

---

# DIPLOMARBEIT

Gesamtprojekt

## C-Track 2.0

Entwurf einer Lokalisierungseinheit für Nutztiere

Samuel Putz 5BHEL

Betreuer: Prof. Dipl.-Ing. Siegbert Schrempf

Simon König 5BHEL

Kooperationspartner: Z\_GIS - Fachbereich Geoinformatik

ausgeführt im Schuljahr 2023/24

---

Abgabevermerk:

Datum: 02.04.2024

übernommen von: Siegbert Schrempf

## Eidesstattliche Erklärung

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

Salzburg, am 02.04.2024

Verfasserinnen / Verfasser:



Samuel Putz

---

Simon König

**DIPLOMARBEIT  
DOKUMENTATION**

Namen der Verfasserinnen / Verfasser	Samuel Putz Simon König
Jahrgang Schuljahr	5BHEL 2023/24
Thema der Diplomarbeit	C-Track 2.0 - Entwurf einer Lokalisierungseinheit für Nutztiere
Kooperationspartner	Z_GIS - Fachbereich Geoinformatik

Aufgabenstellung	Die genaue Lokalisierung von Nutztieren in alpinen Lagen ist von großem praktischen als auch von wissenschaftlichem Interesse. Derzeit am Markt befindliche Lösungen sind technisch unausgereift oder dementsprechend teuer. Deshalb sollte eine einfache und robuste Lokalisierungseinheit für Herdtiere entwickelt werden.
------------------	--

Realisierung	Mithilfe des LoRa Netzwerks werden die GPS-Daten von der Sendereinheit an der Kuh an einen Empfänger auf der Alm übermittelt und zusätzlich in einem kürzeren Zeitintervall auf einer SD-Karte gespeichert. Von dort aus werden die Daten an eine REST-API gesendet und in eine SQL-Datenbank gespeichert. Von dieser werden die Daten an den Nutzer übermittelt.
--------------	---

Ergebnisse	Das Endergebnis ist eine ausfallsichere Lokalisierungseinheit für Nutztiere, welche die gesammelten Daten für die Analyse des sozialen Herdenverhaltens auf einer SD-Karte speichert und dem Landwirt bei der Suche der Tiere hilft.
------------	--

Typische Grafik, Foto etc. (mit Erläuterung)	
<i>Abbildung 1: Projekt Logo</i>	

Teilnahme an Wettbewerben, Auszeichnungen	Keine Teilnahmen	
Möglichkeiten der Einsichtnahme in die Arbeit	Schulbibliothek der HTBLuVA Salzburg	
Approbation (Datum / Unterschrift)	Prüferin / Prüfer	Direktorin / Direktor Abteilungsvorständin / Abteilungsvorstand

## DIPLOMA THESIS

### Documentation

Author(s)	Samuel Putz Simon König
Form Academic year	5BHEL 2023/24
Topic	C-Track 2.0 - Design of a localization unit for livestock
Co-operation Partners	Z_GIS – department of geoinformatics

Assignment of Tasks	The precise location of farm animals in alpine locations is of great practical and scientific interest. Solutions currently on the market are technically immature or correspondingly expensive. Therefore, a simple and robust localization unit for herd animals should be developed.
---------------------	---

Realisation	Using the LoRa network, the GPS data is transmitted from the circuit board on the cow to a receiver on the mountain pasture and is also stored on an SD card at a shorter time interval. From there, the data is sent to a REST API and stored in a SQL database. From here the data is transmitted to the user.
-------------	--

Results	The result brings a fail-safe farm animal locator that stores the collected data in an SD card for both social herd behaviour analysis and can help the farmer locate and check the animals' well-being.
---------	--

Illustrative Graph, Photo  
(incl. explanation)



Abbildung 2: Project Logo

Participation in  
Competitions  
Awards

No participations

Accessibility of  
Diploma Thesis

Library of the Higher Technical College in Salzburg

Approval  
(Date / Sign)

Examiner

Head of College  
Head of Department

## Vorwort

C-Track wurde im vergangenen Jahr initiiert, jedoch blieb es unvollendet. Die Bedeutung dieses Projekts für die Forschung am Zentrum für Geoinformatik (ZGIS), sowie für Bauern in alpinen Regionen ist von großer Tragweite. Aus diesem Grund haben wir es uns zur Aufgabe gemacht, dieses Projekt unter dem Namen C-Track 2.0 fortzusetzen, zu restrukturieren und zu optimieren. Die Idee hinter C-Track erwies sich als äußerst vielversprechend, jedoch stellten sich zahlreiche Herausforderungen auf unserem Weg. Um die Anforderungen der Kooperationspartner und unsere eigenen Ideen umzusetzen, wurden viele Hürden überwunden und neues Wissen angeeignet. Jedes Teammitglied arbeitete dabei in dem Fachgebiet, das er am besten beherrschte.

## Danksagung

Wir möchten uns sehr herzlich bei allen Personen, die uns bei der Realisierung unserer Diplomarbeit unterstützt haben, bedanken.

Besonderen Dank möchten wir an folgende Personen aussprechen:

- Prof. Dipl.-Ing. Siegbert Schrempf für die Unterstützung als unser Projekt Betreuer
- Der ZGIS, insbesondere Frau Dr. Assoc. Prof. Gudrun Wallentin als Ansprechpartnerin für das Projekt.
- Oliver Gollenz für die Unterstützung beim Platinenentwurf
- Bei unseren Vorgängern Samuel Neureiter und Marvin Klabacher
- Arno Krenslehner für Unterstützung im Bereich der drahtlosen Übertragung

Ohne der Unterstützung dieser Personen wäre das Projekt in diesem Ausmaß nicht möglich gewesen.

# 1 Inhalt

---

1	Einleitung .....	11
2	Systemspezifikation .....	12
2.1	Zielbestimmungen .....	12
2.1.1	Musskriterien.....	12
2.1.2	Wunschkriterien .....	12
2.1.3	Abgrenzungskriterien .....	12
2.2	Produkteinsatz .....	13
2.2.1	Anwendungsbereiche.....	13
2.2.2	Zielgruppen .....	13
2.2.3	Betriebsbedingungen .....	13
2.3	Entwicklungsumgebung .....	14
2.3.1	Software.....	14
2.3.2	Hardware .....	14
2.3.3	Orgware .....	14
2.4	Produktfunktionen .....	15
2.5	Produktdaten.....	16
2.6	Produktleistungen.....	17
2.7	Qualitätszielbestimmungen .....	18
2.8	Globale Testszenarien und Testfälle .....	19
3	Organisation - Projektmanagement .....	20
3.1	Projektteam .....	20
3.2	Individuelle Aufgabenstellungen inkl. Arbeits- und Terminplan .....	21
3.2.1	Samuel Putz.....	21
3.2.2	Simon König.....	22
4	Grundlagen und Methoden .....	23
4.1	Sendeeinheit .....	24
4.1.1	Energieversorgung.....	24
4.1.2	Spannungsregler.....	24
4.1.3	GPS-Modul.....	31
4.1.4	Mikrocontroller.....	35
4.1.5	SD-Karte .....	36
4.1.6	Gehäuse der Sendeeinheit .....	40
4.1.7	Programmablauf der Sendeeinheit.....	41
4.2	LoRa Standard .....	43

---

4.2.1	Anforderungen an die Funkübertragung .....	43
4.2.2	Rechtliche Rahmenbedingungen .....	44
4.2.3	Modulation von LoRa-Signalen .....	44
4.2.4	Codierung von LoRa .....	45
4.2.5	Antennen.....	48
4.2.6	SX1276 LoRa-Transceiver .....	49
4.3	Basisstation .....	50
4.3.1	TTGO T-Beam .....	50
4.3.2	PlatformIO .....	50
4.3.3	Programmablauf der Basisstation .....	51
4.4	Server.....	53
4.4.1	Laravel .....	53
4.5	Smartphone Applikation .....	58
4.5.1	Anforderungen der Applikation.....	58
4.5.2	Flutter.....	58
4.6	Verbesserungen .....	66
4.6.1	Platine .....	66
4.6.2	Server .....	66
4.6.3	Smartphone Applikation .....	66
5	Ergebnisse – Abnahme .....	67
6	Literaturverzeichnis .....	69
7	Abkürzungen.....	72
8	Abbildungen .....	73
9	Begleitprotokoll gemäß § 9 Abs. 2 PrO .....	75
9.1	Begleitprotokoll Samuel Putz .....	75
9.2	Begleitprotokoll Simon König .....	76
10	Anhang .....	77

## 1 Einleitung

Die genaue Lokalisierung von Nutztieren in alpinen Lagen ist von großem praktischen, sowie wissenschaftlichem Interesse. Derzeit am Markt befindliche Lösungen sind technisch unausgereift oder entsprechend teuer. Deshalb sollte eine einfache und robuste Lokalisierungseinheit für Herdentiere entwickelt werden.

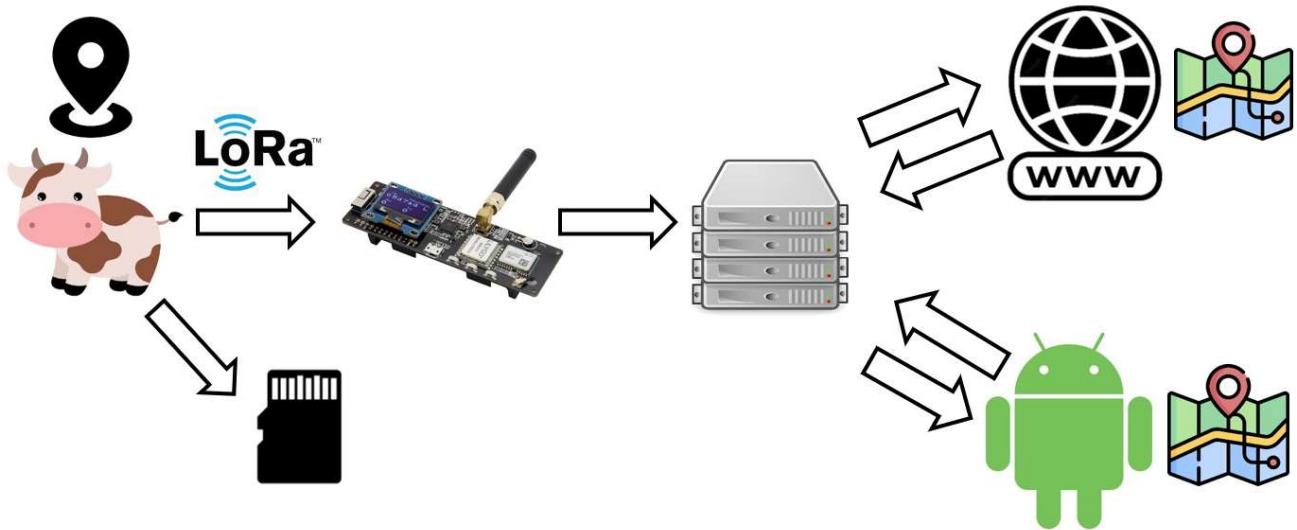


Abbildung 3: Blockschaltbild des Lokalisierungssystems

Mithilfe der LoRa Funkübertragung werden die GPS-Daten von der Sendeplatine an der Kuh an einen Empfänger auf der Alm übermittelt und zusätzlich in einem kürzeren Zeitintervall auf einer SD-Karte gespeichert. Von dort aus werden die Daten an eine REST-API gesendet und in eine SQL-Datenbank gespeichert. Von dieser werden die Daten an den Nutzer übermittelt.

## 2 Systemspezifikation

---

### 2.1 Zielbestimmungen

#### 2.1.1 Musskriterien

- Outdoorfähiges Gehäuse
- GPS-Daten müssen auf SD-Karte gespeichert werden
- Standort auf wenige Meter genau
- Hohe Reichweite der Übertragung
- Geringer Leistungsverbrauch
- Benutzerfreundliche Handyapplikation

#### 2.1.2 Wunschkriterien

- Webseite, um Standort anzeigen zu lassen
- Lange Akkulaufzeit der Sendeeinheit
- Login für User in Handy App

#### 2.1.3 Abgrenzungskriterien

- Die Applikation am Mobiltelefon ist nur mit Android kompatibel
- Die Basisstation muss eine stabile Internetverbindung haben
- Das Mobiltelefon muss einen Internetzugriff haben

## 2.2 Produkteinsatz

### 2.2.1 Anwendungsbereiche

Das Produkt soll in alpinen und schwer zugänglichen Regionen den Viehbetrieb für Landwirte erleichtern, sowie Informationen über das Herdeverhalten bringen.

### 2.2.2 Zielgruppen

Dieses Projekt dient als Hilfestellung für Landwirte in alpinen Regionen sowie eine Erleichterung in der Forschung.

### 2.2.3 Betriebsbedingungen

- Platine:  
Die Platine soll bei jeder Wetterbedingung GPS-Daten aufnehmen und an die Basisstation senden. Die Stromversorgung per Batterie soll dabei über einen langen Zeitraum standhalten. Zielumgebung ist eine offene Weidefläche.
- Basisstation:  
Die Basisstation soll in einer Almhütte geschützt von Wind und Wetter mit einer stabilen Internetverbindung stationiert werden. Zudem soll ein geeigneter Anbringungsort für die Antenne vorhanden sein.
- Smartphone Applikation:
  1. Die Applikation ist für Android Mobiltelefone ausgelegt.
  2. Es wird eine Internetverbindung benötigt, um die aktuellen Daten abzurufen.

## 2.3 Entwicklungsumgebung

### 2.3.1 Software

- EAGLE: Entwurf der Platine (V: 9.6.2)
- Autodesk Fusion: Entwurf des Gehäuses
- Visual Studio Code: Programmierung von Handy App, Backend und Basisstation (V: 1.87.0)
- Microchip Studio: Programmierung des Mikrocontrollers (V: 7.0.2594)

### 2.3.2 Hardware

- ATmega644 PA (Mikrocontroller)
- Elara-1 (GPS-Modul)
- RFM95W (Lora Breakoutmodul)
- TTGO T-Beam 1.1 (Basisstation)
- LPRS-ANT-868-DP-N-F (Antenne Bodenstation)
- ISM 868/915MHz Dipole (Antenne Sender)
- Android Smartphone

### 2.3.3 Orgware

- OneDrive: Cloud Speicher (V: 24.025.0204.0003)
- Draw.io: Diagramme (V: 24.1.0)

## 2.4 Produktfunktionen

### **/F0010/ Empfang der Standortdaten**

Die Sendeeinheit soll mittels einem GPS-Moduls den Standort ermitteln können.

### **/F0020/ Speichern auf SD-Karte**

Die GPS-Daten sollen in einem 5-Minuten Intervall auf einer SD-Karte gespeichert werden.

### **/F0030/ Datenübertragung**

Die GPS-Daten sollen mittels LoRa an den Empfänger übertragen werden.

### **/F0040/ Datenverwaltung**

Die GPS-Daten sollen in einer Datenbank gespeichert und dem Nutzer zur Verfügung gestellt werden.

### **/F0050/ Daten abrufen**

Die GPS-Daten sollen graphisch in einer Applikation für Mobiltelefone und auf einer Website angezeigt werden können.

## 2.5 Produktdaten

### /D0010/ Daten des Benutzers

- Name für Username
- Passwort zur Authentifizierung
- E-Mail zur Anmeldung

### /D0020/ Standortdaten des Weidetieres

- Zeit für zeitliche Einordnung
- Längengrad für Standort
- Breitengrad für Standort
- Kuh Id für Identifizierung des Weidetieres

## 2.6 Produktleistungen

### /L0010/ Benutzerfreundlichkeit

Die Bedienung der Smartphone Applikation soll möglichst einfach und selbsterklärend sein.

### /L0020/ Wetterbeständigkeit

Die Sendeeinheit soll wind- und wasserfest sein.

### /L0030/ Genauigkeit

Die GPS-Daten sollen auf ca. 2m genau sein.

### /L0040/ Laufzeit

Die Stromversorgung der Sendeeinheit soll ca. 100 Tage ohne Batteriewechsel funktionieren.

## 2.7 Qualitätszielbestimmungen

	Sehr wichtig	Wichtig	Weniger wichtig	Unwichtig
<b>Benutzerfreundlichkeit</b>		X		
<b>Wetterbeständigkeit</b>	X			
<b>GPS- Genauigkeit</b>		X		
<b>Laufzeit</b>	X			

Ziel des Projekts ist es eine Lokalisierungseinheit zu entwickeln. Diese soll für den Almbetrieb optimiert werden. Während der Almsaison ist es nicht möglich, die Batterie zu wechseln. Daher ist eine lange Akkulaufzeit essenziell. Die Lokalisierungseinheit muss dem Wetter standhalten können und sollte dabei einen möglichst genauen Standort liefern. Dieser soll in einer einfach gestalteten App leicht einsehbar sein.

## 2.8 Globale Testszenarien und Testfälle

### /T0010/ Empfang der Standortdaten

Die Sendeeinheit kann eine Verbindung zu ausreichend Satelliten aufstellen und die Standortdaten ermitteln.

### /T0020/ Speichern auf SD-Karte

Die GPS-Daten werden in einem definierten Intervall auf einer SD-Karte gespeichert.

### /T0030/ Datenübertragung

Die GPS-Daten werden mittels LoRa an den Empfänger übertragen.

### /T0040/ Datenverwaltung

Die GPS-Daten werden in einer Datenbank gespeichert und sind für die Applikation und die Website zugänglich.

### /T0050/ Daten abrufen

Der Standort wird graphisch in einer Karte sowohl in der Applikation für Mobiltelefone als auch in der Website angezeigt.

## 3 Organisation - Projektmanagement

### 3.1 Projektteam

Name	Individuelle Themenstellung	Klasse	Arbeitsaufwand
<b>Samuel Putz</b>	Entwurf der Hardware-Komponenten	5BHEL	180 Stunden
<b>Simon König</b>	Entwurf der Systemsoftware und der Datenbank	5BHEL	180 Stunden

## 3.2 Individuelle Aufgabenstellungen inkl. Arbeits- und Terminplan

### 3.2.1 Samuel Putz

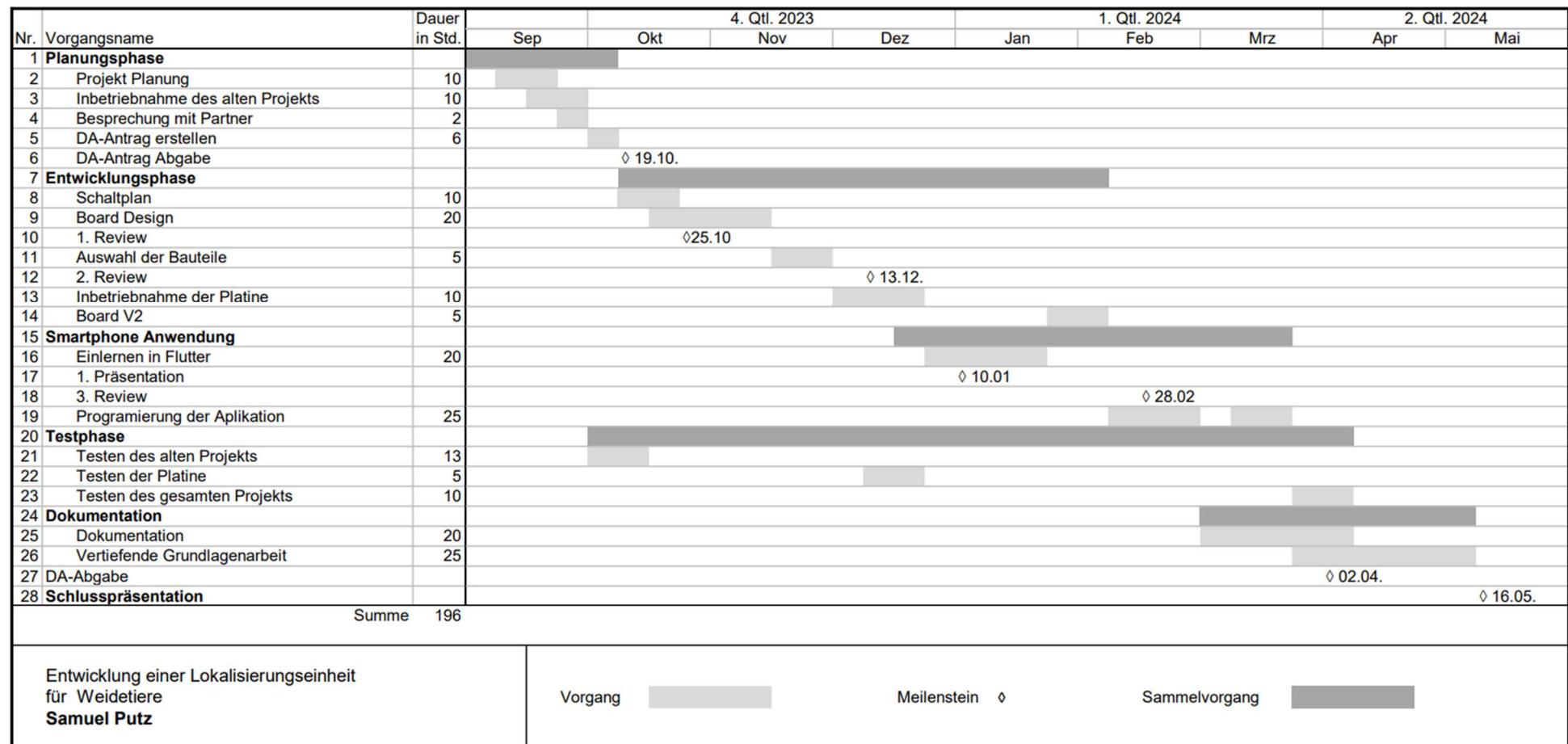


Abbildung 4: GANT-Diagramm Putz

### 3.2.2 Simon König

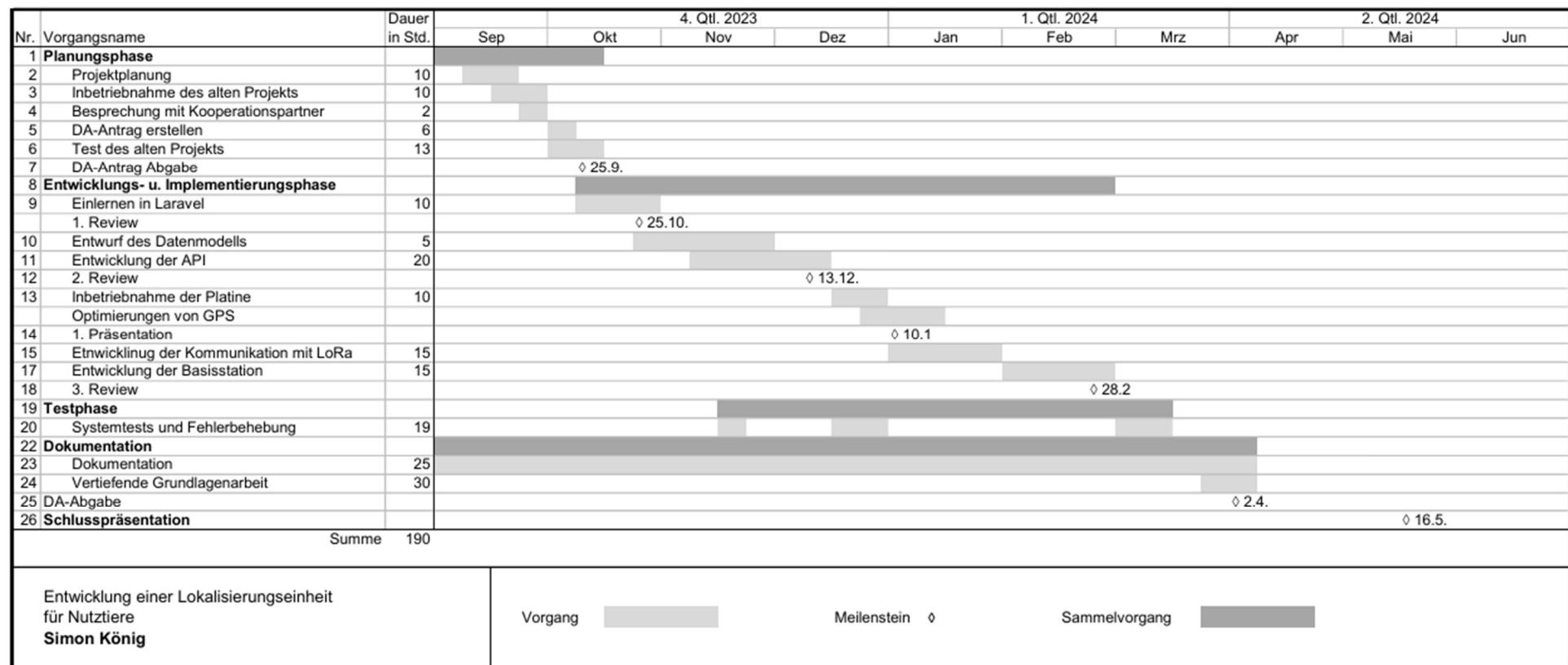


Abbildung 5: GANTT-Diagramm König

## 4 Grundlagen und Methoden

Das System kann in vier grundlegende Blöcke aufgeteilt werden. Die Sendeeinheit ist auf der Kuh befestigt. Diese empfängt die Standortdaten und speichert sie auf einer SD-Karte. Ist ein LoRa-Modul vorhanden, übermittelt diese die Daten an die Basisstation. Die Basisstation empfängt die Daten und schreibt diese mit HTTP-Requests auf den Server. Der Server speichert die Daten und stellt sie für die Applikation wieder zur Verfügung. Der Ablauf und die dafür verwendeten Komponenten wurden in einem Blockdiagramm (siehe Abb 6.) zusammengefasst.

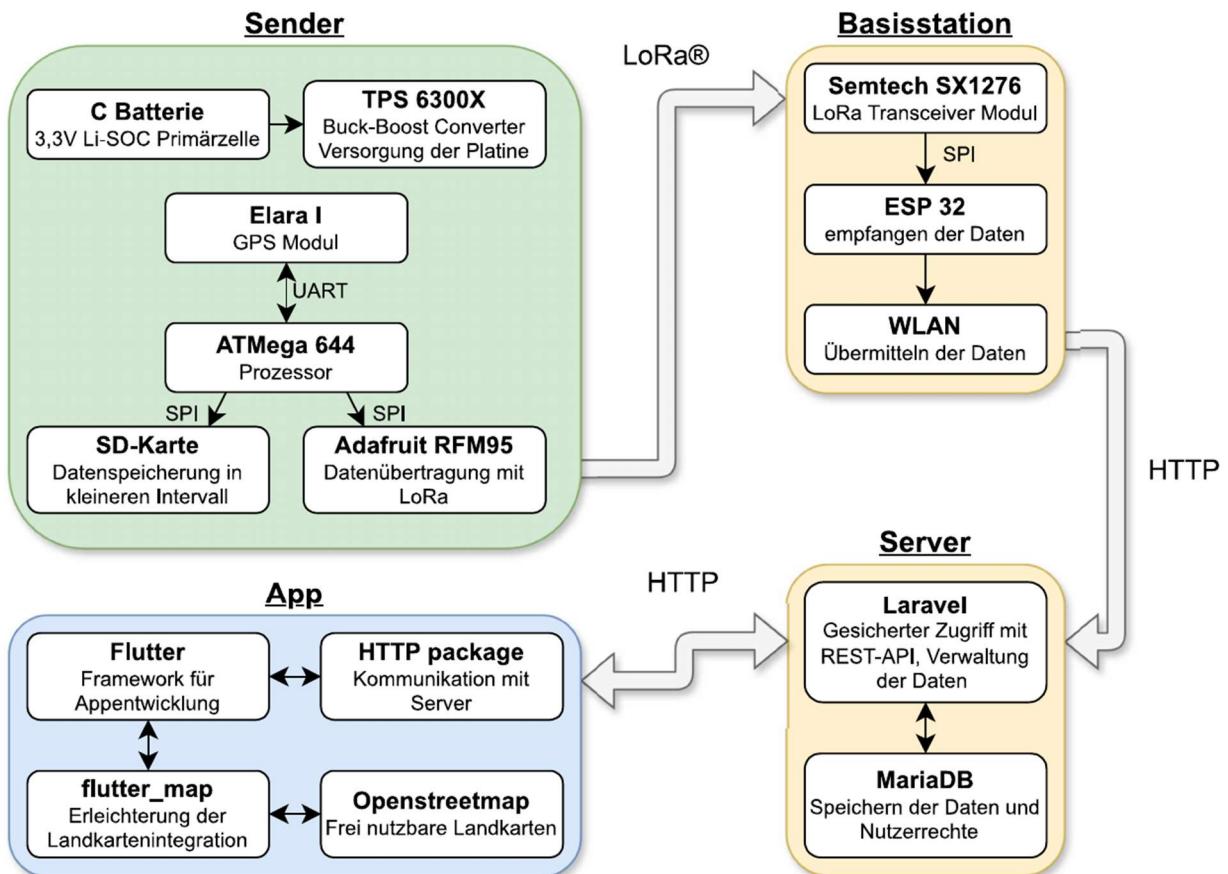


Abbildung 6: Blockdiagramm des C-Track Systems

## 4.1 Sendeeinheit

Die Sendeeinheit, wird am Hals der Kuh angebracht und in einem wasserfesten Gehäuse sicher verschraubt. Das Elara-1 GPS-Modul stellt eine Verbindung mit einem Satelliten her, um GPS-Daten zu empfangen. Diese Daten werden sowohl auf der SD-Karte gespeichert als auch über das RFM95W LoRa Breakoutmodul an die Basisstation übertragen. Die zentrale Steuereinheit der Platine ist ein energieeffizienter ATmega644 PA Mikrocontroller. Zur Gewährleistung einer zuverlässigen Stromversorgung der Bauteile wurden mehrere Spannungsregler integriert.

### 4.1.1 Energieversorgung

Das Ziel des Projekts ist es, einem Landwirt auf der Vierkaseralm während der Sommersaison von 3 Monaten dabei zu unterstützen, die Kühe rund um die Uhr zu überwachen und dem Landwirt deren Position zu übermitteln. Um die Funktionsfähigkeit für diesen Zeitraum zu gewährleisten und um eine gute und stabile Stromversorgung sicherzustellen, musste die Entscheidung zwischen einem Akkumulator und einer herkömmlichen Batterie getroffen werden.

Ein Akkumulator bietet den Vorteil, dass er wiederaufladbar ist und somit die Notwendigkeit eines regelmäßigen Austauschs vermeidet. Aus den folgenden Gründe wurde jedoch eine Batterie für unser Projekt als die effizientere Variante angesehen.

Zum einen bietet eine Batterie eine höhere Energiedichte als ein Akkumulator, was für die kurzzeitig hohen Stromspitzen des GPS-Moduls und des Lora Breakout Boards erforderlich ist. Zum anderen ist eine Batterie nicht so anfällig für Temperaturschwankungen wie ein Akkumulator [1].

Ein Akkumulator benötigt zusätzliche Regler, um vor Tiefenentladung geschützt zu sein. Zum Aufladen wird zudem eine zusätzliche Ladeschaltung sowie ein Port benötigt. Dies erhöht die Komplexität der Schaltung durch eine höhere Anzahl an Bauteilen und steigert dadurch wieder die Kosten. Ein Ladeport erschwert zudem die Wasserdichtheit des Gehäuses, wenn dieser von außen zugänglich sein sollte.

### 4.1.2 Spannungsregler

Eine Batterie verliert im Laufe ihrer Lebensdauer immer mehr an Spannung. Die Bauteile auf der Platine müssen jedoch gemäß den Spezifikationen mit einer konstanten Spannung von 3,3 V versorgt werden. Da die Batterie im geladenen Zustand einen Spannungsbereich von 3,6 V und im entladenen Zustand 2,7 V aufweist, wird ein Spannungsregler benötigt. Für unsere Zwecke wurde ein Buck-Boost-Konverter als die optimale Lösung angesehen, da diese Schaltung die Spannung bei Bedarf sowohl herunter- als auch hinaufregeln kann.

#### 4.1.2.1 Buck Converter

Der Buck Converter ist ein DC - DC - Wandler, um höhere Eingangsspannungen in kleinere Ausgangsspannungen umzuwandeln. Der Vorteil dieser Schaltung liegt in ihrer einfach gehaltenen Bauweise und einer dadurch geringeren Bauteilanzahl. Zudem besitzt der Buck Converter einen wesentlich höheren Wirkungsgrad als Linearregler.

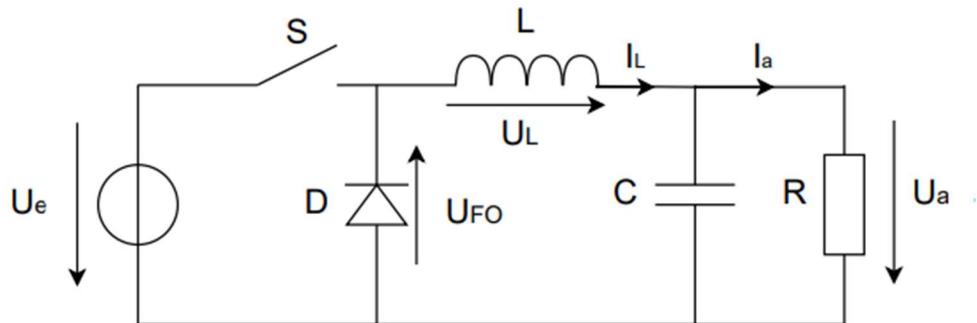


Abbildung 7: Buck Converter Schaltung

Typischerweise ist der Schalter ein MOSFET. Während der Einschaltphase fließt der Strom von der Eingangsspannung über die Spule zum Ausgang und zum Lastwiderstand, wodurch die Spule Energie speichert. Im ausgeschalteten Zustand widersteht die Spule die Änderung des Stroms und somit fließt der Strom weiterhin durch den Lastwiderstand und die Diode. Die Spule setzt die gespeicherte Energie frei, um den Stromfluß aufrechtzuerhalten. In diesem Zustand wird die Ausgangsspannung aufrechterhalten, obwohl der Eingangsspannungswert höher ist.

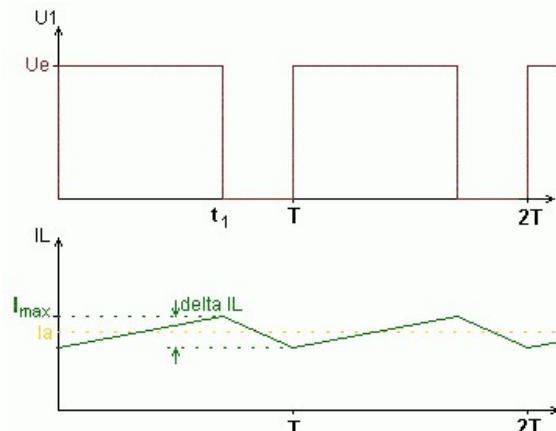


Abbildung 8: Strom und Spannungsverlauf des Buck Converters [38]

Es ist zu erkennen, dass ein sägezahnähnlicher Spulenstrom entsteht. Dies ist auf die Funktionsweise der Induktivität zurückzuführen, da dieser sich gemäß einer e-Funktion lädt und entlädt. Durch das schnelle Ein- und Ausschalten des Schalters befindet sich die Ladekurve jedoch in einem Bereich, in dem die Änderung so klein ist, dass sie quasi linear wirkt und somit den sägezahnähnlichen Stromverlauf erzeugt.

Der Wirkungsgrad liegt typischerweise bei um die 90 Prozent. Verluste in Form von Verlustleistung treten bei Bauteilen wie der MOSFET, der Diode und der Spule auf. Beim MOSFET entstehen zusätzlich weitere Verluste durch den Wechsel zwischen Ein- und Ausgangszuständen. Am Kondensator treten sogenannte kapazitive Verluste auf, da beim Laden und Entladen Energie abgegeben wird.

#### 4.1.2.1.1 Dimensionierung der Mindestinduktion

Bei der Berechnung spielt die Spule eine entscheidende Rolle, da sie so dimensioniert werden muss, dass sie ausreichend Energie speichern kann, um die gewünschte Ausgangsspannung zu erreichen.

Die Auswahlkriterien beziehen sich dabei auf die Schaltfrequenz ( $f$ ) in der der Schalter ein und ausschaltet, dem Ausgangstrom ( $I_A$ ) und der gewünschten Ausgangsspannung ( $U_A$ ), sowie der Eingangsspannung ( $U_E$ ).

Die Stromwelligkeit wird üblicherweise mit ungefähr 20% des Ausgangstroms dimensioniert.

$$\Delta I_L = 0,2 * I_{OUT} \quad (1)$$

Wählt man die Stromwelligkeit zu klein erhöht es den Induktionswert der Spule, was zu höheren Kosten führen kann.

Um den Wert der Spule wird zunächst die Ausgangsspannung Formel benötigt:

$$U_A = \frac{t_{ein}}{T} * U_E \quad (2)$$

Formel (2) wird auf  $t_{ein}$  umgeformt:

$$t_{ein} = \frac{U_A}{U_E} * T \quad (3)$$

Nun wird  $t_{ein}$  (3) in die Formel der Induktivität (4) eingesetzt und anstatt T wird  $1/f$  eingesetzt:

$$L = \frac{1}{\Delta I_L} * (U_E - U_A) * t_{ein} \quad (4)$$

Somit entsteht die finale Formel zur Berechnung der Mindestinduktion:

$$L = \frac{1}{\Delta I_L} * (U_E - U_A) * \frac{U_A}{U_E} * \frac{1}{f} \quad (5)$$

#### 4.1.2.2 Boost Converter

Der Boost Converter ist wie der Buck Converter ein DC-DC-Wandler, welcher diesmal jedoch eine kleinere Eingangsspannung zu einer höheren Ausgangsspannung transformiert. Er besitzt die gleichen Baukomponenten eines Buck Converters und bringt somit dieselben Vor- und Nachteile wie dieser.

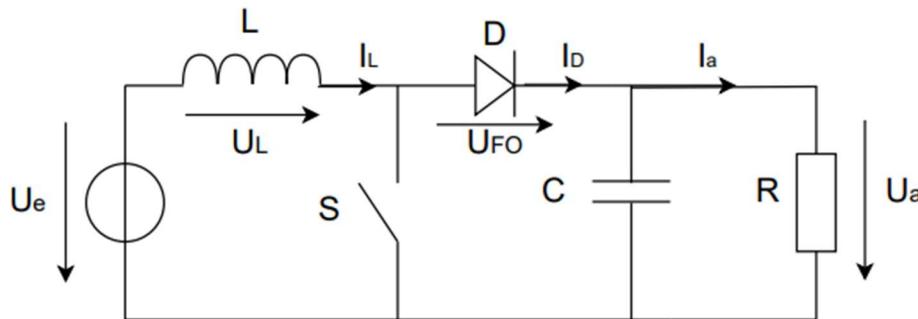


Abbildung 9: Boost Converter

Der Schalter ist hier ebenfalls typischerweise ein MOSFET. Wenn der Schalter geschlossen wird, fließt der Strom über die Spule und den geschlossenen Schalter von Plus nach Minus. Dabei lädt sich die Spule auf. Wird der Schalter geöffnet erhält die Spule den Stromfluss aufrecht. Der Strom fließt jetzt durch die Diode zum Ausgang wobei der Ausgangskondensator aufgeladen wird. Da die Spule, während der Einschaltphase Energie speichert, kann am Ausgang eine höhere Spannung als am Eingang erzielt werden.

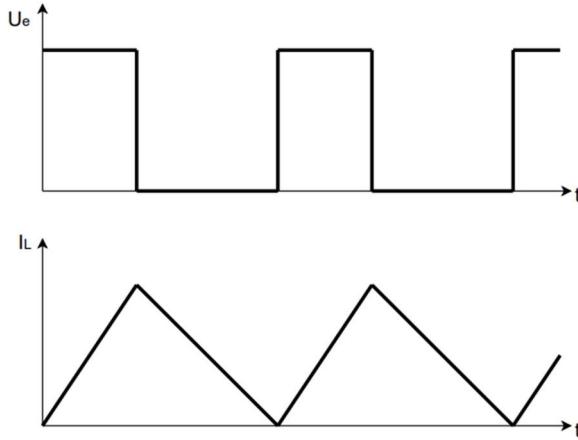


Abbildung 10: Strom und Spannungsverlauf des Boost Converters

Wie beim Buck Converter lässt sich auch in diesem Fall der Sägezahnverlauf des Induktionsstrom auf dasselbe Phänomen der Lade und Entladekurve der Induktivität zurückführen.

Durch die gleichen Bauteile wie für einen Buck Converter ergibt sich ein ähnlicher Wirkungsgrad von rund 90 Prozent, erklärbar durch dieselben Verluste.

#### 4.1.2.2.1 Dimensionierung der Mindestinduktivität:

Ähnlich zum Buck Converter beziehen sich die Auswahlkriterien erneut auf die Schaltfrequenz ( $f$ ) in der der Schalter ein und ausschaltet, dem Ausgangsstrom ( $I_a$ ) und der gewünschten Ausgangsspannung ( $U_a$ ), sowie der Eingangsspannung ( $U_E$ ).

Auch hier wird die Stromwelligkeit mit 20% Prozent des Ausgangstromes gewählt siehe (1).

Um den Wert der Spule wird zunächst die Ausgangsspannung Formel benötigt:

$$U_A = \frac{1}{1 - \frac{t_{ein}}{T}} * U_E \quad (6)$$

Formel (2) wird nun auf  $t_{ein}$  umgeformt:

$$t_{ein} = T - \frac{U_E}{U_A} * T \quad (7)$$

Nun wird  $t_{ein}$  (7) in die Formel der Induktivität (8) eingesetzt und anstatt T wird 1/f eingesetzt:

$$L = \frac{1}{2} * \frac{U_E}{\Delta I_L} * t_{ein} \quad (8)$$

Somit entsteht die finale Formel zur Berechnung der Mindestinduktion:

$$L = \frac{1}{2} * \frac{U_E}{\Delta I_L} * \left( \frac{1}{f} - \frac{U_E}{U_A} * \frac{1}{f} \right) \quad (9)$$

#### 4.1.2.3 Buck Boost Converter

Will man die Vorteile und Effizienz beider Schaltungen in einer einzigen Schaltung kombinieren entsteht der Buck - Boost Converter. Mit der gleichen Bauteilanzahl eines Buck oder Boost Converters kann dieser sowohl eine niedrige Eingangsspannung hochregeln als auch eine hohe Eingangsspannung herunterregeln.

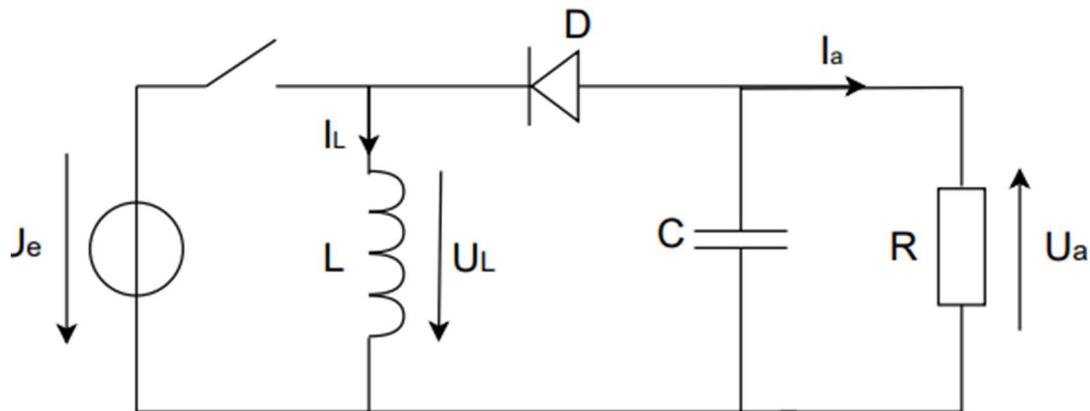


Abbildung 11: Buck - Boost - Converter

Wird der Schalter geschlossen fließt der Strom über die Spule da die Diode in Sperrichtung ist, von Plus nach Minus. Wird der Schalter geöffnet wird die Spule durch ihre Selbstinduktion von einem Verbraucher zu einem Generator und die Spannung dreht sich um. Diesmal fließt der Strom weiter über den Kondensator und über die in Durchlassrichtung betriebene Freilaufdiode, um den Stromfluß aufrecht zu erhalten. Dabei lädt sich der Kondensator so auf das am Ausgang eine negative Spannung abgegriffen werden kann, welche größer bzw. kleiner als die Eingangsspannung sein kann.

Der Buck Boos Converter besitzt zwei Operationsmodi [2]:

- **Continuous Conduction Mode:** In diesem Modus wird der Induktionsstrom nie Null und bleibt positiv. Dies bringt eine höhere Effizienz sowie eine niedrigere Welligkeit. Der Nachteil bei diesem Modus liegt darin, dass ein höherer Induktivitätswert benötigt wird
- **Discontinuous Conduction Mode:** Bei diesem Modus kann es dazu kommen, dass, während den Schaltzyklen kurz kein Induktionsstrom fließt. Höhere Spannungsripple, höhere Schaltverluste und insgesamt weniger Effizienz ist dadurch die Folge. Jedoch wird eine niedrigere Induktivität benötigt.

#### 4.1.2.4 TPS 63001 DRCT

Für unsere Platine entschieden wir uns für einen TPS 63001 Buck Boost Converter. Bei einer Eingangsspannung von 1,8 – 5,5V und einer Ausgangsspannung von 3,3 V entsprach dieser Chip genau unseren Ansprüchen. Zudem bietet er einen Wirkungsgrad von bis zu 96% und liefert einen hohen Ausgangstrom der sowohl für das GPS-Modul als auch den Microcontroller benötigt wird.

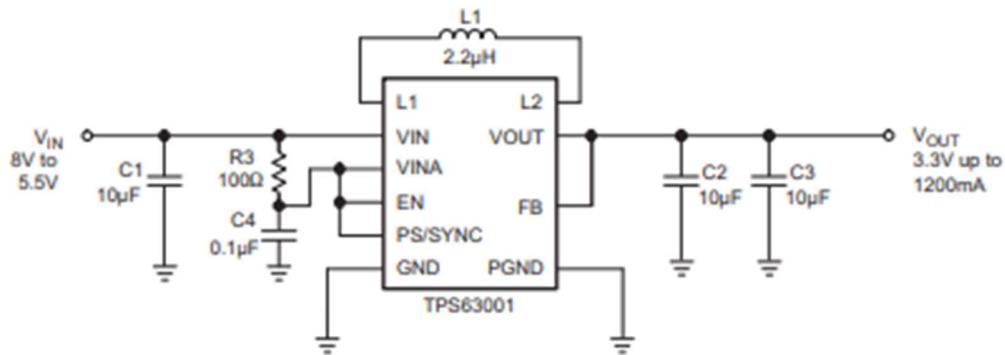


Abbildung 12: Typische Applikation des TPS 63001 [3]

Bei der TPS 6300x Reihe kann man mit Widerständen am Ausgang die gewünschte Ausgangsspannung in einem weiten Bereich konfigurieren. Für unsere Zwecke wurde der TPS 63001 gewählt da dieser so eingestellt ist das er auf ein fixes Ua von 3,3 V regelt und zudem einen Ausgangsstrom von bis zu 1,2 A ausgibt. Die weiteren Bauteilwerte wurden aus dem Datenblatt entnommen [3].

Es wurde eine Messung der Ausgangsspannungsripple des entworfenen Buck Boost Konverters unternommen und mit der Spezifikation verglichen.

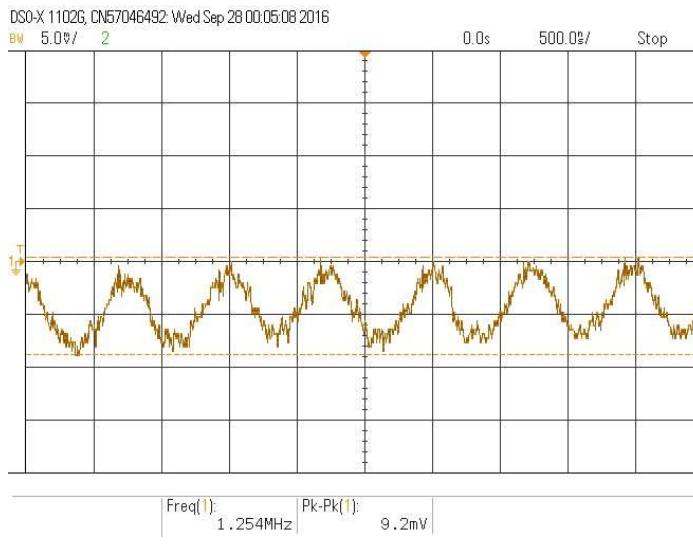


Abbildung 13: Messung der Ausgangripples an der Platine

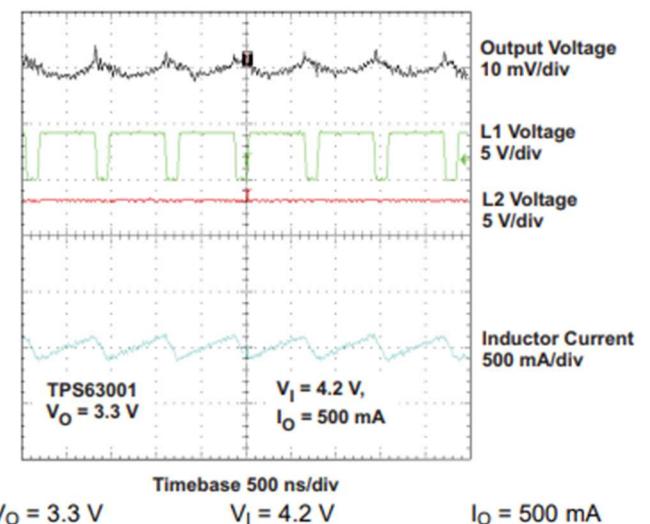


Abbildung 14: Messungen des Datenblattes [3]

Bei einer Eingangsspannung von 5V ergab sich ein Spannungsripple am Ausgang mit einem Wert von 9,2mV. Aus der Spezifikation ergibt sich ein Spannungsripple am Ausgang von um die 10mV bei einer Eingangsspannung von 4,2V. Vergleicht man nun unsere Messung mit der Spezifikation ist zu erkennen das die Rippel unserer Buck Boost Schaltung der Spezifikation nahezu ident sind.

### 4.1.3 GPS-Modul

Für die Standortermittlung wurde das von Würth Elektronik entwickelten Elara-I GPS-Modul verwendet. Das Modul kann aufgrund der integrierten Antenne ohne aufwendige Beschaltung in die Platine integriert werden und ist aufgrund des kleinen Formfaktors gut für Tracker geeignet.



Abbildung 15: Elara-I Modul [4]

Die integrierte Antenne hat eine hohe Empfindlichkeit und kann auch schwache Signale auswerten. Die Position wird entweder über das GPS oder das GLONASS-Satellitensystem auf bis zu 1,5m genau ermittelt [4].

#### 4.1.3.1 Startvorgang

Nach Versorgung mit Spannung beginnt das Modul mit dem Startvorgang. Ist dieser beendet, sendet das Modul einen kurzen Puls über den WAKE\_UP-Pin zum Microcontroller und wechselt in den Stand-by-Modus. Um die Standortermittlung zu starten, muss vom Microcontroller ein Puls an den ON\_OFF-Pin angelegt werden. Dieser muss zwischen 100µs und 1s lang sein[4].

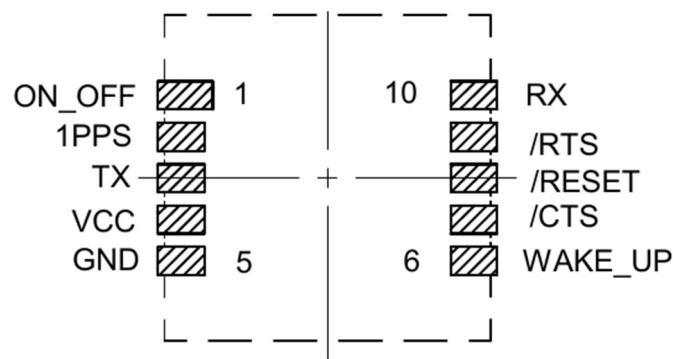


Abbildung 16: Pinout Elara-I Modul

#### 4.1.3.2 Kommunikation

Der Datenaustausch mit dem Elara-1 Modul erfolgt über eine UART-Schnittstelle. Die Daten für die Kommunikation werden in standardisierten Sätzen übertragen. Die Sätze sind nach dem NMEA 0183 Standard aufgebaut. Dieser wurde von der „National Marine Electronic Association“ entwickelt. Für komplexere Funktionen muss die Kommunikation auf proprietäre, von Qualcomm entwickelte, OSP-Sentences umgeschaltet werden. Nach dem Start beginnt das Elara-1 Modul Daten in NMEA-Sentences zu senden.

NMEA-Sentences sind in ASCII codiert. Der Aufbau folgt der Struktur aus Abbildung 17. Die Daten sind in einzelne Felder aufgeteilt. Die einzelnen Felder werden mit einem Beistrich getrennt.

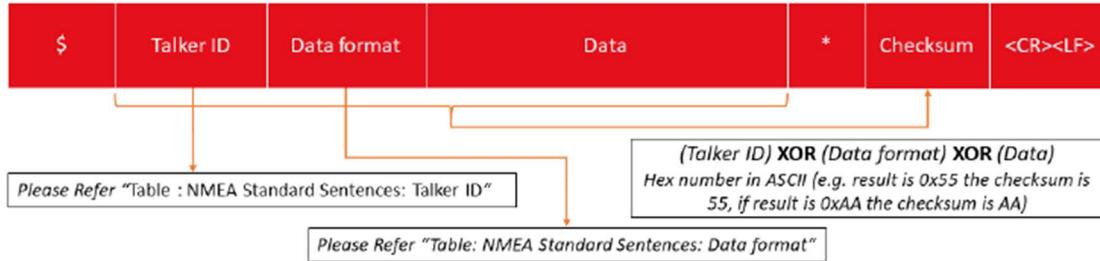


Abbildung 17: Aufbau einer NMEA-Satz [4]

Das Elara-Modul versendet Daten über die Position, über die verwendeten Satelliten, über die Uhrzeit und das Datum und über die Geschwindigkeit und Richtung. Diese Informationen werden in unterschiedlichen Sätzen einzeln übertragen. Welche Informationen in den Felder sind des übertragenen Satzes sind, ist vom Data-Format (Abbildung 17, 3. Feld) abhängig. Diese Formate sind im NMEA 0183 Standard festgelegt. Eine Liste mit allen unterstützten Formaten findet man im Datasheet. Je nach benötigten Informationen können die Formate ein- und ausgeschalten werden.

Um alle Funktionen und Einstellungen, des Moduls nutzen zu können, muss das Modul in den OSP-Modus versetzt werden. OSP-Sentences sind ein von Qualcomm entwickeltes Format, um direkt mit dem Chip zu kommunizieren. Statt mit ASCII Werten arbeitet das System mit Hexadezimalzahlen. Daten werden in binärer Form übertragen. [4]

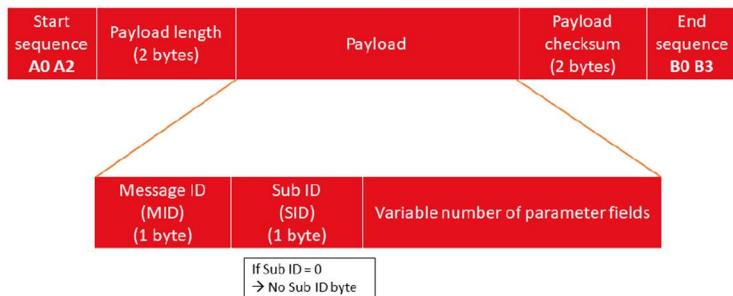


Abbildung 18: Aufbau eines OSP-Satz [4]

OSP-Sätze beginnen mit einer Startsequenz. Die Daten sind bei OSP in Messages aufgeteilt. Die Messages sind mit den Datenformaten aus NMEA vergleichbar. Ähnlich wie bei NMEA wird der Satz mit einer Prüfsumme und einer Endsequenz beendet.

Der OSP-Modus sendet detailliertere Daten über das System und die Positionsermittlung. Anders als bei NMEA können die nicht benötigten Informationen nicht deaktiviert werden. Durch den höheren Verarbeitungsaufwand entstanden Probleme bei der Datenverarbeitung. Deshalb wurde für die Kommunikation der NMEA-Modus gewählt. [4]

NMEA-Sätze im RMC-Format liefern alle benötigten Informationen. Alle anderen Sätze werden deaktiviert. Aus dem RMC-Format erhält man Datum, Uhrzeit, Längen- und Breitengrad, sowie Bewegungsgeschwindigkeit und Richtung. Abbildung 19 zeigt einen typischen NMEA-Satz mit Daten im RMC-Format. Man erkennt den NMEA-Rahmen aus Abbildung 17 mit Satellitensystem, Datenformat, Datenfelder und Prüfsumme.

\$GNRMC,143954.321,A,4749.2698,N,01302.6803,E,0.08,159.09,201223,,,A*74
---

Abbildung 19: NMEA-Sentence mit RMC-Format

Mit der Tabelle aus Abbildung 20 kann man die benötigten Informationen aus einer NMEA-Sentece mit Daten im RMC-Format auslesen. [4]

Field	Description
UTC Time	hhmmss.sss (Hours Minutes Seconds)
Status	A: Data Valid V: Data not Valid
Latitude	ddmm.mmmm (Degree Minutes)
N/S	N: North S: South
Longitude	ddmm.mmmm (Degree Minutes)
E/W	E: East W: West
Speed Over Ground	in Knots
Course Over Ground	in Degrees
Date	ddmmmyy (Day Month Year)
Empty field	Empty field
Empty field	Empty field
Mode	A: Autonomous (standard) N: Output Data Not Valid R: Coarse Position (SV states based on almanac, not ephemeris)

Abbildung 20: RMC-Format [4]

Um die Information über Längen- und Breitegrad in Anwendungen wie Google Maps, oder der flutter\_map Bibliothek nutzen zu können, werden diese in Dezimalgrad umgerechnet. Dazu teilt man den gegebenen Wert in Grad und Minuten auf. Die Minuten können mit einer Division durch 60 in Grad umgerechnet werden und können dann zum Gradwert addiert werden. Dezimalgrad berücksichtigen die Information über die Halbkugel. Ist der Längengrad auf der Südhalbkugel oder der Breitengrad auf der Westhalbkugel muss man der Zahl ein negatives Vorzeichen geben.

Für den Satz aus Abbildung 19 ergeben sich dadurch folgende Berechnungen.

$$\text{Längengrad: } 47 + \frac{49,2698}{60} = 47,821163$$

$$\text{Breitengrad: } 13 + \frac{2,6803}{60} = 13,04467$$

Da der Punkt auf der Nordhalbkugel und auf der Osthalbkugel liegt, sind beide Werte positiv.

#### 4.1.3.3 Trickle Power mode

Das Elara-1 verfügt über spezielle Modi für das Sparen von Energie. Für das Projekt interessant ist der Trickle Power mode. In diesem wechselt das Modul in einem festgelegten Zeitintervall automatisch aus dem Stand-By in den Empfangsmodus. Nach dem Empfangen der Daten wechselt das Modul in den CPU-Modus, um die Informationen auszuwerten. Wurden alle gesammelten Informationen gesendet, wechselt das Modul wieder in den Stand-By Modus. [4]

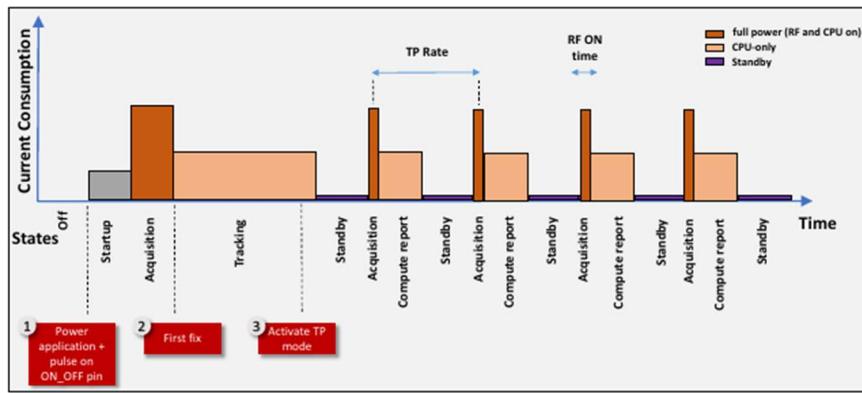


Abbildung 21: Trickle Power mode [4]

Der Modus funktioniert nur, wenn mit OSP-Sentences kommuniziert wird. Da die Kommunikation mit OSP-Sentences nicht zufriedenstellend implementiert werden konnte wird dieser Modus nicht verwendet.

#### 4.1.3.4 Design Rules

Das wichtigste beim Elara 1 Modul ist die Keep Out Area, die im Datenblatt beschrieben wird. Diese ist in einem 3mm Radius um das Modul herum. Um eine symmetrische Ground Flächen Verteilung zu gewährleisten, soll das Modul in der Mitte der Platine platziert werden. Um eine bestmögliche Erdung zu erreichen, wird geraten eine reine Ground Plate am Bottomlayer einzufügen [4].

#### 4.1.4 Mikrocontroller

Als Steuerungselement unserer Platine wurde ein ATmega644 PA gewählt. Da er eine vielseitige Auswahl an Schnittstellen bietet, leistungsfähig ist und sehr stromeffizient arbeitet, ist dieser Chip die perfekte Wahl. Zudem wurde der ATmega644 im Schulunterricht intensiv behandelt, was den Einstieg wesentlich erleichterte.

##### 4.1.4.1 Schnittstellen

Für unser Projekt wurde die SPI und UART-Schnittstelle verwendet

- **SPI:** Diese Schnittstelle wurde für das LoRa Breakout Modul sowie die SD-Karte verwendet. SPI wird grundsätzlich für den Datenaustausch zwischen Microcontroller und anderen Peripherie Geräten wie etwa Sensoren oder Speicherchips verwendet. Der Vorteil von SPI liegt an ihrer hohen Geschwindigkeit und der einfachen Konfiguration
- **UART:** Die UART-Schnittstelle wird in unserem Projekt für die Kommunikation und Ansteuerung des Elara 1 Moduls verwendet. UART wird typischerweise für die Kommunikation zwischen Microcontroller und externe Geräte wie Computer oder andere Microcontroller verwendet. Es bietet eine flexible und zuverlässige Datenübertragung über lange Strecken.

##### 4.1.4.2 Interner Quarz Oszillatior

Der ATmega644 verfügt über einen internen Quarz, wodurch die Notwendigkeit eines externen Quarzes oder eines Oszillators entfällt. Dadurch verringert sich die Komplexität der Schaltung, was die Zuverlässigkeit erhöht und durch die verringerte Bauteilanzahl auch stromsparend wirken kann, sowie preislindernd wirkt.

##### 4.1.4.3 Stromverbrauch

Der ATmega644 ist für den Betrieb mit Batterien optimiert und verfügt dadurch über verschiedene Stromsparmodi. Dies erhöht die Lebensdauer der Batterie und maximiert die Effizienz. Stromsparmodi wie etwa der Power-down-, Standby-, oder Sleep Modus können den Stromverbrauch verringern, indem sie interne Schaltkreise oder Funktionen deaktivieren. Zusätzlich ist es möglich Peripheriegeräte selektiv zu aktivieren, was wiederum einen stromsparenden Effekt hervorruft. Die Taktfrequenz des Microcontrollers kann dynamisch angepasst werden, was ebenso stromsparend wirkt. Besonders hervorhebend ist der niedrige Ruhestrom des Standby Modus, welcher mit dem schon ohnehin niedrigen Stromverbrauch perfekt für unser Projekt ist.

## 4.1.5 SD-Karte

Mit dem Tracker sollte das Herdenverhalten von Kühen analysiert werden. Dafür sollte der Standort lückenlos alle 5 Minuten gespeichert werden. Ausgewertet werden die Daten am Ende der Almsaison.

Damit die Daten gespeichert werden, selbst wenn keine Funkverbindung besteht, ist eine lokale Speicherung erforderlich. Als Speichermedium wurden MicroSD-Karten gewählt. MicroSD-Karten sind die weit verbreitet und lassen sich mit SPI gut in das System einbinden. Durch den kleinen Formfaktor sind MicroSD-Karten gut für einen Tracker geeignet. Der einzige Nachteil ist die kurze Lebensdauer der Speicherzellen.

Für die leichte Auswertung der Daten vom Nutzer ist es erforderlich, die Daten in eine Datei auf einem Dateisystem zu speichern. Aufgrund der Kompatibilität mit vielen Betriebssystemen wurde das Fat32 Dateisystem gewählt.

Da bei jedem GPS-Punkt die gleichen Daten entstehen, werden diese in tabellarischer Form im CSV-Format gespeichert. Es werden die Daten von einem Zeitpunkt in einer Zeile gespeichert. Die einzelnen Daten werden mit einem Beistrich getrennt. Aufgrund der einfachen Struktur können die Daten leicht in eigenen Programmen ausgewertet werden.

### 4.1.5.1 Abnutzung

Das von Microsoft entwickelte FAT-Dateisystem. Arbeitet mit einer Tabelle, die in den ersten Zellen des Speichers liegt. In dieser sind Informationen über Speicherort, Größe und Änderungszeiten der Dateien. Sobald in eine Datei geschrieben wird, muss diese Tabelle aktualisiert werden. [5] Dadurch werden die ersten Zellen stärker beansprucht, was bei Flash Speicher ein Problem ist.

Die Flash-Speicherzellen einer SD-Karte können, je nach Bauform, 1-3 Bit speichern. Single Level Cells, kurz SLC, können nur 1 Bit speichern. Dafür bieten diese höhere Lese- und Schreibgeschwindigkeiten und einen niedrigeren Leistungsverbrauch. Multi- oder Triple Level Cells bieten mehr Speicherplatz auf der gleichen Fläche, können aber nicht so oft beschrieben werden. Während SLCs oft bis zu 100.000 Schreibzyklen überstehen können TLCs schon nach 3000 Schreibvorgängen ausfallen. [6] [7]

Um dem entgegenzuwirken, verbauen viele Speicherhersteller einen sogenannten wear leveling Algorithmus. Dieser unterteilt den Speicher in Blöcke. Der Nutzer schreibt nicht direkt auf den Speicher, sondern in einen logischen Block. Werden Daten in einen Block geschrieben, wird der Speicherbereich mit den wenigsten Schreibzyklen beschrieben und eine Verknüpfung vom logischen Block zu dem physischen Speicherblock erstellt. Dadurch werden die Schreibzyklen auf alle Zellen verteilt und die Lebensdauer erhöht. [6]

Damit die Daten sicher auf die SD-Karte geschrieben werden können, sollte eine SLC SD-Karte mit wear-leveling verwendet werden. Diese Karten werden oft mit dem Zusatz „Für industrielle Anwendungen“ verkauft.

Um die Abnutzung zu reduzieren, wurde geplant, die Standortdaten in einen Buffer zu speichern und nur einmal pro Tag auf die SD-Karte zu schreiben. Aus zeitlichen Gründen konnte dies nicht implementiert werden.

#### 4.1.5.2 Initialisierung

Die einfachste Möglichkeit eine SD-Karte anzusteuern ist der SPI-Modus. Für diesen werden nur die 4 SPI-Leitungen und eine Versorgung benötigt. Um mit allen SD-Karten kompatibel zu sein, muss für den Startprozess eine Frequenz zwischen 100kHz-400kHz verwendet werden. Nach dem Start kann die Frequenz angehoben werden. Die maximale Frequenz variiert nach Hersteller und kann aus dem CRD-Register ausgelesen werden.

Die Kommunikation erfolgt mit Befehlen. Ein Befehl startet mit dem ersten Bit auf LOW, dem zweiten Bit auf HIGH. Darauf folgt der eigentliche Befehl als Hexadezimalzahl. Ein Befehl ist, mit Ausnahme der ACMD-Befehle und des CMD55 Befehls, ein Byte lang. Diesem Befehl werden die benötigten Argumente mitgegeben. Die wichtigsten Befehle für die Kommunikation werden in Abbildung 22 zusammengefasst.

Command Index	Argument	Response	Data	Abbreviation	Description
CMD0	None(0)	R1	No	GO_IDLE_STATE	Software reset.
CMD1	None(0)	R1	No	SEND_OP_COND	Initiate initialization process.
ACMD41(*1)	*2	R1	No	APP_SEND_OP_COND	For only SDC. Initiate initialization process.
CMD8	*3	R7	No	SEND_IF_COND	For only SDC V2. Check voltage range.
CMD9	None(0)	R1	Yes	SEND_CSD	Read CSD register.
CMD10	None(0)	R1	Yes	SEND_CID	Read CID register.
CMD12	None(0)	R1b	No	STOP_TRANSMISSION	Stop to read data.
CMD16	Block length[31:0]	R1	No	SET_BLOCKLEN	Change R/W block size.
CMD17	Address[31:0]	R1	Yes	READ_SINGLE_BLOCK	Read a block.
CMD18	Address[31:0]	R1	Yes	READ_MULTIPLE_BLOCK	Read multiple blocks.
CMD23	Number of blocks[15:0]	R1	No	SET_BLOCK_COUNT	For only MMC. Define number of blocks to transfer with next multi-block read/write command.
ACMD23(*1)	Number of blocks[22:0]	R1	No	SET_WR_BLOCK_ERASE_COUNT	For only SDC. Define number of blocks to pre-erase with next multi-block write command.
CMD24	Address[31:0]	R1	Yes	WRITE_BLOCK	Write a block.
CMD25	Address[31:0]	R1	Yes	WRITE_MULTIPLE_BLOCK	Write multiple blocks.
CMD55(*1)	None(0)	R1	No	APP_CMD	Leading command of ACMD<n> command.
CMD58	None(0)	R3	No	READ_OCR	Read OCR.

\*1:ACMD<n> means a command sequence of CMD55-CID<n>.  
 \*2: Rsv(0)[31], HCS[30], Rsv(0)[29:8]  
 \*3: Rsv(0)[31:12], Supply Voltage(1)[11:8], Check Pattern(0xAA)[7:0]

Abbildung 22: Befehle für die Kommunikation mit SD/MMC-Karten (vgl. [8])

Als Antwort kommt in den meisten Fällen eine R1-Response. Mit einer R1-Antwort erhält man Informationen über die Verarbeitung des Befehls und den Status des Speichermediums und ist ein Byte lang (vgl. Abbildung 23). Bei fehlerfreier Verarbeitung sind alle Bits LOW.

#### R1 Response

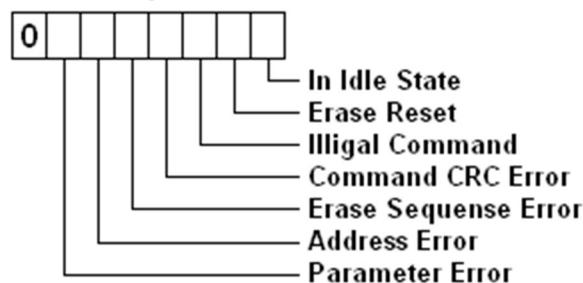


Abbildung 23: R1-Response [8]

Eine Sonderform der R1-Antwort ist die R1b Antwort. Diese wird versendet, wenn die Verarbeitung noch nicht abgeschlossen wurde. Für die Dauer des Prozesses wird MISO auf LOW gesetzt. Nach Erhalt von 0xFF auf der Datenleitung ist die Verarbeitung beendet.

Werden Daten von der SD-Karte ausgelesen, schreibt die SD-Karte eine R3 oder R7 Antwort auf den SPI-Bus. Diese besteht aus einer R1-Antwort, gefolgt von 32 Datenbits, aus welchen der Inhalt der Speicherzelle hervorgeht.

Nachdem die SD-Karte mit Spannung versorgt wurde, dauert es ca. 1ms., bis alle Speicherzellen initialisiert sind. Zu beachten ist, dass die Karte zu dem Zeitpunkt noch kein SPI kompatibles Gerät ist und die Signale von Chipselect nicht einhält. Bevor die Kommunikation gestartet werden kann, benötigt die SD-Karte mindestens 74 Clock Signale. Chipselect und MOSI müssen dabei HIGH sein. Die Karte sollte danach dazu bereit sein, die native Befehle zu verarbeiten.

Um die SD-Karte zu resetten und alle Einstellungen für die Kommunikation zu löschen wird der Befehlt GO\_IDLE\_STATE gesendet. Zu beachten ist, dass der SPI-Modus noch nicht aktiviert ist und eine CRC-Prüfsumme mitgeschickt werden muss. Nach dem Reset testet die Karte das Chipselect Signal. Ist dieses LOW wird der SPI-Modus aktiviert. Als Antwort auf den GO\_IDLE\_STATE Befehl kommt eine R1-Antwort. Wichtig ist, dass das Idle-Bit gesetzt ist.

Um den vollen Funktionsumfang zu nutzen, muss noch, abhängig der Version des SD-Standards, ein SEND\_OP\_COND-Befehl oder ein APP\_SEND\_OP\_COND-Befehl gesendet werden. Um mit allen Versionen kompatibel zu sein, wird empfohlen einen Befehlt zu senden und die Antwort auszuwerten. War der Befehlt nicht erfolgreich muss der andere Befehl verwendet werden. Die SD-Karte verlässt den Idle-Zustand wieder und kann ab jetzt beschrieben und gelesen werden [8].

#### 4.1.5.3 Dateisystem

Damit die Dateien auf möglichst vielen Geräten ausgewertet werden können wird das Fat32 Dateisystem verwendet. Da das Dateisystem seit 1996 existiert, müssen historisch bedingte Einschränkungen beachtet werden. Es gibt keine Schutzmechanismen gegen ausfallende Zellen. Dateinamen dürfen nur Großbuchstaben enthalten und maximal 12 Zeichen lang sein. Daten sind bei defekten Speicherzellen verloren. Zudem kann eine Datei kann maximal 4GB groß sein. [9]

Eine Im CSV-Format gespeicherte Zeile mit Informationen zu Datum, Uhrzeit, Längen- und Breitengrad ist 39 Zeichen lang. Die Textdatei ist im UTF-8 Format kodiert. Ist das zu speichernde Zeichen ein ASCII-Zeichen benötigt man ein Byte um dieses zu Speichern. Alle Zeichen, die für die Speicherung der Daten verwendet werden, sind aus dem ASCII-Zeichensatz.

$$\frac{\text{Daten}}{\text{Stunde}} * \frac{\text{Stunden}}{\text{Tag}} * \text{Länge der Saison} * \text{Größe eines Eintrags} = \text{gesamte Datenmenge}$$

Die Standordaten sollten alle 5 Minuten gespeichert werden. Eine Stunde hat 60 Minuten. Dadurch erhält man 12 Datenpunkte pro Stunde. Eine Almsaison dauert ungefähr 100 Tage.

$$12 * 24 * 100 * 39 = 1123200 \text{ Bytes} \triangleq 1.1232 \text{ Megabyte}$$

Die limitierte Dateigröße stellt somit kein Problem dar.

#### 4.1.5.4 FatFS Bibliothek

Aufgrund der hohen Komplexität von Dateisystemen wurde die FatFS-Bibliothek für das Lesen und Schreiben in Dateien verwendet. Da diese speziell für Microcontroller entwickelt wurde, wird wenig Arbeitsspeicher benötigt.

Das Modul hat jedoch keine Möglichkeiten mit Speichermedien zu kommunizieren. Für die Kommunikation gibt es Funktionsprototypen in einer C-Header Datei. Für diese müssen die Funktionen der Hardware entsprechend ausprogrammiert werden. Um den Prozess zu erleichtern, werden Beispielprojekte zur Verfügung gestellt. Als Ausgangspunkt wurde das Generic-SD Beispiel verwendet. In diesem wurden die Funktionen für die Kommunikation mit SD-Karten ausprogrammiert. Die Funktionen wurden an den Microcontroller angepasst und der Code für die Software-SPI Kommunikation durch Code für das SPI-Modul des Microcontrollers ersetzt.

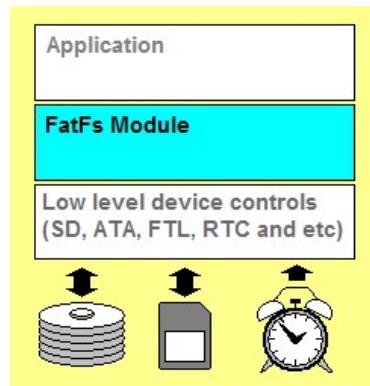


Abbildung 24: typische Struktur für Verwendung der FatFS-Bibliothek [8]

Durch Konfigurationen der benötigten Funktionen in der Datei ffhconf.h kann der Verbrauch des Programmspeichers gesenkt werden. Für das Projekt wurden der Parameter FF\_FS\_MINIMIZE auf drei gesetzt. Dadurch verliert man Funktionen zum Umbenennen, Löschen und Erstellen von Ordnern, sowie zur Suche von Dateien.

Um die Daten formatiert auf die SD-Karte schreiben zu können wurde FF\_USE\_STRFUNC auf zwei gesetzt. Dadurch wird die `f_printf` Funktion verfügbar. Diese Funktion kann einen String formatieren und in einer Datei speichern. Mit der FF\_PRINT\_FLOAT flag wurde die Unterstützung für Gleitkommazahlen aktiviert.

## 4.1.6 Gehäuse der Sendeeinheit

Bevor das Layout der Platine entworfen wurde, mussten wir uns für ein Gehäuse entscheiden. Der ursprüngliche Plan sah vor, ein 3D-Gehäuse selbst zu entwerfen und zu drucken. Das Gehäuse muss wind- und witterfest sein, um im Freien auf der Alm zu funktionieren. Daher wurde der Plan für ein selbstgedrucktes Gehäuse aufgegeben, da es einen erheblichen Zeit- und Kostenaufwand erfordern würde, um ein 3D-gedrucktes Gehäuse wasserdicht zu machen. Aus diesem Grund entschieden wir uns für ein gekauftes Gehäuse.

### 4.1.6.1 Polycarbonat Gehäuse

Es wurde ein Polykarbonat-Gehäuse von RS Components mit der Schutzklasse IP65 gewählt [10]. Die IP-Norm ist eine international anerkannte Klassifizierung, welche in verschiedenen Branchen, wie unter anderem in der Elektronik, verwendet wird und die Eignung eines Produktes für bestimmte Umgebungsbedingungen angibt.

1. Kennziffer	Schutz gegen Fremdkörper und Berührung	2. Kennziffer	Schutz gegen Wasser
0 ungeschützt		ungeschützt	
1 geschützt gegen feste Fremdkörper >50 mm		geschützt gegen Tropfwasser	●
2 geschützt gegen feste Fremdkörper >12 mm		geschützt gegen Tropfwasser unter 15°	
3 geschützt gegen feste Fremdkörper >2,5 mm		geschützt gegen Sprühwasser	■
4 geschützt gegen feste Fremdkörper >1mm		geschützt gegen Spritzwasser	▲
5 geschützt gegen Staub		geschützt gegen Strahlwasser	▲▲
6 dicht gegen Staub	◆	geschützt gegen schwere See	
7 ---	◆	geschützt gegen zweitweises Eintauchen	●●
8 ---		geschützt gegen dauerndes Untertauchen	●● ... m
9 ---		Schutz gegen Wasser bei Hochdruck-/Dampfstrahlreinigung	

Abbildung 25: Tabelle der IP-Schutzklassen [11]

Basierend auf den Informationen in der oben genannten Tabelle (Abb.: 22) entschieden wir uns dafür, dass die IP-Schutzklasse 65 für unsere Anforderungen perfekt geeignet ist. Somit ist das Gehäuse effektiv gegen Staub und andere Fremdkörper geschützt und bietet zudem Schutz vor Strahlwasser, was für den Betrieb auf der Alm ausreichend sein wird.

Nach der Auswahl des Gehäuses wurden die Maße der Platine so festgelegt, dass sie fest und sicher im Gehäuse verschraubt werden kann.

### 4.1.6.2 Luftöffnung

Eine Luftöffnung für ein Gehäuse ist aus mehreren Gründen wichtig. Zum einen ermöglicht sie eine effektive Wärmeableitung und kühlt somit die Bauteile. Zum anderen reduziert sie Feuchtigkeit und verhindert Kondensation im Gehäuse, während sie gleichzeitig einen Druckausgleich zwischen Gehäuse und Umwelt ermöglicht. Wir haben uns für eine M12 Industrial Luftöffnung mit der Schutzklasse IP68 entschieden. Diese gewährleistet einen guten Luftaustausch und schützt gleichzeitig vor dem Eindringen von Wasser und Fremdkörpern [11].

#### 4.1.7 Programmablauf der Sendeeinheit

Das Programm der Sendeeinheit ist dafür verantwortlich, dass die Position und der Zeitpunkt vom GPS-Modul empfangen und verarbeitet werden. Diese Informationen werden danach auf die SD-Karte geschrieben und mit LoRa an die Basisstation übertragen.

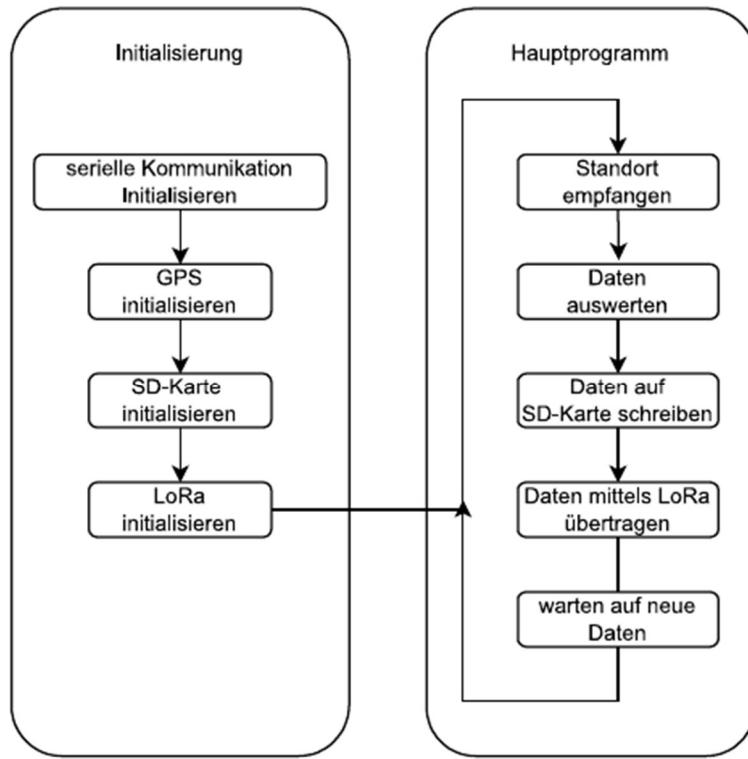


Abbildung 26: Programmablauf der Sendeeinheit

Nach dem Start des Microcontrollers werden die benötigten Variablen erstellt und die Pins und die seriellen Kommunikationsschnittstellen SPI und UART aktiviert.

Mit einem Puls am ON\_OFF-Pin wird das GPS-Modul eingeschalten. Über UART werden die nicht benötigten NMEA-Sätze deaktiviert.

```

/*
    The GPS Module is default in stand-by mode
    wake up with pulse on PD4
*/
PORTD |= (1 << PD4);
_delay_ms(100);
PORTD &= ~(1 << PD4);

uart_sendStr("$PSRF103,1,0,0,1*25\r\n");
...
uart_sendStr("$PSRF103,9,0,0,1*2D\r\n");

```

Um die SD-Karte und LoRa nutzen zu können, werden die beiden Geräte über den SPI-Bus initialisiert. Wichtig ist, dass die SD-Karte nach dem Start kein SPI-Gerät ist und sich nicht an Chipselect hält. Um Probleme bei der Datenübertragung an SPI-Geräte zu vermeiden, muss die SD-Karte vor der Kommunikation mit den anderen SPI-Teilnehmern initialisiert werden (vgl. 4.1.5.2).

```
f_mount(&FatFs, "", 0); // SD initialisieren und Dateisystem mounten

lora = lora_init(); // LoRa init

lora_setSpreadingFactor(12);
lora_set_bandwidth(0b0111); // 125 kHz
lora_putd("Tracker started", 15);
```

In einer While-Schleife wartet das Programm auf Daten von der UART-Schnittstelle. Wird ein Satz empfangen, liest das Programm bis zum Satzende mit '\n'.

```
while (1)
{
    while ((buff[++i] = uart_receive()) != '\n');
    buff[i] = '\0';
```

Die Daten werden mit einer Funktion aus dem Satz ausgelesen und in ein dafür erstelltes Struct geschrieben. Ist der Satz gültig gibt die Funktion eine logisches „true“ zurück und schreibt den Satz im CSV-Format in eine Datei.

```
if (parseRMC(buff, &p))
{
    fr = f_open(&Fil, "pos.txt", FA_OPEN_APPEND | FA_WRITE);

    if (fr == FR_OK)
    {
        f_printf(&Fil, "%02d/%02d/%02d %02d:%02d:%02d;%f;%f\r\n", p.hour, p.minutes,
p.seconds, p.day, p.month, p.year, p.lat, p.lng);

        fr = f_close(&Fil);
    }
}
```

Bei der Initialisierung wird geprüft, ob die Platine ein LoRa-Modul hat. Ist ein LoRa-Modul vorhanden werden die Daten mit der sprintf-Funktion in einen String geschrieben und mit LoRa an die Basisstation übermittelt. Die Daten sind dabei mit Leerzeichen getrennt im Format Tracker ID, Stunden, Minuten, Sekunden, Tag, Monat, Jahr, Längengrad, Breitengrad. Da die sprintf-Funktion des AVR-GCC Compilers keine Kommazahlen unterstützt, werden die Zahlen mit 100.000 multipliziert. Der Standort wird dadurch auf 1,1m genau übertragen.

```
if (lora)
{
    int len = sprintf(buff, "%02d %02d %02d %02d %02d %02d %"
PRIId32 " %" PRIId32 "", T_ID,
                    p.hour,
                    p.minutes,
                    p.seconds,
                    p.day,
                    p.month,
                    p.year,
                    (int32_t)(100000* p.lat),
                    (int32_t)(100000* p.lng));

    lora_putd(buff, len);
}
```

## 4.2 LoRa Standard

LoRa, kurz für Long Range, ist ein von Semtech entwickeltes, patentiertes Funkübertragungsverfahren, welches hohe Reichweiten bei wenig Stromverbrauch erreichen kann. Dadurch konnte sich diese Technik vor allem IOT-Bereich durchsetzen.

LoRa nutzt einen lizenzenfreien Bereich im ISM-Band. Dadurch fallen keine Gebühren an. Durch die Modulationstechnik sind Überlagerungen mit anderen Teilnehmern in diesem Frequenzbereich nur selten ein Problem.

Mithilfe von Parametern wie dem Spreadingfactor und der Coding Rate kann die Übertragung entweder auf hohe Reichweite und robuste Übertragung oder schnelle Übertragungsraten mit geringerer Reichweite angepasst werden [12].

### 4.2.1 Anforderungen an die Funkübertragung

Der Standort sollte für den Landwirt spätestens jede  $\frac{1}{2}$  Stunde aktualisiert werden. Da der Tracker die ganze Saison ohne Batteriewechsel laufen sollte, muss stromsparend gearbeitet werden. Die Kühe können sich auf dem Gebiet der Alm frei bewegen, daher sollten Daten auch über mehrere Kilometer Entfernung stabil übertragen werden.

Eine Möglichkeit wäre, die Daten über das GSM-Netz zu übertragen. Im Gegensatz zu moderneren Mobilfunkstandards wie UMTS wird dieser durch seine weite Verbreitung im Industriebereich nicht abgebaut werden. Die niedrige Datenübertragungsrate ist kein Problem. Allerdings benötigen Module, die mit diesem Netz kommunizieren können, während des Sendevorgangs oft 2A Strom, was bei der Versorgung mit Batterien oder Akkus zum Problem wird. Zudem benötigt jedes dieser Module eine eigene SIM-Karte mit gültigem Tarif, was den Aufwand und die Nutzungskosten erhöht [13].

Eine im IOT-Bereich weit verbreitete alternative ist LoRa. Bekannt ist es Aufgrund der störsicheren Übertragung über hohe Reichweiten und dem geringen Stromverbrauch. Gesendet wird im für Short-Range Devices zugewiesenen Frequenzbereich von 868MHz. Dadurch fallen für das Senden keine weiteren Lizenzgebühren an. Die größte Einschränkung bei LoRa ist eine niedrige Datenrate, die durch den Duty-Cycle, einer gesetzlich vorgegebenen maximalen Sendezeit, noch weiter gesenkt wird.

Bei der Datenübertragung mit LoRa muss zwischen LoRa, der Funktechnik, und LoRaWAN, einem Netzwerkprotokoll, unterschieden werden. LoRaWAN basiert auf der LoRa Übertragung, bietet jedoch eine Netzwerkarchitektur mit Features für die Adressierung der Geräte und Verschlüsselung. In einem LoRaWAN-Netzwerk werden die Daten werden von einem Gateway empfangen und über das Internet an einen Server weitergeleitet [14].

Diese Gateways können entweder selbst errichtet werden, oder von einem Anbieter gegen Bezahlung verwendet werden. Im Gebiet der Vierkaseralm gibt es noch keine öffentlichen LoRa Gateways, es müsste also ein eigenes Gateway errichtet werden.

Aufgrund der erhöhten Abstraktion von LoRaWAN müssen mehr Regeln zur Paketstruktur und zu den Einstellungen der Modulation eingehalten werden. Dadurch erhöht sich die Datenmenge und somit auch die benötigte Leistung.

Da die Funktionen von LoRaWAN nicht benötigt werden, keine Gateways in der Nähe der Vierkaseralm sind und der Energieverbrauch am Tracker steigen würde, wurde auf LoRaWAN verzichtet und eine eigene Basisstation mit der LoRa Funkübertragungstechnik entwickelt.

#### 4.2.2 Rechtliche Rahmenbedingungen

Für die Datenübertragung mit Funk benötigt man eine Lizenz für den Frequenzbereich, in dem die Übertragung stattfindet. Für private Anwendungen wie WLAN, Alarmanlagen oder andere Geräte mit Funkübertragung wurden Bereiche für short-range devices, kurz SRD, freigehalten. Diese können von jedem lizenziert frei benutzt werden. LoRa verwendet das in Europa für SRD freigehaltene Frequenzband von 868MHz – 868,6MHz.

Um die Strahlungsbelastung gering zu halten und allen Funkteilnehmern die Kommunikation zu ermöglichen, wurde die Nutzung des Frequenzbereichs gesetzlich reguliert. In Österreich gelten die in der EU-Kommission beschlossenen Regeln. Diese sind in der „Entscheidung der Kommission vom 9. November 2006 zur Harmonisierung der Frequenznutzung durch Geräte mit geringer Reichweite“ festgehalten. In dieser findet man unter dem Punkt „Funkgeräte mit geringer Reichweite für nicht näher spezifizierte Anwendungen“ die für die EU gültigen Regulierungen.

Es darf maximal mit einer Leistung von 25mW ERP für eine Dauer von 1% einer Stunde gesendet werden. Zusätzlich darf man in diesem Bereich keine Videoanwendungen betreiben.

#### 4.2.3 Modulation von LoRa-Signalen

LoRa verwendet eine Chirp-Spread-Spectrum Modulation. Grundlage dafür ist ein Sinusförmiges Signal, dessen Frequenz sich linear ändert. Dieses Signal wird Chirp genannt. Abhängig von steigender oder fallender Frequenz wird in Up- und Down-Chirp unterschieden.

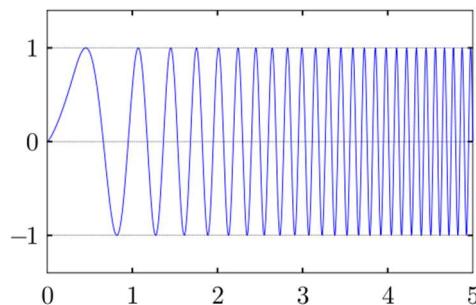


Abbildung 27: Chirp Signal im Zeitbereich [15]

Dieser Chirp kann mit dem Spreadingfactor, kurz SF, und der Bandbreite beeinflusst werden. Mit dem Spreadingfactor kann berechnet werden, über wie viele Samples sich ein Chirp streckt. Die Bandbreite bestimmt den Frequenzbereich, in dem sich der Chirp bewegt. Je länger ein Chirp ist, desto besser lässt er sich auf der Empfängerseite auswerten. Mit einem hohen Spreadingfactor erhält man somit eine gute Reichweite, muss aber eine langsamere Datenübertragung in Kauf nehmen.

Der Chirp bewegt sich zwischen  $-\frac{BW}{2}$  und  $\frac{BW}{2}$ . Die Information wird mit einem Offset im Frequenzgang übertragen. Der Chirp beginnt bei der Offsetfrequenz. Erreicht der Chirp die maximale Frequenz springt er zu  $-\frac{BW}{2}$ . Er endet wieder, sobald er die Offsetfrequenz erreicht hat. Durch die ständige Frequenzänderung sind Überlagerungen mit anderen Funkteilnehmern immer nur für einen kurzen Zeitraum. Das Signal ist daher sehr störsicher.

Die kleinste, nicht mehr teilbare, übertragbare Einheit nennt man ein Symbol. Ein Chirp überträgt jeweils ein Symbol. Die Zustände, die ein Symbol annehmen kann, sind aus der Menge  $[0, 2^{SF} - 1]$ . Nimmt man ein Symbol S aus dieser Menge kann mit der Formel 4 der Frequenzoffset für dieses Symbols berechnet werden.

$$f_{offset} = \frac{BW}{2^{SF}} * S \quad (4)$$

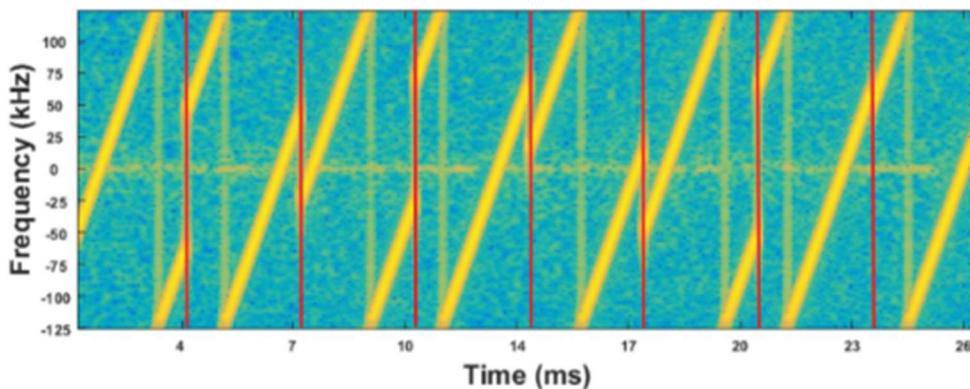


Abbildung 28. Frequenzverlauf eines LoRa-Signals [16]

Um das Signal zu demodulieren, werden die einzelnen Symbole mit einem in die entgegengesetzte Richtung verlaufenden Chirp ohne Frequenzsprung multipliziert. Dadurch verschwinden die Trägerfrequenzen aus dem Signal. Im verbleibenden Signal sind alle Frequenzen konstant. Dieses Signal wird mit der diskreten Fouriertransformation analysiert. Die stärkste auftretende Frequenz ist die Offsetfrequenz. Aus dieser wird das übertragene Symbol berechnet [17] [16].

Im Projekt wurden während des Entwicklungsprozesses ein Spreadingfactor von 6 und eine Bandbreite von 125kHz verwendet. Dadurch war es möglich, den Standort alle 20 Sekunden zu übertragen, was die Entwicklung vereinfachte. Bei dem Versuch die Reichweite zu erhöhen, wurde ab einem Spreadingfactor von 9 die Ungenauigkeit des RC-Oszillators bemerkbar. Durch die längere Übertragungsdauer bei einem höheren Spreadingfactor summiert sich der Fehler auf. Die einzelnen übertragenen Bit wurden dadurch nach unten verschoben.

#### 4.2.4 Codierung von LoRa

Der für die Übertragung wichtigste Punkt ist eine geeignete Codierung. Dadurch können Fehler erkannt und korrigiert werden, was die Reichweite erhöht [17].

Semtech gibt nur wenig Details über den Codierungsprozess bekannt. Allerdings ist es Forschenden durch Analyse der Signale gelungen, die für die Codierung verwendeten Schritte zu rekonstruieren. Um die Fehler zu minimieren und auszugleichen werden mit LoRa übermittelte Daten vor der Übertragung in 4 Schritten codiert [17] [16].

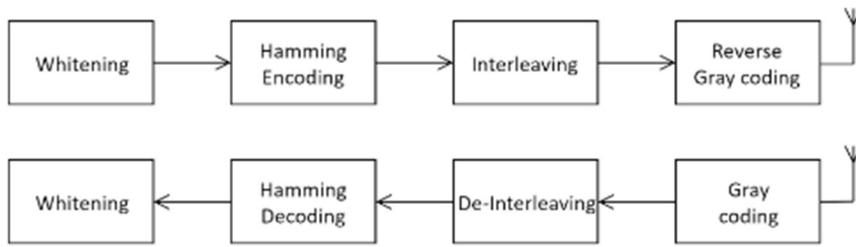


Abbildung 29: Encoding/Decoding von LoRa [16]

#### 4.2.4.1 Whitening

Bei Modulation sollten alle möglichen Symbole vertreten sein. Dadurch sind die übertragenen Symbole im Demodulationsprozess besser unterscheidbar [17].

Um möglichst viele unterschiedliche Symbole zu versenden, werden diese mit einer Pseudozufallssequenz logisch verknüpft. Die Zufallssequenz kommt aus einem Linear Feedback Shift Register, kurz LFRS. Um die Sequenz zu erzeugen, werden die Daten durch das Register geschoben. Die Ausgangssequenz des Registers wird wieder mit den Eingängen von den einzelnen Speicherzellen und den Daten verknüpft. Dadurch entsteht am Ausgang eine lineare Funktion, die abhängig von der Eingangskombination ist und im Empfänger rekonstruiert werden kann [17] [18].

#### 4.2.4.2 Hamming Code

Bei der Funkübertragung können durch Rauschen Fehler auftreten. Um die Fehler zu erkennen, arbeitet LoRa mit dem Hamming Code. Dafür müssen zusätzliche Bits eingebaut werden.

Diese Bits ergänzen eine Reihe von Datenbits, sodass diese eine gerade Anzahl an logischen-1 Bits beinhalten. Da bekannt ist, welche Bits zusammen eine gerade Anzahl ergeben müssen kann man ermitteln, ob ein Fehler entstanden ist. Wird mehr als eines dieser Bits übertragen, werden die Daten in Muster, die sich stellenweise überschneiden, aufgeteilt. Entsteht ein Fehler kann mit den einzelnen Bits, die gerade sein müssten, der Schnittpunkt ermittelt und der Fehler korrigiert werden.

Wie viele dieser zusätzlichen Bits pro Bit von Nutzdaten übertragen werden ist in der Coding Rate festgelegt. Die Coding Rate wird als Verhältnis von 4 Datenbit zu 5-8 übertragenen Bit angegeben. Eine Coding Rate von 4/5 schützt die Daten nicht und kann nur einen Fehler pro Bitgruppe erkennen. Ab 4/7 können mehrere Fehler erkannt und einzelne Fehler behoben werden [17].

#### 4.2.4.3 Interleaving

Während der Übertragung kann das Signal durch Rauschen beschädigt werden. Wird ein Symbol bei der Übertragung geschädigt, ist eine Reihe an Bits falsch.

Fehlerkorrekturcodes wie der Hamming-Code sind darauf ausgelegt, zufällige Fehler wieder zu richten. Das funktioniert nur, wenn einzelne Bits falsch sind. Ist eine Reihe an Bits falsch, können die Daten nicht korrigiert werden. Durch Interleaving wird der Fehler auf mehrere Symbole verteilt. LoRa verwendet dafür diagonal Interleaving. Dazu nimmt man, wie in Abbildung 30 erkennbar, aus jedem zu übertragendem Symbol ein Bit und wechselt nach jedem Symbol ein Bit weiter in Richtung des niedrigenwertigsten Bits [17].

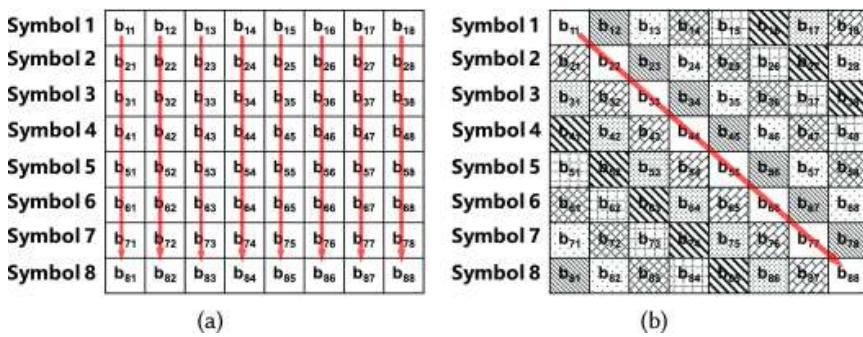


Abbildung 30: Interleaving (vgl. [17])

#### 4.2.4.4 Gray Code

Ein Gray-Code weist numerischen Symbolen eine Bitfolge zu. Das Besondere am Gray-Code ist, dass sich zwischen zwei aufeinander folgenden Symbolen nur in einem Bit unterscheiden. Dadurch lassen sich falsch übertragene Daten leicht erkennen.

In den meisten LoRa-Modems sind ungenaue RC-Oszillatoren verbaut. Beim Abtasten kann es passieren, dass Teile vom falschen Symbol abgetastet werden. Mit der Gray-Codierung wird der dabei entstehende Fehler besser erkennbar. Dadurch ist es wahrscheinlich, dass der Fehler korrigiert werden kann [17].

#### 4.2.4.5 Paketierung

Um den Empfänger zu synchronisieren und die Daten sicher zu übertragen, versendet LoRa versendet die Daten in einer Paketstruktur. Der Aufbau eines Paketes ist in Abbildung 31 zu sehen.

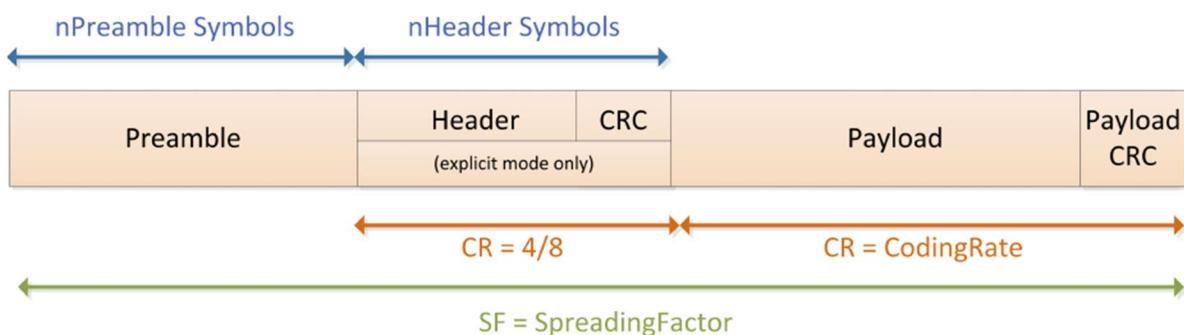


Figure 6. LoRa® Packet Structure

Abbildung 31: LoRa Paket [19]

Mithilfe des Preambles wird der Empfänger synchronisiert. Der Preamble ist standardmäßig 12 Symbole lang und kann nach Bedarf verkürzt oder verlängert werden. Der Header enthält Informationen über die Länge vom Payload, Informationen für die Korrektur von Fehlern und eine CRC-Prüfsumme, mit der die Informationen auf ihre Richtigkeit überprüft werden. Der Header selbst wird immer mit der höchsten Coding Rate übertragen, damit die Informationen möglichst sicher ankommen.

Im Payload befinden sich die Daten, die übertragen werden. Abgeschlossen wird das Paket mit einer Prüfsumme der im Payload übermittelten Daten. Die Coding Rate für diesen Teil kann vom Nutzer je nach Bedarf für hohe Reichweite oder hohe Übertragungsraten festgelegt werden [19].

## 4.2.5 Antennen

Für die Funkübertragung mit LoRa benötigt man Antennen. LoRa nutzt in Europa Frequenzen um 868MHz. Die Antennen müssen für diesen Frequenzbereich ausgelegt sein.

Bei der Übertragung mit hohen Frequenzen treten auf Leitungen Reflexionen auf, was die Übertragungsqualität verschlechtert. Um dem entgegenzuwirken, muss die Leitung mit dem Leitungswiderstand abgeschlossen sein. Dadurch verhält sich die Leitung, wie eine unendlich lange Leitung mit einem Eingangswiderstand gleich dem Wellenwiderstand [20].

Sowohl das TTGO T-Beam, welches als Basisstation verwendet wird, als auch das RFM95W, welches auf der Sendeeinheit verbaute ist, haben einen Wellenwiderstand von  $50\Omega$ . Deshalb ist es wichtig, dass die verwendeten Antennen einen Eingangswiderstand von  $50\Omega$  haben.

Antennen können nach ihrem Strahlverhalten eingeteilt werden. Die Stärke wird mit dem Isotopen Strahler verglichen. Der Isotope Strahler ist eine nicht realisierbare Antenne, welche die Energie in alle Richtungen gleichmäßig abstrahlt. Die Verstärkung wird in alle Richtungen gemessen, verglichen und in einem Richtdiagramm aufgetragen [21].

### 4.2.5.1 Antenne der Sendeeinheit

Für die Sendeeinheit wurde die Molex ISM 868/915MHz Dipol Antenne gewählt. Die Antenne hat ein gutes Rundstrahlverhalten. Durch die flexible Bauform lässt sie sich leicht in das Gehäuse integrieren. Durch den Wellenwiderstand von  $50\Omega$  kann die Antenne verwendet werden. [22]

Der Dipol-Teil ist in einem Klebestreifen verbaut und wird an die Gehäusewand geklebt. Über den UFL-Anschluss wird die Antenne mit dem LoRa Breakout verbunden.

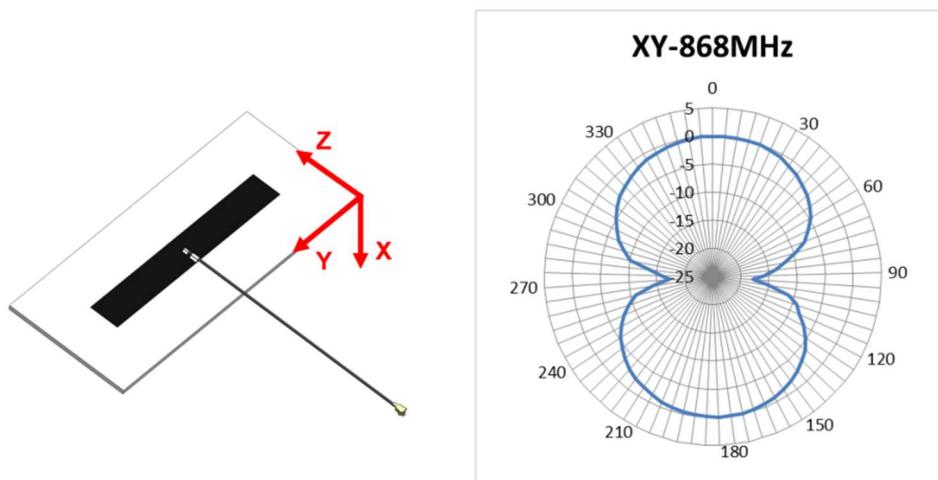


Abbildung 32: Richtdiagramm Molex Dipol [22]

#### **4.2.5.2 Antenne der Basisstation**

Als Antenne für die Basisstation wurde eine Dipolantenne des Herstellers LPRS verwendet. Die Antenne hat ein gutes Rundstrahlverhalten und empfängt Daten aus allen Richtungen ähnlich stark, wie aus dem Richtdiagramm in Abbildung 33 hervorgeht. Die Antenne wurde speziell für den Outdoor-Einsatz entwickelt und wird mit einer Wandhalterung an der Außenseite der Alm befestigt.

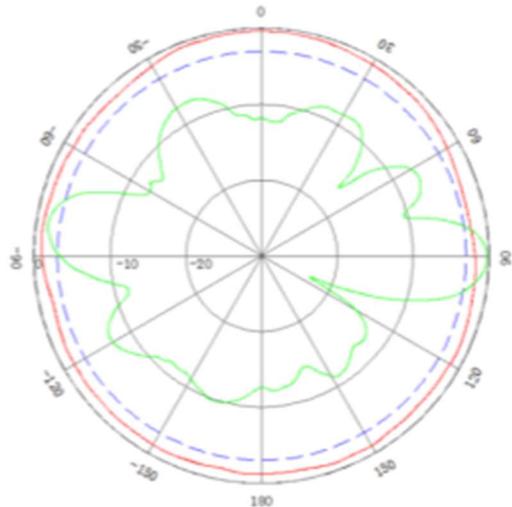


Abbildung 33: Richtdiagramm Antenne Basisstation [23]

#### 4.2.6 SX1276 LoRa-Transceiver

Auf der Sendeeinheit und der Basisstation ist jeweils ein SX1276 LoRa Transceiver verbaut. Bei der Modulation mit LoRa können Signale mit einer Stärke bis zu -148dBm empfangen und ausgewertet werden. Die Kommunikation erfolgt mit einem SPI-Interface. [19]

#### 4.2.6.1 Kommunikation mit dem Transceiver

Mit dem SPI-Interface können einzelne Register beschrieben und gelesen werden. Für die Übertragung wird immer ein Byte mit dem ersten Bit für write-not-read und 7 Adressbits, gefolgt von einem Byte Daten übertragen. Schreiben ist nur zulässig, wenn das Modul im Stand-by-Modus ist. Um den Transceiver in den Stand-by-Modus zu versetzen muss ein Byte mit dem Wert 0 in das Operation Mode Register geschrieben werden. Die zu sendenden und empfangenen Daten werden in einem 256 Byte großem FiFo-Speicher abgelegt. Für Senden und Empfangen gibt es in diesem Speicher zwei frei bewegliche Pointer [19].

#### 4.2.6.2 Übertragung der Daten

Nachdem die Frequenz, die Bandbreite, die Coding Rate und der Spreadingfactor in die jeweiligen Register geschrieben wurden kann mit der Datenübertragung begonnen werden. Ist das Modul im Stand-By-Modus, können die Dateien in den FiFo-Speicher geschrieben werden. Der FiFo-Pointer wird dabei automatisch erhöht. Um die Übertragung zu starten, muss im Modus-Register der Transmit Mode und der Long Range Modus aktiviert werden. Nach dem Versenden der Daten wird ein Interrupt ausgelöst und das Modul wechselt zurück in den Stand-By Modus [19].

## 4.3 Basisstation

Um die Daten auf dem Server zu speichern, müssen diese wieder empfangen und anschließend mit einer http Anfrage weitergeleitet werden. Dafür benötigt die Basisstation eine stabile Anbindung an das Internet und ein LoRa-Receiver Modul.

Aufgestellt wird das Modul auf der Vierkaseralm. Die Verbindung zum Internet kann dort mit WLAN hergestellt werden. Versorgt wird die Basisstation mit einem Micro-USB Kabel, angesteckt an einem Netzteil. Die Antenne wird auf der Außenseite der Alm mit einer Wandhalterung befestigt.

### 4.3.1 TTGO T-Beam

Für die Realisierung der Basisstation wurde die TTGO T-Beam Entwicklungsplatine gewählt. Als Controller ist ESP32 verbaut, wodurch kein extra Modul für WiFi benötigt wird. Die für WLAN-Antennen sind bereits auf der Platine verbaut. Versorgt und programmiert kann das Board mit einem Micro-USB Kabel werden. Für die Kommunikation mit LoRa bietet das Board einen SX1276 LoRa Transceiver. Für die LoRa-Antenne ist ein SMA-Anschluss verbaut [24].

### 4.3.2 PlatformIO

Für die Basisstation wurde die Programmierumgebung PlatformIO gewählt. PlatformIO hat es sich zum Ziel gesetzt, die Entwicklung für Microcontroller möglichst komfortabel und plattformunabhängig zu gestalten. Die Basis ist ein Core, welcher Abhängigkeiten Compiler und das Projekt verwaltet. Um die Funktionen des Cores in einer Entwicklungsumgebung zu nutzen, werden die Funktionen mit einem Plugin in diese integriert. Dieses Plugin ist für viele gängige Entwicklungsumgebungen verfügbar. Aufgrund der weiten Verbreitung und des einfachen Installationsprozesses wurde Visual Studio Code als Editor gewählt.

PlatformIO baut auf existierende Frameworks, wie dem Arduino Framework oder dem ESP-IDF auf, ist aber mit Boards und Microcontrollern von mehreren Herstellern kompatibel. Dadurch erhält man die Möglichkeit, viele Bibliotheken in sein Projekt zu integrieren, ohne von der Hardware der Hersteller und deren Tools Abhängig zu sein [25].

Für die Kommunikation mit dem LoRa-Chip wurde die LoRa Bibliothek von Sandeepmistry verwendet. Diese ist mit den Chips der Semtech SX127X Familie und mit einigen Arduino Boards kompatibel und erleichtert die Kommunikation über LoRa. Zusätzlich diente der Code dieser als Hilfestellung bei der Programmierung der Sendeeinheit [26].

Die empfangenen Daten werden mithilfe der ArduinoJSON Bibliothek in ein JSON-Objekt umgewandelt und mit dem für den ESP32 mitgelieferten HTTP-Client an den Server übermittelt.

### 4.3.3 Programmablauf der Basisstation

Die Basisstation empfängt die mit LoRa übermittelten Daten, wandelt diese in ein JSON-Objekt um und sendet diese über das Internet an den Server. Zu Beginn des Programms die Pins für die Module definiert und alle Bibliotheken initialisiert. Das Hauptprogramm besteht aus einer Schleife, die wartet, bis Daten mit LoRa empfangen wurden. Werden Daten empfangen, werden diese zusammen mit der Empfangssstärke, kurz RSSI, auf der Konsole ausgegeben.

Mit der Funktion `scanf` werden die Daten aus dem empfangenen String wieder ausgelesen und ein struct geschrieben. Die `sscanf`-Funktion aus der Bibliothek des Compilers unterstützt keine Floating Point Zahlen. Deshalb wurden der Längen- und Breitengrad wurde als 32-Bit Integer übertragen. Um die ursprünglichen Werte zu erhalten, muss man die beiden Variablen durch 100.000 dividieren.

```
sscanf(rec.c_str(), "%d %d %d %d %d %d %" PRIId32 " %" PRIId32,
       &position->id,
       &position->hour,
       &position->minutes,
       &position->seconds,
       &position->day,
       &position->month,
       &position->year,
       lat,
       lng
     )
position->lat = (float)lat / 100000;
position->lng = (float)lng / 100000;
```

Das Datum wird aus den einzelnen Elementen zusammengesetzt und die Daten werden in eine Instanz der `JsonDocument`-Klasse mit dem Namen `data` geschrieben.

```
data["latitude"] = position->lat / 100000;
data["longitude"] = position->lng / 100000;

String date = "20" + String(position->year)
    + "/" + String(position->month)
    + "/" + String(position->day)
    + " " + String(position->hour)
    + ":" + String(position->minutes)
    + ":" + String(position->seconds);

data["date"] = date;
```

Das Objekt wird auf der Konsole ausgegeben, in einen String konvertiert und mit einer POST-Anfrage an den Server übermittelt.

```
String POST_DATA;
serializeJson(data, POST_DATA);

if (http.begin(url)) {
    http.addHeader("Content-Type", "application/json");

    int httpResponseCode = http.POST(POST_DATA);

    if (httpResponseCode > 0) {
        Serial.print("HTTP Response code: ");
        Serial.println(httpResponseCode);

        String response = http.getString();
        Serial.println(response);
    } else {
        Serial.print("Error in HTTP request. Fehlercode: ");
        Serial.println(httpResponseCode);
    }
    http.end();
}
```

In Abbildung 34 sind die einzelnen Schritte und Ausgaben aus dem Code auf dem Seriellen Monitor nach einem erfolgreichen Durchlauf erkennbar. In der ersten Zeile sieht man den übertragenen String mit der Restleistung des empfangenen Datenpakets (RSSI) in dBm. Darunter sieht man das erstellte JSON-Objekt. Abschließend wird der Statuscode der HTTP-Übertragung ausgegeben. Der Code 200 bedeutet, dass die Übertragung erfolgreich war.

```
01 08 36 27 16 03 24 4782365 1304247 with RSSI -54

{
  "latitude": 47.82365,
  "longitude": 13.04247,
  "date": "2024/03/16 08:36:27"
}

HTTP Response code: 200
```

Abbildung 34: Serieller Monitor Basisstation

## 4.4 Server

Die Standortdaten sollten jederzeit und ortsunabhängig abrufbar sein. Um das zu ermöglichen, benötigt es eine Serversoftware, welche die Daten über das Internet zur Verfügung stellt. Gespeichert werden die Standorte in einer Datenbank auf diesen Server.

Um den Server möglichst einfach zu entwickeln wurde das Laravel-Framework verwendet. Dieses bietet bereits fertige, einfach in den Code einbaubare Module für komplexe Features wie einen Login oder die Kommunikation mit Datenbanken. Da das Framework Open-Source ist und eine Community an dem Framework arbeitet, ist der Code oft fehlerfreier und sicherer als eine komplette Eigenentwicklung.

### 4.4.1 Laravel

Laravel ist ein von einer Community entwickeltes und von Firmen unterstütztes open-Source PHP-Framework. Es erleichtert die Entwicklung von Webanwendungen und APIs. Für das Framework gibt es viele, fertige Bibliotheken um verschiedenste Features wie ein Bezahlsystem oder einen Login in seine Anwendung einzubauen [27].

Laravel arbeitet mit dem Model-View-Control Muster. Dieses Muster unterteilt die Software in einen Modell Teil, in welchem ein Modell von den Daten definiert wird, einen View Teil, welcher sich um die Visualisierung und Interaktion mit dem Nutzer kümmert und einen Controller, welcher die Modelle auf Anfragen erstellt, ändert oder ausliest.

Modelle sind eine PHP-Klasse, die gespeichert, abgerufen und geändert werden kann. Durch diese wird in dem Framework mit Datenbanken kommuniziert. Der Controller Teil wird verwendet, um diese zu erstellen und zu verändern. Der View-Teil ist bei diesem Projekt die Smartphone-Applikation [28].

#### 4.4.1.1 Eloquent ORM

Ein Object-relational mapper vereinfacht die Kommunikation mit Datenbanken. Für jedes Modell wird eine Tabelle in einer relationalen Datenbank erstellt. Alle Modelle erben von der Model Klasse aus dem ORM. Dadurch stellt jedes Modell bereits Funktionen für die Speicherung, Änderung und Abfrage aus einer Datenbank zur Verfügung.

Mit den Funktionen findAll und find können die Modelle aus der Datenbank zurück in eine Instanz der Klasse geladen werden.

Um die Verbindung zwischen Modellen zu realisieren können die Funktionen „has“, „hasMany“ und „belongsTo“ genutzt werden. Dadurch werden die Daten in der Datenbank miteinander verknüpft und bei Änderung automatisch in allen Tabellen geändert [29].

#### 4.4.1.2 Sanctum Auth

Die Sanctum Bibliothek von Laravel vereinfacht das Erstellen von Projekten mit Nutzern und Nutzerverwaltung. Funktionen, um Nutzer zu erstellen, sich zu registrieren und zu authentifizieren werden bereitgestellt.

Sanctum arbeitet mit Tokens. Diese werden nach der Authentifizierung mittels E-Mail und Passwort an die App des Nutzers übermittelt. Diese schickt den Token bei jeder Anfrage an den Server im Header mit. Dadurch kann die Identität des Nutzers ermittelt und der Zugriff auf die Daten beschränkt werden [30].

#### 4.4.1.3 Modelle und Datenbank

Aufgrund der Verwendung des ORMs ist die Wahl der Datenbank eingeschränkt. Der Laravel ORM ist mit dem Microsoft SQL Server, MariaDB, MySQL, PostgreSQL und SQLite kompatibel. Aufgrund dieser Abstrahierung benötigt ein Wechsel zwischen den Datenbanken außerhalb der Konfiguration keine weiteren Anpassungen.

Für das Projekt wurde MariaDB als Datenbank verwendet. MariaDB ist eine quelloffene Datenbank, die auf Basis des MySQL SQL-Servers entwickelt wurde. Die Community, die den Datenbankserver entwickelt legt großen Wert auf Effizienz und durch die kostenlose Nutzung gibt es eine Vielzahl an Tutorials.

Wird aus den Modellen eine Tabelle erstellt nennt man das eine „migration“. Die an der Datenbank durchgeführten migrations werden zusammen mit der Anzahl wie oft die Tabelle verändert wurde in der migrations Tabelle gespeichert. Schlägt eine migration fehl, wird das in der failed\_jobs Tabelle festgehalten [29].

Das Modell für den User wurde von Sanctum übernommen. Bei der Registrierung muss der Nutzer seinen Namen, ein Passwort und eine E-Mail-Adresse angeben. Für den Nutzer wird ein Account angelegt, mit welchem er sich anmelden, Tracker erstellen und Trackerdaten abrufen kann. In der personal\_access\_tokens Datenbank werden die Tokens, mit denen der Nutzer seine Identität bestätigen kann, gespeichert.

Der Name der Kuh soll in der App angezeigt werden. Dafür muss den Trackern ein Namensfeld zugewiesen werden. Ist das Erstellen erfolgreich, bekommt der Nutzer eine ID, die er auf der Sendeeinheit speichern muss. Der Nutzer, der den Tracker erstellt hat, wird aus dem Token in der Anfrage ausgewertet und in den Tracker-Tabelle gespeichert. Dadurch hat nur dieser Nutzer Zugriff auf den Tracker.

Die Positionen, die vom Tracker an den Server geschickt werden, kommen in das Position Modell. Dieses bietet ein Feld für Längengrad, Breitengrad, den Zeitpunkt, an dem der Standort bestimmt wurde, sowie die ID des Trackers. Dadurch lässt sich bestimmen, zu welcher Kuh der Standort gehört.

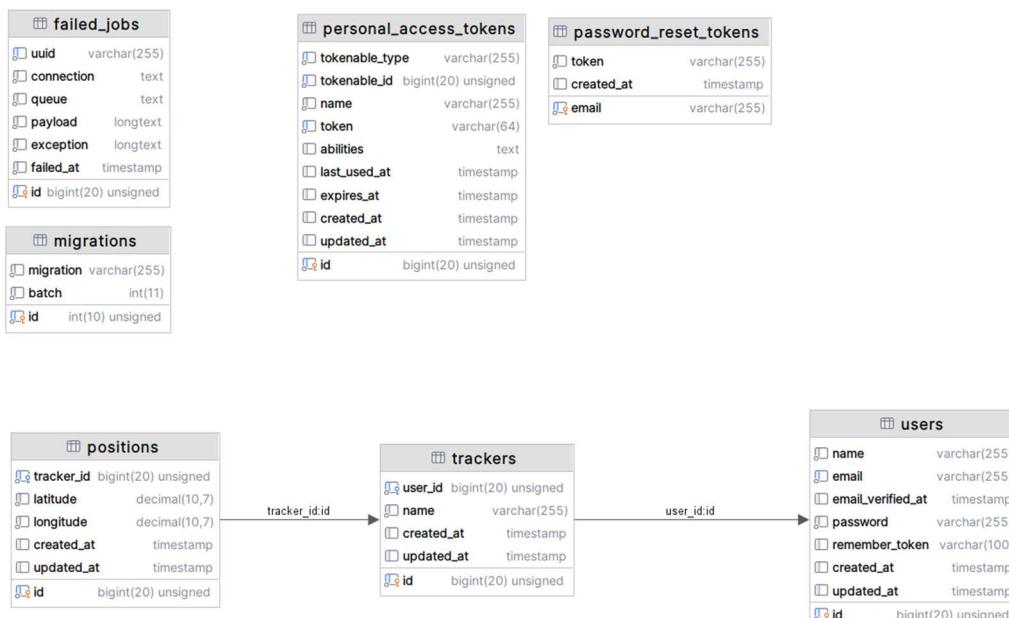


Abbildung 35: Datenbankstruktur

Aus den beschriebenen Modellen entstand die Datenbankstruktur in Abbildung 35. Die Modelle wurden mit dem mitgelieferten Tool artisan erstellt. Ein Modell kann mit dem Befehl `php artisan make:model Tracker -m` erstellt werden.

Der Befehl erstellt das benötigte File für das Modell und das File für die Datenbankmigration. Das File für die Datenbankmigration findet man unter dem Pfad `database/migrations/2024_01_07_110648_create_trackers_table.php`. Dort müssen in der „create“ Funktion die benötigten Variablen angelegt werden.

```
Schema::create('trackers', function (Blueprint $table) {
    $table->id();
    $table->unsignedBigInteger('user_id');
    $table->string('name');
    $table->foreign('user_id')->references('id')->on('users');
    $table->timestamps();
});
```

Das Modell findet man unter App/Models/Tracker.php. In diesem werden die Variablen aus der Datenbank erstellt und die Beziehungen zu anderen Modellen definiert.

```
class Tracker extends Model
{
    use HasFactory;

    protected $fillable = ['user_id', 'name'];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function positions()
    {
        return $this->hasMany(Position::class);
    }
}
```

#### 4.4.1.4 Controller

Ein Controller ist für die Verarbeitung von HTTP-Abfragen verantwortlich. Die Controller sind verantwortlich für den Datenfluss. In ihnen werden Instanzen der Modelle erstellt, bearbeitet, ausgelesen oder gelöscht. Die einzelnen Controller werden im Pfad `app/Http/Controllers` gespeichert.

Um die Tracker zu verwalten, wurde der TrackerController erstellt. Dieser kann Tracker erstellen und auflisten und Positionen zu einem Tracker hinzufügen und wieder abrufen. Bei Abfrage wird mit der `user_id` sichergestellt, dass der Nutzer nur Daten von seinen eigenen Sendeeinheiten bekommt.

```

class TrackerController extends Controller
{
    ...
    public function addPosition(Request $request, $trackerId)
    {
        // Validate the incoming request data
        $request->validate([
            'latitude' => 'required',
            'longitude' => 'required',
            'date' => 'required|date_format:Y-m-d H:i:s'
        ]);

        // Find the tracker
        $tracker = Tracker::findOrFail($trackerId);

        // Add a new position to the tracker
        $position = $tracker->positions()->create([
            'latitude' => $request->latitude,
            'longitude' => $request->longitude,
            'date'=> $request->date,
        ]);

        return response()->json(['position' => $position], 201);
    }

    // get last Position of tracker
    public function getLastPositions(Request $request)
    {
        $trackers = Tracker::where('user_id', $request->user()->id)->get()->toArray();
        $positions = Position::where('tracker_id', array_column($trackers, 'id'))->get()->last();
        return response()->json($positions);
    }
    ...
}

```

Die Funktion bekommt die Anfragedetails und die ID des Trackers als Parameter. Mit der `findOrFail` Funktion des ORMs wird nach dem Tracker gesucht. Existiert der Tracker wird überprüft, ob der Nutzer, der die Anfrage gemacht hat, den Tracker erstellt hat und somit auf die Daten zugreifen darf. Ist das der Fall, wird die Position des Trackers ausgelesen und der letzte Standort im JSON-Format zurückgegeben.

#### 4.4.1.5 Routing der Funktionen

Der Router legt bestimmt, welche Controller (vgl. 4.4.1.4) die Anfrage verarbeiten. Dazu muss in einer der Dateien im `routes` Ordner die Funktion eines Controllers mit einer Adresse verknüpft werden.

Die Anfragen können im Router mit einer sogenannten Middleware gefiltert werden. Sanctum bietet eine Middleware, die überprüft, ob der Nutzer angemeldet ist. Ist der Nutzer nicht angemeldet bekommt er den http-Statuscode 401 zurück. Ist er angemeldet und sendet einen gültigen Token lässt die Middleware die Kommunikation zu und gibt der aufgerufenen Funktion den Nutzer, der die Anfrage erstellt hat, mit. [31]

Die Route, an die die Basisstation die Positionen der Kühe sendet, ist frei zugänglich und die empfangenen Daten werden nicht auf ihre Authentizität geprüft. Da Daten an die Route gesendet werden ist es eine POST-Anfrage. Die trackerID muss beim Aufrufen der Basisstation durch die ID des Trackers der Kuh ersetzt werden.

```
Route::post('/trackers/{trackerId}/positions', [TrackerController::class,
    'addPosition']);
```

Die Routen, an denen ein Nutzer Tracker Daten von den Trackern aufruft, werden von der Sanctum Middleware kontrolliert. An dieser Stelle bekommt der Nutzer Daten, daher ist es eine GET-Anfrage.

```
Route::middleware('auth:sanctum')->get('/trackers/{trackerId}/position',
[TrackerController::class, 'getPosition']);
```

Aus Abbildung 36 geht hervor, welche Routen mit welchen Funktionen der Controller verknüpft wurden. Routen, die mit POST-Anfragen angesprochen werden, sind Rot hinterlegt, Routen für GET-Anfragen in Blau.

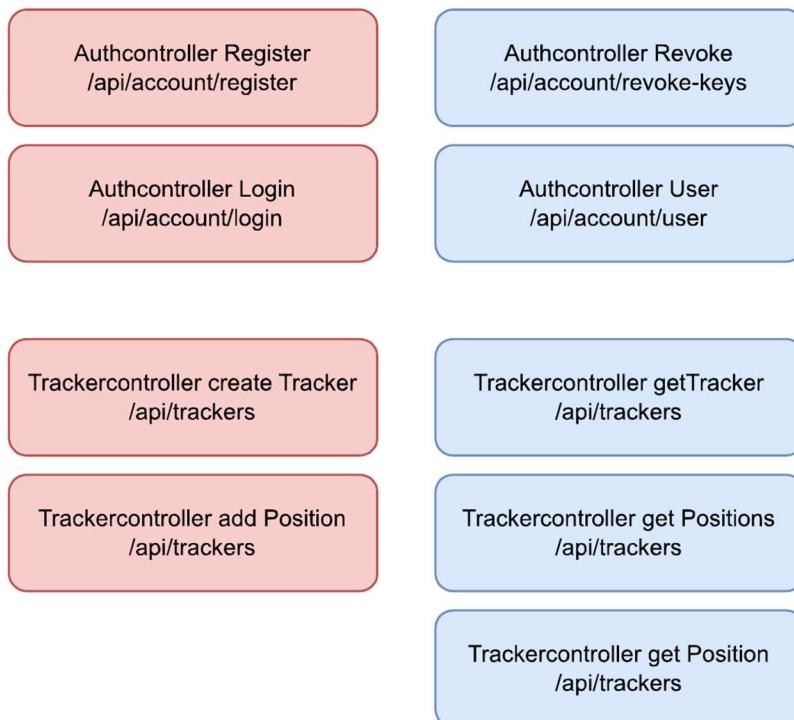


Abbildung 36: Überblick über Routen des Servers

## 4.5 Smartphone Applikation

Nachdem die GPS-Daten in der Datenbank auf dem Server gespeichert wurden, ist eine visuelle Darstellung der Daten als letzter Schritt erforderlich. Wir haben uns dazu entschieden, sowohl eine Handy-Applikation als auch eine Website zu erstellen, die jederzeit mit einer stabilen Internetverbindung abgerufen werden können.

### 4.5.1 Anforderungen der Applikation

Die Hauptanforderung der Applikation ist, sehr simpel und übersichtlich gestaltet zu sein. Die GPS-Daten werden alle 5 Minuten aktualisiert und auf einer Karte dargestellt. Zudem soll die Applikation so programmiert werden, dass der Code leicht verständlich ist und Erweiterungen leicht möglich sind.

### 4.5.2 Flutter

Als Framework für die C-Track haben wir uns für Flutter entschieden. Flutter ist ein Entwicklungs-Kit, welches von Google im Jahr 2017 veröffentlicht wurde. Flutter benutzt die Programmiersprache Dart und es können sogenannte Cross-Plattform-Applikationen damit erstellt werden. Dies bedeutet ein Flutter Programm kann ohne aufwendige Optimierung auf mehreren Plattformen wie Android, IOS, etc, laufen. Gegenüber anderen Frameworks wie React Native, hat Flutter den Vorteil, dass sie eine leicht zu verstehende Plattform bietet, eine sehr hilfsbereite Community dahintersteht und viele Tutorials, welche den Einstieg enorm erleichtern. Weiters bietet Flutter eine angenehme Testumgebung, um die Applikationen zu überprüfen. Wird das Projekt durch die Einbindung von mehreren Widgets größer kann man leicht die Übersicht verlieren. Zudem ist Flutter im professionellen Umfeld eine wenig verbreitetes Framework und wird hauptsächlich von „Hobby Programmierer“ verwendet.

#### 4.5.2.1 Widgets in Flutter

In Flutter sind Widgets die grundlegenden Bausteine für den Aufbau von Benutzeroberflächen. Sie repräsentieren alles in der Applikation, sei es ein Struktur Element wie eine Zeile oder ein Spaltenlayout, oder ein visuelles Element wie Schaltflächen, Text oder Bilder. Widgets können Zustände halten, interaktiv sein und sich, basierend auf Eingaben des Benutzers, Änderung von Daten oder anderen Faktoren, ändern. Zudem kann ein Widget wiederum aus mehreren anderen Widgets bestehen. Grundsätzlich gibt es in Flutter zwei Hauptarten von Widgets, welche die Grundlage für das Erstellen einer Benutzeroberfläche bilden.

- Das Stateless Widget ist unveränderlich. Das heißt, es wird einmal eingestellt und kann nicht verändert werden. Benutzt wird dieses für statische Inhalte und nicht interaktive Elemente, wie Texte oder Icons.
- Das Statefull Widget kann im Gegensatz ihren Zustand ändern und somit dynamische Inhalte oder interaktive Elemente, wie Slider oder Buttons darstellen.

#### 4.5.2.2 Packages in Flutter

Packages in Flutter sind externe Bibliotheken welche Tools und Funktionen bereitstellen, um die Entwicklung von Apps zu erleichtern und die Entwicklungszeit wesentlich verkürzen. Diese Packages können von Google als auch von der Community erstellt werden [32].

Grundsätzlich werden alle Packages in der zentralen Quelle für Packages, *pub.dev*, verwaltet. Um ein Package überhaupt verwenden zu können muss dieses zuerst in der *pubspec.yaml* Datei unter den *dependencies* hinzugefügt werden.

```
dependencies:
  flutter:
    sdk: flutter
  flutter_map: ^6.1.0
  latlong2: ^0.9.0
  url_launcher: ^6.2.1
  flutter_launcher_icons: ^0.13.1
  flutter_svg: ^2.0.10+1
  http: ^1.2.1
```

Abbildung 37: *dependencies* in *pubspec.yaml*

In Abb. 37. sind die verwendeten Packages im vorliegenden Projekt zu sehen. Diese werden immer mit der im Projekt verwendeten Versionsnummer angezeigt.

Mit dem Befehl *flutter pub get* werden diese Packages heruntergeladen. Anschließend können diese im Projekt mithilfe der Import Funktion importiert werden, um die benötigten Funktionen darin verwenden zu können.

Sind Packages veraltet oder haben eine neuere Version, können diese mit dem Befehl *flutter pub upgrade* aktualisiert werden.

#### 4.5.2.3 Erstellen des App Icon

Am Beginn jeder in Flutter programmierten Applikation hat diese standartmäßig das Flutter Logo als App Icon. Um dies zu ändern, muss zuerst eine neue Datei mit dem Namen *flutter\_launcher\_icons.yaml* erstellt werden.

```
flutter_launcher_icons:
  android: "launcher_icon"
  ios: true
  image_path: "assets/icon.png"
  min_sdk_android: 21 # android min sdk min:16, default 21
  web:
    generate: true
    image_path: "assets/icon.png"
    background_color: "#hexcode"
    theme_color: "#hexcode"
  windows:
    generate: true
    image_path: "assets/icon.png"
    icon_size: 48 # min:48, max:256, default: 48
  macos:
    generate: true
    image_path: "assets/icon.png"
```

Anschließend wird der Code aus Abb. 38. hineinkopiert. Im Projekt Verzeichnis wird nun ein Ordner Namens *assets* angelegt in dem alle, für die Applikation benötigten, Dateien oder Bilder eingefügt werden. Als letztes wird nun das gewünschte Bild als *icon.png* gespeichert und in den *dependencies* der *pubspec.yaml* Datei die neuste Version von *flutter\_launcher\_icons* hinzugefügt.

Abbildung 38: Code für Launcher Icon

#### 4.5.2.4 Erstellen der App Bar

Zu Beginn wird die Leiste (AppBar) welche am obersten Rand jeder Flutter Applikation zu sehen ist angepasst und individualisiert.

```
    ...
debugShowCheckedModeBanner: false,
```

Abbildung 39: Code: Debug Banner deaktivieren

Durch der in Abb. 39 gezeigten Code Zeile lässt sich das Debug Banner, welches bei jeder Flutter Applikation im rechten oberen Eck der AppBar befindet, ausschalten.

```
title: const Text(
  'C-Track 2.0', //Text
  textDirection: TextDirection.ltr,

  style: TextStyle(
    fontSize: 30,
    fontWeight: FontWeight.bold, /
    fontStyle: FontStyle.italic, /

  ), // TextStyle
), // Text
```

In Abb 40 ist zu sehen, wie Text Widget erstellt wurde, welches den Namen des Projekts anzeigen lässt.

Wichtig ist die Funktion *textDirection*. Diese Funktion verhindert das beim Kompilieren des Programms die Fehlermeldung einer NullSafetyViolation aufkommt.

Abbildung 40:Code: Text in App Bar

Danach wurden noch die Ecken abgerundet, die Farbe geändert und die Schrift zentriert, um die AppBar anschaulicher zu gestalten. Abb 41 zeigt die fertig erstellte AppBar



Abbildung 41: Fertige AppBar

#### 4.5.2.5 Drop Down Menu

Als weiterer Schritt wurde ein Drop Down Menu in der AppBar erzeugt. In dieser befindet sich die Lizenz für die Benutzung von OpenStreetMap und eine kurze Übersicht über das Projekt.

Wird auf das Drop Down Menu Icon im linken Eck gedrückt, öffnet sich ein sogenanntes Drawer Widget. In diesem Widget befindet sich Knöpfe für die jeweiligen Optionen, sowie unser Projekt Plakat.

Das Projekt Plakat wurde im *DrawerHeader* als *image* eingefügt, um die grundsätzliche Ästhetik und Anschaulichkeit der App zu verbessern.

```
const DrawerHeader(
  decoration: BoxDecoration(
    image: DecorationImage(
      image: AssetImage("assets/ctrack.png"),
      fit: BoxFit.fill,
    ), // DecorationImage
  ), child: null, // BoxDecoration
), // DrawerHeader
```

Hierfür wurde das Plakat als png-Datei im assets Ordner gespeichert und mit dem DrawerHeader Widget im Drop Down Menu angezeigt. (siehe Abb. 42).

Abbildung 42: Code: Drop Down Menu Header

```
ListTile(
  leading: const Icon(Icons.account_circle_rounded),
  title: const Text('About Us'),
  onTap: () {
    Navigator.push(context, new MaterialPageRoute(builder: (context) => new AboutUsPage()));
  },
), // ListTile
```

Abbildung 43: Code: Button einfügen

Um einen Button einzufügen, wird ein ListTile Widget eingefügt, in welchen das Icon, welches dieser Button haben soll, ausgewählt wird. Danach wird dieser Button benannt und im Anschluss wird festgelegt, was dieser beim Drücken machen soll. In unserem Fall wechselt man von der *HomePage* in die sogenannte *AboutUsPage*. Möchte man weitere Buttons einfügen wird auf dieselbe Weise ein ListTile Widget erstellt und der Name, das Icon und die Funktion eingefügt.

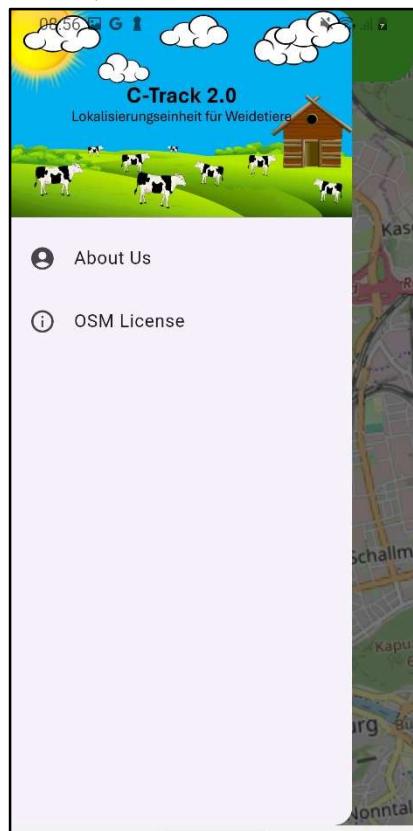


Abb. 44 zeigt das ausgefahrenes Drop Down Menu mit dem Projekt Plakat und den zwei Buttons. Dieses Menu kann nach Belieben um weitere Funktionen erweitert und verbessert werden.

Abbildung 44: Drop Down Menu

Klickt man nun auf einen der zwei Knöpfe wird man auf eine neue Seite weitergeleitet. Als einfachste Lösung den gewünschten Text darzustellen, entschieden wir uns den Text in ein normales txt-File zu kopieren und dieses in den assets Ordner abzulegen und mit folgender Funktion aufzurufen.

```
String data = '';

fetchFileData() async{
    String responseText;
    responseText = await rootBundle.loadString('assets/AboutUs.txt');

    setState((){
        data = responseText;
    });
}
```

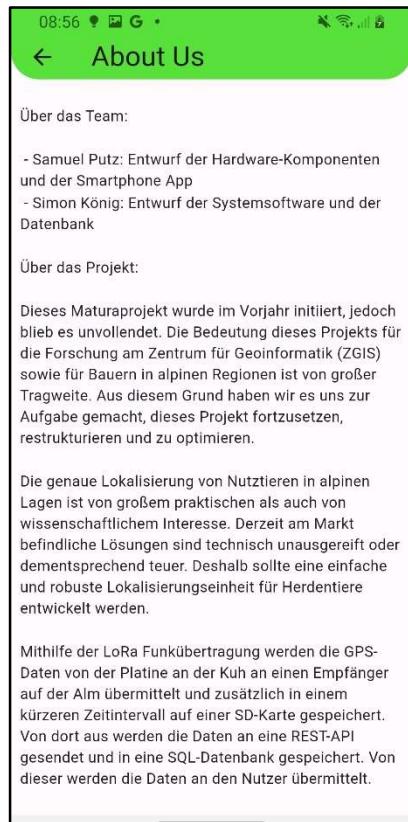
Abbildung 45: Code: txt-File einlesen

Zuerst wird ein String mit dem Namen *data* initialisiert. Nachfolgend wird in der *fetchFileData* Funktion der Inhalt des txt-Files herausgelesen und im Anschluss in den String kopiert.

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

Nun wird diese Funktion im *initState* aufgerufen, sodass der Text angezeigt wird, sobald man von der HomePage in die *AboutUsPage* wechselt.

Abbildung 46: Code: initState



Zuletzt müssen noch Anpassungen über die Größe des Textes getroffen werden, um den Text auf einer Seite anzeigen zu können. Möchte man wieder zurück zur *HomePage* muss man den Pfeil im linken oberen Eck drücken.

Abbildung 47: About Us Page

#### 4.5.2.6 Integration von Open Street Map

Am wichtigsten ist die Landkarte. Wir haben uns für Open Street Map entschieden, da diese frei zu benutzen ist und die Lizenz dafür lediglich in der App einsehbar sein muss. Die Lizenz ist in der Applikation im Drop Down Menu unter dem Punkt *OSM License* abrufbar. Zudem ist die Landkarte aktuell und teils genauer als Google Maps. Der Nachteil von Open Street Map ist, dass es nur mit einer stabilen Internetverbindung funktioniert.

Um Open Street Map in die App einbinden zu können muss Flutter Maps, sowie das *latlong2* Package importiert werden [33] [34].

```
return FlutterMap(
  options: const MapOptions(
    initialCenter: LatLng(47.820317, 13.043026),
    zoom: 11,
    interactionOptions: InteractionOptions(flags: ~InteractiveFlag.doubleTapZoom),
  ), // MapOptions
```

Abbildung 48: Code: Flutter Map

Zur Einbindung wird ein Widget für Flutter Map erstellt, in dem die grundsätzlichen Einstellungen der Karte vorgenommen werden. Diese sind etwa der Startpunkt, der angezeigt werden soll, wenn die Karte geöffnet wird oder den Startzoom.

```
TileLayer get openStreetMapTileLayer => TileLayer(
  urlTemplate: 'https://tile.openstreetmap.org/{z}/{x}/{y}.png',
  userAgentPackageName: 'dev.fleaflet.flutter_map.example',
);
```

Abbildung 49: Code: Tile Layer

Um die Karte von Open Street Map in der Applikation anzeigen zu lassen muss ein sogenanntes *TileLayer* Widget erstellt werden. Danach wird die URL von OpenStreetMap eingefügt, um die Karte in die Applikation zu implementieren.

```
MarkerLayer(markers: [
  Marker(
    point: markerPosition!,
    width: 50,
    height: 50,
    alignment: Alignment.centerLeft,
    child: SvgPicture.asset(
      'assets/cow.svg',
      semanticsLabel: 'Cow',
    ), // SvgPicture.asset
  ), // Marker
]), // MarkerLayer
```

Um die Position des Nutzertieres auf der Karte darstellen zu können ist ein *MarkerLayer* zu erstellen. Hier werden die Größe (width, height), sowie die Position des Markers eingestellt. Die Position bekommt der MarkerLayer über die Verbindung mit dem Server. Zuletzt wurde eine SVG-Datei einer Kuh aus dem assets Ordner als Icon des Markers verwendet.

Abbildung 50: Code: Marker Layer

#### 4.5.2.7 Server Anbindung der Applikation

Um die Standortdaten der Kuh abzubilden, muss sich die Applikation mit dem Server verbinden. Für diese Funktion wird das HTTP-Package benötigt [35].

```
Future<void> fetchMarkerPosition() async {
  String url = 'http://192.168.21.152:8000/api/trackers/1/positions';
  String token = 'Bearer 1|wwGyHyxh1Lx1RHhAuIIGDmGqjZ6saHcQdzjYF9cif7715c39';
```

Abbildung 51: Code: Serververbindung

Um die Verbindung mit dem Server zu erstellen, wird dessen URL hinterlegt, sowie ein Token, welcher als Account Identifizierung dient.

```
try {
  var response = await http.get(
    Uri.parse(url),
    headers: {'Authorization': token},
);
```

Über eine try/catch Funktion wird nun die Verbindung zum Server angefragt und überprüft.

Abbildung 52: Code: Serververbindung

```
if (response.statusCode == 200) {
  Map<String, dynamic> data = jsonDecode(response.body);
  double latitude = data['latitude'];
  double longitude = data['longitude'];
  setState(() {
    markerPosition = LatLng(latitude, longitude);
  });
};
```

Abbildung 53: Code: Marker Position

Ist die Verbindung erfolgreich empfängt die Applikation eine json-Datei die Zeit, Längengrad und Breitengrad enthält. Da diese Datei jedoch unbrauchbar für das Einfügen der Marker Position ist, wird diese json-Datei in einen String umgewandelt und daraus werden die Werte für Längen – und Breitengrad extrahiert und in die *markerPosition* Variable hineingeschrieben.

```
void initState() {
  super.initState();
  fetchMarkerPosition();
  timer = Timer.periodic(const Duration(minutes: 5), (timer) {
    fetchMarkerPosition();
  }); // Timer.periodic
}
```

Abbildung 54: Code: initState

Die *fetchMarkerPosition* Funktion wird nun im *initState* aufgerufen, um die Position der Kuh beim Start der App sofort anzuzeigen. Um alle 5 Minuten die Position aktualisieren zu können wurde ein Timer hinzugefügt der in 5 Minuten Abständen die Funktion erneut ausführt.

#### 4.5.2.8 Fertige Handy App



Abbildung 55: Startbildschirm der Smartphone Applikation

Die Fertige Handy App beinhaltet nun eine Landkarte von Open Street Map, welche beim Start der Applikation sich mit dem Server verbindet, um die Standordaten der Kuh zu erhalten. Danach wird mit einem Kuh Symbol der Standort graphisch angezeigt und alle 5 Minuten aktualisiert.

Im Drop Down Menu befindet sich die benötigte Lizenz von Open Street Map sowie eine kurze Erklärung zu unserem Projekt

In diesem Bild (Abb. 55) ist der Startbildschirm unserer Smartphone Applikation zu sehen welche einen Testwert welcher vom Server gesendet wurde auf einer Landkarte darstellt.

## 4.6 Verbesserungen

### 4.6.1 Platine

Der Trickle Power Mode eignet sich perfekt für das Projekt, konnte aber aufgrund der Komplexität von OSP-Sätzen nicht in das Projekt integriert werden.

Aus Stromspargründen wurde der interne RC-Schwingkreis des ATmega644 verwendet. Die Frequenz ist aber temperatur- und spannungsabhängig. Dadurch wird die Takt Frequenz, insbesondere bei niedrigen Temperaturen, zu ungenau für UART. Es ist zu empfehlen entweder einen Quarz einzubauen, oder die Kommunikation mit dem GPS-Modul auf eine andere Schnittstelle gelegt werden.

Zum Schutz der SD-Karte könnten die Daten in einen Buffer gespeichert werden und gesammelt, das heißt im Burst Mode, auf die SD-Karte geschrieben werden.  
(vgl. 4.1.5.1)

### 4.6.2 Server

Der Server prüft die Daten nicht auf Authentizität. Dadurch könnte ein Angreifer eigene Positionsdaten senden und speichern. Um dem Entgegenzuwirken könnten die Daten mit Kryptografie signiert werden.

### 4.6.3 Smartphone Applikation

In der Applikation wurden die Anmeldung des Servers, die Möglichkeit für mehrere Tracker und die Möglichkeit den Zeitverlauf zu betrachten nicht implementiert.

Zusätzlich war es ein Wunsch der ZGIS, dass das Zeitintervall in der App konfigurierbar sein soll

Der Akkustand der Sendeeinheit soll in der Applikation einsehbar sein

## 5 Ergebnisse – Abnahme

Der aktuelle Projektstand beinhaltet eine funktionstüchtige Platine, welche die GPS-Daten empfängt und diese auf der SD-Karte speichert. Mittels LoRa werden die Standortdaten an den Empfänger übertragen. Über den Empfänger werden die Daten an eine Rest-API gesendet, in eine Datenbank geschrieben und für den Nutzer wieder zur Verfügung gestellt. Die Applikation für das Mobiltelefon verbindet sich zur Rest-API und zeigt die GPS-Daten auf einer Landkarte an.



Abbildung 57: Platine V1 der Sendereinheit



Abbildung 56: Startbildschirm der Smartphone Applikation

In der folgenden Tabelle werden die Ergebnisse der Testfälle, aus dem Lastenheft, veranschaulicht.

Anforderung	Testergebnis	Freigabe (J/N)?
Empfang der Standortdaten (F0010)	Die Verbindung zu einem Satelliten wird hergestellt und die Standortdaten werden ermittelt. (T0010)	J
Speichern auf die SD-Karte (F0020)	Die GPS-Daten werden auf der SD-Karte gespeichert. (T0020)	J
Daten sollen mittels LoRa übertragen werden (F0030)	Die Daten werden von der Sendeeinheit zum Empfänger mittels Lora übermittelt. (T0030)	J
Daten sollen in Datenbank gespeichert werden (F0040)	Die Daten werden in der Datenbank gespeichert und für die App zugänglich gemacht (T0040)	J
Daten sollen über die App abgerufen werden können (F0050)	Die Daten werden von der App abgerufen und graphisch auf einer Landkarte dargestellt. (T0050)	J

## 6 Literaturverzeichnis

---

- [1] „Akku vs Batterie,“ reichelt elektronik , 8 August 2016. [Online]. Available: <https://www.reichelt.de/magazin/ratgeber/akku-vs-batterie-wann-welcher-stromspender-von-vorteil-ist/>. [Zugriff am 2024 03 05].
- [2] „Buck-Boost Converters,“ 15 03 2024. [Online]. Available: <https://www.monolithicpower.com/en/power-electronics/dc-dc-converters/buck-boost-converters>.
- [3] Texas-Instruments, „TPS63001,“ 2023. [Online]. Available: <https://www.ti.com/product/TPS63001?keyMatch=TPS63001&tisearch=search-everything&usecase=GPN-ALT>.
- [4] „Elara-I User Manual,“ 19 März 2024. [Online]. Available: <https://www.we-online.com/components/products/datasheet/2613011037000.pdf>.
- [5] B. Eager, „A tutorial on the FAT file system,“ 17 November 2017. [Online]. Available: <http://www.tavi.co.uk/phobos/fat.html>.
- [6] W. G. Wong, „The Fundamentals Of Flash Memory Storage,“ 20 März 2012. [Online]. Available: <https://www.electronicdesign.com/technologies/embedded/digital-ics/memory/article/21789685/the-fundamentals-of-flash-memory-storage>.
- [7] Kingston, „Was ist Nand?,“ Mai 2021. [Online]. Available: <https://www.kingston.com/de/blog/pc-performance/difference-between-slc-mlc-tlc-3d-nand>.
- [8] „How to Use MMC/SDC,“ 26 Dezember 2019. [Online]. Available: [http://elmcchan.org/docs/mmc/mmc\\_e.html](http://elmcchan.org/docs/mmc/mmc_e.html).
- [9] IONOS, „Was ist FAT32,“ IONOS, 29 November 2021. [Online]. Available: <https://www.ionos.de/digitalguide/server/knowhow/fat32/>. [Zugriff am 1 April 2024].
- [10] „Polycarbonat Gehäuse IP65,“ 25 10 2023. [Online]. Available: <https://at.rs-online.com/web/p/universalgehause/2069086>.
- [11] „IP Schutzklassen,“ 25 03 2024. [Online]. Available: <https://pinlight.eu/e/ip-schutzklassen/>.
- [12] Semtech, „LoRa™ Modulation Basics,“ 2 Mai 2015. [Online]. Available: <https://semtech.my.salesforce.com/sfc/p/E0000000JelG/a/2R0000001OJk/yDEcfAkD9qEz6oG3PJryoHKas3UMsMDa3TFqz1UQOkM>.
- [13] „Design Tips For Incorporating GSM Modules into Embedded Systems,“ 17 März 2024. [Online]. Available: <https://resources.altium.com/p/design-tips-for-incorporating-gsm-modules-into-an-embedded-system>.
- [14] Semtech, „LoRa BasicsN Modem Relay: A Low-Cost Battery Powered Network Extender,“ 8 März 2023. [Online]. Available: <https://info.semtech.com/hubfs/LoRa-Basics%20Modem%20Relay%20A%20Low-Cost%20Battery%20Powered%20Network%20Extender-Whitepaper-F.pdf>. [Zugriff am 16 März 2024].
- [15] Georg-Johann, „Linear-chirp,“ 28 August 2010. [Online]. Available: <https://de.wikipedia.org/wiki/Datei:Linear-chirp.svg>.

- [16] J. Tapparel, „Complete Reverse Engineering of LoRa PHY,“ [Online]. Available: [https://www.epfl.ch/labs/tcl/wp-content/uploads/2020/02/Reverse\\_Eng\\_Report.pdf](https://www.epfl.ch/labs/tcl/wp-content/uploads/2020/02/Reverse_Eng_Report.pdf).
- [17] Z. Xu, S. Tong, P. Xie und J. Wang, „From Demodulation to Decoding: Toward Complete LoRa PHY Understanding and Implementation,“ 31 Jänner 2023. [Online]. Available: <https://dl.acm.org/doi/10.1145/3546869>. [Zugriff am 13 März 2024].
- [18] M. Maxfield, „EETimes,“ 20 12 2006. [Online]. Available: <https://www.eetimes.com/tutorial-linear-feedback-shift-registers-lfsrs-part-1/>. [Zugriff am 2024 März 17].
- [19] Semtech, „Datasheet SX1267-8-9,“ 7 Mai 2020. [Online]. Available: [https://semtech.my.salesforce.com/sfc/p/E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF\\_Fkpgp5kzjiNyiAbqcpqh9qSjE](https://semtech.my.salesforce.com/sfc/p/E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE).
- [20] Elektronik Kompendium, „Wellenwiderstand,“ Elektronik Kompendium, [Online]. Available: <https://www.elektronik-kompendium.de/sites/grd/0301036.htm>. [Zugriff am 15 März 2024].
- [21] Clemen, „Antennen,“ [Online]. Available: [https://www.hs-augsburg.de/~clemen/lehre/Skript\\_Wellen/10Antennen.PDF](https://www.hs-augsburg.de/~clemen/lehre/Skript_Wellen/10Antennen.PDF). [Zugriff am 2024 Februar 28].
- [22] Molex, „ISM 868/915MHz Dipole Flexible Antenna,“ 29 März 2024. [Online]. Available: <https://www.molex.com/en-us/products/part-detail/2067640100?display=pdf>.
- [23] LPRS, „LPRS-ANT-868-DP-N-F,“ 2017. [Online]. Available: <https://lprs.co.uk/assets/files/LPRS-ANT-868-DP-N-F.pdf>.
- [24] LILYGO, „T-Beam Meshtastic,“ [Online]. Available: <https://www.lilygo.cc/products/t-beam-v1-1-esp32-lora-module>.
- [25] PlatformIO, „What is PlatformIO?,“ 6 Februar 2023. [Online]. Available: <https://docs.platformio.org/en/latest/what-is-platformio.html>.
- [26] sandeepmistry, „Arduino LoRa,“ Github, [Online]. Available: <https://github.com/sandeepmistry/arduino-LoRa>. [Zugriff am 16 März 2024].
- [27] „Laravel,“ 17 März 2024. [Online]. Available: <https://laravel.com/>.
- [28] jaydeepsathwara, „Introduction into Laravel and MVC Framework,“ 20 März 2024. [Online]. Available: <https://www.geeksforgeeks.org/introduction-to-laravel-and-mvc-framework/>.
- [29] Laravel, „Eloquent: Getting Started,“ [Online]. Available: <https://laravel.com/docs/10.x/eloquent>.
- [30] Laravel, „Laravel Sanctum,“ [Online]. Available: <https://laravel.com/docs/10.x/sanctum>.
- [31] Laravel, „Routing,“ [Online]. Available: <https://laravel.com/docs/11.x/routing>.
- [32] „pub.dev,“ Google, 2024. [Online]. Available: <https://pub.dev/>. [Zugriff am 20 03 2024].
- [33] „flutter\_map,“ 2024. [Online]. Available: [https://pub.dev/packages/flutter\\_map](https://pub.dev/packages/flutter_map). [Zugriff am 13 02 2024].
- [34] „latlong2,“ 2024. [Online]. Available: <https://pub.dev/packages/latlong2>. [Zugriff am 14 02 2024].
- [35] „http,“ 2024. [Online]. Available: <https://pub.dev/packages/http>. [Zugriff am 10 03 2024].
- [36] „Flutter,“ 2024. [Online]. Available: <https://flutter.dev/>. [Zugriff am 23 03 2024].

- [37] „Recom,“ 2024. [Online]. Available: [https://recom-power.com/de/rec-n-an-introduction-to-buck,-boost,-and-buck!sboost-converters-131.html?0#:~:text=Ein%20Buck%20oder%20auch%20Abw%C3%A4rtswandler,niedrigere%20Ausgangsspannung%20herunterregelt%20\(reduziert\)..](https://recom-power.com/de/rec-n-an-introduction-to-buck,-boost,-and-buck!sboost-converters-131.html?0#:~:text=Ein%20Buck%20oder%20auch%20Abw%C3%A4rtswandler,niedrigere%20Ausgangsspannung%20herunterregelt%20(reduziert)..) [Zugriff am 20 3 2024].
- [38] „Abwärtswandler,“ [Online]. Available: [http://schmidt-walter-schaltnetzteile.de/smPS/abw\\_hilfe.html](http://schmidt-walter-schaltnetzteile.de/smPS/abw_hilfe.html). [Zugriff am 21 03 2024].
- [39] „Aufwärtswandler,“ [Online]. Available: [http://schmidt-walter-schaltnetzteile.de/smPS/aww\\_hilfe.html](http://schmidt-walter-schaltnetzteile.de/smPS/aww_hilfe.html). [Zugriff am 21 03 2024].
- [40] „ATmega644 PA Datasheet,“ 24 März 2024. [Online]. Available: <https://www.mouser.at/ProductDetail/Microchip-Technology/ATMEGA644PA-AU?qs=jWm4idrLpzylZp8jH09Tlw%3D%3D>.
- [41] „Amphenol Luftöffnung,“ 7 März 2024. [Online]. Available: <https://at.rs-online.com/web/p/gehausebeluftung/1749423>.

## 7 Abkürzungen

---

API.....	<i>application programming interface</i>
ASCII .....	American Standard Code for Information Interchange
CSV.....	Comma Separated Values
ERP.....	<i>effective radiated power</i>
FAT.....	<i>File Allocation Table</i>
GPS.....	<i>Global Positioning System</i>
LoRa.....	<i>Long Range</i>
ORM.....	<i>Object Relational Mapper</i>
OSP.....	<i>One Socket Protocol</i>
PHP.....	<i>PHP: Hypertext Preprocessor</i>
RSSI.....	<i>Received Signal Strength Indicator</i>
SD-Karte.....	<i>Sichere Digitale Karte</i>
SLC.....	<i>Single Level Cell</i>
SPI.....	<i>Serial Peripheral Interface</i>
TLC.....	<i>Triple Level Cells</i>
UART.....	Universal Asynchronous Receiver Transmitter

## 8 Abbildungen

---

Abbildung 1: Projekt Logo.....	4
Abbildung 2: Project Logo.....	6
Abbildung 3: Blockschaltbild des Lokalisierungssystems .....	11
Abbildung 4: GANT-Diagramm Putz.....	21
Abbildung 5: GANTT-Diagramm König.....	22
Abbildung 6: Blockdiagramm des C-Track Systems.....	23
Abbildung 7: Buck Converter Schaltung .....	25
Abbildung 8: Strom und Spannungsverlauf des Buck Converters [38] .....	25
Abbildung 9: Boost Converter.....	27
Abbildung 10: Strom und Spannungsverlauf des Boost Converters .....	27
Abbildung 11: Buck - Boost - Converter .....	29
Abbildung 12: Typische Applikation des TPS 63001 [3] .....	30
Abbildung 13: Messung der Ausgangripples an der Platine .....	30
Abbildung 14: Messungen des Datenblattes [3] .....	30
Abbildung 15: Elara-I Modul [4] .....	31
Abbildung 16: Pinout Elara-I Modul .....	31
Abbildung 17: Aufbau einer NMEA-Satz [4].....	32
Abbildung 18: Aufbau eines OSP-Satz [4].....	32
Abbildung 19: NMEA-Sentence mit RMC-Format.....	33
Abbildung 20: RMC-Format [4] .....	33
Abbildung 21: Trickle Power mode [4] .....	34
Abbildung 22: Befehle für die Kommunikation mit SD/MMC-Karten (vgl. [8]) .....	37
Abbildung 23: R1-Response [8] .....	37
Abbildung 24: typische Struktur für Verwendung der FatFS-Bibliothek [8] .....	39
Abbildung 25: Tabelle der IP- Schutzklassen [11] .....	40
Abbildung 26: Programmablauf der Sendeeinheit .....	41
Abbildung 27: Chirp Signal im Zeitbereich [15].....	44
Abbildung 28. Frequenzverlauf eines LoRa-Signals [16].....	45
Abbildung 29: Encoding/Decoding von LoRa [16] .....	46
Abbildung 30: Interleaving (vgl. [17]) .....	47
Abbildung 31: LoRa Paket [19] .....	47
Abbildung 32: Richtdiagramm Molex Dipol [22] .....	48
Abbildung 33: Richtdiagramm Antenne Basisstation [23] .....	49
Abbildung 34: Serieller Monitor Basisstation .....	52
Abbildung 35: Datenbankstruktur .....	54
Abbildung 36: Überblick über Routen des Servers .....	57
Abbildung 37: dependencies in pubspec.yaml.....	59
Abbildung 38: Code für Launcher Icon .....	59
Abbildung 39: Code: Debug Banner deaktivieren.....	60
Abbildung 40:Code: Text in App Bar .....	60
Abbildung 41: Fertige AppBar.....	60
Abbildung 42: Code: Drop Down Menu Header.....	61
Abbildung 43: Code: Button einfügen .....	61
Abbildung 44: Drop Down Menu.....	61

---

Abbildung 45: Code: txt-File einlesen .....	62
Abbildung 46: Code: initState .....	62
Abbildung 47: About Us Page.....	62
Abbildung 48: Code: Flutter Map .....	63
Abbildung 49: Code: Tile Layer .....	63
Abbildung 50: Code: Marker Layer .....	63
Abbildung 51: Code: Serververbindung .....	64
Abbildung 52. Code: Serververbindung .....	64
Abbildung 53: Code: Marker Position .....	64
Abbildung 54: Code: initState .....	64
Abbildung 55: Startbildschirm der Smartphone Applikation.....	65
Abbildung 56:Startbildschirm der Smartphone Applikation.....	67
Abbildung 57: Platine V1 der Sendereinheit .....	67
Abbildung 58: Schaltplan V1.....	77
Abbildung 59: Board V1.....	78
Abbildung 60: Schaltplan V2.....	79
Abbildung 61: Board V2.....	80

## 9 Begleitprotokoll gemäß § 9 Abs. 2 PrO

---

### 9.1 Begleitprotokoll Samuel Putz

Name: Hr. Samuel Putz

Diplomarbeitstitel: C-Track 2.0

KW	Beschreibung	Zeitaufwand
38	Betreuersuche und Besprechung/Planung des Projekts	10h
39	Einarbeiten in altes Projekt	10h
40	Besprechung mit Dr. Gudrun Wallentin	5h
41	Diplomarbeitsantrag erstellen & absenden	6h
42	Einarbeiten & testen des alten Projektes	5h
43	Schaltplan	10h
44	Board-Design	5h
45	Herbstferien	
46	Board-Design & Auswahl der Bauteile; Projektreview	10h
47	Fertigstellung des Boards	10h
48	Krankenstand Projektreview	10h
49	Löten der Platine und erste Testungen	10h
50	Testen des GPS-Moduls und Testen der LoRa Verbindung mit neuen Antennen	10h
51	Aufsetzen und einlernen in Flutter	10h
52	Weihnachtsferien	
1	Weihnachtsferien	
2	Einlernen in Flutter	10h
3	Projektpräsentation ; Projektreview ; Tag der offenen Tür	10h
4	Einlernen in Flutter	10h
5	Präsentation	10h
6	Platine V2	5h
7	Semesterferien	
8	Diplomarbeit schreiben	5h
9	Diplomarbeit schreiben	5h
10	Programmierung der HandyApp	10h
11	Programmierung der HandyApp	10h
12	Testen des Projekts / schreiben der Diplomarbeit	10h
13	Osterferien	

KW ...Kalenderwoche

## 9.2 Begleitprotokoll Simon König

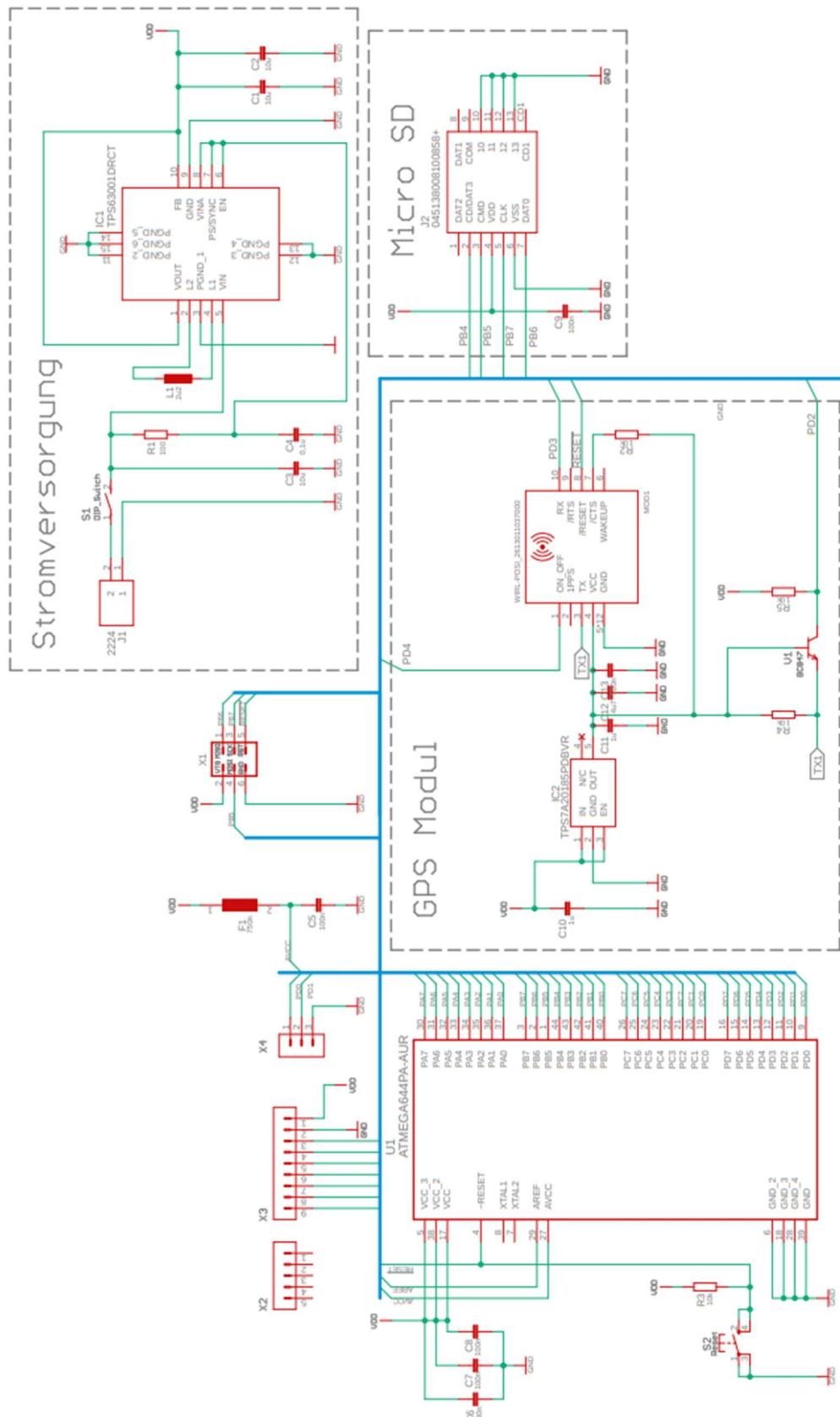
**Name:** Hr. Simon König

**Diplomarbeitstitel:** C-Track 2.0

KW	Beschreibung	Zeitaufwand
38	Betreuersuche und Besprechung/Planung des Projekts	10h
39	Einarbeiten in altes Projekt	10h
40	Besprechung mit Dr. Gudrun Wallentin	5h
41	Diplomarbeitsantrag erstellen & absenden	6h
42	Einarbeiten & testen des alten Projektes	5h
43	Datenbank	10h
44	Datenbank	5h
45	Herbstferien	
46	Auswahl der Bauteile ; Projektreview	10h
47	Rest API	10h
48	Rest API ; Projektreview	10h
49	Löten der Platine und erste Testungen	10h
50	Testen des GPS-Moduls und Testen der LoRa Verbindung mit neuen Antennen	10h
51	Aufsetzen und einlernen in Flutter	10h
52	Weihnachtsferien	
1	Weihnachtsferien	
2	Low Powermode Konfiguration	10h
3	Projektpräsentation ; Projektreview ; Tag der offenen Tür	10h
4	GPS-Trickle Powermode	10h
5	GPS-Trickle Powermode	10h
6	Lora	5h
7	Semesterferien	
8	Lora	5h
9	Diplomarbeit schreiben	5h
10	Diplomarbeit schreiben	10h
11	Optimieren der Sendeeinheit	10h
12	Testen des Projekts / Diplomarbeit schreiben	10h
13	Osterferien	

**KW ...Kalenderwoch**

## 10 Anhang



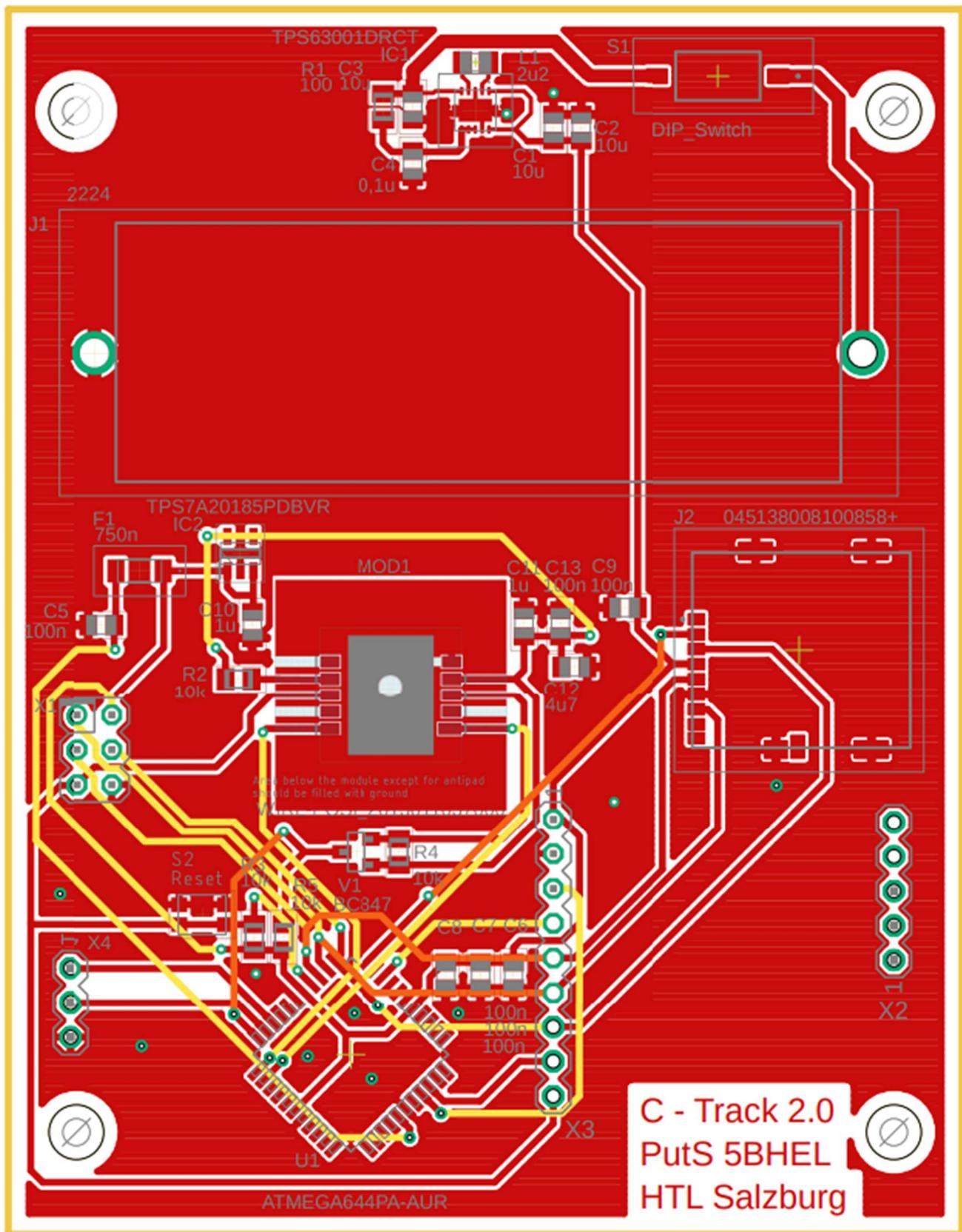
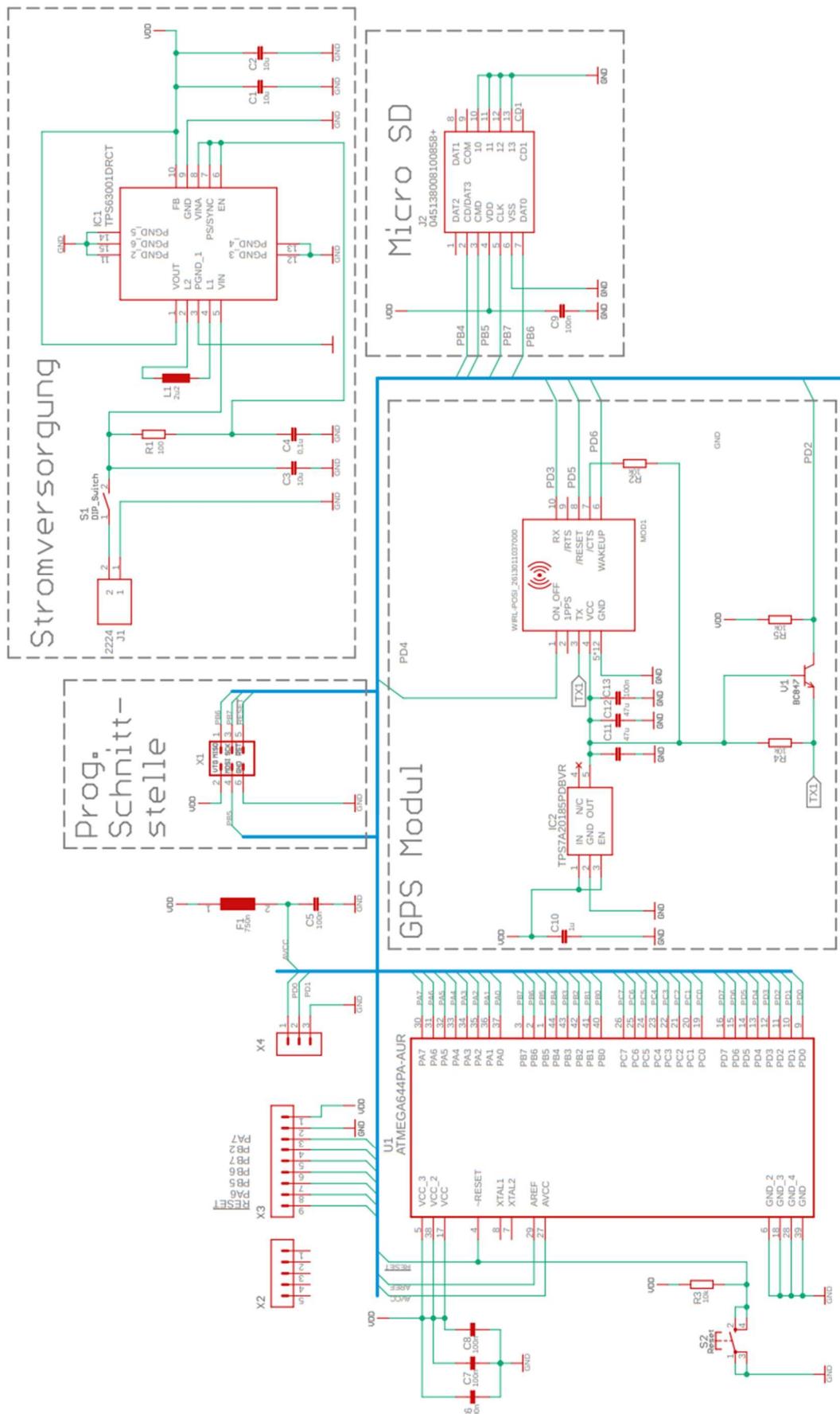


Abbildung 59: Board V1



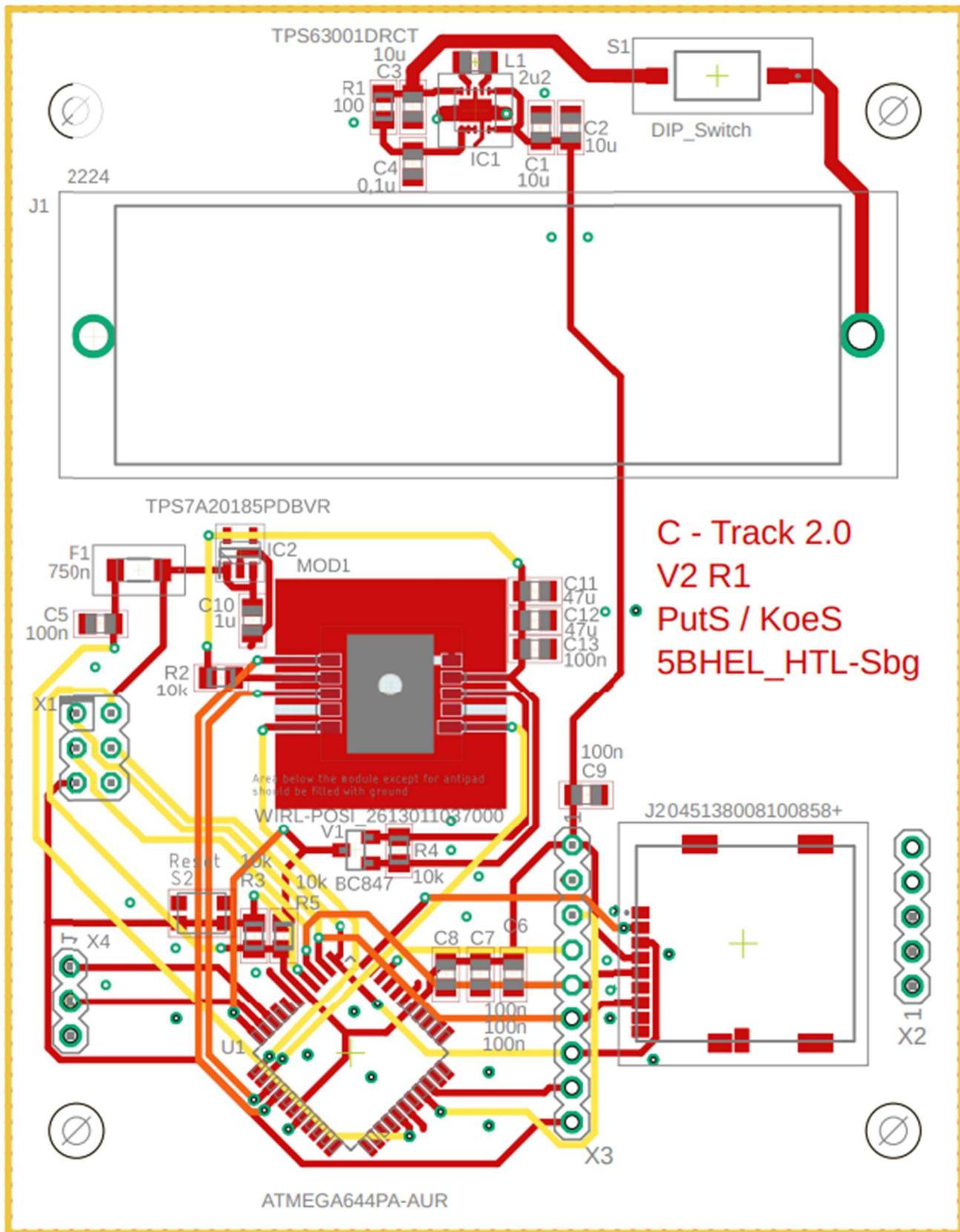


Abbildung 61: Board V2