

MotoData
Ein System zur Erfassung
von Fahrdaten

Diplomarbeit 2022

Dipl. Techniker/in HF Informatik

TEKO Luzern
Industriestrasse 10
6010 Kriens

Auftraggeber / Projektleiter:

Pascal Rusca
Brunnenhöfli 18
6012 Obernau
p.rusca@hotmail.com

Projektleiter:

Janik Schilter
Chilenmattli 7
6055 Alpnach Dorf
janik.schilter@gmail.com

Betreuer:

Dozent TEKO Luzern
Andreas Holzer
andreas.holzer@edu.teko.ch

1 Management Summary

Mehr als ein Drittel der schweren Unfälle mit Motorrädern sind gemäss der Beratungsstelle für Unfallverhütung Alleinunfälle. Dabei stellt eine zu hohe Geschwindigkeit die häufigste Ursache dar. Die tragischen Folgen, welche nicht selten fatal enden, sind Kollisionen mit Hindernissen auf oder abseits der Strasse.

Beherrscht ein Motorradfahrer das sichere Kurvenfahren, so kann er auf potenzielle Hindernisse und unterschiedliche Strassenverhältnisse besser reagieren. Zudem fällt es dem Fahrer leichter, die gewünschte Linienwahl auf der Strasse umzusetzen.

Diese Diplomarbeit befasst sich mit der Analyse der Kurvenfahrt von Motorradfahrenden und ermöglicht eine praktische Lösung, nicht nur für Anfänger, sondern auch für erfahrene Fahrer.

MotoData ist ein System zur Erfassung von Fahrdaten wie Standort, Geschwindigkeit und Kurvenlage. Mittels Sensoren am oder im Motorrad werden Daten gesammelt, welche zu einem späteren Zeitpunkt via Web-Browser eingesehen werden können. Dabei stellt MotoData die erfassten Messpunkte auf einer topographischen Karte zur Verfügung. Somit wird den Motorradfahrenden ermöglicht, die eigenen Fahrkünste abseits von der Strasse zu analysieren und Rückschlüsse daraus zu ziehen.

Das entwickelte System überzeugt durch grosse Erweiterungsmöglichkeiten und geringem Kostenaufwand. Die Wünsche und Erwartungen der Projektbeteiligten wurden erfüllt.

Nach Fertigstellung der Projektarbeit waren beinahe alle Musskriterien vollumfänglich erfüllt. Alle Wunschkriterien konnten erfolgreich umgesetzt werden.

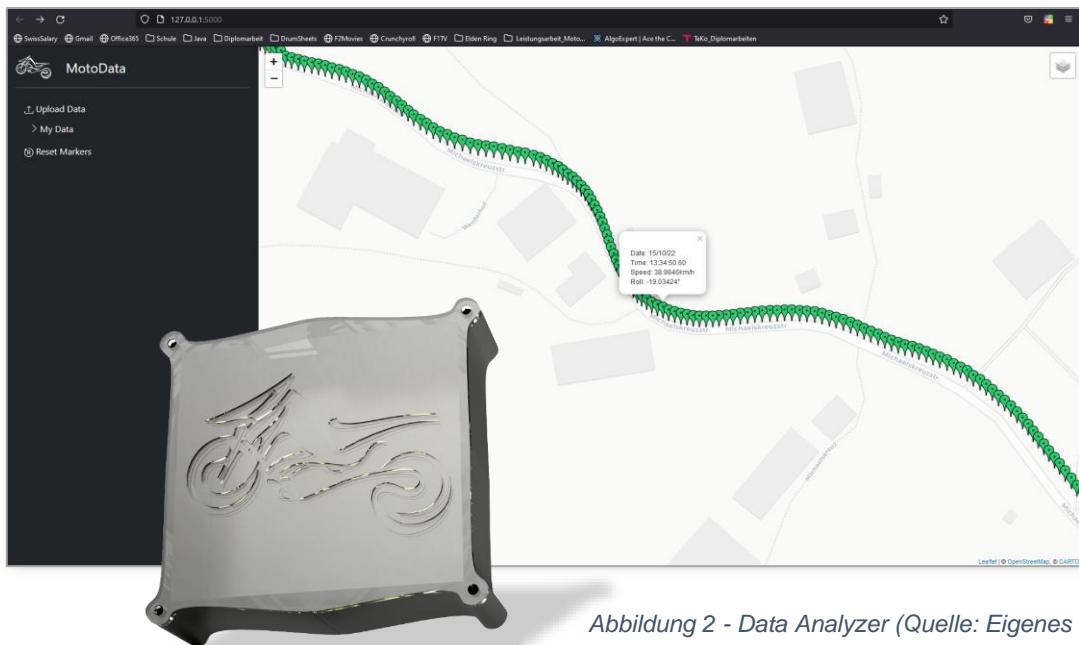


Abbildung 2 - Data Analyzer (Quelle: Eigenes Bild)

Abbildung 1 - Sensorsystem (Quelle: Eigenes Bild)

2 Inhalt

1	Management Summary	2
3	Abbildungsverzeichnis.....	6
4	Glossar	8
5	Einleitung.....	9
5.1	Vorwort	9
6	Projektmanagement	10
6.1	Projektorganisation	10
6.2	Die Projektbeteiligten.....	11
6.2.1	Pascal Rusca	11
6.2.2	Janik Schilter	12
6.3	Projektstrukturplan.....	13
6.4	Termin- und Zeitmanagement.....	14
6.4.1	Projektphasen	14
6.4.2	Meilensteine / Termine	14
6.4.3	Zeitplan.....	15
6.5	Kostenplanung.....	16
6.5.1	Kostenzusammenstellung Honorar:.....	16
6.5.2	Kostenzusammenstellung Material:.....	16
6.6	Risikoanalyse.....	17
6.6.1	Massnahmen.....	18
6.7	Pflichtenheft.....	19
6.7.1	Zielbestimmung	19
6.7.2	Produkteinsatz.....	20
6.7.3	Produktfunktionen.....	20
6.7.4	Qualitätsanforderungen	20
6.7.5	Benutzeroberfläche	20
6.7.6	Technische Produktumgebung	20
7	OOA / Analyse.....	21
7.1	Ist-Zustand.....	21
7.2	Verbesserung	21
7.3	Auswahl und Evaluierung	22
7.3.1	Sensor	22
7.3.2	Systemübersicht.....	24
7.3.3	Data-Analyzer.....	25
8	OOD / Design	26
8.1	State Diagramm.....	26
8.2	Anwendungsfälle Sensor	27
8.2.1	Anwendungsfall Sensor #UC01.....	28
8.2.2	Anwendungsfall Sensor #UC02.....	29
8.2.3	Anwendungsfall Sensor #UC03.....	30
8.3	Anwendungsfälle Data-Analyzer.....	31
8.3.1	Anwendungsfall Data-Analyzer #UC01	32

8.3.2	Anwendungsfall Data-Analyzer #UC02	33
8.3.3	Anwendungsfall Data-Analyzer #UC03	34
8.3.4	Anwendungsfall Data-Analyzer #UC04	35
9	OOP / Implementation	36
9.1	Sensor – Hardware	36
9.1.1	Prototyp Breadboard	36
9.1.2	PCB / Leiterplatine.....	37
9.1.3	Sensorgehäuse	38
9.1.4	Montage	39
9.2	Sensor – Software	40
9.2.1	Micropython.....	40
9.2.2	Entwickelte Bibliotheken.....	40
9.2.3	MicroPython Bibliotheken.....	44
9.2.4	States	45
9.2.5	Webinterface Sensor	49
9.3	Schnittstelle Sensor / Data-Analyzer	52
9.3.1	Sensordaten speichern / CSV-File	52
9.4	Data-Analyzer	53
9.4.1	Sidebar	53
9.4.2	Upload Data	54
9.4.3	Delete Data	54
9.4.4	Set Marker on Map	54
9.4.5	Reset Map	55
9.4.6	Datenauswertung	55
9.4.7	Datenfluss	56
9.4.8	Leaflet.....	57
10	Testbericht / Auswertung.....	59
10.1	Testspezifikation	59
10.2	Sensor: Normale Tests	60
10.3	Sensor: Subnormale Tests	61
10.4	Sensor: Extreme Tests	62
10.5	Data-Analyzer: Normale Tests.....	63
10.6	Data-Analyzer: Subnormale Tests	64
10.7	Data-Analyzer: Extreme Tests	65
11	Abschlussbericht	66
11.1	Sachergebnisse	66
11.1.1	Veränderungen der Aufgabenstellung / Zielsetzung	66
11.1.2	Zielüberprüfung	66
11.1.3	Erfahrungen	67
11.2	Zeitmanagement.....	68
11.2.1	Projektphasen	69
11.3	Kostenübersicht.....	70
11.3.1	Kosten-Controlling Honorar.....	70

11.3.2	Kosten-Controlling Material	70
11.4	Ausblick	71
11.4.1	Restaktivität	71
11.4.2	Ergänzungen und Erweiterungen.....	71
11.5	Schlusswort / Reflexion.....	72
11.5.1	Pascal Rusca	72
11.5.2	Janik Schilter.....	72
11.6	Eigenständigkeitserklärung.....	73
12	Anhang	74
12.1	Abgabe Projektunterlagen	74
12.2	Sitzungsprotokolle	75

3 Abbildungsverzeichnis

<i>Abbildung 1 - Sensorsystem (Quelle: Eigenes Bild)</i>	2
<i>Abbildung 2 - Data Analyzer (Quelle: Eigenes Bild)</i>	2
<i>Abbildung 3 - Tabelle Abkürzungsverzeichnis (Quelle: Eigene Tabelle)</i>	8
<i>Abbildung 4 – Projektorganisation (Quelle: Eigene Abbildung)</i>	10
<i>Abbildung 5 – Pascal Rusca (Quelle: Eigenes Bild)</i>	11
<i>Abbildung 6 - Berufserfahrung Pascal Rusca (Quelle: Eigene Grafik)</i>	11
<i>Abbildung 7 - Janik Schilter (Quelle: Eigenes Bild)</i>	12
<i>Abbildung 8 - Berufserfahrung Janik Schilter (Quelle: Eigene Grafik)</i>	12
<i>Abbildung 9 – Projektstrukturplan (Quelle: Eigene Grafik)</i>	13
<i>Abbildung 10 – Projektphasen (Quelle: Eigenes Diagramm)</i>	14
<i>Abbildung 11 - Tabelle Meilensteine (Quelle: Eigene Tabelle)</i>	14
<i>Abbildung 12 - Zeitplan (Quelle: Eigener Zeitplan)</i>	15
<i>Abbildung 13 - Tabelle Honorar (Quelle: Eigene Tabelle)</i>	16
<i>Abbildung 14- Tabelle Material (Quelle: Eigene Tabelle)</i>	16
<i>Abbildung 15 - Risikoanalyse (Quelle: Eigenes Bild)</i>	17
<i>Abbildung 16 - Tabelle Risiken (Quelle: Eigene Tabelle)</i>	17
<i>Abbildung 17 – Tabelle Zusammenfassung Zieldefinition (Quelle: Eigene Tabelle)</i>	19
<i>Abbildung 18 - Tabelle Microcontroller (Quelle: Eigene Tabelle)</i>	22
<i>Abbildung 19 - Arduino RP2040 Connect (Quelle: www.docs.arduino.cc)</i>	23
<i>Abbildung 20 - Adafruit Ultimate GPS (Quelle: www.adafruit.com)</i>	23
<i>Abbildung 21 - Adafruit MicroSD (Quelle: www.adafruit.com)</i>	23
<i>Abbildung 22 - Systemübersicht (Quelle: Eigene Skizze)</i>	24
<i>Abbildung 23 - Flask Logo (Quelle: www.flask.palletsprojects.com)</i>	25
<i>Abbildung 24 - Bootstrap Logo (Quelle: www.getbootstrap.com)</i>	25
<i>Abbildung 25 - OSM Logo (Quelle: www.osm.ch)</i>	25
<i>Abbildung 26 - Leaflet Logo (Quelle: www.leafletjs.com)</i>	25
<i>Abbildung 27 - State Diagramm Sensor (Quelle: Eigenes Diagramm «PlantUML»)</i>	26
<i>Abbildung 28 - Anwendungsfalldiagramm Sensor (Quelle: Eigenes Diagramm)</i>	27
<i>Abbildung 29 - Tabelle Sensor #UC01 (Quelle: Eigene Tabelle)</i>	28
<i>Abbildung 30 - Programmablauf Sensor #UC01 (Quelle: Eigenes Diagramm)</i>	28
<i>Abbildung 31 - Tabelle Sensor #UC02 (Quelle: Eigene Tabelle)</i>	29
<i>Abbildung 32 - Programmablauf Sensor #UC02 (Quelle: Eigenes Diagramm)</i>	29
<i>Abbildung 33 - Tabelle Sensor #UC03 (Quelle: Eigene Tabelle)</i>	30
<i>Abbildung 34 - Programmablauf Sensor #UC03 (Quelle: Eigenes Diagramm)</i>	30
<i>Abbildung 35 - Anwendungsfalldiagramm Data-Analyzer (Quelle: Eigenes Diagramm)</i>	31
<i>Abbildung 36 - Tabelle Data-Analyzer #UC01 (Quelle: Eigene Tabelle)</i>	32
<i>Abbildung 37 - Programmablauf Data-Analyzer #UC01 (Quelle: Eigenes Diagramm)</i>	32
<i>Abbildung 38 - Tabelle Data-Analyzer #UC02 (Quelle: Eigene Tabelle)</i>	33
<i>Abbildung 39 - Programmablauf Data-Analyzer #UC02 (Quelle: Eigenes Diagramm)</i>	33
<i>Abbildung 40 - Tabelle Data-Analyzer #UC03 (Quelle: Eigene Tabelle)</i>	34
<i>Abbildung 41 - Programmablauf Data-Analyzer #UC03 (Quelle: Eigenes Diagramm)</i>	34
<i>Abbildung 42 - Tabelle Data-Analyzer #UC04 (Quelle: Eigene Tabelle)</i>	35
<i>Abbildung 43 - Programmablauf Data-Analyzer #UC04 (Quelle: Eigenes Diagramm)</i>	35
<i>Abbildung 44 - Prototyp Breadboard (Quelle: Eigene Fritzing-Steckansicht)</i>	36
<i>Abbildung 45 – Leiterplatine Vorderseite (Quelle: Eigener Fritzing-Plan)</i>	37
<i>Abbildung 46- Leiterplatine bestückt (Quelle: Eigenes Bild)</i>	37
<i>Abbildung 47 - Ansicht Gehäuse «Entwicklungsstufe» (Quelle: Eigenes Bild)</i>	38
<i>Abbildung 48 - Gehäuse mit Elektronik (Quelle: Eigenes Bild)</i>	38
<i>Abbildung 49 - Ansicht Gehäuse «Spritzwasserfest» (Quelle: Eigenes Bild)</i>	38
<i>Abbildung 50 - Gehäuse mit Elektronik (Quelle: Eigenes Bild)</i>	38

Abbildung 51 - Gehäuse mit Elektronik (Quelle: Eigenes Bild)	38
Abbildung 52 - Gehäuse montiert auf Motorrad (Quelle: Eigenes Bild).....	39
Abbildung 53 - Tabelle Funktionen GPS_lib (Quelle: Eigene Tabelle).....	40
Abbildung 54 - Tabelle Öffentliche Klassen Variablen GPS_lib (Quelle: Eigene Tabelle)	41
Abbildung 55 - Tabelle Struktur GNRMC Nachricht (Quelle: Eigene Tabelle)	41
Abbildung 56 - Sensor Webinterface "Startseite" (Quelle: Eigenes Bild)	49
Abbildung 57 - Sensor Webinterface "Kalibration Seite" (Quelle: Eigenes Bild)	49
Abbildung 58 - Sensor Webinterface "Init Record" (Quelle: Eigenes Bild)	50
Abbildung 59 - Sensor Webinterface "Record" (Quelle: Eigenes Bild).....	50
Abbildung 60 - Sensor Webinterface "Stop Record" (Quelle: Eigenes Bild)	50
Abbildung 61 - Sensor Webinterface "Error" (Quelle: Eigenes Bild).....	51
Abbildung 62 - Tabelle CSV-File (Quelle: Eigene Tabelle).....	52
Abbildung 63 - Data-Analyzer (Quelle: Eigenes Bild)	53
Abbildung 64 - Sidebar (Quelle: Eigenes Bild)	53
Abbildung 65 - Modal Upload Data (Quelle: Eigenes Bild)	54
Abbildung 66 - Modal Delete File (Quelle: Eigenes Bild)	54
Abbildung 67 - Modal Set Marker (Quelle: Eigenes Bild)	54
Abbildung 68 - Reset Marker (Quelle: Eigenes Bild).....	55
Abbildung 69 - Datenauswertung (Quelle: Eigenes Bild)	55
Abbildung 70 - Datenfluss Data-Analyzer (Quelle: Eigene Grafik)	56
Abbildung 71 - Standortdatenformat DMS (Quelle: Eigene Tabelle)	57
Abbildung 72 - Markercluster (Quelle: Eigene Abbildung).....	57
Abbildung 73 - Sensor: Normale Tests Tabelle Test-Run #1 (Quelle: Eigene Tabelle)	60
Abbildung 74 - Sensor: Normale Tests Tabelle Change Log (Quelle: Eigene Tabelle)	60
Abbildung 75 - Sensor: Normale Tests Tabelle Test-Run #2 (Quelle: Eigene Tabelle)	60
Abbildung 76 - Sensor: Subnormale Tests Tabelle Test-Run #1 (Quelle: Eigene Tabelle).....	61
Abbildung 77 - Sensor: Subnormale Tests Tabelle Change Log (Quelle: Eigene Tabelle)	61
Abbildung 78 - Sensor: Subnormale Tests Tabelle Test-Run #2 (Quelle: Eigene Tabelle).....	61
Abbildung 79 - Sensor: Extreme Tests Tabelle Test-Run #1 (Quelle: Eigene Tabelle).....	62
Abbildung 80 - Sensor: Extreme Tests Tabelle Change Log (Quelle: Eigene Tabelle)	62
Abbildung 81 - Sensor: Extreme Tests Tabelle Test-Run #2 (Quelle: Eigene Tabelle).....	62
Abbildung 82 - Data-Analyzer: Normale Tests Tabelle Test-Run #1 (Quelle: Eigene Tabelle)	63
Abbildung 83 - Data-Analyzer: Normale Tests Tabelle Change Log (Quelle: Eigene Tabelle)	63
Abbildung 84 - Data-Analyzer: Normale Tests Tabelle Test-Run #2 (Quelle: Eigene Tabelle).....	63
Abbildung 85 - Data-Analyzer: Subnormale Tests Tabelle Test-Run #1 (Quelle: Eigene Tabelle).....	64
Abbildung 86 - Data-Analyzer: Subnormale Tests Tabelle Change Log (Quelle: Eigene Tabelle).....	64
Abbildung 87 - Data-Analyzer: Subnormale Tests Tabelle Test-Run #2 (Quelle: Eigene Tabelle).....	64
Abbildung 88 - Data-Analyzer: Extreme Tests Tabelle Test-Run #1 (Quelle: Eigene Tabelle).....	65
Abbildung 89 - Data-Analyzer: Extreme Tests Tabelle Change Log (Quelle: Eigene Tabelle).....	65
Abbildung 90 - Data-Analyzer: Extreme Tests Tabelle Test-Run #2 (Quelle: Eigene Tabelle).....	65
Abbildung 91 - Tabelle Zielüberprüfung (Quelle: Eigene Tabelle).....	66
Abbildung 92 - Zeitplan Abschlussbericht (Quelle: Eigener Zeitplan).....	68
Abbildung 93 - Tabelle Kosten-Controlling Honorar (Quelle: Eigene Tabelle).....	70
Abbildung 94 - Tabelle Kosten-Controlling Material (Quelle: Eigene Tabelle)	70

4 Glossar

Begriff / Abkürzung	Erklärung
Visual Studio Code IDE / VSC	Visual Studio Code ist ein kostenloser Quelltext-Editor von Microsoft.
Thonny IDE	Thonny ist eine integrierte Entwicklungsumgebung für Python/Micropython, die für Anfänger ausgelegt ist.
OOA	Objektorientierte Analyse: Entwicklungsprozess eines Softwaresystems. Es handelt sich dabei um eine Anforderungsanalyse.
OOD	Objektorientiertes Design: Entwicklungsprozess eines Softwaresystems. Es handelt sich dabei um einen Systementwurf.
OOP	Objektorientierte Programmierung: Entwicklungsprozess eines Softwaresystems. Es handelt sich dabei um eine Systemimplementation.
PCB	Printed circuit board: Hierbei handelt es sich um eine Leiterplatte, welche als Träger für elektronische Bauteile dient.
FDM-Drucker	Fused Deposition Modeling: 3D-Drucker Verarbeitet auf einer Spule gewickelte Kunststofffäden. Druckt schichtweise auf eine Druckplattform.
PETG	Polyethylenterephthalat mit Glykol: Thermoplastischer Kunststoff für 3D-Drucker. Zeichnet sich durch hohe Transparenz und niedrige Viskosität aus.
TPU	Thermoplastisches Polyurethan: Thermoplastischer Kunststoff für 3D-Drucker. Mittelweg zwischen Gummi und Kunststoff: elastisch, flexibel, griffig, strapazierfähig, stark
CSV	Comma-separated values: Dateiformat, Textdatei. Wird zur Speicherung und zum Austausch von strukturierten Daten verwendet.

Abbildung 3 - Tabelle Abkürzungsverzeichnis (Quelle: Eigene Tabelle)

5 Einleitung

5.1 Vorwort

Am Ende der Weiterbildung zum Dipl. Techniker/in HF Informatik Fachrichtung Applikationsentwicklung wird eine Diplomarbeit durchgeführt. Ziel der Diplomarbeit ist, ein Projektthema aus dem Themenbereich der Informatik zu wählen und bei der Durchführung das in der Weiterbildung Gelernte praktisch umzusetzen.

Da die beiden Diplomanden Pascal Rusca und Janik Schilter begeisterte Motorradfahrer sind, haben diese sich dazu entschieden, ein System zur Erfassung von Fahrdaten zu entwickeln.

Das Ziel der Diplomarbeit ist, einen funktionstüchtigen Fahrdatenschreiber zu entwickeln. Dieser soll zum Eigengebrauch verwendet werden und ermöglichen, das eigene Motorrad-Fahrverhalten zu verbessern.

6 Projektmanagement

6.1 Projektorganisation

Die Projektbeteiligten und deren Verantwortlichkeiten sind in folgendem Diagramm ersichtlich:

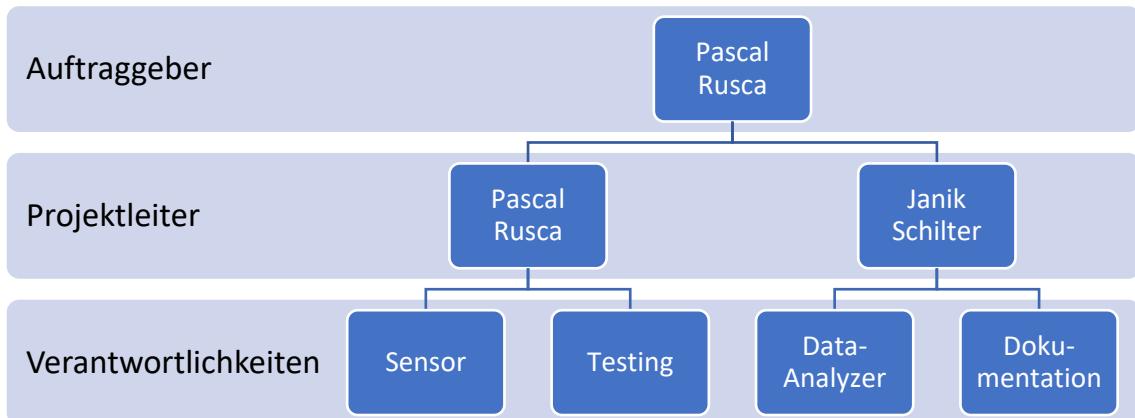


Abbildung 4 – Projektorganisation (Quelle: Eigene Abbildung)

Kommunikationsmittel: Telefon, WhatsApp, MS Teams

Workgroup Plattform: GitBlit (Entwicklung), OneDrive (Dokumentation)

Backup: GitBlit

Developer Tools: Visual Studio Code IDE

Thonny IDE

Dokumentation: MS Office - Word: .docx

MS Office - Excel: .xmls

Fertige Dokumente: PDF (.pdf)

6.2 Die Projektbeteiligten

6.2.1 Pascal Rusca

Auftraggeber / Projektleiter

Pascal Rusca
Brunnenhöfli 18
6012 Obernau
p.rusca@hotmail.com



Abbildung 5 – Pascal Rusca (Quelle: Eigener Bild)

Berufserfahrung



Abbildung 6 - Berufserfahrung Pascal Rusca (Quelle: Eigene Grafik)

6.2.2 Janik Schilter

Projektleiter

Janik Schilter
Chilenmattli 7
6055 Alpnach Dorf
janik.schilter@gmail.com



Abbildung 7 - Janik Schilter
(Quelle: Eigene Bild)

Berufserfahrung

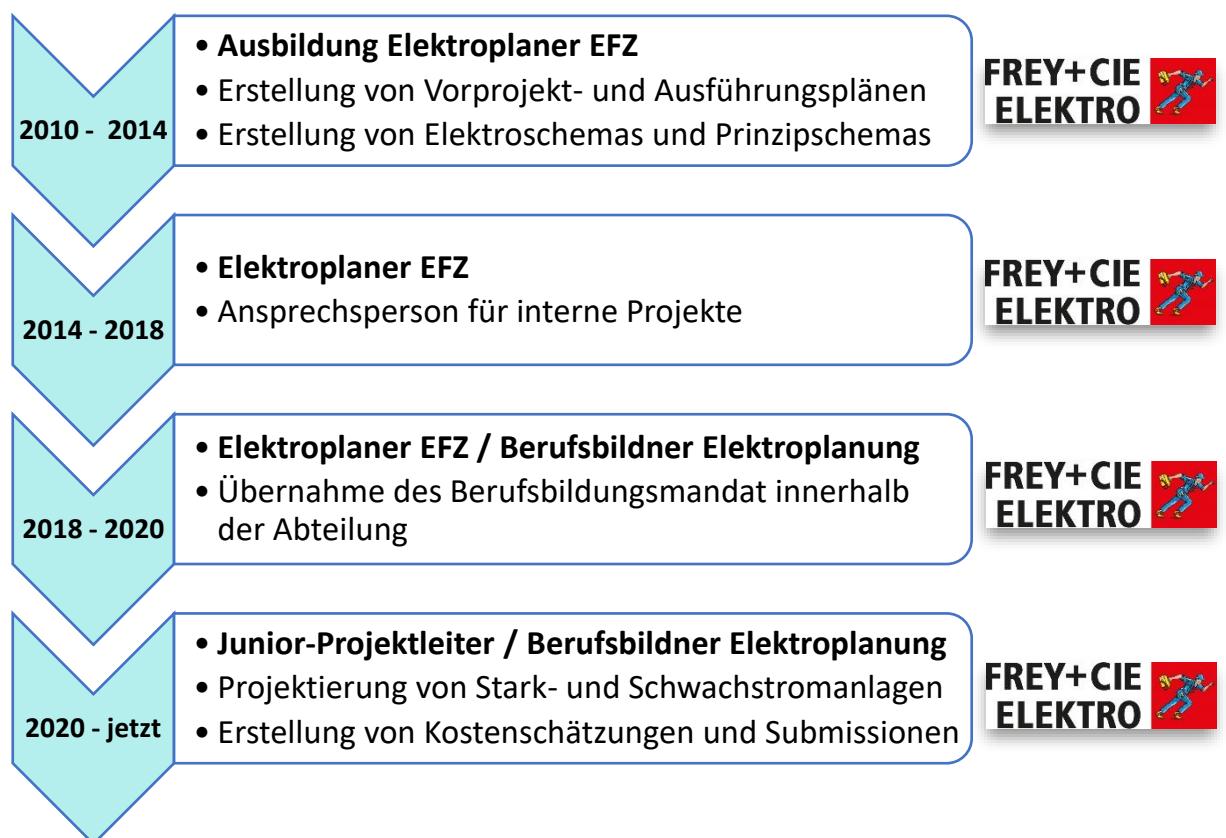


Abbildung 8 - Berufserfahrung Janik Schilter (Quelle: Eigene Grafik)

6.3 Projektstrukturplan

Zur Verdeutlichung der Übersicht wurde ein Projektstrukturplan angelegt. Aus diesem sind das Projekt, die drei verschiedenen Projektphasen (exkl. Projektinitialisierungsphase) und die wichtigsten zu erledigenden Arbeitspakete ersichtlich. Dieser Projektstrukturplan gilt als Leitfaden und hilft, die zu erledigenden Arbeiten den richtigen Projektphasen zu zuordnen.

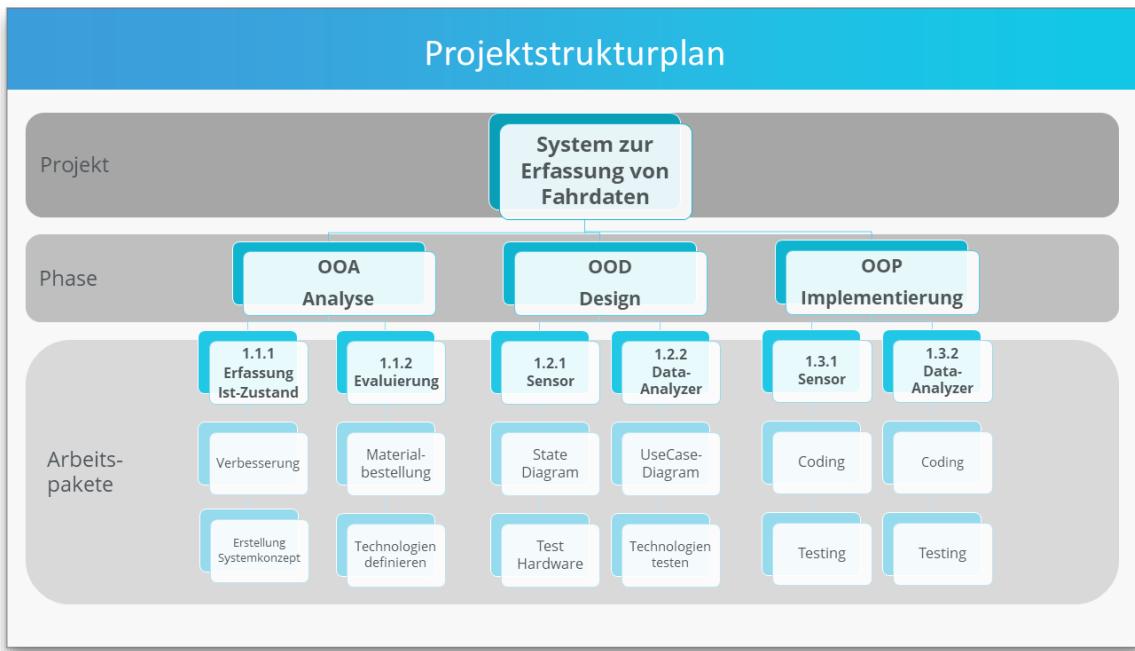


Abbildung 9 – Projektstrukturplan (Quelle: Eigene Grafik)

1.1.1 OOA / Erfassung Ist-Zustand

- Verbesserung gegenüber Ist-Zustand
- Entwurf von Systemkonzept mithilfe eines Systemkonzeptübersichtsdiagramm

1.1.2 OOA / Evaluierung der einzusetzenden Hardware

- Durchführung der Materialbestellung frühestmöglich
- Technologien definieren: Technische Produktumgebung eruieren und festlegen

1.2.1 OOD / Sensor

- Erstellung State Diagram
- Durchführung erster Funktionstests der Hardware

1.2.2 OOD / Data-Analyzer

- Erstellung UseCase-Diagramm (Darstellung der UseCases und Datenflüsse)
- Durchführung erster Tests der eingesetzten Frameworks, Libraries, etc.

1.3.1 OOP / Sensor

- Coding: Erstellung des Quellcodes
- Testing: Erstellung Testkonzept, Testfälle und Durchführung der Tests

1.3.2 OOP / Data-Analyzer

- Coding: Erstellung des Quellcodes
- Testing: Erstellung Testkonzept, Testfälle und Durchführung der Tests

6.4 Termin- und Zeitmanagement

6.4.1 Projektphasen

Das Projekt ist in folgende Phasen gegliedert:

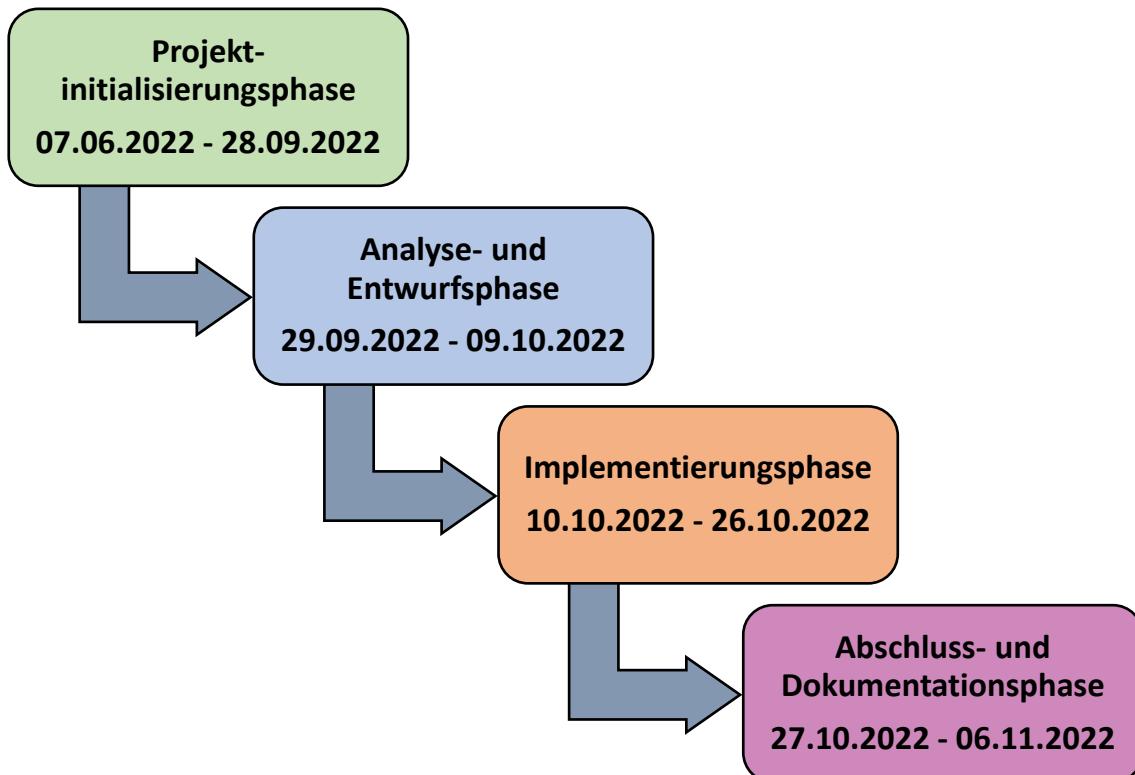


Abbildung 10 – Projektphasen (Quelle: Eigene Diagramm)

6.4.2 Meilensteine / Termine

Eine Auflistung der wichtigsten Meilensteine und Termine:

Meilensteine / Termine	Verantwortung	Start	Ende
Projektinitialisierungsphase	JS	26.09.2022	28.09.2022
Analyse- und Entwurfsphase	PR	29.09.2022	09.10.2022
Implementierungs- und Einführungsphase	PR	10.10.2022	26.10.2022
Abschluss- und Dokumentationsphase	JS	27.10.2022	06.11.2022
Abgabe Diplomarbeit	JS	07.11.2022	07.11.2022
Präsentation	PR	12.11.2022	12.11.2022

Abbildung 11 - Tabelle Meilensteine (Quelle: Eigene Tabelle)

6.4.3 Zeitplan

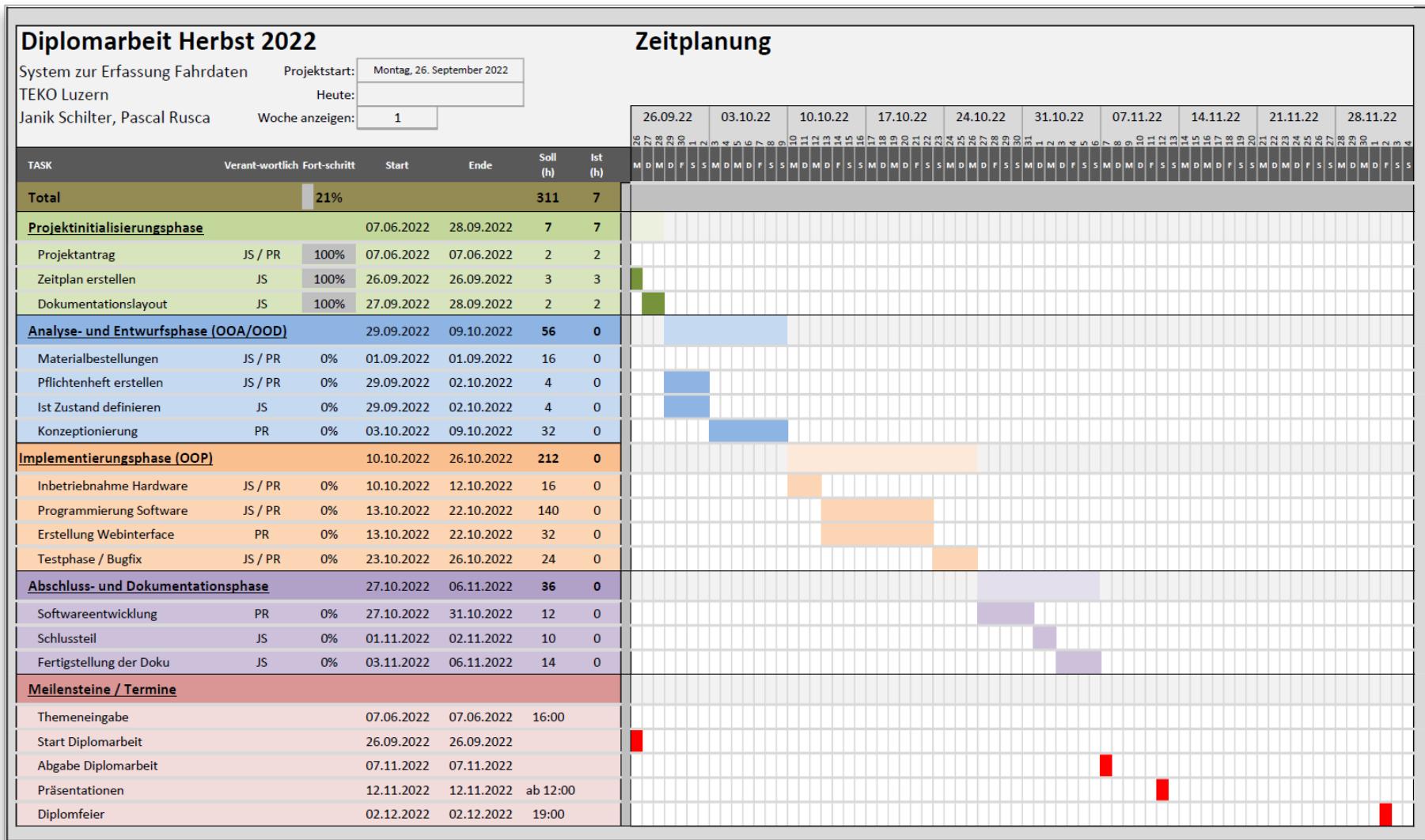


Abbildung 12 - Zeitplan (Quelle: Eigener Zeitplan)

6.5 Kostenplanung

6.5.1 Kostenzusammenstellung Honorar:

Diese Diplomarbeit wurde zwar nicht im gewerblichen Umfeld durchgeführt, das heisst es gab offiziell keinen Auftraggeber oder Kunden, jedoch wurde im Sinne des Projektmanagements trotzdem eine Kostenzusammenstellung erstellt.

Dafür wurde das Honorar mit einem geschätzten Aufwand von 300 Arbeitsstunden festgelegt. Als marktüblichen Stundenansatz wird hier CHF 120.00 exkl. MwSt. verwendet. Daraus ergeben sich Honorarkosten von CHF 36'000.00 exkl. MwSt.

Kostenpunkt	Aufwände geschätzt	Stundenansatz	Kosten
Honorar Projektleiter	300h	CHF 120.00 exkl. MwSt.	CHF 36'000.00 exkl. MwSt.

Abbildung 13 - Tabelle Honorar (Quelle: Eigene Tabelle)

6.5.2 Kostenzusammenstellung Material:

Im Folgenden ist eine Kostenzusammenstellung aller geplanten Hardwarekomponenten ersichtlich. In den folgenden Preisen sind auch die Lieferkosten und allfällige Zollgebühren einberechnet. Daraus ergibt sich ein Gesamttotal von CHF 274.40.

Produkt	Hersteller	Typ	Anzahl	Stückpreis	Preis
Breakoutboard			2	CHF 2.00	CHF 4.00
Mikrocontroller	Arduino	RP2040	2	CHF 25.00	CHF 50.00
SD-Karte	Intenso	16 GB	2	CHF 5.00	CHF 10.00
SD-Modul	Adafruit	Micro SD	2	CHF 5.90	CHF 11.80
GPS-Modul	Adafruit	Ultimate	2	CHF 39.30	CHF 78.60
Div. Bauteile	-	-	2	CHF 10.00	CHF 20.00
Leiterplatine	JLCPCB	-	Pschl.	CHF 50.00	CHF 50.00
Gehäuse	MotoData		Pschl.	CHF 50.00	CHF 50.00
				Total	CHF 274.40

Abbildung 14- Tabelle Material (Quelle: Eigene Tabelle)

6.6 Risikoanalyse

In einem ersten Verfahren wurden alle möglichen Risiken eruiert, welche dann in einer zweiten Phase nach der Eintrittswahrscheinlichkeit und dem Schadenspotential in einer Risikomatrix eingeteilt wurden. Sind das Schadenspotential und die Wahrscheinlichkeit besonders hoch, ist es unausweichlich, dass Massnahmen ergriffen werden müssen, um den Eintritt zu verhindern. Sind im Gegensatz das Potential sowie auch die Wahrscheinlichkeit auf einen Eintritt tief, ist es gut, wenn das Risiko erkannt wird, es müssen jedoch keine Massnahmen ergriffen werden.

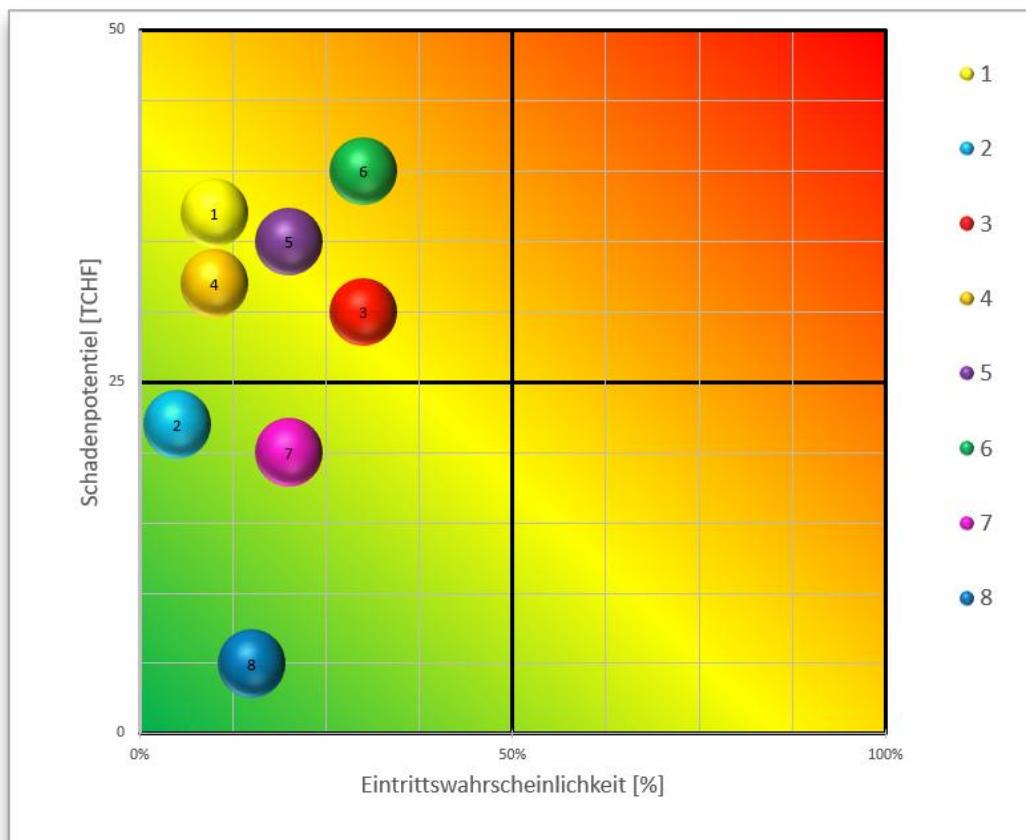


Abbildung 15 - Risikoanalyse (Quelle: Eigenes Bild)

Nr.	Kategorie (Liste)	Eintritts- wahrschein- lichkeit [%]	Schadens- potential [TCHF]	Risiko [TCHF]
1	Ausfall Projektleiter	10%	37	3.7
2	Konflikte zwischen Mitarbeitern	5%	22	1.1
3	Lieferfristen Hardware	30%	30	9
4	Defekt gelieferte Hardware	10%	32	3.2
5	Falsch evaluierte Hardware	20%	35	7
6	Hardwareverlust (Nässe, Vibrationen...)	30%	40	12
7	Softwareprobleme	20%	20	4
8	Datenverlust (Backup)	15%	5	0.75

Abbildung 16 - Tabelle Risiken (Quelle: Eigene Tabelle)

6.6.1 Massnahmen

Wie die Risikoanalyse nun klar aufzeigt, gibt es in dieser Analyse sechs Kategorien, welche unter höherer Beobachtung stehen müssen:

6.6.1.1 #1 Ausfall Projektleiter

Bei Ausfall eines Projektleiters muss schnellstmöglich ein Lösungsweg mit dem Betreuer gefunden werden.

6.6.1.2 #3 Lieferfristen Hardware

Ein grosses Problem ist die Halbleiterknappheit, in den Medien oft auch "Chipknappheit" betitelt. Sie wurde primär durch die Corona-Pandemie verursacht. Produktionswerke mussten heruntergefahren werden, während die Nachfrage deutlich stieg. Damit das Projekt nicht an der Verfügbarkeit von Komponenten scheitert, wurden diese bereits vor dem Start der Diplomarbeit evaluiert und bestellt.

6.6.1.3 #4 Defekt gelieferte Hardware

Bei defekt gelieferter Hardware muss dies möglichst schnell mit dem Lieferanten geklärt werden. Um die Lieferkette zu verkürzen, wurde nur auf Lieferanten in der Schweiz gesetzt.

6.6.1.4 #5 Falsch evaluierte Hardware

Bei falsch evaluierter Hardware muss möglichst schnell eine Alternative gefunden werden, die bei einem Schweizer Lieferant verfügbar ist.

6.6.1.5 #6 Hardwareverlust (Nässe, Vibrationen)

Damit die erfolgreiche Durchführung durch Hardwareverlust nicht gefährdet wird, werden genügend Reservekomponenten bestellt.

6.7 Pflichtenheft

6.7.1 Zielbestimmung

6.7.1.1 Musskriterien

- Erfassung von Telemetriedaten mindestens alle ~500ms
 - o Standort-Daten (Genauigkeit: 5m unter freiem Himmel)
 - o Geschwindigkeit (Genauigkeit: +/- 5km/h)
 - o Neigungswinkel / Kurvenlage (Genauigkeit: +/- 5°, bis max. 60°)
 - o Speicherung der erhobenen Daten auf einem lokalen Speichermedium
 - o Bedienung der Sensorfunktionen via Smartphone
- Möglichkeit zur Einsicht und Auswertung der erhobenen Daten via Data-Analyzer als Browserapplikation.
 - o Topographische Karte mit Messpunkten
 - o Anzeige der Messwerte pro Messpunkt

6.7.1.2 Wunschkriterien

- Data-Analyzer: Überlagerung von bis zu drei gemessenen Aufnahmen
- Erstellung eines PCBs
- Erstellung eines Gerät-Gehäuses

6.7.1.3 Abgrenzungskriterien

- Im Rahmen der Diplomarbeit wird ein funktionstüchtiger Prototyp entwickelt. Es ist nicht Ziel, das Produkt in den Markt einzuführen und zu monetarisieren. Das Produkt soll ausschliesslich im privaten Umfeld der Projektbeteiligten eingesetzt werden.
- Es ist nicht vorgesehen, die Browserapplikation Data-Analyzer im Rahmen dieser Diplomarbeit als Webseite zu hosten.

Ziel	Beschreibung	Muss	Wunsch	Abgrenzung
#1	Sensor: Erfassung von Telemetriedaten mindestens alle ~500ms	X		
#2	Sensor: Genauigkeit Standort-Daten mind. 5m unter freiem Himmel	X		
#3	Sensor: Genauigkeit Geschwindigkeit (+/- 5km/h)	X		
#4	Sensor: Neigungswinkel / Kurvenlage (+/- 5°, bis max. 60°)	X		
#5	Sensor: Speicherung der Daten auf lokalem Speichermedium	X		
#6	Sensor: Bedienung der Sensorfunktionen via Smartphone	X		
#7	Data-Analyzer: Auswertung der erhobenen Daten via Browserapplikation.	X		
#8	Data-Analyzer: Topographische Karte mit den Messpunkten	X		
#9	Data-Analyzer: Anzeige der Messwerte pro Messpunkt	X		
#10	Data-Analyzer: Überlagerung von bis zu drei gemessenen Aufnahmen		X	
#11	Sensor: Erstellung eines PCBs		X	
#12	Sensor: Erstellung eines Gerät-Gehäuses		X	
#13	Sensor / Data-Analyzer: Produkt im Markt einführen			X
#14	Data-Analyzer: Hosting als Webseite			X

Abbildung 17 – Tabelle Zusammenfassung Zieldefinition (Quelle: Eigene Tabelle)

6.7.2 Produkteinsatz

6.7.2.1 Anwendungsbereiche

- Motorrad
- Auto
- Sonstige Fahrzeuge

6.7.2.2 Zielgruppe

Die Zielgruppe umfasst Motorradfahrende, welche gerne deren Fahrstil analysieren und verbessern möchten.

6.7.3 Produktfunktionen

6.7.3.1 Funktionen am Sensor:

- Zugriff auf den Sensor via Smartphone
 - o Möglichkeit zur Kalibrierung des Neigungswinkelsensors
 - o Start und Stopp des Fahrdatenschreibers
 - o Export der Fahrdaten

6.7.3.2 Funktionen Data-Analyzer

- Import und Verwaltung der erfassten Fahrdaten
 - Einsicht der Fahrdaten auf einer topographischen Karte
-

6.7.4 Qualitätsanforderungen

- Erfassung der Fahrdaten mindestens alle ~500ms
 - Möglichkeit zur Kalibrierung des Systems
 - Schutz vor Spritzwasser (Gehäuse)
-

6.7.5 Benutzeroberfläche

6.7.5.1 Design

Das Design des Data-Analyzer soll dem aktuellen «State of the Art» entsprechen.

6.7.5.2 Struktur

Der Data-Analyzer soll intuitiv zu bedienen sein.

6.7.6 Technische Produktumgebung

6.7.6.1 Entwicklungsumgebung

- Visual Studio Code
- PlatformIO

6.7.6.2 Data-Analyzer

- Python 3.8.10
- Flask 2.2.2
- Bootstrap v4.5

7 OOA / Analyse

7.1 Ist-Zustand

Die genaue Wahrnehmung der Umgebung bei Kurvenfahrten auf dem Motorrad ist meistens schwierig, da sich auf andere wichtige Dinge wie zum Beispiel der Verlauf der Kurve, Strassenverhältnisse, andere Verkehrsteilnehmer und noch vieles mehr konzentriert werden muss.

Sich in der Kurve auf die eigene Fahrtechnik zu fokussieren und daraus Rückschlüsse aus dem eigenen Fahrverhalten zu schliessen, ist daher nur bedingt möglich.

Hier kann ein Fahrdatenschreiber Abhilfe schaffen. Ein Fahrdatenschreiber ermöglicht, das Fahrverhalten mithilfe verschiedener Telemetriedaten nach beendeter Motorradtour zu analysieren und daraus Rückschlüsse zu ziehen.

Auf dem Markt gibt es bereits diverse Fahrdatenschreiber. Jedoch sind diese in der Regel erst ab ca. 600.00 Euro aufwärts erhältlich und bieten nicht immer den gewünschten Funktionsumfang.

7.2 Verbesserung

Mit der Erfassung von Telemetriedaten während der Fahrt, können diese auch im Nachhinein ausgewertet werden, ohne dass sich spezifisch an alle Elemente wie Geschwindigkeit, Kurve, etc. erinnert werden muss.

Damit die erfassten Daten einfach ersichtlich sind, werden die erstellten Messpunkte mithilfe des Data-Analyzer auf einer Weltkarte dargestellt. Somit ist die abgefahrenen Route ersichtlich und die Fahrdaten können klar und übersichtlich eingesehen werden.

Ziel dieser Diplomarbeit ist es, ein kostengünstiges System zu entwickeln, das dabei hilft, die Fahrkünste von Motorradfahrenden auf sichere und ungefährliche Weise zu verbessern.

7.3 Auswahl und Evaluierung

Die Hardware wurde wegen eventuellen Lieferengpässen bereits vor Start der Diplomarbeit evaluiert, beschafft und auf Fehler überprüft.

7.3.1 Sensor

7.3.1.1 Microcontroller

Damit die gemessenen Daten einfach vom späteren Produkt extrahiert werden können, sollte es möglich sein, per W-LAN oder Bluetooth eine Verbindung aufzubauen zu können. Ebenfalls wird eine möglichst hohe Rechenkapazität benötigt, um die GPS- und Sensordaten verarbeiten und abspeichern zu können. Um dies alles simultan berechnen zu können, sollte der Mikrocontroller auch über zwei separate ansteuerbare Cores verfügen.

Verglichen wurde ein den Diplomanden geläufiger und schon mehrfach eingesetzter Mikrocontroller, der sogenannte «ESP-32», mit einem neu erschienenen Arduino Board und dem «Raspberry Pi Pico W», welches mit seinem breiten Feature-Sets aus der Masse heraussticht.

	ESP-32	Arduino RP2040 Connect	Raspberry Pi Pico W
Betriebssystem	RTOS	-	-
WLAN/BT	Ja	Ja	Ja
Cores	2	2	2
Speicher	4 MB Onboard	16 MB Onboard	2MB Onboard
Stromverbrauch	80-170 mA		
Deep-Sleep	Ja	Ja	Ja
I2C	Ja	Ja	Ja
GPIO	19	12	30
RTC	Ja	Ja	Ja
SPI	Ja	Ja	Ja
IMU	Nein	Ja	Nein
Formfaktor	48 mm x 26 mm	45 mm x 18 mm	51 mm x 21 mm

Abbildung 18 - Tabelle Microcontroller (Quelle: Eigene Tabelle)

Die Entscheidung fiel auf den «Arduino RP2040 Connect», da dieser im Vergleich zu dem «ESP-32» und dem «Raspberry Pico» einen kleineren Formfaktor besitzt und bereits über ein eingebautes «IMU» verfügt. Dies ermöglicht es, ein Endprodukt mit kleineren Abmessungen zu konzipieren.

Der «Raspberry Pi Pico W» wurde auch aufgrund seiner fehlenden Unterstützung für W-LAN und Bluetooth zum Start der Auslieferungen nicht ausgewählt.

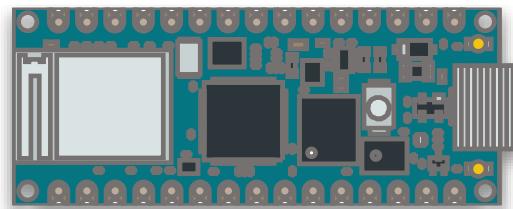


Abbildung 19 - Arduino RP2040 Connect (Quelle: www.docs.arduino.cc)

7.3.1.2 GPS

Auf dem Markt sind viele GPS-Module zu finden, hier wurde sich für ein Produkt von Adafruit entschieden, da die Diplomanden in der Vergangenheit sehr gute Erfahrungen mit diesen Modulen gemacht haben und es alle gängigen GPS-Regionen unterstützt.



Abbildung 20 - Adafruit Ultimate GPS (Quelle: www.adafruit.com)

7.3.1.3 SD-Karte

Um die gemessenen Daten abspeichern zu können, reichen die 16MB Speicher des «RP2040 Connect» nicht aus. Die Benutzer möchten nicht bereits nach einer einzigen Fahrt die Daten vom Gerät herunterladen müssen, nur um zu verhindern, dass die Daten bei der nächsten Fahrt bereits überschrieben werden. Darum haben die Diplomanden entschieden, eine SD-Karte zu verbauen. Auch hier gibt es mehrere Module und die Entscheidung ist wegen den guten Erfahrungen wieder für ein Adafruit Modul gefallen, welches per SPI angesteuert werden kann.

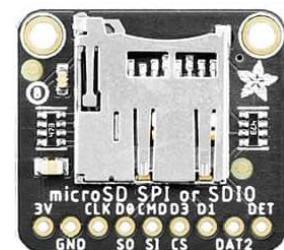


Abbildung 21 - Adafruit MicroSD (Quelle: www.adafruit.com)

7.3.2 Systemübersicht

Untenstehend ist eine Übersicht des geplanten Systems abgebildet:

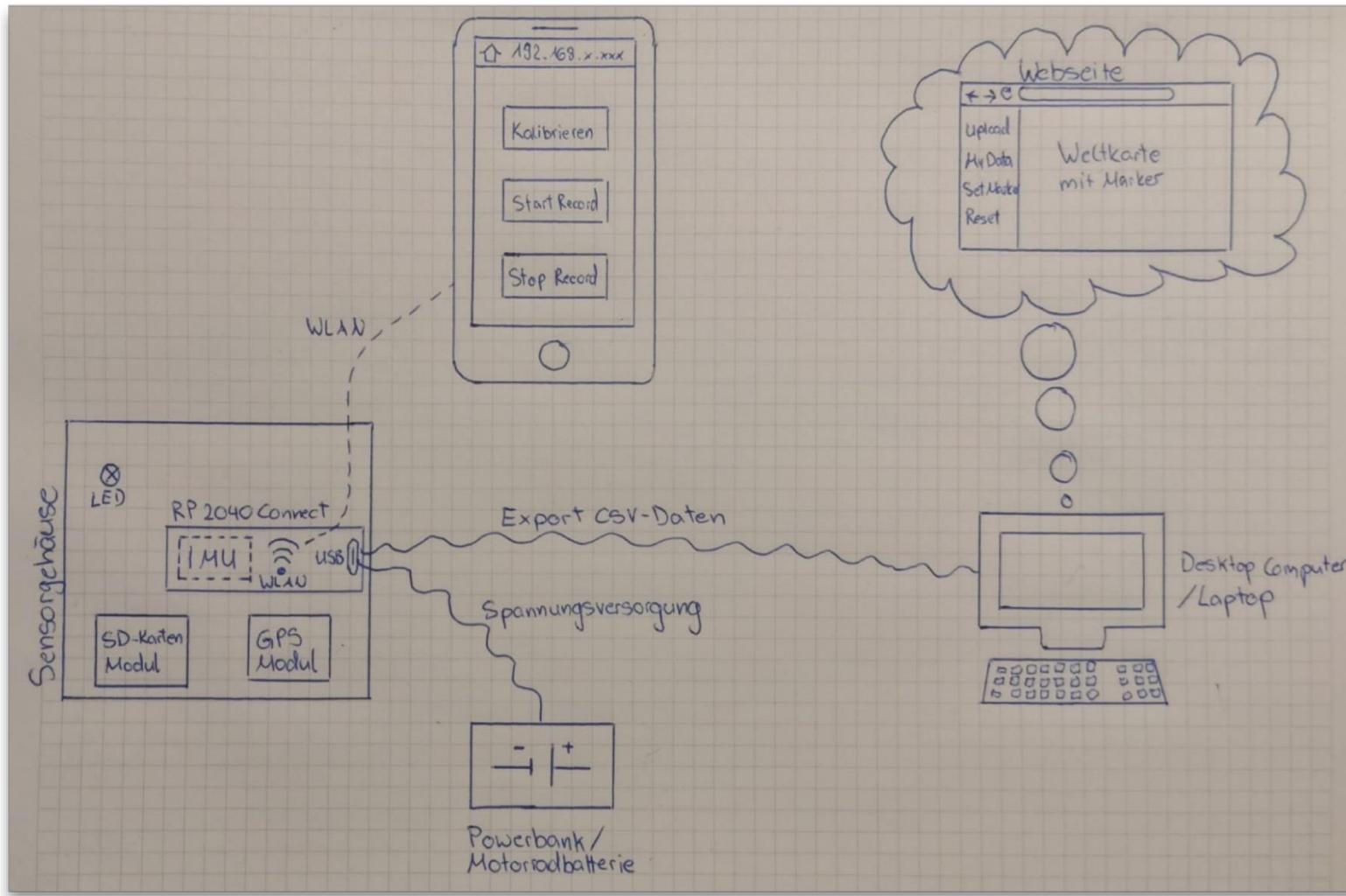


Abbildung 22 - Systemübersicht (Quelle: Eigene Skizze)

7.3.3 Data-Analyzer

7.3.3.1 Web-Framework

Da bei den Diplomanden bereits Erfahrungen mit dem Web-Framework Flask vorhanden sind, wird dieses auch in diesem Projekt eingesetzt.



Abbildung 23 - Flask Logo
(Quelle: www.flask.palletsprojects.com)

7.3.3.2 Styling

Um den Data-Analyzer zu gestalten und dies gleichzeitig für Smartphones, Tablets und Computer bereitzustellen, wurde Bootstrap ausgewählt. Dieses Open Source Toolkit ermöglicht das schnelle Erstellen von responsiven Webseiten.



Bootstrap

Abbildung 24 - Bootstrap Logo
(Quelle: www.getbootstrap.com)

7.3.3.3 Geodaten

OpenStreetMap ist ein Open-Source Projekt, welches frei nutzbare Geodaten sammelt und in einer Datenbank zu Verfügung stellt.



OpenStreetMap

Abbildung 25 - OSM Logo
(Quelle: www.osm.ch)

7.3.3.4 WebGIS Bibliothek

Um die Karte im Data-Analyzer einzubetten und zu gestalten, wird Leaflet verwendet. Leaflet ist eine Open-Source JavaScript-Bibliothek. Sie verwendet HTML5, CSS3 und eignet sich, um eingebettete Karten responsiv darzustellen und zu gestalten.



Abbildung 26 - Leaflet Logo
(Quelle: www.leafletjs.com)

8 OOD / Design

8.1 State Diagramm

Wird der Kontroller mit Strom versorgt, werden automatisch alle Module und Verbindungen initialisiert. Nach erfolgreicher Initialisierung wird auf einem Prozessor der Webserver gestartet, während auf dem Zweiten in den «Ready» State gewechselt wird. Um dem Benutzer zu signalisieren, dass der Sensor bereit ist, wechselt die LED-Farbe auf grün.

Vom «Ready» State kann entweder der Sensor kalibriert werden, dies wird mit einer blauen LED-Farbe angezeigt, oder eine Aufnahme von Telemetriedaten kann gestartet werden.

Im «Record» State werden kontinuierlich Daten gemessen und sogleich abgespeichert, um Datenverluste zu vermeiden. Nach Beenden einer Aufnahme kehrt man in den «Ready» State zurück.

Tritt in einem State ein Fehler auf wird in den «Error» State gewechselt. Dies wird dem Benutzer mit einer violetten LED-Farbe angezeigt.

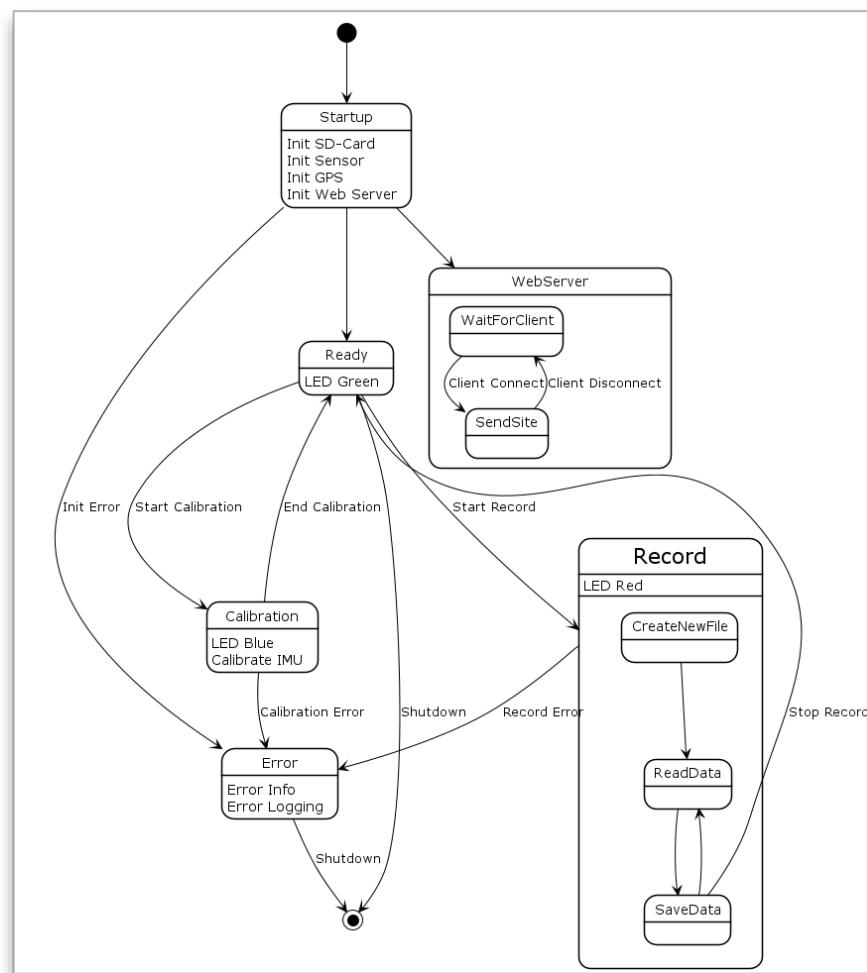


Abbildung 27 - State Diagramm Sensor (Quelle: Eigenes Diagramm «PlantUML»)

8.2 Anwendungsfälle Sensor

Zur Steuerung des Systems während der Motorradtour wird ein minimales Webinterface implementiert. Dabei kann mit dem Smartphone-Browser via WLAN auf den Sensor zugegriffen werden.

Das Webinterface beinhaltet drei verschiedene Funktionen:

1. Kalibrierung:
Um Abweichungen über längere Zeit zu vermeiden, kann der Sensor kalibriert werden. Dies hilft dabei, die gemessenen Daten über eine längere Zeit genau zu halten.
2. Start Recording:
Vor Abfahrt wird die Aufzeichnung via Webinterface gestartet.
3. Stop Recording:
Nach Beendigung der Fahrt kann die Aufzeichnung via Webinterface gestoppt werden.

Die oben genannten Funktionen werden als Anwendungsfälle auf den nächsten Seiten genauer beschrieben.

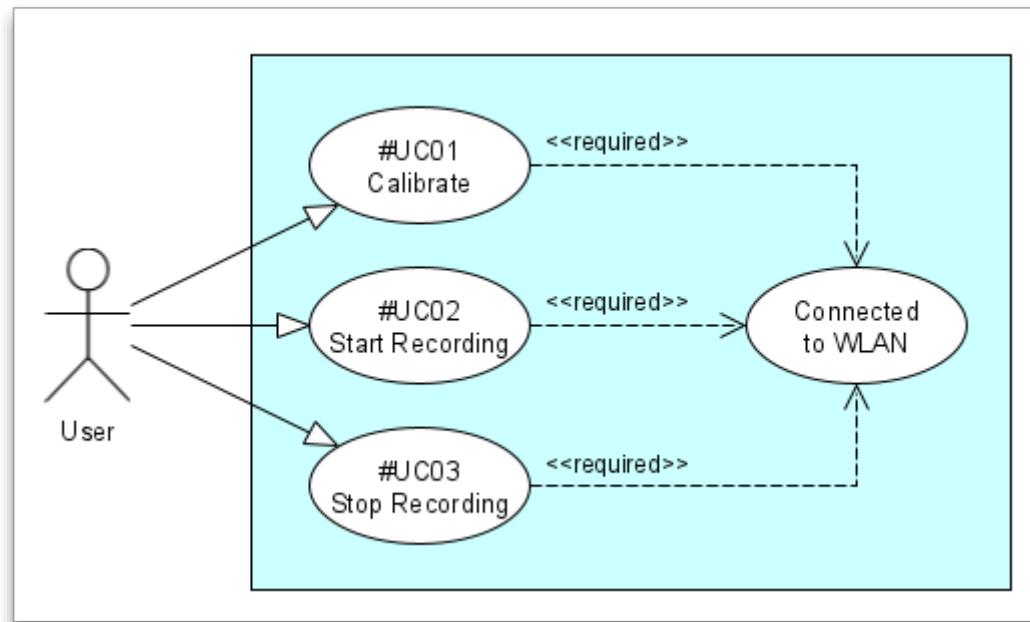


Abbildung 28 - Anwendungsfalldiagramm Sensor (Quelle: Eigenes Diagramm)

8.2.1 Anwendungsfall Sensor #UC01

Beschreibung Anwendungsfall 1: Calibrate	
#ID	#UC01
Name	System kalibrieren
Kurzbeschreibung	Der Benutzer kann das System via Smartphone kalibrieren
Akteur	Benutzer
Ergebnisse	System kalibriert
Vorbedingung	Sensor an Spannungsversorgung angeschlossen
Essenzieller Ablauf	Gemäss Programmablauf-Diagramm

Abbildung 29 - Tabelle Sensor #UC01 (Quelle: Eigene Tabelle)

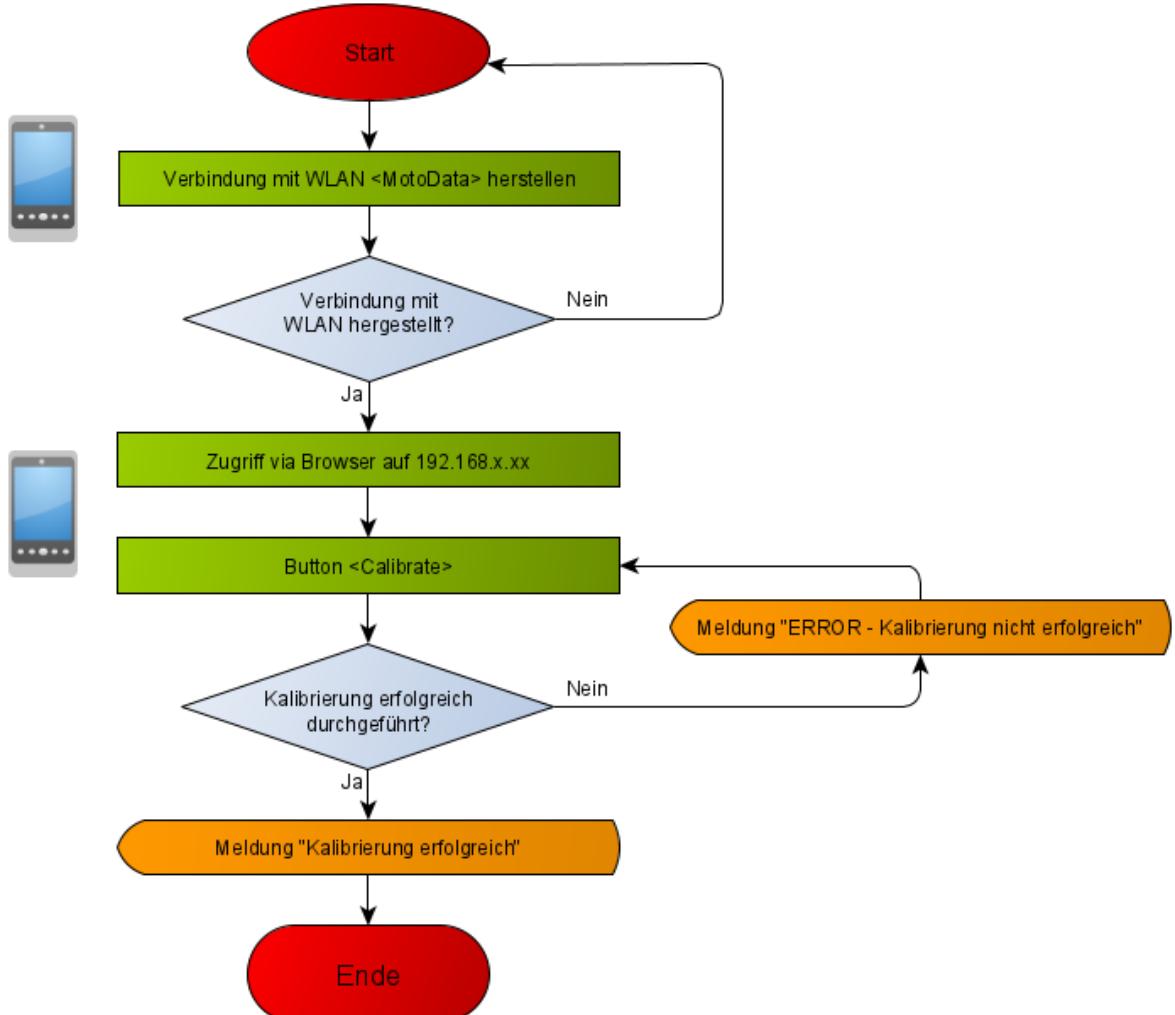


Abbildung 30 - Programmablauf Sensor #UC01 (Quelle: Eigenes Diagramm)

8.2.2 Anwendungsfall Sensor #UC02

Beschreibung Anwendungsfall 2: Start Recording	
#ID	#UC02
Name	Aufzeichnung starten
Kurzbeschreibung	Der Benutzer kann die Aufzeichnung via Smartphone starten
Akteur	Benutzer
Ergebnisse	Aufzeichnung gestartet
Vorbedingung	Sensor an Spannungsversorgung angeschlossen
Essenzieller Ablauf	Gemäss Programmablauf-Diagramm

Abbildung 31 - Tabelle Sensor #UC02 (Quelle: Eigene Tabelle)

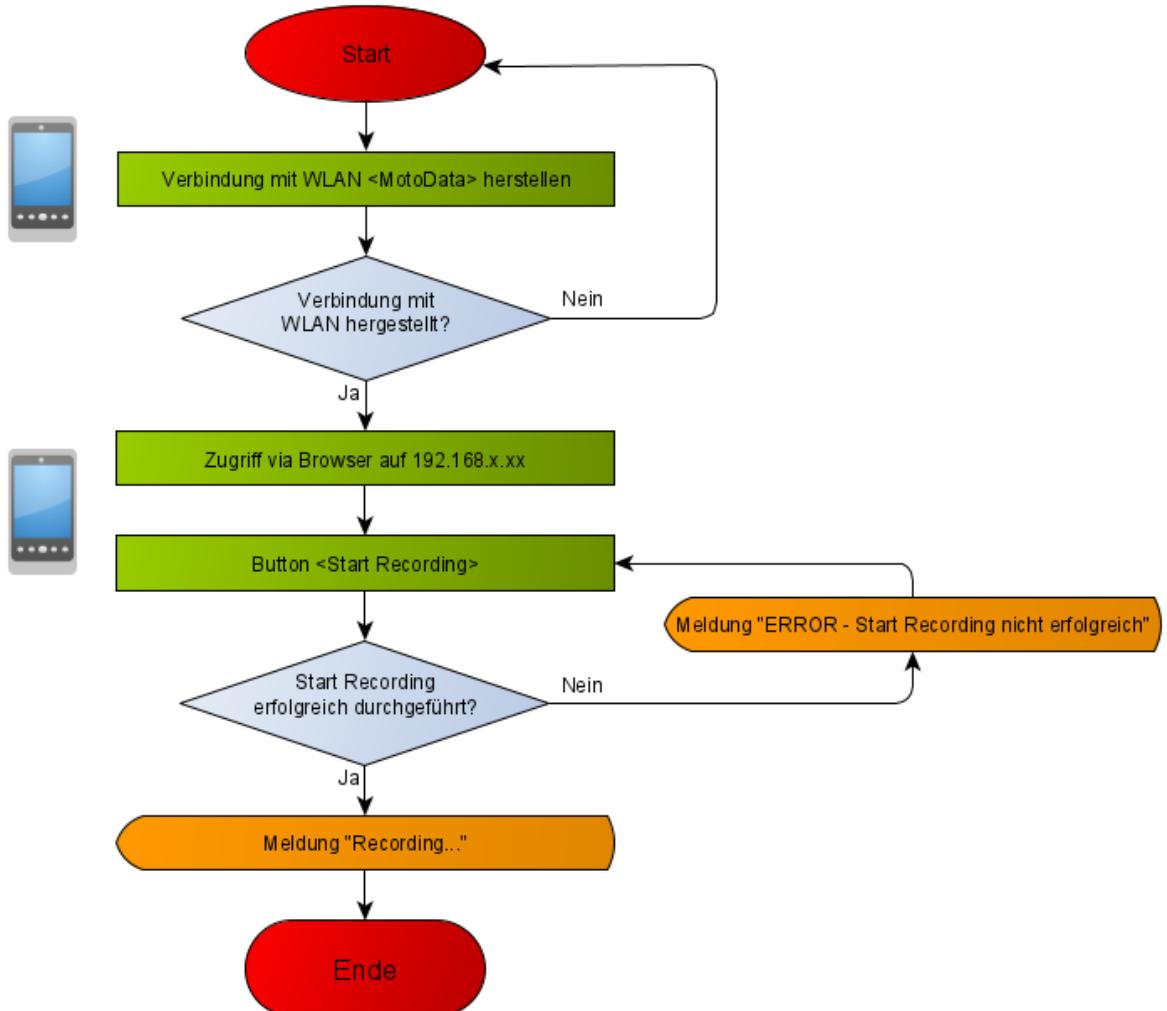


Abbildung 32 - Programmablauf Sensor #UC02 (Quelle: Eigenes Diagramm)

8.2.3 Anwendungsfall Sensor #UC03

Beschreibung Anwendungsfall 3: Stop Recording	
#ID	#UC03
Name	Aufzeichnung stoppen
Kurzbeschreibung	Benutzer kann die Aufzeichnung via Smartphone stoppen
Akteur	Benutzer
Ergebnisse	Aufzeichnung gestoppt / Daten gespeichert
Vorbedingung	«Start Recording» wurde ausgeführt
Essenzieller Ablauf	Gemäss Programmablauf-Diagramm

Abbildung 33 - Tabelle Sensor #UC03 (Quelle: Eigene Tabelle)

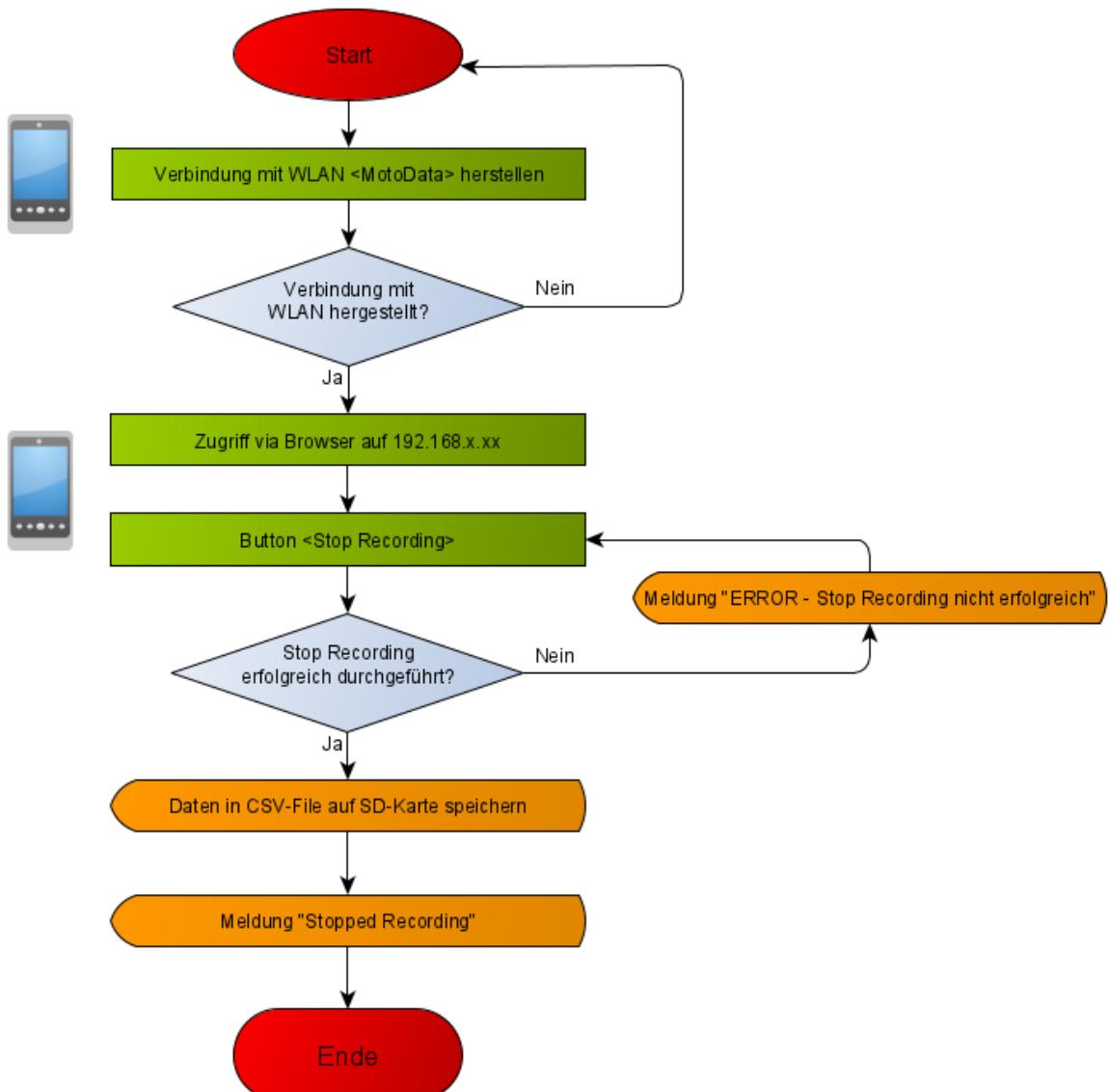


Abbildung 34 - Programmablauf Sensor #UC03 (Quelle: Eigenes Diagramm)

8.3 Anwendungsfälle Data-Analyzer

Der Data-Analyzer beinhaltet vier verschiedene Funktionen:

1. Import File:

Der Sensor speichert die erfassten Messdaten lokal als CSV-File ab. Das File kann via USB-Kabel vom Sensor heruntergeladen werden. Somit kann das File via Data-Analyzer auf den Webserver importiert werden.

2. Delete File:

Ein ausgewähltes CSV-File kann hiermit vom Webserver gelöscht werden.

3. Set Marker on Map:

Dabei wird ein CSV-File ausgewählt und die Messpunkte werden als Marker auf der im Data-Analyzer integrierten Weltkarte angezeigt. Klickt der User auf einen Marker, dann wird diesem die darin enthaltenen Messdaten angezeigt.

4. Reset Map:

Mit der Funktion Reset Map kann die Weltkarte wieder von den angezeigten Markern befreit werden.

Die oben genannten Funktionen werden als Anwendungsfälle auf den nächsten Seiten genauer beschrieben.

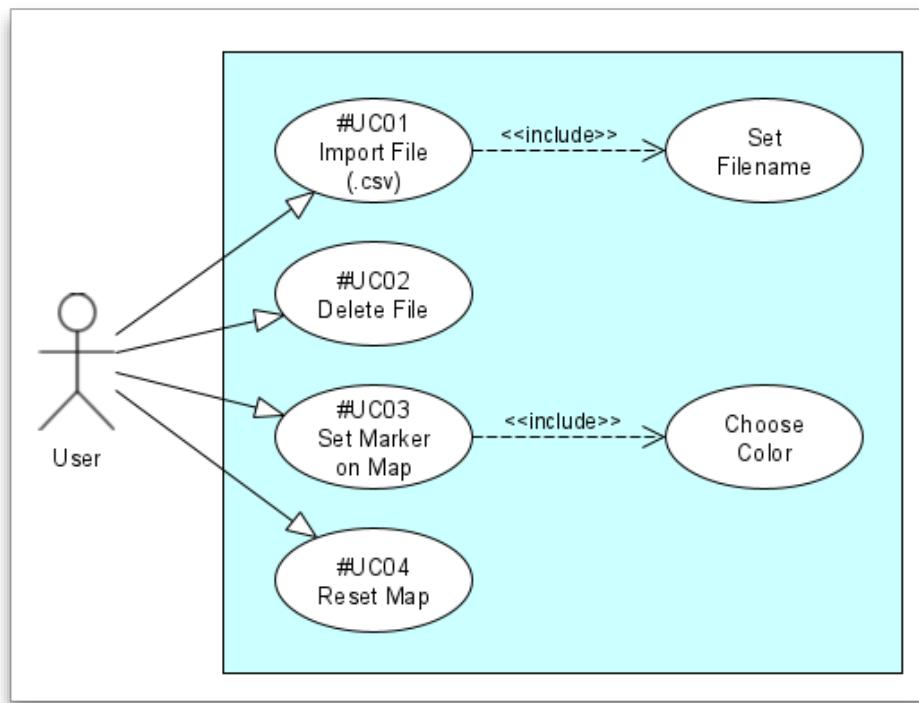


Abbildung 35 - Anwendungsfalldiagramm Data-Analyzer (Quelle: Eigenes Diagramm)

8.3.1 Anwendungsfall Data-Analyzer #UC01

Beschreibung Anwendungsfall 1: Import File	
#ID	#UC01
Name	Daten importieren (.csv)
Kurzbeschreibung	Der Benutzer kann Fahrdaten importieren
Akteur	Benutzer
Ergebnisse	Fahrdaten importiert und gespeichert
Vorbedingung	Fahrdaten (CSV-File) von Sensor exportiert
Nachbedingung	CSV-File auf Server gespeichert
Essenzieller Ablauf	Gemäss Programmablauf-Diagramm

Abbildung 36 - Tabelle Data-Analyzer #UC01 (Quelle: Eigene Tabelle)

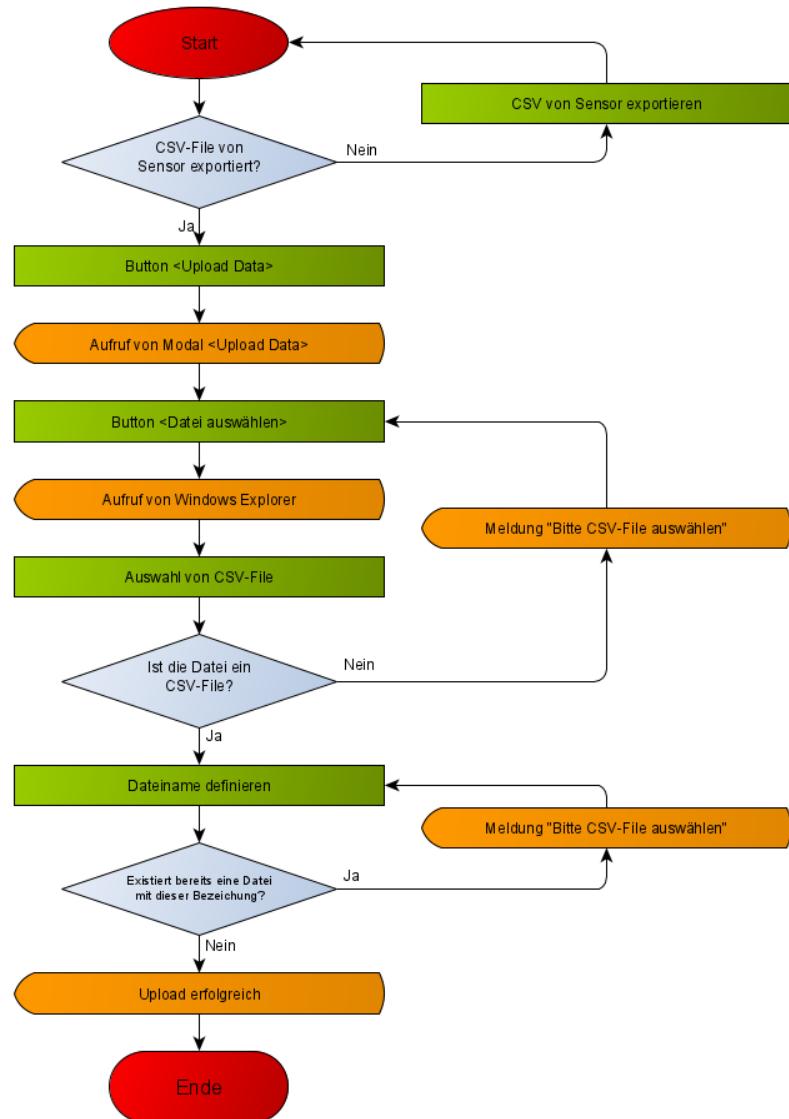


Abbildung 37 - Programmablauf Data-Analyzer #UC01 (Quelle: Eigenes Diagramm)

8.3.2 Anwendungsfall Data-Analyzer #UC02

Beschreibung Anwendungsfall 2: Delete File	
#ID	#UC02
Name	Daten löschen (.csv)
Kurzbeschreibung	Der Benutzer kann Fahrdaten löschen
Akteur	Benutzer
Ergebnisse	Fahrdaten vom Server gelöscht
Vorbedingung	Fahrdaten (CSV-File) auf Server vorhanden
Nachbedingung	CSV-File vom Server gelöscht
Essenzieller Ablauf	Gemäss Programmablauf-Diagramm

Abbildung 38 - Tabelle Data-Analyzer #UC02 (Quelle: Eigene Tabelle)

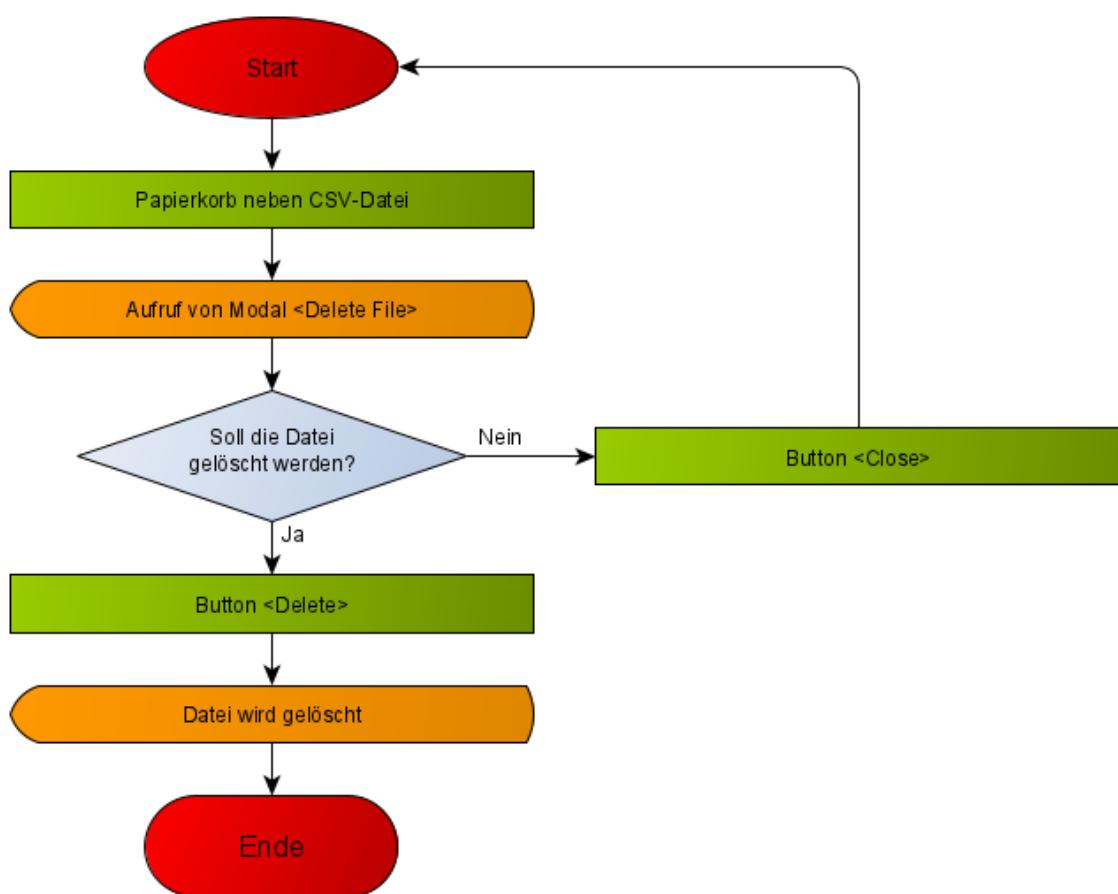


Abbildung 39 - Programmablauf Data-Analyzer #UC02 (Quelle: Eigenes Diagramm)

8.3.3 Anwendungsfall Data-Analyzer #UC03

Beschreibung Anwendungsfall 3: Set Marker on Map	
#ID	#UC03
Name	Marker auf Karte setzen
Kurzbeschreibung	Der Benutzer lässt die Messpunkte in der Karte darstellen
Akteur	Benutzer
Ergebnisse	Messpunkte auf Karte ersichtlich inkl. ausgewählter Farbe
Vorbedingung	Fahrdaten (CSV-File) auf Server vorhanden
Essentieller Ablauf	Gemäss Programmablauf-Diagramm

Abbildung 40 - Tabelle Data-Analyzer #UC03 (Quelle: Eigene Tabelle)

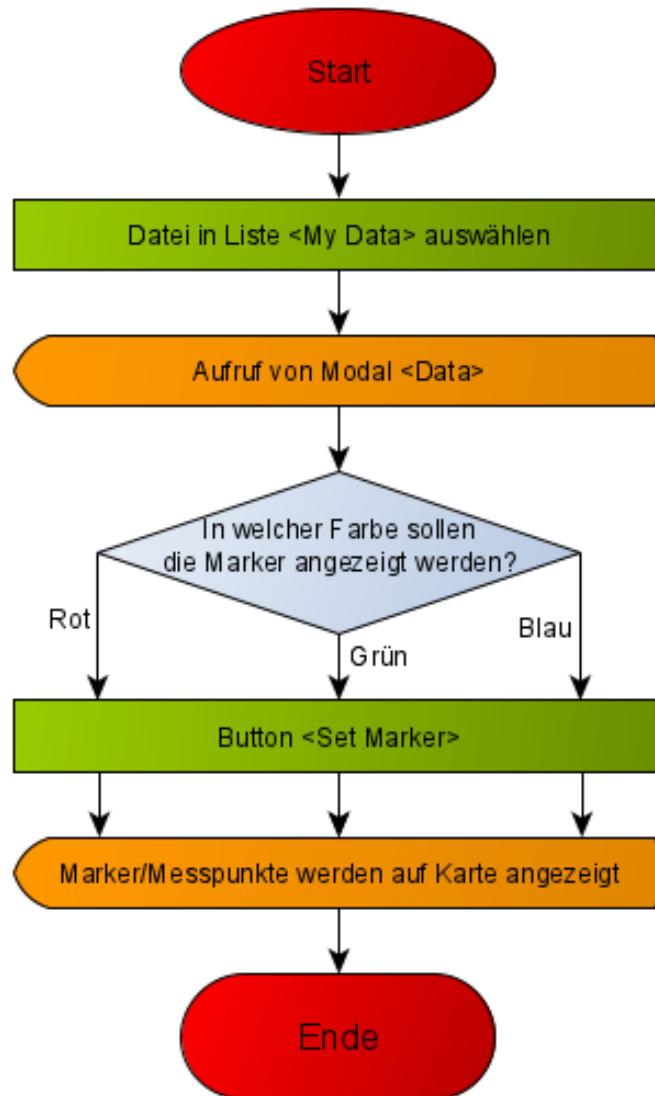


Abbildung 41 - Programmablauf Data-Analyzer #UC03 (Quelle: Eigenes Diagramm)

8.3.4 Anwendungsfall Data-Analyzer #UC04

Beschreibung Anwendungsfall 4: Reset Map	
#ID	#UC04
Name	Karte zurücksetzen
Kurzbeschreibung	Benutzer kann die Karte zurücksetzen (Marker entfernen)
Akteur	Benutzer
Ergebnisse	Es werden keine Marker/Messpunkte mehr angezeigt
Vorbedingung	Marker/Messpunkte auf Karte vorhanden
Essentieller Ablauf	Gemäss Programmablauf-Diagramm

Abbildung 42 - Tabelle Data-Analyzer #UC04 (Quelle: Eigene Tabelle)

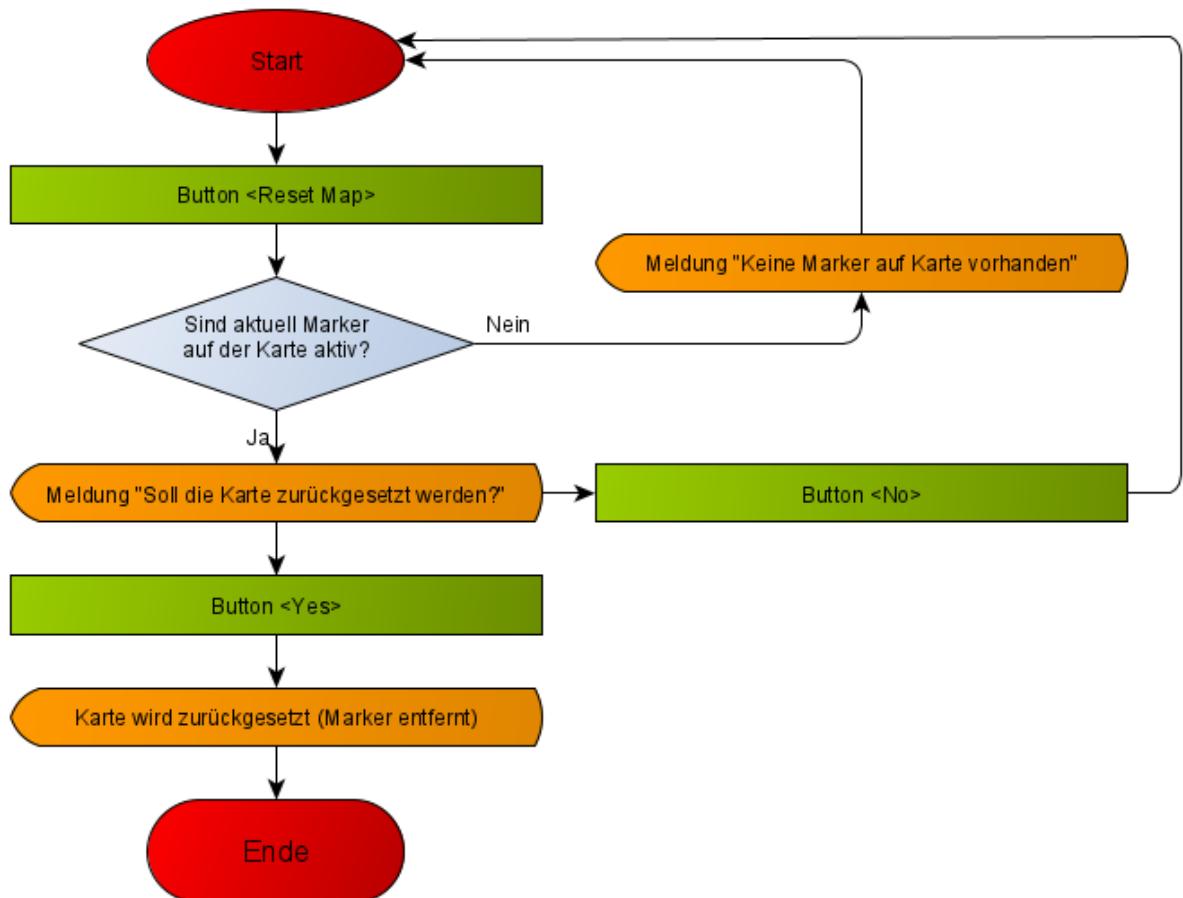


Abbildung 43 - Programmablauf Data-Analyzer #UC04 (Quelle: Eigenes Diagramm)

9 OOP / Implementation

9.1 Sensor – Hardware

9.1.1 Prototyp Breadboard

Um die ersten Versionen der Software zu entwickeln und zu testen, wurde ein Schaltkreis auf einem Breadboard aufgebaut. Ebenfalls war dies ein Testlauf für die Erstellung einer Leiterplatine.

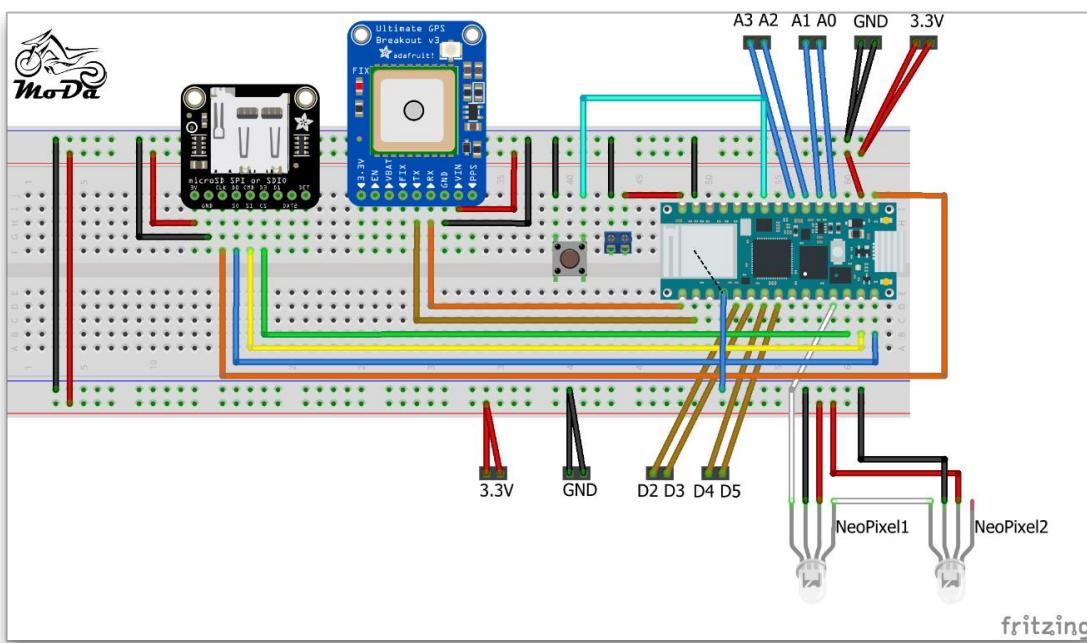


Abbildung 44 - Prototyp Breadboard (Quelle: Eigene Fritzing-Steckansicht)

9.1.2 PCB / Leiterplatine

Um den Betrieb der Schaltung auf einem Motorrad zu ermöglichen, wurde eine Leiterplatine erstellt, die die Komponenten miteinander verbindet.

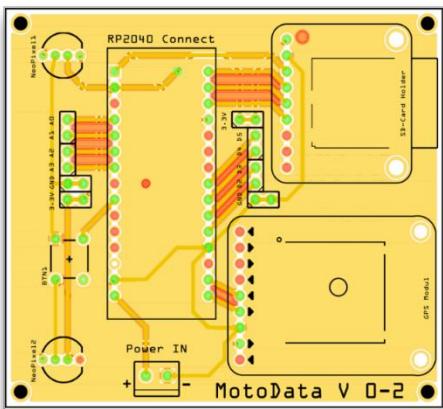


Abbildung 45 – Leiterplatine Vorderseite
(Quelle: Eigener Fritzing-Plan)

Beim Erstellen des Layouts wurde darauf geachtet, dass die Datenverbindungen zwischen den verschiedenen Komponenten möglichst kurz gehalten werden, um Probleme mit Störsignalen zu vermeiden.

Ebenfalls wurde darauf geachtet, dass die Leiterplatine einen möglichst kleinen Formfaktor besitzt. Dies ermöglicht ein einfaches Montieren des Sensors auf dem Motorrad oder anderen Fahrzeugen.

Auch wurden zusätzlich ein Knopf, vier analoge Eingänge, vier digitale Ein/Ausgänge, vier 3.3 V Pins und vier Ground Pins eingeplant, um die Erweiterbarkeit des Sensors zu steigern ohne eine neue Leiterplatine anfertigen zu müssen.

Die Komponenten sind momentan noch mit Stifteleisten auf der Leiterplatine befestigt. Dies ermöglicht es diese für die Test einfach ein- und auszubauen. In der Zukunft werden diese mit der Leiterplatine verlötet, um den Formfaktor noch weiter zu verkleinern.

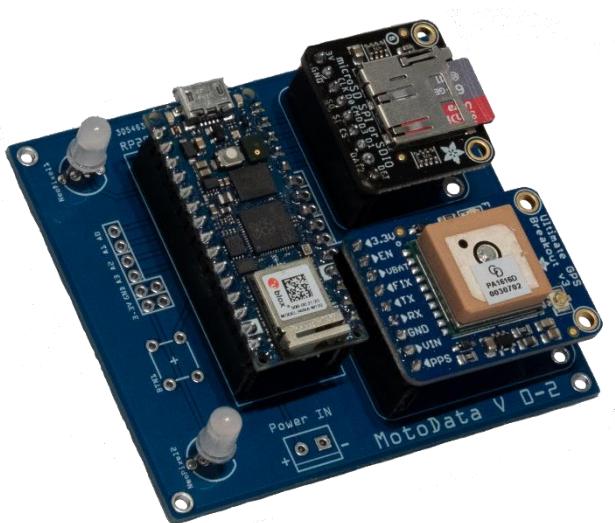


Abbildung 46- Leiterplatine bestückt (Quelle: Eigenes Bild)

9.1.3 Sensorgehäuse

Um den Sensor vor den Elementen zu schützen, wurde ein einfaches Gehäuse konstruiert und mit diversen 3D-Druckverfahren hergestellt.

Das Gehäuse wurde so konzipiert, dass der USB-Port des Mikrocontrollers von aussen zugänglich ist. Dies vereinfacht das Testen im Feld. Dieser Prototyp ist in der frühen Phase nicht spritzwasserfest.

Mithilfe einer Kabeldurchführung, die auf das jeweilige Kabel angepasst wird und einer Deckeldichtung, ist es möglich, das gleiche Gehäuse in einer spritzwasserfesten Konfiguration umzubauen.

Der veränderte Prototyp kann somit leichtem Regen oder dem Wasser beim Durchfahren einer Pfütze standhalten. Das eigentliche Gehäuse wurde mittels einem «FDM» Drucker mit «PETG» gedruckt, da sich dieses Material auch bei höheren Temperaturen nicht von allein verformt und bereits wasserundurchlässig gedruckt werden kann. Die Kabeldurchführung besteht aus demselben Material wie der Grossteil des Gehäuses. Die Dichtung zwischen dem Deckel besteht aus einem weichem «TPU» Kunststoff und gleicht Unebenheiten zwischen dem Gehäuse und dem Deckel aus. Der Deckel wurde mit einem «Resin» Drucker hergestellt. Dieser ermöglicht es ein semitransparentes Bauteil zu erschaffen durch welches man die Status-LEDs des Sensors sehen kann.



Abbildung 47 - Ansicht Gehäuse «Entwicklungsstufe» (Quelle: Eigenes Bild)

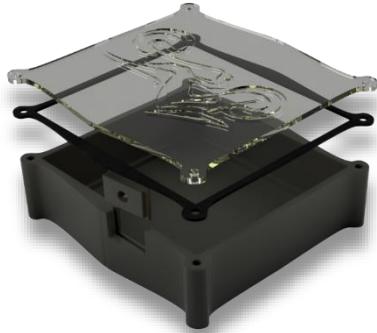


Abbildung 49 - Ansicht Gehäuse «Spritzwasserfest» (Quelle: Eigenes Bild)



Abbildung 48 - Gehäuse mit Elektronik (Quelle: Eigenes Bild)



Abbildung 51 - Gehäuse mit Elektronik (Quelle: Eigenes Bild)



Abbildung 50 - Gehäuse mit Elektronik (Quelle: Eigenes Bild)

9.1.4 Montage

Es wurden bereits Vorbereitungen getroffen, um den Sensor mit verschiedenen Montageplatten, die auf den Boden geschraubt werden können, auszustatten. Dies würde es ermöglichen, den Sensor einfach und ohne Werkzeug auf dem Motorrad zu montieren oder zu entfernen. Zu diesem Zeitpunkt sind jedoch noch keine genaueren Designs ausgearbeitet worden.



Abbildung 52 - Gehäuse montiert auf Motorrad (Quelle:
Eigenes Bild)

In der Abbildung 52 - Gehäuse montiert auf Motorrad (Quelle: Eigenes Bild) ist zu sehen, wie der Sensor im Testgehäuse auf einem Motorrad montiert wurde, um Telemetriedaten erfassen zu können.

Diese Montagemethode funktioniert zwar einwandfrei, ist optisch aber nicht sehr ansprechend und kann nicht einfach montiert oder demontiert werden.

9.2 Sensor – Software

9.2.1 Micropython

Zum Start des Projekts wurde auf das Framework von Arduino gesetzt. Dieses ist weltweit eines der beliebtesten und wird vielfach bei der Programmierung von Microkontroller eingesetzt.

Leider ist dieses Framework noch nicht für Multicore-Microcontroller, wie der «RP2040» ausgelegt. Dies führte zu vielen unerwarteten Fehler. Das fehlende Ressourcenmanaging führte dazu, dass die Cores auf dieselbe Ressource gleichzeitig zugreifen wollten und das System sich dadurch selbst sperrte.

Deshalb wurde auf «MicroPython» gewechselt, welches von Haus aus bereits mehrere Funktionalitäten für Multicore-Microcontroller mitliefert.

9.2.2 Entwickelte Bibliotheken

9.2.2.1 GPS_lib

Es existieren bereits fertige Bibliotheken zum Parsen von GPS-Datenströmen. Diese können jedoch häufiger mehr, als benötigt und führt zu einem Overhead an Funktionalitäten.

Auch setzen die meisten existierenden Bibliotheken voraus, dass die GPS-Daten vollständig sind, daher wurde keine Fehlererkennung eingebaut. Um dies zu verhindern wurde ein eigener GPS-Parser erstellt, der die empfangenen «GPRMC» Nachrichten in die gewünschten Werte umwandelt.

Initialisierung

Um die Funktionen der Bibliothek zu verwenden, muss diese zuerst initialisiert werden.

```
1.     GPS = GPS_lib()
```

Funktionen

Funktion	Beschreibung
GPS.read(GPRMC_MESSAGE)	Diese Funktion prüft die zur Verfügung gestellte «GPRMC» Nachricht auf ihre Vollständigkeit und konvertiert die gelieferten Daten in die gewünschten Werte.

Abbildung 53 - Tabelle Funktionen GPS_lib (Quelle: Eigene Tabelle)

Öffentliche Klassen Variablen

Klassen Variabel	Beschreibung	Daten verlässlich
GPS.time	Aktuelle Zeit (UTC) der GPS-Nachricht. Wird auch geliefert, wenn die Positionsdaten noch nicht valide sind.	Nein
GPS.date	Aktuelles Datum (UTC) der GPS-Nachricht. Wird auch geliefert, wenn die Positionsdaten noch nicht valide sind.	Nein
GPS.fix	Wenn die Positionsdaten valide sind, ist diese Variable wahr.	Ja
GPS.lat	Liefert die Latitude im Format «DDmm.mm»	Ja
GPS.long	Liefert die Longitude im Format «DDmm.mm»	Ja
GPS.speed	Liefert die Geschwindigkeit in km/h	Ja

Abbildung 54 - Tabelle Öffentliche Klassen Variablen GPS_lib (Quelle: Eigene Tabelle)

Struktur GNRMC Nachricht

Diese Nachrichten bestehen aus Datenfelder, welche durch «,» getrennt sind. Jedes dieser Felder ist einem vordefinierten Datentyp zugeordnet.

Beispiel Nachricht die das GPS-Modul liefert:

«\$GNRMC,191759.00,A,4701.95300,N,00815.2780,E,0.35,72.79,121022,,A*41»

Feld	Feld Beschreibung	Format	Beispielwert
1	Nachrichten Typ		\$GNRMC
2	Zeit in UTC	hhmmss.ss	191759.00
3	Positionsstatus	A = Daten valide V = Daten invalide	A
4	Breitengrad	DDmm.mm	4701.95300
5	Breitengrad Richtung	N = Norden S = Süden	N
6	Längengrad	DDDmm.mm	00815.2780
7	Längengrad Richtung	E = Osten W = Westen	E
8	Bodengeschwindigkeit	Knoten	0.35
9	Spurvalidität		72.79
10	Datum	ddmmyy	121022
11	Magnetische Variation	-	-
12	Magnetische Variation O/W	-	-
13	Modus Indikator	A	A
	Checksumme	*hh	*41
	Satzabschluss	-	[CR][LF]

Abbildung 55 - Tabelle Struktur GNRMC Nachricht (Quelle: Eigene Tabelle)

GPS-Nachrichten Verarbeitung

Zuerst wird mithilfe von «Regex» nach dem gesuchten «GNRMC» String gesucht. Ist dieser nicht vorhanden, wird der Fehler zurückgegeben.

```

1.      # -- READ Function (Parsing of the GPS Data)
2.      def read(self, GPSMESSAGE):
3.
4.          GPSMESSAGE = re.search('(\$)(GNRMC).*$', GPSMESSAGE)
```

Konnte der gesuchte String gefunden werden, wird dieser anhand der Trennzeichen aufgeteilt und die jeweiligen Felder den passenden Variablen zugewiesen. Tritt hier ein Fehler auf, wird ebenfalls eine Fehlermeldung zurückgegeben.

```

1.      if(GPSMESSAGE):
2.
3.          GPS_MESG = str(GPSMESSAGE.group(0)).split(",")
4.
5.          try:
6.              if(GPS_MESG[0] == "$GNRMC"):
7.                  self.time = convertTime(GPS_MESG[1])
8.                  self.date = convertDate(GPS_MESG[9])
9.
10.             if(GPS_MESG[2] == "V"):
11.                 self.fix = False
12.
13.
14.             if(GPS_MESG[2] == "A"):
15.                 self.fix = True
16.                 self.lat = GPS_MESG[3]
17.                 self.long = GPS_MESG[5]
18.                 self.speed = float(GPS_MESG[7]) * 1.852
19.
20.         else:
21.             print("No compatible datatype found")
22.             raise GPSNoCompDataType()
23.
24.
25.         except:
26.             raise GPSNoCompDataType()
27.
28.     else:
29.         raise GPSMessageError()
30.         print("GPS Message Error")
```

9.2.2.2 STORAGE_lib

Um das Erstellen von neuen Dateinamen zu erleichtern, wurde eine eigene Bibliothek programmiert, die dies ermöglicht. Diese überprüft alle bereits erfassten Dateinamen und erstellt einen Neuen mit entsprechender Laufnummer.

Initialisierung

Um die Funktionen der Bibliothek zu verwenden, muss diese zuerst initialisiert werden.

```
1. STR = STORAGE_lib()
```

Funktionen

Funktion	Beschreibung
STR.getNewFileName(DIR_LIST)	Diese Funktion erstellt mithilfe einer Liste von Dateien in einem Ordner einen neuen Dateinamen, der noch nicht vorhanden ist und speichert diesen im Objekt ab.
STR.getFilename()	Diese Funktion gibt den erstellten Filenamen des Objekts als String zurück.

9.2.2.3 HTML Seiten Generator

Um nicht für jeden State eine eigene Webseite erstellen zu müssen, wurde mithilfe zweier Funktionen ein Einfaches «Web-Framework» erstellt.

Funktionen

Funktion	Beschreibung
web_content(state)	Diese Funktion erstellt mithilfe des gelieferten State den Inhalt der Webseite und gibt diesen im HTML-Format zurück.
web_page(state)	Diese Funktion erstellt das Grundgerüst einer HTML-Datei und fügt den Inhalt der web_content() Funktion in diesem Gerüst ein. Zudem liefert sie eine fertige HTML-Seite, welche dem User zur Verfügung gestellt werden kann.

9.2.3 MicroPython Bibliotheken

Es wurden diverse Standardbibliotheken von MicroPython verwendet, auf welche hier jedoch nicht weiter eingegangen wird.

9.2.3.1 Winkelmessung

Das IMU oder «Inertial Measurement Unit» wird mit einer Frequenz von ca. 100 Hz abgefragt, um möglichst genaue Winkelwerte berechnen zu können. Auch wird geprüft, ob die Abfrage erfolgreich war oder nicht.

```

1. def readIMU():
2.     global gyroX, gyroY, gyroZ, accelX, accelY, accelZ
3.
4.     try:
5.         accd = IMU.read_accel()
6.         gyd = IMU.read_gyro()
7.
8.         accelX = accd[0]
9.         accelY = accd[1]
10.        accelZ = accd[2]
11.        gyroX = gyd[0]
12.        gyroY = gyd[1]
13.        gyroZ = gyd[2]
14.
15.        return True
16.
17.    except:
18.        return False

```

Danach wird der Winkelwert mittels den Richtungsvektoren des Accelerometers und des Gyroskops berechnet. Die Resultate werden dann in globalen Variablen gespeichert, um diese später mit den GPS-Daten in ein File zu speichern.

```

1. def doCalculations():
2.
3.     global complementaryRoll, complementaryPitch, complementaryYaw
4.
5.     accRoll = math.atan2(accelY, accelZ) * 180 / math.pi
6.     accPitch = math.atan2(-accelX, math.sqrt(accelY * accelY + accelZ * accelZ)) *
180 / math.pi
7.
8.     lastFrequency = float(1000000.0) / lastInterval;
9.
10.    complementaryRoll = complementaryRoll + ((gyroX - gyroDriftX) / lastFrequency)
11.    complementaryPitch = complementaryPitch + ((gyroY - gyroDriftY) /
lastFrequency)
12.
13.    complementaryRoll = 0.98 * complementaryRoll + 0.02 * accRoll
14.    complementaryPitch = 0.98 * complementaryPitch + 0.02 * accPitch

```

Um die Genauigkeit zu verbessern, kann man den Sensor noch kalibrieren. Die Kalibration berechnet die Abweichung der Messwerte über einen definierten Zeitraum bei der Ruheposition. Diese Abweichung wird dann während der Berechnung vom errechneten Wert abgezogen. Dies ist für den eingesetzten Sensor grundsätzlich nicht nötig, da dieser sehr genaue Daten liefert.

Diese Funktionen und mathematische Formeln wurde vom GitHub User «owennewo» übernommen und so umgewandelt, dass diese mit MicroPython und dem eingesetzten IMU funktioniert.

Original Bibliothek für Arduino : https://github.com/owennewo/Arduino_LSM6DS3

9.2.4 States

9.2.4.1 INIT State

Tritt bei der Initialisierung der Hardware ein Fehler auf, wird in den «ERROR» State gewechselt. Falls alle Module und Pins einwandfrei initialisiert werden konnten, wird automatisch in den «READY» State gewechselt.

9.2.4.2 READY State

Ist die Initialisierung abgeschlossen, wird automatisch in den «READY» State gewechselt. Dieser State setzt die LED-Farbe auf Grün, um zu signalisieren, dass der Sensor bereit ist. Danach wartet dieser auf einen State-Wechsel.

```

1.      print("READY")
2.
3.      # - Sets NeoPixels to GREEN
4.      n[0] = (100,0,0)
5.      n[1] = (100,0,0)
6.      n.write()
7.
8.      time.sleep(5)

```

9.2.4.3 INIT_RECORD State

Um eine Aufnahme zu starten, wird der «INIT_RECORD» State aufgerufen. Dieser setzt die LED-Farbe auf Rot und prüft die SD-Karte nach bestehenden Aufnahme Files und erstellt ein Neues mit entsprechender Laufnummer. Gleichzeitig wird der CSV-Header mit den Spaltenbeschriftungen in die Aufnahme Datei geschrieben. Wurde dies erfolgreich abgeschlossen, wechselt der State automatisch auf «RECORD»

```

1.      print("INIT_RECORD")
2.
3.      # - Sets NeoPixels to RED
4.      n[0] = (0,254,0)
5.      n[1] = (0,254,0)
6.      n.write()
7.
8.      # Generate new Filename for the Record
9.      dir_list = os.listdir("/sd")
10.     STR.getNewFileName(dir_list)
11.
12.     # Generates new file with csv Header
13.     try:
14.         f = open("/sd/" + STR.getFilename(), "w")
15.         f.write(DATAHEADER)
16.         f.close()
17.
18.     except OSError:
19.         changeState("ERROR")
20.
21.     changeState("RECORD")

```

Ist die SD-Karte nicht erreichbar, wird in den «ERROR» State gewechselt.

9.2.4.4 RECORD State

Der «RECORD» State ist dazu da, um die Telemetrie Daten der verschiedenen Sensoren auszulesen, umzurechnen und danach in einer Datei abzuspeichern.

Zuerst wird das IMU ausgelesen und die Winkelwerte berechnet.

```

1.     if(readIMU()):
2.         currentTime = time.ticks_us()
3.         lastInterval = currentTime - lastTime # expecting this to be ~100Hz +- 4%
4.         lastTime = currentTime
5.
6.         doCalculations()

```

Danach werden die Daten der seriellen Schnittstelle zum GPS-Modul ausgelesen und in einer Variablen abgespeichert, sofern welche verfügbar sind. Falls die serielle Kommunikation länger als 510 Millisekunden benötigt, wird diese abgebrochen. Dies verhindert ein falsches Auslesen der GPS-Daten.

```

1.     #-- Read Data from Serial if available
2.     nowtime = time.ticks_ms()
3.     while GPSSerial.any():
4.         GPSData += str(GPSSerial.read())
5.
6.     if(time.ticks_ms() - nowtime > 510):
7.         print("GPS Message took to long")
8.         break

```

Sind GPS-Daten in der erwarteten Grösse vorhanden, werden diese von unserer GPS-Bibliothek in die gewünschten Daten umgewandelt und im Objekt gespeichert.

Werden Fehler in der Verarbeitung erkannt, wird dies in einem Error Log erfasst und der Loop wird nicht unterbrochen. Mit Fehler sind nicht komplett oder Format konforme GPS-Daten gemeint.

```

1.     # -- When GPS Data is received it will get processed
2.     if(len(GPSData) < 100 and len(GPSData) > 1):
3.
4.         # -- Parsing GPS Data
5.         try:
6.
7.             GPS.read(GPSData)
8.
9.         except GPSMessageError:
10.             print("GPS Message Error")
11.             errorLog(GPS.time, GPS.date, "GPS Message Error")
12.
13.         except GPSNoCompDataType:
14.             print("GPS Message Error")
15.             errorLog(GPS.time, GPS.date, "no complete GPS Message")

```

Danach wird versucht, eine Datei zu öffnen, die Daten zu schreiben und die Datei anschliessend zu schliessen. Solange die GPS-Daten nicht valide sind, werden keine Positionsdaten abgespeichert.

Ist die SD-Karte nicht erreichbar, wird dies in den Error-Log geschrieben und zum State «ERROR» gewechselt, da dieser Fehler nicht automatisch behoben werden kann.

Ein «MemoryError» hingegen ist nach einem Schreib-Zyklus meistens wieder behoben und wird einfach im Error-Log notiert.

```

1.          # -- File writing
2.          try:
3.              FILE = open("/sd/" + STR.getFilename(), "a")
4.              FILE.write(str(loopCount) + ";" + str(interval) + ";" + )
5.              FILE.write(str(complementaryRoll) + ";" + str(complementaryPitch) + +
6.                  ";")
7.              FILE.write(str(GPS.time) + ";" + str(GPS.date) + ";" +
8.                  str(GPS.fix))
9.              if(GPS.fix):           # Check GPS validity
10.                  FILE.write(";" + str(GPS.lat) + ";" + str(GPS.long) + ";" +
11.                      str(GPS.speed) + "\n")
12.              else:
13.                  FILE.write("\n")
14.              FILE.close()
15.          except OSError as e:
16.              print("OSError:" + str(e))
17.              errorLog(GPS.time, GPS.date, "OSError: File Write" + str(e))
18.              changeState("ERROR")
19.
20.          except MemoryError as e:
21.              print(e)
22.              errorLog(GPS.time, GPS.date, "MemoryError: " + str(e))

```

Zu Testzwecken können die erhaltenen GPS-Daten und die errechneten Werte auch direkt in der Konsole ausgegeben werden. Diese werden für den Betrieb deaktiviert, um die Performance nicht unnötig zu strapazieren.

```

1.          # -- Debug Print
2.          print(GPSData)
3.          print(str(loopCount) + ";" + str(interval) + ";" + ....)

```

9.2.4.5 STOP_RECORD State

Dieser State wechselt nur zurück zum «READY» State und ist dazu da, dass der letzte Schreibzyklus des «RECORD» States abgeschlossen wird.

9.2.4.6 CALIBRATION State

In diesem State wird die Abweichung des IMU-Sensors über einen Zeitraum von 10 Sekunden gemessen, um die Messungen genauer zu machen. Dies ist im Kapitel [9.2.3.1 Winkelmessung](#) genauer beschrieben. Um anzugeben, dass sich der Sensor im «CALIBRATION» State befindet, wird die LED-Farbe auf Blau geändert.

```
1.      print("--- CALIBRATE ---")
2.      n[0] = (0,0,254)
3.      n[1] = (0,0,254)
4.      n.write()
5.
6.      calibrateIMU(1000, 5000)
7.
8.      print("-----")
9.      print("Gyro Drift")
10.     print("x", gyroDriftX)
11.     print("y", gyroDriftY)
12.     print("z", gyroDriftZ)
13.     print("-----")
14.
15.     time.sleep(2)
16.
17.     changeState("READY")
```

9.2.4.7 ERROR State

Tritt in der Software ein Fehler auf, der nicht vernachlässigbar ist, wird in den «ERROR» State gewechselt. Dieser ändert die LED-Farbe auf Violett und der Sensor bleibt nun in diesem State bis dieser von der Spannungsquelle getrennt und wieder verbunden worden ist.

```
1.      print("--- ERROR ---")
2.
3.      # - Sets NeoPixels to PURPLE
4.      n[0] = (254,0,254)
5.      n[1] = (254,0,254)
6.      n.write()
7.
8.      while True:
9.          time.sleep(200)
```

9.2.5 Webinterface Sensor

Das Design des Webinterface für den Sensor wurde zweckgemäss und benutzerfreundlich gestaltet, um die Bedienung so einfach wie möglich zu gestalten.

9.2.5.1 Startseite

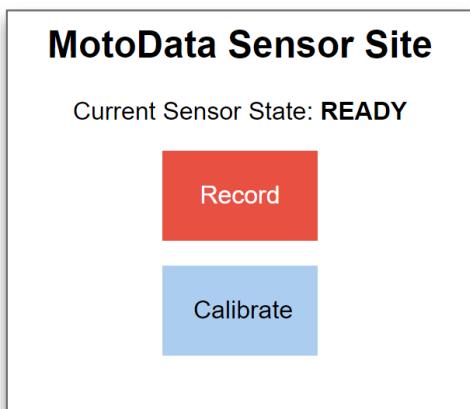


Abbildung 56 - Sensor Webinterface "Startseite"
(Quelle: Eigenes Bild)

Auf der Startseite wird der aktuelle Sensor State angezeigt. Zeigt dieser «READY», kann einer der zwei Buttons gewählt werden. Mit einem Knopfdruck wird der State des Sensors gewechselt und dessen Funktionen ausgeführt.

Button <Record>:
Der State «INIT_RECORD» wird ausgelöst und die Seite [9.2.5.3 Start Recording](#) wird aufgerufen.

Button <Calibrate>:
Der State «CALIBRATE» wird ausgelöst und die Seite [9.2.5.2 Kalibrierung](#) wird aufgerufen.

9.2.5.2 Kalibrierung

Während der Kalibrierung des Sensors wird diese Nachricht angezeigt. Sobald die Kalibrierung abgeschlossen ist, wird wieder auf die Startseite gewechselt.

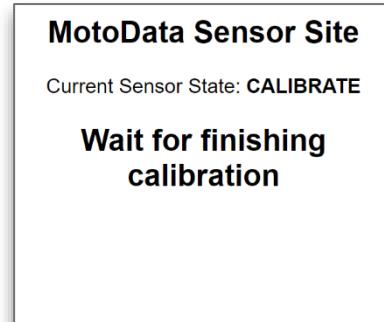
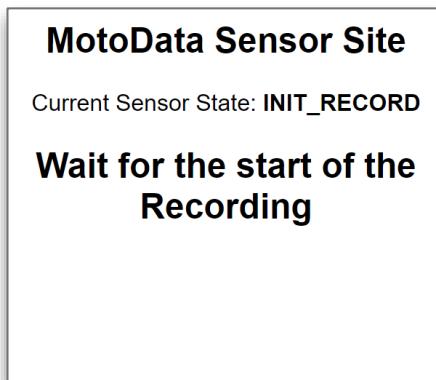


Abbildung 57 - Sensor Webinterface "Kalibration Seite" (Quelle: Eigenes Bild)

9.2.5.3 Start Recording



Wurde der State «INIT_RECORD» ausgelöst, wird eine Seite angezeigt, währenddessen die Initialisierung des Aufnahmeprozesses stattfindet.

Abbildung 58 - Sensor Webinterface "Init Record"
(Quelle: Eigenes Bild)

9.2.5.4 Recording

Sobald die Aufnahme von Daten begonnen hat, ist es nun möglich, diese wieder zu stoppen. Mit einem Knopfdruck auf den Button <Record Stop> wird der State am Sensor zu «STOP_RECORD» gewechselt.

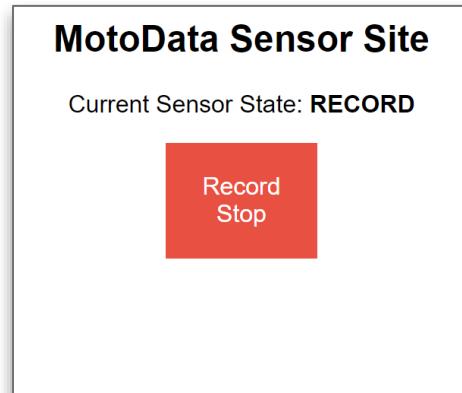


Abbildung 59 - Sensor Webinterface "Record"
(Quelle: Eigenes Bild)



9.2.5.5 Stop Recording

Wurde der State «STOP_RECORD» ausgelöst, wird die Aufnahme der Telemetrie-Daten beendet.

Abbildung 60 - Sensor Webinterface "Stop Record"
(Quelle: Eigenes Bild)

9.2.5.6 Fehlermeldung



Abbildung 61 - Sensor Webinterface "Error" (Quelle:
Eigenes Bild)

9.3 Schnittstelle Sensor / Data-Analyzer

9.3.1 Sensordaten speichern / CSV-File

Die erfassten Messdaten werden vom Sensorsystem als CSV-Datei gespeichert. Diese CSV-Datei kann im Anschluss exportiert werden und via Data-Analyzer eingesehen werden.

In CSV-File werden folgende Spalten erfasst:

1. Loopcount: Einfacher Zähler
2. Interval: Messabstände in Millisekunden
3. Roll: Neigungswinkel (Schräglage Motorrad)
4. Pitch: Neigungswinkel (Wheelie / Endo)
5. Time: Zeitpunkt der Erfassung der Messdaten
6. Date: Datum der Erfassung der Messdaten
7. Fix: GPS-Fix (True bei Verbindung / False bei keiner Verbindung)
8. Latitude: Geografischer Breitengrad in DMS-Form
9. Longitude: Geografischer Längengrad in DMS-Form
10. Speed: Geschwindigkeit in km/h

Loop-count	Interval	Roll	Pitch	Time	Date	Fix	Latitude	Longitude	Speed
1	202	0.2063947	0.5347882	13:27:28.80	15.10.2022	True	4706.6608	826.8672	0
2	200	0.5904114	0.2637805	13:27:29.00	15.10.2022	True	4706.6608	826.8672	0.0926
3	202	-0.871249	1.021008	13:27:29.20	15.10.2022	True	4706.6609	826.8672	0.22224
4	201	0.0314133	1.649578	13:27:29.40	15.10.2022	True	4706.6609	826.8671	0.40744
5	201	1.509643	2.815764	13:27:29.60	15.10.2022	True	4706.6612	826.867	2.2224
6	196	3.668144	3.224023	13:27:29.80	15.10.2022	True	4706.6612	826.8668	6.2042
7	200	3.182706	3.733271	13:27:30.00	15.10.2022	True	4706.6613	826.8667	9.167399
8	202	0.3402784	4.300135	13:27:30.20	15.10.2022	True	4706.6615	826.8665	10.90828
9	197	-4.227105	2.510388	13:27:30.40	15.10.2022	True	4706.6618	826.8662	12.8714
10	201	-9.980918	2.594194	13:27:30.60	15.10.2022	True	4706.6621	826.8659	14.6308
11	199	-12.18678	4.29843	13:27:30.80	15.10.2022	True	4706.6624	826.8655	15.9272
12	201	-13.07209	4.518292	13:27:31.00	15.10.2022	True	4706.6627	826.865	17.85328
13	201	-17.49908	6.710894	13:27:31.20	15.10.2022	True	4706.6632	826.8645	19.37192
14	200	-14.13665	7.290941	13:27:31.40	15.10.2022	True	4706.6637	826.864	21.77952
15	197	-5.927431	5.866784	13:27:31.60	15.10.2022	True	4706.6643	826.8635	23.57596
16	202	-1.986884	6.048906	13:27:31.80	15.10.2022	True	4706.6651	826.863	26.13172
17	207	4.387053	4.516179	13:27:32.00	15.10.2022	True	4706.6659	826.8625	28.29856
18	195	5.680632	5.653797	13:27:32.20	15.10.2022	True	4706.6668	826.8621	30.81728
19	198	7.153287	5.914991	13:27:32.40	15.10.2022	True	4706.6678	826.8617	33.53972
20	199	5.726544	8.641821	13:27:32.60	15.10.2022	True	4706.6689	826.8613	35.81768

Abbildung 62 - Tabelle CSV-File (Quelle: Eigene Tabelle)

9.4 Data-Analyzer

In diesem Kapitel werden die Funktionen des Data-Analyzer genauer beschrieben.

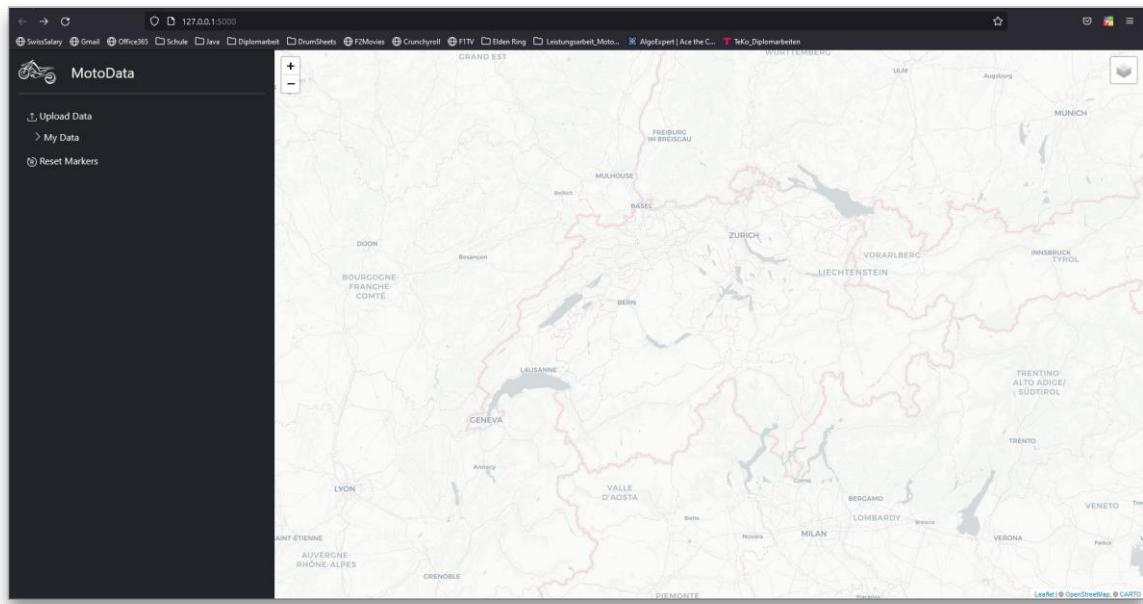
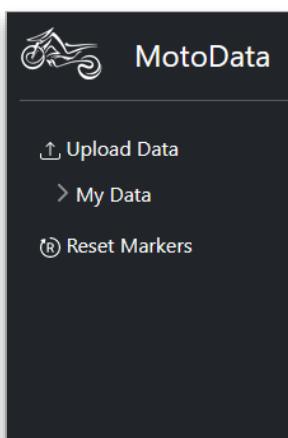


Abbildung 63 - Data-Analyzer (Quelle: Eigenes Bild)

9.4.1 Sidebar



Die Sidebar wurde sehr einfach gehalten, damit eine übersichtliche intuitive Bedienung möglich ist.

Zur Auswahl stehen drei Buttons:

- Upload Data
- My Data
- Reset Markers

Die Funktionalität dieser Buttons wird auf den nächsten Seiten genauer beschrieben.

Abbildung 64 - Sidebar
(Quelle: Eigenes Bild)

9.4.2 Upload Data

Mithilfe des Sidebar-Button <Upload Data> wird das Modal Upload Data aufgerufen. Mit Klick auf den Button <Browse...> kann via Explorer die gewünschte CSV-Datei ausgewählt werden.

Nun wird im Eingabefeld der gewünschte Dateiname ergänzt und mit dem Button <Upload> das CSV-File hinzugefügt.

Die hochgeladene Datei wird nun in der Sidebar unter dem Punkt <My Data> angezeigt.

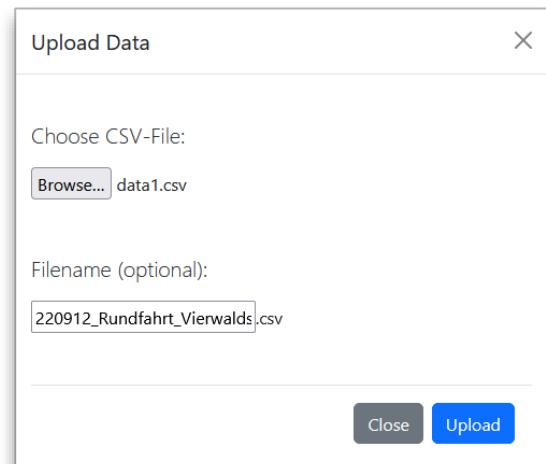


Abbildung 65 - Modal Upload Data
(Quelle: Eigenes Bild)

9.4.3 Delete Data

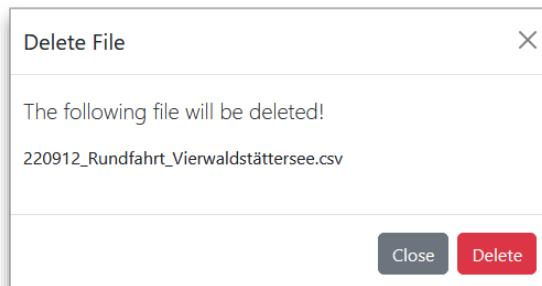


Abbildung 66 - Modal Delete File
(Quelle: Eigenes Bild)

Wird der Sidebar-Button zum Löschen gewählt, so öffnet sich das Modal <Delete File>. Hier kann das Löschen der Datei nun bestätigt oder abgebrochen werden.

9.4.4 Set Marker on Map

Um die hochgeladenen Messdaten nun auf der topographischen Karte anzeigen zu lassen, wird die gewünschte Datei in der Sidebar angeklickt. Dabei öffnet sich das Modal <Set Marker>.

Damit verschiedene Fahrtrouten miteinander verglichen werden können, kann hier eine Farbe gewählt werden.

Die Messdaten werden als Marker mit der gewünschten Farbe auf der Karte dargestellt.

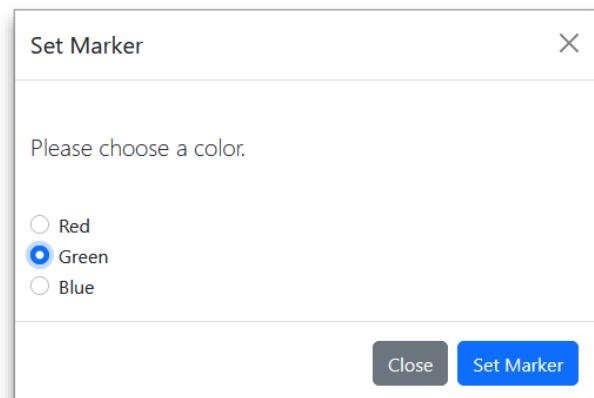


Abbildung 67 - Modal Set Marker
(Quelle: Eigenes Bild)

9.4.5 Reset Map

Um die gesetzten Messpunkte/Marker von der Weltkarte zu entfernen, kann der Button <Reset Markers> in der Sidebar ausgewählt werden. Somit werden alle angezeigten Marker entfernt. Die gespeicherten CSV-Files bleiben jedoch bestehen.



Abbildung 68 - Reset Marker
(Quelle: Eigenes Bild)

9.4.6 Datenauswertung

Wurden die Marker in der Weltkarte gesetzt, so können diese angewählt werden, wobei sich ein Popup des jeweiligen Markers öffnet.

In diesem Popup werden folgende Daten angezeigt:

1. Filename: Altdorf_PARU → Der Dateiname der CSV-Datei
2. Date: 15/10/22 → Datum der Erfassung der Messdaten
3. Time: 13:34:50.60 → Zeitpunkt der Erfassung der Messdaten
4. Speed: 38.9846 km/h → Geschwindigkeit in km/h
5. Roll: -19.03424° → Neigungswinkel (Schräglage Motorrad)

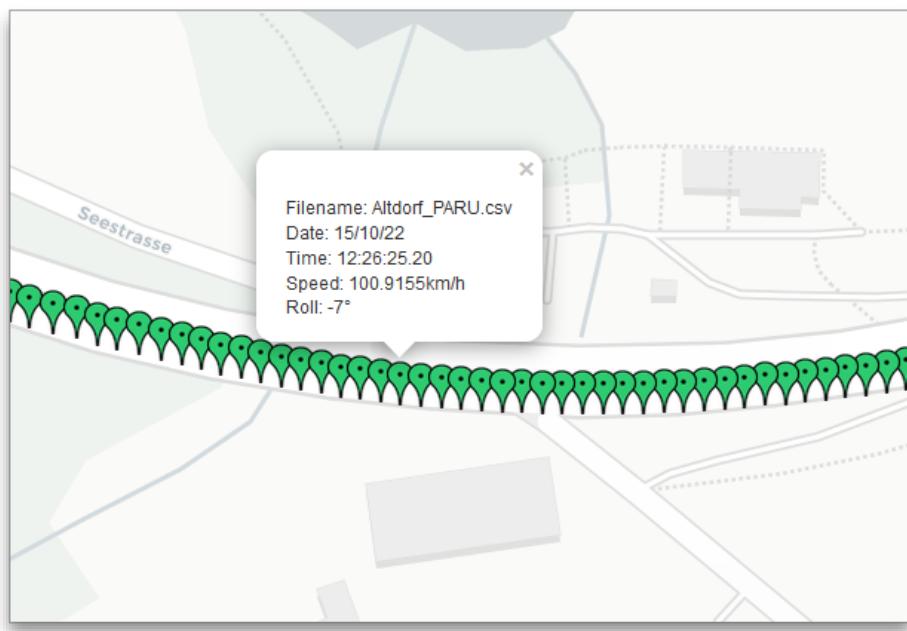


Abbildung 69 - Datenauswertung (Quelle: Eigenes Bild)

9.4.7 Datenfluss

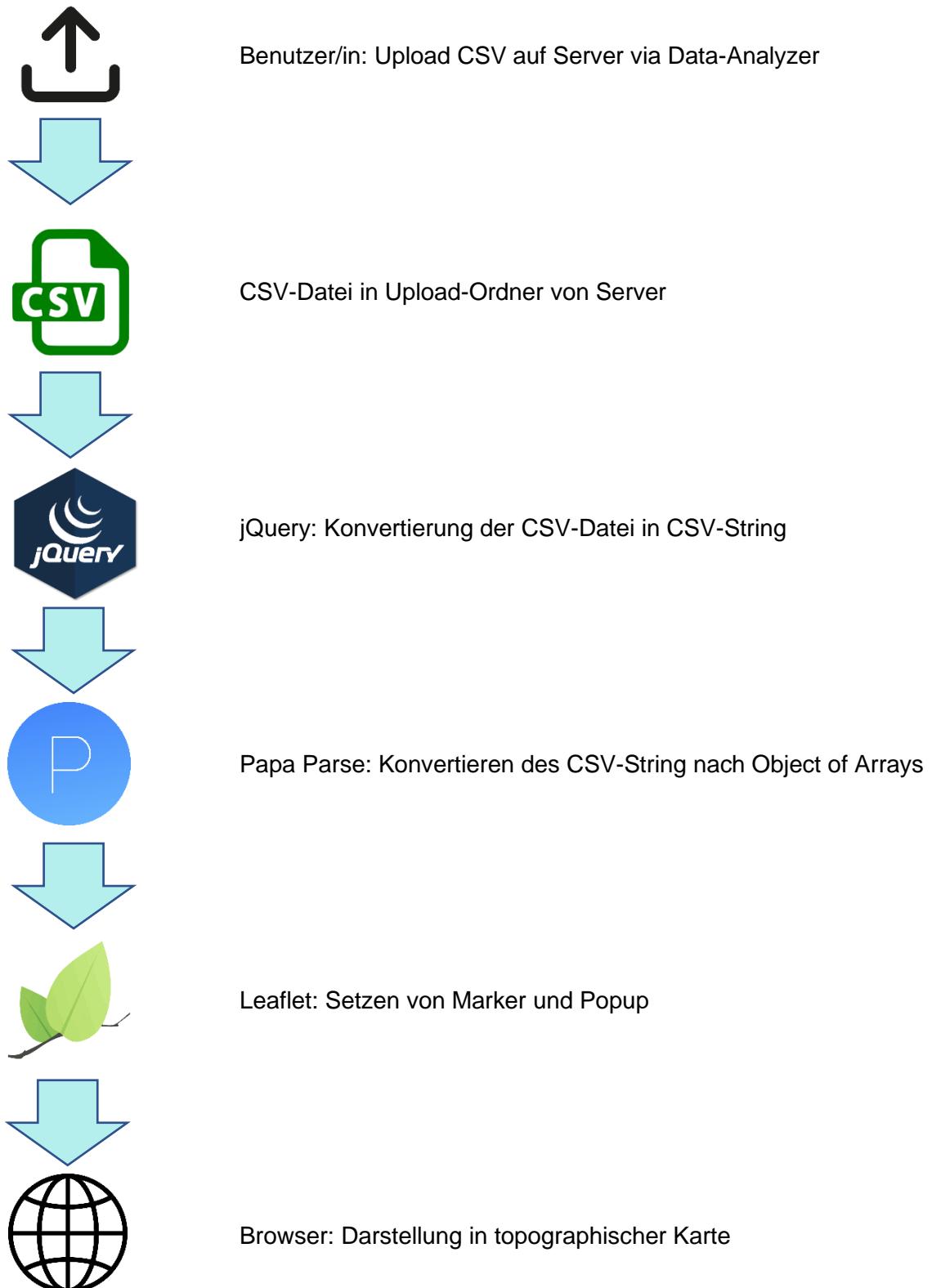


Abbildung 70 - Datenfluss Data-Analyzer (Quelle: Eigene Grafik)

9.4.8 Leaflet

9.4.8.1 Umrechnung LON / LAT

Das Sensorsystem speichert die Standortdaten im CSV-File als DMS-Format (Degrees-Minutes-Seconds).

Lat	Latitude (DDmm.mm)	IIII.II	5107.0017739
Lon	Longitude (DDDmm.mm)	YYYYYY	11402.3291611

Abbildung 71 - Standortdatenformat DMS (Quelle: Eigene Tabelle)

Die verwendete Web-GIS-Bibliothek Leaflet arbeitet mit Standortdaten im Dezimalformat. Daher müssen die Daten mittels Data-Analyzer konvertiert werden.

```
// -- Converts DMS coordinates to Decimal
function convertDMStoD(DMS) {

    result = 0.00000
    wholeDegrees = Math.trunc(0.01 * DMS);
    result = wholeDegrees + (DMS - 100.0 * wholeDegrees) / 60.0;
    return result
}
```

9.4.8.2 Markercluster

Um die Performance des Data-Analyzers zu verbessern, werden sogenannte Cluster eingesetzt. Dadurch werden die Marker bei weitem Zoom nicht einzeln dargestellt, sondern als Cluster. Die Zahl zeigt, wie viele Marker an diesen Punkten vorhanden sind.

```
1. // -- Init MarkerCluster
2. var mcg = L.markerClusterGroup({
3.   chunkedLoading: true,
4.   //spiderfyOnMaxZoom: true
5.   disableClusteringAtZoom: 16
6. });
```

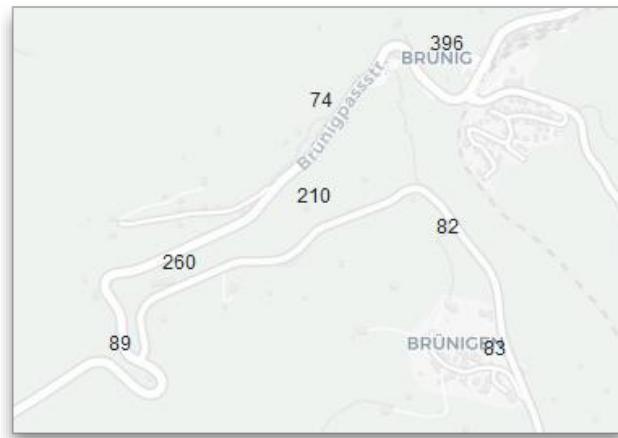


Abbildung 72 - Markercluster (Quelle: Eigene Abbildung)

9.4.8.3 Set Markers

In folgendem Quellcode wird das Parsing der CSV-Datei durchgeführt, das Popup-Icon initialisiert, der Popup-Content definiert, die Standortdaten ins Dezimalformat konvertiert und die Marker inkl. Popup via Leaflet-Library auf der topographischen Karte gesetzt:

```

1.  // -- Sets Markers on the Map
2.  {% for s in mapsettings.settings %}
3.
4.    // Read markers data from CSV
5.    $.get("{{ url_for('static', filename='upload/') }}" + "{{ s.filename }}", function
6.      (csvString) {
7.
8.        var mapIcon = L.icon({
9.          iconUrl:
10.            'http://chart.apis.google.com/chart?chst=d_map_pin_letter&chld=%E2%80%A2|' +
11.              '{{s.color}}' + '&chf=a,s,ee00FFFF',
12.              iconSize: [20, 30], // size of the icon
13.              popupAnchor: [0, -15]
14.        });
15.
16.        // Use PapaParse to convert string to array of objects
17.        var data = Papa.parse(csvString, { header: true, dynamicTyping: true }).data;
18.
19.        for (var i in data) {
20.          var row = data[i];
21.          var markerPopup = "<br/>Filename: " + "{{ s.filename }}" + "<br/>Date: " +
22.            row.Date + "<br/>Time: " + row.Time + "<br/>Speed: " + row.Speed + "km/h<br/>Roll:
23.            " + parseInt(row.Roll) + "°<br/>";
24.
25.          try {
26.            var marker = L.marker([convertDMStoD(row.Latitude),
27.              convertDMStoD(row.Longitude)], {
28.                icon: mapIcon
29.
30.              }).bindPopup(markerPopup, customOptions);
31.
32.            mcg.addLayer(marker);
33.          } catch (error) {
34.            console.error(error);
35.            console.error(row);
36.          }
37.        }
38.      map.addLayer(mcg);
39.    });
40.
41.  {% endfor %}
42.
```

Ist die CSV-Datei falsch formatiert, wird die Darstellung der Messpunkte versucht, jedoch wird eine Fehlermeldung in der Konsole ausgegeben.

Sollte die CSV-Datei zwar richtig formatiert sein, aber es fehlen Datenpunkte, werden diese Messpunkte übersprungen.

10 Testbericht / Auswertung

10.1 Testspezifikation

Testumgebung

Die Testumgebung wurde wie folgt festgelegt:

- Sensor
- Data-Analyzer

Testkategorien

Die Tests wurden in folgende Kategorien untergeordnet:

1. Normale Tests: Testen von zu erwartenden Anwendungsfällen
2. Subnormale Tests: Risiken einbeziehen (Tunnel, Wetter, usw.)
3. Extreme Tests: Kaputte SD-Karte, Überspannungen, usw

Ampelsystem

Die durchgeföhrten Tests werden mit folgendem Ampelsystem durchgeföhrts:

- ↑ Test erfolgreich: Keine weiteren Test-Runs notwendig.
- Test teilweise erfolgreich: Testresultat entspricht nicht den Erwartungen.
Jedoch nicht zwingend weitere Test-Runs notwendig.
- ↓ Test fehlgeschlagen: Es sind weitere Test-Runs notwendig.

10.2 Sensor: Normale Tests

Test-Run #1

In diesen Test-Cases wird das System mit praktisch orientierten Tests geprüft.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
#S1.1	Überprüfung Genauigkeit Standortdaten	+/- 5m unter freiem Himmel	Im Vergleich mit zwei Smartphones (Samsung, Realme) und einem GPS-Navigationssystem befanden sich alle 4 Geräte im gewünschten Bereich.	↑
#S2.1	Überprüfung Genauigkeit Geschwindigkeitsmessung	50 km/h (+/- 5 km/h)	Im Vergleich mit zwei Smartphones (Samsung, Realme) befand sich der Sensor immer im gewünschten Bereich	↑
#S3.1	Überprüfung Genauigkeit Neigungswinkel→Geradeaus	0° (+/- 5°)	Im Vergleich mit einem elektronischen Winkelmesser befand sich der Sensor immer im gewünschten Bereich.	↑
#S4.1	Überprüfung Genauigkeit Neigungswinkel→Kurve	0° (+/- 5°)	Beim mehrmaligen Fahren einer Kurve mit konstanter Geschwindigkeit produzierte der Sensor Messwerte im gewünschten Toleranzbereich	↑

Abbildung 73 - Sensor: Normale Tests | Tabelle Test-Run #1 (Quelle: Eigene Tabelle)

Change Log

Im Change log werden allfällige Änderungen und Lösungsversuche festgehalten.

#ID	Datum	Änderung
-	-	-

Abbildung 74 - Sensor: Normale Tests | Tabelle Change Log (Quelle: Eigene Tabelle)

Test-Run #2

Nach erfolgtem Lösungsversuch werden die fehlgeschlagenen Tests wiederholt.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
-	-	-	-	-

Abbildung 75 - Sensor: Normale Tests | Tabelle Test-Run #2 (Quelle: Eigene Tabelle)

10.3 Sensor: Subnormale Tests

Test-Run #1

In diesen Test-Cases wird das System mit praktisch orientierten Tests geprüft.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
#S1.1	Testfahrt durch Wald	Aussetzer in den GPS-Daten oder falsche Positionsdaten	Aussetzer in den GPS-Daten oder falsche Positionsdaten	→
#S2.1	Testfahrt durch Tunnel	Aussetzer in den GPS-Daten mit anschliessender neuer Positionsfindung	Daten unterbrechen sauber, wenn kein GPS-Signal mehr vorhanden ist. Nach der Positionsfindung sind Daten für bis zu 20 Sekunden ungenau.	→
#S3.1	Testfahrt auf Pässen	Durchgehende GPS-Daten Erhebung	Ungenauer Positionsdaten bei Nähe zu steilen Felswänden, Objekten	→

Abbildung 76 - Sensor: Subnormale Tests | Tabelle Test-Run #1 (Quelle: Eigene Tabelle)

Change Log

Im Change log werden allfällige Änderungen und Lösungsversuche festgehalten.

#ID	Datum	Änderung
-	-	-

Abbildung 77 - Sensor: Subnormale Tests | Tabelle Change Log (Quelle: Eigene Tabelle)

Test-Run #2

Nach erfolgtem Lösungsversuch werden die fehlgeschlagenen Tests wiederholt.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
-	-	-	-	-

Abbildung 78 - Sensor: Subnormale Tests | Tabelle Test-Run #2 (Quelle: Eigene Tabelle)

10.4 Sensor: Extreme Tests

Test-Run #1

In diesen Test-Cases wird das System mit praktisch orientierten Tests geprüft.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
#S1.1	Sensor verkehrt montiert (Fahrtrichtung)	Verkehrte Messwerte	Verkehrte Messwerte	↑
#S2.1	Witterung	Keine Beeinträchtigung	Bis auf ungenauere Positionsdaten konnte keine Beeinträchtigung festgestellt werden	↑
#S3.1	Vibrationen	Hardwarefehler	Es sind keine Fehler aufgetaucht. Die auf Vibration zurückzuführen sind.	↑
#S4.1	Hardwarefehler	Sensor endet in einem «Fehlermodus», nicht mehr bedienbar	GPS-Modul: Es werden keine Daten mehr geschrieben. SD-Modul: Sensor wird stürzt ab	↓

Abbildung 79 - Sensor: Extreme Tests | Tabelle Test-Run #1 (Quelle: Eigene Tabelle)

Change Log

Im Change log werden allfällige Änderungen und Lösungsversuche festgehalten.

#ID	Datum	Änderung
#S4.1	19.10.2022	Sensor: Error Management (GIT Commit 433b92e6e0f0fb1482b0f73f8054f7d3115d353)

Abbildung 80 - Sensor: Extreme Tests | Tabelle Change Log (Quelle: Eigene Tabelle)

Test-Run #2

Nach erfolgtem Lösungsversuch werden die fehlgeschlagenen Tests wiederholt.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
#S4.2	Hardwarefehler	GPS: Error Logging in den Flash Speicher und weiterführendes Daten schreiben, Wechsel in den Error State SD-Modul: Error Logging in den Flash Speicher, Wechsel in den Error State	GPS: Error Logging funktioniert, Wechsel in den Error State funktioniert nicht SD-Modul: Error Logging funktioniert, Sensor stürzt ab	➡

Abbildung 81 - Sensor: Extreme Tests | Tabelle Test-Run #2 (Quelle: Eigene Tabelle)

10.5 Data-Analyzer: Normale Tests

Test-Run #1

In diesen Test-Cases wird das System mit praktisch orientierten Tests geprüft.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
#S1.1	Upload Data	CSV Files werden in der Datenstruktur gespeichert	CSV Files werden in der Datenstruktur gespeichert	↑
#S2.1	Set Marker	Positions-Marker tauchen auf der Karte auf	Positions-Marker tauchen auf der Karte auf	↑
#S3.1	Reset Marker	Positions-Marker werden von der Karte gelöscht	Die Positionsmarker verschwinden auf der Karte	↑
#S4.1	Delete Data	CSV Files werden von der Datenstruktur entfernt	CSV Files werden von der Datenstruktur entfernt	↑

Abbildung 82 - Data-Analyzer: Normale Tests | Tabelle Test-Run #1 (Quelle: Eigene Tabelle)

Change Log

Im Change log werden allfällige Änderungen und Lösungsversuche festgehalten.

#ID	Datum	Änderung
-	-	-

Abbildung 83 - Data-Analyzer: Normale Tests | Tabelle Change Log (Quelle: Eigene Tabelle)

Test-Run #2

Nach erfolgtem Lösungsversuch werden die fehlgeschlagenen Tests wiederholt.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
-	-	-	-	-

Abbildung 84 - Data-Analyzer: Normale Tests | Tabelle Test-Run #2 (Quelle: Eigene Tabelle)

10.6 Data-Analyzer: Subnormale Tests

Test-Run #1

In diesen Test-Cases wird das System mit praktisch orientierten Tests geprüft.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
#S1.1	Falsches Dateiformat	Falsche Dateien werden nicht hochgeladen	Alle Dateien können ausgewählt und hochgeladen werden	↓
#S2.1	Falsche CSV-Formatierung	Darstellung wird abgebrochen	Darstellung crasht mit Fehlermeldungen (JS Konsole)	↓
#S3.1	Fehlende Datenpunkte	Messpunkt wird übersprungen	Darstellung crasht mit Fehlermeldungen (JS Konsole)	↓

Abbildung 85 - Data-Analyzer: Subnormale Tests | Tabelle Test-Run #1 (Quelle: Eigene Tabelle)

Change Log

Im Change log werden allfällige Änderungen und Lösungsversuche festgehalten.

#ID	Datum	Änderung
#S1.1	01.11.2022	Datei Input Feld mit erlaubten Dateitypen ergänzt. Dateiüberprüfung hinzugefügt.
#S2.1	01.11.2022	Marker Erstellung mit Try Catch abgesichert
#S3.1	01.11.2022	Marker Erstellung mit Try Catch abgesichert

Abbildung 86 - Data-Analyzer: Subnormale Tests | Tabelle Change Log (Quelle: Eigene Tabelle)

Test-Run #2

Nach erfolgtem Lösungsversuch werden die fehlgeschlagenen Tests wiederholt.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
#S1.2	Falsches Dateiformat	Falsche Dateien werden nicht hochgeladen	Falsche Dateien können nur über einen Umweg ausgewählt werden. Jedoch werden diese nach der Überprüfung nicht gespeichert.	↑
#S2.2	Falsche CSV-Formatierung	Darstellung wird abgebrochen	Darstellung wird versucht, jedoch wird Fehlermeldung in der Konsole ausgegeben.	↑
#S3.2	Fehlende Datenpunkte	Messpunkt wird übersprungen	Messpunkt wird übersprungen	↑

Abbildung 87 - Data-Analyzer: Subnormale Tests | Tabelle Test-Run #2 (Quelle: Eigene Tabelle)

10.7 Data-Analyzer: Extreme Tests

Test-Run #1

In diesen Test-Cases wird das System mit praktisch orientierten Tests geprüft.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
#S1.1	Falsche Daten (z.B. falsch formatierte Koordinaten)	Datenpunkt wird übersprungen.	Messpunkt wird übersprungen oder es werden falsche Daten angezeigt	→
#S2.1	Vergleich von Fahrdaten (exakt gleiche Position)	Beide Datenpunkte sichtbar	Bei genau exakter Position überlappen sich die Marker und nur der Obere wird angezeigt	→

Abbildung 88 - Data-Analyzer: Extreme Tests | Tabelle Test-Run #1 (Quelle: Eigene Tabelle)

Change Log

Im Change log werden allfällige Änderungen und Lösungsversuche festgehalten.

#ID	Datum	Änderung
-	-	-

Abbildung 89 - Data-Analyzer: Extreme Tests | Tabelle Change Log (Quelle: Eigene Tabelle)

Test-Run #2

Nach erfolgtem Lösungsversuch werden die fehlgeschlagenen Tests wiederholt.

#ID	Testbeschrieb	Gewünschtes Resultat	Testresultat	Ampel
-	-	-	-

Abbildung 90 - Data-Analyzer: Extreme Tests | Tabelle Test-Run #2 (Quelle: Eigene Tabelle)

11 Abschlussbericht

11.1 Sachergebnisse

11.1.1 Veränderungen der Aufgabenstellung / Zielsetzung

Es wurden keine Änderungen an der Aufgabenstellung und Zielsetzung vorgenommen.

11.1.2 Zielüberprüfung

Ziel	Beschreibung	Muss	Wunsch	Abgrenzung	Erfüllt	Teilw. erfüllt	Nicht erfüllt
#1	Sensor: Erfassung von Telemetriedaten mindestens alle ~500ms	X			X		
#2	Sensor: Genauigkeit Standort-Daten mind. 5m unter freiem Himmel	X				X	
#3	Sensor: Genauigkeit Geschwindigkeit (+/- 5km/h)	X			X		
#4	Sensor: Neigungswinkel / Kurvenlage (+/- 5°, bis max. 60°)	X			X		
#5	Sensor: Speicherung der Daten auf lokalem Speichermedium	X			X		
#6	Sensor: Bedienung der Sensorfunktionen via Smartphone	X			X		
#7	Data-Analyzer: Auswertung der Daten via Browserapplikation.	X			X		
#8	Data-Analyzer: Topographische Karte mit den Messpunkten	X			X		
#9	Data-Analyzer: Anzeige der Messwerte pro Messpunkt	X			X		
#10	Data-Analyzer: Überlagerung von bis zu drei gemessenen Aufnahmen		X		X		
#11	Sensor: Erstellung eines PCBs		X		X		
#12	Sensor: Erstellung eines Gerät-Gehäuses		X		X		
#13	Sensor / Data-Analyzer: Produkt im Markt einführen			X			
#14	Data-Analyzer: Hosting als Webseite			X			

Abbildung 91 - Tabelle Zielüberprüfung (Quelle: Eigene Tabelle)

#1: Sensor: Erfassung von Telemetriedaten mindestens alle ~500ms

Die Telemetriedaten können bis zu all 200ms erfasst werden.

#2: Sensor: Genauigkeit Standort-Daten mind. 5m unter freiem Himmel

Auf geraden Strecken kann die definierte Genauigkeit von 5m eingehalten werden. Jedoch gibt es in kurvigen Strecken teilweise höhere Abweichungen.

#4: Sensor: Neigungswinkel / Kurvenlage (+/- 5°, bis max. 60°)

Solange der Sensor fest auf dem Motorrad montiert ist, kann das Ziel eingehalten werden.

11.1.3 Erfahrungen

Im Nachhinein war der «Arduino RP240 Connect» die falsche Wahl. Dieses Mikrokontroller Dev-Board wird im Arduino Universum bereits sehr gut unterstützt, jedoch fehlt die grosse Adaption von vielen Bibliotheken und die Multicore Unterstützung.

Von Arduino wird eine Firmware zur Verfügung gestellt, damit das Dev-Board mit MicroPython betrieben werden kann. Leider ist damit nicht das komplette Feature-Set verfügbar und gewisse Funktionen, wie zum Beispiel die SD-Karten Bibliothek, sind nicht verfügbar.

Wie sich im Nachhinein herausgestellt hat, befindet sich auf diesem Arduino Dev-Board zusätzlich ein ESP-32, der für die W-Lan und Bluetooth Funktionalität zuständig ist.

Für eine neue Version des Sensors würde wahrscheinlich ein ESP-32 mit externem IMU zum Einsatz kommen, da diese besser dokumentiert sind und bereits eine grosse Menge an unterstützten Bibliotheken bestehen.

11.2 Zeitmanagement

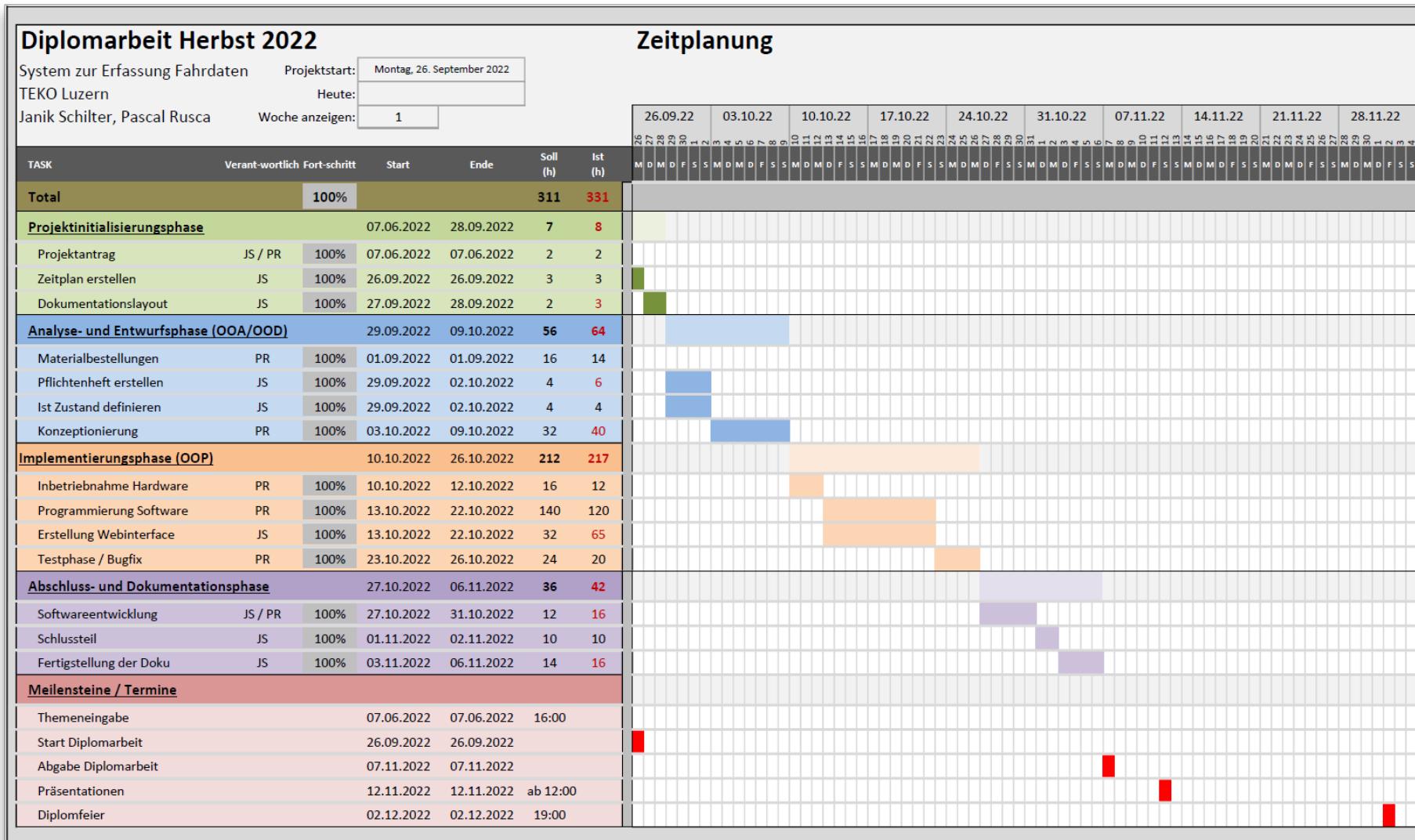


Abbildung 92 - Zeitplan Abschlussbericht (Quelle: Eigener Zeitplan)

11.2.1 Projektphasen

11.2.1.1 Projektinitialisierungsphase

Die Start- und Endtermine der Projektinitialisierungsphase wurden eingehalten.

Der Soll-Aufwand wurde mit sieben Arbeitsstunden abgeschätzt. Die tatsächlichen Aufwände beliefen sich auf acht Arbeitsstunden.

Die Mehraufwände werden in [11.3.1-Kosten-Controlling Honorar](#) beschrieben.

11.2.1.2 Analyse- und Entwurfsphase (OOA / OOD)

Die Start- und Endtermine der Analyse- und Entwurfsphase wurden eingehalten.

Der Soll-Aufwand wurde mit 56 Arbeitsstunden abgeschätzt. Die tatsächlichen Aufwände beliefen sich auf 64 Arbeitsstunden.

Die Mehraufwände werden in [11.3.1-Kosten-Controlling Honorar](#) beschrieben.

11.2.1.3 Implementierungsphase (OOP)

Die Start- und Endtermine der Projektinitialisierungsphase wurden eingehalten.

Der Soll-Aufwand wurde mit 212 Arbeitsstunden abgeschätzt. Die tatsächlichen Aufwände beliefen sich auf 217 Arbeitsstunden.

Die Mehraufwände werden in [11.3.1-Kosten-Controlling Honorar](#) beschrieben.

11.2.1.4 Abschluss- und Dokumentationsphase

Die Start- und Endtermine der Projektinitialisierungsphase wurden eingehalten.

Der Soll-Aufwand wurde mit 36 Arbeitsstunden abgeschätzt. Die tatsächlichen Aufwände beliefen sich auf 42 Arbeitsstunden.

Die Mehraufwände werden in [11.3.1-Kosten-Controlling Honorar](#) beschrieben.

11.2.1.5 Planungsqualität

Die Planung wurde zu Beginn des Projektes detailliert und mit genügend Freiraum erarbeitet, um für allfällige Abweichungen einen Puffer zu haben. Daher konnten die Projektbeteiligten sich auf die geplanten Termine verlassen und diese auch ohne Abweichungen einhalten.

11.3 Kostenübersicht

11.3.1 Kosten-Controlling Honorar

Das geplante Honorar Projektleiter von CHF 36'000 exkl. MwSt. konnte nicht eingehalten werden. Die effektiven Kosten belaufen sich auf CHF 39'720 exkl. MwSt. Dies entspricht einer Überschreitung der Kosten von CHF 3'720 oder 20 Arbeitsstunden.

Kostenpunkt	Budgetiert	Kosten	Eingehalten
Honorar Projektleiter	CHF 36'000 exkl. MwSt.	CHF 39'720 exkl. MwSt.	Nein

Abbildung 93 - Tabelle Kosten-Controlling Honorar (Quelle: Eigene Tabelle)

Diese Mehrkosten entstanden in folgenden Arbeitspaketen:

- Dokumentationslayout:
→ Einbindung von Tabellen in Quellenverzeichnis
- Pflichtenheft:
→ Das Pflichtenheft wurde nach Absprache mit dem Projektbetreuer nochmals überarbeitet, um die zu messenden Daten genauer zu spezifizieren.
- Konzeptionierung:
→ Nach Absprache mit dem Projektbetreuer wurden in der OOD-Phase diverse Anwendungsfalldiagramme erstellt.
- Data-Analyzer:
→ Die Implementation des Data-Analyzer war aufwändiger als in der Projektinitialisierungsphase geschätzt.
- Dokumentationsphase:
→ In Absprache mit dem Projektbetreuer wurden diverse Änderungen und zusätzliche Grafiken sowie Übersichten im Projektbericht integriert.

11.3.2 Kosten-Controlling Material

Die Materialkosten konnten wie geplant eingehalten werden. Es mussten keine zusätzlichen Ersatzteile bestellt werden. Die Komponenten wurden richtig evaluiert.

Kostenpunkt	Budgetiert	Kosten	Eingehalten
Materialkosten	CHF 274.40 exkl. MwSt.	CHF 274.40 exkl. MwSt.	Ja

Abbildung 94 - Tabelle Kosten-Controlling Material (Quelle: Eigene Tabelle)

11.4 Ausblick

11.4.1 Restaktivität

Innerhalb dieser Diplomarbeit sind folgende Restaktivitäten eingeplant:

Sensor:

- Verbesserung GPS Parsing
- Verbesserung Error Handling

Data-Analyzer:

- Verbesserung Kartendarstellung
- Verbesserung Error Handling

11.4.2 Ergänzungen und Erweiterungen

Aus Sicht des Projektteams sind folgende Ergänzungen bzw. Erweiterungen des Projektes für die Zukunft möglich:

Sensor:

- Montageplatte mit Vibrationsschutz
- Push-Meldungen auf Smartphone
- Verbindung Sensor → Smartphone via Bluetooth
- Externe GPS-Antenne zur Verbesserung Empfangsstärke
- Erstellung Produktdatenblatt

Data-Analyzer:

- Öffnen mehrerer Popups gleichzeitig
- Verwaltung CSV-Files über Ordnerstruktur
- Darstellung der Daten mittels Gauges
- Hosting

11.5 Schlusswort / Reflexion

In einer kurzen Stellungnahme der beiden Autoren wird nun ein Rückblick auf die geleistete Arbeit geworfen.

11.5.1 Pascal Rusca

Das Arbeiten an dieser Diplomarbeit hat mir sehr viel Spass bereitet. Es war sehr interessant, eine «neue» Programmiersprache für Mikrocontroller auszuprobieren und auch gleich produktiv einzusetzen. Auch konnte ich zum ersten Mal ein PCB erstellen und mein Wissen in der Konstruktion von Bauteilen einsetzen. Das Arbeiten mit einem neuen Mikrocontroller hat mich einige Stunden und Nerven gekostet. Da bis jetzt wenig Personen damit gearbeitet haben, musste ich viel selbst erarbeiten, was jedoch im Nachhinein sehr lehrreich gewesen ist.

Ich bin mit dem erarbeiteten Produkt sehr zufrieden und freue mich, dieses in der nächsten Zeit noch überarbeiten und verbessern zu können, um es in der nächsten Saison auf vielen Touren einsetzen zu können.

Die Zusammenarbeit im Team hat mir sehr gefallen. Nachdem wir bereits eine Semesterarbeit zusammen umgesetzt haben, war für mich klar, dass Janik ein perfekter Partner für die Diplomarbeit sein wird. Ich konnte sehr von Janiks Know-how im Gestalten von Dokumenten profitieren. Dank dem konstanten Austausch via WhatsApp oder Teams war ich immer auf dem aktuellen Stand und konnte mein Teil der Arbeiten effizient und ohne Verzögerungen durchführen.

Ein grosses Dankeschön geht auch an unseren Betreuer Andreas Holzer. Er hat uns viele wertvolle Tipps gegeben, was die Dokumentation und die Programmierung betrifft.

11.5.2 Janik Schilter

Für mich war die Arbeit an dieser Diplomarbeit sehr interessant. Ich hatte die Möglichkeit, meine Erfahrungen, beispielsweise mit JavaScript, HTML und CSS zu vertiefen. Außerdem habe ich erstmals mit dem Web-Framework Flask und der WebGIS-Bibliothek Leaflet gearbeitet und konnte dabei viel Neues dazulernen.

Letztendlich bin ich mit dem Ergebnis der Diplomarbeit sehr zufrieden und freue mich darauf, das entwickelte System noch bei vielen weiteren Motorradtouren zu verwenden.

Die Arbeit im Projektteam hat mir viel Spass bereitet und ich empfand die Zusammenarbeit als ideal. Durch Projektbesprechungen im Microsoft Teams und dem regelmässigen Austausch über WhatsApp, konnten wir eine einwandfreie Kommunikation und Zusammenarbeit gewährleisten. Außerdem konnte ich sehr von Pascals Know-how profitieren und möchte an dieser Stelle auch meinen Dank an ihn aussprechen.

Zudem möchte ich mich auch gerne bei unserem Betreuer Andreas Holzer bedanken. In unseren gemeinsamen Meetings konnte Andreas wertvolle Tipps geben. Außerdem hat er uns bezüglich des Inhalts der Dokumentation gefordert und Inputs gegeben, welche Diagramme, Grafiken und Informationen noch sinnvoll wären.

11.6 Eigenständigkeitserklärung

„Hiermit erklären wir, dass wir die vorliegende schriftliche Arbeit selbstständig und nur unter Zuhilfenahme, der in den Verzeichnissen oder in den Anmerkungen genannten Quellen angefertigt haben. Wir versichern zudem, diese Arbeit nicht anderweitig als Leistungsnachweis verwendet zu haben.“

Eine Überprüfung der Arbeit auf Plagiate unter Einsatz entsprechender Software darf vorgenommen werden. Wir sind damit einverstanden, dass die TEKO-Luzern zu diesem Zweck entsprechende Dienstleister im In- oder Ausland beauftragen kann, welche von dieser auf Gewährleistung der Datensicherheit kontrolliert werden.“

06.11.2022 / Obernau

Datum / Ort



Unterschrift

06.11.2022 / Alpnach Dorf

Datum / Ort



Unterschrift

12 Anhang

12.1 Abgabe Projektunterlagen

TEKO-Extranet:

- Projektbericht
- Bedienungsanleitung

Abgabe an Betreuer via MS Teams:

- Projektbericht
- Bedienungsanleitung
- Quellcode (.zip)

12.2 Sitzungsprotokolle

Kickoff-Meeting Diplomarbeit vom 29.09.2022

Teilnehmer: Janik Schilter, Pascal Rusca, Andreas Holzer

Testing

- GPS-Daten auf Plausibilität prüfen:
→ GPS-Daten vom System mit Smartphone-Daten vergleichen
- Neigungswinkel Motorrad testen
- Kalibrierung testen: Längere Strecke geradeaus fahren
- Testverfahren: V-Modell

Allgemein

- Bei Winkelberechnungen die Funktion atan2() verwenden
→ verhindert Division durch 0
- Einfluss Vibrationen: Mittelwerte bilden: Tiefpassfilter (RC-Glied, PT1)
- Kalibrierung via Webinterface oder Button am Gerät

Technologien

- PlatformIO

Hardware

- Gehäuse wasserdicht ausführen
→ Taster zur Kalibrierung evtl. Kapazitiv
- Evt. Externe Antenne vorsehen, falls GPS-Empfang zu schwach
- Evt. Wasserwaage in Gehäuse einbauen
- Komponenten auf Printplatte löten (Vibrationen)
- Spannungsversorgung ab Motorradbatterie (ca. 14VDC)

Risiken

- Defekte Hardware (Vibrationen, Frost, Hitze, Feuchtigkeit)
- Falsch ausgewählte Hardware
→ Reservematerial bestellen

Dokumentation

- Dokumentation ca. 1 Woche vor Abgabe an Andreas Holzer zustellen
- Zwischenzeitlich keine anderen Unterlagen an Andreas Holzer zustellen

Weitere Meetings

- Mittwoch 12.10.2022 19:00 Uhr
- Montag 24.10.2022 18:00 Uhr

2. Meeting Diplomarbeit vom 13.10.2022

Teilnehmer: Janik Schilter, Pascal Rusca, Andreas Holzer

Traktanden

- Aktueller Stand Sensor
- Aktueller Stand Data-Analyzer
- Aktueller Stand Dokumentation

Aktueller Stand Sensor

- Pascal hat den aktuellen Stand erklärt

Aktueller Stand Data-Analyzer

- Janik hat den aktuellen Stand erklärt

Aktueller Stand Dokumentation

Der aktuelle Stand der Dokumentation wurde mit Andreas Holzer besprochen.
Nachfolgend sind einige Inputs von Andreas Holzer aufgelistet.

Qualifikationsprofil

- In Absprache mit Andreas Holzer ist kein Qualifikationsprofil und kein beruflicher Lebenslauf in der Dokumentation zu integrieren

Zeitplan / Projektphasen

- Zusätzlich zum Zeitplan wird eine Projektphasenübersicht erstellt.

Pflichtenheft

- Detaillierungsgrad der Ziele erhöhen: z.B. wie häufig sollen Standortdaten gemessen werden? In welchem Bereich muss der Winkel gemessen werden?

Auswahl und Evaluierung

- In Absprache mit Andreas Holzer wird kein morphologischer Kasten oder Entscheidungsmatrix zur Variantenbildung dokumentiert. Dies daher, weil die Komponenten aufgrund aktuell langer Lieferzeiten bereits vor Start der Diplomarbeit bestellt werden mussten.

Dokumentation Sensor

- State Diagramm: Error State ergänzen
- Sequenzdiagramm ergänzen
- GPS_Parsing: Genauer dokumentieren, da eigene Library

Data-Analyzer

- Strukturdiagramm ergänzen (mit den eingesetzten Technologien + Datenfluss)
- Sequenzdiagramm/Ablaufdiagramm ergänzen (mit den Use-Cases)
- Möglichkeit zum Vergleich von Fahrdaten vorsehen
- Verwaltung der gesammelten Daten mit Ordnerstruktur auf Data-Analyzer

Testing

- Testkonzept erstellen (Gesamtübersicht)
- Test-Cases tabellarisch erfassen
- Test-Runs tabellarisch erfassen (mit Ampelsystem)
- Fehlerbehebungen → Changelog

Dokumentation Gesamtsystem

- Grafische Übersicht erstellen mit allen Systemkomponenten (Gesamtübersicht)

Handbuch / Bedienungsanleitung

- Erstellen einer Bedienungsanleitung (Separates Dokument)
- Darin sollen die Inbetriebnahme und Bedienung ersichtlich sein
- Exkl. Erstellung der Hardware (Der Kunde bekommt ein fertiges Gerät)

Quellenangaben

- Für JEDES Bild muss eine Quellenangabe vorhanden sein
- Bei eigenen Bildern, dies entweder bei den Bildern jeweils vermerken, oder eine allgemeine Info bei den Quellenangaben versehen.

Sitzungsprotokolle

- Die Sitzungsprotokolle sollen nicht als separate Dokumente erstellt werden, sondern am Schluss der Dokumentation integriert werden.

Produktdatenblatt

- Evt. Erstellung von Produktdatenblatt

Allgemeines zur Dokumentation

- Keine Wikipedia Erklärungen im Kapitel. Diese gehören ins Glossar.
- Keine Tabelle oder Bild ohne Text

Folgendes soll abgegeben werden

- Dokumentation
- Zeitplan
- Pflichtenheft
- Bedienungsanleitung
- Evt. Produktdatenblatt
- USB-Stick mit allen Unterlagen

Termin Präsentation

- Gemäss Andreas Holzer voraussichtlich am Sa 12.11.2022 ab 12:00 Uhr

Pressepublikation

- Pressepublikation auf externem TOOL (Richtlinien) (z.B. Wie Dell auf Linkedin)

3. Meeting Diplomarbeit vom 24.10.2022

Teilnehmer: Janik Schilter, Pascal Rusca, Andreas Holzer

Traktanden

- Aktueller Stand Sensor
- Aktueller Stand Webinterface/Datenauswertung
- Aktueller Stand Dokumentation

Aktueller Stand Sensor

- Pascal hat den aktuellen Stand erklärt

Aktueller Stand Data-Analyizer

- Janik hat den aktuellen Stand erklärt

Aktueller Stand Dokumentation

Der aktuelle Stand der Dokumentation wurde mit Andreas Holzer besprochen.
Nachfolgend sind einige Inputs von Andreas Holzer aufgelistet.

Sensor – Auswertung Daten

- Evtl. Auswertung der Anzahl verbundener Satelliten
- Evtl. Implementieren von Squelch-Funktion, damit nur ab gewisser Empfangsstärke Daten geschrieben werden.
- Evtl. zusätzliche externe Antenne, damit bessere Empfangsstärke
- Datenspeicherung evtl. mit zwei alternierenden Puffern

Webinterface Sensor

- Möglichkeit die Daten via Webinterface als CSV herunterzuladen

Webinterface Datenauswertung

- Bezeichnung «Webseite» ersetzen durch passenderen Begriff

Anwendungsfälle

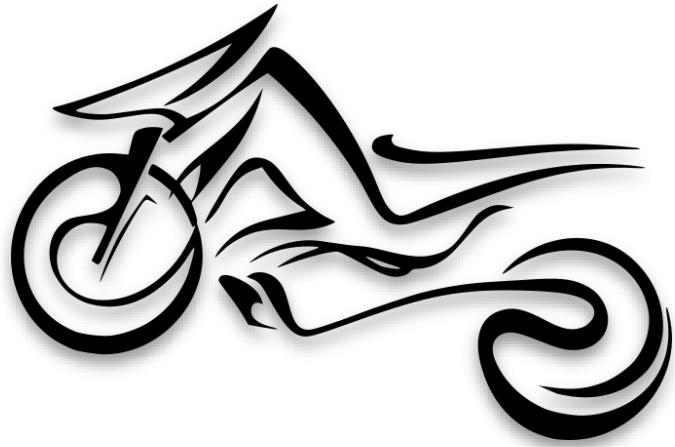
- Kein Titel vor Programmablaufdiagrammen – somit gilt die Tabelle jeweils als Beschreibung des Anwendungsfalls

Testing

- Test-Cases in drei verschiedene Kategorien gliedern.
 1. Normale Tests: zu erwartende Anwendungsfälle testen
 2. Subnormale Tests: Risiken einbeziehen (Tunnel, Wetter, usw.)
 3. Extreme Tests: Kaputte SD-Karte, Überspannungen, usw.

Termin Präsentation

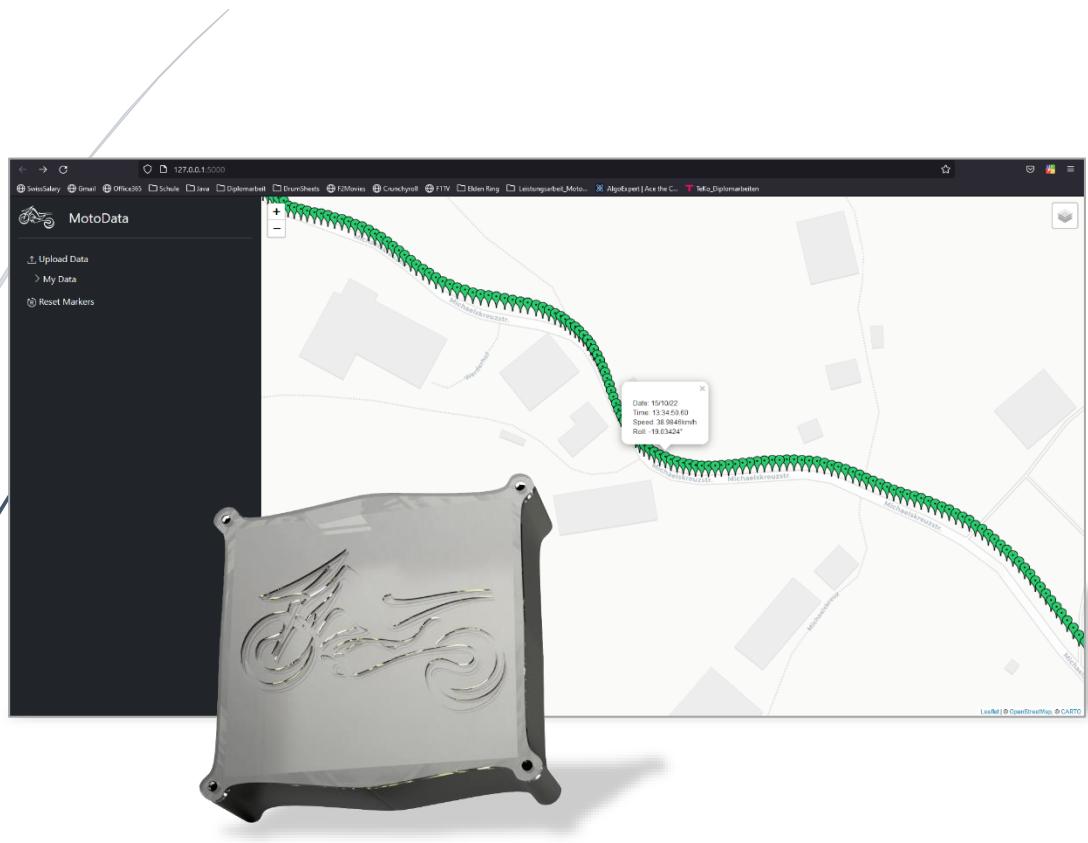
- Definitiven Termin mit Bruno Hammer klären



Bedienungsanleitung

MotoData

Ein System zur Erfassung
von Fahrdaten



1 Inhalt

2 Nutzen dieses Dokuments.....	3
3 Symbole / Piktogramme	3
4 Das Produkt	4
4.1 Sensor	4
4.2 Data-Analyzer	4
5 Setup	5
5.1 Sensor	5
5.1.1 Spannungsversorgung.....	5
5.1.2 Montage.....	6
5.2 Data-Analyzer	7
6 Betrieb.....	8
6.1 Sensor	8
6.1.1 Status-LEDs.....	8
6.1.2 Webinterface.....	8
6.2 Data-Analyzer	12
6.2.1 Vorbereitung	12
6.2.2 Sidebar	12
6.2.3 Upload Data.....	12
6.2.4 Delete Data	13
6.2.5 Set Marker on Map.....	13
6.2.6 Reset Map.....	13
6.2.7 Datenauswertung.....	14

DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 2 / 14

2 Nutzen dieses Dokuments

Diese Bedienungsanleitung soll dem Benutzer dabei helfen, das Produkt korrekt zu verwenden.

Im Kapitel **5 Das Produkt** das Produkt genauer beschrieben und vorgestellt.

Im Kapitel **6 Setup** werden alle Schritte erläutert, welche vor Verwendung des Gerätes notwendig sind.

Im Kapitel **7 Betrieb** wird sichergestellt, dass der Benutzer weiss, wie dieses Produkt zu verwenden ist.

3 Voraussetzungen

Für die Nutzung des Sensors und des Data-Analyzer werden technische Kenntnisse im Bereich WSL und Python vorausgesetzt.

4 Symbole / Piktogramme

In diesem Dokument werden folgende Piktogramme verwendet, um bestimmte Aussagen optisch klar darzustellen:

	Wichtige Information
	Gut zu Wissen
	Achtung / Vorsicht
	Aktion zu vermeiden
	Zwingende Massnahme
	Sensitive oder wichtige Prozedur

DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 3 / 14

5 Das Produkt

5.1 Sensor

Der Sensor wird am oder im Motorrad montiert und erfasst folgende Telemetriedaten:

- Standort-Daten
- Geschwindigkeit
- Neigungswinkel / Kurvenlage



Zur Spannungsversorgung eignet sich eine Powerbank oder der direkte Anschluss an die Motorradbatterie.



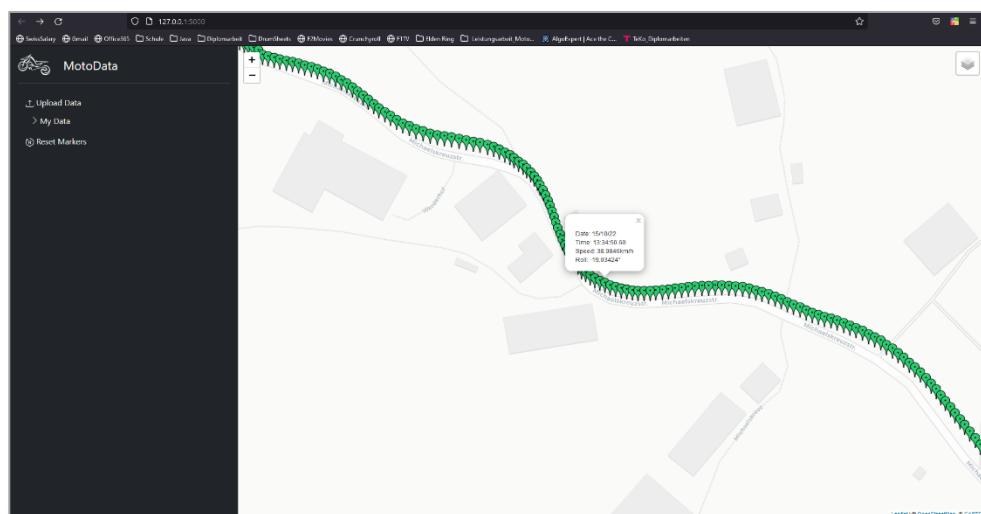
Die Bedienung des Sensors erfolgt über ein Web-Interface via Smartphone.

5.2 Data-Analyzer

Die erfassten Fahrdaten können im Data-Analyzer eingesehen und genauer analysiert werden. Der Data-Analyzer umfasst folgende Funktionen:



- Upload der erfassten Telemetriedaten
- Löschen der Uploads
- Anzeigen der erfassten Telemetriedaten auf einer topografischen Karte
- Es können bis zu drei erfassten Touren gleichzeitig angezeigt werden.
- Die Touren werden zu Unterscheidung in verschiedenen Farben dargestellt.
- Entfernen der angezeigten Messpunkte auf der Karte



DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 4 / 14

6 Setup

6.1 Sensor

6.1.1 Spannungsversorgung

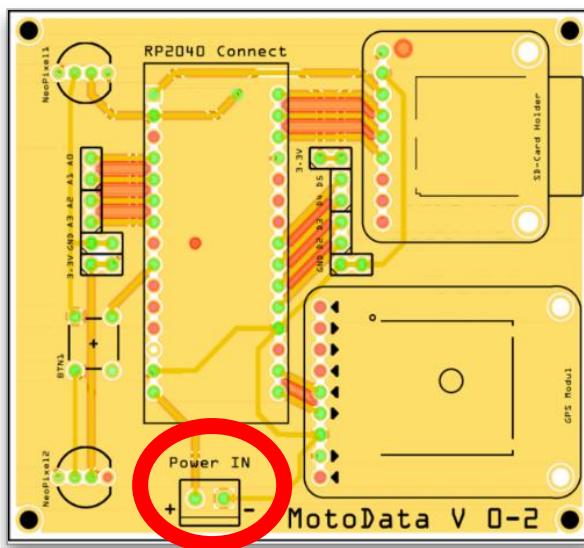
Die Spannungsversorgung erfolgt über zwei möglich Varianten:



1. Spannungsversorgung via Powerbank:
Dabei wird der Sensor via Micro-USB-Kabel angeschlossen.



2. Spannungsversorgung via Motorradbatterie:
Dabei wird der Sensor mit einem CU-Kabel an folgenden Pins angeschlossen:



Achtung:

Die Spannung ab Motorradbatterie muss sich im Bereich von 5-21V DC befinden!

DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 5 / 14

6.1.2 Montage

Damit qualitativ hochwertige Messdaten entstehen empfiehlt es sich bei der Montage des Sensors auf folgende Punkte zu achten:



- **feste Montage:**
der Sensor darf sich während der Fahrt nicht bewegen
- **Lage:**
der Sensor soll möglichst horizontal befestigt werden.
- **Standort:**
Der Sensor kann unter dem Sattel montiert werden. Jedoch empfiehlt es sich, den Sensor beispielsweise auf dem Tank oder dem Rücksitz zu befestigen, damit ein idealer GSP-Empfang erreicht werden kann.



Um eine lange Laufzeit des Produkts zu gewährleisten, empfiehlt es sich bei der Montage Vorkehrungen für den Schutz vor Vibrationen zu treffen.

DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 6 / 14

6.2 Data-Analyzer

6.2.1 Installation

Zur Verwendung des Data-Analyzer werden die Projektdateien von MotoData in ein gewünschtes Installationsverzeichnis kopiert.

Kompatible Betriebssysteme:

- Linux
- Windows / WSL

Installation von Bibliotheken mit Python pip:

```
1. pip install -r requirements.txt
```

6.2.2 Start

Um den Data-Analyzer zu starten, muss der Installationsordner in einer Konsole geöffnet werden. Anschliessend kann die Applikation mittels des Befehls «flask run» gestartet werden. Bei einem erfolgreichen Start wird die IP-Adresse des Data-Analyzer angezeigt. Diese können Sie nun mit dem Browser Ihrer Wahl öffnen.

Untenstehend sehen Sie ein Beispiel wie ein erfolgreicher Start im WSL aussieht.

```
1. jasc@PC-JASC:/mnt/g/Coding/VSC/Diplomarbeit_WS2022_Janik-Pascal/Server$ flask run
2. * Debug mode: off
3. WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
4. * Running on http://127.0.0.1:5000
5. Press CTRL+C to quit
```

DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 7 / 14

7 Betrieb

7.1 Sensor

7.1.1 Status-LEDs

-  LED grün:
Sensor ist einsatzbereit
-  LED rot:
Sensor ist am Messen
-  LED blau:
Sensor ist am Kalibrieren
-  LED violett:
Sensor ist im Fehlermodus

7.1.2 Webinterface



Bitte beachten Sie dabei, dass der Sensor an die Spannungsversorgung angeschlossen ist. Achten Sie darauf, dass die Status-LED-vom Sensor grün ist.



Um das Webinterface nutzen zu können, muss vorgängig eine W-LAN-Verbindung mit dem Sensor aufgebaut werden.



SSID: MotoData-Sensor
Passwort: 12345678

Haben Sie sich mit dem Smartphone im W-LAN angemeldet, so haben Sie die Möglichkeit mit einem Browser Ihrer Wahl auf folgende Adresse zu zugreifen.

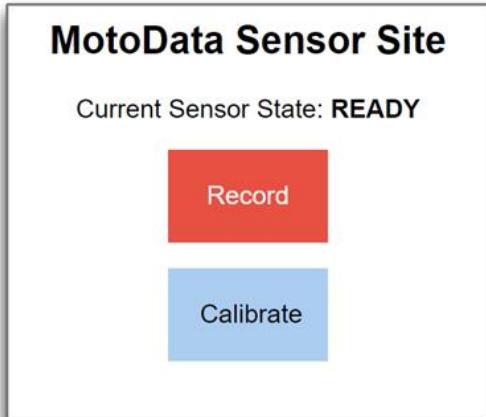


192.168.4.1

DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 8 / 14

7.1.2.1 Startseite

Auf der Startseite wird der aktuelle Sensor Status angezeigt. Zeigt dieser «READY»,



kann einer der zwei Buttons gewählt werden. Mit einem Knopfdruck wird der Status des Sensors gewechselt und dessen Funktionen ausgeführt.

Button <Record>:

Der Status «INIT_RECORD» wird ausgelöst und die Seite

Start Recording wird aufgerufen.

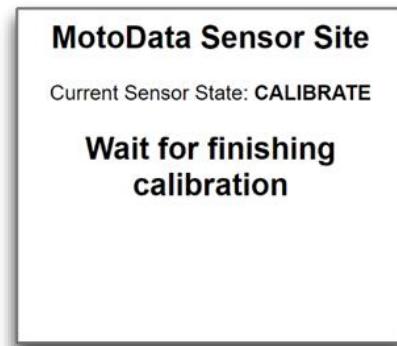
Button <Calibrate>:

Der Status «CALIBRATE» wird ausgelöst und

die [7.1.2.2 Kalibrierung](#) wird aufgerufen.

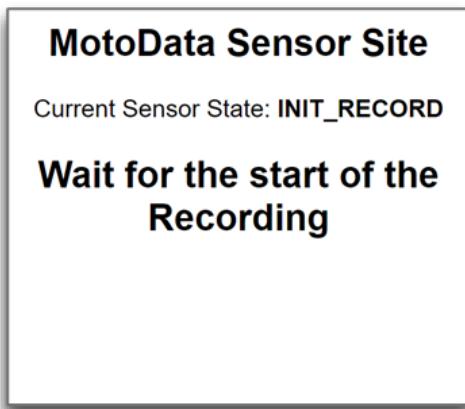
7.1.2.2 Kalibrierung

Während der Kalibration des Sensors wird diese Nachricht angezeigt. Sobald die Kalibration abgeschlossen ist, wird wieder auf die Startseite gewechselt.



DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 9 / 14

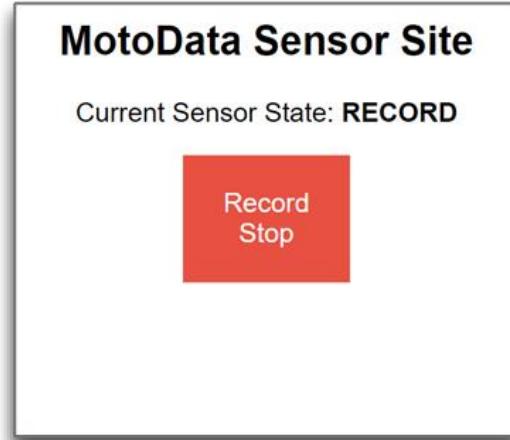
7.1.2.3 Start Recording



Wurde der Status «INIT_RECORD» ausgelöst, wird eine Seite angezeigt, während die Initialisierung des Aufnahmeprozesses stattfindet.

7.1.2.4 Recording

Sobald das Recording gestartet wurde, ist es auch möglich dieses wieder zu stoppen. Mit einem Knopfdruck auf den Button <Record Stop> wird der Status am Sensor gewechselt zu «STOP_RECORD» gewechselt.



DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 10 / 14

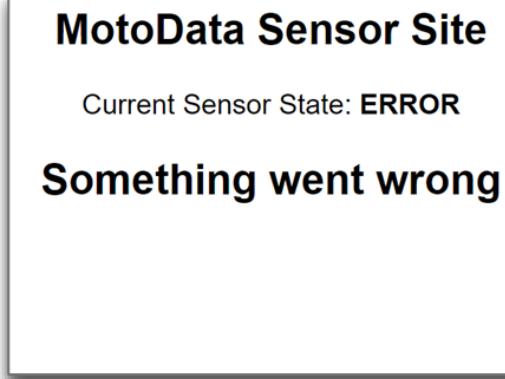
7.1.2.5 Stop Recording



Wurde der Status «STOP_RECORD» ausgelöst, wird die Aufnahme von Telemetrie-Daten beendet.

7.1.2.6 Fehlermeldung

Befindet sich der Player im «ERROR» Status, wird folgende Seite angezeigt.



DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 11 / 14



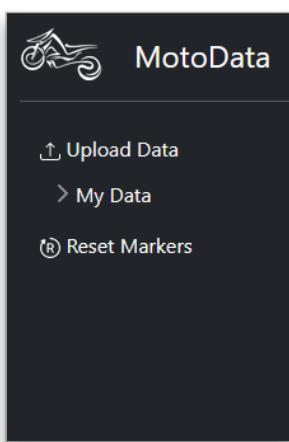
7.2 Data-Analyzer

7.2.1 Vorbereitung



Bevor mit dem Data-Analyzer gearbeitet werden kann müssen die gesammelten Telemetriedaten vom Sensor heruntergeladen werden. Dafür verbinden Sie Ihren Laptop oder Desktop-Computer via Micro-USB-Kabel mit dem Sensor. Nun können Sie das darin enthaltene CSV-File auf Ihren Computer kopieren und anschliessen im Data-Analyzer importieren.

7.2.2 Sidebar



Die Sidebar wurde sehr einfach gehalten, damit eine übersichtliche intuitive Bedienung möglich ist.

Zur Auswahl stehen drei Buttons:

- Upload Data
- My Data
- Reset Markers

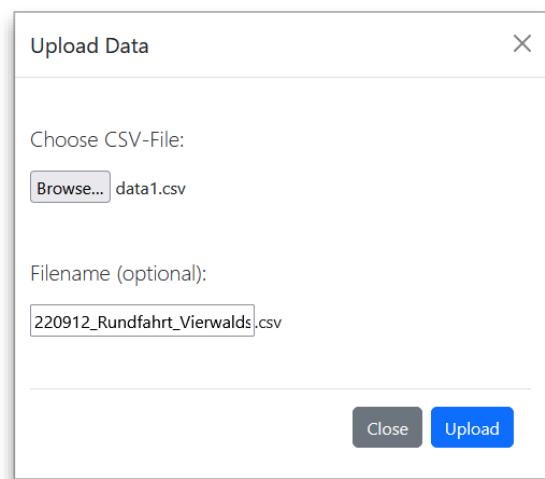
Die Funktionalität dieser Buttons wird auf den nächsten Seiten genauer beschrieben.

7.2.3 Upload Data

Mithilfe des Sidebar-Button <Upload Data> wird das Modal Upload Data aufgerufen. Mit Klick auf den Button <Durchsuchen...> kann via Explorer die gewünschte CSV-Datei ausgewählt werden.

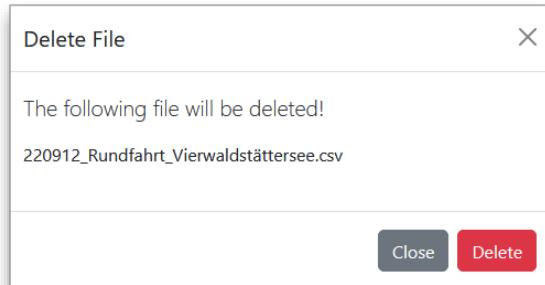
Nun wird im Eingabefeld der gewünschte Dateiname ergänzt und mit dem Button <Daten absenden> das CSV-File hinzugefügt.

Die hochgeladene Datei wird nun in der Sidebar unter dem Punkt <My Data> angezeigt.



DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 12 / 14

7.2.4 Delete Data



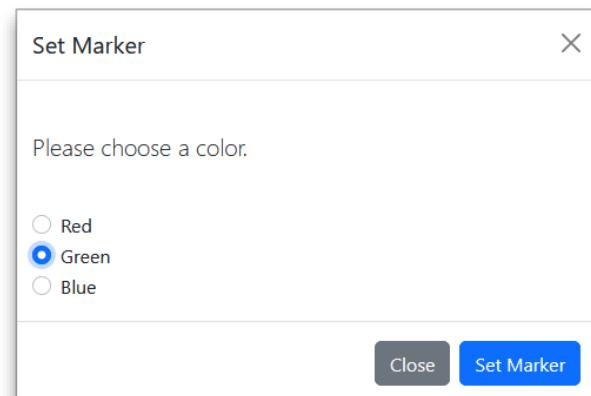
Wird der Sidebar-Button zum Löschen gewählt, so öffnet sich das Modal <Delete File>. Hier kann das Löschen der Datei nun bestätigt oder abgebrochen werden.

7.2.5 Set Marker on Map

Um die hochgeladenen Messdaten nun auf der topografischen Karte anzeigen zu lassen, wird die gewünschte Datei in der Sidebar angeklickt. Dabei öffnet sich das Modal <Set Marker>.

Damit verschiedene Fahrtrouten miteinander verglichen werden können, kann hier eine Farbe gewählt werden.

Die Messdaten werden als Marker mit der gewünschten Farbe auf der Karte dargestellt.



7.2.6 Reset Map

Um die gesetzten Messpunkte/Marker von der Weltkarte zu entfernen, kann der Button <Reset Markers> in der Sidebar ausgewählt werden. Somit werden alle angezeigten Marker entfernt. Die gespeicherten CSV-Files bleiben jedoch bestehen.

Reset Markers

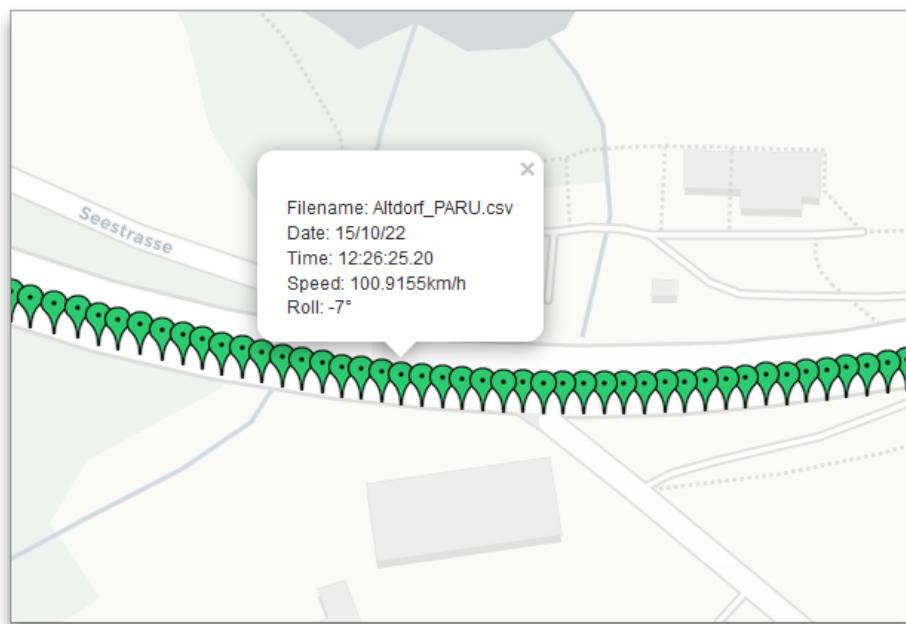
DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 13 / 14

7.2.7 Datenauswertung

Wurden die Marker in der Weltkarte gesetzt, so können diese angewählt werden, wobei sich ein Popup des jeweiligen Markers öffnet.

In diesem Popup werden folgende Daten angezeigt:

- | | |
|------------------------|---|
| 1. Date: 15/10/22 | → Datum der Erfassung der Messdaten |
| 2. Time: 13:34:50.60 | → Zeitpunkt der Erfassung der Messdaten |
| 3. Speed: 38.9846 km/h | → Geschwindigkeit in km/h |
| 4. Roll: -19.03424° | → Neigungswinkel (Schräglage Motorrad) |



DA_2022_Bedienungsanleitung_Name_Vorname_L-TIN19-Mi-a_V0-02.docx []				
Letzte Änderung	Sonntag, 6. November 2022			
Autoren	PARU/JASC	Version:	1.0	Page 14 / 14