

DIPLOMARBEIT

Gesamtprojekt

driverslog

Entwurf eines app-gesteuerten digitalen Fahrtenbuchs

Niklas Parhammer 5AHEL

Betreuer: Dipl. Ing. Siegbert Schrempf

Philip Wienerroither 5AHEL

Daniel Wipplinger 5AHEL

ausgeführt im Schuljahr 2021/22

Abgabevermerk:

Datum: 04.04.2022

übernommen von:

Eidesstattliche Erklärung

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

Salzburg, am 04.04.2022

Verfasserinnen / Verfasser:



Niklas Parhammer



Philip Wienerroither



Daniel Wipplinger

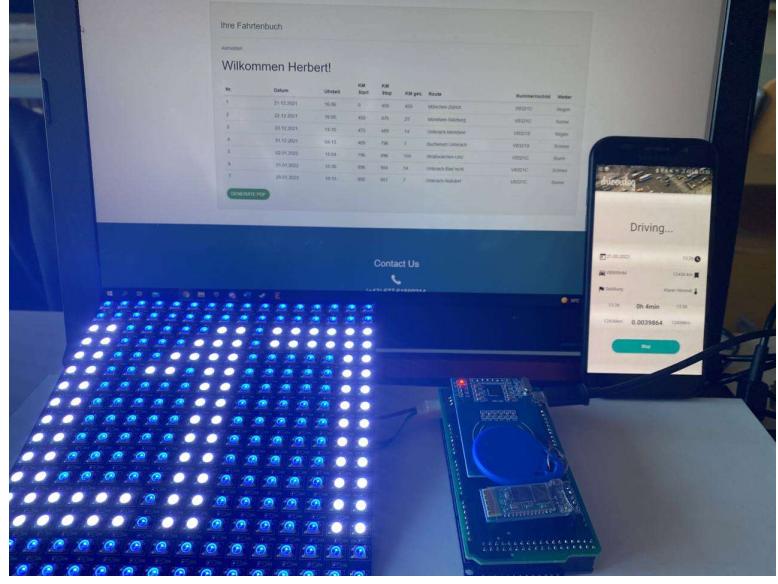
DIPLOMARBEIT**DOKUMENTATION**

Namen der Verfasserinnen / Verfasser	Niklas Parhammer Philip Wienerroither Daniel Wipplinger
Jahrgang Schuljahr	5AHEL 2021/22
Thema der Diplomarbeit	driverslog – smartes Fahrtenbuch
Kooperationspartner	---

Aufgabenstellung	Aktuell müssen Fahrschüler während der Ausbildungszeit Protokoll über ihren Kilometerfortschritt führen. Die Protokollierung erfolgt dabei über ein Formular, welches händisch ausgefüllt wird. Zusätzlich werden Witterungsbedingungen, Fahrbahnverhältnisse und Kilometerstand dokumentiert. Diese analoge Methode, Fahrtenbücher zu führen, ist mühsam und anfällig für Manipulation.
------------------	--

Realisierung	Die Kombination aus Hardware, App und Datenbank dokumentiert Fahrdaten, wie Wetterbedingungen, Kilometerfortschritt und Uhrzeit. Außerdem werden Warnungen für bevorstehende Überprüfungsfahrten versandt, um die Terminplanung zu erleichtern. Implementierte Fahrer-Authentifizierungsmaßnahmen erschweren Manipulationsversuche.
--------------	---

Ergebnisse	Für die Verbindung zwischen Platine und App wird ein Bluetooth Modul verwendet. Dieses übermittelt die Auswertung der RFID Abfrage an das Smartphone. Die App am Smartphone bindet noch alle relevanten Informationen zur Fahrt ein, wie zum Beispiel die gefahrenen Kilometer, die Witterungsbedingungen und das Datum. Anschließend werden alle Daten in einer Datenbank abgespeichert, welche auch von einer Website eingesehen werden kann.
------------	---

Typische Grafik, Foto etc. (mit Erläuterung)	 <p>Abbildung 1: Prototyp</p>
---	--

Teilnahme an Wettbewerben, Auszeichnungen	---
---	-----

Möglichkeiten der Einsichtnahme in die Arbeit	Schulbibliothek der HTBLuVA Salzburg
---	--------------------------------------

Approbation (Datum / Unterschrift)	Prüferin / Prüfer	Direktorin / Direktor Abteilungsvorständin / Abteilungsvorstand
---------------------------------------	-------------------	--

DIPLOMA THESIS

Documentation

Author(s)	Niklas Parhammer Philip Wienerroither Daniel Wipplinger
Form	5AHEL
Academic year	2021/22
Topic	<i>driverslog</i> – smart driver's log
Co-operation Partners	---

Assignment of Tasks	Currently, student drivers must keep a log of their kilometre progress during the training period, which is done in paper form. In addition, weather conditions, road conditions and mileage are documented. This analogue method of keeping mileage logs is cumbersome and susceptible to manipulation.
---------------------	--

Realisation	The combination of hardware, app and database documents driving data such as weather conditions, kilometre progress and time in a simplified way. It also sends alerts for upcoming check rides to facilitate scheduling. Implemented driver authentication measures make tampering attempts more difficult.
-------------	--

Results	A Bluetooth module is used for the connection between the board and the app. It transmits the evaluation of the RFID query to the smartphone. The app on the smartphone also integrates all relevant information about the journey, such as the kilometres driven, the weather conditions and the date. All data is then stored in a database, which can also be viewed on a website.
---------	---

Illustrative Graph, Photo
(incl. explanation)

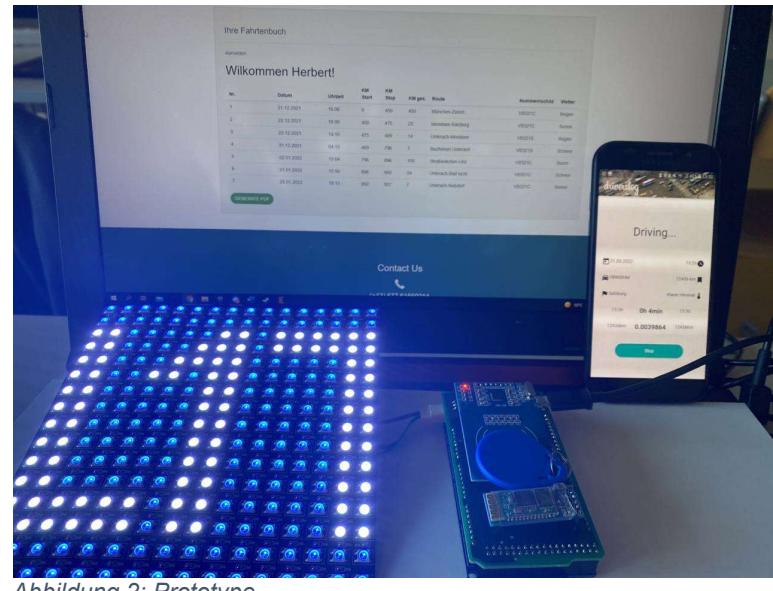


Abbildung 2: Prototype

Participation in
Competitions
Awards

Accessibility of
Diploma Thesis

Library of the HTBLuVA Salzburg

Approval
(Date / Sign)

Examiner

Head of College
Head of Department

Vorwort

Seit Jahren gibt es die Möglichkeit den Führerschein mit weniger Fahrstunden, dafür mit mehr Praxis mit den Eltern zu machen. Dafür muss eine gewisse Kilometeranzahl erreicht und genauestens dokumentiert werden. Dies ist mühsam und nimmt viel Zeit in Anspruch. In unserem Team haben alle diesen Weg zum Führerschein gewählt, aus diesem Grund haben wir es uns zur Aufgabe gemacht eine digitale Lösung zu finden. Ursprünglich kam die Projektidee von unserem Betreuer und Klassenvorstand Prof. Dipl.- Ing Siegbert Schrempf. Gemeinsam mit ihm haben wir die Idee entwickelt und driverslog geschaffen.

Danksagung

Unser herzlicher Dank gilt unserem Projektbetreuer und Klassenvorstand Prof. Dipl.-Ing. Siegbert Schrempf, der uns während der Arbeit an unserem Projekt hilfreich zur Seite stand.

Zusätzlich möchten wir uns bei sämtlichen Werkstättenlehrern, darunter FL. Ing. Dipl.-Päd. Robert Pöttinger und FL. Ing. Dipl.- Päd. Rudolf Lackner bedanken, die uns bei der Fertigung unserer Hardwarekomponenten tatkräftig zur Seite standen.

Inhalt

1	Überblick	15
2	Systemspezifikation.....	16
2.1	Zielbestimmungen	16
2.1.1	Musskriterien.....	16
2.1.2	Wunschkriterien	16
2.1.3	Abgrenzungskriterien	16
2.2	Produkteinsatz.....	17
2.2.1	Anwendungsbereiche	17
2.2.2	Zielgruppen.....	17
2.2.3	Betriebsbedingungen	17
2.3	Produktfunktionen.....	18
2.4	Produktdaten	19
2.5	Produktleistungen.....	20
2.6	Benutzeroberfläche	21
2.6.1	Login/Sign up-Screen	21
2.6.2	Main-Screen.....	21
2.6.3	Neue Fahrt erstellen	23
2.6.4	Webapplikation Startbildschirm.....	25
2.6.5	Registrieren/ Login.....	25
2.7	Qualitätszielbestimmungen	27
2.8	Globale Testszenarien und Testfälle	28
2.9	Entwicklungsumgebung.....	29
2.10	Software	29
2.10.1	Hardware	29
2.10.2	Orgware	29
3	Organisation - Projektmanagement.....	30
3.1	Projektteam	30

3.2 Individuelle Aufgabenstellungen inkl. Arbeits- und Terminplan	31
3.2.1 Niklas Parhammer	31
3.2.2 Philip Wienerroither.....	32
3.2.3 Daniel Wipplinger.....	33
4 Grundlagen und Methoden.....	34
4.1 Sensor Interfaces	35
4.1.1 Parallele Schnittstellen.....	35
4.1.2 Serielle Schnittstellen.....	37
4.1.2.1 Synchrone Schnittstellen	38
4.1.2.2 Asynchrone Schnittstellen	38
4.1.3 Vergleich serielle/parallele Interfaces	39
4.1.4 UART	40
4.1.5 HC-05 Bluetooth-Modul	41
4.1.5.1 Programmierung.....	42
4.1.6 I ² C	43
4.1.6.1 Datenübertragung	44
4.1.6.2 Adressbyte	45
4.1.6.3 Multi-Master Betrieb	46
4.1.7 SPI	47
4.1.7.1 Slave select (SS).....	47
4.1.7.2 Operations Modi	48
4.1.7.3 Senden von Daten.....	49
4.1.8 RC522 RFID Modul.....	49
4.1.8.1 Funktionsweise.....	50
4.1.8.2 Programmierung.....	51
4.1.9 1-Wire	52
4.1.9.1 Spannungsversorgung der Slaves	53
4.1.9.2 Datenübertragung	53

4.1.10	LED-Matrix.....	54
4.1.10.1	Funktionsweise	54
4.1.10.2	Programmierung.....	56
4.1.10.3	Leistungsverbrauch	57
4.2	App-Entwicklung mit Flutter.....	58
4.2.1	Was ist Flutter?	58
4.2.2	Vor-/Nachteile von Flutter	58
4.2.3	Installation.....	59
4.2.4	Ein neues Projekt erstellen	59
4.2.5	App testen.....	60
4.2.6	Packages und Libraries	60
4.2.6.1	Finden von Packages	60
4.2.6.2	Einbinden eines Packages	60
4.2.6.3	Entfernen von Packages	61
4.2.7	Widgets	62
4.2.7.1	Text-Widget	62
4.2.7.2	Container-Widget	62
4.2.7.3	Center-Widget	63
4.2.7.4	ElevatedButton-Widget.....	63
4.2.7.5	Column/Row-Widget	64
4.2.8	App-Icon	65
4.2.9	Login-System.....	66
4.2.9.1	Allgemein.....	66
4.2.9.2	Login/Sign up-Benutzeroberfläche	66
4.2.9.3	Login-Funktion.....	66
4.2.9.4	Sign up-Funktion	67
4.2.9.5	RememberMe.....	68
4.2.10	Bluetooth Verbindung	69

4.2.10.1	Allgemein.....	69
4.2.10.2	Bluetooth Versionen	69
4.2.10.3	Bluetooth-Funktionen in der App	70
4.2.11	Fahrdatenermittlung.....	74
4.2.11.1	Allgemein.....	74
4.2.11.2	Trip-Klasse	74
4.2.11.3	Start einer Fahrt.....	74
4.2.12	Kommunikation mit der Datenbank.....	77
4.2.12.1	Allgemein.....	77
4.2.12.2	GET-Request.....	77
4.2.12.3	POST-Request	77
4.3	Datenbank	78
4.3.1	Datenbank Grundlagen.....	78
4.3.1.1	Datenbankmanagementsystem.....	78
4.3.2	Wahl der Datenbank	78
4.3.2.1	Vergleich SQLite und MySQL.....	78
4.3.3	SQLite.....	79
4.3.3.1	SQLite Browser	79
4.3.3.2	Datenbank erstellen	80
4.3.4	Relationale Datenbank.....	80
4.3.4.1	Primary Key.....	80
4.3.4.2	Foreign Key	81
4.3.4.3	Normalformen.....	81
4.3.4.4	Beziehungen	84
4.3.4.5	Driverslog Beziehungen	85
4.3.4.6	Entity Relationship Modell (ERM).....	86
4.3.4.7	Erstellen der Tabelle	86
4.3.4.8	Structured Query Language (SQL).....	87

4.3.5	Web-Applikation.....	88
4.3.5.1	XAMPP.....	89
4.3.5.2	Scriptssprachen	90
4.3.5.3	PHP Basics	92
4.3.5.4	HTML.....	92
4.3.5.5	CSS.....	93
4.3.5.6	HTML und PHP	94
4.3.5.7	Kommunikation mit der Datenbank	94
4.3.5.8	Query.....	95
4.3.6	Benutzerverwaltung	95
4.3.7	REST API	98
5	Ergebnisse – Abnahme	100
6	Literaturverzeichnis	102
7	Verzeichnisse der Abkürzungen, Abbildungen und Tabellen	106
7.1	Abkürzungen	106
7.2	Abbildungen	107
7.3	Tabellen.....	111
8	Begleitprotokoll.....	113
8.1	Niklas Parhammer.....	113
8.2	Philip Wienerroither	114
8.3	Daniel Wipplinger	115

1 Überblick

Die Smartphone-App ermöglicht die Verbindung zwischen dem Smartphone und dem Identifikationsmodul, welches zur Identifikation des Fahrers und zur Aktivierung des L17-Schildes dient. Die Fahrdaten werden vom Handy auf einen Server geladen, auf welchen man per Smartphone oder Computer zugreifen kann.

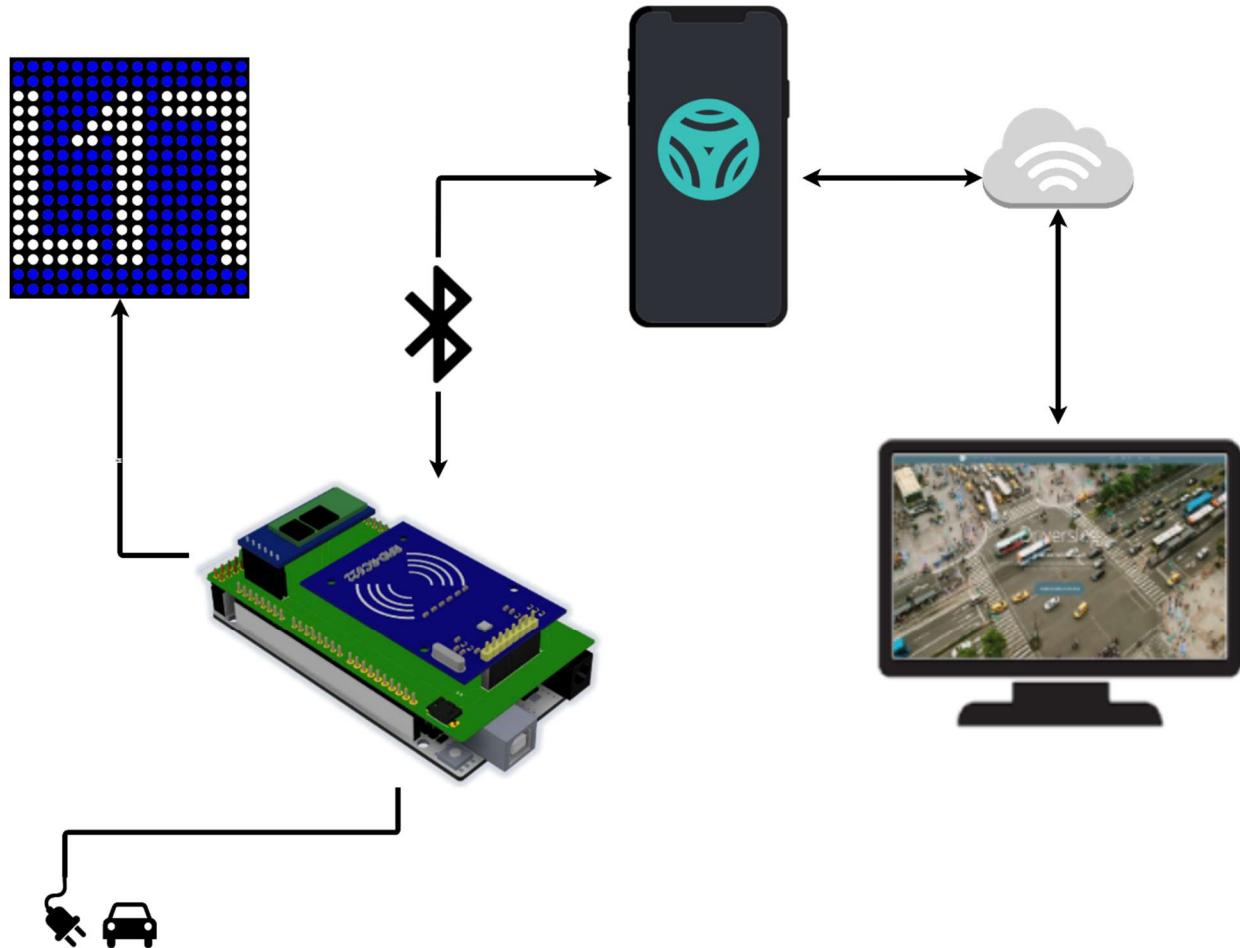


Abbildung 3: Grobentwurf des Projekts, in der Mitte das μ C-System

2 Systemspezifikation

2.1 Zielbestimmungen

2.1.1 Musskriterien

- Das Identifikationsmodul soll den Fahrschüler eindeutig identifizieren.
- Es soll ein Fahrtenprotokoll erstellt werden, welches im PDF-Format zu exportieren ist.
- Es soll während der Fahrt ein L17-Symbol, auf einer Matrix an der Heckscheibe, angezeigt werden.

2.1.2 Wunschkriterien

- Die User-Account-Bearbeitung soll in der App möglich sein
- Die Matrixanzeige soll um ein Frontscheibenmodul erweiterbar sein.

2.1.3 Abgrenzungskriterien

- Es muss ein Smartphone als zentrales Gerät verwendet werden.
- Auf den Fahrzeugkilometerstand wird nicht zugegriffen.
- Das Smartphone muss eine durchgehende Internetverbindung aufweisen.
- Die App ist nur mit Android kompatibel.

2.2 Produkteinsatz

2.2.1 Anwendungsbereiche

Mithilfe des Projekts werden Manipulationen und Verwirrungen während Übungsfahrten vorgebeugt. Zudem kann der Fortschritt jederzeit am Smartphone angezeigt werden.

2.2.2 Zielgruppen

Dieses Projekt dient Fahrschülern zur Erleichterung im Führerscheinübungsprozess, sie können damit ihren Fortschritt und ihre Termine besser im Auge behalten.

2.2.3 Betriebsbedingungen

- Mikrocontrollerplatine:

Sie kann in jeder Umgebung, wo eine stabile Bluetooth Verbindung und eine Spannungsversorgung per Mikro-USB möglich ist, verwendet werden. Zielumgebung ist das Automobil.

- Smartphone App:

Es wird eine dauerhafte Internetanbindung benötigt. Die Fahrfunktionen können nur in unmittelbarer Nähe des Identifikationsmoduls, nach durchgeföhrter Verifizierung, verwendet werden.

- Web-Applikation:

Die Web-Applikation benötigt eine Internetanbindung.

2.3 Produktfunktionen

/F0010/ Übertragung der Daten des Verifikationsmoduls per Bluetooth

Das Smartphone soll sich mittels der Smartphone-App mit dem Identifikationsmodul verbinden und Daten austauschen.

/F0020/ Anzeige eines L17-Symbols auf der LED-Matrix

Mittels Mikrocontroller soll bei eingehendem Bluetooth Signal, an der LED-Matrix, ein L17-Symbol angezeigt werden.

/F0030/ Ermittlung von Fahrdaten

Während des Fahrens sollen relevante Daten wie Datum, Uhrzeit, Kilometer, Strecke und das Wetter ermittelt werden.

/F0040/ Übertragung der Daten des Smartphones in die Datenbank

Mithilfe der Smartphone-Applikation soll das Smartphone Daten in einer externen Datenbank abspeichern und beim nächsten Start wieder aufrufen.

/F0050/ Visualisierung der Fahreinträge und Termine

Die persönlichen Fahreinträge sowie die eingetragenen Termine sollen in der Web-Applikation und in der Smartphone-App ersichtlich sein.

/F0060/ Exportieren eines ausgefüllten Formulars

An einem handelsüblichen Computer, unabhängig des Betriebssystems, soll es möglich sein, seine Fahreinträge in ein PDF-Dokument zu exportieren.

2.4 Produktdaten

/D010/ Benutzerdaten

- ID (eindeutig)
- Benutzername
- Passwort
- Vorname
- Nachname
- Geburtsdatum
- RFID-Code

/D020/ Fahrdaten

- ID (eindeutig, Verknüpfung mit Benutzerdaten)
- Fahrtensummer (eindeutig pro Benutzer)
- Datum
- Zeit beim Start
- Zeit am Ende
- Kilometerstand beim Start
- Kilometerstand am Ende
- Gefahrene Kilometer
- Strecke
- Nummernschild
- Wetter

2.5 Produktleistungen

/L0010/ Benutzerfreundlichkeit

Die Bedienung soll möglichst einfach und selbsterklärend sein.

/L0020/ Fremdstörfestigkeit

Die Funktionalität soll durch die Störungen in einem geschlossenen Auto nicht beeinträchtigt werden.

/L0030/ Genauigkeit

Alle zur Fahrt relevanten Daten sollen so genau wie möglich sein.

/L0040/ Manipulationssicherheit

Die Daten sollen schwer zu manipulieren sein.

2.6 Benutzeroberfläche

2.6.1 Login/Sign up-Screen

Die Bedienung erfolgt über die driverslog-App, welche nach dem Start ein Login-Fenster zeigt. In jenem Fenster besteht die Möglichkeit aus Login und Sign-up zu wählen. Während man sich beim Login, wie in Abbildung 4 ersichtlich, mit einem bereits bestehenden Account anmelden kann, ist es möglich im Sign-up-Fenster (Abbildung 4) einen neuen Account zu erstellen. Das Erscheinungsbild der gesamten App basiert auf den Geräteeinstellungen des Dark Modes.

Nach dem Drücken des „→“-Buttons wird man, sofern die Anmeldung oder Registrierung erfolgreich war, auf die Hauptseite der App weitergeleitet.

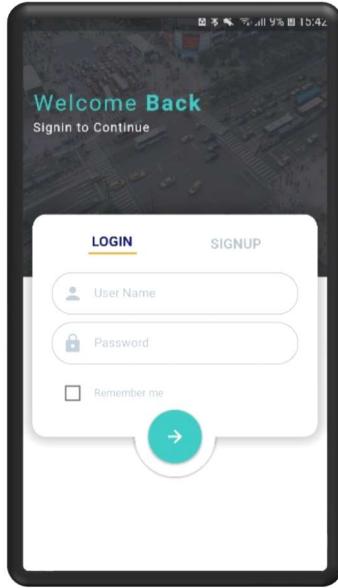


Abbildung 4: Login-Screen

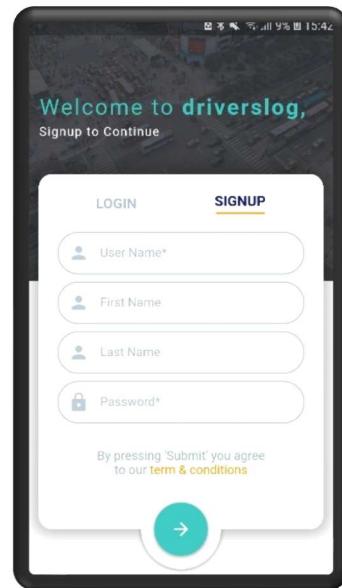


Abbildung 5: Sign-up-Screen

2.6.2 Main-Screen

Die Hauptseite besteht aus den Tabs Home, Route, Calendar und Profil, wie in den folgenden Abbildungen ersichtlich. Zwischen den Tabs kann mithilfe der Navigationsleiste am unteren Bildschirmrand navigiert werden.



Abbildung 6: Home-Screen

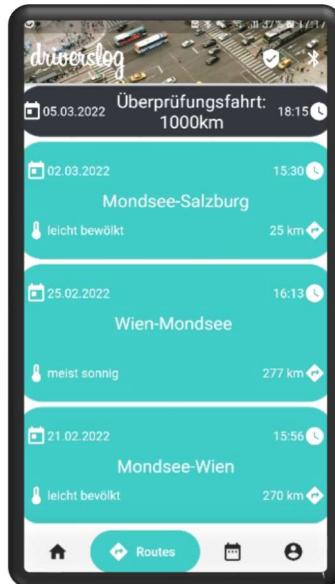


Abbildung 7: Routes-Screen



Abbildung 8: Calendar-Screen

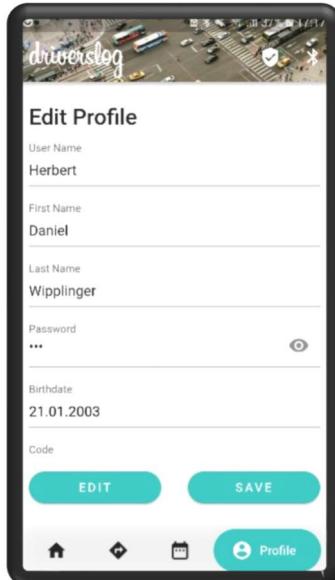


Abbildung 9: Account-Screen

Auf dem „Home“-Screen in Abbildung 6 wird der persönliche Fortschritt veranschaulicht.

Weiters stellen Icons im rechten oberen Bildschirmrand den Verifikationsstatus(links) und die Verbindung mit dem Verifikationsmodul(rechts) dar.

Der zweite Tab „Routes“ (Abbildung 7) zeigt sämtliche bereits gefahrene Strecken und eingetragene Termine.

In der Calendar-Ansicht in Abbildung 8 werden ebenfalls alle Strecken und Termine angezeigt. Weiters kann mit einem Klick auf das „+“ ein neuer Termin eintragen werden.

Abbildung 9 zeigt den letzten Tab, welcher zur Bearbeitung des Accounts dient.

2.6.3 Neue Fahrt erstellen

Um eine neue Fahrt zu erstellen, muss zuerst das Identifikationsmodul per Bluetooth mit dem Smartphone verbunden sein. Sofern das Bluetooth-Symbol noch nicht grün ist, wie in Abbildung 6 markiert, kann man mit einem Klick auf das Symbol, die Liste mit den Bluetooth-Geräten in der Nähe öffnen und das gewünschte Gerät auswählen. Sollte eine Verbindung aufgebaut werden können, wird man auf den Home-Screen zurückgeleitet, mit dem Unterschied, dass das Bluetooth-Symbol grün ist:



Abbildung 10: Bluetooth verbunden

Als nächstes muss sich der Fahrer mit seinem RFID-Chip verifizieren, dazu wird der Chip an das Identifikationsmodul gehalten. Wenn der Code des Chips mit dem Code, welcher im Account hinterlegt ist, übereinstimmt, wird das Verifikationssymbol ebenfalls grün, ansonsten färbt es sich rot:



Abbildung 11: erfolgreiche Verifikation



Abbildung 12: fehlgeschlagene Verifikation

Wenn beide Symbole grün sind, kann eine neue Fahrt gestartet werden, dies erfolgt durch das Drücken des „start new route“-Buttons am Home-Screen (Abbildung 6), wodurch man zu einem Eingabefenster weitergeleitet wird.

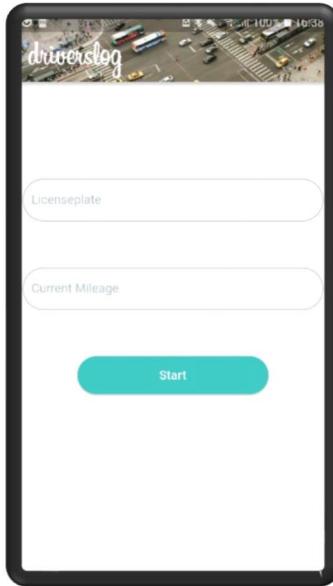


Abbildung 13: neue Fahrt: Eingabefenster

In dem Eingabefenster (Abbildung 13) müssen Autokennzeichen und Kilometerstand eingegeben werden. Nach dem Klicken auf „Start“ wird man auf die Live-Ansicht der Fahrt weitergeleitet, wo aktuelle Daten zur Fahrt angezeigt werden:

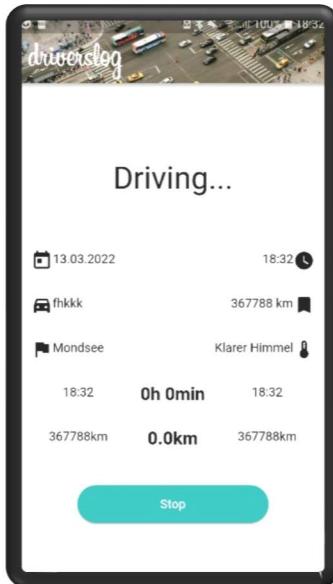


Abbildung 14: Live-Fahrdatenansicht

Mit einem Klick auf den „Stop“-Button wird der Fahrteneintrag gespeichert und man wird auf den Main-Screen zurückgeleitet.

2.6.4 Webapplikation Startbildschirm

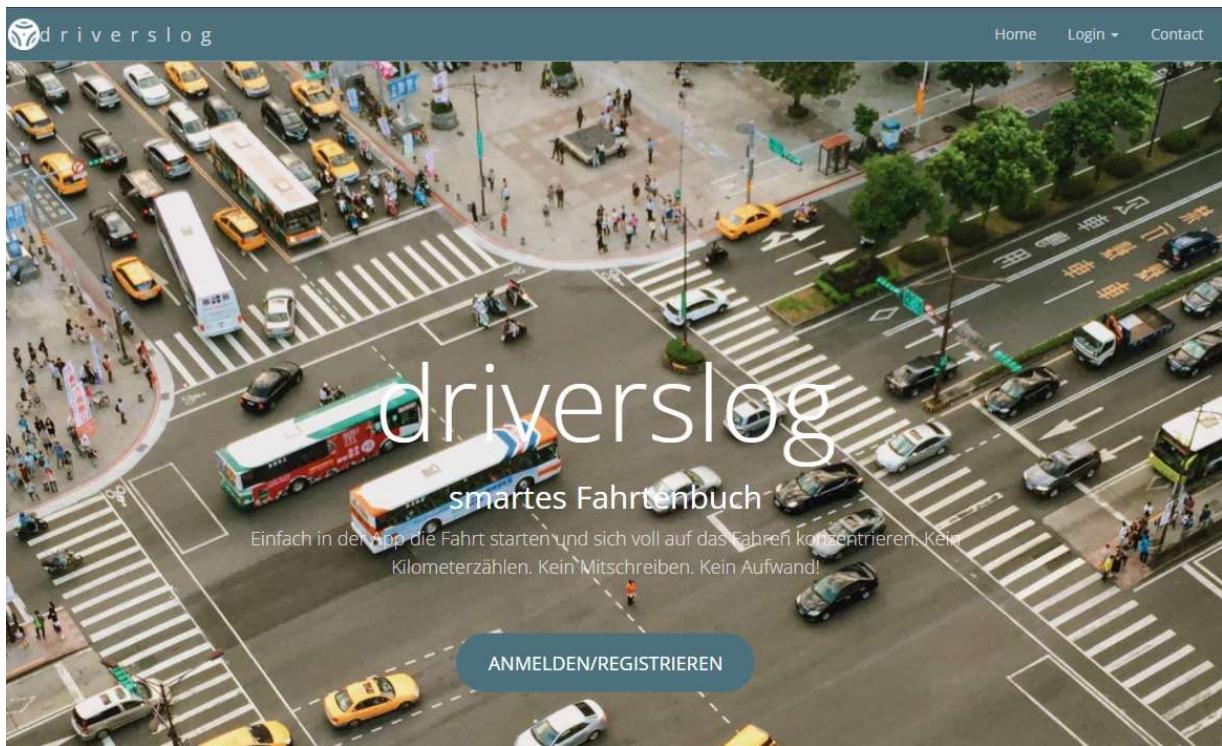


Abbildung 15: Webapplikation Startbildschirm

Wird die Website aufgerufen landet man auf dem Startbildschirm. Hier gibt es die Möglichkeit zu wählen, ob man sich Anmelden oder Registrieren möchte. Außerdem ist eine kleine Zusammenfassung zu finden, in der erklärt wird, was driverslog ist.

2.6.5 Registrieren/ Login

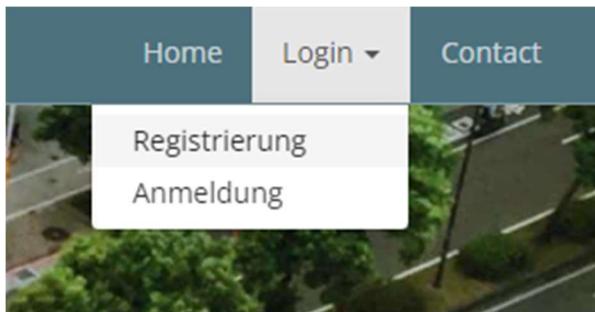


Abbildung 16: Registrieren: neuen Account erstellen

Wenn noch kein Account vorhanden ist, gibt es unter dem Punkt Login (Abbildung 16) die Möglichkeit einen zu erstellen. Wird auf Registrieren geklickt wird man zu einem Formular weitergeleitet, welches die Daten erfasst (Abbildung 17)

Registrieren Sie sich bei driverslog	
Schon Registriert? Melden Sie sich an.	
Registration	
Username	
<input type="text"/>	
Password	
<input type="password"/>	
Firstname	
<input type="text"/>	
Lastname	
<input type="text"/>	
Birthdate (DD.MM.YYYY)	
<input type="text"/>	
ANMELDEN	

Abbildung 17: Formular zum Registrieren

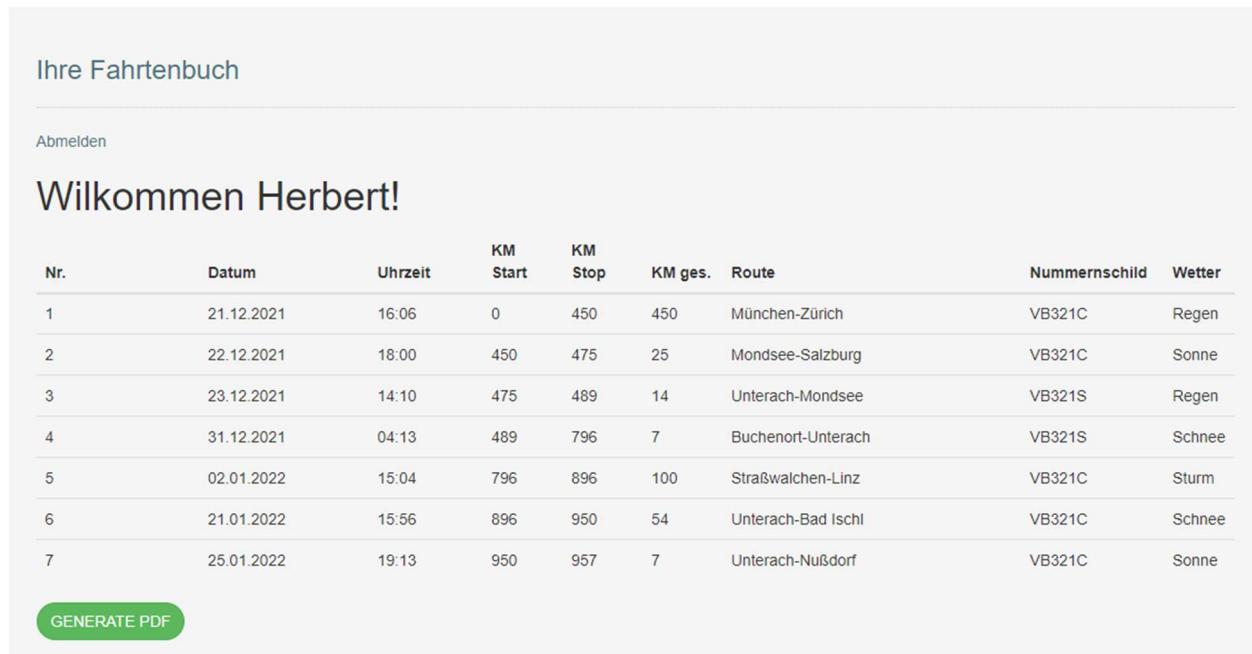


The image shows a login form titled "Login Fahrtenbuch". It includes a message "Noch kein Mitglied? Melden Sie sich jetzt an!", a light blue "Login" button, a "Username" input field, a "Password" input field, and a dark teal "LOGIN" button with a key icon.

Ist der Account erstmalig erstellt, kann man sich entweder unter dem Punkt Login (Abbildung 16), oder direkt bei dem Button (Anmelden/Registrieren) in der Mitte des Startbildschirms (Abbildung 15), anmelden.

Abbildung 18: Login Formular Website

Nachdem man seine Anmelddaten eingegeben hat, wird man auf die Home-Seite (Abbildung 19) weitergeleitet. Auf dieser Seite wird der gesamte Fahrtenfortschritt dargestellt. Außerdem befindet sich in der linken unteren Ecke ein Button, der die Möglichkeit bietet, sein Fahrtenprotokoll im PDF-Format zu exportieren.



The image shows the home page titled "Ihre Fahrtenbuch". It features a "Abmelden" link, a greeting "Wilkommen Herbert!", and a table displaying driving history. A green "GENERATE PDF" button is located at the bottom left of the table.

Nr.	Datum	Uhrzeit	KM Start	KM Stop	KM ges.	Route	Nummernschild	Wetter
1	21.12.2021	16:06	0	450	450	München-Zürich	VB321C	Regen
2	22.12.2021	18:00	450	475	25	Mondsee-Salzburg	VB321C	Sonne
3	23.12.2021	14:10	475	489	14	Unterach-Mondsee	VB321S	Regen
4	31.12.2021	04:13	489	796	7	Buchenort-Unterach	VB321S	Schnee
5	02.01.2022	15:04	796	896	100	Straßwalchen-Linz	VB321C	Sturm
6	21.01.2022	15:56	896	950	54	Unterach-Bad Ischl	VB321C	Schnee
7	25.01.2022	19:13	950	957	7	Unterach-Nußdorf	VB321C	Sonne

Abbildung 19: Home Seite mit Darstellung des Fahrtenfortschritts

2.7 Qualitätszielbestimmungen

	Sehr wichtig	Wichtig	Weniger wichtig	Unwichtig
Benutzerfreundlichkeit	X			
Fremdstörfestigkeit			X	
Genauigkeit		X		
Manipulationssicherheit		X		

2.8 Globale Testszenarien und Testfälle

/T0010/ Übertragung der Daten des Verifikationsmoduls per Bluetooth

Die Smartphone-App kann eine Bluetooth-Verbindung herstellen. Das Senden und Empfangen funktioniert in beide Richtungen.

/T0020/ Anzeige eines L17-Symbols auf der LED-Matrix

Bei entsprechendem Bluetooth-Empfang wird auf der Matrix das Symbol angezeigt.

/T0030/ Ermittlung von Fahrdaten

Während des Fahrens werden die spezifizierten Fahrdaten ermittelt.

/T0040/ Übertragung der Daten des Smartphones in die Datenbank

Die Daten werden nach dem Beenden der Fahrt in die Datenbank gespeichert und beim App-Start wieder aufgerufen.

/T0050/ Visualisierung der Fahreinträge und Termine

Die persönlichen Fahrten und eingetragenen Termine werden in der Web-Applikation und in der Smartphone-App angezeigt.

/T0060/ Exportieren eines ausgefüllten Formulars

Es kann ein PDF-Dokument mit den Fahrten exportiert werden.

2.9 Entwicklungsumgebung

2.10 Software

- EAGLE: zum Entwerfen der Platine (V:9.6.2)
- Arduino IDE: zur Programmierung des Arduino MEGA (V:1.8.16)
- Autodesk Fusion: zum Erstellen der 3D Modelle (V:2.0.12164)
- Visual Studio Code: zum Programmieren von Smartphone-App und Web-Applikation (V:1.65.2)
- Android Studio: als Emulator zum Testen der App (V:2020.3.1)
- Postman: zum Testen der http-Anfragen (V:9.0.8)
- XAMPP: zum hosten eines lokalen Servers (V:3.3.0)
- DB Browser (SQLite): zum Bearbeiten der Datenbank (V:3.12.2)

2.10.1 Hardware

- Arduino MEGA 2560 (Mikrocontroller)
- HC-05 (Bluetooth-Kommunikation)
- RC522 (RFID Identifikation)
- Android Smartphone (Android V:12)

2.10.2 Orgware

- GitLab: zur Versionsverwaltung (V:14.9)
- Microsoft Teams: zur Kommunikation und als Cloud-Speicher (V: 1.5.00.2164)

3 Organisation - Projektmanagement

3.1 Projektteam

NAME	INDIVIDUELLE THEMENSTELLUNG	KLASSE	ARBEITSAUFWAND
NIKLAS PARHAMMER	Ansteuerung von Sensor-Interfaces	5AHEL	180 Stunden
PHILIP WIENERROITHER	Entwicklung von Smartphone-Applikationen	5AHEL	180 Stunden
DANIEL WIPPLINGER	Modellierung von Datenbanken	5AHEL	180 Stunden

Tabelle 1: Aufgabenverteilung

3.2 Individuelle Aufgabenstellungen inkl. Arbeits- und Terminplan

3.2.1 Niklas Parhammer

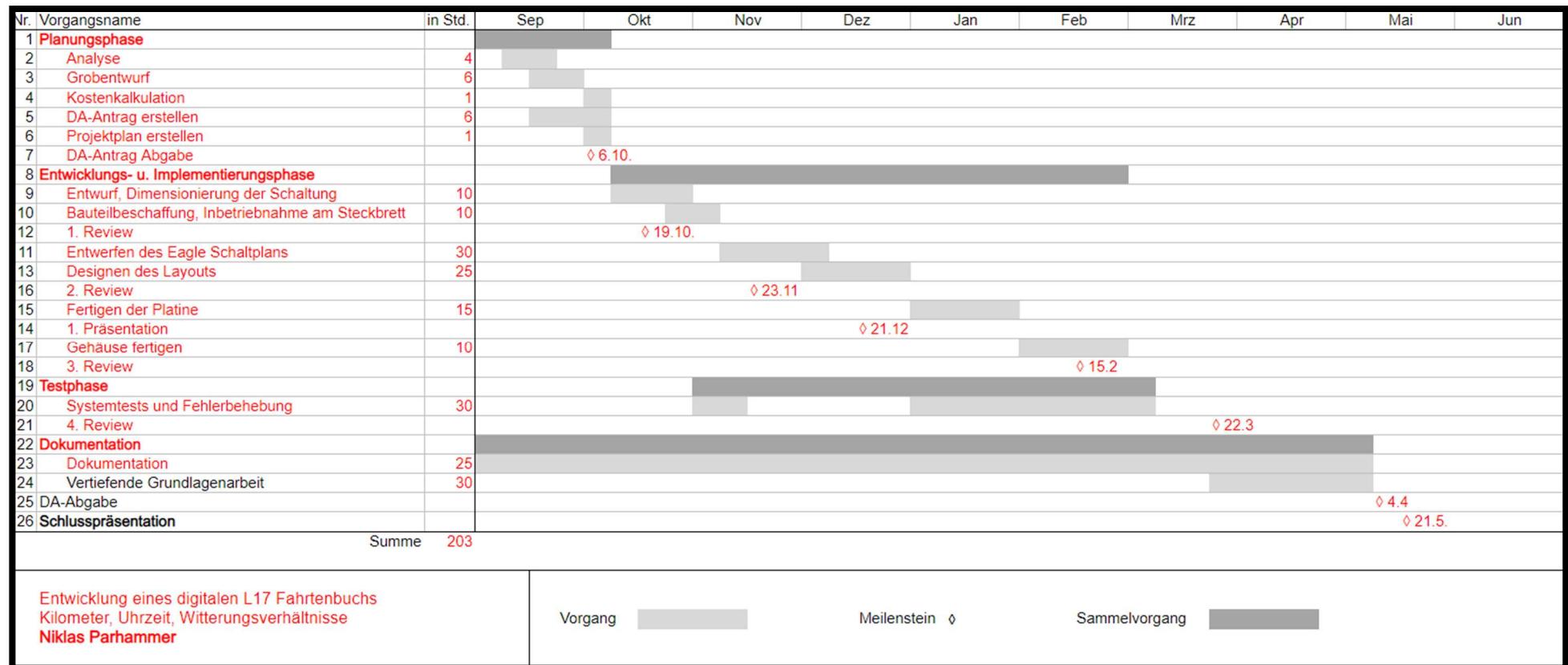


Abbildung 20: GANT-Diagramm Niklas Parhammer

3.2.2 Philip Wienerroither

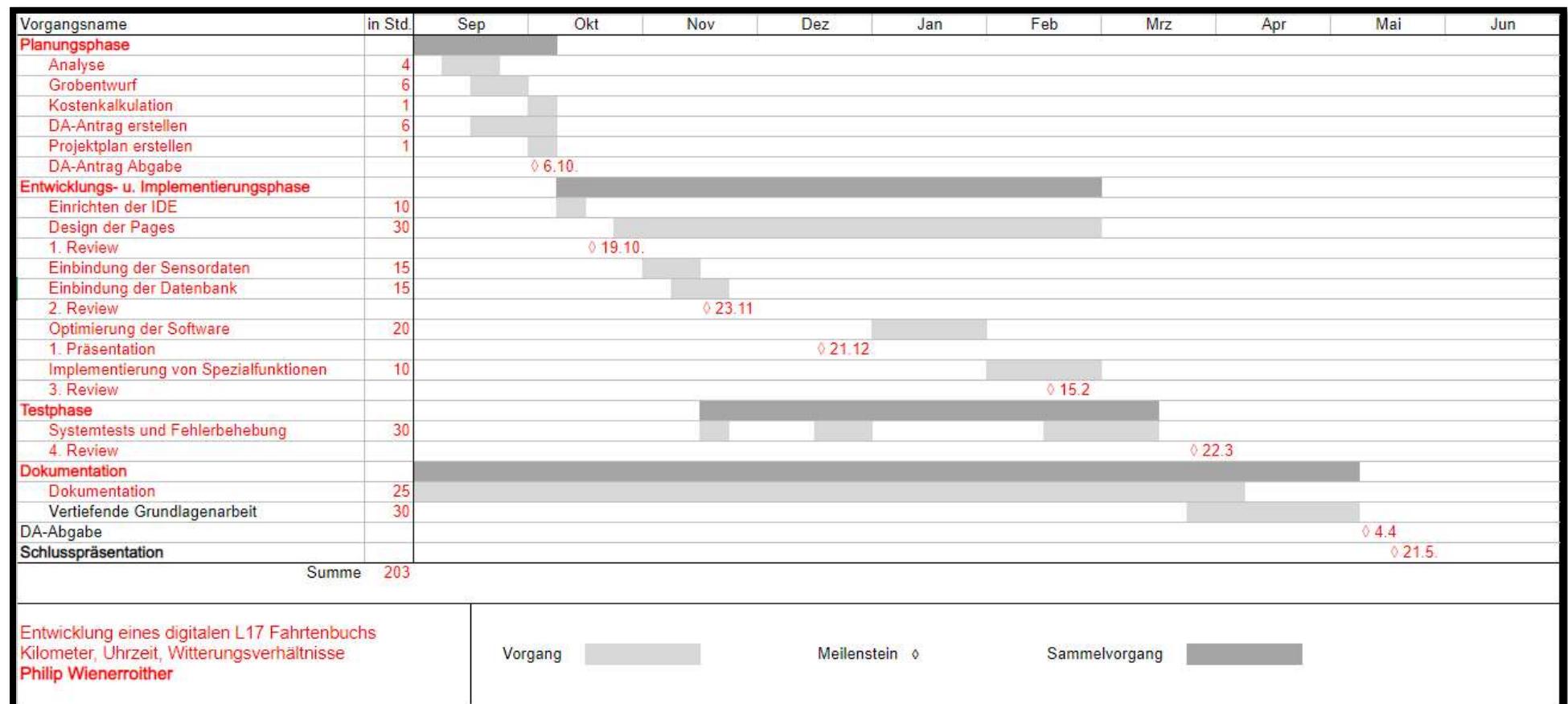


Abbildung 21: GANT-Diagramm Philip Wienerroither

3.2.3 Daniel Wipplinger



Abbildung 22: GANT-Diagramm Daniel Wipplinger

4 Grundlagen und Methoden

Im folgenden Blockschaltbild ist das Projekt in 3 Teilbereiche unterteilt und jeweils einem Teammitglied farblich zugeteilt:

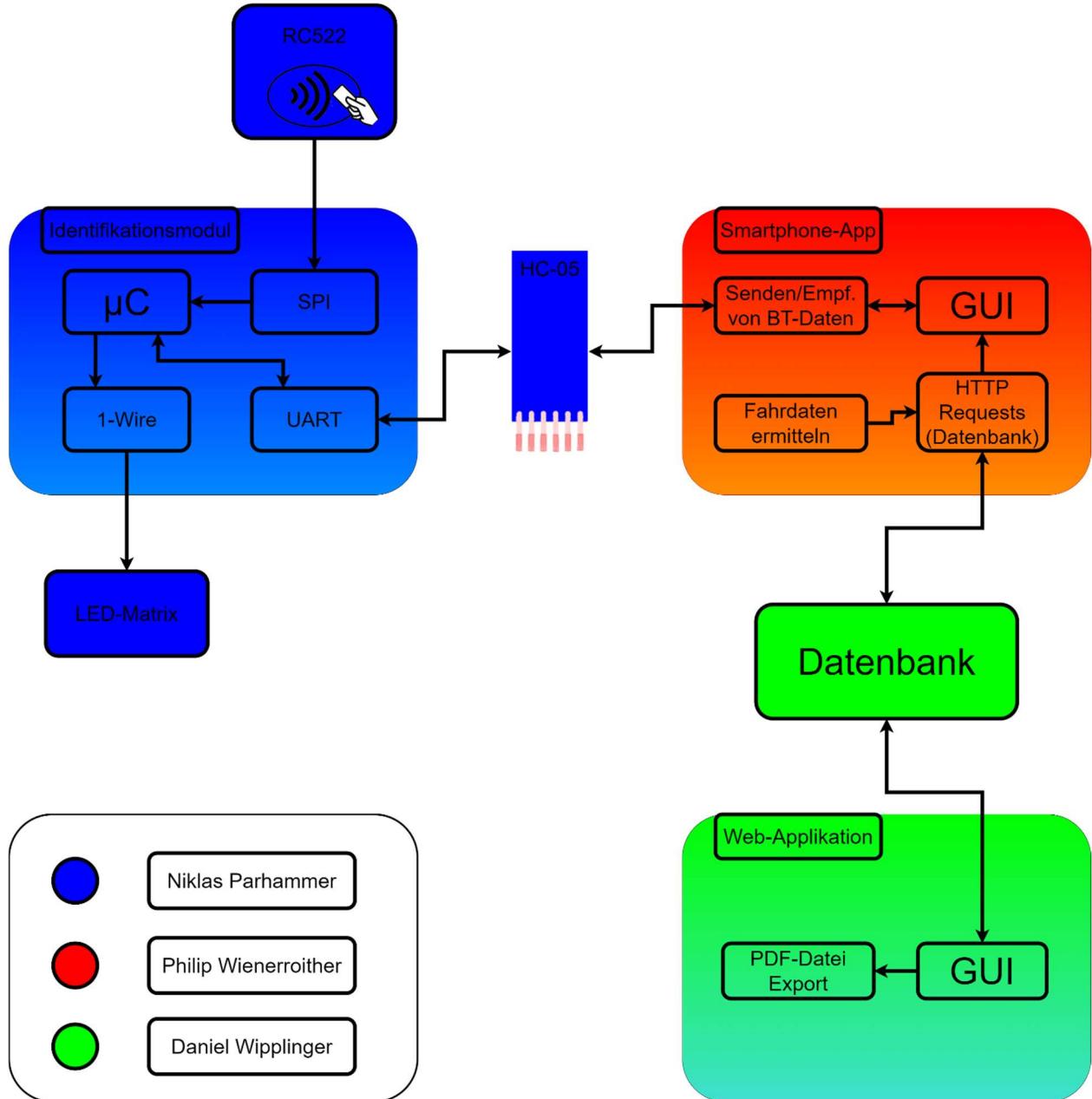


Abbildung 23: Aufgabenbereiche der einzelnen Teammitglieder sind eingefärbt

4.1 Sensor Interfaces

Um eine Kommunikation zwischen zwei verschiedenen Systemen zu ermöglichen, werden Interfaces (Deutsch: Schnittstellen) benötigt. Über gemeinsame Eigenschaften und Protokolle werden Standards, für eine Übertragung elektrischer Signale bzw. für eine Kommunikation festgelegt.

Sensor Interfaces regeln also die Kommunikation zwischen beispielsweise einem Mikrocontroller und einem Sensor. Da je nach Anwendungsgebiet, die Anforderungen an die Kommunikation variieren, haben sich im Laufe der Zeit verschiedene Interfaces etabliert, die jeweils andere Eigenschaften aufweisen. Grundsätzlich gibt es zwei Arten von Schnittstellen: Parallele und Serielle Schnittstellen. [vgl. [1]]

4.1.1 Parallele Schnittstellen

In digitalen Systemen werden nur selten einzelne Bits verarbeitet. Stattdessen werden Bits zu 8, 16, 32, 64 oder mehr gruppiert, um sogenannte Datenworte zu erstellen. Da Daten meistens in Gruppen verarbeitet werden, kann es sinnvoll sein diese nicht hintereinander, sondern parallel zu versenden.

Dies wird durch mehrere Datenleitungen die parallel zwischen den Systemen verlaufen erreicht. Die Länge eines Datenworts entspricht dabei der Anzahl an den parallel verlaufenden Leitungen. In der Praxis werden dafür Flachbandkabel oder gruppierte Leiterbahnen auf Platinen verwendet.

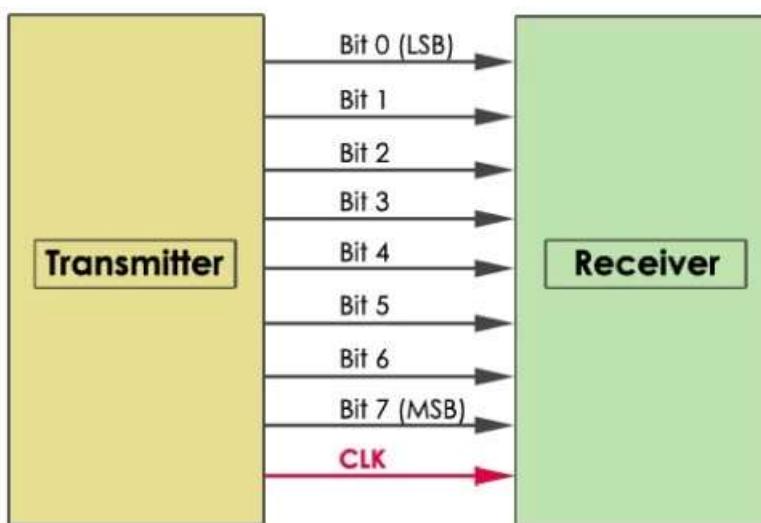


Abbildung 24: Paralleles Interface [vgl. [2]]

Elektrische Signale breiten sich auf den Leitungen mit einer bestimmten Dauer aus. Wenn also Leitungen unterschiedlich lang sind, kommen die versendeten Bits zu verschiedenen Zeitpunkten beim Empfänger an und führen zu Übertragungsfehlern. Leitungsunterschiede ergeben sich dann, wenn eine Leitung abbiegt und nicht mehr gerade verläuft. Die inneren Leitungen werden kürzer als die äußeren, wie man in Abbildung 25 sehen kann.

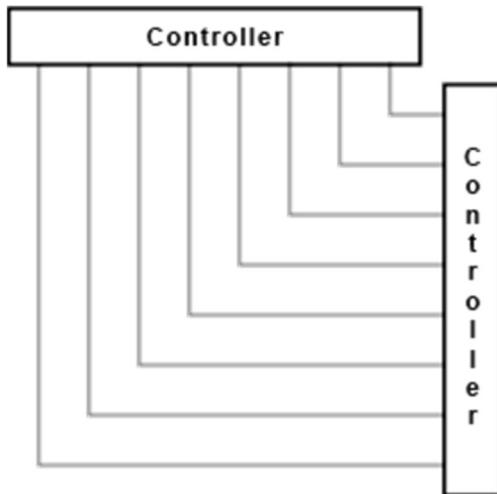


Abbildung 25: Innere Leitungen sind kürzer als Äußere [vgl. [1]]

Zusätzlich dazu führen parasitäre kapazitive und induktive Effekte zwischen den Leiterbahnen zu zeitlichen Verschiebungen der Signalflanken, was bedeutet, dass die Signale auf den Leitungen zu verschiedenen Zeitpunkten am Ziel ankommen.

Um diese Effekte zu vermeiden, wird bei parallelen Schnittstellen darauf geachtet Leitungen und Leitungsbahnen exakt gleich lang zu machen. Auf Platinen erreicht man das nur mit einer mäanderförmigen Leitungsführung (Abbildung 26), die aber sehr viel Platz benötigt.

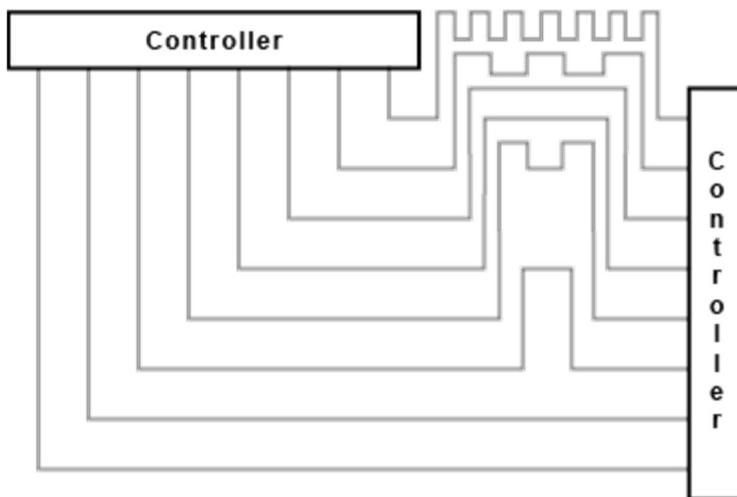


Abbildung 26: Mäanderförmige Leitungsführung zwischen zwei Controllern [vgl. [1]]

Die Laufzeitdifferenzen können durch kürzere Übertragungstrecken, geringere Taktraten und Masseleitungen zwischen den Leitern ausgemerzt werden. Letzteres benötigt aber auch viel Platz und führt zu dicken, unflexiblen Flachbandkabeln. Hohe Taktfrequenzen sind nur über kurze Strecken möglich.

Die parallele Schnittstelle wird heute nur noch in älteren Peripheriegeräten (IEEE 1284: Druckerschnittstelle wie in Abbildung 27) bzw. Rechnern verwendet und gilt als veraltet. [vgl. [1]]



Abbildung 27: IEEE 1284 Druckerschnittstelle

4.1.2 Serielle Schnittstellen

Im Unterschied zu parallelen Schnittstellen werden bei seriellen Schnittstellen, Bits nicht gleichzeitig, sondern hintereinander (Bit für Bit) übertragen. Es wird sich also auf die zeitliche Position der Bits auf dem Leiter konzentriert und nicht auf die räumliche Position der Bits innerhalb einer Gruppe von Drähten. Dafür sind in der geringsten Form nur eine Datenleitung nötig. Oft werden aber mehr Leitungen verwendet, um zum Beispiel eine Vollduplex Verbindung zu erreichen, gewöhnlich aber nie mehr als vier.

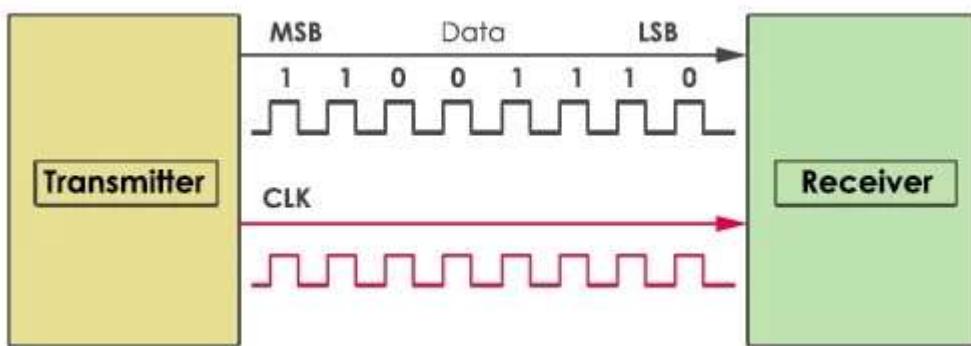


Abbildung 28: Serielle Übertragung, Bit für Bit [vgl. [2]]

Die Kommunikation funktioniert auch über längere Distanzen, die Verkabelung und Implementierung ist einfach und der Kabelaufwand überschaubar. Aufgrund der

einfachen Handhabung. Weiters können serielle Schnittstellen in asynchrone und synchrone eingeteilt werden.

[vgl. [3]]

4.1.2.1 Synchrone Schnittstellen

Synchrone Schnittstellen verfügen über ein externes Taktsignal, dass von allen Systemen genutzt wird. Dieses Taktsignal signalisiert wann ein Bit auf der Datenleitung anliegt. Somit ist die Synchronisation von Sender und Empfänger gegeben, solange das Taktsignal bei allen Teilnehmern gleich ist. Durch unerwünschte physikalische Effekte die vor allem bei hohen Taktraten durch ungünstige Leitungslänge, Temperaturschwankungen, Materialfehler oder kapazitiver Kopplung auftreten, kann es zum Taktversatz (Englisch: clock skew) kommen. Der Taktversatz beschreibt das Phänomen, dass durch einen Zeitversatz des Clock-Signals, die Signale nicht mehr gleichzeitig am Empfänger ankommen. Durch einen mäanderförmigen Aufbau der Leitungen kann man den Taktversatz bei hohen Frequenzen minimieren.

[vgl. [3] [4]]

4.1.2.2 Asynchrone Schnittstellen

Asynchrone Interfaces verfügen über keinen Takt und müssen über andere Parameter gesteuert werden. Alle Datenwörter werden unabhängig voneinander versendet und benötigen zur Synchronisation von Sender und Empfänger Start- und Stop-Bit. Zur Fehlererkennung wird zusätzlich ein Paritäts-Bit angefügt, wie man Anhand eines Beispiels in Abbildung 29 betrachten kann. Werden viele Datenwörter hintereinander übertragen wird jedes Datenwort einzeln mit der nötigen Synchronisationsinformationen in Form von Start-Bit und Stop-Bit versehen. [vgl. [5]]

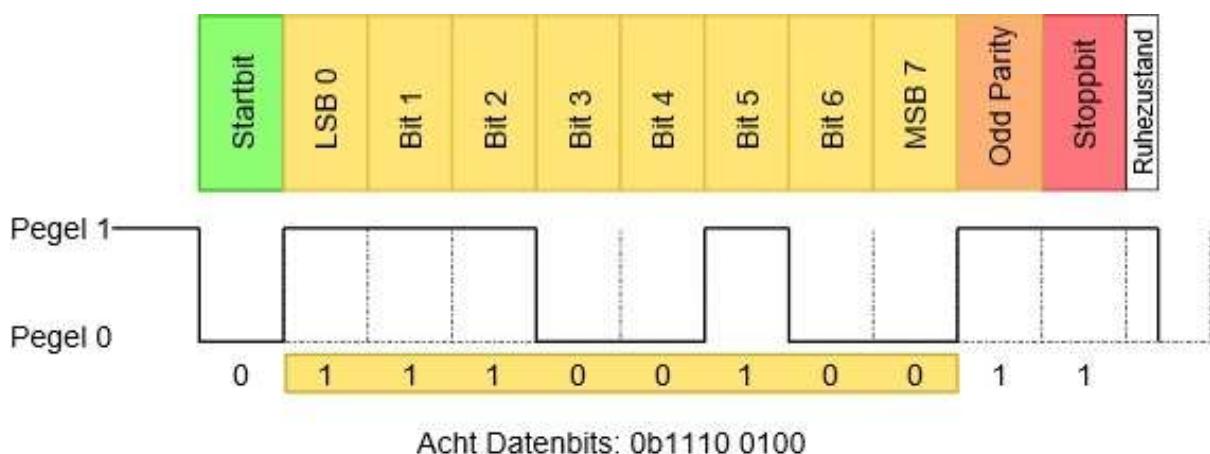


Abbildung 29: asynchrone serielle Übertragung eines Bytes, sowie benötigter Synchronisationsinformationen und ungerader Parität [vgl. [6]]

4.1.3 Vergleich serielle/parallele Interfaces

Serielle und Parallele Interfaces haben jeweils Vor- und Nachteile. Mit parallelen Schnittstellen können hohe Datenraten erreicht werden, jedoch nur auf kurze Distanzen. Die Installation eines parallelen Interface, erfolgt mit dicken Kabeln und ist insgesamt sehr kostspielig.

Serielle Schnittstellen erreichen nicht so hohe Datenraten wie parallele Schnittstellen. Es kann aber über längere Strecken kommuniziert werden. Sie sind einfach zu installieren und simpel erweiterbar. Außerdem sind serielle Interfaces kostengünstig im Vergleich zu parallelen Interfaces

Serielle Interfaces	Parallele Interfaces
Lange Reichweite	Kurze Reichweite
Langsame Übertragung	Schnelle Übertragung
Geringe Kosten	Hohe Kosten
Einfache Installation	Komplexe Installation

Tabelle 2: Übersicht über die Vor- und Nachteile von seriellen und parallelen Interfaces

4.1.4 UART

UART ist eine asynchrone, serielle Punkt zu Punkt Schnittstelle zwischen zwei Geräten. Das Interface benötigt zwei Daten Leitungen (Rx, Tx) und eine Masseverbindung. Wichtig ist, dass, die Rx- und Tx-Leitung, wie in Abbildung 30 zusehen, gekreuzt verdrahtet werden muss.

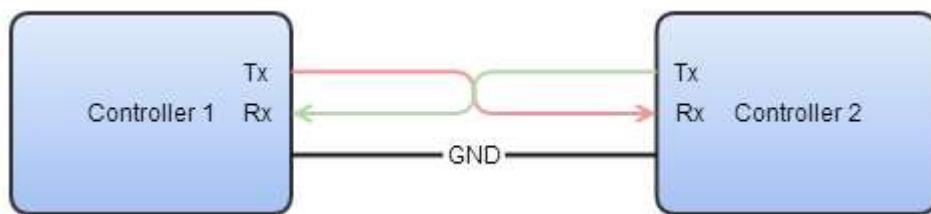


Abbildung 30: UART Interface. RX und TX sind gekreuzt verdrahtet [vgl. [7]]

Da UART eine asynchrone Schnittstelle ist, werden wie in 4.1.2.2 schon erklärt, Start- und Stop- Bits sowie ein Paritätsbit benötigt. Weiters muss bei beiden Teilnehmern die gleiche Baudrate (Übertragungsrate) eingestellt werden. Diese darf maximal um 10% bei den Teilnehmern variieren. Die Logikpegel des UART Interface sind abhängig von den verwendeten Geräten. Die meisten Geräte verwenden TTL-Logikpegel.

Übliche Taktraten
4800 bps
9600 bps
19200 bps
57600 bps
115200 bps

Tabelle 3: Übliche Taktraten des UART-Interface

Bei höheren Baudraten werden oft auch zwei Stop-Bits verwendet, um sicher zu gehen, dass der Empfänger das Stop-Bit richtig abtastet. Wie an dem beispielhaften Datensatz in Abbildung 31 erkennen kann, wird das LSB als erstes verschickt. [vgl. [4]]

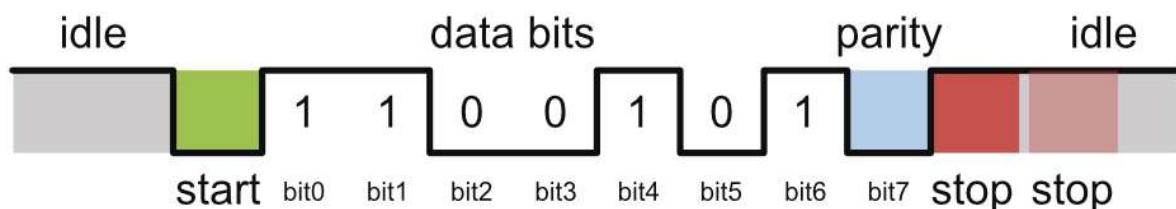


Abbildung 31: Beispielhafter Datensatz einer UART Übertragung [vgl. [4]]

4.1.5 HC-05 Bluetooth-Modul

Für den Datentransfer vom Arduino MEGA zum Smartphone wurde ein schon fertiges Bluetooth-Modul verwendet. Da wir in vorherigen Projekten bereits Erfahrungen mit dem HC-05 Bluetooth gesammelt hatten, wurde dieses Modul auch hier verwendet.

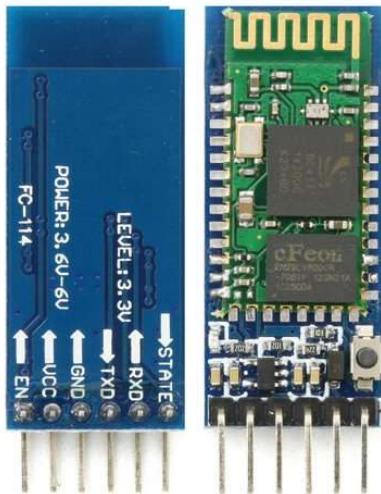


Abbildung 32: HC-05 Bluetooth-Modul

Das Modul wird mit einer Versorgungsspannung von 5V versorgt und ist TTL kompatibel. Weiters gibt es 2 verschiedene Betriebsarten. Einmal den „Data mode“ (default) in dem das Modul über RX und TX mit anderen Geräten kommunizieren kann und zum anderen den „AT Command Mode“. Im Command Mode können Einstellungen wie Baudrate, Name des Moduls und Passwort geändert werden.

Das HC-05 kommuniziert über eine UART Schnittstelle mit dem Arduino MEGA, d.h. die Empfangs- (RXD) und Sendeleitung (TXD) müssen wie in 4.1.4 erklärt, verkreuzt werden.

HC-05	Arduino MEGA
STATE	nC
RXD	10
TXD	49
GND	GND
VCC	5V
EN	nC

Tabelle 4: Pinbelegung des HC-05 im Data Mode

HC-05	Arduino MEGA
STATE	nC
RXD	10
TXD	49
GND	GND
VCC	5V
EN	GND

Tabelle 5: Pinbelegung für AT Command Mode
(EN=GND)

4.1.5.1 Programmierung

Um Daten vom Arduino MEGA an das Smartphone zu senden, muss der Mikrocontroller entsprechend programmiert werden. Die Programmierung erfolgt im Framework Arduino, in welchem man mit C++ arbeitet.

Für die serielle Kommunikation wird die Bibliothek „SoftwareSerial.h“ inkludiert.

```
#include <SoftwareSerial.h>
```

Abbildung 33: Code zum Inkludieren der SoftwareSerial Bibliothek

Als nächstes wird eine Instanz von der Klasse SoftwareSerial erstellt. Die Parameter definieren die RX- und TX-Pins am Arduino MEGA.

```
SoftwareSerial serialBluetooth(10, 49); //Rx and Tx
```

Abbildung 34: Code zum Festlegen von Rx und Tx auf die Pins 10 und 49

Da das HC-05 mit einer Baudrate von 9600 bps konfiguriert wurde, muss die serielle Kommunikation mit derselben Baudrate konfiguriert werden. Hierfür gibt man der Methode begin() die gewünschte Baudrate in Form einer Ganzzahl mit.

```
serialBluetooth.begin(9600); //Initiate Software Serial with 9600 baudrate
```

Abbildung 35: Code zum Einstellen der Baudrate auf 9600 bps

Nun kann mit der Methode print() ein beliebiger Text an das, über Bluetooth verbundenen Smartphone, gesendet werden.

```
serialBluetooth.println("f");
```

Abbildung 36: Code zum Senden des Buchstabens „f“

Für das Empfangen von Daten ist es nötig, zu überprüfen ob überhaupt Daten empfangen werden. Dies wird mit meiner If Verzweigung erledigt, in der man mit der Funktion available() abfragt ob eingehende Bytes verfügbar sind. Falls welche verfügbar sind, werden diese mit der Funktion read() in eine Variable vom Typ character gespeichert.

```
if (serialBluetooth.available())
{
    // read the incoming byte:
    incomingByte = serialBluetooth.read();
}
```

Abbildung 37: Code zum Einlesen von empfangenen Daten

4.1.6 I²C

I²C ist ein weitverbreitetes synchrones, serielles Interface, dass vor allem in der geräteinternen Kommunikation zwischen Schaltungsteilen verwendet wird. Es besteht, wie in Abbildung 30 zusehen, aus einer Datenleitung (SDA) und einer Takteleitung (SCK). Die Signalleitung wird mit Pull-up Widerständen auf High Potential gezogen. Die Teilnehmer, die an der Kommunikation teilnehmen werden in zwei Klassen unterteilt: Master und Slave.

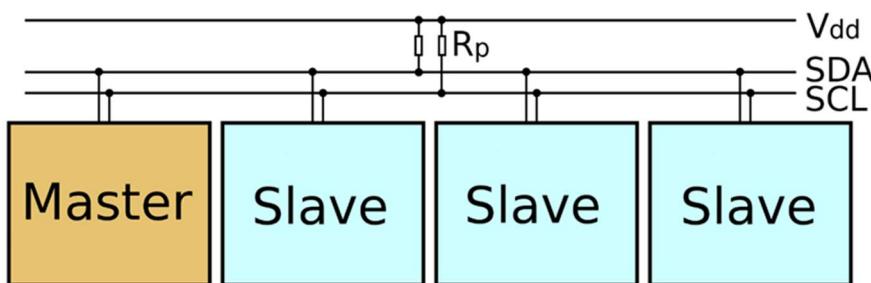


Abbildung 38: Möglicher Aufbau eines I²C Interfaces mit einem Master und drei Slaves [vgl. [8]]

Es kann einen oder mehrere Master geben, und maximal 112 Slaves. Der Master leitet die Kommunikation und fordert den Slave auf, Daten zu senden oder zu empfangen. Alle Slaves können individuell über eine Adresse angesprochen werden. Über einen Broadcast-Kanal können aber auch alle Slaves auf einmal angesprochen werden. Der Master kann z.B. ein Mikrocontroller und der Slave ein Sensor oder wie in diesem Projekt ein RFID-Modul sein. Die Logikpegel hängen von der Versorgungsspannung (Vdd) ab. Low-Pegel (Logische 0) sind bis $0,3 \times V_{dd}$, und High-Pegel (Logische 1) ab $0,7 \times V_{dd}$ definiert.

Die maximale Buslänge lässt sich nicht mit einem bestimmten Wert angeben. Sie ist begrenzt durch die gesamte kapazitive Belastung, die 400pF nicht übersteigen darf. Diese Kapazität hängt von der Leitungslänge, Taktfrequenz und Anzahl der angeschlossenen Geräte ab.

4.1.6.1 Datenübertragung

Im Ruhezustand sind SDA und SCL auf High-Potenzial. Der Zustand von SDA darf sich grundsätzlich nur bei SCL=LOW verändern. Unabhängig von der Richtung der Kommunikation zwischen Master und Slave sind die angelegten Daten bei SCL=HIGH gültig.

Der Master ist von dieser Regel aber zum Teil ausgenommen und darf spezielle Steuerkommandos senden, die darauf basieren SDA, während SCL=HIGH zu ändern.

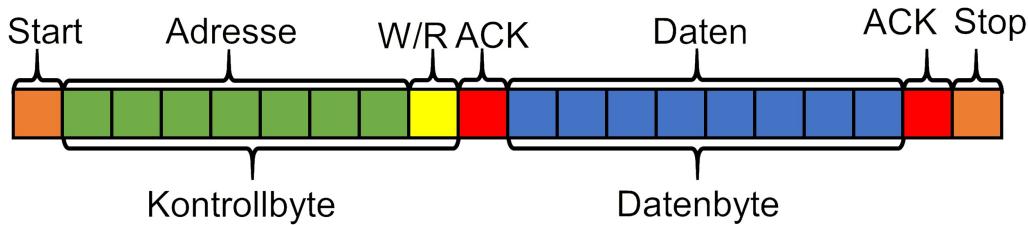


Abbildung 39: Aufbau einer I²C Datenübertragung [vgl. [9]]

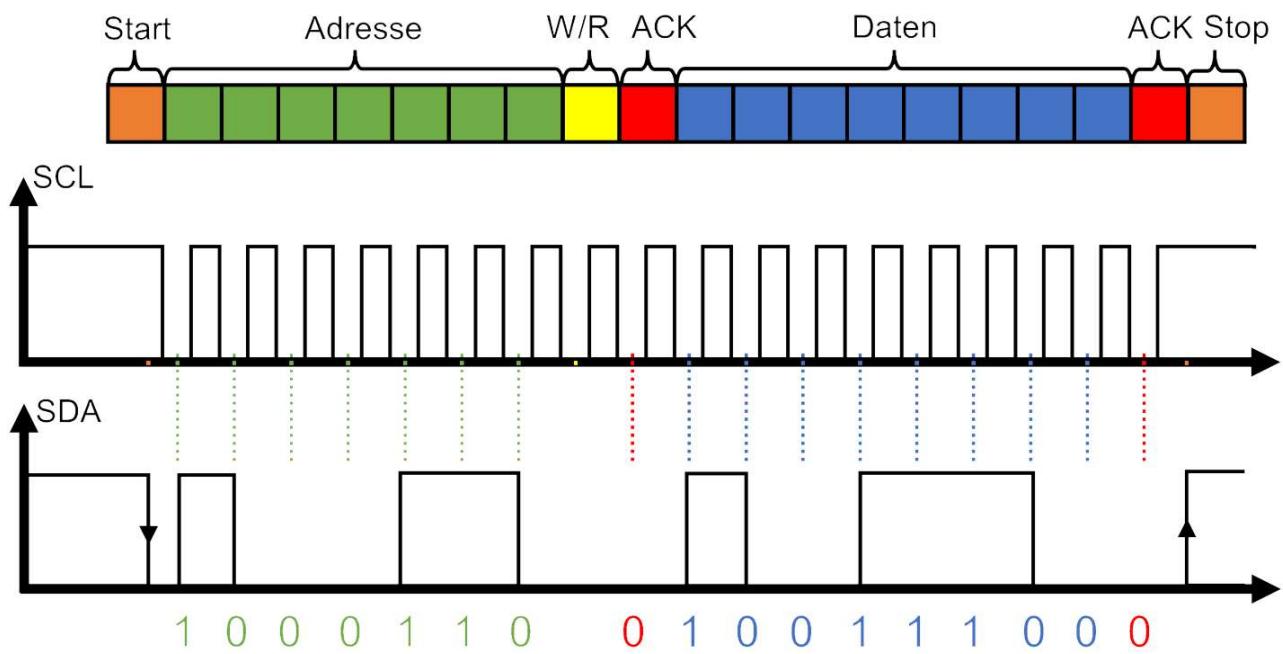
Die Kommunikation startet mit einer Start-Condition. Dabei wird vom Master auf der Datenleitung eine fallende Flanke produziert, während SCL High ist. Danach geht CLK auf 0, und SDA kann sich ändern.

Nun zieht der Sender SDA entweder auf Low (Logische 0 schreiben) oder auf High (Logische 1 schreiben). Der Master zieht SCL wieder auf HIGH, und die Daten werden vom Empfänger interpretiert. Auf diese Weise wird ein Kontrollbyte (siehe 4.1.6.2) übertragen, worauf ein ACK-Bit folgt. Dieses ACK-Bit wird vom Empfänger auf 0 gesetzt, um den Erhalt der Daten zu bestätigen. Falls der Slave beschädigt, beschäftigt oder das empfangene Byte nicht lesbar ist, wird das ACK auf 1 gesetzt und der Sender weiß somit, dass seine Nachricht nicht angekommen ist.

Die beste Reaktion auf ein fehlendes ACK ist es die Übertragung durch die Stop-Condition zu beenden und anschließend erneut zu versuchen.

Eine Stop-Condition wird vom Master durch eine steigende Flanke auf SDA, während CLK auf LOW ist, hervorgerufen. Sie ist das Ende einer I²C Übertragung. Wenn das Kontrollbyte angekommen ist, folgt darauf das Datenbyte, in dem die Information enthalten ist. Es wird genau wie die Adresse mit dem MSB als erstes versendet. Mit dem ACK wird der Erhalt der Information wieder bestätigt und anschließend beendet die Stop Condition die Übertragung.

[vgl. [9]]

Abbildung 40: Timing Diagramm einer I²C Übertragung [vgl. [9]]

Modus	Maximale Übertragungsrate
Standard Mode	0,1 Mbit/s
Fast Mode	0,4 Mbit/s
Fast Mode Plus	1,0 Mbit/s
High Speed Mode	3,4 Mbit/s

Tabelle 6: Die verschiedenen Modi mit ihren Übertragungsraten

4.1.6.2 Adressbyte

Das Kontrollbyte wird an alle Slaves versendet und teilt mit, welcher Slave angesprochen wird und was er zu machen hat. Die ersten 7 Bit bestehen aus der Slave-Adresse. Diese Adresse kann aus einem festen und einem programmierbaren Teil bestehen. Das 8. Bit signalisiert dem Slave ob er Senden oder Empfangen muss. Wenn R/W Low ist, wird geschrieben, bei High wird gelesen. [vgl. [10]]

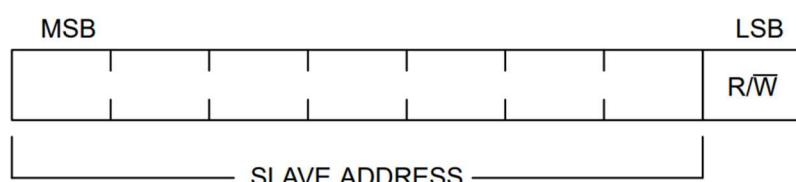


Abbildung 41: Adressbyte [vgl. [10]]

4.1.6.3 Multi-Master Betrieb

Für den Fall, dass mehrere Master in die I²C Kommunikation eingebunden sind, braucht es eine entsprechende Regelung, um Chaos zu vermeiden. Wenn ein Master eine 1 sendet, aber eine 0 detektiert, so weiß er, dass ein anderer Master auch senden will, und muss seine Aktivität einstellen. Somit ergibt es sich, dass der Master, der die niedrige Slave-Adresse anspricht, gewinnt. Wollen die Master den gleichen Slave ansprechen, so gewinnt der Master, der das niedrigere Daten-Byte sendet. Dieses Vorgehen wird Arbitrationsprozess genannt. [vgl. [10]]

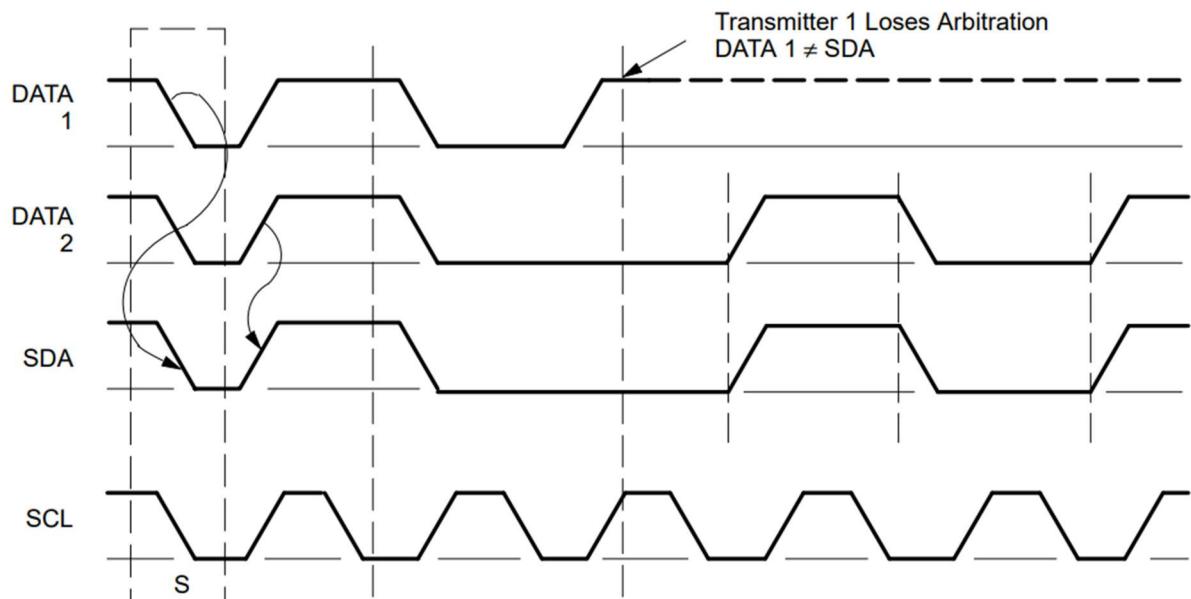


Abbildung 42: Arbitrationsprozess mit zwei Master [vgl. [10]]

4.1.7 SPI

SPI ist ein synchrones serielles Interface. Im Gegensatz zu I²C ist nur ein Master möglich und die Anzahl der Slaves wird nur durch die Anschlüsse am Master beschränkt. Die Schnittstelle ist voll duplexfähig und kann bis zu 20 Mbps übertragen.

SPI besitzt 4 Leitungen:

- SCLK: Taktsignal
- MOSI: Datenleitung Master→Slave
- MISO: Datenleitung Slave→Master
- SS: Slave select

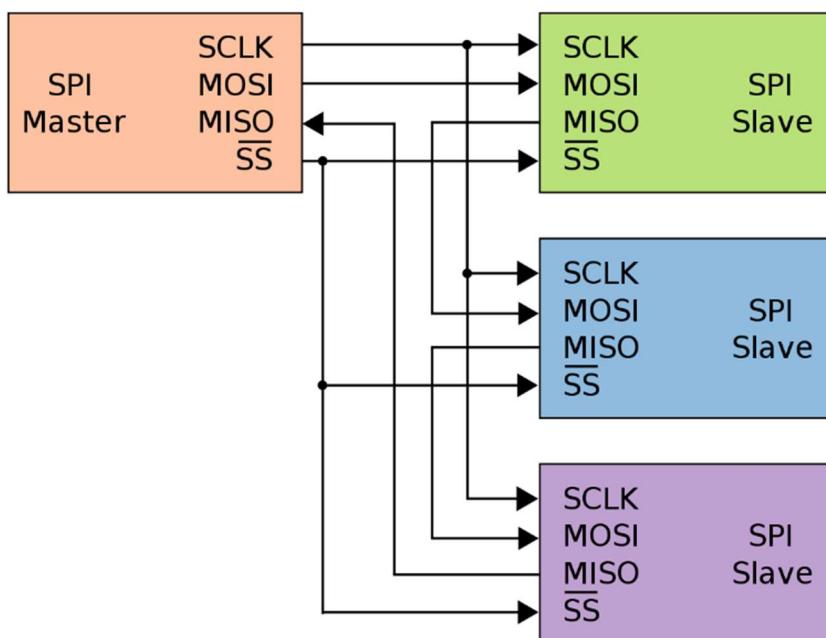


Abbildung 43: SPI Interface mit 3 Slaves [11]

4.1.7.1 Slave select (SS)

Diese Leitung zieht der Master auf LOW um den Slave, der auf dieser Leitung hängt, für eine Kommunikation zu aktivieren.

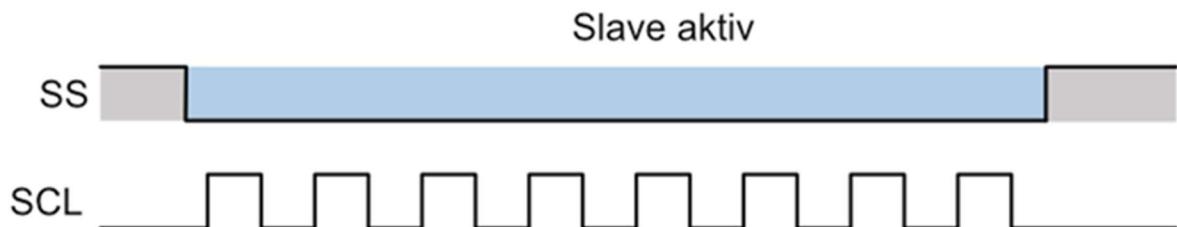


Abbildung 44: Slave wird mittels Slave select aktiviert [vgl. [11]]

4.1.7.2 Operations Modi

Der Master und Slave müssen sich vor der Übertragung über die Synchronisation einigen. Dies erfolgt durch die Polarität (CPOL) und die Phase (CPHA) des Taktsignals. Dabei ergeben sich die in Tabelle 7 aufgelisteten Modi.

Mode Nr.	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Tabelle 7: 4 verschiedene Modes zur Einstellung der Synchronisation

CPOL bestimmt welchen Zustand das Clock-Signal im „idle“ Modus (Leerlauf) hat. Bei CPOL=0 ist der „idle“ Zustand LOW und bei CPOL=1 HIGH.

CPHA legt fest, ob die Daten bei der ersten Flanke (CPHA=0) oder bei der zweiten Flanke (CPHA=1) des Taktsignals gelesen werden.

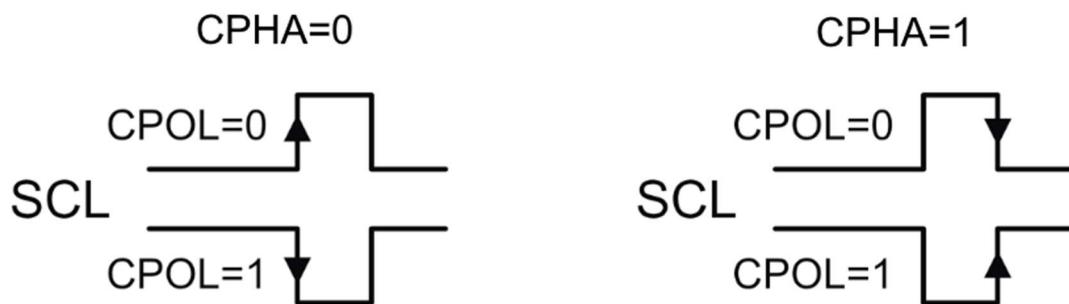


Abbildung 45: 4 verschiedene Modis zur Synchronisation [vgl. [12]]

4.1.7.3 Senden von Daten

Vom Master wird ein Slave über die Slave select Leitung ausgewählt. Danach werden die Daten entweder mit MSB oder LSB zuerst übertragen. Ob MSB oder LSB zuerst gesendet wird ist geräteabhängig und konfigurierbar. Wichtig ist, dass Daten nie an den Flanken des Datensignals gelesen werden, sondern immer in der Mitte. [vgl. [12]]

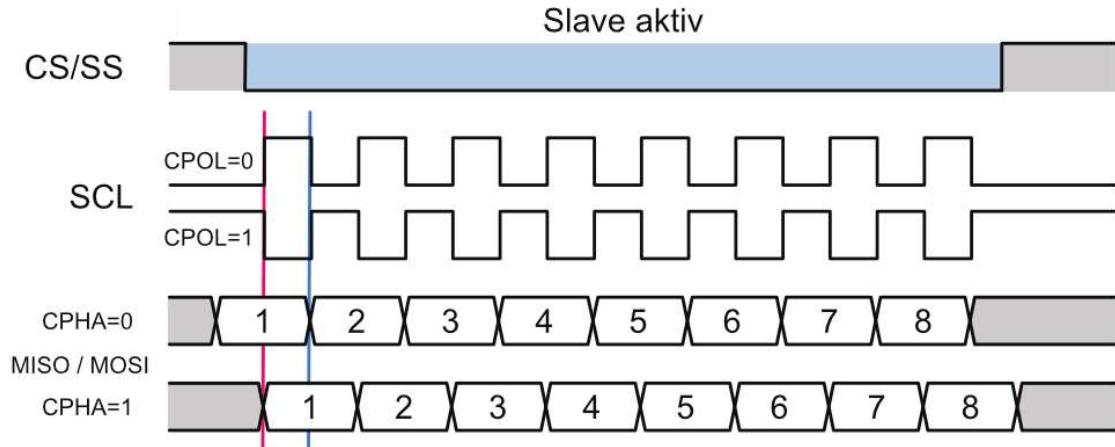


Abbildung 46: Timing Diagramm einer SPI Übertragung [vgl. [12]]

4.1.8 RC522 RFID Modul

Zur RFID-Authentifizierung wurde ein fertiges RC522 RFID Modul verwendet. Das Vorhandensein von application notes und die gute Dokumentation, waren neben dem günstigen Preis die ausschlaggebenden Faktoren für die Wahl des Moduls. Das Modul basiert auf dem MFRC522-Controller (siehe Datenblatt [13]). Zusätzlich zum Modul wird auch eine RFID- Karte und Schlüsselanhänger mitgeliefert.



Abbildung 47: RC522 RFID Modul



Abbildung 48: RFID Schlüsselanhänger und Karte

Die Versorgungsspannung des RC522 beträgt 3,3V. Die Kommunikationspins können aber, laut Spezifikation, auch mit 5V Spannungspegeln operieren, weshalb keine zusätzlichen Pegelwandler nötig sind. Für die Kommunikation zwischen dem RFID-Modul und Arduino MEGA kann zwischen I²C, UART, und SPI Interface gewählt

werden. Da der Arduino MEGA über viele Pins verfügt, kann auch ein Interface, das mehrere Pins benötigt, ohne Nachteile gewählt werden. Da wir nur einen Master (Arduino MEGA) haben, war kein Interface, das mehrere Master erlaubt nötig. Schlussendlich wurde das SPI Interface wegen der schnellen Übertragungsrate und den zahlreichen vorhandenen application notes gewählt.

RC522 Modul	Arduino MEGA
SS	53
SCK	52
MOSI	51
MISO	50
IRQ	nC
GND	GND
RST	46
VCC	3,3V

Tabelle 8: Pinbelegung des RC522 mit SPI-Interface

4.1.8.1 Funktionsweise

Das Modul erzeugt ein elektromagnetisches Feld von 13,56 MHz um mit RFID-Chips (ISO 14443A Standard) zu interagieren. Nähert man die Chips (1kByte Speicherplatz) mindestens 5cm an das Modul, können diese je nach Belieben beschrieben und ausgelesen werden kann. In unserem Fall wird das Modul vor allem dazu verwendet, um die UID eines RFID-Tags auszulesen.

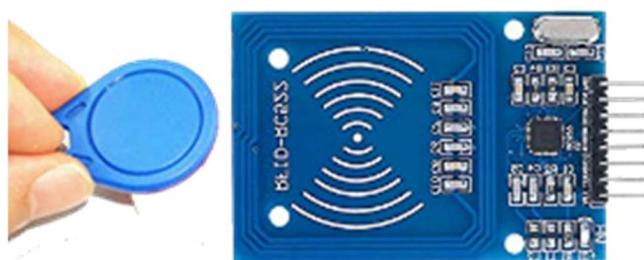


Abbildung 49: RFID-Tag kann ab einem Abstand von 5cm zum Modul beschrieben/gelesen wird

4.1.8.2 Programmierung

Die Programmierung der Kommunikation zwischen Arduino MEGA und RC522 wird ebenfalls mit dem Framework Arduino programmiert.

Speziell für den im RC522 verbauten Chip (MFRC522) wird eine Bibliothek (siehe [14]) zum Beschreiben und Auslesen eines RFID Tags bereitgestellt. Diese Bibliothek muss genau so wie die Standardbibliothek für das SPI Interface inkludiert werden.

```
#include <SPI.h>
#include <MFRC522.h>
```

Abbildung 50: Code zum inkludieren der SPI und MFRC522 Bibliothek

Nun wird eine Instanz der Klasse „MFRC522“ erstellt. Die beiden Parameter definieren SS-Pin und RST-Pin des SPI Interfaces.

```
#define SS_PIN 53
#define RST_PIN 46

MFRC522 mfrc522(SS_PIN, RST_PIN);
```

Abbildung 51: Code zum Definieren von RST und SS Pin

Jetzt wird das SPI Interface und der MFRC522 Chip initialisiert.

```
SPI.begin();           // Initiate SPI bus
mfrc522.PCD_Init(); //Initiate
```

Abbildung 52: Code zum Initialisieren von SPI Interface und MFRC522 Chip

Nun kann mit der Authentifizierung gestartet werden. Um zu überprüfen ob überhaupt ein RFID-Tag in der Nähe ist, wird die Funktion „PICC_IsNewCardPresent()“ der Klasse MFRC522 ausgeführt. Ihr Rückgabeparameter ist entweder „True“ (ein Chip ist verfügbar) oder „False“ (kein Chip ist verfügbar).

```
if ( ! mfrc522.PICC_IsNewCardPresent())
{
    return;
}
```

Abbildung 53: Code zum Überprüfen, ob ein RFID-Tag verfügbar ist

Wenn ein Chip verfügbar ist, wird mit der Methode „PICC_ReadCardSerial()“ die UID ausgelesen. Diese wird in die Klassenvariable „uid“ gespeichert.

```
if ( ! mfrc522.PICC_ReadCardSerial())
{
    return;
}
```

Abbildung 54: Code zum Auslesen der UID

Die ausgelesene UID muss noch in eine bestimmte Form gebracht werden, um dann schlussendlich mit der geforderten UID verglichen zu werden. Das Konvertieren in die gewünschte Form erledigt die for-Schleife in Abbildung 55. Wenn die ausgelesene UID mit der Geforderten übereinstimmt wird die Variable „verified“ auf „True“ gesetzt. Falls sie nicht übereinstimmen, auf „False“.

```
String content= "";
for (byte i = 0; i < mfrc522.uid.size; i++)
{
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
}
content.toUpperCase();
if (content.substring(1) == cardUID)//check if CardUID is correct
{
    verified = true;
}
```

Abbildung 55: Code, der die UID in die gewünschte Form konvertiert und anschließend mit der geforderten UID vergleicht

4.1.9 1-Wire

1-Wire ist eine asynchrone serielle Schnittstelle, welche lediglich eine Datenleitung und eine Masse-Verbindung benötigt. Ein Pull-up-Widerstand zieht die Datenleitung auf High-Potential. Je nach Anzahl der Slaves und Länge der Leitung, variiert die Größe des Widerstands. Die Übertragung erfolgt so wie bei SPI nach dem One-Master/Multi Slave Prinzip. Jeder Slave hat dabei eine fest einprogrammierte 64Bit große Slave-Adresse. Da nur eine Datenleitung vorhanden ist, ist die Kommunikation nur im Halbduplex Mode möglich. [vgl. [15]]

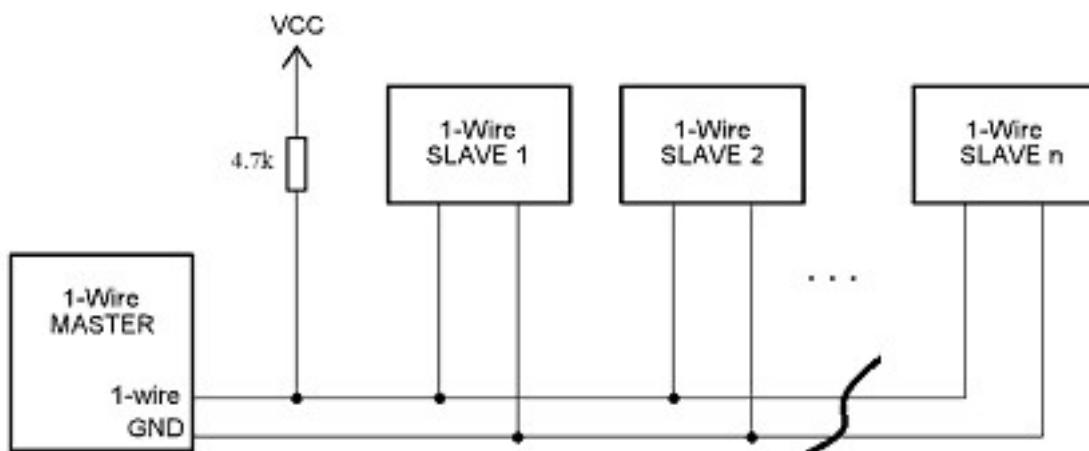


Abbildung 56: 1-Wire Interface mit 4,7kΩ Pull-up-Widerstand [vgl. [16]]

4.1.9.1 Spannungsversorgung der Slaves

Die Slaves werden grundsätzlich über einen internen Kondensator versorgt. Dieser wird über die Datenleitung geladen (siehe Abbildung 57). Diese Methode funktioniert für kleine Netzwerke und schränkt die Übertragungsgeschwindigkeit, Buslänge und Anzahl der Slaves massiv ein. Aus diesem Grund ist es möglich eine externe Versorgung zu verwenden. Die meisten 1-Wire Geräte lassen eine Betriebsspannung von 2,8V bis 5,5V zu. [vgl. [17]]

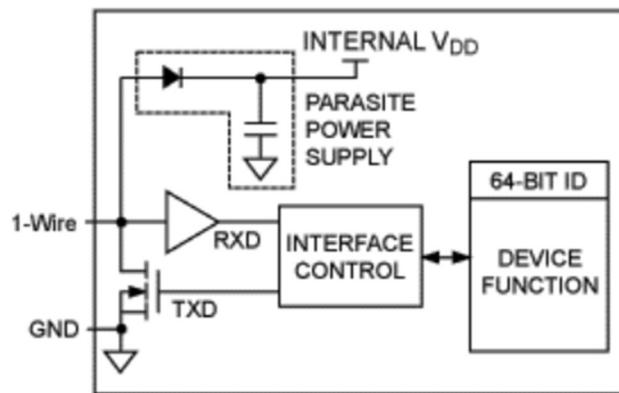


Abbildung 57: Innerer Aufbau eines 1-Wire Slaves mit interner Kapazität [vgl. [18]]

4.1.9.2 Datenübertragung

Da kein Taktsignal vorhanden ist, basiert die Kommunikation zwischen Master und Slave auf „Time Slots“. „Time Slots“ sind Intervalle, in denen ein Bit übertragen wird. Jeder Datentransfer beginnt mit einer vom Master erzeugten fallenden Flanke. Sie ist das Synchronisation Signal, sowohl für den Master als auch für den Slave. Im Standardmodus sind Datenraten von bis zu 16,3 kBit/s möglich.

Zum Schreiben einer logischen 1 zieht der Master die Datenleitung für max. 15µs auf LOW, während der Slave nach 30µs abtastet. Eine 0 wird geschrieben in dem der Master die Datenleitung für min. 60µs auf LOW Potential zieht. Der Slave tastet ebenfalls wieder nach 30µs ab. Auf diese Weise werden nun 8 Bits hintereinander übertragen und ergeben so ein Byte.

Zum Lesen generiert der Master einen min. 1µs langen Impuls, auf den der Slave dann entsprechend reagiert. Wenn der Slave eine 1 Senden will, belässt er die Datenleitung im Ruhezustand → Datenleitung auf 1. Um eine 0 zu senden, zieht der Slave die Datenleitung auf LOW. Um den Bus zurückzusetzen, legt der Master einen 480µs langen LOW-Pegel an.

Im Overdrive Modus werden alle Zeiten um ca. ein Zehntel verkürzt. So sind Datenraten von bis zu 142 KBit/s möglich. [vgl. [19]]

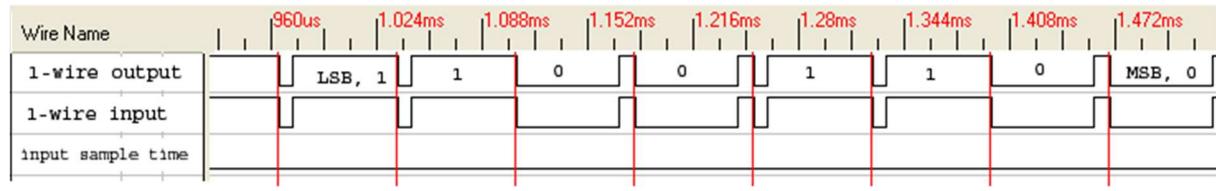


Abbildung 58: Timing Diagramm um 0x33 zusenden [vgl. [17]]



Abbildung 59: Timingdiagramm um 0x01 zusenden

4.1.10 LED-Matrix

Zur Darstellung des L17 Logos wurde eine bereits fertige LED-Matrix gewählt. Es sollte möglich sein, Buchstaben und Ziffern gut sichtbar in Farbe darzustellen. Dafür wurde eine schon fertige 16cm x 16cm große RGB LED-Matrix der Firma BTF-LIGHTING gewählt, bei der jede LED einzeln über 1-Wire ansteuerbar ist.

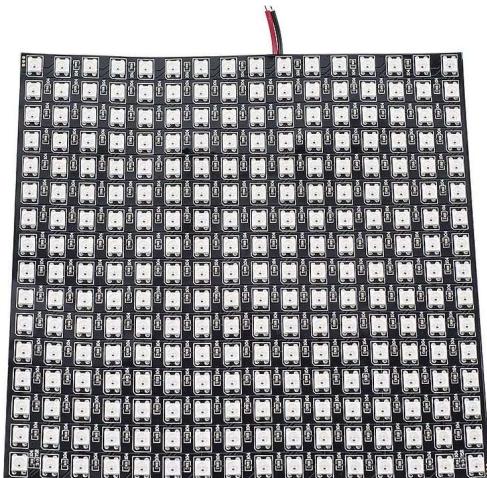


Abbildung 60: Verwendete LED-Matrix mit 256 LEDs

4.1.10.1 Funktionsweise

Die LED-Matrix wurde mit einem 256 LEDs langen LED-Streifen realisiert der serpentinengartig verlegt wurde. In Abbildung 61 ist ein solcher Verlauf am Beispiel einer kleineren Matrix zu sehen. Es wurden Ws2812b LEDs verwendet, die sich von anderen LEDs durch ihren integrierten Schaltkreis unterscheiden, und somit einzeln ansteuerbar sind. Die LEDs verfügen laut Datenblatt über 256 Helligkeitsstufen und insgesamt über 16777216 mögliche Farben. Jede Ws2812b LED besteht also noch aus drei einfachen LEDs (Rot, Grün, Blau).

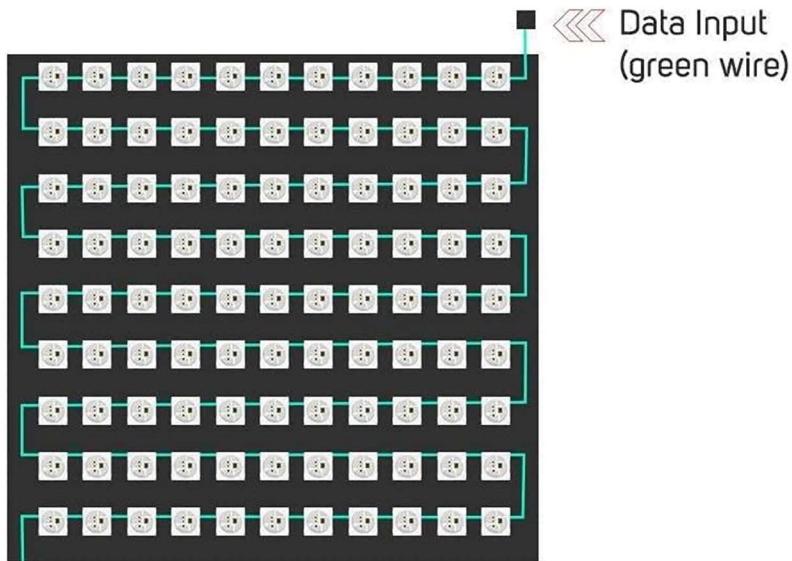


Abbildung 61: Serpentinenartiger Verlauf des LED-Streifens

Der Arduino MEGA kommuniziert mit der LED-Matrix über das 1-Wire Interface und versorgt diese mit 5V. Die benötigten Pull-up-Widerstände und Kondensatoren sind schon auf der LED-Matrix angebracht.

LED-Matrix	Arduino MEGA
DIN	48
5V	5V
GND	GND

Abbildung 62: Pinbelegung der LED-Matrix

Die Ansteuerung der LED-Matrix funktioniert mittels 1-Wire Protokoll folgendermaßen:

- Es werden insgesamt 768 Bytes (256 LEDs x3) hintereinander gesendet
 - 1. Byte → 1. RGB Farbe Rot
 - 2. Byte → 1. RGB Farbe Grün
 - 3. Byte → 1. RGB Farbe Blau
 - 4. Byte → 2. RGB Farbe Rot
 - usw...
- Je größer der im Byte enthaltene Wert desto heller die LED ($2^8=256$ Abstufungen)

4.1.10.2 Programmierung

Die Programmierung erfolgt mit dem Arduino Framework und der Bibliothek „Adafruit_NeoPixel“. Diese Bibliothek muss anfangs inkludiert werden.

```
#include <Adafruit_NeoPixel.h>
```

Abbildung 63: Code zum inkludieren der „Adafruit_NeoPixel“ Bibliothek

Nun wird eine Instanz der Klasse „Adafruit_NeoPixel“ erstellt. Die Parameter definieren die Anzahl der LEDs, den verwendeten Pin am Arduino MEGA und den LED Typ.

```
#define LED_PIN 48  
Adafruit_NeoPixel pixels(256, LED_PIN, NEO_GRB + NEO_KHZ800);
```

Abbildung 64: Code zum Erstellen der Instanz „pixels“

Jetzt können alle LEDs mit einer beliebigen Farbe programmiert werden.

```
pixels.setPixelColor(0, pixels.Color(192, 192, 192));  
pixels.show();
```

Abbildung 65: Code zum Verändern der Farbe einer bestimmten LED

Die Methode „setPixelColor“ besitzt 2 Parameter. Der erste ist der Index der zu ansteuernden RGB-LED (0-255) und der zweite ist für die Farbeinstellung zuständig.

Die Methode „Color“ besitzt drei Parameter (<int> Rot, <int> Grün, <int> Blau) die jeweils einen Wert zwischen 0x00 und 0xFF annehmen können (0x01-FF = ON, 0x00 = OFF).

Um herauszufinden welche Werte die Parameter der Funktion „Color“ für eine bestimmte Farbe haben müssen, muss man den hex/dez code (siehe Abbildung 66) bzw. die Werte für Rot, Grün und Blau ermitteln. Das kann man unter anderem mit dem „Color Picker“ Werkzeug in Paint.net machen. Sobald man das Werkzeug ausgewählt hat, kann man auf die gewünschte Farbe klicken und bekommt im „Colors Fenster“ die entsprechenden Werte geliefert.

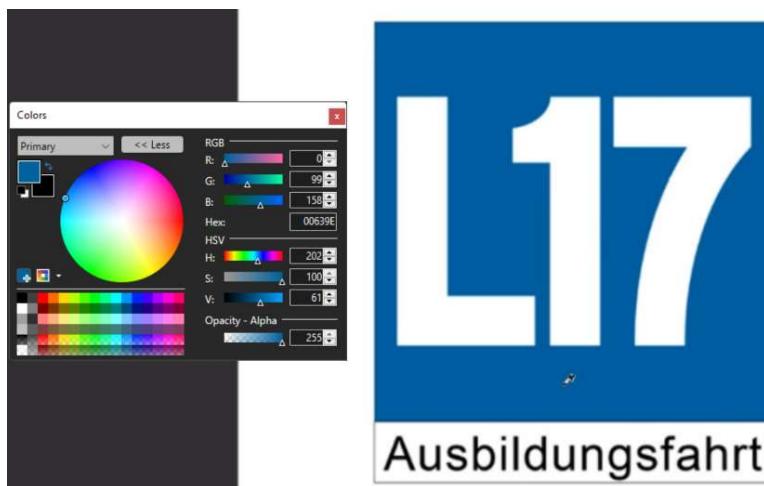


Abbildung 66: Bestimmen der benötigten Werte mit dem Color Picker Werkzeug in Paint.net

Um das Blau des L17 Schildes auf der LED-Matrix abzubilden, müssen die mit dem „Color Picker“ ermittelten Werte, der „Color“ Funktion mitgegeben werden

```
pixels.setPixelColor(0, pixels.Color(0, 99, 158));
pixels.show();
```

Abbildung 67: Code zum Einfärben der ersten LED mit dem L17-Blau

4.1.10.3 Leistungsverbrauch

Der Leistungsverbrauch wurde mit dem abgebildeten L17 Logo gemessen. Es wurden mit einem Amperemeter und einem Voltmeter, wie in Abbildung 68 zusehen, Strom und Spannung gemessen. Anschließend wurde die Leistung mit $P = U * I$ berechnet.

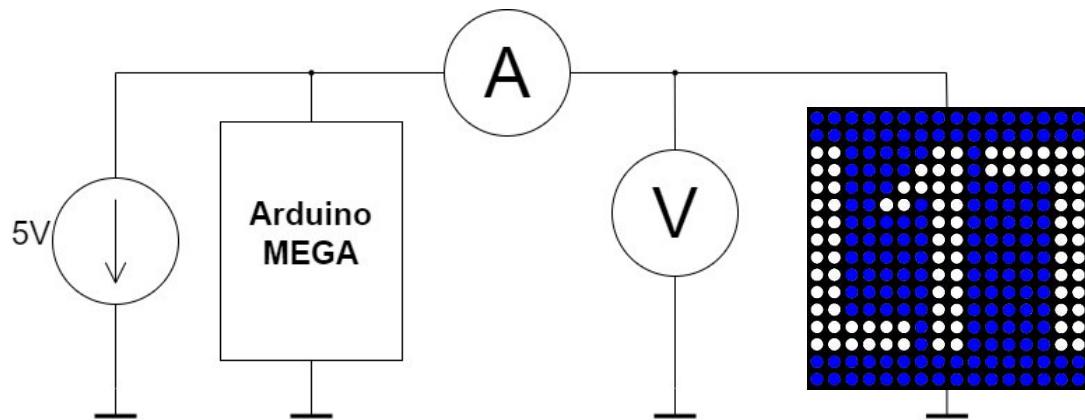


Abbildung 68: Messschaltung

U/V	I/A	P/W
4,94	1,2	5,928

Abbildung 69: Leistungsmessung der LED-Matrix

4.2 App-Entwicklung mit Flutter

4.2.1 Was ist Flutter?

Flutter ist ein Framework zur Entwicklung von Smartphone-Apps, welches von Google im Jahr 2018 als Open-Source-UI-Entwicklungs-Kit veröffentlicht wurde. Im Gegensatz zu anderen App-Entwicklungs-Frameworks, welche meist nur eine Plattform unterstützen, laufen Flutter-Applikationen ohne größere Anpassungen auf einer Vielzahl an Plattformen, wie zum Beispiel Android, iOS, Windows und macOS. Die Code-Basis wird in der Programmiersprache Dart geschrieben. [vgl. [20]]

4.2.2 Vor-/Nachteile von Flutter

Vorteile	Nachteile
<ul style="list-style-type: none">• kompatibel mit allen wichtigen Plattformen• leicht zu erlernen• umfangreiche UI-Bibliotheken• Hot Reload vereinfacht das Testen	<ul style="list-style-type: none">• wird durch die Einbindungen von Widgets schnell unübersichtlich• wenig verbreitet

Abbildung 70: Vor- und Nachteile von Flutter

[vgl. [20]]

Im Gegensatz zu anderen Cross-Platform-Frameworks (z.B. React Native, Xamarin) gibt es sehr viele Bibliotheken, welche einfach eingebunden werden können. Zusätzlich muss bei den meisten anderen Frameworks, bei einer Änderung, die gesamte App neuinstalliert werden.

4.2.3 Installation

Für die Funktionalität der Flutter SDK werden zwei Programme vorausgesetzt:

- PowerShell 5.0 (Standard mit Windows 10)
- git (V: 2.33.0.windows.2)

Sobald beides installiert ist, kann die Flutter SDK installiert werden, welche auch in den Umgebungsvariablen definiert werden muss. Mit folgendem Befehl kann nach der Installation der Status von Flutter in der Konsole abgerufen werden:

- flutter doctor

Damit Flutter verwendet werden kann, müssen folgende Programme noch installiert werden:

- Android SDK
- Android Studio (mit Flutter Plugin)
- Chrome (für Web-Entwicklung)
- Visual Studio Code (mit Flutter Extension)

Wenn man zum Testen der App einen Emulator verwenden möchte, muss der Android-Emulator von Android Studio installiert werden. [vgl. [21]]

4.2.4 Ein neues Projekt erstellen

Mithilfe der Command Palette in VS Code kann „Flutter: New Projekt“ und danach „Application“ ausgewählt werden, womit eine Ordnerstruktur erzeugt wird, welche die Standardkonfiguration und die Standard-App beinhaltet. [vgl. [22]]

4.2.5 App testen

Man kann eine Flutter-App entweder an einem angeschlossenem Android-Gerät testen oder mithilfe des Android Studio Emulators. Dies kann am rechten Ende der Status Bar ausgewählt werden. Danach kann man unter dem Tab „RUN AND DEBUG“ seine App auf das gewählte Gerät installieren. [vgl. [22]]

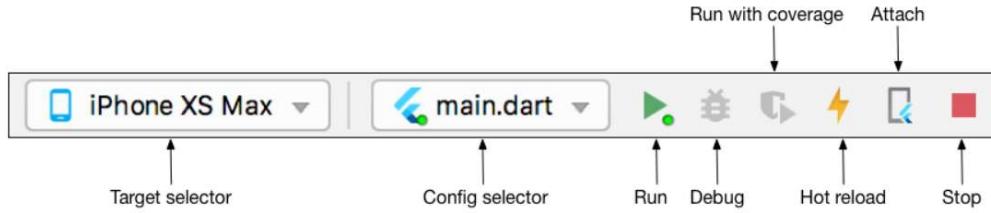


Abbildung 71: App testen [vgl. [22]]

4.2.6 Packages und Libraries

4.2.6.1 Finden von Packages

pub.dev ist der offizielle Package- und Library Manager für Dart und Flutter-Apps. Hier kann man, von der Community erstellte, Bibliotheken suchen und sie später in seine App einbinden. Man findet nahezu für alle Anwendungsbereiche, Bibliotheken welche laufend aktualisiert werden. [vgl. [23]]

4.2.6.2 Einbinden eines Packages

Um vorgefertigte Widgets oder Funktionen verwenden zu können, muss das Package, welches man auf pub.dev gefunden hat, in die pubspec.yaml Datei unter „dependencies:“ eingetragen werden:

```

dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2
  flutter_bluetooth_serial: ^0.4.0
  path_provider: ^2.0.3
  location: ^4.3.0
  http: ^0.13.4
  flutter_launcher_icons: ^0.9.2
  flutter_native_splash: ^1.3.2
  geocoding: ^2.0.2
  weather: ^2.0.1
  step_progress_indicator: ^1.0.2
  flutter_datetime_picker:
    git:
      url: https://github.com/Realank/flutter_datetime_picker.git
      ref: master
  google_nav_bar: ^5.0.5
  line_icons: ^2.0.1
  shared_preferences: ^2.0.13
  syncfusion_flutter_calendar: ^19.4.52
  provider: ^6.0.2

```

Abbildung 72: pubspec.yaml dependencies

Dies kann durch Eintragen des Namens und der Versionsnummer wie in Abbildung 72 erfolgen oder durch folgenden Befehl in der Kommandozeile, welche den Eintrag autogeneriert:

- flutter pub add „Name“

Nach beiden Optionen muss der folgende Befehl zum Herunterladen und Installieren der angegebenen Packages ausgeführt werden:

- flutter pub get

Nach abgeschlossenem Download kann die Bibliothek wie folgt, in der gewünschten Datei importiert werden:

```
import 'package:"Name"/"Name".dart';
```

Abbildung 73: Library-Einbindung

[vgl. [23]]

4.2.6.3 Entfernen von Packages

Um ein eingebundenes Package wieder zu entfernen, ist es möglich den Eintrag aus den pubspec.yaml-dependencies zu löschen oder folgenden Befehl in der Konsole auszuführen:

- flutter pub remove „Name“

Danach muss der Befehl zum Herunterladen ausgeführt werden:

- flutter pub get

[vgl. [23]]

4.2.7 Widgets

In Flutter wird alles, was in der App sichtbar ist und die dazugehörigen Funktionen als Widget bezeichnet. Ein Beispiel dafür ist ein Button und die Funktion, welche ausgeführt wird, wenn ein Klick erfolgt. Widgets können wiederum aus mehreren anderen Widgets bestehen. So bestehen alle App-Fenster aus einer Vielzahl an verschachtelten Widgets. Allgemein werden standardmäßig jene Widgets angezeigt, welche in der build-Methode der MyHomePageState-Klasse zurückgegeben werden. [vgl. [24]]

4.2.7.1 Text-Widget

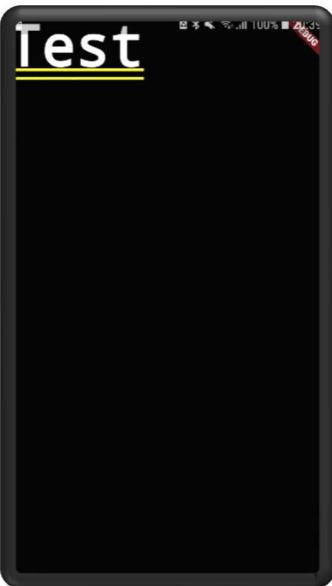


Abbildung 75: Text-Widget-Anzeige

Das Text Widget zeigt den mitgegebenen Text an, durch optionale Parameter können sämtliche Anpassungen vorgenommen werden. Der wichtigste Parameter ist „style:“, wo das Erscheinen (z.B. Farbe, Größe) definiert wird. [vgl. [24]]

```
return const Text("Test",
    style: TextStyle(color: Colors.white, fontSize: 64));
```

Abbildung 74: Code zur Anzeige von Text

4.2.7.2 Container-Widget



Abbildung 76: Container-Widget-Anzeige

Mit dem Container Widget können Bereiche des Bildschirms definiert werden, indem man „width:“ und „height:“ festlegt. Mit „color:“ kann man die Farbe des Bereichs festlegen und mit „child:“ legt man das nächste untergeordnete Widget fest.

[vgl. [24]]

```
return Container(
    color: Colors.blue,
    child: const Text("Test",
        style: TextStyle(color: Colors.white, fontSize: 64)),
);
```

Abbildung 77: Code zur Anzeige eines Containers

4.2.7.3 Center-Widget



Mit dem Center Widget wird das untergeordnete Widget („child“-Attribut), im Bereich des Nächstübergeordneten horizontal und vertikal zentriert. [vgl. [24]]

```
return const Center(
    child: Text("Test", style: TextStyle(
        color: Colors.white, fontSize: 64)),
);
```

Abbildung 78: Code zur Zentrierung von Widgets

Abbildung 79: Anzeige Zentrieren von Widgets

4.2.7.4 ElevatedButton-Widget



Der ElevatedButton erstreckt sich über den gesamten Bereich seines übergeordneten Widgets, er hat keine eigenen Attribute zur Anpassung der Größe, weshalb er meist in Kombination mit einem SizedBox-Widget verwendet wird. Meist ordnet man dem Button ein Text-Widget unter. Jeder Button muss über eine Funktion verfügen, welche aufgerufen wird, wenn er gedrückt wird („onPressed“-Attribut). [vgl. [24]]

Abbildung 80: Anzeige eines Buttons

```
return ElevatedButton(
    onPressed: () {
        print("Test");
    },
    child: const Text("Test",
        style: TextStyle(color: Colors.white, fontSize: 64)));
};
```

Abbildung 81: Code zur Verwendung von Buttons

4.2.7.5 Column/Row-Widget

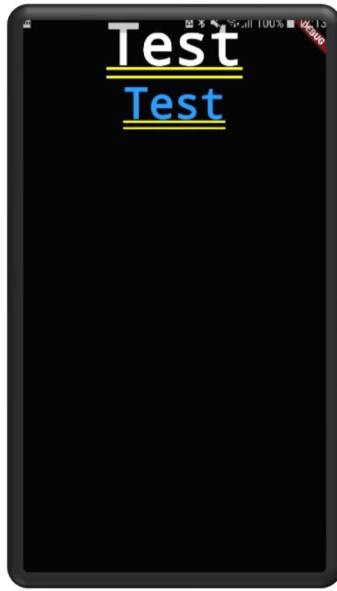


Abbildung 83: Anzeige von Widaets untereinander



Abbildung 82: Anzeige von Widaets nebeneinander

Durch das Column-Widget ist es möglich, Widgets vertikal aneinander zu reihen. Es wird standartmäßig horizontal zentriert. Um Widgets untereinander darzustellen, muss dem Attribut „children:“ eine Liste von Widgets mitgegeben werden, welche dann in chronologischer Reihenfolge angezeigt werden. [vgl. [24]]

```
return Column(
  children: const [
    Text("Test", style: TextStyle(color: Colors.white, fontSize: 64)),
    Text("Test", style: TextStyle(color: Colors.blue, fontSize: 48)),
  ],
);
```

Abbildung 84: Code zur vertikalen Anordnung von Widgets

Das Row-Widget hat die gleiche Funktion, wie das Column-Widget, nur dass es zum horizontalen Ausrichten ist und es standardmäßig vertikal zentriert ist. [vgl. [24]]

```
return Row(
  children: const [
    Text("Test", style: TextStyle(color: Colors.white, fontSize: 64)),
    Text("Test", style: TextStyle(color: Colors.blue, fontSize: 48)),
  ],
);
```

Abbildung 85: Code zur horizontalen Anordnung von Widgets

4.2.8 App-Icon

Standardmäßig wird als App-Icon das Flutter-Logo verwendet, um dies zu verändern ist das Konsolentool `flutter_launcher_icons`, wie andere Bibliotheken einzubinden. Zuerst ist es nötig drei Bilder (.png) zu erstellen, wobei eines das Logo mit transparentem Hintergrund, eines der Hintergrund sein muss und eines das gesamte Icon.



Abbildung 86:
foreground.png



Abbildung 87:
background.png



Abbildung 88: appicon.png

Hierbei ist zu beachten, dass wenn das generierte Icon in der Größenordnung so aussehen soll wie das `appicon.png`, das Logo in Relation zum transparenten Hintergrund, in Abbildung 86, kleiner sein muss. Wenn die Bilder in einem Ordner, im Projektverzeichnis, gespeichert wurden (z.B. `PROJEKT\assets`), müssen die Pfade noch in der `pubspec.yaml`-Datei wie folgt eingefügt werden:

```
flutter_icons:
  image_path: "assets/appicon.png"
  android: true
  ios: true
  remove_alpha_ios: true
  adaptive_icon_background: "assets/background.png"
  adaptive_icon_foreground: "assets/foreground.png"
```

Abbildung 89: App-Icon Pfadeinbindung

Anschließend muss noch der Command zum Generieren des Icons in der Konsole ausgeführt werden und die App neu auf das Gerät geladen werden:

- `flutter pub run flutter_launcher_icons:main`



Abbildung 90: fertiges Logo am Home-Screen

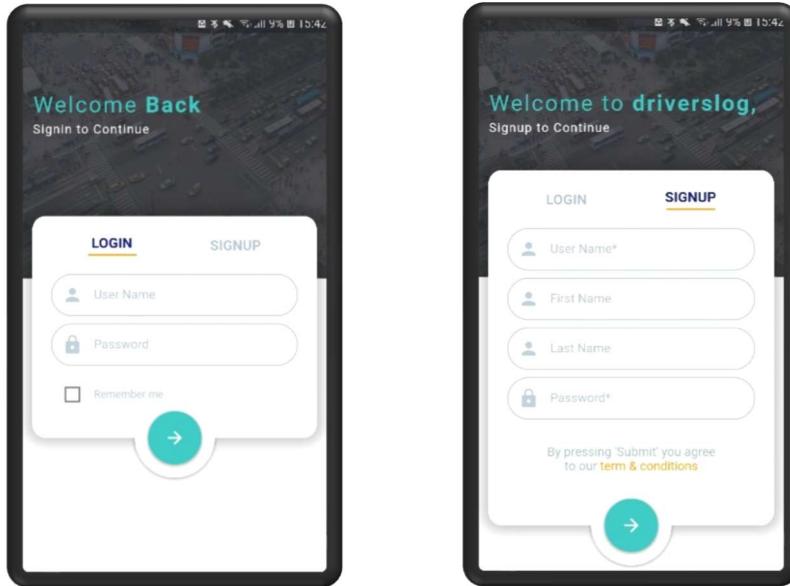
Das generierte Icon wird nur auf Android-Geräten angezeigt, auf IOS-Geräten wird das `appicon.png` angezeigt. [vgl. [25]]

4.2.9 Login-System

4.2.9.1 Allgemein

Es soll in der Datenbank überprüft werden, ob der vom Benutzer eingegebene Username, in der Datenbank eingetragen ist, falls dies der Fall ist, wird das Passwort überprüft, falls dies auch übereinstimmt, wird der User geloggt.

4.2.9.2 Login/Sign up-Benutzeroberfläche



Nach dem Starten der App wird der Login-Screen angezeigt. Es wird überprüft, ob der Benutzer auf SIGNUP klickt, wenn dies der Fall ist, wird eine Variable verändert, wodurch sich die Ansicht verändert.

4.2.9.3 Login-Funktion

Wenn der „→“-Button betätigt wird, wird zuerst überprüft, ob die beiden Felder, Username und Password, ausgefüllt sind und ob man sich aktuell in der Login-Ansicht befindet. Im Anschluss wird mit der .loginUser-Methode überprüft ob der der Username in der Datenbank bereits eingetragen ist. Wenn ja, setzt sie die Attribute des Objekts myUser (_username, _password, _firstname, _lastname, mem_id, _birthdate, _rfid) auf die Werte, welche in der Datenbank gespeichert sind und gibt true zurück, ansonsten gibt sie false zurück. Wenn der Username vorhanden war, wird mit .checkUser das eingegebene Passwort mit dem _password aus der Datenbank verglichen und bei Übereinstimmung das isChecked-Attribut auf true gesetzt. Wenn dies auch erfüllt ist, wird noch die Remember Me-Funktion aufgerufen, alle Fahrdaten des Users heruntergeladen und anschließend zum Main-Screen navigiert.

```

if (!isSignupScreen &&
    usernameController.text != "" &&
    passwordController.text != "") {
  bool isDefined =
    await myUser.loginUser(usernameController.text);
  if (isDefined) {
    myUser.checkUser(passwordController.text);
    if (myUser.getisChecked()) {
      _handleRemeberme(isRememberMe);
      await triplistDisplay.setList();
      Navigator.of(context).pushNamed('/Main');
    } else {
      ...
    }
  }
}

```

Abbildung 91: Login-Funktionsaufruf

4.2.9.4 Sign up-Funktion

Sollte der „→“-Button beim SIGNUP-Screen betätigt werden, wird zuerst überprüft, ob die beiden Pflichtfelder ausgefüllt sind. Die .signinUser-Methode überprüft, ob der Username schon verwendet wird. Ist er noch nicht in der Datenbank vorhanden, wird ein neuer Eintrag mit den eingegebenen Daten erstellt. Im Anschluss wird mit dem neu erstellten Account die .loginUser-Methode ausgeführt, wenn alles erfolgreich war, wird true zurückgegeben. Bei einem Rückgabewert von true wird der Nutzer auf den Main-Screen weitergeleitet.

```

} else if (isSignupScreen) {
  if (usernameController.text == "" ||
      passwordController.text == "") {
    return;
  }
  bool isSet = await myUser.signinUser(
    usernameController.text,
    firstnameController.text,
    lastnameController.text,
    passwordController.text);
  if (isSet) {
    Navigator.of(context).pushNamed('/Main');
  } else {
    ...
  }
}

```

Abbildung 92: Sign up-Funktionsaufruf

4.2.9.5 RememberMe

Die RememberMe-Funktion speichert, sofern der Nutzer die Funktion mit der Checkbox aktiviert, die Login-Daten in den Gerätespeicher, damit man seine Daten nicht jedes Mal neu eingeben muss. Folgende Funktion, welche in der shared_preferences-Library inkludiert ist, wird beim Login ausgeführt und speichert die Login-Daten in den Gerätespeicher:

```
void _handleRememberMe(bool value) {
    isRememberMe = value;
    SharedPreferences.getInstance().then(
        (prefs) {
            prefs.setBool("remember_me", value);
            prefs.setString('username', usernameController.text);
            prefs.setString('password', passwordController.text);
        },
    );
    setState(() {
        isRememberMe = value;
    });
}
```

Abbildung 93: Daten in den Gerätespeicher speichern

Die .loadUserPassword-Funktion wird in der initState-Methode, welche einmalig beim Aufruf der Login/Sign in-Page ausgeführt wird, aufgerufen und versucht Daten aus dem Gerätespeicher zu laden, wenn Daten vorhanden sind, werden sie in die Felder eingefüllt.

```
void _loadUserPassword() async {
    try {
        SharedPreferences _prefs = await SharedPreferences.getInstance();
        var _email = _prefs.getString("username") ?? "";
        var _password = _prefs.getString("password") ?? "";
        var _rememberMe = _prefs.getBool("remember_me") ?? false;
        if (_rememberMe) {
            setState(() {
                isRememberMe = true;
            });
            usernameController.text = _email;
            passwordController.text = _password;
        }
    } catch (e) {
        return;
    }
}
```

Abbildung 94: Daten aus dem Gerätespeicher laden

4.2.10 Bluetooth Verbindung

4.2.10.1 Allgemein

Bluetooth ist eine kabellose Punkt-zu-Punkt-Übertragungsmethode, zur Übertragung von Sprache und Daten zwischen zwei Endgeräten. Es ist für eine kurze Distanz und zur Übertragung geringer Datenmengen gedacht und stellt für jene Daten die Ideallösung dar. Voraussetzung zur Herstellung einer Verbindung ist ein eingebauter Bluetooth-Chip und ein Bluetooth unterstützendes Betriebssystem. Die Adressierung der Geräte erfolgt über die 48 Bit lange MAC-Adresse. [vgl. [26]]

4.2.10.2 Bluetooth Versionen

Bluetooth-Version	Veröffentlichung	Max. Datenrate
Bluetooth 1.0a	Juli 1999	732,2 kbit/s
Bluetooth 1.0b	Dezember 1999	732,2 kbit/s
Bluetooth 1.1	Februar 2001	732,2 kbit/s
Bluetooth 1.2	November 2003	1 Mbit/s
Bluetooth 2.0 + EDR	November 2004	2,1 Mbit/s
Bluetooth 2.1 + EDR	August 2007	2,1 Mbit/s
Bluetooth 3.0 HS	April 2009	24 Mbit/s
Bluetooth 4.0 LE	Dezember 2009	24 Mbit/s
Bluetooth 4.1	Dezember 2013	25 Mbit/s
Bluetooth 4.2	Dezember 2014	25 Mbit/s
Bluetooth 5	Dezember 2016	50 Mbit/s
Bluetooth 5.1	Januar 2019	50 Mbit/s
Bluetooth 5.2	Januar 2020	50 Mbit/s

Tabelle 9: Bluetooth Versionen [vgl. [26]]

Die Versionen 1-3 werden als Bluetooth Classic und seit Bluetooth 4.0 LE werden sie als Bluetooth Low Energy (BLE) bezeichnet. Während Android mithilfe einer Dual-Mode-Implementierung mit allen Versionen kompatibel ist, lassen sich Apple-Geräte nur mit BLE-Modulen verbinden. Da das verwendete Bluetooth Modul (HC-05) mit Bluetooth 2.0 + EDR ausgestattet ist, resultierend daraus ist die Smartphone-App nur mit Android kompatibel. [vgl. [27], [28]]

4.2.10.3 Bluetooth-Funktionen in der App

`flutter_bluetooth_serial` ist ein Package zur Implementierung von Bluetooth-Funktionen, welches nur Bluetooth Classic unterstützt und somit auch nur Android kompatibel ist. Nach dem Einbinden der Library können die Funktionen wie folgt verwendet werden: [vgl. [29]]

Setzen der Bluetooth Berechtigungen

Im Flutter-App-Dateiverzeichnis existiert eine „`AndroidManifest.xml`“-Datei, in der folgende Berechtigungen, zur Verwendung von Bluetooth, vergeben werden müssen:

NAME	BERECHTIGUNG	VERSION
BLUETOOTH	Verbindungsanfrage, Verbindungsannahme, Datenaustausch	<= Android 11
BLUETOOTH_ADMIN	Gerätesuche starten	<= Android 11
BLUETOOTH_SCAN	Gerätesuche starten, Verbindungsanfrage	>= Android 12
BLUETOOTH_CONNECT	Verbinden mit schon zuvor verbundenen Geräten	>= Android 12

Tabelle 10: Bluetooth Berechtigungen

[vgl. [30]]

```
<manifest
    <uses-permission android:name="android.permission.BLUETOOTH"
                      android:maxSdkVersion="30" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
                      android:maxSdkVersion="30" />
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
    <uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
    ...
</manifest>
```

Abbildung 95: Bluetooth Berechtigungen in "AndroidManifest.xml"

Bluetooth-Verwendung in der App

Zuerst muss eine globale Variable, mit variablem Datentyp, erstellt werden, in welcher später die Verbindungsdaten gespeichert werden.

```
var connection;
```

Abbildung 96: globale Variable zum Speichern der Verbindungsdaten

Dann muss eine Variable des Typs `BluetoothState` erstellt werden, in welcher gespeichert wird, ob Bluetooth in den Geräteeinstellungen ein- oder ausgeschalten ist.

```
BluetoothState _bluetoothState = BluetoothState.UNKNOWN;
```

Abbildung 97: *Bluetooth-Status Variable*

In der `initState`-Methode, welche beim ersten Aufruf der Page-Klasse, auf welcher die Bluetooth-Verbindung gestartet werden soll, ausgeführt wird, muss zuerst der Bluetooth-Status abgefragt werden, welcher in der `_bluetoothState`-Variable gespeichert ist. Danach wartet die `.listen`-Methode im Hintergrund auf eine Änderung des Status und verändert die Variable wenn nötig.

```
@override
void initState() {
    super.initState();

    FlutterBluetoothSerial.instance.state.then((state) {
        setState(() {
            _bluetoothState = state;
        });
    });

    FlutterBluetoothSerial.instance
        .onStateChanged()
        .listen((BluetoothState state) {
    setState(() {
        _bluetoothState = state;
    });
});
}
```

Abbildung 98: *initState() Check Bluetooth Status*

Wenn eine Verbindung zwischen der App und einem Bluetooth-Gerät aufgebaut werden soll, müssen folgende Methoden aufgerufen werden:

```

if (!isConnected()) {
  if (!_bluetoothState.isEnabled) {
    bool? requestreturn = await FlutterBluetoothSerial
      .instance
      .requestEnable();
    if (requestreturn != true) {
      return;
    }
  }
  final BluetoothDevice selectedDevice =
    await Navigator.of(context).push(
      MaterialPageRoute(
        builder: (context) {
          return const DiscoveryPage();
        },
      ),
    );
  BluetoothConnection.toAddress(selectedDevice.address)
    .then((_connection) {
      setState(() {
        connection = _connection;
        verifyBT();
      });
    });
}
}

```

Abbildung 99: Bluetooth-Verbindungsauftbau

Zuerst wird überprüft, ob bereits eine Verbindung besteht, falls dies nicht der Fall ist, wird überprüft, ob am Smartphone Bluetooth aktiviert ist, wenn nicht, wird ein Fenster angezeigt, in welchem man es aktivieren kann:

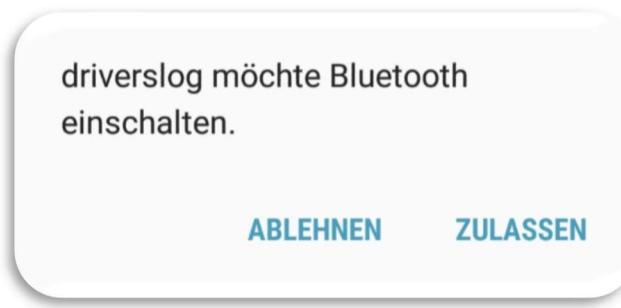


Abbildung 100: Bluetooth Aktivierungsanfrage-Fenster

Auf „ABLEHNEN“ folgt ein Abbruch des Verbindungsauftbaus, wenn „ZULASSEN“ gedrückt wird, wird die Bluetooth-Funktion des Smartphones aktiviert. Als nächstes wird mithilfe des Navigators eine neue Seite aufgerufen, welche eine Liste mit Bluetooth-Geräten in der Nähe anzeigt und bei einem Klick auf das Feld, mit dem richtigen Gerät, die Verbindungsdaten des ausgewählten Geräts zurückgegeben. Alternativ kann, wenn die Verbindung immer mit dem gleichen Gerät stattfindet, direkt die MAC-Adresse

angegeben werden. Jene Daten (MAC-Adresse) werden in der nächsten Funktion zum Verbinden benötigt und in der globalen Variable „connection“ gespeichert. Das Senden und Empfangen von Daten funktioniert wie folgt:

```
void sendMessage(String text) async {
    connection.output.add(utf8.encode(text + "\r\n"));
    await connection.output.allSent;
}
```

Abbildung 101: Funktion zum Senden von Bluetooth-Daten

Der sendMessage-Funktion wird ein Text mitgegeben, welcher bei Aufruf dem Output hinzugefügt wird und anschließend wird der Output an jenes Gerät gesendet, welches in der „connection“-Variable gespeichert ist.

```
void verifyBT() async {
    sendMessage("c");

    connection.input!.listen((Uint8List data) {
        print(ascii.decode(data));
    }).onDone(() {});
}
```

Abbildung 102: Funktion zum Empfangen von Bluetooth-Daten

Die Funktion verifyBT() sendet Daten und wartet, mit der .input!.listen-Methode im Hintergrund, auf eine Rückmeldung, welche in der Konsole ausgegeben wird.

4.2.11 Fahrdatenermittlung

4.2.11.1 Allgemein

Die benötigten Fahrdaten werden mithilfe einer Kombination aus verschiedenen Bibliotheken ermittelt. Die in Abbildung 103 ersichtlichen Parameter werden in einem Objekt gespeichert und später an die Datenbank gesendet.

Fahrtenprotokoll
gemäß § 19 Abs. 8 FSG

Datum	Gefahrene KM	Kilometerstand von bis	Kfz-Kennzeichen	Tageszeit	Fahrtstrecke/-ziel	Straßenzustand, Witterung	Unterschrift des Begleiters	Unterschrift des Bewerbers
-------	--------------	------------------------	-----------------	-----------	--------------------	---------------------------	-----------------------------	----------------------------

Abbildung 103: Fahrtenprotokoll

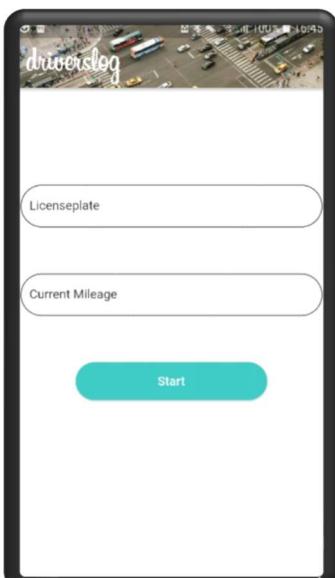
4.2.11.2 Trip-Klasse

In der Trip-Klasse werden alle benötigten Daten zu einer Fahrt gespeichert.

```
class Trip {
    late int tripnr;
    late Date date;
    late Time time;
    late Time stoptime;
    late double kilometres;
    late int mileagestart;
    late int mileagestop;
    late String licenseplate;
    late String startlocation;
    late String stoplocation;
    late String weather;
    ...
}
```

Abbildung 104: Parameter der Trip-Klasse

4.2.11.3 Start einer Fahrt



Wenn eine Fahrt gestartet wird, erscheint ein Screen, auf dem man das Kennzeichen und den aktuellen Kilometerstand des Fahrzeugs eingeben muss. Das Kennzeichen wird in den TextEditingController „licenseplate“ und der Kilometerstand in jenen namens „mileage“ gespeichert. Beim Betätigen des Start-Buttons wird zuerst die Validität des Kennzeichens überprüft. Das heißt das Feld ist nicht leer und hat weniger als 10 Zeichen (8 + 2 potenzielle Leerzeichen). Weiters wird überprüft, ob das Feld mileage leer ist, in welchen nur Ziffern und Punkte eingegeben werden können.

Abbildung 105: Start-Screen einer Fahrt

Wenn alle Bedingungen erfüllt sind, werden mögliche Leerzeichen und Punkte aus den Controllern entfernt und jeweils in einen String gespeichert.

```
void startbuttonPressed(BuildContext context) {
    if (licenseplate.text.length < 11 &&
        licenseplate.text.isNotEmpty &&
        mileage.text.isNotEmpty) {
        String licenseString = licenseplate.text.replaceAll(' ', '');
        String mileageString = mileage.text.replaceAll('.', '');
        Trip localtrip = Trip(mileageString, licenseString);
        tripDisplay = localtrip;
        matrixOn();
        Navigator.of(context).pushNamed('/DrivingDisplay');
    }
}
```

Abbildung 106: aufgerufene Funktion beim Start einer Fahrt

Anschließend wird der Standardkonstruktor aufgerufen, mit welchem ein Objekt localtrip vom Datentyp Trip erstellt wird. Dieses Objekt wird in ein globales Objekt (tripDisplay) gespeichert, um auf dem nächsten Screen darauf zugreifen zu können.

tripnr (Integer)

Die tripnr wird beim ersten Datenbankaufruf (nach dem Login) auf die Länge der Liste, der bereits vorhandenen Fahrten, plus 1 gesetzt.

date(Date), time(Time)

date und time werden, beim Aufruf des Standardkonstruktors, auf das aktuelle Datum bzw. auf die aktuelle Zeit gesetzt. Dies wird durch die von Flutter bereitgestellte Klasse DateTime mit dem .now-Konstruktor, in den jeweiligen Konstruktoren, umgesetzt.

stoptime(Time)

Die stoptime wird am Ende einer Fahrt, durch Drücken des Stop-Buttons, ebenfalls mit DateTime.now() im Date-Konstruktor, auf die aktuelle Zeit gesetzt.

kilometres(double)

kilometres wird im Konstruktor auf 0 gesetzt und während einer Fahrt alle 10 Sekunden, durch das Auslesen der GPS-Daten und das Berechnen der Differenz, aktualisiert. Für das Auslesen der GPS-Daten wird die location-Library verwendet, welche Längengrad und Breitengrad zurückgibt. Mit der Haversine-Formel wird die Differenz zwischen zwei GPS-Punkten berechnet. [vgl. [31]]

$$a =$$

$$\frac{0,5 - \cos((LATL - LAT) * k) + \cos(LAT * k) * \cos(LATL * k) * (1 - \cos((LONGL - LONG) * k))}{2}$$

$$s = 12742 * \sin^{-1}(\sqrt{a})$$

k ... Faktor...k = 0,017453292519943295

a ... Zwischenergebnis

s ... Strecke zwischen den 2 Punkten in km

LONG ... Längengrad des aktuellen Punktes

LAT ... Breitengrad des aktuellen Punktes

LONGL ... Längengrad des letzten Punktes

LATL ... Längengrad des letzten Punktes

[vgl. [32]]

Der resultierende Wert wird zum bisherigen Wert von kilometres addiert.

mileagestop(Integer)

Resultiert, am Ende einer Fahrt, aus der Summe von mileagestart und kilometres.

startlocation(String), stoplocation(String)

Die Ortsnamen werden mithilfe der geocoding-Library, welche durch Längengrad und Breitengrad die aktuelle Adresse zurückgibt, am Beginn bzw. am Ende der Fahrt ermittelt. [vgl. [33]]

```
Future<String> getAdress() async {
    List<geo.Placemark> placemarks =
        await geo.placemarkFromCoordinates(_latitude, _longitude);
    return placemarks[0].locality.toString();
}
```

Abbildung 107: Verwendung der geocoding-Library

weather(String)

Das Wetter wird zu Beginn der Fahrt durch die weather-Library und der startlocation als Parameter ermittelt. Den key erlangt man bei der Registrierung auf der Webseite von OpenWeatherMap. [vgl. [34]]

```
WeatherFactory wf = WeatherFactory(key, language: Language.GERMAN);
Weather w =
    await wf.currentWeatherByCityName(tripDisplay.startlocation);
```

Abbildung 108: Verwendung der weather-Library

4.2.12 Kommunikation mit der Datenbank

4.2.12.1 Allgemein

Die Kommunikation zwischen der Smartphone-App und der Datenbank findet mit HTTP-Requests, wozu die http-Library verwendet wird, statt. Diese Requests werden in der App zum Login und zum Laden und Speichern der Fahrdaten verwendet.

4.2.12.2 GET-Request

Der GET-Request dient zum Laden aus der Datenbank und wird zum Beispiel bei Login-Anfragen verwendet. Um auf die gewünschte Datei am Server zugreifen zu können, muss wie folgt eine Anfrage gesendet werden:

```
Response response = await get(
    Uri.parse('$url/rest/arrestDB/index.php/member/username/$username'));
List result = jsonDecode(response.body);
```

Abbildung 109: GET-Request

Die get-Funktion gibt ein Response-Objekt zurück, welches aus Header und Body besteht. Da in diesem Fall nur der Body relevant ist, wird nur dieser mit der jsonDecode-Funktion, wofür man „dart:convert“ importieren muss, entschlüsselt und in eine Liste gespeichert.

4.2.12.3 POST-Request

Der POST-Request wird zum Speichern von Daten in der Datenbank verwendet und wird in der App zu Beispiel zum Erstellen neuer Accounts verwendet. Um Daten auf den Datenbankserver zu speichern, müssen Sie zuerst mit jsonEncode verschlüsselt werden und dann im Body mitgegeben werden.

```
await post(Uri.parse('$url/rest/arrestDB/index.php/member/'),
  headers: <String, String>{
  'Content-Type': 'application/json; charset=UTF-8',
},
body: jsonEncode(<String, String>{
  "username": username,
  "firstname": firstname,
  "lastname": lastname,
  "password": password
}));
```

Abbildung 110: POST-Request

4.3 Datenbank

4.3.1 Datenbank Grundlagen

Eine Datenbank ist ein System, welches zur Speicherung und Verwaltung von Daten verwendet wird. Auf ein solches Datenverwaltungssystem kann elektronisch zugegriffen werden. Zum Datenbanksystem (DBS) zählen zwei Elemente, zum einen die Datenbank selbst und zum anderen das Datenbankmanagementsystem. Vorteile gegenüber anderen Speichermedien sind etwa die Geschwindigkeit, die Skalierbarkeit und das von mehreren Anwendern gleichzeitig darauf zugegriffen werden kann.

4.3.1.1 Datenbankmanagementsystem

Um eine Datenbank verwalten zu können ist ein Datenbankmanagementsystem (DBMS) notwendig. Es ist eine Systemsoftware, die dem Administrator erlaubt alle notwendigen Einstellungen zu treffen.

4.3.2 Wahl der Datenbank

Anforderung an die Datenbank sind:

- ➔ Wenig Redundanz bei Speicherung der Daten (Mehrfachspeicherung vermeiden)
- ➔ Möglichkeit Daten abzufragen und zu ändern
- ➔ Zuverlässigkeit (immer ansprechbar)
- ➔ Sicherheit (einfache Erstellung von Backups)

Um diese Anforderung zu erfüllen, bietet sich eine relationale Datenbank an. Bei dieser Variante werden die Informationen in Tabellen gespeichert. Um die eingeschriebenen Daten abzufragen und zu manipulieren, wird die Datenbanksprache SQL (Structured Query Language) verwendet.

4.3.2.1 Vergleich SQLite und MySQL

Zwei Vertreter der RDBMS (Relational Database Management System) sind MySQL und SQLite.

Datentypen:

SQLite verfügt über die gängigsten Datentypen wie: Integer, Text (String), Real (ersetzt

in SQLite das Double), Null und Blob. MySQL verfügt über die oben genannten und noch rund 20 weitere.

Systemaufbau:

Im Vergleich zu dem bekanntesten relationalen Datenbanksystem MySQL besteht SQLite aus nur einem einzigen File. Außerdem braucht es nur ca. 25KB an Speicher, wobei MySQL rund 600MB verwendet. Dabei ist zu erwähnen das bei SQLite der Speicherverbrauch mit der Größe der zu speichernden Daten wächst, was die Leistungsoptimierung erschwert.

Benutzerverwaltung:

MySQL hat ein sehr ausgeklügeltes Benutzerverwaltungssystem, welches verschiedene Benutzer verwalten kann und auch zwischen verschiedenen Levels von Zugriffsrechten unterscheiden kann. SQLite im Gegensatz hat keine Benutzerverwaltung. [vgl. [35]]

4.3.3 SQLite

Da es in diesem Fall nicht notwendig ist viele Datensätze zu speichern, fiel die Entscheidung auf SQLite in der Version 3.35.5. Weiters bietet die kompakte Datenbank den Vorteil das Backups schnell und unkompliziert durchführbar sind. Die Datenbank ist mit den SQL-Befehlen bedienbar und bietet eine große Auswahl an verfügbaren Programmierschnittstellen.



*Abbildung 111 SQLite Logo [vgl.
[50]]*

4.3.3.1 SQLite Browser

SQLite Browser ist ein grafisches Frontend und bietet neben dem Erstellen und Bearbeiten einer Datenbank noch viele weitere Features, wie z.B. das Manipulieren von Datensätzen. Darüber hinaus bietet dieses Tool viele Funktionen, um eine bestehende Datenbank zu manipulieren, also gewisse Datensätze zu suchen, und diese zu ändern. [vgl. [36]]

4.3.3.2 Datenbank erstellen

Der nächste Schritt ist die Datenbank zu erstellen. Dafür bietet SQLite Browser die Funktion dies einfach per Knopfdruck zu erledigen.

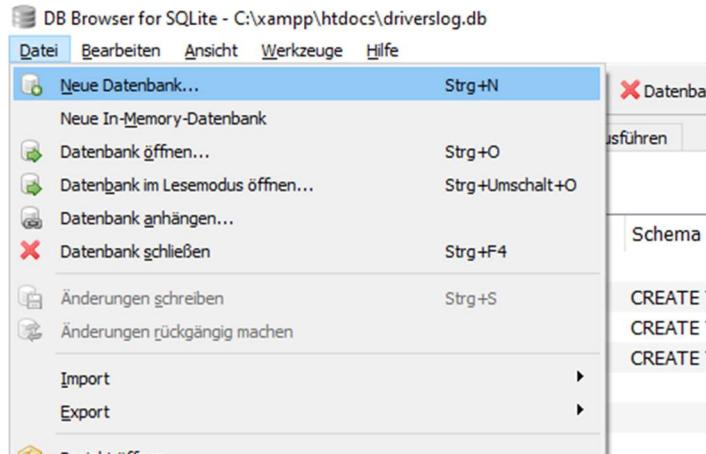


Abbildung 112: Datenbank erstellen mit SQLite Browser

In weiterer Folge galt es herauszufinden, welche Tabellen und Attribute benötigt werden. Aus diesem Grund ist es sinnvoll sich mit dem Prinzip der Relationalen Datenbank auseinanderzusetzen.

4.3.4 Relationale Datenbank

Wie unter 4.3.2.1 erwähnt ist SQLite ein relationales DBS und ist in Entitäten (Tabellen) strukturiert. Dabei ist es wichtig einige grundlegende Begriffe zu definieren.

<u>LieferNR</u>	Name	Adresse	Datum	Artikel
1	Max Mustermann	Musterstraße 3, 4050 Beispieldorf	01.01.2022	Fahrrad

Tabelle 11: Relationales Datenbankmodell: Tabellenbeispiel

Die Spaltennamen (grau schattiert) werden dabei als Attribute bezeichnet. Eine Zeile, also ein Datensatz, wird auch Tupel genannt. Eine weitere wichtige Eigenschaft sind die Schlüssel. Hierbei wird zwischen Primary Keys (Primärschlüssel) und Foreign Keys (Fremdschlüssel) unterschieden. [vgl. [36]]

4.3.4.1 Primary Key

Unter dem Begriff Primary Key versteht man diejenigen Attribute die notwendig sind, um einen Datensatz eindeutig zu identifizieren. In der obigen Tabelle wurde die LieferNR als Primary Key gewählt, weil es das einzige Attribut ist, das unmöglich doppelt vorkommen kann, da eine Liefernummer einzigartig ist. [vgl. [36]]

4.3.4.2 Foreign Key

Ein Foreign Key (Fremdschlüssel) ist der Primary Key aus einer fremden Tabelle. Er dient dazu die beiden Tabellen zu verknüpfen. Im unten angeführten Beispiel gibt es eine Entität Schüler und eine Entität Adresse. Die Postleitzahl ist in der Tabelle Adresse der Primärschlüssel und in der Tabelle Schüler der Fremdschlüssel. [vgl. [36]]

Schüler

Vorname	Nachname	Straße	<u>PLZ</u>
Max	Mustermann	Musterstraße 3	4866

Tabelle 12: Foreign Key Beispiel Schüler

Adresse

PLZ	Ort
4866	Unterach am Attersee

Tabelle 13: Foreign Key Beispiel Adresse

4.3.4.3 Normalformen

Um in einer Datenbank die Redundanz niedrig zu halten, gibt es die Normalformen. Es gilt sein System (wenn möglich) in eine möglichst hohe Normalform zu bringen. Anhand des Beispiels der driverslog Datenbank ist gut zu sehen, wie die Normalformen durchlebt werden. [vgl. [37]]

0.Normalform

In dieser NF (Normalform) werden alle Daten in einer Tabelle gesammelt. Meist wird das zu Beginn der Datenbankentwicklung so gemacht, um einen groben Überblick zu bekommen.

Für driverslog haben sich folgende Attribute gefunden:

trip NR	Date	Time start- stop	Km start- stop	Km	Route	Number plate	Weather	Username	Name	Birthdate	RFID
1	04.03.2022	15:10- 19:30	225- 525	300	Salzburg -Wien	VB 321D	Sonne	d.wipplinger	Daniel Wipplinger	21.01.2003	adg- h7z- as2

Tabelle 14: 0.Normalform

1.Normalform

Um die Datenbank in die erste Normalform zu heben, ist es notwendig die Attribute zu zerkleinern. Grundsätzlich bedeutet das seine Attribute atomar (nicht mehr kleiner zerteilbar) abzuspeichern. Jedoch soll das nur so weit betrieben werden, wie es sinnvoll ist. Im vorliegenden Fall bietet es sich an den Namen in Vor- und Nachnamen zu unterteilen.

tripNR	Date	Start time	Stop time	Start km	Stop km	Km	Route	Number plate	Weather	Username	Firstrame	Lastname	Birthdate	RFID
1	04.03.2022	15:10	19:30	225	525	300	Salzburg-Wien	VB 321D	Sonne	d.wipplinger	Daniel	Wipplinger	21.01.2003	adg-h7z-as2

Tabelle 15: 1.Normalform

2. Normalform

Die zweite NF kann nur erreicht werden, wenn die Erste erfüllt ist, und jedes Nichtschlüsselattribut von einem Schlüsselattribut abhängig ist. Das bedeutet, dass Zusammenhänge in eigene Tabellen ausgelagert werden müssen. Für die bisherige Tabelle war ID + Username der Primary Key. Im nächsten Schritt muss jedes Attribut vom Primärschlüssel voll funktional abhängig sein. Die Attribute Date bis Weather sind alle von der Fahrt abhängig und somit von ID. Ganz anders bei Firstrame, Lastname Birthdate und RFID. Diese sind alle von der Person abhängig, also vom Username.

Benutzer

Username	Firstrame	Lastname	Birthdate	RFID
d.wipplinger	Daniel	Wipplinger	21.01.2003	adg-h7z-as2

Tabelle 16: 2.Normalform Tabelle Benutzer

Fahrt

tripNR	Date	Start time	Stop time	Start km	Stop km	Km	Route	Number plate	Weather
1	04.03.2022	15:10	19:30	225	525	300	Salzburg-Wien	VB 321D	Sonne

Tabelle 17: 2.Normalform Tabelle Fahrten

Da man wissen muss, welche Fahrt, zu welchem Benutzer zuzuordnen ist, bietet es sich an, eine ID für jeden Benutzer anzulegen. Diese ID wird dann in der Fahrtentabelle als Foreign Key eingeführt. Somit lassen sich alle Fahrten einem Benutzer zuordnen.

Benutzer

ID	Username	Firstname	Lastname	Birthdate	RFID
1	d.wipplinger	Daniel	Wipplinger	21.01.2003	adg-h7z-as2

Tabelle 18: 2.Normalform Tabelle Benutzer

Fahrten

tripNR	Date	Start time	Stop time	Start km	Stop km	Km	Route	Numberplate	Weather	ID
1	04.03.2022	15:10	19:30	225	525	300	Salzburg-Wien	VB 321D	Sonne	1

Tabelle 19: 2.Normalform Tabelle Fahrten

3.Normalform

Ziel dieser Normalform ist es, Anomalien zu verhindern. Im Grunde heißt das, dass alle direkt abhängigen Attribute in eine Tabelle auszulagern sind, um mehrfache Abspeicherungen zu vermeiden. Da ein solches Vorgehen bei der driverslog Datenbank nicht sinnvoll wäre, wird es hier anhand eines anderen Beispiels veranschaulicht.

Mitglieder

Vorname	<u>Nachname</u>	Adresse
Max	Mustermann	Musterstraße 3

Tabelle 20: 3.Normalform Tabelle Mitglieder

Lieferung

<u>Liefernummer</u>	Adresse
4	Musterstraße 3

Tabelle 21: 3.Normalform Tabelle Lieferung

Besser wäre es wie folgt:

Mitglieder

Vorname	<u>Nachname</u>	<u>Adressnummer</u>
Max	Mustermann	1

Tabelle 22: 3.Normalform Tabelle Mitglieder

Lieferung

<u>Liefernummer</u>	<u>Adressnummer</u>
1	1

Tabelle 24: 3.Normalform Tabelle Lieferung

Adresse

<u>Adressnummer</u>	Adresse
1	Musterstraße 3

Tabelle 23: 3.Normalform Tabelle Adresse

4.3.4.4 Beziehungen

Ein Objekt kann zu mehreren gleichartigen Objekten eine Beziehung haben. Dabei kann die Anzahl der Objekte variieren. Diese Mengenangaben nennt man Kardinalitäten. [vgl. [38]]

Dabei unterscheidet man zwischen:

- 1:1 Beziehungen

Bei einer 1:1 Beziehung wird jedem Objekt A, in diesem Fall die Tabelle Angestellter, ein Objekt B, hier die Tabelle Personalakte, zugeordnet. Dies ist die am wenigsten verbreitete Form, da es oft möglich ist die Datensätze in einer Tabelle zusammenzufassen.

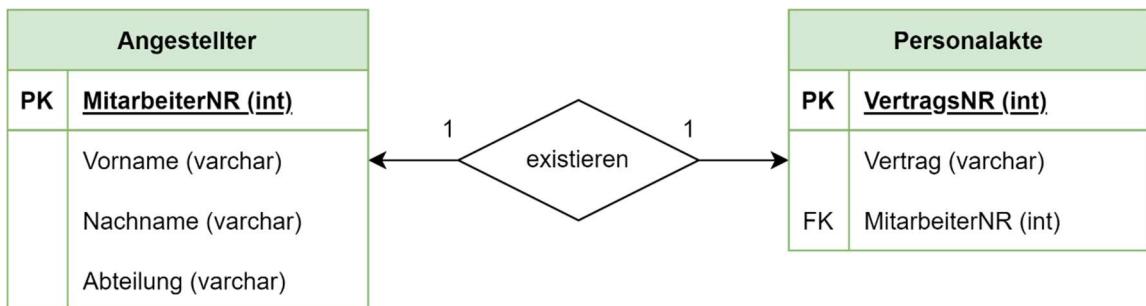


Abbildung 113: Beziehungen 1:1

- 1:n Beziehungen

Eine 1:n Beziehung bedeutet, dass es einseitig mehr Objekte gibt. In einer Bibliothek kann ein Leser ein bis mehrere Bücher ausleihen.

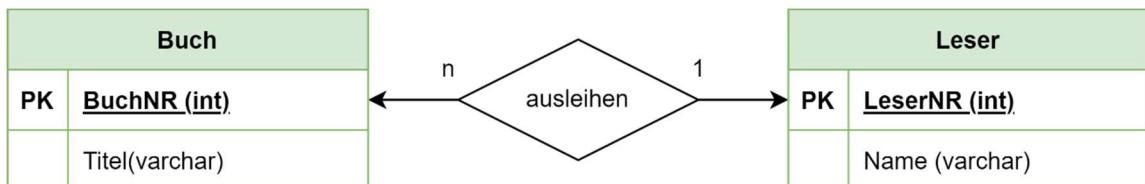


Abbildung 114: Beziehungen 1:n

- n:m Beziehungen

Bei dieser Art von Beziehung können sowohl auf Seite A als auch auf Seite B mehrere Objekte existieren. Im unteren Beispiel können an einem Kurs mehrere Mitarbeiter teilnehmen. Außerdem kann ein Mitglied an mehreren Kursen teilnehmen.

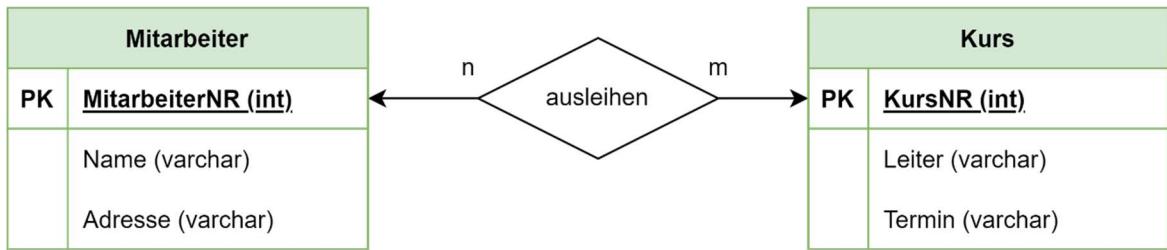


Abbildung 115: Beziehungen n:m

4.3.4.5 Driverslog Beziehungen

In der driverslog Datenbank haben wir eine 1:n Beziehung zwischen den Entitäten Fahrten und Benutzer. Ein Benutzer kann mehrere Fahrten machen, eine Fahrt kann aber nur einem Benutzer zugeordnet werden.

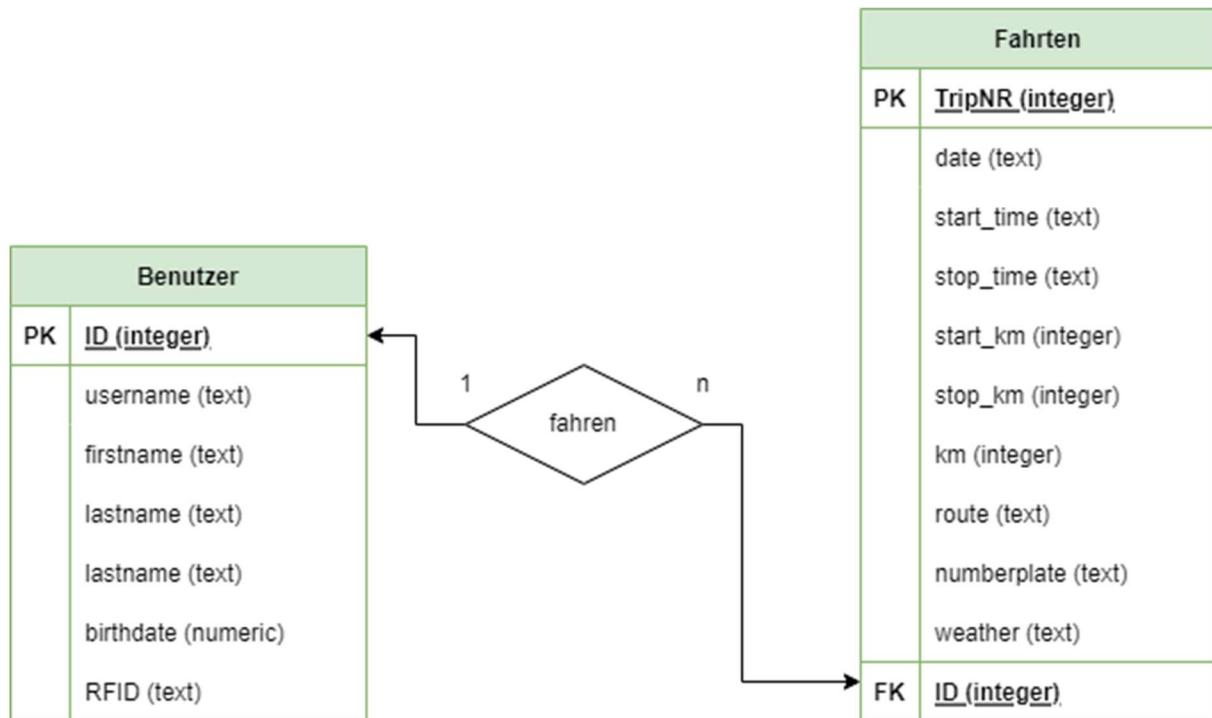


Abbildung 116: Driverslog Beziehungen/ ERM

4.3.4.6 Entity Relationship Modell (ERM)

Um einen Überblick zu gewinnen, wie die Datenbank eigentlich aussehen soll, ist es sinnvoll ein ERM zu erstellen. Dieses Diagramm wird gemäß den Anforderungen an die Datenbank modelliert. Es bietet eine gute Möglichkeit die Beziehungen zwischen den Entitäten darzustellen. Eine Abbildung des driverslog- ERM ist in Abbildung 116 zu sehen.

4.3.4.7 Erstellen der Tabelle

Nachdem klar ist welche Entitäten mit welchen Attributen benötigt werden, können diese nun erstellt werden. SQLite Browser bietet auch dafür ein UI die es ermöglicht die Tabellen zu erstellen und die Beziehungen festzulegen.

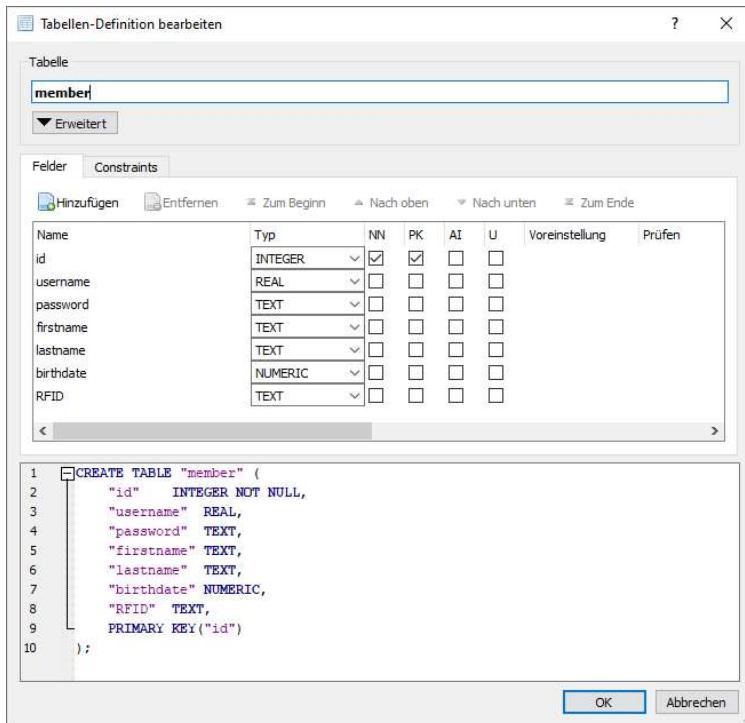


Abbildung 117: Erstellen einer Tabelle mittels SQLite Browser

Mit diesem Schritt ist das Grundgerüst der Datenbank fertig und sie kann mit Daten gefüllt werden. Im nächsten Schritt wird die Verfügbarkeit der Daten in Angriff genommen. Die Fahrschüler wollen sich auf einer Website anmelden und ihre Daten einsehen können.

4.3.4.8 Structured Query Language (SQL)

Nicht immer wird über Tools mit der Datenbank gesprochen. Für diese Anwendungsfälle gibt es die Datenbanksprache SQL. Dabei unterscheidet man noch zwischen DDL (Data Definition Language) und DML (Data Manipulation Language).

DDL

Die Datendefinitionssprache umfasst alle Befehle zur Veränderung der Struktur der Datenbank (z.B. erstellen neuer Tabellen, neuer Spalten)

```
CREATE TABLE member( mem_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
                     username TEXT,  
                     password TEXT,  
                     firstname TEXT,  
                     lastname TEXT,  
                     birthdate NUMERIC);
```

Abbildung 118: Beispiel DDL

DML

Wie sich aus dem Namen Datenmanipulationssprache schon ableiten lässt, handelt es sich hierbei um Statements die Daten löschen, hinzufügen oder verändern.

```
INSERT INTO `member` (username, password, firstname, lastname, birthdate)  
VALUES ('Maxi21', 123, 'Max', 'Mustermann', 21.01.2003);
```

Abbildung 119 Beispiel DML

4.3.5 Web-Applikation

Um von außen auf die Datenbank zugreifen zu können, wird eine WEB-Applikation benötigt. In diesem Fall ist das eine Website, die über Scripts direkt auf die Datenbank zugreift.

Web-Applikationen, müssen nicht heruntergeladen werden, sondern werden direkt über ein Netzwerk aufgerufen. Dabei liegt die tatsächliche Anwendung auf einem Webserver. Der Vorteil dabei ist, dass man von verschiedenen Geräten darauf zugreifen kann. Außerdem können mehrere Benutzer gleichzeitig auf dieselbe Version zugreifen. Es wird zwischen Client-Side-Rendering und Server-Side-Rendering unterschieden.

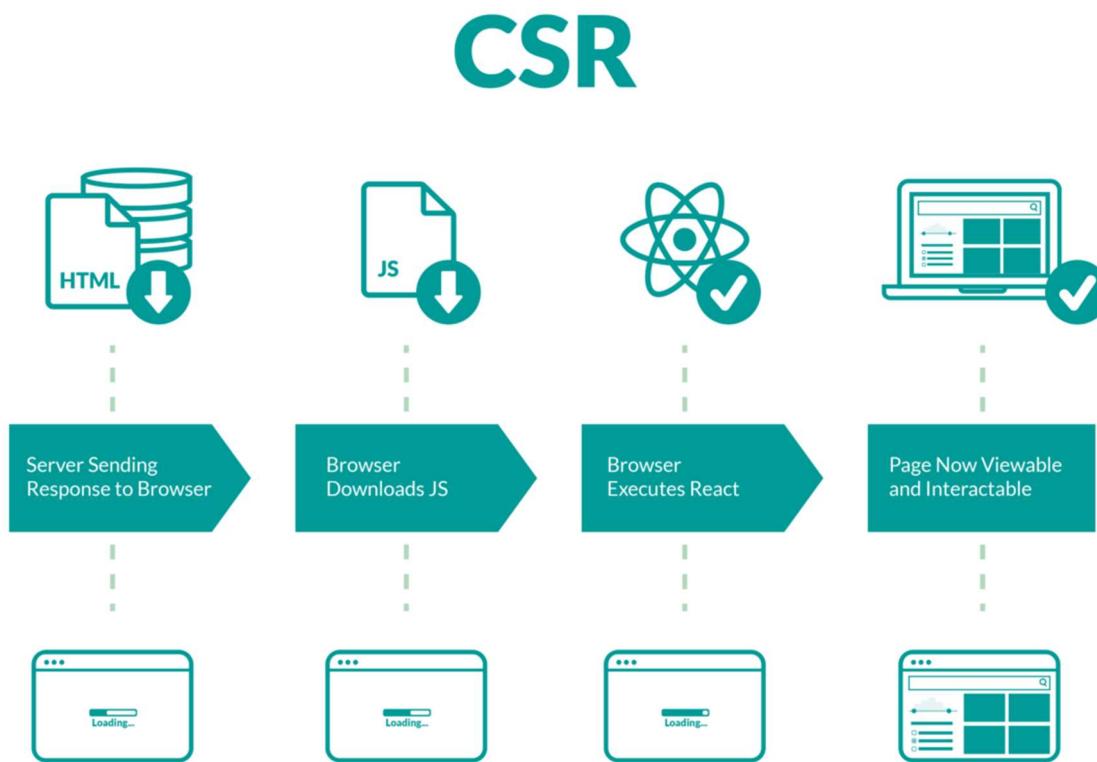


Abbildung 120: Grafik CSR Client-Side-Rendering [vgl. [39]]

Wie es der Name vermuten lässt, wird beim Server-Side-Rendering der Code am Server generiert und dann an den Client gesendet. In der Entwicklung ist ein solches System einfacher, einziger Nachteil ist, dass der Server stärker belastet wird.

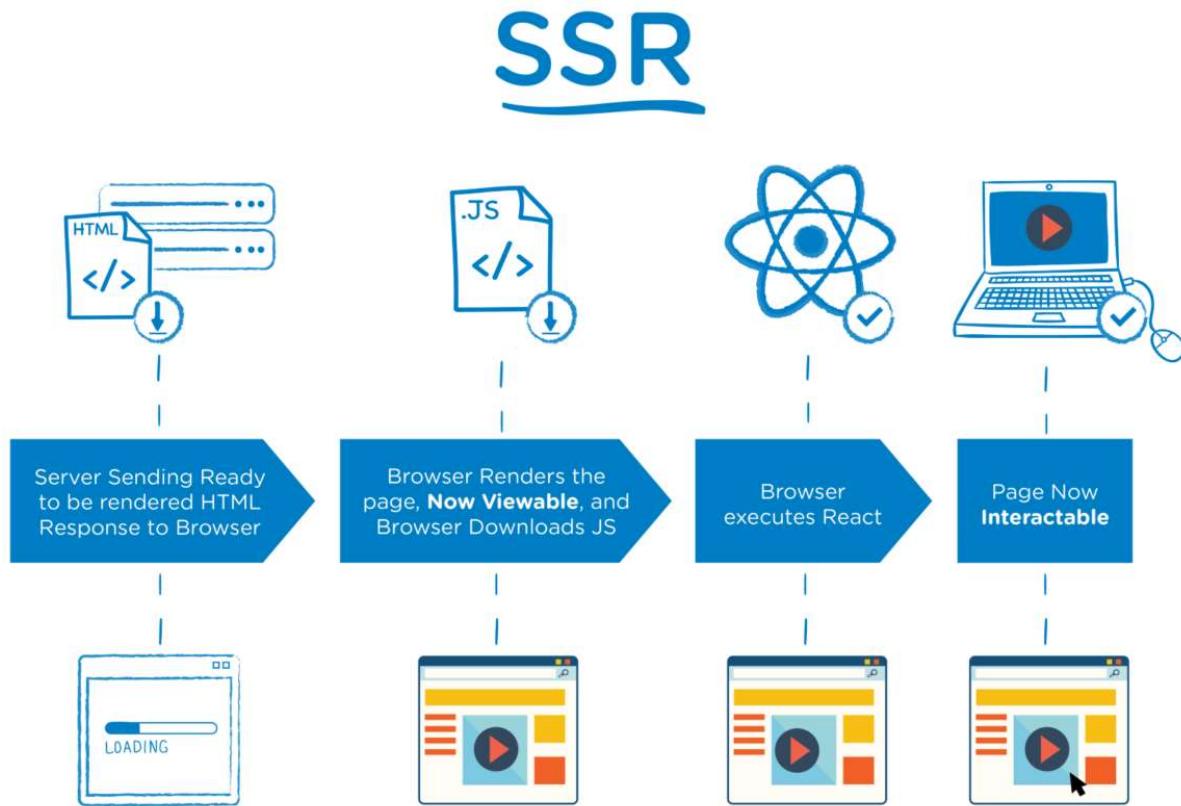


Abbildung 121: Grafik SSR Server-Side-Rendering [vgl. [39]]

Aus Gründen der Komplexität wird ein Backend System mit Server-Side-Rendering verwendet.

4.3.5.1 XAMPP

XAMPP ist ein freies Softwareprogrammpaket. Wobei das A für Apache und M für MySQL (seit 2015 MariaDB) steht. Die zwei P stehen für die Scriptsprachen Perl und PHP. Das X steht für „cross-plattform“, also das es auf verschiedenen Betriebssystemen funktioniert. XAMPP ist für die Entwicklung von Systemen gedacht, nicht etwa für ein Produktivsystem.

Außerdem bietet das Programmpaket noch weitere Dienste wie den FTP-Server (File-



Abbildung 122: XAMPP Logo [vgl. [51]]

Transfer-Protokoll) FileZilla, den Mailserver Mercury, und die Webanwendung phpMyAdmin.

Für die Entwicklung der driverslog WEB-Applikation ist nur der Apache Server interessant.

4.3.5.2 Scriptssprachen

Für die Backendprogrammierung stehen eine Vielzahl an Scriptssprachen zur Verfügung. Die relevantesten sind PHP, JavaScript und Python. Im folgenden Abschnitt werden die Vor und Nachteile erörtert.

- PHP (Hypertext Preprocessor)

PHP ist zum aktuellen Zeitpunkt die am meisten verbreitete Scriptssprache, wenn es um die Gestaltung von dynamischen Webseiten geht. In einigen Details ist PHP mit C oder Perl vergleichbar. Das macht die Syntax leichter verständlich, sofern schon Vorkenntnisse vorhanden sind. Außerdem kann die Scriptssprache in HTML eingebettet werden kann. Die Kombination von PHP und HTML bietet somit den perfekten Bausatz zum Erstellen einer dynamischen Website. Ein Nachteil ist, dass die Sprache nicht für große inhaltsbasierte Webanwendungen geeignet ist. Außerdem ist PHP etwas veraltet und kann in gewissen Punkten nicht mit den State of the Art Technologien mithalten kann. [vgl. [40]]



Abbildung 123: PHP-Logo [vgl. [54]]

- JavaScript

JavaScript oder einfach JS ist eine dynamische Scriptssprache zum Erstellen von interaktiven Webinhalten. Eigentlich ist JS hauptsächlich zur Frontend-Entwicklung gedacht. Durch Frameworks wie Node.js ist die Sprache aber auch für die Backend-Entwicklung brauchbar. Neben der geringen Serverbelastung ist die hohe Geschwindigkeit ein großer Nutzen von JS. Zu den Nachteilen zählt die Inkompatibilität mit älteren Browsern, sowie die Tatsache, dass der Code immer sichtbar ist. [vgl. [41]]



Abbildung 124: Java
Script Logo [vgl. [52]]

- Python

Python ist eine Hochsprache die sich leicht verstehen und lernen lässt. Das ist vor allem der klaren Syntax geschuldet. Außerdem ist die Sprache wenig fehleranfällig und gut skalierbar. Durch die wachsende Community werden auch immer mehr Bibliotheken zu der ohnehin schon großen Sammlung hinzugefügt. Nachteile sind etwa die langsame Geschwindigkeit und der relative hohe Speicherverbrauch. [vgl. [41]]



*Abbildung 125:
Python-Logo [vgl. [55]]*

Fazit

Die dynamischen Elemente der driverslog Website sowie die Datenbankzugriffe und die Benutzerverwaltung werden in PHP realisiert. Zum einen, weil HTML zum Strukturieren der Website verwendet wird, und zum anderen, weil schon Vorkenntnisse in C vorhanden sind.

4.3.5.3 PHP Basics

Oft wird PHP in anderen Dokumenten eingebettet. Um einen PHP Abschnitt zu starten wird der Tag „`<?php`“ verwendet. Am Ende des Abschnitts wird der Tag „`?>`“ zum Beenden verwendet. Ein Text kann mittels „`echo`“ ausgegeben werden.

```
<?php
echo "Hello World"
?>
```

Abbildung 126: PHP-Beispiel

Zum Ausführen muss das Programm auf einem Webserver abgespeichert werden. In unserem Fall ist das der Apache Server von XAMPP. Um das File auf dem Server zu speichern ist es notwendig das Installationsverzeichnis von XAMPP zu suchen. Dort ist der „`htdocs`“ Ordner zu finden. Will man das Script offen verfügbar machen, muss man es an diesem Ort speichern.

Wie in Abbildung 127 zu sehen kann das Programm ausgeführt werden, indem in einem Browser die Adresse eingegeben wird. Im Hintergrund wird dann das Script am Server kompiliert und das Ergebnis an den Client gesendet.

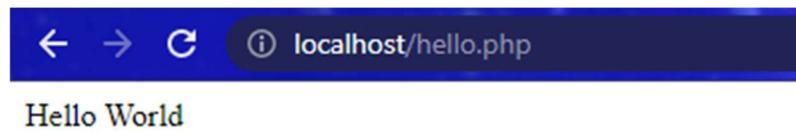


Abbildung 127: PHP-Script ausgeführt

4.3.5.4 HTML

HTML (Hypertext Markup Language) ist nicht wie PHP oder Python eine Programmiersprache, sondern eine textbasierte Auszeichnungssprache zum Strukturieren von Websites. Ein Browser liest den Code aus dem HTML File und stellt diesen grafisch dar. Die Sprache arbeitet mittels Tags. Eine Struktur wird immer mit den spitzen Klammern eingeleitet und durch einen Schrägstrich inmitten der Klammern beendet. [vgl. [42]]

```
<!DOCTYPE html>
<h1> Hello World </h1>
</html>
```

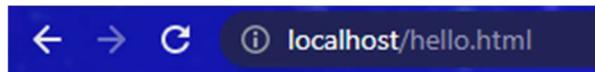
Abbildung 129: HTML-Beispiel



Abbildung 128: HTML-Logo [vgl. [53]]

Die erste Zeile „`<!DOCTYPE html>`“ definiert den Typ des Files. Tags wie „`<h1>`“ bestimmen den Typ der Struktur. Alles was zwischen den Tags steht ist der Inhalt, der strukturiert werden soll. Bei einer Überschrift wäre das dann der Titel. H1 ist die

Heading 1, also die Überschriftenstruktur mit der Nummer eins. Eine genaue Auflistung der verschiedenen Tags ist unter https://de.ryte.com/wiki/HTML_Tags zu finden.



Hello World

Abbildung 130: Test HTML-Page

Klarerweise lässt sich auch das HTML File im Browser öffnen.

4.3.5.5 CSS

Den Großteil der Gestaltung übernimmt nicht HTML selbst, sondern CSS (Cascading Style Sheets). Bei CSS handelt es sich um eine Gestaltungs- und Formatierungssprache. Format, Farbe, Größe, Abstand und Schriftart sind nur einige Beispiele für Dinge, die mit CSS angepasst werden können.

In sogenannten Stylesheets können Formatierungen für HTML Tags festgelegt werden. Diese sind entweder in derselben Datei gespeichert oder werden in einer externen Datei festgehalten, welche dann am Anfang des HTML Files eingebunden werden muss. [vgl. [43]]



Abbildung 131: CSS-Logo

```
h1{
    background-color: lightblue;
    text-align: center;
}
```

Abbildung 132: Beispiel CSS-Formatvorlage

```
<!DOCTYPE html>

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

<h1> Hello World </h1>
</html>
```

Abbildung 133: Beispiel HTML Code mit eingebundenem Stylesheet



Abbildung 134: Beispiel Ansicht HTML mit CSS

CSS-Frameworks sind Bibliotheken mit einer großen Anzahl an Klassen die vordefinierten Formatierungen enthalten. Dies eignet sich besonders für kleinere Projekte, da es nicht notwendig ist eigene Stylesheets zu erstellen.

4.3.5.6 HTML und PHP

Die beiden Sprachen interagieren stark und sind nicht umsonst ein bekanntes Duo zum Erstellen von dynamischen Webseiten. Um PHP-Code in einem HTML Projekt zu verwenden, muss lediglich der Abschnitt mittels den obligatorischen PHP-Klammern eingeleitet werden (siehe 4.3.5.3). Auch in PHP ist es möglich HTML Code einzubetten.

```
<!DOCTYPE html>
<html>
<?php $numb = 1+2; ?>
<h3><?php Echo "" . $numb; ?></h3>
</html>
```

Abbildung 135: Beispiel für PHP-Code in einem HTML Dokument

```
<?php
echo "<h2 align= center>Ich bin ein HTML Inhalt.</h2>"
```

Abbildung 136: Beispiel für HTML Code der in einem PHP-Dokument eingebettet ist.

4.3.5.7 Kommunikation mit der Datenbank

Um in PHP mit einer Datenbank kommunizieren zu können, wird PDO (PHP Data Objects) herangezogen. Es ist eine Library, die einen einheitlichen Zugriff auf SQL-Datenbanken ermöglicht. Die Verwendung wird im folgenden Beispiel erklärt.

```
// connecting the database
$conn = new PDO('sqlite:../driverslog.db');

//Setting connection attributes
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Abbildung 137 Codeausschnitt von der Verbindung zur Datenbank

Im ersten Schritt wird mittels „new PDO“ eine neue Verbindung erstellt. In der Klammer wird zum einen die Art der Datenbank angegeben (hier SQLite) und zum anderen der Pfad wo die Datenbank liegt.

Als nächstes werden die Attribute für die Verbindung gesetzt. Grundsätzlich ist die Verbindung somit fertig eingerichtet.

4.3.5.8 Query

Ist die Verbindung mit der Datenbank hergestellt, kann ein sogenanntes „query“ formuliert werden. In dem nachfolgenden Beispiel ist das ein SQL-Statement, um die Tabelle „member“ zu erstellen. Um das Query auszuführen, wird es mittels dem oben erstellten „\$conn“ ausgeführt (siehe Abbildung 137).

```

/*Query for creating reating the member
table in the database if not exist yet.*/
$query = "CREATE TABLE IF NOT EXISTS member (
    mem_id      INTEGER,
    username    TEXT,
    password    TEXT,
    firstname   TEXT,
    lastname    TEXT,
    birthdate   NUMERIC,
    RFID        INTEGER,
    PRIMARY KEY(mem_id))";

//Executing the query
$conn->exec($query);

```

Abbildung 138: Beispiel für das Ausführen eines Querys in PHP

Das Statement in Abbildung 138 zeigt, wie die Membertabelle mit SQL Statements erstellt wird. Für driverslog wurde die Tabelle via SQLite Browser erstellt (siehe Abschnitt 4.3.4.7).

4.3.6 Benutzerverwaltung

Da SQLite wie oben im Vergleich bereits erwähnt, keine Benutzerverwaltung besitzt, ist es notwendig, diese selbst zu implementieren. Um dies sinnvoll implementieren zu können, wurden die Aufgabenbereiche in fünf PHP-Skripts aufgeteilt.

conn.php

Ist für die Verbindung zur Datenbank zuständig. Da bei der Benutzerverwaltung öfter auf die Datenbank zugegriffen werden muss, wird dieser einmal in ein eigenes Skript implementiert, um dann in den anderen Files eingebunden zu werden. Ein Beispiel für einen Datenzugriff ist unter Abbildung 137 Codeausschnitt von der Verbindung zur Datenbank Abbildung 137 zu sehen. Ein Beispiel für einen Datenzugriff ist unter Abbildung 137 zu sehen.

login.php

Dieses Skript beinhaltet das Login Formular welches überwiegend in HTML eingebettet ist. Im folgenden Abschnitt ist das der Aufbau des Login Fensters zu sehen. In der ersten Zeile des Codes wird die Methode „POST“ festgelegt. Dies ist notwendig um die Eingaben, in diesem Fall Username und Password, als Session-Variablen an andere Files weiterzugeben. Unter „action“ stehen die Zieldateien, an welche die Variablen weitergegeben werden sollen.

```

<!-- Login Form Starts -->
<form method="POST" action="login_query.php", action = "home.php">
    <div class="alert alert-info " align="50">Login</div>
    <div class="form-group">
        <label>Username</label>
        <input type="text" name="username"
               class="form-control" required="required"/>
    </div>
    <div class="form-group">
        <label>Password</label>
        <input type="password" name="password"
               class="form-control" required="required"/>
    </div>
    <?php
        /*checking if the session 'error' is set.
        Error session is the message if the
        'Username' and 'Password' is not valid.*/
        if(ISSET($_SESSION['error'])){
    ?>
        <!-- Display Login Error message -->
        <div class="alert alert-danger">
            <?php echo $_SESSION['error']?></div>
    <?php
        //Unsetting the 'error' session after displaying the message.
        session_unset(); //unset doesn't want statements
    }
    ?>
    <button class="btn btn-primary btn-block" name="login">
        <span class="glyphicon glyphicon-log-in"></span> Login</button>
    </form>
<!-- Login Form Ends -->
```

Abbildung 139 Codeausschnitt login.php

login_query.php

Wie in Abbildung 140 zu sehen, werden die Logindaten an das File login_query.php weitergeleitet. Hier werden der Benutzername und das Passwort mit der Datenbank abgeglichen. Stimmen beide Parameter überein, wird man auf home.php weitergeleitet. Andernfalls wird eine Fehlermeldung ausgegeben.

```
<?php
    session_start();
    require_once 'conn.php'; //for connection to the DB

    if(ISSET($_POST['login'])){ //checking if variables passed from the Session
        $username = $_POST['username'];
        $password = $_POST['password'];
        $_SESSION['un'] = $username;
        //$_SESSION['pd'] = $password;

        $query = "SELECT COUNT(*) as count FROM `member`"
        WHERE `username` = :username
        AND `password` = :password";
        $stmt = $conn->prepare($query);
        $stmt->bindParam(':username', $username);
        $stmt->bindParam(':password', $password);
        $stmt->execute();
        $row = $stmt->fetch();

        $count = $row['count'];

        if($count > 0){
            header('location:home.php');
        }else{
            $_SESSION['error'] = "Invalid username or password";
            header('location:login.php');
        }
    }
?>
```

Abbildung 140: Codeausschnitt login_query.php

index.php

Für Personen, die noch keinen Account haben wird, die Möglichkeit geboten einen zu erstellen. Dafür gibt es ein eigenes Formular. Prinzipiell funktioniert das Äquivalent zum Login Formular (siehe Abbildung 139).

save_member.php

Diese Datei ist das Gegenstück zu index.php. Es speichert die Nutzerdaten des neuen Users in die Datenbank. Der Aufbau des Querys ist dabei ganz ähnlich dem Aufbau in Abbildung 140. Außerdem wird geprüft, ob der Username schon verwendet wurde, um zu verhindern, dass es zwei Konten gibt, die denselben Benutzernamen haben. Dafür wird zuerst eine Abfrage gestellt, in welcher der gewünschte Name mit der gesamten Datenbank abgeglichen wird. Erst dann werden die neuen Daten gespeichert.

home.php

Nach einem erfolgreichen Login Vorgang landet man auf der der Home Seite. Auf dieser wird der Fahrtenfortschritt dargestellt. Außerdem gibt es hier die Möglichkeit sein Fahrtenprotokoll in PDF-Form zu exportieren.

Für den Exportvorgang wird die Open Source Klasse TCPDF in der Version 6.4.4 verwendet [vgl. [44]].

Ihr Fahrtenbuch

Abmelden

Wilkommen Herbert!

Nr.	Datum	Uhrzeit	KM Start	KM Stop	KM ges.	Route	Nummernschild	Wetter
1	21.12.2021	16:06	0	450	450	München-Zürich	VB321C	Regen
2	22.12.2021	18:00	450	475	25	Mondsee-Salzburg	VB321C	Sonne
3	23.12.2021	14:10	475	489	14	Unterach-Mondsee	VB321S	Regen
4	31.12.2021	04:13	489	796	7	Buchenort-Unterach	VB321S	Schnee
5	02.01.2022	15:04	796	896	100	Straßwalchen-Linz	VB321C	Sturm
6	21.01.2022	15:56	896	950	54	Unterach-Bad Ischl	VB321C	Schnee
7	25.01.2022	19:13	950	957	7	Unterach-Nußdorf	VB321C	Sonne

GENERATE PDF

Abbildung 141: Überblick des Fahrtenfortschritts (home.php)

4.3.7 REST API

REST steht für Representational State Transfer und API steht für Applikation Programming Interface. Das Akronym meint damit eine Programmierschnittstelle, die sich an die Gesetze des World Wide Web hält und einen Ansatz für die Kommunikation zwischen Client und Server bietet. Diese Architekturen bedienen sich meist an

standardisierten Verfahren wie HTTP/S, URI oder JSON.

Über eine REST API ist es möglich die HTTP-Requests zu koordinieren und die Abfragen an die Datenbank zu stellen. Somit kann von der App aus in die Datenbank geschrieben werden oder eben auch Daten von ihr abgefragt werden (siehe 4.2.12).

In unserem Fall wurde die arrestDB in der Version 1.0 verwendet [vgl. [45]]. Sie folgt dem Schema der RESTful APIs und ist speziell für SQLite Datenbanken geeignet. Die API besteht aus nur einer Datei und die Datenbankverbindung wird mit PDO realisiert. Auch beim Fehlercode orientiert sich die arrestDB an den HTTP Standards. [vgl. [46]] Ein Beispiel für eine Anfrage an die REST API ist unter 4.2.12 zu finden.

5 Ergebnisse – Abnahme

Der aktuelle Projektstand besteht aus einer Verbindungsplatine, welche die verschiedenen Hardware-Komponenten miteinander verbindet. Die Bluetooth-Verbindung zwischen der Hardware und der App funktioniert. Das L-17 Symbol wird angezeigt. Die Fahrdaten werden erhoben und in der Datenbank gespeichert. Das Fahrtenprotokoll kann auf der Website exportiert und ausgedruckt werden.

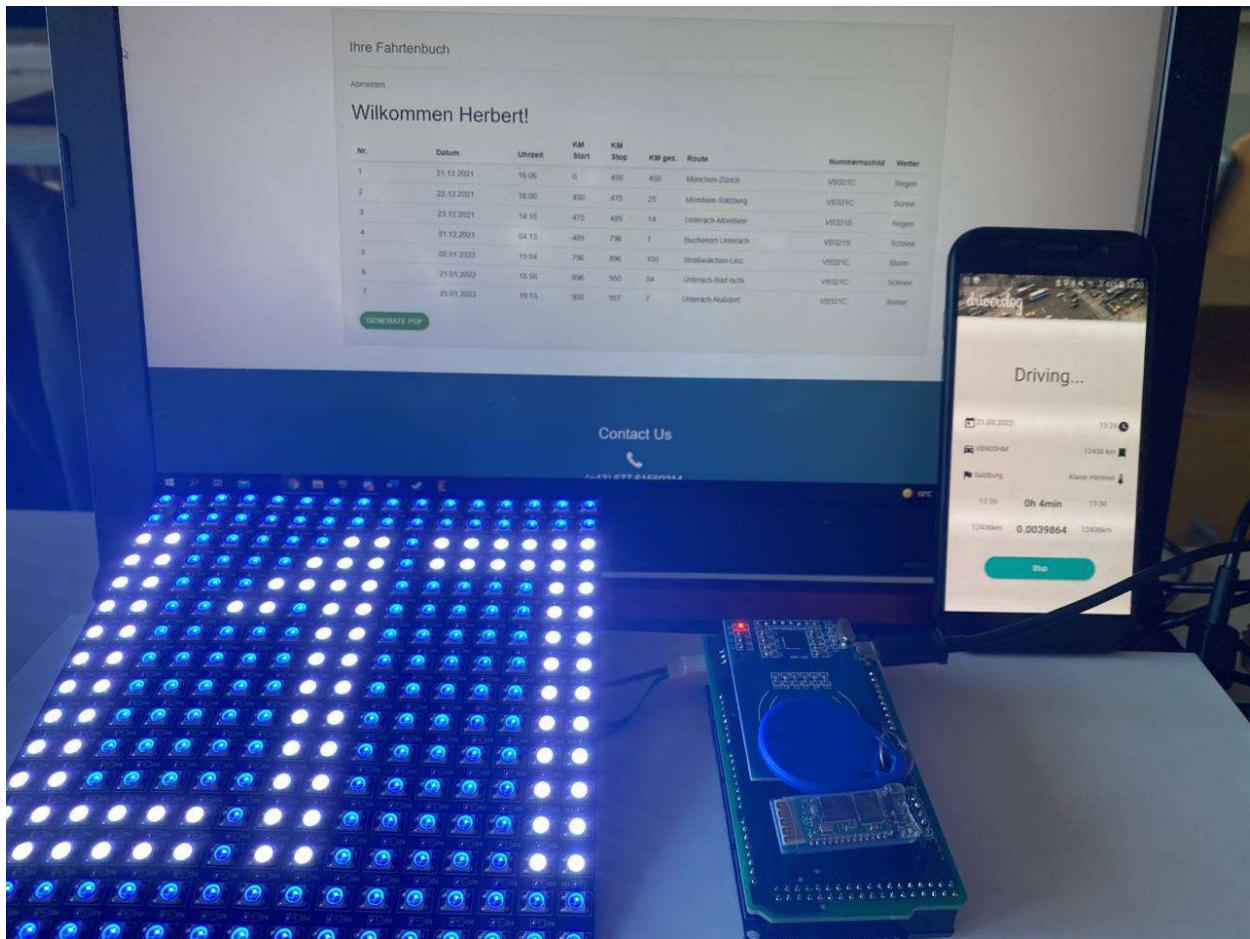


Abbildung 142: Ansicht während einer Fahrt (L17-Symbol, Verifikationsmodul, Handy-App) + Website-Ansicht

In folgender Übersicht werden die Ergebnisse der Testfälle, aus dem Lastenheft, veranschaulicht:

Anforderung	Testergebnis	Freigabe (J/N)
Bluetooth-Datentransfer zwischen Hardware und App. (F0010)	Die Verbindung wird hergestellt und der Datentransfer funktioniert bidirektional. (T0010)	J
L-17 soll auf Anforderung an der LED-Matrix angezeigt werden. (F0020)	Bei entsprechendem Bluetooth-Empfang wird auf der Matrix das Symbol angezeigt. (T0020)	J
Während der Fahrt sollen relevante Daten zur Fahrt ermittelt werden. (F0030)	Während der Fahrt werden die Fahrdaten ermittelt. (T0030)	J
Die Fahrdaten sollen in eine externe Datenbank gespeichert werden, um beim nächsten Mal wieder aufgerufen zu werden. (F0040)	Fahrdaten werden in der Datenbank abgespeichert und können jederzeit abgerufen werden. (T0040)	J
Alle Fahreinträge und Termine sollen in der App und auf der Website angezeigt werden. (F0050)	Persönlichen Daten werden in der App und auf der Website angezeigt. (T0050)	J
Das ausgefüllte Formular soll auf der Website, im PDF-Format, exportierbar sein. (F0060)	Das PDF-Dokument kann auf der Website exportiert werden. (T0060)	J

Tabelle 25: Testfälle und Ergebnisse

6 Literaturverzeichnis

- [1] E. Kompendium. [Online]. Available: <https://www.elektronik-kompendium.de/sites/com/1904111.htm>. [Zugriff am 22 3 2022].
- [2] O. Weis. [Online]. Available: <https://www.serial-port-monitor.org/de/articles/serial-communication/>. [Zugriff am 22 03 2022].
- [3] S. AG. [Online]. Available: <https://www.samsongroup.com/document/l153de.pdf>. [Zugriff am 22 03 2022].
- [4] Edi. [Online]. Available: <https://edistechlab.com/wie-funktioniert-uart/?v=fa868488740a>. [Zugriff am 22 3 2022].
- [5] JIMBLOM. [Online]. Available: <https://learn.sparkfun.com/tutorials/serial-communication/all>. [Zugriff am 22 03 2022].
- [6] R. Rahner. [Online]. Available: <https://www.rahner-edu.de/grundlagen/signale richtig-verstehen/xbee-api-und-at-befehl/>. [Zugriff am 22 03 2022].
- [7] ElabPeter. [Online]. Available: [https://robotfreak.de/elab-wiki/index.php?title=Serielle_Schnittstelle_\(UART\)](https://robotfreak.de/elab-wiki/index.php?title=Serielle_Schnittstelle_(UART)). [Zugriff am 22 3 2022].
- [8] T. Mathias. [Online]. Available:
https://en.wikipedia.org/wiki/I%C2%B2C#/media/File:I2C_controller-target.svg.
[Zugriff am 3 4 2022].
- [9] E. Fernandez. [Online]. Available: <http://fmh-studios.de/theorie/informationstechnik/i2c-bus/>. [Zugriff am 22 3 2022].
- [10] P. Semiconductors. [Online]. Available: https://www.i2c-bus.org/fileadmin/ftp/i2c_bus_specification_1995.pdf. [Zugriff am 03 04 2022].
- [11] Wikipedia. [Online]. Available:
https://en.wikipedia.org/wiki/Serial_Peripheral_Interface#Clock_polarity_and_phase. [Zugriff am 22 3 2022].
- [12] Edi. [Online]. Available: <https://edistechlab.com/wie-funktioniert-spi/?v=fa868488740a>. [Zugriff am 22 3 2022].
- [13] N. Semiconductors. [Online]. Available: <https://www.nxp.com/docs/en/datasheet/MFRC522.pdf>. [Zugriff am 2 4 2022].
- [14] M. Balboa. [Online]. Available: <https://github.com/miguelbalboa/rfid>. [Zugriff am 2 4 2022].

- [15] T. Dresden. [Online]. Available: https://tu-dresden.de/ing/informatik/ti/vlsi/ressourcen/dateien/dateien_studium/dateien_lehstuhsseminar/vortraege_lehrstuhlseminar/hs_ws_0506/falk_niederlein.pdf?lang=en. [Zugriff am 22 3 2022].
- [16] ScienceProg. [Online]. Available: <https://scienceprog.com/understanding-1-wire-interface/>. [Zugriff am 22 3 2022].
- [17] Wikipedia. [Online]. Available: <https://de.wikipedia.org/wiki/1-Wire#Timing>. [Zugriff am 22 3 2022].
- [18] M. I. Products. [Online]. Available: <https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1796.html>. [Zugriff am 2 4 2022].
- [19] I. Solberg. [Online]. Available: <https://docplayer.org/13128648-Am-langen-draht-der-1-wire-bus.html>. [Zugriff am 22 3 2022].
- [20] ionos. [Online]. Available: <https://www.ionos.de/digitalguide/websites/web-entwicklung/was-ist-flutter/>. [Zugriff am 8 3 2022].
- [21] G. LLC. [Online]. Available: <https://docs.flutter.dev/get-started/install/windows>. [Zugriff am 22 3 2022].
- [22] G. LLC. [Online]. Available: <https://docs.flutter.dev/get-started/test-drive>. [Zugriff am 22 3 2022].
- [23] G. LLC. [Online]. Available: <https://docs.flutter.dev/get-started/codelab>. [Zugriff am 22 3 2022].
- [24] G. LLC. [Online]. Available: <https://docs.flutter.dev/development/ui/widgets-intro>. [Zugriff am 22 3 2022].
- [25] fluttercommunity.dev. [Online]. Available: https://pub.dev/packages/flutter_launcher_icons. [Zugriff am 22 3 2022].
- [26] ionos. [Online]. Available: <https://www.ionos.de/digitalguide/server/knowhow/bluetooth/>. [Zugriff am 22 3 2022].
- [27] E. Kompendium. [Online]. Available: <https://www.elektronik-kompendium.de/sites/kom/2107121.htm>. [Zugriff am 22 3 2022].

- [28] I. Studio. [Online]. Available:
https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf. [Zugriff am 22 3 2022].
- [29] pub.dev. [Online]. Available: https://pub.dev/packages/flutter_bluetooth_serial.
[Zugriff am 22 3 2022].
- [30] G. LLC. [Online]. Available:
<https://developer.android.com/guide/topics/connectivity/bluetooth/permissions>.
[Zugriff am 22 3 2022].
- [31] bernos.dev. [Online]. Available: <https://pub.dev/packages/location>. [Zugriff am 22 3 2022].
- [32] S. Biswas. [Online]. Available: <https://ichi.pro/de/fluttern-erstellen-eines-routenrechners-mit-google-maps-124698433777593>. [Zugriff am 22 3 2022].
- [33] baseflow.com. [Online]. Available: <https://pub.dev/packages/geocoding>. [Zugriff am 22 3 2022].
- [34] cachet.dk. [Online]. Available: <https://pub.dev/packages/weather>. [Zugriff am 22 3 2022].
- [35] E. S., „Hostinger,“ [Online]. Available: <https://www.hostinger.com/tutorials/sqlite-vs-mysql-whats-the-difference/>. [Zugriff am 9 3 2022].
- [36] R. Schragl, *Einführung DB*, 2022.
- [37] R. Schragl, *Normalformen DB*.
- [38] R. Schragl, *ERM Modellierung*.
- [39] „affde,“ [Online]. Available: <https://www.affde.com/de/client-side-rendering-vs-server-side-rendering.html>. [Zugriff am 31 3 2022].
- [40] „PHP.net,“ [Online]. Available: <https://www.php.net/manual/de/intro-whatis.php>.
[Zugriff am 9 3 2022].
- [41] S. Tondon, „Medium,“ [Online]. Available: <https://medium.com/predict/python-vs-php-vs-javascript>. [Zugriff am 10 3 2022].
- [42] „Schulhomepage,“ [Online]. Available:
<https://www.schulhomepage.de/webdesign/html>. [Zugriff am 11 3 2022].
- [43] „checkdomain,“ [Online]. Available:
<https://www.checkdomain.de/hosting/lexikon/css/>. [Zugriff am 11 3 2022].
- [44] TCPDF, „TCPDF,“ [Online]. Available: <https://tcpdf.org/>. [Zugriff am 29 03 2022].

- [45] alixaxel, „github,“ [Online]. Available: <https://github.com/alixaxel/ArrestDB>. [Zugriff am 29 03 2022].
- [46] M. D. S. / F. Karlstetter, „cloudcomputing,“ [Online]. Available: <https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/>. [Zugriff am 12 3 2022].
- [47] K. ELECTRONIC. [Online]. Available: <https://web.archive.org/web/20030727145801/http://www.pci-card.com/schnittstellen.html>. [Zugriff am 22 03 2022].
- [48] A. Pandit. [Online]. Available: <https://circuitdigest.com/tutorial/serial-communication-protocols>. [Zugriff am 22 3 22].
- [49] Swaroop. [Online]. Available: <https://www.codrey.com/embedded-systems/serial-communication-basics/>. [Zugriff am 22 3 2022].
- [50] „Wikipedia,“ [Online]. Available: <https://de.wikipedia.org/wiki/SQLite>. [Zugriff am 22 3 2022].
- [51] „Wikipedia,“ [Online]. Available: https://de.wikipedia.org/wiki/Datei:Xampp_logo.svg. [Zugriff am 9 3 2022].
- [52] „Wikipedia,“ [Online]. Available: <https://de.wikipedia.org/wiki/JavaScript>. [Zugriff am 22 3 2022].
- [53] „Moorhuhn,“ [Online]. Available: [https://moorhuhn.fandom.com/de/wiki/Moorhuhn_\(HTML_5\)](https://moorhuhn.fandom.com/de/wiki/Moorhuhn_(HTML_5)). [Zugriff am 22 3 2022].
- [54] „Wikipedia,“ [Online]. Available: <https://de.wikipedia.org/wiki/PHP>. [Zugriff am 22 3 2022].
- [55] „Wikipedia,“ [Online]. Available: [https://de.wikipedia.org/wiki/Python_\(Programmiersprache\)](https://de.wikipedia.org/wiki/Python_(Programmiersprache)). [Zugriff am 22 3 2022].

7 Verzeichnisse der Abkürzungen, Abbildungen und Tabellen

7.1 Abkürzungen

μ C	Mikrocontroller
ACK	acknowledge
BLE	Bluetooth Low Energy
CLK	Clock
CSS	Cascading Style Sheets
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
DDL	Data Definition Language
DML	Data Manipulation Language
ERM	Entity Relationship Modell
FTP	File Transfer Protocol
GND	Ground
GPS	Global Positioning System
GUI	Grafical User Interface
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
I ² C	Inter-Integrated Circuit
ID	Identification
JS	JavaScript
LSB	Least Significant Bit
MISO	Master Input, Slave Output
MOSI	Master Output, Slave Input
MSB	Most Significant Bit
NF	Normalform
PDO	PHP Data Objects
RDBMS	Relational Database Management System
RFID	Radio Frequency Identification
SCK	Serial Clock

SDK	Software Development Kit
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SS	Slave Select
UART	Universal Asynchronous Receiver / Transmitter
UI	User Interface
UID	User-Identification
VCC	Voltage at the common collector

7.2 Abbildungen

Abbildung 1: Prototyp	5
Abbildung 2: Prototype	8
Abbildung 3: Grobentwurf des Projekts, in der Mitte das µC-System	15
Abbildung 4: Login-Screen	21
Abbildung 5: Sign-up-Screen.....	21
Abbildung 6: Home-Screen.....	22
Abbildung 7: Routes-Screen.....	22
Abbildung 8: Calendar-Screen.....	22
Abbildung 9: Account-Screen	22
Abbildung 10: Bluetooth verbunden.....	23
Abbildung 11: erfolgreiche Verifikation	23
Abbildung 12: fehlgeschlagene Verifikation	23
Abbildung 13: neue Fahrt: Eingabefenster	24
Abbildung 14: Live-Fahrdatenansicht	24
Abbildung 15: Webapplikation Startbildschirm.....	25
Abbildung 16: Registrieren: neuen Account erstellen	25
Abbildung 17: Formular zum Registrieren	25
Abbildung 18: Login Formular Website.....	26
Abbildung 19: Home Seite mit Darstellung des Fahrtenfortschritts.....	26
Abbildung 20: GANT-Diagramm Niklas Parhammer.....	31
Abbildung 21: GANT-Diagramm Philip Wienerroither.....	32
Abbildung 22: GANT-Diagramm Daniel Wipplinger	33
Abbildung 23: Aufgabenbereiche der einzelnen Teammitglieder sind eingefärbt	34
Abbildung 24: Paralleles Interface [vgl. [2]]	35

Abbildung 25: Innere Leitungen sind kürzer als Äußere [vgl. [1]].....	36
Abbildung 26: Mäanderförmige Leitungsführung zwischen zwei Controllern [vgl. [1]] ...	36
Abbildung 27: IEEE 1284 Druckerschnittstelle	37
Abbildung 28:Serielle Übertragung, Bit für Bit [vgl. [2]]	37
Abbildung 29: asynchrone serielle Übertragung eines Bytes, sowie benötigter Synchronisationsinformationen und ungerader Parität [vgl. [6]].....	38
Abbildung 30: UART Interface. RX und TX sind gekreuzt verdrahtet [vgl. [7]].....	40
Abbildung 31: Beispielhafter Datensatz einer UART Übertragung [vgl. [4]].....	40
Abbildung 32: HC-05 Bluetooth-Modul	41
Abbildung 33: Code zum Inkludieren der SoftwareSerial Bibliothek	42
Abbildung 34: Code zum Festlegen von Rx und Tx auf die Pins 10 und 49	42
Abbildung 35: Code zum Einstellen der Baudrate auf 9600 bps.....	42
Abbildung 36: Code zum Senden des Buchstabens „f“	42
Abbildung 37: Code zum Einlesen von empfangenen Daten.....	42
Abbildung 38: Möglicher Aufbau eines I ² C Interfaces mit einem Master und drei Slaves [vgl. [8]].....	43
Abbildung 39: Aufbau einer I ² C Datenübertragung [vgl. [9]]	44
Abbildung 40: Timing Diagramm einer I ² C Übertragung [vgl. [9]]	45
Abbildung 41: Adressbyte [vgl. [10]]	45
Abbildung 42: Arbitrationsprozess mit zwei Master [vgl. [10]].....	46
Abbildung 43: SPI Interface mit 3 Slaves [11].....	47
Abbildung 44: Slave wird mittels Slave select aktiviert [vgl. [11]].....	47
Abbildung 45: 4 verschiedene Modis zur Synchronisation [vgl. [12]]	48
Abbildung 46: Timing Diagramm einer SPI Übertragung [vgl. [12]]	49
Abbildung 47: RC522 RFID Modul	49
Abbildung 48: RFID Schlüsselanhänger und Karte	49
Abbildung 49: RFID-Tag kann ab einem Abstand von 5cm zum Modul beschrieben/gelesen wird	50
Abbildung 50: Code zum inkludieren der SPI und MFRC522 Bibliothek	51
Abbildung 51: Code zum Definieren von RST und SS Pin	51
Abbildung 52:Code zum Initialisieren von SPI Interface und MFRC522 Chip	51
Abbildung 53: Code zum Überprüfen, ob ein RFID-Tag verfügbar ist	51
Abbildung 54: Code zum Auslesen der UID	51

Abbildung 55: Code, der die UID in die gewünschte Form konvertiert und anschließend mit der geforderten UID vergleicht.....	52
Abbildung 56: 1-Wire Interface mit 4,7k Ω Pull-up-Widerstand [vgl. [16]]	52
Abbildung 57: Innerer Aufbau eines 1-Wire Slaves mit interner Kapazität [vgl. [18]]	53
Abbildung 58: Timing Diagramm um 0x33 zusenden [vgl. [17]].....	54
Abbildung 59: Timingdiagramm um 0x01 zusenden.....	54
Abbildung 60: Verwendete LED-Matrix mit 256 LEDs	54
Abbildung 61: Serpentinenartiger Verlauf des LED-Streifens	55
Abbildung 62: Pinbelegung der LED-Matrix.....	55
Abbildung 63: Code zum inkludieren der „Adafruit_NeoPixel“ Bibliothek	56
Abbildung 64: Code zum Erstellen der Instanz „pixels“	56
Abbildung 65: Code zum Verändern der Farbe einer bestimmten LED	56
Abbildung 66: Bestimmen der benötigten Werte mit dem Color Picker Werkzeug in Paint.net	57
Abbildung 67: Code zum Einfärben der ersten LED mit dem L17-Blau	57
Abbildung 68: Messschaltung	57
Abbildung 69: Leistungsmessung der LED-Matrix.....	57
Abbildung 70: Vor- und Nachteile von Flutter	58
Abbildung 71: App testen [vgl. [22]]	60
Abbildung 72: pubspec.yaml dependencies	61
Abbildung 73: Library-Einbindung.....	61
Abbildung 74: Code zur Anzeige von Text	62
Abbildung 75: Text-Widget-Anzeige	62
Abbildung 77: Code zur Anzeige eines Containers	62
Abbildung 76: Container-Widget-Anzeige.....	62
Abbildung 78: Code zur Zentrierung von Widgets	63
Abbildung 79: Anzeige Zentrieren von Widgets.....	63
Abbildung 80: Anzeige eines Buttons	63
Abbildung 81: Code zur Verwendung von Buttons	63
Abbildung 82: Anzeige von Widgets nebeneinander	64
Abbildung 83: Anzeige von Widgets untereinander	64
Abbildung 84: Code zur vertikalen Anordnung von Widgets.....	64
Abbildung 85: Code zur horizontalen Anordnung von Widgets.....	64
Abbildung 86: foreground.png	65

Abbildung 87: background.png	65
Abbildung 88: appicon.png	65
Abbildung 89: App-Icon Pfadeinbindung	65
Abbildung 90: fertiges Logo am Home-Screen	65
Abbildung 91: Login-Funktionsaufruf	67
Abbildung 92: Sign up-Funktionsaufruf	67
Abbildung 93: Daten in den Gerätespeicher speichern	68
Abbildung 94: Daten aus dem Gerätespeicher laden	68
Abbildung 95: Bluetooth Berechtigungen in "AndroidManifest.xml"	70
Abbildung 96: globale Variable zum Speichern der Verbindungsdaten	70
Abbildung 97: Bluetooth-Status Variable	71
Abbildung 98: initState() Check Bluetooth Status	71
Abbildung 99: Bluetooth-Verbindungsaufbau	72
Abbildung 100: Bluetooth Aktivierungsanfrage-Fenster	72
Abbildung 101: Funktion zum Senden von Bluetooth-Daten	73
Abbildung 102: Funktion zum Empfangen von Bluetooth-Daten	73
Abbildung 103: Fahrtenprotokoll	74
Abbildung 104: Parameter der Trip-Klasse	74
Abbildung 105: Start-Screen einer Fahrt	74
Abbildung 106: aufgerufene Funktion beim Start einer Fahrt	75
Abbildung 107: Verwendung der geocoding-Library	76
Abbildung 108: Verwendung der weather-Library	76
Abbildung 109: GET-Request	77
Abbildung 110: POST-Request	77
Abbildung 111 SQLite Logo [vgl. [50]]	79
Abbildung 112: Datenbank erstellen mit SQLite Browser	80
Abbildung 113: Beziehungen 1:1	84
Abbildung 114: Beziehungen 1:n	84
Abbildung 115: Beziehungen n:m	85
Abbildung 116: Driverslog Beziehungen/ ERM	85
Abbildung 117: Erstellen einer Tabelle mittels SQLite Browser	86
Abbildung 118: Beispiel DDL	87
Abbildung 119 Beispiel DML	87
Abbildung 120: Grafik CSR Client-Side-Rendering [vgl. [39]]	88

Abbildung 121: Grafik SSR Server-Side-Rendering [vgl. [39]].....	89
Abbildung 122: XAMPP Logo [vgl. [51]].....	89
Abbildung 123: PHP-Logo [vgl. [54]].....	90
Abbildung 124: Java Script Logo [vgl. [52]]	90
Abbildung 125: Python-Logo [vgl. [55]]	91
Abbildung 126: PHP-Beispiel.....	92
Abbildung 127: PHP-Script ausgeführt	92
Abbildung 128: HTML-Logo [vgl. [53]]	92
Abbildung 129: HTML-Beispiel	92
Abbildung 130: Test HTML-Page	93
Abbildung 131: CSS-Logo	93
Abbildung 132: Beispiel CSS-Formatvorlage.....	93
Abbildung 133: Beispiel HTML Code mit eingebundenem Stylesheet.....	93
Abbildung 134: Beispiel Ansicht HTML mit CSS.....	93
Abbildung 135: Beispiel für PHP-Code in einem HTML Dokument	94
Abbildung 136: Beispiel für HTML Code der in einem PHP-Dokument eingebettet ist. .	94
Abbildung 137 Codeausschnitt von der Verbindung zur Datenbank.....	94
Abbildung 138: Beispiel für das Ausführen eines Querys in PHP.....	95
Abbildung 139 Codeausschnitt login.php	96
Abbildung 140: Codeausschnitt login_query.php.....	97
Abbildung 141: Überblick des Fahrtenfortschritts (home.php)	98
Abbildung 142: Ansicht während einer Fahrt (L17-Symbol, Verifikationsmodul, Handy-App) + Website-Ansicht	100

7.3 Tabellen

Tabelle 1: Aufgabenverteilung	30
Tabelle 2: Übersicht über die Vor- und Nachteile von seriellen und parallelen Interfaces	39
Tabelle 3: Übliche Taktraten des UART-Interface	40
Tabelle 4: Pinbelegung des HC-05 im Data Mode.....	41
Tabelle 5: Pinbelegung für AT Command Mode (EN=GND)	41
Tabelle 6: Die verschiedenen Modi mit ihren Übertragungsraten	45
Tabelle 7: 4 verschiedene Modes zur Einstellung der Synchronisation.....	48
Tabelle 8: Pinbelegung des RC522 mit SPI-Interface	50

Tabelle 9: Bluetooth Versionen [vgl. [26]]	69
Tabelle 10: Bluetooth Berechtigungen.....	70
Tabelle 11: Relationale Datenbankmodel: Tabellenbeispiel	80
Tabelle 12: Foreign Key Beispiel Schüler.....	81
Tabelle 13: Foreign Key Beispiel Adresse	81
Tabelle 14: 0.Normalform	81
Tabelle 15: 1.Normalform	82
Tabelle 16: 2.Normalform Tabelle Benutzer	82
Tabelle 17: 2.Normalform Tabelle Fahrten	82
Tabelle 18: 2.Normalform Tabelle Benutzer	83
Tabelle 19: 2.Normalform Tabelle Fahrten	83
Tabelle 20: 3.Normalform Tabelle Mitglieder	83
Tabelle 21: 3.Normalform Tabelle Lieferung.....	83
Tabelle 22: 3.Normalform Tabelle Mitglieder	83
Tabelle 23: 3.Normalform Tabelle Adresse	83
Tabelle 24: 3.Normalform Tabelle Lieferung.....	83
Tabelle 25: Testfälle und Ergebnisse	101

8 Begleitprotokoll

8.1 Niklas Parhammer

KW	Beschreibung	Zeitaufwand
37	Projektfindung	10h
38	Funktionsliste, Entwurf Blockschaltbild, Namensfindung	10h
39	Projektantrag	10h
40	Projektantrag verbessern, Gantt-Diagramm	10h
41	Schaltungsentwurf	10h
42	Projektantrag finalisiert, Hardware-Bestellung	10h
43	Herbstferien	
44	RFID-Modul in Betrieb genommen	10h
45	Labor-Übung (1-Wire), RFID Authentifizierung	10h
46	RFID Authentifizierung mit Smartphone als Tag Recherche, 1Wire Protokoll	10h
47	Steuerung der LED-Matrix	10h
48	Leistungsmessung der LED-Matrix in unserem Fall und mit maximaler Auslastung	10h
49	RFID Sensor funktioniert nicht -> Problemfindung	10h
50	Lösung nicht gefunden, Um keine Zeit zu verschwenden auf Arduino MEGA gewechselt -> RFID-Sensor funktioniert	10h
51	Fotos des Projektteams erstellen und bearbeiten	10h
52	Ferien	
01	Ferien	
02	Ansteuerung des Bluetooth-Modul, RFID-Modul und LED-Matrix als auch Kommunikation mit der App mittels Programmierung des Arduino MEGA	10h
03	Präsentation vorbereiten (Powerpoint)	10h
04	Eagle Schaltplan, Layout und 3D Model Anfertigung	10h
05	Schaltplan und Layout fertiggestellt und FA verschickt	10h
06	Erste Version der Platine gelötet und getestet -> schlechte LED-Matrix Montierung, Fehlerhafte Pinbelegung	10h
07	Ferien	
08	Schaltplan und Layout anpassen (Steckverbinder hinzugefügt)	10h
09	Zweite Version der Platine gelötet und vollständig getestet	10h

KW ...Kalenderwoche

8.2 Philip Wienerroither

KW	Beschreibung	Zeitaufwand
37	Projektfindung	3h
38	Funktionsliste, Entwurf Blockschaltbild, Namensfindung	4h
39	Projektantrag, IDE Installation	10h
40	Gant-Diagramm, Projektantrag	10h
41	Bluetooth Modul (HC-05) Inbetriebnahme	5h
42	Projektantrag final absenden und Hardwarebestellung	5h
43	Ferien	
44	Einlesen in Flutter-Bluetooth	5h
45	Laborübung (1-Wire-interface)	5h
46	Bluetooth implementieren	10h
47	Bluetooth BLE-Fehler beheben, App-Design-Planung	10h
48	App-Design (Tab-Übersicht)	10h
49	App-Funktion (Navigation), Strukturierung	10h
50	Abstimmung BT Kommunikation (keys), BT-Keys ändern	10h
51	Umsetzung einer Testschaltung (BT - Kommunikation), Umsetzung einer Testdatenbank (DB – Kommunikation)	10h
52	Ferien	
01	Ferien	
02	Systemtest ohne DB, Layoutdesign während einer Fahrt,	10h
03	DB-Connection, Login System mit DB, Präsentationsentwurf	10h
04	App-Icon, Splash Screen, GPS-Data-Basics, Überarbeitung Präsentation	10h
05	Settings UI	10h
06	UI Überarbeitung	10h
07	Remember me-Funktion, Dark Mode	10h
08	Kalender implementieren	10h
09	Bug Fixes und UI-Änderungen	10h

KW ...Kalenderwoche

8.3 Daniel Wipplinger

KW	Beschreibung	Zeitaufwand
37	Projektfindung	3,5h
38	Funktionsliste, Entwurf Blockschaltbild, Namensfindung	4h
39	Projektantrag	5h
40	Überarbeiten von Projektantrag, individualisieren des Gantt-Diagramm	8h
41	Einarbeiten in das Thema Datenbanken	10h
42	Informieren über die diversen Programme und Funktionen von SQLite	6h
43	Adminer einrichten und Fehlersuche	10h
44	Einarbeiten in das SQLite	10h
45	Laborübung über OneWire/ CSV in SQLite Datenbank einführen	10h
46	CSV in SQLite Datenbank einführen/ Script, um neue Daten hochzuladen	10h
47	Login System für die DB einführen	10h
48	Arbeiten am Login System (User mit Zugehörigen Daten verknüpfen)	10h
49	Dynamische IP für Website/ Login System Username einmalig machen	10h
50	Username ist einmalig und soll via Session in andere Scripts übertragen werden	15h
51	Daten für den Benutzer via Username anzeigen	10h
52	Ferien	0h
01	Ferien	5h
02	Website verbessern/ Rest-API einrichten	10h
03	Präsentation vorbereiten/ Rest-API konfigurieren	10h
04	Design der Website Header	10h
05	Designprobleme: Logo –Behoben +neue Login Page	10h
06	Designproblem mit Home-Page behoben + einarbeiten in PDF	11h
07	PDF exportfähig gemacht und formatiert	14h
08	Fehlerbehebung bei der Website/ Anpassung der Datenbank	10h
09	Fehlerbehebung bei der Rest API	12h

KW ...Kalenderwoche