Aperta - Das smarte Garagentor

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informationstechnologie mit
Ausbildungschwerpunkt Medientechnik

Eingereicht von:
Benjamin Golic
David Hauser
Simon Koll

Betreuer:
Prof. Christian Aberger

Leonding, April 2022

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 bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022 Benjamin Golic & David Hauser & Simon Koll

Abstract

APERTA is a garage door system that was developed by Benjamin Golic, David Hauser and Simon Koll as part of their diploma thesis. The system can be retrofitted to electric garage doors regardless of the manufacturer and enables entry into the world of the Internet of Things, the IOT. Depending on the configuration, the system consists of up to 3 components, each of which represents an access option to the garage. These are a classic numpad, an RFID reader, and a camera, which is used for license plate recognition. In the future, these can be expanded to include other or further authentication options. A web dashboard is used to manage the combinations of the numpad , NFC card details or license plates. These can be provided with a time validity range. There is also an online store where the product can be configured and spare parts or expansion modules can be purchased. The dashboard, which was implemented using Angular, communicates with the Node-JS server as the backend. It acts as the core, passes the information to the MongoDB database and is available for requests from the Raspberry Pi, which is responsible for the hardware.

Abbildung 1: Rendered Prototyp

Zusammenfassung

APERTA ist ein Garagentorsystem, welches von Benjamin Golic, David Hauser und Simon Koll im Rahmen der Diplomarbeit entwickelt wurde. Das System ist herstellerunabhängig bei elektrischen Garagentoren nachrüstbar und ermöglicht den Einstieg in die Welt des Internet der Dinge, dem IOT. Das System besteht, je nach Konfiguration, aus bis zu 3 Komponenten, welche je eine Zutrittsmöglichkeit zur Garage darstellen. Es handelt sich hierbei um ein klassisches Nummernfeld, einem RFID-Lesegerät, und einer Kamera, welche für eine Kennzeichenerkennung genutzt wird. Zukünftig können diese noch um andere oder weitere Möglichkeiten der Authentifizierung erweitert werden. Über ein Web-Dashboard werden die Nummernfeldkombinationen, NFC-Kartendetails oder Kennzeichen verwaltet. Diese können mit einem zeitlichen Gültigkeitsbereich versehen werden. Darüber hinaus gibt es einen Onlineshop, in welchem das Produkt konfiguriert werden kann sowie Ersatzteile oder Erweiterungsmodule erworben werden können. Das Dashboard, welches mithilfe von Angular umgesetzt wurde, kommuniziert mit dem Node-JS Server als Backend. Dieser fungiert als Herzstück und reicht die Informationen an die MongoDB-Datenbank weiter und steht für Anfragen des Raspberry Pi, welcher für die Hardware zuständig ist, zur Verfügung.

Abbildung 2: Gerenderter Prototyp

Inhaltsverzeichnis

1 Einleitung

1.1 Problemsituation [SK]

Autos sind aus dieser Welt nicht mehr wegzudenken. Alleine in Österreich sind mehr als 5 Millionen Personenkraftwagen zugelassen. Dieser Bestand wuchs nach Angaben der Statistik Austria seit mehr als 15 Jahren kontinuierlich an.[1] Um die Sicherheit der Insassen gewährleisten zu können, wird empfohlen, das Fahrzeug in einer Garage oder einem überdachten Gebiet abzustellen. Dort sind Umwelteinflüsse wie Hagel keine große Gefahr mehr. Neue Garagen haben oftmals elektrische Tore, die mit einem Nummernfeld oder einem Handsender aus dem Auto selbst geöffnet werden können. Diese Handsender können jedoch bei der Anfahrt an die Garage in der Eile nicht gefunden werden, oder die Batterie kann sich unbemerkt entleeren. Falls zudem kein Nummernfeld oder ähnliches vorhanden ist, gibt es bei Verlust des Handsenders keine Möglichkeit, die Garage zu öffnen. Somit muss man aus dem Auto aussteigen und die Notentriegelung des Systems betätigen. Mit der immer weiter voranschreitenden Vernetzung der nahen Umwelt, wie Haustüren mit Fingerabruck-Zugangskontrolle oder mit dem Smartphone bedienbaren Jalousien, kommt mit APERTA das IOT in die Garage. APERTA ist ein Komplettsystem, welches bei Garagen mit elektrischem Tor nachgerüstet werden kann. Es besteht die Möglichkeit, mithilfe eines Nummernfeldes oder einer NFC-Karte das Garagentor wie gewohnt zu öffnen. Was APERTA aber auszeichnet ist eine integrierte Kennzeichenerkennung, bei der man keinen Handsender oder ähnliches mehr benötigt. Das Kennzeichen wird über das Web-Dashboard eingegeben, und das Tor kann dann bei Annäherung an das Tor dieses erkennen und steuert den Toröffnungsmechanismus an. Dazu wird ein Relais verwendet, welches wie handelsübliche Handschalter an der Innenseite der Garage den Steuerstromkreis der Garage schließt.

1.2 Ziele[SK]

Das Team hat sich vor Entwicklungsbeginn einige Ziele Gesteckt, welche das Projekt erfüllen muss, um einen Mehrwert für potentielle Kunden zu bieten. Zu diesen Zielen zählen:

- *Herstellerunabhängigkeit:* APERTA soll bei jedem Garagentor mit bereits verbautem Motor nachrüstbar sein. So können mehr potentielle Käufer angesprochen werden.
- *Modularität:* Aufgrund der vielen Möglichkeiten kann APERTA für manche Käufer in der Vollausstattung nicht geeignet sein. Das Produkt soll daher im Onlineshop konfiguriert werden können, sodass bestimmte Komponenten entfernt werden können. Diese sollen nachträglich eingebaut werden können.
- *Übersichtliche Verwaltung:* Um den Nutzer zu unterstützen, soll die Verwaltung von Nummernfeldkombinationen, NFC-Details und Kennzeichen einfach und schnell funktionieren. Der Käufer soll mithilfe von Texteingaben neue Einträge hinzufügen können und diese mit einem Knopfdruck wieder entfernen können.
- *Intuitiver Bestellvorgang:* Um die Erfahrung für den Käufer von Anfang an gut zu gestalten, soll ein optisch ansprechender Onlineshop der erste Kontakt mit dem Produkt sein.

2 Technologien

2.1 Auswahl der Technologie - Client

2.1.1 Unterschied Framework und Library

2.1.2 Technologie zur Entwicklung des Frontends

2.1.3 Angular

2.1.4 React

2.2 Auswahl der Technologie - Datenbank[SK]

Die Datenbank spielt für das Projekt eine sehr wichtige Rolle, daher wurden hier folgende Kriterien aufgestellt:

- Das Projekt besteht aus Hard- und Software, die Daten sowohl abspeichern als auch abfragen. Das kann je nach Anwendungsfall unterschiedlich sein.
- Vom Dashboard können beispielsweise Kennzeichen mit Gültigkeitsdauer, aber auch nur einfache NFC-Codes gesendet werden.
- In der Zukunft soll die Möglichkeit bestehen, weitere Zutrittsmöglichkeit hinzuzufügen. Daher muss sich die Datenbank der sich ändernden Datenstruktur anpassen können.

2.2.1 Relationale Datenbanken

Das Team befand sich nun vor der Entscheidung, ein relationales Datenbanksystem zu verwenden, oder ein nicht relationales Datenbanksystem. Die größten Unterschiede hierbei sind, dass bei relationalen SQL-Datenbanken den gespeicherten Daten Tabellen vorgegeben sind, das sogenannte Schema. Das ist bei NoSQL-Datenbanken ebenfalls möglich, jedoch optional. Relationale Datenbanken verfolgen das ACID-Prinzip. ACID steht für

- *Atomicity*

  Alle Änderungen der Datenbank werden als einzige Operation verarbeitet. Entweder werden alle Änderungen wie Inserts, Updates usw. durchgeführt, oder keine davon.

- *Consistency*

  Zu Beginn und zum Ende jeder Transaktion sind die Daten konsistent. Wenn man als Beispiel eine Geldüberweisung darstellt, ist bei "Consistency"die Gesamtsumme der Geldmittel auf beiden Konten am Anfang und am Ende jeder Transaktion gleich.

- *Isolation*

  Andere Transaktionen haben keine Einsicht in die Transaktion. Isolation bedeutet also, dass parallel laufende Transaktionen sich wie serialisierte verhalten.

- *Durability*

  Die Daten bleiben nach Ende der Transaktion bestehen und auch bei einem kompletten Systemausfall nicht revidiert.

[2]

2.2.2 MongoDB

Das Team entschied sich für eine der bekanntesten NoSQL-Datenbanken, **MongoDB**. Diese bietet einige Vorteile gegenüber den relationalen SQL-Datenbanken:

- *Skalierbarkeit:* MongoDB Datenbanken zeichnen sich durch ihre ausgezeichnete horizontale Skalierbarkeit aus. Horizontale Skalierbarkeit bedeutet, dass die Datenbank sich problemlos über mehrere Server verteilen kann, ohne die Funktionsfähigkeiten zu beeinträchtigen.

- *Verfügbarkeit*

- *Flexibilität*

[3]

2.3 Auswahl der Technologie - Kennzeichenerkennung[SK]

Das Herz von APERTA ist die Kennzeichenerkennung. Dieses Alleinstellungsmerkmal separiert das Projekt von möglicher Konkurrenz. Um eine schnelle, und problemfreie Lösung zu liefern, versuchte das Team, eine für österreichische Kennzeichen optimierte Lösung zu implementieren. Dabei spielen Faktoren wie der Aufnahmewinkel der Kamera, die Distanz zum Kennzeichen, sowie die Lichtsituation eine entscheidende Rolle. Weiters muss aus den Einzelbildern der Kamera das Kennzeichen erkannt werden und danach die Buchstaben aus dem Bild extrahiert werden. Um dies zu vollbringen, werden 2 Libraries verwendet.

2.3.1 OpenCV:

OpenCV steht für Open Source Computer Vision und ist eine frei Zugängliche Library, welche meist in Bereichen wie Machine Learning oder Machine Vision ihren Einsatz findet. Unternehmen können kostenlos auf die Bibliothek zugreifen, verändern und weiterentwickeln. Die Basis bilden die mehr als 2500 klassischen und neuen Algorithmen für das maschinelle Sehen. Diese haben unterschiedliche Spezialisierungen, wie unter anderem die Erkennung von Gesichtern, Objekten und Kamerabewegungen, die Generierung von 3D-Modellen, Ähnlichkeiten in Bildern zu finden sowie Markierungen in Augmented Reality anzuzeigen. Zu den mehr als 18 Millionen Downloads und mehr als 47.000 Benutzern zählen meist Unternehmen, Forschungsgruppen oder Regierungsstellen. Bekannte Namen hier sind Google, Microsoft, Intel oder Sony.[4] OpenCV hat eine modulare Struktur, was bedeutet, dass das Paket mehrere gemeinsam genutzte oder statische Bibliotheken enthält. Die folgenden Module sind verfügbar:

- Kernfunktionalität (core) - ein kompaktes Modul, das grundlegende Datenstrukturen definiert, darunter das dichte mehrdimensionale Array Mat und grundlegende Funktionen, die von allen anderen Modulen verwendet werden.
- Bildverarbeitung (imgproc) - ein Bildverarbeitungsmodul, das lineare und nichtlineare Bildfilterung, geometrische Bildtransformationen (Größenänderung, affines und perspektivisches Warping, generisches tabellenbasiertes Remapping), Farbraumkonvertierung, Histogramme usw. umfasst.

- Videoanalyse (Video) - ein Videoanalysemodul, das Algorithmen zur Bewegungsschätzung, Hintergrundsubtraktion und Objektverfolgung umfasst.
- Kamerakalibrierung und 3D-Rekonstruktion (calib3d) - grundlegende Geometriealgorithmen für mehrere Ansichten, Einzel- und Stereokamerakalibrierung, Objektposenschätzung, Stereokorrespondenzalgorithmen und Elemente der 3D-Rekonstruktion.
- 2D Features Framework (features2d) - Erkennung auffälliger Merkmale, Deskriptoren und Deskriptor-Matcher.
- Objekterkennung (objdetect) - Erkennung von Objekten und Instanzen der vordefinierten Klassen (z. B. Gesichter, Augen, Tassen, Menschen, Autos usw.).
- High-Level-GUI (highgui) - eine leicht zu bedienende Schnittstelle zu einfachen UI-Funktionen.
- Video I/O (videoio) - eine einfach zu bedienende Schnittstelle für Videoaufnahmen und Videocodecs.
- Einige andere Hilfsmodule, wie z.B. FLANN- und Google-Test-Wrapper, Python-Bindings, und andere.

2.3.2 Tesseract:

Tesseract ist eine Texterkennungs-Engine welche von Google entwickelt wird.

Optische Zeichenerkennung oder Optical Character Reading (OCR) ist die elektronische oder mechanische Umwandlung von Bildern mit getipptem, handgeschriebenem oder gedrucktem Text in maschinell kodierten Text, sei es aus einem gescannten Dokument, einem Foto eines Dokuments, einem Szenenfoto (z. B. der Text auf Schildern und Werbetafeln in einem Landschaftsfoto) oder aus einem Untertiteltext, der einem Bild überlagert ist (z. B. aus einer Fernsehsendung). [5]

Um besser zu verstehen, wie OCR funktioniert, hilft dieses Prozessdiagramm in der folgenden Abbildung an. Aus Sicht des Endbenutzers ist der OCR-Prozess sehr einfach: Er verarbeitet das Bild und erhält den bearbeitbaren Text.

Abbildung 3: Prozessdiagramm der Optische Zeichenerkennung
[6]

Tesseract bietet die Möglichkeit, Text aus Bildern zu extrahieren. Dies kann in vielen verschiedenen Programmiersprachen erfolgen, oder über eine graphische Nutzeroberfläche eines Drittanbieters.[7] Unterstützt wird Tesseract durch einen Python-Wrapper mit dem Namen **pytesseract**, welcher Bilder wie .jpeg, .png, .gif und viele mehr laden kann, sowie den gelesenen Text ausgeben kann anstatt ihn in einer Datei abspeichern zu müssen.[8]

2.4 Auswahl der Technologie - Backend[SK]

2.4.1 Anforderungen an das Backend

Für das Backend kamen mehrere Technologien in Frage, wie unter anderem Java, JavaScript, Python, PHP, C♯, und viele mehr. Um das Backend zu realisieren, muss die Technologie einige bestimmte Eigenschaften besitzen.[9]

- *Java:*

  Die Vorteile von Java liegen in der Fehlerbehandlung, sowie in Bereichen wie Multithreading und Performanz. Die strikte Fehlerbehandlung führt dabei aber zum Verlust von Flexibilität und Kompaktheit des Codes.

- *JavaScript:*

  Die Syntax von JavaScript ähnelt der von Java. Entwickelt als Scripting-Sprache für HTML, ist JavaScript einfach zu lernen und zu benutzen. Bei der Entwicklung von Websites kann JavaScript direkt in den Quellcode der HTML-Seite eingearbeitet werden. Aber auch im Backend-Bereich kann mit NodeJS in JavaScript entwickelt werden.

- *Python:*

Python ist eine der mit Abstand am leichtesten zu lesenden Programmiersprachen. Die flache Hierarchie ermöglicht ein einfaches Verständnis von Programmen und Codestücken. Weiters macht Python den Entwickler auf jeden Fehler aufmerksam, wenn dieser nicht ausdrücklich ignoriert werden soll. Jedoch ist Python manchmal langsamer in der Ausführung als die konkurrierenden Sprachen. Zusätzlich ist durch die Verwendung von Leerzeichen zur Einrückung ein häufiger Fehlergrund hinzugekommen.

- *PHP:*

  Die PHP Syntax erinnert an eine Mischung aus C, Java und Perl. Das Ziel von PHP ist es, Entwickler schnell und einfach dynamisch generierte Webpages zu bauen. Die vermischte Syntax ist jedoch etwas chaotisch, darum ist es leicht sich in falschen Angewohnheiten zu verirren und Sicherheitslücken offen zu lassen.

Aufgrund vorhandener Vorkenntnisse standen für das Team 3 der oben genannten Technologien zur Auswahl:

- Java,
- JavaScript und
- Python

2.4.2 Verwendung von NodeJS

Von diesen konnte sich JavaScript durchsetzen. Die Gründe dafür waren:[10]

- NPM: Der NPM oder *Node Package Manager*, ist ein Paketmanager für JavaScript, welcher bei NodeJS standardmäßig mitgeliefert wird. Bei NPM werden wiederverwendbare Programmteilen veröffentlicht. Diese können mittels des NPM eingenen Command Line Interfaces installiert werden. Weiters bietet der NPM eine integrierte Versionsverwaltung der Pakete sowie eine Verwaltung der Abhängigkeiten. In diesem Projekt wurden beispielsweise die Module *express* und *mongodb* verwendet. *express* ist ein Framework, welches vor allem in NodeJS Projekten verwendet wird. Die Vorteile von Express sind unter anderem die dem Team bereits bekannte Programmiersprache JavaScript, die Unterstützung der Google V8 engine für bessere Performance, die Robustheit bei einer Vielzahl an HTTP-Anfragen, sowie die einfache Einbindung weiterer Module und Drittanbieterapplikationen. [11]

mongodb stellt dem Entwickler eine API zur Verfügung, welche die Nutzung einer MongoDB-Datenbank stark vereinfacht.

- Verwendung einer NoSQL Datenbank: Aufgrund des Formates, mit dem die Daten aus dem Frontend kommen, bat sich eine nicht relationale Datenbank für das Team an. Die dokumentenorientierte Datenbank MongoDB ist bekannt für ihre hohe Verfügbarkeit, sowie die gute Skalierbarkeit.[12]

- Behandlung von JSON: NodeJS zeichnet sich durch seine einfach Verwendung von JSON-Daten aus. Diese können ohne Parsing oder andere Konvertierungen verarbeitet und darauf zugegriffen werden. Dank NodeJS können JSON Objekte mittels REST-API Anfragen direkt für den Client bereitgestellt werden. Danke NodeJS kann eine Einfache Verbindung zwischen Frontend-Clients und dem Backend-Server geschaffen werden.

2.5 Auswahl der Technologie - Hardware[SK]

2.5.1 Anforderungen an die Hardware

Das Projekt sollte so vielseitig wie möglich, jedoch auch so kompakt wie möglich sein. Dazu musste auf kleine Komponenten gesetzt werden. Diese soll dennoch leistungsfähig genug sein, um jede der drei Zugangsmöglichkeiten parallel zu verwalten.

Raspberry Pi

Die Wahl des Herzstückes fiel auf einen Raspberry Pi. Der Raspberry Pi ist ein vollwertiger Computer, welcher etwas größer als eine Kreditkarte ist. Er besitzt alle bekannten Anschlüsse eines normalgroßen PCs, wie HDMI-Ausgänge für Monitore, USB-Ports für Peripherie wie Maus, Tastatur oder Webcams, sowie einen LAN-Port für eine kabelgebundene Netzwerkverbindung. Als Betriebssystem des Raspberry Pi wurde das vom Hersteller empfohlene Raspbian OS verwendet. Dieses bietet eine grafische Benutzeroberfläche, sowie die Möglichkeit den Raspberry auch ohne angeschlossenen Monitor betreiben zu können. [13]

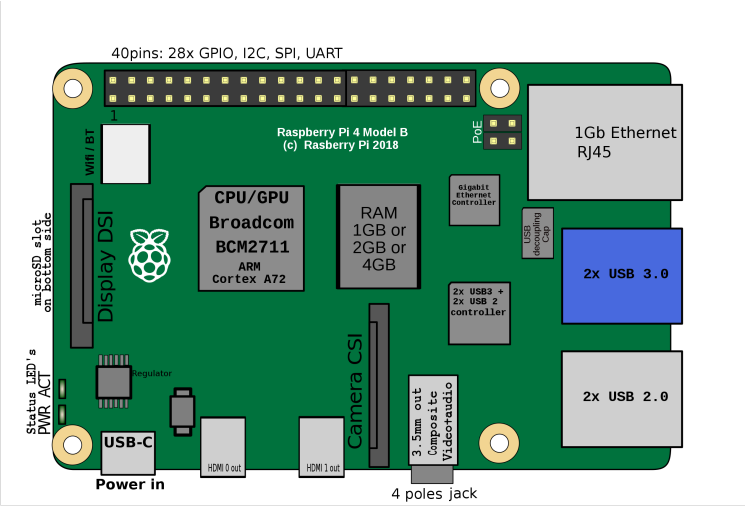

Abbildung 4: Komponenten eines Raspberry Pi [14]

Auszeichnungsmerkmale des Raspberry Pi sind unter anderem die geringen Anschaffungskosten von ab 35 US-$, sowie seine leistungsstarken Komponenten.

Tabelle 1: Übersicht der Komponenten des Raspberry Pi [15]

Komponente	Spezifikation	Besonderheiten
Prozessor	Broadcom BCM2711 - Quad Core Prozessor @ 1.5GHz	ARM Architektur 64-Bit SoC
RAM	1GB, 2GB, 4GB oder 8GB LPDDR4 SDRAM	Tatktung von 3200MHz
USB	2 USB 3.0 Ports 2 USB 2.0 Ports	
GPIO	40 Pin Header	Abwärtskompatibel mit Vorgängermodellen
Display	2 micro-HDMI Ports	jeweils bis zu 4k60 möglich
Speicher	Micro-SD Kartenslot	Speicherplatz für Betriebssystem und Daten
Strom	5V Eingang über USB-C Port 5V Ausgang über GPIO-Header	Anforderung an Stromquelle: mindestens 3A

Der Raspberry Pi ist einer der am weitesten verbreiteten Ein-Platinen-Computer der Welt. Trotz der im Verhältnis zu größeren Systemen schwache Leistung im Jahr 2020 mehr als 7 Millionen mal verkauft worden. Daraus ergibt sich ein Marktanteil von allen PCs von 2.69%. Für ein ausgewogenes Verhältnis zwischen Kompaktheit und Leistung griff man auf einen Raspberry Pi 4 Model B in der Ausführung mit 4GB Arbeitsspeicher zurück. Weiters waren die Anschaffungskosten von etwa 100$ ein weiterer Grund für die Auswahl. [16]

Kernkomponenten des Raspberry Pi

GPIO-Header

Zu den Kernkomponenten, die den Raspberry Pi von anderen PC-Systemen unterscheidet, ist der GPIO-Header. GPIO steht für General Purpose Input / Output, und kann wörtlich zu Allzweckeingabe bzw. -ausgabe übersetzt werden. Sie bezeichnen selbst programmierbare Ein- und Ausgänge, die auf dem Raspberry Pi als angelötete Pins zur Verfügung stehen. Der Raspberry kann über diese Schnittstellen digitale Signale von außen annehmen, sowie auch Signale abgeben. Der Raspberry Pi in der Ausführung Model 4 B hat einen 40-köpfigen GPIO-Header in Form einer Stiftleiste mit zwei Reihen. Davon gibt es einige GPIOs mit bestimmten Zusatzfunktionen, wie I2C, SPI oder seriellen Schnittstellen. Weiters gibt es Pins, welche vom Raspberry eine +5V-Spannung, eine +3.3V Spannung, oder die Möglichkeit der Erdung liefern.

GPIO-Belegung und elektrische Eigenschaften

Grundsätzlich kann in elektronischen Systemen auf elektrische Eigenschaften zurückgegriffen werden, die beobachtet werden müssen. Oft sind diese Eigenschaften Grenzwerte des Systems, welche nicht überschritten werden dürfen. Wird dies ignoriert, kann das System nach Start beschädigt werden und im weiteren Verlauf Defekte aufweisen. Die Eingangsspannung des Raspberry Pi beträgt zwar 5V, jedoch arbeitet der Prozessor selbst nur mit 3.3V. Daher haben auch die GPIOs nur 3.3V zur Verfügung. Dies gilt für die Ausgangsspannung, jedoch auch für die Eingangsspannung, da sonst der Chip des Raspberry Pi beschädigt werden kann. GPIOs sind empfindliche Schnittstellen, denn sie können schon bei geringen Stromstärken Schaden nehmen. Theoretisch ist eine Stromstärke von 16mA (Milliampere) möglich, wobei diese nie benötigt wird, da die GPIOs schon mit einer Stromstärke von 0.5mA geschalten werden können. Um die Langlebigkeit des Raspberry Pi zu gewährleisten, sollten nie mehr als 8mA von einem GPIO abgegeben werden. Es gibt jedoch Ausnahmen, wie die +5V-Pins. Diese bieten für externe Schaltungen eine Spannung bis zu 5V an, sind jedoch ebenfalls bei der Stromentnahme begrenzt. Man spricht hier von etwa 25mA pro 5V-Pin. Sollte dies für eine Schaltung nicht ausreichen, kann auf externe Stromquellen zurückgegriffen werden, wie eine Stromversorgung über ein separates Netzteil mit USB-Anschluss, oder die Versorgung über einen der verfügbaren USB-Ports des Raspberry Pi selbst.

[18]

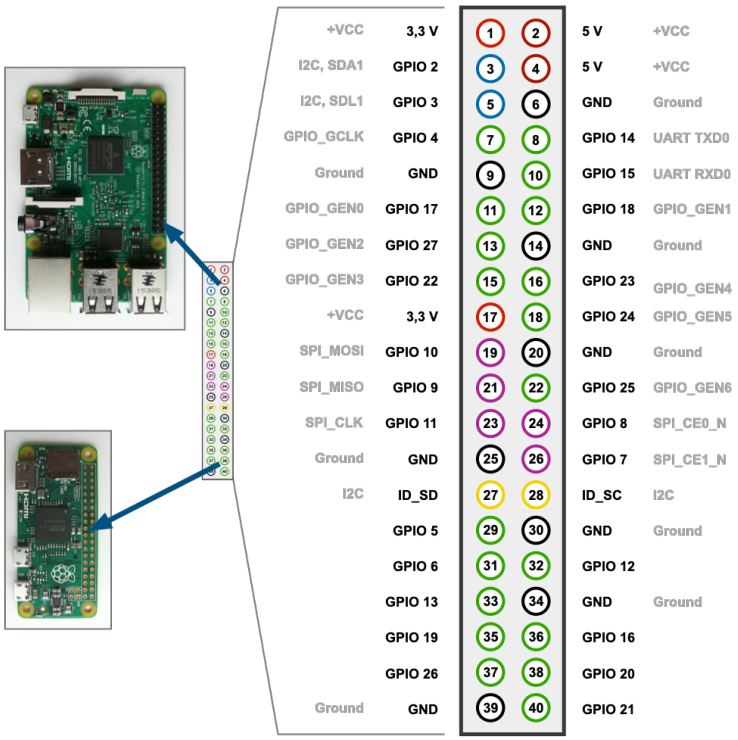

Abbildung 5: Belegung der GPIOs eines Raspberry Pi Model 4B oder Raspberry Pi Zero

[17]

NFC-Leser

Numpad

Kamera

Die Kamera ermöglicht dem Raspberry Pi, die Kennzeichen zu sehen und darauf die Kennzeichen zu erkennen. Dazu wird bei APERTA eine handelsübliche Webcam verwendet, die über einen der beiden USB 3.0 Ports am Raspberry angeschlossen wird. Um genug Auflösung für die Kennzeichenerkennung zu gewährleisten, wurde auf eine Webcam zurückgegriffen, welche mit bis zu 1920 Pixeln mal 1080 Pixeln aufnehmen kann. Alternativ wäre auch eine Raspberry Pi Camera möglich gewesen, jedoch wurde die aufgrund ihres kurzen Flachbandkabels nicht verwendet, um die Kamera auch in größere Entfernung vom Raspberry Pi nutzen zu können.

Abbildung 6: Webcam mit gleichen Spezifikationen
[19]

Abbildung 7: Raspberry Pi Camera Module 2
[20]

Display

Um dem Nutzer vor der Garage mitzuteilen, was gerade geschieht, wird ein Display verwendet, auf dem mitgeteilt wird, ob die Kombination, welche auf dem Nummernfeld eingegeben wurde, korrekt ist, oder ob die NFC-Karte autorisiert ist. Da auf diesem Display nur kurze Textausgaben angezeigt werden, fiel die Entscheidung auf ein LCD-Display gesetzt, welches in zwei Zeilen beschrieben werden kann. Dieses bat zudem weitere Vorteile wie die geringen Kosten von 9€ pro Stück, die Spannungsversorgung durch den Raspberry Pi selbst, sowie die einfache Möglichkeit, Text darauf auszugeben. Im Lieferumfang des Displays war zudem ein I2C Serial Adapter, welcher durch seine 4 benötigen Ports um 8 Pins auf dem Raspberry Pi weniger benötigt, als das direkt angeschlossene Display.

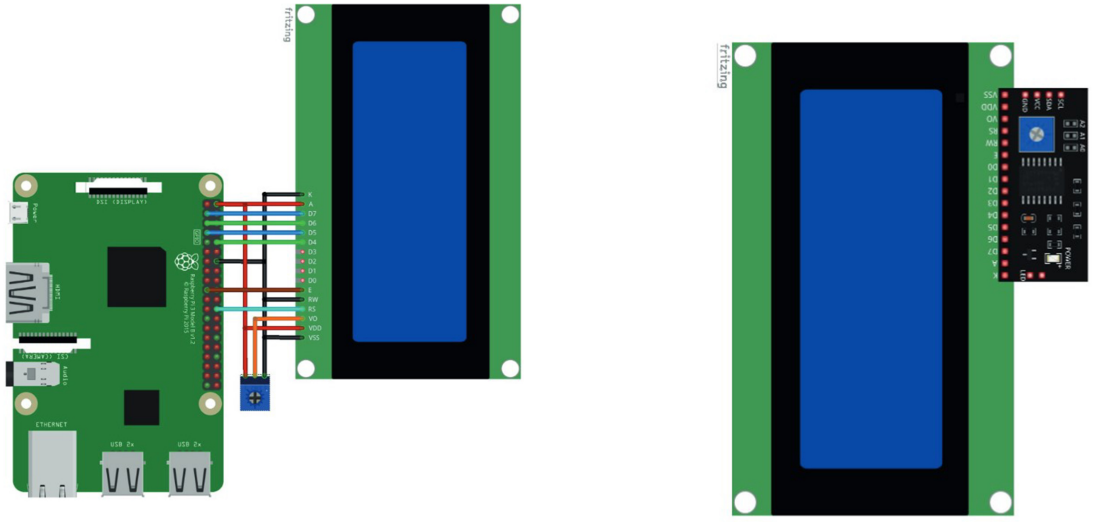

Abbildung 8: Display direkt angeschlossen / Display über I2C Adapter angeschlossen

Die technischen Daten des Displays lauten:

- 4 Zeilen zu je 20 Zeichen beschreibbar
- Blaue Hintergrundbeleuchtung
- 5V Versorgungsspannung

[21]

Relais

Handelsübliche Garagentore werden mit einer Spannung von 230 Volt betrieben. Der Raspberry Pi diese nicht direkt ansteuern, da diese Spannung die interne Elektronik des Pi zerstören würde. Dennoch ist es nötig, wie bei einem Schalter den Steuerstromkreis des Garagentores zu schließen, um den Öffnungsmechanismus zu aktivieren. Um dies zu erreichen, wird ein Relais verwendet, welches in den externen Stromkreis geschalten wird und wie eine Brücke den Stromkreis schließen kann. Ein Relais besteht aus einer Spule aus Draht und einem Metallkern. Schickt man Strom durch den Draht, wird der Kern magnetisiert. Ohne Strom verschwindet das Magnetfeld des Kerns wieder.

Das Relais ist ein Elektromagnet, welcher durch den Steuerkreis einen Eisenanker zu sich zieht und somit den Arbeitsstromkreis schließt. Der Arbeitsstromkreis kann unabhängig vom Steuerkreis aufgebaut sein und auch unterschiedliche Spannungen und Stromstärken besitzen. Wichtig ist nur, dass das richtige Relais für den Arbeitskreis

verwendet wird. Im Fall diese Projektes wird ein Relais verwendet, welches für bis zu 250 Volt Gleichstrom des Arbeitskreises verwendet werden kann. [22]

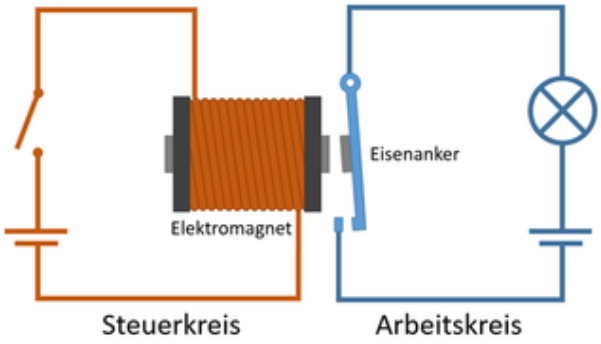

Abbildung 9: Funktionsweise eines Relais als Schema
[23]

Steuerung der Hardware

Um die Komponenten miteinander zu verbinden, war eine Programmiersprache notwendig, die mit den über GPIOs angeschlossenen Bauteilen kommunizieren kann. Dazu gab es eine Handvoll von Programmiersprachen, welche für die Entwicklung von Projekten mit dem Raspberry Pi in Frage kamen.

- *Scratch:* Scratch ist eine baukastenartige Programmiersprache, bei der Befehlsblöcke mithilfe der Maus aneinander angereiht werden. Entwickelt vom MIT Media Lab, richtet Scratch an Programmierneulinge und Kinder, da es sie unterstützen soll, programmieren zu lernen und Code lesen und verstehen zu können.
- *Python:* Python zählt zu den meistverwendeten Programmiersprachen in Verbindung mit dem Raspberry Pi. Es zeichnet sich durch seine einsteigerfreundliche Syntax und die riesige Community aus, welche es ermöglicht, auf eine Vielzahl an Frameworks und Libraries zurückzugreifen. Python beschränkt sich dabei nicht auf einen bestimmten Einsatzbereich, sondern kann für die Entwicklung von graphischen Nutzeroberflächen, im Webdevelopment, zum Trainieren von Künstlichen Intelligenzen sowie für Automatisierung verwendet werden.
- *JavaScript:* JavaScript ist nicht nur eine Erweiterung von HTML als Scripting Sprache, sondern viel mehr eine ganz eigenständige umfangreiche Programmiersprache. Sie wird meistens mit Webentwicklung in Verbindung gebracht, ist jedoch auch fähig, bereits bestehende Applikationen zu erweitern.
- *Java:* Java ist eine der vielseitigsten Programmiersprachen, da sie erlaubt, unabhängig von Betriebssystem zu entwickeln, ohne den Code für jede Plattform

verändern zu müssen. Mit mehr als 3 Milliarden Geräten, auf denen Java läuft, ist sie eine der meist verbreiteten Programmiersprachen.

- *C:* C ist einer der stärksten Konkurrenten von Java. Raspbian OS, das Betriebssystem des Raspberry Pi, wurde beispielsweise in C geschrieben. C zeichnet sich durch einen klar strukturierten Programmierstil und die Möglichkeit, Arbeitsspeicher direkt anzusprechen, aus. Die Haupteinsatzgebiete von C sind die Entwicklung von Betriebssystemen und Compilern.
- *C++:* Vergleicht man C mit C++, kann sich C++ mit objektorientierter Programmierung auszeichnen. Die Kombination aus prozeduraler und objektorientierterer Programmierung machen C++ zu einer Allzwecklösung, mit welcher von Betriebssystemen über Spiele bis hin zu Webbrowsern alles entwickelt werden kann.
- *Perl:* Die Perl Org. hat mit Perl eine Sprache entwickelt, welche für fast jede Aufgabe, die mit C oder C++ Libraries zu tun hat, geeignet ist. Die große Auswahl an Libraries und Modulen sprechen trotz der geringen Bekanntheit für Perl, welche weiters für die Webentwicklung, Systemadministration, GUI-Entwicklung und vielem mehr verwendet werden kann.
- *Erlang:* Erlang ist eine relativ unbekannte Sprache, da sie meist für Industrieapplikationen verwendet wird. Auszeichnungsmerkmale von Erlang sind beispielsweise die Fähigkeit, extrem skalierbare Echtzeitsysteme zu entwickeln. Auch bei dezentralisierten Systemen ist Erlang eine gute Wahl, da das Programm bei Ausfall eines Rechners aus dem Cluster problemlos weiter arbeiten kann. Verwender sind unteranderem Banken und Telekommunikationsunternehmen.

[24] Durch die einfache Möglichkeit, mit den GPIOs zu arbeiten, sowie der einfach verständlichen Syntax, wurde Python verwendet. Das Team konnte somit zusätzlich auf bereits bestehende Kenntnisse aufbauen.

3 Lösungsansätze

3.1 Profil Management

3.2 Webshop

3.3 Automatic Number Plate Recognition (ANPR)[SK]

Die Kennzeichenerkennung stellt das Alleinstellungsmerkmal des Projektes dar. Der OpenCV-Python-Code für die Nummernschilderkennung umfasst drei Hauptschritte. Der erste Schritt ist die Erkennung von Nummernschildern. Die Konturfunktion wird verwendet, um die rechteckigen Objekte im Bild zu erkennen und das Nummernschild zu finden. Der zweite Schritt ist die Zeichensegmentierung. Sobald die Kontur das Nummernschild erkannt hat, müssen wir es ausschneiden und als neues Bild speichern. Und der letzte Schritt ist die Zeichenerkennung. Die optische Zeichenerkennung wird auf dem ausgeschnittenen Bild durchgeführt, um das Kennzeichen zu erkennen.

3.3.1 Überprüfung der Kennzeichen

Die erkannten Buchstaben werden genutzt, um eine Funktion aufzurufen, welche diese mit den zugelassenen Kennzeichen vergleicht. Dazu werden die zugelassenen Kennzeichen vom Server abgefragt, welcher diese aus der Datenbank lädt. Die Abfrage erfolgt über einen REST-Abfrage. REST ist weder ein Protokoll, noch ein Standard. REST steht für REpresentational State Transfer, API für Application Programming Interface, welche gemeinsam eine Programmierschnittstelle bilden, mit der eine Anwendung mit einem Server kommunizieren kann. Dies wird meist mit dem HTTP-Protokoll genutzt, um Services über URLs zu erreichen. Dazu stehen die HTTP-Methoden GET, POST, PUT und DELETE zur Verfügung.

- *GET:* GET ist eine Methode, welche einen Inhalt eines Servers abruft.
- *POST:* POST ist eine Methode, um vom Client Daten an den Server zu senden, welcher diese weiter verarbeiten und in die Datenbank speichern kann.

- *PUT:* PUT bietet die Möglichkeit, bereits bestehende Daten zu ändern.
- *DELETE:* DELETE bietet die Möglichkeit, bestehende Daten zu löschen.

[25] Neben der Kennzeichenerkennung bietet das Projekt noch 2 weitere Zutrittsmöglichkeit, nämlich das Nummernfeld sowie die NFC-Funktionalität. Damit alle Zutrittsmöglichkeiten parallel funktionieren, wird mit der Bibliothek *threading* parallel gearbeitet. Hiermit können mehrere Funktionen parallel gestartet werden. Dies kann beispielsweise wie folgt erfolgen:

```
1  import time
2  from threading import Thread
3      def funktion_1():
4          while True:
5              print("Funktion 1")
6              time.sleep(1)
7
8
9      def funktion_2():
10         while True:
11             print("Funktion 2")
12             time.sleep(1)
13
14
15     thread_1 = Thread(target=funktion_1)
16     thread_2 = Thread(target=funktion_2)
17
18     thread_1.start()
19     thread_2.start()
```

Listing 1: Funktionsweise von Multiprocessing

`thread_1` und `thread_2` sind Threads, welche parallel ausgeführt werden. Sie bekommen als Parameter eine Funktion, welche ausgeführt werden soll, in diesem Fall `funktion_1` und `funktion_2`. Mit `start()` werden die Threads gestartet. [26]

3.4 Backend

Der Server bildet das Rückgrat des Projektes. Hier werden die Daten aus der Datenbank abgefragt. Diese werden von einem Client über einen REST-Aufruf abgefragt. Der Server kann zudem die neu zugelassenen Kennzeichen in die Datenbank speichern.

3.4.1 Oracle VM

Der Server läuft auf einer Instanz einer Oracle Virtual Machine, ebenso die MongoDB-Datenbank. Die Oracle VM bietet die Möglichkeit, eine leistungsstarke Basis für den

Server sowie die Datenbank bereitzustellen. Die technischen Daten der Instanz sind:

Tabelle 2: Technische Daten der Oracle VM Instanz

Komponente	**Wert**
CPU	2 OCPU
RAM	16 GB
Netzwerk	1.4 GBit Bandbreite

Im Vergleich zu anderen Cloudanbietern wie AWS, Microsoft oder Google, verwendet die Oracle Cloud Infrastructure, kurz OCI, anstelle von virtuellen CPUs sogenannte OCPUs. Jede vCPU ist als ein Hyperthread eines Intel Xeon-Kerns definiert. Ein Standard-Intel-Prozessorkern mit aktiviertem Hyperthreading hat 2 Threads.

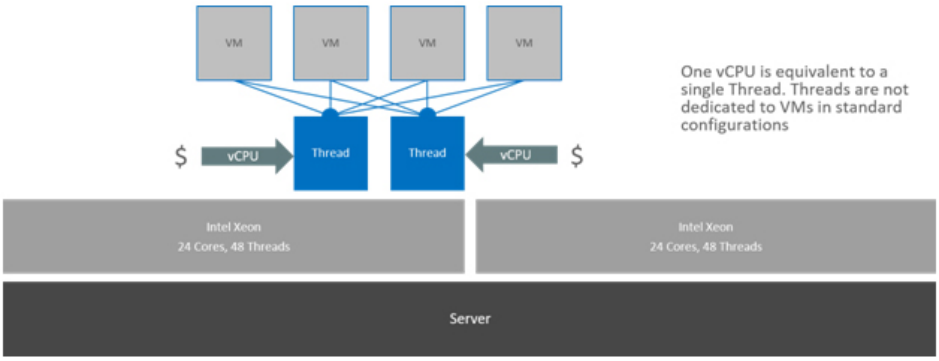

Abbildung 10: Visualisierung von vCPUs
[27]

Eine OCPU ist definiert als die CPU-Kapazität, die einem physischen Kern eines Intel Xeon-Prozessors mit aktiviertem Hyperthreading oder einem physischen Kern eines Oracle SPARC-Prozessors entspricht.

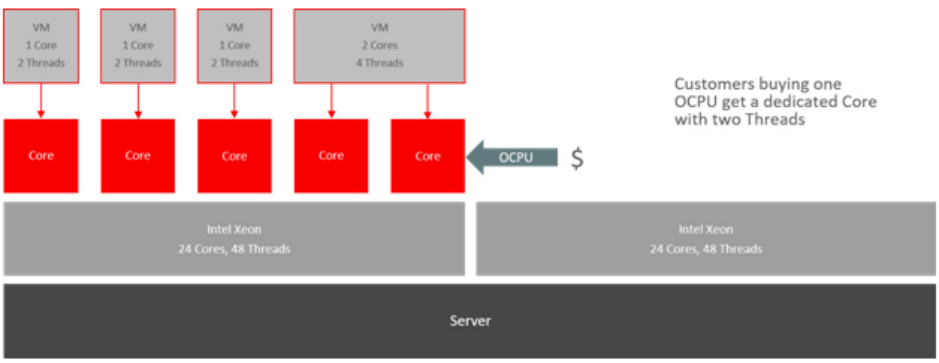

Abbildung 11: Visualisierung von OCPUs
[28]

[29]

3.5 MongoDB

Die Datenbank läuft auf der selben Oracle VM Instanz wie das Backend. Somit gibt es zwischen Server und Datenbank keine zusätzliche Latenz. Da MongoDB eine nicht relationale Datenbank ist, werden die Daten in sogenannten Collections anstelle von Tabellen abgespeichert. Je nach abzuspeichernder Zugangsmöglichkeit, verändert sich das Format der Daten. Für ein Kennzeichen müssen die Daten wie folgt strukturiert sein:

Kennzeichen

```
1     {
2         "_id": "623f449cb8c1b2f4f64c4351",
3         "licenseplate_id": 1,
4         "licenseplate": "RO 108DV",
5         "time_created": 1647252201,
6         "active": true
7     }
```

Listing 2: Aufbau eines Kennzeichen in der Datenbank

Das Attribut `_id` wird automatisch generiert und ist ein eindeutiger Schlüssel, welcher die eindeutige Identifizierung eines Datensatzes darstellt. Mithilfe des `licenseplate_id` wird eine eindeutige Kennzeichen Identifikationsnummer gespeichert. `licenseplate` ist das Kennzeichen, welches vom Client übergeben wurde. `time_created` ist die Zeit, zu der das Kennzeichen erstellt wurde. `active` gibt an, ob das Kennzeichen der aktiv ist.

Nummernfeld

Bei einer Nummernfeld-Kombination sieht der Datensatz wie folgt aus:

```
1     {
2         "_id": "622f181d35a3c2ce232acad9",
3         "numpad_id": 1,
4         "numpad_code": "123456",
5         "time_created": 1647252201,
6         "active": true
7     }
```

Listing 3: Aufbau einer Nummernfeld-Kombination in der Datenbank

NFC

Jede NFC-Karte hat eine eindeutige, 10-stellige Kartenidentifikationsnummer. Diese kann mittels eines RFID-Lesers ausgelesen werden. In der Datenbank werden die NFC-Informationen wie folgt gespeichert:

```
1     {
2         "_id": "622f781cfbd4e8de471a4035",
3         "rfid_id": 3,
4         "rfid_code": "01928384756",
5         "time_created": "123456789",
6         "active": true
7     }
8
```

Listing 4: Aufbau einer NFC-Karte in der Datenbank

4 Systemarchitektur

4.1 Übersicht der Systemarchitektur

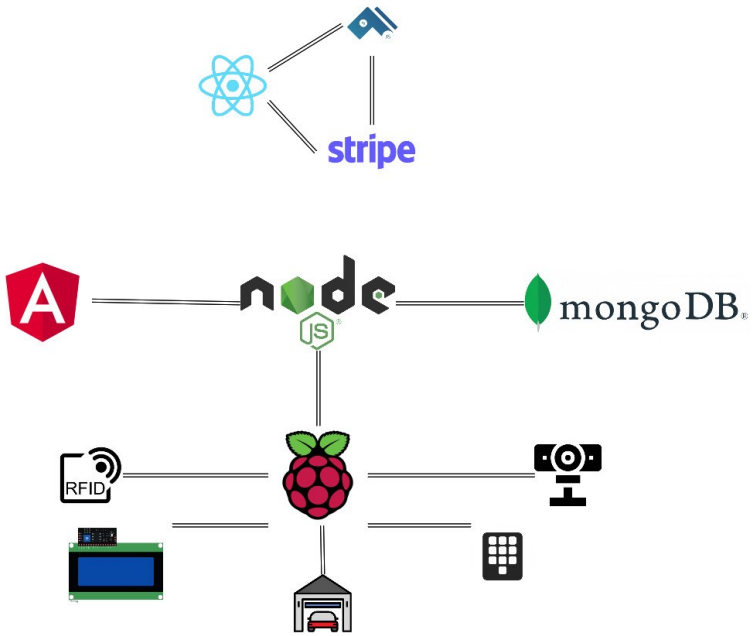

Abbildung 12: Systemarchitektur

Diese Diplomarbeit setzt sich aus zwei voneinander unabhängigen Systemen zusammen. Das Shopsystem, bestehend aus einem React-Frontend, kommuniziert mit zwei Bibliotheken, CommerceJS und Stripe, welche für die Produktverwaltung, sowie den Bezahlvorgang zuständig sind. Das Angular-Frontend des Dashboard in Verbindung mit dem NodeJS Server, der MongoDB Datenbank sowie dem Raspberry bietet das zweite System. Der Raspberry ist über GPIOs mit dem RFID-Leser, dem Nummernfeld und dem Display verbunden. Weiters wurde die Kamera an einen der USB 3.0 Ports des Raspberry angeschlossen.

4.2 Frontend (Angular-Applikation)

4.3 Frontend (React-Applikation)

4.4 Backend (NodeJS-Server)[SK]

Das Backend besteht aus einer JavaScipt Datei, in welcher sämtliche Endpunkte definiert werden. Weiters benötigt das Backend einige Pakete, um beispielsweise mit der Datenbank zu kommunizieren. Diese werden in `package.json` aufgeführt. Bei erstmaliger Installation wird die Datei automatisch durchlaugfen, und installiert alle benötigten Dateien und Pakete. Dazu wird der Terminal-Befehl `npm install` ausgeführt. Der Node Package Manager installiert alle Pakete im Ordner `node_modules`.
Der Server selbst wird weiters eingeteilt in:

- *Imports:* Zu beginn werden alle benötigten Pakete importiert. Dies geschiet mittels `require()`.
- *Setup:* In weiterer Folge werden benötigte Variablen, wie die Verbindungs-URL der MongoDB Datenbank, definiert. Da die Datenbank und der Server auf der selben Maschine laufen, der Host auf `mongodb://127.0.0.1:27017` gesetzt. Der weitere Aufbau der UTL besteht aus Parametern wie zum Beispiel der `serverSelectionTimeOutMS`.
- *Header:* Um den Zugriff des Raspberry auf den Server zu gewährleisten, müssen die Header-Parameter gesetzt werden. Dies geschieht mittels `app.use()` und einer als Parameter übergebenen Funktion, in der mit `res.setHeader()` die Header-Parameter gesetzt werden. Diese bestimmen welche IP-Adressen auf den Server zugreifen können, welche Methoden verwedet werden dürfen, sowie die zugelassenen Header bei Anfragen des Clients.

4.5 Kennzeichenerkennung [SK]

5 Umsetzung

5.1 Implementierung der Kennzeichenerkennung[SK]

```
1      import requests
2      import json
3      import string
4      from relais import initiateOpeningSequence
5
6
7      def checkPlate(text):
8          print(type(text))
9          r = requests.get('http://130.162.215.116/get-licenseplates'
       )
10         val = json.loads(r.text)
11         table = str.maketrans('', '', string.ascii_lowercase)
12         text = text.strip()
13         for value in val:
14             print("value: ", value["licenseplate"])
15             print("value no whitespace:", value["licenseplate"].
       replace(" ", ""))
16             print("text from anpr: ", text)
17             if value["active"]:
18                 print(text.replace(" ", "").translate(table) ==
19                       value["licenseplate"].translate(table).
       replace(" ", ""))
20                 if text.replace(" ", "").translate(table) == value[
       "licenseplate"].translate(table).replace(" ", ""):
21                     print("licenseplate recognized, initiating
       opening sequence")
22                     initiateOpeningSequence()
23                 else:
24                     print("not recognized, staying closed")
25
26
```

Listing 5: checkLicensePlate code

```
1  import cv2, os
2  import imutils
3  import numpy as np
4  import pytesseract
5  import re
6  from checkLicensePlate import checkPlate
7  def npr():
8      cap = cv2.VideoCapture(0)
9      if not cap.isOpened():
10         print("cannot open camera")
11         exit()
12     while(True):
13         ret, frame = cap.read()
14         if not ret:
15             print("Can't receive frame (stream end?). Exiting ...")
16             break
```

```
17          cv2.imshow("Frame", frame)
18          gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # convert
        to grey scale
19          gray = cv2.bilateralFilter(gray, 11, 17, 17)  # Blur to
        reduce noise
20          edged = cv2.Canny(gray, 30, 200)  # Perform Edge detection
21          cnts = cv2.findContours(edged.copy(), cv2.RETR_TREE,
22                                  cv2.CHAIN_APPROX_SIMPLE)
23          cnts = imutils.grab_contours(cnts)
24          cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:10]
25          screenCnt = None
26          for c in cnts:
27              peri = cv2.arcLength(c, True)
28              approx = cv2.approxPolyDP(c, 0.018 * peri, True)
29              if len(approx) == 4:
30                  screenCnt = approx
31                  break
32          if screenCnt is None:
33              detected = 0
34              print("No contour detected")
35              exit()
36          else:
37              detected = 1
38          if detected == 1:
39              cv2.drawContours(frame, [screenCnt], -1, (0, 255, 0),
        3)
40          mask = np.zeros(gray.shape, np.uint8)
41          new_image = cv2.drawContours(mask, [screenCnt], 0, 255,
        -1,)
42          new_image = cv2.bitwise_and(frame, frame, mask=mask)
43          (x, y) = np.where(mask == 255)
44          (topx, topy) = (np.min(x), np.min(y))
45          (bottomx, bottomy) = (np.max(x), np.max(y))
46          Cropped = gray[topx:bottomx+1, topy:bottomy+1]
47          text = pytesseract.image_to_string(Cropped, config='--psm
        11')
48          print("text before replacing:", text)
49          text_replaced = re.sub('[^a-zA-Z0-9 \n\.]', '', text)
50          text_replaced = text_replaced.replace(" ", "")
51          print("Detected Number is:", text_replaced)
52
53          checkPlate(text_replaced)
54          cv2.imshow("Frame", frame)
55          cv2.imshow('Cropped', Cropped)
56          break
57      cv2.destroyAllWindows()
58      cap.release()
59      os.system("python new_anpr.py")
60
```

Listing 6: ANPR code

```
1           import RPi.GPIO as GPIO
2   import time
3
4   in1 = 7
5   GPIO.setmode(GPIO.BOARD)
6   GPIO.setup(in1, GPIO.OUT)
7
8   GPIO.output(in1, False)
9
10  def initiateOpeningSequence():
11      GPIO.output(in1,True)
12      time.sleep(0.1)
```

```
13      GPIO.output(in1,False)
14      GPIO.cleanup()
15
```

Listing 7: Relais code

5.2 Implementierung des Backend [SK]

```
1  const express = require('express')
2  const bodyParser = require('body-parser')
3  const app = express()
4  app.set('view engine', 'ejs')
5  const MongoClient = require('mongodb').MongoClient
6  connstring = 'mongodb://127.0.0.1:27017/?directConnection=true&
       serverSelectionTimeoutMS=2000&appName=mongosh+1.2.3'
7  app.use(bodyParser.urlencoded({ extended: true }))
8  app.use(function(req, res, next) {
9      // Website you wish to allow to connect
10     res.setHeader('Access-Control-Allow-Origin', '*');
11     // Request methods you wish to allow
12     res.setHeader('Access-Control-Allow-Methods', 'GET, POST,
       OPTIONS, PUT, PATCH, DELETE');
13     // Request headers you wish to allow
14     res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With
       ,content-type');
15     // Set to true if you need the website to include cookies in
       the requests sent
16     // to the API (e.g. in case you use sessions)
17     res.setHeader('Access-Control-Allow-Credentials', true);
18     // Pass to next layer of middleware
19     next();
20 });
21 MongoClient.connect(connstring, { useUnifiedTopology: true })
22     .then(client => {
23         console.log('Connected to Database')
24         const db = client.db('aperta')
25
26         const licenseplateCollection = db.collection('licenseplate'
       )
27         const rfidCollection = db.collection('rfid')
28         const numpadCollection = db.collection('numpad')
29             // All your handlers here...
30         app.delete('/delete-rfid', (req, res) => {
31             console.log("deleting rfid code" + req.body.rfid_code)
32             rfidCollection.findOneAndDelete({ "rfid_code": req.body
       .rfid_code }).then(result => { res.sendStatus(200) })
33
34         })
35         app.delete('/delete-numpad', (req, res) => {
36             console.log("deleting numpad " + req.body.numpad_code)
37             numpadCollection.findOneAndDelete({ "numpad_code": req.
       body.numpad_code }).then(result => { res.sendStatus(200) })
38         })
39         app.delete('/delete-licenseplate', (req, res) => {
40             console.log("deleting licenseplate " + req.body.
       licenseplate)
41             licenseplateCollection.findOneAndDelete({ "licenseplate
       ": req.body.licenseplate }).then(result => { res.sendStatus(200)
        })
42         })
```

```
43          app.post('/add-licenseplate', (req, res) => {
44              licenseplateCollection.findOneAndDelete({ "licenseplate
    ": req.body.licenseplate }).then(result => {
45                  licenseplateCollection.insertOne(req.body)
46                      .then(result => {
47                          console.log('licenseplate added')
48                          res.send(200)
49                      })
50              })
51              .catch(error => console.error(error))
52          })
53          app.post('/add-numpad', (req, res) => {
54              numpadCollection.findOneAndDelete({ "numpad_code": req.
    body.numpad_code }).then(result => {
55                  numpadCollection.insertOne(req.body)
56                      .then(result => {
57                          console.log('numpad code added')
58                          res.send(200)
59                      })
60              })
61              .catch(error => console.error(error))
62          })
63          app.post('/add-rfid', (req, res) => {
64              console.log(req.body.rfid_code)
65              rfidCollection.findOneAndDelete({ "rfid_code": req.body
    .rfid_code }).then(result => {
66                  rfidCollection.insertOne(req.body)
67                      .then(result => {
68                          console.log('rfid code added')
69                          res.send(200)
70                      })
71              })
72              .catch(error => console.error(error))
73          })
74          app.get('/get-numpad-codes', function(req, res) {
75              db.collection('numpad').find().toArray()
76                  .then(results => {
77                      console.log("retrieving numpad codes")
78                      res.send(results)
79                  })
80                  .catch(error => console.error(error))
81                  // do something here
82          })
83          app.get('/get-licenseplates', function(req, res) {
84              licenseplateCollection.find().toArray()
85                  .then(results => {
86                      console.log("retrieving licenseplates")
87                      res.send(results)
88                  })
89                  .catch(error => console.error(error))
90                  // do something here
91          })
92          app.get('/get-rfid-codes', function(req, res) {
93              db.collection('rfid').find().toArray()
94                  .then(results => {
95                      console.log("retrieving rfid codes")
96                      res.send(results)
97                  })
98                  .catch(error => console.error(error))
99                  // do something here
100         })
101
102
103     }).catch(console.error)
```

```
104
105 app.listen(3000, function() {
106     console.log('listening on 3000')
107 })
108
109
```

Listing 8: Server code

5.3 Implementierung des Displays[SK]

```
1      #!/usr/bin/python3
2
3  from signal import signal, SIGTERM, SIGHUP, pause
4  from rpi_lcd import LCD
5  import time
6  lcd = LCD()
7  # python3 -c 'from testDisplay import displayText; displayText
       ("123456789")'
8
9
10 def safe_exit(signum, frame):
11     exit(1)
12
13
14 def displayText(writingString):
15     try:
16         signal(SIGTERM, safe_exit)
17         signal(SIGHUP, safe_exit)
18
19         outString = writingString
20
21         lcd.text("Text Read:", 1)
22         lcd.text(outString, 2)
23
24         time.sleep(5)
25     finally:
26         lcd.clear()
27
28
29 def readNFCUnsuccessful():
30     try:
31         signal(SIGTERM, safe_exit)
32         signal(SIGHUP, safe_exit)
33
34         lcd.text("Error at reading", 1)
35         lcd.text("Please try again", 2)
36
37         time.sleep(5)
38     finally:
39         lcd.clear()
40
41 def readNumpadUnsuccessful():
42     try:
43         signal(SIGTERM, safe_exit)
44         signal(SIGHUP, safe_exit)
45
46         lcd.text("Wrong combination", 1)
47         lcd.text("Please try again", 2)
48
```

```
49          time.sleep(5)
50      finally:
51          lcd.clear()
52
53
54
```

Listing 9: Display code

5.4 Implementierung des Relais[SK]

```
1  import RPi.GPIO as GPIO
2  import time
3
4  in1 = 7
5  GPIO.setmode(GPIO.BOARD)
6  GPIO.setup(in1, GPIO.OUT)
7
8  GPIO.output(in1, False)
9
10 def initiateOpeningSequence():
11     GPIO.output(in1,True)
12     time.sleep(0.1)
13     GPIO.output(in1,False)
14     GPIO.cleanup()
```

Listing 10: Relais code

6 Persönliche Ziele

6.1 Projektverlauf

6.2 Erkenntnisse von Benjamin Golic

6.3 Erkenntnisse von David Hauser

6.4 Erkenntnisse von Simon Koll

Das Projekt nahm immer mehr Substanz an, je weiter die Entwicklung fortschritt. Es kamen neue Ideen hinzu, die Anfangs noch gar nicht im Raum standen. Die Erkenntnisse und Inhalte des ITP-, INSY- sowie SEW-Unterrichts haben mich in der

7 Zusammenfassung

Literaturverzeichnis

[1] S. Austria, „Vorläufiger Fahrzeug-Bestand am 28.02.2022," Zuletzt besucht am 25.03.2022. Online verfügbar: https://www.statistik.at/wcm/idc/idcplg?IdcService=GET_PDF_FILE&RevisionSelectionMethod=LatestReleased&dDocName=062059

[2] IBM, „ACID-Eigenschaften für Transaktionen," Zuletzt besucht am 28.03.2022. Online verfügbar: https://www.ibm.com/docs/de/cics-ts/5.4?topic=processing-acid-properties-transactions

[3] IONOS, Zuletzt besucht am 28.03.2022. Online verfügbar: https://www.ionos.de/digitalguide/websites/web-entwicklung/mongodb-vorstellung-und-vergleich-mit-mysql/

[4] O. Team, Zuletzt besucht am 29.03.2022. Online verfügbar: https://opencv.org/about/

[5] Wikipedia, Zuletzt besucht am 30.03.2022. Online verfügbar: https://en.wikipedia.org/wiki/Optical_character_recognition

[6] B. Baruno, Zuletzt besucht am 30.03.2022. Online verfügbar: https://medium.com/zeals-tech-blog/introduction-to-tesseract-ocr-84d3eff6f9df

[7] tesseract ocr, Zuletzt besucht am 29.03.2022. Online verfügbar: https://tesseract-ocr.github.io/tessdoc/

[8] S. Hoffstaetter, Zuletzt besucht am 29.03.2022. Online verfügbar: https://github.com/madmaze/pytesseract

[9] Learneroo, Zuletzt besucht am 25.03.2022. Online verfügbar: https://www.learneroo.com/modules/9/nodes/620

[10] J. D. Tomislav Capan, Zuletzt besucht am 25.03.2022. Online verfügbar: https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

[11] S. CM, Zuletzt besucht am 25.03.2022. Online verfügbar: https://www.techomoro.com/what-are-the-benefits-of-using-express-js-for-backend-development/

[12] MongoDB, Zuletzt besucht am 25.03.2022. Online verfügbar: https://www.mongodb.com/why-use-mongodb

[13] R. P. Foundation, Zuletzt besucht am 25.03.2022. Online verfügbar: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

[14] W. Commons, Zuletzt besucht am 26.03.2022. Online verfügbar: https://upload.wikimedia.org/wikipedia/commons/thumb/e/ef/RaspberryPi_4_Model_B.svg/1200px-RaspberryPi_4_Model_B.svg.png

[15] R. P. Foundation, Zuletzt besucht am 26.03.2022. Online verfügbar: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[16] I. ARC, Zuletzt besucht am 26.03.2022. Online verfügbar: https://www.industryarc.com/Report/19454/industrial-raspberry-pi-market.html

[17] E. Kopendium, „Raspberry Pi: GPIO - General Purpose Input Output - Grafik," Zuletzt besucht am 28.03.2022. Online verfügbar: https://www.elektronik-kompendium.de/sites/raspberry-pi/bilder/raspberry-pi-gpio.png

[18] E. Kompendium, „Raspberry Pi: GPIO - General Purpose Input OutputRaspberry Pi: GPIO - General Purpose Input Output," Zuletzt besucht am 28.03.2022. Online verfügbar: https://www.elektronik-kompendium.de/sites/raspberry-pi/2002191.htm

[19] Sandberg.world, Zuletzt besucht am 28.03.2022. Online verfügbar: https://cdn.sandberg.world/products/images/lg/133-97_lg.jpg

[20] R. P. Foundation, Zuletzt besucht am 26.03.2022. Online verfügbar: https://images.prismic.io/rpf-products/ffa68a46-fd44-4995-9ad4-ac846a5563f1_Camera%20V2%20Hero.jpg?ixlib=gatsbyFP&auto=compress%2Cformat&fit=max&q=50&w=600&h=400

[21] AZ-Delivery, „HD44780 2004 LCD Display Bundle 4x20 Zeichen mit I2C Schnittstelle," Zuletzt besucht am 26.03.2022. Online verfügbar: https://cdn.shopify.com/s/files/1/1509/1638/products/1.Main_1x_HD447802004LCDDisplayBundleGrun4x20ZeichenmitI2CSchnittstelle_500x.jpg?v=1597657362

[22] Homecomputermuseum.de, „Relais," Zuletzt besucht am 28.03.2022. Online verfügbar: https://www.homecomputermuseum.de/technik/rechnen-mit-strom/relais/

[23] ——, Zuletzt besucht am 28.03.2022. Online verfügbar: https://www.homecomputermuseum.de/fileadmin/user_upload/Technik/relais.png

[24] S. Shawn, Zuletzt besucht am 28.03.2022. Online verfügbar: https://www.seeedstudio.com/blog/2020/02/25/which-raspberry-pi-programming-language-should-you-use-comparison-guide/

[25] M. D. S. . F. Karlstetter, Zuletzt besucht am 29.03.2022. Online verfügbar: https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/

[26] P. Documentation, Zuletzt besucht am 29.03.2022. Online verfügbar: https://docs.python.org/3/library/threading.html

[27] V. Nohejl, Zuletzt besucht am 30.03.2022. Online verfügbar: https://www.industry-era.com/images/article/vCPU.jpg

[28] ——, Zuletzt besucht am 30.03.2022. Online verfügbar: https://www.industry-era.com/images/article/OCPU.jpg

[29] ——, Zuletzt besucht am 30.03.2022. Online verfügbar: https://www.industry-era.com/Cloud-Services-Pricing.php

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang