

Abschlussarbeit

Gesamtprojekt
Datenlogger für Motorräder

Entwicklung der Schaltung

Dominic FROSTL 4AFELC-5

Betreuer: Ing. Marc Prantl

Entwicklung der Software

Manuel GSCHWANDTNER 4AFELC-9

Betreuer: Ing. Reinhard Rzepa

ausgeführt im Schuljahr 2021/22

Abgabevermerk:

Datum: 15. April 2022 übernommen von:

Eidesstattliche Erklärung:

Ich erkläre an Eides statt, dass ich die vorliegende Abschlussarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

St. Pölten, 15. April 2022

Dominic FROSTL

Manuel GSCHWANDTNER

Abschlussarbeit DOKUMENTATION

Namen der Verfasser/innen	Dominic Frostl / Manuel Gschwandtner
Jahrgang / Klasse Schuljahr	4AFELC / 2021-2022
Thema der Abschlussarbeit	Datenlogger für Motorräder

Aufgabenstellung	<p>Als Abschlussprojekt soll ein Datenlogger entworfen werden, mit welchem Fahrtdaten bei einem Motorrad aufgezeichnet werden können. Dafür soll ein Gerät mit Rücksicht auf Kompaktheit und Witterungsbeständigkeit sowie Erschütterungsresistenz geplant werden. Weiters sollen die gesammelten Daten über eine drahtlose Verbindung an einem Webserver hochgeladen werden können. Auf diesem soll es möglich sein die gesammelten Daten anhand einer Karte, welche die gefahrene Route anzeigt, intuitiv zu betrachten.</p>
Realisierung	<p>Als Herzstück des Projekts wurde der ESP32 eingesetzt, als Sensoren wurden ein Gyroskop, der MPU6050, ein GPS-Modul, das BN-220 verwendet. Zum Speichern der Daten wurde ein SD-Karten-Modul zusammen mit einer 8-GB-SD-Karte verwendet. Ein mehrteiliges Gehäuse wurde aus dem Kunststoff PETG mit dem Ender 3 – 3D Drucker erzeugt. Um das zusammengesetzte Gehäuse abzudichten, wurde an den ausschlaggebenden Stellen Moosgummi sowie ein O-Ring eingesetzt. Für die Montage am Motorrad wurde das gängige System der Sony GoPro Kamera verwendet.</p> <p>Zur Umsetzung des Servers wurde Node.js in Kombination mit einer MySQL-Datenbank verwendet. Jeder Teil des Servers wurde für bessere Verwaltungsmöglichkeiten sowie, einfache Installation in einen Docker-Container verpackt.</p>

Ergebnisse	Der Datenlogger sowie die Website sind voll funktionsfähig. Der Datenlogger kann auch bei Schlechtwetter ohne Bedenken eingesetzt werden. Daten können über ein vom Nutzer konfiguriertes Netzwerk hochgeladen werden. Diese Daten können nach dem Upload auf der Website mithilfe einer Karte und Diagrammen visualisiert werden.
------------	--

Typische Grafik, Foto etc. (mit Erläuterung)	 <p>In der oben abgebildeten Grafik sieht man den fertigen Datenlogger montiert und einsatzbereit.</p>
---	---

Teilnahme an Wettbewerben, Auszeichnungen	
---	--

Möglichkeiten der Einsichtnahme in die Arbeit	Die Arbeit kann in der HTBLuVA St. Pölten, Waldstraße 3, 3100 St. Pölten eingesehen werden.
---	---

Approbation (Datum / Unterschrift)	Prüfer/in	AV Dipl.-Ing. W. U. KURAN Abteilungsvorstand
---------------------------------------	-----------	---


PROJECT THESIS DOCUMENTATION

Author(s)	Dominic Frostl / Manuel Gschwandtner
Form Academic year	4AFELC / 2021-2022
Topic	Datalogger for motorcycles

Assignment of tasks (conceptual formulation/job definition)	<p>This graduation project is about the design and development of a datalogger meant for motorized vehicles with a focus on motorcycles. The goals on the hardware side are building a compact, shock resistant device which can withstand various weather conditions. The collected data is to be wirelessly transmitted to a remote Server, which stores and visualizes the collected data.</p>
--	---

Implementation	<p>An ESP32 was used for the purpose of recording, storing and uploading the sensor data. The needed data is provided by a GPS-module (BN-220) and a gyroscope (MPU6050). An SD-module in combination with an 8 GB SD-card stores the collected data. The housing was printed with the help of a 3D printer (Ender 3). Weatherproofing was accomplished thanks to o-rings and foam rubber at critical areas. A GoPro camera mount was chosen as a mounting solution due to its great availability.</p> <p>The server uses Node.js and MySQL to provide the needed functionality. These two services are each contained in a Docker-Container for better management and to simplify the installation process.</p>
----------------	--

Results	Both the website and the data logger are fully operational. The usage at bad weather conditions is possible without any problems. Data can be uploaded after connecting to a user defined WiFi-network, which is then viewable in the Web-interface.
---------	--

Illustrative graph, photo (incl. explanation)	 <p>In the picture above the device is fully assembled, ready for use and mounted on a motorcycle.</p>
Participation in competitions, Awards	

Accessibility of project thesis	The project thesis can be viewed at the HTL St. Pölten, Waldstraße 3, 3100 St. Pölten.
---------------------------------	--

Approval (Date / Sign)	Examiner	AV Dipl.-Ing. W. U. KURAN Head of Department
------------------------	----------	---

Inhaltsverzeichnis

1	Einleitung.....	1
1.1	Ausgangslage	1
1.2	Zielsetzung.....	1
1.2.1	Individuelle Zielsetzung Dominic Frostl.....	1
1.3	Individuelle Zielsetzung Manuel Gschwandtner	1
1.4	Terminplanung	2
1.4.1	Individueller Terminplan Dominic Frostl.....	2
1.4.2	Individueller Terminplan Manuel Gschwandtner	3
2	Konzept des Datenloggers	4
2.1	Generelles.....	4
2.2	Sensoren.....	4
2.3	Mobilität.....	4
2.4	Tracking	5
2.5	Datenverarbeitung.....	5
2.6	Schutz der Elektronik	5
2.7	Montage	6
2.8	Benutzerfreundlichkeit.....	6
3	Technische Umsetzung	6
3.1	Prozessor	6
3.2	Datenspeicher	7
3.3	Gyroskop.....	7
3.4	GPS.....	8
3.5	Energieversorgung.....	8
3.6	Akku	9
3.7	Gehäuse.....	9
4	Prototyp I	10
4.1	Aufbau.....	10
4.2	Verbinden des GPS-Moduls.....	11
4.3	Verbinden des Gyroskops	11
4.4	Verbinden des SD-Karten-Moduls.....	11
4.5	Entwicklung der ersten Testsoftware.....	12
5	Prototyp II	12

5.1	Aufbau	12
5.2	Akku.....	13
5.3	Spannungswandler	13
5.4	Überwachung des Ladezustands.....	13
5.4.1	Berechnungen.....	14
5.5	Laden des Akkus	15
5.6	Erweiterung der Testsoftware	15
5.6.1	Ladezustand.....	15
5.6.2	Dateierstellung	16
5.6.3	Datenverarbeitung.....	17
5.6.4	Benötigte Speichermenge	17
5.7	Programmablauf	18
5.8	Weitere Funktionen.....	20
5.8.1	Programmstatus.....	20
5.8.2	Verbindung mit dem Webserver.....	20
5.8.3	Konfiguration des Ziel-Netzwerkes.....	21
5.8.4	Verarbeitung der Zugangsdaten.....	21
5.8.5	Upload-Vorgang	22
6	Entwicklung der Leiterplatte	23
6.1	Analyse der Ladeplatine	23
6.1.1	TP4056.....	24
6.1.2	DW01-P.....	24
6.2	Analyse des SD-Karten-Moduls.....	26
6.2.1	Nicht benötigte Komponenten	27
6.2.2	CD74HC4050M.....	27
6.3	Die Hauptschaltung	28
6.4	Die entflochtene Platine	29
6.4.1	Anmerkungen.....	29
6.4.2	Befestigung im Gehäuse	30
6.4.3	Fehlerkorrekturen	30
7	Entwicklung des Gehäuses.....	31
7.1	Schutz vor Umwelteinflüssen.....	31
7.2	Entworfenene Teile.....	32
7.3	Zusammenbau.....	33
7.4	Montage am Motorrad	34

8	Einleitung Server	35
9	Prototyp der Website	35
10	Backend	36
10.1	VPS.....	36
10.2	Caddy2.....	36
10.3	Docker.....	37
10.4	Node.js.....	37
10.5	Services, Libraries und Frameworks.....	38
10.5.1	Express	38
10.5.2	Prisma	38
10.5.3	Auth0.....	38
10.6	Datenbankaufbau.....	39
10.7	Private-API.....	39
10.7.1	api/upload.....	39
10.7.2	POST.....	39
10.7.3	api/key	39
10.7.4	GET	39
10.7.5	POST.....	39
10.7.6	DELETE	40
10.7.7	PATCH	40
10.7.8	api/tours.....	40
10.7.9	GET	40
10.7.10	PATCH	40
10.7.11	DELETE	41
10.8	Public-API	41
11	Frontend	41
11.1	Libraries und Frameworks.....	41
11.1.1	MDBootstrap	41
11.1.2	Font Awesome	42
11.1.3	Google Maps API	42
11.1.4	SweetAlert2	42
11.1.5	Chart.js.....	43
11.1.6	Chroma.js.....	43
11.2	Seiten und Elemente.....	43
11.2.1	Allgemeine Information.....	43

11.2.2	Navigation	44
11.2.3	Routen	44
11.2.4	Geräte	45
11.2.5	Manuelles Hochladen	45
11.2.6	Nachrichten.....	46
11.2.7	Karte und Visualisierung	46
12	Entwicklungswerkzeuge.....	48
12.1	Node Modules	48
12.1.1	Gulp	48
12.1.2	Browseroptimierung	49
12.1.3	SCSS	50
12.1.4	ESLint und JSHint.....	50
12.1.5	Morgan.....	51
12.1.6	Nodemon	51
12.1.7	Jest	51
12.2	GUI-Anwendungen.....	52
12.2.1	Prisma Studio	52
12.2.2	Portainer	52
12.2.3	RESTEDClient	53
12.2.4	Visual Studio Code	53
12.3	Installation & Update Script	53
13	Ergebnisse von Tests	54
13.1	Ladezeit der Karte	54
13.2	API-Request Ladezeit.....	54
13.3	Erstellen der Docker Container	54
14	Zukunftspläne	55
14.1	Log-in	55
14.2	Map Visualisierung	55
14.3	Desktop App.....	55
15	Ergebnisse	56
16	Betriebswirtschaftliche Kalkulation	57
16.1	Entwicklungskosten	57
16.1.1	Materialkosten.....	57
16.2	Produktionskosten	57
16.3	Verkaufspreis	57

16.4	Break-Even-Point.....	58
17	Literaturverzeichnis	59
18	Verzeichnis der Abbildungen, Tabellen und Begriffe	61
18.1	Abbildungsverzeichnis	61
18.2	Tabellenverzeichnis	62
18.3	Begriffsverzeichnis und Abkürzungen Server	63
18.3.1	Allgemeine Begriffe	63
18.3.2	Backend	63
18.3.3	Docker	63
18.3.4	Prisma	63
18.3.5	API.....	63
18.3.6	Frontend	63
18.3.7	Entwicklungswerkzeuge	63
19	Begleitprotokoll gemäß § 9 Abs. 2 PrO	64
20	Diplomandenseminare.....	66
20.1	Protokoll zum 1. Diplomandenseminar.....	66
20.2	Protokoll zum 2. Diplomandenseminar.....	67
20.3	Protokoll zum 3. Diplomandenseminar.....	68
20.4	Protokoll zum 3. Diplomandenseminar.....	69
21	Anhang	69

0 Vorwort

Mit der Entwicklung dieses Projektes wollten wir das während der Schullaufbahn erlernte Wissen einsetzen und mit persönlichen Interessen verbinden.

Danksagung

Wir, Dominic Frostl und Manuel Gschwandtner, möchten uns bei allen Lehrern bedanken, die uns bei der Entwicklung dieses Projektes unterstützt, sowie beim Verfassen dieser Arbeit geholfen haben.

Weiters möchte ich, Dominic Frostl, mich bei meiner Freundin und deren Familie bedanken, welche mich im Laufe meiner Schullaufbahn, während des Abschlussjahres und beim Verfassen dieser Arbeit ebenfalls unterstützt haben.

Ich, Manuel Gschwandtner, möchte mich ebenfalls bei meiner Familie und meinen Freunden für die Unterstützung während meiner Schullaufbahn bedanken.

1 Einleitung

1.1 Ausgangslage

Motorradfahren ist für viele ein Hobby, aber auch gleichzeitig ein Sport, bei dem es darum geht, seine eigenen Fähigkeiten stetig zu verbessern. Um die Veränderung besser erkennen zu können, sind Fahrtdaten wie Neigungswinkel, Beschleunigungskräfte und Durchschnittsgeschwindigkeit notwendig. Um diese Daten und die gefahrene Route aufzuzeichnen werden ein Datenlogger sowie eine passende Website, welche die gesammelten Daten klar visualisiert, benötigt.

1.2 Zielsetzung

Das in dieser Abschlussarbeit vorgestellte Gerät erleichtert die Datenaufnahme und ermöglicht dadurch vergangene Fahrten einzusehen, um den eigenen Fahrstil anzupassen. Mit der dazu entwickelten Website können die gesammelten Daten grafisch dargestellt werden.

1.2.1 Individuelle Zielsetzung Dominic Frostl

Der gesamte Datenlogger wird von Grund auf entwickelt. Dies beinhaltet die Programmierung der Software, das Zeichnen des Schaltplans sowie das Entflechten und Bestücken der daraus entstehenden Leiterplatte. Auch die Planung eines geeigneten Gehäuses und dessen Montage am Lenker eines Motorrads.

1.3 Individuelle Zielsetzung Manuel Gschwandtner

Die Website soll übersichtlich gestaltet werden und in den meistverbreiteten Webbrowsern verwendbar sein. Die Nutzung der Seite mithilfe von Mobilgeräten soll möglich sein. Daten eines Nutzers sollen nur von diesen bearbeitbar und anschaulich sein.

1.4 Terminplanung

Meilenstein	Termin
Konzept erstellt	6. September 2021
Bauteile lagernd	4. Oktober 2021
Prototyp funktionsfähig	6. Dezember 2021
Testsoftware funktionstauglich	27. Dezember 2021
Gerät funktionstauglich	7. Februar 2022
Website online	28. März 2022

Tabelle 1: Meilensteine

1.4.1 Individueller Terminplan Dominic Frostl

Persönliche Meilensteine	Termin
Konzept erstellt	6. September 2021
Bauteile verglichen und festgelegt	16. September 2021
Bauteile lagernd	4. Oktober 2021
Bauteile auf Funktionstauglichkeit getestet	5. Oktober 2021
Erste Softwareteile entwickelt	10. Oktober 2021
Geschwindigkeitstests des Gyroskops	11. Oktober 2021
Funktionstest des GPS-Moduls	12. Oktober 2021
Prototyp funktionsfähig	6. Dezember 2021
Auswertung der ersten Datensammlungen	7. Dezember 2021
Testsoftware funktionstauglich	27. Dezember 2021
Test der Akkulaufzeit	29. Dezember 2021
Leiterplatte entworfen	20. Jänner 2022
Leiterplatte bestückt	31. Jänner 2022
Testeinpassung in das Gehäuse	4. Februar 2022
Gerät funktionstauglich	7. Februar 2022
Erste Montage am Motorrad	21. März 2022
Erfolgreiche Fahrtdatensammlung	21. März 2022

Tabelle 2: Individueller Terminplan Dominic Frostl

1.4.2 Individueller Terminplan Manuel Gschwandtner

Persönliche Meilensteine	Termin
Konzept erstellt	6. September 2021
Map-Optionen getestet	10 Oktober 2021
Datenbank-Optionen getestet	21 Oktober 2021
Upload mit Hardware getestet	12 November 2021
Testsoftware funktionstauglich	27. Dezember 2021
Github Repository erstellt	1 Januar 2022
Diverse Anmeldemethoden getestet	18 Januar 2022
Update-Script funktionstauglich	25 Januar 2022
Website Elemente für Mobilgerät optimiert	20 Februar 2022
Website online	28. März 2022

Tabelle 3: Individueller Terminplan Manuel Gschwandtner

2 Konzept des Datenloggers

2.1 Generelles

Das Ziel ist es, einen Datenlogger für Motorräder zu entwerfen, der in der Lage ist verschiedene Fahrtdaten aufzuzeichnen. Als Herzstück des Projektes wurde der ESP32 gewählt, dieser ist sehr kompakt, bietet aber auch viele Anschlussmöglichkeiten für Sensoren. Für den fertigen Prototypen wurde das ESP32 DevC Kit gewählt. Nach ausreichender Planung wurde dadurch kein Platz auf der Hauptplatine verloren, sondern gewonnen, da dieses Entwicklungsboard, im Gegensatz zum ESP-WROOM-32, welcher direkt auf der Hauptplatine platziert werden müsste, mit Stiftleisten erhöht über den restlichen Komponenten der Platine platziert werden kann. Weiters kann so der Mikrocontroller einfach entfernt werden, um diesen bei einem Defekt zu tauschen oder neu zu programmieren.

2.2 Sensoren

Um das Aufzeichnen von Fahrtdaten zu ermöglichen, wurden ein Gyroskop, das MPU6050, und ein GPS-Modul eingesetzt. Nach genaueren Tests wurde festgestellt, dass das GPS-Modul zu ungenau und auch nicht kompakt genug für dieses Projekt war, deswegen wurde auf das BN-220 von Beitian gewechselt. Beim Testen des Gyroskops ist aufgefallen, dass die Orientierung des Sensors, in diesem Fall aufrechtstehend auf der Hauptplatine, ein ausschlaggebender Faktor für brauchbare Daten ist.

2.3 Mobilität

Aufgrund des Einsatzgebietes muss es möglich sein, das Gerät ohne direkte Netzverbindung betreiben zu können, also ist die Verwendung eines fest verbauten Akkus notwendig. Dieser soll fest im Gerät verbaut und über die entwickelte Hardware wiederaufladbar sein sowie eine ausreichende Laufzeit bieten, um auch längere Fahrten zu ermöglichen. Ursprünglich wurde festgelegt, zwei Zellen mit jeweils 3000 mAh Kapazität einzusetzen. Aufgrund dessen, dass der maximale Stromverbrauch der fertigen Schaltung nicht mehr als 200 mA beträgt, wurde jedoch auf eine Zelle verzichtet, um Platz zu sparen, da selbst mit einer einzigen Zelle eine Laufzeit von bis zu 16 Stunden erreicht werden kann.

2.4 Tracking

Die gefahrene Route, die dabei auftretenden Beschleunigungskräfte und Höchstgeschwindigkeiten sollen aufgezeichnet werden. Auch Rotationsbewegungen und Temperaturen sollen in bestimmten Zeitabständen gespeichert werden. Um dies zu ermöglichen, wurde ein SD-Karten-Modul in Betracht gezogen. Als Speichermedium wurde eine 8 GB-SD-Karte der Klasse 10 von Intenso gewählt, wobei eine SD-Karte der Klasse 10 mehr als ausreichende Lese- und Schreibgeschwindigkeiten bietet, als für die Datenmenge, die geschrieben wird, benötigt wird.

2.5 Datenverarbeitung

Auch das Verarbeiten der Daten soll kabellos erfolgen, indem die gespeicherten Daten über eine W-LAN-Verbindung an einen Webserver übertragen werden. Um das möglichst reibungslos zu gestalten, werden die Zugangsdaten des Heimnetzwerkes über einen vom ESP eröffneten Zugangspunkt bearbeitbar sein. Auch das Hinzufügen einer eindeutigen ID erfolgt über diesen Weg.

2.6 Schutz der Elektronik

Das Gehäuse muss wetterbeständig sein, da es auf dem Lenker eines Motorrads nicht vor Regen und Schmutz geschützt ist. Auch die auftretenden Vibrationen müssen im Gehäuse abgedämpft werden, um die Komponenten vor Beschädigung zu schützen. Besonders empfindliche Bauteile wie das GPS werden mit zwei Millimeter dickem Moosgummi gepolstert, um auch die stärksten Erschütterungen abzdämpfen. Auch die fest verbaute Lithium-Polymer-Zelle wird mit demselben Material als Polster verbaut. Die Hauptplatine wird innerhalb des Gehäuses verschraubt. Die Stellen, an denen die Platine verschraubt wird, werden Pads aus Moosgummi eingesetzt, um auch dort einen Dämpfungseffekt zu erwirken.

2.7 Montage

Das Anbringen an den Lenker soll über das gängige System der von Sony entwickelten Montagevorrichtung der GoPro Kamera erfolgen. Diese bietet ausreichend Beweglichkeit, um das Gerät aufrecht zu positionieren. Die Software muss in der Lage sein, Datensammlungen verschiedener Tage getrennt und speicherschonend zu sichern. So können Routen über mehrere Tage hinweg gespeichert werden, wenn zum Beispiel ein längerer Roadtrip geplant ist.

2.8 Benutzerfreundlichkeit

Eine vereinfachte Signalisierung über den Ladezustand des Akkus, des Betriebszustandes und des Upload-Status soll über drei RGB-LEDs an den Benutzer weitergegeben werden. Durch das Verwenden von mehrfarbigen Dioden, im Gegensatz zu einfarbigen, wird weniger Platz benötigt, um dieselben Informationen anzuzeigen.

3 Technische Umsetzung

3.1 Prozessor



Abb. 1: Der ESP32 von Espressif

Quelle:

<https://www.mouser.at/images/espressifsystems/lrg/ESP32-DEVKITC.jpg>

Zur Realisierung des Projektes wurde der Mikrocontroller ESP32 (siehe Abb. 1) der Firma Espressif verwendet; dieser wurde ausgewählt, da er als Dual Core Prozessor mit einer Taktrate von 240 MHz und 520 Kilobyte Arbeitsspeicher gute Performance bietet, aber trotzdem mit einem maximalen Stromverbrauch von 150 mA wenig Leistungsbedarf hat. Außerdem bietet dieser Mikroprozessor weitere Features wie einen 12 Bit Analog-Digital-Wandler, welcher für das Überwachen der Batteriespannung eingesetzt wird. Die Datenübertragung an den Webserver wird über das bereits im ESP32 integrierte WLAN-Modul mit dem Verbindungsstandard 802.11 b/g/n durchgeführt.

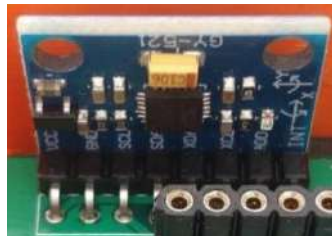
3.2 Datenspeicher



*Abb. 2: SD-Karten-Schaltung
auf der Hauptplatine*

Der Datenlogger muss in der Lage sein, viele Daten zu speichern. Um das zu realisieren, werden die einzelnen Datensätze von den Sensoren abgefragt und nach Überprüfung deren Validität auf ein externes Speichermedium geschrieben. Hier wurde eine SD-Karte der Klasse 10 von Intenso mit einer Speicherkapazität von 8 Gigabyte verwendet. Diese wurde mit einem CMOS Chip – dem CD74HC4050M – als Datenpuffer und einem SD-Kartensteckplatz (siehe Abb. 2) auf der Hauptplatine angebracht.

3.3 Gyroskop



*Abb. 3: Der MPU6050
aufrechtstehend montiert*

Beschleunigungskräfte, Rotationskräfte und Temperatur werden von einem MPU6050 aufgezeichnet und an den ESP weitergegeben. Die Kräfte werden dreidimensional, also in X-, Y- und Z-Richtung aufgezeichnet. Dieser Sensor wurde auf der Hauptplatine als Tochterplatine aufrechtstehend (siehe Abb. 3) verbaut, um die richtige Ausrichtung der jeweiligen Achse zu gewährleisten.

3.6 Akku



Abb. 6: Die verwendete Lithium-Polymer-Zelle

Die Stromversorgung des ESP32 wird mithilfe einer 3000 mAh Lithium-Polymer-Zelle realisiert. Diese hat eine Nennspannung von 3,7 V und deren Potenzial ist variabel von 2,6 V – 4,2 V, wobei die Zelle bei 2,6 V entladen und bei 4,2 V vollgeladen ist. Eine weitere Möglichkeit wäre gewesen, eine 18650er Zelle zu verwenden, doch diese ist wegen ihrer zylindrischen Bauform eher ungeeignet.

3.7 Gehäuse

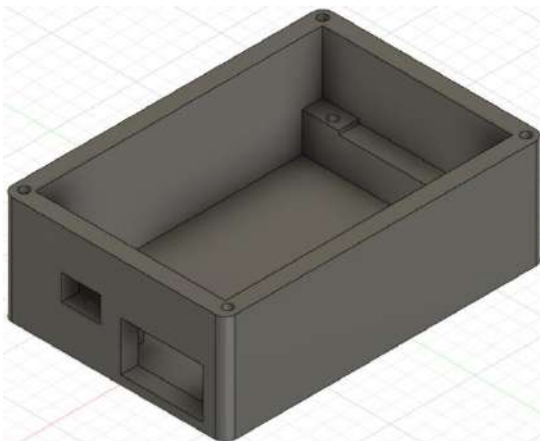


Abb. 7: Der Unterteil des Gehäuses

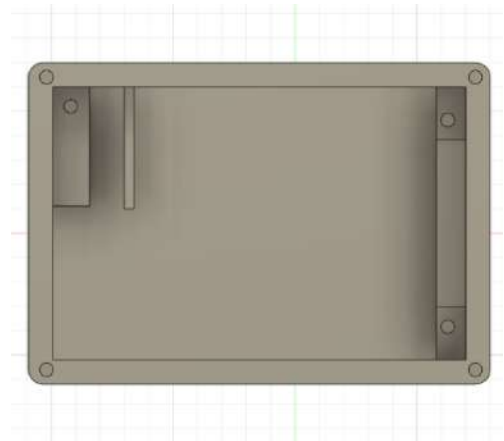


Abb. 8: Das Gehäuse betrachtet von oben

Da der Datenlogger am Lenker eines Motorrads befestigt wird, muss das Gehäuse dafür wetterbeständig und erschütterungsresistent sein. Um die Witterungsbeständigkeit zu erreichen, wurde das Gehäuse aus PETG per FDM Verfahren hergestellt. Schutz gegen unebene Fahrbahnen, starke Stöße und den üblich aufkommenden Vibrationen am Lenker bietet eine Auskleidung mit zwei Millimeter starkem Moosgummi um die empfindlichen Komponenten wie den Akku, das GPS Modul und die Hauptplatine.

4 Prototyp I

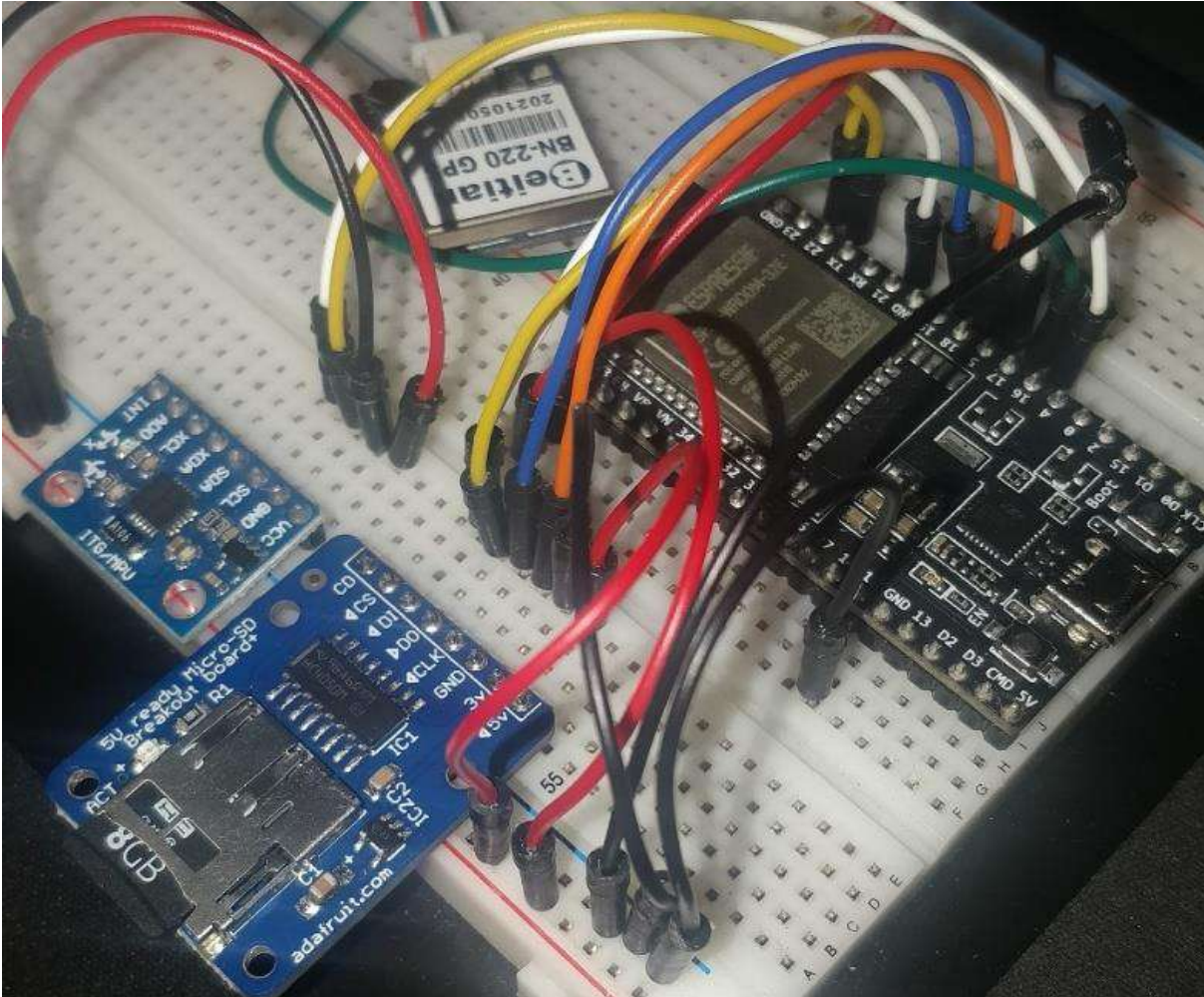


Abb. 9: Aufbau des ersten Prototyps auf dem Steckbrett

4.1 Aufbau

Der erste Prototyp wurde auf dem Steckbrett verdrahtet – um diesen zu testen, musste ein grundlegendes Programm auf den ESP32 geschrieben werden, womit die Sensordaten erstmals ausgelesen werden konnten. Dieses Programm wurde in der Arduino IDE geschrieben und kompiliert. Für die drei verwendeten Module gab es jeweils eine Library, die eingebunden wurde. Diese können in der Arduino IDE direkt heruntergeladen werden.

4.2 Verbinden des GPS-Moduls

Das GPS-Modul wurde über die serielle Schnittstelle an den GPIOs 16(RX) und 17(TX) mit dem ESP32 verbunden. Hier sollte beachtet werden, dass RX zu TX und TX zu RX verbunden werden muss. Mit der „TinyGPS++“ Library kann das GPS-Modul angesteuert werden, um dessen Daten auszulesen. Aus dieser Bibliothek wurden Methoden verwendet, um Daten wie Längengrad, Breitengrad, Höhe, Geschwindigkeit und die Anzahl der zum jeweiligen Zeitpunkt verbundenen Satelliten abzufragen. Weitere Methoden wurden nicht benötigt.

4.3 Verbinden des Gyroskops

Der MPU6050 wurde über den I²C Bus verbunden, SCL zu GPIO 21 und SDA zu GPIO 22. Für das Gyroskop wurde die dazugehörige Bibliothek von Adafruit verwendet. Diese bietet die Möglichkeit, Beschleunigungskräfte, Rotationskräfte und die Temperatur auszulesen. Beim MPU6050 sollte die Bandbreite des Beschleunigungssensors für den jeweiligen Zweck eingestellt werden.

Beim Motorradfahren treten üblicherweise Kräfte auf, die nicht größer als 2g sind. Deswegen kann die Bandbreite des Beschleunigungssensors auf 8g gesetzt werden, um sowohl eine hohe Genauigkeit zu erreichen als auch die Erkennung eines Unfalls zu ermöglichen. Der integrierte Tiefpass-Filter wurde auf 5Hz gesetzt, um die Störgeräusche, die am Lenker des Motorrads auftreten können, zu filtern. Der Rotationsbereich wurde in der Testsoftware auf 500° eingestellt.

4.4 Verbinden des SD-Karten-Moduls

Für das SD-Karten-Modul von Adafruit wurden die bereits in der Arduino IDE mitgelieferten Bibliotheken – „SD.h“ und „FS.h“ verwendet. Auf dem Board befinden sich mehrere Eingänge und Ausgänge, die richtig mit dem ESP32 verbunden werden müssen, um auf die SD-Karte schreiben zu können. Folgende Verbindungen sind notwendig, um die Funktionalität zu gewährleisten. Der CS Pin muss mit dem GPIO 5 verbunden werden. DI und DO müssen respektive auf GPIO 23 und GPIO 19 angeschlossen werden. Als letztes wird noch die

Verbindung von CLK zu GPIO 18 benötigt. Wenn alle Pins richtig verbunden worden sind, sollte es möglich sein eine Datei auf eine eingesetzte SD-Karte zu schreiben.

4.5 Entwicklung der ersten Testsoftware

Bei der Programmierung des ersten Prototyps ging es lediglich darum, die verwendeten Komponenten auf deren Funktionstauglichkeit zu prüfen. Dafür wurden die benötigten Bibliotheken in das Programm importiert und verwendet, um einfache Sensordaten auszulesen.

Diese Daten wurden über den Seriellen Monitor, welcher von der Arduino IDE bereitgestellt wird, ausgegeben. Dafür wurde die Methode „println(const char*);“ der Klasse „Serial“ verwendet. Die Funktion des SD-Karten-Moduls wurde durch das Erstellen eines Textdokuments, ohne Inhalt, getestet.

Weiters wurden Daten des GPS am Serial Monitor ausgegeben, um die Anlaufzeit des GPS-Moduls zu testen. Daraus ergab sich eine Wartezeit von über einer Minute in Gebäuden und knapp unter 30 Sekunden im Außenbereich.

5 Prototyp II

5.1 Aufbau

Beim zweiten Aufbau ging es darum den ersten Prototypen mobil zu machen. Um das zu ermöglichen, wurde im Voraus darauf geachtet, dass alle Komponenten mit einer Versorgungsspannung von 3,3 V betrieben werden können. Dadurch wird nur ein einziger Spannungswandler benötigt, welcher die Spannung des Akkus auf 3,3 V bringt. Das ist nötig, weil der verwendete Lithium-Polymer-Akku keine konstante Spannungsquelle ist und Spannungen im Bereich von 2,6 V – 4,2 V, abhängig vom Ladezustand, liefert.

5.2 Akku

Hierfür wurde eine 3000 mAh, 3,7 V Nennspannung, Lithium-Ionen-Zelle im Formfaktor 18650 verwendet. Diese wurde mithilfe einer Fassung mit Stecklitzen an das Steckbrett angeschlossen.

5.3 Spannungswandler

Um die Spannung des Akkus auf die benötigte Versorgungsspannung von 3,3 V zu wandeln, wurde ein LD1117AV33 eingesetzt. Dieser ist in der Lage, Spannungen von 2,6 V – 15 V auf 3,3 V zu regeln. Der untere Wandlungsbereich ist hier besonders wichtig, da der Akku bei niedrigerem Ladestand unter die benötigte Versorgungsspannung fällt.

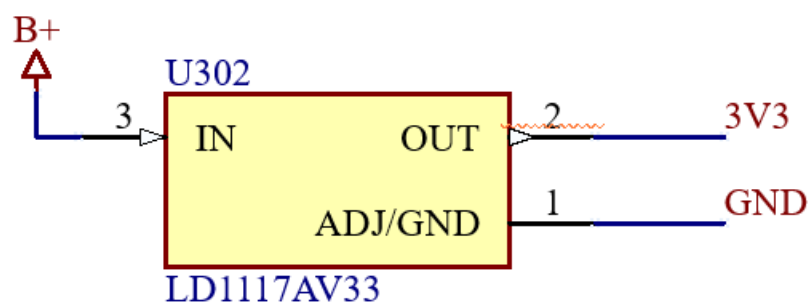


Abb. 10: Die Beschaltung des Spannungsreglers

Da der Akku durch den Tiefenentladungsschutz der Ladeplatine bei einer Spannung von 2,8 V durch einen MOSFET getrennt wird, ist der Bereich des Spannungswandlers ausreichend. Weiters ist die Trennung des Akkus bei 2,8 V auch schonender für die Zelle, was wiederum ihre Lebenszeit verlängert.

5.4 Überwachung des Ladezustands

Damit auch der Endnutzer den ungefähren Ladezustand des Akkus erkennen kann, wurde eine RGB-LED an den ESP32 angeschlossen. Diese zeigt die Farben Grün für >80% Ladezustand, Blau für 20% - 80% und Rot für <20%.

Um den Ladezustand am ESP32 einlesen zu können, wird der integrierte Analog-Digital-Converter verwendet. Standardmäßig ist die Auflösung des ADC auf 12 Bit eingestellt, was bedeutet, dass das eingehende Signal in 4096 (also 2^{12}) Stufen geteilt wird. Die Spannung des

Akkus muss halbiert werden, da am Mikroprozessor nicht mehr als 3,3 V an einen Eingang angelegt werden dürfen und auch der ADC nur Spannungen messen kann, die im Bereich von 150 mV – 2450 mV liegen.

Das wird mit einem 1:1-Spannungsteiler sichergestellt. Dieser besteht aus vier identischen Widerständen von jeweils 1 MΩ. Die Widerstände wurden so hoch dimensioniert, um den Stromfluss zu minimieren, was wiederum die Akkulaufzeit verbessert. Da der Strom sehr gering ist, können hier 0,25 Watt Widerstände verwendet werden. Durch das Teilen der Spannung ist die höchste Spannung, die an GPIO 34 (ADC1, Channel 6) anliegt, niemals größer als 2,1 V. Die genaue Verbindung mit dem ESP32 kann man in Abbildung 11 einsehen.

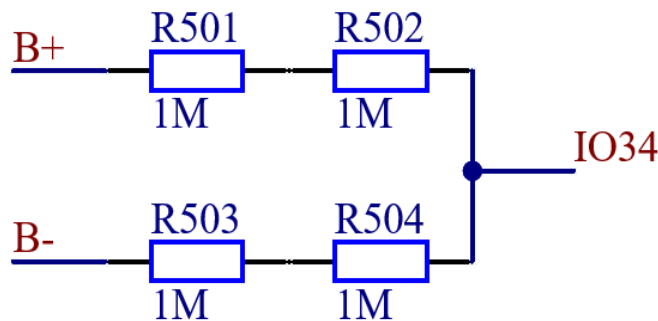


Abb. 11: Das Schaltbild des Spannungsteilers

5.4.1 Berechnungen

Mit der Formel: $ADC = \frac{V_{Ref} * ADC_{Steps}}{ADC_{Input}}$ kann der gelesene Wert des ADC berechnet werden.

ADC_{Input} ist die Spannung, welche an GPIO 34 anliegt. ADC_{Steps} ist die Auflösung des ADC, also 4096 (12 Bit). Bei der Auflösung sollte beachtet werden, dass in der Formel die nullte Stufe berücksichtigt werden muss. Deswegen wird an dieser Stelle 4095 eingesetzt. V_{Ref} ist die Referenzspannung für den Analog-Digital-Converter, welche bei verschiedenen ESP32 unterschiedlich sein kann. Sie kann von 1,0 V – 1,2 V sein. Dadurch kommt es bei dieser Berechnung zu Abweichungen. Tatsächlich wurde auf dem in diesem Projekt verwendeten ESP32 ein Wert von 2134 gelesen. Laut der Formel würde die Arduino-Funktion „analogRead(34);“ den Wert 2145 zurückgeben. Mit diesem Wert kann nun im Programm festgestellt werden, ob der Akku vollständig geladen ist.

5.5 Laden des Akkus

Für das Aufladen des Akkus wurde eine vorgefertigte Ladeplatine verwendet, welche gleichzeitig einen Tiefenentladungs- und Überladungsschutz bietet. Dadurch wird verhindert, dass der Akku während des Betriebs zu stark entladen wird.

Diese Platine wurde mit angelöteten Drähten mit dem Steckbrett verbunden. Es sollte unbedingt darauf aufgepasst werden, wie die Batterie mit der Platine verbunden wird, da ein falsches Verbinden die Funktion der Ladeplatine umgeht oder diese sogar beschädigen kann. Die Platine verfügt über vier Pads, welche mit „B+“ „B-“ „OUT+“ „OUT-“ beschriftet sind, wobei „OUT+“ und „B+“ auf der Platine verbunden sind.

Die Verbindung des Akkus wurde mit einem XH 2.54 mm Verbinder realisiert, welcher im Vorhinein auf der Platine montiert wurde. Auf dem Akku wurde das Gegenstück bereits ab Werk montiert.

Die genaue Schaltung des Lademoduls kann in Abbildung 18 betrachtet werden.

5.6 Erweiterung der Testsoftware

5.6.1 Ladezustand

Das Anzeigen des Ladezustands des Akkus wird mithilfe des ADC1 ermöglicht. Die genaue Funktion wurde bereits in Unterkapitel 3.2.3 erklärt. Hierfür muss der IO-Modus des GPIO 34 auf „Input“ gesetzt werden. Dies ermöglicht das Auslesen des ADC-Wertes.

Für die Anzeige des Ladezustandes wurde eine RGB-LED einprogrammiert, welche abhängig vom ADC-Wert, unterschiedliche Farben anzeigt. Um diese Diode ansteuern zu können, müssen drei GPIOs als Ausgang verfügbar sein. Hierbei muss berücksichtigt werden, dass nicht jeder GPIO Port des ESP32 fähig ist, als Ausgang verwendet zu werden.

In Abbildung 12 sieht man einen Teil des Codes, mit welchem die Farbe der Status-LED geändert wird. Dieser kurze Ausschnitt lässt diese LED grün leuchten, solange sich der Akku über einem Ladezustand von 80% befindet.

```
292  if (adcResult > 2000 && !wasBelow80 && !noBattery) {  
293      digitalWrite(LED3_PIN_RED, LOW);  
294      digitalWrite(LED3_PIN_BLUE, LOW);  
295      digitalWrite(LED3_PIN_GREEN, HIGH);  
296  }
```

Abb. 12: Code-Ausschnitt, in welchem die einzelnen Pins der LED entsprechend des Ladezustands gesetzt werden

5.6.2 Dateierstellung

Für das Speichern der Daten auf der SD-Karte wurden die benötigten Bibliotheken („FS.h“ und „SD.h“) importiert. Diese bieten Funktionen, um das Vorhandensein einer SD-Karte zu überprüfen, deren Größe zu bestimmen und diese zu beschreiben sowie Dateien auszulesen. Mit der Methode „open(const char* path, int mode)“ der Klasse „SD“ kann eine Datei geöffnet oder erstellt werden. Um eine Datei zu erstellen, muss der Dateipfad und Dateiname mit Endung angegeben, sowie der Modus auf „FILE_WRITE“ gesetzt werden. Erstellt wird in der Software eine .csv-Datei, da diese speicherschonend viele Daten aufnehmen kann.

In der folgenden Abbildung 13 kann man die notwendigen Parameter der Methode entnehmen. Hier gilt es unbedingt zu beachten, dass der Aufruf dieser Methode lediglich einen Pointer, welcher auf den Beginn der Datei zeigt, zurückgibt; daher muss dieser unbedingt in einer Variable des Typs „File“ abgespeichert werden. Jede weitere Datei-Operation erfolgt dann über die Methoden der Klasse „File“.

5.6.3 Datenverarbeitung

```
267 file = SD.open("/data-" + tmmmmjjjj + ".csv");
268 if (!file) {
269     Serial.println("File not found!");
270     Serial.println("Creating file: " + (String)("/data-" + tmmmmjjjj + ".csv"));
271     file = SD.open("/data-" + tmmmmjjjj + ".csv", FILE_WRITE);
272     file.print("Acc_X,Acc_Y,Acc_Z,Rot_X,Rot_Y,Rot_Z,tempC,time,lat,lng,alt,speed_kmh,satellite_amount;\r\n");
273     if (!file) {
274         Serial.println("File could not be written");
275         return;
276     }
277     Serial.println("File written! check SD card now!");
278     file.close();
279 } else {
280     Serial.println("File for today already exists. Appending to File...");
281 }
```

Abb. 13: Der Teil des Codes der für die Erstellung der Datei zuständig ist

Eine .csv (Comma Separated Values)-Datei speichert Datensätze indem einzelne Werte mit einem Komma getrennt und ganze Datensätze mit einem Semikolon getrennt werden. Nach jedem Datensatz muss auch ein „Carriage Return“ geschrieben werden. Das „Carriage Return“ ist die binäre Form der Enter-Taste.

Die Daten der Sensoren werden vor dem Schreiben auf das Speichermodul zu einem String verkettet und dann in die bereits erstellte Datei geschrieben. Damit zwischen verschiedenen Tagen unterschieden werden kann, wird im Programm beim Erstellen der Datei so lange gewartet, bis das GPS-Modul das aktuelle Datum liefert. So kann das Gerät zum Beispiel bei einer Fahrtpause ausgeschaltet werden und später bei erneutem Fahrtantritt wieder eingeschaltet werden. Die bereits erstellte Datei wird dann lediglich für das erneute Beschreiben geöffnet.

5.6.4 Benötigte Speichermenge

Die Datensätze werden in vordefinierten Zeitabständen von 50 Millisekunden gespeichert. Durch die geringen Zeitabstände wird die Route genauer abgetastet und schnelle Kurven werden detailreich aufgezeichnet.

Die dabei entstehende Datenmenge ist gering genug, sodass der Datenlogger über einhundert Jahre ununterbrochen Daten speichern kann, ohne den vorhandenen Speicher der 8-Gigabyte-SD-Karte komplett zu füllen. Dennoch werden die einzelnen Dateien, nach erfolgreichem Upload auf den Webserver, von der SD-Karte gelöscht, da die hochgeladenen Dateien in einer Datenbank bestehen bleiben.

5.7 Programmablauf

Der Programmablauf wird mithilfe eines Flow-Charts visualisiert. Dieses wurde auf der Website Lucidchart.com gezeichnet. In diesem Diagramm wurden die Farben der LEDs zum jeweiligen Zeitpunkt nicht eingezeichnet, da diese für den tatsächlichen Programmablauf nicht ausschlaggebend sind.

Sollte einer der Komponenten beim Startvorgang einen Fehler verursachen, wird der Programmablauf sofort gestoppt und LED₁ bleibt durchgehend rot. Sind alle Sensoren funktionsfähig, wird in einer Schleife überprüft, ob das GPS-Modul bereits ein korrektes Datum liefert. Während dieser Schleife blinkt LED₁ blau. Sollte das GPS-Modul noch kein korrektes Datum empfangen haben, gibt es für Tag, Monat und Jahr null aus.

Wenn das Datum korrekt ausgelesen wird, wird eine Datei mit dem aktuellen Datum auf der SD-Karte erstellt. Sollte bereits eine Datei mit diesem Namen auf dem Speichermedium existieren, wird diese stattdessen geöffnet, um Daten anzufügen. Bei erfolgreichem Erstellen bzw. Öffnen der Datei leuchtet LED₁ durchgehend grün.

Nach dem Öffnen der Datei werden die Daten des GPS-Moduls ausgelesen. Da die Höhe der letzte Wert ist, welcher von null auf den tatsächlichen Wert springt, wird dieser verwendet, um die Validität der Daten zu überprüfen. Ist die Korrektheit der Daten gegeben, werden diese gesammelt und formatiert in die erstellte .csv-Datei geschrieben.

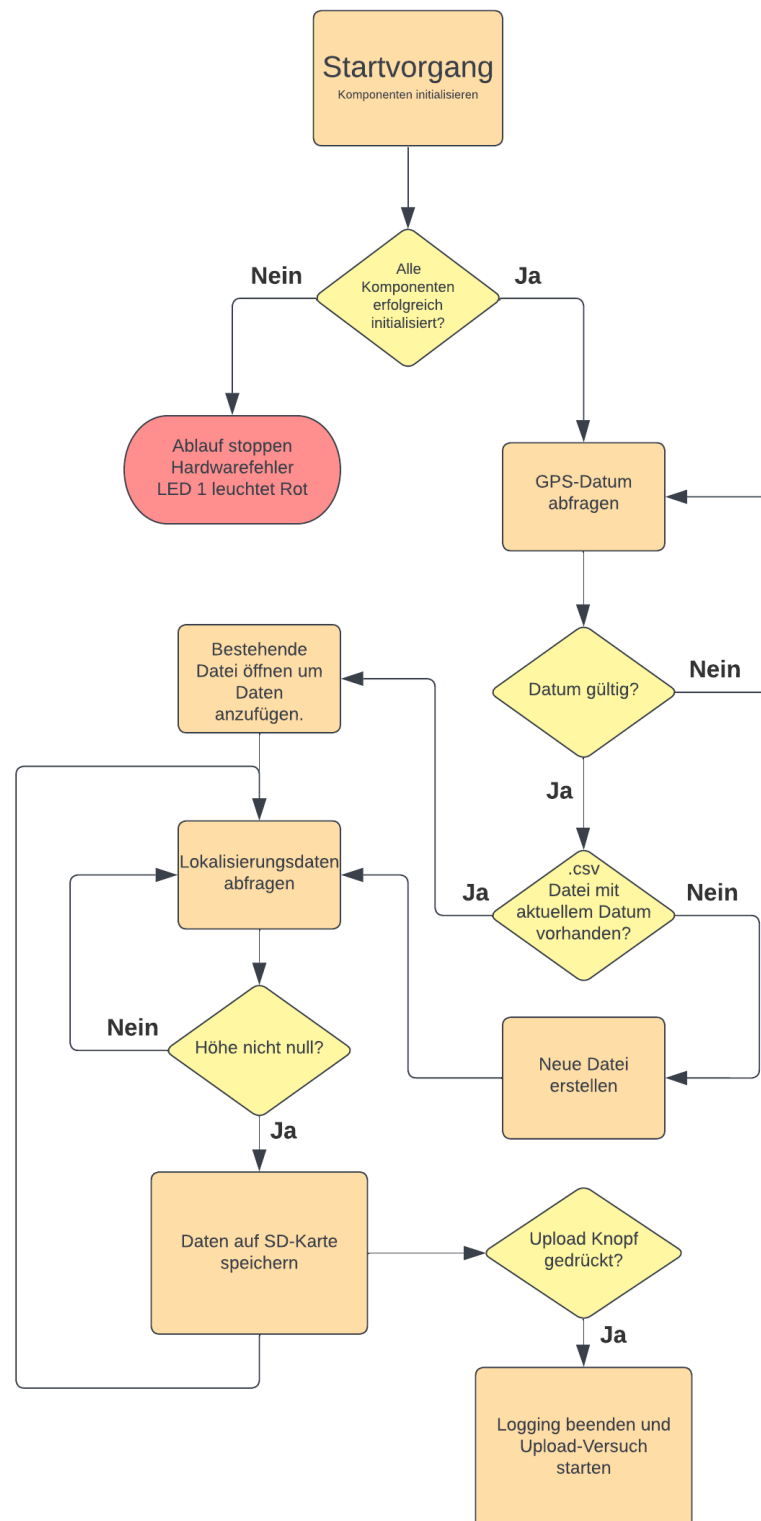


Abb. 14: Flow-Chart des Programmablaufs

5.8 Weitere Funktionen

Für die Verbesserung der Benutzerfreundlichkeit wurden optische Komponenten verbaut, um die Bedienung zu vereinfachen. Es wurden 2 weitere RGB-LEDs für den Programmstatus und für den Verbindungsstatus mit dem WLAN verwendet. Ein Taster startet bei Betätigung den Upload-Vorgang. Dieser Taster wurde mit einem Pull-Down Widerstand beschalten. Dadurch wird der GPIO 12, solange der Taster nicht gedrückt ist, gegen Masse gezogen, wodurch an diesem für den Mikroprozessor logisch ein LOW anliegt.

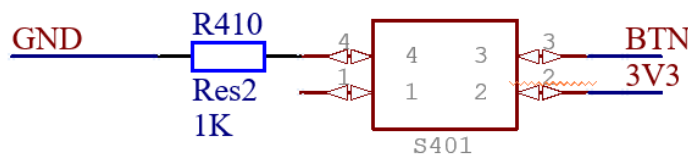


Abb. 15: Beschaltung des Tasters

5.8.1 Programmstatus

Um den Programmstatus, ohne den Serial Monitor, grob überwachen zu können, wurde eine weitere RGB-LED verbunden. Diese leuchtet während der Initialisierung der einzelnen Komponenten durchgehend rot. Nach der Initialisierung wird auf das aktuelle Datum, welches das GPS-Modul liefert, gewartet. Dies wird durch blaues Blinken signalisiert. Wenn das Datum verfügbar ist und die .csv-Datei zum Beschreiben erstellt bzw. geöffnet wurde, leuchtet die Diode durchgehend grün. Erst wenn das GPS-Modul die notwendigen Daten zur Aufzeichnung liefert, also Längengrade, Breitengrade und Höhe, beginnt diese LED grün zu blinken und die Fahrt kann begonnen werden.

5.8.2 Verbindung mit dem Webserver

Das integrierte WLAN-Modul des ESP32 ist in der Lage, sowohl als Zugangspunkt als auch als Station gestartet zu werden. Da es möglich ist, den Modus während der Laufzeit zu ändern, bietet dies die Möglichkeit, die Zugangsdaten des Ziel-WLAN-Netzwerkes zu ändern, ohne diese im Code festlegen zu müssen.

5.8.3 Konfiguration des Ziel-Netzwerkes

Für das Ändern der Zugangsdaten sowie des API-Keys wird ein einfacher Webserver auf dem ESP32 gehostet. Auf diesen kann jederzeit zugegriffen werden, indem man sich mit dem Telefon oder dem Computer mit dem Netzwerk verbindet, welches der ESP32 eröffnet hat. Nach der Verbindung mit dem ESP32 muss im Webbrowser des Endgeräts lediglich die IP-Adresse des ESP32 eingegeben werden, welche standardmäßig 192.168.4.1 ist. Im Webbrowser erscheint ein simples Formular in welches die SSID, das Passwort und der API-Key eingegeben werden müssen. In der folgenden Abbildung befindet sich der Code, welcher ausgeführt wird, sobald das Formular auf der Website mit dem „OK“ Knopf abgeschickt wird.

```
server.on("/setting", [] () {  
    String wifi_ssid = server.arg("ssid");  
    String wifi_pass = server.arg("pass");  
    String api_key = server.arg("key");  
    settings.begin("WiFicredentials");  
    settings.putString("STA_SSID", wifi_ssid);  
    settings.putString("STA_PASS", wifi_pass);  
    if (api_key != "") {  
        settings.putString("API_KEY", api_key);  
    }  
    settings.end();  
    Serial.println(wifi_ssid);  
    Serial.println(wifi_pass);  
    Serial.println(api_key);  
    credentialsSet = true;  
});
```

Abb. 16: Code-Ausschnitt, in welchem die eingegebenen Daten in den persistenten Speicher des ESP32 geschrieben werden

5.8.4 Verarbeitung der Zugangsdaten

Die Eingegebenen Parameter werden in den Speicherbereich des ESP32 geschrieben, welcher nach dem Ausschalten nicht gelöscht wird. Dieser Speicher wird auch EEPROM (Electrically Erasable Programmable Read-Only Memory) genannt. Das wird durch die Library „Preferences.h“ ermöglicht. Mit dieser wird ein Objekt der Klasse „Preferences“ mit dem Namen „settings“ erstellt welches den Zugriff auf weitere Methoden ermöglicht. Die Methode „begin(const char*);“ erstellt einen sogenannten Namespace mit dem übergebenen Namen. Innerhalb dieses Namespaces werden Key:Value Paare gespeichert. Wie in Abbildung 16 zu sehen ist, wird ein Bereich mit dem Namen „WiFicredentials“ erstellt, in welchem dann die Zugangsdaten und der API-Key gespeichert werden.

5.8.5 Upload-Vorgang

Wird nun auf den Upload-Taster gedrückt, versucht der ESP32 sich mit den vorab eingegebenen Zugangsdaten mit dem Netzwerk zu verbinden. Während des Verbindungsversuchs blinkt die dritte LED blau. Sollte der Versuch fehlschlagen, bleibt diese nach 10 Sekunden rot. Nach erfolgreichem Verbindungsversuch und Upload leuchtet diese LED durchgehend grün.

```
225 while (dir = file.openNextFile()) {
226     Serial.println(dir.name());
227     if (!client.connect("api.esp32-tour-tracker.tk", 80)) {
228         Serial.println("Connection FAILED!");
229         return 1;
230     }
231
232     int req_lenght = dir.size();
233     Serial.println("Connected ok!");
234     client.println("POST / HTTP/1.1");
235     client.println("Host: api.esp32-tour-tracker.tk");
236     client.println("User-Agent: ESP32");
237     client.println("apikey: " + api_key);
238     client.println("Content-Type: text/plain");
239     client.println("Content-Length: " + (String)req_lenght);
240     client.println("");
241     while (dir.available() > 0) {
242         char nc = dir.read();
243         parter += nc;
244         digitalWrite(LED2_PIN_BLUE, !digitalRead(LED2_PIN_BLUE));
245         if (dir.peek() == ';') {
246             char nc = dir.read();
247             parter += nc;
248             client.print(parter);
249             parter = "";
250         }
251     }
252     client.println("");
253     client.println("");
254     client.println("");
255     Serial.println("done");
256     SD.remove(dir.name());
257     Serial.println("File deleted");
258 }
```

Abb. 17: Code-Ausschnitt, in welchem der Upload über einen POST-Request an den Webserver erfolgt

In der obigen Abbildung befindet sich der Code, welcher in der Lage ist, mehrere .csv-Dateien hochzuladen. Dies ist notwendig, da es auch möglich sein muss Daten über mehrere Tage hinweg zu sammeln, wenn man einen Road-Trip macht. Dafür wird dem File Objekt ein Pointer zum Wurzelverzeichnis der SD-Karte übergeben. Mit der Methode „openNextFile();“ wird automatisch die erste Datei auf der SD-Karte geöffnet. Da diese Methode -1 zurückgibt, falls sich keine weitere Datei mehr im Wurzelverzeichnis befindet, kann so gleichzeitig die while-Schleife beendet werden. Jede Datei wird nach dem Hochladen automatisch gelöscht, um den zuvor verbrauchten Speicher wieder freizugeben.

6 Entwicklung der Leiterplatte

Ein wichtiger Faktor für die Mobilität dieses Projektes ist eine kompakte Bauform; das wurde auch bei der Entwicklung der Leiterplatte berücksichtigt. Somit wurden Komponenten platzsparend, aber übersichtlich platziert. Um eine Leiterplatte zu entwerfen, muss zuerst ein Schaltplan gezeichnet werden. Dieser wurde mit der Software Altium Designer gezeichnet und später für die Leiterplatte entflochten.

6.1 Analyse der Ladeplatine

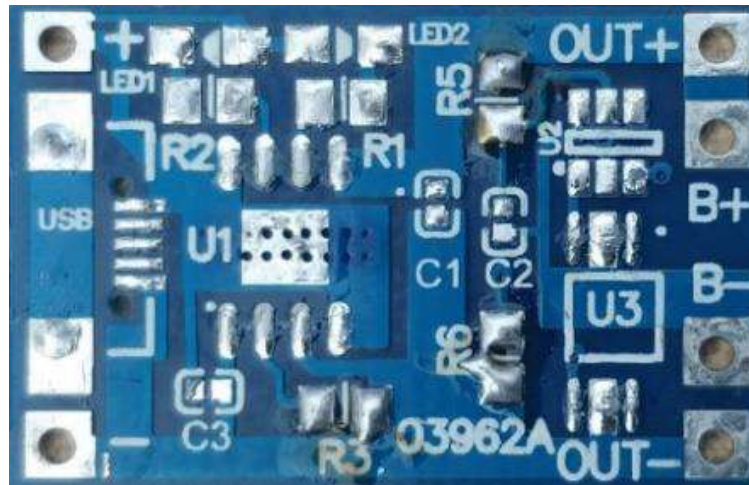


Abb. 18: Die Oberseite des Lademoduls ohne Bauteile



Abb. 19: Die Unterseite der Ladeplatine

Damit die essenziellen Komponenten direkt auf der Hauptplatine verbaut werden konnten, mussten diese zuerst rückwärtsentwickelt werden. Hierfür wurden die Bauteile der jeweiligen Platine entfernt und hochauflösende Bilder der Module gemacht. So konnte der Schaltplan von den einzelnen Platinen abgelesen werden und in Altium neu gezeichnet werden. Auch die Datenblätter der einzelnen Bauteile wurden herangezogen, um die genaue Funktionalität der Schaltung genauer zu verstehen.

6.1.1 TP4056

Dieser IC ist dafür zuständig, einen konstanten Stromfluss bei einer Spannung von 4,2 V zu erzeugen. In dieser Konfiguration wird, solange die Chiptemperatur nicht zu hoch ist, die Zelle mit 1A bei 4,2 V geladen. Sollte die Temperatur zu sehr ansteigen, reduziert der Chip automatisch den Stromfluss, um ein Überhitzen zu verhindern. Weiters bietet dieser Chip zwei GPIOs (7 und 6), auf welchen der Ladestatus angezeigt wird. Eine blaue LED leuchtet, wenn keine Zelle angeschlossen ist oder eine angeschlossene vollgeladen ist. Während des Ladevorgangs leuchtet eine rote LED.

6.1.2 DW01-P

Da der TP4056 keinen Tiefentladungsschutz bietet, muss mithilfe des DW01-P ein MOSFET gesteuert werden. Der DW01-P trennt bei 2,4 V mit einer Genauigkeit von ± 50 mV durch den MOSFET den negativen Anschluss des Lademoduls, wodurch der Betrieb des Gerätes sofort unterbrochen wird. Weiters entsperrt der Chip erst ab einer Akkuspannung von 3,0 V den MOSFET, wodurch das Gerät wieder verwendet werden kann. Bei dem MOSFET handelt es sich um einen FS8205A, einem IC in dem zwei identische N-Kanal MOSFETs in einem TSSOP-8 Gehäuse untergebracht sind. Diese sind in der Schaltung parallel verbunden, um den FET typischen $R_{DS\text{ON}}$, welcher bei diesem Transistor einen Wert von 34 m Ω hat, zu halbieren.

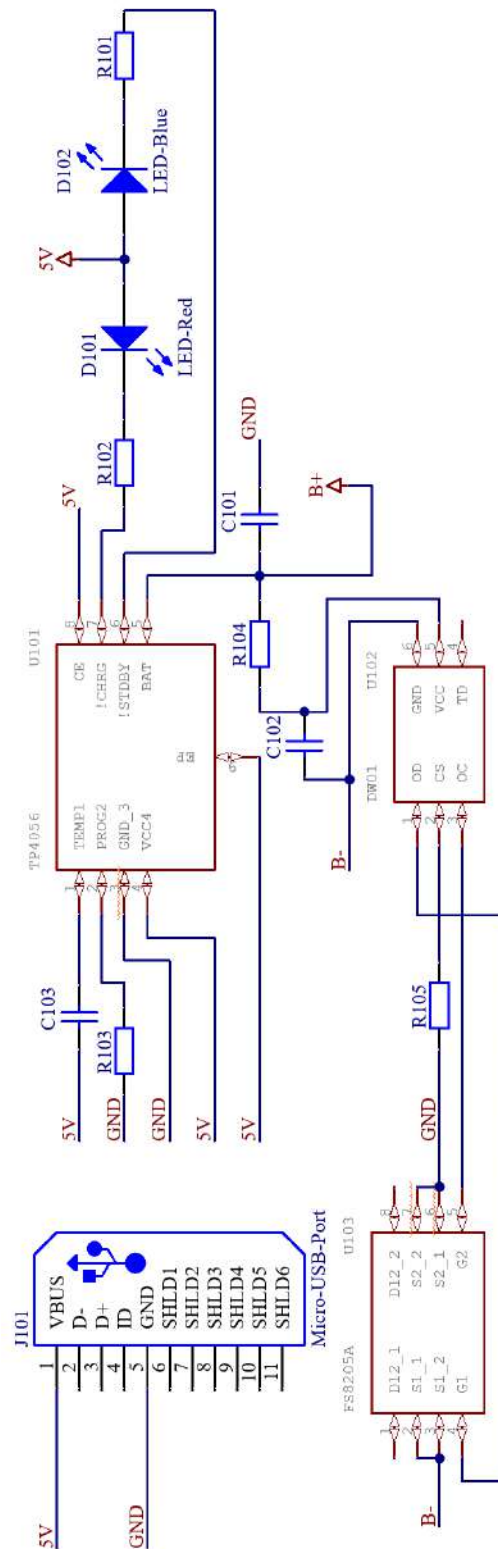


Abb. 20: Der gezeichnete Schaltplan der Ladeplatine

6.2 Analyse des SD-Karten-Moduls

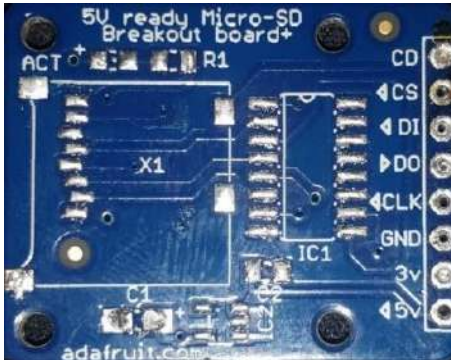


Abb. 21: Die Oberseite des SD-Karten-Moduls ohne Bauteile

Für die Analyse dieses Moduls wurde das Datenblatt des CD74HC4050M zur Hand genommen. Dadurch wurde klar, dass dieser Level Converter in diesem Anwendungsfall als Speicherpuffer zum Einsatz kommt. Das Datenblatt des Spannungswandlers IC2 wurde nicht benötigt, da dieser beim späteren Leiterplattendesign nicht verwendet wird.

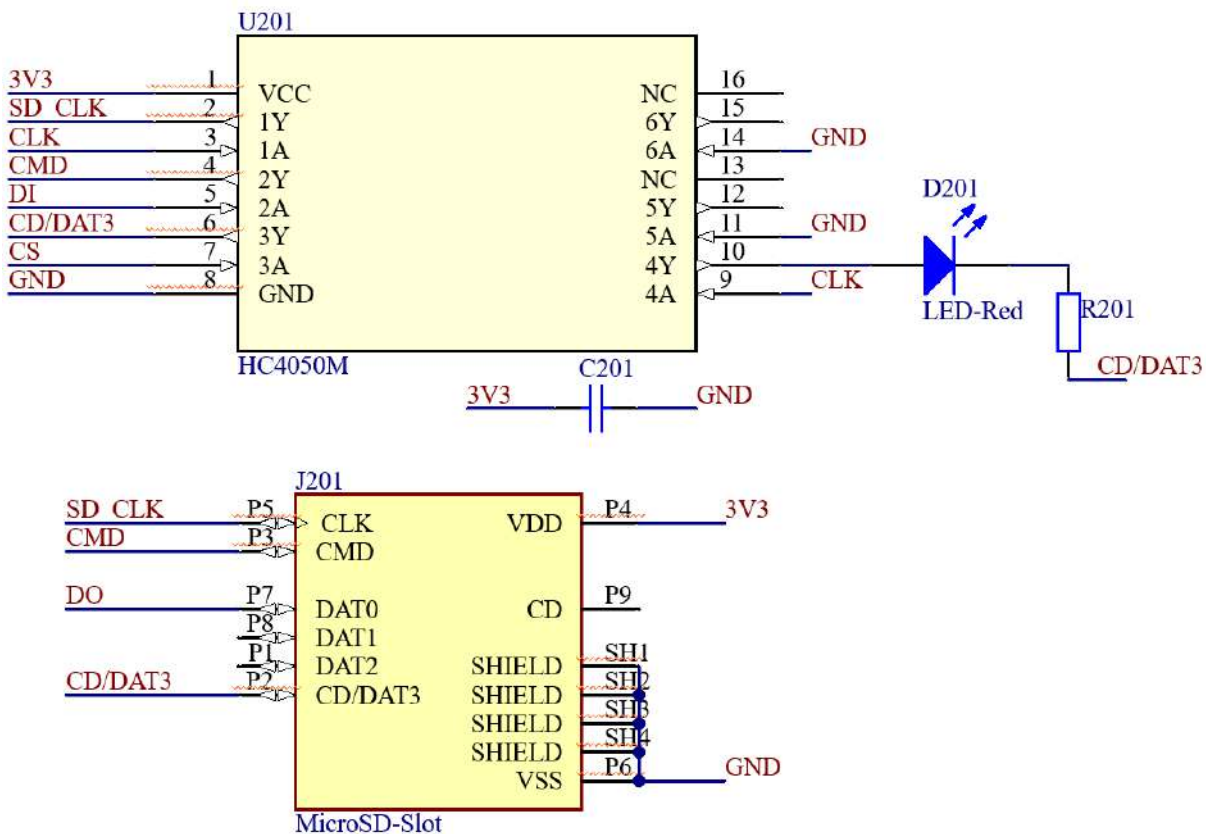


Abb. 22: Der resultierende Schaltplan aus Abb. 18

6.2.1 Nicht benötigte Komponenten

Auf dem Speichermodul ist ein Spannungsregler von 5,0 V zu 3,3 V ab Werk mitverbaut. Dieser ist im Hinblick auf dieses Projekt wegen der bereits bestehenden Versorgung mit 3,3 V nicht notwendig und wurde deswegen nicht in den Schaltplan aufgenommen.

6.2.2 CD74HC4050M

Dieser IC ist ein Level Converter, welcher als Zwischenspeicher verwendet wird. Beim Lesen der Daten von der SD-Karte werden einzelne Datenblöcke darin bis zum nächsten Taktsignal des Mikrocontrollers gespeichert. Aufgrund des niedrigen Datendurchsatzes ist diese Methode schnell genug, um die Datensätze in den vordefinierten Zeitabständen zu speichern.

6.3 Die Hauptschaltung

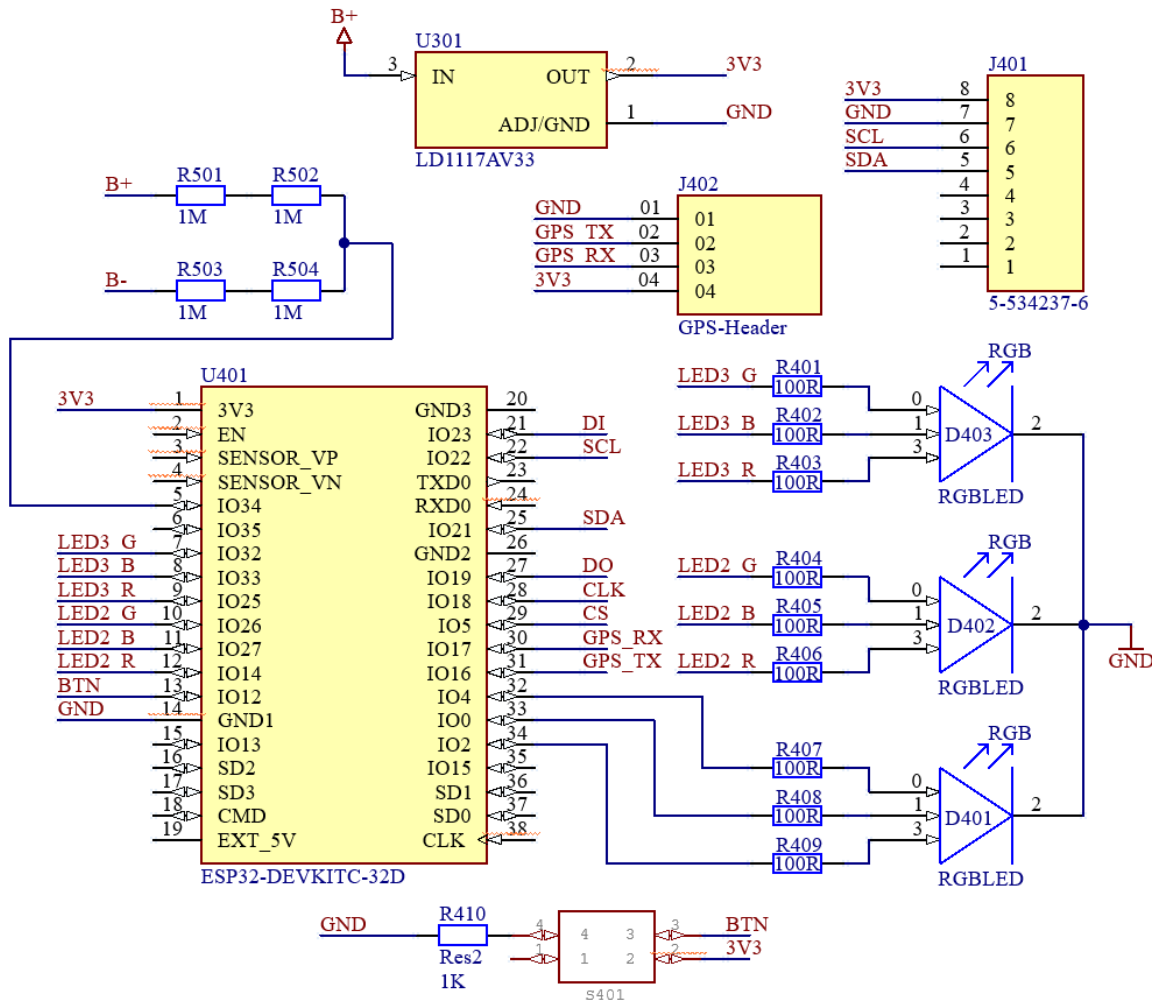


Abb. 23: Der Hauptschaltplan mit weiteren Komponenten wie dem Taster und den LEDs

Wie auf dem Schaltplan in Abb. 16 zu sehen ist, wurden sowohl für das Gyroskop als auch das GPS-Modul eigene Platzhalter verwendet. Dies ermöglicht das Hinzufügen eines Footprints. Als Footprint wurden jeweils Steckerleisten mit unterschiedlicher Pinanzahl gewählt. Für das Gyroskop wurde eine 8x1 Steckerleiste und für das GPS-Modul eine 4x1 Steckerleiste gewählt. Der Abstand zwischen den einzelnen Pins beträgt 2,54 mm. Die Footprints waren nur nötig, weil dadurch beim Entflechten des Prints einfacher der richtige Pol-Abstand eingehalten werden konnte, da das Gyroskop bereits eine achtpolige Steckleiste fest verbaut hat. Das GPS-Modul wurde mit dem im Lieferumfang enthaltenen Kabel mit der Hauptplatine verbunden. Der Taster wurde so mit dem GPIO 12 verbunden, dass bei Betätigung an diesem Eingang ein logisches HIGH anliegt.

6.4 Die entflochtene Platine

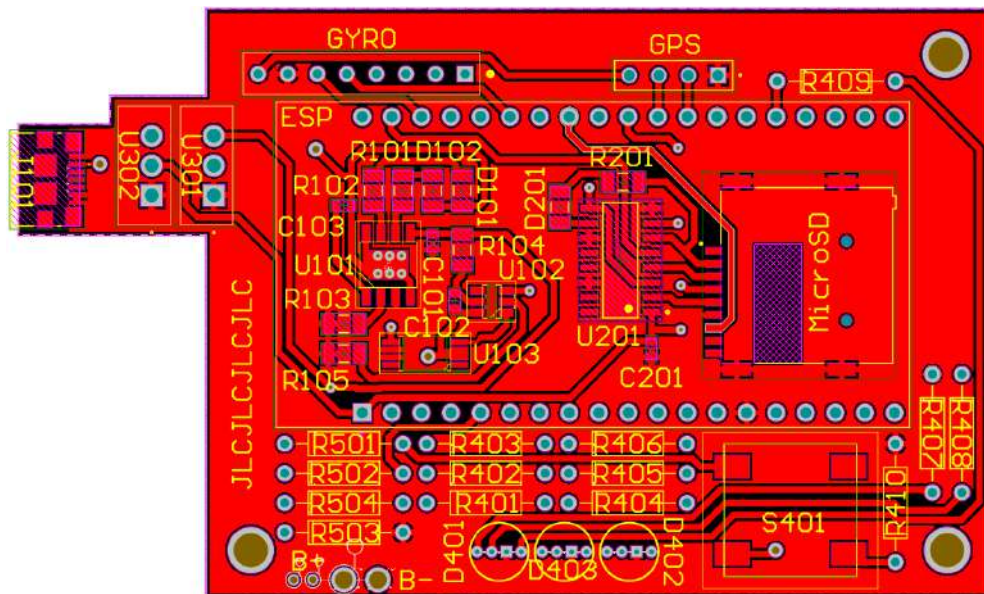


Abb. 24: Der Top-Layer der entflochtenen Platine

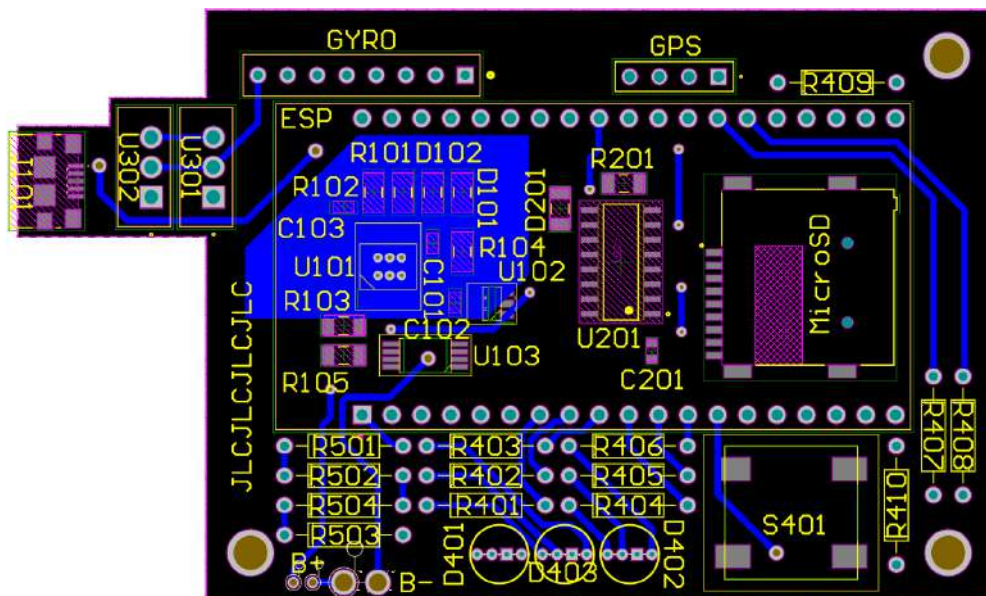


Abb. 25: Der Bottom-Layer der entflochtenen Platine

6.4.1 Anmerkungen

Wie in Abbildung 23 gut zu erkennen ist, wurde für den TP4056 (U101) eine Kupferfläche am Bottom Layer angelegt und mit Vias durchkontaktiert. Die Massefläche fungiert als Kühlkörper und verhindert damit das Überhitzen des ICs. Um Platz zu sparen, wurde das ESP32 Entwicklungsboard mit Sockelsteckleisten über einen Teil der restlichen Schaltung platziert.

Für die Montage wurden 19-polige Leisten als Sockel verwendet. Dadurch wird ermöglicht, den Mikrocontroller bei Bedarf aus dem Gerät zu entfernen, um diesen etwa neu zu programmieren oder bei einem Defekt auszutauschen. Weiters ermöglichen diese auch die Platzierung des Mikrocontrollers über der restlichen Schaltung.

6.4.2 Befestigung im Gehäuse

Für die Montage im Gehäuse wurden an drei Ecken der Leiterplatte Löcher eingezeichnet. Von diesen wurden im Gehäuse wegen notwendiger Anpassungen an der Platine nur zwei verwendet. Die Befestigung wird mit zwei M2 Schrauben durchgeführt. Zwischen Platine und Gehäuse befinden sich an den Verschraubungsstellen kleine Moosgummi Pads, die gegen stärkere Erschütterungen schützen.

6.4.3 Fehlerkorrekturen

Beim Bestücken der Leiterplatte wurden zwei Fehler entdeckt, diese konnten aber einfach behoben werden. Eine Ground-Verbindung des TP4056 wurde beim Zeichnen des Schaltplans übersehen. Hierfür wurde lediglich eine Stelle der umliegenden Massefläche vom Schutzlack befreit und diese mit dem betroffenen Pin verlötet.

Beim Hauptschalter, welcher ursprünglich an den zwei größeren „B-“ Pads montiert werden sollte, wurde zu spät bemerkt, dass diese Methode, das Gerät auszuschalten, das Laden während ausgeschaltetem Gerät verhindert. Um die Ladefähigkeit auch bei ausgeschaltetem Gerät zu ermöglichen, wurden diese Pads dauerhaft miteinander verbunden. Außerdem wurde die 3,3 V liefernde Leiterbahn, welche vom Spannungswandler U_{302} gespeist wird, aufgetrennt und über den Hauptschalter verbunden.

Da der Spannungswandler dadurch dennoch verbunden bleibt, kann es hier zu Verlusten des Akkuladezustandes kommen, jedoch waren diese bisher gering genug, sodass sich die Akkuspannung innerhalb von 3 Wochen nur um 0,01 V verringert hat, was durchaus im Toleranzbereich liegt. Eine andere Möglichkeit wäre gewesen, bei der Planung des Prints den Ausgang der Ladeschaltung über einen zweipoligen Hauptschalter zu trennen.

7 Entwicklung des Gehäuses

7.1 Schutz vor Umwelteinflüssen

Um die Elektronik vor Regen, starken Erschütterungen und Vibrationen zu schützen, wurde mittels Autodesk Fusion 360 ein Gehäuse entworfen, welches die empfindlichsten Komponenten, wie das GPS-Modul, schützt. Dafür wurde Moosgummi in verschiedenen Stellen des Gehäuses angebracht, um Bauteile wie den Akku und das GPS-Modul vor besonders starken Erschütterungen zu bewahren. Um das gesamte Gerät wasserdicht zu machen, wurde einerseits beim 3D-Druck der Einzelteile darauf geachtet, so wenige Einzelteile wie möglich zu verwenden, um weitere mögliche Fehlerstellen zu vermeiden und andererseits wurden Aussparungen für Dichtungen beim Designen des Gehäuses mit eingeplant.

Dennoch ist es notwendig mehrere Öffnungen an der Seite des Gehäuses zu haben, da zum einen der Hauptschalter in das Gehäuse eingepasst werden muss und auch der Micro-USB-Port, welcher für das Laden des Akkus benötigt wird bei Bedarf zugänglich sein muss. Der Hauptschalter wurde mit einem dafür geeigneten Kleber in die dafür freigelassene Stelle eingepresst. Für die Ladeöffnung wurde ein Verschlussstopfen modelliert und dieser mit einem O-Ring versehen. Für den Taster wurde eine Öffnung an der Oberseite des Gehäuses freigelassen. Um diese abzudichten, wurde ein zwei Millimeter dickes, rundes Stück Moosgummi darübergeklebt. Moosgummi wirkt hier außerdem nicht nur als Dichtung, da es aufgrund seiner Flexibilität auch verwendet wird, um indirekt den Taster zu betätigen. Die Öffnung für den Ladeanschluss wird mit einem Stöpsel, um welchen sich ein O-Ring befindet, verschlossen.

7.2 Entworfenene Teile

Es ergab sich ein Gehäuse, das aus einem unteren Teil besteht, welcher sämtliche Elektronik beinhaltet und einem oberen Teil, der das Gehäuse vollendet. Letzterer wurde mit Aussparungen entworfen, in welche später eine Fensterdichtung eingeklebt wurde. Des Weiteren wurde auf der Oberseite auch eine kreisrunde Öffnung, für den Taster auf der Hauptplatine, freigelassen und um diese eine Vertiefung von einem Millimeter. Die Öffnung wurde mit einem Kanal gedruckt, welcher im zusammengebauten Zustand kurz vor dem Taster endet. Für diesen Kanal wurde ein Stift gedruckt, welcher als Verlängerung des Tasters fungiert.

Beim unteren Teil wurde darauf geachtet, dass alle Komponenten passgenau eingelegt werden können. Es wurde eine Trennwand im Gehäuse eingeplant, die verhindert, dass der Akku verrutschen kann. Weitere Erhöhungen im Inneren des Gehäuses sorgen dafür, dass die Hauptplatine sicher über dem Akku platziert werden kann. Löcher in den inneren Erhöhungen sowie den äußeren Wänden bieten die Möglichkeit nach dem Drucken, M2-Gewinde aus Messing in das Gehäuse einzupressen. Das wurde mit einem LötKolben gemacht, da damit das Plastik erhitzt werden kann, sodass das Gewinde ohne besonders viel Druck in das vorgefertigte Loch hineingeschoben werden kann. Das verhindert mechanischen Stress, der beim Pressen ohne Wärme entstehen würde und sorgt auch für eine bessere Haftung des Gewindes im Gehäuse, da sich das Plastik an das Metall anfügt. Außerdem besteht die Gefahr, dass beim Einpressen ohne Wärme die Gehäusewand beschädigt wird, da diese nicht genug umschließendes Material bietet, um die dabei entstehende mechanische Spannung auszugleichen.

Für das GPS-Modul wurde eine Halterung gedruckt, welche innerhalb des Gehäuses mit der Platine verschraubt wird. Dadurch wird das GPS-Modul an die höchstmögliche Stelle im Gehäuse gebracht, um den bestmöglichen Empfang zu erreichen, und gleichzeitig die Leiterplatte fixiert.

7.3 Zusammenbau



Abb. 26: Unterteil des Gehäuses mit eingesetztem Akku

Für den Zusammenbau wurde die Leiterplatte bestückt, der Hauptschalter mit einem Stecker versehen und in das Gehäuse eingeklebt. Sämtliche Sensoren sind bereits mit der Hauptplatine verbunden, damit diese nur mehr in das Gehäuse gelegt werden muss. Zwei Schichten Moosgummi wurden in der Größe des Akkus ausgeschnitten. Eine der beiden Schichten wird zuerst in das Gehäuse gelegt und darauf der Akku.

Als Nächstes wird die zweite Schicht Moosgummi auf den Akku gelegt. Vor dem Einsetzen der Leiterplatte wird der Stecker des Hauptschalters mit dem dazugehörigen Gegenstück auf der Platine verbunden. Nach einer erfolgreichen Funktionsüberprüfung wird die Leiterplatte mit dem Gehäuse verschraubt.



Abb. 27: Test der Funktionstauglichkeit vor dem Verschrauben der GPS-Halterung



Abb. 29: Die GPS-Halterung

Ist die Leiterplatte verschraubt, kann der letzte fehlende Gehäuseteil eingebaut werden. Das GPS-Modul wird hier, wie der Akku, mit 2 Schichten



Abb. 28: Der fertig bestückte Unterteil des Gehäuses

Moosgummi in die Halterung eingeschoben. Danach kann diese im Gehäuse montiert werden.

Als Nächstes muss der Gehäusedeckel vorbereitet werden. Dafür wird Fensterdichtung in den freigelassenen Rahmen geklebt und die Öffnung für den Taster mit Moosgummi verschlossen.



Abb. 30: Der Gehäusedeckel mit Dichtung und Schrauben



Abb. 31: Finaler, funktionstauglicher Prototyp

Ist die Taster-Verlängerung in den Oberteil des Gehäuses eingesetzt worden, kann das Gehäuse geschlossen werden.

7.4 Montage am Motorrad



Abb. 32: Der Datenlogger, montiert und einsatzbereit

Für die Montage an einem Lenker kann auf der Unterseite des Gehäuses ein GoPro Mount verschraubt werden. Das GoPro Mount bietet ausreichend Beweglichkeit, um das Gerät richtig auszurichten, was notwendig ist, um korrekte Daten zu lesen.

8 Einleitung Server

Der folgende Teil der Abschlussarbeit befasst sich mit der Konfiguration und Funktionalität des Servers sowie den verwendeten Werkzeugen und Technologien im Entwicklungsprozess.

Anfangs wird der erstellte Prototyp vorgestellt, welcher dann in das fertige Projekt ausgebaut wurde. Darauf folgt eine Erklärung des Frontend und Backend und der jeweils verwendeten Frameworks und Libraries. Als Nächstes werden diverse verwendete Entwicklungswerkzeuge und verwendete grafische Applikationen vorgestellt. Die letzten zwei Teile befassen sich mit diversen Lösungen von Hindernissen und mit der Zukunft des Projektes.

9 Prototyp der Website

Der Prototyp bestand aus einem lokal laufenden Node.js-Server. Durch das Aufrufen von dessen URL wurden Werte aus einer CSV-Datei ausgegeben und auf einer Karte visualisiert. Dieser Prototyp hatte noch keine Funktionalität in Bezug auf andere Fahrtdaten, wie etwa Geschwindigkeit.

Auf dieser Basis wurden diverse Optionen zur Darstellung und Speicherung der Daten getestet. Weiters wurden statische Bedienelemente erstellt, um diverse Layouts von Elementen vor dem Designprozess zu testen.

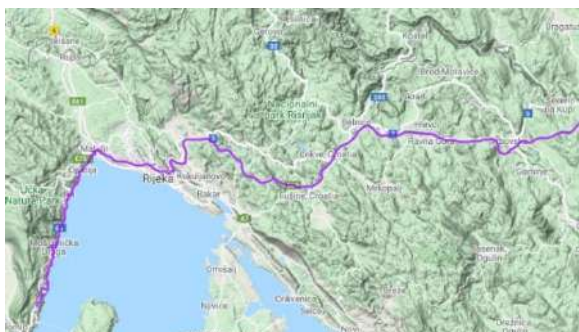


Abb. 34: Google Maps Test.



Abb. 33: Mapbox Test.

10 Backend

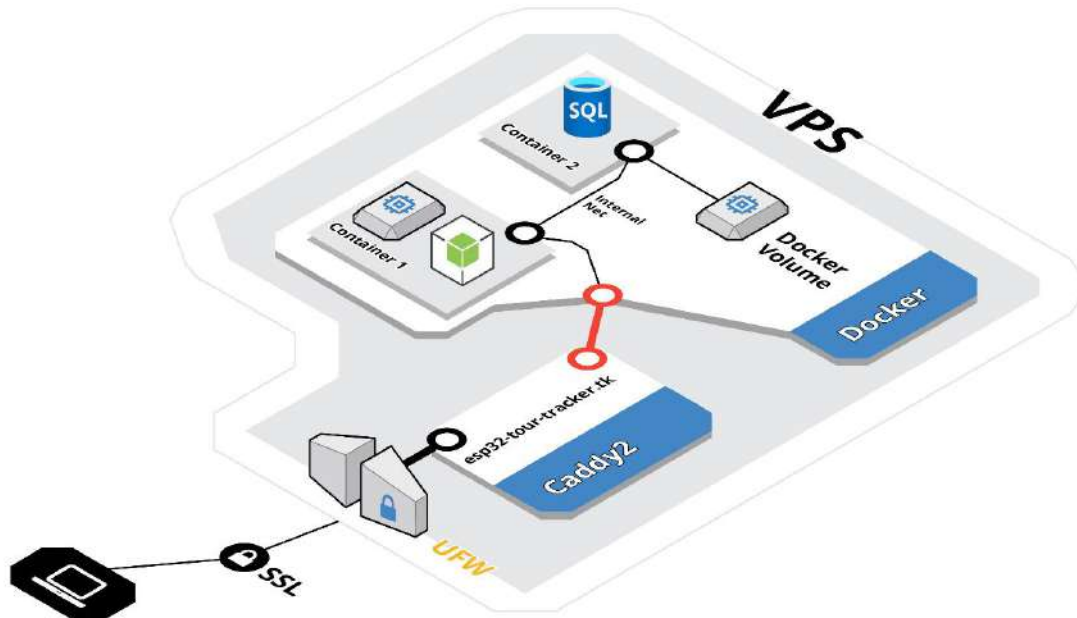


Abb. 35: Serveraufbau grafisch dargestellt.

10.1 VPS

Zum Hosten des Services wurde ein VPS der Firma Contabo verwendet, die Domain „esp32-tour-tracker.tk“ wurde bei Freenom erworben.

10.2 Caddy2

Aufgrund der automatischen Verwaltung und Erstellung von SSL-Zertifikaten wurde Caddy2 als Proxyserver gewählt; des Weiteren bietet der modulare Aufbau des Servers gute Übersicht über die Funktionen und Einstellungen der diversen Bestandteile.

zur Konfigurierung kann entweder ein sogenannter Caddyfile oder die lokal verfügbare REST-API verwendet werden. Mithilfe der „Config Adapters“ stehen auch Sprachen wie zum Beispiel YAML, JSON und TOML zur Verfügung. Funktionen wie Hosten von statischen Dateien, Lastverteilung und Protokollierung der empfangenen beziehungsweise gesendeten Daten werden auch unterstützt.

10.3 Docker

Docker wurde zwecks Containerisierung der Server verwendet. Dies ermöglicht das einfache Hosten unter diversen Betriebssystemen und Linux-Distributionen ohne Migrationsaufwand der Service.

Der erste Container beinhaltet einen MySQL-Server, welcher zum Speichern von Daten wie etwa Notizen und anderen nutzerspezifischen Informationen dient. Der zweite Container befasst sich mit der Node.js Runtime, welche für APIs, die Weboberfläche und Datenbankzugriffe zuständig ist.

Zwischen den beiden Containern gibt es ein Docker-internes Netzwerk zum Informationsaustausch. Des Weiteren wurde mithilfe von Caddy2 dem Node.js-Container der Datenaustausch mit dem Internet ermöglicht. Dieses Set-up hat den Vorteil, dass die Datenbank und Node.js nicht direkt über das Internet erreichbar sind und dadurch besser gegen diverse Angriffe geschützt sind.

In Bezug auf Datenspeicher gibt es für die MySQL-Datenbanken ein Docker-Volume, welches auch nach dem neu Erstellen oder Löschen des Containers bestehen bleibt. Das Speichern der CSV-Dateien wird direkt von dem Node.js-Container übernommen.

Es wurde ebenfalls das Verwenden einer externen Datenbank getestet, hierbei musste nur der MySQL-Server, definiert in dem Umgebungsvariablen geändert werden.

10.4 Node.js

Als Backend wurden drei Optionen in Erwägung gezogen: Laravel, Python und Node.js. Aufgrund der guten Dokumentation und leichten Erweiterbarkeit fiel die Entscheidung auf Node.js.

10.5 Services, Libraries und Frameworks

10.5.1 Express

Express wurde als Web-Framework verwendet, da es einen robusten Funktionsumfang für Web- und Mobilapplikationen bietet. Des Weiteren beinhaltet Express API-Middleware und HTTP-Funktionen mit hervorragender Dokumentation. Einbinden von zusätzlicher Middleware für Funktionen wie etwa Authentifizierung oder Protokollierung ist ebenfalls mit der Hilfe von Node Modules möglich.

10.5.2 Prisma

Prisma ist ein Typesafe ORM, welches zur Kommunikation zwischen Datenbank und Node.js verwendet wurde. Aufbau und Beziehungen der Datenbank werden mit einem Schema definiert, das während der Entwicklung geändert und bearbeitet werden kann. Dieses Schema kann dann zu der Datenbank migriert werden. Sofern dabei keine bereits beschriebenen Spalten geändert oder umbenannt werden, ist dies auch ohne Datenverlust möglich.

Das Schema bietet ebenfalls eine bessere Übersicht über Relationen und Dateitypen, da die gesamte Datenbank innerhalb einer Datei betrachtet werden kann.

Prisma kann mit diversen Datenbanken (PostgreSQL, MySQL, MongoDB, ...) verbunden werden, für dieses Projekt wurde, wie bereits erwähnt, ein MySQL-Server gewählt.

Es wurde ebenfalls Firebase getestet, da ein Großteil des Funktionsumfangs jedoch nicht benötigt wurde, fiel die Entscheidung auf Prisma.

10.5.3 Auth0

Als Log-in-System wurde Auth0 in Kombination mit OpenID verwendet. Der Hauptentscheidungsgrund dafür war, das einfache Einbinden von externen Log-in Optionen wie Google und GitHub. Zur Verbindung mit Node.js wurde ein Node Module namens „express-openid-connect“ verwendet, welches durch Umgebungsvariablen konfiguriert wurde. Requests von nicht authentifizierten Geräten werden auf die Log-In-Seite umgeleitet.

10.6 Datenbankaufbau

Die Datenbank ist in drei Tabellen unterteilt: „user“, „api_keys“ und „tours“.

Die „user“-Tabelle speichert den Auth0 Token des jeweiligen Users und hat den Zweck, diverse Daten einem User zuzuordnen. In der „api_key“-Tabelle werden vom Nutzer generierte API-Keys und deren Namen gespeichert. Das Speichern der Touren-Informationen erfolgt in der „tours“-Tabelle, hier wird der Dateiname der CSV-Dateien, der Name der Route und Notizen des Nutzers gespeichert. Name und Notizen können vom Nutzer je nach Wunsch geändert werden.

10.7 Private-API

10.7.1 api/upload

10.7.2 POST

Die Upload-API hat einen POST-Endpoint, welcher das manuelle Hochladen von CSV Dateien ermöglicht. Diese Dateien werden in einem Ordner, basierend auf der User ID in dem Node.js-Container abgelegt.

10.7.3 api/key

10.7.4 GET

Antwort mit allen API-Keys eines Nutzers und den dazugehörigen Namen.

```
1  [
2    {
3      "key": "YrL60XJtykRV30spqkb9UQuh2pVTWdd9qqHs5rhHdqnYaKLwZX...",
4      "name": "Peugeot 207"
5    },
6    {
7      "key": "x2e0nnW6dXbRtlvChia2RavjVchDDnP7ovgnumamnlg3jTlMh1XV...",
8      "name": "Suzuki GS500"
9    }
10 ]
```

Abb. 36: Beispielausgabe „api/key/“ GET.

10.7.5 POST

Generiert einen Key und speichert diesen in der Datenbank. Der Standardname dieses Keys ist „Tracker“.

10.7.6 DELETE

Löscht einen API-Key des Nutzers, in Zusammenhang mit der mitgesendeten ID. Wenn ein Nutzer zum Beispiel 2 Keys besitzt und einen API-Request für die ID 3 sendet, wird kein Key gelöscht und er erhält „id doesn't exist“ als Antwort. Keys von anderen Usern können somit nicht gelöscht werden.

10.7.7 PATCH

Dient zur Namensänderung des API-Keys, ist wie DELETE relativ zu den API-Keys des Nutzers.

10.7.8 api/tours

10.7.9 GET

Sendet alle vorhandenen Routen eines Nutzers. Hier wird ebenfalls kontrolliert, ob seit dem letzten Request eine Datei in Bezug auf den Nutzer gelöscht oder hinzugefügt wurde.

```
1 {
2   {
3     "name": "Stp. Wanderung",
4     "notes": "Lorem ipsum dolor sit amet, consectetur adipi.",
5     "file": "data-18-3-2022.csv"
6   },
7   {
8     "name": "Kn. Motorrad",
9     "notes": "The quick brown fox jumped over the lazy dog.",
10    "file": "data-27-1-2022.csv"
11  }
12 }
```

Abb. 37: Beispielausgabe „api/tours/“ GET.

```
1 {
2   "9.96",
3   "0.04",
4   "2.50",
5   "29.44",
6   "22.94",
7   "26.68"
8 }
```

Abb. 38:
Beispielausgabe
„api/tours/“ GET
mit Datei.

Wenn ein Dateiname mitgesendet wird, besteht die Response aus Informationen über Temperatur und Geschwindigkeit der Tour (Maximal-, Minimal- und Durchschnittswert).

10.7.10 PATCH

Dient zum Ändern des Namens und der Notizen einer Route. Erwartet die ID der Route und den gewünschten Namen sowie die gewünschte Notiz. Wenn nur ein Wert übermittelt wird, bleibt der andere unverändert, somit ist zum Beispiel nur das Ändern der Notiz möglich.

10.7.11 DELETE

Hierbei wird eine gewünschte Datei in einen „trash“-Ordner verschoben und ist somit für den Nutzer nicht mehr erreichbar. Je nach Einstellung des Servers werden Dateien älter als zum Beispiel 10 Tagen aus dem „trash“-Ordner gelöscht.

10.8 Public-API

Des Weiteren gibt es zum Hochladen via eines Trackers eine Public-API, welche ohne Log-in aufrufbar ist. Requests an diese API werden basierend auf dem mitgesendeten API-Key abgelehnt oder akzeptiert und dem jeweiligen Nutzer zugeordnet. Wird ein falsche API-Key gesendet, erhält man „api key doesn't exist“. Übertragen werden die Dateien im „text/plain“ Format, wodurch keine Konvertierungen oder Codierungen auf Seiten des Trackers nötig sind. Dieser Teil der API befindet sich auf einer separaten Top-level-Domain namens „api“.

11 Frontend

11.1 Libraries und Frameworks

11.1.1 MDBootstrap

MDBootstrap ist ein unter der MIT-Lizenz herausgegebenes Framework, basierend auf Bootstrap. Es bietet eine einfache Implementierung von UI-Elementen basierend auf dem Material Design Standard. Beispiel für UI-Komponenten sind: Input-Gruppen, Akkordeons und Tabellen. Für spezifische Anpassungen beinhaltet das Framework diverse CSS Klassen mit vordefinierten Funktionen, sogenannten „CSS Utility Classes“. Diese Klassen können unter anderem Ausrichtungsoptionen, Farboptionen und Abstände diverser Elemente beinhalten.

Eine Pro-Version ist ebenfalls verfügbar, welche mehr UI-Komponenten wie etwa Diagramme mit sich bringt. Weiters bietet sie Support bei diversen Problemen und Bugs.

Für dieses Projekt wurde die kostenlose Version verwendet.

11.1.2 Font Awesome

Font Awesome ist eine auf Webseiten und Apps spezialisierte Icon Library, welche diverse Symbole für UI-Elemente und Firmenlogos zur Verfügung stellt. Einen Großteil der Icons gibt es in verschiedenen Varianten, mit etwa dünneren oder zweifarbigen Komponenten. Diese Library bietet ebenfalls eine Pro-Version, welche mehr Stylingoptionen und Icons beinhaltet.

11.1.3 Google Maps API

Die Google Maps API ermöglicht das Einbinden diverser Karten und deren Anpassung. Einstellungen wie Zentrierung und Zoom können mithilfe von Javascript vorgenommen werden.

Durch Javascript können ebenfalls diverse Elemente wie: Pop-ups, Markierungen, Routen und Overlays zur Karte hinzugefügt werden. Erstellte Elemente könne auch dynamisch angepasst und geändert werden. Diverse Anpassungen an Steuerelementen und dem Interface sind ebenfalls möglich.

Als Alternativen wurden Mapbox, Leaflet und Grafana getestet. Aufgrund der besseren Visualisierung von Satellitenaufnahmen und Street View fiel die Entscheidung jedoch auf Google Maps.

11.1.4 SweetAlert2

SweetAlert2 ist ein moderner Ersatz für Javascript Alerts und wurde für diverse Benachrichtigungen und Bestätigungsaufforderungen an den Nutzer verwendet.

Das Einbinden von Eingabefeldern für zum Beispiel Text oder Nummerneingabe ist auch möglich. Die eingegebenen Daten können dann mithilfe einer sogenannten Callback-Funktion, welche nach der erfolgreichen Eingabe oder Bestätigung aufgerufen wird, verarbeitet werden.

Anpassungen können mithilfe des Setzens von CSS-Klassen vorgenommen werden. Hierbei muss aber das Styling des gewünschten Elements deaktiviert werden, ansonst werden die Standardstile von der Library priorisiert.

11.1.5 Chart.js

Das Darstellen der Diagramme übernimmt die Javascript Library Chart.js. Das Rendern findet auf einem HTML5-Canvas statt, wodurch gute Performance in allen modernen Webbrowsern gegeben ist. Darstellung von großen Datensätzen mit 1 bis 2 Millionen Werten ist problemlos möglich.

Chart.js bietet ebenfalls Funktionen zur Änderung diverser Einstellungen eines Diagramms, wie zum Beispiel die maximal zulässigen Werte pro Achse. Auch das Aktualisieren eines gesamten Datensatzes beziehungsweise eines Diagramms ist ohne Neu-Laden der Seite möglich.

11.1.6 Chroma.js

Chroma.js ist eine einfache Library zum Erstellen von Farbverläufen, des Weiteren kann von bereits definierten Farben die Sättigung, Helligkeit und der Farbton angepasst werden. Sie wurde hauptsächlich zur Farbtonänderung der Strecke basierend auf Geschwindigkeit verwendet.

11.2 Seiten und Elemente

11.2.1 Allgemeine Information

Das Design ist stark von Mobile-Applikationen und Web-Apps inspiriert. Jedes Element ist für unterschiedliche Bildschirmgrößen und Mobilnutzung optimiert.

Inhalte wie etwa Route-Informationen und API-Keys werden via der API geladen, wodurch ohne Neu-Laden der Seite Elemente aktualisiert und verändert werden können. Zum Senden und Empfangen dieser Request wurde die Javascript Fetch-API verwendet.

11.2.2 Navigation

Auf Mobilgeräten sind die Navigationselemente durch ein Hamburgermenü erreichbar. Dropdown und Hamburgermenü besitzen beide eine Animation beim Schließen und Öffnen.



Abb. 40: Desktop Navigation.



Abb. 39: Mobile Navigation.

11.2.3 Routen

Routen werden durch einen API-Request angefordert. Die Abfragezeit dieser Request liegt zwischen 5 bis 10 Millisekunden.

Verwendete API-Endpoints:

api/tours

- GET
- PATCH
- DELETE



Abb. 42: Routenliste Mobilgeräte geöffnet.



Abb. 41 Routenliste geschlossen.

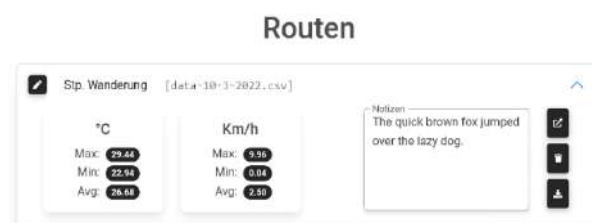


Abb. 43: Routenliste geöffnet.

11.2.4 Geräte

Bei dieser Ansicht wird auf Mobilgeräten die API-Key Vorschau ausgeblendet und eine Änderung an den Buttons vorgenommen.

Umgesetzt wurde dieser Effekt mithilfe des Erstellens von zwei unterschiedlichen Button-Elementen, welche basierend auf Bildschirmbreite ein- und ausgeblendet werden.

Verwendete API-Endpoints:

/api/keys

- GET
- POST
- DELETE
- PATCH



Abb. 45: Geräteansicht Desktop.



Abb. 44: Geräteansicht Mobilgeräte.

11.2.5 Manuelles Hochladen

Bei dieser Seite waren keine spezifischen Anpassungen nötig.

Verwendete API-Endpoints:

/api/upload

- POST

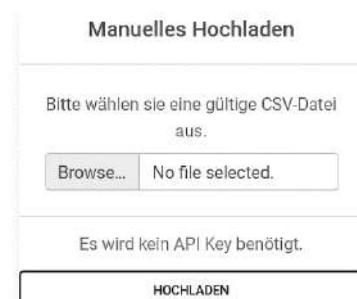


Abb. 46: Manuelles Hochladen.

11.2.6 Nachrichten

Hier sind drei SweetAlert2-Nachrichten aus dem Projekt zu sehen. Ein- und Ausblenden der Elemente sowie die zu sehenden Icons sind animiert.

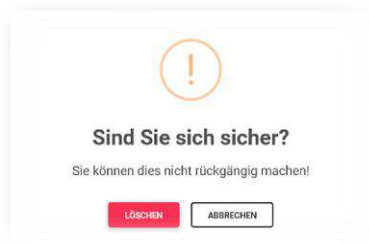


Abb. 48: Namensänderung Aufforderung.

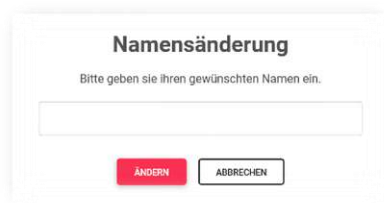


Abb. 49: Bestätigung vor dem Löschen eines Elements.

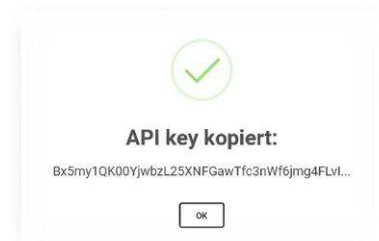


Abb. 47: Bestätigung des Kopierens.

11.2.7 Karte und Visualisierung

Die Visualisierung der Daten wird in drei Ansichten unterteilt: die Kartenansicht, die wirkenden Kräfte und allgemeine Daten.

Beim Aufrufen der Seite ist vorerst die Karte und die aktuelle Route zu sehen. Hierbei ist die Geschwindigkeit durch den Farbverlauf der Strecke sichtbar. Der Farbverlauf basiert auf der maximalen und minimalen Geschwindigkeit der Route und passt sich somit automatisch an diverse Daten an. (Blau: langsam, Gelb: durchschnittlich, Rot: schnell)



Abb. 50: Google Maps Ansicht mit Route.

Durch die links oben zu sehenden Buttons ist der Rest der Datenvisualisierung aufrufbar.

Die Kräfte sind hierbei durch das Selektieren eines gewissen Bereiches wählbar und in Rotation- und Achsenkräfte aufgeteilt. Auf dem Diagramm zum Auswählen des Bereiches ist die Geschwindigkeit der Route zu sehen. Hierbei wird die Achsenskalierung der sechs Diagramme automatisch angepasst, wodurch das Betrachten von starken sowie schwachen Kräften möglich ist.



Abb. 51: Diagramme zur Visualisierung der Kräfte.

Die zweite Ansicht befasst sich mit dem Rest der Daten wie Temperatur und Höhenmeter. Hierbei wurde ebenfalls die Skalierung der Achsen an die Daten angepasst.

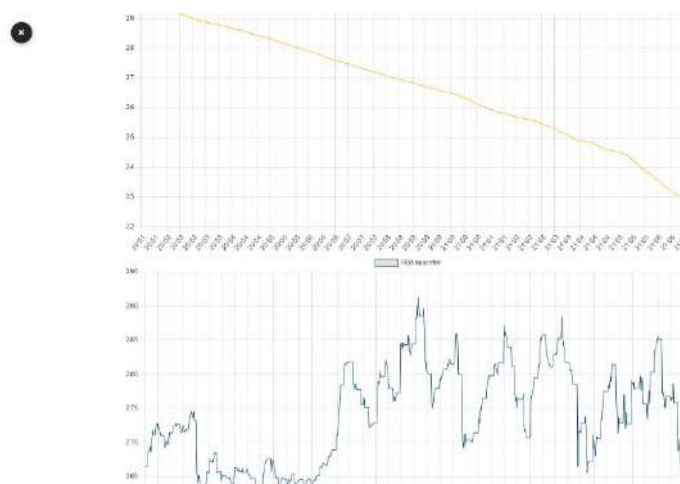


Abb. 52: Diagramme zur Visualisierung diverse Daten.

12 Entwicklungswerkzeuge

12.1 Node Modules

12.1.1 Gulp

Gulp ist ein Automatisierungs-Werkzeug, mit welchem diverse Aufgaben definiert und ausgeführt werden können. Des Weiteren unterstützt Gulp sogenannte „Watch Task“, welche durch Änderungen von Dateien oder ein externes Signal ausgeführt werden.

Funktionen können mithilfe des „pipe()“ Befehls aneinandergereiht, genauer gesagt dessen Ausgabe kann an darauffolgende Funktionen übergeben werden.

```
1  gulp.task('min-css', function () {
2    return gulp.src('./source/private/css/*.scss')
3      .pipe(concat('script.scss'))
4      .pipe(sassCSS().on('error', sassCSS.logError))
5      .pipe(autoprefixerCSS())
6      .pipe(minifyCSS({ compatibility: 'ie8' }))
7      .pipe(rename('style.min.css'))
8      .pipe(gulp.dest('./source/public/'));
9  });
```

Abb. 53: Beispielaufgabe für Gulp.

Mithilfe dieser Funktionalität wurden alle Javascript und CSS-Dateien minimiert und für diverse Webbrowser optimiert.

Frontend Libraries wie MDBootstrap und Chart.js wurden ebenfalls auf diese Art eingebunden. Da sie aber bereits diverse Browseroptimierung beinhalten, wurden sie nur minimiert.

Libraries und Konfigurationsdateien wurden auf zwei Dateien aufgeteilt (styles.min.css und styles.min.lib.css). Grund dafür ist, dass die Libraries vor dem eigentlichen Code eingebunden werden müssen. Dieses Problem könnte auch mithilfe einer Art Definitionsdatei gelöst werden, in welcher alle benötigten Dateien in bestimmter Reihenfolge importiert werden.

Da das Kompilieren der Libraries aufgrund der Größe sehr lange dauert, wurde aber die Aufteilung auf zwei Dateien gewählt. Weiters ermöglicht dieser Weg das Erstellen von neuen Dateien für spezifische Funktionen oder Seiten, welche automatisch eingebunden und verarbeitet werden.

```
1  [01:24:42] Starting 'min-js'...
2  [01:24:43] Finished 'min-js' after 173 ms
3  [01:31:49] Starting 'min-js-lib'...
4  [01:31:57] Finished 'min-js-lib' after 7.97 s
```

Abb. 54: Kompilierzeit der JS-Dateien.

Die Reihenfolge dabei ist in Bezug auf CSS irrelevant, da Styles von Elementen nur jeweils einmal definiert werden oder auf spezifische Elemente beschränkt sind. In Javascript ist die Reihenfolge ebenfalls nicht wichtig, da der Interpreter einen Vorgang namens „Hoisting“ verwendet, welcher das Verwenden einer Funktion vor der Deklaration ermöglicht.

12.1.2 Browseroptimierung

Um den Browsersupport zu verbessern, wurde ein sogenannter „Autoprefixer“ namens PostCSS verwendet. Dieser fügt diverse Anbieter-Präfixe für CSS-Definitionen für Browser wie Firefox, Chrome und Safari hinzu. Diese Präfixe kann man sich wie Erweiterungen zur Verbesserung der definierten Eigenschaften unter den zuvor erwähnten Webbrowsern vorstellen.

Dies zeigt speziell im Bereich Safari und anderen Browsern, basierend auf der WebKitEngine, eine große Verbesserung in der Darstellung und dem Verhalten der Webseite.

```
1  // vor Autoprefixer
2  ::placeholder {
3    color: gray;
4  }
```

Abb. 55: CSS-Beispiel ohne Browseroptimierung.

```
1  // nach Autoprefixer
2  ::-moz-placeholder {
3    color: gray;
4  }
5  :-ms-input-placeholder {
6    color: gray;
7  }
8  ::placeholder {
9    color: gray;
10 }
```

Abb. 56: CSS-Beispiel mit Browseroptimierung

12.1.3 SCSS

SCSS kann man sich als CSS-Erweiterung ähnlich wie TypeScript zu Javascript vorstellen.

Es ermöglicht die bessere Verschachtelung von Definitionen und Verwalten von Variablen, sowie Schleifen.

Zum Kompilieren auf CSS wurde Gulp verwendet. Hier ist ein Beispiel zur Verschachtelung und Übergabe von Elementen zu sehen.



```
CSS
1  .info {
2    background: DarkGray;
3    box-shadow: 0 0 1px rgba(169, 169, 169, 0.25);
4    color: #fff;
5  }
6
7  .alert {
8    background: DarkRed;
9    box-shadow: 0 0 1px rgba(139, 0, 0, 0.25);
10   color: #fff;
11 }
12
13 .success {
14   background: DarkGreen;
15   box-shadow: 0 0 1px rgba(0, 100, 0, 0.25);
16   color: #fff;
17 }
```

Abb. 58: CSS-Beispiel.



```
SCSS
1  @mixin theme($theme: DarkGray) {
2    background: $theme;
3    box-shadow: 0 0 1px rgba($theme, .25);
4    color: #fff;
5  }
6
7  .info {
8    @include theme;
9  }
10 .alert {
11   @include theme($theme: DarkRed);
12 }
13 .success {
14   @include theme($theme: DarkGreen);
15 }
```

Abb. 57: SCSS-Beispiel mit Übergabe von Werten.

12.1.4 ESLint und JSHint

Die zwei Module wurden zur Validierung und Kontrolle aller Javascript-Dateien verwendet. In Kombination mit den dazugehörigen IDE-Extensions können die Informationen direkt bei der Entwicklung angezeigt werden.

Ein Beispiel wäre, wenn eine Variable nach der Definition nicht verwendet wird, bekommt man eine Information darüber. Weiters werden Richtlinien über zum Beispiel Funktions-Verschachtelung und diverse andere „Best Practices“ angezeigt.

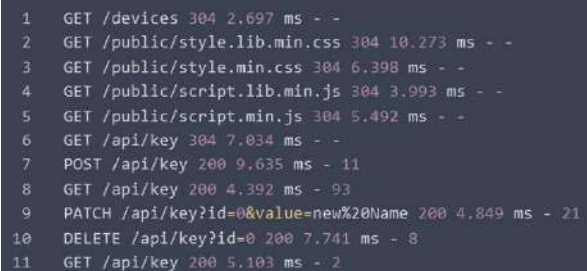
Diese Module sind aber in keiner Weise verpflichtend, sie erleichtern nur die Entwicklung, da der Programmcode einheitlicher und leichter zu lesen wird.

Konfigurationsdatei: eslintrc.js, jshintrc

12.1.5 Morgan

Morgan ist ein Node Module zur Überwachung von HTTP-Requests.

Für dieses Projekt wurde der „dev“ Modus verwendet, wobei die Request-Methode, URL, Status, Antwortgeschwindigkeit und Länge der gesendeten Daten angezeigt werden. Dies war speziell bei der Entwicklung des Frontends aufgrund der diversen API-Zugriffe sehr hilfreich. Auch im Bereich Performance bietet das Modul einen guten Einblick. Bei Fehlern, welche zu Abstürzen des Servers führten, wurde das Debuggen ebenfalls erleichtert, da die zuletzt aufgerufene URL vor dem Absturz ausgegeben wird.



```
1 GET /devices 304 2.697 ms - -
2 GET /public/style.lib.min.css 304 10.273 ms - -
3 GET /public/style.min.css 304 6.398 ms - -
4 GET /public/script.lib.min.js 304 3.993 ms - -
5 GET /public/script.min.js 304 5.492 ms - -
6 GET /api/key 304 7.034 ms - -
7 POST /api/key 200 9.635 ms - 11
8 GET /api/key 200 4.392 ms - 93
9 PATCH /api/key?id=0&value=new%20Name 200 4.849 ms - 21
10 DELETE /api/key?id=0 200 7.741 ms - 8
11 GET /api/key 200 5.103 ms - 2
```

Abb. 59: Morgan Ausgabe bei diversen Aufrufen der Website.

12.1.6 Nodemon

Um Änderungen des Programmcodes zu vereinfachen, wurde das Node Module Nodemon verwendet. Es ermöglicht die Überwachung von Konfigurationsdateien und serverrelevanten Dateien sowie Ordnern und initialisiert bei Änderungen einen Neustart des Servers.

12.1.7 Jest

Jest ist ein Testing-Framework, spezialisiert auf Javascript. Es ermöglicht das einfache Definieren von Tests diverser Funktionen in einer „test.js“ Datei.

Speziell das Verhalten von Funktionen bei Eingabe invalider Werten konnte dadurch schnell kontrolliert werden. Es ermöglichte beispielsweise die Definition von 10 Tests mit unterschiedlichen Eingangsparametern.

Im Bereich der Datenbank wurde unter anderem ein Eintrag automatisch erstellt, ausgelesen und gelöscht.



```
1 test('api_key delete non existing', () => {
2   expect(api_keys_delete(999)).toBe(NULL);
3 });
```

Abb. 60: Jest Testbeispiel einer Funktion.

12.2 GUI-Anwendungen

12.2.1 Prisma Studio

Zum Überwachen der Datenbank und diverser Daten wurde Prisma Studio verwendet. Es ähnelt in vieler Hinsicht Werkzeugen wie phpMyAdmin, ist jedoch auf das Darstellen beziehungsweise Bearbeitung von Daten und weniger auf Konfiguration der Datenbank spezialisiert. Besonders im Bereich Datenrelation war dies sehr hilfreich, da man sofort erkannte, welche und wie viele Einträge zu einem Nutzer gehören.

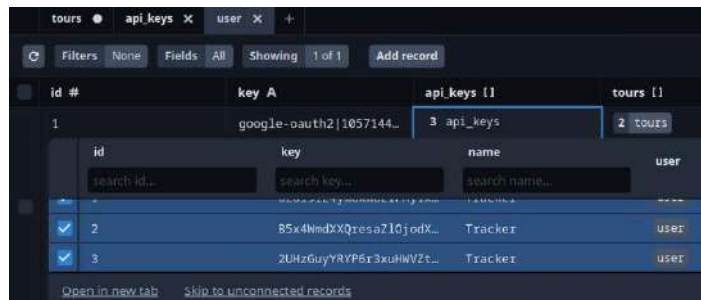


Abb. 61 Prisma Studio Ansicht eines Nutzers in der Datenbank.

Da das Node Module von Prisma dieses Tool bereits enthält, war keine extra Konfiguration oder Installation nötig.

12.2.2 Portainer

Portainer bietet ein grafisches Userinterface zur Überwachung und Konfigurierung der diversen Container. Die Installation erfolgt über einen Docker-Container, welcher Zugriff auf den Docker-Socket des Betriebssystems hat. Es gibt auch die Möglichkeit, mehrere Sockets von zum Beispiel entfernten Servern über sogenannte „Agents“ hinzuzufügen. Es besteht auch die Möglichkeit, Service wie Kubernetes und Azure hinzuzufügen.

In dem Webinterface ist unter anderem das Überwachen der MySQL und Node.js Logs möglich. Weiters kann direkt eine Verbindung zu der Konsole eines Containers hergestellt werden, sofern dieses in der Konfiguration erlaubt ist. Verwendete Ressourcen der Container wie, RAM, CPU und Netzwerknutzung können ebenfalls überwacht werden.

Speziell im Bereich der Umgebungsvariablen und Netzwerkeinstellungen, war die Möglichkeit Änderungen vorzunehmen und zu kontrollieren sehr hilfreich.

12.2.3 RESTEDClient

Bei der Entwicklung der API wurde eine Firefox Extension namens RESTEDClient verwendet. Sie ermöglichte einfaches Senden von API-Requests mit diversen Eingabeparametern und Speichern alter Requests.

Anfangs wurde Insomnia verwendet, der Wechsel zu RESTEDClient war aufgrund diverser Authentifizierungsprobleme mit Auth0 notwendig.

12.2.4 Visual Studio Code

Als IDE wurde Visual Studio Code (VS Code) verwendet. VS Code unterstützt diverse Programmiersprachen und ist mithilfe von Extensions erweiter- und veränderbar.

Im Bereich Zeitaufzeichnung wurde eine Extension namens WakaTime verwendet, welche die automatische Zeitaufzeichnung individueller Git-Commits ermöglicht. Auch Statistiken über Bearbeitungszeit einzelner Datei und verwendeter Programmiersprachen sind aufrufbar.

Um die Konfigurierung des VPS zu erleichtern, wurde eine Extension namens "Remote - SSH" verwendet, welche das direkte Verbinden zu Ordnern und Dateien über SSH ermöglicht. Einstellungen am Proxyserver konnten dadurch ebenfalls leichter vorgenommen werden. Weiters wurden Extensions für zum Beispiel Prisma, JSHint, JSLint und Git verwendet.

12.3 Installation & Update Script

Um das Updaten des VPS zu erleichtern, wurde ein Bash-Script geschrieben, welches die lokale Version des Projekts mit der aktuellen Version auf Github vergleicht und basierend darauf ein Update initialisiert. Um die Entwicklung zu erleichtern, gibt es vier Optionen, welche das Verhalten des Scripts ändern. Die Option zum Stoppen des Node.js-Containers wurde hinzugefügt, da in der Entwicklungsphase Node.js außerhalb des Containers verwendet wurde.

-h	Ausgabe der verfügbaren Optionen.
-f	Docker Container werden ohne Versionskontrolle aktualisiert.
-d	Den Port 3306 am MySQL Container Lokal freigeben.
-s	Den Node.js Container nach dem Erstellen stoppen

Tabelle 4: Optionen Installation und Update Scripts

13 Ergebnisse von Tests

13.1 Ladezeit der Karte

Anfangs wurden die Routen-Daten mithilfe eines API-Endpoints geladen und dann auf der Karte beziehungsweise den Diagrammen visualisiert. Dies verursachte eine Verzögerung von circa 500 bis 900 Millisekunden zwischen dem Anzeigen der Karten und dem Erscheinen der Werte. Auch mit dem parallelen Ausführen dieser zwei Vorgänge konnte die Wartezeit nur minimal verbessert werden.

Um diese Ladezeit zu reduzieren, wurde eine Art Server-Side-Rendering verwendet, wobei die Werte direkt beim Aufruf der Website in den HTML-Code eingebunden werden. Hiermit wurde zwar die Zeit des initialen Request auf circa 200 Millisekunden verlängert, die allgemeine Ladezeit wurde aber merkbar reduziert.

13.2 API-Request Ladezeit

Um laufende API-Requests zu visualisieren, wurde oben rechts in der Navigationsleiste ein animiertes Ladesymbol hinzugefügt. Des Weiteren werden während dieses Vorgangs diverse Eingaben des Nutzers ignoriert, um das Senden von zum Beispiel: fünfmal denselben Request zu verhindern.



Abb. 62: Ladesymbol der API-Requests.

Diese zwei Funktionen sind bei einer guten Internetanbindung nicht bemerkbar, bei jedoch schlechter Verbindung sehr hilfreich, da der Nutzer Feedback zu seinen Eingaben bekommt.

13.3 Erstellen der Docker Container

Der MySQL-Container benötigt circa fünf Sekunden für diverse Initialisierungen, bevor Zugriffe auf die Datenbank möglich sind. Dies hatte zur Folge, dass der Node.js-Container vor Verfügbarkeit der Datenbank das Prisma-Schema kontrollieren beziehungsweise migrieren wollte. Um dieses Problem zu lösen, wurde ein Tool namens: „docker-compose-wait“ verwendet. Dieses Tool ermöglicht das Kontrollieren der Datenbank Verfügbarkeit, bevor diverse Zugriffe stattfinden.

14 Zukunftspläne

14.1 Log-in

Im Laufe der Entwicklung wurde eine selbst hostbare Alternative zu Auth0 namens SuperTokens gefunden. Diese würde die Abhängigkeit von Auth0 eliminieren und mehr Kontrolle über das Aussehen und Verhalten der Log-in und Registrierungsseiten bieten.

Da SuperTokens dieselbe Funktionalität bietet, können weiterhin Service wie etwa Google zur Anmeldung verwendet werden.

Ein Nachteil dabei wäre die zusätzliche Belastung des Servers bei einer hohen Anzahl von Nutzern. Für dieses Projekt sollte dies aber kein Problem sein, da es nicht als kommerzielles Produkt gedacht ist.

14.2 Map Visualisierung

Die Map-Visualisierung ist zurzeit hauptsächlich für Desktop-Computer und Laptops optimiert. Um bessere Bedienung für Mobilgeräte zu bieten, müssten diverse Untermenüs und Diagramme anders dargestellt beziehungsweise aufgebaut werden.

14.3 Desktop App

Der Umbau auf eine Desktop-App mithilfe des Electron Frameworks wäre ebenfalls eine Option: Hierbei könnte unter anderem das direkte Öffnen diverser Dateiformate möglich sein und eine Art universelles GPS-Visualisierungsprogramm erstellt werden. Da Electron ebenso auf Node.js basiert, könnte ein Großteil des geschriebenen Programmcodes weiterverwendet werden.

Ein Problem dabei ist der Google API-Key, da dieser nicht auf eine URL beschränkt werden kann und dadurch von jedem verwendbar wäre. Ein Lösungsansatz dafür wäre der Wechsel zu einem Service wie Openlayers, wo kein API-Key zur Verwendung benötigt wird. Hierbei müsste man nur die Kartendarstellung anpassen, diverse Diagramme könnten direkt übernommen werden.

15 Ergebnisse

Der Datenlogger sowie die Website sind voll funktionsfähig. Der Datenlogger kann auch bei Schlechtwetter ohne Bedenken eingesetzt werden. Daten können über ein vom Nutzer konfiguriertes Netzwerk hochgeladen werden. Diese Daten können nach dem Upload auf der Website mithilfe einer Karte und Diagrammen visualisiert werden.

16 Betriebswirtschaftliche Kalkulation

16.1 Entwicklungskosten

$$\text{Hardware} = 91 \text{ h} * 60 \frac{\text{€}}{\text{h}} = 5.460 \text{ €}$$

$$\text{Software} = 105 \text{ h} * 53 \frac{\text{€}}{\text{h}} = 5.565 \text{ €}$$

$$\text{Bauteile} = 118 \text{ €}$$

$$\text{Gesamtkosten} = 11.143 \text{ €}$$

16.1.1 Materialkosten

Komponente	Menge	Preis inkl. Ust.	Preis exkl. Ust.
ESP32-DevKitC V4	1	8,40 €	7,- €
MPU6050	1	3,02 €	2,52 €
MicroSD breakout board+	1	7,82 €	6,52 €
Beitian BN-220	1	16,69 €	13,91 €
Intenso 8GB SD-Karte	1	6,35 €	5,29 €
SD push-push Slot	1	1,72 €	1,43 €
Mikroe Li-Po 3k mAh 3,7 V	1	12,76 €	10,63 €
SMD-Taster	1	2,71 €	2,26 €
RGB-LED	3	0,60 €	0,50 €
3D Filament PETG 19,40 €/kg	235g	4,56 €	3,80 €
Leiterplatte	1	0,40 €	0,33 €
Versand	/	29,50 €	24,58 €
Gesamt	/	94,53 €	78,77 €

Tabelle 5: Materialkosten

16.2 Produktionskosten

$$\text{Zusammenbauen} = 1 \text{ h} * 40 \frac{\text{€}}{\text{h}} = 40 \text{ €}$$

16.3 Verkaufspreis

$$\text{Verkaufspreis} = 118,77 \text{ €} + 30 \text{ €} = 148,77 \text{ €} \approx 149 \text{ €}$$

16.4 Break-Even-Point

$$BEP = \frac{11.143 \text{ €}}{149 \text{ €} - 118,77 \text{ €}} \approx 369 \text{ Stück}$$

17 Literaturverzeichnis

- (12. 12 2021). Von QuadMeUp: <https://quadmeup.com/how-to-connect-gps-to-esp32/> abgerufen
- (03. 12 2021). Von Randomnerdtutorials: <https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/> abgerufen
- (03. 12 2021). Von Randomnerdtutorials: <https://randomnerdtutorials.com/esp32-data-logging-temperature-to-microsd-card/> abgerufen
- (14. 12 2021). Von Randomnerdtutorials: <https://randomnerdtutorials.com/esp32-save-data-permanently-preferences/#example2> abgerufen
- (14. 12 2021). Von Randomnerdtutorials: <https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/> abgerufen
- (10. 1 2021). Von Caddy2 Reverse Proxy: <https://caddyserver.com/docs/quick-starts/reverse-proxy> abgerufen
- (20. 12 2021). Von Stack Overflow "CSV Parsing": <https://stackoverflow.com/questions/53031531/how-can-i-convert-from-csv-to-array-json-string-in-node-js/53031629#53031629> abgerufen
- (9. 12 2021). Von Material Design for Bootstrap: <https://mdbootstrap.com/docs/standard/> abgerufen
- (3. 12 2021). Von Google Maps Platform Documentation: <https://developers.google.com/maps/documentation> abgerufen
- (19. 12 2021). Von Devdocs Express: <https://devdocs.io/express/> abgerufen
- (20. 11 2021). Von Chart.js: <https://www.chartjs.org/docs/2.7.3/> abgerufen
- (29. 12 2021). Von Prisma: <https://www.prisma.io/docs/> abgerufen
- (29. 11 2021). Von SweetAlert2: <https://sweetalert2.github.io/#input-types> abgerufen
- (1. 12 2021). Von Mapbox: <https://www.mapbox.com/> abgerufen
- (11. 12 2021). Von Fontawesome Suche: <https://fontawesome.com/v5/search> abgerufen
- (19. 11 2021). Von Leafletjs: <https://leafletjs.com/> abgerufen
- (2. 11 2021). Von Visual Studio Code: <https://code.visualstudio.com/> abgerufen
- (5. 12 2021). Von Insomnia: <https://insomnia.rest/> abgerufen
- (11. 1 2022). Von Auth0 Express: <https://auth0.com/docs/quickstart/webapp/express> abgerufen
- (10. 3 2022). Von GitHub "docker-compose-wait": <https://github.com/ufoscout/docker-compose-wait/> abgerufen
- (21. 2 2022). Von GitHub "Prisma Null constraint violation": <https://github.com/prisma/migrate/issues/498> abgerufen
- (14. 1 2022). Von Chroma.js: <https://gka.github.io/chroma.js/> abgerufen
- (20. 1 2022). Von SASS: <https://sass-lang.com/> abgerufen
- (13. 2 2022). Von Npmjs "gulp-documentation": <https://www.npmjs.com/package/gulp-documentation> abgerufen
- (14. 3 2022). Von Portainer: <https://www.portainer.io/> abgerufen
- (24. 3 2022). Von Jest: <https://jestjs.io/> abgerufen
- (29. 1 2022). Von Prisma: <https://www.prisma.io/studio> abgerufen

- (11. 1 2022). Von Npmjs "nodemon": <https://www.npmjs.com/package/nodemon> abgerufen
- (11. 1 2022). Von Npmjs: <https://www.npmjs.com/package/morgan> abgerufen
- (27. 2 2022). Von RESTEDClient: <https://github.com/RESTEDClient/RESTED> abgerufen
- Hart, M. (12. 12 2021). Von Arduiniana: <http://arduiniana.org/libraries/tinygpsplus/> abgerufen
- Heiko. (14. 12 2021). Von Unsinnsbasis: <https://unsinnsbasis.de/esp32-preferences/> abgerufen

18 Verzeichnis der Abbildungen, Tabellen und Begriffe

18.1 Abbildungsverzeichnis

Abb. 1: Der ESP32 von Espressif Quelle: https://www.mouser.at/images/espressifsystems/lrg/ESP32-DEVKITC.jpg	6
Abb. 2: SD-Karten-Schaltung auf der Hauptplatine.....	7
Abb. 3: Der MPU6050 aufrechtstehend montiert.....	7
Abb. 4: Das BN-220 von Beitian Quelle: https://m.media-amazon.com/images/I/711BvTrb-FL_AC_SX569_.jpg	8
Abb. 5: Die Ladeschaltung für den Akku auf der Hauptplatine.....	8
Abb. 6: Die verwendete Lithium-Polymer-Zelle.....	9
Abb. 7: Der Unterteil des Gehäuses	9
Abb. 8: Das Gehäuse, betrachtet von oben.....	9
Abb. 9: Aufbau des ersten Prototyps auf dem Steckbrett.....	10
Abb. 10: Die Beschaltung des Spannungsreglers	13
Abb. 11: Das Schaltbild des Spannungsteilers	14
Abb. 12: Code-Ausschnitt, in welchem die einzelnen Pins der LED entsprechend des Ladezustands gesetzt werden.....	16
Abb. 13: Der Teil des Codes der für die Erstellung der Datei zuständig ist	17
Abb. 14: Flow-Chart des Programmablaufs	19
Abb. 15: Beschaltung des Tasters	20
Abb. 16: Code-Ausschnitt, in welchem die eingegebenen Daten in den persistenten Speicher des ESP32 geschrieben werden	21
Abb. 17: Code-Ausschnitt, in welchem der Upload über einen POST-Request an den Webserver erfolgt	22
Abb. 18: Die Oberseite des Lademoduls ohne Bauteile.....	23
Abb. 19: Die Unterseite der Ladeplatine	23
Abb. 20: Der gezeichnete Schaltplan der Ladeplatine	25
Abb. 21: Die Oberseite des SD-Karten-Moduls ohne Bauteile.....	26
Abb. 22: Der resultierende Schaltplan aus Abb. 18	26
Abb. 23: Der Hauptschaltplan mit weiteren Komponenten wie dem Taster und den LEDs....	28
Abb. 24: Der Top-Layer der entflochtenen Platine	29
Abb. 25: Der Bottom-Layer der entflochtenen Platine.....	29
Abb. 26: Unterteil des Gehäuses mit eingesetztem Akku	33
Abb. 27: Test der Funktionstauglichkeit vor dem Verschrauben der GPS-Halterung	33
Abb. 28: Der fertig bestückte Unterteil des Gehäuses.....	33
Abb. 29: Die GPS-Halterung.....	33
Abb. 30: Der Gehäusedeckel mit Dichtung und Schrauben	34
Abb. 31: Finaler, funktionstauglicher Prototyp	34
Abb. 32: Der Datenlogger, montiert und einsatzbereit.....	34
Abb. 33: Mapbox Test.	35
Abb. 34: Google Maps Test.....	35
Abb. 35: Serveraufbau grafisch dargestellt.	36
Abb. 36: Beispielausgabe „api/key/“ GET.	39

Abb. 37: Beispielausgabe „api/tours/“ GET.	40
Abb. 38: Beispielausgabe „api/tours/“ GET mit Datei.	40
Abb. 39: Mobile Navigation.	44
Abb. 40: Desktop Navigation.....	44
Abb. 41 Routenliste geschlossen.	44
Abb. 42: Routenliste Mobilgeräte geöffnet.....	44
Abb. 43: Routenliste geöffnet.	44
Abb. 44: Geräteansicht Mobilgeräte.	45
Abb. 45: Geräteansicht Desktop.	45
Abb. 46: Manuelles Hochladen.....	45
Abb. 47: Bestätigung des Kopierens.....	46
Abb. 48: Namensänderung Aufforderung.....	46
Abb. 49: Bestätigung vor dem Löschen eines Elements.....	46
Abb. 50: Google Maps Ansicht mit Route.....	46
Abb. 51: Diagramme zur Visualisierung der Kräfte.	47
Abb. 52: Diagramme zur Visualisierung diverse Daten.	47
Abb. 53: Beispielaufgabe für Gulp.	48
Abb. 54: Kompilierzeit der JS-Dateien.....	49
Abb. 55: CSS-Beispiel ohne Browseroptimierung.	49
Abb. 56: CSS-Beispiel mit Browseroptimierung.....	49
Abb. 57: SCSS-Beispiel mit Übergabe von Werten.	50
Abb. 58: CSS-Beispiel.....	50
Abb. 59: Morgen Ausgabe bei diversen Aufrufen der Website.....	51
Abb. 60: Jest Testbeispiel einer Funktion.....	51
Abb. 61 Prisma Studio Ansicht eines Nutzers in der Datenbank.....	52
Abb. 62: Ladesymbol der API-Requests.....	54

18.2 Tabellenverzeichnis

Tabelle 1: Meilensteine.....	2
Tabelle 2: Individueller Terminplan Dominic Frostl.....	2
Tabelle 3: Individueller Terminplan Manuel Gschwandtner.....	3
Tabelle 4: Optionen Installation und Update Scripts.....	53
Tabelle 5: Materialkosten.....	57
Tabelle 6: Begleiterprotokoll	65

18.3 Begriffsverzeichnis und Abkürzungen Server

18.3.1 Allgemeine Begriffe

BE / Backend	Serverseitiger Code, welcher nicht für den Nutzer sichtbar ist.
FE / Frontend	Grafisches Interface, mit dem der Nutzer interagiert.
Framework	Vorgefertigte Funktionen und Elemente mit diversen Anpassungsoptionen.
Library	Vorgefertigter Programmcode, um Aufgaben zu erleichtern und zu optimieren.

18.3.2 Backend

API	Interface zum Austausch von Daten zwischen zwei Bestandteilen oder Geräten.
Middleware	Software zum Datenaustausch und Datenübergabe innerhalb eines Programmes.
Node Module	Erweiterungsmodul für die Node.js Runtime.
VPS	Virtualisierter Server in einem Rechenzentrum, welcher über eine öffentliche IP erreichbar ist.

18.3.3 Docker

Containerisierung	Verpacken von Software und erforderlichen Komponenten in einem isolierten Container.
Docker-Volume	Persistentes Speichern auf den ein spezifischer Container zugreifen kann.

18.3.4 Prisma

ORM	Verkapslung von Daten und Funktionen in Objekten.
Schema	Beschreibung des Aufbaus von Daten und deren Beziehung.
Typesafe	Garantie und Kontrolle, dass ein gewünschter Datentyp gegeben ist.

18.3.5 API

API-Endpoint	Eine Endung des Kommunikationskanals einer API.
Request-Header	Kontext und Zusatzinformation zu einem API-Request.

18.3.6 Frontend

Responsive	Anpassung für Mobilgeräte (Smartphone, Tablet)
UI	Design und Erscheinungsbild der Website

18.3.7 Entwicklungswerkzeuge

IDE	Entwicklungsumgebung, welche gängige Entwicklertools in eine grafische Oberfläche vereint.
-----	--

19 Begleitprotokoll gemäß § 9 Abs. 2 PrO

KW	Kandidat	Tätigkeit	Zeit [h]
35 / 2021	Dominic Frostl	Planung des Projektkonzepts; Bauteilwahl	5
41 / 2021	Dominic Frostl	Steckbrettaufbau der Prototypen	8
42 / 2021	Dominic Frostl	Recherche für die Entwicklung der Testsoftware sowie erste Umsetzung der Testsoftware	5
43 / 2021	Dominic Frostl	Erweiterung der Testsoftware	7
44 / 2021	Dominic Frostl	Anpassung der Schaltung und Implementierung RGB-LEDs	4
46-49 / 2021	Dominic Frostl	Fertigstellung des Prototyps auf dem Steckbrett; Erweiterung der Software	11
47-52 / 2021	Manuel Gschwandtner	Erstellung des Prototyps; testen diverse Maps Anbietern; testen unterschiedlicher Datenbanken	16
51 / 2021	Dominic Frostl	Finalisierung der Software	4
1-3 / 2022	Dominic Frostl	PCB-Design; Analyse Gyroskop; Analyse SD-Karten-Modul; Bestellung der Leiterplatte; Entwerfen des Gehäuses	26
1-3 / 2022	Manuel Gschwandtner	Git Einrichtung, Installation; Konfiguration einiger Node Modules	8
4 / 2022	Manuel Gschwandtner	Einrichten von Prisma; Datenbankstruktur; Erstellen von NPM-Scripts	4
5 / 2022	Manuel Gschwandtner	API-Key Verwaltung erstellt; Auth0 eingebunden und getestet	9
6 / 2022	Dominic Frostl	Fertigstellung Gehäuse; Fehlerkorrekturen der Leiterplatte; Zusammenbau; Erste Montage am Motorrad	21

6 / 2022	Manuel Gschwandtner	API-Key Verwaltung angepasst; Uploaden von Daten ohne Log-in eingebunden	5
7 / 2022	Manuel Gschwandtner	Style Anpassungen: Buttons und Inputs	1
8 / 2022	Manuel Gschwandtner	Umstrukturierung des Codes; Aufteilung in: Private, Public, Datenbank & Filesystem	3
9 / 2022	Manuel Gschwandtner	ESLint hinzugefügt + Anpassung des Programmcodes; Navigation umgebaut; Kartenansicht geändert und Farbe basierend auf Geschwindigkeit eingebaut	5
10 / 2022	Manuel Gschwandtner	Animationen hinzugefügt; JSHint hinzugefügt, Ladesymbol und Input Blockierung erstellt; Gulp: Autopräfix und SCSS angepasst	16
11 / 2022	Manuel Gschwandtner	SweetAlert2 hinzugefügt und zu allen Elementen hinzugefügt; neue Ansicht für API- Keys und Routen erstellt; Anpassungen für Mobilgeräte	21
12 / 2022	Manuel Gschwandtner	Kartenansicht angepasst und Lightbox hinzugefügt; Docker-Aufbau geändert, Installation / Update Script erstellt; Website auf Live-Server installiert	9
13 / 2022	Manuel Gschwandtner	Kartenansicht: Diagramme angepasst; Kartenansicht selektierbaren Bereich hinzugefügt	8

Tabelle 6: Begleiterprotokoll

20 Diplomandenseminare

20.1 Protokoll zum 1. Diplomandenseminar

„Datenlogger für Motorräder“

Termin: 29.11.2021, 16:40, HTBLuVA St. Pölten

Teilnehmer: Ing. Marc Prantl, Manuel Gschwandtner, Dominic Frostl

Name	Notizen
Kandidaten	Statusbericht über laufende Arbeiten, derzeitige Herausforderungen
Betreuer	Nächstes Seminar am 24.01.2021

FORTSCHRITT DER ARBEITEN:

Dominic Frostl:

Abgeschlossene Aufgaben	Erledigt am
Konzepterstellung, AFP/Pflichtenheft	06.09.2021
Bauteile rechtzeitig bestellt und geliefert kritisch: neuer SD-Cardreader kommt erst am 30.11.2021	04.10.2021
Aufbau Prototyp: Kommunikation mit Sensoren und GPS OK Daten auf SD-Karte speichern noch ausständig	20.11.2021

Aufgabe	Fällig bis
HW-Prototyp finalisieren, SD-Karte einbinden	06.12.2021

Manuel Gschwandtner:

Abgeschlossene Aufgaben	Erledigt am
Vorarbeiten für SW	laufend
Express.js-API (Upload-Test), CSV-Parser für Datenimport	20.11.2021

Aufgabe	Fällig bis
Testsoftware fertigstellen	27.12.2021

20.2 Protokoll zum 2. Diplomandenseminar**„Datenlogger für Motorräder“**

Termin: 24.01.2022, 16:50, HTBLuVA St. Pölten

Teilnehmer: Ing. Marc Prantl, Manuel Gschwandtner, Dominic Frostl

Name	Notizen
Kandidaten	Statusbericht über laufende Arbeiten, derzeitige Herausforderungen
Betreuer	Nächstes Seminar am 28.02.2022

FORTSCHRITT DER ARBEITEN:**Dominic Frostl:**

Abgeschlossene Aufgaben	Erledigt am
Software fertig	27.12.2021
PCB Design entworfen und bestellt	20.01.2022
PCB bestückt und funktionstauglich kritisch: Lieferverzug PCB wegen chinesischem Neujahr	laufend
Gehäusedesign fertiggestellt	laufend
Einbau, Funktionstests	laufend

Aufgabe	Fällig bis
Abdichtung Gehäuse	07.02.2022

Manuel Gschwandtner:

Abgeschlossene Aufgaben	Erledigt am
CSV-Datei Ausgabe auf Karte (Testsoftware)	20.12.2021
Datenbank Einrichtung: Prisma	4.01.2022

Aufgabe	Fällig bis
Website online	28.03.2022

20.3 Protokoll zum 3. Diplomandenseminar

„Datenlogger für Motorräder“

Termin: 28.02.2022, 16:50, HTBLuVA St. Pölten

Teilnehmer: Ing. Marc Prantl, Manuel Gschwandtner, Dominic Frostl

Name	Notizen
Kandidaten	Statusbericht über laufende Arbeiten, derzeitige Herausforderungen
Betreuer	Nächstes Seminar am 28.03.2022

FORTSCHRITT DER ARBEITEN:

Dominic Frostl:

Abgeschlossene Aufgaben	Erledigt am
Gehäuse gedruckt	02.02.2022
Abdichtung Gehäuse	07.02.2022
PCB bestückt	14.02.2022
Montagevorrichtung	16.02.2022

Aufgabe	Fällig bis

Manuel Gschwandtner:

Abgeschlossene Aufgaben	Erledigt am
Docker: Anpassung, Optimierung	28.01.2022
Auth0 Anmeldung eingebunden	3.02.2022
Website Elemente für: API-Keys, Manuelles Hochladen	20.02.2022

Aufgabe	Fällig bis
Website online	28.03.2022

20.4 Protokoll zum 3. Diplomandenseminar

„Datenlogger für Motorräder“

Termin: 28.03.2022, 16:50, HTBLuVA St. Pölten

Teilnehmer: Ing. Marc Prantl, Manuel Gschwandtner, Dominic Frostl

Name	Notizen
Kandidaten	Statusbericht über laufende Arbeiten, derzeitige Herausforderungen
Betreuer	---

FORTSCHRITT DER ARBEITEN:

Manuel Gschwandtner:

Abgeschlossene Aufgaben	Erledigt am
Kartenansicht: Diagramme + selektierbarer Bereich, Routenübersicht	28.02.2022
API-Änderungen, Design-Änderungen (SweetAlert2 und Animationen)	2.03.2022
Website Online	27.03.2022

Aufgabe	Fällig bis
Website online	28.03.2022

21 Anhang

Die für dieses Projekt verwendeten Datenblätter, befinden sich auf dem beigelegten Datenträger.

ESP32-WROOM-32

Datasheet

NOT RECOMMENDED
FOR NEW DESIGNS
(NRND)



Version 3.3
Espressif Systems
Copyright © 2022

1 Overview

ESP32-WROOM-32 is a powerful, generic Wi-Fi + Bluetooth + Bluetooth LE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

At the core of this module is the ESP32-D0WDQ6 chip*. The chip embedded is designed to be scalable and adaptive. There are two CPU cores that can be individually controlled, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz. The chip also has a low-power coprocessor that can be used instead of the CPU to save power while performing tasks that do not require much computing power, such as monitoring of peripherals. ESP32 integrates a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I2S and I2C.

Note:

* For details on the part numbers of the ESP32 family of chips, please refer to the document [ESP32 Datasheet](#).

The integration of Bluetooth®, Bluetooth LE and Wi-Fi ensures that a wide range of applications can be targeted, and that the module is all-around: using Wi-Fi allows a large physical range and direct connection to the Internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5 μ A, making it suitable for battery powered and wearable electronics applications. The module supports a data rate of up to 150 Mbps, and 20 dBm output power at the antenna to ensure the widest physical range. As such the module does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity.

The operating system chosen for ESP32 is freeRTOS with LwIP; TLS 1.2 with hardware acceleration is built in as well. Secure (encrypted) over the air (OTA) upgrade is also supported, so that users can upgrade their products even after their release, at minimum cost and effort.

Table 1 provides the specifications of ESP32-WROOM-32.

Table 1: ESP32-WROOM-32 Specifications

Categories	Items	Specifications
Certification	RF certification	See certificates for ESP32-WROOM-32
	Wi-Fi certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Test	Reliability	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μ s guard interval support
	Center frequency range of operating channel	2412 ~ 2484 MHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and Bluetooth LE specification
	Radio	NZIF receiver with -97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH

[Not Recommended For New Designs \(NRND\)](#)

2 Pin Definitions

2.1 Pin Layout

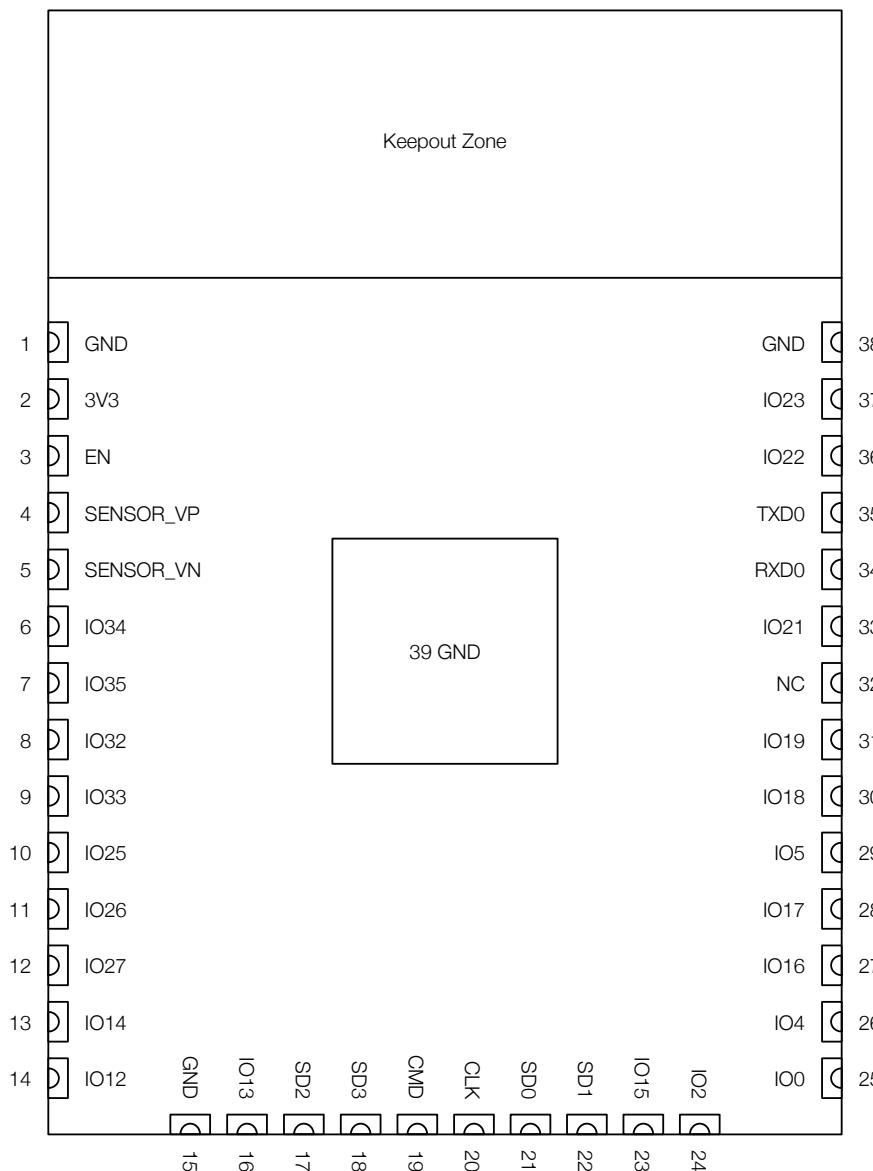


Figure 1: ESP32-WROOM-32 Pin Layout (Top View)

2.2 Pin Description

ESP32-WROOM-32 has 38 pins. See pin definitions in Table 2.

Table 2: Pin Definitions

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply
EN	3	I	Module-enable signal. Active high.

[Not Recommended For New Designs \(NRND\)](#)

Name	No.	Type	Function
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Ground
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2*	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3*	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD*	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/CLK*	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0*	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1*	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPICLK, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Ground

BN-220

GPS Module + Antenna

DataSheet

Revision: 5.0

Date:2015.10



BN-220 GPS Module + Antenna Datasheet

Features:

Parameter	Specification	
Electrical Characteristics	Chipset	u-blox M8030-KT
	Receiving Format	GPS,GLONASS,Galileo,BeiDou,QZSS and SBAS
	Frequency	GPS L1, GLONASS L1, BeiDou B1, SBAS L1, Galileo E1
	Channels	72 Searching Channel
Sensitivity	Tracking	-167dBm
	Reacquisition	-160dBm
	Cold start	-148dBm
	Hot start	-156dBm
Accuracy	Position Horizontal	2.0 m CEP 2D RMS SBAS Enable (Typical Open Sky)
	Velocity	0.1m/sec 95% (SA off)
	Timing	1us synchronized to GPS time
Acquisition Time	Cold Start	26s
	Warm start	25s
	Hot start	1s
Data and Update Rate	Support Rate	4800bps to 921600bps,Default 9600bps
	Data Level	TTL or RS-232,Default TTL level
	Data Protocol	NMEA-0183 or UBX, Default NMEA-0183
	Single GNSS	1Hz-18Hz
	Concurrent GNSS	1Hz-10Hz,Default 1Hz
Operational Limits	Altitude	50,000m Max
	Velocity	515m/s Max
	Acceleration	Less than 4g
Power consumption	VCC	DC Voltage 3.0V-5.5V,Typical: 5.0V
	Current	Capture 50mA@5.0V
Mechanical Specifications	Dimension	22mm*20mm*6mm
	Weight	5.3g
	Connector	1.00mm spacing between the 4pins patch seat
Environment	Operating temp	-40 °C ~ +85°C
	Storage Temp	-40°C ~ +105°C
LED	built-in LED	TX LED:blue.The data output, TX LED flashing
		PPS LED:red.PPS LED not bright when GPS not fixed,flashing when fixed

BN-220 GPS Module + Antenna Datasheet

Pin Description:

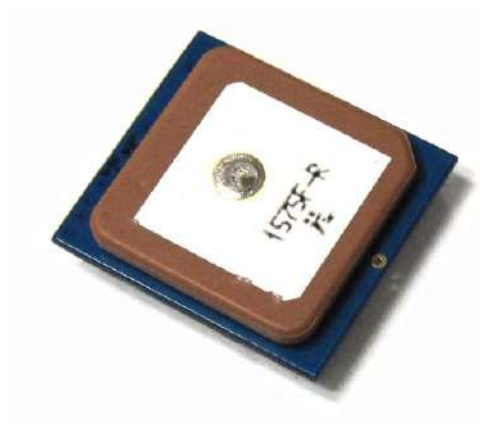


PIN	PIN Name	I/O	Description
1	GND	G	Ground
2	TX	O	Serial Data Output.
3	RX	I	Serial Data Input.
4	VCC	I	DC 3.0V - 5.5V supply input,Typical: 5.0V

LED:

- 1.TX LED:blue.The data output, TX LED flashing
- 2.PPS LED:red.PPS LED not bright when GPS not fixed, flashing when fixed.

Rear view:



BN-220 GPS Module + Antenna Datasheet

Message Structure:

\$xxGGA,time,lat,NS,long,EW,quality,numSV,HDOP,alt,M,sep,M,diffAge,diffStation*cs<CR><LF>

Example:

\$GPGGA,092725.00,4717.11399,N,00833.91590,E,1,08,1.01,499.6,M,48.0,M,,*5B

Field No	Name	Unit	Format	Example	Description
0	xxGGA	-	string	\$GPGGA	GGA Message ID (xx = current Talker ID)
1	time	-	hhmmss.ss	092725.00	UTC time
2	lat	-	ddmm.mmmmm	4717.11399	Latitude (degrees & minutes)
3	NS	-	character	N	North/South indicator
4	long	-	dddmm.mmmmm	00833.91590	Longitude (degrees & minutes)
5	EW	-	character	E	East/West indicator
6	quality	-	digit	1	0:No Fix / Invalid 1:Standard GPS (2D/3D) 2:Differential GPS 6:Estimated (DR) Fix
7	numSV	-	numeric	08	Number of satellites used
8	HDOP	-	numeric	1.01	Horizontal Dilution of Precision
9	alt	m	numeric	499.6	Altitude above mean sea level
10	uAlt	-	character	M	Altitude units: meters (fixed field)
11	sep	m	numeric	48.0	Geoid separation: difference between geoid and mean sea level
12	uSep	-	character	M	Separation units: meters (fixed field)
13	diffAge	s	numeric	-	Age of differential corrections (blank when DGPS is not used)
14	diffStation	-	numeric	-	ID of station providing differential corrections (blank when DGPS is not used)
15	cs	-	hexadecimal	*5B	Checksum
16	<CR><LF>	-	character	-	Carriage return and line feed

Message Structure:

\$xxGLL,lat,NS,long,EW,time,status,posMode*cs<CR><LF>

Example:

\$GPGLL,4717.11364,N,00833.91565,E,092321.00,A,A*6

Field No	Name	Unit	Format	Example	Description
0	xxGLL	-	string	\$GPGLL	GLL Message ID (xx = current Talker ID)
1	lat	-	ddmm.mmmmm	4717.11364	Latitude (degrees & minutes)
2	NS	-	character	N	North/South indicator
3	long	-	dddmm.mmmmm	00833.91565	Longitude (degrees & minutes)

BN-220 GPS Module + Antenna Datasheet

4	EW	-	character	E	East/West indicator
5	time	-	hhmmss.ss	092321.00	UTC time
6	status	-	character	A	V = Data invalid or receiver warning, A = Data valid
7	posMode	-	character	A	Positioning mode
8	cs	-	hexadecimal	*60	Checksum
9	<CR><LF>	-	character	-	Carriage return and line feed

Message Structure:

\$xxGSA,opMode,navMode{,sv},PDOP,HDOP,VDOP,systemId*cs<CR><LF>

Example:

\$GPGSA,A,3,23,29,07,08,09,18,26,28,,,,,1.94,1.18,1.54,1*0D

Field No	Name	Unit	Format	Example	Description
0	xxGSA	-	string	\$GPGSA	GSA Message ID (xx = current Talker ID)
1	opMode	-	character	A	Operation mode M:Manually set to operate in 2D or 3D mode A:Automatically switching between 2D or 3D mode
2	navMode	-	digit	3	Navigation mode 1:Fix not available 2:2D Fix 3:3D Fix
Start of repeated block (12 times)					
3 + 1*N	sv	-	numeric	29	Satellite number
End of repeated block					
15	PDOP	-	numeric	1.94	Position dilution of precision
16	HDOP	-	numeric	1.18	Horizontal dilution of precision
17	VDOP	-	numeric	1.54	Vertical dilution of precision
18	systemId	-	numeric	1	NMEA defined GNSS System ID NMEA v4.1 and above only
19	cs	-	hexadecimal	*0D	Checksum
20	<CR><LF>	-	character	-	Carriage return and line feed

Message Structure:

\$xxGSV,numMsg,msgNum,numSV,{,sv,elv,az,cno},signalId*cs<CR><LF>

Example:

\$GPGSV,3,1,10,23,38,230,44,29,71,156,47,07,29,116,41,08,09,081,36,0*7F

\$GPGSV,3,2,10,10,07,189,,05,05,220,,09,34,274,42,18,25,309,44,0*72

\$GPGSV,3,3,10,26,82,187,47,28,43,056,46,0*7

BN-220 GPS Module + Antenna Datasheet

Field No	Name	Unit	Format	Example	Description
0	xxGSV	-	string	\$GPGSV	GSV Message ID (xx = GSV Talker ID)
1	numMsg	-	digit	3	Number of messages, total number of GSV messages being output
2	msgNum	-	digit	1	Number of this message
3	numSV	-	numeric	10	Number of satellites in view
Start of repeated block (1..4 times)					
4 + 4*N	SV	-	numeric	23	Satellite ID
5 + 4*N	elv	deg	numeric	38	Elevation (range 0-90)
6 + 4*N	az	deg	numeric	230	Azimuth, (range 0-359)
7 + 4*N	cno	dBH	numeric	44	Signal strength (C/N0, range 0-99), blank when not tracking
End of repeated block					
5.. 16	signalId	-	numeric	0	NMEA defined GNSS Signal ID (0 = All signals) NMEA v4.1 and above only
6.. 16	cs	-	hexadecimal	*7F	Checksum
7.. 16	<CR><LF>	-	character	-	Carriage return and line feed

Message Structure:

\$xxRMC,time,status,lat,NS,long,EW,spd,cog,date,mv,mvEW,posMode,navStatus*cs<CR><LF>

Example:

\$GPRMC,083559.00,A,4717.11437,N,00833.91522,E,0.004,77.52,091202,,,A,V*57

Field No	Name	Unit	Format	Example	Description
0	xxRMC	-	string	\$GPRMC	RMC Message ID (xx = current Talker ID)
1	time	-	hhmmss.ss	083559.00	UTC time, see note on UTC representation
2	status	-	character	A	Status V:Navigation receiver warning A :Data valid, see position fix flags description
3	lat	-	ddmm.mmmmm	4717.11437	Latitude (degrees & minutes), see format description
4	NS	-	character	N	North/South indicator
5	long	-	dddmm.mmmmm	00833.91522	Longitude (degrees & minutes), see format description
6	EW	-	character	E	East/West indicator

BN-220 GPS Module + Antenna Datasheet

7	spd	Knots	numeric	0.004	Speed over ground
8	cog	degrees	numeric	77.52	Course over ground
9	date	-	ddmmyy	091202	Date in day, month, year format, see note on UTC representation
10	mv	degrees	numeric	-	Magnetic variation value (blank - not supported)
11	mvEW	-	character	-	Magnetic variation E/W indicator (blank - not supported)
12	posMode	-	character	-	Mode Indicator, see position fix flags
13	navStatus	-	character	V	Navigational status indicator (V = Equipment is not providing navigational status information)
14	cs	-	hexadecimal	*57	Checksum
15	<CR><LF>	-	character	-	Carriage return and line feed

Message Structure:

\$xxVTG,cogt,T,cogm,M,knots,N,kph,K,posMode*cs<CR><LF>

Example:

\$GPVTG,77.52,T,,M,0.004,N,0.008,K,A*06

Field No	Name	Unit	Format	Example	Description
0	xxVTG	-	string	\$GPVTG	VTG Message ID (xx = current Talker ID)
1	cogt	degrees	numeric	77.52	Course over ground (true)
2	T	-	character	T	Fixed field: true
3	cogm	degrees	numeric	-	Course over ground (magnetic), not output
4	M	-	character	M	Fixed field: magnetic
5	knots	knots	numeric	0.004	Speed over ground
6	N	-	character	N	Fixed field: knots
7	kph	km/	numeric	0.008	Speed over ground
8	K	-	character	K	Fixed field: kilometers per hour
9	posMode	-	character	A	Mode Indicator, see position fix flags description
10	cs	-	hexadecimal	*06	Checksum
11	<CR><LF>	-	character	-	Carriage return and line feed



InvenSense Inc.

1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A.

Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104

Website: www.invensense.com

Document Number: PS-MPU-6000A-00

Revision: 3.4

Release Date: 08/19/2013

MPU-6000 and MPU-6050

Product Specification

Revision 3.4



3 Product Overview

3.1 MPU-60X0 Overview

MotionInterface™ is becoming a “must-have” function being adopted by smartphone and tablet manufacturers due to the enormous value it adds to the end user experience. In smartphones, it finds use in applications such as gesture commands for applications and phone control, enhanced gaming, augmented reality, panoramic photo capture and viewing, and pedestrian and vehicle navigation. With its ability to precisely and accurately track user motions, MotionTracking technology can convert handsets and tablets into powerful 3D intelligent devices that can be used in applications ranging from health and fitness monitoring to location-based services. Key requirements for MotionInterface enabled devices are small package size, low power consumption, high accuracy and repeatability, high shock tolerance, and application specific performance programmability – all at a low consumer price point.

The MPU-60X0 is the world's first integrated 6-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. With its dedicated I²C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion™ output. The MPU-60X0 MotionTracking device, with its 6-axis integration, on-board MotionFusion™, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices, guaranteeing optimal motion performance for consumers. The MPU-60X0 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I²C port. The MPU-60X0 is footprint compatible with the MPU-30X0 family.

The MPU-60X0 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ± 250 , ± 500 , ± 1000 , and $\pm 2000^\circ/\text{sec}$ (dps) and a user-programmable accelerometer full-scale range of $\pm 2g$, $\pm 4g$, $\pm 8g$, and $\pm 16g$.

An on-chip 1024 Byte FIFO buffer helps lower system power consumption by allowing the system processor to read the sensor data in bursts and then enter a low-power mode as the MPU collects more data. With all the necessary on-chip processing and sensor components required to support many motion-based use cases, the MPU-60X0 uniquely enables low-power MotionInterface applications in portable applications with reduced processing requirements for the system processor. By providing an integrated MotionFusion output, the DMP in the MPU-60X0 offloads the intensive MotionProcessing computation requirements from the system processor, minimizing the need for frequent polling of the motion sensor output.

Communication with all registers of the device is performed using either I²C at 400kHz or SPI at 1MHz (MPU-6000 only). For applications requiring faster communications, the sensor and interrupt registers may be read using SPI at 20MHz (MPU-6000 only). Additional features include an embedded temperature sensor and an on-chip oscillator with $\pm 1\%$ variation over the operating temperature range.

By leveraging its patented and volume-proven Nasiri-Fabrication platform, which integrates MEMS wafers with companion CMOS electronics through wafer-level bonding, InvenSense has driven the MPU-60X0 package size down to a revolutionary footprint of 4x4x0.9mm (QFN), while providing the highest performance, lowest noise, and the lowest cost semiconductor packaging required for handheld consumer electronic devices. The part features a robust 10,000g shock tolerance, and has programmable low-pass filters for the gyroscopes, accelerometers, and the on-chip temperature sensor.

For power supply flexibility, the MPU-60X0 operates from VDD power supply voltage range of 2.375V-3.46V. Additionally, the MPU-6050 provides a VLOGIC reference pin (in addition to its analog supply pin: VDD), which sets the logic levels of its I²C interface. The VLOGIC voltage may be $1.8V \pm 5\%$ or VDD.

The MPU-6000 and MPU-6050 are identical, except that the MPU-6050 supports the I²C serial interface only, and has a separate VLOGIC reference pin. The MPU-6000 supports both I²C and SPI interfaces and has a single supply pin, VDD, which is both the device's logic reference supply and the analog supply for the part. The table below outlines these differences:



MPU-6000/MPU-6050 Product Specification

Document Number: PS-MPU-6000A-00
Revision: 3.4
Release Date: 08/19/2013

Primary Differences between MPU-6000 and MPU-6050

Part / Item	MPU-6000	MPU-6050
VDD	2.375V-3.46V	2.375V-3.46V
VLOGIC	n/a	1.71V to VDD
Serial Interfaces Supported	I ² C, SPI	I ² C
Pin 8	/CS	VLOGIC
Pin 9	AD0/SDO	AD0
Pin 23	SCL/SCLK	SCL
Pin 24	SDA/SDI	SDA



5 Features

5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ± 250 , ± 500 , ± 1000 , and $\pm 2000^\circ/\text{sec}$
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5 μ A
- Factory calibrated sensitivity scale factor
- User self-test

5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500 μ A
- Low power accelerometer mode current: 10 μ A at 1.25Hz, 20 μ A at 5Hz, 60 μ A at 20Hz, 110 μ A at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

5.3 Additional Features

The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I²C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I²C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 g shock tolerant
- 400kHz Fast Mode I²C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)



6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		°/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		°/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		°/s	
Linear Acceleration Sensitivity	Static		0.1		°/s/g	
SELF-TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	1
GYROSCOPE NOISE PERFORMANCE						
Total RMS Noise	FS_SEL=0 DLPCFG=2 (100Hz)		0.05		°/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to 10Hz		0.033		°/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		°/s/√Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		256	Hz	
OUTPUT DATA RATE						
	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME						
ZRO Settling (from power-on)	DLPCFG=0 to ±1°/s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*



MPU-6000/MPU-6050 Product Specification

Document Number: PS-MPU-6000A-00
Revision: 3.4
Release Date: 08/19/2013

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		g	
	AFS_SEL=1		±4		g	
	AFS_SEL=2		±8		g	
	AFS_SEL=3		±16		g	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/g	
	AFS_SEL=1		8,192		LSB/g	
	AFS_SEL=2		4,096		LSB/g	
	AFS_SEL=3		2,048		LSB/g	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		mg	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		μg/√Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*



MPU-6000/MPU-6050 Product Specification

Document Number: PS-MPU-6000A-00
Revision: 3.4
Release Date: 08/19/2013

6.3 Electrical and Other Common Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
TEMPERATURE SENSOR						
Range			-40 to +85		°C	
Sensitivity	Untrimmed		340		LSB/°C	
Temperature Offset	35°C		-521		LSB	
Linearity	Best fit straight line (-40°C to +85°C)		±1		°C	
VDD POWER SUPPLY						
Operating Voltages		2.375		3.46	V	
Normal Operating Current	Gyroscope + Accelerometer + DMP		3.9		mA	
	Gyroscope + Accelerometer (DMP disabled)		3.8		mA	
	Gyroscope + DMP (Accelerometer disabled)		3.7		mA	
	Gyroscope only (DMP & Accelerometer disabled)		3.6		mA	
	Accelerometer only (DMP & Gyroscope disabled)		500		µA	
Accelerometer Low Power Mode Current	1.25 Hz update rate		10		µA	
	5 Hz update rate		20		µA	
	20 Hz update rate		70		µA	
	40 Hz update rate		140		µA	
Full-Chip Idle Mode Supply Current			5		µA	
Power Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value			100	ms	
VLOGIC REFERENCE VOLTAGE						
Voltage Range	MPU-6050 only					
Power Supply Ramp Rate	VLOGIC must be ≤VDD at all times	1.71		VDD	V	
Normal Operating Current	Monotonic ramp. Ramp rate is 10% to 90% of the final value			3	ms	
			100		µA	
TEMPERATURE RANGE						
Specified Temperature Range	Performance parameters are not applicable beyond Specified Temperature Range	-40		+85	°C	

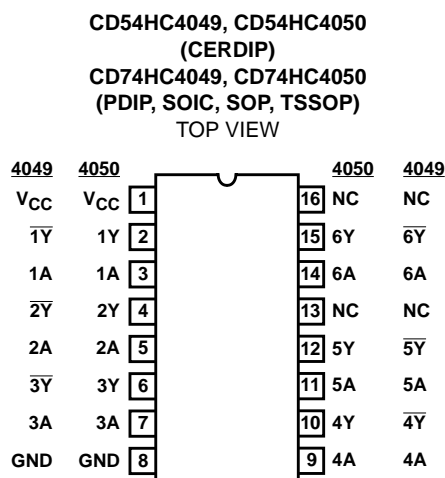
February 1998 - Revised February 2005

High-Speed CMOS Logic Hex Buffers, Inverting and Non-Inverting

Features

- Typical Propagation Delay: 6ns at $V_{CC} = 5V$, $C_L = 15pF$, $T_A = 25^\circ C$
- High-to-Low Voltage Level Converter for up to $V_I = 16V$
- Fanout (Over Temperature Range)
 - Standard Outputs 10 LSTTL Loads
 - Bus Driver Outputs 15 LSTTL Loads
- Wide Operating Temperature Range . . . $-55^\circ C$ to $125^\circ C$
- Balanced Propagation Delay and Transition Times
- Significant Power Reduction Compared to LSTTL Logic ICs
- HC Types
 - 2V to 6V Operation
 - High Noise Immunity: $N_{IL} = 30\%$, $N_{IH} = 30\%$ of V_{CC} at $V_{CC} = 5V$

Pinout



Description

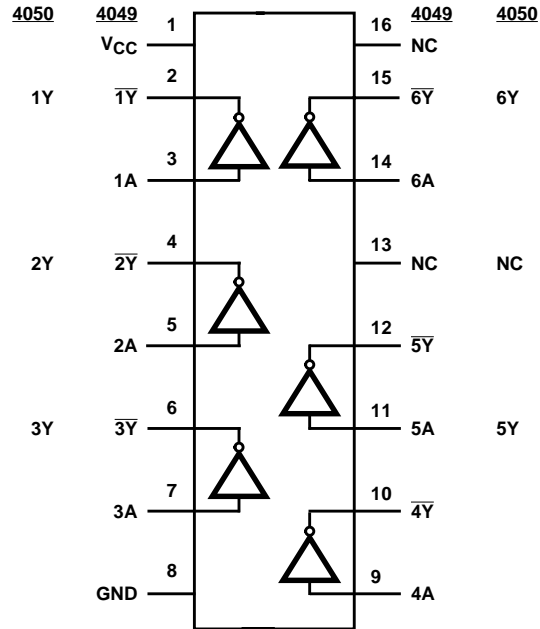
The 'HC4049 and 'HC4050 are fabricated with high-speed silicon gate technology. They have a modified input protection structure that enables these parts to be used as logic level translators which convert high-level logic to a low-level logic while operating off the low-level logic supply. For example, 15-V input pulse levels can be down-converted to 0-V to 5-V logic levels. The modified input protection structure protects the input from negative electrostatic discharge. These parts also can be used as simple buffers or inverters without level translation. The 'HC4049 and 'HC4050 are enhanced versions of equivalent CMOS types.

Ordering Information

PART NUMBER	TEMP. RANGE (°C)	PACKAGE
CD54HC4049F3A	-55 to 125	16 Ld CERDIP
CD54HC4050F3A	-55 to 125	16 Ld CERDIP
CD74HC4049E	-55 to 125	16 Ld PDIP
CD74HC4049M	-55 to 125	16 Ld SOIC
CD74HCT4050MT	-55 to 125	16 Ld SOIC
CD74HC4049M96	-55 to 125	16 Ld SOIC
CD74HC4049NSR	-55 to 125	16 Ld SOP
CD74HC4049PW	-55 to 125	16 Ld TSSOP
CD74HC4049PWR	-55 to 125	16 Ld TSSOP
CD74HC4049PWT	-55 to 125	16 Ld TSSOP
CD74HC4050E	-55 to 125	16 Ld PDIP
CD74HC4050M	-55 to 125	16 Ld SOIC
CD74HC4050MT	-55 to 125	16 Ld SOIC
CD74HC4050M96	-55 to 125	16 Ld SOIC
CD74HC4050NSR	-55 to 125	16 Ld SOP
CD74HC4050PW	-55 to 125	16 Ld TSSOP
CD74HC4050PWR	-55 to 125	16 Ld TSSOP
CD74HC4050PWT	-55 to 125	16 Ld TSSOP

NOTE: When ordering, use the entire part number. The suffixes 96 and R denote tape and reel. The suffix T denotes a small-quantity reel of 250.

Functional Diagram

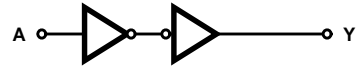


Logic Diagrams

HC4049



HC4050



CD54HC4049, CD74HC4049, CD54HC4050, CD74HC4050

Absolute Maximum Ratings

DC Supply Voltage, V_{CC}	−0.5V to 7V
Input Voltage Range	−0.5V to 16V
DC Input Diode Current, I_{IK}	
For $V_I < -0.5V$	−20mA
DC Output Diode Current, I_{OK}	
For $V_O < -0.5V$ or $V_O > V_{CC} + 0.5V$	±20mA
DC Output Source or Sink Current per Output Pin, I_O	
For $V_O > -0.5V$ or $V_O < V_{CC} + 0.5V$	±25mA
DC V_{CC} or Ground Current, I_{CC} or I_{GND}	±50mA

Operating Conditions

Temperature Range (T_A)	−55°C to 125°C
Supply Voltage Range, V_{CC}	
HC Types	.2V to 6V
HCT Types	4.5V to 5.5V
DC Input Voltage, V_I	.0V to 15V
DC Output Voltage, V_O	0V to V_{CC}
Input Rise and Fall Time	
2V	1000ns (Max)
4.5V	500ns (Max)
6V	400ns (Max)

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

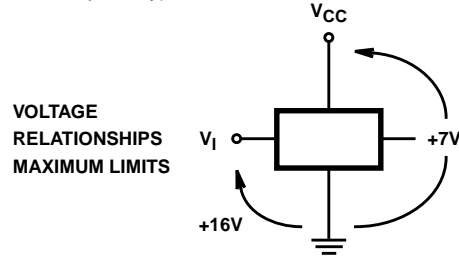
NOTE:

- The package thermal impedance is calculated in accordance with JESD 51-7.

Thermal Information

Package Thermal Impedance, θ_{JA} (see Note 1):

E (PDIP) Package	67°C/W
M (SOIC) Package	73°C/W
NS (SOP) Package	64°C/W
PW (TSSOP) Package	108°C/W
Maximum Junction Temperature (Hermetic Package or Die)	175°C
Maximum Junction Temperature (Plastic Package)	150°C
Maximum Storage Temperature Range	−65°C to 150°C
Maximum Lead Temperature (Soldering 10s)	300°C
(SOIC - Lead Tips Only)	



DC Electrical Specifications

PARAMETER	SYMBOL	TEST CONDITIONS		V _{CC} (V)	25°C			−40°C TO 85°C		−55°C TO 125°C		UNITS
		V _I (V)	I _O (mA)		MIN	TYP	MAX	MIN	MAX	MIN	MAX	
HC TYPES												
High Level Input Voltage	V _{IH}	-	-	2	1.5	-	-	1.5	-	1.5	-	V
				4.5	3.15	-	-	3.15	-	3.15	-	V
				6	4.2	-	-	4.2	-	4.2	-	V
Low Level Input Voltage	V _{IL}	-	-	2	-	-	0.5	-	0.5	-	0.5	V
				4.5	-	-	1.35	-	1.35	-	1.35	V
				6	-	-	1.8	-	1.8	-	1.8	V
High Level Output Voltage CMOS Loads	V _{OH}	V _{IH} or V _{IL}	−0.02	2	1.9	-	-	1.9	-	1.9	-	V
			−0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
			−0.02	6	5.9	-	-	5.9	-	5.9	-	V
High Level Output Voltage TTL Loads			−4	4.5	3.98	-	-	3.84	-	3.7	-	V
−5.2			6	5.48	-	-	5.34	-	5.2	-	V	
Low Level Output Voltage CMOS Loads	V _{OL}	V _{IH} or V _{IL}	0.02	2	-	-	0.1	-	0.1	-	0.1	V
			0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
			0.02	6	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			4	4.5	-	-	0.26	-	0.33	-	0.4	V
			5.2	6	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I _I	V _{CC} or GND	-	6	-	-	±0.1	-	±1	-	±1	μA
		15	-	6	-	-	±0.5	-	±5	-	±5	

CD54HC4049, CD74HC4049, CD54HC4050, CD74HC4050

DC Electrical Specifications (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS		V _{CC} (V)	25°C			-40°C TO 85°C		-55°C TO 125°C		UNITS
		V _I (V)	I _O (mA)		MIN	TYP	MAX	MIN	MAX	MIN	MAX	
Quiescent Device Current	I _{CC}	V _{CC} or GND	0	6	-	-	2	-	20	-	40	μA

Switching Specifications Input t_r, t_f = 6ns

PARAMETER	SYMBOL	TEST CONDITIONS	V _{CC} (V)	25°C			-40°C TO 85°C		-55°C TO 125°C		UNITS
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
HC TYPES											
Propagation Delay, nA to nȳ HC4049 nA to nY HC4050	t _{PLH} , t _{PHL}	C _L = 50pF	2	-	-	85	-	105	-	130	ns
			4.5	-	-	17	-	21	-	26	ns
			6	-	-	14	-	18	-	22	ns
		C _L = 15pF	5	-	6	-	-	-	-	-	ns
Transition Times (Figure 1)	t _{TLH} , t _{THL}	C _L = 50pF	2	-	-	75	-	95	-	110	ns
			4.5	-	-	15	-	19	-	22	ns
			6	-	-	13	-	16	-	19	ns
Input Capacitance	C _I	-	-	-	-	10	-	10	-	10	pF
Power Dissipation Capacitance (Notes 2, 3)	C _{PD}	-	5	-	35	-	-	-	-	-	pF

NOTES:

- C_{PD} is used to determine the dynamic power consumption, per gate.
- P_D = V_{CC}² f_i (C_{PD} + C_L) where f_i = Input Frequency, C_L = Output Load Capacitance, V_{CC} = Supply Voltage.

Test Circuit and Waveform

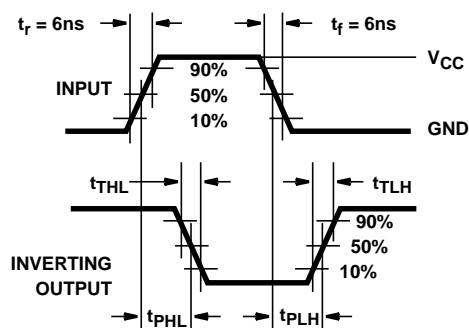


FIGURE 1. HC AND HCU TRANSITION TIMES AND PROPAGATION DELAY TIMES, COMBINATION LOGIC

TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8

DESCRIPTION

The TP4056 is a complete constant-current/constant-voltage linear charger for single cell lithium-ion batteries. Its SOP package and low external component count make the TP4056 ideally suited for portable applications. Furthermore, the TP4056 can work within USB and wall adapter.

No blocking diode is required due to the internal PMOSFET architecture and have prevent to negative Charge Current Circuit. Thermal feedback regulates the charge current to limit the die temperature during high power operation or high ambient temperature. The charge voltage is fixed at 4.2V, and the charge current can be programmed externally with a single resistor. The TP4056 automatically terminates the charge cycle when the charge current drops to 1/10th the programmed value after the final float voltage is reached.

TP4056 Other features include current monitor, under voltage lockout, automatic recharge and two status pin to indicate charge termination and the presence of an input voltage.

FEATURES

- Programmable Charge Current Up to 1000mA
- No MOSFET, Sense Resistor or Blocking Diode Required
- Complete Linear Charger in SOP-8 Package for Single Cell Lithium-Ion Batteries
- Constant-Current/Constant-Voltage
- Charges Single Cell Li-Ion Batteries Directly from USB Port
- Preset 4.2V Charge Voltage with 1.5% Accuracy
- Automatic Recharge
- two Charge Status Output Pins
- C/10 Charge Termination
- 2.9V Trickle Charge Threshold (TP4056)
- Soft-Start Limits Inrush Current
- Available Radiator in 8-Lead SOP Package, the Radiator need connect GND or impending

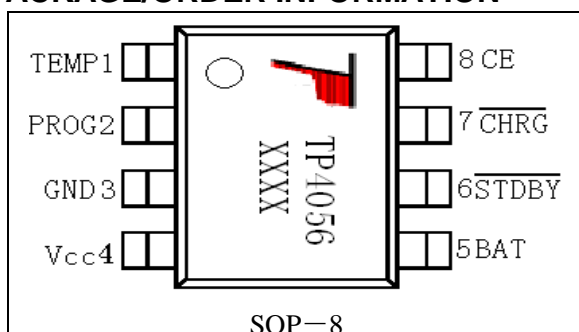

ABSOLUTE MAXIMUM RATINGS

- Input Supply Voltage(V_{CC}): -0.3V~8V
- TEMP: -0.3V~10V
- CE: -0.3V~10V
- BAT Short-Circuit Duration: Continuous
- BAT Pin Current: 1200mA
- PROG Pin Current: 1200uA
- Maximum Junction Temperature: 145°C
- Operating Ambient Temperature Range: -40°C~85°C
- Lead Temp.(Soldering, 10sec): 260°C

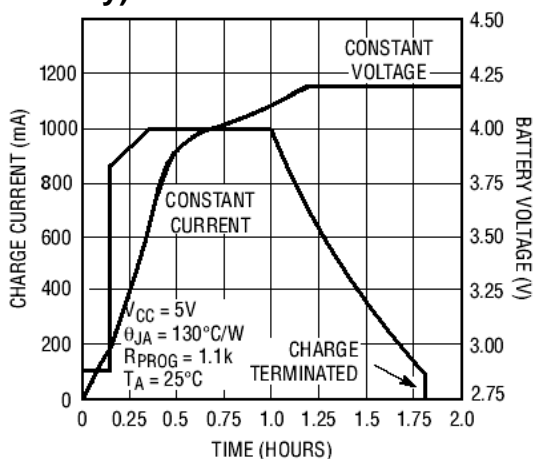
APPLICATIONS

- Cellular Telephones, PDAs, GPS
- Charging Docks and Cradles
- Digital Still Cameras, Portable Devices
- USB Bus-Powered Chargers,Chargers

PACKAGE/ORDER INFORMATION

	SOP-8
<p>photo</p> 	<p>ORDER PART NUMBER TP4056-42-SOP8-PP</p> <p>PART MARKING TP4056</p>

Complete Charge Cycle (1000mAh Battery)



TEMP(Pin 1) :Temperature Sense Input Connecting TEMP pin to NTC thermistor's output in Lithium ion battery pack. If TEMP pin's voltage is below 45% or above 80% of supply voltage V_{IN} for more than 0.15S, this means that battery's temperature is too high or too low, charging is suspended. The temperature sense function can be disabled by grounding the TEMP pin.

PROG(Pin 2): Constant Charge Current Setting and Charge Current Monitor Pin charge current is set by connecting a resistor R_{ISET} from this pin to GND. When in precharge mode, the ISET pin's voltage is regulated to 0.2V. When in constant charge current mode, the ISET pin's voltage is regulated to 2V. In all modes during charging, the voltage on ISET pin can be used to measure the charge current as follows:

GND(Pin3): Ground Terminal

$$I_{BAT} = \frac{V_{PROG}}{R_{PROG}} \times 1200 \quad (V_{PROG}=1V)$$

Vcc(Pin 4): Positive Input Supply Voltage V_{IN} is the power supply to the internal circuit. When V_{IN} drops to within 30mv of the BAT pin voltage, TP4056 enters low power sleep mode, dropping BAT pin's current to less than 2uA.

BAT(Pin5): Battery Connection Pin. Connect the positive terminal of the battery to BAT pin. BAT pin draws less than 2uA current in chip disable mode or in sleep mode. BAT pin provides charge current to the battery and provides regulation voltage of 4.2V.

STDBY(Pin6): Open Drain Charge Status Output When the battery Charge Termination, the \overline{STDBY} pin is pulled low by an internal switch, otherwise \overline{STDBY} pin is in high impedance state.

CHRG (Pin7): Open Drain Charge Status Output When the battery is being charged, the \overline{CHRG} pin is pulled low by an internal switch, otherwise \overline{CHRG} pin is in high impedance state.

CE(Pin8): Chip Enable Input. A high input will put the device in the normal operating mode.

Pulling the CE pin to low level will put the YP4056 into disable mode. The CE pin can be driven by TTL or CMOS logic level.

ELECTRICAL CHARACTERISTICS

The ● denotes specifications which apply over the full operating temperature range, otherwise specifications are at $T_A=25^{\circ}C$, $V_{CC}=5V$, unless otherwise noted.

SYMBOL	PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
V_{CC}	Input Supply Voltage		●	4.0	5	8.0	V
I_{CC}	Input Supply Current	Charge Mode, $R_{PROG} = 1.2k$	●		150	500	μA
		StandbyMode(Charge Terminated)	●		55	100	μA
		Shutdown Mode (R_{PROG} Not Connected, $V_{CC} < V_{BAT}$, or $V_{CC} < V_{UV}$)	●		55	100	μA
V_{FLOAL}	Regulated Output (Float) Voltage	$0^{\circ}C \leq T_A \leq 85^{\circ}C$, $I_{BAT}=40mA$		4.137	4.2	4.263	V
I_{BAT}	BAT Pin Current Text condition: $V_{BAT}=4.0V$	$R_{PROG} = 2.4k$, Current Mode	●	450	500	550	mA
		$R_{PROG} = 1.2k$, Current Mode	●	950	1000	1050	mA
		Standby Mode, $V_{BAT} = 4.2V$	●	0	-2.5	-6	μA
I_{TRIKL}	Trickle Charge Current	$V_{BAT} < V_{TRIKL}$, $R_{PROG}=1.2K$	●	120	130	140	mA
V_{TRIKL}	Trickle Charge Threshold Voltage	$R_{PROG}=1.2K$, V_{BAT} Rising		2.8	2.9	3.0	V
V_{TRHYS}	Trickle Charge Hysteresis Voltage	$R_{PROG}=1.2K$		60	80	100	mV
T_{LIM}	Junction Temperature in Constant Temperature Mode				145		$^{\circ}C$

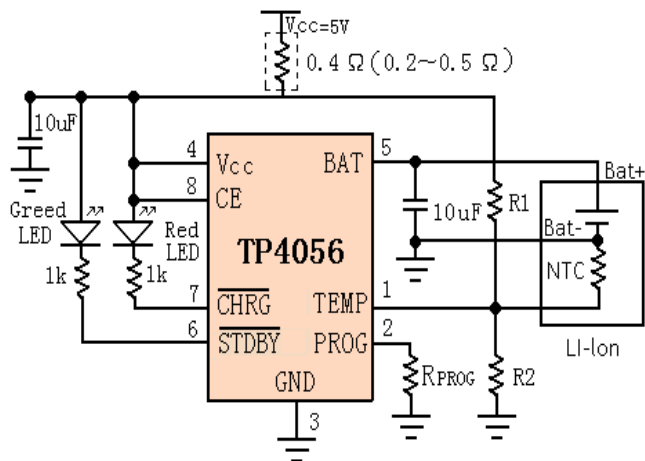
indicator light state

Charge state	Red LED $\overline{\text{CHRG}}$	Green LED $\overline{\text{STDBY}}$
charging	bright	extinguish
Charge Termination	extinguish	bright
Vin too low; Temperature of battery too low or too high; no battery	extinguish	extinguish
BAT PIN Connect 10u Capacitance; No battery	Green LED bright, Red LED Coruscate T=1-4 S	

Rprog Current Setting

R _{PROG} (k)	I _{BAT} (mA)
10	130
5	250
4	300
3	400
2	580
1.66	690
1.5	780
1.33	900
1.2	1000

TYPICAL APPLICATIONS



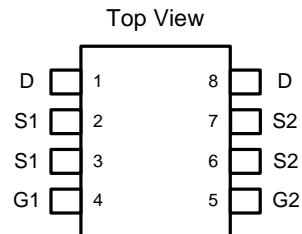
FS8205A

Dual N-Channel Enhancement Mode MOSFET

Features

- 20V/6A,
 $R_{DS(ON)} < 25m\Omega$ @ $V_{GS}=4.5V$
 $R_{DS(ON)} < 34m\Omega$ @ $V_{GS}=2.5V$
- Super High Dense Cell Design
- Reliable and Rugged
- Lead Free Available (RoHS Compliant)

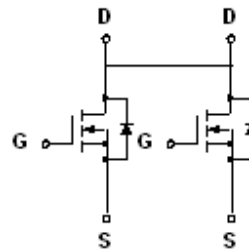
Pin Description



TSSOP-8

Applications

- Portable Equipment and Battery Powered Systems.



N Channel MOSFET

FS8205A

Absolute Maximum Ratings (T_A=25°C Unless Otherwise Noted)

Symbol	Parameter	Rating	Unit
V _{DSS}	Drain-Source Voltage	20	V
V _{GSS}	Gate-Source Voltage	±8	
I _D *	Continuous Drain Current	V _{GS} =4.5V 6	A
I _{DM} *	300µs Pulsed Drain Current		
I _S *	Diode Continuous Forward Current	1	A
T _J	Maximum Junction Temperature	150	°C
T _{STG}	Storage Temperature Range	-55 to 150	
P _D *	Maximum Power Dissipation	T _A =25°C 1.25	W
		T _A =100°C 0.5	
R _{θJA} *	Thermal Resistance-Junction to Ambient	100	°C/W

Notes :

*Surface Mounted on 1in² pad area, t ≤ 10sec.

Electrical Characteristics (T_A=25°C Unless Otherwise Noted)

Symbol	Parameter	Test Condition	8205A			Unit
			Min.	Typ.	Max.	
Static Characteristics						
BV _{DSS}	Drain-Source Breakdown Voltage	V _{GS} =0V, I _{DS} =250μA	20			V
I _{DSS}	Zero Gate Voltage Drain Current	V _{DS} =16V, V _{GS} =0V			1	μA
		T _J =85°C			30	
V _{GS(th)}	Gate Threshold Voltage	V _{DS} =V _{GS} , I _{DS} =250μA	0.5	0.7	1.5	V
I _{GSS}	Gate Leakage Current	V _{GS} =±8V, V _{DS} =0V			±100	nA
R _{DS(ON)} ^a	Drain-Source On-state Resistance	V _{GS} =4.5V, I _{DS} =6A		20	25	mΩ
		V _{GS} =2.5V, I _{DS} =5.2A		27	34	
Diode Characteristics						
V _{SD} ^a	Diode Forward Voltage	I _{SD} =1A, V _{GS} =0V		0.8	1.3	V
t _{rr}	Reverse Recovery Time	I _{DS} =6A, dI _{SD} /dt=100A/μs		14		ns
Q _{rr}	Reverse Recovery Charge			5		nC

FS8205A

Electrical Characteristics (Cont.) (T_A=25°C Unless Otherwise Noted)

Symbol	Parameter	Test Condition	8205A			Unit
			Min.	Typ.	Max.	
Dynamic Characteristics ^b						
R _G	Gate Resistance	V _{GS} =0V,V _{DS} =0V,F=1MHz		5.5		Ω
C _{iss}	Input Capacitance	V _{GS} =0V, V _{DS} =10V, Frequency=1.0MHz		595		pF
C _{oss}	Output Capacitance			140		
C _{rss}	Reverse Transfer Capacitance			125		
t _{d(ON)}	Turn-on Delay Time	V _{DD} =10V, R _L =10Ω, I _{DS} =1A, V _{GEN} =4.5V, R _G =6Ω		3.5	7	ns
T _r	Turn-on Rise Time			13.5	25	
t _{d(OFF)}	Turn-off Delay Time			32	58	
T _f	Turn-off Fall Time			6.6	13	
Gate Charge Characteristics ^b						
Q _g	Total Gate Charge	I _{DS} =6A, dI _{SD} /dt=100A/μs		21	29	nC
Q _{gs}	Gate-Source Charge			1.3		
Q _{gd}	Gate-Drain Charge			3.3		

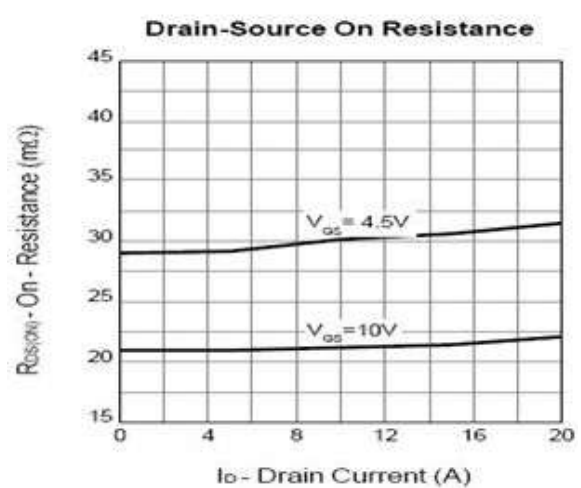
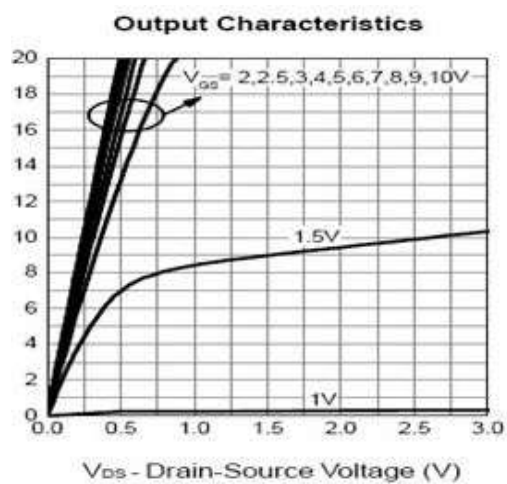
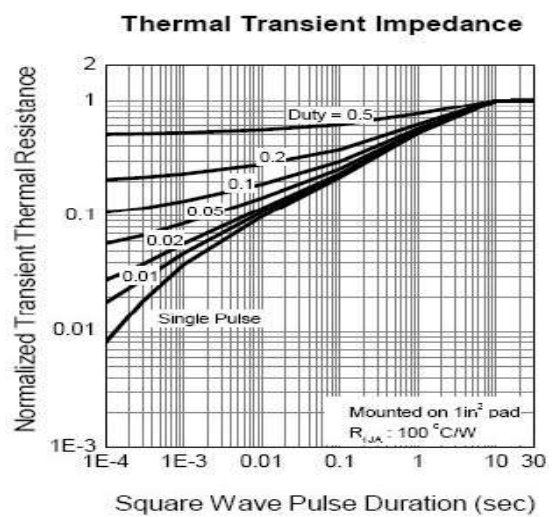
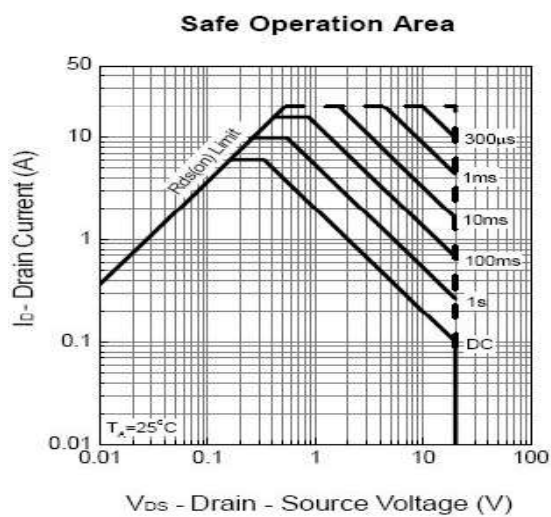
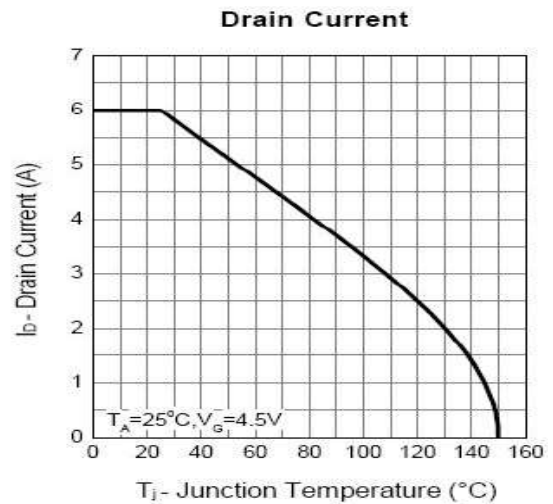
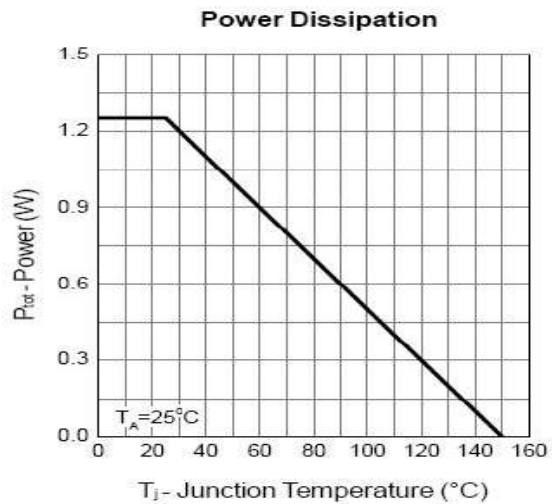
Notes :

a : Pulse test ; pulse width≤300μs, duty cycle≤2%.

b : Guaranteed by design, not subject to production testing.

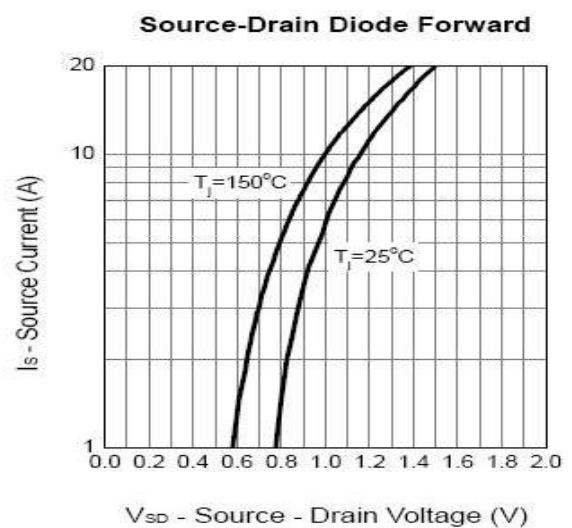
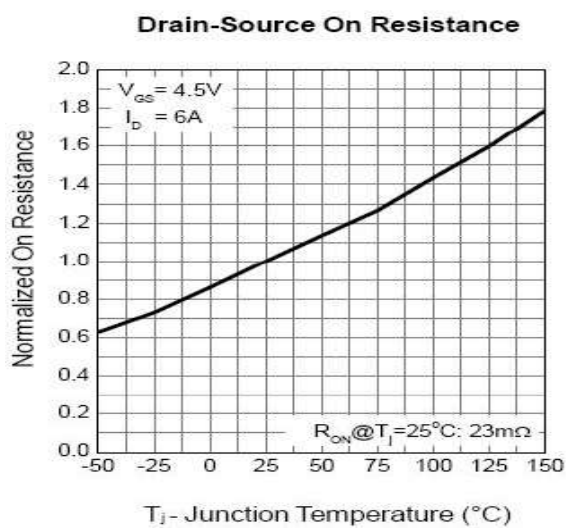
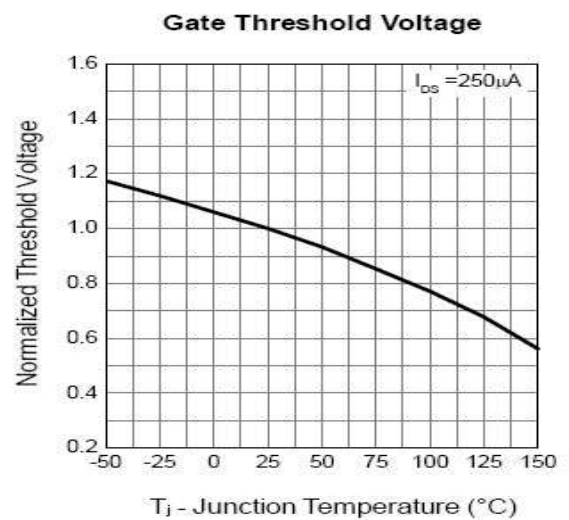
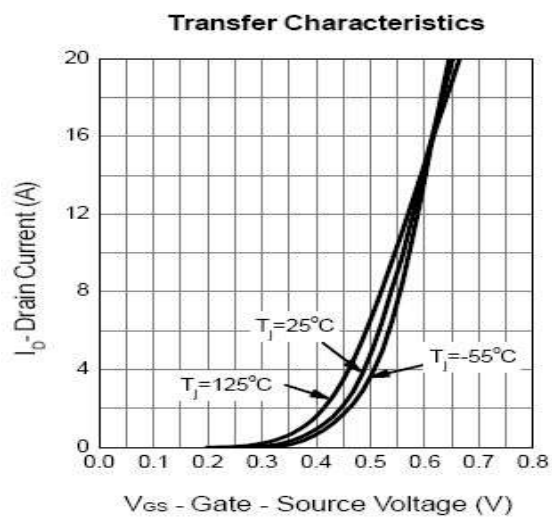
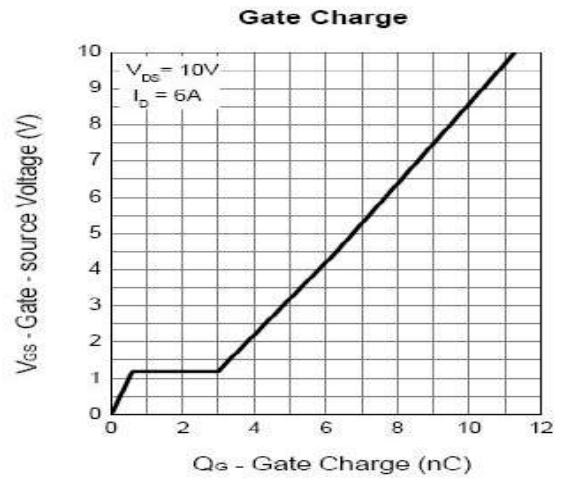
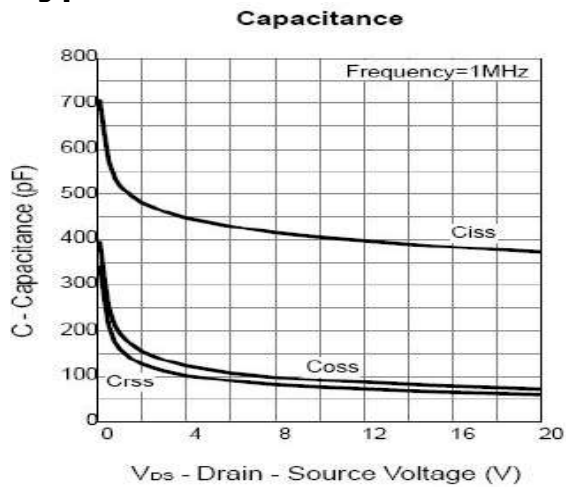
FS8205A

Typical Characteristics



FS8205A

Typical Characteristics





One Cell Lithium-ion/Polymer Battery Protection IC

General Description

The DW01-P battery protection IC is designed to protect lithium-ion/polymer battery from damage or degrading the lifetime due to overcharge, overdischarge, and/or overcurrent for one-cell lithium-ion/polymer battery powered systems, such as cellular phones.

The ultra-small package and less required external components make it ideal to integrate the DW01-P into the limited space of battery pack. The accurate $\pm 50\text{mV}$ overcharging detection voltage ensures safe and full utilization charging. The very low standby current drains little current from the cell while in storage.

Features

- Reduction in Board Size due to Miniature Package SOT-23-6.
- Ultra-Low Quiescent Current at $3\ \mu\text{A}$ ($V_{\text{CC}}=3.9\text{V}$).
- Ultra-Low Power-Down Current at $0.1\ \mu\text{A}$ ($V_{\text{CC}}=2.0\text{V}$).
- Precision Overcharge Protection Voltage $4.25\text{V} \pm 50\text{mV}$
- Load Detection Function during Overcharge Mode.
- Two Detection Levels for Overcurrent Protection.
- Delay times are generated by internal circuits. No external capacitors required.

Ordering Information

DW01-P

PACKAGE TYPE
SOT-23-6 (Pb-free)

TEMPERATURE RANGE
 $-40^{\circ}\text{C} \sim +85^{\circ}\text{C}$

OVERCHARGE PROTECTION
 $4.25\text{V} \pm 50\text{mV}$

Applications

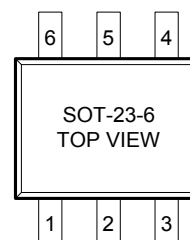
- Protection IC for One-Cell Lithium-Ion / Lithium-Polymer Battery Pack

Product Name List

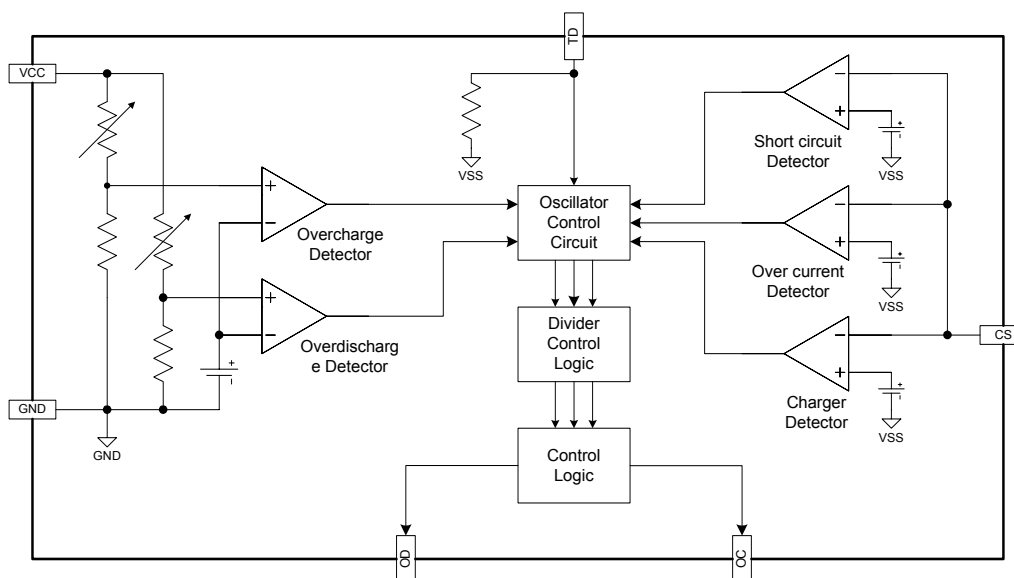
Model	Package	Overcharge detection voltage [V _{OCP}] (V)	Overcharge release voltage [V _{OCR}] (V)	Overdischarge detection voltage [V _{ODP}] (V)	Overdischarge release voltage [V _{ODR}] (V)	Overcurrent detection voltage [V _{OI1}] (mV)
	SOT-23-6					
DW01-P	DW01-P	4.250±0.050	4.050±0.050	2.40±0.100	3.0±0.100	150±30

Pin Configuration

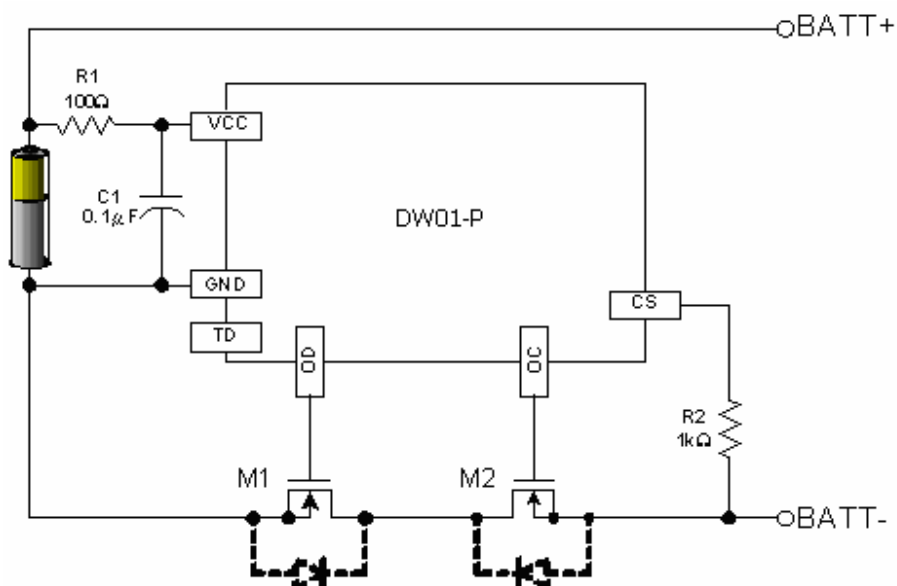
Pin No.	Symbol	Description
1	OD	MOSFET gate connection pin for discharge control
2	CS	Input pin for current sense, charger detect
3	OC	MOSFET gate connection pin for charge control
4	TD	Test pin for reduce delay time
5	VCC	Power supply, through a resistor (R1)
6	GND	Ground pin



Functional Block Diagram



Typical Application Circuit



Absolute Maximum Ratings

(GND=0V, Ta=25°C unless otherwise specified)

Item	Symbol	Rating	Unit
Input voltage between Vcc and GND *	Vcc	GND-0.3 to GND+10	V
OC output pin voltage	Voc	Vcc -24 to Vcc +0.3	V
OD output pin voltage	Vod	GND-0.3 to Vcc +0.3	V
CS input pin voltage	Vcs	Vcc -24 to Vcc +0.3	V
Operating Temperature Range	TOP	-40 to +85	°C
Storage Temperature Range	TST	-40 to +125	°C

Note: DW01-P contains a circuit that will protect it from static discharge; but please take special care that no excessive static electricity or voltage which exceeds the limit of the protection circuit will be applied to it.

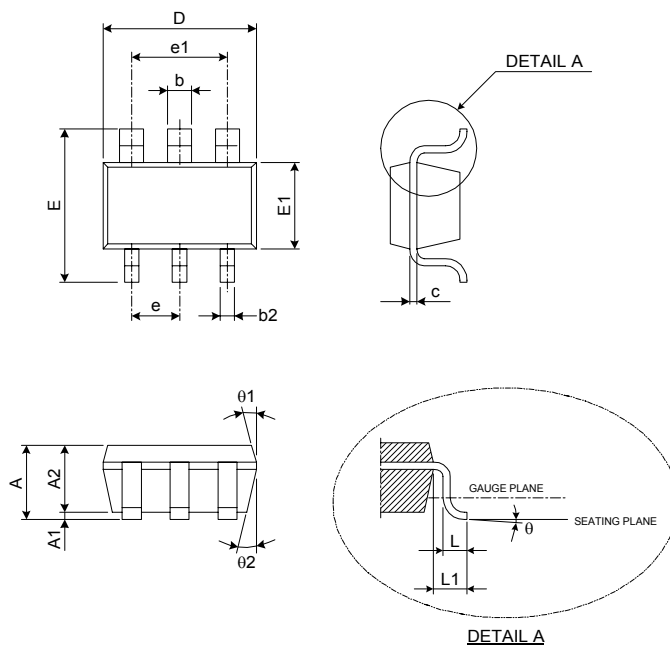
Electrical Characteristics

(Ta=25°C unless otherwise specified)

PARAMETER	TEST CONDITIONS	SYMBOL	Min	Typ	Max	UNIT
Supply Current	V _{CC} =3.9V	I _{CC}		3.0	6.0	μA
Power-Down Current	V _{CC} =2.0V	I _{PD}			0.1	μA
Overcharge Protection Voltage	DW01-P	V _{OCP}	4.20	4.25	4.30	V
Overcharge Release Voltage		V _{OCR}	4.00	4.05	4.10	V
Overdischarge Protection Voltage		V _{ODP}	2.30	2.40	2.50	V
Overdischarge Release Voltage		V _{ODR}	2.90	3.00	3.10	V
Overcurrent Protection Voltage		V _{OIP} (V _{OI1})	120	150	180	mV
Short Current Protection Voltage	V _{CC} =3.6V	V _{SIP} (V _{OI2})	1.00	1.35	1.70	V
Overcharge Delay Time		T _{OC}		80	200	ms
Overdischarge Delay Time	V _{CC} =3.6V to 2.0V	T _{OD}		40	100	ms
Overcurrent Delay Time (1)	V _{CC} =3.6V	T _{OI1}		10	20	ms
Overcurrent Delay Time (2)	V _{CC} =3.6V	T _{OI2}			500	μs
Charger Detection Threshold Voltage		V _{CHA}	-1.2	-0.7	-0.2	V
OD Pin Output "H" Voltage		V _{DH}	V _{CC} -0.1	V _{CC} -0.02		V
OD Pin Output "L" Voltage		V _{DL}		0.1	0.5	V
OC Pin Output "H" Voltage		V _{CH}	V _{CC} -0.1	V _{CC} -0.02		V
OC Pin Output "L" Voltage		V _{CL}		0.1	0.5	V

Package Outline

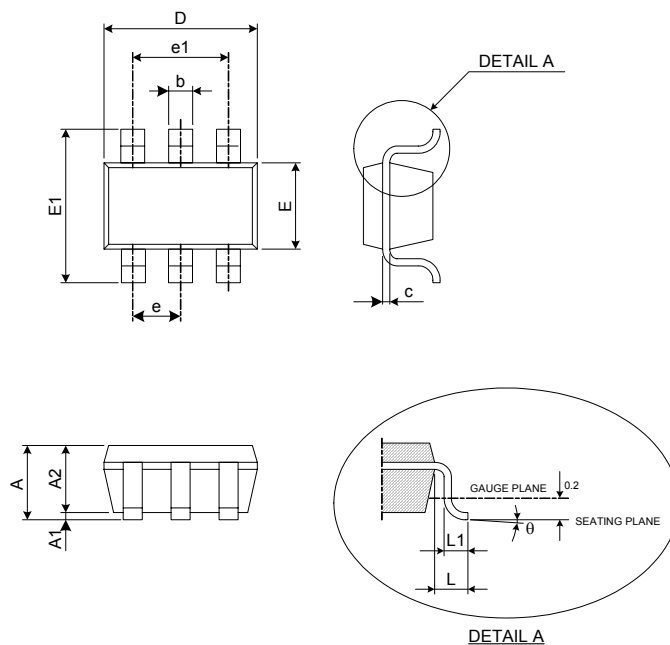
Dimension (Package A)



Unit : mm

SYMBOL	MIN.	TYP.	MAX.
A	1.05	-	1.35
A1	0.05	-	0.15
A2	1.00	1.10	1.20
b	0.40	-	0.55
b2	0.25	-	0.40
c	0.08	-	0.20
D	2.70	2.90	3.00
E	2.60	2.80	3.00
E1	1.50	1.60	1.70
L	0.35	0.45	0.55
L1	0.60 REF.		
e	0.95 BSC.		
e1	1.90 BSC.		
θ	0°	5°	10°
θ1	3°	5°	7°
θ2	6°	8°	10°

Dimension (Package B)

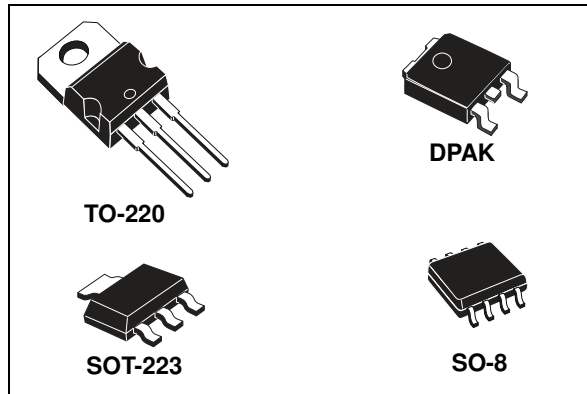


Unit : mm

SYMBOL	MIN.	TYP.	MAX.
A	1.050	-	1.250
A1	0.000	-	0.100
A2	1.050	-	1.150
b	0.300	-	0.400
c	0.100	-	0.200
D	2.820	-	3.020
E	1.500	-	1.700
E1	2.650	-	2.950
e	0.950 TYP		
e1	1.800	-	2.000
L	0.700 REF		
L1	0.300	-	0.600
θ	0°	-	8°

Adjustable and fixed low drop positive voltage regulator

Datasheet - production data



Description

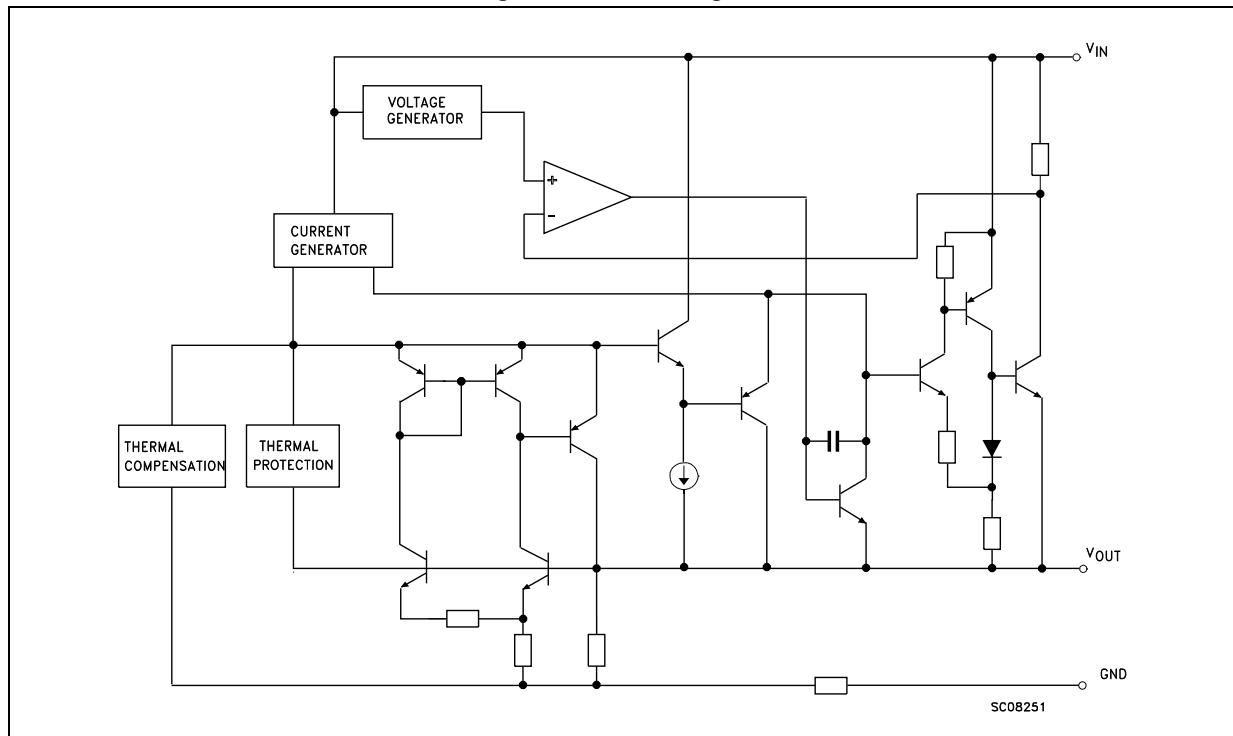
The LD1117 is a low drop voltage regulator able to provide up to 800 mA of output current, available even in adjustable version ($V_{REF} = 1.25\text{ V}$). Concerning fixed versions, are offered the following output voltages: 1.2 V, 1.8 V, 2.5 V, 2.85 V, 3.3 V and 5.0 V. The device is supplied in: SOT-223, DPAK, SO-8 and TO-220. The SOT-223 and DPAK surface mount packages optimize the thermal characteristics even offering a relevant space saving effect. High efficiency is assured by NPN pass transistor. In fact in this case, unlike than PNP one, the quiescent current flows mostly into the load. Only a very common $10\text{ }\mu\text{F}$ minimum capacitor is needed for stability. On chip trimming allows the regulator to reach a very tight output voltage tolerance, within $\pm 1\%$ at $25\text{ }^{\circ}\text{C}$. The adjustable LD1117 is pin to pin compatible with the other standard. Adjustable voltage regulators maintaining the better performances in terms of drop and tolerance.

Features

- Low dropout voltage (1 V typ.)
- 2.85 V device performances are suitable for SCSI-2 active termination
- Output current up to 800 mA
- Fixed output voltage of: 1.2 V, 1.8 V, 2.5 V, 3.3 V, 5.0 V
- Adjustable version availability ($V_{REF} = 1.25\text{ V}$)
- Internal current and thermal limit
- Available in $\pm 1\%$ (at $25\text{ }^{\circ}\text{C}$) and 2% in full temperature range
- Supply voltage rejection: 75 dB (typ.)

1 Diagram

Figure 1. Block diagram



3 Maximum ratings

Table 1. Absolute maximum ratings

Symbol	Parameter		Value	Unit
$V_{IN}^{(1)}$	DC input voltage		15	V
P_{TOT}	Power dissipation		12	W
T_{STG}	Storage temperature range		-40 to +150	°C
T_{OP}	Operating junction temperature range	for C version	-40 to +125	°C
		for standard version	0 to +125	°C

1. Absolute maximum rating of $V_{IN} = 18$ V, when I_{OUT} is lower than 20 mA.

Table 2. Thermal data

Symbol	Parameter	SOT-223	SO-8	DPAK	TO-220	Unit
R_{thJC}	Thermal resistance junction-case	15	20	8	5	°C/W
R_{thJA}	Thermal resistance junction-ambient	110	55	100	50	°C/W

Refer to the test circuits, $T_J = 0$ to $125\text{ }^{\circ}\text{C}$, $C_O = 10\text{ }\mu\text{F}$, unless otherwise specified.

Table 6. Electrical characteristics of LD1117#33

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
V_O	Output voltage	$V_{in} = 5.3\text{ V}$, $I_O = 10\text{ mA}$, $T_J = 25\text{ }^{\circ}\text{C}$	3.267	3.3	3.333	V
V_O	Output voltage	$I_O = 0$ to 800 mA , $V_{in} = 4.75$ to 10 V	3.235		3.365	V
ΔV_O	Line regulation	$V_{in} = 4.75$ to 15 V , $I_O = 0\text{ mA}$		1	6	mV
ΔV_O	Load regulation	$V_{in} = 4.75\text{ V}$, $I_O = 0$ to 800 mA		1	10	mV
ΔV_O	Temperature stability			0.5		%
ΔV_O	Long term stability	1000 hrs, $T_J = 125\text{ }^{\circ}\text{C}$		0.3		%
V_{in}	Operating input voltage	$I_O = 100\text{ mA}$			15	V
I_d	Quiescent current	$V_{in} \leq 15\text{ V}$		5	10	mA
I_O	Output current	$V_{in} = 8.3\text{ V}$, $T_J = 25\text{ }^{\circ}\text{C}$	800	950	1300	mA
eN	Output noise voltage	$B = 10\text{ Hz}$ to 10 kHz , $T_J = 25\text{ }^{\circ}\text{C}$		100		μV
SVR	Supply voltage rejection	$I_O = 40\text{ mA}$, $f = 120\text{ Hz}$, $T_J = 25\text{ }^{\circ}\text{C}$ $V_{in} = 6.3\text{ V}$, $V_{ripple} = 1\text{ V}_{PP}$	60	75		dB
V_d	Dropout voltage	$I_O = 100\text{ mA}$		1	1.1	V
		$I_O = 500\text{ mA}$		1.05	1.15	
		$I_O = 800\text{ mA}$		1.10	1.2	
	Thermal regulation	$T_a = 25\text{ }^{\circ}\text{C}$, 30 ms Pulse		0.01	0.1	%/W

Figure 12. Drawing dimension TO-220 (type STD-ST Dual Gauge)

