



Logistisches Informationssystem für Taxis unter Wahrung der Privatsphäre

Benutzerhandbuch

Autoren: Menzi Stephan, Jaggi Hansjürg

Datum: 21.01.2016

Inhaltsverzeichnis

1 Handbuch	2
1.1 Tracking Gerät	2
1.1.1 Tracking beginnen	2
1.1.2 Tracking beenden	2
1.2 Karten-Applikation	3
1.2.1 Fahrzeugdetails	3
1.2.2 Zurückgelegte Route eines Fahrzeugs	4
1.2.3 Fahrzeugübersicht und Filter	4
1.2.4 Standort suchen	5
1.2.5 Route zwischen zwei Standorten anzeigen	5
1.2.6 Route von Fahrzeug zu einem Standort anzeigen	6
1.3 Administrations-Oberfläche	7
1.3.1 Applikations-Konfigurationen bearbeiten	7
1.3.2 Fahrer, Smartphone oder Fahrzeug hinzufügen	8
1.3.3 Fahrer, Smartphone oder Fahrzeug bearbeiten	8
1.3.4 Fahrer, Smartphone oder Fahrzeug entfernen	9
1.3.5 Positions-Daten exportieren	9
1.4 Installation	10
1.4.1 Tracking Gerät	10
1.4.2 Server	15
1.5 Konfiguration des Tracking-Geräts anpassen	16
2 Installation	17
2.1 Raspberry PI	17
2.1.1 Betriebssystem	17
2.1.2 Zugriff per SSH	17
2.1.3 Anpassen der primären Partition	17
2.1.4 Benötigte Software installieren	18
2.2 Server	19
2.2.1 Zeitzone anpassen	19
2.2.2 Python / PIP	19
2.2.3 Apache & MySQL	19
2.2.4 Mosquitto	20
2.2.5 Python Module	20
Quellenverzeichnis	22

1 Handbuch

1.1 Tracking Gerät

1.1.1 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 Fahrzeuges 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.

1.1.2 Tracking beenden

Das Tracking des Fahrzeuges 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.

1.2 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.

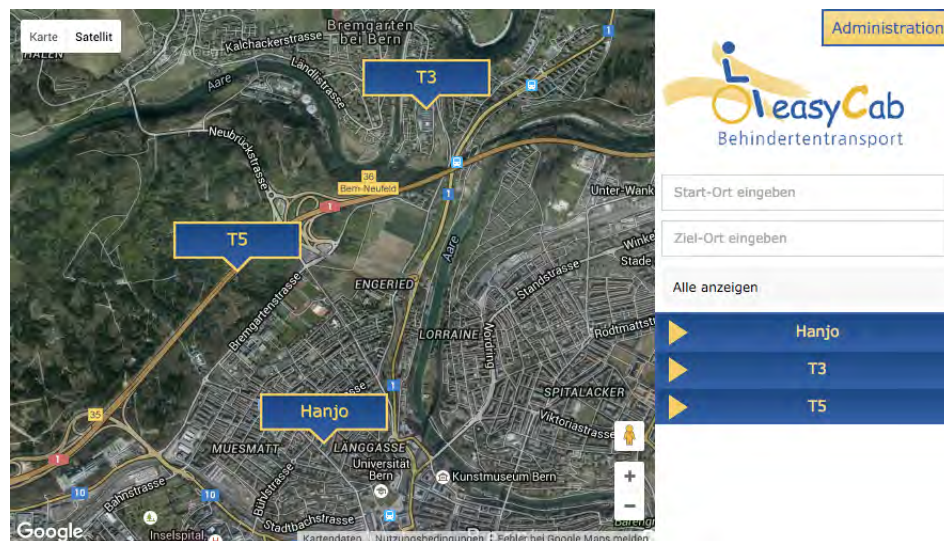


Abbildung 1.1: Initiale Ansicht der Kartenapplikation

1.2.1 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.

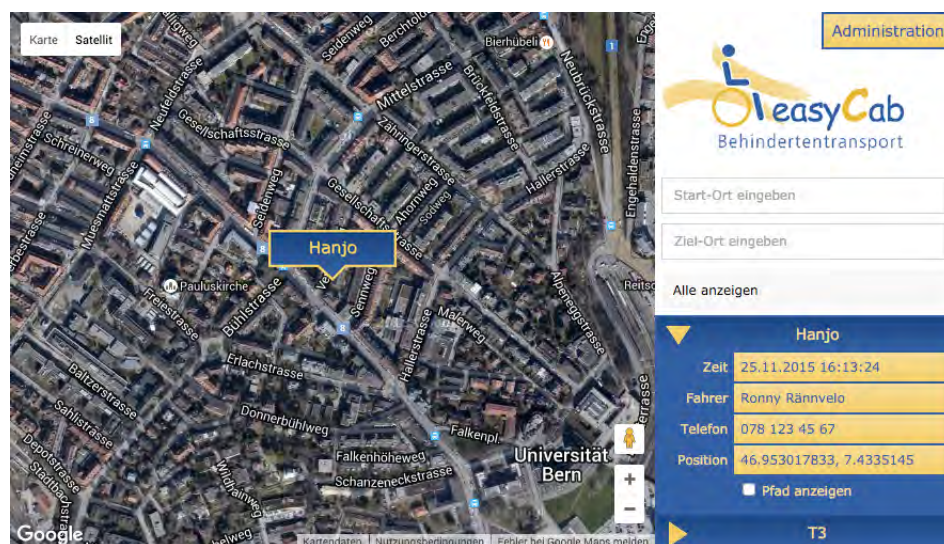


Abbildung 1.2: Detailansicht eines Fahrzeugs

1.2.2 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.

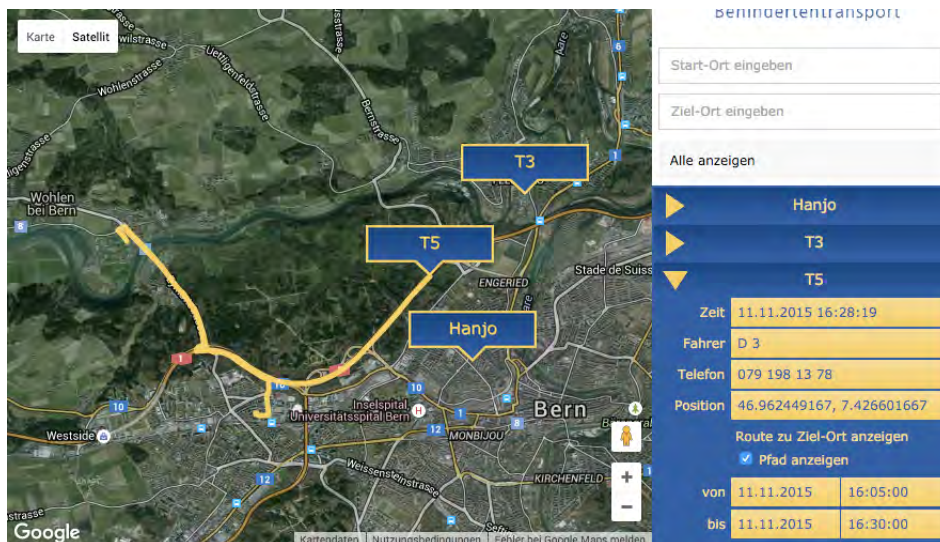


Abbildung 1.3: Zurückgelegte Route eines Fahrzeugs

1.2.3 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.

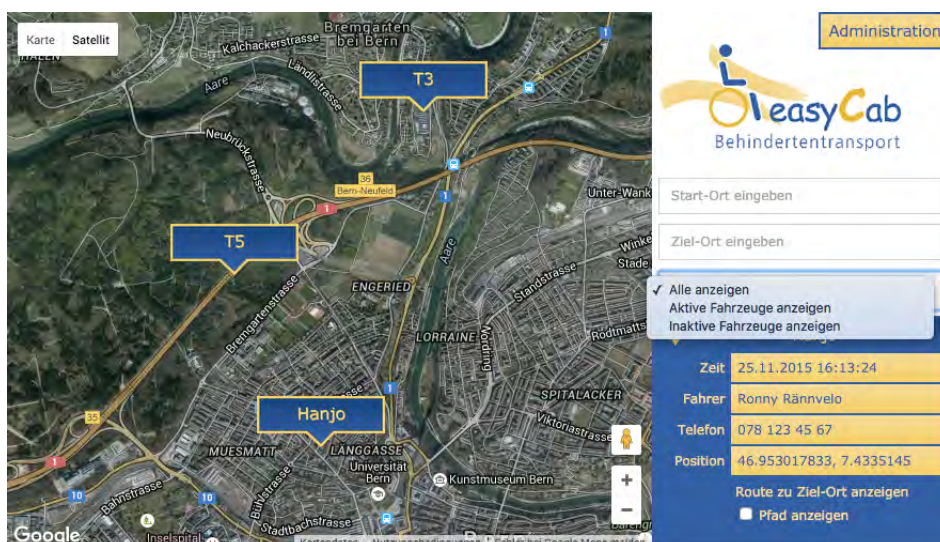


Abbildung 1.4: Karten Filterfunktion

1.2.4 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 1.5: Auto-Complete Funktion

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

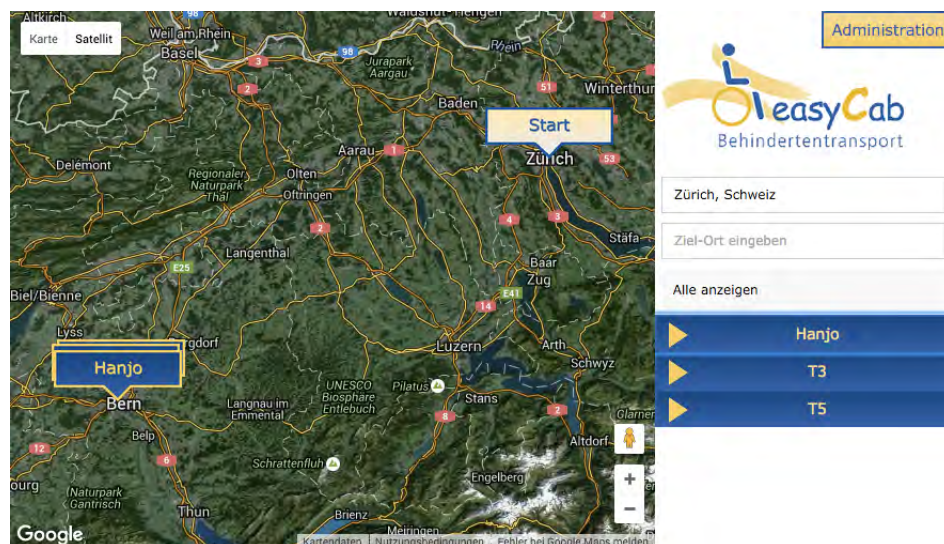


Abbildung 1.6: Anzeige eines gesuchten Standorts

1.2.5 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.

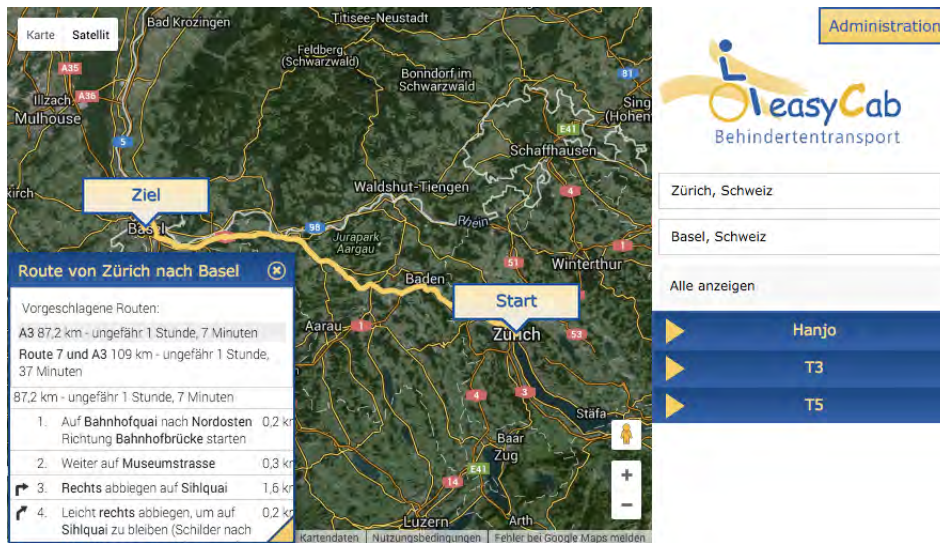


Abbildung 1.7: Route zwischen zwei Standorten

1.2.6 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.

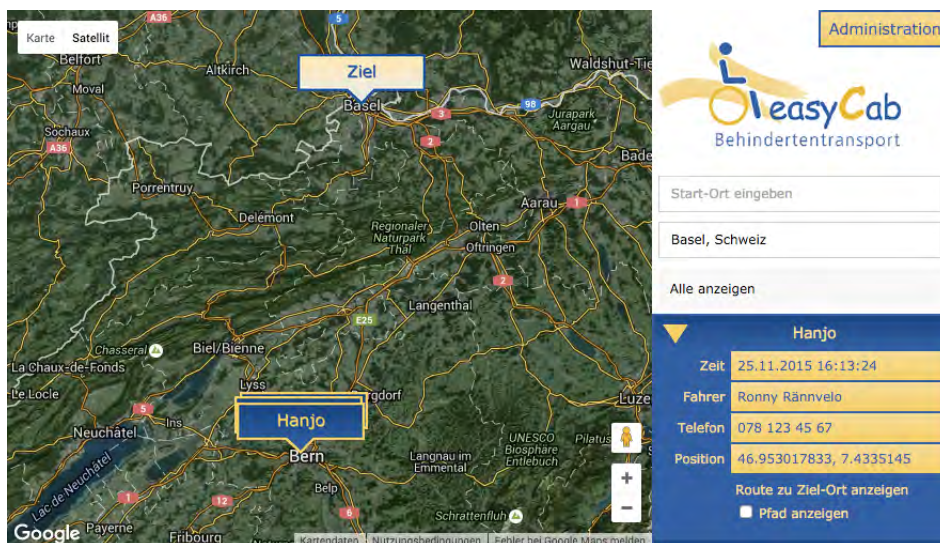


Abbildung 1.8: Ansicht nachdem ein Ziel-Standort gesucht wurde

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 1.9: Route zwischen einem Fahrzeug und Standort

1.3 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.

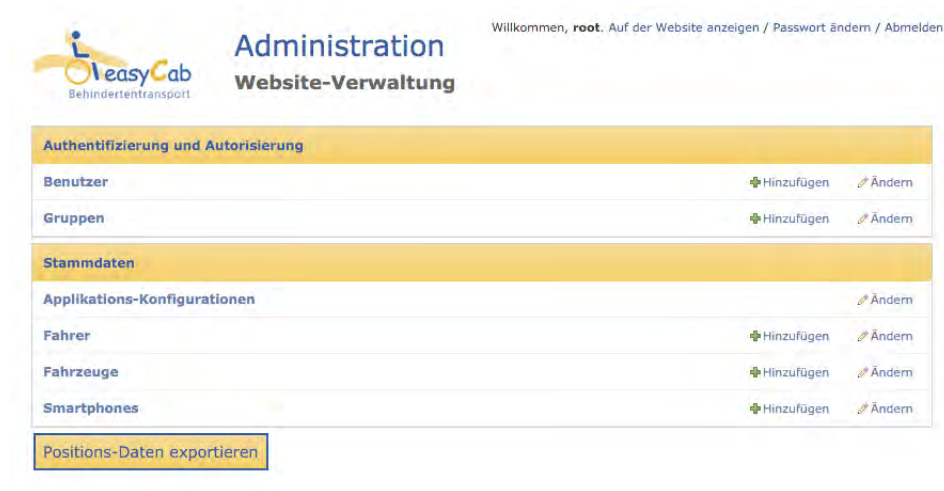


Abbildung 1.10: Initiale Ansicht der Administrations-Oberfläche

1.3.1 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' interface 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 1.11: Ansicht "Applikations-Konfigurationen bearbeiten"

1.3.2 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' interface 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 1.12: Hinzufügen eines neuen Stammdaten-Elements

1.3.3 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' interface 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 1.13: Bearbeiten eines Stammdaten-Elements

1.3.4 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 1.14: Entfernen von Stammdaten-Elementen

1.3.5 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 1.15: Overlay Formular “Positions-Daten exportieren”

1.4 Installation



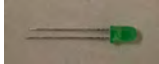





1.4.1 Tracking Gerät

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

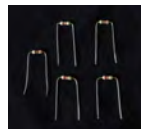


Abbildung 1.16: gebogene Widerstände

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

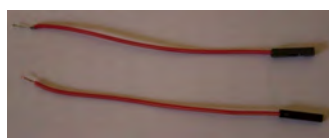


Abbildung 1.17: Kabel abisoliert und verlötet

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 Bein. Zur Sicherheit die Spezifikation der LED lesen.

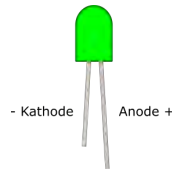


Abbildung 1.18: LED

Die LED anlöten und die überstehenden Beine abschneiden.

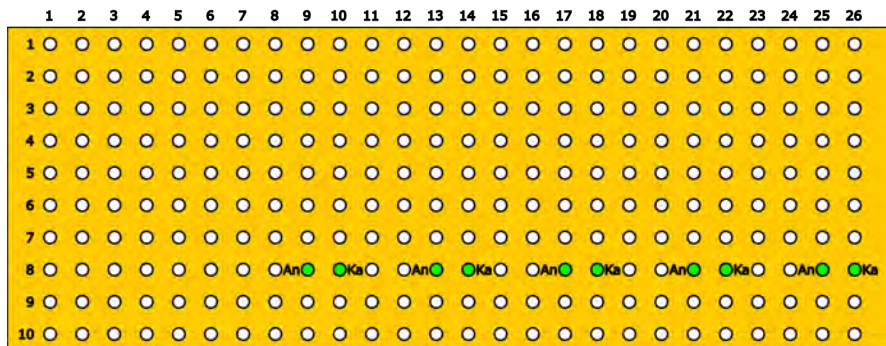


Abbildung 1.19: Platine: Schritt 1

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

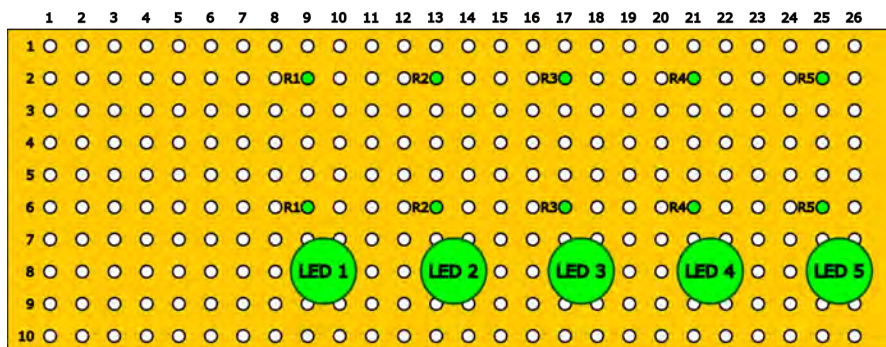


Abbildung 1.20: Platine: Schritt 2

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

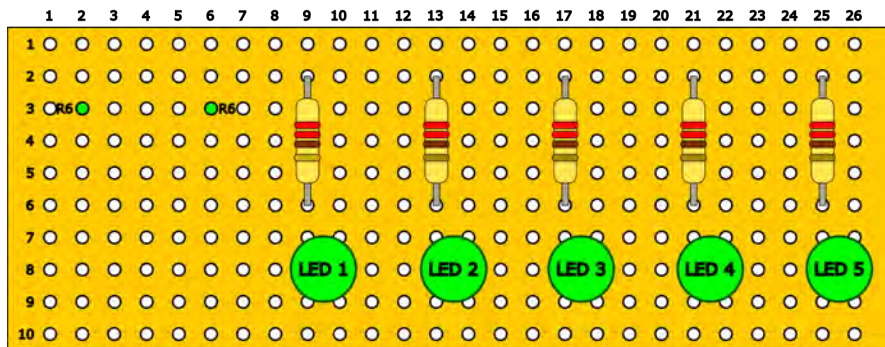


Abbildung 1.21: Platine: Schritt 3

5. Button einsetzen und anlöten.

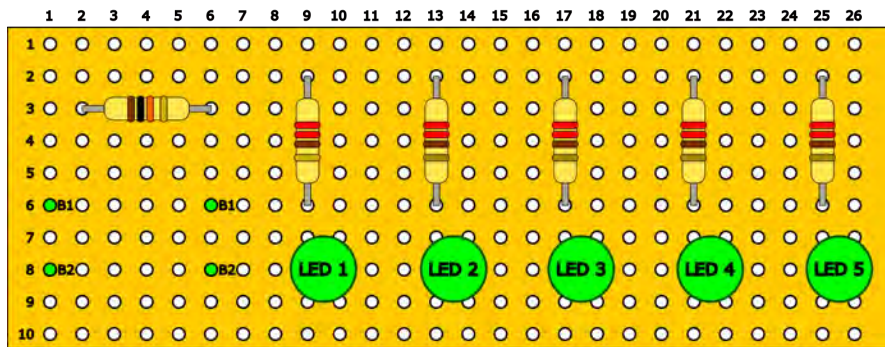


Abbildung 1.22: Platine: Schritt 4

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

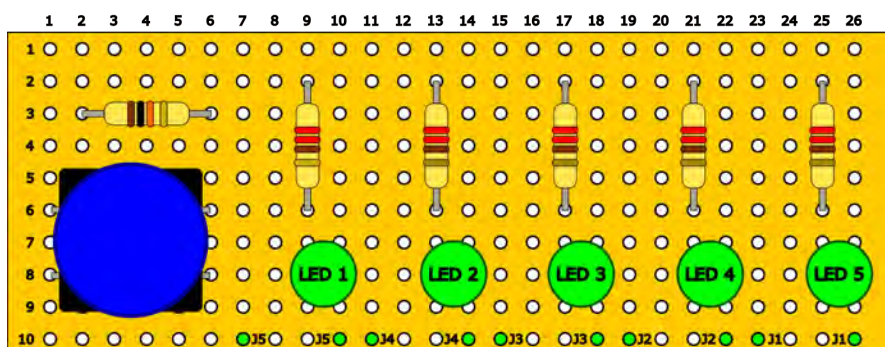


Abbildung 1.23: Platine: Schritt 5

7. GPIO-Kabel einsetzen und löteten.

Kabel 1 bis 6 mit den Widerständen verbinden.

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

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

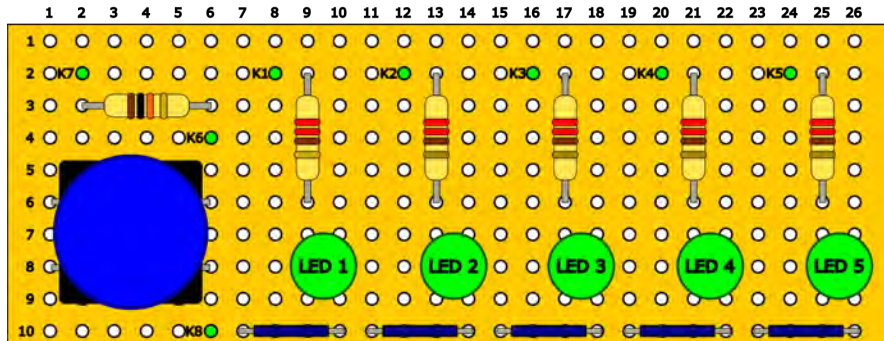


Abbildung 1.24: Platine: Schritt 6

8. Fertig

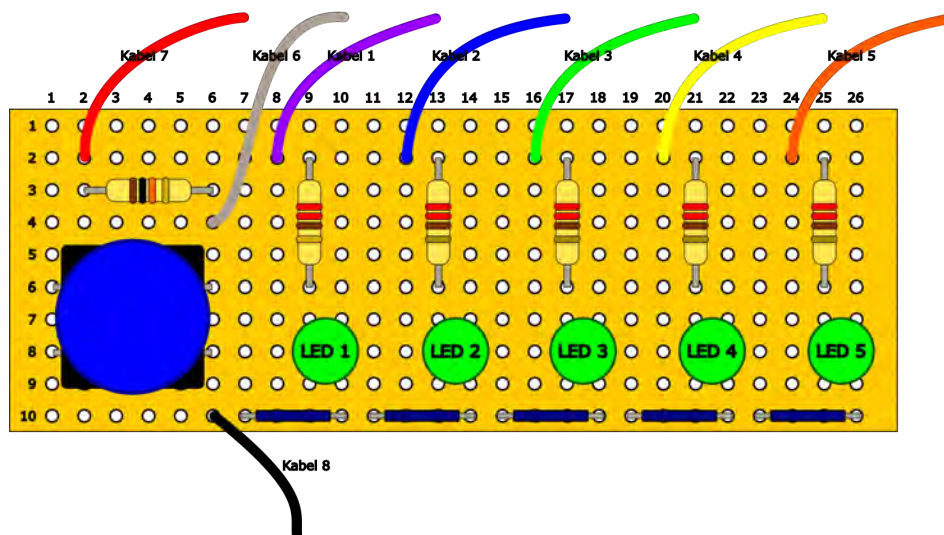


Abbildung 1.25: Fertige Platine

Software

Die Image-Datei `easycabian.img` enthält das Betriebssystem und alle benötigte Software für das Tracking-Gerät. 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 2 beschreibt die Neuinstallation des Betriebssystems und der Software.

Gehäuse

Das Gehäuse kann mit gängigen 3D Druckern hergestellt werden. Die dafür benötigten Vorlagen sind die Dateien `case-bottom.stl` für den unteren Teil und `case-top.stl` für den oberen Teil des Gehäuses. Die beiden Dateien befinden sich im Verzeichnis `3d-model` im EasyCabApp Git-Repository[1]. Für 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)

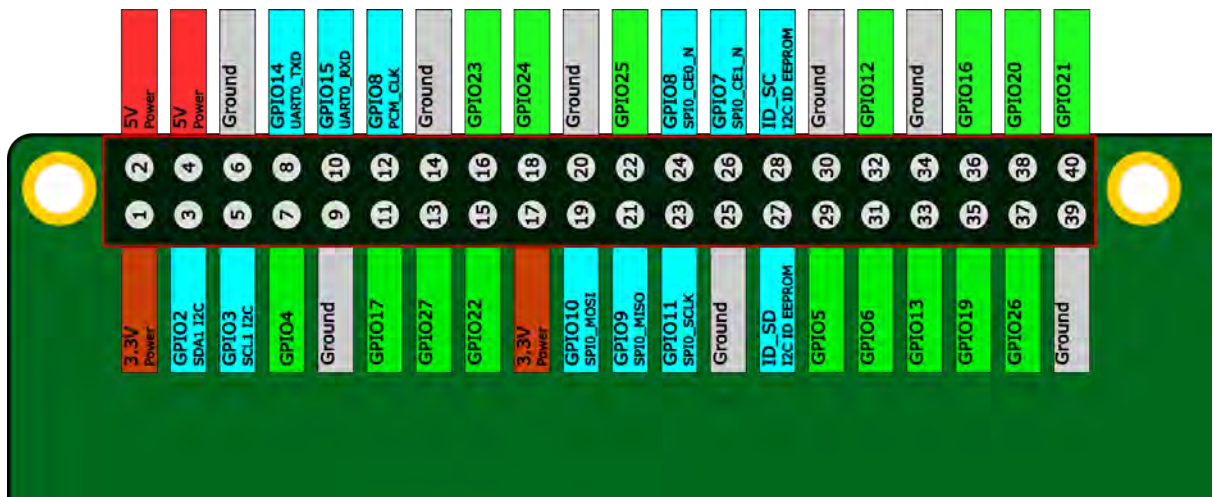


Abbildung 1.26: GPIO Layout Raspberry Pi B+ und 2

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.

1.4.2 Server

Die Installation des Servers wird in der Installationsanleitung unter Kapitel 2.2 behandelt.

1.5 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 1.2: 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.

2 Installation

2.1 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".

2.1.1 Betriebssystem

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

Die Seite "RPi Easy SD Card Setup"[3] erläutert, wie die ISO-Datei auf eine Micro-SD-Karte transferiert werden kann.

2.1.2 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
```

2.1.3 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
```

2.1.4 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[1] 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
```

2.2 Server

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

2.2.1 Zeitzone anpassen

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

```
$ dpkg-reconfigure tzdata
```

2.2.2 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
```

2.2.3 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
```

2.2.4 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
```

2.2.5 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[1] 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[2] 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.

Quellenverzeichnis

- [1] Easycabapp git repository. [Online]. Available: <https://github.com/hanjo77/EasyCabApp>
- [2] Easycabapp git repository, python server komponenten. [Online]. Available: <https://github.com/hanjo77/EasyCabApp-Server-Python>
- [3] Rpi easy sd card setup. [Online]. Available: http://elinux.org/RPi_Easy_SD_Card_Setup