Diplomarbeit

Motorrad Dashboard

Jonas Kuster

Abbildung 1: Titelblatt Motorrad Dashboard

Diplomand:	Jonas Kuster
Höhere Fachschule:	TEKO Luzern
Klasse:	L-TEL-22-Do-a
Fachrichtung:	Dipl. Elektrotechniker HF
Ort, Datum:	Luzern, 9. November 2025

Abbildung 2: Projektdaten

1. Inhaltsverzeichnis

1 Management Summary

1.1 Ausgangslage

Aufbauend auf dem im Unterricht vermittelten Grundlagenwissen konnte ich in meiner Diplomarbeit ein selbstgewähltes Projekt umsetzen und mein Wissen praxisnah vertiefen. Ich entschied mich für die Entwicklung eines Motorrad Dashboards, ein Projekt, das mein technisches Interesse mit vielfältigen fachlichen und methodischen Anforderungen verbindet. Mithilfe von Projektmanagement Methoden setzte ich das Vorhaben strukturiert um und entwickelte eine eigene technische Lösung mit spezifischen Funktionen für den Motorradbereich. Dabei konnte ich mein technisches Verständnis erweitern und zugleich meine Selbstständigkeit, mein strukturiertes Arbeiten und meine Problemlösungsfähigkeit stärken, wichtige Kompetenzen für meine berufliche Zukunft.

1.2 Vorgehen

Im Rahmen meiner Diplomarbeit an der TEKO entwickelte ich eigenständig ein Konzept für ein Motorrad Dashboard. Das im Unterrichtsfach Projektmanagement erlernte Wissen half mir, das Projekt strukturiert zu planen und umzusetzen. Die Arbeit gliederte ich in vier Phasen. In der Projektinitialisierung erstellte ich ein Pflichtenheft, analysierte die Ausgangslage und definierte eine Zielscheibe als Leitfaden. In der Projektplanung legte ich den Ablauf fest, um Zeitrahmen und Fortschritt gezielt zu überwachen. In der Realisierung setzte ich das Konzept praktisch um, gestaltete das Design und programmierte das Dashboard. In der Abschlussphase reflektierte ich den Projektverlauf, identifizierte Verbesserungspotenziale und hielt meine wichtigsten Erkenntnisse im Kapitel Reflexion fest.

1.3 Ergebnisse

Ich habe ein Motorrad Dashboard entwickelt, das GPS-Geschwindigkeit, Höhe, Fahrtrichtung und Rundenzeiten zuverlässig anzeigt. Über das WLAN können Rundenzeiten, Extremwerte und GPS-Daten auf ein Smartphone übertragen werden. Die Bedienung erfolgt über fünf Knöpfe, die auch mit Handschuhen nutzbar sind. Das staub- und spritzwassergeschützte Gehäuse ist stabil am Lenker befestigt, sodass es während der Fahrt sicher sitzt. Die Stromversorgung erfolgt über die 12V-Batterie, abgesichert durch eine 2A-Sicherung und eine Schottky-Diode als Schutz gegen Verpolung. Ein Zustandsdiagramm dokumentiert die übersichtliche Softwarestruktur mit allen Menüs und Funktionen.

1.4 Ausblick

Zukünftige Versionen des Motorrad Dashboards könnten durch verschiedene Erweiterungen verbessert werden. Eine einstellbare Displayhelligkeit würde die Ablesbarkeit und Energieeffizienz erhöhen. Zudem könnten alle Messwerte, inklusive Geschwindigkeit, Höhe, Kompassrichtung und GPS-Daten, live auf dem Smartphone angezeigt werden. Eine Kartenfunktion würde die Navigation direkt über das Dashboard ermöglichen. Ergänzend könnten Zeitmessungen von 0–100 km/h sowie eine Uhrzeitanzeige integriert werden, um die Funktionalität und Benutzerfreundlichkeit weiter zu steigern.

2 Beruflicher Lebenslauf

Jonas Kuster
Kleinfeldstrasse 20, 6233 Büron | Jonas.kuj@bluewin.ch

Berufserfahrung:

Elektroinstallateur, CKW GBT | 2020 – heute
- Planung, Installation und Wartung elektrischer Anlagen
- Fehlerdiagnose und Durchführung von Prüfungen
- Kundenberatung und enge Zusammenarbeit im Team

Lehre Elektroinstallateur EFZ, CKW GBT | 2016 – 2020
- Praxisorientierte Ausbildung in Montage, Inbetriebnahme und Wartung elektrischer Systeme
- Arbeitssicherheit und normgerechtes Arbeiten

Weiterbildung:

Elektrotechniker HF, TEKO | 2023 – 2025
- Projektierung, Planung, Inbetriebnahme und Wartung elektrotechnischer Produkte und Anlagen
- Entwicklung elektronischer und digitaler Schaltungen
- Test- und Prüfsysteme konzipieren, moderne Technologien und Simulationstools einsetzen
- Programmierung von Mikroprozessoren und speicherprogrammierbaren Steuerungen (SPS)
- Führungsaufgaben in Produktion, Service oder Prüffeld
- Technische Schulungen planen und durchführen
- Projekt- und Qualitätsmanagement, Förderung energieeffizienter Prozesse

Kernkompetenzen:
- Elektrotechnische Planung und Installation
- Fehlerdiagnose und Wartung
- SPS-Programmierung und Schaltungsentwicklung
- Projektmanagement und Qualitätssicherung
- Teamführung und Kundenkommunikation
- Anwendung moderner Technologien und Simulationstools

3 Projektinitialisierung

Die Projektinitialisierung bildet den Startpunkt eines Projekts und legt die Grundlagen für einen erfolgreichen Verlauf. In diesem Kapitel wird der Projektauftrag definiert, die Ausgangslage analysiert und die Endergebnisse festgelegt. Ziel ist es, Klarheit über Ziele, Verantwortlichkeiten und Rahmenbedingungen zu schaffen. Eine solide Projektinitialisierung legt den Grundstein für eine effektive Planung und Umsetzung des Projekts.

3.1 Richtlinien TEKO

TEKO
SCHWEIZERISCHE FACHSCHULE

Richtlinien Diplomarbeit

alle Studienrichtungen

Ziel und Zweck

Mit der Diplomarbeit zeigst du, dass du eine Aufgabenstellung innerhalb einer vorgeschriebenen Zeitspanne selbstständig, umfassend und zweckmässig lösen sowie sauber dokumentieren und präsentieren kannst.

Daten

Orientierung Diplomarbeit	KW 18/19, 2025 durch Abteilungsvorstand
Themenabgabe	Montag, 16. Juni 2025, 16.00 Uhr
Start Diplomarbeit	Montag, 15. September 2025
Abgabe Diplomarbeit	Montag, 10. November 2025, 16.00 Uhr
Upload Onlinepublikation	7 Tage vor Präsentation
Präsentationen	KW 48/49 2025
Diplomfeier	Freitag, 12. Dezember 2025, 19.00 Uhr

Ablauf

1. Orientierung

Zu Beginn des letzten Semesters orientiert dich der Abteilungsvorstand oder die Schulleitung über die Diplomarbeit, die einzuhaltenden Termine sowie erste Vorbereitungsarbeiten.

2. Themensuche

Anschliessend an die Orientierung suchst du ein Thema, welches du in deiner Diplomarbeit behandeln willst. Geeignet sind Themen aus deinem Berufsumfeld mit direktem Bezug zur Studienrichtung. In der Regel ist die Diplomarbeit eine Einzelarbeit, umfangreiche Diplomarbeiten können auch zu zweit erarbeitet werden.

3. Themeneingabe

Du reichst dein Thema gemäss der Vorlage «Themeneingabe» via Dateiupload im Extranet unter «Praktika» ein.

4. Themenabstimmung

In Absprache mit Abteilungsvorstand und Fachdozierenden wird das Thema beurteilt und wenn notwendig präzisiert bzw. abgegrenzt.

5. Start Diplomarbeit

Wir bestätigen dir dein gewähltes Diplomthema und den Namen der dich betreuenden Person vor Startdatum der Diplomarbeit. Das genaue Datum, Zeit und Ort der Diplomarbeitspräsentation teilen wir dir während der Erstellung der Diplomarbeit mit.

6. Betreuung

Vereinbare zu Beginn der Diplomarbeit mit der betreuenden Person zwei Vorzeigetermine. Den ersten Termin vereinbarst du am Anfang, damit die Zielvorgaben abgestimmt und die Aufgabenstellung besprochen werden können. Ein zweiter Vorzeigetermin nach der Hälfte der zur Verfügung stehenden Zeit dient dazu, den Stand der Arbeit zu kontrollieren und eventuelle Korrekturen anzubringen.

Besprechungspunkte:

- Standortbestimmung in Bezug auf die Zielsetzung
- Aufzeigen allfällig aufgetretener Probleme mit Lösungsvorschlägen
- Aufbau und Struktur der Dokumentation
- Fragen zur Vorbereitung der Präsentation und Onlinepublikation

7. Abgabe Diplomarbeit

- Je ein elektronisches Exemplar in einer pdf-Datei an das TEKO-Sekretariat via Upload im Extranet unter «Praktika» und – wenn verlangt – an die betreuende Person via E-Mail.
- Sofern die betreuende Person dies für die Korrektur verlangt: ein Exemplar gebunden im Format A4.

Der Dateiname des pdf-Dokuments muss wie folgt aufgebaut sein: DA_Jahr _Name_Vorname_Klasse.pdf.

8. Onlinepublikation

Zur Diplomarbeit gehört auch eine Onlinepublikation der Diplomarbeit. Diese wird auf einer von der TEKO zur Verfügung gestellten Plattform realisiert und dient dazu, deine Arbeit einem breiten Publikum nachhaltig zugänglich zu machen. Die Vorstellung der Onlinepublikation ist fester Bestandteil der Diplomarbeitspräsentation. Den Zugang zur Plattform werden wir dir 2 Wochen vor der Diplomarbeitspräsentation via Mail bekanntgeben.

Die Publikation umfasst folgende Inhalte: Kurzbeschreibung des Ergebnisses resp. des Produktes, Nutzen / Mehrwert für den Auftraggeber, Veranschaulichung mittels frei wählbarer Elemente (z.B. Fotos, Filme...).

Abbildung 3: FA19-Richtlinien Diplomarbeit_HF_2025 1/2

9. **Schlussbesprechung und Präsentation**
   Den Abschluss der Diplomarbeit bildet die Präsentation in Anwesenheit der betreuenden Person und des neutralen Experten. Du hast 15 Minuten Zeit für die Präsentation deiner Arbeit und 5 Minuten für die Beantwortung von Fragen der Experten. Der Inhalt der Präsentation richtet sich nach den in der Aufgabenstellung definierten Vorgaben bzw. nach den mit der betreuenden Person besprochenen Punkten.

Vertrauliche Daten in der Diplomarbeit

Vertrauliche Diplomarbeiten musst du durch die Schule schriftlich bewilligen lassen. Auf der Titelseite muss klar der Vermerk «vertraulich» sichtbar sein. Bei vertraulichen Arbeiten erhältst du die physischen Exemplare deiner Diplomarbeit nach der Rekursfrist zurück. Für die Onlinepublikation kannst du selbst bestimmen, welche Inhalte du zugänglich machen willst, oder ob deine Publikation offline bleiben soll. Die TEKO geht keine Vertraulichkeitsvereinbarungen irgendwelcher Art ein.

Form, Aufwand

Die Diplomarbeit verfasst du mit einem Textverarbeitungsprogramm im Format A4, einseitig beschrieben, fortlaufend nummeriert, gebunden oder in einem Ordner (wenn verlangt). Rechne mit ca. 150 – 250 Stunden Arbeitsaufwand. Die Arbeit umfasst eine schriftliche Dokumentation und je nach Aufgabenstellung bzw. Fachrichtung einen zeichnerisch-konstruktiven und/oder experimentellen Teil (z.B. Modell, Prototyp, Labor-Aufbau etc.).

Sofern Arbeitsunterlagen aus deiner Unternehmung verwendet werden, stelle sicher, dass du dazu das Einverständnis besitzt. Benützte Quellen sind lückenlos aufzuführen.

Gliederung der Diplomarbeit

Im Rahmen des Studiums hast du ein breites Grundlagenwissen im Projektmanagement aufgebaut und in Projekt- und Semesterarbeiten angewendet. Die Dokumentation der Diplomarbeit orientiert sich an genau diesen Grundsätzen. Die Arbeit soll beim Leser den Eindruck einer sinnvollen und geschlossenen Arbeit hinterlassen. Es muss ein «roter Faden» ersichtlich sein. Der chronologische Ablauf soll folgende Struktur aufweisen:

- Deckblatt: Titel der Diplomarbeit, Name der Schule, Name des Diplomanden, Ausbildung und Jahr, wenn vertraulich: Vermerk "vertraulich" auf der Titelseite
- Inhaltsverzeichnis
- Management Summary, max. 2 A4-Seiten
- Kurzer beruflicher Lebenslauf
- Aussagekräftiges Qualifikationsprofil (sofern verlangt, max. 2 A4-Seiten). Die Form ist grundsätzlich frei wählbar. Jedoch muss der Bezug zur Studienrichtung basierend auf dem Rahmenlehrplan klar erkennbar sein.
- Aufgabenstellung / Pflichtenheft / Zieldefinition
- Terminplan (soll/ist)
- Lösung der Aufgabe: Berichte / Programme / Ablaufpläne / Lösungswege / Entwürfe / Zeichnungen / Modelle / usw.
- Vollständige Quellenangaben und Literaturverzeichnis
- Reflexion Weg zum Ziel sowie "Lessons learned"
- Persönliches Schlusswort, Verdankungen und Eigenständigkeits-Erklärung

Bewertung der Diplomarbeit

Dein/e Betreuer/in bewertet deine Diplomarbeit anhand eines vorgegebenen Beurteilungsrasters (Schwierigkeitsgrad, Projektinitialisierung und -planung, Realisierung, Dokumentation, Präsentation, Onlinepublikation) in Abstimmung mit dem/der Experten/in. Bei Diplomarbeiten, welche im Auftrag eines Betriebes gemacht werden, fliesst die Bewertung der betrieblichen Fachperson in angemessener Form in die Gesamtbeurteilung ein. Bei Gruppenarbeiten kann jeder Diplomand einzeln bewertet werden. Die Schlussnote wird auf eine Zehntelsnote gerundet.

Diplomierung

Die Diplomarbeit ist bestanden, wenn du mindestens die Note 4.0 erreicht hast. Zur Erlangung des Diploms musst du zudem sämtliche finanziellen Verpflichtungen gegenüber der TEKO erfüllt haben. Bei Nichtbestehen kann die Diplomarbeit einmalig mit neuem Thema wiederholt werden. Du bist selbst für die Einhaltung sämtlicher Termine verantwortlich. Massgebend ist das Datum des Uploads oder E-Mail Versands. Zu spät eintreffende Arbeiten gelten als nicht eingereicht.

Fristerstreckung

Eine Fristerstreckung des Abgabetermins der Diplomarbeit kann nur bei nachgewiesener Krankheit mit Arztzeugnis oder Militärdienst mit Marschbefehl geltend gemacht werden.

Verwertungsrecht

Diplomarbeiten, welche im Besitze der TEKO bleiben, sind für folgende Verwendungen vorgesehen: Ausstellungen, Orientierungen, Anerkennungsverfahren und als Rekursbeweise.

Beschwerden und Rekurs

Gegen promotionsrelevante Noten können Sie innert 14 Tagen seit Bekanntgabe mit schriftlicher Begründung Rekurs erheben. Erste Rekursinstanz ist die Schulleitung.

Abbildung 4: FA19-Richtlinien Diplomarbeit_HF_2025 2/2

3.2 Auftrag Konkretisierung

Schweizerische Fachschule
TEKO

Auftrag zur Konkretisierung des Themas für die Diplomarbeit

Liebe/r Diplomand/in

Vor rund 2 Monaten hast du deinen Vorschlag für die Diplomarbeit beim TEKO-Sekretariat eingereicht. Von der Höheren Fachschule TEKO wurde ich als Diplomlehrer mit der Betreuung deiner Diplomarbeit beauftragt.

Damit du optimal in die Realisierung deiner Diplomarbeit starten kannst und ich dich darin optimal begleiten kann, muss im Vorfeld der folgende Auftrag bis spätestens **12. September 2025** erledigt, das Ergebnis im erstellten TEAMS-Kanal abgelegt und den Link per E-Mail an urs.gloggner@edu.teko.ch zugestellt werden:

Auftrag:

- Erstelle für deine Diplomarbeit ein Pflichtenheft mit folgenden Inhalten:

Inhalt	Beschreibung
1. Einleitung	In der Einleitung ist die Ausgangslage der Diplomarbeit mit folgenden Inhalten zu beschreiben: Bei Arbeiten im Auftrag einer Unternehmung: – Kurzvorstellung der Firma inkl. Kerngeschäft (max. 10 Sätze) – Überleitung mit der Antwort auf die Frage: Woraus ist die Idee für das Thema der Diplomarbeit entstanden? Bei Themen ausserhalb eines Betriebes (z.B. für private oder gemeinnützige Zwecke): – Kurzbeschreibung, was auf dem Markt bereits vorhanden ist und als Grundlage verwendet wird. – Überleitung mit der Antwort auf die Frage: Weshalb soll die Arbeit realisiert werden? Inwiefern unterscheidet sich das Ergebnis Ihrer Arbeit von bereits vorhandenen Produkten auf dem Markt?
2. Fachexperte	Bei Arbeiten im Auftrag einer Unternehmung ist eine Person aus dem Betrieb erforderlich, welche die technischen Anforderungen an die Arbeitsergebnisse definiert, das Pflichtenheft genehmigt und die Arbeit in Bezug auf die Lösung aufgrund der Vorgaben (Bewertungsraster) der TEKO bewertet. Bei Arbeiten ausserhalb eines Betriebes muss diese Rolle eine Person übernehmen, welche in diesem Arbeitsgebiet über Expertenwissen verfügt. Vom Fachexperten müssen folgende Informationen vorhanden sein: Vorname, Name, Beruf, Rolle/Funktion, Adresse, E-Mail, Telefon
3. Inhalt	– Richtziel – Ziele (Endergebnisse / Erfolgskriterien) inkl. detaillierte Beschreibung der Funktionalitäten sowie der technischen Anforderungen – schematische Skizzen: • System, Teilsystem und Umsystem • Schnittstellen • etc. – weitere Angaben, welche die Arbeit in technischer resp. methodischer und didaktischer Hinsicht konkretisieren resp. von der Firma vorgegeben sind – Für die Erstellung des Pflichtenheftes dürfen sofern vorhanden firmenspezifische Vorlagen verwendet werden.

1/3

Abbildung 5: Auftrag Vorbereitung Diplomarbeit Urs Gloggner 1/3

Schweizerische Fachschule
TEKO

Auftrag zur Konkretisierung des Themas für die Diplomarbeit

Begleitung während der Diplomarbeit

Die Abgabe der Aufgabenstellung sowie die Entgegennahme des vom Fachexperten unterschriebenen Pflichtenheftes und die Klärung allfälliger Fragen erfolgt durch den Diplomlehrer **online** mittels **TEAMS** am **15. September 2025 um 17:00 – 17:30 Uhr**.

Der erste Vorzeigetermin findet am **Montag, 22. September 2025** von **16:30 bis ca. 18:05 Uhr Online** mittels **TEAMS** statt:

16:30 – 16:45 Uhr	**Aschwanden Pirmin**
16:50 – 17:25 Uhr	**Egger Elik und Wallimann Quirin**
17:30 – 17:45 Uhr	**Fleischmann Simon**
17:50 – 18:05 Uhr	**Kuster Jonas**

Im Rahmen dieses Vorzeigetermins werden die folgenden Punkte innerhalb des vorgegebenen Zeitfensters besprochen:

Besprechungspunkt	Wer	Zeitdauer
1. Ziele mit konkreten Endergebnissen und Erfolgskriterien	Diplomand	4'
2. Projektstruktur- sowie Projektablaufplanung inkl. Meilensteine	Diplomand	4'
3. Aufzeigen allfälliger aufgetretenen Probleme (technische, logistische, theoretische etc.) mit Lösungsvorschlägen	Diplomand	4'
4. Fragen und Verschiedenes	Diplomand und Diplomlehrer	3'

Damit die Zeiten eingehalten werden können, ist eine seriöse Vorbereitung zwingend erforderlich.

2/3

Abbildung 6: Auftrag Vorbereitung Diplomarbeit Urs Gloggner 2/3

Schweizerische Fachschule
TEKO

Auftrag zur Konkretisierung des Themas für die Diplomarbeit

Der zweite Vorzeigetermin findet am **Montag, 20. Oktober 2025** von **16:30 bis ca. 18:05 Uhr Online** mittels **TEAMS** statt:

16:30 – 16:45 Uhr	**Aschwanden Pirmin**
16:50 – 17:25 Uhr	**Egger Elik und Wallimann Quirin**
17:30 – 17:45 Uhr	**Fleischmann Simon**
17:50 – 18:05 Uhr	**Kuster Jonas**

Besprechungspunkt	Wer	Zeitdauer
1. Standortbestimmung in Bezug auf die genehmigten Ziele inkl. allfällige Zielanpassungen	Diplomand	5'
2. Aufbau und Struktur der Dokumentation	Diplomand	5'
3. Fragen und Verschiedenes inkl. Abgabe des spezifischen Beurteilungsformulars für den Fachbetreuer	Diplomand	5'

Falls Fragen zu den Inhalten dieses Dokumentes auftreten, so stehe ich dir unter der E-Mail - Adresse urs.gloggner@edu.teko.ch oder telefonisch unter **+41-79-211 02 33** zur Verfügung.

Ich wünsche allen eine gute Vorbereitung, damit ihr in rund 2 Wochen optimal mit der Diplomarbeit starten könnt.

mit besten Grüssen

Urs Gloggner
Diplomlehrer und Experte

3/3

Abbildung 7: Auftrag Vorbereitung Diplomarbeit Urs Gloggner 3/3

3.3 Projektauftrag

TEKO
SCHWEIZERISCHE FACHSCHULE

Diplomarbeit

für Herren	***Kuster Jonas***
Abteilung	***Elektrotechnik***
Fachgebiet	***HW- und SW-Entwicklung***
Thema	***Motorrad-Dashboard***

Fachlehrer	Gloggner Urs
Prüfungsexperte	Schenker Jörg
Ausgabedatum	15. September 2025
Abgabedatum	10. November 2025

15.09.2025 *1/2*

Abbildung 8: Projektauftrag Urs Gloggner 1/2

TEKO

Aufgabenstellung

Mit der Diplomarbeit werden Sie das während dem gesamten Studium aufgebaute Grundlagenwissen an einer praxisorientierten Aufgabe in Ihrer Studienrichtung anwenden und vertiefen. Dabei geht es für Sie im Wesentlichen um die konsequente Zielorientierung, die Anwendung der Phasen des Projektmanagements sowie die konsequente Umsetzung von definierten Rahmenbedingungen.

Im Rahmen der Vorbereitungsarbeiten wurden Sie eingehend über diese Rahmenbedingungen sowie die Durchführung der Diplomarbeit informiert. Bei der Realisierung der Diplomarbeit müssen Sie nun den folgenden Punkten besondere Beachtung schenken:

- Die einzelnen Schritte, welche als Summe zum Ergebnis geführt haben, müssen beschrieben sein.
- Wurden im Rahmen der Lösungsfindung verschiedene Varianten entworfen und geprüft, muss die Evaluation der gewählten Variante mittels einer dazu geeigneten Methode dokumentiert und begründet sein.
- Bei der Dokumentation der Diplomarbeit ist auf einen logischen Aufbau, eine saubere Gliederung sowie auf gute Verständlichkeit zu achten. Details sind in den Richtlinien zur Diplomarbeit beschrieben.
- Der Value-add (Mehrwert) des Ergebnisses für den Auftraggeber muss aus der Dokumentation klar erkennbar und nachvollziehbar sein.
- Für die optimale Begleitung muss dem Diplomlehrer ein wöchentlicher Statusbericht (gemäss Vorlage Extranet TEKO) zugestellt werden. Dies ermöglicht dem Diplomlehrer den Verlauf unmittelbar mitverfolgen zu können und bei Bedarf gezielt zu reagieren.
- Der Fachexperte muss die Diplomarbeit gemäss den Kriterien auf dem durch den Diplomlehrer abgegebenen Bewertungsformular beurteilen. Der Fachexperte sendet das Formular bis spätestens eine Woche nach Abgabe der Diplomarbeit an den Diplomlehrer.
- Die Präsentation der Diplomarbeit muss so ausgestaltet sein, dass eine aussenstehende, fachunabhängige Person innerhalb von 15 Minuten die folgenden Punkte erkennen und bewerten kann:
  - Auftrag sowie Sinn und Zweck
  - Endergebnisse und Erfolgskriterien
  - besondere Herausforderungen auf dem Weg zum Ziel inkl. gewählte Lösungsansätze mit Begründung, jedoch ohne sich in Details zu verlieren
  - Demo des Produktes, Prototyps etc.
  - Reflexion: Zielerreichung, Weg zum Ziel, lessons learnt
- Im Anschluss an die Präsentation des Ergebnisses muss innerhalb von 5 Minuten die Onlinepublikation der Diplomarbeit vorgestellt werden. Die Onlinepublikation wird bewertet und muss so ausgestaltet sein, dass Aussenstehende in einer attraktiven, kreativen und leicht verständlichen Form sich über die erarbeitete Diplomarbeit informieren können.

Die Begleitung und die Bewertung der Diplomarbeit durch den Diplomlehrer richten sich konsequent nach dem vom Auftraggeber genehmigten und unterzeichneten Pflichtenheft, den im Dokument "Auftrag zur Konkretisierung des Themas für die Diplomarbeit" definierten Rahmenbedingungen sowie dem durch den Diplomlehrer abgegebenen Notenraster.

15.09.2025,

Abbildung 9: Projektauftrag Urs Gloggner 2/2

3.4 Analyse der Ausgangslage

Die Diplomarbeit der HF hat das Ziel, das im Unterricht vermittelte Grundlagenwissen praxisnah und eigenständig anzuwenden sowie zu vertiefen. Dabei ist die Diplomarbeit so aufgebaut, dass ich die erlernten Phasen des Projektmanagements gezielt anwende. Dies fördert nicht nur mein technisches Verständnis, sondern auch meine organisatorischen und planerischen Kompetenzen. Durch die Analyse der Ausgangslage konnte ich wertvolle Erkenntnisse gewinnen, die als Grundlage für die weiteren Kapitel dieser Arbeit dienen.

3.4.1 Problemfelder der Projektaufgabenstellung

Es gibt mehrere Problemfelder der Projektaufgabenstellung. Zusammengefasst kann man sie auf die folgenden Punkte einkürzen:

Wenn:

- die Ziele des Projekts nicht klar definiert sind oder unterschiedlich interpretiert werden können, kann dies zu Verwirrung und Missverständnissen führen.
- die erforderlichen Ressourcen wie Budget, Zeit, Personal oder technische Unterstützung nicht ausreichend bereitgestellt werden, kann dies die Durchführung des Projekts erschweren oder sogar unmöglich machen.
- der Zeitplan für das Projekt zu eng gesetzt oder unrealistisch ist, kann dies zu Stress, Überlastung und Qualitätsproblemen führen.
- es keine klaren Kriterien gibt, anhand deren der Erfolg des Projekts gemessen werden kann, kann dies zu Unsicherheit und fehlender Motivation führen.

3.4.2 Gescheiterte ähnliche Projekte

Die Gründe für das Scheitern ähnlichen Projekten sind vielfältig. Häufig liegen sie in den unterschiedlichen Interessen der beteiligten Teams oder in finanziellen Beschränkungen. Im folgenden Abschnitt haben wir die wichtigsten Ursachen zusammengefasst.

3.4.2.1 Verzögerungen bei der Materialbeschaffung

Lieferengpässe und Verzögerungen bei der Beschaffung benötigter Komponenten führen zu Verzögerungen im gesamten Projektablauf. Dadurch, dass viele Materialien direkt aus dem asiatischen Raum geliefert werden und einen weiten Lieferweg haben, kann es schnell zu Verspätungen kommen oder auch vorkommen, dass ein Paket verloren geht.

3.4.2.2 Ungenauigkeiten in Schaltplänen und Verdrahtungsfehler

Fehlerhafte oder unpräzise Schaltpläne können dazu führen, dass wichtige Verbindungen falsch hergestellt werden, was die Funktionstüchtigkeit des Endprodukts beeinträchtigt oder im schlimmsten Fall die Komponenten verbrennt. Ebenfalls führen diese Fehler dazu, dass zusätzliche Tests durchgeführt werden müssen, da Fehlfunktionen auftreten.

3.4.2.3 Kontaktprobleme an Lötstellen (kalte Lötstellen)

Kalte Lötstellen entstehen, wenn Lötverbindungen nicht richtig ausgeführt werden und es somit zu Kontaktproblemen kommt. Diese Kontaktprobleme können den Ausfall der Elektronik provozieren, was wiederum mühselige Test- und Fehlersuche bedeutet.

3.4.2.4 Einsatz ungeeigneter Bauteile

Der Einsatz von qualitativ minderwertigen oder falschen Bauteilen kann dazu führen, dass das System nicht die gewünschten Leistungsanforderungen erfüllt. Dies ist oft auf mangelnde Kenntnis über die Anforderungen oder auf Budgetkürzungen zurückzuführen, die den Kauf hochwertiger Komponenten verhindern.

3.4.2.5 Mangelhafte oder unzureichende Stromversorgung

Eine unzureichend ausgelegte Stromversorgung führt zu Instabilitäten im System und kann empfindliche Komponenten beschädigen. Es ist deshalb vorgängig genau abzuklären, welche Leistung die jeweiligen Bauteile benötigen, damit eine geeignete Versorgungsquelle gewählt werden kann.

3.4.2.6 Störungen durch unerwünschte Frequenzen

Fremdfrequenzen führen häufig zu unerklärlichen Störungen, welche nur sehr mühselig behoben werden können. Elektronische Geräte inklusive ihren Schaltkomponenten müssen schon vorgängig so angeordnet werden, dass sie sich gegenseitig nicht stören.

3.4.2.7 Fehlende oder unzureichende Testprozesse

Testprozesse spielen eine entscheidende Rolle, um die Funktionalität und Stabilität eines Systems sicherzustellen. Ohne gründliche Tests, die verschiedene Einsatzbedingungen simulieren, bleiben potenzielle Fehler unentdeckt und führen im späteren Betrieb oft zu Ausfällen. Da die verschiedenen Testverfahren für uns noch unbekannt sind, kann es auch aufgrund der geringen Kenntnisse zu fehlerhaften Testversuchen kommen.

3.4.3 Bestehende Angebote

Nachrüstbare Motorraddashboards auf dem Markt bieten heute eine Vielzahl von Funktionen, die weit über die Standardinstrumente eines Motorrads hinausgehen. Sie erfassen in Echtzeit relevante Fahr- und Umgebungsdaten und verbessern damit Sicherheit, Komfort und Übersicht für den Fahrer. Typische Geräte messen Temperaturen von Motor, Kühl- und Ölkreislauf sowie die Aussentemperatur und unterstützen so die Überwachung des Fahrzeugs. Lichtsensoren passen die Hintergrundbeleuchtung des Dashboards automatisch an die Lichtverhältnisse an und steuern teilweise auch Zusatzscheinwerfer, um die Lesbarkeit bei Tag und Nacht zu optimieren.

Moderne Nachrüst-Dashboards verfügen häufig über GPS-Modul und digitale Kompassfunktionen, die sowohl Navigation als auch Orientierung erleichtern. Zusätzlich bieten viele Geräte Anzeigen für Geschwindigkeit, Drehzahl, Batteriezustand und weitere wichtige Fahrzeugdaten. Einige Dashboards integrieren erweiterte Smart-Funktionen wie Warnmeldungen bei kritischen Temperaturen, Geschwindigkeitsalarme oder Hinweise zum Batteriestatus. Damit ermöglichen diese Systeme eine umfassende Kontrolle über das Motorrad und verbessern das Fahrerlebnis spürbar.

Auf dem Markt existieren Dashboards in unterschiedlichen Ausführungen, von einfachen Modellen mit wenigen analogen Anzeigen bis hin zu voll digitalen Displays mit Touchscreen und umfangreicher Sensorik. Die Geräte sind in der Regel mit den Bordspannungen der Motorräder kompatibel und lassen sich an Lenker oder Armaturenbrett nachrüsten. Hersteller achten zudem darauf, dass die Produkte den einschlägigen Sicherheitsnormen und Richtlinien entsprechen, sodass sie zuverlässig und störungsfrei betrieben werden können.

Insgesamt zeigt sich, dass nachrüstbare Motorraddashboards ein breites Spektrum an Funktionen vereinen und sich individuell an die Bedürfnisse des Fahrers anpassen lassen. Sie bieten nicht nur erweiterte Informationsmöglichkeiten, sondern erhöhen auch Komfort, Übersicht und Sicherheit während der Fahrt und sind daher ein zunehmend populäres Zubehör für Motorradfahrer.

3.4.4 Gesetzliche Grundlage

Ein selbstgebauter Zusatz-Dashboard, das auf dem Motorrad die Geschwindigkeit und andere Messwerte anzeigt, darf in der Schweiz grundsätzlich verwendet werden, sofern dadurch die vorgeschriebenen und zulassungspflichtigen Fahrzeuginstrumente nicht beeinträchtigt werden. Entscheidend ist, dass die Originalen Anzeigen jederzeit gut ablesbar, funktionsfähig und nicht verdeckt sind. Das Zusatzgerät darf deshalb lediglich als ergänzende Informationsquelle dienen und die primären Anzeigen nicht ersetzen.

Bei der Konstruktion und Montage ist darauf zu achten, dass das Gerät den Fahrer nicht ablenkt, keine blendenden oder blinkenden Effekte erzeugt und mechanisch so befestigt ist, dass ein Herunterfallen oder ein Lösen während der Fahrt ausgeschlossen ist. Ebenfalls wichtig ist, dass durch Verkabelung oder Anbringung keine sicherheitsrelevanten Teile des Motorrads blockiert oder verändert werden. Im Falle einer polizeilichen Kontrolle oder eines Unfalls kann das Zusatzgerät geprüft werden.

Für sicherheitskritische oder zulassungspflichtige Anzeigen empfiehlt es sich, auf nach ISO/ECE geprüfte Nachrüstlösungen zurückzugreifen. Versicherungsfragen sind im Schadensfall individuell zu beurteilen; grundsätzlich erhöht ein unsachgemäss angebrachtes Eigenbaugerät das Risiko von Haftungs- oder Deckungsklärungen.

3.4.5 Fazit

Die Analyse bestehender nachrüstbarer Motorraddashboards zeigt, dass theoretisch und praktisch erlerntes Wissen angewendet werden kann. Durch die Betrachtung der verschiedenen Funktionen von Temperatur- und Lichtsensoren über GPS- und Kompassmodul bis hin zu erweiterten Smart-Funktionen, werden sowohl technische als auch planerische Kompetenzen gefördert. Die Auseinandersetzung mit Rahmenbedingungen wie Kompatibilität, Montageanforderungen und Sicherheitsnormen hilft, potenzielle Herausforderungen frühzeitig zu erkennen und Lösungen zu planen. Eine sorgfältige Auswahl der Produkte, die Integration verschiedener Sensoren und eine präzise Dokumentation der Funktionen sind entscheidend, um das Projektziel erfolgreich zu erreichen und die Erkenntnisse für zukünftige Anwendungen nutzbar zu machen.

3.5 Pflichtenheft

Diplomarbeit Motorrad Dashboard

Pflichtenheft für die Diplomarbeit

Inhaltsverzeichnis

Abbildung 10: Pflichtenheft 1/6

Diplomarbeit Motorrad Dashboard

TEKO Schweizerische Fachschule

1. Einleitung

Auf dem Markt existieren bereits verschiedene Motorrad-Dashboards und GPS-Lap-Timer. Viele dieser Systeme sind jedoch in ihrer Funktionalität eingeschränkt, z. B. limitierte Rundenzahl beim Lap-Timer, ungenaue Tachoanzeigen oder fehlende Zusatzfunktionen wie Höhenmessung und Kompass.

Idee und Abgrenzung
Das geplante Motorrad Dashboard soll zusätzliche Funktionen vereinen, die in bestehenden Lösungen nicht oder nur teilweise verfügbar sind:
- GPS-basierte Geschwindigkeitsmessung.
- Speicherung von Geschwindigkeiten und Höhenwerten auf einer SD-Karte.
- Anzeige von Himmelsrichtungen mittels eines digitalen Kompasses.
- Lap-Time-Funktion, die die einzelnen Rundenzeiten anzeigt, sowie die Zeit vom ganzen Run.
- Speicherung von Maximalgeschwindigkeiten und Höhen.
- Datenübertragung und -analyse per WLAN

Das System unterscheidet sich somit von bestehenden Lösungen durch seine Vielseitigkeit, erweiterte Speichermöglichkeiten und die einfache Integration am Motorradlenker.

2. Fachexperte

Vorname: Andreas
Name: Holzer
Beruf: Dozent TEKO
Rolle/Funktion: Dozent Robotik & SPS
Adresse: Winkelriedstrasse 64, 6003 Luzern
E-Mail: andreas.holzer@edu.teko.ch
Telefon: 079 428 00 70

Abbildung 11: Pflichtenheft 2/6

Diplomarbeit Motorrad Dashboard

TEKO Schweizerische Fachschule

3. Ziele

Richtziel

Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.

Endergebnisse

1. GPS-Geschwindigkeit messen und auf der SD-Karte speichern.
2. Höhenanzeige + Speicherung der min./max. Höhen.
3. Kompassanzeige (N, NO, O, SO, S, SW, W, NW) mit Gradzahl.
4. Lap-Time Funktion: Messung, Anzeige und Speicherung.
5. WLAN-Zugriff auf gespeicherte Daten.
6. Das Motorrad Dashboard ist wetterfest und die Bedienung des Menüs ist über 4 Taster möglich.
7. Das Motorrad Dashboard ist am Lenkrad befestigt.

Erfolgskriterien

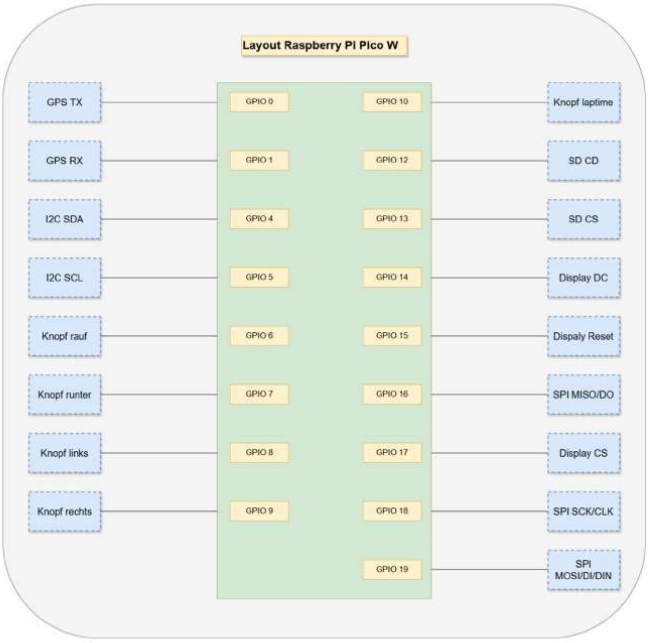
1. Abweichung zwischen GPS-Geschwindigkeit und Referenzmessung beträgt max. ±3 km/h und sind in lesbarer Textstruktur auf der SD-Karte gespeichert.
2. Anzeige zeigt die aktuelle Höhe in Metern über Meeresspiegel und speichert während einem Tag den niedrigsten und höchsten Wert.
3. Anzeige wechselt korrekt zwischen den 8 Haupt-Himmelsrichtungen. Genauigkeit der Gradzahl beträgt ±5°.
4. Daten sind im Lap-Time-Menü abrufbar und nach Fahrtende auf der SD-Karte verfügbar.
5. WLAN-Hotspot lässt sich über das Menü starten und zeigt die Gespeicherten Daten an.
6. Bedienung der Taster funktioniert mit Motorradhandschuhen. Gehäuse ist staub- und spritzwassergeschützt.
7. Das Dashboard ist stabil am Lenker fixiert, sodass es sich während der Fahrt nicht lockert.

Abbildung 12: Pflichtenheft 3/6

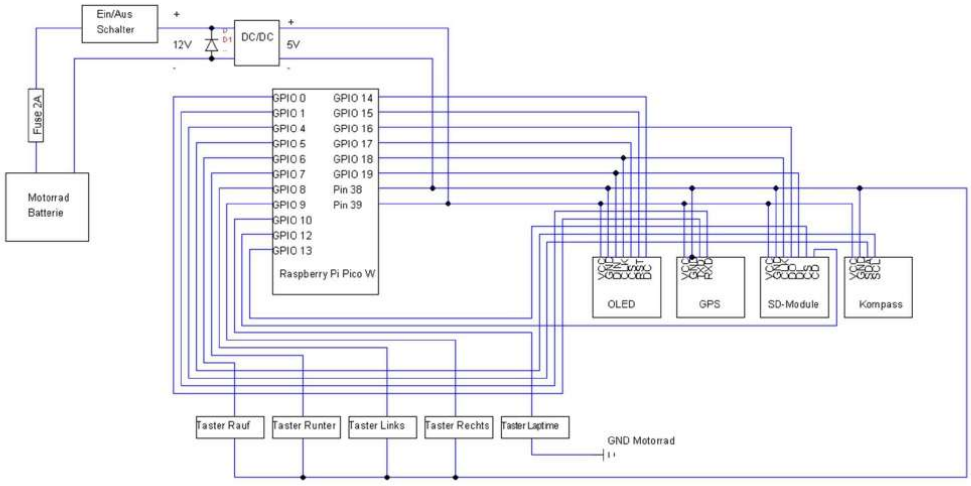
Diplomarbeit Motorrad Dashboard

Schematische Skizzen

Geplantes Layout:



Geplanter Schaltplan:



Jonas Kuster L-TEL-22-Do-a Seite 4

Abbildung 13: Pflichtenheft 4/6

Diplomarbeit Motorrad Dashboard

Geplante Ansteuerung:

Was	Ansteuerung	Pins
GPS- Modul	UART- Bus	TXD, RXD
Display	SPI- Bus	CLK, DIN, CS, RST
Magnetometer	I2C- Bus	SDA, SCL
SD- Module Adafruit	SPI- Bus	CLK, DO, DI, CS, CD
Taster	GPIO	Input

Weitere Angaben

- Technische Umsetzung mit Raspberry Pi Pico W, GPS, Magnetometer, Display und SD-Kartenmodul.
- Spannungsversorgung über 12V vom Motorrad mit Konverter auf 5V.
- Benutzerfreundliches Bedienkonzept mit 8 Menü Ansichten.
- Gehäuse mit Display und 4 bedienknöpfe. Lap- Time Taster in der Nähe des linken Griffs.

Jonas Kuster L-TEL-22-Do-a Seite 5

Abbildung 14: Pflichtenheft 5/6

Diplomarbeit Motorrad Dashboard

4. Unterschrift

Diplomand: Jonas Kuster Jonas. K Datum: 04.09.2025

Fachexperte: Andreas Holzer Datum: 11.9.2025

Abbildung 15: Pflichtenheft 6/6

3.6 Zielscheibe

Richtziel:
Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.

Endergebnisse

1. Das Motorrad Dashboard ist wetterfest, gut am Lenker befestigt und die Bedienung des Menüs ist über 5 Taster möglich.
2. Das Motorrad Dashboard wird über die 12V-Motorradbatterie gespeist und ist mit Verpolungs- und Überlastschutz ausgestattet.
3. Das Motorrad Dashboard hat ein Ansichtsmenü, in dem man die:
   - 3.1. GPS-Geschwindigkeit ablesen kann.
   - 3.2. Höhen anzeigen kann.
   - 3.3. Himmelsrichtung per Kompass und per Gradzahl bestimmen kann.
   - 3.4. Rundenzeit messen kann.
   - 3.5. Gespeicherten Rundenzeiten einsehen kann.
   - 3.6. Maximale und minimale Höhe und die Höchstgeschwindigkeit per tag einsehen kann.
   - 3.7. Gespeicherten Daten per WLAN einsehen kann.
4. Das Programm ist mit einem Zustandsdiagramm visuell dargestellt.

Kunde

- TEKO
- Jonas Kuster

Sinn und Zweck

Die Arbeit dient dazu, Fahrdaten zu sammeln und auszuwerten sowie die Orientierung durch zusätzliche Informationen wie Kompass und Höhenmessung zu verbessern. So entsteht ein praktisches Tool für sportliche Analysen und sichere Navigation auf Touren.

Erfolgskriterien

1. Die Bedienung der Taster funktioniert mit Motorradhandschuhen, das Gehäuse ist staub- und spritzwassergeschützt und stabil am Lenker fixiert, sodass es sich während der Fahrt nicht lockert.
2. Das Motorrad Dashboard funktioniert an 12V, ist mit einer 2A-Sicherung abgesichert und besitzt eine Crowbar-Diode, die bei Falschpolung die Sicherung auslöst.
3. Das Ansichtsmenü des Motorrad Dashboards erfüllt:
   - 3.1. Die GPS-Geschwindigkeit hat zu einer Referenzmessung nicht mehr als ± 3km/h Abweichung.
   - 3.2. Die Höhe ist gegenüber einer Referenzmessung auf ± 5m genau.
   - 3.3. Die Anzeige stimmt mit einem Referenzkompass überein und zeigt eine maximale Abweichung von ±5°.
   - 3.4. Es werden die aktuelle, die Letze und die Schnellste Runde von einem Turn angezeigt.
   - 3.5. Das Speicherlayout der Rundenzeiten werde von 80% der Personen als benutzerfreundlich angesehen.
   - 3.6. Das Speicherlayout der Extremwerte werde von 80% der Personen als benutzerfreundlich angesehen.
   - 3.7. Die gespeicherten Daten können über das WLAN auf einem Smartphone eingesehen werden.
4. Es ist ein Zustandsdiagramm erstellt, welches die Logik, Menüs und Anzeigezustände des Programms abbildet.

Abbildung 16: Zielscheibe

4 Projektplanung

Die Projektplanung bildet die Grundlage für eine strukturierte und effiziente Umsetzung meines Vorhabens. Zu den zentralen Bestandteilen zählen der Projektstrukturplan, der das Projekt in einzelne Arbeitspakete unterteilt, sowie das Gantt-Diagramm, das die zeitliche Abfolge und Abhängigkeiten zwischen den Aufgaben visualisiert. Diese Werkzeuge ermöglichen eine transparente Planung, eine gezielte Ressourcennutzung sowie die termingerechte Umsetzung der Projektziele. Im folgenden Kapitel sind die entsprechenden Pläne aufgeführt.

4.1 Projektstrukturplanung

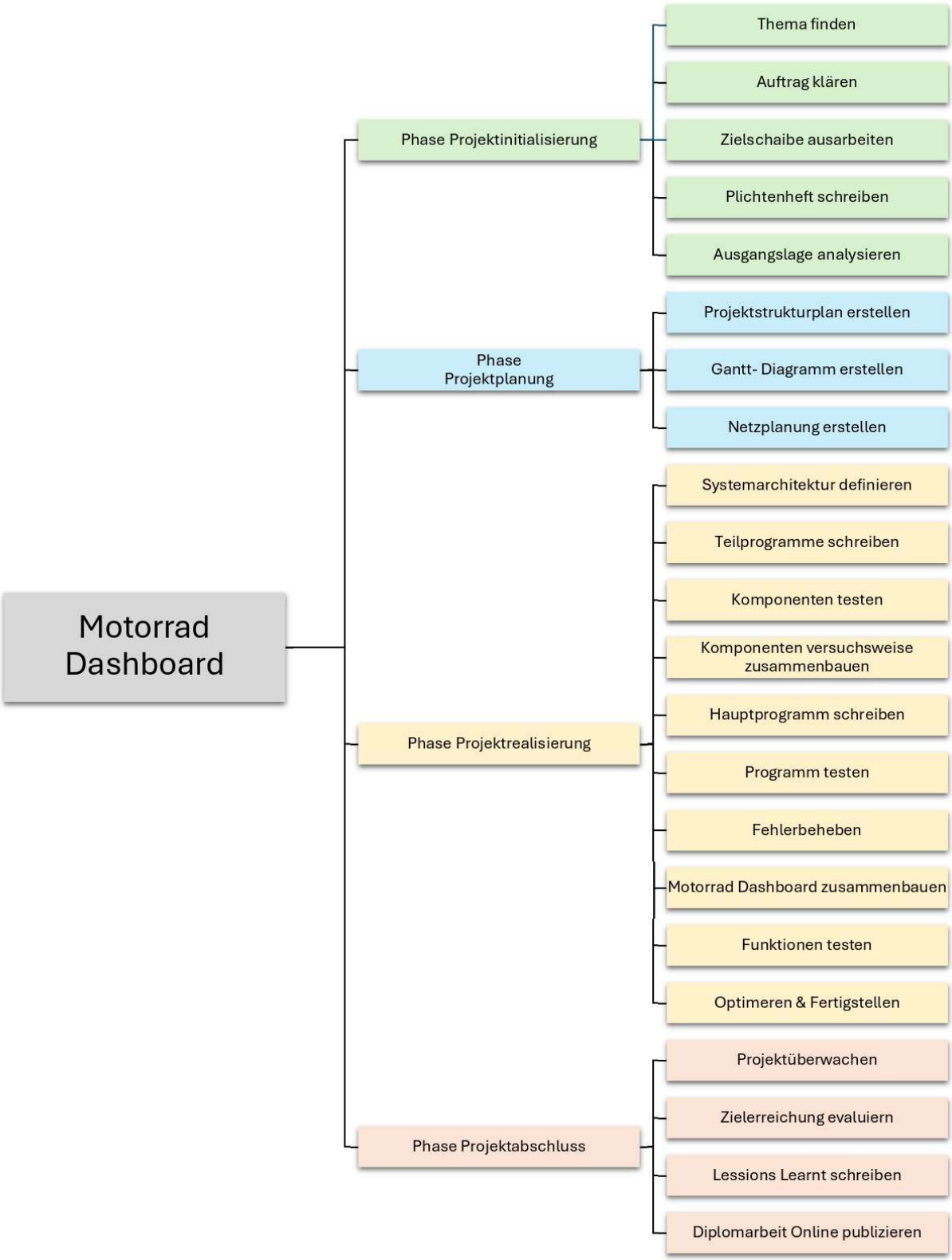


Abbildung 17: Projektstrukturplan

4.2 Projektablaufplanung

4.2.1 Gantt Diagramm

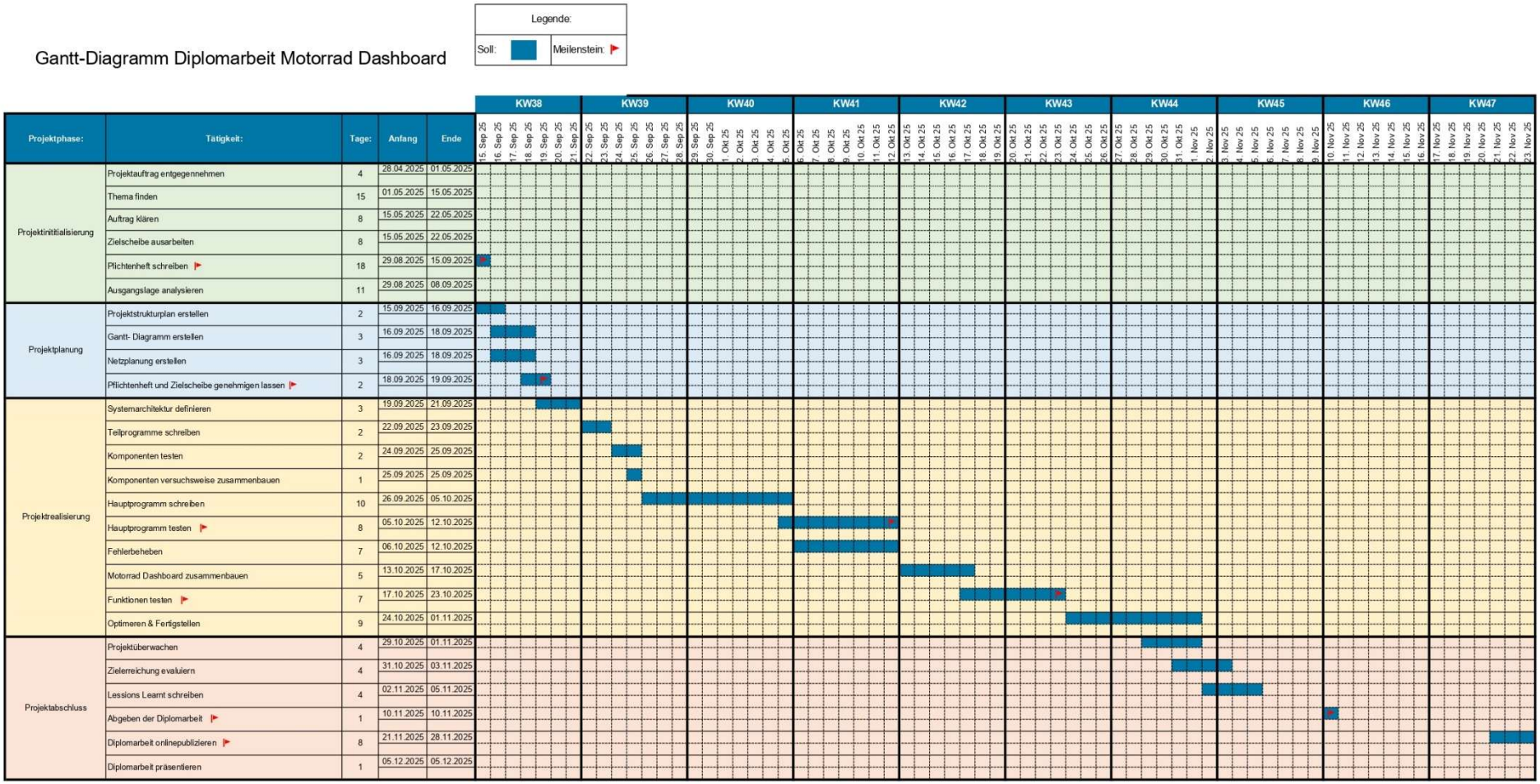


Abbildung 18: Gantt Diagramm

4.2.2 Netzplan

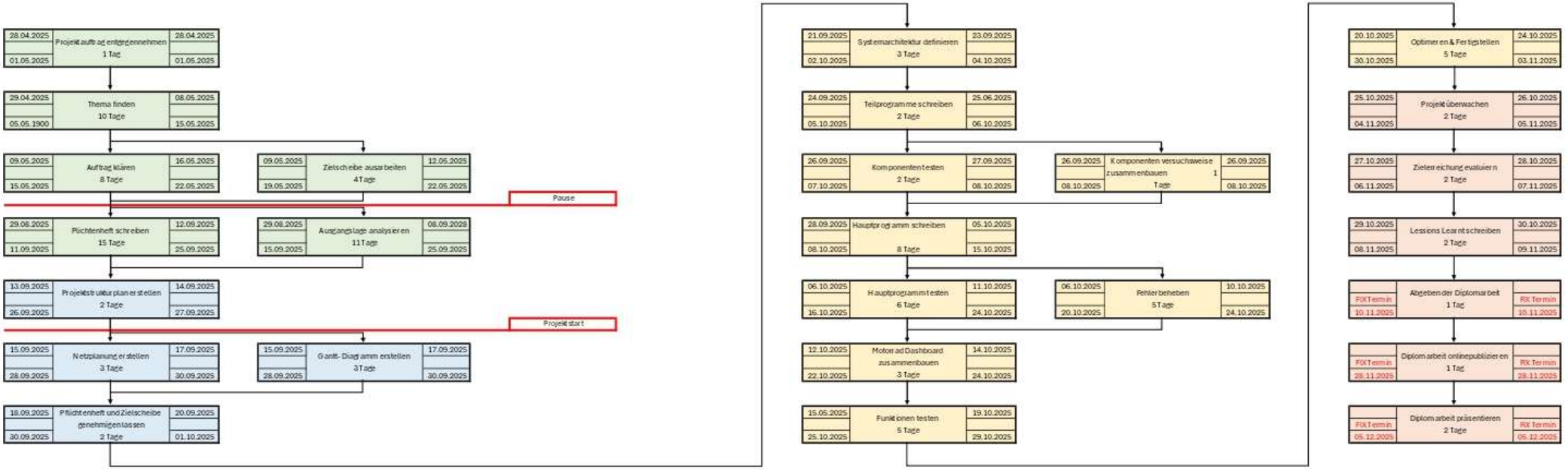


Abbildung 19: Netzplan

5 Projektrealisierung

In diesem Kapitel gebe ich einen umfassenden Einblick in den Entwicklungsprozess meines Projekts. Beginnend mit der Themenfindung und der Ausarbeitung der Projektidee, zeige ich, wie die Grundlagen für die weitere Planung geschaffen wurden. Darauf aufbauend werden die Schema- und Designentwürfe vorgestellt, die eine erste Struktur und das optische Konzept festlegten. Anschliessend erläutere ich die durchgeführten Überlegungen und wie diese zur Erstellung des finalen Produkts beigetragen haben. Nach der Auswahl der passenden Komponenten, dem Programmieren und dem Zusammenbau gehe ich auf die Gebrauchsanleitung ein, um die Anwendung zu erleichtern. Abschliessend dokumentiere ich die Kostenkontrolle und weitere abschliessende Massnahmen. Dieses Kapitel bildet somit den gesamten Prozess von der Idee bis zur finalen Umsetzung ab.

5.1 Themenfindung / Themenausarbeitung

Zu Beginn meines Projekts stellte ich mir die Frage, wie ich am Besten ein geeignetes Thema für meine Diplomarbeit finden könnte. Mir war es wichtig, ein Thema zu wählen, das sowohl meine fachlichen Interessen abdeckt als auch einen persönlichen Bezug hat. Um meine Gedanken zu sammeln und mögliche Themengebiete zu erkunden, entschied ich mich für die Methode des Mind-Maps. Diese Technik ermöglichte es mir, ohne feste Struktur einfach erste Ideen zu notieren und diese später zu verfeinern oder zu erweitern. Für mich war das besonders hilfreich, da ich so auf eine visuelle und intuitive Weise meine Interessen ordnen konnte.

5.2 Mind-Map

Um ein geeignetes Thema für meine Diplomarbeit zu finden, begann ich damit, meine Gedanken mithilfe einer Mind-Map zu strukturieren. Ziel war es, verschiedene Interessenfelder und mögliche Projektideen zu erfassen, um daraus ein sinnvolles, praxisnahes und zugleich spannendes Thema abzuleiten. Das Mind-Map diente mir als visuelles Hilfsmittel, um meine Ideen übersichtlich darzustellen und Zusammenhänge zwischen verschiedenen Bereichen zu erkennen. Ich entschied, mich auf fünf Hauptgruppen zu konzentrieren: Motorrad, Zuhause, Robotik, Gadget und Geschäft. Diese Kategorien deckten sowohl persönliche als auch fachliche Interessen ab und boten eine gute Grundlage für die weitere Themenfindung.

Schon bald richtete sich mein Fokus verstärkt auf den Bereich Motorrad, da mich dieses Thema nicht nur aus technischer Sicht interessiert, sondern auch eine starke persönliche Relevanz hat. Ich fahre selbst Motorrad und beschäftige mich regelmässig mit technischen Verbesserungen und Individualisierungen. Innerhalb dieses Zweiges des Mind-Map markierte ich mir die aus meiner Sicht interessantesten Themen mit einem roten Kreis. Diese waren: Lap-Time, Höhenmesser, Kompass, Geschwindigkeit sowie Log-Daten.

Ich entschied mich bewusst gegen die Integration einer Beschleunigungs- oder Neigungswinkelerfassung, da ich mit solchen Funktionen in der Vergangenheit schlechte Erfahrungen gemacht habe.

Besonders die Themen Kompass und Höhenmesser fand ich interessant, da ich auf meinem Motorrad kein integriertes Navigationssystem habe. Ein digitaler Kompass und eine Höhenanzeige würden mir helfen, mich unterwegs besser zu orientieren. Mit der Anzeige der Himmelsrichtung wüsste ich jederzeit, in welche Richtung ich fahre, und durch die Höhenmessung hätte ich zusätzlich Informationen über den Streckenverlauf.

Die Funktionen Lap-Time und Geschwindigkeit wiederum wählte ich mit Blick auf eine geplante Nutzung auf der Rennstrecke. Mein aktuelles Motorrad verfügt zwar bereits über einen integrierten Lap-Time, dieser ist jedoch auf maximal 30 Runden begrenzt und bietet keine Möglichkeit zur Speicherung oder detaillierten Auswertung der Daten. Darüber hinaus ist die standardmässige Geschwindigkeitsanzeige bei Motorrädern aufgrund gesetzlicher Vorschriften meist mit einer Sicherheitstoleranz von etwa 5 Prozent nach oben versehen. Ich wollte daher eine präzisere, GPS-basierte Lösung implementieren, die mir verlässlichere Werte liefert.

Ein weiterer zentraler Punkt war für mich die Möglichkeit, alle erfassten Daten auf einer SD-Karte zu speichern, um sie später am Computer auslesen und analysieren zu können. Dadurch ergeben sich zusätzliche Nutzungsmöglichkeiten, etwa zur Darstellung der gefahrenen Strecke, zur Auswertung von Geschwindigkeit und Höhenprofilen oder zur Optimierung meines Fahrverhaltens auf der Rennstrecke.

Insgesamt kristallisierte sich somit ein klarer Themenbereich heraus, der sowohl technisch anspruchsvoll als auch persönlich motivierend ist. Die Entwicklung eines eigenen, erweiterten Motorrad Dashboards, das mir nicht nur während der Fahrt wichtige Informationen liefert, sondern auch Möglichkeiten zur späteren Analyse und Weiterentwicklung bietet.

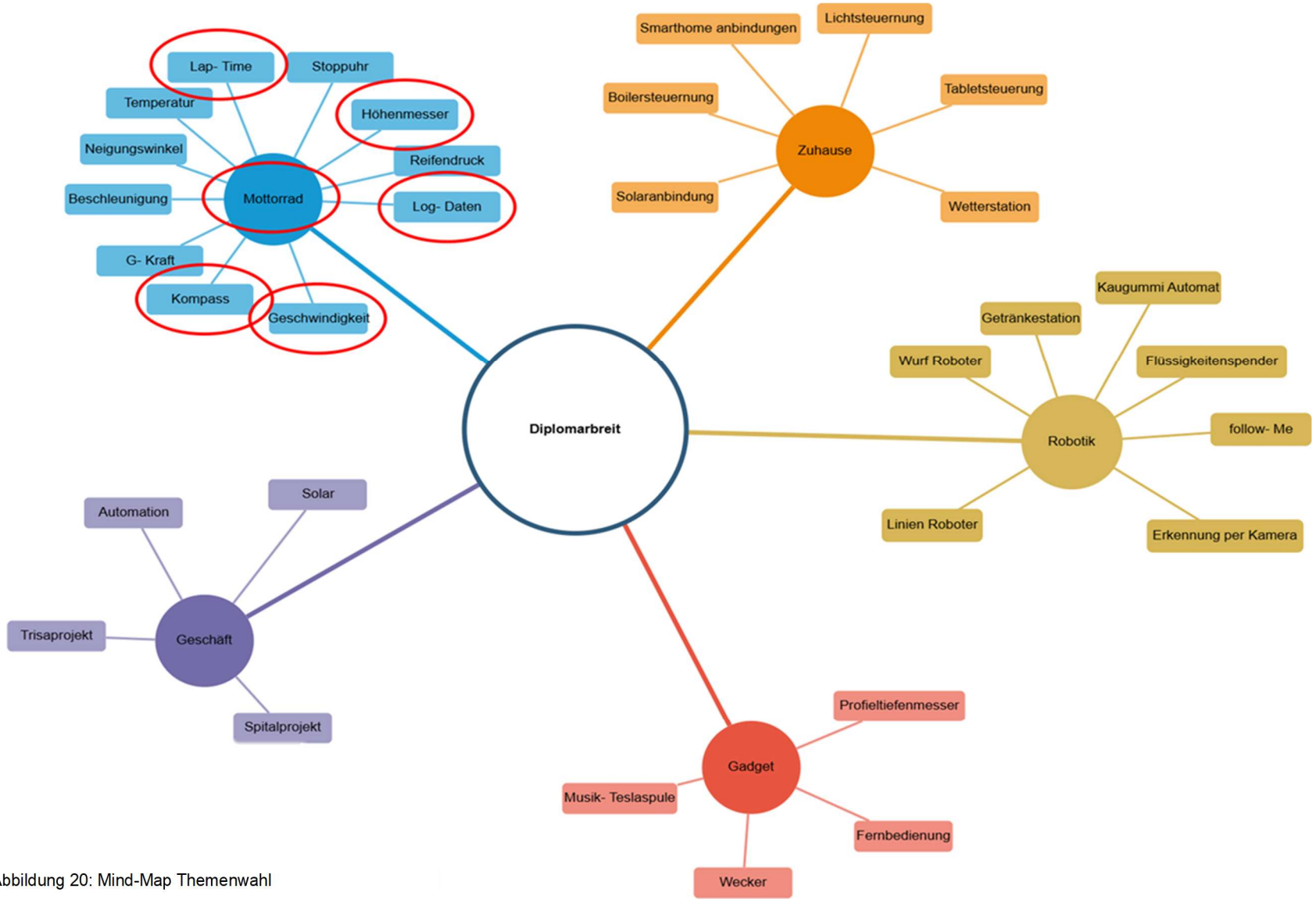


Abbildung 20: Mind-Map Themenwahl

5.3 Themeneingabe

Um meine Idee für die Diplomarbeit einzugeben habe ich die Ansätze für ein Motorrad Dashboards vom Mind-Map zusammengefasst und diese bei der TEKO eingereicht. In diesem Dokument beschrieb ich kurz, was ich machen möchte, was es können soll und wie die Erfolgskriterien aussehen könnten.

TEKO
SCHWEIZERISCHE FACHSCHULE

Die Themeneingabe kann als Word-Formular im Extranet unter „Praktika" heruntergeladen werden.

Themeneingabe Diplomarbeit

Name	Kuster
Vorname	Jonas
Adresse, Ort	Wybärgstrasse 14
Tel: P, G	+41 79 568 89 69
E-Mail	Jonas.kuster@edu.teko.ch
Klasse	L-TEL-22-Do-a
Fachgebiet	Elektrotechnik
Auftraggeber	TEKO

Thema	Motorrad Dashboard
Kurzbeschrieb	Als Motorradfahrer möchte ich meine Fahrdaten erfassen und auswerten. Das serienmässige Dash ist dafür zu eingeschränkt. Es speichert nur 30 Lap- Runden, zeigt ungenaue Geschwindigkeit und keine Höhe oder Himmelsrichtung. Für Rennstrecke und Tour will ich daher ein eigenes System entwickeln, das alle relevanten Daten zuverlässig anzeigt und speichert. Ziel ist es, ein multifunktionales Motorrad-Dashboard zu entwickeln, das GPS-basierte Daten wie Geschwindigkeit, Höhe und Position erfasst, die Himmelsrichtung digital anzeigt, Rundenzeiten speichert und alle Informationen auf einer SD-Karte ablegt. Die Anzeige erfolgt über ein Display, das zwischen mehreren Ansichten umgeschaltet werden kann. Alle Funktionen sollen über wasserdichte Taster am Lenker bedienbar sein. Die Elektronik wird vibrationsresistent verbaut und aus der Motorrad-Batterie versorgt. Damit möchte ich ein zuverlässiges, erweiterbares und robustes System realisieren, das sowohl für den Strassen als auch den Trackeinsatz geeignet ist.
Erfolgskriterien	Die GPS-Geschwindigkeits anzeige, die Höhenanzeige, der Kompass, der Lap-Timer, die Datenspeicherung und die Anzeigemodi sind Realisiert. Rundenzeiten inkl. max. Geschwindigkeit & Höhenunterschied werden korrekt erfasst und gespeichert. System ist vibrations und wetterresistent. Projekt ist vollständig dokumentiert.

Abbildung 21: Themeneingabe

5.4 Entwurf

Als meine Idee für die Diplomarbeit bewilligt wurde erstellte ich aus dem Mind-Map zunächst ein Entwurfsdokument, in dem ich meine Ideen zur Umsetzung grob festhielt. Dieser Prozess verlief fliessend, da mir immer wieder neue Gedanken oder Verbesserungsvorschläge kamen, die ich ergänzte. So wuchs das Dokument Schritt für Schritt, bis daraus schliesslich das Konzept für das Motorrad Dashboard entstand. Zu Beginn waren darin noch mehr Themen enthalten, als ich später tatsächlich verfolgte, auch solche, die nicht in meinem Mind-Map rot markiert waren. Dies lag daran, dass ich zunächst alle Ideen ungefiltert notierte und mir erst im weiteren Verlauf des Schreibens gezielt Gedanken darüber machte, welche Aspekte ich tatsächlich umsetzen wollte. Auf diesem Dokument ist auch das erste Mal ersichtlich, wie ich die verschiedenen Anzeigen strukturieren möchte und wie das Speicher-Layout in etwa aussehen soll.

```
Motorrad Dash
-------------
Idee Erklärung:
Mein Motorrad hat ein eingebauter Lap-Timer aber der geht nur auf 30 runden.
Ich möchte die echt Geschwindigkeit per GPS messen da mein Tacho nicht genau ist.
Es interessiert mich auf was für Höhen ich auf Pässen so unterwegs bin.
Für meine Orientierung auf Reisen mit dem Motorrad wäre ein Kompass Ideal

Idee 1: 27.03.2025
-------
- GPS Geschwindigkeit
- höhenmesser
- Himmerlsrichtung anzeige (N,NO,O,SO,S,SW,W,NW)
- G-Kraft Anzeige
- Neigungswinkel anzeige (im guten Bereich grüne zahl, danach rot)
- Alle daten speichern auf einer Speicherkarte, min und max behalten bis Reset, dann werte wieder auf 0 stellen
- 12V Speisung ab Motorrad Batterie, System mit 5V
- Montage an Lenkrad
- alles auf einer Platine
- Max und min Werter des Neigungswinkels und G-Kräfte Speichern

- Taster für Kalibrierung Neigung
- Taster für Reset auf Standardwerte => löscht max und min auf SD-Karte
- Ein- Aus Schalter
- Mode Taster: Umschalten zwischen nur Neigung, G-Anzeige, Himmelsrichtung, gespeicherte min und max werte oder alle variablen Werte
- Backlight Taste, Einstellung der Display Helligkeit (ein, aus)
(muss wasserdicht sein)

Probleme
- Vibration => Gumminoppen zur Montage
- Kalibrieren der Neigung => Standardeinstellung auf einer Geraden Fläche Setzen
- Messwerte Stabilisieren => Kalman-Filter oder Komplementärfilter zur Stabilisierung der Messwerte
- Kalibrierung auf die Montage am Motorrad

Bauteile
- für Himmelsrichtung HMC5883L oder QMC5883L
- Neigung und G-Kräfte MPU6050
- Display OLED i2C oder SPI (WaveShare 128x160 1.8inch LCD-Display Modul, LCD)
- Speicher über microSD-Modul =>(Purecrea Micro SD Card Reader Modul)
- Mikrocontroller RP Pico H
- Speisung 12V zu 5V LM2596 => MP1584 ist kleiner
- Diode für Verpolungsschutz
- Display mit Plexiglas- oder Gorilla-Glas-Abdeckung für Schutz
- Wasserfestes Gehäuse (Plastik oder Aluminium)
- 5x Drucktaster wasserdicht mit Dichtungsring

Frontplatte:

- Ausschnitt für Display mit Schutzglas
- Knöpfe Wasserfestes

Display                 Taster
_________________________
¦                       ¦   o   Ein-Aus
¦   Neigungswinkel      ¦   o   Kalibrierung Neigungswinkel
¦                       ¦   o   Reset Standardwerte
¦   G-Anzeige           ¦   o   Mode Taster
¦                       ¦   o   Backlight
¦   Himmelsrichtung     ¦
¦_______________________¦

Kritisches Problem
- Die Neigung lässt sich nur bei gleichbleibender Gweschwindigkeit messen
- Heisst, in der Kurve Gas geben verfälscht die Neigungsanzeige


Idee 2: 27.03.2025
-------

- GPS Geschwindigkeit
```

Abbildung 22: Entwurf Seite 1

```
- Höhenanzeige
- Himmelsrichtungsanzeige (N,NO,O,SO,S,SW,W,NW mit Gradanzeige)
- Lap time
- Lap time daten auf SD Karte speichern Zeit+ Max Speed pro runde(separate 5V Einspeisung)=>(Purecrea Micro SD Card Reader Modul)
- 12V Speisung ab Motorrad Batterie, System mit 5V
- Montage an Lenkrad
- Taster für Löschen=> löscht lap-daten auf SD-Karte
- Ein-Aus Schalter
- Lap-Taster( Langklick=> Stopp / Kurzklick=> Start, runde
- Mode Taster: Umschalten zwischen Geschwindigkeit, Höhe, Himmelsrichtung, Lap-Time, gespeicherte Lap-Time oder alle anzeigen
- Backlight Taste, Einstellung der Display Helligkeit (ein, aus)
- Montage an Lenkrad nähe Dash

Frontplatte:

- Auschnitt für display mit Schutzglas
- Knöpfe Wasserfestes

Display                 Taster
_________________________
¦                       ¦   o   Ein-Aus
¦   GPS Geschwindigkeit ¦   o   Lap-Taster (evt. extern, nähe bei Grif)
¦                       ¦   o   Lösch Taster
¦   Höhenanzeige        ¦   o   Mode Taster (evt. mit Steuerkreuz, Probleme zum Lap-Time scrollen)
¦                       ¦   o   Backlight
¦   Himmelsrichtung     ¦
¦                       ¦
¦   Lap-Time            ¦
¦_______________________¦

Lap-Time Speicher Layout:

Run 1
1. 1.30.55 / 210km/h
2. 1.31.36 / 212km/h
3. 1.39.89 / 209km/h
usw....
Run 2
1. 1.30.55 / 210km/h
2. 1.31.36 / 212km/h
3. 1.39.89 / 209km/h
usw....
Run 3
1. 1.30.55 / 210km/h
2. 1.31.36 / 212km/h
3. 1.39.89 / 209km/h
usw....

- bei jedem Neustart der Lap-Time titel (Run X) setzen

Probleme
- Vibration => Gumminoppen zur montage o. ähnliches
- Wo Montage

Bauteile
- für Himmelsichtrichtung HMC5883L oder QMC5883L
- Display OLED i2C oder SPI (WaveShare 128x160 1.8inch LCD-Display Modul, LCD)
- Speicher über microSD-Modul =>(Purecrea Micro SD Card Reader Modul) max und min werte speichern als festen speicher
- Mikrocontroller RP pico H
- Speisung 12V zu 5V MP1584 ist kleiner
- Überspannungsschutz (TVS-Diode)
- Sicherung (1.5A-2A)
- Diode für Verpolungsschutz
- Display mit Plexiglas- oder Gorilla-Glas-Abdeckung für Schutz
- Wasserfestes Gehäuse (Plastik oder Aluminium)
- 5x Drucktaster wasserdicht mit Dichtungsring

Idee 3: 01.05.2025
-------
```

Abbildung 23: Entwurf Seite 2

```
140
141   - GPS Geschwindigkeit
142   - Höhenanzeige
143   - Himmelsrichtungsanzeige (N,NO,O,SO,S,SW,W,NW mit Gradanzeige)
144   - Tour time
145   - Tour time daten auf SD-Karte speichern Zeit+ Max Speed pro runde(separate 5V
      Einspeisung)=>(Purecrea Micro SD Card Reader Modul)
146   - 12V Speisung ab Motorrad Batterie, System mit 5V
147   - Montage an Lenkrad
148   - Taster für Löschen=> löscht Tour-daten auf SD-Karte
149   - Ein-Aus Schalter
150   - Tour Taster( Langklick=> Tour fertig / Kurzklick=> Start, pause
151   - Mode Taster: Umschalten zwischen Geschwindigkeit, Höhe, Himmelsrichtung, Tour-time,
      gespeicherte Tour-time oder alle anzeigen
152   - Backlight Taste, Einstellung der Display Helligkeit (ein, aus)
153   - Montage an Lenkrad nähe dash
154
155   Frontplatte:
156
157   - Ausschnitt für Display mit Schutzglas
158   - Knöpfe Wasserfestes
159
160   Display                 Taster
161   _________________________
162   ¦                       ¦   o   Ein-Aus
163   ¦   GPS Geschwindigkeit ¦   o   Lap-Taster (evt. extern, nähe bei Grif)
164   ¦                       ¦   o   Lösch Taster
165   ¦   Höhenanzeige        ¦   o   Mode Taster (evt. mit Steuerkreuz, Probleme zum
      Tourtime scrollen)
166   ¦                       ¦   o   Backlight
167   ¦   Himmelsrichtung     ¦
168   ¦                       ¦
169   ¦   Lap- Time           ¦
170   ¦_______________________¦
171
172   Tour- Time Speicher Layout:
173   Tour 1
174   Fahrt 1.30.55 / 210km/h
175   Pause 1.31.36
176   Fahrt 1.39.89 / 209km/h
177   usw....
178   Tour 2
179   Fahrt 1.30.55 / 210km/h
180   Pause 1.31.36
181   Fahrt 1.39.89 / 209km/h
182   usw....
183
184   - bei jedem Neustart der Tour-time titel (Tour X) setzen
185
186   Probleme
187   - Vibration => Gumminoppen zur Montage o. ähnliches
188   - Wo Montage
189
190   Bauteile
191   - für Himmelsichtrichtung HMC5883L oder QMC5883L
192   - Display OLED i2C oder SPI (WaveShare 128x160 1.8inch LCD-Display Modul, LCD)
193   - Speicher über microSD-Modul =>(Purecrea Micro SD Card Reader Modul) max und min
      werte speichern als festen speicher
194   - Mikrocontroller RP Pico H
195   - Speisung 12V zu 5V MP1584 ist kleiner
196   - Überspannungsschutz (TVS-Diode)
197   - Sicherung (1.5A-2A)
198   - Diode für Verpolungsschutz
199   - Display mit Plexiglas- oder Gorilla-Glas-Abdeckung für Schutz
200   - Wasserfestes Gehäuse (Plastik oder Aluminium)
201   - 5x Drucktaster wasserdicht mit Dichtungsring
202
203   Idee 4: 01.05.2025
204   -------
205
206   - GPS Geschwindigkeit
207   - Höhenanzeige
208   - Himmelsrichtungsanzeige (N,NO,O,SO,S,SW,W,NW mit Gradanzeige), auf Einzel anzeige
```

Abbildung 24: Entwurf Seite 3

```
mit digitalem Kompass
- Lap time, mit Gesamtzeit anzeige
- Lap time daten auf SD karte speichern (Zeit+ Max Speed+ Max Höhenunterschied pro runde)(separate 5V Einspeissung)=>(Purecrea Micro SD Card Reader Modul)
- Max min speichern Höhe, Max speichern Geschwindigkeit
- 12V Speisung ab Motorrad Batterie, System mit 5V
- Montage an Lenkrad
- Taster für Löschen=> löscht daten auf SD-Karte
- Ein-Aus Schalter
- Lap-Taster(Kurzklick=> Start, runde / Langklick=> Stopp)
- Mode Taster: Umschalten zwischen Geschwindigkeit, Höhe, Himmelsrichtung, Lap-time, gespeicherte Lap-Time oder alle anzeigen + daten scrollen
- Backlight Taste, Einstellung der Display Helligkeit (ein, aus)
- Montage an Lenkrad nähe Dash

Frontplatte:

- Ausschnitt für Display mit Schutzglas
- Knöpfe Wasserfestes

Display Hauptmenü           Steuerkreuz+Taster
_____________________
¦                     ¦   o   Ein-Aus
¦   GPS Geschwindigkeit ¦   o   Lap-Taster (evt. extern, nähe bei Grif)
¦                     ¦   o   Lösch- taster (kombination mit lap taster lap löschen, sonst nur max min löschen)
¦   Höhenanzeige      ¦   o   Mode Taster (evt mit mit Steuerkreuz, probleme zum Laptime scrollen, links rechts=>blätern, hoch runter=>scrollen)
¦                     ¦   o   Backlight
¦   Himmelsrichtung   ¦   o   Lösch-Taster (Kombination mit Lap-Taster, Lap löschen,sonst nur max und min)
¦                     ¦
¦                     ¦
¦_____________________¦

Lap-Time Speicher Layout: Lap, min, sec, zehntel, Geschwindigkeit, Höhenunterschied, am Schluss Gesamtzeit vom Run

Run 1 (01.05.2025 14:25)
1. 1.30.55 / 210kph / 20m
2. 1.31.36 / 212kph / 21m
3. 1.39.89 / 209kph / 20m
usw....
Gesamtzeit: 22.52.33

Run 2 (02.05.2025 11:54)
1. 1.30.55 / 210kph / 20m
2. 1.31.36 / 212kph / 20m
3. 1.39.89 / 209kph / 20m
usw....
Gesamtzeit: 21.57.52

Run 3 (02.05.2025 14:56)
1. 1.30.55 / 210kph / 21m
2. 1.31.36 / 212kph / 20m
3. 1.39.89 / 209kph / 21m
usw....
Gesamtzeit: 23.01.21

- bei jedem Neustart der Lap-Time titel (Run X + Datum und Zeit per GPS abrufen) setzen

min/max Höhe und max Geschwindigkeit Layout:
514m / 3250m / 215kph

Probleme
- Vibration => Gumminoppen zur Montage o. ähnliches
- Wo Montage

Bauteile
- für Himmelsichtrichtung QMC5883L
- Display OLED i2C oder SPI (WaveShare 128x160 1.8inch LCD-Display Modul, LCD)
- Speicher über microSD-Modul =>(Purecrea Micro SD Card Reader Modul) max min werte
```

Abbildung 25: Entwurf Seite 4

```
    speichern als festen speicher
274 - Mikrocontroller RP Pico H
275 - Speisung 12V zu 5V MP1584 ist kleiner
276 - Überspannungsschutz (TVS-Diode)
277 - Sicherung (1.5A-2A)
278 - Diode für Verpolungsschutz
279 - Display mit Plexiglas- oder Gorilla-Glas-Abdeckung für Schutz
280 - Wasserfestes Gehäuse (Plastik oder Aluminium)
281
282 ┌──────────────────────┐
283 │                      │
284 │                      │
285 │                      │
286 │  GPS Geschwindigkeit │
287 │                      │
288 │                      │
289 │                      │
290 └──────────────────────┘
291
292 ┌──────────────────────┐
293 │                      │
294 │                      │
295 │                      │
296 │  Höhenanzeige        │
297 │                      │
298 │                      │
299 │  min         max     │
300 └──────────────────────┘
301
302 ┌──────────────────────┐
303 │                      │
304 │                      │
305 │  Lap-Time            │
306 │                      │
307 │  Letzte Runde        │
308 │                      │
309 │  Schnellste Runde    │
310 └──────────────────────┘
311
312 ┌──────────────────────┐
313 │Himmelsrichtung in Grad│
314 │          N           │
315 │                      │
316 │  W       O----   O   │
317 │                      │
318 │          S           │
319 │                      │
320 └──────────────────────┘
321
322
```

Abbildung 26: Entwurf Seite 5

5.5 Konzept Funktionen Motorrad Dashboard

Nachdem ich mich auf das Thema Motorrad Dashboard konkretisiert hatte, begann ich mit der konzeptionellen Planung meines Projekts. Mein Ziel war es, ein System zu entwerfen, welches mehrere nützliche Funktionen in einem kompakten, alltagstauglichen Gerät vereint, das direkt am Motorrad montiert werden kann. In einem ersten Entwurf habe ich die Anforderungen und die gewünschten Funktionen festgehalten, um eine klare Vorstellung davon zu bekommen, wie das Endprodukt aussieht und funktionieren soll.

Das geplante Dashboard sollte in erster Linie grundlegende Fahrdaten erfassen und anzeigen. Dazu zählt insbesondere die Geschwindigkeit, die mittels GPS ermittelt werden soll. Parallel dazu sollte auch die aktuelle Höhe über dem Meeresspiegel aufgezeichnet werden, was ebenfalls über das GPS-Modul erfolgen kann. Beide Werte Geschwindigkeit und Höhe sollten dabei laufend auf einer SD-Karte gespeichert werden, sodass sie zu einem späteren Zeitpunkt auf einem Computer analysiert werden können.

Ein weiterer zentraler Bestandteil meines Konzepts war die Anzeige der Himmelsrichtung über einen digitalen Kompass. Diese Funktion ist besonders hilfreich für Tourenfahrten, da sie auch ohne Navigationssystem eine grobe Orientierung ermöglicht. Die Anzeige sollte neben den klassischen Himmelsrichtungen (N, NO, O, SO, S, SW, W, NW) auch den Winkel in Grad anzeigen, sodass jederzeit eine präzise Ausrichtung ablesbar ist.

Darüber hinaus war es mir wichtig, das Dashboard mit einer Lap-Time-Funktion auszustatten. Diese soll es ermöglichen, Rundenzeiten zu messen, das vor allem auf der Rennstrecke von grossem Nutzen ist. Neben der reinen Zeitmessung sollten dabei auch zusätzliche Daten wie die maximale Geschwindigkeit und der grösste Höhenunterschied pro Tag erfasst und gespeichert werden. Alle gespeicherten Daten sollten im Gerät über ein Menü aufrufbar sein, damit ich jederzeit einen Überblick über die wichtigsten Werte erhalte.

Für die Bedienung des Systems plante ich vier Taster, die wie ein Steuerkreuz angeordnet sein sollten, um die Navigation durch das Menü so intuitiv wie möglich zu gestalten. Die Taster links und rechts am Steuerkreuz sollen die Menüansichten wechseln, während die Taster auf und ab das Scrollen im Menü ermöglichen soll. Ergänzend dazu sollte ein Lap-Time Taster am Lenkrad montiert werden, der fünf Funktionen erfüllt. Wenn man im Menü bei der Lap-Time Funktion ist, soll ein zwei sekundier Tastendruck zum Stoppen der aktuellen Messung sein, während ein kurzer Tastendruck das Starten der Messung oder Setzen einer neuen Runde auslösen sollte.

Wenn man in einem der beiden Speichermenüs ist, muss ein drei sekundier Tastendruck das Löschen des Speichers verursachen. Bei der Hauptanzeige im Untermenü sollte ein kurzer Tastendruck das WLAN aus oder einschalten. Zusätzlich sollte ein Ein-/Ausschalter vorgesehen werden, da das System direkt an die 12-Volt-Batterie des Motorrads angeschlossen ist. Die Betriebsspannung für die verwendeten Komponenten liegt jedoch bei 5 Volt. Dies möchte ich mit einem Spannungsregler realisieren.

Mittels WLAN ist es vorgesehen, eine Verbindung zum Raspberry Pi Pico W aufzubauen und direkt auf den Inhalt der SD-Karte zuzugreifen. So könnten gespeicherte Daten bequem ohne Ausbau der SD-Karte eingesehen werden. Ein zusätzlich geplanter Vorteil besteht darin, die erfassten GPS-Daten beispielsweise über Google My Maps zu übertragen, um die gefahrene Strecke grafisch auszuwerten.

5.6 Konzept Menü Motorrad Dashboard

Um die während der Fahrt gesammelten Informationen nicht nur aufzuzeichnen, sondern auch direkt am Gerät anzeigen und nachvollziehen zu können, konzipierte ich ein Menüsystem. Ziel war es, dem Benutzer eine einfache Möglichkeit zu bieten, alle wichtigen Daten direkt über das Display des Dashboards abzurufen.

Das Menü sollte in verschiedene Ansichtsseiten unterteilt werden, zwischen denen man bequem mit Hilfe von Tastern nach links und rechts wechseln kann. Dadurch ist es möglich, während der Fahrt oder im Stand zwischen verschiedenen Informationsansichten hin und her zu schalten. Die Navigation soll einfach und selbsterklärend sein, um eine sichere Bedienung auch mit Handschuhen und während kurzer Stopps zu ermöglichen.

Im Zentrum des Menüs steht die Hauptanzeige. Diese zeigt die drei wichtigsten Livedaten gleichzeitig an. Die aktuelle GPS-Geschwindigkeit, die momentane Höhe über dem Meeresspiegel und die aktuelle Himmelsrichtung. Diese Ansicht dient als Standardansicht während der Fahrt, da sie die wichtigsten Informationen kompakt zusammenfasst. Zusätzlich sollte im Untermenü, das durch den Knopf runter erreichbar ist, das WLAN ein oder ausgeschaltet werden. Wenn das WLAN eingeschalten ist, wird ein WLAN-Hotspot über den Raspberry Pi Pico aufgebaut, wobei die zugewiesene IP-Adresse angezeigt wird. Über diese IP-Adresse kann anschliessend bequem per HTML-Browser auf das Gerät zugegriffen werden, um die gespeicherten Daten direkt auszulesen.

Darüber hinaus gibt es mehrere sogenannte Nebenanzeigen, die jeweils spezielle Detailinformationen liefern.

Auf der ersten Nebenanzeige wird ausschliesslich die Geschwindigkeit angezeigt. Diese reduzierte Anzeigeform soll eine bessere Lesbarkeit bieten, wenn der Fokus nur auf der Fahrgeschwindigkeit liegt.

Auf der zweiten Nebenanzeige wird die aktuelle Höhe angezeigt, ergänzt durch den bisher niedrigsten und den höchsten aufgezeichneten Höhenwert seit Beginn der Fahrt. Diese Werte geben einen Überblick über das Höhenprofil der aktuellen Route und können insbesondere bei Fahrten im Gebirge interessant sein.

Auf der dritten Nebenanzeige soll die Himmelsrichtung in Grad angezeigt werden. Zu dieser Anzeige ergänzend, soll ein digitaler Kompass hinzugefügt werden. Diese Ansicht ist vor allem hilfreich zur Orientierung, wenn kein Navigationssystem verwendet wird.

Auf der vierten Nebenanzeige werden Informationen zur aktuellen Runde angezeigt. Dazu gehört die laufende Rundenzeit, die zuletzt gefahrene Rundenzeit sowie die schnellste aufgezeichnete Runde der Session.

Auf der fünften Nebenanzeige sind alle bisher gespeicherten Rundenzeiten ersichtlich. Damit lassen sich vergangene Runden nachvollziehen und vergleichen, ohne dass dafür ein Computer notwendig ist.

Auf der sechste Nebenanzeige werden die maximale Höchstgeschwindigkeit, die höchste Höhe sowie die niedrigste Höhe, die während der Fahrt gespeichert wurden, angezeigt. Diese Informationen liefern interessante Einblicke über den gesamten Fahrverlauf.

Die Navigation zwischen diesen sieben Anzeigen wird nicht nur seitlich mit dem linken und rechten Taster möglich sein, sondern auch vertikal. Mit einem oberen und einem unteren Taster kann durch gespeicherte Werte, wie in den Rundendaten, gescrollt werden.

Zur Verdeutlichung und besseren Planung habe ich eine grafische Skizze angefertigt, die das geplante Menüsystem zeigt. Diese diente mir als visuelle Grundlage für die Programmierung und half dabei, die Menüstruktur logisch und übersichtlich zu gestalten. Mein Ziel war es, ein Bedienkonzept zu entwickeln, das sowohl funktional als auch benutzerfreundlich ist und gleichzeitig die wichtigsten Informationen schnell zugänglich macht, ohne den Fahrer zu überfordern.

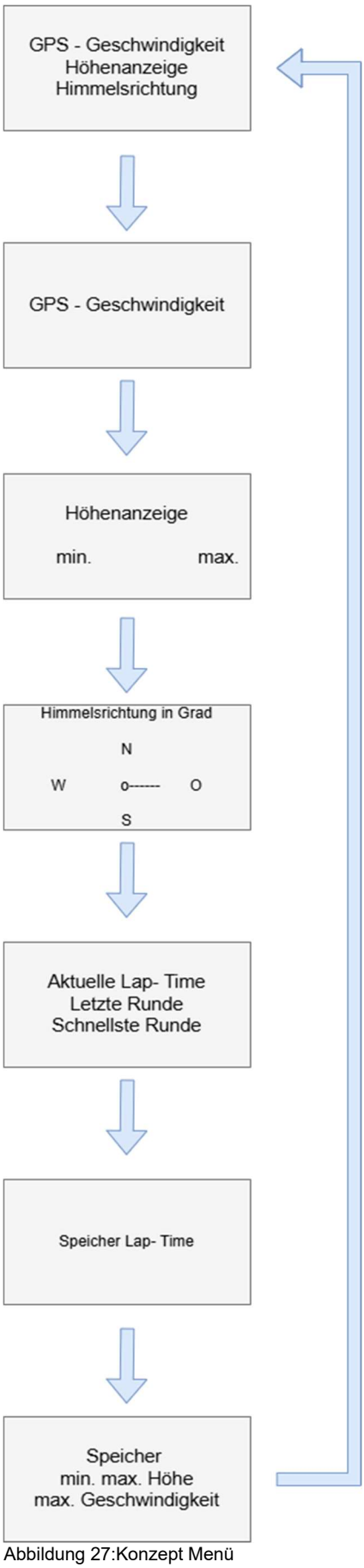


Abbildung 27:Konzept Menü

5.7 Konzept Speicher Motorrad Dashboard

Für das Speichern der während der Fahrt gesammelten Daten auf der SD-Karte habe ich mir im Vorfeld ausführlich Gedanken gemacht. Ziel war es, die Daten nicht nur technisch korrekt zu erfassen, sondern auch so zu strukturieren, dass sie später leicht ausgewertet, gelesen und sinnvoll miteinander verglichen werden können. Ein wesentlicher Bestandteil des Systems ist die Lap-Time Funktion, also die Aufzeichnung der einzelnen Rundenzeiten während einer Session. Jede Runde soll dabei mit einer fortlaufenden Nummer versehen werden. Zusätzlich sollen die Minuten, Sekunden und Zehntelsekunden angezeigt werden. Damit ist es möglich, die Performance einzelner Runden im Detail zu analysieren. Ein weiterer wichtiger Punkt ist die zeitliche Einordnung der Daten. Um nachvollziehen zu können, an welchem Tag welcher Turn gefahren wurden, wird jede Session mit dem entsprechenden Datum versehen. Dieses Datum soll dabei direkt aus dem GPS-Modul entnommen werden. Zusätzlich zur Rundenzeit wird auch die Gesamtzeit der Session berechnet. Diese gibt an, wie lange die gesamte Fahrt gedauert hat. Ein hilfreicher Wert zum Vergleich mit vorherigen Fahrten. Auch diese Information wird am Ende jeder Fahrt gespeichert und in einer gut lesbaren Struktur zusammengefasst.

Das folgende Beispiel zeigt, wie das Layout für die Speicherung der Lap-Time auf der SD-Karte aussehen soll:

Turn 1 (02.05.2025)
1: 1.30.55
2: 1.31.36
3: 1.39.89
usw...
Gesamtzeit: 22.52.33

Turn 2 (02.05.2025)
1: 1.30.55
2: 1.31.36
3: 1.39.89
usw...
Gesamtzeit: 21.57.52

Turn 3 (03.05.2025)
1: 1.30.55
2: 1.31.36
3: 1.39.89
usw...
Gesamtzeit: 23.01.21

Neben den Rundenzeiten sollen auch bestimmte Extremwerte gespeichert werden, wie etwa die minimale und maximale Höhe sowie die höchste Geschwindigkeit, die während eines Tages erreicht wurde. Auch für diese Werte möchte ich ein klares Layout verwenden. Hierfür wird der jeweilige Tag aus dem GPS übernommen und zusammen mit den genannten Werten aufgezeichnet. Für die Geschwindigkeit habe ich mich bewusst dafür entschieden, die Einheit kph (kilometres per hour) zu verwenden, da diese Schreibweise in meinen Augen kompakter und leserlicher ist als das in Europa übliche km/h (Kilometer pro Stunde).

Das geplante Layout für die Speicherung dieser Extremwerte sieht wie folgt aus:

01.05.2025:
514m / 1250m / 215kph

04.05.2025:
321m / 861m / 125kph

08.07.2025:
632m / 2634m / 360kph

Mit dieser Form der Datenspeicherung verfolge ich das Ziel, meine Fahrdaten strukturiert, übersichtlich und langfristig nachvollziehbar zu erfassen. Durch die einfache Textstruktur ist es auch möglich, die Daten später ohne viel Aufwand auf einen Computer zu übertragen und in Tabellen oder Diagramme einzulesen. So lassen sich Fahrten im Nachhinein detailliert auswerten.

Um die Lap-Time oder die Extremwerte zu löschen, soll man ins jeweilige Menü gehen können und dann den Lap-Taster einige Zeit lang gedrückt halten.

5.8 Technischer Aufbau

5.8.1 Komponenten Auswahl

Um mit den Planungsarbeiten für mein Projekt beginnen zu können, musste ich mich zunächst für die passenden Komponenten entscheiden. Dabei legte ich besonderen Wert auf Kompatibilität mit dem Raspberry Pi Pico, auf kompakte Bauweise sowie auf bereits vorhandene Erfahrungen aus dem Unterricht.

5.8.1.1 Mikrocontroller: Raspberry Pi Pico

Die Wahl des Mikroprozessors fiel auf den Raspberry Pi Pico, da wir im Unterricht intensiv mit diesem gearbeitet haben und ich mich mit seiner Programmierung bereits gut auskenne. Ausserdem bietet er eine Vielzahl von Schnittstellen wie UART, I^2C und SPI, die für die spätere Ansteuerung von Sensoren notwendig sind.

5.8.1.2 GPS-Modul zur Geschwindigkeits- und Höhenmessung

Zur Messung von Geschwindigkeit und Höhe benötigte ich ein GPS-Modul. Nach meiner Recherche im Internet entschied ich mich für ein Modul von Purecrea. Dieses Modul wird häufig verwendet, wodurch es eine grosse Community gibt, die bereits damit gearbeitet hat, was bei der Fehlersuche und bei der Umsetzung sehr hilfreich ist. Ein weiteres wichtiges Merkmal dieses Moduls ist die integrierte Batterie, die die Verbindung stabiler und weniger störungsanfällig macht. Angesteuert wird das GPS-Modul über den UART-Bus, was vom Raspberry Pi Pico W unterstützt wird.

5.8.1.3 Magnetometer für die Kompassfunktion

Für die Orientierung mithilfe eines Kompasses habe ich mich für ein Magnetometer, ebenfalls von Purecrea, entschieden. Dieses Modul wird über den I^2C-Bus angesteuert, welcher ebenfalls vom Raspberry Pi Pico W unterstützt wird. Besonders überzeugt hat mich bei diesem Sensor der geringe Stromverbrauch sowie die kompakte Bauform. Da ich das Dashboard möglichst platzsparend bauen möchte, war das ein entscheidender Faktor bei der Auswahl.

5.8.1.4 OLED-Display zur Datenanzeige

Zur Anzeige der Messwerte verwende ich ein OLED-Display von WaveShare mit einer Auflösung von 128x128 Pixeln, was einer Diagonale von 1,5 Zoll beziehungsweise einer Grösse von etwa 27x27 mm entspricht. Diese Grösse ist ausreichend für meine Anzeigezwecke. Ich habe mich bewusst für ein OLED entschieden, da diese im Vergleich zu LCDs auch bei Sonnenlicht gut ablesbar sind. Dies ist so weil ein OLED den besseren Kontrast hat als ein LCD. Was ein grosser Vorteil bei direkter Sonneneinstrahlung ist. Das Display wird über den SPI-Bus angesteuert.

5.8.1.5 Bedienung: Wasserdichte Taster

Für die Bedienung des Menüs habe ich mich für vier wasserdichte Drucktaster der Marke RND entschieden. Diese lassen sich auch mit Handschuhen gut bedienen, was für den Einsatz am Motorrad wichtig ist. Für die Lap-Time-Funktion habe ich zusätzlich einen separaten Taster direkt am Lenkrad montiert. Diesen habe ich im Motorradzubehörshop Polo gekauft. Es handelt sich um einen Taster der Marke Hashiru. Ich habe mich für diesen entschieden, da er gut auf mein Motorradlenkrad passt.

5.8.1.6 Datenspeicherung: microSD-Karte mit Kartenleser

Zur Speicherung der Daten habe ich mich für eine 2GB microSD-Karte und ein SD-Kartenlesemodul von Purecrea entschieden. Leider habe ich dieses Modul bei einem Test beschädigt. Ich wollte eine 5V-Spannung anlegen, habe dabei aber versehentlich VCC und GND vertauscht. Seitdem war das Modul nicht mehr ansprechbar. Daraufhin habe ich ein neues, noch kleineres Kartenlesemodul von Adafruit gekauft. Dieses Modul lässt sich sowohl mit 3,3V als auch mit 5V betreiben und bringt zusätzlich integrierte Pullup-Widerstände für den SPI-Bus mit, was ein grosser Vorteil ist. Auch dieses Modul wird, wie das Display, über den SPI-Bus angesteuert.

5.8.1.7 Stromversorgung: DC-DC-Wandler

Um mein Dashboard zu betreiben, benötige ich eine stabile Stromversorgung. Hierfür habe ich einen DC-DC-Wandler von Purecrea verwendet. Dieser kann Eingangsspannungen von 4,5V bis 28V in eine Ausgangsspannung von 0,8V bis 18V umwandeln. Die Ausgangsspannung lässt sich bequem über ein integriertes Potentiometer einstellen. Die Eingestellte Spannung am Ausgang wird stabilisiert. Der Wandler ist kompakt gebaut, produziert wenig Wärme und kann mit bis zu 3 Ampere belastet werden, was für mein Projekt mehr als ausreichend ist.

5.8.1.8 Schutzmassnahmen: Sicherung und Verpolungsschutz

Um die Stromversorgung abzusichern, habe ich eine 2A Flachsicherung von Littelfuse eingesetzt. Diese ist für Kfz-Anwendungen gedacht und passt in einen Flachsicherungshalter, den ich direkt an der Batterie anschliessen möchte. Um zusätzlich einen Verpolungsschutz sicherzustellen, habe ich eine Crowbar-Schaltung eingebaut. Dabei handelt es sich um eine Schottky-Diode, die zwischen Plus und Minus nichtleitend geschaltet ist. Wird die Spannung jedoch falsch herum angeschlossen, wird die Diode leitend und erzeugt absichtlich einen Kurzschluss, woraufhin die Sicherung auslöst. Ich habe mich für eine Schottky-Diode mit 5A Belastbarkeit und 100V Spannungsfestigkeit entschieden. Diese ist bewusst etwas überdimensioniert, damit im Fehlerfall die Sicherung und nicht die Diode beschädigt wird. Ein weiterer Vorteil dieser Lösung ist, dass keine Spannung über der Diode verloren geht und keine zusätzliche Wärmeentwicklung stattfindet, im Gegensatz zu klassischen Schutzdioden.

5.8.1.9 Ein-/Ausschalter und Verbindung der Komponenten

Da das Dashboard direkt an die Batterie angeschlossen ist, habe ich ausserdem einen wasserdichten Kippschalter verbaut, mit dem sich das System einfach ein- und ausschalten lässt. Zur Verbindung aller Komponenten habe ich mich für eine Lochrasterplatte entschieden. Auf dieser kann ich die Bauteile fest verlöten und bei Bedarf trotzdem einfach austauschen oder neu verbinden. Das bietet Flexibilität für spätere Änderungen oder Reparaturen.

5.8.1.10 Dimensionierung

Um sicherzustellen, dass die Dimensionierung meiner Komponenten korrekt ist, habe ich eine Berechnung des maximalen Stromverbrauchs durchgeführt. Dafür habe ich für alle aktiven Bauteile den jeweils höchsten Stromverbrauch recherchiert und diese Werte anschliessend addiert.

Was	**Maximaler Stromverbrauch**
Raspberry Pi WH	160 mA
GPS-Modul	60 mA
Display	35 mA
Magnetometer	0.7 mA
Konverter 12V-5V	2 mA
SD-Modul Adafruit	150 mA
Total	407.7 mA

Abbildung 28:Berechnung

Das Ergebnis zeigt, dass mein System im Extremfall nur etwa 407.7mA benötigt. Ich habe jedoch eine Stromversorgung mit 2 A vorgesehen. Das bedeutet, dass ich einen Sicherheitsfaktor von rund 5 eingeplant habe. So bin ich auf der sicheren Seite und die Schutz- und Versorgungskomponenten sind ausreichend dimensioniert.

Hier ist die Liste der verschiedenen Komponenten, mit denen ich arbeiten werde:

Was	**Verwendung**
Wippschalter IP65	Ein/Aus Schalten des Dashboards
Drucktaster IP65	Für die Steuerung des Menüs
Raspberry Pi WH	Mikrokontroller für mein Projekt
GPS-Modul	Geschwindigkeit und Höhe auslesen
Lenkrad Taster	Lap-Time stoppen
Display	Anzeige von Informationen
Magnetometer	Für den Kompass
SD-Karte	Speichern von Log-Daten
Konverter 12V-5V	Spannung herabsetzen
Sicherung 2A	Schutzeinrichtung
Sicherungshalter	Montage der Sicherung
Diode für Crowbar-Schaltung	Schutzeinrichtung
Leiterplatten Kit	Komponenten verbinden
SD-Modul Adafruit	Schreibt und liest die SD-Karte

Abbildung 29:Komponenten

5.8.2 Schema

Nachdem ich nun genau weiss, mit welchen Komponenten ich arbeiten werde, war es für mich wichtig, mir einen detaillierten Überblick darüber zu verschaffen, welche Schnittstellen und Anschlüsse jede einzelne dieser Komponenten besitzt. Diese Information ist entscheidend, um die Bauteile korrekt mit dem Raspberry Pi Pico W zu verbinden, die richtigen Kommunikationsprotokolle zu nutzen und mögliche Konflikte bei der Pinbelegung frühzeitig zu erkennen. Im Folgenden habe ich deshalb alle verwendeten Komponenten zusammen mit ihren jeweiligen Anschlussarten übersichtlich in einer Tabelle aufgeführt.

Was	**Ansteuerung**	**Pins**
GPS-Modul	UART-Bus	TXD, RXD
Display	SPI-Bus	CLK, DIN, CS, RST
Magnetometer	I^2C-Bus	SDA, SCL
SD-Modul Adafruit	SPI-Bus	CLK, DO, DI, CS, CD
Taster	GPIO	Input

Abbildung 30:Ansteuerung

Auf Grundlage der zuvor zusammengestellten Übersicht habe ich ein Grobschema erstellt, das die Belegung der GPIO-Pins am Raspberry Pi Pico W darstellt. Dieses Schema zeigt genau, welche Pins ich für welche Komponenten vorgesehen habe. Es hilft mir dabei, die Verdrahtung strukturiert zu planen und Überschneidungen oder Konflikte bei der Pin-Vergabe frühzeitig zu erkennen.

Für die UART-Kommunikation habe ich die GPIO-Pins 0 und 1 verwendet. Diese sind die Standardpins für UART auf dem Raspberry Pi Pico W und eignen sich ideal für die Ansteuerung des GPS-Moduls. Auch beim I2C-Bus habe ich mich an die Standardbelegung gehalten und die GPIOs 4 und 5 verwendet. Über diese Pins wird das Magnetometer angesteuert. Die Menüführung des Dashboards erfolgt über vier Taster, die ich auf die GPIOs 6 bis 9 gelegt habe. Dabei ist GPIO 6 für die Navigation nach oben zuständig, GPIO 7 für unten, GPIO 8 für links und GPIO 9 für rechts. Für die Lap-Time Funktion, die über einen separaten Taster am Lenker ausgelöst wird, habe ich GPIO 10 vorgesehen. Das SD-Kartenmodul benötigt ebenfalls mehrere GPIOs. GPIO 12 dient dabei zur Erkennung, ob eine Karte eingesteckt ist (Chip Detect), während GPIO 13 als Chip-Select-Pin verwendet wird. Für das OLED-Display habe ich GPIO 14 als Data/Command-Pin genutzt, GPIO 15 für den Reset und GPIO 17 für den Chip-Select. Die restlichen SPI-Signale habe ich auf die standardmässigen Pins gelegt: GPIO 16 übernimmt die Funktion des MISO (Master In, Slave Out), GPIO 18 ist für den Takt (SCK) zuständig, und GPIO 19 übernimmt den MOSI-Kanal (Master Out, Slave In). Durch strukturierte Planung ist sichergestellt, dass alle Komponenten zuverlässig angeschlossen werden können und der Aufbau übersichtlich bleibt.

Nachdem ich das Grobschema für die GPIO-Belegung erstellt hatte, war der nächste Schritt, einen Schaltplan zur Verdrahtung zu zeichnen. In diesem Plan habe ich alle Komponenten übersichtlich eingezeichnet und ihre jeweiligen Verbindungen dargestellt, einschliesslich der Spannungsversorgung, Schalter und Dioden. So erhielt ich bereits vor dem Aufbau einen guten Überblick darüber, wie alles miteinander verbunden wird und ob alle Anschlüsse korrekt belegt wurden.

Im Schaltplan ist ausserdem zu erkennen, dass die Stromversorgung der einzelnen Komponenten direkt vom DC-DC-Wandler erfolgt. Das hat den Hintergrund, dass der Raspberry Pi Pico W selbst nur maximal etwa 300 mA bereitstellen kann. Wenn jedoch alle angeschlossenen Komponenten gleichzeitig mit voller Leistung laufen, liegt der Gesamtstromverbrauch bei über 400 mA. Um zu verhindern, dass der Raspberry Pi Pico W beschädigt wird, versorge ich die Verbraucher daher direkt über den Wandler.

Etwas Besonderes ist die Verdrahtung des Lap-Time Tasters. Dieser ist über GND mit dem Motorradrahmen verbunden, da der Schalter am Lenker montiert ist und über das Gehäuse bereits eine Verbindung zur Fahrzeugmasse besteht. Aus diesem Grund muss ich vom Raspberry Pi Pico W lediglich die 3,3 V vom GPIO 10 zum Schalter führen, wodurch die Verkabelung vereinfacht wird.

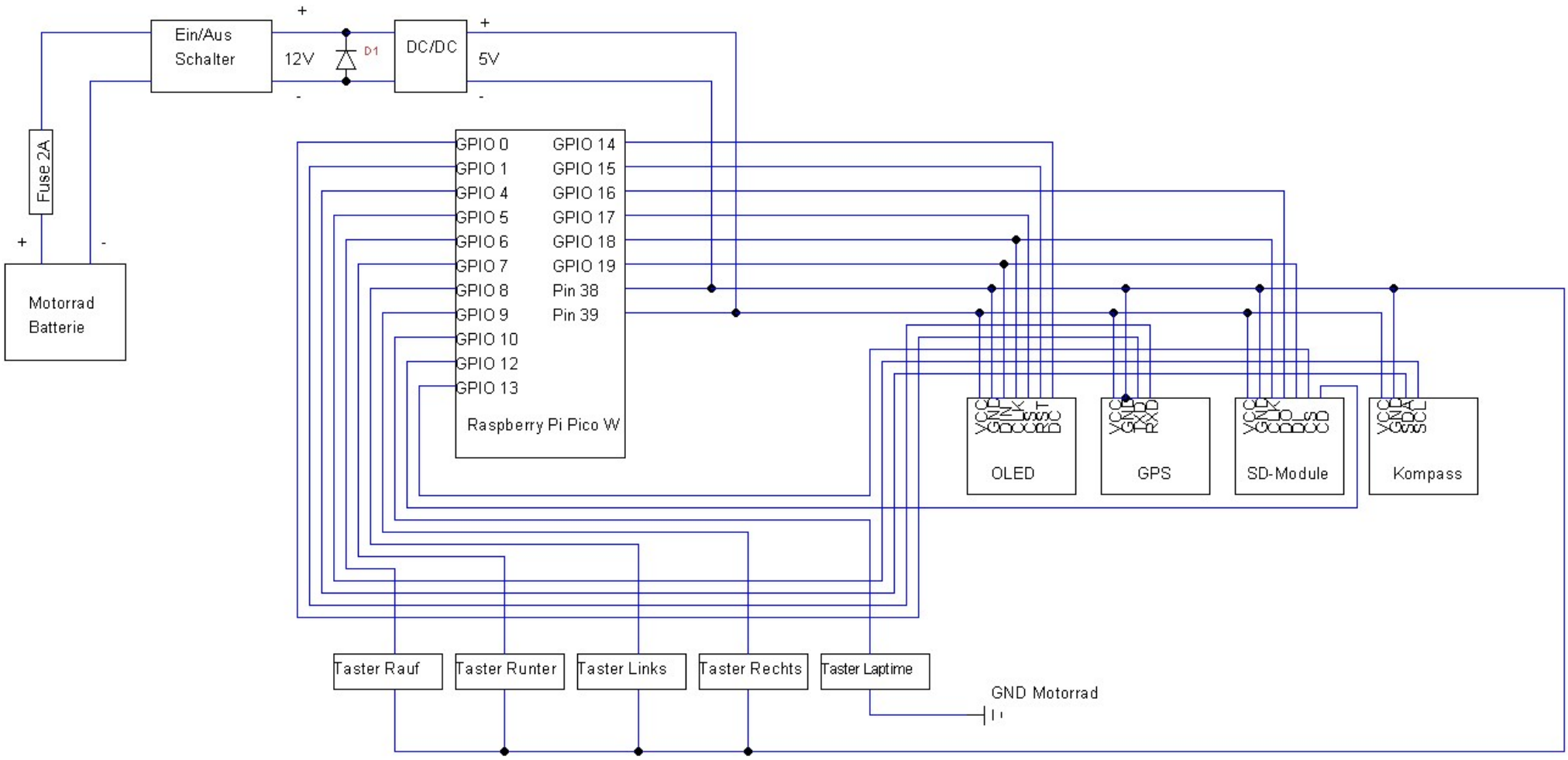


Abbildung 31:Schaltplan

5.9 Programm

In diesem Kapitel möchte ich den Weg vom ersten Entwurf bis hin zum fertigen Programm genauer erklären. Dabei geht es nicht nur darum, wie die einzelnen Teile entwickelt wurden, sondern auch darum, wie sie am Ende zusammenwirken. Schritt für Schritt wird aufgezeigt, welche Überlegungen bei der Umsetzung eine Rolle gespielt haben, wie die Struktur des Programms aufgebaut ist und welche Prinzipien dahinterstecken.

5.9.1 Test Programme

Um sicherzugehen, dass die einzelnen Bauteile für mein Projekt geeignet sind und wie gewünscht funktionieren, habe ich für jede Komponente ein eigenes Testprogramm erstellt. Diese Programme dienen dazu, die grundlegende Funktionsweise zu überprüfen, erste Testergebnisse sichtbar zu machen und die Kommunikation zwischen Mikrocontroller und Bauteil zu testen. So konnte ich Schritt für Schritt feststellen, ob sich die Komponenten zuverlässig einsetzen lassen.

5.9.1.1 Display

Das Programm, das ich geschrieben habe, ist ein Test für ein SPI-Display. Nachdem die Adafruit_GFX-, Adafruit_SSD1327- und SPI- Bibliothek geladen und die Pins für das Display festgelegt werden, startet das Gerät zunächst eine serielle Verbindung mit dem Computer, damit Meldungen ausgegeben werden können. Anschliessend wird das Display initialisiert. Falls es nicht gefunden wird, bleibt das Programm an dieser Stelle stehen. Klappt die Initialisierung, wird der Bildschirm gelöscht und zur Kontrolle das Wort "Start" gross angezeigt. Danach beginnt die Endlosschleife. In der Schleife merkt sich das Programm die aktuelle Zeit und prüft fortlaufend, ob seit der letzten Aktualisierung mindestens eine Sekunde vergangen ist. Wenn das der Fall ist, wird der Bildschirm erneut gelöscht, die aktuelle Ziffer des Zählers angezeigt und derselbe Wert zusätzlich an den Computer geschickt. Der Zähler läuft von 0 bis 9 und springt dann wieder zurück auf 0. Auf diese Weise wird sichtbar, dass das Display korrekt über die SPI-Schnittstelle angesprochen werden kann und regelmässig neue Inhalte darstellt.

```
void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - lastUpdate >= interval) {
    lastUpdate = currentMillis;

    display.clearDisplay();
    display.setCursor(50, 40);
    display.print(counter);
    display.display();

    Serial.print("Zählerstand: ");
    Serial.println(counter);

    counter++;
    if (counter > 9) counter = 0;
  }
```

Abbildung 32: Code Display

5.9.1.2 GPS

In diesem Testprogramm wird zuerst die Bibliothek SoftwareSerial eingebunden. Diese erlaubt es, eine serielle Schnittstelle über selbst definierte Pins zu verwenden. In diesem Fall werden Pin 0 und Pin 1 genutzt, wobei nur der TXD-Pin des GPS-Moduls wirklich gebraucht wird, da hier die Daten nur vom GPS-Modul gesendet werden. Im Setup-Teil startet der Mikrocontroller zunächst die Verbindung zum Computer mit 115200 Baud, wartet kurz, bis diese bereit ist, und meldet sich mit dem Text "GPS Test gestartet". Danach wird die zweite serielle Verbindung zum GPS-Modul mit der Geschwindigkeit von 9600 Baud gestartet. Im Loop-Teil prüft das Programm ständig, ob vom GPS-Modul Daten eintreffen. Wenn ja, wird jedes empfangene Zeichen gelesen und direkt an den Computer weitergeleitet. Auf dem seriellen Monitor erscheinen dadurch die Rohdaten, die das GPS-Modul liefert. So kann man einfach überprüfen, ob das Modul korrekt angeschlossen ist und Daten sendet.

```
void setup() {
  Serial.begin(115200);
  while (!Serial); // Warten, bis USB bereit ist
  Serial.println("GPS Test gestartet...");

  gpsSerial.begin(9600); // GPS-Modul Baudrate (häufig 9600)
}

void loop() {
  if (gpsSerial.available()) {
    char c = gpsSerial.read();
    Serial.write(c); // GPS-Daten an den Monitor ausgeben
  }
}
```

Abbildung 33: Code GPS

5.9.1.3 Kompass

Das Testprogramm, welches ich für den Kompass geschrieben habe, baut eine Verbindung per I²C-Schnittstelle vom Kompass auf den Raspberry Pi Pico W auf. Zuerst werden die nötigen Bibliotheken eingebunden. Diese sind die Wire für die I²C-Kommunikation und die DFRobot_QMC5883 als Treiber für den Kompass. Anschliessend werden die beiden Pins für die I²C-Verbindung festgelegt. SCL_PIN für die Taktleitung und SDA_PIN für die Datenleitung. Mit diesen Einstellungen wird ein Kompass-Objekt erstellt, das mit der Standardadresse 0x0D angesprochen wird. Im Setup-Teil startet das Programm die serielle Verbindung mit dem Computer, damit später Messwerte ausgegeben werden können. Danach werden die I²C-Pins konfiguriert und die Verbindung gestartet. Mit compass.begin() wird geprüft, ob der Sensor erreichbar ist. Falls dies nicht der Fall ist, bleibt das Programm an dieser Stelle stehen. Im Loop-Teil werden die Rohdaten des Magnetfelds eingelesen und an den Computer geschickt. So lässt sich überprüfen, ob der Kompasssensor korrekt angeschlossen ist und zuverlässig Daten liefert.

```
void loop() {
  sVector_t mag = compass.readRaw();

  float X = mag.XAxis;
  float Y = mag.YAxis;
  float Z = mag.ZAxis;

  Serial.print("X: ");
  Serial.print(X);
  Serial.print("Y: ");
  Serial.print(Y);
  Serial.print("Z: ");
  Serial.println(Z);

  delay(500);
}
```

Abbildung 34: Code Kompass

5.9.1.4 SD-Modul

Für dieses Testprogramm habe ich die SPI Bibliothek für die Kommunikation über die serielle Schnittstelle und die SD Bibliothek als Treiber für das Arbeiten mit der Speicherkarte verwendet. Danach werden die Pins festgelegt. SD_CS für die Chip-Select-Leitung, SD_CD als erkennungs Pin für die Karte sowie die SPI-Leitungen SCK, MISO und MOSI. Im Setup-Teil startet der Mikrocontroller eine serielle Verbindung zum Computer, damit Meldungen angezeigt werden können. Anschliessend wird die SD-Karte initialisiert. Solange die Karte nicht gefunden wird, bleibt das Programm an dieser Stelle hängen. Sobald sie erfolgreich gestartet ist, erscheint die Meldung "SD gestartet". Danach folgt ein Block, in dem wahlweise Dateien geschrieben, gelesen oder gelöscht werden können. Zuerst wird die Datei text.txt erstellt. In dieser wird "Hallo Welt von der SD-Karte!" geschrieben. Danach wird die Datei text.txt gelesen und an den Computer geschickt, damit man lesen kann, was auf der SD-Karte geschrieben worden ist. Danach wird die Datei text.txt wieder gelöscht. Im Loop-Teil überprüft das Programm regelmässig den Status des Erkennungspins. Ist der Pegel auf LOW, wird gemeldet, dass keine Karte erkannt wurde. Ansonsten wird angezeigt, dass eine SD-Karte vorhanden ist. Alle 500 Millisekunden wird der Zustand erneut überprüft. Auf diese Weise lässt sich einfach nachvollziehen, ob die SD-Karte korrekt angeschlossen ist, ob Dateien darauf geschrieben, gelesen und gelöscht werden können und ob der Ein- oder Ausbau der Karte erkannt wird.

```
File file = SD.open("/text.txt", FILE_WRITE);
if (file) {
  Serial.println("Schreibe in Datei...");
  file.println("Hallo Welt von der SD-Karte!");
  file.println("Noch eine Zeile.");
  file.close();
  Serial.println("Datei wurde erfolgreich geschrieben.");
} else {
  Serial.println("Konnte Datei nicht schreiben.");
  }
}
{
File file = SD.open("/text.txt");
if (file) {
  Serial.println("Dateiinhalt von text.txt:");
  while (file.available()) {
    String line = file.readStringUntil('\n');
    Serial.println(line);
  }
  file.close();
} else {
  Serial.println("Konnte Datei nicht öffnen.");
  }
}
{
if (SD.exists("/text.txt")) {
  if (SD.remove("/text.txt")) {
    Serial.println("Datei wurde gelöscht.");
  } else {
    Serial.println("Datei konnte nicht gelöscht werden.");
  }
} else {
  Serial.println("Datei existiert nicht.");
  }
}
```

Abbildung 35: Code SD-Modul

5.9.1.5 Hotspot

Das Programm richtet den Raspberry Pi Pico W als WLAN-Hotspot mit integriertem Webserver ein. Zunächst werden die WiFi und WebServer Bibliotheken eingebunden, die die Netzwerkfunktionen und die Verarbeitung von HTTP-Anfragen übernehmen. Anschliessend werden eine SSID und ein Passwort für das WLAN festgelegt, das der Raspberry Pi Pico W später als Access Point zur Verfügung stellt. Im Setup-Teil startet zuerst die serielle Verbindung, um Meldungen am Computer anzeigen zu können. Danach wird mit WiFi.softAP(ssid, password) ein eigener WLAN-Hotspot erzeugt. Der Mikrocontroller erhält dabei automatisch eine IP-Adresse, die über WiFi.softAPIP() ermittelt und im seriellen Monitor ausgegeben wird. Damit weiss der Benutzer, unter welcher Adresse der Pico erreichbar ist. Anschliessend wird ein einfacher HTTP-Handler definiert. Die Funktion handleRoot erzeugt eine kleine HTML-Seite. Darin steht ein Begrüssungs Text sowie die aktuelle IP-Adresse des Pico W. Zum Schluss wird der Webserver gestartet. Im Loop-Teil kümmert sich server.handleClient() darum, alle eingehenden Anfragen von verbundenen Geräten zu bearbeiten. Ruft man mit einem Smartphone oder Computer die IP-Adresse des Raspberry Pi Pico W im Browser auf, erscheint die vorbereitete Webseite. Damit lässt sich leicht testen, ob der Pico W als WLAN-Hotspot funktioniert und ob der eingebaute Webserver erreichbar ist.

```
// Root-Seite
void handleRoot() {
  String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'><title>Pico W</title></head><body>";
  html += "<h1>Hallo vom Pico W!</h1>";
  html += "<p>Du bist mit dem Hotspot verbunden.</p>";
  html += "<p>IP-Adresse: ";
  html += WiFi.softAPIP().toString();
  html += "</p></body></html>";

  server.send(200, "text/html", html);
}
```

Abbildung 36: Code Hotspot

5.9.2 Systemarchitektur

Um klar zu definieren, was genau programmiert werden soll, habe ich das Programm mithilfe einer State Machine visualisiert. Diese Darstellung zeigt, welche Anzeige in welchem Zustand aktiv ist und welche Funktion die einzelnen Taster jeweils auslösen. Die Visualisierung dient mir während der Programmierung als Übersicht und Kontrolle, um sicherzustellen, dass alle Funktionen umgesetzt sind und der Code wie vorgesehen arbeitet.

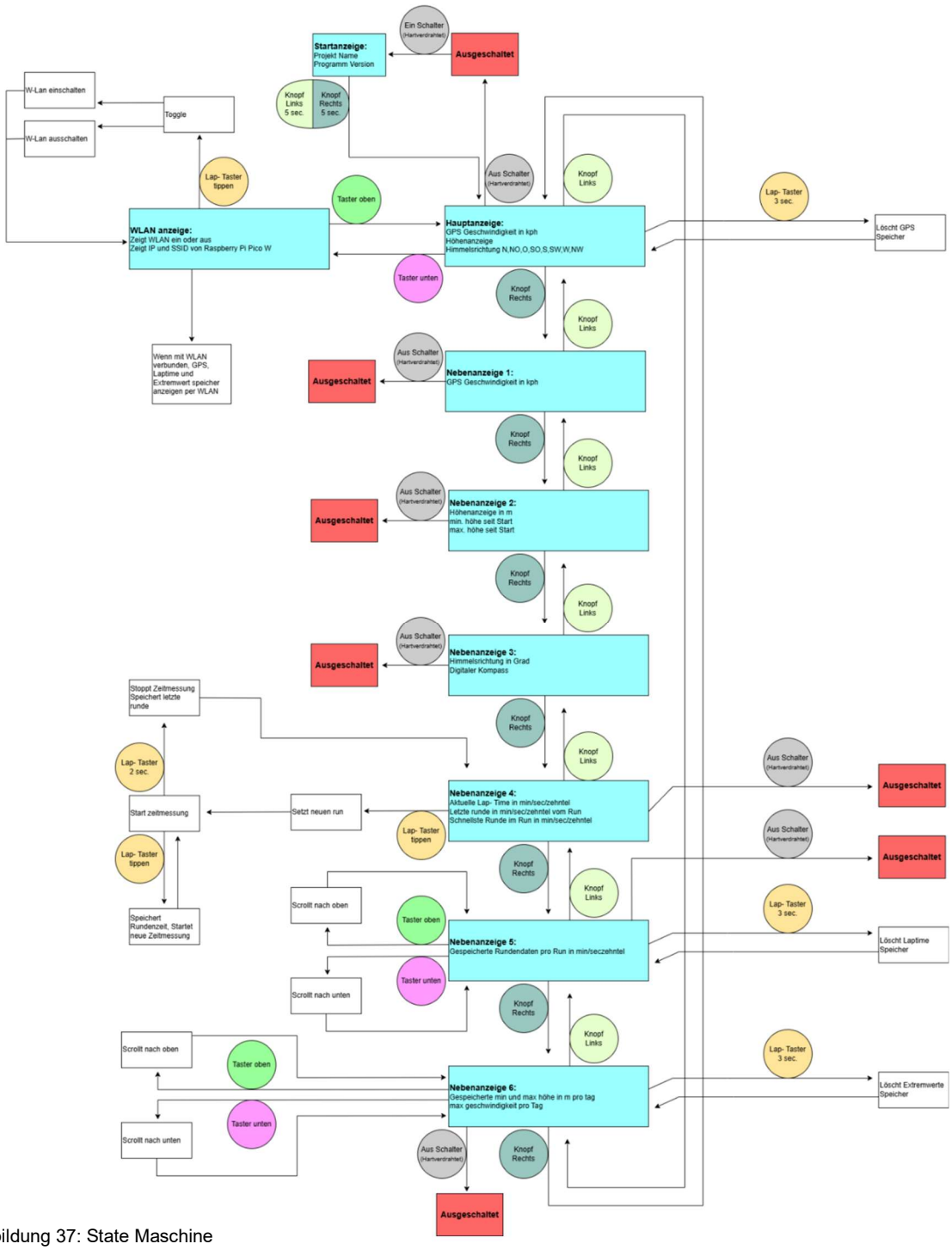


Abbildung 37: State Maschine

5.9.3 Haupt Programm

Das Programm für das Motorrad Dashboard wurde in der Arduino ide Umgebung realisiert und ist in C++ geschrieben. Es verarbeitet und zeigt Mess- und GPS-Daten. Ziel des Systems ist die Aufzeichnung von Rundenzeiten, Extremwerten, wie die minimum und maximum Höhe, der Geschwindigkeit, GPS-Tracking sowie die Anzeige der Informationen auf einem OLED-Display. Die Implementierung gliedert sich in zwei Abschnitte: die Initialisierung im Setup und die kontinuierliche Verarbeitung in der Hauptschleife.

5.9.3.1 Programmfunktionen

Im Setup werden zunächst die Hardwarekomponenten konfiguriert. Hierzu zählen die Taster, die als digitale Eingänge definiert werden, um die Navigation durch den Benutzer zu ermöglichen. Dazu erfolgt die Initialisierung der SD-Karte, auf der Messdaten wie Lap-Time, Extremwerte oder Trackinginformationen gespeichert werden. Das OLED-Display wird ebenfalls initialisiert, wobei Textgrösse, Kontrast und Farbe festgelegt werden, um eine gut lesbare Darstellung von Menüs sicherzustellen. Für die GPS-Funktionalität wird die serielle Schnittstelle initialisiert, wodurch eine Datenübertragung von GPS-Modul zu Mikrocontroller gewährleistet wird. Zusätzlich werden Variablen zur Erstellung von Messungen, Laufnummern und Rundenzeiten vorbereitet, sodass das System direkt nach dem Start betriebsbereit ist.

Die Hauptschleife steuert alle dynamischen Abläufe des Systems. Zunächst werden die Eingaben der Taster abgefragt und verarbeitet. Die Tasten rechts und links ermöglichen die Navigation durch verschiedene Menüansichten, wobei eine zyklische Durchlaufsteuerung implementiert ist. Das WLAN-Menü ist dabei von der seitlichen Navigation ausgenommen, um unbeabsichtigtes Verlassen zu verhindern. Die Tasten runter und rauf dienen dem Scrollen in Listenansichten, wie z. B. Lap-Time - oder Extremwerttabellen, die blockweise angezeigt werden.

Ein zentrales Feature ist die Lap-Time Funktion. Ein kurzer Tastendruck startet eine Messung. Ein weiterer kurzer Tastendruck erfasst eine neue Runde. Währenddessen wird die beste Runde automatisch aktualisiert und angezeigt. Ein langer Tastendruck von mehr als zwei Sekunden beendet die Messung, speichert die gesammelten Daten auf der SD-Karte und aktualisiert den Run-Counter. Dieser speichert die Anzahl der Turns, die gemessen wurden. Zusätzlich erlaubt ein gedrückt halten des Tasters von drei Sekunden das gezielte Löschen von Lap-Time, Extremwerte oder Trackingdateien, abhängig vom gerade aktiven Menüpunkt.

Parallel hierzu werden GPS-Daten kontinuierlich eingelesen und verarbeitet. Die Daten werden über die serielle Schnittstelle ausgelesen und an die TinyGPSPlus Bibliothek übergeben. Dabei werden Geschwindigkeit, Höhe, Richtung und Datum extrahiert. Ein Puffersystem sammelt GPS-Koordinaten, die einmal pro Minute auf der SD-Karte

gespeichert werden, um eine kontinuierliche Trackingaufzeichnung zu gewährleisten. Die Richtung wird über den GPS-Kompass ermittelt, sofern eine Mindestgeschwindigkeit von 3 km/h erreicht wird, andernfalls zeigt das System standardmässig nach Norden. Eine Glättungsfunktion sorgt für stabile Messwerte, insbesondere bei Höhe und Geschwindigkeit.

Das System speichert zudem Extremwerte, indem es tägliche minimum und maximum für Höhe sowie die maximale Geschwindigkeit prüft. Werden neue Extremwerte erreicht, werden diese automatisch in entsprechenden Dateien gespeichert, sodass eine lückenlose Aufzeichnung der Daten möglich ist. Diese werden jeweils am Ende eines Tages automatisch zusammengefügt, damit man diese einsehen kann.

Die Darstellung auf dem OLED-Display erfolgt menüabhängig. Das Hauptmenü zeigt Geschwindigkeit, Höhe und Richtung an, während die Nebenmenüs detaillierte Informationen liefern, wie einzelne Geschwindigkeit oder Höhenwerte, einen grafischen Kompass, Lap-Time einschliesslich aktueller, letzter und bester Runde sowie den Status von WLAN und Netzwerkinformationen (IP-Adresse und SSID). Bei aktiviertem WLAN werden eingehende HTTP-Anfragen von einem eingebetteten Webserver verarbeitet, sodass die gespeicherten Daten zusätzlich über ein Netzwerk abgerufen werden können.

5.9.3.2 Code Realisierung

Bei der Programmierung ging ich nach der State Machine vor und kreuzte die Funktionen, die ich realisiert hab ab, um den Überblick zu behalten. Mein Menü besteht aus Case 0 bis Case 7. Der Case 0 ist meine Hauptanzeige. Case 1- 6 sind die Nebenanzeigen. Auf diesen werden verschiedene Fahrdaten angezeigt. Case 7 ist für die WLAN-Aktivierung und Deaktivierung sowie für die Anzeige der IP-Adresse, damit man sich mit dem Raspberry Pi Pico verbinden kann. Alle Case-Blöcke sind in einem switch-Block zusammengefasst. Abhängig vom Wert der Variablen menuIndex, wird der jeweils passende Case ausgeführt. Dadurch muss nicht jede Bedingung wie bei mehreren if-Abfragen geprüft werden. Durch dieses Vorgehen wird der Loop-Teil schneller.

5.9.3.2.1 Header

Zu Beginn des Programmcodes befindet sich ein Header, der eine übersichtliche Zusammenfassung der wichtigsten Informationen zum Projekt enthält. Der Header beschreibt zunächst den Zweck des Programms, nämlich die Erfassung, Auswertung und Darstellung von Fahrdaten eines Motorrads.

Anschliessend sind im Header die Kontaktdaten von mir, die Bildungseinrichtung sowie die verwendeten Bibliotheken und Quellen aufgeführt. Diese Übersicht dokumentiert, auf welchen Softwarekomponenten das Projekt basiert. Der Header weist ausserdem darauf hin, dass die Bibliotheken unter einer MIT- oder BSD-Lizenz stehen und deren Lizenzbedingungen bei einer Weiterverwendung beachtet werden müssen.

Darüber hinaus enthält der Header einen Abschnitt mit einem Hinweis zur Weiterverwendung des Programms. Dieser fordert dazu auf, den Urhebervermerk beizubehalten und eigene Anpassungen oder Erweiterungen nachvollziehbar zu dokumentieren. Dadurch wird eine transparente Weiterentwicklung des Projekts ermöglicht.

Ein weiterer wichtiger Bestandteil ist der Versionsverlauf, in dem alle Änderungen chronologisch festgehalten sind. Jede Version ist mit einer Versionsnummer, einem Datum und einer kurzen Beschreibung der durchgeführten Anpassung versehen. Dieser Abschnitt zeigt die fortlaufende Entwicklung des Programms, von der ersten Grundstruktur über die Integration der Sensoren, die Erweiterung der Display- und WLAN-Funktionen bis hin zur Implementierung von Mess-, Speicher- und Webschnittstellen.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Motorrad_Dashboard
//====================
//
// Beschreibung Programm
//======================
// Das Programm erfasst und zeigt wichtige Fahrdaten eines Motorrads an. Es misst die GPS-Geschwindigkeit,
// bestimmt die Kompassrichtung, ermittelt die Höhe und ermöglicht die Messung von Laptimes.
// Alle Informationen werden übersichtlich auf einem Display dargestellt.
// Die gespeicherten Werte sind über einen integrierten WLAN-Hotspot auch per Webinterface abrufbar.
//
// Jonas.kuj@bluewin.ch
// Jonas.Kuster@edu.teko.ch
// TEKO Luzern
//
```

Abbildung 38: Abschnitt Header

5.9.3.2.2 Bibliotheken

Unter dem Header werden die benötigten Bibliotheken eingebunden, welche die Grundlage für die verschiedenen Hardware- und Softwarefunktionen des Projekts bilden. Diese Bibliotheken erweitern die Standardfunktionen der Arduino Umgebung und ermöglichen die Ansteuerung sowie Kommunikation der einzelnen Komponenten.

Die Board-Library für den Raspberry Pi Pico W stammt aus dem offiziellen GitHub-Repository von Earle Philhower. Sie stellt die grundlegende Unterstützung für den Mikrocontroller bereit und beinhaltet alle notwendigen Funktionen für die Nutzung von Ein- und Ausgängen, der seriellen Schnittstellen sowie der integrierten WLAN-Funktion.

Die Adafruit GFX Library sowie die Adafruit SSD1327 Library ermöglichen die grafische Darstellung von Informationen auf dem verwendeten OLED-Display. Während die GFX-Bibliothek allgemeine Zeichenfunktionen wie Text-, Linien- und Rechteckdarstellung bereitstellt, kümmert sich die SSD1327-Library um die hardwareseitige Kommunikation mit dem Display über die SPI-Schnittstelle.

Die Bibliotheken SPI.h und Wire.h stammen aus der Board-Library und dienen der Kommunikation über die SPI- und I²C-Schnittstellen. Damit können Sensoren, Displays und Speichermodule effizient mit dem Mikrocontroller verbunden werden. Die Bibliothek SoftwareSerial.h ermöglicht zudem die Verwendung zusätzlicher serieller Schnittstellen über frei wählbare Pins, was insbesondere für den GPS-Empfänger von Vorteil ist.

Für die Erfassung der Himmelsrichtung wird die DFRobot_QMC5883-Library eingesetzt. Sie stellt die Funktionen zur Initialisierung und Auswertung des Magnetometers bereit und erlaubt so die Berechnung der Kompassrichtung in Grad.

Die Bibliotheken WiFi.h und WebServer.h ermöglichen die Einrichtung eines WLAN-Hotspots sowie die Realisierung eines einfachen Webservers auf dem Pico W. Über diesen können gespeicherte Messdaten wie Rundenzeiten, Extremwerte oder Trackinginformationen im Browser angezeigt werden.

Abschliessend sorgt die SD.h-Bibliothek für die Kommunikation mit der SD-Karte, auf der alle Messwerte und Logdateien gespeichert werden. Die TinyGPSPlus-Library von Mikal Hart verarbeitet die über die serielle Schnittstelle empfangenen GPS-Daten und stellt daraus Geschwindigkeit, Position, Höhe und Zeitinformationen bereit.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Bibliotheken
//Board Library von: https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json
#include <Adafruit_GFX.h>         //Adafruit GFX Library by Adafruit + Adafruit BusI0 by Adafruit
#include <Adafruit_SSD1327.h>     //Adafruit SSD 1327 by Adafruit + Adafruit BusI0 by Adafruit
#include <SPI.h>                  //von Board Library
#include <SoftwareSerial.h>       //von Board Library
#include <DFRobot_QMC5883.h>      //DFRobot_QMC5883 by DFRobot
#include <Wire.h>                 //von Board Library
#include <WiFi.h>                 //von Board Library
#include <WebServer.h>            //von Board Library
#include <SD.h>                   //von Board Library
#include <TinyGPSPlus.h>          //TinyGPSPlus by Mikal Hart
```

Abbildung 39: Abschnitt Bibliotheken

5.9.3.2.3 Variablen, Objekte, Pins

Vor dem Setup werden die Hardware-Pins, globalen Variablen und Objekte definiert, die für den Betrieb des Motorrad Dashboards notwendig sind. Diese Definitionen bilden die Schnittstelle zwischen Mikrocontroller, Sensoren, Display, SD-Karte und Bedienelementen und sorgen so für eine strukturierte und nachvollziehbare Hardwareansteuerung im gesamten Programm.

Zunächst werden die Display-Pins festgelegt. Das verwendete OLED-Display besitzt eine Auflösung von 128x128 Pixeln und wird über die SPI-Schnittstelle angesteuert.

```
////////////////////////////////////////
//Display-Pins
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 128
#define DISPLAY_CS    17 //Chip Select
#define DISPLAY_DC    14 //Data/Command
#define DISPLAY_RESET 15
```

Abbildung 40: Display Pins

Die Pins DISPLAY_CS, DISPLAY_DC und DISPLAY_RESET definieren dabei die Steuerleitungen für Chip Select, Daten und Befehlsumschaltung sowie den Reset-Eingang. Diese Konfiguration ermöglicht eine direkte und performante Kommunikation mit dem Displaytreiber.

Anschliessend folgen die SPI-Pins, über die neben dem Display auch andere Geräte wie die SD-Karte angebunden werden. Die Pins SPI_MISO, SPI_MOSI und SPI_SCK übernehmen die klassischen SPI-Funktionen für den Datenaustausch zwischen Mikrocontroller und den angeschlossenen Komponenten.

Die SD-Karte wird über die Pins SD_CD (Card Detect) und SD_CS (Chip Select) angebunden. Damit kann das Programm sowohl erkennen, ob eine Speicherkarte eingelegt ist, als auch gezielt den Zugriff über die SPI-Schnittstelle steuern.

```
////////////////////////////////////
//SPI-Pins
#define SPI_MISO 16 //(DO/MISO)
#define SPI_MOSI 19 //(DI/DIN/MOSI)
#define SPI_SCK  18 //(CLK/SCK)
```

Abbildung 41: SPI-Pins

Für den I²C-Bus sind die Pins SCL_PIN und SDA_PIN definiert, welche die Verbindung zum Magnetometer herstellen. Über diesen Bus können mehrere Geräte mit nur zwei Leitungen parallel betrieben werden.

```
////////////////////////////////////
//I2C-Pins
#define SCL_PIN 5 //I2C SCL auf GPIO 5
#define SDA_PIN 4 //I2C SDA auf GPIO 4
```

Abbildung 42: I²C Pins

Die Taster Eingaben werden über fünf GPIO-Pins realisiert. Diese Knöpfe (KNOPF_RAUF, KNOPF_RUNTER, KNOPF_LINKS, KNOPF_RECHTS, KNOPF_LAPTIME) dienen zur Navigation im Menü, zur Bedienung der Lap-Time Funktion sowie zum Wechseln zwischen verschiedenen Anzeigemodi. Um Fehlbetätigungen zu vermeiden, werden für jeden Knopf Variablen gespeichert, die den letzten Schaltzustand erfassen. Damit lassen sich die Flankensteuerungen und die Entprellung softwareseitig umsetzen.

```
////////////////////////////////////////////
//Knöpfe
#define KNOPF_RAUF 6
#define KNOPF_RUNTER 7
#define KNOPF_LINKS 8
#define KNOPF_RECHTS 9
#define KNOPF_LAPTIME 10
int menuIndex = 0;          //  Menüpunkt (0..6)
const int maxMenu = 6;      // höchste Zahl
bool lastKNOPF_RAUF = HIGH;
bool lastKNOPF_RUNTER = HIGH;
bool lastKNOPF_LINKS = HIGH;
bool lastKNOPF_RECHTS = HIGH;
bool lastKNOPF_LAPTIME = HIGH;
```

Abbildung 43: Knöpfe Pins

Darauf folgt die Initialisierung des Display-Objekts der Klasse Adafruit_SSD1327. Dieses Objekt ermöglicht die grafische Ausgabe über die zuvor definierten Pins und stellt Funktionen zum Zeichnen, Schreiben und Aktualisieren der Anzeige bereit.

```
////////////////////////////////////////////////////////////////////////////////////////////////
// Display-Objekt SPI0
Adafruit_SSD1327 display(SCREEN_WIDTH, SCREEN_HEIGHT, &SPI, DISPLAY_DC, DISPLAY_RESET, DISPLAY_CS);
```

Abbildung 44: Display Objekt

Für die GPS-Datenverarbeitung wird ein serielles Objekt über SoftwareSerial eingerichtet, das mit dem GPS-Modul kommuniziert. Über die Bibliothek TinyGPSPlus werden daraus Werte wie Geschwindigkeit, Höhe und Datum ermittelt. Verschiedene Zeitvariablen regeln die Aktualisierungsintervalle, um eine gleichmässige Datenanzeige sicherzustellen.

```
////////////////////////////////////////////////////////////////
//GPS-Objekt
SoftwareSerial gpsSerial(0, 1); // GPIO 0= TXD, GPIO 1= RXD Pins
char lastGpsChar = ' ';
TinyGPSPlus gps;
int gpsspeed = 0;
int gpsaltitude = 0;
int gpsdate = 0;
unsigned long lastDateUpdate = 0;
unsigned long lastSpeedUpdate = 0;
unsigned long lastAltitudeUpdate = 0;
unsigned long lastGpsUpdate = 0;
unsigned long lastSaveTime = 0;
const unsigned long GPS_INTERVAL = 1000;      // 1 Sekunde
const unsigned long SAVE_INTERVAL = 60000;   // 1 Minuten
String gpsBuffer = "";
unsigned long messageStartTime = 0;
bool messageActive = false;
String messageLine1 = "";
String messageLine2 = "";
File trackFile;
```

Abbildung 45: GPS-Objekt

Darüber hinaus werden mehrere Dateiobjekte definiert, die für das Schreiben und Lesen von Messdaten auf der SD-Karte zuständig sind. Ebenso existieren Variablen zur Verwaltung der Rundenzeiten, zur Speicherung der höchsten und niedrigsten Höhen sowie zur Erfassung der maximalen Geschwindigkeit eines Tages.

5.9.3.2.4 Setup

Die Setup-Funktion bildet den grundlegenden Initialisierungsteil des Motorrad Dashboard Programms und wird beim Start des Mikrocontrollers einmalig ausgeführt. In dieser Funktion werden alle benötigten Hardware-Komponenten, Kommunikationsschnittstellen sowie wichtige Startparameter vorbereitet, damit das System im späteren Betrieb fehlerfrei funktioniert.

Zunächst wird über Serial.begin(115200) die serielle Schnittstelle gestartet, um über den USB-Anschluss Debug-Ausgaben an den Computer zu senden. Dies ermöglicht eine einfache Fehleranalyse und dient der Kontrolle, ob alle Module korrekt initialisiert wurden.

Im Anschluss wird die I²C-Schnittstelle konfiguriert, über die Sensoren wie der Magnetometer (Kompass) kommunizieren. Dazu werden die definierten Pins SCL_PIN und SDA_PIN gesetzt und der Bus mit Wire.begin() aktiviert. Dadurch ist der Mikrocontroller bereit, Daten mit I²C-Geräten auszutauschen.

Danach folgt die Initialisierung des OLED-Displays. Mit display.begin() wird geprüft, ob das Display korrekt erkannt wird. Sollte dies fehlschlagen, bleibt das Programm in einer Endlosschleife stehen und gibt über die serielle Konsole eine Fehlermeldung aus. Bei erfolgreicher Erkennung wird das Display gelöscht, die Textgrösse auf 1 gesetzt, die Schriftfarbe auf weiss konfiguriert und der Bildschirm aktiviert.

```
//Display
if (!display.begin()) {
  Serial.println("Display nicht erkannt!");
  while (true) {
      delay(1000); // Endlosschleife, verhindert Weiterlaufen
  }
```

Abbildung 46: Display begin

Anschliessend wird das GPS-Modul über eine serielle Schnittstelle gestartet (gpsSerial.begin(9600)). Damit beginnt die Kommunikation mit dem GPS-Empfänger, der später Positionsdaten, Geschwindigkeit, Höhe und Zeitinformationen liefert. Eine kurze Rückmeldung in der seriellen Konsole bestätigt den erfolgreichen Start des Moduls.

Der nächste Abschnitt betrifft die SD-Karte, die als permanenter Datenspeicher dient. Nach dem Befehl SD.begin(SD_CS) wird geprüft, ob die Karte vorhanden und lesbar ist. Wenn keine SD-Karte erkannt wird, wird das Programm gestoppt, um Datenverlust zu verhindern.

Im Anschluss werden mehrere gespeicherte Werte aus Dateien auf der SD-Karte geladen, um die gespeicherten Messdaten vom letzten Programmstart wiederherzustellen. Aus der Datei RunCounter.txt wird die Anzahl der bisherigen Turns eingelesen. Falls diese Datei noch nicht existiert, wird der Zähler auf null gesetzt. Die Dateien maxSpeedTag.txt, maxHoeheTag.txt, minHoeheTag.txt und DateTag.txt enthalten die Extremwerte und das Datum des letzten Nutzungstags. Diese werden beim Start des Systems in die entsprechenden Variablen geladen, damit aktuelle Messungen mit Vorherigen verglichen werden können.

```
// DateTag laden
DateTagFile = SD.open("/DateTag.txt", FILE_READ);
if (DateTagFile) {
    String lastLine;
    while (DateTagFile.available()) {
        lastLine = DateTagFile.readStringUntil('\n');
    }
    DateTag = lastLine;
    delay(10);
    DateTagFile.close();
} else {
    DateTag =""; // erste Benutzung
}
```

Abbildung 47: DateTag laden

Daraufhin werden alle Bedientasten als Eingänge mit internem Pull-up-Widerstand konfiguriert. Dadurch sind die Tasten standardmässig auf HIGH und schalten auf LOW, wenn sie gedrückt werden. Diese Tasten dienen zur Menüsteuerung, Navigation und Auslösung der Lap-Time-Funktion.

```
//Knöpfe
pinMode(KNOPF_RAUF,INPUT_PULLUP);
pinMode(KNOPF_RUNTER,INPUT_PULLUP);
pinMode(KNOPF_LINKS,INPUT_PULLUP);
pinMode(KNOPF_RECHTS,INPUT_PULLUP);
pinMode(KNOPF_LAPTIME, INPUT_PULLUP);
```

Abbildung 48: Knöpfe Konfiguration

Zum Abschluss wird ein kurzer Begrüssungsbildschirm auf dem Display angezeigt, auf dem der Schriftzug „Motorrad Dashboard“ sowie die Programmversion (versionX) erscheinen. Diese Startanzeige bleibt, bis durch gemeinsames Drücken der linken und rechten Taste das Programm gestartet wird.

5.9.3.2.5 Case 0

```
case 0: { // Hauptanzeige
    display.clearDisplay();
    if (!messageActive) {
      display.setTextSize(2);
      display.setCursor(10, 20);
      if (gpsspeed == 1) {
          display.print(gps.speed.kmph(), 0);
          display.println(" kph");
      }else{
          display.println("---- kph");
      }
      display.setCursor(10, 56);
      if (gpsaltitude == 1) {
          display.print(gps.altitude.meters(), 0);
          display.println(" m");
      }else{
          display.println("---- m");
      }
      display.setCursor(10, 92);
      display.print("Rtg: ");
      display.println(dir);
    }
```

Abbildung 49: Case 0

Der Menüpunkt Case 0 dient als Hauptanzeige des Systems und liefert dem Benutzer eine übersichtliche Darstellung der aktuellen Informationen. In dieser Ansicht wird die aktuelle Geschwindigkeit, die Höhe und die Richtung angezeigt, sofern gültige GPS-Daten vorliegen. Die Geschwindigkeit wird in Kilometern pro Stunde dargestellt und nur dann angezeigt, wenn das GPS ein gültiges Geschwindigkeitssignal liefert; ansonsten wird ein Platzhalter in Form von "---- kph" angezeigt, um anzuzeigen, dass keine Daten verfügbar sind. Parallel dazu erfolgt die Darstellung der Höhe. Dabei werden die GPS-Höhenwerte zunächst geglättet, um kurzfristige Schwankungen zu reduzieren, und anschliessend auf dem Display ausgegeben. Falls keine gültigen Höhenwerte vorliegen, wird ebenfalls ein Platzhalter "---- m" angezeigt.

Die Richtung wird anhand des aktuellen GPS-Kurses berechnet und anschliessend geglättet, um ungenaue oder schwankende Messwerte zu minimieren. Über die Funktion zur Richtungsbestimmung wird die Gradzahl in eine Himmelsrichtung umgewandelt, beispielsweise Nord, Ost, Süd oder West. Liegen keine ausreichenden GPS-Daten oder keine ausreichende Geschwindigkeit vor, zeigt das System stattdessen einen Platzhalterwerte in Form von "Rtg. ----" an.

Zusätzlich unterstützt Case 0 die Anzeige der Nachricht, dass die Tracking Dateien gelöscht wurden. Diese Nachricht wird für einen festen Zeitraum von zwei Sekunden eingeblendet und anschliessend automatisch ausgeblendet. Steuerparameter wie messageActive und messageStartTime überwachen dabei die Dauer und den Status der Nachrichtenanzeige.

Für die Darstellung werden verschiedene Variablen verwendet, darunter Flags, die die Verfügbarkeit von GPS-Daten anzeigen (gpsspeed, gpsaltitude, gpsdate), die aktuelle Ausrichtung (heading), die ermittelte Himmelsrichtung (dir) sowie die Textinhalte der Nachrichten (messageLine1, messageLine2). Die Anzeige selbst wird über die Displayfunktionen gesteuert, die die Textgrösse, Position und Aktualisierung regeln. Durch die kontinuierliche Aktualisierung bei jedem Durchlauf der Hauptschleife wird sichergestellt, dass die angezeigten Werte immer den aktuellen Stand der Messungen widerspiegeln.

5.9.3.2.6 Case 1

Der Menüpunkt Case 1 dient als Nebenanzeige für die Geschwindigkeit. In dieser Ansicht wird ausschliesslich die aktuelle Geschwindigkeit in Kilometern pro Stunde dargestellt. Die Geschwindigkeit wird nur dann ausgegeben, wenn das GPS gültige Daten liefert, die über die Variable gpsspeed geprüft werden. Liegt kein gültiges Signal vor, erscheint ein Platzhalter "---- kph", um dem Benutzer zu signalisieren, dass keine verlässlichen Messwerte verfügbar sind.

```
case 1: {//Nebenanzeige 1
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(10, 56);
    if (gpsspeed == 1) {
      display.print(gps.speed.kmph(), 0);
      display.println(" kph");
    }else{
      display.setCursor(10, 56);
      display.println("---- kph");
    }
    break;
}
```

Abbildung 50: Case 1

Die Anzeige erfolgt mittig auf dem Display, wobei die Textgrösse auf 2 gesetzt wird, um die Werte gut lesbar zu präsentieren. Die Positionierung und Darstellung werden über die Displayfunktionen gesteuert, wobei setCursor die Koordinaten auf dem Display definiert. Die Aktualisierung der Anzeige erfolgt bei jedem Durchlauf der Hauptschleife, sodass die Werte in Echtzeit dargestellt werden.

5.9.3.2.7 Case 2

Case 2 stellt die Höhenanzeige des Geräts dar. In diesem Menü wird die aktuelle Höhe zusammen mit den minimalen und maximalen Höhen angezeigt, die über das GPS ermittelt werden. Zunächst wird die aktuelle Höhe über die Funktion smoothAltitude() geglättet, um Schwankungen in den Messwerten auszugleichen und eine stabilere Anzeige zu erhalten. Die geglättete Höhe (avgAltitude) wird in der oberen Zeile des Displays ausgegeben, während die minimal gemessene Höhe (minAltitude) und die maximal gemessene Höhe (maxAltitude) in den folgenden Zeilen dargestellt werden. Ob die Höheninformationen angezeigt werden können, wird über die Variable gpsaltitude geprüft; ist diese nicht gesetzt, erscheint ein Platzhalterwert "---- m", um anzuzeigen, dass aktuell keine gültigen GPS-Höhenwerte vorliegen. Die Anzeige verwendet die Textgrösse 2 für eine gute Lesbarkeit. Case 2 ermöglicht so eine schnelle Übersicht über die Höhenentwicklung während der Aktivität und verknüpft die GPS-Datenverarbeitung mit der Visualisierung auf dem OLED-Display.

```
case 2: {//Nebenanzeige 2
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(10, 48);
    if (gpsaltitude == 1) {
      // 1. Zeile: Durchschnittshöhe
      display.setCursor(10, 20);
      display.print(avgAltitude, 0);
      display.println(" m");

      // 2. Zeile: Min
      display.setCursor(10, 56);
      display.print("-: ");
      display.print(minAltitude, 0);
      display.println(" m");
```

Abbildung 51: Case 2

5.9.3.2.8 Case 3

In diesem Menü wird die aktuelle Fahrtrichtung dargestellt, basierend auf dem GPS-Kurs. Zunächst wird überprüft, ob ein GPS-Fix vorhanden ist und das Gerät sich mit einer Geschwindigkeit von mehr als 2 km/h bewegt. Falls ja, wird der GPS-Kurs (gps.course.deg()) genutzt, geglättet über die Funktion glaetungMesswert(), um abrupte Richtungsänderungen zu reduzieren. Falls die Bedingungen nicht erfüllt sind, zeigt der Kompass standardmässig nach Norden.

Auf dem Display ist ein Kreis gezeichnet, der als Kompassrahmen dient. Die vier Haupt-Himmelsrichtungen (N, O, S, W) werden über drawLine() und display.print() markiert, wobei die Position der Buchstaben relativ zum aktuellen Heading rotiert werden. Zusätzlich wird ein Zeiger von der Kreismitte nach oben gezeichnet, der die aktuelle Richtung visualisiert. Unterhalb des Zeigers wird die aktuelle Gradzahl angezeigt, ebenfalls abhängig von der Verfügbarkeit eines GPS-Fixes. Wenn der Fix nicht vorhanden ist, wird ein Platzhalter mit "----" angezeigt.

```
for (int i = 0; i < 4; i++) {
    float rad = (winkelH[i] - heading) * DEG_TO_RAD; // Drehung gegen heading
    int xMark = CENTER_X + cos(rad - HALF_PI) * (RADIUS - 5);
    int yMark = CENTER_Y + sin(rad - HALF_PI) * (RADIUS - 5);
    int xOuter = CENTER_X + cos(rad - HALF_PI) * RADIUS;
    int yOuter = CENTER_Y + sin(rad - HALF_PI) * RADIUS;
    display.drawLine(xMark, yMark, xOuter, yOuter, SSD1327_WHITE);
```

Abbildung 52: Case 3 Kreis

5.9.3.2.9 Case 4

Case 4 ist für die Lap-Time Anzeige zuständig und zeigt aktuelle, letzte sowie beste Rundenzeiten während einer Messung an. Dieses Menü ist eng mit der Messlogik des Geräts verknüpft, die beim Drücken des Lap-Time Knopf gestartet oder gestoppt wird. Zunächst wird überprüft, ob eine Messung aktiv ist (timingActive). Ist dies der Fall, wird die aktuelle Rundenzeit berechnet, indem die Differenz zwischen der aktuellen Zeit (millis()) und dem Startzeitpunkt der Runde (lapStart) genommen wird. Die Zeit wird in Minuten, Sekunden und Hundertstel zerlegt, damit sie auf dem Display gut lesbar formatiert werden kann.

```
// Beste Runde
display.setCursor(10, 55);
display.print("Beste: ");
if (valuesAvailable) {
  unsigned int minutes = bestLap / 60000;
  unsigned int seconds = (bestLap % 60000) / 1000;
  unsigned int hundredths = (bestLap % 1000) / 10;
  display.print(minutes);
  display.print(":");
  if (seconds < 10) display.print("0");
  display.print(seconds);
  display.print(".");
  if (hundredths < 10) display.print("0");
  display.println(hundredths);
} else {
    display.println("----");
}
```

Abbildung 53: Case 4 beste Runde

Danach wird die letzte Runde angezeigt, die beim Loslassen des Lap-Time Buttons gespeichert wurde. Über ein Array lapTimes[] werden die einzelnen Rundenzeiten gesammelt, wobei die Funktion formatLapTime() genutzt wird, um die Zeit in einem konsistenten Format darzustellen. Zusätzlich wird die beste Runde ermittelt, indem die aktuelle Runde mit, der bisher besten verglichen wird (bestLap). Ist die Zeit kürzer als die in bestLap gespeicherte Zeit wird diese überschrieben. Schliesslich wird die aktuelle Rundennummer angezeigt, die während des Messvorgangs hochgezählt wird.

5.9.3.2.10 Case 5

Case 5 dient der Anzeige und Verwaltung der gespeicherten Laptimes. Dieses Menü liest die Datei Laptime.txt von der SD-Karte ein, in der alle bisherigen Rundenzeiten gespeichert sind. Zunächst wird geprüft, ob die Datei existiert und ob sie Daten enthält. Ist die Datei leer oder existiert nicht, zeigt das Display Laptime Speicher leer an. Andernfalls werden die Zeilen der Datei ausgelesen und abhängig von displayStartLine auf dem Display angezeigt, sodass ein Scrollen durch viele Runden möglich ist.

```
} else { // Datei existiert nicht
    display.setCursor(10, 49);
    display.println("Laptime Speicher ");
    display.setCursor(10, 63);
    display.println("leer");
}
```

Abbildung 54: Laptime leer

Die Anzeige erfolgt zeilenweise, wobei jede Zeile eine Rundeninformation enthält, und maximal 13 Zeilen gleichzeitig dargestellt werden. So wird verhindert, dass der Text nicht über den Bildschirmrand hinaus angezeigt wird. Der Zugriff auf die Datei erfolgt über die Standard-SD-Kartenfunktionen (SD.open(), readStringUntil('\n') und close()), wodurch ein einfaches Lesen und Anzeigen der Daten gewährleistet ist.

Zusätzlich ist in Case 5 das Löschen von Lap-Time integriert. Hierfür wird der Lap-Time Knopf für drei Sekunden gedrückt. Während der Knopf gedrückt gehalten wird, startet ein Timer (pressStart), und nach Ablauf von drei Sekunden werden die Dateien Laptime.txt und RunCounter.txt gelöscht. Gleichzeitig wird der Anzeige-Startindex displayStartLine auf 0 zurückgesetzt, und der Run-Zähler runNumber wird ebenfalls auf 0 zurückgesetzt. Um mehrfaches Auslösen der Löschung zu verhindern, wird die Variable longPressDone auf true gesetzt.

5.9.3.2.11 Case 6

Case 6 ist für die Anzeige der Extremwerte zuständig. Extremwerte umfassen unter anderem maximale Geschwindigkeit, maximale Höhe und minimale Höhe, die über den Tag hinweg vom GPS-Modul erfasst und in der Datei Extremwerte.txt auf der SD-Karte gespeichert werden. Im Menü wird die Datei geöffnet und zeilenweise ausgelesen, um die gespeicherten Werte auf dem Display darzustellen. Dabei wird ein Puffer verwendet, um die Zeilen korrekt zu verarbeiten, insbesondere wenn die Zeilen unterschiedlich lang sind. Ähnlich wie bei Case 5 werden maximal 13 Zeilen gleichzeitig angezeigt, und der Startpunkt für die Anzeige kann durch Scrollen angepasst werden. Wenn die Datei nicht existiert oder leer ist, wird die Meldung Extremwert Speicher leer auf dem Display ausgegeben.

```
const int bufferSize = 64; //Platz für längste Zeile
char lineBuffer[bufferSize];
int lineIndex = 0;
int yPos = 0;
```

Abbildung 55: Puffer Extremwerte

Zusätzlich ermöglicht Case 6 das Löschen der Extremwerte-Datei. Hierfür wird der Lap-Time Knopf ebenfalls für 3 Sekunden gedrückt. Während der Button gedrückt gehalten wird, wird die Datei Extremwerte.txt gelöscht und der Anzeige-Startindex displayStartLineExt auf 0 zurückgesetzt. Die Variable longPressDone verhindert mehrfaches Auslösen, sodass das Löschen nur einmal pro langen Druck erfolgt.

5.9.3.2.12 Case 7

Case 7 ist für die WLAN-Anzeige und Steuerung zuständig. In diesem Menü wird dem Benutzer der aktuelle WLAN-Status angezeigt, also ob das WLAN-Modul aktiviert (wlanAn = true) oder deaktiviert (wlanAn = false) ist. Zusätzlich werden die aktuelle IP-Adresse des Access Points und die konfigurierte SSID dargestellt, sodass der Benutzer sofort sieht, über welches Netzwerk das Gerät erreichbar ist.

```
case 7: { // WLAN-Anzeige
    display.clearDisplay();
    display.setTextSize(1);
    display.setCursor(10, 20);
    display.print("WLAN: ");
    display.println(wlanAn ? "An" : "Aus");
    display.setCursor(10, 50);
    if (wlanAn) {
        display.print("IP: ");
        display.println(WiFi.softAPIP());
    } else {
        display.print("IP: ");
        display.println("- - - -");
    }
    display.setCursor(10, 80);
    display.print("SSID: ");
    display.println(ssid);
    break;
}
```

Abbildung 56: Case 7

Die Bedienung erfolgt über den Lap-Time-Knopf. Ein kurzer Druck aktiviert bzw. deaktiviert den integrierten Webserver und das WLAN. Wird das WLAN gestartet, wird ein Access Point mit der angegebenen SSID und dem Passwort erstellt, und mehrere HTTP-Routen (handleRoot, handleLaptimes, handleTracking, handleExtremwerte) werden eingerichtet. Der Webserver wird anschliessend gestartet, sodass die gespeicherten Daten wie Lap-Time, Tracking-Daten oder Extremwerte über einen Webbrowser abgerufen werden können. Wird das WLAN wieder ausgeschaltet, stoppt der Webserver, das WLAN wird deaktiviert und das Modul wird in den Modus WIFI_OFF gesetzt, um Energie zu sparen.

5.9.3.3 Code Funktionen

Um die Hauptschleife übersichtlich und strukturiert zu gestalten, habe ich für bestimmte Programmabläufe externe Funktionen erstellt. Diese ermöglichen es, wiederkehrende oder komplexe Prozesse auszulagern und den Code dadurch klarer zu gliedern. In der Hauptschleife werden anschliessend nur noch die entsprechenden Funktionsaufrufe eingesetzt, was den Programmablauf nicht nur übersichtlicher, sondern auch leichter wartbar und erweiterbar macht.

5.9.3.3.1 Kompass Glättung

Zur Reduzierung starker Schwankungen bei den Messwerten habe ich eine Funktion entwickelt, die die Richtungswerte des Kompasses glättet. Zunächst wird eine Konstante ZAHLEN_MENGE mit dem Wert 50 definiert, die die Grösse des Puffers für die zuletzt gemessenen Richtungswerte festlegt. Anschliessend wird ein Array richtungsBuffer deklariert, das die letzten 50 Richtungswerte vom Typ float speichert. Die Variable ZahlenIndex verfolgt die aktuelle Position im Puffer, während die boolesche Variable buffer angibt, ob der Puffer bereits einmal komplett gefüllt wurde.

```
float glaetungMesswert(float neueRichtung) {
  richtungsBuffer[ZahlenIndex] = neueRichtung;
  ZahlenIndex = (ZahlenIndex + 1) % ZAHLEN_MENGE;
  if (ZahlenIndex == 0) buffer = true;
  int count = buffer ? ZAHLEN_MENGE : ZahlenIndex;
  float sumSin = 0;
  float sumCos = 0;
  for (int i = 0; i < count; i++) {
    float rad = richtungsBuffer[i] * DEG_TO_RAD;  // Grad -> Bogenmass
    sumSin += sin(rad);
    sumCos += cos(rad);
  }
```

Abbildung 57: Kompass Glättung

Die Funktion glaetungMesswert(float neueRichtung) übernimmt einen neuen Richtungswert als Eingabeparameter. Zunächst wird der neue Wert in das richtungsBuffer-Array an der aktuellen Position ZahlenIndex geschrieben. Danach wird der Index schrittweise erhöht und mittels Modulo-Operation auf die Puffergrösse zurückgesetzt, sodass der Puffer als Ringpuffer arbeitet und alte Werte automatisch überschreibt. Sobald der Index wieder auf null zurückspringt, wird die Variable buffer auf true gesetzt, um anzuzeigen, dass alle 50 Werte gefüllt sind.

Anschliessend wird die Anzahl der tatsächlich verwendeten Werte bestimmt: Wenn der Puffer einmal gefüllt wurde, werden alle 50 Werte berücksichtigt, sonst nur die bisher eingefügten Werte. Zur Berechnung des geglätteten Richtungswerts werden die Sinus- und Kosinus Anteile aller Werte summiert. Dies geschieht, um kreisförmige Mittelwerte korrekt zu berechnen, da Winkel sich zyklisch über 360° erstrecken und ein normales Mitteln fehlerhaft wäre. Zum Beispiel würde der Mittelwert von 359° und 1° falsch berechnet, wenn man einfach die Werte mittelt.

Mit den Summen der Sinus- und Kosinus Werte wird anschliessend die arctan2-Funktion aufgerufen, um den durchschnittlichen Winkel in Bogenmass zu berechnen. Dieser wird danach wieder in Grad umgerechnet. Schliesslich wird sichergestellt, dass der zurückgegebene Wert im Bereich von 0 bis 360° liegt. Das Ergebnis der Funktion ist somit ein geglätteter Richtungswert, der die kurzfristigen Schwankungen der Messungen reduziert und eine stabile Anzeige oder weitere Verarbeitung ermöglicht.

5.9.3.3.2 Höhe Glättung

Eine ähnliche Funktion habe ich für die Höhe geschrieben. Diese dient der Glättung von Höhenmesswerten und gleichzeitig der Erfassung von Minimal- und Maximalwerten der Höhe. Zunächst wird die Konstante NUM_ALT_SAMPLES auf 3 gesetzt, was die Grösse des Puffers für die zuletzt gemessenen Höhenwerte definiert. Das Array altitudeBuffer speichert diese Werte, während altitudeIndex die aktuelle Position im Puffer verfolgt und altitudeBufferFull angibt, ob der Puffer bereits einmal komplett gefüllt wurde. Zusätzlich werden zwei Variablen minAltitude und maxAltitude initialisiert, um die minimal und maximal gemessene Höhe zu speichern.

```
float smoothAltitude(float newAlt) {
    altitudeBuffer[altitudeIndex] = newAlt;
    altitudeIndex = (altitudeIndex + 1) % NUM_ALT_SAMPLES;
    if (altitudeIndex == 0) altitudeBufferFull = true;

    // Mittelwert berechnen
    int count = altitudeBufferFull ? NUM_ALT_SAMPLES : altitudeIndex;
    float sum = 0;
    for (int i = 0; i < count; i++) sum += altitudeBuffer[i];
    float avg = sum / count;

    // Min/Max aktualisieren
    if (newAlt < minAltitude) minAltitude = newAlt;
    if (newAlt > maxAltitude) maxAltitude = newAlt;

    return avg;
}
```

Abbildung 58: Höhe Glätten

Die Funktion smoothAltitude(float newAlt) erhält einen neuen Höhenwert als Eingabe. Dieser wird zunächst im Puffer an der aktuellen Position altitudeIndex gespeichert, danach wird der Index erhöht und durch Modulo-Operation auf die Puffergrösse zurückgesetzt, sodass alte Werte zyklisch überschrieben werden. Sobald der Index wieder auf null zurückspringt, wird das Flag altitudeBufferFull auf true gesetzt.

Im nächsten Schritt wird ein geglätteter Höhenwert berechnet, indem der Durchschnitt der aktuell vorhandenen Werte im Puffer genommen wird. Falls der Puffer noch nicht voll ist, werden nur die bisher eingefügten Werte berücksichtigt; ist der Puffer vollständig, werden alle drei Werte gemittelt. Gleichzeitig werden die Variablen minAltitude und maxAltitude aktualisiert, falls der neue Wert kleiner bzw. grösser ist als die bisher gespeicherten Extremwerte. Die Funktion gibt schliesslich den geglätteten Mittelwert zurück, der kurzfristige Schwankungen der Höhenmessung reduziert und eine stabilere Anzeige oder weitere Verarbeitung ermöglicht.

5.9.3.3.3 Extremwerte schreiben

Die Funktion updateExtremwerte() ist dafür zuständig, die Tages-Extremwerte für Höhe und Geschwindigkeit zu verwalten, zu vergleichen und auf der SD-Karte zu speichern. Zu Beginn wird die Datei /DateTag.txt geöffnet, um das Datum des letzten gespeicherten Eintrags auszulesen. Dabei wird die Datei zeilenweise gelesen, sodass die letzte Zeile in der Variablen DateTagSD gespeichert wird. Falls die Datei nicht existiert, wird DateTagSD als leerer String gesetzt. Anschliessend wird das aktuelle Datum vom GPS-Modul in DateTag formatiert, wobei Tag, Monat und Jahr in einem dd.mm.yyyy-Format gespeichert werden. Beide Strings werden getrimmt, um mögliche Leerzeichen zu entfernen.

Wenn das aktuelle Datum von dem zuletzt gespeicherten Datum abweicht, wird die Datei /Extremwerte.txt geöffnet, um die Tageswerte neu zu speichern. Dabei wird das vorherige Datum, die maximal und minimal gemessene Höhe sowie die maximale Geschwindigkeit in die Datei geschrieben. Danach werden temporäre Dateien angelegt, alte Extremwertdateien gelöscht und die neuen Werte in die jeweiligen SD-Dateien geschrieben, um Konsistenz zu gewährleisten. Abschliessend wird das aktuelle Datum in /DateTag.txt gesichert.

```
if (!DateTag.equals(DateTagSD)) {
    extFile = SD.open("/Extremwerte.txt", FILE_WRITE);
    if(extFile){
        extFile.println(String("Datum: ") + String(DateTagSD));
        extFile.println(String("Max Hoehe: ") + String((int)maxHoeheTag) + " m");
        extFile.println(String("Min Hoehe: ") + String((int)minHoeheTag) + " m");
        extFile.println(String("Max Geschw.: ") + String((int)maxSpeedTag) + " kph");
        extFile.println("");
        extFile.flush();
        extFile.close();
```

Abbildung 59: Extremwerte schreiben

Im Anschluss werden die einzelnen Extremwerte überprüft und gegebenenfalls aktualisiert. Ist updatedmaxHoehe auf true gesetzt, wird die Datei /maxHoeheTag.txt geöffnet, um den letzten gespeicherten Maximalwert auszulesen. Wenn der aktuelle Maximalwert höher ist als der gespeicherte, wird er zunächst in eine temporäre Datei geschrieben, die Originaldatei gelöscht und die temporäre Datei umbenannt. Dasselbe Prinzip gilt analog für updatedminHoehe und die Datei /minHoeheTag.txt, sowie für updatedmaxSpeed und die Datei /maxSpeedTag.txt. Dadurch wird sichergestellt, dass immer die jeweils höchsten, niedrigsten oder schnellsten Messwerte des Tages dauerhaft auf der SD-Karte gesichert werden.

Die Funktion kombiniert somit Dateiverwaltung, Vergleich aktueller Messwerte mit bestehenden Daten, sowie sichere Aktualisierung von Extremwerten auf der SD-Karte, um konsistente und belastbare Tagesstatistiken zu gewährleisten. Alle Schreibvorgänge nutzen temporäre Dateien und Flush-Operationen, um Datenverlust bei plötzlichen Stromunterbrechungen zu vermeiden. Verzögerungen (delay) sorgen für die notwendige Stabilität beim Zugriff auf die SD-Karte.

5.9.3.3.4 Webserver

Die Funktion handleRoot() bildet das Webinterface des Systems. Sie wird aufgerufen, sobald ein Benutzer im Browser die IP-Adresse des Geräts aufruft, und generiert dynamisch eine vollständige HTML-Seite, die über das integrierte WLAN-Modul ausgeliefert wird. Ziel dieser Seite ist es, dem Benutzer eine einfache grafische Oberfläche zur Verfügung zu stellen, über die die aufgezeichneten Daten (Rundenzeiten, Extremwerte und Trackingdaten) direkt eingesehen werden können.

Die Funktion erstellt zunächst den HTML-Grundaufbau inklusive CSS- und JavaScript-Code. Über CSS wird das Layout der Seite definiert, insbesondere ein Tab-System, das zwischen den drei Datenansichten "Laptimes","Extremwerte" und "Tracking" umschaltet. JavaScript steuert die Interaktivität der Seite: Es verwaltet die Tabs, sorgt für das aktive Hervorheben des aktuell geöffneten Reiters und ruft alle 500 ms über die Fetch-API die aktuellen Daten vom Webserver ab. Dadurch werden die Inhalte automatisch aktualisiert, ohne dass die Seite manuell neu geladen werden muss. Beim ersten Laden der Seite wird automatisch der Tab "Laptimes" geöffnet.

```
void handleRoot() {
  String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'><title>Pico W</title>";
  // CSS für Tabs
  html += "<style>";
  html += ".tab { overflow: hidden; border-bottom: 1px solid #ccc; }";
  html += ".tab button { background-color: inherit; border: none; outline: none; cursor: pointer; padding: 10px 20px; transition: 0.3s; }";
  html += ".tab button:hover { background-color: #ddd; }";
  html += ".tab button.active { background-color: #bbb; }";
  html += ".tabcontent { display: none; padding: 10px; }";
  html += "</style>";
```

Abbildung 60: handleRoot CSS

Nach dem Aufbau des HTML- und Skriptteils fügt die Funktion drei Schaltflächen hinzu, die jeweils den entsprechenden Datenbereich öffnen. Unterhalb dieser Buttons befinden sich die eigentlichen Inhaltsbereiche, die zunächst den Platzhaltertext "Lade... " enthalten. Die gesamte Seite wird anschliessend als HTML-Dokument an den Browser gesendet, wodurch der Benutzer über das WLAN vom Raspberry Pi Pico eine übersichtliche Oberfläche zur Datenanzeige erhält.

Die drei weiteren Funktionen handleLaptimes(), handleExtremwerte() und handleTracking() dienen als Backend-Endpunkte für die Datenbereitstellung und werden jeweils über die Fetch-Aufrufe des JavaScripts angesprochen. Jede dieser Funktionen prüft, ob die zugehörige Datei auf der SD-Karte existiert, und liest deren Inhalt zeichenweise aus. Anschliessend wird der Textinhalt über den Webserver mit dem MIME-Typ "text/plain" an den Browser übertragen, sodass dieser im entsprechenden Tab angezeigt werden kann.

Falls eine Datei nicht vorhanden ist, beispielsweise wenn noch keine Rundenzeiten aufgezeichnet wurden, sendet die Funktion eine entsprechende Statusmeldung wie "Keine Laptimes vorhanden" oder "Keine Trackingwerte vorhanden". Dadurch erhält der Benutzer stets eine klare Rückmeldung über den aktuellen Datenstand.

5.9.3.3.5 formatLapTime

Die Funktion formatLapTime wandelt eine Zeitangabe in Millisekunden in ein gut lesbares Format für Rundenzeiten um. Zunächst werden die Minuten, Sekunden und Hundertstelsekunden aus der Gesamtzeit berechnet.

```
String formatLapTime(unsigned long timeMs) {
    unsigned int minutes = timeMs / 60000;
    unsigned int seconds = (timeMs % 60000) / 1000;
    unsigned int hundredths = (timeMs % 1000) / 10;
    char buffer[20];
    sprintf(buffer, "%u:%02u.%02u", minutes, seconds, hundredths);
    return String(buffer);
}
```

Abbildung 61: Funktion formatLapTime

Anschliessend werden diese Werte mithilfe von sprintf zu einem String im Format MM:SS.HH zusammengefügt. Dabei übernimmt sprintf die Aufgabe, die Zahlenwerte in Text zu konvertieren und in einer bestimmten Form zu formatieren, zum Beispiel mit davor stehenden Nullen für Sekunden und Hundertstelsekunden. Die Funktion gibt diesen fertigen String zurück, sodass die Rundenzeit übersichtlich auf dem Display des Motorraddashboards angezeigt werden kann.

5.9.4 Probleme und Lösungen

Während der Testphase des Programms traten immer wieder verschiedene Fehler und unerwartete Verhaltensweisen auf, die analysiert und behoben werden mussten. Diese Test- und Korrekturphase war ein wichtiger Bestandteil der Entwicklungsarbeit, da sie dazu beitrug, die Stabilität, Zuverlässigkeit und Genauigkeit des Systems zu verbessern. Durch systematisches Testen sowohl auf dem Schreibtisch als auch unter realen Bedingungen am Motorrad, konnten viele kleine, aber entscheidende Probleme identifiziert und schrittweise gelöst werden.

5.9.4.1.1 SD-Karten Speicher

Eines der ersten grösseren Probleme trat beim Speichern und Lesen von Daten auf der SD-Karte auf. Nach einer gewissen Zeit im Betrieb erkannte das SD-Modul die Karte plötzlich nicht mehr, und das gesamte Programm fror ein. Dieses Verhalten war unregelmässig, manchmal geschah es direkt nach dem Einschalten, in anderen Fällen erst nach einigen Minuten Laufzeit.

Zu Beginn habe ich den Raspberry Pi Pico über die USB-Schnittstelle mit dem Computer verbunden, um mithilfe des seriellen Monitors die Programmausgaben zu beobachten. Dabei fiel mir auf, dass während des Betriebs die Verbindung zur SD-Karte verloren ging. Als ersten Schritt formatierte ich die Karte neu und setzte sie wieder ein. Zunächst funktionierte das Programm wieder, jedoch nur für etwa fünf Minuten, bevor derselbe Fehler erneut auftrat. Ich überprüfte daraufhin die Karte am PC und erhielt eine Warnmeldung, dass beschädigte Dateien auf der SD-Karte gefunden wurden. Beim Versuch, diese Dateien zu öffnen, stellte ich fest, dass einige unlesbar waren oder nur aus wirren, fehlerhaften Zeichen bestanden.

Nach einigen Recherchen im Arduino Forum fand ich den Hinweis, dass das Problem vermutlich daherkommt, dass die Dateien auf der SD-Karte nicht korrekt oder zu spät geschlossen werden. Dadurch kann es zu Schreibfehlern im Dateisystem kommen.

Um dieses Problem zu beheben, passte ich meinen Code an. Anstatt direkt in die Zieldatei zu schreiben, liess ich das Programm zunächst eine temporäre Datei anlegen. Wenn der Schreibvorgang erfolgreich abgeschlossen war, wird die alte Datei gelöscht und die temporäre Datei in den endgültigen Dateinamen umbenannt. Diese Methode erhöhte bereits die Stabilität merklich, löste das Problem jedoch noch nicht vollständig. Schliesslich fügte ich nach jedem Schliessen einer Datei eine kurze Pause mit dem Befehl delay() ein, um sicherzustellen, dass der Schreibvorgang auf der SD-Karte vollständig abgeschlossen ist, bevor der nächste Zugriff erfolgt. Nach dieser Anpassung trat der Fehler kein einziges Mal mehr auf, die SD-Karte arbeitete fortan zuverlässig und ohne Datenverluste.

```
tmpmaxHoeheTagFile = SD.open("/maxHoeheTag.tmp", FILE_WRITE);
if(tmpmaxHoeheTagFile){
    tmpmaxHoeheTagFile.println(maxHoeheTag); // Wert schreiben
    tmpmaxHoeheTagFile.flush();
    tmpmaxHoeheTagFile.close();
    delay(10);
    if (SD.exists("/maxHoeheTag.txt")) SD.remove("/maxHoeheTag.txt");
    SD.rename("/maxHoeheTag.tmp", "/maxHoeheTag.txt");
}
```

Abbildung 62: tmp Datei

Dieser Prozess zeigte deutlich, wie wichtig eine saubere Dateiverwaltung und ausreichend Pufferzeit bei Speichervorgängen auf Mikrocontrollern ist, insbesondere bei Systemen, die regelmässig Daten auf SD-Karten schreiben.

5.9.4.1.2 SD-Karten Schreibzyklus

Zu Beginn meines Projekts schrieb ich alle Messwerte direkt auf die SD-Karte, um sicherzustellen, dass im Falle eines Systemabsturzes oder eines plötzlichen Stromausfalls keine Daten verloren gehen. Dieses Vorgehen funktionierte anfangs sehr zuverlässig und vermittelte mir ein gutes Gefühl von Datensicherheit. Doch bei weiterer Recherche stellte ich fest, dass SD-Karten nur eine begrenzte Anzahl an Schreibzyklen besitzen, deutlich weniger, als ich ursprünglich angenommen hatte.

Hochwertige SD-Karten verfügen in der Regel über etwa 100.000 Schreibzyklen pro Speicherzelle. Als ich diese Zahl auf mein Projekt umrechnete, wurde mir das Problem schnell bewusst. Wenn ich zum Beispiel jede Sekunde den aktuellen Standort auf die SD-Karte schreibe, sind nach rund 27 Stunden Dauerbetrieb bereits so viele Schreibvorgänge erfolgt, dass die SD-Karte theoretisch an ihre Verschleissgrenze gelangen könnte.

Um die Lebensdauer der SD-Karte deutlich zu verlängern, musste ich also eine effizientere Speicherstrategie entwickeln. Ich entschied mich dafür, die Tracking-Daten nur noch einmal pro Minute zu speichern, anstatt jede Sekunde. Dazu speichere ich den Standort jede Sekunde in einen Puffer, der im RAM vom Raspberry Pi Pico gespeichert ist. Ist eine Minute seit letzter Speicherung auf der SD-Karte vergangen, schreibe ich den ganzen Inhalt des Puffers auf die SD-Karte. Die Rundenzeiten (Lap-Times) werden nun lediglich beim Beenden einer Messung geschrieben, und die Extremwerte (wie maximale Geschwindigkeit oder Höhe) nur dann, wenn sich tatsächlich ein neuer Rekordwert ergibt und auch dann nur der einzelne betroffene Wert, nicht die gesamte Datendatei. Diese Änderungen erforderten eine Umstrukturierung des Codes, da die Schreiblogik anders gestaltet werden musste. Durch diese Anpassungen konnte ich jedoch eine deutlich längere Lebensdauer der SD-Karte erreichen.

```
if (millis() - lastSaveTime > SAVE_INTERVAL) {
    if (gpsBuffer.length() > 0) {  // Nur speichern, wenn Daten vorhanden sind
        bool exists = SD.exists("/tracking.txt");
        trackFile = SD.open("/tracking.txt", FILE_WRITE);
```

Abbildung 63: Puffer schreiben

5.9.4.1.3 Magnetometer

Der Magnetometer, den ich als Kompass-Sensor in meinem Motorrad Dashboard verwende, misst die Magnetfelder der Erde. Aus diesen Messwerten lässt sich die magnetische Nordrichtung bestimmen, was im Laboraufbau zunächst hervorragend funktionierte. Um eventuelle Abweichungen korrigieren zu können, ergänzte ich den Code mit einer Offset-Variable, mit der sich der Kompass manuell kalibrieren lässt. Ich ging davon aus, dass die Bordelektronik des Motorrads möglicherweise magnetische Störfelder erzeugt, die das Messergebnis beeinflussen könnten und wollte diesen Effekt ausgleichen.

Doch als ich das System schliesslich am Motorrad montierte, zeigte sich ein völlig anderes Problem. Der Kompass zeigte konstant nach Süden, unabhängig von der tatsächlichen Fahrtrichtung. Nach kurzer Überlegung wurde mir klar, woran das lag, der Benzintank des Motorrads besteht aus Metall und ist magnetisch. Dadurch überlagerte sein Magnetfeld das natürliche Erdmagnetfeld, sodass der Sensor nicht mehr den Erdmagnetismus, sondern das Magnetfeld des Tanks mass. Der Kompass zeigte also immer in Richtung Tank. Ich versuchte daraufhin, das Motorrad Dashboard an einer anderen Stelle zu montieren, doch das Magnetfeld des Motorrads war insgesamt zu stark, sodass der Kompass auch dort keine zuverlässigen Messwerte liefern konnte.

Um dieses Problem zu lösen, entschied ich mich, den Code für die Richtungsbestimmung anzupassen. Anstatt die Kompasswerte aus dem Magnetometer zu verwenden, berechne ich nun die Fahrtrichtung aus den GPS-Daten. Diese Lösung umgeht das Problem des gestörten Magnetfelds vollständig. Allerdings bringt sie auch zwei entscheidende Nachteile mit sich:

Erstens steht die Richtungsanzeige nur zur Verfügung, wenn das GPS-Modul ein gültiges Signal (GPS-Fix) hat. In Tiefgaragen oder Tunneln, wo kein Satellitenempfang möglich ist, fällt die Kompassfunktion also aus.

Zweitens kann die Fahrtrichtung nur während der Bewegung bestimmt werden, das GPS-System erkennt sonst nicht, in welche Richtung sich das Motorrad orientiert. Um fehlerhafte Anzeigen zu vermeiden, habe ich im Code zusätzlich eine Bedingung eingebaut, dass der Kompass erst ab einer Geschwindigkeit von mindestens 2 km/h aktiv ist. Grund dafür ist, dass das GPS im Stillstand oft eine minimale Abweichung in der Positionsmessung aufweist, die zu scheinbaren Bewegungen führen kann. Diese Schwankungen hätten sonst dazu geführt, dass der Kompass zufällige Richtungen anzeigt, obwohl das Motorrad steht.

```
// Gradzahl im Kompass unterhalb des Zeigers
//int gradX = CENTER_X - 20; // mittig
int gradY = CENTER_Y - (RADIUS - 75); // direkt unter Zeiger
//display.setCursor(gradX, gradY);
/*display.print((int)heading);
display.println(" Grad");
*/

if (gpsFix && speedKmh > 2.0) {
    String gradText= String((int)heading) + " Grad";
    int textWidth = gradText.length() *6;
    int gradX = CENTER_X - (textWidth /2);
    display.setCursor(gradX, gradY);
    display.print(gradText);
    //display.print((int)heading);
    //display.println(" Grad");
} else {
    String gradText = "----";
    int textWidth = gradText.length() *6;
    int gradX = CENTER_X - (textWidth /2);
    display.setCursor(gradX, gradY);
    display.println(gradText);
}
```

Abbildung 64: Kompass Anpassung

Mit dieser Anpassung funktioniert die Richtungsanzeige nun zuverlässig während der Fahrt und bietet trotz der Einschränkungen eine praxisnahe und stabile Lösung für den Einsatz am Motorrad.

5.10 Laboraufbau

Um die einzelnen Komponenten meines Projekts zu testen, habe ich jeweils ein passendes Testprogramm auf den Raspberry Pi Pico geladen und die dafür erforderliche Schaltung aufgebaut. Auf diese Weise konnte ich feststellen, ob die jeweiligen Bauteile wie vorgesehen angesteuert werden und korrekt funktionieren.

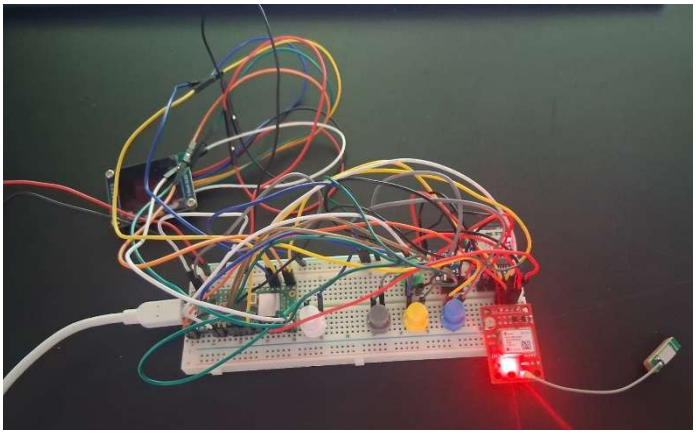


Abbildung 65: Laboraufbau

Nachdem die Einzelkomponenten erfolgreich getestet waren, wollte ich überprüfen, ob mein Gesamtprogramm in der Praxis funktioniert. Dazu habe ich das Motorrad Dashboard zunächst provisorisch nach meinem Schaltplan auf einem Breadboard aufgebaut. Mit diesem Aufbau konnte ich zwei Dinge gleichzeitig testen. Erstens, ob mein Schaltplan korrekt gezeichnet war, und zweitens, ob das Programm wie geplant arbeitete.

Während des Aufbaus unterlief mir jedoch ein Fehler. Beim Anschliessen des SD-Moduls vertauschte ich die 5V- und GND- Leitungen. Dies bemerkte ich erst, als das Programm keine Verbindung zur SD-Karte herstellen konnte. In der Folge habe ich das defekte Modul durch ein neues, etwas kompakteres SD-Modul ersetzt. Danach funktionierte das Ansprechen der SD-Karte.

Im weiteren Verlauf lud ich regelmässig das aktuelle Programm auf den Raspberry Pi Pico, um den Fortschritt zu prüfen und zu sehen, welche Funktionen bereits korrekt arbeiteten und welche noch angepasst werden mussten. Durch dieses Vorgehen konnte ich mein System Schritt für Schritt verbessern und sicherstellen, dass sowohl Hardware als auch Software zuverlässig zusammenspielten.

Während den Tests fiel das GPS-Modul plötzlich aus, und ich wusste zunächst nicht, woran es lag. Mir kam nur der Gedanke, dass vielleicht eine Überspannung vom Netzteil den Schaden verursacht hatte. Also bestellte ich ein neues Modul und baute es wieder ein. Danach funktionierte es wieder einwandfrei. Seitdem habe ich die Netzteilspannung vorsorglich von 5 V auf 4,5 V reduziert.

5.11 Zusammenbau

Nachdem das Programm vollständig entwickelt und auf dem Laboraufbau erfolgreich getestet war, begann ich mit dem endgültigen Zusammenbau des Dashboards. Zuerst überlegte ich mir, welches Gehäuse sich am besten für den Einbau eignen würde. Mir war von Anfang an klar, dass es wasserdicht und aus Kunststoff bestehen musste, damit das GPS-Modul und der Magnetometer nicht beeinträchtigt werden. Ausserdem wollte ich das Display so einbauen, dass das gesamte System auch nach der Montage wasserdicht blieb. Nach einiger Überlegung kam mir die Idee, eine Kunststoffdose mit transparentem Deckel zu verwenden. Dadurch konnte ich das Display an der Innenseite des Deckels befestigen und lediglich die Schraubenlöcher abdichten.

Anschliessend mass ich die Gabelbrücke meines Motorrads, da ich das Dashboard dort montieren wollte. Dabei stellte ich fest, dass das Gehäuse maximal 13 Zentimeter lang sein durfte. Nach kurzer Suche fand ich ein passendes Modell mit den Massen 12 x 8 x 5.5 Zentimeter, das meine Anforderungen erfüllte. Für den Aufbau der Elektronik verwendete ich zwei Lochrasterplatinen, die ich übereinander befestigte. Diese hatte ich noch von einem früheren Projekt an der TEKO übrig. Auf diesen Platinen steckte ich die einzelnen Bauteile und positionierte den Ein- und Ausschalter sowie die Kabelverschraubung für die Spannungsversorgung und den Lap-Time-Taster an den vorgesehenen Stellen. Danach überprüfte ich anhand der Abmessungen meines Probeaufbaus, ob das gewählte Gehäuse ausreichend Platz bot, was sich bestätigte. Ich bestellte die Dose und lud mir die technische Zeichnung herunter, um die Bohrungen und Ausschnitte genau planen und einzeichnen zu können.

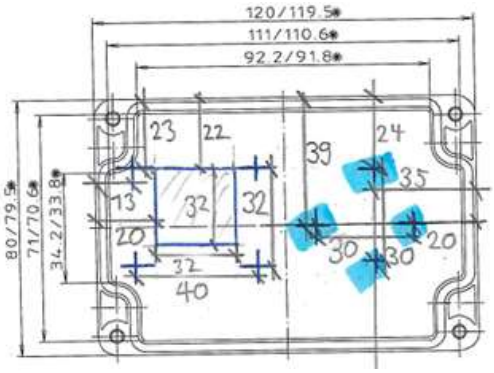
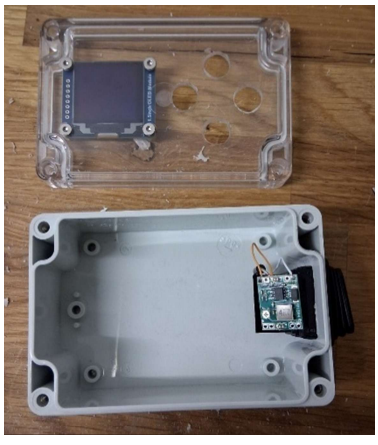


Abbildung 66: Skizze Deckel

Auf der rechten Seite des Deckels plante ich die vier Taster, die in Form eines Kreuzes angeordnet sind. Der Abstand zwischen den einzelnen Tastern beträgt jeweils 1,5 Zentimeter, gemessen von Mitte zu Mitte. Auf der linken Seite des Deckels positionierte ich das Display. Dieses setzte ich so weit wie möglich nach links, achtete jedoch darauf, dass es sich noch sauber und stabil anschrauben lässt. An der linken Gehäuseseite markierte ich in der Mitte der Fläche den Punkt, von dem aus ich die Abmessungen für den Ausschnitt des Hauptschalters bestimmte. Von da aus vermasste ich die Ausschnitts Grösse, damit der Hauptschalter danach in der Mitte sitzt.



Auf der rechten Gehäuseseite markierte ich ebenfalls die Mitte, um dort das Loch für die Kabelverschraubung zu Zeichnen. Die Bohrungen für die vier Taster legte ich mit einem Durchmesser von 1,2 Zentimetern fest, während das Loch für die Kabelverschraubung einen Durchmesser von 1,62 Zentimetern erhielt.

Als das Gehäuse ankam, legte ich den Testaufbau hinein, um sicherzugehen, dass alles passte. Anschliessend bohrte ich die geplanten Löcher und fertigte die notwendigen Ausschnitte. Im unteren Teil der Dose setzte ich links den Hauptschalter und rechts die Kabelverschraubung ein, während ich am Deckel zunächst keine Komponenten monierte. Danach begann ich mit dem Zusammenlöten der Komponenten.

Abbildung 67: Ausschnitte Gehäuse

Um den Überblick zu behalten, druckte ich mir den Schaltplan aus und markierte jede fertiggestellte Verbindung mit einem orangen Leuchtstift. So wusste ich jederzeit genau, wie weit ich war und konnte nach einer Pause problemlos weiterarbeiten. Damit die Bauteile nicht frei in der Luft hingen, befestigte ich sie mit doppelseitigem, dickem Klebeband auf der Platine. Das diente gleichzeitig als Schutz vor den Vibrationen, die beim Fahren auftreten. Leitung für Leitung lötete ich alle Verbindungen, bis die gesamte Schaltung fertiggestellt war.

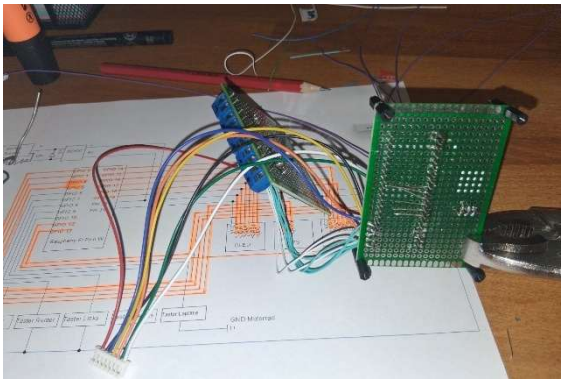


Abbildung 68: Zusammenlöten

Für die Bedientasten verwendete ich Schraubklemmen und für das Display das mitgelieferte Kabel mit Stecker, damit der Deckel bei Bedarf komplett abgenommen werden kann, ohne Lötstellen trennen zu müssen. Als die Elektronik vollständig verdrahtet war, setzte ich sie in das Gehäuse ein und verschraubte sie durch den Boden. Danach folierte ich den Deckel mit schwarzer Folie und schnitt eine Aussparung für das Display aus, sodass nur der Bildschirm sichtbar blieb und das Innere des Gehäuses blickdicht war. Für die vier Bedientasten auf dem Dashboard schnitt ich passende Löcher in die Folie, setzte die Taster ein und verschraubte sie, damit sich die Folie an diesen Stellen nicht ablöst. Die Folie brachte ich so auf, dass sie an den Ecken leicht überlappt und so einen sauberen, durchgehenden Abschluss bildet. Auch für die Schrauben des Deckels fertigte ich kleine Öffnungen, damit sich dieser problemlos befestigen lässt.

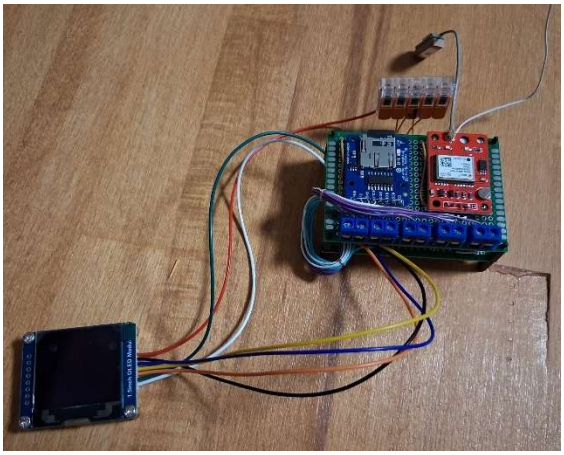


Abbildung 69: Fertige Elektronik

Als das Dashboard fertig montiert war, begann ich mit dem Einbau am Motorrad. Den Lap-Time-Taster befestigte ich am linken Lenkerstummel und führte das Kabel bis zur vorgesehenen Position des Dashboards. Die Spannungsversorgung schloss ich über den vorhandenen Ladestecker an, der sich in der Verkleidung neben der Batterie befindet. Dafür musste ich die Seitenverkleidung entfernen, um das Kabel sauber von der Gabelbrücke bis zur Batterie zu führen. Anschliessend verband ich den Lap-Time Taster und die Stromversorgung mit den vorgesehenen Klemmen, die ich eingebaut hatte, um bei Bedarf die Spannung einfach trennen zu können. Danach verband ich diese Klemmen mit den Leitungen, die ich zuvor an den DC-DC-Stepdown-Wandler gelötet habe.

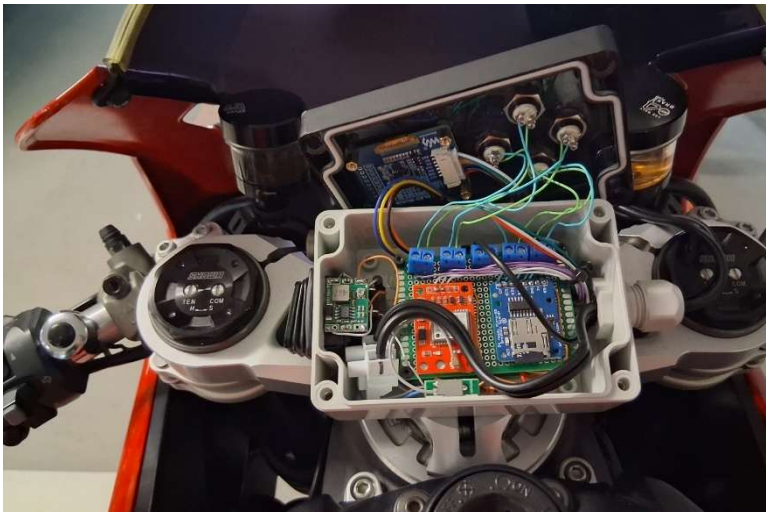


Abbildung 70: Montage Elektronik

Nachdem alles angeschlossen war, überprüfte ich die Spannung. Am Eingang mass ich 13,35 Volt, und auf der Sekundärseite stellte ich mithilfe des integrierten Potentiometers eine Spannung von 4,9 Volt ein. Danach lötete ich die sekundäre Seite an. Der Minuspol geht direkt an den Ground des Raspberry Pi Pico, während der Pluspol über eine Verteilerklemme geführt wird, an der alle Komponenten angeschlossen sind. Dadurch kann jedes Bauteil bei Bedarf einzeln spannungsfrei geschaltet werden.

Nachdem alle Verbindungen gemacht waren, schaltete ich das Dashboard ein und stellte erfreut fest, dass alles wie geplant funktionierte. Zum Schluss befestigte ich die GPS-Antenne an der Innenseite des Gehäuses, damit sie sicher fixiert ist, schraubte den Deckel drauf und führte schliesslich meine erste Probefahrt mit dem vollständig montierten Dashboard durch.

Nachdem alles wie vorgesehen funktionierte, trennte ich die Spannungsversorgung sowie die Leitung zum Lap-Time Taster auf und montierte passende Steckverbindungen. Dazu verwendete ich isolierte Rundstecker, Rundsteckhülsen, Flachsteckzungen und Flachsteckhülsen. Am Pluskabel, das von der Batterie kommt, brachte ich eine Flachsteckhülse an. Auf der Seite, die zum Dashboard führt, montierte ich eine Flachsteckzunge. Beim Minuskabel verwendete ich hingegen eine Rundsteckhülse auf der Batterieseite und einen Rundstecker auf der Seite zum Dashboard. Durch die Verwendung unterschiedlicher Steckertypen für Plus und Minus ist sichergestellt, dass die Leitungen nicht vertauscht und somit keine falschen Verbindungen hergestellt werden können.

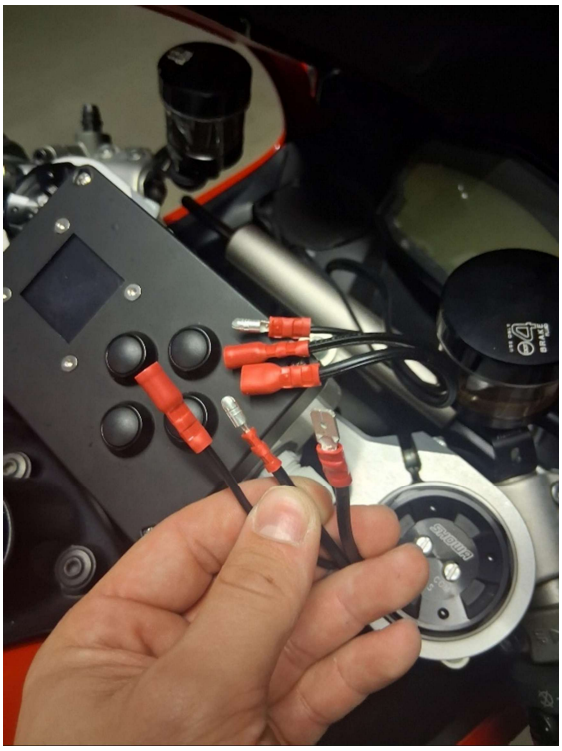


Abbildung 71: Verbindungsstecker

Beim Lap-Time Taster ging ich nach demselben Prinzip vor. Auf der Seite des Tasters montierte ich einen Rundstecker, auf der Seite des Dashboards eine passende Rundsteckhülse. Diese Anordnung wählte ich, weil an der Dashboard-Seite eine Spannung von 3,3 Volt anliegt, die so auch dann sicher isoliert bleibt, wenn kein Stecker verbunden ist. Dieses Prinzip entspricht dem der Batterieanschlüsse, bei denen ebenfalls sichergestellt ist, dass offene Leitungen keine Spannung führen können.

Am Lap-Time Taster montierte ich einen Rundstecker, da der Minus von der Batterie, mit einer Rundsteckhülse ausgestattet ist. Auf diese Weise kann man diese verbinden, ohne dass ein Kurzschluss entsteht. Hätte ich stattdessen eine Flachsteckzunge verwendet, bestünde die Gefahr, dass diese versehentlich mit dem Pluspol verbunden wird, was beim Betätigen des Lap-Time Tasters zu einem Kurzschluss führen würde. Ausserdem verhindert diese Lösung, dass am GPIO-Pin des Raspberry Pi Pico jemals die 12 Volt Batteriespannung anliegt. Stattdessen kann nur der Minuspol auf den Pin gelegt werden, was für die Schaltung unbedenklich ist.

Für die Montage des Dashboards am Motorrad machte ich mir lange Gedanken. Ursprünglich hatte ich geplant, eine Gabelmontage Halterung von Quad Lock zu verwenden, um das Dashboard daran zu befestigen. Leider gibt es für mein Motorrad jedoch keine passende Halterung, da diese eine gerade Lenkrohraufnahme voraussetzt. Bei meinem Motorrad befindet sich am oberen Ende des Lenkkopfschafts jedoch eine Schaftmutter, die nicht bündig mit dem Lenkrohr abschliesst. Dadurch war es nicht möglich, die Quad-Lock-Halterung zu montieren und diese Idee musste ich verwerfen.

Daraufhin überlegte ich verschiedene Alternativen, beispielsweise das Dashboard mithilfe von Klemmen direkt an der Gabelbrücke zu befestigen. Allerdings fand ich keine zufriedenstellende Lösung, bei der die Gabelbrücke nicht verkratzt oder beschädigt würde. Ich informierte mich auch über verschiedene Lenkerbefestigungen, doch an meinem Motorradlenker befindet sich zu wenig freier Platz, um eine solche Halterung montieren zu können.
Schliesslich entschied ich mich für eine einfache, aber funktionale Variante. Die Befestigung des Dashboards erfolgt über zwei Kabelbinder, die um die Gabelbrücke gelegt und festgezogen werden. Diese Methode ermöglicht eine schnelle Montage und Demontage und bietet dennoch einen stabilen Halt während der Fahrt.


Abbildung 72: Griff links


Abbildung 73: Griff rechts

Ein Nachteil dieser Befestigungsart besteht jedoch darin, dass beim wiederholten Montieren und Demontieren neue Kabelbinder verwendet werden müssen, was zu einem gewissen Materialverbrauch führt. Trotz dieses kleinen Nachteils hat sich diese Lösung in der Praxis als zuverlässig und praktikabel erwiesen.

5.12 Komponent

Im folgenden Kapitel sind sämtliche Bauteile aufgelistet, welche für das Projekt verwendet wurden. Dabei werden sowohl die elektronischen Komponenten als auch die Materialien des Gehäuses und der Stromerschliessung vorgestellt.

5.12.1 Komponenten Elektronik

Produkt:	Bild:	Link / QR-Code:
Raspberry Pi WH		Rasberry Pi WH
GPS-Modul		GPS Modul
Display		Display
Magnetometer		Magnetometer

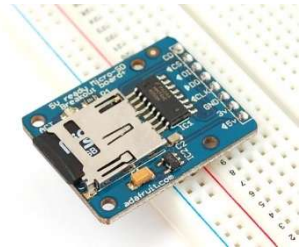
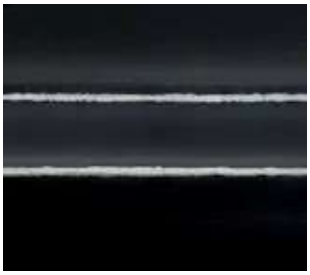
SD-Karte		SD-Karte
DC-DC Step-Down Abwärtswandler		DC-DC Step-Down Abwärtswandler
Sicherung 2A		Sicherung 2A
Schottky-Diode DO-201		Schottky-Diode DO-201
Leiterplatten Kit		Leiterplatten-Kit
SD-Modul Adafruit		SD-Modul Adafruit

Abbildung 74: Komponentenliste Elektronik

5.12.2 Komponenten Gehäuse und Zubehör

Produkt:	Bild:	Link / QR-Code:
Wippschalter IP65		Wippschalter IP65
Drucktaster IP65		Drucktaster IP65
Lenkrad Taster		Lenkrad Taster
Gehäuse		Gehäuse
SAE-Verlängerungskabel		SAE-Verlängerungskabel

Rundstecker		Rundstecker
Rundsteckhülsen		Rundsteckhülsen
Flachsteckzunge		Flachsteckzunge
Flachsteckhülsen		Flachsteckhülsen
HEX Schrauben Inox M3x20		HEX Schrauben Inox M3x20
Plotter Folie		Plotter Folie

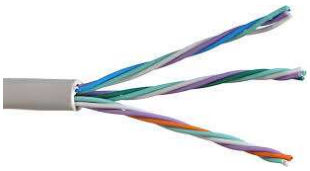
U72 3x4x0.5		U72 3x4x0.5
Powerbond Montageband		Powerbond Montageband

Abbildung 75: Komponentenliste Gehäuse und Zubehör

5.13 Gebrauchsanleitung

5.13.1 Einleitung

Das Motorrad Dashboard bietet eine Vielzahl an Funktionen, die sowohl der Information als auch der Analyse des Fahrverhaltens dienen. Es zeigt in Echtzeit die aktuelle Geschwindigkeit, die Höhe sowie die Fahrtrichtung an. Darüber hinaus verfügt es über eine Lap-Time Funktion, mit der die gefahrenen Rundenzeiten erfasst und gespeichert werden. Diese können später bequem eingesehen und miteinander verglichen werden.

Zusätzlich speichert das System die täglichen Maximal- und Minimalwerte der Höhe sowie die höchste erreichte Geschwindigkeit. Eine integrierte Tracking-Funktion ermöglicht es, nachzuvollziehen, welche Strecken während einer Fahrt zurückgelegt wurden. Alle gespeicherten Daten, einschliesslich Tracking, Rundenzeiten und Extremwerte, können direkt über ein verbundenes Smartphone ausgelesen werden.

Diese Dokumentation beschreibt die Inbetriebnahme, Nutzung und Wartung des Systems und bietet einen umfassenden Überblick über die Funktionsweise des Motorrad Dashboards.

5.13.2 Inbetriebnahme

1. **Vorbereitung:**

    - **Dashboard montieren:** Befestigen Sie das Motorrad Dashboard am Motorrad, damit es festsitzt und die Sicht auf die Pflichtanzeigen nicht verdeckt wird.

    - **Lap-Time-Taster positionieren:** Montieren Sie den Lap-Time Taster in der Nähe des linken Griffes, sodass Sie ihn bequem mit dem Daumen betätigen können.

    - **Verbindung des Lap-Time-Tasters:** Stecken Sie das Kabel des Lap-Time Tasters über den Rundstecker in die Rundsteckhülse am Dashboard.

    - **Batterie anschliessen:** Verbinden Sie die Plus- und Minus- Hülsen des Batteriekabels mit den vorgesehenen Steckern. Achten Sie darauf, dass der Minuspol über den Rundstecker und der Pluspol über den Flachzungenstecker angeschlossen wird.

2. **Überprüfung:**

    - Achten Sie darauf, dass die Batteriespannung 12 V beträgt. Falls eine andere Spannung vorliegt, muss diese über das Potentiometer am DC-DC-Wandler eingestellt werden, sodass die Sekundärseite 5 V nicht überschreitet.
    - Schalten Sie das Dashboard über den Hauptschalter ein. Es sollte nun ein Startbildschirm auf dem Display erscheinen. Bleibt das Display dunkel, prüfen Sie bitte, ob eine SD-Karte eingesetzt ist. Das Dashboard unterstützt SD-Karten von 2 GB bis 32 GB, formatiert im FAT32-Format.

5.13.3 Bedienung des Dashboards

1. **Startbildschirm:**

   Drücken Sie den linken und rechten Taster miteinander für 3 Sekunden. Dann kommen Sie zum Hauptmenüpunkt.

2. **Menüfunktionen:**

   Das Motorrad Dashboard verfügt über eine Hauptanzeige und mehrere Nebenanzeigen, die unterschiedliche Informationen darstellen:

   Die Hauptanzeige zeigt die aktuelle Geschwindigkeit in km/h, die aktuelle Höhe in Metern. Die Fahrtrichtung wird Ihnen angezeigt, sofern sich das Motorrad schneller als 2 km/h bewegt.

   Die Nebenanzeige 1 gibt die aktuelle Geschwindigkeit in km/h wieder, während die Nebenanzeige 2 die aktuelle Höhe sowie die tiefste und höchste Höhe seit dem Einschalten des Dashboards in Meter über Meer anzeigt.

   In der Nebenanzeige 3 wird ein digitaler Kompass dargestellt. Die Nebenanzeige 4 dient zur Nutzung der Lap-Time Funktion, während in der Nebenanzeige 5 die gespeicherten Rundenzeiten abgerufen werden können.

   Die Nebenanzeige 6 zeigt die Extremwerte an. Schliesslich ermöglicht die Nebenanzeige 7 die Steuerung des WLAN-Moduls, um die Verbindung ein- oder auszuschalten.

3. **Steuerung:**

Anzeige	Taster links	Taster rechts	Taster rauf	Taster runter	Taster Lap-Time
Hauptanzeige	Wechselt zu Nebenanzeige 6	Wechselt zu Nebenanzeige 1	----	Wechselt zu Nebenanzeige 7	3 Sekunden Druck löscht Tracking Datei
Nebenanzeige 1	Wechselt zu Hauptanzeige	Wechselt zu Nebenanzeige 2	----	----	----
Nebenanzeige 2	Wechselt zu Nebenanzeige 1	Wechselt zu Nebenanzeige 3	----	----	----
Nebenanzeige 3	Wechselt zu Nebenanzeige 2	Wechselt zu Nebenanzeige 4	---	----	----
Nebenanzeige 4	Wechselt zu Nebenanzeige 3	Wechselt zu Nebenanzeige 5	----	----	Kurzer Druck startet/ setzt neue Runde, 2 Sekunden druck stoppt die Messung
Nebenanzeige 5	Wechselt zu Nebenanzeige 4	Wechselt zu Nebenanzeige 6	Scrollt nach oben	Scrollt nach unten	3 Sekunden druck löscht Laptime Datei
Nebenanzeige 6	Wechselt zu Nebenanzeige 5	Wechselt zu Hauptanzeige	Scrollt nach oben	Scrollt nach unten	3 Sekunden druck löscht Extremwerte Datei
Nebenanzeige 7	----	----	Wechselt zu Haupanzeige	----	Schaltet WLAN ein/aus

Abbildung 76: Steuerung Dashboard

4. **WLAN-Verbindung:**
    - WLAN einschalten
    - Mit dem WLAN PicoW_Hotspot verbinden
    - Passwort eingeben: 12345678
    - Im Browser die angezeigte IP-Adresse eingeben
    - Gewünschter Tab öffnen
5. **Tracking:**
    - Mit dem WLAN PicoW_Hotspot verbinden
    - Tracking Tab wählen
    - Alle Datenpunkte kopieren, mit Tracking, Latitude und Longitude
    - Auf die Webseite https://www.gpsvisualizer.com/convert_input gehen
    - Output Format auf GPX setzen
    - Datenpunkte einfügen
    - Konvertieren
    - Datei Downloaden
    - Auf die Webseite https://www.google.com/maps/d/u/0/ gehen
    - Neue Karte erstellen
    - Importieren anklicken
    - Die vorhin heruntergeladene GPX-Datei auswählen
6. **Ausschalten:**
    - Zum Ausschalten kippen Sie den Hauptschalter auf 0.
7. **Demontage:**
    - Schalten Sie das Dashboard aus.
    - Trennen Sie das Batteriekabel vom Dashboard.
    - Trennen Sie das Kabel des Lap-Time-Tasters vom Dashboard.
    - Entfernen Sie anschliessend das Motorrad Dashboard vom Fahrzeug.

5.13.4 Fehlersuche

Problem	Mögliche Ursache	Lösung
Keine Funktion 1	Batterie leer, Batterieanschluss fehlt, oder Hauptschalter nicht eingeschaltet	Prüfen Sie die Stromversorgung.
Keine Funktion 2	Überlast oder Kurzschluss	Überprüfen Sie die Flachsicherung
Keine Funktion 3	Falsche Spannung	Überprüfen Sie die Batterie und die Sekundärspannung
Display leuchten nicht	Display defekt oder Stecker ist rausgefallen	Überprüfen Sie das Displaykabel
Keine Werte bei Höhe, Geschwindigkeit oder Richtungsanzeige	Kein GPS-Empfang	Gehen Sie an einen Ort mit freier Sicht zum Himmel
Taster reagieren nicht	Taster Anschluss defekt oder Verbindungsunterbruch	Überprüfen Sie die Taster Klemmen
Daten werden nicht gespeichert	SD-Karte ist voll oder defekt	Überprüfen Sie die SD-Karte
System reagiert nicht	Das System hat einen Fehler	Ausschalten, 10 Sekunden warten, Einschalten

Abbildung 77: Fehlersuche

5.13.5 Sicherheitshinweise

- Berühren Sie niemals die Platine oder andere leitende Komponenten.
- Halten Sie das Dashboard von Feuchtigkeit und extremen Temperaturen geschützt.
- Verwenden Sie nur eine 2A Flachsicherung
- Stellen Sie sicher, dass alle Lötstellen sauber und frei von Kurzschlüssen sind.
- Lassen Sie das Dashboard nicht zu lange eingeschaltet, wenn der Motor nicht läuft.

5.14 Kostenkontrolle

Ich habe sämtliche Komponenten und Materialien in der Schweiz bezogen, nicht etwa über Aliexpress, was zwar günstiger ist, dafür aber deutlich längere Lieferzeiten hat. Zudem finde ich den Grundgedanken, die Schweizer Wirtschaft durch Einkäufe im Inland zu stärken, überzeugend genug, um ein wenig mehr Geld in die Hand zu nehmen.

Die Gesamtkosten für das benötigte Material belaufen sich auf CHF 323.50. Diese setzen sich aus verschiedenen Ausgabenpunkten zusammen:

1. Elektronik Komponenten - CHF 153.3
2. Gehäuse Komponenten - CHF 128.65
3. Lieferkosten Backyard Racing - CHF 8.90
4. Lieferkosten Conrad - CHF 9.95
5. Lieferkosten Distrelec - CHF 8.00
6. Dreimalige Lieferkosten Digitec - CHF 14.70

Die Materialien wurden über mehrere Händler beschafft, um ein optimales Preis-Leistungs-Verhältnis zu erreichen.

Nachdem mir sowohl das GPS- als auch das SD-Karten-Modul kaputtgegangen waren, musste ich zweimal bei Digitec nachbestellen. Das hat mich zusätzlich CHF 45.60 gekostet.

Auch wenn durch defekte Modul zusätzliche Kosten entstanden sind, bin ich mit der Materialbeschaffung insgesamt sehr zufrieden. Alle Komponenten konnten rechtzeitig und in guter Qualität bezogen werden.

5.15 Kosten-Nutzen-Analyse

Vergleicht man die Gesamtkosten von CHF 323.50 für das selbst entwickelte Motorrad Dashboard mit handelsüblichen Lösungen, wird schnell deutlich, dass der Eigenbau auf den ersten Blick teurer erscheint. Einfache Tachos oder GPS-basierte Anzeigen für Motorräder sind im Handel bereits ab rund CHF 50 erhältlich. Diese Geräte erfordern keine Entwicklungszeit, sind sofort einsatzbereit und bieten meist die grundlegenden Funktionen wie Geschwindigkeits- oder Kilometeranzeige.

Allerdings lassen sich solche Standardgeräte kaum mit dem individuell entwickelten Dashboard vergleichen. Das selbst programmierte System bietet eine deutlich grössere Funktionsvielfalt. Neben der Geschwindigkeitsanzeige zeigt es auch die aktuelle Höhe sowie die Fahrtrichtung über einen digitalen Kompass an. Zudem verfügt es über eine Lap-Time Funktion zur Aufzeichnung und Speicherung von Rundenzeiten sowie eine Tracking-Funktion, mit der gefahrene Strecken nachverfolgt werden können. Diese Kombination ist bei kommerziellen Produkten in dieser Preisklasse nicht erhältlich.

Darüber hinaus bietet das Projekt einen erheblichen Lern- und Erfahrungswert. Durch die eigenständige Entwicklung, Programmierung und den Aufbau der Elektronik wurden wertvolle Kenntnisse in den Bereichen Mikrocontroller-Programmierung, Sensorik, Schaltungsaufbau und Systemintegration erworben. Dieser Wissenszuwachs stellt einen immateriellen, aber bedeutenden Mehrwert dar, der über den reinen Produktnutzen hinausgeht.

Insgesamt ist das selbstgebaute Dashboard zwar kostenintensiver und aufwändiger in der Herstellung als ein handelsübliches Motorrad Display, doch rechtfertigen die zusätzlichen Funktionen, die individuelle Anpassbarkeit sowie der hohe Lerngewinn den höheren Aufwand. Für Nutzer, die lediglich eine einfache Geschwindigkeitsanzeige benötigen, reicht ein Standardprodukt aus, wer jedoch ein vielseitiges, personalisiertes und technisch anspruchsvolles System möchte, profitiert deutlich vom Eigenbau.

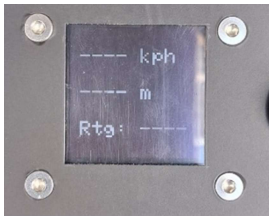
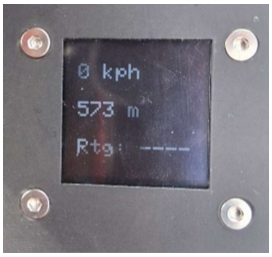
5.16 Inbetriebnahme

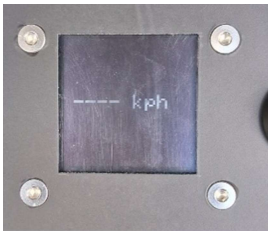
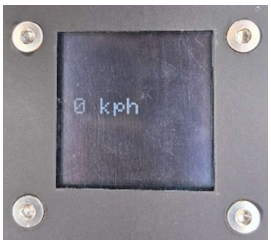
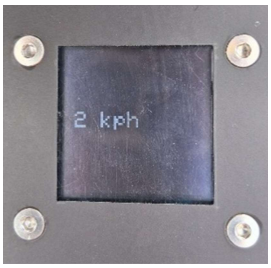
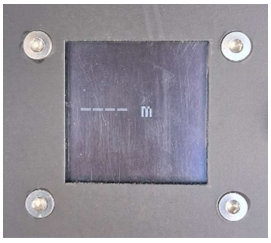
Vor dem Einschalten wurden folgende Punkte überprüft:

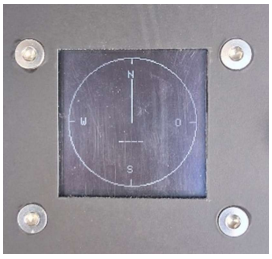
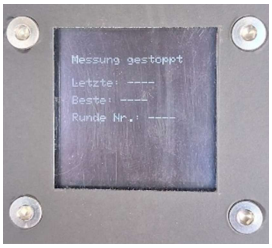
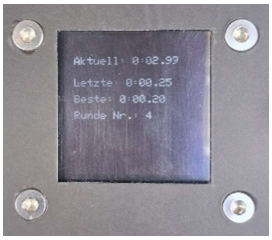
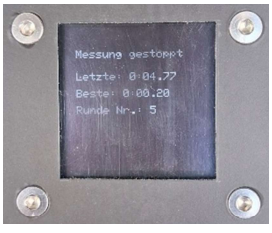
- Alle Bauteile sind gemäss Schaltplan verbaut.
- Lötstellen sind sauber und fehlerfrei.
- Kabelverbindungen sind korrekt angeschlossen.
- Alle Verbindungen mittels Multimeter durchgepiepst.
- Alle Schrauben sind angezogen
- Plus und Minuspol sind korrekt angeschossen
- Die Sekundärspannung ist auf 4.9V eingestellt

5.17 Funktions Kontrolle Display Anzeigen

Bei der Funktionsprüfung wurde die Displayanzeige des Dashboards in verschiedenen Situationen getestet. Die Tabelle zeigt die Ergebnisse dieser Tests und vergleicht sie mit den erwarteten Anzeigen. Alle Anzeigen haben dabei einwandfrei funktioniert, egal ob mit oder ohne GPS-Empfang, im Stillstand oder während der Fahrt.

Zustand	Ist Displayanzeige	Soll Displayanzeige
Dashboard eingeschaltet Startanzeige		Zeigt die Version und Namen des Programms.
Im Menüpunkt Hauptanzeige (Case 0) Ohne GPS empfang		Zeigt für die Geschwindigkeit, Höhe und Richtung einen Platzhalter.
Im Menüpunkt Hauptanzeige (Case 0) Mit GPS empfang stillstehend (< 3 km/h)		Zeigt für die Geschwindigkeit und Höhe den Wert an und für die Richtung einen Platzhalter.
Im Menüpunkt Hauptanzeige (Case 0) Mit GPS empfang in Bewegung (> 2km/h)		Zeigt für die Geschwindigkeit, Höhe und Richtung den Wert an.
Im Menüpunkt Hauptanzeige (Case 0) Lap-Time Taster 3 Sekunden gedrückt		Informiert, dass die Tracking Datei gelöscht wurde.

Im Menüpunkt Nebenanzeige 7 (Case 7) WLAN aus		Zeigt den WLAN-Status, die SSID des WLAN und einen Platzhalter für dir IP-Adresse.
Im Menüpunkt Nebenanzeige 7 (Case 7) WLAN ein		Zeigt den WLAN-Status, die IP-Adresse und die SSID des WLAN.
Im Menüpunkt Nebenanzeige 1 (Case 1) Ohne GPS empfang		Zeigt einen Platzhalter für die Geschwindigkeit.
Im Menüpunkt Nebenanzeige 1 (Case 1) Mit GPS empfang stillstehend		Zeigt die Aktuelle Geschwindigkeit.
Im Menüpunkt Nebenanzeige 1 (Case 1) Mit GPS empfang in Bewegung		Zeigt die Aktuelle Geschwindigkeit.
Im Menüpunkt Nebenanzeige 2 (Case 2) Ohne GPS empfang		Zeigt einen Platzhalter für die Höhe.

Im Menüpunkt Nebenanzeige 2 (Case 2) Mit GPS empfang		Zeigt die Aktuelle Höhe, die minimale und maximale Höhe, seitdem das Dashboard eingeschaltet wurde.
Im Menüpunkt Nebenanzeige 3 (Case 3) Ohne GPS empfang oder mit GPS empfang stillstehend (< 3 km/h)		Zeigt einen Platzhalter für die Richtung.
Im Menüpunkt Nebenanzeige 3 (Case 3) Mit GPS empfang in Bewegung (> 2km/h)		Zeigt die Richtung in Grad an und als Digitalen Kompass.
Im Menüpunkt Nebenanzeige 4 (Case 4) Keine Messung gestartet		Informiert, dass keine Zeitmessung läuft. Zeigt einen Platzhalter für die Letzte runde, die Beste runde und die runde Nummer an.
Im Menüpunkt Nebenanzeige 4 (Case 4) Messung gestartet		Zeigt die Aktuelle Runde an, die Zeiter der Letzten Runde, die Zeit der Besten runde und die Aktuelle Runden Nummer.
Im Menüpunkt Nebenanzeige 4 (Case 4) Messung gestoppt		Informiert, dass keine Zeitmessung läuft. Zeigt die Zeiter der Letzten Runde, die Zeit der Besten runde und die letzte Runden Nummer.

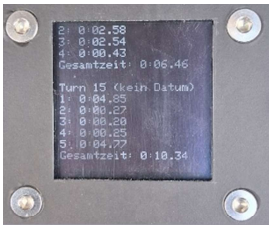
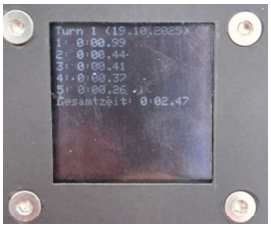
Im Menüpunkt Nebenanzeige 5 (Case 5) Messung ohne GPS empfang		Informiert, dass kein Datum vorhanden ist. Zeigt die Turn Nummer, die gemessenen Runden und die Gesamtzeit an.
Im Menüpunkt Nebenanzeige 5 (Case 5) Messung mit GPS empfang		Zeigt die Turn Nummer, das Datum, die gemessenen Runden und die Gesamtzeit an.
Im Menüpunkt Nebenanzeige 5 (Case 5) Lap-Time Taster 3 Sekunden gedrückt		Informiert, dass die Laptime Datei gelöscht wurde.
Im Menüpunkt Nebenanzeige 6 (Case 6)		Zeigt das Datum, die maximale Geschwindigkeit sowie die minimal und maximale höhe eines Tages.
Im Menüpunkt Nebenanzeige 6 (Case 6) Lap-Time Taster 3 Sekunden gedrückt		Informiert, dass die Extremwerte Datei gelöscht wurde.

Abbildung 78: Display Anzeige Kontrollen

5.18 Funktions Kontrollen Tasten

Bei dieser Funktionsprüfung wurden alle Taster in den verschiedenen Anzeigen ausprobiert und mit den Soll-Funktionen verglichen. Alle Taster haben genau so funktioniert, wie sie sollten und es gab keine Abweichungen.

Auch zusätzliche Tests, wie langes oder mehrfaches Drücken, wurden durchgeführt. Diese hatten keine weiteren Funktionen hervorgerufen.

5.18.1 Startanzeige

- Taster links:
  - Soll Funktion: Wenn mit Taster rechts 3 Sekunden gedrückt, Wechsel zu Hauptanzeige
  - Ist Funktion: Wenn mit Taster rechts 3 Sekunden gedrückt, Wechsel zu Hauptanzeige
- Taster rechts:
  - Soll Funktion: Wenn mit Taster links 3 Sekunden gedrückt, Wechsel zu Hauptanzeige
  - Ist Funktion: Wenn mit Taster links 3 Sekunden gedrückt, Wechsel zu Hauptanzeige
- Taster rauf:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Taster runter
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Lap-Time Taster:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion

5.18.2 Hauptanzeige:

- Taster links:
  - Soll Funktion: Wechselt zu Nebenanzeige 6
  - Ist Funktion: Wechselt zu Nebenanzeige 6
- Taster rechts:
  - Soll Funktion: Wechselt zu Nebenanzeige 1
  - Ist Funktion: Wechselt zu Nebenanzeige 1
- Taster rauf:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Taster runter
  - Soll Funktion: Wechselt zu Nebenanzeige 7
  - Ist Funktion: Wechselt zu Nebenanzeige 7
- Lap-Time Taster:
  - Soll Funktion: 3 Sekunden langer druck löscht Tracking Datei
  - Ist Funktion: 3 Sekunden langer druck löscht Tracking Datei

5.18.3 Nebenanzeige 1

- Taster links:
  - Soll Funktion: Wechselt zu Hauptanzeige
  - Ist Funktion: Wechselt zu Hauptanzeige
- Taster rechts:
  - Soll Funktion: Wechselt zu Nebenanzeige 2
  - Ist Funktion: Wechselt zu Nebenanzeige 2
- Taster rauf:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Taster runter
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Lap-Time Taster:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion

5.18.4 Nebenanzeige 2

- Taster links:
  - Soll Funktion: Wechselt zu Nebenanzeige 1
  - Ist Funktion: Wechselt zu Nebenanzeige 1
- Taster rechts:
  - Soll Funktion: Wechselt zu Nebenanzeige 3
  - Ist Funktion: Wechselt zu Nebenanzeige 3
- Taster rauf:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Taster runter
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Lap-Time Taster:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion

5.18.5 Nebenanzeige 3

- Taster links:
  - Soll Funktion: Wechselt zu Nebenanzeige 2
  - Ist Funktion: Wechselt zu Nebenanzeige 2
- Taster rechts:

  - Soll Funktion: Wechselt zu Nebenanzeige 4
  - Ist Funktion: Wechselt zu Nebenanzeige 4
- Taster rauf:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Taster runter
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Lap-Time Taster:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion

5.18.6 Nebenanzeige 4

- Taster links:
  - Soll Funktion: Wechselt zu Nebenanzeige 3
  - Ist Funktion: Wechselt zu Nebenanzeige 3
- Taster rechts:
  - Soll Funktion: Wechselt zu Nebenanzeige 5
  - Ist Funktion: Wechselt zu Nebenanzeige 5
- Taster rauf:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Taster runter
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Lap-Time Taster:
  - Soll Funktion:
    - Wenn Zeitmessung nicht aktiv: kurzer Druck startet die Zeitmessung
    - Wenn Zeitmessung aktiv: kurzer Druck setzt neue Rundenzeitmessung, langer Druck stoppt Zeitmessung
  - Ist Funktion:
    - Wenn Zeitmessung nicht aktiv: kurzer Druck startet die Zeitmessung
    - Wenn Zeitmessung aktiv: kurzer Druck setzt neue Rundenzeitmessung, langer Druck stoppt Zeitmessung

5.18.7 Nebenanzeige 5

- Taster links:
  - Soll Funktion: Wechselt zu Nebenanzeige 4
  - Ist Funktion: Wechselt zu Nebenanzeige 4
- Taster rechts:
  - Soll Funktion: Wechselt zu Nebenanzeige 6
  - Ist Funktion: Wechselt zu Nebenanzeige 6
- Taster rauf:
  - Soll Funktion: Scrollt 13 Zeilen nach oben
  - Ist Funktion: Scrollt 13 Zeilen nach oben
- Taster runter
  - Soll Funktion: Scrollt 13 Zeilen nach unten
  - Ist Funktion: Scrollt 13 Zeilen nach unten
- Lap-Time Taster:
  - Soll Funktion: 3 Sekunden langer druck löscht Laptime Datei
  - Ist Funktion: 3 Sekunden langer druck löscht Laptime Datei

5.18.8 Nebenanzeige 6

- Taster links:
  - Soll Funktion: Wechselt zu Nebenanzeige 5
  - Ist Funktion: Wechselt zu Nebenanzeige 5
- Taster rechts:
  - Soll Funktion: Wechselt zu Hauptanzeige
  - Ist Funktion: Wechselt zu Hauptanzeige
- Taster rauf:
  - Soll Funktion: Scrollt 13 Zeilen nach oben
  - Ist Funktion: Scrollt 13 Zeilen nach oben
- Taster runter
  - Soll Funktion: Scrollt 13 Zeilen nach unten
  - Ist Funktion: Scrollt 13 Zeilen nach unten
- Lap-Time Taster:
  - Soll Funktion: 3 Sekunden langer druck löscht Extremwerte Datei
  - Ist Funktion: 3 Sekunden langer druck löscht Extremwerte Datei

5.18.9 Nebenanzeige 7

- Taster links:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Taster rechts:
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Taster rauf:
  - Soll Funktion: Wechselt zu Hauptanzeige
  - Ist Funktion: Wechselt zu Hauptanzeige
- Taster runter
  - Soll Funktion: keine Funktion
  - Ist Funktion: keine Funktion
- Lap-Time Taster:
  - Soll Funktion:
    - Wenn WLAN deaktiviert: kurzer Druck schaltet das WLAN ein
    - Wenn WLAN aktiviert: kurzer druck schaltet das WLAN aus
  - Ist Funktion:
    - Wenn WLAN deaktiviert: kurzer Druck schaltet das WLAN ein
    - Wenn WLAN aktiviert: kurzer druck schaltet das WLAN aus

5.19 Funktion Kontrolle Tracking

Ich habe das Tracking während einer Probefahrt getestet. Zuvor habe ich die bestehende Tracking-Datei über die Hauptanzeige gelöscht, damit nur die aktuelle Fahrt aufgezeichnet wird. Die Route führte von Schlierbach nach Rickenbach, anschliessend nach Beromünster, weiter über die "Chommle" nach Eich, dann nach Sempach und entlang des Sees auf der Hauptstrasse über Schenkon und Geuensee nach Büron. So konnte ich die Genauigkeit des Trackings bei unterschiedlichen Geschwindigkeiten prüfen.

In Büron angekommen, habe ich das WLAN des Dashboards eingeschaltet und mein Smartphone damit verbunden. Anschliessend habe ich den kompletten Datensatz der Tracking Datei kopiert und in eine GPX-Datei umgewandelt. Diese GPX-Datei habe ich auf My Google Maps hochgeladen und das Tracking ausgewertet.

Dabei lässt sich erkennen, dass ich in Schlierbach das Dashboard eingeschaltet habe und Richtung Rickenbach fuhr. Auf dieser Strecke gibt es eine Kurve, die durch einen kurzen Waldabschnitt führt. Hier hat das GPS kurzzeitig den Kontakt verloren, vermutlich aufgrund der eingeschränkten Verbindung durch die Bäume. Beim Tankstopp in Beromünster ist zudem gut zu sehen, dass die Position bei stehendem Fahrzeug leicht verwackelt und ungenau ist. Laut einem Forum ist dies aber bei reinem GPS-Tracking normal.

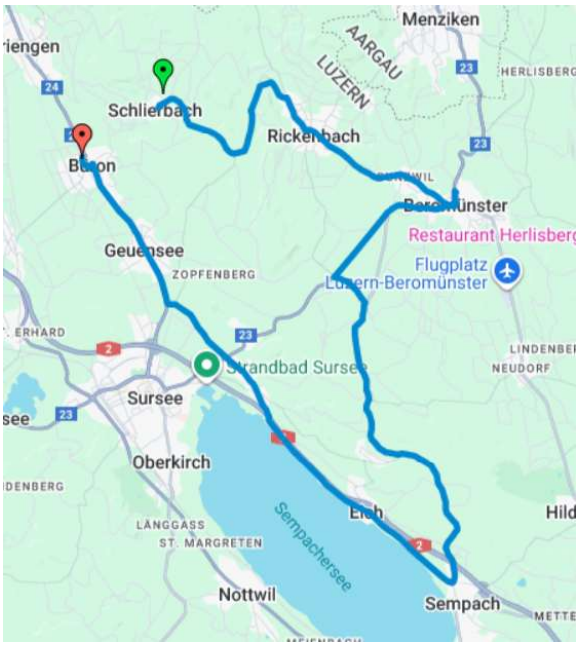


Abbildung 79: Tracking Test

Abgesehen davon funktioniert das Tracking sehr zuverlässig und ich bin mit dem Ergebnis sehr zufrieden.

6 Projektabschluss

Der Projektabschluss ist der letzte Schritt eines Projekts und beinhaltet die Überprüfung, Dokumentation und Bewertung, um sicherzustellen, dass die Projektziele erreicht wurden. Es umfasst auch die archivierte Aufbewahrung von Projektdokumenten und die Erstellung von Abschlussberichten. Der Projektabschluss dient dazu, Erfahrungen zu reflektieren, Lehren zu ziehen, den Projektfortschritt zu bewerten und sicherzustellen.

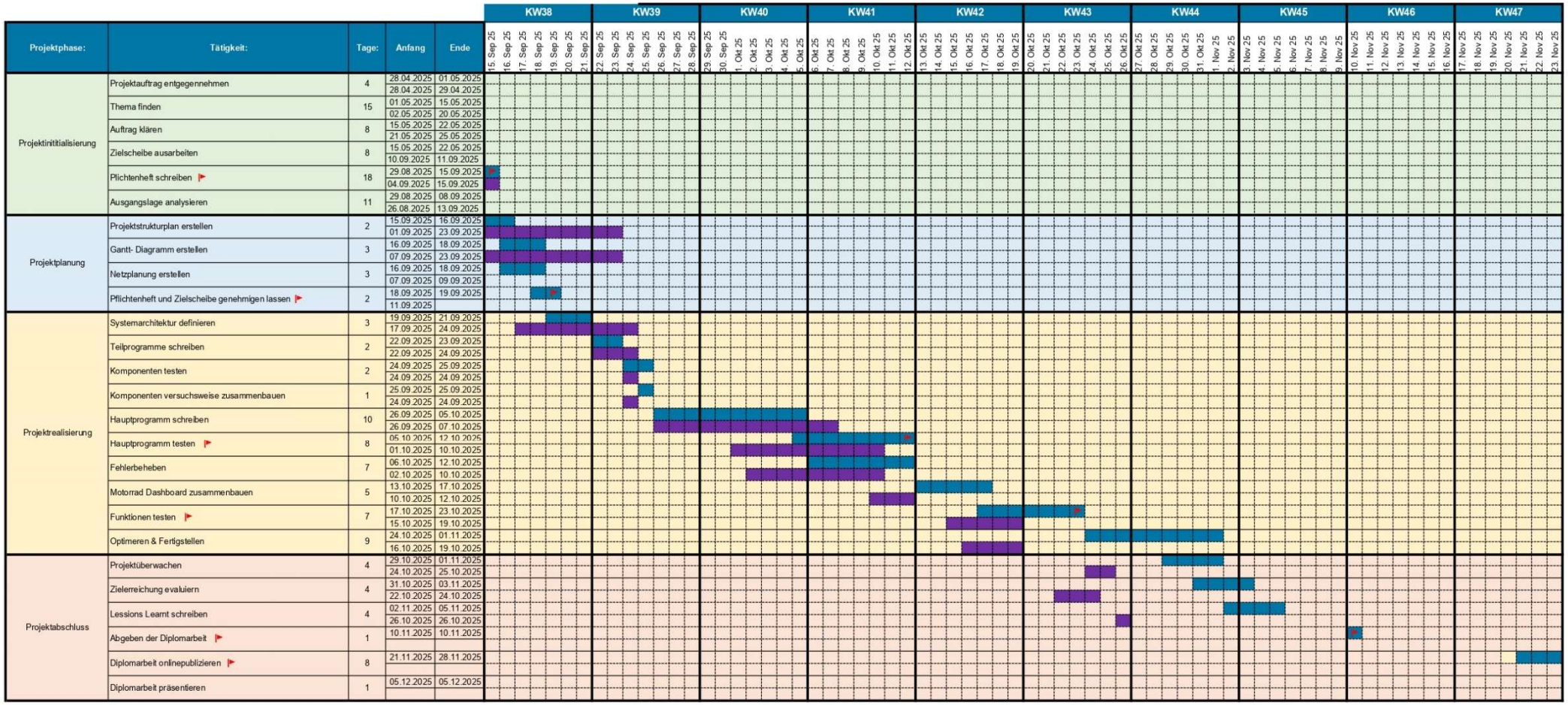
6.1 Projektüberwachung

Die Projektüberwachung ist ein entscheidender Schritt in der erfolgreichen Durchführung eines Projekts. Sie ermöglicht es, den Fortschritt, die Ressourcen und die Leistung des Projekts zu verfolgen und sicherzustellen, dass es auf Kurs bleibt. Abweichungen können dadurch erkannt und durch die Analyse begründet werden. Im folgenden Unterkapitel habe ich den Ist-Wert in der Farbe Violett im Gantt-Diagramm ergänzt.

6.1.1 Ablaufplanung – Controlling

Gantt-Diagramm Diplomarbeit Motorrad Dashboard

Legende: Soll: Ist: Meilenstein:



Projektphase:	Tätigkeit:	Tage:	Anfang	Ende
Projektinitialisierung	Projektauftrag entgegennehmen	4	28.04.2025	01.05.2025
			28.04.2025	29.04.2025
	Thema finden	15	01.05.2025	15.05.2025
			02.05.2025	20.05.2025
	Auftrag klären	8	15.05.2025	22.05.2025
			21.05.2025	25.05.2025
	Zielscheibe ausarbeiten	8	15.05.2025	22.05.2025
			10.09.2025	11.09.2025
	Pflichtenheft schreiben	18	29.08.2025	15.09.2025
			04.09.2025	15.09.2025
	Ausgangslage analysieren	11	29.08.2025	08.09.2025
			26.08.2025	13.09.2025
Projektplanung	Projektstrukturplan erstellen	2	15.09.2025	16.09.2025
			01.09.2025	23.09.2025
	Gantt- Diagramm erstellen	3	16.09.2025	18.09.2025
			07.09.2025	23.09.2025
	Netzplanung erstellen	3	16.09.2025	18.09.2025
			07.09.2025	09.09.2025
	Pflichtenheft und Zielscheibe genehmigen lassen	2	18.09.2025	19.09.2025
			11.09.2025	
Projektrealisierung	Systemarchitektur definieren	3	19.09.2025	21.09.2025
			17.09.2025	24.09.2025
	Teilprogramme schreiben	2	22.09.2025	23.09.2025
			22.09.2025	24.09.2025
	Komponenten testen	2	24.09.2025	25.09.2025
			24.09.2025	24.09.2025
	Komponenten versuchsweise zusammenbauen	1	25.09.2025	25.09.2025
			24.09.2025	24.09.2025
	Hauptprogramm schreiben	10	26.09.2025	05.10.2025
			26.09.2025	07.10.2025
	Hauptprogramm testen	8	05.10.2025	17.10.2025
			01.10.2025	10.10.2025
	Fehlerbeheben	7	06.10.2025	12.10.2025
			02.10.2025	10.10.2025
	Motorrad Dashboard zusammenbauen	5	13.10.2025	17.10.2025
			10.10.2025	12.10.2025
	Funktionen testen	7	17.10.2025	23.10.2025
			15.10.2025	19.10.2025
	Optimieren & Fertigstellen	9	24.10.2025	01.11.2025
			16.10.2025	19.10.2025
Projektabschluss	Projektüberwachen	4	29.10.2025	01.11.2025
			24.10.2025	25.10.2025
	Zielerreichung evaluiern	4	31.10.2025	03.11.2025
			22.10.2025	24.10.2025
	Lessions Learnt schreiben	4	02.11.2025	05.11.2025
			26.10.2025	26.10.2025
	Abgeben der Diplomarbeit	1	10.11.2025	10.11.2025
	Diplomarbeit onlinepublizieren	8	21.11.2025	28.11.2025
	Diplomarbeit präsentieren	1	05.12.2025	05.12.2025

Abbildung 80: Gantt Diagramm Controlling

6.1.2 Fazit Ablaufplanung

Bei der Analyse fällt auf, dass die obersten vier Arbeitspakete zeitlich vor dem offiziellen Startdatum der Diplomarbeit liegen. Die Aufgaben Projektauftrag entgegennehmen, Thema finden, Auftrag klären und Zielscheibe ausarbeiten fanden in der Vorbereitungsphase statt und sind daher nicht in die eigentliche Projektdauer eingeflossen. Aus diesem Grund wurde die Zeitleiste des Diagramms nicht über diesen Zeitraum hinaus verlängert.

Die Genehmigung des Pflichtenhefts und der Zielscheibe dauerte deutlich länger als ursprünglich geplant. Der Grund dafür war, dass mein Projektlehrer Urs Gloggner mit den ersten Ausarbeitungen noch nicht vollständig zufrieden war. Daher überarbeitete ich beide Dokumente mehrfach und liess sie jeweils erneut überprüfen, was zu einer Verzögerung führte.

Auch bei der Definition der Systemarchitektur benötigte ich mehr Zeit als vorgesehen. Während dieser Phase entstanden immer wieder neue Ideen, wie das Menü übersichtlicher und benutzerfreundlicher gestaltet werden könnte, was zusätzliche Anpassungen erforderlich machte.

Während der Programmierung des Hauptprogramms testete ich meinen Code fortlaufend parallel am Laboraufbau. Dadurch konnte ich unmittelbar überprüfen, ob der Code funktionierte, welche Teile noch angepasst werden mussten und welche bereits zuverlässig liefen. Obwohl dieser Ansatz die Programmierphase etwas verlängerte, hatte er den Vorteil, dass das Testen und die Fehlerbehebung bereits frühzeitig abgeschlossen werden konnten. Somit konnte der Zusammenbau des Dashboards sowie dessen abschliessende Tests früher als geplant begonnen und abgeschlossen werden.

Dieser zeitliche Vorteil erwies sich als optimal, da ich mich ab diesem Zeitpunkt verstärkt auf die Dokumentation konzentrieren konnte und der grösste Arbeitsdruck nachliess. Dadurch war es mir möglich, bereits frühzeitig mit dem Projektabschluss zu beginnen und die gesamte Diplomarbeit vor der Abgabe noch in Ruhe kontrollieren zu lassen.

Die untersten drei Arbeitspakete sind im Diagramm nicht mit einem Ist-Zustand versehen, da sie nach dem Abgabetermin der Arbeit liegen. Zum Zeitpunkt der Abgabe konnte daher keine Bewertung des tatsächlichen Fortschritts vorgenommen werden.

6.2 Evaluation der Zielerreichung

Die Evaluation der Zielerreichung ist ein entscheidender Schritt, um den Erfolg meines Projekts zu bewerten. Dabei analysiere ich, inwieweit die definierten Ziele umgesetzt wurden und ob das Endprodukt den gestellten Anforderungen entspricht. Dies funktioniert am einfachsten, indem man Endergebnisse und Erfolgskriterien gegenüberstellt.

Endergebnisse	Erfolgskriterien
Das Motorrad Dashboard ist wetterfest, gut am Lenker befestigt und die Bedienung des Menüs ist über 5 Taster möglich.	Die Bedienung der Taster funktioniert mit Motorradhandschuhen, das Gehäuse ist staub- und spritzwassergeschützt und stabil am Lenker fixiert, sodass es sich während der Fahrt nicht lockert.
Das Motorrad Dashboard blieb während einer einstündigen Probefahrt stabil am Lenker befestigt und verrutschte nicht. Die Taster liessen sich problemlos auch mit Motorradhandschuhen bedienen. Zudem hat das Dashboard einen Wassertest erfolgreich bestanden, wodurch die Wetterfestigkeit des Gehäuses bestätigt wurde.	
Das Motorrad Dashboard wird über die 12V-Motorradbatterie gespeist und ist mit Verpolungs- und Überlastschutz ausgestattet.	Das Motorrad Dashboard funktioniert an 12V, ist mit einer 2A-Sicherung abgesichert und besitzt eine Crowbar-Diode, die bei Falschpolung die Sicherung auslöst.
Das Messgerät zeigt eine Batteriespannung von 13,35 V an. Zur Absicherung wurde eine 2 Ampere Flachsicherung eingebaut, und zusätzlich wurde eine Shottky-Diode parallel zum Hauptschalter verlötet, um die Schaltung vor möglichen Falschpollungen zu schützen.	
Das Motorrad Dashboard hat ein Ansichtsmenü, in dem man die GPS-Geschwindigkeit ablesen kann.	Die GPS-Geschwindigkeit hat zu einer Referenzmessung nicht mehr als ± 3km/h Abweichung.
Die Geschwindigkeitsanzeige des Dashboards stimmt mit der Geschwindigkeit der App GPSTest überein.	
Das Motorrad Dashboard hat ein Ansichtsmenü, in dem man die Höhen anzeigen kann.	Die Höhe ist gegenüber einer Referenzmessung auf ± 5m genau.
Die Höhenanzeige des Dashboards weist eine Abweichung von etwa 3–4 Metern auf und reagiert etwas langsamer als die Höhenanzeige der App GPSTest.	

Abbildung 81: Evaluation Zielerreichung 1

Das Motorrad Dashboard hat ein Ansichtsmenü, in dem man die Himmelsrichtung per Kompass und per Gradzahl bestimmen kann.	Die Anzeige stimmt mit einem Referenzkompass überein und zeigt eine maximale Abweichung von ±5°.
Die Kompassanzeige des Dashboards stimmt mit der Gradanzeige der App GPSTest überein.	
Das Motorrad Dashboard hat ein Ansichtsmenü, in dem man die Rundenzeit messen kann.	Es werden die aktuelle, die Letze und die Schnellste Runde von einem Turn angezeigt.
Die Funktion für die Messung von Runden wurde entwickelt. Es werden die aktuelle, die letzte und die schnellste Runde eines Turns angezeigt.	
Das Motorrad Dashboard hat ein Ansichtsmenü, in dem man die Gespeicherten Rundenzeiten einsehen kann.	Das Speicherlayout der Rundenzeiten werde von 80% der Personen als benutzerfreundlich angesehen.
Das Dashboard verfügt über eine Speicherfunktion, in der die aufgezeichneten Rundenzeiten gespeichert und anschliessend eingesehen werden können. Von 12 befragten Personen gaben 12 an, dass das Speichermenü übersichtlich gestaltet ist. Dies entspricht 100% der Befragten.	
Das Motorrad Dashboard hat ein Ansichtsmenü, in dem man die Maximale und minimale Höhe und die Höchstgeschwindigkeit per tag einsehen kann.	Das Speicherlayout der Extremwerte werde von 80% der Personen als benutzerfreundlich angesehen.
Das Dashboard verfügt über eine Speicherfunktion, in der die aufgezeichneten Extremwerte gespeichert und anschliessend eingesehen werden können. Von 12 befragten Personen gaben 12 an, dass das Speichermenü übersichtlich gestaltet ist. Dies entspricht 100% der Befragten.	

Abbildung 82: Evaluation Zielerreichung 2

Das Motorrad Dashboard hat ein Ansichtsmenü, in dem man die Gespeicherten Daten per WLAN einsehen kann.	Die gespeicherten Daten können über das WLAN auf einem Smartphone eingesehen werden.
Das Dashboard bietet die Möglichkeit, das WLAN einzuschalten, um es mit einem Smartphone zu verbinden. So können Rundenzeiten, Extremwerte und das GPS-Tracking direkt über das Mobilgerät eingesehen werden.	
Das Programm ist mit einem Zustandsdiagramm visuell dargestellt.	Es ist ein Zustandsdiagramm erstellt, welches die Logik, Menüs und Anzeigezustände des Programms abbildet.
Es wurde ein Zustandsdiagramm mit dem Programm draw.io erstellt. Darin sieht man die verschiedenen zustände, Menüs und Funktionen des Motorrad Dashboards	

Abbildung 83: Evaluation Zielerreichung 3

6.3 Umfrage

Ich habe insgesamt 12 Personen unterschiedlichen Alters mein Motorrad Dashboard testen lassen. Nach einer kurzen Einführung konnten sie das Dashboard selbstständig ausprobieren. Anschliessend bat ich alle Testpersonen, eine Umfrage auszufüllen, die ich über Microsoft Forms erstellt habe, um so Feedback und Bewertungen für meine Diplomarbeit zu erhalten.

6.3.1 Frage 1: Alter der Testperson

Abbildung 84: Alter

Wie man aus der Grafik heraus erkennen kann, habe ich eine Person unter 20 Jahren, vier Personen zwischen 20 und 29 Jahren, jeweils zwei Personen zwischen 30 und 39 sowie 40 und 49 Jahren und drei Personen im Alter von 50 Jahren oder älter zu meinem Motorrad Dashboard befragt.

6.3.2 Frage 2: Wie ansprechend ist das Design des Dashboards?

Abbildung 85: Bewertung Design

In dieser Grafik ist zu erkennen, dass fünf Personen mit dem Design meines Dashboards sehr zufrieden waren, während sieben Personen mehrheitlich zufrieden waren.

6.3.3 Frage 3: Wie gut kann man das Dashboard bedienen?

Abbildung 86: Bewertung Bedienung

Für die Bedienung des Dashboards mit den fünf Tastern gaben zehn Personen an, dass es sehr einfach war. Eine Person empfand es als einfach und jemand hatte ein wenig Mühe.

6.3.4 Frage 4: Wie übersichtlich sind die Speicherlayouts?

Abbildung 87: Bewertung Anzeigestruktur

Diese Grafik zeigt, dass das Speicherlayout des Dashboards von den Befragten insgesamt positiv bewertet wurde. Die Hälfte der Befragten gab an, damit sehr zufrieden zu sein, während die andere Hälfte sie als zufriedenstellend empfand.

6.3.5 Frage 5: Ist die Reaktionsgeschwindigkeit zufriedenstellend?

Abbildung 88: Bewertung Reaktionszeit

Aus dieser Frage geht hervor, dass die Reaktionszeit des Dashboards von den meisten Teilnehmern sehr positiv bewertet wurde. Elf von zwölf befragten Personen empfanden die Geschwindigkeit als sehr gut und problemlos nutzbar. Lediglich eine Person bewertete die Reaktionszeit als ausreichend.

6.3.6 Frage 6: Welche Funktionen sollte ich ergänzen?

Bei dieser Frage hatten die Testpersonen die Möglichkeit, zusätzliche Funktionen zu nennen, die sie sich für das Dashboard wünschen würden. Folgende Vorschläge wurden gemacht:

- Eine Zeitmessung für die Beschleunigung von 0 auf 100 km/h
- Einstellbare Displayhelligkeit
- Anzeige des Neigungswinkels
- Anzeige der aktuellen Uhrzeit auf dem Dashboard
- Eine Kartenfunktion ähnlich wie bei Google Maps
- Ein Regenradar

6.3.7 Frage 7: Gibt es störende Funktionen?

Alle Testpersonen gaben an, dass sie keine Funktion auf dem Motorrad Dashboard als störend oder überflüssig empfanden. Jede der vorhandenen Anzeigen und Bedienelemente wurde als sinnvoll und gut platziert wahrgenommen.

6.3.8 Frage 8: Gesamteindruck des Motorrad Dashboards

Abbildung 89: Bewertung Gesamteindruck

In dieser Grafik ist zu erkennen, dass neun Personen einen sehr zufriedenen Eindruck vom Dashboard hatten, während drei Personen es als zufriedenstellend empfanden. Insgesamt lässt sich daraus schliessen, dass das Dashboard von den Testpersonen als übersichtlich und benutzerfreundlich wahrgenommen wurde.

6.3.9 Frage 9: Anmerkungen und Kritik

Bei dieser Frage hatten die Testpersonen die Möglichkeit, Anmerkungen und Kritik zu meiner Diplomarbeit zu hinterlassen. Folgende Rückmeldungen wurden erfasst:

- Es ist ein tolles Endergebnis, das sich wirklich sehen lassen kann
- Die Folierung an den Ecken könnte etwas sauberer sein
- Schönes Design
- Grössere Schrift wäre wünschenswert.

6.4 Reflexion

Im Laufe meines Projekts gab es Höhen und Tiefen, ich habe Herausforderungen gemeistert und Erfolge gefeiert. Jeder Schritt, den ich unternommen habe, war von Bedeutung und hat mir wertvolle Erkenntnisse vermittelt. Das Kapitel "Refexion" lädt mich dazu ein, einen Blick zurückzuwerfen und die wertvollen Lehren zu reflektieren, die ich aus dieser Reise gezogen haben.

6.4.1 Weg zum Ziel

Durch die Arbeit an meinem Motorrad Dashboard habe ich sehr viel gelernt und konnte auch zahlreiche Erfahrungen aus früheren Projekten einbringen. Besonders in den Bereichen strukturierte Planung, Fehleranalyse und systematische Umsetzung konnte ich mein bisheriges Wissen anwenden und erweitern.

Der Start in das Projekt war anspruchsvoller, als ich zunächst erwartet hatte. Zwar hatte ich viele Ideen im Kopf, doch keine davon überzeugte mich wirklich, bis ich begann, mich intensiver mit der konkreten Umsetzung und Strukturierung zu beschäftigen. Erst durch eine klare Planung und das Aufstellen eines realistischen Zeitplans bekam das Projekt die nötige Richtung.

Besonders lehrreich war für mich die Phase der Inbetriebnahme. Ich hatte anfangs die Hoffnung, dass der Aufbau nach Plan direkt funktionieren würde, doch die Realität sah anders aus. Immer wieder traten unvorhergesehene Probleme auf, etwa mit dem SD-Modul oder dem GPS-Empfänger, welcher ohne zu wissen weshalb nicht mehr funktionierte. Anstatt mich über Fehler zu ärgern, begann ich, sie als Teil des Lernprozesses zu sehen. Durch gezieltes Testen, Prüfen der Schaltung und schrittweises Anpassen des Programms konnte ich jedes Problem nach und nach beheben und das Dashboard schliesslich zuverlässig in Betrieb nehmen.

Auch der technische Lernzuwachs war enorm. Ich konnte mein Wissen in Elektronik, Mikrocontroller-Programmierung (C++ in Arduino IDE) und Sensorintegration deutlich vertiefen. Besonders spannend war es, die verschiedenen Module, wie GPS, OLED-Display, Magnetometer und Taster, so zu kombinieren, dass sie fehlerfrei zusammenarbeiten. Dadurch habe ich nicht nur gelernt, komplexe Schaltungen aufzubauen, sondern auch, Software und Hardware gezielt aufeinander abzustimmen.

Insgesamt hat mir das Projekt gezeigt, wie wichtig Planung, Ausdauer und Genauigkeit sind. Auch wenn es zwischendurch Rückschläge gab, war es sehr motivierend zu sehen, wie aus einer Idee ein voll funktionsfähiges Produkt entstand. Besonders stolz bin ich darauf, dass das Dashboard am Ende alle geplanten Funktionen erfüllt und sich im Praxistest bewährt hat. Die Arbeit hat mir nicht nur im technischen Bereich viel gebracht, sondern auch meine Fähigkeiten im Projektmanagement und in der Problemlösung deutlich verbessert.

6.4.2 Lessons learned

1. **Klare Planung ist entscheidend:** Eine detaillierte Projektstruktur und ein realistischer Zeitplan erleichtern die Umsetzung und helfen, den Überblick zu behalten. Ohne klare Planung können selbst einfache Projekte schnell unübersichtlich werden.

2. **Systematisches Vorgehen bei Problemen:** Fehler treten immer wieder auf, sei es bei der Hardware oder Software. Wichtig ist, geduldig zu bleiben, Probleme Schritt für Schritt zu analysieren und systematisch zu testen, anstatt frustriert aufzugeben.

3. **Technisches Wissen gezielt einsetzen:** Kenntnisse in Elektronik, Mikrocontroller-Programmierung und Sensorintegration sind essenziell. Durch die Kombination von Software und Hardware lassen sich komplexe Systeme erfolgreich aufbauen.

4. **Flexibilität und Anpassungsfähigkeit:** Unerwartete Probleme wie Ausfälle von Modulen (SD, GPS) erfordern Anpassungen im Aufbau oder in der Software. Flexibilität ermöglicht, trotzdem Fortschritte zu machen.

5. **Teamarbeit und Kommunikation:** Auch wenn das Projekt überwiegend eigenständig durchgeführt wurde, ist der Austausch von Erfahrungen, Ideen und Lösungsansätzen mit anderen hilfreich, um Probleme schneller zu erkennen und zu lösen.

6. **Geduld und Ausdauer zahlen sich aus:** Kleine Rückschläge sind normal, langfristiger Erfolg entsteht durch konsequentes Arbeiten, Ausprobieren und schrittweise Verbesserung.

6.5 Verdankung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich während der Erstellung meiner Diplomarbeit unterstützt haben.

Mein besonderer Dank gilt meinem Projektlehrer Urs Gloggner für die fachliche Betreuung und die konstruktiven Rückmeldungen. Ebenso danke ich Andreas Holzer für die Übernahme der Rolle als Fachexperte.

Ein herzliches Dankeschön gilt auch allen Testpersonen, die mein Motorrad Dashboard getestet und bewertet haben. Ebenso danke ich meiner Freundin Tamira Arnold für das sorgfältige Gegenlesen und Korrigieren meiner Dokumentation.

Mein besonderer Dank gilt ausserdem meiner Familie und meinen Freunden, die mir während der intensiven Arbeitsphase stets Verständnis entgegengebracht und mich in jeder Hinsicht unterstützt haben.

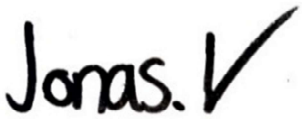
Ohne diese Unterstützung wäre das erfolgreiche Gelingen dieser Diplomarbeit nicht möglich gewesen.

7 Redlichkeitserklärung

Der Verfasser bestätigt mit seiner Unterschrift, dass die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als die angegebenen Hilfsmittel erstellt wurde.

Die aus fremden Quellen (einschliesslich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit ist in gleicher oder ähnlicher Form noch nicht vorgelegt worden.



Luzern, 9. November 2025

Jonas Kuster

8 Abbildungsverzeichnis

9 Literatur- und Quellenverzeichnis

adafruit. (2025). *https://www.adafruit.com/*.

Amazon. (2025). *https://www.amazon.de/*.

Arduino_Forum. (kein Datum). *https://forum.arduino.cc/*.

Arduino_IDE. (2025). *https://www.arduino.cc/en/software/*.

Arduino-Pico. (2025). *https://arduino-pico.readthedocs.io/en/latest/*.

Backyard_Racing. (2025). *https://www.backyard-racing.ch/*.

bastelgarage. (2025). *https://www.bastelgarage.ch/*.

Carfoil. (2025). *https://www.carfoil.ch/de/*.

Chip. (2025). *https://www.chip.de/*.

Conrad. (2025). *https://www.conrad.ch*.

Digitec. (2025). *https://www.digitec.ch/de*.

Distrelec. (2025). *https://www.distrelec.ch/de/*.

Drawio. (2025). *https://www.drawio.com/*.

Elektro_Material. (2025). *https://www.elektro-material.ch/de/shop*.

Forms. (2025). *https://forms.office.com/Pages/DesignPagev2.aspx*.

GitHub. (2025). *https://github.com/*.

Google. (2025). *https://www.google.com/*.

Google_My_Maps. (2025). *https://www.google.com/maps/d/u/0/*.

GPSvisualizer. (2025). *https://www.gpsvisualizer.com/convert_input*.

Gruppe 4. (2023). *Semeseterarbeit Projekt Insekten-Sterben.* TEKO Luzeren: J,A,D,K.

Holzer, A. (2025). Div. Dokumentationen. TEKO Luzern, TEAMS.

Holzer, A. (2025). Div. Programme. TEKO Luzern, TEAMS.

ilovepdf. (2025). *https://www.ilovepdf.com/pdf_to_jpg*.

Jumbo. (2025). *https://www.jumbo.ch/de*.

loxforum. (2025). *https://www.loxforum.com/forum/german*.

OpenAI. (2025). *ChatGPT.* https://chatgpt.com/.

Polo_Motorrad. (2025). *https://www.polo-motorrad.com/de-ch/*.

Praktische_Elektronik. (2025). *https://praktische-elektronik.dr-k.de/*.

Quad_Lock. (2025). *https://www.quadlockcase.eu/*.

Räber, J. (2023). *Leitfaden (von Skript Projektmanagement) .* TEKO Luzern.

Raspberry_Pi. (2025). *https://forums.raspberrypi.com/*.

Reddit. (2025). *https://www.reddit.com/*.

Schweizerische_Fachschule_TEKO. (2025). *https://www.teko.ch/*.

SpellBoy. (2025). *https://www.spellboy.com/rechtschreibpruefung/*.

Starlane_Preformance_Electronics. (2025). *https://www.starlane-shop.de/*.

TinyCAD. (2025). *https://www.tinycad.net/*.

Wikipedia. (2025). *https://de.wikipedia.org/*.

XLMoto. (2025). *https://www.xlmoto.ch/*.

YouTube. (2025). *https://www.youtube.com/*.

10 Beilagenverzeichnis

1	Mind-Map	→ Ideensammlung für die Projekteingabe
2	Pflichtenheft	→ Grundlage für die Entwicklung und Umsetzung
3	Zielscheibe	→ Auflistung von Erfolgskriterien und Endergebnissen
4	Netzplan	→ Aufzeigen des Kritischen Pfads
5	Projektstrukturplan	→ Auflistung der Arbeitspakete nach Projektphase
6	Gantt Diagramm	→ Gantt Diagramm inkl. Controlling
7	Grobschema	→ Belegungsplan der GPIOs des Raspberry Pi Pico
8	Schaltplan	→ Kompletter Verdrahtungsplan
9	State Maschine	→ Zustandsdiagramm der Logik
10	Programm	→ Software zum Motorrad Dashboard inkl. Bibliotheken
11	Plan Gehäuse	→ Vermasster ausschnittplan des Gehäuses
12	Gebrauchsanleitung	→ Anleitung zur Handhabung des Motorrad Dashboards
13	Statusberichte	→ Statusberichte von KW38 bis KW45

Diplomarbreit

- Mottorrad
  - Lap- Time
  - Stoppuhr
  - Temperatur
  - Höhenmesser
  - Neigungswinkel
  - Reifendruck
  - Beschleunigung
  - Log- Daten
  - G- Kraft
  - Kompass
  - Geschwindigkeit
- Zuhause
  - Smarthome anbindungen
  - Lichtsteuernung
  - Boilersteuernung
  - Tabletsteuerung
  - Solaranbindung
  - Wetterstation
- Robotik
  - Kaugummi Automat
  - Getränkestation
  - Wurf Roboter
  - Flüssigkeitenspender
  - follow- Me
  - Linien Roboter
  - Erkennung per Kamera
- Geschäft
  - Solar
  - Automation
  - Trisaprojekt
  - Spitalprojekt
- Gadget
  - Profieltiefenmesser
  - Musik- Teslaspule
  - Fernbedienung
  - Wecker

Pflichtenheft für die Diplomarbeit

Inhaltsverzeichnis

1. Einleitung

Auf dem Markt existieren bereits verschiedene Motorrad-Dashboards und GPS-Lap-Timer. Viele dieser Systeme sind jedoch in ihrer Funktionalität eingeschränkt, z. B. limitierte Rundenzahl beim Lap-Timer, ungenaue Tachoanzeigen oder fehlende Zusatzfunktionen wie Höhenmessung und Kompass.

Idee und Abgrenzung

Das geplante Motorrad Dashboard soll zusätzliche Funktionen vereinen, die in bestehenden Lösungen nicht oder nur teilweise verfügbar sind:
- GPS-basierte Geschwindigkeitsmessung.
- Speicherung von Geschwindigkeiten und Höhenwerten auf einer SD-Karte.
- Anzeige von Himmelsrichtungen mittels eines digitalen Kompasses.
- Lap-Time-Funktion, die die einzelnen Rundenzeiten anzeigt, sowie die Zeit vom ganzen Run.
- Speicherung von Maximalgeschwindigkeiten und Höhen.
- Datenübertragung und -analyse per WLAN

Das System unterscheidet sich somit von bestehenden Lösungen durch seine Vielseitigkeit, erweiterte Speichermöglichkeiten und die einfache Integration am Motorradlenker.

2. Fachexperte

Vorname: Andreas
Name: Holzer
Beruf: Dozent TEKO
Rolle/Funktion: Dozent Robotik & SPS
Adresse: Winkelriedstrasse 64, 6003 Luzern
E-Mail: andreas.holzer@edu.teko.ch
Telefon: 079 428 00 70

3. Ziele

Richtziel

Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.

Endergebnisse

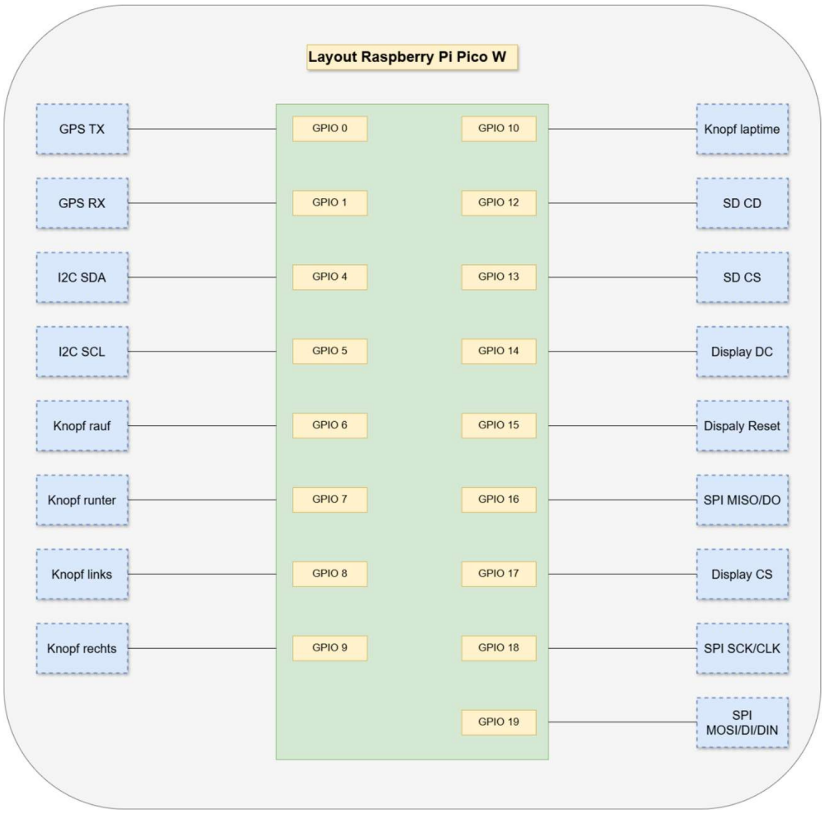
1. GPS-Geschwindigkeit messen und auf der SD-Karte speichern.
2. Höhenanzeige + Speicherung der min./max. Höhen.
3. Kompassanzeige (N, NO, O, SO, S, SW, W, NW) mit Gradzahl.
4. Lap-Time Funktion: Messung, Anzeige und Speicherung.
5. WLAN-Zugriff auf gespeicherte Daten.
6. Das Motorrad Dashboard ist wetterfest und die Bedienung des Menüs ist über 4 Taster möglich.
7. Das Motorrad Dashboard ist am Lenkrad befestigt.

Erfolgskriterien

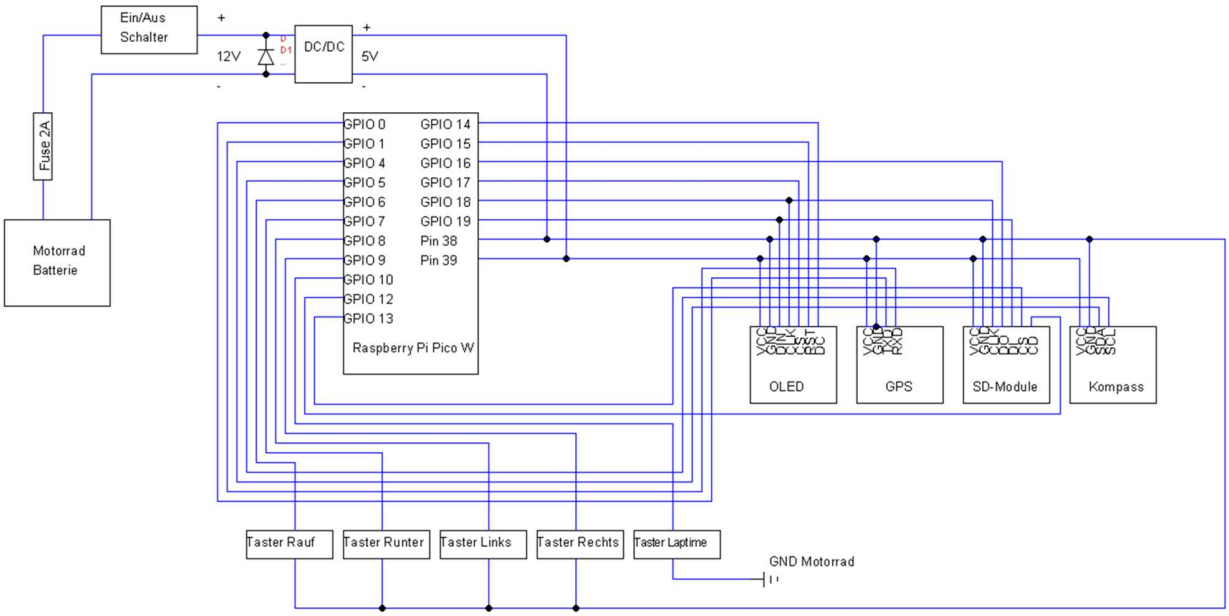
1. Abweichung zwischen GPS-Geschwindigkeit und Referenzmessung beträgt max. ±3 km/h und sind in lesbarer Textstruktur auf der SD-Karte gespeichert.
2. Anzeige zeigt die aktuelle Höhe in Metern über Meeresspiegel und speichert während einem Tag den niedrigsten und höchsten Wert.
3. Anzeige wechselt korrekt zwischen den 8 Haupt-Himmelsrichtungen. Genauigkeit der Gradzahl beträgt ±5°.
4. Daten sind im Lap-Time-Menü abrufbar und nach Fahrtende auf der SD-Karte verfügbar.
5. WLAN-Hotspot lässt sich über das Menü starten und zeigt die Gespeicherten Daten an.
6. Bedienung der Taster funktioniert mit Motorradhandschuhen. Gehäuse ist staub- und spritzwassergeschützt.
7. Das Dashboard ist stabil am Lenker fixiert, sodass es sich während der Fahrt nicht lockert.

Schematische Skizzen

Geplantes Layout:



Geplanter Schaltplan:



Geplante Ansteuerung:

Was	Ansteuerung	Pins
GPS- Modul	UART- Bus	TXD, RXD
Display	SPI- Bus	CLK, DIN, CS, RST
Magnetometer	I2C- Bus	SDA, SCL
SD- Module Adafruit	SPI- Bus	CLK, DO, DI, CS, CD
Taster	GPIO	Input

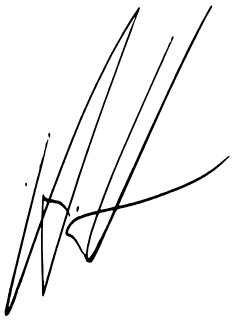
Weitere Angaben

- Technische Umsetzung mit Raspberry Pi Pico W, GPS, Magnetometer, Display und SD-Kartenmodul.
- Spannungsversorgung über 12V vom Motorrad mit Konverter auf 5V.
- Benutzerfreundliches Bedienkonzept mit 8 Menü Ansichten.
- Gehäuse mit Display und 4 bedienknöpfe. Lap- Time Taster in der Nähe des linken Griffs.

4. Unterschrift

Diplomand: Jonas Kuster Jonas. K Datum: 04.09.2025

Fachexperte: Andreas Holzer Datum: 11.9.2025



Richtziel:
Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.

1. Das Motorrad Dashboard ist wetterfest, gut am Lenker befestigt und die Bedienung des Menüs ist über 5 Taster möglich.
2. Das Motorrad Dashboard wird über die 12V-Motorradbatterie gespeist und ist mit Verpolungs- und Überlastschutz ausgestattet.
3. Das Motorrad Dashboard hat ein Ansichtsmenü, in dem man die:
   3.1. GPS-Geschwindigkeit ablesen kann.
   3.2. Höhen anzeigen kann.
   3.3. Himmelsrichtung per Kompass und per Gradzahl bestimmen kann.
   3.4. Rundenzeit messen kann.
   3.5. Gespeicherten Rundenzeiten einsehen kann.
   3.6. Maximale und minimale Höhe und die Höchstgeschwindigkeit per tag einsehen kann.
   3.7. Gespeicherten Daten per WLAN einsehen kann.
4. Das Programm ist mit einem Zustandsdiagramm visuell dargestellt.

- TEKO
- Jonas Kuster

Endergebnisse

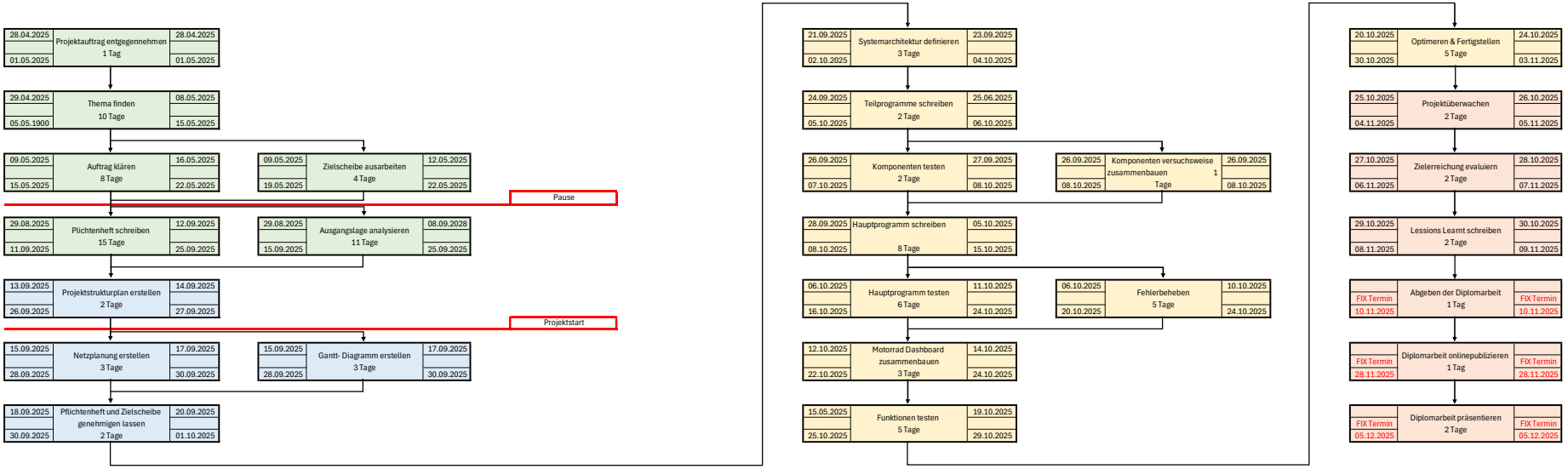
Kunde

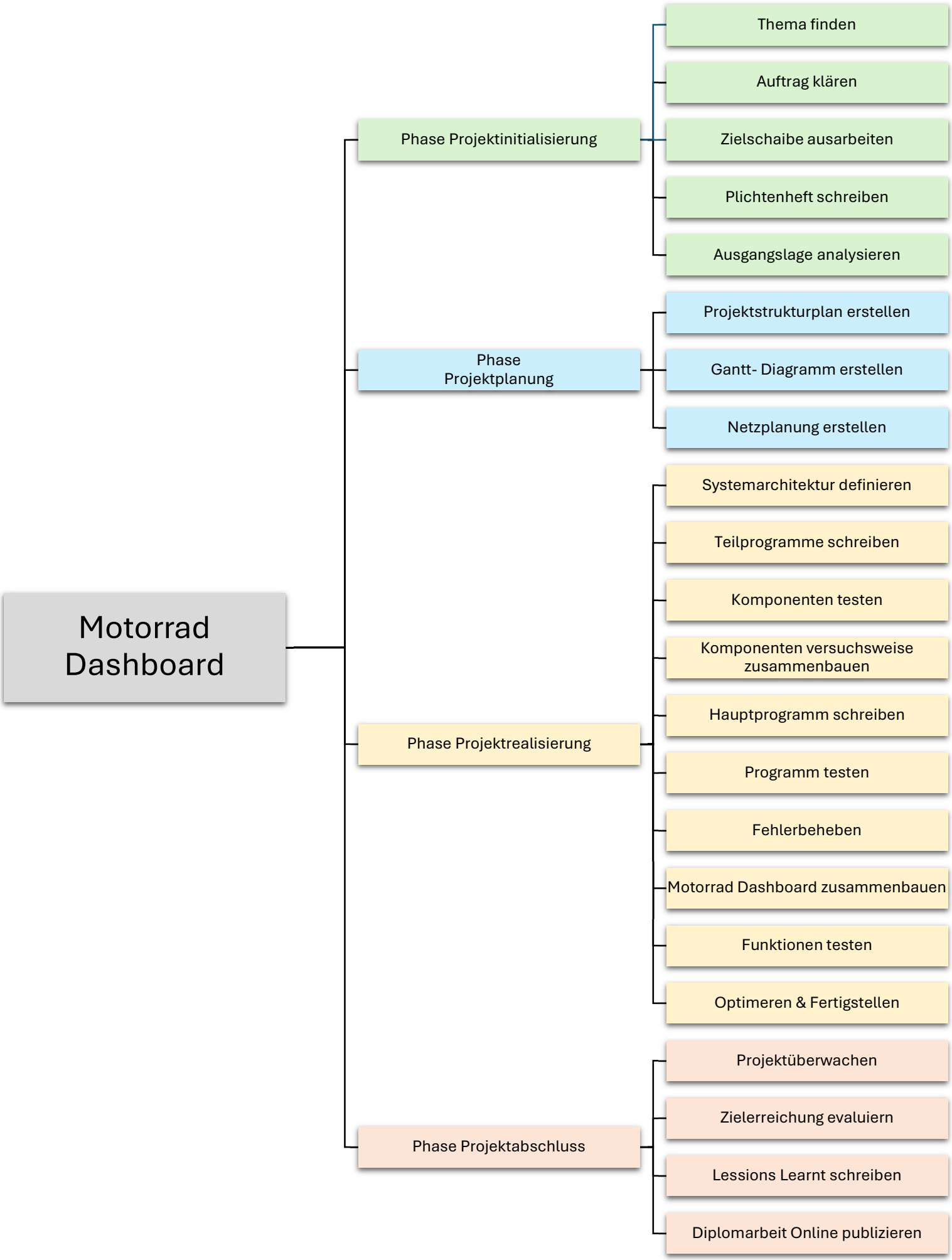
Sinn und Zweck

Die Arbeit dient dazu, Fahrdaten zu sammeln und auszuwerten sowie die Orientierung durch zusätzliche Informationen wie Kompass und Höhenmessung zu verbessern. So entsteht ein praktisches Tool für sportliche Analysen und sichere Navigation auf Touren.

Erfolgskriterien

1. Die Bedienung der Taster funktioniert mit Motorradhandschuhen, das Gehäuse ist staub- und spritzwassergeschützt und stabil am Lenker fixiert, sodass es sich während der Fahrt nicht lockert.
2. Das Motorrad Dashboard funktioniert an 12V, ist mit einer 2A-Sicherung abgesichert und besitzt eine Crowbar-Diode, die bei Falschpolung die Sicherung auslöst.
3. Das Ansichtsmenü des Motorrad Dashboards erfüllt:
   3.1. Die GPS-Geschwindigkeit hat zu einer Referenzmessung nicht mehr als ± 3km/h Abweichung.
   3.2. Die Höhe ist gegenüber einer Referenzmessung auf ± 5m genau.
   3.3. Die Anzeige stimmt mit einem Referenzkompass überein und zeigt eine maximale Abweichung von ±5°.
   3.4. Es werden die aktuelle, die Letze und die Schnellste Runde von einem Turn angezeigt.
   3.5. Das Speicherlayout der Rundenzeiten werde von 80% der Personen als benutzerfreundlich angesehen.
   3.6. Das Speicherlayout der Extremwerte werde von 80% der Personen als benutzerfreundlich angesehen.
   3.7. Die gespeicherten Daten können über das WLAN auf einem Smartphone eingesehen werden.
4. Es ist ein Zustandsdiagramm erstellt, welches die Logik, Menüs und Anzeigezustände des Programms abbildet.

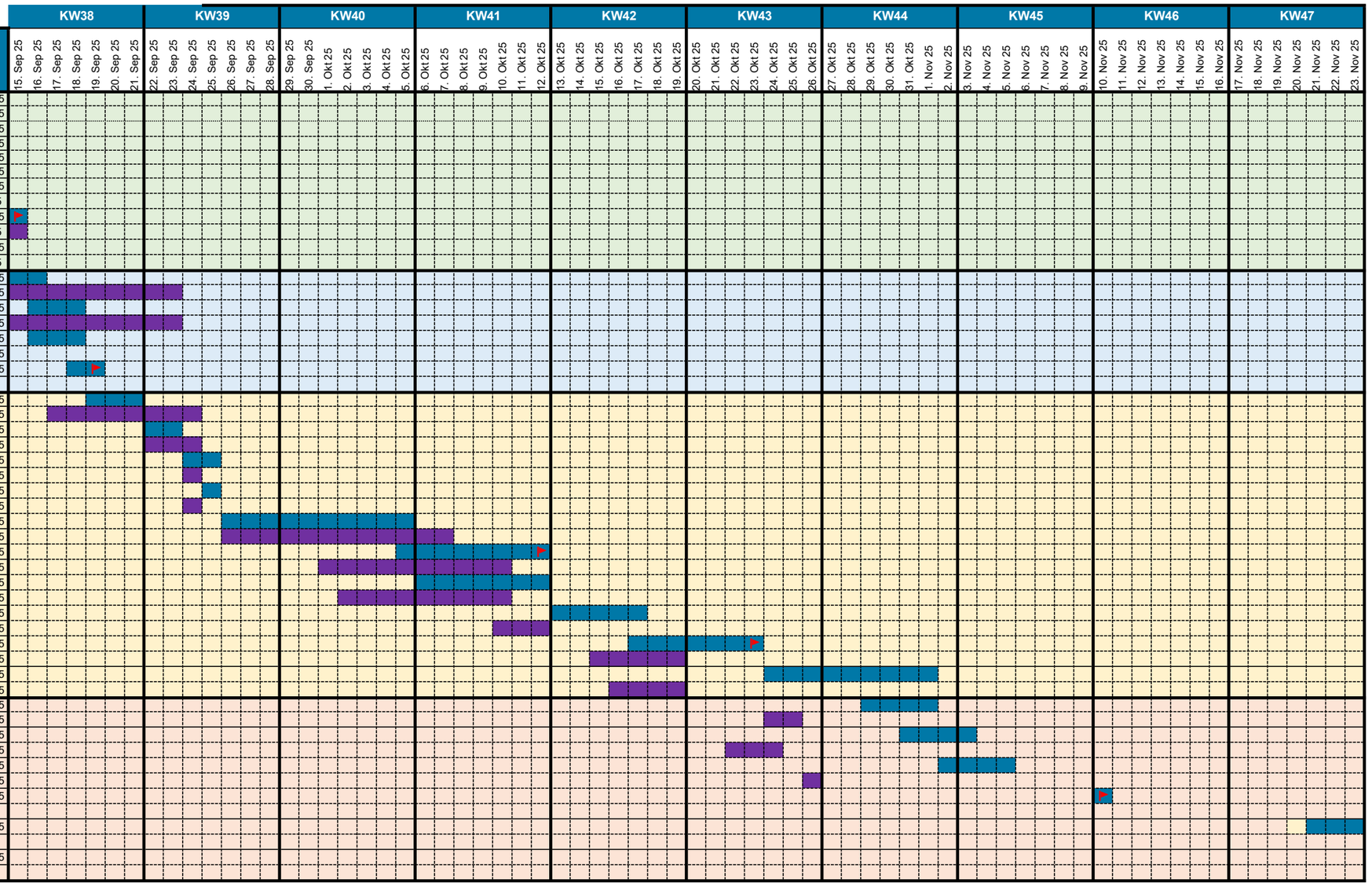

28.04.2025
Projektauftrag entgegennehmen
1 Tag
28.04.2025
01.05.2025
01.05.2025
29.04.2025
Thema finden
10 Tage
08.05.2025
05.05.1900
15.05.2025
09.05.2025
Auftrag klären
8 Tage
16.05.2025
15.05.2025
22.05.2025
09.05.2025
Zielscheibe ausarbeiten
4 Tage
12.05.2025
19.05.2025
22.05.2025
Pause
29.08.2025
Plichtenheft schreiben
15 Tage
12.09.2025
11.09.2025
25.09.2025
29.08.2025
Ausgangslage analysieren
11 Tage
08.09.2028
15.09.2025
25.09.2025
13.09.2025
Projektstrukturplan erstellen
2 Tage
14.09.2025
26.09.2025
27.09.2025
Projektstart
15.09.2025
Netzplanung erstellen
3 Tage
17.09.2025
28.09.2025
30.09.2025
15.09.2025
Gantt- Diagramm erstellen
3 Tage
17.09.2025
28.09.2025
30.09.2025
18.09.2025
Pflichtenheft und Zielscheibe genehmigen lassen
2 Tage
20.09.2025
30.09.2025
01.10.2025
21.09.2025
Systemarchitektur definieren
3 Tage
23.09.2025
02.10.2025
04.10.2025
24.09.2025
Teilprogramme schreiben
2 Tage
25.06.2025
05.10.2025
06.10.2025
26.09.2025
Komponenten testen
2 Tage
27.09.2025
07.10.2025
08.10.2025
26.09.2025
Komponenten versuchsweise zusammenbauen
1 Tage
26.09.2025
08.10.2025
08.10.2025
28.09.2025
Hauptprogramm schreiben
8 Tage
05.10.2025
08.10.2025
15.10.2025
06.10.2025
Hauptprogramm testen
6 Tage
11.10.2025
16.10.2025
24.10.2025
06.10.2025
Fehlerbeheben
5 Tage
10.10.2025
20.10.2025
24.10.2025
12.10.2025
Motorrad Dashboard zusammenbauen
3 Tage
14.10.2025
22.10.2025
24.10.2025
15.05.2025
Funktionen testen
5 Tage
19.10.2025
25.10.2025
29.10.2025
20.10.2025
Optimeren & Fertigstellen
5 Tage
24.10.2025
30.10.2025
03.11.2025
25.10.2025
Projektüberwachen
2 Tage
26.10.2025
04.11.2025
05.11.2025
27.10.2025
Zielerreichung evaluiern
2 Tage
28.10.2025
06.11.2025
07.11.2025
29.10.2025
Lessions Learnt schreiben
2 Tage
30.10.2025
08.11.2025
09.11.2025
FIX Termin
10.11.2025
Abgeben der Diplomarbeit
1 Tag
FIX Termin
10.11.2025
FIX Termin
28.11.2025
Diplomarbeit onlinepublizieren
1 Tag
FIX Termin
28.11.2025
FIX Termin
05.12.2025
Diplomarbeit präsentieren
2 Tage
FIX Termin
05.12.2025


Motorrad Dashboard
Phase Projektinitialisierung
Thema finden
Auftrag klären
Zielschaibe ausarbeiten
Plichtenheft schreiben
Ausgangslage analysieren
Phase Projektplanung
Projektstrukturplan erstellen
Gantt- Diagramm erstellen
Netzplanung erstellen
Phase Projektrealisierung
Systemarchitektur definieren
Teilprogramme schreiben
Komponenten testen
Komponenten versuchsweise zusammenbauen
Hauptprogramm schreiben
Programm testen
Fehlerbeheben
Motorrad Dashboard zusammenbauen
Funktionen testen
Optimeren & Fertigstellen
Phase Projektabschluss
Projektüberwachen
Zielerreichung evaluiern
Lessions Learnt schreiben
Diplomarbeit Online publizieren

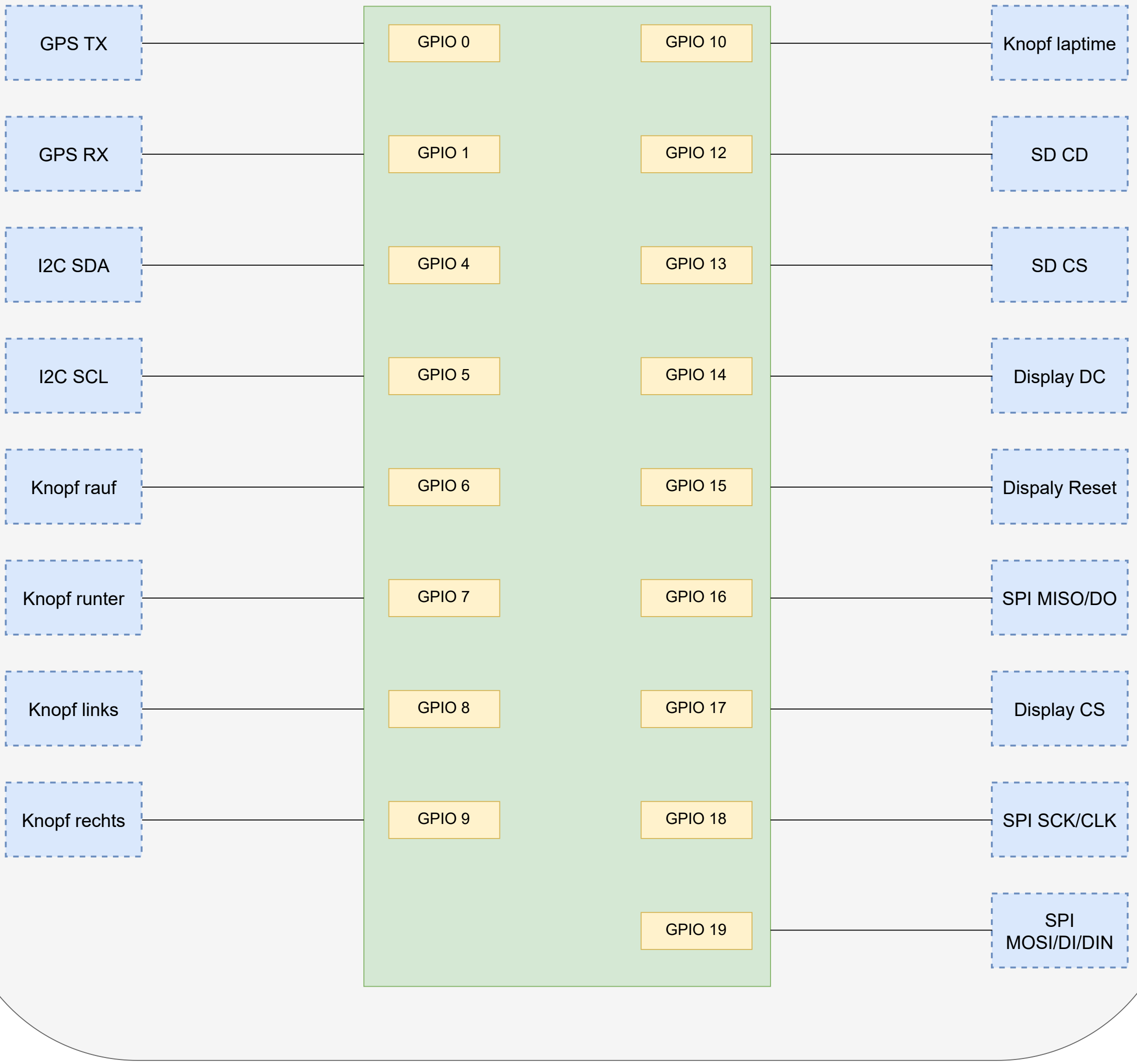
Gantt-Diagramm Diplomarbeit Motorrad Dashboard

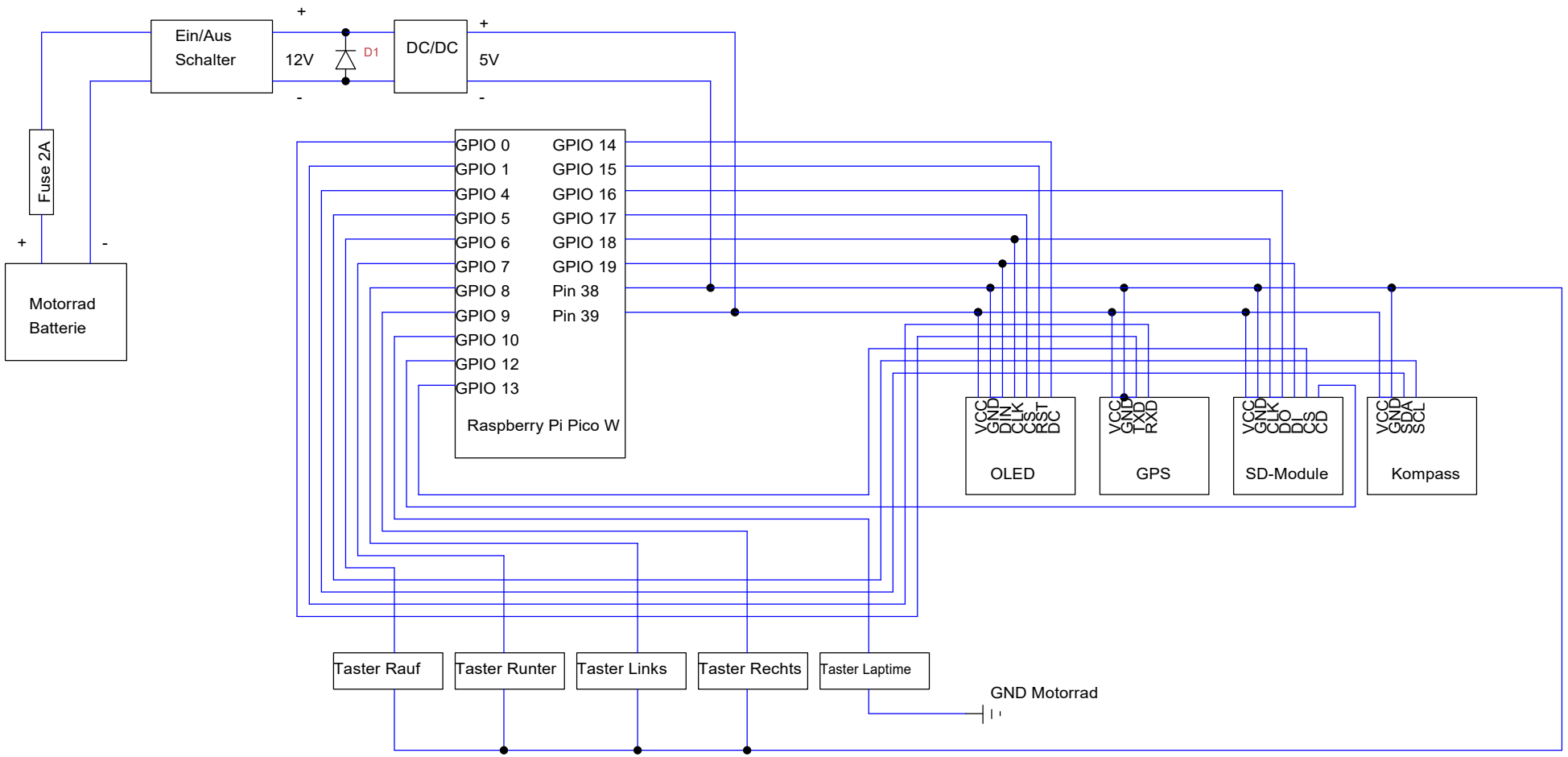
Legende:		
Soll:	Ist:	Meilenstein:

Projektphase:	Tätigkeit:	Tage:	Anfang	Ende
Projektinitialisierung	Projektauftrag entgegennehmen	4	28.04.2025	01.05.2025
			28.04.2025	29.04.2025
	Thema finden	15	01.05.2025	15.05.2025
			02.05.2025	20.05.2025
	Auftrag klären	8	15.05.2025	22.05.2025
			21.05.2025	25.05.2025
	Zielscheibe ausarbeiten	8	15.05.2025	22.05.2025
			10.09.2025	11.09.2025
	Plichtenheft schreiben	18	29.08.2025	15.09.2025
			04.09.2025	15.09.2025
	Ausgangslage analysieren	11	29.08.2025	08.09.2025
			26.08.2025	13.09.2025
Projektplanung	Projektstrukturplan erstellen	2	15.09.2025	16.09.2025
			01.09.2025	23.09.2025
	Gantt- Diagramm erstellen	3	16.09.2025	18.09.2025
			07.09.2025	23.09.2025
	Netzplanung erstellen	3	16.09.2025	18.09.2025
			07.09.2025	09.09.2025
	Pflichtenheft und Zielscheibe genehmigen lassen	2	18.09.2025	19.09.2025
			11.09.2025	
Projektrealisierung	Systemarchitektur definieren	3	19.09.2025	21.09.2025
			17.09.2025	24.09.2025
	Teilprogramme schreiben	2	22.09.2025	23.09.2025
			22.09.2025	24.09.2025
	Komponenten testen	2	24.09.2025	25.09.2025
			24.09.2025	24.09.2025
	Komponenten versuchsweise zusammenbauen	1	25.09.2025	25.09.2025
			24.09.2025	24.09.2025
	Hauptprogramm schreiben	10	26.09.2025	05.10.2025
			26.09.2025	07.10.2025
	Hauptprogramm testen	8	05.10.2025	12.10.2025
			01.10.2025	10.10.2025
	Fehlerbeheben	7	06.10.2025	12.10.2025
			02.10.2025	10.10.2025
	Motorrad Dashboard zusammenbauen	5	13.10.2025	17.10.2025
			10.10.2025	12.10.2025
	Funktionen testen	7	17.10.2025	23.10.2025
			15.10.2025	19.10.2025
	Optimeren & Fertigstellen	9	24.10.2025	01.11.2025
			16.10.2025	19.10.2025
Projektabschluss	Projektüberwachen	4	29.10.2025	01.11.2025
			24.10.2025	25.10.2025
	Zielerreichung evaluiern	4	31.10.2025	03.11.2025
			22.10.2025	24.10.2025
	Lessions Learnt schreiben	4	02.11.2025	05.11.2025
			26.10.2025	26.10.2025
	Abgeben der Diplomarbeit	1	10.11.2025	10.11.2025
	Diplomarbeit onlinepublizieren	8	21.11.2025	28.11.2025
	Diplomarbeit präsentieren	1	05.12.2025	05.12.2025



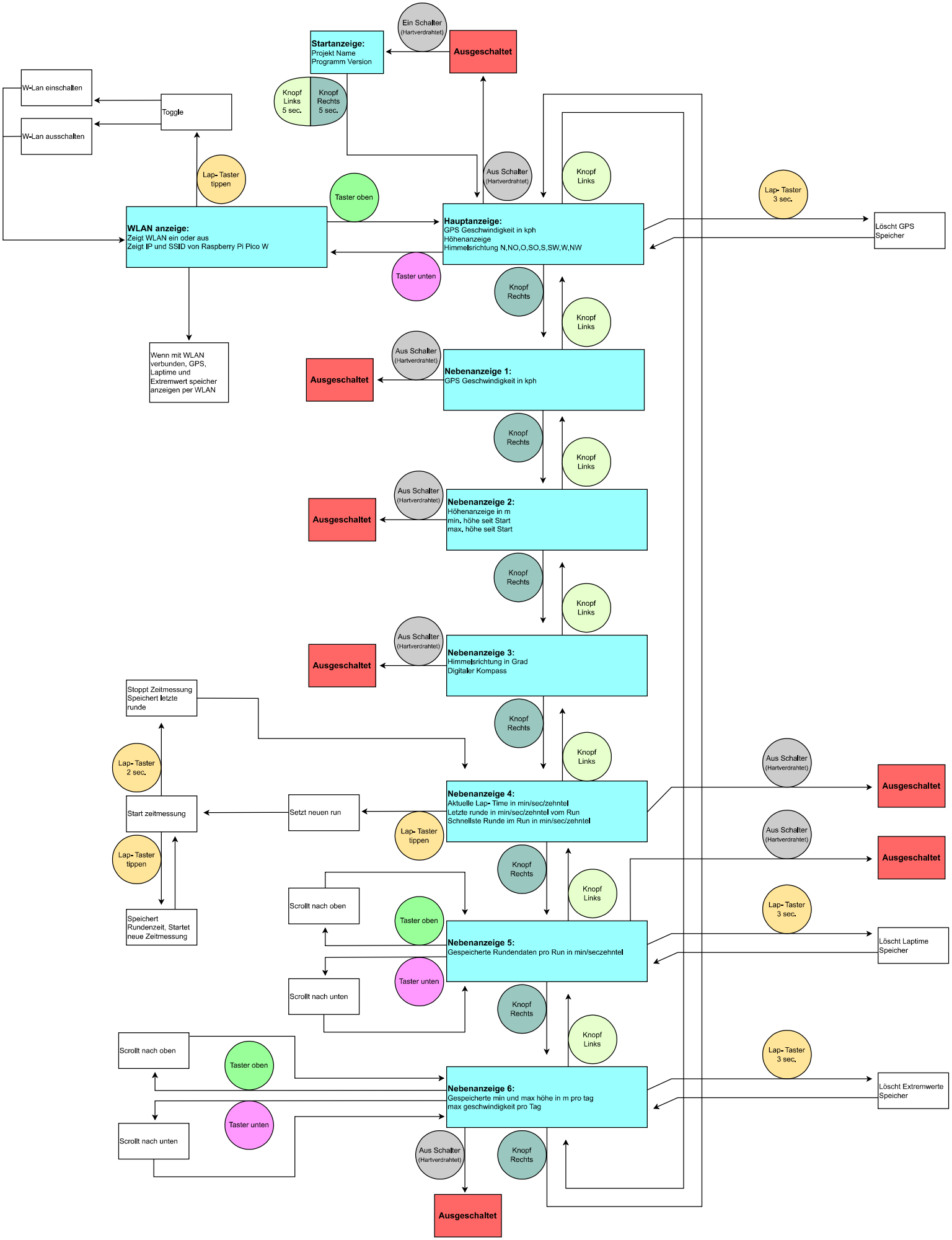
Layout Raspberry Pi Pico W





Title Motorrad Dashboard		
Author Jonas Kuster		
File C:\Users\Jonas\Download ... 08_Schaltplan.dsn		Document 1
Revision 1.5	Date 23.10.2025	Sheets 1 of 1

Motorrad Dashboard – State Machine



```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Motorrad_Dashboard
//====================
//
// Beschreibung Programm
//======================
// Das Programm erfasst und zeigt wichtige Fahrdaten eines Motorrads an. Es misst die GPS-Geschwindigkeit,
// bestimmt die Kompassrichtung, ermittelt die Höhe und ermöglicht die Messung von Laptimes.
// Alle Informationen werden übersichtlich auf einem Display dargestellt.
// Die gespeicherten Werte sind über einen integrierten WLAN-Hotspot auch per Webinterface abrufbar.
//
// Jonas.kuj@bluewin.ch
// Jonas.Kuster@edu.teko.ch
// TEKO Luzern
//
// Verwendete Bibliotheken und Quellen
//====================================
// Board Library für Raspberry Pi Pico W:
// https://github.com/earlephilhower/arduino-pico
//
// Adafruit GFX Library:
// https://github.com/adafruit/Adafruit-GFX-Library
//
// Adafruit SSD1327 OLED Display Library:
// https://github.com/adafruit/Adafruit_SSD1327
//
// DFRobot QMC5883 Kompass-Sensor:
// https://github.com/DFRobot/DFRobot_QMC5883
//
// TinyGPSPlus:
// https://github.com/mikalhart/TinyGPSPlus
```

```
//
// Board-eigene Bibliotheken (Arduino Pico Core):
// - SPI.h
// - Wire.h
// - SoftwareSerial.h
// - WiFi.h
// - WebServer.h
// - SD.h
//
// Hinweis: Die jeweiligen Bibliotheken stehen unter der MIT- oder BSD-Lizenz.
// Bei Weiterverwendung dieses Codes sind deren Lizenzbedingungen zu beachten.
//
// Hinweis zur Weiterverwendung
//==============================
// Bei Weiterverwendung des Codes bitte Urhebervermerk und diesen Header beibehalten.
// Anpassungen und Erweiterungen sind erlaubt, bitte ebenfalls dokumentieren.
//
// Versionsverlauf
//=================
// v01_00: 26.09.2025 Initialversion
// v01_01: 26.09.2025 GPS Grundcode hinzugefügt
// v01_02: 26.09.2025 Magnetometer Grundcode hinzugefügt + I2C Initialisierung
// v01_03: 26.09.2025 Display Grundcode hinzugefügt + SPI Initialisierung
// v01_04: 26.09.2025 GPS Werteglättung hinzugefügt
// v01_05: 26.09.2025 Grundstruktur des Programms überarbeitet
// v01_06: 26.09.2025 Header überarbeitet
// v01_07: 27.09.2025 Library Quellen hinzugefügt
// v01_08: 27.09.2025 Library in Ordner hinzugefügt
// v01_09: 27.09.2025 SD Libary und Code hinzugefügt
// v01_10: 28.09.2025 Hauptanzeige Kompass
// v01_11: 28.09.2025 Hauptanzeige WLAN + Menüstruktur
```

```
// v01_12: 29.09.2025 Taster auf Flankensteuerung
// v01_13: 29.09.2025 Taster links/rechts und rauf/runter gewechselt
// v01_14: 30.09.2025 Library TinyGPSPlus hinzugefügt & Höhe- und Geschwindigkeitsfunktion hinzugefügt
// v01_15: 30.09.2025 Kompass Ansicht hinzugefügt
// v01_16: 01.10.2025 Lap- Time Funktion hinzugefügt
// v01_17: 02.10.2025 Lap- Time Speicher hinzugefügt
// v01_18: 02.10.2025 Lap- Time scrollen hinzugefügt
// v01_19: 02.10.2025 Lap- Time Speicher per WLAN einsehen
// v01_20: 03.10.2025 Extremwerte Speicher + Wlan hinzufügen
// v01_21: 04.10.2025 Welcome Screen
// v01_22: 04.10.2025 GPS Update der Anzeige Case 0-2 bug Behebung
// v01_23: 04.10.2025 GPS Datum bug beheben Laptime
// v01_24: 05.10.2025 GPS Tracking hinzugefügt
// v01_25: 05.10.2025 RunCounter.txt bug Behebung
// v01_26: 06.10.2025 Extremwerte.txt bug Behebung
// v01_27: 06.10.2025 Speichern von Extremwerten
// v01_28: 06.10.2025 Laptime Messung bug Behebung + Speicher auf SD via Buffer
// v01_29: 07.10.2025 Extremwerte speichern nach Tag
// v01_30: 07.10.2025 v2. Extremwerte speichern nach Tag
// v01_31: 08.10.2025 v2. Extremwerte Date Update bug Behebung
// v01_32: 13.10.2025 Kompass auf GPS Daten umbau
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Version
const char* versionX = ("v01_32");

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Bibliotheken
//Board Library von: https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json
#include <Adafruit_GFX.h>         //Adafruit GFX Library by Adafruit + Adafruit BusI0 by Adafruit
```

```
#include <Adafruit_SSD1327.h>     //Adafruit SSD 1327 by Adafruit + Adafruit BusI0 by Adafruit
#include <SPI.h>                  //von Board Library
#include <SoftwareSerial.h>       //von Board Library
#include <DFRobot_QMC5883.h>      //DFRobot_QMC5883 by DFRobot
#include <Wire.h>                 //von Board Library
#include <WiFi.h>                 //von Board Library
#include <WebServer.h>            //von Board Library
#include <SD.h>                   //von Board Library
#include <TinyGPSPlus.h>          //TinyGPSPlus by Mikal Hart

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Display-Pins
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 128
#define DISPLAY_CS    17 //Chip Select
#define DISPLAY_DC    14 //Data/Command
#define DISPLAY_RESET 15

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//SPI-Pins
#define SPI_MISO 16 //(DO/MISO)
#define SPI_MOSI 19 //(DI/DIN/MOSI)
#define SPI_SCK  18 //(CLK/SCK)

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//SD-Pins
#define SD_CD 12 //Card Detect
#define SD_CS 13 //Chip Select


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
//I2C-Pins
#define SCL_PIN 5 //I2C SCL auf GPIO 5
#define SDA_PIN 4 //I2C SDA auf GPIO 4

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Knöpfe
#define KNOPF_RAUF 6
#define KNOPF_RUNTER 7
#define KNOPF_LINKS 8
#define KNOPF_RECHTS 9
#define KNOPF_LAPTIME 10
int menuIndex = 0;         //  Menüpunkt (0..6)
const int maxMenu = 6;     // höchste Zahl
bool lastKNOPF_RAUF = HIGH;
bool lastKNOPF_RUNTER = HIGH;
bool lastKNOPF_LINKS = HIGH;
bool lastKNOPF_RECHTS = HIGH;
bool lastKNOPF_LAPTIME = HIGH;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Display-Objekt SPI0
Adafruit_SSD1327 display(SCREEN_WIDTH, SCREEN_HEIGHT, &SPI, DISPLAY_DC, DISPLAY_RESET, DISPLAY_CS);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//GPS-Objekt
SoftwareSerial gpsSerial(0, 1); // GPIO 0= TXD, GPIO 1= RXD Pins (RXD braucht es nicht)
char lastGpsChar = ' ';
TinyGPSPlus gps;
int gpsspeed = 0;
int gpsaltitude = 0;
int gpsdate = 0;
```

```
unsigned long lastDateUpdate = 0;
unsigned long lastSpeedUpdate = 0;
unsigned long lastAltitudeUpdate = 0;
unsigned long lastGpsUpdate = 0;
unsigned long lastSaveTime = 0;
const unsigned long GPS_INTERVAL = 1000;      // 1 Sekunde
const unsigned long SAVE_INTERVAL = 60000;   // 1 Minuten
String gpsBuffer = "";
unsigned long messageStartTime = 0;
bool messageActive = false;
String messageLine1 = "";
String messageLine2 = "";
File trackFile;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Höhe Glättung
#define NUM_ALT_SAMPLES 3
float altitudeBuffer[NUM_ALT_SAMPLES];
int altitudeIndex = 0;
bool altitudeBufferFull = false;

float minAltitude = 1e6;
float maxAltitude = -1e6;

float smoothAltitude(float newAlt) {
    altitudeBuffer[altitudeIndex] = newAlt;
    altitudeIndex = (altitudeIndex + 1) % NUM_ALT_SAMPLES;
    if (altitudeIndex == 0) altitudeBufferFull = true;

    // Mittelwert berechnen
    int count = altitudeBufferFull ? NUM_ALT_SAMPLES : altitudeIndex;
```

```
    float sum = 0;
    for (int i = 0; i < count; i++) sum += altitudeBuffer[i];
    float avg = sum / count;

    // Min/Max aktualisieren
    if (newAlt < minAltitude) minAltitude = newAlt;
    if (newAlt > maxAltitude) maxAltitude = newAlt;

    return avg;
}


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Kompass-Objekt mit I2C-Adresse 0x0D -> wird von GPS gebraucht
//DFRobot_QMC5883 compass(&Wire, 0x0D);
float heading = 0.0;
float winkelOffset = 0.0; //offset für Komass bei fixmontage (metallische abweichung)

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Kompass Glättung
#define ZAHLEN_MENGE 50
float richtungsBuffer[ZAHLEN_MENGE];
int ZahlenIndex = 0;
bool buffer = false;

float glaetungMesswert(float neueRichtung) {
  richtungsBuffer[ZahlenIndex] = neueRichtung;
  ZahlenIndex = (ZahlenIndex + 1) % ZAHLEN_MENGE;
  if (ZahlenIndex == 0) buffer = true;
  int count = buffer ? ZAHLEN_MENGE : ZahlenIndex;
  float sumSin = 0;
```

```
  float sumCos = 0;
  for (int i = 0; i < count; i++) {
    float rad = richtungsBuffer[i] * DEG_TO_RAD;  // Grad -> Bogenmass
    sumSin += sin(rad);
    sumCos += cos(rad);
  }
  float avgAngle = atan2(sumSin / count, sumCos / count) * RAD_TO_DEG;  // Bogenmass -> Grad
  if (avgAngle < 0) avgAngle += 360.0;
  return avgAngle;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Kompass N,NO,O,SO,S,SW,W,NW
String getDirection(float heading) {
  if (heading >= 337.5 || heading < 22.5)       return "N";
  else if (heading >= 22.5 && heading < 67.5)   return "NO";
  else if (heading >= 67.5 && heading < 112.5)  return "O";
  else if (heading >= 112.5 && heading < 157.5) return "SO";
  else if (heading >= 157.5 && heading < 202.5) return "S";
  else if (heading >= 202.5 && heading < 247.5) return "SW";
  else if (heading >= 247.5 && heading < 292.5) return "W";
  else if (heading >= 292.5 && heading < 337.5) return "NW";
  return "?";
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Kompassanzeige
#define RADIUS 60
#define CENTER_X (SCREEN_WIDTH/2)
#define CENTER_Y (SCREEN_HEIGHT/2)
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Lap- Time + Extremwerte speicher
//case 5
unsigned int runNumber = 0;
unsigned long runStart = 0;
File lapFile;
File runFile;
File tmpFile;
int displayStartLine = 0;          // erste Zeile der aktuellen Seite
unsigned long lastScrollTimeRAUF = 0;
unsigned long lastScrollTimeRUNTER = 0;
unsigned long totalLapTime = 0; // Laptime Gesamtzeit
#define MAX_LAPS 100
unsigned long lapTimes[MAX_LAPS];  // Array für die Rundenzeiten
unsigned int lapCount = 0;         // Anzahl der aktuell gespeicherten Runden
unsigned long pressStart = 0;
bool timingActive = false;
bool valuesAvailable = false;
unsigned long lapStart = 0, lastLap = 0, bestLap = 0;
bool longPressDone = false;
unsigned int lapNumber = 0;
bool firstLapSkipped = false;

//case 6
float minHoeheTag = 1e6;
float maxHoeheTag = -1e6;
float maxSpeedTag = 0;
String DateTag;
float minHoeheTagSD = 1e6;
float maxHoeheTagSD = -1e6;
float maxSpeedTagSD = 0;
```

```
String DateTagSD;
int displayStartLineExt = 0; // Scrollposition für Extremwerte.txt
unsigned long lastScrollTimeRAUF_Ext = 0;
unsigned long lastScrollTimeRUNTER_Ext = 0;
float altitude;
float speed;
bool updatedmaxHoehe;
bool updatedminHoehe;
bool updatedmaxSpeed;
File extFile;
File maxHoeheTagFile;
File minHoeheTagFile;
File maxSpeedTagFile;
File DateTagFile;
File tmpmaxHoeheTagFile;
File tmpminHoeheTagFile;
File tmpmaxSpeedTagFile;
File tmpDateTagFile;
int firstupdate = 0;
String stdateStr;


//allgemein
const unsigned long scrollDebounce = 50; // ms
String formatLapTime(unsigned long timeMs) {
    unsigned int minutes = timeMs / 60000;
    unsigned int seconds = (timeMs % 60000) / 1000;
    unsigned int hundredths = (timeMs % 1000) / 10;
    char buffer[20];
    sprintf(buffer, "%u:%02u.%02u", minutes, seconds, hundredths);
    return String(buffer);
```

```
}
#define MAX_LINES_DISPLAY 13
char dateStr[20];

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Case 6 Extremwerte schreiben
void updateExtremwerte() {
        DateTagFile = SD.open("/DateTag.txt", FILE_READ);
        if (DateTagFile) {
            String lastLine;
            while (DateTagFile.available()) {
                lastLine = DateTagFile.readStringUntil('\n'); // letzte Zeile lesen
            }
            DateTagSD = lastLine;
            DateTagFile.close();
        } else {
            DateTagSD ="";
        }
        sprintf(dateStr, "%02d.%02d.%04d", gps.date.day(), gps.date.month(), gps.date.year());
        DateTag = dateStr;
        DateTagSD.trim();
        DateTag.trim();

        if (!DateTag.equals(DateTagSD)) {
            extFile = SD.open("/Extremwerte.txt", FILE_WRITE);
            if(extFile){
                extFile.println(String("Datum: ") + String(DateTagSD));
                extFile.println(String("Max Hoehe: ") + String((int)maxHoeheTag) + " m");
                extFile.println(String("Min Hoehe: ") + String((int)minHoeheTag) + " m");
                extFile.println(String("Max Geschw.: ") + String((int)maxSpeedTag) + " kph");
                extFile.println("");
```

```
            extFile.flush();
            extFile.close();
            delay(1);
            if (SD.exists("/maxHoeheTag.txt")) SD.remove("/maxHoeheTag.txt");
            if (SD.exists("/minHoeheTag.txt")) SD.remove("/minHoeheTag.txt");
            if (SD.exists("/maxSpeedTag.txt")) SD.remove("/maxSpeedTag.txt");
            if (SD.exists("/DateTag.txt")) SD.remove("/DateTag.txt");
            delay(10);
        }
            DateTagFile = SD.open("/DateTag.txt", FILE_WRITE);
            DateTagFile.println(dateStr); // Wert schreiben
            DateTagFile.flush();
            DateTagFile.close();
            delay(10);
    }

if(updatedmaxHoehe == true){
    maxHoeheTagFile = SD.open("/maxHoeheTag.txt", FILE_READ);
    if (maxHoeheTagFile) {
        String lastLine;
        while (maxHoeheTagFile.available()) {
            lastLine = maxHoeheTagFile.readStringUntil('\n'); // letzte Zeile lesen
        }
        maxHoeheTagSD = lastLine.toInt();
        maxHoeheTagFile.close();
    } else {
        maxHoeheTagSD = -1e6;
    }

    if (maxHoeheTag > maxHoeheTagSD){
        tmpmaxHoeheTagFile = SD.open("/maxHoeheTag.tmp", FILE_WRITE);
```

```
        if(tmpmaxHoeheTagFile){
            tmpmaxHoeheTagFile.println(maxHoeheTag); // Wert schreiben
            tmpmaxHoeheTagFile.flush();
            tmpmaxHoeheTagFile.close();
            delay(10);
            if (SD.exists("/maxHoeheTag.txt")) SD.remove("/maxHoeheTag.txt");
            SD.rename("/maxHoeheTag.tmp", "/maxHoeheTag.txt");
        }
    }

}

if(updatedminHoehe == true){
    minHoeheTagFile = SD.open("/minHoeheTag.txt", FILE_READ);
    if (minHoeheTagFile) {
        String lastLine;
        while (minHoeheTagFile.available()) {
            lastLine = minHoeheTagFile.readStringUntil('\n'); // letzte Zeile lesen
        }
        minHoeheTagSD = lastLine.toInt();
        minHoeheTagFile.close();
    } else {
        minHoeheTagSD = 1e6;
    }

    if (minHoeheTag < minHoeheTagSD){
        tmpminHoeheTagFile = SD.open("/minHoeheTag.tmp", FILE_WRITE);
        if(tmpminHoeheTagFile){
            tmpminHoeheTagFile.println(minHoeheTag); // Wert schreiben
            tmpminHoeheTagFile.flush();
            tmpminHoeheTagFile.close();
```

```
            delay(10);
            if (SD.exists("/minHoeheTag.txt")) SD.remove("/minHoeheTag.txt");
            SD.rename("/minHoeheTag.tmp", "/minHoeheTag.txt");
        }
    }

}

if(updatedmaxSpeed == true){
    maxSpeedTagFile = SD.open("/maxSpeedTag.txt", FILE_READ);
    if (maxSpeedTagFile) {
        String lastLine;
        while (maxSpeedTagFile.available()) {
            lastLine = maxSpeedTagFile.readStringUntil('\n'); // letzte Zeile lesen
        }
        maxSpeedTagSD = lastLine.toInt();
        maxSpeedTagFile.close();
    } else {
        maxSpeedTagSD = 0;
    }

    if (maxSpeedTag > maxSpeedTagSD){
        tmpmaxSpeedTagFile = SD.open("/maxSpeedTag.tmp", FILE_WRITE);
        if(tmpmaxSpeedTagFile){
            tmpmaxSpeedTagFile.println(maxSpeedTag); // Wert schreiben
            tmpmaxSpeedTagFile.flush();
            tmpmaxSpeedTagFile.close();
            delay(10);
            if (SD.exists("/maxSpeedTag.txt")) SD.remove("/maxSpeedTag.txt");
            SD.rename("/maxSpeedTag.tmp", "/maxSpeedTag.txt");
        }
```

```
        }

    }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//PICO Hotspot
const char* ssid = "PicoW_Hotspot";
const char* password = "12345678";
bool wlanAn = false;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// PICO Hotspot HTTP-Webserver auf Port 80
WebServer server(80);


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// PICO Hotspot Root-Seite
void handleRoot() {
  String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'><title>Pico W</title>";
  // CSS für Tabs
  html += "<style>";
  html += ".tab { overflow: hidden; border-bottom: 1px solid #ccc; }";
  html += ".tab button { background-color: inherit; border: none; outline: none; cursor: pointer; padding: 10px 20px; transition: 0.3s; }";
  html += ".tab button:hover { background-color: #ddd; }";
  html += ".tab button.active { background-color: #bbb; }";
  html += ".tabcontent { display: none; padding: 10px; }";
  html += "</style>";

  // JavaScript für Tabs + Fetch
  html += "<script>";
```

```
html += "var currentTab='';";
html += "function openTab(tabName, btn){";
html += "  var tabcontent=document.getElementsByClassName('tabcontent');";
html += "  for(var i=0;i<tabcontent.length;i++){tabcontent[i].style.display='none';}";
html += "  var tabbuttons=document.getElementsByClassName('tabbtn');";
html += "  for(var i=0;i<tabbuttons.length;i++){tabbuttons[i].className=tabbuttons[i].className.replace(' active','');}";
html += "  document.getElementById(tabName).style.display='block';";
html += "  btn.className+=' active';";
html += "  currentTab=tabName;";
html += "}";
// Tabs aktualisieren
html += "function updateTab(){";
html += "  if(currentTab=='Laptimes'){fetch('/laptimes').then(r=>r.text()).then(d=>{document.getElementById('Laptimes').innerText=d;});}";
html += "  else if(currentTab=='Extremwerte'){fetch('/extremwerte').then(r=>r.text()).then(d=>{document.getElementById('Extremwerte').innerText=d;});}";
html += "  else if(currentTab=='Tracking'){fetch('/tracking').then(r=>r.text()).then(d=>{document.getElementById('Tracking').innerText=d;});}";
html += "}";
html += "setInterval(updateTab,500);";
html += "window.onload=function(){document.getElementById('defaultTab').click();};";
html += "</script></head><body>";

// Tab Buttons
html += "<div class='tab'>";
html += "<button class='tabbtn' id='defaultTab' onclick=\"openTab('Laptimes',this)\">Laptimes</button>";
html += "<button class='tabbtn' onclick=\"openTab('Extremwerte',this)\">Extremwerte</button>";
html += "<button class='tabbtn' onclick=\"openTab('Tracking',this)\">Tracking</button>";
html += "</div>";

// Tab Content
html += "<div id='Laptimes' class='tabcontent'><pre>Lade...</pre></div>";
html += "<div id='Extremwerte' class='tabcontent'><pre>Lade...</pre></div>";
html += "<div id='Tracking' class='tabcontent'><pre>Lade...</pre></div>";
```

```
  html += "</body></html>";

  server.send(200, "text/html", html);
}

////////////////////////////////////////////////////////////////////////////////////////////
// Handler für Laptime-Datei
void handleLaptimes() {
  if (SD.exists("/Laptime.txt")) {
    lapFile = SD.open("/Laptime.txt", FILE_READ);
    String content = "";
    while (lapFile.available()) {
      content += (char)lapFile.read();
    }
    lapFile.close();
    server.send(200, "text/plain", content);
  } else {
    server.send(200, "text/plain", "Keine Laptimes vorhanden.");
  }
}

////////////////////////////////////////////////////////////////////////////////////////////
// Handler für Extremwerte-Datei
void handleExtremwerte() {
  if (SD.exists("/Extremwerte.txt")) {
    extFile = SD.open("/Extremwerte.txt", FILE_READ);
    String content = "";
    while (extFile.available()) {
      content += (char)extFile.read();
    }
```

```
    extFile.close();
    server.send(200, "text/plain", content);
  } else {
    server.send(200, "text/plain", "Keine Extremwerte vorhanden.");
  }
}

////////////////////////////////////////////////////////////////////////////////////////////
// Handler für Track-Datei
void handleTracking() {
  if (SD.exists("/tracking.txt")) {
    trackFile = SD.open("/tracking.txt", FILE_READ);
    String content = "";
    while (trackFile.available()) {
      content += (char)trackFile.read();
    }
    trackFile.close();
    server.send(200, "text/plain", content);
  } else {
    server.send(200, "text/plain", "Keine Trackingwerte vorhanden.");
  }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// void setup
void setup(){
  Serial.begin(115200);

  //I2C
  Wire.setSCL(SCL_PIN);
  Wire.setSDA(SDA_PIN);
```

```
Wire.begin();

//Display
if (!display.begin()) {
  Serial.println("Display nicht erkannt!");
  while (true) {
      delay(1000); // Endlosschleife, verhindert Weiterlaufen
  }
}
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1327_WHITE);
display.display();

//GPS
gpsSerial.begin(9600); // GPS-Modul Baudrate
Serial.println("GPS gestartet");

//Kompass
// while (!compass.begin());
//Serial.println("Kompass gestartet");

//SD
SD.begin(SD_CS);
if (!SD.begin(SD_CS)) {
  Serial.println("SD-Karte nicht erkannt!");
  while (true) {
      delay(1000); // Endlosschleife, Programm stoppt hier
  }
}
Serial.println("SD gestartet");
```

```
pinMode(SD_CD, INPUT_PULLUP);
// Run-Zähler laden
runFile = SD.open("/RunCounter.txt", FILE_READ);
if (runFile) {
    String lastLine;
    while (runFile.available()) {
        lastLine = runFile.readStringUntil('\n'); // letzte Zeile lesen
    }
    runNumber = lastLine.toInt();
    delay(10);
    runFile.close();
} else {
    runNumber = 0; // erste Benutzung
}

  // maxSpeedTag laden
maxSpeedTagFile = SD.open("/maxSpeedTag.txt", FILE_READ);
if (maxSpeedTagFile) {
    String lastLine;
    while (maxSpeedTagFile.available()) {
        lastLine = maxSpeedTagFile.readStringUntil('\n'); // letzte Zeile lesen
    }
    maxSpeedTag = lastLine.toInt();
    delay(10);
    maxSpeedTagFile.close();
} else {
    maxSpeedTag = 0; // erste Benutzung
}

  // maxHoeheTag laden
maxHoeheTagFile = SD.open("/maxHoeheTag.txt", FILE_READ);
```

```
if (maxHoeheTagFile) {
    String lastLine;
    while (maxHoeheTagFile.available()) {
        lastLine = maxHoeheTagFile.readStringUntil('\n'); // letzte Zeile lesen
    }
    maxHoeheTag = lastLine.toInt();
    delay(10);
    maxHoeheTagFile.close();
} else {
     maxHoeheTag = -1e6;
}

  // minHoeheTag laden
  minHoeheTagFile = SD.open("/minHoeheTag.txt", FILE_READ);
  if (minHoeheTagFile) {
      String lastLine;
      while (minHoeheTagFile.available()) {
          lastLine = minHoeheTagFile.readStringUntil('\n'); // letzte Zeile lesen
      }
      minHoeheTag = lastLine.toInt();
      delay(10);
      minHoeheTagFile.close();
  } else {
      minHoeheTag = 1e6; // erste Benutzung
  }

  // DateTag laden
  DateTagFile = SD.open("/DateTag.txt", FILE_READ);
  if (DateTagFile) {
      String lastLine;
      while (DateTagFile.available()) {
```

```
            lastLine = DateTagFile.readStringUntil('\n'); // letzte Zeile lesen
        }
        DateTag = lastLine;
        delay(10);
        DateTagFile.close();
    } else {
        DateTag =""; // erste Benutzung
    }

//Knöpfe
pinMode(KNOPF_RAUF,INPUT_PULLUP);
pinMode(KNOPF_RUNTER,INPUT_PULLUP);
pinMode(KNOPF_LINKS,INPUT_PULLUP);
pinMode(KNOPF_RECHTS,INPUT_PULLUP);
pinMode(KNOPF_LAPTIME, INPUT_PULLUP);

display.setCursor(10, 49);
display.println("Motorrad Dashboard");
display.setCursor(46, 63);
display.println(versionX);
display.display();

    while (digitalRead(KNOPF_LINKS) == HIGH || digitalRead(KNOPF_RECHTS) == HIGH) {
        delay(100);
    }
    unsigned long startTime = millis();
    while (millis() - startTime < 3000) {
        if (digitalRead(KNOPF_LINKS) == HIGH || digitalRead(KNOPF_RECHTS) == HIGH) {
            while (digitalRead(KNOPF_LINKS) == HIGH || digitalRead(KNOPF_RECHTS) == HIGH) {
                delay(10);
            }
```

```
            startTime = millis();
        }
        delay(10);
    }
    display.clearDisplay();
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//void loop
void loop() {

  // RECHTS
    int readingRECHTS = digitalRead(KNOPF_RECHTS);
    if (readingRECHTS == LOW && lastKNOPF_RECHTS == HIGH) { // fallende Flanke
        if (menuIndex != 7) {  // nur LINKS/RECHTS erlauben, wenn nicht WLAN-Menü
            menuIndex++;
            if (menuIndex > maxMenu) menuIndex = 0;
        }
    }
    lastKNOPF_RECHTS = readingRECHTS;

  // LINKS
    int readingLINKS = digitalRead(KNOPF_LINKS);
    if (readingLINKS == LOW && lastKNOPF_LINKS == HIGH) { // fallende Flanke
        if (menuIndex != 7) {  // nur LINKS/RECHTS erlauben, wenn nicht WLAN-Menü
            menuIndex--;
            if (menuIndex < 0) menuIndex = maxMenu;
        }
    }
    lastKNOPF_LINKS= readingLINKS;
```

```
// RUNTER
int readingRUNTER = digitalRead(KNOPF_RUNTER);
if (readingRUNTER == LOW && lastKNOPF_RUNTER == HIGH) { // fallende Flanke
    if (menuIndex == 0) menuIndex = 7; // vom Hauptmenü ins WLAN-Menü
    else if (menuIndex == 5) {        // Scrollen nur in Laptime-Anzeige
        displayStartLine += MAX_LINES_DISPLAY;
        lapFile = SD.open("/Laptime.txt", FILE_READ);
        if (lapFile) {
            int totalLines = 0;
            while (lapFile.available()) {
                lapFile.readStringUntil('\n');
                totalLines++;
            }
            lapFile.close();

            if (displayStartLine > totalLines - MAX_LINES_DISPLAY) {
                displayStartLine = max(0, totalLines - MAX_LINES_DISPLAY);
            }
        }
        lastScrollTimeRUNTER = millis();
    }
    else if (menuIndex == 6) {        // Scrollen in Extremwerte-Anzeige
        displayStartLineExt += MAX_LINES_DISPLAY;
        extFile = SD.open("/Extremwerte.txt", FILE_READ);
        if (extFile) {
            int totalLines = 0;
            while (extFile.available()) {
                extFile.readStringUntil('\n');
                totalLines++;
            }
            extFile.close();
```

```
            if (displayStartLineExt > totalLines - MAX_LINES_DISPLAY) {
                displayStartLineExt = max(0, totalLines - MAX_LINES_DISPLAY);
            }
        }
        lastScrollTimeRUNTER = millis();
    }
  }
  lastKNOPF_RUNTER = readingRUNTER;

  // RAUF
  int readingRAUF = digitalRead(KNOPF_RAUF);
  if (readingRAUF == LOW && lastKNOPF_RAUF == HIGH) { // fallende Flanke
      if (menuIndex == 7) menuIndex = 0;           // WLAN-Menü verlassen
      else if (menuIndex == 5) {                   // Scrollen nur in Laptime-Anzeige
          if (displayStartLine >= MAX_LINES_DISPLAY) displayStartLine -= MAX_LINES_DISPLAY;
          else displayStartLine = 0;
          lastScrollTimeRAUF = millis();
      }
      else if (menuIndex == 6) {                   // Scrollen in Extremwerte-Anzeige
          if (displayStartLineExt >= MAX_LINES_DISPLAY) displayStartLineExt -= MAX_LINES_DISPLAY;
          else displayStartLineExt = 0;
          lastScrollTimeRAUF = millis();
      }
  }
  lastKNOPF_RAUF = readingRAUF;

// Laptime / WLAN
  int readingLAPTIME = digitalRead(KNOPF_LAPTIME);
  // Fallende Flanke
  if (readingLAPTIME == LOW && lastKNOPF_LAPTIME == HIGH) {
```

```
pressStart = millis();
longPressDone = false; // Reset
// WLAN
if (menuIndex == 7) {
    if (!wlanAn) {
        WiFi.softAP(ssid, password);
        IPAddress ip = WiFi.softAPIP();
        server.on("/", handleRoot);
        server.on("/laptimes", handleLaptimes); //einzelaufruf
        server.on("/tracking", handleTracking); //einzelaufruf
        server.on("/extremwerte", handleExtremwerte); //einzelaufruf
        server.begin();
        wlanAn = true;
        Serial.println("WLAN + Webserver gestartet");
    } else {
        server.stop();
        WiFi.softAPdisconnect(true);MAX_LINES_DISPLAYtracking

        WiFi.disconnect(true);
        WiFi.mode(WIFI_OFF);
        wlanAn = false;
        Serial.println("WLAN + Webserver gestoppt");
    }
}
// Start Messung (kurzer Druck)
if (menuIndex == 4 && !timingActive) {
    lapStart = millis();
    timingActive = true;
    lapCount = 0;
    lapNumber = 0;
    bestLap = 0;
```

```
        totalLapTime = 0;
        valuesAvailable = false;
        firstLapSkipped = false;

        // Run-Zähler hochzählen + Startzeit merken
        runNumber++;
        runStart = millis();

        // RunCounter auf SD schreiben
        tmpFile = SD.open("/RunCounter.tmp", FILE_WRITE);
        if(tmpFile){
            tmpFile.println(runNumber); // Wert schreiben
            tmpFile.flush();
            tmpFile.close();
            delay(10);
            if (SD.exists("/RunCounter.txt")) SD.remove("/RunCounter.txt");
            SD.rename("/RunCounter.tmp", "/RunCounter.txt");
        }
    }
}

// Steigende Flanke (kurzer Druck, Runde)
if (readingLAPTIME == HIGH && lastKNOPF_LAPTIME == LOW) {
    if (menuIndex == 4 && timingActive) {
        if (!firstLapSkipped) {
            // Erstes Loslassen nach Start: keine Runde speichern
            firstLapSkipped = true;
            lapStart = millis(); // Startzeit für erste echte Runde
        } else {
            // Normale Runde
            unsigned long now = millis();
```

```
            unsigned long lapTime = now - lapStart;
            lastLap = lapTime;
            if (bestLap == 0 || lapTime < bestLap) bestLap = lapTime;

            lapStart = now; // Startzeit für nächste Runde
            valuesAvailable = true;

            // Runde im Array speichern
            if (lapCount < MAX_LAPS) {
                lapTimes[lapCount++] = (lapTime / 10) * 10; // auf 10 ms abschneiden
                lapNumber++;
            }
        }
    }
}

// Langer Druck (>2 Sekunden) = Stoppen
if (menuIndex == 4 && readingLAPTIME == LOW && !longPressDone) {
    if (millis() - pressStart >= 2000 && timingActive) {
        // letzte Runde speichern
        unsigned long now = millis();
        unsigned long lapTime = now - lapStart;
        if (lapTime >= 2000) lapTime -= 2000; {  // Start-Delay abziehen
            lastLap = lapTime;
            if (bestLap == 0 || lapTime < bestLap) bestLap = lapTime;
        }
        if (lapCount < MAX_LAPS) {
            lapTimes[lapCount++] = (lapTime / 10) * 10;
            lapNumber++;
        }
        // RunCounter auf SD schreiben beim Stoppen
```

```
tmpFile = SD.open("/RunCounter.tmp", FILE_WRITE);
if(tmpFile){
    tmpFile.println(runNumber); // Wert schreiben
    tmpFile.flush();
    tmpFile.close();
    delay(10);
    if (SD.exists("/RunCounter.txt")) SD.remove("/RunCounter.txt");
    SD.rename("/RunCounter.tmp", "/RunCounter.txt");
}
timingActive = false;
valuesAvailable = true;
longPressDone = true;

// Alle Runden auf SD schreiben
lapFile = SD.open("/Laptime.txt", FILE_WRITE);
if (lapFile) {
    lapFile.print("Turn ");
    lapFile.print(runNumber);
    lapFile.print(" (");
    if (gpsdate == 1) {
        sprintf(dateStr, "%02d.%02d.%04d", gps.date.day(), gps.date.month(), gps.date.year());
        if (strcmp(dateStr, "00.00.2000") == 0) lapFile.print("kein Datum");
        else lapFile.print(dateStr);
    } else lapFile.print("kein Datum");
    lapFile.println(")");

    // Runden schreiben
    totalLapTime = 0;
    for (unsigned int i = 0; i < lapCount; i++) {
        lapFile.print(i + 1);
        lapFile.print(": ");
```

```
                lapFile.println(formatLapTime(lapTimes[i]));
                totalLapTime += lapTimes[i];
            }

            // Gesamtzeit
            lapFile.print("Gesamtzeit: ");
            lapFile.println(formatLapTime(totalLapTime));
            lapFile.println(); // Leerzeile zwischen Runs
            lapFile.close();
        }

        // Array zurücksetzen
        lapCount = 0;
        totalLapTime = 0;
    }
}

lastKNOPF_LAPTIME = readingLAPTIME;

//Löschen Case 5
if (menuIndex == 5) {
    int readingLAPTIME = digitalRead(KNOPF_LAPTIME);

    // Fallende Flanke: Start der Messung für langen Druck
    if (readingLAPTIME == LOW && lastKNOPF_LAPTIME == HIGH) {
        pressStart = millis();
        longPressDone = false; // Reset
    }

    // Langer Druck: Laptime Dateien löschen
    if (readingLAPTIME == LOW && !longPressDone) {
```

```
        if (millis() - pressStart >= 3000) { // 3 Sekunden gedrückt halten
            while(!SD.remove("/Laptime.txt"));
            while(!SD.remove("/RunCounter.txt"));
            displayStartLine = 0;  // Anzeige zurücksetzen
            runNumber = 0;         // Zähler zurücksetzen
            longPressDone = true;  // mehrfaches Auslösen verhindern
        }
    }
    lastKNOPF_LAPTIME = readingLAPTIME;
}
//Löschen Case 6
if (menuIndex == 6) {
    int readingLAPTIME = digitalRead(KNOPF_LAPTIME);

    // Fallende Flanke: Start der Messung für langen Druck
    if (readingLAPTIME == LOW && lastKNOPF_LAPTIME == HIGH) {
        pressStart = millis();
        longPressDone = false; // Reset
    }

    // Langer Druck: Extremwerte-Datei löschen
    if (readingLAPTIME == LOW && !longPressDone) {
        if (millis() - pressStart >= 3000) { // 3 Sekunden gedrückt halten
            SD.remove("/Extremwerte.txt");
            displayStartLineExt = 0;  // Anzeige zurücksetzen
            longPressDone = true;     // mehrfaches Auslösen verhindern
        }
    }

    lastKNOPF_LAPTIME = readingLAPTIME;
}
```

```
  //Case 0
  if (menuIndex == 0) {
      int readingLAPTIME = digitalRead(KNOPF_LAPTIME);

      // Fallende Flanke: Button wurde gerade gedrückt
      if (readingLAPTIME == LOW && lastKNOPF_LAPTIME == HIGH) {
          pressStart = millis();
          longPressDone = false; // Reset
      }

      if (readingLAPTIME == LOW && !longPressDone) {
          if (millis() - pressStart >= 3000) {  // 3 Sekunden gedrückt halten
              gpsBuffer = "";
              SD.remove("/tracking.txt");
              // Display-Message
              messageLine1 = "Tracking Datei";
              messageLine2 = "entfernt";
              messageStartTime = millis();
              messageActive = true;
              longPressDone = true;
          }
      }
      lastKNOPF_LAPTIME = readingLAPTIME;
  }

//Kompass
  /*sVector_t mag = compass.readRaw();
  float X = mag.XAxis, Y = mag.YAxis;
  heading = atan2(Y, X) * 180.0 / PI;
  heading += winkelOffset;
```

```
  if (heading < 0) heading += 360.0;
  heading = glaetungMesswert(heading);
  String dir = getDirection(heading);

  String dir;
  if (heading >= 0) {
      dir = getDirection(heading);
  } else {
      dir = "?"; // keine Richtung
  }*/

  //GPS Kompass
  float gpsCourse = 0;
  bool gpsFix = gps.location.isValid() && gps.course.isValid();
  float speedKmh = gps.speed.kmph();
  String dir;

  if (gpsFix && speedKmh > 2.0) {
      gpsCourse = gps.course.deg();
      heading = glaetungMesswert(gpsCourse);
      dir = getDirection(heading);
  } else {
      heading = 0;     // Kompass zeigt Norden
      dir = "----";    // keine Richtung verfügbar
  }

//GPS
  while (gpsSerial.available() > 0) {
      char c = gpsSerial.read();
      gps.encode(c);  // GPS-Daten an TinyGPS++ übergeben
      if (gps.speed.isUpdated()) {
```

```
        lastSpeedUpdate = millis();
    }
    if (gps.altitude.isUpdated()) {
        lastAltitudeUpdate = millis();
    }
    if (gps.date.isUpdated()) {
        lastDateUpdate = millis();
    }
}

// Tracking
if (millis() - lastGpsUpdate > GPS_INTERVAL) {  // jede Sekunde GPS prüfen
    if (gps.location.isUpdated()) {
        double latitude = gps.location.lat();
        double longitude = gps.location.lng();
        // in Puffer speichern
        gpsBuffer += String(latitude, 6) + "," + String(longitude, 6) + "\n";
    }
    lastGpsUpdate = millis();
}

// Puffer Datei schreiben Tracking
if (millis() - lastSaveTime > SAVE_INTERVAL) {
    if (gpsBuffer.length() > 0) {  // Nur speichern, wenn Daten vorhanden sind
        bool exists = SD.exists("/tracking.txt");
        trackFile = SD.open("/tracking.txt", FILE_WRITE);
        if (trackFile) {
            // Wenn Datei neu ist, Kopfzeile schreiben
            if (!exists) {
                trackFile.println("Tracking");
                trackFile.println("Latitude, Longitude");
```

```
            }
            // Puffer auf SD schreiben
            trackFile.print(gpsBuffer);
            trackFile.close();
            gpsBuffer = "";  // Puffer leeren
            lastSaveTime = millis();
        }
    }
}

if (((millis() - lastDateUpdate) < 2000) && (gpsspeed == 1)) {
    gpsdate = 1;
} else {
    gpsdate = 0;
}

gps.speed.kmph();
if((millis()-lastSpeedUpdate) < 2000) {
    gpsspeed = 1;
} else{
    gpsspeed = 0;
}

if((millis()-lastAltitudeUpdate) < 2000) {
    gpsaltitude = 1;
} else{
    gpsaltitude = 0;
}

// GPS Glättung aufrufen
float avgAltitude = 0;
```

```
if (gpsaltitude == 1) {
    avgAltitude = smoothAltitude(gps.altitude.meters());
}

//Extremwerte Speichern
sprintf(dateStr, "%02d.%02d.%04d", gps.date.day(), gps.date.month(), gps.date.year());
stdateStr = String(dateStr);
stdateStr.trim();
DateTag.trim();

if (!stdateStr.equals(DateTag) && gpsspeed == 1 && gpsaltitude == 1 && (!stdateStr.equals("00.00.2000"))) {
    updateExtremwerte();
    DateTag = stdateStr;
    delay(5);
    minHoeheTag = 1e6;
    maxHoeheTag = -1e6;
    maxSpeedTag = 0;
}

if (gpsdate == 1) {
    updatedmaxHoehe = false;
    updatedminHoehe = false;
    updatedmaxSpeed = false;

    // Höhe prüfen
    if (gpsaltitude == 1) {
        altitude = avgAltitude;
        if (altitude > maxHoeheTag) {
            maxHoeheTag = altitude;
            updatedmaxHoehe = true;
        }
```

```
            if (altitude < minHoeheTag) {
                minHoeheTag = altitude;
                updatedminHoehe = true;
            }
        }

        // Geschwindigkeit prüfen
        if (gpsspeed == 1) {
            speed = gps.speed.kmph();
            if (speed > maxSpeedTag) {
                maxSpeedTag = speed;
                updatedmaxSpeed = true;
            }
        }

        if ((updatedmaxSpeed || updatedminHoehe || updatedmaxHoehe) && gpsspeed == 1 && gpsaltitude == 1) {
            updateExtremwerte();
        }
    }



// Menü-Anzeigen
display.clearDisplay();
display.setTextColor(SSD1327_WHITE);

switch (menuIndex) {
    case 0: { // Hauptanzeige
        display.clearDisplay();
        if (!messageActive) {
          display.setTextSize(2);
```

```
      display.setCursor(10, 20);
      if (gpsspeed == 1) {
          display.print(gps.speed.kmph(), 0);
          display.println(" kph");
      }else{
          display.println("---- kph");
      }
      display.setCursor(10, 56);
      if (gpsaltitude == 1) {
          display.print(gps.altitude.meters(), 0);
          display.println(" m");
      }else{
          display.println("---- m");
      }
      display.setCursor(10, 92);
      display.print("Rtg: ");
      display.println(dir);
    }
  if (messageActive) {
      display.setTextSize(1);
      display.setCursor(10, 49);
      display.println(messageLine1);
      display.setCursor(10, 63);
      display.println(messageLine2);
      if (millis() - messageStartTime >= 2000) {
          messageActive = false;
      }
  }
  break;
}
case 1: {//Nebenanzeige 1
```

```
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(10, 56);
    if (gpsspeed == 1) {
      display.print(gps.speed.kmph(), 0);
      display.println(" kph");
    }else{
      display.setCursor(10, 56);
      display.println("---- kph");
    }
    break;
}
case 2: {//Nebenanzeige 2
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(10, 48);
    if (gpsaltitude == 1) {
      // 1. Zeile: Durchschnittshöhe
      display.setCursor(10, 20);
      display.print(avgAltitude, 0);
      display.println(" m");

      // 2. Zeile: Min
      display.setCursor(10, 56);
      display.print("-: ");
      display.print(minAltitude, 0);
      display.println(" m");

      // 3. Zeile: Max
      display.setCursor(10, 92);
      display.print("+: ");
```

```
      display.print(maxAltitude, 0);
      display.println(" m");
  } else {
      display.setCursor(10, 56);
      display.println("---- m");
      }
    break;
}
case 3: {//Nebenanzeige 3
    display.clearDisplay();
    display.setTextSize(1);

    // Kreis
    display.drawCircle(CENTER_X, CENTER_Y, RADIUS, SSD1327_WHITE);

    // Haupt-Himmelsrichtungen
    const char* labels[] = {"N", "O", "S", "W"};
    int winkelH[4] = {0, 90, 180, 270};

    for (int i = 0; i < 4; i++) {
        float rad = (winkelH[i] - heading) * DEG_TO_RAD; // Drehung gegen heading
        int xMark = CENTER_X + cos(rad - HALF_PI) * (RADIUS - 5);
        int yMark = CENTER_Y + sin(rad - HALF_PI) * (RADIUS - 5);
        int xOuter = CENTER_X + cos(rad - HALF_PI) * RADIUS;
        int yOuter = CENTER_Y + sin(rad - HALF_PI) * RADIUS;
        display.drawLine(xMark, yMark, xOuter, yOuter, SSD1327_WHITE);

        // Buchstaben N,O,S,W
        int xText = CENTER_X + cos(rad - HALF_PI) * (RADIUS - 15);
        int yText = CENTER_Y + sin(rad - HALF_PI) * (RADIUS - 15);
        display.setCursor(xText - 3, yText - 3);
```

```
      display.setTextSize(1);
      display.setTextColor(SSD1327_WHITE);
      display.print(labels[i]);
  }

  // Zeiger fix nach oben (Nord)
  int x = CENTER_X;
  int y = CENTER_Y - (RADIUS - 20);
  display.drawLine(CENTER_X, CENTER_Y, x, y, SSD1327_WHITE);

  // Gradzahl im Kompass unterhalb des Zeigers
  //int gradX = CENTER_X - 20; // mittig
  int gradY = CENTER_Y - (RADIUS - 75); // direkt unter Zeiger
  //display.setCursor(gradX, gradY);
  /*display.print((int)heading);
  display.println(" Grad");
  */

if (gpsFix && speedKmh > 2.0) {
    String gradText= String((int)heading) + " Grad";
    int textWidth = gradText.length() *6;
    int gradX = CENTER_X - (textWidth /2);
    display.setCursor(gradX, gradY);
    display.print(gradText);
    //display.print((int)heading);
    //display.println(" Grad");
} else {
    String gradText = "----";
    int textWidth = gradText.length() *6;
    int gradX = CENTER_X - (textWidth /2);
    display.setCursor(gradX, gradY);
```

```
        display.println(gradText);
    }
    break;
}
case 4: {//Nebenanzeige 4
    display.clearDisplay();
    display.setTextSize(1);
    // Aktuelle Runde
    if (timingActive) {
        unsigned long currentLap = millis() - lapStart;
        unsigned int minutes = currentLap / 60000;
        unsigned int seconds = (currentLap % 60000) / 1000;
        unsigned int hundredths = (currentLap % 1000) / 10;
        display.setCursor(10, 20);
        display.print("Aktuell: ");
        display.print(minutes);
        display.print(":");
        if (seconds < 10) display.print("0");
        display.print(seconds);
        display.print(".");
        if (hundredths < 10) display.print("0");
        display.println(hundredths);
    } else {
        display.setCursor(10, 20);
        display.println("Messung gestoppt");
    }

    // Letzte Runde
    display.setCursor(10, 40);
    display.print("Letzte: ");
    if (valuesAvailable) {
```

```
  unsigned int minutes = lastLap / 60000;
  unsigned int seconds = (lastLap % 60000) / 1000;
  unsigned int hundredths = (lastLap % 1000) / 10;
  display.print(minutes);
  display.print(":");
  if (seconds < 10) display.print("0");
  display.print(seconds);
  display.print(".");
  if (hundredths < 10) display.print("0");
  display.println(hundredths);
} else {
    display.println("----");
}

// Beste Runde
display.setCursor(10, 55);
display.print("Beste: ");
if (valuesAvailable) {
  unsigned int minutes = bestLap / 60000;
  unsigned int seconds = (bestLap % 60000) / 1000;
  unsigned int hundredths = (bestLap % 1000) / 10;
  display.print(minutes);
  display.print(":");
  if (seconds < 10) display.print("0");
  display.print(seconds);
  display.print(".");
  if (hundredths < 10) display.print("0");
  display.println(hundredths);
} else {
    display.println("----");
}
```

```
  // Rundennummer
  display.setCursor(10, 70);
  display.print("Runde Nr.: ");
  if (valuesAvailable) {
      display.println(lapNumber);
  } else {
      display.println("----");
  }
  break;
}
case 5: {//Nebenanzeige 5
    display.clearDisplay();
    display.setTextSize(1);
    lapFile = SD.open("/Laptime.txt", FILE_READ);
    if (lapFile) {
      display.setCursor(0, 0);
      if (lapFile.size() == 0) {  // Datei leer
          display.setCursor(10, 49);
          display.println("Laptime Speicher ");
          display.setCursor(10, 63);
          display.println("leer");
      } else {
          int lineIndex = 0;
          int yPos = 0;
          while (lapFile.available() && yPos < MAX_LINES_DISPLAY) {
              String line = lapFile.readStringUntil('\n');
              if (lineIndex >= displayStartLine) {
                  display.setCursor(0, yPos * 10);
                  display.println(line);
                  yPos++;
```

```
                }
                lineIndex++;
            }
        }
        lapFile.close();
    } else { // Datei existiert nicht
        display.setCursor(10, 49);
        display.println("Laptime Speicher ");
        display.setCursor(10, 63);
        display.println("leer");
    }
    break;
}
case 6: { // Nebenanzeige 6
    display.clearDisplay();
    display.setTextSize(1);

    extFile = SD.open("/Extremwerte.txt", FILE_READ);
    if (extFile) {
        display.setCursor(0, 0);

        if (extFile.size() == 0) { // Datei leer
            display.setCursor(10, 49);
            display.println("Extremwert Speicher");
            display.setCursor(10, 63);
            display.println("leer");
        } else {
            const int bufferSize = 64; //Platz für längste Zeile
            char lineBuffer[bufferSize];
            int lineIndex = 0;
            int yPos = 0;
```

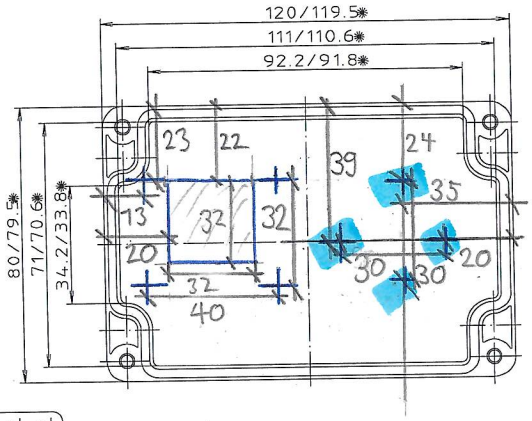
```
        int c;
        int charPos = 0;
        while ((c = extFile.read()) != -1 && yPos < MAX_LINES_DISPLAY) {
            if (c == '\n' || charPos >= bufferSize - 1) {
                lineBuffer[charPos] = '\0'; // Null-Terminator
                if (lineIndex >= displayStartLineExt) {
                    display.setCursor(0, yPos * 10);
                    display.println(lineBuffer);
                    yPos++;
                }
                charPos = 0; // Puffer zurücksetzen
                lineIndex++;
            } else {
                lineBuffer[charPos++] = (char)c;
            }
        }
        // letzte Zeile, falls keine \n am Ende
        if (charPos > 0 && yPos < MAX_LINES_DISPLAY && lineIndex >= displayStartLineExt) {
            lineBuffer[charPos] = '\0';
            display.setCursor(0, yPos * 10);
            display.println(lineBuffer);
        }
    }

    extFile.close();
} else { // Datei existiert nicht
    display.setCursor(10, 49);
    display.println("Extremwert Speicher");
    display.setCursor(10, 63);
    display.println("leer");
```

```
      }
      break;
  }
    case 7: { // WLAN-Anzeige
        display.clearDisplay();
        display.setTextSize(1);
        display.setCursor(10, 20);
        display.print("WLAN: ");
        display.println(wlanAn ? "An" : "Aus");
        display.setCursor(10, 50);
        if (wlanAn) {
            display.print("IP: ");
            display.println(WiFi.softAPIP());
        } else {
            display.print("IP: ");
            display.println("- - - -");
        }
        display.setCursor(10, 80);
        display.print("SSID: ");
        display.println(ssid);
        break;
    }
}
display.display();
if (wlanAn) {
  server.handleClient();  // HTTP-Anfragen verarbeiten
  }
  //debug
  static unsigned long jetzt = 0; // einmal initialisieren
  if (millis() - jetzt > 2000) {  // prüfen, ob 2 Sekunden vergangen sind
      Serial.print("speed: ");
```

```
        Serial.println(gpsspeed);
        Serial.print("altitude: ");
        Serial.println(gpsaltitude);
        Serial.print("date: ");
        Serial.println(gpsdate);
        Serial.print("SD Datum: ");
        Serial.println(DateTagSD);
        Serial.print("Datum: ");
        Serial.println(DateTag);
        Serial.print("max H: ");
        Serial.println(maxHoeheTag);
        Serial.print("min H: ");
        Serial.println(minHoeheTag);
        Serial.print("max S: ");
        Serial.println(maxSpeedTag);
        jetzt = millis();  // Timer zurücksetzen
    }
    //Serial.print("Zeit: ");
    //Serial.println(millis()-lastDateUpdate);

}


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//void loop1
void loop1(){

}
```

= 12,5 Ø

= 16,2 Ø

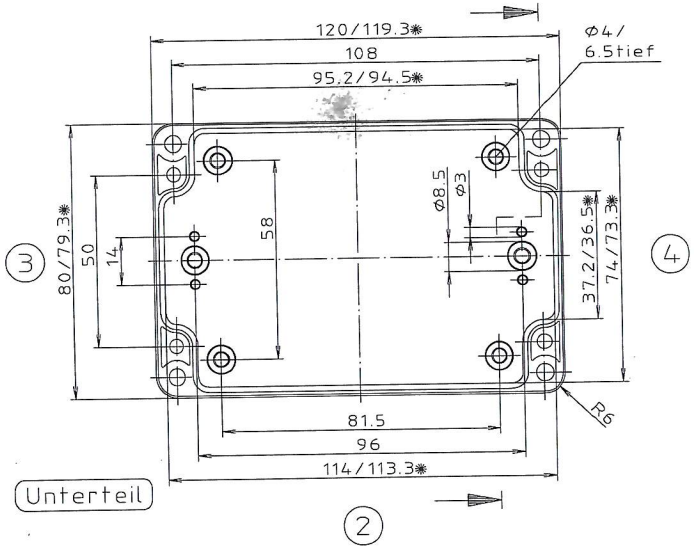


Deckel

1

* = Maß durch Formkonizität nach unten verringert.
Freimaß - Toleranz nach DIN 16901-130

Zylinderschraube M4 x 24/10
ähnl. DIN EN ISO 7045

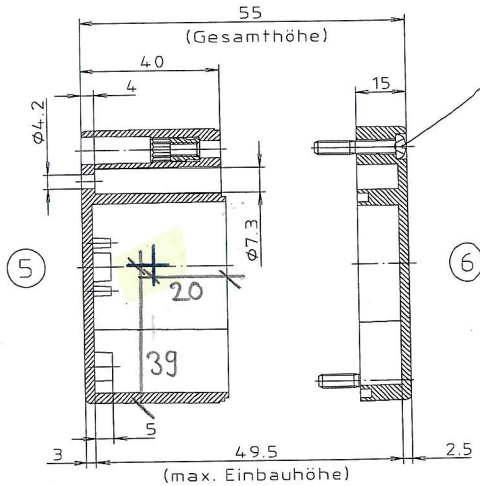


Unterteil

2

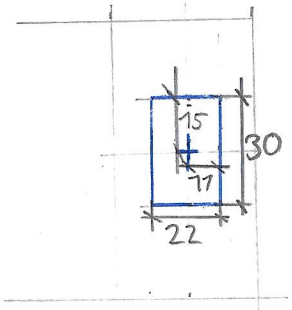
3

4



5

6



M = 1:0,45

ABS-Gehäuse können durch eine andere Schwindung ca. 0,1-0,25% größer sein als gezeichnete PC-Gehäuse

"Schutzvermerk nach DIN 34 beachten"

Konvertierung nicht zulässig!

Maßstab: 1:1

Zchn.Nr: 2K 215.028.03

Artikel: M/T 215 mit Nocken
Katalogzeichnung

3601	22.04.02	BOPLA A Phoenix Mecano Company
3499	15.02.2000	DISK: 2 501 102
Ae.Nr:	Datum:	Datum: 25.10.93 OK

Gebrauchsanleitung

Einleitung

Das Motorrad Dashboard bietet eine Vielzahl an Funktionen, die sowohl der Information als auch der Analyse des Fahrverhaltens dienen. Es zeigt in Echtzeit die aktuelle Geschwindigkeit, die Höhe sowie die Fahrtrichtung an. Darüber hinaus verfügt es über eine Lap-Time Funktion, mit der die gefahrenen Rundenzeiten erfasst und gespeichert werden. Diese können später bequem eingesehen und miteinander verglichen werden.

Zusätzlich speichert das System die täglichen Maximal- und Minimalwerte der Höhe sowie die höchste erreichte Geschwindigkeit. Eine integrierte Tracking-Funktion ermöglicht es, nachzuvollziehen, welche Strecken während einer Fahrt zurückgelegt wurden. Alle gespeicherten Daten, einschliesslich Tracking, Rundenzeiten und Extremwerte, können direkt über ein verbundenes Smartphone ausgelesen werden.

Diese Dokumentation beschreibt die Inbetriebnahme, Nutzung und Wartung des Systems und bietet einen umfassenden Überblick über die Funktionsweise des Motorrad Dashboards.

Inbetriebnahme

1. **Vorbereitung:**

    - **Dashboard montieren:** Befestigen Sie das Motorrad Dashboard am Motorrad, damit es festsitzt und die Sicht auf die Pflichtanzeigen nicht verdeckt wird.

    - **Lap-Time-Taster positionieren:** Montieren Sie den Lap-Time Taster in der Nähe des linken Griffes, sodass Sie ihn bequem mit dem Daumen betätigen können.

    - **Verbindung des Lap-Time-Tasters:** Stecken Sie das Kabel des Lap-Time Tasters über den Rundstecker in die Rundsteckhülse am Dashboard.

    - **Batterie anschliessen:** Verbinden Sie die Plus- und Minus- Hülsen des Batteriekabels mit den vorgesehenen Steckern. Achten Sie darauf, dass der Minuspol über den Rundstecker und der Pluspol über den Flachzungenstecker angeschlossen wird.

2. **Überprüfung:**

    - Achten Sie darauf, dass die Batteriespannung 12 V beträgt. Falls eine andere Spannung vorliegt, muss diese über das Potentiometer am DC-DC-Wandler eingestellt werden, sodass die Sekundärseite 5 V nicht überschreitet.
    - Schalten Sie das Dashboard über den Hauptschalter ein. Es sollte nun ein Startbildschirm auf dem Display erscheinen. Bleibt das Display dunkel, prüfen Sie bitte, ob eine SD-Karte eingesetzt ist. Das Dashboard unterstützt SD-Karten von 2 GB bis 32 GB, formatiert im FAT32-Format.

Bedienung des Dashboards

1. **Startbildschirm:**

    Drücken Sie den linken und rechten Taster miteinander für 3 Sekunden. Dann kommen Sie zum Hauptmenüpunkt.

2. **Menüfunktionen:**

    Das Motorrad Dashboard verfügt über eine Hauptanzeige und mehrere Nebenanzeigen, die unterschiedliche Informationen darstellen:

    Die Hauptanzeige zeigt die aktuelle Geschwindigkeit in km/h, die aktuelle Höhe in Metern. Die Fahrtrichtung wird Ihnen angezeigt, sofern sich das Motorrad schneller als 2 km/h bewegt.

    Die Nebenanzeige 1 gibt die aktuelle Geschwindigkeit in km/h wieder, während die Nebenanzeige 2 die aktuelle Höhe sowie die tiefste und höchste Höhe seit dem Einschalten des Dashboards in Meter über Meer anzeigt.

    In der Nebenanzeige 3 wird ein digitaler Kompass dargestellt. Die Nebenanzeige 4 dient zur Nutzung der Lap-Time Funktion, während in der Nebenanzeige 5 die gespeicherten Rundenzeiten abgerufen werden können.

    Die Nebenanzeige 6 zeigt die Extremwerte an. Schliesslich ermöglicht die Nebenanzeige 7 die Steuerung des WLAN-Moduls, um die Verbindung ein- oder auszuschalten.

3. Steuerung:

Anzeige	Taster links	Taster rechts	Taster rauf	Taster runter	Taster Lap- Time
Hauptanzeige	Wechselt zu Nebenanzeige 6	Wechselt zu Nebenanzeige 1	----	Wechselt zu Nebenanzeige 7	3 Sekunden Druck löscht Tracking Datei
Nebenanzeige 1	Wechselt zu Hauptanzeige	Wechselt zu Nebenanzeige 2	----	----	----
Nebenanzeige 2	Wechselt zu Nebenanzeige 1	Wechselt zu Nebenanzeige 3	----	----	----
Nebenanzeige 3	Wechselt zu Nebenanzeige 2	Wechselt zu Nebenanzeige 4	---	----	----
Nebenanzeige 4	Wechselt zu Nebenanzeige 3	Wechselt zu Nebenanzeige 5	----	----	Kurzer Druck startet/ setzt neue Runde, 2 Sekunden druck stoppt die Messung
Nebenanzeige 5	Wechselt zu Nebenanzeige 4	Wechselt zu Nebenanzeige 6	Scrollt nach oben	Scrollt nach unten	3 Sekunden druck löscht Laptime Datei
Nebenanzeige 6	Wechselt zu Nebenanzeige 5	Wechselt zu Hauptanzeige	Scrollt nach oben	Scrollt nach unten	3 Sekunden druck löscht Extremwerte Datei
Nebenanzeige 7	----	----	Wechselt zu Haupanzeige	----	Schaltet WLAN ein/aus

4. **WLAN-Verbindung:**
    - WLAN einschalten
    - Mit dem WLAN PicoW_Hotspot verbinden
    - Passwort eingeben: 12345678
    - Im Browser die angezeigte IP-Adresse eingeben
    - Gewünschter Tab öffnen
5. **Tracking:**
    - Mit dem WLAN PicoW_Hotspot verbinden
    - Tracking Tab wählen
    - Alle Datenpunkte kopieren, mit Tracking, Latitude und Longitude
    - Auf die Webseite https://www.gpsvisualizer.com/convert_input gehen
    - Output Format auf GPX setzen
    - Datenpunkte einfügen
    - Konvertieren
    - Datei Downloaden
    - Auf die Webseite https://www.google.com/maps/d/u/0/ gehen
    - Neue Karte erstellen
    - Importieren anklicken
    - Die vorhin heruntergeladene GPX-Datei auswählen
6. **Ausschalten:**
    - Zum Ausschalten kippen Sie den Hauptschalter auf 0.
7. **Demontage:**
    - Schalten Sie das Dashboard aus.
    - Trennen Sie das Batteriekabel vom Dashboard.
    - Trennen Sie das Kabel des Lap-Time-Tasters vom Dashboard.
    - Entfernen Sie anschliessend das Motorrad-Dashboard vom Fahrzeug.

Fehlersuche

Problem	Mögliche Ursache	Lösung
Keine Funktion 1	Batterie leer, Batterieanschluss fehlt, oder Hauptschalter nicht eingeschaltet	Prüfen Sie die Stromversorgung.
Keine Funktion 2	Überlast oder Kurzschluss	Überprüfen Sie die Flachsicherung
Keine Funktion 3	Falsche Spannung	Überprüfen Sie die Batterie und die Sekundärspannung
Display leuchten nicht	Display defekt oder Stecker ist rausgefallen	Überprüfen Sie das Displaykabel
Keine Werte bei Höhe, Geschwindigkeit oder Richtungsanzeige	Kein GPS-Empfang	Gehen Sie an einen Ort mit freier Sicht zum Himmel
Taster reagieren nicht	Taster Anschluss defekt oder Verbindungsunterbruch	Überprüfen Sie die Taster Klemmen
Daten werden nicht gespeichert	SD- Karte ist voll oder defekt	Überprüfen Sie die SD-Karte
System reagiert nicht	Das System hat einen Fehler	Ausschalten, 10 Sekunden warten, Einschalten

Sicherheitshinweise

- Berühren Sie niemals die Platine oder andere leitende Komponenten.
- Halten Sie das Dashboard von Feuchtigkeit und extremen Temperaturen geschützt.
- Verwenden Sie nur eine 2A Flachsicherung
- Stellen Sie sicher, dass alle Lötstellen sauber und frei von Kurzschlüssen sind.
- Lassen Sie das Dashboard nicht zu lange eingeschaltet, wenn der Motor nicht läuft.

Projekt: Motorrad Dashboard

Statusbericht: KW38

Projektleiter	**Projektziele**	**Verteiler**
• Jonas Kuster	Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.	• Urs Gloggner

Gesamt-beurteilung	**Projektverlauf**	**Projektklima**	**Termine**	**Risiken**	**Ressourcen**
	■ □ □	■ □ □	■ □ □	■ □ □	■ □ □
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand

- Projektplanung abgeschlossen
- Systemarchitektur Definiert
- Bereit für das Gespräch am 22.09.2025

Was läuft gut?

- Terminplan eingehalten

Was läuft nicht gut?

- -----

Geplante nächste Schritte / getroffene Massnahmen

- Teilprogramme schreiben und Komponenten Tests durchführen.

Vorlage von: Stefan Thöni, Josef Räber

Projekt: Motorrad Dashboard

Statusbericht: KW39

Projektleiter	Projektziele	Verteiler
• Jonas Kuster	Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.	• Urs Gloggner

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
					
Tendenz					

Aktueller Projektstand

- Komponenten getestet
- Komponenten versuchsweise zusammen gebaut

Was läuft gut?

- Komponenten Funktionieren wie gewünscht.

Was läuft nicht gut?

- Zeitmanagement

Geplante nächste Schritte / getroffene Massnahmen

- Hauptprogramm schreiben
- Die Uhrzeit während dem Programmieren besser im Auge behalten.

Vorlage von: Stefan Thöni, Josef Räber

Projekt: Motorrad Dashboard

Statusbericht: KW40

Projektleiter	Projektziele	Verteiler
• Jonas Kuster	Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.	• Urs Gloggner

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
					
Tendenz					

Aktueller Projektstand

- Hauptprogramm schreiben
- Hauptprogramm Testen und verbessern

Was läuft gut?

- Grossteil des Programms ist fertig und funktioniert

Was läuft nicht gut?

- Gespeicherte Dateien auf der SD- Karte werden zum Teil beschädigt und ich weiss noch nicht warum.

Geplante nächste Schritte / getroffene Massnahmen

- Hauptprogramm fertig schreiben, Tests abschliessen und den Zusammenbau starten.
- Herausfinden weshalb die Dateien auf der SD- Karte zum teil beschädigt sind.

Vorlage von: Stefan Thöni, Josef Räber

Projekt: Motorrad Dashboard

Statusbericht: KW41

Projektleiter	Projektziele	Verteiler
• Jonas Kuster	Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.	• Urs Gloggner

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
					
Tendenz					

Aktueller Projektstand

- Programm getestet & Korrigiert
- Dashboard Zusammengebaut und am Motorrad montiert

Was läuft gut?

- Test alle erfolgreich mit Laboraufbau des Dashboards
- Zusammenbau lief gut

Was läuft nicht gut?

- Evtl. Probleme mit dem Kompass, da zu viel Metall am Motorrad (noch nicht richtig getestet)

Geplante nächste Schritte / getroffene Massnahmen

- Test der Funktionen bei betrieb auf dem Motorrad
- Dokumentation auf den aktuellen Arbeitsstand bringen

Vorlage von: Stefan Thöni, Josef Räber

Projekt: Motorrad Dashboard

Statusbericht: KW42

Projektleiter	Projektziele	Verteiler
• Jonas Kuster	Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.	• Urs Gloggner

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
					
Tendenz					

Aktueller Projektstand

- Dashboard vollständig getestet
- Alle Verbesserungen abgeschlossen
- An Motorrad montiert und getestet

Was läuft gut?

- Das Dashboard funktioniert planmässig

Was läuft nicht gut?

- Derzeit arbeite ich fast ausschliesslich an der Diplomarbeit, sodass für andere Aufgaben nur sehr wenig Zeit bleibt.

Geplante nächste Schritte / getroffene Massnahmen

- Dokumentation fertig Schreiben
- Dokumentation gegenlesen und korrigieren lassen

Vorlage von: Stefan Thöni, Josef Räber

Projekt: Motorrad Dashboard

Statusbericht: KW43

Projektleiter	Projektziele	Verteiler
• Jonas Kuster	Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.	• Urs Gloggner

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
					
Tendenz					

Aktueller Projektstand

- Dokumentation fertig geschrieben

Was läuft gut?

- Die Dokumentation ist fertig zum Gegenlesen.

Was läuft nicht gut?

Geplante nächste Schritte / getroffene Massnahmen

- Dokumentation gegenlesen und korrigieren lassen
- Text Vorbereitung Online Publikation

Vorlage von: Stefan Thöni, Josef Räber

Projekt: Motorrad Dashboard

Statusbericht: KW44

Projektleiter	Projektziele	Verteiler
• Jonas Kuster	Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.	• Urs Gloggner

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
					
Tendenz					

Aktueller Projektstand

- Korrektur der Dokumentation zurück bekommen
- Text für Online Publikation vorbereitet

Was läuft gut?

- Korrektur der Dokumentation rechtzeitig zurück bekommen

Was läuft nicht gut?

Geplante nächste Schritte / getroffene Massnahmen

- Dokumentation der Korrektur anpassen und Fertigstellen.
- Dokumente für den Anhang der Dokumentation fertigstellen und sammeln.

Vorlage von: Stefan Thöni, Josef Räber

Projekt: Motorrad Dashboard

Statusbericht: KW45

Projektleiter	**Projektziele**	**Verteiler**
• Jonas Kuster	Entwicklung eines modularen Motorrad-Dashboards, das Geschwindigkeits-, Höhen- und Richtungsdaten per GPS und Kompass erfasst, speichert und benutzerfreundlich darstellt.	• Urs Gloggner

Gesamt-beurteilung	**Projektverlauf**	**Projektklima**	**Termine**	**Risiken**	**Ressourcen**
	■ □ □	■ □ □	■ □ □	■ □ □	■ □ □
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand

- Dokumentation Fertiggestellt
- Anhang Zusammengestellt

Was läuft gut?

- Bereit zur Abgabe

Was läuft nicht gut?

Geplante nächste Schritte / getroffene Massnahmen

- Abgabe der Dokumentation
- Vorbereiten der Präsentation

Vorlage von: Stefan Thöni, Josef Räber