

# 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, 31. März 2021

---

Philippe Kraft

---

Dennis Köb

---

Samuel Bleiner

## Kurzfassung

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.

## Abstract

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.

## Vorwort

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.

## Danksagung

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. Ein lesenswertes Buch ist »Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]«<sup>1</sup>

<sup>1</sup>Albert Einstein. »Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]«. In: *Annalen der Physik* 322.10 (1905), S. 891–921. doi: <http://dx.doi.org/10.1002/andp.19053221004>.

# Inhaltsverzeichnis

<b>Eidesstattliche Erklärung</b>	i
<b>Kurzfassung</b>	ii
<b>Abstract</b>	iii
<b>Vorwort</b>	iv
<b>Danksagung</b>	v
<b>1 Projektteam</b>	8
<b>2 Projektbetreuer</b>	9
<b>3 Auftragnehmer</b>	10
<b>4 Projektplanung</b>	11
<b>5 Rechtliches</b>	12
<b>6 Einleitung</b>	13
<b>7 Projektantrag</b>	14
<b>8 Kennzeichenerkennung</b>	15
8.1 Anforderungen . . . . .	15
8.2 Vorstudie . . . . .	16
8.3 Bildverarbeitung . . . . .	17
8.3.1 Einleitung . . . . .	17
8.3.2 Bilaterale Filterung . . . . .	17
8.3.3 Thresholding . . . . .	18
8.3.4 Erosion . . . . .	19
8.3.5 Farbraum . . . . .	20
8.3.5.1 RGB . . . . .	20
8.3.5.2 Graustufen . . . . .	20

8.3.5.3	BGR . . . . .	21
8.3.6	Konturerkennung . . . . .	21
8.4	Kennzeichenerkennungsprogramm . . . . .	22
8.4.1	Einleitung . . . . .	22
8.4.2	Programmiersprache . . . . .	22
8.4.3	Konzept . . . . .	22
8.4.3.1	Bildaufnahme . . . . .	22
8.4.3.2	Kennzeichenerfassung . . . . .	23
8.4.3.3	Kennzeichensegmentierung . . . . .	23
8.4.3.4	Zeichenerkennung . . . . .	23
8.4.3.5	Anbindung an Datenbank . . . . .	23
8.4.4	Ablauf . . . . .	24
8.4.5	Verwendete Libraries . . . . .	25
8.4.5.1	Versionsübersicht der verwendeten Libraries . . . . .	25
8.4.5.2	Jupyter Notebook . . . . .	26
8.4.5.3	Numpy . . . . .	27
8.4.5.4	OpenCV . . . . .	27
8.4.5.5	Matplotlib . . . . .	29
8.4.5.6	Keras . . . . .	30
8.4.5.7	Tensorflow . . . . .	31
8.4.5.8	local_utils . . . . .	33
8.4.5.9	Scikit-learn . . . . .	33
8.4.5.10	Requests . . . . .	34
8.4.5.11	Gpiozero . . . . .	34
8.4.5.12	Picamera . . . . .	35
8.4.6	Wichtige Programmteile . . . . .	36
8.4.6.1	WPOD-NET-Modell laden . . . . .	36
8.4.6.2	Kennzeichen lokalisieren . . . . .	38
8.4.6.3	Bild aufnehmen . . . . .	39
8.4.6.4	Zeichen mittels Konturerkennung finden . . . . .	40
8.4.6.5	Bildverarbeitung . . . . .	41

8.4.6.6	Visualisierung mittels Matplotlib . . . . .	42
8.4.6.7	Modell für Zeichenerkennung laden und anwenden . . . . .	44
8.4.6.8	Resultat mittels API an Datenbank senden . . . . .	46
8.5	Raspberry Pi . . . . .	47
8.5.1	Einleitung . . . . .	47
8.5.2	Wahl des Raspberry Pi . . . . .	47
8.5.3	Kamera . . . . .	48
8.6	Maschinelles Lernen . . . . .	49
8.6.1	Einleitung . . . . .	49
8.6.2	Definition . . . . .	50
8.6.3	Vergleich Maschinelles Lernen / Bildverarbeitung . . . . .	50
8.6.4	Verwendete Modelle für Maschinelles Lernen . . . . .	52
8.6.4.1	WPOD-NET . . . . .	52
8.6.4.2	MobileNetV2 . . . . .	53
8.7	Modell Training . . . . .	54
<b>9</b>	<b>Fahrzeugerkennung</b> . . . . .	<b>58</b>
9.1	Anforderungen . . . . .	58
9.2	Vorstudie . . . . .	58
9.3	Erkennung von Metallen über Spulen . . . . .	58
9.3.1	Messung ferromagnetischer Metalle . . . . .	61
9.3.1.1	RL-Oszillator mit Timer Baustein . . . . .	62
9.3.2	Messung paramagnetischer Metalle . . . . .	67
9.3.2.1	LC Oszillatoren . . . . .	68
9.3.2.2	Einfluss von Induktivitätsänderungen auf LC-Oszillatoren . . . . .	69
9.3.2.3	Messung der Frequenz eines Colpitts-Oszillator . . . . .	70
9.4	RS485 Bussystem . . . . .	74
9.4.1	Überblick . . . . .	74
9.4.1.1	ASCII . . . . .	74
9.4.1.2	UART . . . . .	74
9.4.1.3	RS485 . . . . .	75

9.4.1.4	Aufbau . . . . .	78
9.4.2	Elektrische Spezifikation . . . . .	78
9.4.3	Implementation eines eigenen Protokolls . . . . .	78
9.5	Mikrokontroller Slave-Geräte . . . . .	78
9.5.1	Überblick . . . . .	78
9.5.2	Atmega328PB . . . . .	78
9.5.3	Peripherie des Mikrocontrollers . . . . .	78
9.5.3.1	Spannungswandler . . . . .	78
9.5.3.2	RS485 Pegelwandler . . . . .	78
9.5.3.3	Digitale Ein- und Ausgänge . . . . .	78
9.5.4	Layout des Slave-Gerätes . . . . .	79
9.5.5	Gehäuse . . . . .	79
9.6	USB-Master . . . . .	79
9.6.1	USB-Bussadapter Gerät . . . . .	79
9.6.1.1	Überblick . . . . .	79
9.6.1.2	FT232RL . . . . .	79
9.6.1.3	Spannungsversorgung . . . . .	79
9.6.1.4	USB-C Anschluss . . . . .	79
9.6.1.5	Layout des Master-Geräts . . . . .	79
9.6.1.6	Gehäuse . . . . .	79
9.6.2	Master Programm . . . . .	79
9.6.2.1	Benötigte Software . . . . .	79
9.6.2.2	Adressvergabe . . . . .	79
9.6.2.3	Frequenzauslesung . . . . .	79
9.6.2.4	Auswertung . . . . .	79
9.6.2.5	API-Post . . . . .	79
9.6.3	RaspberryPi als Mastergerät . . . . .	79
9.6.3.1	SSH Remote Zugriff . . . . .	79
9.6.3.2	Code Deployment . . . . .	80
9.6.3.3	Unitest . . . . .	81

<b>10 Webinterface</b>	<b>81</b>
10.1 Einleitung . . . . .	81
10.2 Verwendete Technologien . . . . .	81
10.2.1 HTML . . . . .	81
10.2.1.1 Beispielhafte HTML Seite . . . . .	82
10.2.2 CSS . . . . .	83
10.2.3 JavaScript . . . . .	85
10.2.4 PHP . . . . .	86
10.2.5 TailwindCSS . . . . .	86
10.2.6 Laravel . . . . .	86
10.2.6.1 Routing . . . . .	88
10.2.6.2 Blade Templates . . . . .	89
10.2.6.3 Controllers . . . . .	90
10.2.6.4 Artisan CLI . . . . .	91
10.2.6.5 Migrations . . . . .	92
10.2.6.6 Eloquent ORM . . . . .	94
10.2.6.7 Laravel Sanctum . . . . .	95
10.3 Lokale Entwicklungsumgebung mit Laragon . . . . .	95
10.3.1 Benötigte Software . . . . .	95
10.3.2 Konfiguration von PHP . . . . .	96
10.3.3 Installation von phpMyAdmin . . . . .	98
10.4 Lokale Entwicklungsumgebung mit WSL und Docker . . . . .	99
10.4.1 Benötigte Software . . . . .	99
10.4.2 Installation von WSL . . . . .	99
10.4.2.1 2. Schritt: Virtual Machine Aktivieren . . . . .	99
10.4.2.2 3. Schritt: Linux Kernel Update . . . . .	100
10.4.2.3 4. Schritt: WSL 2 . . . . .	100
10.4.2.4 5. Schritt: Linux Distribution herunterladen . . . . .	100
10.4.3 Installation von Docker . . . . .	100
10.5 Production Server . . . . .	101
10.5.1 Benötigte Software . . . . .	102

10.5.2 Installation des LAMP Stacks . . . . .	102
10.5.2.1 Apache . . . . .	102
10.5.2.2 PHP . . . . .	103
10.5.2.3 MariaDB . . . . .	103
10.5.2.4 phpMyAdmin . . . . .	104
10.5.2.5 Webinterface Virtual Host . . . . .	107
10.5.2.6 Installation von Composer . . . . .	108
10.5.3 Deployment mit Github Actions . . . . .	108
10.5.3.1 Git Setup . . . . .	108
10.5.3.2 Deploy Script . . . . .	109
10.5.3.3 Server Deploy Script . . . . .	110
10.5.3.4 Github Action . . . . .	111
10.6 Grundlegender Aufbau Frontend . . . . .	113
10.6.1 Components . . . . .	113
10.6.1.1 Anonymous Components . . . . .	113
10.6.1.2 Components erstellen . . . . .	113
10.6.1.3 Components verwenden . . . . .	114
10.6.1.4 Attribute übergeben . . . . .	114
10.6.2 Layouts . . . . .	115
10.6.3 Navbar . . . . .	119
10.6.4 Sidebar . . . . .	121
10.6.5 Footer . . . . .	124
10.7 Funktionen . . . . .	124
10.7.1 Dashboard . . . . .	124
10.7.2 Login- und Registersystem . . . . .	127
10.7.3 News . . . . .	127
10.7.4 Benutzerverwaltung . . . . .	127
10.7.5 Rechte- und Rollenverwaltung . . . . .	127
10.7.6 Parkplatzverwaltung . . . . .	127
10.7.7 Kennzeichenverwaltung . . . . .	127
10.7.8 Erkennungsverlauf . . . . .	127

10.7.9 Seiten Einstellungen . . . . .	127
10.7.10 API Schlüssel . . . . .	127
10.7.11 Displays . . . . .	127
10.7.12 Notifications . . . . .	127
10.7.13 Profil . . . . .	127
10.7.14 Lokalisierung . . . . .	127
<b>10.8 Performance und Sicherheit . . . . .</b>	<b>127</b>
10.8.1 Form Validation . . . . .	127
10.8.2 Authentifizierung . . . . .	127
10.8.3 Authentisierung . . . . .	127
10.8.4 SQL Injection . . . . .	127
10.8.5 Cross-Site-Scripting . . . . .	127
10.8.6 Lighthouse . . . . .	127
<b>11 Zusammenfassung und Ausblick</b>	<b>128</b>
<b>12 Anhang</b>	<b>129</b>
<b>Abbildungsverzeichnis</b>	<b>130</b>
<b>Codeverzeichnis</b>	<b>134</b>
<b>Abkürzungsverzeichnis</b>	<b>137</b>
<b>Literaturverzeichnis</b>	<b>138</b>

## 1 Projektteam

## 2 Projektbetreuer

### 3 Auftragnehmer

## 4 Projektplanung

## 5 Rechtliches

## 6 Einleitung

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.

## 7 Projektantrag

## 8 Kennzeichenerkennung

### 8.1 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.

## 8.2 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.

## 8.3 Bildverarbeitung

### 8.3.1 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.

### 8.3.2 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 1: Vor Bilateraler Filterung



Abbildung 2: Nach Bilateraler Filterung

In Abbildung 1 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 2 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.

### 8.3.3 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 3: Graustufenbild



Abbildung 4: Binäres Bild nach Thresholding

In Abbildung 3 sieht man ein solches Graubild welches nur verschieden Graustufen aufweist. In Abbildung 4 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 Otsu<sup>2</sup>, welche mit dem Schwellenwert die Pixel in Vordergrund und Hintergrund unterteilt.

#### 8.3.4 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

<sup>2</sup>Benannt 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 5: Binäres Bild nach Thresholding



Abbildung 6: Nach Erosion

In Abbildung 5 und 6 sieht man die Anwendung der Erosion. Die Konturen der einzelnen Zeichen im Kennzeichen sind in Abbildung 6 nach der Erosion deutlicher erkennbar als davor.

### 8.3.5 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:

#### 8.3.5.1 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.

#### 8.3.5.2 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.

### 8.3.5.3 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.

### 8.3.6 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 1985<sup>3</sup> und liefert eine Liste mit allen Konturen. Die Konturen werden in der Liste als ein Array von Koordinaten abgespeichert.

## 8.4 Kennzeichenerkennungsprogramm

### 8.4.1 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.

### 8.4.2 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.

### 8.4.3 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.

#### 8.4.3.1 Bilddaufnahme

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

<sup>3</sup>Topological 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.

#### **8.4.3.2 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 Jung<sup>4</sup>. 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.

#### **8.4.3.3 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.

#### **8.4.3.4 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 MobileNetV2<sup>5</sup>, welches mit einem Datensatz von über 35 000 Bildern auf die Erkennung von Zeichen aus Bildern trainiert wurde.

#### **8.4.3.5 Anbindung an Datenbank**

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

<sup>4</sup>License Plate Detection and Recognition in Unconstrained Scenarios

<sup>5</sup>Neuronales 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.

#### 8.4.4 Ablauf

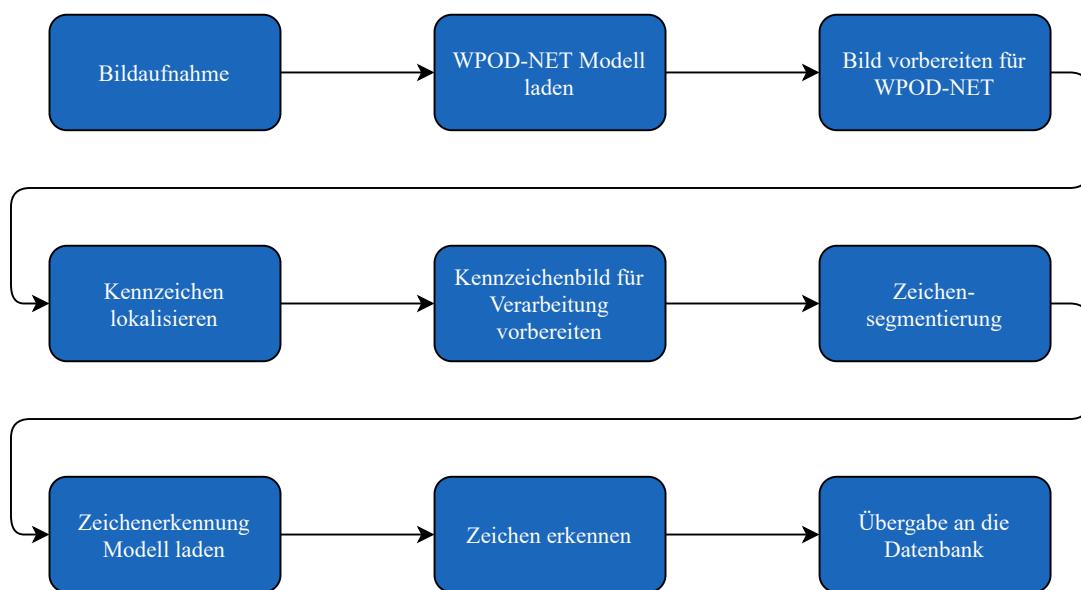


Abbildung 7: Ablaufdiagramm der Kennzeichenerkennung

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“<sup>6</sup> 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 Kennzeichenbild 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.

<sup>6</sup>Modell für Maschinelles Lernen für Kennzeichenerfassung

## 8.4.5 Verwendete Libraries

### 8.4.5.1 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 Raspbian<sup>7</sup>, 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

<sup>7</sup>Raspbian 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

#### 8.4.5.2 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 pip<sup>8</sup> 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

<sup>8</sup>Paketverwaltungsprogramm 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.

#### 8.4.5.3 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

#### 8.4.5.4 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 Vision<sup>9</sup> 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-Learning<sup>10</sup> dar. In

<sup>9</sup>Die Computergestützte Auswertung und Verarbeitung von Kamerabildern

<sup>10</sup>Methode 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.

#### 8.4.5.5 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.

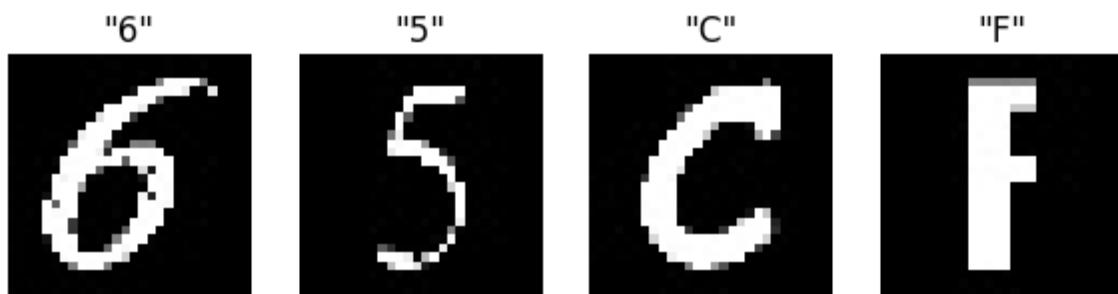


Abbildung 8: Beispiel einer Anwendung von Matplotlib

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

```
1 pip install matplotlib
```

Code 8: PIP Installation von Matplotlib

#### 8.4.5.6 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

#### 8.4.5.7 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.

#### 8.4.5.8 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](https://github.com/quangnhat185/Plate_detect_and_recognize/blob/master/local_utils.py)

#### 8.4.5.9 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

#### 8.4.5.10 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

#### 8.4.5.11 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:

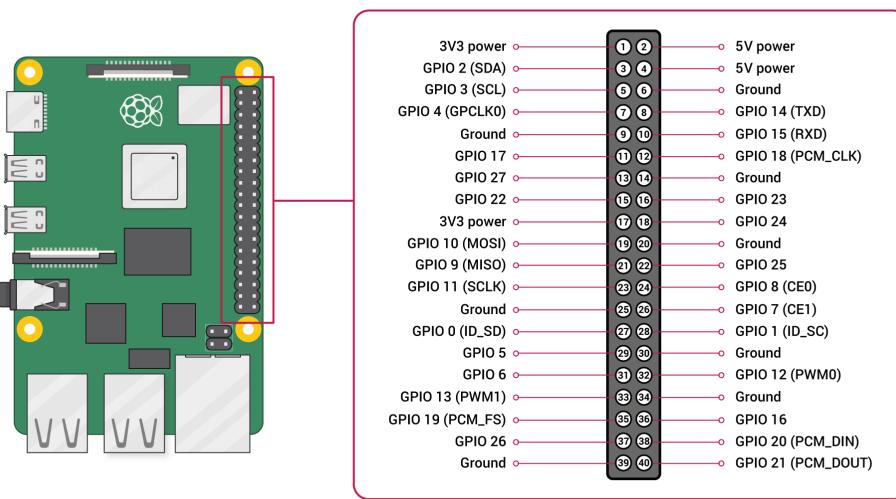


Abbildung 9: GPIO-Pins des Raspberry Pi

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

#### 8.4.5.12 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

#### 8.4.6 Wichtige Programmteile

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

##### 8.4.6.1 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

#### 8.4.6.2 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

#### 8.4.6.3 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
15     ↳ /image.jpg')
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.

#### 8.4.6.4 Zeichen mittels Konturerkennung finden

```

1 cnt, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
2     ↳ 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

#### 8.4.6.5 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.

#### 8.4.6.6 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.

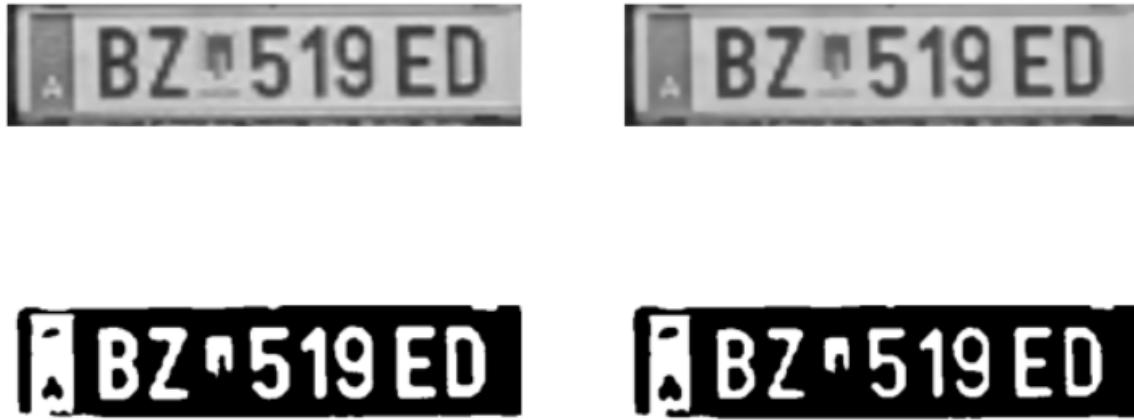


Abbildung 10: Visualisierung mit Matplotlib

#### 8.4.6.7 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 11: Visualisiertes Ergebnis der Kennzeichenerkennung

#### 8.4.6.8 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=[('image',('kennzeichen.png',open('lp_image/lp.png','rb'),'image/png'))]
12
13 header = {"Authorization": "Bearer"
14     ↳ 5|4NcJgCrR9FkeTFeTvRGHLQqoJBeY8IDqxborpuuS"}
15
16
17 #Senden der Daten an die Datenbank und Ausgabe der Response
18
19 resp = requests.post(url, data=lp_result, files=lp_files, headers=header)
20 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.

## 8.5 Raspberry Pi

### 8.5.1 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

### 8.5.2 Wahl des Raspberry Pi

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

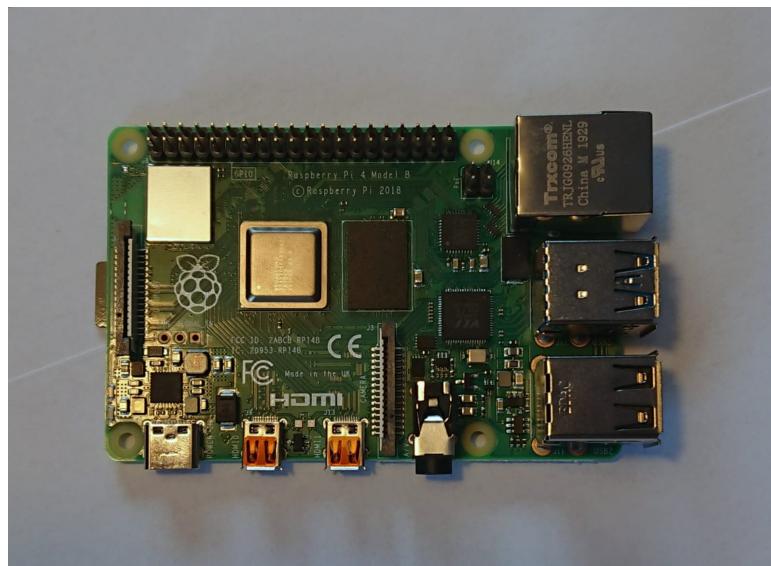


Abbildung 12: Raspberry Pi

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.

### 8.5.3 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 13: Raspberry Pi mit Kamera

## 8.6 Maschinelles Lernen

### 8.6.1 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.

### 8.6.2 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.

### 8.6.3 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 14: 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.

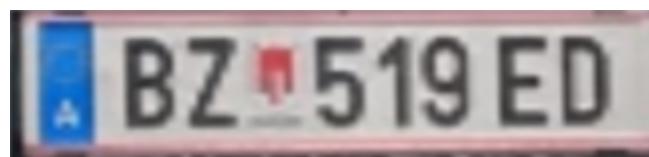


Abbildung 15: Resultat mit maschinellem Lernen

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.

#### 8.6.4 Verwendete Modelle für Maschinelles Lernen

##### 8.6.4.1 WPOD-NET

Das „Warped Planar Object Detection Network“ (WPOD-NET) ist ein künstliches Neuronales Netz, welches von Sérgio Montazolli Silva und Cláudio Rosita Jung<sup>11</sup> 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 YOLOv2<sup>12</sup> 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.

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

<sup>12</sup>J. 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.

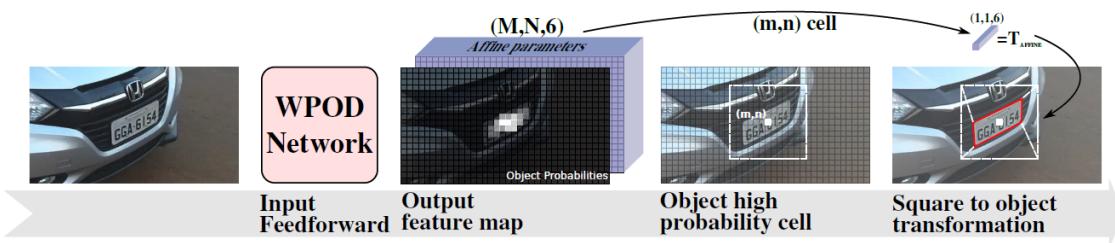


Abbildung 16: Ablauf von WPOD-NET

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.

#### 8.6.4.2 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.

## 8.7 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.

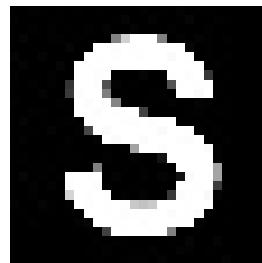


Abbildung 17: Beispiel eines Zeichens aus dem Datensatz

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     baseModel = MobileNetV2(weights="imagenet", include_top=False,
4         ↳ input_tensor=Input(shape=(80, 80, 3)))
5
6     headModel = baseModel.output
7     headModel = AveragePooling2D(pool_size=(3, 3))(headModel)
8     headModel = Flatten(name="flatten")(headModel)
9     headModel = Dense(128, activation="relu")(headModel)
10    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"])
19
20 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,
9                             verbose=0),
10                        ModelCheckpoint(filepath="License_character_recognition.h5",
11                                      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.

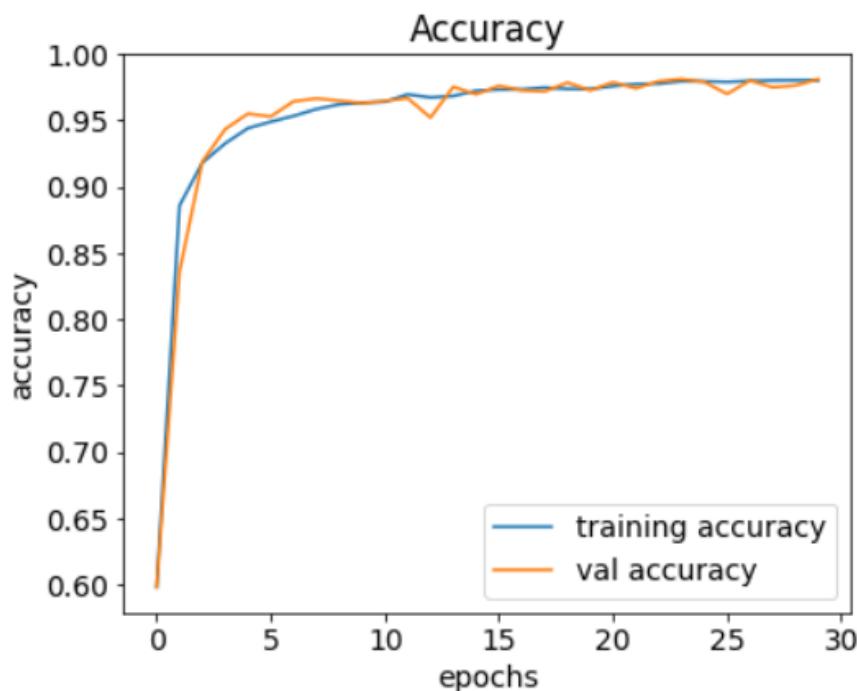


Abbildung 18: Verlauf des Modell Trainings

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.

## 9 Fahrzeugerkennung

### 9.1 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.

### 9.2 Vorstudie

#### 9.3 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  $\phi$ . Es gelten für diese Größen der folgende Zusammenhänge:

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

Wobei

$\Phi$  = 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

$\mu_r$  = relative Permeabilität

$A$  = Fläche der Spule

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

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

Der Fluss  $\Phi$  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  $\Phi$  und somit die magnetische Flussdichte  $B$  beeinflussen kann.

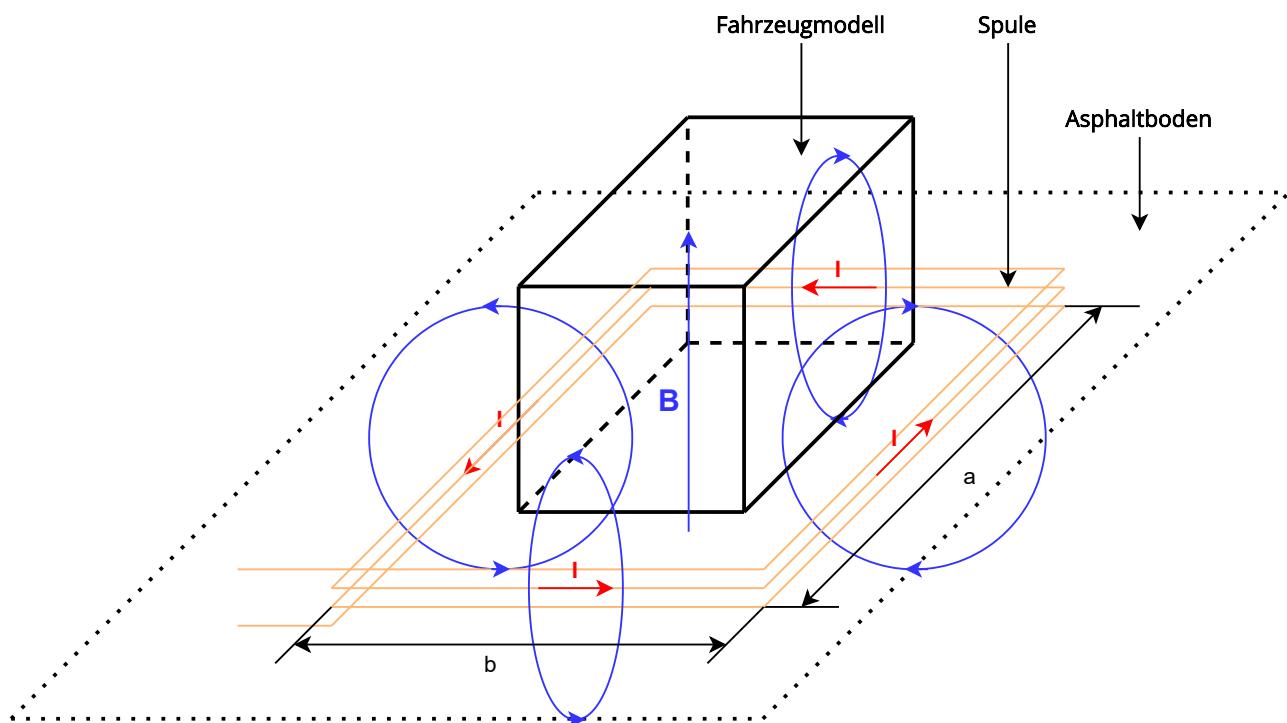


Abbildung 19: Installation der Spule

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

$\Psi$  = 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.

### 9.3.1 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.

### 9.3.1.1 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.

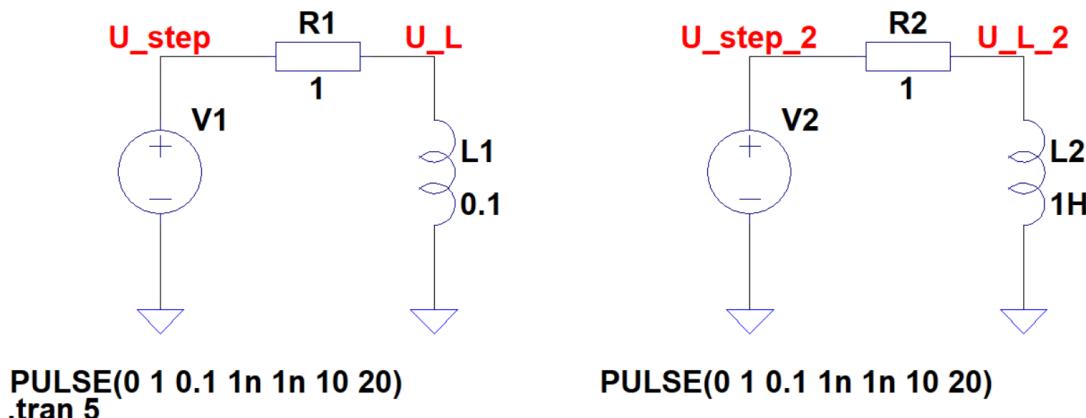


Abbildung 20: LTSPice Blockbild zweier RL-Glieder

Die daraus resultierende Simulation im Zeitbereich ergibt folgendes Ergebnis: hmm

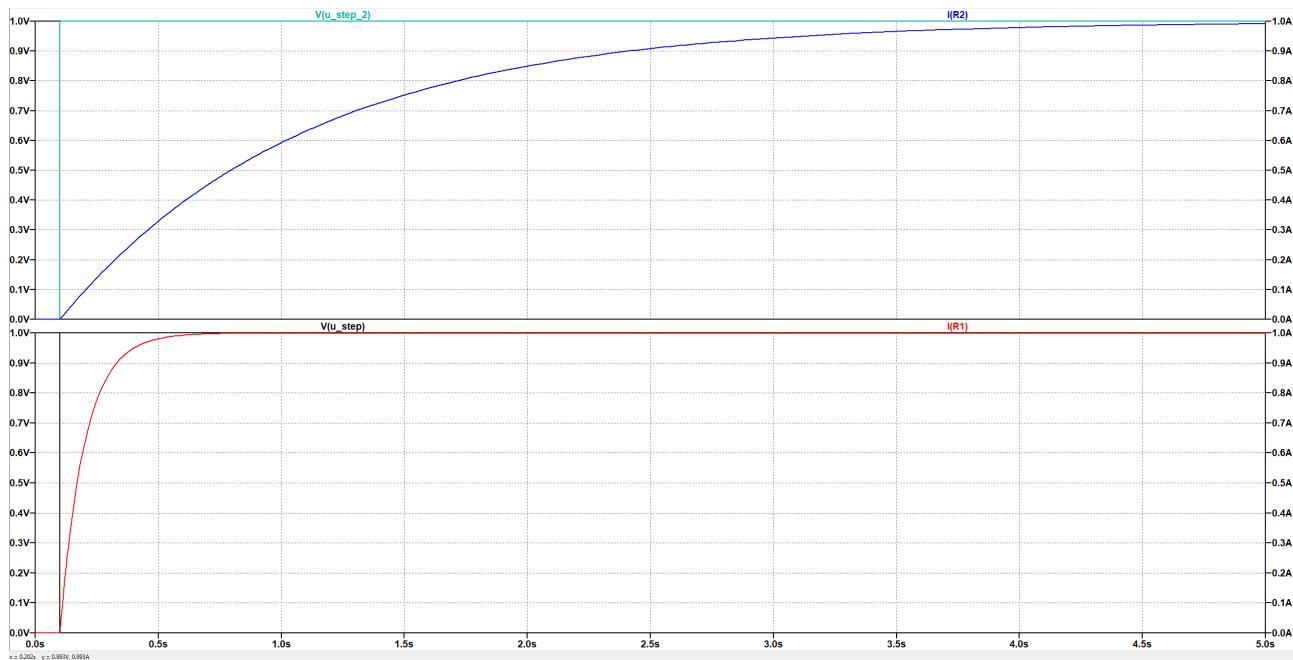


Abbildung 21: Stromkurven zweier RL-Glieder

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

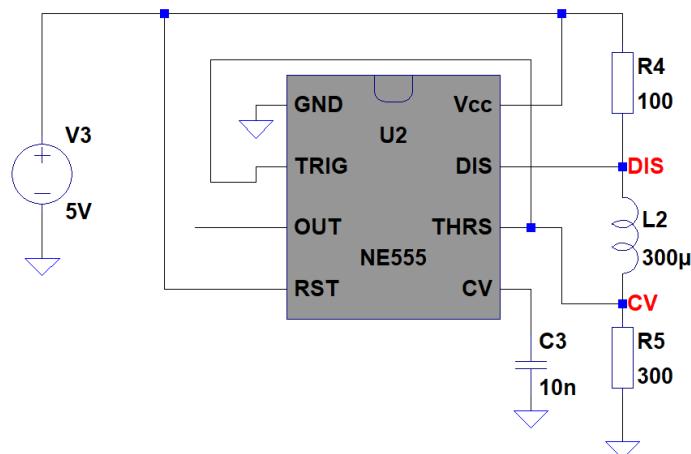
$R$  = Widerstand in  $\Omega$

$U_0$  = Spannung die nach dem einschalten anliegt in V

$t$  = Zeit in s

$\tau$  = 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 22: RL-Oszillator mit NE555

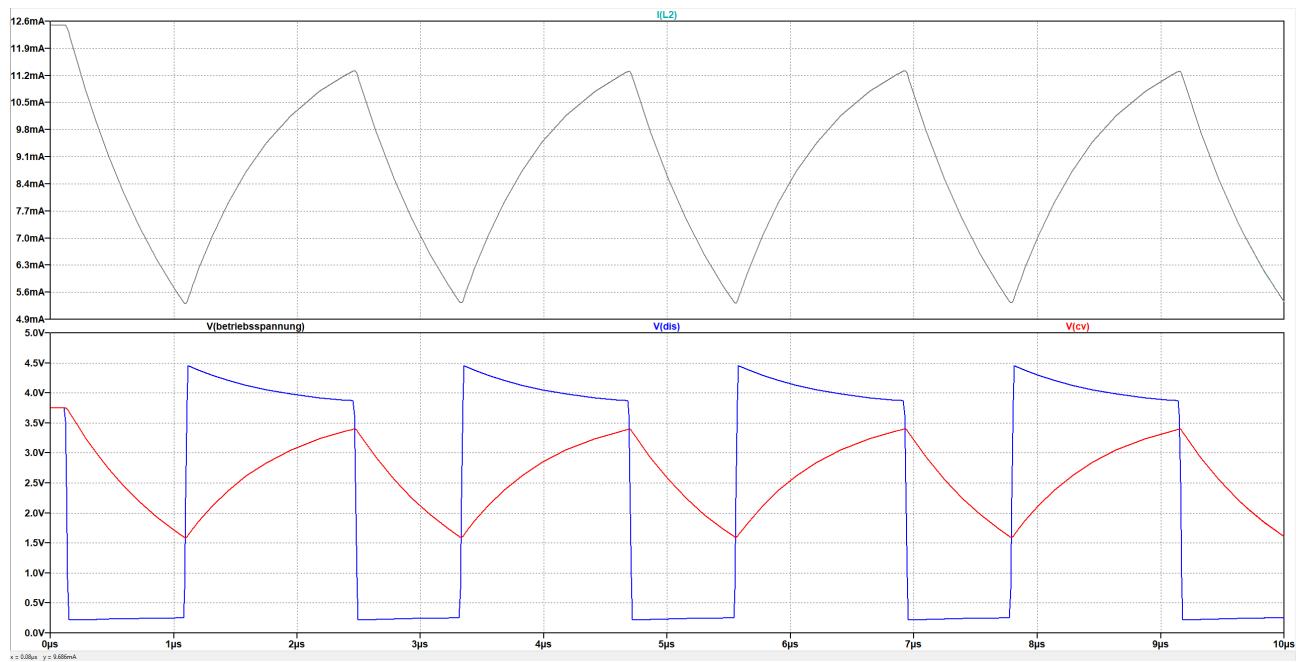


Abbildung 23: RL-Oszillator mit NE555 Zeitsignale

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  $L2 = 1 \text{ mH}$ .

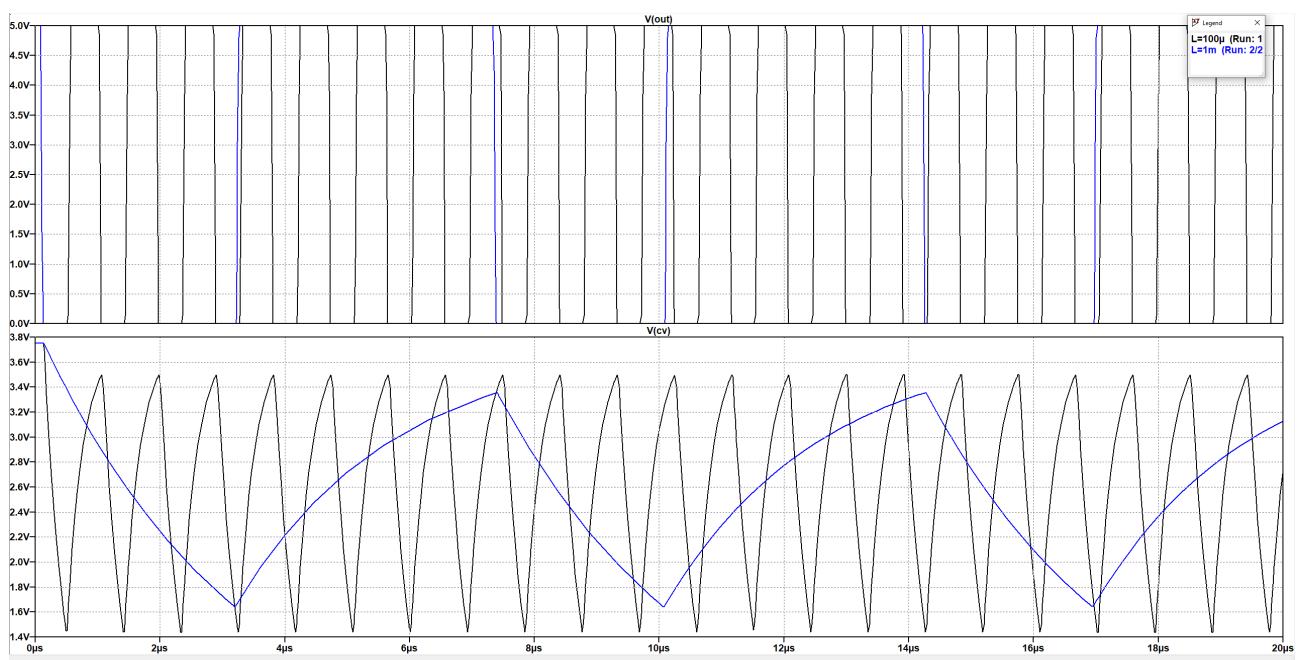


Abbildung 24: Vergleich des NE555-Oszillators bei unterschiedlichen L

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

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

### 9.3.2 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  $\Phi$ . Nach der Gleichung 5 reduziert sich danach die Induktivität mit zunehmender Frequenz.

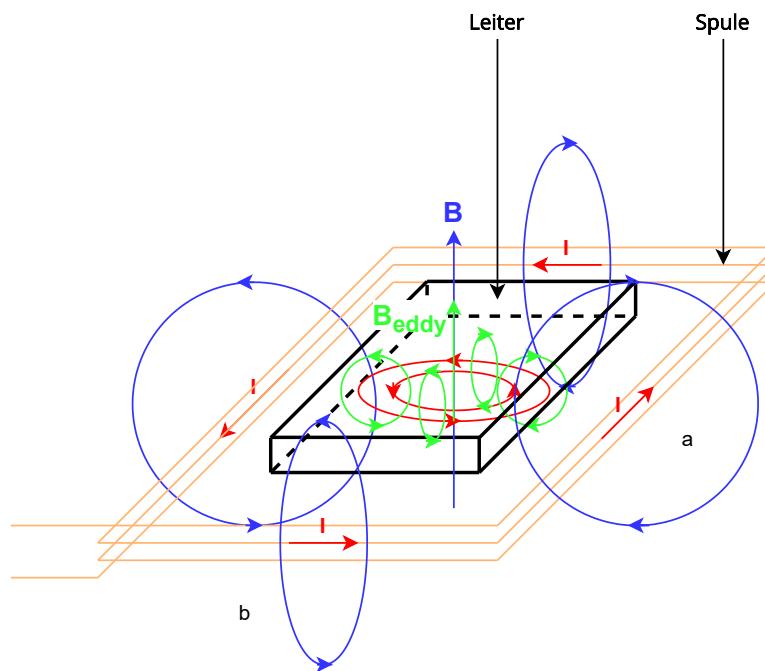


Abbildung 25: Eddy Currents in einem Leiter

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 Fahrzgerkennung liegen.

### 9.3.2.1 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.

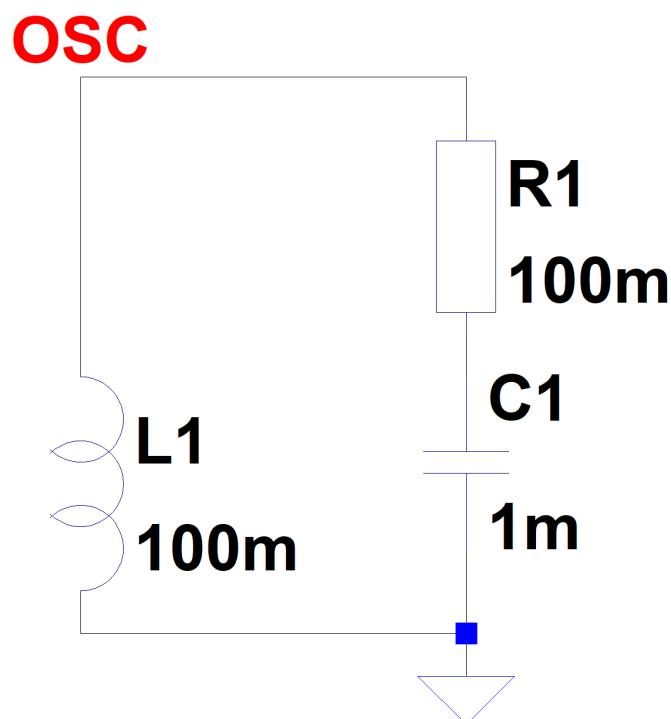


Abbildung 26: Parallelschwingkreis in LTSpice

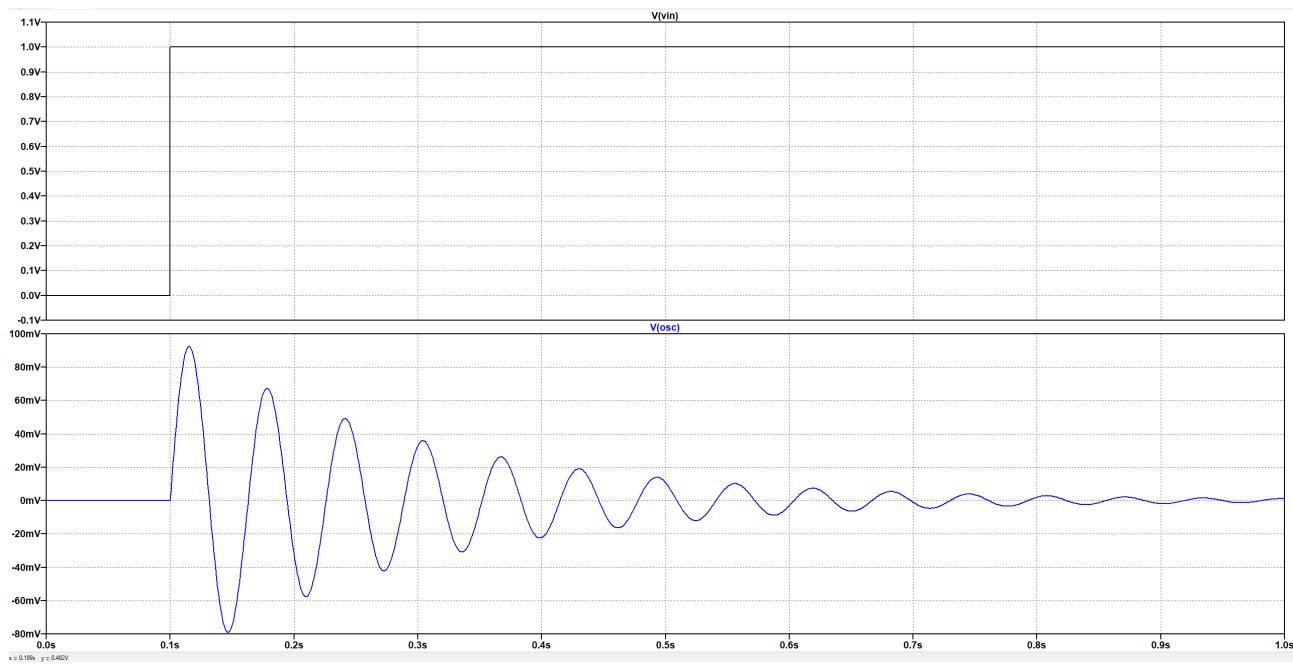


Abbildung 27: Gedämpfte Schwingung

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

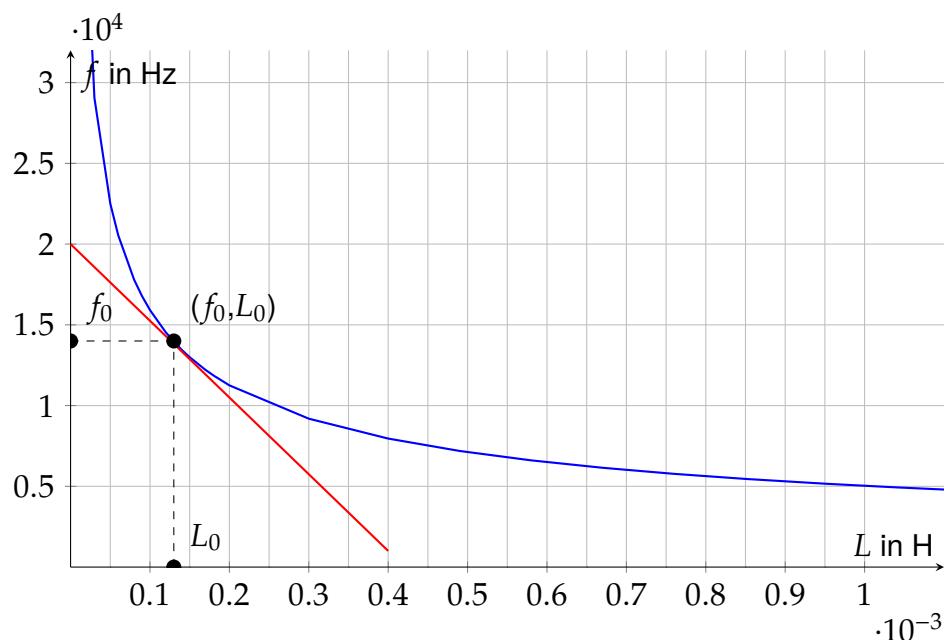
$C$  = Kapazität des Kondensators

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.

### 9.3.2.2 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  $\Delta L$  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  $f(L \pm \Delta L) = \Delta f \approx df$

### 9.3.2.3 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:

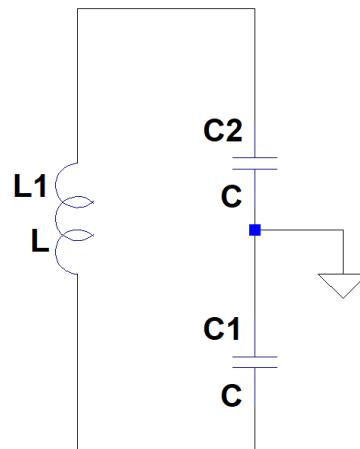


Abbildung 28: Colpitts LC-Glied

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  $C_{ges}$ .

$$C_{ges} = \frac{C_1 \cdot C_2}{C_1 + C_2} \quad (12)$$

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

$$f = \frac{1}{2 \cdot \pi \sqrt{L \cdot \left( \frac{C_1 \cdot C_2}{C_1 + C_2} \right)}} \quad (13)$$

Wobei

$f$  = Resonanzfrequenz des Colpitts-Oszillators

$L$  = 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  $180^\circ$  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.

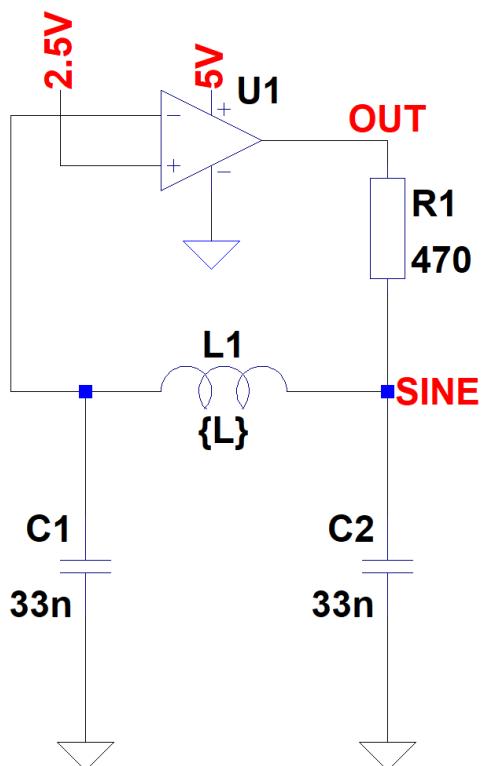


Abbildung 29: Aktiver Colpitts-Oszillator

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:

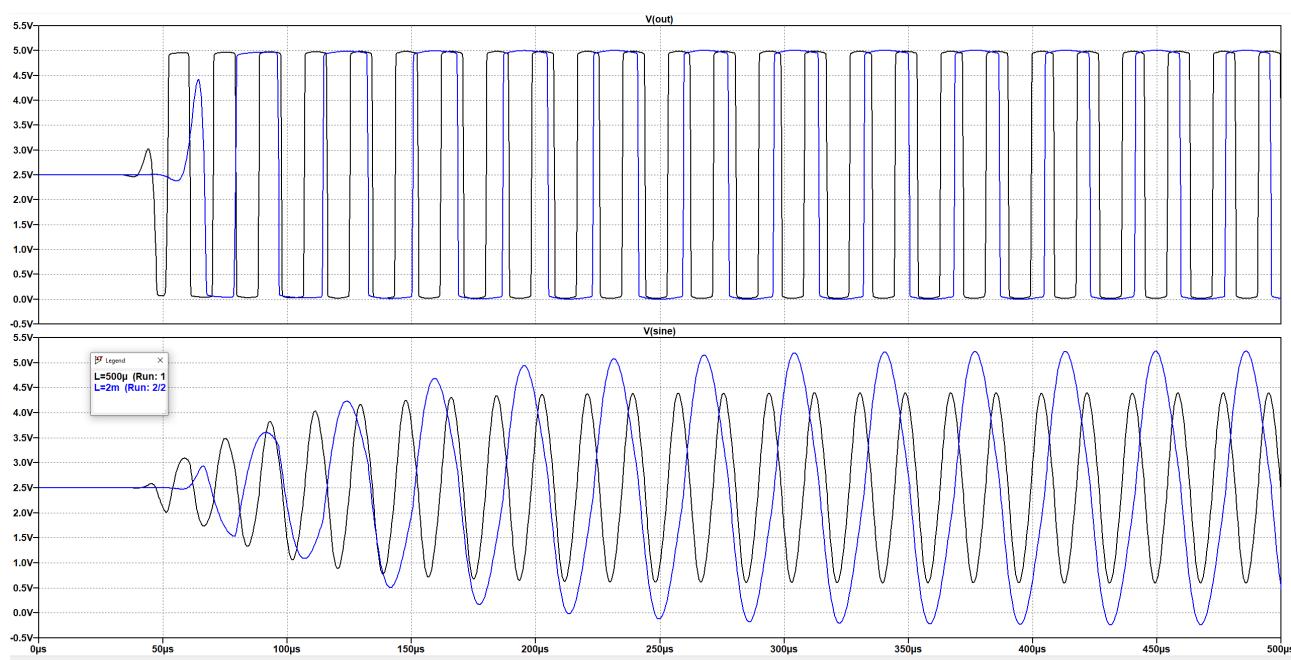


Abbildung 30: Zeitsignale Colpitts-Oszillator

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

## 9.4 RS485 Bussystem

### 9.4.1 Überblick

#### 9.4.1.1 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.

#### 9.4.1.2 UART

Unter UART versteht man eine 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 diese 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.

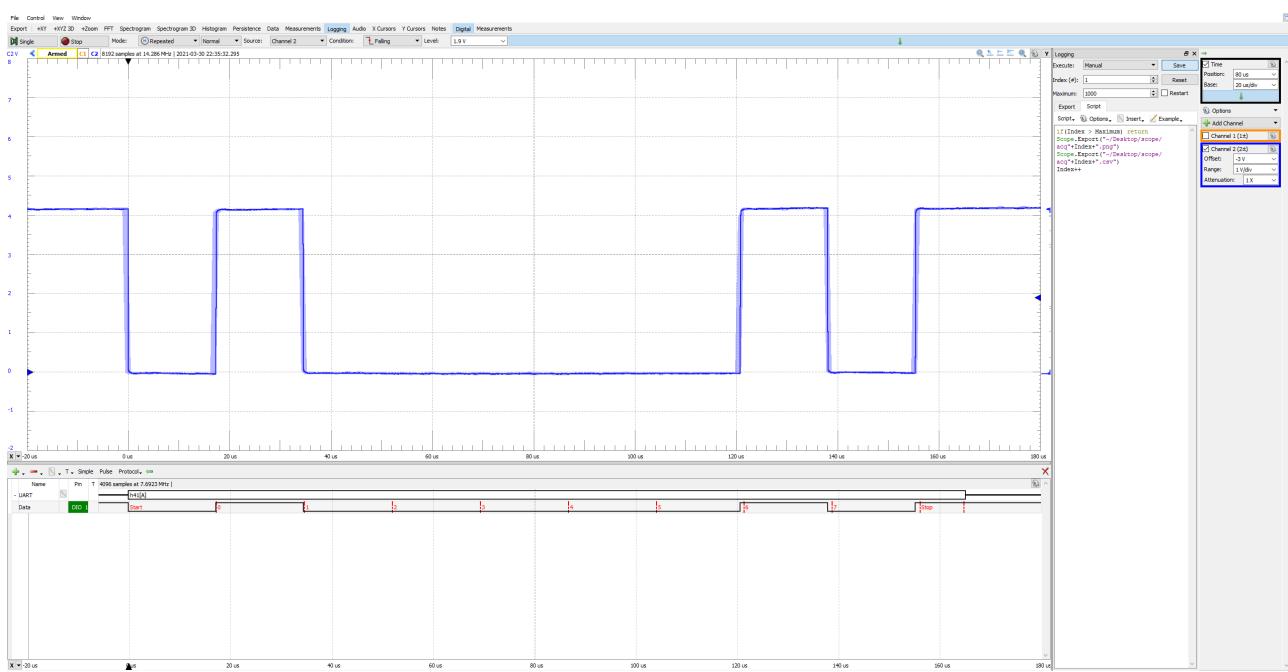


Abbildung 31: Beispiel einer UART Zeichenübertragung

#### 9.4.1.3 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.

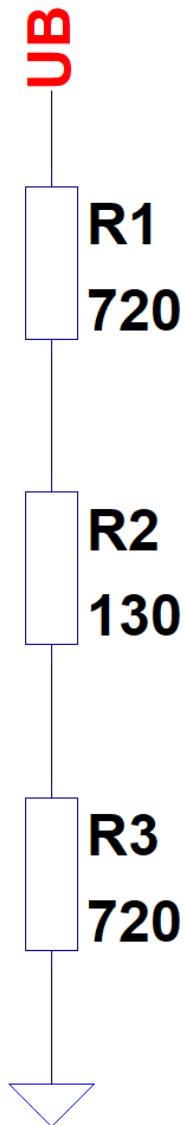


Abbildung 32: RS485 Widerstandsnetzwerk

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
$\overline{RE}$	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*

$X = \text{Don't care}$

$\text{High-Z} = \text{High impedance}$

\*Shutdown mode for MAX481/MAX483/MAX487

Abbildung 33: 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.

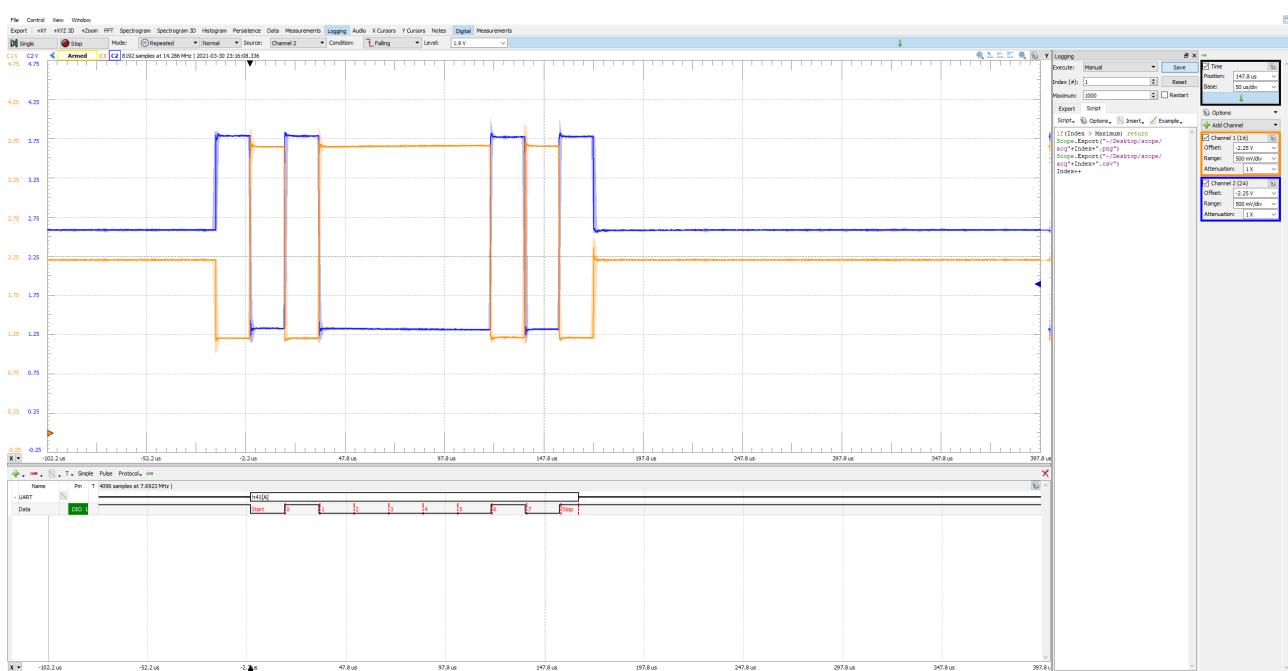


Abbildung 34: Beispiel einer RS485 Zeichenübertragung

#### 9.4.1.4 Aufbau

#### 9.4.2 Elektrische Spezifikation

#### 9.4.3 Implementation eines eigenen Protokolls

### 9.5 Mikrokontroller Slave-Geräte

#### 9.5.1 Überblick

#### 9.5.2 Atmega328PB

#### 9.5.3 Peripherie des Mikrocontrollers

##### 9.5.3.1 Spannungswandler

##### 9.5.3.2 RS485 Pegelwandler

##### 9.5.3.3 Digitale Ein- und Ausgänge

**9.5.4 Layout des Slave-Gerätes****9.5.5 Gehäuse****9.6 USB-Master****9.6.1 USB-Bussadapter Gerät****9.6.1.1 Überblick****9.6.1.2 FT232RL****9.6.1.3 Spannungsversorgung****9.6.1.4 USB-C Anschluss****9.6.1.5 Layout des Master-Geräts****9.6.1.6 Gehäuse****9.6.2 Master Programm****9.6.2.1 Benötigte Software****9.6.2.2 Adressvergabe****9.6.2.3 Frequenzauslesung****9.6.2.4 Auswertung****9.6.2.5 API-Post****9.6.3 RaspberryPi als Mastergerät****9.6.3.1 SSH Remote Zugriff**

### 9.6.3.2 Code Deployment

### 9.6.3.3 Unitest

## 10 Webinterface

### 10.1 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.

### 10.2 Verwendete Technologien

#### 10.2.1 HTML

HTML steht hierbei für Hypertext Markup Language und ist eine Auszeichnungssprache welche vom World Wide Web Consortium (W3C)<sup>13</sup> 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 HTML5<sup>14</sup> und wurde am 28. Oktober 2014 vom W3C vorgelegt.



Abbildung 35: HTML5 Logo

<sup>13</sup><https://www.w3.org>

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

### 10.2.1.1 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 36: Einfache HTML Seite von <https://www.w3.org/html/logo>

### 10.2.2 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 3<sup>15</sup>. Um die Funktion darzustellen wird die vorherige HTML Seite nun mit CSS ergänzt.

<sup>15</sup><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.

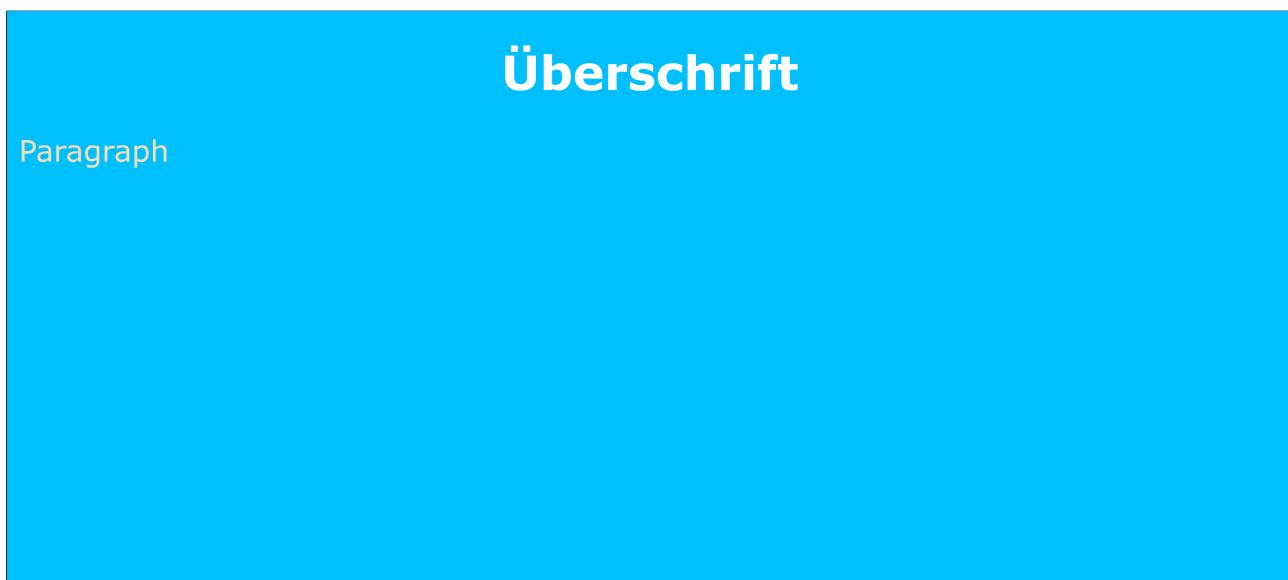


Abbildung 37: Einfache HTML Seite mit CSS

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 w3schools<sup>16</sup> zu finden.

### 10.2.3 JavaScript

Es folgt nun eine weitere sehr wichtige Technologie und die meist verwendete Programmiersprache überhaupt laut der Stack Overflow Developer Survey 2020<sup>17</sup>. 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 Frontend<sup>18</sup> mehr beschränkt, es möglich mit Frameworks wie Node.js auch Backend<sup>19</sup> Applikationen zu schreiben und somit ist es möglich Full-Stack<sup>20</sup>-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 International<sup>21</sup> veröffentlicht und die aktuellste Version ist **ECMA-262**.

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

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

<sup>18</sup>Grafische Benutzeroberfläche mit der der Benutzer interagiert

<sup>19</sup>Verarbeitung von Daten auf beispielsweise einem Server

<sup>20</sup>Front- und Backend

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

#### 10.2.4 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 8<sup>22</sup>.

#### 10.2.5 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.

#### 10.2.6 Laravel

Laravel<sup>23</sup> 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.

<sup>22</sup><https://www.php.net/docs.php>

<sup>23</sup><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

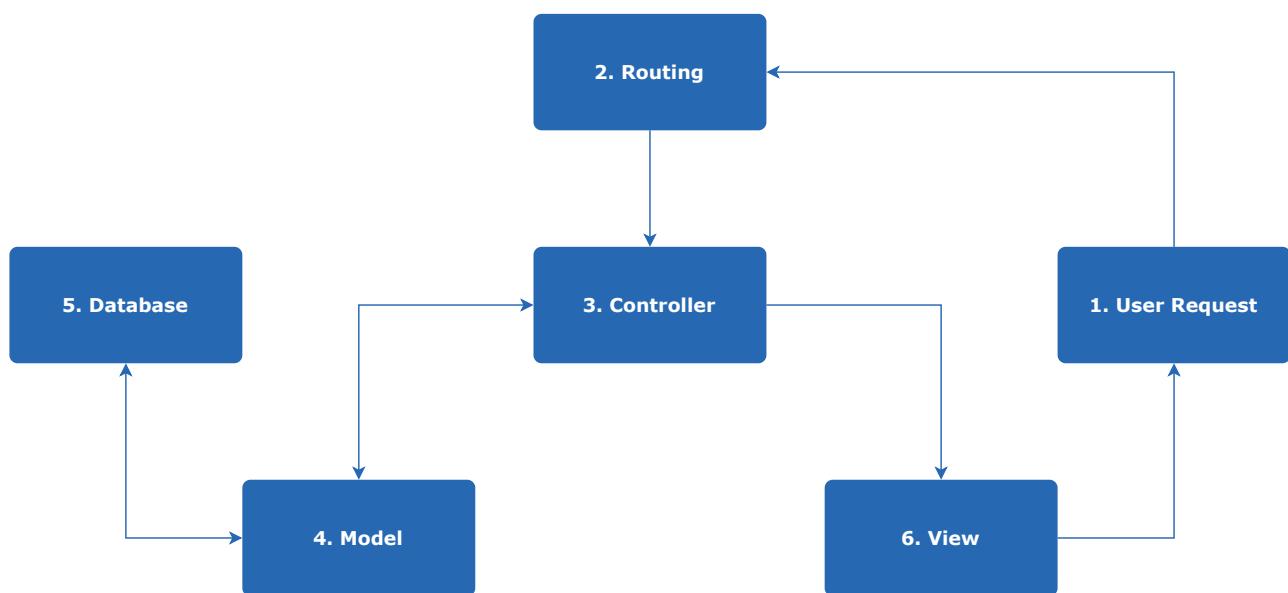


Abbildung 38: Laravel MVC Muster

#### 10.2.6.1 Routing

In Laravel erfolgt 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

#### 10.2.6.2 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.

#### 10.2.6.3 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

#### 10.2.6.4 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

#### 10.2.6.5 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.

#### 10.2.6.6 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.

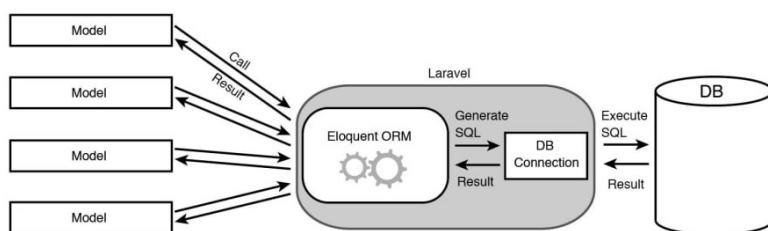


Abbildung 39: Eloquent ORM Workflow von <https://dev.to/xenoxdev/mastering-laravel-eloquent-orm-the-eloquent-journey-part-1-1571>

### 10.2.6.7 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.

## 10.3 Lokale Entwicklungsumgebung mit Laragon

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

### 10.3.1 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

### 10.3.2 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.

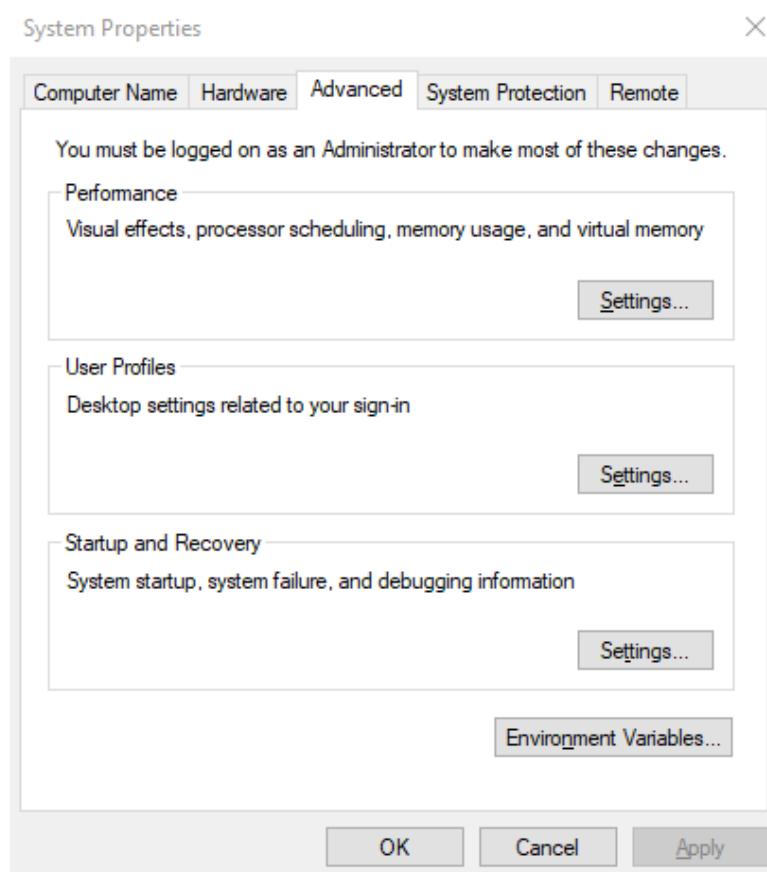


Abbildung 40: Advanced System Settings

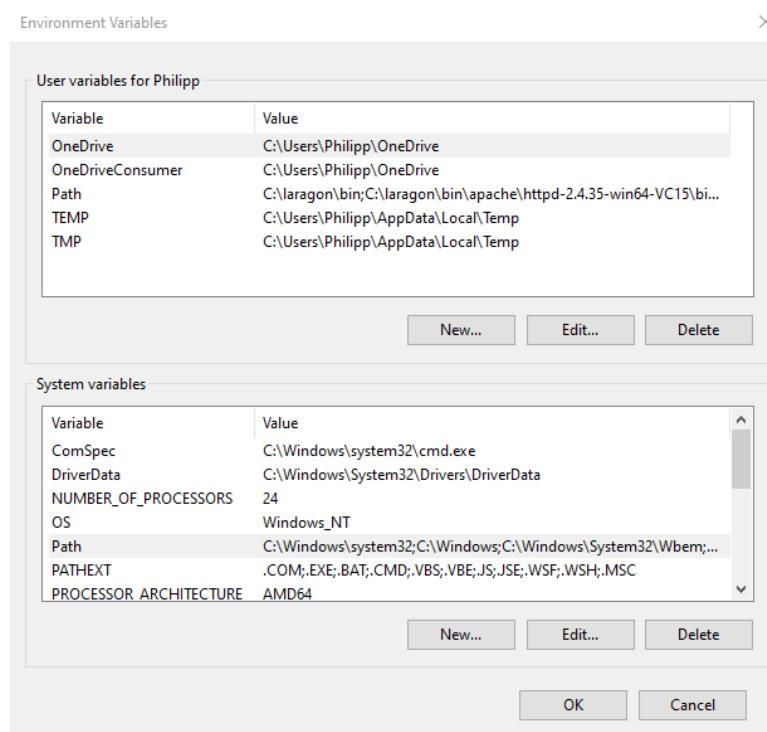


Abbildung 41: System Variables

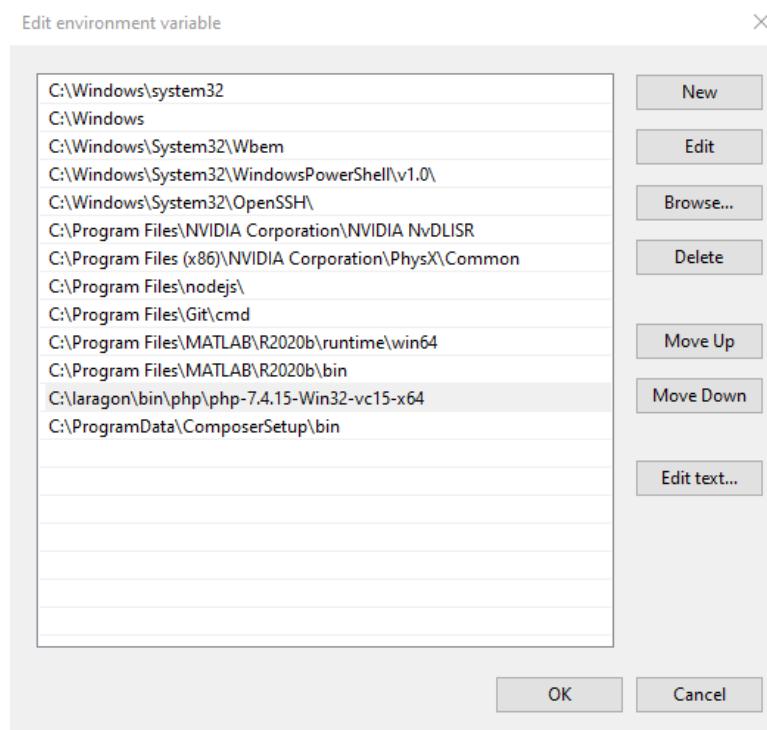


Abbildung 42: Environment Variables

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 43: PHP Version

Somit ist PHP korrekt konfiguriert.

### 10.3.3 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.

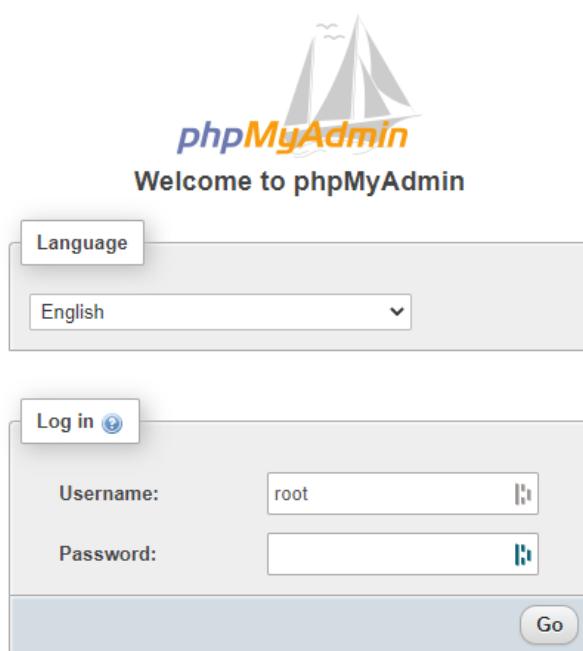


Abbildung 44: phpMyAdmin Webinterface

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

## 10.4 Lokale Entwicklungsumgebung mit WSL und Docker

### 10.4.1 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

### 10.4.2 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 45: WSL Feature Feature aktivieren

#### 10.4.2.1 2. Schritt: Virtual Machine Aktivieren

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

Code 46: Virtual Machine Feature aktivieren

Nach diesem Schritt ist ein Neustart des Computers notwendig.

#### 10.4.2.2 3. Schritt: Linux Kernel Update

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

#### 10.4.2.3 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 47: WSL 2 auswählen

#### 10.4.2.4 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>).

### 10.4.3 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.

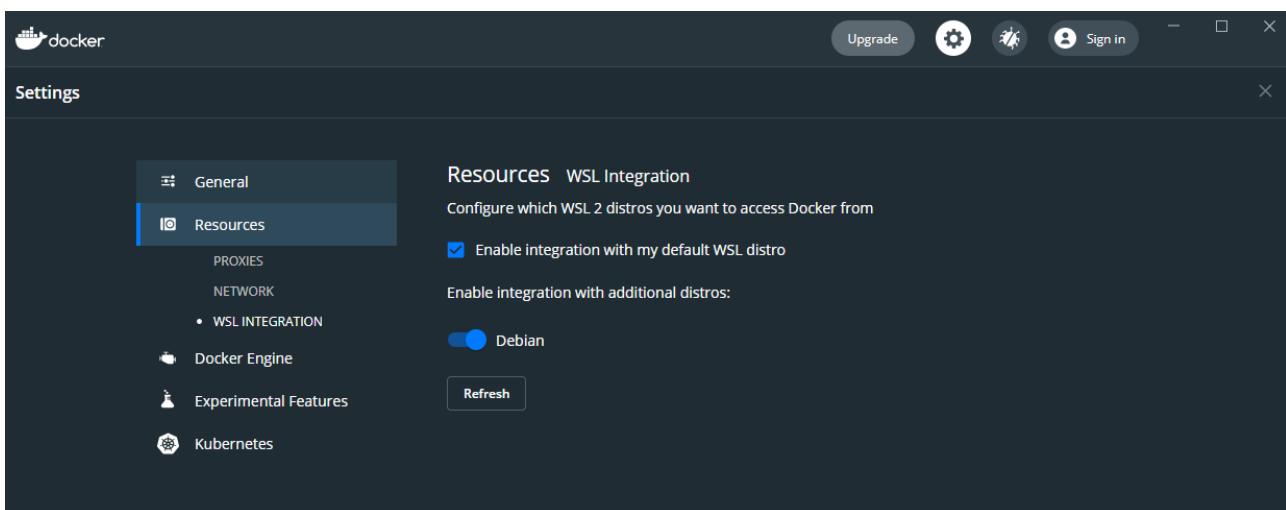


Abbildung 45: Docker WSL Integration

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 46: Docker Container Steuerung

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

## 10.5 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.

### 10.5.1 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

### 10.5.2 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 48: Respositories updaten

#### 10.5.2.1 Apache

Nun kann der Apache Web Server installiert werden.

```
1 apt install apache2
```

Code 49: 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>).

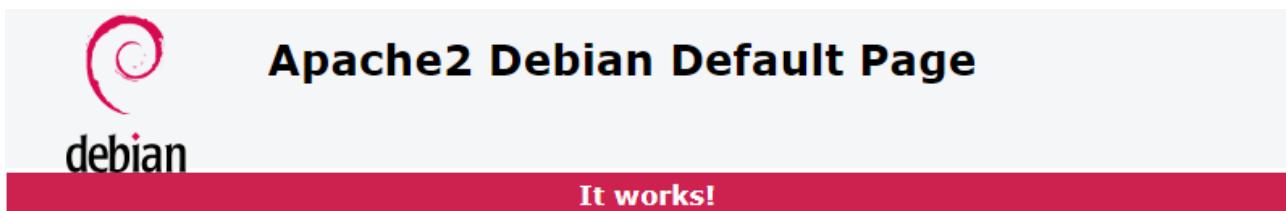


Abbildung 47: Debian Default Page

Erscheint die Debian Default Page ist Apache korrekt installiert.

### 10.5.2.2 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 50: PHP installieren

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

### 10.5.2.3 MariaDB

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

Code 51: Mariadb installieren

Nun muss MariaDB noch konfiguriert werden.

```
1 mysql_secure_installation
```

Code 52: 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 53: MariaDB konfiguration

#### 10.5.2.4 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 54: phpMyAdmin Download

Anschließend muss das Archiv entpackt werden.

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

Code 55: 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 56: phpMyAdmin Rechte und Verzeichnisse

Nun muss eine Konfigurations Datei erstellt werden und dort muss ein Blowfish Secret<sup>24</sup> 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 57: 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 58: phpMyAdmin Blowfish Secret und TempDir

Zuletzt muss der Apache Web Server konfiguriert werden.

<sup>24</sup>32 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 59: `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 60: 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.

#### 10.5.2.5 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 61: 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.

### 10.5.2.6 Installation von Composer

Die Installation von Composer gestaltet sich relativ einfach.

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

Code 62: 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 63: Composer Setup

### 10.5.3 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>).

#### 10.5.3.1 Git Setup

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

```
1 apt install git
```

Code 64: 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 65: 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.

#### 10.5.3.2 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 66: deploy.sh

### 10.5.3.3 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 67: serverdeploy.sh

#### 10.5.3.4 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 68: 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.

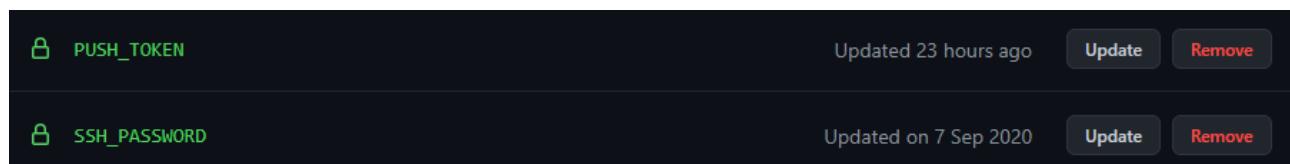


Abbildung 48: Action Secrets

```
deploy
succeeded 23 hours ago in 1m 10s

> ✓ Set up job 5s
> ✓ Build appleboy/ssh-action@master 5s
> ✓ Run actions/checkout@v2 3s
> ✓ Set up Node 0s
> ✓ Run npm install 20s
> ✓ Run npm run production 20s
> ✓ Commit built assets 3s
> ✓ Deploy to production 14s
> ✓ Post Run actions/checkout@v2 0s
> ✓ Complete job 0s
```

Abbildung 49: Github Action Übersicht

## 10.6 Grundlegender Aufbau Frontend

Da das Ziel ist, dass das Frontend<sup>25</sup> 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.

### 10.6.1 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.

#### 10.6.1.1 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.

#### 10.6.1.2 Components erstellen

<sup>25</sup>Präsentationsebene in Form der grafischen Benutzeroberfläche

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 69: primary.blade.php

Im Code 69 ist eine Variable mit dem Namen `slot` verwendet worden. Diese Variable wird später beim verwenden automatisch mit dem Inhalt zwischen dem HTML Element ersetzt.

#### 10.6.1.3 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 70: Verwendung eines Button Components

#### 10.6.1.4 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 71: Modularer Button Component

Somit ist dieser Button Component nun vollständig Modular.

### 10.6.2 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

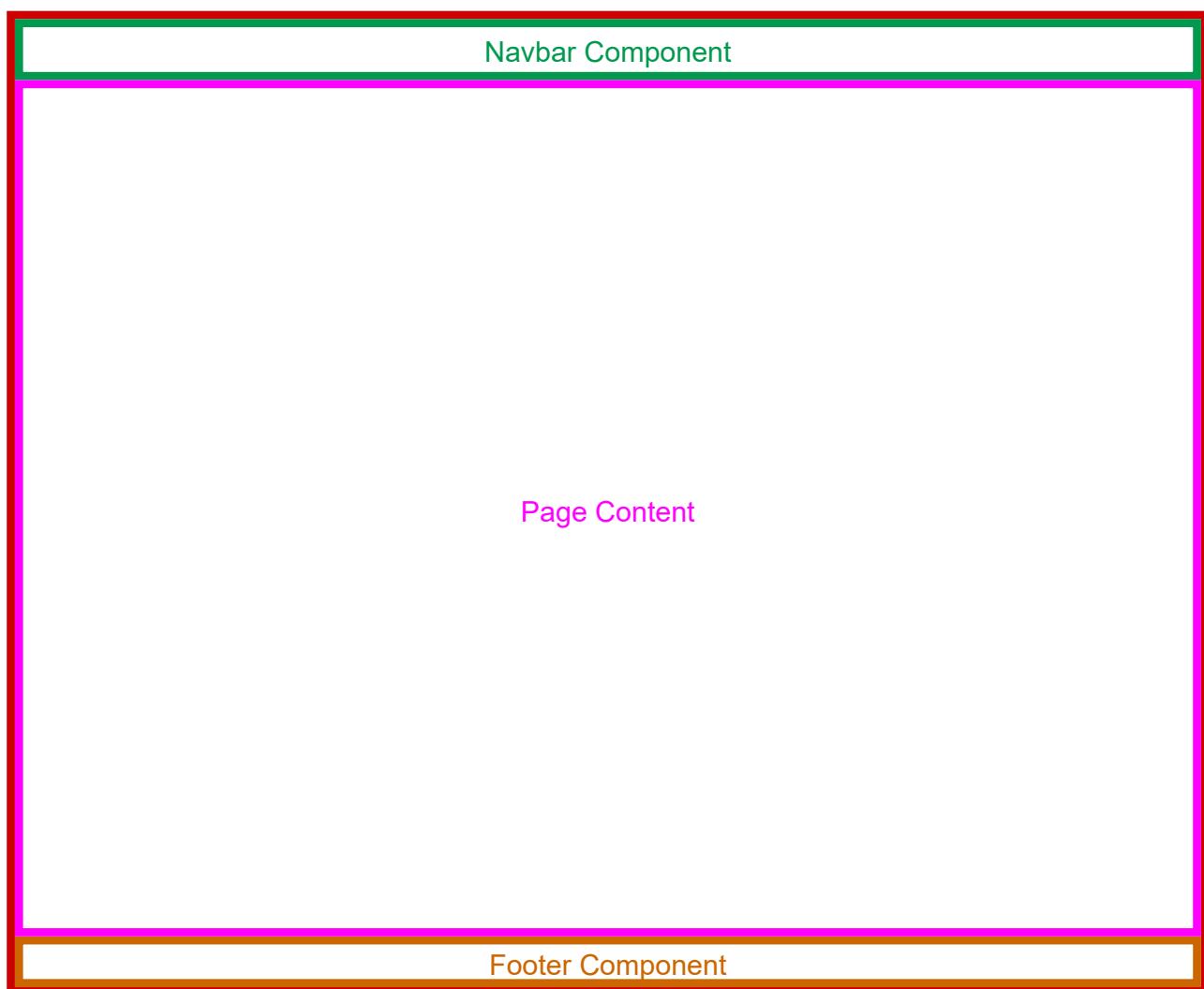


Abbildung 50: App Layout

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

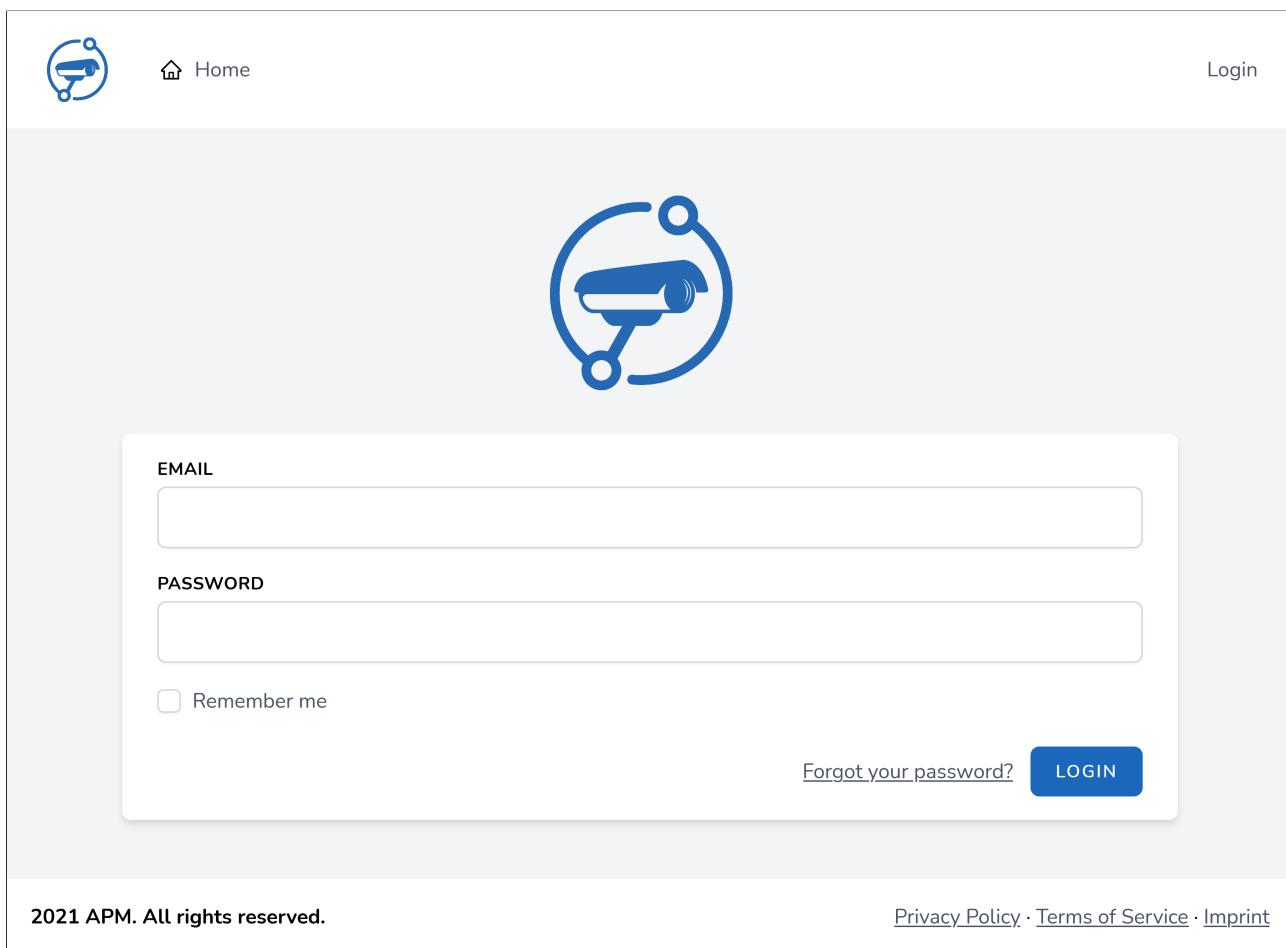


Abbildung 51: Login Seite mit App Layout

In der Abbildung 51 wird das App Layout verwendet. Dabei sind die drei Components erkennbar.

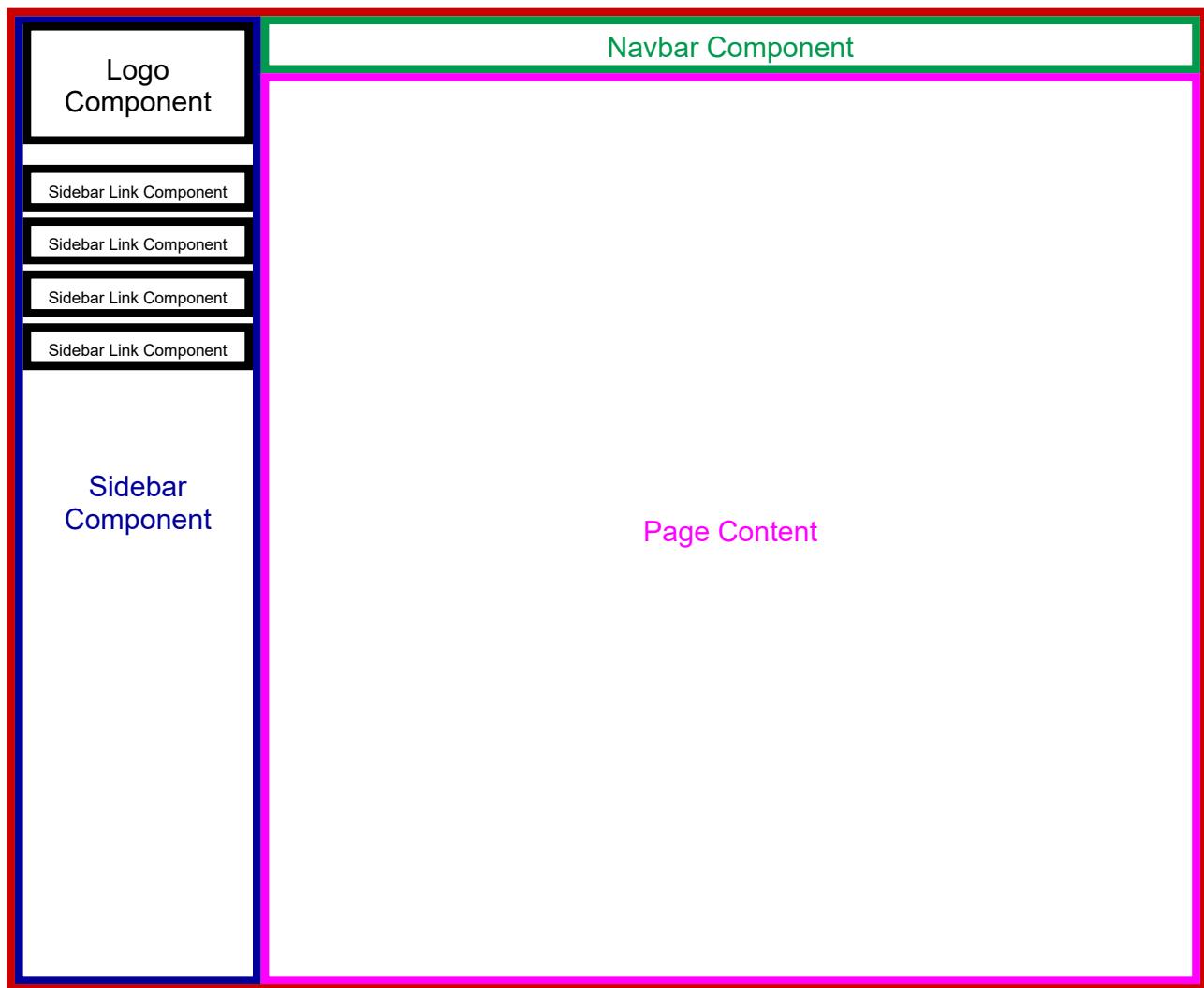


Abbildung 52: Admin Layout

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 Navbar<sup>26</sup>.

<sup>26</sup>Navigation Bar engl. für Navigationsleiste

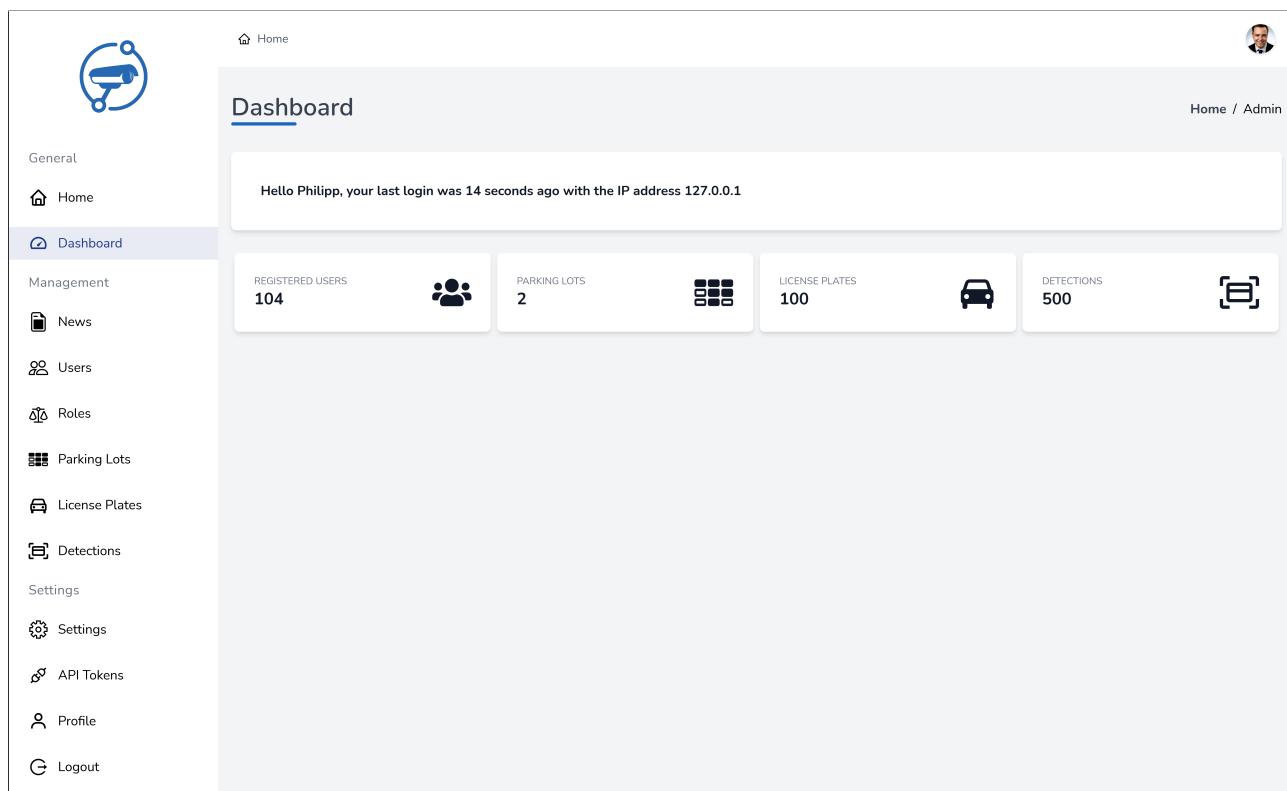


Abbildung 53: Dashboard mit Admin Layout

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.

### 10.6.3 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 72: 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 54: Navbar Variante 1



Abbildung 55: 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>
@endif

```

Code 73: 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 56: Navbar Variante 3

#### 10.6.4 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-Symbol<sup>27</sup> in der Navigationsleiste geöffnet, somit wird wichtiger Platz gespart.

Für das dynamische Verhalten der Sidebar wird JavaScript verwendet. Es wird nicht Vanilla<sup>28</sup> JavaScript verwendet, sondern AlpineJS<sup>29</sup>. Dabei folgt das Lightweight Framework dem gleichen

<sup>27</sup>Drei waagrechte Striche ähneln einem Hamburger

<sup>28</sup>Bedeutet so viel wie ohne Zusätze

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

Prinzip wie TailwindCSS nur eben für JavaScript.

```
1 <div x-data="{ sidebarOpen: false }" class="flex h-screen bg-gray-200
  ↳ z-0">
2 </div>
```

Code 74: Sidebar mit AlpineJS

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 75: 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.

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

Code 76: Sidebar Link

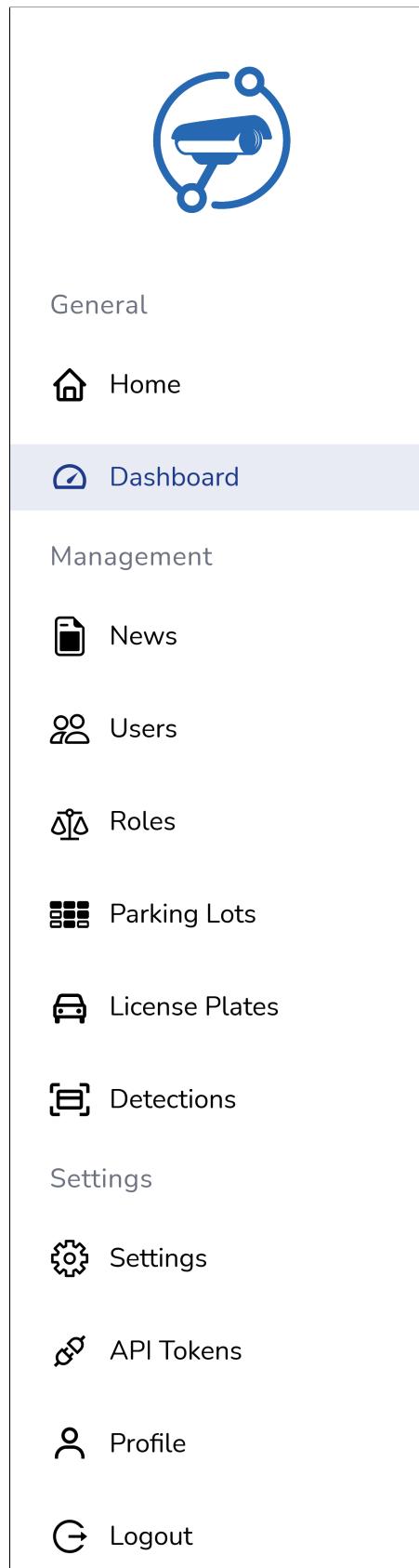


Abbildung 57: Sidebar Desktop

### 10.6.5 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 58: Footer

## 10.7 Funktionen

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

### 10.7.1 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 IP-Adresse.

Um diese Daten abzugreifen 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 77: AuthenticatedSessionController.php

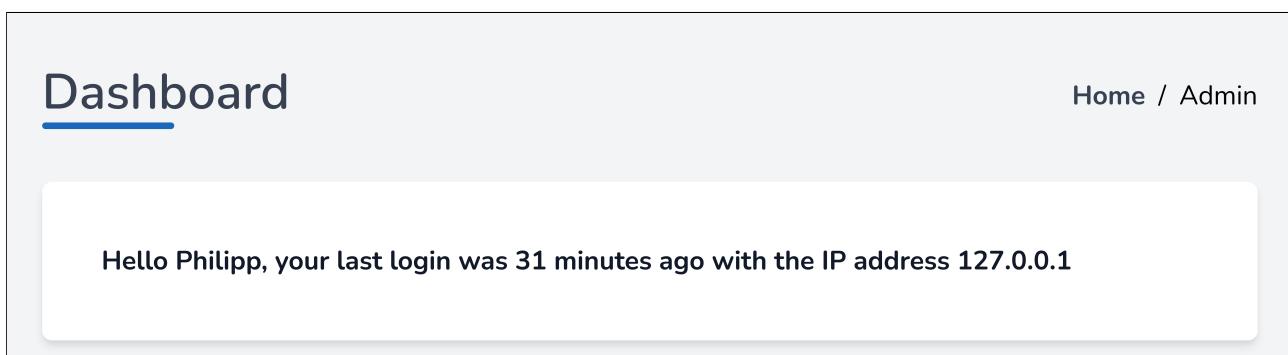


Abbildung 59: Dashboard Sicherheitsnachricht

Neben der Sicherheitsnachricht werden einige statistische Zahlen über folgendes dargestellt:

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

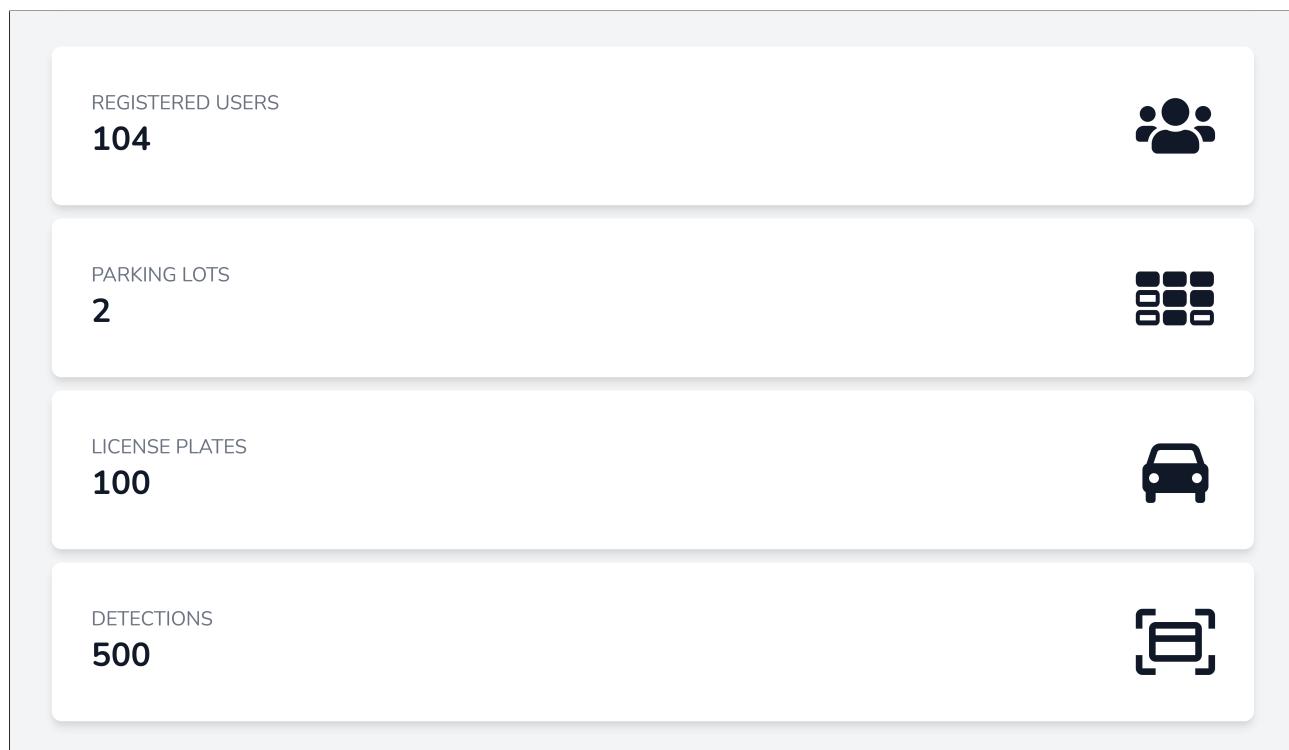


Abbildung 60: Dashboard Statistik

## **10.7.2 Login- und Registersystem**

### **10.7.3 News**

### **10.7.4 Benutzerverwaltung**

### **10.7.5 Rechte- und Rollenverwaltung**

### **10.7.6 Parkplatzverwaltung**

### **10.7.7 Kennzeichenverwaltung**

### **10.7.8 Erkennungsverlauf**

### **10.7.9 Seiten Einstellungen**

### **10.7.10 API Schlüssel**

### **10.7.11 Displays**

### **10.7.12 Notifications**

### **10.7.13 Profil**

### **10.7.14 Lokalisierung**

## **10.8 Performance und Sicherheit**

### **10.8.1 Form Validation**

### **10.8.2 Authentifizierung**

### **10.8.3 Authentisierung**

### **10.8.4 SQL Injection**

### **10.8.5 Cross-Site-Scripting**

### **10.8.6 Lighthouse**

## 11 Zusammenfassung und Ausblick

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.

## 12 Anhang

## Abbildungsverzeichnis

1	Vor Bilateraler Filterung . . . . .	18
2	Nach Bilateraler Filterung . . . . .	18
3	Graustufenbild . . . . .	19
4	Binäres Bild nach Thresholding . . . . .	19
5	Binäres Bild nach Thresholding . . . . .	20
6	Nach Erosion . . . . .	20
7	Ablaufdiagramm der Kennzeichenerkennung . . . . .	24
8	Beispiel einer Anwendung von Matplotlib . . . . .	30
9	GPIO-Pins des Raspberry Pi . . . . .	35
10	Visualisierung mit Matplotlib . . . . .	44
11	Visualisiertes Ergebnis der Kennzeichenerkennung . . . . .	45
12	Raspberry Pi . . . . .	47
13	Raspberry Pi mit Kamera . . . . .	48
14	Fehler bei Konturerkennung . . . . .	51
15	Resultat mit maschinellem Lernen . . . . .	51
16	Ablauf von WPOD-NET . . . . .	53
17	Beispiel eines Zeichens aus dem Datensatz . . . . .	54
18	Verlauf des Modell Trainings . . . . .	57
19	Installation der Spule . . . . .	60
20	LTSpice Blockbild zweier RL-Glieder . . . . .	62
21	Stromkurven zweier RL-Glieder . . . . .	63
22	RL-Oszillator mit NE555 . . . . .	64
23	RL-Oszillator mit NE555 Zeitsignale . . . . .	65
24	Vergleich des NE555-Oszillators bei unterschiedlichen L . . . . .	66
25	Eddy Currents in einem Leiter . . . . .	67
26	Parallelschwingkreis in LTSpice . . . . .	68
27	Gedämpfte Schwingung . . . . .	69
28	Colpitts LC-Glied . . . . .	71
29	Aktiver Colpitts-Oszillator . . . . .	72

30	Zeitsignale Colpitts-Oszillator . . . . .	73
31	Beispiel einer UART Zeichenübertragung . . . . .	75
32	RS485 Widerstandsnetzwerk . . . . .	76
33	Logische Tabelle für $U_{AB}$ . . . . .	77
34	Beispiel einer RS485 Zeichenübertragung . . . . .	78
35	HTML5 Logo . . . . .	81
36	Einfache HTML Seite von <a href="https://www.w3.org/html/logo">https://www.w3.org/html/logo</a> . . . . .	83
37	Einfache HTML Seite mit CSS . . . . .	85
38	Laravel MVC Muster . . . . .	88
39	Eloquent ORM Workflow von <a href="https://dev.to/xenoxdev/mastering-laravel-eloquent-orm-the-eloquent-journey-part-1-1571">https://dev.to/xenoxdev/mastering-laravel-eloquent-orm-the-eloquent-journey-part-1-1571</a> . . . . .	94
40	Advanced System Settings . . . . .	96
41	System Variables . . . . .	97
42	Environment Variables . . . . .	97
43	PHP Version . . . . .	98
44	phpMyAdmin Webinterface . . . . .	98
45	Docker WSL Integration . . . . .	101
46	Docker Container Steuerung . . . . .	101
47	Debian Default Page . . . . .	103
48	Action Secrets . . . . .	112
49	Github Action Übersicht . . . . .	113
50	App Layout . . . . .	116
51	Login Seite mit App Layout . . . . .	117
52	Admin Layout . . . . .	118
53	Dashboard mit Admin Layout . . . . .	119
54	Navbar Variante 1 . . . . .	120
55	Navbar Variante 2 . . . . .	120
56	Navbar Variante 3 . . . . .	121
57	Sidebar Desktop . . . . .	123
58	Footer . . . . .	124
59	Dashboard Sicherheitsnachricht . . . . .	125

60 Dashboard Statistik . . . . . 126

## Tabellenverzeichnis

## Codeverzeichnis

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

30	Bildverarbeitungsalgorithmen . . . . .	42
31	Anwendung einer Visualisierung mit Matplotlib . . . . .	43
32	predict_from_model . . . . .	44
33	Anwendung der Zeichenerkennung . . . . .	45
34	Senden der Ergebnisse an die Datenbank . . . . .	46
35	Erstellen des Modells basierend auf MobileNetV2 . . . . .	55
36	Trainieren des Modells und Einstellungen anpassen . . . . .	56
37	index.html . . . . .	82
38	style.css . . . . .	84
39	web.php . . . . .	89
40	example.blade.php . . . . .	90
41	UserController.php . . . . .	91
42	create_users_table.php . . . . .	93
43	Eloquent Query . . . . .	94
44	Raw SQL Query . . . . .	94
45	WSL Feature Feature aktivierens . . . . .	99
46	Virtual Machine Feature aktivieren . . . . .	99
47	WSL 2 auswählen . . . . .	100
48	Respositorys updaten . . . . .	102
49	Apache installlieren . . . . .	103
50	PHP installlieren . . . . .	103
51	Mariadb installlieren . . . . .	104
52	MariaDB Secure Installation . . . . .	104
53	MariaDB konfiguration . . . . .	104
54	phpMyAdmin Download . . . . .	104
55	phpMyAdmin Entpacken . . . . .	105
56	phpMyAdmin Rechte und Verzeichnisse . . . . .	105
57	phpMyAdmin Konfigurationsdatei erstellen . . . . .	105
58	phpMyAdmin Blowfish Secret und TempDir . . . . .	105
59	phpmyadmin.conf . . . . .	106
60	Virtual Host aktivieren . . . . .	106

61	apm.conf . . . . .	107
62	Download Composer Installer . . . . .	108
63	Composer Setup . . . . .	108
64	Git Installation . . . . .	108
65	Git Remote Origin . . . . .	109
66	deploy.sh . . . . .	109
67	serverdeploy.sh . . . . .	110
68	main.yml . . . . .	112
69	primary.blade.php . . . . .	114
70	Verwendung eines Button Components . . . . .	114
71	Modularer Button Component . . . . .	115
72	Ausschnitt 1 navigation.blade.php . . . . .	120
73	Ausschnitt 2 navigation.blade.php . . . . .	121
74	Sidebar mit AlpineJS . . . . .	122
75	Sidebar Header . . . . .	122
76	Sidebar Link . . . . .	122
77	AuthenticatedSessionController.php . . . . .	125

## 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

## Literaturverzeichnis

Einstein, Albert. »Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]«. In: *Annalen der Physik* 322.10 (1905), S. 891–921. doi: <http://dx.doi.org/10.1002/andp.19053221004>.

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>.