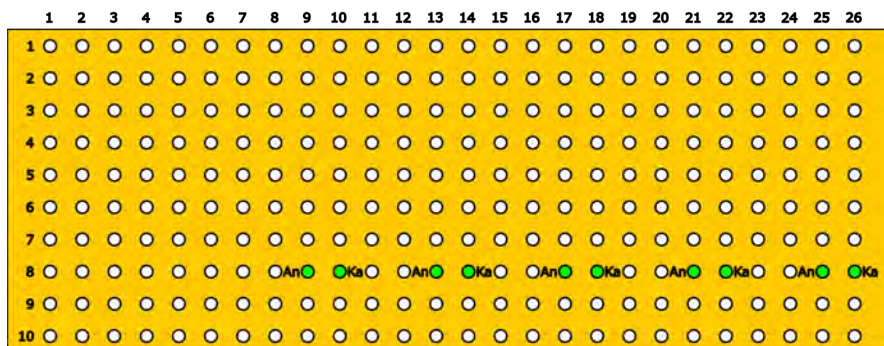


Abbildung 9.19: Platine: Schritt 1

- Die 220Ω Widerstände einsetzen und anlöten.
Die überstehenden Beine abschneiden und Verbindung zur Anode der LED herstellen.

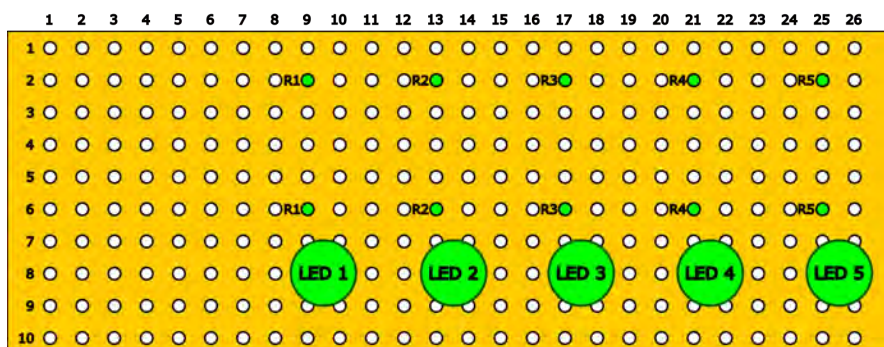


Abbildung 9.20: Platine: Schritt 2

4. 10k Ω Widerstand einsetzen und anlöten.
Die überstehenden Beine abschneiden.

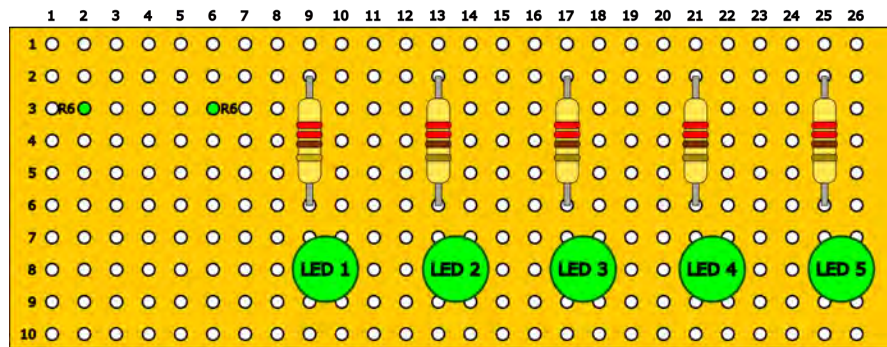


Abbildung 9.21: Platine: Schritt 3

5. Button einsetzen und anlöten.

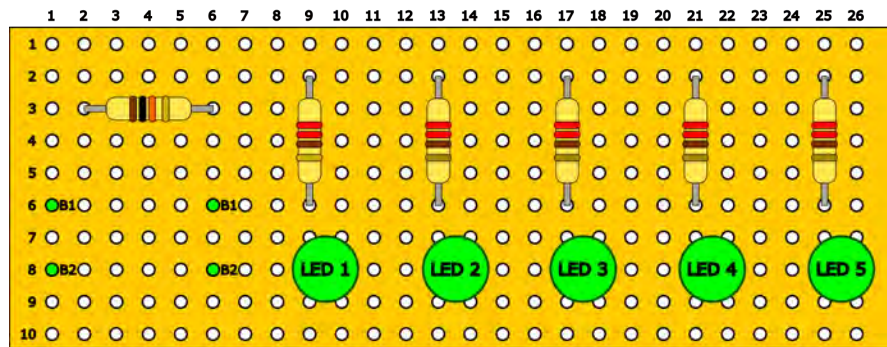


Abbildung 9.22: Platine: Schritt 4

6. Jumper einsetzen und löten.
Die überstehenden Beine abschneiden und Verbindung zu den Kathoden und dem nächsten Jumper herstellen.

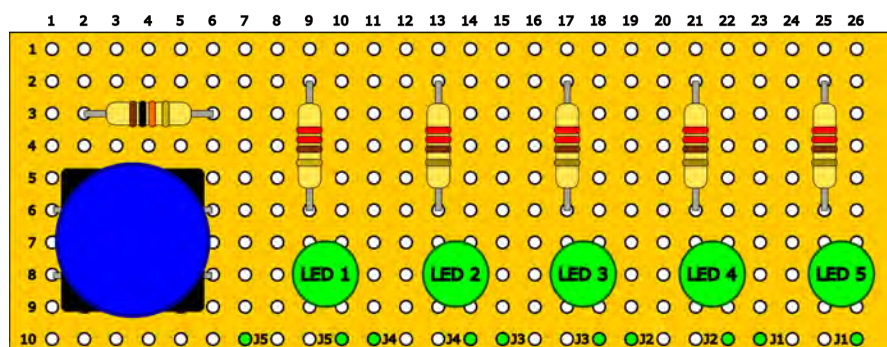


Abbildung 9.23: Platine: Schritt 5

7. GPIO-Kabel einsetzen und lten.

Kabel 1 bis 6 mit den Widerstnden verbinden.

Kabel 7 mit dem Widerstand und dem Bein des Buttons verbinden.

Kabel 8 mit dem Jumper und dem Bein des Buttons verbinden.

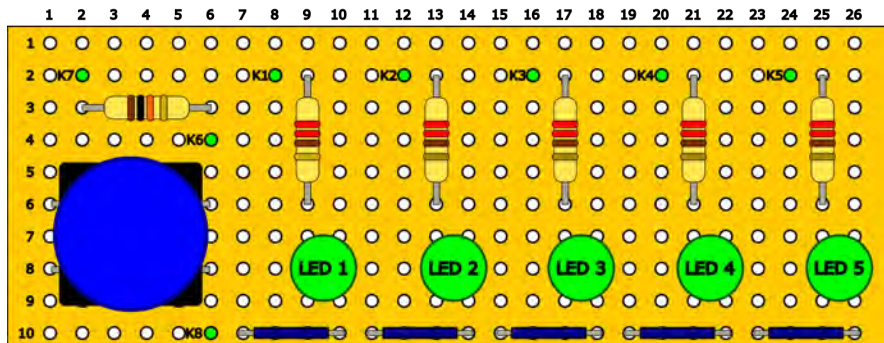


Abbildung 9.24: Platine: Schritt 6

8. Fertig

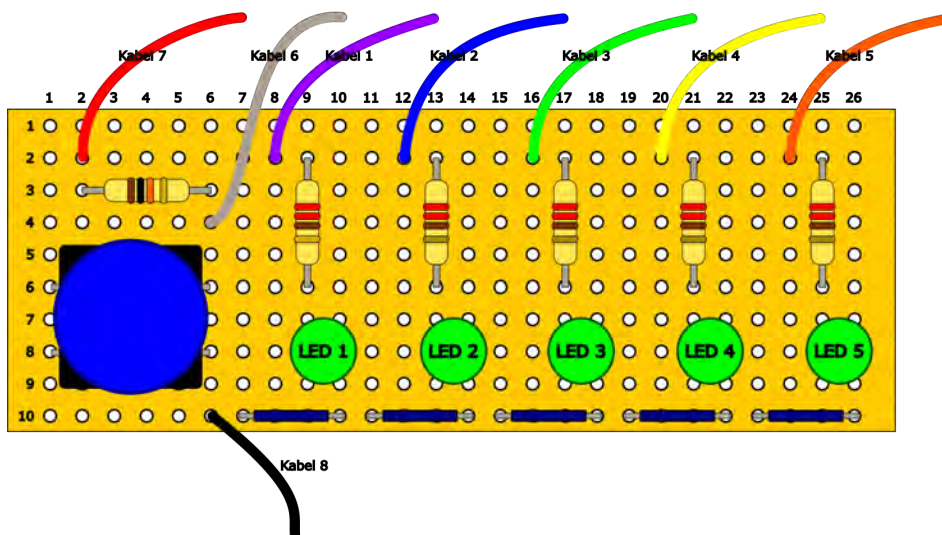


Abbildung 9.25: Fertige Platine

Software

Die Image-Datei `easycabian.img` enthlt das Betriebssystem und alle bentigte Software fr das Tracking-Gert. Die Micro-SD Karte mit einem geeigneten Adapter an den PC anschliessen und die Image-Datei mit einer entsprechenden Software auf die SD-Karte schreiben. Dann muss nur noch die Karte in den SD-Karten Slot des Raspberry Pi eingeschoben werden.

Das Kapitel 10 beschreibt die Neuinstallation des Betriebssystems und der Software.

Gehuse

Das Gehuse kann mit gngigen 3D Druckern hergestellt werden. Die dafr bentigten Vorlagen sind die Dateien `case-bottom.stl` fr den unteren Teil und `case-top.stl` fr den oberen Teil des Gehuses. Die beiden Dateien befinden sich im Verzeichnis `3d-model` im EasyCabApp Git-Repository[4]. Fr den genauen Druckvorgang lesen Sie die Dokumentation des Herstellers.

Falls kein 3D-Drucker verfügbar ist, können die Dateien auch mit einem online 3D-Druck Dienst wie zum Beispiel www.teil3.ch gedruckt werden.

Endmontage

1. Raspberry Pi in das untere Gehäuseteil einsetzen.
2. LED-Platine montieren.
3. Kabel auf die Pins setzen.

- Kabel 1 auf Pin 18 (GPIO 24)
- Kabel 2 auf Pin 16 (GPIO 23)
- Kabel 3 auf Pin 15 (GPIO 22)
- Kabel 4 auf Pin 13 (GPIO 27)
- Kabel 5 auf Pin 11 (GPIO 17)
- Kabel 6 auf Pin 22 (GPIO 25)
- Kabel 7 auf Pin 17 (3.3V Power)
- Kabel 8 auf Pin 20 (Ground)



Abbildung 9.26: GPIO Layout Raspberry Pi B+ und 2

4. Tinkerforge NFC Leser und Master-Brick mit Tinkerforge Kabel verbinden.
5. Den MasterBrick per USB-Kabel mit dem Raspberry Pi verbinden.
6. Zuerst MasterBrick in das obere Gehäuseteil einsetzen, anschliessend NFC Leser einklinken.
7. Die beiden Gehäuseteile zusammenstecken.
8. GPS-Dongle einsetzen.
9. Micro-USB-Kabel anschliessen.

9.4.2 Server

Die Installation des Servers wird in der Installationsanleitung unter Kapitel 10.2 behandelt.

9.5 Konfiguration des Tracking-Geräts anpassen

Die Konfiguration des Tracking-Geräts kann mit einem USB Memory-Stick vorgenommen werden.

Hierzu wird zuerst eine Text-Datei mit dem Dateinamen `config.json` und folgender Struktur erstellt:

```
{  
  "mqtt_url": "46.101.17.239",  
  "mqtt_port": 1883,  
  "python_web_path": "/data"  
}
```

Titel	Datentyp	Beschreibung
mqtt_url	String	URL des Servers, entweder IP-Adresse oder Domain-Name
mqtt_port	Ganzzahl	Port, welcher auf dem Server in der Mosquitto-Konfiguration definiert wurde
python_web_path	String	Pfad, unter welchem auf dem Server die Django-Umgebung erreicht werden kann / unter welchem die Karten-Applikation aufgerufen wird. Diese Einstellung hängt mit der Apache / WSGI-Konfiguration des Servers zusammen.

Tabelle 9.2: Aufschlüsselung der Datei `config.json`

Diese Datei wird in das Hauptverzeichnis des Memory-Sticks kopiert. Anschliessend muss der Memory-Stick an einem freien USB-Anschluss des Tracking-Geräts angeschlossen werden. Nach 5 Sekunden kann der Memory-Stick wieder entfernt werden und die Daten wurden auf dem Tracking-Gerät hinterlegt. Nach einem Neustart des Tracking-Geräts sind die Einstellungen aktiv.

10 Installation

10.1 Raspberry Pi

Die Standardinstallation kann direkt mit der `easycab.img`-Datei auf eine SD-Karte geladen werden.

Diese Anleitung soll aufzeigen, wie die Image-Datei entstanden ist und nötigenfalls rekonstruiert werden kann und betrifft daher nur den Akteur "Maintainer".

10.1.1 Betriebssystem

Das Minibian OS von <http://sourceforge.net/projects/minibian/> herunterladen.

Die Seite "RPi Easy SD Card Setup"[10] erläutert, wie die ISO-Datei auf eine Micro-SD-Karte transferiert werden kann.

10.1.2 Zugriff per SSH

Damit die weiteren Schritte durchgeführt werden können, muss eine Verbindung als Benutzer `root` zur Konsole des Raspberry Pi aufgebaut werden. Nun muss das Raspberry Pi per Ethernet an ein Netzwerk angeschlossen werden.

Das Passwort für den `root`-Benutzer lautet `raspberrypi`. Es ist nun möglich, sich per SSH mit dem Raspberry Pi zu verbinden:

```
ssh root@minibian
```

10.1.3 Anpassen der primären Partition

Nach dem erfolgreichen Login sollte zuerst die Grösse der Root-Partition der Grösse der SD-Karte angepasst werden.

Dies kann mit `fdisk` gelöst werden:

```
$ fdisk /dev/mmcblk0
```

Anschliessend folgende Befehle eingeben:

- `p` - Momentanen Anfangspunkt der primären Partition notieren
- `d, 2` - Primäre Partition löschen
- `n, p, 2` - Neue primäre Partition erstellen. Die einzugebenden Werte können meist bestätigt werden. Falls sich der Anfangspunkt von dem vorher notierten Anfangspunkt unterscheiden sollte, muss dieser geändert werden.
- `w` - Schreiben der Partitionstabelle

Nun muss das System neu gestartet werden:

```
$ shutdown -r now
```

Nach dem Neustart muss der Befehl `resize2fs` ausgeführt werden, damit das Dateisystem an die neue Grösse des Geräts angepasst wird:

```
$ resize2fs /dev/mmcblk0p2
```

10.1.4 Benötigte Software installieren

Vor der Software-Installation müssen die Paket-Quellen aktualisiert werden:

```
$ apt-get update
$ apt-get upgrade
```

Installation der benötigten Debian-Pakete:

```
$ apt-get install gpsd gpsd-clients python-gps libusb-1.0-0 libudev0 pm-utils
python-qt4 python-qt4-gl python-opengl python-serial python-pip python-rpi.
gpio gvfs ipheth-utils libimobiledevice-utils gvfs-backends gvfs-bin gvfs-
fuse ifuse python-rpi.gpio python-dev
```

Für Python benötigte Pakete per PIP installieren:

```
$ pip install paho-mqtt tinkerforge python-daemon pycrypto pycrypto-on-pypi
ecdsa
```

Tinkerforge brickd Treiber und brickv Viewer installieren:

```
$ wget http://download.tinkerforge.com/tools/brickd/linux/
brickd_linux_latest_armhf.deb
$ dpkg -i brickd_linux_latest_armhf.deb
$ wget http://download.tinkerforge.com/tools/brickv/linux/brickv_linux_latest.
deb
$ dpkg -i brickv_linux_latest.deb
```

Anschliessend können per sftp (Port 22) die benötigten Scripts aus dem EasyCabApp Git-Repository[4] abgelegt werden.

Aus dem Verzeichnis /raspbi/bash:

- /root/restart-gpsd.sh
- /etc/init.d/buttond
- /etc/init.d/easycabd
- /etc/init.d/flashconfigd
- /etc/network/interfaces
- /lib/udev/iphoneconnect
- /lib/udev/rules.d/90-iphone-tether.rules

Aus dem Verzeichnis /raspbi/python:

- /usr/local/python/button-daemon.py
- /usr/local/python/easycab-daemon.py
- /usr/local/python/flashconfig-daemon.py
- /usr/local/python/ledhandler.py
- /usr/local/python/config.json

Raspberry Pi für USB-Tethering vorbereiten, Berechtigungen anpassen und easycabd, buttond und flashconfigd Daemons bei Systemstart ausführen:

```

$ chmod 755 /lib/udev/iphoneconnect
$ mkdir /media/iPhone
$ chmod +x -R /root/*.sh
$ chmod 755 /etc/init.d/easycabd
$ chmod 755 /etc/init.d/buttond
$ chmod 755 /etc/init.d/flashconfigd
$ chmod 755 /lib/udev/iphoneconnect
$ mkdir /var/log/easycabd
$ mkdir /var/log/buttond
$ mkdir /var/log/flashconfigd
$ update-rc.d easycabd defaults
$ update-rc.d buttond defaults
$ update-rc.d flashconfigd defaults

```

10.2 Server

Für die Server-Installation wird vorausgesetzt, dass zuerst eine Verbindung per SSH mit dem entsprechenden Server hergestellt wurde.

10.2.1 Zeitzone anpassen

Damit die Zeit jeweils korrekt übermittelt wird, muss zuerst die Zeitzone entsprechend korrigiert werden:

```
$ dpkg-reconfigure tzdata
```

10.2.2 Python / PIP

Um Python mit den Entwickler-Tools und dem PIP Paket-Installer vorzubereiten, müssen folgende Befehle über die Konsole ausgeführt werden:

```

$ apt-get update
$ apt-get install python-pip python-dev build-essential uid-dev xsltproc
  docbook-xsl git libcurl4-gnutls-dev libexpat1-dev gettext libz-dev libssl-
  dev libc-ares-dev cmake
% $ pip install --upgrade pip
% pip install --upgrade virtualenv

```

10.2.3 Apache & MySQL

Zur Darstellung und Speicherung der Daten wird sowohl der Apache Webserver als auch ein MySQL-Server benötigt. Diese können folgendermassen installiert werden:

```

$ apt-get install apache2 apache2-mpm-prefork apache2-utils libexpat1 ssl-cert
  mysql-server libapache2-mod-auth-mysql python-mysqldb libapache2-mod-wsgi
$ service mysql start
$ mysql_secure_installation
$ service mysql restart
$ service apache2 restart

```

10.2.4 Mosquitto

Zuerst muss libwebsockets installiert werden:

```
$ mkdir /root/mosquitto/
$ cd /root/mosquitto/
$ wget http://git.warmcat.com/cgi-bin/cgit/libwebsockets/snapshot/libwebsockets-1.22-chrome26-firefox18.tar.gz
$ tar -xzf libwebsockets-1.22-chrome26-firefox18.tar.gz
$ apt-get install autoconf
$ cd libwebsockets-1.22-chrome26-firefox18/
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
```

Mosquitto muss manuell mit Websocket-Unterstützung kompiliert werden. Daher wird Mosquitto in der Version 1.4 zuerst vom Git Repository geladen:

```
$ cd /root/mosquitto/
$ git clone https://git.eclipse.org/r/mosquitto/org.eclipse.mosquitto
$ cd org.eclipse.mosquitto/
$ git checkout origin/master
```

Anschliessend muss die Datei `config.mk` angepasst werden. Hierbei ist sicher zu stellen, dass die Websockets-Option auf `yes` gesetzt wird:

```
WITH_WEBSOCKETS=yes
```

Somit kann Mosquitto folgendermassen kompiliert und installiert werden:

```
$ make
$ make test
$ make install
$ useradd -r -m -d /var/lib/mosquitto -s /usr/sbin/nologin -g nogroup mosquitto
$ touch /etc/mosquitto/mosquitto.conf
$ /usr/local/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

10.2.5 Python Module

Für die Darstellung der Daten auf der Website sowie für die Anbindung an die MySQL-Datenbank müssen zusätzliche Python Module installiert werden:

```
$ pip install --allow-external mysql-connector-python python-daemon pycrypto
numpy mosquitto django-cors-headers django-restframework django-markers
```

Im EasyCabApp Git-Repository[4] unter sind im Unter-Verzeichnis `server` sämtliche Scripts abgelegt, welche für die Server-Installation verwendet werden. Die Konfigurations-Dateien und Scripts aus dem Verzeichnis `bash` müssen noch in die entsprechenden Ziel-Verzeichnisse kopiert werden.

Die Dateien im Verzeichnis `python` sind in einem eigenen Git Sub-Repository[5] abgelegt und können direkt in das Verzeichnis `/usr/local/python` geklont werden.

```
$ mkdir /root/easycab/
$ cd /root/easycab/
$ git clone https://github.com/hanjo77/EasyCabApp .
$ git pull
$ cd EasyCabApp/server/
$ cp -f bash/easycabd.sh /etc/init.d/
```

```

$ chmod 755 /etc/init.d/easycabd
$ cp -f bash/mosquittd /etc/init.d/
$ chmod 755 /etc/init.d/mosquittd
$ cp -f bash/wsgi.conf /etc/apache2/mods-enabled/
$ cp -f bash/mosquitto.conf /etc/mosquitto/
$ mkdir /usr/local/python
$ cd /usr/local/python/
$ git clone https://github.com/hanjo77/EasyCabApp-Server-python .
$ git pull
$ update-rc.d easycabd defaults
$ update-rc.d mosquittd defaults
$ service apache2 restart

```

Erstellen einer Datenbank easycab und eines gleichnamigen Benutzers mit Passwort raspberry:

```

$ mysql -u root -p
mysql> create database easycab;
mysql> grant usage on *.* to easycab@localhost identified by 'raspberry';
mysql> grant all privileges on easycab.* to easycab@localhost;
mysql> exit;

```

Falls ein anderes Passwort oder ein anderer Benutzername gewählt wurde, muss dies noch in der Datei `/usr/local/python/django/easycab/settings.py` angepasst werden.

Nun müssen die Django-Models in der Datenbank abgebildet werden und das Passwort des Root-Benutzers für die Django-Administrations-Oberfläche muss geändert werden. Ausserdem muss in der Applikations-Konfiguration ein Eintrag mit Standard-Werten erstellt werden.

```

$ cd /usr/local/python/django/
$ python manage.py migrate
$ python manage.py changepassword
$ python manage.py migrate --run-syncdb
$ python manage.py migrate collectstatic
$ mysql easycab -u easycab -p
mysql> insert into data_appconfig ('position_update_interval', '
    maximal_data_storage_days', 'encryption_key', 'session_timeout') values
    (10, 90, '0123456789abcdef', 60);
mysql> exit;

```

In der Datei `/etc/crontab` werden folgende Zeilen eingefügt, damit die Positionsdaten nach der in der Konfiguration hinterlegten Anzahl Tage gelöscht werden.

```

/etc/mosquitto/mosquitto.conf
15 0 * * * root /usr/bin/python /usr/local/python/django/remove-old-
    positions.py

```

Letztendlich müssen die entsprechenden URLs in der Datei `/usr/local/python/django/data/static/js/easycab-util.js` angepasst werden, damit die JavaScript-Skripts über die korrekten Web-Pfade, respektive über die korrekten MQTT-URL und -Port kommuniziert:

```

djangoRootPath: "http://url.des.servers/data",
adminRootPath: "http://url.des.servers/data/admin",
mqttUrl: "url.des.servers", // Host / IP for mqtt connection
mqttPort: 10001, // Port for mqtt connection

```

Die Applikation kann nun über die URL des Servers und Pfad `/data` aufgerufen werden.

11 Fazit

11.1 Ergebnis

Allgemein

Wir konnten einen Prototypen entwickeln, der den gestellten Anforderungen genügt. Wir haben viele Einzelteile entwickelt und zu einer Tracking-Lösung zusammengesetzt.

Dadurch, dass der Fahrer das Gerät einfach ausschalten kann, wurde die Anforderung an den Schutz der Privatsphäre umgesetzt.

Die Programmiersprache Python eignet sich für Projekte auf einem Raspberry Pi besonders gut. Es existieren viele Module, welche die Steuerung des Raspberry Pi erleichtern. Auch für die Server-Komponenten erwies sich Python als sehr flexible Sprache. Django vereinfachte zudem den Aufbau einer Administrations-Oberfläche und bietet die Möglichkeit, zu einem späteren Zeitpunkt einfach die Datenbank-Technologie auszuwechseln.

Obwohl erste Testgeräte Anfang November an EasyCab ausgeliefert wurden, wurden diese leider zu wenig getestet und lieferten deshalb auch nicht genügend Testdaten. Da wir selber auch keine eigenen Autos besitzen, konnten auch von unserer Seite keine eigenen, längeren Fahrtest durchgeführt werden.

Optionen

Die Optionen "Gehäuse" und "Konfiguration per USB Memory-Stick" konnten ausgeführt werden. Für die Optionen "Fahrzeugdaten" und "Stressbutton" war nicht genügend Zeit vorhanden. Zudem wurden während des laufenden Projekts neue Anforderungen eingebracht, welche als wichtiger eingestuft und deshalb vorgezogen wurden.

Vorgehensweise / Zeitplan

Von Anfang des Projektes an der Dokumentation zu arbeiten hat sich als sehr vorteilhaft erwiesen. Ein Grossteil der Dokumentation war dadurch schon Ende Dezember fertig und wir konnten im Januar diese verfeinern, überprüfen und überarbeiten.

Der Zeitplan konnte nicht ganz eingehalten werden, die Verzögerungen waren aber so gering, dass das eingerechnete Zeitpolster diese gut aufgehoben hat. Verzögert hat sich die Auslieferung der Testgeräte.

Das 3D-Modell für das Gehäuse war zwar schon relativ früh entworfen, der effektive Druck des Gehäuses verzögerte sich dann aber, da der Drucker des Kollegen, welcher sich anfänglich anbot, leider defekt war und wir relativ lange nach Alternativen suchten.

11.2 Zukunft

Mit diesem Projekt konnten wir einen Prototypen für ein Tracking-Gerät implementieren. Für ein marktreifes Produkt sind aber noch weitere Arbeitsschritte notwendig, die den Projektumfang übersteigen.

So müsste das Gerät intensiver getestet werden. Mangels Zeit und auch materieller Ressourcen wurde nie mit einer grossen Anzahl Fahrzeugen gleichzeitig getestet. Ausserdem erfolgten die Tests nicht in einem realen Umfeld sondern nur innerhalb Gebäuden oder im öffentlichen Verkehr.

Für ein fertiges Produkt müsste der in diesem Projekt entwickelte Prototyp der LED und des Buttons durch eine geätzte Platine ersetzt werden.

Das in diesem Projekt entstandene Gehäuse ist zudem noch weit von der Perfektion entfernt und erfordert ein gewisses Mass an Feingefühl, damit die zum Teil filigranen Kunststoff-Teile beim Zusammenbau des Tracking-Geräts nicht zerbrechen.

11.3 Persönliches Fazit

Die Entwicklung des Tracking-Lösung war ein vielseitiges und spannendes Projekt. Wir konnten nicht nur Software entwickeln und theoretisches Wissen studieren, sondern auch mit einem Einplatinencomputer und Sensoren arbeiten. Weitere spannende Herausforderungen waren die Datenübertragung per MQTT und das Tethering mit dem Smartphone. Schlussendlich konnten wir uns auch handwerklich und gestalterisch einbringen, indem wir eine elektronische Schaltung gebaut und ein Kunststoffgehäuse gestaltet haben.

Wir haben die Programmiersprache Python kennen und schätzen gelernt. Die Sprache sehr mächtig und dennoch leicht verständlich. Das Konzept, Code-Blöcke durch Einrücken zu definieren zwingt den Programmierer ausserdem dazu, leserlichen Code zu schreiben.

Die Entwicklung des Plastik-Gehäuses gab uns die Möglichkeit, Erfahrungen mit dem Thema 3D-Druck zu sammeln.

11.4 Dank

Wir danken Herrn Mohamed Mokdad für das Drucken der Prototypen des Gehäuses und Frau Marianne Wyss für das Korrekturlesen der Dokumentation. Ausserdem danken wir Herrn Hannes Rodel dafür, dass er uns seine Löt-Station auslieh um die LED und den Button zu löten und Herrn Dr. Reto König und der Firma EasyCab dafür, dass wir dieses interessante Projekt umsetzen durften.

12 Arbeits-Journal

18. September 2015		
Finden und Aufteilen der zu erledigenden Tasks.	jaggh1 menzs2	
Bestellen einer Server-Instanz bei DigitalOcean.	jaggh1	
Brainstorming für Use-Cases.	jaggh1 menzs2	
Anpassen der Dokumentation.	jaggh1 menzs2	
Besprechung mit Betreuer		
19. September		
Ausarbeiten der Storyboards	menzs2	
Installation und Konfiguration des Servers	jaggh1	
20. September		
Ausarbeiten der Storyboards	menzs2	
21. September		
Anpassen der Karten-Darstellung (Live-Update) über Web-socket	jaggh1	
25. September		
Ausarbeiten der Storyboards	jaggh1 menzs2	
Definieren der Milestones	jaggh1 menzs2	
Einleitung und Anforderungsdokument	jaggh1 menzs2	
Struktur des Dokuments überarbeitet	jaggh1 menzs2	
29. September		
Grundinstallation von Django und Aufbau von ersten Views	jaggh1	
1. Oktober		
Dokumentation erweitern	jaggh1	
2. Oktober		
Erstellen eines ersten Klons des Entwicklungs-Tracking-Geräts erfolgreich, allerdings funktioniert Bluetooth-Tethering unter Android noch nicht einwandfrei	jaggh1	
4. Oktober		
Dokumentation mit Aktoren erweitert, Storyboards sortiert nach Aktoren	jaggh1	
9. Oktober		
Überarbeitung und Ergänzung der Dokumentation aufgrund von Inputs des Betreuers, Kapitel Ausführung hinzugefügt	jaggh1 menzs2	
10. Oktober		
Kapitel Vorgehen, Anforderung und Zeitplan überarbeitet	jaggh1 menzs2	
13. Oktober		
Entwurf des Schaltplans für LED und Buttons	menzs2	

14. Oktober		
Entwurf des Schaltplans für LED und Buttons		menzs2
16. Oktober		
Entwurf des Schaltplans für LED und Buttons		menzs2
Tethering für verschiedene Smartphone-Betriebssysteme, Überarbeiten Server-Komponente		jaggh1
21. Oktober		
Überarbeiten der Storyboards aufgrund Anpassungen an LED- und Button-Layout		jaggh1
23. Oktober		
Marktübersicht, Anforderungen angepasst		jaggh1 menzs2
Layout Journal angepasst		menzs2
Überarbeiten des Schaltplans für LEDs und Buttons		menzs2
Überarbeiten Storyboards		jaggh1
Erweitern Administrations-Oberfläche mit Modell ''AppConfig'' für Konfigurations-Parameter		jaggh1
28. Oktober		
Überarbeiten der Logik auf Tracking-Gerät, dass in den Positions-Nachrichten nur noch die Session-ID anstelle von Fahrer, Smartphone und Taxi mitgesendet werden		jaggh1
30. Oktober		
Arbeiten an Benutzerhanduch		menzs2
Bugfixing auf Karten-Applikation, Probleme mit Datums-Formatierungen und Setzen von Bounds zum Anpassen der Ansicht auf alle vorhandenen Marker		menzs2
02. November		
LED experimentell aufgebaut, Script für Steuerung der LEDS mit Python		menzs2
03. November		
Funktionen für Buttons.		menzs2
04. November		
Threading für LED und Button.		menzs2
05. November		
Dokumentation LED und Button.		menzs2
11. November		
Auslieferung von Testgeräten an EasyCab GmbH.		jaggh1 menzs2
Erster Test in einem Fahrzeug		jaggh1 menzs2
13. November		
Dokumentation anpassen und erweitern		jaggh1 menzs2
Routen suchen		jaggh1
Treffen mit Hr. Flückiger		jaggh1 menzs2
17. November		
Erweitern Karten-Funktionalitäten und Dokumentation gemäß zusätzlichen Anforderungen von EasyCab		jaggh1

20. November		
Erster Prototyp der LED und Buttons auf Platine		menzs2
Implementation Positions-Daten Export		jaggh1
Dokumentation anpassen und erweitern		jaggh1 menzs2
21. November		
Dokumentation anpassen und erweitern		menzs2
22. November		
Dokumentation anpassen und erweitern		jaggh1 menzs2
Implementation Filter-Funktionalitäten für Positions-Daten Export		jaggh1
25. November		
Anpassungen Tethering von Bluetooth zu USB		jaggh1
Gehäuse-Design vorbereiten		jaggh1
Dokumentation anpassen und erweitern		jaggh1
27. November		
Dokumentation anpassen und erweitern		jaggh1 menzs2
Gehäuse-Design überarbeiten		jaggh1
Abstract anpassen und erweitern		menzs2
Handbuch anpassen und erweitern		menzs2 jaggh1
01. Dezember		
Erweitern der Dokumentation		jaggh1
04. Dezember		
Erweitern der Dokumentation		menzs2
Erweitern des Benutzerhandbuch		menzs2
06. Dezember		
Erweitern des Benutzerhandbuchs		jaggh1
Erweitern der Dokumentation		menzs2
08. Dezember		
Einfügen von LedButtonHandler in EasycabDaemon		menzs2
13. Dezember		
Überarbeiten Darstellung der Tests		jaggh1
17. Dezember		
Anpassung des EasyCabDaemon		menzs2
Anpassungen und Lokalisierung Administrations-Oberfläche		jaggh1
18. Dezember		
Erweitern Benutzer-Handbuch		jaggh1
Anpassungen Administrations-Oberfläche		jaggh1
20. Dezember		
Erweitern Benutzer-Handbuch		menzs2
Vereinheitlichen der Konfigurationen		jaggh1
21. Dezember		
Handler für LED und Buttons		menzs2
Kleine Reparatur an Prototyp-Platine		menzs2

Integration LED in Raspberry Pi EasyCab-Daemon	jaggh1
22. Dezember	
Anpassen Benutzerhandbuch	menzs2
Kleine Reparatur an Prototyp-Platine	menzs2
Integration Verschlüsselung bei Positionsdaten-Übertragung	jaggh1
23. Dezember	
Anpassen Dokumentation	menzs2
Erweitern Book Abstract	menzs2
Implementation Raspberry Pi Konfiguration per USB Memory-Stick	jaggh1
Erweitern Stories, Tests, Installation und Handbuch	
28. Dezember	
Anpassen Dokumentation, Grammatik und Rechtschreibung	menzs2
Erweitern Book Abstract	menzs2
03. Januar	
Gestaltung Plakat	menzs2
Erweitern Book Abstract	menzs2
05. Januar	
Anpassen Dokumentation	menzs2
07. Januar	
Anpassen JavaScript, Beheben von gelegentlichen Problemen beim Laden der Seite	jaggh1
08. Januar	
Anpassen Dokumentation	menzs2 jaggh1
09. Januar	
Aktualisieren des Gehäuses	jaggh1
Platinen bestücken	menzs2
10. Januar	
Test Server-Installation, Anpassungen an Dokumentation	jaggh1
11. Januar	
Anpassungen an Dokumentation und Book Abstract	menzs2
12. Januar	
Anpassungen an Dokumentation	menzs2
Redesign des 3D-Gehäuse-Modells	jaggh1
13. Januar	
Erweitern der Dokumentation	jaggh1
15. Januar	
Erweitern der Dokumentation	jaggh1 menzs2
Vorbereiten der Präsentation	jaggh1 menzs2
16. Januar	
Anpassen Dokumentation, Grammatik und Rechtschreibung	jaggh1 menzs2
Vorbereiten der Präsentation	jaggh1 menzs2
17. Januar	
Erweitern der Dokumentation	jaggh1 menzs2

Vorbereiten der Präsentation	jaggh1 menzs2
Test Installation Raspberry Pi	jaggh1
Platinen bestücken	menzs2
<hr/>	
18. Januar	
Erweitern der Dokumentation	jaggh1 menzs2
Vorbereiten der Präsentation	jaggh1 menzs2
<hr/>	
19. Januar	
Erweitern der Dokumentation	jaggh1 menzs2
<hr/>	
20. Januar	
Erweitern der Dokumentation	jaggh1 menzs2
Vorbereiten der Präsentation	jaggh1 menzs2
<hr/>	
21. Januar	
Abgabe der Dokumentation	jaggh1 menzs2
<hr/>	

Glossar

- AES** **A**dvanced **E**ncryption **S**tandard ist eine Blockchiffre, die als Nachfolger für DES im Oktober 2000 vom National Institute of Standards and Technology (NIST) als Standard bekanntgegeben wurde. Der Algorithmus besitzt variable, voneinander unabhängige, Block- und Schlüssellängen von 128, 160, 192, 224 oder 256 Bit und bietet ein sehr hohes Maß an Sicherheit; erst mehr als zehn Jahre nach seiner Standardisierung wurde der erste theoretisch interessante, praktisch aber nicht relevante Angriff gefunden.. i, 42
- Bluetooth** Ein Standard für Datenübertragung per Funktechnik. Benannt nach dem dänischen König Harald Blauzahn. 22, 23, 29, 30
- CAN-Bus** **C**ontroller **A**rea **N**etwork ist ein serielles Bus-System das als Feldbus Aktoren und Sensoren mit einem Steuerungsgerät verbindet. Es wurde von der Robert Bosch GmbH ab 1983 als Steuergerät für die Automobil-Industrie entwickelt. Mittlerweile wird es auch in andere Fahrzeug-Typen, in medizinischen oder industriellen Anlagen eingesetzt.. 6, 10, 22
- CSV** **C**omma-**S**eparated **V**alues. i, 20, 59
- Django** Django ist ein Web Application Framework in Python, welches sich am Model-View-Controller Konzept orientiert.. 8, 22, 23, 24, 36, 39, 40, 42, 74, 79, 80
- Elektrischer Widerstand** Der elektrische Widerstand R gibt an, wie stark der elektrische Stromfluss durch bestimmte Bauteile begrenzt wird. Die Masseinheit ist Ohm, Zeichen Ω .. 26
- GPIO** **G**eneral **P**urpose **I**nterface **O**utput. Ein Pin eines Chips. 25, 27, 30, 31, 32, 39, 72
- GPS** **G**lobal **P**ositioning **S**ystem. Globales Navigationssatellitensystem zur Positionsbestimmung, das ursprünglich von US-Verteidigungsministerium entwickelt wurde.. i, 2, 6, 9, 11, 12, 13, 22, 29, 32, 38, 43, 44, 45, 60, 68, 73
- JSON** **J**ava**S**cript **O**bject **N**otation. Standard zum Austausch von Daten zwischen Applikationen mit verschiedenen Technologien. JSON-Parser existieren für fast alle Programmiersprachen.. 24, 39
- LED** **L**ight **E**mitting **D**iode. Ein Halbleiterbauteil das Licht austrahlt und den Eigenschaften einer Diode entspricht, also den Strom nur in eine Richtung durchlässt.. i, 4, 6, 11, 12, 13, 16, 25, 26, 27, 29, 30, 31, 32, 33, 35, 39, 43, 44, 45, 46, 47, 48, 60, 68, 69, 70, 73, 80, 81
- MAC-Adresse** Die MAC-Adresse ist die Hardware-Adresse jedes einzelnen Netzwerkadapters, die als eindeutiger Identifikator des Geräts in einem Rechnernetz dient.. 15
- MQTT** MQTT ist ein leichtgewichtiges Protokoll zur Übertragung von Telemetrie-Daten. Weitere Informationen zu MQTT sind im Kapitel 6.11 zu finden.. 4, 23, 24, 25, 29, 38, 39, 41, 79, 81
- NFC** **N**ear **F**ield **C**ommunication. Ein Übertragungsstandard zum kontaklosen Datenaustausch über kurze Strecken.. 8, 9, 11, 12, 13, 14, 15, 22, 24, 29, 38, 43, 44, 45, 60, 68, 73
- Python** Python ist eine interpretierte, objektorientierte Programmiersprache. i, 3, 24, 31, 36, 38, 39, 40, 42, 76, 77, 78, 80, 81

Tethering Verbindung des Smartphones an einen Rechner, um die Internet-Verbindung zu teilen. 9, 11, 12, 16, 22, 23, 29, 30, 44, 45, 48, 76, 81

USB **U**niversal **S**erial **B**us ist ein serielles Bussystem zur Verbindung eines Computers per Kabel mit externen Geräten.. i, 10, 11, 12, 14, 16, 21, 22, 23, 29, 30, 33, 39, 43, 44, 45, 48, 60, 67, 68, 73, 74, 76, 80

UUID **U**niversally **U**nique **I**Dentifier, wird verwendet, um Objekte eindeutig zu identifizieren. 14, 15, 41

VPN **V**irtual **P**rivate **N**etwork bezeichnet ein virtuelles, in sich geschlossenes Kommunikationsnetz, welches ein bestehendes Kommunikationsnetz als Transportmedium verwendet.. 2

WLAN **W**ireless **L**ocal **A**rea **N**etwork bezeichnet ein lokales Funknetz der IEEE-802.11-Familie.. 22, 29

WSGI **W**eb **S**erver **G**ateway **I**nterface. Eine Spezifikation, welche beschreibt wie ein Webserver mit Web-Applikationen kommuniziert und wie Web-Applikationen zusammen verkettet werden können um eine Anfrage zu verarbeiten. 22, 74

Quellenverzeichnis

- [1] Advanced encryption standard aes. [Online]. Available: https://de.wikipedia.org/wiki/Advanced_Encryption_Standard
- [2] Blender opensource 3d modellierungs-software. [Online]. Available: <http://www.blender.org>
- [3] Cryptojs - javascript implementations of standard and secure cryptographic algorithms. [Online]. Available: <https://code.google.com/p/crypto-js/>
- [4] Easycabapp git repository. [Online]. Available: <https://github.com/hanjo77/EasyCabApp>
- [5] Easycabapp git repository, python server komponenten. [Online]. Available: <https://github.com/hanjo77/EasyCabApp-Server-Python>
- [6] Fritzing. [Online]. Available: <http://fritzing.org/home/>
- [7] Gpio electrical specifications. [Online]. Available: <http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/raspberry-pi/gpio-pin-electrical-specifications>
- [8] Minibian os. [Online]. Available: <https://minibianpi.wordpress.com/>
- [9] Python cryptography toolkit. [Online]. Available: <https://pypi.python.org/pypi/pycrypto>
- [10] Rpi easy sd card setup. [Online]. Available: http://elinux.org/RPi_Easy_SD_Card_Setup
- [11] B. Croston. Python library, rpi.gpio modul. [Online]. Available: <https://pypi.python.org/pypi/RPi.GPIO/0.5.11>
- [12] R. Light. Mosquitto mqtt manpage. [Online]. Available: <http://mosquitto.org/man/mqtt-7.html>
- [13] P. Schnabel. Elektronik kompendium. [Online]. Available: <http://www.elektronik-kompendium.de>

Abbildungsverzeichnis

2.1	Task-Liste	5
5.1	Story: Fahrer meldet sich mit NFC-Chip an und wird getrackt	11
5.2	Story: Fahrer hat keinen NFC-Chip dabei	12
5.3	Story: Fahrer will nicht getrackt werden	13
5.4	Story: Gerät zurücksetzen	13
5.5	Story: Einbau eines Tracking-Geräts in ein Fahrzeug	14
5.6	Story: Ein neues Tracking-Gerät vorbereiten	14
5.7	Story: Ein neues Fahrzeug hinzufügen	14
5.8	Story: Einen neuen Fahrer hinzufügen	15
5.9	Story: Ein neues Smartphone hinzufügen	15
5.10	Story: Pairing von Tracking-Gerät und Smartphone	16
5.11	Story: Konfiguration auf Tracking-Gerät anpassen	16
5.12	Story: Nur aktive Fahrzeuge anzeigen	17
5.13	Story: Nur inaktive Fahrzeuge anzeigen	17
5.14	Story: Zurückgelegte Strecke eines Fahrzeugs anzeigen	18
5.15	Story: Ort auf der Karte suchen	18
5.16	Story: Route berechnen	19

5.17	Story: Route von Fahrzeug als Startpunkt berechnen	19
5.18	Story: Daten auswerten	20
6.1	Netzwerk Überblick	21
6.2	Domänen-Modell der Datenbank	23
6.3	MQTT	25
6.4	Knotenregel	26
6.5	Maschenregel	26
7.1	Erster Prototyp des Tracking-Geräts	29
7.2	LED und Button: Erster Entwurf	31
7.3	LED und Button: Zweiter Entwurf	31
7.4	LED und Button: Versuchsaufbau	32
7.5	GPIO Layout Raspberry Pi B+ und 2	33
7.6	Prototyp der LED/Button Platine	33
7.7	3D-Ansicht des ersten Gehäuse-Prototypen	34
7.8	3D-Ausdruck des ersten Prototypen	34
7.9	3D-Ausdruck des zweiten Prototypen	35
7.10	3D-Ausdruck des dritten Prototypen	35
7.11	Gehäuse mit aufgeklebten Icons in gelb und blau	36
7.12	Erster Prototyp der Karten-Applikation	37
7.13	Erster Prototyp der Administrations-Oberfläche	37
7.14	Karte mit Such- und Streckenberechnungs-Funktionalität	38
9.1	Initiale Ansicht der Kartenapplikation	61
9.2	Detailansicht eines Fahrzeugs	61
9.3	Zurückgelegte Route eines Fahrzeugs	62
9.4	Karten Filterfunktion	62
9.5	Auto-Complete Funktion	63
9.6	Anzeige eines gesuchten Standorts	63
9.7	Route zwischen zwei Standorten	64
9.8	Ansicht nachdem ein Ziel-Standort gesucht wurde	64
9.9	Route zwischen einem Fahrzeug und Standort	65
9.10	Initiale Ansicht der Administrations-Oberfläche	65
9.11	Ansicht "Applikations-Konfigurationen bearbeiten"	66
9.12	Hinzufügen eines neuen Stammdaten-Elements	66
9.13	Bearbeiten eines Stammdaten-Elements	66
9.14	Entfernen von Stammdaten-Elementen	67
9.15	Overlay Formular "Positions-Daten exportieren"	67
9.16	gebogene Widerstände	69
9.17	Kabel abisoliert und verlötet	69
9.18	LED	70
9.19	Platine: Schritt 1	70
9.20	Platine: Schritt 2	70
9.21	Platine: Schritt 3	71
9.22	Platine: Schritt 4	71
9.23	Platine: Schritt 5	71
9.24	Platine: Schritt 6	72
9.25	Fertige Platine	72
9.26	GPIO Layout Raspberry Pi B+ und 2	73



Erklärung der Diplomandinnen und Diplomanden *Déclaration des diplômé-e-s*

Selbständige Arbeit / *Travail autonome*

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Par ma signature, je confirme avoir effectué ma présente thèse de bachelor de manière autonome. Toutes les sources d'information (littérature spécialisée, discussions avec spécialistes etc.) et autres ressources qui m'ont fortement aidé-e dans mon travail sont intégralement mentionnées dans l'annexe de ma thèse. Tous les contenus non rédigés par mes soins sont dûment référencés avec indication précise de leur provenance.

Name/Nom, Vorname/Prénom

Hansjörg Jaggi

Datum/Date

21.01.2016

Unterschrift/Signature

HJ

Dieses Formular ist dem Bericht zur Bachelor-Thesis beizulegen.
Ce formulaire doit être joint au rapport de la thèse de bachelor.



Erklärung der Diplomandinnen und Diplomanden *Déclaration des diplômé-e-s*

Selbständige Arbeit / *Travail autonome*

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Par ma signature, je confirme avoir effectué ma présente thèse de bachelor de manière autonome. Toutes les sources d'information (littérature spécialisée, discussions avec spécialistes etc.) et autres ressources qui m'ont fortement aidé-e dans mon travail sont intégralement mentionnées dans l'annexe de ma thèse. Tous les contenus non rédigés par mes soins sont dûment référencés avec indication précise de leur provenance.

Name/Nom, Vorname/Prénom

Stephan Menzi

Datum/Date

21.01.2016

Unterschrift/Signature

Dieses Formular ist dem Bericht zur Bachelor-Thesis beizulegen.
Ce formulaire doit être joint au rapport de la thèse de bachelor.