

DIPLOMARBEIT

Advanced Parking Monitoring (APM)



Durchgeführt von

Philipp Kraft

Dennis Köb

Samuel Bleiner

Betreuer

Dipl.-Ing. Christoph Stüttler

Abgabevermerk

Original, 05.04.2021

Dipl.-Ing. Christoph Stüttler

Digital, 05.04.2021

AV Dipl.-Ing. Leopold Moosbrugger

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Rankweil, 1. April 2021

Philipp Kraft

Dennis Köb

Samuel Bleiner

Kurzfassung

In der nachfolgenden Arbeit wird ein System vorgestellt, welches die Parkplatzverwaltung und -überwachung vereinfachen soll. Um zu erreichen, dass diese Lösung modular und individuell auf jedem Parkplatz eingesetzt werden kann, werden zwei eigenständige Module, eines für die Kennzeichenerkennung und eines für die Fahrzeugerkennung, entwickelt. Diese Module sollen ihre Daten auf eine Datenbank hochladen können, welche dann über eine intuitive Web-Applikation verwaltet werden kann. Mithilfe der Kennzeichenerkennung sollen die unterschiedlichen Fahrzeuge eindeutig identifiziert werden und damit auch der Status des Fahrzeuges auf dem Parkplatz überwacht werden können. Die Fahrzeugerkennung detektiert den Status einer einzelnen Parklücke und dadurch kann überwacht werden welche Parklücke besetzt ist und welche nicht. Diese Daten werden in der Web-Applikation anschaulich dargestellt, welche neben umfangreichen Darstellungsmöglichkeiten auch eine eigene Benutzeroberfläche für Parkplatzbetreiber und Nutzer bietet. Dieses System ist durch seinen modularen Aufbau in der Lage sowohl Firmenparkplätze als auch öffentliche Parkplätze jeglicher Größe zu bedienen.

Abstract

In the following paper, a system is presented, which shall improve parking lot surveillance and management. To make sure that this solution can be used individually on every kind of parking lot, there will be developed two independent modules, one for license plate detection and one for vehicle detection. These modules shall be able to send their data to a database, which can be accessed and controlled via an intuitive web application. With the license plate detection, it shall be possible to clearly identify each individual vehicle and watch their status on the parking lot. The vehicle detection detects the status of each parking spot and therefore it can be monitored which parking spot is currently taken and which is free to be used. These data are visualized in the web application, which offers extensive display options as well as different user interfaces for parking lot owners and customers. Through its modular structure, the system can provide great service for company parking lots and public parking lots of any size.

Vorwort

In den Jahren unserer Ausbildungszeit ist uns immer wieder aufgefallen, dass viele öffentliche gebührenpflichtige Parkplätze nicht mit neuen Innovationen betrieben werden. Daraufhin haben wir uns durch zusätzliche Anregung von unserem Betreuungslehrer Dipl.-Ing. Christoph Stüttler dazu entschieden diese Thematik für unsere Diplomarbeit heranzuziehen.

Diese Diplomarbeit soll aufzeigen wie eine modernes System zur Parkplatzverwaltung aussehen könnte. Es werden in dieser Arbeit verschiedene Themenbereiche wie Webinterfaces, künstliche Intelligenz und Mikrokontroller behandelt.

Danksagung

Mit dieser Seite wollen wir uns bei allen Personen bedanken die uns zum Erfolg unserer Diplomarbeit geführt haben. Wir bedanken uns ausdrücklich bei unserem Betreuungslehrer Dipl.-Ing. Christoph Stüttler für seine fachliche Expertise sowie für alle kreativen Ratschläge während des gesamten Betreuungszeitraumes.

Unser hauptsächlicher Dank gilt der Firma Omicron für die finanzielle Unterstützung ohne welche die Diplomarbeit in dieser Form nicht existieren würde.

Ein herzliches Dankeschön geht an unsere Eltern für ihre Hilfe und für ihren Beistand während unserer gesamten Ausbildungszeit.

Inhaltsverzeichnis

Eidesstattliche Erklärung	i
Kurzfassung	ii
Abstract	iii
Vorwort	iv
Danksagung	v
1 Projektteam	9
2 Projektbetreuer	10
3 Projektplanung	11
3.1 OpenProject	11
3.2 Phasen	11
3.3 Arbeitspakete	11
3.4 Gantt-Diagramm	13
3.5 Zeitaufwendungen	14
4 Einleitung	16
5 Projektantrag	17
6 Kennzeichenerkennung	18
6.1 Anforderungen	18
6.2 Vorstudie	19
6.3 Bildverarbeitung	20
6.3.1 Einleitung	20
6.3.2 Bilaterale Filterung	20
6.3.3 Thresholding	21
6.3.4 Erosion	22
6.3.5 Farbraum	23

6.3.5.1	RGB	23
6.3.5.2	Graustufen	23
6.3.5.3	BGR	24
6.3.6	Konturerkennung	24
6.4	Kennzeichenerkennungsprogramm	25
6.4.1	Einleitung	25
6.4.2	Programmiersprache	25
6.4.3	Konzept	25
6.4.3.1	Bildaufnahme	25
6.4.3.2	Kennzeichenerfassung	26
6.4.3.3	Kennzeichensegmentierung	26
6.4.3.4	Zeichenerkennung	26
6.4.3.5	Anbindung an Datenbank	26
6.4.4	Ablauf	27
6.4.5	Verwendete Libraries	28
6.4.5.1	Versionsübersicht der verwendeten Libraries	28
6.4.5.2	Jupyter Notebook	29
6.4.5.3	Numpy	30
6.4.5.4	OpenCV	30
6.4.5.5	Matplotlib	32
6.4.5.6	Keras	33
6.4.5.7	Tensorflow	34
6.4.5.8	local_utils	36
6.4.5.9	Scikit-learn	36
6.4.5.10	Requests	37
6.4.5.11	Gpiozero	37
6.4.5.12	Picamera	38
6.4.6	Wichtige Programmteile	39
6.4.6.1	WPOD-NET-Modell laden	39
6.4.6.2	Kennzeichen lokalisieren	41
6.4.6.3	Bild aufnehmen	42

6.4.6.4	Zeichen mittels Konturerkennung finden	43
6.4.6.5	Bildverarbeitung	44
6.4.6.6	Visualisierung mittels Matplotlib	45
6.4.6.7	Modell für Zeichenerkennung laden und anwenden	47
6.4.6.8	Resultat mittels API an Datenbank senden	49
6.5	Raspberry Pi	50
6.5.1	Einleitung	50
6.5.2	Wahl des Raspberry Pi	50
6.5.3	Kamera	51
6.6	Maschinelles Lernen	52
6.6.1	Einleitung	52
6.6.2	Definition	53
6.6.3	Vergleich Maschinelles Lernen / Bildverarbeitung	53
6.6.4	Verwendete Modelle für Maschinelles Lernen	55
6.6.4.1	WPOD-NET	55
6.6.4.2	MobileNetV2	56
6.7	Modell Training	57
7	Fahrzeugerkennung	61
7.1	Anforderungen	61
7.2	Vorstudie	61
7.3	Erkennung von Metallen über Spulen	61
7.3.1	Messung ferromagnetischer Metalle	64
7.3.1.1	RL-Oszillator mit Timer Baustein	65
7.3.2	Messung paramagnetischer Metalle	70
7.3.2.1	LC Oszillatoren	71
7.3.2.2	Einfluss von Induktivitätsänderungen auf LC-Oszillatoren	72
7.3.2.3	Messung der Frequenz eines Colpitts-Oszillator	74
7.4	RS485 Bussystem	77
7.4.1	Benutzte Standards	77
7.4.1.1	ASCII	77

7.4.1.2	USB	77
7.4.1.3	UART	77
7.4.1.4	RS485	78
7.4.1.5	RJ45	81
7.4.2	Aufbau	81
7.4.2.1	Funktion der Adresslogikleitung	83
7.4.2.2	Versorgungsleitungen VCC und GND	85
7.4.3	Implementation eines eigenen Protokolls	86
7.4.3.1	Aufbau von Datenframes	87
7.4.3.2	Steuerabläufe	88
7.5	Mikrokontroller Slave-Geräte	88
7.5.1	Überblick	88
7.5.2	Atmega328PB	88
7.5.3	Peripherie des Mikrocontrollers	88
7.5.3.1	Spannungswandler	88
7.5.3.2	RS485 Pegelwandler	91
7.5.3.3	Digitale Ein- und Ausgänge	91
7.5.4	Layout des Slave-Gerätes	91
7.5.5	Gehäuse	91
7.6	USB-Master	91
7.6.1	USB-Bussadapter Gerät	91
7.6.1.1	Überblick	91
7.6.1.2	FT232RL	91
7.6.1.3	Spannungsversorgung	91
7.6.1.4	USB-C Anschluss	91
7.6.1.5	Layout des Master-Geräts	91
7.6.1.6	Gehäuse	91
7.6.2	Master Programm	91
7.6.2.1	Benötigte Software	91
7.6.2.2	Adressvergabe	91
7.6.2.3	Frequenzauslesung	91

7.6.2.4	Auswertung	91
7.6.2.5	API-Post	91
7.6.3	RaspberryPi als Mastergerät	92
7.6.3.1	SSH Remote Zugriff	92
7.6.3.2	Code Deployment	92
7.6.3.3	Unittest	93
8	Webinterface	93
8.1	Einleitung	93
8.2	Verwendete Technologien	93
8.2.1	HTML	93
8.2.1.1	Beispielhafte HTML Seite	94
8.2.2	CSS	95
8.2.3	JavaScript	97
8.2.4	PHP	98
8.2.5	TailwindCSS	98
8.2.6	Laravel	98
8.2.6.1	Routing	100
8.2.6.2	Blade Templates	101
8.2.6.3	Controllers	102
8.2.6.4	Artisan CLI	103
8.2.6.5	Migrations	104
8.2.6.6	Eloquent ORM	106
8.2.6.7	Laravel Sanctum	107
8.2.6.8	Sessions	107
8.2.6.9	Middleware	107
8.2.6.10	Service Provider	107
8.2.6.11	Form Validation	107
8.3	Lokale Entwicklungsumgebung mit Laragon	108
8.3.1	Benötigte Software	108
8.3.2	Konfiguration von PHP	109

8.3.3	Installation von phpMyAdmin	112
8.4	Lokale Entwicklungsumgebung mit WSL und Docker	113
8.4.1	Benötigte Software	113
8.4.2	Installation von WSL	113
8.4.2.1	2. Schritt: Virtual Machine Aktivieren	113
8.4.2.2	3. Schritt: Linux Kernel Update	114
8.4.2.3	4. Schritt: WSL 2	114
8.4.2.4	5. Schritt: Linux Distribution herunterladen	114
8.4.3	Installation von Docker	114
8.5	Production Server	115
8.5.1	Benötigte Software	116
8.5.2	Installation des LAMP Stacks	116
8.5.2.1	Apache	116
8.5.2.2	PHP	117
8.5.2.3	MariaDB	117
8.5.2.4	phpMyAdmin	118
8.5.2.5	Webinterface Virtual Host	121
8.5.2.6	Installation von Composer	122
8.5.3	Deployment mit Github Actions	122
8.5.3.1	Git Setup	122
8.5.3.2	Deploy Script	123
8.5.3.3	Server Deploy Script	124
8.5.3.4	Github Action	125
8.6	Grundlegender Aufbau Frontend	127
8.6.1	Components	127
8.6.1.1	Anonymous Components	127
8.6.1.2	Components erstellen	127
8.6.1.3	Components verwenden	128
8.6.1.4	Attribute übergeben	128
8.6.2	Layouts	129
8.6.3	Navbar	133

8.6.4	Content Header	135
8.6.5	Session Alerts	136
8.6.6	Sidebar	138
8.6.7	Footer	141
8.7	Funktionen	141
8.7.1	Dashboard	141
8.7.2	Benutzerverwaltung	143
8.7.2.1	Liste aller Benutzer	144
8.7.2.2	Benutzer erstellen/editieren	145
8.7.2.3	Profilbilder	146
8.7.3	Rollen- und Rechteverwaltung	148
8.7.3.1	Liste der verfügbaren Rechte	148
8.7.3.2	Vergabe von Rechten und Rollen mit Slugs	150
8.7.3.3	Authentisierung	151
8.7.3.4	Rolle erstellen	152
8.7.4	Neuigkeiten	153
8.7.5	Parkplatzverwaltung	155
8.7.5.1	Parklücken	156
8.7.6	Seiten Einstellungen	157
8.7.6.1	Funktion der Einstellungen	158
8.7.7	Profil	159
8.7.8	Lokalisierung	160
8.7.8.1	Funktion der Sprachauswahl	160
8.8	API	162
8.8.1	Einleitung	162
8.8.2	Autorisierung mit Bearer Token	162
8.8.3	API Tokens erstellen	163
8.8.4	Erkennungen	164
8.8.4.1	GET /api/v1/detections	164
8.8.4.2	GET /api/v1/detections/{id}	164
8.8.4.3	POST /api/v1/detections	167

8.8.5	Kennzeichen	167
8.8.5.1	GET /api/v1/plates	167
8.8.5.2	GET /api/v1/plates/{id}	168
8.8.6	Parkplätze	169
8.8.6.1	GET /api/v1/lots	169
8.8.6.2	GET /api/v1/lots/{id}	170
8.8.7	Parklücken	172
8.8.7.1	GET /api/v1/spots	172
8.8.7.2	GET /api/v1/spots/{id}	173
8.8.7.3	POST /api/v1/lots/{id}/spots	175
8.8.8	Einstellungen	177
8.8.8.1	GET /api/v1/settings	177
8.8.9	Benutzer	178
8.8.9.1	GET /api/v1/users	178
8.8.9.2	GET /api/v1/users/{id}	179
9	Zusammenfassung und Ausblick	181
10	Anhang	182
Abbildungsverzeichnis		183
Tabellenverzeichnis		187
Codeverzeichnis		188
Abkürzungsverzeichnis		192
Literaturverzeichnis		194

1 Projektteam



Philipp Kraft

E-Mail: Mail@Philipp-Kraft.com



Dennis Köb

E-Mail: Dennis.Koeb@gmail.com



Samuel Bleiner

E-Mail: Bleiner.Samuel@gmail.com

2 Projektbetreuer



Dipl.-Ing. Christoph Stüttler

E-Mail: Christoph.Stuettler@ht-rankweil.at

3 Projektplanung

Die Projektplanung ist der wichtigsten Teil des Projektmanagements. Die Rahmenbedingungen für das Projekt wurden bereits beim Antrag der Diplomarbeit festgelegt.

3.1 OpenProject

Für das Projektmanagement wird die Softwareanwendung OpenProject¹ verwendet. Der Vorteil dieser Software ist, dass diese nicht wie eine normale Desktop-Anwendung lokal auf dem Rechner des Benutzers läuft, sondern auf einem Server. Der Vorteil davon ist, dass das kollaborative arbeiten stark vereinfacht wird. OpenProject biete die Möglichkeit die Anwendung in der eigenen Infrastruktur zu installieren und bietet somit eine vollständige Kontrolle über die eigenen Daten.

3.2 Phasen

Die gesamte Projektplanung ist in vier Phasen aufgeteilt:

- **Themenfindungsphase**

In der Themenfindungsphase geht es um das finden des expliziten Themas, dabei werden die Rahmenbedingungen mit dem Betreuer festgelegt.

- **Planungsphase**

In der Planungsphase werden Arbeitspakete erstellt und den einzelnen Mitgliedern des Projektes verteilt.

- **Entwicklungsphase** Die Entwicklungsphase ist der längste und wichtigste Teil des Projekts, dabei werden alle Arbeitspakete so gut wie möglich abgearbeitet.

- **Abschlussphase** In der Abschlussphase geht es um das Schreiben der Diplomarbeit so wie letzte technische Feinschliffe in der Software und Hardware.

3.3 Arbeitspakete

Grundsätzlich sind die Arbeitspakte in drei Themenbereiche aufgeteilt:

¹<https://www.openproject.org>

- Kennzeichenerkennung (Samuel)
- Fahrzeugerkennung (Dennis)
- Webinterface (Philipp)

TYPE	ID ↑	SUBJECT
PHASE	42	Themenfindungsphase
PHASE	43	▼ Planungsphase
TASK	62	Logo
TASK	63	Projektname
TASK	64	Aufgabenverteilung
TASK	65	Individuelle Recherche
MILESTONE	44	Abgabe Antrag Schule
MILESTONE	54	Hochladen des Antrags
PHASE	55	▼ Entwicklungsphase
TASK	49	▼ Kennzeichenerkennung
TASK	72	OpenCV Kennzeichenerkennung
TASK	75	Python Kennzeichenerkennung mit Machine Learning
TASK	77	RaspberryPi Kamera
TASK	78	Kennzeichenerkennungssoftware auf RaspberryPi
TASK	79	Kamerabilder als Input
TASK	80	API Anwendung
TASK	51	▼ Webinterface
TASK	58	Rollenverwaltung
TASK	59	Kennzeichen
TASK	70	Benutzerverwaltung
TASK	71	Rechtesystem
TASK	73	Erkennungen
TASK	74	Parkplätze
TASK	76	API

Abbildung 1: Arbeitspakete Teil 1

TASK	84	▼ Fahrzeugerkennung
TASK	85	Simulation der Schaltungen
TASK	86	Schaltungsanalyse
TASK	87	Konzeptprototyp
TASK	88	Slave Schaltungsdesign
TASK	89	PCB-Design
TASK	90	Aufbau & Messung des Slave Prototyp
TASK	91	Gehäuse CAD zeichnen und 3D drucken
TASK	92	Slave Firmware schreiben
TASK	93	Master Anwendung schreiben
TASK	94	Raspberry Pi Umgebung einrichten / Remote Zugriff
PHASE	57	Abschlussphase
MILESTONE	60	Zwischenpräsentation
MILESTONE	66	Vorabgabe
MILESTONE	67	Elektronik Forum
MILESTONE	68	Hochladen der Arbeit
MILESTONE	69	Schriftliche Abgabe

Abbildung 2: Arbeitspakete Teil 2

3.4 Gantt-Diagramm

Gantt-Diagramme oder Balkenplan sind spezielle Balkendiagramme um verschiedene Arbeitspakete oder Aktivitäten auf einer Zeitachse auf einer Zeitachse darzustellen.

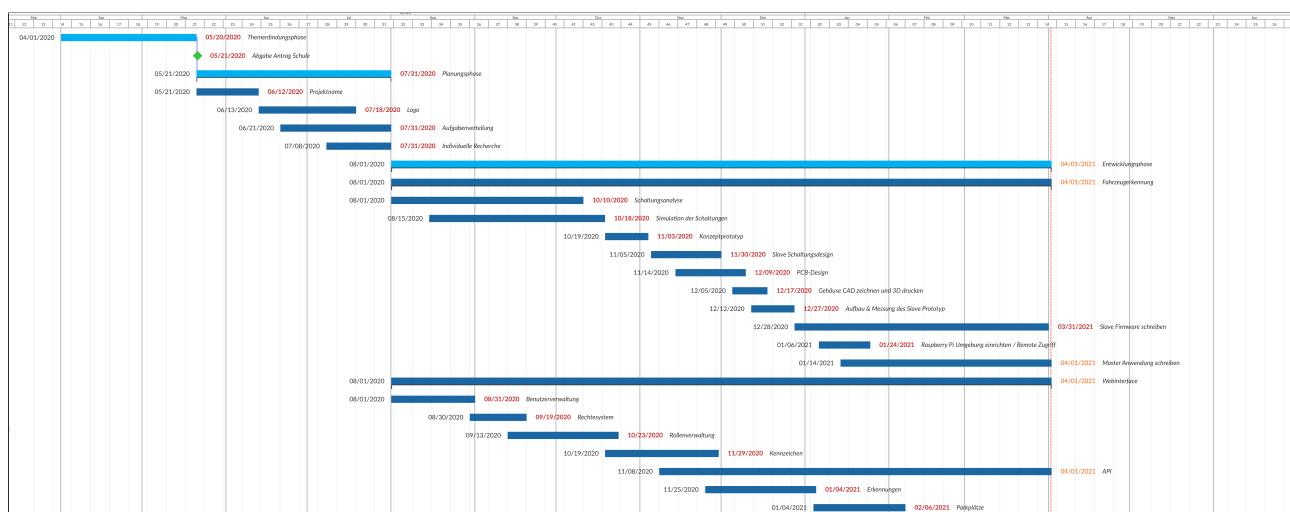


Abbildung 3: Gantt-Chart Teil 1

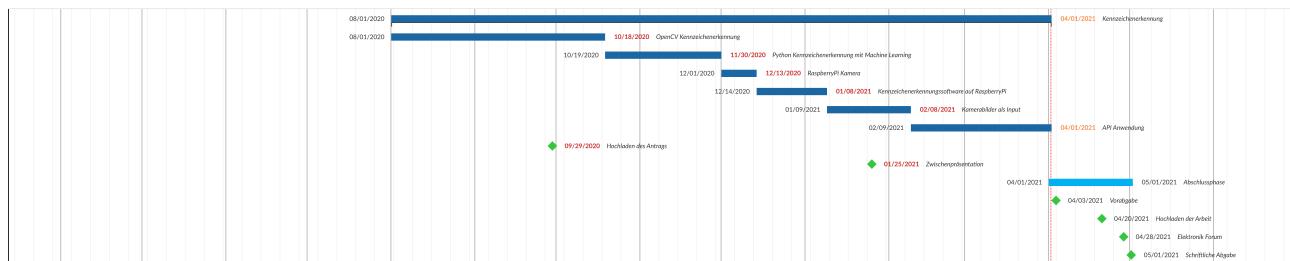


Abbildung 4: Gantt-Chart Teil 2

3.5 Zeitaufwendungen

Die Dokumentation der Zeitaufwendungen erfolgt über ein Tabellenkalkulationsprogramm, dabei wird folgendes Dokumentiert:

- Datum
- Typ
- Beschreibung
- Aufwand in Stunden

Die komplette Übersicht der aufgewendeten Stunden ist im Anhang ersichtlich.

Zeitaufwendungen

Philipp Kraft	181.5 h
---------------	---------

Dennis Köb	177 h
------------	-------

Samuel Bleiner	112.5 h
----------------	---------

Gesamtaufwand:	471 h
----------------	-------

Tabelle 1: Zeitaufwendungen Übersicht

4 Einleitung

Das Ziel dieser Arbeit ist es eine Vereinfachung von Parkplatzüberwachungen zu erstellen, welche möglichst überall eingesetzt werden kann. Dazu wird die Arbeit in drei individuelle Teile aufgeteilt, welche einfach miteinander verbunden werden können, um so einen modularen Aufbau bereitzustellen. Im ersten Teil geht es um die Erkennung der Kennzeichen von Fahrzeugen. Dies wird beim Betreten und Verlassen des Parkplatzes eingesetzt, um eindeutig festzustellen welches Fahrzeug sich im Moment auf dem Parkplatz befindet. Zudem können dadurch unerlaubte Fahrzeuge entdeckt werden und anschließend mögliche Schritte eingeleitet werden. Im zweiten Teil geht es um die Erkennung von Fahrzeugen in einer Parklücke. Dazu wird eine Spule unter jeder Parklücke verwendet, mit welcher festgestellt werden kann, ob sich im Moment dort ein Fahrzeug befindet. Dadurch kann festgestellt werden welche Parkplätze besetzt sind und im Falle von mehrstöckigen Parkhäusern kann die Auslastung der einzelnen Etagen überwacht werden. Im dritten Teil geht es dann noch um die Darstellung und Verwaltung dieser Daten. Dazu wird eine Web-Applikation bereitgestellt, welche alle Daten übersichtlich bereitstellt und für den Parkplatzbetreiber und den Kunden unterschiedliche Benutzeroberflächen bietet. Zudem kann die Web-Applikation mit wenig Aufwand von jedem Parkplatzbetreiber personalisiert werden, was vor allem für Firmen interessant ist, wenn sie diese in ihr Intranet einbauen möchten. Die Verbindung dieser drei Teile erfolgt über eine eigens dafür geschriebenen API, wodurch die Daten über das Internet übertragen werden können.

5 Projektantrag

6 Kennzeichenerkennung

6.1 Anforderungen

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

6.2 Vorstudie

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque

tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

6.3 Bildverarbeitung

6.3.1 Einleitung

Die Bildverarbeitung ist ein zentrales Thema in dieser Applikation für Kennzeichenerkennung. Sie wird für die Zeichensegmentierung verwendet, sowie für die Vorbereitung von Bildern für andere Algorithmen. Im Folgenden werden die verwendeten Bildverarbeitungsfunktionen aufgelistet und deren Funktionsweise erläutert.

6.3.2 Bilaterale Filterung

Bilaterale Filterung ist eine Methode für eine kantenerhaltende Weichzeichnung eines Bildes.

Bei der Berechnung für den Farbwert des Ausgabepixels werden die benachbarten Pixel nicht nur mit ihrer Entfernung gewichtet, sondern auch mit ihrem eigenen Farbwert. Dadurch können einzelne farbliche Ausreißer herausgefiltert werden. Dies ist vor allem in der Bildverarbeitung wichtig,

da dadurch die wichtigen Eigenschaften eines Bildes, wie zum Beispiel Kanten, erhalten bleiben und verarbeitet werden können, aber einzelne abweichende Pixel herausgefiltert werden wodurch unnötige Informationen entfernt werden.



Abbildung 5: Vor Bilateraler Filterung



Abbildung 6: Nach Bilateraler Filterung

In Abbildung 5 kann man ein Bild von verschiedenen Lebensmitteln sehen. Wenn man genau hinsieht erkennt man vor allem bei den Blättern im Hintergrund und beim Brot viele detailreiche Texturen. Diese Texturen haben keine wichtige Texturen und sind deswegen unnötig. Um die Bildverarbeitung zu vereinfachen wendet man deswegen die bilaterale Filterung auf dieses Bild an, um diese detailreichen Texturen zu vereinfachen. In Abbildung 6 sieht man das Bild nach der bilateralen Filterung. Wenn man hier dann wieder genauer auf die Blätter und das Brot sieht, erkennt man, dass die detailreichen Texturen weichgezeichnet wurden, aber die Kanten sind genauso gut erkennbar wie vor der Filterung.

6.3.3 Thresholding

Das Thresholding oder auch Schwellenwertverfahren wird in der Bildverarbeitung verwendet, um Bilder zu segmentieren. Aus einem Graubild kann dadurch ein Binäres Bild erzeugt werden.

Bei diesem Verfahren wird ein bestimmter Schwellwert (En.: Threshold) definiert, welcher mit den Grauwerten der einzelnen Pixel des Bildes verglichen wird. Wenn der Grauwert den Schwellwert überschreitet, wird dieser durch einen weißen Pixel ersetzt und wenn der Grauwert kleiner als der Schwellwert ist, wird dieser durch einen schwarzen Pixel ersetzt. Dadurch erhält man ein Bild welches nur noch zwei Farben hat, Schwarz und Weiß. Dies wird deswegen eingesetzt, da dadurch

viele Bildverarbeitungsalgorithmen schneller arbeiten und die Effizienz gesteigert wird.



Abbildung 7: Graustufenbild



Abbildung 8: Binäres Bild nach Thresholding

In Abbildung 7 sieht man ein solches Graubild welches nur verschieden Graustufen aufweist. In Abbildung 8 sieht man das Bild nach dem Thresholding. Hier kann man nur noch das Boot mit den Menschen erkennen. Dies ist nicht nur für schnellere Bildverarbeitungsalgorithmen wichtig, sondern wird auch zur Objekterkennung in Bildern verwendet.

Um den Schwellwert zu bestimmen kann man diesen entweder variieren bis das gewünschte Ergebnis erscheint oder man verwendet Methoden, welche den Schwellwert automatisch bestimmen. Eine der bekanntesten Methoden zur Schwellwertbestimmung ist die Methode von Otsu², welche mit dem Schwellenwert die Pixel in Vordergrund und Hintergrund unterteilt.

6.3.4 Erosion

Erosion ist eine Funktion der Bildverarbeitung und ist in die morphologische Bildverarbeitung einzutragen. Diese beschäftigt sich primär mit der Verarbeitung von binären Bildern, welche man nach Thresholding erhält.

Erosion benötigt zwei Eingaben, das binäre Bild und einen Kernel. Der Kernel ist dabei die Angabe, nach welcher die Erosion durchgeführt wird. Der Kernel ist auch eine binäre Struktur, welche über jeden einzelnen Pixel des binären Bildes geschoben wird. Wenn der Kernel komplett mit

²Benannt nach Nobuyuki Otsu

dem binären Bild übereinstimmt, behält dieser Pixel seinen Wert und ansonsten wird er invertiert. Dabei muss jedoch darauf geachtet werden, dass die Polarität des binären Bildes und des Kernels übereinstimmt, da sonst die Erosion nicht richtig funktioniert. Als Resultat erhält man danach ein deutlicheres Bild bei welchem einzelne Pixelfehler herausgefiltert wurden und die Konturen besser erkennbar sind.



Abbildung 9: Binäres Bild nach Thresholding



Abbildung 10: Nach Erosion

In Abbildung 9 und 10 sieht man die Anwendung der Erosion. Die Konturen der einzelnen Zeichen im Kennzeichen sind in Abbildung 10 nach der Erosion deutlicher erkennbar als davor.

6.3.5 Farbraum

Der Farbraum eines Bildes enthält alle möglichen Farben eines Farbmodells. Das Farbmodell beschreibt dabei die Parameter, aus welchen die einzelnen Farben gebildet werden. Dies ist in der Bildverarbeitung relevant, da verschiedene Funktionen der Bildverarbeitung, unterschiedliche Farbräume verwenden und dieser deswegen korrekt eingestellt werden muss.

In dieser spezifischen Applikation werden die folgenden Farbräume verwendet:

6.3.5.1 RGB

RGB ist einer der häufigsten und bekanntesten Farbräume. Er basiert auf den drei Grundfarben Rot, Grün und Blau und wird vor allem bei Bildschirmen und in der Fotografie genutzt. Die Farben setzen sich in diesem Modell aus dem jeweiligen Rot-, Grün- und Blauanteil der einzelnen Pixel zusammen.

6.3.5.2 Graustufen

Bei einem Graustufen-Bild, zu sehen in Abbildung 3, hat jeder Pixel einen Wert von 0 bis 255. Diese

Werte erstrecken sich also von Schwarz bis Weiß und dazwischen liegen verschiedene Grautöne. Dieser Farbraum wird in der Bildverarbeitung häufig verwendet, da Konturen einfacher erkennbar sind und es nur einen Parameter gibt, welcher verarbeitet werden muss, wodurch die Effizienz diverser Algorithmen gesteigert werden kann. Zudem wird dieser Farbraum auch oft in Verbindung mit Thresholding verwendet.

6.3.5.3 BGR

Der BGR ist ein relativ unbekannter und wenig verwendeter Farbraum, da er sehr ähnlich zum RGB-Farbraum ist. Der einzige Unterschied zwischen diesen beiden liegt in der Anordnung der Parameter. Bei BGR sind die Parameter spiegelverkehrt zu RGB, das heißt es kommt zuerst der Blauanteil, dann der Grünanteil und zum Schluss der Rotanteil. Insgesamt ergibt dies für die einzelnen Pixel zwar die gleichen Farben, aber die Funktionen der Bildverarbeitung müssen trotzdem das Bild im passenden Farbraum erhalten. So verwendet zum Beispiel die Funktion „imread“ von OpenCV den BGR-Farbraum und die Funktion „im Show“ von Matplotlib verwendet den RGB-Farbraum. Wenn man diese Funktionen also nacheinander anwendet, muss dazwischen der Farbraum umgewandelt werden.

6.3.6 Konturerkennung

Die Konturerkennung ist eine wichtige Funktion in der Bildverarbeitung mit welcher Objekte in einem Bild gefunden werden können. In dieser Applikation wird sie für die Zeichensegmentierung eingesetzt.

Die Konturerkennung wird hauptsächlich bei binären Bildern verwendet. Eine Kontur kann dabei wie im Folgenden definiert werden. Man überprüft jeden einzelnen Pixel und sieht nach, ob ein benachbarter Pixel einen anderen Farbwert aufweist. Falls dies zutrifft muss der zu prüfende Pixel zu einer Kontur gehören. Wenn dies auf mehrere zusammenhängende Pixel zutrifft, bedeutet das, dass diese zusammen eine Kontur bilden.

Die Funktion „findcontours“ von OpenCV, welche in dieser Applikation verwendet wird, ist eine Funktion für Konturerkennung und kann weiße Objekte auf einem schwarzen Hintergrund erkennen.

Sie basiert auf dem Algorithmus von Suzuki von 1985³ und liefert eine Liste mit allen Konturen. Die Konturen werden in der Liste als ein Array von Koordinaten abgespeichert.

6.4 Kennzeichenerkennungsprogramm

6.4.1 Einleitung

Die Software ist der wichtigste und größte Teil der Kennzeichenerkennung. Sie erhält ein Bild, in welchem ein Auto mit einem Kennzeichen enthalten ist und liefert am Ende dieses Kennzeichens und sendet dieses dann automatisch an die Datenbank. Die Software kann entweder über Bildverarbeitung oder mit Modellen für Maschinelles Lernen realisiert werden. Der erste Ansatz bei dieser Anwendung war mit klassischer Bildverarbeitung, welche aber nicht die gewünschte Genauigkeit erreicht hat, weswegen dann auf Maschinelles Lernen gewechselt wurde.

6.4.2 Programmiersprache

Die verwendete Programmiersprache für die Kennzeichenerkennung ist Python. Python ist eine höhere Programmiersprache, welche übersichtlich und leicht lesbar ist. Sie ist vor allem für Bildverarbeitung und Anwendungen mit Maschinellem Lernen gut geeignet, da es dafür hoch optimierte und effiziente Bibliotheken gibt wie zum Beispiel OpenCV, Numpy und Tensorflow. Dadurch ist Python für diese Anwendung besser geeignet als zum Beispiel C++. Dieses wäre zwar normalerweise effizienter, bietet aber weniger optimierte Bibliotheken in diesem Bereich, wodurch es hier weniger gut geeignet ist.

6.4.3 Konzept

Das Programm für die Kennzeichenerkennung basiert auf fünf Stufen. Die erste Stufe ist die Bilddaufnahme, die zweite ist die Kennzeichenerfassung mittels Maschinellem Lernen, die dritte ist die Kennzeichensegmentierung mithilfe von Bildverarbeitung, die vierte ist die Zeichenerkennung mittels Maschinellem Lernen und die fünfte ist die Anbindung an die Datenbank.

6.4.3.1 Bilddaufnahme

Um ein Bild verarbeiten zu können und aus diesem ein Kennzeichen auslesen zu können, muss

³Topological structural analysis of digitized binary images by border following

zuerst ein Bild vorliegen. Dieses wird über den RaspberryPi mit der RaspberryPi-Kamera aufgenommen. Um das Bild aufzunehmen, muss einfach ein Auslöser aktiviert werden und dann wird das Bild aufgenommen und im richtigen Ordner abgespeichert. Zuvor wird noch überprüft ob sich in diesem Ordner bereits ein Bild befindet und falls eines vorhanden ist wird es gelöscht. Dadurch werden mögliche Fehler durch mehrere Bilder verhindert.

6.4.3.2 Kennzeichenerfassung

Die Kennzeichenerfassung hat die Aufgabe, das Kennzeichen im Eingabebild zu lokalisieren. Dies geschieht mittels Maschinellem Lernen mit dem Modul WPOD-NET von Sérgio Montazolli Silva und Cláudio Rosita Jung⁴. Dieses verwendet zuerst das Modul YOLOv2 welches zur Echtzeitobjekterkennung verwendet werden kann und in dieser Anwendung zur Erkennung von Fahrzeugen verwendet wird. Danach werden die Koordinaten des Kennzeichens ermittelt und dieses aus dem Bild ausgeschnitten und abgespeichert.

6.4.3.3 Kennzeichensegmentierung

Die Kennzeichensegmentierung hat das Ziel die einzelnen Zeichen im Kennzeichen zu separieren und so zu vorbereiten, dass die darauffolgende Zeichenerkennung damit arbeiten kann. Dazu wird das Bild mit dem Kennzeichen zuerst in Graustufen konvertiert, dann mit einem bilateralen Filter gefiltert, mit Thresholding in ein binäres Bild umgewandelt und anschließend mittels Erosion besser erkennbar gemacht. Danach werden im verarbeiteten Bild die Konturen gesucht, sortiert und anhand dieser die einzelnen Zeichen herausgefiltert.

6.4.3.4 Zeichenerkennung

Die letzte Stufe der Kennzeichenerkennung ist die Zeichenerkennung. In dieser werden die einzelnen Zeichen erkannt und als Text abgespeichert. Dies funktioniert über eine eigenes Neuronales Netz basierend auf MobileNetV2⁵, welches mit einem Datensatz von über 35 000 Bildern auf die Erkennung von Zeichen aus Bildern trainiert wurde.

6.4.3.5 Anbindung an Datenbank

Nachdem das Kennzeichen als Text abgespeichert wurde, muss diese Information in die Daten-

⁴License Plate Detection and Recognition in Unconstrained Scenarios

⁵Neuronales Netz für Computer Vision

bank übergeben werden. Dazu wird die eigene API angewandt, welcher man diese Informationen übergeben muss und als Rückgabe die Information bekommt, ob sich das Fahrzeug nun innerhalb oder außerhalb des Parkplatzes befindet.

6.4.4 Ablauf

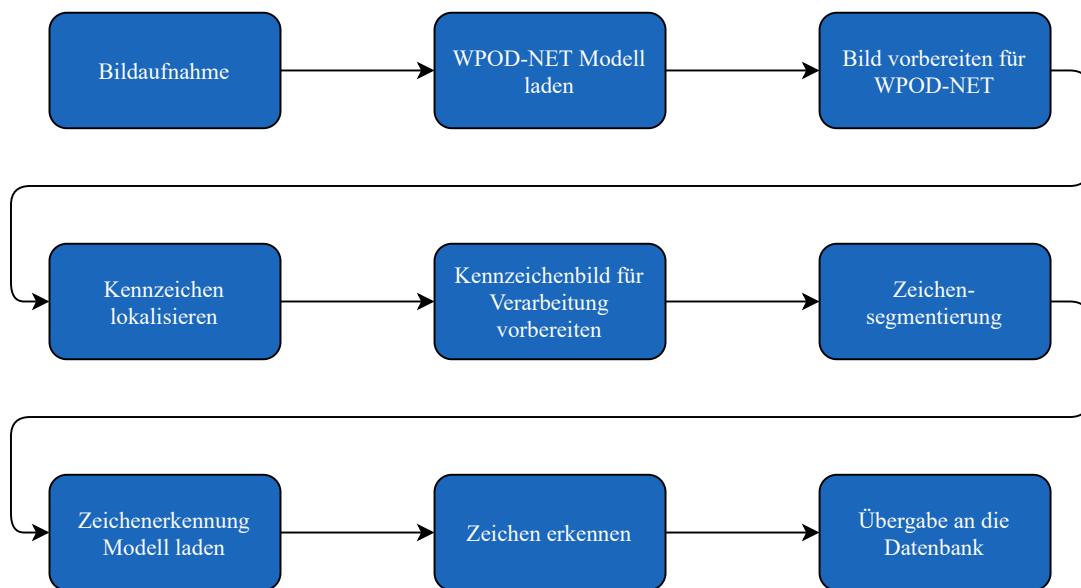


Abbildung 11: Ablaufdiagramm der Kennzeichenerkennung

Im oberen Diagramm ist der Ablauf des Programms angegeben. Zuerst wird mit einem Button der Auslöser betätigt und damit das Foto aufgenommen. Dann wird das erste Modell für Maschinelles Lernen für die Kennzeichenerkennung „WPOD-NET“⁶ geladen. Bevor das Bild diesem Modell übergeben werden kann, muss es noch angepasst werden damit das Modell damit arbeiten kann. Danach kann damit das Kennzeichen im Bild lokalisiert werden. Im Anschluss wird dieses Kennzeichnenbild mit mehreren Bildverarbeitungsalgorithmen verarbeitet, um dann die einzelnen Zeichen zu segmentieren. Danach kann dann das Modell für die Zeichenerkennung geladen werden und dieses dann auch angewendet werden, um das Ergebnis zu erhalten. Dieses Ergebnis wird dann noch mit einer API an die Datenbank übergeben.

⁶Modell für Maschinelles Lernen für Kennzeichenerfassung

6.4.5 Verwendete Libraries

6.4.5.1 Versionsübersicht der verwendeten Libraries

Im Folgenden werden die verwendeten Versionen der benötigten Libraries aufgelistet, um das Programm zu starten. Dies wird noch einmal unterteilt in die Versionen, welche auf Windows 10 benötigt werden und jene auf Raspbian⁷, da auf Raspbian manche Versionen noch nicht verfügbar sind, neuere und verbesserte Versionen verfügbar sind oder manche Versionen nur auf komplizierten Umwegen installierbar sind.

Windows:

- h5py = 2.10.0
- imutils = 0.5.3
- Keras = 2.4.3
- matplotlib = 3.3.2
- notebook = 6.1.5
- numpy = 1.18.5
- opencv-python = 4.4.0.44
- scikit-learn = 0.23.2
- tensorflow = 2.3.1
- requests = 2.24.0

Raspbian:

- h5py = 2.10.0
- imutils = 0.5.3
- Keras = 2.4.3
- matplotlib = 3.3.3

⁷Raspbian ist ein Unix-basiertes Betriebssystem für den Raspberry Pi

- notebook = 6.1.5
- numpy = 1.19.4
- opencv-python = 4.4.0.46
- scikit-learn = 0.24.0
- tensorflow = 2.4.0
- requests = 2.21.0

6.4.5.2 Jupyter Notebook

Jupyter Notebook ist eine nützliche Erweiterung für Python, wenn es um den Bereich der Daten-Visualisierung und Maschinelles Lernen geht. Sie ist eine Unteranwendung des Open-Source Projektes „Project Jupyter“, welches von großen Partnern wie Microsoft und Google unterstützt und verwendet wird. Mit Jupyter Notebook ist es möglich in einer Web-Anwendung ein Live-Script abzuarbeiten und in Kombination mit Matplotlib die Daten visuell darzustellen, abzuspeichern und zu überwachen. Im Gegensatz zur direkten Darstellung in der Entwicklungsumgebung, wird der letzte Durchgang des Scripts, inklusive aller Bilder und Grafiken gespeichert. Zudem ist es auch hervorragend für das Training und die dazugehörende Überwachung von Modulen für Maschinelles Lernen geeignet. Außerdem bietet Jupyter Notebook noch die Möglichkeit mehrere Scripts parallel abzuarbeiten und diese individuell zu überwachen. In dieser Applikation wird es für ein solches Training und für die anschauliche Visualisierung von Daten genutzt.

Die Installation von Jupyter Notebook ist simpel, da man es einfach über pip⁸ mit dem folgenden Befehl installieren kann:

```
1 pip install notebook
```

Code 1: PIP Installation von Jupyter Notebook

Um das Jupyter Notebook aufzurufen muss im Terminal „jupyter notebook“ eingegeben werden und dann öffnet sich automatisch die Web-Applikation. Danach hat man Zugriff auf komplett

⁸Paketverwaltungsprogramm für Python

Dateistruktur des eigenen Projektes und kann die einzelnen Python-Scripts starten. Dabei ist zu beachten, dass man keine gewöhnliche .py-Datei benötigt, sondern eine .ipynb-Datei benötigt, wozu man einfach eine Kopie des normalen Scripts erstellt und die Dateiendung ändert.

6.4.5.3 Numpy

Numpy ist eine weit verbreitete Library für Python, welche sich mit der Berechnung und Verarbeitung von mehrdimensionalen Arrays beschäftigt. Damit können komplexe Berechnungen und Algorithmen oft effizienter verarbeitet werden, wobei aber beachtet werden muss, dass der Code an die mehrdimensionalen Arrays angepasst werden muss. Numpy selbst ist auch eine Voraussetzung für OpenCV, welches Numpy-Arrays verwendet, um Bilder zu verarbeiten. Dazu werden 3-dimensionale Arrays verwendet, wodurch die Verarbeitung dieser Bilder und die Bildverarbeitungsalgorithmen schneller sind. In dieser Applikation wird Numpy in Zusammenarbeit mit OpenCV verwendet und auch für die weitere Verarbeitung der Daten von OpenCV, um zum Beispiel die Koordinaten des Kennzeichens zu speichern.

Um Numpy zu installieren muss folgender Befehl in der Konsole eingegeben werden:

```
1 pip install numpy
```

Code 2: PIP Installation von Numpy

6.4.5.4 OpenCV

OpenCV ist die wichtigste und mächtigste Library die in dieser Applikation verwendet wird. Sie wurde ursprünglich in C++ geschrieben, aber es gibt auch eine Version in Python, welche in dieser Applikation verwendet wird. OpenCV ist eine freie Library für Bildverarbeitung, Computer Vision⁹ und Maschinelles Lernen, welche von Intel gestartet wurde und mittlerweile die wichtigste und am weitesten verbreitete Library in diesem Bereich ist. Sie verwendet Numpy-Arrays, um für eine effiziente Verarbeitung von Bildern zu sorgen. Einige der wichtigsten Funktionen stellen die klassischen Bildverarbeitungsalgorithmen wie zum Beispiel Filterung und Farbraumanpassungen, die Verarbeitung und Auswertung von Kamerabildern und auch die Kompatibilität mit Deep-Learning¹⁰ dar. In

⁹Die Computergestützte Auswertung und Verarbeitung von Kamerabildern

¹⁰Methode für Maschinelles Lernen welche Neuronale Netze nutzt

dieser Applikation wird OpenCV für diverse Bildverarbeitungsalgorithmen, die Zusammenarbeit mit Modellen für Maschinelles Lernen, die Verarbeitung eines Kamerabildes und das allgemeine Arbeiten mit Bildern verwendet.

Die Installation von OpenCV ist dabei etwas komplizierter als bei anderen Libraries.

Windows:

Unter Windows ist die Installation vergleichbar mit anderen Libraries, es muss einfach der folgende Befehl in der Konsole eingegeben werden:

```
1 pip install opencv-python
```

Code 3: PIP Installation von OpenCV

Raspbian:

Auf dem Raspberry Pi gestaltet sich die Installation von OpenCV um einiges schwieriger, da die Installation nicht über pip gemacht werden kann, sondern es manuell kompiliert werden muss.

Als erstes werden alle Tools und Bibliotheken installiert, welche für OpenCV benötigt werden. Dazu verwendet man folgenden Befehl im Terminal:

```
1 $ sudo apt-get install build-essential git cmake pkg-config libjpeg8-dev  
    ↳ libtiff4-dev libjasper-dev libpng12-dev libavcodec-dev  
    ↳ libavformat-dev libswscale-dev libv4l-dev libgtk2.0-dev  
    ↳ libatlas-base-dev gfortran
```

Code 4: Installation benötigter Tools und Bibliotheken für OpenCV

Danach kann man mit dem nächsten Befehl OpenCV von einem GitHub-Repository klonen:

```
1 $ sudo apt-get install build-essential git cmake pkg-config libjpeg8-dev
  ↳ libtiff4-dev libjasper-dev libpng12-dev libavcodec-dev
  ↳ libavformat-dev libswscale-dev libv4l-dev libgtk2.0-dev
  ↳ libatlas-base-dev gfortran
```

Code 5: Klonen von OpenCV von GitHub

Im nächsten Schritt wird OpenCV mit den folgenden Befehlen kompiliert:

```
1 cd ~/opencv && mkdir build && cd build
2
3 cmake -D CMAKE_BUILD_TYPE=RELEASE \
4 -D CMAKE_INSTALL_PREFIX=/usr/local \
5 -D INSTALL_PYTHON_EXAMPLES=ON \
6 -D INSTALL_C_EXAMPLES=ON \
7 -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
8 -D BUILD_EXAMPLES=ON ..
9 make -j4
```

Code 6: Kompilieren von OpenCV

Wenn das Kompilieren erfolgreich beendet wurde, kann OpenCV abschließend installiert werden.

```
1 $ sudo make install && sudo ldconfig
```

Code 7: Abschließende Installation von OpenCV

Danach sollte OpenCV fertig installiert und eingerichtet sein und es kann in einem Python Projekt verwendet werden.

6.4.5.5 Matplotlib

Matplotlib ist eine Python-Library mit welcher Daten und Berechnungen visuell dargestellt werden

können. Damit können statische, animierte und interaktive Diagramme erstellt werden, was die Datenauswertung um einiges erleichtert. Die Syntax ähnelt sehr stark jener von MATLAB, wodurch die Bedienung sehr einfach ist, wenn man schon Erfahrung mit MATLAB hat. Wie auch in MATLAB kann man die Diagramme beliebig anordnen und dadurch das Layout der Darstellung selbst festlegen. Es funktioniert auch hervorragend in Zusammenarbeit mit Jupyter Notebook, wodurch man die Daten in einer Web-Applikation visualisieren kann. Die visuelle Darstellung von Daten ist vor allem in der Entwicklung und beim Arbeiten mit visuellen Ergebnissen wie Bildern von Vorteil. In dieser Applikation wird Matplotlib für die Darstellung der Ergebnisse von Bildverarbeitungsalgorithmen, die Ausgabe von Daten und Resultaten und auch für Test- und Entwicklungszwecke verwendet. Im unteren Bild kann man ein Beispiel sehen bei welchem Matplotlib verwendet wird, um einige Bilder mit einem Titel anzuzeigen.

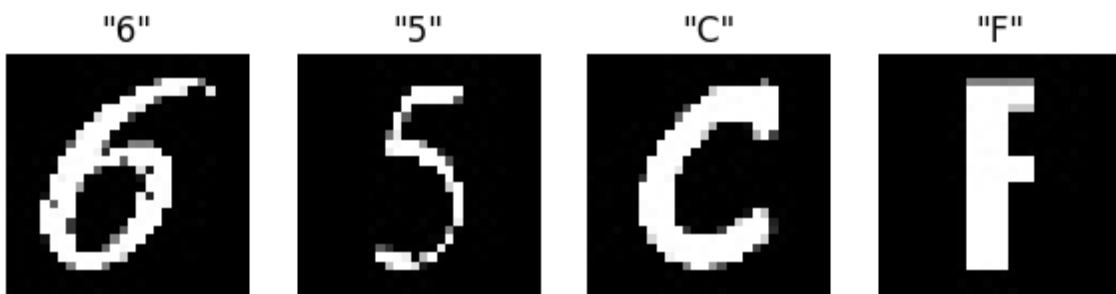


Abbildung 12: Beispiel einer Anwendung von Matplotlib

Um Matplotlib zu installieren muss das Folgende in der Konsole eingegeben werden:

```
1 pip install matplotlib
```

Code 8: PIP Installation von Matplotlib

6.4.5.6 Keras

Keras ist eine Deep-Learning Schnittstelle für diverse Machine Learning Frameworks wie zum Beispiel Tensorflow oder Theanos. Damit wird die Bedienung und Anwendung dieser Frameworks vereinfacht und es bietet auch diverse Funktionen, um Inputs mit den Modellen für Maschinelles Lernen kompatibel zu machen. Keras ist ein Teil von Tensorflow, wird aber eigenständig weiterentwickelt, um die Kompatibilität mit anderen Machine Learning Frameworks aufrechtzuerhalten. In

dieser Applikation wird es in Zusammenarbeit mit Tensorflow verwendet, um mit den verwendeten Modellen für Maschinelles Lernen zu arbeiten.

Um Keras zu installieren führt man folgenden Befehl in der Konsole aus:

```
1 pip install keras
```

Code 9: PIP Installation von Keras

6.4.5.7 Tensorflow

Tensorflow ist eines der weltweit beliebtesten Frameworks für Maschinelles Lernen und wird von weltweit erfolgreichen Firmen wie zum Beispiel Google, AMD oder auch Intel verwendet. Es bietet eine umfassende Plattformen für jegliche Anwendungen für Maschinelles Lernen und ist in Zusammenarbeit mit APIs wie zum Beispiel Keras leicht zu verwenden. In dieser Applikation wird damit ein neuronales Netz trainiert und mehrere Modelle für Maschinelles Lernen verwaltet und verwendet.

Bei der Installation von Tensorflow muss beachtet werden, dass sich diese von Windows zu Raspbian stark unterscheidet. Raspbian unterstützt offiziell die benötigte Version von Tensorflow noch nicht und deswegen muss dieses über ein paar Umwege installiert werden.

Windows:

Die Installation von Tensorflow unter Windows ist einfach, da man es wie andere Libraries einfach über pip installieren kann.

```
1 pip install tensorflow
```

Code 10: PIP Installation von Tensorflow

Raspbian:

Bei Raspbian muss Tensorflow manuell kompiliert werden, da die aktuelle Version noch nicht offiziell unterstützt wird. Diese funktioniert aber trotz dieses Umweges einwandfrei und wird mit den folgenden Befehlen in der Konsole installiert.

Mit dem ersten Befehl werden die benötigten Tools installiert:

```
1 $ sudo apt-get install cmake curl
```

Code 11: Benötigte Tools für Tensorflow

Danach kann man die neuste Tensorflow Version von GitHub herunterladen:

```
1 $ wget -O tensorflow.zip  
→ https://github.com/tensorflow/tensorflow/archive/v2.4.0.zip
```

Code 12: Tensorflow von GitHub herunterladen

Anschließend muss Tensorflow entpackt werden:

```
1 $ unzip tensorflow.zip  
2 $ mv tensorflow-2.4.0 tensorflow  
3 $ cd tensorflow
```

Code 13: Entpacken von Tensorflow

Im nächsten Schritt müssen noch zusätzlich erforderliche Libraries installiert werden:

```
1 $ ./tensorflow/lite/tools/make/download_dependencies.sh
```

Code 14: Zusätzlich erforderliche Librarys

Danach kann die Installation kompiliert werden:

```
1 $ ./tensorflow/lite/tools/make/build_aarch32_lib.sh
```

Code 15: Kompilieren von Tensorflow

Danach muss man die Installation mit den folgenden Befehlen abschließen:

```
1 $ cd ~/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers
2 $ mkdir build
3 $ cd build
4 $ cmake ..
5 $ make -j4
6 $ sudo make install
7 $ sudo ldconfig
```

Code 16: Abschließen der Installation von Tensorflow

Nach diesem Schritt sollte Tensorflow Installation funktionieren und man kann es wie jede andere Library in Python einbinden.

6.4.5.8 local_utils

local_utils ist ein einfaches Python-Script, welches die Arbeit mit WPOD-NET im Bereich der Kennzeichenerkennung vereinfacht und in dieser Applikation für die einfachere Nutzung von WPOD-NET verwendet wird. Es beinhaltet die Funktion „detect_lp“ mit welcher ein Bild des Kennzeichens und die Koordinaten des Kennzeichens ermittelt werden können. Dieses Script kann über den folgenden Link von GitHub heruntergeladen werden: https://github.com/quangnhat185/Plate_detect_and_recognize/blob/master/local_utils.py

6.4.5.9 Scikit-learn

Scikit-learn ist eine freie Library für Maschinelles Lernen und basiert auf Numpy und Matplotlib. Es wird vor allem verwendet, um mit großen visuellen Datensätzen neuronale Netze zu trainieren. Es bietet Funktionen, um Modelle anzupassen, Daten vorzubereiten, Modelle zu trainieren und

ähnliches. In dieser Applikation wird es für die Normalisierung von Labels verwendet, um diese für Modelle für Maschinelles Lernen anzupassen und für das zufällige Aufteilen von Arrays in Test und Trainingsteile, welche für das Training eines Neuronalen Netzes benötigt werden.

Für die Installation von Scikit-learn muss die folgende Zeile in der Konsole ausgeführt werden:

```
1 pip install scikit-learn
```

Code 17: PIP Installation von Scikit-learn

6.4.5.10 Requests

Requests ist eine Library mit welcher HTTP-Anfragen in Python vereinfacht werden. Sie wird häufig verwendet, um mit APIs zu kommunizieren und ist eine der beliebtesten Python Librarys, da sie für API-Anwendungen so gut wie notwendig ist. In dieser Applikation wird sie für die Kommunikation zu einer eigenen API für den Datenbankzugriff verwendet. Dadurch ist es ohne größere Schwierigkeiten möglich, das Bild des Kennzeichens, die ID für den jeweiligen Parkplatz und das Resultat des Kennzeichens an die eigene Datenbank zu senden und auch eine Antwort zu erhalten, ob der Zugriff erfolgreich war. Um die Daten mit dieser Library zu übertragen müssen diese einfach nur in einem Dictionary eingetragen werden und dann mit dem korrekten Schlüsselwort über die API gesendet werden.

Die Installation von Requests erfolgt mit folgendem Befehl in der Konsole:

```
1 pip install requests
```

Code 18: PIP Installation von Requests

6.4.5.11 Gpiozero

Gpiozero ist eine Python-Library für den Raspberry Pi. Mit ihr ist es möglich auf die GPIO-Pins des Raspberry zuzugreifen, Signale von diesen auszulesen und Signale an diesen auszugeben. Dies wird oft verwendet, um Sensoren auszuwerten oder um Aktoren zu steuern, da diese Pins frei

programmierbar sind, wodurch man jede beliebige Anwendung realisieren kann. Der Raspberry Pi kann zwar nur Aktoren in seiner Leistungskategorie ansteuern, dies kann aber zum Beispiel mit einem Relais umgangen werden, wodurch der Raspberry Pi zu einem mächtigen Werkzeug für die Sensorik und Aktorik wird. Ein Beispiel für die möglichen Anwendungen ist ein Button oder eine Lampe. In dieser Applikation wird mit dieser Library ein gedrückter Button detektiert, welcher als Auslöser für die Kamera fungiert.

In der folgenden Abbildung sieht man ein Bild der GPIO-Pins des Raspberry Pi, welche mit dieser Library angesteuert werden können:

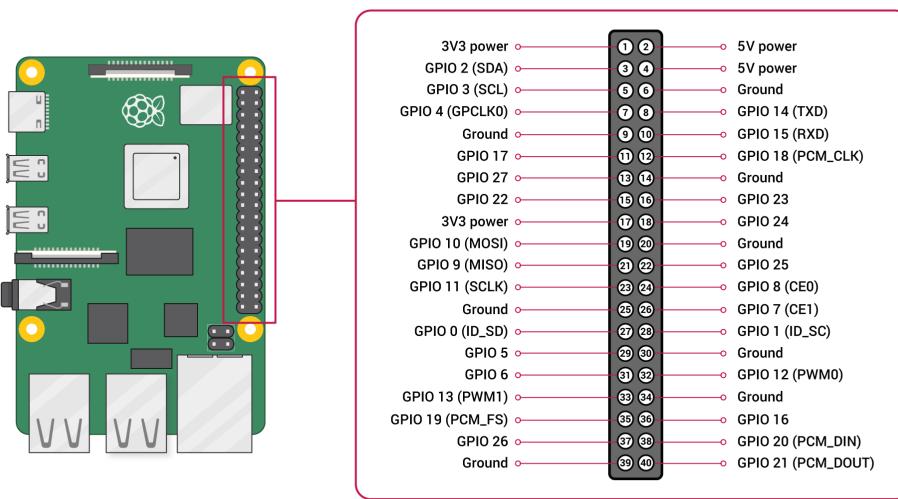


Abbildung 13: GPIO-Pins des Raspberry Pi

Diese Library kann nur auf dem Raspberry Pi installiert werden. Um die Library Gpiozero zu installieren kann pip verwendet werden:

```
1 pip install gpiozero
```

Code 19: PIP Installation von Gpiozero

6.4.5.12 Picamera

Die Library picamera wird verwendet, um auf das Kameramodul des Raspberry Pi zuzugreifen.

Die Library bietet etliche Funktionen an, mit welchen die Kamera gesteuert werden kann, wie zum Beispiel Fotos aufzunehmen, Videoaufnahmen zu starten und zu beenden, Fotoeinstellungen zu ändern und ähnliches. Die Library ist notwendig, wenn man über Python das Kameramodul bedienen will. In dieser Applikation wird sie verwendet, um ein Vorschaubild des aufgenommenen Bildes auf dem Bildschirm darzustellen, falls ein Bildschirm angeschlossen wird und um im Anschluss das Bild aufzunehmen und korrekt zu skalieren.

Die Installation von picamera ist nur auf dem Raspberry Pi möglich, da nur dieser damit arbeiten kann. Für die Installation von picamera muss folgender Befehl in der Konsole eingegeben werden:

```
1 pip install picamera
```

Code 20: PIP Installation von Picamera

6.4.6 Wichtige Programmteile

Im folgenden Abschnitt werden die wichtigsten Teile des Kennzeichenerkennungsprogrammes genauer betrachtet und erklärt.

6.4.6.1 WPOD-NET-Modell laden

```

1  def load_model(path):
2
3      try:
4
5          path = splitext(path)[0]
6
7          with open ('%s.json' % path, 'r') as json_file:
8
9              model_json = json_file.read()
10
11             model = model_from_json(model_json, custom_objects={})
12
13             model.load_weights('%s.h5' % path)
14
15             print("-> Model loading finished")
16
17             return model
18
19     except Exception as e:
20
21         print(e)

```

Code 21: WPOD-NET Modell laden

Für das Lokalisieren des Kennzeichens wird das Modell für Maschinelles Lernen WPOD-NET verwendet. Dieses basiert auf YOLOv2, mit welchem eine Echtzeitobjekterkennung möglich ist und wurde für diese Anwendung dazu angepasst, dass es möglich ist Kennzeichen in einem Bild zu lokalisieren. WPOD-NET versucht zuerst auf Grundlage von YOLOv2 in dem übergebenen Bild ein Fahrzeug zu erkennen, erstellt dann auf dieser Grundlage eine Region, in welcher sich mit hoher Wahrscheinlichkeit das Kennzeichen befindet und liefert dann die Koordinaten des Kennzeichens im Bild. Im oberen Code sieht man die Funktion „load_model“ mit welcher das WPOD-Modell mit den gewichteten Werten geladen wird. Dazu muss dieser Funktion nur der Pfad für das WPOD-NET File übergeben werden. Die Funktion versucht dann ein .json-File zu öffnen und erstellt aus diesem mit der Funktion „model_from_json“ ein Model mit dem WPOD-NET arbeiten kann. Danach wird im selben Ordner in dem auch das .json-File gefunden wurde nach einem .h5-File gesucht. Dieses enthält die gewichteten Werte für das Modell und wird dann geladen. Wenn alles funktioniert hat, erhält man als Rückgabe das Modell mit den geladenen Werten. Im unteren Code sieht man die Anwendung dieser Funktion. Zuerst wird dabei der Pfad für das WPOD-NET File definiert und dieser Pfad wird dann der Funktion „load_model“ übergeben und das Modell als Rückgabewert in der Variable „wpod_net“ abgespeichert.

```

1   wpod_net_path = "wpod-net.json"
2   wpod_net = load_model(wpod_net_path)

```

Code 22: Anwendung des WPOD-NET Modells

6.4.6.2 Kennzeichen lokalisieren

```

1   def get_plate(image_path, Dmax=608, Dmin=256):
2       vehicle = preprocess_raw_image(image_path)
3       ratio = float(max(vehicle.shape[:2])) / min(vehicle.shape[:2]))
4       side = int(ratio * Dmin)
5       bound_dim = min(side, Dmax)
6       _, LpImg, _ , cor = detect_lp(wpod_net, vehicle, bound_dim,
7                                       lp_threshold=0.5)
7       return LpImg, cor

```

Code 23: get_plate

Um das Kennzeichen zu lokalisieren wird die Funktion „get_plate“ angewendet. Dieser Funktion muss man den Pfad für das aufgenommene Bild übergeben. Danach wird auf dieses Bild die Funktion „preprocess_raw_image“ angewandt, welches im folgenden Code ersichtlich ist:

```

1   def preprocess_raw_image(image_path):
2       img = cv2.imread(image_path)
3       img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
4       img = img / 255
5       img = cv2.resize(img, (224,224))
6       return img

```

Code 24: Bild vorbereiten für WPOD-NET

Die Funktion „preprocess_raw_image“ liest das Bild mittels OpenCV ein, ändert dann den Farbraum von BGR zu RGB da OpenCV das Bild mit BGR einliest, WPOD-NET aber RGB benötigt und anschließend wird noch die Größe des Bildes angepasst.

Danach geht es wieder in der Funktion „get_plate“ weiter. Dort wird danach in den nächsten Codezeilen das Seitenverhältnis des Bildes definiert und in einer Variable abgespeichert. Dies ist für die nächste Funktion „detect_lp“ notwendig. Diese Funktion ist eine Hilfsfunktion für den Umgang mit WPOD-NET, welcher man das aufgenommene und vorbereitete Bild übergeben muss und als Rückgabe ein Bild des Kennzeichens und die Koordinaten des Kennzeichens zurückgibt. Im unteren Codeteil befindet sich die Anwendung der Funktion „get_plate“ wobei bei erfolgreicher Anwendung der Funktion noch zusätzlich die Koordinaten des Kennzeichens im Terminal ausgegeben werden.

```
1 input_image = image_path[0]
2 LpImg,cor = get_plate(input_image)
3 print("-> License Plate found in:",splitext(basename(input_image))[0])
4 print("-> Coordinates of License Plate: \n", cor)
```

Code 25: Kennzeichen lokalisieren

6.4.6.3 Bild aufnehmen

```
1 #Kamerabildvorschau
2 camera.start_preview(alpha=220)

3

4 #Warten bis Button gedrückt wird
5 button.wait_for_press()

6

7 #Altes Bild löschen
8 try:
9     os.remove('/home/pi/Documents/Kennzeichenerkennung/Plate_examples
10     /image.jpg')
```

```
10 except OSError:  
11     pass  
12  
13 #Bild schließen und abspeichern  
14 camera.capture('/home/pi/Documents/Kennzeichenerkennung/Plate_examples  
→ /image.jpg')  
15 camera.stop_preview()
```

Code 26: Bild aufnehmen

In diesem Codeteil befindet sich die Aufnahme des Fahrzeuggbildes. Im ersten Schritt wird dabei die Kameravorschau geöffnet, damit es bei einem angeschlossenen Bildschirm einfacher ist das Foto aufzunehmen. Diese Vorschau ist leicht durchsichtig auf dem Bildschirm damit man trotzdem alles andere bedienen kann. Danach wird gewartet bis der Button gedrückt wird. Auf diese Weise dient der Button als Auslöser für die Kamera. Wenn der Button gedrückt wurde, wird zuerst versucht im gewünschten Zielordner das vorhergehende Bild zu löschen damit es zwischen den Bildern zu keinem Konflikt kommt. Dieses vorhergehende Bild wird auch nicht wieder benötigt, da das Bild des Kennzeichens im vorhergehenden Programmdurchlauf schon an die Datenbank übergeben wurde. Erst danach wird das eigentliche Foto für diesen Programmdurchlauf aufgenommen und im Zielordner abgespeichert. Danach wird die Kameravorschau wieder beendet.

6.4.6.4 Zeichen mittels Konturerkennung finden

```
1 cnt, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL,  
→ cv2.CHAIN_APPROX_SIMPLE)
```

Code 27: Konturen finden

Der nächste wichtige Abschnitt des Programmes beschäftigt sich damit, aus dem Kennzeichenbild die einzelnen Zeichen herauszusuchen und einzeln darzustellen, damit diese dann von einem weiteren Modell für Maschinelles Lernen erkannt werden können. Dazu wird als erstes die Funktion

„findContours“ von OpenCV angewendet, welcher man ein binäres Bild des Kennzeichens übergeben muss und als Rückgabe ein Array von Koordinaten aller gefundenen Konturen ausgibt.

```

1  def sort_contours(contours):
2      i = 0
3      boundingBoxes = [cv2.boundingRect(c) for c in contours]
4      (contours, boundingBoxes) = zip(*sorted(zip(contours, boundingBoxes),
5          key=lambda b: b[1][i], reverse = False))
6
7      return contours

```

Code 28: Konturen sortieren

Im nächsten Schritt wird die Funktion „sort_contours“ benötigt, welche die Konturen aus dem vorherigen Array sortiert. Sortieren bedeutet dabei, dass zuerst Rechtecke bei den Konturen aufgespannt werden und mit diesen die Konturen von links nach rechts geordnet werden, damit sich am Ende beim Resultat des Kennzeichens die korrekte Reihenfolge ergibt.

```

1  for c in sort_contours(cnt):
2      (x, y, w, h) = cv2.boundingRect(c)
3      if h / lp_img.shape[0] >= 0.5 and h / lp_img.shape[0] <= 0.95:
4          seperated = erosion[y:y+h,x:x+w]
5          seperated = cv2.resize(seperated, dsize=(30, 60))
6          hierarchy, seperated = cv2.threshold(seperated, 220, 255,
7              cv2.THRESH_BINARY + cv2.THRESH_OTSU)
8          found_characters.append(seperated)
9
10     print("-> Detected {} characters".format(len(found_characters)))

```

Code 29: Filterung der korrekten Konturen

6.4.6.5 Bildverarbeitung

Im gesamten Programm wird ab und zu ein Bildverarbeitungsalgorithmus verwendet, aber zwischen der Kennzeichenlokalisierung und der Zeichenerfassung befindet sich der Hauptteil davon. Um die einzelnen Zeichen zu lokalisieren müssen zuvor einige Bildverarbeitungsalgorithmen und Funktionen angewandt werden und an anderen Stellen werden meistens die gleichen Funktionen angewandt wie dort. Im folgenden Code sieht man diese Bildverarbeitung:

```
1 lp_img = cv2.convertScaleAbs(LpImg[0], alpha=(255))
2 api_img = cv2.cvtColor(lp_img, cv2.COLOR_BGR2RGB)
3 gray = cv2.cvtColor(lp_img, cv2.COLOR_BGR2GRAY)
4 blur = cv2.bilateralFilter(gray, 11, 15, 15)
5 thresh = cv2.threshold(blur, 127, 255, cv2.THRESH_BINARY_INV +
→ cv2.THRESH_OTSU)[1]
6 kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
7 erosion = cv2.erode(thresh, kernel, iterations=1)
```

Code 30: Bildverarbeitungsalgorithmen

In der ersten Zeile wird auf das Kennzeichenbild die Funktion „convertScaleAbs“ angewandt welche jeden Array-Eintrag zu einem Farbwert konvertiert mit dem OpenCV arbeiten kann. Danach wird in einem eigenen Schritt der Farbraum von BGR zu RGB gewechselt, da die API für die Übertragung zur Datenbank ein RGB-Bild des Kennzeichens benötigt. Für die eigentliche Weiterverarbeitung wird der Farbraum in der nächsten Zeile von BGR zu einem Graustufenbild geändert. Anschließend wird darauf ein bilaterales Filter angewandt, welches unnötiges Bildrauschen entfernt, um die weiteren Schritte zu vereinfachen. Danach wird auf das dadurch entstandene Bild Thresholding angewandt, um es in ein binäres Bild zu konvertieren, welches für die Konturerkennung notwendig ist. Um die Qualität dieses binären Bildes noch etwas zu verbessern wird danach mit einem 3x3 Kernel Erosion verwendet. Danach ist das Bild fertig vorbereitet für die Konturerkennung.

6.4.6.6 Visualisierung mittels Matplotlib

Um die einzelnen Bildverarbeitungsmechanismen zu überprüfen, die Werte anzupassen oder relevante Daten darzustellen, ist es hilfreich mit Hilfe der Library Matplotlib eine Visualisierung

durchzuführen. Zusätzlich wird auch noch Jupyter Notebook verwendet, wodurch die Visualisierung nur in der Web-Applikation sichtbar ist, und damit werden nicht unzählige Tabs geöffnet. Im unteren Code sieht man eine solche Anwendung von Matplotlib.

```
1  plt.figure(figsize=(10,5))
2  plt.subplot(2,2,1)
3  plt.axis(False)
4  plt.imshow(gray, cmap='gray', vmin = 0, vmax = 255)
5  plt.subplot(2,2,2)
6  plt.axis(False)
7  plt.imshow(blur, cmap='gray', vmin = 0, vmax = 255)
8  plt.subplot(2,2,3)
9  plt.axis(False)
10 plt.imshow(thresh, cmap='gray', vmin = 0, vmax = 255)
11 plt.subplot(2,2,4)
12 plt.axis(False)
13 plt.imshow(erosion, cmap='gray', vmin = 0, vmax = 255)
```

Code 31: Anwendung einer Visualisierung mit Matplotlib

Prinzipiell funktioniert die Library Matplotlib gleich wie eine figure in MATLAB. Im oberen Codeteil werden damit die einzelnen Bildverarbeitungsalgorithmen dargestellt. Zuerst muss eine figure erstellt werden, in welcher die einzelnen Subplots platziert werden. Damit kann man die zusammengehörenden Visualisierungen gruppieren. Danach muss für jedes Bild ein eigener Subplot erzeugt werden, welchen man mit den mitgegebenen Parametern in der figure platzieren kann. Danach wird bei allen Bildern die Achsenbeschriftung deaktiviert, da diese hier nicht benötigt wird. Danach wird das gewünschte Bild ausgewählt und da alle Bilder in Graustufen sind, wird diese angepasst und der Farbwertebereich auf 0 bis 255 gesetzt. Im unteren Bild sieht man die Ausgabe dieser Visualisierung.

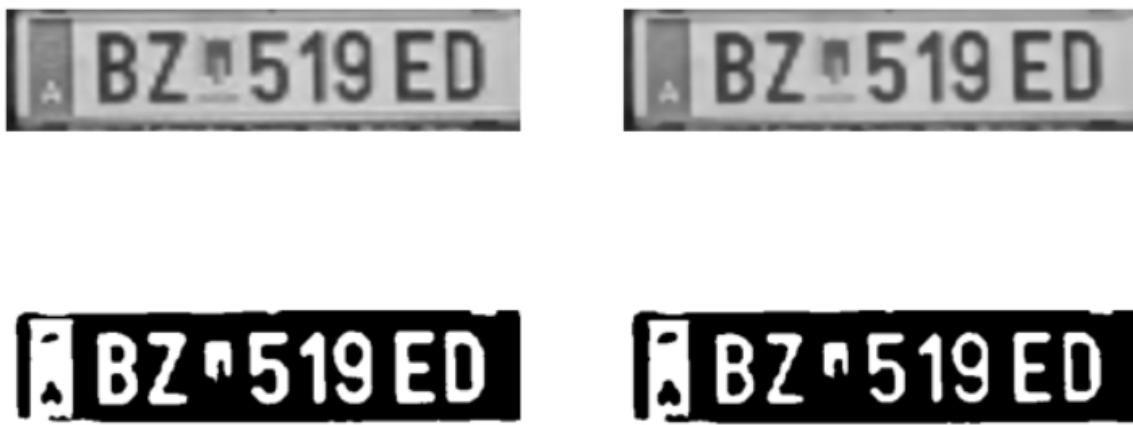


Abbildung 14: Visualisierung mit Matplotlib

6.4.6.7 Modell für Zeichenerkennung laden und anwenden

Für die Zeichenerkennung wird ein eigenes Neuronales Netz verwendet, welches auf MobileNetV2 basiert. Dafür wird primär die Funktion „predict_from_model“ verwendet, welche im unteren Code ersichtlich ist.

```

1 def predict_from_model(image, model, labels):
2     image = cv2.resize(image,(80, 80))
3     image = npy.stack((image,),*3, axis = -1)
4     prediction = labels.inverse_transform([npy.argmax
5         → (model.predict(image[npy.newaxis,:,:]))])
6     return prediction

```

Code 32: predict_from_model

Vor der Anwendung dieser Funktion muss wieder, wie beim ersten Modell für Maschinelles Lernen, das Modell zuerst geladen werden. Danach kann diese Funktion verwendet werden, welcher man das Bild einer Zeichens und das Model übergeben muss. Die Funktion „predict_from_model“ wendet dann nur dieses Modell an und übergibt das vermutete Ergebnis in ein Array.

```

1  fig = plt.figure(figsize = (15,3))
2  cols = len(found_characters)
3  grid = gridspec.GridSpec(ncols= cols, nrows= 1, figure = fig)
4
5  result = ''
6  for i, character in enumerate(found_characters):
7      fig.add_subplot(grid[i])
8      title = npy.array2string(predict_from_model(character, model,
9          labels))
10     plt.title('{}'.format(title.strip("[]"), fontsize=20))
11     result += title.strip("[]")
12     plt.axis(False)
13
14     plt.imshow(character, cmap='gray')
15
16 print("-> License Plate:", result)

```

Code 33: Anwendung der Zeichenerkennung

Im oberen Bild sieht man die Anwendung der vorherigen Funktion. Prinzipiell wird für jedes vorher gefundene Zeichen das Modell angewendet und das Resultat von einem Array zu einem String konvertiert. Dieser String wird dann zum Ergebnis-String hinzugefügt, wodurch man am Ende das komplette Kennzeichen in einem String hat. Zusätzlich wird das komplette Kennzeichen mit den vermuteten Ergebnissen mit Matplotlib visualisiert. Dies ist im unteren Bild ersichtlich.



Abbildung 15: Visualisiertes Ergebnis der Kennzeichenerkennung

6.4.6.8 Resultat mittels API an Datenbank senden

Wenn das Programm zu einem Resultat gekommen ist, muss dieses noch an die Datenbank übergeben werden. Der dafür zuständige Code ist im nächsten Codeteil ersichtlich.

```

1 #API-Definitionen
2
3 url = "http://dev.philipp-kraft.com/api/v1/detections"
4
5 lp_result = {}
6
7 lp_files = {}
8
9 lp_result["name"] = result
10 lp_result["parking_lot_id"] = "2"      #Parkplatz-ID
11 lp_files=[]
12   ('image',('kennzeichen.png',open('lp_image/lp.png','rb')), 'image/png')
13 ]
14
15 header = {"Authorization": "Bearer"
16   ↪  "5|4NcJgCrR9FkeTFeTvRGHLQqoJBeY8IDqxborpuuS"}
17
18
19 #Senden der Daten an die Datenbank und Ausgabe der Response
20
21 resp = requests.post(url, data=lp_result, files=lp_files, headers=header)
22 print("-> API-Response:",resp.text)

```

Code 34: Senden der Ergebnisse an die Datenbank

Im ersten Schritt werden die URL für die API und für die Ergebnisse zwei Dictionaries erstellt. In das erste Dictionary wird der String für das Kennzeichen und die ID-Nummer des Parkplatzes eingetragen. In das zweite Dictionary kommt das Bild des Kennzeichens. Im nächsten Schritt muss noch ein Bearer Token definiert werden, welcher auf der APM-Website generiert werden kann. Dieser wird zur Autorisierung benötigt. Danach können diese Daten über einen Post-Befehl an die API gesendet werden. Um den Status dieser Kommunikation zu überprüfen wird im Anschluss noch die Antwort der API im Terminal ausgegeben. Bei erfolgreicher Kommunikation antwortet diese mit dem aktuellen Status des Fahrzeuges auf dem Parkplatz.

6.5 Raspberry Pi

6.5.1 Einleitung

Damit die Software der Kennzeichenerkennung arbeiten kann, benötigt es eine geeignete Hardware. Diese muss dabei die folgenden Eigenschaften aufweisen, um für die Kennzeichenerkennung geeignet zu sein:

- Möglichst klein
- Nicht zu teuer
- Möglichkeit eine Kamera anzuschließen
- Schnell
- Internetanbindung

6.5.2 Wahl des Raspberry Pi

In dieser Applikation wird ein RaspberryPi 4B 2GB verwendet, da er alle zuvor genannten Eigenschaften am besten erfüllt.

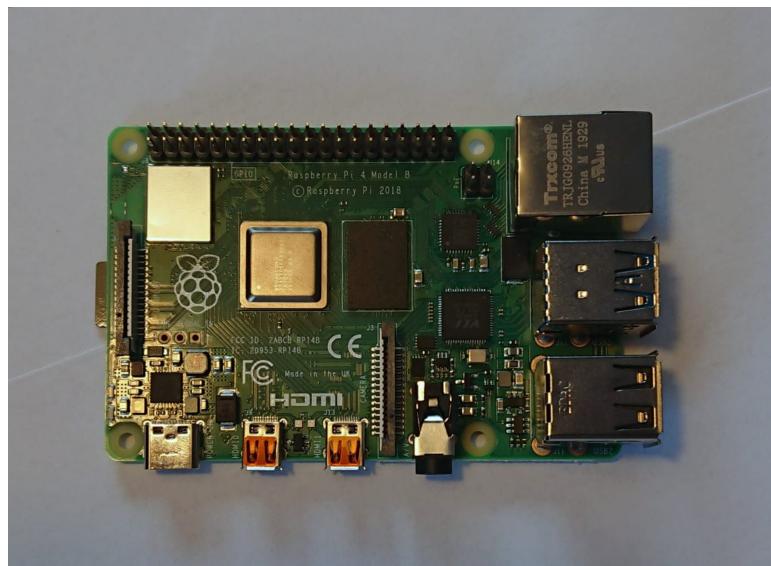


Abbildung 16: Raspberry Pi

Er hat die Größe einer Scheckkarte, wodurch es möglich ist ein kompaktes Gehäuse zu bauen, welches man einfach montieren kann. Er ist mit 35€ pro Stück nicht zu teuer. Er hat einen integrierten Kameraanschluss und es gibt unzählige Kameras, welche damit kompatibel sind. Er hat für seine Größe eine sehr gute Rechenkraft und ist damit in der Lage das komplette Programm für die Kennzeichenerkennung in einer akzeptablen Zeit abzuarbeiten. Er besitzt außerdem einen Ethernet-Anschluss und ein WLAN-Modul, wodurch er sehr einfach mit dem Internet verbunden werden kann.

6.5.3 Kamera

Für den Raspberry Pi muss zudem noch die passende Kamera ausgewählt werden, um die Bilder von den Kennzeichen aufzunehmen. Dafür gibt es Unmengen an kompatiblen Kameras, aber hier wird das Originalzubehör von RaspberryPi verwendet. Dies hat den Grund, dass diese Kamera leicht zu verwenden, sehr klein, mit Schrauben einfach zu befestigen und günstig ist. Zudem liefert sie ein qualitativ hochwertiges Bild, welches für die Software gut verarbeitbar ist.



Abbildung 17: Raspberry Pi mit Kamera

6.6 Maschinelles Lernen

6.6.1 Einleitung

Die Kennzeichenerkennung ist eine sehr komplexe Anwendung, da es unzählige Kennzeichen gibt und jedes einzelne davon erkannt werden sollte. Dabei ist nicht nur auf verschiedene Länderkennzeichen zu achten, welche sich stark voneinander unterscheiden, sondern auch auf Sonderkennzeichen, welche sich sogar auf nationaler Ebene von den normalen Kennzeichen unterscheiden. Zudem besteht noch die Problematik, dass sich die Kennzeichen oft in unterschiedlichen Zuständen befinden, wie zum Beispiel mit Dreck beschmutzt oder ähnliches. Das bedeutet, dass eine in der Praxis anwendbare Kennzeichenerkennung mit all diesen Gegebenheiten zureckkommen muss, damit das Kennzeichen in nahezu jeder möglichen Situation erkannt werden kann. Um dies zu erreichen wird das Programm für die Kennzeichenerkennung in drei größere Teile aufgeteilt. Die Kennzeichenlokalisierung, die Zeichenerfassung und die Zeichenerkennung. Die Zeichenerfassung gestaltet sich relativ einfach, da diese bereits ein Bild des Kennzeichens erhält, in dem ansonsten keine größeren Störfaktoren beinhaltet sind. Diese muss also nur die Konturen innerhalb dieses Bildes finden und diese so sortieren, dass nur noch die einzelnen Zeichen übrig bleiben. Bei der Kennzeichenlokalisierung und der Zeichenerfassung gestaltet sich die Lage schon etwas schwieriger. Die Kennzeichenlokalisierung steht vor dem Problem, dass wenn man klassisch mit der Bildverarbeitung nach einer passenden Kontur für das Kennzeichen sucht, kann dieser Algorithmus durch Störfaktoren wie ein zufällig ähnlich großes Orts- oder Straßenschild getäuscht werden, wodurch es zu einem Fehler kommt. Auch Fenster oder ähnlich proportionierte Gegenstände können zu solch einem Fehler führen. Die Lösung dafür befindet sich im Bereich des Maschinellen Lernens. Anstatt nur nach einer Kontur zu suchen, wird mit solch einem Modell gezielt nach einem Kennzeichen gesucht. Dies ist möglich, da so ein Modell mit vielen möglichen Eingaben und den passenden Ausgaben dazu trainiert wird ein Muster zwischen den Ein- und Ausgaben zu finden, mit welchem es dann ähnliche Probleme selbstständig lösen kann. Auch bei der Zeichenerkennung bietet sich solch ein Modell an, da es darauf trainiert werden kann, einzelne Zeichen mit einer hohen Genauigkeit zu erkennen. Hier wäre es zwar ohne Maschinelles Lernen auch möglich ein relativ gutes Ergebnis zu erzielen, indem man mittels Korrelation von Vergleichsbildern das Zeichen findet, welches am wahrscheinlichsten passen würde, aber in dieser Applikation wird auch hierfür ein Modell für Maschinelles Lernen verwendet.

6.6.2 Definition

Maschinelles Lernen beschreibt ein Modell, welches zuerst in einer Trainingsphase trainiert werden muss, um danach ähnliche Probleme näherungsweise lösen zu können. Dazu wird in der Trainingsphase ein Datensatz bereitgestellt welcher mögliche Eingaben und meistens auch die passenden Ausgaben enthält. Durch die vielen verschiedenen Eingaben und Ausgaben ist es für das Modell möglich ein Muster zwischen diesen zu erkennen, welches durch dieses Training verbessert werden kann. Es gilt dabei, je länger die Trainingsphase dauert desto genauer wird das Modell. Der Aufwand besteht dabei darin, die gigantischen Datensätze zur Verfügung zu stellen, da diese oft zuvor erstellt werden müssen. Zudem ist das Training ein sehr rechenintensiver Vorgang, welcher für ein gutes Ergebnis oft mehrere Stunden benötigt. In den letzten Jahren hat sich im Bereich des Modell Trainings eine neue Entwicklung namens „Deep Learning“ verbreitet, mit welcher eine höhere Genauigkeit als mit bisherigen Modellen erreicht werden kann. Dabei werden künstliche Neuronale Netze erstellt, welche der Vernetzung von Neuronen in Lebewesen nachempfunden sind. Diese künstlichen Neuronen sind Verknüpfungspunkte zwischen Eingang und Ausgang des Modells und sind je nach Komplexität in verschiedene Schichten aufgeteilt. Um diese künstlichen Neuronen zu trainieren, werden deren Gewichtungswerte angepasst. Deswegen muss bei solchen Modellen auch oft eine Datei mit den gewichteten Werten eingebunden werden.

6.6.3 Vergleich Maschinelles Lernen / Bildverarbeitung

Zum Beginn dieser Applikation wurde ein Ansatz ohne den Einsatz von Maschinellem Lernen verfolgt. Dazu wurde ein Testprogramm geschrieben, welches das Kennzeichen über eine Konturerkennung lokalisieren sollte. Dies führte aber zu häufig auftretenden Fehlern. Einer dieser Fehler ist in der unteren Abbildung ersichtlich.



Abbildung 18: Fehler bei Konturerkennung

Wie in der oberen Abbildung erkennbar ist, hat das Programm nicht das Kennzeichen erkannt, sondern ein Fenster im Hintergrund, welches eine ähnliche Kontur wie das Fenster aufweist. Dies ist deswegen aufgetreten, da das Kennzeichen über seine Kontur gefunden werden sollte und dazu alle Konturen im Bild aufgelistet werden und aus dieser Liste dann das Kennzeichen herausgesucht werden sollte. Dies führte in dieser Testversion aber nahezu nie zu einem Ergebnis, welches in der Praxis anwendbar gewesen wäre. Durch eine Anpassung der Software wäre es zwar möglich gewesen die Genauigkeit zu verbessern und dafür zu sorgen, dass mit einer größeren Wahrscheinlichkeit die richtige Kontur ausgewählt wird, aber bei einer Kontur mit den gleichen Proportionen wie ein Kennzeichen, wäre die Konturerkennung wieder schnell an ihre Grenzen gestoßen. Deswegen wurde der Ansatz komplett verändert und für die Kennzeichenlokalisierung ein Modell für Maschinelles Lernen verwendet, welches auf deutlich elegantere und effizientere Weise ein besseres Resultat erzielen kann.

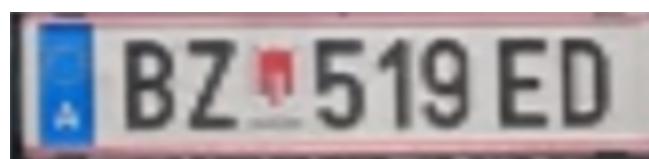


Abbildung 19: Resultat mit maschinellem Lernen

Zum Vergleich mit dem Testprogramm sieht man im oberen Bild das Ergebnis der Variante mit Maschinellem Lernen mit demselben Eingangsbild. Dabei wurde ohne Probleme das Kennzeichen perfekt erkannt und im weiteren Programmablauf aus dem Eingangsbild ausgeschnitten.

6.6.4 Verwendete Modelle für Maschinelles Lernen

6.6.4.1 WPOD-NET

Das „Warped Planar Object Detection Network“ (WPOD-NET) ist ein künstliches Neuronales Netz, welches von Sérgio Montazolli Silva und Cláudio Rosita Jung¹¹ entwickelt wurde. Es wurde entwickelt, um aus Bildern die Kennzeichen von Fahrzeugen herauszusuchen und bietet im Gegensatz zu anderen Modellen den entscheidenden Vorteil, dass es bei diesem Modell irrelevant ist aus welchem Winkel das Bild vom Kennzeichen aufgenommen wird. Andere Modelle funktionieren oft nur wenn das Foto vom Kennzeichen direkt von vorne aufgenommen wurde, da ansonsten das Bild verzerrt wird und dadurch das Kennzeichen nicht mehr ausgelesen werden kann. Dieses Modell schafft es auch aus einem von der Seite aufgenommen Bild das Kennzeichen in den meisten Fällen gut genug herauszulesen, dass dieses Bild danach weiterverarbeitet werden kann. Dies bietet den Vorteil, dass die Kamera für das Gerät nicht fix verbaut werden muss, sondern theoretisch auch in einem mobilen Gerät installiert werden kann.

Die Kennzeichenerkennung dieses Modells basiert auf zwei Schritten. Zuerst wird im Bild nach einem Fahrzeug gesucht und dieses ausgeschnitten, um danach mit WPOD-NET das Kennzeichen aus diesem Bild herauszusuchen. Die Fahrzeugarkennung basiert dabei auf dem YOLOv2¹² Netzwerk, welches für Echtzeitobjekterkennung verwendet wird und neben vielen anderen Gegenständen auch in der Lage ist Fahrzeuge zu erkennen. Nach dieser Fahrzeugarkennung wird das Bild in das WPOD-NET eingegeben.

¹¹Montazzolli, Sérgio & Jung, Claudio. (2018). License Plate Detection and Recognition in Unconstrained Scenarios.

¹²J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

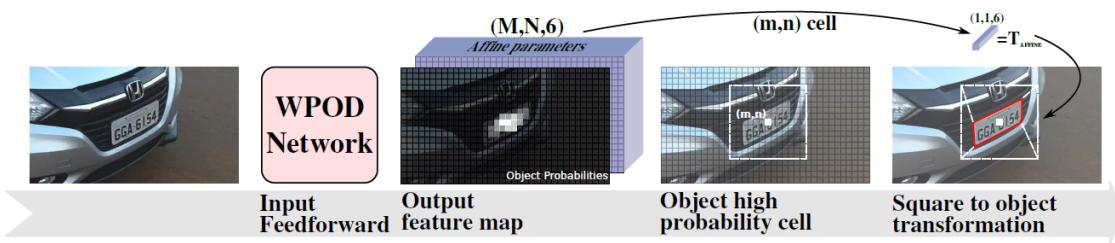


Abbildung 20: Ablauf von WPOD-NET

Im oberen Bild ist der prinzipielle Ablauf der Kennzeichenerkennung ersichtlich. Zu Beginn wird das Bild mit dem Kennzeichen in das WPOD-NET eingegeben und dieses gibt dann eine Matrix mit gewichteten Werten zurück, welche für jede einzelne Zelle eine Wahrscheinlichkeit angibt, dass sich dort ein Kennzeichen befindet. Um die Zelle, die am wahrscheinlichsten ist, wird dann ein Rechteck gespannt in welchem sich mit sehr hoher Wahrscheinlichkeit das Kennzeichen befindet. Danach wird auf die Wahrscheinlichkeitswerte für das gesamte Rechteck Thresholding angewandt, um zu bestimmen in welchen Zellen sich das Kennzeichen befindet. Dadurch hat man dann diejenige Region gefunden, in der sich das Kennzeichen befindet und kann dieses, wenn nötig in ein horizontal und vertikal ausgerichtet Rechteck umwandeln. Dadurch erhält man dann ein Bild des Kennzeichens.

6.6.4.2 MobileNetV2

MobileNetV2 ist die Grundlage für das eigene trainierte Neuronale Netz. MobileNetV2 ist eine Grundlage für diverse Neuronale Netze und speziell für Anwendungen im Bereich der Computer Vision geeignet. Einige Beispiel dafür sind die Erkennung von Gesichtsmerkmalen, Echtzeitobjekterkennung, Unterscheidung diverser Hunderassen oder ähnliches. Dabei ist das Modell je nach Bedürfnissen einstellbar, womit für eine schnellere und bessere Leistung weniger Schichten für das Neuronale Netz verwendet werden, oder aber für eine sehr genaue und präzise Funktion mehr Schichten verwendet werden. Dies ist dadurch möglich, dass das Netzwerk auf einer variablen Schichtanzahl aufgebaut ist. In dieser Applikation wird es deswegen verwendet, da damit in diesem Bereich eine hohe Präzision erzielbar ist, welche für die Kennzeichenerkennung benötigt wird.

6.7 Modell Training

Um das Modell für die Zeichenerkennung zu trainieren wurde ein eigenes Programm geschrieben, welches das Modell mittels Tensorflow trainiert. Dazu wird ein Datensatz von 35 000 Bildern von Zeichen verwendet, bei welchen die richtige Ausgabe aus dem Dateiname auslesbar ist. Im folgenden Bild sieht man ein Beispiel für so ein Bild aus dem Datensatz.

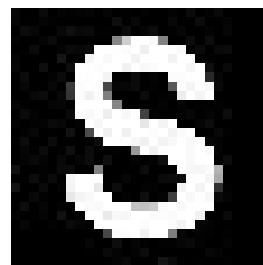


Abbildung 21: Beispiel eines Zeichens aus dem Datensatz

Das Modell wird durch die verschiedenen Schriftarten und die variierende Qualität der Bilder nicht direkt auf die Erkennung von Zeichen aus einem Kennzeichen trainiert, sondern auf die Erkennung von Zeichen in jeglicher Situation. Dies benötigt zwar länger, um das Modell zu trainieren, aber es ist dadurch genauer und kann auch in anderen Anwendungen ohne größere Umstellungen verwendet werden.

In den nächsten Codeteilen sieht man ein paar wichtige Ausschnitte aus dem Code für das Modell Training.

```

1 def create_model(lr=1e-4, decay=1e-4/30,
2     ↳ training=False, output_shape=y.shape[1]):
3
4     modelBuilder = MobileNetV2(weights="imagenet", include_top=False,
5         ↳ input_tensor=Input(shape=(80, 80, 3)))
6
7     headModel = modelBuilder.output
8
9     headModel = AveragePooling2D(pool_size=(3, 3))(headModel)
10
11    headModel = Flatten(name="flatten")(headModel)
12
13    headModel = Dense(128, activation="relu")(headModel)
14
15    headModel = Dropout(0.5)(headModel)

```

```

9   headModel = Dense(output_shape, activation="softmax")(headModel)

10

11 model = Model(inputs=baseModel.input, outputs=headModel)

12

13 if training:

14     for layer in baseModel.layers:

15         layer.trainable = True

16         optimizer = Adam(lr=lr, decay = decay)

17         model.compile(loss="categorical_crossentropy",
18                         → optimizer=optimizer, metrics=["accuracy"])

18

19 return model

```

Code 35: Erstellen des Modells basierend auf MobileNetV2

Im ersten Teil sieht man das grundlegende Erstellen des Modells auf der Grundlage von MobileNetV2. Zusätzlich werden noch die Einstellungen für Tensorflow getroffen wie zum Beispiel die Größe der Eingabe und weitere Parameter, welche von Tensorflow für die Erstellung des Modells benötigt wird. Im unteren Teil wird noch definiert welcher Optimizer verwendet werden soll und dass während des Trainings die Genauigkeit des Modells aufgenommen werden soll.

```

1 INIT_LR = 1e-4

2 EPOCHS = 30

3

4 model = create_model(lr=INIT_LR, decay=INIT_LR/EPOCHS, training=True)

5

6 BATCH_SIZE = 64

7

8 my_checkpointer = [EarlyStopping(monitor='val_loss', patience=5,
→ verbose=0),
→ ModelCheckpoint(filepath="License_character_recognition.h5",
→ verbose=1, save_weights_only=True)]

```

```
9   result = model.fit(image_gen.flow(trainX, trainY, batch_size=BATCH_SIZE),  
10      steps_per_epoch=len(trainX) // BATCH_SIZE,  
11      validation_data=(testX, testY),  
12      validation_steps=len(testX) // BATCH_SIZE,  
13      epochs=EP0CHS, callbacks=my_checkpointer)
```

Code 36: Trainieren des Modells und Einstellungen anpassen

Im zweiten Teil wird zuerst die Standard Lernrate des Modells und die Anzahl der Epochen definiert. Die Epochen sind eine Art Meilenstein, nach welchem die Genauigkeit dargestellt und beurteilt wird, um den Trainingserfolg zu beurteilen. Außerdem werden nach jeder Epoche die gewichteten Werte angepasst und abgespeichert. Danach wird noch eingestellt nach welcher Anzahl an Epochen ohne signifikante Verbesserung das Training frühzeitig beendet werden soll, und wohin die gewichteten Werte abgespeichert werden sollen. Danach werden Tensorflow die Daten übergeben, um das Training zu beginnen. Die Dauer dieses Vorgangs unterscheidet sich von Computer zu Computer, und hat bei dieser Applikation in etwa 10 Stunden gedauert.

Im nächsten Bild sieht man den Trainingsverlauf des Modells über 30 Epochen.

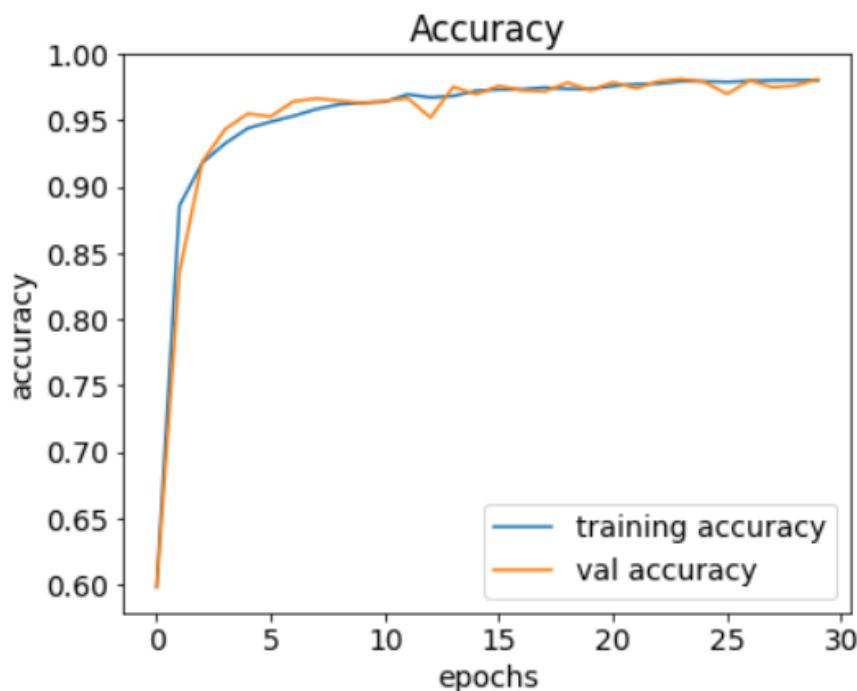


Abbildung 22: Verlauf des Modell Trainings

Im oberen Bild kann man erkennen, dass die Genauigkeit des Modells bereits nach wenigen Epochen einen Wert von über 90% erreicht hat und nach den 30 Epochen auf einen Endwert von 98,02% erreicht hat. Das bedeutet, dass bei einem Kennzeichen mit durchschnittlich 7 Zeichen eine insgesamte Genauigkeit von 86,93% erreicht wird. Diese Genauigkeit könnte mit einem längeren Training theoretisch auf einen Wert von nahezu 100% gesteigert werden, aber dazu benötigt es einen stärkeren Computer, welcher zudem noch eine ganze Weile nicht benutzt werden kann.

7 Fahrzeugerkennung

7.1 Anforderungen

Das Ziel der Fahrzeugerkennung ist es Fahrzeuge auf mehrere Parklücken eines Parkplatzes zu erkennen. Die daraus gewonnenen Zustände sollen an das Webinterface übermittelt und an den jeweilige Parklücken über LEDs ausgegeben werden.

7.2 Vorstudie

7.3 Erkennung von Metallen über Spulen

Grundsätzlich werden in der Realität häufig Spulen verwendet, welche unter dem Asphalt verbaut sind, um darüberliegende Fahrzeuge zu detektieren. Als Messprinzip wird die Änderung des magnetischen Widerstands R_m bei konstanter magnetischer Spannung U_m und daraus resultierende magnetischen Fluss ϕ . Es gelten für diese Größen der folgende Zusammenhänge:

$$\Phi = \frac{U_m}{R_m} \quad (1)$$

Wobei

Φ = magnetischer Fluss

R_m = magnetischer Widerstand

U_m = magnetische Spannung

Der magnetische Widerstand lässt sich wiederum durch die Eigenschaften der Spule bestimmen.

Die Formel hierfür lautet:

$$R_m = \frac{N \cdot (2a + 2b)}{\mu_0 \cdot \mu_r \cdot A} \quad (2)$$

Wobei

$$A = a \cdot b \quad (3)$$

N = Anzahl der Windungen der Spule

a = Breite der Spule in m

b = Länge der Spule in m

$\mu_0 = 1,2566 \cdot 10^{-6} \text{ N/A}^2$ = magnetische Feldkonstante

μ_r = relative Permeabilität

A = Fläche der Spule

Bis auf die relative Permeabilität sind alle anderen Variablen konstant. Daraus lässt sich schlussfolgern, dass der magnetische Fluss Φ anhand der Gleichung 1 und 2 proportional zur relativen Permeabilität ist.

$$\Phi \propto \mu_r \quad (4)$$

Der Fluss Φ hängt somit auch von den Materialien ab durch die er fließt. In der nächsten Abbildung kann man erkennen, wie ein Fahrzeug über eine im Boden installierte Spule den magnetischen Fluss Φ und somit die magnetische Flussdichte B beeinflussen kann.

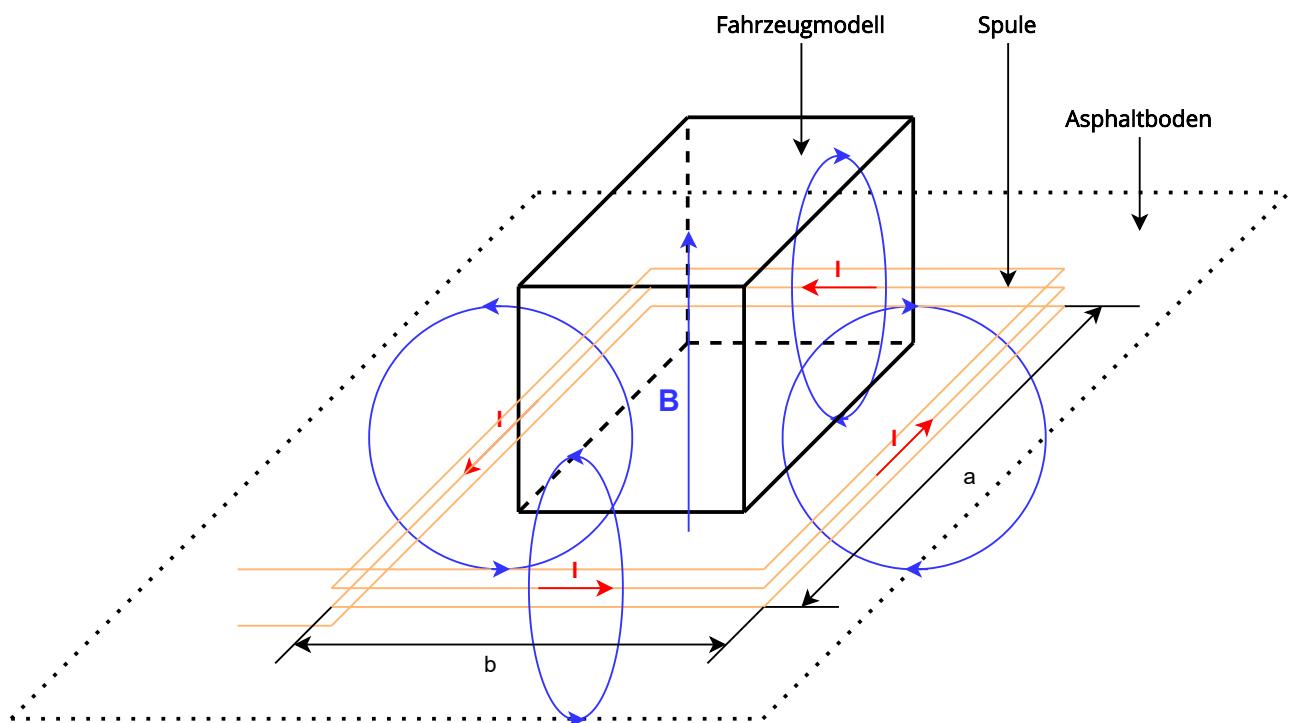


Abbildung 23: Installation der Spule

Um diese Größen auslesbar zu machen, müssen diese magnetischen bei einer direkten Messung in elektrische Größen umgewandelt werden. Hierfür gelten folgende Zusammenhänge:

$$L = \frac{N \cdot \Phi}{I} = \frac{\Psi}{I} \quad (5)$$

Wobei

L = Induktivität der Spule

I = Strom der durch die Spule fließt

N = Anzahl der Windungen der Spule

Ψ = Verkettete Fluss

$$u(t) = L \cdot \frac{di(t)}{dt} \quad (6)$$

Wobei

L = Induktivität der Spule

$i(t)$ = Strom der durch die Spule fließt zum Zeitpunkt t

$u(t)$ = Spannung die an der Spule anliegt zum Zeitpunkt t

t = Zeit in s

So lässt sich bei Bekanntheit von Strom und Spannung auf die Induktivität und mit der Gleichung 5 und mit den Zusammenhang 4 auf die magnetischen Eigenschaften des Materials rückschließen. Diese Art der Detektion bietet viele praktische Vorteile.

- **Größerer Messbereich**

Im Vergleich zu anderen Detektionsmethoden wie einer Leichtschranke kann man einen größeren Bereich durch die Wirkfläche der Spule abdecken. So lassen sich auch kleinere Kraftfahrzeuge wie Motorräder oder Mopeds besser erkennen.

- **Schutz vor Umweltfaktoren**

Durch den Verbau im Boden ist die Messeinrichtung vor Umwelteinflüssen wie Regen, Frost, hohen beziehungsweise niedrigen Temperaturen und Korrosion besser geschützt. Dies verringert auch den Einfluss dieser Störfaktoren auf die Eigenschaften Spule und somit auf die daraus resultierenden Messergebnisse.

- **Ausschließung von Materialien**

Alle nicht metallische Stoffe werden von diesem Messprinzip nicht wahrgenommen. So können Verschmutzungen wie Blätter und Staub, welche visuelle Sensoren stören können, die Detektion nicht behindern.

7.3.1 Messung ferromagnetischer Metalle

Um diese Detektionsverfahren zu verstehen muss zuerst der Zusammenhang zwischen Induktivität und relativer Permeabilität verstanden werden. Aus den Gleichungen 2, 1 und 5 kann die folgende Proportionalität ermittelt werden:

$$L \propto \mu_r \quad (7)$$

Bei ferromagnetischen Stoffen wie Eisen ist die relative Permeabilität sehr viel größer als 1. Wenn nun ein Auto oder eine anderes Vehikel sehr viel Eisen beinhaltet wirkt sich dies steigernd auf die Induktivität der Spule aus. Die folgenden Verfahren nutzen dieses Prinzip um die Belegung einer Parklücke zu bestimmen.

7.3.1.1 RL-Oszillator mit Timer Baustein

Bei diesem Messverfahren wird die Änderung der Induktivität L über die Änderung einer Stromladekurve über einen Widerstand R ermittelt. Eine Simulation in LTSPice mit folgendem Schaltbild kann die Auswirkung auf eine Ladekurve bei gleicher Spannung und gleichem Widerstand gut darstellen.

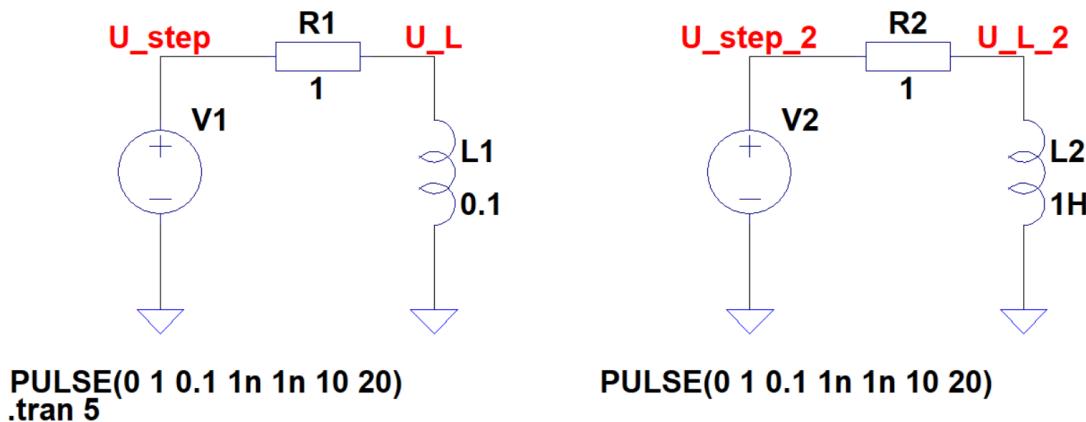


Abbildung 24: LTSPice Blockbild zweier RL-Glieder

Die daraus resultierende Simulation im Zeitbereich ergibt folgendes Ergebnis: hmm

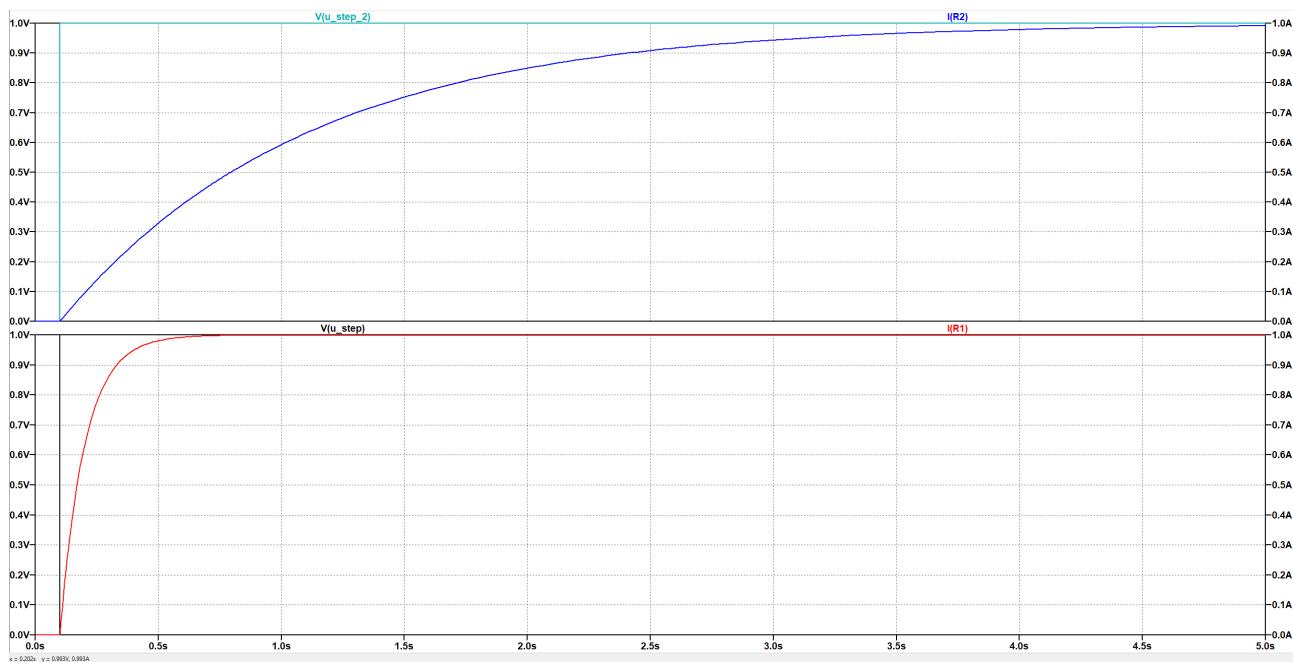


Abbildung 25: Stromkurven zweier RL-Glieder

Es ist aus der letzten Abbildung erkennbar, dass sich eine größere Induktivität verlangsamt auf die Zunahme des Stromes auswirkt. Für Einschaltvorgänge kann diese Stromkurve für RL-Glieder folgendermaßen beschrieben werden.

$$i(t) = \frac{U_0}{R} \cdot (1 - e^{-\frac{t}{\tau}}) \quad (8)$$

$$\tau = \frac{L}{R} \quad (9)$$

Wobei

$i(t)$ = Strom der durch die Spule fließt zum Zeitpunkt t

L = Induktivität der Spule in H

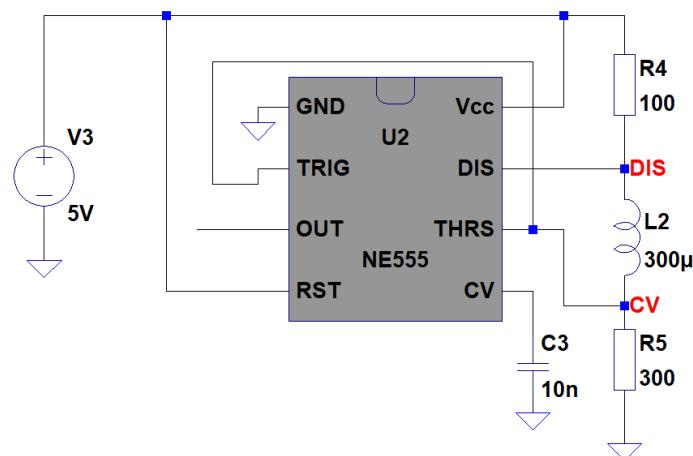
R = Widerstand in Ω

U_0 = Spannung die nach dem einschalten anliegt in V

t = Zeit in s

τ = Zeitkonstante in $\frac{1}{s}$

Timer Bausteine wie der NE555 nutzen diese solche Verlaufskurven wenn sie als astabile Kippstufe konfiguriert sind. Es gibt einen Eingang dieses Baustein meist CV für Control Voltage der die Spannung überwacht. Überschreitet diese Spannung zweit drittel der Betriebsspannung schaltet er einen weiteren Pin, meist DIS für Discharge, auf 0V beziehungsweise auf Masse. Unterschreitet jedoch die Spannung am CV Pin ein drittel der Betriebsspannung so wird der DIS Pin auf Betriebsspannung geschalten. Diese beiden Pins sind über RC- oder RL- Glieder verbunden. Es entsteht somit ein Oszillator, welcher die Spannung am DIS Pin anhebt und absenkt. Der NE555 kann dieses Signal über einen internen Komparator in ein Rechtecksignal umwandeln, welches besser von Mikrokontrollern ausgelesen werden kann. Der Mikrokontroller kann so die Anzahl der einkommenden Takte über einen gewissen Zeitraum zählen und daraus eine Oszillatofrequenz errechnen. Die Zeitsignale lassen sich in LTSPice mit dem NE555 als Timer-Baustein simulieren.



.tran .01m

Abbildung 26: RL-Oszillator mit NE555

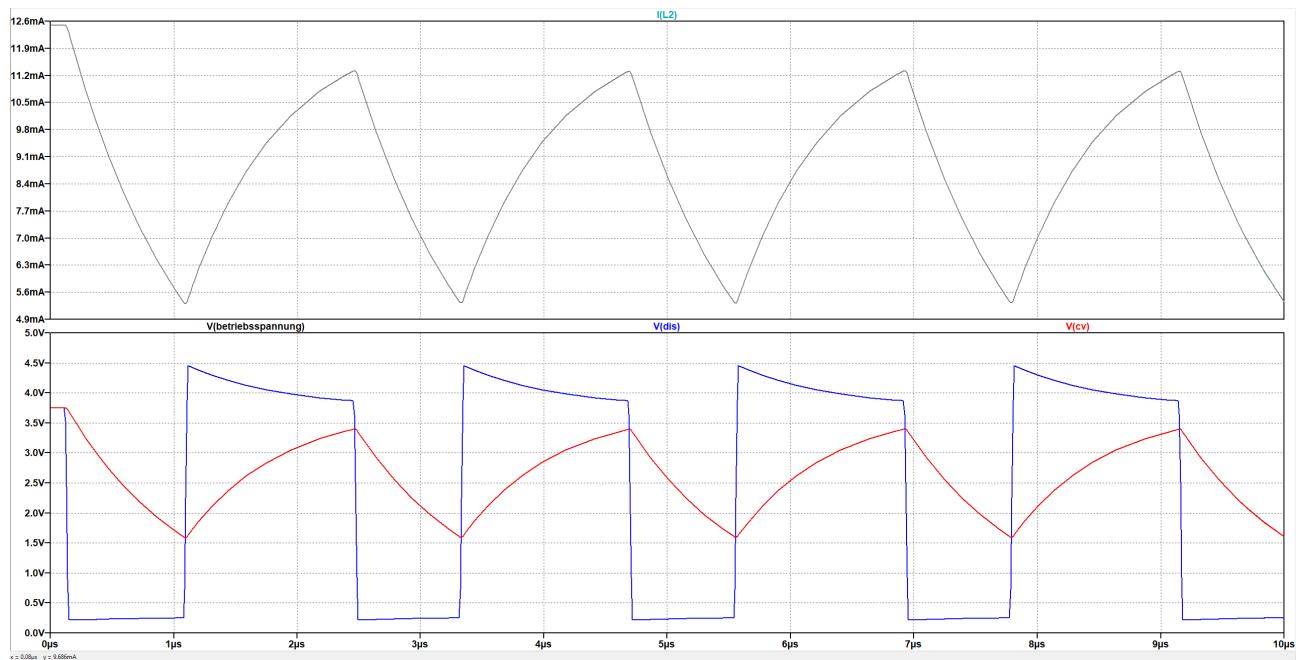


Abbildung 27: RL-Oszillator mit NE555 Zeitsignale

Im oberen Diagramm ist der Verlauf des Stromes durch die Spule zu erkennen der wie erwartet zu- und abnimmt. Im unteren Diagramm ist in schwarz die Betriebsspannung als Referenz in rot der CV Pin und in blau der DIS Pin des NE555. Das rote Signal wechselt zwischen zwei dritteln der Betriebsspannung und einem drittel der Betriebsspannung mit einer Frequenz von $f = 438 \text{ kHz}$ bei einer Induktivität von $L = 300 \mu\text{H}$, einem Widerstand $R4 = 100\Omega$ und einem Widerstand $R5 = 300\Omega$. Die Simulation entspricht hier der Theorie, jedoch bricht das blaue Signal am PIN DIS in der Einschaltphase ein. Der Grund dafür ist der interne Widerstand des NE555, der ab einem gewissen Stromverbrauch für einen Spannungsabfall sorgt. Da es meist besser ist mit niedrigen Frequenzen zu arbeiten ist es nach den Gleichungen 8 und 9 entweder nötig die Induktivität zu erhöhen oder die Widerstände zu verkleinern. Die Induktivität lässt sich aber entweder durch erhöhen der Windungen oder durch Vergrößerung der Spule erreichen, was in vielen Fällen nicht möglich ist oder zunehmend kostspielig ist. Die Reduktion der Widerstände führt zu einer Zunahme des Stromes und der Verlustleistung, was auch unerwünscht ist. Aus diesen Gründen ist der RL-Oszillator nur für hohe Frequenzen geeignet. In der nächsten Abbildung sieht man die Schaltung bei verschiedenen Induktivitäten nämlich bei $L2 = 100 \mu\text{H}$ und bei $L2 = 1 \text{ mH}$.

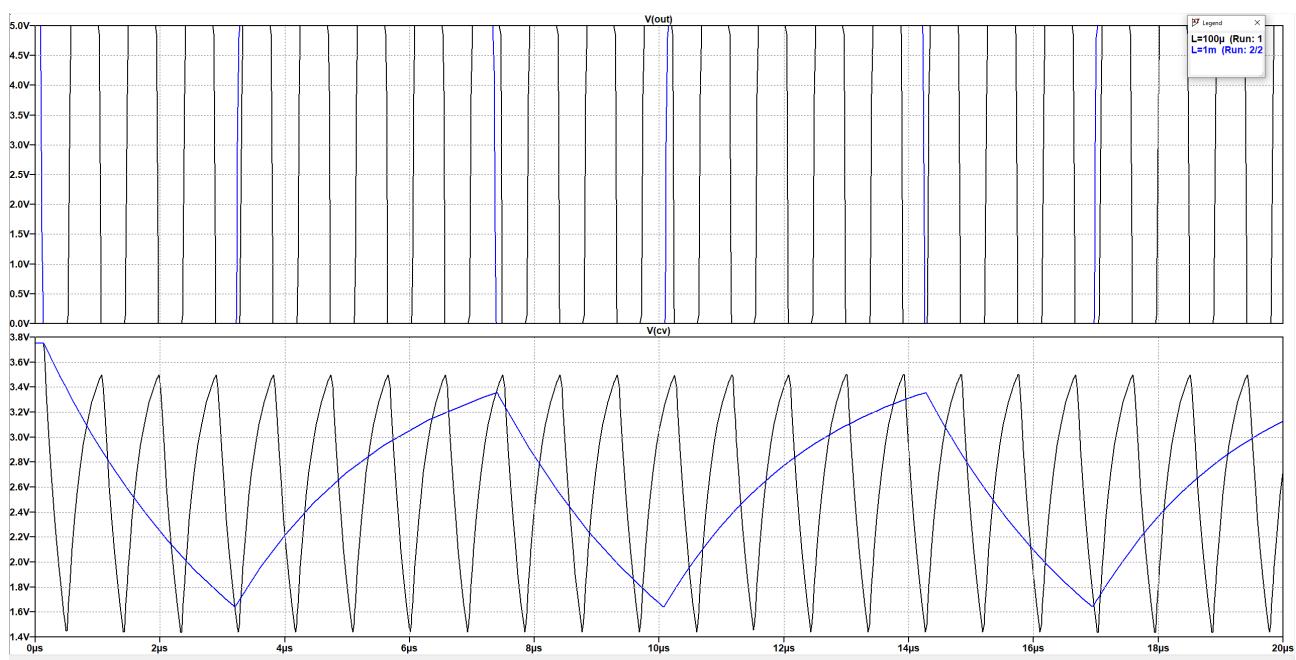


Abbildung 28: Vergleich des NE555-Oszillators bei unterschiedlichen L

Eine Verzehnfachung der Induktivität führt zu einer Reduktion der Oszillatorkreisfrequenz um den Faktor 10.

$$f \propto \frac{1}{L} \quad (10)$$

7.3.2 Messung paramagnetischer Metalle

Da nicht alle Metalle ferromagnetische Eigenschaften haben werden viele Stoffe wie Aluminium, Kupfer oder diverse Legierungen von einer niederfrequenten Messung der Induktivität nicht wahrgenommen. Ein Effekt, welcher bei elektrischen Leitern abhilfe schaffen kann, sind die Eddy Currents. Wenn in einem Leiter ein magnetisches Feld einwirkt erzeugt dieses im Objekt Ströme, die wiederum ihr eigenes magnetische Feld erzeugen. Die fließenden Ströme erzeugen im Objekt Ohmsche Verluste, welche normalerweise unerwünscht sind. Sie wirken auch einer Änderung der magnetischen Feldes entgegen, da sie selbst Energie brauchen um ihre Richtung zu ändern. Je größer die Frequenz der eingespeisten Felddichte B desto stärker wirkt die Flussdichte B_{eddy} senkend auf die Gesamtflussdichte und so auf den magnetischen Fluss Φ . Nach der Gleichung 5 reduziert sich die Induktivität mit zunehmender Frequenz.

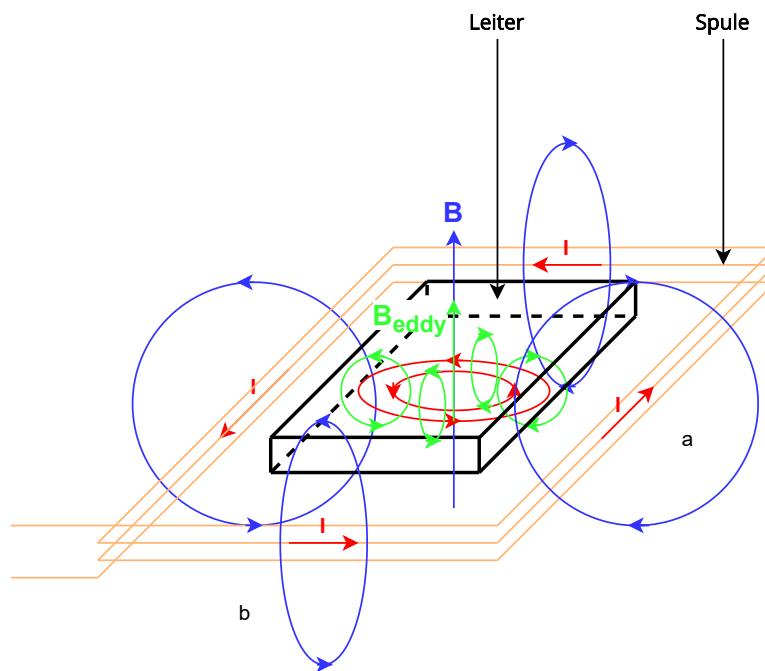


Abbildung 29: Eddy Currents in einem Leiter

Bei sehr hohen Frequenzen ist jedoch zu beachten, dass der Skin-Effekt die Querschnittsfläche, durch die der Wechselstrom fließen kann, reduziert. Somit nimmt der Effekt der Eddy-Currents ab. Diese Phänomene werden jedoch nicht in dieser Arbeit behandelt, da sie außerhalb des Rahmens der Fahrzeugerkennung liegen.

7.3.2.1 LC Oszillatoren

Im Vergleich zum RL-Oszillator, der über eine astabile Kippstufe lauft, kann ein LC Oszillator seine Resonanzfrequenz nur durch das verstellen der Induktivität und der Kapazität eines Kondensators verändern. Zudem gibt es in der Theorie keine Wirkverluste, da die Energie abwechselnd zwischen magnetischer und elektroscher Energie umgewandelt wird. In der Realität gibt es jedoch ohmsche Verluste, die zu einem Abklingen der Schwingung führen. Es braucht daher ein aktives Glied wie ein Transistor oder ein Operationsverstärker, der den Oszillator mit Energie versorgt.

Der zeitliche Verlauf einer ungedämpften Schwingung kann in LTSPice simuliert werden indem wir eine Sprungfunktion auf ein LC Glied geben und einen Widerstand in Serie zum Kondensator oder zur Spule geben.

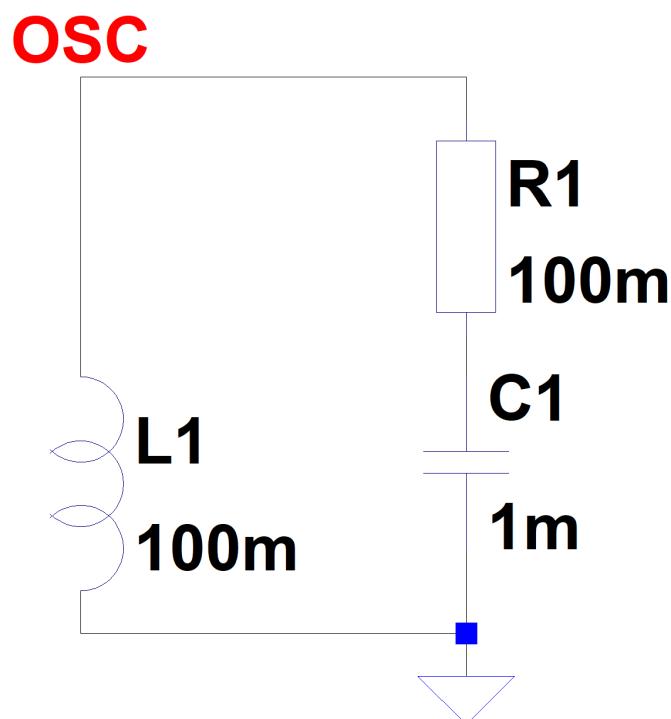


Abbildung 30: Parallelschwingkreis in LTSpice

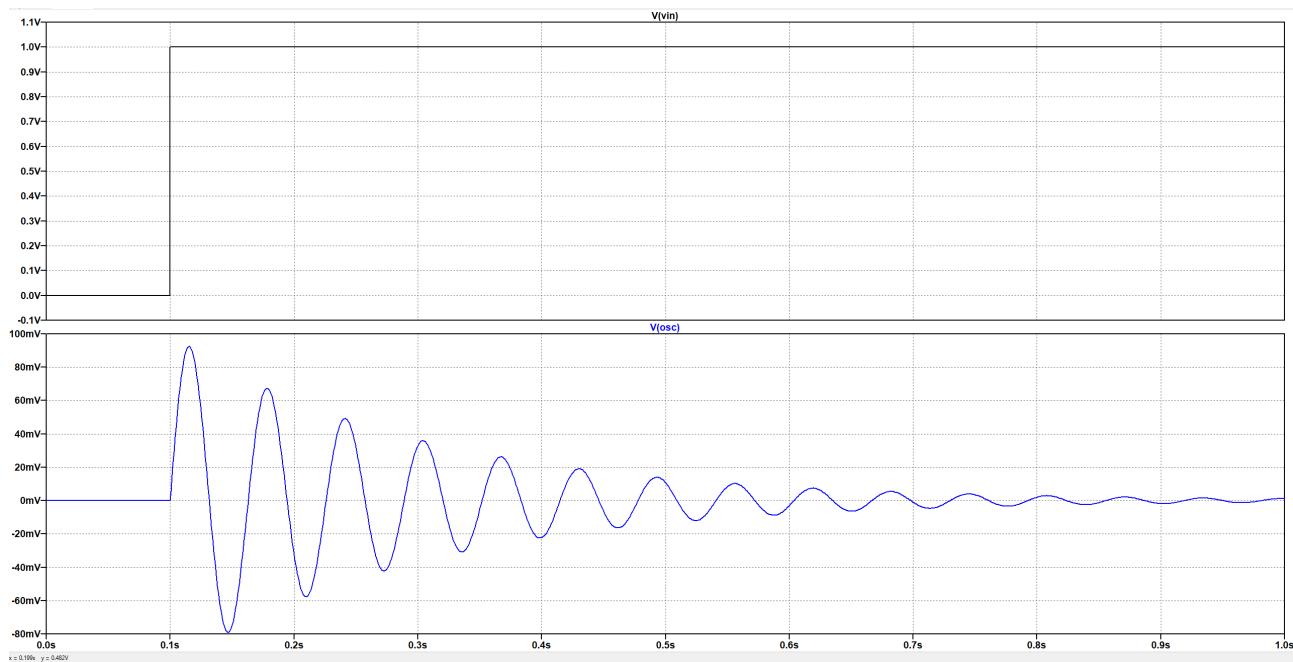


Abbildung 31: Gedämpfte Schwingung

Aus der Simulation ergibt sich über eine Cursormessung für die angegebenen Werte $L = 100 \text{ mH}$ und $C = 100 \text{ mF}$ eine Resonanzfrequenz einen Wert von $f = 15.85 \text{ Hz}$. Die Resonanzfrequenz für einfache LC-Schwingkreise lässt sich mit der Thomsonschen Schwingungsgleichung errechnen:

$$f = \frac{1}{2 \cdot \pi \sqrt{L \cdot C}} \quad (11)$$

Wobei

f = Resonanzfrequenz des LC-Gliedes

L = Induktivität der Spule

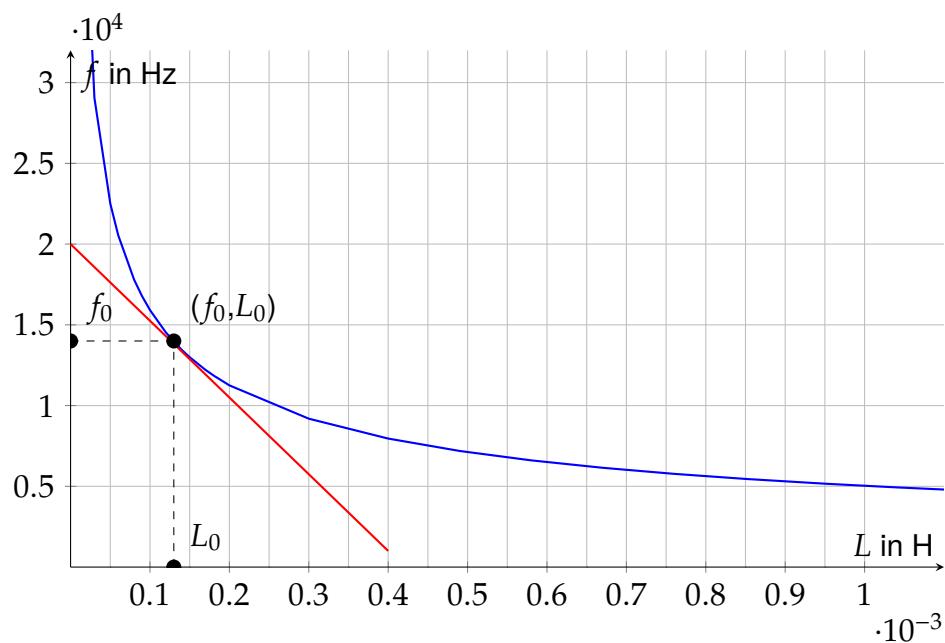
C = Kapazität des Kondensators

Wenn man die Werte $L = 100 \text{ mH}$ und $C = 100 \text{ mF}$ in die Gleichung 11 einsetzt erhält man eine Resonanzfrequenz von $f = 15.915 \text{ Hz}$. Allgemein tritt in LC-Glieder bei einer Phasenlage von $\varphi_{LC} = 180^\circ$ und bei der Frequenz nach der Thomsonschen Gleichung 11 Resonanz auf.

7.3.2.2 Einfluss von Induktivitätsänderungen auf LC-Oszillatoren

Für die Fahrzeugerkennung ist es von Relevanz zu wissen wie die Frequenz dieser Oszillatoren

auf die Änderung der Induktivität der im Boden verbauten Spule reagiert. Wenn eine Parklücke frei ist besitzt die Spule eine Induktivität von L_0 und hat nach der Thomsonschen Gleichung 11 eine Resonanzfrequenz f_0 . Für kleine Änderungen ΔL kann eine Näherung der Frequenzänderung gemacht werden indem man in den Punkt (f_0, L_0) in den Graphen der Thomsonschen Gleichung aufgetragen über die Induktivität L eine Tangente in diesen Punkt legt. Bei einer Kapazität von $C = 1 \mu\text{F}$ erhält man folgenden Graphen:



Mit dieser Näherung kann man mit des Differentials der Thomsonschen Gleichung 11 die Änderung des Funktionswertes $f(L \pm \Delta L) = \Delta f \approx df$

$$\begin{aligned} df &= \frac{\partial f}{\partial L} \Big|_{L_0} \cdot dL \\ df &= \frac{\partial}{\partial L} \left(\frac{1}{2\pi\sqrt{L_0 C}} \right) \Delta dL \\ df &= -\frac{1}{4\pi} (L_0 C)^{-\frac{3}{2}} \cdot \Delta L \end{aligned}$$

Für relative Änderungen der Induktivität ΔL um den Wert L_0 gilt:

$$\frac{df}{f_0} = -\frac{1}{2C} \cdot \frac{\Delta L}{L_0} \quad (12)$$

Das bedeutet, dass eine kleine relative Änderung der Induktivität $\frac{\Delta L}{L_0}$ bei gleicher Kapazität C für eine halb so große relative Änderung in der Frequenz führt. Dieser Ansatz gilt nur bei kleinen ΔL .

7.3.2.3 Messung der Frequenz eines Colpitts-Oszillatoren

Der Colpitts-Oszillatoren ist eine Variante des LC-Oszillatoren und wird in der Praxis oft mit einem aktiven Element betrieben. Im Vergleich zu einem einfachen LC-Glied besitzt dieser Oszillatoren zwei Kondensatoren. Das Schaltbild sieht so aus:

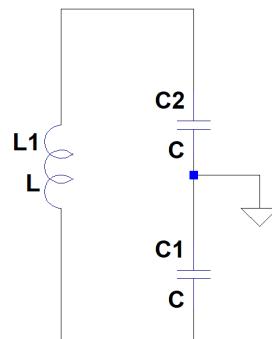


Abbildung 32: Colpitts LC-Glied

Dieses LC-Glied wird als Rückkoppelglied eingesetzt und sorgt für die nötige Phasendrehung von $\varphi_{LC} = 180^\circ$. Wie man in der obigen Abbildung sehen kann liegen die Kondensatoren C_1 und C_2 in Serie. Es ergibt sich daher eine Gesamtkapazität von C_{ges} .

$$C_{ges} = \frac{C_1 \cdot C_2}{C_1 + C_2} \quad (13)$$

Durch einsetzen in die Thomsonsche Gleichung erhält man für die Resonanzfrequenz eines Colpitts-Oszillatoren:

$$f = \frac{1}{2 \cdot \pi \sqrt{L \cdot \left(\frac{C_1 \cdot C_2}{C_1 + C_2} \right)}} \quad (14)$$

Wobei

f = Resonanzfrequenz des Colpitts-Oszillatoren

L = Induktivität der Spule

C_1 = Kapazität des Kondensators C_1

C_2 = Kapazität des Kondensators C_2

Das aktive Element wird an die Anschlüsse der Spule angeschlossen muss selbst eine Phasendrehung von 180° liefern. Zusätzlich muss die Verstärkung des Gliedes größer als 1 damit eine dauerhafte Schwingung entstehen kann. Mit einem Operationsverstärker muss der nicht invertierende Eingang auf eine Bezugsspannung gelegt werden. Diese Bezugsspannung wird oft in die Mitte des Versorgungsspannungsbereich gelegt. Bei einem Spannungsbereich von 0 V bis 5 V ist eine Bezugsspannung von 2,5 V sinnvoll. Der invertierende Eingang wird an das Colpitts LC-Glied angeschlossen. Der Ausgang des Operationsverstärker wird über einen Widerstand an das LC-Glied geschalten, damit das Rückkoppelglied besser vor Übersteuerungen am Ausgang des Operationsverstärkers geschützt ist.

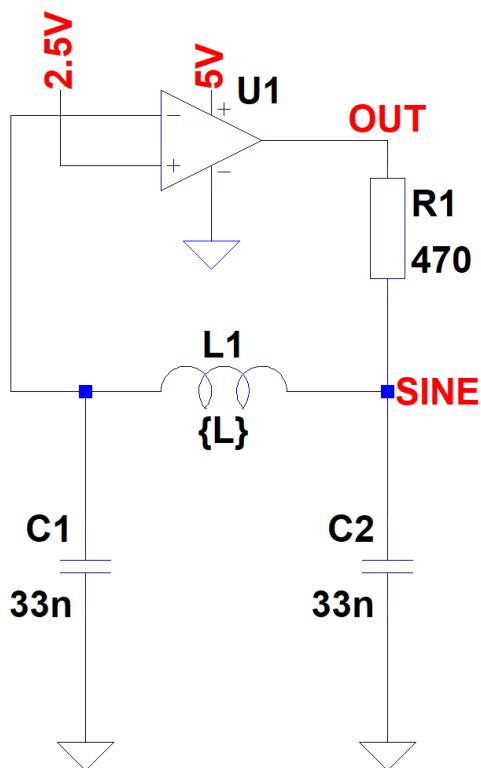


Abbildung 33: Aktiver Colpitts-Oszillator

Wenn man nun verschiedene Werte für Induktivität einsetzt ergibt sich durch die Simulation dieses Schaltbildes folgende Zeitsignale für den Ausgang des Operationsverstärker OUT und der Spannung SINE:

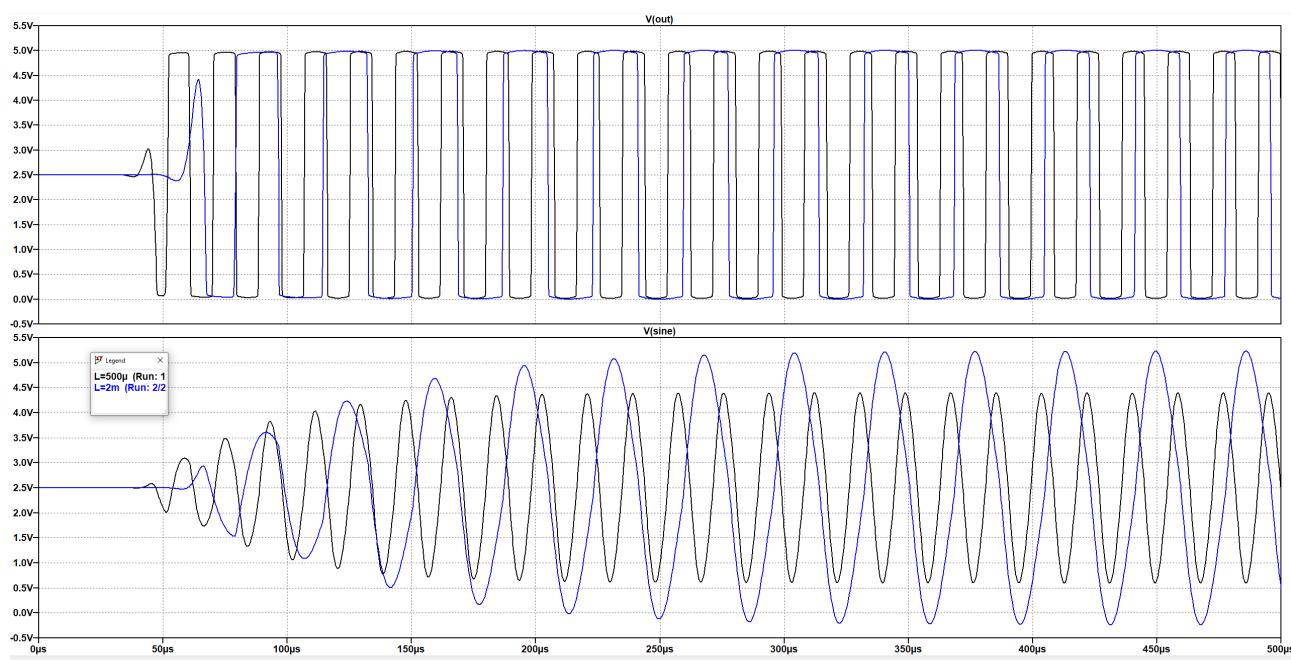


Abbildung 34: Zeitsignale Colpitts-Oszillator

Es ist ersichtlich, dass eine Zunahme der Induktivität zu einer Abnahme in der Frequenz der beiden Signale führt.

7.4 RS485 Bussystem

7.4.1 Benutzte Standards

7.4.1.1 ASCII

ASCII ist eine amerikanischer Standard zur Kodierung einer 7 Bit langen folge. Dieser Standard ermöglicht es Zeichen wie 'A' oder 'b' sowie auch Zahlen und gewissen Sonderzeichen in Bits umzusetzen.

7.4.1.2 USB

Unter USB versteht man eine serielle Bussystem, welches dazu dient Peripherie Geräte an einen Computer zu verbinden. USB ermöglicht es Geräte während der Laufzeit des Betriebssystems anzustecken und wieder zu entfernen. Je nach Version des USB Standards ist es möglich unterschiedliche Datenraten und Leistungen zu verwenden, wobei neuere Standards Rückkompatibel zu alten Versionen sind. Für die Ansteuerung des Bussystems ist es notwendig einen Computer anzuschließen. Der USB Standard bietet sich hier an, da er weit verbreitet ist und daher mit vielen Computern kompatibel ist.

7.4.1.3 UART

Unter UART versteht man eine asynchrone serielle Schnittstelle. Die elektrische Schnittstelle braucht daher auch keine Taktleitung die dem Empfänger der Daten mitteilt wann er zu lesen hat. Stattdessen gibt es zwei Datenleitungen nämlich eine für das Senden TX und eine für das Lesen RX. Auf diesen Datenleitungen befinden sich nur die Bitströme des Empfängers und des Senders. Sie besitzen einen Ruhepegel von logisch 1. UART synchronisiert sich idem es beim Start der Daten ein Startbit schickt, welche Sender und Empfänger synchronisiert. Diese beiden Geräte brauchen daher auch genaue interne Uhren um sich über den Zeitraum des Datenaustausches synchron zu halten. Die Dauer dieses Datenaustausches hängt vor allem von der Bitrate auch genannt Baudrate ab aber auch die Konfiguration der Anzahl von Datenbits und Paritätsbits haben Einfluss auf diese Größe.

In der folgenden Abbildung ist zeitliche Verlauf einer Datenleitung dargestellt es wird das Zeichen

'A' in ASCII codiert übertragen bei einer Baudrate von 57600 Zeichen pro Sekunde.

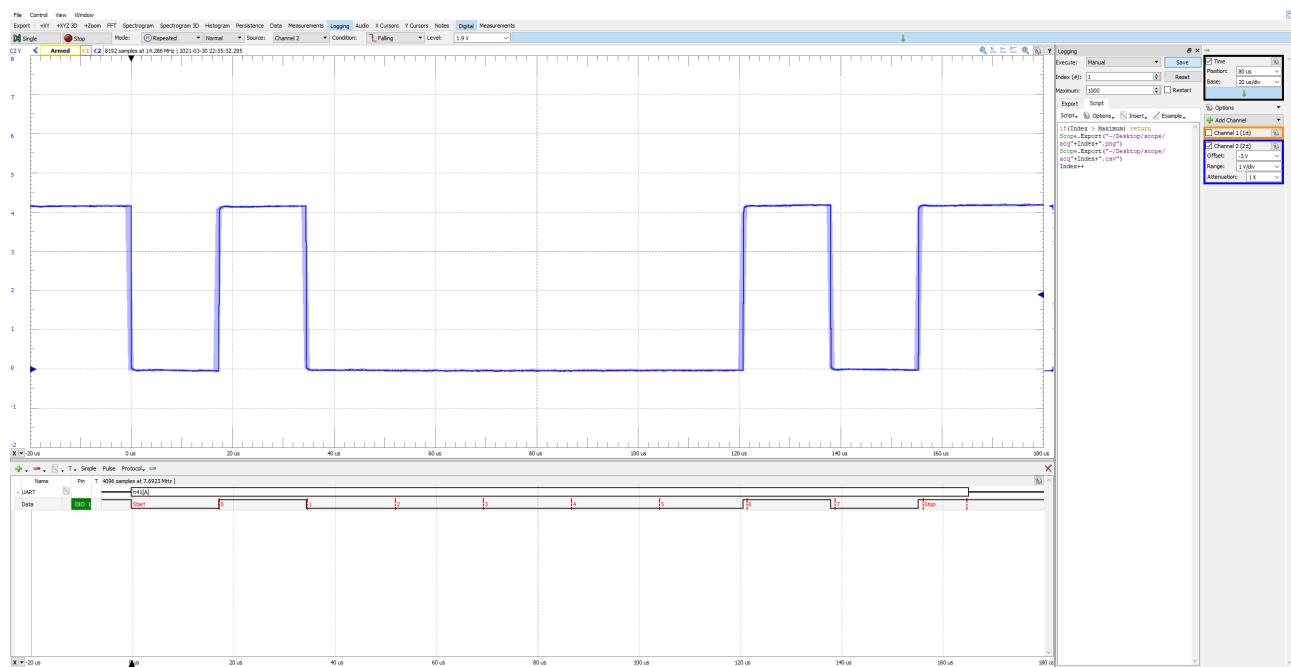


Abbildung 35: Beispiel einer UART Zeichenübertragung

7.4.1.4 RS485

RS485 ist ein Standard für eine physikalische Schnittstelle für Datenübertragung von UART Daten. Er besitzt zwei symmetrische Leitung oft mit A und B bezeichnet, dessen Differenzspannung für die Auswertung des Datenstromes herangezogen wird. Er ist dafür ausgelegt mehrere Geräte an die gleiche Leitungen anzuschließen. Die maximale Anzahl an Geräten ist mit 32 fix vorgegeben. Es ist zusätzlich notwendig die Spannungen U_A und U_B im Ruhezustand über einen IC oder einen Spannungsteiler auf fixe Potentiale zu legen. Der Spannungsteiler muss wie folgt aussehen um dem Standard zu entsprechen.

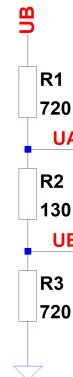


Abbildung 36: RS485 Widerstandsnetzwerk

Um diese physikalische Umwandlung zu ermöglichen ist eine Umwandlungs IC erforderlich. In dieser Arbeit wurde ausschließlich der MAX485CSA für Wandlung zwischen den Standardpegeln der UART und die der RS485 Schnittstelle verwendet.

Dieser IC ist Bussfähig und kann daher seine Ausgänge hochohmig machen. Er besitzt folgende Eingänge und Ausgänge:

- **!RE: Read enable**

Wenn dieser Eingang auf logisch 0 gesetzt ist wertet der IC des Differenzsignal U_{AB} aus und liefert das Ergebnis an den Ausgang RO. Ansonsten setzt er den Ausgang RO in den Tristate beziehungsweise auf hochohmig.

- **DE: Data enable**

Wenn dieser Eingang auf logisch 1 gesetzt ist wird die Spannung am Eingang DI in die zugehörigen Pegel U_A und U_B umgewandelt. Ansonsten setzt er den Eingang DI in den Tristate beziehungsweise auf hochohmig.

- **RO: Read out**

Ist die eingelesene Spannung die sich logisch aus der Differenzspannung U_{AB} ergibt wenn !RE auf logisch 0 ist.

- **DI: Data In**

Ist die eingehende Spannung die auf die RS485 Schnittstelle geschrieben werden soll, wenn der Eingang DE auf logisch 1 ist.

Die Auswertung der Differenzspannung U_{AB} kann auch in der folgenden Tabelle aus dem Datenblatt des MAX485CSA entnommen werden.

Table 2. Receiving

INPUTS			OUTPUT
RE	DE	A-B	RO
0	0	$\geq +0.2V$	1
0	0	$\leq -0.2V$	0
0	0	Inputs open	1
1	0	X	High-Z*

X = Don't care

High-Z = High impedance

**Shutdown mode for MAX481/MAX483/MAX487*

Abbildung 37: Logische Tabelle für U_{AB}

In der folgenden Abbildung wird wieder an 'A' ASCII kodiert versendet nur dieses mal wird eine RS485 Schnittstelle mit den entsprechenden Pegeln verwendet verwendet. In blau ist die Spannung U_A und in orange die Spannung U_B zu sehen. Darunter befindet sich das logische UART Signal am Ausgang RO.

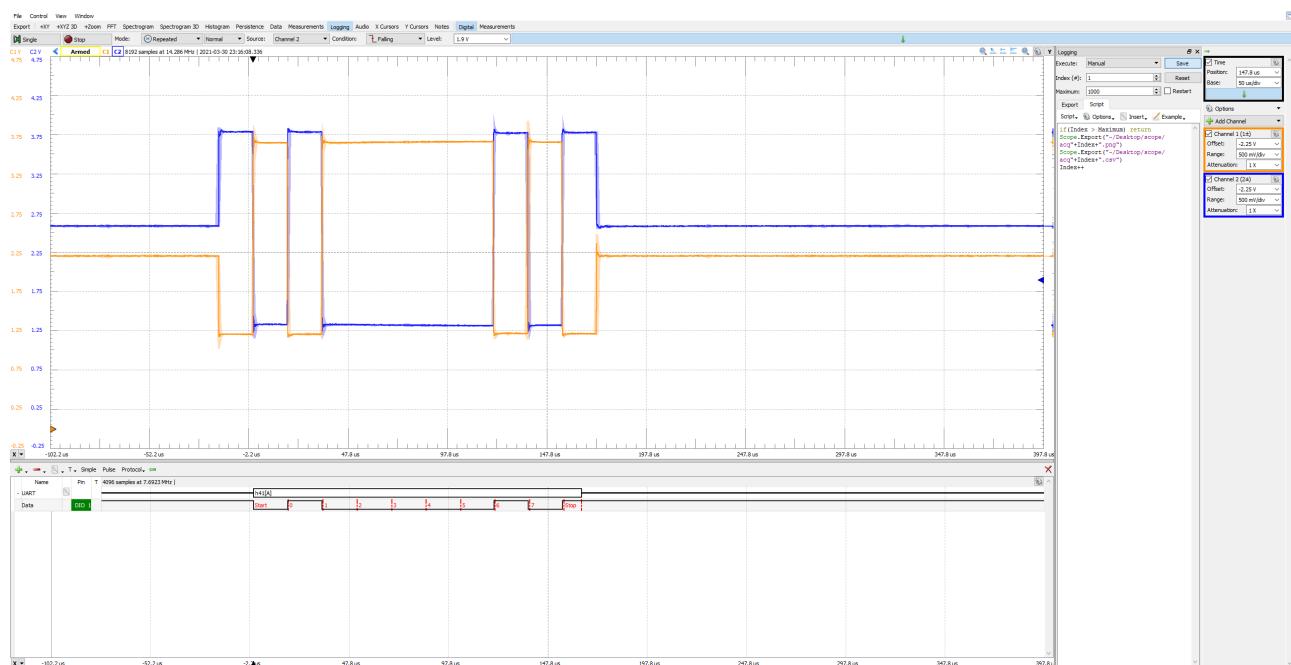


Abbildung 38: Beispiel einer RS485 Zeichenübertragung

7.4.1.5 RJ45

Um die Anzahl der Kabel zu minimieren und um das Anschließen von Geräten an den Bus einfach zu halten ist ein mehradriger Stecker mit zugehörigem Kabel nötig.

Der RJ-45 Stecker ist eine genormter Stecker und besitzt insgesamt 8 Adern und ist für Kommunikationsanschlüsse gedacht. Er kann daher größere Datenraten zulassen und viele auf dem Markt vorhandene Kabel haben zusätzliche Abschirmungen, die die Datenleitungen vor elektrischen Störungen schützen.



Abbildung 39: RJ45 Steckerbuchse und Kabel

7.4.2 Aufbau

Das Bussystem wir über eine Master Slave Struktur verwaltet was heißt das nur das ein einzelnes Master-Gerät als erstes andere Teilnehmer am Bus ansprechen kann. Slave-Geräte können nur

nach Anfrage des Masters auf den Bus schreiben. Neben der Steuerung der Kommunikation sollte das Master-Gerät aus praktischen Gründen ebenfalls die Slave-Geräte mit Energie versorgen und ihnen eindeutige Adressen zuweisen können.

In den unteren Abbildungen sind der logische Aufbau und der physikalische Aufbau zu sehen. Auf die einzelnen Funktionen diverser Leitungen wird in den folgenden Paragraphen eingegangen.

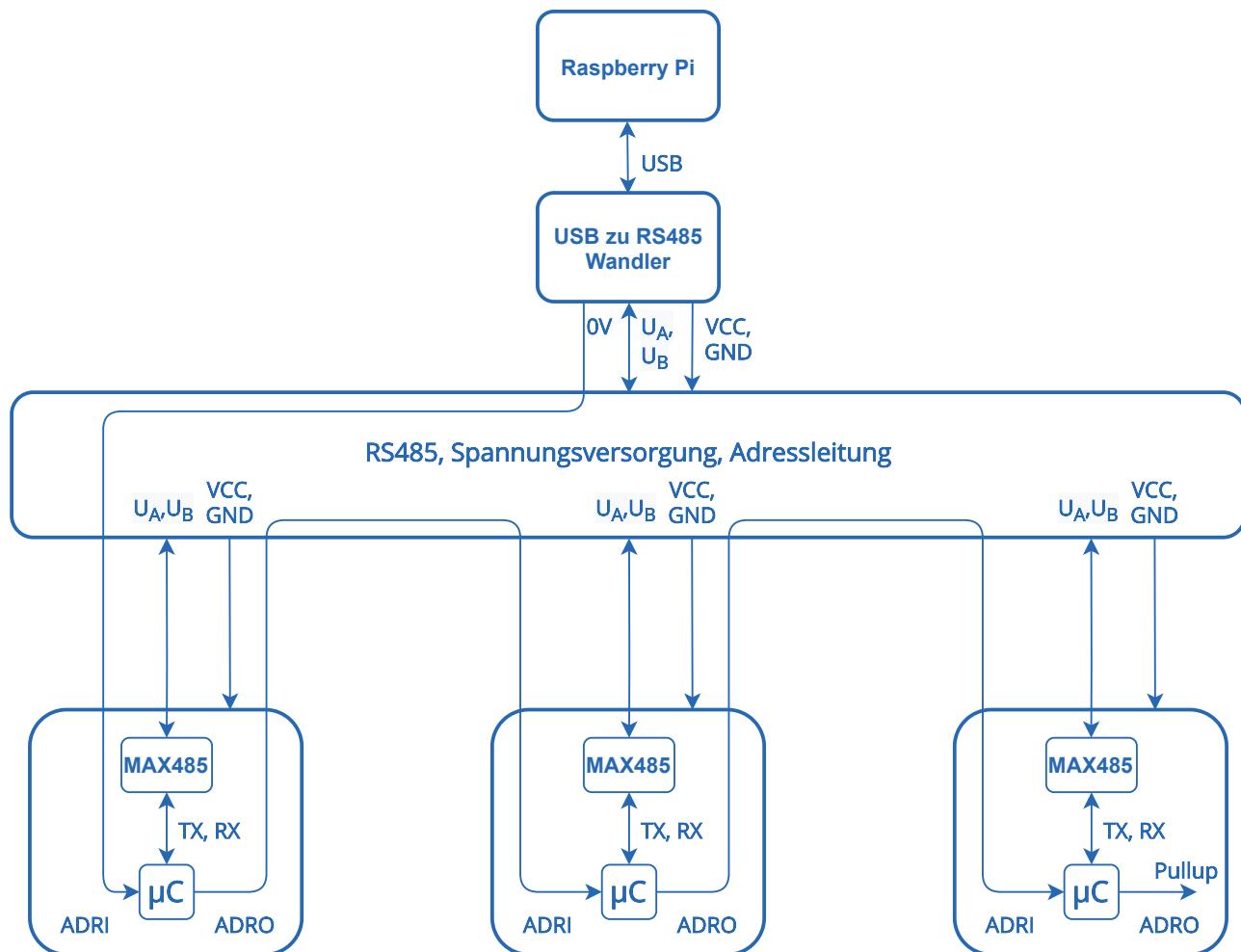


Abbildung 40: Logische Übersicht des Bussystems



Abbildung 41: Physische Übersicht des Bussystems

Die elektrische Belegung der Leitungen ist wie in der folgenden Tabelle festgelegt worden. Mit dieser Belegung ist es möglich die Slave-Geräte an den RS485 Datenbus zu legen, sie mit Spannung zu versorgen und die Adressvergabe mit einer zusätzlichen Logikleitung zu regeln.

Pinnummer	Farbe	Netz	Beschreibung
1	Orange / weiß	GND	Masse
2	Orange	+5V	5 Volt Versorgungsspannung
3	Grün / weiß		unbelegt
4	Blau	A	RS485 positives Datensignal
5	Blau / weiß	B	RS485 negatives Datensignal
6	Grün	VCC	Positive Versorgungsspannung +9V bis 30V
7	Braun / weiß	GND	Masse
8	Braun	ADR	Logiksignal zur Vergabe der Slave Adressen

Tabelle 2: RJ45 allgemeine Pinbelegung

7.4.2.1 Funktion der Adresslogikleitung

Da alle Teilnehmer des RS485-Busses parallel an den Datenleitung A und B liegen kann der Master sie voneinander nicht unterscheiden. Dieses Problem beseitigt die Adressleitung, welche die einzelnen Geräte logisch aneinanderreihrt.

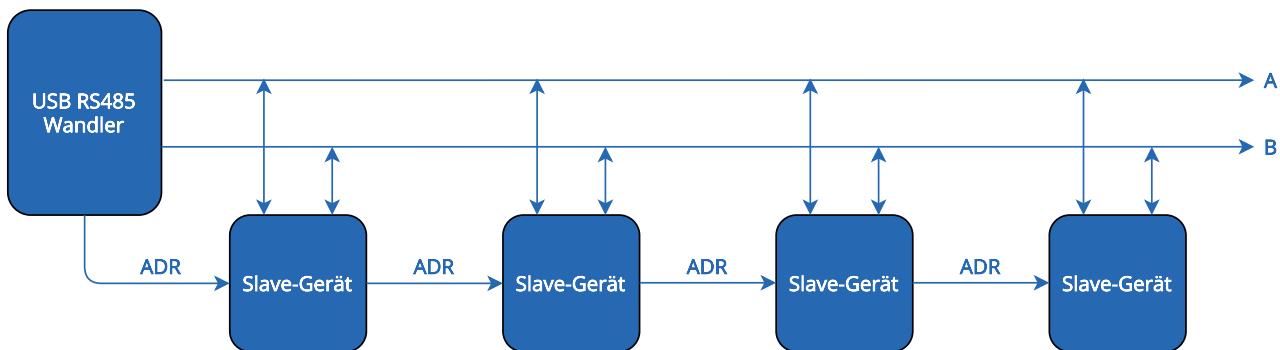


Abbildung 42: Anordnung der Slave-Geräte mit Adresslogikleitung

Jedes Slave-Gerät setzt seine Adresslogikleitung über einen Pullup-Widerstand auf logisch 1 wenn es keine Adresse zugeordnet bekommen hat. Als erstes in der Reihe ist der Master, der seine Adresslogikleitung auf logisch 0 setzt und so dem ersten Gerät in der Kette bekannt gibt, dass es eine Adresse zugewiesen bekommen hat. Wenn es diese Adresse bekommen hat legt das Slave-Gerät seine ausgehende Adresslogikleitung ADRO auf logisch 0. So kann das nächste Gerät in der Kette erkennen, dass es eine Adresse zugewiesen kriegt, indem es auf die eingehende Adressleitung ADRI achtet.

In den folgenden Abbildungen sieht man wie die Geräte über den RJ45 Stecker elektrisch am Bus hängen. Der Master hat nur einen Stecker und hat seine Adresslogikleitung fix auf logisch 0 gelegt. Die Slaves hingegen brauchen zwei Stecker um die Adresskette zu bilden und unterscheiden zwischen eingehender Adresslogikleitung ADRI und ausgehender Adresslogikleitung ADRO. Wenn jedoch die Slaves falsche herum angeschlossen werden und ADRI und ADRO vertauscht sind kann dieser Fehler vom Mikrokontroller erkannt und behoben werden. Die genauen Details dieser Funktion werden in späteren Abschnitten erklärt.

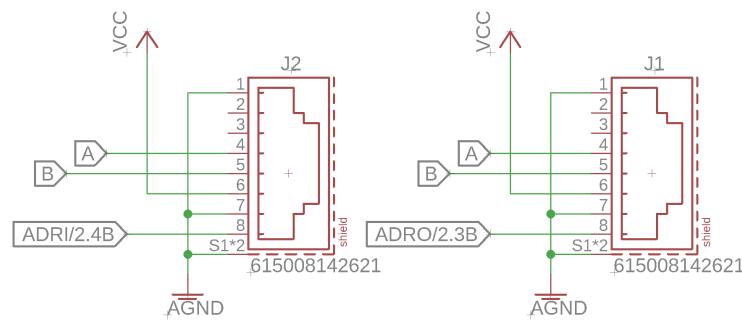


Abbildung 43: RJ45 Pinbelegung des Slave-Geräts

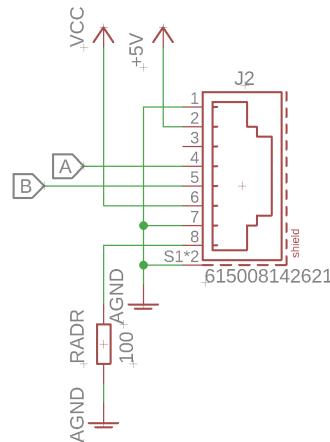


Abbildung 44: RJ45 Pinbelegung des Master-USB-Geräts

7.4.2.2 Versorgungsleitungen VCC und GND

Da es für jeden Parkplatz ein Slave-Gerät gibt ist die Länge der Busleitung nicht unbeträchtlich. Es kann so entlang der Leitung zu abfallen in der Versorgungsspannung aufgrund des Widerstands der langen Leitung. Um dem entgegenzuwirken muss die Versorgungsspannung VCC bezogen auf Masse größer als die benötigte Spannung an den Geräten sein. Die Slave-Geräte besitzen alle einen Linearregler der für einen konstante Spannung von 5V sorgt. Die Eingangsspannung des Spannungsreglers muss je nach Ausführung um eine gewisse Differenzspannung größer sein als die am Ausgang benötigte Spannung sein. Dieser Wert wird als *Dropout Voltage* bezeichnet. Für den IC L78L05ACD lässt sich aus dem Datenblatt folgende Spannung nachlesen: 2 V

Table 4. Electrical characteristics of L78L05C - Refer to the test circuits, $T_J = 0$ to 125°C , $V_I = 10\text{ V}$, $I_O = 40\text{ mA}$, $C_I = 0.33\text{ }\mu\text{F}$, $C_O = 0.1\text{ }\mu\text{F}$ unless otherwise specified

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_O	Output voltage	$T_J = 25^\circ\text{C}$	4.6	5	5.4	V
V_O	Output voltage	$I_O = 1$ to 40 mA , $V_I = 7$ to 20 V	4.5		5.5	V
		$I_O = 1$ to 70 mA , $V_I = 10\text{ V}$	4.5		5.5	
ΔV_O	Line regulation	$V_I = 8.5$ to 20 V , $T_J = 25^\circ\text{C}$			200	mV
		$V_I = 9$ to 20 V , $T_J = 25^\circ\text{C}$			150	
ΔV_O	Load regulation	$I_O = 1$ to 100 mA , $T_J = 25^\circ\text{C}$			60	mV
		$I_O = 1$ to 40 mA , $T_J = 25^\circ\text{C}$			30	
I_d	Quiescent current	$T_J = 25^\circ\text{C}$			6	mA
		$T_J = 125^\circ\text{C}$			5.5	
ΔI_d	Quiescent current change	$I_O = 1$ to 40 mA			0.2	mA
		$V_I = 8$ to 20 V			1.5	
eN	Output noise voltage	$B = 10\text{ Hz}$ to 100 kHz , $T_J = 25^\circ\text{C}$		40		μV
SVR	Supply voltage rejection	$V_I = 9$ to 20 V , $f = 120\text{ Hz}$ $I_O = 40\text{ mA}$, $T_J = 25^\circ\text{C}$	40	49		dB
V_d	Dropout voltage			2		V

Abbildung 45: Elektrische Eigenschaften des L78L05ACD

Daraus lässt sich schließen dass der IC mindestens eine Spannung von 7 V braucht um zu funktionieren. Als Eingangsspannung für alle Messung wurde 9 V verwendet. Nach Datenblatt ist die Eingangsspannung auf einen Maximalwert von 30 V beschränkt. Es ist zu beachten, dass eine Erhöhung der Eingangsspannung zu mehr Verlusten am Linearregler führt und somit für eine schlechte Effizienz sorgt.

7.4.3 Implementation eines eigenen Protokolls

RS485 spezifiziert keine Protokoll unter welchem Busteilnehmer untereinander kommunizieren. Es gibt bereits viele fertige Lösungen wie zum Beispiel Modbus, welches auf einer Master/Slave-Architektur basiert. Es ist aber auch möglich ein eigenes Protokoll zu erstellen. In den folgenden Absätzen wird der Aufbau und die Funktion eines eigenen implementierten Protokolls für die Fahrzeugerkennung, erklärt.

7.4.3.1 Aufbau von Datenframes

Mit UART ist es möglich, unter der Konfiguration von 8 Datenbits, eine Byte pro Datenaustausch zu senden. Diese Bytes bilden die Grundbausteine unseres sogenannten Datenframes, welcher eine Aneinanderreihung von Bytes ist. Jedes Byte kann eine unterschiedliche Funktion haben. Es kann zu einem einfache Daten beinhalten oder dem anderen Teilnehmer bestimmte Steueranfragen mitteilen. Die Datenframes sind in folgende Grundstruktur aufgeteilt.

Datenframe																
Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Name	START	ADR	CTRL	ARG	DATA_0 bis DATA_7								STOP	LF		
Hexwert	0x02													0x03	0x11	

Tabelle 3: allgemeiner Aufbau eines Datenframes

- **START, STOP und LF**

Diese Bytes helfen den Geräte den Anfang und des Ende eines Frames zu erkennen. Das START-Byte signalisiert den Anfang des Datenframes und hat einen fixen von **0x02** während das Byte STOP mit dem fixen Hexwert von **0x03** das Ende das Ende signalisiert. Zusätzlich kommt am Ende noch das LF-Byte mit einem Hexwert von **0x11**. In ASCII kodiert ist das das Zeichen einer neue Zeile in einem Text und ist vor allem nützlich, da viele Bibliotheken dieses Zeichen als Endzeichen zu Datenauslesung aus einem seriellen Buffer verwenden.

- **ADR**

Jedes Gerät am Bus ist unter seiner eigenen Adresse erreichbar welche aus zwei Bytes besteht. Die Adressen werden über den Master verteilt, der eine fixe Adresse von **0x01** besitzt. Zusätzlich sind alle Slaves unter einer Broadcastadresse von **0x00** erreichbar. Es ergibt sich mit zwei Bytes eine Adressraum der Größe $2^{16} = 65536$ inklusive Master und Broadcast.

- **CTRL**

Das Kontrollbyte CTRL signalisiert einen bestimmten Steuerbefehl. Mit ihm kann man eine Frequenzmessung der Spule an einem Slave auslösen oder einzelne LEDs ein- und ausschalten.

- **ARG**

Die Argumentbytes ARG1 und ARG2 geben dem Steuerbefehl zusätzliche Daten mit. So kann man zum Beispiel zwischen LED1 und LED0 und einschalten und ausschalten unterscheiden.

- **DATA**

Hier versteht man 8 Datenbytes, die für den Austausch von größeren Werten, wie einer Frequenz von nutzen sind. Diese Datenbytes sind in ASCII kodiert um Konflikte mit den START, STOP und LF Bytes zu vermeiden. Jedes Datenbyte kann daher einen Hexwert von $0x0$ bis $0xF$. Der größtmögliche Wert, der sich verschicken lässt ist daher $0xFFFFFFFF$ was in Dezimal einem Wert von $2^{32} - 1 = 4\,294\,967\,294$ entspricht.

7.4.3.2 Steuerabläufe

Um eine Aktion auszuführen schreibt der Master auf den Bus ein Datenframe und wartet auf die Antwort des angesprochenen Slaves.

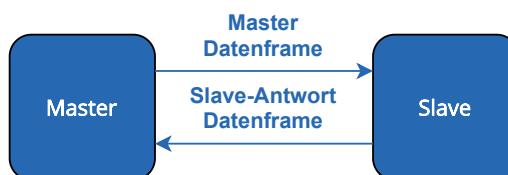


Abbildung 46: Ablauf von Steueranfragen

Die Art des Steuerbefehls wird durch das CTRL-Byte angegeben und definiert so die Inhalte der Bytes. Es muss aber zwischen einem Master Datenframe und einem Slave-Response Datenframe unterschieden werden. In den nächsten Abbildung werden die möglichen Steuerbefehle für Master und Slave in Tabellenform angeführt.

7.5 Mikrokontroller Slave-Geräte

7.5.1 Überblick

7.5.2 Atmega328PB

7.5.3 Peripherie des Mikrocontrollers

7.5.3.1 Spannungswandler

ADR_MSB		ADR_LSB		CTRL		ARG_1		MASTER		DATA_0	
DEC	HEX	DEC	HEX	ASCII	DEC	HEX	Function	ASCII	DEC	HEX	NAME
0	0x00	0	0x00	A	65	0x41	ADR	G	71	0x47	GIVE
0	0x00	0	0x00	R	82	0x52	REMOVE	0	0x00	-	ADR_MSB
var	var	S	83	0x53	GET_STATE	0	0x00	-	0	0x00	-
var	var	F	70	0x46	GET_FRQ	0	0x00	-	0	0x00	-
var	var	L	76	0x4C	GET_L	0	0x00	-	0	0x00	-
var	var	0	48	0x30	I00	D	68	0x44	SET_IN_OUT	var	IN_OUT
var	var				B	66	0x42	READ	0	0x00	-
var	var			I	73	0x49	SET	var	ON_OFF	-	-
var	var	1	49	0x31	I01	D	68	0x44	SET_IN_OUT	var	IN_OUT
var	var				B	66	0x42	READ	0	0x00	-
var	var			I	73	0x49	SET	0	0x00	ON_OFF	-
var	var	C	67	0x43	CALIBRATE	-	-	-	0	0x00	FRQ_MSB
var	var								0	0x00	FRQ_LSB
var	var			0x00					-	-	-
var	var			0x00					0	0x00	-
var	var	P	80	0x50	PING	-	-	-	0	0x00	-

Abbildung 47: Master Datenframe

ADR_MSB	ADR_LSB	CTRL			ARG_1			SLAVE RESPONSE			DATA_0												
		DEC	HEX	DEC	HEX	ASCII	DEC	HEX	Function	ASCII	DEC	HEX	NAME	DEC	HEX	NAME	-	-	-	-	-	-	-
0	0x00	1	0x01	A	65	0x41	ADR	G	71	0x47	GIVE	1	0x01	-	-	-	-	-	-	-	-	-	-
0	0x00	0	0x00	S	83	0x53	GET_STATE	R	82	0x52	REMOVE	0	0x00	-	0	0x00	-	100_IN_OUT	100_STATE	101_IN_OUT	101_STATE	-	-
0	0x00	1	0x01	F	70	0x46	GET_FREQ	O	0	0x00	-	0	0x00	-	0	0x00	-	FREQ_MSB	FREQ	FREQ_FRG	FREQ_LSB	-	-
0	0x00	1	0x01	L	76	0x4C	GET_L	O	0	0x00	-	0	0x00	-	0	0x00	-	L	-	-	-	-	-
0	0x00	1	0x01	O	48	0x30	0x30	D	68	0x44	SET_IN_OUT	var	var	IN_OUT	var	var	IN_OUT	-	-	-	-	-	-
0	0x00	1	0x01	O	0	B	66	0x42	READ	O	0x00	-	0	0x00	-	0	0x00	-	READ_100	-	-	-	-
0	0x00	1	0x01	O	0	I	73	0x49	SET	var	var	ON/OFF	ON	0	OFF	READ_100	-	-	-	-	-	-	-
0	0x00	1	0x01	I	49	0x31	101	D	68	0x44	SET_IN_OUT	var	var	IN_OUT	IN	OUT	IN_OUT	-	-	-	-	-	-
0	0x00	1	0x01	O	0	B	66	0x42	READ	O	0x00	-	0	0x00	-	0	0x00	-	READ_101	-	-	-	-
0	0x00	1	0x01	O	0	I	73	0x49	SET	O	0x00	-	0	0x00	-	0	0x00	-	ON/OFF	READ_101	-	-	-
0	0x00	1	0x01	C	67	0x43	CALIBRATE	-	-	-	-	0	0x00	-	F0_MSB	F0_LSB	-	-	-	-	-	-	-
0	0x00	1	0x01	O	0x00	0	-	-	-	-	0	0x00	-	0	0x00	-	-	-	-	-	-	-	
0	0x00	1	0x01	P	80	0x50	PING	-	-	-	0	0x00	-	ADR_MSB	ADR_LSB	LOCAL_ADR_MSB	LOCAL_ADR_LSB	CTRL	ARG_1	ARG_2	CROSSOVER	-	-

Abbildung 48: Slave Response Datenframe

7.5.3.2 RS485 Pegelwandler

7.5.3.3 Digitale Ein- und Ausgänge

7.5.4 Layout des Slave-Gerätes

7.5.5 Gehäuse

7.6 USB-Master

7.6.1 USB-Bussadapter Gerät

7.6.1.1 Überblick

7.6.1.2 FT232RL

7.6.1.3 Spannungsversorgung

7.6.1.4 USB-C Anschluss

7.6.1.5 Layout des Master-Geräts

7.6.1.6 Gehäuse

7.6.2 Master Programm

7.6.2.1 Benötigte Software

7.6.2.2 Adressvergabe

7.6.2.3 Frequenzauslesung

7.6.2.4 Auswertung

7.6.2.5 API-Post

7.6.3 RaspberryPi als Mastergerät

7.6.3.1 SSH Remote Zugriff

7.6.3.2 Code Deployment

7.6.3.3 Unitest

8 Webinterface

8.1 Einleitung

Das Webinterface hat auf der einen Seite die Aufgabe die Kommunikation mit der Kennzeichenerkennung und der Fahrzeugerkennung sicherzustellen und auf der anderen Seite die Verwaltung und Darstellung der gewonnenen Daten.

8.2 Verwendete Technologien

8.2.1 HTML

HTML steht hierbei für Hypertext Markup Language und ist eine Auszeichnungssprache welche vom World Wide Web Consortium (W3C)¹³ entwickelt wird. Hypertext Markup Language (HTML) ist De-Facto-Standard um Inhalte in Browsern darzustellen. HTML ist dabei aber nicht für die visuelle Darstellung verantwortlich sondern nur für die semantische Struktur. Der Sinn dahinter ist, dass der Inhalt und die Vorgaben an die Darstellung möglichst gut getrennt ist. Für die Formatierung kommt die Stylesheet-Sprache Cascading Style Sheets (CSS) zum Einsatz, welche ebenfalls vom World Wide Web Consortium entwickelt wird. Die aktuellste Version der HTML Spezifikation ist HTML5¹⁴ und wurde am 28. Oktober 2014 vom W3C vorgelegt.



Abbildung 49: HTML5 Logo von <https://www.w3.org/html/logo>

¹³<https://www.w3.org>

¹⁴<https://www.w3.org/2014/10/html5-rec.html.en>

8.2.1.1 Beispielhafte HTML Seite

Eine HTML Seite setzt sich aus einer Vielzahl von sogenannten Elementen zusammen. Ein Element besteht aus einem Start Tag und aus einem End Tag, der Inhalt wird zwischen diese Tags geschrieben. Nun folgt eine einfache HTML Seite, welche die Grundlegenden Funktionen von HTML und CSS darlegen soll.

```
1  <!DOCTYPE html>
2
3  <html>
4  <head>
5    <title>Titel</title>ü
6  </head>
7
8  <body>
9    <h1>Überschrift</h1>
10   <p>Paragraph</p>
11
12 </body>
13 </html>
```

Code 37: index.html

Das Element `<!DOCTYPE html>` deklariert, dass die folgende Seite den HTML5 Standard verwendet. Danach folgt mit `<html>` das Wurzelement, dass alle anderen Elemente beinhaltet. Das `<head>` Element beinhaltet verschiedene Metadaten d.h. Daten die nicht angezeigt werden. Im oben gezeigten Beispiel Code 37 wird nur der Title des Dokuments gesetzt, dieser wird im Browser Tab angezeigt. Es können aber auch noch andere Daten gesetzt bzw. eingebunden werden:

- Character Set
- Styles
- Scripts
- Viewport

- Sonstige Metainformationen (Author, Keywords)

Überschrift

Paragraph

Abbildung 50: Einfache HTML Seite von <https://www.w3.org/html/logo>

8.2.2 CSS

Wie bereits angesprochen ist Cascading Style Sheets (CSS) für die Formatierung bzw. die visuelle Darstellung der einzelnen HTML-Elemente verantwortlich. Der Standard wird wie bei HTML vom W3C spezifiziert und die aktuellste Version ist CSS3 was so viel bedeutet wie CSS Level 3, wobei nur einzelne Teile als Empfehlung durch das W3C vorgelegt wurden, beispielweise das CSS Color Module Level 3¹⁵. Um die Funktion darzustellen wird die vorherige HTML Seite nun mit CSS ergänzt.

¹⁵<https://www.w3.org/TR/css-color-3>

```
1  body {  
2      background-color: deepskyblue;  
3  }  
4  
5  h1 {  
6      color: white;  
7      text-align: center;  
8      font-family: verdana;  
9  }  
10  
11 p {  
12     color: wheat;  
13     font-family: verdana;  
14     font-size: 20px;  
15 }
```

Code 38: style.css

Nun muss dieses Stylesheet nur noch im <head> Tag mit

```
<link rel="stylesheet" href="style.css">
```

 eingebunden werden.

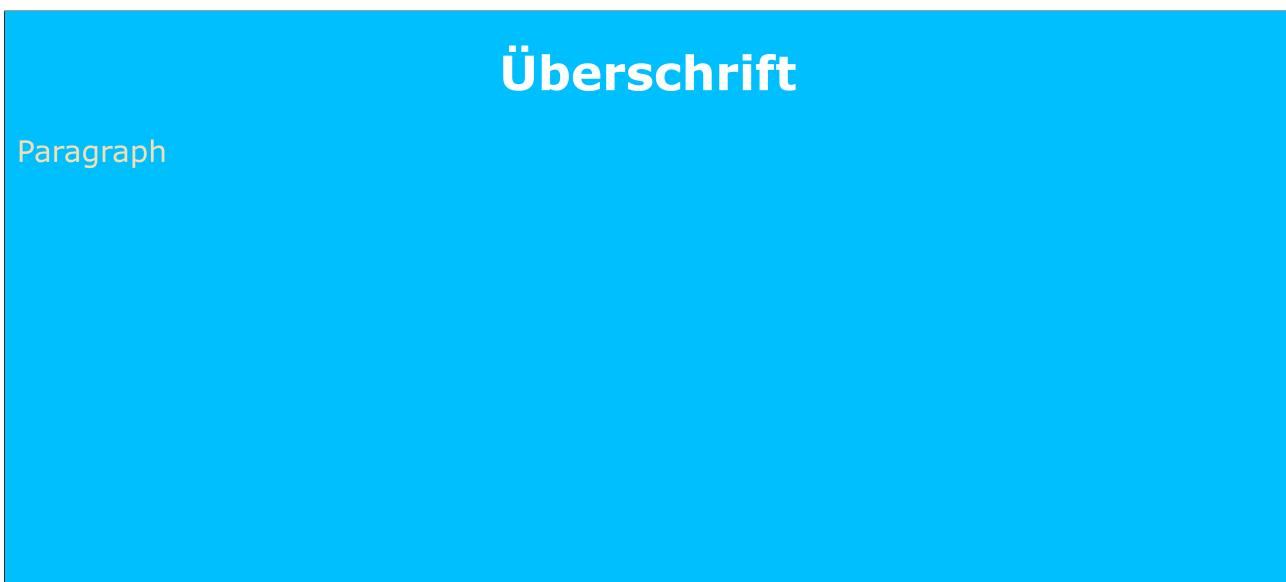


Abbildung 51: Einfache HTML Seite mit CSS

Es lässt sich nun erkennen, dass sich die Webseite deutlich verändert hat. Dabei bieten HTML und CSS bieten noch viel mehr Funktionen, eine ausführliche Dokumentation der Funktionen sind auf der Website w3schools¹⁶ zu finden.

8.2.3 JavaScript

Es folgt nun eine weitere sehr wichtige Technologie und die meist verwendete Programmiersprache überhaupt laut der Stack Overflow Developer Survey 2020¹⁷. JavaScript (JS) ermöglicht es dynamische Webseiten zu erstellen, dabei wird der Code direkt lokal im Browser ausgeführt. Jedoch ist JavaScript nicht mehr nur auf das Frontend¹⁸ mehr beschränkt, es möglich mit Frameworks wie Node.js auch Backend¹⁹ Applikationen zu schreiben und somit ist es möglich Full-Stack²⁰-Anwendungen vollständig mit JavaScript zu entwickeln. Eine der wichtigsten Anwendungsbereiche ist die Manipulation von Elementen über das Document Object Model (DOM). Der Standard wird unter dem Namen ECMA Script von der Organisation Ecma International²¹ veröffentlicht und die aktuellste Version ist **ECMA-262**.

¹⁶<https://www.w3schools.com/html> und <https://www.w3schools.com/css>

¹⁷<https://insights.stackoverflow.com/survey/2020>

¹⁸Präsentationsebene in Form der grafischen Benutzeroberfläche

¹⁹Verarbeitung von Daten auf beispielsweise einem Server

²⁰Front- und Backend

²¹<https://www.ecma-international.org>

8.2.4 PHP

Hypertext Preprocessor (PHP) ist eine Skriptsprache um dynamische Webseiten zu realisieren, jedoch wird nicht wie bei JavaScript der Code auf dem Client ausgeführt sondern auf dem Server, dort wird die HTML-Ausgabe generiert und dem Client zugesendet. Somit ist es nicht möglich den Code als Benutzer zu betrachten. Grundsätzlich ist PHP als synchrone Sprache geplant worden, es ist jedoch auch möglich asynchron zu Programmieren um die Performance zu steigern. Die aktuellste Version ist PHP 8²².

8.2.5 TailwindCSS

Es gibt eine Vielzahl von CSS-Frameworks, das Ziel ist ein einfacheres und schnelleres erstellen von Webseiten, dazu gehören neben TailwindCSS folgende relevanten Frameworks.

- Bootstrap (<https://getbootstrap.com>)
- Foundation (<https://get.foundation>)
- Materialize (<https://materializecss.com>)

Viele von diesen CSS Frameworks verwenden vorgefertigte Components welche direkt verwendet werden können, dies führt dazu, dass die Entwicklung sehr rasch ist. Dort unterscheidet sich TailwindCSS von den anderen CSS Frameworks, dort existieren sogenannte Utility-Classes, diese können direkt im HTML auf die einzelnen Elemente angewendet werden.

8.2.6 Laravel

Laravel²³ ist ein Open-Source PHP-Framework, es erleichtert die Entwicklung und erhöht die Sicherheit und auch durch die zahlreichen First-Party-Packages bietet Laravel ein sehr hochwertiges Ecosystem. Da Laravel ein sehr wichtiger Bestandteil des Webinterfaces ist wird später noch genauer auf die einzelne Funktionen des Frameworks eingegangen. Laravel wird seit Juni 2011 entwickelt und erhält jährlich eine neue Version, aktuell ist Laravel 8 die neuste Version.

²²<https://www.php.net/docs.php>

²³<https://laravel.com/>

Das Framework folgt dem sogenannten Model-View-Controller (MVC) Muster das bedeutet, dass die Programmierlogik in drei verschiedene Teile unterteilt wird. Der Sinn dahinter ist, dass die Anwendung dadurch sehr flexibel ist und später leichter erweitert werden kann oder einzelne Teile wiederverwendet werden können.

- **Model**

Hier befindet sich die Datenstruktur der Anwendung. In Laravel kann dies beispielsweise das Model `User` sein.

- **View**

In der View befindet sich die Präsentationsebene, im Fall von Laravel sind das Components und Layouts.

- **Controller**

In den Controllern der Anwendung befindet sich die Logik um beispielsweise durch das Absenden eines Formulares einen Eintrag in der Datenbank zu erstellen.

Ein Anfrage auf eine Webseite läuft in 6. Schritten ab:

- **1. Schritt:** Der Benutzer führt eine HTTP-Anfragemethode aus
- **2. Schritt:** Die Anfrage wird geroutet
- **3. Schritt:** Der Controller interagiert mit dem Model
- **4. Schritt:** Das Model greift auf die Datenbank zu
- **5. Schritt:** Der Controller liefert die Daten an eine View
- **6. Schritt:** View mit den Daten werden an den Client gesendet

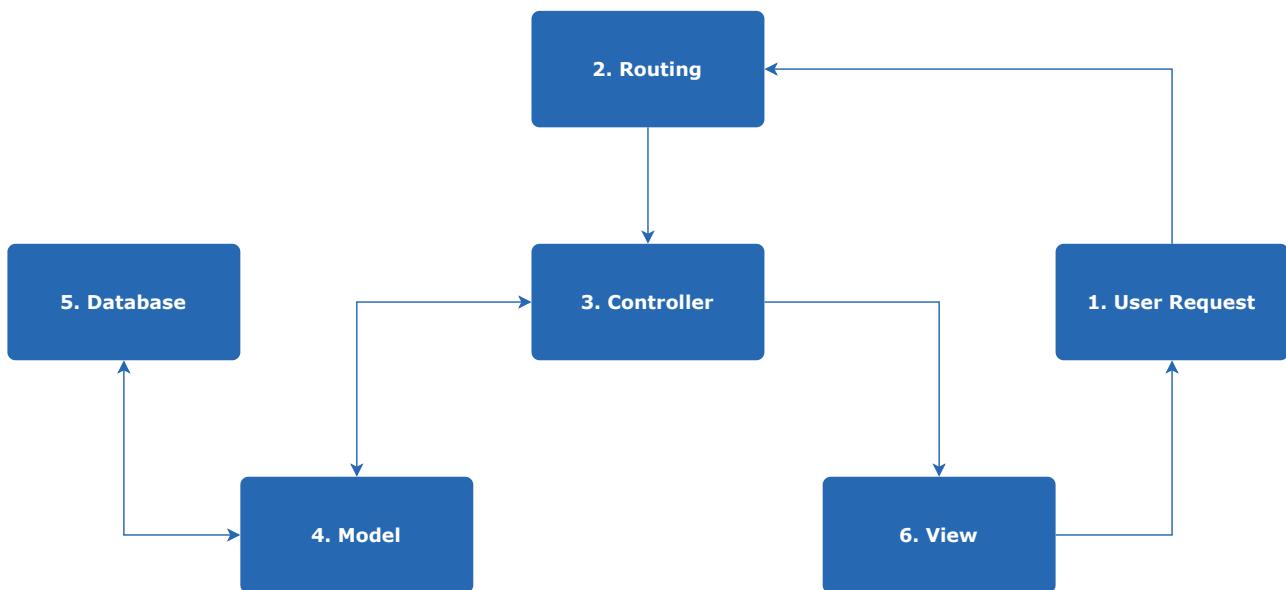


Abbildung 52: Laravel MVC Muster

8.2.6.1 Routing

In Laravel erfolgt dies durch ein Routefile unter `routes/web.php`. API Requests haben ein eigenes Routefile unter `routes/api.php` und erhalten einen `/api` Prefix. Diese Dateien werden dann automatisch durch einen Service Provider von Laravel geladen.

Der Laravel Router erlaubt folgende HTTP-Anfragemethoden:

- **GET** (Fordert Ressource an)

```
Route::get($uri, $callback);
```

- **POST** (Neue Ressource erstellen)

```
Route::post($uri, $callback);;
```

- **PUT** (Ressource ersetzen oder erstellen)

```
Route::put($uri, $callback);
```

- **PATCH** (Ressource ändern)

```
Route::patch($uri, $callback);
```

- **DELETE** (Löscht Ressource)

```
Route::delete($uri, $callback);
```

- **OPTIONS** (Liste von unterstützen Methoden des Servers)

```
Route::options($uri, $callback);
```

Im folgenden Ausschnitt Code 39 werden ein Teil der Routen von den Benutzern dargelegt. Dabei lässt sich erkennen, dass die Routen sich in einer Gruppe befinden, welche den Prefix /admin und die Middleware verified hat das bedeutet, dass der Benutzer eingeloggt sein muss um diese Routen aufzurufen. Dabei verweisen die Routen auf einzelne Methoden im UserController. Schlussendlich werden mit der name-Methode die Routen benannt, damit sie später im Code einfach referenziert werden können.

```
1 <?php
2 Route::group(['middleware' => 'verified', 'prefix' => 'admin'],
3 function () {
4     Route::get('users', [UserController::class,
5         'index'])->name('users.index');
6     Route::get('users/create', [UserController::class,
7         'create'])->name('users.create');
8     Route::post('users', [UserController::class,
9         'store'])->name('users.store');
10 });
11 
```

Code 39: web.php

8.2.6.2 Blade Templates

Blade ist eine Template Engine, die mit Laravel mitgeliefert wird. Blade erleichtert und vereinfacht das Verwenden von PHP Code in HTML, dabei wird der Blade Syntax in normalen PHP Code compiliert. Blade Dateien werden mit der Extension .blade.php erstellt. Damit der Inhalt der einzelnen Seiten dynamisch ist müssen Daten übergeben werden, dies erfolgt über die Controller.

Um nun Inhalt innerhalb eines Blade Files anzuzeigen verwendet Blade doppelt Geschweifte Klammern.

```
1 Hallo, {{ $user->full_name }}.
```

Code 40: example.blade.php

Im Code 40 wird nun auf das übergebene Model User zugegriffen und der Name ausgegeben.

8.2.6.3 Controllers

Es besteht theoretisch die Möglichkeit die komplette Logik für die Bearbeitung von Anfragen direkt in den Route Files zu platzieren, dies ist jedoch sehr unübersichtlich und deshalb ist es sinnvoll diese Logik in Controller Klassen auszulagern. Dabei werden ähnliche Anfragen zusammengeführt, so ist es sinnvoll alle Anfragen wie in Code 39 im gleichen Controller zusammenzuführen.

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5
6 use Illuminate\Http\Request;
7 use App\Models\User;
8 use Illuminate\Support\Facades\Hash;
9 use App\Http\Requests\StoreUser;
10 use App\Http\Requests\UpdateUser;
11 use App\Models\Role;
12
13 use Image;
14
15
16 class UserController extends Controller
17 {
18     public function index()
19     {
20         $this->authorize('index', User::class);
21
22         $users = User::orderBy('id', 'asc')->paginate(25, ['*'],
23             ['users']);
24
25         return view('users.index')->with('users', $users);
26     }
27
28     . . .
```

Code 41: UserController.php

8.2.6.4 Artisan CLI

Artisan ist ein Kommandozeilen Tool von Laravel und zentraler Bestandteil und stellt eine Fülle an sehr nützlichen Befehlen dem Entwickler zur Verfügung. Mithilfe von Artisan können neue Controller, Models, Migrations und vieles sehr einfach erstellt werden. Da es sehr viele Befehle gibt folgen hier nur die wichtigstens Befehle:

- **php artisan list**

Zeigt eine Liste aller verfügbaren Befehle an

- **php artisan serve**

Startet einen PHP Development Server unter dem Port 8000

- **php artisan make:model <name>**

Erstellt ein neues Model

- **php artisan make:migration <name>**

Erstellt ein neue Migrations

- **php artisan make:controller <name>**

Erstellt ein neuer Controller

- **php artisan migrate**

Führt die Datenbank Migration Files aus

8.2.6.5 Migrations

Datenbank Migrations sind praktisch Vorlagen wie einzelne Tabellen in der Datenbank auszusehen haben. Migrations erleichtern die Handhabung von SQL Datenbank im Team mithilfe von Source Control wie Git um einiges.

```
1 <?php
2 class CreateUsersTable extends Migration
3 {
4     public function up()
5     {
6         Schema::create('users', function (Blueprint $table) {
7             $table->id();
8             $table->string('first_name');
9             $table->string('last_name');
10            $table->string('email')->unique();
11            $table->string('avatar')->default('default.png');
12            $table->timestamp('email_verified_at')->nullable();
13            $table->string('password');
14            $table->string('last_login_ip')->nullable();
15            $table->timestamp('last_login_at')->nullable();
16            $table->rememberToken();
17            $table->timestamps();
18        });
19    }
20
21    public function down()
22    {
23        Schema::dropIfExists('users');
24    }
25 }
```

Code 42: create_users_table.php

Im Code 42 sind zwei Methoden zu erkennen, up und down, in der up Methode werden neue Tabellen und Spalten in der Datenbank erstellt, bei der down Methode wird alles von der up Methode rückgängig gemacht, in diesem Fall wird die komplette Tabelle gelöscht. Um nun die Datenbank

zu migrieren muss in der Artisan CLI Befehl `php artisan migrate` ausgeführt werden, dabei werden alle Migrations im Verzeichnis `database/migrations` ausgeführt.

8.2.6.6 Eloquent ORM

Eloquent ist ein Object-relational mapping (ORM) d.h. um auf die SQL Datenbank zuzugreifen müssen keine Raw SQL Queries ausgeführt werden. Im Code erscheint die Datenbank als objektorientierte Datenbank und erleichtert somit dem Umgang mit der Datenbank. Dabei entspricht besitzt jede Tabelle in der Datenbank einem dazugehöriges Model, dieses Model wird verwendet um im Code mit der Datenbank zu interagieren. Ein weiter Vorteil ist, dass der Code sehr übersichtlich und leicht lesbar ist, jedoch wird die Ausführungszeit leicht erhöht, was jedoch keine große Rolle bei kleinen- bis mittelgroßen Webseiten spielt.

```

1 <?php
2 User::where('first_name', 'Philipp')->first();
```

Code 43: Eloquent Query

```

1 SELECT * FROM `users` WHERE `first_name` = Philipp
```

Code 44: Raw SQL Query

Beim Vergleich zwischen dem Code 43 und Code 44 ist es erkenntlich, dass der Eloquent Query verständlicher ist. Besonders bei komplexeren Queries ist Eloquent den RAW SQL Queries im Bezug auf die Lesbarkeit stark überlegen.

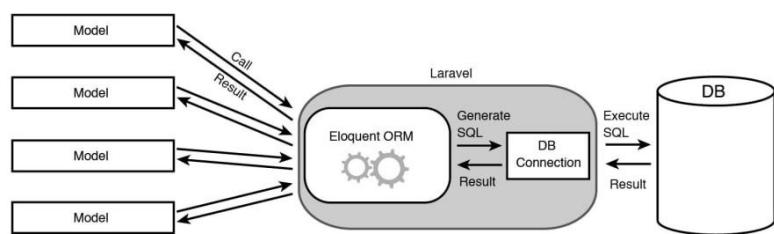


Abbildung 53: Eloquent ORM Workflow von <https://dev.to/xenoxdev/mastering-laravel-eloquent-orm-the-eloquent-journey-part-1-1571>

8.2.6.7 Laravel Sanctum

Laravel Sanctum ist ein First-Party-Package von den Entwicklern von Laravel und ermöglicht eine einfache Authentifizierung mithilfe von API-Tokens. Dabei wird der vom Benutzer erstellte API-Token im HTTP Header `Authorization` mitgeliefert und somit autorisiert. Da es sich hierbei um ein Sicherheitskritisches Modul handelt ist es besonders wichtig, dass dieser Code nahezu Fehlerfrei ist und dies wird durch kontinuierliche Updates und Patches garantiert.

8.2.6.8 Sessions

Das HTTP Netzwerkprotokoll ist ein Zustandsloses Protokoll, da aber immer über einen Benutzer Informationen über mehrere Seiten beibehalten werden sollen wird ein System benötigt um diese Informationen zu speichern. Sessions lösen dieses Problem und funktionieren wie folgt. Ein Benutzer besucht eine Seite und besitzt im lokalen Speicher (Cookie) eine Session ID vom Server. Wenn der Benutzer nun eine andere Seite der gleichen Webseite besucht wird diese Session ID dem Server wieder gesendet und er kann den Benutzer zuordnen. Laravel verwendet als Backend für die Sessions Memcached oder Redis.

8.2.6.9 Middleware

Eine Middleware erleichtert es HTTP Anfragen zu filtern bzw. zu begutachten. Beispielsweise wird im Webinterface eine Middleware für das richtige setzen der Sprache verwendet. Bei jedem Aufruf einer Seite überprüft die Middleware welcher Wert in der Sessions des Benutzers steht um so die passende Sprache auszuwählen.

8.2.6.10 Service Provider

Service Provider sind da um bestimmte Funktionen einer Anwendung zu initialisieren. Beispielsweise wird im Webinterface ein `SettingsServiceProvider` verwendet um die Einstellungen aus der Datenbank in das Laravel Einstellungssystem zu übernehmen.

8.2.6.11 Form Validation

Für bestimmte Formulare ist es notwendig zu überprüfen ob die Angaben korrekt sind, beispielsweise dass ein Passwort eine bestimmte Länge hat oder ob eine E-Mail ein @-Zeichen enthält. Um diese HTTP-Anfragen zu überprüfen müssen Regeln festgelegt, diese werden in Laravel in `Request` Files festgelegt.

```
1 <?php
2 public function rules()
3 {
4     return [
5         'first_name' => ['string', 'max:255'],
6         'last_name' => ['string', 'max:255'],
7         'email' => ['string', 'email', 'max:255'],
8         'password' => ['nullable', 'string', 'min:8', 'confirmed'],
9     ];
10 }
```

Code 45: UserUpdate Request

Im Code 45 werden Regeln für den Vornamen, Nachnamen, die Email und für das Passwort festgelegt. Die verfügbaren Regeln sind durch Laravel definiert. Liefert eine Request Methode *false* wird der Controller nicht weiter ausgeführt und die nicht korrekt aus gefüllten Felder werden in der Session gespeichert und können dem User als Fehlernachricht dargestellt werden.

8.3 Lokale Entwicklungsumgebung mit Laragon

Für die Programmierung des Webinterfaces müssen zuerst einige Vorkehrungen getroffen werden, dazu zählt zu einem die Installation von benötigter Software und deren Konfiguration.

8.3.1 Benötigte Software

- **Laragon** (<https://laragon.org>)

Beinhaltet mehrere Softwarepakete die für die Entwicklung notwendig sind.

- Apache HTTP Server
- MySQL
- PHP

- **phpMyAdmin** (<https://www.phpmyadmin.net>)
Webinterface für MySQL
- **Composer** (<https://getcomposer.org>)
Paketmanager für PHP
- **Git** (<https://git-scm.com>)
Versionskontrolle
- **Visual Studio Code** (<https://code.visualstudio.com>)
Quelltext-Editor

8.3.2 Konfiguration von PHP

Um PHP Befehle von der Kommandozeile auszuführen muss die Installation zuerst in den Windows Path Variables hinzugefügt werden.

Dies erfolgt durch die Advanced System Settings ► Environment Variables ► System Variables. Dort kann nun die Path Variable editiert werden und der Pfad hinzugefügt werden in welchem die php.exe liegt.

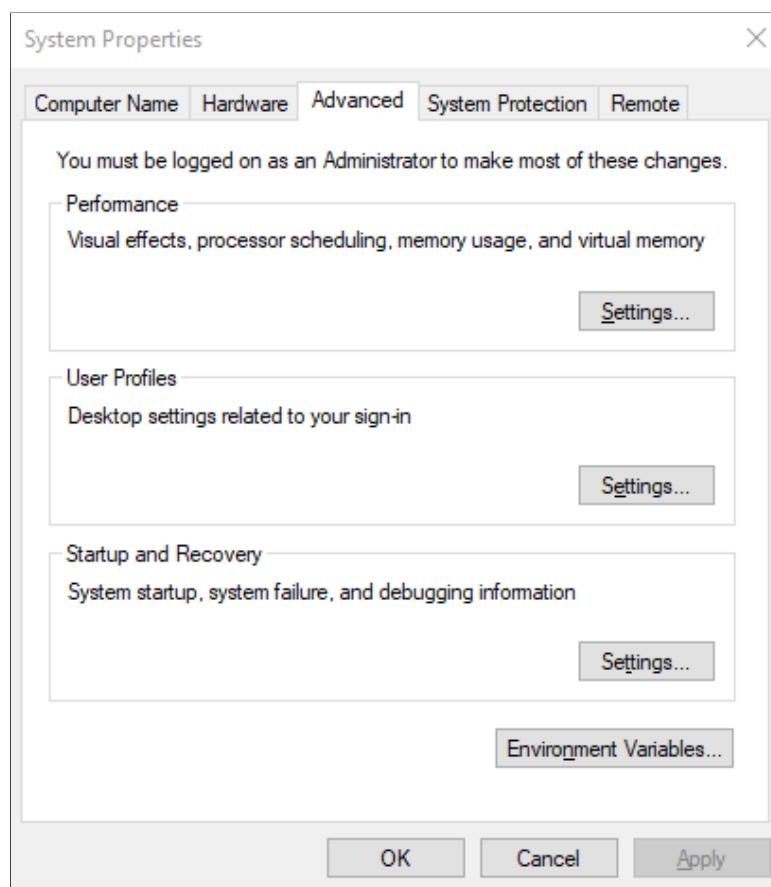


Abbildung 54: Advanced System Settings

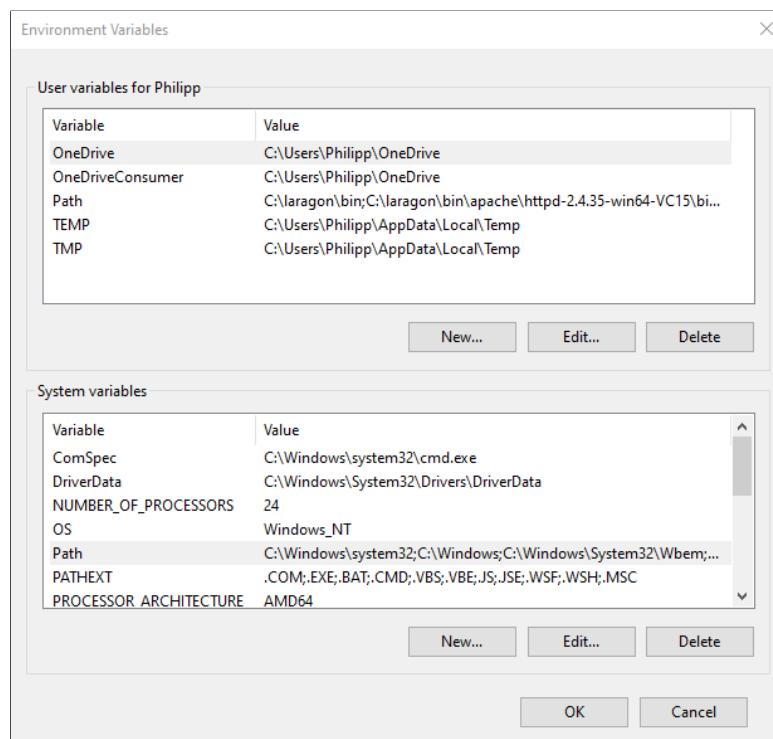


Abbildung 55: System Variables

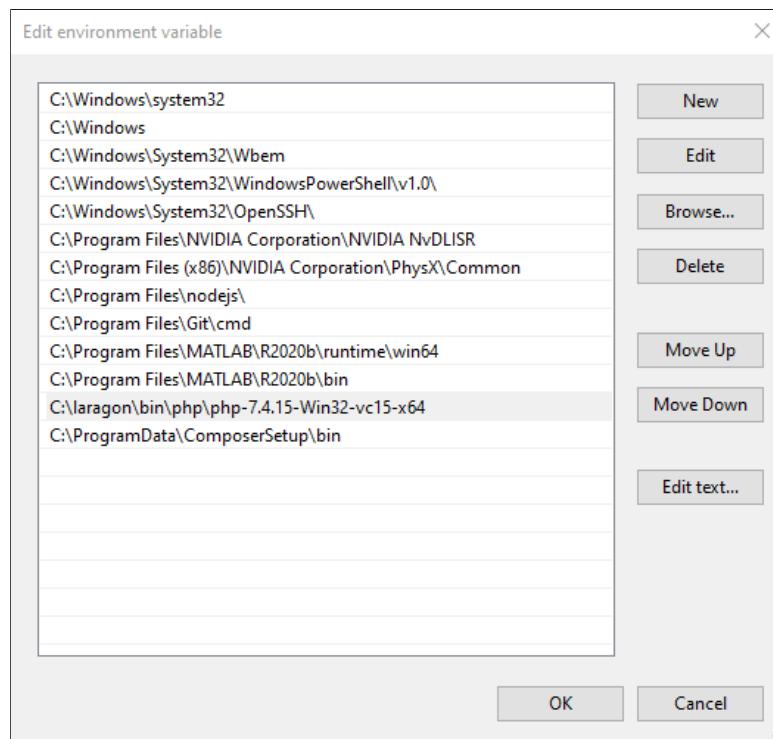


Abbildung 56: Environment Variables

Die korrekte Konfiguration kann durch die Kommandozeile geprüft werden, dort muss das Befehl

php -v ausgeführt werden. Dabei ist zu beachten, dass nach dem hinzufügen der Path Variable die gewählte Kommandozeile neu gestartet werden muss.

```
C:\Users\Philipp>php -v
PHP 7.4.15 (cli) (built: Feb 2 2021 20:47:45) ( ZTS Visual C++ 2017 x64 )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
```

Abbildung 57: PHP Version

Somit ist PHP korrekt konfiguriert.

8.3.3 Installation von phpMyAdmin

phpMyAdmin ist ein Tool, welches den Umgang mit MySQL Datenbanken mit einem Webinterface erleichtert. Die aktuellste Version lässt sich von <https://www.phpmyadmin.net/downloads> downloaden. Dieses Archiv muss entpackt werden und ausgehend vom Laragon Root Verzeichnis in das Verzeichnis /etc/apps kopiert werden. Um die Installation zu überprüfen muss der Apache HTTP Server und der MySQL Server gestartet werden, nun sollte bei einer korrekten Installation das Webinterface von phpMyAdmin unter <http://localhost/phpmyadmin> erreichbar sein.

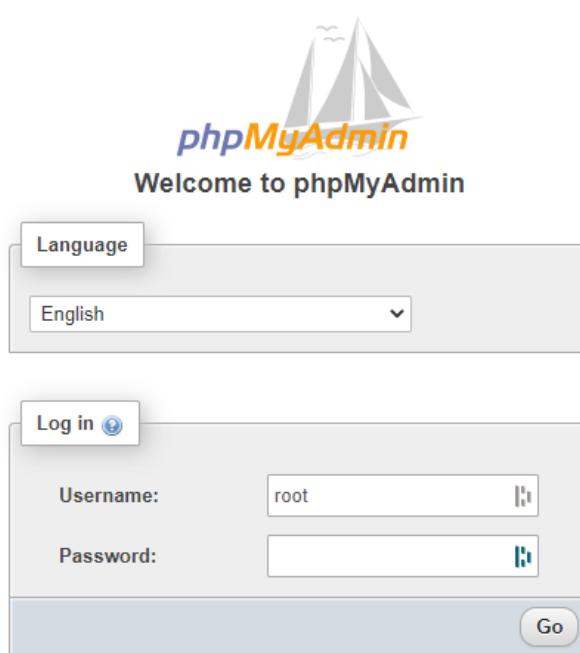


Abbildung 58: phpMyAdmin Webinterface

Es ist nicht notwendig ein Passwort einzugeben, da Standardmäßig kein Passwort gesetzt wird.

8.4 Lokale Entwicklungsumgebung mit WSL und Docker

8.4.1 Benötigte Software

- **Docker** (<https://www.docker.com>)
Ermöglicht Isolation von Anwendungen mit Containervirtualisierung
- **WSL** (<https://docs.microsoft.com/en-us/windows/wsl>)
Kompatibilitätsschicht für Linux Anwendungen unter Windows 10
- **Visual Studio Code** (<https://code.visualstudio.com>)
Quelltext-Editor

8.4.2 Installation von WSL

Zuerst müssen einige Einstellungen in Windows getroffen werden um später eine Linux Distribution herunterzuladen können. Diese Befehle können über die Kommandozeile mit Administrativen Rechten ausgeführt werden.

```
1  dism.exe /online /enable-feature
   ↳ /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

Code 46: WSL Feature Feature aktivieren

8.4.2.1 2. Schritt: Virtual Machine Aktivieren

```
1  dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all
   ↳ /norestart
```

Code 47: Virtual Machine Feature aktivieren

Nach diesem Schritt ist ein Neustart des Computers notwendig.

8.4.2.2 3. Schritt: Linux Kernel Update

Nun muss ein Linux Kernel Update installiert werden, die aktuelle Version ist unter <https://aka.ms/wsl2kernel> zu finden.

8.4.2.3 4. Schritt: WSL 2

Nach dem Neustart des Computers sollte es nun möglich sein WSL 2 als Version auszuwählen.

```
1 wsl --set-default-version 2
```

Code 48: WSL 2 auswählen

8.4.2.4 5. Schritt: Linux Distribution herunterladen

Zuletzt kann eine Linux Distribution aus dem Windows Store heruntergeladen werden, in diesem Fall Debian (<https://www.microsoft.com/de-de/p/debian>).

8.4.3 Installation von Docker

Die aktuellste Version von Docker Desktop für Windows lässt sich am einfachsten über die offizielle Website von Docker herunterladen (<https://docker.com>). Nach der Installation muss noch die WSL Integration aktiviert werden. Dazu muss in den Einstellungen unter Resources ► WSL Integration und dort muss der Haken bei *Enable integration with my default WSL distro* gesetzt werden und die installierte Linux Distribution muss unten aktiviert werden.

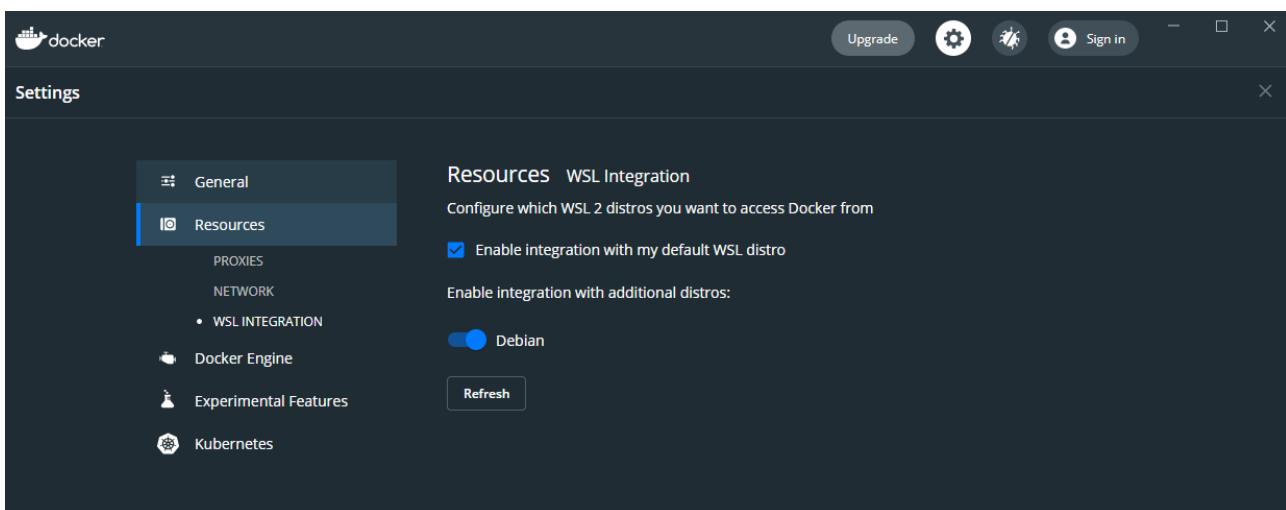


Abbildung 59: Docker WSL Integration

Somit ist die Lokale Entwicklungsumgebung mit WSL und Docker abgeschlossen, die benötigte Software wird später automatisch durch Laravel Sail in einem Docker Container installiert.

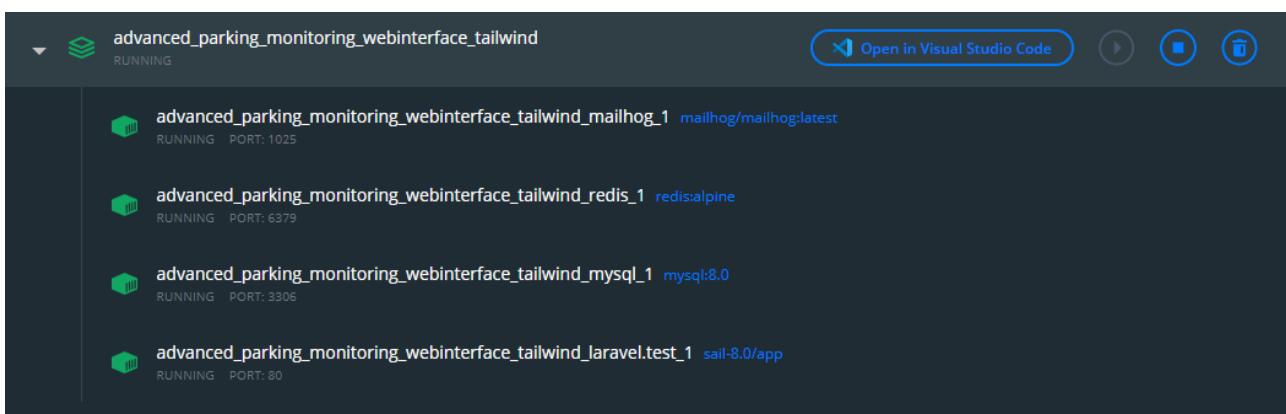


Abbildung 60: Docker Container Steuerung

Es ist somit möglich die Services welche im Container in der Linux Distribution laufen über die Docker Desktop Anwendung zu steuern.

8.5 Production Server

Der Production Server bzw. der Live Server ist der Server wo sich die Webanwendung befindet und die Endbenutzer zugreifen, dieser Server wird auch einfach mit Production abgekürzt. Der Production Server ist in diesem Fall ein Virtual Private Server mit dem Betriebssystem Debian 10, welcher bei einem Internet-Hosting Unternehmen mit Sitz in Deutschland gehostet wird.

8.5.1 Benötigte Software

Für den Live Server wird der sogenannte „LAMP“ Stack verwendet. LAMP steht dabei für die Software **L**inux, **A**pache, **M**ySQL und **P**HP.

- **Apache Web Server** (<https://httpd.apache.org>)
HTTP Server
- **MariaDB** (<https://mariadb.org>)
Fork von MySQL
- **PHP** (<https://www.php.net>)
Hypertext Preprocessor (PHP)
- **phpMyAdmin** (<https://www.phpmyadmin.net>)
Webinterface für MySQL
- **Composer** (<https://getcomposer.org>)
Paketmanager für PHP
- **Git** (<https://git-scm.com>)
Versionskontrolle

8.5.2 Installation des LAMP Stacks

Bevor die Software Pakete installiert werden sollte die Linux Software/Update Repository geupdated werden.

```
1 apt-get update && apt-get upgrade
```

Code 49: Respositories updaten

8.5.2.1 Apache

Nun kann der Apache Web Server installiert werden.

```
1 apt install apache2
```

Code 50: Apache installieren

Die Installation kann nun leicht überprüft werden indem man im Browser die IP bzw. die dazugehörige Domain öffnet, in diesem Fall: (<http://dev.philipp-kraft.com>).



Abbildung 61: Debian Default Page

Erscheint die Debian Default Page ist Apache korrekt installiert.

8.5.2.2 PHP

Neben PHP werden auch einige PHP Extensions benötigt.

```
1 apt install wget php php-cgi php-mysqli php-pear php-mbstring php-gettext
  ↳ libapache2-mod-php php-common php-phpseclib php-mysql
```

Code 51: PHP installieren

Die Installation kann einfach mit dem Befehl `php -v` überprüft werden.

8.5.2.3 MariaDB

```
1 apt install mariadb-server
```

Code 52: Mariadb installieren

Nun muss MariaDB noch konfiguriert werden.

```
1 mysql_secure_installation
```

Code 53: MariaDB Secure Installation

Dabei wird dem root MySQL User ein Passwort gesetzt, Anonyme Benutzer gelöscht und es werden Test Datenbanken gelöscht.

Nun wird ein neuer Benutzer mit root Berechtigungen erstellt.

```
1 mysql
2 GRANT ALL ON *.* TO 'admin'@'localhost' IDENTIFIED
3 BY 'password' WITH GRANT OPTION;
4 flush privileges;
5 exit
```

Code 54: MariaDB konfiguration

8.5.2.4 phpMyAdmin

Die aktuelle Version von phpMyAdmin kann von (<https://www.phpmyadmin.net/downloads>) bezogen werden und mit dem wget Befehl heruntergeladen werden.

```
1 wget https://files.phpmyadmin.net/phpMyAdmin/5.0.4
2 /phpMyAdmin-5.0.4-all-languages.tar.gz
```

Code 55: phpMyAdmin Download

Anschließend muss das Archiv entpackt werden.

```
1 tar xvf phpMyAdmin-5.0.4-all-languages.tar.gz
```

Code 56: phpMyAdmin Entpacken

Als nächstes muss das entpackte Archiv in einen anderen Pfad verschoben werden und zusätzlich müssen einige Rechte und Verzeichnisse angepasst werden.

```
1 mv phpMyAdmin-5.0.4-all-languages /usr/share/phpmyadmin  
2 mkdir -p /var/lib/phpmyadmin/tmp  
3 chown -R www-data:www-data /var/lib/phpmyadmin  
4 mkdir /etc/phpmyadmin/
```

Code 57: phpMyAdmin Rechte und Verzeichnisse

Nun muss eine Konfigurations Datei erstellt werden und dort muss ein Blowfish Secret²⁴ angegeben werden und den Pfad für ein Temporäres Verzeichnis.

```
1 cp /usr/share/phpmyadmin/config.sample.inc.php  
→ /usr/share/phpmyadmin/config.inc.php
```

Code 58: phpMyAdmin Konfigurationsdatei erstellen

und am Ende dieser Datei müssen folgende zwei Zeilen eingefügt werden.

```
1 $cfg['blowfish_secret'] = 'H20xcGXxf1Sd8JwrwVlh6KW6s2rER63i';  
2 $cfg['TempDir'] = '/var/lib/phpmyadmin/tmp';
```

Code 59: phpMyAdmin Blowfish Secret und TempDir

Zuletzt muss der Apache Web Server konfiguriert werden.

²⁴32 Zeichen String für Cookie-Authentifizierung

Im Verzeichnis /etc/apache2/sites-available muss eine neue Konfiguration angelegt werden `phpmyadmin.conf`.

```
1 Listen 9000
2
3 <VirtualHost *:9000>
4     ServerName localhost
5
6     <Directory /usr/share/phpmyadmin>
7         AllowOverride None
8         Require all granted
9     </Directory>
10
11    DocumentRoot /usr/share/phpmyadmin
12
13    ErrorLog ${APACHE_LOG_DIR}/phpmyadmin.error.log
14    CustomLog ${APACHE_LOG_DIR}/phpmyadmin.access.log combined
15 </VirtualHost>
```

Code 60: `phpmyadmin.conf`

Nun kann diese Virtual Host Konfigurations Datei aktiviert werden und danach muss der Apache Web Server neu gestartet werden.

```
1 a2ensite phpmyadmin
2 systemctl restart apache2
```

Code 61: Virtual Host aktivieren

Diese Konfiguration ermöglicht es, dass das Webinterface von phpMyAdmin über den Port 9000 (`http://dev.philipp-kraft.com:9000`) erreichbar ist und nicht wie Standardmäßig vorgesehen über das Verzeichnis /phpmyadmin (`http://dev.philipp-kraft.com/phpmyadmin`), dies bietet

einen Sicherheitsvorteil.

8.5.2.5 Webinterface Virtual Host

Nun muss noch eine Virtual Host Konfiguration für das Webinterface selbst erstellt werden.

```
1 <VirtualHost *:80>
2   ServerAdmin webmaster@localhost
3   DocumentRoot /var/www/apm/public
4
5   <Directory />
6     Options FollowSymLinks
7     AllowOverride All
8   </Directory>
9
10  <Directory /var/www/apm>
11    Options Indexes FollowSymLinks MultiViews
12    AllowOverride All
13    Order allow,deny
14    allow from all
15  </Directory>
16
17  ErrorLog ${APACHE_LOG_DIR}/error.log
18  CustomLog ${APACHE_LOG_DIR}/access.log combined
19 </VirtualHost>
```

Code 62: apm.conf

Diesmal wird auf den Standard HTTP Port 80 gehört und dieser führt in das Verzeichnis /var/www/apm/public. Zusätzlich werden noch Directives gesetzt, damit das Standard .htaccess File von Laravel richtig funktionieren kann.

8.5.2.6 Installation von Composer

Die Installation von Composer gestaltet sich relativ einfach.

```
1 wget -O composer-setup.php https://getcomposer.org/installer
```

Code 63: Download Composer Installer

Nun muss das Setup ausgeführt werden und damit das `composer` Befehl Global verfügbar ist wird Composer in den Pfad `/usr/local/bin` verschoben.

```
1 php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

Code 64: Composer Setup

8.5.3 Deployment mit Github Actions

Unter Deployment versteht man die automatische Installation von Software, in diesem Fall auf einem Linux Server. Erreicht wird das durch zwei Bash Scripts und mit Github Actions (<https://github.com/features/actions>).

8.5.3.1 Git Setup

Sollte auf dem Server noch kein Git installiert sein, lässt sich das wie folgt installieren.

```
1 apt install git
```

Code 65: Git Installation

Im Verzeichnis `/var/www/apm` soll sich später das Webinterface befinden, deshalb muss in diesem Pfad Git konfiguriert werden. Dazu wird die Remote URL konfiguriert.

```
1 git config remote.origin.url 'https://{{TOKEN}}@github.com/  
2 Philipp-Kraft/Advanced_Parking_Monitoring_Webinterface.git'
```

Code 66: Git Remote Origin

Da beim Github Account eine Zwei-Faktor-Authentisierung verwendet wird muss die Authentifizierung mit einem Personal access token erfolgen. Dieser kann unter <https://github.com/settings/tokens> erstellt werden, der erstellte Token kann dann einfach in der URL eingefügt werden.

8.5.3.2 Deploy Script

Das Deploy-Script wird auf der Lokalen Entwicklermaschine ausgeführt. Das Script wechselt in den Production Branch und merged mit dem Main Branch, dieser Push in den Production Branch löst dann die Github Action aus.

```
1 #!/bin/sh  
2 set -e  
3  
4 #vendor/bin/phpunit  
5  
6 (git push) || true  
7  
8 git checkout production  
9 git merge main  
10  
11 git push origin production  
12  
13 git checkout main
```

Code 67: deploy.sh

8.5.3.3 Server Deploy Script

Das Server Deploy Script `server_deploy.sh` versetzt die Laravel Applikation in den Wartungsmodus und lädt sich vom deploy Branch den Code auf den Server herunter, danach werden einige Befehle ausgeführt.

```
1 #!/bin/sh
2
3
4 echo "Deploying application . . ."
5
6 # Enter maintenance mode
7 php artisan down
8
9 # Update codebase
10 git fetch origin deploy
11 git reset --hard origin/deploy
12
13 # Install dependencies based on lock file
14 composer install --no-interaction --prefer-dist --optimize-autoloader
15
16 # Migrate database
17 php artisan migrate:refresh --seed
18
19 # Clear cache
20 php artisan optimize
21
22 # Exit maintenance mode
23 php artisan up
24
25 echo "Application deployed!"
```

Code 68: serverdeploy.sh

8.5.3.4 Github Action

Github Actions ist ein Projekt von Github, welches es ermöglicht Automatisierungen in den Bereichen Entwicklung, Testing und Deployment durchzuführen.

Als erstes muss ein Workflow erstellt werden, dieser wird im Root-Verzeichnis des Projekts erstellt `APM\github\workflows\main.yml`.

```
1  name: Deploy Laravel app
2
3  on:
4    push:
5      branches: [ production ]
6
7  jobs:
8    deploy:
9      runs-on: ubuntu-latest
10     steps:
11       - uses: actions/checkout@v2
12         with:
13           token: ${{ secrets.PUSH_TOKEN }}
14       - name: Set up Node
15         uses: actions/setup-node@v1
16         with:
17           node-version: '12.x'
18       - run: npm install
19       - run: npm run production
20       - name: Commit built assets
21         run: |
22           git config --local user.email "action@github.com"
23           git config --local user.name "GitHub Action"
```

```

24      git checkout -B deploy
25
26      git add -f public/
27      git commit -m "Build front-end assets"
28      git push -f origin deploy
29
30      - name: Deploy to production
31          uses: appleboy/ssh-action@master
32
33          with:
34
35              username: root
36
37              host: dev.philipp-kraft.com
38
39              password: ${{ secrets.SSH_PASSWORD }}
40
41              script: 'cd /var/apm && ./server_deploy.sh && chown -R
42                  www-data.www-data /var/apm && chmod -R 755 /var/apm && chmod
43                  -R 777 /var/apm/storage'

```

Code 69: main.yml

Dieser Workflow setzt einen Ubuntu Server in der Cloud auf und baut dort die Assets zusammen, damit die Downtime auf dem Production Server möglichst gering ist. Danach wird eine SSH Verbindung zum Production Server aufgebaut und dort wird das Bash-Script `server_deploy.sh` ausgeführt. Gleichzeitig werden einige Berechtigungen angepasst.

In der Repository muss nun noch das SSH-Passwort und der Personal access token hinterlegt werden. Dies geschieht über **Settings ▶ Secrets**. Dort können nun über **New repository secrets** die Secrets hinterlegt werden.

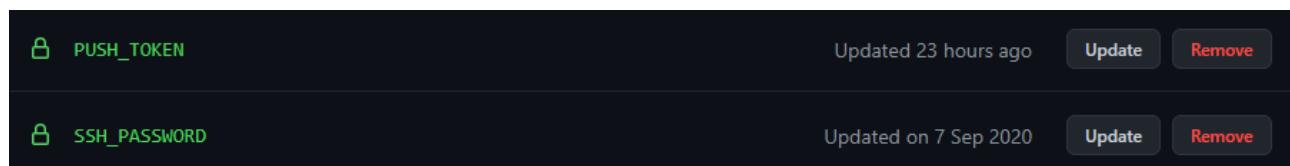


Abbildung 62: Action Secrets

```

deploy
succeeded 23 hours ago in 1m 10s

> ✓ Set up job
> ✓ Build appleboy/ssh-action@master
> ✓ Run actions/checkout@v2
> ✓ Set up Node
> ✓ Run npm install
> ✓ Run npm run production
> ✓ Commit built assets
> ✓ Deploy to production
> ✓ Post Run actions/checkout@v2
> ✓ Complete job

```

Abbildung 63: Github Action Übersicht

8.6 Grundlegender Aufbau Frontend

Da das Ziel ist, dass das Frontend des Webinterfaces möglichst Modular aufgebaut ist und so wenig Code wie möglich wiederholt wird. Ermöglicht wird dies durch das aufbauen der Seite mithilfe von Components.

8.6.1 Components

Components sind praktisch kleine Bausteine aus denen die komplette Seite aufgebaut ist. Components sind kein natives Feature von HTML/CSS oder PHP, diese Funktion wird von der Template Engine Blade bereitgestellt, deshalb werden diese auch oft Blade Components genannt.

8.6.1.1 Anonymous Components

Es gibt viele verschiedene Möglichkeiten Components zu erstellen und verschiedene Konventionen am, einfachsten sind aber die *Anonymous Components*, diese haben den Vorteil, dass diese in einer Datei verwaltet werden können und somit sehr einfach zu handhaben sind.

8.6.1.2 Components erstellen

Das Erstellen von einem Component wird nun anhand eines Buttons gezeigt. Da dieser als Anonymous Component angelegt wird muss dieser mit keiner Klasse assoziiert werden, es wird einfach

im Pfad resources\views\components ein Ordner mit dem Namen buttons angelegt und darin ein Blade File mit dem Namen primary.blade.php. Dort kann nun der gewünschte HTML Code platziert werden.

```
1 <button type="submit" class="inline-flex items-center px-4 py-2
  ↵ bg-apm-blue...>
2   {{ $slot }}
3 </button>
```

Code 70: primary.blade.php

Im Code 70 ist eine Variable mit dem Namen slot verwendet worden. Diese Variable wird später beim verwenden automatisch mit dem Inhalt zwischen dem HTML Element ersetzt.

8.6.1.3 Components verwenden

Es stellt sich nun die Frage wie man dieses erstelle Component nun verwendet. Die Blade Components verwenden den gleichen Syntax wie ein normales HTML Element, mit dem Unterschied dass ein x- vor dem Namen des Components angeführt werden muss. Da sich der vorhin erstellte Component in einem Ordner befindet muss das auch angegeben werden, dabei wird kein Slash wie üblich um einen Pfad anzugeben verwendet sondern ein Punkt, es muss auch keine Extensions angegeben werden.

```
1 <x-buttons.primary>Press me!</x-buttons.primary>
```

Code 71: Verwendung eines Button Components

8.6.1.4 Attribute übergeben

Auch wenn viele Components ohne Problem überall ohne Veränderung verwendet werden können, ist es gewünscht bei manchen Components beispielweise eine zusätzliche Klasse anzugeben um die Größe des Elements zu verändern. Erreicht wird das mit der attributes Variable im Blade File

des Components, da aber oft schon Attribute definiert sind ist es möglich mit der `merge` Methode die Attribute zusammenzuführen.

```
1 <button {{ $attributes->merge(['type' => 'submit'], 'class' => 'inline-flex
  ↵ items-center px-4 py-2 bg-apm-blue...') }}>
2   {{ $slot }}
3 </button>
```

Code 72: Modularer Button Component

Somit ist dieser Button Component nun vollständig Modular.

8.6.2 Layouts

Da auf den meisten Seiten des Webinterfaces fast das gleiche Layout beibehaltet ist es sinnvoll diesen Inhalt in ein Component umzuwandeln. Auch Layouts sind Components.

Es gibt im Webinterface folgende Layouts:

- **app.blade.php**

Layout für Allgemeine Seiten

- **admin.blade.php**

Layout für die Administrativen Seiten mit einer Sidebar und Page Header

- **display.blade.php**

Besonderes Layout für die Display Seiten

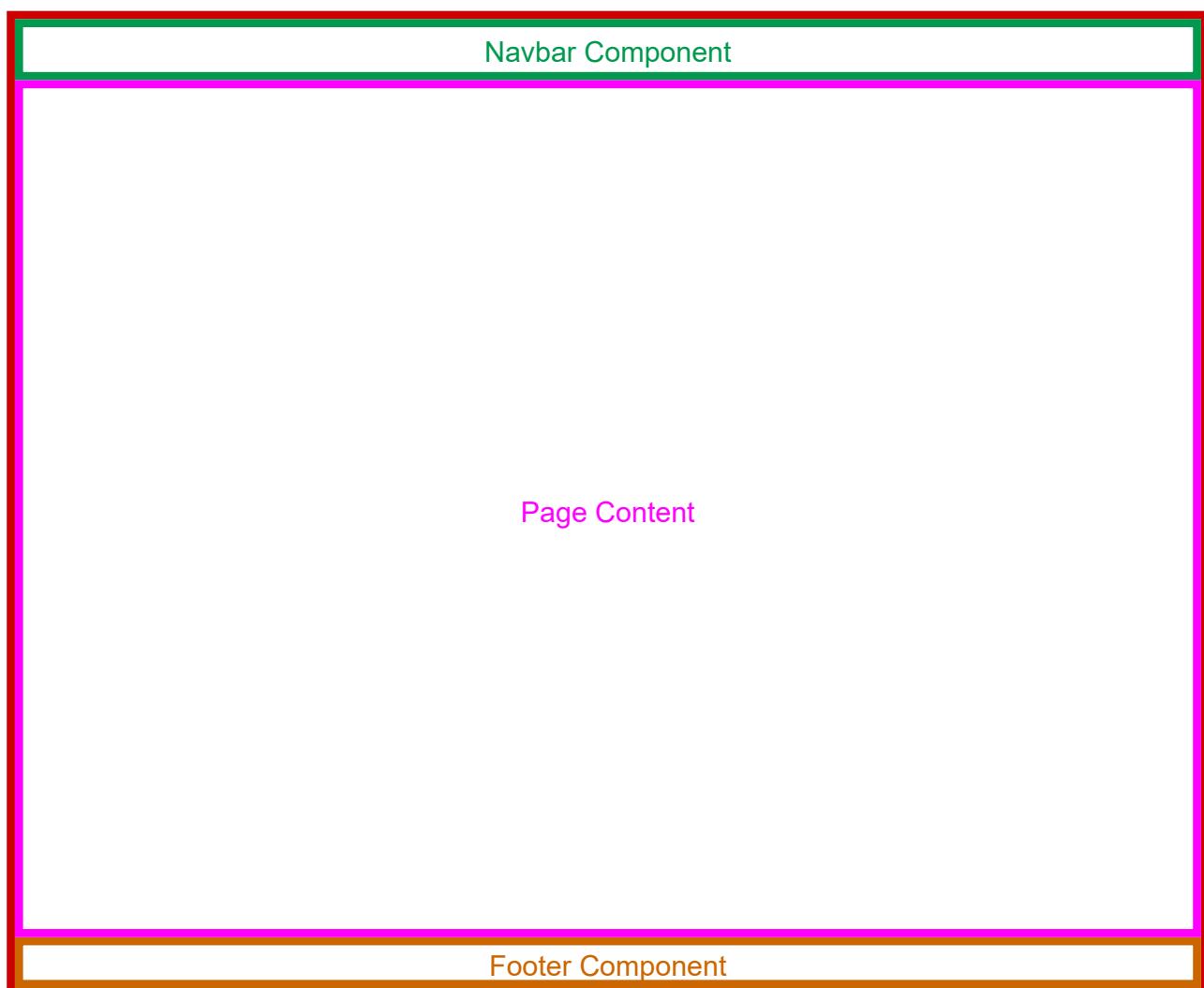


Abbildung 64: App Layout

Das App Layout ist das Standardmäßige Layout, es besteht aus drei Components, der Navigation Bar, dem Page Content und aus einem dynamische Footer.

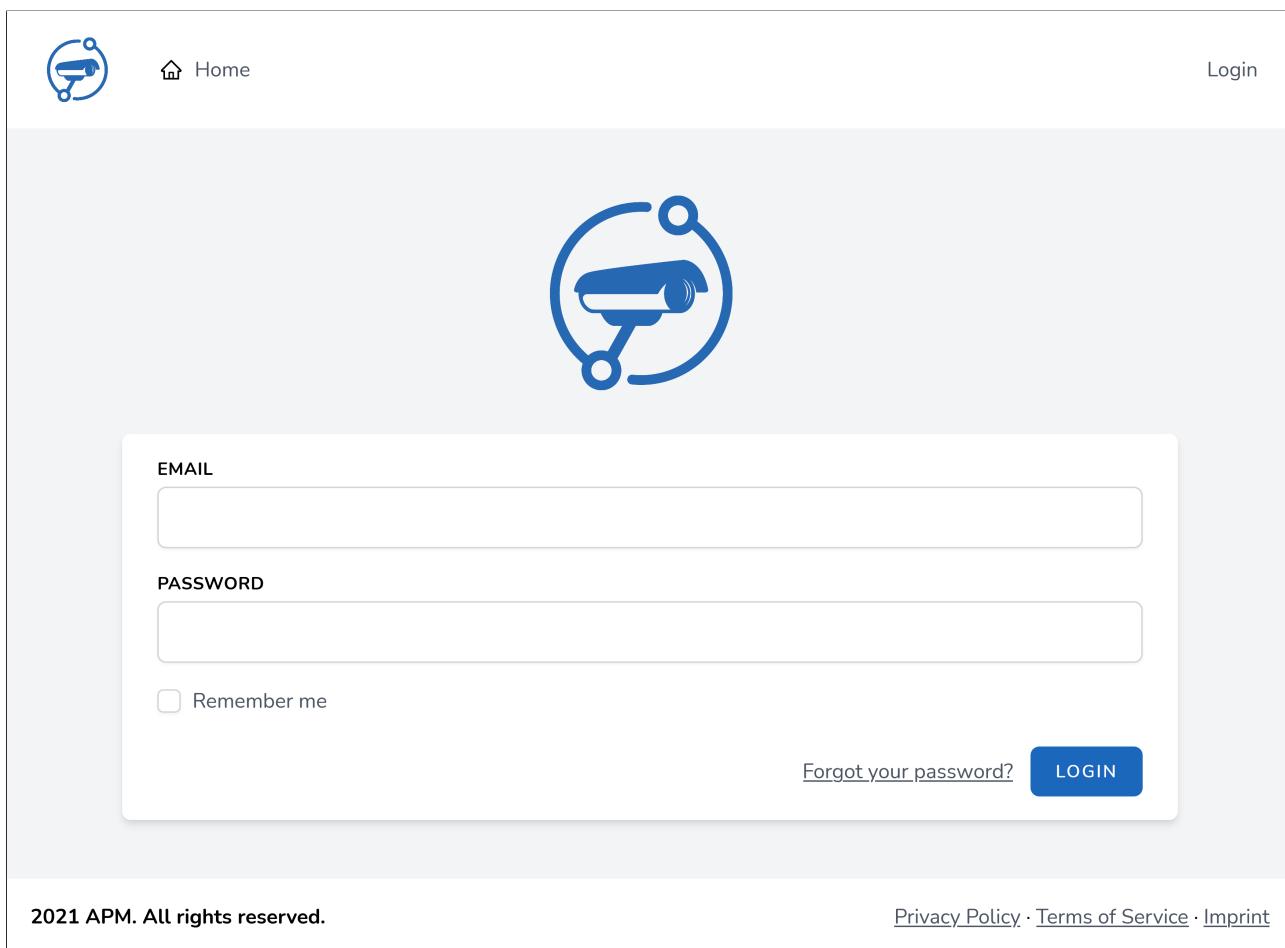


Abbildung 65: Login Seite mit App Layout

In der Abbildung 65 wird das App Layout verwendet. Dabei sind die drei Components erkennbar.

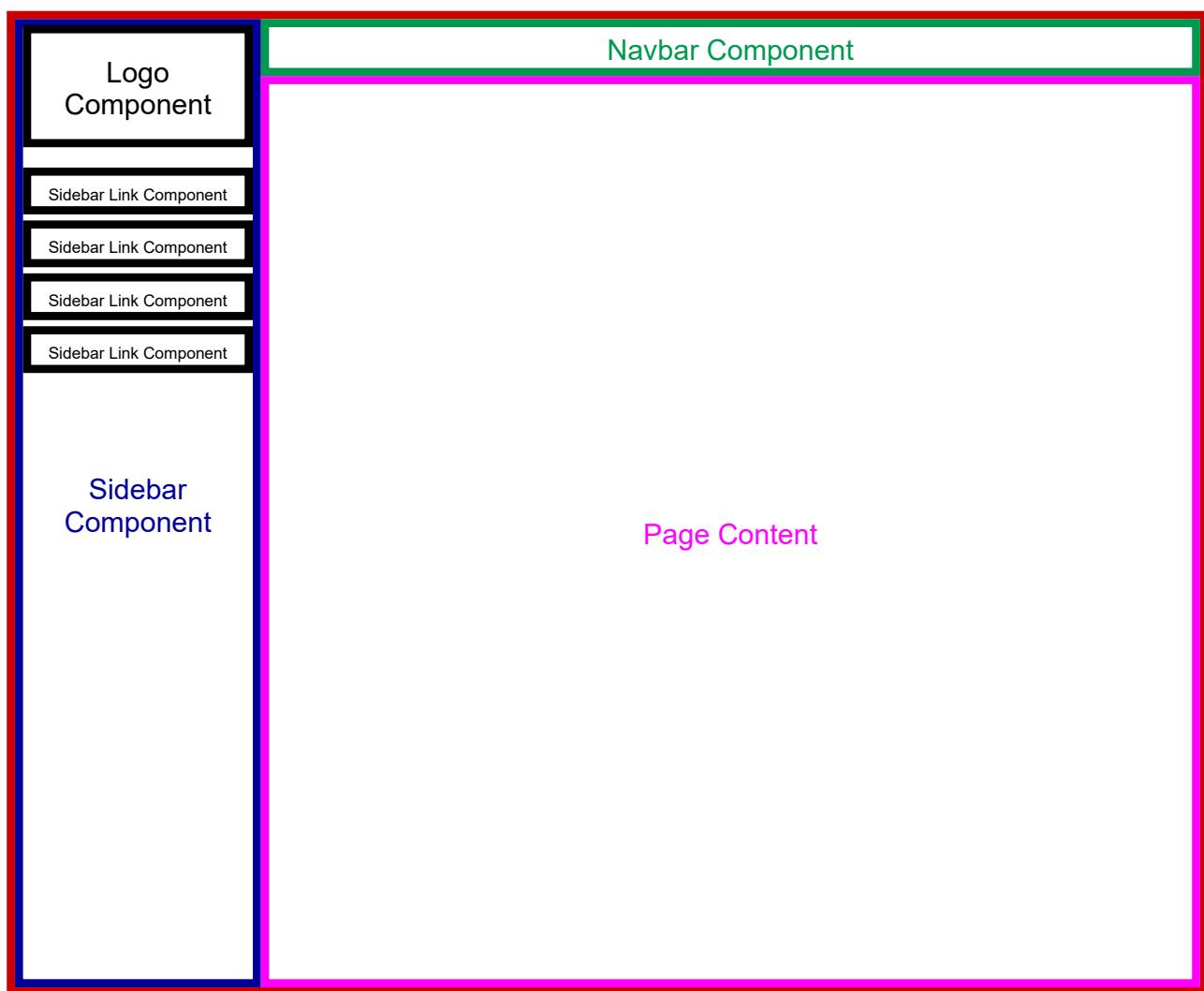


Abbildung 66: Admin Layout

Das Admin Layout besitzt eine zusätzliche Sidebar mit Links zu verschiedenen Administrativen Seiten, im Vergleich zum Standardmäßigen besitzt das Admin Layout keinen Footer und eine leicht veränderte Navbar²⁵.

²⁵Navigation Bar engl. für Navigationsleiste

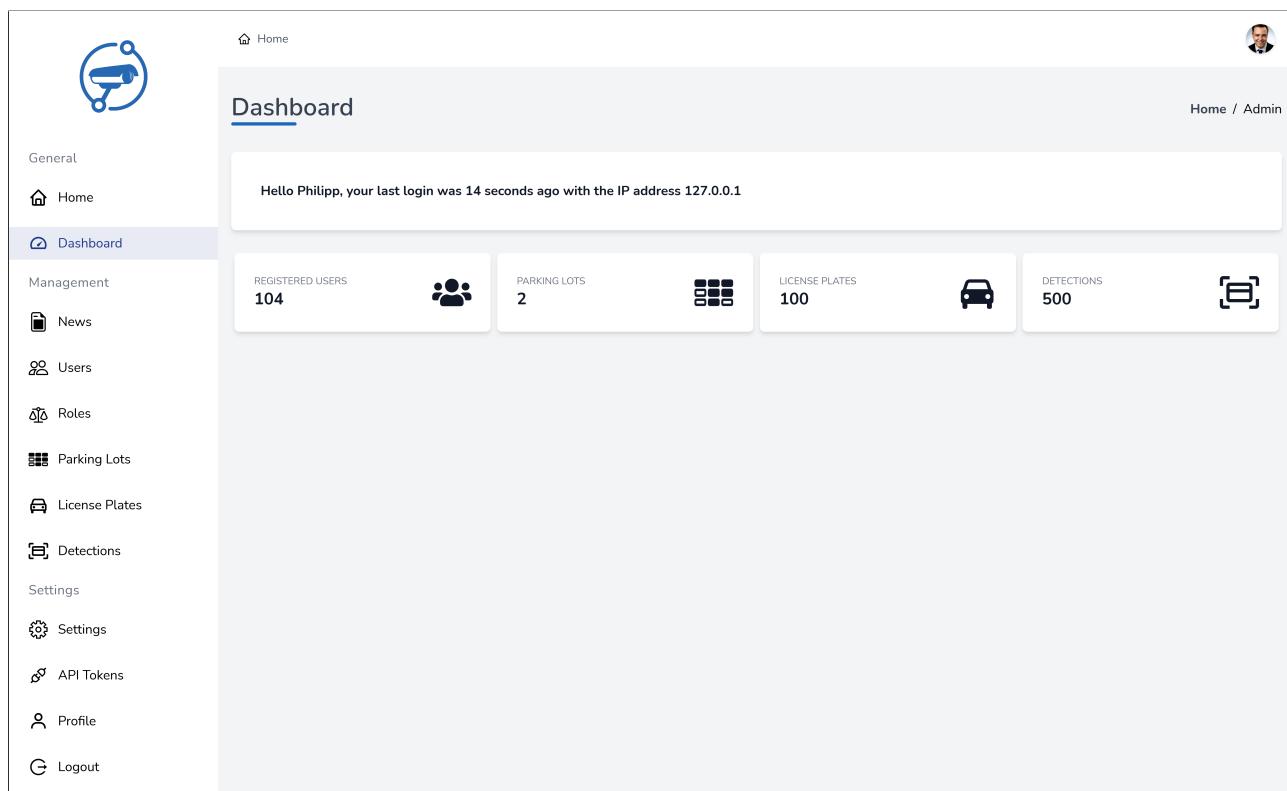


Abbildung 67: Dashboard mit Admin Layout

Im Dashboard wird das Admin Layout verwendet, somit befindet sich Links vom Page Content nun eine Sidebar mit weiteren Links zu anderen Administrativen Seiten.

Um nun die Funktion besser zu verstehen wird nun auf den Code einzelner Components genauer eingegangen.

8.6.3 Navbar

Die Navigationsleiste auch Navbar genannt ist ein zentrales Element der Webseite um verschiedene Seiten anzusteuern. Dabei veränderte sich diese dynamisch, je nachdem ob man eingeloggt ist oder ob ein bestimmtes Feature wie das Registrieren von neuen Benutzern deaktiviert ist.

```
1 @if (!(\Request::is('admin/*') || \Request::is('admin')))  
2     <a class="flex title-font font-medium items-center text-gray-900 mr-8">  
3         <x-application-logo class="max-h-12" />  
4     </a>  
5     @if (!Auth::guest())  
6         <a href="{{ route('dashboard') }}>{{ __('Dashboard') }}</a>  
7     @endif  
8     @endif
```

Code 73: Ausschnitt 1 navigation.blade.php

Der Ausschnitt 1 regelt das Aussehen der Navigationsleiste wenn ein Benutzer eingeloggt ist. Dabei wird kein Logo angezeigt wenn der Benutzer eine administrative Seite öffnet, da das Logo bereits in der Sidebar vorhanden ist. Ebenfalls wird auf nicht administrativen Seiten ein zurück zum Dashboard Link angezeigt.

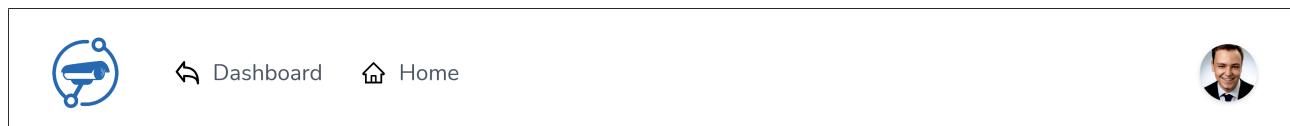


Abbildung 68: Navbar Variante 1



Abbildung 69: Navbar Variante 2

```

1  @if (Auth::guest())
2
3      <div class="flex items-center">
4          <a href="{{ route('login') }}>{{ __('Login') }}</a>
5          @if (Route::has('register'))
6              <a href="{{ route('register') }}>{{ __('Register') }}</a>
7          @endif
8      </div>
9  @endif

```

Code 74: Ausschnitt 2 navigation.blade.php

Im zweiten Ausschnitt geht es um die Login/Register Links, diese werden nur nicht eingeloggten Benutzern angezeigt. Ob der Register Link angezeigt wird ist abhängig von den getätigten Einstellungen in den Seiten Einstellungen.



Abbildung 70: Navbar Variante 3

8.6.4 Content Header

Um die Navigation zu erleichtern besitzt das Webinterface bei Seiten mit dem admin-Layout unter der Navigationsleiste einen Content Header. Der Content Header setzt sich aus dem Seiten Titel und den sogenannten Breadcrumbs²⁶ zusammen. Dabei befindet sich ganz links der aktuelle Seiten Titel, welcher ebenfalls im Browser Tab angezeigt wird und rechts die Brotkrümelnavigation. Die Breadcrumbs geben dabei dem Benutzer an in welcher Verzweigung er sich aktuell befindet.

Die Breadcrumbs werden aus der URL aufgebaut, dabei werden die einzelnen Segmente zerlegt und in einer Liste mit Hyperlinks zusammengeführt.

²⁶engl. für Brotkrümel, entspricht einer Navigation

```
1 <li><a href="/" class="text-gray-700 font-bold">Home</a></li>
2 <?php $link = ''; ?>
3 @for ($i = 1; $i <= count(Request::segments()); $i++)
4 @if (($i < count(Request::segments())) & ($i > 0))
5   <?php $link .= '/' . Request::segment($i); ?>
6   <li><span class="mx-2"/></span></li>
7   <li><a href="<?= $link ?>" class="text-gray-700 font-bold">{{
8     ucwords(str_replace('-', ' ', Request::segment($i))) }}</a></li>
9 @else
10  <li><span class="mx-2"/></span></li>
11  <li>{{ ucwords(str_replace('-', ' ', Request::segment($i))) }}</li>
12 @endif
13 @endfor
```

Code 75: Erstellung der Breadcrumbs

Edit License Plate

Home / Admin / Plates / 1 / Edit

Abbildung 71: Content Header

8.6.5 Session Alerts

Um einen Benutzer zu informieren, dass eine Anfrage erfolgreich ausgeführt wurde oder ob es einen Fehler gab werden *Session Alerts* verwendet.

Wird beispielsweise ein neues Kennzeichen im System angelegt, so wird der Benutzer nicht nur zu der Liste aller Kennzeichen geleitet, sondern es wird in der Session des Browsers eine Nachricht mitgespeichert.

```

1 <?php
2 return redirect(route('plates.index'))->with('success', __('The license
→ plate was successfully created!'));

```

Code 76: Controller mit success Nachricht

Um diese Nachricht nun anzuzeigen wurde ein Component entworfen welcher nur angezeigt wird, wenn eine solche Session Nachricht vorhanden ist. Neben einer *success* Nachricht gibt es auch noch eine *error* Nachricht, diese wird bei fehlerhaften Angaben in einem Formular verwendet.

```

1 @if (session('success'))
2 <div class="bg-green-200 px-6 py-4 mb-4 rounded-md text-lg flex
→ items-center justify-items-stretch">
3   <span class="text-green-800">{{ session('success') }}</span>
4 </div>
5 @endif

```

Code 77: Session Alert Component

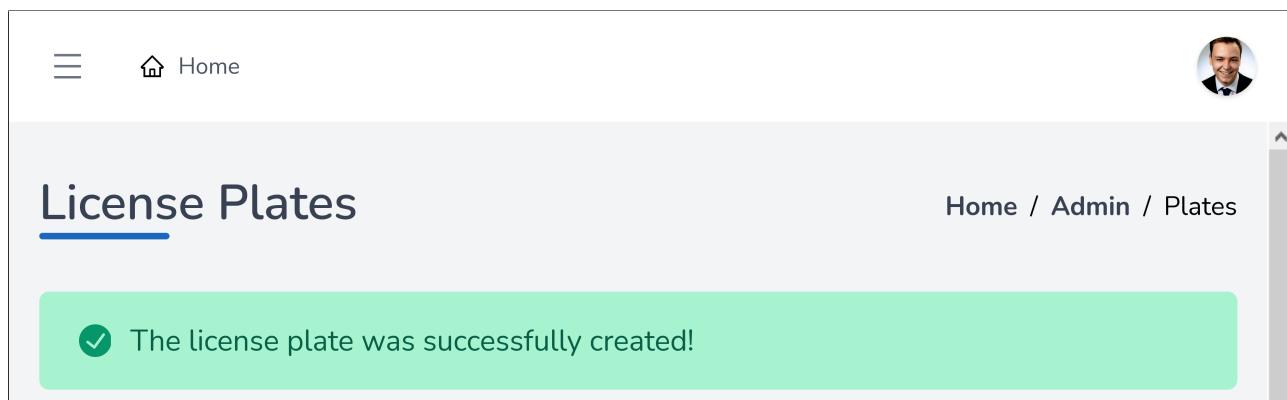


Abbildung 72: Success Session Alert

8.6.6 Sidebar

Die Sidebar ist das zentrale Element für administrative Benutzer des Webinterfaces. Über die Sidebar sind alle wichtigen Seiten auf einen Blick zu sehen und erreichen, dabei spielt es keine Rolle ob auf dem Smartphone oder auf dem Desktop, denn die Sidebar wurde so implementiert, dass sie auf kleineren Auflösungen eingeklappt wird und nur auf Wunsch mit einem das Hamburger-Menü-Symbol²⁷ in der Navigationsleiste geöffnet, somit wird wichtiger Platz gespart.

Für das dynamische Verhalten der Sidebar wird JavaScript verwendet. Es wird nicht Vanilla²⁸ JavaScript verwendet, sondern AlpineJS²⁹. Dabei folgt das Lightweight Framework einem ähnlichen Prinzip wie TailwindCSS nur eben für JavaScript.

Die Sidebar Einträge bestehen aus zwei Components, dem Sidebar Header Component und dem Sidebar Link Component. Der Sidebar Header Component ist da um die verschiedenen Links in verschiedene Sektionen aufzuteilen, damit die Übersichtlichkeit gewährt bleibt.

```
1 <x-sidebar-header>{{ __('General') }}</x-sidebar-header>
```

Code 78: Sidebar Header

Der Sidebar Link Component ist für die Links zuständig. Dieser besteht aus einem SVG-Icon und dem Namen der Seite. Die Route wird dem Component über :href übergeben und nicht mit href, da die URL mit einer Helper Methode angegeben wird und ansonsten von Blade nicht compiliert wird und als tatsächlicher Hyperlink angenommen wird.

²⁷Drei waagrechte Striche ähneln einem Hamburger

²⁸Bedeutet so viel wie ohne Zusätze

²⁹<https://github.com/alpinejs/alpine>

```
1 <x-sidebar-link :href="route('home')"  
2   ↳ :active="request()->routeIs('home')">  
3   <span class="iconify h-6 w-6" data-icon="bx:bx-home"  
4     ↳ data-inline="false"></span>  
5   <span class="mx-3">{{ __('Home') }}</span>  
6 </x-sidebar-link>
```

Code 79: Sidebar Link

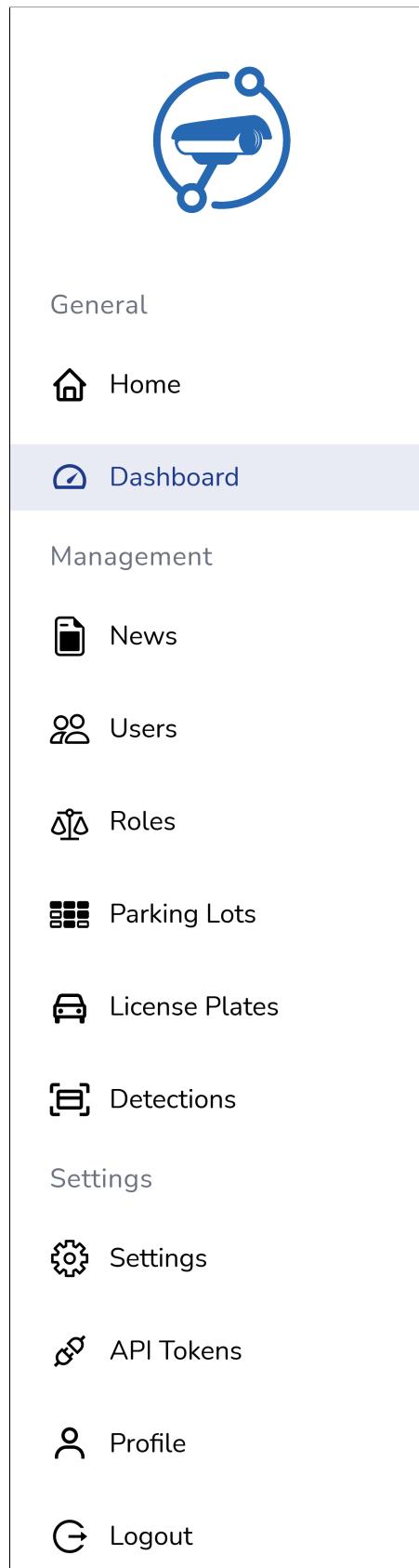


Abbildung 73: Sidebar Desktop

8.6.7 Footer

Der Footer ist dazu da um Informationen rund um Impressum, Datenschutzverweis, Geschäftsbedingungen und Urheberrechtshinweis darzustellen.

Grundsätzlich ist besitzt der Footer des Webinterfaces auf der linken Seite einen variablen Text der sich frei über die Seiten Einstellungen einstellen lässt und auf der rechten Seiten Verweise auf die Datenschutzbestimmungen, Allgemeine Geschäftsbedingungen und Impressum welche sich ebenfalls über die Seiten Einstellungen konfigurieren lassen. Befindet sich auf diesen Seiten kein Inhalt werden diese auch nicht im Footer angezeigt.

2021 APM. All rights reserved.

[Privacy Policy](#) · [Terms of Service](#) · [Imprint](#)

Abbildung 74: Footer

8.7 Funktionen

Das Webinterface besitzt eine große Anzahl an Features und Konfigurationsmöglichkeiten. In den folgenden Seiten werden alle Funktionen ausführlich mit deren technischen Funktion erklärt.

8.7.1 Dashboard

Das Dashboard ist praktisch die Startseite für eingeloggte Benutzer, als erstes wird dem Benutzer eine Sicherheitsnachricht über den letzten Login angezeigt. Dabei wird die Zeit seit dem letzten Login angegeben und die dabei verwendete IP-Adresse.

Um diese Daten zu speichern wird im `AuthenticatedSessionController` die `store` Methode ergänzt. Dabei wird nachdem sich der Benutzer erfolgreich Authentifiziert hat die IP-Adresse und der aktuelle Zeitpunkt in die Datenbank geschrieben.

```
1 <?php
2 $user = Auth::user();
3 $user->last_login_at = Carbon::now()->toDateTimeString();
4 $user->last_login_ip = $request->getClientIp();
5 $user->save();
```

Code 80: AuthenticatedSessionController.php Store Methode

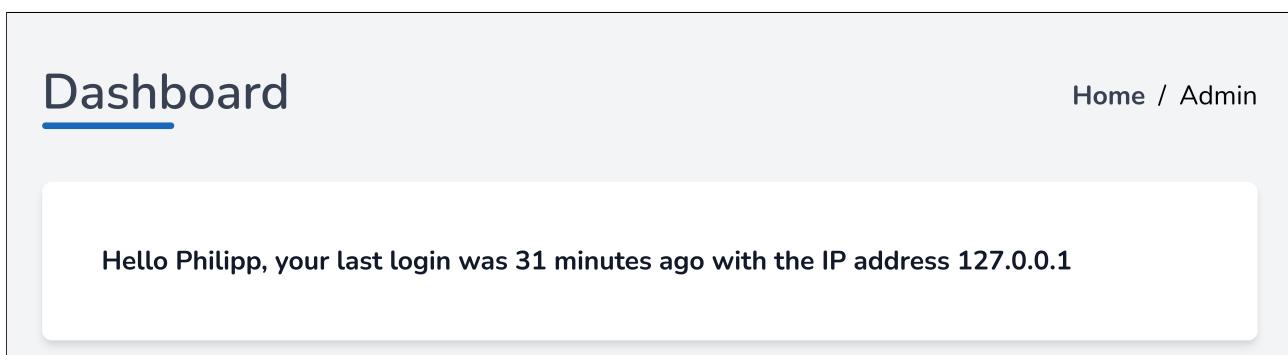


Abbildung 75: Dashboard Sicherheitsnachricht

Neben der Sicherheitsnachricht werden einige statistische Zahlen über das System dargestellt:

- Anzahl der registrierten Benutzer
- Anzahl der Parkplätze im System
- Anzahl der bekannten Kennzeichen
- Anzahl der erkannten Kennzeichens

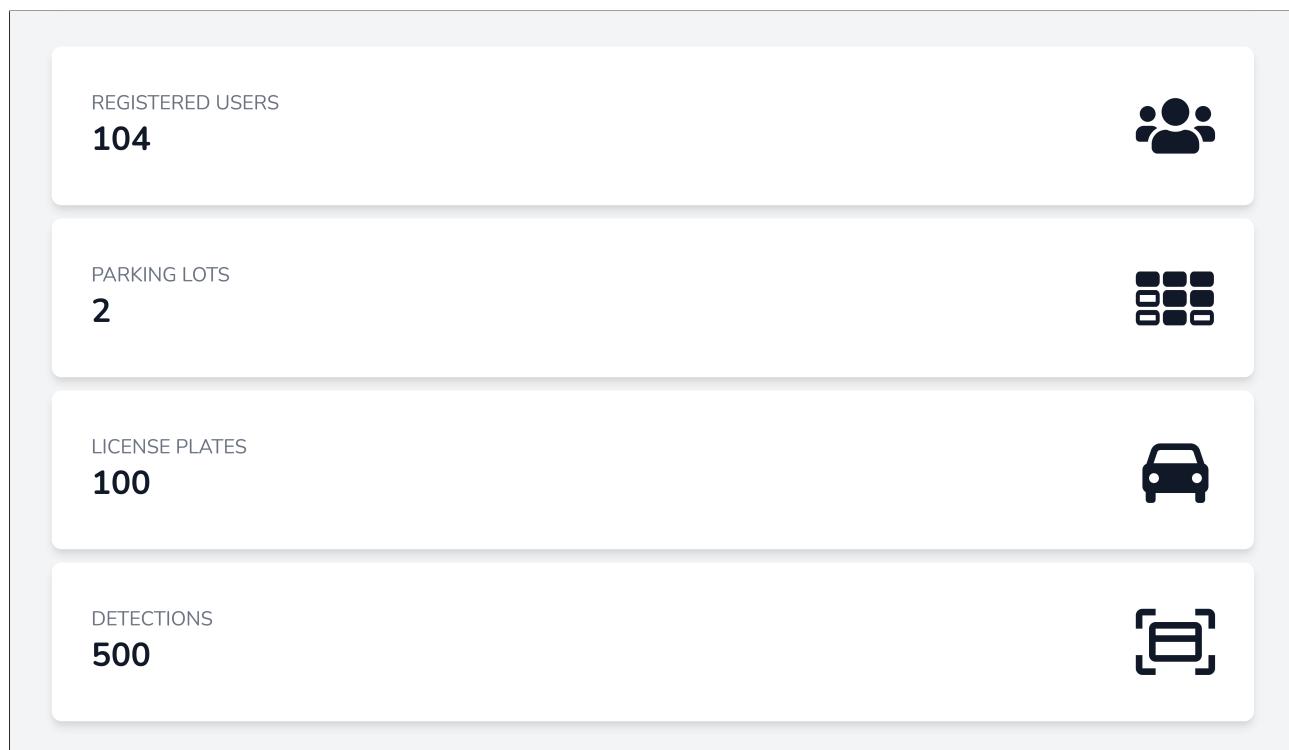


Abbildung 76: Dashboard Statistik

8.7.2 Benutzerverwaltung

Die Benutzerverwaltung ist ein zentrales Steuerelement des Webinterfaces und wichtig für die Zugriffskontrolle. Für die Benutzerverwaltung sind alle CRUD-Operationen³⁰ verfügbar.

³⁰Grundlegende Operationen: Create, Read, Update und Delete

Users
id (bigint)
first_name (string)
last_name (string)
email (string)
avatar (string)
email_verified_at (datetime)
password (string)
remember_token (string)
created_at (datetime)
updated_at (datetime)

Abbildung 77: Benutzer ER

8.7.2.1 Liste aller Benutzer

Die Startseite der Benutzerverwaltung ist eine Liste von allen Benutzern im System. Dort sind die wichtigsten Informationen direkt abzulesen wie Name, Rolle und Email.

Users						Home / Admin / Users
USERS						CREATE NEW USER
User	Roles	Email	Created	Updated	Actions	
 Philipp Kraft	Owner	Mail@Philipp-Kraft.com	1 day ago	1 hour ago	  	
 Dennis Köb	Owner	Dennis.Koeb@gmail.com	1 day ago	1 day ago	  	
 Samuel Bleiner	Owner	Bleiner.Samuel@gmail.com	1 day ago	1 day ago	  	
 Christoph Stüttler	Owner	Christoph.Stuettler@htl-rankweil.at	1 day ago	1 day ago	  	

Abbildung 78: Liste aller Benutzer

8.7.2.2 Benutzer erstellen/editieren

Neue Benutzer können mit dem Button rechts oben über Create new user angelegt werden. Einzelne Benutzer besitzen drei mögliche Aktionen in der letzten Spalte, dabei gibt es Betrachten (Blau), Editieren (Gelb) und Löschen (Rot). Es können folgende Informationen eines Benutzers konfiguriert werden:

- Profilbild
- Vorname
- Nachname
- Email
- Passwort
- Rollen

Edit User

Home / Admin / Users / 1 / Edit



Philipp Kraft

General Settings

FIRST NAME

Philipp

LAST NAME

Kraft

EMAIL

Mail@Philipp-Kraft.com

Account Security

NEW PASSWORD

CONFIRM NEW PASSWORD

Roles

Owner
 Editor

Administrator
 Viewer

[DELETE USER](#)

[SAVE USER](#)

Abbildung 79: Benutzer editieren

8.7.2.3 Profilbilder

Jeder Benutzer hat die Möglichkeit ein Profilbild zu besitzen und dies wird überall verwendet wo ein Benutzer erwähnt wird.

Um ein Bild hochzuladen muss beim editieren/erstellen auf das Profilbild bzw. auf das Standardprofilbild geklickt werden, es öffnet sich nun ein Upload-Dialog des Betriebssystems und erlaubt es ein Bild auszuwählen.

```

1 <div class="flex justify-center items-center mt-6">
2   <label for="avatar">
3     <input type="file" name="avatar" id="avatar" style="display:none;" />
4     
6   </label>
7 </div>

```

Code 81: Profilbild Upload

Bevor das Bild nun auf dem Server gespeichert wird ist es notwendig es mit der PHP Image Library **Intervention**³¹ das Bild auf die passende Größe zu skalieren. Nach diesem Schritt wird das Bild unter dem Pfad /uploads/avatars/ mit einem zufälligem Namen abgespeichert und mit dem Benutzer assoziiert.

```

1 <?php
2 if ($request->hasFile('avatar')) {
3   $avatar = $request->file('avatar');
4   $filename = time() . '.' . $avatar->getClientOriginalExtension();
5
6   Image::make($avatar)->resize(600, 600, function ($constraint) {
7     $constraint->aspectRatio();
8   })->save(public_path('/uploads/avatars/' . $filename));
9
10  $user->avatar = $filename;
11 }

```

Code 82: UserController.php Profilbild Upload

³¹<http://image.intervention.io/>

8.7.3 Rollen- und Rechteverwaltung

Die Rollen- und Rechteverwaltung ist eng mit der Benutzerverwaltung verbunden. Dabei ist das System so aufgebaut, dass es verschiedene Rechte für bestimmte Anfragen gibt (z.B. Benutzer editieren), diese Vielzahl an Rechten können verschiedenen Gruppen zugewiesen werden. Schliessendlich kann man diese Gruppen einzelnen Benutzern zuweisen.

Es gibt nun drei relevante Modelle (User, Role und Permission), dabei besteht zwischen dem User und dem Role Model eine m zu n Beziehung. Ebenfalls besteht zwischen dem Role Model und dem Permission Model eine m zu n Beziehung.

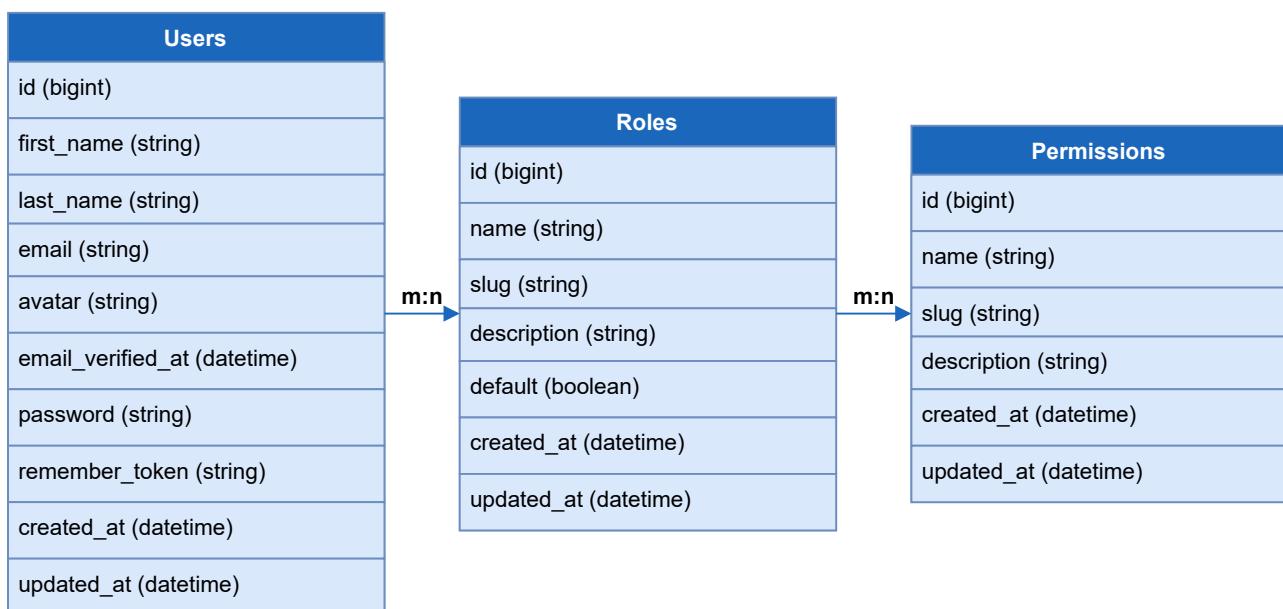


Abbildung 80: Benutzer, Rollen und Rechte ER

8.7.3.1 Liste der verfügbaren Rechte

Die Rechte von einzelnen Funktionen sind ähnlich aufgebaut:

- **Index <Model>**: Liste aller einzelnen Elemente der Model
- **Create <Model>**: Formular für das erstellen eines neuen Elements der Model
- **Store <Model>**: Neues Element der Model in die Datenbank speichern

- **Show <Model>**: Einzelnes Element der Model anzeigen
- **Edit <Model>**: Formular für das editieren eines Elements der Model
- **Update <Model>**: Vorhandenes Element der Model in der Datenbank aktualisieren
- **Destroy <Model>**: Element der Model aus der Datenbank löschen

Das Webinterface besitzt 43 verschiedene Rechte.

Name	Slug	Beschreibung
Index Post	index-post	Liste aller Neuigkeit anzeigen
Create Post	create-post	Neuigkeit erstellen
Store Post	store-post	Neuigkeit speichern
Show Post	show-post	Neuigkeit anzeigen
Edit Post	edit-post	Neuigkeit editieren
Update Post	update-post	Neuigkeit aktualisieren
Destroy Post	destroy-post	Neuigkeit löschen
Index User	index-user	Liste aller Benutzer anzeigen
Create User	create-user	Benutzer erstellen
Store User	store-user	Benutzer speichern
Show User	show-user	Benutzer anzeigen
Edit User	edit-user	Benutzer editieren
Update User	update-user	Benutzer aktualisieren
Destroy User	destroy-user	Benutzer löschen
Index License Plate	index-license-plate	Liste aller Kennzeichen anzeigen
Create License Plate	create-license-plate	Kennzeichen erstellen
Store License Plate	store-license-plate	Kennzeichen speichern
Show License Plate	show-license-plate	Kennzeichen anzeigen
Edit License Plate	edit-license-plate	Kennzeichen editieren
Update License Plate	update-license-plate	Kennzeichen aktualisieren
Destroy License Plate	destroy-license-plate	Kennzeichen löschen
Index Detection	index-detection	Liste aller Erkennungen anzeigen

Index Parking Lot	index-parking-lot	Liste aller Parkplätze anzeigen
Create Parking Lot	create-parking-lot	Parkplätze erstellen
Store Parking Lot	store-parking-lot	Parkplätze speichern
Edit Parking Lot	edit-parking-lot	Parkplätze editieren
Update Parking Lot	update-parking-lot	Parkplätze aktualisieren
Destroy Parking Lot	destroy-parking-lot	Parkplätze löschen
Store Parking Spot	store-parking-spot	Parkplätze speichern
Destroy Parking Spot	destroy-parking-spot	Parkplätze löschen
Index Role	index-role	Liste aller Rollen anzeigen
Create Role	create-role	Rolle erstellen
Store Role	store-role	Rolle speichern
Show Role	show-role	Rolle anzeigen
Edit Role	edit-role	Rolle editieren
Update Role	update-role	Rolle aktualisieren
Destroy Role	destroy-role	Rolle löschen
Index Setting	index-setting	Liste aller Einstellungen anzeigen
Update Setting	update-setting	Einstellungen aktualisieren
Index Token	index-token	Liste aller API Schlüssel anzeigen
Store Token	store-token	API Token speichern
Destroy Token	destroy-token	API Token löschen

Tabelle 4: Verfügbare Rechte

8.7.3.2 Vergabe von Rechten und Rollen mit Slugs

Da es bei der großen Anzahl von verschiedenen Rechten kompliziert wäre die Rechte und Rollen mit einer ID zu vergeben wird ein Slug³² verwendet.

³²Benutzerfreundlicher Name

```
1 <?php
2 public function givePermissionBySlug($slugs)
3 {
4     if (!isset($slugs))
5         return;
6
7     foreach ($slugs as $slug) {
8         $permission = Permission::where('slug', $slug)->first();
9         $this->permissions()->attach($permission);
10    }
11 }
```

Code 83: Permission Vergabe Methode

Mit der givePermissionBySlug Methode ist es nun möglich Rollen eine Vielzahl von Rechten zuzuordnen, dabei muss nur ein Array mit den Slugs der einzelnen Rechte übergeben werden.

Im Codeausschnitt 84 vom RoleSeeder wird eine neue Rolle mit dem Namen **Administrator** und dem Slug **admin** erstellt, gleichzeitig werden einige Rechte dieser Gruppe zugewiesen.

```
1 <?php
2 Role::create(['name' => 'Administrator', 'slug' => 'admin'])
3 ->givePermissionBySlug([
4     'index-user', 'create-user', 'store-user', 'show-user', 'edit-user'
5 ]);
```

Code 84: Rolle erstellen und Rechte zuweisen

Eine ähnliche Methode wird verwendet um einem Benutzer eine oder mehrere Rollen zuzuordnen.

8.7.3.3 Authentisierung

Für das Überprüfen ob ein Benutzer die korrekten Rechte besitzt werden Laravel Policies verwendet.

Es wird für jedes Modell eine Policy erstellt und für jedes Recht eine Methode. Diese Methoden werden nun innerhalb eines Controllers aufgerufen, liefert die Methode `true` wird die Anfrage fortgesetzt, liefert die Methode `false` erhält der Benutzer einen 403 Forbidden HTTP Error.

```
1 <?php
2 public function index(User $user)
3 {
4     $permission = Permission::where('slug', 'index-user')->first();
5     return $user->hasRole($permission->roles);
6 }
```

Code 85: UserPolicy Index

```
1 <?php
2 public function index()
3 {
4     $this->authorize('index', User::class);
5     . . .
6 }
```

Code 86: UserController Index

8.7.3.4 Rolle erstellen

Eine neue Rolle kann über Roles ► Create new role erstellt werden. Dabei können folgende Informationen konfiguriert werden:

- Rollename
- Slug
- Beschreibung
- Rechte

- Standardrolle

Die gewünschten Rechte werden mithilfe von Checkboxen ausgewählt. Die zusätzliche Option Standardrolle erlaubt es diese Rolle automatisch neue registrierten Benutzern zuzuweisen.

Create Role

Home / Admin / Roles / Create

New role

General Settings

NAME	SLUG
<input type="text" value="Tester"/>	<input type="text" value="tester"/>
DESCRIPTION	
<input type="text" value="Eine Rolle für Tester"/>	

Permissions

<input checked="" type="checkbox"/> Index Post <input checked="" type="checkbox"/> Store Post <input checked="" type="checkbox"/> Edit Post <input type="checkbox"/> Destroy Post <input type="checkbox"/> Create User <input type="checkbox"/> Show User <input type="checkbox"/> Update User <input type="checkbox"/> Index License Plate <input type="checkbox"/> Store License Plate <input type="checkbox"/> Edit License Plate <input type="checkbox"/> Destroy License Plate <input type="checkbox"/> Index Parking Lot <input type="checkbox"/> Store Parking Lot <input type="checkbox"/> Edit Parking Lot <input type="checkbox"/> Destroy Parking Lot <input type="checkbox"/> Destroy Parking Spot <input type="checkbox"/> Create Role <input type="checkbox"/> Show Role <input type="checkbox"/> Update Role <input type="checkbox"/> Index Setting <input type="checkbox"/> Index Token <input type="checkbox"/> Destroy Token	<input checked="" type="checkbox"/> Create Post <input checked="" type="checkbox"/> Show Post <input checked="" type="checkbox"/> Update Post <input type="checkbox"/> Index User <input type="checkbox"/> Store User <input type="checkbox"/> Edit User <input type="checkbox"/> Destroy User <input type="checkbox"/> Create License Plate <input type="checkbox"/> Show License Plate <input type="checkbox"/> Update License Plate <input type="checkbox"/> Index Detection <input type="checkbox"/> Create Parking Lot <input type="checkbox"/> Show Parking Lot <input type="checkbox"/> Update Parking Lot <input type="checkbox"/> Store Parking Spot <input type="checkbox"/> Index Role <input type="checkbox"/> Store Role <input type="checkbox"/> Edit Role <input type="checkbox"/> Destroy Role <input type="checkbox"/> Update Setting <input type="checkbox"/> Store Token
---	---

Other Settings

DEFAULT ROLE

Add new users automatically to this role

Abbildung 81: Benutzer erstellen

8.7.4 Neuigkeiten

Das Neuigkeiten Feature ist eine Möglichkeit um Besucher des Webinterfaces rasch über eine wichtige Neuigkeiten zu informieren. Dabei ist das System ähnlich wie ein Blog aufgebaut und zeigt die einzelnen News auf der Startseite der Webseite an. Es sind alle CRUD-Operationen verfügbar.

Edit News

[Home](#) / [Admin](#) / [Posts](#) / [1](#) / [Edit](#)

Post Number 1

General Settings

TITLE

Post Number 1

CONTENT



Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

body p

Other Settings

PUBLISHED

Display this news on the home site

ANONYMOUS

Display this news with no author name

[DELETE NEWS](#)

[SAVE NEWS](#)

Abbildung 82: Neuigkeit editieren

Neben dem Titel und Inhalt gibt es noch zwei besondere Einstellungen:

- **Published**

Um eine Neuigkeit auf der Startseite für Besucher anzuzeigen muss diese Einstellung aktiviert

sein.

- **Anonymous**

Ist es nicht erwünscht, dass der Name und das Profilbild des Autors auf der Startseite angezeigt wird, kann dies mit dieser Option deaktiviert werden.

Für das editieren des Inhalts wird der Quelloffene WYSIWYG-HTML-Editor CKEditor 5³³ verwendet.

Die Startseite besitzt einen modifizierten Content Header mit dem akutellen Datum und Uhrzeit.

The screenshot shows the homepage of the HTBLuVA Rankweil website. At the top, there is a header with the school's logo and name. Below the header, the word "NEWS" is prominently displayed. Two news items are listed:

- Post Number 1** (30. March 2021) by Philipp Kraft: The post content is a placeholder text in Latin (Lorem ipsum).
- Post Number 2** (30. March 2021) by Anonymous user: This post also contains placeholder Latin text.

Abbildung 83: Home Neuigkeiten

8.7.5 Parkplatzverwaltung

Ein Benutzer kann beliebig viele Parkplätze im Webinterface anlegen, ein Parkplatz besteht dabei aus einem Namen und einer Beschreibung, ein Parkplatz kann ebenfalls ein Bild hinzugefügt werden ähnlich wie bei den Benutzern. In der Abbildung 84 ist zu erkennen, dass ein Parkplatz eine 1:n Beziehung zu den Parklücken (Parking Spots), sowie zu den Erkennungen (Detections).

³³<https://ckeditor.com>



Abbildung 84: Parking Lots, Parking Spots und Detections ER

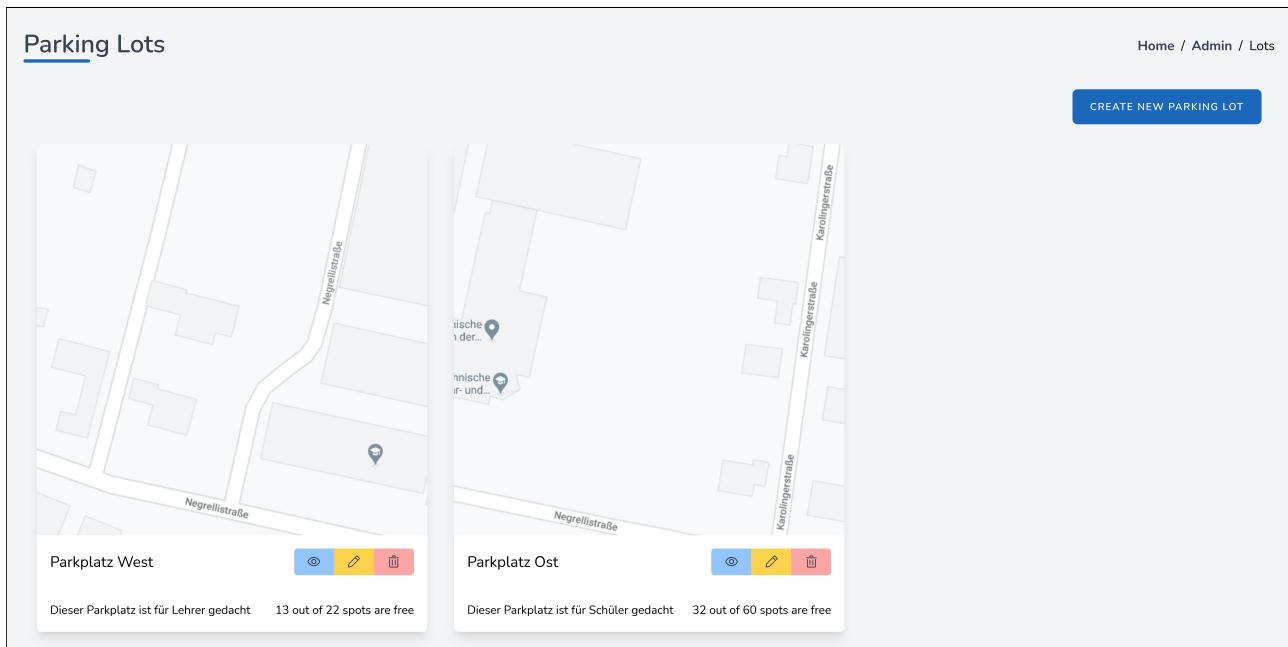


Abbildung 85: Übersicht aller Parkplätze

8.7.5.1 Parklücken

Eine Parklücke besteht aus einer Adresse und dem Status. Die Adresse ist eine von der Hardware gesetzte Zahl um die einzelnen Parklücken mit der API zu identifizieren und anzusteuern. Der Status ist dafür da um anzuzeigen ob eine Parklücke besetzt oder frei ist.

Show Parking Lot

Home / Admin / Lots / 3

PARKING SPOTS					
ID	ADDRESS	STATUS	UPDATED	ACTIONS	
3	100	Occupied	2021-02-10 13:41:36		
4	110	Free	2021-02-02 12:45:11		
5	120	Free	2021-02-08 02:20:36		
6	130	Occupied	2021-02-08 10:18:43		

DETECTIONS						
LICENSE PLATE	PARKING LOT	NEW PARKING STATUS	ALLOWANCE STATUS	ASSOCIATED USER	DETECTED	
B509DYY	No image available	Parking Space 1	Outside	Disallowed	Kristofer Bayer	2021-03-23 02:10:57
FK802HT	No image available	Parking Space 1	Outside	Allowed	Adonis Runolfsdottir	2021-03-16 05:27:25
FK12BR2I	No image available	Parking Space 1	Outside	Disallowed	Darrel Walsh	2021-03-03 06:09:57

Abbildung 86: Parklücken und Erkennungen eines Parkplatzes

8.7.6 Seiten Einstellungen

In den Seiten Einstellungen ist es möglich als Benutzer bestimmte Aspekte des Webinterfaces zu konfigurieren dazu zählt:

- Seiten Titel (key: `site_title`)
- Standardsprache (key: `default_language`)
- Seiten Logo
- Erlauben, dass sich neue Benutzer registrieren (key: `allow_new_users`)
- Footer Einstellungen (key: `footer_content`)
- Inhalt der Datenschutzbestimmungsseite (key: `privacy_policy`)
- Inhalt der Allgemeinen Geschäftsbedingungen (key: `terms_of_service`)
- Inhalt des Impressums (key: `imprint`)

8.7.6.1 Funktion der Einstellungen

Settings
id (bigint)
key (string)
value (text)

Abbildung 87: Einstellunge Tabelle

Die Einstellungen werden in der Datenbank gespeichert, in der Datenbank besteht eine Einstellung aus einem key und einer value. key ist dabei vorgegeben und die value kann frei durch das GUI konfiguriert werden.

Damit der Browser des Benutzers über die aktuellen Einstellungen bescheid weiß wird mithilfe eines Service Providers die in der Datenbank vorhandenen Einstellungen in die Laravel Konfiguration übernommen.

```

1 <?php
2
3 public function boot()
4 {
5     if (Schema::hasTable('settings')) {
6         config()->set('settings', \App\Models\Setting::pluck('value',
7             'key')->all());
8     }
9 }
```

Code 87: SettingsServiceProvider

Nun ist es möglich im Projekt überall mit der config() Helper Methode die Seiten Einstellungen abzurufen. Somit ist es beispielsweise möglich den Seiten Titel des Webinterfaces in den Layouts anzugeben.

```
1 <title>{{ config('settings.site_title') }}</title>
```

Code 88: Seiten Titel Konfiguration

8.7.7 Profil

Das Profil ist dazu, da um seine eigenen Benutzerinformationen zu betrachten und zu bearbeiten. Es sind die gleichen Einstellungen wie in der Benutzerverwaltung einstellbar, jedoch nur über seinen eigenen Account. Eine zusätzliche Einstellung stellt die UI-Sprache dar, dort kann jeder Benutzer seine gewünschte UI-Sprache konfigurieren.

Abbildung 88: Profil

Erreichbar ist das Profil über die Navigationsleiste oben rechts mit einem Klick auf das Profilbild, es öffnet sich ein Dropdown Menü mit dem Eintrag Profil.

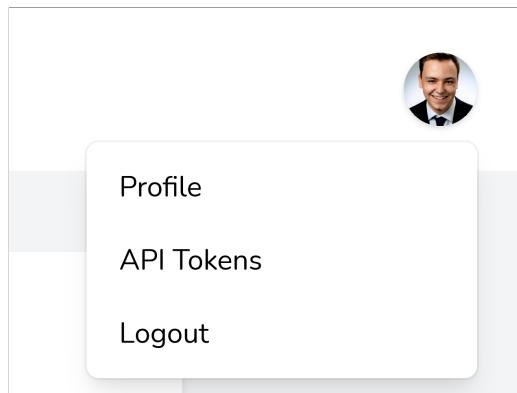


Abbildung 89: Navigationsleiste Dropdown

8.7.8 Lokalisierung

Lokalisierung ist sehr ähnlich zur Übersetzung, aber anstatt einer wörtlichen Übersetzung wird die Sprache und das Zielpublikum angepasst. Gegenbenfalls werden auch Maßeinheiten, Datums- und Uhrzeitformate angepasst.

Das Webinterface besitzt die Möglichkeit zwischen Deutsch und Englisch auszuwählen, hierbei wurde das Webinterface zuerst in Englisch verfasst und dann in das Deutsche lokalisiert.

8.7.8.1 Funktion der Sprachauswahl

Wie bereits in der Sektion 8.7.7 erklärt kann ein Benutzer in den Profileinstellungen sich eine Sprache je nach Bedürfnis auswählen, dabei wird diese Sprache nur lokal bei diesem Benutzer angepasst.

Bei der Sprachauswahl wird ein `select` Element verwendet, wenn ein Benutzer nun eine andere Option auswählt wird dieser weitergeleitet, beispielsweise bei einem Klick auf English wird dieser auf `/locale/en` weitergeleitet.

```

1  <?php
2
3  <select name="language" onChange="location = this.value;">
4
5  <option {{ Session::get('locale') == 'en' ? 'selected' : '' }}>
6    English</option>
7
8  <option {{ Session::get('locale') == 'de' ? 'selected' : '' }}>
9    Deutsch</option>
</select>

```

Code 89: Spracheauswahl

Beim Aufruf dieser Route wird in die Session des Benutzers die gewünschte Sprache geschrieben und der Benutzer wird direkt wieder zu der Ursprungsseite zurück geschickt. Diese Vorgang geht so schnell, dass ein Benutzer nicht bemerkt, dass die Seite gewechselt wurde.

```

1  <?php
2
3  Route::get('locale/{locale}', function ($locale) {
4
5    Session::put('locale', $locale);
6
7    return redirect()->back();
8
9 })->name('locale');

```

Code 90: Spracheauswahl Route

Nun steht in der Session des Benutzers die gewünschte Sprache, diese muss nun angewandt werden. Für das Anwenden der Sprache wird eine Middleware verwendet. Die Localization Middleware wird bei jedem Aufruf einer Route ausgeführt, dabei wird geprüft ob der Benutzer den Eintrag locale besitz, welcher vorhin durch den Service Provider gesetzt wurde und dann wird die gewünschte Sprache gesetzt.

```
1 <?php
2 if (\Session::has('locale')) {
3     \App::setlocale(\Session::get('locale'));
4 } else {
5     \App::setlocale(config('settings.default_language'));
6     \Session::put('locale', config('settings.default_language'));
7 }
```

Code 91: Localization Middleware

Besitzt der Benutzer keine locale Variable in seiner Session bedeutet, dass dieser noch keine Sprache festgelegt hat, somit wird automatisch die ausgewählte Standardsprache aus den Seiten Einstellungen ausgewählt.

8.8 API

8.8.1 Einleitung

Application programming interface (API) ist der Teil des Webinterfaces der mit der Hardware kommuniziert. Diese Programmierschnittstelle ist sehr einfach zu verwenden und wird mit Laravel Sanctum abgesichert. Die API ermöglicht ein bearbeiten von lesen von Daten aus dem Webinterface mithilfe von HTTP-Anfragen.

8.8.2 Autorisierung mit Bearer Token

Wie bereits besprochen erfolgt die Autorisierung der HTTP-Anfragen mit Laravel Sanctum. Sanctum verwendet dabei einen sogenannten Bearer Token. Dieser Bearer Token wird bei Anfragen einfach in der Anfrage im HTTP-Header mitgeschickt und der Server gleicht dies mit den verschlüsselten Token in der Datenbank ab erlaubt oder verweigert den Zugriff.

8.8.3 API Tokens erstellen

Über das Webinterface kann entweder über die Sidebar oder über das Dropdown Menü in der Navigationsleiste das Menü *API Tokens* erreicht werden. Dort können nun neue API Tokens erstellt werden, zur späteren Identifikation kann diesem API Token mit einem Namen versehen werden, dieser Name spielt aber ansonsten keine Rolle.

The screenshot shows a web interface for managing API tokens. At the top, there's a navigation bar with 'Home / Admin / Tokens'. Below it, a section titled 'Create API Token' contains a 'TOKEN NAME' input field and a 'CREATE NEW TOKEN' button. Below this, another section titled 'Active API Tokens' lists a single token named 'Mein API Token' with the note 'LAST USED: Never' and a red 'DELETE TOKEN' button.

Abbildung 90: API Token

Beim erstellen des API Tokens öffnet sich einmalig ein Popup mit dem Bearer Token, dieser muss nun abgespeichert werden, denn der Token wird aus Sicherheitsgründen nur einmalig angezeigt.

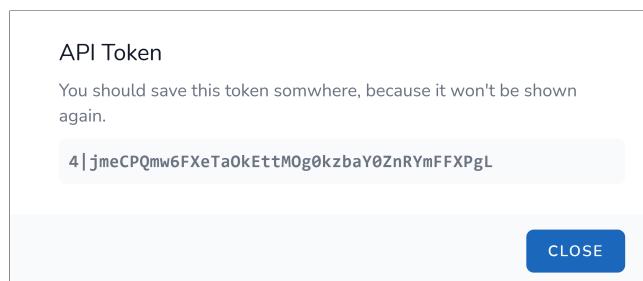


Abbildung 91: API Token Popup

8.8.4 Erkennungen

8.8.4.1 GET /api/v1/detections

Erkennungen GET	
Pfad	/api/v1/detections
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
None	
Query Parameter	
None	

Tabelle 5: GET /api/v1/detections

Die Rückgabe entspricht der Rückgabe von GET /api/v1/detections/{id}, jedoch wird ein Array von Erkennungen geliefert.

8.8.4.2 GET /api/v1/detections/{id}

Erkennungen GET	
Pfad	/api/v1/detections
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
id	Required: true
	Type: int
	Description: Detection Id
Query Parameter	
None	

Tabelle 6: GET /api/v1/detections/{id}

```
1  {
2      "id": 1,
3      "license_plate_id": 69,
4      "parking_lot_id": 3,
5      "parking_status": "Outside",
6      "image": null,
7      "created_at": "2021-03-23T02:10:57.000000Z",
8      "updated_at": "2021-03-09T02:08:54.000000Z",
9      "license_plate": {
10          "id": 69,
11          "user_id": 74,
12          "name": "B509DYY",
13          "country": "Austria",
14          "parking_status": "Outside",
15          "allowance_status": "Disallowed",
16          "created_at": "2020-11-19T17:26:03.000000Z",
17          "updated_at": "2021-03-20T21:12:24.000000Z"
18      },
19      "parking_lot": {
20          "id": 3,
21          "name": "Parking Space 1",
22          "description": null,
23          "image": "default.png",
24          "created_at": "2021-03-31T22:40:24.000000Z",
25          "updated_at": "2021-03-31T22:42:57.000000Z"
26      }
27 }
```

Code 92: Beispielhafte GET /api/v1/detections/{id} Rückgabe

8.8.4.3 POST /api/v1/detections

Erkennungen POST	
Pfad	/api/v1/detections
Beschreibung	
HTTP-Methode	POST
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Query Parameter	
name	Required: true
	Type: string
	Description: License plate name
parking_lot_id	Required: true
	Type: int
	Description: Parking Lot Id
Query Parameter	
None	

Tabelle 7: POST /api/v1/detections

Die Rückgabe entspricht der Rückgabe von GET /api/v1/detections/{id}.

8.8.5 Kennzeichen

8.8.5.1 GET /api/v1/plates

Kennzeichen GET	
Pfad	/api/v1/plates
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
None	
Query Parameter	
None	

Tabelle 8: GET /api/v1/plates

Die Rückgabe entspricht GET /api/v1/plates/{id}, jedoch wird ein Array mit einzelnen Kennzeichen geliefert.

8.8.5.2 GET /api/v1/plates/{id}

Kennzeichen GET	
Pfad	/api/v1/plates
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
id	Required: true
	Type: int
	Description: License Plate Id

Tabelle 9: GET /api/v1/plates/{id}

```
1  {
2      "id": 1,
3      "user_id": 8,
4      "name": "FK8VD0",
5      "country": "Austria",
6      "parking_status": "Inside",
7      "allowance_status": "Unknown",
8      "created_at": "2020-12-05T07:04:31.000000Z",
9      "updated_at": "2021-03-22T09:31:46.000000Z",
10     "detections": [
11         {
12             "id": 149,
13             "license_plate_id": 1,
14             "parking_lot_id": 2,
15             "parking_status": "Inside",
16             "image": null,
17             "created_at": "2021-03-12T18:18:40.000000Z",
18             "updated_at": "2021-03-22T11:46:44.000000Z"
19         },
20         ...
21     ],
22 }
```

Code 93: Beispielhafte GET /api/v1/plates/{id} Rückgabe

8.8.6 Parkplätze

8.8.6.1 GET /api/v1/lot

Parkplätze GET	
Pfad	/api/v1/lots
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
None	
Query Parameter	
None	

Tabelle 10: GET /api/v1/lots

Die Rückgabe entspricht GET /api/v1/lots/{i\}, jedoch wird ein Array von einzelnen Parkplätzen geliefert.

8.8.6.2 GET /api/v1/lots/{id}

Parkplätze GET	
Pfad	/api/v1/lots
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
id	Required: true
	Type: int
	Description: Parking Lot Id
Query Parameter	
None	

Tabelle 11: GET /api/v1/lots/{id}

```
1  {
2      "id": 3,
3      "name": "Parking Space 1",
4      "description": null,
5      "image": "default.png",
6      "created_at": "2021-03-31T22:40:24.000000Z",
7      "updated_at": "2021-03-31T22:42:57.000000Z",
8      "detections": [
9          {
10             "id": 1,
11             "license_plate_id": 69,
12             "parking_lot_id": 3,
13             "parking_status": "Outside",
14             "image": null,
15             "created_at": "2021-03-23T02:10:57.000000Z",
16             "updated_at": "2021-03-09T02:08:54.000000Z"
17         },
18         ...
19     ]
20 }
```

Code 94: Beispielhafte GET /api/v1/lots/{id} Rückgabe

8.8.7 Parklücken

8.8.7.1 GET /api/v1/spots

Parklücke GET	
Pfad	/api/v1/spots
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
None	
Query Parameter	
None	

Tabelle 12: GET /api/v1/spots

Die Rückgabe entspricht GET /api/v1/spots/{id}, jedoch wird ein Array von einzelnen Parklücken geliefert.

8.8.7.2 GET /api/v1/spots/{id}

Parklücke GET	
Pfad	/api/v1/spots
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
id	Required: true
	Type: int
	Description: Parking Spot Id
Query Parameter	
None	

Tabelle 13: GET /api/v1/spots/{id}

```
1   {
2     "id": 40,
3     "address": null,
4     "parking_lot_id": 2,
5     "occupied": 0,
6     "created_at": "2021-03-22T17:51:10.000000Z",
7     "updated_at": "2021-02-13T23:23:55.000000Z",
8     "parking_lot": {
9       "id": 2,
10      "name": "Parkplatz Ost",
11      "description": "Dieser Parkplatz ist für Schüler gedacht",
12      "image": "parkplatz_ost.png",
13      "created_at": "2021-03-30T15:44:19.000000Z",
14      "updated_at": "2021-03-30T15:44:19.000000Z"
15    }
16 }
```

Code 95: Beispielhafte GET /api/v1/spots/{id} Rückgabe

8.8.7.3 POST /api/v1/lots/{id}/spots

Parklücken POST	
Pfad	/api/v1/lots/{id}/spots
Beschreibung	
HTTP-Methode	POST
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
id	Required: true
	Type: int
	Description: Parking Lot Id
Query Parameter	
address	Required: true
	Type: int
	Description: Hardware Adresse
occupied	Required: true
	Type: bool
	Description: 1 for occupied 0 for free

Tabelle 14: POST /api/v1/lots/{id}/spots

```
1   {
2     "id": 101,
3     "address": 1337,
4     "parking_lot_id": 1,
5     "occupied": 0,
6     "created_at": "2021-03-31T22:27:21.000000Z",
7     "updated_at": "2021-03-31T22:27:21.000000Z",
8     "parking_lot": {
9       "id": 1,
10      "name": "Parkplatz West",
11      "description": "Dieser Parkplatz ist für Lehrer gedacht",
12      "image": "parkplatz_west.png",
13      "created_at": "2021-03-30T15:44:19.000000Z",
14      "updated_at": "2021-03-30T15:44:19.000000Z"
15    }
16 }
```

Code 96: Beispielhafte POST /api/v1/lots/{id}/spots Rückgabe

8.8.8 Einstellungen

8.8.8.1 GET /api/v1/settings

Einstellungen GET	
Pfad	/api/v1/settings
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
None	
Query Parameter	
None	

Tabelle 15: GET /api/v1/settings

```

1   [
2     {
3       "id": 1,
4       "key": "site_title",
5       "value": "APM"
6     },
7     ...
8   ]

```

Code 97: Beispielhafte GET /api/v1/settings Rückgabe

8.8.9 Benutzer

8.8.9.1 GET /api/v1/users

Benutzer GET	
Pfad	/api/v1/users
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
None	
Query Parameter	
None	

Tabelle 16: GET /api/v1/users

Die Rückgabe entspricht GET /api/v1/users/{id}, jedoch wird ein Array von einzelnen Benutzern geliefert.

8.8.9.2 GET /api/v1/users/{id}

Benutzer GET	
Pfad	/api/v1/users
Beschreibung	
HTTP-Methode	GET
Authentifizierung	Bearer Token
Rückgabetyp	application/json
Path Parameter	
id	Required: true
	Type: int
	Description: User Id
Query Parameter	
None	

Tabelle 17: GET /api/v1/users/{id}

```

1   {
2     "id": 1,
3     "first_name": "Philipp",
4     "last_name": "Kraft",
5     "email": "Mail@Philipp-Kraft.com",
6     "avatar": "philipp.png",
7     "email_verified_at": "2021-03-30T15:44:18.000000Z",
8     "last_login_ip": "127.0.0.1",
9     "last_login_at": "2021-04-01 09:22:01",
10    "created_at": "2021-03-30T15:44:18.000000Z",
11    "updated_at": "2021-04-01T09:22:01.000000Z",
12    "license_plates": []
13 }
```

Code 98: Beispielhafte GET /api/v1/users/{id} Rückgabe

9 Zusammenfassung und Ausblick

Zusammenfassend ist zu erwähnen, dass alle drei Teile durchaus realisierbar sind. Die Kennzeichenerkennung ist in der Lage einzelne Fahrzeugschilder mit ausreichender Genauigkeit auszulesen. Die Detektion der Fahrzeuge auf den einzelnen Parklücken ist auch sehr zuverlässig und erfüllt daher auch seine Grundaufgabe. Die API erfüllt ebenfalls die gewünschte datenübermittelnde Funktion mit dem Webinterface. Es ist möglich in diesem die einzelnen Parkplätze, deren Fahrzeuge und freie Parklücken einzusehen und zu verwalten. Diese Web-Applikation ist auch für jeden Kunden anpassbar.

Abschließend kann gesagt werden, dass dieses Verwaltungssystem dem Benutzer viele Vorteile bringt und das Überwachen des Parkplatzes vereinfacht.

10 Anhang

Abbildungsverzeichnis

1	Arbeitspakete Teil 1	12
2	Arbeitspakete Teil 2	13
3	Gantt-Chart Teil 1	14
4	Gantt-Chart Teil 2	14
5	Vor Bilateraler Filterung	21
6	Nach Bilateraler Filterung	21
7	Graustufenbild	22
8	Binäres Bild nach Thresholding	22
9	Binäres Bild nach Thresholding	23
10	Nach Erosion	23
11	Ablaufdiagramm der Kennzeichenerkennung	27
12	Beispiel einer Anwendung von Matplotlib	33
13	GPIO-Pins des Raspberry Pi	38
14	Visualisierung mit Matplotlib	47
15	Visualisiertes Ergebnis der Kennzeichenerkennung	48
16	Raspberry Pi	50
17	Raspberry Pi mit Kamera	51
18	Fehler bei Konturerkennung	54
19	Resultat mit maschinellem Lernen	54
20	Ablauf von WPOD-NET	56
21	Beispiel eines Zeichens aus dem Datensatz	57
22	Verlauf des Modell Trainings	60
23	Installation der Spule	63
24	LTSPice Blockbild zweier RL-Glieder	65
25	Stromkurven zweier RL-Glieder	66
26	RL-Oszillator mit NE555	67
27	RL-Oszillator mit NE555 Zeitsignale	68
28	Vergleich des NE555-Oszillators bei unterschiedlichen L	69
29	Eddy Currents in einem Leiter	70

30	Parallelschwingkreis in LTSpice	71
31	Gedämpfte Schwingung	72
32	Colpitts LC-Glied	74
33	Aktiver Colpitts-Oszillator	75
34	Zeitsignale Colpitts-Oszillator	76
35	Beispiel einer UART Zeichenübertragung	78
36	RS485 Widerstandsnetzwerk	79
37	Logische Tabelle für U_{AB}	80
38	Beispiel einer RS485 Zeichenübertragung	80
39	RJ45 Steckerbuchse und Kabel	81
40	Logische Übersicht des Bussystems	82
41	Physische Übersicht des Bussystems	83
42	Anordnung der Slave-Geräte mit Adresslogikleitung	84
43	RJ45 Pinbelegung des Slave-Geräts	85
44	RJ45 Pinbelegung des Master-USB-Geräts	85
45	Elektrische Eigenschaften des L78L05ACD	86
46	Ablauf von Steueranfragen	88
47	Master Datenframe	89
48	Slave Response Datenframe	90
49	HTML5 Logo von https://www.w3.org/html/logo	93
50	Einfache HTML Seite von https://www.w3.org/html/logo	95
51	Einfache HTML Seite mit CSS	97
52	Laravel MVC Muster	100
53	Eloquent ORM Workflow von https://dev.to/xenoxdev/mastering-laravel-eloquent-orm-the-eloquent-journey-part-1-1571	106
54	Advanced System Settings	110
55	System Variables	111
56	Environment Variables	111
57	PHP Version	112
58	phpMyAdmin Webinterface	112
59	Docker WSL Integration	115

60	Docker Container Steuerung	115
61	Debian Default Page	117
62	Action Secrets	126
63	Github Action Übersicht	127
64	App Layout	130
65	Login Seite mit App Layout	131
66	Admin Layout	132
67	Dashboard mit Admin Layout	133
68	Navbar Variante 1	134
69	Navbar Variante 2	134
70	Navbar Variante 3	135
71	Content Header	136
72	Success Session Alert	137
73	Sidebar Desktop	140
74	Footer	141
75	Dashboard Sicherheitsnachricht	142
76	Dashboard Statistik	143
77	Benutzer ER	144
78	Liste aller Benutzer	145
79	Benutzer editieren	146
80	Benutzer, Rollen und Rechte ER	148
81	Benutzer erstellen	153
82	Neuigkeit editieren	154
83	Home Neuigkeiten	155
84	Parking Lots, Parking Spots und Detections ER	156
85	Übersicht aller Parkplätze	156
86	Parklücken und Erkennungen eines Parkplatzes	157
87	Einstellunge Tabelle	158
88	Profil	159
89	Navigationsleiste Dropdown	160
90	API Token	163

91 API Token Popup 164

Tabellenverzeichnis

1	Zeitaufwendungen Übersicht	15
2	RJ45 allgemeine Pinbelegung	83
3	allgemeiner Aufbau eines Datenframes	87
4	Verfügbare Rechte	150
5	GET /api/v1/detections	164
6	GET /api/v1/detections/{id}	165
7	POST /api/v1/detections	167
8	GET /api/v1/plates	168
9	GET /api/v1/plates/{id}	168
10	GET /api/v1/lots	170
11	GET /api/v1/lots/{id}	171
12	GET /api/v1/spots	173
13	GET /api/v1/spots/{id}	174
14	POST /api/v1/lots/{id}/spots	176
15	GET /api/v1/settings	178
16	GET /api/v1/users	179
17	GET /api/v1/users/{id}	180

Codeverzeichnis

1	PIP Installation von Jupyter Notebook	29
2	PIP Installation von Numpy	30
3	PIP Installation von OpenCV	31
4	Installation benötigter Tools und Bibliotheken für OpenCV	31
5	Klonen von OpenCV von GitHub	32
6	Kompilieren von OpenCV	32
7	Abschließende Installation von OpenCV	32
8	PIP Installation von Matplotlib	33
9	PIP Installation von Keras	34
10	PIP Installation von Tensorflow	34
11	Benötigte Tools für Tensorflow	35
12	Tensorflow von GitHub herunterladen	35
13	Entpacken von Tensorflow	35
14	Zusätzlich erforderliche Librarys	35
15	Kompilieren von Tensorflow	36
16	Abschließen der Installation von Tensorflow	36
17	PIP Installation von Scikit-learn	37
18	PIP Installation von Requests	37
19	PIP Installation von Gpiozero	38
20	PIP Installation von Picamera	39
21	WPOD-NET Modell laden	40
22	Anwendung des WPOD-NET Modells	41
23	get_plate	41
24	Bild vorbereiten für WPOD-NET	41
25	Kennzeichen lokalisieren	42
26	Bild aufnehmen	43
27	Konturen finden	43
28	Konturen sortieren	44
29	Filterung der korrekten Konturen	44

30	Bildverarbeitungsalgorithmen	45
31	Anwendung einer Visualisierung mit Matplotlib	46
32	predict_from_model	47
33	Anwendung der Zeichenerkennung	48
34	Senden der Ergebnisse an die Datenbank	49
35	Erstellen des Modells basierend auf MobileNetV2	58
36	Trainieren des Modells und Einstellungen anpassen	59
37	index.html	94
38	style.css	96
39	web.php	101
40	example.blade.php	102
41	UserController.php	103
42	create_users_table.php	105
43	Eloquent Query	106
44	Raw SQL Query	106
45	UserUpdate Request	108
46	WSL Feature Feature aktivierens	113
47	Virtual Machine Feature aktivieren	113
48	WSL 2 auswählen	114
49	Respositorys updaten	116
50	Apache installieren	117
51	PHP installieren	117
52	Mariadb installieren	118
53	MariaDB Secure Installation	118
54	MariaDB konfiguration	118
55	phpMyAdmin Download	118
56	phpMyAdmin Entpacken	119
57	phpMyAdmin Rechte und Verzeichnisse	119
58	phpMyAdmin Konfigurationsdatei erstellen	119
59	phpMyAdmin Blowfish Secret und TempDir	119
60	phpmyadmin.conf	120

61	Virtual Host aktivieren	120
62	apm.conf	121
63	Download Composer Installer	122
64	Composer Setup	122
65	Git Installation	122
66	Git Remote Origin	123
67	deploy.sh	123
68	serverdeploy.sh	124
69	main.yml	126
70	primary.blade.php	128
71	Verwendung eines Button Components	128
72	Modularer Button Component	129
73	Ausschnitt 1 navigation.blade.php	134
74	Ausschnitt 2 navigation.blade.php	135
75	Erstellung der Breadcrumbs	136
76	Controller mit success Nachricht	137
77	Session Alert Component	137
78	Sidebar Header	138
79	Sidebar Link	139
80	AuthenticatedSessionController.php Store Methode	142
81	Profilbild Upload	147
82	UserController.php Profilbild Upload	147
83	Permission Vergabe Methode	151
84	Rolle erstellen und Rechte zuweisen	151
85	UserPolicy Index	152
86	UserController Index	152
87	SettingsServiceProvider	158
88	Seiten Titel Konfiguration	159
89	Spracheauswahl	161
90	Spracheauswahl Route	161
91	Localization Middleware	162

92	Beispielhafte GET /api/v1/detections/{id} Rückgabe	166
93	Beispielhafte GET /api/v1/plates/{id} Rückgabe	169
94	Beispielhafte GET /api/v1/lots/{id} Rückgabe	172
95	Beispielhafte GET /api/v1/spots/{id} Rückgabe	175
96	Beispielhafte POST /api/v1/lots/{id}/spots Rückgabe	177
97	Beispielhafte GET /api/v1/settings Rückgabe	178
98	Beispielhafte GET /api/v1/users/{id} Rückgabe	180

Abkürzungsverzeichnis

PK Philipp Kraft

DK Dennis Köb

SB Samuel Bleiner

APM Advanced Parking Monitoring

HTML Hypertext Markup Language

W3C World Wide Web Consortium

CSS Cascading Style Sheets

PHP Hypertext Preprocessor

JS JavaScript

DOM Document Object Model

MVC Model-View-Controller

ORM Object-relational mapping

CLI Command-line interface

SQL Structured Query Language

UART Universal Asynchronous Receiver Transmitter

ASCII American Standard Code for Information Interchange

IC Integrated Circuit

SVG Scalable Vector Graphics

USB Universal Serial Bus

WYSIWYG What You See Is What You Get

CRUD Create Read Update Delete

MSB Most Significant Byte

LSB Least Significant Byte

API Application programming interface

Literaturverzeichnis

World Wide Web Consortium. *HTML5 Logo*. [Online; Abgerufen am 31. März 2021]. 2014. URL:

<https://www.w3.org/html/logo>.

XenoX. *Eloquent ORM Workflow*. [Online; Abgerufen am 31. März 2021]. 2020. URL: https :

// dev . to / xenoxdev / mastering - laravel - eloquent - orm - the - eloquent - journey - part - 1 - 1571.